Administrator's and Developer's Guide

NetPhantom

Version 7.6

Document Revision 1

4 October 2023

Mindus SARL

NetPhantom®

Version 7.6

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 System Overview 3
- 1.2 NetPhantom Editor 3
  - Eclipse IDE Integration 4
- 1.3 NetPhantom Server 4
- 1.4 NetPhantom Client 4
  - Swing – The Java Foundation Classes 4
  - Server Choice 4
  - Java Client API 4
- 1.5 Load Balancing 4
- 1.6 Communication between Client and Server 5

2 **System Requirements** **7**
- 2.1 Hardware Platform 7
  - The CPU 7
  - Physical RAM 7
  - Disk Space 7
- 2.2 Operating Environment 8
  - Operating System 8
  - Java Environment 8
- 2.3 Network 8
- 2.4 Client System Requirements 8

3 **NetPhantom License System** **9**
- 3.1 Introduction 9
- 3.2 License Management Made Simple 9
- 3.3 Server Types and States 10
  - Backup Server vs. Primary Server 10
  - Connecting State 10
  - Active State 10
  - Non-active State 10
  - Non-connected State 10
  - What Happens When Servers Change State 10
- 3.4 Running the License Manager 11
  - Running the License Manager on a Different Machine 11
- 3.5 Using NetPhantom License Manager as a Windows Service 11
- 3.6 Configuring the License Manager 11
  - The “license.ini” File 11
- 3.7 Configuring the License Manager from Server Administration 12
  - The License System Dialog 12
  - License Manager Configuration 14
- 3.8 Distribution of Licenses by the License Manager System 18
  - Server Connections to the License Manager System 18
  - License Distribution 19
    - Specific License Distribution 20
- 3.9 Test License Code 20

4 **Getting Started** **21**
- 4.1 Installation 21
  - Installation using the SETUP.EXE 21
  - Installation using an Installation Image (JAR) 21
- 4.2 Startup Scenario 22
  - Start the NetPhantom Server 22
    - Initial Server Configuration 22
    - Server Log Output 22
    - Start a NetPhantom Client or a Browser 22
  - Removing Initial User Authentication for Server Administration 24
- 4.3 NetPhantom License System 24

4.4 Directory Structure ... 24
4.5 Starting the Server ... 26
Windows ... 27
UNIX ... 27
Executing the Server Start Command ... 27
Return Codes ... 27
Batch Files for Server Execution ... 28
4.6 Using NetPhantom Server as a Windows Service ... 29
The Event Viewer ... 30
4.7 NetPhantom Windows Services ... 31
Installing a Windows Service ... 31
Uninstallation of a Windows Service ... 32
Registry Keys for a Windows Service ... 33
HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\NetPhantomServer: ... 33
HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\NetPhantomServer\Parameters: ... 33
HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\Eventlog\Application\NetPhantom Server: ... 33
4.8 Event Logging to the Windows Event log ... 34
4.9 Linux Service/Deamon ... 34
4.10 Linux Port configuration ... 35
4.11 Software Updates ... 35

5 Running the NetPhantom Client ... 37
5.1 NetPhantom Client Parameters ... 37
Port/Host and Backup Servers ... 40
Specifying Runtime Application IDs ... 41
User Variables ... 41
5.2 Standalone Application ... 42
5.3 Inside a Browser ... 42
5.4 NetPhantom Client with SmartApplet ... 42
The SmartApplet Definition File – SmartApplet.ini ... 42
5.5 The draggable applet ... 43
5.6 The Application INI file ... 44
Client Properties for an Application ... 44
Bitmap in Application Area ... 45
Background Color in Application Area ... 45
Tooltip Text ... 46
Hiding Unauthorized Menu Items in NetPhantom Session Booster ... 47
5.7 Using Dynamic Data Exchange – DDE ... 48
Running NetPhantom Client with DDE as a Java Application ... 48

6 Running the NetPhantom Editor ... 49
6.1 Overview ... 49
Building ... 49
Eclipse integration ... 50
Execution ... 50
Distribution compilation ... 50
Java Runtime Environment ... 50
6.2 Terminal Editor ... 52
6.3 Panel Editor ... 53
Panel setting – Old and New style ... 53
6.4 Integrated Server ... 54
6.5 Batch Build ... 54
Requirements ... 55
Performing a Batch Build ... 55
Return Codes ... 55

7 Configuring the Server ... 57
Automatic Start of Server Configuration ... 57

7.1 Base Section ... 57
7.2 User Exit Print Section ... 60
7.3 Port Section ... 60
7.4 Country Settings Section ... 61
7.5 Event Filter Section ... 62
7.6 Trace Section ... 62
7.7 Font Substitution on Client Section ... 62
Definition of a Font Substitution List ... 62
Definition of a Client Specific Font Substitution List ... 63
Remapping a Font ... 63
The NetPhantom System VIO font ... 64
Fixed Font Metrics ... 64
Example of Font Sections ... 65
7.8 Runtime Application Section ... 65
7.9 Host Section ... 66
Host Session Definitions for 3270 and 5250 ... 66
Terminal Types ... 67
Supported 3270 Terminal Types ... 67
Supported 5250 Terminal Types ... 67
3270 Numeric Field Override Option ... 68
Custom Host Datastream Exit ... 68
The EE package ... 68
Host Datastream Problem Determination Aid ... 68
Other Host Settings ... 68
7.10 Web Server Section ... 69
7.11 The Servlet/CGI Section ... 71
7.12 The Web Applications Section ... 71
7.13 Remote Applications Section ... 72
7.14 Event Messenger Section ... 73
7.15 SSL Settings Section ... 73
7.16 Session Pooling Data ... 75
7.17 Load Balancing Section ... 75
7.18 Cluster Controller Section ... 75
7.19 Setting up Web Server Files for NetPhantom ... 76

8 NetPhantom – Connectivity, Internet, and Security ... 79
Firewalls ... 79
Encryption ... 79
Java Applet Limitation ... 79
8.1 NetPhantom Topologies ... 79
NetPhantom Intranet and Extranet Topology ... 80
NetPhantom Internet Topology ... 80
8.2 HTTP Tunneling and the NetPhantom Web Server ... 81
HTML Document for HTTP Tunneling ... 81
8.3 SSL – Secure Socket Layer ... 82
HTML Document for SSL ... 82

9 NetPhantom Web Server ... 83
9.1 Hypertext Transfer Protocol – HTTP ... 83
9.2 NetPhantom Web Server ... 83
9.3 NetPhantom and HTML ... 84
9.4 Servlets or CGIs ... 84
9.5 Introduction to User Authentication ... 84
Purpose of NetPhantom Web Server Authentication ... 84
HTTP Authentication ... 84
9.6 Using NetPhantom Web Server Authentication ... 85
9.7 NetPhantom Redirection Service ... 85
9.8 NetPhantom Web Server – Technical ... 86
Introduction to the HTTP Authentication Schemes ... 86
Basic Authentication Scheme ... 86

Auth-params for Basic Authentication Scheme ... 87
Digest Authentication Scheme ... 87
Auth-params for Digest Authentication Scheme ... 87
Example of Digest Authentication Messages ... 88

10 **NetPhantom Load Balancing ... 89**
10.1 Introduction ... 89
10.2 Load Balancing Techniques ... 89
10.3 Load Balancing Redirection ... 90
10.4 Slave/Controller Server Communication ... 90
Load Balancing Qualification ... 91
Controller Information to Slave ... 91
Slave Information to Controller ... 91

11 **NetPhantom Cluster Controller ... 93**
11.1 Executing the Cluster Controller ... 93
Memory Requirements ... 94
Licensing – Number of Users ... 94
Sample Batch File for Windows ... 94
Sample Script File for UNIX ... 95
11.2 Running the Cluster Controller as a Windows Service ... 95
11.3 Special Server Functions with Cluster Controller ... 95
Server Shutdown ... 95
Server Restart ... 95
Server Upgrade ... 96
Steps to perform a Server Upgrade ... 96
Internal versus External Upgrade ... 96
11.4 Configuring the Cluster Controller ... 96
Event Log and Trace Files ... 98
Manual Configuration in server.ini ... 99
Example of Manual Configuration in server.ini ... 100

12 **NetPhantom SecureLogin ... 101**
12.1 GSM Phone Network – SMS Messaging ... 101
12.2 Configuring NetPhantom SecureLogin ... 102
Configuring a CGI ... 102
Configuring a Resource ... 102
12.3 Configuration of Securelogin.ini File ... 103

13 **SSL – Secure Socket Layer ... 105**
13.1 Introduction to Cryptography ... 105
Cryptography and NetPhantom ... 105
Basic Terminology ... 105
Basic Cryptosystems ... 105
Digital Signatures ... 106
Cryptographic Hash Functions ... 106
Digital Certificates ... 106
13.2 The SSL System ... 106
Overview ... 106
How Does SSL Work? ... 107
What Does SSL Provide? ... 107
What is TLS? ... 107
13.3 NetPhantom's Cryptographic Module ... 107
What Does NetPhantom Implement? ... 107
Cipher Suites ... 107
Cipher Suite Strength ... 108
Cipher Suite Key Exchange ... 108
Choosing a Cipher Suite ... 109
CA Certificates ... 110
Personal Certificates ... 110
Identity Keystores ... 110

Certificate Choice ... 111
13.4 NetPhantom SSL Configuration ... 111
Certificate Creation ... 111
Server Configuration ... 112
Cipher Suites ... 113
CA Certificate Files ... 113
Identities ... 114
Parameters ... 114
Maximum Use of Temporary Keys ... 115
Maximum Lifetime of Temporary Keys ... 115
Session Cache Capacity ... 115
Session Cache Timeout ... 116
Disallow TLSv1 Support ... 116
ServerSocket Implementor ... 116
13.5 Creating an Identity ... 116
13.6 Creating a Self-Signed Certificate ... 117
13.7 Working with Authorized or Denied Client Certificates ... 117
Authorized Certificates ... 117
Revoked Individual Certificates ... 118
Revoking Certificates using CRLs ... 118
13.8 Viewing and Refreshing Authorized or Revoked Certificates ... 118

14 Client Certificates ... 121
14.1 Introduction ... 121
14.2 Technology Overview ... 121
14.3 Potential Uses ... 121
User Authentication ... 121
Access Control ... 122
Password Replacement ... 122
14.4 Configuration ... 122
Creating Client Certificates ... 122
14.5 Client Certificates and Certificate Revocation ... 123
Installing Client Certificates on the Server ... 123
Installing Revoked Client Certificates on the Server ... 124
Installing Client Revocation List (CRL) Files ... 124
Installing the Client Certificates on Client ... 125
Browser ... 125
SSL Files Directory ... 126
Client Certificates ... 126
User-accepted CA Certificates ... 126
Prompting for Client Certificate Passphrase ... 126
Example of Client Certificate Request and Installation for a Client ... 126
Running NetPhantom Client as Application with Client Certificate ... 130
14.6 Restricting an Application Resource ... 131

15 NetPhantom Starter ... 133
15.1 Overview ... 133
15.2 NetPhantom Starter ... 133
Web Server Authentication ... 133
Web Server Redirection ... 134
15.3 NetPhantom Starter with SSL ... 134
Accepted Server Certificates ... 134
Client Certificate Password ... 134
15.4 Installation of the NetPhantom Starter on the Client ... 134
Changing the Behavior of NetPhantom Starter on the Client ... 136
15.5 Installing Multiple NetPhantom Starters on the Client ... 136
15.6 Installation of the NetPhantom Starter on the Server ... 137
15.7 Package Definition File ... 137
15.8 Application Chooser Definition File ... 138
Editing the Application Chooser Definition File ... 138

15.9 Switches for PackageFileMaker ... 139
Client Files (-f) ... 139
The List File ... 140
Class (-c) ... 140
Arguments (-a) ... 140
Classpath (-p) ... 141

Common Server Administration Functions ... 143
15.10 Backup Server(s) ... 143
15.11 Server Events ... 143
15.12 Event Filtering ... 144
15.13 Tracing ... 144
Example of Binary Trace Output ... 145
Example of Verbose Trace Output ... 145
15.14 Host LU Mapping ... 146
TCP/IP to Host LU Name Mapping ... 146
15.15 User Authentication ... 146
Enabling User Authentication for an Application ... 146
Client User Interface for Logon ... 147
15.16 National Language Support ... 147
Changing the Language ... 147

16 **The Server Administration Program ... 149**
16.1 License Configuration ... 150
16.2 File Transfer ... 150
Security ... 150
File Manager ... 151
File Transfer from Client to Server ... 152
File Transfer from Server to Client ... 152
16.3 Server Configuration ... 153
General Settings ... 153
Country Settings ... 154
Event Filter Settings ... 155
Trace Settings ... 156
Font Substitution ... 157
Remapping the Font ... 160
Applications ... 161
Host Settings ... 163
Host Client Settings ... 167
3287 Host Printer Settings ... 170
Client Settings ... 171
String Caching to Increase Performance ... 171
Configuration of String Caching ... 171
16.4 Port Configuration ... 173
16.5 Configuring the Web Server. ... 174
IP Address ... 176
Load Balancing ... 177
Defining a Common Gateway Interface ... 178
Defining a Web Application ... 178
Remote Apps ... 180
Specifying Domains ... 181
Secure Login ... 183
Access Control ... 184
Resources ... 185
16.6 Proxy Configuration ... 187
16.7 SSL Configuration ... 188
16.8 Creating Client Certificates ... 188
16.9 Event Messengers ... 188
16.10 Managing Groups ... 189
16.11 Managing Users ... 190

Password Expiration Interval 190
Creating a New User 190
Changing or Copying a User Definition 192
SecureLogin Definition for a User 192
16.12 Server Memory Usage 193
16.13 Thread Usage 194
16.14 Server Statistics 195
Host Statistics 196
Client Statistics 196
Client Ping 197
Web Server 198
Load Balancing 199
16.15 Server Shutdown 199
16.16 Server Restart 199
Soft Restart 200
Hard Restart 200
Hard Java VM Restart 200
16.17 Upgrade Server 201
16.18 Upgrade Editor and Server 202
16.19 Server Upgrade Log 202
16.20 Reload TCPIP/LU Mapper 202
16.21 Enabling, Disabling or Reloading Applications 203
16.22 Publish Applications 203
16.23 Client Connections 208
16.24 Broadcast Message 209
16.25 New Client Connections 210
16.26 Display Terminal Session 210
16.27 Keyboard Remapping 211
Note Concerning the Ctrl Key 212
16.28 Terminal Window Colors 212
16.29 Get Client Font Metrics 213
16.30 Event Log 213
16.31 Trace Settings 214
16.32 Event Filtering 215
16.33 Trace & Event Properties 215
16.34 Toolbox API 216

17 The Remote Command Line Utilities 219
17.1 Concurrent User Count 219
17.2 Remote Server Administration 220
NOP Command 221
SHUTDOWN Command 221
MEMORY Command 221
BROADCAST Command 221
APPS Command 222
CLIENTS Command 222
DISABLE Command 222
ENABLE Command 222
LOCK Command 222
UNLOCK Command 222
PUT/PUTR Commands 223
KILL Command 223
RESTART Command 223
RESTARTWS Command 223
UPGRADE Command 224

18 Remote Assistance 225
18.1 VNC Server Settings 226

19 The Telnet Tracer 231
19.1 Telnet Tracer 231

Start Command ... 231
Installation ... 231
19.2 Telnet Trace Files ... 232
Data Format ... 232
Transaction header ... 232
Transaction body ... 232
Logic Variables ... 232
Usage ... 232
Variables ... 232
Suggested Usage ... 233
19.3 Telnet Playback ... 233
Start Command ... 233
Installation ... 233

20 **The NetPhantom Stress Tools ... 235**
20.1 Stress Tool ... 235
Options ... 235
ClientAmount ... 235
ClientArgs ... 235
Usage ... 235
20.2 StressNoGUI Tool ... 235
Options ... 235
Usage ... 236

21 **Overview of the Client Architecture ... 237**
21.1 Schematic Client Structure ... 237
21.2 Component Model ... 237
Component Types ... 238

22 **Overview of the Server Architecture ... 239**
22.1 Schematic Server Overview ... 239
22.2 Mainframe Host Communication ... 240
22.3 Server Components ... 241
22.4 Relationship between Server Components ... 241
Virtual Session Manager/Client Session ... 241
Virtual Session Manager/Runtime Application Data ... 241
Virtual Session Manager/Virtual Message Box ... 241
Virtual Session Manager/API ... 241
Virtual Session Manager/Host Session Manager ... 241
Virtual Session Manager/Virtual Session ... 242
API/Runtime Application Data ... 242
API/Host Session Manager ... 242
Virtual Session/Host Session Manager ... 242
Virtual Session/Virtual Panel ... 242
Virtual Panel/Host Session Manager ... 242
Virtual Panel/Host Session ... 242
Virtual Panel/Virtual Message Box ... 242
Virtual Panel/Virtual Components ... 242
Virtual Components/Runtime Panel ... 242
Virtual Components/Host Session ... 242
Runtime Application Data/Runtime Panel ... 242

23 **Building an Application ... 243**
23.1 NetPhantom Project ... 243
23.2 Application References ... 249
23.3 Compile distribution in NetPhantom Editor ... 249
23.4 Import Screens ... 249
23.5 Import Existing Project ... 250
23.6 Import Current Project in Eclipse Workspace ... 250
23.7 Unsupported REXX Functions ... 250
23.8 Troubleshooting REXX Code ... 250

REXX Source Sample 251
NetRexx Converted REXX Source Sample 251

24 **NetPhantom HTML Integration 253**
24.1 Technology Overview 253
24.2 Design Potential 253
24.3 Limitations 253
24.4 Overview 253
NetPhantom HTML Integration 253
NetPhantom Editor 254
Restrictions for NetPhantom HTML Integration 254
Session Cookies 254
Browser Buttons "Back" and "Forward" 255
24.5 The Web Application 255
Tutorial Samples in HTML 258
24.6 Step-by-Step Quick Start 258
Location of Files 259
Web File Resource Redirection 260
24.7 NetPhantom Markup Tags in HTML 261
Supported HTML elements 261
Invalid HTML Document 261
Server Side Include – SSI 261
HTML-Included Servlets or CGIs 262
24.8 Variable Substitution in HTML Text 262
24.9 Variable Substitution in HTML Tags 263
24.10 Variable Substitution in Scripts 263
24.11 Variables in HTML Documents 263
General Variables 263
HTTP Related Variables 264
NetPhantom Application Variables 266
24.12 NetPhantom Web Application 267
Normal HTML Text 267
NetPhantom Variable Substitution 268
24.13 Character Set 268
24.14 The FORM Element 269
24.15 Focus in HTML Forms 269
24.16 The SCRIPT Element 270
24.17 Configuration Reference 270
CGI Settings 271
Web Application Settings 271
Resource Settings 273
24.18 Control Reference 274
HTML to NetPhantom References 274
Description of Terms Used in this Section 275
The Controls 275
Push Button – Input Submit 276
Entry Field – Input Text 277
Multiple Line Entry Field – Textarea 277
Checkbox – Input Checkbox 278
Radio Button – Input Radio 278
Combination Box – Select 279
Listbox – Table 280
Hyperlink 283
Change Current Frame 285
24.19 Various Functions 286
Browser Capabilities 286
Disabled Attribute for User Input Elements 287
Using Label Tag for Hidden Controls 288
Message Box HTML File 288
General Limitations 289

REXX Application Limitations ... 289
REXX Object Events ... 289
24.20 NetPhantom Copy HTML ... 290
COPYHTML.INI File ... 290
Control Specific Added Attributes ... 292
Control Conversion ... 292
Static Text ... 292
Output Text ... 292
Entry Field ... 292
MLE ... 293
Combination Box ... 293
Spin Button ... 294
Push Button ... 294
Checkbox ... 295
Radio Button ... 295
List Box ... 296
Group Box ... 297
Menu Item ... 297
24.21 HTML Application Errors ... 297
24.22 Frequently Asked Questions ... 298

25 **The NetPhantom Terminal Window ... 299**
25.1 Terminal Properties ... 299
Sample properties file contents ... 300
25.2 Changing Terminal Session ... 300
25.3 Configuring the Terminal Window ... 301
25.4 Manual Configuration of the Terminal Window ... 302
25.5 Special Requirements for *terminal.jar* ... 303
25.6 Default Implementation in TerminalApplication ... 303
25.7 Extending the TerminalApplication ... 304
Special requirements for `terminal.jar` ... 304
Default implementation in `TerminalApplication` ... 305

26 **The NetPhantom Mail Utility ... 307**
26.1 The MailForm CGI ... 307
Configure the MailForm CGI ... 307
Configure a Resource ... 307
MailForm Tags ... 307
26.2 Alphabetical List of MailForm Tags ... 309
email_bcc ... 309
email_cc ... 309
email_doc ... 309
email_error_doc ... 310
email_field_separator ... 310
email_from ... 310
email_host ... 311
email_output ... 311
email_server ... 311
email_subject ... 312
email_success_doc ... 312
email_to ... 312
email_use_ssl ... 312
email_port ... 313
email_auth_user ... 313
email_auth_password ... 313
26.3 MailForm Example ... 314
The Form Definition ... 316
The Tag Definitions ... 316
Fields for User Information ... 316
Input Field for Resetting the Form ... 316

Input Field for Sending the Form ... 316
26.4 Formatting the Mail ... 316
26.5 Formatting with the Use of a Template ... 317
Template Command Tags ... 317
26.6 Alphabetical List of Template Command Tags ... 317
set_field ... 317
unused_tag ... 317
26.7 MailForm Template Example ... 318
26.8 The NetPhantom REXX API for Mail ... 318
Return Values ... 319
SendEMail Commands ... 319
26.9 Alphabetical List of REXX SendEMail API Commands ... 319
GetLastError ... 319
Initialize ... 320
Send ... 320
SetAuthPassword ... 320
SetAuthUser ... 320
SetBCC ... 320
SetCC ... 321
SetFromEmail ... 321
SetPasswordUserID ... 321
SetReplyEmail ... 321
SetHost ... 322
SetContentType ... 322
SetMsgBody ... 322
SetPort ... 322
SetServer ... 323
SetSSL ... 323
SetSubject ... 323
SetToEmail ... 323
26.10 The NetPhantom Java API for Mail ... 323
26.11 The NetPhantom Standalone Mail Utility ... 324
TO Parameter ... 324
CC Parameter ... 324
BCC Parameter ... 324
FROM Parameter ... 324
REPLY Parameter ... 324
HOST parameter ... 324
SERVER Parameter ... 324
PORT Parameter ... 324
SSL Parameter ... 325
USERID Parameter ... 325
PASSWORD Parameter ... 325
SUBJECT Parameter ... 325
MESSAGE Parameter ... 325
TYPE Parameter ... 325
FILE Parameter ... 325
NLSMESSAGE ... 325
DEBUG ... 325

27 The NetPhantom REXX API for SMS ... 327
27.1 Return Values ... 327
27.2 SendSMS Commands ... 327
27.3 Alphabetical List of REXX SendSMS API Commands ... 328
Initialize ... 328
SendMessage ... 328
27.4 The NetPhantom Standalone SMS Utility ... 328
ApplicationId Parameter ... 329
Server Parameter ... 329
Port Parameter ... 329

Key File Name Parameter 329
Phone Number(s) Parameter 329
Message Parameter 329

The NetPhantom OneLogin 331
27.5 Requirements 331
Extra Classes 331
27.6 Installation 331
The Logon Host Screen 331
The Logon Panel 331
The DLL Object 331
27.7 Functions 332
Add a Client Manually 332
Password Change 332
27.8 What Happens Inside 332
27.9 REXX API Methods 333
Change password 333
Add user 334

28 **Session Pooling 335**
28.1 Session Pool Actions 335
28.2 Server Shutdown or Restart 336
28.3 Configuration of Session 336
28.4 Session Pool Scripting 336
File Format 337
Action Tags 337
Script Tags 338
Tag – SCREEN 338
Tag – CONDITIONS 339
Tag – IF 339
Tag – WHILE 340
Tag – BREAK 340
Tag – SET 340
Tag – SEND 341
Tag – RESET 341
Tag – WAIT 342
Tag – HOSTERROR 342
Tag – ONERROR 343
Tag – LOG 343
Tag – TRACE 343
Tag – RETURN 343
Tag – DISPOSE 344
28.5 Custom Function Calls 344
Function Calls in External Classes 344
Extending the Session Pooling Handler 345

29 **Appendix A – Code Pages 347**
29.1 OEM Code Pages 347
29.2 ANSI Code Pages 348
29.3 EBCDIC Code Pages 348
29.4 ISO Code Pages 349
29.5 Windows Code Pages 350
29.6 DOS Code Pages 350

30 **Appendix B – REXX Conversion 351**
30.1 REXX to NetRexx 351
R2NR Syntax 351
The Basic Algorithm 352
Translation Rules 352
Declaration File 353
Variable Name Substitution and Variable Declaration 353

Goto's 354
"Conditional" Comments 354
Unsupported REXX Features 354

31 **Appendix C – NetPhantom Considerations 355**
31.1 Using Java Threads in NetPhantom 355

32 **Appendix D – Client National Language 357**

33 **Appendix E – User Exits 359**
33.1 The Event User Exit 359
33.2 The LU Mapper User Exit 360
33.3 The User Authentication User Exit 361
33.4 The Telnet Host Interface 361

34 **Appendix F – File Transfer from Server to Client 365**
34.1 Panel Part 365
34.2 Server Part 365
34.3 Class Structure on the Server 366

35 **Appendix G – Component Cross-Reference 367**
35.1 Report Structure and Data 367
Objects 367
Popup Menu 368
Text File 369
Application 369
Panels 369
Host Data 370
Control Bars 370
35.2 How to Use the Cross-Reference 371

Index 373

Preface – Intended Audience

The *Administrator's and Developer's Guide to NetPhantom* is not intended as end user documentation but rather as a guide and reference for NetPhantom application developers, system administrators and programmers. It should be noted that the documentation, especially *The Developer's Guide*, has been written with the assumption that the reader has some familiarity with Phantom Hurricane.

As NetPhantom is a broad product, we have divided the documentation into two sections: the *Administrator's Guide* and the *Developer's Guide*. Of course, there are areas of overlapping interest, but generally administrators will want to focus on the first 21 chapters and the appendices, while NetPhantom application developers will benefit most from chapters 21-29 as well as the appendices. We recommend that programmers read chapters 19 and 20 and Appendix B, as well as the online NetPhantom Java API and Source Listings.

A separate documentation, *NetPhantom Eclipse Developer's Reference*, is intended for developers that will use the NetPhantom Editor together with the Eclipse Integrated Development Environment (IDE).

1 Introduction

NetPhantom consists of three parts – the NetPhantom Editor, the NetPhantom Server and the NetPhantom Client. The Server communicates with the Mainframe or AS/400 hosts and runs the NetPhantom applications. The Client displays the graphical panels and perhaps the terminal window when no graphical panel is available.

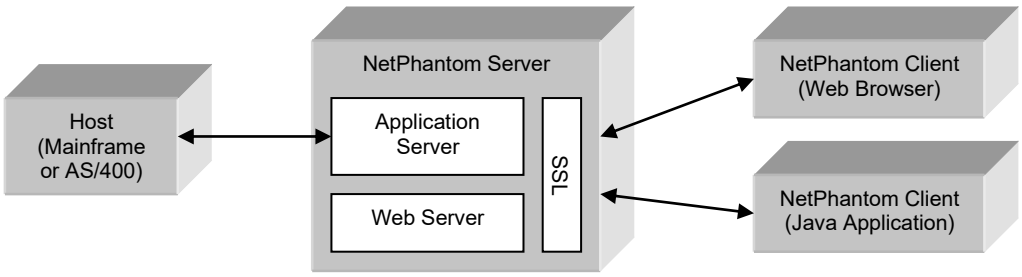
An integrated Web Server is included in the NetPhantom Server to facilitate handling of NetPhantom Clients. Communication between the server and the client can optionally be done securely using SSL – Secure Socket Layer.

The NetPhantom Editor is integrated into the Eclipse IDE to provide a modern and powerful development environment, fine-tuned for source code editing and application development.

1.1 System Overview

NetPhantom is divided into a Server and a Client. The entire system is written in Java. The Server and Client communicate with each other, i.e. the Client never communicates directly with the host session.

The Client runs on a local workstation and interacts with the user. The Server handles the connection to the host and the administration of NetPhantom.



The basic architecture of NetPhantom

Because NetPhantom is based on Intranet/Internet technique, the protocol used for communication between Client and Server is standard TCP/IP.

NetPhantom is a distributed GUI application, where most of the program logic is located on the server. An analogy to NetPhantom is the relationship between Mainframe and Terminal, where the Mainframe represents the NetPhantom Server and the Terminal is the NetPhantom Client, the Client having an advanced GUI rather than a CUI (Character-based User Interface). The Mainframe does all the work; the Terminal is "dumb". The NetPhantom Client is like a "dumb" GUI Terminal.

1.2 NetPhantom Editor

An integrated editor for application development is included in NetPhantom. The editor has the following features (in addition to previous versions):

- Eclipse IDE integration.
- Direct communication between host/terminal editors.
- Automatic application build (in background).
- Multiple application merge.

In this development environment, applications can also be tested in a real server environment, running on the development computer. This means that most aspects of the

running application can be simulated before it has been deployed on a live or dedicated test server. This simplifies the development/testing process in a significant way.

This also means that a large part of the *Server Configuration* functionality is available in the NetPhantom Editor.

Eclipse IDE Integration

The Eclipse IDE integration is covered in *NetPhantom Eclipse Developer's Reference*, a separate documentation due to its nature in *NetPhantomEclipse.PDF* installed with the Server.

1.3 NetPhantom Server

The Server is a Java application with no Java Security issues, for example it can read/write files, open and create sockets, etc. It can also use the JNI (Java Native Interface) to perform some system dependent tasks, such as server event notification using a third-party product.

The largest amount of code (as well as primary memory) is in the server. The server must be able to read the runtime files, connect to the host, handle host screen changes, and create virtual panels that are reflected to the client. These virtual panels and the reflection logic are how NetPhantom distributes the presentation logic from the server to the client and transports user input and interaction to the server.

1.4 NetPhantom Client

The Client is generic and is/should be kept as small as possible. As it is written as a Java applet *and* application at the same time, the client can run inside a browser, e.g. over the Internet, as well as standalone on the PC or NC (Network Computer).

Swing – The Java Foundation Classes

The Swing class library, also called JFC (Java Foundation Classes) provides a set of different looks-and-feels such as *Metal* (JLF – Java look-and-feel), *Nimbus* or *Windows*.

The GUI interface is mainly handled by standard components in the JFC (Swing) class library.

Server Choice

The client will decide which server to use depending on the following rules:

- A server is not connectable using TCP/IP.
- The server refuses the client connection, e.g. because it is in a state of shutdown, the application requested is blocked for current use by the server administrator, or the maximum number of concurrent users on the server has been reached.

Java Client API

A client API is provided for Java classes or programs to interface to the client. See the Java package `se.entra.phantom.client`.

1.5 Load Balancing

NetPhantom load balancing allows the load of concurrent users to be spread over several different servers. The system uses redirection to connect a client to the appropriate server. At least one server acts as a controller while the others serve as slaves. When a new client tries to establish a connection, the controller will check the see which server itself included, should take the connection.

1.6 Communication between Client and Server

The communication between the Client and the Server is based on data streams called *transactions*. These transactions will have different structures depending on whom they are meant for. The receiver knows how to extract the transaction information.

Examples of transactions are transactions for creating a new object on the client screen, transactions sent from client to server because of some user action on a control or component on the client screen, transactions sent from host to client because of a host application updating a screen, etc.

The meta-data sent from the client is translated and handled on the server. This data consists of changed GUI components as well as triggering action (typically a push button or a menu item). The server notifies the client of screen updates and other events.

The meta-data is a NetPhantom Transaction (see Java class `se.entra.phantom.common.Transaction`). A typical transaction to create a GUI panel from the server to the client is 0.5-5 KB. An update from the client to the server is very small, typically less than 200 bytes.

The client only displays and processes the GUI built in the Runtime Application. It doesn't need to know anything about a host, thus all host-related data in the application as well as the panels is handled only by the server and not passed on to the client. For this reason, the communication between the server and the client is called *Transaction Oriented*.

The transactions between the server and the client can be encrypted using SSL (Secure Socket Layer).

2 System Requirements

Note: NetPhantom Client and Server are pure Java, not restricted to a specific platform. These requirements are intended as a guide to understanding the basic needs of the Server.

2.1 Hardware Platform

The CPU

Some tested platforms are:

IA32	32-bit Intel processors and clones
IA64	64-bit Intel processors and clones
Sparc	Servers and Workstations
PowerPC	IBM RS/6000 or pSeries Server

Physical RAM

The primary memory requirements can be calculated by the following formula:

```
550 + sslSize + appSize * 10 + client * 0.5 + httpReq * 0.2 + cache
    + htmlCache * 10
```

where:

sslSize	is 40 MB times the amount of SSL settings configured (and used). If SSL is not used, this value is zero,
appSize	is the file size (in MB) of all the runtime files the server loads,
client	is the number of concurrent clients,
httpReq	is the amount of simultaneous HTTP requests that the web server processes,
cache	the size of the resource cache used for the web server (except HTML resources),
htmlCache	the size of the HTML document cache size used for the web server.

For a normal server with 2 standard runtime applications (each 10 MB in file size) and 500 concurrent users, the server memory requirements would be approximately 1 GB (SSL is assumed and up to 50 simultaneous HTTP requests).

A Linux 64-bit server would require a dual core Intel 2.6 GHz with 4 GB RAM. A 32-bit Java VM for NetPhantom Server can only handle up to 500 clients due to memory limitations. A 64-bit Java VM can allocate more memory, thus handle more clients. A single 64-bit NetPhantom Server Java VM process should not handle more than 2000 clients; consider using the NetPhantom Cluster Controller to cluster several servers on the same physical server.

It is recommended to use a 64-bit JVM for the server in all cases.

Disk Space

Minimum 80 Mbytes of disk space is needed for NetPhantom Server. Additional space depends on the size of your applications.

2.2 Operating Environment

Operating System

The operating system must support a full native-threaded Java environment, BSD-style sockets, and TCP/IP Networking. Some tested environments are:

IA32	The 32-bit versions of Microsoft Windows 2000/2003/2008/XP/Vista or Windows 7 to 10. Linux Kernel 2.6 or newer preferred. Apple OS X 10.6.3 or newer preferred.
IA64	The 64-bit versions of Microsoft Windows Server 2003, 2008 (R2), 2012 (R2), 2016, 2019 or Windows XP/Vista, Windows 7 to 11. Linux Kernel 2.6 or newer preferred. Apple OS X 10.6.3 or newer preferred.
Sparc x86	Solaris 2.6, and better (server only). Solaris 10 or better (server only).
PowerPC	AIX (server only).

Java Environment

The Java environment must be Java Platform: version 1.8.0 update 151 or better for the NetPhantom Server and Editor.

It is recommended to use Open JDK version 17 or better in 64-bit.

2.3 Network

The server must be connected and operational on a TCP/IP network before attempting to install or run the NetPhantom Server.

Connection to a TN3270 (for Mainframe) and/or TN5250 (for AS/400) server is needed.

At least two available TCP/IP ports for inbound connections for clients and administrators.

The NetPhantom Server installs its own Web Server. It is usually connected to port 80, this may create a conflict if you already have a Web server connected to that port.

2.4 Client System Requirements

The NetPhantom Client can operate in two ways: as a Java application, use Java 8 JNLP or use NetPhantom Starter.

A Java Environment is needed:

- Java Platform (version 1.8.0_151 minimum, or Java 11 to 21).
- Screen resolution of 640 x 480 or better, a mouse and 1 GB of RAM.
- A minimum processor speed of 800 MHz.
- A TCP/IP connection to a server through a network or dial-up line.

3 NetPhantom License System

3.1 Introduction

NetPhantom Servers use a license system to control the number of concurrent users per server and to enable certain features, such as SSL. Each server must be issued a unique license code.

A valid license can be in three different states:

- Activated: The license is ready to use with no limitations (other than those for expiration date, number of users etc.).
- Not yet activated: The license code has a grace period of 90 days to be activated. Until that point, the license will be available for use as activate, but after the grace period it will be inactivated (i.e. will not allow the use of the server).
- Expired: The license had a built-in expiration date which has been reached. The license is not possible to use.

The information about the licenses: status, support agreement, contact information, is stored in the NetPhantom license database. Communication between the NetPhantom Server and the license database enables efficient maintenance of the license and the NetPhantom installation.

However, in more complex installations, for example when several servers are run in a load balancing solution using a Cluster Controller and having backup servers for redundancy, the license system becomes unmanageable.

The License Manager is therefore the preferred license solution and should be used as soon as several servers are present but is not required for “simple” stand-alone servers or Cluster Controller solutions without backup servers.

3.2 License Management Made Simple

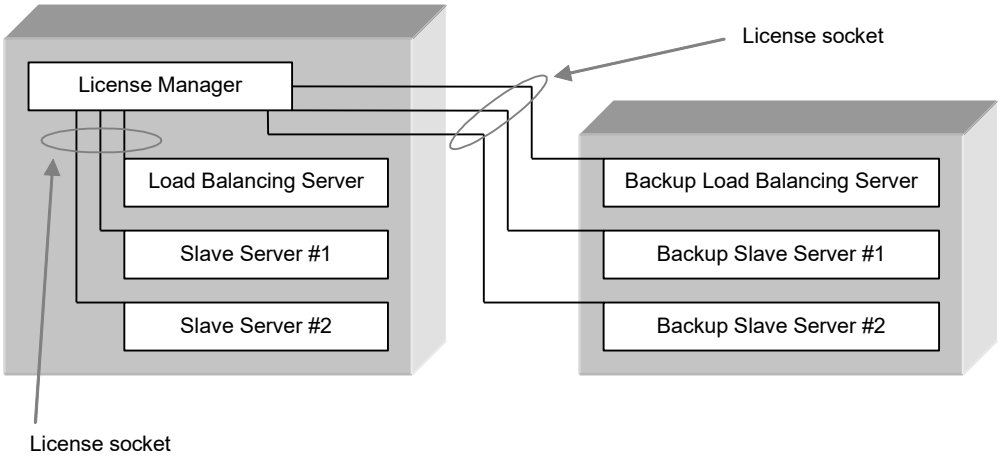
The License Manager is a small server software that runs on a machine as a stand-alone Java process or as a Windows Service. The License Manager uses a single license code and in turn controls all server licenses of an entire solution: cluster controller(s), load balancing controller(s) and slave server(s).

It distributes all enabled additional features, such as SSL, to all connected servers. This allows great flexibility to change the server configuration (server count, IP addresses, load balancing, slave servers and/or backup servers) without requiring new license codes. Each server has a permanent socket connection to the License Manager that informs the respective server of how many client connections are allowed.

A site running one or several NetPhantom Servers can choose to use the License Manager to control all licenses on the servers. The license servers can also be set up to allow backup configuration for redundancy.

Settings for the License Manager are:

- Number of concurrent users in total (called shared users) or divided into Java Clients, HTML Clients and RAPP Clients,
- Enabled features (such as SSL),
- Maximum number of active servers (backup servers are not included, hence backup servers can be added without additional licenses).



3.3 Server Types and States

The following is an introduction to terminology we will be using later in the document. We do not go into detail in this section. See section *Distribution of Licenses by the License Manager* for more information.

Backup Server vs. Primary Server

Using the License Manager, a new server type has been added: the *Backup Server*. A backup server is used to create redundancy for another server, the *Primary Server*, in case this server stops functioning. A primary server may be a Load Balancing Master server, a Load Balancing Slave server or simply a single server. What makes it the primary server is that it is initially enabled for client connections. A backup server is initially *not* configured to accept client connections. It will only begin accepting client connections if the primary server to which it is assigned becomes unavailable for some reason. Support for having multiple backup servers for one primary server is also provided.

Connecting State

There is a short period in which a server has been started but has not yet had time to identify with the License Manager and thus has not been issued its licenses. The server is in this state only for a matter of milliseconds.

Active State

A server is in the Active state when it has been identified by the License Manager and is enabled for client connections. Under normal circumstances the Primary server is the active server.

Non-active State

Only backup servers can be in a non-active state. A Primary server is always in Active state unless it for some reason loses contact with the License Manager in which case it enters non-connected state (see below). In the non-active state, a server is in contact with the License Manager, but will not accept client connections. This is the typical state of a backup server.

Non-connected State

Both server types may be in a non-connected state. This occurs when the License Manager determines that the connection to a server is broken.

What Happens When Servers Change State

To determine if servers are present, the License Manager listens for socket transaction from the connected servers with a certain read timeout. This will detect not only network problems (i.e. if the socket to the server is broken), but also enables the License Manager to

check if a server is heavily overloaded or hung. In these cases, the License Manager will close the connection to that server and assume it is no longer present, thus enabling backup server(s) to take its place, if this is configured.

The backup server connects to the License Manager and informs it of the server to which it is assigned. If the Primary server to which the backup server has been allocated loses its connection with the License Manager for some reason, the backup server will temporarily change to active state. When the Primary server once again establishes a connection with the License Manager, the backup server will return to non-active state. New client connections will not be allowed, but those clients still connected to the backup server will continue to run with the backup server until the user closes the client.

3.4 Running the License Manager

The License Manager is started from the NetPhantom installation directory using JDK 1.6.0_30 (or better) as:

```
java -Xmx16m -classpath NetPhantomServer.jar
     com.netphantom.licmgr.LicenseManager
```

If there is a network problem and the socket port 1791 used to listen for server connections cannot be opened, a retry to open this socket is performed every 30 seconds.

Running the License Manager on a Different Machine

It is also possible to run the NetPhantom License Manager on a machine that does not have a NetPhantom server installation. This requires that you copy the `NetPhantomServer.jar` file and, if the License Manager will be run on a Windows Server, the `WindowsNTEventLog.dll` file.

3.5 Using NetPhantom License Manager as a Windows Service

The NetPhantom License Manager can be run as a Windows Service under Windows 2000/2003/XP/Vista, Windows Server 2008/2012 (R2)/2016/2019 or Windows 7/8/8.1/10).

The purpose of having the server running as a Windows Service is that services do not require a user to be logged on to run. The server process runs in a daemon mode without console or GUI.

Please see the NetPhantom Windows Services chapter for details on how to install and run the license manager as a service.

3.6 Configuring the License Manager

When the License Manager is started, it tries to load the `license.ini` file in the current directory containing the settings as described below. These are the settings that will specify how the connection technically will work. If this file is not found, the License Manager will use default settings (also described below).

The “license.ini” File

The file `license.ini` contains all settings that the License Manager needs. By default, this file is not present and thus all default values apply.

The following settings are loaded from the file:

```
; ********************************************************************
; *
; *  LICENSE MANAGER CONFIGURATION FILE
; *
; ********************************************************************

[base]

; The length of the listening queue for the server socket
; (default is 50).
queueLength=50

; Maximum time to wait for a server to identify itself in seconds
; (default is 20 seconds).
connectionWaitTime=20

; Maximum time to wait for read time-out in seconds, must be larger
; than aliveCheckTime (default is 60 seconds).
readTimeout=10

; Time in seconds between checking if a server is alive (default
; is 20 seconds).
aliveCheckTime=20

; Log file settings.
logFile=license.log
;appendLogFile=1

; Mechanism for license distribution: "even license distribution" (=1)
; [default], or (=0) is "first come, first served".
evenDistribution=1

; Port that should be used by the license manager. Default is 1791.
lmPort=1791
```

In addition to these values, the actual configuration of the license manager is kept in an encrypted file. These values are configured through the user interface.

3.7 Configuring the License Manager from Server Administration

The NetPhantom Server Administration program is used to configure the servers that connect to the License Manager as well as the License Manager itself.

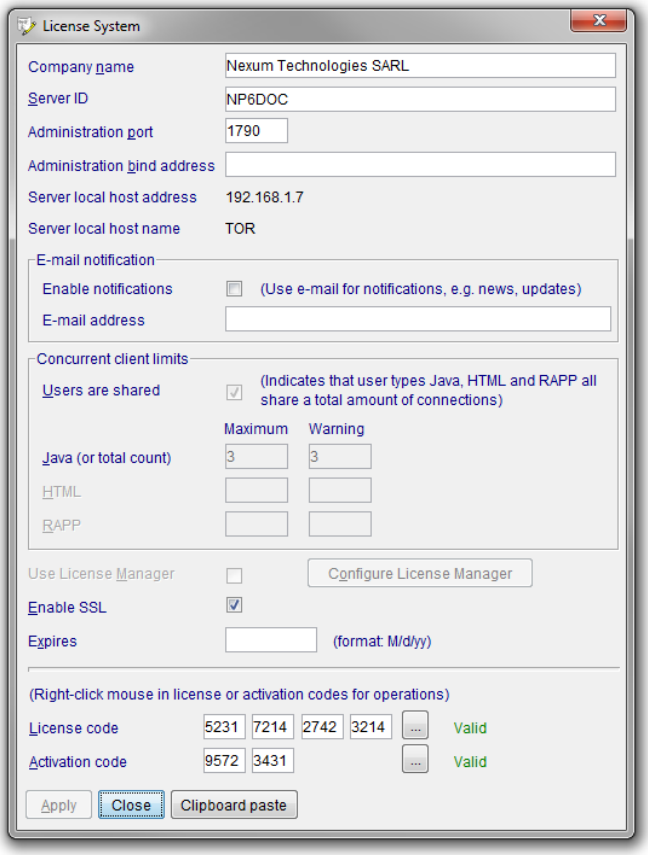
When a default License Manager installation is first started the following output is printed on the screen:

```
NetPhantom(R) License Manager, Version 7.50
-------------------------------------------

(C) Copyright Mindus, 2023.
All rights reserved.

05 may 2023  11:51:14.816  -    -    Using settings in 'license.ini'
05 may 2023  11:51:14.916  -    -    Listening on port 1791 for IP
   address 10.254.1.10, localAddress = 10.254.1.10, localName =
   netphantom.com, queueLength = 50
05 may 2023  11:51:14.936  -    -    The license code is not valid,
   bindAddress '', port 1791, companyName 'My Company Inc.',
   maximumServerCount 0, Java 0, HTML -1, RAPP -1, features '',
   connectionWaitTime 20000, readTimeout 60000, aliveCheckTime 20000
```

In the example above, the License Manager is accessible at the IP address `10.254.1.10` or the DNS name `netphantom.com`.

The License System Dialog

Start the Server Administration Program using a NetPhantom Java Client and select the menu item **Server – License**. The following dialog box is displayed:



The License System dialog allows you to allocate licenses to different types of clients.

Company name Fill in the name of your company, e.g. `My Company Inc`.

Server ID Enter the server's name or IP address here e.g. `GHOST`.

Administration port This is the port that will be used by the Server Administration Program. Default is 1790.

Concurrent client limits Enter appropriate values for **Java (or total count)**, **HTML** and **RAPP** users including possible check of the option **Users are shared**. These are the total numbers of concurrent clients for the NetPhantom Server being configured.

Use License Manager If you will be using the License Manger to control the distribution of licenses, check the **Use License Manager** option to enable the **Configure License Manager** button. This must be done for *all* servers that will be handled by the License Manager.

The following options are only enabled when you are *not* using the License Manager to handle license distribution. If you will be using the License Manager, these options are set in the **Configure License Manager** dialog, which is accessed via the **Configure License Manager** button, see the *License Manager Configuration* below.

Enable SSL Check to enable SSL encryption.

Expires A text string denoting a date when the license will expire. If left empty the license will never expire.

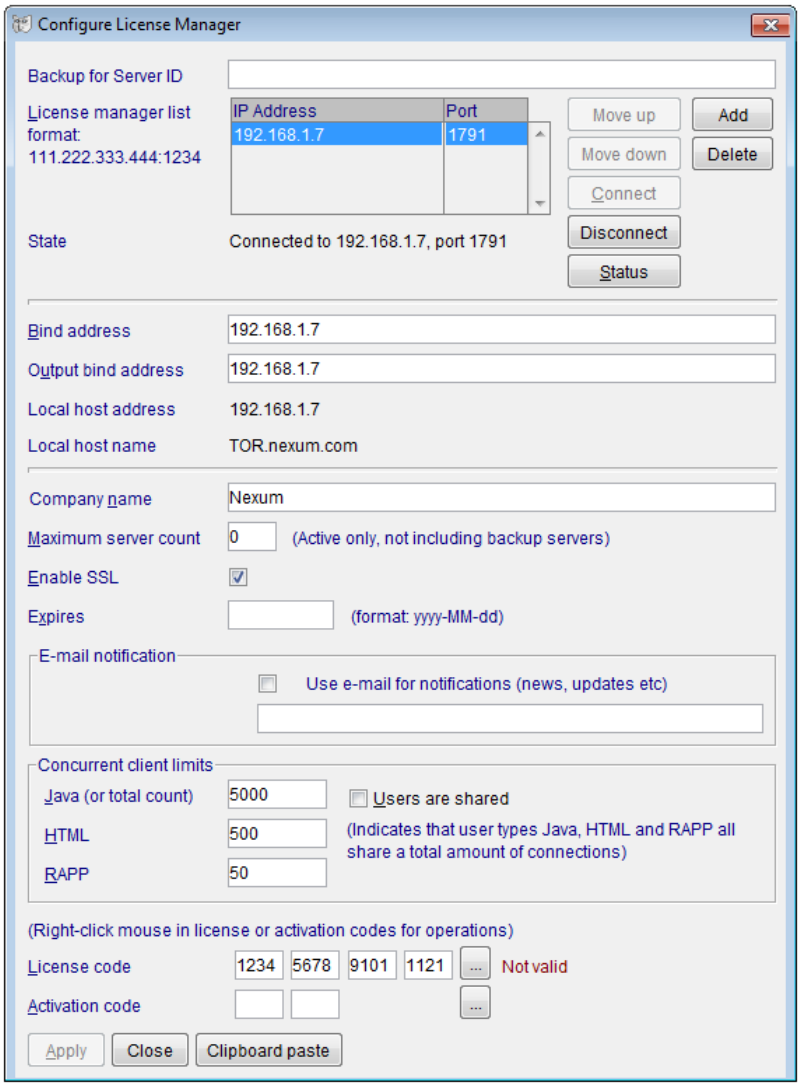
License code Enter the license code for the NetPhantom Server.

License code *(right-click)*	• Request license code by mail: Generate an email with all necessary information to generate a license code. The recipient of the email is NetPhantom support. • Request license using clipboard data: Generate the same data as above but put it in the system clipboard to be transferred to NetPhantom support by other means available. • Deactivate license code by mail: Generate an email to be sent to NetPhantom support requesting that the current code should be deactivated. • Deactivate license using clipboard data: Generate the same data as above but put it in the system clipboard to be transferred to NetPhantom support by other means available.
Activation code	Enter the activation code for the license.
Activation code *(right-click)*	• Activate on-line: send a direct request to the NetPhantom support license database asking for an activation code. If the support agreement for this license is active, the activation code will be returned immediately. • Activate by email: Generate an email with all the necessary information to generate an activation code. The recipient of the email is NetPhantom support. The answer in this case will be sent back by email. • Activate using clipboard data: Generate the same data as above but put it in the system clipboard to be transferred to NetPhantom support by other means available.
Clipboard paste	When an answer to either a license or an activation code is received by email, the content can be put in the system clipboard (Ctrl+C) and subsequently pasted into the form.

License Manager Configuration

License Manager Configuration consists of two parts: the configuration of the NetPhantom server information for the License Manager and the configuration of the License Manager settings themselves. The License Manager settings need only be configured once. For each succeeding NetPhantom server, the License Manager information will be displayed in the **Configure License Manager** dialog after the NetPhantom server has successfully connected to the License Manager. However, for *each* server you must configure the connection to the License Manager, i.e. specify the **Host address** and **Port** of the License Manager (in the License Manager list) as well as indicate if the server is *Primary* or *Backup* in the **Backup for Server ID** field.

To begin configuration of the License Manager, fill in the IP address of your License Manager, and then press **Connect**.



The License Manager configuration dialog box controlled from the Server Administration program of a NetPhantom Server.

When the server has connected to the License Manager, all entry fields are enabled, and the license information can be entered. This configures the License Manager remotely from the server in question.

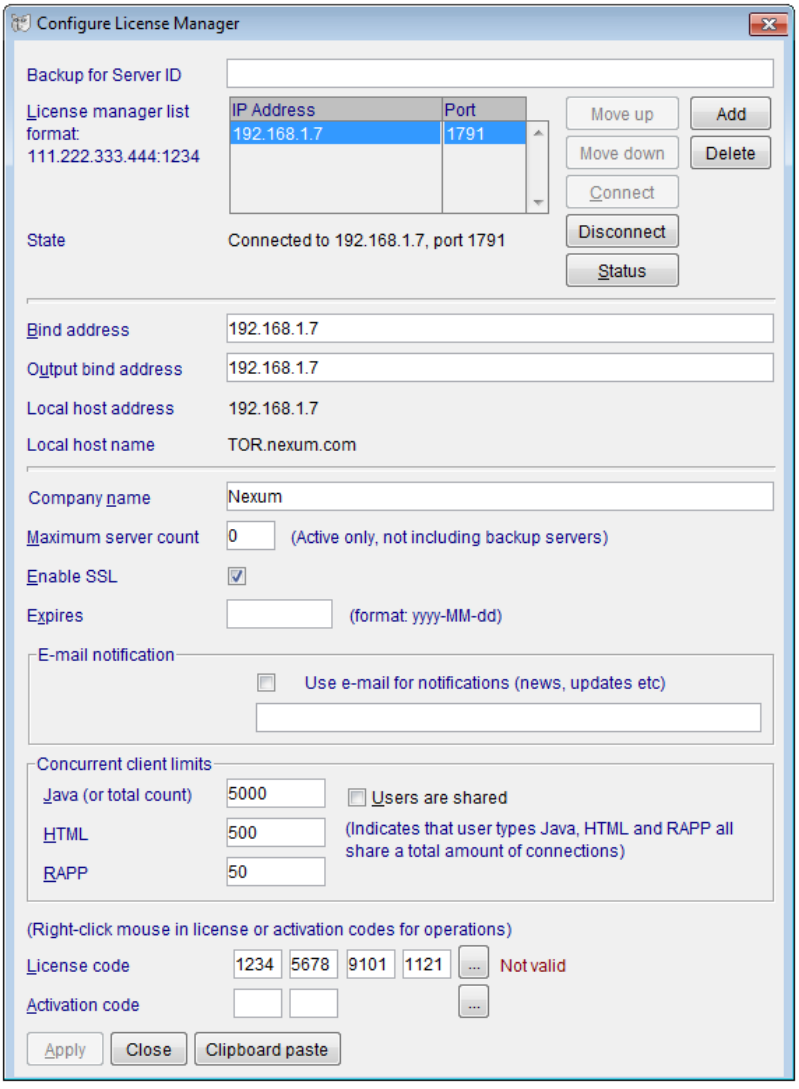
Configuring a License Manager means that all servers that are currently connected to the license manager in question will be disconnected. When connecting to the License Manager, an option to connect in read-only mode is given. Connecting in read-only mode is done only to fetch the license information and that means that it is not editable and thus cannot be transferred to the License Manager. If this mode is used, the servers connected will not be disconnected from the License Manager.

The following information applies to remote License Manager configuration:

Backup for Server ID	If the NetPhantom server is a Backup Server, enter the Server ID of the Primary NetPhantom Server it will be backing up.

License manager list	The list of License Manager's that are included in the solution. The first entry in the list is the main license manager and the subsequent are the backup license manager(s), in the order they will be used in the case of fail-over. The list consists of pairs of IP addresses and port numbers, identifying where the License Manager is installed and the port to use in the communication.
State	Displays the current state of the NetPhantom server in relation to the License Manager (see below).
Bind address	The bind address for the License Manager. This bind address can be a DNS name or an IP number. A bind address ensures that the License Manager will only bind on a single address rather than all addresses. *Note that this is not the same bind address as for the NetPhantom Server itself!*
Local host address/Local host name	These values are displayed only and show the Local host name and address that is currently used for the License Manager.
Company name	Enter the name of your company. This should be set to the same Company name as specified for the Server License Configuration (and is therefore specified twice).
Maximum server count	The maximum number of active (primary) servers, i.e. do not include backup servers in this number. Zero means that no maximum server count is used (but must match the license code).
Enable SSL	Check to enable SSL encryption.
Concurrent client limits	Enter appropriate values for **Java (or total count)**, **HTML** and **RAPP** users including possible check of the option **Users are shared**. Remember that these are the total numbers of concurrent clients for all NetPhantom servers per client type.
License code	Enter the license code for the NetPhantom server.
Expires	A text string denoting a date when the license will expire. If left empty the license will never expire.
License code	Enter the license code for the NetPhantom server.
License code (right-click)	• Request license code by mail: Generate an email with all necessary information to generate a license code. The recipient of the email is NetPhantom support. • Request license using clipboard: Generate the same data as above but put it in the system clipboard to be transferred to NetPhantom support by other means available. • Deactivate license code by mail: Generate an email to be sent to NetPhantom support requesting that the current code should be deactivated. • Deactivate license using clipboard: Generate the same data as above but put it in the system clipboard to be transferred to NetPhantom support by other means available.
Activation code	Enter the activation code for the license.

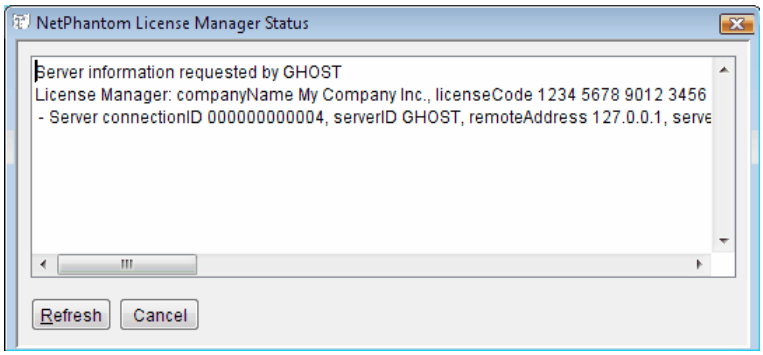
Activation code (right-click)	• Activate on-line: send a direct request to the NetPhantom support license database asking for an activation code. If the support agreement for this license is active, the activation code will be returned immediately. • Activate by email: Generate an email with all the necessary information to generate an activation code. The recipient of the email is NetPhantom support. The answer in this case will be sent back by email. • Activate using clipboard data: Generate the same data as above but put it in the system clipboard to be transferred to NetPhantom support by other means available.
Clipboard paste	When an answer to either a license or an activation code is received by e-mail, the content can be put in the system clipboard (Ctrl+C) and subsequently pasted into the form.



*When the **Apply** button is pressed, the panel data is transferred to the License Manager for processing. A message box is presented with the action taken; normally the License Manager is restarted automatically with the new settings.*

The License Manager configuration in the panel is sent to the License Manager when the **Apply** button is pressed. The License Manager checks the license code and is restarted with the new settings. After about 10 seconds, the server reconnects with the License Manager. If the server does not connect, the License Manager will probably receive information that causes it not to be able to create the server socket (for example the port is already in use, or the bind address is invalid).

The **Status** button will display status information from the License Manager.



The information in the multiple line entryfield in the example above is:

```
Server information requested by GHOST
License Manager:
    companyName My Company Inc.,
    licenseCode 1234 1234 1234 1234 Invalid,
    bindAddress netphantom.entra.se,
    localAddress netphantom.entra.se,
    localName netphantom.entra.se,
    maximumServerCount 0,
    currentServerCount 1,
    currentConnections 0 (Java 0, HTML -1, RAPP -1),
    peakConnections 1 (Java 1, HTML 0, RAPP 0)
 - Server connectionID 000000000001,
    serverID GHOST,
    remoteAddress 10.254.0.40,
    serverAddress 10.254.0.40,
    serverName ghost,
    masterServer n/a,
    backupForServer n/a,
    Java 100(1), HTML -1(0), RAPP -1(0),
    peakConnections 1 (Java 1, HTML 0, RAPP 0)
```

For the line `Java 100(1), HTML -1(0), RAPP -1(0)` above, the number in parenthesis indicates the current user count.

3.8 Distribution of Licenses by the License Manager System

Server Connections to the License Manager System

Before discussing license distribution, it is important to understand how server connections are handled. The following rules apply for servers connecting to the License Manager System (the primary License Manager or one of the License Manager backups).

- A server is denied socket connection if the maximum number of active servers has been reached (and the connecting server is not a backup server for an active primary server).
- A server is denied socket connection if its settings are invalid: backup server ID is the same as its own server ID or no server ID at all is specified.
- A primary server (i.e. has no backup server ID specified) is accepted if the maximum server count is not exceeded. When this server connects, all its backup servers will enter “non-active state”, i.e. will no longer accept new client

connections, but any existing client connections in the backup servers are preserved. After server connection, all available licenses will be distributed to the active servers (see *License Distribution* below).

- When a backup server connects to the License Manager and no primary server is active, it will temporarily enter the “active state” until the Primary server connects. If other backup servers already have the “active state”, the connecting backup server will remain in “non-active state”, i.e. will not receive client connections. After server connection, all available licenses will be distributed to the active servers (see *License Distribution* below).

- When a primary server connects, it will cause all backup servers to enter “non-active state” if they are not presently in that state. After server connection, all available licenses will be distributed to the active servers (see *License Distribution* below).

- When an activated server (primary or backup) connects, a user license distribution will be initiated in the License Manager (see *License Distribution* below).

The following rules apply for servers when they disconnect from the License Manager (or if the License Manager finds them “hung” or non-responsive):

- When a primary server disconnects, the first available backup server will be activated, i.e. will enter a temporary “active state”. This backup server will receive the maximum client connections it is configured for, assuming another server does not use these. After server disconnection, all available licenses will be distributed to the active servers (see *License Distribution* below).

- When a backup server that is currently in “active state” disconnects, any other backup server for the primary server will be activated, i.e. will enter a temporary active state. This backup server will receive the maximum client connections it is configured for, assuming another server does not use these. After server disconnection, all available licenses will be distributed to the active servers (see *License Distribution* below).

When backup servers have been in a temporary active state, they will inform the License Manager whenever a client disconnects, thus enabling this client license to be distributed among the active servers.

License Distribution

The License Manager and the servers can request usage of maximum concurrent client counts as either “shared” or specific by type (Java, HTML or RAPP). The License Manager is configured to have a maximum count of shared clients or a maximum count of the specific client types.

When an active server is present in the License Manager and License Distribution is going to take place, four (4) different possibilities exist (see *Specific License Distribution* below).

When the License Manager distributes available licenses, it uses one of two available mechanisms: “first come, first served” or “even” distribution.

The default is the “first come, first served” mechanism. When no more licenses are available, remaining servers will receive a maximum user count for less than what they expect (maybe none). This is, however, dynamically recalculated if a server connects or disconnects (see *Server Connections to the License Manager* above).

The alternative mechanism tries to distribute the licenses evenly over the connected servers. For example: 5 servers want 750, 750, 750, 500 and 500 licenses respectively, but the License Manager is providing a maximum of 2500 in total. Using the "first come, first served" mechanism, the first 3 servers connecting to the License Manager would receive

750 licenses each (remains 250), the 4th server 250 and the 5th no licenses at all. With the new "even license distribution", the total number of licenses is first computed, in this case 3250. If the first 3 servers connect, they will get 750 each, and 250 licenses will remain. But when the 4th server connects, the available license count is exceeded by 250. The first 3 servers would then receive 750*2500/2750 = 681 and the 4th server 500*2500/2750 = 454 (a perceptive reader will note that 3 licenses will remain). When all 5 servers connect, the 3 first will get 750*2500/3250 = 576, server 4 and 5 gets 500*2500/3250 = 384 (and 4 licenses will remain). Please see *Chapter Configuring the License Manager* for the description of how to configure this alternative mechanism.

Specific License Distribution

The following describes how licenses are distributed depending on the configuration of the License Manager and the connecting server regarding shared or specific user types:

1. Shared users for both License Manager and server:
   requested users up to available level is set for the server.

2. Specific user types for License Manager and server:
   requested users up to available level is set for the server for each user type.

3. Shared users for License Manager but specific user types for server:
   The total requested specific user count will be used as the base for how many users the server will receive for its specific user types (a proportional value).

4. Specific users for License Manager but shared users for the server:
   The requested user count for all user types will be the base for how many clients to be assigned for the server, and the License Manager will proportionally deduct the assigned client amount from its specific user types.

The scenarios 3 and 4 are not recommended because they are hard to understand for an administrator (although mathematically it is a simple operation). It is strongly recommended to use the same user type (shared or specific) for all servers and the License Manager.

3.9 Test License Code

To enable testing a complex and large NetPhantom solution using load balancing, backup servers and perhaps also inactive machines that can quickly be put into production, a "test license code" is now available for License Managers. This license code grants all settings and features for a License Manager and thus for connecting servers, but only for a period of 2 (two) hours. A NetPhantom Server that connects to a License Manager with a test license code will automatically shut down after 2 hours unless it connects to another License Manager with either a permanent (normal) license code or a demo license code. The test code is: "0000 0000 0000 0000".

4 Getting Started

This chapter describes how to perform a normal NetPhantom installation.

The full installation of NetPhantom will require about 80 MB of hard disk space. Please see the `index.html` document on the root of the CD, which includes updates to the information provided with NetPhantom.

4.1 Installation

Certain servers do not have access to a graphical user interface, such as a remote Telnet terminal session to a UNIX machine. In this case, see the section *Installation using an Installation Image (JAR)* below. If a graphical user interface is available, you may use the installation procedure that follows.

Note: The result of both installations is identical in terms of directories and files.

Installation using the SETUP.EXE

This installation will only work under Windows. This setup allows you to install the NetPhantom Server and Editor together. The JAR installation below only installs the Server on any Java supported platform.

NetPhantom Quick Start is available in another setup program, see the separate *NetPhantom Eclipse Developer's Reference* PDF document. We strongly recommend using this setup program as it installs NetPhantom, Eclipse and OpenJDK Java version 8 to the latest available.

Installation using an Installation Image (JAR)

In the root directory of the NetPhantom CD, there is a file called `preinstall.jar`. You may use this file to extract the installation image that the *Browser Installation* would otherwise create.

Note: It is recommended to use this installation procedure for servers using e.g. a remote access to a UNIX machine (on a Telnet Terminal Session).

Steps to create the installation image:

1. Create a directory on the destination and make it the current directory.
2. Unzip the `preinstall.jar` file or use the Java utility `jar` with the command:

```
jar -xf preinstall.jar
```

Please note that the user permissions for different startup scripts (`startclient.sh`, `startserver.sh` and `startlicensemanager.sh`) need to be changed before they can be executed. This is done with the help of the command:

```
chmod 700 *.sh
```

You should now have a full NetPhantom installation.

4.2 Startup Scenario

The following scenario describes the startup of the NetPhantom Server and a Client.

Start the NetPhantom Server

Once the NetPhantom installation is complete, start the NetPhantom Server (as described in section *4.5 Starting the Server* below). If there is a configuration problem, the server administration program will automatically be started when a client connects to the server.

Initial Server Configuration

The NetPhantom Server is by default configured to use the following ports:

80	Default HTTP port.
789	The "NetPhantom Client" port.
8080	Another value for an HTTP port.
1789	Another value for the "NetPhantom Client" port.
18080	Another value for an HTTP port.
28080	Another value for an HTTP port.

When the server starts, it will try to use the ports listed above, and if a port could not be used, an error event is displayed in the server log.

Server Log Output

The following text shows the typical output of the NetPhantom Server log when using the *Initial Server Configuration*.

```
NetPhantom Editor Version 7.60 (Build 8200).
(C) Copyright Mindus SARL, 2023.

05 may 2023  10:32:58.467  <NetPhantom>      <System>    ---               I  SI0016    Event log
started
05 may 2023  10:32:58.484  <NetPhantom>      <System>    ---               I  SI0001    The server is
starting up: environment settings:
  NetPhantom version: 7.60 (Build 8200)
  Server ini file   : C:\NetPhantom 7 QS\server.ini
  OS name           : Windows 10
  OS version        : 11.0
  Java version      : 17.0.7
  Java vendor       : Eclipse Temurin
  Java home         : C:\NetPhantom 7 QS\jdk
  Current directory : C:\NetPhantom 7 QS
  Class path        : NetPhantomServer.jar;NetRexxR.jar;openxml-1.2-np.jar;bcprov.jar;...
05 may 2023  10:32:59.392  <NetPhantom>      <System>    ---               I  SI0071    License
System: License code is valid, maximum 3 concurrent shared users
05 may 2023  10:33:00.178  <NetPhantom>      <System>    ---               I  SI0021    The User
Authentication Interface is started
05 may 2023  10:33:00.178  <NetPhantom>      <System>    ---               I  SI0005    The server is
loading runtime application: userAuthenticationApplication = authentication/userauth.jar
05 may 2023  10:33:00.382  <NetPhantom>      RtLoader    0000000000000016  I  <n/a>     Creating
runtime class loader, classpath =
npjar:file:/C:/NetPhantom%207/samples/sendmail/sendmail.jar!/;npjar:file:/C:/NetPhantom%207/samples/se
ndmail/sendmail.jar!/bin/, system class path = NetPhantomServer.jar;NetRexxR.jar;...
.........................
05 may 2023  10:33:00.466  <NetPhantom>      <System>    ---               I  SI0047    Creating
server socket: port ID HTTP, port 80 (map to 80), queue length 50, read timeout 30000, accepter
threads 3
05 may 2023  10:33:00.530  <NetPhantom>      <System>    ---               I  SI0037    The Web Server
is started
05 may 2023  10:33:00.596  <NetPhantom>      <System>    ---               I  SI0052    Starting
server socket, port ID: HTTP
05 may 2023  10:33:00.701  <NetPhantom>      <System>    ---               I  SI0010    New
connections from clients are allowed
05 may 2023  10:33:00.701  <NetPhantom>      <System>    ---               I  SI0002    The server is
in a ready state
```

Start a NetPhantom Client or a Browser

A browser can then be used to connect to the address

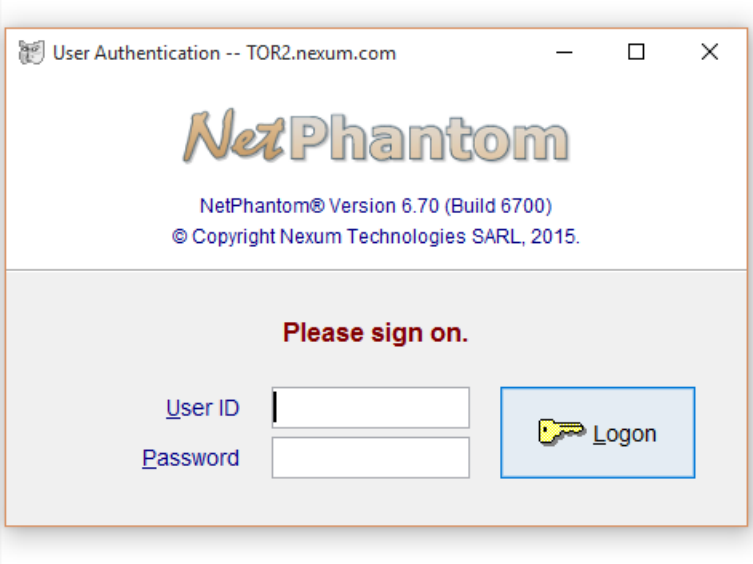
`http://localhost:`*`port`*

or start the NetPhantom Client using the list of ports above.

Note: If the browser is not started on the server, replace `localhost` with the host name or IP address of the server.

The NetPhantom Welcome page is displayed. This page contains links to such things as the Server Administration program, the Java API documentation, the installation program for NetPhantom Starter and several sample applications.

You can now click one of the sample applications links or the Server Administration link to start the NetPhantom Client. If the `server.ini` file is not correctly configured, the Server Administration program will start automatically.



Once the NetPhantom Server Administration Client is connected, log on as user ID ***admin*** with the password ***secret*** to reconfigure the server to your needs.

Removing Initial User Authentication for Server Administration

You can remove the administrator password in the server administration program (initial user *admin*, password *secret*) using the menu item **Server - Configure web server - Resources** tab, select the `SERVERADMIN` application and uncheck the option **Authentication**, then press **OK**.

If you have forgotten the password and have access to the server files, remove the entry `SERVERADMIN` in the `resources.ini` file. This will remove the resource definition of the `SERVERADMIN` application that is set to authenticate users.

4.3 NetPhantom License System

A license is required for running the server. This license is dependent on the number and type of concurrent users, the administration port and the IP address or host name. When the server is started the first time, the license is not valid. The server administration program will start with the first client connection, displaying a dialog box that enables an administrator to fill in the license code. If NetPhantom is used in conjunction with SSL, a license is also required to enable SSL.

Contact your contracting partner or Mindus SARL for information about licensing issues.

See *Chapter NetPhantom License System* for a complete description of how the license system works in NetPhantom.

4.4 Directory Structure

Below is a file listing with directories of all files that are installed with NetPhantom.

Filename	Description
applet/*	Files used by the SmartApplet CGI.
authentication/*	The User Authentication Runtime file and text file.
authentication/Image	The image files for User authentication.
htdocs/*	The HTML documents for the Web Server.

htdocs/APIDocumentation/*	The online API documentation for NetPhantom.
htdocs/DefaultStarter/*	Default NetPhantom Starter files used for a client.
htdocs/DefaultStarterSSL/*	Default NetPhantom Starter (SSL version) files used for a client.
htdocs/docimage/*	Contains the image files used in the NetPhantom HTML pages.
htdocs/hterrors/*	The HTML documents used to display Web Server error messages.
htdocs/info/*	The HTML files for product fixes and news, etc.
htdocs/NetPhantomStarterSetup/*	The NetPhantom Starter installation files.
htdocs/ NetPhantomStarterSSLSetup/*	The NetPhantom Starter (SSL version) installation files.
htdocs/samples/*	Files required for the NetPhantom Samples.
htdocs/source/*	Contains source code for CGIs, the SmartApplet etc.
htdocs/tutor/*	The cascading stylesheet, flash and shockwave files for the fancy tutorial.
htdocs/webapps/*	The .GIF files for disabled checkboxes and radio buttons as well as the messge.html file.
Image/*.gif	Images stored in GIF files.
rconsole/gui/*	The Server Administration program.
samples/*	NetPhantom sample applications.
securelogin/*	The NetPhantom development and runtime files for Secure Login as well as the HTML pages.
BrowserIdentification.ini	The file used to determine which browser a user is running.
footer.nrx header.nrx	The header and footer files used in the REXX-to-NetRexx conversion.
httperror.ini	The ini file describing the error documents used for the Web Server.
mail.jar	The Java archive contains the classes for the mail utility in NetPhantom.
mimetype.ini	The listing of MIME file types for the Web Server.
NetPhantom.map	Sample MAP file for the Default LU Mapper.
NetPhantomClient.jar	The jar file contains all the classes for the NetPhantom Client.
NetPhantomServer.jar	The jar file for the NetPhantom Server, Remote Command Line Utilities, the NetRexxConversion program, etc.

NetPhantomCluster-Controller*.class	The class files used to run the NetPhantom Cluster Controller.
NetRexxR.jar	A jar file for the NetRexx Runtime.
NetRexxC.jar	A jar file for the NetRexx Compiler.
npcdde.dll	A Windows 32-bit DLL used for the NetPhantom Client DDE engine.
openxml-1.2-np.jar	Handles HTML and XML files.
r2nrNT.exe	The program for conversion of REXX source files to NetRexx.
resources.ini	The ini file used for the server resources (files, directories, applications and CGIs).
rules	The rules file for REXX-to-NetRexx conversion.
server.ini	The ini file for the server.
server.phm	The text file for the server.
server.preloadClasses	File indicating what Java classes to load during server start-up.
SmartApplet.ini	Initialization file for the NetPhantom Web Server Smart Applet CGI.
bcprov.jar	Bouncy Castle implementation of JSSE.
startserver.sh startclient.sh startlicensemanager.sh	Sample batch files to start the NetPhantom Server, Client or License Manager on a UNIX shell.
npservice.sh	Using NetPhantom Server as a service under a UNIX system.
startserver.bat startclient.bat startlicensemanager.bat	Sample batch files to start the NetPhantom Server, Client or License Manager on a Windows-based system.
UpgradeNetPhantomServer.class	This class is used to upgrade the NetPhantom server.
users.ini	The ini file where all user groups and individual user definitions are stored.

4.5 Starting the Server

Normally, the NetPhantom Server requires changes in the configuration. For more information, see section *Server Configuration* below.

Change directory to the installation location of NetPhantom. Change the CLASSPATH for the Java environment to include the current directory and `NetPhantomServer.jar, NetRexxR.jar, bcprov.jar, bcpkix.jar, bcutil.jar, acme-np.jar, openxml-1.2-np.jar, activation.jar` and `mail.jar`.

You can check your Java version by typing the following from a command prompt from the directory where the java.exe file is located:

for java:

```
java –version
```

Windows

```
set CLASSPATH=%CLASSPATH%;.
set CLASSPATH=%CLASSPATH%;NetPhantomServer.jar
set CLASSPATH=%CLASSPATH%;NetRexxR.jar
set CLASSPATH=%CLASSPATH%;acme-np.jar
set CLASSPATH=%CLASSPATH%;activation.jar
set CLASSPATH=%CLASSPATH%;mail.jar
set CLASSPATH=%CLASSPATH%;openxml-1.2-np.jar
set CLASSPATH=%CLASSPATH%;bcprov.jar
set CLASSPATH=%CLASSPATH%;bcpkix.jar
set CLASSPATH=%CLASSPATH%;bcutil.jar
```

UNIX

The commands below assume the "normal" UNIX shell (/bin/sh).

```
set CLASSPATH=$CLASSPATH:.
set CLASSPATH=$CLASSPATH:NetPhantomServer.jar
set CLASSPATH=$CLASSPATH:NetRexxR.jar
set CLASSPATH=$CLASSPATH:acme-np.jar
set CLASSPATH=$CLASSPATH:activation.jar
set CLASSPATH=$CLASSPATH:mail.jar
set CLASSPATH=$CLASSPATH:openxml-1.2-np.jar
set CLASSPATH=$CLASSPATH:bcprov.jar
set CLASSPATH=$CLASSPATH:bcpkix.jar
set CLASSPATH=$CLASSPATH:bcutil.jar
export CLASSPATH
```

Executing the Server Start Command

To start executing the server, run the following.

```
java [params] se.entra.phantom.server.Start [restart] [serverIniFile]
```

where `params` for Oracle Java must specify the amount of memory for the Server:

```
-Xms64M -Xmx128M
```

The above parameters will set the initial Java heap size to 64 MB and the maximum Java heap size to 128 MB. These values need to be adjusted depending on the size and number of NetPhantom runtime applications as well as the number of concurrent users.

Each concurrent user requires 500 KB of server memory. Each runtime application requires approximately 10 times the size of the NetPhantom runtime file. If the maximum Java heap size is small, the server may run slowly due to heavy garbage collection.

The optional parameter **`restart`** is used to indicate that the server should support *Hard JVM Restart* from the server administration program. If it is not specified, the server cannot restart with this option.

The **`serverIniFile`** option allows you to specify which name of the ini file that should be used at startup.

Return Codes

The following return codes apply for the server process and are used for batch file execution of the server:

0	Success, stop the server (server start complete).
1	Failure starting or restarting the server.
1000	Perform a server restart.
1001	Perform upgrade of server.

Batch Files for Server Execution

The following sample could be used to run the server under a Windows platform. This sample is also provided in the root of the NetPhantom install directory as the file `startserver.bat`:

```
@echo off
setlocal

rem --- The classpath ---
set
CP=.;NetPhantomServer.jar;NetRexxR.jar;bcprov.jar;bcpkix.jar;bcutil.ja
r;acme-np.jar;openxml-1.2-np.jar;mail.jar;activation.jar

rem --- The memory settings ---
set MEM=-Xms64m -Xmx300m

echo *** Starting the NetPhantom Server ***
goto start

:upgrade
echo *** Upgrading the NetPhantom Server ***
java -classpath . UpgradeNetPhantomServer > upgrade.log
if errorlevel 1 goto uperror

:restart
echo *** Restarting the NetPhantom Server ***

:start
java %MEM% -classpath %CP% se.entra.phantom.server.Start restart %1
if errorlevel 1001 goto upgrade
if errorlevel 1000 goto restart
if errorlevel 1 goto error

echo *** The NetPhantom Server has stopped executing ***
goto done

:uperror
echo *** Error upgrading the NetPhantom Server ***
goto done

:error
echo *** Error when starting or restarting the NetPhantom Server ***

:done
endlocal

pause
exit
```

The following sample could be used to run the server using a "normal" UNIX shell (such as /bin/sh). This sample is also provided in the root of the NetPhantom install directory as the file `startserver.sh`:

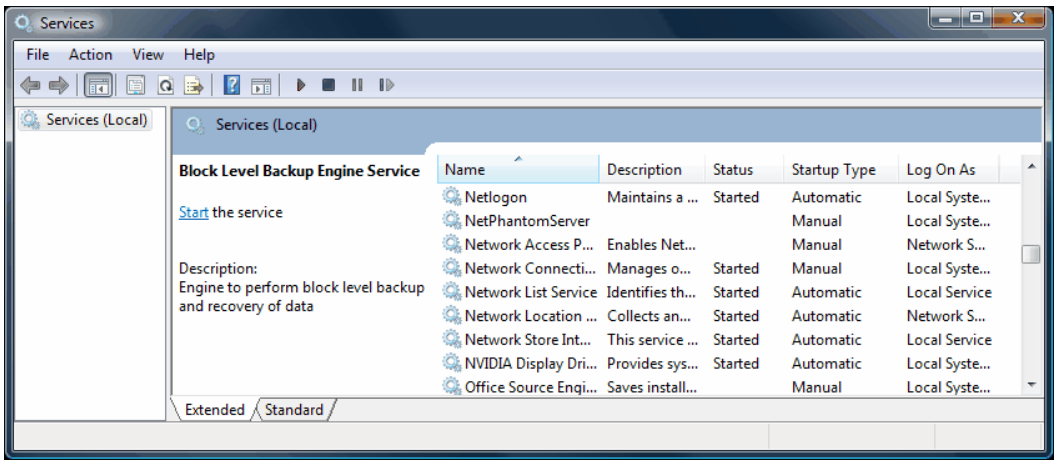
```
#!/bin/sh
# NetPhantom Startup script

COMMAND='java -Xms128m -Xmx128m -classpath
.:NetPhantomServer.jar:NetRexxR.jar:bcprov.jar:bcpkix.jar:bcutil.jar:o
penxml-1.2-np.jar:acme-np.jar:mail.jar:activation.jar
se.entra.phantom.server.Start restart'
RSTCMD='java -classpath . UpgradeNetPhantomServer'

echo "*** Starting the NetPhantom Server ***"

while true
  do
    $COMMAND
    RC=$?
    if [ $RC -eq 1000 ] || [ $RC -eq 232 ]; then
      echo "*** Restarting the NetPhantom Server ***"
    elif [ $RC -eq 1001 ] || [ $RC -eq 233 ]; then
      echo "*** Upgrading the NetPhantom Server ***"
      $RSTCMD > upgrade.log
```

```
        if [ $? != 0 ]; then
          echo "*** Error when upgrading the NetPhantom Server ***"
          exit 1
        fi
        echo "*** Restarting the NetPhantom Server after upgrade ***"
      elif [ $RC -eq 0 ]; then
        echo "*** The Server has been stopped normally ***"
        exit 0
      else
        echo "*** Error when starting or restarting the NetPhantom
Server ***"
        exit 1
      fi
    done
```

4.6 Using NetPhantom Server as a Windows Service

The NetPhantom Server can be run as a Windows Service under Windows 2000/XP/Vista, Windows Server 2003/2008/2012 (R2)/2016/2019/2022 or Windows 7 to 11. Please see the NetPhantom Windows Services for details of how to install the service.

By default, the NetPhantom Server Service requires manual start. To change the setting, Select **Services** from the **Control Panel**. The following dialog box is then displayed:

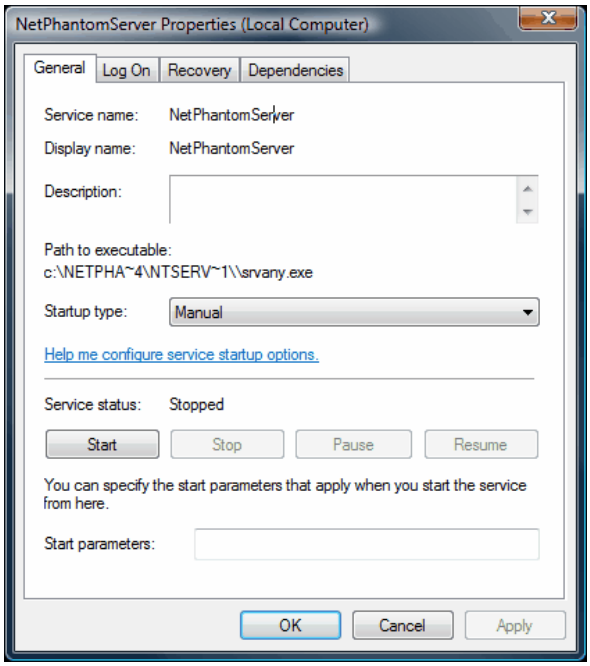


*Select **Startup** or double-click the NetPhantom Server entry.*
This will enter the startup dialog box.

The NetPhantom Server Service does not use any *Startup Parameters*. In the dialog box for **Startup**, choose the *Manual* or *Automatic* startup type.

When *Automatic* startup type is selected, Windows will start NetPhantom using the *administrator* account.

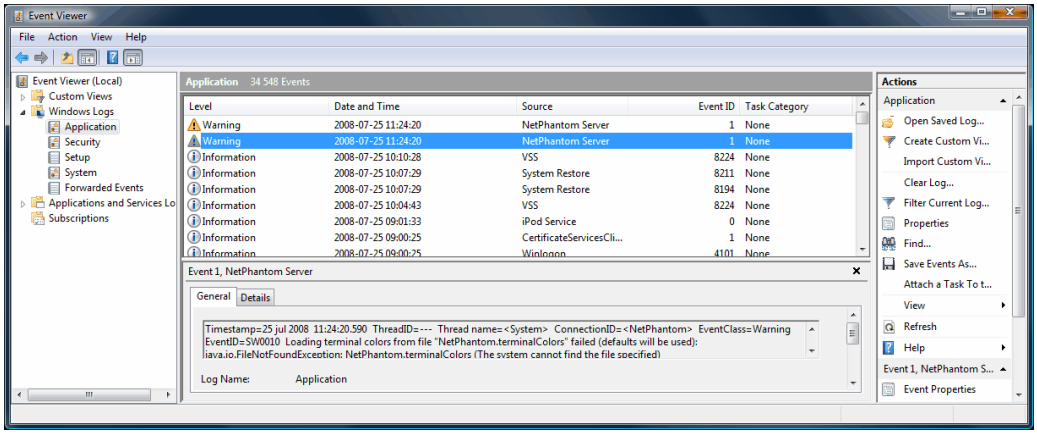
If the service was installed with lower privileges than administrator, you may also choose to use an account (using *This account* and specifying a *password*). Note that the password must be changed in this dialog box if the user password is changed with the Windows user profile.



The Event Viewer

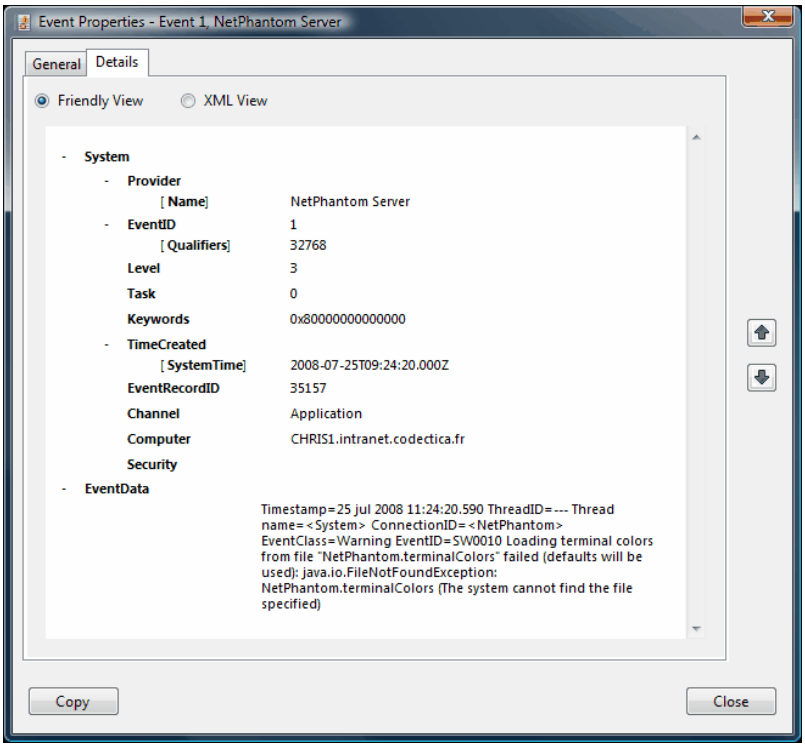
All events that are not filtered out are logged to the NetPhantom log file, but also to the Windows Event Viewer, in the *Application* log. It may be useful to define the appropriate log filtering using the server administration program in order minimize the number of events for the Windows Application Log.

The following window is displayed when the Event Viewer is started from the Administrative Tools.



The Application Log displays the Category and Event for each NetPhantom Server log entry. Double-click on an entry to display extra information that NetPhantom provides.

Each entry in the Application Log contains the same data as provided in the NetPhantom Server event log.



The log entry contains an exact timestamp when the NetPhantom Server first queued the event to send it to the Windows Application Log.

4.7 NetPhantom Windows Services

This feature enables the NetPhantom Server, Cluster Controller, and License Manager to be executed as a Windows Service. This allows the NetPhantom process to run using the latest and best performing Java VM available (optionally using Hotspot Server Just-In-Time with option *–server*). The services created are started with the current directory set to the NetPhantom Server installation root directory (and drive).

The service can be installed several times; the only requirement is that the service name be different. The tools used to do this are included in the Microsoft Resource Kit for Windows and are shipped with NetPhantom for convenience purposes. These tools are compatible with Windows Server 2003/2008/2012 (R2)/2019/2022 or Windows 7 to 11.

Under the server root directory, the subdirectory *ntservice* contains the required files. A batch file (*is.bat*) is used to install or uninstall the service(s), and 3 utilities that belong to the Windows Resource Kit (*Sc.exe*, *Srvany.exe* and *Regini.exe*).

A new command line option *-ntservice* (specified after the start class) enables the following server processes to run as a service:

- NetPhantom Server,
- NetPhantom Cluster Controller,
- NetPhantom License Manager.

This option disables the Logoff signal for the server process that otherwise causes the process to be closed when the console user logs off Windows.

Installing a Windows Service

Note: the shell used must be explicitly run with administrator privileges. This is done by starting the shell by right clicking the icon and selecting “Run as Administrator”.

To install a service, follow the steps below.

1. Start a command prompt and set the current directory to the server installation root.
2. Enter the following command:

```
ntservice\is INSTALL serviceName program javaDir memory
[SERVERVM]
```

where:

serviceName	is the name of the service to use ***WITHOUT spaces or quotes***.
program	can be `LICENSEMANAGER`, `CLUSTERCONTROLLER` or `SERVER`.
javaDir	is the base directory for the installation of the JDK.
memory	is the memory size in *MB* to use for the Java process.
NTEVENTLOG	indicates if the Windows Event log should be used to log events for the service.
SERVERVM	indicates if the HotSpot Server VM should be used (included with the JDK or better and NOT the JRE). This parameter is optional.

This will create a Windows Service that is manually started at Windows boot when the network has been started (service dependencies to *Remote Procedure Call (RPC)* and *TCP/IP Protocol Driver* are added for the service in order to make them work properly when e.g. a server is rebooted).

The service **Startup type** can be changed from *Manual* to *Automatic* by using **Services** from **Administrative Tools** in the **Control Panel**.

Example:

The command

```
ntservice\is INSTALL NetPhantomServer SERVER c:\jdk17 750
NTEVENTLOG SERVERVM
```

will create a service that is run with a command such as

```
c:\jdk17\bin\java.exe -server -Xrs -Xms750m -Xmx750m
se.entra.phantom.server.Start -ntservice
-nteventlog
```

Note the option *-Xrs* that reduces signals from the operating system (required when running as a Windows Service).

Note: Do not use spaces in the *serviceName* and do not enclose it within quotes.

Uninstallation of a Windows Service

To uninstall a service, follow the steps below.

1. Start a command prompt and set the current directory to the server installation root.

2. Enter the following command:

```
ntservice\is UNINSTALL serviceName
```

where:

serviceName Is the name of the service to use **WITHOUT spaces or quotes**.

This will remove a previously created Windows Service of the name ServiceName.

Note: Do not use spaces in the *serviceName* and do not enclose it within quotes.

Registry Keys for a Windows Service

The keys and values below are set when creating the Service *NetPhantomServer* using the command

```
ntservice\is INSTALL NetPhantomServer SERVER
"C:\jdk11" 750 SERVERVM
```

HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\NetPhantomServer:

DependOnService	REG_MULTI_SZ	RpcSs TcpIp
	REG_DWORD	0x00000001
ImagePath	REG_EXPAND_SZ	C:\NetPhantom 7 QS\ntservice\srvany.exe
	REG_SZ	LocalSystem
Start	REG_DWORD	0x00000003
Type	REG_DWORD	0x00000010

HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\NetPhantomServer\Parameters:

AppDirectory	REG_SZ	C:\NetPhantom 7 QS
Application	REG_SZ	C:\jdk11\bin\java.exe
AppParameters	REG_SZ	-Xrs -cp . StartService C:\jdk17\bin\java.exe -Xrs -Xms750m -Xmx750m -classpath .;NetPhantomServer.jar;NetRexxR.jar; bcprov.jar; bcpkix.jar;bcutil.jar;openxml-1.2-np.jar;acme-np.jar;mail.jar;activation.jar se.entra.phantom.server.Start -ntservice

HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\Eventlog\Application\NetPhantom Server:

This key is set when the server starts execution with the service name NetPhantom Server.

EventMessageFile	REG_EXPAND_SZ	C:\NetPhantom 7\WindowsNTEventLog64.dll
TypesSupported	REG_DWORD	0x00000007

4.8 Event Logging to the Windows Event log

The *WindowsNTEventLog.dll* (for 32-bit) or *WindowsNTEventLog64.dll* (for 64-bit) contains the code to do this using JNI (Java Native Interface) in the NetPhantom Java package *windowsnt.eventlog*.

By specifying the option *-nteventlog* (specified after the start class) the following server processes will log events to the Windows Event Log:

- NetPhantom Server,
- NetPhantom Cluster Controller,
- NetPhantom License Manager.

The NetPhantom Cluster Controller and the License Manager logs events with the originator *NetPhantom Cluster Controller* or *NetPhantom License Manager*.

NetPhantom Server normally logs events with the originator *NetPhantom Server* unless the entry `originator=originatorText` in the `[base]` section in *server.ini*. Another option in `[base]` section in *server.ini* is `logInformational={0 | 1}` that can be set to zero (0) or one (1) if informational events should be logged to the Windows Event Log or not.

NetPhantom Service normally logs events with the *NetPhantom Service* unless the entry `originator=originatorText` in the `[base]` section in *service.ini*.

There are two ways to run the NetPhantom Client. The first is to execute it as a Java application, the second is as a Java applet inside a browser.

This chapter will cover how to run the client under different operating systems and browsers.

> **Note:** The recommended method of running the client in a browser environment is to use the *SmartApplet*. See *NetPhantom Client with SmartApplet* below for more information.

4.9 Linux Service/Deamon

The NetPhantom service can be started as a Linux service using the script `npservice.sh` (note: if the service is not run with *root* privileges and ports with numbers lower than 1024 is to be used, please refer to chapter 4.10). Before launching the script, the entry NP_BIN in the script should be edited to point to the server startup script in the NetPhantom installation.

Before using it, set execution rights to the file with the command:

```
chmod 700 npservice.sh
```

The syntax of the script is:

```
npservice.sh start|stop|status|try-restart|restart|force-
  reload|reload|probe
```

```
start: Start daemon with startproc. If this fails the return value is set
       appropriately by startproc.

stop: Stop daemon with killproc.

try-restart|condrestart: Do a restart only if the service was active before.

restart: Stop the service and regardless of whether it was running or not,
         start it again.

force-reload: Signal the daemon to reload its config.

reload: Like force-reload, but if daemon does not support signaling,
        do nothing.

status: Check status with checkproc

     0 - service up and running

     1 - service dead, but /var/run/  pid  file exists

     2 - service dead, but /var/lock/ lock file exists

     3 - service not running (unused)

     4 - service status unknown
```

Note: This script is developed and tested under SuSe Linux. For the use in other distributions, compatibility issues might have to be addressed.

4.10 Linux Port configuration

If the ports with lower numbers than 1024 are to be used and root privileges cannot be applied, the ports need to be remapped under Linux. To do this, a script `npports.sh` is supplied.

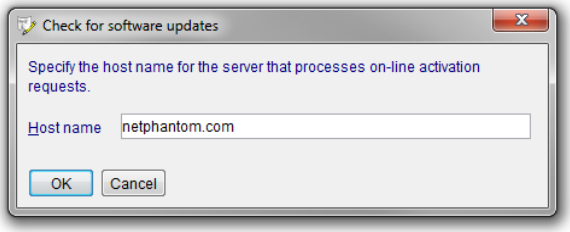
In the script two ports are remapped by default: 8080 → 80 and 8789 → 789. If this is to be changed or amended, the script needs to be changed accordingly. Also note that the NetPhantom port configuration needs to be configured according to which ports are to be remapped (please see chapter 7.3). In the example the HTTP port should be configured to 8080 and the NPCLIENT port should be configured to use 8789.

The syntax of the script is:

```
npports.sh start|stop|status|try-restart|restart|force-
  reload|reload|probe

start: Add ports configuration

stop: Remove port configuration.

try-restart|condrestart|restart|reload|force-reload: restart mapping.
```

4.11 Software Updates

Software updates can be fetched automatically by performing a **Server** – **Check for software updates** command (or **Server** – **Check for software updates** in the NetPhantom Editor). Before the connection to the update server is done, a dialog enables a change of server to connect to, if necessary.



When a request is sent to the update server, information of the current configuration of the system is transferred. Based on this, the list of relevant patches is calculated and displayed on a web page.

The list contains the date when the patch was issued, a link to the downloadable patch file and a link to the description of the patch (which problems were addressed etc.). For the NetPhantom Editor the patch file is an executable installation file. For this type of installation, the editor must be stopped, and the file launched manually.

5 Running the NetPhantom Client

5.1 NetPhantom Client Parameters

The parameters used to start the NetPhantom Client are common to the sections *Standalone Application* and *Inside a Browser* below.

```
java –classpath NetPhantomClient.jar
  se.entra.phantom.client.Phantom app:serverAdmin
```

Parameter name	Description
HOST	The host name or the IP address of the NetPhantom Server. This is normally not specified for a NetPhantom Client running inside the browser, because it is defaulted to be the host name of the Web Server. For a standalone application, `localhost` is assumed. There may be a list of host names for all backup servers.
PORT	The port number(s) of the NetPhantom Server(s). If the `port` parameter is not specified, the default port number 789 is assumed. There may be several ports specified with this parameter to define the list of backup servers.
APP	The application ID(s) to run. These applications are defined and loaded in the NetPhantom Server using the `server.ini` file. See the section *Specifying Runtime Application IDs* below for more information.
HOSTID	The host session ID to connect to. This value can be `A-Z` or `'.'` (Period) for no host connection. The host session ID must be defined in the `server.ini` file.
RESURL	Optional setting that defines from where resources such as images, other class files (in e. g. user windows) and help documents are loaded. Include the transport protocol and optionally a port number, e.g. `http://yourserver.com:8080/Image`. This setting may be very useful for NetPhantom Clients that use SSL as all images will be loaded without SSL, thus loading them much faster. The setting is also useful for a NetPhantom Client running with NetPhantom Starter, because the resources (images, class files, etc.) do not then need to be installed locally.
RESJAR	The option `RESJAR` enables image resources to be loaded from JAR files instead of over e.g. HTTP. Specify `RESJAR:file1.jar:file2.jar`, etc when running the Client as a Java Application. If the `RESJAR` parameter is not specified, the default `NetPhantomClientImages.jar` is assumed.
OUTSIDE	Specifies if the panels that are created should be placed inside the browser (`=0`) or outside (`=1`). This parameter does not apply when running the client as Standalone application.
DDE	Specifies the name to use for DDE connections. The default is "PHANTOM". This parameter must be limited to 8 characters for compatibility reasons.

HTTP	Specifies if HTTP tunneling is used to communicate with the server. See the section *HTTP Tunneling and the NetPhantom Web Server* for more information. Set `HTTP=1` to activate this option or `HTTP=0` (the default) to disable it.
SSL	Specifies the class name of the SSL implementation on the NetPhantom Client. When this parameter is not specified, SSL is disabled. See *Chapter SSL – Secure Socket Layer* for more information.
USERVAR	Lists all user variables that should be transferred to the server to *Global Variables*. See *User Variables* below for more information.
DRAG	Sets the type of window dragging effect. 1 to set "faster" window drag inside browser or MDI (default for Windows). For slower client CPUs set `DRAG=2`. This specifies outline window drag (default for Motif) and solves problems of "choppy" drawing when windows are dragged.
UNICODE	Set `UNICODE=1` to set Unicode for all string/character transactions to the server.
PROXY	Host name and port number of the (authenticating) proxy with a colon (":") separator. Optionally, this text is preceded by "`*SOCKS:`" to enable proxy support for SOCKS. For browser configurations, the `PROXY:*SYSTEM` is advised as this will try to get the current proxy browser setting from (in priority order): 1. the system definition. 2. the java definition (control panel – note: only the specific proxy definition will be used, script reference is ignored). To turn off proxy processing, specify `PROXY:*DIRECT`.
TYPEAH	All keystrokes during a panel session lock are played back when the session unlocks. To enable type-ahead, specify the client option `TYPEAH:1`.
PRTDLG	The print dialog box is displayed once only for each printing type (host print, report printing, print window, etc) when the client parameter `PRTDLG:1` is specified.
EDIT	Enhanced edit capabilities in NetPhantom Java Client with support for undo/redo and pop-up menus. To enable the pop-up menu support, specify `EDIT:1`. To enable both pop-up menu support as well as undo/redo functionality, set `EDIT:2` (default). To disable this functionality, specify `EDIT:0`
DRAGDROP	Drag-and-drop is enabled in panels by default. Entry field controls (entry field, multiple line entry field, combination box and spin button) support both drag and drop, while list boxes only support drag operations. If a panel has a rectangle marking as described above, dragging the selection with mouse to another application will transfer both the panel text and the image. If only the image is required, press the Control key, and hold it down while performing the drag operation. To disable drag-and-drop for the entry fields and list boxes, set the client parameter `DRAGDROP:0`.

POPMENU	The default behavior is that standard menu items (cut, paste…) are put in the beginning of the menu. This order can be reversed by setting the client parameter `POPMENU:1`.
VERBOSE	Use verbose logging of events on the client. This log is available in the Java console window on the client computer.
DRAGGABLE	`true` indicates that the applet should be possible to drag (using Control + Left Mouse button) outside the browser. In doing so, the application will still be running and a shortcut for launching it later is installed on the desktop.
DRAGICON	Indicate to the user that the applet in question is possible to "drag-install" ad described before.
JNLP_REF	A reference to the jnlp file to be installed. Note: this parameter is only meaningful if the `DRAGGABLE` parameter is set to `true`.
JAVA_ARGUMENTS	Forward java runtime arguments to the java virtual machine running the applet. For example, `-Xms24m -Xmx256m` will specify the memory allocation.
PRTV3	For the print-window function (Ctrl+P), the standard Windows printing (as it was in NetPhantom versions prior to 5) dialog can be used. This is for example to enable that printer settings are kept between client sessions.
LAF	Sets the client Look-and-Feel. Specify the name of the Java class or one of the names Mac, Metal, Nimbus or Windows, e.g. `LAF:Nimbus`.
SAVESETTINGS:*n* where *n* = 0 or 1 (zero or one)	`SAVESETTINGS:0` disables creation of the Client-local properties typically created when the user reconfigures the Terminal settings. This new option will effectively skip reading an already present properties file as well as the writing of a one, thus letting the Client settings be taken from the Server every time.
F10:*n* and F22:*n* where *n* = 0 or 1 (zero or one)	Certain host applications use F10 and F22 as commands in combination with a possible pass-through of the function keys. The function keys F10 and Shift+F10 are reserved to activate the menu bar and a context menu respectively. In Windows, you can also activate the menu bar by pressing the Alt key and releasing it, and often the keyboards provide a special context-menu button. The F10 and Shift+F10 could therefore be used for the host application as users are used to when working with the host terminal application. Two new Client options are added, `F10:0` and `F22:0`, to turn off F10 and Shift+F10 respective menu and pop-up menu activation.

