# MEDIA-S

## USER AND DEVELOPER GUIDE

Document Date: April 3, 2003
Author: SideSpace Solutions, Inc.
WWW: http://www.sidespace.com/products/medias/

# TABLE OF CONTENTS



## PURPOSE

This document is designed to give end users an overview of how they can use Media-S to protect and access multimedia content. This document assumes that readers are familiar with the Media-S system architecture as outlined in the "Media-S System Design Document", available from the Media-S homepage at http://www.sidespace.com/products/medias/

A brief overview of Media-S will be provided, followed by a tour of the Media-S feature-set. Media-S programming interfaces and will then be documented.

## Media-S OVERVIEW

Media-S is designed to be an open, freely available DRM solution for multimedia content. While Media-S could be used to protect any type of multimedia file, this implementation will concentrate on the securing of Ogg Vorbis audio files.

At a high-level, implementing Media-S will require the following system components:

- Encryption program that will take as input a normal OGG file and encrypt the file according to user preferences to create an Media-S file. This program will be used by content providers and will be knows as "oggsenc".
- Player program that can acquire a content license and decrypt and play Media-S files. This player will be functionally equivalent to ogg123. For this player to work, it will require the use of a "license vault" that will store and manage a user's licenses. This program will be known as "oggs123".
- License Server that can grant licenses for specific content. License servers can be run by content providers to securely modify remote content licenses once a license has expired.

If more information about the functionality of Media-S is required, it is highly recommended that readers consult the "Media-S System Design Document", available from the Media-S homepage at http://www.sidespace.com/products/medias/

## A LINUX TOUR OF Media-S

Now that readers are familiar with the architecture of Media-S, it is time to take a brief tour of how you can use Media-S to protect multimedia content. Please note that the contents of this section will change from release to release. To minimize confusion it is recommended that the latest version of this document and the Media-S source code are used. To use this tour, you will need the following items:

- A snapshot of Media-S source code (available from the homepage) that has oggsenc, oggs123, and create-license compiled
- A plain, ordinary Ogg Vorbis file (in this case we will use a file named "test.ogg")

To use Media-S, simply perform the following steps:

### 1. Encrypt the Media-S file using oggsenc

oggsenc is the Media-S encoder for Ogg Vorbis content. This program takes a normal Ogg file as input and generates an encrypted Media-S file. Using oggsenc to encrypt content can be done via the following command line:

```
./oggsenc -i ~/test.ogg -o ~/test.oggs -k 1234 -K 1234
```

The output of this program will be an encrypted file called test.oggs. The parameters used are as follows:

| Parameter | Meaning |
|---|---|
| -i <filename> | Name of the input file |
| -o <filename> | Name of the output file |
| -K <string> | The vendor key to use for the protected file |
| -k <string> | The content-specific encryption key |

Additional command line parameters that could be used are described in the "oggsenc Usage" section below.

### 2. Create a license to decrypt the Media-S file using create-license

Now that an Media-S file has been generated, it is time to create a license that will grant a user the permissions required to manipulate the Media-S file. Normally, this license creation would be performed by a background process that would be invisible to the user. The create-license program was developed as a diagnostic tool and should not be distributed by end-users by the content provider.

The command line used to create a license to "PLAY" the Media-S file created above is as follows:

```
./create-license -i ~/test.oggs -e "12/25/2003" -a PLAY -d "12/26/
2003" -K 1234 -t
```

This command line will create a play license for the test.oggs file and deposit this license into your license vault (a simple DBM file). This license vault is stored by default in "~/.mediasvault".

The command line parameters used above are as follows:

| Parameter | Meaning |
|---|---|
| -i <filename> | Name of the input file |

| | |
|---|---|
| -e <date> | Global license expiration date in MM/DD/YYYY format |
| -K <string> | The vendor key to use for the protected file |
| -a <action> | The action being granted a license. Usually “PLAY”, “STREAM”, or “BURN”. |
| -d <date> | The expiration of the particular permission in MM/DD/YYYY format |
| -t | Whether this license is active or not. If the license is not active, do not pass this parameter. |

Additional command line parameters that could be used are described in the “oggsenc Usage” section below.

### 3. Decrypt the Media-S file using oggs123

Now that we have an encrypted file and a license to “PLAY” the content, we will run oggs123 on the encrypted file and, instead of playing the file, simply decrypt the file contents. Of course, the end user must not have this ability, but for the purposes of this demonstration we will decrypt our oggs file.

