



LightScribe Linux API Reference Manual

About this Manual

This document is the official LightScribe Linux API (LSAPI) Reference Manual for LightScribe's Generation I Monochrome technology. Hewlett-Packard (HP) reserves the right to provide updates or revisions to this reference manual.

This document's purpose is to provide information about LSAPI to third-party software developers to support LightScribe Direct Disc Labeling within their Linux application(s).

Legal Disclaimers and Licensing Claims

Copyright

(c) Copyright 2006 Hewlett-Packard Development Company, LP

The *LightScribe Linux API Reference Manual* is published by Hewlett-Packard Company (Palo Alto, CA, USA). All rights reserved. Reproduction in whole or in part is prohibited without express, prior written permission of HP.

Disclaimer

The information contained herein is believed to be accurate as of the date of publication. However, HP shall not be liable for any damages, including indirect or consequential, from use of the LightScribe Direct Disc Labeling System or reliance on the accuracy of this document.

No warranties, either expressed or implied, are made by HP.

No support beyond this document will be provided by LightScribe or HP.

Licensing

A Trademark License from HP is required for the use of the LightScribe trademark.

Contact for Additional Information

For further information regarding the LightScribe Trademark License, please contact LightScribe through its web site (www.LightScribe.com).

Contents

1. General Information	3
1.1. Introduction	3
1.2. LightScribe System Architecture.....	3
1.3. LightScribe on Linux	4
1.4. Installing the LSS	4
1.5. Installing the Linux SDK	5
2. LightScribe API.....	6
2.1. Integrating LightScribe Functionality.....	6
2.2. LSAPI Use Guidelines	6
2.2.1. General Application Flow.....	6
2.2.2. Label Modes	7
2.2.3. Object Lifecycles.....	9
2.2.4. Media	9
2.2.5. LSS Updates.....	10
2.2.6. Drive Exclusivity.....	10
2.2.7. Bitmaps	11
2.2.8. Threading.....	12
2.2.9. Error Handling.....	12
2.2.10. Permissions	13
2.3. Use Cases.....	13
2.3.1. Enumerating LightScribe Drives	13
2.3.2. Identifying Media.....	14
2.3.3. Rendering	15
2.3.4. Labeling	15
2.3.5. Canceling a Label	16
2.3.6. Generic Labeling.....	16
2.3.7. LSS Updates.....	16
3. LSAPI Reference.....	18
3.1. Virtual Objects.....	18
3.1.1. LS_DiscPrintMgr Functions	18
3.1.2. LS_EnumDiscPrinters Functions.....	19
3.1.3. LS_DiscPrinter Functions	20
3.1.4. LS_DiscPrintSession Functions	31
3.1.5. General Functions	34
3.2. Data Structures	35
3.2.1. Typedefs	35
3.2.2. Enumerations.....	35
3.2.3. Structures	37
3.3. Error Codes.....	39
3.3.1. Groups of Errors	39
3.3.2. End User Messaging	40
3.3.3. Error Code Reference.....	43
3.4. Label Modes Specification	46
3.4.1. Title Mode Label	46
3.4.2. Content Mode Label	47
3.4.3. Full Mode Label	48

1. General Information

1.1. Introduction

The LightScribe Direct Disc Labeling solution allows users to generate high-resolution, silk-screen quality labels, similar to labels on commercial music CDs, quickly, easily and inexpensively using the same equipment used to write the data portion of the optical disc. LightScribe is designed to extend the capabilities of existing media, hardware, and software with minimal modifications to the existing components or manufacturing lines. The LightScribe drive creates an image on the label side of the LightScribe optical disc media by directly marking the surface with the infrared laser.

The LightScribe solution consists of a LightScribe enabled optical disc drive, LightScribe media, LightScribe System Software and LightScribe-enabled labeling software. The end user interacts with the LightScribe system via LightScribe enabled software.

This LightScribe API (LSAPI) Reference Manual is provided to enable the use of the LightScribe system in applications via a well known API.

1.2. LightScribe System Architecture

LightScribe provides a complete, integrated system consisting of:

- LightScribe capable optical disc drive (ODD)
- LightScribe media
- LightScribe enabled application software (ISV developed)
- LightScribe System Software (LightScribe developed)

Each part of the system plays an important role in the overall success of LightScribe Direct Disc Labeling. The LightScribe ODD provides the physical hardware and the appropriate firmware. The LightScribe media provides the physical label surface. The LightScribe enabled application provides the user interface to allow the user to interact with the LightScribe system and design their label. The LightScribe System Software (LSS) provides LSAPI, the LightScribe Print Engine, and all LightScribe-specific drive communications.