FONT*:type[:value]*	The fonts can be scaled option can be specified as: `DPISCALE[:dpi]` Scales the font as if the DPI from the system was set to *`dpi`*. `DPIFACTOR[:dpi]` Sizes the font point sizes to integer-rounded values according to the *`dpi`* value. `SCALE[:percent]` Scales all fonts using the Java *"Font.derive"* with the percent value (without % sign). `FACTOR[:percent]` Sizes the font sizes to integer-rounded values with the percent value (without % sign) . Unspecified `percent` or `dpi` reads the setting from the system.
SCALEIMAGES	`SCALEIMAGES:0` disables the automatic image scaling that is relative to the font scaling.
@filename	The *`filename`* is loaded using the "UTF-8" character set, and all its lines are used as parameters as they are, i.e. with empty lines, indentation, spaces, line endings, tabs, etc. The parameters loaded from the file will therefore replace the *`@filename`* parameter itself.
BGCLR:*#rrggbb*	Only supported in the Windows (XP and Classic) look-and-feels. The parameter is specified as a six-digit HEX color prefixed with '#' much like in HTML. An example to set the background color to darker gray is `BGCLR:#e0e0e0` as the default Windows background color is `#f0f0f0` (and sometimes not dark enough).
NOBEEP*:option1:... :optionN*	Option to disable "beeps" with the options below. Multiple options can be specified, separated by a colon (':'), e.g. *NOBEEP:HOST:HOST_ERR:TERMINAL*. The possible options for disabling beeps are: `*` All beeps. `MSGBOX` Message box shown with non-question type. `INVALID` Invalid operation. `MOUSE` Invalid mouse operation such as click, drag. `HOST` 3270/5250 host induced terminal alarm beeps. `HOST_ERR` Alarm for server error in 3270/5250 session. `TERMINAL` Invalid operation in 3270/5250 terminal. `SERVER` Server-side, API-induced from REXX, Java or System.

Port/Host and Backup Servers

The client parameters PORT and HOST can list a series of servers to use. There are three methods to do this but *running the client in a browser only supports method 1*.

1. `HOST` is not defined for applets, otherwise `HOST=serverAddressOrName`. The `PORT` parameter is a list of port IDs, e.g.

   ```
   <param name=port value="789,8080,28080">
   ```

   or

   ```
   port:789:8080:28080.
   ```

2. Multiple hosts, same port:

```
HOST:addr1:addr2:addr3... PORT:28080
```

3. Individual host-port settings:

```
HOST:addr1:addr2:addr3... PORT:port1:port2:port3...
```

   The server 1 will use `addr1:port1`, server 2 `addr2:port2`, etc. The list of host addresses and port numbers must be equal in argument count.

Specifying Runtime Application IDs

The application syntax is as follows for the standalone Java application:

```
app:[!]appID1[,appID2,...]
```

or in the HTML applet as a parameter:

```
<param name = "APP" value = "[!]appID1[,appID2,…]">
```

Merge-on-the-fly is an option that enables a user to run several applications at the same time. It will begin by searching for the first application for an identity that matches the host screen. When NetPhantom comes to a host screen without a match in the first application, it will begin searching the next application until a match is found, and so on. Because the applications are not actually merged into one, there is no problem with matching screen identities.

Start search for matching screen from first application
This option will cause NetPhantom to always begin searching for matching screens in the first runtime file when merge-on-the-fly is used. Normally, NetPhantom "stays within the same" runtime file until no host screen matches. It then starts searching from the first file.

Each client may run any combination of the loaded applications on the server with the Merge-on-the-fly technique as described above. To use the setting *"Start search for matching screen from first application"*, specify an exclamation mark before the first application.

User Variables

The NetPhantom Client can transfer user variables to the NetPhantom Server at startup. This is typically used when some kind of parameter-defined HTML exists, and initial processing needs to be done in the NetPhantom Server in e.g. REXX-converted code. An example of this is when integrating a web-based application with NetPhantom.

The user variables to be transferred should be listed in the USERVAR parameter with the following syntax:

```
USERVAR:var1[,var2,…]
$VAR1:variableData1
$VAR2:variableData2
```

for Java standalone application or as follows in the HTML applet as a parameter:

```
<param name = "USERVAR" value = "VAR1[,VAR2,…]">
<param name = "$VAR1"   value = "variableData1">
<param name = "$VAR2"   value = "variableData2">
```

The variables, `$VAR1`, `$VAR2`, etc., above will be defined in the *Global Variables* in the NetPhantom Server and can then be accessed from REXX-converted code by the *GlobVar* functions.

5.2 Standalone Application

The client can be executed in the following way:

```
java –classpath NetPhantomClient.jar
   se.entra.phantom.client.Phantom
   host:hostname
   [ app:[!]appID1[,appID2,...] ]
   [ hostid:id ]
   [ http:boolean ]
   [ port:port1[,port2,…] ]
   [ ssl:className ]
   [ …userVariablesAsAbove… ]
```

5.3 Inside a Browser

To run NetPhantom inside a browser, an HTML file must be created and placed on a Web Server for access by the Web browser. See *NetPhantom Client with SmartApplet.*

5.4 NetPhantom Client with SmartApplet

SmartApplet is a new HTML "tag" that simplifies the implementation of the NetPhantom Client HTML document. It is included in the NetPhantom Web Server by default (the NetPhantom Web Server must be started).

SmartApplet is an HTML-included CGI or servlet that checks the client's browser.

The CGI is server-based and uses the "User Agent" string (the browser name, version, etc.) provided for all browsers. The configuration file `SmartApplet.ini` is loaded using settings in the section `BrowserSelection`. All items in this section ending with `.ID` are assumed to be definitions of the browser string using Windows-like wild cards (* for any character, ? for a single character, ^ to escape the meaning of next character e.g. ^* means a '*' must match).

When a match is found for an item, the same item's name, but ending with `.FILE` is used to replace the SmartApplet tag.

The SmartApplet Definition File – SmartApplet.ini

By default, the SmartApplet ini file contains:

```
[BrowserSelection]
; File name for no browser match.
sorry          = applet/sorry.html

; IE6+ under Windows
ie.id          = Mozilla/4.0 (*MSIE*)
ie.file        = applet/ns6.html

; Netscape 4 under Windows
netscape4.id   = Mozilla/?.*
(*Win*),!*MSIE*,!*Netscape/6*,!*Netscape/7*,!*Netscape/8*
netscape4.file = applet/ns6.html

; Solaris w/Navigator 4. -- e.g. Mozilla/4.02 [en] (X11; l; SunOS 5.6 sun4u)
solaris.id     = Mozilla/4.* (*X11*)
solaris.file   = applet/ns6.html

; Mozilla 5+ (Netscape 6+, Mozilla Firefox, Safari, ...)
mozilla5.id    = Mozilla/5*
mozilla5.file  = applet/ns6.html

mozilla6.id    = Mozilla/6*
mozilla6.file  = applet/ns6.html

; Opera
opera.id       = Opera*
```

```
opera.file     = applet/ns6.html
```

The applet tag using SmartApplet that is for example:

```
      <%@include cgi = "smartapplet"
         width       = 100%
         height      = 100%
         port        = 8080
         app         = "app1,app2"
         hostid      = "B">
```

The content of the `applet/ns6.html` file is for example:

```
<body bgcolor="@#BGCOLOR@" style="margin:0px;overflow-x:hidden;overflow-
y:hidden;overflow:auto;"><applet
  width="@#APPLETWIDTH@"
  height="@#APPLETHEIGHT@"
  code="@#STARTCLASS@"
  archive="/NetPhantomClient.jar@#__NP__CLIENTJARS@"
  name="@#APPLETID@"
  alt="The NetPhantom Client Applet cannot run because the browser can't run
Java Applets">
 <param name=APP value="@#APP@">
 <param name=HOST value="@#HOST@">
 <param name=PORT value="@#PORT@">
 <param name=PROXY value="@#PROXY@">
 <param name=TYPEAH value="@#TYPEAH@">
 <param name=EDIT value="@#EDIT@">
 <param name=DRAGDROP value="@#DRAGDROP@">
 <param name=HOSTID value="@#HOSTID@">
 <param name=LANG value="@#LANG@">
 <param name=RESURL value="@#RESURL@">
 <param name=RESJAR value="@#RESJAR@">
 <param name=OUTSIDE value="@#OUTSIDE@">
 <param name=DRAG value="@#DRAG@">
 <param name=PRTDLG value="@#PRTDLG@">
 <param name=DDE value="@#DDE@">
 <param name=HTTP value="@#HTTP@">
 <param name=SSL value="@**SSLLEVEL1@">
 <param name=APPBGCOLOR value="@#APPBGCOLOR@">
 <param name=PRTV3 value="@#PRTV3@">
 <param name=POPMENU value="@#POPMENU@">
 <param name=VERBOSE value="@#VERBOSE@">
 <param name=DRAGGABLE value="@#DRAGGABLE@">
 <param name=JNLP_REF value="@#JNLP_REF@">
 <param name=JAVA_ARGUMENTS value="@#JAVA_ARGUMENTS@">
 <param name=DRAGICON value="@#DRAGICON@">
 <param name=WS value="@#WS@">
 <param name=USERVAR value="SECURELOGIN,PSB.INI.DIR,@#USERVAR@">
 <param name="$PSB.INI.DIR" value="@#PSB.INI.DIR@">
 <param name="$SECURELOGIN" value="@#SECURELOGIN@">
Your browser doesn't understand the Applet tag, thus the NetPhantom Client
Java Applet can't run!
</applet></body>
```

SmartApplet also supports using another section in the `SmartApplet.ini` file instead of the default section `BrowserSelection`. Use the CGI parameter `section = "SectionName"`.

For more information about NetPhantom HTML Variables, see *Chapter NetPhantom Web Server*.

For a listing of the SmartApplet Java Code, see the *Source Listing* link on the of the NetPhantom Server index page.

5.5 The draggable applet

Modern Java technology allows for applications running as applets in a browser to be extracted and run as “normal” applications. As an extension of this concept, the NetPhantom system allows for “one-click installation” of applet applications.

The draggable applet icon.

The draggable quality is configured in the publish application part of the server administration interface (see chapter 16.22). To indicate that an application has a draggable applet client, an icon can be included on the on the applet background as shown above.

An applet is extracted from the browser by pressing Control, Left Mouse button and selecting a frame in the applet area (or any part of it for an application panel client) and subsequently dragging the applet out of the browser. When the applet is released, three things will happen:

- The applet background will disappear, leaving the application running as a normal java application.
- The application is installed on the desktop.
- The browser will show a message indicating that the application is now installed.

Apart from the removed applet background, the application state is maintained from before the drag started. When the application is re-launched using the desktop shortcut, it is run exactly like configured for the Java web start option in the application publication definition (see chapter 16.22).

5.6 The Application INI file

Support has been added that allows you to set the following per application: client properties, bitmaps/background color for the application area in NetPhantom Session Booster (PSB), tooltip text for each application component, a private directory of the application class loader, and to hide unauthorized menu items in PSB.

To use the new features described in detail below, a new INI file is required for the application. It should be stored in the same directory as the runtime file (.NPR), but with the file extension ".INI". If you have an application called `MYAPP.NPR` in the directory `somewhere/myapp`, the INI file should be named `MYAPP.INI` (note: in upper case) in the same directory. The text file should be saved in ISO-8859-1 codepage (e.g. the same codepage as for "normal" HTML pages [notepad], i.e. not in EBCDIC).

Example of an INI file for an application:

```
[base]
something=0

[clientProperties]
someProperty=X
another=Y
```

Client Properties for an Application

Client properties can now be defined in the Application INI file (as described above), in the `[clientProperties]` section.

These properties will be transferred to the client upon initialization of the runtime file and can be retrieved with the Java API function `SessionManager.getProperty`. In cases where multiple runtime files are used ("merge on-the-fly"), the initialization of the runtime file depends on the matching host screen and can sometimes be done in a later stage, thus not directly when the client connects to the server.

Bitmap in Application Area

The application area attributes (where panels are displayed when running inside an application panel or a browser) can now be specified.

In the application.INI file, specify the entry `desktop.image=`*`FileName`* and `desktop.color` in the `[clientProperties]` section as described below.

The desktop.image setting uses two parameters x and p just in front of the file name. These parameters indicate:

- **x** how the bitmap should be stretched, 0=no stretching, 1=stretch but keep the aspect/ratio, 2=stretch to fit in desktop area.
- The second parameter **y** indicates the position of the bitmap when not stretched to fit in the desktop area with a number position similar to the numeric keypad on the keyboard, i.e. 7=top-left, 8=top-middle, 9=top-right, 4=left-center, 5=center, 6=right-center, 1=bottom-left, 2=bottom-center, 3=bottom-right.

The *FileName* parameter should be a file relative to the `RESURL` of the client (e.g. "Image/myback.gif") or a specification of how and where to load the image (e.g. "http://myserver/Image/myback.gif").

When many runtime files are used ("merge on-the-fly") and one bitmap has been set for the desktop area and a new runtime file becomes initialized, the bitmap will remain unless the new runtime file has specified a new image in the Application INI file (or an empty bitmap).

Background Color in Application Area

Like the bitmap specification, a background color can be set for the application area when a bitmap that is not stretched to fit the panel is not present.

Specify the entry `desktop.color=`*`color`* in the `[clientProperties]` section in the application INI file. This setting must be an RGB (Red-Green-Blue) value in hexadecimal format, e.g. "0xC0C0C0" for dark gray, or a name of a system property (that specifies the color).

For example, the setting `desktop.color=0x005C5C` could be a desktop background color.

The bitmap and color properties can also be specified as parameters to the NetPhantom Client, when running as a Java Application:

```
java se.entra.phantom.client.Phantom
     desktop.image:05http://myserver/image.gif
     desktop.color:0x005C5C
```

or when running as an Applet:

```
<applet width="100%" height="100%"
        code="se.entra.phantom.client.Phantom"
        archive="/NetPhantomClient.jar">
  <param name="desktop.image" value="/image.gif">
  <param name="desktop.color" value="0x005C5C">
</applet>
```

Tooltip Text

An application can have tooltip text for every component supporting this in an application, e.g. the pull-down menus, menu items and push buttons.

The tooltip text should be either a simple text string (e.g. "Select this item to perform something") or in HTML format (e.g. "<html>This is a longer text that
can be written
on multiple lines").

On the server side, a Java API is available ("setTooltipText" in `VirtualPullDownMenu`, `VirtualMenuItem` and `VirtualControl` or "setProperty" in `VirtualInterface`).

For a runtime application, a predefined tooltip text can be specified using the Application INI file setting `tooltipTextFile=`*`filename`* in the `[base]` section, for example:

```
[base]
tooltipTextFile=MYAPP.TT
```

The file "MYAPP.TT" should be in the format described below (in the ISO-8859-1 codepage). Note that `PANELID` and `CONTROLID` are separated by a dash (-).

```
; Comment line begins with a semicolon, but
; comment line(s) are not allowed directly
; after a "PANELID-CONTROLID" line.
; Empty lines are skipped.
PANELID-CONTROLID:
Text that will be displayed on a single line
PANELID-CONTROLID:
<html>A text in HTML format that can be displayed over
multiple lines and MUST end with </html>
PANELID-CONTROLID:
*html-filename
```

The *`html-filename`* specifies an external file (relative to the runtime file) and must be a valid HTML file (in the ISO-8859-1 codepage).

The `CONTROLID` can also be set to `*MENU`***x*** where ***x*** specifies the pull-down menu number (as main pull-down menus do not have a control ID).

The specialication could be `-CONTROLID` and this will place the tooltip on all panels with a `CONTROLID` (also menus, submenus, and pop-up menu items).

Tooltip texts are displayed after a short while when the mouse pointer is moved over a component. However, a longer text may be specified for menu items, and the end-user can sometimes not be able to read all the contents of the tooltip. Special settings for how tooltip texts are displayed for menu items are therefore available and should be specified in the `[clientProperties]` section of the Application INI file, see below.

The Java/Swing ToolTipManager contains numerous properties for configuring how long it will take for the tooltips to become visible, and how long until they are hidden. Consider a panel that has different tooltip texts based on where the mouse is. When the mouse moves into the panel and over a region that has a valid tooltip, the tooltip will become visible after `initialDelay` milliseconds. After `dismissDelay` milliseconds the tooltip will be hidden. If the mouse is over a region that has a valid tooltip, and the tooltip is currently visible, when the mouse moves to a region that doesn't have a valid tooltip the tooltip will be hidden. If the mouse then moves back into a region that has a valid tooltip within `reshowDelay` milliseconds, the tooltip will immediately be shown, otherwise the tooltip will be shown again after `initialDelay` milliseconds.

The following settings are available for displaying tooltip texts for menu items:

`tooltip.initialDelay`	The number of milliseconds to delay (after the mouse pointer has paused) before displaying the tooltip.
`tooltip.dismissDelay`	The number of milliseconds to delay before taking away the tooltip.
`tooltip.reshowDelay`	Used to specify the amount of time before the user has to wait initialDelay milliseconds before a tooltip will be shown. That is, if the tooltip is hidden, and the user moves into a region of the same component that has a valid tooltip within reshowDelay milliseconds the tooltip will immediately be shown. Otherwise, if the user moves into a region with a valid tooltip after reshowDelay milliseconds, the user will have to wait an additional initialDelay milliseconds before the tooltip is shown again.

Example MYAPP.TT file:

```
ADMIN-*MENU1:
<html>
<body>Provides file transfer capabilities between
server and client and also to quit the application.
</body>
</html>

ADMIN-*MENU2:
<html>
<body>The following topics are handled here:
<ol>
<li>Licensing,
<li>Configuration of the server,
<li>Access management (users and groups),
<li>Service functions,
<li>Upgrade functions.
</ol>
</body>
</html>

ADMIN-*MENU3:
Handles enabling and reloading of defined and loaded
applications

ADMIN-*MENU4:
Performs client related tasks

ADMIN-*MENU5:
Performs server auditing, tracing and event logging tasks

ADMIN-*MENU6:
The About menu

ADMIN-FILEMAN:
One of the most complex panels on the client side!

ADMIN-ABOUT:
The most common panel in all applications!
```

Hiding Unauthorized Menu Items in NetPhantom Session Booster

Menu items that a user is not allowed to access are normally disabled in NetPhantom Session Booster. Support to hide them rather than to disable them is now added for NetPhantom. To activate this new setting, specify `hideUnauthorizedMenuItems=1` in PSB.INI.

5.7 Using Dynamic Data Exchange – DDE

The NetPhantom Client includes support for DDE in the same manner as previous versions. This support includes interaction with the *Global Variables* or with *local code* written in REXX (for NetPhantom this code is converted into Java).

DDE support is available only under Windows running as a *Java Application*.

Running NetPhantom Client with DDE as a Java Application

The NetPhantom Client support for DDE is done using a Win32 Dynamic Link Library (DLL) called *NPCDDE.DLL*. This DLL is called from Java using the Java Native Interface (JNI). The requirement of the Java Virtual Machine is merely JNI.

When running the client with DDE, this DLL must be in the current directory or in the system path (e.g. \Windows\SYSTEM32 for a normal Windows installation).

This DLL may also be distributed to the client using **NetPhantom Starter**. For further information, see *Chapter NetPhantom Starter.*

6 Running the NetPhantom Editor

The NetPhantom Editor is an integral part of the NetPhantom system. To launch the Editor, open the NetPhantom folder on the desktop or in the start menu and select the Editor icon.

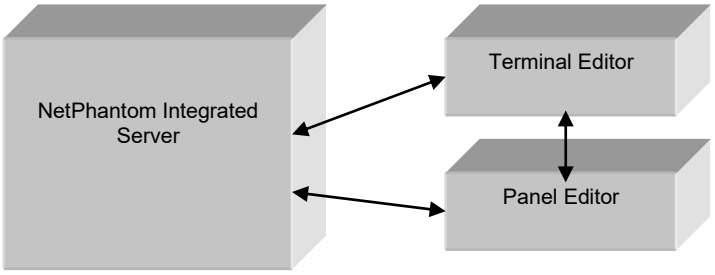
The NetPhantom Editor icon.

The Editor uses the same license system as the rest of the system but requires a specific license code to run. When launching the Editor for the first time, the user will be asked to supply a license code as described in chapter License Configuration.

This chapter contains an overview of the NetPhantom and its functions.

6.1 Overview

The NetPhantom development environment consists of three major parts:



The basic architecture of the NetPhantom Editor

The two visible parts of the development environment are the Terminal and the Panel Editors. The Integrated Server runs as a background process.

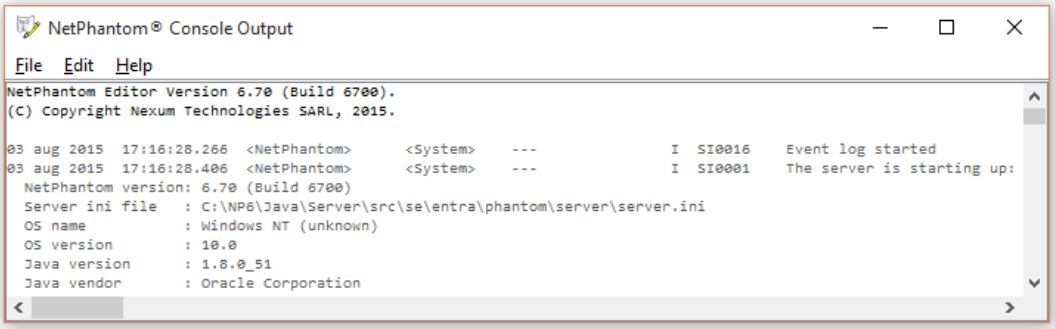
In addition to specifics (described below), the editor parts handle projects. A project can be seen as the development aspect of an application. That means that all the components that are relevant to the application are specified, and their relations are defined. How to define a project is described in chapter Building an Application.

Building

Once a project has been defined, it can be built. This means that references are verified and compilable objects are built. The build can either be launched explicitly (menu item **File – Build Project**) or it can be run in the background, launched automatically (menu item **File – Build Automatically**) when a change in the source components requires it.

The compilable items can all be marked as needing re-building by selecting the menu item **File – Clean…**

The output of the build process can be seen in the Java Console.



In addition to the build output, the full server log of the integrated server is available in the Java Console.

The Java Console can be opened by selecting the menu File – Java Console or by pressing the Java Console icon.

Eclipse integration

If Eclipse integration is available, the corresponding Eclipse instance can be opened and brought to front, by pressing the Eclipse Icon.

Execution

The application that corresponds to the current project can be test run during development. The icons for controlling the execution are available in the File menu and in the top tool bar.

The icons to start, start debug (in debug mode) and stop execution respectively.

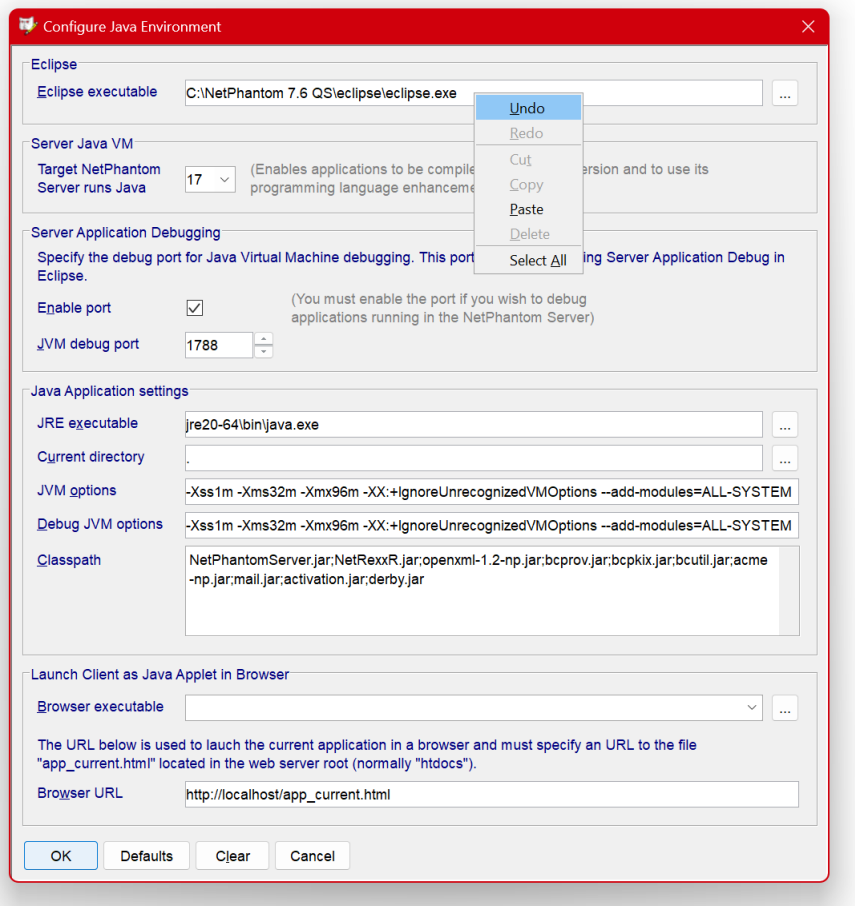
The application can be executed in normal mode or in debug mode. The configuration of the corresponding java environments is described below. The application can also be launched as an applet in a browser. This alternative, together with the other ways of launching the application, are available under menu File - Run

Distribution compilation

When a project is consistent, meaning that it contains no unresolved references or build errors, it can be compiled into a distribution. This is the actual application artifact. When finished, this can then be deployed on a NetPhantom server. The compilation of the current project is started by selecting **File – Compile distribution** for NetPhantom 7 Servers.

Java Runtime Environment

The NetPhantom development environment is heavily dependent on different java runtime environments. These are configured by selecting **Options – Configure Java environment**.

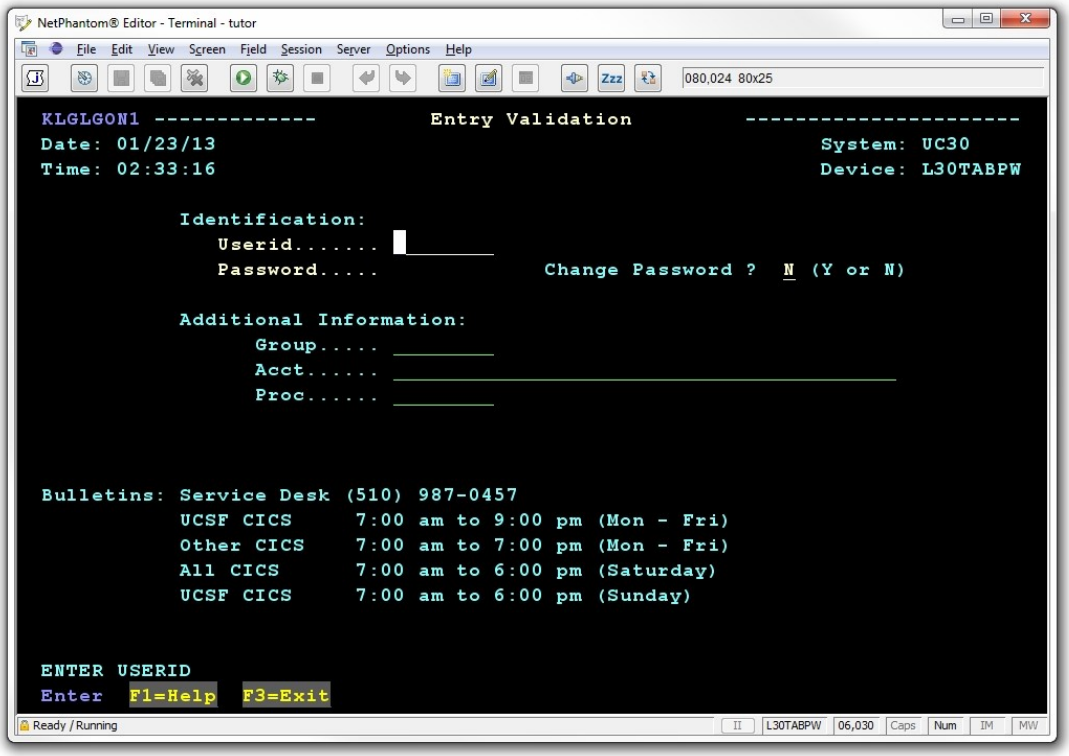


Java Environment configuration.

Eclipse executable	If Eclipse integration is specified, the (full, absolute) path of the installation is specified.
Target NetPhantom Server runs Java	Select this option to enable the compilation of Java code with Java 8 compliance or higher.
Enable port	Enable a port to make remote debugging of a running application possible.
JVM debug port	The port to use for remote debugging.
JRE executable	The java runtime executable that should be used when running the development environment (full, absolute path).
Browser executable	The browser to use when the application is launched as an applet. If this field is left blank, the system default browser will be used.
Current directory	Which directory should be used as current, i.e. if relative paths are used in the environment, what directory should it be relative to.
JVM options	Options used when starting a separate process for executing a test application.
Debug JVM options	Options used when starting a separate process for executing a test application in debug mode.

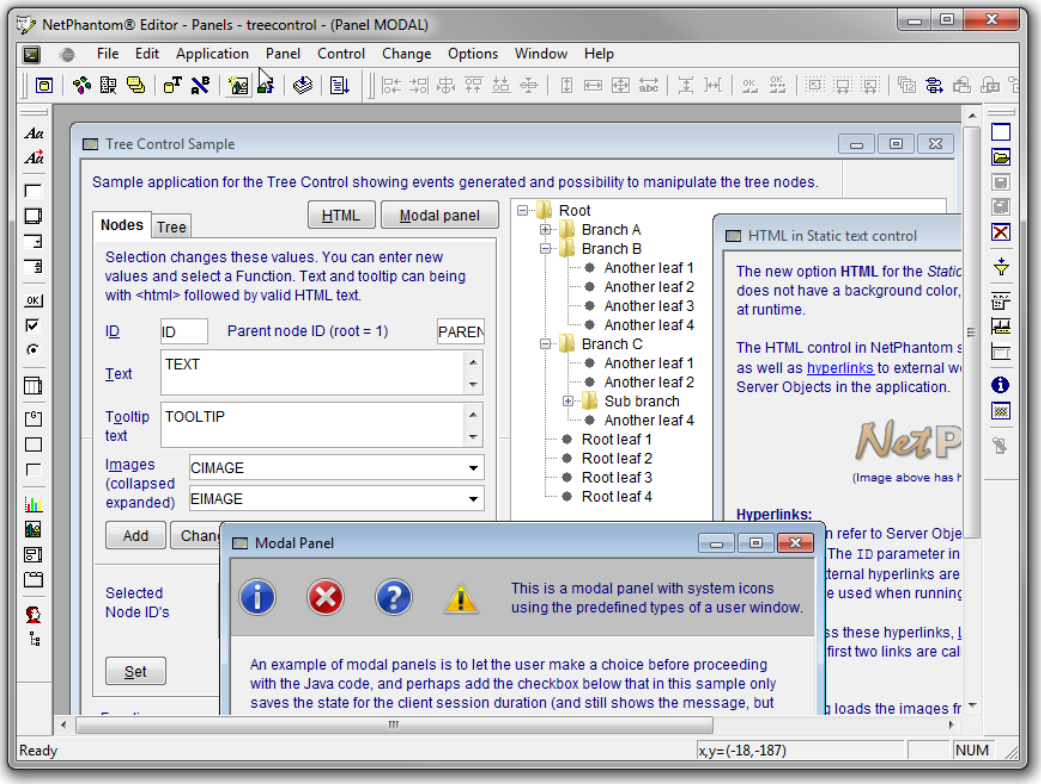
Classpath	Class path to be used by both the java compiler and the runtime environment.
Browser executable	Select one of the browsers detected from the combobox (Internet Explorer, Firefox, Chrome, Opera or Safari), or specify the executable directly. If left empty, the system default browser will be used.
Browser URL	Specify the URL for the browser to start the application. This must point to the file "app_current.html" to the machine and optional port to use. Defaults are "localhost" and the port is left unspecified which will use port 80.

6.2 Terminal Editor



The Terminal Editor handles the definition of host screens. The details of how this is done are described in the *Phantom Hurricane – User's Guide and Reference* document in *Chapter 6*.

6.3 Panel Editor



In the Panel Editor the client interface is defined, and the host screens are connected to this interface. The details of how this is done can be located in the Phantom Hurricane – User's Guide and Reference document in chapter 7.

Panel setting – Old and New style

The panels in the Editor are now rendered with the Java components as opposed to previous Editor versions using Win32 components. The Win32 components for entry field, combination box and spin buttons used "insets" so that pixel alignment of the control's borders was difficult.

The panel setting *Old style* controls this behavior. The following rules apply to *Old style*:

- entry fields are outset by 3 pixels in each direction,
- spin buttons are one font (the control font, not the panel font) in height plus 3 pixels above and below,
- combination boxes are sizes with their drop-down list, but the entry field part is one (control) font height plus 2 to 3 pixels above and below.

Control size and position is thus font size relative with a mix of pixel values, which is why alignment is difficult, because the panel coordinate system is based on dialog box coordinates, i.e. X is 1/4 in panel font width and Y 1/8 of panel font height.

The *New style* for panels treats the above controls entry field part equally with the exact bounds selected. Combination boxes define the size of the drop-down list in the settings dialog of the control. With this *New style* setting, it is easier to align controls.

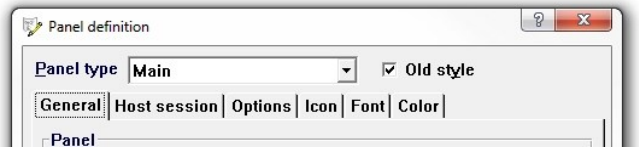
When a panel has been converted to *New style*, there is no going back (unless undo history is still available). NetPhantom converts the controls appropriately by adjusting their sizes

and settings. The combination box will have the number of drop-down lines in the list calculated. The other adjustment settings are taken from the `PHANTOM.INI` file:

```
[controls]
combobox.height=11
entryfield.height=11
spinbutton.height=11
entryfield.adjust.up=2
entryfield.adjust.down=1
entryfield.adjust.left=2
entryfield.adjust.right=2
```

The values are in dialog box coordinates, i.e. X is 1/4 in font width and Y 1/8 of font height. The combination box will use the height `combobox.height`, as the spin button with `spinbutton.height` and the entry field will be adjusted up/down/left/right with the values `entryfield.adjust.`*`direction`*. The settings also apply to the creation of new controls in a panel, where `entryfield.height` applies to the entry field (no adjustment value used).

The *Old/New style* setting is set on a per-panel basis and the conversion from *Old style* to *New style* is initiated in the panel definition dialog.



The Old style option, uncheck for New style.

Again, please note that once the conversion has been done, there is no way to reverse it – converted panels will remain converted.

6.4 Integrated Server

When an application is ready to be tested or debugged, it can be deployed and run on the integrated server. This server is in fact a normal NetPhantom Server running in the background. This means that all relevant application functions/behavior is visible and testable.

Since the application environment is a real server, this means that it needs to be configured much in the same way as a normal NetPhantom server is. This can be done through the interface available under the menu **Server** in the Terminal Editor. The other server configuration alternative is to start a Server Administration Client and log on to the background server.

The details of how to configure the integrated server are available in the *The Server Administration Program chapter*.

6.5 Batch Build

NetPhantom provides a command line interface to the build system in order to use it in conjunction with external Make systems such as Ant. This is typically used when several applications are modules of a larger application. The larger application could consist of just a configuration "umbrella" of which applications to include. From such configuration, the resulting final application would require compiling the module applications first, followed by the "umbrella" application.

Using Batch Build or Compile Distribution, this can be accomplished, with the final result as a *Jar* file.

Requirements

Batch building requires the Java *Development Kit* (JDK), not the Runtime Environment (JRE), as the Java compiler is part of the JDK. The JDK version can be 1.8 or better. The JDK version can be 1.8 or better. If you are creating applications targeting a NetPhantom Server running Java 8 to 17, use the appropriate JDK version.

Performing a Batch Build

The batch building uses a Java command line interface without a GUI that can easily be integrated with build tools such as Ant. Basically the parameters to the batch build are the applications to build.

The syntax is as follows:

```
java -classpath NetPhantomServer.jar;NetRexxR.jar
  se.entra.phantom.server.phed.CompileDistribution
  [-IgnoreUnresolved] [-NetPhantom5] project-directories
```

or use the batch file provided:

```
CompileDistribution.bat [-IgnoreUnresolved] [-NetPhantom5]
  project-directories
```

The *`project-directories`* are a list of project directories (including path), e.g. `"C:\Project dir with space\Module1" N:\Umbrella`. A project directory that has spaces in the path must be enclosed with quotes.

To specify an INI file containing spaces in the path or the file name, surround the entire path and file name with quotes.

Available options:

`-IgnoreUnresolved` Ignores unresolved files.

Return Codes

The possible return codes from the Batch Building are:

0 = OK,
1 = Severe error,
2 = At least one error occurred when compiling a distribution,
9 = Syntax error, e.g. no application directory specified.

7 Configuring the Server

The server is configured in the `server.ini` file. This file is used at startup of the NetPhantom Server. The file must exist in the current directory when the server is started. It consists of several sections described below.

Automatic Start of Server Configuration

Whenever the NetPhantom Server is not properly configured, the server will enter configuration mode and the Server Administration program will start for all clients.

Normally, editing of this `server.ini` file is not required. The server administration program is used by a NetPhantom Client. There are times, however, when this is not possible:

- Severe server configuration problems or invalid settings.
- All ports on the server are occupied, thus no NetPhantom Client can connect to the Server.

For these cases, please revert to manual server configuration as described below.

7.1 Base Section

codePageOEM = Cp437	OEM code page setting for NetPhantom Runtime files. See *Appendix A – Code Pages*.
codePageAnsi = Cp1252	Ansi code page setting for e.g. EE files. See *Appendix A – Code Pages*.
codePageISO = 8859_1	ISO code page setting for HTML files. Valid ISO codepages: 8859_1 to 8859_9. See *Appendix A – Code Pages*.
textFile = server.phm	The text file used for all server messages and strings. This file may be translated into any other language.
textFileLanguage=*ID*	Specify the default language for text files.
textFileLanguages = de,ch=de,at=de,sv	A comma-separated list of the languages supported by the text files. When the same file is to be used for more than one country code, use the '=' to indicate which text file language will be used. In the example Swiss and Austrian are set to German: `ch=de, at=de`
codePageTextFile.de = Cp437	The code page setting for the text file above. See *Appendix A – Code Pages*.
textFile.de = server_de.phm	The specification of the actual name of the text file per language, in this example, German.
preloadClassFile = server.preloadClasses	Specifies a file name that will be used to preload classes to avoid start-up performance issues (typically when the server runs on AS/400). Leave this line empty or comment it to disable this functionality when running on other machines such as Windows and UNIX.

serverGUI = *boolean*	Indicates if the server should run in GUI mode (=1) or not (=0). Set to 0 for production use. This parameter must be set to 0 for non-Windows server environments.
displayFields = *boolean* displayPopups = *boolean* displayHiddenText = *boolean*	Enable (=1) host field, host pop-up window and/or hidden text display in the server terminal window when running in GUI mode for debug purposes. Set all values=0 for production use
inactivityWarning = 5 inactivityTimeout = 10	The timeout value in minutes when a client connection is automatically closed after inactivity. Leave this blank or commented to use indefinite timeout. A warning message (`CLI0026-28`) can be displayed when *inactivityWarning* is set to a positive value (less than *inactivityTimeout*).
forcedTimeout=180	The *forcedTimeout* value is used to kill a hung client session (or when a session is really old). The default value is 3 hours.
pingClientTimeout = *nn*	Set the interval in minutes for the server to ping the client. This setting is used to prevent the connection between client/server over a router or dial-up line from being dropped after a period of inactivity.
traceFile1 = traceFile1.trc traceFile2 = traceFile2.trc maxTraceFileSize = 1000 eventLogToScreen = *boolean* redirectScreenOutput = *boolean* appendTraceFile = *boolean*	Trace settings. Set *"maxTraceFileSize="* to inhibit automatic file rotation. The file size is KB. The default event log user exit will print the output to the screen if *eventLogToScreen* is different from zero (`eventLogToScreen=0`). Set to "1" to redirect all screen output of stdout and stderr to the trace files.
eventExit = se.entra.phantom.server.event.DefaultEventExit	The class name for the Event User Exit. See the Java *EventInterface* for more information.
eventFile1 = logFile1.log eventFile2 = logFile2.log maxEventFileSize = 1000 appendEventFile = *boolean*	Event exit settings. Set *"maxEventFileSize="* to inhibit automatic file rotation. The file size is KB.
EventHistoryCount = 200	The *eventHistoryCount* variable specifies the number of saved events that the server administration program can retrieve. The minimum value is 10, maximum 1000, and default 100.
LUMapperExit = se.entra.phantom.server.lumapper.DefaultLUMapperExit	The class name of the 3270/5250 LU Mapper User Exit. If this setting is not defined, the server will use no LU Mapper. See the Java *LUMapperInterface* for more information.
LUMapperFileName = NetPhantom.map	The filename used by the Default LU Mapper. This filename contains a map of client addresses and LU names.
UserAuthenticationExit = se.entra.phantom.server.authentication.DefaultUserAuthenticationExit	The class name of the User Authentication User Exit.
UserAuthenticationApplication = authentication/userauth.jar	The runtime file used for User Authentication.

AdminExit = se.entra.phantom.server.rconsole.DefaultAdminExit	The class name of the Administration User Exit.
adminAccessControl = ADMIN	Enables domain validation for Load Balancing and Remote commands, as well as for User Authentication.
adminPortDisabled = *boolean*	If set to 1, the admin port (default 1790) will be completely disabled. Note: this will also disable the possibility to use Load Balancing.
ServerAdminRuntime = rconsole/GUI/rconsole.jar	The Server Administration program. Leave this entry empty if remote configuration of the server is not allowed.
ListHeaderBackground = 168,152,144 listBodyBackground = 255,255,193	The list header and body default background colors in RGB format.
ClientLookAndFeel = com.sun.java.swing.plaf.windows.WindowsLookAndFeel	The client look-and-feel setting for Swing. The following are allowed: javax.swing.plaf.metal.MetalLookAndFeel com.sun.java.swing.plaf.windows.WindowsLookAndFeel com.sun.java.swing.plaf.motif.MotifLookAndFeel com.sun.java.swing.plaf.mac.MacLookAndFeel Note that Windows look-and-feel is only supported on Windows platforms.
PrintWindowKeystroke = 80,0,1,0	The print window keystroke (virtualKey, shift, control, alt), where virtualKey is a VK_* value in java.awt.event.KeyEvent, shift, control and alt are set to one or zero for the key augmentation. Leave entry invalid or set virtualKey to zero to disable the functionality.
SkipDrives = A:,B:	A list of drives to be skipped when running List directories. On Windows systems list directories will search the root directory of all drives. Removable drives will create an error. Use skipDrives to avoid this. **Note:** There is no graphical interface in the Server Administration program to this entry. It must be edited directly in `server.ini`.
highlightCurrentNotebookPage = *boolean*	Highlighting of current notebook page under Windows L&F.
stringCache.count = 0 stringCache.minLen = 0 stringCache.maxLen = 0	The String Cache parameters (empty or zero for none). String caching can increase client performance by 50% but requires more memory in the client
resolveDNSName = *boolean*	Once a client DNS name has been looked up, it is cached, so no further lookups are performed. By default, DNS look-up is disabled to enhance performance.
allowHiddenButtonTrigger = *boolean*	Set to 1 to allow a hidden button to be activated by the defined action key (e.g. Escape, F1-F12)

7.2 User Exit Print Section

A user exit can be defined as handling printing.

ClassName = se.entra.phantom.client.UserExitPrint	The class name of the user exit class.
Keystroke = 80,1,1,0	The activation key for the user exit print function (virtualKey, shift, control, alt), where virtualKey is a VK_* value in java.awt.event.KeyEvent, shift, control and alt are set to one or zero for the key augmentation. Leave entry invalid or set virtualKey to zero to disable the functionality.
FontOverride = Courier	The override font for all controls except list boxes.
FontListOverride = Courier	The override font for list boxes.
ClipChar = #	This character is appended to all texts that are clipped.

7.3 Port Section

The port section defines the server socket ports (TCP/IP connections) that will be used to listen to NetPhantom Client connections, HTTP requests and SSL connections (HTTPS or NetPhantom Clients using SSL). The ports to use are specified with the `portID` item, and an individual port setting is done with the items:

portID	Comma separated list of ports to be used.
portID.description	A descriptive text for the port entry.
portID.port	The port number must be from 1 to 65535.
portID.bindAddress	The `bindAddress` argument can be used on a multi-homed host for a server socket that will only accept connect requests to one of its addresses. If `bindAddress` is left empty, it will default accepting connections on any/all local addresses. The address is either a IP name or a raw IP address (e.g. `123.234.132.243`).
portID.queueLength	The maximum queue length for incoming connection indications (a request to connect). If a connection indication arrives when the queue is full, the connection is refused.
portID.sslSection	The section in this INI file describes the SSL settings. Leave empty if no SSL is required. This allows different SSL settings for different ports. Note that each SSL section requires extra server memory when used.
portID.externalSSL	Flag (0/1) indicates that SSL is provided on this port, but by external means such as hardware or a proxy SSL server.
portID.loadBalance	Flag (0/1) indicates if this port should be used for load balancing. This flag must be set (1=one) in both the slave and controller server to enable load balancing.

Note: Using port numbers below 1000 under a UNIX system requires *super user* privileges (or the *daemon* user). This means that a "normal" user cannot run the server using the default HTTP port 80 and/or default HTTPS (SSL) port 443.

Example (defines HTTP, SSL and NPCLIENT but only loads NPCLIENT and HTTP):

```
portID=HTTP,NPCLIENT

HTTP.description=Web Server and NetPhantom Client HTTP Tunneling
HTTP.port=80
HTTP.bindAddress=
HTTP.queueLength=50
HTTP.sslSection=
HTTP.externalSSL=0
HTTP.loadBalance=1

NPCLIENT.description=NetPhantom Client
NPCLIENT.port=789
NPCLIENT.bindAddress=
NPCLIENT.queueLength=50
NPCLIENT.sslSection=
NPCLIENT.externalSSL=0
NPCLIENT.loadBalance=1

SSL.description=Secure Web Server and NetPhantom SSL Client
SSL.port=443
SSL.bindAddress=
SSL.queueLength=50
SSL.sslSection=SSL
SSL.externalSSL=0
SSL.loadBalance=1
```

7.4 Country Settings Section

listSeparator = ; thousandSeparator = decimalSeparator = .	Number information: number separator when a list of numbers is used, thousand separator and the decimal separator. The only entry that may not be empty is the *decimalSeparator*.
clockSeparator = : clockAM = clockPM =	Time information. To use a 24-hour clock, leave *clockAM* as empty string.
dateFormat = 2 dateSeparator = -	Date information. *dateFormat* is: 0 - Month, Day, Year 1 - Day, Month, Year 2 - Year, Month, Day
sortRule=< ' '< '!'< '#'< '$'< '%'< '&'< '('< ')'< '*'< '+'< ','< '-'< '.'< '/'<\0< 1< 2< 3< 4< 5< 6< 7< 8< 9< ':'< ';'< '='< '?'< '@'< a,A< b,B< c,C< d,D< e,E< f,\F< g,G< h,H< i,I< j,J< k,K< l,L< m,M< n,N< o,O< p,P< q,Q< r,R< s,S< t,T< u,U< v,\V< w,W< x,X< y,Y< z,Z	The *sortRule* is used to specify the rules for the order in which characters are sorted. See the javadoc for a description of the java.text.RuleBasedCollator which extends java.text.Collator.

7.5 Event Filter Section

The form of these commands must be:

```
EventFilter ---+-+-- ALL        --+--- >
               | |                 |
               | +-- client_id  --+
               |                   |
               +---- , -----------+

          > --- [INCLUDE:event_id] --- >

          > --- [EXCLUDE:event_id] --- >

          > --- [DEFAULT:event_id]
```

where `client_id` can be a comma-separated (no spaces in between) list of several `client_ids`, and `event_id` is a comma-separated list (no spaces in between) of `eventIDs` as defined in `server.phm` and/or a range of `eventIDs` and/or using the `*` wildcard to accept any `eventID` starting with the text preceding the `*`.

To specify all events, use the event range string `*-*`.

By default, all events are turned on.

7.6 Trace Section

This section defines which traces should be turned on or off when the server is started.

Syntax:

```
Trace client subsystem level
```

where *client* is `ALL`, *subsystem* is `Telnet`, `Host`, `Client` or `API`, and *level* is `Off`, `Binary`, `Verbose`, `BinaryVerbose`.

By default, all traces are `Off`.

Example:

```
trace ALL TELNET Binary
trace ALL HOST   BinaryVerbose
```

7.7 Font Substitution on Client Section

NetPhantom implements a font substitution mechanism that allows a specific font on a client to be remapped to another font. This mapping mechanism also allows different font mapping for different categories of clients.

Definition of a Font Substitution List

In `server.ini` in the `[FontSubstitution]` section, specify a list of sections that should be used, for example:

```
substitutionList=FontWIN
```

There is no limit to the number of sections specified in the `substitutionList`.

Definition of a Client Specific Font Substitution List

The section has a name (in this example `FontWIN`) and the following content:

To disable font substitution for different clients and just use the default font substitution under the `[FontSubstitution]` section, comment or remove the entry `substitutionList`.

```
[FontWIN]
;OsName      = Windows XP
;OsVersion   =
;JavaVersion = 1.6
JavaVendor   = Sun Microsystems Inc.
;isApplet    = 1

10 0 = 12BI Courier, 12BI Monospaced
8  2 = 12BI Courier, 12BI Monospaced
10 0 = 12   Courier, 12   Monospaced
```

There are five criteria that can be used to determine which clients will use the specific font substitution section rather than the default one. They are:

To disable a criterion, i.e. to make it enforced by default, comment the criterion in question. The example above will cause a client running a Java Virtual Machine by "Sun Microsystems Inc." to use the client-specific font substitution.

Remapping a Font

The following fonts are used on the client. Font index 0 is the default font.

Font index	Java name	Windows name	OS/2 name
0	Dialog	System	System Proportional
1	Monospaced	Courier New	Courier
2	SansSerif	MS Sans Serif	Helv
3	Serif	MS Serif	Tms Rmn
4	Monospaced	Fixedsys	System Monospaced
5	Monospaced	Terminal	(Phantom) System VIO
6	Dialog	System	System Proportional Non-ISO

The Java fonts may not exist, so the following table maps the names to other fonts when the client is started:

Name	Alternate(s)
Dialog	DialogInput, SansSerif, Helvetica
Monospaced	Courier
SansSerif	Helvetica
Serif	TimesRoman

Each font substitution is done with the point size, a space, and the font index (0-6). After the equal sign, specify point size, a space and, then font name. Several such font entries may be specified separated by a comma. The client will try to find the matching font, and if found, it will be used instead of the normal font.

Syntax:

```
PointSize[Styles] FontIndex =
      PointSize1[Styles] FontName1[:heightDiff1]
   [, PointSize2[Styles] FontName2[:heightDiff2]]
   [, ...etc...]
```

where `[Styles]` (note: directly after point size without space) can be:

`B`	Bold
`I`	Italic
`BI`	Bold and Italic

and `PointSize` range is 4-255 and `FontIndex` range 0-6.

The `[:heightDiff]` is optional and causes the height of the font to be adjusted by the number specified with `heightDiff` in pixels (the number may be negative). This new font height is only used by list boxes and is intended to make a perfect match for a font under the Windows environment with a font for the NetPhantom Client.

Example: map Monospaced Bold 10 to Monospaced/Courier Bold Italic 8:

```
10B 1 = 8BI Monospaced, 8BI Courier
```

The NetPhantom System VIO font

The NetPhantom System VIO only exists under Windows. As the NetPhantom Client can run on any platform, a replacement font must be found that matches the size and proportions of the original font.

The NetPhantom System VIO font (index 5) is therefore mapped to Monospaced (index 4) in the NetPhantom Server. See the table below for remapping.

When panels are created in previous versions of the system with e.g. list boxes without scroll bars with this font, and the size of the list box and its columns are very closely related to the font, changes in size may be done using font substitution on the client.

Size	**Mapped to**
6 – 10	Monospaced 8
11 – 12	Monospaced Bold 10
> 12	Monospaced Bold (with same size)

Note: The remapping of the *NetPhantom System VIO* font is done prior to any other font substitution, so remapping this font in the font substitution requires a remap of the *Monospaced* font (*index 4*) with a possible *Bold* attribute as the table above shows.

Fixed Font Metrics

Problems with panel appearance due to differences in the measurements of certain font properties between Java Virtual Machines and/or versions and also between operating systems can be solved with the Fixed Font Metrics function in the Server Administration Client.

It creates a file called `fontmetrics.dat` that contains the height, ascent, descent and average character width for all enumerated fonts. These settings are based on the font measurements in the machine used to create the .DAT file. A reference can then be added to the `[FontSubstitution]` section of `server.ini`.

```
fontMetrics=fontmetrics.dat
```

These metrics will then take precedence over those of different VMs and operating systems. This was a method used with NetPhantom versions 5 or earlier and should not be used anymore.

Example of Font Sections

The example below shows a font substitution that is specific for Windows and then a default font substitution (when a client doesn't match the criteria):

```
[FontSubstitution]
substitutionList=FontWIN

; Example: map Monospaced Bold 10 to
; Monospaced/Courier Bold Italic 8:
10B 1 = 8BI Monospaced, 8BI Courier

 [FontWIN]
;OsName=Windows XP
;OsVersion=
;JavaVersion=1.6
JavaVendor=Sun Microsystems Inc.
;isApplet=1

10 0 = 12B Courier, 12B Monospaced
8 2 = 10B Courier, 10B Monospaced
```

7.8 Runtime Application Section

The Runtime Application section specifies NetPhantom applications to load into memory during server startup. The applications are shared by all users. Each user may run any combination of the loaded applications with the Merge-on-the-fly technique.

Each application is assigned a name (other than `load` and `defaultApplication`, which are reserved). This is done as:

```
APPID = path/RUNTIME_APPLICATION.EXTENSION
```

The `APPID` is the "name" by which the application is known to the clients.

There must be an entry such as:

```
load = APP1 ADM GAME
```

to define the applications that the server will load during startup.

Because clients can request applications that the server may not have loaded or may not request any specific application, a default application must also be specified as:

```
defaultApplication = APP1
```

Note: The application ID `SERVERADMIN` is reserved for the Server Administration program and should not be used in the `[Application]` section.

Here is an example of a typical `[Application]` section for a server running under Windows:

```
[Application]
defaultApplication = APP2
load = APP1 APP2 SAMPLES_REPORT

APP1 = /applications/myapp.jar
```

```
APP2 = /applications/myapp2.jar
SAMPLES_REPORT = samples/report/report.jar
NP_CLIENT_TERMINAL=terminal/terminal.jar
```

7.9 Host Section

The host section defines the different host IDs available for clients. A client chooses a host ID with the parameter:

Applets `<param name=HOSTID value="n">`

Java applications `HOSTID:n`

Each host session can be defined only once within the range A-Z. The REXX function *HostConnect* is used to change between the sessions. A client may request no host connection using the parameter:

Applets `<param name="HOSTID" value=".">`

Java applications `HOSTID:.`

Host Session Definitions for 3270 and 5250

The host session entry has the following syntax:

```
ID = Type HostAddress [ HostPort [ PeerData ] ]
```

The parameters are:

ID	Short session ID (`A-Z`).
Type	`3270`, `5250` or `EE`.
HostAddress	IP address or name (*nothing* for Type = `EE`).
HostPort	Value (optional: default is `23`).
PeerData	User data used by the emulator peer (optional). For default 3270/5250 terminal sessions, this *PeerData* is a terminal type or a list of terminal types.
*HCP	A host code page can be specified per session. *HCP=*validCodePage* See Appendix A Code Pages for a list of valid code pages.
**options*	There are several options that all begin with an asterisk and are all handled by the Server Administration Program.

`3270` and `5250` entries may look like:

```
A = 3270 6.1.11.112 23 3279-3,3278-3,3279-2,3278-2
B = 3270 sibil.unige.ch 23 *HCP=Cp1141
C = 5250 entra2.entra.se 23 3179-2
```

The Host session definition may also contain a description and a list of the allowed terminal sessions. They are defined as follows:

```
A=EE nothing 0 samples/TUTORIAL.EEM
A.description=The Tutorial Application
A.allowedSessions=BCDEFMRT
```

The following entries are used when you have defined your own terminal runtime application and are not using the default terminal application class:

```
A.application=runtime application it refers to
A.class=my.terminal.application
```

Terminal Types

The *PeerData* in the configuration is used for 3270/5250 as the terminal types. Several terminal types may be specified. When the Telnet session is established, a negotiation of which terminal type to use will take place. If no terminal type could be negotiated, the host session will be closed with an error.

For both 3270 and 5250, it is important to specify the correct terminal type, because it is used to decide the terminal emulator presentation size (i.e. the screen size).

See the tables below for valid terminal types.

Supported 3270 Terminal Types

Model	Display type
3278-2	Default, 24 x 80
3278-3	Default, 32 x 80
3278-4	Default, 43 x 80
3278-5	Default, 27 x 132
3279-2	Extended attributes, 24 x 80
3279-3	Extended attributes, 32 x 80
3279-4	Extended attributes, 43 x 80
3279-5	Extended attributes, 27 x 132

Note: Certain TN3270 hosts require the terminal type to be specified as model-x-E in order to display extended attributes, e.g. 3279-3-E.

Supported 5250 Terminal Types

Model	Display type
3179-2	24 x 80 color (*)
3196-A1	24 x 80 monochrome
5292-2	24 x 80 color
5291-1	24 x 80 monochrome
5251-11	24 x 80 monochrome
3477-FC	27 x 132 color (*)
3477-FG	27 x 132 monochrome
3180-2	27 x 132 monochrome

Note: (*) above indicates suggested terminal types.

3270 Numeric Field Override Option

The 3270 host may optionally use the *Numeric Field Override* setting. To activate it, specify `*NFO` before the terminal types. This option allows all characters to pass through "Numeric Only" fields and on to the host. Standard 3270 host applications allow the definition of data input fields as Numeric Only or Alpha-Numeric. The 3270-defined "numeric" characters typically include the following:

Example:

```
A = 3270 6.1.11.112 23 *NFO,3279-3,3278-3,3279-2,3278-2
```

Custom Host Datastream Exit

The host may optionally use another transport layer for the datastream instead of the default TN3270/TN5250. The implementing Java class must implement the `se.entra.phantom.server.TelnetHost` interface. See *Appendix G – User Exits – The Telnet Host Interface* or the online documentation.

To specify the custom host datastream exit, define the host session as:

```
hostID = {3270 | 5250} ipAddress
*className[,*NFO][,terminalTypeList]
```

The EE package

For demonstration purposes, the EE (Emulator Emulator) package is available. For more information on EE, see the *Phantom Hurricane User's Guide and Reference*. The EE entry syntax is as follows:

```
A=EE nothing 0 samples/TUTORIAL.EEM
```

Host Datastream Problem Determination Aid

The host datastream may be played back for debugging and problem determination purposes if the Telnet Binary Trace is turned on for just one client. To play the trace back (with e.g. the *Enter* key to display the next screen) specify the following (on a single line):

```
A = 3270 c:\traceFile.trc 23
   *se.entra.phantom.server.Telnet3270PlayBack,3278-3
B = 5250 c:\traceFile.trc 23
   *se.entra.phantom.server.Telnet5250PlayBack,3179-2
```

Other Host Settings

hostCodePage = Cp037	The `hostCodePage` is a global setting for all host sessions. See *Appendix A – Code Pages* for valid EBCDIC code pages. This setting is required. To specify different codepages for various host sessions see *Host Session Definitions for 3270 and 5250*, above.
defaultHostSession = *sessionID*	The `defaultHostSession` is the host session that the client will use if it has not specified the parameter `HOSTID` or if it is invalid. This setting is required.
windowBounds = 0,0,587,428	The `windowBounds` is default terminal window bounds in pixels (x,y,width,height). It allows you to specify the exact size and location of the first panel.
disableHostAlarm = *boolean*	Disables host alarm beep when GUI panels are displayed.

boxDrawing = *boolean*	Enables (=1) 3270 box drawing in the 3270 Extended Data Stream (Graphical Escape or APL2 character set).
boldTerminalFont = *boolean*	To use bold terminal for for the client (=1) or not (=0).
terminalFontName terminalFontSize = 0	The initial terminal font to use (if present, Monospaced otherwise). Set the terminalFontSize=0 to use auto font sizing, otherwise specify the font size.
cursorHeight = 100 cursorNonBlinking = *boolean*	Cursor height (in % of font height, empty=100%) Set to 0 if you want the cursor to blink.
typeAhead = 1	If type-ahead is enabled, REXX functions such as HostSend will return zero (OK) even if the host session is locked, but the keystrokes will be queued. The Reset key (or @R), or an error occurring in the host session clears the queue.
keyboardAutoReset = *boolean*	Sets auto reset for the keyboard.
smartInsertMode = 1 PCInsertMode = 1	Smart insert mode equates space with 0. PC insert mode keeps the insert setting you ahe chose until the next time you change rather than losing them when you send Enter for example.

7.10 Web Server Section

The NetPhantom Web Server is configured with two sections: `WebServer` and `CGI`.

httpTunneling = 0	Specifies if HTTP tunneling is enabled (=1) or not (=0). Must use a Web Server (defined above).
headerReadTimeout = 0	This timeout is used when reading the header portion of the HTTP request and the form data in a POST request. The value is in milliseconds (zero=indefinite, default=indefinite).
keepAlive = 0	This setting is used to control if "Keep-Alive" connection is supported or not (default=0, see *keepAliveTimeout* for more information).
keepAliveTimeout = 20000	This timeout controls the waiting time between a valid HTTP request-reply and the next request from the client using a "Keep-Alive" connection. Once a header is detected, the "headerReadTimeout" value will be used. The setting is quite important because the server might be able to send a big amount of data to a client that is located e.g. on a slow connection to the Internet, causing the next request to be sent only once the client has read the full server reply. This will then cause the server to have to wait a considerable amount of time before the next request is present. The value is in milliseconds (zero=indefinite, default=20000 i.e. 20 seconds).
logHTTPRequests = 0	Logging of all HTTP requests to the event log (0=off, 1=on). Default is *off*.

documentRoot = htdocs documentError = hterrors	The location of the document root and the error documents. The *documentError* directory should be relative to the *documentRoot*
documentDefault = index.html index.htm index.shtml index.shtm	Specifies the document for errors, the root for documents and the default index document(s).
fileSystemCaseSensitive = 0	Specifies if the file system is case sensitive (e.g. UNIX). Note that this is not the machine system setting but rather the file system for *documentError* and *documentRoot*.
replyServerString=	The HTTP Server Identification string. Empty or not specified sets the string as "NetPhantom/*X.YY* (Build *NN*)". This entry can be used to obfuscate the server identity.
defaultCacheSize = 3000 htmlCacheSize = 1000	Cache parameters in KB. The *defaultCache* is used for images, etc. The *htmlCache* is used for parsed documents. The memory requirements are about 10 times the cache size due to pre-parsing. Specify zero to disable the cache.
maximumResourceSize = 128	The maximum size of a resource to be placed in the cache (in KB).
sendBlockSize = 8	The block size for sending data to the client (in KB).
nonceTimeout = 5	Specifies the timeout value in minutes that a "nonce value" can be reused. This value is used for HTTP authentication. In order to disable the reuse of "nonce values", set this parameter to zero (i.e. will require a unique "nonce" for each HTTP request). Note that setting this value too low affects the browser response time negatively.
maximumRequests = 250	Specifies the maximum number of simultaneously processed HTTP requests. Each HTTP request is processed in a separate thread, and this parameter is used to avoid server flooding due to e.g. attacks.
warningRequestCount = 200	The *warningRequestCount* is the limit of simultaneous HTTP requests before an event is logged. When this limit is reached, it indicates a bottleneck or heavy server load.
hostAddress =	The IP address of the server for multi-home networks.
hostName = useIPAddressAsDNSName =	The name of the server for multi-homed networks. Setting *hostName=RawIP* uses the host address instead.
htmlWriterIndenting = 0	Should parsed HTML code be parsed, default is no indent. 0=no indent, 1=do indent. **Note:** There is no graphical interface in the Server Administration program to this entry. It must be edited directly in `server.ini`.

htmlWriterPreserveSpaces = 1	Should parsed HTML code preserve or strip spaces from source HTML. Spaces are preserved by default. 0=strip spaces, 1=preserve spaces. **Note:** There is no graphical interface in the Server Administration program to this entry. It must be edited directly in `server.ini`.

7.11 The Servlet/CGI Section

The Web Server supports CGI's (Common Gateway Interface). The CGI's are not discussed in-depth in this section. A CGI can be a "normal" CGI or an "HTML-included" CGI, the latter form is defined as "<*cginame*>".

name = JavaClass [, classpath] <*name*> = JavaClass [, classpath]	Defines a CGI as "normal" or "HTML-included". The name is then used to define a Web Server CGI resource. The optional *classpath* can be specified and should be colon (;) separated.

7.12 The Web Applications Section

This section lists the items in the definition of web applications for the web server.

```
[WebApps]
FANCYTUTOR=B,TUTOR,No,0,No,5,HTMLAPP
NORMALTUTOR=B,TUTOR,No,0,No,5,HTMLAPP
```

The definition consists of:

Name	The name of the web application.
Host session	The host session ID that should be used for the web application. This ID must be defined under the Host section of server.ini.
Selected applications	A blank delimited list of NetPhantom applications to be included in the web application. These applications must be defined under the Applications section of server.ini.
Start search for matching screen from first application	Set to **Yes** if you want NetPhantom to begin searching for a matching panel from the first application in the list of selected applications rather than from the current application.
Wait stable delay	This is an optional waiting period to make sure that the host has delivered all information to the NetPhantom Server. This value is specified in milliseconds.

Cookie required	Set to **Yes** if a cookie should be required for the application. A cookie (a locally stored information file) is used to identify the client with the server. The server assigns a unique identifier for each client session passed to the browser when the application starts (and only then). This identifier is sent from the browser with each new request (POST and/or GET), thus enabling the server to check the authenticity of the client with the session. **Note:** It is highly recommended that you use this option! It guarantees that another intruding (malicious) client cannot "steal" the session. Combined with the usage of SSL it then becomes the preferred, most secure client/server communication (that is stateless as HTTP is in its design). When a browser session is closed, the cookie is automatically destroyed.
Session time-out	The session time-out is the time that the NetPhantom Server will wait for the client to respond before disconnecting from the host. This value is specified in minutes.
CGI name	The CGI name for all web applications is HTMLAPP.
Application start Start new session = 0 Use previous session = 1 Restart previous session = 2	Options for HTML Web Application that allow you transfer global variables between sessions. Start a new session starts a new session with empty global variables. Use previous session transfers all global variables that are not session-specific to the new session. Restart previous session sends the START message to the Application object, which must return 1 to keep the session alive.

7.13 Remote Applications Section

This section contains the definitions for remote applications that are accessed from external applications via the RAPP API.

```
RAPPAPI=R,SAMPLES_RAPPAPI,No,2,se.entra.phantom.server.rapp.DefaultRemote
Application
```

The items in the definition are:

Name	The Remote Application must have a unique name. It can be the same as the name of the NetPhantom Application.
Host session	Select a host session for the application. The host session must be defined on the Host page of the Configure server dialog box.
Selected applications	The list of applications that will be used for the remote application.
Start searching for matching screens from first application	This option is related to the use of Merge-on-the-fly and causes NetPhantom to begin searching for a matching screen from the first application in the list of selected applications rather than from the current application.

Session time-out	You may choose to set a timeout for the remote application. This number is in minutes.
Class name	The class name is the name of the class that implements the server side RemoteApplication. The default class is se.entra.phantom.server.rapp.DefaultRemoteApplication

7.14 Event Messenger Section

The Event Messenger section lists and defines the Event Messengers for the server. It is not included in *server.ini* by default. You must add the section yourself as described below or using the Server Administration program.

Directly under the Event messenger section is a list of event messengers and the classes they implement. This is followed by a separate section for each event messenger that contains the items specific to that class as in the example below.

```
[EventMessenger]
SMS=se.entra.phantom.server.extra.SMSEventMessenger
;MailEventMessenger=se.entra.phantom.server.extra.MailEventMessenger

[SMS]
smsPhoneNumbers=+46709790557,+46709790844
smsAppId=entraib
smsServer=10.254.0.170
smsServerPort=2099
smsKeyFile=d:/programfiles/apache group/servlets/entraib
eventFilter=ALL EXCLUDE:*-*

[MailEventMessenger]
eventFilter=ALL EXCLUDE:*-* INCLUDE:SW0010-SW0011
SenderName=phantom
ServerName=entragroup.com
ReceiverMail=administrator@entragroup.com,john.smith@support.com
MailSubject=NetPhantom Server Event Message
```

See *Event Messengers* for more information.

7.15 SSL Settings Section

Settings for SSL are stored in separate sections. There may be several SSL settings groups for individual ports. The following table describes the items in an SSL section.

cipherSuites = comma-separated-list-of-files	Specify acceptable cipher suites separated by a comma. See *Chapter SSL – Secure Socket Layer* for more information.
caCertificates = comma-separated-list-of-files	Specify CA certificate files to use in the CA certificate chain. The list is comma-separated. A file may be DER or PEM-encoded X.509 certificate. A PEM-encoded certificate is automatically detected and decoded to raw form.
identityFile = id.p12	The *identityFile* is a comma-separated list of identities.
identityUser	The *identityUser* is the User ID holding the password (or pass phrase); for the identity. The User ID is managed under the Server Administration program, menu item **Server - Manage users**.

maxTemporaryKeyLifetime = 1440	Set maximum lifetime of temporary keys. Use this value to set the maximum lifetime of temporary keys. This is the length of time, measured from the creation of the key, after which it will be destroyed, regardless of how many times it has been used. This is important for security (when the value is high), to prevent attacks on temporary keys, but has consequences for efficiency. The SSL specification recommends a value of 24 hours (=1440 minutes). The minimum is 0.
maxTemporaryKeyUses = 500	Set maximum uses of temporary keys. Use this value to set the maximum number of times that temporary keys will be reused. This is important for security (when the value is high), to prevent attacks on temporary keys, but has consequences for efficiency. The SSL specification recommends a value of 500. The minimum is 1.
sessionCacheCapacity = 100	Set a session cache capacity limit. The session cache is used to improve SSL efficiency. New connections made between a familiar client and server can use cached security parameters to eliminate the costly asymmetric operations associated with the full SSL handshake. Specifies the maximum number of session cache entries that will be supported. Limit this to prevent memory problems. The suggested value is 100, minimum is 1.
sessionCacheTimeout = 60	Set a session cache timeout. Specifies the maximum duration in minutes that session cache entries will be maintained. You will want to expire old entries for security reasons. Existing entries older than the specified timeout are automatically purged. The suggested value is 1 hour (= 60 minutes), minimum is zero and implies no caching.
supportSSLv2Hello = 1	Support SSLv2-compatible connections. This flag instructs the server to accept SSLv2 hellos from SSLv3 capable clients. The SSLv2 hello is only supported for compatibility purposes. Its use should be phased out as soon as possible.
suppressSSLv3 = 0	Disallow SSLv3 support. This flag instructs SSL to suppress SSLv3; i.e., only support TLSv1 connections. You can't suppress both SSLv3 and TLSv1.
suppressTLSv1 = 0	Disallow TLSv1 support. This flag instructs SSL to suppress TLSv1; i.e., only support SSLv3 connections. You can't suppress both SSLv3 and TLSv1.

7.16 Session Pooling Data

The syntax for the session pooling entry is:
runtime ID.host ID=poolEnabled,min,max,pingTime,sriptFile

For example:

```
SAMPLES_TICKET.T=1,1,1,15,samples/ticket/ticket.xml
```

runtimeID	The application ID of the runtime file as defined on the **Applications** page of the **Configure – Base** dialog.
hostID	The ID specified for the host session on the **Host** page of the **Configure – Base** dialog.
poolEnabled = *boolean*	Set 1 to enable the pool and 0 to disable.
Min = *nn* Max = *nn*	Minimum and maximum number of pooled sessions.
pingTime = *nn*	Time in seconds for a ping to be sent to the pooled session to keep them alive. Set to 0 to disable.
scriptFile = *filename*	The path to the XML file is used to handle the session pooling script. The script is used to handle e.g. log in and session disposal. See *Session Pool Scripting*.

7.17 Load Balancing Section

This section specifies whether load balancing is to be used for the server and, if so, whether the server is controller or slave.

Role	This flag indicates whether load balancing is disabled=0 or the server is a load-balancing controller=1, slave=2 or controller+slave=3
readTimeout = 10	This is the time in seconds that the server will wait to receive something before checking to see if the other server is up. It should not be set to less than 5 seconds.
serverAddress =	If the server is a load-balancing slave, specify the IP address of the controller server.
serverPort = 1790	The administration port number of the controller server.
adminUserID	This setting is used by programs that wish to contact the NetPhantom server via the administration port such as the Remote Command Line Utility or when using load balancing. They must contact the server using this ID and password.

7.18 Cluster Controller Section

The NetPhantom Cluster Controller is a program that runs several NetPhantom Server processes on the same machine using load balancing. It should not be confused with the controller server defined for load balancing.

serverCount = 0	The number of servers in the cluster. Must be greater than 1 for the cluster controller to work. When set to 0 the Cluster Controller is disabled.
executeCommand	The execute command to start the NetPhantom Server. Default is: `java -cp .;samples; NetPhantomServer.jar; NetRexxR.jar; NetRexxC.jar; acme-4j.jar; bcprov.jar; bcpkix.jar; bcutil.jar; openxml-1.2-np.jar; mail.jar; activation.jar se.entra.phantom.server.Start`
upgradeCommand	The command for server upgrade when internal upgrade is used. Internal upgrade is required when running the Cluster Controller as a Windows Service. The default command is: `java -classpath . UpgradeNetPhantomServer` If not running the Cluster Controller as a service, use External upgrade, i.e. leave the upgradeCommand parameter blank.
logFile = *filename*	Events in the Cluster Controller can be logged separately. Specify a filename for the Cluster Controller log.
appendLogFile = *boolean*	Indicate whether the log should append after a restart (1) or empty the log and begin again (0).

The remaining parameters for the Cluster Controller are server unique. They are the administration port, bind address, IP address and DNS name of the server. The format is the parameter name followed by a dot (.) and the server number as in the example below.

```
adminPort.1=1790
bindAddr.1=127.0.0.1
ipAddr.1=127.0.0.1
dnsName.1=localhost
```

7.19 Setting up Web Server Files for NetPhantom

This section only applies when the NetPhantom Web Server is not used, i.e. when other third-party Web Servers, such as Apache, are used.

When NetPhantom is used for clients running browsers, the clients will connect to a web server. Due to security issues of the Java environment that the browsers can impose on the NetPhantom Client, the NetPhantom server and the web server should be located on the same physical machine (can be redirected, but this is not discussed here).

Perform the following steps to set up the web server:

1. Change the directory to the web server root directory, i.e. from where the *index* HTML document is loaded.

2. Create a subdirectory `NetPhantom` and set it as current.

3. Copy the `NetPhantomClient.jar` file into this directory.

4. Copy the `Image` directory from the NetPhantom installation recursively into the current directory (to which `NetPhantomClient.jar` was copied in step 3). This will make sure that all required images are accessible to the client.

5. Create an HTML document for the NetPhantom Client.
6. Optionally perform a Java Plug-in HTML conversion on this document.

8 NetPhantom – Connectivity, Internet, and Security

As NetPhantom gives the ability to transport an entire application or a small part of it to the Internet or an Extranet, several issues are immediately raised:

- Data transport
- Security

Security is the premier issue, while the data transport issue results directly from transporting data. With the advent of the Internet, everyone was amazed at the ease of connectivity regarding transporting data. Some people took advantage of this situation and listened/grabbed/modified some of the transported data. Moreover, a user on the Internet could obtain access to a company's internal network.

Firewalls

The firewall was the solution for companies that wanted to have access to the Internet and be able to allow "some" access to certain predefined resources inside the firewall, e.g. the Company Web Server.

By creating firewalls – some very smart ones, some less smart – the transport of TCP/IP data became heavily restricted.

The setting "Proxy" for NetPhantom Client enables communication through these firewalls or proxys.

Encryption

Data encryption can also be performed from 40 to 256 bits today, with adequate speed. In the beginning, only very sensitive information was encrypted, but nowadays the use of encryption is more widespread, along with digital signing, etc.

The data exchanged between the NetPhantom client and server is encrypted by default with a simple "garbling" algorithm to avoid sniffing of such things as user Ids and passwords. This is accomplished by the `Crypto` interface available on both the client and the server. See the online API documentation for more information.

For more advanced encryption using SSL see *ChapterSSL – Secure Socket Layer*.

Java Applet Limitation

The NetPhantom Client communicates with the NetPhantom Server using Sockets (and Secure Sockets when using SSL). When the NetPhantom Client is running in a browser, the Java Security Manager requires the applet to contact the host from which the Java code (jar) came. The applet may contact the same or another port on the host from which the Java "*codebase*" code came.

The NetPhantom Client loads all additional Java class files from the "codebase". All resources such as images or HTML help documents, however, are loaded from the *"documentbase"*. The "documentbase" is the place from which the HTML document containing the <APPLET> tag is loaded.

8.1 NetPhantom Topologies

This section will discuss the different ways to physically configure NetPhantom installation.

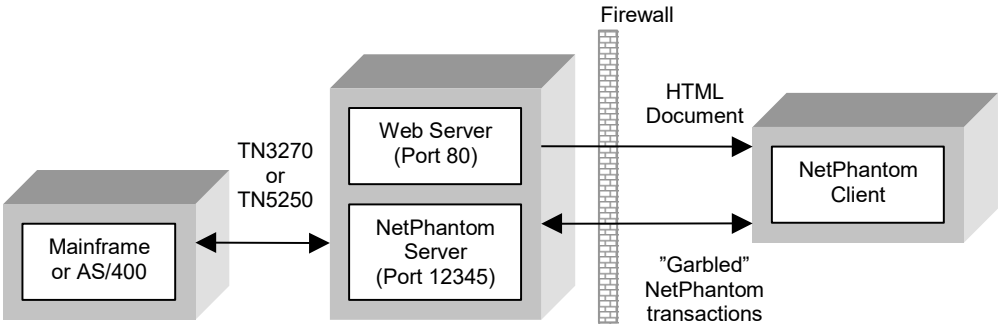
When transporting data over the Web (or even on an Extranet), there may be firewalls stopping data traffic on TCP/IP ports other than 80 (the default port for HTTP). HTTP is

the protocol used for transporting HTML pages and their referenced resources (e.g. GIF images).

In general, most companies allow a user to surf the Internet. These users will be able to access a NetPhantom Server to run an application. The means to do this is by tunneling the NetPhantom transactions inside the HTTP protocol. This is called *HTTP Tunneling*.

NetPhantom Intranet and Extranet Topology

On an Intranet, the TCP/IP data traffic is normally not restricted by firewalls that prevent access to certain hosts and/or TCP/IP ports. For Extranets, it is very common to have a firewall. In the case described here, the firewall must be opened for data traffic on both port 80 (default HTTP port) as well as port 12345 (the NetPhantom port).



The diagram above shows a single-server solution. The NetPhantom server may very well be located in another physical machine.

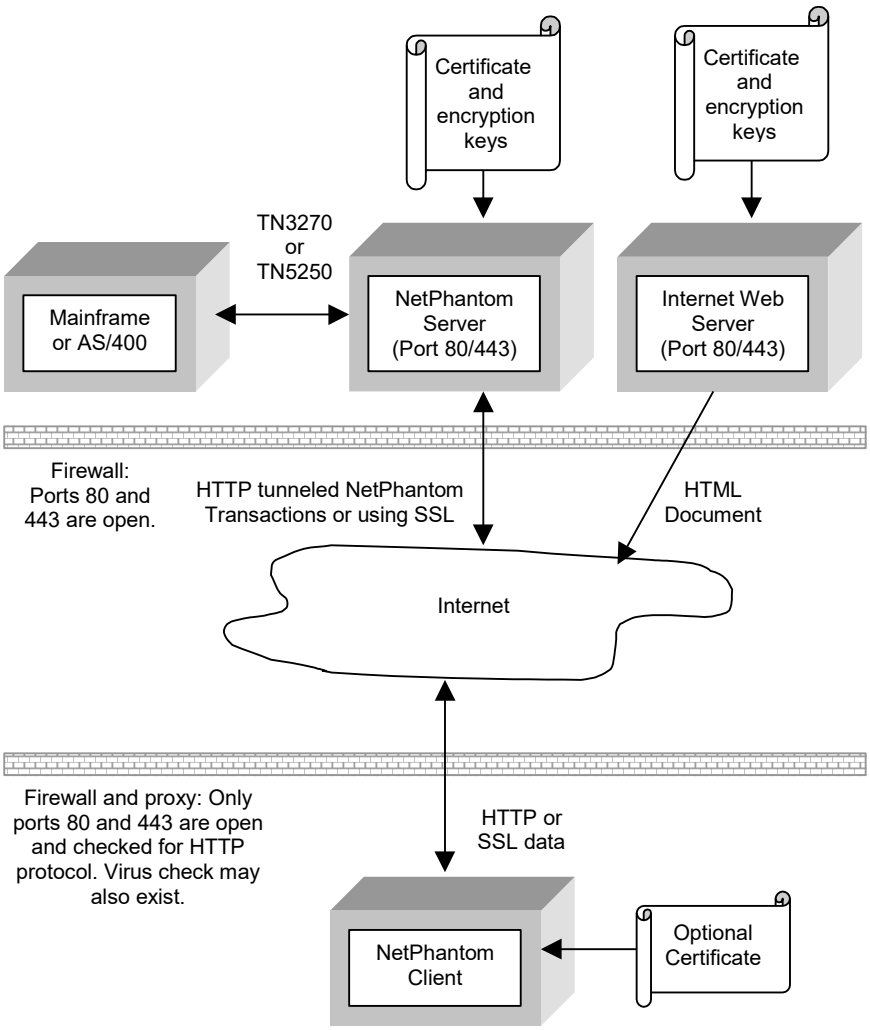
NetPhantom Internet Topology

This topology assumes that firewalls exist on both the NetPhantom server site as well as a possible firewall on the client side. The firewall on the client side is not controllable by the NetPhantom Server administrator nor could anybody require the client-side firewall to have a specific port open. Due to the Internet, one doesn't know where a client could be in the world.

Depending on the sensitivity of the data, HTTP tunneling of NetPhantom "garbled" transactions or SSL can be used. Normally HTTP data is transported over port 80. SSL stands for Secure Socket Layer and encrypts the data between two points, normally a server and a client. The default TCP/IP port for SSL is 443. Most companies allow outbound access to other sites on the Internet over SSL.

The most secure way for distributing a NetPhantom application over the Internet (or even Extranet) is to use SSL. The following topics are covered:

- Data integrity (by means of digital signing of transactions). An external intercepting program cannot change the data exchanged between the server and the client.
- Data security (the transactions are encrypted with e.g. a 128/256 bit algorithm). Another party cannot read the data.
- Server authentication (using a certificate on the server). The client is guaranteed to exchange data with a known source.
- Optional client authentication (using a certificate installed on the client). This guarantees that the client is allowed to exchange data with the server. The drawback is that a certificate file must be shipped out to the client and then installed.



The diagram above shows two physical machines for the NetPhantom Server and the web server. The two servers can be merged into one machine, but if the NetPhantom integrated Web Server is not used, the network must be configured in such a way that two IP addresses are assigned to the server machine.

8.2 HTTP Tunneling and the NetPhantom Web Server

To support HTTP tunneling (as well as SSL) fully, a Web Server is included in the NetPhantom server. To enable HTTP tunneling, the NetPhantom Web Server must be enabled. There is a section in `server.ini` called `[WebServer]`. Enable the HTTP tunneling with the entry:

```
httpTunneling = 1
```

HTML Document for HTTP Tunneling

The following HTML document is a sample of how to use HTTP tunneling. It is assumed that the HTML document is loaded from the web server at address *http://your.webserver.com/somedirectory*.

```
<html>
<body>
<applet width = 800 height = 600
codebase      = "http://your.netphantomserver.com/"
code          = se.entra.phantom.client.Phantom
archive       = "NetPhantomClient.jar">
```

```
<param name   = "http" value = "1">
<param name   = "port" value = "80">
</applet>
</body>
</html>
```

The reason for having a codebase *http-address* that differs from the address *http://your.webserver.com/somedirectory* is that one wants to use the normal web server to serve the number of HTTP requests that may be received from *http://your.webserver.com/somedirectory*. The applet is loaded from *http://your.netphantomserver.com/*, because the applet will otherwise get a Java Security Exception when it wants to communicate with the NetPhantom Server.

8.3 SSL – Secure Socket Layer

This version of NetPhantom provides support for SSL from Baltimore Technologies. See *Chapter SSL – Secure Socket Layer* for more information.

HTML Document for SSL

The following HTML document is a sample of how to use SSL encryption. It is assumed that the HTML document is loaded from the web server at address *https://server.com/somedirectory*.

```
<html>
<body>
<applet width = 800 height = 600
codebase    = "https://server.com/"
code        = se.entra.phantom.client.Phantom
archive     = "NetPhantomClient.jar">
<param name = "SSL" value = "1">
<param name = "port" value = "443">
<param name = "RESURL" value = "http://server.com/">
</applet>
</body>
</html>
```

The example above also defines all images (and other resources) for the client to be loaded over `HTTP` instead of `HTTPS` (using the `RESURL` parameter). This increases the download speed for all new images.

9 NetPhantom Web Server

The NetPhantom Server optionally runs a Web Server fully integrated with the server process. This web server is activated by default.

9.1 Hypertext Transfer Protocol – HTTP

HTML documents as well as resources that are referred to by the HTML document (such as GIF images) are transported between the server and the client using HTTP.

An HTML document is sent once from the server to the requesting client. Only when the client issues a new request or posts data by means of an HTML form, can the server reply with a new or updated HTML document. The communication between the requesting client and the replying server is disconnected once the document is transferred. This is the nature of the normal HTML data transport (HTTP).

A new TCP/IP connection is created each time a request is posted or sent to the server. Many TCP/IP data packets are sent back and forth between the client and the server to retrieve just the document. If this document contains images, this process may be repeated many times over. The NetPhantom Java Client performs so much better because it has a permanent connection to the server.

HTTP communication usually takes place over TCP/IP connections. The default port is TCP 80, but other ports can be used. This does not preclude HTTP from being implemented on top of any other protocol on the Internet, or on other networks.

In HTTP/1.0, most implementations used a new connection for each request/response exchange. In HTTP/1.1, a connection may be used for one or more request/response exchanges, although connections may be closed for a variety of reasons.

The web server uses all defined ports of the NetPhantom server to listen for HTTP requests. If a request is of NetPhantom Client HTTP Tunneling format, this request will be redirected to the HTTP Tunneling engine of NetPhantom Server that will handle the communication with the client from then on.

9.2 NetPhantom Web Server

The Web Server in NetPhantom has the following built-in features:

- Supports HTTP/1.0 and HTTP/1.1 protocols.
- Tight, fast and efficient integration with the “normal” NetPhantom application on the server
- HTML document parsing and substitution using variables in the document
- Servlets or CGIs
- HTML-included Servlets or CGIs
- Server Side Include of documents
- Resource-sensitive LRU (Least Recently Used) caching of (pre-parsed) HTML documents and other resources such as GIF images
- SSL with 128/256-bit data encryption or more
- User authentication on CGI/Servlet, file and/or directory level
- Redirection service on file, directory, or server level

- The Web Server can serve as a replacement of a normal web server, but is mainly intended to provide the integration of HTML documents with the NetPhantomized host application(s)

- The NetPhantom Web Server complies with HTTP/1.1 but only implements the following HTTP requests GET, HEAD or POST. The other requests (OPTIONS, PUT, DELETE, TRACE and CONNECT) are rejected with the status code 405 (Method Not Allowed).

9.3 NetPhantom and HTML

The web server reads and parses all HTML documents. The documents can contain variable data that is replaced with the required data when the document is sent to the client.

An HTML document is assumed to be "HTML" when the file extension exists in the HTML document type definition in `mimetype.ini` as:

```
text/html        html htm shtml shtm
```

For in-depth information about NetPhantom and HTML, see *NetPhantom Markup Tags in HTML*.

9.4 Servlets or CGIs

The NetPhantom Web server supports Java Servlets or CGIs by means of two interfaces:

To use an HTML-included CGI, define it in the Web Server configuration in the server administration program. Then create a CGI resource that will set the user rights and the location of the CGI on the web server.

The typical usage of Servlets or CGIs is in processing responses of HTML forms, dynamic (content) HTML documents containing dynamic data, etc.

For more information, see the *NetPhantom Java API*.

9.5 Introduction to User Authentication

This introduction is divided into two basic sections:

1. *NetPhantom Web Server Authentication*, which explains its purpose.

2. *HTTP Authentication*, which introduces the reader to the context of HTTP authentication. Additional information can be found in the Network Working Group's Request for comments RFC2616 and RFC2617.

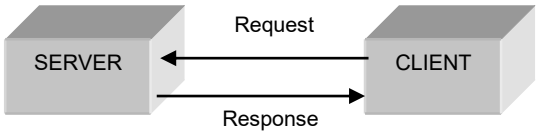
Purpose of NetPhantom Web Server Authentication

Some server resources must be protected for various reasons. The NetPhantom Authentication module was created to make it possible to protect resources handled by the NetPhantom Web Server from unlimited access.

HTTP Authentication

The HTTP authentication procedure consists of two parts: the request message from a client and the response message from the server.

The request message from the client contains information that the server needs in order to determine whether the client is authorized to make the request in question. The request can, for example, be a request to get information from the server or a request to send information to a server program or directory.



If the client is authorized to make the request, the server fulfils the request. Otherwise, the server responds with a 401 (Unauthorized) response message, which contains information about:

- The server resources for which authentication can be performed.
- The type of authentication scheme supported by the server (*Digest* or/and *Basic*).
- Data (data typical for supported authentication scheme) the next request should contain.

The mechanism of client/server communication by which it is possible for a server to authenticate a client's request is the challenge-response authentication mechanism defined in the HTTP/1.0 and HTTP/1.1 protocols (see *NetPhantom Web Server – Technical* below).

9.6 Using NetPhantom Web Server Authentication

Certain server resources must be protected for a variety of reasons. The NetPhantom Authentication module protects resources by providing tools to limit access.

The resources handled by a NetPhantom Web Server are defined and configured with the NetPhantom server administration program. Select the menu item **Server – Configure web server**, and click the **Resources** tab. The notebook page for configuration of resources allows defined resources to be protected from unlimited access by checking the option **SSL is required** and/or the option **Authentication**.

When the **Authentication** option is checked, authentication will be performed on all client requests for that resource. The **Allow Basic** option will also be enabled.

The **Allow Basic** option activates Basic authentication. This means that browsers using HTTP version 1.0 will be able to send authentication requests for the resource in question. If **Allow Basic** is left unchecked, only Digest authentication will be allowed. Thus, all browsers using HTTP version 1.0 will *not* be able to send authentication requests for the resource in question.

A resource that requires authentication is connected to one or more user groups that are authorized to have access to that resource. Definition and administration of user groups is done via the menu item **Server – Manage Groups**. The user group must be activated by selecting the **Group is active** option in the **Groups and Group Definition** dialog box.

Once a group has been created, the users can be added using the **Server – Manage users** menu item.

9.7 NetPhantom Redirection Service

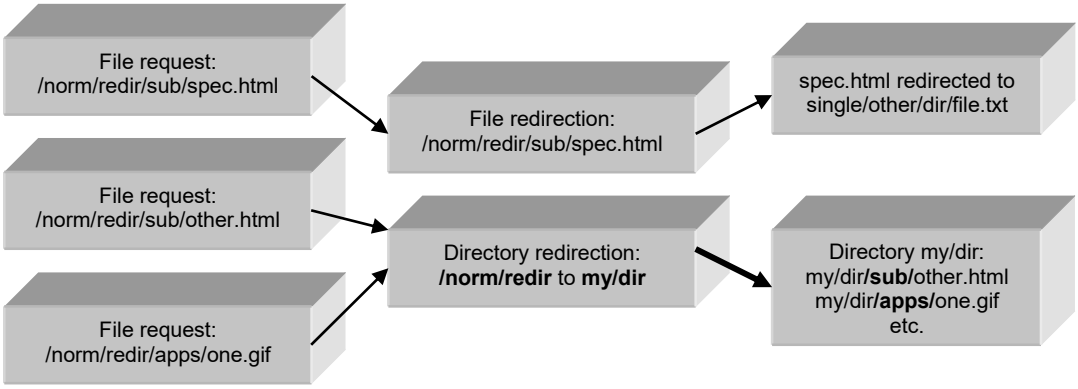
The NetPhantom Web Server can redirect individual files or a whole directory to any other file or directory of the server's file system. The requesting parties (web browsers) just refer to the files or directories as a normal resource.

The definition of a redirection is done in the web server **Resource Definition** in the server administration program (menu item **Configure web server – Resource** tab). This definition contains information for optional authentication and SSL usage but can also specify a server-local file resource.

For a specific file redirection, the server-local file resource must point to a single file. This means that a directory redirection must be done to a server-local directory resource on its file system.

When the NetPhantom Web Server processes a request, the entire file is first checked for a redirection. If no redirection is found, a search for a redirection of each subdirectory of the request is performed.

The following diagram shows a typical file redirection scenario:



9.8 NetPhantom Web Server – Technical

Introduction to the HTTP Authentication Schemes

HTTP provides a simple challenge-response authentication mechanism that may be used by a server to challenge a client request and by a client to provide authentication information.

An authentication scheme consists of two major logical parts:

auth-scheme which represents the "name" of the authentication scheme,

auth-params which represents a set of parameters, -token and value pairs, or –token and token pairs, that must be set according to the authentication scheme.

The following two sections contain a simplified explanation of *Basic* and *Digest* authentication schemes, which includes a brief description of the auth-params for the two schemes. This is intended to facilitate the reader's understanding of the specification of the NetPhantom Web Server Authentication presented later in this document.

Basic Authentication Scheme

The protocol referred to as "HTTP/1.0" includes the specification for a Basic Access Authentication scheme. The Basic Access Authentication scheme is not considered to be a secure method of user authentication since the username and password are passed unencrypted over the network.

- The "basic" authentication scheme is based on the model that the client must authenticate itself with a user-ID and a password. To receive authorization, the client sends the userid and password, separated by a single colon (":") character, within a base64 encoded string in the credentials.

The challenge response message from the server that makes it possible for a client to request authentication contains a WWW-authentication header.

```
WWW-Authenticate = "WWW-Authenticate" ":" "Basic" basic-challenge
```

Where `Basic` represents the *auth-scheme* and the basic-challenge represents the *auth-params*.

Auth-params for Basic Authentication Scheme

- **realm:** A string to be displayed to users so they know which username and password to use.

Digest Authentication Scheme

The authentication procedure of the Digest Access Authentication scheme does not send the password in clear text. Instead, it sends a checksum that consists of a digest of a set of parameters referred to as the authentication scheme's auth-params.

The challenge response message from the server that makes it possible for a client to request authentication contains a WWW-authentication header.

```
WWW-Authenticate = "WWW-Authenticate" ":" "Digest" digest-challenge
```

Where `Digest` represents the *auth-scheme* and the digest-challenge represents the *auth-params*.

```
Digest-challenge = 1#(realm | [domain] | nonce |
                      [digest-opaque] | [stale] | [algorithm])
```

Auth-params for Digest Authentication Scheme

- **realm:** A string to be displayed to users so they know which username and password to use.
- **domain:** A comma-separated list of URIs (URI = uniform resource identifier). The client can use this information to determine the set of URIs for which the same authentication information should be sent.
- **opaque:** A string of data specified by the server that should be returned by the client unchanged. It is recommended that this string be base64 or hexadecimal data.
- **nonce:** A server-specified data string which may be uniquely generated each time a 401 response occurs.
- **stale:** A flag indicating that the previous request from the client was rejected because the nonce value was stale. If stale is TRUE (in upper or lower case), the client may simply wish to retry the request with a new encrypted response, without re-prompting the user for a new username and password.
- **algorithm:** A string indicating a pair of algorithms used to produce the digest and a checksum. If this is not present it is assumed to be "MD5".
- **qop-options:** A value indicating the "quality of protection" the server supports. The value "auth" indicates authentication protection (the checksum sent from client does not include the values of the actual content of the HTTP message). The value "auth-int" indicates authentication with integrity protection (the checksum sent from client includes the values of the actual content of the HTTP message).
- **auth-param:** A value not used today that exists for future extensions of the authentication scheme.

Example of Digest Authentication Messages

The following example (as presented in the Network Working Group's RFC2616 and RFC2617) assumes that an access-protected document is being requested from the server via a GET request. The URI of the document is "http://www.nowhere.org/dir/index.html". Both client and server know that the username for this document is "Mufasa", and the password is "Circle Of Life" (with one space between each of the three words). The first time the client requests the document, no Authorization header is sent, so the server responds with:

```
HTTP/1.1 401 Unauthorized
WWW-Authenticate: Digest
  realm="testrealm@host.com",
  qop="auth,auth-int",
  nonce="dcd98b7102dd2f0e8b11d0f600bfb0c093",
  opaque="5ccc069c403ebaf9f0171e9517f40e41"
```

The client may prompt the user with the username and password, after which it will respond with a new request, including the following:

```
Authorization: Digest username="Mufasa",
  realm="testrealm@host.com",
  nonce="dcd98b7102dd2f0e8b11d0f600bfb0c093",
  uri="/dir/index.html",
  qop=auth,
  nc=00000001,
  cnonce="0a4f113b",
  response="6629fae49393a05397450978507c4ef1"
```

10 NetPhantom Load Balancing

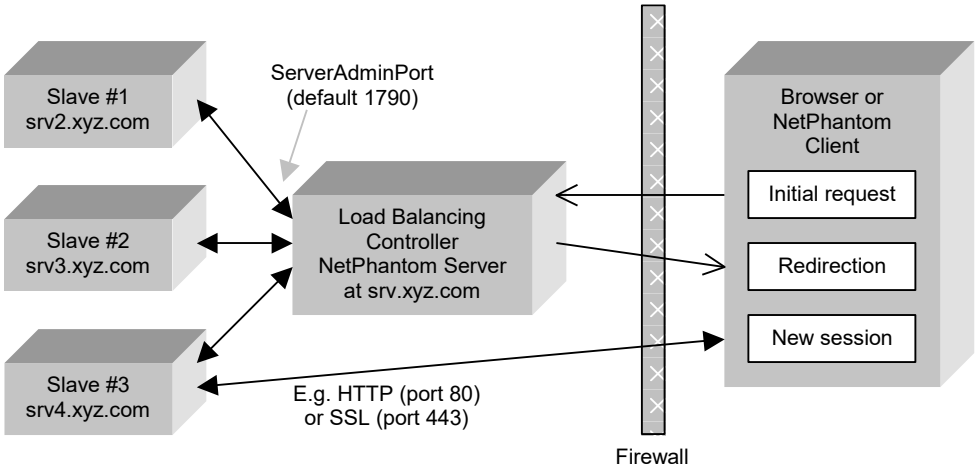
10.1 Introduction

The NetPhantom Load Balancing system enables several NetPhantom servers to balance the load of concurrent users. It uses the technique of redirection as opposed to “proxy” because it is more efficient in terms of server load. One NetPhantom Server acts as the “load balancing controller” and the other servers as “load balancing slaves”.

> **Note:** A NetPhantom Server is considered as a process running on a physical server with a predefined IP address. Several NetPhantom Servers can run on the same physical machine but require different IP addresses (for load balancing). This could be necessary if the Java Virtual Machine load becomes too high.

The slave servers connect to the controller and constantly inform it of the current number of users and the configured maximum number of concurrent users. They also inform the controller of the number of HTTP requests currently in queue for processing.

If user authentication is used for web resources or NetPhantom applications, the user validation is performed in the slave. For instance, if a user changes his/her password, the server where this operation took place informs the other servers of this event, resulting in an immediate update of the user databases. If multiple changes to the groups and/or users are made, these changes are also distributed to the other servers. The group/user database is also transferred to the slaves when they first contact the controller.



The Load Balancing system does not provide a “fail-over” between two servers because very often there is an underlying 3270 or 5250 host session that cannot be “moved” to another server.

> **Note:** Load balancing is not enabled if the web server is not started.

10.2 Load Balancing Techniques

Load balancing is done by one of the means in the table below. To enable load balancing, check the option **Load balancing** on the **Resources** page of the Web server configuration dialog for the resource(s) in question.

Resource type	Load balancing technique
FS – File System **FSU** – File System (Unparsed) **CGI** – Common Gateway Interface	These resource types are "pure" web resources such as HTML documents, GIF images, etc. Load balancing redirects (if required) the request to the server that has the lowest amount of queued web server requests compared to the configured maximum.
CGIA – CGI for Applications **FSA** – File System (Application) **NA** – NetPhantom Application **WA** – Web Application	CGIA is normally a CGI running a session-based application affecting the number of concurrent users of HTML type. FSA is typically used to indicate an HTML document that contains the NetPhantom Java Client applet definition. As the applet normally cannot communicate with a server other than the one from which the HTML document was loaded, the redirection to the slave server must be done for the HTML document itself, prior to the applet loading in the client. Load balancing for these resource types is done by redirecting (if required) the request to a server that has the least number of concurrent users in comparison to the configured maximum amount.
RD – Redirection	Load balancing does not apply for this resource type.

10.3 Load Balancing Redirection

Redirection from the controller to a slave is done for HTTP requests using the HTTP status code *302 – Found*. For NetPhantom Clients running as Java applications, the application is requested to change servers (if required) to the slave in question. The client may not respond to this request if the slave server cannot be contacted for some reason. The SERVERADMIN application is *never* redirected. If the client is running as a Java applet, it will not be redirected; therefore the HTML document must be redirected using the resource type FSA – File System (Application).

Redirection uses the name of the slave server as set in the **Web server** configuration panel on the **Address** page.

> **Note:** Redirection from the controller server may not be required in all cases; the controller is itself included in the available servers list when the most appropriate server is selected.

10.4 Slave/Controller Server Communication

The slaves communicate with the controller server constantly over the Server Administration Port (default 1790). If a slave cannot contact a controller when the web server is started (i.e. when load balancing also is started), it will try to connect to the controller in intervals of 10 seconds. If a slave loses the connection to its controller, it will try reconnecting every 10 seconds. Load balancing is stopped when the web server is stopped.

Load Balancing Qualification

For a resource type to be load balanced, the following criteria must be met:

- the *Port ID* must match between controller and slave (e.g. the port number does not need to be the same),
- the port must be started and enabled for load balancing.

For performance and controller-slave communication reasons, load balancing do not take into account if the slave is not configured as the controller for a resource type (this includes a NetPhantom or Web Application and also a disabled NetPhantom application). This would mean that if you want to disable a NetPhantom application, you should do this in the controller.

Controller Information to Slave

The controller server informs the slaves of the following events:

- user and group definitions when slave starts or when an administrator has modified these definitions,
- password change for a specific user (this can originate from the user changing his password on the controller or on another slave),

Slave Information to Controller

The slave server informs the controller of the following events:

- port states, i.e. if they are started and used for load balancing or stopped (when load balancing status for a port is changed, it is always stopped prior to a restart),
- change of user password,
- “new client connections” accept or reject state,
- concurrent user count,
- maximum concurrent user count,
- web server request queue count,
- maximum web server queue count.

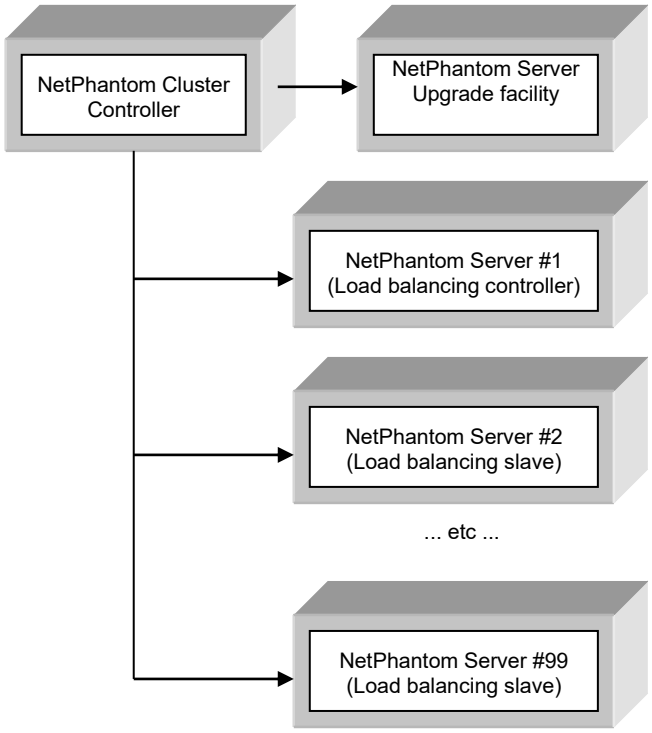
11 NetPhantom Cluster Controller

This section requires knowledge of NetPhantom Load Balancing (see *Chapter NetPhantom Load Balancing* for more information).

