

## **DENODO ITPILOT 4.5 USER MANUAL**

Update Nov 18<sup>th</sup>, 2009

NOTE

This document is confidential and is the property of denodo technologies (hereinafter denodo).

No part of the document may be copied, photographed, transmitted electronically, stored in a document management system or reproduced by any other means without prior written permission from denodo.

## INDEX

<b>PREFACE</b> .....	<b>1</b>
<b>SCOPE</b> .....	<b>1</b>
<b>WHO SHOULD USE THIS DOCUMENT</b> .....	<b>1</b>
<b>SUMMARY OF CONTENTS</b> .....	<b>1</b>
<b>1 INTRODUCTION</b> .....	<b>1</b>
<b>1.1 DENODO ITPILOT ENVIRONMENTS</b> .....	<b>2</b>
1.1.1 Administration Tool.....	3
1.1.2 Generation Environment .....	3
1.1.3 Execution Environment.....	4
1.1.4 Maintenance Environment.....	4
<b>2 DISTRIBUTION OF ENVIRONMENTS</b> .....	<b>5</b>
<b>2.1 DISTRIBUTION OF THE GENERATION ENVIRONMENT</b> .....	<b>5</b>
<b>2.2 DISTRIBUTION OF THE EXECUTION ENVIRONMENT</b> .....	<b>5</b>
<b>2.3 DISTRIBUTION OF THE MAINTENANCE ENVIRONMENT</b> .....	<b>6</b>
<b>3 INSTALLATION AND INITIAL CONFIGURATION</b> .....	<b>8</b>
<b>4 EXECUTION</b> .....	<b>9</b>
<b>4.1 STARTING UP THE ADMINISTRATION TOOL</b> .....	<b>9</b>
<b>4.2 STARTING UP THE BROWSER POOL</b> .....	<b>9</b>
<b>4.3 STARTING UP THE WRAPPER SERVER</b> .....	<b>9</b>
<b>4.4 STARTING UP THE MAINTENANCE SERVER</b> .....	<b>9</b>
<b>4.5 STARTING UP THE MAINTENANCE GRAPHICAL TOOL</b> .....	<b>10</b>
<b>4.6 STARTING UP THE PDF CONVERSION SERVER</b> .....	<b>10</b>
<b>5 WEB ADMINISTRATION TOOL</b> .....	<b>11</b>
<b>5.1 STARTING UP THE SERVERS</b> .....	<b>11</b>
<b>5.2 CONFIGURING THE BROWSER POOL</b> .....	<b>11</b>
5.2.1 Identification of pool and assignment of ports.....	13
5.2.2 Browser, Download and Cache Parameter Configuration.....	15
5.2.3 Proxy with authentication .....	16
5.2.4 Pool size and policy for reusing browsers .....	17
5.2.5 Initializing the pool.....	19
5.2.6 Firefox Installation Base Path Configuration .....	19
5.2.7 Executing and stopping the Browser Pool .....	20
<b>5.3 CONFIGURATION OF THE WRAPPER SERVER</b> .....	<b>20</b>
5.3.1 Access to Wrapper Server .....	20
5.3.2 List of Wrappers.....	21
5.3.3 Selecting location of the associated browser pool .....	23
5.3.4 Port Assignment.....	23
5.3.5 Configuration of HTTPClient with no Browser Pool.....	23
5.3.6 Password change .....	24
5.3.7 Loading new wrappers from VQL files.....	24
5.3.8 Creating a Web Service .....	24
<b>5.4 CONFIGURING THE MAINTENANCE SERVER</b> .....	<b>26</b>

5.4.1	Access to the Maintenance Server.....	27
5.4.2	Server Configuration Data .....	27
5.4.3	Selecting location for the associated browser pool .....	31
5.4.4	Selecting location of wrapper server.....	31
5.4.5	Capabilities and limitations of the Maintenance Server.....	31
<b>6</b>	<b>MAINTENANCE SERVER GRAPHICAL CONFIGURATION AND MONITORING TOOL .....</b>	<b>33</b>
6.1.1	Configuration of the wrappers to be maintained .....	33
6.1.2	An overview of the Graphical Tool .....	35
6.1.3	Configuration of the Maintenance Environment .....	36
6.1.4	Monitoring the Maintainable Wrappers.....	37
<b>7</b>	<b>ANNEX A: DEPRECATED FEATURES .....</b>	<b>40</b>
<b>7.1</b>	<b>ACTIVEX CONTROL FOR AUTOMATIC BROWSING SEQUENCE RUNNING IN CLIENT BROWSERS .....</b>	<b>40</b>
	<b>REFERENCES .....</b>	<b>41</b>

**FIGURES**

<b>Figure 1</b>	Bookshop form .....	2
<b>Figure 2</b>	ITPilot Environments and Components .....	3
<b>Figure 3</b>	Distribution of the Generation Environment .....	5
<b>Figure 4</b>	Distribution of the Execution Environment .....	6
<b>Figure 5</b>	Relationship Between Execution and Maintenance Environments .....	7
<b>Figure 6</b>	Login page of the Administration tool .....	11
<b>Figure 7</b>	Browser pool Tab .....	12
<b>Figure 8</b>	Server Addition Page .....	13
<b>Figure 9</b>	Identification and Assignment .....	14
<b>Figure 10</b>	Browser behaviour .....	16
<b>Figure 11</b>	Proxy with Authentication .....	16
<b>Figure 12</b>	Size and Reuse Policy .....	17
<b>Figure 13</b>	Pool Initialization .....	19
<b>Figure 14</b>	Wrapper Server Configuration Window .....	20
<b>Figure 15</b>	Wrapper server Connection .....	21
<b>Figure 16</b>	Wrapper Execution Page .....	22
<b>Figure 17</b>	Pool Browser Localization .....	23
<b>Figure 18</b>	Configuration parameters of the http client in the Wrapper Server .....	24
<b>Figure 19</b>	Loading Wrappers Using VQL Files .....	25
<b>Figure 20</b>	List of Wrappers with Loaded "Webmail" .....	25
<b>Figure 21</b>	Web Service Export Page .....	26
<b>Figure 22</b>	Maintenance Administration Main Page .....	27
<b>Figure 23</b>	Maintenance database Parameters .....	28
<b>Figure 24</b>	Wrapper Change Notification Parameters .....	29
<b>Figure 25</b>	Port Assignment Parameters .....	29
<b>Figure 26</b>	Edition of Verification Rules .....	31
<b>Figure 27</b>	Locating the browser pool .....	31
<b>Figure 28</b>	Locating the Wrapper Server .....	31
<b>Figure 29</b>	Maintenance Graphical Tool Loading Window .....	33
<b>Figure 30</b>	Maintenance Graphical Tool .....	34
<b>Figure 31</b>	Main elements of the Maintenance Graphical Tool .....	35
<b>Figure 32</b>	Rule Configuration in the Maintenance Graphic Tool .....	36
<b>Figure 33</b>	Tests Tab .....	37
<b>Figure 34</b>	Queries Tab .....	38
<b>Figure 35</b>	Database Tab .....	38
<b>Figure 36</b>	Information about the selected wrapper .....	39
<b>Figure 37</b>	Graphs Tab .....	39

## PREFACE

### SCOPE

This document serves as introduction, administration and user guide of Denodo ITPilot.

### WHO SHOULD USE THIS DOCUMENT

This document is aimed at administrators that want to install the software, and to use the Denodo ITPilot administration tool.

### SUMMARY OF CONTENTS

More specifically, this document describes:

- An introduction to ITPilot
- The different functioning environments of ITPilot
- The configuration of each of the Denodo ITPilot components in the execution and maintenance environments

## 1 INTRODUCTION

Most data available on the World Wide Web (hereinafter Web) can be obtained only by means that are friendly for Web users, but not useful for automatic and mechanical processing by software applications. Nowadays, many Web sites offer *ad hoc* query interfaces with forms that return the data required in lists comprising semi-structured responses encoded in HTML documents. This part of the Web – accessed through different types of forms and/or interfaces that return data automatically obtained from internal databases – is normally called “Hidden Web”).

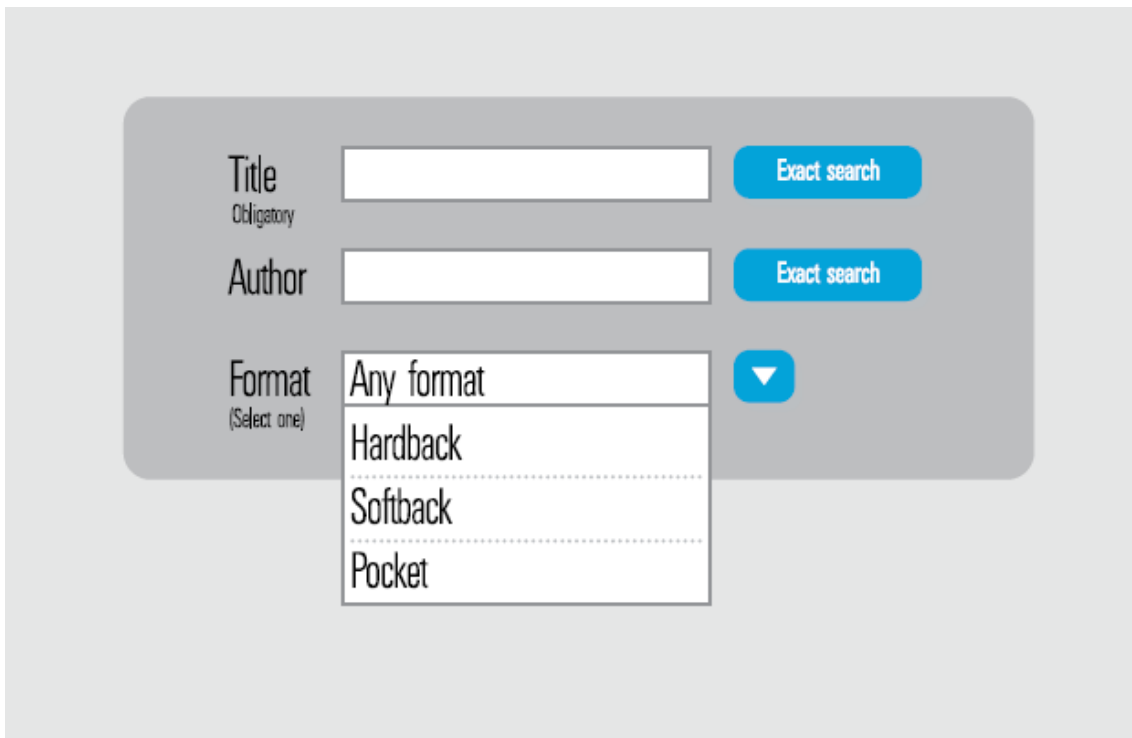
This “Hidden Web” is by no means a small part of the whole WWW and contains a huge amount of data which, in many cases, are of great quality and interest to users. Web sites like e-shops (that provide their catalogs in this way) and search engines for data of a scientific, health, patenting or financial nature are good examples of this. It is also often the case that these Web sites are private access (i.e. a user/password is required to access them), have an advanced query interface (allowing data searches in respect of different subject matters) and/or return results in the form of lists of items encoded in HTML with links to related pages that contain more data on each item (e.g. generally e-shops return a list of results, but with the option for the user to “click” on the title to access another page with commentaries on the product, photos, related products, etc.).

Other common complications arise from the use of technologies such as JavaScript, dynamic HTML or session maintenance systems that further complicate automated access to data contained in these Web sites.

In addition to the problem of accessing these sources with “hidden” data, applications that want to use these data are also frequently faced with the problem of results being returned in HTML, which is a tag language defined for visual display by users that never publishes metadata of any type on the structure and/or semantics of the results generated. Neither does it structurally differentiate navigation elements (menus), graphic panels and data useful to the user. The problem of extracting the relevant data contained in HTML pages thus also arises.

Example: Look at this example of an Internet bookshop with a search form as shown in Figure 1. The form obliges users to specify a value for the attribute ‘TITLE’ and gives them the option of entering a value for the attribute ‘AUTHOR’ and for the attribute ‘FORMAT’ (restricting a group of values).

The bookshop returns a result list with data on TITLE, AUTHOR, FORMAT, PUBLISHER and PRICE.



The image shows a search form for a bookshop. It consists of three rows of input fields and buttons. The first row has a text input field labeled 'Title' with the subtext 'Obligatory' and a blue button labeled 'Exact search'. The second row has a text input field labeled 'Author' and a blue button labeled 'Exact search'. The third row has a dropdown menu labeled 'Format' with the subtext '(Select one)'. The dropdown menu is open, showing three options: 'Any format', 'Hardback', and 'Pocket'. To the right of the dropdown menu is a blue button with a downward-pointing triangle.

**Figure 1** Bookshop form

This case summarizes the difficulties an application faces when attempting to extract structured data from Web environments: accessing Web sources, navigating through transactional environments, option selecting and, finally, extracting data from semi-structured data.

