# Roxen™ Administrator Manual

roxen Platform

www.roxen.com

# Introduction

This manual describes how to install, configure and maintain a Roxen Challenger server in a secure way. It is meant for system administrators and anyone interested in the inner workings of Challenger. The manual consists of the following chapters:

**Introduction** This chapter, introducing the concepts.

**Installing** How to install Roxen Platform or Challenger on a Windows or Unix system.

**Handling Challenger** Using Challenger's configuration interface.

**Virtual Servers** Creating virtual servers using HTTP, HTTPS or FTP.

**Configuration examples** Example configurations for some common server types.

**Modules** How Challenger's modules work and how to install and configure them.

**File systems** Modules that handle files and directories.

**RXML tags** Modules that handle RXML tags.

**Graphics** Modules that provide graphical capabilities to Challenger.

**Proxy** Modules for running Challenger as a proxy server.

**Miscellaneous modules** Modules that provide special functionality.

**Access Control** The access control system in SiteBuilder.

**Security considerations** How to avoid potential security hazards.

**Scripting** Using Challenger for your own scripting.

**Databases** Modules for communicating with SQL databases.

**SiteBuilder** Configuring the file handling in SiteBuilder.

**IntraSeek** Setting up and configuring the IntraSeek search engine.

**LogView** Setting up and configuring LogView.

**FrontPage** Installing and using MicroSoft FrontPage server extensions.

**Upgrading** Upgrading to a newer version.

**Third party extensions** Using modules and scripts that are distributed by third parties.

**Portability** Running Challenger and scripts based on Challenger on different operating systems.

**Reporting bugs** Identifying and report bugs.

**Appendix** Available modules.
HTTP response codes.
Migrating to Roxen.

## Server administration

The main task of the administrator is to provide the users with the services they need, while keeping the server functional and secure. A public web site is by definition reachable by any user on the internet, some of whom may harbor ill intent. Not only the web server itself but also all scripts and custom-made modules must be secure. With the most powerful Challenger modules, users can create pages with scripting and database accesses, pages that in themselves can be potential security hazards.

The administrator has to know about these security hazards and draw the line between functionality and security. The administrator can limit who gets to use which modules, in case not all users can be trusted with the security of the server.

The administrator also has to make sure the system is running smoothly. The main tools are the log files and event log. Most problems will be reported there.

The world of Challenger and the web are not static. New releases and third party software appears, and users demand more functionality. It is the administrator who has to determine if new software can be safely installed.

Besides knowing the Challenger server, the administrator needs to be familiar with Internet security and security issues related to the operating system the server is running on. It will not help that the Challenger server is made secure if the underlying operative system is not.

## Different versions

This manual covers Roxen Challenger versions 1.3. It uses Pike 0.6 and is available for several different Unix flavors as well as Windows. The slight differences between the Unix and Windows versions are documented in the Portability chapter.

Roxen Challenger 1.3 is available with support for encryption with the HTTP protocol. Because of export control restrictions of cryptography Challenger is available with support for weak, 40-bit, encryption and with support for strong, 128/168-bit, encryption. Apart from the encryption support the different versions are identical. You can find out which version you have by reading the first page of the configuration interface.

# Concepts

## Modules

Challenger is extremely modular. Every aspect and function in Challenger is handled by a module. Even such a simple task as reading a file from the file system needs at least one module. This enables administrators to customize their servers to almost any extent, by choosing the right modules. Challenger can be anything from a simple HTTP server, a proxy server to a FTP server.

All modules are not equal. Most modules provide some simple functionality, while others provide complex functionality that have security implications. The administrator has to take this into account before deploying such modules. This manual contains warnings about modules that have security implications, and a security chapter that all administrators should read thoroughly.

## Virtual file system

Challenger uses a *virtual file system* to determine which module will be used to handle a certain request. Modules are *mounted* on the virtual file system much like how file systems are mounted on UNIX. If a module is mounted on `/foo/bar/` it will be used to handle a request to `http://my.site/foo/bar/req?17`.

One special property of Challenger's virtual file system is that several modules may be mounted on overloading mount points. If one module is mounted on `/foo/` and another on `/foo/bar/` both modules are configured to handle a request to `http://my.site/foo/bar/req?17`. The priority of the modules determines which module will be used. In case the first module cannot handle the request it will be passed on to the second module.

Overloading file systems makes it possible to configure Challenger so that what is seen on the web can appear very different from how it appears on the file system. What appears as a single directory on the web might in reality be the union of several directories and databases.

## Protocols

Different protocols are used to find information resources on the Internet. Today HTTP and HTTPS are the most popular ones. Yesterday FTP was the most used protocol. Tomorrow, another set of protocols might be more popular. Challenger's modularity makes it possible to handle any number of protocols by creating new protocol modules. It is even possible to let one web site be reachable by several protocols. Thus any web site or applications built with Challenger can be accessed through new protocols, with no changes to the web site or application.

HTTP stands for Hyper Text Transfer Protocol and is the simple protocol used by web servers. HTTPS is HTTP over Secure Socket Layer, SSL. It is in like HTTP, but also supports encryption, thus making it much harder to eavesdrop on the communication. Encryption is necessary for transferring sensitive data over the Internet.

Apart from HTTP and HTTPS, Challenger also handles FTP, the File Transfer Protocol, an older protocol that was popular before HTTP. FTP is still in use today for downloading programs and uploading content to web sites.

As far as Challenger is concerned, there is essentially no difference between a HTTP request and a FTP request, so any feature available through HTTP is also available through FTP. This makes it quite possible, for instance, to create a database driven dynamic FTP site.

# Installing

## Installing

Challenger is written in the Pike programming language and requires Pike to run, so when you install Challenger, Pike will also be installed. By default, Pike will be installed inside Challenger's directory structure. If you want to use Pike for other tasks as well you can install it elsewhere.

Challenger contains its own fairly complex directory structure. It is helpful for administrators to be acquainted with this structure, which is essentially the same as the one stored in the distribution package.

Challenger is started by the shell script `server/start`. For an automatic restart of Challenger when the system is rebooted, this script should be called during the boot procedure of the operating system.

Challenger consists of two processes, the *start script* and a Pike process. The *start script* will be running to ensure a restart of Challenger in case the Pike process is killed or the user chooses to restart Challenger.

This chapter contains examples showing how to configure a Challenger server for some specific purposes. We have chosen to describe the following types.

**Standard Server** Generic web server, perhaps for the company home pages, with some useful modules added.

**Secure Server** Secure web server, using HTTPS to ensure no one can eavesdrop on the communication.

**User's Home-pages** Server for providing the user's home pages, giving each user her own domain using IP-less virtual hosting.

**FTP Server** Classic FTP server.

**Proxy Server** HTTP, SSL and FTP proxy server.

## Windows

The Windows version is distributed as a self-extracting `.EXE` file, containing a graphical installation program.

Run the self-extracting archive file and follow the instructions given by the installation wizard. You will be prompted to select where you want to store Challenger.

You can choose to install Challenger as a service, which is probably what you want if you intend to use it as your web server. You can also choose to launch Challenger directly after installing.

### Starting Challenger

If you installed Challenger as a service it should start itself at boot-time. Otherwise you start it with the appropriate menu. By default Challenger will use the configuration port 22202, so connecting to http://localhost:22202/ should take you to the configuration interface.

You can also start Challenger by double-clicking on the `ntroxenloader.pike` program in the install directory.

### Running Challenger as a Windows NT Service

You can install Challenger as a service at a later time by running `server\bin\roxen_service.exe -install`. You can remove it with `server\bin\roxen_service.exe -remove`.

When Challenger is running as a service you start or stop it with the *Services* control panel. Or you can run `net start roxen_service` or `net stop roxen_service`.

### Uninstalling Challenger

Challenger is uninstalled like any program with the *Add/Remove Programs* control panel.

## Unix

To install the Unix version of Challenger, you will need the GNU `tar` or `gunzip/gzip` program, as Challenger is normally distributed as a `tar.gz` archive. To install the source package on Unix you will need an ANSI C compiler. It is currently not possible to install the source distribution from scratch on Windows, since the compilation process is quite complex.

### Binary distribution

**1** `cd`
to the directory where you want Challenger to be installed and place the archive there.

**2** Issue the following commands to unpack the archive:
If you have GNU tar:

tar xzf *Roxen_1.3*.tar.gz

If you don't have GNU tar:

gunzip *Roxen_1.3*.tar.gz
tar xf *Roxen_1.3*.tar

Now move on to the section *Finishing*.

### Source distribution

**1** `cd`
to the working directory where you want to compile Challenger.

**2** Issue the following commands to unpack the archive:
If you have GNU tar:

tar xzf *Roxen_1.3*.tar.gz

If you don't have GNU tar:

```
gunzip Roxen_1.3.tar.gz
tar xf Roxen_1.3.tar
```

**3** Then type:

`cd` *Roxen_1.3*

to change to the Challenger directory.

**4** To continue with the *default* installation, type:

```
make install
```

This will configure Challenger with default values, compile it and then install it in `/usr/local/roxen/`. If you are happy with this then you can move on to step 5.

If you prefer a *custom* installation, type:

`./configure --prefix=`*path to Challenger*

prefix defaults to `/usr/local/`, witch places Challenger in `/usr/local/roxen/`

This will take a while. When it's done, type:

```
make install
```

**5** The Pike interpreter and various helper programs will now be compiled and then installed.

You can later on move the `roxen` directory that was created when you typed `make install` to anywhere in your file system. Roxen does not keep any absolute paths.

## Finishing the installation

**1** Type:

```
cd roxen/server/
```

**2** Start the install script by typing `./install`

**3** Answer the questions and connect to the configuration interface URL.

## Installing Pike separately

For a binary installation, follow the steps for *Binary distribution* above to unpack the tar archive.

**1** `cd`
to the `roxen` directory.

**2** Type:

```
make install_pike
```

For a source installation, follow the steps for *Source distribution* for Challenger above, but type

```
make install_all
```

instead of just `make install`.
or type

```
make install_pike
```

if you have already installed Challenger.

# Starting the server

On Unix, Challenger is started by the `server/start` script. The script is normally invoked without options since most configurations are done via the configuration interface. But in special cases options might be needed.

## Making Challenger start automatically

The start script should be invoked by the system startup files so Challenger is started automatically in case the computer is restarted. How this is done depends on which operating system you use. Note that Challenger might get a different path and belong to different groups when started by the startup files as compared to when it is started manually. If you experience problems that Challenger or scripts run by Challenger has problems starting other program this is one common cause.

**o** BSD
Edit `/etc/rc.local`, and add
`(cd /`*path to Challenger*`/roxen/server/;./start)`

**o** SysV (e.g. Solaris, IRIX)
Copy `tools/init.d_roxen` to `/etc/init.d/roxen`.

```
cp tools/init.d_roxen /etc/init.d/roxen
```

Make a symlink in `/etc/rc3.d`

```
ln -s /etc/init.d/roxen /etc/rc3.d/S90roxen
```

**o** DigitalUnix, HP-UX
Copy `tools/init.d_roxen` to `/sbin/init.d/roxen`.

```
cp tools/init.d_roxen /sbin/init.d/roxen
```

Make a symlink in `/sbin/rc3.d`

```
ln -s /sbin/init.d/roxen /sbin/rc3.d/S90roxen
```

**o** Linux / RedHat
Copy `tools/init.d_roxen` to `/etc/rc.d/init.d/roxen`

```
cp tools/init.d_roxen /etc/rc.d/init.d/roxen
```

Make a symlink in `/etc/rc.d/rc3.d`

```
ln -s /etc/rc.d/init.d/roxen
/etc/rc.d/rc3.d/S90roxen
```

In most cases you have to edit the copy of `tools/init.d_roxen` and update `roxenhome` to your *path to Challenger*.

## Options to the start script

**--version** Get only version information. Will not start Challenger.

**--help** Get help. Will not start Challenger.

**--log-dir=dir** Change the log directory. Defaults to `logs/`, or rather `../logs/` since the current directory is `server/`.

**--config-dir=dir** Change the configuration directory. Defaults to `configuration/`, or rather `../configuration/` since the current directory is `server/`. Must be set to allow several Challenger servers to share the same files.

**--with-threads** Run with threads if they are available.

**--without-threads** Disable threads.

**--with-profile** Enable runtime profiling of requests. The profiling information can be accessed through the action *Development/Debug information for developers.*

**--with-file-profile** Enable runtime profiling of requests on a per file bases. The profiling information can be accessed through the action *Development/Debug information for developers.*

**--with-keep-alive** Enable keep alive for the HTTP protocol module.

**--once** Run the server once with the debug log sent to stderr.

**--gdb** Run the server in gdb.

**--program** *program* Start a different program with Challenger's Pike interpreter.

**--with-debug** Enable debug mode.

**--without-debug** Disable all debug modes.

**--with-fd-debug** Enable file descriptor debugging.

**--truss** (Solaris only). Run the server in truss showing all system calls.

**--pid-file=file** Store the process id of the server and the start script in this file. Defaults to /tmp/roxen_$UID

### Arguments passed to pike

**-D***symbol* Define the symbol symbol.

**-d***level* Set the runtime pike debug level. This only works if pike has been compiled with debug.

**-s***size* Set the stack size.

**-M** *path* Add the path to the pike module path.

**-I** *path* Add the path to the pike include path.

**-dt** Turn off tail recursion optimization.

**-t** Turn on pike level tracing.

**-t***level* Turn on more verbose pike tracing. This only works if pike has been compiled with debug.

### Environment Variables

**ROXEN_ARGS** The contents of this environment argument will be treated as default arguments. Same syntax as the options.

**ROXEN_CONFIGDIR** Same as --config-dir=`dir`.

**ROXEN_PID_FILE** Same as --pid-file=`file`.

**ROXEN_LANG** The default language for language related tags. Defaults to *en* for English.

## Directory structure

Challenger will always run with the `server/` directory as current directory. All relative paths will have this directory as their origin.

### Directories

`server/` contains the read-only program files. When Challenger is upgraded the server directory with all its contents will be replaced. The files in `server/` should be left alone, otherwise important customizations might be lost when upgrading.

`server/start` is the script used to start Challenger.

`server/install` is the script used to install a new Challenger.

`server/bin/` contains binaries.

`server/modules/` contains the modules for that release of Challenger. Do not install your own modules here, put them in `local/modules/` instead. The directories where Challenger looks for modules can be configured in *Global Variables/Module directories.*

`server/nfonts/` contains fonts for the graphics capabilities of Challenger. Do not install your own fonts here, put them in `local/fonts/` instead. The directories where Challenger looks for fonts can be configured in *Global Variables/Fonts....*

`server/rxml_packages/` contains RXML packages that are included in the distribution.

`local/` is reserved for locally installed modules, fonts etc. It will never be touched by an upgrade procedure.

`local/modules/` preferred location for your own or third party modules.

`local/fonts/` preferred location for your own fonts.

`local/rxml_packages/` preferred location for your own or third party RXML packages.

`logs/` is the location for log files, unless changed with the *log-dir* option for the start script or the *Global Variables/Log directory prefix* variable. Several Challenger servers can share the same log directory.

`logs/debug/` contains the debug logs, one log for each Challenger server named after the configuration directory used by the specific server. The debug logs are rotated at each restart. The current debug log will always be called `servername.1`, the last `servername.2` ... until `servername.5`. The debug messages can be sent to syslog instead by changing the *Global Variables/Logging method* variable.

`logs/<virtual server name>/Log` is the default name for the access log for that Virtual Server. What is logged and where it is logged can be changed in the *Server Variables/Logging..* variables.

`logs/<virtual server name>/Accessed.db` and `Accessed.names` are the access database files used by the `<accessed>` tag. Whether the `<accessed>` tag should be enabled and in that case where the database files should be stored can be configured in the *Main RXML parser* module.

`configurations/` is the default location for the configuration files, and can be changed by the *config-dir* option for the start script. **This directory must be unique for each Challenger server.** The configuration file format is undocumented and may be changed in future releases. A Challenger running on unix will reload its configuration files if it receives a **SIGHUP** signal.

`gtext_cache/` is the cache directory for the graphical text tag. It can be shared by several Challenger servers.

`platform/` contains the extra modules used by the full Roxen Platform. Like the `server/` directory, this will be replaced when doing an upgrade, so it should be considered read-only.

### Pike

`bin/` contains the binaries for Pike and hilfe, the interactive Pike front end. The Pike executable is also hard linked as roxen so that it will appear with that name when you use the `ps` command.

`include/pike/` contains C include files for making Pike modules.

`lib/pike/include/` contains include files available in Pike programs.

`lib/pike/modules/` contains Pike modules available in Pike programs. The `.so` files are modules written in C, the `.pmod` files are modules written in Pike. If the Pike interpreter is compiled statically, which is always the case for the Windows version, the C modules will be compiled into the Pike binary instead.

`lib/pike/master.pike` is the master class for pike that controls a lot of Pike's behavior. For example how Pike classes are loaded.

## Standard server

Generic web server, perhaps for the company home pages, with some useful modules added.

**1** Create a virtual server using the *Generic server* type.

**2** Add and configure some additional modules of your choice. Perhaps:

**Pike tag** This module can be used for advanced applications, allowing the use of pike code in the HTML/RXML pages. But be careful with the the security considerations.

**3** Configure the file system module.

**Filesystem** Search path: Fill in the location, in the computers file system, where the HTML pages that should be used for the site will be located.
Example: `/usr/www/company_name/`

**4** Configure the Server variables.

**o** Configure a port.
The default port for web servers using the HTTP protocol is 80 and this is probably the port you want to use for a public information server.

**5** Click *Save* to save the server configuration on disk. The server will also try opening the port. You will be asked if the chosen URL for the site is correct. If you want to use another port than 80, you probably need to alter this.

Note: Ports below 1024 can only be used if the server is run, or at least started, as root.

## Secure server

Secure web server, using HTTPS to ensure no one can eavesdrop on the communication. You must have one of the encrypted versions of Challenger, either the 40-bit version with weak encryption or the 128/168-bit version with strong encryption, to use HTTPS.

**1** Create a certificate.

You need a certificate that ensures that your web site really is your web site. If you already have a certificate use this. Otherwise create one with the *Security/Generate a RSA key and a self-signed certificate...* action. This will produce two files, a certificate file as well as a RSA key file. The later file must be kept secret.

**2** Create a virtual server using the "Generic server" type.

**3** Configure the file system module.

**Filesystem** Search path: Fill in the location, in the "normal" file system, where the HTML pages that should be used for the site will be located.
Example: `/usr/www/company_name/`

**4** Configure the Server variables.

**o** Configure a port.
The default port for web servers using the HTTPS protocol is 443. You also have to fill in the location of your certificate file as well as the RSA key file. server.

**5** Click *Save* to save the server configuration on disk. The server will also try opening the port. You will be asked if the chosen URL for the site is correct. If you want to use another port than 80, you probably need to alter this.

Note: Ports below 1024 can only be used if the server is run, or at least started, as root.

Your server will not be secure until you get a certificate signed by a Certificate Authority like VeriSign or Thawte. In order to get a signed certificate you first produce a Certificate Signing Request through the *Security/Generate a Certificate Signing Request for an RSA key...* action.

## User's home pages

Server for providing user's home pages, giving each user her own domain using IP-less virtual hosting.

**1** Create a virtual server using the *Generic server* type.

**2** Configure the Server variables for a virtual server that should handle the IP-less virtual hosting.

- **o** Configure a port, preferably 80 which is the standard HTTP port.

**3** Configure DNS

For each user to get her own domain these domains need to be configured in DNS. The easiest way to do this is to use a CNAME-post with a wild card, that way you don't have to configure each domain individually. Such a post could look like this:

`*.userCNAMEwww`

You need a late version of bind 8 for this to be supported.

**4** Configure the *User Filesystem* module.

To enable the virtual user hosting you need to set the *Virtual User Hosting* variable. You might also need to configure the *Public directory* variable, to the directory in the user's home directory that should contain their web pages.

**5** Configure the *User database and security* module.

The *User database and security* module communicates with the operating systems user database in order to find and authenticate users.

**6** Click *Save* to save the server configuration on disk. The server will also try opening the port. You will be asked if the chosen URL for the site is correct. If you want to use another port than 80, you probably need to alter this.

Note: Ports below 1024 can only be used if the server is run, or at least started, as root.

## FTP server

Classic FTP server.

**1** Create a virtual server using the "FTP server" type.

**2** Configure the modules.

- **o** Filesystem - Search path: Fill in the location, in the "normal" file system, where the files should be stored.
  Example: `/usr/ftp/`

**3** Configure the Server variables.

- **o** Configure a port. The default port when using the FTP protocol is 21.

**4** Click *Save* to save the server configuration on disk. The server will also try opening the port. You will be asked if the chosen URL for the site is correct. If you want to use another port than 80, you probably need to alter this.
Note: Ports below 1024 can only be used if the server is run, or at least started, as root.

## Proxy server

HTTP, SSL and FTP proxy server.

**1** Create a new virtual server using the *Proxy server* type.
The following modules should be added:

- **o** FTP proxy
- **o** HTTP proxy
- **o** SSL proxy

**2** Configure the proxy modules.

You do usually not want outsiders to be able to connect to your proxy server, since they might be able to use the proxy server to reach your internal network. This is especially important for the *SSL proxy* module. Therefore you should add a security pattern to each module, by changing the *Builtin variables/Proxy security: Patterns* variable. A suitable value is `allow ip=194.52.182.*` with 194.52.182.* replaced with your network.

**3** Configure the Server variables.

- **o** Configure a port. Since all requests to the server will be sent as http, 80 is the obvious choice, but any other port will do.

**4** Click *Save* to save the changes to disk. The server will also try and open the chosen port. If the port opens successfully, the proxy server will now be operational.

Note: Ports below 1024 can only be used if the server is run, or at least started, as root.

# Handling

Challenger's configuration interface is accessed through its own port and protocol. The user will be prompted for a user name and password before gaining access to it.

When you install the server with the `install` script you get to choose which port and protocol the configuration interface should use.

The settings for the configuration interface can be changed in *Global Variables/Configuration interface...*

### Configuration port

The configuration interface will usually run on a non-standard port. If you need to access it from outside your local network you might have to reconfigure your firewall to allow accesses to that port. If you have forgotten the port you can find it in the debug log, usually `logs/debug/default.1`.

### Server protocol

You can choose to use either the HTTP or HTTPS protocol for the configuration interface. The default and recommended is HTTPS which gives an encrypted connection. Using HTTP is a potential security hazard, since the administrator's user name and password will be sent in clear text. To be able to use HTTPS you must have the full version of Challenger, with SSL and encryption.

### Certificate

When you install the configuration server with the HTTPS protocol, it will use a test certificate. This is not secure, since the certificate is the same in all Roxen Challenger distributions. You should replace this with a self signed-certificate as soon as possible, and consider getting a certificate from a certificate authority. The HTTPS page explains how to create certificates.

### User name and password

When you connect to the configuration server for the first time you will be asked to set a user name and a password. This will become the user name and password required to access the configuration interface. They can be changed by the *Security/Change password and/or username...* action.

## User interface

### Tabs

The Challenger configuration interface is divided into *Virtual Servers*, *Global Variables*, *Event Log* and *Actions*. The parts are divided by tabs, selecting a tab will take you to the corresponding part.

**Virtual servers** This is where the separate virtual servers are configured. The virtual servers chapter deals thoroughly with the concept of virtual servers.

**Global Variables** This is where options global to the whole Challenger server are configured, for example the port of the configuration interface.

**Event Log** The event log shows important events. These events are also sent to the debug log. The logs should be viewed regularly, to make sure the server is operating smoothly.

**Actions** Contains a number of tools for getting status about the server and performing maintenance.

**Manual** Where this manual is found.

### Folding/unfolding

The configuration interface is based on fold/unfold arrows. If you click on one of the arrows pointing right it will change into a down-pointing arrow and all the underlying items with their associated variables will become visible; they *unfold*. Clicking on a down-pointing arrow will fold the arrow again.


*Folding/Unfolding arrows*

### Focusing

It is also possible to click on the main menu items themselves, not only unfold them. Doing this is called *focusing*. Sometimes focusing is necessary to access certain functions and it also gives you a clearer overview of the variables residing under each item. We consider every clickable item a *node*.

When you have focused on an item, you will see an up pointing arrow to the left of the item. This item will now be the top-most visible header.


*Focusing on fonts*

Clicking on the up-pointing arrow will get you back one level in the configuration interface.