Users can create sophisticated graphical labels by creating their label design with labeling software, inserting a LightScribe disc in the drive upside down (label side facing the drive's laser), and burning the design onto the label side of the disc.

The LightScribe System Software (LSS) provides the exclusive access to LightScribe functionality on the ODD. Without the presence of the LSS, the ODD appears simply as a regular ODD to both the operating system and applications. In many respects, the LSS operates in the role of a driver as it is required to be present to access LightScribe functionality

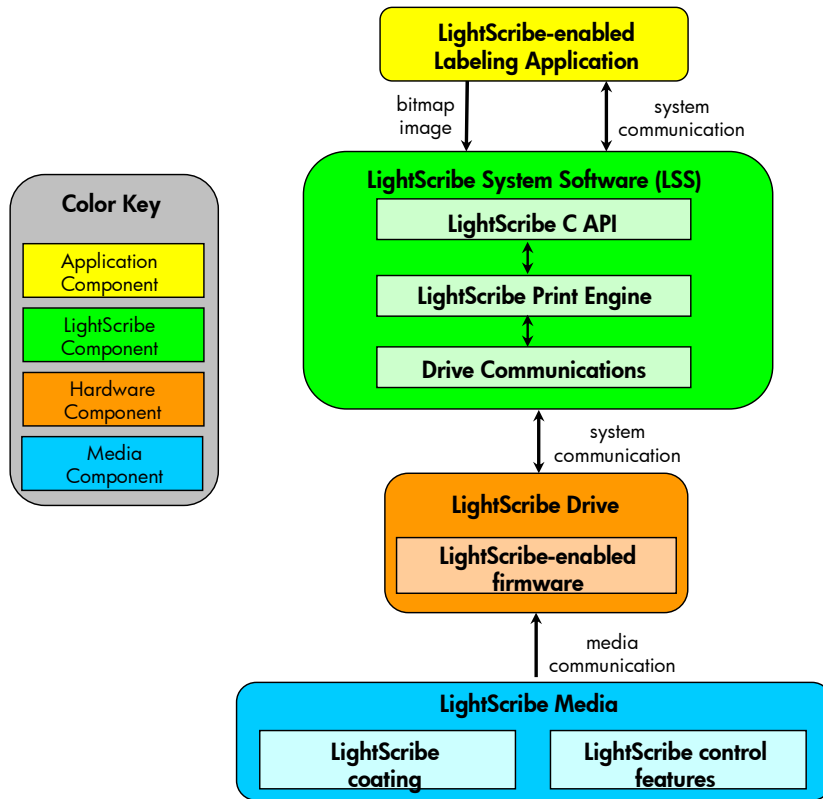


Figure 1.1. LightScribe System Architecture

1.3. LightScribe on Linux

LightScribe System Software is validated on SuSE 9.1, 9.2, 9.3 and 10.0. However, the LSS should install on any x86-based Linux distribution using Linux kernel 2.6 and RPM.

1.4. Installing the LSS

The LightScribe System Software (LSS) is available directly from the LightScribe web site (<http://www.lightscribe.com>). The LSS must be installed on the system to access LightScribe functionality. The LSS is delivered as a standard RPM package.

In addition to the LSAPI, print engine and drive communications, the LSS contains a set of optimized labeling parameters for all LightScribe drives and media. As such, the LSS needs to be periodically updated to support new drives and media. These updates are also available from the LightScribe web site as they become available.

There are several error codes defined that indicate that an update to the LSS is needed or recommended. When these codes are returned, the application must provide a method to retrieve and update the LSS. Further information can be found in the LSS Updates section.

1.5. Installing the Linux SDK

The Linux SDK contains everything required to integrate LightScribe functionality into your labeling application(s). The Linux SDK is delivered as a standard RPM package.

Please note that the SDK does not install the LSS directly. You must also install the LSS to enable using the SDK.

The SDK license is installed to `/usr/share/doc/lightscribe-sdk/SDKlicense.rtf`

Include files for the SDK are installed to `/usr/include`

SDK documentation is installed to `/usr/share/doc/lightscribe-sdk/docs`

The Isprint sample application is installed to `/usr/share/doc/lightscribe-sdk/sample/Isprint`

The LightScribe library, although not installed as part of this SDK, is installed to `/usr/lib`

2. LightScribe API

2.1. Integrating LightScribe Functionality

LSAPI presents a relatively simple and straightforward method to integrate LightScribe functionality into your labeling application.

