



xmoovStream Server 0.8 Documentation

<http://stream.xmoov.com> ↗

Table Of Contents

1. Welcome	4
2. Getting Acquainted	5
2.1 What Is the xmoovStream Server	5
2.2 Features	5
2.3 System Requirements	5
2.4 Package Contents	6
2.5 How The xmoovStream Server Works	7
2.6 The xmoovStream Server File Structure	7
3. Before You Install	8
3.1 Check Your Installation Environment	8
3.2 Quick Start	8
3.3 Pre-installation Planning	8
4. Installation	9
4.1 Installing xmoovStream	9
4.1.1 Upload xmoovStream	9
4.1.2 Make Logs Folder Writable	9
4.1.3 Include xmoovStream	9
4.1.4 Upload xmoovStream Helper	9
4.2 Protected Files	9
4.2.1 Setup Protected Files Location	9
4.3 Install xmoovStream Servers	10
4.3.1 Create Server Directories	10
4.3.2 Create index.php file	10
4.3.3 Create htaccess file	10

5. The xmoovStream Class	11
5.1 Using the xmoovStream Class	12
5.2 Properties and Methods	13
5.2.1 xmoovStream Class Initializer	13
5.2.2 Public xmoovStream Class Methods	13
5.2.3 Public xmoovStream Class Properties	13
6. Creating Custom Server Configuration Files	14
6.1 URL Rewriting	15
7. Configuration	16
7.1 Configuration Settings	16
7.2 Configuring xmoovStream Logging	17
7.2.1 Error Logging	17
7.2.2 Activity Logging	17
7.3 Configuring Bandwidth Control	18
7.3.1 Buffer Size	18
7.3.2 Burst Size	18
7.3.3 Bandwidth Throttling	18
7.4 Configuring MIME Types	18
8. Hot Link Protection	19
8.1 Using Hot Link Protection	19
8.1.1 Include xmoovStream	19
8.1.2 Initiate the xsToken Class	19
8.1.3 Generate Token Key	19
8.1.4 Generate Protected Link	19
8.2 Accessing Hot Link Protected Files	20
8.3 xsToken Class Methods and Configuration	20
9. Terms and Resources	21

1. Welcome

First of all, we would like to thank you for downloading xmoovStream and supporting open source software. A great deal of effort has been invested into making pseudo streaming available to people with small budgets and limited resources. The xmoovStream Server project is very valuable to us as it has proven to be a useful and considerable contribution to the open source community. Today xmoovStream has become a serious solution for web masters and content publishers of all kinds. We will continue to improve xmoovStream and expand its capabilities to its maximum potential. We hope you find xmoovStream a useful and worthy companion for your projects. Please feel free to [contact us](#) ↗ if you have any suggestions on how we can improve the xmoovStream Server.

Get Acquainted

This documentation covers all aspects of the xmoovStream Server. If you are new to xmoovStream, we strongly recommend you start with chapter **2** to make sure your web server meets the necessary requirements and your hosting company allows such a system. You will also get a rough idea of how the xmoovStream Server works and differs from other php based http streaming systems.

Quick Installation

If you are installing for the first time, we recommend you do a plug-and-play quick install as described in chapter **3.2**. Here you will learn how to setup the xmoovStream Server in just a few minutes.

Installation

If you have experience with xmoov-php or other php based http streaming scripts, you may skip ahead to chapter **4**, which jumps right into a proper xmoovStream Server installation.

Advanced xmoovStream

Chapters **5 - 8** delve deeper into the xmoovStream Server core functions and configurations. Here you will learn how to customize and extend the xmoovStream Server to meet your specific needs.

Links and Information

For more information and links to helpful resources, jump to chapter **9**.

Happy xmoovStreaming!

2. Getting acquainted

2.1 What is the xmoovStream Server

The xmoovStream Server is a centralized http streaming server written in PHP. It sits between your files and the world giving you control over many aspects of how your content is accessed. Amongst its many features, the xmoovStream Server offers pseudo streaming for video content. Pseudo Streaming is the ability to randomly access parts of a video that have not yet been loaded. This release of xmoovStream gives you such capability for Adobe® Flash® video players when streaming FLV video files and for Apple® Quicktime® Player X and iPhone® when streaming MP4 files. Random access video is not yet supported for Adobe® Flash® video players when streaming MP4 files.