Clicking on the active tab, *Global Variables* in this case, returns you to an *unfocused* state. Note that the status row in the upper right corner always indicates where in the configuration interface you are.

## Color encoding

The configuration interface uses color encoding. As soon as you have made some changes you will notice that the blue-colored arrows turn red. The red arrows are meant to draw your attention to where there are unsaved changes. In order to make your changes permanent you will have to click on the *Save* button at the bottom of the page. The arrows will then turn blue again.

## Buttons

There are usually one or more buttons at the bottom of the page. On some nodes more buttons will appear to reflect what can be changed at that node.


*Configuration interface buttons*

**Save** Save changes of the configuration to disk. Changes made will take effect immediately, but they will not be saved on disk until you click this button. Restarting the server before saving to disk will undo the changes.

The *Save* button will only appear when there are unsaved changes.

**New virtual server** Create a new virtual server. The *New virtual server* button appears on the top node of the *Virtual Servers* tab.

**Delete this server** Delete this virtual server. The *Delete this server* button appears on the virtual server nodes.

**Add module** Add a module to this virtual server. The *Add module* button appears on the virtual server nodes.

**Copy module** Add another copy of this module to the current virtual server. This option will only be available for modules that may have more than one copy run simultaneously. The *Copy module* button appears on the module nodes.

**Reload module** Reload this module. The module will be recompiled and then reloaded. For developing or upgrading modules. The *Reload module* button appears on the module nodes. This option is only available if *More options* has been selected.

**Delete module** Delete this module. The *Delete module* button appears on the module nodes.

**More options/Fewer options** The configuration interface allows you to choose between fewer and more options respectively. By default set to fewer options. More options can be chosen by simply pressing the *More options* button at the bottom of the page.

**Fold all** When you have made several changes or descended deep into the configuration hierarchy you can fold all the sub menus by clicking on the *Fold All* button at the bottom of the page.

**Unfold level** The *Unfold level* button unfolds all sub menus on this level.

**Unfold modified** The *Unfold modified* unfolds all nodes with unsaved changes.

**Clear module caches** This will clear the module caches.

# Restarting

The server uses a process which runs for a long time, sometimes months, while modules are installed and reloaded. Challenger is designed for this, but during months of operation many things can happen. If the server starts using more resources than normal, or responses are slower, it might be a good idea to restart it.

Restarting can be done either with the *Shutdown/Shut down Roxen* action or by killing its pike process. The start script will ensure that the server is restarted.

Restarting the server takes from about 10 seconds up to a few minutes depending on server configuration. This is intentional, as Challenger still needs to finish serving the requests which were active when the shutdown procedure was initiated. The only noticeable effect when restarting should be that the server will not respond during startup.

## Automatically restarting Challenger

It is possible to schedule restarts at regular intervals, setting *Global Variables/Automatic Restart...* The variable specifies the number of days between automatic restarts. Scheduling an automatic restart every 30th day might be a good idea.

## Anti Block System

The fact that Challenger consists of only one process can sometimes be a drawback. Should the process hang or enter an endless loop, the whole site will be down. To recover from such situations, Challenger offers an ABS, or Anti Block System, to be configured under *Global Variables/Anti-Block-System.* The ABS will check if the server responds within a certain amount of time and else restart the server automatically. You can set the timeout the ABS will use before checking if the server is still up, by changing the *Global Variables/Anti-Block-System/Timeout* variable. If this timeout is too fast the ABS may assume that your server is hanged while it just slow to respond. By examining old debug logs it is possible to see if the ABS has unnecessarily restarted the server.

## PID file

The process ID number of the pike process and the start script will be written to a file, by default `/tmp/roxen_pid:$uid`. It can be configured through *Global Variables/PID file.* The PID file can be used by an external program or script to restart the server.

## Threads

Challenger can use threads on operating systems that support them. When using threads several requests can be handled in parallel. The main advantage of this is that requests that take a long time to process will not block the server, and other requests will be served even if a previous request is still being processed. It is therefore recommended that you run with threads on all operating systems that supports it.

However, the speed gained by running with threads is usually limited. Most requests to a Challenger server take a very short time to handle, and such requests gain next to nothing by running in parallel. One exception is pages served over NFS from another computer. Fetching data over NFS is a slow, blocking operation, thereby benefiting from being able to run in parallel.

On multi-CPU machines the worst case scenario is that the overhead for thread synchronization makes the performance worse when using threads than when not using threads.

The number of threads Challenger will use is configured in the *Global Variables/Number of threads to run* variable. To disable threads altogether use the `--without-threads` option to the start -script.

Note that even if Challenger only uses one thread, modules can create threads for themselves, if they intend to do something that will take a long time.

## Log files

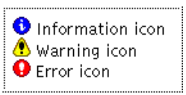
Challenger logs unusual events in two different places: the *Event Log* tab in the configuration interface, and a debug log file. It is a good idea to keep an eye on at least one of these regularly, since important events are logged there.

Challenger also logs all accesses to a virtual server in an access log.

### Event Log

The *Event Log* will display a list of events that have occurred such as when the server was started, which virtual servers were enabled and when the administrator logged in. The list also displays errors such as run-time errors or failure to compile a module. Error messages will if possible include a Pike backtrace, making it easy to pin-point the cause.

Each event is preceded by an icon that indicates whether this is an information, warning or error message.




*Icons*

### Debug Log

The debug log contains the same information as the Event Log. Its default location is `logs/debug/servername.1` but this can be changed by the *log-dir* option to the start script or the *Global Variables/Log directory prefix* variable. The debug log is rotated at each server restart, the five most recent debug logs being kept.

The debug log can be sent to syslog by changing the *Global Variables/Logging method* variable.

### Access Log

Challenger logs each access to a virtual server in an access log. Its default location is logs/<virtual server name>/Log and can be changed by the *log-dir* option to the start script, the *Global Variables/Log directory prefix* variable or the *Server Variables/Logging...* variable.

It is important to keep the access log under surveillance, since it will show exactly how the server is used. This will tell which pages are popular as any pages that cause errors. Attempts to break into the server will also often show up in the access log. Since the access log contains so much data a log analyzing tool, like Roxen LogView, is the preferred way of studying it.

Because of the access log size, you will probably have to rotate and compress them once in a while. Challenger cannot do this by itself, you have to do it manually or with an external script. Just move the log file and Challenger will, within a minute, start using a new log file.

The log function can be configured in the *Server Variables/Logging...* variables. The default is the Common Logfile format, which is understood by all log file analyzers.

How things are logged can also be changed by enabling a log module. The *User logger* module gives access logs for each user's home pages. Other log modules are the *Client logger* and *Logging disabler*.

## Status

There are a number of ways to check the status of a Challenger server.

### Logs

Regular surveillance of the logs is of course important to detect unusual events and errors.

### CPU and Memory Usage

Keeping an eye on CPU and memory usage of Challenger's Pike process is always good. A major increase of CPU or memory usage is often worth investigating.

On a Unix system, the commands `ps` or `top` can be helpful. It is, however, worth noticing that the memory usage reported by those commands is not always entirely relevant, since some of it can be caused by memory mapped files.

Challenger's own idea of how much memory it uses can be found under the *Development/Debug information for developers* action (it will appear only if *More options* has been selected).

### Status and Debug Info

For every virtual server there is a page in the configuration interface, *Status and debug information*, showing how busy that virtual server is.

### Access Requests

The number of requests for the server is showed in the *Status/Access request status* action. If the server has been unusually busy, the access logs for the virtual servers will tell why. This is best done with a log analyzer tool, such as Roxen LogView.

The *Status* actions show information about:

**Access / request status** Shows the amount of data handled since last restart.

**Current FTP sessions** Lists all active FTP sessions and what files they are currently transferring.

**Extended process status** Shows detailed process status on Solaris and Linux.

**List Available Fonts...** Lists all available fonts.

**Open files** Shows a list of all open files.

**Pipe system status** Shows the number of data shuffling channels.

**Process status** Shows various information about the pike process.

**Thread status** Shows various information about the threads in Challenger.

### Caches

The *Cache/Cache status* action shows how well the built in caches in Challenger are working.

## Maintenance

Challenger contains a number of tools for maintenance tasks. Most of them are found on the *Actions* tab. These tools are mostly useful if the Challenger server is used for development purposes.

### Restarting

The most common remedy when unusual things happen is to restart the server with the *Shutdown/Shut down Roxen* action. This makes the server start fresh and reread all data from disk.

### Cache / Flush caches

The *Cache/Flush caches* action allows the administrator to empty Challenger's built-in caches.

### Maintenance

The *Maintenance* actions are tools for doing maintenance tasks.

**Check your Roxen configuration for problems ...** Performs several sanity checks of the configuration and proposes suitable adjustments when discovering problems.

**Clear Event Log** It is possible to clear all or specified types of events from the event log if it grows too large.

**Reload configuration from disk** Forces a reload of all configuration information from the configuration files. This works as an undo in case you have unsaved configuration changes you don't want to keep.

# Virtual servers

Each virtual server handles a web site, complete with modules and module configurations. To make a virtual server reachable by the outside world, you will have to configure its *Server Variables/Listen ports* variable to select which protocol, IP address and port number the virtual server are to use.

## Protocols

The protocols Challenger support are HTTP, HTTPS and FTP. HTTP, Hyper Text Transfer Protocol, is the basic protocol used by web servers. The default port for HTTP is 80.

HTTPS is HTTP over SSL, the Secure Socket Layer, which provides encrypted communication. The default port for HTTPS is 443.

FTP, File Transfer Protocol, is an older protocol that is still used for some purposes. The default port for FTP is 21.

## IP address and port

All communication over the Internet is channeled through IP addresses and ports. Each computer handles one or a few IP addresses. For each IP address, there are 65,535 available ports. Each protocol has its own default port, which is the preferred port for communication in that protocol.

Usually, it is possible to use any port by specifying it explicitly. Thus, the URL *http://www.roxen.com/* will use the default port, 80 in this case, while the URL *http://skuld.idonex.se:4711/* will use the port 4711.

## DNS

The basic IP addresses consist of numbers, for example *194.52.202.32*, that are not particularly easy to remember. Domain Name System, DNS, helps by suppling symbolic names, for example *www.roxen.com* that are translated into the IP addresses needed for communication.

The symbolic names are handled by name servers. For example, Idonex's name server handles domain *roxen.com*, among others. Configurations of new names under a domain are done by configuring the name server.

## IP-less HTTP

Many protocols make it necessary for each information service on the Internet to have its own IP address. Thus, each web site and FTP site would need one IP address. If one computer were to handle more than one information service it would have to handle more than one IP address.

In HTTP, the web site wanted is also sent through the protocol in the Host header. Thus, it is possible to make several web sites share the same IP address and port. When a virtual server in Challenger is configured for IP-less HTTP, it will have no port of its own. Instead it will rely on getting requests from another virtual server that does listen to the port.

Really old browsers such as Netscape 1.0, do not support the Host header, and therefore can not connect to web sites using IP-less HTTP. It is strongly recommended to upgrade such browsers.

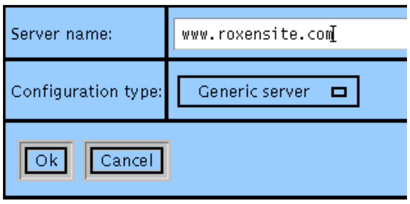
## Interfaces

Each IP address that a computer handles is bound to an interface. An interface is an ethernet card, a modem, or some other physical network hardware. A computer, or router, connected to more than one physical network needs one IP address per network interface.

It is also possible to configure *virtual interfaces* for the sole purpose of handling more IP addresses. How this is done varies between operating systems. If you want to handle more than one HTTPS or FTP server on each computer, or don't like IP-less HTTP, you will need to know how to set up additional virtual interfaces.

# Creating

A new virtual server is created by pressing the *New Virtual Server* button on the *Virtual Servers* tab. The first thing you will have to do is choose a name for your virtual server. The name will be used to identify it in the configuration interface.




*Creating a new virtual server*

## Configuration types

The next thing to choose is what kind of server you want to create. The configuration type determines which modules will be pre-installed in your new virtual server. There are a number of configuration options to choose from.

**Bare bones** This configuration creates a virtual server without any modules pre-installed. With this configuration type you get total control of your new server.

**Basic server** This configuration contains a few basic modules you need to get going.

**FTP server** This configuration provides you with the modules you need to run a FTP server. The configuration also provides a pre-configured FTP port.

**Proxy server** This configuration provides you with all of the proxy server modules you need to run Challenger as a proxy server.

**Generic server** This configuration contains the most common and popular modules used with Challenger. It is the default configuration.

**Manual viewing server** Configuration for viewing the on-line manuals, compelete with all modules necessary for running the interactive examples.

**SiteBuilder server** This configuration creates a complete SiteBuilder virtual server, with LogView and IntraSeek installed as well.

**SiteBuilder FTP edit** This configuration creates a FTP site for editing against a SiteBuilder module running in another virtual server.

**SiteBuilder view area** This configuration is suitable for making a view area of a SiteBuilder module running in another virtual server available.

## Configure Server Variables

Your virtual server will now be installed containing a number of modules. Some of these modules may require further configuration, please refer to the chapter about that specific module.

One thing that you will have to configure is how your virtual server is to be accessed by the outside world. If it is to be accessed through IP-less HTTP, you only need to fill in the *Server Variables/Server URL* variable. Please refer to the HTTP page for more information about IP-less HTTP.

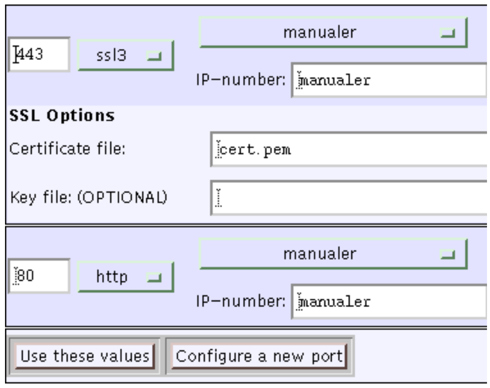
To use any other protocol you need to configure a port for it. This is done through the *Server Variables/Listen ports* variable. There you press the *Configure a new port* button.

**Port** Enter the port number you want your virtual server to bind to. The default port depends on the protocol used, 80 for HTTP, 443 for HTTPS or 21 for FTP.

**Protocol** Please select which protocol you want your virtual server to respond to. You can choose between the following protocols: FTP, HTTP, HTTPS or tetris, default is HTTP.

If you want to you can bind your virtual server to several ports and protocols. For example, you may want to configure a virtual server to handle both an HTTP and an HTTPS port. Or you might want to have both an HTTP and a FTP port, so users can upload pages through FTP.

**Host** Please select or enter which interface the port should bind to. The interface is identified either by its host name or IP address. The default is ANY, which means that the port will be bound to every interface and answer to every IP address the computer handles.




*Configuring listen ports*

Once you have made your choices press the *Use these values* button. When you later press the *save* button you will be prompted to enter the right *Server Variables/Server URL* variable as a sanity check.

The port might not always be opened properly. You can always find out if it has been opened by zooming in on your virtual server on the *Virtual Servers* tab. In case the port was not opened properly a restart of the server might fix the problem. If not it is probable that the server does not have the privilege to open the port, or some other process has already opened it. It is common that the default FTP port, 21, is already in use by the Unix FTP daemon and that the FTP daemon must be disabled in the `/etc/inetd.conf` file.

# HTTP

HTTP is a very fast and reliable transfer protocol. The default port for HTTP is 80.

Because HTTP transfers are made in clear text, it is not wise to use it to transfer passwords if the networks involved in the transfer are not secure.

### IP-less HTTP

It is possible to run several HTTP servers on the same IP address and port, by identifying the servers through the Host header in the HTTP protocol. Since IP addresses are becoming a scarce commodity IP-less virtual servers are becoming increasingly common.

To use IP-less virtual servers you need to configure DNS so that several different names translate to the same IP address. One of your virtual servers needs to listen to this IP address. That virtual server must also contain the *IP-less Virtual Hosting* module.

For the other virtual servers you will only need to configure their *Server Variables/Server URL*. The *IP-less virtual hosting* will use the host header and the virtual servers *Server Variables/Server URL* to determine which virtual servers are to handle which request. The *User Filesystem* module can also be used to host several sites with IP-less HTTP. It does this within one virtual server, which means that all sites share the same modules. This sharing is very resource inexpensive and makes it possible to give each user her own domain. IP-less HTTP does not work for proxy servers. That is, a proxy server must have its own IP address. A proxy server can however, proxy requests to web servers that are using IP-less HTTP.

### IP-less virtual hosting module

The IP-less virtual hosting module does not need to be configured in any way. Just add it to a virtual server that is listening to a port, and it will send requests to other virtual servers.

## HTTPS

HTTPS is an encrypted version of HTTP implemented through the Secure Socket Layer, SSL, standard. The encrypted secure connection is created by running an ordinary HTTP connection on top of an encrypted SSL connection. Except for this, HTTPS is like HTTP.

### Creating a HTTPS port

To set up a HTTPS port you enter the protocol, HTTPS, and port number, by default 443, as you would for an HTTP port. When you press the *Use these values* button you will get two new options, *Certificate file* and *Key file.* If your certificate also contains your private RSA key you only need to fill in the *Certificate file* option. Otherwise you will have to fill in both options. The demo certificate included with Challenger contains the private RSA key, while the certificates you get from a Certificate Authority does not. It is not possible to run HTTPS IP-less. This is because the certificate contains the name of the web site and the certificate is used before the server gets a chance to see the Host header of the HTTP protocol.

### Certificates

One fundamental property of secure communication is that you must be certain whom you are communicating with. On the Internet you use DNS to find a web server, but DNS is not secure. Therefore, you need to be able to check that DNS really connected you to the right web server.

This is done through certificates. A certificate is digitally signed by a Certificate Authority, and contains information about the web server. The browser can check that the information and the digital signature are correct, as long as it knows about the Certificate Authority that has issued the certificate.

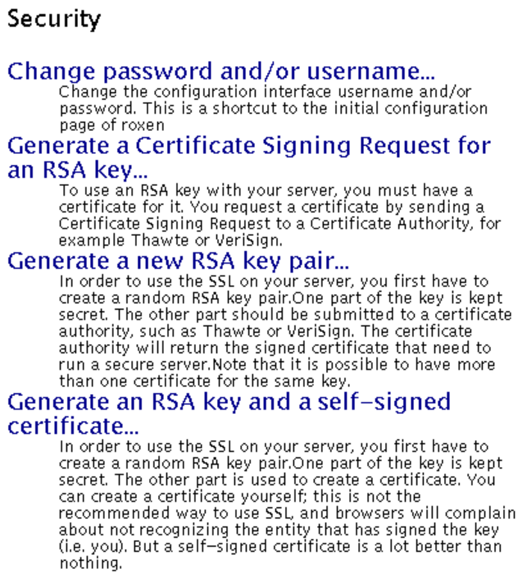
In order to get a real certificate, you must first create a certificate signing request and then send that request to a Certificate Authority, who will then check that you are whom you claim to be, and then give you a certificate that they have digitally signed.

It is also possible to create your own self-signed certificates. This goes against the whole idea of certificates, since they don't really provide any added security. Therefore browsers will show a warning dialog when they encounter a web site with a self-signed certificate. But the self-signed certificate might do until you get a real certificate.

### Generating an RSA key pair

The certificates and digital signatures used are based upon public key cryptography and the RSA algorithm. It essentially works by creating a key pair, where one key can decrypt what the other key has encrypted. One key in the pair becomes your public key that you give to the world, the other key becomes your private key, which you keep secret. This last thing, to key your private key secret, is important, for anyone who has your private key can decrypt the traffic from your web site, even though you are using HTTPS. You should consider running your secure web server on a machine with few users and high security, to make it hard for others to get hold of the private key. If you want many users to be able to modify the web site, it is advisable to consider running an extra, separate server for doing the HTTPS part. That server could then use the *HTTP Relay* module to relay requests to the normal web server.

You can generate an RSA key pair with the *Security/Generate a new RSA key pair...* action. You should keep on to the key pair since you will need it later, after you get your signed certificate.




*Requesting and generating certificates*

### Creating a certificate signing request

The next step is to create a certificate signing request. This is done through the *Security/Generate a Certificate Signing Request for an RSA key...*. You will need an RSA key pair to run this action. This action will prompt you for information about your organization and your web site. When you have filled in all information it will generate a standard certificate signing request that you can send to a Certificate Authority, such as VeriSign or Thawte.

### Creating a self-signed certificate

You create a self-signed certificate with the action *Security/ Generate a RSA key and a self-signed certificate...*. It creates a RSA key file as well as a certificate file.

### Demo certificate

Included with Challenger is a demo certificate, `server/demo_certificate.pem`. It will be used by the installation script when installing a Challenger server which uses HTTPS for its configuration port. It should however be changed as soon as possible to a self-signed certificate, or even better a real certificate. Since anybody that downloads Roxen Challenger gets the same certificate, it is possible to eavesdrop on the encrypted traffic if the demo certificate is used.

## FTP

FTP is an older protocol for accessing information resources that was the primary standard before HTTP. It provides a structure and interface that resembles a file system more closely than HTTP does. This makes it better for uploading information to a web site, since it is always possible to get directory listings through FTP. FTP is still often used for downloading programs.

FTP cannot be run encrypted. It is however possible to use the port forwarding features of SSH to run FTP encrypted, if SSH is installed both in the client and server. FTP uses two network connections, it is quite easy to make the command channel encrypted while it is hard to ensure that the data channel is encrypted. Since all passwords are sent through the command channel it is often enough to encrypt it.

### Creating an FTP Port

You create a FTP port by choosing the FTP protocol and a port number, the default is 21. After that, the following server variables can be used to control the behavior of the FTP service:

**Allow anonymous FTP** Whether to allow anonymous ftp users or not. If allowed it will be possible to log on as *anonymous* or *ftp* with any password, or get information with no user authenticating at all.

**Allow FTP guest users** If allowed the FTP protocol module will allow accesses by users who were not authenticated correctly, according to the current user database. The main reason for this option is that the whole web site might not use the same authentication system. Users may, for instance, use *.htaccess* files with their own user and password database to control the access to their own pages.

When using this option users will not know right away if they typed in their password correctly. Any password will do, since the actual authentication will not take place until they try to access restricted pages. This is more in line with how HTTP works than how FTP usually works.

**Allow named FTP** If set, the FTP protocol module will allow authenticated users to log on.

**FTP user session limit** Limits the number of simultaneous FTP sessions a user can have. By default 0, which means an unlimited number of sessions.

**Shell database** If set, the FTP protocol module will use the shell database together with the user database to determine whether the user should have FTP access. This way, Challenger becomes more compatible with the standard Unix FTP daemons. Entering an empty value will disable the check.

### Using FTP

For Challenger there is no difference between a FTP and a HTTP request. When used for uploading, an FTP upload will be equivalent to doing an HTTP PUT. Whether uploading will be allowed or not is up to the file system module used. The normal *Filesystem* module contains an option *Handle the PUT method* which controls if it will be possible to upload files or not.

## Tetris

It is always nice to relax with a good game of Tetris. To configure a Tetris server, choose a *Bare bones* server to begin with. Then choose *tetris* as the protocol. To play a game, use a Telnet connection like: `telnet »host« »port«`

### Controls

You control tetris by using these keys:
left: j
rotate: k
right: l
move down: SPACE
pause: p
quit: q

## Logs

Challenger provides access log files in the Common Log Format for each virtual server. It is possible to configure exactly what should be logged and how by the *Server Variables/Logging...* variables.

Logging can also be done through special logging modules. A few such modules are included with Challenger.

### User logger

Logs accesses to a user's home pages in a separate access log. Each user that wants such an access log must create an `AccessLog` file, in her home page directory.

**AccessLog filename** The filename of the access log file, by default `AccessLog`.

**Logging Format** Configuration of how things should be logged.

**Only log in userlog** If set, no entry will be written in the normal log.

**Private logs** The directories that are to get their own log files. Either use a specified PATH or a pattern. `/foo/` will use `/foo/AccessLog`, `/users/%s/` will use an `AccessLog` in all user directories, providing that the users directories are mounted on `/users/`. All PATH's are relative to Challenger's virtual file system.

### Logging disabler

This module can be used to turn off logging of files, specified with a regexp.

**Logging for** All files whose PATH in the virtual file system match these patterns will be logged, unless they match any of the *No logging for* patterns.