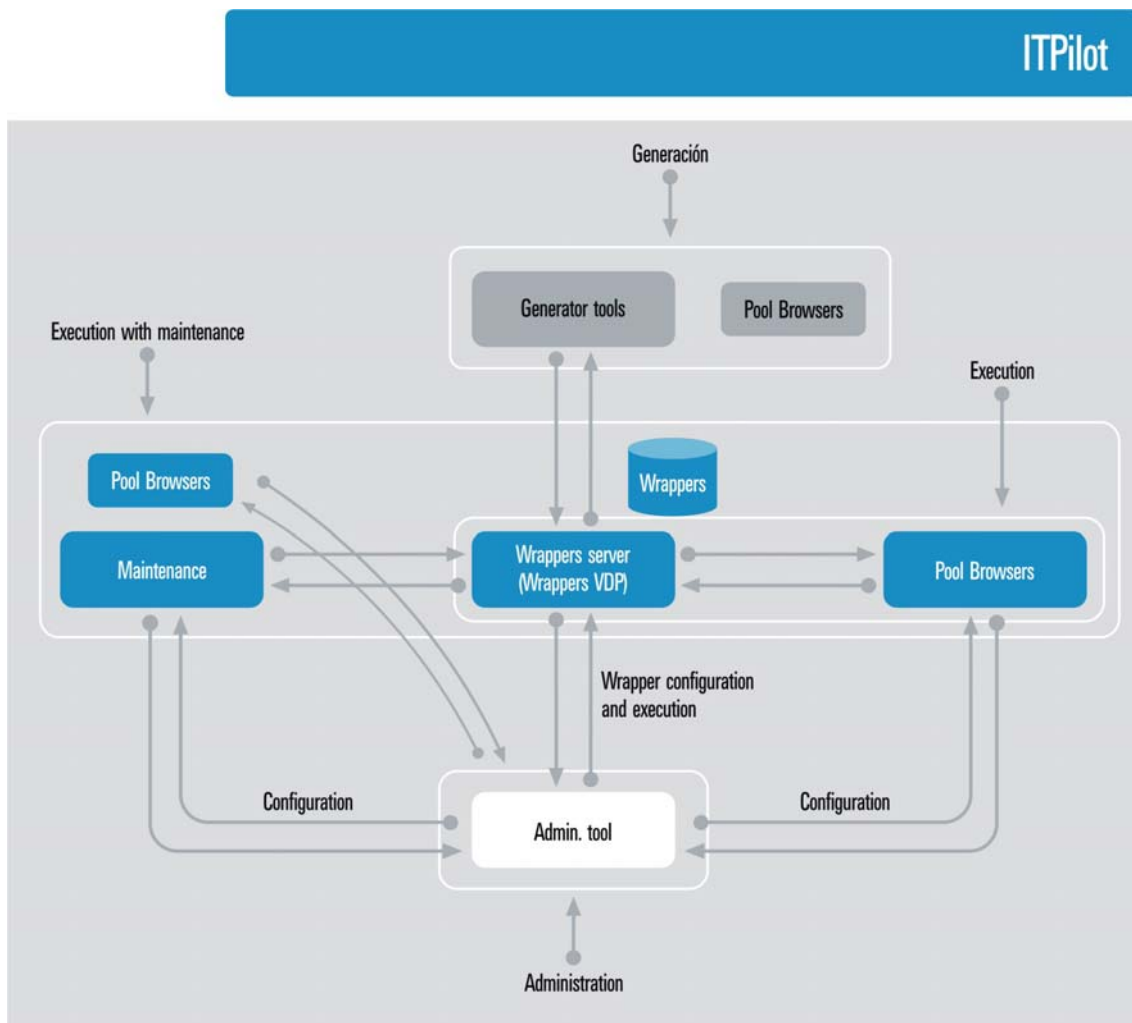
Denodo ITPilot is the Denodo Technologies solution for easy access to and structuring of datasets on the Web; this process involves constructing an abstraction from the specific Web source called wrapper that isolates the client applications from the intrinsic characteristics of this site (access protocol, native data structure, etc.). ITPilot provides a distributed and scalable environment for generating, executing and maintaining wrappers.

This manual presents Denodo ITPilot and provides instructions for correct installation, recommendations on the different types of architecture it supports, as well as a guide to the execution and maintenance environment. The components of ITPilot are introduced in this same section; next section will provide an overview of the recommended architectures. Chapter 3 gives a detailed description of the installation process for each of the components, while Chapter 4 does the same job with the server startup procedure. Chapter 5 explains the ITPilot Web Administration Tool and how to export a wrapper as a Web Service. Finally, Chapter 6 deals with the ITPilot Maintenance Server Graphical Configuration and Monitoring Tool.

## 1.1 DENODO ITPILLOT ENVIRONMENTS

Denodo ITPilot facilitates wrapper generation, execution and maintenance in Web sources in a simple and dynamic way. Three Environments exist, each of which facilitates one of the aforementioned actions and all are managed through the Administration Tool. Each environment contains a series of Components described below. Figure 2 shows the relationships between Environments and their Components.





**Figure 2** ITPilot Environments and Components

### 1.1.1 Administration Tool

The execution environment configuration is managed via the ITPilot Administration tool. This is a Web application that can be deployed in Web containers and that meets servlet and JSP specifications, and communicates with the ITPilot servers (wrapper server, browser pool and maintenance server) to configure their execution settings.

### 1.1.2 Generation Environment

This environment includes the group of components necessary for creating wrappers from DEXTL data extraction specifications generator (see [DEXTL], [GENER]) and NSEQL navigation sequences (see [NSEQL], [GENER]). The components it uses are as follows:

- Generation Tools: tools for generating data extraction specifications and navigation sequences are graphical applications that allow a non-technical user to create Web wrappers. For more information we recommend reading the Denodo ITPilot Generation Environment Manual [GENER].
- Generation Browser Pool: this environment uses a browser pool internally to check the navigation sequences and final specification.
- PDF Conversion Server: this environment makes use of an embedded PDF conversion Server, which transforms PDF documents to HTML, based on [PDFBOX], and is automatically started. This server will be

used by the Generation Tool to perform the required conversions. The execution time conversions are performed by the Execution Environment's PDF conversion server, which is a separate component.

In addition and although it does not belong to this environment *per se*, generator tools may need to store the wrapper created. The Wrapper Server in the Execution Environment is used to do this (see next section 1.1.3).

### 1.1.3 Execution Environment

This is the continued operation environment, in which the user can use previously created wrappers to launch queries on isolated sources. This use may be direct (through an API or publishing the wrapper as a Web Service) or through other products such as Denodo Virtual DataPort, with which Denodo ITPilot is fully integrated. The components that make up this environment are as follows:

- Wrapper Server: this is the component responsible for storing wrappers for accessing. These include a remote interface for statement execution.
- Browser Pool: when a wrapper is executed, a browser type can be selected: IEBrowser (automatic navigation module based on Microsoft Internet Explorer [IE]), Firefox [FRFOX] or a Denodo browser based on HTTP as an access method. In this case, the wrapper server uses the browser pool to minimize the time required to create browser instances. This pool can be configured from the administration tool.
- PDF Conversion Server: this is the component responsible for transforming PDF documents to HTML, so their content can be extracted by Denodo ITPilot.

### 1.1.4 Maintenance Environment

The Maintenance Environment adds functionality on top of the Execution Environment, and complements it by making the deployed wrappers more robust and reducing the manual maintenance effort. As Web sources are autonomous and independent of the wrappers, they can change over time, and these changes can invalidate the current access mode, whereby the wrappers no longer extract the data properly. Denodo ITPilot offers an automated maintenance tool that allows wrappers to be repaired automatically by detecting the changes referred to above. Although this will be dealt with in more depth in section 5.4, its basic functioning is as follows:

- The wrapper server stores all the wrappers in each of the Web sources.
- The system uses the "check frequency" configuration parameter to check each wrapper for changes.
- When a change is detected in a source, the actions to be taken can be configured.
  - o One possible action is to send an e-mail informing of the change.
  - o The other option is to regenerate and edit the wrapper. Actions can be strung together and can be implemented by users.

The components of this environment, apart from those already mentioned in the execution environment, are as follows:

- Maintenance Server: component responsible for detecting automatically any change happened in the sources and for regenerating the wrappers. It communicates with the wrapper server to request all the wrappers to maintain, and to obtain the query execution results over them (which will be used to check possible changes, and during the regeneration process).
- Browser Pool of the Maintenance Server: browser pool used in the regeneration phase.
- PDF Conversion Server: this is the component responsible for transforming PDF documents to HTML, so their content can be extracted by ITPilot.

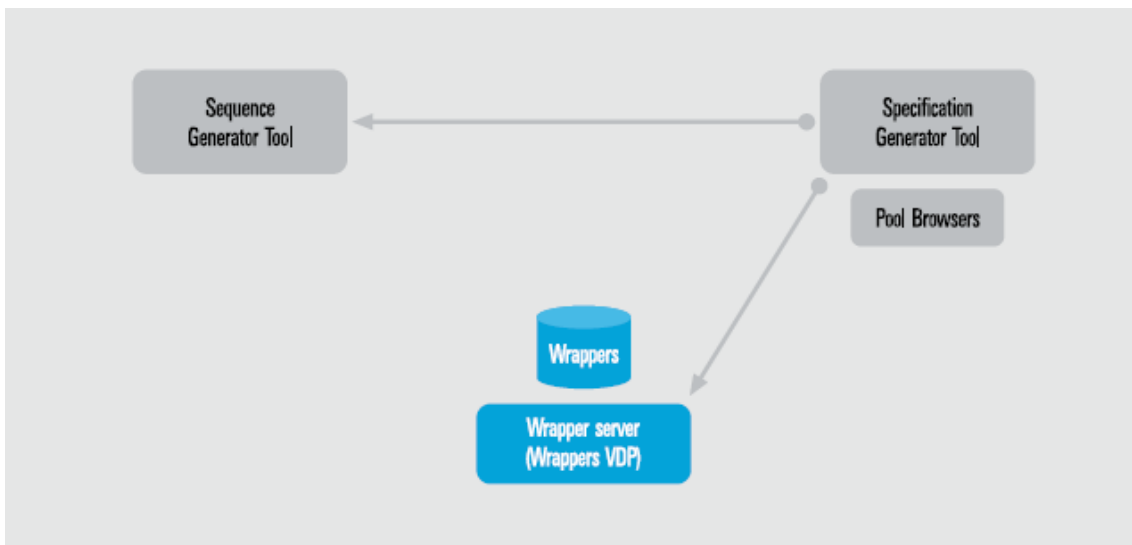
As mentioned earlier, a detailed explanation of this environment is provided in section 2.3 of this same manual.

The next section recommends different distribution architectures for these components. Chapter 3 gives details of the installation and configuration processes for each of the ITPilot environments.

## 2 DISTRIBUTION OF ENVIRONMENTS

### 2.1 DISTRIBUTION OF THE GENERATION ENVIRONMENT

As mentioned in the preceding section, the Generation Environment allows wrappers to be created in a visual and simple way. This environment requires the installation of two components: the specifications generator tool and the navigation sequences generator tool. The wrapper server of the execution environment may also be accessible (this is optional: users also have the option of storing the wrapper in a local file that can be manually added to the wrapper server). Figure 3 shows the relationship between the elements.



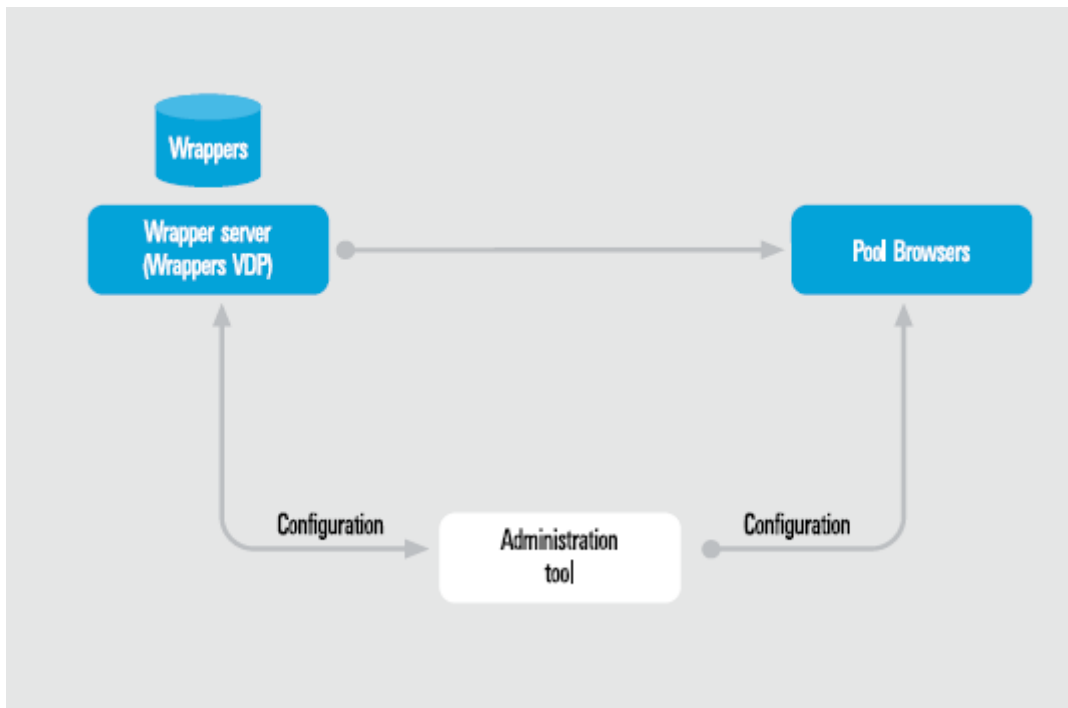
**Figure 3** Distribution of the Generation Environment

The wrapper server belongs to the execution environment, whereby it is normally installed in a separate machine in the production environment.

This manual does not aim to explain how to install, operate and handle the tools in this environment. For more information please refer to [GENER] for instructions on installation and operation and [DEXTL] and [NSEQL] for detailed information on specification and sequence definition languages.

### 2.2 DISTRIBUTION OF THE EXECUTION ENVIRONMENT

Denodo ITPilot operates in the execution environment, where actions are executed on wrappers that encapsulate the Web sources from which data are to be extracted. Three components are required in this case: the Web administration tool (independent of the environment, but used here), the wrapper server and the browser pool. Figure 4 describes the relationship between these elements.



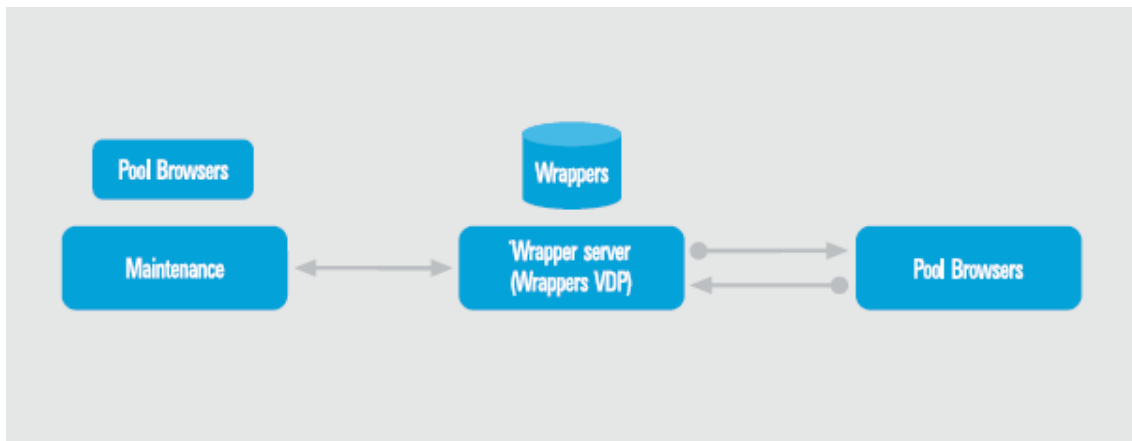
**Figure 4** Distribution of the Execution Environment

As the wrapper server can be used in different environments and due to its possible workload, it is recommended that it be installed in a machine that is independent of the rest of the system. The browser pool can be found either in the same machine as the wrapper server or in a separate machine; in general, this depends on the maximum number of browsers that can be open during system execution.