2.2 Features

- Centralized http streaming server with multi MIME support
- Logging and email error reporting
- Hot link protection
- Bandwidth throttling and burst loading
- Http pseudo streaming for Flash® video players, Apple® Quicktime® Player X and iPhone®
- Included xmoovStream Audio and Video Players

2.3 System Requirements

- **PHP** Enabled **Web Server** 4+
- **safe_mode** off for large file support
- **mod_rewrite** engine on and htaccess support (or other URL rewriting equivalent) for pretty URLs
- **MetaData** injected FLV files for random access support

The xmoovStream Server has been coded for maximum compatibility in the most restrictive environments. The only mandatory requirement is a **PHP** enabled web server.

The PHP **safe_mode** needs to be deactivated in order to support large file sizes and bandwidth throttling. Having **safe_mode** turned on will restrict the length of time the xmoovStream Server is allowed to be active. This means downloads will be abruptly interrupted if the download time exceeds the maximum allowed execution time for a script.

The Apache **rewrite_engine** or other URL rewriting equivalent should be enabled but is not a must. Without it you will not be able to use nice and sexy URLs. More about URL rewriting in chapter 6.1.

In order to random access FLV video files, they must be injected with the proper **MetaData**. You can find a list of open source **MetaData** injectors at <http://stream.xmoov.com/support/metadata-injectors> ↗

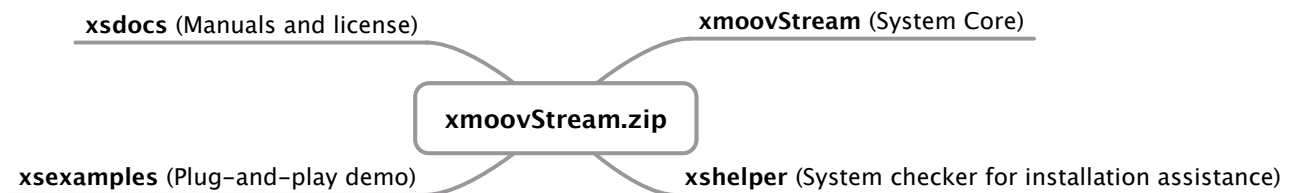
2. Getting acquainted

2.4 Package Contents

The xmoovStream package comes with everything you need to get you xmoovStreaming quickly. The zip package is actually a plug-and-play installation which will give you instant results if you upload the contents to your server as is. The included xmoovStream Audio and Video players round this package up supplying you with everything you need to start publishing rich media content right away.

The xmoovStream package includes the following:

- **xsdocs:** Manuals and license
- **xsexamples:** Plug-and-play demo to help get you get started quickly
- **xmoovStream:** xmoovStream core system
- **xshelper:** System checker for installation assistance



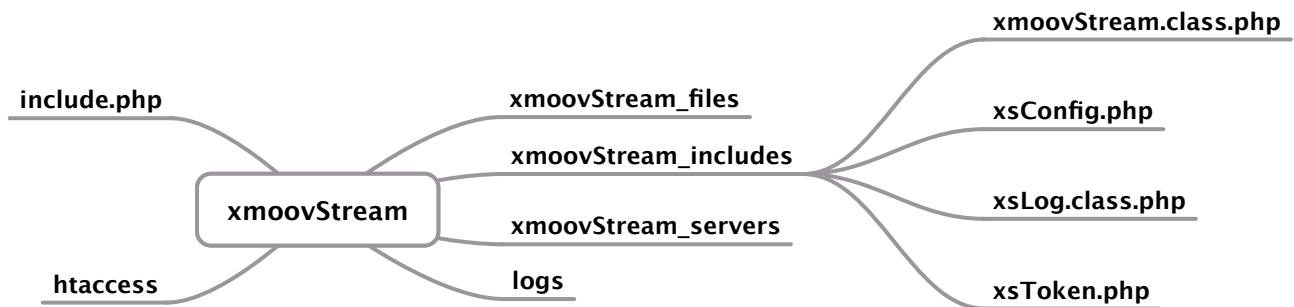
2. Getting acquainted

2.5 How The xmoovStream Server Works

The xmoovStream Server is different from xmoov-php and other streaming scripts. It is a centralized system that sits in a fixed location while still allowing you to create multiple streaming servers anywhere on your site. Rather than accessing a single script directly, the xmoovStream Server is used by including **include.php** into a php file and calling the **xmoovStream (parameters)** function. The parameters tell the xmoovStream Server which server configuration file to use. Server configuration files hold a unique set of rules that define the conditions for any given streaming environment.

2.6 The xmoovStream Server File Structure

The center of the xmoovStream Server system is the xmoovStream folder itself. You will find an **include.php** file, **htaccess** file and four folders in its root. The **xmoovStream_includes** folder holds all of the xmoovStream core functions and the configuration file. You will find the xmoovStream Audio and Video players in the **xmoovStream_files** folder.