The NetPhantom Cluster Controller is a program that runs several NetPhantom Server processes on the same machine using load balancing. It manages the servers by starting them as new processes and allows them to be restarted individually. Keep in mind, the cluster controller is a program for running multiple NetPhantom servers and should not be confused with the load-balancing controller, which is a NetPhantom server.

Clustering several servers is important for large installations with thousands of concurrent users because a server process in a Java Virtual Machine can be limited in resources, e.g. physical memory, amount of threads or socket connections.

The Cluster Controller allows all NetPhantom Servers to share a single `server.ini` file with all the settings for the servers.



The first NetPhantom Server acts as a load balancing controller and all other servers as slaves. The NetPhantom Servers may optionally all be load balancing slave servers to a controller server on another machine.

The Cluster Controller can also:

- Run the NetPhantom Server Upgrade facility.
- Run as a Windows Service (under Windows Server 2008(R2), 2012(R2), 2019, 2022 or Windows 7 to 11).
- Allow the NetPhantom Servers to use another Java VM (e.g. *JRE 1.8.0 64-bit* for the Cluster Controller and *JDK 11 with server VM* for the NetPhantom Servers).

11.1 Executing the Cluster Controller

To start executing the server, run the following:

```
Java [params] -classpath .
   NetPhantomClusterController [restart] [serverIniFile]
```

The option *restart* should only be used when running the Cluster Controller from a script or batch file and when the upgrade process is run outside the Cluster Controller (see *Server Upgrade*, below)

Memory Requirements

The Cluster Controller uses very little RAM, so it can be set to e.g. 8 MB using the –Xmx and –Xms parameters. If an *OutOfMemoryError* exception occurs, increase the amount of RAM for the process. The minimum value depends on the JVM and type (32 or 64 bit).

Licensing – Number of Users

When the Cluster Controller runs the servers, the configured maximum number of concurrent users and the warning count of concurrent users is divided by the number of servers in the cluster (rounded up to nearest multiple of 10).

This means, for example, that a maximum concurrent client count of 700 divided over 3 clustered servers will result in each server having the maximum concurrent client count set to 240.

The server license is only configured for the first server (and its IP address or host name) in the Server Administration Program.

Sample Batch File for Windows

```
@echo off

echo *** Starting the NetPhantom Cluster Controller ***
goto start

:upgrade
echo *** Upgrading the NetPhantom Server ***
java -classpath . UpgradeNetPhantomServer > upgrade.log
if errorlevel 1 goto uperror

:restart
echo *** Restarting the NetPhantom Cluster Controller ***

:start
java -classpath . NetPhantomClusterController restart
if errorlevel 1001 goto upgrade
if errorlevel 1000 goto restart
if errorlevel 1 goto error

echo *** The NetPhantom Cluster Controller has stopped executing ***
goto done

:uperror
echo *** Error upgrading the NetPhantom Server ***
goto done

:error
echo *** Error starting the NetPhantom Cluster Controller ***

:done
```

Sample Script File for UNIX

```
#!/bin/sh
# NetPhantom Startup script

COMMAND='java -classpath . NetPhantomClusterController restart'
RSTCMD='java -classpath . UpgradeNetPhantomServer'

echo "*** Starting the NetPhantom Cluster Controller ***"

while true
  do
    $COMMAND
    RC=$?
    if [ $RC -eq 1000 ] || [ $RC -eq 232 ]; then
      echo "*** Restarting the NetPhantom Cluster Controller ***"
    elif [ $RC -eq 1001 ] || [ $RC -eq 233 ]; then
      echo "*** Upgrading the NetPhantom Server ***"
      $RSTCMD > upgrade.log
      if [ $? != 0 ]; then
        echo "*** Error when upgrading the NetPhantom Server ***"
        exit 1
      fi
      echo "*** Restarting the NetPhantom Cluster Controller
            after upgrade ***"
    elif [ $RC -eq 0 ]; then
      echo "*** The Cluster Controller has been stopped normally ***"
      exit 0
    else
      echo "*** Error when starting or restarting the
            NetPhantom Cluster Controller ***"
      exit 1
    fi
  done
```

11.2 Running the Cluster Controller as a Windows Service

The NetPhantom Cluster Controller can be run as a Windows Service under Windows Server 2003/2008(R2)/2012(R2)/2016/2019/2022 or Windows 7 to 11.

The purpose of having the server running as a Windows Service is that services do not require a user to be logged on to run. The server process runs in a daemon mode without console or GUI.

Please see NetPhantom Windows Services for details of the service installation process.

11.3 Special Server Functions with Cluster Controller

There are three functions that behave differently for NetPhantom Servers when used with the Cluster Controller as described below.

Server Shutdown

When shutting down a server with the server administration program or with the remote command line utility, the server that received the command will shut down. All servers must be shut down. When all servers running in the Cluster Controller are shut down, the Cluster Controller process will exit (or the Windows Service will stop).

Server Restart

When restarting a server running in the Cluster Controller using the server administration program with option **Hard JVM restart**, the server in question will perform this function alone, except when a pending *Server upgrade* is present (see below).

Server Upgrade

The server upgrade function, using the server administration program, should be carefully controlled by the administrator when using the Cluster Controller.

Improper handling could lead to inconsistencies in various files, typically Java classes.

Steps to perform a Server Upgrade

1. **Shut down** all servers except the first one.
2. Perform the **Server – Upgrade** using the server administration program.
3. Select **Server – Restart** with option **Hard JVM Restart** selected as suggested by the upgrade routine.

When the server is stopped, the Cluster Controller will perform the Server Upgrade function internally (if configured for it), otherwise exit the process with the return code 1001 (perform external Server Upgrade). When the upgrade process is finished, the Cluster Controller will restart all the servers again, even the ones that were shut down.

Internal versus External Upgrade

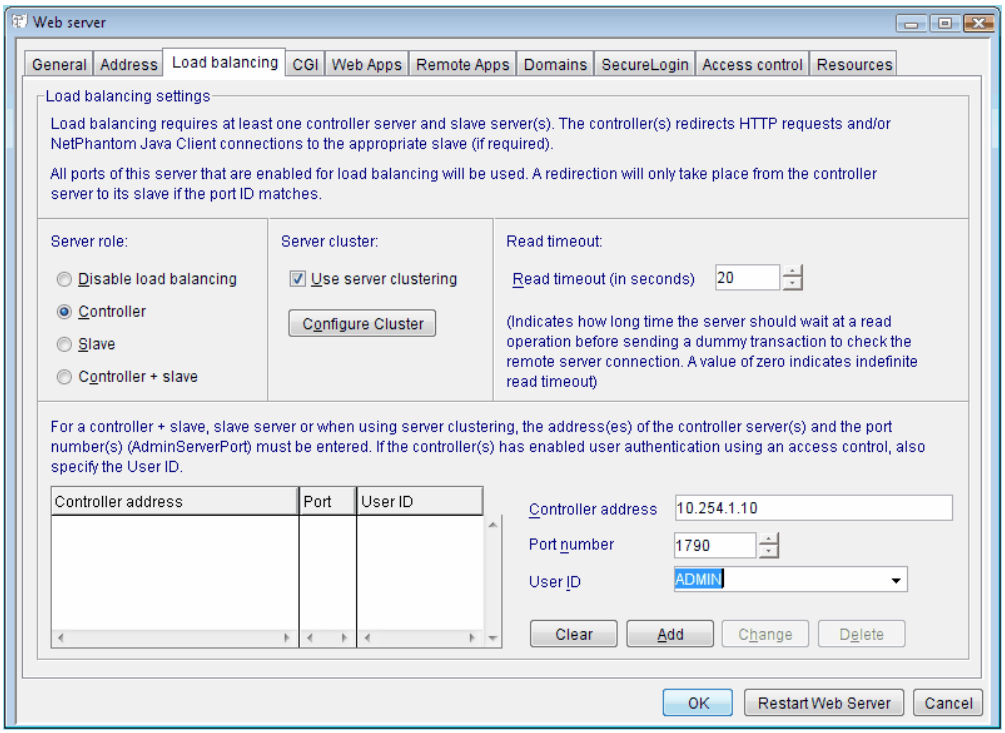
Internal Upgrade is required when running the Cluster Controller as a Windows Service. Otherwise, the External Upgrade should be used. The External Upgrade updates both the Cluster Controller and the Server.

If you run an internal upgrade when *not* running the Cluster Controller as an Service, the Server will be upgraded, but the Cluster Controller will not be updated until it is stopped and restarted.

11.4 Configuring the Cluster Controller

The best way to configure the Cluster Controller is to run *a single server* (stand-alone), then start the Server Administration Program. After the server is configured, the Cluster Controller can be started using the new settings.

The Cluster Controller is configured via the NetPhantom Server Administration program from **Load balancing** notebook page with the menu item **Server – Configure Web Server**. It is important not to confuse the Server that functions as load balancing controller with the program Cluster Controller.

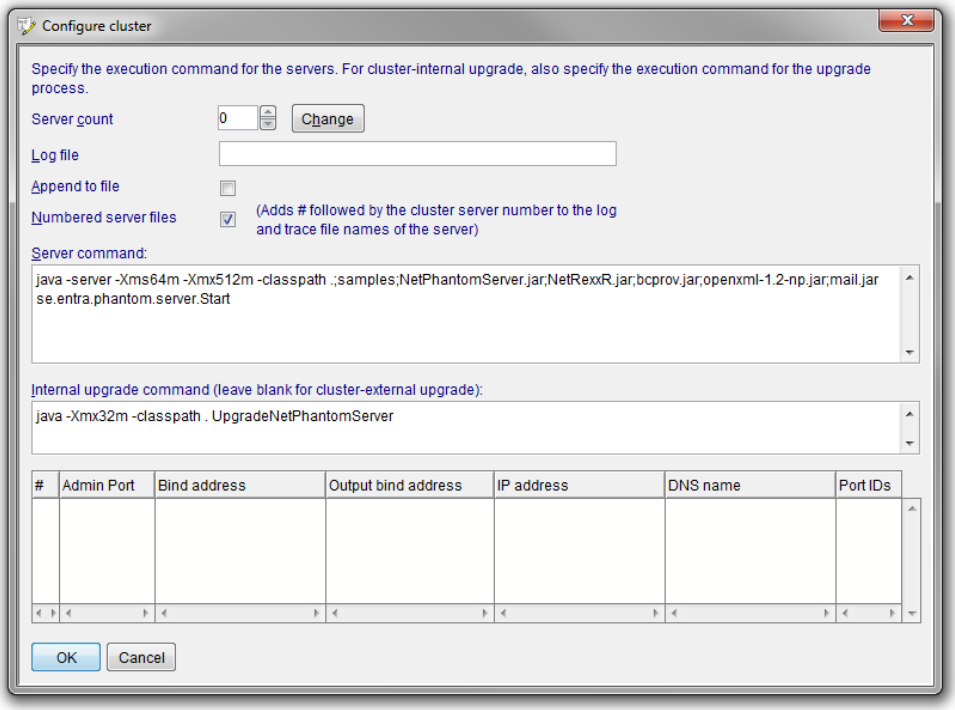


*The Load balancing notebook page in the **Configure Web Server** panel.*

The option **Use server clustering** must be checked, and the cluster configured using the push button **Configure Cluster**.

The first NetPhantom Server that will be started from the Cluster Controller is either a load balancing **Controller** or a load balancing **Slave**. All remaining NetPhantom Servers will become *slaves*.

Make sure to specify the **Controller address** and **Port number** (see image above) of the *first* NetPhantom Server if the option **Controller** is selected, otherwise (if the option **Slave** is selected) the address and port number of the controller server in question. The latter case is used to cluster servers over multiple physical (server) machines.



*It is important to specify the correct JVM execution command and parameters for **Server command** and optionally **Internal upgrade command**.*

1. Enter the **Server count** and press **Change**.

2. Fill in the **Server command** with the appropriate Java VM program while also setting up the *classpath* and optionally the memory requirements (e.g. `-Xmx512m`).

3. If *Internal upgrade* is used, specify the **Internal upgrade command**.

4. For each server, specify the **Admin Port**, the **Bind address** used for all the ports, the **IP address** used for the Web Server and the **DNS name**, also used for the Web Server.

Notes:

- The definition for each server in the list will replace the value specified for the server administration port, the server administration bind address, the bind address(es) in port(s) definition.

- When running the Cluster Controller, certain combinations of Java VM (or when running as a Windows Service) do not allow use of the server GUI (set `ServerGUI=0` in `server.ini`).

Event Log and Trace Files

The event log(s) and trace file(s) are specified in `server.ini`, for example:

```
eventFile1=NetPhantom.log
eventFile2=NetPhantom2.log
```

and

```
traceFile1=NetPhantom.trc
traceFile2=NetPhantom2.trc
```

When several servers are clustered, the number or index of the server is appended to the file name, e.g. NetPhantom Server number 3 will have its event log files set to

`NetPhantom.log3` and `NetPhantom2.log3`. The same principle applies to the trace files.

Note that the server events are logged only in these files when running the Cluster Controller as a Windows Service, i.e. the server events will not be placed in the Windows Event Log. Only the Cluster Controller log is output to the Windows Event Viewer *Application* log.

Manual Configuration in server.ini

The Cluster Controller configuration is read from the `[Cluster]` section in `server.ini`.

ServerCount = *nn*	The count of NetPhantom Servers in the cluster. The value must range from 1 to 99.
ExecuteCommand = *java ... se.entra.phantom.server.Start*	The execute command to start a NetPhantom Server. The Java VM can differ from that of the Cluster Controller and should also have the *classpath* set correctly.
UpgradeCommand = *java ... UpgradeNetPhantomServer*	If *Internal upgrade* is used, i.e. the Cluster Controller runs the upgrade process, the *UpgradeCommand* must be specified. Leave it blank if the upgrade process is run from a batch file or script outside of the Cluster Controller.
AdminPort.*nn* = 1790	Specifies the port number for the NetPhantom Remote Administration Commands. This value will override the port setting for the server in question.
BindAddress.*nn* = *ipAddress*	Specifies the IP address that will be used when creating the *AdminPort*. This value will override the address setting for the server in question. This value will also override any bind addresses for the server port configuration, i.e. all ports will bind to this address.
IPAddr.*nn* = *ipAddress*	The IP address used by the Web Server. Normally, set this value equal to the *BindAddress* above. This value will override the setting in the `[WebServer]` section, item *`HostAddress`* in `server.ini` for the server in question.
DNSName.*nn* = *host_name* or *ipAddress*	The name of the server for multi-homed networks. This value is used by the Web Server for redirection, etc. This value will override the setting in the `[WebServer]` section, item *`HostName`* in `server.ini` for the server in question.

Example of Manual Configuration in server.ini

```
[Cluster]
serverCount=4

executeCommand=java -classpath .;samples;NetPhantomServer.jar;NetRexxR.jar;
    acme-4j.jar;bcprov.jar;bcpkix.jar;bcutil.jar;openxml-1.2-np.jar;
    mail.jar;activation.jar se.entra.phantom.server.Start

upgradeCommand=java -classpath . UpgradeNetPhantomServer

adminPort.1=1790
bindAddr.1 =10.254.1.10
ipAddr.1   =10.254.1.10
dnsName.1  =netphantom.entra.se

adminPort.2=1790
bindAddr.2 =10.254.0.37
ipAddr.2   =10.254.0.37
dnsName.2  =netphantom2.entra.se

adminPort.3=1790
bindAddr.3 =10.254.0.38
ipAddr.3   =10.254.0.38
dnsName.3  =netphantom3.entra.se

adminPort.4=1790
bindAddr.4 =10.254.0.39
ipAddr.4   =10.254.0.39
dnsName.4  =netphantom4.entra.se
```

12 NetPhantom SecureLogin

This system provides means to securely authenticate a user before a NetPhantom application session is started. It also enables initial conversation with a legacy host system to bypass a login screen and go directly to the part of the system that displays user account information. Parameters and values can be assigned uniquely on a user basis. See the *Configuration* sections below.

12.1 GSM Phone Network – SMS Messaging

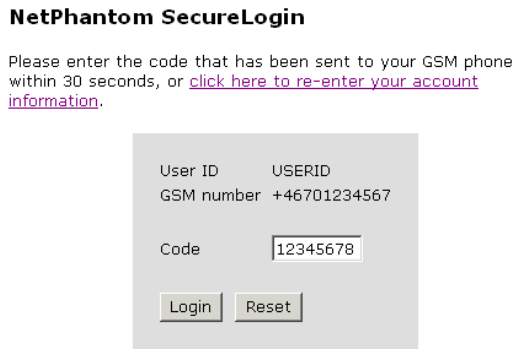
The system requires users to be registered at the NetPhantom server prior to being authenticated. An optional pin code for the user is also supported. Currently, authentication is done by means of a unique code that is generated and sent to the user over the GSM mobile phone network. A user can register one or several phone numbers.

Once the user receives the code as an SMS message to the phone, the code must be entered within a short period of time that is set in the Secure Login profile; the normal timeout is 30 seconds. The user is then authenticated, and the defined application is started. The application can be either a NetPhantom Java Client application or a NetPhantom Web Application (in HTML).

Initial screen where you enter the User ID, a GSM phone number and optionally a Pin code.

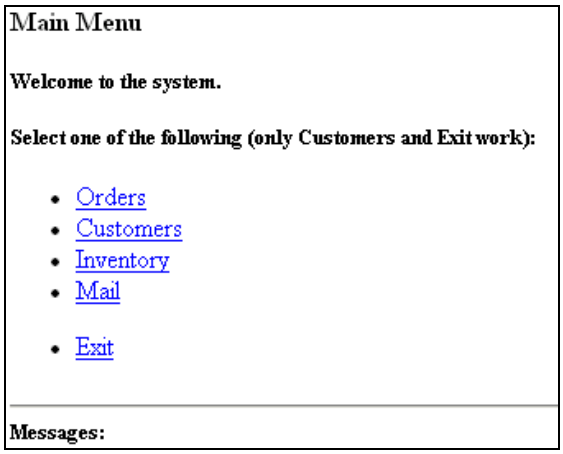
Once the login data is filled in, you press **Continue**. If the User ID is registered in the NetPhantom Server, the phone number will be checked against the valid phone numbers for the user. A Pin code can also be used as an additional check.

If the user information is correct, an SMS message containing a code will be sent to the mobile phone. If the user information is in error, no SMS message will be sent, but the dialog interface with the user will not reflect this.



The user must enter the code within a short period of time; otherwise new user information has to be entered causing a new code.

Once the user receives the SMS message containing the 8-digit code, it must be entered at the code field. If the code entered matches and is entered within the defined time limit, the application will start. In the NetPhantom SecureLogin sample, the "Normal tutorial" in HTML is displayed, bypassing the initial Sign on screen by entering a valid user ID and password programmatically.



The tutorial page is displayed after the Sign on screen has been programmatically completed (and therefore bypassed).

12.2 Configuring NetPhantom SecureLogin

The configuration is stored in the file `resources.ini` in the server's current directory. Before configuring the INI file, a CGI and a web resource must be configured.

Configuring a CGI

A web server "CGI" must be configured as the example below indicates (in Server Administration, **Configure – Web server – CGI** tab). Specify the **Java class name** as `se.entra.phantom.securelogin.SecureLoginCGI`, and make sure the option **HTML included CGI** is ***not*** checked.

The configuration of the "CGI" is done on the Server Administration, ***Configure – Web server – CGI*** *tab.*

Configuring a Resource

For each application using the SecureLogin system, a resource must be configured. The base name of the resource is used to read the profile definition from the `resources.ini` file.

The configuration of the resource is done in the Server Administration, ***Configure – Web server – Resources*** *tab.*

In the example above, the name of the resource is `/yourapp/SecureLoginSample.jsp`. The section in the `resources.ini` file will therefore be `[SecureLoginSample]`. This means that several applications using SecureLogin can be defined and maintained in a single file.

12.3 Configuration of Securelogin.ini File

The configuration file must contain a `[section]` with the name of the base of the web resource name (e.g. `/yourapp/SecureLoginSample.jsp` must be defined in the `[SecureLoginSample]` section).

Deleted=0	A flag used to tell the server that this section should be ignored, since you cannot literally delete the sections from the .ini file
maxAge = 30	Maximum age of a session in seconds. If this value is exceeded, the unique code is invalidated.
NpAppMaxAge = 300	The maximum age that a session can have while waiting for a Java Client to start the NetPhantom Application (default 5 minutes).
CountryCode = 46	Country code (without leading zeros, e.g. 46 for Sweden).
PinCode = 1	Flag indicating if pin code is required or not (zero "*0*" or one "*1*").
CodeLength = 6	The number of characters in the unique code.
welcomePage = ./securelogin/html/welcome.html	The welcome page (redisplayed for incomplete information).
passwordPage = ./securelogin/html/password.html	The password page for the code from the SMS.
failedPage = ./securelogin/html/failed.html	The general failure page (cannot send SMS, wrong code, etc).
SmsMessage = Please enter *@#CODE@* as the code within 45 seconds	You can edit the SMS message that will be sent to the user. The unique code is referenced with the *@#CODE@* variable.
colorOK = #000000 colorError = #CC0000	Colors for OK or error. These colors can be used in the HTML documents.
SendSMS = 0	Flag that tells the server whether or not to send a secure login SMS when the application is being debugged.

13 SSL – Secure Socket Layer

13.1 Introduction to Cryptography

Cryptography is a very broad term and can be defined as the art of keeping messages secret. In our information society, cryptography has become the basis for privacy, trust, electronic payments, and corporate security among other things. Once in the domain of pen and paper, cryptography has now evolved into the era of advanced mathematical algorithms and calculations – the cryptosystems. A cryptosystem is a locked box needing a key to open it to see what is inside. The message travels inside this box, allowing the message to be read only by someone possessing a key.

Cryptography and NetPhantom

NetPhantom uses this obfuscating technology to create secure dialog between the client and the NetPhantom server. There are many kinds of cryptosystems, each having their own advantages and disadvantages. This document will introduce the cryptosystems included with NetPhantom, helping you decide which is the best solution for you.

Basic Terminology

If someone sends a message to a receiver without any obfuscation of the message, this is called **plaintext** or **cleartext**. However, the possibility that someone reads it during transmission is present.

If cryptography is used, the disguised message is called a **ciphertext**. This message will only be readable to those with the secret key.

Encryption is the procedure to convert the plaintext to ciphertext and **Decryption**, the retrieval of the plaintext from the ciphertext.

A **cryptosystem** or **cipher** is a collection of mathematical algorithms created to disguise the plaintext from other people. These other people are enemies, opponents, interlopers, eavesdroppers, or third parties.

The **key** is the means of decrypting the message; it could be comprised of a password, passphrase or even a smartcard containing the key.

Basic Cryptosystems

In the documentation below, please note that it refers to general rules of cryptography and that the text is not always up to date in regard to today's standards of high encryption.

Some cryptographic algorithms rely on the secrecy of the algorithms but are not adequate for real-world use. Modern ciphers rely on a key to control encryption and decryption. Without the key, the message is impossible to decrypt.

There are two major classes of key-based cryptosystems: **symmetric** and **asymmetric**. A symmetric cipher uses the same key to encrypt and decrypt the message; it is also called **secret-key** cipher. An asymmetric cipher, on the other hand, uses one key to encrypt and another to decrypt; also called **public-key** cipher.

Symmetric and asymmetric ciphers both have their advantages and disadvantages. Symmetric ciphers have the advantage of speed. They are about 100 to 1000 times faster than typical asymmetric ciphers, making them the cipher of choice when you are encrypting files for local storage. The disadvantage is the key, since the same is used for encryption and decryption; it needs to be decided before both parties start transmitting ciphertext. Any third party that gets hold of the key will also be able to decrypt the ciphertext. The strength of a symmetric cipher is roughly measured by its key length. A 40-bit key is weak, whereas

a 256-bit key is strong. Well-known symmetric ciphers include DES, Triple DES, RC2, RC4 and IDEA.

Asymmetric ciphers use two keys, one for encryption the public key – and one for decryption – the secret key. Asymmetric ciphers permit the encryption key to be public, making them the preferred form of encryption between two parties. This allows anyone to encrypt a message but only the person holding the secret key to decrypt it. The advantage of asymmetric ciphers lies in this ability to distribute the encryption key without being detrimental to the security of the message. Well-known asymmetric ciphers include RSA, Diffie-Hellman, DSA and forms of Elliptic Curve Cryptography.

Digital Signatures

Digital signatures are used to verify the true origin and creation time of the message, thus adding two more uses of cryptography: **Authenticity** and **Timestamping**.

Asymmetric ciphers can be used to create digital signatures. A digital signature is a block of data created using someone's secret key. To verify the identity of the person, their public key can be used to create a match. The cipher is designed so that a valid signature is not possible to create without knowledge of the secret key.

To timestamp documents or messages the sender **signs** the message and its timestamp with their secret-key, certifying that the message existed at the stated time.

Cryptographic Hash Functions

Cryptographic hash functions are one-way encryption systems – they cannot be decrypted – used as a **tamper-proofing** agent. They are used to creating a **hash value** of a message: a unique value that verifies that a message has not been tampered with. Changing just one letter of the message will result in a completely different total hash value. Well-known hash functions include MD5 and SHA.

Digital Certificates

A digital certificate is an electronic identification. It is created for placement as reference with a **Certification Authority (CA)** or **Trusted Third Party (TTP)**. The Trusted Third Party verifies your identity by some out-of-band method (by telephone or meeting you in person) and finally signs your details. The Certification Authority attaches their Certificate to yours thus vouching for your identity.

A digital certificate contains the following information:

- Personal information about you such as your name, your company, and your email address.
- An expiration date.
- Your public key.
- A unique serial number used for reference in a database.
- A digital signature to guarantee authenticity.

13.2 The SSL System

Overview

SSL is the abbreviation of Secure Socket Layer. It is a hybrid of symmetric and asymmetric ciphers originally created by Netscape. It combines the speed of a symmetric cipher and ease of key distribution of an asymmetric cipher. The mixture of both techniques creates a cryptographic implementation that is perfect for exchange of lengthy streams of real-time information between two parties.

How Does SSL Work?

Using an asymmetric cipher to encrypt the whole stream of information would be very slow, so instead, the secret key for a symmetric cipher is encrypted and sent to the other party. Once the other party has the secret key, the stream of information can be encrypted using a fast-symmetric cipher.

SSL runs on top of TCP/IP and below the application protocols such as HTTP, LDAP, IMAP, or Telnet. It allows an SSL-enabled server to authenticate itself to an SSL-enabled client, allows the client to authenticate itself to the server, and allows both machines to establish an encrypted connection.

What Does SSL Provide?

SSL provides solutions to the fundamental concerns about communication over the Internet.

SSL Server Authentication allows a user to confirm a server's identity. SSL-enabled client software uses standard digital signatures to check that the server's ID is valid and has been issued by a certificate authority. This confirmation is important if, for example, the user is sending a credit card number over the network and wants to check the receiving server's identity.

SSL Client Authentication allows a server to confirm a user's identity. Using the same techniques as those used for server authentication, SSL-enabled server software can confirm that a client's certificate is valid. This confirmation might be important if, for example, the server is a bank sending confidential financial information to a customer, and it wants to check the recipient's identity.

An Encrypted SSL Connection creates a secure and confidential information stream between the SSL-enabled client and server. In addition, all data sent over the connection is protected from tampering or corruption. For an encrypted connection, SSL is not limited to any specific asymmetric or symmetric cipher; many different algorithms can be used.

What is TLS?

TLS is an updated version of SSL. There are no significant changes from the perspective of the user, and internally just a few algorithmic improvements have been made. TLS has a mechanism to fall back to SSL if either the client or server does not support the new protocol, so transition to TLS should be problem free. An interesting feature of TLS is that it uses only patent-free algorithms, so a TLS server can be TLS-compliant without encountering patent licensing problems.

13.3 NetPhantom's Cryptographic Module

What Does NetPhantom Implement?

NetPhantom uses the java standard toolkit JSSE (Java Secure Socket Extension) for creating SSL and TLS connections. It is fully compliant with SSL version 3.0 SSL and TLS versions 1.0 through 1.3 (depending on Java version) and provides full protection against the described PKCS attack. Today only TLS 1.1 (or higher) and onwards are accepted as secure protocols.

Cipher Suites

A cipher suite is an object that specifies the asymmetric, symmetric and hash algorithms that will be used to secure an SSL connection. The asymmetric algorithm is used to verify the identity of the server (and optionally, that of the client) and to securely exchange secret key information. The symmetric algorithm is used to encrypt the bulk of data transmitted across the SSL connection. The hash algorithm is used to protect transmitted data against modification during transmission. The length of the keys that will be used in both the symmetric and asymmetric algorithms must also be specified. For example, a cipher suite that indicates it is **export strength** will, in accordance with US export restrictions, restrict

the length of the symmetric keys to 40 bits and that of the asymmetric encryption keys to 512 bits.

When a client makes an SSL connection to a server, it sends a list of the cipher suites that it is capable of and willing to use. The server compares this list with its own supported cipher suites and chooses the first cipher proposed by the client that it is capable of and willing to use. Once determined, the continuing connection is encrypted with this cipher.

Cipher Suite Strength

When configuring a support factory, for either a client or a server, you must therefore specify a list of cipher suites. NetPhantom identifies five different strengths of cipher suite. These are:

- **Export-strength:** These suites contain 512-bit asymmetric algorithms (RSA or Diffie-Hellman) combined with 40-bit symmetric algorithms (RC2-40, RC4-40, DES-40) and either the SHA or MD5 hash algorithm.

- **Non-export-strength:** These are 1024-bit asymmetric algorithms (RSA or Diffie-Hellman) combined with symmetric algorithms of at least 56 bits (DES, triple DES, IDEA, RC4-128) and either SHA or MD5.

- **Strong:** These are 1024-bit asymmetric algorithms (RSA or Diffie-Hellman) combined with 128-bit+ symmetric algorithms (triple DES, IDEA, RC4-128) and either SHA or MD5. Newer algorithms for 256-bit are also available when the Java VM is configured for that using the Unlimited JCE Policy. See the Java JCE documentation for more information.

- **Mixed strength:** These cipher suites contain a full combination of all the above.

- **Export-1024-strength:** These cipher suites are experimental suites supporting exportable 1024-bit asymmetric and 56-bit symmetric algorithms.

If you have more stringent requirements, for example if you do not trust RC4 or MD5 or if you feel that triple DES is too slow, you can create your own list containing cipher-suites that meet your needs.

Cipher Suite Key Exchange

When choosing a cipher suite, in addition to its strength, you have the following choices of the asymmetric algorithm employed for key exchange, i.e., for exchanging the secret information that is used to derive encryption keys.

- **RSA:** These non-exportable cipher suites use the public key encoded in the server's certificate to encrypt a piece of secret data for transfer from the client to server. This secret is then used at both endpoints to compute encryption keys. The asymmetric operations involved in this key exchange are an encryption (at the client) and a decryption (at the server).

- **RSA-export:** These exportable cipher suites have two options: If the server's public key is exportable (<= 512 bits or <= 1024 bits, depending on the cipher suite) and can be used for encryption, then it is used as above. Otherwise, the server generates an exportable temporary RSA key pair and transmits the public key to the client, signed by its certificate (i.e. by the private key associated with the certificate). The client verifies the signature on the public key, and then uses it as above to transmit a secret to the server. The asymmetric operations involved in this key exchange are key pair generation (at the server; however, temporary keys can be reused so this cost is amortized over multiple connections), signing (at the server), verification (at the client), encryption (at the client) and finally decryption (at the server).

- **EDH-DSA/RSA:** These cipher suites use Diffie-Hellman for key exchange: The server generates a temporary Diffie-Hellman key pair (and associated parameters) of the

appropriate strength (512 bits for normal exportable cipher-suites, otherwise 1024). It then transmits the public key, signed by its certificate, to the client. The client verifies the public key with the server's certificate key (using RSA or DSA, depending on the cipher suite) and then itself generates a key pair. It transmits its public key to the server and then both client and server derive a shared secret from which they derive the encryption keys. The asymmetric operations involved in this key exchange are key pair generation (at the server; however, temporary keys can be reused so this cost is amortized over multiple connections), signing (at the server), verification (at the client), key pair generation (at the client) and finally secret derivation (at both client and server).

- **ADH:** These cipher suites use Diffie-Hellman for key exchange, exactly as above, however no certificates are used to prevent the server's public key from being compromised during the initial transfer. Asymmetric operations are, thus, temporary pair generation (at both client and server) and secret derivation (at both client and server).

Your choice of cipher suite(s) will depend upon your environment and security requirements. Typically, the RSA-based cipher suites are more useful than those using Diffie-Hellman, being faster and more widely compatible, however the latter are subject to fewer patent problems.

Ephemeral Diffie-Hellman can incur some connection delays because it requires expensive asymmetric parameters and key generation, however pregeneration and reuse of these keys can alleviate these problems somewhat.

You should only use Anonymous Diffie-Hellman if you require a non-authenticated connection and understand the possible consequences of such a connection: Certificates are not used, leaving the protocol vulnerable to a man-in-the-middle attack.

Choosing a Cipher Suite

To choose from among cipher suites, the following guidelines may help:

NetPhantom is a product created outside the United States, meaning you are not restricted to using export-strength cipher suites.

- At the server, if you wish to offer your clients **complete** control over the strength of the connection, you should use the mixed-strength cipher suites. This will allow the client to choose between weak or strong encryption but, for compatibility reasons, not the export-1024-strength suit; you must be explicit if you wish to support these.

- At the server, if you require that clients use strong encryption, you should use the strong or non-export-strength cipher suites.

- At the client, if you are not concerned with encryption strength, use either mixed-strength (preferred) or export-strength cipher suites. Export-only cipher-suites have the disadvantage that you cannot use them to connect to a strong server. Mixed-strength suites have the disadvantage that the server may elect to use a strong cipher, which is slightly slower.

- At the client, if you require strong encryption, you should use strong or non-export-strength cipher suites. The client cannot connect to an export-only server when using these suites.

Beyond the strength issue, you should pick a key exchange algorithm that is appropriate for your environment, as outlined above: RSA if the patent issues are not prohibitive. Otherwise, EDH/DSA for a patent-free, albeit slightly more costly key exchange. Finally, ADH if you do not want to use certificates and understand the security implications.

CA Certificates

CA certificates are certificates that supply NetPhantom with the identities and public keys of known and trusted certification authorities. When NetPhantom receives a certificate from a peer, it verifies the certificate against its trusted CA certificates. If the certificate was issued by a trusted CA and the certificate signature matches the CA's public key, it is accepted, otherwise NetPhantom will issue an error.

When a client connects to a server, the server will transmit its personal certificate. The client will then verify this server certificate against its known and trusted CA certificates. The client must have access to the certificate of the CA that ultimately issued the server's certificate, or it will reject the certificate. Typically, server certificates are issued by a select few top-level CAs, so just a couple of CA certificates will usually serve most purposes.

SSL also supports a client authentication facility whereby a client must also present its own certificate to the server for inspection. In this case, CA certificates must also be installed in the server. Unlike server authentication, however, the server indicates to the client which CAs it trusts, and the client must present a personal certificate issued by one of the indicated CAs.

CA certificates can be loaded from a file system using various utility functions provided by NetPhantom and installed in a support factory with the appropriate support functions.

Personal Certificates

Personal certificates are certificates that supply NetPhantom with the identity and public key of the server or client itself. Personal certificates are always associated with a private key that allows NetPhantom to perform asymmetric signing and decryption. The combination of a personal certificate and its associated private key is referred to, by NetPhantom, as an Identity. In fact, an identity is usually associated with a sequence of additional certificates; starting with the certificate of the entity that issued the personal certificate, followed by the certificate of the next issuer in line, and so on, up to the self-signed certificate of the top-level issuing CA. For example, Bob may have a personal certificate issued by his boss Jim, whose certificate was issued by the top-level CA Sheila. In this case, Bob will present a *chain* of certificates Bob, Jim, Sheila. Then, anyone who trusts either Jim or Sheila can verify Bob's certificate against his or her own list of trusted CAs.

In a basic NetPhantom setup, only servers require that an identity be installed. When a client connects to a server, the server will transmit its personal certificate (along with any others in the certificate chain) and use the associated private key to prove its identity to the client and to exchange secret key data. This personal certificate should include a chain of certificates up to a CA that the client will trust. In an intranet environment this can be a local CA; in an Internet environment it should be one of the standard top-level CAs.

If client authentication is used, the client also requires that an identity is installed. When the server requests client authentication, the client will return its certificate chain and use the associated private key to prove its identity to the server. The client must be able to present a certificate that was issued by one of the CAs trusted by the server; it is possible to install multiple identities and then whichever matches one of the requested CAs will be used.

Identity Keystores

NetPhantom manipulates identities using Identity Keystores. An identity keystore is a collection of identities, which are themselves an associated private key and certificate chain.

To create an identity keystore, you must obtain and install identities. NetPhantom provides support for loading identities from the file system, in DER encoding, in the PKCS#12 format (.p12 or .pfx).

Certificate Choice

It is important that a server's personal certificate strength matches any cipher suites that are used. It is somewhat futile to use full-strength (1024/128 bit) security with a weak certificate (512 bit). On the other hand, using export-crippled (512/40 bit) security with a strong certificate (2048 bit) provides only strong assurances of authenticity, and has a time cost of periodic temporary asymmetric key generation at the server. In normal use, you may wish to match the cipher suites you will offer to the certificates you provide.

The trend is towards stronger certificates. Today most certificate authorities do not offer certificates weaker than 2048 bit or lower.

NetPhantom provides some degree of extra support for this issue. If you install two identities, one strong and one weak, NetPhantom will automatically use whichever is appropriate for the negotiated cipher suite. If you wish to only use a strong certificate, or only use a weak certificate, install a single identity during setup.

Client certificates are only used for authentication purposes and so are not subject to strength limitations. The client should install the strongest personal certificate that it has available. In an intranet environment, it need only install an identity issued by the trusted intranet CA. In an Internet environment, it should install identities issued by all certification authorities that it has available because during the SSL handshake, present a personal certificate issued by a CA that is trusted by the server, which may not be known ahead of time.

13.4 NetPhantom SSL Configuration

Certificate Creation

Before the server can be configured for SSL, you have to have a private key and a certificate that contains the corresponding public key. If you don't already have a certificate, you can either apply for a certificate from a CA (Certificate Authority), or you can create a self-signed certificate. The key pair, as it is known, need only be generated once. After that, it is valid until the certificate expires.

There are different approaches to generating your key pair and creating a certificate from the public key:

- Get the CA to generate the key pair and certificate, and then send them to you.
- Generate the key pair locally. Create a "Certificate request" containing your private key and send it to the CA.
- If you are maintaining a network with several servers, generate all their key pairs centrally, get them all certified, and distribute them all simultaneously.

We would usually recommend generating the key pair locally. This ensures that the private key data only ever exists on your machine, which is obviously much better for security purposes. It is also the situation that SSL was designed for. However, it means that every server individually must seed its own random number generator.

NetPhantom includes all the tools needed to create certificate requests, to sign these requests – for test purposes – and to create Identities.

The recommended steps for certificate creation are the following:

1. Generate the key pair and Certificate Request, using the Server Administration Program. For a detailed description of how to create the certificate request. For more information about how to create a self-signed certificate, see section *Creating a Self-Signed Certificate*.

2. Send the Certificate Request to a Certificate Authority (CA) to have your certificate certified.

3. Once the Certificate is returned from the CA, you need to create an identity that binds together the certificate and the private key. For the self-signed certificate, the identity will automatically be created. This is described in section *Creating an Identity*. When you have a certificate and an identity for the server, you can configure the server for SSL.

If a certificate is needed for testing purposes, one may be created without contacting the Certificate Authority. This is not recommended for a Production environment. A Certificate Request can be signed with your NetPhantom Server's own private key to create a Certificate.

The preferred Certificate Authorities such as:

Verisign Inc.	http://www.verisign.com
Thawte Certification	http://www.thawte.com
RSA Security	http://www.rsa.com

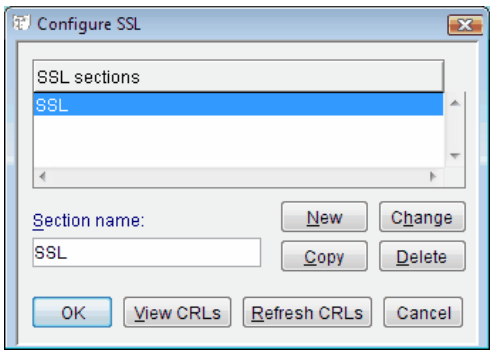
The server certificate allows you to begin creating other certificates from certificate requests by signing them.

You can also create client certificates. They can be distributed to the clients that will be using SSL to connect to your server. This is described in section *Configuration*.

Server Configuration

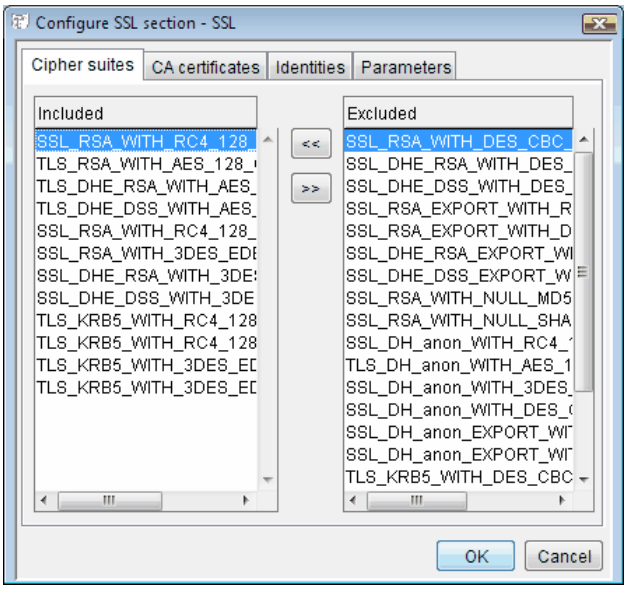
The server configuration is maintained in the `server.ini` file. In this file the SSL options may be changed and modified to your liking. All settings for SSL are saved in an SSL section of the `server.ini` file (default is `SSL`).

Most of these settings can be defined and/or changed through the Server Administration Program's graphical interface. To edit the SSL settings, start the Server Administration Program and select **Configure – SSL** from the **Server** menu (or **Server – Advanced – SSL Configure SSL** in the NetPhantom Editor). You will be presented with the **Configure SSL** panel. From this panel, you can create a new SSL section, change an existing one, copy an existing one to a new name, or delete a section.



The Configure SSL dialog is used to manage the various SSL sections in the server.ini file.

The default SSL section, called SSL, should be the only item in the list. To change it, select it in the list, and click the **Change** button. A new panel, called **Configure SSL section – *section name*** will be displayed.



The Configure SSL section – section name dialog allows you to set the specific settings for each separate SSL section.

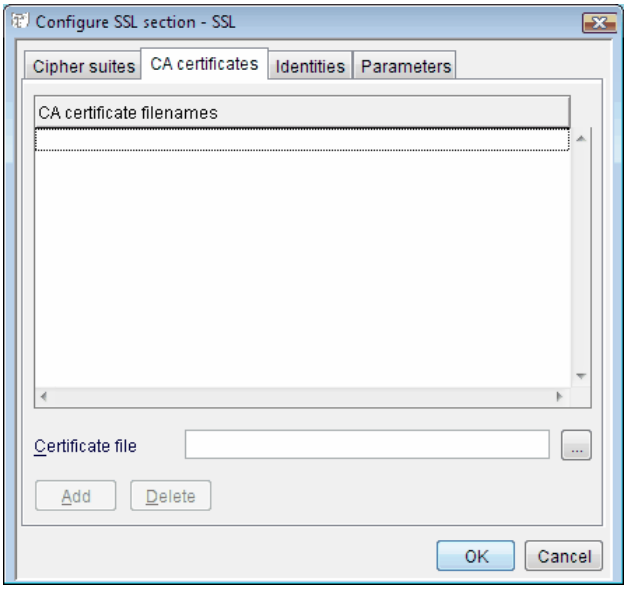
Cipher Suites

The **Configure SSL section** panel has four notebook tabs. From the first notebook page you select which cipher suites to accept. This list will be stored under the *cipherSuites* option in the `server.ini` file in order of selection from left to right, left being the preferred choice. If you are editing the `server.ini` file manually, keep in mind that the Cipher Suites must be comma-separated.

```
cipherSuites = SSL_RSA_WITH_RC4_128_MD5, SSL_RSA_WITHDES_CBC_SHA
```

CA Certificate Files

The second notebook page shows the CA certificate files you have incorporated in the server. The CA certificate files are used for the CA certification chain. To add new CA certificate files, specify the relative path or use the **Browse** button, then click the **Add** button. The **Delete** button removes CA certificates from the list.

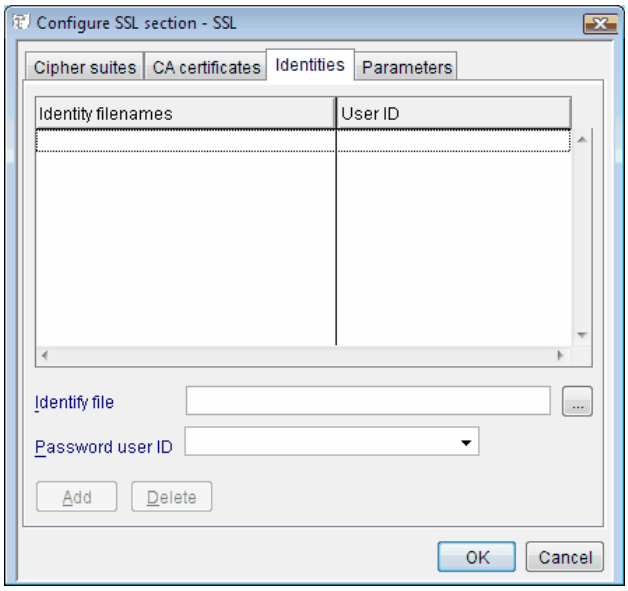


Use the CA certificates page to maintain the list of certificate files on the Server.

This list will be stored under the *caCertificates* option in the `server.ini` file. The list is comma separated. The files are DER encoded X.509 certificates.

Identities

The third notebook page holds a list of identity files. The Identity file is an object that binds together one or more certificates with a private key. The **Password user ID** is the ID for a *password user*, a user holding a password (or pass phrase) for the identity. The User ID is managed under the Server Administration Program, menu item **Server – Manage users** menu (or **Server – Advanced - Users…** in the NetPhantom Editor).



Identity files bind certificates to their private key.

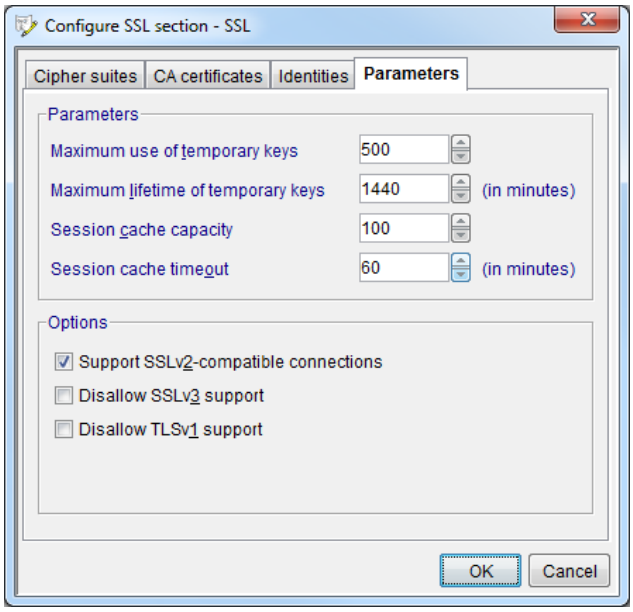
To add a new identity, enter the filename where the identity is stored, and add the **Password user ID** that has the password that is the same as the pass phrase for the identity's private key.

This list will be stored under the *identitiyFile* and *identityUser* options in the `server.ini` file.

```
identityFile = identity1.p12[,identity2.p12]
identityUser = user1[,user2]
```

Parameters

The **Parameters** page is used to define details concerning use of temporary keys, session cache parameters and the types of connections to allow.



*The **Parameters** notebook page contains parameter settings and options for the SSL configuration.*

Maximum Use of Temporary Keys

This setting is used to set maximum number of times that a temporary key will be reused before being destroyed. This is important for security (when the value is high), to prevent attacks on temporary keys, but has consequences for efficiency. The SSL specification recommends a value of 500. The minimum is 1. This setting will be stored under *maxTemporaryKeyUses* in the `server.ini` file.

```
maxTemporaryKeyUses = 500
```

Maximum Lifetime of Temporary Keys

This is the length of time from the creation of the key, after which it will be destroyed, regardless of how many times it has been used. This is important for security (when the value is high), to prevent attacks on temporary keys, but has consequences for efficiency. The SSL specification recommends a value of 24 hours (=1440 minutes). The minimum is 0. This setting will be stored under *maxTemporaryKeyLifetime* in the `server.ini` file.

```
maxTemporaryKeyLifetime = 1440
```

Session Cache Capacity

The session cache is used to improve SSL efficiency. New connections made between a familiar client and server can use cached security parameters to eliminate the costly asymmetric operations associated with the full SSL handshake.

This parameter sets the session cache capacity limit. It specifies the maximum number of session cache entries that will be supported. Limit this to prevent memory problems. The suggested value is 100; the minimum is 1.

This setting will be stored under *sessionCacheCapacity* in the `server.ini` file.

```
sessionCacheCapacity = 100
```

Session Cache Timeout

This parameter specifies the maximum duration in minutes that session cache entries will be maintained. You will want to expire old entries for security reasons. Existing entries older than the specified timeout are automatically purged. The suggested value is 1 hour (=60 minutes); the minimum is zero and implies no caching.

This setting will be stored under *sessionCacheTimeout* in the `server.ini` file.

```
sessionCacheTimeout = 60
```

Disallow TLSv1 Support

This flag instructs SSL to suppress TLSv1; i.e., only support TLS 1.1 or better connections.

This setting will be stored under *suppressTLSv1* in the `server.ini` file.

```
suppressTLSv1 = 0
```

ServerSocket Implementor

There is one setting that is not configurable from the Server Administration program interface. It is the *ServerSocketImplementor.* The only way to change it is to edit the `server.ini` file.

This option is used to enable or disable NetPhantom's Java JSSE module and replace it with another. The value is the name of the class file implementing the ServerSocket Interface.

```
serverSocketClass = se.entra.phantom.server.ssl.NetPhantomJSSEServerSocket
```

13.5 Creating an Identity

The purpose of an Identity file is to hold the certificates and private key. It is secured and locked with a Pass Phrase to prevent tampering. In a certificate-based public key infrastructure, your identity consists of your certificate (which contains your name and public key) and your private key. You give your certificate to other individuals to provide CA-authenticated information about yourself; then you use your private key to prove that you own the certificate.

An identity for the server must be created before you can configure the server for SSL. If you have created a self-signed certificate, the identity will have been created automatically. If not, select the menu item **Server – Certificate wizard…** (or **Server – Advanced – SSL – Certificate wizard…**).

The creation of the server identity certificate is wizard driven and consists of two steps:

1. Creating a certificate request.
2. Creating the server identity.

Select "Create certificate request" to launch the process. This will enable the creation of a certificate request which can then be sent to a certificate authority for creation and signing. The information needed is the distinguished name (identification of the certificate and its owner, geographically and logically) and the way the the certificate should be signed.

The output of this step is the certificate request (PKCS#10 format) itself and the corresponding key (PKCS#8 format).

When the certificate has been delivered, step #2 can be done; the actual creation of the server certificate. To do this, start the wizard and select "Create server identity".

The response can be in different forms depending on the certificate authority. The supported formats are: X509 certificate and PKCS#7 (a certificate chain). The supported encoding of these entities are DER (binary) and PEM (text). For the certificate to be complete, other certificates might be required. These are used to form what is called a "certificate chain". Whether of not this is necessary depends on the certificated authority and their procedures (the certificate response should be supplied with a specification of what intermediate certificates are needed). The wizard provides the means to build and automatically sort this chain.

For the creation of the keystore, the key generated (as described above) when creating the certificate request must be used. This key is fetched from the SSL directory on the server.

Next, the certificate chain is validated. This means a check to see if all certificates are available and that the root certificate is valued and issued by a trusted authority. Please note that the validation must not always be fully successful for the certificates to be used. This is the case when using self-signed certificates or when trial certificates issued by authorities are used. The pass phrase for the keystore used to store the identity must be supplied.

The wizard will allow an optional step to install the identity on the server you are currently running on. To this end, an existing SSL section must be selected. Once this is done, the certificate will be used in the section to secure the SSL port. If the port is not open before, the wizard will attempt to do so.

Once created the Identity file can be loaded into the Server to initiate the SSL connections.

13.6 Creating a Self-Signed Certificate

You can also create a self-signed certificate. This will not be as widely trusted as a certificate from a CA but might be acceptable within a single organization with a small network of clients.

A self-signed certificate should contain the same information about you and your organization as a certificate from a CA, so most of the fields that you must fill in are like those you fill in when you create a request for a certificate.

To create a self-signed certificate, start the Server Administration Program and select **Certificate wizard…** from the **Server** menu (or **Server – Advanced – SSL – Certificate wizard…**).

The creation of the self-signed certificate is wizard driven – select "Create self-signed server certificate" to launch the process.

13.7 Working with Authorized or Denied Client Certificates

When you use certificates to identify a client, whether these certificates are issued from the NetPhantom Server or from e.g. VeriSign, NetPhantom provides means to authorize individual certificates or all certificates issued to clients from an issuer (CA). If individual authorized certificates are used, more administration is required on the server side. If another party issues the certificates, this administration could become unmanageable.

NetPhantom therefore supports the revocation of client certificates. This means that all authorized certificates (individual or all from a CA) are accepted, unless the certificate has been revoked. Certificates can be revoked using Certificate Revocation Lists (CRLs) or on an individual basis.

Authorized Certificates

NetPhantom protects a resource by an Access Control that has a name. This is done using the Server Administration menu item **Server – Configure – Web server** (or **Server – Web Server…**)

If Access Control uses the option **Client certificates**, then all certificates will be accepted if they pass verification of the installed CAs in the SSL engine, except those in CRLs (see below). If you wish to use individual certificates, check the option **Certificate present on server** in the Access Control notebook page.

Each authorized certificate stored in DER files must be stored with unique filenames in a directory:

```
clientcerts/access_control_name
```

where the *access_control_name* is the name of the Access Control in *lower case*.

Revoked Individual Certificates

For each access control defined using the SSL option **Client certificates**, you must create a directory called:

```
revokedcerts/access_control_name
```

where the *access_control_name* is the name of the Access Control in *lower case*. All revoked individual certificates should be placed in this directory, stored in unique filenames. The files must be stored in DER encoded format.

> **Note:** The option **Client certificates** alone enables Revoked Individual Certificates, i.e. the option **Certificate present on server** is only used for Authorized Certificates.

Revoking Certificates using CRLs

NetPhantom reads all files in the directory:

```
crls
```

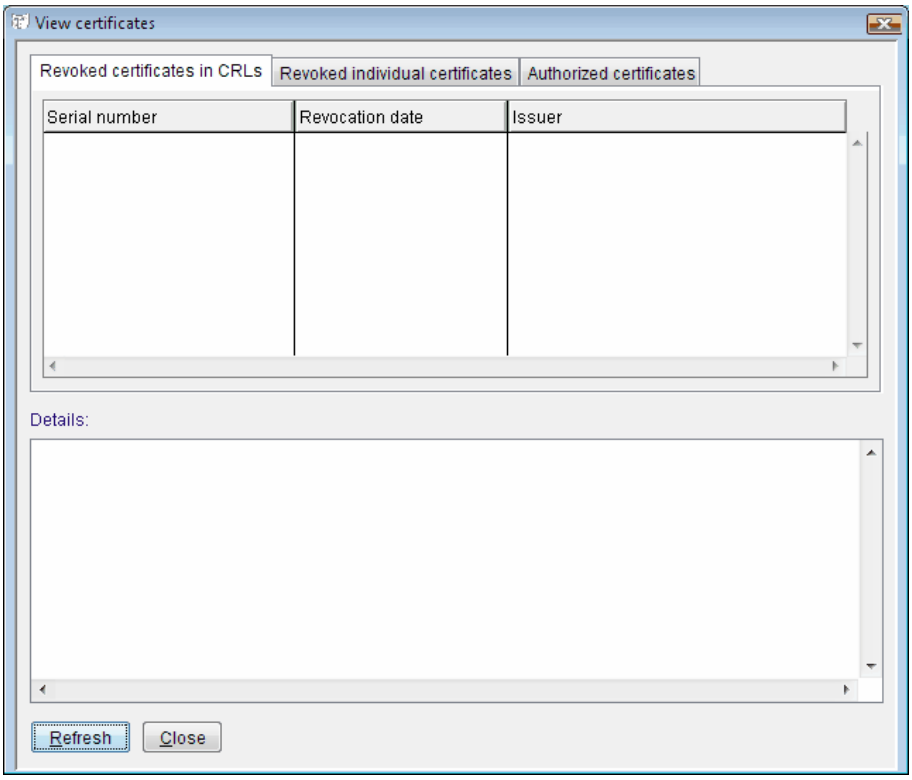
as soon as SSL is configured and used by the server. These files should be in *X.509 Certificate Revocation List* file format, in DER-encoded format.

These CRL files are normally downloaded from the CA that issued the certificates, then placed in the `crls` directory in unique filenames.

If several CRL files from the same CA exist, NetPhantom will merge these CRLs in order to create a single CRL for this CA.

13.8 Viewing and Refreshing Authorized or Revoked Certificates

The authorized or revoked certificates (individual or in CRLs) are normally loaded when the server starts. To view these certificates, select the menu item **Server – Configure – SSL** (or **Server – Advanced – SSL – Configure SSL…**) in the Server Administration Program followed by the push button **View CRLs**.



The panel displays the CRLs, revoked individual certificates as well as authorized certificates. To view details about the revoked or authorized certificate, select it in the list.

When new certificates are authorized or when a new CRL is downloaded from a CA, the server needs refreshing, i.e. causing a reload of CRLs, revoked individual certificates per Access Control as well as authorized certificates per Access Control. This is done either in the panel above using the **Refresh** button or in the **Configure SSL** panel with the **Refresh CRLs** button.

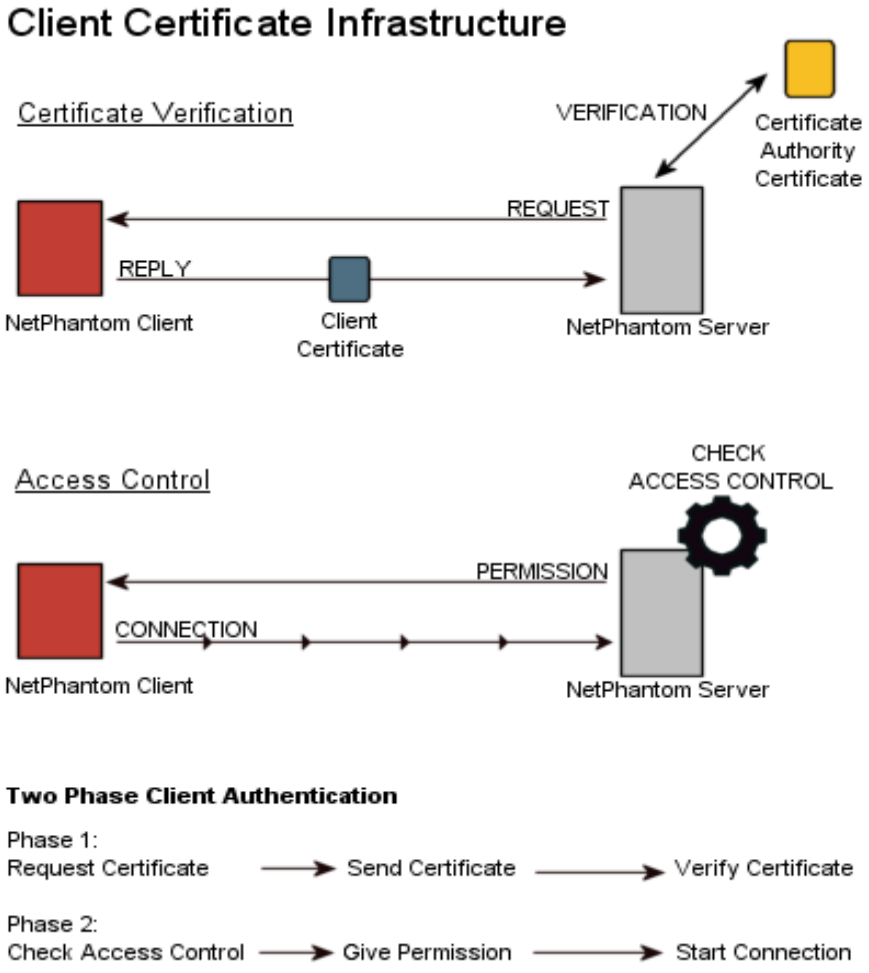
14 Client Certificates

14.1 Introduction

Client certificates, which are also known as Digital Ids or Digital Signatures, provide a stronger level of security and user authentication. The concept is based on public key infrastructure (PKI), which you can read about in various books.

By giving each user a client certificate, NetPhantom can customize users' security access. Client certificates are keys for different doors. These keys are used as a replacement or addition to the typical username and password authentication, creating a stronger and more effective security solution.

14.2 Technology Overview



14.3 Potential Uses

Client certificates have quite a few uses in a multi-user environment. Here are a few possibilities:

User Authentication

Client certificates allow NetPhantom to recognize each user connecting to provide authentication and close tracking.

Access Control

Through NetPhantom access control, restrictions can be added to application resources that can only be accessed with authorized client certificates.

Password Replacement

Client certificates can be used to replace username and password dialog box authentication. Certificates can be made to expire after a certain time.

14.4 Configuration

To use SSL client certificates in your NetPhantom infrastructure, extensive planning and configuration needs to be done. This guide will take you through the steps to implement this infrastructure with NetPhantom.

Creating Client Certificates

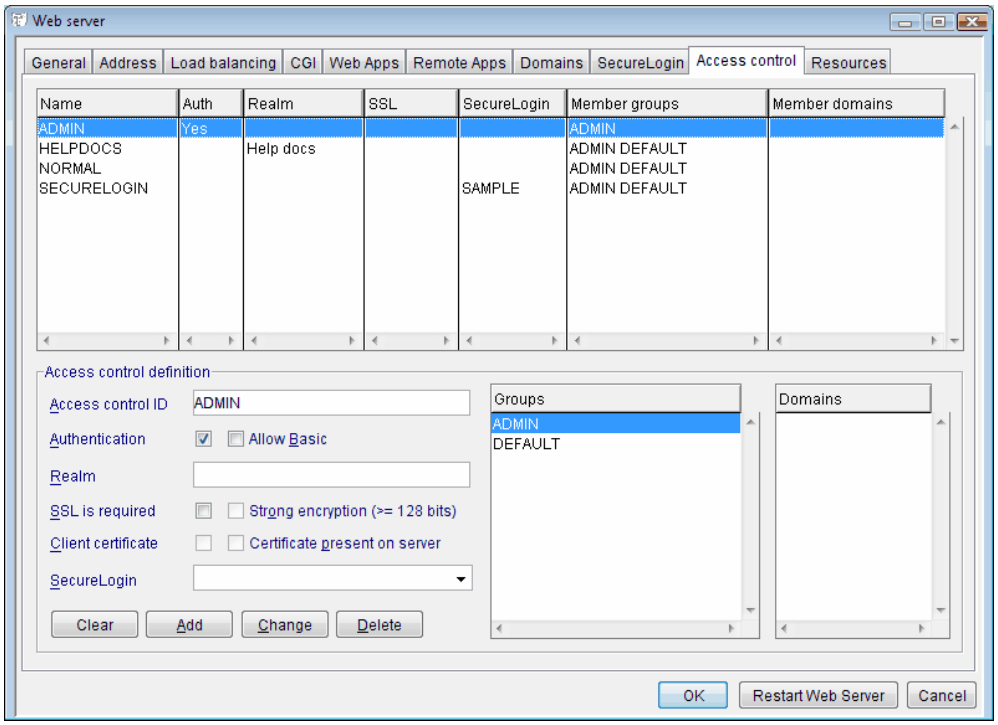
The first step is to produce the client certificates. They are created in the NetPhantom Server Administrator under the **Server – Certificate wizard…** menu (or **Server – Advanced – SSL – Certificate wizard…**).

The creation of the client certificates is wizard driven – select "Create client certificates" to launch the process.

The first step is to choose an identity to use as the base for signing the client certificates. Any PKCS#12 keystore containing a valid certificate chain can be used for this purpose. If an identity which is not already used as a server identity is chosen, the password for the keystore must be specified. The keystore used must be in the SSL directory (`ssl_data`) in the server root or a subdirectory.

The next step is to create a set of client certificates to be created. The client's name will be the basis for the files created containing the certificates. The set of client certs is created in batch mode.

The created client certificates will be prefixed with the string `CID_` (to distinguish them from the other types of certificates and keystores) and is available in the SSL directory in the server root. To finalize, the wizard will enable installation of the created server certificates in the server. This means that they will be assigned an access profile (or a set of profiles) and stored in the server corresponding to that assignment. The file structure used for this purpose can be found in the server root and called `clientcerts`.



1. Define the name for this access control.
2. Check the **SSL is required** checkbox.
3. Check the **Client Certificate** and **Certificate present on server** checkboxes.
4. Select the user groups to be included in this access control.

14.5 Client Certificates and Certificate Revocation

The Server can handle client certificates in three ways:

1. Every client certificate that should be allowed must be present on the server. This is very secure but requires a lot of administration if the number of certified clients becomes large, or if the clients can request certificates using e.g. Microsoft Certificate Services.
2. Individual revoked client certificates can be placed in a directory on the server. These directories are associated with the configured Access Control.
3. All client certificates that are valid and authenticated by a CA certificate present on the server (in the SSL configuration) are accepted, except those in a Certificate Revocation List (CRL). Most client certificate creation software provides means to download the latest CRL. The CRL contains the serial number of the client certificate (a unique number for the issuer of the certificate) and a revocation date.

Installing Client Certificates on the Server

Once access control has been defined, the client certificates will need to be moved to an appropriate directory.

The base directory is `clientcerts`, which should be in the server directory. Under this directory, create a directory name for each access control ID in lower case. For example, an access control called SECRET should have a directory called secret.

Serverdirectory`/clientcerts/secret/`

In each directory client certificates must be placed according to the access you would like to give to the client. As soon as the client certificate is placed in the directory, you will be joining that certificate to the corresponding access control.

The **Certificate present on server** checkbox sets the Server to check if the certificate sent by the client is in this directory. If the certificate is present, the Server will allow the client to connect to the resource. This option should be used to maintain extra security and the possibility of quickly revoking client certificates.

Installing Revoked Client Certificates on the Server

When Access Control is created with the **Client Authentication** option checked (and optionally **Certificate present on server**), a special directory is also used for revoked client certificates, i.e. users that are no longer allowed to use the resource associated with the access control.

The base directory is `revokedcerts`, which should be in the server directory. Under this directory, create a directory name for each access control ID in lower case. For example, an access control called SECRET should have a directory called secret.

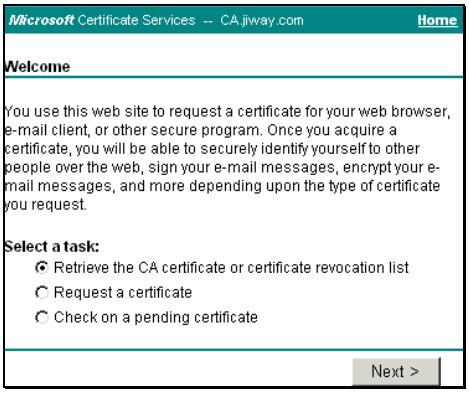
```
Serverdirectory/revokedcerts/secret/
```

In each directory client certificates must be placed according to the access you would like to give to the client. As soon as the client certificate is placed in the directory, you will revoke that certificate from the corresponding access control and its associated resources, even if the certificate is present in the base directory `clientcerts`.

When the server starts, these certificates are loaded and used for checking revoked certificates.

Installing Client Revocation List (CRL) Files

Most certificate service software can issue a CRL file, e.g. Microsoft Certificate Services provides this facility by means of downloading CRLs for a Certificate Issuer.



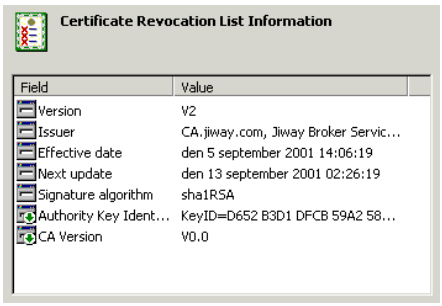
The initial screen in the browser using the web-based interface of Microsoft Certificate Services.

Next, download the certificate revocation list and store it in a file in the directory

```
Serverdirectory/crls/
```

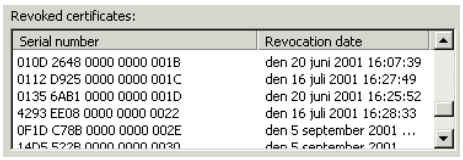
These files must be in DER encoded X.509 Certificate Revocation format for the Server to be able to read them. This directory can contain many CRL files, each for the same or different issuer (CA).

When the server starts, the CRLs present in this directory are loaded and used to check for revoked certificates.



When this file is opened under Windows 2000, the above screen displays information about the certificate issuer (CA).

The certificate revocation list contains the revoked certificate serial number along with the date when it was revoked.



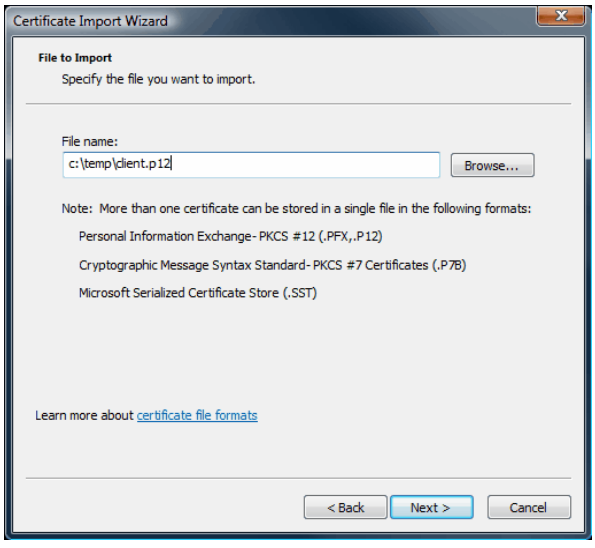
Some CRLs contain extensions to this data, e.g. the code explaining why the certificate was revoked.

Installing the Client Certificates on Client

Client certificates, depending on your infrastructure and installation, need to be installed in your web browser and the NetPhantom Client.

Browser

For browser installation, transfer the client certificates to the client computer. With Internet Explorer, open the **Tools - Internet Options** menu and choose the **Content** tab. The **Certificates** button brings you to the tool to import certificates.



The **Import** button starts with a Wizard that allows you to select the client certificate and import it. Once imported, it will be present in your certificate database and automatically sent on demand to the NetPhantom server.

Note: Remember the password you used to lock the certificate since the installation wizard will prompt you for it.

SSL Files Directory

The base directory for storing the SSL files is in the user's home directory in a directory called `.NetPhantom7`. The reason for this directory structure is to avoid a directory that could potentially be read-only for a user and to allow the same machine to provide support for multiple users.

Client Certificates

Client Certificates for a user should be stored in a directory called `clientcerts` under this SSL Files Directory.

Example:
`C:\Users\`*`NPUser`*`\.NetPhantom7\clientcerts` for Windows Vista and
`/home/`*`NPUser`*`/.NetPhantom7/clientcerts` for Linux (where *NPUser* is the user name specified when logging in to the Windows or Linux server).

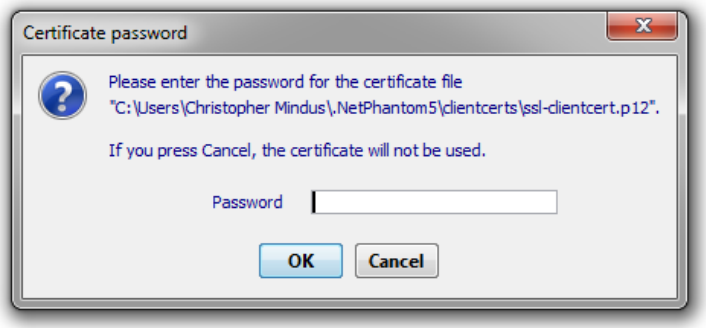
User-accepted CA Certificates

All CA certificates permanently accepted by a user are stored in a directory called `cacerts` under this SSL Files Directory as DER-encoded certificate files named `ua_ca_`*`nnn`*`.cer`.

Prompting for Client Certificate Passphrase

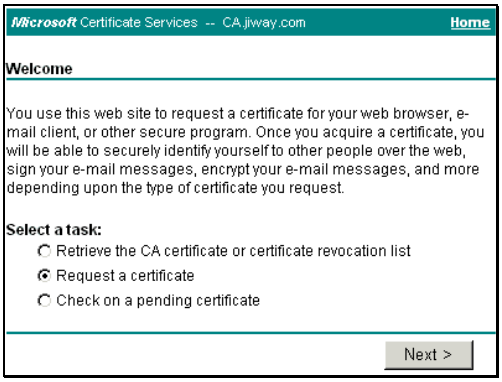
When the client certificate is required, it needs to have the *passphrase* that was specified when the certificate was created. If this password is set to `entra`, this will be assumed and the user will not be prompted for a password, *however this is not recommended because it is very unsecure!*

When the NetPhantom Client requests a certificate passphrase the following dialog box is displayed:

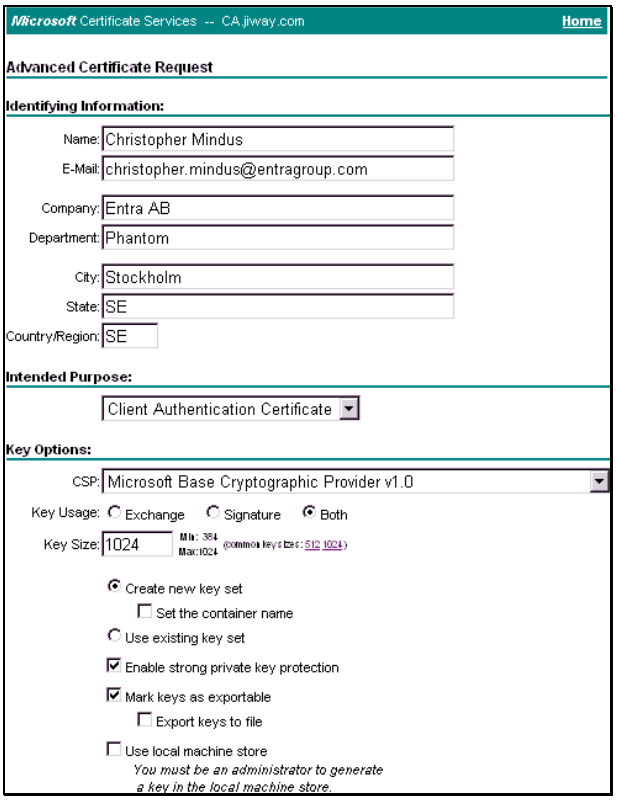


Example of Client Certificate Request and Installation for a Client

This example shows the steps to perform to request and install a client certificate using Microsoft Certificate Services. The certificate will be requested for use with Internet Explorer, for NetPhantom Client inside the browser as well as for a stand-alone NetPhantom Java Application using e.g. NetPhantom Starter SSL.



The certificate will now be requested from the issuer (the CA) using *Advanced request* and *Using a form* with the information that follows below:



Make sure to fill in your personal information correctly and take the following steps:

1. Make sure **Key Usage** is set to **Both**.

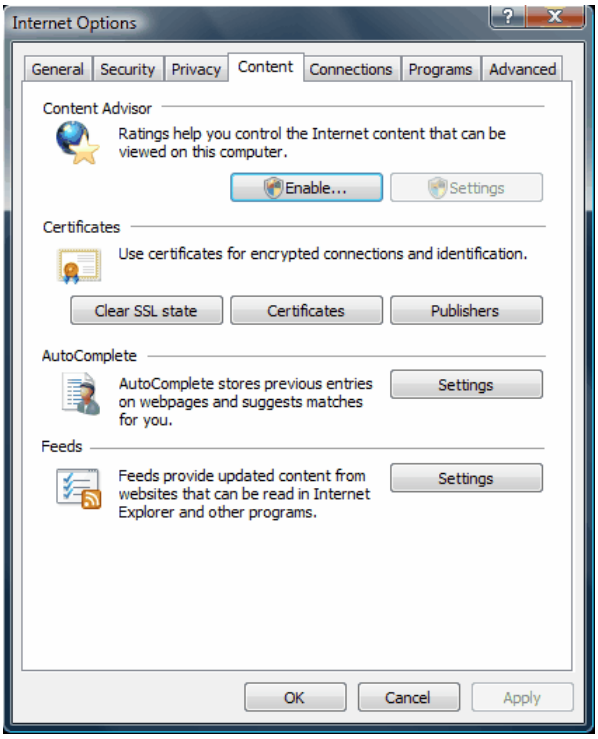
2. Change **Key Size** to `2048`. A smaller value is not appropriate for a 128/256-bit SSL encryption.

3. Check the option **Enable strong private key protection** to enable 128/256-bit SSL cipher suites for the encryption.

4. Check the option **Mark key as exportable** in order to be able to create a certificate file that can be used with the NetPhantom Client running as an Applet or as a Java Application (e.g. with NetPhantom Starter).

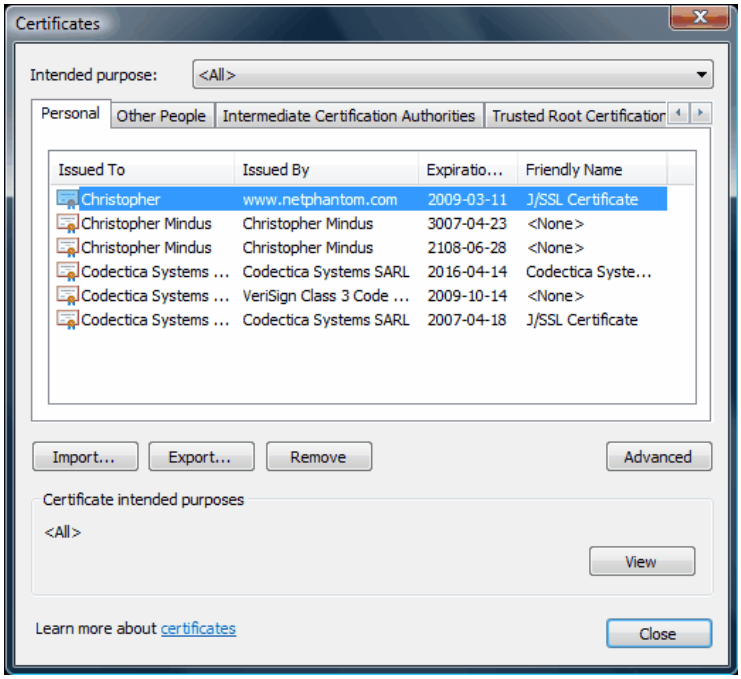
Then submit your request. Eventually the CA issues the certificate. At this point, return to the initial web page to install the certificate.

Once installed, select the menu item **Tools – Internet Options** in Microsoft Internet Explorer. Select the **Content** tab.



Select the push button **Certificates** to view the installed certificates.

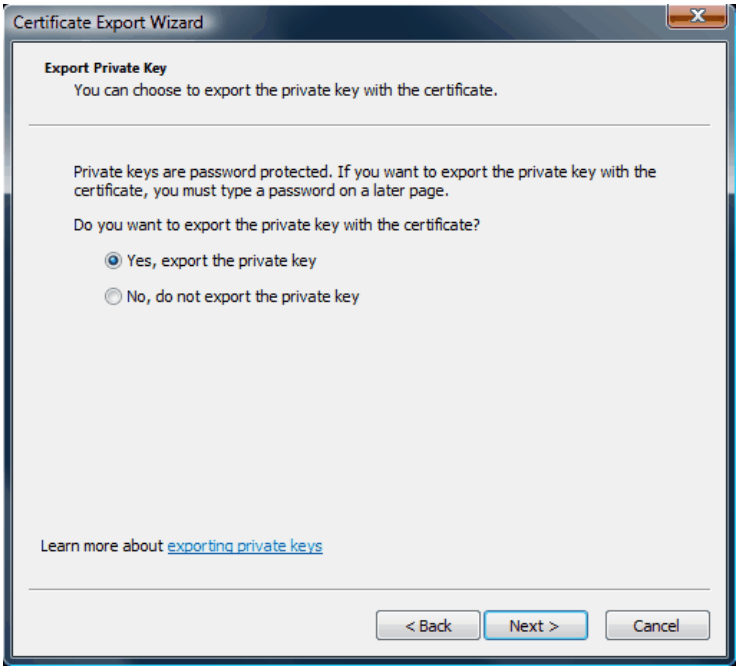
In the certificates dialog box below, select the certificate you just installed. Select the **Export** push button.



There can be several personal certificates in this dialog box.
Make sure to select the correct one.

The Certificate Export Wizard is then started.

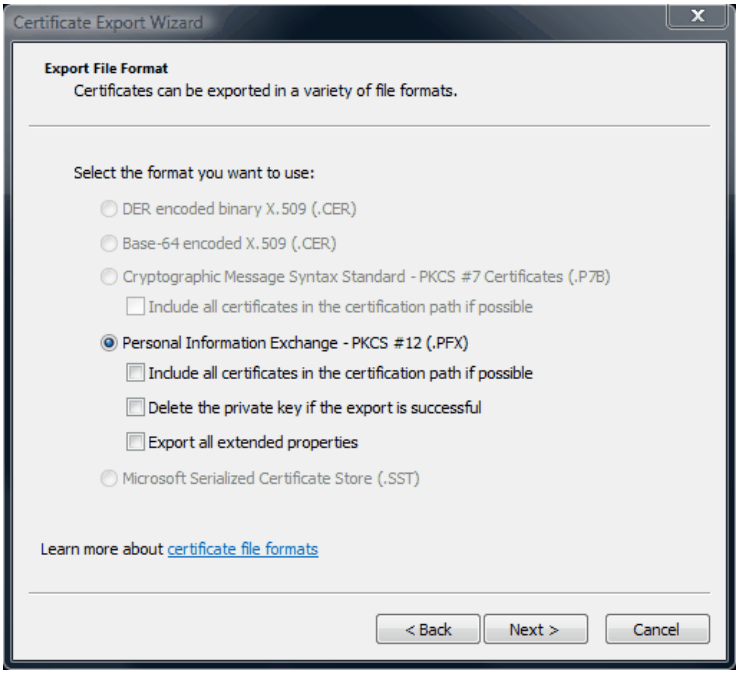
Note: Make sure that to remember the passphrase you used for the private key (using the password `entra`, you will not be prompted for the passphrase, but this is not very secure!)



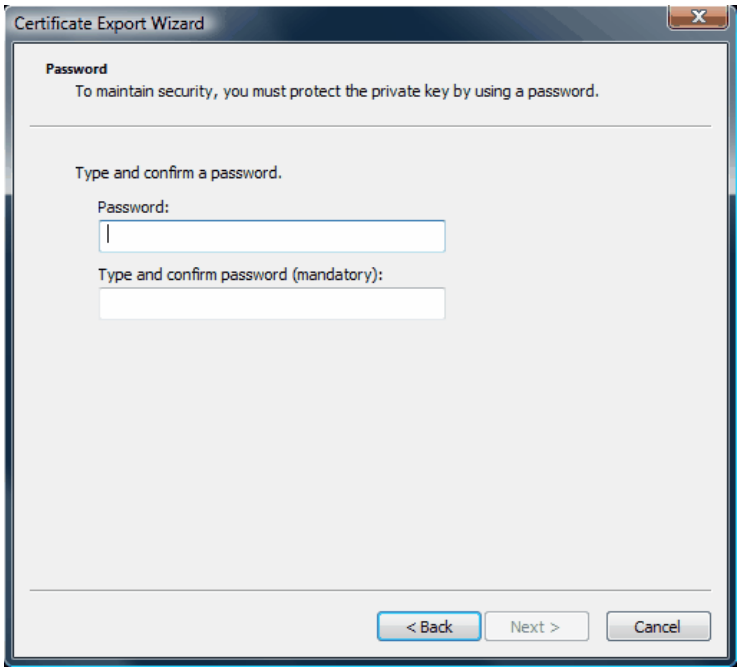
*Select option **Yes, export the private key**. Then press **Next**.*

Now the format of the certificate file must be specified. Make sure that the options below are checked:

1. **Include all certificates in the certification path if possible**,
2. **Enable strong protection (requires IE 6.0, or above)**.



Press the **Next** button. The panel where you enter the password for the private key is now displayed.



Now press the **Next** button and choose the file name, then complete the certificate export wizard process. Follow the steps below to enable NetPhantom Client to use the certificate.

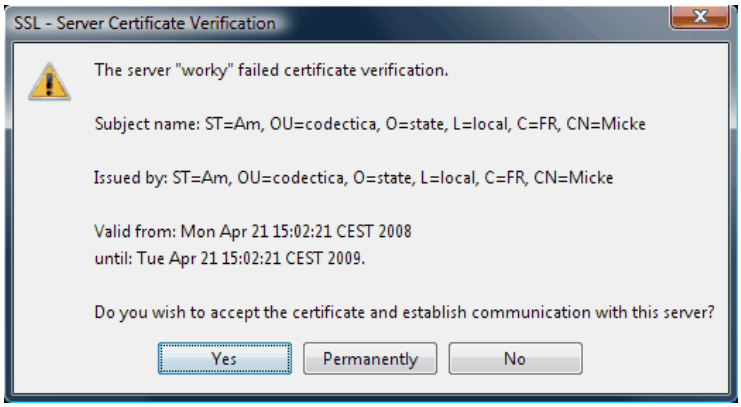
1. For drive-letter-based clients (Windows) make sure that the directory `<UserHomeDir>\.NetPhantom7\clientcerts` exists. If not, create it.

2. For non-drive-letter-based clients (e.g. Unix) the directory is is user home directory `.NetPhantom7/UserName/clientcerts` e.g. `homedir/.NetPhantom7/UserName/clientcerts`.

3. Copy the newly created certificate file into this `clientcerts` directory.

Running NetPhantom Client as Application with Client Certificate

When the NetPhantom Client running as a Java Application is started (e.g. via NetPhantom Starter), the following dialog box might be shown:



This dialog box is only shown when the server has a certificate that is not issued by publicly known CA's e.g. VeriSign, Thawte or RSA.

Select **Yes, for this session** or **Yes, permanently** if you wish to accept this server certificate. The accepted certificates are stored in files called `Ua_ca_nnn.cer` in a subdirectory called cacerts under the SSL files directory for the NetPhantom Client running as an Application. No such file is created for the Applet version of the client.

14.6 Restricting an Application Resource

To restrict an application resource, use the NetPhantom Administration console and select the **Configure – Web server** menu item. On the **Resources** page, configure a resource as described in section *14.5 Configuring the Web Server – Resources* choosing the access control you created for the certificates.

15 NetPhantom Starter

15.1 Overview

The purpose of NetPhantom Starter is to make it possible to run NetPhantom as a Java application without the need to go through a normal installation procedure, or the need to upgrade to future versions. The NetPhantom Starter automatically upgrades the local version once a newer version is available on the server. Running NetPhantom as an application frees the user from the constraints imposed by Web browsers. Among other things, the client will gain better integration into the user's desktop and increased start up speed, since it is no longer downloaded at every startup but only when a new version is released and available on the server.

NetPhantom Starter includes many features such as redirection, automatic updating of the client files, support for backup servers and an application chooser.

With NetPhantom Starter, the NetPhantom Client will always run as a Java application. The initial installation is accomplished through an Internet browser applet. This is the easiest way for NetPhantom administrators to install NetPhantom Starter on the user machines. It is so easy that most users will be able to perform the installation themselves. After NetPhantom Starter is installed and run for the first time, the user can run the NetPhantom Client Java application as easily as any other application and will always have the latest released program version.

15.2 NetPhantom Starter

NetPhantom Starter is a Java program that starts a locally installed or previously downloaded NetPhantom Client on the user’s machine.

Before the NetPhantom Client is started, Starter performs the following steps:

1. It first checks a File or a Web Server for the Package Definition File. This Package definition file is used to specify which files must be installed, updated or deleted on the local machine or to redirect this request to a Package Definition file located somewhere else.

2. If a defined file is present on the local machine, it is checked against the server for the correct version/release by the file size and file timestamp.

3. If these checks do not match, a new version of the file is automatically downloaded.

Once the local machine’s NetPhantom Client files are synchronized to the local machine, the client invokes the specified start class.

A separate program that manages the Package Definition File is distributed with this package. The administrator should use this program to make changes in the Package Definition File.

Web Server Authentication

The NetPhantom Starter (with or without SSL) supports Web Server Authentication (or HTTP Authentication). This means that the user is prompted for a user ID and a password.

NetPhantom Starter implements both the “Basic” and “Digest” authentication schemes as defined by W3C (but does not support the extended authentication type “Digest qop=auth-int”).

Web Server Redirection

The current version of NetPhantom Starter does not support HTTP redirections. *This means that the resources required by the NetPhantom Starter must not be load balanced or otherwise redirected*.

15.3 NetPhantom Starter with SSL

NetPhantom Starter is also available with 128/256-bit SSL support.

The purpose of this version is to provide a secure connection to a web server during the time of a potential file download and update, and then later to use the NetPhantom Client with SSL to communicate securely over Internet to the NetPhantom Server.

As for the NetPhantom Client, there is support for *Client Certificates*. See the section on *Client Certificates above* for more information.

Accepted Server Certificates

When the NetPhantom Starter establishes communication with a server whose certificate doesn't verify against the accepted server certificates, the following dialog box is displayed:

If the user accepts the server certificate for this session, this information is passed on to the NetPhantom Client to avoid further user interaction.

Client Certificate Password

When a client certificate is used (and the password is not "`entra`"), the user must supply the password for the certificate. The user must remember the password for each individual certificate. However, this information is passed on to the NetPhantom Client (or an upgraded NetPhantom Starter) to avoid prompting for the password again.

15.4 Installation of the NetPhantom Starter on the Client

All users must perform an initial installation of the NetPhantom Starter. When new file versions are available, these will be automatically downloaded and installed on the client machine.

NetPhantom Starter with SSL is available in a separate installation program with the same installation procedure as for the non-SSL version.

The first time the user starts NetPhantom Starter, she will be asked to enter the URL address to the server where the Package Definition File is located. The Package Definition file is called `client.pkg`.

If backup servers are used, several server names can be specified, as a comma separated list. For example, if the servers are named *myserver1* and *myserver2*, the URL address to the server can be specified as:

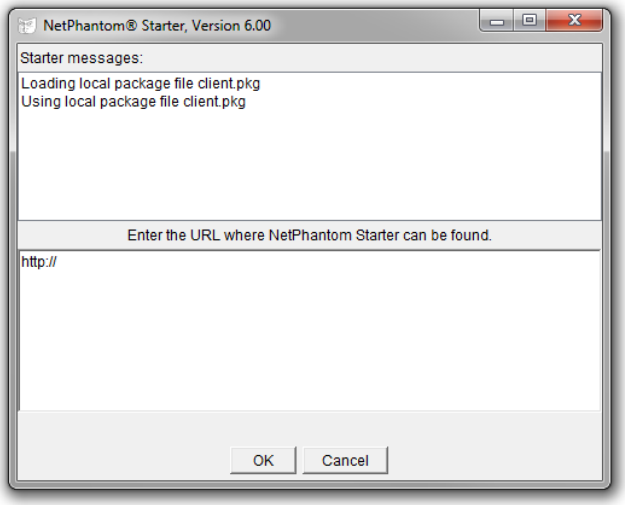
```
http://myserver1:8080,http://myserver2:1789
```

The full Server URL specification is:

```
[proxy:phost:pport:]http[s]://shost:sport,…
```

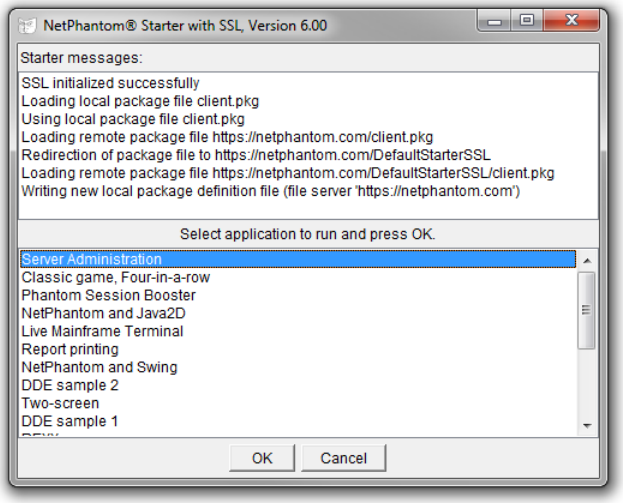
where the *proxy* settings are optional.

This initial Package Definition file will either redirect to another Package Definition file, if it contains a redirect command or incomplete information, or it will load the application specified in the Definition file.



The defined application will then be transferred to the local machine. If the Package Definition File `client.pkg` lacks information in the `programClassName` field or the `classpath` field, the Application Chooser Package Definition File will be used. It is called `applications.pkg`. Once loaded, you will then be presented with a list of possible applications. These choices are defined in `applications.pkg`.

The format of these Package Definition files is explained in *Package Definition File.*



The next time the user starts NetPhantom Starter the program will read the server address from the locally stored `client.pkg` file and then go through the same steps.

Changing the Behavior of NetPhantom Starter on the Client

It is possible to change the default behavior of NetPhantom Starter on the Client by modifying the line beginning with the text "Cmd=", in the `NetPhantom Starter [SSL] 6.ini` file under section `[Application]`. This line is normally not present because no parameters are passed.

Parameters for "Cmd=" can be added as:

```
[baseURL] [-C startclass [-CP classpath] [-A arg1 [argN]]]
```

`[baseURL]`	The default *target server* to use, normally manually entered.
`-C startclass`	To specify another class than the default start class for the client. It can also be used to specify additional start arguments to the start class, when combined with the –A argument.
`-CP classpath`	Specifies additional classpath that will be used by the start class.
`[-A arg1 [arg2 [argN]]]`	Specifies one or more command line arguments to be passed on to the start class to the *NetPhantom Client*.

For example, to have a Starter that automatically starts the Server Administration application, without first displaying the application list, the line in the `NetPhantom Starter [SSL] 7.ini` file could be changed according to the example below.

```
[Java Runtime Environment]
VM Provider=any
Maximum Version=15
Main Class=NetPhantomStarterSSL
Minimum Version=1.8
[Product Information]
Upgrade Code={90E0E55E-6CE8-482C-86E0-3877811CF558}
Product Name=NetPhantom Starter SSL 7 (64-bit)
Product Code={4FFC661E-D305-4551-A787-621C0F02B605}
[Application]
Override WorkingDir=yes
Application Type=gui
Failure Check=yes
[Class Path]
Class Path=C:\Users\chris\Documents\NetPhantom Starter SSL 7 (64-bit)\;
```

15.5 Installing Multiple NetPhantom Starters on the Client

It is possible to install multiple NetPhantom Starters on the same client. There are, however, a few things that must be kept in mind when doing this.

After the first starter has been installed, make sure to rename the shortcut and the menu item, before installing the second Starter. If this is not done, the first Starter's menu item and desktop shortcut will be overwritten, even if the Starters are installed in separate directories.

15.6 Installation of the NetPhantom Starter on the Server

Perform the following steps to create a web server installation of the NetPhantom Starter:

1. Create a directory where the NetPhantom Starter can be installed (e.g. make a copy of the `htdocs/NetPhantomStarterSetup` directory).

2. Create a second directory that should be specified as the web server directory when the users first start their NetPhantom Starter. This directory should contain all files (including a complete directory structure) required on the user's machines.

3. In the second directory, run the *Package File Maker* program as described below to create the `client.pkg` file. Make sure all referenced files are in the directory.

15.7 Package Definition File

This file defines which files should reside on the local machine or a redirect to another Package Definition File. The name of the Initial Package Definition File is `client.pkg`.

A separate Java program is constructed to help the NetPhantom Server Administrator in the making of the Package Definition File is distributed together with the NetPhantom Server package. See *Switches for PackageFileMaker* below.

Here are some examples for the Package Definition File for both cases: loading and redirecting.

For loading files:

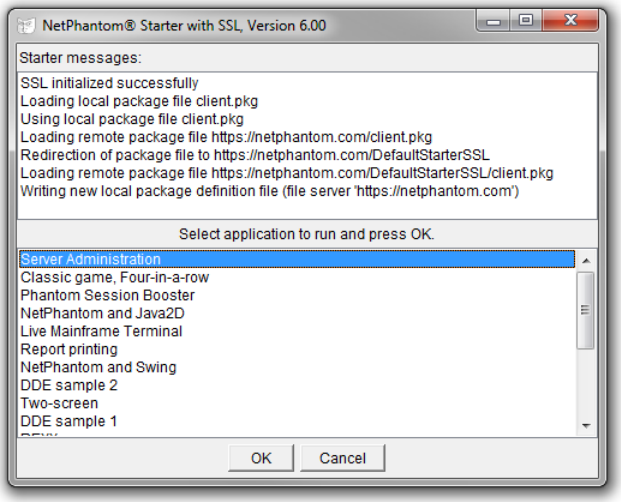
```
#This is the NetPhantom Starter Package Definition file.
#Tue Oct 15 10:22:30 CEST 2018
fileServer=http\://ghost
file.count=9
file.9=1600,2389542803,se/entra/phantom/auth/b.class
file.8=7613,1799155035,se/entra/phantom/auth/Auth.class
file.7=31717,3574699597,NetPhantomStarter.class
file.6=2786,3589007474,se/entra/phantom/auth/a.class
file.5=98304,2497140241,npcdde.dll
file.4=231,3470080688,se/entra/phantom/auth/AuthCB.class
file.3=1093,2622527136,NetPhantomStarter$NLabel.class
file.2=119724,3834027497,NetPhantomClientImages.jar
file.1=434106,2764433140,NetPhantomClient.jar
classpath=NetPhantomClient.jar;.
pgmarg.3=app:RESURSBOKNING
pgmarg.2=port:8080
pgmarg.1=host:www.netphantom.com
```

For redirecting files:

```
# This file redirects the NetPhantom Starter [SSL] to the correct
location.
redirection=/DefaultStarter
redirectionSSL=/DefaultStarterSSL
```

15.8 Application Chooser Definition File

The name of the Application Chooser Definition File is `applications.pkg`. It is used to define a list of applications that can be loaded by the client. Once this Definition File loaded by the client, NetPhantom Starter will present the user with a list of applications.



Editing the Application Chooser Definition File

In a standard NetPhantom installation, the `applications.pkg` file is located in the `/htdocs/DefaultStarter` directory. It can be edited with a standard text editor.

An example of the Application Chooser Definition File for two applications:

```
apps=SERVERADMIN,S_EXCELDEM

programClassName=se.entra.phantom.client.Phantom
classPath=NetPhantomClient.jar;.

SERVERADMIN.text=Server administration
SERVERADMIN.pgmarg.1=port:@**SERVERPORT@
SERVERADMIN.pgmarg.2=ssl:0
SERVERADMIN.pgmarg.3=hostid:.
SERVERADMIN.pgmarg.4=app:SERVERADMIN
SERVERADMIN.pgmarg.5=host:@**SERVERNAME@
SERVERADMIN.pgmarg.6=resurl:http://@**SERVERNAME@:@**SERVERPORT@

S_EXCELDEM.text=DDE sample 2
S_EXCELDEM.pgmarg.1=port:@**SERVERPORT@
S_EXCELDEM.pgmarg.2=ssl:0
S_EXCELDEM.pgmarg.3=hostid:C
S_EXCELDEM.pgmarg.4=app:SAMPLES_EXCELDEM
S_EXCELDEM.pgmarg.5=host:@**SERVERNAME@
S_EXCELDEM.pgmarg.6=resurl:http://@**SERVERNAME@:@**SERVERPORT@/sample
```

Each application definition entry begins with the application name. For reasons of simplicity we recommend defining the same application name as you defined for the application in the server.ini file.

`Apps`	A comma (,) separated list of applications to be displayed in the Application Chooser window. Applications that are defined but not listed in the `apps` list will *not* be shown in the chooser.
`programClassName`	The name of the implementing class. This entry should not be edited.

classPath	The classpath for the program. This entry should not be edited.
myApp.text=	The text that will be displayed to the user when the Application Chooser window is displayed. It should be user friendly as with SERVERADMIN.text=Server administration
myApp.pgmarg.1= port@**SERVERPORT@	The port number on the server. This entry should not be edited.
myApp.pgmarg.2=ssl:0	This argument is a flag to indicate whether or not the application uses SSL 0=False, 1=True
myApp.pgmarg.3=hostid: A	The host ID must correspond to a host session defined in on the Host page in the Server Administration program.
myApp.pgmarg.4=app: myApp	The app argument is the name of the application as it is specified on the Applications page in the Server Administration program.
myApp.pgmarg.5=host: @**SERVERNAME@	This is the server's name or address. This entry should not be edited.
myApp.pgmarg.6=resurl: http://@**SERVERNAME@: @**SERVERPORT@/myApp	Here you enter the path to where the application is located. The //@**SERVERNAME@:@**SERVERPORT@ part of the path should not be edited.

The @**SERVERNAME@ and the @**SERVERPORT@ tags ensure that the correct servername and port will be used, even when the client is redirected to a backup server or load balancing slave server.

15.9 Switches for PackageFileMaker

PackageFileMake is a separate Java program constructed to help the NetPhantom Server Administrator in the making of the Package Definition File; it is distributed together with the NetPhantom Server package.

This program is launched in the following way:

```
java PackageFileMaker
```

and takes the following arguments:

```
-f    NetPhantomStarter.class [all-files-that-should-be-included]
      (Everything on one line)
-f    @files.lst (this file can be used to specify all the client files)
      (Everything on one line)
-c    Class-to-invoke-by-NetPhantomStarter
-a    Starting-class-arguments [next-starting-class-argument] […]
-p    ClassPath for the client/local machine.
```

Note that the "-a" switch must come last in the order of used switches.

Client Files (-f)

The –f switch should be followed by the files that the client will have stored locally. Files should be separated with spaces. PackageFileMaker must be run from its current directory. This is usually the directory in which the NetPhantomServer.jar file is located, since the PackageFileMaker program will be included in the NetPhantomServer.jar file.

The client files that you include must be specified with their path relative to this current directory, as in the following example:

```
-f clientfiles/NetPhantomClient.jar
```

and *not* in this fashion:

UNIX:

```
-f /home/…/clientfiles/NetPhantomClient.jar
```

Windows:

```
-f C:\clientfiles\NetPhantomClient.jar
```

The file list should include all the files listed in the `client.pkg` file plus any additional files you want to be downloaded to the client from the server. Note that the `NetPhantomStarter*.class` files should always be included in the Package Definition File. If these files are not included, the client will delete them and be unable to perform a restart next time it tries to launch the application. The PackageFileMaker program will give you an error message and exit if these files are omitted. They are included to make it possible to download new features to these classes in the future. These classes should be placed in the installation "root" directory.

You can specify any number of files that you want to include in the package definition file, but your command line will probably not have enough space to write down all these files. Instead, you have the option of putting all the file references in a separate list file. That list file then must be passed to the PackageFileMaker in the following way: (the list filename here is `clientfiles.lst`)

```
-f @clientfiles.lst
```

The "@" character signals that this file contains all files that should be included in the package definition file.