### 2.3 DISTRIBUTION OF THE MAINTENANCE ENVIRONMENT

This environment should be executed together with the execution environment and allows ITPilot to monitor changes in sources from which data are extracted and automatically regenerate wrappers as required (see section 1.1.4). The maintenance server, which uses a browser pool, can be executed in the same machine as the wrapper server, although it is a distributed component, whereby we recommend that it be installed in another machine.

Figure 5 shows the relationship between this environment and the execution environment.



**Figure 5** Relationship Between Execution and Maintenance Environments

The basic process of the maintenance server is the following: when executing a query against a wrapper, it is sent along with the produced results to the maintenance module. When this module receives the query and associated results, they will be stored in a relational database and, at the same time, the necessary tests will be executed in order to determine whether that wrapper has changed or not.

Each test (configurable by the user, see section 5.4) is executed by handing that query plus its results as parameters. Each test returns a result between 0 and 100 (where 0 means that the condition is not accomplished at all and 100 that is absolutely successful) which is stored in a result manager.

Next, an evaluating process is launched which determines if the wrapper has changed in terms of the results of the tests. This evaluator needs both the results from the last tests, and the evaluation rules.

If the wrapper changes, the maintenance system selects the subset of all these stored queries which will be used by regenerating the wrapper.

When the query results are saved in the database, an expiry time is assigned to each of them. The expired results are deleted on a period basis.

The next section describes the installation steps for each of the components.

### **3 INSTALLATION AND INITIAL CONFIGURATION**

The *Denodo Platform Installation Guide* [DENINST] provides all information required to install Denodo ITPilot, including hardware and software minimum requirements, and instructions to use the installation tool and for the initial configuration of the system.

## 4 EXECUTION

Once the installation process has terminated, the servers are ready to run. Each server found on the same machine as the administration tool can be started up directly from the Web tool itself, as dealt with in section 5.1. If this is not the case, they have to be started up in the machines in which they reside.

### 4.1 STARTING UP THE ADMINISTRATION TOOL

The administration tool can be started from the Denodo Platform Control Center (see the Denodo Platform Installation Guide [DENINST]), or by using the following scripts available in the `DENODO_HOME/bin`:

- `itpilot_webadmin_startup`: starts up the administration tool.
- `itpilot_webadmin_shutdown`: stops the administration tool.

Once the application has been properly displayed, the administration tool will be available in `http://domain:9090/webadmin/denodo-itpilot-admin`

### 4.2 STARTING UP THE BROWSER POOL

The browser pool can be started from the Denodo Platform Control Center (see the Denodo Platform Installation Guide [DENINST]), or by using the following scripts available in the path `DENODO_HOME/bin`:

- `browserpool_startup`: starts up the browser pool.
- `browserpool_shutdown`: stops the remote pool and all the browsers contained in it.

### 4.3 STARTING UP THE WRAPPER SERVER

The execution server can be started from the Denodo Platform Control Center (see the Denodo Platform Installation Guide [DENINST]), or by using the following scripts available in the path `DENODO_HOME/bin`:

- `vqlserver_startup`: starts up the Wrapper Server.
- `vqlserver_shutdown`: stops the Wrapper Server.
- `vqlserver`: with the options `startup` and `shutdown`.

### 4.4 STARTING UP THE MAINTENANCE SERVER

The maintenance server can be started from the Denodo Platform Control Center (see the Denodo Platform Installation Guide [DENINST]), or by using the following scripts available in the path `DENODO_HOME/bin`:

- `maintenance_startup`: starts up the Maintenance Server.

- `maintenance_shutdown`: stops the Maintenance Server.

#### 4.5 STARTING UP THE MAINTENANCE GRAPHICAL TOOL

The maintenance graphical tool, described in detail in section 6, can be started by using the `maintenanceGUI_startup` scripts available in the path `<DENODO_HOME>/bin`.

#### 4.6 STARTING UP THE PDF CONVERSION SERVER

The PDF conversion server can be started from the Denodo Platform Control Center (see the Denodo Platform Installation Guide [DENINST]), or by using the script `PdfConversionsServer.exe` which resides in the `DENODO_HOME/bin` directory, and that allows the server to be started up and stopped. The server follows this format:

```
PDFConversionsServer ([-start | -shutdown] [-conf='confFile'])
```

, where `-start` means that the conversion server must be started, `-stop` means that it must be stopped, and `confFile`, as the server configuration file. By default, it can be found at `DENODO_HOME/conf/iebrowser`, with the name `IEBrowserConfiguration.properties`.



## 5 WEB ADMINISTRATION TOOL

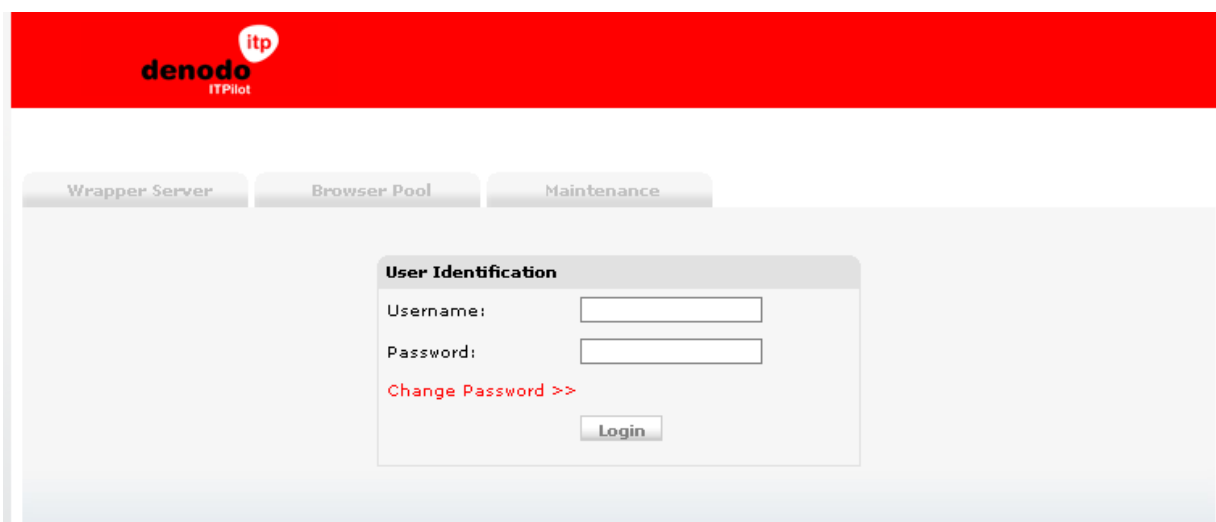
The ITPilot administration tool allows the execution and maintenance environments to be managed in a simple and uniform way. This is a Web application that controls both the wrapper server and the browser pool as well as the maintenance server – in case of having maintenance enabled. Figure 6 displays the aspect of the tool after startup and access (through the URL domain:port/webadmin/denodo-itspilot-admin/ and with user admin and empty password as initial access).

The tool is visually composed of the elements described in the preceding figure.

- Server Selection Area: it is here that the user can select which server is to be configured: wrapper server, browser pool or maintenance server, where it is used for administration of the execution environment.
- Work Area: this area displays the configuration data relevant for each server.


The Web administration tool can be used to configure and, in specific cases, start up and stop the different servers that make up the execution and maintenance environments.

The following section describes the series of steps to be taken to configure and administer the execution and maintenance environments.



**Figure 6** Login page of the Administration tool

### 5.1 STARTING UP THE SERVERS

The administration tool can be used to manage the configuration of each of the servers. The servers must be started up and running before they can be configured via the web administration tool. If they are in the same machine as the administration tool, they may be started up from it directly by using the button Start/Stop . If the servers are found distributed across machines other than where the administration tool resides, these must be started up beforehand.

### 5.2 CONFIGURING THE BROWSER POOL

The wrappers that implement the navigation sequences through NSEQL programs require that the ITPilot execution

environment has access to a browser pool. The configuration options for his component are described in this section.

The pool browsers can belong to three different types:

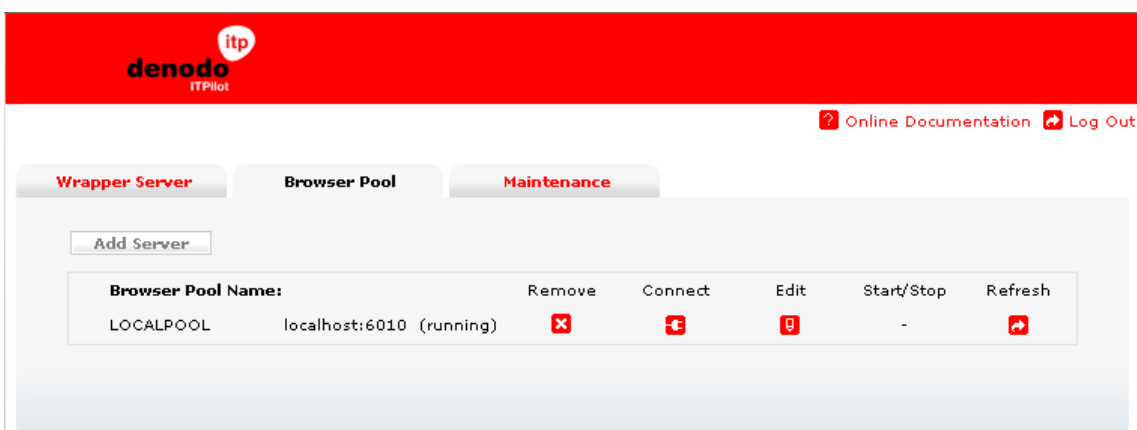
- IEBrowser: browsers based on Microsoft Internet Explorer.
- Firefox: browsers based on Mozilla Firefox.
- HTTPBrowser: browsers based on Denodo HTTPBrowser, a GUI-less, HTTP-based browser with Javascript support.

One pool may contain browsers that belong to one or more of these types. The browser type to use for each browsing sequence is defined at wrapper generation time: it can be set to a specific browser type or it can be set to use whatever type of browser is configured in the browser pool (please see the Denodo ITPilot Generation Environment Guide [GENER] for more information).


A first aspect to bear in mind is that the IEBrowser- and Firefox -type browsers in the pool will use the configuration established for *Microsoft Internet Explorer* and/or *Firefox* in the system in which the pool is executed.

It is also necessary to consider the security options and cookies, as the pool browsers will act according to said configuration.

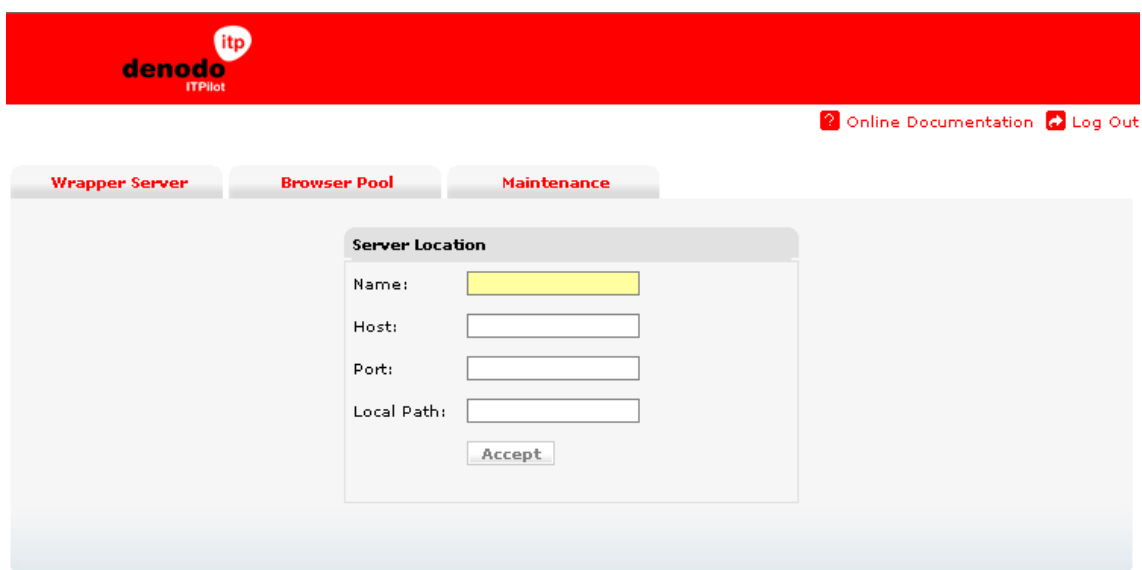
The browser pool is configured in the "Browser Pool" panel of the administration tool in the ITPilot execution environment. Figure 7 shows this window.



**Figure 7** Browser pool Tab

In the first place, the connection settings of each browser pool to be managed must be indicated. The "AddServer"  button is used for this, which displays a window like that shown in Figure 8. The fields to be completed are as follows:

- Name: server identifier name.
- Host: address where it is found.
- Port: server listening port.
- Local path: optional, to indicate that the server is local; by adding the local path, where the application is, the user will be able to start up and stop the browser pool from the graphical administration tool.



**Figure 8** Server Addition Page

The connection settings of an already added server can be edited by pressing the “Edit” button that leads to the same configuration window mentioned earlier. The “Start/Stop” button will be visible, if - and only if - the “Local Path” field has been properly completed when configuring the pool.

Of course, any number of pools can be added as needed, although the architecture considerations in section 2 of this document should be taken into account.

Once the pool has been configured, connect it by clicking on the “Connect” button. If the connection is successful, the parameter set that can be configured by the user will appear in the window below the server details.

The existing configuration parameters can be divided into several groups, each of which can be accessed in the administration tool panel: pool global parameters, IEBrowser configuration parameters, Firefox configuration parameters and Denodo HTTPBrowser configuration parameters.