- **include.php**: For inclusion in php documents in order to access the xmoovStream system
- **htaccess**: Protects the xmoovStream folder from public access
- **xmoovStream_files**: The default folder for your protected files
- **xmoovStream_includes**: xmoovStream core functions folder
 - **xmoovStream.class.php**: The xmoovStream class
 - **xsConfig.php**: The configuration file where the defaults are defined
 - **xsLog.class.php**: The xmoovStream logging class
 - **xsToken.class.php**: The xmoovStream hot link protection class
- **xmoovStream_servers**: Server configuration files folder
- **logs**: The default folder for the log files

3. Before You Install

3.1 Check Your Installation Environment

Hosting companies have very specific terms and conditions in regards to how you are allowed to utilize their service. It is important you make yourself aware of these conditions before installing xmoovStream. Failing to do so may result in termination of service by your hosting company.

3.2 Quick Start

XmoovStream has been packaged to operate plug-and-play right out of the box to give you a quick start. The only thing you need to do is unzip the package and upload all the contents to the public root of your server. This will place the xmoovStream core folder in the public accessible root of your server; however, it is not recommend you leave things this way. Follow the instructions in chapter 4 to do a proper installation.

3.3 Pre-installation Planning

The best place to start your planning is deciding where your files are going to be accessible from. You should start by making a list of URLs for later reference during the installation process.

Example:

videos:	http://mysite.com/videos/myvideo.flv
audio:	http://mysite.com/audio/mysound.mp3
images:	http://mysite.com/images/myimage.jpg
downloads:	http://mysite.com/downloads/myfile.zip

Of course there will be no files at these locations; but rather an **index.php** and **htaccess** file which will translate the URL into php **\$_GET** values and pass them on to the xmoovStream Server. The possibilities are endless in how you can use these URLs to control the xmoovStream Server, which we will get to later in this documentation.

4. Installation

4.1 Installing xmoovStream

4.1.1 Upload xmoovStream: Upload the xmoovStream folder to your server, preferably outside of the public accessible root. This will be the permanent location of the xmoovStream system. Do not change or move any files in this folder yet. If your provider does not allow you access outside the public root you may use the included htaccess file to deny public access to the directory.

4.1.2 Make Logs Folder Writable: If you would like to use the logging feature make sure the `/xmoovStream/logs/` folder is writable. This is a good idea if you are using xmoovStream for the first time.

4.1.3 Include xmoovStream: Open `xshelper/index.php` and enter the full server path to `/xmoovStream/include.php`. This should look something like this:

```
include( '/home/www/mysite.com/xmoovStream/include.php' );
```

4.1.4 Upload xmoovStream Helper: Upload the `xshelper` folder to the public root of your web server and surf to `http://mysite.com/xshelper/`. You will see a system information page if you have set the proper path; or an error message if it is incorrect. You only have to get this right once. You can use this same path for the upcoming steps. **Note:** This folder is for installation assistance only! Do not leave this file on your server as it poses a security risk. Remove this folder as soon as you have everything running properly.

4.2 Protected Files

Now that you have xmoovStream properly installed it's time to set up the location of your xmoovStream protected files. The default folder is `/xmoovStream/xmoovStream_files/` located inside the xmoovStream folder you uploaded in step 4.1.1. Follow these instructions If you would like to change the default file path or skip to step 4.3 if you would like to use the default file location.

4.2.1 Setup Protected Files Location: Open the file `/xmoovStream/xmoovStream_includes/xsConfig.php` and edit the constant `XS_FILES`. This should point to the root of your xmoovStream protected file path. Any subfolders can be defined in the individual server configuration files. The default setting is `(XS_DIR . '/xmoovStream_files')`. When you make changes and upload, you can go back and refresh the welcome page to see if things are still in order.