The command line for oggs123 is:

```
./oggs123 -i ~/test.oggs -o ~/newtest.ogg -K 1234
```

The command line parameters we used are as follows:

| **Parameter** | **Meaning** |
|---|---|
| -i <filename> | Name of the input Media-S file |
| -o <filename> | Name of the output ogg file |
| -K <string> | The vendor key to use for the protected file |

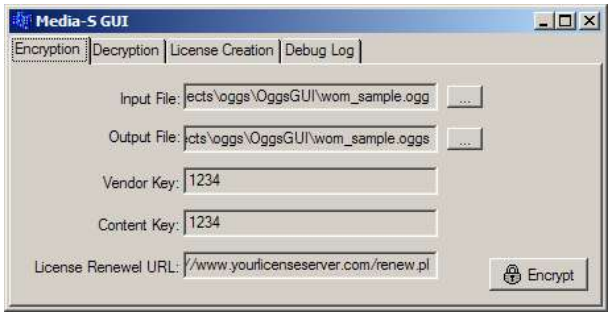
The output of this command is a file that will be identical to our original file. Comparing the two files via the following command confirms successful decryption:

```
diff ~/test.ogg ~/newtest.ogg
```

Now that a complete encryption-decryption cycle has been performed, it is time to document the above programs and libraries in more detail.

## A WINDOWS TOUR OF Media-S

Windows users have the option of using either the command-line clients documented above, or using a GUI tool known as Media-S GUI (or the Media-S demo application). This application takes most of the functionality exposed in the command line clients and puts a nice graphical interface in front of them:




For a tour of Media-S using this GUI, it is recommended that you read the document entitled "Media-S Demo For Windows Quick-Start Guide". This document (and GUI) are available at http://www.sidespace.com/products/medias/

## CONTENT PROVIDER IMPLEMENTATION GUIDE

Now that a brief introduction to Media-S has been provided, it is time to examine the application of Media-S in a real-world environment. This section will list the steps necessary to protect content using Media-S.

To utilize Media-S, content providers must perform the following steps:

1. Modify the salt in common.h to be a unique string. This step ensures that the default Media-S sample distribution will not be able to access your unencrypted stream data. For content providers wishing to add another layer of protection, it is recommended that the OpenSSL cipher's be changed from the default 3DES to another format that is secret to the content provider.
2. Modify oggs123 to play/stream/burn the decrypted content. oggs123 as provided in the Media-S distribution is a sample player that does not contain advanced functions such as playlist recognition or CD-burning. Any needed functionality will have to be added by content providers.
3. Statically compile libmediasdecrypt into your modified oggs123. This step ensures that a trojan libmediasdecrypt cannot be used to bypass the security provided by libmediasdecrypt .
4. Create a license server capable of generating Media-S licenses in the same fashion as create-license currently works. This step will be responsible for ensuring that requests are made by valid clients, and the clients have the necessary access privileges to retrieve the requested license.
5. Make the protected Media-S content available with your libmediasdecrypt -enabled player. This final step prepares your content in a form that can be easily used by your customers.

## Media-S PROGRAM REFERENCE

This section will detail the command line options for Media-S programs oggsenc, oggs123, and create-license.

### oggsenc

oggsenc is designed to allow content providers to create encrypted content that, in conjunction with a license, can be used in a flexible manner by end users. oggsenc accepts the following parameters:

| Parameter | Meaning |
|---|---|
| -i <filename> | Name of the input file |
| -o <filename> | Name of the output file |
| -K <string> | The vendor key to use for the protected file |
| -k <string> | The content-specific encryption key |
| -d | Print verbose debug information |
| -h | Display brief usage instructions |
| -v | Display program version |
| -u <url> | The renewal URL to be used for acquiring a license to this content. This parameter is not required. |
| -x <filename> | Name of the file to read XML meta-data from. Supersedes the above key and renewal parameters. |
| -y <filename> | Name of the file to read the content-specific encryption key from. |

### oggs123

oggs123 is provided as a sample program to demonstrate the functions of libmediasdecrypt (which are described in the developer's reference later in this document). oggs123 accepts the following command-line options:

| Parameter | Meaning |
|---|---|
| -i <filename> | Name of the input Media-S file |
| -o <filename> | Name of the output ogg file |
| -K <string> | The vendor key to use for the protected file |
| -d | Print verbose debug information |
| -h | Display brief usage instructions |
| -v | Display program version |
| -y <filename> | Name of the file to read the content-specific encryption key from. |