The global parameters include parameter groups for identification of the pool and system port assignment, behavior of the browser pool and HTML conversion configuration.

The parameters for each of the three browser types include the following groups of parameters: browser configuration, support for proxies with authentication, pool size and browser assignment policies and, finally, initialization parameters.

At last, the Firefox browsers include an additional parameter group to set the Firefox installation to be used.

The following subsections deal with each of these parameter groups respectively. The text indicates whenever the available options vary according to the browser type.

### **5.2.1 Identification of pool and assignment of ports**

The parameters of this group are:

- **TYPE OF BROWSER:** browser type to be used in the pool by default (it will be used by the wrappers configured to use the default browser type)

- IEBrowser: Internet Explorer browser.
- Firefox: Firefox browser.
- HTTP: Denodo HTTPBrowser browser
- PORT. Port in which the browser pool listens to requests.
- INITIAL\_PORT. Each browser of the pool listens to requests in a port. The value of this parameter determines the port number to be used as the first one to assign port numbers to the browsers. From this number consecutive port numbers will be used in an ascending order.
- SHUTDOWN\_PORT: port in which the server listens the Shutdown signal in order to be stopped.
- AUXILIARY\_PORT: The auxiliary port is used by the pool for communications with its clients.

Figure 9 shows the administration tool page where to configure those parameters.



Port Identification and Port Assignment			
Type of Browsers	IEBrowser		
PORT	6001	SHUTDOWN PORT	6002
INITIAL PORT	6100	AUXILIARY PORT	6003

**Figure 9** Identification and Assignment

#### 5.2.1.1 Comparison between the browser pool and the http client

The ITPilot http client (Denodo HTTPBrowser) embeds a JavaScript engine, which allows it to perform complex navigations in several web sources. When deciding what type of browsing tool to use (a browser pool or an http client) to extract information from a web source, the following factors must be taken into account:

1. Efficiency. The http client is more efficient than the IE/Firefox browsers, since it is lighter. This implies an increase in the response time when accessing sources, and a decrease in the CPU load of the machine which houses the browser pool; this feature is very important when several parallel executions are required.
2. The http client can not execute some of the NSEQL navigation commands (see [NSEQL]).
3. The http client does not interpret code written in VBScript.
4. In some pages, ITPilot's JavaScript engine may process JavaScript code in a different way than Internet Explorer or Firefox do. This is because these browsers' interpreters might be laxer regarding the syntax used by the web pages. In these cases, the desired behavior will be that of Internet Explorer or Firefox, since very probably, the target pages have been designed to work correctly with those browsers.

### 5.2.1.2 HTML conversion configuration

This section shows how to configure the conversion tools from Microsoft Word/Excel and PDF to HTML so that the content of those resources can be extracted by ITPilot:

- PDF To HTML converter: conversion tool type used to transform the PDF resource into HTML
  - Acrobat HTML: uses the HTML conversion tool from the Adobe Acrobat Professional software (it is required that this product be installed).
  - Acrobat Text: uses the plain text conversion tool from the Adobe Acrobat Professional software, from which ITPilot generates an HTML file (it is required that this product is installed).
  - PDF Box: uses the PDFBox library [PDFBOX] (bundled with the Denodo Platform) to generate the HTML page.
- Conversion Server port: port where the conversion Server will be listening from.
- Acrobat Prof. Plugins Directory: path where the Acrobat Professional plugins reside.
  - In this case, besides updating the directory in the administration tool, the plugin `DDEPdfToHtml.api`, which resides in the `<DENODO_HOME>/dll/itpilot` directory, must be copied to the `Acrobat/plugin_ins` directory, wherever Adobe Acrobat is installed.
- Open Office Lib Directory: directory where the Open Office class library can be found. Open Office is used for the conversions from Microsoft Word and Microsoft Excel to HTML.

## 5.2.2 Browser, Download and Cache Parameter Configuration

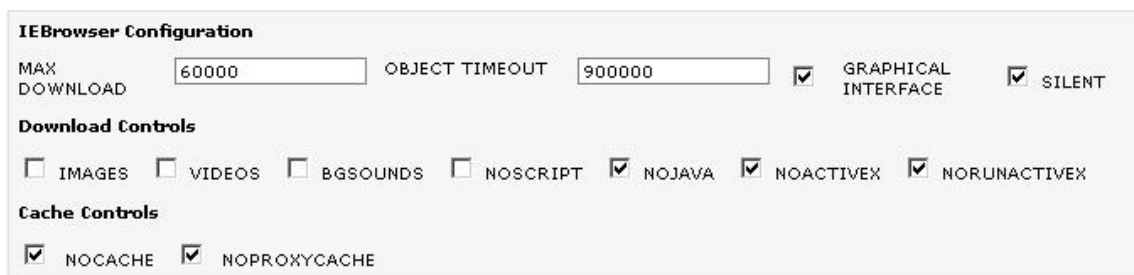
The parameters of this group can be configured for each of the three browser types. The parameters are:

- `MAX_DOWNLOAD`: Indicates the maximum time a browser will wait to download a page (in milliseconds).
- `OBJECT_TIMEOUT`: Maximum time (in milliseconds) that a browser can be used outside the pool to deal with a wrapper request. When this time lapses, the browser is destroyed. If the value of this parameter is less than 0, the browser can remain outside the pool indefinitely.
- `DOWNLOAD_CONTROLS`. This group of parameters allows the type of contents that should be downloaded by the pool browsers to be specified. The content types whose download can be configured are: images, videos, background sounds, script programs, Java applets and ActiveX components (for download and execution purposes). If Firefox is used as browser, only the following parameters can be configured: images, javascript, java, cache and proxy. If Denodo HTTPBrowser is used, only the following parameters can be configured: javascript, java, cache and activeX.
- `CACHE_CONTROLS`. This group of parameters is for specifying, whether or not the pool browsers should

use the local cache and/or the proxy cache. Firefox and HTTPBrowser do not allow the proxy cache option to be configured. HTTPBrowser can configure an additional parameter for the maximum number of JavaScript files that can store in its JavaScript cache ("Maximum number of cached JavaScript files"): cached scripts will be executed faster than the ones that have to be downloaded again.

- **GRAPHICAL\_INTERFACE.** Indicates whether or not the pool browsers will display a graphical interface. To optimize system efficiency applications in production do not normally display browser graphical interfaces. However, it may be useful to turn the interface on for debugging purposes. This parameter can only be configured when Internet Explorer is being used as browser.
- **SILENT:** with this option activated, none of the dialogs that the browser normally opens is displayed (this includes JavaScript dialogs, certificate warnings, etc.).

Figure 10 shows the administration tool page where to configure these parameters.



**IEBrowser Configuration**

MAX DOWNLOAD: 60000    OBJECT TIMEOUT: 900000     GRAPHICAL INTERFACE     SILENT

**Download Controls**

IMAGES     VIDEOS     BGSOUNDS     NOSCRIPT     NOJAVA     NOACTIVE     NORUNACTIVE

**Cache Controls**

NOCACHE     NOPROXYCACHE

**Figure 10** Browser behaviour

### 5.2.3 Proxy with authentication

If the Internet is accessed through a proxy with authentication, the following parameters (for all three browser types) must be configured:

- **PROXY\_LOGIN:** user login in the proxy.
- **PROXY\_PASSWORD:** user password in the proxy.
- **PROXY\_DOMAIN:** (Windows 2000): Windows domain.

Figure 11 shows the administration tool page where to configure these parameters.



**Proxy with Authentication**

PROXY LOGIN:     PROXY DOMAIN:     PROXY PASSWORD:

**Figure 11** Proxy with Authentication

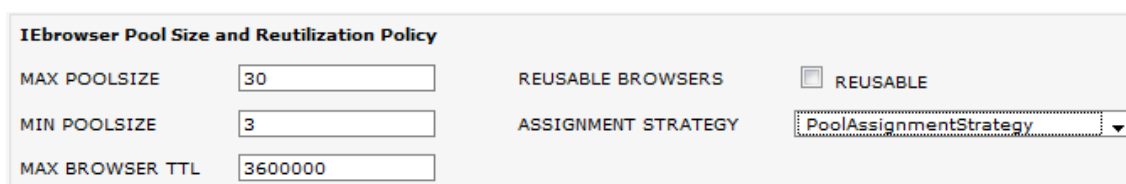
**NOTE:** the proxy server parameters must be correctly defined in IE/Firefox if this option is used. If the browser is not correctly configured to browse through the proxy server, the ITPilot server will ignore these parameters.

#### 5.2.4 Pool size and policy for reusing browsers

The parameters of this group are available for all three browser types; these are:

- **MAX\_POOLSIZE:** Maximum number of browsers in the pool.
- **MIN\_POOLSIZE:** Minimum number of browsers. The system will not reuse browsers already existing in the pool unless the current number is equal to or greater than the value of this parameter.
- **REUSABLE\_BROWSERS.** Indicates if the pool browsers can be reused to deal with more than one request. Enabling browser reusability increases the efficiency of most applications; however, it may not be suitable in some cases, where dealing with a previous request changes the browser response to subsequent requests (for example, through the use of cookies).
- **ASSIGNMENT\_STRATEGY.** Allows to select the assignment strategy to be used by the browser pool. The PoolAssignmentStrategy strategy attempts to assign a browser to each request the status of which allows the number of navigation steps required to deal with a request to be minimized. Otherwise, the SimplePoolAssignmentStrategy strategy assigns any free browser to each request. If reuse is deactivated (REUSABLE\_BROWSERS=false), then the ASSIGNMENT\_STRATEGY value is ignored. The next section (5.2.4.1) explains the implications of this parameter in more detail.
- **MAX BROWSER TTL.** Maximum Time to Live of a browser. If a browser is active more than the specified time, it will be removed and a new one will be created with the same page loaded as the former browser. This is useful because, due to known problems in some versions of Microsoft Internet Explorer, when using this type of browser, performance may degrade if the browser has been open for too long.

Figure 12 shows the administration tool page where to configure these parameters.



IEbrowser Pool Size and Reutilization Policy			
MAX POOLSIZE	<input type="text" value="30"/>	REUSABLE BROWSERS	<input checked="" type="checkbox"/> REUSABLE
MIN POOLSIZE	<input type="text" value="3"/>	ASSIGNMENT STRATEGY	<input type="text" value="PoolAssignmentStrategy"/>
MAX BROWSER TTL	<input type="text" value="3600000"/>		

**Figure 12** Size and Reuse Policy

##### 5.2.4.1 Browser Reuse Policies

It often occurs that navigation sequences executed by a specific wrapper share a series of initial common steps; for example, imagine that a wrapper has been created to automate the search process in a specific Web source. The source requires an authentication process that involves the introduction of a user name and a password. In our example, let us imagine that the wrapper uses the same login/password pair for all source accesses.

Using Denodo ITPilot to create this wrapper (for more information see [GENER]) an initial navigation sequence would

be created that would execute the following steps:

1. Connect to the source home page.
2. Complete the authentication form with the login/password and press the "Submit" or "Enter" button to authenticate.
3. Once authenticated, click on the link that accesses the search page.
4. Complete the search form with the required query.
5. The server returns a page with the query results.

The first three steps are common to all queries made to the wrapper. The difference between one query and the next only arises in step four, when the search form is completed according to the specific query to be made at each moment in time.

It would be nice to save time on these first three steps in each query: ideally, when a new query is received, one browser is already authenticated and situated in the search page of the source to which the new request could be assigned. The browser searches immediately (step 4) and returns the results (step 5), thus avoiding wasting time in steps 1-3.

Denodo ITPilot supports this "intelligent" reuse of browsers through the combined use of the following mechanisms:

- "Back" navigation sequences. A *back navigation sequence* is responsible for returning a browser to a state in which it can be reused in future requests to the same wrapper. When the wrapper in our example has made a query to the source, the browser used to execute the navigation sequence stays in the query results page (step 5). For the browser to be used for a new wrapper query it must return to the search page (step 4). The sequence responsible for achieving this is the aforementioned back sequence. A wrapper can obtain a back sequence in two ways:
  - Explicitly: the wrapper creator can specify a back navigation sequence for a wrapper in the Search tab in the "Back Sequence" option in the sequence loading area of the specifications generator (see [GENER]).
  - Implicitly: if the STATE assignment strategy has been activated in the browser pool (ASSIGNMENT\_STRATEGY= PoolAssignmentStrategy, see next point) and a wrapper does not have an explicitly defined back sequence, then Denodo ITPilot will attempt to obtain a suitable back sequence for the wrapper depending on its previous executions. Normally Denodo ITPilot requires at least two wrapper executions before being able to determine, whether a back sequence that is appropriate to the wrapper exists.
- Browser pool assignment strategy PoolAssignmentStrategy. If this browser assignment strategy is activated, then, when the pool receives a request to execute a specific navigation sequence, it then searches amongst the active browsers to see, if any is free that is already in one of the intermediate pages of the sequence, thus avoiding having to repeat it in its entirety. Continuing with our example, if the pool receives a request to execute a navigation sequence to search our source and a browser is already situated in the source search page (probably due to the fact that this browser was used for a previous request with the same wrapper and, subsequently, the wrapper back sequence was executed on it), then execution of the new sequence to said browser is assigned, which will then only follow steps 4 and 5 of same, thus avoiding the cost of steps 1-3.

As mentioned in the preceding section (5.2.4), it is not always advisable to reuse browsers (REUSABLE BROWSERS options checked). It can occur that dealing with a previous request changes the browser response to subsequent requests (for example, through the use of cookies), which makes its reuse inadvisable. The typical case is when an attempt is made to access a source in which another browser is authenticated; often, when navigating to the home page the entry form is not requested again (login/password), whereby the sequence will fail, as it cannot find it.

However, using the PoolAssignmentStrategy strategy sometimes it is possible to reuse browsers in this scenario, if



all accesses to the source share the same login/password pair, as this strategy prevents the browser from trying to execute the authentication steps again, as it considers them part of the initial common steps.

If there are cookie sessions in the source and a different login/password pair is used for each access, then REUSABLE BROWSERS must be unchecked.

When it is possible to reuse a browser from a previous query, it is good to do so, even if the sequence executes from the beginning, because you save having to create a new browser for each query (if the pool load is high, this is very noticeable).