4. Installation

4.3 Install xmoovStream Servers

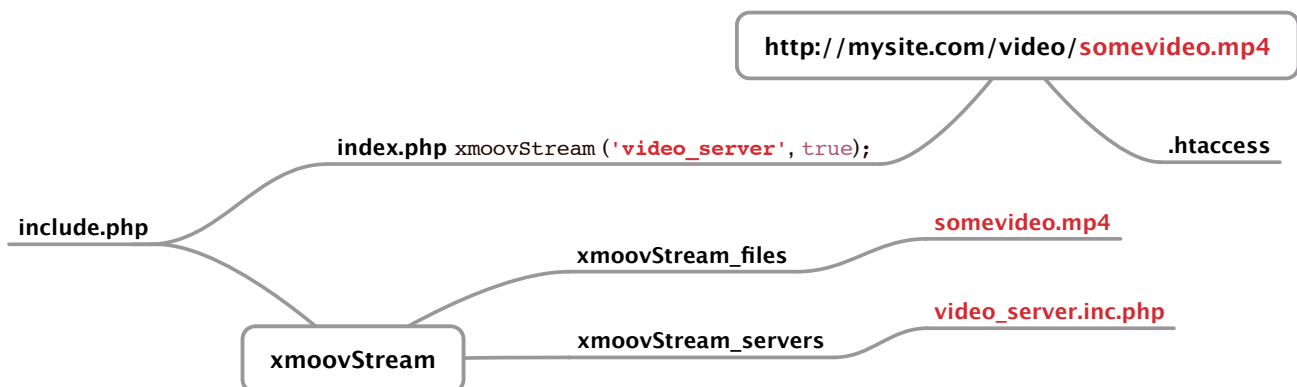
Now that we have xmoovStream installed, it is time to install your server(s). Remember the list of URLs we made in the planning stage? Now you need to create these directories on your web server. Each directory will require an **index.php** and **htaccess** file. The **index.php** needs two (2) lines of code. The first line includes **include.php** just like we used in step 4.1.3. The second line calls the **xmoovStream()** function. The first parameter tells the xmoovStream Server which server configuration file to use. The second parameter tells the xmoovStream Server to begin the download immediately. Find more in depth information about xmoovStream server configuration files in chapter 6.

4.3.1 Create Server Directories: Create the directories in your public_html folder from your planning notes.

4.3.2 Create index.php File: Create an **index.php** document in each directory with the proper code. If you are setting up a video streaming server your code should look like this:

```
1. <?php
2.     include( '/home/www/mysite.com/xmoovStream/include.php' );
3.     xmoovStream( 'video_server', true );
4. ?>
```

This example tells the xmoovStream Server to load **video_server.inc.php** and begin the download immediately.



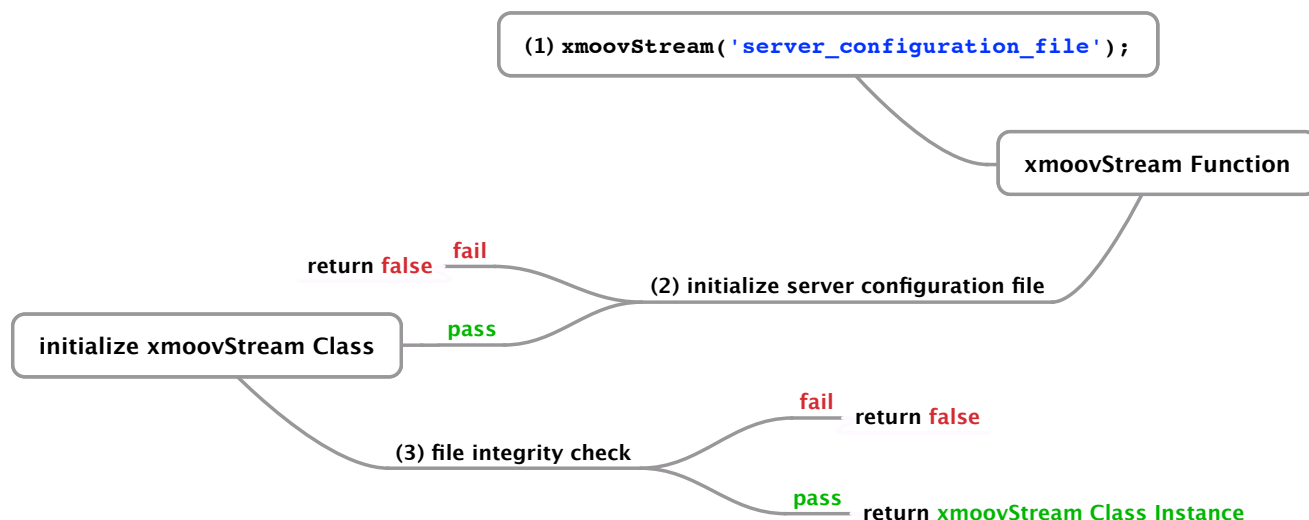
4.3.3 Create htaccess File: If you open the **server.inc.php** file, you will find example htaccess code in the comments. Create the `.htaccess` file (*do not forget the (.) dot before the file name*) and paste the code into the file. Upload the folder(s) to your web server and you're all set. If do not use Apache you will need to use another URL rewriting equivalent in order to use nice URLs. More about this in chapter 6.1.