**No logging for** All files whose PATH in the virtual file system match these patterns will be excluded from logging.

### Client logger

This module simply logs the *user-agent* field in a log file. Normally, this field is not logged in access logs, though it is possible to configure Challenger to log it there.

**Client log file** The file to log to.

## Messages

### Resource not found

The *Server Variables/Messages.../No such file* variable contains the error page that will be returned if Challenger could not find a page. It can be customized for each web site to contain information guiding the user to the information she most likely wanted.

### FTP Welcome

The *Server Variables/Messages.../FTP Welcome* variable contains the welcome message that will be shown when you connect to an FTP port.

### For Developers

The *Global Variables/Show the internals* variable control how detailed the *Internal server error* message should be. This is the message Challenger gives when an error occurred while the request was processed. If set to *yes* a Pike backtrace will be shown, something which greatly facilitates debugging for a developer. On the other hand, the message might seem a bit confusing or intimidating to a regular user, the option is mainly intended to for use on development sites.

# Modules

A module is an addition to a virtual server, adding to or modifying the server's functionality in some manner. The module is made up of a Pike object that is run inside Challenger. Each module can be configured through the configuration interface.

Which modules are enabled and how they are configured determines how the web site will behave, or even if it should be a web site at all. If only proxy modules are enabled the virtual server will not host a web site but rather be a proxy server.

Modules come in different flavors, or types. Each module belongs to one or more types. Some types, like the *Authentication* and *Directory* types, are special and you can only enable one per virtual server. It is on the other hand possible to enable any number or modules of more common types, like the *Location* or *Parser* types. Some modules may themselves be installed more than once, but that is a module dependent property not something determined by the module type.

The module type determines what services the module provides. A *Location* module will provide files, either from a real file system, from a database or from some other source. A *Parser* module provides new RXML tags, that can be used like HTML tags.

The module types are designed so that modules can cooperate. The idea is that each module should provide a basic functionality, that can be combined by the functionality provided by other modules. That way the administrator can tailor the configuration of her web site to her needs, by choosing the right set of modules.

## Installing modules

In order to add a module to a virtual server press the *Virtual Servers* tab and then focus on the name of the server. As soon as you have done this you will see the *Add module* button at the bottom of the page. Pressing the *Add module* button will display a list of all available modules. A module is selected by pressing the module name header. Below each module header is a brief explanation of the module. Each module adds its configurable parameters under the *Virtual servers* tab.

Some modules require other modules to work. When installing such a module the modules it requires will also be installed. It will not be possible to delete the supporting modules without first deleting the module that requires them.

### Configuring the module path

Challenger will search for modules in the directories configured in the *Global Variables/Module directories* variable. It doesn't matter where you store your modules as long as the *path* to their directory is configured in this variable.

On each module's node you can see where that particular module resides.

You will have to do a reload on the *Add module* page before any new modules will be shown.

### Upgrading a module

To upgrade a module you must first replace the files for the old version with the files for the new version. Then you focus on the module's node and press the *Reload Module* button. If there are any problems with compiling the new version they will show up here. If so, you can always move the files for the old version back and no harm will be done.

### Compilation errors

If a module gets a compilation error this will be entered into the *Event Log* and the debug log.

## Configuring modules

How a module can be configured depends on what module it is. Each module contains its own set of configuration options. However, there are some options that are available for all modules.

### Status and debug info

*Status and debug info* shows specific information about the usage of that module. This option exists for modules that provide this extra information.

### Builtin variables

The builtin variables are available for all modules, but the number of builtin variables vary dependent on the type of the module. The possible builtin variables are:

**Comment** The *Comment* variable makes it possible to write a comment about this module, that can be seen by anyone having access to the configuration interface.

**Module name** The *Module name* variable makes it possible to rename the module in the configuration interface.

**Priority** The priority of a module determines which module gets to handle a request, when there are several modules of the same type enabled. The module with the highest priority gets to try first. In case that module failed to handle the request, the module with the next highest priority gets to try. This continues until a module is found that can handle the request, or all modules of that type has been tried.

If two modules have the same priority it is not defined who gets to handle the request.

**Security: Patterns** The *Security: Patterns* variable determines who gets to access this module. It is possible to limit access to certain computers, networks or users. See the Security considerations chapter for more information.

**Security: Realm** In case access to the module is limited to certain users this variable will be used when asking for the user name and password. It will usually show up on the password dialog shown by the web browser.

**Security: Security level** The *Security: Security level* variable determines which other module may be part of serving the request. This is usually usable in case you only trust some users to use certain modules, for example the *Pike tag* module. By giving the *Filesystem* module serving pages from untrusted users a lower security level than the *Pike tag* module it will not be possible for them to write pages that use the `<pike>` tag.

More information about security level can be found on the trust levels page of the security considerations chapter.

## Module types

Almost all functionality in Challenger exists in different modules. Each module has a distinct task. Several modules can cooperate in the creation of a page that will be sent to the user.

What task a module has is determined by its type. More complex modules can be of several types and thus perform more than one task.

**Authentication** modules handles authentication of, and information about, users. The most common type of Authentication modules are modules that import the user database from the operating system Challenger is running on. The information provided by an Authentication module is often used by other modules, such as the *User Filesystem* module. It is only possible to have one Authentication module per virtual server.

**Directory** modules deals with directory listings and index files. If the requested resource is a directory, a directory module either tries to find a suitable index file or to create a page with a directory listing. It is necessary to have a Directory module in order to get index files, such as `index.html`, to work. It is only possible to have one Directory module per virtual server.

**Extension** modules handle virtual files, with a certain extension. Each time a request is made to a URL ending with that extension, the extension module will be called. There are no Extension modules in the Challenger distribution.

**File extension** modules deal with files with a particular extension, such as `.html` or `.gif`. The file in question must first have been delivered by a Location module. The *ISMAP image-maps* module is a File extension module.

**Filter** modules filter data that is just about ready to be sent to the browser. This can be used, as the name suggests, to filter out parts of the data that should not be sent.

**First try** modules are called before all other module types, except for Authentication modules. This is used to catch certain types of requests, for instance, to block access to your server from certain IP addresses or to send a warning message to the administrator if the server is accessed outside working hours.

**Last try** modules are called when all other modules have failed to produce anything at all from the request. A Last try module could give an elaborate error message.

**Location** modules deal with file systems, fetching files and directories. A Location module could work with a real file system or a purely virtual one. For example, it could fetch files from a database instead of a file system.

Most web applications are also implemented as Location modules. Pike and CGI scripts work much like a Location module.

Each Location module is mounted somewhere on Challenger's virtual filesystem. Several Location modules may be mounted on overlapping mount points. If the module have the same priority, the module with the longest mount point will be called first. Thus a module mounted on `/schedule/server/` will be called before a module mounted on `/schedule/`.

**Logging** modules perform some logging of information about the requests. This could be done by writing log files or in some other way. The Logging module decides whether the request has been logged properly or whether the request should also be logged by the built-in log system.

**The Main parser** module handles all RXML parsing. The module handles the interface to Parser modules. A Main parser must be installed for any RXML parsing to take place.

The Main parser must somehow get pages to parse. This is usually done by making the module a File extension module as well. Thus all files with a certain extension will be parsed.

There can only be one Main parser module in each virtual server. The *Main RXML parser* module is the only Main parser module included in the Challenger distribution.

**Parser** modules define one or more RXML tag. This is one of the most common user created modules. Making new tags available is an excellent way of making functionality available to the users of the server.

The Parser modules are called upon by a Main parser module. If no Main parser module is installed in the virtual server, no RXML parsing will take place.

**Protocol** modules set the protocols the virtual servers can use. It handles a network connection to a port and then sends the request on to Challenger. HTTP, HTTPS and FTP are examples of protocol modules included in the Challenger distribution.

**Provider** modules provides other modules with different services and extra functionality. They do not in themselves have anything to do with the normal request handling.

**Proxy**

**Types** module sets the content type of a file, if it hasn't already been done by the preceding modules. This is usually done by looking up the Mime type for a certain file extension in a database. There can only be one Types module in each virtual server. The *Content Types* is the only Types module in the Challenger distribution.

**URL** modules receives one URL and returns another. In other words, the URL modules transform requests into other requests. This is useful when a web page is published under several names, or when a web page has moved.

# Filesystems

To make files available via a Challenger server there are a few different types of modules which cooperate.

The source of all files are one or more *Location* modules. They provide files as well as directory information. *Location* modules can be mounted on overlapping mount points, in which case the union of the files provided by the modules will become available.

A *Directory* module is used to handle index files, such as `index.html`, or produce directory listings.

*File extension* modules handle files of a certain extension. The *Main RXML parser* module is a file extension module that usually handles `.html` or `.rxml` files.

If no *File extension* module handles a certain extension the *Content types* module will be called to determine the appropriate content-type for a file.

## Filesystem modules

Filesystem modules are mounted on Challenger's virtual filesystem and handle requests for files and directories. This is usually done through fetching files from a directory on the host computer. But there are modules that fetch files from a CVS repository or another Challenger server.

### Filesystem

The *Filesystem* module is the most basic filesystem module and it simply fetches files from a directory on the host computer.

**Handle DELETE** If set, the DELETE method can be used to delete files in the file system. This is most useful if the FTP protocol is used.

**Handle PUT** If set, PUT can be used to upload files to the file system.

**Mount Point** Where the module will be mounted on the virtual filesystem.

**Require authentication for modification** Only allows authenticated users to use methods other than GET and POST. **Turning this option off makes it possible for anyone accessing the web site to edit pages**.

**Search Path** From which directory on the host computer the files will be fetched.

### User Filesystem

Makes it possible to access files in the user's home directories. All home directories containing a directory whose name is specified in the *Public directory* variable will be accessible.

The *User Filesystem* module needs an authentication module that can provide it with information about the users, such as where their home directory is located. The *User database and security* is such an authentication module.

**Banish list** This is a comma-separated list of users, that should not have any home page.

**Look in users homedir** If set, the user's files are looked for in the home directory of the user, according to the *Public directory* variable. Otherwise, the *Search path* variable is used to find a directory with the same name as the user.

**Only owned files** If set, only those files that a user really owns can be sent. This makes it impossible for a user to publish sensitive information by making a symlink from her home page directory.

**Password users only** Only users possessing a valid password on the system are allowed to have public directories.

**Public directory** This is the location of the home page directory. Assume that it is set to `.public`, that the module has the mount point `/users/` and that Per's home directory is `/home/per`. Now, when the file `/users/per/foo/` is accessed the module will try to find the file or directory `/home/per/.public/foo/`.

**Virtual User Hosting** If set, virtual user hosting is enabled. This means that the module will look at the *host* header to determine which user's directory to access. If this is set, you access the user's directory with http://user.domain.com/ instead of http://user.domain.com/user/. To set this up you need to add CNAME entries to DNS for all your users pointing to the IP addresses of this virtual server.

### CVS File system

This module can access files under CVS, Concurrent Versions System, control. It has options to retrieve differences between two versions, older versions and many other CVS tricks. You need to have CVS installed to use this module.

**CVS (sub)module** There are two ways to specify which directory tree in the repository is to be mounted:

module/subdirectory

Where module is a module defined in the CVS repository, and subdirectory is a (possibly empty) path to a subdirectory of the module.

/path

Where path is the full path to a directory, starting at the CVS root. I.e., the module database in the CVS repository is not used.

**CVS repository** Where CVS stores its files.

**Mount point** This is where the module will be inserted in the name space of your server.

**Path for locating binaries** Colon separated list of directories to search for the `cvs` and `rcs` binaries.

## Restricted File system

A restricted file system that makes it possible to mount each user's home directory on the same URL. The file system will prompt the user for her login name and password, and then only show her files. The most common application for this is for uploading content with FTP.

**Hide path to the home-directory** Hides the path to the home directory if enabled.

For example, if the user *foo* has the home directory */home/foo* and this is enabled, he will see his files in /.

If this is not enabled, he would see them in **/home/foo**.

## Secure File System

The secure file system module works much like the ordinary file system module, but with regular expression based access control. It is also possible to do the authentication via a form.

**Security patterns** This is a list with entries on the form `filepattern: security level=pattern`. Each entry must be in one of the forms listed below.

`filepattern:` **allow ip**`=pattern`
`filepattern:` **deny ip**`=pattern`
`filepattern:` **allow user**`=pattern`

In patterns, `*` matches one or more characters and `?` matches one single character. Please note that the expressions are tested in order, so if you have `*: allow host = *` as the first line, it will not matter whatever you add further down. Everything will still be allowed.

**Use FORM authentication** If set, instead of returning a *HTTP authentication needed* header, it produces a page containing a login form.

## Mirror Filesystem

A mirror file system mirrors the virtual file tree of another Challenger server. The file system connects to a Mirror Server using Roxen RPC. The connection is done in clear text, it is currently not possible to configure it to use encryption.

The search path of the Mirror Filesystem is used as a cache. The directory must only be used for this cache and should not be shared between multiple mirror file systems. Do not let this module connect to a mirror server in the same Challenger server. This will cause your server to hang up with no way to recover, except by manually editing the configuration files and restarting the server forcefully.

**Mirror Server** The location to mirror from. This is not the HTTP location, but the one entered in the *mirror server* on the remote site.

**Mirror Refresh** Check the pages this often, in hours. Please note that the pages are only reloaded from the source server if they have actually changed. At most one file per second will be checked. The update may therefore take a while.

**Search path** The cache directory this mirror filesystem should use.

## Mirror Server

This module makes it possible to mirror the site, or part of the site, to other Challenger servers. The mirror server will access pages as a normal web user, so it is only possible to mirror the parts of the server that is available to the web. No pages that are password protected will be mirrored.

**Variables:**

**Base URL** The start location of the part of the web site that should be mirrored. By default `/` which will mirror the whole site.

**Mirror Server port** The port that the mirror server should bind to. Specified as `IP address:port number`. IP address can be *any* in which case it will bind to all IP addresses the computer handles.

# Directory listing modules

A directory listing module handles accesses to directories, that is URLs ending with a `/`. Usually an index file, for example `index.html`, is fetched and presented. But if no index file is present it can be preferable to show a listing of all files and directories present in the directory.

A directory listing module handles the fetching of an index file or generating the directory listing. Unlike most types of modules it is only possible to have one directory listing module per virtual server.

A directory listing module works on Challenger's virtual file system. That means that it will show the union of all filesystem modules mounted on overlapping mount points.

## Directory parsing module

This is the most used directory listing module. It handles index files as well as generated directory listings. It shows the directory listings with folding/unfolding directories.

**Allow directory index file overrides** If set, you can force Challenger to send the directory listing even if an index file is present. This is done by adding a dot to the URL. Thus `http://www.my.site/dir/.` would show a directory listing even if `http://www.my.site/dir/` gave you an index file.

**Index files** This is a list of names of possible index files. If any file with such a name is present in the directory it will be shown.

## Enhanced directory listings

This module takes the fold/unfold paradigm one step further, it can actually unfold the files themselves. It needs the *Main RXML parser* and *Flik* module to function.

**Include file size** If set, it includes the size of the file in the directory listing.

**Include readme files** If set, the contents of readme files, `README` and `README.html`, will be included in the directory listing.

## Fast directory module

This is a simpler and faster directory listing module. It only prints a list of directories and files, without any fancy fold/unfold buttons.

## Index files only

The *Index files* module will only fetch index files. It will never generate any directory listings.

# Content types

Over HTTP the type of a file is determined by its MIME content type. A GIF image has the content type `image/gif` while a HTML page has the content type `text/html`.

The *Content types* module, the only one of its kind, handles giving each file a content type. This is done through matching the extension of a file with a content type. Thus `.gif` files are given the content type `image/gif` and `.html` files are given the content type `text/html`.

Which extension should be matched to which content type is of course fully configurable. A file with the most common extensions and content types is included with Challenger. The *Content types* module must be enabled for Challenger to operate as a web server. You should only remove it if you are certain that you know what you are doing.

**Default content type** This is the default content type which is used if a file lacks extension or if the extension is unknown.

**Extensions** A list of extensions and their corresponding content types. The format is as follows:

| Extension Type | Encoding |
|---|---|
| gif | image/gif |
| gz STRIP | gnuzip |
| #include | <etc/extensions> |
| #include | <etc/more-ext> |

`STRIP` causes Roxen to add the encoding to the Content-encoding header, strip this extension and try again. A file named `roxen.tar.gz` would not only get the Content-encoding `x-gzip`, but also the Content-type `application/unix-tar`.

The `#include` directive causes files containing more content type definitions to be included. The syntax for those files is the same as the syntax for this variable.

# File extension modules

The following modules are file extension modules that come with Challenger.

## CGI executable support

This module can execute CGI-scripts both from a special directory and on an extension basis. It supports the CGI/1.1 interface. For more information see the CGI page.

## Fast CGI executable support

This module provides support for the Fast CGI interface. For more information see the Fast CGI page.

## ISMAP image-maps

This module enables Roxen to handle image maps. See the Image maps chapter in the "Web Site Creator"-manual for more information.

**Mapfile extension** The extension of the files containing image maps.

## Main RXML parser

This module is necessary for parsing of RXML tags. For more information, see the Main RXML parser page.

## Pike script support

This module makes it possible to have scripts written in Pike that are handled internally in the Challenger server. The scripts themselves work much like CGI scripts. For more information, see the Pike scripts page.

# RXML tags

The Roxen Macro Language, RXML, provides a number of tags which are used in the same way as HTML tags. Before the pages are sent to the browser, the RXML tags are parsed by the *Main RXML Parser* module, which in turn calls other modules to handle each tag. The RXML tags are then translated into normal HTML.

Some RXML tags are only available when a certain module is loaded. These modules and tags are described briefly in the following sections.

Note: The *Main RXML Parser* module **must** be loaded for any RXML parsing to be performed at all.

## Main RXML parser

This module must be installed if you want your Roxen to parse any RXML tags. It does contain the actual RXML parser as well as some RXML tags. Other modules may supply more RXML tags.

This module is a file extension module, you choose the extensions of the files to be parsed. For some sites that may be any `.html` or `.htm` file. Other sites may want to limit parsing to special files, maybe `.rxml` files.

**Access log** This variable controls whether to enable the `<accessed>`. The `<accessed>` tag uses two files, `logs/<virtual server name>/Accessed.db` and `Accessed.names` to store how many times a page has been visited. Since the tag takes more resource than most others it can be disabled.

**Extensions to accesscount** By default the `<accessed>` tag only counts accesses to pages that contain the `<accessed>` tag itself. This means that all pages you put an `<accessed>` on will start counting from zero.

This option makes it possible to specify a few file extensions, access files ending with these extensions will always be counted. Thus it will be possible to count accesses to all `.html` files. This access counting will take a significant amount of resources.

**Extensions to parse** File endings that show which file extensions should be run through the RXML parser.

**SSI support...**

**NSCA and Apache SSI support** If set, Roxen will parse NSCA / Apache server side includes.

**execute command** If set and if server side include support is enabled, Roxen will accept NSCA / Apache exec commands, `<--#exec cmd=`*command*`-->`. This has security implications, refer to the SSI page for more information.

**execute command gid** The group id of processes started by SSI exec commands.

**execute command uid** The user id of processes started by SSI exec commands.

## RXML packages

Packages are a way of defining new RXML tags, or redefine old ones, in a simple manner. Only the administrator can install new packages, but all users can use the packages once they are installed.

A package typically consists of a header of this form:

```
<info>
version="version"
name="Example Package"
doc="This is a simple example of an RXML pack-
age."
</info>
```

followed by one or more <define> tags defining new RXML tags. The file is placed in the directory `local/rxml_packages` directory and given the name by which users are meant to access the package.

Once a package is installed, all users on the server in question can do

`<use package=`*package_name*`>`

to gain access to the named package. As far as the individual users are concerned, the effect will be the same as if they had manually written the `<define>` tags themselves.

## Countdown

This module adds the `<countdown>` tag which counts the time until a specific date/time.

For more information see the countdown tag page in the "Web Site Creator"-manual.

## Flik

Adds the `<fl>` tag, which is used to create folding lists.

For more information, see the fl tag page in the "Web Site Creator"-manual.

### Example

First
Second

## Indirect href

This module gives the `<ai>` container tag, which can be used instead of `<a href=>`. The *Indirect href* module uses a simple database to store URLs and symbolic names. The `<ai>` tag takes a symbolic name as argument and will replace itself with a `<a href=>` tag with the corresponding URL as argument.

The main advantage is that if one of the pages you link to change URL you only have to change it once, in the database.

**Indirect hrefs** The contents of the database, as a symbolic name followed by a URL on each line.

## Obox

This module adds the `<obox>` tag, which draws outlined boxes.

For more information see the obox tag page in the Web Site Creator manual.

### Example

**Sample box**
This is just a sample box.

## Pike tag

Adds the `<pike>` tag, which allows to embed pike code into RXML pages. For more information see the Pike tag page in the Scripting chapter. The Pike tag module is a potential security hazard and should not be enabled unless the administrator understands the security implications of this particular module.

## SED

Adds the `<sed>` tag which emulates a subset of operations from the Unix program `sed`. For more information see the sed tag page in the Web Site Creator manual.

## Tablify

This module enables the `<tablify>` tag which generates tables from tabular data.

For more information see the tablify tag page in the Web Site Creator manual.

### Example

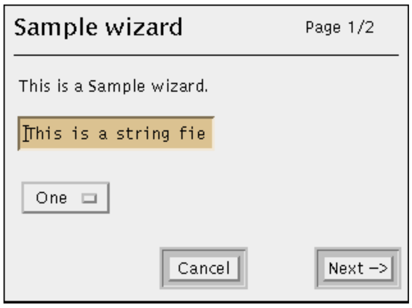
| Product | Price |
|---|---|
| Roxen Screen Saver | $49.50 |
| Roxen T-Shirt | $29.95 |

## Wizard

This module enables the `<wizard>` tag. This tag can be used to create user interfaces like the wizards found on the *Actions* tab.

For more information see the wizard tag page in the Web Site Creator manual.

### Example



# Graphics

Challenger can dynamically draw graphical images that are sent to the browser. This can be used in a number of ways. Headers that need to be graphical to get the right typefaces can be created by the server instead of being created manually. This makes it far easier to handle since the headers are the only text in an RXML tag. To create new headers or change old ones you only need to change the tag, there is no need to use a drawing program.

Challenger can also draw dynamic diagrams. Instead of creating a static report you can utilize Challenger to make the report contain diagrams that always show the current data in your database.

Dynamically drawing graphical images takes more CPU than serving static pages. If you run your server with several threads this should not have any impact on performance. All graphical modules make use of caches, which means that the graphics on a page will only be drawn once regardless of how many times the page is requested.

## Fonts

Challenger supports TrueType fonts, as well as the older bitmapped Roxen font format. The two fonts included in the distribution, *avant_garde* and *lucida* is in the older format. It is however much better to use TrueType fonts, since it is possible to scale them to any size with no bad effects.

On Windows Challenger should be able to use all installed fonts. Since there are no special place for installed fonts on Unix you will have to make sure that the *Global Variables/Fonts.../Font directories* variable includes the directories where your fonts are stored. In case you want to install fonts that are only to be used by Challenger we suggest using the `local/fonts` directory.

## Image file formats

Challenger's graphical tags generate GIF images. Some tags use images as input, `<gtext>` can for example use an image as background. The images used for input can be in GIF, JPEG, PNM or PNG format.

## Graphical text

This module defines the `<gtext>` tag that renders text into GIF images. For more information see the gtext page in the Web Site Creator manual.

**Append .fmt (gif, jpeg etc) to all images** Append the default extension of the image format to all URLs generated by gtext.

**Default maximum text-length** By default, the module will not try to render texts longer than this. The purpose of this is to prevent common coding errors such as mismatched tags from causing the graphics text module to parse the entire document. This can be overridden by using the "maxlen=..." attribute in the tag.

**Normalize colors in parsed tags** Challenger can handle colors in many different formats, RGB (#rrggbb), CMYK (@c,m,y,k) and as English words (black). Not all versions of browsers support those color formats. If this variable is set the *gtext* module will convert the colors in the HTML tags specified in the *Tags to parse for color* variable to the RGB format (#rrggbb).

**Parse tags for document colors** If set, parse the specified tags for document colors.

**Tags to parse for color** When rendering an image it is important to know the background color. The *gtext* module can try to find out the background color by parsing HTML tags where you can set background color. This way the user does not have to explicitly set the background in the `<gtext>` tag. On the other hand, parsing all those tags will use CPU. Often more CPU than the actual image generation, since every one of these tags will be parsed for every RXML page, regardless of whether there are any `<gtext>` tags on the page or not.