LSAPI is defined in two header files, `lightscribe.h` and `lightscribe_errors.h`. To use LSAPI, include these two header files in your application and link against `liblightscribe.so`

In addition to the C-based API, a simple c++ wrapper is included as well. Include “`lightscribe_cxx.h`” in your application to use the c++ wrapper.

The Linux SDK also includes a sample command-line labeling application, `LSprint`, to show basic functionality and usage of LSAPI.

2.2. LSAPI Use Guidelines

LSAPI is a straightforward C-based API. At a high level, LSAPI provides functions for enumerating LightScribe drives on the system, querying drive and media information, providing print previews and actual LightScribe labeling.

Important notes:

- LSAPI is a C-based API and as such, there are no true objects or classes. However, throughout this document, we present the API in logical groups similar to a true object structure. This is both for ease of use of the API and to imply lifecycle to using these functions.
- For additional readability in this document, LSAPI virtual objects are often referred to versus their associated handle object. For instance, `LS_DiscPrinter` is often used versus `LS_DiscPrinterHandle`. Although the reader must keep in mind that `LS_DiscPrinter` is a virtual object and does not exist in LSAPI; only `LS_DiscPrinterHandle` can actually be used directly with LSAPI.

2.2.1. General Application Flow

The general use case for using LSAPI:

1. The user has created a label using the LightScribe-enabled labeling application and selects to burn the label
2. The application enumerates the LightScribe drives on the user's system
3. The application calls `LS_DiscPrinter_Validate` on each drive
4. The user selects the drive to use
5. The user selects any options for the label and selects to burn the label
6. The application probes and verifies that LightScribe media is in the drive and oriented label side down
7. The application starts a `LS_DiscPrintSession` on the drive
8. The application generates an in-memory bitmap of the image to label

9. The application calls `LS_DiscPrintSession_PrintDisc` passing in the bitmap
10. LSAPI executes the image preparation step
11. LSAPI executes the image labeling step
12. The application uses callbacks to track labeling progress during both of these steps
13. LSAPI finishes the label
14. The application destroys all LSAPI objects
15. The application continues after the label has completed

2.2.2. Label Modes

LightScribe technology labels in concentric rings starting from the inner radius moving to the outer radius. LSAPI provides three label modes, title, content and full, which optimize on the labeling technology.

The "Full Label" mode allows the maximum label size covering the entire disc surface. This mode also allows the greatest flexibility from the user perspective, as there are no restrictions on the amount of image data to be put on the label as long as it is contained within the disc boundaries. It is useful to note that only radii with label content will actually be labeled.



Figure 2.1. Full Label Mode example

The "Content Label" mode provides a moderately wide band in which the label content must fit. Templates based upon the "Content Label" contain a title and content listings arranged as curved text within this band. All label content outside of the defined "Content Label" band will be clipped when specifying this label mode.

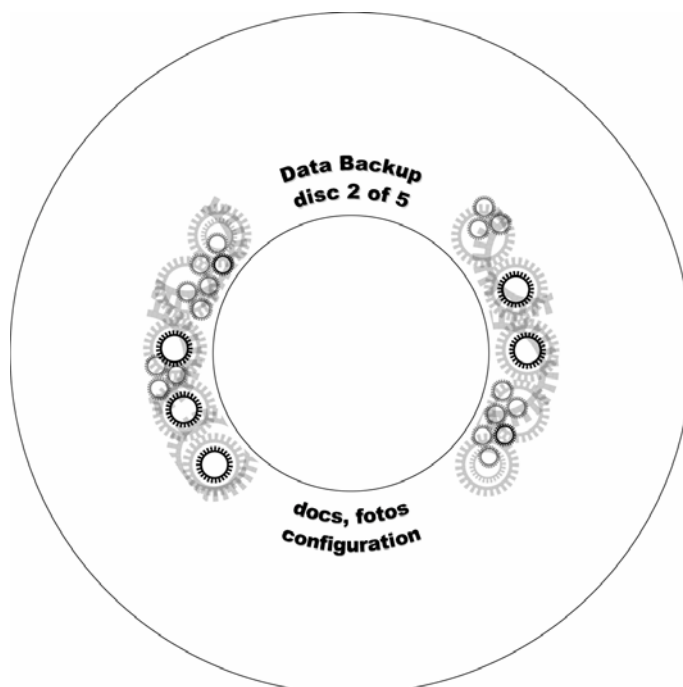


Figure 2.2. Content Label Mode example

The "Title Label" mode provides a smaller band in which the label content must fit. In addition, the "Title Label" provides the fastest label performance. Templates based upon the "Title Label" contain an upper and lower band containing limited content such as title and date. All label content outside of the defined "Title Label" band will be clipped when specifying this label mode.