5. The xmoovStream Class

In the following chapters we will get more in-depth in the xmoovStream class and server configuration.

The xmoovStream class is the core of the xmoovStream Server. To use it you must always first include **/xmoovStream/include.php** in a php document. To initiate the class you must call the **xmoovStream()** function. The **xmoovStream()** function takes two parameters, the first of which is required and tells the xmoovStream Server which server configuration file to use. The second parameter is optional and tells the xmoovStream Server whether or not to begin the download immediately. If you set the second parameter to Boolean false or leave it empty, you can start the download later by calling the xmoovStream class **download()** function.

Once invoked, the **xmoovStream()** function will initialize the server configuration file. If it returns Boolean true it will attempt to initialize the xmoovStream class. The xmoovStream class then checks if the file exists, security integrity and if the mime type is allowed and returns a Boolean result. If all is in order the **xmoovStream()** function will return the class instance. If not it will return a Boolean false.



Once the xmoovStream class is initiated, you will have access to all of the xmoovStream class properties and methods. You can use these to add additional functionality to your streaming servers or retrieve file information for display in php generated content pages.

5. The xmoovStream Class

5.1 Using the xmoovStream Class

The following is an example of an **index.php** document like the one we created in chapter 4.3.2. In this example we redirect the client to an error page if the **xmoovStream()** function fails.

```
1. <?php
2.     include('/home/www/mysite.com/xmoovStream/include.php');
2.     $myXmoovStream = xmoovStream('server_configuration');
3.     if($myXmoovStream) {
4.         $xmoovStream->download();
5.     } else {
6.         header('location: http://mysite.com/error_page.html');
7.     }
8. <?php
```

As previously mentioned you can also use the xmoovStream class to retrieve file information for display in php generated content pages. This example shows how to retrieve file information using the **file_info** server configuration file.

```
1. <?php
2.     include('/home/www/mysite.com/xmoovStream/include.php');
3.     $myXmoovStream = xmoovStream('file_info');
3.     if($myXmoovStream && $myXmoovStream->set_file('/path/file.ext')){
4.         $file_size = $myXmoovStream->human_file_size;
5.         $file_type = $myXmoovStream->mime;
6.         echo "File Name: file.ext ($file_type, $file_size)";
7.     }
8. ?>
```

5.2 Properties and Methods

The following is a list of properties and methods used by the xmoovStream class.

5.2.1 xmoovStream Class_INITIALIZER:

Method (DataType:Parameter) : Description
xmoovStream (String: configuration file, Boolean: start download): Invokes the xmoovStream class. Returns the class instance or Boolean false if the server configuration file fails or something goes wrong.

5.2.2 Public xmoovStream Class Methods:

Method (DataType:Parameter) : Description
set_config (Array or String: configuration): Sets the configuration variables overriding the defaults defined in xsConfig.php.
set_file (String: file) : Sets the streaming file. This function will check security, if the file exists and if the mime type is allowed and return a Boolean result.
get_mime_type (none): Returns the mime type of the current streaming file.
get_file_size (Boolean: human readable) : Returns the size of the current streaming file. Setting the parameter to Boolean true will give you a human readable format.
download () : Initiates the file download.

5.2.3 Public xmoovStream Class Properties:

Property {DataType} : Description
version {String} : xmoovStream Server version.
min_php_version {String} : Minimum required php version.
file {String} : The full path to the current streaming file.
file_name {String} : The current streaming file.
file_modified {Integer} : The date the current streaming file was last modified.
config {Array} : The current configuration array.
mime {String} : The mime type of the current streaming file.
file_size {Integer} : The file size of the current streaming file.
human_file_size {String} : The human readable file size if the current streaming file.

6. Creating Custom Server Configuration Files

Server configuration files are what make the xmoovStream Server so versatile. They are located in **/xmoovStream/xmoovStream_servers/**. There is a set of pre-made configuration files in the package that you can customize; but you can also make your own from scratch. The rules are simple and straight forward. Each file requires a constant which gives it a name for the logs and one function **init_server()** that returns a configuration array to the xmoovStream Server. Any setting defined in the array will override the settings defined in **xsConfig.php**. You will find a complete list of configuration variables in chapter 7.