The List File

Specify each file on a separate line in the list file. Remember that file paths must be relative. See example below:

```
NetPhantomStarter.class
NetPhantomStarter$NLabel.class
NetPhantomClient.jar
ProgramFiles/clientsettings.ini
ProgramFiles/clientmessage.txt
```

Class (-c)

The "–c" switch specifies the class that should be invoked by NetPhantom Starter after it has checked the file versions. Most often it will be used in the following way:

```
-c se.entra.phantom.client.Phantom
```

This will start the NetPhantom client application.

> **Note:** It is very important that the class you specify is included in the clients' CLASSPATH. Otherwise, the client application will not start. You may do this with the "-p" option.

Arguments (-a)

The "-a" switch specifies the arguments that should be passed to the class that is invoked by NetPhantom Starter. This switch must be the last one specified. In the normal case, these

arguments are the ones that are passed to the se.entra.phantom.client.Phantom class in order to make the application start correctly.

Classpath (-p)

The “-p” switch specifies the classpath that the client should use to load the classes. Each file or directory must be separated with a semi-colon and directories should only use URL format (with forward-slashes)

This program produces the Package Definition File with the filename `client.pkg`. This file should *not* contain a reference to itself.

Common Server Administration Functions

In the text below, running multiple NetPhantom Servers on the same machine requires the usage of different server configuration files (`server.ini`). Since they reside in the same directory, the server.ini files must each have a unique name that can be specified at server startup by appending the name to the initiation command. There must also be different log and trace files for each server, which is specified in each `server.ini`.

Example: `java se.entra.phantom.server.Start server2.ini`

15.10 Backup Server(s)

Note: Most web servers provide a redirection service, which overrides the restriction of "another server". Thus, in the case of a browser solution, the web server will redirect the request to a NetPhantom Server by redirecting the browser to another web server that contains the backup NetPhantom Server. Therefore, this section of the document only applies to a Java application solution.

As the NetPhantom Server can drive many concurrent clients, there must be a way to change the server software or to shut it down for various reasons (there may of course be many other situations in which this is also required).

To do this, another NetPhantom Server is started on the server, listening to client connections at a specific TCP/IP port number. This can be done several times to have several backup servers. The most common case is to have a single backup server running. This backup server is not a physical unit, but rather another process running on "a" server (can be the same or another physical server for a Java application client solution, *must* be the same for a client browser solution).

The backup server uses the same setting information as the primary server (if located on the same physical server), except that it is a backup server of e.g. number 1, so only the TCP/IP port number changes. The backup server(s) normally resides on the same physical machine as the primary server.

The client software will decide which server to use based on a list of servers specified in the server list (in host and port parameters for the client). The list will be used to determine the server of choice, which will then be used for the actual host session. The client will contact the server, and if it is connectable, will negotiate a host connection with the server. If this fails because of connectivity or because the server denies the connection, the client will try the next server in the list until the end of the list is reached (at this point, the client will display an error message). Keep in mind that if the server or the host denies a user, the client will not continue to search for the list of available servers.

15.11 Server Events

The NetPhantom Server produces many events, such as client connection and logon, warnings and error events. As these events are multiplied by the amount of client connections, the server has a way of filtering such events to ease the administrator's tasks. This is important because there can be so many events induced from a single server. If there are many servers, the event count could become unmanageable.

All events are always stored on the local server in a text file. This file is specified using the server administration program or can be edited manually in the `server.ini` configuration file. A single Java class that can be redirected using a "user exit" processes all events. See *Server Event User Exit* for more information. Only filtered events are passed on to the Server Event User Exit, see *Event Filtering* below.

15.12 Event Filtering

The event filtering mechanism is divided into client sessions, included, and excluded events. The following command is used for event filtering:

```
EventFilter client [include=range] [exclude=range]
```

where *client* is a list of client session IDs or `ALL`. To see the list of current clients, use the `CLIENTS` command (see *Remote Server Administration*). The *range* specifies the events to include or exclude as:

```
 ---+-- EventID ----------+---
    |                     |
    +-- EventID-EventID --+
```

The `EventID` is the text ID used in the NLS file below. It can be terminated with "*" which means all text IDs that begin with the specified text. Several event IDs can be specified, using a comma "," as separator. The complete list of event IDs will be specified at a later stage.

The command is cumulative, e.g. if the first command is

```
EventFilter ALL include=ALL
```

followed by

```
EventFilter 3197 exclude=ALL
```

will cause all events to be recorded at the server administrator console, except for client session `3197`.

In the `server.ini` file, a default server startup filtering is setup in the section `[EventFilter]`. All lines up to the next section that start with the `EventFilter` command (as above) are treated as event filter commands.

15.13 Tracing

Tracing can be turned on or off for a particular client session or for all client sessions. There are always four levels of trace:

- no trace
- binary trace
- verbose trace
- binary and verbose trace.

The following traces exist:

- Telnet datastream
- Host datastream
- Client datastream
- The REXXMigration/NetRexxMigration API

The trace options for server administration can be interpreted by the user exit to handle more advanced tracing. The following administrator command for tracing exist:

```
Trace client subsystem level
```

where:

client
client connection number | `ALL` | `NEW`

subsystem
`Telnet | Host | Client | API`

and

level is:

`Off | Binary | Verbose | BinaryVerbose`

This command can be issued from the server process, by the remote command utility or from the Server Administration program.

Example of Binary Trace Output

```
21 sep 2021  11:18:40.610  0000000000000001  Telnet      0000000000000006
READ TELNET
 C0 F0 3C E4 50 40 29 02 42 F2 C0 F8 3C E5 60 40     ..<.P@).B...<.`@
 29 02 42 F4 C0 F0 C3 96 94 94 81 95 84 40 7E 7E     ).B..........@~~
 7E 6E 29 03 41 F4 42 F5 C0 40 3C E5 7E 00 3C E6     ~n).A.B..@<.~.<.
 F0 40 29 02 42 F1 C0 F0 C6 F1 7E C8 85 93 97 3C     .@).B.....~....<
 E6 7D 40 C6 F7 7E C2 92 A6 84 3C E7 C9 40 C6 F8     .}@..~....<..@..
 7E C6 A6 84 3C E7 D4 40 C6 F1 F0 7E D3 85 86 A3     ~...<..@...~....
 3C E7 61 40 C6 F1 F1 7E D9 89 87 88 A3 3C E7 F4     <.a@...~.....<..
 40 29 01 C0 7C E3 C3 E4 F0 F0 F4 F1 40 29 02 42     @)..|.......@).B
 F1 C0 F0 40 40 11 E5 6E 13 FF EF .. .. .. .. ..     ...@@..n........

21 sep 2021  11:18:40.610  0000000000000001  Telnet      0000000000000006
READ DATASTREAM
 0D C3 29 02 42 F1 C0 F0 E3 C3 E4 F0 F0 F4 F1 40     .C…1.0TCU0041
 29 02 42 F1 C0 F0 3C 40 5D 40 D5 E5 89 61 E3 D7     ...1.0. ) Nvi/TP
 E7 40 D4 85 95 A4 40 C6 96 99 40 29 02 42 F1 C0     X Menu For ...1.
 F0 C5 E7 E3 C3 D4 40 40 40 29 02 42 F4 C0 F0 3C     0EXTCM   ...4.0.
 C1 50 40 29 02 42 F7 C0 7C 3C C2 60 40 29 02 42     A& …7…B- …
 F4 C0 F0 D1 A4 94 97 40 4B 40 7A 29 02 42 F5 C0     4.0Jump . :…5.
 F8 D7 C6 F2 F4 40 29 02 42 F4 C0 F0 C3 94 84 92     8PF24 …4.0Cmdk
 85 A8 40 4B 40 7° 29 02 42 F5 C0 F8 C5 D5 E3 C5     ey . :...5.8ENTE
 D9 40 40 40 29 02 42 F4 C0 F0 C3 94 84 83 88 99     R   …4.0Cmdchr
 40 4B 40 7A 29 02 42 F5 C0 F8 5A 3C C3 D7 40 29      . :…5.8..CP .
 02 42 F4 C0 F0 E2 A8 A2 A3 85 94 40 4B 40 7° 29     ..4.0System . :.
 02 42 F5 C0 F8 D5 D6 E5 E3 D7 E7 3C C3 F0 40 29     ..5.8NOVTPX.C0 .
 02 42 F4 C0 F0 D4 85 95 A4 40 4B 40 7A 29 02 42     ..4.0Menu . :…
```

Example of Verbose Trace Output

```
21 sep 2021  11:18:40.610  0000000000000001  Telnet      0000000000000006
DATASTREAM READ: COMMAND: ERASE WRITE ALTERNATE

21 sep 2021  11:18:40.620  0000000000000001  Telnet      0000000000000006
DATASTREAM: ORDER: SFE [0: 42/f1 c0/f0]

21 sep 2021  11:18:40.620  0000000000000001  Telnet      0000000000000006
DATASTREAM: CHARACTER DATA ['TCU0041 ']

21 sep 2021  11:18:40.620  0000000000000001  Telnet      0000000000000006
DATASTREAM: ORDER: SFE [9: 42/f1 c0/f0]

21 sep 2021  11:18:40.620  0000000000000001  Telnet      0000000000000006
DATASTREAM: ORDER: RA [10->29: ' ']

21 sep 2021  11:18:40.620  0000000000000001  Telnet      0000000000000006
DATASTREAM: CHARACTER DATA ['Nvi/TPX Menu For ']

21 sep 2021  11:18:40.620  0000000000000001  Telnet      0000000000000006
DATASTREAM: ORDER: SFE [46: 42/f1 c0/f0]

21 sep 2021  11:18:40.620  0000000000000001  Telnet      0000000000000006
DATASTREAM: CHARACTER DATA ['EXTCM   ']
```

```
21 sep 2021  11:18:40.620  0000000000000001  Telnet      0000000000000006
DATASTREAM: ORDER: SFE [55: 42/f4 c0/f0]
```

15.14 Host LU Mapping

NetPhantom provides a mechanism to map a particular client to a 3270 and/or 5250 LU name for the host. This is accomplished in two ways:

- a map table that translates TCP/IP addresses into LU names,
- a user exit in Java that programmatically establishes the LU name to be used (see *Appendix G – User Exits – LU Mapper User Exit* for more information).

TCP/IP to Host LU Name Mapping

This file with the name specified in `server.ini` as `LUMapperFileName=filename` contains lines of text data as:

```
IP-address=Host_LU_Name
```

For example:

```
193.12.212.202=CLI1234
```

This file is optional and is not used if the entry `LUMapperExit` in `server.ini` is non-existent. It is loaded into primary memory when the server starts or can be reloaded using a server administrator function.

15.15 User Authentication

This feature is optional and is disabled by default. It enables custom-made client authentication for a particular product, e.g. *RACF* for mainframes. The feature is written as a Java class that implements a specific Java interface. See *Appendix G – User Exits – The User Authentication User Exit API* for further information.

The users may log on to an active server (e.g. a server that does not have logon inhibit state or is not shutting down). Once the user is accepted, the actual host session for the user is established.

Enabling User Authentication for an Application

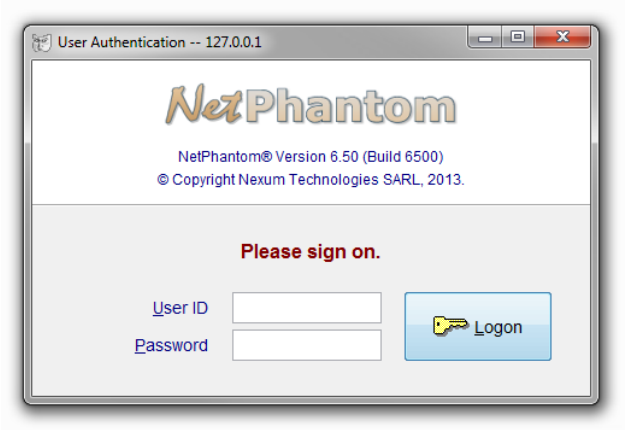
First, you must customize the user exit. Once this is done, specify a “star” (*) in front of the definition of the application in the `[Application]` section in `server.ini` as:

```
*USERAUTHCHECKEDAPPLICATION=yourdir/yourapp.jar
```

User authentication can also be done using the default user authentication exit in conjunction with a web server resource, a NetPhantom Application (NA) resource. When such a resource is defined as an **Authenticate** resource in the server administration program (menu item **Configure web server – Resources** tab), the user ID and password must match an (active) user that belongs to an active group that belongs to the resource, the NetPhantom Application, in question.

Client User Interface for Logon

The following user interface is used for client logon to the server:



If the user password has expired, the client is prompted to change the password.

If the user authentication is done by means of a NetPhantom Application (NA) resource of the NetPhantom Web Server, the new password is saved in the **users.ini** file.

The dialog boxes above are NetPhantom panels in a runtime application `USERAUTH.PHR` with the texts externalized to a text file to facilitate translation to another language. This application is also delivered with NetPhantom for editing in the NetPhantom Editor. This means that the dialog boxes may be redesigned to include other texts and e.g. a logon bitmap image.

15.16 National Language Support

NetPhantom provides a mechanism for national languages. The solution is delivered with English language as default but can be customized in a special text file.

All messages for administrators as well as messages to the clients are stored in a special text file named `server.phm`. This file consists of a text ID and a text string much like the text file used in previous versions of the system. The server displays a message according to the text ID and possible parameters.

Changing the Language

All text Ids for the client follow the naming convention `CLInnnn`. The server text Ids are named `Setnnnn`. The *t* stands for the type of message where *t* is:

- I informational
- W warning
- E error
- C critical

A normal text editor can be used to edit the `server.phm` text file. The code page for this file is specified in `server.ini` as `TextFileCodePage=`*nnn*. This code page can be both IBM (OEM) ASCII, e.g. DOS Latin-1 Cp850, or Windows code page, e.g. ISO Latin-1 Cp1252.

There is no support for changing the text file while the server is running. A server shutdown and restart will be required if this text file is changed.

16 The Server Administration Program

The Server Administration application is in `rconsole/gui/rconsole.jar`. This section describes the different functions that it can perform.

To disable remote usage of the Server Administration program, leave the item `serverAdminRuntime` in the `[base]` section in `server.ini` empty. See also *The Remote Command Line Utilities* chapter.

The client will start the Server Administration program when the application `SERVERADMIN` is specified, e.g. on the command line

```
java … se.entra.phantom.client.Phantom app:SERVERADMIN
```

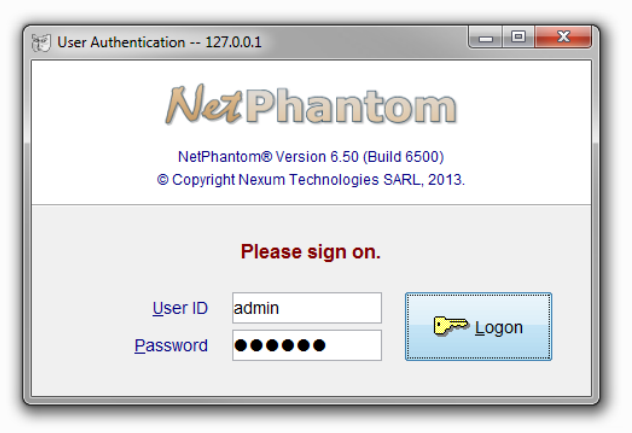
or in an applet with the parameter

```
<param name="app" value="SERVERADMIN">
```

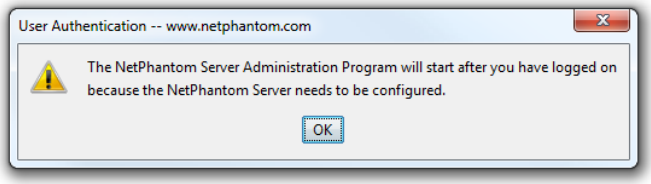
The first time the Server Administration program is started you will have to enter

User ID	`admin`
Password	`secret`

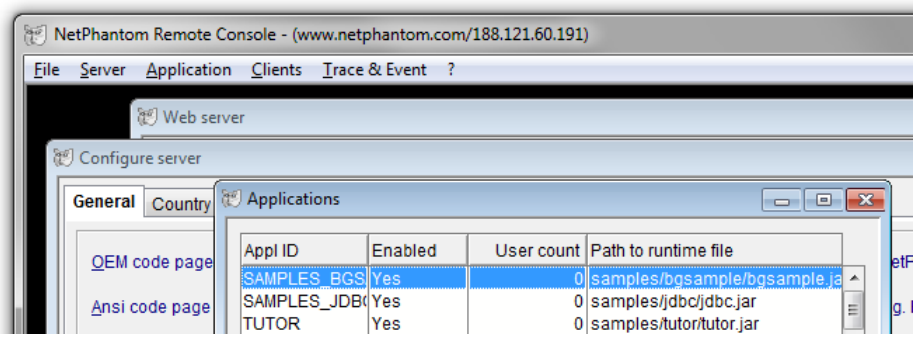
The user definition for **`admin`** can then be edited via the **Server – Manage users** menu item to not require a password.



After successful user authentication, the Server Administration window will be displayed.



The user receives a warning if the server is not properly configured. This includes license related issues, which typically occur when starting the server for the first time after installation (see next chapter and chapter NetPhantom License System for details).



The NetPhantom Server Administration program has an MDI (Multiple Document Interface).

A major part of the functions available through the server administration program is also available in the NetPhantom Editor. The editor runs a local server for testing/debugging of new application. This server instance is configured through the same basic interface as the is used by the normal server administration.

The main difference is that the interface is integrated in the editor (i.e. there is no need to log on to a separate client running an MDI). Also, the editor interface does not contain the full set of functions available in the server administration program. The following sub-chapters describe the administration functions and the way they are accessed.

16.1 License Configuration

Licensing for NetPhantom is based on the number of concurrent users, the types of users, the administration port number and the IP address or the server's host name. To activate your NetPhantom license select the menu item **Server – License** (or **Options - License** in the NetPhantom Editor)

The **NetPhantom License Code** dialog box provides you with your Server IP address and IP name. See section *NetPhantom License System* for more information.

16.2 File Transfer

File Transfer allows you copy files between your local machine and a NetPhantom Server, this helps in situations where the server is physically located in a remote location and certain files need to be transferred to it. Through the Administration Console, files may be copied to or from the file system of the NetPhantom Server. The Administration Console supplies three utilities for transferring files: the File manager (a powerful function allows copying or synchronizing of entire directories) as well as two different dialog panels for transfers of a single file: client to server and server to client.

In addition, it is possible to transfer full applications in one easy step instead of transferring an application file by file with the File Transfer utility. This is done from the Applications page of the Configure server dialog box. See *Applications* below for a description.

This function is not available in the NetPhantom Editor.

Security

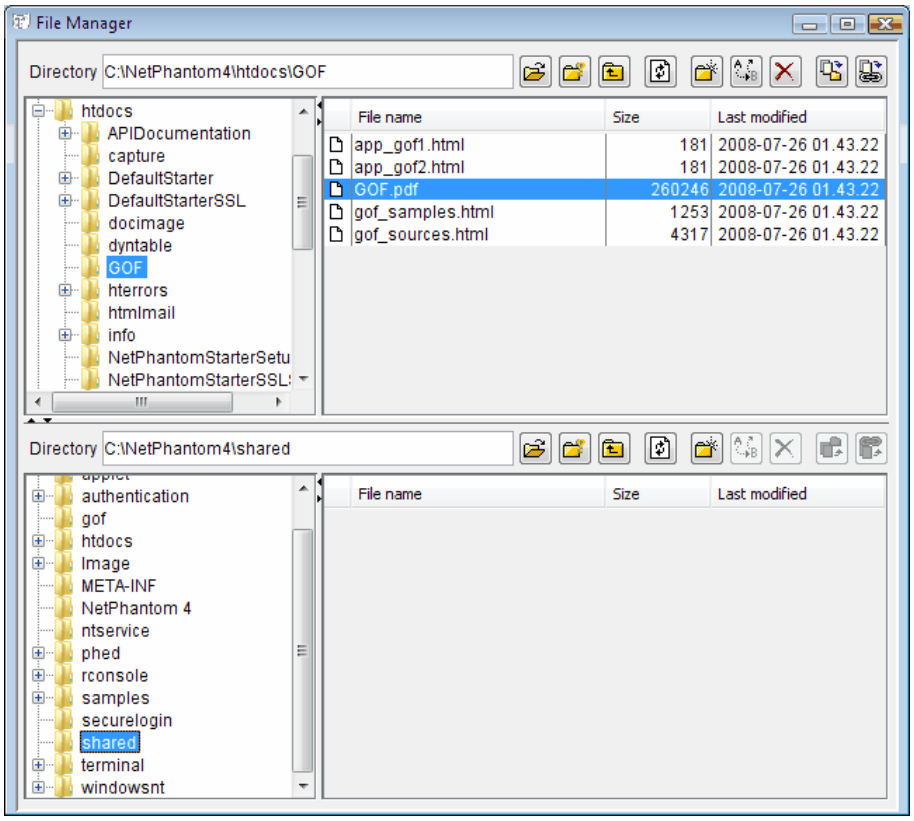
For security reasons, users of the Server Administration program are restricted from transferring/browsing files above the current directory. To allow server admin users access to files above the current directory, place the *NetPhantom.allowFullDiskRights* file in the parent directory of the current NetPhantom Server directory. It allows file transfer and browsing of files above the current directory (or using another drive for Windows). This file is installed with the Developer Kit for developer's installation.

File Manager

The File Manager design should be familiar to most PC users. The upper half of the dialog displays directories and files on the server, while the bottom half shows the directories and files on the client. The **Directory** entry field can be used to specify the root directory.

The window to the left displays a directory tree in which you select the directory that contains the files you wish to copy/synchronize. The window to the right shows the directory listing of individual files and directories contained in the selected directory. Here you select the files you wish to copy/synchronize.

> **Note:** If you are running the Server Administrator in a browser, you must use the `https` protocol or you will get a security exception.

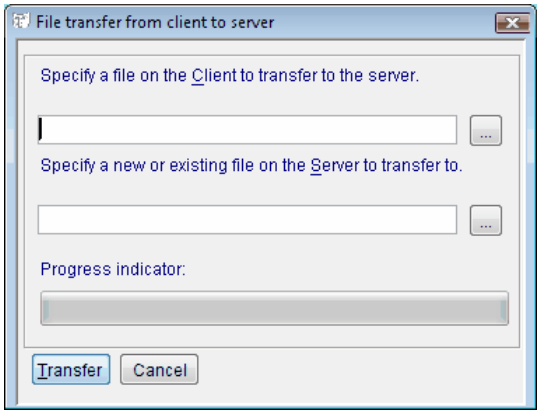


	Opens the server or client directory as the root directory.
	Selects a server or client directory as the root directory.
	Opens the parent directory of the selected directory, i.e. goes up one level.
	Refreshes the directory tree and listing.
	Creates a new directory on the server or client.
	Renames the selected file.
	Deletes the selected file on the server or client. This is a very powerful function that can delete everything from a client or a server. For this reason, a message will first be displayed asking if you are sure you want to delete the file(s).

	Copies selected files/directories to the client. Copies selected files/directories to the server.
	Synchronizes the server file with the client or vice versa by transferring only the files that differ by comparing hash codes. **Note:** It performs a cyclic redundancy check in a 32-bit value for the entire file using the CRC32 algorithm. In most cases this is sufficient, but on rare occasions it may cause files to seem identical resulting in a file not being synchronized.

File Transfer from Client to Server

To transfer a file from your client machine to the NetPhantom Server machine use the menu item **File – Transfer client to server**.



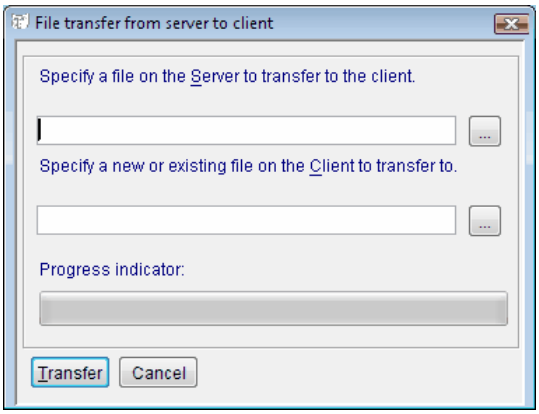
The File transfer from client to server panel.

In the **Specify a file on the Client to transfer to the server** field, you must enter the filename of the file you wish to transfer, or you may use the […] button to browse for the file. Note that the files shown in the browser window are located on your local machine.

In the **Specify a new or existing file on the Server to transfer to** field, you must enter the name of the destination file for the transfer. This file may already exist; in this case it will be overwritten with the new file. You may also use the […] button to browse the files on the server but note that the files shown are located on the remote server.

File Transfer from Server to Client

To transfer a file from the NetPhantom Server machine to your local machine use the menu item **File – Transfer server to client**.



The File transfer from server to client panel.

In the **Specify a file on the Server to transfer to the client** field, you must enter the filename of the file you wish to transfer, or you may use the […] button to browse for the file. Note that the files shown in the browser window are located on the remote server machine.

In the **Specify a new or existing file on the Client to transfer to** field, you must enter the path to and name of the local file where the remote file will be transferred. This file may already exist; in this case it will be overwritten with the new file.

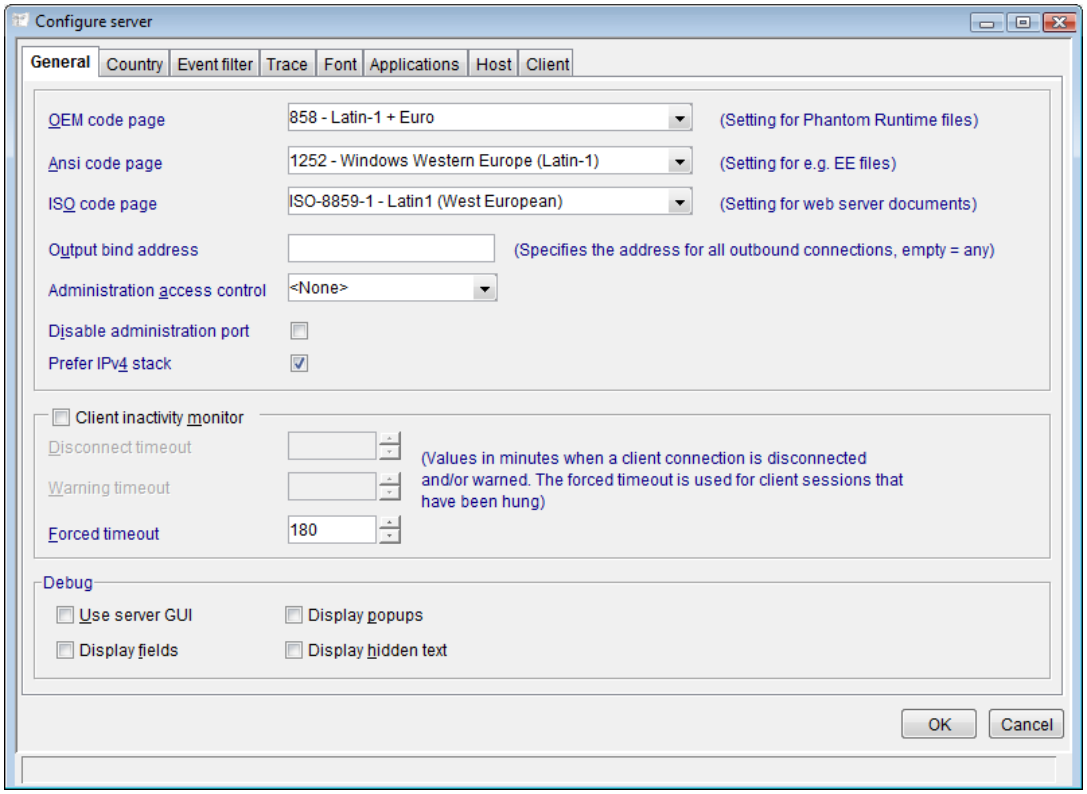
NetPhantom also supports file transfer through the implementation of a user window. See *Appendix H – File Transfer from Server to Client*.

16.3 Server Configuration

The "Configure Server message box" is displayed whenever the server is not properly configured. This may also happen the first time the server is started due to licensing issues.

The dialog box to configure the server is opened by selecting the menu item **Server – Configure – Base** (or **Server – Base configuration…** in the NetPhantom Editor). Changes made here are updated to the `server.ini` file.

General Settings

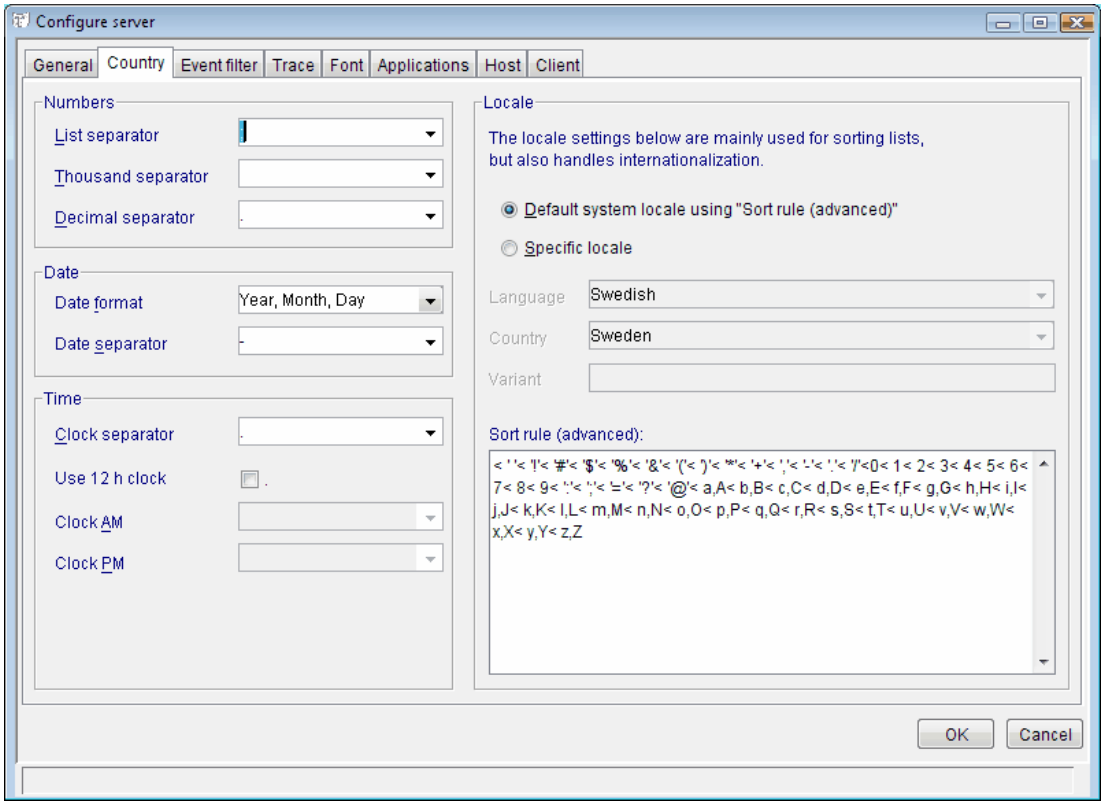


The General page defines the basic settings for server configuration.

OEM code page setting for NetPhantom Runtime files	NetPhantom only uses Unicode but needs to access text files and runtime application files stored in OEM or ANSI code pages. The NetPhantom Server contains Java classes to handle code page conversion to and from Unicode. Choose the appropriate OEM code page setting for NetPhantom Runtime files from the combination box.
Ansi code page	The ANSI code page setting is used for such things as EE files.
ISO code page	The ISO code page setting is used for web server documents.

Disable administration port	This option disables the administration port and makes it impossible from a load balancing system to work, along with the Remote Command Utilities. This option is only used in smaller installations when a very high security is required.
Prefer IPv4 stack	By default, this option is selected and lists any connections from other machines in the IP version 4 format, e.g. 123.456.789.098.
Client inactivity monitor	Check this option to set the timeout value in minutes for automatic disconnection due to client inactivity. Leave the value blank to use an indefinite timeout. An inactivity warning can be sent to the client prior to disconnection by specifying a timeout value that is less than the disconnection timeout. A Forced timeout is used to kill a hung client session. The default value is 3 hours (180 minutes).
Debug	*All options should be unchecked for production use and for servers installed in UNIX environments.*
Use server GUI	Check this option to run the server in GUI mode.
Display Fields	Enable host field display in the server terminal window when running in GUI mode for debug purposes.
Display Popups	Enable display of pop-up screens in the server terminal window when running in GUI mode for debug purposes.
Display hidden text	Enable display of hidden text in the server terminal window when running in GUI mode for debug purposes.

Country Settings

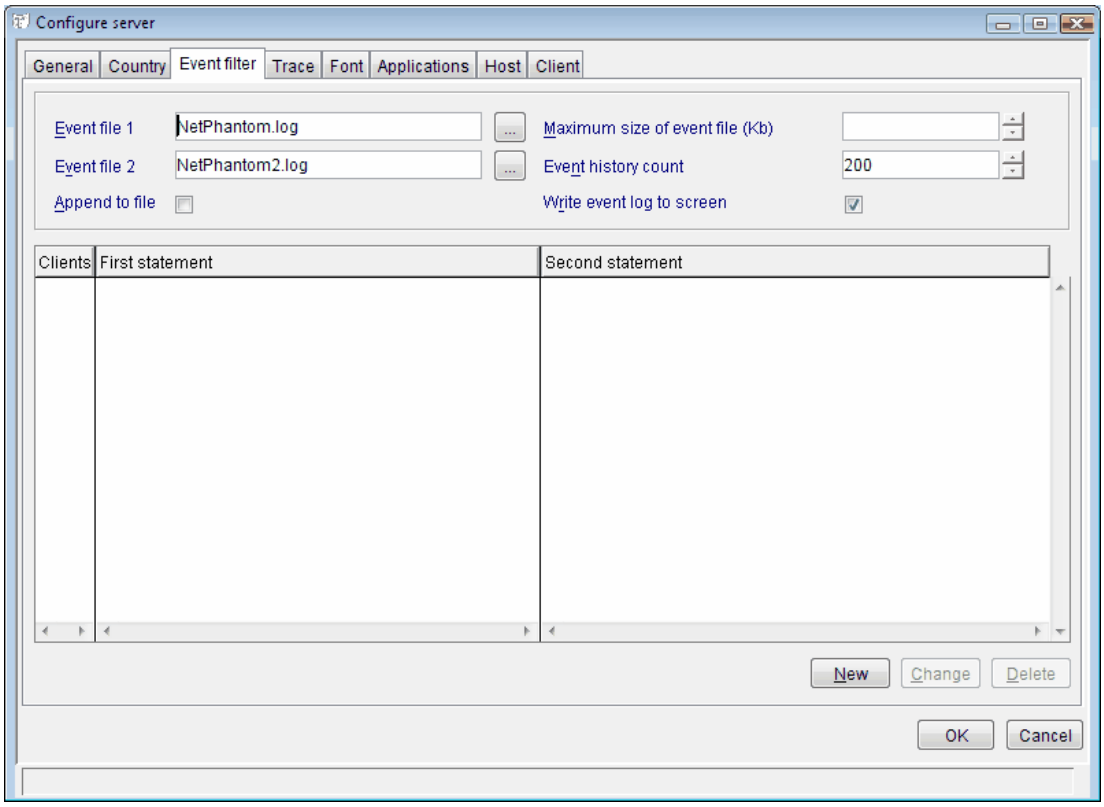


The Country page is used to specify the format and separators for numbers, time and date.

Numbers	From the combination boxes, select the characters that will be used as list separator, thousand separator and decimal separator. The only entry that may not be left empty is the Decimal separator.
Date	Select the date format and the character that will be used to separate the number in the date from the combination boxes. Date format can be: Month, Day, Year Day, Month, Year Year, Month, Day
Time **Clock separator** **Use 12 h clock** **Clock AM/PM**	 Select the character to be used to separate the hours and minutes. Check this option to if you *do not* want the time shown in 24-hour format. These options are enabled when the 12-hour clock is selected. Select the appropriate abbreviation to represent the morning and afternoon hours.

Event Filter Settings

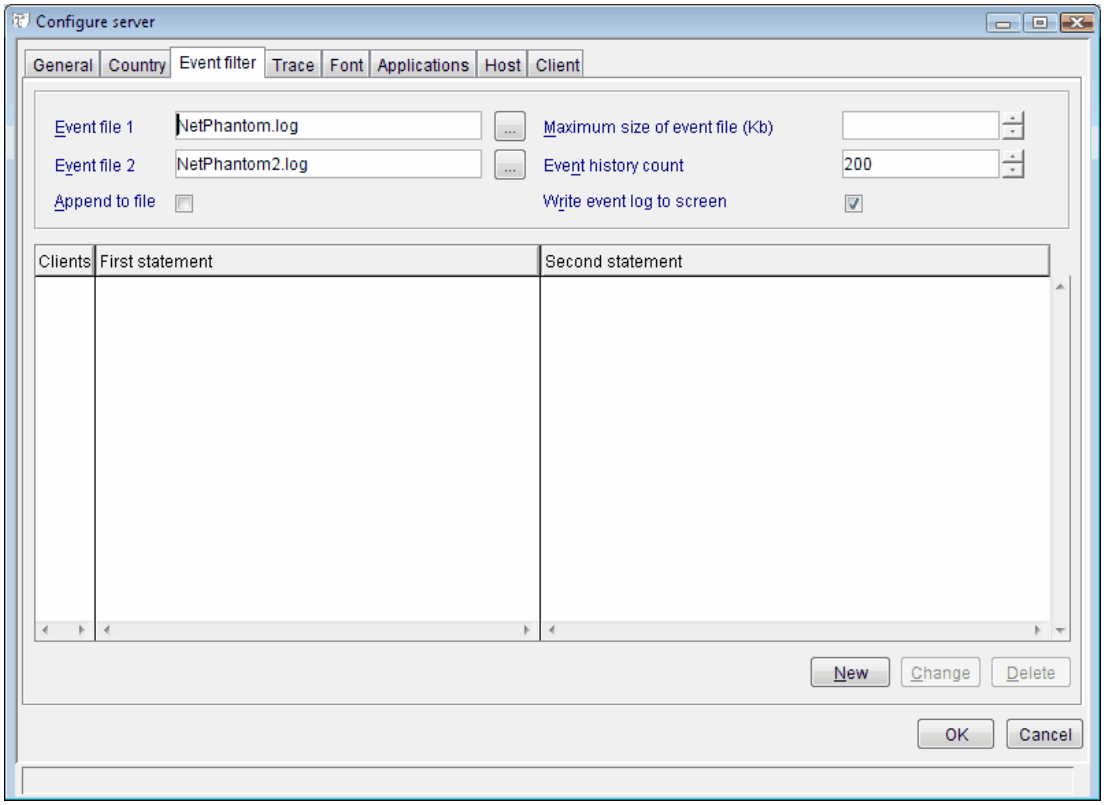
The **Event Filter** page is used to define the events that are to be included and excluded for all clients. This is desirable to prevent unnecessary entries from being written to the log files.



Event file 1 **Event file 2**	Specify the names of the two files that will be used to log events. The files will automatically be rotated when they reach their maximum size. The option **Append to file** will cause the first file not to be overwritten when the server starts.

Maximum size of event file (Kb)	The maximum event file size determines when the event log files will be rotated. To inhibit automatic file rotation, set Maximum size to 0.
Event history count	The Event history count specifies the number of saved events that the server administration program can retrieve. The minimum value is 10, maximum 1000 and default 100.
Write event log to screen	Except when the server is run as a Windows Service, the event log is written to the screen as well as to the log files. If the server is installed on the AS/400, for example, you will need to uncheck this option to prevent the AS/400 from generating a huge print file.

To create a new filter or edit an existing filter, click the **New** button, or select the filter to be changed and click the **Change** button to display the **Add/edit event filter** dialog box.

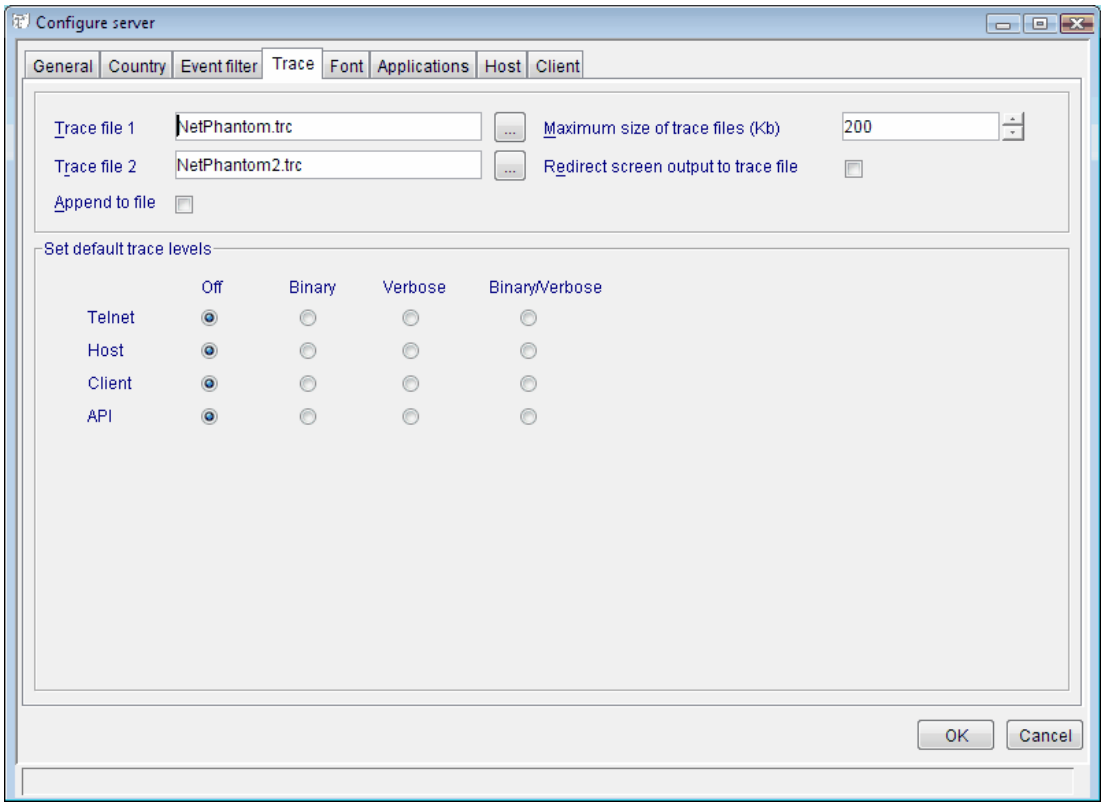


*The event filter string can be edited by hand, e.g. to use the * wildcard.*

The center list contains all events. Use the arrows below the lists to move events to and from the **Included** and **Excluded** lists. The radio buttons **Included first** and **Excluded first** are used to specify which statement should come first. When the lists have been edited as desired, click the **Create string** button. The event filter string is created and displayed and can be edited in the entry field at the bottom of the dialog box. Click **OK** to return to the **Configure server** dialog.

Trace Settings

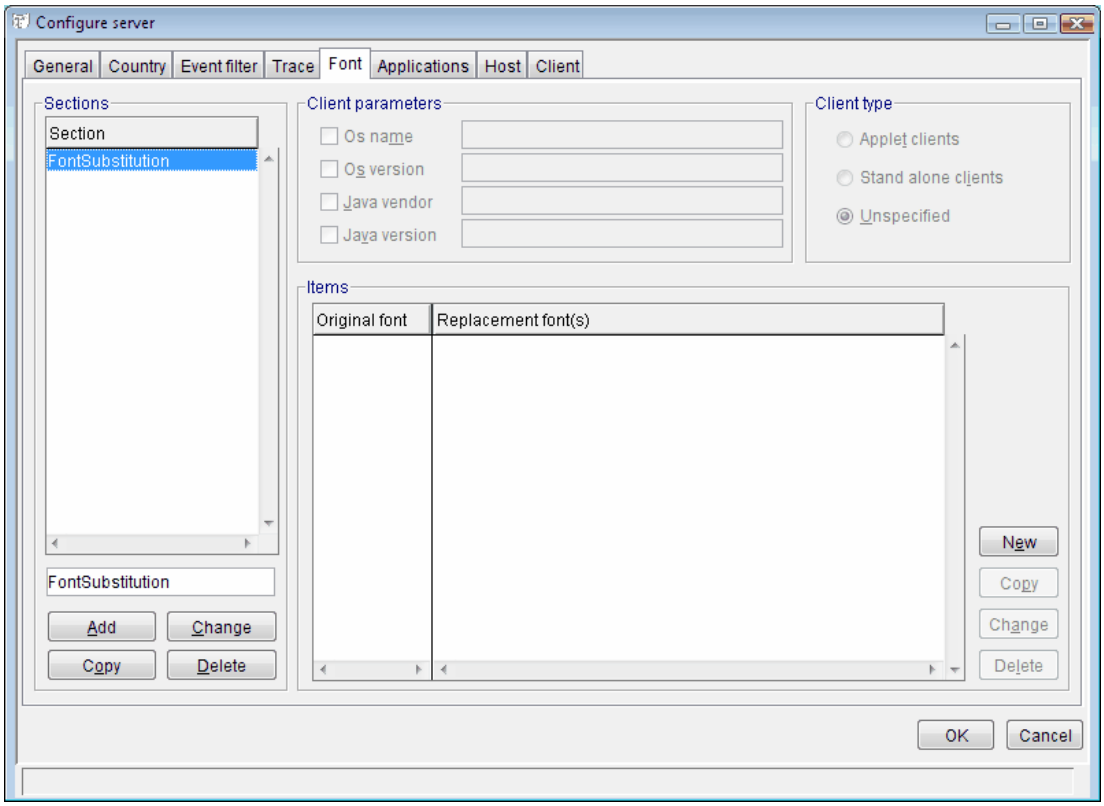
This section defines which traces should be turned on or off for all clients when the server is started. By default, all traces are turned off.



Trace file 1 **Trace file 2**	Specify the names of the two files that will be used to log events. The files will automatically be rotated when they reach their maximum size. The option **Append to file** will cause the first file not to be overwritten when the server starts.
Append to file	Check this box if you want a new trace appended to an existing trace file.
Maximum size of trace files (Kb)	The maximum trace file size determines when the trace files will be rotated. To inhibit automatic file rotation, set Maximum size to 0.
Redirect screen output to trace file	Check this box to have the screen output written to the trace file rather than the screen.
Set default trace levels	To specify that a trace should be turned on when the server is started, click the radio button for the desired trace level.

Font Substitution

The Font page is used to allow a specific font on a client to be remapped into another font. This mapping mechanism also allows different font mapping for different categories of clients.



The Font page allows you to define both default and client specific font substitutions.

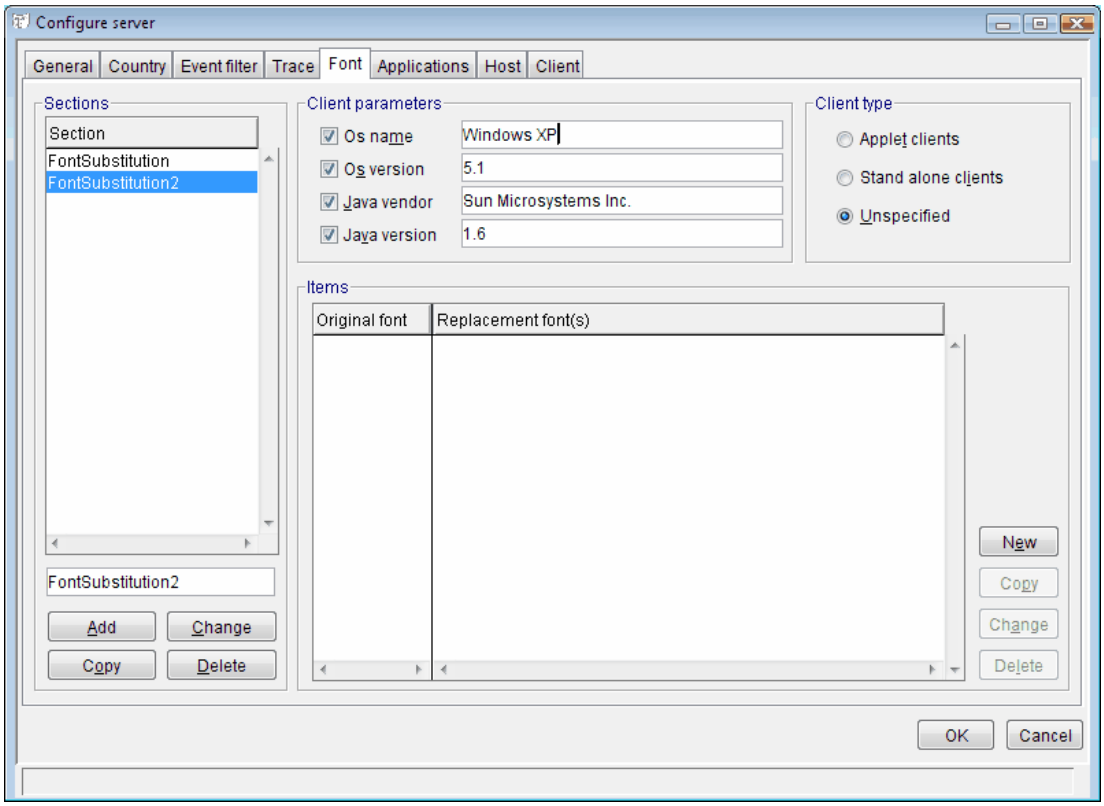
To remap a font, you must first create a section.

1. Enter the name of the new section in the entry field below the **Sections** list and click the **Add** button. For example, `FontSubstitutionWIN`.

2. Select the new section in the **Sections** list and the **Client Parameters**, **Client type** and **Items** sections will be enabled.

   The **Client Parameters** and **Client type** options allow you to define client specific font substitutions, while the **Items** list contains the list of original and replacement fonts.

Let us continue by creating a client specific font substitution.



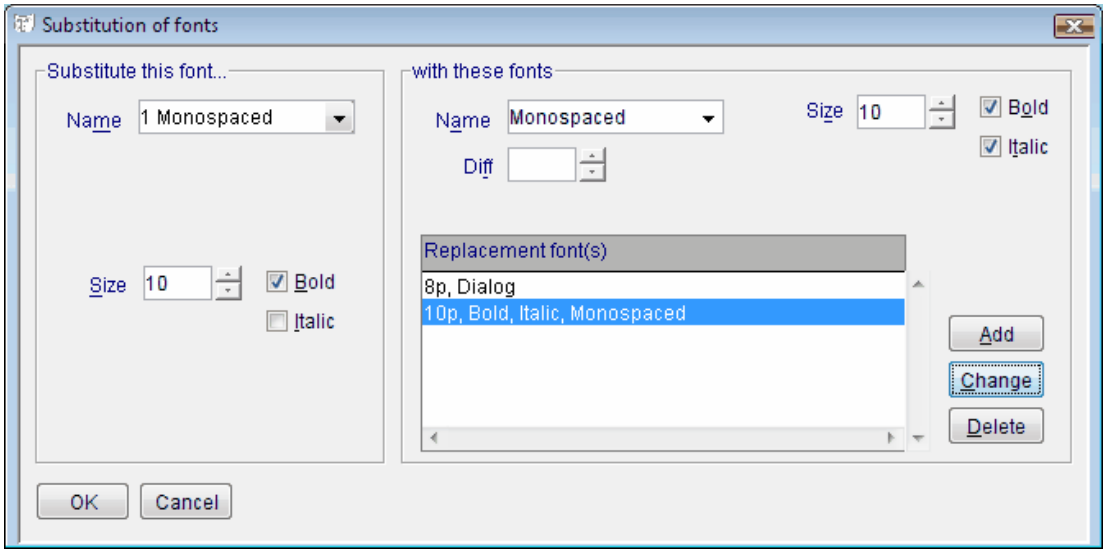
1. Check the boxes for the criteria to be used to determine that a particular client will use the new font substitution section rather than the default substitution section. In this case we will check all boxes and fill the corresponding entry fields with the following information.

   Os name `Windows XP`
   Os version `5.1`
   Java vendor `Sun Microsystems Inc`
   Java version `1.6`

2. To complete the client selection criteria, select the type of client, in this case we will choose to use this section when running the client as a Standalone application.
   We are now ready to remap the font.

Remapping the Font

1. To specify the original and replacement font, click the **New** button to the right of the **Items** list. The **Substitution of fonts** dialog box will be displayed.



2. Select the name of the original font from the **Name** combination box under **Substitute this font...**, for example, `Monospaced`.

3. Select the font size in points, `10`, and check the **Bold** option.

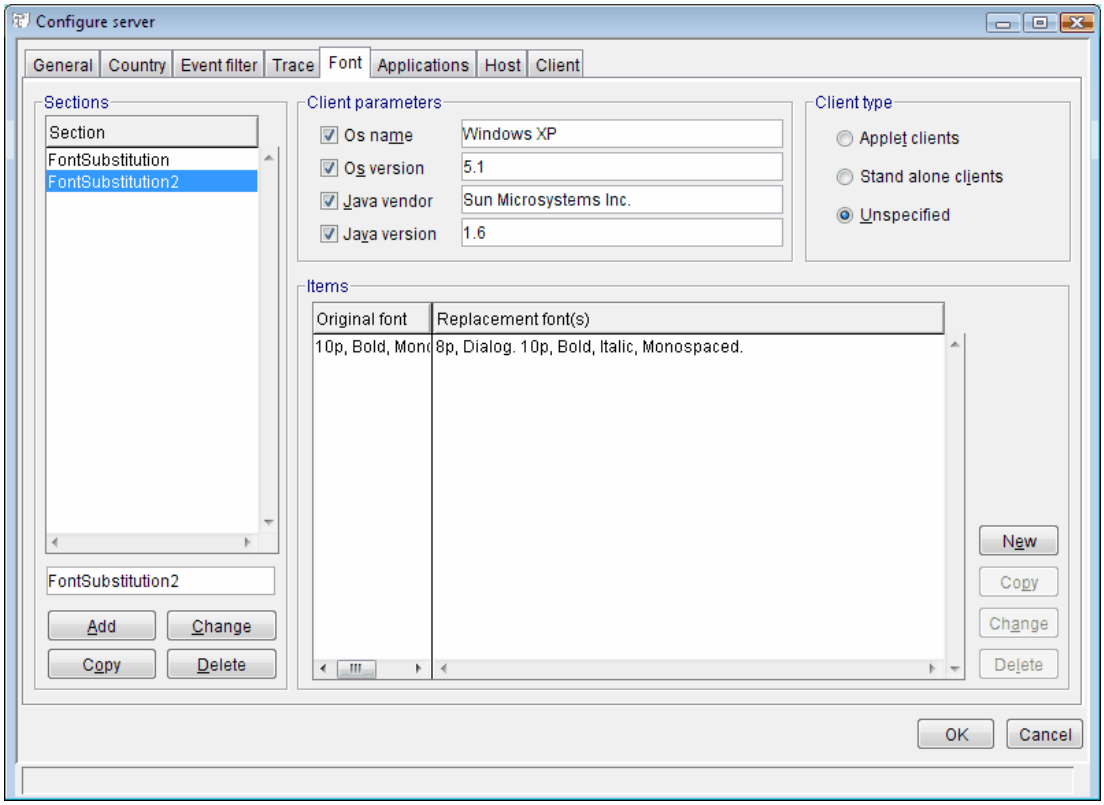
4. You may choose more than one replacement font under **with these fonts**, but they are added one at a time to the replacement font(s) list. Specify the following:

   Name `Dialog`
   Size `8`
   Bold `Checked`
   Italic `Empty`
   Diff `Empty` (see note below)

5. Click the **New** button and the replacement font is added to the list. Repeat the procedure with the following information:

   Name `Monospaced`
   Size `10`
   Bold `Checked`
   Italic `Checked`
   Diff `Empty` (see note below)

6. Click **OK** to return to the **Font** page.

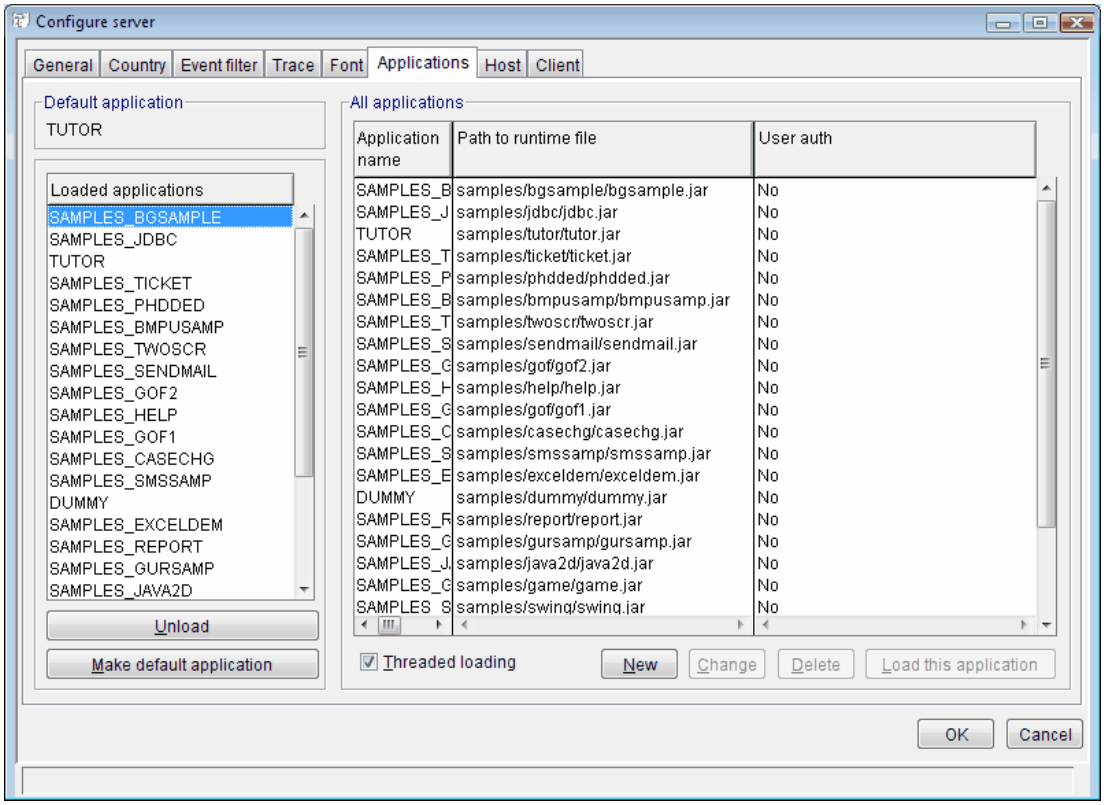


You have now created a client specific font substitution. All Standalone clients, running under Windows XP v. 5.1, using Java Virtual Machine 1.6 will replace Monospaced 10 Bold with Monospaced 10 Bold/Italic

Note: The **Diff** option causes the height of the font to be adjusted by the number specified in pixels (the number may be negative). This new font height is only used by list boxes and is intended to make a perfect match for a font under e.g. the Windows environment with a font for the NetPhantom Client. This allows you to make a NetPhantom application look exactly like it did in previous versions of the system.

Applications

The **Applications** page is used to specify the NetPhantom applications that will be loaded into memory during server startup. The applications are shared by all users. Each user may run any combination of the loaded applications with the Merge-on-the-fly technique.



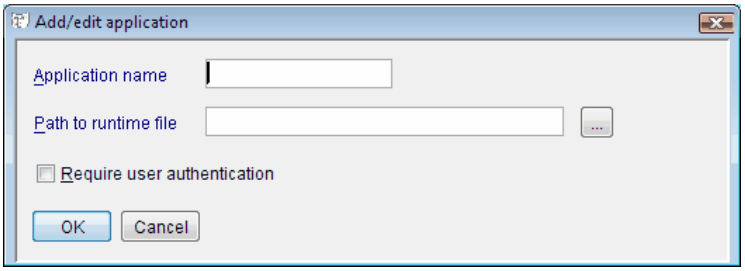
The New, Change and Delete buttons are used to maintain the complete list of All applications, while the Load and Unload buttons add and remove applications from the list of applications to be loaded at server start.

This page contains two lists: **All applications**, which is a complete list of all defined applications, and **Loaded applications,** which is a list containing only those applications to be loaded at server start.

Because clients can request applications that the server may not have loaded or may not request any specific application, a default application must also be specified. Select the default application in the list of loaded applications and click the **Make default application** button.

To add an application to the list of loaded applications, select it in the **All applications** list and click the **Load this application** button. To remove an application from the list of loaded applications, select it and click the **Unload** button.

To define a new application, click the **New** button.

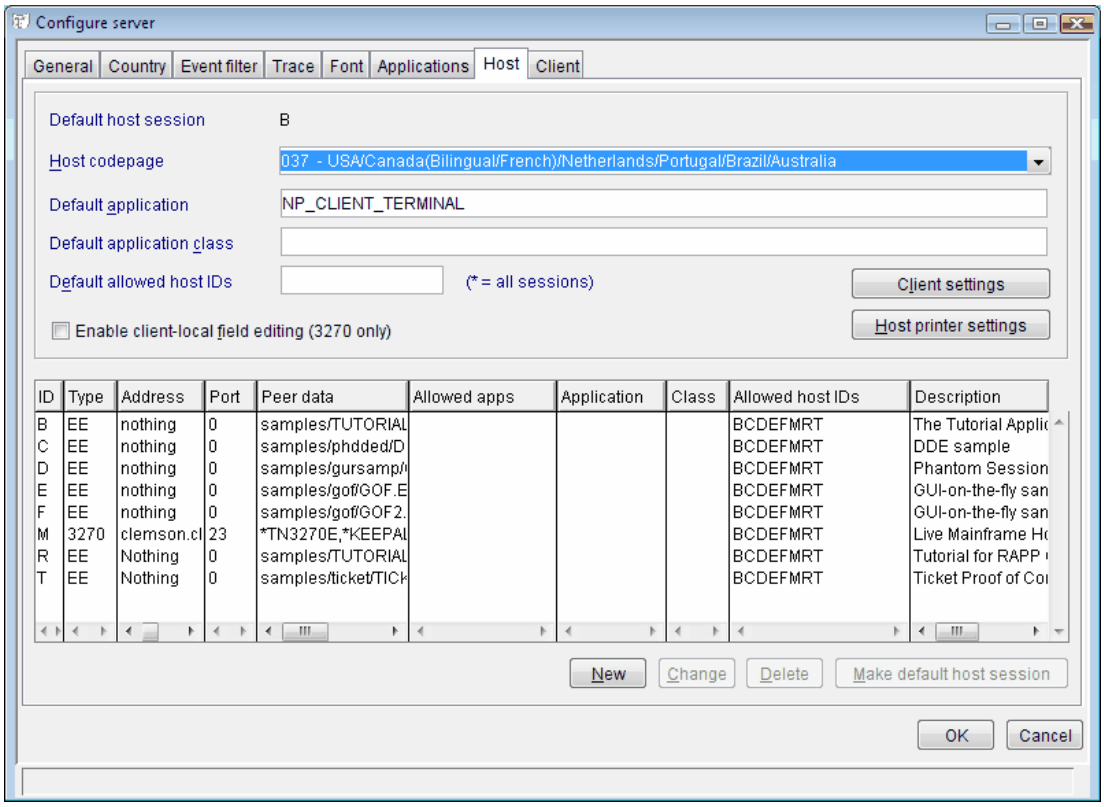


Enter a name for the application and specify the path to the runtime file (.jar). Check the **Require user authentication** checkbox only if the application will be using the User Authentication User Exit for user authentication, otherwise leave it blank. For more on this see *Appendix F – User Exits, The User Authentication User Exit*.

When you click the **OK** button, the application will be added to the list of defined application, and you will be asked whether you wish to load the application. If you click **No**, you can load the application later.

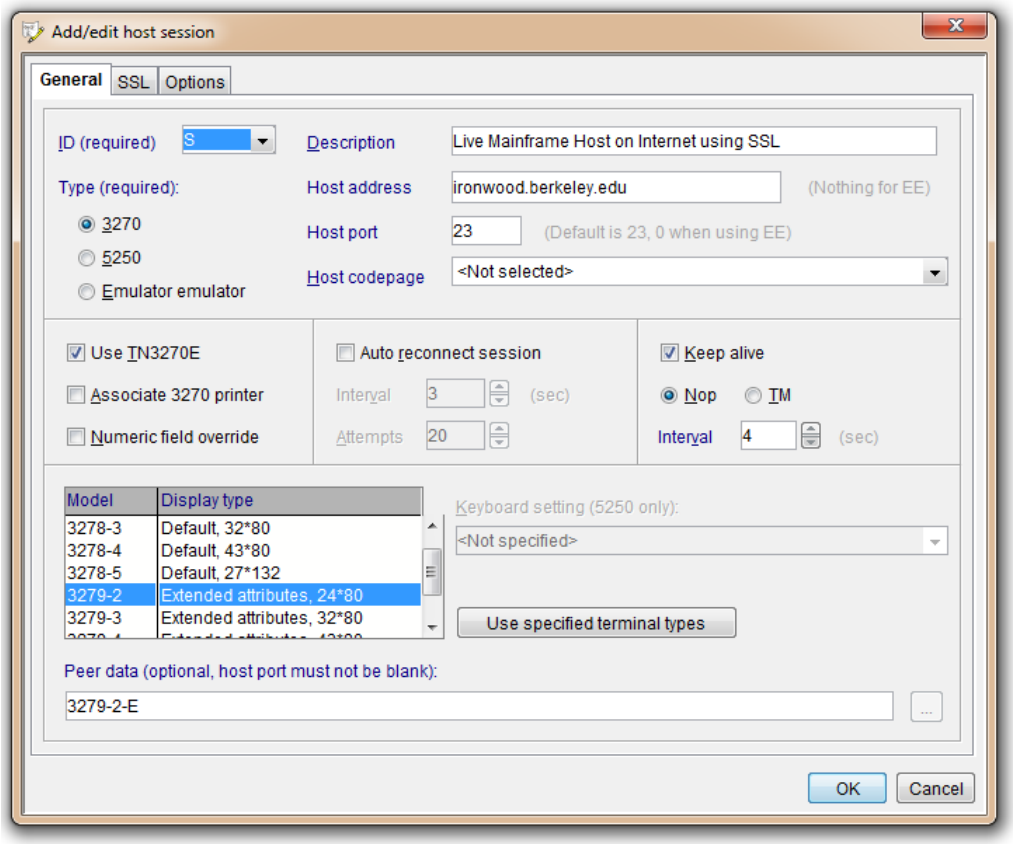
Host Settings

The **Host** page contains a list of the different host session IDs available for clients. Each host session can be defined only once within the range A-Z. The REXX function *HostConnect* is used to change between the sessions. A client may request no host connection using the parameter *HostConnect(".")*



The New, Change and Delete buttons are used to create and maintain the list of host session IDs.

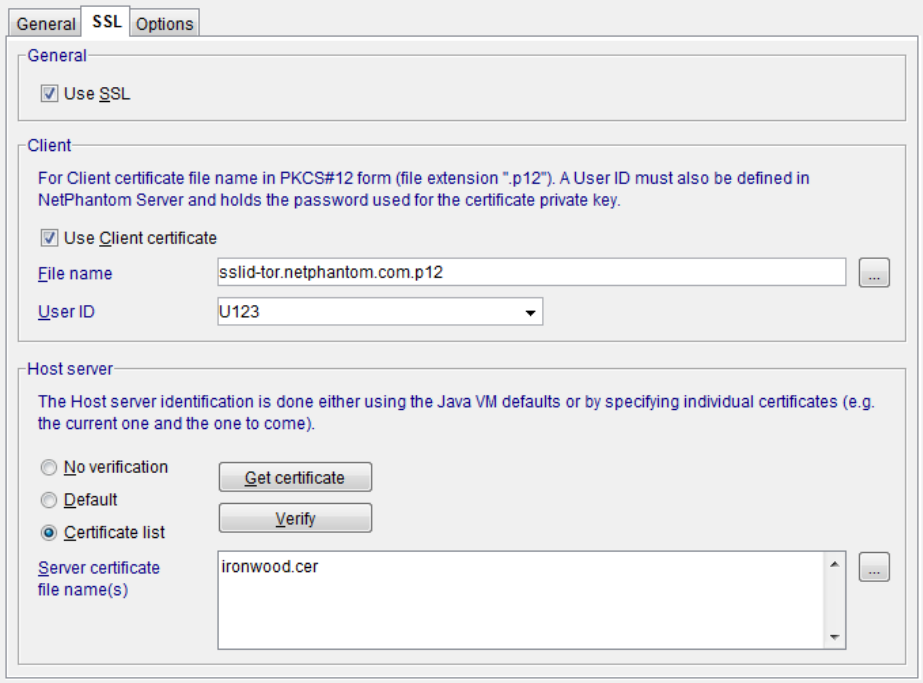
Host codepage	The Host codepage is a global setting for all host sessions. Select the appropriate EBCDIC codepage from the combination box.
Default host session	A default host session is required. This is the host session that the client will use if it has not specified the parameter `HOSTID` or is invalid. Select the desired host session from the list and click the **Make default host session** button.



ID (required)	Select a Short session ID from A-Z from the combination box.
Type (required)	Select 5250 for an AS/400 session, 3270 for a mainframe session or Emulator emulator when running NetPhantom with `ee`.
Host Description (optional)	A brief description of the host.
Host address (required)	Enter the host address, i.e. the IP address or name of the host. If you are using ee, the host address is defined as “Nothing”
Host port (optional)	The Host port definition is optional. Default is 23.
Peer data (optional)	It is only possible to specify peer data if a host port has been defined. The peer data in the configuration is used for 3270/5250 as the terminal type. Several terminal types may be specified. When the Telnet session is established, a negotiation of which terminal type to use will take place. If no terminal type could be negotiated, the host session will be closed with an error. It is important to specify the correct terminal type, because it is used to decide the terminal emulator presentation size (i.e. the screen size).
Use TN3270E	Enables full TN3270E protocol, which is required for printing.
Associate 3270 printer	This option starts a printer session associated with the terminal session. It requires that the host be configured for “associate printer” support for the TN3270E protocol.

Numeric field override	The 3270 host may the **Numeric field override** setting. This option allows all characters to pass through “Numeric Only” fields and on to the host. Standard 3270 host applications allow the definition of data input fields as Numeric Only or Alpha-Numeric. The 3270-defined “numeric” characters typically include the following: The digits 0 through 9 All capitalized alpha characters (A through Z) The following special characters: + - _ < > | , . : * ! \ “ @ $ % & / ( ) { } ? ~
Keyboard setting (5250 only)	Choose a predefined keyboard setting. It is only available if 5250 host type is selected.
Use specified terminal types	When the desired terminal types have been selected in the **Model/Display type** list, click this button to create the string that will be sent to the `server.ini` file. It will be displayed in the entry field at the bottom of the dialog box and may be edited if desired.

The port number used for Telnet with SSL is normally 992 but can very well be 23 as for plain Telnet connections, much depending on the host server configuration.

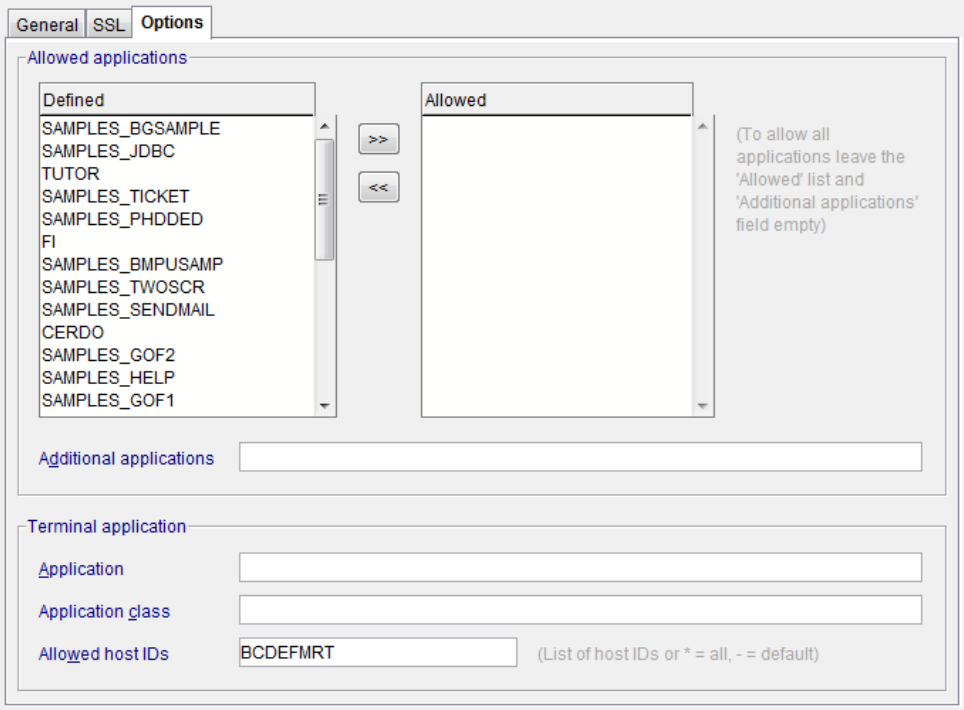


The NetPhantom Server can identify itself using a **Client certificate** and this should be specified in the **File name** below the **Use Client certificate** checkbox. This file name must refer to a file containing a client certificate of a file in the PKCS#12 format (**.p12** or .pfx). This file can be loaded using a password, and as the server needs to load the certificate itself, create a **User ID** that holds the password of the file. Specify the User ID in the combination box.

Communication between NetPhantom and the **Host server** can be verified upon connection using a certificate. Use the push buttons **Get certificate** to establish a live connection to the host server and then retrieve the certificate. This certificate is then stored in a file on the NetPhantom Server. Several certificates can be used, typically when a new certificate will be applied to the host server at a future time, and that the NetPhantom Server must be up-and-running without interruption.

Another way of identifying the Host server is by using the **Default** Java Virtual Machine mechanism to verify the identity of the server, e.g. if it has a certificate signed by VeriSign and that VeriSign is a trusted certificate authority in the Java VM.

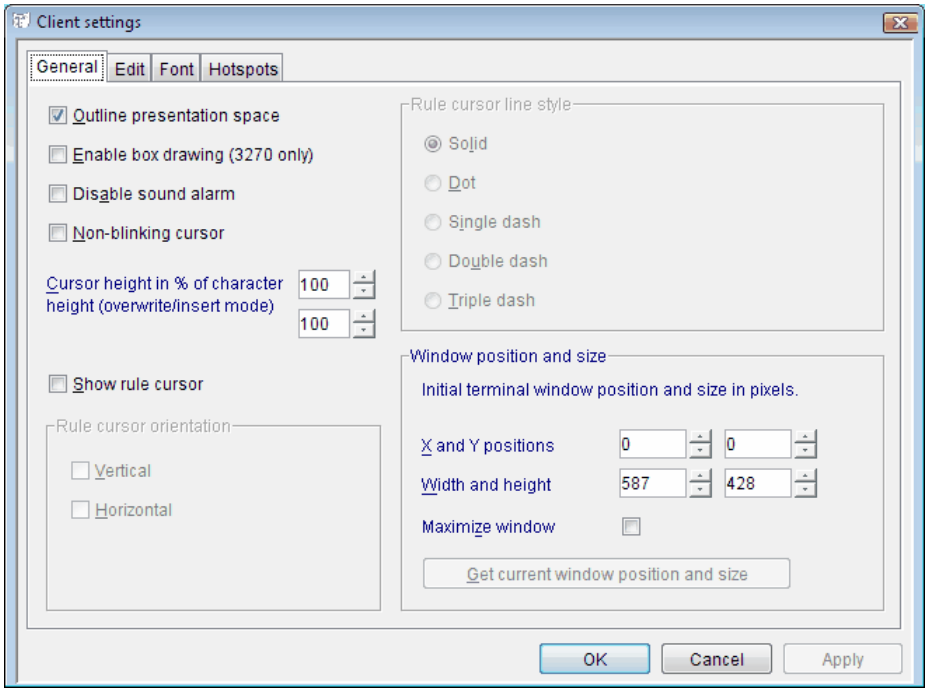
The **Verify** push button is used to verify the connection between NetPhantom Server and the Host server with the optional Client certificate and/or Host server identification check.



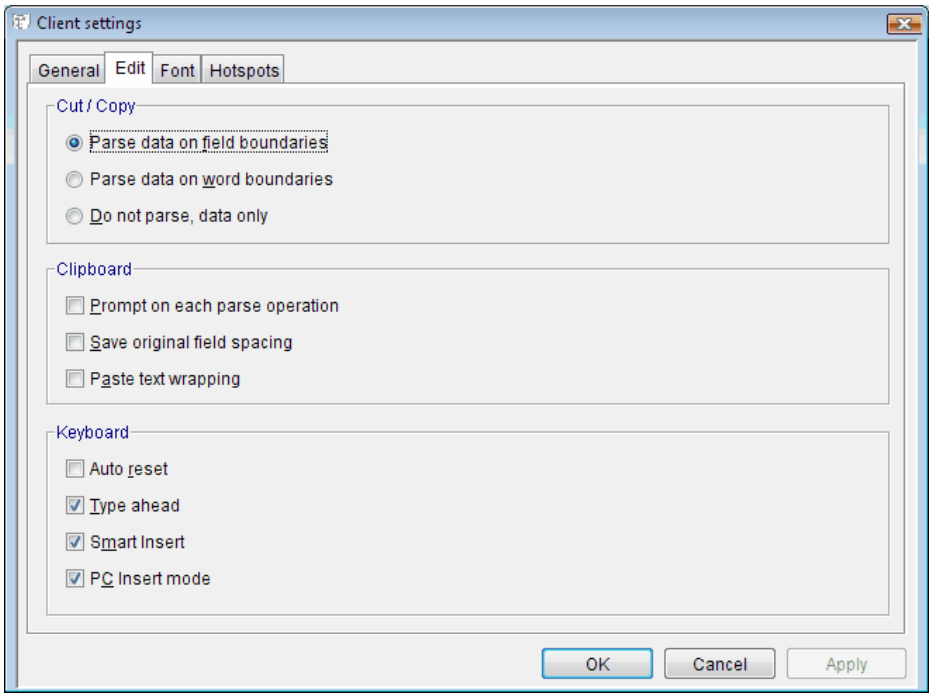
Allowed applications	You can optionally choose to specify which applications will be allowed to use the host session. Select the desired applications from the **Defined** list and click the arrow button to move to the **Allowed** list. Applications not in the **Defined** list can be added in the **Additional applications** field. To allow all applications access to the host session, simply leave the **Allowed** list and **Additional applications** field empty.
Additional applications	Extra applications that were not in the list above.
Terminal application	If a specific customized terminal application is required, it is specified here. For more information, see chapter Host Settings.

Host Client Settings

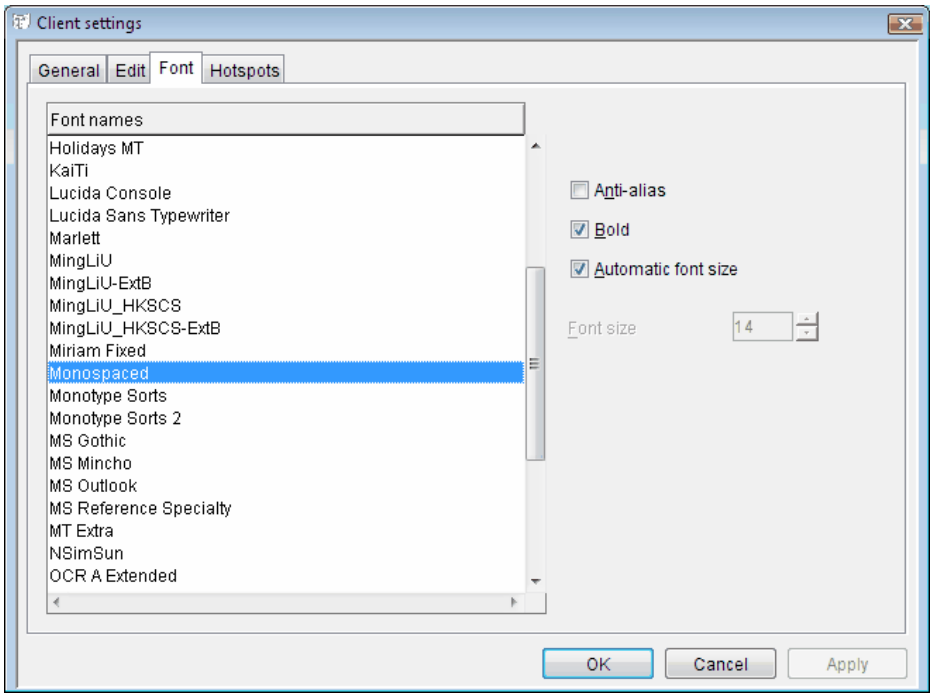
The host client settings are accessed using the push button on the **Host** tab.



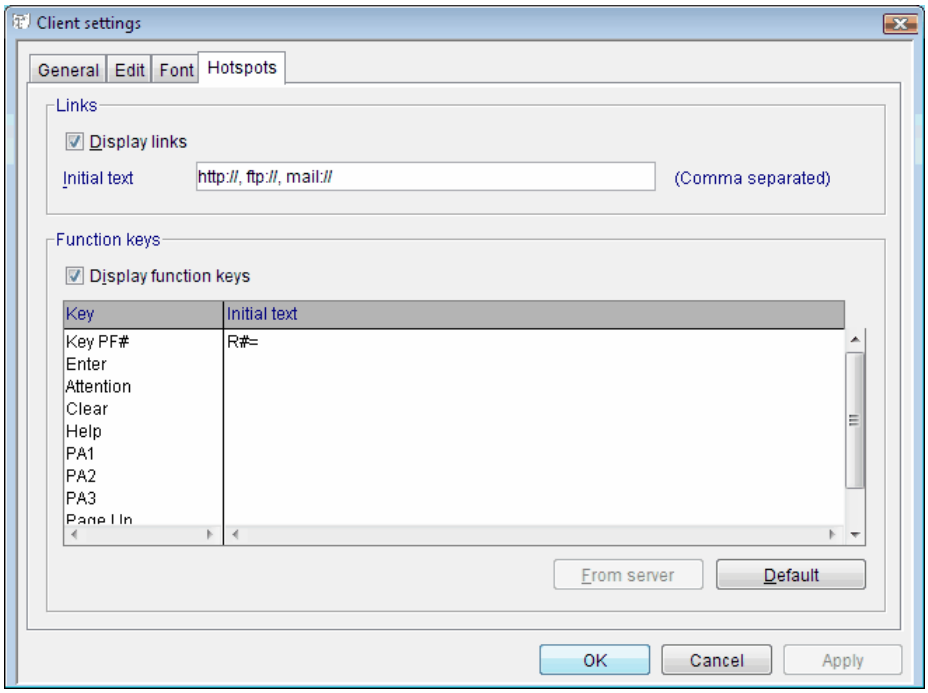
Outline presentation space	Use a framed terminal window.
Enable box drawing (3270 only)	Use line graphics to represent the 3270 type box (pop-up etc) character graphics.
Disable sound alarm	Do not use sound alarm.
Non-blinking cursor	Do not use blinking cursor.
Cursor height	This option specifies how much of the cursor that should be shown.
Rule cursor	Configuration of the rule cursor (= horizontal/vertical lines aligned to the cursor.)
Initial terminal window position and size in pixels	You can specify the initial X and Y positions and size of the terminal screen for clients.



Cut/Copy	Specify preferences for cut/copy actions: • Parse and include the data located at field bounderies. • Parse and include the data located at word bounderies. • Do not include any data from bounderies.
Clipboard	Specify clipboard preferences: • Prompt for the cut/copy preferences each time. If not selected, use preferences above. • Keep the field spacing when pasting. • Wrap text when pasting in a limited area
Keyboard	Specify keyboard preferences: • Auto reset. When the keyboard enters an error state after e.g. “field full”, you normally have to press the Reset key. With this option enabled, any keystroke will automatically perform a reset before the keystroke is processed. • Type ahead, allows you to input text when the host session is locked: those keystrokes will then be processed. • Smart insert mode (a space at the end of a field is the same as if it was never typed by the user, i.e. the “null” character). • PC Insert mode (keep the insert setting you have chosen until the next time you change it rather than losing the insert mode when you send Enter for example).



Font names	Selection of one of the available fonts.
Anti-alias	Use font smoothing.
Bold	Use bold font.
Automatic font size	Decide on the font size based on available space (in run-time).
Font size	Explicitly set the font size. For this alternative to be available, the automatic font size alternative must not be selected.

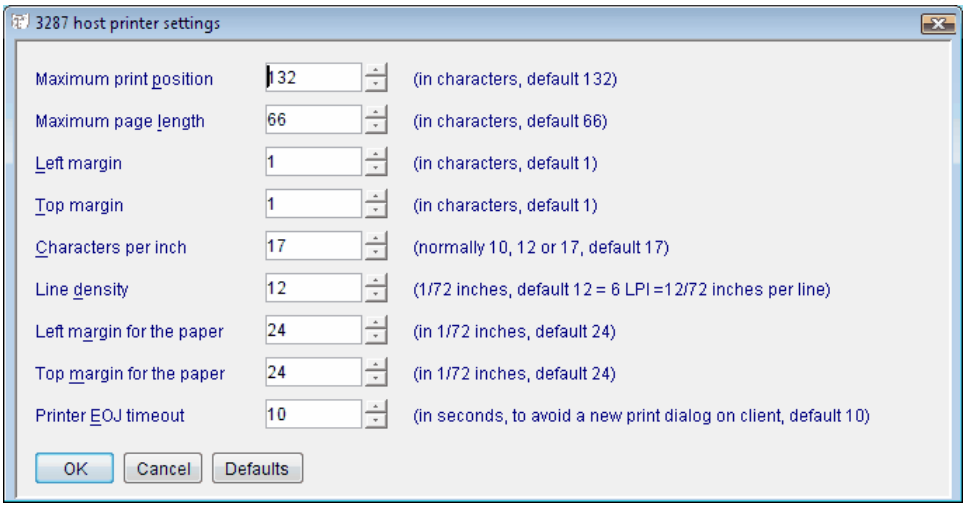


Links	Configuration of hypertext hotspot. The comma-separated strings specify the text that triggers a string to be interpreted as a hyperlink. If the text is in the beginning of a field or after two space signs the string is displayed as a hyperlink in the terminal. When activated, the hotspot will perform an action corresponding to the supported protocols: http*[s]*:// - open the standard web browser ftp*[s]*:// - open the standard ftp application mailto: - open the standard mail application As input, the string following the initial text is used when the hotspot is activated.
Function keys	Configuration of the function key hotspots. The parsing/identification of this type is the same as for the hyperlink-type. The list contains first a generic function key configuration and then a list of named function keys. The generic function key uses a '#' sign as an indicator for a number 1 to 24. This corresponds to the function key that is sent when the hotspot is activated. The named function key will always send the corresponding function key when activated.

3287 Host Printer Settings

The 3287 host printer settings are accessed using the push button on the **Host** tab.

The printer settings may also be set at runtime by a server Java class accessing the Java API. See the JavaDoc for more information, in conjunction with the LU Mapper User Exit.



*The **3287 host printer settings** dialog is used to specify settings, such as margin size, characters per inch ad line density.*

Client Settings

The Client page is used to configure client string caching and set other client attributes such as the look-and-feel and the Print window keystroke.

String Caching to Increase Performance

NetPhantom uses an algorithm for string caching. When a panel is created on the client side or when an existing panel is updated, a transaction is sent from the NetPhantom Server. This transaction defines the panel components and attributes. A panel contains a lot of text strings in menu items, static text, listbox cells, etc. Many of these strings exist in several panels, so caching these strings makes the transactions smaller. Very often an end-user works with a limited set of panels/functions in the application. The strings for the panels are then be cached even better, thus reducing the transaction sizes even more.

The reduction of the transaction size using string caching is on average about 30 %. If the NetPhantom Client is connected to the Server using SSL and a slow communication line, the performance increases by about 50 %.

Configuration of String Caching

The string caching algorithm is configured using three parameters: number of cached strings (`stringCount`), minimum length of a string (`minLen`) and the maximum length of a string (`maxLen`). The length of the strings is one byte per character (for codepage ISO 8859-1) and twice the character count for Unicode (e.g. Greek).

The `stringCount` parameter range is 50 to 8192, `minLen` range is 4 to 2047 and `maxLen` range is `minLen + 1` to 2048. To disable string caching, set `stringCount` to zero.

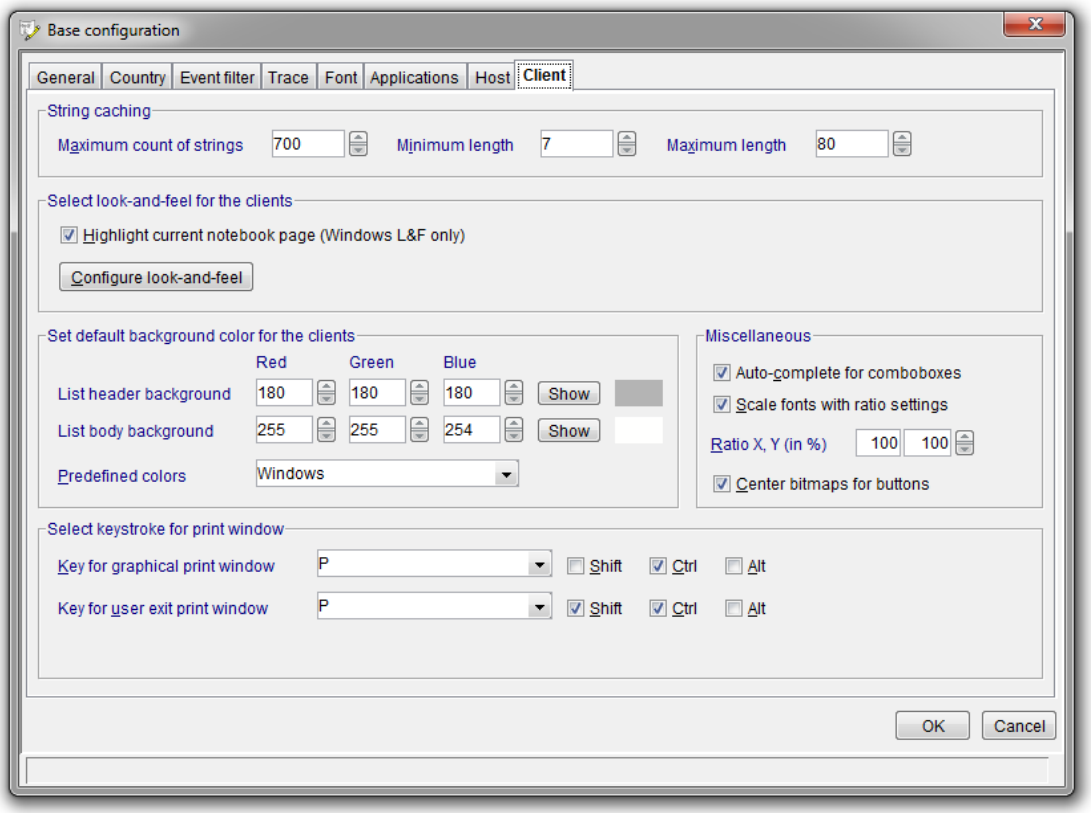
The string caching requires more memory on the client and server side. The client cache memory requirement is about:

```
2 * stringCount * averageStringLength
```

where `averageStringLength` normally is `(minLen * 5 + maxLen) / 6`.

> **Note:** As the cache is client-related, the memory requirement in the server must be multiplied with the number of concurrent users.

Good values for string caching have been proven to be Maximum count 700, Minimum length 7 Maximum length 80.



String caching **Maximum count** **Minimum length** **Maximum length**	To deactivate string caching set maximum count to 0. Be sure to use the algorithm above to calculate the client memory requirements. **Maximum count** is from `50 to 8192` **Minimum length** is `4 to 2047` bytes **Maximum length** is `Minimum length + 1 to 2048`
Select look-and-feel for clients	Select the client look-and-feel setting for Swing. Note that Windows look-and-feel is only supported on Windows platforms.
Highlight current notebook page	This option is only available for Windows look-and-feel. The tab of the current notebook page will be lighter than all the non-selected tabs (see picture above).
Set default background color for the clients	Here you can specify the list header and body default background colors in Red Green Blue (RGB) format. You can also choose from a list of predefined colors.
Miscellaneous	Specify the options for panel scaling if required, along with possible font scaling at the same time. Also choose the option for the auto-complete function in comboboxes.
Keystroke for graphical print window	Set a Key or combination of keys that will be used to send the current window as a bitmap to the printer.
Key for user exit print window	When a user exit is configured in the [UserExitPrint] section in server.ini, the keystroke combination is activated and is configured as a combination keys.

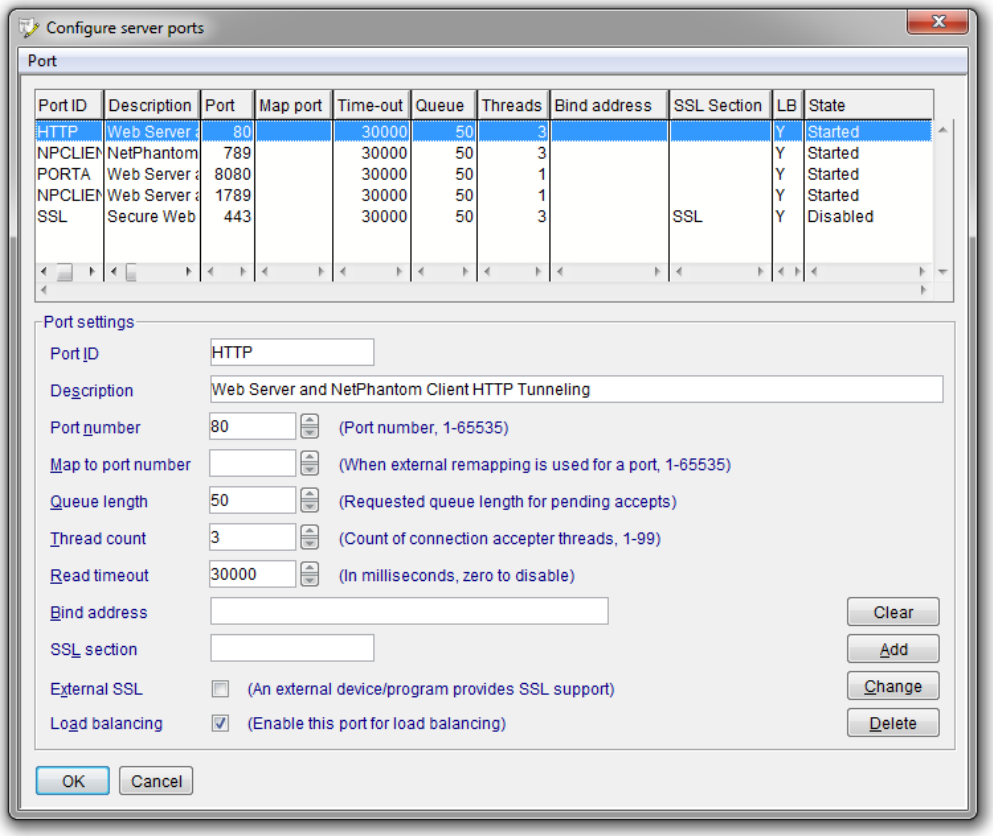
16.4 Port Configuration

The dialog box to configure server ports is opened by selecting the menu item **Server – Configure – Ports** (or **Server – Configure ports…** in the NetPhantom Editor).

A new port is configured by filling in the **Port settings** information and then clicking the **New** button. To edit an existing port configuration, select it in the list and click the **Change** button.

The popup menu for server ports is activated by clicking the right mouse button. From this menu you can start, stop, disable and close the selected port.

Start	Starts the socket.
Stop	The socket will still listen but will reject connections.
Close	The socket will be closed, i.e. it will stop listening.
Disable	The socket will be stopped and will not be started at the next server start.

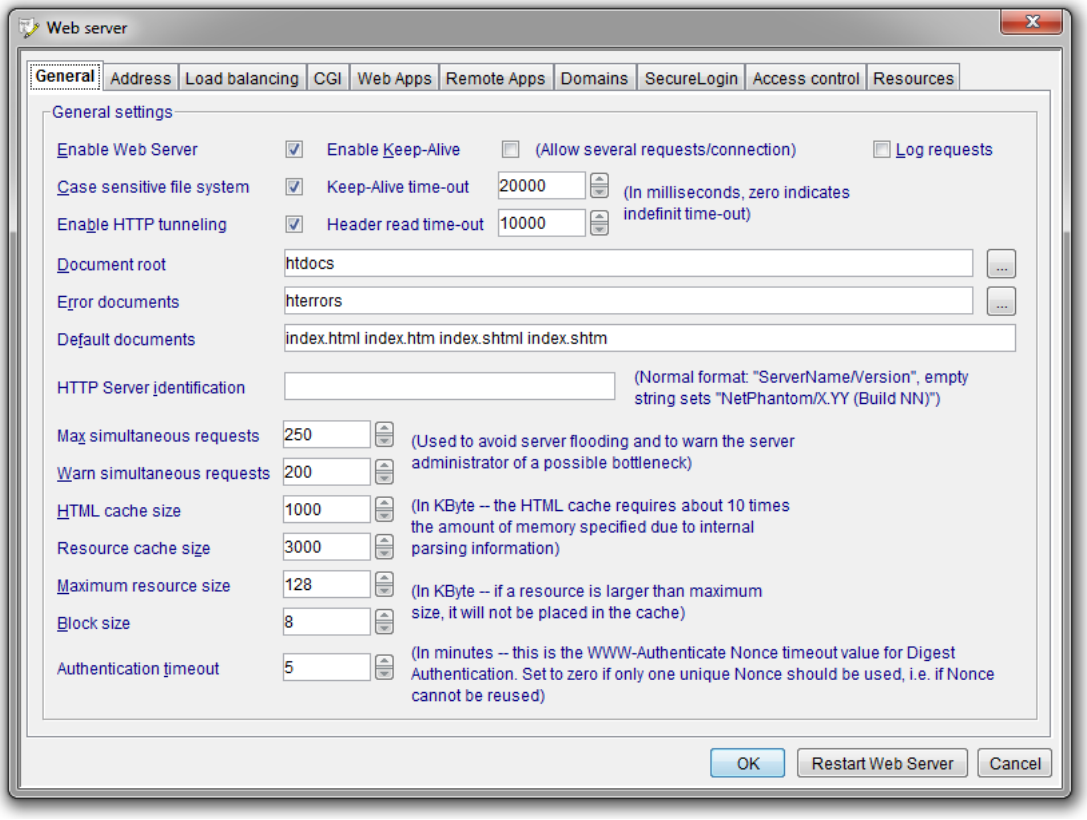


Port ID	Specify the ID by which the port will be known.
Description	Enter a descriptive text for the port entry.
Port number	Specify the port number in the range 1 to 65535. The de facto standard port for HTTP is 80 and for SSL is 443, although other port numbers can be used.
Queue length	The maximum queue length for incoming connection indications (a request to connect). If a connection indication arrives when the queue is full, the connection is refused.

Bind address	The bind address argument can be used on a multi-homed host for a server socket that will only accept connect requests to one of its addresses. If the bind address is left empty, it will accept connections on any/all local addresses. The address can be either an IP name or a raw IP address (e.g. 123.234.132.243).
SSL section	The section in the `server.ini` file describes the SSL settings. Leave empty if no SSL is required. This allows different SSL settings for different ports. Note that each SSL requires extra server memory.
External SSL	Indicates that SSL is provided on this port but by external means such as hardware or a proxy SSL server.
Load balancing	Check if the port should be used for load balancing. Keep in mind that load balancing must be enabled in both the slave and the controller server.

16.5 Configuring the Web Server.

The dialog box to configure the web server is opened by selecting the menu item **Server – Configure – Web server** (or **Server – Web server…** in the Editor).



The General page contains the basic settings of the Web Server.

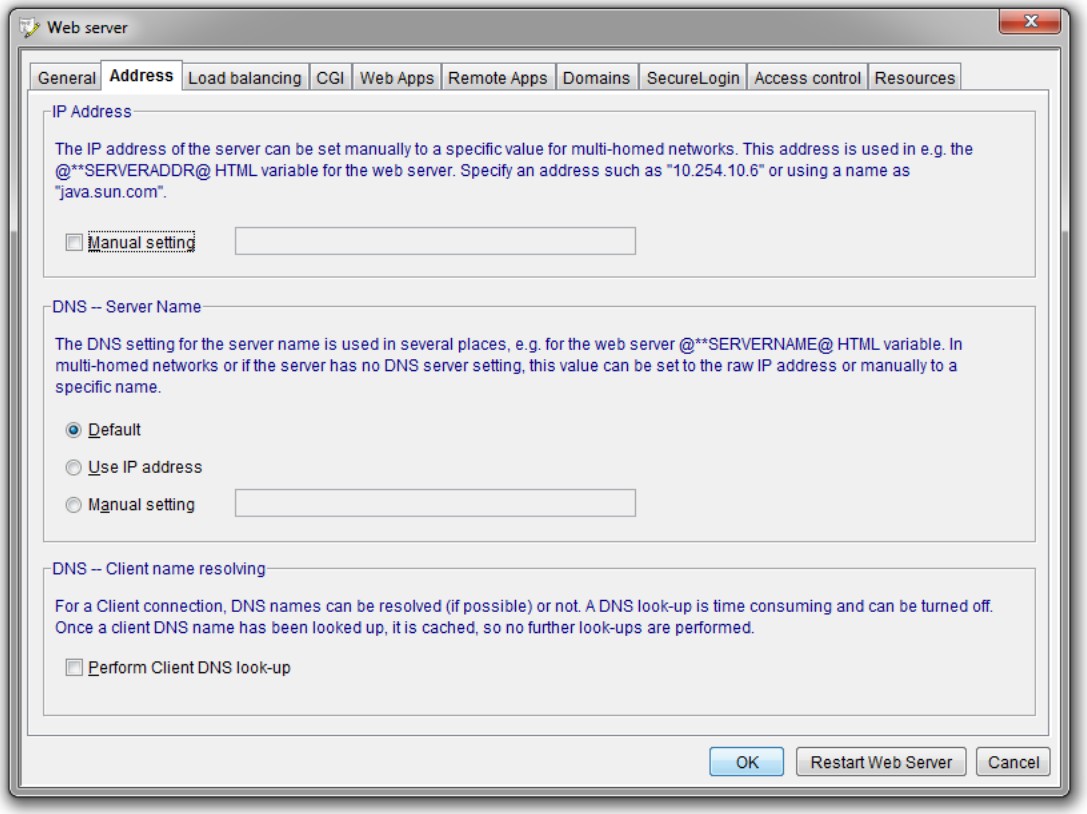
Enable Web Server	Check to enable the web server. Remove the check to run a different web server, e.g. Apache.
Case sensitive file system	Check this option if the file system is case sensitive, e.g. UNIX file system is case sensitive. This prevents unintended duplication of resources in the cache.

Enable HTTP tunneling	This option must be checked to use HTTP tunneling. A web server must also be specified, either by checking the **Enable Web server** option above or by specifying another web server that supports and is configured for HTTP tunneling.
Enable Keep-Alive	This setting is used to control if "Keep-Alive" connection is supported or not (default is *disabled*).
Keep-Alive Time-out	This timeout controls the wait time between a valid HTTP request-reply and the next request from the client using a "Keep-Alive" connection. Once a header is detected, the "headerReadTimeout" value will be used. The setting is quite important because the server might be able to send a big amount of data to a client that is located e.g. on a slow connection to the Internet, causing the next request to be sent only once the client has read the full server reply. This will then cause the server to have to wait a considerable amount of time before the next request is present. The value is in milliseconds (zero=indefinite, default=20000 i.e. 20 seconds).
Header read time-out	This timeout is used when reading the header portion of the HTTP request and the form data in a POST request. The value is in milliseconds (zero=indefinite, default=indefinite).
Log requests	On HTTP requests, all responses will be written to the event log.
Document root	Enter the location of the document root.
Error documents	Enter the location of the error documents.
Default documents	When a user specifies a directory without a specific file, the server will search that directory for the default documents. List the default documents in order of importance. The list should be separated only by blanks.
HTTP server identification	When a web server communicates with a client it sends an identification string indicating what type of server it is. For security reasons this string may be set to another value.
Max simultaneous requests	Set the maximum number of simultaneous requests that will be accepted. Each HTTP request is processed in a separate thread. This parameter is used to avoid server flooding due to e.g. attacks.
Warn simultaneous requests	Set a warning level for simultaneous requests. When this number is exceeded, a warning event will be sent to allow the administrator to prevent bottlenecks from occurring.
HTML cache size	Set the cache size for the HTML cache. This is used for parsing documents and requires about 10 times the amount of memory specified due to pre-parsing. Specify zero to disable the cache.
Resource cache size	Set the resource cache size. This cache contains other resources such as images, etc.
Maximum resource size	Set a limit for the size of individual resources that will be placed in the cache. Resources that exceed this size will not be cached.
Block size	The size of the transport block should be specified in Kbytes. Four Kbytes is a fairly optimal setting in most cases.