Figure 2.3. Title Label Mode example

The application specifies which label mode to use while labeling via the `LS_LabelMode` enumeration.

Exact specifications for each label mode may be found in the Label Modes Specification section.

2.2.3. Object Lifecycles

In general, all LSAPI objects should be used and then destroyed as soon as possible to allow other applications and the system access to the drive(s). In addition, LSAPI does not track any Plug and Play changes that may occur. It is strongly suggested that the application creates and destroys LSAPI objects only when needed.

During the lifetime of the `LS_EnumDiscPrinters` enumeration, an instance of `LS_DiscPrinter` exists for each LightScribe-enabled drive on the system. The `LS_DiscPrinter` objects maintain an open channel of communication with the drive, which, depending upon the drive, may interfere with other software or communications with the drive. It is important to obtain only transient objects of `LS_EnumDiscPrinters` and immediately extract the necessary information. Similarly, a `LS_DiscPrinter` instance should only be kept around only when necessary.

If an object of `LS_EnumDiscPrinters` exists when another object is created via `LS_DiscPrintMgr` then the second object of `LS_EnumDiscPrinters` will be identical to the first. Any drives that have been added or removed between the calls to `LS_EnumDiscPrinters` will not show up or go away. That is, `LS_DiscPrinter` returned in the subsequent enumeration will be the same as those supplied in the previous enumerations. This behavior is necessary to ensure that the `LS_DiscPrinter` objects in all instances of `LS_EnumDiscPrinters` maintain an open channel of communication with the associated drive.

If an application needs a new and accurate enumeration reflecting the current set of drives, the application needs to destroy all existing `LS_EnumDiscPrinters` objects before creating a new instance of `LS_EnumDiscPrinters`. When such a re-enumeration takes place, the underlying hardware is re-examined so it will reflect the current state of the system

2.2.4. Media

The LightScribe system allows unique identification of the selected drive and LightScribe media. The LSS contains optimized drive and media labeling parameters. The print engine uses both the selected drive capabilities and the LightScribe media in the drive to generate the best possible label for the specific drive and media combination. As new drives and media types become available, updates to the LSS are required to use optimized labeling on these new media types (See LSS Updates section)

Although the LSS may not contain the optimized labeling parameters for the specific media being used, LSAPI can often label the media, but possibly at reduced quality, by specifying “generic” labeling. Please refer to the Generic Labeling section for further information.

In addition to labeling, generating a print preview also requires the size and characteristics of the LightScribe media in the drive to provide an accurate preview image. Parameters on the `LS_DiscPrintSession_PrintPreview` function allow the application to either attempt to identify the media in the drive or to ignore the media for generating the preview image.

It is suggested that applications indicate to their users as soon as possible to insert their LightScribe media label side down for the best and most accurate labeling experience.

2.2.5. LSS Updates

The LSS contains optimized drive and media labeling parameters. As such, to support new drives and media types, updates to the LSS may be required.

A set of error codes and a mechanism for retrieving the location to look for updates are built into LSAPI. This set of errors constitute the LSS Update Error Group. When any of these error codes are returned, the application should prompt the user whether to look for an update or not. In some cases, labeling may still be possible without an update such as with Generic Labeling. In other cases, an update may be required before labeling can commence.

Note that an update is never guaranteed to resolve any of these error conditions.

LSAPI provides a mechanism to retrieve the location the user should be directed sent to when looking for an update. For details, refer to the LSS update use case in section LSS Updates.

2.2.6. Drive Exclusivity

LSAPI drive exclusivity is provided only between applications using LSAPI. LSAPI does not provide global, system-wide drive exclusivity. Given this restriction, the goals for exclusivity are:

- Allow an application to gain exclusive use to a drive not just for the labeling process but also for some of the precursor steps such as tray control and media detection
- Fail attempts to gain exclusive access immediately rather than blocking
- Work between separate LightScribe-enabled applications
- Work across threads in an application and even within a single thread
- Provide a mechanism to release exclusivity even if the application crashes

These goals are met using global semaphores which are created as needed by LSAPI. This means that only one instance `LS_DiscPrinter` can have exclusive access to a drive across all programs running on the machine. This access is automatically requested whenever a side-effecting operation is invoked, such as tray behavior, media detection and labeling. If the exclusive use is not granted, then the `LS_E_DriveInUse` error is returned. If exclusive use is granted, it is released at the end of the function call.