## create-license

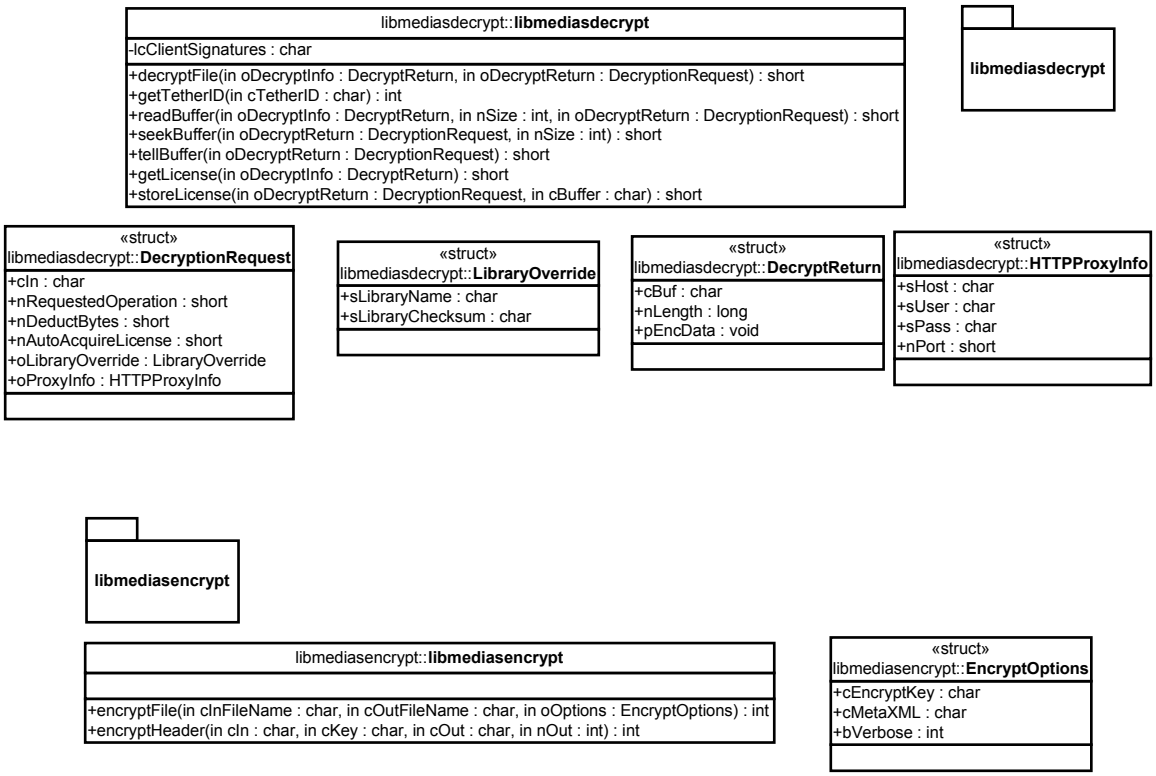
create-license is another sample program that demonstrates license creation. This function exercises the license creation functions of libmediasdecrypt . create-license will use the following command line parameters:

| Parameter | Meaning |
|---|---|
| -i <filename> | Name of the input file |
| -e <date> | Global license expiration date in MM/DD/YYYY format |
| -K <string> | The vendor key to use for the protected file |
| -a <action> | The action being granted a license. Usually “PLAY”, “STREAM”, or “BURN”. |
| -d <date> | The expiration of the particular permission in MM/DD/YYYY format |
| -t | Whether this license is active or not. If the license is not active, do not pass this parameter. |
| -d | Print verbose debug information |
| -h | Display brief usage instructions |
| -v | Display program version |
| -f | Force the license to overwrite any previous license vault entries. |

# Media-S DEVELOPER REFERENCE

This section will document the structures and functions exposed in libmediasdecrypt and provide samples of their usage.

To help understand the various functions, please recall the following diagram from the System Architecture document:




# STRUCTURES

There are two key structures used in Media-S that are described below:

| **Name**: decryptRequest |
| --- |
| **Description**: This structure is used to initialize Media-S decryption. Contains options about the type of decryption being requested. |

**Declaration**:

```
struct decryptRequest
{
   char  *cIn;
    int      nRequestedOperation;
    int      nDeductBytes;
    int      nAutoAcquireLicense;
    char  *cVendorKey;
    struct libraryInfoList *lOverride;
    struct httpProxyInfo   *oProxy;
};
```

**Member Detail**:

| Name | Type | Description |
|---|---|---|
| cIn | String | The name of the input file. |
| nRequestedOperation | int | The type of decryption operation being requested. Usually one of the constants PLAY_REQUEST, STREAM_REQUEST, BURN_REQUEST. |
| nDeductBytes | int | The number of bytes to decrypt before decrypting any usage meters in a user's license vault. |
| nAutoAcquireLicense | int | If a license to decrypt the content is not found, this field specifies whether a license should automatically be acquired via the renewal URL stated in the license vault. Setting this value to 0 means do not attempt to acquire the license, anything else is equivalent to true. |
| cVendorKey | char* | The vendor decryption key to use to decrypt the Media-S content. |
| lOrderrideList | struct libraryInfoList* | A list of acceptable override libraries. This member's documentation may change once this feature gets implemented in Media-S. |
| oProxy | struct httpProxyInfo* | HTTP proxy information that can be used for streaming content, or license acquisition requests. |

**Name**: decryptReturn

**Description**: This structure is use to maintain decryption state and is passed as a parameter to all decryption related functions.