Authentication timeout	Set the *nonce* timeout value for Digest Authentication, i.e. the number of minutes that a *nonce* can be reused. If the timeout value is set to 0, a unique *nonce* will be used for each request. However, setting this value too low affects browser response time negatively.

IP Address

Problems with IP address conflict and DNS server name conflict may arise in multi-homed networks. To prevent this, the IP address and server name can be set manually. This setting will then override any other address/server name. You can also choose to use the IP address as DNS server name.

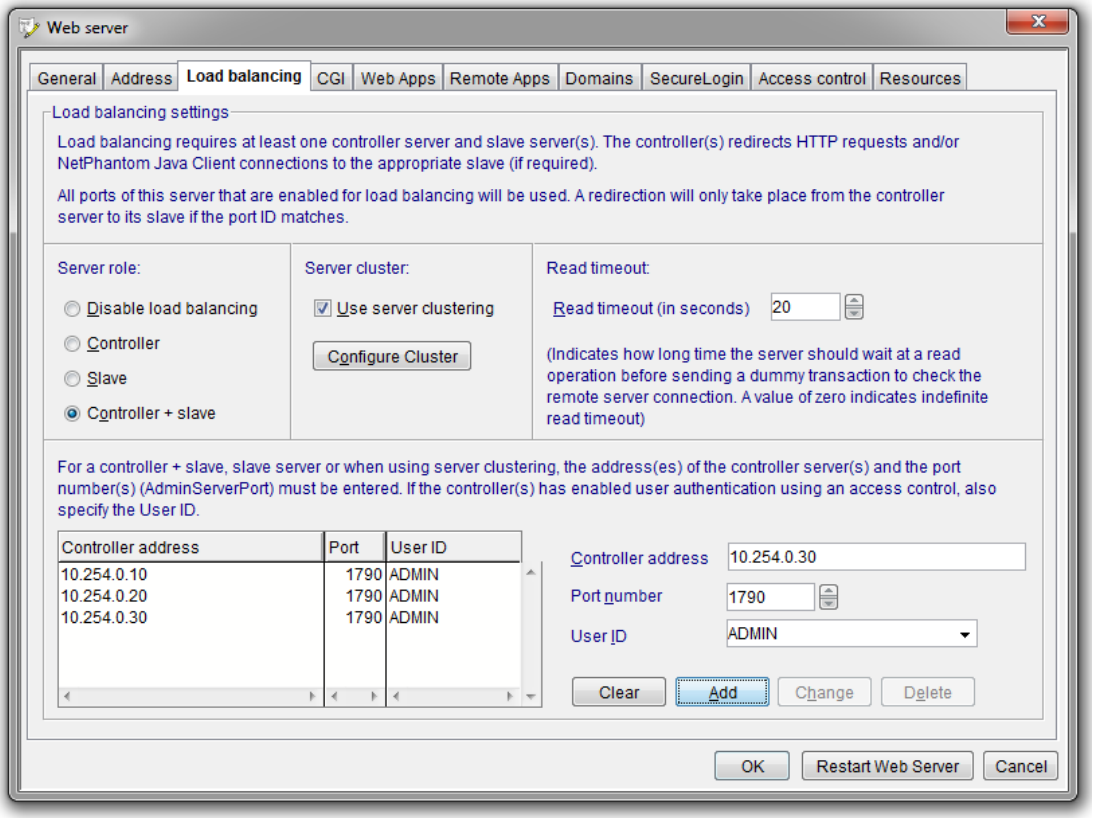


The settings on the Address page are used to prevent address conflicts for the web server.

IP Address-Manual setting	Sets the server's IP address.
DNS – Server Name Default	A DSN look-up will be performed. This can affect performance negatively.
DNS – Server Name Use IP address	If an IP address has been set manually it will be used, otherwise the IP address of the server will be used.
DNS – Server Name Manual setting	If the server's name is set manually this value will be used. Of the three options this results in the fastest performance.
Perform Client DNS look-up	Once a client DNS name has been looked up, it is cached, so no further lookups are performed. By default, DNS look-up is disabled to enhance performance.

Load Balancing

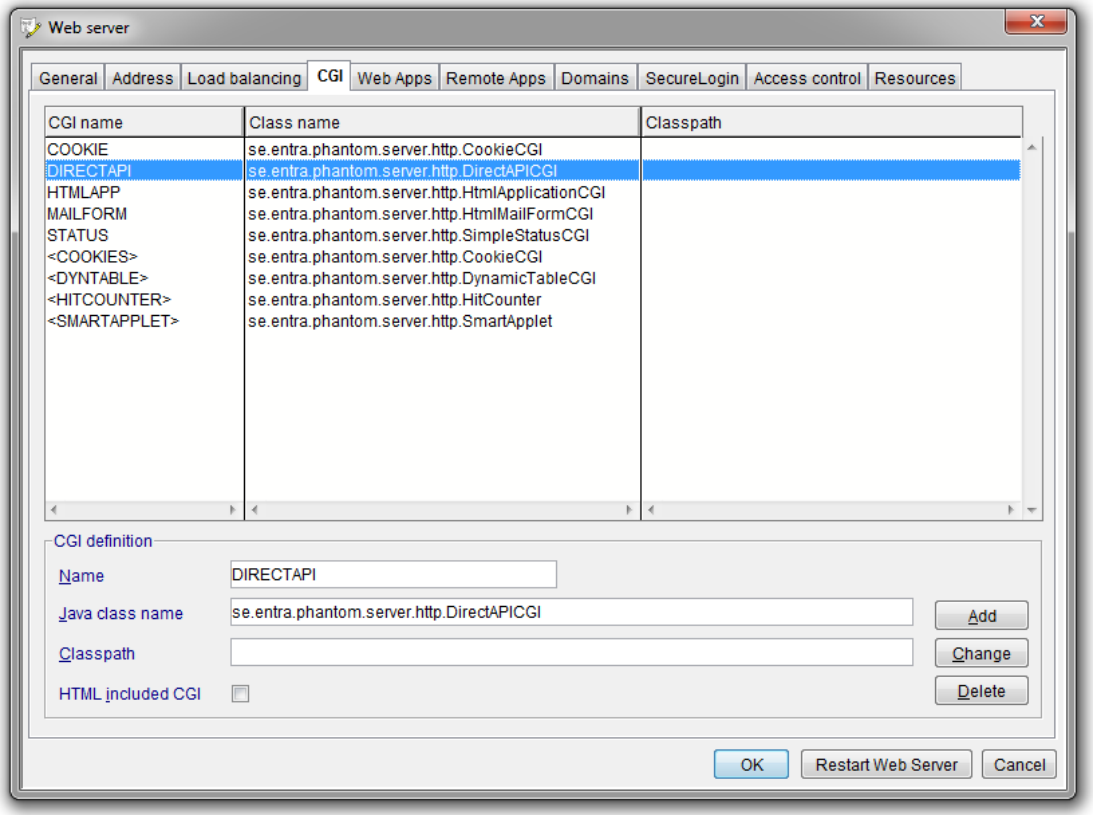
For a detailed description of load balancing see *Chapter 9 NetPhantom Load Balancing*.



The Load balancing page is used to set the server as either a slave or the controller. You can also disable load balancing for the server on this page.

Disabled	Disables load balancing for the server.
Controller	Specifies that the server is the controller server.
Slave	Specifies that the server is a slave.
Controller + slave	Specifies if a server shall be able to switch between the role of controller and slave. This setting can be used, for example, with a NetPhantom Server Farm configuration, see separate documentation.
Use server clustering	Check this option if the NetPhantom Cluster Controller is used to run NetPhantom Servers. See *Chapter NetPhantom Cluster Controller* for more information on how to use and configure it (using the **Configure Cluster** button).
Controller address	This field is only enabled for slave and controller+slave servers (or in connection with the Cluster Controller program). It is used to set the address of the controller server. The address may be an IP address or a server name.
Port number	This field is only enabled for slave servers (or in connection with the Cluster Controller program). It is used to set the administration port number of the controller. In most cases this will be 1790.

Defining a Common Gateway Interface

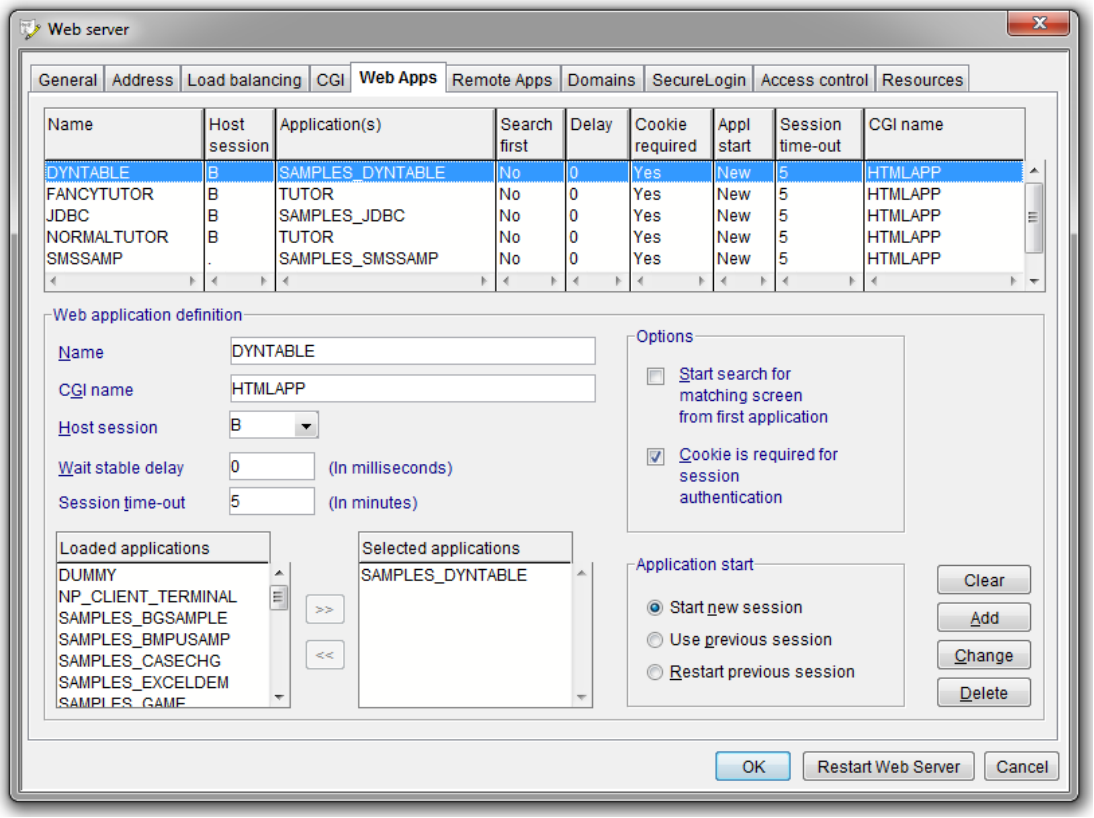


AN HTML included CGI can easily be distinguished from "normal" CGIs, as its name appears in the format <CGINAME> in the CGI Name column.

Name	Enter the name of the CGI.
Java class name	Enter the CGI's Java class name. The Java class name must be in the classpath.
Classpath	The (local) classpath that will be used when running the CGI class. That means that all refeneced classes must be possible to find using this class path for the CGI to run.
HTML included CGI	Any HTML document can contain a reference to an HTML-included Servlet/CGI. The typical usage of Servlets or CGIs is in processing responses of HTML forms, dynamic (content) HTML documents containing dynamic data, etc. Check this box if the CGI is an HTML included CGI.

Defining a Web Application

Web applications are HTML versions of NetPhantom applications. They have the advantage of fast download times as no client applet is downloaded, only the HTML page. It is therefore a good solution for quickly presenting a lightweight version of your legacy systems over the Internet. But keep in mind that HTML does not support the advanced GUI functionality.

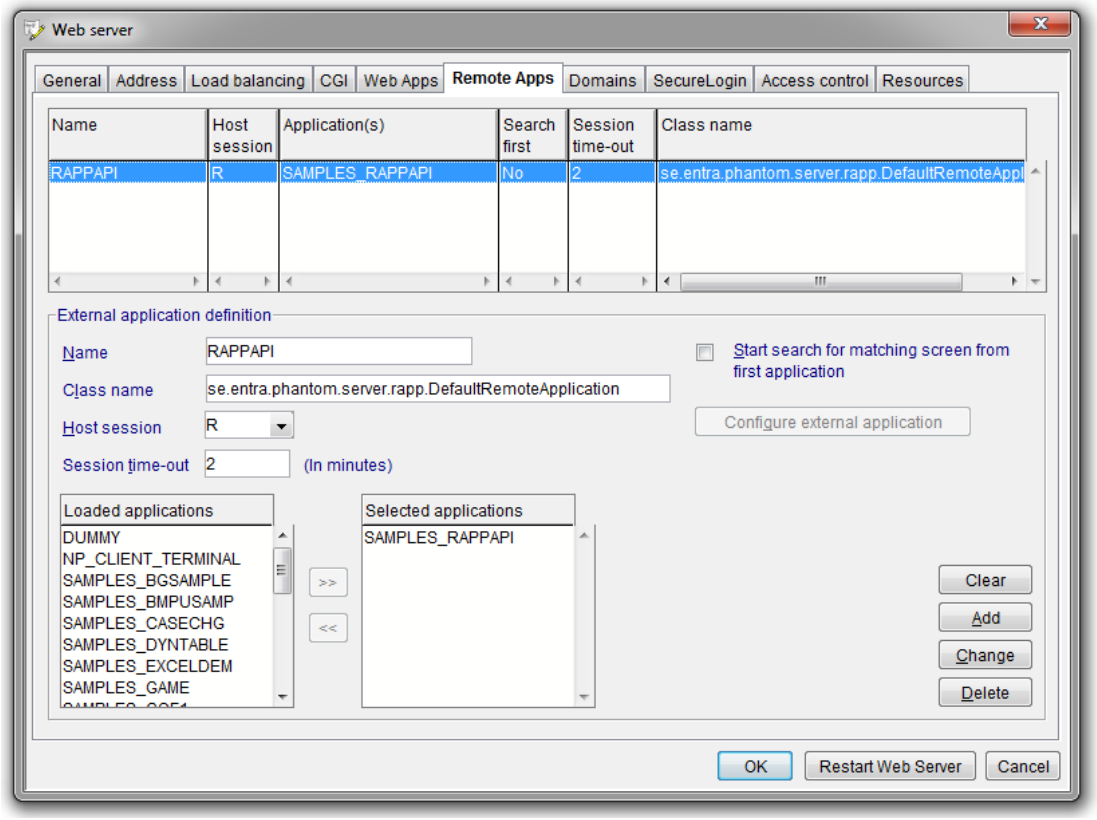


Name	Specify the name of your choice, e.g. `NameOfWebApp`. Keep in mind that this is the same name that will be used in the directory structure, `C:\Phantom\NetPhantom\MyApp\`*`NameofWebApp`*`\html`, as well as the name specified on the **Resources** page as **CGI/Filename**.
CGI name	The CGI setting references the CGI application to be run. In this case HTMLAPP.
Host session	Select the appropriate host session The host session must be defined on the **Host** page of the **Configure server** dialog box.
Wait stable delay	This is an optional waiting period to make sure that the host has delivered all information to the NetPhantom Server before the browser is updated. This value is specified in milliseconds.
Session time-out	The session time-out is the time that the NetPhantom Server will wait for the client to respond before disconnecting from the host. This value is specified in minutes.
Loaded applications	A list of the applications that are loaded on the NetPhantom Server. Select from this list and click the arrow button to add applications to the list of **Selected applications**.
Selected applications	The list of applications that will be used for the Web application.
Start searching for matching screens from first application	This option is related to the use of Merge-on-the-fly and causes NetPhantom to begin searching for a matching screen from the first application in the list of selected applications rather than from the current application.

Cookie is required for session authentication	A cookie (a locally stored information file) is used to identify the client with the server. The server assigns a unique identifier for each client session passed to the browser when the application starts (and only then). This identifier is sent from the browser with each new request (POST and/or GET), thus enabling the server to check the authenticity of the client with the session. **Note:** It is highly recommended that you use this option! It guarantees that another intruding (malicious) client cannot "steal" the session. Combined with the usage of SSL, it then becomes the preferred, most secure client/server communication (that is stateless as HTTP is in its design).
Application start	Options for HTML Web Application that allow you transfer global variables between sessions. **Start new session** Starts a new session with empty global variables. **Use previous session** Transfers all global variables that are not session-specific to the new session. **Restart previous session** Sends the START message to the Application object, which must return 1 to keep the session alive.

Remote Apps

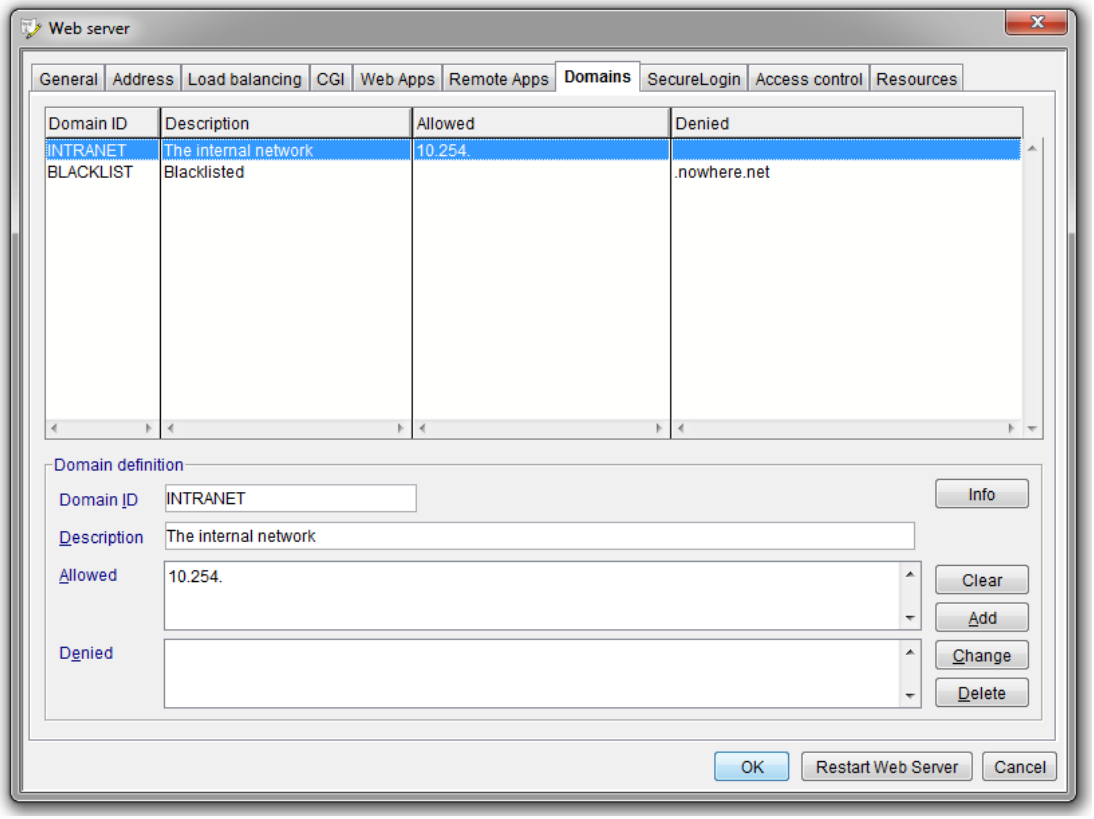
The Remote Apps page is used to define NetPhantom applications that will communicate with external applications that make use of the NetPhantom RAPP API. As with Web applications, before defining the remote application you must define an application and a host session in the **Configure server** dialog. A remote application also requires a **Resource** that points to it. For information on using the RAPP API for NetPhantom Remote applications see the document *RAPP Tutorial - Using the NetPhantom RAPP API* and the RAPP Sample included in a default NetPhantom installation.



Name	The Remote Application must have a unique name. It can be the same as the name of the NetPhantom Application.
Class name	The class name is the name of the class that implements the server side RemoteApplication. The default class is *se.entra.phantom.server.rapp.DefaultRemoteApplication*
Host session	Select a host session for the application. The host session must be defined on the **Host** page of the **Configure server** dialog box.
Session time-out	You may choose to set a timeout for the remote application. This number is in minutes.
Loaded applications	A list of the applications that are loaded on the NetPhantom Server. Select from this list and click the arrow button to add applications to the list of **Selected applications**.
Selected applications	The list of applications that will be used for the remote application.
Start searching for matching screens from first application	This option is related to the use of Merge-on-the-fly and causes NetPhantom to begin searching for a matching screen from the first application in the list of selected applications rather than from the current application.
Configure external application	The functionality for configuration of external applications is not yet implemented.

Specifying Domains

Specify a **Domain ID** and enter a descriptive text for the domain. The **Allowed** and **Denied** entry fields are used to specify the IP addresses or DNS that will be allowed/denied access to the domain.



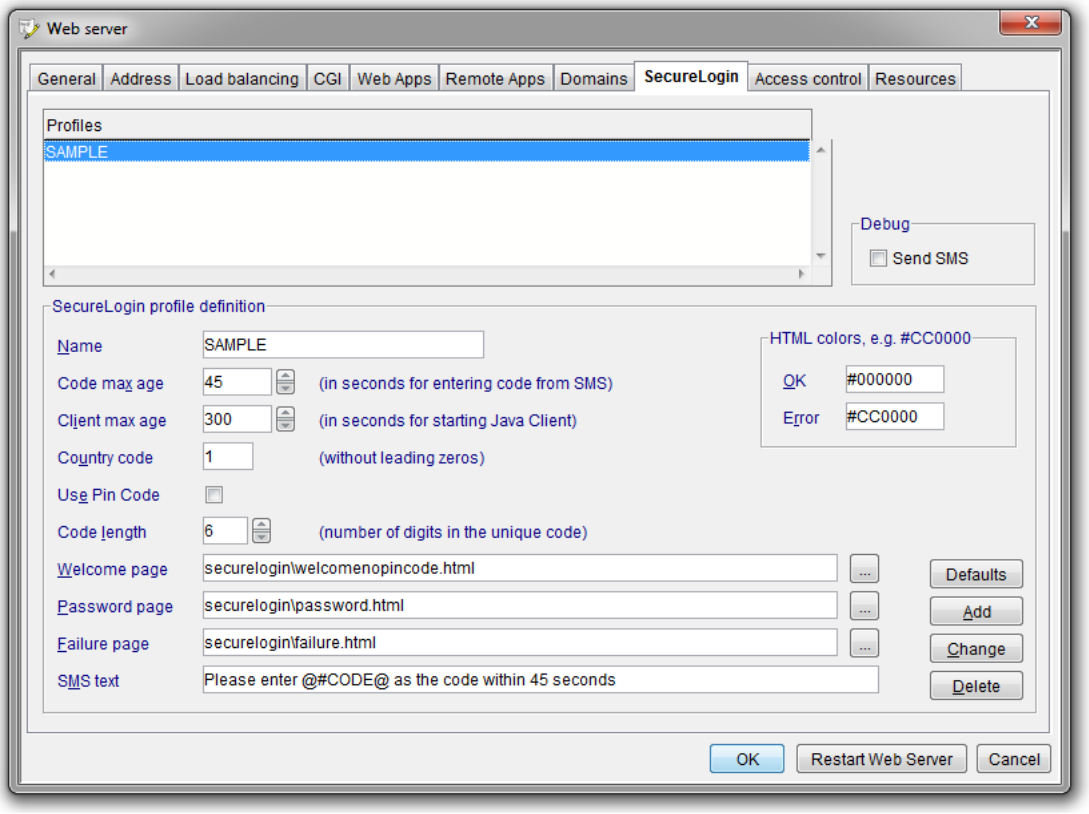
The format for all entries are either sub-forms of an IP address (class A-D), such as "123.234." or "1.2.3.4" (called domain class or an individual IP address) or a DNS name "some.comp.cn" or a sub-form ".comp.cn".

By default, all clients are allowed. Multiple IP addresses (or sub-forms) alternated with DNS names (or sub-forms) can be specified separated by spaces.

Warning: If you enter a DNS name (i.e. a "non-pure IP address" containing other characters than digits or "dot"), a DNS look-up will be performed on the server.

A DNS name instead of an IP address may affect performance dramatically for the Web Server due to extra TCP/IP traffic but has little impact on a NetPhantom Java Client (a web browser causes this check to be performed for each HTML page resource as opposed to the NetPhantom Java Client doing it once the session is established with the server).

Secure Login



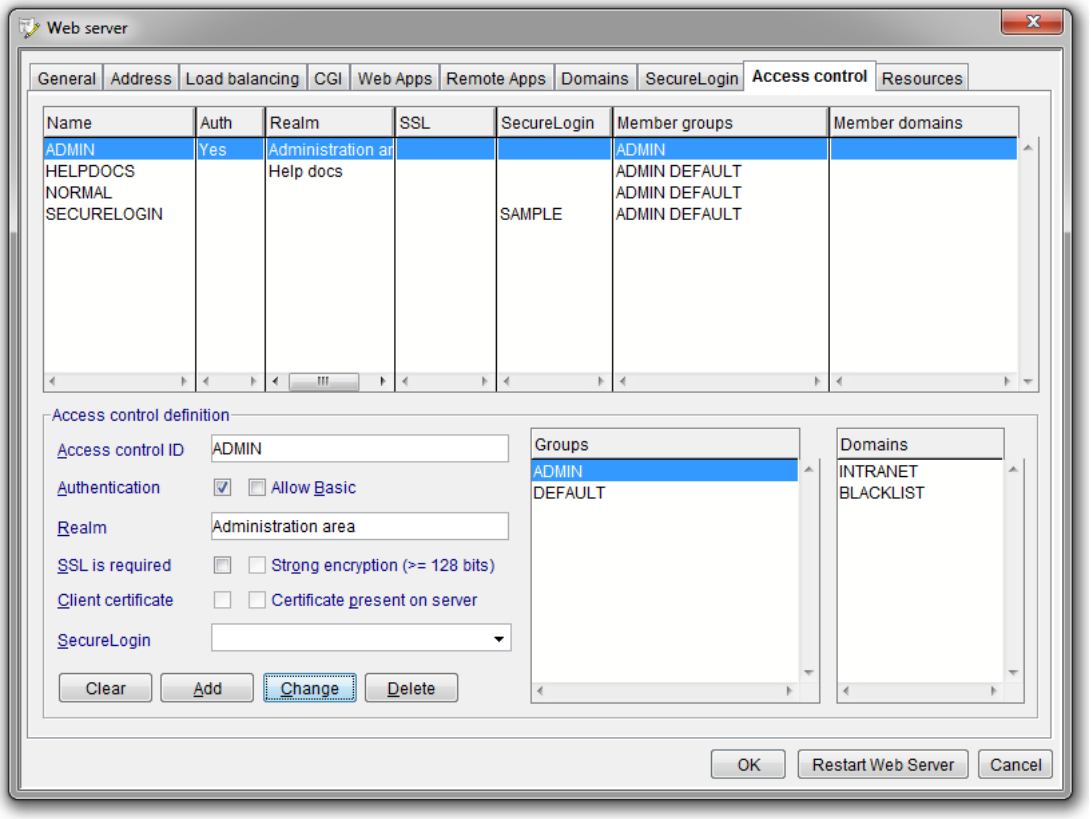
The SecureLogin page is used to define profiles.

Profiles (list box)	This is a list of defined Secure Login profiles. To change an existing profile, select it in the list and edit the values in the **SecureLogin profile definition** group box. Then click the **Change** button.
Debug **Send SMS**	Check the Send SMS checkbox if you want Secure Login to send the unique code even when the application is being debugged. If this option is not checked, the message will be displayed on the Secure Login HTML page.
Name	Specify the name of the Secure Login profile.
Code max age	Set the maximum age of a session in seconds. If this value is exceeded, the unique code is invalidated.
Client max age	Set the maximum age that a session can have while waiting for a Java Client to start the NetPhantom Application (default is 5 minutes).
Country code	Enter the telephone country code without leading zeros.
Use Pin code	Check this option if a pin code should be required in addition to the unique code.
Code length	Select the number of digits that the unique code should contain.
Welcome page	The welcome page (redisplayed for incomplete information).
Password page	The password page for the code from the SMS.
Failure page	The general failure page (cannot send SMS, wrong code, etc).

SMS text	Here you can customize the SMS text that is sent together with the unique code. The `@#CODE@` variable with be given the value of the unique code.
HTML colors	Set the colors for OK or error. These colors can be used in HTML documents.

Access Control

The Access control acts as an intermediary filter for resource definitions.



The Access control page defines various access types that can be used for web resources.

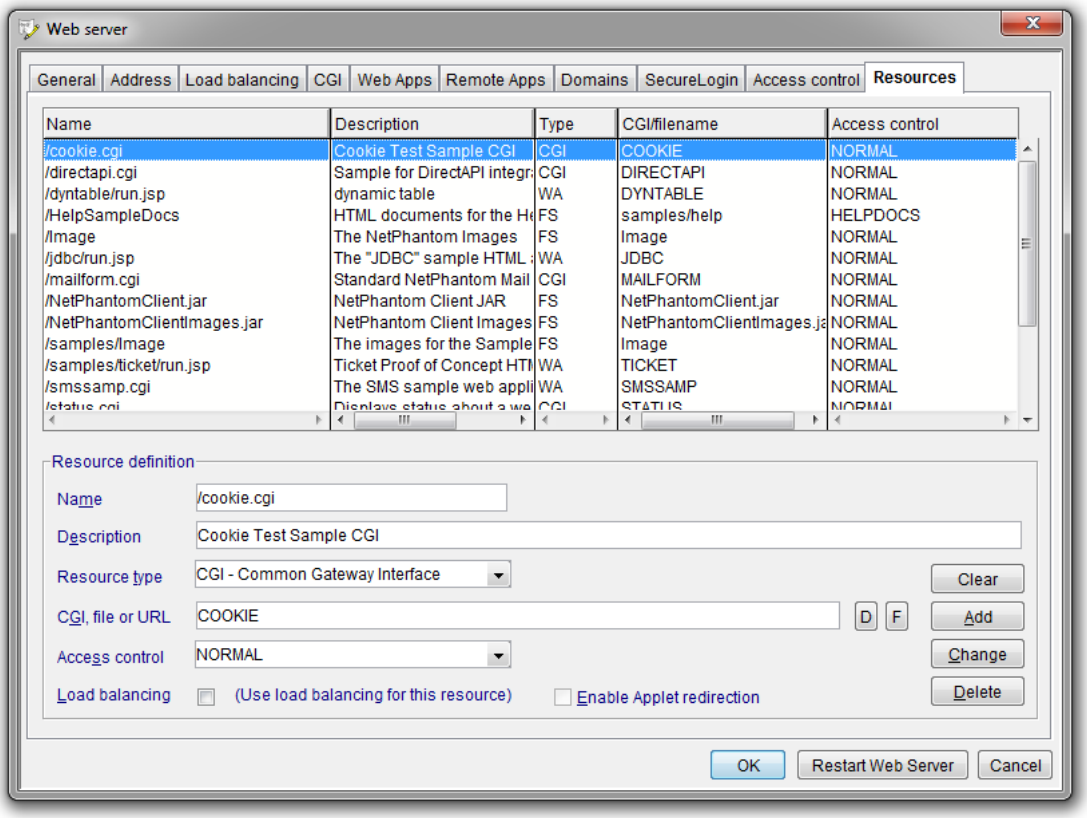
Access control ID	The name of the access control.
Authentication	Check the **Authentication** box to protect the resource from unlimited access. Users who are members of authorized groups will be required to enter a password before gaining access to the resource. NetPhantom supports both Basic and Digest Authentication. This option can be used separately or in conjunction with SSL.
Allow Basic	Check this option if Basic Authentication should be allowed for the resource. This enables browsers using HTTP version 1.0 to send requests. However, Basic Authentication is not considered to be secure. Leave the option unchecked to allow only Digest Authentication.
Realm	This is the string that will be displayed to the users to let them know which password to use when the resource requires user authentication.

SSL is required	SSL is used to protect resources from unlimited access and can be used separately or in conjunction with Authentication.
Strong encryption	If the option **SSL is required** is checked, you can choose to use **Strong encryption** of 128 bits or more. This option is not supported by all browsers.
Client certificate	This option will require a client certificate. If there is no client certificate, the user will not be granted access to the resource.
Certificate present on server	If a client certificate is required, you can specify that it must be located on the server.
Groups	Select the groups that should be authorized to use this resource. If Authentication is used, at least one member group *must* be defined. For **File system** resources, the HTTP reply 403-forbidden is sent to the client if no groups are assigned to the resource.
Domains	Choose the appropriate domain definitions for access control.

Resources

The elements of a resource definition vary to some extent depending on the type of resource. There are nine resource types, which are explained below.

Resource type	Description
FS – File System **FSU** – File System (Unparsed) **CGI** – Common Gateway Interface	These resource types are "pure" web resources such as HTML documents, GIF images, etc. FSU is used when you want to increase performance by specifying a directory that contains HTML pages with no dynamic elements.
CGIA – CGI for Applications **FSA** – File System (Application) **NA** – NetPhantom Application **WA** – Web Application	CGIA is normally a CGI running a session-based application affecting the amount of concurrent HTML users. FSA is typically used to indicate an HTML document that contains the NetPhantom Java Client applet definition. NA is a NetPhantom application. WA is a web application that consists of HTML documents, GIFs, etc.
RD – Redirection	Load balancing does not apply for this resource type.

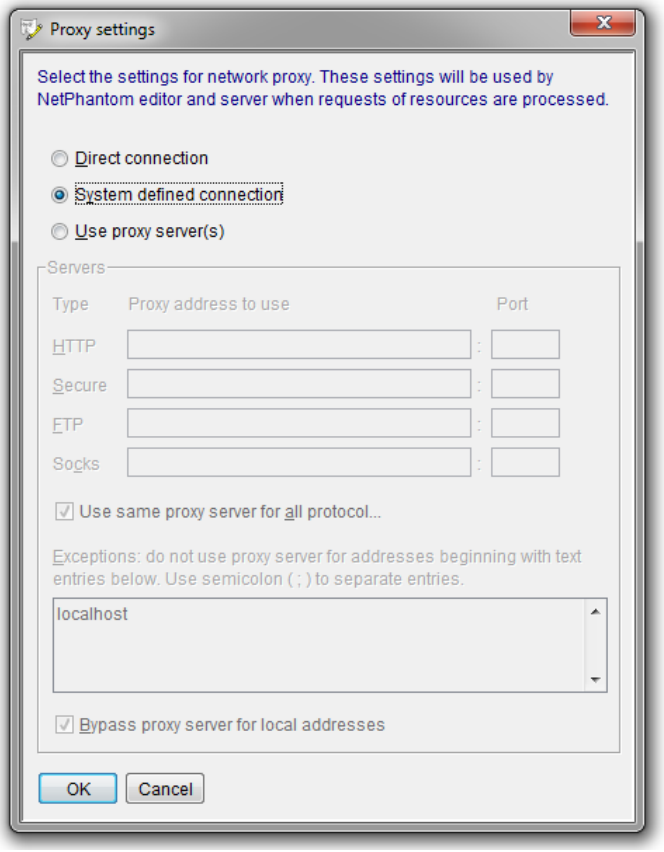


A resource must be defined for all web applications and for file redirection on the web server.

Name	Specify the resource name. **FS** The path to a file or a directory on the server **FSA** The path to a file or a directory on the server **FSU** The path to a file or a directory on the server **NA** The application ID assigned in `server.ini` **WA** The name of the directory that contains the html files **CGI** The CGI name as defined in the CGI page **CGIA** The CGI name as defined in the CGI page **RD** The redirection resource name
Description	Enter a description of the resource.
Resource type	Select the resource type from the combination box.
CGI/filename	Enter the name of the CGI or file. The buttons to the right of the entry field open the Directory list and the File list.
Access control	Select the Access control for the resource definition. The access control allows you to have different definitions of the same resource.
Load balancing	Check this option if the resource should use load balancing. For information on load balancing techniques for various resource types see *Chapter NetPhantom Load Balancing.*

16.6 Proxy Configuration

The dialog box to configure the use of a network proxy for the server is opened by selecting the menu item **Server – Configure – Proxy settings…**(or **Server – Proxy settings…** in the NetPhantom Editor).



Type	The type of proxy to use: - Direct connection (i.e. use no proxy) - System defined (use the system wide proxy definition) - Proxy servers (specify explicitly how proxies should be used for this application. See below)
Protocol	Specify per protocol which proxy server to use. The following protocols are available: - HTTP - Secure (HTTPS - SSL protected) - FTP (File Transport Protocol) - SOCKS (direct socket communication)
Use same proxy server	Use the proxy server specified for HTTP for all protocols.
Exceptions	Do not use proxy communication for connections with the (semicolon separated) list of addresses.
Bypass	Do not use proxy communication for local connections.

16.7 SSL Configuration

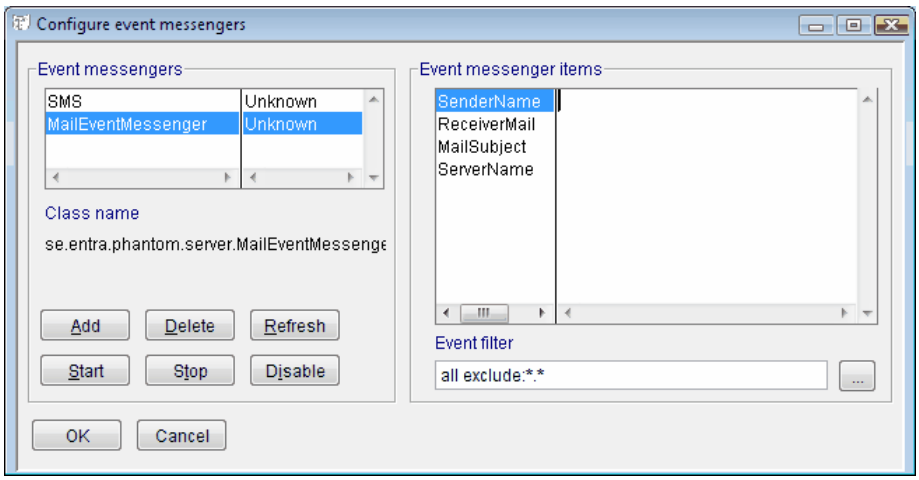
The SSL configuration panel is used to set the preferences for the Secure Socket Layer between the Server and the Client. This configuration tool allows the administrator to set the Cipher Suites, to add Certificates, to add Identities, and to set the connection parameters. For a detailed discussion of NetPhantom and SSL see *Chapter SSL – Secure Socket Layer*.

16.8 Creating Client Certificates

Client certificates are required if you intend to use SSL. See *Chapter Client Certificates* for a complete discussion of client certificates.

16.9 Event Messengers

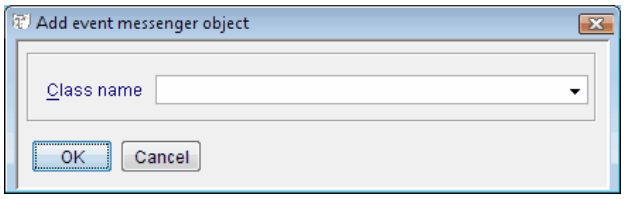
The **Configure event messengers**' dialog is used to change the settings for existing event messenger as well as to add new event messengers. NetPhantom includes two default event messengers: SMS and MailEventMessenger. This function is not available in the NetPhantom Editor.



The Configure event messengers' dialog is used to specify the items contents for the SMS and Mail classes.

Event messengers	Contains a list of all sections in the `EventMessengers` section of the server.ini file.
Event messenger items	When you select an event messenger in the first list, the items specific to that event messenger are displayed in this list. Items may be edited in the second column, but no items may be added to or deleted from the list.
Class name	The **Class name** field displays the class name for the selected event messenger. This field can be edited. The class name specified here must implement the interface se.entra.phantom.server.event.EventInterface or se.entra.phantom.server.event.EventInterface2.
Event filter	The **Event filter** field displays events that are included and excluded for the selected event messenger. This string can be edited by hand or via the **Browse** button which opens the **Add/edit event filter** dialog box.
Add/Delete	These buttons are used to add and remove event messengers.
Start/Stop/Disable	These buttons are used to set the state of the event messenger.

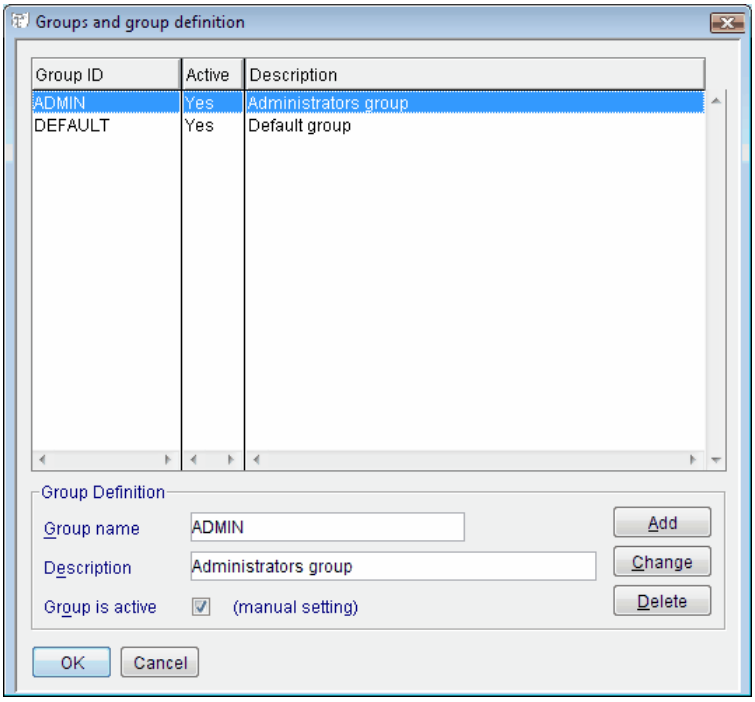
You can add your own event messengers by clicking the **Add** button. The **Add event messenger object** dialog box is displayed.



Enter the class name for your new event messenger and click **OK**. You can now select it in the **Event messengers** list to edit the **Event messenger items**.

16.10 Managing Groups

Groups are used to designate which users are authorized to use resources. It is an important part of the authentication process.



*To manage group definitions, select the menu item **Server – Manage groups**.*

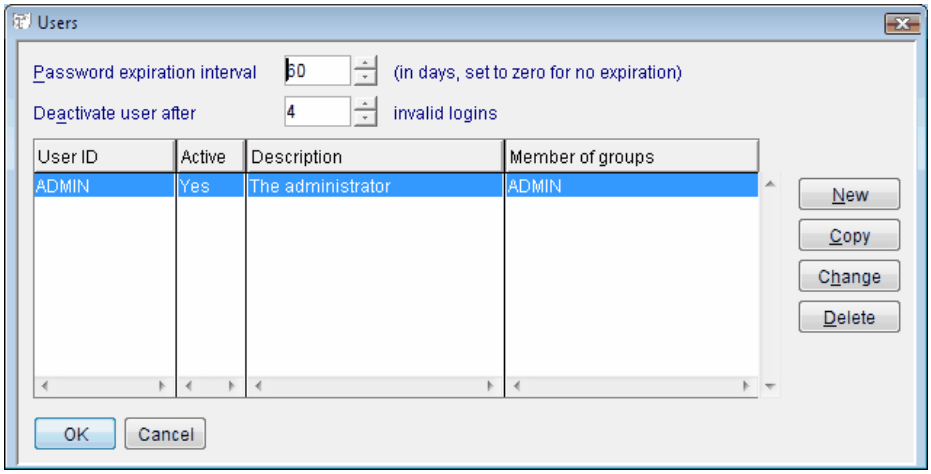
To create a group, enter a **Group name** that will be the group ID and a **Description** that indicates the type of users that will be members of this group. Check the **Group is active** checkbox if the group should be activated. Click the **New** button and the group will be added to the list.

The **Change** button is used to edit the selected group definition, for example to deactivate an active group. The **Delete** button removes the selected group from the list.

The dialog box to configure groups is opened by selecting the menu item **Server – Manage groups** (or **Server – Advanced - Groups…** in the Editor).

16.11 Managing Users

Password Expiration Interval



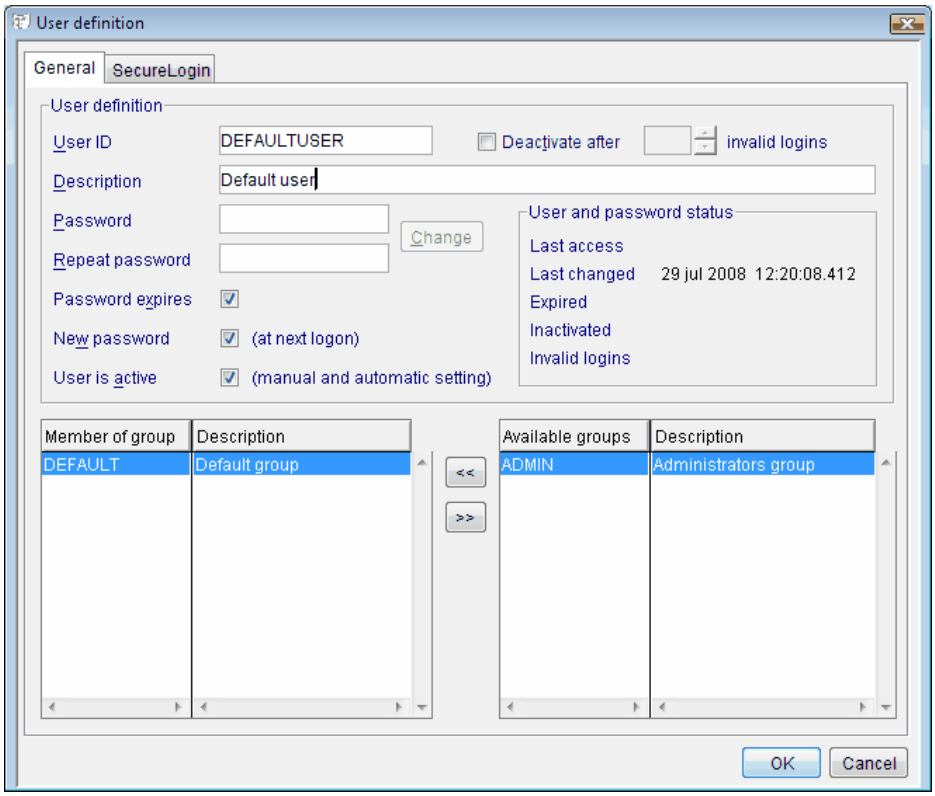
*The **Users** dialog box contains the list of defined user IDs.*

The **Password expiration interval** is only used for user authentication of NetPhantom applications, not for browser user authentication**.** The **Deactivate user after *NN* invalid logins** is used to disable a user profile from login (can be overridden for specific users, see below).

The dialog box to configure groups is opened by selecting the menu item **Server – Manage users** (or **Server – Advanced - Users…** in the Editor).

Creating a New User

The **New** button is used to create a new user definition.



The User definition can be used to create a new user or to reset the user password.

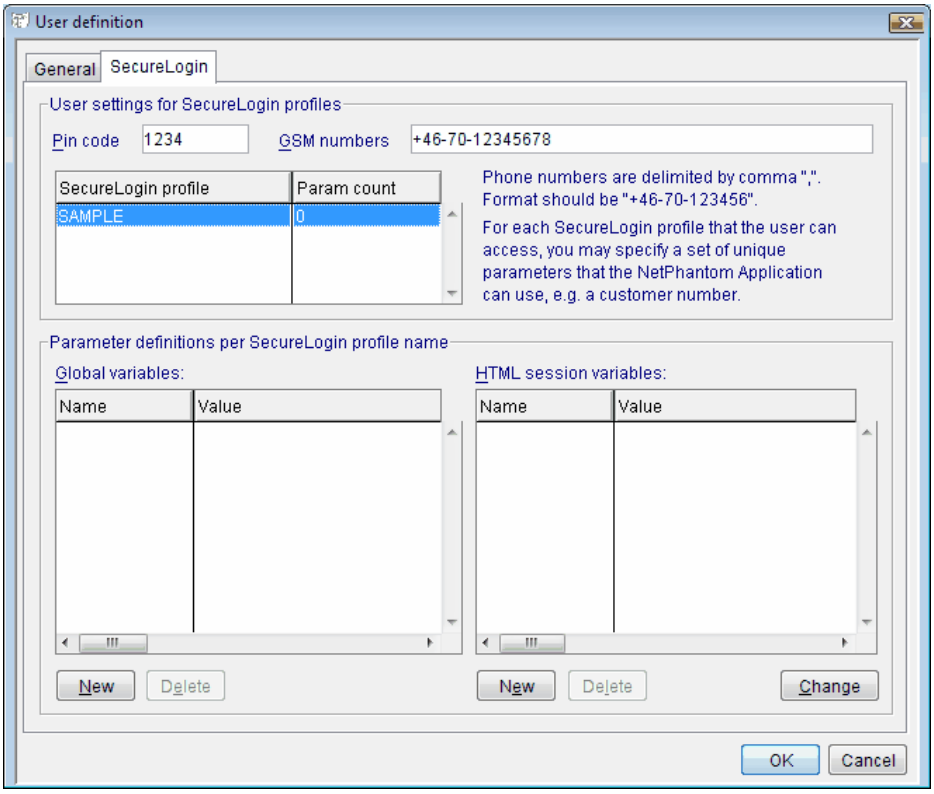
User ID	Enter a unique ID for the user. The number of characters may not exceed 8.
Description	Fill in a description of the user, for example the user's full name.
Deactivate after *NN* logins	Overrides the setting for *Manage Users*. Setting this value to *zero* indicates that the user profile will never become deactivated due to invalid logins.
Password	Enter the user's password.
Repeat password	Re-enter the password to verify.
Change	The **Change** button is enabled when the **User definition** has been opened via the **Change** button in the **User** dialog box. Click this button to enable the **Password** and **Repeat password** fields. The user password can now be changed.
Password expires	Check this option if the user password should expire periodically. See *Password Expiration Interval* above.
New password	If this option is checked, the user will be prompted to change the password at the next logon. See *Password Expiration Interval* above.
User is active	Activates the user.
Password status	Here you can see the history of the password.
Group list	Select the appropriate groups from the **Available** list and use the arrow button to move them to the **Member of** list. These are the groups to which the user will belong.

Changing or Copying a User Definition

The **Change** button is used when a detail of a user definition needs to be changed. For example, this could be deactivation of the user, a change in the password, or a change in the groups to which the user belongs. To activate the password fields, click the **Change** button.

Often there are large groups of users that share the same definition except for the User ID and Password. The **Copy** button facilitates the creation of new Users. Select the User upon which the new definition will be based and click the **Copy** button. The **User definition** dialog is opened containing the same information as the selected definition except for the User ID and Password fields, which are empty.

SecureLogin Definition for a User



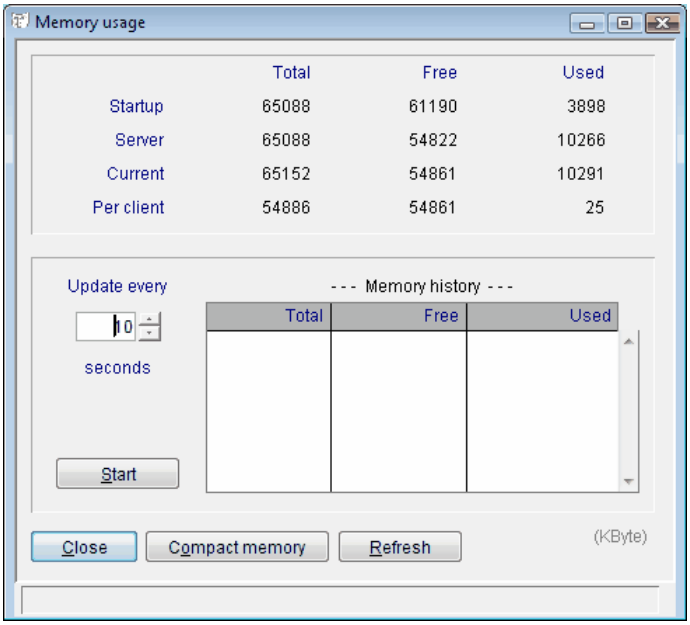
The SecureLogin Definition for a user is specified in the second notebook page in the User Definition dialog box.

Pin code	Specifies the pin code (can be alphanumeric but in upper case). This Pin code is used by NetPhantom SecureLogin depending on the settings for the protected web server resource.
GSM numbers	Specify all numbers that are valid for this user separated by a comma. The numbers should be prefixed by the international code (+) followed by the country code, etc.
Parameter definitions per SecureLogin profile name	When a SecureLogin application starts, **Global variables** can be defined for this user. The REXX code then refers to these variables as *GV_nnn*. If the SecureLogin application is of HTML type, **HTML session variables** may also be defined here. These can be used in HTML code (e.g. <#HV_*nnn*>). See chapters ***Error! Reference source not found.*** and *Variables in HTML Documents* for more information.

For more information on how to configure NetPhantom SecureLogin, see *Chapter NetPhantom SecureLogin*.

16.12 Server Memory Usage

The server memory usage is displayed by selecting the menu item **Server – Memory usage**. As the amount of available, free, or used memory cannot be exactly determined in Java (especially not with different Java Virtual Machines), all these three memory values are displayed in the window.



The memory usage displays total memory for the Java Virtual Machine, the reported free memory and the difference – the used memory. All figures are in Kilobytes.

The Java VM garbage collector may be invoked with the **Compact memory** button. The number of bytes freed is displayed in the status bar. Note that this value may be negative if the garbage collection finishes, and other client sessions consume memory at the same time.

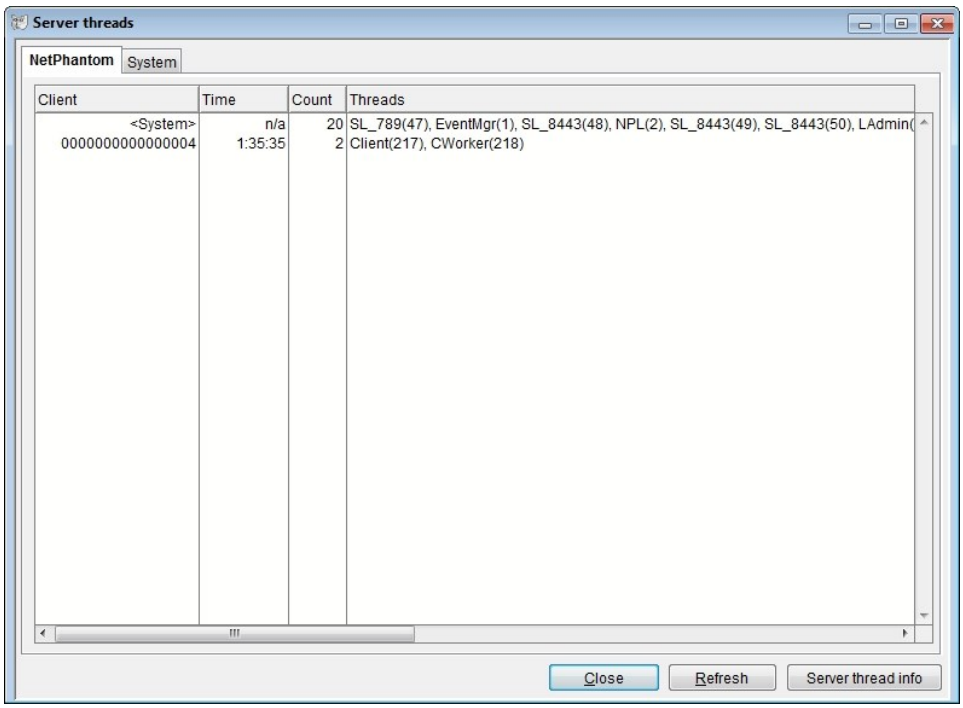
The **Startup** memory amount is the amount reported when the first server class is loaded. When the server is in ready state, the **Server** memory is read. The **Per client** amount is the memory usage for all clients when the used server memory is removed. The number of users then divides the amount. For very few users, this amount is high, but as soon as more than 20 users are connected, the amount of memory drops to 300-400 KB.

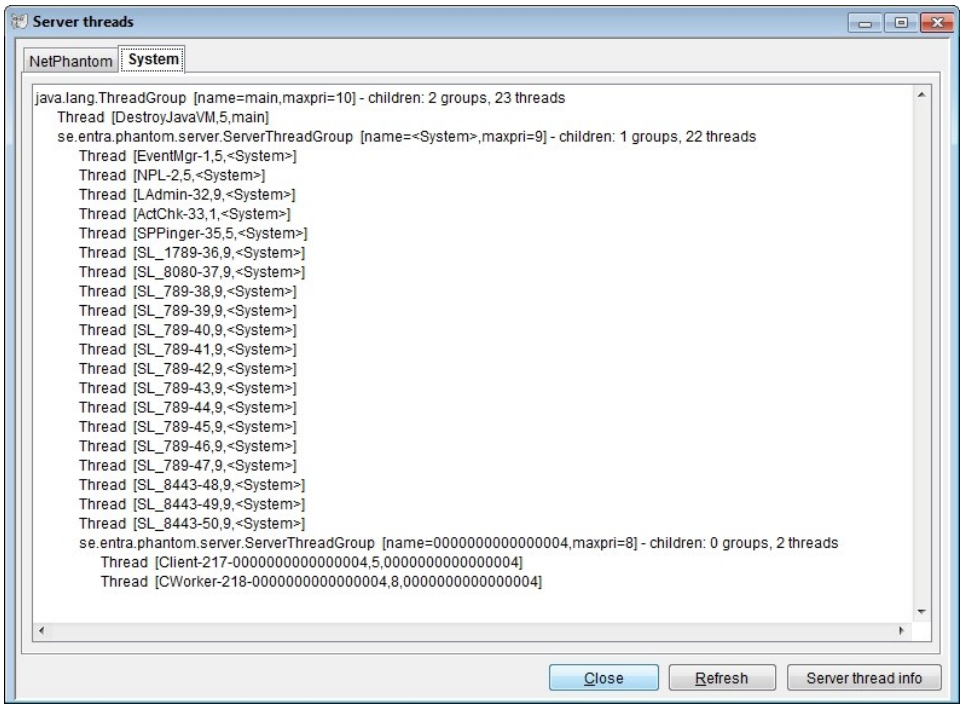
The memory may be refreshed automatically with a certain interval. Remember that this requires server CPU processing; so, don't use very short intervals!

The dialog box to access server memory monitoring is opened by selecting the menu item **Server – Memory usage** (or **Server – Advanced - Memory…** in the NetPhantom Editor).

16.13 Thread Usage

The Server thread dialog lists each client and the number of threads it is running. The Threads column displays the name of the classes running the threads.





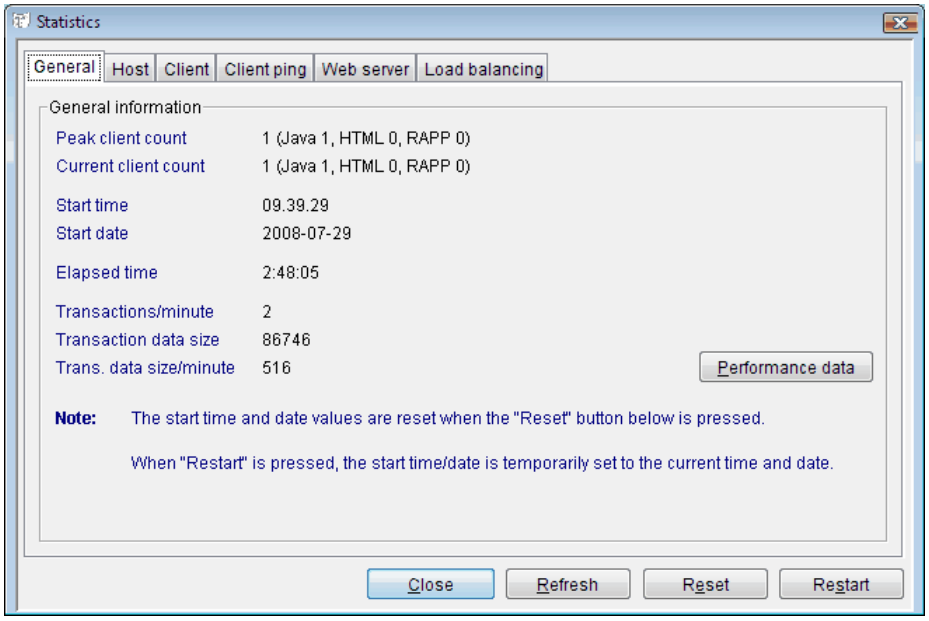
The dialog box to access the thread monitoring function is opened by selecting the menu item **Server – Thread usage** (or **Server – Advanced - Threads…** in the NetPhantom Editor).

Detailed information about the threads can be fetched through the interface. The collected information includes lock/synchronize/running states and stack traces.

16.14 Server Statistics

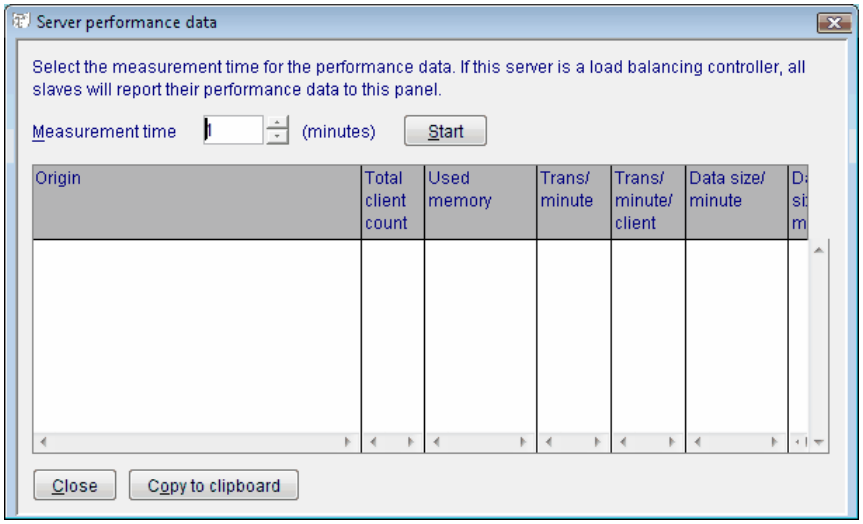
The dialog box to access server statistics is opened by selecting the menu item **Server – Statistics** (or **Server – Advanced – Statistics…** in the NetPhantom Editor).

The server statistics show the peak value for the number of concurrent users since the server was started or the statistics reset using the **Reset** button.

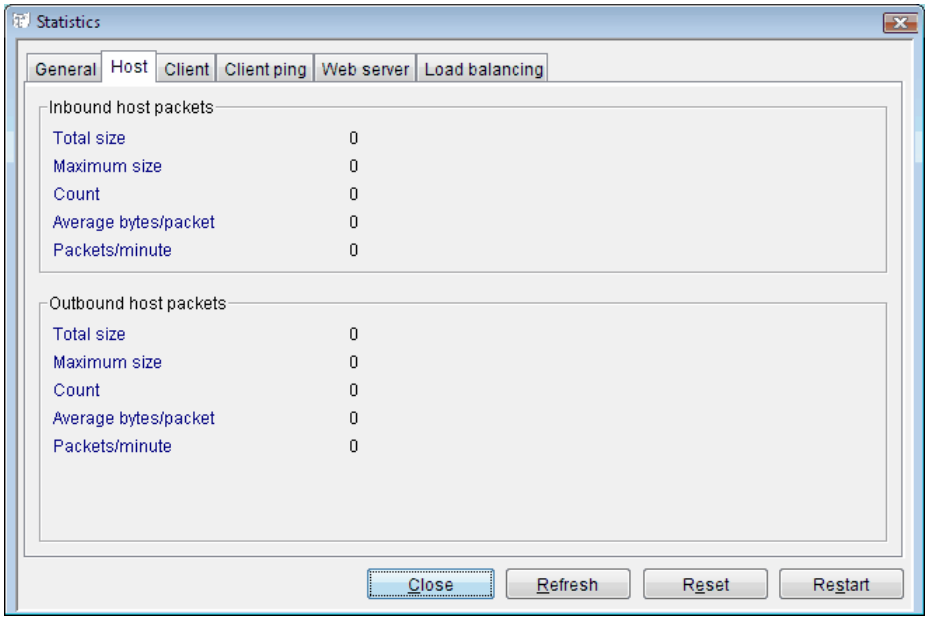


The host packets and client transactions can be used to monitor the network traffic to and from the NetPhantom Server. All figures relating to size are in bytes.

The **Performance data** button activates the **Server performance data** dialog.

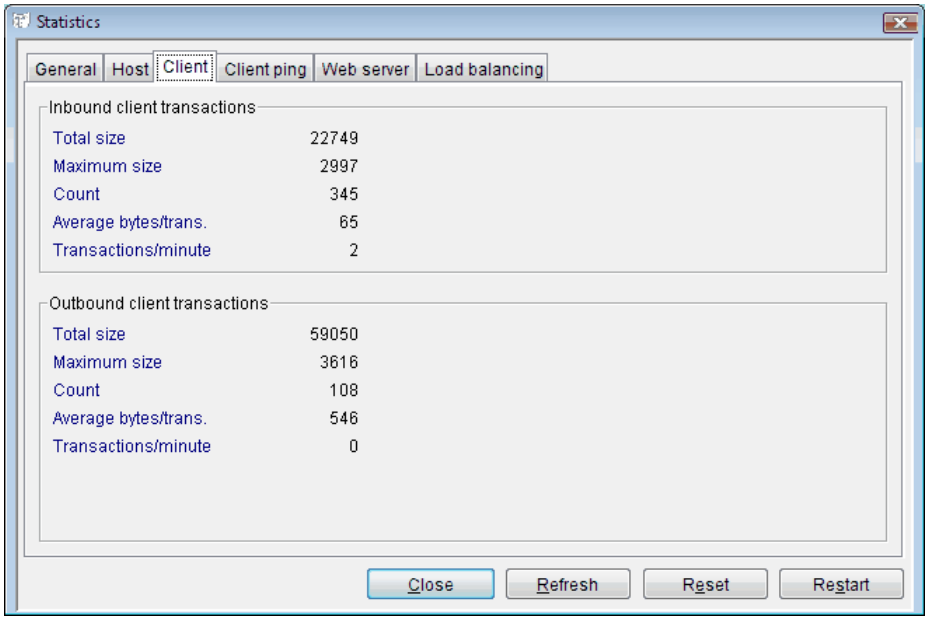


Host Statistics



The host packets are 3270 or 5250 data streams between all the host sessions and the NetPhantom server. Inbound host packets are data stream packets received *from* the host while the outbound host packets are *sent* from the server.

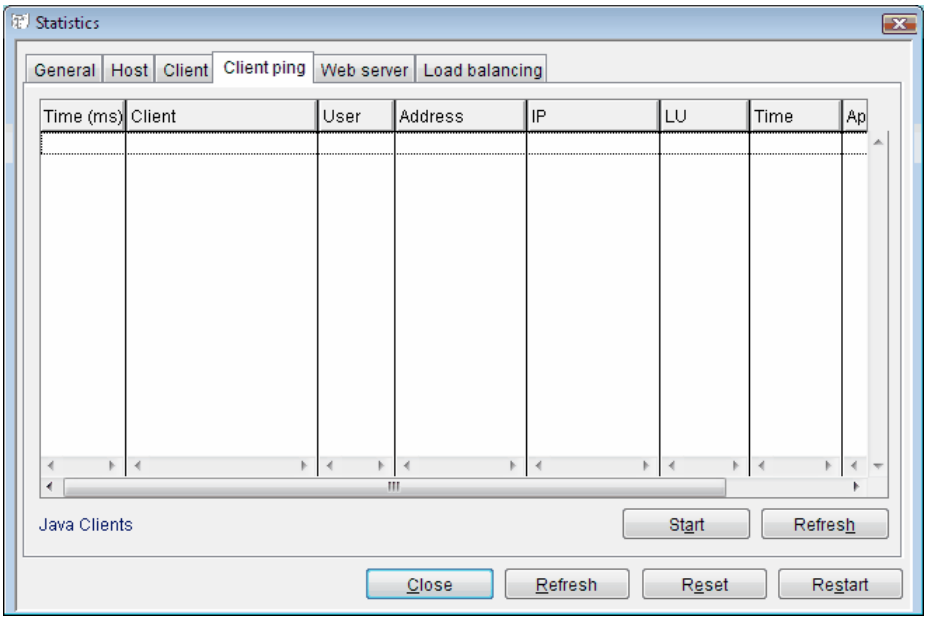
Client Statistics



The client transactions are the data sent between the NetPhantom Server and all the clients. Inbound transactions come *from* clients; outbound transactions are sent *to* clients.

Client Ping

The client ping tool in the NetPhantom Administration Program is used to test the response time of clients connected to the server.



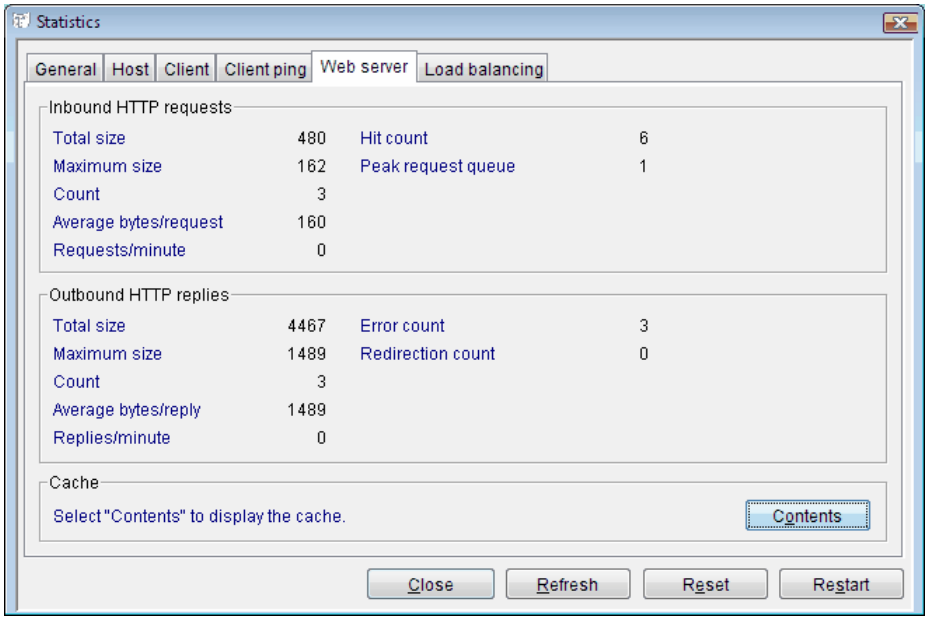
When the **Start** button is pressed, all the Java clients get a queue request in the server. When this request is processed by the client session queue thread a "ping" transaction is sent to the client to measure the server response. The client will respond with another "ping" to the server. When the server gets the client "ping" request, it responds back to the client by another transaction. The client measures the difference in time (in milliseconds) between (before) sending the ping to the server and (after) reception of the server "ping reply". This *time measurement* of "server responsiveness" is then sent back to the server. The **Refresh** button in this notebook page is used to fill in responses sent from all active Java Clients into the first column, initially containing the text "-".

It may seem complicated to do it in this manner, but a client could be modally waiting for a message box, thus not processing incoming transactions. These clients would only reply to the server once the message box is removed on the client side. However, there might also be a very small window in time between ping request/reply where the client is requested to display a message box. This would cause the client to include the user to click in the message box into the ping reply time, thus making it complete in error.

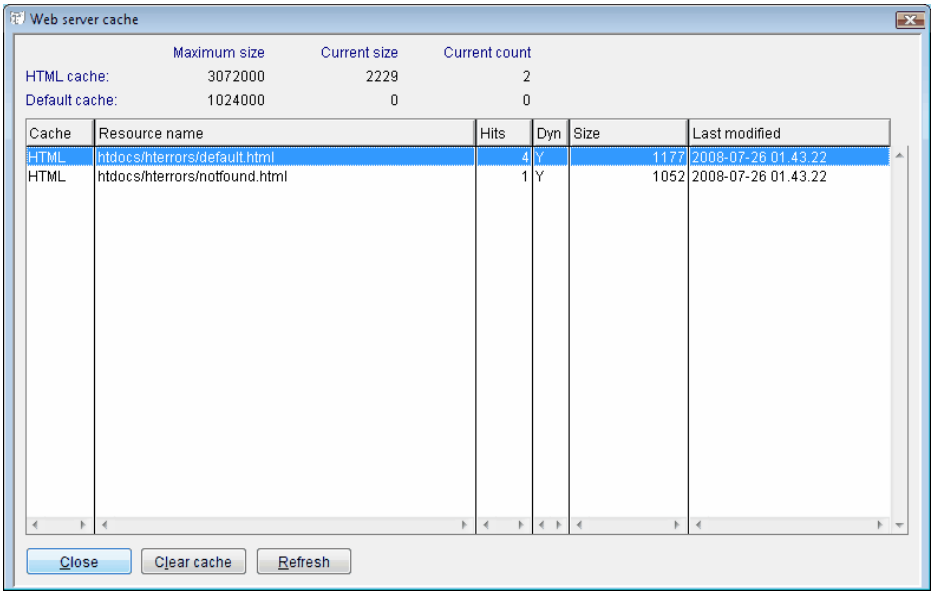
Also note that these measurement values will be very spread out if using a mix of HTTPS connections to Internet users and some "in-house" users with fast connections using HTTP.

These values are intended to be used along with *StressNoGUI* as they can be copied into a spreadsheet for performing calculations. See also *The NetPhantom Stress Tools*.

Web Server



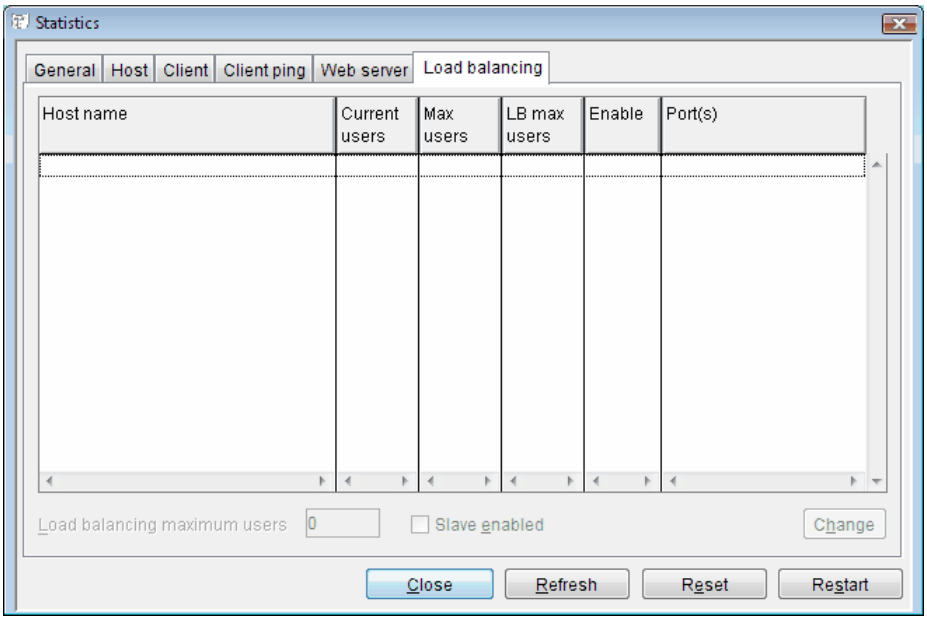
Click the **Contents** button on the Web server page to view the contents of the web server cache. The contents of both the HTML cache and the default cache are displayed here.



The web server cache can be cleared by clicking the Clear contents button.

Load Balancing

The **Load-balancing** page lists each port on each server, the current number of users and the maximum number of users.



16.15 Server Shutdown

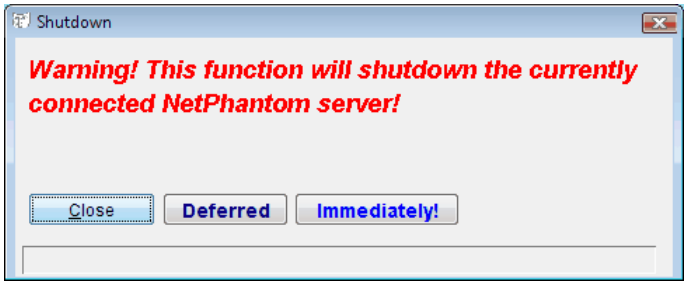
Use the menu item **Server – Shutdown** to stop the NetPhantom Server (this function is not available in the NetPhantom Editor).

*Before the server shutdown be sure to let all users know this by using the **Broadcast message** function and reject new client requests for connection.*

Server shutdown may be immediate or deferred. Selecting the deferred shutdown will cause the server to exit when all users are disconnected. Remember that the Server Administration program is connected users for the server. While the shutdown is in progress, no new client connections are allowed.

16.16 Server Restart

Use the menu item **Server – Restart** (in both the normal server administration interface and in the NetPhantom Editor) to restart the NetPhantom Server.



*Before the server is restarted be sure to inform all users using the **Broadcast message** function.*

Server restart may be immediate or deferred. Selecting the deferred restart will cause the server to restart when all users are disconnected. Remember that the Server Administration program is connected users for the server. While restarting is in progress, no new client connections are allowed.

Soft Restart

The soft restart of the server causes the server to restart without closing the connection to the current Server Administration program.

> **Note:** The best way to restart the server is using the *Hard Java VM Restart*.

A soft restart does not:

- reload the server text file (`server.phm` or related locate files),
- create another event manager, so all event manager settings change since restart are ignored,
- create new connection listeners, so changes to the client port, the client queue size, etc, are ignored,
- create a new window when the server runs in GUI mode (`serverGUI=1`).
- reload class files that are already loaded (in the Java ClassLoader). This does not apply to REXX-converted programs used with NetPhantom applications.

Hard Restart

The hard restart closes all client connections including the current Server Administration program. Once the server is stopped, a full garbage collection is performed in order to free unallocated memory back to the operating system. After this, the server is restarted.

> **Note:** The best way to restart the server is using the *Hard Java VM Restart*.

Below is a list of things that are not handled in the *Hard Restart*. If these things still need to be handled, use the *Hard Java VM Restart*.

A hard restart does not:

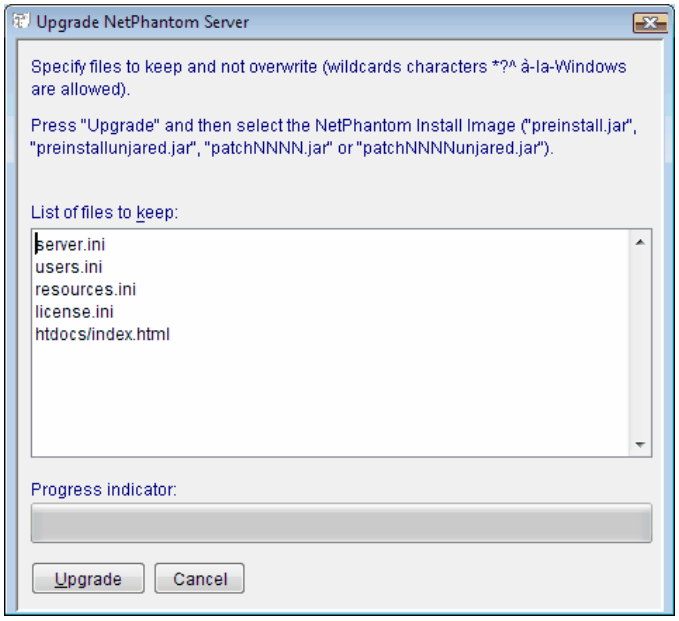
- reload the server text file (`server.phm` or related locate files),
- create another event manager, so all event manager settings change since restart are ignored,
- create new connection listeners, so changes to the client port, the client queue size, etc, are ignored,
- reload class files that are already loaded (in the Java ClassLoader). This does not apply to REXX-converted programs used with NetPhantom applications.

Hard Java VM Restart

This is the preferred way to restart the server. It has the same effect as if the server was shut down and then restarted with a new Java Virtual Machine. However, this requires external program control to check for the server return code.

16.17 Upgrade Server

The menu item **Upgrade server** facilitates the installation of NetPhantom patches.



The **Upgrade NetPhantom Server** dialog box allows you to specify a list of files that should *not* be overwritten when the new files/patches are installed.

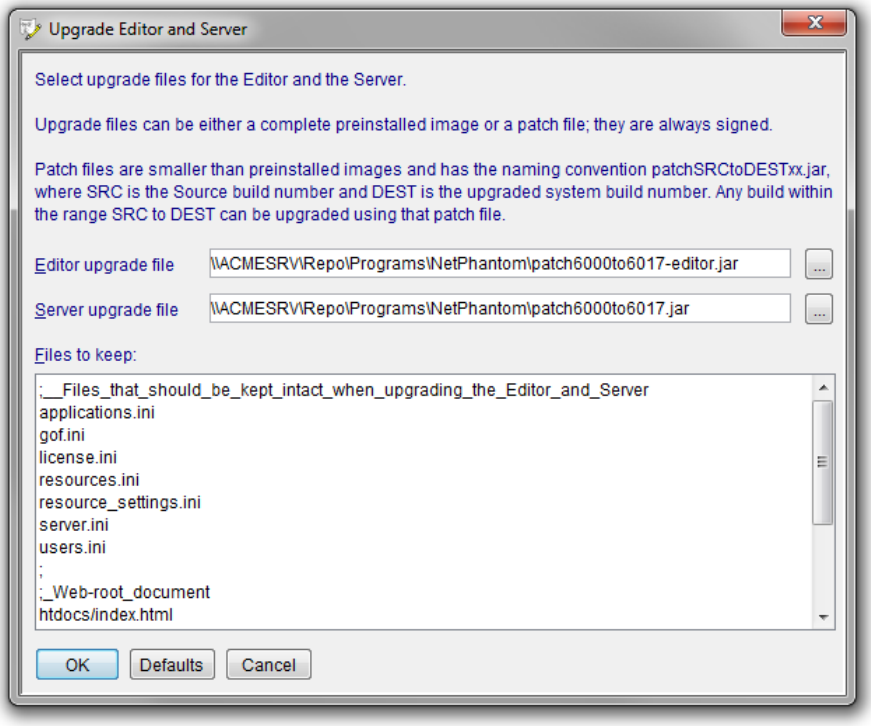
Click the **Upgrade** button and select the desired .jar file.

When the upgrade transfer is complete, a message will be displayed informing you of this and reminding you that the server must be restarted. This can be done via the menu item Restart. See *Server Restart* above.

> **Note:** After upgrade of a 'localhost' use the **Server – Restart – Deferred** option unless you have selected to do so by the message boxes presented.

16.18 Upgrade Editor and Server

In the Editor, a general upgrade facility is available under **Help – Upgrade Editor and Server …** This facilitates both Editor and Server upgrades.



The **Upgrade Editor and Server** dialog box allows you to specify a list of files that should *not* be overwritten when the new files/patches are installed. The list supports wildcard specification (? * ^).

Click the respective file selection button to select the desired .jar file. The jar files can either be a complete preinstall image of a patch file. The patch file contains only the delta between two versions. By convention, these files are named patchSRCtoDEST.jar and patchSRCtoDEST-editor.jar (SRC and DEST denote four-digit version numbers).

When the upgrade transfer is complete, a message will be displayed informing you of this and reminding you that the Server/Editor must be restarted.

16.19 Server Upgrade Log

When a server upgrade is performed, a log is created containing the information of which files were retained in their old version, i.e. from the kept list, and which have been replaced. The upper listbox contains information about the upgrade program itself including the name of the current log file. This name is also displayed above the second listbox. The upgrade-log files are saved to the root directory of the Server.

16.20 Reload TCPIP/LU Mapper

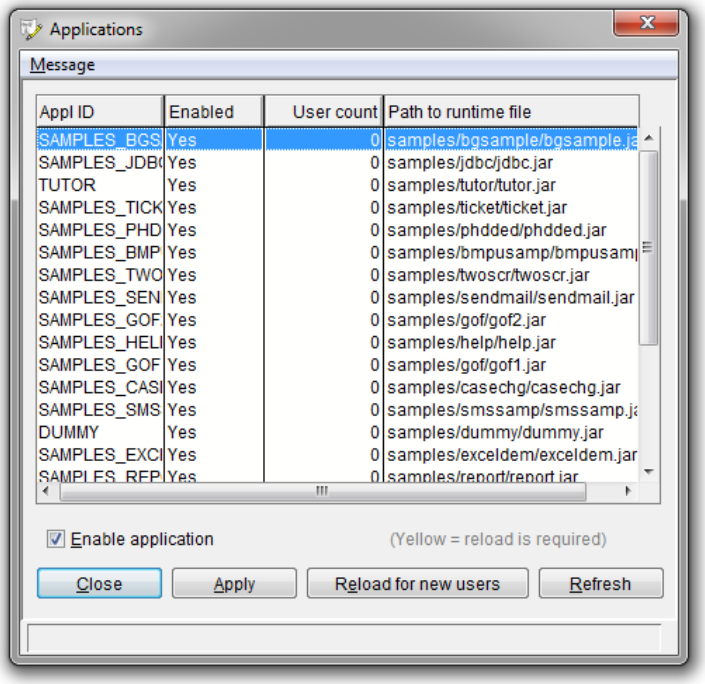
This function will cause the method `refresh()` in the LU Mapper Interface to be called. The `DefaultLUMapperExit` will reload the contents of the text file specified in `server.ini` as `LUMapperFileName`.

This function is not available in the Editor.

16.21 Enabling, Disabling or Reloading Applications

With the menu item **Application – Applications (**or **Server – Applications…** in the Editor**)**, the list of all configured and loaded applications is displayed.

The server normally loads several runtime files into memory when it starts. If one or several of these files need to be updated, they will have to be reloaded using this function.



In the list, the number of concurrent users that references an application is displayed.

To reload an application, select it in the list and press **Reload for new users**. The reload of the runtime application is done immediately. All new client connections will use the newly loaded application.

When a application definition (the .jar file) has been updated in the file system, the application should be reloaded for the new definition to be used. This is indicated by the list item background being colored yellow. Note: The application needs to be explicitly reloaded as indicated. No *auto-update* function is available.

If you wish to disable or enable users to access an application, select it in the list, check or uncheck **Enable application** and then press **Apply**.

To send a message to all users of a particular application, select the application in the list, right-click the mouse button and select the pop-up menu item **Application message**.

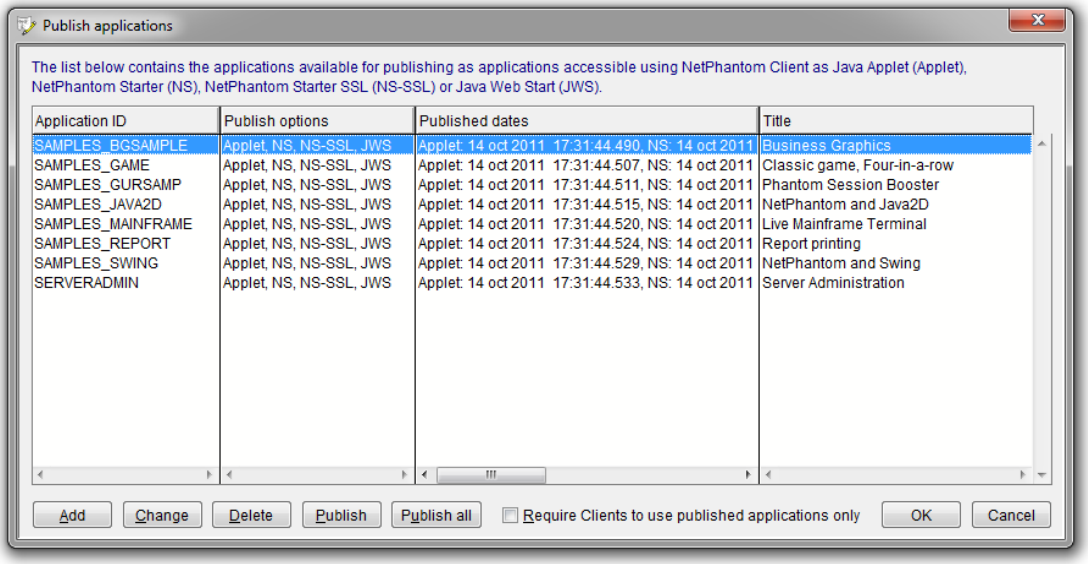
> **Note:** Changes made here are not saved in the `server.ini` file. This means that at restart these settings will be lost. If changes should be saved in `server.ini`, use the **Applications** page in **Configure server** dialog box. See *Applications* above.

16.22 Publish Applications

With the menu item **Application – Publish Applications (**or **Server – Publish applications** in the NetPhantom Editor**)**, the way that applications are published is specified and performed.

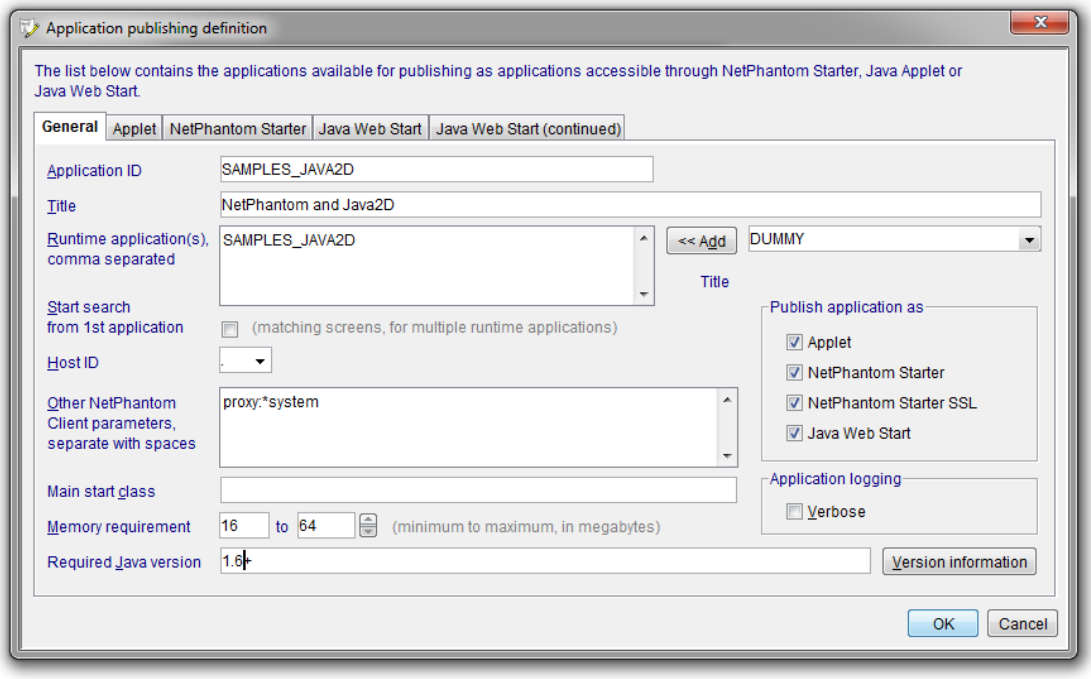
The available types of publication are Java Applet, NetPhantom Starter (encrypted using SSL and unencrypted) and Java Web Start. For each of these types a wide range of options are available for tailoring the way that the applications become available to connected users.

Publication is in fact also a way to structure application functionality. A publication definition can contain several runtime applications. The combined set of functions and panels are then available through the defined application publication (in the defined forms). A similar thing can be achieved by merging runtime applications (please see chapter *Building an Application*). The fundamental difference here is that the runtime application structure is defined dynamically at run-time when using the publishing alternative. This is as opposed to the merge application alternative, where the hierarchy is established at build time. Please note that the two ways of establishing application are not mutually exclusive. Merged applications can be published together with other runtime applications.



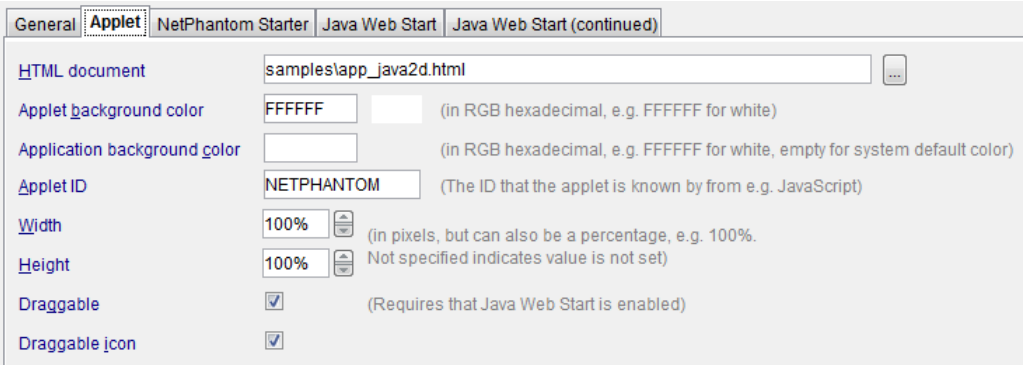
Create a new or edit an existing published definition.

The list contains all publish specifications – ID, types, publish date and title. The different types are Java Applet, NetPhantom Starter (unencrypted and SSL) and Java Web Start (JNLP).



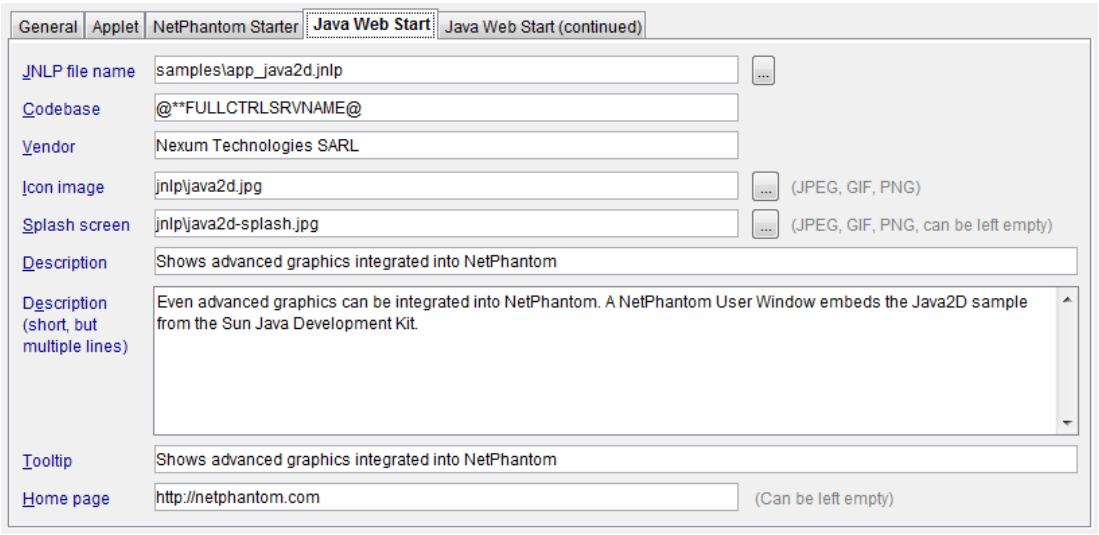
Application ID	All users referencing the application that you select in the combination box.
Title	The title of the application. Displayed in the application overview list and used as the title for the respective publication alternatives.
Runtime application(s)	A list of which runtime applications that should be part of the application publication. Specifying multiple runtime applications means that all the functions and panels in the the set of applications are available through the application publication they belong to.
Start search from 1st application	If checked the search order for matching screens will be the same as the applications are listed above. If not checked, the current runtime application is searched first.
Host ID	The host id to use. Must be defined before.
Other NetPhantom Client Parameters	Additional NetPhantom parameters to use. Please see chapter 5.1
Main start class	The java class to use as start point for application.
Memory requirement	Two numbers mean: • initial Java heap size (16 to 1000 MB). • maximum Java heap size (initial heap size to 1000 MB). Note: This specification applies to all the ways of running the application.

Required Java version	The version attribute can not only specify an exact version but can also specify a list of versions, called a version string. A version string is an ordered list of version-ranges separated by spaces. A version range is either a version-id, a version-id followed by a star (*), a version-id followed by a plus sign (+) , or two version-ranges combined using an ampersand (&) . The star means prefix match, the plus sign means this version or greater and the ampersand means the logical AND of the two versionRanges. For example: `11+ 10.* 9.0.2+ 1.8.0_151+` The meaning of the above is: the Java version that either has version 11 or better, any Java 10 version, Java 9.0 update 2 (or better update) or any Java 1.8 update 151 (or better update).
Publish options	Select the type(s) of ways that the application is to be published: Applet, Starter, SSL Starter and Java Web Start.
Application logging	Option to use verbose logging for the client (i.e the direct logging available on the Java console).



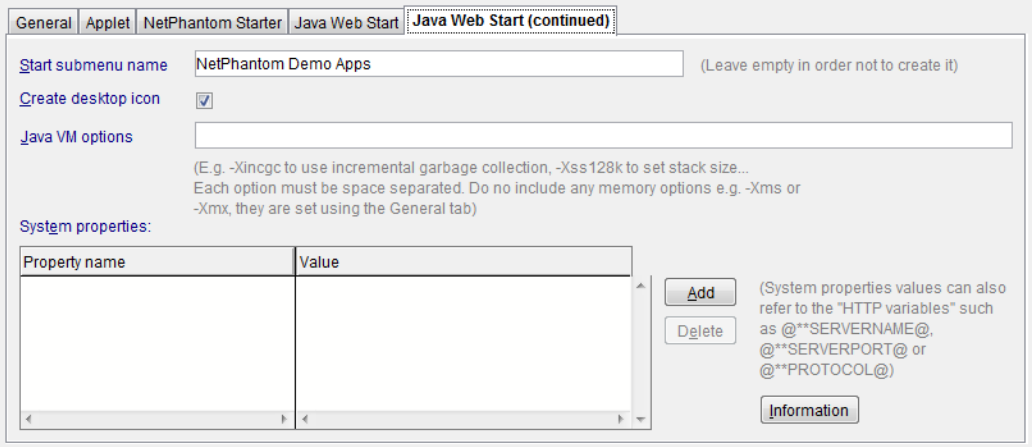
HTML document	The name of the html page container page. This name needs to have a ".html" or ".htm" suffix. The applet is published under this name at the root of the server so that it can be accessed in a browser under the address: http://<server name or IP>/<page name>
Background color	The background color to be displayed in the brower when the applet is accessed.
Applet ID	The identification that can be used if JavaScripts are present in the containing HTML document.
Width	The width of the applet in the browser in percent (using the '%' suffix) or in pixels (without suffix).
Height	The height of the applet in the browser in percent (using the '%' suffix) or in pixels (without suffix).
Draggable	Allow the applet to be dragged outside the browser, and that way also install the application on the desktop.
Draggable icon	Indicate that the applet is draggable with a small icon in the lower right corner of the applet area.

File name	The starter package file name. Must have a ".pkg" suffix. This name will be used as the name available in the application selection function when using the NetPhantom Starter.
File name (SSL)	The same as above but for the version where communication with the client is protected with the help of encryption.
Java VM options	Java virtual machine options (in addition to the memory requirements specified under the "General" tab). There is a vast number of parameters that can be specified here. They control the behavior, performance and debugging of the Virtual Machine. These options can change between versions so please refer to the Java documentation for the specifics of these options.



JNLP file name	The JNLP file name. Must have a ".jnlp" suffix. The web start application is published under this name in the root of the server so that it can be accessed in a browser under the address: http://<server name or IP>/<jnlp applikation name>
Code base	The code base is used by the java classloader. The default value refers to the document base of the server (using an HTML variable. For more information on HTML variables, please see *NetPhantom Markup Tags in HTML*). This can, of course, be changed or added to.
Vendor	The vendor attribute to be used in the web starter definition. When an application has been installed this information is available in the control panel associated to the application.

Icon image	The image to use as icon for the application. JPEG, GIF or PNG format. This image is used as the icon in the start menu and for the desktop shortcut when these start alternatives are configured (please see the next screen description).
Splash screen	The image to use as a splash screen when the application is started. JPEG, GIF or PNG format. This value can be left blank, and, in that case, it means that no splash screen should be used.
Description	The single line description.
Description (short, but multiple lines)	The "short" description attribute that can consist of multiple lines.
Tooltip	The tooltip text to use for the application. The tooltip test is displayed when hovering over the desktop shortcut or the start menu item.
Home page	The home page attributes it to the jnlp application (can be left empty). When an application has been installed this information is available in the control panel associated with the application.

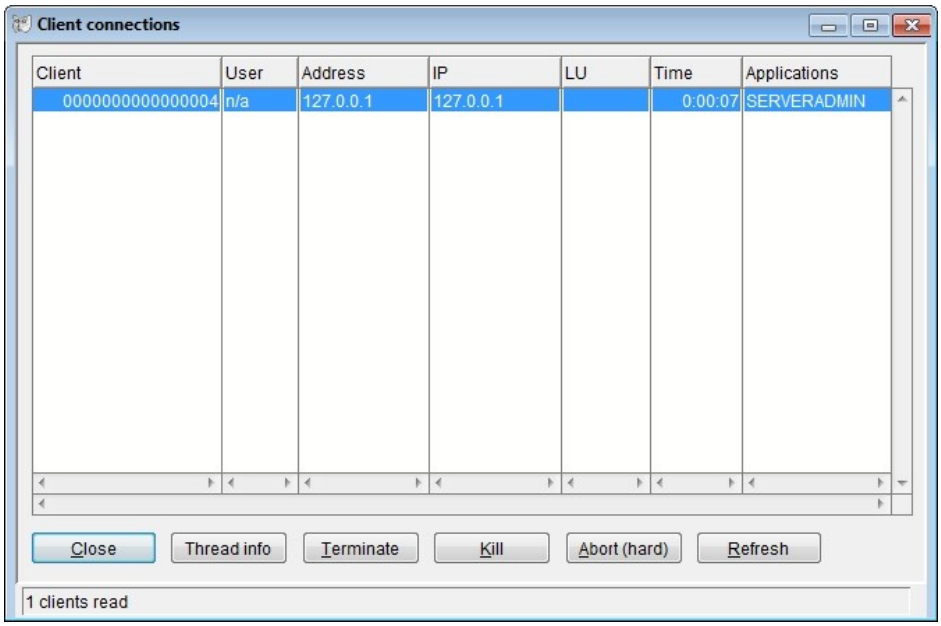


Start submenu name	If a name is specified here, it will be used to generate a submenu under the "Start" menu. If it is left blank, no submenu is created.
Create desktop icon	When the application is accessed the first time, create a desktop shortcut with the defined icon.
Java VM options	Java virtual machine options for the server-side execution of the application. Note: These are additional to the specification of the memory allocation under the "General" tab.
System properties	System properties to use when executing the jnlp application. Note that the "HTML variables" can be used here (e.g. @**SERVERNAME@ or @**PROTOCOL@). For more information on HTML variables, please see *NetPhantom Markup Tags in HTML*.

16.23 Client Connections

Use the menu item **Clients – Client connections (or Server – Advanced – Client connections…)** to display a list of all currently connected clients. In this window, you can send messages to a particular client or terminate the connection (to dispose of a client

connection). It is good practice to send a message to a client *before* terminating its connection.



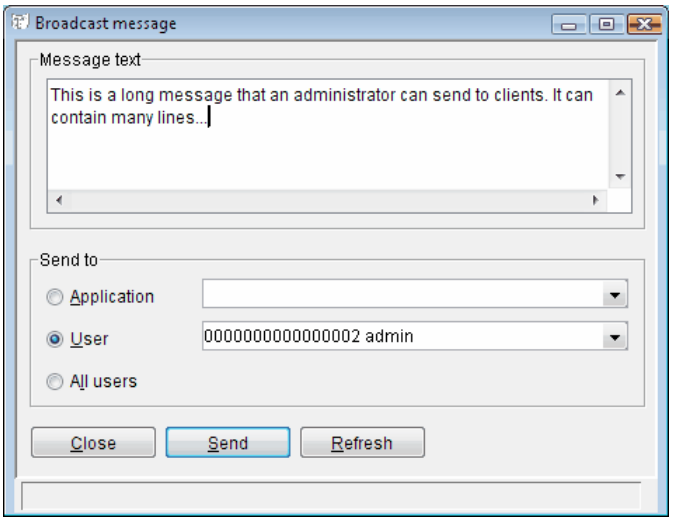
The server can terminate a particular client connection. The client will display a message box stating that the server administrator terminates the connection.