### 5.2.5 Initializing the pool

The browser pool can be configured to automatically initialize a certain number of browsers belonging to any of the three types, with a specific navigation sequence. This functionality is useful, when the navigation sequences to be executed by application share a series of initial steps (e.g. establishing a session through an authentication process) with which we want to save time when executing requests. Using this functionality and the PoolAssignmentStrategy assignment policy it is possible to improve the response times of the system in these cases.

For each required navigation sequence two parameters must be specified:

- POSITION. NSQL program that implements the navigation sequence (e.g. "navigate,http://www.denodo.com,1;")
- INITIAL\_BROWSERS. Number of browsers we want the pool to initialize with this navigation sequence.

If no navigation sequence is specified in this section, the pool will not automatically start up any browser, when it initializes; instead it does so as requests are received.

Figure 13 shows the administration tool page where to configure these parameters.



The screenshot shows a configuration window titled "Pool Initialization". It contains two input fields: "POSITION" and "INITIAL BROWSERS". Both fields are currently empty.


Figure 13 Pool Initialization

### 5.2.6 Firefox Installation Base Path Configuration

This section shows how to configure Firefox to be used in the ITPilot execution environment.

- Firefox Home directory: Firefox installation base path.
  - In this case, besides updating the directory in the administration tool, the plugin `<DENODO_HOME>/setup/itpilot/dll/iebrowser/denodo-runtime.xpi` must be installed by executing the `firefox -install-global-extension denodo-runtime.xpi` command from that same directory.
  - Firefox does not provide a plugin uninstaller, therefore if required, it will have to be deleted manually as a directory in the firefox installation (usually, `\extensions\{800f0371-e961-44b9-97a6-2d9d8b7147b8}`).

### 5.2.7 Executing and stopping the Browser Pool

The browser pool can be started and stopped from the Denodo Platform Control Center (see the Denodo Platform Installation Guide [DENINST]), or by using the “Start/Stop”  button in the browser pool configuration window. This last option is only possible when it is located in the same machine in which the web administration tool is executed and the connection settings in the tool include the local path to the server.

It is also possible to start up or stop the pool using command line. The following scripts are available for this in the path `DENODO_HOME/bin`:

- `browserpool_startup`. Starts the browser pool.
- `browserpool_shutdown`. Stops the remote pool and all the browsers contained in it.

It is important to remember that in order for changes to the pool configuration to take effect the pool has to be stopped and restarted.

## 5.3 CONFIGURATION OF THE WRAPPER SERVER

The wrapper server configuration window (see Figure 14) allows the administrator to control all the configuration parameters of the stated server, as well as monitor and execute different wrappers that are stored. The configurable elements of this window are detailed below.

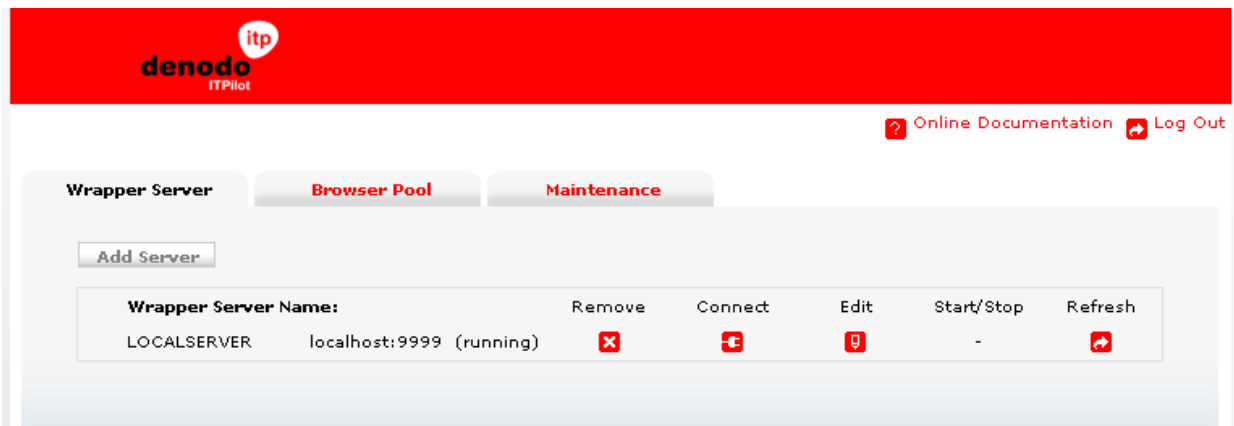



Figure 14 Wrapper Server Configuration Window

### 5.3.1 Access to Wrapper Server

As it can be seen in Figure 14, this area shows the group of wrapper servers currently available in time, as well as the possibility of adding new ones. When a new Server is added, the host and port where it is located can be configured in the server connection settings (remember that if it is configured as residing in the same machine as the administration tool, the wrapper server will be allowed to be directly started up from the web interface; otherwise, it should be started manually following the instructions in section 5.1).

To configure one of the servers of the list, connect to it in order to access its management panel, by pressing the  button of the desired wrapper server. A window is shown where the user must type the login and password to connect the wrapper server (the default values are “admin/admin” in case ITPilot is the only Denodo product installed). It is possible for the system to remember these data during the session by pressing the “Remember my ID”

checkbox.

A new page will be shown like the one in Figure 15. From here the contents of the server can be managed; as each server contains several databases, the first thing that has to be selected is what database is going to be configured. To change the database to inspect, use the “Selected Database” combo box. In case of having installed only Denodo ITPilot, then there will be only one database available (“itpilot”). Once the selected database has been changed, a list of wrappers contained in said database will be displayed.

The following sections explain the different settings that can be configured in the wrapper server. After modifying any of the settings, the “Save Changes” button must be clicked for the system to store all the changes. The server must be restarted after a configuration change for the changes to become effective.

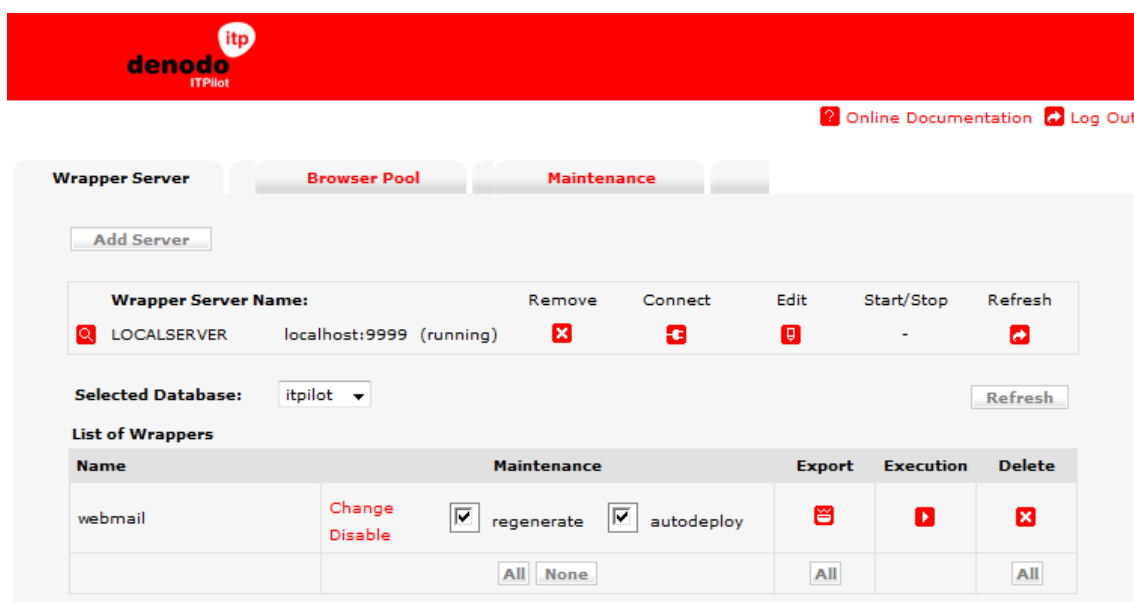



Figure 15 Wrapper server Connection


### 5.3.2 List of Wrappers

Once the system has connected to the wrapper server, the Web tool displays the list of wrappers contained in this server, under the section “List of wrappers”. Also, the total number of wrappers in the database is displayed next to the title of the section. The data displayed for each of the wrappers is as follows:



- Name: the name of the wrapper.
- Maintenance: selection option that indicates whether the wrapper selected will be maintained automatically or not. Clicking on the “Enable” link will activate the wrapper maintenance. It is possible to enable the maintenance for all the wrappers of the database at the same time, by using the “All” button. Once the maintenance is enabled for a wrapper, it can be disabled again by clicking the “Disable” link, or for all the wrappers by clicking the “None” button. With the wrapper maintenance enabled, the following options are available:
  - o The parameter “regenerate” indicates whether ITPilot must try to automatically regenerate the wrapper when a source change is detected or not.
  - o The parameter “autodeploy” sets the system to automatically install the regenerated wrapper in the server, replacing the old version. If this parameter is not selected, the new wrapper is stored in the path `DENODO_HOME/metadata/maintenance-regenerations` (with `DENODO_HOME` being the folder where the Denodo Platform is installed in). The replaced versions are stored in the `metadata/maintenance-backup` path, adding the date in

which the wrapper was updated.

When a wrapper cannot be automatically maintained, the web administration tool will show the  icon next to the “Enable” link. Despite not being maintainable, the wrapper can still be configured as “Enabled”; this will make the source to be monitored by ITPilot so that, in the event of the source changing and the wrapper needing to be modified, an electronic mail will be sent to notify the situation (see section 5.4).

- Export: by clicking on the  button, a new web page is shown from where the wrapper specification can be exported to the file specified by the user. The option is provided to include in the file the scanners that the wrapper uses.

Selecting the “All” button in the “Export” column will export all the wrappers of the database.

- Execution: by clicking on the  button, the wrapper can be executed as seen in the next section.
- Delete: pressing the button , the wrapper is eliminated from the server. Clicking the “All” button in the “Delete” column will delete all the wrappers of the database.

### 5.3.2.1 Wrapper Execution

The administration tool allows queries to be made to the wrappers through the “Execution” option mentioned earlier.

Figure 16 displays the Execution window. The checkboxes on the left of the screen allow to select which output fields of the wrapper are going to be displayed in the result table. The text fields of the right part of the screen are used to set the values of the input parameters of the wrapper; the mandatory fields are displayed with the “(Mandatory)” text next to the input field.

After filling the input values and selecting the output fields, by clicking the “Execute” button the administration tool will start the execution of the selected query. To achieve this, it communicates with the wrapper server and invokes the typed query for the specific wrapper; this is communicated to the appropriate data source. The results, properly structured, are shown in the execution window result list in an asynchronous way: the individual rows will appear in the results table as soon as they are available.

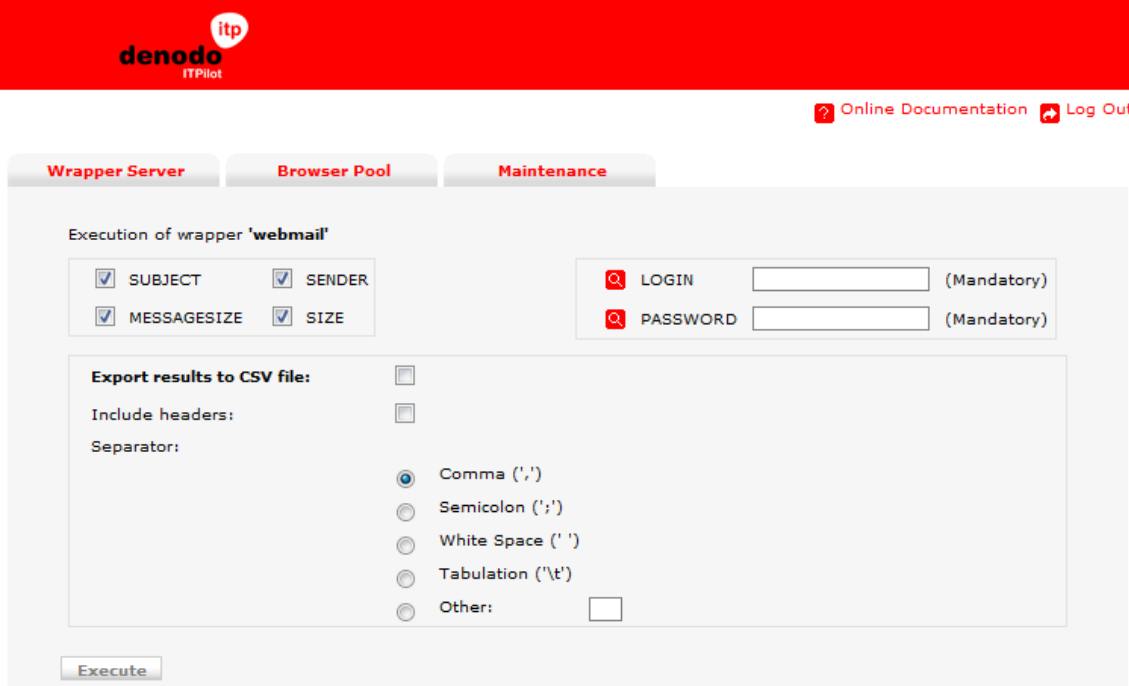


Figure 16 Wrapper Execution Page

### 5.3.2.2 Exporting results as a CSV-formatted file

Before pressing the Execute button, the results of the execution can be configured to be stored in a CSV (Comma-Separated Value)-type file (where Denodo ITPilot allows the definition of the character used as a separator between columns). When the execution is finished, the browser will show a standard download dialog to allow to save the results in the desired folder/file.

The “Include Headers” checkbox allows to include the names of the output fields as headers in the CSV file.

### 5.3.3 Selecting location of the associated browser pool

The wrapper server requests browser instances to the browser pool when a wrapper that performs web browsing is executed. The administration tool allows to configure what browser pool is going to receive the wrapper server requests, by using the selector in the “Browser pool configuration parameters” section (see Figure 17). This selector displays all the browser pools that have been configured in the “Browser Pool” tab of the web administration tool (as discussed before).

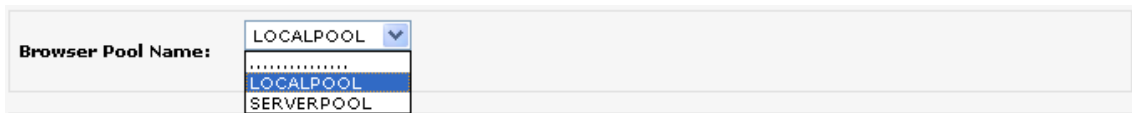


Figure 17 Pool Browser Localization

### 5.3.4 Port Assignment

In this section the following parameters can be configured:

- Application Port: port through which the wrapper server listens and waits for requests.
- Shutdown Port: port through which the server listens and waits for the Shutdown signal.
- Auxiliary Port: used for communications between the browser pool and the wrapper server.