If you later decide to change this variable to be more restrictive, some pages that rely on this feature might stop working. If you are concerned about CPU usage you should be restrictive from day one. You will have to reload this module or restart Challenger to make changes to this variable take effect.

### Example


## Business Graphics

This modules defines the `<diagram>` tag, that can be used to draw charts and graphs. For more information see the diagram page in the Web Site Creator manual.

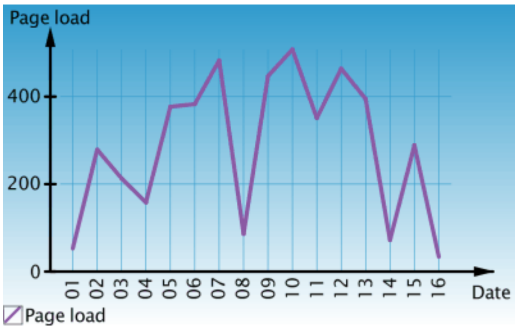
**Limits...**

**Max height** The maximum height of the generated image.

**Max string length** The maximum length of text strings used in the diagrams.

**Max width** The maximum width of the generated image.

### Example




## Pike image module

This module adds two new tags, the `<gclock>` tag and the `<pimage>` tag. The `<gclock>` tag draws animated clocks while the `<pimage>` container tag works like a `<pike>` tag with a few extra functions especially suited for image generation. For more information see the gclock and pimage pages in the "Web Site Creator"-manual.

**Mountpoint** The URL-prefix for the pike image module. (Location in the virtual file system)

**The PIMAGE tag is enabled** If set, the `<pimage>` tag will be available for use. This tag has the same security considerations as the `<pike>` tag. If not set, only the `<gclock>` tag will be available.

The `<pimage>` tag is a potential security hazard and should not be enabled unless the administrator understands the security implications of this particular tag. For more information see the Pike tag page in the Scripting chapter.

### Example

# Proxy

A proxy server is used to give access to network services indirectly. All accesses are handled through the proxy server. There are two main reasons for using proxy servers; to enhance security and to reduce network bandwidth.

The rationale for enhancing security through use of a proxy server is that only the computer running the proxy server will access the Internet directly. Therefore it is only that computer that is vulnerable to attacks from the Internet. And since securing one computer is much simpler than securing all computers, security is enhanced.

Network bandwidth can be reduced by the proxy caching requests. If several users request the same page it will only need to be fetched once over the network. Unfortunately not many pages are cacheable, since there are so much dynamic pages out there. But it is usually possible to cache images.

Challenger's proxy capabilities are handled by the *HTTP-Proxy*, *SSL Proxy* and *FTP gateway* module. The simplest way to set up a Challenger proxy server is to create a new virtual server and choose the configuration type *Proxy*.

A proxy server must bind to its own port, it is not possible to run it IP-less. Nor is it possible to have an *IP-less virtual hosting* module enabled in the proxy server. Strange things will happen.

Since the proxy functionality is implemented as modules it is possible use them in other ways too. For example, it is possible to fetch www-pages through ftp by enabling the *HTTP-Proxy* on a virtual server with an ftp port.

It is important not to enable any dangerous modules in a virtual server with proxy modules. It is perfectly possible to get a *.pike file fetched through the *FTP Proxy* module to run through the *Pike script support* module.

It might be necessary to enter a security pattern for each proxy module, so that it can only be accessed from within your network. Otherwise your proxy server may give outsiders access to servers within your network. This is especially important for the *SSL Proxy* module.

## HTTP proxy

The *HTTP-Proxy* module is a filesystem module that fetches files from other web servers. It is usually used as a proxy for HTTP, by being mounted on the mount point `http:/`.

The *HTTP-Proxy* module will cache requests to disk, if the disk cache is enabled.

**Logfile** If a filename is entered proxy requests will be logged to that file, as well as to the normal access log file.

## SSL proxy

The *SSL proxy* module implements the Connect method, an extension to HTTP that can be used to tunnel HTTPS requests. Since an HTTPS request is encrypted it is not possible to proxy it, nor to cache it. Unfortunately, this also means that an SSL proxy can be used to tunnel any TCP/IP connection, something that might break your security scheme. Therefore, it is possible to limit the ports to which the module can connect.

In Netscape terminology, a virtual server with this module enabled is a secure proxy.

Outsiders might get access to your network by connecting to the SSL proxy. Therefore, it is recommended that you use security patterns to limit access to this module to hosts within your network.

**Allowed Ports** Limit access to certain ports. Enter 443 here if you only want to allow connections to the standard HTTPS port.

**Connection refused message** Message to send to the user in case it was not possible to connect to the server.

**Forbidden Ports** Forbid access to certain ports.

**No such host message** Message to send to the user if it is impossible to locate the server.

## FTP gateway

The *FTP gateway* module is a filesystem module that fetches files from FTP servers. It is usually used as a proxy for FTP, by being mounted on the mount point `ftp:/`.

The *FTP gateway* module does not cache requests.

**Logfile** If a filename is entered, proxy requests will be logged to that file, as well as to the normal access log file.

## Disk cache

The disk cache subsystem of Challenger handles caching for proxy modules. Currently only the *HTTP-Proxy* module caches requests. The disk cache is enabled by the global variable *Global Variables/Proxy disk cache.../Enabled*.

It is the disk cache system that handles which requests are to be cached. Files in the disk cache are removed by a garbage collector process, that is run at specified intervals.

**Base Cache Dir** The directory where the disk cache should be kept.

**Bytes per second** How the garbage collector should treat file size. Each byte of file size will be equal to a certain number of seconds of age. This way, really large files will be treated as older files and removed before smaller files of the same age.

**Clean size** Minimum number of megabytes to remove per garbage collect.

**Garbage collector logfile** If set, a log file will be kept with information about removed and refreshed files, as well as cache and disk status.

**Keep without Content-Length** Keep files that have no Content-Length header. This will make the cache store more files, but it will also be possible that partially downloaded files, as well as dynamic pages, are cached.

**Last resort** How many days to keep files with no Expire or Last-Modified headers.

**Maximum number of files** Maximum number of files to be cached.

**Minimum available free space and inodes** What percentage of disk space and disk inodes must be available on the file system with the cache. By setting this variable, it is possible to make sure that the disk cache will never fill the entire file system.

**Number of hash directories** The number of directories the cache should consist of. If this variable is changed, it will no longer be possible to access the old cache.

**Refresh on Last-Modified** If set, the disk cache will use the Last-Modified header to determine how long a file should be cached, if the file doesn't contain any Expire header. The file will be kept until it has doubled it's age.

**Size** The maximum size of the cache, in megabytes.

# Miscellaneous modules

The modules described in this chapter do not easily fit into the other categories. Some of them have quite unique abilities.

## Demo module

This module makes it possible to develop RXML interactively, on a web page. Once you have written your RXML all you have to do is press *Show* to see if it works. This is great for trying out new tags and for teaching RXML. This module is a potential security hazard, since anyone who has access to it has access to all installed modules that use RXML. Therefore, access should be limited to internal users.

**Mount point** This is where the module will be located in the virtual filesystem.

**Builtin variables/Security: Patterns:** It is highly recommended to insert a security pattern that limits who has access to this module. For example by the security pattern `allow user=any`.

## Language

This module handles documents available in different languages. Documents get an extra suffix, specifying the languages. For instance, `.sv` would be a resource in Swedish, and `.en` one in English.

To decide which resource to send to which user, the module looks at a cookie or a prestate. It does not use the `Accept-Language` header of HTTP, since the semantics differ. Accept-Language is based upon the assumption that the user chooses a fixed list of languages once, and then automatically gets pages accordingly. The *Language* module is based upon the assumption that the user wants to know which languages a document is available in, and be able to make the choice any time. The user might well want to look at the different translations of a page to see if one translation contains more information.

The *Language* module is also a directory parsing module. It has to be in order to handle `index.html` files in different languages.

**<language>** Prints the language the current page is in. See the language page of the Web Site Creator manual for more information.

**<available_languages>** Gives a list of all other languages the current page is available in, and links to them. See the available_languages page of the "Web Site Creator"-manual for more information.

**<unavailable_language>** Shows the language the user requested the page in, if the page is not available in that language. See the unavailable_language page of the "Web Site Creator"-manual for more information.

### Variables

**Default language** This is the default language for documents on this site. It is used when deciding which language file to send when the user has made no choice. Files without a language extension are considered to be in this language.

**Languages** This specifies which languages are supported by the site. Support for each language is defined on one row, on the form *language code*, *language name* and optionally one or more *next language code*.

An example follows below.

```
sv Svenska en de
en English de
de Deutsch en
```

The *language name* is used in the tags to show what language the page is in and to create links to the other translations.

*next language codes* are used to determine which language to use if the one selected is not available. To find a page in an appropriate language, languages are tried as follows:

1. The selected language, stored as a prestate.
2. The user agent's accept-headers.
3. The default language.
4. The default `next-language-codes` for the default language.
5. All languages, in the order listed in this variable.

Empty lines as well as lines beginning with `#` or `//` will be ignored.

**Flag directory** The path to a directory holding small GIF format image files of flags or other symbols, representing the various languages.

language-code**.selected.gif** Shown to indicate that the page is in that selected language, usually by the header-module.

language-code**.available.gif** Shown as a link to the page in that language. Only shown if the page is available in that language.

language-code**.unavailable.gif** Shown to indicate that the user has selected a language that this page is not available in.

language-code**.dir.selected.gif** Shown to indicate that the directory entry will be shown in that language.

language-code**.dir.available.gif** Shown as a link to the directory entry translated to that language.

**Text only** If set the tags will default to text only.

**Use config** If set the module will use cookies for storing the users language preference. Will use prestates otherwise.

## Spell checker

Adds a new container tag, `<spell>`, that will check its contents for common English misspellings. See the spell tag page in the "Web Site Creator"-manual for more information.

## Kill frame-tag

This module defines a tag, `<killframe>`, which inserts JavaScript that will prevent others from using your page in a frame. It also strips any occurrences of `index.html` from the end of the URL. For more information see the killframe tag page in the "Web Site Creator"-manual.

### Variables

**Kill trailing 'indexfiles'?** When set, the killframe module will remove occurrences of index files (as set in the active directory module) from the end of the URL, leaving only a slash.

# Security considerations

One of the most important tasks for a Challenger administrator is to maintain security. A web server is reachable by any user on the internet, some which might harbor ill intent. The whole server, including all user or third-party scripts, must be made secure. It is necessary to have an understanding of the issues involved.

One fundamental task is to decide who to trust, and to what extent. Do you trust that the programmer of a third-party script has taken security considerations into account when making the script? Do you trust your users not to abuse the `<pike>` tag?

It is also important to watch your log files. Problems, such as attempted break-ins, will often show up in the log files. You should watch the access logs as well as the debug or event log.

An important task is to educate your users and programmers about security. You are the one who has a vested interest in security, users and programmers will often see it as a nuisance, unless they are taught about its importance.

## Trusting code

The easy sharing of code via the web is both a blessing and a curse. It makes it simple to get the functions you want. On the other hand, it makes it easy to compromise your security. You leave the security of your server in the hands of the programmer who has made scripts you install, so you better make sure that it is a programmer who takes the security issues seriously.

You should always check who has written a script and who maintains it. You should make sure that the programmer and maintainer describes the security issues in a serious way in the documentation of the script.

## User input

The main problem when programming on the Web is handling user input. Unlike a normal application, a web application cannot limit what may be sent as user input. It must be able to handle anything, be it megabytes of machine code in a field meant for a single word.

There are two problems associated with user input, *buffer overruns* and *special characters*. Buffer overruns in a problem that can occur when writing scripts in compiled languages like C or C++. In such languages, it is tempting to make fix-length buffers. The problem is that an attacker can execute machine code by sending more input than the length of the buffers.

Special characters are a problem for any script that in itself runs external programs or connects to databases. Some characters have special meaning for the command interpreter or the SQL interpreter. The programmer must therefore make sure to quote any occurrence of such characters.

When using RXML together with dangerous tags, like the `<pike>` tag, it is very important to quote <, > and &. If any output tags are used, like `<formoutput>` or `<sqloutput>` it becomes necessary to quote : as well.

## Pike scripts or modules

Challenger's built-in scripting capabilities use Pike, which makes it impossible to create buffer overrun bugs. There is simply no way to execute machine code from Pike. Pike also contains safe methods for starting external programs, as well as methods for quoting dangerous characters when connecting to databases.

When using RXML to connect to databases, dangerous characters must also be quoted. This is done by default. For more information about RXML quoting see the formoutput page in the Web site Creator manual.

## Trusting users

We can distinguish between three different categories of users. How much you trust a user will depend on which category she belongs to.

**External users** People looking at your web site.

**Internal users** People that can create and maintain web pages.

**Administrators** People who have access to Challenger's configuration interface.

External users are usually not trusted at all. Most of your work is to ensure that external users cannot break the security of the web server. There are several different access control schemes available that can be used to limit access to certain pages for certain networks or users.

Internal users are trusted to a degree. How much depends on your organization. For example, if you are administrator for a school or ISP, you will probably not trust internal users very much. If, on the other hand, the internal users are colleagues you meet daily, you will probably trust them to a higher degree. It is possible to use trust levels to give different internal users access to different modules.

Administrators have access to the whole web server. The question is whether they also have access to the machine the web server runs on. This determines which user to run the server as.

## Trustlevels

Trustlevels is a builtin permission system that can be used to give access to different modules for different internal users. That way, you can limit access to potentially dangerous modules to trusted users.

## Challenger user

A question you should ask yourself when installing Challenger, is whether to run Challenger as a normal user, with limited system privileges or, as the superuser, with unlimited system privileges. Server processes are traditionally run as superuser since a server usually has to perform some tasks that need superuser privileges. Servers run by the superuser are usually easier to run and maintain but they also have the potential of becoming more serious security hazards since an attacker, who gains access to such a server, will have access to more or less the entire system.

Challenger is no exception. It needs superuser privileges for essentially two tasks, opening privileged ports and starting CGI scripts as the user who owns the script. Challenger is written to be able to run as superuser securely, but here, the configurability and extensibility can be problematic. If the server is configured the wrong way, any user who can write web pages can break into the server. Extensions to Challenger that are not written with security in mind might even make it possible for outsiders to break into the server.

So lets take a closer look at the two reasons why Challenger needs to be superuser and how Challenger can be run without being superuser.

### Privileged ports

The default ports for HTTP, 80, HTTPS, 443 and FTP, 21 are all privileged. That means that only the superuser can open them. If Challenger is to use the privileged ports, which is necessary for it to handle a normal web site, Challenger needs to have superuser privileges when opening the ports.

### CGI scripts

Challenger has the option of running a CGI script as the user it belongs to. This can only be done if Challenger has the privilege to start processes as another user, a privilege that belongs to the superuser only.

### Roxen Application Launcher

It is necessary to run as root if Roxen Application Launcher is to be used to enable SiteBuilder users to start local editors, in named mode. That is, if we use the operating systems access control functionality for protecting the files exported from SiteBuilder to the local editor. If used in anonymous mode it is not necessary to run as root.

### Running as a normal user

There are three ways of running Challenger as a normal user. It can simply be started as that user, it can switch to that user temporarily, or it can switch to that user permanently. Switching to another user is controlled by the *Global Variables/Change uid and gid to* and *Global Variables/Change uid and gid permanently* variables.

### Starting as a normal user

Starting as a normal user is the safest way, since Challenger then won't run for a single second with superuser privileges. Unfortunately, this also means that Challenger has no way of opening the privileged ports, and another program will have to do this and relay the ports to Challenger. Incidentally that other program could be a Challenger using the *HTTP Relay* module.

### Temporary changing to a normal user

If run in this mode Challenger will drop superuser privileges and switch to a normal user, but it will switch back to superuser when it needs to. This will improve security, but since the server still runs as superuser now and then, a malicious and expert user who breaks the security of the server may still gain superuser privileges. **This option should not be used together with threading. Strange things happen to threaded programs that switch user**.

### Permanently changing to a normal user

With this option, Challenger will run as superuser until it has opened its ports, and then switch to a normal user with no possibility of switching back. Two consequences of this are that if you configure a new privileged port you have to restart the server for the change to take effect and that CGI scripts can only be run as the server's user.

All modules will be loaded while the server is still running as superuser. This means that you still have to trust all the modules you have enabled. It is also important that the Challenger user does not have permission to change the module or any file in `server/`. Nor should the server be able to change its `configurations/Global_Variables` file. The first is to ensure that an attacker cannot replace the server with her own program or module. The second is to ensure that an attacker can't change back the setting that makes the user change permanent.

### Using simple front end

One way of having it all, both security and functionality, is to use a front end that opens the privileged port and then relays requests to a server running as a normal user. The front end could be a Challenger server using the *HTTP Relay*. It could as well be a simpler relay program, or maybe a dispatcher or router.

## Restricted pages

Control over who gets to see certain information or use a certain service can be achieved in three ways; RXML, security patterns or .htaccess files. All three ways have one thing in common, they make use of an authentication module. The authentication module contains a user database with user names, passwords and information about the users. Challenger comes with three such modules, the *User database and security* module, the *SQL User database* module and the *LDAP directory authorization* module.

In RXML, access control is achieved through use of the `<if>` tag. It is possible to make a very fine-grained access control system with this tag, since access to information on the actual pages can be limited. See the If page in the Web Site Creator manual for more information.

Security patterns are used to limit access to an entire module. For example, it can be used to make sure only internal users can access the *Demo* module.

.htaccess is a standard used by many web servers and can be used to limit access to certain directories or files. It is implemented through the *.htaccess support* module. More information about .htaccess can be found in the security chapter in the "Web Site Creator"-manual.

## Security patterns

Security Patterns are one of the variables under *Builtin variables*. They can be used to control who gets to access a certain module.

If security patterns are not used, anyone can access the module. Once a security pattern is entered only users who are matched by the security pattern will be granted access.

The patterns are scanned from top to bottom. Each line contains a rule with a pattern matching users who should be affected by the rule.

## Rules

The rule determines what should happen in case the user or computer is matched by the pattern. Should the request be denied or granted?

**accept** A user matched by an accept rule will be granted access, unless she is matched by a deny rule further down. The processing of patterns will continue, in order to determine if such a deny rule exists.

**allow** A user matched by an allow rule will be granted access. No further processing of the patterns is required.

**deny** A user matched by a deny rule will be denied access. No further processing of the patterns is required.

## Patterns

Each rule contains one of the following patterns, that are used to match users that are to be affected by the rule:

**ip=IP/bits** Grant/deny access from a network, IP specifies the network address, bits the size of the network. `allow 194.52.182.0/8` will grant access to any machine who's IP-address starts with 194.52.182.

**ip=IP:mask** Grant/deny access from a network, IP specifies the network address, mask the subnet mask. `allow 194.52.182.0:255.255.255.0` will grant access to any computer whose IP address starts with 194.52.182.

**ip=pattern** Grant/deny access from a named network. `allow *.idonex.se` will grant access to any computer who is named something ending with idonex.se. * is used to match zero or more letters while ? matches only one letter.

**user=username** Grant/deny access to a specified user. The user will be authenticated using the authentication module or .htaccess password file. This option is usually silent, that is, a user will not be prompted for a user name and password if she fails to authenticate properly.

`user=any` will grant/deny access to anyone with a valid account.

## Example

```
allow user=any
allow ip=194.52.182.0/8
```

Will grant access to any user who is authenticated or comes from the network 194.52.182.0.

## .htaccess support

Support for NCSA/Apache .htaccess files. See the security chapter in the "Web site Creator"-manual for information about how to use .htaccess files.

**Cache the failures** This causes the *.htaccess support* module to remember where it failed to find a .htaccess file. This improves performance significantly, but users must do a reload to get a new .htaccess file to take effect.

**Deny file list** Files to which access will always be denied, no matter who tries to access them. Usually .htaccess related files are protected this way to make it impossible for an external user to get any information about how the access control system works.

## User database and security

This module fetches user account information from the operating system's user database. It is used when the users that should have access to the web server already have accounts on the system. This module must be used for modules enabled if the *CGI scripting support* module or the *Pike script support* module are to be able to run scripts as the user who owns the script.

## Variables

**Password database request method** How to get account information from the operating system. Choose one of the following options:

**ypcat** Use the `ypcat` program. For systems that use yp.

**file** Read accounts from the password file specified in the *Password database file* variable.

**shadow** Read accounts on a system that uses shadowed password files. You need to specify a password file in the *Password database file* variable as well as a shadow password file in the *Password database shadow file* variable.

**niscat** Use the `niscat` program. The same as `ypcat` for Solaris systems, where yp has been renamed nis.

**getpwent** Use the system call `getpwent` to get user information. This will work on all systems, but is slower than the other methods.

**none** Don't import any user account information. Let any username through no matter what password was submitted.

# Trust levels

Trustlevels are a system that can be used to limit access to certain modules depending on where the file originates. It can be used to give trusted persons access to potentially hazardous modules like the *CGI executable support* module or the *SQL-module*.

Trustlevels work by setting the *Builtin variables/Security: Security level* variable of modules.

A request is initially assigned a trust level equal to the security level of the filesystem module from which it originates.

The request will only be able to pass through modules with an equal or lower security level than the request's trust level. Modules that has a higher security level will be ignored, like they were not even enabled.

If the request passes through a module with a lower security level than the request's trust level, the trust level will be lowered to the security level. If the request tries to pass through additional modules it will use the lowered trust level.

## Example

```
CGI executable support - trust level 1
Pike tag - trust level 1
Filesystem - trust level 1
User Filesystem - trust level 0
```

The *Filesystem* module can contain CGI scripts or pages using the `<pike>` tags. We can assume that only trusted users can write files that are handled by this module.

User home pages, that are handled by the *User Filesystem* module may not, on the other hand, contain CGI scripts nor use the `<pike>` tag.

# Scripting

One of the most exciting things about the web is that you can make your own applications, that will be reachable by anyone in the world. Furthermore, programming for the web is often simpler than traditional GUI programming. Even small applications can get nice graphical user interfaces by creating dynamic HTML pages. Challenger is one of the best environments for creating such applications.

As with all good things, there are drawbacks. Since an application on the web is reachable by any number of users, some with malicious intent, programming errors can have drastic effects. While many users may not understand this, the administrator of a web server must.

The important thing is that all user input must be handled with caution. Where the programmer thought he would get a small name he might get ten megabytes of machine code. If the program fails to handle that kind of input, troubles might follow.

Building web applications with Challenger in Pike reduces the risks and consequences of making such mistakes, but it does in no way eliminate them.

Challenger also supports CGI scripts for doing scripting. It is far easier to make fatal mistakes when programming CGI scripts than it is with Pike scripts or modules. Most CGI scripts that can be downloaded from the web have not been written with security in mind. As system administrator, you must determine which scripts are safe and which ones are not, and consider your site's security policy.

It is always a good idea to keep track of Challenger's log files. If outside users try to break in through CGI scripts, it will most often show up in the log files. Especially since they will usually try to break in through a few common CGI scripts.

This chapter describes Challenger's various ways of supporting script programming from a system administrator's viewpoint.

## Pike modules

Pike modules are, strictly speaking, not *scripts* in the same sense as the other script types. Rather, they are a way of extending the server's functionality. Most of Challenger's functions are implemented as Pike modules. However, since Pike modules are often a convenient way of doing things that might otherwise have been done with regular scripts, Pike modules belong in this overview.

Pike modules can be used to implement new RXML tags, support new communication protocols, and numerous other things that would be impossible or difficult with other script types. Pike modules are also persistent entities in the server, as opposed to ordinary CGI scripts, which are executed in a single-shot fashion, starting anew each time they are invoked. FastCGI scripts are somewhat persistent.

From a security viewpoint, Pike modules must be treated with utmost care. Pike modules have access to the whole server. Thus, only trusted users should be able to write their own Pike module, and Pike modules should only be downloaded from reputable web sites. Apart from the security considerations, Pike modules that take a long time executing can also cause performance degradation and other service disruptions, especially if the server isn't running with enough threads.

Challenger is an excellent tool for developing Pike modules. Compile time and run time error messages will be reported, with a Pike backtrace so the problem can be pinpointed. All such error messages will be reported on the *Event Log* tab and in the debug log. They will also be reported on the web page where the error occurred, unless the *Global Variables/Show the internals* variable is disabled.