The list box is of multiple selection type, so several client connections can be terminated at the same time.

Detailed information about threads related to a specific client connection can be fechted through the interface. The collected information includes lock/synchronize/running states and stack traces.

16.24 Broadcast Message

The Broadcast message window is displayed when you select the menu item **Clients – Broadcast message** (or **Server – Advanced – Send Message**… in the Editor). A message can be sent to all users, all users using a particular application or to a single user.



*The status bar displays the number of clients that received the message once the **Send** button is pressed.*

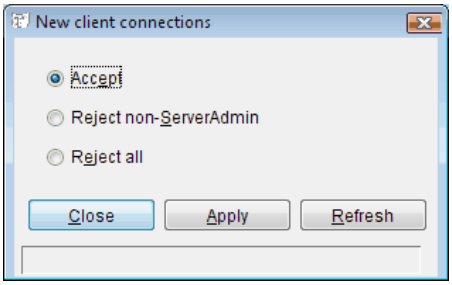
Use the radio buttons to select the destination type:

Application	All users referencing the application that you select in the combination box.
User	A single user that you select in the combination box.
All users	All currently connected users will receive the message.

Note: This function may take a while to complete because a client connection may be in such a state that the message-sending function is blocked until the client exits the state in question.

16.25 New Client Connections

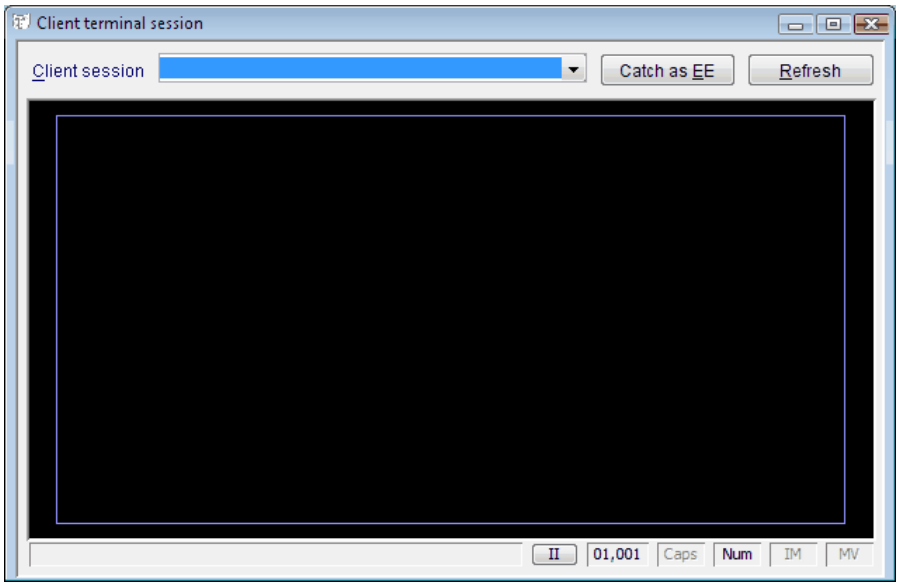
Selecting the menu item **Clients – New client connections** (or **Server – Advanced – New client connections…** in the Editor) will display the following window:



This function enables the server to accept or reject all new client connections. This eases for example the reloading of the runtime file. It can also be used to direct all new clients to a backup server because the current server must be stopped for service.

16.26 Display Terminal Session

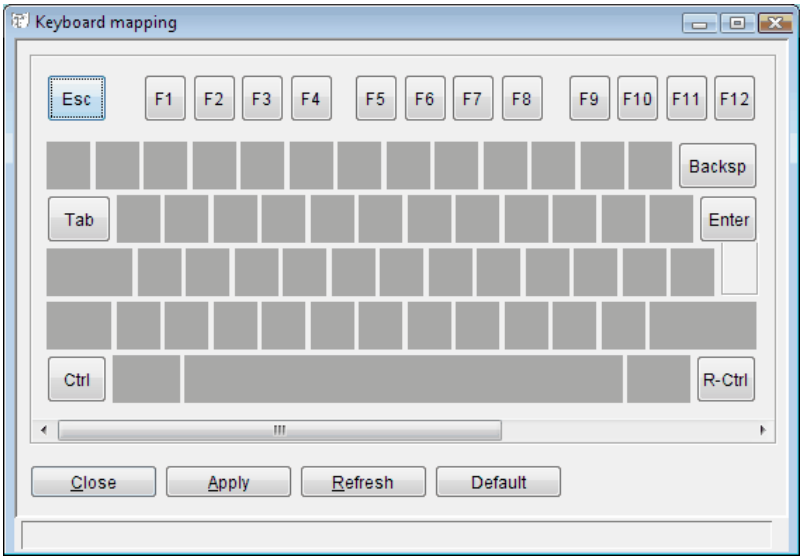
You can view the terminal session for a client via the menu item **Clients – Display terminal session**. Select the desired client from the combination box and click the **Refresh** button. The corresponding terminal screen is displayed, and you will be able to navigate in the host system. This may be useful for debugging.



First choose the client session and click the Refresh button to see the selected client terminal screen.

16.27 Keyboard Remapping

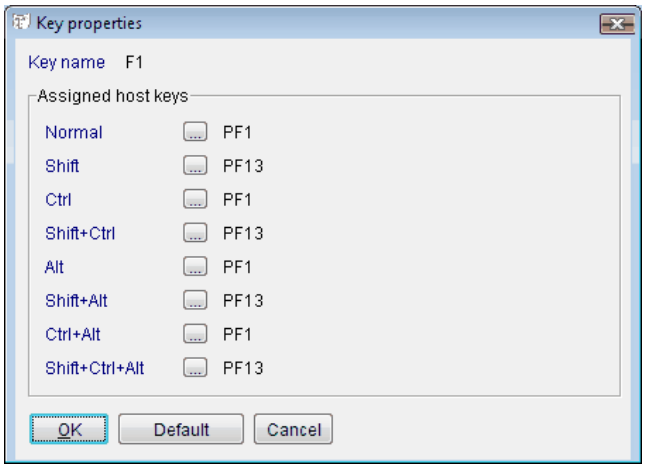
The keyboard mapping window is displayed when you select the menu item **Clients - Keyboard remapping** (or **Edit – Properties… - Keyboard** in the Editor). This keyboard remapping is only used in the terminal part and not used in the graphical panels.



Use the horizontal scroll bar to see the right part of the keyboard (because the keyboard is wide and often doesn't fit in the window).

Pressing the **Default** button in the Keyboard mapping window will cause the entire keyboard remap to be reset to default.

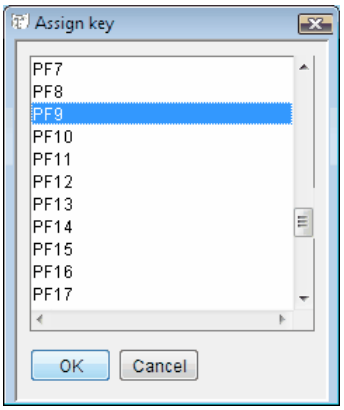
To remap the keyboard, select the key you wish to remap. The following dialog box is displayed:



Each key has eight key augmentation combinations. Each key augmentation is set individually.

Select the key augmentation you wish to remap and press the **[...]** button to the right to change the mapping.

If you press **Default**, all definitions for the key augmentations are reset to their default mapping.



Note: The list shows all available key functions for both 3270 and 5250 sessions. Some keys may not be implemented in the respective emulator.

Once a keyboard remap is complete, press the **Apply** button. All current client connections will receive the new keyboard remap *immediately*, so be careful!

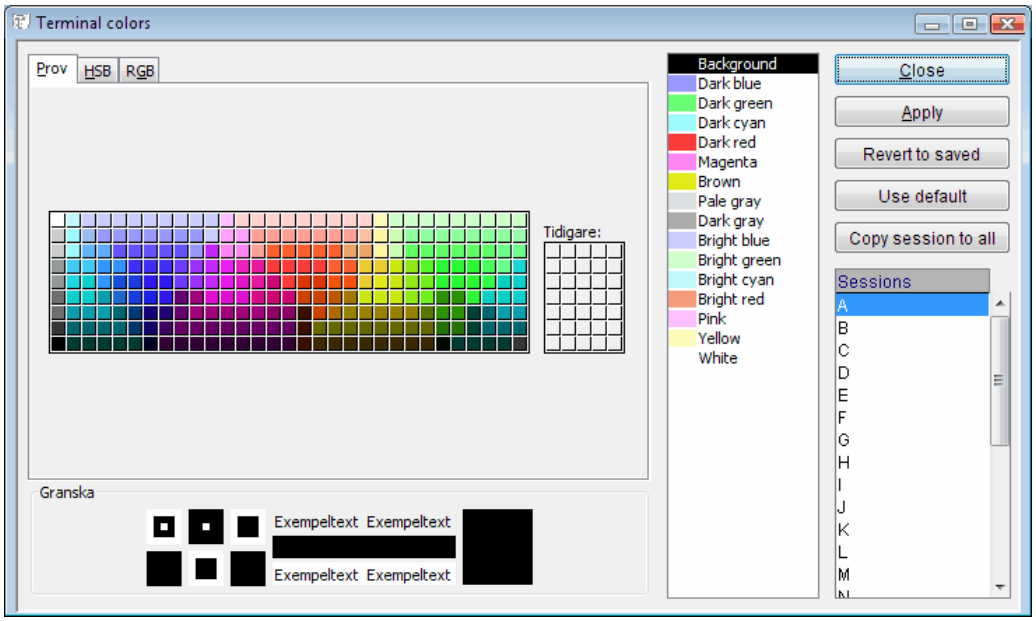
Note Concerning the Ctrl Key

It is very common in the "terminal" world to use the left and/or right control keys for different usage, e.g. left control key is *Reset* and the right control key is *Enter*. In Java, there is no way of differentiating between the two keys, so the left *and* the right control keys have the same keyboard mapping. There are two other keys that have the same functions: the *Enter* key (just below the backspace) and the *Enter* key on the keypad (Java does not differentiate there either).

Also, keep in mind that when you remap the Ctrl key, you must remap both the **Normal** function and the **Ctrl** function in the **Key properties** dialog.

16.28 Terminal Window Colors

Use menu item **Clients – Terminal window colors** (or **Edit – Properties… - Colors** in the NetPhantom Editor) to remap the colors displayed for terminal windows on the client side for each respective host session ID. The terminal windows displayed on the server when it runs in GUI mode also use these colors.



To remap a color for host session A, select session A in the **Sessions** list box, the color you wish to remap in the other list box and finally the requested color.

You can remap several colors without affecting the current setting for clients. However, when you press the **Apply** button, all new client connections or client connections that change host session (using e.g. the HostConnect function) will receive a new color setting.

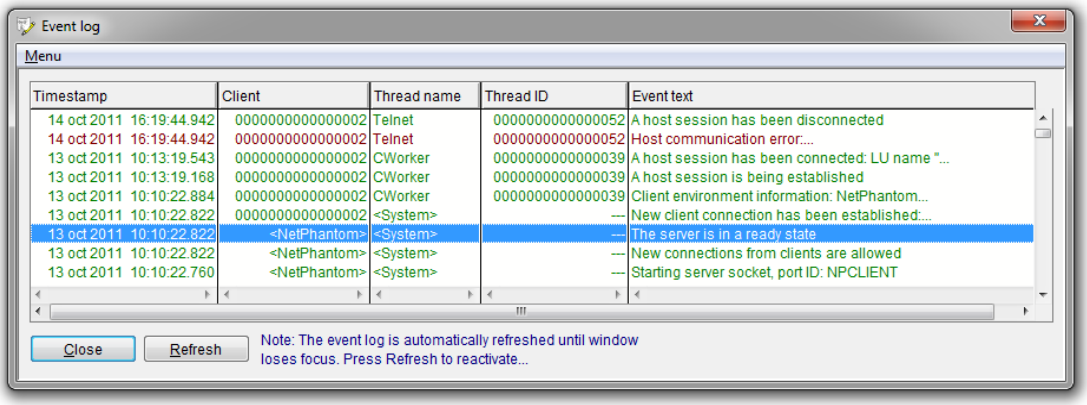
Use the **Revert to saved** button to restore the colors that were set prior to all your changes, for all host sessions. Pressing **Use default** will reset all colors to the default colors for the selected host session.

16.29 Get Client Font Metrics

Get the font metrics from the client and send them to the server to be written to the font metrics configuration (the *fontmetrics.dat* file). This so that all clients can use the same font metrics.

16.30 Event Log

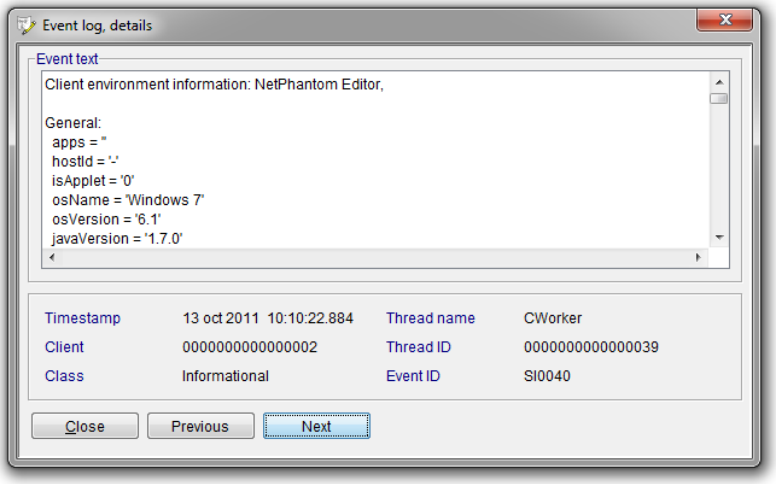
All events recorded for a server are normally saved to a file. Only a certain number of the last events are saved for retrieval by the menu item **Trace & Event – Event log** (or **Server – Event log…** in the NetPhantom Editor). The settings for the event file(s) and the count of events saved (history) are set in the NetPhantom server file `server.ini`.



The event log displays events in different colors depending on the level of importance. Information events are shown in green; warnings are displayed in blue, and errors in

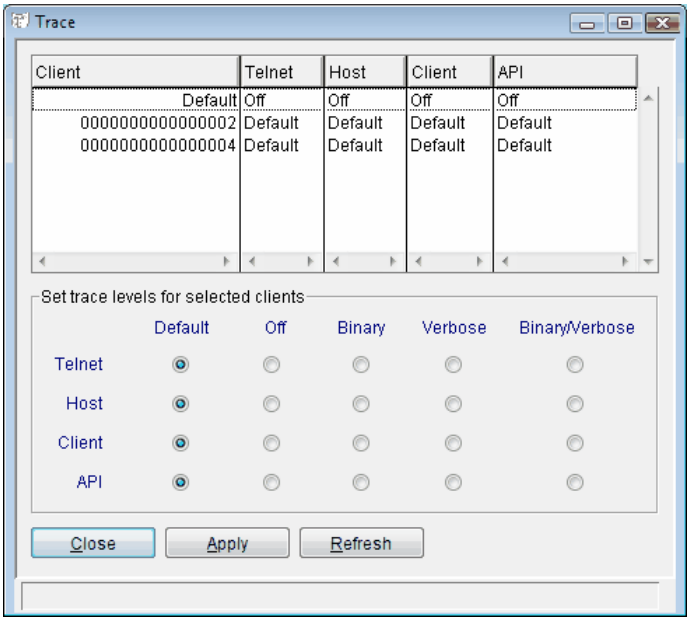
magenta while critical events appear in red. Any event that do not fall into one of these categories will be displayed in black. This makes it easier for administrators to quickly find the type of events they are looking for.

A detailed description of the event can be viewed by double-clicking the event in the event log. The **Previous** and **Next** buttons allow you to easily navigate between event details to get an overview of the chain of events.



16.31 Trace Settings

NetPhantom provides four subsystems, Telnet/Host/Client/API, that can be traced with any combination of "no trace", "binary" and "verbose". Select the menu item **Trace & Event – Trace settings** (or **Server – Advanced – Trace settings…** in the NetPhantom Editor) to change the current trace setting globally or for an individual client connection.

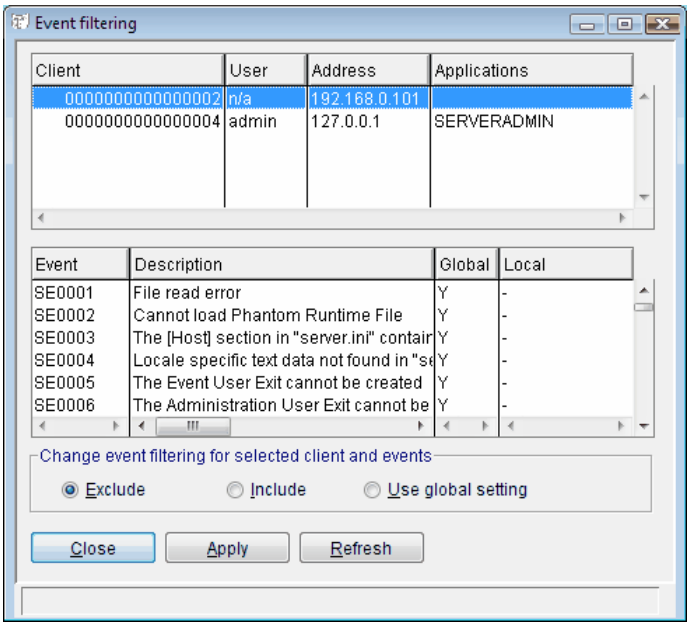


Select the default setting (top line in the list box) or a particular client connection. Set the radio buttons for each subsystem to the requested trace level. Then press the **Apply** button.

Note: Changes made here are not saved in the `server.ini` file. This means that at restart these settings will be lost.

16.32 Event Filtering

By default, all events in NetPhantom are saved in the event log (if not configured otherwise using the NetPhantom Server configuration file `server.ini`).



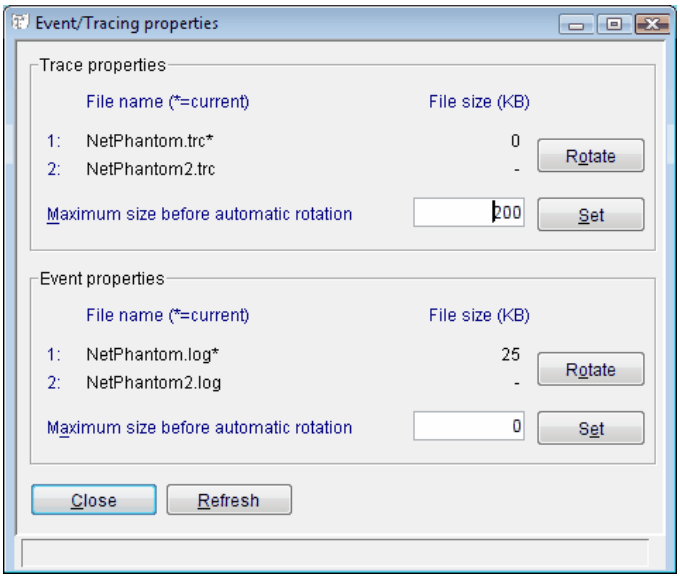
The lower list box shows all available events with their descriptions along with the global setting and the setting for the selected client session. A "–" (minus sign) in the rightmost column indicates the use of the global setting. "Y" indicates the event is included in the log and "N" indicates that it is excluded.

In the upper list box, select the client connection for which you wish to change the event filtering setting. In the lower list box select the events you wish to change and then click the **Exclude** or **Include** radio button. The **Use global settings** button should be selected for events that should follow the event filter settings defined for `ALL` clients in the server administration program. See *Event Filter Settings* above. Press the **Apply** button to implement the changes.

Note: Changes made here are not saved in the `server.ini` file. This means that at restart these settings will be lost.

16.33 Trace & Event Properties

Trace output and event log entries are normally written to files. Use the menu item **Trace & Event – Properties** (or **Server – Advanced – Trace properties…** in the Editor) to change the setting for automatic file rotation or to rotate the files.



A "" (star) after the filename indicates that the file is the currently used one. The value to the right of the filename indicates the current file size in KBytes.*

Press the **Rotate** button to rotate the trace or event file. This button is disabled if file rotation is disabled (see NetPhantom Server configuration file `server.ini`).

To change the setting for automatic file rotation, enter a value and press the **Set** button. To remove automatic file rotation, enter "0" (zero) as the file size and press **Set**.

> **Note:** Changes made here are not saved in the `server.ini` file. This means that at restart these settings will be lost.

16.34 Toolbox API

NetPhantom provides an API which allows the implementation of additional functionality from the Server Administration program. The GUI for new functions is built by the Editor. Additionally, some new classes must be created for the implementation, taking care of the input and output, and finally doing the work. All the defined functions are available via the **File – Toolbox…** (or **Options – Toolbox…** in the NetPhantom Editor) menu item.

To implement a new function, first a GUI resource file (or files) should be created using the Editor. The GUI resources should be stored in the NetPhantom installation structure. The next step is to create a Java class that uses the resource file and implements the dynamic behavior of the function. The resulting file(s) should be compiled and stored in a file structure under the installation directory of NetPhantom. Note that the java class structure (including packages) must be reflected in the structure.

Finally, the `toolbox.ini` file should be edited to declare the function. A new entry should be added as:

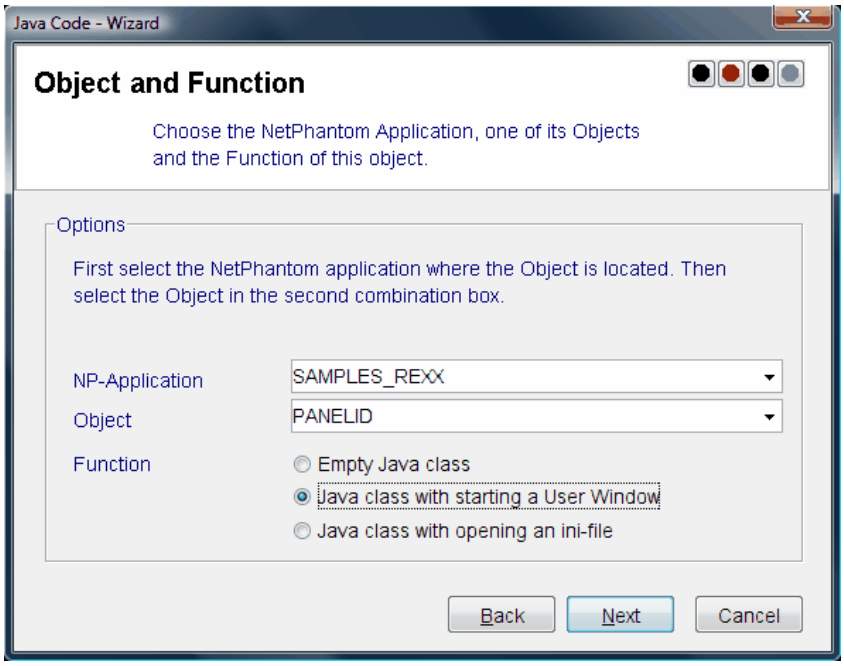
```
<Tool ID>=<Text for menu item>
<Tool ID>.app=<path in NetPhantom install>/<resource
     file name>
<Tool ID>.panel=<Initial panel ID>
<Tool ID>.class=<class name including package>
```

```
<Tool ID>.descr=<Description text (optional)>
```

A sample called "Generic Java Code Wizard" shows an easy way to install such helpful functions. The wizard supports creating Java source code that can be used in further developing of NetPhantom applications.

The features are:

- adjusting package and class names automatically,
- providing code for integrating custom configuration files (*.ini),
- providing code to integrate a NetPhantom Object with a User Window.



Any type of information from NetPhantom Server can be accessed, using the instance of the class *VirtualSessionManager*. Alternatively, code can be produced to use private ini-files. In this case the class *IniFile* provided by the NetPhantom API is used.

Another use can be a shortcut from a NetPhantom Object to a *UserWindow*. In this case, some source code is created to get access to a *UserWindow* (only the control ID defined in the Editor has to be assigned or adjusted) and to force the *UserWindow* to send a *Transaction* to the client.

Code generation result dialog.

17 The Remote Command Line Utilities

The NetPhantom Server can be accessed remotely by means of remote command line utilities. There are two utilities:

- concurrent user count,
- remote server administration.

These utilities are described below.

17.1 Concurrent User Count

The Concurrent User Count program connects to a number of NetPhantom servers to determine the maximum number of clients that have concurrently been connected to each server since it was started.

This command line utility is started this way:

```
java -cp NetPhantomServer.jar
    se.entra.phantom.util.ConcurrentUsers [-V] [-I]
    ServerAddress:adminPort
    [ ServerAddress2:adminPort2 ]
    [ etc ]
```

ServerAddress
: a valid IP address or DNS entry. The *ServerAddress* must be separated from the *adminPort* by a colon.

adminPort
: the port number of the contacted server's administration port, which is defined in the contacted NetPhantom server's `server.ini` file (see *Chapter Base Section* above).

The consecutive server addresses are separated by spaces.

The Verbose option:

The Verbose option displays the output in a human readable format as opposed to the usual machine parsable.

It can be defined with the following non-case sensitive syntax:

```
-V
-VERBOSE
/V
/VERBOSE
```

The Ignore option:

The Ignore option causes failures in contacting a NetPhantom Server to be ignored.

It can be defined with the following non-case sensitive syntax:

```
-I
-IGNORE
/I
/IGNORE
```

Return codes:

Normal	Verbose mode

```
nn              Total number of concurrent users nn.
999999          Error, plus error message
                telling where error occurred.
```

17.2 Remote Server Administration

This command line utility is started as:

```
java -cp NetPhantomServer.jar
    se.entra.phantom.radmin.ServerCommand
    ServerAddress:Port [Options]
    Command [Parameters]
```

The command line utility contacts the server and performs one command. The server connection is then closed.

Options:

```
-V          (Verbose)
-USER       (Administration user name)
-PW         (Administration user password)
-PASSWORD   (Administration user password)
```

Commands:

```
NOP
SHUTDOWN
MEMORY
BROADCAST
APPS
CLIENTS
DISABLE
ENABLE
LOCK
UNLOCK
PUT
PUTR
KILL
RESTART
RESTARTWS
UPGRADE
```

Return Codes:

Normal	Verbose mode
0	OK,
1	Error,
5	Syntax Error,
9	Host not responding.

The Verbose Option:

The Verbose option displays the output in a human readable format as opposed to the usual machine parsable.

It can be defined with the following non-case sensitive syntax:

```
-V
-VERBOSE
/V
/VERBOSE
```

The Authentication Options:

The Authentication options are used to specify the administration authentication data for accessing the server.

It can be defined with the following non-case sensitive syntax:

```
-USER:<user name>
-PW:<user password>
-PASSWORD:<user password>
/USER:<user name>
/PW:<user password>
/PASSWORD:<user password>
```

NOP Command

This function is used to check if the NetPhantom Server is running on a specific machine and port location. It sends a NOP (No Operation) signal to the server and waits for a reply.

```
ServerCommand TargetServer:Port [-V] NOP
```

Parameters:

```
none
```

SHUTDOWN Command

Terminates the NetPhantom server.

```
ServerCommand TargetServer:Port [-V]
     SHUTDOWN [*IMMED]
```

Parameters:

```
*IMMED    Shuts down the server immediately
          without waiting for all clients to
          end their sessions.
```

MEMORY Command

Displays information about the Memory usage of the NetPhantom Server environment and the memory load of each client connection.

```
ServerCommand TargetServer:Port [-V] MEMORY
```

Parameters:

```
none
```

BROADCAST Command

Broadcasts a message to the connected clients.

```
ServerCommand TargetServer:Port [-V]
    BROADCAST [ ALL | APP AppID | CLIENT ClientID ]
    Message
```

Parameters:

```
ALL                 Broadcasts to all clients.
APP AppI            Broadcasts to all clients running the
                      application AppID
CLIENT ClientID     Broadcast to clientID
```

Examples:

Sends the message "this is a test message" to all clients on server 127.0.0.1

```
ServerCommand 127.0.0.1:789 broadcast all "this is a test
message"
```

Sends the message "this is a test message" to all clients running application A

```
ServerCommand 127.0.0.1:789 broadcast app A "this is a test
message"
```

Sends the message "this is a test message" to client 2

```
ServerCommand 127.0.0.1:789 broadcast client 2 "this is a test
message"
```

APPS Command

Displays the applications available to the clients.

```
ServerCommand TargetServer:Port [-V]
    APPS [RELOAD AppID]
```

Parameters:

`RELOAD` *`AppID`* `Reloads the application` *`AppID`*

CLIENTS Command

Displays information about the connected clients.

DISABLE Command

Disables an application from being run. When an application is disabled, the clients can no longer load the disabled application.

```
ServerCommand TargetServer:Port [-V] DISABLE AppID
```

Parameters:

`AppID Application identification`

ENABLE Command

Enables a disabled application. This will allow the clients to load the application.

```
ServerCommand TargetServer:Port [-V] ENABLE AppID
```

Parameters:

```
None
```

LOCK Command

Locks the NetPhantom Server for future client connections. Once the Server is locked no clients are allowed to connect.

```
ServerCommand TargetServer:Port [-V] LOCK
```

Parameters:

```
None
```

UNLOCK Command

Unlocks the NetPhantom Server. Allows clients to connect to the Server.

```
ServerCommand TargetServer:Port [-V] UNLOCK
```

Parameters:

```
None
```

PUT/PUTR Commands

The PUT command will transfer files from to the destination directory on the Server.

```
ServerCommand TargetServer:Port [-V]PUT[R] destination
filespec1 filespec2 ... filespecNN
```

The difference between PUT and PUTR is that PUT operates on file basis, whereas PUTR operates on file basis recursively, including subdirectories in *filespec*. The wild card characters à-la-Windows (*?^) are supported.

Parameters:

```
destination  destination on the server relative to the
NetPhantom installation directory.

fileSpecNN  the specification of the file(s) to transfer
using absolute path.
```

KILL Command

Terminates a client connection from the Server. The *ClientID* will be presented with an information message prior to disconnection.

```
ServerCommand TargetServer:Port [-V] KILL ClientID
```

RESTART Command

The restart of the server can be Soft, Hard or Hard Java VM restart, with the option to perform this immediately or deferred (i.e. when all users have closed their sessions).

```
ServerCommand TargetServer:Port [-V] RESTART
                                [SOFT | HARD | HARDJVM] [*IMMED]
```

Parameters:

```
SOFT     Soft restart (see chapter Server Restart)

HART     Hard restart (see chapter Server Restart)

HARTJVM  Hard JVM restart (see chapter Server Restart)

IMMED    Do not defer restart until all users have closed
         their sessions
```

RESTARTWS Command

Restart the web server. This is always done immediately.

```
ServerCommand TargetServer:Port [-V] RESTARTWS
```

Parameters:

```
None
```

UPGRADE Command

The UPGRADE command will transfer the *patchFileName* file to the server directory to the file "upgrade.jar" and mark the server process that it can be upgraded (this will be performed at the next server restart using "Hard Java VM restart".

```
ServerCommand TargetServer:Port [-V] UPGRADE patchFileName
```

The patch file name should be a file called "patchXXXXtoYYYY.jar" or "preinstall.jar". Once the patch file is transferred, use the RESTART command below to complete the server upgrade.

18 Remote Assistance

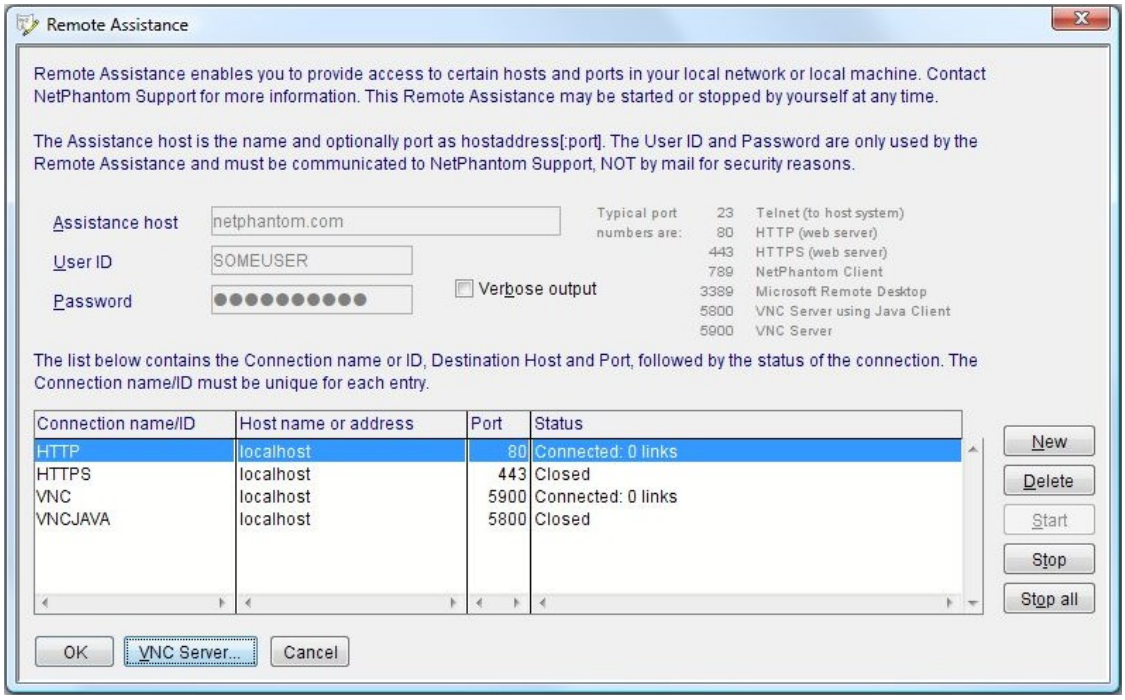
The NetPhantom Remote Assistance feature consists of three parts:

- the developer's end integrated into NetPhantom Editor,
- a Remote Assistance Server,
- the support-side of Remote Assistance.

First of all, NetPhantom Editor must be enabled for Remote Assistance. To do so, please contact NetPhantom Support. This is required for security reasons. The developer's end of Remote Assistance establishes a 128/256 bit encrypted and secured SSL connection with the Remote Assistance Server (for example *netphantom.com*). Proxy support by tunneled SSL or Socks is provided if required. The connection has a *Name or ID* associated with a *User ID* and *Password*. With these three parameters, the support-side of Remote Assistance also establishes an SSL secure connection with the Remote Assistance Server that creates a connection to both ends if the three parameters match. This effectively establishes a tunnel between the two ends: developer and support. Several communication links can be established in parallel through this tunnel. At the developer's end, a connection initiated from the support-side can only be directed to a particular host or address (typically *localhost*, i.e. the local developer machine) along with a specific port number (*5900* for VNC).

NetPhantom installs the VNC Server (Free Edition) software from RealVNC that can be used to enable NetPhantom Support to help a developer "drive" his machine interactively to help solve an issue or for educational purposes. It can also be used by a developer to show support for what happens in the system when an application is running.

The dialog box below is shown when the menu item **Help - Remote Assistance** is selected. The parameters that are common to all connections are the *Remote Assistance Server* (its host name), the *User ID* and the *Password*. These settings are stored in-memory if the dialog box is closed and brought back up. When the Editor is closed, these settings are saved in *server.ini*, a part of the Password for security reasons. Started connections remain open until explicitly stopped, regardless of if the dialog box is closed. When the Editor is closed, the connections are also closed (and the settings saved in *server.ini*).



Above you can see that 4 connections are defined, HTTP (80), HTTPS (443), VNC (5900) and VNC for Java client (5800), where the HTTP and VNC ports are started, but no communication links from the support-side are yet established. Once at least one connection is started, the three entry fields (Assistance host, User ID and Password) are disabled.

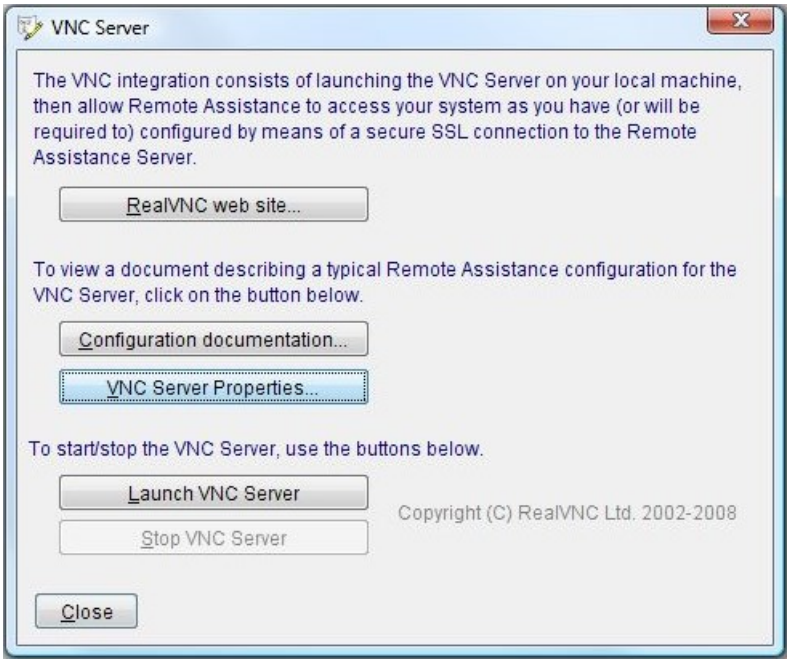
To launch or stop the VNC Server, click on the **VNC Server** button. Follow this link for more information and how to configure the VNC Server.

18.1 VNC Server Settings

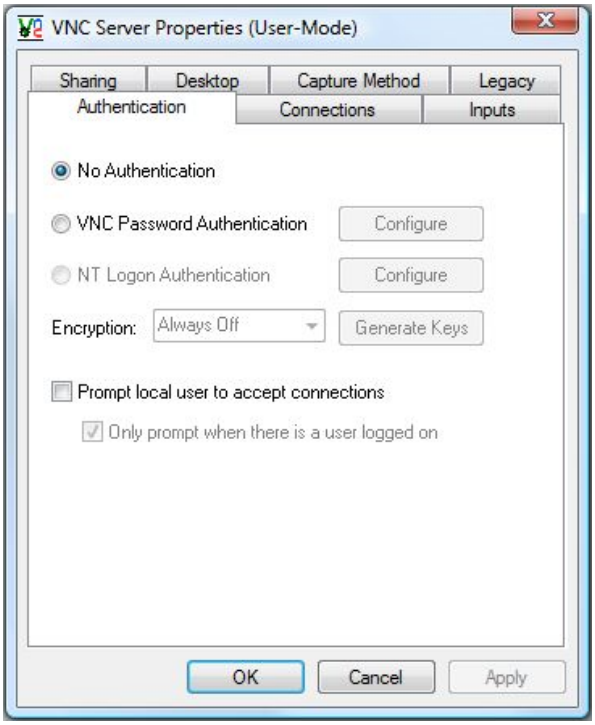
The VNC Server installed with NetPhantom is the VNC Server Free Edition that has some limited capabilities compared to the full versions. This version is a free version from RealVNC Limited. Only the Server and its Settings programs are installed under the *RealVNC* sub-directory. The main limitation is that communication between the Server and Client is done without encryption. As this communication is tunneled through the *NetPhantom Remote Assistance Server* using a 128/256 bit SSL encrypted connections, it becomes secure

VNC stands for *Virtual Network Computing*. It is remote control software which allows you to view and fully interact with one computer desktop (the “VNC Server”) using a simple program (the “VNC Viewer”) on another computer desktop anywhere on the Internet. The two computers don't even have to be the same type, so for example you can use VNC to view a Windows Vista desktop at the office on a Linux or Mac computer at home. For ultimate simplicity, there is even a Java viewer, so that any desktop can be controlled remotely from within a browser without having to install software.

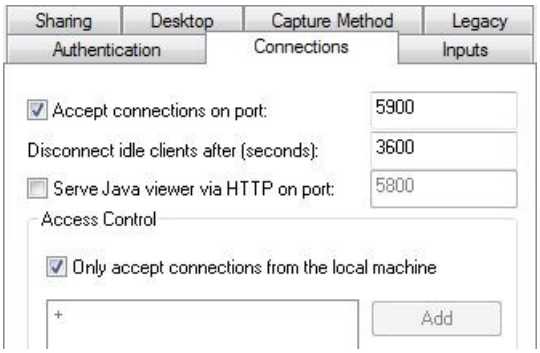
When configuring the VNC Server, a dialog box as follows is shown:



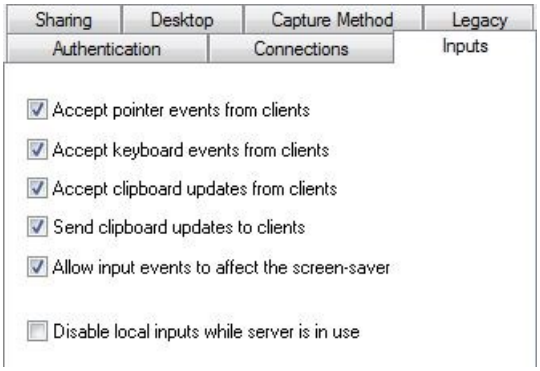
Click the button **VNC Server Properties** to bring up the dialog box below. Make sure to *select* the option **No authentication** and *unselect* the check box **Prompt local user to accept connections**.



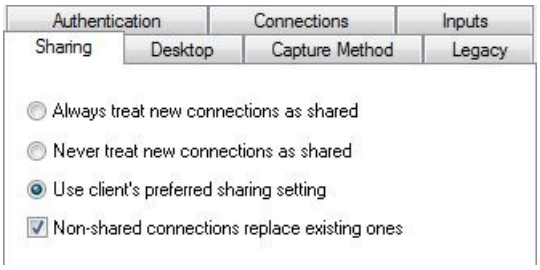
The following tab, **Connections**, requires the options **Accept connections on port: 5900** and on *Access Control*, the option **Only accept connection from the local machine**. NetPhantom Remote Access is only working on your computer, so there is no need to enable connections from other computers.



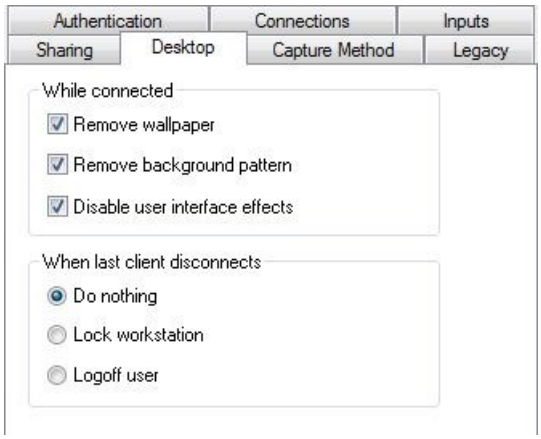
On the **Inputs** tab, set the check boxes as show below:



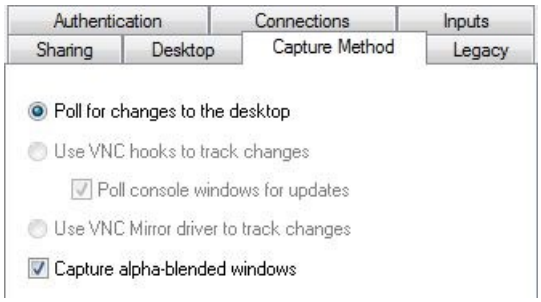
There are no special requirements for the **Sharing** tab, the defaults are shown below:



In order to speed up image transfer, check the options **While connected** (*Remove wallpaper*, *Remove background pattern* and *Disable user interface effects*, such as fade, scrolling) that are used for enhanced screen display. These options are located on the **Desktop** tab as shown below. Also make sure to *select* the option **Do nothing** as action **When the last client disconnects**.



Screen capturing is handled on the **Capture Method** tab, and it is recommended to select **Poll for changes to the desktop** and **Capture alpha-blended windows**. When a VNC Client Viewer connects, it has a setting for the color depth of the images transferred.



The **Legacy** tab can be left with defaults as below:

19 The Telnet Tracer

Telnet tracing records the telnet sessions between a NetPhantom Server and a host, it allows the debugging of applications and the creation of test cases for support. This feature is built into two stand-alone utilities: the tracer tool and the playback tool.

The tracer is a tunnel that the telnet stream passes through. It records and identifies each stream to allow tracing of multiple clients. Recordings are placed in a data file to recreate them with the playback tool.

The playback tool reads the traced data file and replays the interaction to a NetPhantom server and a client.

19.1 Telnet Tracer

The telnet tracer must be run and inserted between the NetPhantom Server and the host. The tracer can run on the same physical machine as the Server or on a separate machine, it is completely transparent to the operation of the applications and host.

The data is stored in EBCDIC format.

Start Command

The telnet tracer is contained in the `NetPhantomServer.jar` and initiates with the following command:

```
java –classpath NetPhantomServer.jar
se.entra.phantom.tracer.TelnetTracer [%]host:port
[!]tracefile [localport]
```

Required Parameters:

Host
The host that the tracer will record.

Port
The telnet port of the host, the default is 23.

Tracefile
The file where the telnet data is saved.

Optional Parameters:

Localport
The local telnet port, in case port 23 is being used.

%
Sets the socket in byte-per-byte mode.

!
The data is stored in ASCII format.

Installation

To install the Telnet Tracer, some changes are needed in the setup of NetPhantom Server. These changes are to redirect the network traffic to the Telnet Tracer instead of going directly to the host.

Start the Telnet Tracer with the Start command (see above). Specify which host will be recorded and the file where the data will be stored.

In the *Server Administration* program use the **Host** page that is in the *Configure Server* base panel. Modify or add a host setting. It should specify the machine where Telnet Tracer

is running or *localhost* if you are running the tracer on the same machine as the NetPhantom Server. If you have set the tracer to use a special local port, also set that in the host settings.

Restart the NetPhantom Server and all client sessions that connect to the server will be transparently recorded into the data file.

19.2 Telnet Trace Files

The trace file is editable to allow modification of the telnet data and to add programmed logic. This provides the ability to create an efficient data sample for bug reports, performance tests and demonstrations.

Data Format

Transaction header

Each transaction has a header line that contains a time stamp and information regarding the transaction. The header contains the date, the time, and the event information. The event information can indicate the direction of the transaction, the client number, and the status.

Example:

```
 21 sep 2021   10:02:42.531  <NetPhantom>      <System>   ---     TRACE STARTED
```

Transaction body

The header is followed by the transaction body. The body contains a hexadecimal and ASCII representation of the data.

Example:

```
85 94 7A 20 00 20 C5 D5 E3 D9 C1 F4 00 00 20 11     em:...ENTRA4....
03 01 3A E2 85 93 85 83 A3 40 96 95 85 40 96 86     ...Select one of
40 A3 88 85 40 86 96 93 93 96 A6 89 95 87 7A 20      the following:.
11 05 06 20 F1 4B 00 E4 A2 85 99 40 A3 81 A2 92     ....1..User task
```

Logic Variables

Logic variables are logic commands that are inserted into a recorded trace file to allow programmed loops and delays. They are used to create special test files from recorded sessions. As the telnet playback trace file are ASCII text files, they can be edited with a normal text editor to insert these special variables.

Usage

The logic variables are placed on a new line between the transaction header and the transaction body. In the following manner:

```
21 sep 2021 10:02:53.562  <NetPhantom> <System> --- >1: inbound data
*WAIT 3000
00 0A 12 A0 00 00 04 00 00 0C FF EF .. .. .. .. ................
```

Variables

**LOOP*

This variable is used to loop to the trace file and repeat the playback of the telnet session. It should be placed on the line above the line where you want to restart. If the generated trace file is too large, you may want to delete the end of the file.

**WAIT 3000*

This variable creates an **x** milliseconds wait before it continues with the next transaction.

Suggested Usage

The modified trace files can be used in conjunction with the Stress tools to test performance of your computing environment.

See chapter *The NetPhantom Stress Tools* for information on how to set this up.

19.3 Telnet Playback

The Telnet Playback reads the saved data and replays it to the NetPhantom server.

Start Command

The telnet playback is contained in the `NetPhantomServer.jar` and initiates with the following command:

```
java –classpath NetPhantomServer.jar
se.entra.phantom.tracer.TelnetPlayback Tracefile [port]
```

Required Parameters:

Tracefile
The file where the telnet data will be read from

Installation

To playback a telnet session, start the Telnet playback using the Start command. Use the data file that you recorded with the Telnet Tracer.

If you have just traced a host, you can use the same NetPhantom settings as when you recorded. Just shut down the tracer program and start the playback utility.

If not, in the Server Administration program use the **Host** page that is in the Configure Server base panel. Modify or add a host setting. It should specify the machine where Telnet Playback is running. This would be *localhost* if you run the playback on the same machine as the NetPhantom Server.

Connect a NetPhantom Client to the Server for the recorded session to be recreated.

20 The NetPhantom Stress Tools

The NetPhantom Stress tools are used to test how your network environment and your system environment handle a high load of NetPhantom activity. These tools create an arbitrary amount of client connections to the NetPhantom Server.

20.1 Stress Tool

The stress tool creates an environment for performance testing of your NetPhantom installation by connecting to a NetPhantom Server with a defined number of clients. If used with TelnetPlayback and an edited trace file, it can create an environment that performs a realistic sample of application workload.

> **Note:** Stress will load up a number of full graphically enabled clients on the same machine, if a large number of clients are started it can create a considerable number of windows on your desktop.

Options

ClientAmount

The number of clients to start.

ClientArgs

The arguments that are used to initiate client connections. Refer to this manual for possible NetPhantom Client arguments. Some common arguments are:

Host:*host*
App:*application*
Hostid:*host id*

Usage

The Stress test tool is a standalone java application that is run with the following command:

```
Set CLASSPATH=NetPhantomServer.jar
java se.entra.phantom.test.Stress ClientAmount [ClientArgs]
```

20.2 StressNoGUI Tool

The *StressNoGUI* application connects a defined number of clients to the NetPhantom Server. These clients have been stripped of their graphical interface to provide the potential to simulate a larger amount of client interactivity from a single machine. It also provides more advanced features like displaying the time difference between transactions, the window for the moving average, and warnings on the time differences.

Options

DispDiff:1

will display each sliding average value.

MinOut:0,1,2

The detail of the output to the screen. 0 is everything, 1 is normal and 2 is silent.

It is preferred to put the *StressNoGUI* in silent mode because displaying text creates a bottleneck with many clients.

MinDiff:NN

if diff time is lower than NN, a warning is displayed.

MaxDiff:NN

if diff time is higher than NN, a warning is displayed.

Window:NN

the number of values used for the moving average (more than 10 is recommended, preferably 50).

Usage

The *StressNoGUI* test tool is a standalone java application that is run with the following command:

```
Set CLASSPATH=NetPhantomClient.jar;NetPhantomServer.jar
java se.entra.phantom.test.StressNoGUI ClientAmount DispDiff:1
MinOut:1 MinDiff:NN MaxDiff:NN Window:NN [ClientArgs]
```

21 Overview of the Client Architecture

When the client establishes contact with the server(s) the following actions will be performed:

- Access to the server and application(s). If this fails, the client will try accessing another (backup) server.
- Establishment of the actual session to the server.
- The server establishes the connection to the host.
- The server confirms connection to the host (successful or unsuccessful). At this point, the GUI transactions start.

21.1 Schematic Client Structure

All the component instances of NetPhantom panels have a corresponding virtual component in the server. For a closer description of the virtual server components, see *Server Components*.

The structure of the client is as follows:

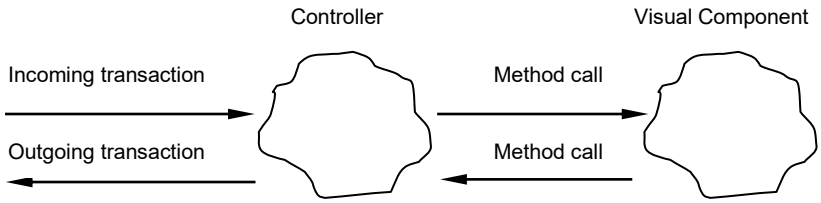
The Panel Data above is more complex. It contains:

- Panel specific data
- Menus
- Controls

21.2 Component Model

The panels, menus and controls all share the same structure and are called components. The visual components on the client are made up of two distinct parts. One is the actual GUI component, and the other is its controller component.

When the controller receives a transaction from the server, the visual component will be called using its methods such as *setText*, *setForegroundColor*. When the user interacts with the visual component, it will call the appropriate methods in the controller. If the component is an action component, this will trigger a transaction from the client to the server with all components of the current panel that have changed followed by the action component index.



The relation between a List Box and its Controller

The controller will in fact be made up of two parts, one part that is common to all types of components, and one that is unique to the type of component it controls.

Component Types

Here follows a list of all available component types.

- menus and menu items
- static text
- output text
- entry field
- multiple line entry field
- combination box
- spin button
- push button
- check box
- radio button
- list box
- group box
- frame/rectangle
- business graphics
- external image control
- subwindow
- notebook
- user window
- toolbar and status bar

22 Overview of the Server Architecture

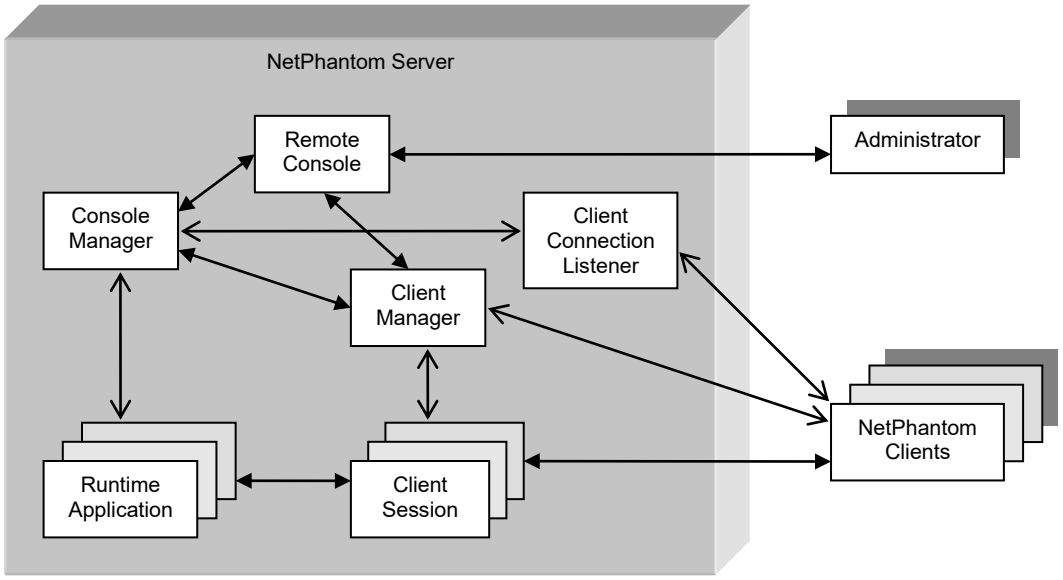
The server normally runs as a "daemon" process in non-GUI mode, i.e. it does not display graphical windows on the console. It can also run in GUI mode, which displays the terminal session windows for each client session. These windows can also be operated by means of physical input. The latter mode is used for testing/debugging NetPhantom Applications.

As the server is written in Java, it can run on Windows or Unix/Linux platforms, if the Java environment requirements are met. This can be useful for NetPhantom developers who wish to install a server on their development machine for testing purposes.

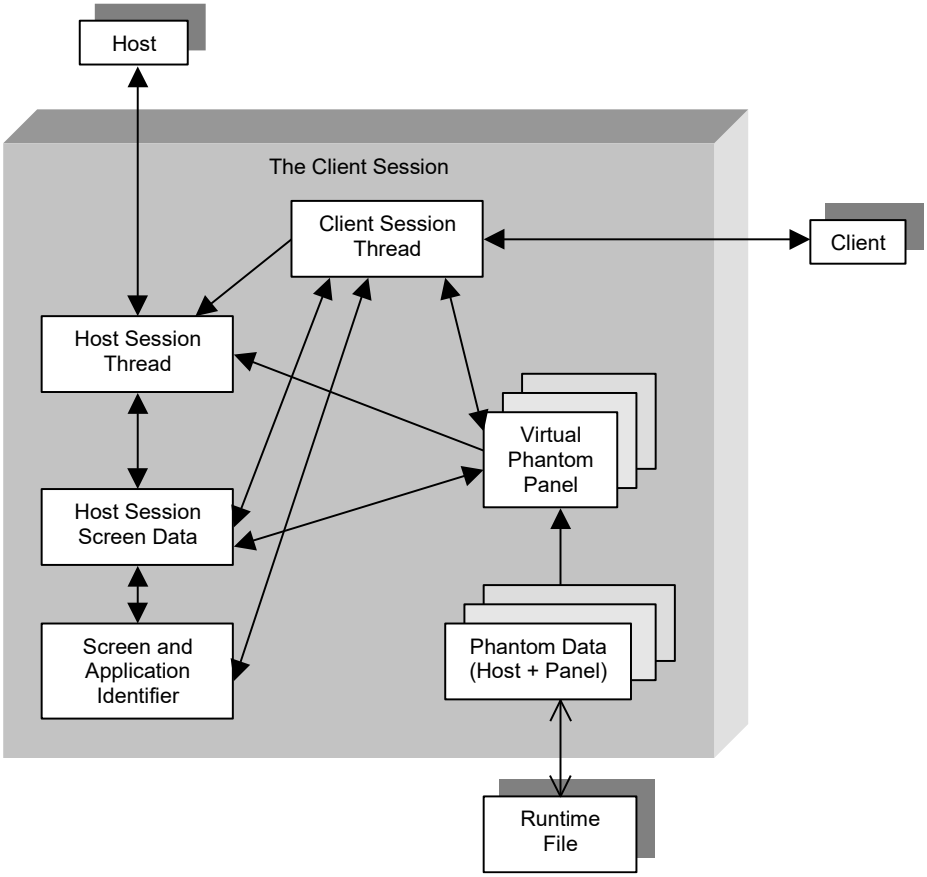
The non-GUI server mode requires several functions that can be performed by an administrator. See *Chapter The Server Administration Program* for more information.

22.1 Schematic Server Overview

Below, thin arrows indicate a loose relationship between components and the black arrows a tight relationship.



The "Client Session" box above is schematically described below:



22.2 Mainframe Host Communication

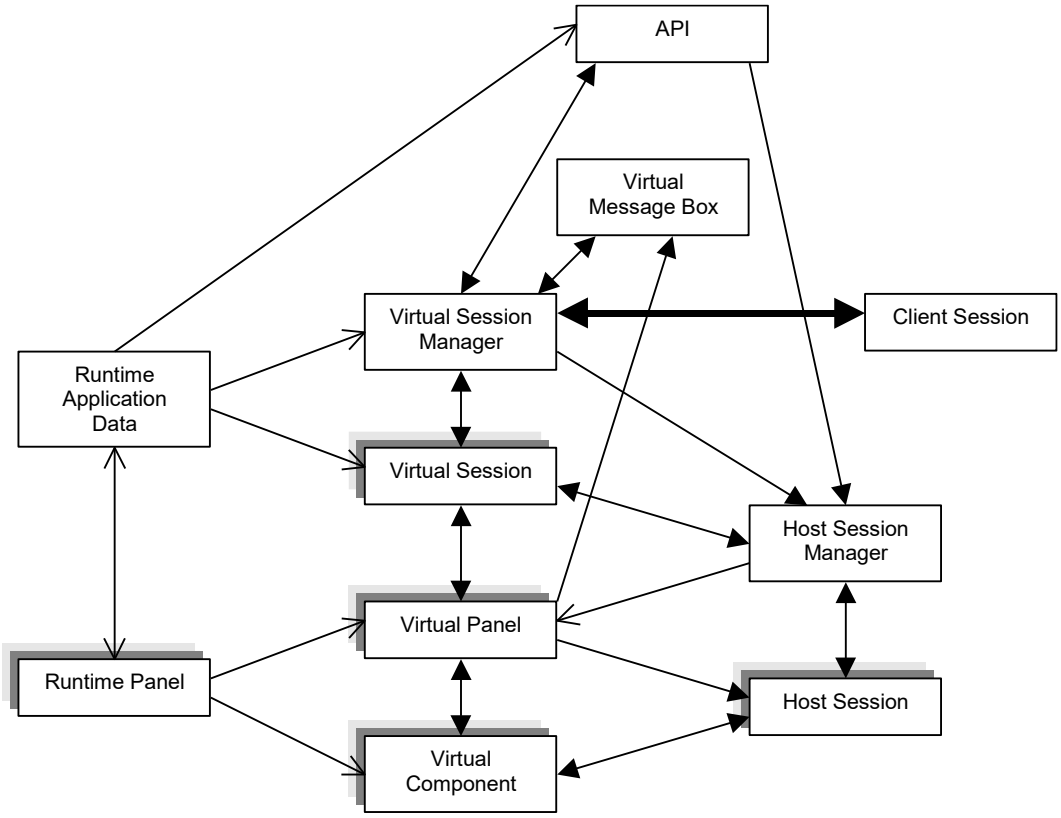
The server communicates with the mainframe host using the TN3270E protocol or the AS/400 using TN5250E. These protocols have become a de-facto standard for emulation within TCP/IP-based networks. The protocol is basically a “simple” version of the full SNA-based 3270/3250 protocol developed by IBM. Not all the features within 3270/5250 are supported by standard TN3270/5250 today. In the Telnet world there exist two separate versions of the protocol: TN3270/5250 and TN3270E/5250E. To cope with the drawbacks of the standard TN3270/5250 protocol, the new E-protocol has been enriched with functionality to simulate a full SNA 3270/5250 terminal PU.

From our perspective, NetPhantom is only interested in pure emulation, but the need to map a client to an LU requires the use of TN3270E/5250E.

22.3 Server Components

This section is intended for programmers and describes the architecture of the server components.

The following diagram indicates the relationship between the server components.



22.4 Relationship between Server Components

Virtual Session Manager/Client Session

All transactions between the client and the server are performed from here.

Virtual Session Manager/Runtime Application Data

The virtual session manager retrieves data from the runtime file.

Virtual Session Manager/Virtual Message Box

The virtual message box is created by the virtual session manager (or the virtual panel) and is managed by the virtual session manager.

Virtual Session Manager/API

The virtual session manager starts Java code using the *ObjectCalling* Java interface. The API uses a set of methods to manipulate the client session such as creating new panel sessions, switching sessions, creating panels, or changing the contents/properties of the virtual components.

Virtual Session Manager/Host Session Manager

The virtual session manager uses a set of methods to control the host session manager such as creating a new host session, setting a host session in foreground, and stopping a host session.

Virtual Session Manager/Virtual Session

The virtual session manager and the virtual session have a very tight connection. The methods called by the session manager are typically: create, clientUpdated, hostUpdated, isCreateRequired, isUpdateRequired, fireCreate, fireUpdate and dispose.

API/Runtime Application Data

The API can directly access parts of the runtime application data.

API/Host Session Manager

The API can start/stop host sessions. It can also communicate directly with the host session through the host session manager. This includes methods such as sending strings and retrieving data from the host session screen.

Virtual Session/Host Session Manager

The virtual session maps a virtual session to a host session by changing to the appropriate host session.

Virtual Session/Virtual Panel

The virtual session typically creates and disposes of virtual panels, calls methods such as clientUpdated, hostUpdated, fireCreate and fireUpdate.

Virtual Panel/Host Session Manager

The virtual panel requests the host session manager for the current host session interface. This interface is then used to access or set the host required data.

Virtual Panel/Host Session

The virtual panel communicates with the host session that it receives from the host session manager. The communication is typically setting/retrieving a host field and sending keystrokes.

Virtual Panel/Virtual Message Box

The virtual panel causes a virtual message box to cause a transaction to the client by directing it to the virtual session manager.

Virtual Panel/Virtual Components

The virtual panel forwards calls from the virtual session such as fireCreate, fireUpdate, hostUpdated, and clientUpdated. The virtual component issues the method call setUpdated to the virtual panel. This causes a fireUpdate to be called later to the virtual components.

Virtual Components/Runtime Panel

The virtual component will retrieve its base data from its respective NetPhantom component in the panel part of the runtime file.

Virtual Components/Host Session

The virtual components request or set host fields and send keystrokes to the host session that is retrieved from the virtual panel (the virtual panel retrieves the host session from the host session manager).

Runtime Application Data/Runtime Panel

The runtime application directs which runtime panel is to be used depending on the currently matching host screen or a particular panel ID.

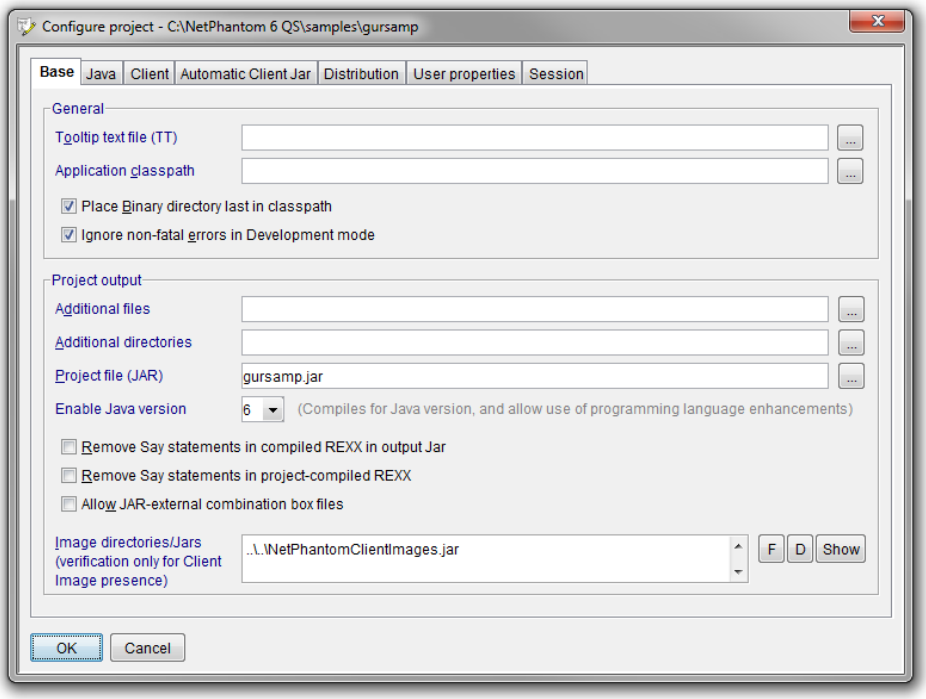
23 Building an Application

To build a NetPhantom application, a project must be defined (or imported). The project structure holds all the application resources and references.

The NetPhantom Editor (and Eclipse IDE, if running in integrated mode) allows for these resources to be developed and maintained. At the end of development, the application can be built into a distribution.

23.1 NetPhantom Project

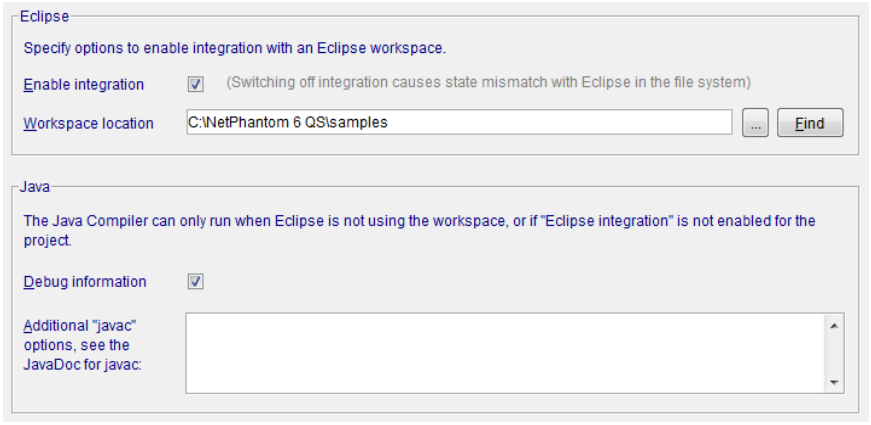
There are a number of settings for the NetPhantom project.



The Base page is used mainly to specify the structure of the project and how it should be built into a distributable file.

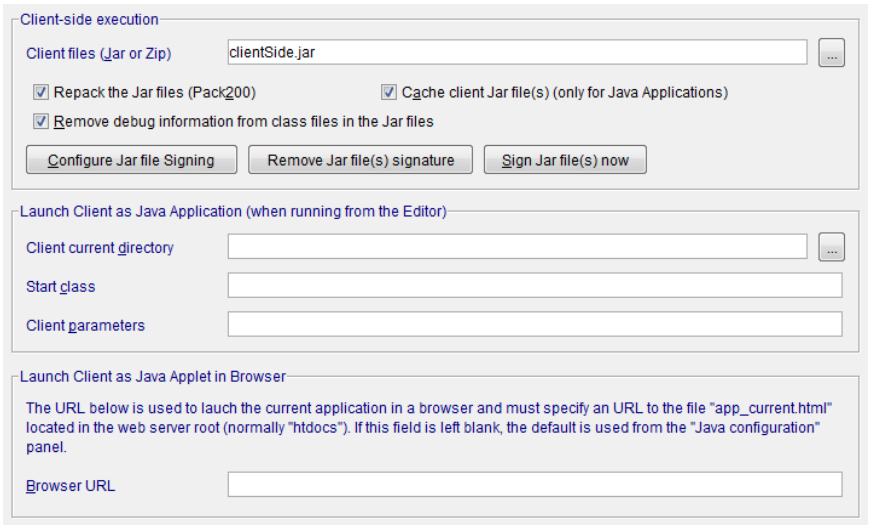
Entry	Description
Tooltip text file (TT)	The file that holds the tooltip texts for the project (optional).
Application classpath	A list of jar files and directories that is included in the project classpath. These files will be included in the project file when creating a distribution. (optional)
Place Binary directory last in classpath	This gives the opportunity to specify how the Java classes are loaded when running the application.
Ignore non-fatal errors…	If selected, do not interrupt the loading of a project that contains errors. This is so that projects under development or conversion should be possible to open and amended. It is still not possible to build a distribution while a project contains errors or is incomplete.

Project file (JAR)	The name of the project file. That is the .jar file that contains all components necessary to execute the corresponding application(s).
Compression level	A number that indicates how much the project file should be compressed. The available numbers are between 0-9, where 0 means no compression (fast process/big project file) and 9 means maximal compression (slow process/small project file).
Additional files	A list of files that should be included in the application distribution.
Additional directories	A list of directories that should be included in the application distribution.
Project file (JAR)	The name of the project file. That is the .jar file that contains all components necessary to execute the corresponding application(s).
Enable Java version	The application will be compiled for the indicated Java version.
Remove Say statements in compiled REXX in output Jar	If checked, all trace statements will be removed in the compilation of REXX code when a distribution is built. Allows for more efficient execution.
Remove Say statements in project compiled REXX	If checked, all trace statements will be removed in the compilation of REXX code when the application is launched in the editor. Allows for more efficient execution.
Allow JAR-external combination …	If selected, this project can make use of combination box files that reside outside the project file. This is to make it possible to change the runtime behavior of the application for this type of controls.
Image directories/Jars …	The items in the specified lisr are searched for image references. This is only used for the verification step of the application before creating a distribution.



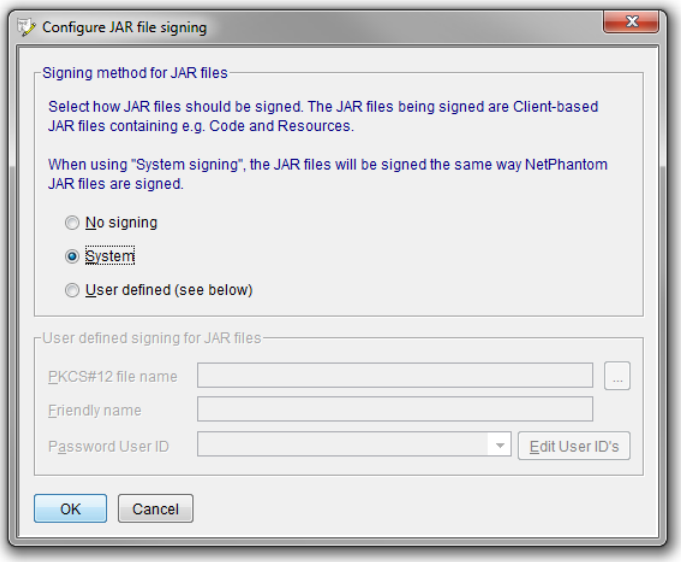
The Java page is used to specify if and how the Editor should be integrated with Eclipse.

Enable integration	If checked, the NetPhantom Editor will try to establish contact with an Eclipse IDE. For details about this integration, please see the *NetPhantom Eclipse* document.
Workspace location	A specification of which Eclipse workspace to use in integration is selected.
Debug information	If the Editor is running without Eclipse integration, REXX and Java will be compiled by the Editor. This option indicates that debug information should be generated in this compilation.
Additional "javac" options …	Additional Java compilation options used when the Editor is doing the compilation of REXX and Java.



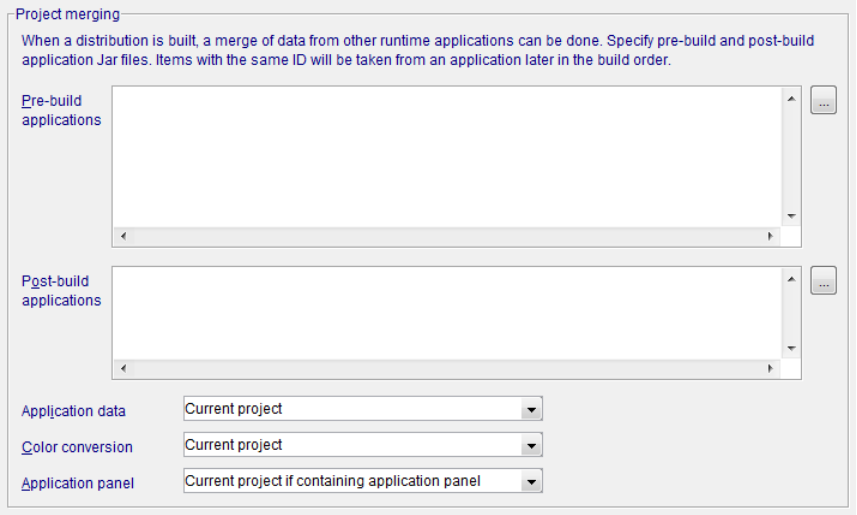
Entry	Description
Client files	Resource files (.jar or .zip) files that are transferred to the client when executed. These resources can then be referred to by the client. When a client is started at a subsequent time, a check is made to verify the version of the file, if the file stored locally on the client file system has not been updated on the server, the file does not need to be transferred again and the local copy is used.
Repack the Jar files (Pack200)	If this option is checked, Jar files are packed using the Java gzip compressor.
Cache client JAR files	If this option is checked, all resource files are transferred immediately when the client is started. If not, these files are transferred only when referenced.
Remove debug information …	This option allows the class files to be stripped of debug information before they are put into a distribution. This reduces the size of the application file.
Configure Jar file Signing	Configuration of the signing of the client jar files.

Remove Jar file(s) signature	Any signed client files will have the Jar signature removed.
Sign Jar file(s) now	Sign all client files.
Client current directory	The file directory that is used as the base when executing the client.
Start class	The name of the class that should be called when the client is started.
Client parameters	Parameters that are read by the client at start-up.
Browser URL	The URL is used to launch the application in a browser.



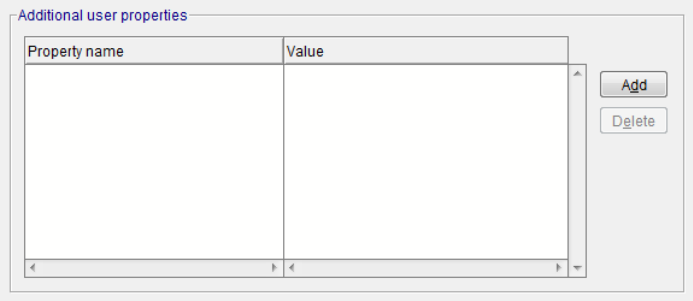
The JAR file signing page is used to specify how client jar files should be signed.

Entry	**Description**
Signing method	No signing System: Use the same form of signing as used for the NetPhantom Client JAR file. User defined: Use a specific certificate to sign the JAR file(s).
PKCS#12 file name	The local keystore file. This file contains the certificate to be used when signing along with the private key. It is sometimes called an identity file or personal exchange file.
Friendly name	The name or alias that the certificate is “called” in the PKCS#12 file.
Password User ID	The user id to use when signing the files. The user ID contains the password to use for the private key of the certificate. Click the button **Edit User ID’s** to manage the User ID’s and their passwords.



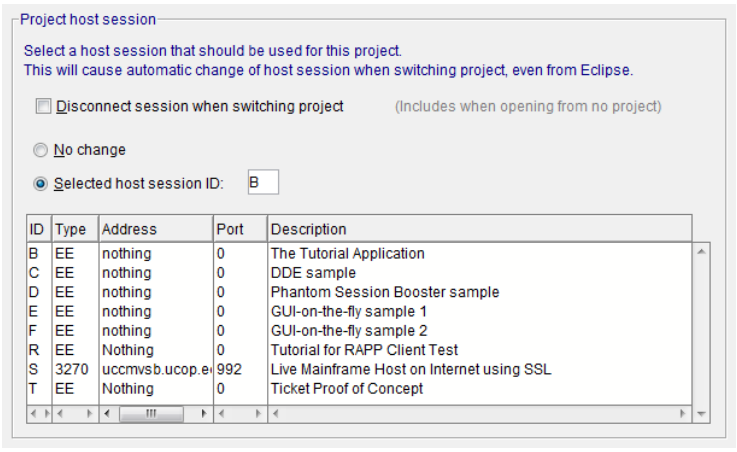
The Distribution page is used to specify if this project depends on other projects and, in that case, in which order they should be referenced.

Entry	Description
Pre-build applications	A list of applications that should be referenced before the current. The order in the list is significant.
Post-build applications	A list of applications that should be referenced after the current. The order in the list is significant.
Application data	Which application data to use, in the case the project is merged with other projects (pre- and/or post-build). The available alternatives are: • First project • First project containing application panel • Current project • Last project containing application panel • Last project The significant order is the list of pre-build applications, the current application and lastly, the list of post-build applications. The default choice is the “Current project”.
Color conversion	Which host color conversion table to use in case the project is merged. The available alternatives and default choice are the same as for the application data.
Application panel	Which application panel to use in case the project is merged. The available alternatives are: • First project containing application panel • Current project if containing application panel (if not, the first project containing such is used) • Last project containing application panel The default choice is “Current project if containing application panel”.



The User properties page is used to define additional user properties used when running the application.

Entry	Description
Additional user properties	A list of property names and corresponding values. These are available to use in the application program code. The way to access them is to read the application .ini file and access them under the section “User”. Note: If the project is merged, these values are handled in the following way: If the property name is defined in multiple applications, the value is taken from the current project definition if it exists there, otherwise it is taken from the pre- and post- build projects. In that case, definitions further back in the order (further “post”) will override previous definitions.

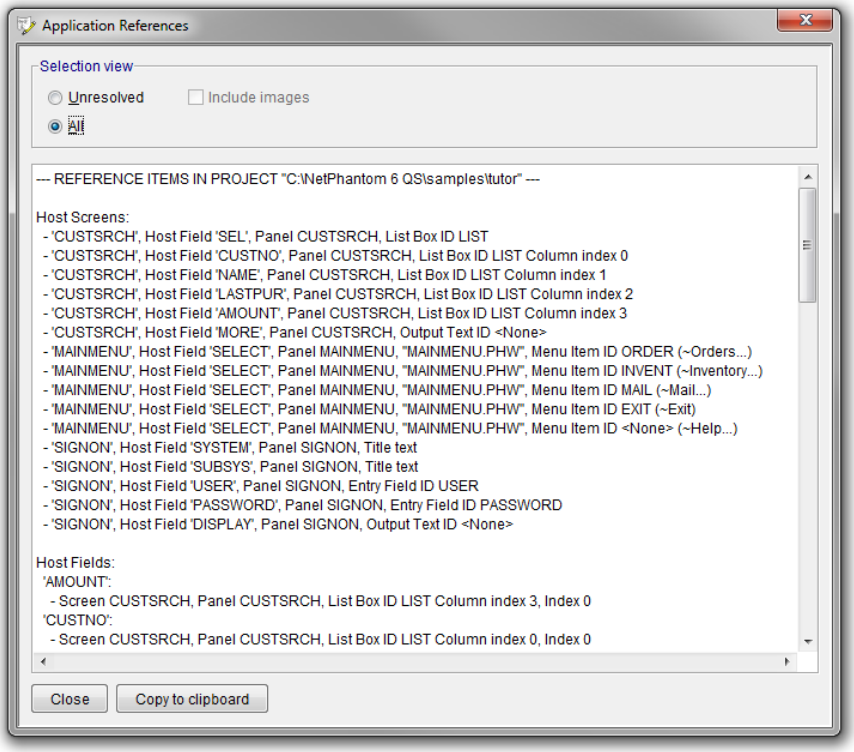


The Session configuration specifies which host session should be associated with the application.

Entry	Description
Disconnect session when switching project	If this option is checked, the host session will be terminated when the associated project is switched out. If int, the same session is reused when the project is opened again.
No change/Selected host session ID	The selected session ID (as defined in the server configuration). No change means that the current session is kept when switching to this project.

23.2 Application References

The references used in the current application can be listed by selecting **File – Application references …**



The listing can be done in two modes:

- All
- Unresolved

The listing contains all references in the application in the following order: Host Screens, Host Fields, Objects, Panels, Controls and Bar files. The list states the reference first and where it is references after.

The **Unresolved** option lists only unsatisfied references. This is a powerful tool when searching for errors in an application.

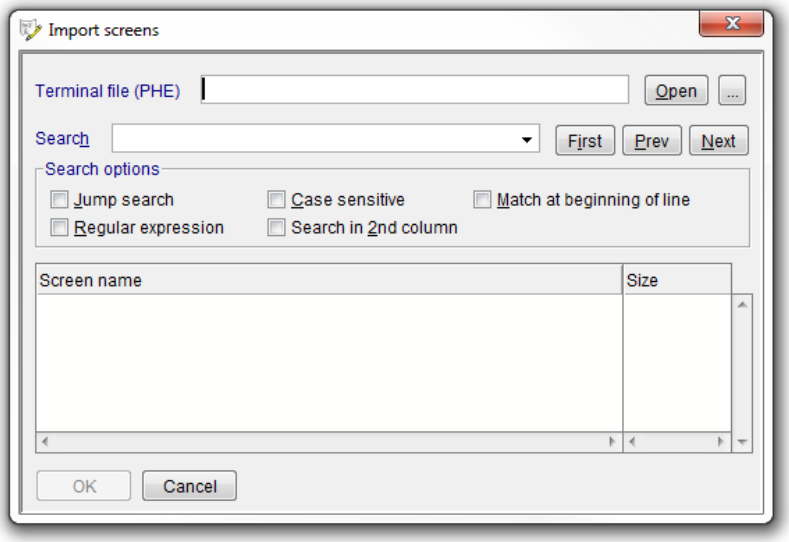
23.3 Compile distribution in NetPhantom Editor

Use the NetPhantom Editor menu item **File – Compile distribution** to create a self-contained file with all the required files for an application.

This menu item compiles the REXX code and verifies the integrity of the application. It then creates the application JAR file, the project distribution. This JAR file contains all required files (the text file(s) `.PHM`, the translation table `.PHX`, required toolbar and status bar files `.PHB`, the Help ID table file .PHH and so on), all into the corresponding runtime file (`.jar`).

23.4 Import Screens

Host screens can be imported from external projects. This can be done by selecting **File – Import screens…**



The terminal file (.PHE) is selected first. After that several screens can be selected and added to the screen definitions of the current application.

A search tool is available to make the selection of screens easier.

23.5 Import Existing Project

An existing or older version of a NetPhantom application can be imported as a project in NetPhantom 6 by chosing **File – Import existing project …** This process is described in detail in the *NetPhantom Eclipse* document.

Please note that when a project has been converted to version 6, it cannot be run on older versions of the NetPhantom Server, nor can it be opened in an older Editor.

23.6 Import Current Project in Eclipse Workspace

The current project can be imported and opened as a project in Eclipse by chosing **File – Import current project in Eclipse workspace …**This process is described in detail in the *NetPhantom Eclipse* document.

23.7 Unsupported REXX Functions

The following list of REXX functions are not supported by NetPhantom:

HostSetScreen, HostSendPassword	Used by the Phantom Processor program in Phantom Hurricane 2.x.
HostLockSession, HostDisable	No terminal screen is used.
GetInput	Used for Phantom Processor in Phantom Hurricane 2.x.
RxVarSave, RxVarRestore, RxVarDelete	REXX variables are too tied to the REXX language.

23.8 Troubleshooting REXX Code

During the conversion of REXX code to NetRexx and compilation to Java Byte Code, there may be occasions where this will fail.

REXX is an interpretive language that only checks the code it is running, whereas the conversion process checks it all. The conversion process will fail for the following reasons:

- Missing END statement.
- Missing OTHERWISE statement in SELECT.
- An unsupported instruction is encountered (INTERPRET, PUSH, PULL, QUEUE, DROP, CALL ON/OFF, etc).

The REXX source files are in the `rexxsrc` directory in the project structure. In the conversion process, the generated files (.java and .nrx) files are put in the `gensrc` directory. This enables debugging of the REXX code.

If the conversion process fails, check all the `.rexxerr` files available (all error files are deleted after a successful conversion, so any error files remaining after a conversion process will contain some error text). These files are also located in the `gensrc` directory.

Upon successful compilation, the class files are put in the `bin` directory.

REXX Source Sample

```
/* REXX:

   CK010
   Included Msg:CRT CMD
   Special for panel ACK010.
   If user is NET*, send CLEAR when PF5 is pressed
   Called from a push button on the panel
*/

Parse Arg Id, Msg, Str

Select
  When Msg='CRT' Then Call PanSetCtlStyle id, 2
  When Msg='CMD' Then
    Do
      If Substr(HostGetFld('','USERID'),1,3) = 'NET'
      Then Call HostSend '@C' /* Send CLEAR */
      Else Call HostSend '@5' /* Send F5    */
    End
  Otherwise
End
Return 0
```

NetRexx Converted REXX Source Sample

Notice the original line numbers to the left. This line number may be useful when tracking down REXX code problems.

```
/*  --  */ package css.rexx; import se.entra.phantom.server.NetRexxMigration;
/*  --  */ class CK010 extends NetRexxMigration
/*   10 */   method start($arg1, $arg2, $arg3) public signals Exception
/*  --  */     ID$local$ = 'ID'
/*  --  */     MSG$local$ = 'MSG'
/*  --  */     STR$local$ = 'STR'
/*  --  */     goto$ = '$start'
/*  --  */     loop label goto$ while goto$ <> 'label$0'
/*  --  */       select
/*  --  */         when goto$ == '$start' then
/*  --  */           do
/*   10 */             parse $arg1 ID$local$
/*   10 */             parse $arg2 MSG$local$
/*   10 */             parse $arg3 STR$local$
/*   12 */             select
/*   13 */               when MSG$local$="CRT" then
/*   13 */                 PanSetCtlStyle(ID$local$,"2")
/*   14 */               when MSG$local$="CMD" then
/*   15 */                 do
/*   16 */                   if (HostGetFld("","USERID","0")).substr("1","3")="NET" ..
```

```
/*   17 */                    HostSend("@C","0","0","0")
/*   16 */                  else
/*   18 */                    HostSend("@5","0","0","0")
/*   19 */                end
/*   20 */              otherwise
/*   12 */            end
/*   22 */            return "0"
/*  --  */          end
/*  --  */       end
/*  --  */     end
/*  --  */     return ''
/*  --  */ -- Starter method
/*  --  */ method start($__$s$__$=string,$__$i$__$=int,$__$st$__$=string) signals Ex..
/*  --  */   super.start($__$s$__$,$__$i$__$,$__$st$__$
```

24 NetPhantom HTML Integration

NetPhantom HTML integration allows NetPhantom to deliver an HTML interface to web-browsers without the need for loading the Java applet; this gives quick access to simple applications that do not need a full graphical user interface. HTML is the perfect solution for an Internet presence and for giving your legacy applications a lightweight interface.

24.1 Technology Overview

The HTML interface runs as a layer above the existing NetPhantom application, allowing an easy progression to this new format. Each HTML file references a NetPhantom panel that is displayed for the client thanks to the special NetPhantom web application function. The NetPhantom Server links the panels to the appropriate pages, then dynamically compiles the pages with the data received from the host and connects each control on the HTML page to the corresponding control on the NetPhantom panel.

24.2 Design Potential

With this HTML interface many techniques can be used to make the interface very interactive and eye pleasing. Industry standard design technologies such as CSS, DHTML and client-side scripting (JavaScript, VBScript) may be used, giving the interface designer a good selection of tools to work with. By using Cascading Style Sheets (CSS), the fonts, design and colors in an entire application can be changed immediately without the need to edit hundreds of HTML files, while interface interactivity may be produced with JavaScript.

24.3 Limitations

As with every technology, web applications have their limitations. Since they are based on HTML, their disadvantages are the same as those of HTML and revolve mainly around visual discrepancies and differences between different web browsers (Microsoft Internet Explorer versus Firefox, Opera, etc,). In addition, NetPhantom functionality and interactivity is not available in HTML. Therefore, the main purpose of the web application solution is to make smaller host solutions widely available on the Internet.

24.4 Overview

The NetPhantom Web Server and HTML Integration provides the speed and functionality needed to serve a client with normal HTML data and resources such as GIF images, as well as to handle integration of an HTML document with the AS/400 or mainframe host system.

NetPhantom HTML Integration

As opposed to NetPhantom Java Client panels that are created with the NetPhantom Editor, an HTML editor such as FrontPage or DreamWeaver is used to create the HTML interface. Special tags are placed in the HTML document that direct NetPhantom to replace them with data coming from the host, special variables, or even NetPhantom Server-centric code (such as REXX-converted code).

Data input to the host application or server-code is done by means of HTML forms. Forms in HTML support the following graphical components (but have other names in HTML, the terms below are NetPhantom-related):

- entry fields
- multiple-line entry fields
- combination boxes
- tables

- push buttons
- images

The tables in HTML enable usage of NetPhantom list boxes that can be line-selectable and/or editable.

NetPhantom Editor

To develop a web application, a NetPhantom application built with the NetPhantom Editor is required. Each HTML page in the web application must have a corresponding NetPhantom panel.

Restrictions for NetPhantom HTML Integration

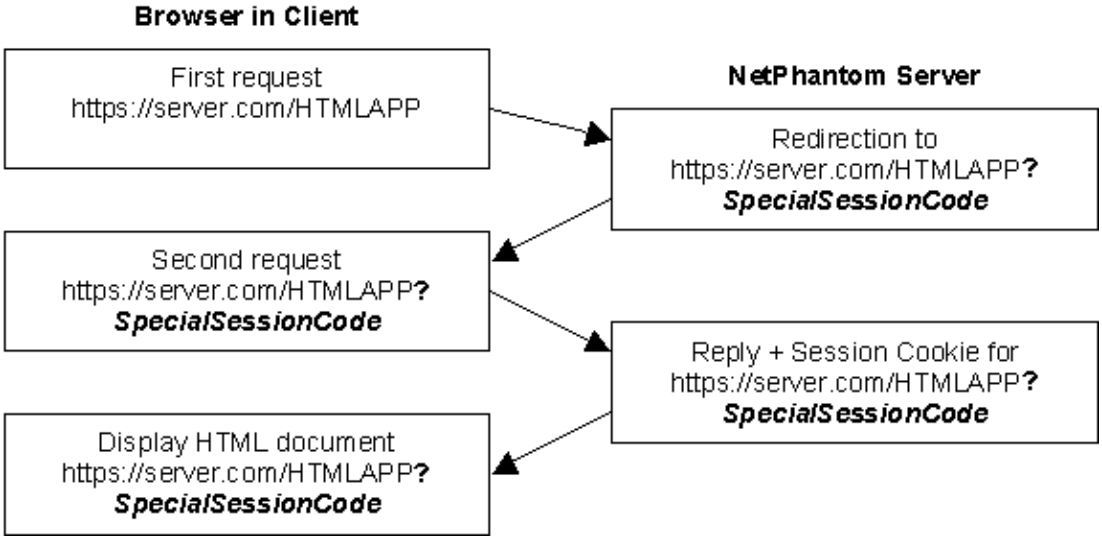
This section lists the restrictions imposed on NetPhantom's HTML Integration.

- When using HTML with NetPhantom, no terminal emulator exists; thus "falling" into the emulator will cause the client session to be closed.
- All screens in an application must therefore be identified and a panel must somehow be linked to the host screen. Stand-alone panels can also be used when no host screen exists.
- The HTML version of NetPhantom will display an HTML document for each panel, with the name `panelid.html` in a subdirectory of the application directory called `webApplicationName/html`. The NetPhantom panels don't need to be very fancy, unless the NetPhantom Java Client is used in parallel.
- For all actions concerning NetPhantom application usage, i.e. push buttons in the HTML form or special hypertext links, a connection to a push button or a menu item is required.

The HTML documents created for the panels can refer to host fields, text files, etc., as well as to the GUI controls in the panel.

Session Cookies

NetPhantom HTML applications use cookies for session identification. This setting is optional but is highly recommended. The cookie contains a session identification string that is uniquely created for each new session. This cookie is a "session cookie" and is not stored on the end-user machine. If the browser window is closed, the cookie will be destroyed.



Initial flow of data traffic when starting a NetPhantom HTML Application. The SpecialSessionCode looks like "d41f9f407754087877bfa3bfb329a6c16a6c2300".

Browser Buttons "Back" and "Forward"

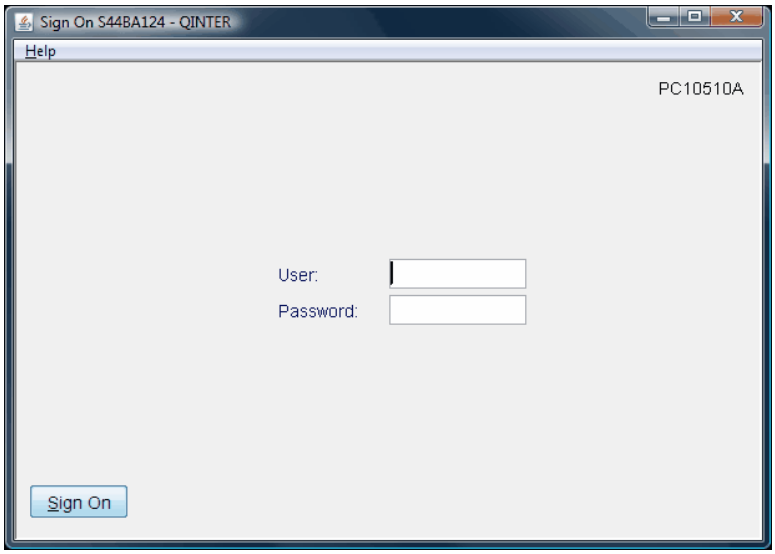
As the HTML application is normally subject to the presentation logic (what panel is displayed, etc) of the underlying host system, the buttons "Back" and "Forward" have no functionality.

If a user press one of these buttons they may end up out of sync with the host system. To remedy this, click the Refresh button and the session cookie will be used to refresh the HTML panel in the browser.

24.5 The Web Application

An HTML solution starts with the creation of a NetPhantom panel connected to a host screen. In this example, the host Sign On screen will be used. This screen contains host field connections (a user field, a password field, and a static text in top right corner) and the control button SIGNON that sends Enter to the host. Each control must be defined with a control ID to establish the appropriate connections.

The panels in this example are from the NetPhantom Tutorial. Controls that were not given a control ID in the tutorial have been modified and assigned an ID for the HTML sample.

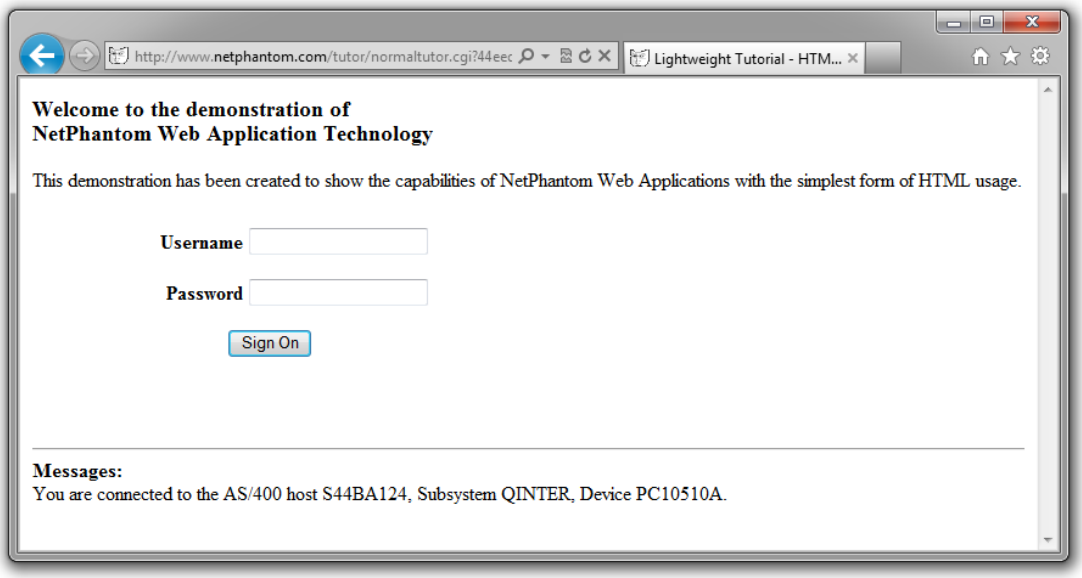


This is a panel inside the NetPhantom Panel Editor.

The HTML file may now be created with any editor. NetPhantom identifies the area in the file to parse and modify by the `<FORM>` tag. Each control must have its attribute name set to correspond with the NetPhantom panel control ID. For example, the button `<INPUT type="submit">` must have the name attribute set to the control ID of NetPhantom *Sign On* button.

The original HTML source:

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
<meta http-equiv="Content-Language" content="en-us">
<title>Lightweight Tutorial - HTML Demo - Sign On</title>
</head>
<body bgcolor="#FFFFFF">
  <h3>Welcome to the demonstration of<br>
      NetPhantom Web Application Technology</h3>
  <p>This demonstration has been created to show the
     capabilities of NetPhantom Web Applications with the
     simplest form of HTML usage.
  <form method="POST">
    <table border="0" width="390" height="56">
      <tr>
        <td width="170" height="40"><p align="right"><b>Username</b></td>
        <td width="206" height="40">
          <input type="text" name="USER" size="20">
        </td>
      </tr>
      <tr>
        <td width="170" height="40"><p align="right"><b>Password</b></td>
        <td width="206" height="40">
          <input type="password" name="PASSWORD" size="20">
        </td>
      </tr>
      <tr>
        <td width="390" height="40" colspan="2">
          <p align="center"><input type="submit" name="SIGNON">
        </td>
      </tr>
    </table>
    <p> </p>
  </form>
  <hr><b>Messages:</b>
  <br>You are connected to the AS/400 host <@SYSTEM@>,
  Subsystem <@SUBSYS@>, Device <@DISPLAY@>.
</body>
</html>
```



The lightweight tutorial displayed in Internet Explorer.

When the user starts the browser and connects to the Web Application, the NetPhantom application is started and connection to the host is established. The Sign On panel is loaded and the source HTML is combined with data from the NetPhantom panel to produce the final HTML, which is sent as a reply to the client browser. In other words, the design of the HTML stays the same, but the content reflects the information from the NetPhantom panel.

The HTML source that NetPhantom sends to the browser:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
    "http://www.w3.org/TR/WD-html-in-xml/DTD/xhtml1-strict.dtd">
<HTML>
    <HEAD>
        <META http-equiv="Content-Type" content="text/html;
              charset=iso-8859-1">
        <META http-equiv="Content-Language" content="en-us">
        <TITLE>Lightweight Tutorial - HTML Demo - Sign On</TITLE>
    </HEAD>
    <BODY bgcolor="#FFFFFF">
        <H3>Welcome to the demonstration of<BR> NetPhantom Web
            Application Technology</H3>
        <P>This demonstration has been created to show the
            capabilities of NetPhantom Web Applications with the
            simplest form      of HTML usage.   </P>
        <FORM method="POST" action="http://ghost.entra.se/tutor/
normaltutor.cgi?c6bbaced13e75e2be096696401193237c355c300">
            <TABLE border="0" width="390" height="56">
                <TR>
                    <TD width="170" height="40"><P align="right">
                            <B>Username</B>
                        </P>
                    </TD><TD width="206" height="40"><INPUT type="text"
                      name="USER" size="20" maxlength="10">
                    </TD></TR>
                <TR>
                    <TD width="170" height="40"><P align="right">
                            <B>Password</B>
                        </P>
                    </TD><TD width="206" height="40"><INPUT
                      type="password" name="PASSWORD" size="20"
                      maxlength="10">
                    </TD></TR>
                <TR>
                    <TD width="390" height="40" colspan="2">
                      <P align="center">
                         <INPUT type="submit" name="SIGNON" value="Sign On">
                        </P>
                    </TD></TR>
            </TABLE>
            <P> </P>
        </FORM>
        <HR>
        <B>Messages:</B>
        <BR>You are connected to the AS/400 host S44BA124, Subsystem
        QINTER, Device PC10510A.
  </BODY>
</HTML>
```

Tutorial Samples in HTML

On a normal NetPhantom installation, two samples are provided that are both built on the NetPhantom Tutorial. One sample is called *Lightweight* and one is called *Fancy*. They are available under *Web Applications* in the menu bar of the index web page.

24.6 Step-by-Step Quick Start

1. Create a NetPhantom application with the NetPhantom Editor or use an existing application and make sure that Control IDs are set on all controls that should operate in the web application.

2. Perform a **Compile distribution**.

3. Use **Server Administration** to add the application to the server and set the host session.

4. Select the **Server – Configure web server** menu item. For more information, please read the manual in the *Configure Web Server* section.

5. On the **General** page, make sure that **Enable web server** is checked.

6. Select the **CGI** notebook tab. Make sure the Web Application module is present, if not, set the name to `HTMLAPP`, the Java class name to `se.entra.phantom.server.http.HtmlApplicationCGI` and press the **Add** button.

7. Select the **Web Applications** notebook tab. Specify a **Name** of your choice (*NameOfWebApp*, see 9 below), the **CGI name** is `HTMLAPP` (see step 6), the appropriate **Host session** and select the **Applications** (from step 3). Press the **Add** button.

8. Select the **Resources** notebook tab. Specify a **Name** (that will define the request URI in the browser, for example `MYWEBRES`), a **Description**, set the **Resource type** to `WA - Web Application`, and the **CGI/file name** (which should be the name of the web application set in step 7) and select an **Access control**. Press the **Add** button.

9. Create a directory directly under the runtime directory with the name ***html*** (for example `<project base directory>\`*`NameOfWebApp`*`\html`). The *NameOfWebApp* is the name given to the Web Application resource in step 7. Put all HTML files in this directory (see step 10).

10. Create HTML files that have the same name as the NetPhantom panel IDs. *The HTML file names must be lowercase, and the extension must be ".html", NOT ".htm"*. The Editor menu item Edit – Copy HTML can be used to copy controls from the NetPhantom panels and paste them in any editor or a Web page designer. See *Copy HTML* below.

    Use the *Control Reference* section of this chapter to modify the HTML controls that are linked to your NetPhantom controls by the Control IDs as needed.

    NOTE! Make sure all references to NetPhantom controls are placed within a `<FORM>` tag.

11. Restart the Server.

12. Start a web browser and specify the address `[`*`hostname`*`]/[`*`resourceName`*`]`, for example `localhost/mywebres`.

Location of Files

For HTML applications to run properly, it is essential that files be located in the correct directory or that redirection is used. Here we describe the required directory structure followed by a description of resource redirection. Please note that "…" below indicates the location of the NetPhantom directory. In this example we assume that you have installed NetPhantom in `C:\NetPhantom 6\`, which is the default directory for the *NetPhantom Jump-Start for Developers*.

Let's also assume that you have:

1. a NetPhantom Application called `YOURHTML.jar` compiled in the `…\NetPhantom 6\yourhtml` directory,

2. a Web Application named `YOURWEBAPP` (this name will be appended in lower case to the location of the runtime file `.jar` directory followed by `\html`, i.e. `yourwebapp`**`\html`**),

3. a Web Resource named **`/yourdir/subdir`**`/webapp.cgi` that is a `WA – Web Application` resource type mapped onto `YOURWEBAPP` (see #2 above),

4. the Web Server document root is located at …\NetPhantom 6\htdocs.