The only absolute mandatory setting the xmoovStream Server requires is **file** so it knows which file to stream. This is usually taken from a **\$_GET** variable. You should always validate **\$_GET** variables using the PHP **isset** function before you apply them. The following is an example of a basic server configuration file.

```
1. <?php
2.     define('XS_PROVIDER', 'Your custom server name');
3.     function init_server() {
4.         if(isset($_GET['file'])) {
5.             return array('file' => $_GET['file'], etc...);
6.         }
7.         return false;
8.     }
9. ?>
```

XmoovStream server configuration files MUST be saved as **my_custom_server.inc.php**. Please note that **"my_custom_server"** can be what ever you want but you must not forget the **".inc.php"** at the end.

6. Custom xmoovStream Servers

6.1 URL Rewriting

URL rewriting is the process of intercepting an incoming request and automatically redirecting it to a different URL. Rewritten URLs (sometimes known as short, nice URLs, or search engine friendly - SEF) are used to provide shorter and more relevant-looking links to web pages. The xmoovStream Server uses URL rewriting to transform URLs into **\$_GET** variables. With it, a URL like:

`http://mysite.com/video/video.mp4` can be internally translated to: `http://mysite.com/video?file=video.mp4`

A **rewrite engine** is the software that handles URL rewriting. Different server software use different rewrite engines. Apache uses the module **rewrite_engine** for URL rewriting. It uses htaccess files, also known as **Distributed Configuration Files**. The xmoovStream Server package comes with preconfigured htaccess files for the Apache mod **rewrite_engine** only. They contain instructions to the web server regarding how files are allowed to be accessed. If you use **another rewrite engine**, you will have to install the necessary files needed for URL rewriting yourself. Each server configuration file has a description of the rewritten URL and the query strings needed to support the server configuration.

7. Configuration

The configuration file is located in `/xmoovStream/xmoovStream_includes/xsConfig.php`.

The settings are stored in an array called `$defaults` and apply to every streaming server unless overridden by the server configuration file. Any setting defined in a server configuration file will override the default setting defined in `xsConfig.php`.

7.1 Configuration Settings

Setting {DataType} : Description
file {String} : The file to be streamed.
position {Integer} : The start byte position of the streaming file.
output_file_name {String} : File name for output to the client. This allows you to define a file name that differs from the original.
file_path {String} : Server path to the streaming file. This allows you to set different file paths for each streaming server.
use_error_handling {Boolean} : Enables error logging and output.
use_activity_handling {Boolean} : Enables activity logging.
log_errors {Boolean, Array} : If set to Boolean:true, the xmoovStream Server will log all errors. If you use an array, the xmoovStream Server will only log error types defined in the array. <i>A list of error types can be found in section 5.3.</i>
log_activity {Boolean, Array} : If set to Boolean:true, the xmoovStream Server will log all activity. If you use an array, the xmoovStream Server will only log activity types defined in the array. <i>A list of activity types can be found in section 5.4.</i>
show_errors {Boolean, Array} : If set to Boolean:true, the xmoovStream Server will output errors to the browser. If you use an array, the xmoovStream Server will only show error types defined in the array. <i>A list of error types can be found in section 5.3.</i>
notify_errors {Boolean, Array} : If set to Boolean:true, the xmoovStream Server will send errors to the email address defined in notify_email . If you use an array, the xmoovStream Server will only notify error types defined in the array. <i>A list of error types can be found in section 5.3.</i>
notify_email {String} : The email address to send errors to.
use_http_range {Boolean} : Enables http_range handling. This is needed for iPhone streaming and download resuming.
force_download {Boolean} : Enables the "Save As" dialog box on the client browser.
buffer_size {Integer} : Buffer size in Kilobytes. <i>Find more information in section 5.3.1.</i>
burst_size {Integer} : Burst size in Kilobytes. <i>Find more information in section 5.3.2.</i>
throttle {Integer} : Throttle speed in Kilobytes per second. <i>Find more information in section 5.3.3.</i>
mime_types {Array} : Array of allowed MIME file types. <i>Find more information in section 5.4.</i>

7. Configuration

7.2 Configuring xmoovStream Logging

The xmoovStream logging system uses flat files to log errors and script activity. Log files are named using the current server date, which means there will be a unique log file for each day. Activity logging is primarily for debugging purposes and should be configured carefully, as this can cause excessive server activity. Apple iPhones use **http_range** to stream video files, thus causing a huge amount of server communication producing a large amount of logging data. This will result in large amounts of **file_access** and **partial_download_complete** entries in your log files to be generated.

7.2.1 Error Logging: You can configure the xmoovStream Server to log, show and report only specific kinds of errors by defining the error types in an array. The following error types apply to **log_errors**, **show_errors** and **notify_errors**.