An application can explicitly request an exclusive lock lasting longer than a single function call by using `LS_DiscPrinter_AddExclusiveUse`. `LS_DiscPrinter_ReleaseExclusiveUse` is used to release this lock. These calls can be nested providing the number of calls to `LS_DiscPrinter_AddExclusiveUse` matches the number of calls to `LS_DiscPrinter_ReleaseExclusiveUse`.

Note that when the `LS_DiscPrinter` object is destroyed, it releases any outstanding exclusive locks, however, this destruction may not happen when one expects as `LS_EnumDiscPrinters` also hangs onto references to enumerated objects. This cleanup feature should be treated as an extra robustness feature and not relied upon. The application must always explicitly release exclusive use.

There are additional exclusive locks around the creation of a print session such that only one `LS_DiscPrintSession` bound to a particular drive can exist at one time. Additional calls to `LS_DiscPrinter_OpenPrintSession` will fail with a `LS_E_DriveInUse` error. Similarly, when a print session exists on a drive, no new instances of `LS_DiscPrinter` bound to that drive can call `LS_DiscPrinter_AddExclusiveUse`.

Which leads to a general application strategy for exclusivity:

- An application should call `LS_DiscPrinter_AddExclusiveUse` on the `LS_DiscPrinter` object representing the drive as soon as the application requires. A failure to acquire the lock should be handled with an error dialog.
- After acquiring the lock, the application has effective ownership of the drive, with respect to other applications using LSAPI.
- The application can open/close drive trays, lock/unlock drive trays, query for media, etc.
- Once the application is ready to preview or label, open a `LS_DiscPrintSession` on the drive by calling `LS_DiscPrinter_OpenPrintSession`.
- While the `LS_DiscPrintSession` is active, the application should avoid making calls to `LS_DiscPrinter` that could conflict with the call to `LS_DiscPrintSession`. This effectively means that `LS_DiscPrinter` should not be used until labeling or previewing is completed and the `LS_DiscPrintSession` has been destroyed.
- After labeling is completed, the application destroys the `LS_DiscPrintSession`. The application may also unlock the tray and eject the disc on behalf of the user.
- The application releases the exclusive use on the drive by calling `LS_DiscPrinter_ReleaseExclusiveUse`.
- Finally, the application releases all outstanding references to `LS_DiscPrinter` and `LS_EnumDiscPrinters` that created it if the enumeration has not already been released.

In the case of a catastrophic failure of the application or in appropriate signal handlers (SIGKILL, SIGILL, etc) the application must call `LS_LastChanceCleanup`. This function provides cleanup of the global semaphores and other LSAPI structures.

2.2.7. Bitmaps

The image to label must be passed to LSAPI as an in-memory bitmap. Currently, only Windows-style of bitmaps are supported by LSAPI:

- Bitmaps have 24bit depth of color information. Bitmaps should be presented to LSAPI in full color as they will be converted to grayscale as part of the LSAPI image processing pipeline and will be tuned to best match the drive and media capabilities.
- Bitmaps are Windows bitmap formats V3, V4 and V5. Bitmaps are uncompressed DIB format; JPEG and PNG compression are not supported
- Colorspace support in V4 and V5 headers and color profile support in V5 headers is ignored
- Bitmaps are upside down or right-side-up. Positive heights and negative heights in the `biHeight` member of the `BITMAPINFOHEADER` are supported with a positive height meaning a classic OS/2-style upside down bitmap
- Bitmaps are not scaled up or down unless requested via LSAPI parameters
- Bitmaps have white, RGB(255,255,255), as the background. Black, RGB(0,0,0), is the maximum darkness

- Bitmaps have the `biXPelsPerMeter` and `biYPelsPerMeter` fields in their `BITMAPINFOHEADER` set to the resolution of the image in pixels per meter. If these are unset then the caller will receive a `LS_E_InvalidSource` error when labeling or previewing the image. It is incorrect to set these to the bitmaps' resolution in pixels per inch.
- The bitmap header is in native-endian format
- The bitmap image data is in little-endian format

The caller of `LS_DiscPrintSession_PrintDisc` must guarantee that the bitmap data and header must remain at the specified memory locations until the call returns.

2.2.8. Threading

In general, an application should only use LSAPI from one thread at a time. More complex interactions are possible but strongly discouraged.