**Declaration**:

```
struct decryptReturn
{
    unsigned char* cBuf;
    int  nLength;
    void* pEncData;
};
```

**Member Detail**:

| Name | Type | Description |
|---|---|---|
| cBuf | unsigned char* | Unencrypted buffer. |
| nLength | int | Length of the unencrypted buffer. |
| pEncData | void* | Pointer to internal decryption state information. This variable should not be accessed by developers. |

## METHODS

**Method**: getTetherID

**Description**: This function is used to determine the unique ID that locks a specific user to a specific machine. Ideally, this function will be a concatenation of some hardware and environment specific settings.

**Parameters:**

| Name | Type | Description |
|---|---|---|
| cTetherID | char** | Pointer to the character buffer that will be allocated and updated by this function. |

**Return Value**: integer, 0 for success

**Sample Usage**:

```
char *cTetherID = NULL;
getTetherID(&cTetherID);
```

**Method**: decryptFile

**Description**: This function initializes the decryption engine on a specific file.

**Parameters:**

| Name | Type | Description |
|---|---|---|

| | | |
|---|---|---|
| oDecryptInfo | DecryptRequest* | Decryption options. |
| cIn | char* | Name of the file or URL to decrypt. |
| oDecryptReturn | DecryptReturn** | Pointer to a structure that will be used for decryption-specific information. |

**Return Value**: integer, 0 for success

**Sample Usage**:

```
DecryptRequest *oInfo = NULL;

int nBytes;
DecryptReturn *oRet = NULL;
oInfo = (DecryptRequest*)malloc(sizeof(DecryptRequest));
//init our structures
oInfo->nRequestedOperation = PLAY_REQUEST;
oInfo->nDeductBytes = 0;
oInfo->nAutoAcquireLicense = 0;
oInfo->cVendorKey = NULL;
oInfo->lOverride = NULL;
oInfo->oProxy = NULL;

ret = decryptFile(oInfo, caIn, &oRet);

//see if we were successful
if (ret)
{
    printf (stderr, "ERROR: Decryption Unsuccessful\n");
}
```

**Method**: readBuffer

**Description**: This function reads bytes from an initialized DecryptReturn object.

**Parameters:**

| **Name** | **Type** | **Description** |
|---|---|---|
| oDecryptInfo | DecryptRequest* | Decryption options. |
| nSize | int | Number of bytes to read |
| oDecryptReturn | DecryptReturn** | Pointer to a structure that will be used for decryption-specific information. Must be initialized via decryptFile. |

**Return Value**: integer, 0 for success

**Sample Usage**:

```
ret = readBuffer(oInfo, 1024, &oRet);
```

| **Method**: seekBuffer | | |
|---|---|---|
| **Description**: This function seeks to a specific position in a DecryptReturn object. | | |
| **Parameters:** | | |
| **Name** | **Type** | **Description** |
| oDecryptReturn | DecryptReturn** | Pointer to a structure that will be used for decryption-specific information. Must be initialized via decryptFile. |
| nPos | int | Position to seek to. |
| **Return Value**: integer, 0 for success | | |
| **Sample Usage**:<br>`ret = seekBuffer(&oRet, 1000);` | | |

| **Method**: tellBuffer | | |
|---|---|---|
| **Description**: This function returns the specific position in the DecryptReturn buffer. | | |
| **Parameters:** | | |
| **Name** | **Type** | **Description** |
| oDecryptReturn | DecryptReturn** | Pointer to a structure that will be used for decryption-specific information. Must be initialized via decryptFile. |
| **Return Value**: integer, position in the buffer | | |
| **Sample Usage**:<br>`position = tellBuffer(&oRet);` | | |

| **Method**: getLicense |
|---|
| **Description**: This function attempts to get a license for a specific piece of content from the user's license vault or the license server. |
| **Parameters:** |

| Name | Type | Description |
|---|---|---|
| oDecryptInfo | DecryptInfo* | Decryption request object. |
| cIn | char* | File or URL that the license request is being made for |
| **Return Value**: integer, 0 for success | | |
| **Sample Usage**:<br>`ret = getLicense(oInfo, cFileName);` | | |

| **Method**: storeLicense | | |
|---|---|---|
| **Description**: This function stores a specific license blob into the user's license vault. | | |
| **Parameters:** | | |
| **Name** | **Type** | **Description** |
| oDecryptReturn | DecryptReturn** | Decryption result object |
| cLicenseBlob | char* | License blob to store |
| **Return Value**: integer, 0 for success | | |
| **Sample Usage**:<br>`ret = storeLicense(&oReturn, cLicenseBlob);` | | |