### 5.3.5 Configuration of HTTPClient with no Browser Pool

The http client (Denodo HTTPBrowser) can be executed from the ITPilot wrapper server without having to start a browser pool in a separate process, since the wrapper server internally manages an HTTP client pool itself. This internal pool can be configured in the “HTTP configuration parameters” section. At generation time, a user can choose if an independent pool will be used to run the HTTPBrowser clients, or if they will be executed in the internal pool of the wrapper server (please see [GENER]). The parameters are basically the same ones that were described in section 5.2.

The screenshot displays the 'HTTP configuration parameters' section of the Wrapper Server administration interface. It is organized into several distinct panels:

- HTTP Configuration:** Includes a 'MAX DOWNLOAD' field set to 60000 and an empty 'OBJECT TIMEOUT' field.
- Download Controls:** Features three unchecked checkboxes: 'NOSCRIPT', 'NOJAVA', and 'NOACTIVEX'.
- Cache Controls:** Includes an unchecked 'NOCACHE' checkbox and a 'Maximum number of cached JavaScript files' field set to 150.
- Proxy with Authentication:** Contains three empty input fields for 'PROXY LOGIN', 'PROXY DOMAIN', and 'PROXY PASSWORD'.
- HTTP Pool Size and Reutilization Policy:** Includes 'MAX POOLSIZE' (250), 'MIN POOLSIZE' (empty), and 'MAX BROWSER TTL' (empty) fields. It also has 'REUSABLE BROWSERS' (unchecked), 'REUSABLE' (checked), and 'ASSIGNMENT STRATEGY' (SimplePoolAssignmentStrategy).
- PDF Conversion Configuration:** Features a 'PDF to HTML Converter' dropdown (Acrobat HTML), an empty 'Conversion server port' field, and an empty 'Acrobat Prof. plugins directory' field.

Figure 18 Configuration parameters of the http client in the Wrapper Server

### 5.3.6 Password change

The button “Change password” lets the user change the access password of the wrapper server which the user is currently connected to.

### 5.3.7 Loading new wrappers from VQL files

Although wrappers are normally exported from the specifications generation tool to the wrapper server, VQL files containing the definition of a wrapper can also be loaded manually from the administration tool (for more information about VQL, please read [VQL]). This is useful when the specification has been produced entirely manually. To do so, first select the file to load by typing the path to the file in the input field next to the “Load VQL File” button or clicking the “Choose” button and graphically selecting the file. Once the path is ready, click on the “Load VQL File”. The wrapper will appear in the list of database wrappers from which it has been loaded.

### 5.3.8 Creating a Web Service

The wrappers deployed in the execution server can be invoked in different ways. The native Denodo ITPilot Java API can be used to access the wrappers, obtain their data structure and run queries on them from a Java application. Another option is to show these wrappers through SOAP- or REST-type Web Services. At last, ITPilot lets users export their wrappers as small web applications that accept a URL with input parameters and return a page with the results obtained after the execution. A description of the use of these options can be found in the ITPilot Developer's Guide [DESAR]. In the case of Web Services, they are created from the Web administration tool.

This section describes the procedure to generate a Web Service based on an example included in the ITPilot

distribution. Therefore, before following the directions the appropriate wrapper must be deployed in the server. To do so, select the “webmail.vql” file in the ITPilot installation path in “samples/itpilot/itp-clients/scripts/”, click on “Browse...” and then on “Load VQL” (see Figure 19). The wrapper will appear in the list of wrappers, as shown in Figure 20.

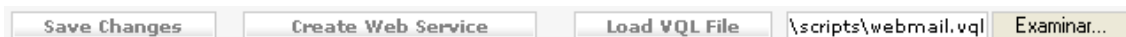


Figure 19 Loading Wrappers Using VQL Files

Name	Maintenance	Export	Execution	Delete
google	Enable			
webmail	Enable			
	<input type="button" value="All"/> <input type="button" value="None"/>	<input type="button" value="All"/>		<input type="button" value="All"/>

Figure 20 List of Wrappers with Loaded “Webmail”

You can then generate the Web service by clicking on the “Create Web Service” button on the execution server tab, after which a page will appear as shown in Figure 21 from where the Web Service to be generated is described:

- Web Service Name: name to be given to this service. For example, “webmailws”.
- Wrapper Service URL: this is the URL of the wrapper server where the wrapper is deployed, to be used to access the wrapper through the Web Service. The default value is “//localhost:9999/itpilot”, where “localhost:9999” is the domain and port where the run server resides and “itpilot” is the database where the wrapper is deployed.
- Login/Password: login and password to access ITPilot. In this case and by default, “admin/admin”.
- Query Timeout: maximum waiting time for a query result (left blank to take the default value).
- ChunkTimeout: maximum waiting time between two consecutive results (also left blank).
- ChunkSize: chunk size for each operation (also left blank).
- Web Service Style: Web Service style to generate (RPC or DOCUMENT). Some Web Service consuming applications may require one specific style.

**'itpilot' Web Service configuration parameters**

Web Service Name (\*)

Wrapper server URL (\*)

Login (\*)

Password

Query Timeout

Chunk Timeout

Chunk Size

Web Service style  
 RPC  
 Document

Add Operation	Wrapper Name	OBL Operation Name	OPT Operation Name	OPT Fields
<input type="checkbox"/>	webmail	<input type="text"/>	<input type="text"/>	

**Figure 21** Web Service Export Page

Once the data describing access to the server have been configured, the next step involves defining the Web service operations. Denodo ITPilot allows to generate three operations per wrapper. The first one will contain all the mandatory parameters (those marked as searchable and mandatory attributes in the specification, see [GENER]. Where there are no mandatory parameters, the query would be run without parameters); the second one will additionally contain any searchable and optional attributes selected in the "OPT FIELDS" column. In this example, there are no optional parameters and, therefore, only one operation will be created, known as "getMails", by writing this name in the text field of the "OBL Operation Name" column corresponding to the "webmail" wrapper. Select the "Add Operation" column to mark the operation to be added to the web service. Lastly, the third operation contains all mandatory and optional attributes.

ITPilot allows to generate the Web Service as a .war file. Optionally, it can also generate the WSDL file for the SOAP-type Web Services. Pressing the "Create Web Service" and "Create WSDL" buttons, the user will be able to locally store those files.

Where required, this action can also be tested using the sample programs to be found in the ITPilot installation path in the directory samples/itpilot/itpilot-clients. The samples/itpilot/itpilot-clients/README file should be read.

## 5.4 CONFIGURING THE MAINTENANCE SERVER

Denodo ITPilot offers a component for automatic maintenance of wrappers, that detects changes in the web sources and rebuilds the wrappers that access them by analyzing those changes, thus making said wrappers more robust and reducing the manual maintenance time. The main concept on which this component is based is the storage of the results of successful queries of the wrappers. When a change is detected in the source (by configurable rules that detect invalid results from a wrapper), the results of past queries are matched against the changed web source, resulting in patterns that lead to the regeneration of the wrapper; this new wrapper will work correctly with the modified web source, without having to be edited manually.



This component is made available through the maintenance server, whose configuration process through the Web administration tool is detailed in this section.

### 5.4.1 Access to the Maintenance Server

As it can be seen in Figure 22, this area displays the group of maintenance servers that are available at the moment, together with the possibility of adding new ones. Normally only one will be started, but if the size or quantity of sources requires it, the option of simultaneously running several of them is also available. When adding a new maintenance server to the list, its connection settings (domain and port, and optionally local route) can be configured in the same way as the settings for browser pools and wrapper servers (remember that if the server is configured with a local route it will be possible to start and stop it from the administration tool; otherwise, it can be started manually following the instructions in section 5.1 or by using the Denodo Platform Control Center - see the Denodo Platform Installation Guide [DENINST]).

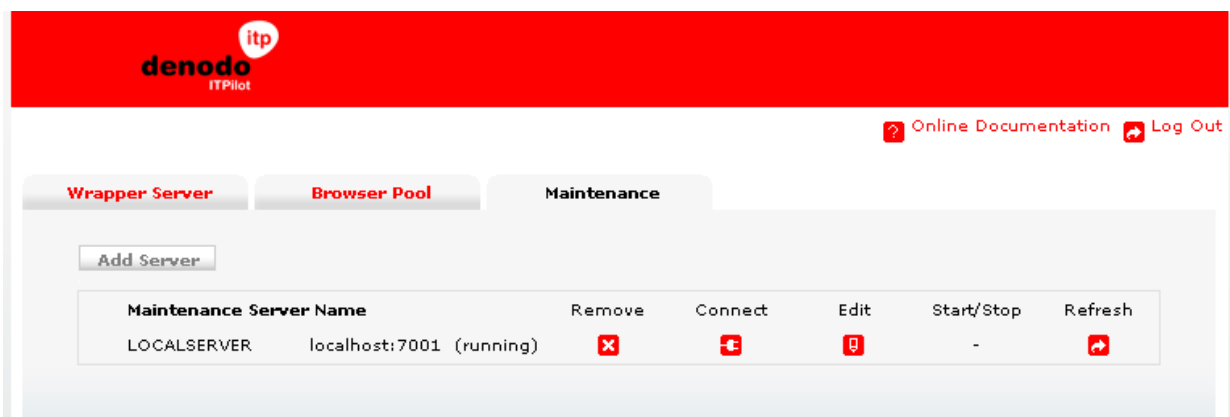


Figure 22 Maintenance Administration Main Page

The fields to be completed are as follows:

- Name: a name to identify this server.
- Host: address where it is found.
- Port: port where the server is listening.
- Local path: optional, to indicate that the server is local; by adding the local path, where the application is, the user will be able to start up and stop the maintenance server from the graphical administration tool.

### 5.4.2 Server Configuration Data

Once the system has connected to the maintenance server (either by clicking the tab or editing the access data, as in the previous section), the Web tool displays the server configuration data. The following settings are available:

#### 5.4.2.1 Database parameters

- Provider: database provider (by default: `derby`). The possible values are `derby`, `mysql`, `postgresql`, `oracle`. When selecting a value, sensible default values will be assigned to the rest of database parameters, when applicable.
- JDBC URL: URL access to the Database for the JDBC driver (by default: `jdbc:derby:maintenance`).
- User/Password: user and access password (by default: `maintenance/maintenance`).
- JDBC driver: JDBC driver to be used (by default: `org.apache.derby.jdbc.EmbeddedDriver`). If the driver is not distributed in the Denodo Platform, it must be placed in the `DENODO_HOME/extensions/thirdparty/lib` directory or

its path must be added to the `DENODO_EXTERNAL_CLASSPATH` environment variable so Denodo can find it.

- Pool size: maximum number of simultaneous connections the pool will allow (by default: 5).
- Test query: test query executed on the DBMS. The connection pool, before assigning any of the free connections in the queue, will check to verify if the connection is valid or not (by default: `SELECT * FROM ping_table`).

Figure 23 shows these configurable parameters in the maintenance server tab:

Connected to	<b>LOCALSERVER</b>
<b>Database Parameters</b>	
Provider	<input type="text" value="derby"/>
JDBC URL	<input type="text" value="jdbc:derby:maintenance"/>
User	<input type="text" value="maintenance"/>
Password	<input type="text" value="maintenance"/>
JDBC Driver	<input type="text" value="org.apache.derby.jdbc.EmbeddedDriver"/>
Pool Size	<input type="text" value="5"/>
Test Query	<input type="text" value="SELECT * FROM ping_table"/>

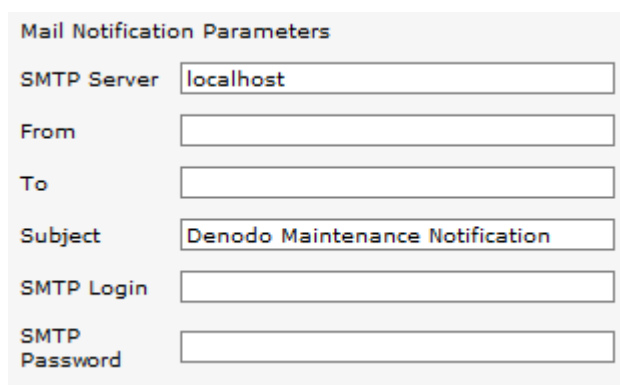
**Figure 23** Maintenance database Parameters

#### 5.4.2.2 E-mail notification parameters

These parameters will be used to notify via e-mail those changes detected in the sources:

- SMTP Server: name of the mail server
- From: e-mail address from which the notification is emitted.
- To: e-mail address to which the notification is sent.
- Subject: e-mail subject.
- SMTP Login (optional)
- SMTP Password (optional)

Figure 24 shows these configurable parameters in the maintenance server tab:



The screenshot shows a form titled "Mail Notification Parameters" with the following fields and values:

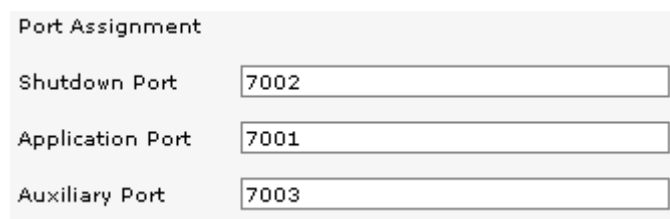
Parameter	Value
SMTP Server	localhost
From	
To	
Subject	Denodo Maintenance Notification
SMTP Login	
SMTP Password	

**Figure 24** Wrapper Change Notification Parameters

#### 5.4.2.3 Port Assignment Parameters

- Application Port: port used by the maintenance server to communicate with the wrapper server.
- Shutdown Port: port used by the server to wait for the Shutdown signal in order to finish its execution if received.
- Auxiliary Port: communication port between the maintenance server and its clients.

Figure 25 shows these configurable parameters.