Specifically:

- For all objects other than `LS_DiscPrintSession`, if two threads attempt to call into methods on the same object, the second thread will be blocked until the first completes.
- For `LS_DiscPrintSession`, re-entrant calls will not be blocked but will return with the `LS_E_ReentrantCall` error.
- Progress callbacks during labeling are invoked on the same thread that made the currently active `LS_DiscPrintSession_PrintDisc` call.
- Only one thread in a single process can use the print engine to either label or preview a label. Other threads trying to call the relevant `LS_DiscPrintSession` functions will fail with `LS_E_ReentrantCall` error.
- Overall, an application can pass objects freely between threads, but only one thread will be able to make effective use of those objects at one time.

One of the implications of this threading model is that while an application can pass LSAPI objects between threads, calling different methods from different threads is not the preferred mode of use. It is better if only one thread uses an object at one time, making all the calls to that object.

The preferred model for using LSAPI is to pass `LS_DiscPrintSession` to a worker thread created specifically for the labeling process. This worker thread initiates the label process via `LS_DiscPrintSession_PrintDisc` and receives all callbacks leaving the main thread free to update the GUI.

It is strongly recommended that all multithreaded code is tested on both multi-CPU and hyperthreaded systems.

2.2.9. Error Handling

All LSAPI functions return a `LS_Error` value which must be checked for `LS_OK` by the application for each and every function. When an error occurs, the error is logged to the `~/lightscribe/log`. The log messages may be useful for debugging purposes.

Please refer to the Error Codes section for additional information. Please note the groups of errors that each function may return. Any function may return any of the Global error group. Specific functions may return other errors as well.

In addition, to help create an easy to use and best possible labeling user experience, please refer to the section on suggested end user error messaging. It is strongly encouraged that all applications use this messaging as a minimum for all listed error conditions.

2.2.10. Permissions

Due to the nature of LSAPI SCSI commands used to communicate with the LightScribe drive, the application calling LSAPI must have effective permissions of root. It is recommended that the application has the setuid bit set (u+s) with root ownership to enable this.

It is strongly suggested to the developer to attempt to minimize the security risks of running as root

To help minimize potential security risks, the developer may implement strategies to reduce the exposure of running with root permissions:

- Change the effective userid of the application to non-root permissions as much as possible, and certainly when executing the UpdateShellCommand see LSS Updates section, by using calls such as seteuid(2).
- Isolate all calls into LSAPI into a separate application, with appropriate setuid bits set, to allow the labeling application to run as the normal user.

In addition to these strategies, the application must take special care when executing the UpdateShellCommand on behalf of the user. It is strongly suggested that the effective user id is set to non-root permissions before executing this command as this could present a significant security risk.

2.3. Use Cases

For convenience, we outline some of the common LSAPI use cases.

2.3.1. Enumerating LightScribe Drives

Every LightScribe-enabled device that is compatible with LSAPI has one corresponding LS_DiscPrinter instance. LS_DiscPrinter may be used to probe for media, query the drive's capabilities, start a disc print session, etc. Instances of LS_DiscPrinter may be obtained via the LS_EnumDiscPrinters object.

For example, to retrieve a list of all LightScribe-enabled drives on the system:

1. Create an instance of LS_DiscPrintMgr
2. Use LS_DiscPrintMgr to create and retrieve the LS_EnumDiscPrinters object which contains an enumeration of all LightScribe-enabled drives
3. Use LS_EnumDiscPrinters to retrieve the total number of LightScribe-enabled drives on the system
4. Use LS_EnumDiscPrinters to retrieve each instance of LS_DiscPrinter representing each physical LightScribe-enabled drive
5. Call LS_DiscPrinter_Validate on each LS_DiscPrinter instance to validate that the drive is ready and capable of labeling. Save any errors for presenting to the user.
6. Retrieve the drive string of each LS_DiscPrinter by calling LS_DiscPrinter_GetPrinterDisplayName. You can retrieve additional information about the drive by calling other LS_DiscPrinter functions.

7. Destroy each instance of `LS_DiscPrinter` as soon as you are finished using functions calling into the specific drive
8. Destroy the `LS_EnumDiscPrinters` and `LS_DiscPrintMgr` objects.
9. In the application GUI, display the list of drives found on the system.
10. Recreate the `LS_DiscPrintMgr` and `LS_EnumDiscPrinters` objects and recreate the desired `LS_DiscPrinter` object for further use.

2.3.2. Identifying Media

An application must verify that the media in the drive is in fact LightScribe media and it is ready for labeling including that it is oriented label side down. Various LSAPI error codes are defined to help the application do this.

Note that any function querying for media can potentially be a time consuming function as the drive will attempt to read the media identification data around the center of the disc if it has not already done so.

1. Obtain the `LS_DiscPrinter` object for the specified drive
2. Call `LS_DiscPrinter_GetCurrentMedia` carefully capturing the `LSError` return value
3. The application proceeds accordingly based on the `LSError` value, potentially calling `LS_DiscPrinter_GetCurrentMedia` again.