## Pike scripts

Pike scripts are an easy and quick way of doing scripting in Challenger. Since Pike is also the language that Challenger uses internally, Pike scripts are efficient, and easy to handle.

Support for Pike scripts in Roxen Challenger is provided by two different modules, the general *Pike script support* module, and the *Restricted Pike script support* module. The difference is mainly that while the former gives the scripts access to the whole server, the latter runs each user's scripts in a separate server process running with that user's access privileges, greatly reducing the potential security problems that might result from faulty scripts.

As with Pike modules, error messages are reported complete with a Pike backtrace. This makes development fast since it is simple to pinpoint the problem.

Pike scripts are usually persistent. That means they will be compiled only once, and invoked repeatedly. This makes them fast and efficient.

### Pike script support

**Extensions** The extensions of Pike scripts.

**Fork execution** Whether to invoke Pike scripts by forking or handle them internally. Forking is slower and uses more resources but is also more secure since the forked script will run as the user who owns the script and cannot take control over the server. Fork execution does not work with threading.

## Pike tag

The Pike tag is an RXML tag, allowing the user to insert bits of Pike code into RXML pages. This is very convenient, if the function needed exists as a RXML tag, use it, otherwize, write a small Pike function that solves the problem. Unfortunatly it is also dangerous. The Pike code within the tag has full access to the server. Therefore, it is important that only trusted users have access to the Pike tag. Refer to the chapter about trust levels for information about how you can limit the number of users who have access to a dangerous tag. It is also very important to make sure that all user input is quoted properly before being inserted in a web page.

As with all Pike development in Challenger, error messages will be reported together with a backtrace, making it easy to pinpoint any problems.

The *Pike tag* module contains no configurable variables.

## CGI

CGI scripts are the most common way of doing scripting, being supported by virtually every web server. They work by starting an external program for each request. The program can be a compiled program or a script written in something else, for example like perl or python.

The advantages of CGI scripts are that they can be used to run any kind of scripts written in any language, and that they are portable between different web servers. The disadvantages are that they are resource hungry, needing to start an external program for each request. CGI scripts are also one of the most common security hazards on the web. Most often because they were not written with security in mind.

In Challenger, CGI scripts are supported through the *CGI executable support*. The module can be configured to either run the CGI scripts as the user who has written them, or to run all CGI scripts as a user with low privileges.

It is possible to use CGI scripts together with RXML. Either by letting the output of the CGI script be parsed with the RXML parser or by executing the CGI scripts with the `<cgi>` tag.

Even if CGI scripts are run so they cannot hurt the server itself, they can often hurt the user who owns them. Most users do not understand how CGI scripts work, but will download and install them from the net, with no thought of security. It is often better to provide the functions the user's want as RXML tags. The user will understand RXML tags better, since they are like HTML tags, and the administrator will get better control over the server.

### CGI executable support

**Allow listing of cgi-bin directory** If set, the users can get a directory listing of the cgi-bin directory.

**Allow symlinks** If set, allows symbolic links to binaries owned by the directory owner. Other symlinks are still disabled.

This option has an effect only if the *Run user scripts as owner* variable is set and is available only when the server is run as root. .

**CGI-bin path** The module's location in the virtual filesystem. By default the module will also handle one or more extensions, from any filesystem.

**CGI-script extensions** Extension of files to be handled as CGI scripts. The *Handle *.cgi* variable has to be set for this option to have any effect.

**Handle *.cgi** This handles files ending with the extensions configured in the *CGI-script extensions* variable. If set, files with these extensions will be handled as CGI scripts, regardless from which filesystem they were fetched.

**Limits**

**Priority** This option affects the nice value of the CGI processes. If it is set to a higher value CGI scripts might get more CPU than the actual web server, something which might not always be a good idea.

**Log CGI errors to...** Where to log error messages from a CGI script, or rather any output the script writes to *stderr*. By default the error messages will be sent to the debug log file.

**Parse RXML in CGI-scripts** If this option is set the output of the CGI script is sent through the RXML parser. The parsing will take place after the CGI script has finished processing, nothing will be sent to the user until the CGI script finishes. This option will not work if you have CGI scripts that does animations or other things that require them to send data over a long time to the user. It is however possible to enable two *CGI executable support* modules, one to handle scripts that are to be RXML parsed and another to handle normal CGI scripts.

This is option is only available if you have chosen to *More options*.

**Provide the <cgi> tag** If set it will be possible to execute CGI scripts via the `<cgi>` tag.

**Run scripts as** Which user to run the CGI scripts as. This will default to nobody if nothing is specified. This option is only available when Challenger is run as root.

**Run user scripts as owner** If set, scripts in user home directories will be run as the user. This overrides the *Run scripts as* variable. This option is only available when the server is run as root.

**Search path** The location of the CGI-bin directory in the read file system.

**Set the supplementary group access list** If this option is set the script will be run with membership in all the users supplementary groups, i.e. the groups in the `/etc/group` file.

**Treat non-executable files as ordinary files** If this flag is set, files that does not have the executable bit set will be treated as normal files and sent to the user. If the flag is not set attempts to get such files will result in an error message.

## SSI

SSI is short for Server Side Include, and is essentially a way of running CGI scripts within HTML pages. They are like RXML tags, but not as flexible. SSI is a standard that works on several web servers.

SSI support is handled through the *Main RXML parser* and is turned off by default.

SSI works like a simple pre-processor for HTML, and reads directives that are hidden in HTML comments. One directive, `<!--#exec-->`, allows a user to execute programs and is thus a potential security hazard. The programs will be executed as a user with few privileges, by default *nobody*. The directive can also be disabled altogether.

## FastCGI

FastCGI is a more efficient way of doing things than regular CGI scripts. FastCGI works by launching an external process which keeps running once started, processing one request after another, instead of having a new process started for each request.

Challenger's FastCGI support is handled through the *Fast-CGI executable support* module.

As with CGI scripts, there are security hazards involved with running FastCGI scripts. They are most commonly caused by the low quality of existing FastCGI scripts.

### Fast-CGI executable support

The *Fast-CGI executable support* module has the same variables as the *CGI executable support* module, plus an additional variable.

### Variables

**Number of simultaneous copies to run** Specifies how many copies of each script to run simultaneously. If a script takes a long time to process request it can improve overall performance to run several scripts simultaneously.

## Servlets

Java Servlets is a new form of scripting that is gaining popularity because it allows portable server extensions that do not rely on starting external binaries. Servlets are written in Java and compiled to Java byte code before they can be used in a web server.

In Challenger, Servlets are supported through the *Java servlet bridge* module, which transforms a Servlet into a Location Module. The *Java servlet bridge* module can have any number of copies, so any number of Servlets can be installed on a single virtual server.

To run the Servlet support you need a Pike that has been compiled with Java support. The binary versions of Challenger are not compiled for Java support. To compile a Pike with Java support you have to install JDK 1.2 and then compile Pike. Pike's Java module should detect the Java environment automatically.

### Java servlet bridge

**Class name** The name of the Java class implementing the Servlet. The module will look for the corresponding `.class` file in the *Code directory* (see below).

**Code directory** The location of the `.class` files for this Servlet in the real file system. The path is relative to the server directory. Multiple Servlets can reside in the same code directory, as long as they have unique class names.

**Parameters** Servlet-specific parameters can be set here, see the documentation of the Servlet in question to find out what parameters it supports. Each parameter must be placed on a separate line, with the name and the value separated with an equal (=) sign.

**Servlet location** This Servlet's location in the virtual file system. Any URL starting with this prefix will be handled by this Servlet. The rest of the URL will be provided to the Servlet as "Path Info".

Note that reloading the bridge module will reload the Servlet itself from its `.class` files as well, which can be useful when developing new Servlets.

# Databases

Combining databases with the web has many uses. The web is very good for presenting data from databases and for making database driven applications available to the whole world. Challenger includes modules for database connections. These modules, together with such modules as *Business Graphics* and *Wizard*, makes it simple to do reports from databases as well as applications. Challenger also contains a module that uses a table stored in a SQL database for doing user authentication.

Challenger needs a Pike module to be installed if it is to connect to a certain kind of SQL database. By default, modules for the free databases ODBC, mSQL, MySQL and Postgres are provided. Modules for connecting to Oracle, Informix are available with the full Roxen Platform.

### Database URLs

A connection to a database is specified with an URL-like syntax:

```
dbtype://user:password@db.host/dbname
```

The database type *dbtype* is one of msql, mysql, postgres, or odbc. The *user* and *password* are used for authentication of the user in the database server, *db.host* is the name of the machine running the database server and *dbname* specifies the name of the particular database.

### Symbolic names

You do usually not want to specify a full database URL in a RXML tag. With the *SQL Databases* module you can give symbolic names to database URLs. This makes it unnecessary to have any database passwords in the actual web pages. It also makes it possible to change databases without changes to the pages.

### Security considerations

Your foremost security consideration when it comes to databases is to make sure that only the SQL queries you intend get sent to the database. This means handling user input in such a way that it can never change the actual SQL query. This is done through quoting. The formoutput page in the Web Site Creator manual documents shows how to do it in RXML.

To reduce your risks, use the access control system of your database to make sure Challenger only has permission to do what it actually needs to do. If you use Challenger to provide reports from the database, Challenger should only be able to read tables, never modify them.

## SQL module

The *SQL Module* module provides the RXML programmer with three tags for executing SQL queries; `<sqloutput>`, `<sqltable>` and `<sqlquery>`. All three connect to a SQL database and execute an SQL query, the only difference between them is how the result from the query is handled.

**Default SQL-database host** The database URL to use by default in the RXML tags. If the correct database URL is specified here with username and password, these variables do not have to be specified in the RXML tags on the pages.

However, this only works if only one database is used. Connecting to more databases requires the *SQL Databases* module to be installed.

## SQL databases

The *SQL Databases* module makes it possible to bind database URLs to symbolic names. These names can later be used instead of the full URLs when referring to the databases, resulting in shorter and less cluttered RXML code.

This gives an extra layer of security, since it makes it possible for example to write RXML pages with SQL tags without knowing the password for the database. It also eliminates the potential security risk associated with storing of names and passwords on RXML pages that is mentioned in the SQL tags section.

Another advantage is that it becomes very easy to change to another database.

**Table** The actual list of symbolic names and complete database URLs. Each line contains a name followed by the database URL.

## SQL user database

Sometimes Roxen Challenger needs to access data about the users. The main reason for this is user authentication, but Roxen can also do other things with data from the user database, for example, displaying them on a webpage.

The SQL User Database module enables Roxen Challenger to keep such user data in an SQL database. The data is stored in a table with the columns *username*, *passwd*, *uid*, *gid*, *gecos*, *homedir* and *shell*. These columns correspond to the fields in a UNIX password file, and are the fields that Challenger Authentification modules use. The database must contain the *username* and *passwd* columns, other columns can be replaced by default values. It is also possible to add extra columns when needed.

**Cache entries** This flag defines whether the module will cache the database entries or not. Makes accesses faster, but changes in the database will not show immediately. Recommended.

**Close the database if not used** Setting this will save resources when the module is not used.

**Database close timer** How many seconds of inactivity it should take before the database connection is closed.

**Defaults...**

**Gecos** Default in case there is no *gecos* column.

**Group ID** Default in case there is no *gid* column.

**Home Directory** Default in case there is no *homedir* column.

**Login Shell** Default in case there is no *shell* column.

**User ID** Default in case there is no *uid* column. Some modules require the uid to be unique, so it is safer to acctually store a uid in the database.

**Disable Userlist** One of the features of an authentification module is to get a listing of all users. In the case of a large user database the call can take a significant amount of time to process. The feature can therefore be disabled.

**SQL server...** The database URL to the database containing the users.

**Passwords table** The name of the table containing the users.

# ODBC

To connect to a database through ODBC is a bit different from using the databases supportred natively. You need to install a ODBC driver for the database and then configure it to connect to your database manager. On Windows the configuration is simple and done through the ODBC control panel.

On Unix the configuration is done through an `.odbc.ini` file. The `.odbc.ini` file is usually found in the home directory of the user Challenger was started as. It can also be specified in the `ODBCINI` environment variable.

## .odbc.ini

The `.odbc.ini` file can contain configurations for several databases. The syntax is as follows:

```
[name]
Driver = path
option = value
...
```

*Name* is the name you give the database. *Driver* is the path to the ODBC driver, in form of a dynamic library. *Option* is various options that will be forwarded to the driver. Which options are available differ between different drivers. Usually the location of the database manager and maybe the name and password of the database user.

An example of a `.odbc.ini` file:

```
[toronto_wp]
# white page of metro Toronto
Driver = /usr/lib/odbc/oracle.so
<....>

[netnews]
# NNTP netnews group
Driver = /usr/lib/odbc/nnodbc.so
Server = news.empress.com

[rnd_test]
# data source for R&D test
Driver = /home/r_d/odbc/empodbc.so
URL = empodbc://rnd.empress.com:6322/rnd_test/
testdb
```

## Database URL

The database URL for a database reached through ODBC is `odbc://`**name**`/`, where *name* is the name given the database in the `.odbc.ini` file or the ODBC control panel. It is possible to supply the database user and password in the database URL as well, or put them in the `.odbc.ini` file.

# LDAP

The LDAP directory tags interact with stand alone LDAP directory servers as well as LDAP accessible directories, like Novell NDS or Microsoft Active Directory.

They can be used to create web applications based on data stored in directory, like centralized user administration (for ISP), address book manipulations and so on. The LDAP directory authentication module can enable the Roxen server to authenticate against a LDAP directory.

### Connection attributes

A connection to a directory is determined by the following attributes:

- host
- basedn
- user
- password

*host* is the name of the machine running the LDAP server and *basedn* specifies the subtree of the particular directory tree. The *user* and *password* are used for user authentication in the LDAP server.

### Security Considerations

Your foremost security consideration when it comes to LDAP server is to make sure that only the LDAP operations you intend get sent to the server. This means handling user input in such a way that it can never change the actual LDAP operation. This is done through quoting. The formoutput page in the User manual shows how to do this in RXML.

To reduce your risks, use the access control lists of your LDAP server to make sure Challenger only has permission to do what it actually needs to do. If you use Challenger to provide reports from the directory, then the server should only be able to search the directory, never modify it.

## LDAP module

The *LDAP Module* provides the RXML programmer with four tags for doing LDAP operations; `<ldap>`, `<ldapoutput>`, `<ldapfor>` and `<ldapelse>`. The `<ldap>` is used to connect to an LDAP server and do an LDAP operation like `add`, `delete`, `modify` or `replace`. The `<ldapoutput>` is used to connect to an LDAP server and do an LDAP search operation. The `<ldapfor>` and the `<ldapelse>` tag can only be used within `<ldapoutput>` where the `<ldapelse>` tag is used to return error strings if an LDAP operation fails.

**Defaults...** This page allows to define default values for connection attributes, thus such attributes do not have to be specified in the RXML tags on the pages. However, this only works if one directory is used. Connecting to more directories requires explicitely defined connection attributes.

**LDAP server location** Default in case there is no *host* attribute.

**password** Default in case there is no *password* attribute.

**username** Default in case there is no *user* attribute.

**LDAP search base DN** Default in case there is no *basedn* attribute.

## LDAP user database

Sometimes Roxen Challenger needs to access data about the users. The main reason for this is user authentication, but Roxen can also do other things with the data from the user database, for example, displaying them on a web page.

The LDAP User Database module enables Roxen Challenger to keep such user data in a LDAP directory. The data is stored in objects with the attributes (**as defined in RFC2307**) *uid*, *userpassword*, *uidnumber*, *gidnumber*, *gecos*, *homedirectory* and *loginshell*. These columns correspond to the fields in a UNIX password file, and are the fields that Challenger Authentication modules use. The directory object must contain the *uid* and *userpassword* attributes, whereas theother attributes can be replaced by default values. It is also possible to add extra columns when needed.

**Access mode** This switch sets the authentification mode of the module. The mode can be *user* or *guest*.

With mode is changing several variables will be folded/ unfolded.

**guest** This mode is used for first-time users of LDAP based authentication. The connection to the LDAP server is done by user definitions in the configuration interface within the LDAP server sub menu.

This mode is not recommended for real using! The user defined for connection to the LDAP server must to have read permission to the whole subtree. This is, of course a security risk.

After connection the LDAP server an object corresponding to the search filter (see bellow) is searched and if user is founded his attribute *userpassword* is checked.

**user** The connection to the LDAP server is done as real user his DN is constructed by the following formula:

[LDAP server->bind template] + [LDAP server->Base name]
for example (bind template='uid=%u%' and base name='o=UniBASE Ltd.,c=CZ'):

if user='hop' than
DN='uid=hop,o=UniBASE Ltd.,c=CZ'

If the connection is successful, then if is required existence of some attribute and/or her value, this is checked.

If some attributes aren't retrieved then is used defaults one.

**Access type** The type of LDAP operation used for checking password (Guest mode only) and required attribute (User mode only.

Only 'search' type is implemented.

**Cache entries** This flag defines whether the module will cache the user entries or not. Makes accesses faster, but changes in the directory will not show immediately.

**Close the directory if not used** *Guest mode only*

Setting this will save resources when the module is not used.

**Defaults...**

**Gecos** Default gecos.

**Gecos map** The name of LDAP attribute mapped to gecos field.

**Group ID** Default in case there is no Group ID attribute.

**Group ID map** The name of LDAP attribute mapped to Group ID field.

**Home Directory** Default in case there is no Home Directory attribute.

**Home Directory map** The name of LDAP attribute mapped to Home Directory field.

**Login Shell** Default in case there is no Login Shell attribute.

**Login Shell map** The name of LDAP attribute mapped to Login Shell field.

**User ID** Default in case there is no User ID attribute.

**User ID map** The name of LDAP attribute mapped to User ID field.

**Username add** Setting this will add user name to path to default directory. Mostly used in environment where all home directories have the same parent directory.

**Search template ID** The template used by LDAP search operation as filter for retrieving user object. *%u%* will be replaced by user name.

**Directory connection close timer** *Guest mode only*

How many seconds of inactivity it should take before the directory connection is closed.

**LDAP query depth** *Guest mode only*

Scope used by LDAP search.

**LDAP server...**

**Base name** The distinguished name to use as a base for queries. The value is also used for user DN creating (User mode only).

Typically, this would be an 'o' or 'ou' entry local to the DSA which contains the user entries.

**Bind template** *User mode only*

The template for creating user DN. The *Base name* will be added as suffix.

**Directory search username** *Guest mode only*

This user name will be used to authenticate when connecting to the LDAP server.

**Directory user's password** *Guest mode only*

The password used to authenticate connection to directory.

**Location** Name of host running the LDAP server with the authentication information.

**Required attribute** *User mode only*

The attribute name which must be present for successfully authentication. Can be empty.

**Required value** *User mode only*

The value of required attribute which must be present for successfully authentication. Can be empty.

# SiteBuilder

SiteBuilder is installed by creating a SiteBuilder virtual server, with the *New virtual server* button on the top node of the *Virtual Server* tab.

The second step when installing SiteBuilder is to create the users that are to have access to the content editor. If you already have the users in the user database of the operating system you can do this by enabling and configuring the *AC: OS user import* module. Please see the Access Control chapter for information about how to manually create your users and give them the right permissions.

The third step is to make sure your users can edit files with the editor of their choice. This means using Roxen Application Launcher, which will enable the users to use a local editor running on their computer. On the server end some sort of file sharing between the web server and the user's computers need to be installed and SiteBuilder configured to use it. See the Roxen Application Launcher page for detailed information how to do this.

All files used by SiteBuilder are stored in a separate directory structure. When moving a SiteBuilder site between different servers all you have to do is to configure the *SiteBuilder storage* in the *SiteBuilder* module to point at this directory. The directory structure can be copied to another location, no absolute paths are used.

A SiteBuilder server may spawn several virtual servers. Work areas should be mounted in different virtual servers since the site might contain absolute paths and thus not work when mounted anywhere but "/". To be able to edit through FTP you have to use another virtual server, since all the modules of a normal SiteBuilder virtual server will otherwise show up in directory listings.

When using several virtual servers the *SiteBuilder Work Area Filesystem* and *AC: User database* modules must be enabled, and configured to connect to the appropriate SiteBuilder. The *SiteBuilder*, *SiteBuilder Content Editor* and any Access Control modules but *AC: User database* should not be enabled in such a virtual server. Other modules, like the *SiteBuilder Tags* module can be enabled if their functionality is wanted.

SiteBuilder uses an external program, `cvs`, for the version control functionality. The cvs binary is compiled and shipped with Roxen Platform, you need not notice it.

## Modules

### SiteBuilder

This is the main module of SiteBuilder.

**SiteBuilder storage** The path to the directory structure where SiteBuilder will store its site contents.

### SiteBuilder Content Editor

This module provides a user interface for administration and editing of a SiteBuilder site. It should be enabled in the same virtual server as the *SiteBuilder* module.

**Content editor location** Where in the virtual file system the content editor should reside.

**SiteBuilder** Selects which SiteBuilder the content editor should use.

### SiteBuilder Tags Module

This module contains tags to embed SiteBuilder functionality, such as dynamically generated menus, in your RXML pages.

### SiteBuilder Work Area Filesystem

The file system module used by SiteBuilder. It is either run in *view* or *edit* mode. In view mode all pages are parsed by the template system and RXML before being sent to the user. In edit mode the unparsed pages are shown, it is most often used to make pages available for edit over FTP.

It is not necessary to have a work area in *edit* mode to be able to edit through the Content Editor. Nor is it necessary to enable *SiteBuilder Work Area Filesystem* modules for all work areas. A cookie based system makes it possible for the content editor to show pages on other work areas, through one *SiteBuilder Work Area Filesystem* module.

**Download mode** Sets whether the module should be in *edit* or *view* mode. There are also two special edit modes, *All work areas* and *All sites* that can be used to make all work areas or all SiteBuilder sites available for editing.

**Workarea** The workarea that this file system should access.

**Location** The location of the module in the virtual file system.

### Access Control

Contains the AC system used by other modules to authenticate users to resources. Should be enabled in the same virtual server as the *SiteBuilder* module.

**Administrator interface backdoor** When active, the user and password for the configuration interface also gives full privileges in the access control interface.

**All resources backdoor** When enabled, the user and password for the configuration interface also gives full privileges for all protected resources.

## AC: Basic HTTP and FTP authentication

Performs authentication through HTTP and FTP for Access Control. This module is usually enabled automatically.

## AC: Internal password

Internal user database, that can be accessed through the Access Control system. This module is usually enabled automatically.

## AC: OS user import

Module for using users from the operating system within Access Control. Described in the Access Control chapter.

## AC: AC database user import

Module used for importing users and groups from Access Control systems in other virtual servers.

## AC: User Database

Module for connecting to an Access Control in another virtual server. For use with a virtual server for FTP or to make a work area available. It is also necessary for Challenger tags that handle user information to work.

## Templates

The *Templates* module handles SiteBuilder's template system.

**Default layout template** Does not have any effect in SiteBuilder, but rather when *Templates* is used stand-alone.

**Log errors?** Whether to log errors from *Templates* in the debug log.

## Navigation

The *navigation* module draws graphical menus.

**Cache directory** Where the cache of generated images should be stored.

**Mountpoint** The directory in the virtual file system from where the generated images will be sent.

# FTP

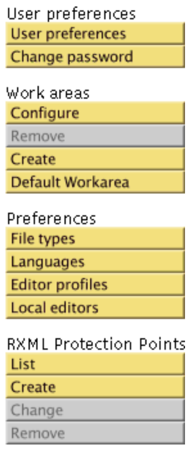
To make your SiteBuilder site available through FTP you need to create a virtual server of the type *SiteBuilder FTP edit* and configure a port using the FTP protocol.

The *AC: User Database* should be configured to connect to the Access Control in the right SiteBuilder server.

The *SiteBuilder Work area Filesystem* module should have its *Download mode* variable configured to *All work areas*. The *SiteBuilder* variable should be configured to the appropriate SiteBuilder site. With this configuration all work areas will be available for edit through FTP.

# Configuration

The *Configuration* tab in the content editor is used for configuration of a SiteBuilder installation.




*Configure options in SiteBuilder*

Under *User preferences* each user can personalize the content editor. See the User manual for more information about these options.

Under *Work areas* work areas are created, removed or configured. It is also possible to set a default work area for editing on this site. See the Work areas page for more information.

Under *RXML protection points* RXML protection points are handled. See the Access control chapter for more information.