The screenshot shows a form titled "Port Assignment" with the following fields and values:

Parameter	Value
Shutdown Port	7002
Application Port	7001
Auxiliary Port	7003

**Figure 25** Port Assignment Parameters

#### 5.4.2.4 Other configuration parameters

- Server address: this is either the name or the IP address of the computer where the maintenance server is located. This parameter is used by the maintenance server to automatically configure the wrapper server, so the wrapper server knows how to contact the maintenance server. Because of this, the Server address has to be a name or IP address that is reachable from the computer where the wrapper server is being executed.

#### 5.4.2.5 Edition of Verification Rules

The ITPilot automatic maintenance system requires the generation of a set of rules to check which wrappers have changed. The administrator can create as many rules as desired, and they can affect a single wrapper or the whole set. If the administrator wishes to select a wrapper as the single target of a rule, that wrapper must have been executed at least once while the Maintenance Server is active.

Rules are composed by entries, each of which is a check of the wrapper or wrappers. When all of the entries are successfully checked, that rule is activated.

The activation of any of the rules of a wrapper is enough to consider that the wrapper has changed.

Figure 26 shows an example in which a couple of rules have been defined; the first one is composed by a set of three entries, and the second rule by a single one. Remember that every entry must be verified for the rule to be considered valid, therefore validating the rule and proceeding to start the automatic maintenance.

Each entry in the rules consists of one of the tests described in the following paragraphs. Each test will return a percentage value (where 100% means the total accomplishment of the checking performed in that test):

- *ZeroResults*: checks whether the source returns any result or not. The intuition behind this test is that if a significant number of queries do not return any results, a possible reason is a malfunctioning of the current wrapper. This test will return "0" if there are no results, and, on the contrary, "100".
- *Compatibility*: checks the compatibility between the results and the query. E.g. if "**title=java**" is searched, then the returned results should contain the word "java" in the "**title**" field of the extracted tuples. The opposite would mean that the current wrapper might not be correctly extracting the data from that field, and thus it might be necessary to regenerate it. The percentage value is calculated proportionally to the number of tuples which verify the compatibility test with regard to the total ones.
- *Consistency*: checks whether the results match the regular expressions defined in the wrapper metadata (see [GENER]). The intuition behind this test is similar to the previous test: if the results do not verify the pointed out regular expressions, it is probable that the current wrapper is not correctly performing the extraction process, and thus, it must be regenerated. The percentage value is calculated proportionally to the number of tuples which verify the regular expressions with regard to the total ones.
- *Invariability*: checks that a certain result percentage of the results of some query is maintained when that same query is executed some time later. The intuition behind this test is that, in some sources, very abrupt changes in the extracted results for a same query alongside time, might indicate a malfunctioning of the current wrapper. The percentage value is calculated proportionally to the number of tuples which are kept since last query executions with regard to the total ones.
- *Pagination*: checks that in every intermediate result page returned by the wrapper (all but the last one), the number of returned tuples is the same. If any intermediate page does have fewer results than others, this could mean that the wrapper is omitting some relevant results (take into account that web sources usually paginate their results in intervals with a fixed number of results for each one). The returned percentage value is calculated as a function of the deviation of the obtained number of tuples with regard to the expected number of tuples. This expected number of tuples is calculated by supposing that each intermediate page returns the maximum number of results obtained for some of the pages.
- *ResultsNumber*: checks that the number of tuples obtained in successive executions of a same query alongside time is similar. The intuition behind this test is that, in some sources, very abrupt changes in the number of extracted results for a same query could indicate a malfunctioning of the current wrapper. The percentage value is calculated proportionally to the deviation of the number of tuples returned by the query with regard to the average of the last executions of that query.

The verification rule editor allows the configuration of each entry in the following manner:

- **Test**: test to perform (Invariability, Pagination, etc.)
- **Amount**: number of executions of the wrapper that this test must carry out for this entry to be activated. This quantity must be taken contextually to the execution interval which will be taken into account, as configured in "Interval".
- **Interval**: wrapper executions which are taken into account in this test. The value "0" indicates the last execution performed, "1" is the one before the last, and so on.
- **Values**: each test execution returns an integer value between "0" and "100", closer to "0" when the results are worse with regard to the performed test. This parameter determines the range of values which would activate the test.

Let us now consider the example of Figure 26. In the first rule, their entries mean the following:

- First entry: it uses the test "ResultsNumber". It will get activated when the returned percentage value by any query is below 50% in at least one (amount=1) of the last ten executions, except for the last one (interval= 1-10).
- Second entry: it will be activated when the result for the ResultsNumber test is 0 in the last execution of

- any query.
- Third entry: it will get activated when the result for the Pagination test is 0 in the last execution of any query.

Figure 26 Edition of Verification Rules

#### 5.4.3 Selecting location for the associated browser pool

During wrapper maintenance the server requires that the “iebrowser” component be used as an access method, whereby a browser pool should be used. Its location can be indicated in the administration tool window by selecting in the combo box the name used in the “Browser Pools” tab to identify it (see Figure 27).

Figure 27 Locating the browser pool

#### 5.4.4 Selecting location of wrapper server

Likewise, the maintenance server needs to access the wrapper server, where the wrappers in execution are stored, so that it can detect changes and regenerate them automatically. In the list “Wrapper Server Name” you can select the required wrapper server from all those that have been created in the “Wrapper Server” tab (see Figure 28).

Figure 28 Locating the Wrapper Server

Clicking on the “Save Changes” button allows you to store all the changes made. In each tab there is a “Save Changes” button, which saves the configuration, i.e. it sends the new configuration to the corresponding server for it to be stored on disk. For the new configuration to take effect the corresponding server must be relaunched.

#### 5.4.5 Capabilities and limitations of the Maintenance Server

The ITPilot maintenance server includes advanced functionalities for automatically maintaining web wrappers when the target web sources change. When a wrapper is maintainable, ITPilot will be able to regenerate the wrapper automatically in a **high percentage of cases**. Nevertheless, not all wrappers are maintainable. This document briefly describes the requirements a wrapper must verify in order to be maintainable.

#### 5.4.5.1 Query Wrapper Model

The automatic maintenance support is restricted to wrappers following what we call the “Query Wrapper” model. The workflows of the wrappers according to such model verify some structural patterns:

1. The wrapper starts with a navigation sequence which navigates to a certain page containing some data to be extracted (usually, this initial sequence involves filling some web query form using some input parameters of the wrapper). For instance, an initial sequence can navigate to the query form of an Internet bookshop, fill the *title* and *author* fields and submit the query. In Denodo ITPilot this step is accomplished using a SEQUENCE component.
2. The data records contained in the page reached by the former sequence are extracted. For instance, in our example, the books present in the page obtained as response of submitting the form are extracted. In ITPilot this step is accomplished using an EXTRACTOR component.
3. The page containing the extracted data records may contain outbound links to navigate to other pages containing the following “result intervals”. These new pages can contain links to still more results. For instance, in our example, each page may contain 25 books verifying the query and a ‘Next’ link allows accessing the following interval. In ITPilot this step is accomplished using a NEXT\_INTERVAL\_ITERATOR component.
4. The pages containing data records may contain outbound links to access “detail” pages for each record. The info in that detail pages may need to be extracted. In our example, we may have a “More info” link associated to each book allowing to extract additional information about each book. In ITPilot, navigation to detail pages is usually accomplished using an EXTRACTOR\_SEQUENCE component (that replaces the RECORD\_SEQUENCE component of previous versions of ITPilot).
5. Finally, for each data record extracted, some post-processing operations may be needed. For instance, in our example, an arithmetic expression could be used to compute the final price of each book from the extracted attributes ‘price’ and ‘special discount’. Post-processing operations are usually performed in ITPilot using components such as CONDITION, EXPRESSION or RECORD\_CONSTRUCTOR.

The only mandatory steps are (1) and (2). The other steps may be present or not.

The ITPilot wrapper generation tool includes a button which allows to automatically check if a wrapper verifies the restrictions imposed by the query wrapper model. This does not imply that the wrapper can be successfully regenerated by the maintenance server. The following section describes some of the existing additional constraints.

#### 5.4.5.2 Additional Constraints

If a wrapper is maintainable, it will be successfully regenerated in a high percentage of cases. Nevertheless, in some cases the regeneration process may fail. These are some of the main cases when the regeneration will not be successful:

- If any of the new navigation sequences after the change in the source, require launching pop-up windows using Javascript, the wrapper will not be regenerated. If the pop-ups that need to be used are launched using the target attribute of an anchor, then they are supported.
- The component used in ITPilot to extract structured data records from a HTML page is called EXTRACTOR. One of the options of the EXTRACTOR component allows establishing a FROM clause to limit the area of the page where data records will be searched for. If any of the new EXTRACTOR components after the change in the source requires this clause, the wrapper will not be successfully regenerated.
- If the new navigation sequence required to access detail pages requires filling a form, then detail sequences will not be regenerated.
- The EXTRACTOR component internally uses elements called *tagsets* for data extraction programs. Although it is rare, after the change, the source may require different *tagsets* from the ones previously used. In that case, the regeneration process will fail.
- If the updated source shows web dialogs, the regeneration process will not be successful.

## 6 MAINTENANCE SERVER GRAPHICAL CONFIGURATION AND MONITORING TOOL

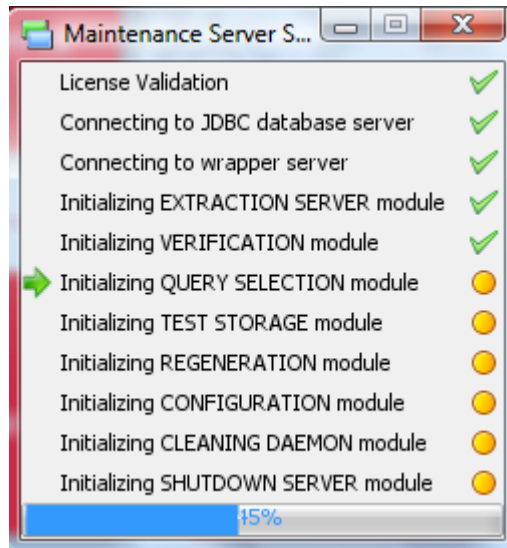
Denodo provides a graphical configuration and monitoring tool that can be used in production time to monitor the actions of the maintenance Server, once the wrapper Server is running.

The remainder of this chapter is divided in two sections:

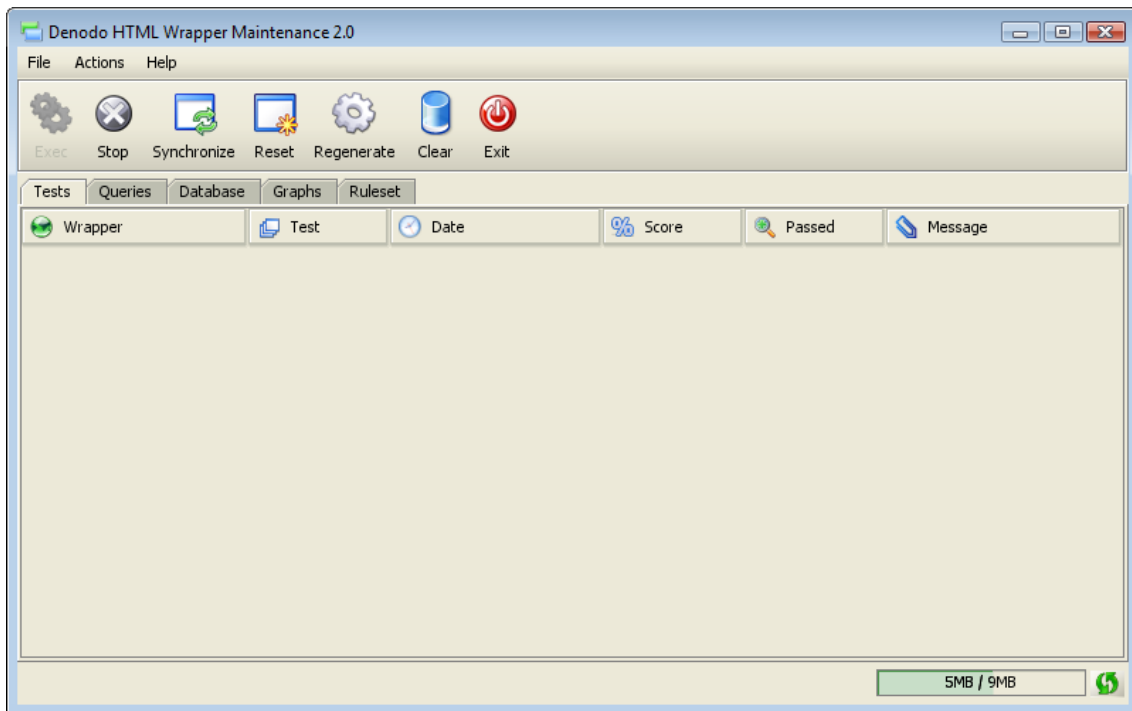
- Configuration section, that explains how to configure the different rules and tests for each wrapper loaded in the execution Server (as it was already explained in section 5.4.2.5), and the different parameters of the graphical tool.
- How to monitor the wrappers that have been configured as maintainable in the wrapper server

### 6.1.1 Configuration of the wrappers to be maintained

After following the steps described in section 4.5, the graphical tool will be started. Please remember that the maintenance server must be already running. Figure 29 shows the window that appears right after starting.