The application can then take the appropriate action depending upon the return value and in the case of a `LS_OK` return value, the contents of the `LS_MediaInformation` structure.

For the specific return values of:

- `LS_E_InvalidMediaOrientation` indicates that LightScribe media was found in the drive but it is not oriented with the label side down. The application must prompt the user to flip their LightScribe disc so labeling can commence.
- `LS_E_NoMedia` indicates that no media was found in the drive
- `LS_E_NonLightScribeMedia` indicates that media was found in the drive but the media is not LightScribe media. This error code can also be returned if the LightScribe disc is dirty or has smudges.
- `LS_E_UnableToReadMediaParams` indicates that media was found in the drive but LSAPI was not able to read the media information. This can be caused by a dirt or smudges on the disc
- `LS_E_UnsupportedMedia`, `LS_E_UnsupportedMediaId` indicate media which has been identified but which LSAPI does not have the optimized labeling parameters for. The application must present an option to the user to look for a LSS update, proceed with generic labeling or cancel. See the Generic Labeling section for further information.
- `LS_E_IncompatibleMedia`, `LS_E_UnsupportedMediaNoUpdate`, `LS_E_MediaNotSupportedByDrive`, `LS_E_InvalidMediaParams` all indicate that the media in the drive is LightScribe media but is incompatible with the selected drive. Labeling is not possible with this combination of drive and media

It is strongly suggested that the application is developed with this media identification step in a loop as several attempts of identifying media including potential user interaction, such as opening the drive tray or prompting the user to flip the media label side down, may be required.

To aid the application in this step, suggested end user messaging around each of these errors are in the End User Messaging section.

2.3.3. Rendering

When an application is ready to label a disc, it should render the desired image to an in-memory bitmap of the appropriate size and resolution. This can be determined by function calls to `LS_DiscPrinter`.

1. The application takes the user's choice of drive and queries the drive for media
2. The application queries the drive for its capabilities with the chosen media, in particular, the resolution of the device in the chosen label mode and what the inner and outer radii of the printable area will be.
3. The application calculates what the appropriate bitmap size for the label is either by rendering at the full resolution of the label at the current label quality or rendering at a chosen quality and let the print engine scale.
4. Create an in-memory Windows bitmap of the rendered image
5. Verify that the `biXPelsPerMeter` and `biYPelsPerMeter` fields in the `BITMAPINFOHEADER` are set appropriately to pixels per meter.
6. Pass the in-memory bitmap to LSAPI via the `LS_DiscPrintSession_PrintDisc` function as listed below
7. Delete the in-memory bitmap after labeling is completed

2.3.4. Labeling

The following use case covers labeling an in-memory bitmap.

1. Obtain the `LS_DiscPrinter` object for the specified drive.
2. Gain exclusive use of the drive using `LS_DiscPrinter_AddExclusiveUse`.
3. Verify that the media in the drive is LightScribe media and insert label side down, prompting the user as necessary. See the Identifying Media use case for more details.
4. Lock the drive tray with `LS_DiscPrinter_LockDriveTray`. This will prevent the user from ejecting the drive tray during labeling.
5. Call `LS_DiscPrinter_OpenPrintSession` to obtain a `LS_DiscPrintSession` object.
6. Create a `LS_PrintCallbacks` structure and set the function pointers appropriately to the functions in your application to receive the callbacks.
7. Call `LS_DiscPrintSession_SetProgressCallback` to set the callbacks for the label process.
8. Call `LS_DiscPrintSession_PrintDisc` passing in the in-memory bitmap to start the labeling process. Since `LS_DiscPrintSession_PrintDisc` will block until the labeling is complete. It is suggested to call `LS_DiscPrintSession_PrintDisc` from a specific label thread independent of the GUI thread.
9. While labeling, receive callbacks on the specified callback functions and use this information to update the application GUI for label progress.
10. When `LS_DiscPrintSession_PrintDisc` returns, signal the application GUI that the label is complete and perform any additional user interactions.
11. Destroy the `LS_DiscPrintSession` by calling `LS_DiscPrintSession_Destroy`.
12. Destroy the `LS_DiscPrinter` object by calling `LS_DiscPrinter_Destroy`.

2.3.5. Canceling a Label

To cancel a label, the application must provide the appropriate value via the `AbortLabel` callback.