Error Type : Description
file_open : The defined file could not be opened.
file_empty : A file was not defined.
file_path : The path to the streaming file could not be found.
404 : The streaming file could not be found.
security : A security error occurred.
file_type : The requested file type is not allowed.
time_limit : The PHP set_time_limit function failed. This can be due to safe_mode being enabled on your server.
magic_quotes : The PHP set_magic_quotes_runtime function failed.

7.2.2 Activity Logging: You can also configure the xmoovStream Server to log only specific kinds of activity by defining the activity types in an array. The following is a list of activity types.

Activity Type : Description
file_access : A file has been accessed.
flv_random_access : An FLV file has been accessed pass the 0 byte position.
download_complete : A file download has completed after being accessed from the 0 byte position.
partial_download_complete : A file download has completed after being accessed pass the 0 byte position.
user_abort : A file download has been aborted.

The **log_errors**, **log_activity**, **show_errors** and **notify_errors** settings should be configured as follows:

```
'log_errors' => array('file_open', 'file_path', '404', 'security', etc...)
```

7. Configuration

7.3 Configuring Bandwidth Control

This release has greatly improved on bandwidth control functionality. All of the bandwidth control settings (`buffer_size`, `burst_size` and `throttle`) are configured in Kilobytes. Use these settings with great caution as they can cause unwanted behavior if configured incorrectly.

7.3.1 Buffer Size: The `buffer_size` setting tells the xmoovStream Server how much data to send to the client per packet. The default setting is 8 (8.192) Kilobytes which is the maximum and should not be changed if you have limited knowledge in this area. For the PHP savvy, the `buffer_size` value is used for the second parameter in the PHP `fread` function.

7.3.2 Burst Size: The `burst_size` setting enables the xmoovStream Server to send an unthrottled initial burst of data to the client. If a value over 0 Kilobytes is defined, the xmoovStream Server will not throttle bandwidth until the defined amount of data has been sent to the client. This is helpful when streaming media files to players. The proper `burst_size` setting will give players shorter pre-buffer times and cause them to start playing faster.

7.3.3 Bandwidth Throttling: Bandwidth throttling lets you set the maximum speed in which files can be downloaded from your server. Bandwidth throttling can allow a higher number of clients to stream files at a more steady rate but will also cause the script to be active for a longer period of time. The `throttle` setting is defined in Kilobytes per second.

7.4 Configuring MIME Types

The `mime_types` setting takes an array of files extensions and their corresponding mime types. The xmoovStream script will use the mime type when sending the file type header to the browser. The `mime_types` setting should be configured as follows:

```
'mime_types' => array('.flv' => 'video/x-flv', '.mp4' => 'video/mp4', etc...)
```

8. Hot Link Protection

XmoovStream offers simple but effective hot link protection for your files. It uses an encrypted token key, which is assigned to a file name and saved in a cookie on the client's computer. The token key must first be generated by calling the **setToken** function and passing the file name as a parameter. You can now generate a link using the token key as part of the URL or as a **\$_GET** parameter. **Example:**

`http://mysite.com/downloads/tokenKey/file.zip` **or** `http://mysite.com/downloads/file.zip?tokenKey`

Next, the passed token key is validated in the server configuration file. You can then output alternative content or redirect the client if the token key is invalid.

8.1 Using Hot Link Protection

The following code **MUST** be inserted into a page **BEFORE** any HTML output is generated.

8.1.1 Include xmoovStream: You must first include `/xmoovStream/include.php` into a content page.

```
include( '/home/www/mysite.com/xmoovStream/include.php' );
```

8.1.2 Initiate the xsToken Class: Create a variable and initiate the **xsToken** Class.

```
$tokenHandler = new xsToken();
```

8.1.3 Generate Token Key: Call the **setToken** function and pass the file you would like to protect in the parameters. This function will return the generated token key.

```
$myToken = $tokenHandler->setToken( 'file.ext' );
```

8.1.4 Generate Protected Link: Use the value returned in step 7.1.3 to generate the protected link.

```
$myLink = 'http://mysite.com/downloads/' . $myToken . '/file.ext';
```

If you are not able to use htaccess files you can also pass the token key as a **\$_GET** parameter.

```
$myLink = 'http://mysite.com/downloads/file.ext?token=' . $myToken;
```

You can also generate the link on a later page. Once a key has been generated for a file you can retrieve it on a later page by repeating steps 7.1.1 - 7.1.2 and using the **getToken()** function as follows.

```
$myToken = $tokenHandler->getToken( 'file.ext' );
```