Under *Preferences* the following can be configured:

**File types** In SiteBuilder the file type of each file is stored in that file's meta data. The possible file types are configured here, as well as how SiteBuilder should treat files of that file type. This is done by choosing handler for each file type, where the handler is a small module that handles viewing, uploading and downloading of files.

**Languages** The language of each file can also be set in the meta data of the file. Possible values are configured here. Currently the language setting only affects which character set it will use.

**Editor profiles** Editor profiles are part of the support for usage of different editors. The editor profile controls which file types can be edited and how to start the editor. How the editor should find the file being edited is configured here as well. Once the editor profiles are created each user can choose which profile to use. How to configure the editor profiles are described on the Roxen Application Launcher page.

**Local editors** The Roxen Application Launcher for Windows uses the registry to store its configuration. This wizard can be used to create `.reg` files with different configurations. See the Roxen Application Launcher page for more information.

## File types

A SiteBuilder site handles different file types. The file types are presented to the user as a string describing the file type and an icon when doing directory listings. It is possible to use the same icons in file listings done in RXML, by using the `<sb-output>` tag.

File types have handlers, that are plugins that handle upload, download and viewing of the file. It is important that each file type get the right handler, the *text* handler would for example not work very well with an image file. For new file types the *default* handler should be used.

The handlers are:

**default** For any file type that cannot be handled by other handlers. Only the basic SiteBuilder functionality will be available for files handled by the *default* handler.

**image** For handling image files.

**html** For handling HTML files.

**text** For handling any text file. It will be possible to edit with the Content Editor.

**menu** For handling menu files, that should have the content-type `sitebuilder/menu`.

**template** For handling template files, that should have the content-type `sitebuilder/template`.

File types are configured with the *File types* button under the *configuration*. For each file type the following data can be configured:

**Content type** The MIME content type of the file type. If the content type begins with `roxen/` the file will not be handled by SiteBuilder but rather sent unmodified to Challenger. That means that Challenger's *file extension* modules will be used to handle the file. It also means that the correct file extension must be used.

**Name** The name of the file type. Will be presented to the user.

**Image** The image to display in the content editor for this file type.

**Handler** Which handler to use for this file type.

**Download** Whether it should be possible to download this file type or if it should only be handled through the interface in the content editor.

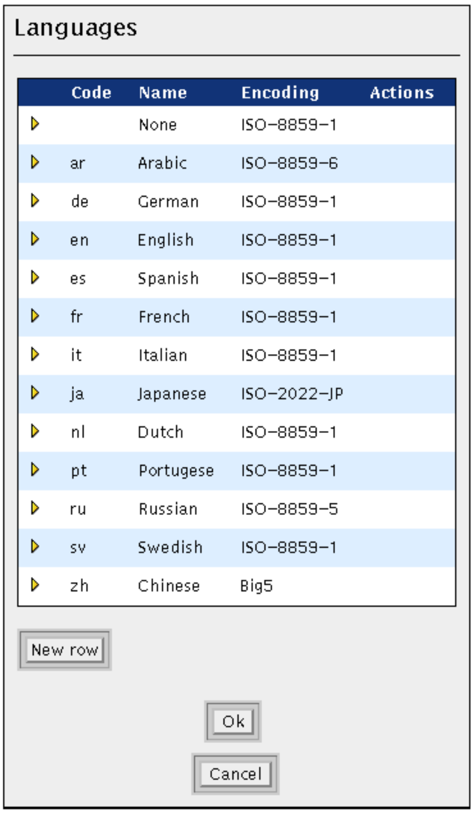
**Parse** Whether this type of files should be run through the template system and the RXML parser. Will only have effect if the handler supports parsing.

**Extensions** The file extensions of this file types. Is used to automatically determine an appropriate file type when uploading a new file. It is by no means necessary for files to have these extensions.

**Default File** This is a path to either a file or directory. The file, or files, will be used as default files in the *Edit new file* wizard. It is necessary to have default files to be able to create new files of the file type from within SiteBuilder.

## Languages

For each file it is possible to set in the meta data what language it is in. This will affect which encoding it is supposed to have. The *Languages* button under *Configure* can be used to configure the languages that should be available for the site.




*Languages wizard*

**Code** The ISO language code for this language. This is also the information you get when from the `<sb-output>` tag.

**Name** The name of the language. This will be used when the user chooses language.

**Encoding** The character encoding used for documents in this languages. Will be sent to the browser.

# Roxen Application Launcher

Roxen Application Launcher is a program that lets the user use any program to edit files handled SiteBuilder. For this to work SiteBuilder must be able to share files with the users' computers. This might require an external file sharing program, like *Samba*.

When the user chooses to edit a file SiteBuilder copies the file to a special shared directory and sends a response to the user's web browser. The response has the content type `application/x-roxen-launcher` and contains the path to the file and the content type of the file. If Roxen Application Launcher has been installed the web browser will recognize the content type `application/x-roxen-launcher` and start Roxen Application Launcher. Roxen Application Launcher will look at the response and launch the appropriate editor program for the content type in the response, with the path to the file as argument.

Several steps are necessary for this to work:

## Installing Roxen Application Launcher

Roxen Application Launcher must be installed on each user's computer. Roxen Application Launcher exists in a Windows and a Unix version. Both versions can be found in the `tools/` directory of the Platform distribution.

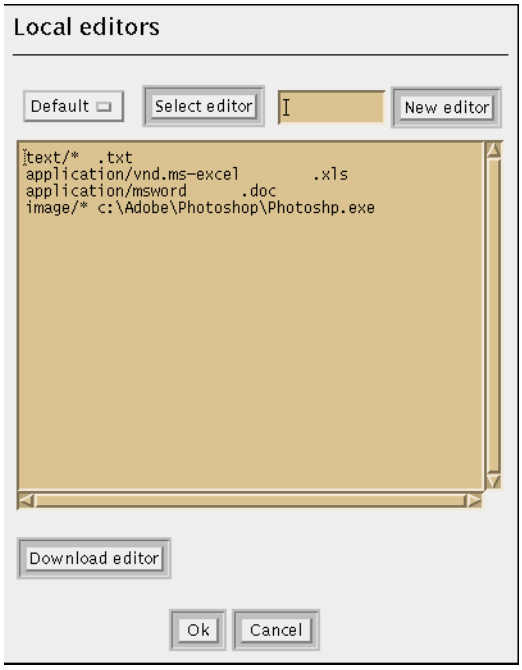
The Windows version is distributed as an self-extracting `.EXE` file. It should install Roxen Application Launcher as well as do the necessary configuration for it to be recognized by any web browser. It is however often necessary to configure it further, see the *Local editors* section for how.

The Unix version is a simple pike script, `tools/roxen_launcher.pike`. It needs to be installed on the Unix system and each web browser must also be installed to recognized it. In Netscape this is done in the Navigator/Applications part of Edit/Preferences. The MIMEType should be `application/x-roxen-launcher`, the application `/usr/local/bin/roxen_launcher.pike %s`.

The Unix version is configured through `.roxen_launcherrc` files. By default the user's home directory as well as `/etc/` will be searched for a `.roxen_launcherrc` file. See the .roxen_launcherrc section for the syntax.

## Local editors

The *Local editors* wizard is used to create registry configuration files for the Windows version of the Roxen Application Launcher. The user needs to download such a file and doubleclick on it to install that configuration.




*Local editors*

The configuration contains a content type followed by either a file extension or the full path to a program. Wild cards can be used to some extent, the content type `image/*` and `*/*` are valid but not `image/g*` nor `*/gif`.

Roxen Application Launcher will match the content type it got from SiteBuilder against the content types in the configuration, one line at a time. When it finds a match it will try to start the appropriate application. If a file extension is given it will try to start the application that handles that file extension. If a path is given it will try to start that program.

The *Download editor* button will download the `.reg` file for the current configuration. Doubleclicking on the `.reg` file will read the configuration into the registry.

The reason that the installation needs to be done in the registry, rather than being part of the editor profile, is to gain security. The web server should not be able to start any program on the user's computer, only the programs that has been configured as editors.

## .roxen_launcherrc

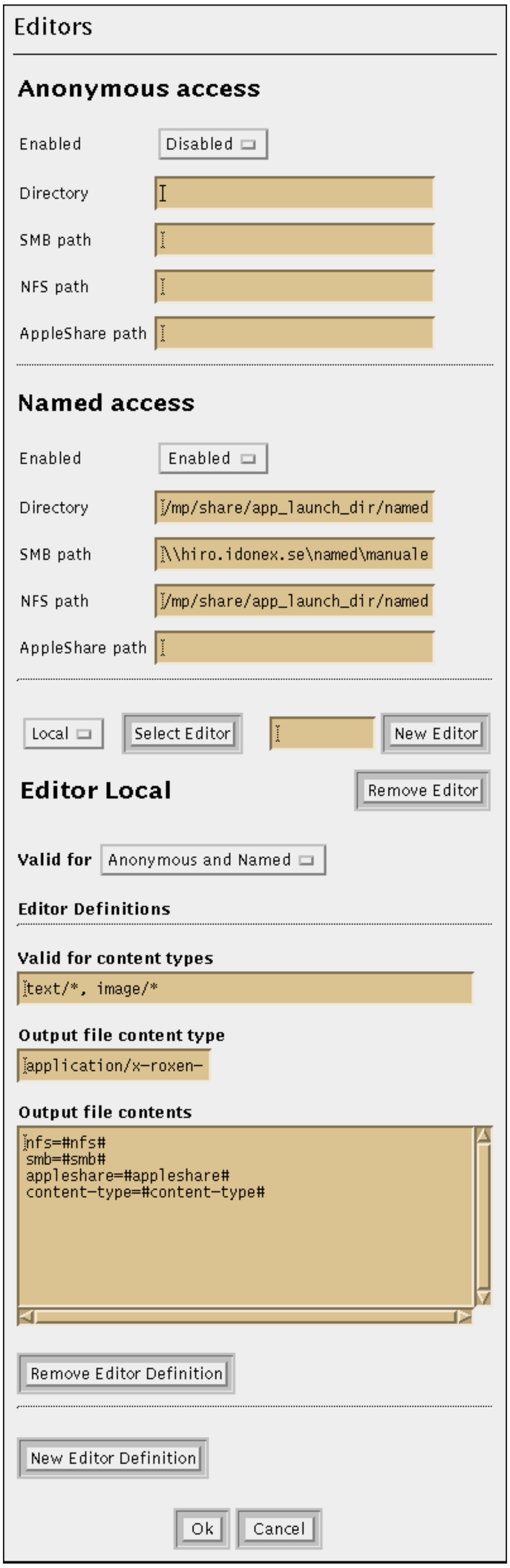
The `.roxen_launcherrc` file is used to configure the Unix version of the Roxen Application Launcher. As with most Unix applications each user can have their own configuration. A sample `.roxen_launcherrc` file can look like this:

```
text/*emacsclient $nfs
image/*gimp $nfs
```

Each line contains a content type followed by the program that should be used to handle that content type. The content type use the same limited type of wild cards as the Windows version. $nfs will be replaces with the path to the file that are to be edited.

## Editor profiles

The bulk of the configuration are done in the *Editor profiles* wizard. Here several different editor profiles can be configured, each user can then choose one of those profiles. It will however usually be sufficient with one editor profile.




*Editor profiles*

The support for local editors comes in two flavors, *anonymous* and *named*. Anonymous is used for users that exist in SiteBuilder, but not on the operating systems user database. Named is used for users that are imported into SiteBuilder from the operating systems user database.

Anonymous support works by writing the shared files to a directory that is not readable to anyone. SiteBuilder will create a subdirectory with each file in. That sub directory will be readable to anyone and the file both readable and writable to anyone. But the name of the sub directory will be known only to the user. It is very important that it is not possible to find out the name of the sub directories by listing the shared directory.

The anonymous support are configured through the following variables:

**Enabled** Whether the anonymous support should be enabled at all.

**Directory** The directory where SiteBuilder will create its sub directories. The directory must be created by the administrator and it should not be possible for anyone to read it, not even the SiteBuilder user. The SiteBuilder user must however have write permission to the directory, so it can create sub directories there.

**SMB path** The path to the directory using Windows file sharing. If running on Unix a file sharing program such as *Samba* will have to be used to make it available. The sharing should be configured to be available without password.

**NFS path** The path to the directory using NFS. Only necessary if the Unix version of Roxen Application Launcher is used.

**AppleShare path** The path to the directory using AppleShare. Some sort of AppleShare sharing software will have to be used. Only necessary if the Macintosh version of Roxen Application Launcher us used.

Named support works by writing the actual files with as user who owns them. This will only work for users that has been imported from the operating system to SiteBuilder. Furthermore SiteBuilder will have to be run as root, otherwise it will not be able to create files as another user.

SiteBuilder will create a sub directory for each user who uses the named support. These sub directories will be created with the correct file permissions so that only the appropriate user has access to it. Within each sub directory a directory structure matching the site will be created.

The named support are configured through the following variables:

**Enabled** Whether the named support should be enabled at all.

**Directory** The directory where SiteBuilder will create its files. Unlike the anonymous support this directory may well be readable by anyone. File accesses will be protected by the operating systems access control system.

**SMB path** The path to the directory using Windows file sharing. If running on Unix a file sharing program such as *Samba* will have to be used to make it available. The sharing should be configured so that each user has to log on with user name and password.

**NFS path** The path to the directory using NFS. Only necessary if the Unix version of Roxen Application Launcher is used.

**AppleShare path** The path to the directory using AppleShare. Some sort of AppleShare sharing software will have to be used. Only necessary if the Macintosh version of Roxen Application Launcher us used.

When the anonymous and/or named directories has been configured it is time to configure one or more editor profiles. Usually one editor profile will be suitable. Each editor profile contains one or more editor definitions. Usually one editor definition is enough. The reason for more than one editor profile and more than one editor definition is to enable SiteBuilder to be user together with thin client solutions like WinFrame or Tarantella.

Each editor profile contains a name and whether it is valid for anonymous, named or both anonymous and named access.

Each editor definition contains the following variables:

**Valid for content types** This is a comma separated list of content types that can be edited by this editor definition. The now familiar limited type of wild cards can be used. Each site will probably need to configure this to reflect the file types used on the site.

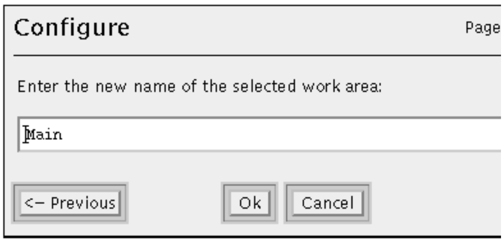
**Output file content type** The content type of the file sent to the browser when the user chooses to edit a file. Should be `application/x-roxen-launcher` if the Roxen Application Launcher is used.

**Output file contents** The contents of the file sent to the browser when the user chooses to edit a file. If the Roxen Application Launcher is used the file contents should be:

```
nfs=#nfs#
smb=#smb#
appleshare=#appleshare#
content-type=#content-type#
```

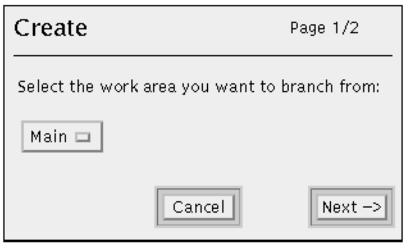
## Work Areas

A work area is a copy of the entire site. Work areas are used for editing new versions of the site without affecting the live web site. When you install a SiteBuilder you get a work area named Main, this is the work area you should use for your live web site. It is possible to rename it with the *Work areas/Configure* button under the *Configuration* tab.




*Renaming work area*

You create a new work area with the *Work areas/Create* button under the *Configuration* tab. When you create the work area you get to choose which work area it should be a copy of. This should usually be the work area used for the live site.




*Creating work area*

Changes made in one work area are transfered to another work area with the *Join* button under the *Files* tab. Either an entire directory structure or a single file is joined.

If your users always use the Content Editor for viewing their pages you don't have to do anything to make a new work area available. When a user uses the *View* button a cookie is set so that she gets the file from the correct work area. If you want to make it possible to view the work area without using the Content Editor you have to mount it somewhere. Either under another path in the virtual file system, or by creating a new *SiteBuilder view area* virtual server.

# Access Control

Access Control is the permission handling system in SiteBuilder, allowing detailed configuration of different users' access to different parts of any SiteBuilder site as well of the SiteBuilder system itself.

Access control handles different users' permissions to certain protection points. A protection point is a resource being protected, for example a directory or the content editor. A user has read, write or no permission on each protection point.

The permissions are indirect. Users have membership in groups, the groups then have permission on protection classes. Each protection class contains one or more protection points. Thus users belong to groups while protection points belong to protection classes.

The access control interface is reached via the *Access Control* tab in the content editor. When selecting the tab, three additional tabs will be visible; *Identities*, *Permissions* and *Resources*.

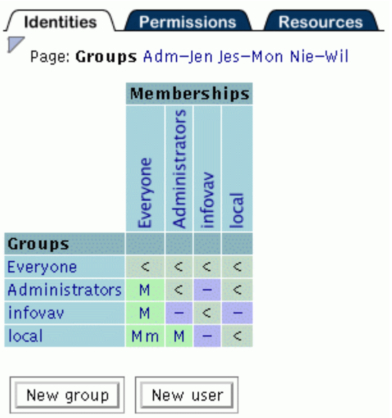
**Identities** The *identities* tab is where users and groups are handled and group membership assigned.

**Permissions** The *permissions* tab is where permissions are set for groups on protection classes.

**Resources** The *resources* tab allows for grouping of protection points into protection classes, as well as configuration of the protection points and protection classes.

## Users and groups

Users, groups and memberships are handled on the *Identities* tab. It consists of a table with users and groups to the left and the groups they can be members of on the top. Not all groups and users are shown on the same page, links on the very top lead to pages that contain a suitable number of groups or users.




*Identities*

Memberships are assigned by clicking on the squares in the intersection between the users or groups to the left and the groups they become members of on the top. A click will change the membership state, which is indicated by a letter. It might take several clicks to circle to the wanted state. The different states are:

| | |
|---|---|
| – | Not a member. |
| M | Explicit membership, set by the user. |
| m | Indirect membership, given because this user or group is member of another group that has an explicit membership. |
| Mm | Both an explicit and indirect membership. |
| a | Automatic membership, imported from another source. |
| ( ) | Explicitly not a member. Used to counteract an unwanted automatic membership. |
| < | Not possible to make this group a member, since two groups cannot be members of each other. The group Administrators cannot be a member of the group local because local is already a member of Administrators. |

### Users

A user in SiteBuilder contains a full name as well as a user name used when to log on. The password is handled by an access control authentication module. Each user can have more than one authentication module or even more than one copy of an authentication module. That way a user could have an additional password by having an extra copy of the *Internal Password* module.

Users are often imported from another source, such as the operating system's user database. If so, the user configurations are done in the module importing the users. See the User databases page for more information.

It is however also possible to create users directly in the access control interface. It is done by pressing the *New user* button, and filling in the user name as well as the user's full name. By default a copy of the *Internal password* authentication module will be added as well. It is used to give the user a password to log in with. It is possible to write the password itself, or give a version encrypted with the unix crypt function. The later is good for sharing passwords between different sites.

Users that are created directly can be mixed with imported users. That way users from within the company could be imported while users from customers are created in the access control interface.

To later change or remove an user, just click on the user name in the user listing on the left.

### Groups

Groups contain members that are either users or other groups. As with users groups can be imported from other sources, such as the operating systems user database, or created directly in the access control interface. Memberships are imported together with the groups, thus making it possible to import the entire security settings from another source. It is still possible to change memberships to imported groups in the access control interface. Such changes will however be local to the access control interface, they will not be exported back to the original source.

There are a special group, *Everyone*. Request that could not be linked to any user, such as a request with no authentication information or a request with the wrong password, will get the permission of the *Everyone* group.

To create a new group press the *New group* button and give a name to the group.

## User databases

Users can be imported to SiteBuilder from different user databases; either from another access control database or from the operating system. There are two modules in the SiteBuilder distribution that can handle this automatically.

### AC: AC database user import

Imports users and groups from another access control database. The imported data is updated automatically in periodic intervals.

### AC: OS user import

Imports users and groups from the operating system using various methods. The imported data is updated automatically in periodic intervals.

**file** Read users, groups and passwords from `/etc/passwd, /etc/group` style files.

**shadow** As file, but passwords are also read from `/etc/shadow, /etc/gshadow` style files.

**ypcat** As `file`, but the `ypcat` program is used in non-privileged mode to get the files.

**niscat** As `file`, but the `niscat` program, available on Solaris systems, is used in privileged mode to get the files.

**getpwent** Read users, groups and passwords using the `getpwent` and `getgrent` system calls. Should work on all systems, but might be slower than other methods.

## Protection points

A protection point is a resource that is handled by the Access Control system. The resource can be a directory, a single file, the graphical interface, the access control system itself or a work area.

SiteBuilder creates a number of protection points automatically when the system is set up or a new work area is created and also when some additional modules are added to the server.

It is also possible to create protection points from the Content Editor both for files and directories and for use in RXML. A protection point for a file is created with the *Add protection point* under the *Files* tab. A protection point for use in RXML is created by the *Create RXML Protection Point* button under the *Configurations* tab.

### Protection points in external modules

Other modules, not included in the SiteBuilder distribution, might also add their own protection points to the Access Control system. The protection points will show up automatically in the Access Control configuration interface when the module is added to a SiteBuilder server.

## Protection points in SiteBuilder

SiteBuilder automatically adds a number of protection points with separate protection classes which are used to control access to the different parts of SiteBuilder.

A typical editor would need write permission for Content Editor, at least one work area and one or more Dir or File classes to be able to create or edit files.

For a public site, the group Everyone needs read permission to a work area and one or more Dir or File classes.

### Access Control

This controls permission for the access control interface. Read permission is required to be able to see the access control information as well as *Access Control* tab itself. Write permission is needed to make any changes to the access control information.

### Content Editor

Controls access to the content editor, the graphical interface of SiteBuilder. To change anything, edit files, changing permissions or settings, in the SiteBuilder, write permission for this protection class is required.

### Dir: /

This controls the access to the files and directories of the site. It is also possible to create new protection points for single files or directories. A protection point for a directory protects all files and subdirectories of that directory, unless a file or subdirectory has a protection point of its own. Thus the protection point for *Dir: /* will protect all files and directories until protection points for other directories and/or files are created.

New protection points for files and directories are created by the *Add protection point* button under the *Files* tab.

The group *Everyone* should have read permission for *Dir: /* on a public site.

### Plugin: IntraSeek

Users with write permission to this protection point will be able to configure IntraSeek under the *Plugins* tab. This protection point will be generated automatically when the *IntraSeek* module is installed and will then copy the permissions form the *Content Editor* protection point.

### Plugin: LogView

Users with read permission to this protection point can view statistics using LogView under the *Plugins* tab. Users with write permission can configure LogView. This protection point will be generated automatically when the *LogView* module is installed and will then copy the permissions form the *Content Editor* protection point.

### Work area administration

To be able to create a new work area or change settings for the current one, write permission is required for this protection class.

### Work area: Main

The *Main* work area is created automatically when setting up SiteBuilder. It controls access to the files in the Main work area. Whenever a new work area is created, another protection class is also created for it.

The protection point for a work area is combined with the protection point for a file, to find out what permission a user has to a file in a certain work area. The combined permission equals the lowest permission of the two combined protection points. Thus a user will have read permission to a file in a work area she has read permission to, eventhough she has write permission to the protection point protecting the actual file.

Editors need write permissions for at least one work area to be able to edit files.

To make a site accessible in public the group *Everyone* needs read permissions to the work area containing the live site. Which files will be accessible is controlled by Dir: and File: protection classes.

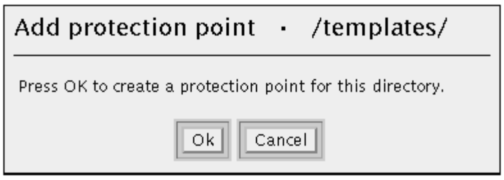
## Adding protection points

In addition to the protection points created automatically by SiteBuilder it is possible to create additional protection points.

### Protection points in file structure

Protection points can be created for directories or single files in the virtual file system of SiteBuilder.

When zoomed in on a directory other than "/" or a file the button *Add protection point* will be available. Clicking on the button and then confirming the action will result in the creation of a new protection point.