The following directories and files would be used:

…\NetPhantom 7\yourhtml**\yourwebapp\html**
Location of the HTML files that map onto the NetPhantom panels (see #1 above). Let's also assume that these HTML files refer to GIF images with a relative URL, e.g. **images/**logo.gif. These would then be placed in …\NetPhantom 6\yourhtml**\yourwebapp\html\images**.

…\NetPhantom 7\yourhtml**\yourwebapp\html\images**
The location of the images used in the design phase of the HTML documents for the NetPhantom panels (see above).

…\NetPhantom 7\htdocs**\yourdir\subdir\images**
The location of all external references to e.g. images of the HTML files as well as other resources (see #3 and #4 above). The files in the directory …\NetPhantom 6\yourhtml**\yourwebapp\html\images** would normally have to be copied into this directory (instead of copying files, see *Web File Resource Redirection* below).

…\NetPhantom 7\htdocs**\webapps**
Location of the GIF files used for disabled checkboxes and radio buttons and the file messagebox.html. This directory is fixed by NetPhantom and cannot be changed (it is always relative to the document root).

Web File Resource Redirection

In order not to duplicate a lot of files from files used within the HTML documents, the NetPhantom Web Server allows creation of **File Resource** redirections. This can be done on a file or a directory level. The directory level would typically be used for Web Applications.

Following the example above, instead of copying files from the directory …\NetPhantom 7\yourhtml**\yourwebapp\html\images** to …\NetPhantom 7\htdocs**\yourdir\subdir\images**, you can create a redirection as follows:

In the **Server – Configure web server** menu item, select the tab **Resources**. Enter the following items:

Name	The name of the resource /yourdir/subdir/images. This is the name of the directory in which the browser expects to find the images (if you append the document root this becomes \NetPhantom 7\htdocs**\yourdir\subdir\images).**
Description	E.g. File redirection for images and other resources for the YOURWEBAPP application.
Resource type	Select FS - File System
CGI/File name	\NetPhantom 6\yourhtml\yourwebapp\html\image
Access control	Select an access control allowing normal access to the images.

Then click the **Add** button.

24.7 NetPhantom Markup Tags in HTML

Special NetPhantom HTML markup tags (elements) can be inserted inside the HTML document. These elements indicate modifications of the HTML document that the server will perform before sending it to the client.

All documents with the HTML file extension `html` will be pre-parsed when running with a Web Application. This prevents subsequent parsing of the document, thus allowing a faster reply from server (if the parsed document is still cached). The HTML parser supports all documents up to and including HTML version 4 (“loose” or “transitional”, not “strict” or XHTML).

For performance reasons a separate, size-tunable cache is available to optimize server memory. This is required because parsing the HTML document is only done at first load request (or a subsequent load request if the document content has changed, i.e. the file timestamp is not the same). Each client request for a parsed document will only cause the list of HTML elements or tags to render themselves in the appropriate way.

The elements identified and processed/changed/updated by NetPhantom are:

```
<!-- Comment -->
<%@INCLUDE file="relativeURL" [...]>
<variable>
<SCRIPT [...]>
<INPUT [...]>
<FORM [...]>
<SELECT [...]>
<TEXTAREA [...]>
<OPTION [...]>     inside a <select> tag only
```

Supported HTML elements

The following elements are handled as valid HTML elements:

```
A, ABBR, ACRONYM, ADDRESS, APPLET, AREA, B, BASE, BASEFONT,
BDO, BIG, BLOCKQUOTE, BODY, BR, BUTTON, CAPTION, CENTER,
CITE, CODE, COL, COLGROUP, DD, DEL, DFN, DIR, DIV, DL, DT,
EM, EMBED, FIELDSET, FONT, FORM, FRAME, FRAMESET, H1, H2, H3,
H4, H5, H6, HEAD, HR, HTML, I, IFRAME, IMG, INPUT, INS, KBD,
LABEL, LEGEND, LI, LINK, MAP, MENU, META, NOFRAMES, NOSCRIPT,
OBJECT, OL, OPTGROUP, OPTION, P, PARAM, PRE, Q, S, SAMP,
SCRIPT, SELECT, SMALL, SPAN, STRIKE, STRONG, STYLE, SUB, SUP,
TABLE, TBODY, TD, TEXTAREA, TFOOT, TH, THEAD, TITLE, TR, TT,
U, UL, VAR.
```

Invalid HTML Document

If an HTML document is found to be invalid for some reason, the server will cause the NetPhantom Client Session to be stopped (if one is present) and event log entries will describe the situation. If no NetPhantom Client Session is present, the document is sent as is to the client.

> **Note:** A NetPhantom Client Session is a session with the host system running e.g. the Mainframe or the AS/400.

Server Side Include – SSI

Any HTML document can contain a Server Side Include (SSI) of a part of another HTML document or even text file.

To include another document in an HTML document, use the following element:

```
<%@INCLUDE file="relativeURL"% convert="yes|no">
```

If the document included has the file extension HTML (i.e. `html` or `htm` regardless of character case), this file is assumed to contain at least partially valid HTML contents and will therefore be parsed.

If the document included has another file extension (e.g. `txt`), it will not be parsed. This can be used to speed up processing if the included document really contains HTML elements but doesn't need to be parsed (e.g. because there are no NetPhantom Variables in it).

Optionally the document can be converted into HTML tags for special characters (e.g. Yen/Euro signs, national characters). This is done with the option `convert="yes"`. By default, this option is `"no"`.

HTML-Included Servlets or CGIs

Any HTML document can contain a reference to an HTML-included Servlet/CGI. This is done using the following format:

```
<%@include cgi="name" [param1="value1"] ...>
```

The HTML-include CGI must be specified in the web server in the CGI section (becomes entries such as `<name> = className` in the `[CGI]` section of `server.ini`).

These CGIs must implement one of the two Java interfaces:

```
se.entra.phantom.server.http.IncludeCgiPrintInterface
se.entra.phantom.server.http.IncludeCgiHtmlResourceInterface
```

For more information, see the *NetPhantom Java API*.

24.8 Variable Substitution in HTML Text

Variables should have the format:

```
<@specialVariable@>
```

to be substituted as the following example illustrates:

```
<tr>
  <td><font size="1" face="Verdana">Server local date</font></td>
  <td><font size="1" face="Verdana"><@**DATE@></font></td>
</tr>
<tr>
  <td><font size="1" face="Verdana">Server local time</font></td>
  <td><font size="1" face="Verdana"><@**TIME@></font></td>
</tr>
```

The example could be replaced by the following when sent to the client:

```
<tr>
  <td><font size="1" face="Verdana">Server local date</font></td>
  <td><font size="1" face="Verdana">01.02.2000</font></td>
</tr>
<tr>
  <td><font size="1" face="Verdana">Server local time</font></td>
  <td><font size="1" face="Verdana">12:34:56</font></td>
</tr>
```

If variable tags appear elsewhere than in the main HTML document text, they are not substituted or handled in any way.

24.9 Variable Substitution in HTML Tags

Normally HTML tags have the following format:

```
<tagname param1 param2 = "quoted text" param3 = value>
```

Variable substitution is only performed inside a valid HTML tag when located inside quoted text. The format for the variables is:

```
@specialVariable@
```

No variable substitution is performed elsewhere in an HTML tag or element.

Example:

```
<applet codebase = "." Port="@**SERVERPORT@">
```

could perhaps become:

```
<applet codebase = "." Port="789">
```

24.10 Variable Substitution in Scripts

A script can be of e.g. JavaScript type inside an HTML document:

```
<script language='JavaScript'>
<!--
function resize()
  {
  var rs=@#RESIZE@;
  if ( rs!=0 )
    document.applet.setSize(document.body.clientWidth,
                            document.body.clientHeight)
  }
// -->
</script>
```

Inside a script, variable substitution is enabled from the opening `<script…>` tag to the ending `</script>`. No special treatment of text in or outside quoted text is performed. In the sample above, the line `var rs=@#RESIZE@;` is replaced by the HTTP session variable `resize`, so the line would perhaps become `var rs=1;`.

24.11 Variables in HTML Documents

Variables can be incorporated into HTML documents as HTML elements. The NetPhantom Variables have the format:

```
<variable>
```

See the table below for valid variables.

General Variables

The following variables can be used whether the HTML document comes from a NetPhantom application or not.

`@**DATE@`	Current date in long format.
`@**TIME@`	Current time in long format.
`@**UTC@`	Current date and time in Universal Time Convention (UTC) format.

@**MODDATE@	The last modified date of the current resource in long format.
@**MODUTC@	The last modified date and time of the current resource in Universal Time Convention (UTC) format.
@**PROGDIR@	NetPhantom directory.
@**PROGVER@	NetPhantom program version string.
@**PROGBUILD@	The program version in the format 6.60 (Build *nnnn*).
@**PROGVER2@	The program version according to HTTP standard `product comment`, e.g. NetPhantom/7.10 Java/11.2_08
@**OSVER@	Operating System version.
@**OSNAME@	Operating System name.
@**JAVAVER@	Java version.
@**JAVAVENDOR@	Java vendor.

HTTP Related Variables

The variables below are related to the HTTP session.They can be used whether the HTML document comes from a NetPhantom application or not.

@**AGENT@	The user agent of the requesting client, i.e. the browser name and version.
@**ERRCODE@	The current error code of the HTTP request/reply. A value of zero indicates that nothing has been sent to the client yet and no error has occurred. Otherwise, this value is 200 (HTTP OK).
@**ERRTEXT@	The current error text of the HTTP request/reply or empty string for no error.
@**ERREXTRA@	Extra error information available for certain error codes.
@**CLIENTPORT@	The client port number.
@**CLIENTADDR@	The client host address.
@**CLIENTNAME@	The host name of the client or the IP address as `nnn.mmm.ooo.ppp` if none is found.
@**SERVERPORT@	The port of the server that the client is accessing.
@**CTRLSERVERPORT@	The port of the controller server that the client is accessing. Used with load balancing.
@**SERVERADDR@	The host address of the server.

`@**SERVERNAME@`	The host name of the server or the IP address as `nnn.mmm.ooo.ppp` if none is found.
`@**CTRLSERVERNAME@`	The host name of the controller server for load balancing.
`@**ADMINPORT@`	The port number used for remote server administration, e.g. the remote command line utilities.
`@**FULLSRVNAME@`	The full server's name including the protocol, i.e. `http[s]://fullsrvname`.
`@**FULLCTRLSRVNAME@`	The full server's name of the controller server including the protocol. Used with load balancing.
`@**HTTPMETHOD@`	The method used to handle the request (GET or PUT).
`@**PROTOCOL@`	The protocol used (evaluates to `http` or `https`).
`@**URI@`	The Uniform Request Identifier for the resource last requested by the client.
`@**URIPARAMS@`	The parameters passed along with the resource request, i.e. the string after the '?' of the Uniform Request Identifier. If no parameters are used, an empty string is returned.
`@**SSLLEVEL`*index*`@`	The data encryption level of SSL being used. Evaluates to an empty string if no SSL is used. Index Description 1 "0" or "1" if encrypted or not. 2 "0" or "1" if strong encrypted (>=128 bits) or not. 3 The number of effective bits for encryption. 4 The name of the cipher being used.
`@**SSLSTRING@`	Evaluates to `SSL` if the connection is using SSL, otherwise to an empty string. This variable is useful to specify the name of the client .jar file, e.g. `"NetPhantomClient@**SSLSTRING@.jar"`.
`@**SSLUSE@`	Evaluates to 1 (one) if this is an SSL connection to the client or 0 (zero) if not, typically used with NetPhantom Client parameter `SSL:@**SSLUSE@`.
`@#`*`HTTPVariable`*`@`	Evaluates to an HTTP user variable named *HTTPVariable* that certain CGIs create, update and/or use. See the Java class `se.entra.phantom.server. http. HttpSession` for more information. The variable text is converted into HTML valid text.
`@#-`*`HTTPVariable`*`@`	Evaluates to an HTTP user variable named *HTTPVariable* that certain CGIs create, update and/or use. See the Java class se.entra.phantom.server.http.HttpSession for more information. The variable text is used unchanged, i.e. it can contain HTML tags that will be placed in the final document.

NetPhantom Application Variables

The variables listed below can only be used when running against a NetPhantom application. If the variables cannot be found or the document is not running against a NetPhantom application, the variable evaluates to an empty string.

`@Hostfield[:line]@`	Inserts the contents of a host field. Optionally specify an index (one-based) of the line in the host field.
`@%Hostfield[:line]@`	Runs the contents of a host field through the translation table and inserts the result. Optionally specify an index (one-based) of the line in the host field.
`@*TextID@`	References a text string in the text file (`.PHM`) with ID *TextID*.
`@-ControlID@`	References a text string of the *ControlID*. See Control ID References in the Phantom User's Guide and Reference.
`@-ControlID:[x].[y]@`	References a text string of the *ControlID* that must be a list box, a combo box or a spin button. The `X` and `Y` parameters reference the column and line in the list box. If the Y parameter is omitted, i.e. `ControlID:X.`, the text string is taken from the list of the combo box or the spin button. If the `X` parameter is omitted, i.e. `ControlID:.Y`, the text is taken from the header of the list box at column `Y`. Both the `X` and `Y` parameters are one-based (e.g. first line = 1).
`@=Globalvar@`	The contents of a NetPhantom global variable.
`@**FOCUS@`	Returns the Control ID of the NetPhantom control that currently has focus. This is used for setting focus via JavaScript to the correct control in an HTML page.
`@**MW@`	Message Waiting, 5250 only. `1` if MW flag is on in the OIA, otherwise `0`.
`@**PANELID@`	The ID of the panel that caused the display of the HTML document.
`@**SCRID@`	The ID of the current screen.
`@**RTCRT@`	Timestamp of the runtime file (`.jar`).
`@**APPDIR@`	Application directory.
`@**APPNAME@`	Application name (`.jar` or `.PHA` file without extension).
`@**CLIENTID@`	The client session identifier.

@**APPACTION@	Contains the base of an HTTP hyperlink for an application. It consists of the protocol (HTTP or HTTPS), the server's name or address, optionally a port number and the name of the Web Application resource followed by a unique session identifier. This variable is typically used for the ACTION in a FORM or for hyperlinks in normal HTML documents.
@**CAPPACTION@	As with @**APPACTION@, but the @**CAPPACTION@ variable is used for HTML frame documents.
@**MSGBOXTITLE@	The title of the current message box
@**MSGBOXTEXT@	The text of the current message box.

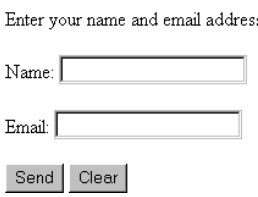
24.12 NetPhantom Web Application

Data entry between the client and server is done using an HTML FORM.

The following shows the listing of a typical form document:

```
<html>
<head>
<meta http-equiv="Content-Type"
content="text/html; charset=iso-8859-1">
<title>Who are you?</title>
</head>
<body bgcolor="#FFFFFF">
<p>Enter your name and email address and press &amp;quot;Send&amp;quot;.</p>
<form action="http://my.server.com/reg" method="POST" name="form">
    <input type="hidden" name="hidden" value="value">
    <p>Name: <input type="text" size="20" name="name"></p>
    <p>Email: <input type="text" size="20" name="email"></p>
    <p><input type="submit" name="sendButton" value="Send">
    <input type="reset" value="Clear"></p>
</form>
</body>
</html>
```

and the result in a browser would look like this:



Normal HTML Text

The normal text in HTML documents is processed for *NetPhantom Variables* inside a tag such as:

```
<h1>Category - <@CATEG@></h1>
<p>The person "<@NAME@>" has the email <@EMAIL@> and the phone
<@PHONE@>.</p>
```

would be replaced by:

```
<h1>Category - Power User</h1>
<p>The person "Im A. User" has the email power@user.com and
the phone 55512345.</p>
```

if the host fields CATEG, NAME, EMAIL and PHONE in the NetPhantom application has the values "Power User", "Im A. User", "power@user.com" and "55512345" respectively.

NetPhantom Variable Substitution

In HTML mark-up tags or elements, there may be quoted text. If this quoted text is not removed or substituted with other contents due to the nature of the element (see the HTML elements below), *NetPhantom Variable Substitution* can be used.

The variables are listed in *Variables in HTML Documents*.

However, substitutions take place between two at-signs (`@`) if this string corresponds to a *variable*. In order to write the at-sign, anyway, simply specify it twice (`@@`). If an invalid variable string is encountered, no action will be taken.

The examples

```
1: <INPUT type="submit" value="@*IDSEND@">
2: <INPUT type="submit" value="@**NOTHING@">
```

would become:

```
1: <INPUT type="submit" value="Envoyer">
2: <INPUT type="submit" value="@**NOTHING@">
```

if the text file of the NetPhantom application with text ID `IDSEND` contains the text `Envoyer`. In the second example, there is no variable named `"@**NOTHING@"` defined, so no action is taken.

24.13 Character Set

The default character set (or codepage) for HTML documents and Forms is ISO-8859-1 (or Latin 1). This can be changed to another default ISO character set in the Server Administration Program (or in the *server.ini* file). The requirement is that the server and the clients (browsers) agree to use the same charset.

However, if the HTML page with the Form should be used internationally, there might be a need for solution that handles all kinds of characters. This solution is called UTF-8 and is a byte encoded version of the Unicode charset. Characters specified in UTF-8 can easily be converted into Unicode and then converted back again.

The complete solution is to set two attributes in the form tag, as seen in bold below:

```
<form name="FormSample" method="post"
 action="http://server_name/status.cgi"
 enctype="multipart/form-data"
 accept-charset="utf-8">
```

The default content type for forms is *application/x-www-form-urlencoded*. When a user presses a Submit button, the browser should send back form names and values to the server. These field names and values are placed in the header field data and are separated by the "&" sign.

multipart/form-data is a content type created to be able to send binary data in forms, such as binary files. The content type is described in RFC 2046. There is no support for files in NetPhantom, but this format makes it possible for the browser to send any kind of character data to the server.

The **accept-charset** attribute specifies to the browser, what character sets the server can accept. The support for this attribute is a bit limited in the browsers, but it can still be used. We want to use the **utf-8** character set (UTF-8), which means that all possible characters from any character set can be entered by the user and identified in the server.

All HTML files in the NetPhantom server are parsed and NetPhantom searches for the form tags with **enctype="multipart/form-data"**. If this kind of form tag is found, two new hidden fields are dynamically added to the form before it is sent to the client. The reason for adding these fields is to make NetPhantom understand in which character set (**charset**) the field data is sent back from the browser. These fields are handled by NetPhantom and are not included in the HeaderField hash table accessed by the CGI.

The first hidden field is **definition_charset** which contains the same information found in the **accept-charset** attribute in the form tag, this should in this case be **utf-8**. The second hidden field is **definition_charset_teststring** which is used as a work-around to verify that the browser is actually using the UTF-8 character set.

24.14 The FORM Element

The syntax of the FORM element with NetPhantom should be:

```
<form action="URL" method="POST" ...>
```

If NetPhantom locates a FORM element, it will automatically be modified so that appropriate action and method are performed. Only these element attributes are affected, other attributes, valid or not, will remain.

The FORM element always requires an ending `</FORM>` tag. All data input elements (such as INPUT, SELECT) listed below are only handled inside the beginning and ending FORM elements.

24.15 Focus in HTML Forms

In a NetPhantom application focus is initially set according to cursor position in the host, or if no corresponding control is available, to the first editable control. In HTML focus can be tricky as even the type of browser can affect where focus is set. JavaScript can be used ensure that focus is set to the correct control in the HTML document.

```
<script LANGUAGE="JavaScript"><!--
function setFocusField(setTo,formName)
  {
  if (setTo=="")
    return false;
  if (document.all)
    {
    var comp=document[formName];
    if (comp)
      {
      if (comp[setTo])
        {
        if (comp[setTo].length>1)
          {
          for (var i=0; i<comp[setTo].length; i++)
            {
            if (comp[setTo][i].checked)
              {
              comp[setTo][i].focus();
              break;
              }
            }
          }
        }
      else
        comp[setTo].focus();
      }
    }
  return true;
  }
// -->
</script>
```

The script requires two items to work: the control name and the name of the form in which the control lies. The `@**FOCUS@` variable is used to get the name of the control that currently has focus in the NetPhantom panel. This name is the same for the control in the HTML document. You must also be sure that the form has been given a name. For example:

```
<form action="URL" method="POST" name="MYFORM" ...>
```

Style sheets can be used to simplify naming of forms.

24.16 The SCRIPT Element

When inside a SCRIPT element, normal variable substitution is done as described in *Variables in HTML Documents*. However, the substitution of strings between two at-signs (`@`) will also take place if this string corresponds to a *variable*. To write the at-sign, simply specify it twice (`@@`).

This is valid between the beginning `<SCRIPT>` and ending `</SCRIPT>` tags. If an invalid variable string is encountered, no action will be taken.

There is no processing of any other elements or tags inside SCRIPT.

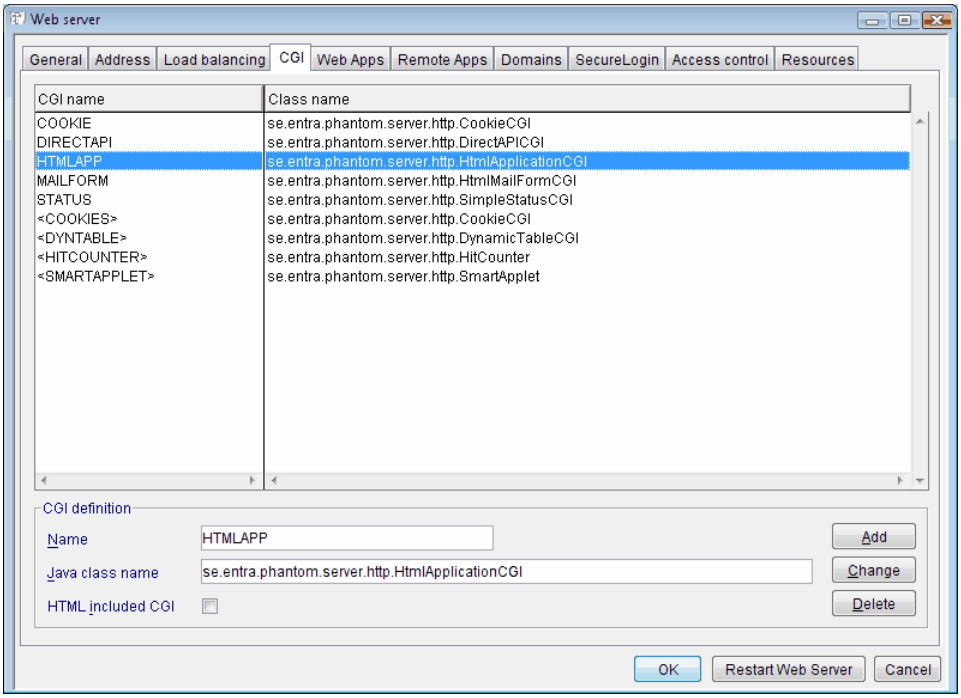
24.17 Configuration Reference

For configuration of HTML Integration and Web applications, the NetPhantom remote administration console is used. Execution of a web application usually involves defining and verifying the settings in three sections:

1. CGI settings
2. Web Application settings
3. Resources settings

CGI Settings

The CGI resource settings are defined in the **Server – Configure – Web server** menu item under the **CGI** notebook tab. Normally, the default settings are enough. This section is provided for special cases in which you need to replace the web application module.



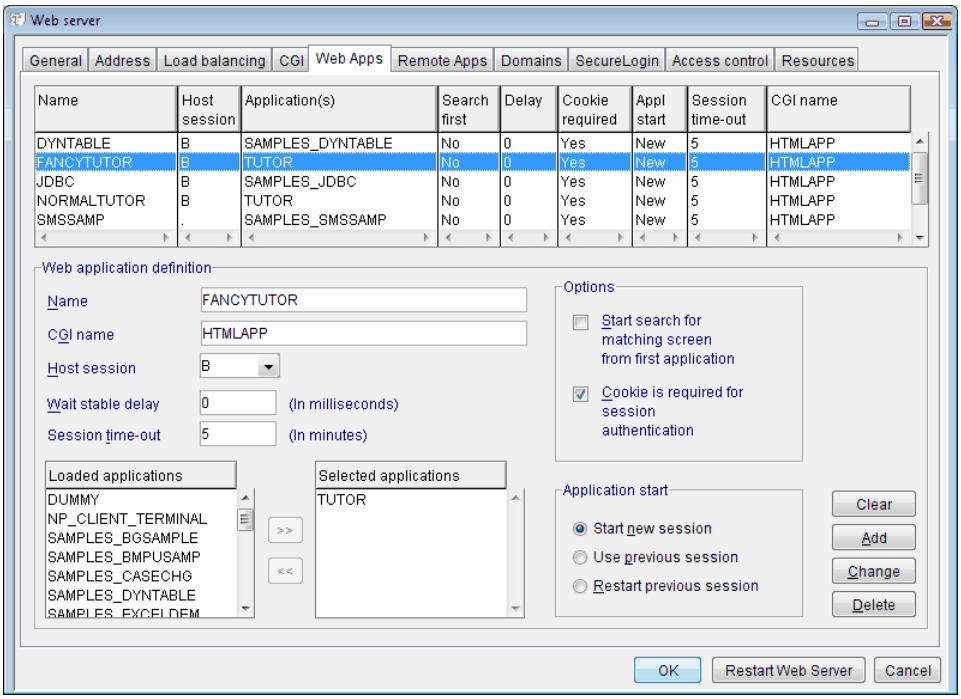
The CGI tab under the Configure Web Server Menu.

A CGI should be defined with the following settings:

Name	The name identifies the CGI to NetPhantom. This should be set to `HTMLAPP`.
Java class name	The name of the Java class file that holds the logic of the CGI. The standard web application module is: *`se.entra.phantom.server.http.HtmlApplicationCGI`*
HTML included CGI	This defines whether the CGI can be included in an HTML document using the `<%@INCLUDE>` tag or not. *This option is not checked for Web applications.*

Web Application Settings

The web application settings are defined using the **Server – Configure – Web server** menu item under the **Web Applications** notebook tab.



The Web Applications tab under the Configure Web Server Menu.

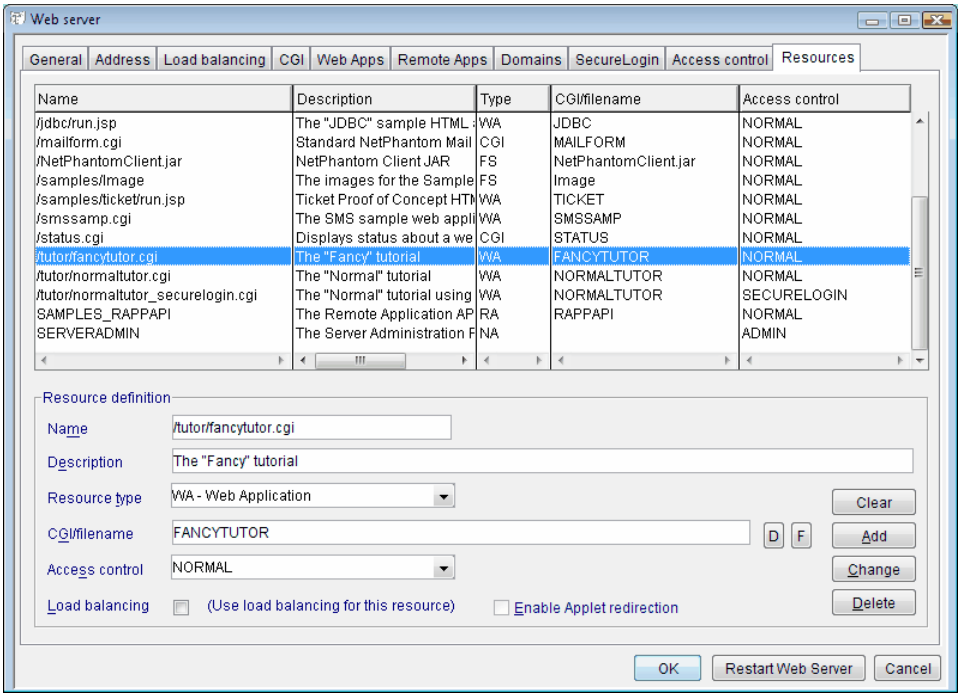
A web application should be defined with the following settings:

Name	The name of the web application.
CGI name	The CGI setting references the CGI application to be run, in this case, it is the web application module.
Host session	The host session to use for the application.
Wait stable delay	The Wait stable delay is an optional wait period to make sure that the host has delivered all information to the NetPhantom Server. This value is specified in milliseconds.
Session time-out	The session time-out is the time that the NetPhantom Server will wait for the client to respond before disconnecting from the host. This value is specified in minutes.
Loaded Applications	A list of applications that are loaded on the NetPhantom Server.
Selected Applications	The list of applications that will be used for the Web Application. Use the arrow buttons to transfer from the loaded application list to this list.
Start searching for matching screen from first application	Start searching for a matching panel from the first application in the list instead of the current application.

Cookie is required for session authentication	A cookie (a locally stored information file) is used to identify the client to the server. The server assigns a unique identifier for each client session passed to the browser when the application starts (and only then). This identifier is sent from the browser with each new request (POST and/or GET), thus enabling the server to check the authenticity of the client with the session. **Note:** It is highly recommended that you use this option! It guarantees that another intruding (malicious) client cannot "steal" the session. Combined with the usage of SSL, it then becomes the preferred, most secure client/server communication (that is stateless as HTTP is in its design). If the browser window is closed, the cookie will be destroyed.

Resource Settings

The HTML resource settings are defined in the **Server – Configure – Web server** menu item under the **Resources** notebook tab.



The Resources tab under the Configure Web Server Menu.

A resource for the web application should be defined with the following settings:

Name	The name of the web resource. This name will define the URI for the browser. For example, setting the name to `/apps/htmlapp` makes the URI to the application `http[s]://servername/apps/htmlapp`.
Description	An optional description of the web resource to facilitate administration.
Resource type	The resource type must be set to `WA - Web Application`.
CGI/File name	For web applications, this references the Web application to be run.

Access control	Allows you to apply secure access control to your web resource. See the *Access Control* section in the manual.
Load balancing	Check this option if load balancing should be done for the resource. See *Chapter 8 NetPhantom Load Balancing* and *14.5 Configuring the Web Server - Load Balancing* for more information.

24.18 Control Reference

This chapter requires knowledge of Hypertext Mark-Up Language (HTML).

The <FORM> tag is the delimiter to where NetPhantom will parse the HTML file and read all tags, attributes, and text. Five tags are used as connection mechanisms with NetPhantom: the <INPUT> tag, the <SELECT> tag, the <TABLE> tag, the <TEXTAREA> tag and the <A> tag. Each tag must be located inside the <FORM> section and reference the appropriate control ID.

All the usual HTML tags and attributes are supported and can be set additionally to the NetPhantom specific attributes. They can be found in any HTML reference guide.

HTML to NetPhantom References

NetPhantom reads all tags, attributes and text that are placed inside the form tag. Four tags are used for connection to NetPhantom: `INPUT`, `SELECT`, `TABLE` and `A (href)`. All tags that should be connected to NetPhantom controls must be placed within the `FORM` tag and contain an ID for reference to the control.

HTML	NetPhantom
`INPUT type="submit"`	Push button or menu item.
`INPUT type="text"`	Entry field or output text.
`INPUT type="checkbox"`	Checkbox.
`INPUT type="radio"`	Radio button.
`SELECT`	Combination box or spin button.
`TABLE`	List box.
`TEXTAREA`	Multiple line entry field (MLE), entry field or output text.
`A href="…"`	Push button or menu item.

Description of Terms Used in this Section

Server	NetPhantom Server that also run as HTML application and web server.
HTML source	The template file that is used together with the NetPhantom panel to create the HTML output.
HTML output	The resulting HTML page that is displayed in the client browser.
Tag	HTML tag.
Attribute	HTML tag attribute.
Panel	NetPhantom panel.
Control	NetPhantom control, such as entry field or push button.
CTLID	Reference to any NetPhantom control.

The Controls

For all controls in a form except for the table, all attributes not processed by NetPhantom are passed on to the client.

All controls always follow the visibility and enabled state of the connected NetPhantom control. If a control is hidden, the entire HTML tag is automatically removed. For disabled controls, see the section *Disabled Attribute* below for more information.

In most cases, a connection to a mainframe or an AS/400 exists for NetPhantom control. But this might not always be the case. All HTML form elements are always connected to NetPhantom controls. Very often the case is that they in turn are connected to some host field. Again, this is not at all a requirement. A standalone application in NetPhantom can be built in such a way that no host requirement exists.

Push Button – Input Submit

This tag is used to create an interactive push button that will activate the target NetPhantom control.

HTML tag name	`INPUT`
Valid NetPhantom controls	Push Button Menu Item
Required tag attributes	`type="submit" name="`*`CTLID`*`"`
New NetPhantom attributes	`NpText="HTML" | "NP"` Defines if the submit text should be taken from the HTML source or the NetPhantom Push button. The default value is `"NP"`.
HTML visual behavior	Creates a submit push button.

Examples	`<INPUT type="submit" name="`*`CTLID`*`" **NpText="NP"**>`
	
	`<INPUT type="submit" name="`*`CTLID`*`" **NpText="HTML"** value="Submit form">`

Generated HTML	`<INPUT type="submit" name="`*`CTLID`*`" **value="Cancel"**>`
	
	`<INPUT type="submit" name="`*`CTLID`*`" value="Submit form">`

The text of the push button will be generated automatically from the text shown on the NetPhantom push button referenced by *CTLID*. The hotkey or mnemonic character symbol (~) and any eventual image references will be removed from the text. By using the `NpText` attribute set to `"HTML"`, the HTML submit text (in `value="text"`) can be used instead of the NetPhantom push button text.

There are two special *CTLID*s that can be used to refer to the page buttons in a list box; to refer to the page up button specify *CTLID* as `"REF=`*`LISTID`*`+TYPE=PGUP"` and the page down button as `"REF=`*`LISTID`*`+TYPE=PGDN"`.

Note: Always use a submit button to enable the form data that the user inputs to reach the server – hyperlinks never send the form data!

Entry Field – Input Text

This tag creates an editable input field that will be updated with the contents of the host.

HTML tag name	`INPUT`
Valid NetPhantom controls	Entry field Output text
New NetPhantom attributes	`NpDisableToText` This option converts the input tag to pure text instead of setting it disable if the input tag is connected to a NetPhantom output text or to a disabled entry field.
Required tag attributes	`type="text" name="`*`CTLID`*`"`
HTML visual behavior	Creates an entry field or static text depending on connected NetPhantom control.
Example	`<INPUT type="text" name="`*`CTLID`*`" size="`*`SIZE`*`">`
Generated HTML	`<INPUT type="text" name="`*`CTLID`*`" size="`*`SIZE`*`"` **`maxlength="`*`PHMAXSIZE`*`" value="Phantom field contents"`**`>`

The control size is defined in the HTML source while the maximum size and value are set automatically by the settings in the NetPhantom entry field. If the *CTLID* references an output text control, the `INPUT` tag in the HTML output will be replaced by text data only.

Multiple Line Entry Field – Textarea

This tag creates an editable multiple line entry field (MLE) that will be updated with the contents of the host.

HTML tag name	`TEXTAREA`
Valid NetPhantom controls	Multiple line entry field Entry field Output text
New NetPhantom attributes	None.
Required tag attributes	`name="`*`CTLID`*`"`
HTML visual behavior	Creates a multiple line entry field or static text depending on connected NetPhantom control.

Example	`<TEXTAREA name="`*`CTLID`*`" cols="20" rows="3">`
	`This text will be removed`
	`by NetPhantom`
	`</TEXTAREA>`

Generated HTML	`<TEXTAREA name="`*`CTLID`*`" cols="20" rows="3">`
	`Net`**`Phantom field contents`**
	`</TEXTAREA>`

The control size in width and height is only specified in the HTML document, as opposed to the tag `<INPUT type="text">` above. NetPhantom only replaces all text between the `<TEXTAREA>` and the `</TEXTAREA>` tags. If the *CTLID* references an output text control, the `TEXTAREA` tag in the HTML output will be replaced by text data only. Be careful, the visual behavior differs greatly between input field and text data!

Checkbox – Input Checkbox

This tag creates an HTML checkbox control.

HTML tag name	`INPUT`
Valid NetPhantom controls	Checkbox
Required tag attributes	`type="checkbox" name="`*`CTLID`*`"`
New NetPhantom attributes	None.
HTML visual behavior	Creates a checkbox.
Example	`<INPUT type="checkbox" name="`*`CTLID`*`">`
Generated HTML	`<INPUT type="checkbox" name="`*`CTLID`*`" **checked**>`

The checkbox "checked" state is reflected from the NetPhantom checkbox state.

Radio Button – Input Radio

This creates a radio button.

HTML tag name	INPUT
Valid NetPhantom controls	Radio button
Required tag attributes	`type="radio" name="`*`GROUPNAME`*`" value="`*`CTLID`*`"`
New NetPhantom attributes	None.
HTML visual behavior	Creates a radio button.
Example	`<INPUT type="radio" name="RADIOGROUP" value="`*`CTLID`*`">`
Generated HTML	`<INPUT type="radio" name="RADIOGROUP" value="`*`CTLID`*`" **checked**>`

The checked option is set from the NetPhantom radio button checked status.

The name specified in the *GROUPNAME* is used by the browser in the same way NetPhantom handles radio button groups, i.e. when a radio button in the same group is selected, all other radio buttons belonging to the same group are unselected.

Note: The difference between radio buttons and other input types is that NetPhantom connected control name is set in the value attribute and the name attribute is used as a radio button group name.

Combination Box – Select

Creates a combination box.

HTML tag name	SELECT
Valid NetPhantom controls	Combination Box Spin Button
Required tag attributes	name="*CTLID*"
New NetPhantom attributes	None.
HTML visual behavior	Creates a combination box.
Example	<SELECT name="*CTLID*"> **This text will be removed by NetPhantom.** </SELECT>
Generated HTML	<SELECT name="*CTLID*"> **<OPTION>combo row 1** **<OPTION selected>Combo row 2** **<OPTION>combo row 3** </SELECT>

The select options (rows) are inserted from NetPhantom combination box contents. If the combination box is host converted, only the text line is shown, not the host value.

Listbox – Table

This creates a table of data from the host. The current implementation of the TABLE processing in NetPhantom does not allow usage of the COLGROUP tag. NetPhantom provides a mechanism to propagate the TD tag settings of the first row (or the template row) in the list to all the others. This also makes older browsers visually render the table better.

HTML tag names	TABLE, TR and TD
Valid NetPhantom controls	List box
Required <TABLE> tag attributes	id="*CTLID*"
New NetPhantom attributes for <TABLE> tag	NpSelection="RADIO" | "CHECK" | "LINK" | "NONE" Defines the selection type(s). "RADIO" is radio buttons to the left of the table for single selection, "CHECK" is checkboxes to the left for multiple selection and "LINK" is hyperlink for single selection. Note that hyperlinks do not send the form data to the server! "RADIO" or "CHECK" can be combined with "LINK" for both types of selection, e.g. "RADIO, LINK". The default value is "RADIO" for single selection lists and "CHECK" for multiple selection lists. The attribute is used together with the NpSelectAction and NpLinkColumn attributes, see below. NpSelectAction="*CTLID*" Button or menu to activate when selection is made, this attribute is used together with the NpSelection type "LINK" (see above). NpLinkColumn="1"-"nn" | "ALL" Specifies that the column index "1"-"nn" (must be a single value, e.g. "2") in the table should contain the selection hyperlink when the NpSelection="LINK" is selected. "ALL" sets that all columns should contain hyperlinks. The default is NpLinkColumn="1". If the selected column contains empty cells, an underscore will be displayed as hyperlink (unless overridden using CSS). NpColor="**HTML**" | "NP" The table color is taken from the HTML source or the NetPhantom list box definition. Default value is "HTML". NpColorConv="ON" | "**OFF**" Use NetPhantom color conversion. This option overrides NpColor="HTML" for color converted columns. The default is "OFF". Note that color conversion does not work on hyperlinks or when column is editable. NpCheckHostSelection="Yes" | "**No**" Option for selectable list box with radio button, checkbox, or hyperlink selection. Checks the list box selection field of each

	row to see if the selection for that row is enabled. This option is typically used for list boxes with selections every second row.

New NetPhantom Attributes for <TR> tag	`NpKeep="Header" | "Footer"`
	Creates one or more static headers/footers. All rows specified as `NpKeep="Header"` will be placed at the top of the table and all `NpKeep="Footer"` will be placed at the bottom of the table.
	`NpTemplate="Header" | "Row" | "Nextrow"`
	Specifies if the row will serve as a header template or a row template. If set to `Header`, the HTML header will override the contents of the NetPhantom list box header.
	For a description of `"Nextrow"`, see *Using Multiple Template Rows* below.

Optional <TD> tag attributes	`width="`*`WIDTH`*`"` `align="`*`ALIGN`*`"`

New NetPhantom Attributes for <TD> tag	`NpColumn="1-`*`nn`*`"`
	Allows you to change the order in which the NetPhantom columns are displayed in HTML.
	`NpPageColumnImages`
	This attribute specifies that a page up/down column should be created to the right in the list. The `NpPageDownSrc` attribute should also be set.
	`NpPageDownSrc`
	Relative file reference to the page down image to be used with the page up/down column. See `NpPageColumnImages`.

Example	`<TABLE id="`*`CTLID`*`">`
	`  <TR>`
	`    <TD width="20">Column 1</td>`
	`    <TD width="30" align="right">Column 2</td>`
	`    <TD width="10">Column 3</td>`
	`  </TR>`
	`  <TR>`
	`    <td width="20">Another line of data</td>`
	`    <TD width="30">This line is removed</td>`
	`    <TD width="30">. . .</td>`
	`  </TR>`
	`</TABLE>`

Generated HTML	`<TABLE id="`*`CTLID`*`">`
	`  <TR>`
	`    <TD width="20">Ph row 1 col 1</td>`
	`    <TD width="30" align="right">Ph row 1 col 2</td>`
	`    <TD width="10">Ph row 1 col 3</td>`
	`  </TR>`
	`  <TR>`
	`    <TD width="20">Ph row 2 col 1 </td>`
	`    <TD width="30" align="right">Ph row 2 col 2</td>`
	`    <TD width="10">Ph row 2 col 3</td>`
	`  </TR>`
	`</TABLE>`

Behavior

The table tag is not a typical form tag, but it can be placed within a form. If the table is specified within a form and has the ID attribute set to a NetPhantom list box control, then the table is filled using data from this NetPhantom list box.

In the HTML source, the <TABLE> tag is used as a template for the generated HTML output. Tag attributes such as background color and border size are reflected in the outputted table from the template row as described below.

When NpTemplate="HEADER" the first row will be kept intact (with nested tags, text, etc.), while the second row is used as the template to build the other rows. No header row is built from the NetPhantom list box control when this attribute is set.

A header row is not created if the header texts of all the columns in the NetPhantom list box control are empty.

When NpTemplate="ROW", if the list box contains headers, the first row will contain the headers from NetPhantom, and the subsequent rows will be filled with the list box data. A header row is not created if the header texts of all the columns in the NetPhantom list box control are empty.

NetPhantom reads all the list box rows (including the header) to get the contents of the table and the column alignment. The HTML table template determines how many columns should be created. For example, if the NetPhantom list box contains 4 columns and the HTML template contains 3 columns, the result will be the content of the first 3 NetPhantom columns. The <td> attribute NpColumn can be used to change the order of the columns. For instance, if you want to show NetPhantom's columns 4, 1 and 2 in that order, the template could look like this:

```
<td width="132" align="left" height="117" NpColumn="4">4</td>
<td width="132" align="left" height="117" NpColumn="1">1</td>
<td width="132" align="left" height="117" NpColumn="2">2</td>
```

If not specified, the NpColumn attribute will be set to 1 by default and will increase by 1 for each row. This means that the template could also look like this:

```
<td width="132" align="left" height="117" NpColumn="4">4</td>
<td width="132" align="left" height="117" NpColumn="1">1</td>
<td width="132" align="left" height="117">2</td>
```

If selection is allowed in the NetPhantom list box, a new left column with radio buttons will be created in the table by default. These radio buttons will select the correct line in the NetPhantom list box and allow the submit button to send the appropriate selection string and then the send key. In the case of a multiple selection NetPhantom list box, a checkbox will be generated instead to allow multiple rows to be selected. To disable table selection, set the table attribute NpSelectAction="NONE".

Selection in HTML may also be done by placing hypertext in one or all the columns of the table. This is a good solution for a single selection NetPhantom list box.

> **Note:** if hyperlinks are used, the form data will not be sent to the server! For more information, see the NpSelection, NpSelectAction and NpLinkColumn attributes in the TABLE tag.

A page up/down column has a similar appearance as a Windows scrollbar but will only handle page up and down through the list box. The column is typically inserted as the rightmost column in a table. The first row contains a page up image and the last row a page down image. The second row contains a non-breaking space and a rowspan between the second row and the next last row.

Both page up and page down images are defined in the template row but in different ways. The page up image is created as a real image reference under the <td> tag but the page up reference is placed as an attribute within the <td> tag. This means that the page up image is visual in the HTML editor but the page down is not. The copy to HTML functionality will add a dummy page down image to the last row so that it also is visual in the HTML editor.

The reason for having both definitions in the template row is to have better runtime performance.

```
<td NpPageColumnImages NpPageDownSrc="my_images/pgdn.gif"><img
border="0" src="my_images/pgup.gif"></td>
```

If the list box contains an editable column and the host field is not protected, all appropriate cells in the table column will be made editable using input tags. If the template row contains an input tag, this will be used as a template. NetPhantom always sets the maxsize, name and value attributes.

Using Multiple Template Rows

The purposes of multiple template rows are many. For example, you might want to set different colors on every other row. Another example is a subfile on the host that contains two different types of rows that should be displayed differently in the HTML page.

One row in the table template is usually set as template row, this is done with the attribute `NpTemplate = "Row"`. By setting another HTML row with `NpTemplate = "Nextrow"`, an additional row is set as template in this table. The result is that every odd numbered row uses the first (row) template, and every even numbered row will use the second (nextrow) template. By adding yet another "nextrow" row in the template, every third row will use this template, and so on.

The rule is that one `NpTemplate = "Row"` must be specified first followed by one or more `NpTemplate = "Nextrow"`. They will be set as a template in the order in which they appear.

Hyperlink

This links a hyperlink to a specific push button or menu item.

HTML Tag Name	`A`
Valid NetPhantom controls	Push button Menu Item
Required tag attributes	`NpControlID="CTLID"`
HTML Visual Behavior	Links a hyperlink to a NetPhantom control.

Example	`<A NpControlID="`*`CTLID`*`">`
	`  Hyperlink text`
	`</A>`

Generated HTML	`<A href="http://1.2.3.4/npapp.cgi?f04b16453+`*`CTLID`*`">`
	`  Hyperlink text`
	`</A>`

The *CTLID* is a reference to a NetPhantom push button or menu item. The hyperlink will inherit the style attributes, such as disabled and hidden from the NetPhantom control.

There are two special *CTLID*s that can be used to refer to the page buttons in a list box; to refer to the page up button specify *CTLID* as `"REF=`*`LISTID`*`+TYPE=PGUP"` and the page down button as `"REF=`*`LISTID`*`+TYPE=PGDN"`.

Note that when using hyperlinks, the form data is not sent to the server.

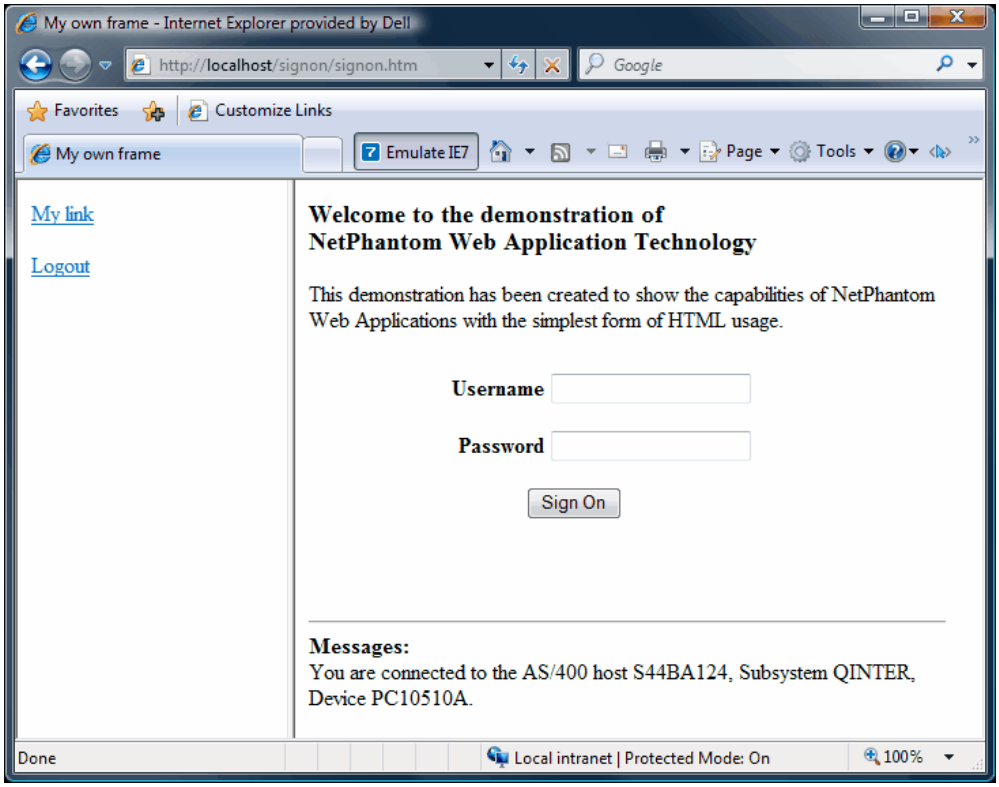
25.19 Using Frames

The tags <FRAMESET> and <FRAME> are used to build HTML frames. Usually, one main file contains the frame definition and references other HTML files with the contents of these frames.

Example of simple frame file with a menu area on the left side and a content area to the right:

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
<title>My own frame</title>
</head>
<frameset cols="200,*">
<frame name="menu" src="menufile.htm">
<frame name="main" src="mainfile.htm">
<noframes>
<body>
<p>This page uses frames, but your browser doesn't support them.
</body>
</noframes>
</frameset>
</html>
```

The <NOFRAMES> tag in the example contains the text to print if the browser doesn't support frames.



The lightweight tutorial with frames displayed in Internet Explorer.

In the example above, the left frame is used as a static menu, which in this case contains static hyperlinks connected to push buttons or menu item in the current panel. This means that these push buttons must exist on every panel in the application.

The right frame area is the NetPhantom HTML content, which is the same as when not using frames. From now on, we will call it the main area.

To connect a NetPhantom HTML application to frames, just connect the first panel in the application (typically the signon panel) to an HTML frame file instead of the ordinary HTML signon file. Then set one of the frames to reference the actual signon file. Except for this first frame file, all other HTML files connected to panels should be handled as usual.

For HTML integration with frames, the topmost <FRAMESET> tag contains some new NetPhantom attributes. All attributes must be set correctly; otherwise refresh and nested frames will not work properly.

- NpPanelConnection is a reference to the panel that loaded this frame file.
- NpStartFile is the first file for the panel content that is displayed in the main area, for example signon.htm. The actual file reference in the main frame is ignored.
- NpMainTag is the tag name of the main area.

Example of HTML integration frame file signon.htm:

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
<title>My own frame</title>
</head>

<frameset cols="200,*" NpPanelConnection="signon"
NpStartFile="signon_content.htm" NpMainTag="main">
<frame name="menu" src="menufile.htm">
<frame name="main" src="sigon_content.htm">
<noframes>
<body>
<p>This page uses frames, but your browser doesn't support them.
</body>
</noframes>
</frameset>
</html>
```

Example of menufile.htm:

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
<title>Menu</title>
<base target="main">
</head>

<body>
<p><a href="@**CAPPACTION@+BTOK">My link</a>
<p><a href=@**CAPPACTION@+BACK">Logout</a>
</body>
</html>
```

Note that base target is set to main so that the result of a hyperlink selection is displayed in the main frame area. The hyperlinks in the body section reference the NetPhantom variable CAPPACTION, which is a constant version of the APPACTION variable. The CAPPACTION variable is needed because the menu page never gets updated. The CAPPACTION variable is not recommended for hyperlinks in the main area.

Change Current Frame

Typically, only one frame is used for a NetPhantom application; it is initiated from the first panel in the application. It is possible to change the frame for a specific part of the application, but there are some things to consider.

As an example: the host system MYSYS contains 20 screens, 15 of them constitute the base part of the system and the other 5 are new. These two parts should have different frames.

The panel LASTBASE is the entry point into the new part, and the panel NEWPART is the first panel in the new part.

To create a new frame from the panel NEWPART, create a new frame file and set the <FRAMESET> NetPhantom attributes. When this HTML file is created, the current frame for the HTML integration will change into this frame file. But there is a problem; the newly created frame will be displayed within the main area as a sub frame instead of being the top frame. The only acceptable solution to this is to use a JavaScript in the head area of the frame file.

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
<title>My own frame</title>
<SCRIPT language="javascript">
if (self != top) top.location.href = self.location.href
</SCRIPT>
</head>

<frameset rows="50,*" NpPanelConnection="newpart"
NpStartFile="newpart_content.htm" NpMainTag="main">
<frame name="header" src="header.htm">
<frame name="main" src="sigon_content.htm">
<noframes>
<body>
<p>This page uses frames, but your browser doesn't support them.
</body>
</noframes>
</frameset>
</html>
```

Now the new frame will remain the current NetPhantom application frame, this means that when returning to the base part of the application this frame will still be the active one. To prevent this, create a frame file for the LASTBASE panel that has the same design as the signon frame file.

24.19 Various Functions

Browser Capabilities

It is to be expected that clients with old browsers will be using the web application**.**

To identify the different browsers, a text file containing settings called **BrowserIdentification.ini** exists. This file contains the **User Agent** strings of browsers that have certain capabilities. These capabilities are used by NetPhantom web applications.

The base section should contain two entries for each browser:

```
name_of_browser.id      = identification_string
name_of_browser.section = browser_capabilities_section
```

Specify the identification_string as a wild card string between commas.

All the specified criteria must match. Use the wild card "?" to replace a single character or "*" to replace any count of characters. If a string should NOT match, precede it with "!".

The browser_capabilities_section should contain the name of a section in the ini file that defines:

```
disable = boolean   (support for DISABLE attribute for input fields, etc)
label   = boolean   (support for <LABEL> tag)
button  = boolean   (support for <BUTTON> tag)
```

where boolean should be 0 or 1 (zero or one).

When a browser cannot be identified, it is assumed to have the lowest capabilities.

Currently, only Internet Explorer 5 or better Mozilla support HTML 4.0, e.g. the *Disabled Attribute for User Input Elements* (see below). Other browsers such as Firefox, Opera, and alike are also supported.

Example of `BrowserIdentification.ini`:

```
[Base]

; IE7.x
Ie7.id            = Mozilla/4.0*MSIE 7*
Ie7.section       = html4


;
;*************************************************************************
; *
; *
; *  BROWSER CAPABILITIES SECTIONS
; *
; *
;
;*************************************************************************
;
; Each section should have the settings:
;
;  disable = boolean   (support for DISABLE attribute for input
;  fields, etc)
;  label   = boolean   (support for <LABEL> tag)
;  button  = boolean   (support for <BUTTON> tag)
;
; where "boolean" should be 0 or 1 (zero or one).
;

[html4]
disable=1
label=1
button=1

[ie4]
disable=0
label=0
button=1
```

Disabled Attribute for User Input Elements

With HTML 4.0, most tags can be set with a very convenient disabled attribute for tags connected to disabled NetPhantom controls. When the browser doesn't support the disabled attribute (see *Browser Capabilities* above), the following action is taken:

HTML tag	Replaced by
`INPUT type="submit"`	Text.
`INPUT type="text"`	Text.
`INPUT type="checkbox"`	Image of disabled checkbox.
`INPUT type="radio"`	Image of disabled radio button.
`<SELECT>`	Text.
`<TEXTAREA>`	Text.

The images of the disabled checkbox and radio buttons are:

Image	File name
	checkboxDisable.gif
	checkboxDisableChecked.gif
	radioDisable.gif
	radioDisableChecked.gif

The images are loaded from the webapps directory under the document root (unless it has been redirected).

Using Label Tag for Hidden Controls

In HTML 4.0, the label tag can be used together with one of the supported control tags that doesn't have a label of its own. For example, when the label is used on a radio button, the user can press the label to select the radio button.

Radio button example:

```
<label for="radioid">
<input type="radio" name="ctlid" id="radioid">
Radio label text
</label>
```

If the referenced NetPhantom control is hidden, the label will be removed as well as the button. For this to work, the radio button tag must be placed between the label start tag and end tag, as in the example above.

These tags can be combined with the LABEL tag.

HTML tag
INPUT type="text"
INPUT type="checkbox"
INPUT type="radio"
<SELECT>
<TEXTAREA>

Message Box HTML File

In NetPhantom, message boxes are popup windows with information and some choices. This functionality is handled automatically in a Web Application by using a special message box HTML file. This HTML page is displayed as the main HTML page until removed by the user.

To change the default HTML appearance for message boxes, change the file webapps/messagebox.html in the server document root directory.

A specific message box file for an application is used instead of the file above, if such a file exists (full file path is *runtime_directory*/webAppName/html/messagebox.html).

Tags that are useful in the message box HTML file:

<@**MSGBOXTITLE@>	Displays the message box title.
<@**MSGBOXTEXT@>	Displays the message box text.
<MSGBOX>	Displays the message box buttons. NOTE! This tag should be placed within a <FORM> tag.

The following is the default `messagebox.html` file:

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
<title><@**MSGBOXTEXT@></title>
</head>
<body bgcolor="#FFFFFF">
  <h3><font face="Verdana"><@**MSGBOXTITLE@></font></h3>
  <form method="POST">
    <@**MSGBOXTEXT@>
    <p><MSGBOX>
  </form>
</body>
</html>
```

General Limitations

The HTML client sends a request to the NetPhantom server. It then expects one (and only one) answer to that request. If the panel changes after the reply is sent, the client will not know until he sends the next request. This means that the NetPhantom server needs to be as sure as possible that the right panel is sent.

The logic for standard client requests (such as *submit*) is that the server updates the panel in several steps and waits for everything to be stable before the next step is made. There is also the possibility to set and additional wait after the update is ready (see the *WebApplication* definition).

REXX Application Limitations

In general, there are no problems using REXX macros because they are a part of standard thread processing. When using REXX applications, you must make sure to set the lock flag (`PanLock(1)`). Even then there is a major risk that the NetPhantom server might send a response to the HTML client before the REXX application is started. To prevent this, set the lock flag from the initial REXX macro and then call the REXX application.

REXX Object Events

The events that are normally sent from the NetPhantom client (such as `SELECT` and `FOCUS`) are NOT sent from the HTML client. Only the `COMMAND` event (from submit or hyperlink) is sent from the HTML client.

The server events (such as `HOSTCHANGE` and `CREATE`) are normally sent to the objects, but make sure that the event is processed between the HTML client request and the NetPhantom server reply. For example: you write a REXX application connected to the `COMMAND` event of a push button that changes the content of a host field and then removes the panel lock flag. Then the NetPhantom server might reply to the client before the `HOSTCHANGE` event is activated.

24.20 NetPhantom Copy HTML

This menu item converts panel controls into HTML code that is placed on the clipboard. The HTML code is used to simplify the NetPhantom HTML integration process (see separate document). The generated code contains the basic functionality but remains to be graphically designed.

If one or more controls in the panel are selected, only these controls will be converted to HTML tags. If no control is selected, all controls and menus will be converted and will be placed within a `<FORM>` tag.

The HTML conversion requires that all referenced controls have a control ID. In normal NetPhantom development this is not always the case (only list boxes and controls connected to REXX usually have control IDs). When the **Copy HTML** menu item is activated, all controls are checked for control ID. If a control does not have a control ID but has a host field reference, you will be asked if this reference also should be set as control ID. If there is no host field reference, you will be asked if NetPhantom should generate a unique control ID.

Copy HTML creates two clipboard formats: `CF_HTML` and `CF_TEXT`. When using an HTML development tool, the HTML is inserted directly into the graphical environment, but when pasting into a text editor, the `CF_TEXT` format is used. Some applications – such as Microsoft Word – will use the `CF_HTML` format if available to create visual controls. If not available, the `CF_TEXT` format will be used to insert the text. If you want to copy only the text to Word, there is an option in the copyhtml.ini file, `ClipboardFormat=TEXTONLY` (see below), to ensure that only the `CF_TEXT` format is generated.

COPYHTML.INI File

This INI file is used to customize the copy to HTML behavior. It is located in the same directory as the NetPhantom Editor executable file. If this file does not exist, default values will be used (see item descriptions).

```
; COPYHTML.INI
; The base section redirect to another section
[BASE]
SECTION=DEFAULT

[DEFAULT]
; Create control IDs from hostfield.
AutoCreateFromHostField                =ASK

; Generate control IDs.
AutoCreate                             =ASK

; Convert editable combination boxes to "Entry fields".
EditComboToEntry                       =NO

; Convert editable spin buttons to "Entry fields".
EditSpinToEntry                        =NO

; Clipboard formats
;   HTMLTEXT=both CF_HTML and CF_TEXT
;   TEXTONLY only CF_TEXT
ClipboardFormat                        =HTMLTEXT

; List box border width (0-nn).
ListBoxBorder                          =1

; Copy list box header to HTML and set NpTemplate="HEADER".
ListHeaderToHTML                       =NO

; List box page up and down images (for page up/down column).
ListPageUpImage                        =
ListPageDownImage                      =

; Host connected output text is be converted to <input
; type="text"> with size set to length of host field. The
```

```
; NpDisableToText flag is also set. This option is used to handle
; output text in tables at design time.
HostOutputToInputText                   =NO

; Control specific added attributes.
EntryField                              =
MLE                                     =
CombinationBox                          =
SpinButton                              =
PushButton                              =
CheckBox                                =
RadioButton                             =
ListBox                                 =
MenuItem                                =
```

The `[BASE]` item `SECTION` is always read first; this is a fast way to change between different settings.

Description of `COPYHTML.INI` items:

AutoCreateFromHostField (*Yes/No/Ask*)	Setting for controls that have a host field connection but no control ID. Yes – always assign host field name as control ID. No – never assign a control ID Ask – display a message box for each control.
AutoCreate (*Yes/No/Ask*)	Setting for controls that have a no host field connection and no control ID. Yes – always generate a control ID. No – never generate a control ID Ask – display a message box for each control.
EditComboToEntry (*Yes/No*)	Because there is no HTML support for editable combination boxes you must choose between a non-editable (but selectable) `<SELECT>` tag and an editable `<INPUT>` tag with no drop-down list. Set "Yes" to convert all editable combination boxes to "Entry fields" or "No" to create input prohibited `<SELECT>` tags.
EditSpinToEntry (*Yes/No*)	Same as EditComboToEntry but for spin button.
ClipboardFormat (*HTMLTEXT/TEXTONLY*)	The default is `HTMLTEXT` to create both HTML and TEXT clipboard format. `TEXTONLY` only creates the clipboard TEXT format.
ListBoxBorder (*0-nn*)	Set the border attribute for the `<TABLE>` tag, default value is 2.
ListHeaderToHTML (*Yes/No*)	Copy the current NetPhantom list box header to an HTML row and set the row attribute `NpTemplate="HEADER"`.
HostOutputToInputText (*Yes/No*)	Host connected output text will be converted to `<INPUT type="text">` with size set to length of host field. The `NpDisableToText` flag is also set. This option is used to handle output text in tables at design time.
ListPageUpImage (Image file)	References to page up/down image files to be used with a page up/down column to the right in the table. These

ListPageDownImage (image file)	images are typically up/down arrows. If both values are specified and the list box has the **Page up/down buttons** option set, the table page up/down column will be created. Note that a relative path should be used.

Control Specific Added Attributes

All selected NetPhantom controls are converted into different HTML tags. For these tags, some attributes are created automatically by the conversion (see *Control Conversion* below). If you want to add extra default attribute parameters (such as style) for a specific control type, use the corresponding item in the INI file. Note that if you set an attribute that is created automatically, both your attribute and the automatic one will be generated.

Control Conversion

The following control types are converted to HTML code.

Phantom control type	Converted into HTML code
Static text	Text
Output text	HTML variable reference
Entry field	Tag <INPUT type"text">
MLE	Tag <TEXTAREA>
Combination box	Tags <SELECT> and <OPTION>
Spin button	Tags <SELECT> and <OPTION>
Push button	Tag <INPUT type="submit">
Checkbox	Tags <LABEL> and <INPUT type="checkbox">
Radio button	Tags <LABEL> and <INPUT type="radio">
List box	Tags <TABLE>, <TR> and <TD>
Group box	Text
Menu item	Tag <INPUT type="submit">

Static Text

Static text is converted to HTML text and any mnemonic is removed.

Output Text

Output text is converted to the HTML variable reference `<@-CTLID@>`. Mnemonics are removed.

Entry Field

Entry fields are converted to the HTML tag `<INPUT>`.

- Attribute type is set to `text` or `password` depending on entry field type.

- User attributes from `copyhtml.ini` are added.
- Attribute `name` is set to control ID.
- Attribute `size` is set to Entry field width.

Example:

```
<p><INPUT type="text" name="USER" size="15">
```

MLE

MLEs are converted to the HTML tag `<TEXTAREA>`.

- Attribute `name` is set to MLE control ID.
- Attribute `rows` is set to the MLE height.
- Attribute `cols` is set to the MLE width.
- User defined attributes from `copyhtml.ini` are added.
- Closing `</TEXTAREA>` tag is added.

Example:

```
<p><TEXTAREA name="MLE" rows="6" cols="24">
</TEXTAREA>
```

Combination Box

Combination boxes are by default converted to the HTML tag `<SELECT>`; see also *EditComboToEntry* user parameter in the `copyhtml.ini` file.

- Attribute `name` is set to combination box control ID.
- User defined attributes from `copyhtml.ini` are added.
- Available contents in the combination box are added as HTML tags `<OPTION> value`, even if the combination box contents are originating from a combination box file. Only the user contents are shown, not any eventual host conversion value. These option values are only to simplify the visual editing and are replaced with real data.

Example:

```
<p><SELECT name="SELLADDR">
   <OPTION>Yes
   <OPTION>No
   <OPTION>No info
</SELECT>
```

Spin Button

Spin buttons are by default converted to the HTML tag <SELECT>. The exception is when a spin button value range is set. In this case it is converted into the HTML tag <INPUT>; see also *EditSpinToEntry* user parameter in the `copyhtml.ini` file.

- Attribute `name` is set to spin button control ID.
- Attribute `size` is set to `1`.
- User defined attributes from `copyhtml.ini` are added.
- Available contents in the spin button are added as HTML tags <OPTION> value. Only the user contents are shown not any eventual host conversion value.

Example:


```
<p><SELECT name="SPIN" size="1" multiple >
    <OPTION> Spin Button
    <OPTION> Selections
</SELECT>
```

Push Button

Push buttons are converted to the HTML tag <INPUT>.

- Attribute `type` is set to `submit`.
- Attribute `value` is set to push button text. Mnemonics and images are removed.
- Attribute `name` is set to the push button control ID.
- User attributes from `copyhtml.ini` item "PushButton" is added.

Example:


```
<p><INPUT type="submit" value="End" name="F_1">
```

Checkbox

Checkboxes are converted into two tags: <LABEL> and <INPUT>. The label is associated with the input tag. The input tag is nested within the label tag because of the automatic hidden functionality that removes both the input and label tags when the NetPhantom control is hidden (see *NetPhantom Web Application using HTML* documentation).

- Label attribute `for` is set to checkbox control ID for association.
- Input attribute `type` is set to `checkbox`.
- Input user defined attributes from `copyhtml.ini` are added.
- Input attribute `name` is set to checkbox control ID.
- Input attribute `id` is set to checkbox control ID for association with label.
- Label text (checkbox text) is added.
- Closing </LABEL> tag is added.

Example

 Gold Club member

```
<p><LABEL for="F_6">
<INPUT type="checkbox" name="F_6" id="F_6">
Gold Club member
</LABEL>
```

Radio Button

Radio buttons are converted into two tags: <LABEL> and <INPUT>. The label is associated with the input tag. The input tag is nested within the label tag because of the automatic hidden functionality that removes both the input and label tags when the NetPhantom control is hidden (see *NetPhantom Web Application using HTML*).

- Label attribute `for` is set to radio button control ID for association.
- Input attribute `type` is set to `radio`.
- Input user defined attributes from `copyhtml.ini` are added.
- Input attribute `name` is set to a generated radio group name. This name stays the same when multiple radio buttons are in tab sequence.
- Input attribute `value` is set to radio button control ID.
- Input attribute `id` is set to radio button control ID for association with label.
- Label text (radio button text) is added.
- Closing </LABEL> tag is added.

Example:

Excellent

```
<p><LABEL for="F_1">
<INPUT type="radio" name="group1" value="F_1" id="F_1">
Excellent
</LABEL>
```

List Box

List boxes are converted to the HTML tag <TABLE>.

- Attribute `id` is set to list box control ID.
- Attribute `border` is set from copyhtml.ini item `ListBoxBorder`.
- User attributes from `copyhtml.ini` item `ListBox` are added.
- List box header is added if it exists. If the `copyhtml.ini` item `ListHeaderToHTML` is set to `Yes`, the attribute `NpTemplate="HEADER"` will be added to the header row.
- List box rows are created from list box height (minimum 1 row) and are filled with sample text. The first row will contain the attribute – `NpTemplate="ROW"`.
- Column width is set as percent of the full list.
- If the list box has the **Page up/down buttons** option set and copyhtml.ini contains the ListPageUpImage and ListPageDownImage image references, a page up/down column will be inserted to the right in the table.

Example:

Customer number	Name	Last purchase	Amount	
ABCDEFGH	ABCDEFGHIJKLMNOPQRSTUVW	ABCDEF	ABCDEFGHI	
ABCDEFGH	ABCDEFGHIJKLMNOPQRSTUVW	ABCDEF	ABCDEFGHI	
ABCDEFGH	ABCDEFGHIJKLMNOPQRSTUVW	ABCDEF	ABCDEFGHI	

```
<p><TABLE id="LIST" border="3">
  <TR NpTemplate="HEADER">
    <TD width="16%">Customer number</TD>
    <TD width="51%">Name</TD>
    <TD width="12%">Last purchase</TD>
    <TD width="20%">Amount</TD>
  </TR>
  <TR NpTemplate="ROW">
    <TD width="16%">ABCDEFGH</TD>
    <TD width="51%">ABCDEFGHIJKLMNOPQRSTUVWXY</TD>
    <TD width="12%">ABCDEF</TD>
    <TD width="20%">ABCDEFGHIJ</TD>
    <TD NpPageColumnImages NpPageDownSrc="my_images/pgdn.gif">
      <IMG border="0" src="my_images/pgup.gif"></TD>
  </TR>
  <TR>
    <TD width="16%">ABCDEFGH</TD>
    <TD width="51%">ABCDEFGHIJKLMNOPQRSTUVWXY</TD>
    <TD width="12%">ABCDEF</TD>
    <TD width="20%">ABCDEFGHIJ</TD>
```

```
    <TD rowspan="1"> </TD>
  </TR>
  <TR>
    <TD width="16%">ABCDEFGH</TD>
    <TD width="51%">ABCDEFGHIJKLMNOPQRSTUVWXY</TD>
    <TD width="12%">ABCDEF</TD>
    <TD width="20%">ABCDEFGHIJ</TD>
    <TD><img border="0" src="my_images/pgdn.gif"></TD>
  </TR>
</TABLE>
```

Group Box

Group box titles are converted to HTML text. Mnemonics are removed.

Menu Item

Menu items are converted in the same way as push buttons to HTML tag <INPUT>. They are placed last and are separated from the controls by a horizontal rule <HR>.

- Attribute `type` is set to `submit`.
- Attribute `value` is set to menu item text. Mnemonics and images are removed.
- Attribute `name` is set to the menu item control ID.
- User attributes from `copyhtml.ini` item `MenuItem` are added.

Example:


```
Bottom of Form
<HR>
<INPUT type="submit" value="Help..." name="F_2">
<INPUT type="submit" value="About..." name="F_3">
```

24.21 HTML Application Errors

Application errors such as client time out or invalid session ID are generally displayed in the document root `\hterrors\NetPhantomApplicationError.html`. The HTTP session variable `@**ERREXTRA@` contains the error message text.

Certain application error messages can be customized by overriding the `NetPhantomApplicationError.html` with one of the following specified HTML files. These files should also be placed in the `\hterrors` directory. If they are not found there, the standard error HTML file will be displayed.

Description	**HTML filename**
Maximum current user count reached	`NetPhantomAppErr_MaxCurrUsers.html`
Invalid cookie session ID	`NetPhantomAppErr_InvalidCookie.html`
Invalid session ID	`NetPhantomAppErr_InvalidSession.html`
Host session failed	`NetPhantomAppErr_HostSessionFailed.html`
Host screen not recognized	`NetPhantomAppErr_HostScrNotFound.html`

Client session timed out	`NetPhantomAppErr_ClientTimeOut.html`
Application exit	`NetPhantomAppErr_AppExit.html`
Client is disposed	`NetPhantomAppErr_ClientDisposed.html`

24.22 Frequently Asked Questions

Q1. The correct panel is displayed, but none of the controls are connected.

A1. All the controls (tags) must be placed within a `<FORM>` tag.

Q2. When the HTML application returns to the "main" HTML page after a message has been displayed, the wrong control has focus.

A2. Focus can be set via JavaScript. See section *Focus in HTML Forms*.

Q3. When I use an input-submit tag everything works OK, but when changing it to a hyperlink it doesn’t work.

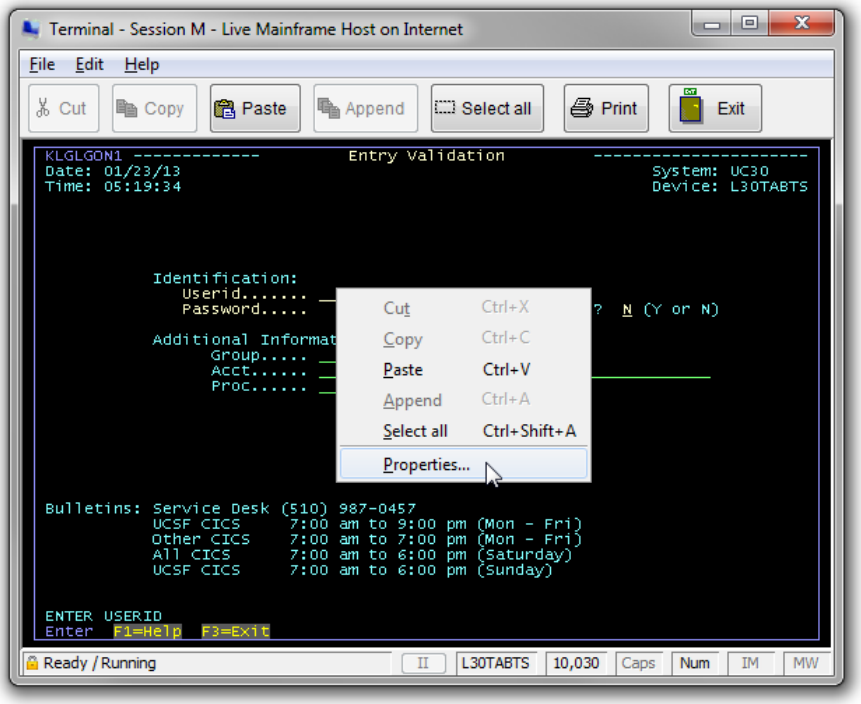
A3. Hyperlinks don’t update form data and can only be used when there is no need for the client to update any data.

25 The NetPhantom Terminal Window

A terminal emulator window look-and-feel as well as full clipboard cut/copy/paste functionality is provided, along with user-defined settings, e.g. fonts, colors, rule cursor, window size and placement.

The terminal window and its dialog boxes are a NetPhantom Runtime application called *terminal.jar* in the *terminal* subdirectory on the Server. This application can be customized and e.g. translated into another language.

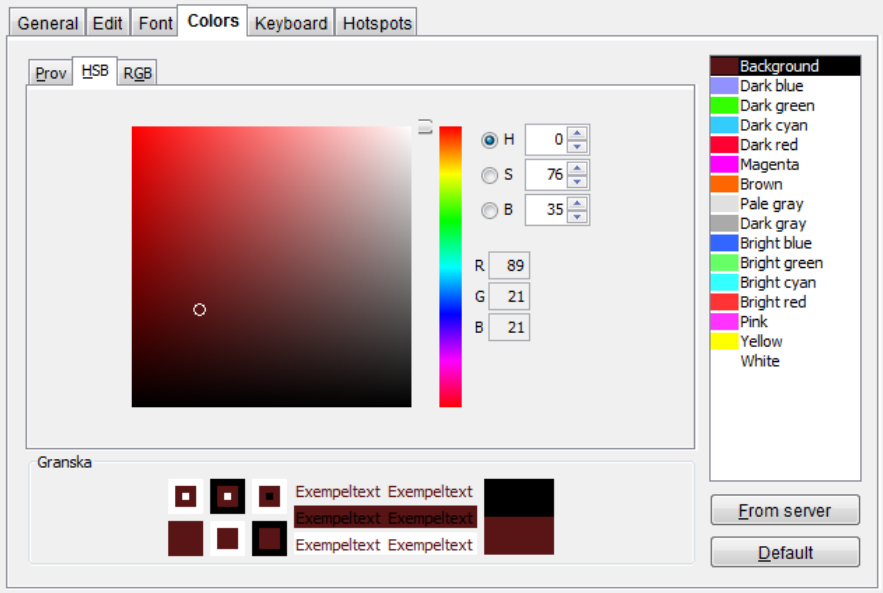
But this is not all: it is extendable by means of new toolbar buttons, menu items, pull-down menus, dialog boxes, etc., with code written in Java on the server-side -- no extension is required whatsoever on the client side.



To extend the Terminal Application with new functionality, see chapter 25.7.

25.1 Terminal Properties

Terminal properties can now be specified on an end-user basis; a properties file is stored on the client machine. The properties are display options including rule cursor, edit options, the terminal font and colors, window size and placement.



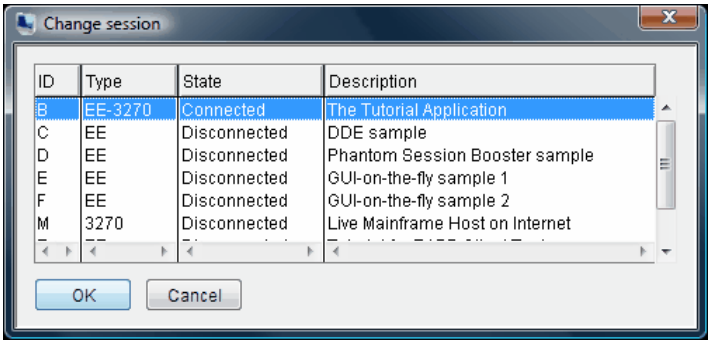
The properties file name is `.NetPhantomClient.properties` and is stored in the directory of the users' home directory concatinated with the user ID, e.g. "`C:\Documents and Settings\myuserid\myuserid`" under Windows XP (the reason to concatinate with the user ID is that some operating systems does not provide a "personal" home directory on the PC, rather a "global" home directory for all users).

Sample properties file contents

```
#Thu Sep 21 11:25:45 CEST 2021
CursorNonBlinking=0
RuleCursorLineStyle=1
PasteTextWrapping=1
HostColors=0000009696f988ed88a5f5f5f26f6f...
BoldFont=1
PromptOnParse=0
SaveOriginalFieldSpacing=0
FontSize=0
RuleCursor=0
WindowBounds=286,182,841,593
FontName=Lucida Console
RuleCursorOrientation=3
DoNotOutlinePresentationSpace=0
CutCopyParsing=0
```

25.2 Changing Terminal Session

If enabled by the server administrator, the current terminal session can be changed without affecting the currently connected session(s). To access the **Change session** dialog box, choose the menu item **File - Change session**.

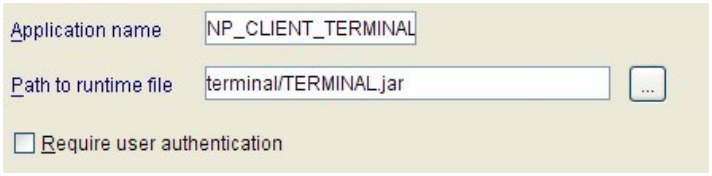


25.3 Configuring the Terminal Window

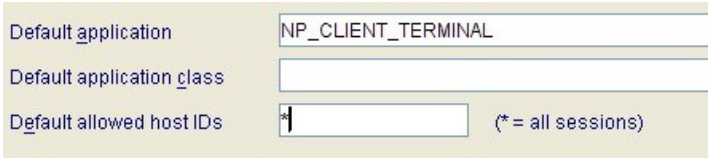
Use the Server Administration program to configure the Terminal Window and follow the points below.

Select the menu item **Configure - Server - Base**.

Select the **Applications** notebook tab, and make sure that an Application name **NP_CLIENT_TERMINAL** exists and points to the runtime file **terminal/TERMINAL.jar**. If not, add a new application with the following data, and make sure to load it.



Select the **Host** notebook tab. Default settings that are used for all host sessions (unless specifically overridden for a host session, are specified on the notebook tab in the fields below.



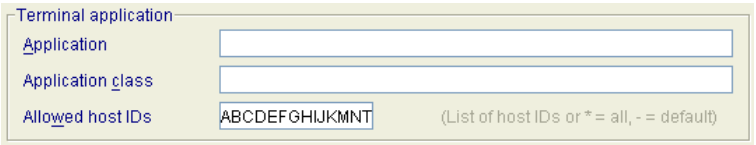
Default application should be set to a *loaded* Application name.

Default application class is used for extending the terminal application (see chapter 25.4 below) and should be left blank.

A user can change terminal session, and to control to which session he/she is allowed to change to, specify a list of host IDs in **Default allowed host IDs**. The special entry ***** indicates all configured host sessions.

Each host session now has a description (that is displayed in the terminal window title bar and in the **Change session** dialog box):

To override the default terminal application settings for a specific host session, use the fields as described below.



Application should specify a *loaded* Application name.

The **Application class** is used for extending the terminal application (see chapter 25.4 below) and should be left blank.

A user can change terminal session, and in order to control to which session he/she can change to, specify a list of host IDs in **Allowed host IDs**. The special entry ***** indicates all configured host sessions, and **-** specifies the default settings for all sessions. Leaving this entry blank will the allowed host IDs to be empty and thus can a user not change session.

Restart the server.

25.4 Manual Configuration of the Terminal Window

First the runtime application used for the terminal window must be defined and loaded as an application. This is done in the [Application] section as below (items in bold are the required changes):

```
[Application]
load=MYAPP1 MYAPP2 NP_CLIENT_TERMINAL MYAPP3
NP_CLIENT_TERMINAL=terminal/TERMINAL.jar
```

Then, the host session(s) must be configured. Each host session can be configured individually (e.g. with a different language of the TERMINAL.jar application. To avoid the requirement to specify an application for all host sessions, a default application can be specified. The table below describes all items that apply to the terminal application settings in the [host] section:

To use the default terminal application provided with NetPhantom, specify:

```
[host]
defaultApplication=NP_CLIENT_TERMINAL
```

`defaultApplication`	If specified, should be the name of the terminal application (e.g. `NP_CLIENT_TERMINAL` as the example above showed).
`defaultApplicationClass`	If specified, the class name of the terminal application Java implementation should be extending the server class `se.entra.phantom.server.TerminalApplication`.
`defaultAllowedSessions`	A default list of host session IDs that an end-user can change session to. If this entry is empty or not present, the menu item **File - Change session** will be disabled unless overridden by a host ID-specific definition. The list of host sessions corresponding to the specified host IDs are displayed (in the specified order) in the listbox in the dialog. The host IDs should be specified without separator, e.g. the host IDs A, M, B, Q and C (in that order) are specified as `defaultAllowedSessions=AMBQC`. The wild-card character `*` can be used to specify all available host sessions.
X`.application` (where *X* is the host session ID)	If specified, there should be the name of the terminal application. If not specified, the `defaultApplication` will be used.
X`.class` (where *X* is the host session ID)	If specified, the class name of the terminal application Java implementation should be extending the server class `se.entra.phantom.server.TerminalApplication`. If not specified, the `defaultApplicationClass` will be used.

X.description (where *X* is the host session ID)	An optional description of the terminal session. This description is seen in the title bar of the terminal window and in the **Change session** dialog box.
X.allowedSessions (where *X* is the host session ID)	A list of host session IDs that an end-user can change session to. If this entry is empty, the menu item **File - Change session** will be disabled. If it is not specified, the defaultAllowedSessions entry is used instead. The wild-card character * can be used to specify all available host sessions.

25.5 Special Requirements for *terminal.jar*

There are several special requirements for the terminal application described below:

1. Only one main panel may exist, and it must be named TERMINAL.
2. All other panels must be of pop-up type.
3. No panel can or should be host-connected.
4. Push buttons or menu items may not use the entry "Next panel".
5. No NetPhantom Object connections may exist, and absolutely no REXX code.
6. All interaction with the panels must be done with Java code.
7. Extending application must extend from se.entra.phantom.server.TerminalApplication.
8. Another requirement is that panels must be created with the following code:

```
VirtualSessionManager vsm=panel.getVirtualSessionManager();
VirtualPanelSession vps=vsm.getCurrentTerminalSession();
VirtualPanel newPanel=null;
if ( vps!=null )
  {
  VirtualPanelListener ll=new YourVirtualComponentAdapter();
  newPanel=vps.createTerminalPanel("PANELID",ll);
  }
```

25.6 Default Implementation in TerminalApplication

The class se.entra.phantom.server.TerminalApplication that all extending classes for the terminal application should use performs some default operation that can be overridden by the extending class.

- Upon onPanelCreate the title bar text in the main panel (TERMINAL) is updated from e.g. "Terminal Application - ?" to "Terminal Application - Session X - Description", where the "X" is the current host session ID and the "Descritpion" is the text specified in server.ini [base] section as X.description=Description.
- Upon onPanelCreate the host sessions that the user can change to is stored in the variable allowedSessions and the menu item with the Control ID "CHGSESS" is disabled if the X.allowedSessions entry in server.ini [host] section (or defaultAllowedSessions) is not found or empty.
- The following Control IDs perform actions in the onAction method:

EXIT	performExit
PROPS	performProperties
CHGSESS	performChangeSession
CUT	performCut
COPY	performCopy
PASTE	performPaste
APPEND	performAppend
SELALL	performSelectAll
CONNECT	performConnect
PRINT	performPrint
PRTWIN	performPrintWindow
ABOUT	performAbout

Upon changes in the selection rectangle in the terminal part the menu items and pop-up menu items CUT, COPY and APPEND are enabled or disabled.

For a detailed description of the terminal application, see the Java documentation or the source listings.

25.7 Extending the TerminalApplication

The terminal application is a NetPhantom Runtime application located in the `terminal` directory on the server side and is called `TERMINAL.PHR`. This file is compiled with NetPhantom Editor with the option "Backwards compatible".

If you need another language for this application, just translate the `TERMINAL.PHM` file as all texts is externalized.

Special requirements for `terminal.jar`

There are several special requirements for the terminal application described below:

1. Only one main panel may exist and must be named TERMINAL.
2. All other panels must be of pop-up type.
3. No panel can or should be host-connected.
4. Push buttons or menu items may not use the entry "Next panel".
5. No NetPhantom Object connections may exist, and absolutely no REXX code.
6. All interaction with the panels must be done from Java code.
7. Extending application must extend from `se.entra.phantom.server.TerminalApplication`.

Another requirement is that panels must be created with the following code:

```
VirtualSessionManager vsm=panel.getVirtualSessionManager();
VirtualPanelSession vps=vsm.getCurrentTerminalSession();
VirtualPanel newPanel=null;
if ( vps!=null )
  {
  VirtualPanelListener ll=new YourVirtualComponentAdapter();
  newPanel=vps.createTerminalPanel("PANELID",ll);
  }
```

Default implementation in `TerminalApplication`

The class `se.entra.phantom.server.TerminalApplication` that all extending classes for the terminal application should use performs some default operation that can be overridden by the extending class.

- Upon `onPanelCreate` the title bar text in the main panel `TERMINAL` is updated from e.g. "Terminal Application - ?" to "Terminal Application - Session V - Description", where the "V" is the current host session ID and the "Descritpion" is the text specified in "server.ini" [base] section as "V.description=Description".

- Upon `onPanelCreate` the host sessions that the user can change to is stored the the variable `allowedSessions` and the menu item with the Control ID "CHGSESS" is disabled if the "V.allowedSessions" entry in "server.ini" [host] section (or "defaultAllowedSessions") is not found or empty.

- The following Control IDs perform actions in the `onAction` method:

`EXIT`	performExit
`PROPS`	performProperties
`CHGSESS`	performChangeSession
`CUT`	performCut
`COPY`	performCopy
`PASTE`	performPaste
`APPEND`	performAppend
`SELALL`	performSelectAll
`CONNECT`	performConnect
`PRINT`	performPrint
`PRTWIN`	performPrintWindow
`ABOUT`	performAbout

- Upon changes in the selection rectangle in the terminal part the menu items and pop-up menu items CUT, COPY and APPEND are enabled or disabled.

26 The NetPhantom Mail Utility

The NetPhantom mail utility can be used for sending mail from the NetPhantom server.

There are four ways of accessing the mail utility in NetPhantom. These are:

1. Using a MailForm CGI from an HTML page.
2. By using the REXX API.
3. By use of the NetPhantom Java API.
4. By use of NetPhantom standalone mail utility.

Each of these methods will be described in detail below.

26.1 The MailForm CGI

Before you can send mail from an HTML page, you must configure the MailForm CGI and a web resource for the CGI. This is done in the Server Administration Client via the menu item **Server – Configure – Web server**.

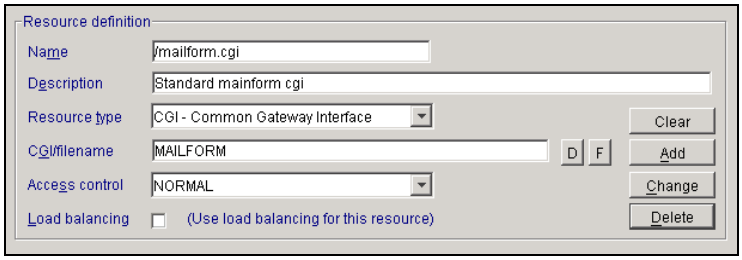
Configure the MailForm CGI

The Java class name must be set to `se.entra.phantom.server.http.HtmlMailFormCGI`. The option HTML included CGI should *not* be checked.

*Click the **CGI** tab in the **Web server** dialog box to configure the CGI.*

Configure a Resource

The **Name** of the MailForm CGI should start with a forward slash and the **Resource type** must be `CGI`.



*Click the **Resource** tab in the **Web server** dialog box to configure a resource for the MailForm CGI.*

The MailForm CGI can be used to send mail directly from an HTML page. This is done by creating an HTML FORM. The form's action is set to the MailForm CGI.

The HTML FORM used for sending a mail must include several tags from which the CGI gets the information it needs to be able to send the mail. These tags can be divided into required tags and optional tags.

MailForm Tags

The tags identified by the MailForm CGI are:

Name	Type
email_to	Required
email_cc	Optional
email_bcc	Optional
email_from	Required
email_server	Required
email_host	Optional
email_use_ssl	Optional
email_port	Optional
email_auth_user	Optional
email_auth_password	Optional
email_field_separator	Optional
email_doc	Required
email_success_doc	Optional
email_error_doc	Optional
email_output	Optional

These tags will be described in detail below. They are implemented as input fields in the HTML FORM, where the tag is the input field name. The input field can be of type "hidden".

Any input fields in the HTML FORM that do not correspond to a MailForm tag will be inserted into the mail message. All tags identified by the MailForm CGI will be saved in HTTP user variables and can be accessed from the HTML document that will be shown after the CGI has ended its processing (see below).

```
<html>
  <head>
    <title>Mail Confirmation</title>
    <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
    <style>
      <!--
        td  { font-family: Verdana; font-size: 10pt }
        input { font-family: Verdana; font-size: 10pt }
        textarea { font-family: Verdana; font-size: 10pt }
       -->
    </style>
  </head>
  <body bgcolor="#FFFFFF" text="#000000" topmargin="0" leftmargin="2">
    <table border="0" cellpadding="0" cellspacing="0" width="434">
      <tr>
        <td valign="top">
          <font size="4" face="Verdana, Arial, Helvetica, sans-serif">
          <b>Mail Confirmation</b>
          </font>
          <p>
          Your mail has been send to address <b><@#EMAIL_TO@></b>.
          The mail's subject was <b><@#EMAIL_SUBJECT@></b>.
          <p>
          This HTML-page, which is defined as the <i>email_success_doc</i>
```

```
            in the sample mail forms, uses the HTTP-variables
            &lt;@EMAIL_TO@&gt; and &lt;@EMAIL_SUBJECT@&gt; to get the
            recipient and the subject for the send mail.
          </td>
        </tr>
      </table>
    </body>
</html>
```

If an error occurs, the error message can be retrieved from the HTTP variable `<@#EMAIL_ERROR@>`.

26.2 Alphabetical List of MailForm Tags

email_bcc

<INPUT type=*type* name="**email_bcc**" value=*value*>

The value for this field will be used to get the email address(es) of the recipient of a blind carbon copy. If more than one email address is needed, separate them with commas or semi-colons.

Parameters

type
text | hidden

value
The email address(es) of the recipient(s) of a blind carbon copy.

Example

Sets the recipient of a blind carbon copy of this mail to John Smith in the company.

```
<INPUT type="hidden"
       name="email_bcc"
       value="john.smith@company.com">
```

email_cc

<INPUT type=*type* name="**email_cc**" value=*value*>

The value for this field will be used to get the email address(es) of the recipient of a carbon copy. If more than one email address is needed, separate them with commas or semi-colons.

Parameters

type
text | hidden

value
The email address(es) of the recipient(s) of the carbon copy.

Example

Sets the recipient of a carbon copy of this mail to a salesperson in the company.

```
<INPUT type="hidden"
       name="email_cc"
       value="our.seller@company.com">
```

email_doc

<INPUT type=*type* name="**email_doc**" value=*value*>

The value for this field will be used as the default document that will be sent to the user after they submit the MailForm. If this field is used together with both the *email_success_doc* and the *email_error_doc*, then this value will be ignored.

Comments

The current directory is the directory where the resource is located or the `htdocs` directory if no resource directory exists.

Parameters *type*
text | hidden

value
The name of the document to send to the user.

Example Sets the document that will be sent to the user as `http://webserver/resource/confirm.html`.

```
<INPUT type="hidden"
       name="email_doc"
       value="confirm.html">
```

email_error_doc

<INPUT type=*type* name="**email_error_doc**" value=*value*>

The value for this field will be used as the document that will be sent to the user after they submit the MailForm, if the sending of the mail failed. If this tag is specified, it will take precedence over the *email_doc* tag when the sending of the mail is unsuccessful.

Parameters *type*
text | hidden

value
The name of the document to send to the user after a mail has been sent unsuccessfully.

Example Sets the document that will be sent to the user to `nosuccess.html`.

```
<INPUT type="hidden"
       name="email_error_doc"
       value="nosuccess.html">
```

email_field_separator

<INPUT type=*type* name="**email_field_separator**" value=*value*>

The value for this field will be used to separate the input field name from its value in the email message. If this tag is omitted, an "=" (equal sign) will be used as default. This tag will be ignored if *email_output* defines the use of a template for formatting the message.

Parameters *type*
text | hidden

value
The string to be used as the field separator.

Example Sets the field separator for this email message to " : ".

```
<INPUT type="hidden"
       name="email_field_separator"
       value=" : ">
```

email_from

<INPUT type=*type* name="**email_from**" value=*value*>

The value for this field will be used together with the *email_server* tag to build the email address of the sender. If this value only sets the sender's email name, then it will be combined with the server name provided with the *email_server* field.

Parameters *type*
text | hidden

value
The user part of the sender's address.

Example Sets the user part of the sender part of this mail to the IT department in the company.

```
<INPUT type="hidden"
       name="email_from"
       value="it.dept">
```

email_host

<INPUT type=*type* name="**email_host**" value=*value*>

The value for this field will be used to specify the host address for the outgoing mail server. If the address to the outgoing mail is *mail*, then there is no need to specify it since *mail* is the default address. But if the address is something else, it must be specified.

Parameters *type*
text | hidden

value
The address to the outgoing mail server.

Example Sets the address to the outgoing mail server.

```
<INPUT type="hidden"
       name="email_host"
       value="mymail.company.com">
```

email_output

<INPUT type=*type* name="**email_output**" value=*value*>

The value for this field will be used as the name for a template file to be used for formatting the message. The filename including a path relative to the server directory.

Parameters *type*
text | hidden

value
The name of the template file.

Example Sets the name of the template file that will be used for formatting to `mymailtemplate.txt`.

```
<INPUT type="hidden"
       name="email_output"
       value="mymailtemplate.txt">
```

email_server

<INPUT type=*type* name="**email_server**" value=*value*>

The value for this field will be used together with the *email_from* tag to build the email address of the sender. If the value provided by the *email_from* field only contains the sender's email name, then it will be combined with the server's name to create the sender's complete email address.

Parameters *type*
text | hidden

value
The server part of the sender's email address.

Example Sets the user part of the server part of this mail to the company.

```
<INPUT type="hidden"
       name="email_server"
       value="company.com">
```

email_subject

<INPUT type=*type* name="**email_subject**" value=*value*>

The value for this field will be used as the header for the email.

Parameters *type*
text | hidden

value
The header for this email message.

Example Sets the header for this email message to Request Form.

```
<INPUT type="hidden"
       name="email_subject"
       value="Request Form">
```

email_success_doc

<INPUT type=*type* name="**email_success_doc**" value=*value*>

The value for this field will be used as the document that will be sent to the user after they submit the MailForm, if the sending of the mail succeeds. If this tag is specified, it will take precedence over the *email_doc* tag when the sending of the mail is successful.

Parameters *type*
text | hidden

value
The name of the document to send to the user after a mail has been sent successfully.

Example Sets the document that will be sent to the user to `success.html`.

```
<INPUT type="hidden"
       name="email_success_doc"
       value="success.html">
```

email_to

<INPUT type=*type* name="**email_to**" value=*value*>

The value for this field will be used as the email address(es) of the recipient. If more than one email address is needed, separate them with commas or semi-colons.

Parameters *type*
text | hidden

value
The email address(es) of the recipient(s).

Example Sets the recipient for this mail to the order department in the company.

```
<INPUT type="hidden"
       name="email_to"
       value="order@company.com">
```

email_use_ssl

<INPUT type=*type* name="**email_use_ssl**" value=*value*>

The value for this field will be used to set the usage of SSL for mail sending. The value can be 0 or 1, or “true” or “false”.

Parameters *type*
text | hidden

value
The SSL flag (0, 1, true or false).

Example Sets the mail sending to use SSL.

```
<INPUT type="hidden"
       name="email_use_ssl"
       value="true">
```

email_port

<INPUT type=*type* name="**email_port**" value=*value*>

The port to use for the mail sening. Normally this value is 25 for non-SSL and 110 or 995 for SSL. The port number can be 1-65535.

Parameters *type*
text | hidden

value
The port number (1-65535).

Example Sets the mail sending port..

```
<INPUT type="hidden"
       name="email_port"
       value="995">
```

email_auth_user

<INPUT type=*type* name="**email_auth_user**" value=*value*>

The authentication user if the receiver mail system so requires.

Parameters *type*
text | hidden

value
The SSL flag.

Example Sets the mail sending authentication user.

```
<INPUT type="hidden"
       name="email_auth_user"
       value="myuser">
```

email_auth_password

<INPUT type=*type* name="**email_auth_password**" value=*value*>

The authentication password if the receiver mail system so requires.

Parameters *type*
text | hidden

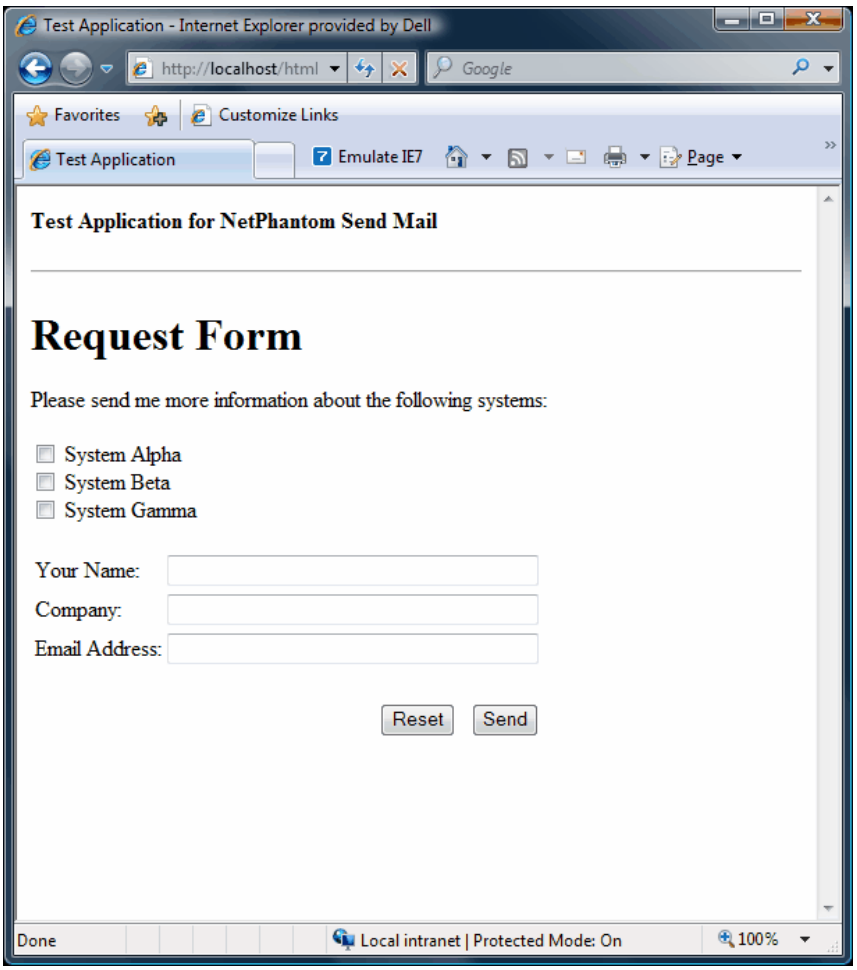
value
The SSL flag.

Example Sets the mail sending authentication password.

```
<INPUT type="hidden"
       name="email_auth_password"
       value="myuser">
```

26.3 MailForm Example

In this example a simple request form has been created. When the user has filled in this form and presses the Send button, a mail will be sent using the MailForm CGI.



A simple request form.

The HTML page for this request form has been created with the following HTML code:

```
<HTML>
  <HEAD>
    <TITLE>Test Application</TITLE>
  </HEAD>
  <BODY bgcolor="#ffffff">
    <H4>Test Application for NetPhantom Send Mail</H4>
    <HR>
    <FORM method="post"
          action="http://localhost/mailform.cgi">
      <H1>Request Form</H1>
      <INPUT type="hidden"
             name="email_to"
             value="john.smith@netphantom.com">
      <INPUT type="hidden"
             name="email_from"
             value="phantom">
      <INPUT type="hidden"
             name="email_server"
             value="netphantom.com">
      <INPUT type="hidden"
             name="email_field_separator"
             value="=">
      <INPUT type="hidden"
             name="email_subject"
```

```
        value="My Request Form">
    <INPUT type="hidden"
        name="email_doc"
        value="mail_confirm.html">
    <INPUT type="hidden"
        name="email_success_doc"
        value="mail_ok.html">
    <INPUT type="hidden"
        name="email_error_doc"
        value="mail_error.html">
    Please send me more information about the following systems:
    <BR>
    <BR>
    <INPUT type="checkbox" name="system1" value="selected">
    System Alpha
    <BR>
    <INPUT type="checkbox" name="system2" value="selected">
    System Beta
    <BR>
    <INPUT type="checkbox" name="system3" value="selected">
    System Gamma
    <BR>
    <BR>
    <TABLE>
      <TR>
        <TD>Your Name:</TD>
        <TD><INPUT type="text"
                   name="name"
                   maxlength="80"
                   size="40"></TD>
      </TR>
      <TR>
        <TD>Company:</TD>
        <TD><INPUT type="text"
                   name="company"
                   maxlength="80"
                   size="40"></TD>
      </TR>
      <TR>
        <TD>Email Address:</TD>
        <TD><INPUT type="text"
                   name="email"
                   maxlength="80"
                   size="40"></TD>
      </TR>
      <TR>
        <TD> </TD>
        <TD> </TD>
      </TR>
      <TR>
        <TD></TD>
        <TD align="right">
          <INPUT type="reset" value="Reset">
           
          <INPUT type="submit" value="Send">
        </TD>
      </TR>
    </TABLE>
    <BR>
  </FORM>
 </BODY>
</HTML>
```

In the code above, all the visible input fields have been placed inside a table for formatting reasons.

The interesting parts of the above code are described in detail in the following sections.

The Form Definition

To use the MailForm CGI, an HTML form must be defined. The form's method must be *post*, and the action must be the URL to the CGI resource. All the input fields to be processed by the CGI must be placed inside the form.