**Figure 29** Maintenance Graphical Tool Loading Window



**Figure 30** Maintenance Graphical Tool

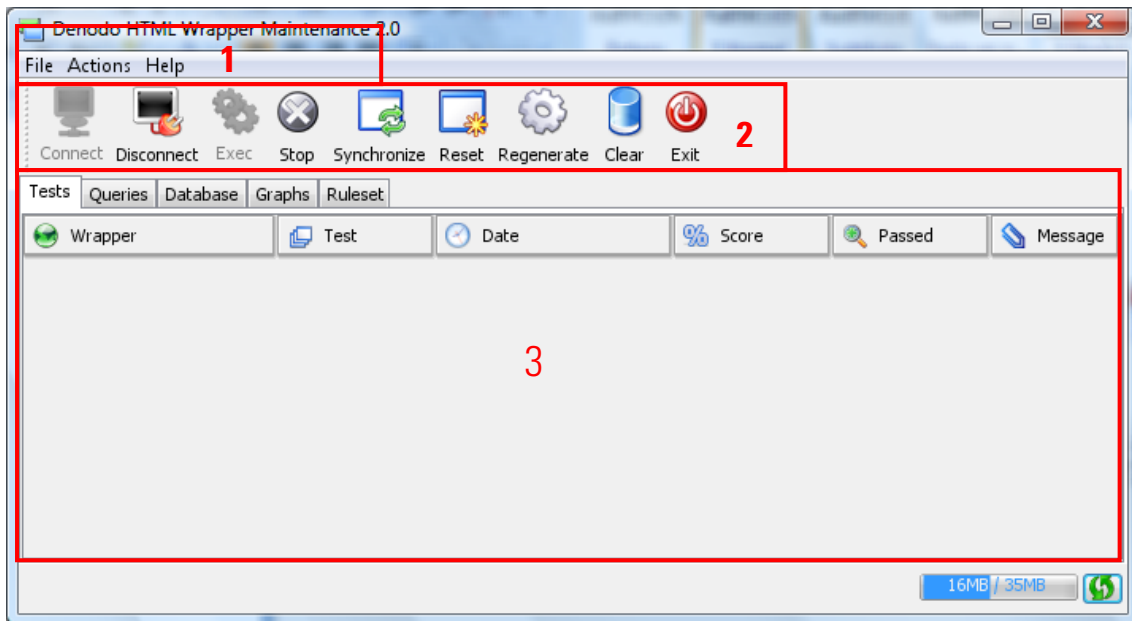
It is possible to enable the automatic regeneration process of the maintenance Server. This action is useful when no wrapper regeneration is required, but just sending an email alert (configured in the Email Configuration Parameters area of the ITPilot Web Administration tool, see section 5.4.2.2) when any of the wrapper rules is activated.

The automatic regeneration can be enabled and disabled for all wrappers, but not in an individual manner. To disable the automatic regeneration for all wrappers residing in the server, the steps are the following:

1. Stop the maintenance server if it is running.
2. Edit the file  
`<DENODO_PLATFORM>\conf\maintenance\MaintenanceConfiguration.xml`  
with a text editor (the use of notepad in MS Windows systems is not recommended).
  - Search and replace the string `<handler enabled="true" name="Regeneration"/>` by `<handler enabled="false" name="Regeneration"/>`
  - Save the file.
3. Start the maintenance Server



## 6.1.2 An overview of the Graphical Tool



**Figure 31** Main elements of the Maintenance Graphical Tool

The main screen of the maintenance configuration and monitoring tool is divided in three main areas, as shown in Figure 31.

1. Menu: The menu gives access to all functionalities of the administration tool.
  - File:
    - Close GUI
    - Exit Application
  - Actions:
    - Enable maintenance server engine
    - Stop maintenance server
    - Wrapper server synchronization: it executes a synchronization process with the wrapper server.
    - Database clean & resynchronization: it cleans a re-synchronizes the maintenance database.
    - Regenerate wrapper
    - Clear screen information
  - Help:
    - About
2. Actions Buttons:

- **Exec**: enables the maintenance server.
  - **Stop** maintenance server
  - **Synchronize**
  - **Reset**: cleans and re-synchronizes the maintenance database
  - **Regenerate the** wrapper
  - **Clear** screen information
  - **Exit**
3. **Main tabs**: this is the main area of the tool, from where all information about maintainable wrappers can be configured, and where all their rules can be configured and monitored (this action is identical to the one offered in the Web Administration tool, in section 5.4).
- **Tests**: this tab shows information about each of the performed executions, and the configured test results obtained after each execution.
  - **Queries**: this tab shows information about the different queries performed.
  - **Database**: this tab shows all information stored in the maintenance database.
  - **Graphs**: this tab lets users create different graphs from both the wrappers and tests that are executed in a specific period of time.
  - **Ruleset**: lets users create and configure different rules, required by the maintenance server.

### 6.1.3 Configuration of the Maintenance Environment








The ITPilot automatic maintenance server requires the generation of a ruleset that lets users check if the wrappers have changed. The administrator may create as many rules as necessary, and they can affect a single wrapper or the complete set.

TestName	Amount	Interval	Values
ALL WRAPPERS	1	10	50
ITPILOT.WEBMAIL	1	2	0
ALL WRAPPERS	1	0	0

**Figure 32** Rule Configuration in the Maintenance Graphical Tool

Figure 32 shows an example, in which a rule has been defined; this rule is composed by three tests. For more information about each of the rules, please read section 5.4.2.5. Though defining the rules from the web administration tool is highly recommended, this action can also be performed from the monitoring tool;

1.  *Addition of a new rule*: it creates a new rule for all wrappers in the database, or for a specific one.

2.  *Addition of a new test*: it creates a new test in the selected rule.
3.  *Rule deletion*: it deletes the selected rule, without deletion confirmation. After deleting the rule, the  button ("Rule storage") must be pressed to confirm changes.
4.  *Return to Previous State: Stored Rules*: it goes back to the previous stored state, getting rid of the latest changes.
5.  *Rule storage*: it stores the latest changes.
6.  (to the right of each test) *Test deletion*: after deleting the test, the  button ("Rule storage") must be pressed to confirm changes.

#### 6.1.4 Monitoring the Maintainable Wrappers

The maintenance graphical tool lets users monitor the execution of the maintainable wrappers that have been loaded into the wrapper Server. This can be achieved by using the following tabs:

- Tests: this tab displays information about the tests that have been executed, and their results. The information available is the following:


Tests					
Wrapper	Test	Date	Score	Passed	Message
itpilot.webmail	ZeroResults	2008-02-11 12:39:48	100	✓	21 results found in 1 pages
itpilot.webmail	ZeroResults	2008-02-11 12:39:20	100	✓	21 results found in 1 pages

**Figure 33** Tests Tab

- Wrapper: database and wrapper name
  - Test: name of the test that has been executed
  - Date: execution date
  - Score: test result
  - Passed: it reports whether the test was successful or not (if so, it does not participate in the rule activation).
  - Message: provides additional information about the execution.
- Queries: this tab displays information about the queries that have been executed against the wrappers.

Wrapper	Query	Date	Results	Pages
itpilot.webmail	"PASSWORD = DeMo.04" "LOGIN = demos"	2008-02-11 12:39:48	21	1
itpilot.webmail.MainExtractor	"PASSWORD = DeMo.04" "LOGIN = demos"	2008-02-11 12:39:47	21	2
itpilot.webmail	"PASSWORD = DeMo.04" "LOGIN = demos"	2008-02-11 12:39:20	21	1
itpilot.webmail.MainExtractor	"PASSWORD = DeMo.04" "LOGIN = demos"	2008-02-11 12:39:19	21	2

**Figure 34** Queries Tab

- Wrapper::database, wrapper and, optionally, Extractor component name.
  - Query: query that has been executed against the wrapper server.
  - Date: execution date.
  - Results: number of returned results
  - Pages: if related to an Extractor component, number of pages used in execution; if related to a wrapper, the value is always 1.
- Database: this tab displays all information stored in the maintenance database. It is divided in two additional tabs, which are shown with the symbol .

Wrapper	Extractors	Valid Data	Outdated Data	Tests
itpilot.webmail	[MainExtractor]	4 query(s) with 84 row(s)	0 query(s) with 0 row(s)	0

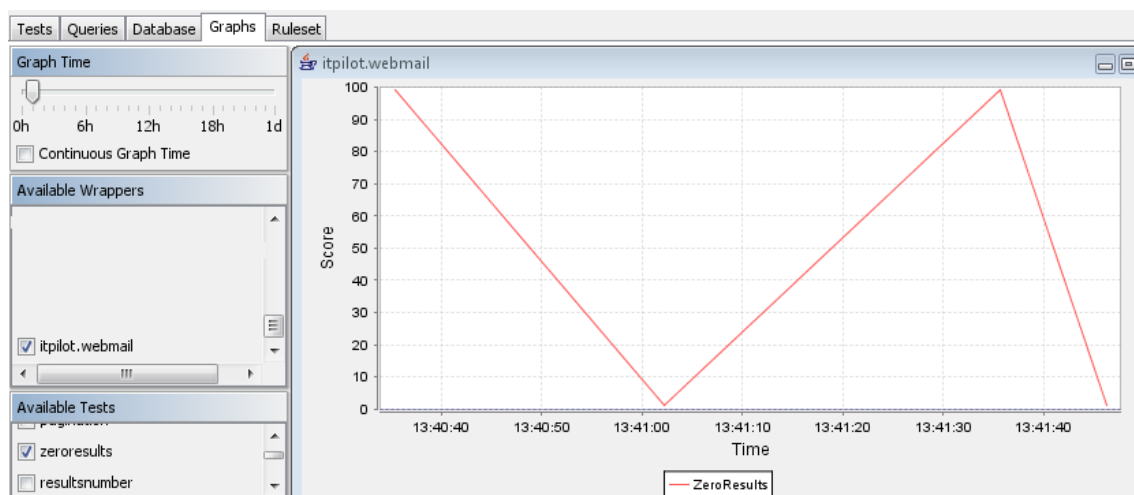
**Figure 35** Database Tab

- *Tab 1*: information about the wrappers that are stored in the maintenance database.
  - *Wrapper*: database and wrapper name
  - *Extractors*: name of the wrapper's Extractor components
  - *Valid Data*: number of queries and updated (not outdated) results, as stored in the maintenance database for that wrapper.
  - *Outdated Data*: number of queries and outdated results stored in the maintenance database for that wrapper.
  - *Tests*: number of tests stored in the maintenance database for that wrapper.
- *Tab 2*: information about the wrapper that is selected in the selection box.

Wrapper	Query	Date	Rows	Status	ID	View
itpilot.webmail	LOGIN=demos PASSWORD=DeMo.04	2008-11-04 17:19:23	21	✓	9	
itpilot.webmail.DetailPageExtractor	LOGIN=demos PASSWORD=DeMo.04	2008-11-04 17:19:23	21	✓	8	
itpilot.webmail.MainPageExtractor	LOGIN=demos PASSWORD=DeMo.04	2008-11-04 17:19:22	21	✓	7	

**Figure 36** Information about the selected wrapper

- *Wrapper*: database, wrapper and Extractor component name
  - *Query*: query used for executing the wrapper
  - *Date*: execution date
  - *Row*: number of results for this query execution
  - *Status*: status of the execution (correct, incorrect, ...)
  - *ID*: execution identifier
  - *View*: button to access detailed information. It will open a new tab showing internal information about this identifier.
- **Graphs**: this tab shows different graphs from the wrappers and the queries that have been executed in a specific timeframe. The choice of which wrappers and tests are to be supervised must be done before the wrapper starts being executed.



**Figure 37** Graphs Tab

## 7 ANNEX A: DEPRECATED FEATURES

### 7.1 ACTIVE X CONTROL FOR AUTOMATIC BROWSING SEQUENCE RUNNING IN CLIENT BROWSERS

ITPilot includes an ActiveX control that enables a Web server to provoke the automatic running of any browsing sequences in a client browser, provided that this browser has been configured to permit this type of action. An example of using this function is a Web automation process such as automatic authentication in a Web application ("autologin"). This is carried out using an ActiveX control installed in the local machine from where a specific browsing action is to be run. This function is extremely useful when some type of Web automation involving automatic browsing is required.

The operation is as follows: the SeqExeAX.cab control is found in the activex/itpilot/ path in the ITPilot distribution installation directory. This control can either be saved in a Web server to be accessed via http or, if the control is already recorded in the local system, it can be accessed through the Windows register via its CLSID. Once this action is complete, Web sites can be created that enable automatic browsing through the addition of the following elements to the HTML code:

```
<object
  CLASSID="CLSID:<CLSID of component SeqExeAX
  CODEBASE="http://<access path to control
>/SeqExeAX.cab#version=<SeqExeAX component version">
<param
  name="Sequence"
  value="NSEQL browse sequence">
```

The CLSID and SeqExeAX component version can be found in the SeqExeAX.inf file in the SeqExeAX.cab component (this can be opened from any unzip tool on the market as if it were a zipped file). The browse sequence is specified in NSEQL language, explained in detail in [NSEQL].

A Microsoft Internet Explorer browser can be launched with the Web server that contains the .cab control running, and the site containing the aforementioned elements can be loaded.

NOTE: It is important to note that the browser must be configured to enable the running of ActiveX controls, which is often done by customizing the security tab in Tools->Internet Options or by selecting the "Low Level" security option in the required Web content area (e.g. "Local Intranet" if it is a local site or "Internet" if the site being run in the sequence is accessible over the Internet).

If the browser is opened with the aforementioned site, it can be seen how the browser automatically runs the browse sequence described in the "value" attribute of the "param" element.

The feature is currently deprecated in ITPilot.

## REFERENCES

- [BEA] BEA Systems Application Server. <http://www.bea.com>
- [DENINST] Denodo Platform 4.5 Installation Guide. Denodo Technologies, 2008.
- [DESAR] Denodo ITPilot 4.5 Developer Guide. Denodo Technologies, 2008.
- [DEXTL] Denodo DEXTL 4.5 Manual. Denodo Technologies, 2008.
- [GENER] Denodo ITPilot 4.5 Generator Environment Guide. Denodo Technologies, 2008.
- [FFOX] Mozilla Firefox Browser. <http://www.firefox.com>
- [IE] Microsoft Internet Explorer. <http://www.microsoft.com/windows/ie/>
- [ISO639] Language codes ISO-639 (<http://www.ics.uci.edu/pub/ietf/http/related/iso639.txt>)
- [J2SE] Java 2 Standard Edition. <http://java.sun.com/j2se/>
- [LIN] Linux Fedora Core 3 Distribution, <http://www.fedora.org>
- [LOG4J] The Log4j Project. Apache Software Foundation. <http://logging.apache.org/log4j/docs/>
- [MYSQL] MySQL Open Source Database. <http://www.mysql.com>
- [NSEQL] Denodo NSEQL 4.5 Manual. Denodo Technologies, 2008.
- [ORA] Oracle 9. <http://www.oracle.com>
- [PDFBOX] PDF document management Java Library, PDFBox. <http://www.pdfbox.org/>
- [POST] PostgreSQL Open Source Database. <http://postgresql.org>
- [SUN] Sun Microsystems. <http://java.sun.com>
- [TOM] Jakarta Tomcat 4.x.x servlet and JSP container, <http://jakarta.apache.org/tomcat>
- [VQL] Denodo Virtual DataPort 4.5 Advanced VQL Guide. Denodo Technologies, 2008.
- [WIND] Microsoft Windows Operating Systems, <http://www.microsoft.com>