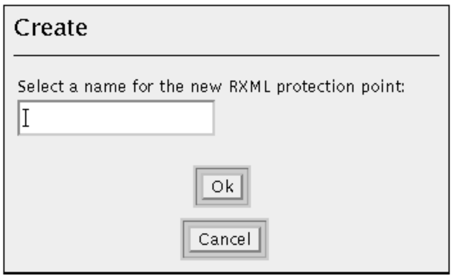



*Adding protection point*

The permissions for then new protection point can be configured in the Access Control interface. By default they are the same as for the parent directory.

### Protection points in RXML

It is possible to create protection points to be used inside the content and template files. They are handled through the *Configuration* tab in the Content Editor.




*Creating protection point*

Such a protection point can be used together with a `<sb-if>` tag placed around restricted information included in a web page.

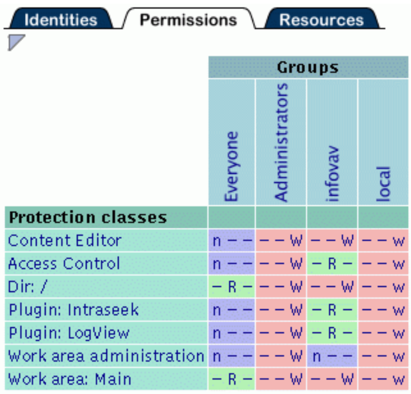
## Permissions

Permissions are set between groups and protection classes. All members of a group will inherit the permissions of the group for each protection class.

If a user is a member of several groups and the groups have different permissions for a protection class, the user will have the most privileged permission.

There are three kinds of permissions to be set between a group and a protection class.

| | |
|---|---|
| n/N | No rights |
| r/R | Read permission |
| w/W | Write permission |




| Protection classes | Everyone | Administrators | infovav | local |
|---|---|---|---|---|
| Content Editor | n – – | – – W | – – W | – – w |
| Access Control | n – – | – – W | – R – | – – w |
| Dir: / | – R – | – – W | – – W | – – w |
| Plugin: Intraseek | n – – | – – W | – R – | – – w |
| Plugin: LogView | n – – | – – W | – R – | – – w |
| Work area administration | n – – | – – W | n – – | – – w |
| Work area: Main | – R – | – – W | – – W | – – w |

*Permissions*

### Setting permissions

Permissions are set by clicking one of the three columns. Clicking on a column will toggle permissions between:

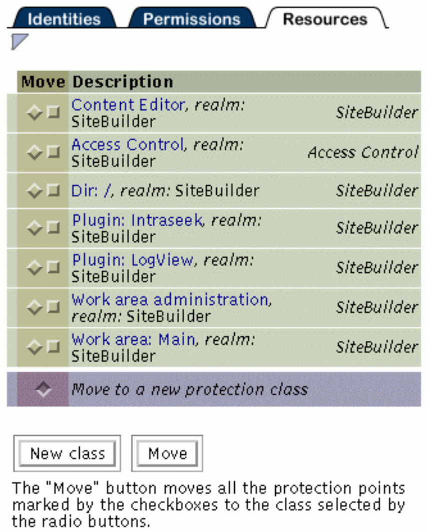
| | |
|---|---|
| – | Not set |
| lower-case letter | Inherited permission, if present |
| capital letter | Direct permission |

# Protection classes

A protection class is essentially a group of protection points. They are used to simplify handling of several protection points that should have the same permissions. A protection class will however often contain only one protection point.

## Resources

On the resources page the protection classes can be configured. Protection points can be moved between protection classes.




*Resources*

# IntraSeek

This manual describes how to install, configure and maintain a Roxen IntraSeek search engine both inside a SiteBuilder environment and outside. It is meant for system administrators only.

### Administration

The main task of the administrator is to provide the users with the services they need. This is done by configuring IntraSeek into numerous profiles based on the users specific need. It is also important that a search engine isn't launched onto a site when traffic is at it's peak.

### Concepts

**Profile** A unique combination of:

- What to search
- Where to search
- When to search
- What to avoid

**Crawler** The part of the search engine that is searching the world wide web for documents matching the profile. Robot is an other word for crawler.

## Directories

**Crawler.pmod/** The Pike source to the crawler and its components.

**Crawler.pmod/filters/** Filters for different document types.

**crawlerid/** The Id's of the present crawler configurations.

**databases/** Stores the different search databases which is the result of a crawler search.

**modules/** Modules that handles interaction between the user and IntraSeek.

**resource/** Information on how to parse the different languages IntraSeek supports.

## Configuring

To configure IntraSeek inside Challenger focus on the *Virtual Servers* and select the server on which you've installed IntraSeek. Most of the time there is no need to change many of the IntraSeek module variables, unless any of the special cases described below occurs. Here is a listing of the module variables.

**Admin Password & Admin user name** The password and name you use to access the Intraseek graphical configuration interface. The default values are the standard roxen administrators user name and password. Will by default be the same as the Challenger configuration interface user and password. You can change these values, or simply keep them as they are.

When used in SiteBuilder a protection point in access control is used instead.

**Administrator email** Should contain your own mail address. It will be used by the crawler when presenting itself to the web server. If you wish to be be anonymous it can be left empty. This email address will not be on any web page, but only in the HTTP protocol request header.

**Engine home location** Should point to where you have copied your IntraSeek files. This should default to the correct directory.

**Mount point for config** Where IntraSeek mounts on the server URL. May be left unchanged.

When used in SiteBuilder IntraSeek is located under the *Plugins* tab instead.

**Profile config location** Is by default the same as *Engine home location*. If you plan to run IntraSeek on only one virtual server, this is fine.

If you want to share the same IntraSeek code and base directory for several virtual servers, you should change the *Profile config location* path. The path points to a location where the profile configurations are saved. Several virtual servers can share the same *Engine home location*, but if *Profile config location* is left untouched, they will *all* use the same profiles - which, for example, means that they will all try to autolaunch crawlers at the same time. More on this on the Advanced profile configuration page.

If you are using a port other than port 80, make sure that the Server Variables/Server URL is correct and that it contains the proper port as well.

### Go to IntraSeek

When the Challenger configuration is done direct your web browser to IntraSeek's dedicated graphical configuration interface. If you have a SiteBuilder server running focus on the *Plugins/IntraSeek* tab. If you don't have a SiteBuilder click on the link *Go to the administration page* inside the IntraSeek module configuration interface.

## Creating new profile

Profiles defines how and which web pages and servers are to be indexed by the crawler. To create a new profile select *New profile wizard* and follow the online instructions.

Below, the basic configuration variables for a profile is described, while the more advanced variables are described later, on the Advanced profile configuration page.

**Profile id** A unique identification for the profile. It should be a short identifying text, and must not contain any spaces. For example, the id could be: `my_profile`.

**Profile name** The contents of the profile name will be seen on the selection tab on the search page shown to the outside world. For example, this could be *My test search*.

**Activated** Should be left at *yes* for now.

**Storage directory** Is a search path in your file system. Ends with a "/". Intraseek has automatically created a special directory for storage of the databases, but you can change this to any path in the file-system.

**Working directory** Is a search path in your file system. Ends with a "/". This is where data from the crawlers' data gatherings will be stored. Due to nature of the workings of the data base, it is advantageous for this to be situated on a fast disk, This will increase the speed of the process by several hundred per cent.

**Startpages** Where you specify a set of pages for the crawler to start at. It is usually sufficient to state the URL of the main page of the site you are about to index, since an IntraSeek crawler will follow all links it finds. Separate the various URLs by putting them on separate lines. For example: `http://my.server.com/~sysadm/`

**Accept pattern** Specifies which pages are to be accepted by the crawler. There are some very important things to consider here:

**1** Always limit the crawler to stay within your site. If you don't, it will, without any warning, crawl out on the worldwide web.

**2** Since the accept and avoid patterns really are regexps, they should read `^http://www.foo.com/*` instead of `www.foo.com/*` if you want to make sure not to index `http://gazonk.www.foo.com/`.

**3** Separate the various accept patterns by putting them on separate lines. For example, this could be `my.server.com/~webmaster/*`.

**Avoid pattern** Specifies what sort of pages the crawler will avoid. Already specified are file types that contain information the crawler can't index. If inappropriate, these may be removed in order to have the crawler index these file types.

For example, if you specify `*/~webmaster/non-public/` here, the crawler will avoid `~webmaster/non-public/` on all servers. If you specify `*my.server.com/~root/*`, `/~root/` will not be indexed on the server `my.server.com`.

Remember to check arguments to CGI scripts and the like. For instance, directory listings can sometimes enter infinite loops. If any such are present, it is recommended that `*?*` be added here.

Check up on the crawler while it is running, by checking its log file, so that it doesn't go into a loop, run amok, etc.

Finally, on the last page of the *New profile wizard* pages, press *OK* to save the new profile. Technical notes: all profiles are saved in the text file `ENGINE_HOME/profiles.txt`. If no id is specified, a new unique id will be generated.

## Indexing

Select the *Crawlers* tab at the top of the page to enter the Crawler Control page. From this page, you can start and stop crawlers, and view their current status. First on the page is a list of crawlers ready to start, below this is the list of active crawlers.

**o** At this point your test profile should be in the list of ready crawlers. Select *Launch Now* to start a crawler.

**o** Now, your profile should appear in the list of *Active Crawlers* with the status *starting up*. Click *Reload* on your browser to update this information.

**o** After a while (Click *Reload* a few times, every few seconds) the status of your crawler will change to *Running!* and another few options will appear.

If this does not happen within a few seconds, something has probably gone wrong. The most likely cause is that Pike failed to start. Check:

**o** That you have specified the correct *Engine home location* path in the Intraseek Module Configuration.

**o** Challenger's debug log for any hints about what went wrong.

If you find the error, click *Zap!* to return the crawler to the *Ready to launch* list.

**o** If all went well, the crawler is now running. You can select either *View status* to see a summary of the progress, or *View log* to see a more verbose log.

**o** If you select *Zap!*, the crawler will be killed by force, and the entire data gathering aborted. The crawler will then be *Ready to launch* again, and none of the information gathered will be used. Any temporary files will also be deleted.

**o** If you select *Halt!* the crawler will instead be stopped. It will then save all its information to the data bases, and also save a freeze-file, `ID.scheduler_freeze.is`, so that the data-gathering can be continued. It will say *Halting... (saving)* in the status box, and then *Halted* when everything has been saved.

### Afterwards

When the crawler has finished running, and indexed all the available web pages, two data bases with the names `n.ID.pages.yabu` and `n.ID.index.yabu`, and a flag-file `ID.new.flag` will have been created, where ID is the name of the profile.

These will be renamed to `a.ID.index.yabu` and `a.ID.pages.yabu` when the search engine finds the flag-file whereupon the flag-file will be deleted and the new data bases used.

More information on these is to be found in the Storage of Data Bases chapter of the technical documentation.

You will notice a slight pause the first time you use the new data base. The reason for this is that the IntraSeek module replaces the old data base files with the new ones. This delay can last anywhere from seconds to minutes depending on file system speed and data base size.

Logs have been generated as well. Select the *Logs* tab in the configuration interface to see these. The main crawler log tells you how different crawlers have been started and stopped. This log can be cleared by selecting *Delete the main log*.

### Standards

IntraSeek supports the Robots Exclusion Protocol which is a method that allows Web site administrators to indicate to visiting robots which parts of their site should not be visited by the robot. When a Robot visits a Web site, say `http://www.foobar.com/`, it firsts checks for `http://www.foobar.com/robots.txt`. Within this document it is stated if the crawler may retrieve the document or not.

We do however recommend the use of `<meta name=robots>` instead, as this is more flexible. To maintain the "/robots.txt" file, it is necessary to have root privileges, while any HTML writer can control the META method locally.

### Frames & indexing

Frames are a big problem for all search engines. Most site maintainers optimize their web pages for the latest versions of Netscape and MS Internet Explorer, ignoring the `<noframes>` options. Right now, no search engine on the market can follow frames and reconstruct the exact view (frame set). It is very difficult, if not impossible, to keep track of all dynamic frames and frame sets, and if JavaScripts are used to generate links, it becomes even harder.

IntraSeek follows links found in frame sets. This is an acceptable solution, although not a good one.

## Languages

The IntraSeek output in search forms and summaries can be translated into several languages. It should be noted that IntraSeek does not translate the search result summaries but only the static IntraSeek output information like buttons, summary status, etc. See the IntraSeek tags pages how to use the language support.

The languages in the table below are supported by default in IntraSeek.

| Country | Code | Country | Code |
|---|---|---|---|
| Danish | da | English | en |
| Finnish | fi | French | fr |
| German | de | Hungarian | hu |
| Italian | it | Lithuanian | lt |
| Norwegian | no | Portuguese | pt |
| Romanian | ro | Slovenian | sl |
| Spanish | es | Swedish | se |

### Character sets

IntraSeek has additional support for five charsets, of which iso-8859-1 is the default one.

**iso-8859-1 (Latin 1)** Covers Albanian, Catalan, Danish, Dutch, English, Faeroese, Finnish, French, German, Irish, Icelandic, Italian, Norwegian, Portuguese, Spanish, and Swedish.

**iso-8859-2 (Latin 2)** Covers Czech, German, Hungarian, Polish, Romanian, Slovak, Sorbian, Slovenian, Serbian, Croatian.

**iso-8859-3 (Latin 3)** Covers Esperanto, Galician, Maltese, and Turkish.

**iso-8859-4 (Latin 4)** Covers Estonian, Latvian, and Lithuanian.

**iso-8859-9 (Latin 5)** Covers the Same as Latin 1, except that Turkish is covered instead of Icelandic.

It is important for the IntraSeek crawler to know which character set a document is using for the purpose of being able to supply the users browser with a proper query response. One problem is the lowercase representation of words. If for example the user is searching for *FOO* or *FoO* he should still get the same results, so to make a correct lowercase representation of international characters the crawler has to know which character set to use.

To make sure that documents are delivered with the correct content type several methods can be used. The `<meta>` tag proposed by Netscape is one means but we recommend using the RXML tag `<header>` as it adds the content type information to the HTTP header, making it less browser dependent than the `<meta>` tag. The `<header>` tag should be placed within the `<head>` container. Remember that when writing the name of the charset it should always be stated in lowercased letters.

## Logs

IntraSeek will create a few log files as it works. These can all be seen from the IntraSeek graphical configuration interface under the *Logs* tab.

### Main logs

First of all, it is possible to view the main crawler log. It contains some general information on when crawlers were started and how their searches ended. Delete it when it grows too big.

Below the main crawler log you will find the main engine log. In this log you can see which profiles are mounted, and when they were mounted. Every time the search module is restarted, something will appear here. Delete it when it grows too big.

## Query logs

The query logs are found below the main engine log, in a new table with an entry for each profile. You can view the complete query log for any profile, as well as get a top 100 list of the most popular search terms. If you want to, you can delete these logs as well when they grow too big, unless collecting statistics for the most searched terms interests you.

Note that error messages generated by the crawler when something goes really wrong appears in the Roxen debug log as well as under the Roxen *Event log* tab. Such errors will not be written to any of the IntraSeek logs.

## Site structure logs

As IntraSeek sends its crawlers wandering through your site, it records all erroneous or strange links it finds and creates a log from this. One log is created for each profile.

For example, you can find information like this in the log:

---

**Error: These URLs couldn't be found:**

---

1. http://www.compa.com/products/webhotel/jenspage.html:
   – Linked from:
   http://www.compa.com/products//webhotel/
   – Linked from: http://www.compa.com/

In this case, IntraSeek has found a link to `http://www.compa.com/products/webhotel/jenspage.html` that it couldn't follow. In this example, if we look at the two web pages mentioned as "Linked from" we find a link `jenspage.html` on each of them.

Faulty links are not all that is recorded in this log, IntraSeek also reports web connections that failed, pages that were reported as forbidden or documents containing non-indexable data such as binary files.

# Advanced profile

This page contains the more advanced configuration options not mentioned on the *New profile page* are described.

**Activated profile** If *active* the intraseek engine will try to mount the data base at startup, and it will also be scheduled for automatic crawler launches.

**Automated update of the data bases** Defines at which intervals the data bases should be automatically updated. If you want the crawler for a profile to be automatically launched at certain intervals to keep the index data bases up-to-date, you can use this function. The selections should be rather self explanatory. If you select *Never!* at the first selection, the two following, *days* and *time*, will be ignored and crawlers will not be automatically launched.

For more status reports on scheduled crawlers, look at the *logs* page.

**Crawler log detail level** Selects how much information the crawler should write in the log.

| | Fatal Errors | Errors | Warnings. Reports. Scheduler info. | Rejects. Accepts |
|---|---|---|---|---|
| **Full (Default)** | Yes | Yes | Yes | Yes |
| **Medium** | Yes | Yes | Yes | No |
| **Short** | Yes | Yes | No | No |
| **None** | Yes | No | No | No |

| | Fatal Errors | Errors | Warnings. Reports. Scheduler info. | Rejects, Accepts |
|---|---|---|---|---|
| **Full (Default)** | Yes | Yes | Yes | Yes |
| **Medium** | Yes | Yes | Yes | No |
| **Short** | Yes | Yes | No | No |
| **None** | Yes | No | No | No |

**Crawler walk pause** The crawler is quite fast. If you index pages outside your own net, you should slow it down somewhat by changing the *Crawler walk pause* between each document download, which is the number of seconds the crawler will pause before fetching the next document. This keeps it from causing large loads on the web servers of others.

**Crawler nice increment** Sets the Unix nice level of the crawler process.

**Stop lists** Specifies a lists of words that are so common that they should not be indexed. Several stop lists can be specified. Select none, one or more depending on which languages are used on your site's pages. The stop lists are stored as ordinary text files in the directory `ENGINE_HOME/resource/`.

**Additional stop words** Indicates extra stop words that should be filtered. If *Yoyodyne Productions* is present on every one of your pages, it may be a good idea to specify *yoyodyne* and *productions* here. The disadvantage is that it will not then be possible to search for the words yoyodyne or productions, the advantage is that the data base files will be smaller, and searches faster. For further technical details on this function, check the memory usage section.

**Query Logs active** If enabled queries will be logged to disk, for top 100 statistics, and such.

**Safety save** This value says how many pages a crawler will go through before automatic saving and reorganization of its data base. For further technical details on this function, check the memory usage section.

**Max documents to download** Specifies the maximum number of pages the crawler will index. It is a good idea to specify a maximum here. In case something should go wrong, you avoid having the entire partition filled by a huge data base. Going wrong usually means that the robot has become lost on the Internet, due to erroneously written accept and avoid patterns.

**Crawler page fetch Timeout** Defines how many seconds will pass before the download of a page will be aborted. For example, if a crawler can connect to a page, but doesn't get anything from the web server in the other end, it could patiently wait for information - forever, if it hadn't been for this setting. Enter how many seconds you allow the fetcher to do its work.

**Site structure logging** Creates logs of web site errors and warnings. If active, site structure logs will be generated for this profile. See the logs chapter for information on site structure logs. If you are not interested in the site structure log, you can turn it off here and save crawler time consumption, space on disc and memory. You will benefit from less memory usage by the crawler, and avoid logs that take place on disc. The operations controlling the log will be disabled as well.

**Max size of query logs** Is specified in bytes. When a query log exceeds this size, it will be moved to a `.bak` file. The old `.bak` file will be removed.

**Number of max quick links displayed** If you have a search resulting in several hundred pages, a list of several links to the next pages of result will be displayed below the list of summaries. The *quick links* referred to here, are the maximum number of links to show.

**Number of documents summaries** Defines how many search summaries to display at every page.

**Quoted search enabled** If set to *Yes*, the users of the search engine can use quotation marks to search for a phrase. For example, a search for "John Carl Smith" will search for persons with this name. Without quotes, the search would return any pages that use any of those common names.

Note that an extra data base will be used to store the extra information, if this setting is enabled. With the current implementation of full text searches we cannot guarantee good performance for data bases covering more than 1000 documents. If you have more documents, turn this option off, or IntraSeek can sometimes get stuck with heavy calculations for several seconds.

**Wildcards enabled** If enabled, the users of the search engine can use quotation marks (?) and asterisks (*) to broaden searches. A search for *net** might match netscape, nethack, network and so on. A search for *int??net* matches intranet as well as internet.

Note that IntraSeek requires that the user specifies at least three characters in front of the * notation, and that there is no distinction made between lower- and uppercase searches. Also note that an extra data base will be used to store the extra information, if this setting is enabled.

**Summary text length** Is the length of the summaries displayed along with the search results and the link and the hit percent. If used on the web page, the Meta description will be used for this, otherwise the first part of the document becomes a summary.

# Technical document

This a short introduction to some of the technical aspects to IntraSeek If you are not interested in the technical questions of the inner workings of IntraSeek you can comfortable disregard this chapter.

## Storage of databases

IntraSeek currently uses Yabu as the data base handler. Also note that no logic is placed in the database, all the boolean search operations, tree structure calculations and wildcard matching is done by IntraSeek. The database is only used in a basic way, just to store data, and to retrieve it.

When collecting new information the crawler uses a separate data base. The reason is that it should be possible for the web users to search even while the crawler collects new information. A flag file is written to tell the search engine to swap the new data base in when it is finished, overwriting the old one.

## Memory usage

The memory used by Pike and IntraSeek varies a great deal, depending on your operating system.

However, the more pages IntraSeek collects, the more memory it uses, as it keeps the site structure, its errors and two stacks in memory, which pages to visit, and which have been visited.

The index of words is also kept in memory, but written to disc at certain intervals, called *safety saves*. A safety save dumps the index to a yabu data base, then clears the memory it used. Also, the disc data bases are reorganized to keep them down in size. When a reorganization is running, you can notice files that start and end with the "#" in the temporary storage directory.

By default, these saves occur every 500 pages. You can lower this if you run into memory problems. However, you shouldn't increase this, if you do, you will get a structure data base that grows faster and faster, and the crawler will consume more and more memory.

If you are not interested in statistics for the site, the log with broken links and such, you can disable this feature to gain memory. To do this, go to the profile configuration and change the variable *Site structure logs* to *no*.

To limit the size of the index both in memory and on disc, stop lists are supported. A stop list contains short, "meaningless" words that are filtered out. For example, in the English stop list words like "the", "and" and "it" appear. Use one or several stop lists covering the language you usually use on the pages you run the crawler through.

*Max download per document* defines, in characters, how much of a document that should be downloaded. This is used to limit index size. Normally, if you download the 100000 first characters of a document, it is very likely that there are enough terms to cover the content of the document. This will of course vary depending on the type of information present on your site. If all words are important (even those at the end of large documents) you should increase this value to higher values, for instance, 999999.

Log files are stored on disc and also take up some memory. Make sure to delete them every now and then by using the configuration interface.

## Rejects and accepts

Reject and accept patterns are unique for every profile. By default, the reject pattern comes with a lot of standard avoids, mainly to reject files ending with `.gif`, `.gz` and so on. The *Accept pattern* is by default empty.

The reject and accept rules are applied when IntraSeek is about to schedule a new URL for visit.

**1** First, IntraSeek matches the reject patterns. If any of them match, IntraSeek will not visit the URL, and no further match-checks are made on this URL.

**2** After that, the accept patterns are matched. If any match is found, the URL will be accepted, and no further match-checks are made on this URL.

**3** Finally, if the URL was neither rejected or accepted, IntraSeek will reject the URL by default.

# LogView

This is the administrator's part of the LogView manual. This part contains information of how to install and configure LogView, but to increase the understanding of these subjects, we start out with a deeper survey over the LogView system.

### Internal and external logs

LogView handles two types of logs; logs that LogView collects itself, one for each virtual server it is installed to, and log files in Common log format that are created in other ways and imported to LogView. The first kind contains the most information and can therefore be used to create more complex statistics. All logs that LogView creates are compressed to save disk space.

### Statistics groups

A statistics group consists of one or more logs as well as a configuration of statistics functions. Statistics groups makes it possible to view statistics from several log files, for example if the web site is handled by several web servers. It is also possible to configure exactly what statistics should be calculates, by choosing which statistics functions to enable.

### Databases

LogView keeps a database for each function enabled on each statistics group. In these databases, LogView stores information that has been extracted from the logs by the functions. For example, if the function *Most common browser* is enabled on a group, a database is created to store the number of hits per used browser type. Inside the databases, the data is divided into different tables per a time unit, most often per day or hour. In this way, when the user requests a specific kind of statistics for a specific group and time interval, LogView can easily get the information needed to calculate the statistics from the database.