8. Hot Link Protection

8.2 Accessing Hot Link Protected Files

The final step in setting up hot link protection is creating a server configuration file that will validate the token. The following example is similar to the server configuration file described in chapter 6 with the proper code necessary to validate the token. This example will output alternative content if the token is invalid.

```
1. <?php
2.     define('XS_PROVIDER', 'Your custom server name');
3.     function init_server() {
4.         if(isset($_GET['file']) && isset($_GET['token'])) {
5.             $tokenHandler = new xsToken();
6.             if($tokenHandler->isValid($_GET['file'], $_GET['token'])) {
7.                 $file = $_GET['file'];
8.             } else {
9.                 $file = 'alternative_content.ext';
10.            }
11.            return array('file' => $file, etc...);
12.        }
13.        return false;
14.    }
15. ?>
```

8.3 xsToken Class Methods and Configuration

The following is an overview of the functions and configuration constants used by the xsToken Class.

Constant: Description

XS_SALT : Encryption salt for extra security.
--

XS_TOKEN_EXPIRE : Token expiration time. The default is one hour (3600)
--

Method (DataType:Parameter) : Description

setToken(String:FileName) : Creates a token and saves to the client computer.
--

getToken(String:FileName) : Retrieves a token for a given file.
--

isValid(String:FileName, String:Token) : Checks the validity of a token for a given file.
--

9. Terms and Resources

Bandwidth Throttling	http://en.wikipedia.org/wiki/Bandwidth_throttling
Hot Link	http://en.wikipedia.org/wiki/Inline_linking
Pseudo Streaming	http://stream.xmoov.com/support/faq/pseudo-streaming
Random Access	http://stream.xmoov.com/support/faq/pseudo-streaming
Web Server	http://en.wikipedia.org/wiki/Web_server
URL rewriting	http://en.wikipedia.org/wiki/Rewrite_engine
Apache	http://httpd.apache.org
PHP	http://php.net
PHP Class	http://php.net/manual/en/language.oop5.php
Constant	http://php.net/manual/en/function.constant.php
Cookie	http://php.net/manual/en/features.cookies.php
Salt	http://php.net/manual/en/function.crypt.php
Encryption	http://php.net/manual/en/function.crypt.php
\$_GET	http://php.net/manual/en/reserved.variables.get.php
array	http://php.net/manual/en/language.types.array.php
isset	http://php.net/manual/en/function.isset.php
return	http://php.net/manual/en/function.return.php
Boolean	http://php.net/manual/en/language.types.boolean.php
String	http://php.net/manual/en/ref.strings.php
Integer	http://php.net/manual/en/language.types.integer.php
include	http://php.net/manual/en/function.include.php
fread	http://php.net/manual/en/function.fread.php
mod_rewrite	http://httpd.apache.org/docs/1.3/mod/mod_rewrite.html
htaccess	http://httpd.apache.org/docs/1.3/howto/htaccess.html
safe_mode	http://php.net/manual/en/features.safe-mode.php
http_range	http://www.w3.org/Protocols/rfc2616/rfc2616-sec14.html
mime	http://www.w3schools.com/media/media_mimeref.asp
MP4	http://en.wikipedia.org/wiki/MPEG-4
FLV	http://en.wikipedia.org/wiki/Flash_Video
Quicktime	http://en.wikipedia.org/wiki/QuickTime
MetaData Injector	http://stream.xmoov.com/support/metadata-injectors

xmoovStream Server 0.8 Documentation (revision 2)

Updated January, 8 2010. <http://stream.xmoov.com/>

Created by Eric Lorenzo Benjamin jr. (xmoov.com), Edited by Sarina Bagby (xmoov.com).

This documentation, as well as the software described in it, is furnished under license and may be used or copied only in accordance with the terms of the license. The content of this manual, furnished for informational use only, is subject to change without notice, and should not be construed as a commitment by Eric Lorenzo Benjamin jr. (xmoov.com). Every effort has been made to ensure that the information in this manual is accurate. Eric Lorenzo Benjamin jr. (xmoov.com) assumes no responsibility or liability for any errors or inaccuracies that may appear in this documentation.

This Documentation was created using Free and Open Source software



MindNode <http://www.mindnode.com>



Bean <http://bean-osx.com>

Macromedia Flash is a registered trademark of Adobe Systems, Inc. Mac OS, iPhone and QuickTime are trademarks of Apple Inc. registered in the U.S. and other countries. All other product and company names are trademarks or registered trademarks of their respective holders.