```
<FORM method="post"
      action="http://webserver/mailform.cgi">
.
.
.
</FORM>
```

The Tag Definitions

All the required tags and some optional tags have been defined as hidden input fields. Some of these tags could be created as input fields of type "text" if the user should be able to edit them.

The name of these input fields is set to the tag name, and their value is set to the value determined by the designer of the HTML page.

```
<INPUT type="hidden"
       name="email_to"
       value="john.smith@netphantom.com">
```

Fields for User Information

In this example there are two different types of fields for the user to fill in. There are three checkboxes and three entry fields. Observe that if checkboxes are used in the form and no template for the formatting of the mail message is used, only checked checkboxes will be included in the message.

A value must be specified for the checkboxes. It will be used as the checkbox value in the message. In this example the value has been set to "selected".

```
<INPUT type="checkbox" name="system1" value="selected">
```

For the entry fields, the value entered in the field will be its value. In this example, the entry field's maxlength and its size have been specified.

```
<TD><INPUT type="text" name="name" maxlength="80" size="40"></TD>
```

Input Field for Resetting the Form

It is customary to add a push button for resetting all the editable fields in the form. This is done by specifying an input field with the type set to *reset*. The value is the text to be displayed on the button.

```
<INPUT type="reset" value="Reset">
```

Input Field for Sending the Form

A second push button is used for sending the form to the MailForm CGI. This button is created by specifying an input field with the type set to *submit*. The value is the text to be displayed on the button.

```
<INPUT type="submit" value="Send">
```

26.4 Formatting the Mail

The *email_output* field is used to determine the mail format. If no *email_output* field has been specified, the default format will be used. If you wish to use a template to format the mail, enter the template name in the value field of the *email_output* field. Formatting with the help of a template will be described in the next section.

If the default formatting is used, the message will contain the input fields and their values, one field per row. The field name will be separated from its value either by the string defined by the *email_field_separator* value, or if no *email_field_separator* has been given, by an equal sign (=).

26.5 Formatting with the Use of a Template

The *email_output* field can be used to specify the name of a template file to be used for formatting the message.

The template file is normal text file where tags can be used to specify where the values from the different input fields should be placed. These tags will then automatically be replaced with the value of the corresponding input field during formatting.

The tags are specified by the input field name between carets (^). For example, if you have an entry field with the name *company*, the corresponding tag in the template file would be:

```
^company^
```

See *MailForm Template Example* below.

Template Command Tags

Template command tags are tags that are used to specify special formatting functions. These command tags start with ^@, and end with @^. They will always be removed from the formatted message. Command tags must be on a single line.

Currently there are only two command tags, and they are:

Name	Short description
set_field	Sets fields in the mail CGI
unused_tag	Processing of unused tags

Below follows a more detailed description of the template command tags.

26.6 Alphabetical List of Template Command Tags

set_field

^@**set_field**(fieldname,value)@^

Sets a field in the mail CGI. This command makes it possible to set fields without the fields being visible from the browser, by choosing view source. If the same field is set from the mail form and from the template with this command, it will be given the value set by this command.

Comments — The *value* parameter must not contain the comma (,) character.

Parameter

fieldname
The name of the field.

value
The value for the field.

Example — The following command will set the "from" field for the mail.

```
^@set_field(email_from,phantom@netphantom.com)@^
```

unused_tag

^@**unused_tag**(command)@^

The command for this tag will determine what should be done with unused tags. This command can, for example, be used to remove unselected checkboxes from the formatted message.

Parameter *command*
remove_line | remove_tag | leave

Example The following command tag will have removed the unused tag(s) in the messages.

```
^@unused_tag(remove_tag)@^
```

26.7 MailForm Template Example

Below is a simple example of a template file that could be used with the mail form in the example above.

```
^@unused_tag(remove_tag)@^
Request for information
Please send information about the following systems:
System Alpha: ^system1^
System Beta: ^system2^
System Gamma: ^system3^
Send the information to:
Name: ^name^
Company: ^company^
E-mail: ^email^
```

Below is an example of how the received mail might look using the above template. In this example *System Alpha* and *System Beta* were checked, and *System Gamma* was unchecked.

Request for information
Please send information about the following systems:
System Alpha: selected
System Beta: selected
System Gamma:
Send the information to:
Name: Jones Smith
Company: MyCompany
E-mail: jones.smith@mycompany.com

Notice that the initial template command has been removed, all the tags except `^system3^` have been replaced with data from the mail form, and the tag `^system3^` has been removed, when processing the template command.

26.8 The NetPhantom REXX API for Mail

The NetPhantom's REXX API can also be used for sending mail. This makes it possible to give any NetPhantom application mailing functionality.

The mail functionality is handled using *a new REXX command*: **SendEMail**.

The syntax for the SendEMail command is:

```
rc=SendEMail(parameter1 [, parameter2])
```

The following simple example demonstrates the required commands:

```
rc = SendEMail("Initialize")
rc = SendEMail("SetToEmail","susan.andersson@support.com")
rc = SendEMail("SetFromEmail","jones.smith")
rc = SendEMail("SetServer","company.com")
msgText=PanGetCtlData("MESSAGE")
rc = SendEMail("SetMsgBody",msgText)
rc = SendEMail("Send")
```

Return Values

The value returned by the SendEMail command will be one of the following:

0 Command executed successfully

1 Error

9 Undefined command

SendEMail Commands

The commands that can be used for sending mail are listed below, followed by a detailed description of the functions in alphabetical order. The commands are not case sensitive. The required commands are shown in bold font.

Initialize	Initializes the mail functionality.
SetToEmail	Sets the recipient's email address.
SetCCEmail	Sets the carbon copy recipient's address.
SetBCCEmail	Sets the blind carbon copy recipient's address.
SetFromEmail	Sets the sender's email name.
SetServer	Sets the sender's domain name.
SetHost	Sets the host address for the outgoing mail server
SetSSL	Sets the usage of SSL for sending mail.
SetPort	Sets the port for sending to message, typically with SSL (110 or 995, default is 25).
SetAuthUser	Sets the authentication user for the receiving mail system (if required).
SetAuthPassword	Sets the authentication password for the receiving mail system (if required).
SetPasswordUserID	Set the authentication password from user ID.
SetSubject	Sets the mail subject.
SetMsgBody	Sets the message body.
SetContentType	Sets the content type for the body message, e.g. "text/html". Default is "text/plain".
Send	Send the message.
GetLastError	Retrieves the last error.

26.9 Alphabetical List of REXX SendEMail API Commands

GetLastError

rc=**SendEMail***("GetLastError")*

Gets a description of the last error**.**

Return Value

A description of the last error.

Example

```
rc = SendEMail("GetLastError")
```

Initialize

rc=**SendEMail***("Initialize")*

Initializes the mail functionality. This command has to be executed before any of the other SendEMail commands can be executed.

Return Value *0*

Initialization successful

Example

```
rc = SendEMail("Initialize")
```

Send

rc=**SendEMail***("Send")*

Send the mail.

Return Value *0*

Mail sent successfully.

1

An error occurred during the send operation. A description of the error can be fetched with the SendEMail("GetLastError") command.

Example

```
rc = SendEMail("Send")
```

SetAuthPassword

rc=**SendEMail***("SetAuthPassword",password)*

Sets the authentication password for the receiving mail system.

Return Value *0*

Password set successfully.

SetAuthUser

rc=**SendEMail***("SetAuthUser",userID)*

Sets the authentication user for the receiving mail system.

Return Value *0*

User ID set successfully.

SetBCC

rc=**SendEMail***("SetBCC")*

Sets the blind carbon copy recipient's address. More than one address can be specified by separating the addresses with a comma (,) or semi-colon (;).

Return Value *0*

Address set successfully.

1

The SendEMail("Initialize") command has not been executed.

Examples

```
rc = SendEMail("SetBCC","jones.smith@company.com")
rc = SendEMail("SetBCC","jones.smith@company.com,
eve.andersson@company.com")
```

```
rc = SendEMail("SetBCC","jones.smith@company.com;
eve.andersson@company.com")
```

SetCC

rc=**SendEMail***("SetCC")*

Sets the carbon copy recipient's address. More than one address can be specified by separating the addresses with comma (,) or semi-colon (;).

Return Value *0*

Address set successfully.

1

The SendEMail("Initialize") command has not been executed.

Examples

```
rc = SendEMail("SetCC","jones.smith@company.com")

rc = SendEMail("SetCC","jones.smith@company.com,
eve.andersson@company.com")

rc = SendEMail("SetCC","jones.smith@company.com;
eve.andersson@company.com")
```

SetFromEmail

rc=**SendEMail***("SetFromEmail")*

Sets the sender's email name, or the sender's complete email address. If this command only sets the senders email name, then it will be combined with the server's name provided with the *SetServer* command.

Return Value *0*

Address set successfully.

1

The SendEMail("Initialize") command has not been executed.

Examples

```
rc = SendEMail("SetFromEmail","jones.smith")
rc = SendEMail("SetServer","company.com")

rc = SendEmail("SetFromEmail","jones.smith@company.com")
```

SetPasswordUserID

rc=**SendEMail***("SetPasswordUserID",userID)*

Sets the authentication password for the receiving mail system. The password is retrieved from the NetPhantom system using the user identification specified.

Return Value *0*

Password set successfully.

1

User ID cannot be found.

Examples

```
rc = SendEMail("SetPasswordUserID","user_1")
```

SetReplyEmail

rc=**SendEMail***("SetReplyEmail")*

Sets the reply-to email address.

Return Value *0*

Address set successfully.

1

The SendEMail("Initialize") command has not been executed.

SetHost

rc=**SendEMail***("SetHost")*

Sets the host address for the outgoing mail server. If the address to the outgoing mail server is *mail*, then there is no need to specify it since *mail* is the default address. But if the address is something else, it must be specified.

Return Value *0*

Host address set successfully.

1

The SendEMail("Initialize") command has not been executed.

Examples

```
rc = SendEMail("SetHost","mymail.company.com")
```

SetContentType

rc=**SendEMail***("SetContentType")*

Sets the mail content type for the message body, e.g. “text/html”. The default type is “text/plain”.

Return Value *0*

Successful.

Example

```
rc = SendEMail("SetContentType","text/html")
```

SetMsgBody

rc=**SendEMail***("SetMsgBody")*

Sets the message body for the email.

Return Value *0*

Message body set successfully.

1

The SendEMail("Initialize") command has not been executed.

Example

```
MsgText=PanGetCtlData("MESSAGE")
rc = SendEMail("SetMsgBody",msgText)
```

SetPort

rc=**SendEMail***("SetPort",port)*

Sets the port for the receiving mail system when sending mail (default is 25, 110 or 995 is common for SSL). The port number must be between 1 and 65535.

Return Value *0*

Port set successfully.

SetServer

rc=**SendEMail***("SetServer")*

Sets the sender's server name. If the value provided by the *SetFromEmail* command only contains the senders email name, then it will be combined with the server's name to create the sender's complete email address.

Return Value *0*
Server name set successfully.

1
The SendEMail("Initialize") command has not been executed.

Example

```
rc = SendEMail("SetFromEmail","jones.smith")
rc = SendEMail("SetServer","company.com")
```

SetSSL

rc=**SendEMail***("SetSSL")*

Sets the mail sending protocol to use SSL.

Return Value *0*
SSL protocol set successfully.

SetSubject

rc=**SendEMail***("SetSubject")*

Sets the subject for this email.

Return Value *0*
Subject set successfully.

1
The SendEMail("Initialize") command has not been executed.

Example

```
rc = SendEMail("SetSubject","Test mail")
```

SetToEmail

rc=**SendEMail***("SetToEmail")*

Sets the recipient's email address. More than one address can be specified by separating the addresses with a comma (,) or semi-colon (;).

Return Value *0*
Address set successfully.

1
The SendEMail("Initialize") command has not been executed.

Examples

```
rc = SendEMail("SetToEmail","jones.smith@company.com")
```

26.10 The NetPhantom Java API for Mail

The NetPhantom Java API for mail can be used to access the mail utility from java classes that are added to a NetPhantom application. For example, the mail utility could be accessed from a user window in this way.

For a detailed description of the NetPhantom Java API for mail, please read the javadoc for the *SendMail* class in the *se.entra.phantom.server.extra* package.

26.11 The NetPhantom Standalone Mail Utility

It is also possible to use the NetPhantom's mail utility as a standalone application directly from the command line on the server. This makes it possible to send a mail directly from a script (`.BAT` file in a Windows environment).

The standalone mail utility is started as:

```
java -cp NetPhantomServer.jar;mail.jar
          se.entra.phantom.server.extra.MailCommand
          parameter parameter … parameter
```

The available parameters are (required parameters are shown in bold-italic):

TO, CC, BCC, ***FROM***, **REPLY**, **SERVER**, HOST, PORT, SSL, USERID, PASSWORD, SUBJECT, ***MESSAGE***, NLSMESSAGE, TY**PE**, FILE, DEBUG

TO Parameter

This parameter is used to specify the recipient's mail-address. If there is more than one recipient, separate them with a comma (,) or semi-colon (;). This parameter is required.

```
to:jones.smith@mycompany.com
```

CC Parameter

Sets the recipient of carbon copy. If there is more than one recipient, separate them with a comma (,) or semi-colon (;).

```
cc:jones.smith@mycompany.com
```

BCC Parameter

Sets the recipient of carbon copy. If there is more than one recipient, separate them with a comma (,) or semi-colon (;).

```
bcc:jones.smith@mycompany.com
```

FROM Parameter

Sets the sender's email address. This will be combined with the *server* parameter to form the complete email address. This parameter is required.

```
from:rose.anderson
```

REPLY Parameter

Sets the mail's reply-to address.

```
reply:jones.smith@mycompany.com
```

HOST parameter

Sets the host address for the outgoing mail server. If the address to the outgoing mail server is *mail*, then there is no need to specify it, since *mail* is the default address. But if the address is something else, it has to be specified.

SERVER Parameter

Sets the domain for the sender's email server. This will be combined with the *from* parameter to form the complete email address. This parameter is required.

```
server:mycompany.com
```

PORT Parameter

Sets the port used for mail sending (default is 110, 995 is often used for SSL).

SSL Parameter

Set the use of SSL to send the mail. This parameter can be 0, 1, true or false.

USERID Parameter

Sets the authentication user for the receiving mail system.

PASSWORD Parameter

Sets the authentication password for the receiving mail system.

SUBJECT Parameter

Sets the header for the email. If the header contains spaces, it should be inside double-quotes (").

```
subject:Test
"subject:This is a test"
```

MESSAGE Parameter

Sets the email message. If the message contains more than one word, is should be inside double-quotes ("). The message parameter can be replaced with the *file* parameter.

```
"message:This is a small test message."
```

TYPE Parameter

Sets the content type of the message body. The default type is “text/plain”, and typically this could be set to “text/html”.

FILE Parameter

Sets the message part of the email, when the mail message has been stored as a text file in ISO-8859-1 encoding. Set the parameters value to the filename.

```
file:message.txt
```

Note: The *message* and *file* parameters should not be used at the same time. If both parameters are used by accident, then the message will be set to the message specified by the *message* parameter.

NLSMESSAGE

Sets a test email message body and subject. The message contains national characters, Arabic and Chinese text. The subject contains e.g. Swedish characters. There is no parameter.

```
nlsmessage
```

DEBUG

Sets debugging mode. The MailCommand will print out the entire conversation with the mail server. There is no parameter.

```
debug
```

27 The NetPhantom REXX API for SMS

The NetPhantom REXX API can be used for sending SMS. This makes it possible to give any NetPhantom application SMS functionality. The SMS functionality is handled using *a new REXX command:* **SendSMS.**

The syntax for the SendSMS command is:

```
rc = SendSMS( parameter1 [, parameter2, parameter3] )
```

The following simple example demonstrates the two available commands:

```
/* check if sms is available. */
/* 0 is available, 1 is not available */
rc = SendSMS('INITIALIZE')
if(rc==1) then
  do
    rc = Message(2,4,'Unable to send SMS. The SMS utility is
    not present on this server.')
    return 0
  end

/* Send message to all recipients */
msg    = PanGetCtlData('MSG')
count  = PanListGetNum('LIST1')

do i=count by -1 to 1
  curNum = PanListGetData('LIST1',i)
  rc     = SendSMS('SENDMESSAGE',curNum,msg)
end
Return 0
```

27.1 Return Values

The value returned by the SendSMS command will be one of the following:

0
Command executed successfully

1
The SMS utility is not installed

2
Error

9
Undefined command

27.2 SendSMS Commands

The commands that can be used for sending an SMS are listed below, followed by a detailed description of the functions in alphabetic order. The commands are not case sensitive.

Initialize	Checks whether the SMS-utility is installed
SendMessage	Sends the message

27.3 Alphabetical List of REXX SendSMS API Commands

Initialize

rc = **SendSMS**(*"Initialize")*

Receives the current status of the SMS utility.

Return value

0
SMS utility is installed.

1
SMS utility is not installed.

Example

```
rc = SendSMS('Initialize')
```

SendMessage

rc = **SendSMS**(*"SendMessage"*)

Sends a message.

Return value

0
The message was successfully sent.

1
SMS utility is not installed

2
An error occurred when trying to send the message.

Example

```
rc = SendSMS('SendMessage','+491751235467','This is a
message text')
```

27.4 The NetPhantom Standalone SMS Utility

You can also use the SMS utility as a standalone application directly from the command line on the server. This makes it possible to send an SMS directly from a script (e.g. a `.bat` file in as Windows environment).

The Standalone SMS utility is started as:

```
java se.entra.phantom.server.extra.SMSCommand
```

The available parameters are (all parameters are required):

- ApplicationId
- Server
- Port
- Key file name
- Phone number (s)
- Message

ApplicationId Parameter

This parameter is used to specify the application id to use when connecting to the Bluebird server. The application ID must be registered on the Bluebird server.

Server Parameter

The IP address of the Bluebird server.

Port Parameter

The port to connect to on the Bluebird server.

Key File Name Parameter

The path to the key file. The key file is created on the Bluebird server when registering the application ID. The file is then manually transferred to the NetPhantom server. At each logon to Bluebird the key file is sent as a Java file object to Bluebird in order to authenticate the user (the NetPhantom server in this case).

Phone Number(s) Parameter

A single or comma-separated list of phone numbers that will receive the message. A phone number must begin with the international prefix (e.g. +46 for Sweden, +49 for Germany) in order to work.

Message Parameter

The message text. The maximum length of any single SMS message is 160 characters. If the message contains more than one word, it should be inside double quotes (").

The NetPhantom OneLogin

The NetPhantom OneLogin utility reduces the number of times a user needs to logon to a system when he/she runs more than one application on the same machine. When a user performs a successful logon procedure from a certain machine, he does not have to repeat that logon procedure if

1. The new client connects to the same server as the previous client.
2. The new client uses the same host.
3. The application implements the OneLogin feature.
4. A logon has already been performed using OneLogin.

27.5 Requirements

There are no requirements on the client. OneLogin is added to the application during development.

Extra Classes

All required classes are provided with normal NetPhantom installation.

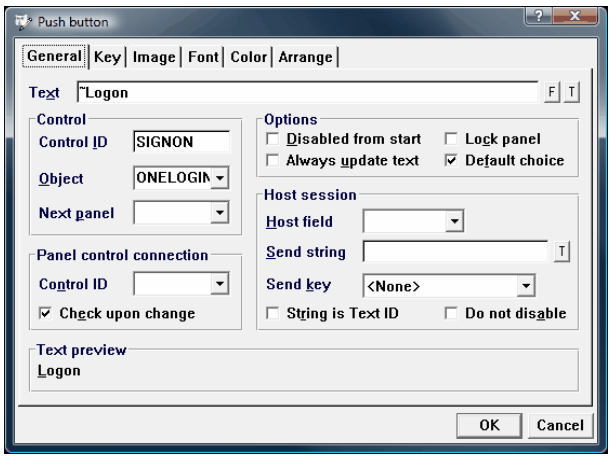
27.6 Installation

The Logon Host Screen

In order to work, three fields have to be identified in the host: `USERID`, `PASSWORD` and `SYSTEM`. They reference the user entry field, the password entry field and the system output field.

The Logon Panel

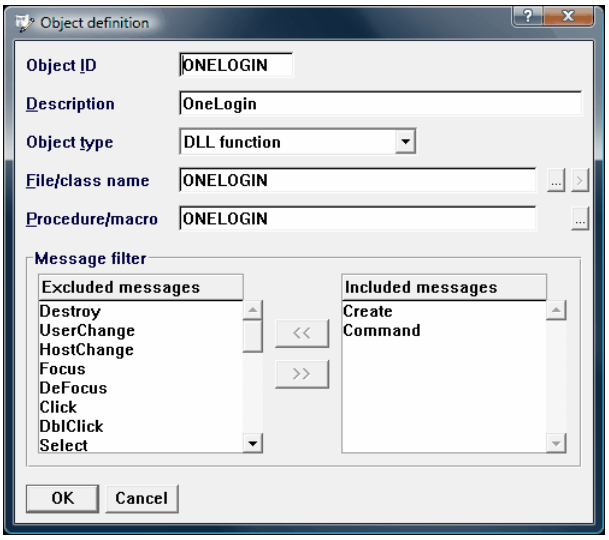
The logon panel must have a control (preferably a button) with id `SIGNON`. This control connects to the DLL function called `ONELOGIN`. The button should not send any other keystroke to the host.



Definition of the pushbutton connected to the DLL function.

The DLL Object

The developer creates a dummy DLL function called `ONELOGIN`. In the object definition the object ID, the filename and the procedure/macro are set to `ONELOGIN`. In the message filter, the included messages are `CREATE` and `COMMAND`.



DLL function definition

27.7 Functions

Add a Client Manually

If you have an application or system that does not make it possible to use the OneLogin utility, you may manually add clients to the table held by ONELOGIN via a call in REXX code to the ONELOGIN DLL. A code example is given at the end of this chapter.

Password Change

OneLogin supports password changes via REXX code. This updates the password used by OneLogin, not the password of the host system. Therefore, updating a password in OneLogin should only be done after a successful change of password on the host system. Code example is given at the end of this chapter.

27.8 What Happens Inside

The NetPhantom control connects to the ONELOGIN DLL function which calls the corresponding java class with `CREATE` as action parameter. This class holds a table with client information based on IP addresses. An internal check against the NetPhantom server removes all clients from the table that are no longer active. Then, it searches the table for a record of the client IP address. If no record is found, then it is treated as a new logon.

If it finds a matching record, an additional check matches the value specified in the `SYSTEM` host field. If it is the same system, OneLogin retrieves the user ID and password from the table, fills these values in to the host fields named `USERID` and `PASSWORD` and sends an `ENTER` keystroke to the host. This creates a successful logon.

If it is a new logon, a normal logon procedure must be performed. Once the `USERID` and `PASSWORD` entry fields are filled, the user pushes the button connected to the ONELOGIN DLL that calls the java class with the `COMMAND` action parameter. It checks the client IP address, but no record will be found. An `ENTER` keystroke is sent and a normal logon is done using information entered by the user. If the logon is successful the client IP address is stored in the table with the user ID, the password, and the system name.

27.9 REXX API Methods

Change password

rc = **CallObject('ONELOGIN','CHGPWD','CMD',** *params* **)**

Changes the user password.

The code should get called when a successful change of password in the host is done.

Call the ONELOGIN object with control ID 'CHGPWD', action 'CMD' and a string containing the system, the old password, and the new password.

The parameters should be space delimited. For example:

```
params = system || ' ' || oldPwd || ' ' || newPwd
```

Return value

0

Command executed successfully.

1

The last parameter contains an invalid number of sub strings.

2

Error.

3

No record found for this ip-address.

Example

In the example below the `SYSTEM` variable is set to empty string since this change should apply for several host systems.

```
Parse Arg argId, argMsg, argStr

Signal On Error
Signal On Syntax

/*system = 'ENTRA4'*/
system = ' '
oldScreen = HostGetScreen()

oldPwd = HostGetFld('',F_6)
newPwd = HostGetFld('',F_1)

rc = HostSend('@E')
rc = HostWait()

newScreen = HostGetScreen()

if (oldScreen != newScreen) then do
rc = CallObject('ONELOGIN','CHGPWD','CMD',system || ' ' || oldPwd || '
' || newPwd )
            say 'Return code from password change: ' || rc
            end
else do
            say 'Password change not initiated.'
            end

Return 0
```

Add user

rc = CallObject('ONELOGIN','ADDUSR','CMD', params **)**

Adds the current user to the table holding connected clients.

The parameter string should be a space-delimited string containing system, user and password (in this order). For example:

```
params = system || ' ' || user ' ' || password
```

If this information should apply to every host system, the system variable can be omitted.

Note: It is only possible to add the user currently running the application, as OneLogin uses an internal client ID variable to identify a client and this variable is not available when writing the REXX-code.

Return value

0

Command executed successfully.

1

The last parameter contains an invalid number of sub strings (must be two or three).

2

Error.

Example

```
Parse Arg argId, argMsg, argStr

Signal On Error
Signal On Syntax

system  = HostGetFld('',SYSTEM)
usr     = HostGetFld('',USERID)
pwd     = HostGetFld('',PASSWORD)

if (usr='' | pwd='') then do
             Message(2,2,'Please provide both userID and password')
             return 0
end

rc = CallObject('ONELOGIN','ADDUSR','CMD',system || ' ' || usr || ' '
|| pwd )
say 'Return code from addUser: ' || rc

if (rc=0) then
             rc = HostSend('@E')
Return 0
```

28 Session Pooling

The NetPhantom Session Pool is typically used to speed up the initial connection to a host session for an application. It also enables e.g. pre-logon including additional navigation and re-use of previous sessions, as well as “proper” logoff when a client is disconnected.

Each pool is tied to a runtime *application ID* and a *host ID*. When a runtime application is configured for pooling, it will receive a session from the pool for the requested host session ID. When multiple runtime applications are used, it is sufficient to define a pool for one of the applications.

28.1 Session Pool Actions

The session pool performs the following steps (from here, “pool” is used instead of “session pool”):

- At start of a pool, an XML definition and script file is loaded and parsed (see the *Session Pool Script* section below).
- When the pool is started, the minimum number of sessions is started in parallel using a thread for each session. Each session then starts one Telnet thread (for 3270 and 5250), or two when a 3270 printer is associated with the (3270) terminal session. The *Start* script is then executed.
- When a session is started but not in use by an application, it can optionally be “pinged” at regular intervals in order to “keep the session alive”, e.g. by making sure that the session is not logged off from the host system. The *Ping* script is then executed.
- At initial terminal session request (e.g. when application starts or when *HostConnect* is issued), a terminal session is taken from the pool if and only if the session is *Checked*, i.e. defined as OK to use. Only sessions that are in a ready state are used, i.e. a session will not be *Checked* if it is currently performing a *Start*, *Ping*, *Reclaim* or *Dispose* script.
- When the client connection is closed, lost, or timed out due to inactivity, the session will be reclaimed to the pool. The *Reclaim* script is then executed. If the pool already contains the maximum number of sessions, the *Dispose* script is executed. The *Dispose* script is also executed when an administrator closes a pool or terminates a connection.
- The pool always makes sure to have the minimum number of sessions available in the pool. At any point in time, several sessions can be started if the count of sessions becomes less than the minimum amount.

When a pool is started, its runtime application will be disabled until the pool has successfully started all terminal sessions and completed running the *Start* script.

If a pool runs out of *Checked* sessions, the requesting application (thread) will be blocked until the session has been successfully *Started* and *Checked*.

> **Note:** There is no limit to the number of terminal sessions for applications, but the time required to start the application would be strongly increased in terms of session start-up if no session is available in the pool (*Started* and *Checked*). This will result in long delays for end users.

The following actions will cause a change of the **Connection ID** associated with a session:

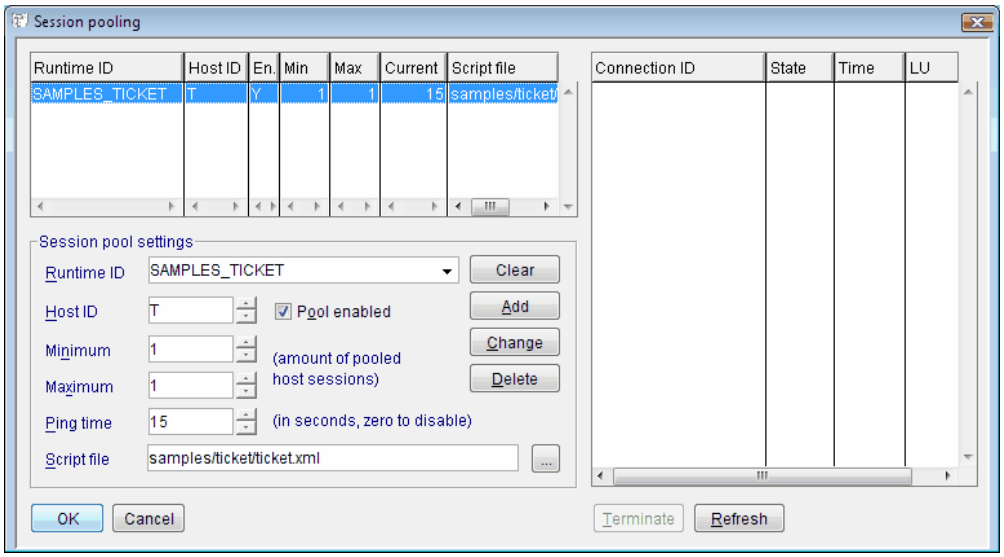
1. Pooled session transferred to a requesting application, after processing of the *Check* script.
2. Successful reclaim of a session to the pool, after processing of the *Reclaim* script.

28.2 Server Shutdown or Restart

When the NetPhantom Server is shut-down or restarted, all pool sessions are disposed of using the *Dispose* script, unless the *Immediate* shut-down or restart option is selected.

28.3 Configuration of Session

The Session Pooling system is defined using the Server Administration program using the menu item **Server – Configure – Session pooling** (or **Server – Advanced – Session Pooling…** in the NetPhantom Editor).



The panel is also used to display the current session pool status.

To create a new pool, optionally press **Clear**, then fill in the required information in the **Session pool settings**, followed by **Add**.

To change a pool's settings, select it in the top-left list box, change the information in **Session pool settings**, then press **Change**.

The setting **Pool enabled** only applies to the pool being "started", i.e. if an application is referred to in the **Runtime ID**, this application *will not be disabled* if the pool is disabled (normal processing as if no pool existed then applies).

Note that all changes in the terminal session pools are performed immediately!

To display the status of all session's status for a pool, click in the top-right list box followed by **Refresh**.

The states displayed for a session are: *Starting*, *Start*, *Ready*, *Ping*, *Check*, *Reclaim*, *Dispose* and *In use*.

You can terminate a host session by clicking on the session(s) in the rightmost list box to stop followed by **Terminate**. You cannot stop a session owned by an active application session. The **Connection ID** in the list box is equivalent to the **Client** (ID) number for active clients.

The (elapsed) **Time** for a session is reset when a client requests the host session and when the session has been reclaimed.

28.4 Session Pool Scripting

When a session is started, "pinged", checked for validity, reclaimed or is about to be disposed, the *DefaultSessionPoolingHandler* reads what to do for the action in question.

This is done in a script definition file. The script definition file is written in XML as described below.

Each session pool can have a NetPhantom Runtime file associated with it. This eases the interaction with the host system, to check for matching screens, use host fields, etc. The Runtime file is only used for screen identification and host field processing, i.e. no panels or objects (REXX or other) are used.

File Format

The file must have the following format (note that XML is case sensitive as opposed to HTML):

```
<?xml version="1.0"?>

<SessionPoolingScript
  version="1.00"
  implclass="se.entra.phantom.server.DefaultSessionPoolingHandler"
  runtimeid="TUTOR">
<!--runtime file can also be specified as: runtimefile="/filename" -->

<extensions>
  <function name="SaveGlobalHostData"
            method="saveGlobalHostData"
            class="com.acme.SessionPool"/>

  <function name="YourHostCheck" method="yourHostCheck"/>
</extensions>

<actions>
  <action name="START" maxtime="seconds">
    <script>
      *** the script here ***
    </script>
  </action>

  <action name="PING" maxtime="seconds">
    <script>
      *** the script here ***
    </script>
  </action>

  <action name="CHECK" maxtime="seconds">
    <script>
      *** the script here ***
    </script>
  </action>

  <action name="RECLAIM" maxtime="seconds">
    <script>
      *** the script here ***
    </script>
  </action>

  <action name="DISPOSE" maxtime="seconds">
    <script>
      *** the script here ***
    </script>
  </action>
</actions>
</SessionPoolingScript>
```

Any main tags not recognized are not processed. The *action* tag must contain a *script* tag describing what “instructions” are to be processed. For the *action* tag, a parameter *maxtime* can be specified in seconds. If the script has not completed within this time limit, the session is considered invalid and is disposed of (not calling the script of the *dispose* action).

Action Tags

There should always be five action tags specified in the script definition file, having different name values as listed below:

START	Called when a session starts in the pool. The start time is set before the terminal session is created. The script is called once the host session is unlocked and has a non-empty host screen. Typically, the script would navigate in the host so as to reach the "starting point screen".
PING	Called periodically by the session pool manager (optional setting). This enables, for example, the script to perform some action in the host to make sure it is not going to be disconnected by or logged from the host. The script should also verify that the host is at the "starting point screen".
CHECK	Called to check if a session is OK for usage for a new client session. Typically, the script would check that the host session is unlocked and not in error, and that the current screen is the "starting point screen".
RECLAIM	Called when a client session is closed in order to reclaim the host session into the pool. This enables the script to, for example, back out from a series of host screens to a starting point. When the reclaim action is called, the user who owned the host session could have navigated "deep down" in the host system.
DISPOSE	Called prior to disposing of the terminal session. Typically, the script would logoff from the host system in a controlled way. Note that the current host screen could be "anywhere" in the host system.

Script Tags

The *DefaultSessionPoolingHandler* is a Java class provided as a default handler for the session pooling. It provides by default the option to add external functions in new classes (using the *extensions* section in the XML file). This is done using *Custom Method Calls* (see refering section below).

An extending Java class can also override it to provide a higher or more specialized functionality. For more advanced processing, the script tag handling can be overridden. For more information, see JavaDoc for the *DefaultSessionPoolingHandler* class.

The defined *Script Tags* by default are listed below.

Tag – SCREEN

```
<screen name="ABC" [multiple]>
  *** instructions ***
</screen>
```

Description:

The *instructions* inside the block are executed if the current host screen name matches uniquely. The option *multiple* allows other screen names to match as well. If this option is set, the current host screen is set to the screen named in the tag.

If the screen does not match, execution continues at the tag below.

Returns:

True if the screen matches (uniquely), *false* otherwise.

Tag – CONDITIONS

```
<conditions [negate]>
  *** conditions ***
</conditions>
```

Description:

The *conditions* tag should normally be used together with the *if*, *elseif* and the *while* tags. All tags inside the "conditions block" are combined with the logical *and* operator. The *conditions* tag can also be used to stop processing in a block as soon as e.g. a *wait* tag returns *false*.

The *negate* option logically negates each condition tag processed before the logical *and* operation is performed. That is, a return code of *true* in a condition tag stops execution of the remaining tags, returning a value of *false*.

Returns:

True if all tags inside the *conditions* block are true, *false* otherwise. Processing of execution of the tags inside the block is immediately stopped if one of the tags returns *false* (or *true* for the *negate* option).

Tag – IF

```
<if>
  <conditions [negate]>
    *** conditions ***
  </conditions>
  *** instructions ***
</if>
<elseif>
  <conditions [negate]>
    *** conditions ***
  </conditions>
  *** instructions ***
</elseif>
<else>
  *** instructions ***
</else>
```

Description:

The test instruction for if/elseif/while can be written as a block of several instructions using the *conditions* tag or using any other single instruction tag returning either true or false. The *elseif* block can be repeated any number of times or not specified at all. The *else* block may only be specified once and can also be omitted.

Returns:

True if there was an *if* or *elseif* condition that was true, *false* otherwise. This applies when *no instructions* have been carried out. The return code is otherwise the return code of the last executed instruction (i.e. tag).

Tag – WHILE

```
<while>
  <conditions [negate]>
    *** conditions ***
  </conditions>
  *** instructions ***
</while>
```

or the *do-while* style:

```
<while>
  *** instructions ***
  <conditions [negate]>
    *** conditions ***
  </conditions>
</while>
```

Description:

The *while* instructions are executed if the *conditions* are true or if no *conditions* tag can be found, the first instruction following the while tag is used as "a single conditions" instruction (see test instruction above).

Returns:

The return code of the last executed instruction (i.e. tag).

Tag – BREAK

```
<break [true]/>

<break [false]/>
```

Description:

Breaks out of a *while* loop. If not inside a *while* loop, this tag would have the same processing as a *return* tag, optionally setting the return code to *true* or *false*. If the return code is not set, the last return code is used.

Returns:

True or *false* options if the parameters are set, otherwise the return code of the last instruction (i.e. tag).

Tag – SET

```
<set hostfield="NAME" [line="nn"] text="text"/>

<set hostfield="NAME" [line="nn"] userid="USERID"/>

<set cursor [relative] column="xx" row="yy"/>

<set cursor hostfield="NAME" [line="nn"] [pos="charpos"]/>

<set timeout="milliseconds"/>
```

Description:

This tag can be used to set data in a host field of the currently matching host screen. It can also set the cursor position to *xx* and *yy* with *relative* as an option (used when inside a host pop-up window).

The *userid* option will set the password for the *"USERID"* that is specified in the NetPhantom Users using the Server Administration program.

Note: all numeric parameters are "one-based" for *line*, *pos*, *column*, *row*.

Note: the currently matching screen is only set when the *screen* or *wait screen* tag has been executed prior to this tag (this also applies to host pop-up window recognition only executed with these tags).

An error with this function will resume execution at the next *onerror* tag.

Returns:

True if the function caused a change, *false* otherwise (i.e. cursor didn't move, timeout didn't change, host field already contained the text to set).

Tag – SEND

```
<send string="string"/>

<send userid="USERID"/>
```

Description:

This tag is used to send a series of keystrokes to the terminal session formatted according to EHLLAPI codes (e.g. "userid@Tpassword@E").

The *userid* option will send the password for the *"USERID"* that is specified in the NetPhantom Users using the Server Administration program.

The *send* tag always waits for the session to be unlocked (due to e.g. sending an AID key such as Enter).

An error with this function will resume execution at the next *onerror* tag.

Note that the *type-ahead* of keystrokes is enabled only if this option is enabled for host sessions using the Server Administration program.

Returns:

Always *True*.

Tag – RESET

```
<reset/>
```

Description:

This tag resets any error conditions present in host session, after e.g. typing data in a protected field or receiving a System Message for 5250.

Returns:

True if the function was successful or no error state existed for the session, *false* otherwise (host was not in an error state that could not be reset due to e.g. communications failure).

Tag – WAIT

```
<wait time="milliseconds"
      [hostlocked] [hoststable]/>
<wait screen="SCREENNAME" [multiple]
      [hostlocked] [hoststable]
      [time="milliseconds"]
      [timeout="milliseconds"]/>

<wait hosttext="text" [nocase] [relative]
      column="xx" row="yy" [width="nn" [wildcard]]
      [hostlocked] [hoststable]
      [time="milliseconds"]
      [timeout="milliseconds"]/>

<wait hostfield="NAME" text="text" [nocase]
      [line="nn"] [wildcard]
      [hostlocked] [hoststable]
      [time="milliseconds"]
      [timeout="milliseconds"]/>
```

Description:

The *wait* tag is used to wait a specified time in milliseconds or a maximum time in milliseconds (default 60.000 milliseconds or changed by the *set timeout* tag).

It is also used to wait for the host lock state, the host session to be stabilized within the timeout value, a screen to match or a host text to be present (using absolute or relative screen position, also including hidden/non-display text as an option). When the *time* option is defined with waiting functions for the host (e.g. *screen* and/or the *hostlocked* or *hoststable* options), this function will wait an additional *time* before returning, e.g. if the host screen is changed or the cursor moved during this time, the wait time is reset to zero.

Note that the (optional) parameters *column*, *row* and *line* are “one-based” numerical values.

The *wildcard* match option enables text to be matched with Windows-like wildcards:

?	is any character,
*	is any number of characters,
^	is the escape character, i.e. the meaning of ‘?’ and ‘*’ becomes the actual character that directly follows,
^^	two escape characters become the escape character itself (‘^’).

This tag can also be used as a *condition* tag.

Returns:

True when the condition is reached within the timeout time or *false* if not.

Tag – HOSTERROR

```
<hosterror/>
```

Description:

An error state is typically set if data is input in a protected field, there is some communications failure or, for 5250, if a “System Message” is present. To remove this condition, the reset key (“@R”) must be sent or the *reset* tag executed.

Returns:

This tag returns *true* if the host is presently in an error state. If no error condition is present, *false* is returned.

Tag – ONERROR

```
<onerror>
  *** instructions ***
</onerror>
```

Description:

When an error condition occurs in a script, the next *onerror* tag instructions are executed. If the session is not in an error state, this tag is skipped.

Returns:

True if there was an error condition when the tag was reached, *false* otherwise.

Tag – LOG

```
<log [level="nn"] text="text" [lasterror]/>
```

Description:

Logs an event in the NetPhantom Event Log. The value *nn* can be *0*=informational (default), *1*=warning or *2*=error. It can also be the first character, i.e. *I*, *W* or *E*.

If the *lasterror* parameter is specified, the last encountered error message for the terminal session is added. *Note for 5250 sessions:* a system error message always begins with the text “5250: ”.

Returns:

Always *True*.

Tag – TRACE

```
<trace text="text" [lasterror] [dumpscreen]/>
```

Description:

Adds text to the trace file.

If the *lasterror* parameter is specified, the last encountered error message for the terminal session is added. Note: for 5250 sessions, a system error message always begins with the text “5250: ”.

If the parameter *dumpscreen* is present, the host screen character data is appended.

Returns:

Always *True*.

Tag – RETURN

```
<return true/>

<return false/>
```

Description:

Stops execution in a script with the specified return code.

Returns:

True or *false* depending on the parameter.

Tag – DISPOSE

```
<dispose/>
```

Description:

Immediately disposes of this pooled session without executing a single additional tag.

Returns:

This tag does not return a value, an exception is thrown and the script is terminated.

28.5 Custom Function Calls

Custom function calls are encouraged to fulfill advanced issues with session pooling, e.g. to logon with a particular user and perform tasks that are not handled with the ones provided in the NetPhantom class *DefaultSessionPoolingHandler*.

There are two types of functions: a function inside an extending class of *DefaultSessionPoolingHandler* or a "standalone" function in another external class.

The definition in the XML file header contains the following:

```
<implclass>classNameAndPackage</implclass>

<extensions>
  <function name  ="SaveGlobalHostData"
            method="saveGlobalHostData"
            class ="com.acme.SessionPool"/>

  <function name="YourHostCheck" method="yourHostCheck"/>
</extensions>
```

The implementing class (above *ClassNameAndPackage*) is normally `se.entra.phantom.server.DefaultSessionPoolingHandler`.

Extension functions are specified within the `<extensions>` tag.

Function Calls in External Classes

These custom functions are static in the defined class and only has the following "abstract" function style:

```
/**
 * Your method (saveGlobalHostData) description.
 *
 * @throws  SessionPoolingDisposed  when a script has been
 *                                  disposed.
 */
public static void saveGlobalHostData(
  SessionPoolingScriptData data)
  throws SessionPoolingDisposed
  {
  // Each function should increase the current tag element
  // for the next statement of execution.
  data.setNextCurrentElement();
  }
```

The class is loaded when the XML file loaded and parsed by the session pool. There is no instance of this class created at any time by the session pooling framework. Access to pooled session data or the global session pooling data is done using the parameter of class `SessionPoolingScriptData`. The return code of an external function is either *true* or *false* and this value is set in the variable `returnCode` in the `SessionPoolingScriptData` class instance (a new instance of this class is created each time a new script [start, ping, check, etc] is executed for a pooled session).

The extending functions are defined in the XML file inside the `<extensions>` tag as:

```
<function name  ="NameOfFunctionInScript"
          method="methodName"
          class ="packageAndClassName"/>
```

If the `class` parameter is not specified, it is assumed that the function is located inside the implementing class of the session pooling handler (functions inside the implementing class are not static, they are declared as instance methods, i.e. without "*static*").

This is the code of a function implementing a simple trace operation:

```
/**
 * Adds text to the trace file.
 *
 * Return code: always true.
 *
 * @throws  SessionPoolingDisposed  when a script has been disposed.
 */
public static void myTrace(SessionPoolingScriptData data)
  throws SessionPoolingDisposed
  {
  // Get the text for the trace and print it to the screen.
  String text=data.getParamString("text");
  System.out.println(">> myTrace: "+text);

  // Set return code.
  data.returnCode=true;

  // Set next statement to execute.
  data.setNextCurrentElement();
  }
```

External functions of this type normally cover special needs to start and handle a pooled session.

Extending the Session Pooling Handler

Before you consider extending the default session pooling handler, try to solve the problems with "normal" external functions. If this is not sufficient, follow the procedure described below:

1. Create a handler class extending from `se.entra.phantom.server.DefaultSessionPoolingHandler`.

2. Override the static method `createSessionPoolingData` if you require extra global data. If you override this method, you can return a class instance of a class extending `se.entra.phantom.server.SessionPoolingData`. See the JavaDoc of `SessionPoolingData` and `createSessionPoolingData`, and their source code for more information if this is required.

3. Implement and change the behavior of the session pooling handler by overriding appropriate method calls, etc.

An instance of your extending class will be created for each pooled session and if you have created an extending class for `SessionPoolingData`, a new instance of that class will be created for each session pool, i.e. will be global for all your handler instances in the same session pool.

29 Appendix A – Code Pages

Under Windows and OS/2, the OEM and ANSI code pages are used, and for Mainframe and AS/400, it is the EBCDIC code pages. Other operating systems can use other code pages.

NetPhantom only uses Unicode but needs to access text files and runtime application files stored in OEM or ANSI code pages.

The NetPhantom Server contains Java classes to handle code page conversion to and from Unicode. The code page definition is done in `server.ini`.

29.1 OEM Code Pages

Name	Description
Cp437	United States/Australia/New Zeeland/South Africa
Cp737	Greek
Cp775	Baltic
Cp850	Latin-1
Cp858	Latin-1 + Euro
Cp852	Latin-2
Cp860	Portuguese
Cp861	Icelandic
Cp862	Hebrew
Cp863	Canadian French
Cp864	Arabic
Cp865	Nordic
Cp866	Russian
Cp868	Pakistan
Cp921	Latvia/Lithuania
Cp922	Estonia
Cp1098	Iran (Farsi)/Persian

29.2 ANSI Code Pages

Name	Description
Cp1252	Windows Western Europe (Latin-1)
Cp1250	Windows Eastern Europe (Latin-2)
Cp1251	Windows Cyrillic
Cp1253	Windows Greek
Cp1254	Windows Turkish
Cp1255	Windows Hebrew
Cp1256	Windows Arabic
Cp1257	Windows Baltic
Cp1258	Windows Vietnamese
MS874	Windows Thai

29.3 EBCDIC Code Pages

Name	Description
Cp037	EBCDIC USA/Canada(Bilingual/French)/Netherlands/ Portugal/Brazil/Australia
Cp1140	EBCDIC USA/Canada(Bilingual/French)/Netherlands/ Portugal/Brazil/Australia + Euro
Cp1046	Open Edition US EBCDIC
Cp273	EBCDIC Austria/Germany
Cp1141	EBCDIC Austria/Germany + Euro
Cp277	EBCDIC Denmark/Norway
Cp1142	EBCDIC Denmark/Norway + Euro
Cp278	EBCDIC Finland/Sweden
Cp1143	EBCDIC Finland/Sweden + Euro
Cp280	EBCDIC Italy
Cp1144	EBCDIC Italy + Euro
Cp284	EBCDIC Spain/Catalan/Latin America (Spanish)
Cp1145	EBCDIC Spain/Catalan/Latin America (Spanish) + Euro
Cp285	EBCDIC United Kingdom/Ireland
Cp1146	EBCDIC United Kingdom/Ireland + Euro
Cp297	EBCDIC France
Cp1147	EBCDIC France + Euro
Cp420	EBCDIC Arabic
Cp424	EBCDIC Hebrew
Cp500	EBCDIC 500V1 (Belgium/Canada/Switzerland)
Cp1148	EBCDIC 500V1 (Belgium/Canada/Switzerland) + Euro

Cp838	EBCDIC Thailand extended SBCS
Cp855	EBCDIC Cyrillic
Cp857	EBCDIC Turkish
Cp869	EBCDIC Modern Greek
Cp870	EBCDIC Multilingual Latin-2 (Serbia)
Cp1153	EBCDIC Multilingual Latin-2 (Serbia) + Euro
Cp871	EBCDIC Iceland
Cp1149	EBCDIC Iceland + Euro
Cp874	EBCDIC Thai
Cp875	EBCDIC Greek (New)
Cp918	EBCDIC Pakistan (Urdu)
Cp1025	EBCDIC Multilingual Cyrillic (Bulgaria/Bosnia/Herzegovina/Macedonia-FYR) + Euro
Cp1026	EBCDIC Latin-5 (Turkey)
Cp1155	EBCDIC Latin-5 (Turkey) + Euro
Cp1097	EBCDIC Iran(Farsi)/Persian
Cp1112	EBCDIC Latvia/Lithuania
Cp1156	EBCDIC Latvia/Lithuania + Euro
Cp1122	EBCDIC Estonia
Cp1157	EBCDIC Estonia + Euro
Cp1123	EBCDIC Ukrane
Cp1158	EBCDIC Ukrane + Euro

29.4 ISO Code Pages

8859_1	ISO-8859-1 - Latin1 (West European)
8859_2	ISO-8859-2 - Latin2 (East European)
8859_3	ISO-8859-3 - Latin3 (South European)
8859_4	ISO-8859-4 - Latin4 (North European)
8859_5	ISO-8859-5 - Cyrillic
8859_6	ISO-8859-6 - Arabic
8859_7	ISO-8859-7 - Greek
8859_8	ISO-8859-8 - Hebrew
8859_9	ISO-8859-9 - Latin5 (Turkish)
8859_11	ISO-8859-11 - Thai
8859_13	ISO-8859-13 – Latin7 – (Baltic)
8859_15	ISO-8859-15 – Latin9

29.5 Windows Code Pages

Cp1250	Windows Eastern Europe (Latin-2)
Cp1251	Windows Cyrillic
Cp1252	Windows Western Europe (Latin-1)
Cp1253	Windows Greek
Cp1254	Windows Turkish
Cp1255	Windows Hebrew
Cp1256	Windows Arabic
Cp1257	Windows Baltic
Cp1258	Windows Vietnamese

29.6 DOS Code Pages

Cp437	DOS United States/Australia/New Zeeland/South Africa
Cp737	DOS Greek
Cp775	DOS Baltic
Cp850	DOS Latin1
Cp858	DOS Latin1 - Euro
Cp852	DOS Latin2
Cp860	DOS Portuguese
Cp861	DOS Icelandic
Cp862	DOS Hebrew
Cp863	DOS Canadian French
Cp864	DOS Arabic
Cp865	DOS Nordic
Cp866	DOS Russian
Cp868	DOS Pakistan
Cp869	DOS Modern Greek

30 Appendix B – REXX Conversion

This section describes how REXX code is converted into NetRexx that is then compiled into Java byte code. This process is normally taking place in the background if the Editor option **Background compile** is set, otherwise during the **Compile distribution** process.

The output of this process will be the extracted REXX sources in the application, other files such as `.nrx`, `.class`, `.crossref` and possibly `.err` files in the case of a conversion problem.

30.1 REXX to NetRexx

IBM's interpreted script language REXX is a language well suited for the development of application specific programs. However, REXX' heritage is in old mainframe script languages, and REXX is not based on "new" language features such as object-oriented systems development and strong typing, exception handling and internetworking. Therefore, IBM has developed a new – compiling – script language called NetRexx. In many ways, NetRexx is like REXX in that it is based on string manipulation, and NetRexx has functions for easy parsing of strings similar to REXX. However, NetRexx mainly resembles Java. Examples of this are NetRexx' stronger variable scopes, its variable typing and the absence of goto. The intermediate output of the NetRexx compiler is Java source.

Due to the quantity of existing REXX application programs, there was a need for a program to convert a REXX script to a NetRexx class. This program is called REXX to NetRexx, or R2NR for short.

The objective of this section is to show how this program is used and to describe the basic components of its operation. Some of the operations performed by R2NR are quite technical and require knowledge of compiler construction and programming language syntax and semantics, especially the *goto conversion*. These techniques are not at all described in depth in this document. The reader should be familiar with REXX, NetRexx and Java.

R2NR Syntax

R2NR can be invoked from the command line if manual REXX conversion is required:

```
r2nrnt [options] rexx-file
```

R2NR will give the output of the NetRexx converted source program on standard output, in the form of a NetRexx class. The options that can be set are:

Option	Description
-c class name	Sets the name of the class. If not specified, it will be the filename of the REXX file without path and extension.
-d declaration file	This option can be used to create or update a declaration file (see below). If not specified, no declaration file will be updated.
-h header lines file **-H** explicit header lines	The lowercase option reads header lines from the specified filename. These header lines are output before the class definition, unaltered. The uppercase will read one header line directly from the command line. Remember to quote the header line properly.

-i inheritance string	This option sets the string that will follow the class declaration. This string will typically contain information about inheritance, i.e., "extends BaseClass". Remember to quote the string properly.
-m first method's name	Sets the name of the first method. If not specified, it will be named "start". All other methods will be named after their respective REXX label.
-p post lines file **-P** explicit post line	Similar to **–h** and **–H**, but these lines will follow the class definition.
-r translation rules file **-R** explicit translation rule	Adds translation rules. See below.
-s	Skip construct (say or trace).
-3	Forces the first method to have three parameters.

The Basic Algorithm

The program converts the REXX code in five steps.

1. Reading of source file and lexical parse. *(lex)*
2. LALR-parse and creation of program tree. *(yacc)*
3. Reading of variable and function conversion rules.
4. Extraction of class global variables.
5. Output of all methods, in NetRexx syntax.

Points 1 and 2 are based on the REXX standard[1].

Translation Rules

To facilitate the incorporation of translated REXX code into a NetRexx framework, functions can be substituted to match the new calling sequences. For example, the syntax to convert the value of variable `A` to uppercase is in REXX "`upper(A)`" but in NetRexx "`A.upper`". This conversion can be accomplished by the translation rule:

```
upper(*a) -> *a.upper
```

Notice the syntax: REXX code to the left of the arrow, NetRexx code to the right. `*a` is a placeholder which can match any expression in the source code. One can also specify to match against an explicit string within single quotes "`'`" or symbols. Notice that the left side is not case sensitive, but the right side will preserve capitalization. On the left side, default values can be specified:

```
foo(*bar, *baz='0') -> *baz.foo(*bar.bar(*bar))
```

Here, "`foo(3+4)`" will be translated into "`'0'.foo((3+4).bar(3+4))`". It is also possible to let the last argument match zero or many arguments:

---

[1] Proposed ANSI standard X3J18-199X.

```
foo(*bar, *baz[]) -> foo(*bar, *baz)
```

In this case, "`foo(0, 1, 3, 5)`" will be translated into code similar to:

```
temporary = 3
temporary[0] = 1
temporary[1] = 3
temporary[2] = 5
foo(0, temporary)
```

Declaration File

This section does not require reading. It has been provided as an example of what can be done using the declaration file.

To facilitate automatic creation of supportive classes for a large NetRexx framework, R2NR can create a "declaration file", which consists of one line per method defined in the translation in the syntax:

```
class-name method-name number-of-parameters
```

Lines previously in the declaration file relating to the class currently being compiled are removed. The other lines are not altered. So, one course of action could be:

1. Translate all REXX scripts while creating a declaration file.
2. Create a supportive "link class" automatically using information from the declaration file.
3. Create rules that make use of the "link class". This should also be done automatically.
4. Translate all REXX scripts again, using the new rules.

If the declaration file contains the line "`CLASSONE start 2`", the "link class" may include code such as:

```
method call_CLASSONE_start(arg1, arg2)
  ...create new CLASSONE instance called c1...
  c1.start(arg1, arg2)
```

The new rules may include:

```
classone(*arg1,*arg2) -> call_CLASSONE_start(*arg1,*arg2)
```

Variable Name Substitution and Variable Declaration

In REXX, the keywords `PROCEDURE` and `EXPOSE` are used to create a local scope. In NetRexx, a variable is local if it is declared (through assignment) in a local scope. The translation maintains scope locality by making all procedure variables end in "`$local`". Global variables end in "`$global`" in order to avoid reserved NetRexx and Java words.

All simple variables are initialized to the string of their uppercase name. That is, the REXX program "`SAY hello`", will output "`HELLO`". R2NR maintains this fashion by always assigning the variables to their uppercase names. So, the REXX program will be translated into (wrapper code excluded):

```
HELLO = "HELLO"
SAY HELLO
```

All stem variables must be initialized in NetRexx, and this is also taken care of by R2NR, by assigning the empty string to the stem variables.

Goto's

REXX includes the goto semantics, while NetRexx does not. However, R2NR supports goto's by converting all code including goto labels to a large loop/select construct. This is done by rewriting `IF`, `DO`, `SELECT`, `ITERATE`, and `LEAVE`. This leads to quite unreadable code, and it may perform somewhat more slowly, but since the resulting NetRexx/Java code is compiled, it runs faster than the REXX code anyway.

"Conditional" Comments

In order to further facilitate semi-automatic conversions, there is a way to insert separate, unconverted, statements into the NetRexx code. There is also a way of making R2NR ignore some REXX code.

To insert REXX statements that will be converted to NetRexx, but should not be used in the normal REXX program, use the following syntax:

```
/*NETREXX:   REXX statement(s)   */
```

To insert NetRexx statements, use the following syntax:

```
/*NRX:   NetRexx statement(s)   */
```

To make NetRexx ignore REXX statements, use this syntax:

```
/*REXX(*/   REXX statements(s)   /*)REXX*/
```

Notice that these methods work on statement level, not on expression level. So, if the variable `a` should be assigned the value 10 in NetRexx, but the value 11 in REXX, one should write:

```
/*NRX:       a=10           */
/*REXX(*/    a=11    /*)REXX*/
```

(As opposed to `a=/*NRX:10*//*REXX(*/11/*)REXX*/`).

Unsupported REXX Features

- REXX has an exception system which can ignore or queue an exception, or condition. This mechanism is supported by the way in which the interpreter works. It is not supported by Java and thus not by NetRexx. The constructions CALL ON/OFF are not supported.
- REXX can interpret strings as code with the INTERPRET instruction. This is not supported.
- The DROP keyword is not supported.
- PUSH/PULL/QUEUE are not supported but can be substituted with other functions.
- TRACE is not supported. It is ignored.
- STREAM function is supported. Some limitations exist (see the Java API documentation for more information).

31 Appendix C – NetPhantom Considerations

This section describes various issues and considerations when using NetPhantom.

31.1 Using Java Threads in NetPhantom

The NetPhantom Server runs a multi-user environment, and the threads are grouped into a system group and per-user. Whenever there is a serious problem in the Java code for a particular user, causing a "crash" (i.e. an unhandled exception or error), the NetPhantom thread takes care of this by logging events and then shutting down the particular client session.

The NetPhantom threads all use the class `se.entra.phantom.server.ServerThread` that is extended from the Java `Thread` class. It contains information that is relevant when performing tracing (using `se.entra.phantom.server.BinaryTrace`) or event logging (in `se.entra.phantom.server.EventManager`).

In other words, *ALWAYS use the `ServerThread` class* when you start new threads. See the samples below of how to use the `ServerThread` instead of the Java `Thread`.

```
// Directly start a new thread.
new Thread(threadRunner).start();

// Or a longer way...
Thread t=new Thread(threadRunner);
t.start();
```

Using ServerThread instead would look like this:

```
import se.entra.phantom.server.ServerThread;
...code...

// Directly start a new thread.
new ServerThread(threadRunner,"MyThread").start();

// Or a longer way...
ServerThread t=new ServerThread(threadRunner,"MyThread");
t.start();
```

Internally, creating a new thread using `ServerThread` will check if the current thread is of the `ServerThread` instance. If it is, the NetPhantom Client information (in `se.entra.phantom.server.ClientConnectionData`) is retrieved from the current thread and initializes the new thread with this data.

32 Appendix D – Client National Language

The NetPhantom Client provides a mechanism for national language support (NLS). NetPhantom is delivered with English language as default but can be customized as described below.

All strings are stored in the Server file `server.phm` as `CLInnnn=xxx` entries, apart from strings in the static class `se.entra.phantom.client.LocalizedTextStrings`. This class may be replaced with another language.

See the JavaDoc of the class *se.entra.phantom.client.LocalizedTextStrings* for more information.

Here is the listing of class code:

```
package se.entra.phantom.client;

/**
 * The localized text string class is used to keep all
 * strings that cannot come from the server after it
 * has been contacted.
 *
 * <p>This class is static only, so all texts for the client
 * use a single language.
 */
public class LocalizedTextStrings
{
  /**
   * Title message in "Connecting..." dialog box.
   */
  public static final String connectTitle=
    "Connecting to Server";

  /**
   * First message in "Connecting..." dialog box.
   */
  public static final String loading=
    "Loading, please wait...";

  /**
   * Second message in "Connecting..." dialog box.
   */
  public static final String contactingServer=
    "Contacting the primary NetPhantom Server";

  /**
   * Third message in "Connecting..." dialog box.
   */
  public static final String contactingBackupServer
    ="Contacting the backup NetPhantom Server";

  /**
   * Loading application message in "Connecting..." dialog box.
   */
  public static final String loadingApplication=
    "Loading application...";

  /**
   * Initial communications failure with NetPhantom Server.
   */
  public static final String communicationsFailure
    ="Communications failure with NetPhantom Server. Press OK to exit.";

  /**
   * Initial communications failure with NetPhantom Server.
   */
  public static final String communicationsFailure1
    ="Failure connecting to NetPhantom Server:";

  public static final String server=
    "Server";
```

```
  public static final String error=
    "Error";

  public static final String communicationsFailure2=
    "Press OK to exit.";

  /**
   * Exit message heading.
   */
  public static final String exitTitle=
    "Exiting NetPhantom Client";

  /**
   * Parameter error message.
   */
  public static final String paramError=
    "Parameter error. Closing the program...";
}
```

33 Appendix E – User Exits

The NetPhantom Server contains several "user exits" that allow default implementation of e.g. event notification to be rewritten to transport all events to the IBM Tivoli framework.

This section shows the different user exits.

33.1 The Event User Exit

```
package se.entra.phantom.server;
import java.io.IOException;

/**
 * The server has a global event manager running in a thread. A static method
 * queueEvent is used to create and queue an event. The thread will then empty
 * the event queue and pass the events on to the ServerInterface. This user exit
 * is dynamically loaded from a definition in SERVER.INI.
 */
public interface EventInterface
{
  /**
   * Processes a single queued event, e.g. writes it to a file.
   *
   * <p>Note that the methods in this interface can be called from different
   * threads, so the implementing code may have to be e.g.
   * <code>synchronized</code>.
   *
   * @throws  IOException  if e.g. the file cannot be created.
   */
  public abstract void logEvent(ServerEvent e) throws IOException;

  /**
   * Gets the current event filename.
   *
   * @returns  null  for no filename.
   */
  public abstract String getEventFileName();

  /**
   * Gets the current event file length.
   *
   * @returns  -1  if no file is used.
   */
  public abstract long getEventFileLength();

  /**
   * Sets the maximum file size of a trace file. If set to zero, no
   * limit is used.
   *
   * @returns  true  if successful, false for invalid size (100 KB minimum
   * [102400]).
   */
  public abstract boolean setMaximumFileSize(long maxSize);

  /**
   * Gets the maximum file size of a trace file. If set to
   * zero, no limit is used.
   *
   * @returns  -1  if no file is used.
   */
  public abstract long getMaximumFileSize();

  /**
   * Rotates the event information if stored in e.g. a file.
   *
   * <p>Note that the methods in this interface can be called from different
   * threads, so the implementing code may have to be e.g.
   * <code>synchronized</code>.
   *
   * @returns  true  for success or false for failure.
   */
  public abstract boolean rotateInformation();
```

```
  /**
   * Disposes of the class instance.
   *
   * <p>Note that the methods in this interface can be called from different
   * threads, so the implementing code may have to be e.g.
   * <code>synchronized</code>.
   */
  public abstract void dispose();
}
```

33.2 The LU Mapper User Exit

```
package se.entra.phantom.server;
import java.io.IOException;
import java.net.InetAddress;

/**
 * This interface must be supported by the user exit that handles
 * the 3270/5250 LU Mapping.
 */
public interface LUMapperInterface
{
  /**
   * Gets the 3270/5250 LU name from an internet address
   * (and/or user name).
   *
   * <p>The <code>hostIPAddress</code> parameter specifies the entry
   * in the "server.ini" under the [host] section. The <code>sessionID</code>
   * is zero for session 'A', one for session 'B', etc.
   *
   * @return
   *    null              for client not allowed,
   *    <br>empty string  for no LU name,
   *    <br>LUName        as a String.
   */
  public abstract String mapClientToHostLU(ClientConnectionData ccd,
                                           String host,
                                           int port);

  /**
   * This method is called whenever a client of a particular LU name
   * has disconnected from the server (note: not from the host session,
   * because the NetPhantom API provides methods such as HostConnect
   * and HostDisconnect).
   */
  public abstract void clientDisconnected(ClientConnectionData ccd);

  /**
   * Use this method to return a list of client mappings. The string array
   * returned is twice the size of the LU mappings. The first string contains
   * the client IP address and the second contains the LU name.
   *
   * @return
   *    null              for no loaded list exist,
   *    <br>array         of strings.
   */
  public abstract String [] getClientLUMappings();

  /**
   * Refreshes the contents in case of e.g. a filename used to handle
   * the mapping.
   *
   * @throws  IOException  if the file cannot be loaded.
   */
  public abstract void refresh() throws IOException;

  /**
   * Called prior to deferencing in order to clean up.
   */
  public abstract void dispose();
}
```

33.3 The User Authentication User Exit

```
package se.entra.phantom.server;

/**
 * This interface must be supported by the user exit that handles
 * user authentication.
 */
public interface UserAuthenticationInterface
{
  /**
   * Checks if UserID and Password is required for a particular
   * client IP-address.
   */
  public abstract boolean isUserIDPasswordRequired(ClientConnectionData ccd);

  /**
   * Verifies a UserID and Password entered on the client by a user.
   *
   * Return  null              if OK,
   *         empty string      to prompt user to change the password or
   *         an error string   that will be displayed on the client
   *                           side.
   */
  public abstract String verifyUserIDPassword(ClientConnectionData ccd,
                                              String password,
                                              int previousRetryCount);

  /**
   * Changes the password for a UserID.
   *
   * Return  null  if OK or an error message string.
   */
  public abstract String changePassword(ClientConnectionData ccd,
                                        String oldPassword,
                                        String newPassword);

  /**
   * Upon an error for verifyUserIDPassword or changePassword,
   * this method is called to check if further retries should be
   * done, or if the client connection should be dropped.
   */
  public abstract boolean isRetryValid(ClientConnectionData ccd,
                                       int previousRetryCount);

  /**
   * Called to perform clean-up.
   */
  public abstract void dispose();
}
```

33.4 The Telnet Host Interface

The Telnet Host Interface is defined as:

```
package se.entra.phantom.server;

import java.io.IOException;

/**
 * The Host Telnet communication. It establishes the connection to the
 * host and initiates the 3270 or 5250 data stream.
 */
public interface TelnetHost
{
  /**
   * Enables the Telnet NEW-ENVIRON TN5250E negotiation.
   */
  public abstract void enableNewEnviron();

  /**
   * Initializes the Telnet session after the construction
   * for a 3270/5250 terminal type list.
   *
```

```
   * @exception  IOException   if an I/O error occurs.
   */
  public void initialize(TelnetInterface i,String terminalList) throws
IOException;

  /**
   * Opens the connection to the host at a specified port number.
   *
   * @exception  IOException   if an I/O error occurs.
   */
  public abstract void open(String host,int port) throws IOException;

  /**
   * Closes the current connection.
   */
  public abstract void close();

  /**
   * Writes an amount of bytes. The data is appended with an
   * end-of-record (IAC EOR). Each 0xFF character in the buffer
   * is escaped with 0xFF (i.e. IAC IAC).
   *
   * @exception  IOException   if an I/O error occurs.
   */
  public abstract void write(short buf[], int n) throws IOException;
}
```

The default interface is Telnet and is created in the *TelnetThread* class.

Callbacks to the NetPhantom Server engine are done using the *TelnetInterface*:

```
package se.entra.phantom.server;

import java.io.IOException;

/**
 * Use this interface to implement a telnet session owner.
 */
public interface TelnetInterface
{
  /**
   * The host accepted the options for the terminal
   * type and went into BINARY mode. This means that
   * 3270/5250 data stream will follow. Inform the host session
   * listener of a connect change.
   *
   * @exception  IOException   if an I/O error occurs.
   */
  public abstract void hostAcceptedDataStream(String terminalType)
                  throws IOException;

  /**
   * Processes the inbound 3270 or 5250 data stream.
   *
   * @exception  IOException   if an I/O error occurs.
   */
  public abstract void processInboundDataStream(short [] inputBuf,
                                                int inputBufLen)
                  throws IOException;

  /**
   * The session has been disconnected due to link failure or if
   * the remote host disconnected due to e.g. a time-out.
   */
  public abstract void sessionDisconnected(String description);

  /**
   * Returns the host destination address for the session.
   */
  public abstract String getHostAddress();

  /**
   * Returns the Telnet port number.
   */
  public abstract int getHostPort();
```

```
  /**
   * Gets the session ID (A is 0, B is 1...).
   */
  public abstract int getSessionID();

  /**
   * Gets the client connection data.
   */
  public abstract ClientConnectionData getClientConnectionData();
}
```

34 Appendix F – File Transfer from Server to Client

The following section describes the use of the RCONSOLE_FILETRANSFER user window and how to implement a server-to-client file transfer.

34.1 Panel Part

The panel needs a user window with DLL-name: `RCONSOLE` and Entry point name: `FILETRANSFER`. If the user should be able to specify which file to transfer and where to save it, a pair of input controls are needed as well. The code example below assumes one button to trigger the transfer, one Cancel button, one button to specify the target file, one button to specify the source file and one entry field to manually specify the source file. It also uses a second user window to display a progress bar (`RCONSOLE_PROGRESSBAR`).

34.2 Server Part

The developer needs to write a new class, which should handle the file transfer (see code example). This class must implement the `AdminFileTransferListener` interface in order to communicate with the user window class `RCONSOLE_FILETRANSFER` and its peer on the client side (see diagram 1).

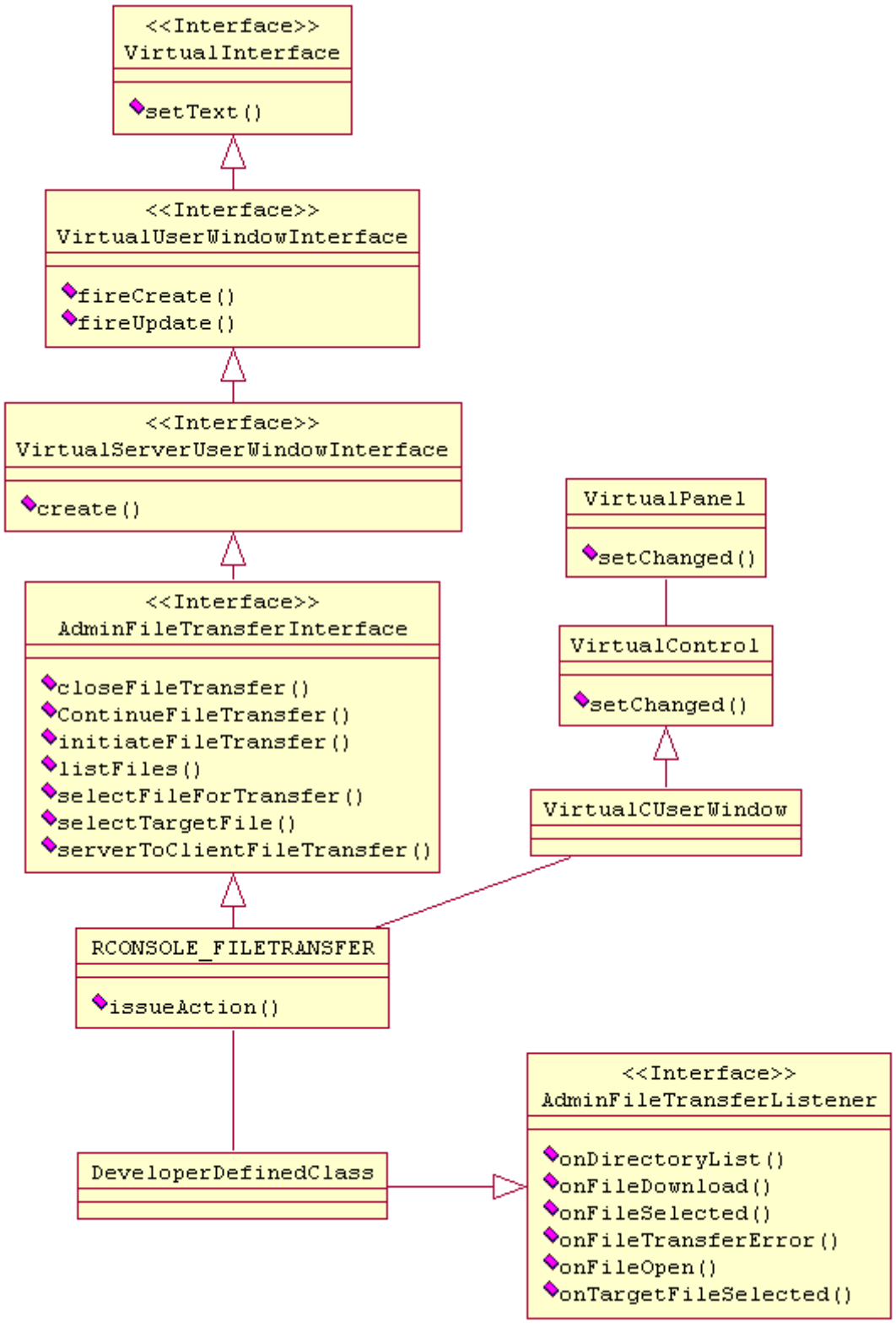
To start with, your class needs to know the path to the file to be transfered and also the path on the client where the file should be saved. Relative and/or absolute paths can be used. Both paths must point to a file. The method `RCONSOLE_FILETRANSFER.SelectTargetFile` (calls back `to AdminFileTransferListener.onTargetFileSelected`) can be used to prompt the user to specify a target file on the client. The `AdminConfigSelectFile` class can be used to prompt the user to specify a source file on the server.

Next step is to start sending the file using the `RCONSOLE_FILETRANSFER.serverToClientFileTransfer` method. The file should be sent in chunks of 4k (byte[4096]). When the client is ready to receive next chunk of data, the `AdminFileTransferListener.onFileOpen` is called which should trigger the next call to `RCONSOLE_FILETRANSFER.serverToClientFileTransfer.`

The example code below uses the above mentioned ways to prompt the user for source and target file. The user also has the option to specify the file paths manually. After validation of the paths, it starts itself as a new thread that takes care of the call to `RCONSOLE_FILETRANSFER.serverToClientFileTransfer.` This thread sends a chunk of data and then waits until it is notified by the `AdminFileTransferListener.onFileOpen` method. When the file transfer is complete a message box is displayed to the user. If any error occurs on the client side during the file transfer the `AdminFileTransferListener.onFileTransferError` method is called.

34.3 Class Structure on the Server

Diagram 1. The `DeveloperDefinedClass` and its relation to other important classes. (Not all methods are included.)



35 Appendix G – Component Cross-Reference

The NetPhantom Cross-Reference is still present for compatibility for customers processing the report XML file programmatically. NetPhantom and Eclipse provides interactive cross-referencing that is far more powerful…

NetPhantom Cross-Reference generates a report of all components of a NetPhantom application. By component we mean any kind of constituent part of a NetPhantom application such as an output field, panel, host screen, text ID or a list box.

The input for NetPhantom Cross-Reference is a Runtime file, and the output is a document written in XML format. The document output declares which components the NetPhantom application in question consists of and lists the references between those components.

A NetPhantom application consists of many components, such as controls and objects[2] as well as the references between those components. When developing and maintaining a NetPhantom application, it is important to keep track of all these parts.

It is possible to get a lot of information about the components of a NetPhantom Application in the NetPhantom Editor. Still, there are components that are harder to get a total overview about from the information presented in NetPhantom. The reason for this is that some components, such as objects and text file base IDs, can be referenced by almost any component in an application. Thus, information about the total number of references must be searched for in as many places as there are references. Searching for references in this manner can be very time-consuming!

The purpose of NetPhantom Cross-Reference is to give NetPhantom developers a clear overview of all components and references in a NetPhantom application.

35.1 Report Structure and Data

The data structure of the XML document is based on "the parameters (components) that can be considered to be most interesting for a NetPhantom application developer to get information about as a complement to the information found in the NetPhantom tool. Those components are the ones about which it is more difficult to get a total overview from the information presented in the NetPhantom tool.

The information in the XML document is presented in four main sections according to the four main subjects:

1. Objects
2. Popup menus
3. Text file
4. Application

Objects

Object is a term representing objects created and used in a NetPhantom Application.

The section for information about objects (<OBJECT>) consists of information about its data (<DATA>), i.e. name, or type, and about its references (<REFERENCE>). References

[2] Phantom objects are not objects in the usual sense. Their role is to define and make possible the connection between Phantom controls and panels and external code

consist of information about all components in the application that have a reference to the object in question.

Example:

```
<OBJECT>
  <DATA>
    <ID>PF2PF2</ID>
    <LONGID>OBJECT/PF2PF2/1</LONGID>
    <TYPE>OBJ_REXXMACRO</TYPE>
    <FILE>PF2PF2</FILE>
    <PROCEDURE>PF2PF2</PROCEDURE>
  </DATA>
  <REFERENCES>
    <REFERENCE>
    <TYPE>LISTBOX</TYPE>
    <ID>LIST</ID>
    <MAINID>CUSTSRCH</MAINID>
    <LONGID>LISTBOX/LIST/CUSTSRCH/0</LONGID>
  </REFERENCES>
</OBJECT>
```

Popup Menu

The section for popup menus (<POPUPMENUS>) presents information about popup menus created and used in the NetPhantom application.

The information about popup menus consists of information about its data (<DATA>), i.e. name, or type, it's popup menu items (<MENUITEM>) and details about its references (<REFERENCE>). References consist of all components in the application that have a reference to the popup menu in question.

Popup menu item data (<MENUITEM>) is also presented in this section.

Example:

```
<POPUPMENUS>
  <POPUPMENU>
    <DATA>
      <ID>SHOW</ID>
      <LONGID>POPUPMENU/SHOW/1</LONGID>
    </DATA>
    <CONNECTIONS>
      <CONNECTION>
        <ID>SHOW</ID>
        <REFERENCES>
          <REFERENCE>
            <TYPE>PUSHBUTTON</TYPE>
            <MAINID>CUSTSRCH</MAINID>
            <LONG_ID>PUSHBUTTON/SHOW/CUSTSRCH/2</LONG_ID>
          </REFERENCE>
        </REFERENCES>
      </CONNECTION>
    </CONNECTIONS>
    <MENUITEMS>
      <MENUITEM>
        <DATA>
          <ID>SHOW</ID>
          <LONGID>POPUPMENU/SHOW/1</LONGID>
          <OBJECT/>
          <HOSTFIELD/>
          <CONNECTED_CONTROL_ID/>
          <NEXTPANEL>
        </DATA>
      </MENUITEM>
    </MENUITEMS>
  </POPUPMENU>
</POPUPMENUS>
```

Text File

The section for text file (<TEXTFILE>) presents information about the text file used in the NetPhantom Application, i.e. *same_name_as_the_application_file*.PHM.

This section declares all base IDs (<ITEM>) and texts (<TEXT>) that are the text file's data (<DATA>) together with information about all components that have references (<REFERENCE>) to those base IDs.

Example:

```
<TEXTFILE>
  <ITEMS>
    <ITEM>
      <DATA>
        <TEXTID>ABC00002</TEXTID>
        <TEXT>This is a an example!</TEXT>
      </DATA>
      <REFERENCES>
        <REFERENCE>
          <ID>PANELD'S NAME</ID>
          <MAINID/>
          <LONGID>PANEL/ PANELD'S NAME /8</LONGID>
        </REFERENCE>
      </REFERENCES>
    </ITEM>
  </ITEMS>
</TEXTFILE>
```

Application

This section consists of three main parts.

1. Panels
2. Host data
3. Control bars

Panels

Information about panels (<PANEL>) consists of information about a panel's data, message boxes, menus and controls. Information about message boxes (<MESSAGEBOX>), menus (<MENU>) and controls (<CONTROL>) consist of no references, since these are the components that reference the objects, popup menus, screens, host fields and text file base IDs.

A control bar that is not saved and used as a template is presented in the <CONTROL> section for the panel to which it belongs.

Example:

```
<PANEL>
  <DATA>
    <ID>CUSTGEN</ID>
    <LONGID>PANEL/CUSTGEN/7</LONGID>
    <OBJECT/>
    <TITLE>~~CREDIT</TITLE>
    <TYPE>NBPAGE</TYPE>
    <SCREEN/>
    <NOTEBOOK_PANEL_ID>NOTEPAN</NOTEBOOK_PANEL_ID>
    <CONTROLCOUNT>9</CONTROLCOUNT>
    <FONT>0</FONT>
    <COLOR>0</COLOR>
  </DATA>
  <CONTROLS>
    <CONTROL>
      <DATA>
        <ID>COSTNO</ID>
        <LONGID>INPUTTEXT/COSTNO/CUSTGEN/0</LONGID>
        <MAINID>CUSTGEN</MAINID>
        <OBJECT/>
```

```
          <POPUPMENU>SHOW</POPUPMENU>
          <TYPE>ENTRYFIELD</TYPE>
          <HOSTFIELD>COSTNO</HOSTFIELD>
        </DATA>
      </CONTROL>
    <CONTROLS>
```

Host Data

Information about host data consists of information about:

1. Macros
2. Screens

Information about macro (<MACRO>) and screen (<SCREEN>) consists of information about their data (<DATA>) and references (<REFERENCE>) from panels to screens.

Screens in turn consist of host fields (<HOSTFIELD>) for which data (<DATA>) and references (<REFERENCE>) also are presented in this section.

Example:

```
<HOSTDATA>
  <MACROS>
    <MACRO>
      <DATA>
        <ID>LIST ALL CUSTOMERS</ID>
      </DATA>
    </MACRO>
  </MACROS>
    <HOSTSCREENS>
      <HOSTSCREEN>
        <DATA>
          <ID>SIGNON</ID>
          <IS_POPUP>NO</IS_POPUP>
        </DATA>
        <REFERENCES>
          <REFERENCE>
            <TYPE>PANEL</TYPE>
            <ID>SIGNON</ID>
            <MAINID>SIGNON</MAINID>
            <LONGID>PANEL/SIGNON/6</LONGID>
          </REFERENCE>
        </REFERENCES>
        <HOSTFIELDS>
          <HOSTFIELD>
            <DATA>
              <ID>DISPLAY</ID>
            </DATA>
            <REFERENCES>
              <REFERENCE>
                <TYPE>INPUTTEXT</TYPE>
                <ID/>
                <MAINID>SIGNON</MAINID>
                <LONGID>INPUTTEXT//SIGNON/3</LONGID>
              </REFERENCE>
            </REFERENCES>
          </HOSTFIELD>
```

Control Bars

Control bars (<CONTROLBAR>) that are used in the application and saved as templates are presented in this section (<DATA>). Data about a control bar's bar controls (<BARCONTROL>) is also presented here.

A control bar that not is saved and used as a template is presented in the <CONTROL> section for the panel to which it belongs (see *Panels* above).Example:

```
<CONTROLBARS>
  <CONTROLBAR>
    <DATA>
      <IS_TEMPLATEFILE>YES</IS_TEMPLATEFILE>
      <ID>STATBAR</ID>
      <LONGID>CONTROLBAR/STATBAR</LONGID>
    </DATA>
    <BARCONTROLS>
      <BARCONTROL>
        <DATA>
          <LONGID>BARCONTROL/1</LONGID>
          <MAINID/>
          <TYPE>PHANTOMCBARMESSAGE</TYPE>
        </DATA>
      </BARCONTROL>
```

35.2 How to Use the Cross-Reference

The generated document can be used in a browser such as Internet Explorer, which makes it possible to use search functions available in the browser itself.

Except for host screens, host fields, files and macros, observe that all components presented in the XML document have been given an extra identifier - the LongID - as a complement to the ID assigned to the component in the Runtime file. The purpose of this extra ID is to avoid situations in which components have no identifier and are thus very hard to identify in context. This situation can occur often as an ID is not always mandatory for many components of a NetPhantom application.

With this extra identifier, an application component is always identifiable.

Index

3

3270
Numeric field override 165
printer 164
3287 Host Printer Settings 167, 170

A

Access control 184, 186
SSL required option 185
Add user 334
ADH 109
Administration bind address 154
Administration port
Disable 154
Ansi code page 153
ANSI Code Pages 348
Application INI File 43, 44
Background color for application area 45
Bitmap in application area 45
Client properties 44
Tooltip text 46
Applications
Enable,Disable,Reload 203
Authentication timeout 176
Authenticity 106
AutoCreate 291
AutoCreateFromHostField 291

B

Backup Server 10
Backup Server(s) 143
Binary Trace Output 145
Bind address 174
Bluebird 329
Broadcast Message 209
Browser Capabilities 286
Browser Version
Disabled Attribute 287
BrowserIdentification.ini 286

C

CA Certificates 110, 113
Case sensitive file system 174
Certificate present on server 123, 124, 185
Certificates
Client 111, 121
Access control 122
Installing on the Client 125
Installing on the Server 123
Password replacement 122
User authentication 121
Uses 121
Certification Authority 106
CGI
Definition 178
CGI interfaces 84
Change password 333
Cipher suites 73, 107, 113
Export strength 107
Export-1024-strength 108
Export-strength 108
Mixed-strength 108
Non-export-strength 108
Selection guidelines 109
Strong 108
Ciphers
Asymmetric 106
Symmetric 105
ciphertext 105
Classpath
OS/2 27
UNIX 27
Windows 27
Client Application Properties 44
Client Architecture
Component model 237
Component types 238
Schematic 237
Client Certificate 185
Client Connections 208
Client inactivity monitor 154
Client National Language 357
Client Ping 197
Client Settings 171
Client Statistics 196
Client Types
Terminal Window 299
Cluster Controller 93
Memory requirements 94
Windows Service 93
Code Pages
ANSI 348
EBCDIC 348
OEM 347
codebase 79
http-address 82
Controller address
load balancing 177
Cookies 180
Copy HTML 290
Checkbox 295
Combo Box 293
Control Conversion 292
Entry Field 292
Group Box 297
List Box 296
Menu Item 297
MLE 293
Output Text 292
Push Button 294
Radio Button 295
Spin Button 294
Static Text 292

COPYHTML.INI 290
Country Settings 154
Cross-Reference
  Application 369
  Control Bars 370
  Host Data 370
  Panels 369
  Popup Menu 368
  Report Structure and Data 367
  Text File 369
  Usage 371
Cryptography
  Certification Authority 106
  Digital certificates 106
  Hash Functions 106
  Introduction 105
  Tamper-proofing 106

D

DDE 37, 48, 138
  NetPhantom Starter 48
Declaration File 353
decryption 105
Digital certificates 106
Digital IDs 121
Digital Signatures 106, 121
Directory structure
  NetPhantom installation 24
Disabled Attribute 287
Display Terminal Session 210
documentbase 79
Domains 181

E

EBCDIC Code Pages 348
EDH-DSA/RSA 108
EditComboToEntry 291
EditSpinToEntry 291
email_bcc 309
email_cc 309
email_doc 309
email_error_doc 310
email_field_separator 310
email_from 310, 311
email_output 311
email_server 311
email_subject 312
email_success_doc 312
email_to 312, 313
Enable HTTP tunneling 175
Enable Web Server 174
Encrypted SSL Connection 107
Encryption 79, 105, 106
Event Filter 144, 155
Event Log 213
Event Messengers 188
  Event filter 188
  Event messenger items 188
  manual configuration 73
Event User Exit 359
EventID 144
Export strength 107

F

File Manager 151
File Transfer 150
  Client to Server 152
  Security 150
  Server to Client 152
Firewalls 79, 80
Focus in HTML Forms 269
Font remapping 160
Font substitution 62, 157
  Defining a font substitution list 62
  Fixed Font Metrics 64
  Remapping a font 63
FORM 268, 269
Frames 284
  Change Current Frame 285

G

Generic Windows Service
  Installation 31
  Registry Keys 33
  Uninstallation 32
  Uninstalling the Generic NetPhantom Server Service 31
GetLastError 319

H

Hard Java VM Restart 200
Hard Restart 200
Hash Functions 106
Host codepage 163
Host session 143, 163, 341
  3270
    Custom Host Datastream Exit 68
    Numeric field override 68
    terminal types 67
  5250
    terminal types 67
  default 68, 163
  Definition 66
  Numeric field override 165
  Peer data 67, 164
Host Session IDs 37
  Definition 163
Host Session Manager 241
Host Statistics 196
HostOutputToInputText 291
HTML Application Errors 297
HTML cache 175
HTML forms
  ISO code page 153
HTML Forms

Focus 269
HTML included CGI 178, 271
HTML Integration
Browser Buttons 255
Browser Capabilities 286
Change Current Frame 285
Checkbox 295
Combo Box 293
Control Conversion 292
Control Reference 274
Copy HTML 290
Design Potential 253
Entry Field 292
FAQ 298
FORM Element 268, 269
Frames 284
General Variables 263
Group Box 297
HTML to NetPhantom References 274
HTML-Included CGIs 262
HTTP Related Variables 264
Hyperlink 283
Input Checkbox 278
Input Radio 278
Input Submit 276
Input Text 277
Invalid HTML Document 261
Limitations 253, 289
List Box 296
Markup Tags 261
Menu Item 297
Message Box HTML File 288
MLE 293
NetPhantom Application Variables 266
Output Text 292
Push Button 294
Radio Button 295
Restrictions 254
REXX Application Limitations 289
REXX Object Events 289
SCRIPT Element 270
Select 279
Server Side Include 261
Session Cookies 254
Spin Button 294
Static Text 292
Supported HTML elements 261
Table 280
Table behavior 281
Technology Overview 253
Terminology 275
Textarea 277
Variable Substitution 268
Variable Substitution in Scripts 263
Variable Substitution in Tags 263
Variable Substitution in Text 262
Variables in HTML Documents 263
HTTP 83
HTTP tunneling 38, 69
HTML document 81

I

Initialize 320, 328
ISO code page 153

J

Java Threads 355

K

key exchange algorithm 109
Keyboard Remapping 211
Ctrl key 212
key augmentation 211

L

Label Tag for Hidden Controls 288
Licence Manager
Configuration 11, 14
Configuration via Server Administration 12
License distribution 18
license.ini 11
Test Code 20
Licence System Dialog 12
License Manager
Starting the Licence Manager 11
Windows Service 11
License Manager 9
Backup Server 10
Primary Server 10
Server states 10
Server types 10
Licensing 9, 24, 150
SSL 24
Linux Port configuration
Linux Service
ListBoxBorder 291
ListHeaderToHTML 291
ListPageDownImage 291
ListPageUpImage 291
Load balancing 174, 177, 185, 186, 199, 274
Cluster Controller 93
Communication between Slave/Controller 90
Controller 177
Controller address 177
Controller info to slave 91
Introduction 89
Port number 177
Qualification 91
Slave 177
Slave info to controller 91
Techniques 89
LU Mapper User Exit 360
LU Mapping 146

M

Mail REXX API
GetLastError 319
Initialize 320
Send 320
SetBCC 320, 322, 323
SetCC 321
SetFromEmail 321, 322
SetMsgBody 322
SetServer 323
SetSubject 323
SetToEmail 323
Mail Utility 307
Fields for User Information 316
Form definition 316
Formatting the Mail 316
Formatting with a Template 317
MailForm Tags 307
Resetting the Form 316
REXX API 318
Sending the Form 316
Standalone 323, 324
Tag definitions 316
Template Command Tags 317
Mail Utility Template
set_field 317
unused_tag 318
MailForm CGI
CGI Definition 307
Resource Definition 307
MailForm Example 314
MailForm Tags 307
email_bcc 309
email_cc 309
email_doc 309
email_error_doc 310
email_field_separator 310
email_from 310, 311
email_output 311
email_server 311
email_subject 312
email_success_doc 312
email_to 312, 313
MailForm Template Example 318
Managing Groups 189
Managing Users 190
New User 190
Markup Tags in HTML 261
Max simultaneous requests 175
Merge-on-the-fly 41, 65, 72, 161, 179, 181
Message Box HTML File 288

N

National Language Support 147
Changing the language 147
NetPhantom Client
applet limitations 79
as standalone application 42
Inside a browser 42
Start
Parameters 37
with SmartApplet 42
NetPhantom Compile Settings 250
NetPhantom Considerations
Java Threads 355
NetPhantom Server
Start 22, 26
Batch files 28
Return codes 27
NetPhantom Server engine
Callbacks 362
NetPhantom Starter 37, 133
Application Chooser Definition File 138
Backup servers 135
Installation on Client 134
Installation on Server 137
Multiple Installation 136
Package Definition File 133, 134, 135, 137
Package Definition File Switches 139
SSL 134
NetPhantom tag attributes
Input Submit 276
Input Text 277
Table 280
Table Data 281
Table Row 281
Header 281
Nextrow 281, 283
Row 281
Using multiple row templates 283
NetPhantom Terminal Window 299
NetPhantom Topologies 79
Internet 80
Intranet/Extranet 80
New Client Connections 210
Nonce timeout 176
Numeric field override 165

O

OEM code page 153
OEM Code Pages 347
OneLogin
Add client manually 332
Add user 334
Change password 333
DLL Object 331
Installation 331
Logon Panel 331
Logon Screen 331
Password change 332
Requirements 331
REXX API 333
What Happens Inside 332
OS/2
Classpath 27

P

Package Definition File 133, 134, 135, 137
Package Definition File Switches 139
passphrase 105
Peer data 67
Personal Certificates 110
Phantom Hurricane
  Copy HTML 290
Port Configuration 173
  Bind address 174
  Port ID 173
  Port number 173
  Queue length 173
  SSL section 174
Port ID 173
Port number 173
Ports
  Default configuration 22
Primary Server 10
print window 172
private key 110
PSB
  Hide unauthorized menu items 47
public key 105, 106, 110
public key infrastructure 121

Q

Queue length 173

R

R2NR
  "Conditional" Comments 354
  Basic Algorithm 352
  declaration file 351
  Declaration File 353
  first method's name 352
  Goto's 354
  header lines file 351
  inheritance string 352
  Translation Rules 352
  translation rules file 352
  Unsupported REXX Features 354
  Variable Declaration 353
R2NR Syntax 351
Registry Keys
  Windows Service 33
Reload TCPIP/LU Mapper 202
Remote Applications 72, 180
Remote Command Line Utilities
  Concurrent User Count 219
  Remote Server Administration 220
Remote Server Administration
  APPS Command 222
  BROADCAST Command 221
  CLIENTS Command 222
  DISABLE Command 222
  ENABLE Command 222
  KILL Command 223
  LOCK Command 222
  MEMORY Command 221
  NOP Command 221
  SHUTDOWN Command 221
  UNLOCK Command 222
Require user authentication 162
Resource cache 175
Resource types 185
Resources 185
  Restricting an application resource 128, 131
RESURL parameter 37
REXX API for Mail 318
REXX API for OneLogin 333
REXX API for SMS 327
REXX Conversion
  NetRexx Converted REXX Source Sample 252
  REXX Source Sample 251
  Troubleshooting REXX Code 251
  Unsupported REXX Features 354
  Unsupported REXX Functions 251
REXX2NetRexx (R2NR) 351
RSA 108
RSA-export 108
Runtime application
  definition 65
  Host session 66
    Custom Host Datastream Exit 68
Runtime Application Data 242
Runtime Panel 242

S

SCRIPT 270
secret key 105
secret-key 105, 106
SecureLogin 101, 183
  Client max age 183
  Code length 183
  Code max age 183
  Configuration 102
    CGI 102
    Resource 102
  Country Code 183
  GSM Phone Network 101
  Pin code 183
  profile definition 183
  SMS Messaging 101
  SMS text 184
Self-Signed Certificate
  Create 117
Send 320
SendMessage 328
SendSMS Commands 327
Server 202
Server Administration Program 149
  Applications 203
  Applications page 161

Broadcast Message 209
Client Connections 208
Client Settings 171
Country Settings 154
Disable remote usage 149
Display Terminal Session 210
Event Filter 155
Event Filtering 215
Event Log 213
Event Messengers 188
File Manager 151
File transfer 150
Font substitution 157
Host settings 163
Keyboard Remapping 211
Managing Groups 189
Managing Users 190
New Client Connections 210
Port Configuration 173
Removing user authentication 24
Server Memory Usage 193
Server Restart 199
Server Shutdown 199
Server Statistics 195
Terminal Window Colors 212
Thread Usage 194
Trace & Event Properties 215, 216
Trace Settings 156, 214
Upgrade Server 201, 202
Web Server 174
Server Architecture
Components 241
Mainframe Host Communication 240
Relationship between Components 241
Schematic 239
Server configuration
Base 153
manual 57
Base section 57
Country settings 61
Event Messengers 73
Font substitution 62
Host section 66
Port section 60
Remote applications 72
Runtime application section 65
Servlet/CGI section 71
SSL settings 73
Trace section 62
Web applications 71
Web server 69
Server Administration Program 149
Server Events 143
Server log output 22
Server Memory Usage 193
Server Restart 199
Hard Java VM Restart 200
Hard Restart 200
Soft Restart 200
Server Shutdown 199
Server states 10
Server States
Active State 10
Changing states 10
Connecting State 10
Non-active State 10
Non-connected State 10
Server Statistics 195
Client Ping 197
Client Statistics 196
Host Statistics 196
Load Balancing 199
Web Server 198
Server types 10
server.ini 57, 59, 70, 71, 146, 203, 214, 215
Using multiple server.ini files 143
server.phm 57, 60, 62, 147, 200, 357
Session Cookies 180, 254, 273
Session Pool Actions 335
Session Pool Scripting
Action tags 337
BREAK 340
CHECK 338
CONDITIONS 339
DISPOSE 338, 344
File format 337
HOSTERROR 342
IF 339
LOG 343
ONERROR 343
PING 338
RECLAIM 338
RESET 341
RETURN 343
SCREEN 338
Script tags 338
SEND 341
SET 340
START 338
TRACE 343
WAIT 342
WHILE 340
Session Pooling 335
Configuration 336
Custom Function Calls 344
Extending the Session Pooling Handler 345
Function Calls in External Classes 344
multiple runtime applications 335
Server Shut-down or Restart 336
Session time-out 179, 272
set_field 317
SetBCC 320, 322, 323
SetCC 321
SetFromEmail 321, 322
SetMsgBody 322
SetServer 323
SetSubject 323
SetToEmail 323

Slave 177
SmartApplet 42, 43
SmartApplet.ini 42
SMS
ApplicationId 329
Initialize 328
key file name 329
Message 329
Phone numbers 329
Port 329
REXX API 327
SendMessage 328
Server 329
Standalone 328
Socket
Start 173
Soft Restart 200
Software updates
SSL 7, 37, 38, 60, 61, 73, 79, 82, 105, 188
ADH 109
CA Certificates 110, 113
Certificates
Create a certificate 111
Cipher suites 73, 107, 113
Selection guidelines 109
Client Authentication 107
Configuration 111
Cookies 273
EDH-DSA/RSA 108
Encrypted Connection 107
External 174
HTML document 82
Identity
Add identity 114
Create 116
Password user ID 114
Identity pouches 110
manual configuration 73
NetPhantom and export restrictions 109
NetPhantom Starter 134
Parameters 114
Disallow TLSv1 Support 116
Maximum Lifetime of Temporary Keys 115
Maximum Use of Temporary Keys 115
ServerSocket Implementor 116
Session Cache Capacity 115
Session Cache Timeout 116
Personal Certificates 110
RSA 108
RSA-export 108
Server Authentication 107
Server configuration 112
TSL 107
with session cookie 180
SSLCertificates
Client 111, 121
Standalone Mail Utility 323, 324
Standalone SMS Utility 328
Start
NetPhantom Client
Parameters 37
User variables 38, 41
NetPhantom Server 22, 26
NetPhantom Server
Batch files 28
Stress Tools
Options 235
StressNoGUI 197
Usage 235
StressNoGUI
Options 235
Usage 236
String Caching 171
Configuration 171
System Requirements 7
Client 8
CPU 7
Disk Space 7
Java Environment 8
Network 8
Operating System 8
Physical RAM 7

T

Tamper-proofing 106
Telnet Host Interface 361
Telnet Playback
Installation 233
Start Command 233
Telnet Trace Files
Data Format 232
Logic Variables 232
Suggested Usage 233
Telnet Tracer 231
Installation 231
Start Command 231
Terminal Window 299
Changing Terminal Session 300
Configuration 301
Manual 302
Properties 299
Terminal Application 303, 304
Requirements 303
Terminal Window Colors 212
TextFileCodePage 147
Third party web servers 76
Thread Usage 194
Timestamping 106
TLS 107
TN3270E 164
Tooltip text 46
dismissDelay 47
initialDelay 47
reshowDelay 47
Trace
Binary Output 145
Verbose Output 145

Trace & Event Properties 215, 216
Trace Settings 156, 214
Tracing 144
Translation Rules 352
Troubleshooting REXX Code 251

U

UNIX
  Classpath 27
Unsupported REXX Functions 251
unused_tag 318
Upgrade Server 201, 202
Use server GUI 154
User authentication 58, 121
  Allow Basic 184
  Basic 85, 86
  Digest 87
    auth-params 87
  Enable 146
  HTTP authentication 84
  Introduction 84
  Password Expiration Interval 190
  Purpose 84
  Realm 184
  Technical
    Authentication Schemes 86
  Usage 85
  User Exit 162, 361
User Definition
  Change/Copy 192
User Exits 359
  Event 359
  LU Mapper 360
  Telnet Host Interface 361
  User Authentication 162, 361
User variables 38, 41
User Window
  RCONSOLE_FILETRANSFER 365

V

Variable Substitution 268
Verbose Trace Output 145
Virtual Components 242
Virtual Message Box 242
Virtual Panel 242
Virtual Session 242
Virtual Session Manager 241

W

Wait stable delay 179, 272
Warn simultaneous requests 175
Web Application 255
  Browser Capabilities 286
  CGI Settings 271
  Configuration Reference 270
  FAQ 298
  Location of Files 259
  Message Box HTML File 288
  Resource Redirection 260
  Resource Settings 273
  Session time-out 272
  Settings 271
  Step-by-Step 258
  Wait stable delay 272
Web Applications
  Defining 178
Web server
  Access Control 184
  built-in features 83
  CGI interfaces 84
  Defining a CGI 178
  DNS 176
  Domains 181
  HTML cache 175
  HTML documents 84
  *HTTP **tunneling*** 38, 81
  IP Address 176
  Load balancing 177
  manual configuration 69
    Remote applications 72
    Servlet/CGI 71
    Web applications 71
  Max simultaneous requests 175
  Redirection service 85
  Remote Apps 180
  Resource cache 175
  Resources 185
    Restricting an application resource 128, 131
  SecureLogin 183
  third party web servers 76
  Warn simultaneous requests 175
  Web Applications 178
Windows
  Classpath 27
Windows Event Log
  Logging to 34
Windows Service
  Event viewer 30
  Generic 31
  License Manager 11
Write event log to screen 156