### Updates

Once a day, once an hour, or when the administrator uses the *Update* command, LogView performs an update operation. This has effects on both logs and databases.

All imported external logs are checked, and the last hour's hits are appended to a corresponding internal LogView log. The logs created by LogView itself are not affected, since they are updated automatically after every access to their virtual server.

The databases of the statistics groups are updated with the data gathered during the last hour, making the new data visible in the statistics.

## Installing

LogView comes with the Roxen Campaign Suite and Roxen Platform distributions, and is installed automatically. In Challenger Pro you will have to enable it manually on each virtual server LogView should handle statistics for. For Roxen Platform LogView is installed automatically when you enable a SiteBuilder virtual server.

When the module is enabled the next step is usually to go to LogView configuration interface. In case of Challenger Pro this is by default found on `http://virtual.server.name/logview`. For Platform it is found under *LogView* in the *Plugins* tab.

However, it is a good idea to check if the default values need any modification. This is done by selecting the *LogView* option that has now appeared in the list of installed modules.

In the following sections, the different options under the LogView module configuration are explained.

**Admin password** Here the password for LogView administration can be changed. By default, this value is the same as the password used for Challenger administration.

Note: The number of the stars symbolizing the password is always four, regardless of the actual number of letters in the password.

**Admin user name** The user name of the LogView administrator, by default the same as the Challenger administrator.

**Enable <logview> tag** If disabled, the LogView tag will only work inside the LogView user interface pages. If enabled, the tag can be used on any page on the virtual server, as described in the section *Putting reports on other pages* in the User's manual. The default value is *No.*

**Log path** Specifies the directory where LogView will put its own logs. The default value is

`../logs/new/logview.data/`

*Note:* Please use *different* directories for every LogView installation. Otherwise the state of the internal LogView files will become unclear and LogView will fail to operate correctly.

**Minimize load** If set to *Yes* LogView will try not to run updates concurrently even if it is used in several virtual servers.

**Mount path** Specifies the place in the virtual file system of the server where the LogView user interface can be accessed. The default value is `/logview/`.

**Update hour** Which hour LogView should update the statistics groups. Can be set to *Every hour* if the statistics should be updated each hour rather than once every day.

**Update nice value** The nice value of the update process. If set to a higher value the update process will not be as efficient at getting system resources as other processes running on the computer. Often it is more important that the actual web server gets system resources than the update process, and in that case it makes sense to give it a high nice value.

**Viewer password** Specifies the password needed to access the user interface, if password protection is enabled.

Note: the number of the stars symbolizing the password is always four, regardless of the actual number of letters in the password.

By default, this value is empty.

**Viewer user name** If this field contains a name, the user has to log in with this user name and the password specified above. Only one user can be specified.

By default, this value is empty.

# Configuration

As mentioned earlier, LogView is installed with a default configuration. If this configuration needs to be changed, this is done via the LogView administration interface. This is a part of the LogView user interface that can only be accessed by the administrator.

If LogView is run as a part of Roxen Platform, the administration interface will be found under as a *LogView* tab under the *Plugins* tab in the Content Editor.

The administrator of the Challenger server can also reach the configuration interface at the URL pointing at the subdirectory `admin` under the LogView directory itself;

`http://virtual.server.name/logview/admin`

The password used here is the same as for the configuration interface of the Challenger server.

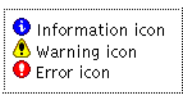
# Event log

On the Event log page, two kinds of logs can be viewed; the *Admin log* and the *Summary log.*

## The Admin log

This is a log of events originating from the administration pages, such as the administrator logging in and configuration changes being made. The log also contains messages about any errors that have occurred.

The informational messages in the log are preceded by an information icon, while the warning and errors messages are preceded by warning and error icons respectively:




*Icons*

## The Summary log

The Summary log is much more detailed and contains a summary of automatic LogView activities, such as importing logs and updating statistics databases. This log is mainly useful for tracking down errors.

# Configuration page

In this page, a number of aspects of LogView's behavior can be configured. All of these aspects have a section below.

## Update

Here, the administrator can manually force an instant update of the statistics gathering, which otherwise takes place once an hour. This can be used for presenting really fresh statistics, or for testing purposes. *Update status* shows whether the update process is already running at the moment, and if so, its process number.

## Logs

A wizard guides the administrator through the steps needed to configure the logs.

**1 Choose a log to configure** By default, LogView is configured with one log, the log of the virtual server that LogView is installed on. This log has the same name as the server. More logs can be added using the option *Import logs* below.

**2 Configure Status and Log name**

**Status** States whether the information from this log should be updated or not. Internal logs are *Active* or *Disabled*, external are *Imported* or *Disabled.*

**Log Name** The name used to refer to the log, for example when configuring statistics groups.

**3 Configure miscellaneous attributes** Some of the attributes use Pike regular expressions, see Appendix C for the syntax.

**Download files** Served files with names matching this regexp will be counted in the statistics over downloaded files.

**Page regexp** Served files with names matching this regexp will be counted in the statistics over served pages.

**Query filter regexp** This regexp is used for filtering the query part of a request in the *popular pages with queries* function. Only the query variables that matches the regexp will be displayed.

**Server alias** This regexp is used for separating internal links from external when keeping track of the visitor's moves using the *Pathways* functions.

**Server domain name** This value is used when importing Common log files, which does not always specify the domain of the server.

**Server URL** This URL is used when creating links in tables, for example links to the pages in a table over popular pages.

## Statistics groups

The statistics group are configured in another wizard.

**1 Select the group to configure** A new group can be added by choosing that option and filling in the name.

**2 Specify the logs to include in this group** These logs are divided into:

**o** logs from the current virtual server, either the server's own log or imported logs

**o** logs from "remote" virtual servers on which LogView is also installed

In the list, check the logs that should be included.

**3 Choose the kind of statistics to collect** Just select the function groups that are of interest.

**4 Fill in some parameters:**

**Max table size** This restricts the amount of data to be stored in the database for each of the functions enabled on this statistics group, to save disk space and computational time.

**Ignore hosts** Here, a regular expression matching hosts that are to be ignored in the statistics can be specified.

**Ignore pages** A regular expression matching pages that are to be ignored in the statistics. This is useful in intranets where statistics are wanted from all hosts excluding certain pages i.e. LogView's.

**Example**

The sites

```
http://rhino.ceros.com/
http://hippo.potamus.com/
```

are located at the sister companies Ceros and Potamus. The administrators of the sites only want statistics of hits from outside these two companies. This is done by setting Ignore hosts to the value

```
(ceros\.com$)|(potamus\.com$)
```

## Import logs

Using this wizard, the administrator can specify existing logs in the Common log format to be imported into LogView.

**1 Common Log Format file name**
Specify the full path to the log file.

**2 Log name**
Specify a unique name for this log.

**3 Fill in values for a number of log attributes**
Follow the same procedure as in the *Logs* section above.

**4 Select a statistics group to contain the new log**
A new statistics group can be added by selecting that option and specifying the name.

**5 Configure the statistics group**
This is done as described in the section *Statistics groups* above.

## Delete logs

After selecting the log that is to be deleted, the administrator has the choice of keeping or deleting the log configuration. Selecting to keep will only delete the contents of the log, and LogView will continue to extract statistics from the log in the future. Selecting to delete will make LogView forget about the log altogether, as well as delete the log file.

## Rebuild/delete statistics group

After selecting the appropriate group, the administrator has the choice of only deleting the gathered data or the entire group, just as was the case with the logs in the previous section. Selecting Rebuild will only delete the data and then rebuild it from the logs. This could be used when a new log is added to the group, and the statistics for the past up to this point has to be recalculated. Selecting Delete will delete the group as well as the data.

## Report maker

A report is a page containing LogView tags tailored by the LogView administrator. Here such pages can be created.

**1** Select the report to edit.

**2** Edit the RXML code and add LogView tags to create the desired kind of diagrams or tables.
Here the *Advanced page* in the LogView interface can be of great help - first try out different parameters to get the right look of the diagram or table, and then press the *Tag* button and copy-and-paste the LogView tag into the report.
If the report should also be delivered by email, select this in the Report by email box.

**3** If report by email is chosen, some additional information has to be specified. Fill in the email address and edit the From and Subject fields if necessary. Also select how often and, if once a week, on what week day the report should be sent, in the Frequency box.

**4** Confirm the values as usual, and also choose whether a test mail should be sent right away.

## Delete report

In the *Delete report* wizard the administrator can select a report and delete it.

# Troubleshooting

The first thing to do in case of a problem is to check LogView's error log files and, failing that, the main Roxen server's error log files. Both LogView and Roxen try to generate comprehensible reports on errors that occur.
Specific problems:

## The installation of LogView failed

This is typically due to one of the following things:

**o** The server may not have write permission in the log files directory, or may lack permission to read some of the files in the LogView distribution. The main server's error log files will contain information about which file accesses has failed.

**o** LogView was instructed to store its logs and groups files in a directory other than the default `logview.data`, and this directory did not exist. Such directories must be created manually.

# FrontPage

FrontPage is a Microsoft tool for developing web pages. FrontPage needs its own proprietary server extensions to function properly. There are no native server extensions for Challenger but there is support for running the Apache versions.

FrontPage is a limited tool when it comes to developing web sites. It works for one user, but in a multi-user environment better tools are needed. The full Roxen Platform provides the needed functionality for true multi-user web site development.

## Installing

This chapter describes how to install FrontPage98 support in your Challenger server.

Apart from an installed Challenger server, you will need FrontPage98 extensions for your operating system, they can be downloaded from RTR, http://www.rtr.com/.

FrontPage98 may not be supported on all platforms that Roxen supports.

The process of installing FrontPage98 support in a Roxen server is as follows:

**1** Install the FrontPage98 extensions on your system, in the directory `/usr/local/frontpage`, using the script included with it. If the program asks if you want to install *root web*, *sub/per-user webs* or *virtual webs*, say no. This can be installed later.

**2** Create a directory where FrontPage can store its files. The directory is called *Document Root* further on.

**3** Add the *FrontPage Script support* module to the virtual server where you want FrontPage support. Configure the module variables as follows:

**FrontPage/Document Root** specifies where the FrontPage webs are located in the file system.

**Run script as** specifies which user to run the FrontPage scripts as. The user needs read and write privileges in the *Document Root* directory.

**4** Add a *Filesystem*-module and configure the module variable *Search Path* to point to the *document root* directory.

**5** Create the FrontPage98 configuration file. This is done via the `addfp98.pike` script that can be found in the `roxen/tools/` directory. The syntax of the script is described below:

```
pike addfp98.pike -d <domain> -l <login>
-p <passwd> -r <document-root>
[--port=<port>]
```

**-d <domain>** The host and domain name of the server.

**-l <login>** The login name to be used to edit pages with FrontPage clients.

**-p <passwd>** The password to be used to edit pages with FrontPage clients.

**-r <document-root>** The location in the file system where FrontPage webs are stored.

**--port=<port>** The port number of the server.

### Example

```
pike addfp98.pike -d www.roxen.com -l www -p
gazonk
-r /usr/local/frontpage/content
```

The administrator will have to use the `fpsrvadm.exe` utility in the FrontPage98 extensions distribution to create, delete, and rename subwebs; Challenger's FrontPage98 support does not support these operations through the FrontPage clients.

# Upgrading

This chapter describes the process of upgrading your Roxen Challenger server. Either the whole server can be upgraded to a new release, or single modules can be upgraded separately.

## New releases

Before upgrading you should read through the release notes of the new release, in case there are incompatible changes. If there are, there will be special instructions on how to upgrade.

Challenger is upgraded simply be installing the new version over the old one. This is done by doing a normal installation with your Challenger directory as target directory. See the Installation chapter for more information.

The installation script will first move the `server` directory to `server.old` and then create a new `server` directory for the new release. Since only program files are stored in `server` this won't affect any user configurations. This is why it is important not to install any third-party modules in `server`.

After completing the installation of the new release the running server must be shut down, by using the *Shutdown/ Shut down Roxen* action. After that the new version can be started by running the start script, `server/start`. Everything should work out of the box.

If problems are encountered you can easily downgrade to the previous release by shutting down the new server, moving the `server.old` directory to `server` and running the start script again. If a `server.old` directory exists, it will be moved to `server.older` when you install the new release. If a `server.older` directory exists, it will be deleted. If you want to keep a particular Challenger release you should rename the `server.old` or `server.older` directory. It might not be as simple to downgrade to an older version after you have started using the new release.For example, there might be modules in the new release that were not present in the old one.

## Modules

It is possible to upgrade individual Challenger modules. For most modules this is done simply by replacing the module's `.pike` file with a new version, and reloading the module with the *Reload this module* button. Some modules consist of more than one file, in which case all files must be upgraded.

This is the normal way to upgrade third-party modules.

# Third party extensions

Challenger handles three types of third-party extensions, *Challenger modules*, *CGI-scripts* and *Java servlets*. With any extension you install on your web server you must take the security issues into account. You must trust the programmer of the extension to have written it with security in mind. The main problem is how user input is handled, what will happen if the extension gets a megabyte of machine code instead of the line of text it expected.

### Challenger modules

Challenger modules are the preferred way of extending Roxen Challenger. Modules become part of Challenger and their functionality can be used by other modules. In effect, modules have to do less and provide more. A module does not have to do as much, which means that it will be shorter and the risk of bugs will be reduced.

Challenger takes care of a lot of potential security issues by itself. For example, it is impossible to make a buffer overrun bug that enables an attacker to execute machine code. But, this does not mean that the programmer of a Challenger module can ignore security considerations. A Challenger module will always be run with the same permission as the Challenger server and any security hazard in the module will affect the whole server. Therefore, we recommend you only run modules distributed by reputable web sites.

### CGI-scripts

Most third-party extensions for web servers are available as CGI-scripts, because they will work on all web servers. In Challenger, the *CGI executable support* module handles CGI-scripts.

When writing CGI-scripts, the programmer has to take care of all security issues herself. This is currently not a skill all programmers possess, it has only been an issue for a little while. Therefore, CGI-scripts are a major cause of security hazards today.

Challenger has the option of running CGI-scripts as a low privilege user. Use this to make sure that the CGI-script can cause as little damage as possible. Only download CGI-scripts from web sites that show that they treat security issues seriously.

### Java servlets

Java servlets are extensions that share many of the good properties of Challenger modules. They are run within the web server, and therefore uses less resources CGI scripts. They are implemented using a interpreted object-oriented language, thereby making it harder to make mistakes that affect the security of the server.

## Installing

### Challenger modules

Challenger modules are installed by installing the module's files somewhere the server can find them. Usually, in `local/modules`, but, by configuring the *Global Variables/Module directories* variable they can be placed anywhere. Do not place any third-party modules in the `server/modules` directories since `server` is replaced when the Challenger server is upgraded.

After installing the module's files and making sure Challenger can find them, the module can be enabled by the *Add module* button. It might take a reload on the *Add module* page before the new module is available.

### CGI scripts

Before installing a CGI-script you should consider whether a Challenger module exists that can provide the same functions. If not, you should follow the installation procedure that comes with the script. This will probably describe how you install a CGI-script in another web server, but it shouldn't matter much. Challenger works like any other web server when it comes to CGI-scripts.

### Java servlets

A Java servlet is installed by creating a copy of the *Java servlet bridge* module for that servlet. See the Servlets page in the Scripting chapter for more information.

# Portability

Roxen Challenger 1.3 is available for Unix as well as Windows.

All modules in the Challenger distribution work on both Unix and Windows. Most third-party or custom-made modules will also work on both platforms. Modules are programmed in Pike and Pike is, in itself, platform independent. Normally, both Unix and Windows will look the same to a module, with few exceptions.

## Unixes

Roxen Challenger is currently supported by Idonex on the following platforms:

- Linux 2.1 and 2.2 (RedHat)
- Solaris 2.5, 2.6 and 7
- HP-UX 10.20
- AIX 4.2
- IRIX 6.3
- Digital Unix 4.0
- FreeBSD 2.2.6
- Windows NT 4, SP 3

Roxen Challenger will run on a number of other platforms as well. It does for example run on Windows 95 and 98, even though we do not recommend using these platforms for running a web site but rather for internal development. Since Challenger is available with source code it has also been reported to compile and run on other Unix platforms.

## Windows 95/98/NT

Roxen Challenger supports Windows NT 4 SP 3 and works as a development environment on Windows 95 and 98 as well.

Generally, Pike scripts and RXML code can be moved from Unix to Windows systems more or less directly, but there are a few details to keep in mind:

- File names. Unix uses a slash "/" to separate components in path names, while Windows uses backslash "\". However, internally Windows accepts "/" as a path separator. Unlike Unix paths on Windows are case insensitive. If a file `index.html` exists it is impossible to create a file `Index.html`. Windows also suffers from short file names, a file name can always be abbreviated to be 11 characters long. It is therefore often not possible to treat file names or paths as being unique on Windows, it is usually possible create another form of the file name or path that will be accepted.
- There are differences in terms of memory management, user identities, and file protections. While not generally a problem for the basic running of Challenger, security issues may need special attention.

The Roxen Challenger distribution is adjusted for these things, but third-party modules and user scripts might need some manual adjustment.

# Reporting bugs

We are committed to make Challenger a software of high quality that does not cause problems for its users. For us to reach that goal it is important that we get reports about bugs encountered while using the software.

For a report to be useful it must give us enough information to pin-point the problem. If we don't understand what the problem is, it is hard to fix it. Before sending any bug report, you should read the bug report critically and think whether you would be able to reproduce the bug with the information in the bug report. If you can't, we probably can't either.

Idonex does not provide free support for our products. We will use your input to help fix bugs and improve the quality of our manuals, but we will not be able to reply to all bug reports. It is better to post questions on the Roxen Challenger mailing-list where Challenger users probably will help you.

## Is it relly a bug?

Before reporting a bug, it is best to verify that the problem is actually caused by a previously unknown bug in the Challenger server. Several resources are available to help with this task:

**The Roxen Challenger FAQs** Lists of Frequently Asked Questions exist on the Roxen web site.

**The Roxen Challenger mailing list** You may want to join the Roxen Challenger mailing list, where users of Roxen Challenger help each other. Challenger developers follow the mailing list and will take note of any bugs reported on it. It is also possible to browse the archive of old messages. Refer to the contact page at the Roxen web site.

## Necessary information

The developers will need some specific information to identify and correct a bug. Ideally, you should provide all the information below.

**Description** A description of the circumstances in which the problem occurred, and of the problem. It should be possible to reproduce the bug with only the information contained in the description.

**Error message** The text version of the error message, available from the error report page.

**Version information** Version numbers of the following:

- Operating system
- Challenger
- Pike
- Any Challenger modules involved
- Any databases involved

## How to report

When you have collected all the relevant information, go to the bug report page on the Roxen web site and follow the instructions. If you cannot access this page, mail to the bug report address bugs@roxen.com.

# Appendix

This appendix contains the following documents:

**Available modules** List of the available modules.

**HTTP response codes** List of HTTP response codes and their explanations.

**Migrating to Roxen Challenger** Explaining how to convert sites previously provided by other servers.

## Available modules

| Module | Type |
|---|---|
| .htaccess support | Authentication, filter, last try, URL |
| AC: AC database user import | Authentication |
| AC: Basic HTTP and FTP authentication | Authentication |
| AC: HTTPS only authentication | Authentication |
| AC: Internal password | Authentication |
| AC: OS user import | Authentication |
| AC: User Database | Authentication |
| Access Control | Authentication, location |
| Automatic sending of compressed files | First try |
| Business Graphics | Location, parser |
| CGI executable support | Last try, location, file extension |
| CVS File system | Location |
| Client logger | Logger |
| Config tab-list | Location, parser |
| Configuration interface | Location |
| Content types | Content-type |
| Countdown | Parser |
| Demo module | Location |
| Directory parsing module | Directory |
| Enhanced directory listings | Directory, parser |
| FTP gateway | Location, proxy |
| Fast directory module | Directory |
| Fast-CGI executable support | Last try, location, file extension |
| Filesystem | Location |
| Folder list tag | Parser |
| Fnord! | Parser |
| Frontpage Script support | Location |
| Graphics text | Location, parser |
| HTTP-proxy | Location, proxy |
| HTTP-Relay | First try, last try |
| IP-less virtual hosting | First try |
| ISMAP image-maps | File extension |
| Incoming filesystem | Location |
| Index files only | Directory |
| Indirect href | Parser |
| IntraSeek Search Engine | Location, parser |
| Killframe tag | Parser |
| Language | Directory, parser, URL |
| LogView | Location, logger, parser |
| Logging disabler | Logger |
| Main RXML parser | File extension, main parser, parser |
| Mirror Filesystem | Location |
| Navigation | Parser |
| Outlined box | Parser |
| PATH_INFO support | Last try |
| Pike Image Module | Location, parser |
| Pike script support | File extension |
| Pike tag | Parser |
| Redirect Module | First try |
| Restricted filesystem | Location |
| Secure File System | Location |
| SED tag | Parser |
| SQL databases | ... |
| SQL user database | Authentication |
| SQL-module | Parser, provider |
| SSL Proxy | First try, proxy |
| Secure file system module | Location |
| SiteBuilder | - |
| SiteBuilder Content Editor | Location |
| SiteBuilder Tags Module | Parser |
| SiteBuilder Work Area Filesystem | Location |
| Spell checker | Parser |
| Tablify | Parser |

## HTTP response codes

When an HTTP server has received and treated a request a response code is transferred back to the client. This table contains the possible responses the server can send.

| Code | Meaning |
|---|---|
| 200 Document follows | The request has been fulfilled and an entity corresponding to the requested resource is being sent in the response. |
| 201 Created | The request has been fulfilled and resulted in a new resource being created. |
| 202 Accepted | The request has been accepted for processing, but the processing has not been completed. |
| 203 Provisional information | The returned meta information in the Entity-Header is not the definitive set as available from the origin server, but is gathered from a local or a third-party copy. |
| 204 No Content | The server has fulfilled the request but there is no new information to send back. If the client is a user agent, it should not change its document view. |
| 300 Moved | The requested resource is available at one or more locations and a preferred location could not be determined via content negotiation. |
| 301 Moved permanently | Requires the \QLocation' header, see the <head> tag. The requested resource has been assigned a new permanent URI and any future references to this resource must be done using the returned URI. |
| 302 Moved temporarily | Requires the \QLocation' header, see the <head> tag. The requested resource resides temporarily under a different URI. Since the redirection may be altered on occasion, the client should on future requests from the user continue to use the original Request-URI and not the URI returned in the URI-header field and Location fields. |
| 304 Not modified | The document has not been modified. |
| 400 Bad Request | The request had bad syntax or was inherently impossible to satisfy. The client is discouraged from repeating the request without modifications. |
| 401 Access denied | The request requires user authentication. The response must include a WWW-Authenticate header field containing a challenge applicable to the requested resource. The client may repeat the request with a suitable Authorization header field. This might, for instance, be used inside a <deny user=...> tag. |
| 402 Payment Required | The user has to pay you to get the information. |
| 403 Forbidden | The request is forbidden for a reason that is unknown to the client. |
| 404 No such file or directory | The server has not found anything matching the Request-URI. No indication is given of whether the condition is temporary or permanent. |
| 405 Method not allowed | The method specified in the Request-Line is not allowed for the resource identified by the Request-URI. |
| 408 Request timeout | n/a |
| 409 Conflict | n/a |
| 410 This document is no more | The requested resource is no longer available at the server and no forwarding address is known. |
| 500 Internal Server Error | n/a |
| 501 Not implemented | n/a |
| 502 Service Unavailable | n/a |
| 503 Gateway Timeout | n/a |

## Migrating to Roxen

Challenger can take care of many of the special features of other well-known servers. Basically it is quite painless to switch to Challenger. Below you will find the common special features of other servers and how Challenger treats them.

### Image maps

### NCSA Image map tip

By adding a redirect from `/your_cgi-bin_dir/imagemap/` to `/`, Challenger will handle all files using the NCSA image map script internally. This means that you will not have to change any links to get your old image maps to work with Challenger.

### Server Side Includes

Challenger is compatible with NCSA/Apache style Server Side Includes, SSI. However, we suggest that you rewrite your files to take advantage of Challenger's native features instead, features that can accomplish the same things, but faster, mainly the various RXML commands.

By default, support for the execute script command `<!-- #exec -->` is turned off, while the others are on. This is controlled in the Main RXML parser.

Server side includes are internally treated like RXML tags.

## CGI

All your existing CGI scripts should work but be sure to read the section on the CGI script module.