1. LSAPI is burning a label
2. The application is watching label progress via the callbacks set in the `LS_PrintCallbacks` structure.
3. The user selects to cancel the label via the application GUI
4. In the subsequent call from LSAPI to the `AbortLabel` callback, the application provides a true value
5. LSAPI cancels the labeling process and `LS_DiscPrintSession_PrintDisc` returns with a `LS_E_Aborted` error code
6. The application indicates to the user that the label has been successfully canceled.

The `AbortLabel` callback is called both during the image preparation step and the actual labeling step. Either step supports canceling.

Please note that canceling during the labeling step will result in a partially labeled disc.

2.3.6. Generic Labeling

In the scenario where the LSS does contain the optimized labeling parameters for the media in the drive, generic labeling may still be possible on the media. This scenario is possible whenever LSAPI returns either `LS_E_UnsupportedMedia` or `LS_E_UnsupportedMediaId` from any method which attempts to identify media such as `LS_DiscPrinter_GetCurrentMedia` or `LS_DiscPrintSession_PrintDisc`.

In this scenario, the application should always promote attempting to get a LSS update to provide the best possible labeling. However, since a LSS update will almost assuredly require internet access and may introduce a delay, the application should always provide several options:

1. Look for a LSS update to provide the optimal labeling (recommended)
2. Label using generic labeling
3. Cancel the label

If the user selects to proceed with generic labeling, the application should call `LS_DiscPrintSession_PrintDisc` again setting the appropriate `LS_MediaOptimizationLevel`.

Refer to the LSS Updates section for the use case to look for a LSS update.

2.3.7. LSS Updates

Any error code defined in the LSS Update Error Group indicates that an LSS update may be either suggested or required. In some cases, an update is required before continuing (such as missing resource information) in other cases, and update is recommended (such as generic labeling).

LSAPI provides functions to indicate to the user where to go look for a LSS update.

1. The user selects to look for a LSS update
2. The application calls `LS_GetUpdateShellCommandSize` to retrieve the size of the update shell command
3. The application allocates a char string of the appropriate size

4. The application calls `LS_GetUpdateShellCommand` to retrieve the actual shell command
5. The application changes the effective user id to a non-root user id (if applicable)
6. The application passes this string to the `system(3)` command (or similar command) thereby executing the command on behalf of the user
7. The application changes the effective user id back to a root user id (if applicable)

In the default scenario, the update shell command takes the user to the LightScribe web site for updates. The application should indicate to the user that an Internet connection will be required to look for the update.

Note that since an application using LSAPI must be run with effective root permissions, it is strongly suggested that the call to execute the contents of the `UpdateShellCommand` are not run with these same root permissions as this presents a significant security risk. Please refer the to [Permissions](#) section for more information.

3. LSAPI Reference

3.1. Virtual Objects

All LSAPI functions return a `LS_Error` value which must be checked for `LS_OK` for each and every call. Please refer to the Error Codes section for additional information. Please note the classes of errors that each function may return. Any function may return any of the Global error group. Specific functions may return other errors as well.

3.1.1. LS_DiscPrintMgr Functions

This virtual object allows an application to create an enumeration all LightScribe disc printers on the system. This object is the entry point for any application's use of LightScribe.

Functions

- LSError LS_DiscPrintMgr_Create(LS_DiscPrintMgrHandle* mgrHandle)
- LSError LS_DiscPrintMgr_Destroy(LS_DiscPrintMgrHandle* mgrHandle)
- LSError LS_DiscPrintMgr_EnumDiscPrinters(LS_DiscPrintMgrHandle mgrHandle, LS_EnumDiscPrintersHandle* enumHandle)

3.1.1.1. LSError LS_DiscPrintMgr_Create(LS_DiscPrintMgrHandle* mgrHandle)

This function creates and returns a handle to a `LS_DiscPrintMgr`. The returned object can be used to enumerate all LightScribe disc printers on the system.

The returned object must be destroyed as soon as the enumeration of disc printers has been retrieved.

Parameters

<i>mgrHandle</i>	A LS_DiscPrintMgrHandle to store the returned LS_DiscPrintMgr
-------------------------	---

Returns

Global error group

3.1.1.2. LSError LS_DiscPrintMgr_Destroy(LS_DiscPrintMgrHandle* mgrHandle)

This function destroys and deallocates the `LS_DiscPrintMgr` object pointed to by the passed in handle.

Parameters

<i>mgrHandle</i>	The LS_DiscPrintMgrHandle of the LS_DiscPrintMgr to destroy
-------------------------	---

Returns

Global error group

**3.1.1.3. LSError LS_DiscPrintMgr_EnumDiscPrinters(LS_DiscPrintMgrHandle mgrHandle
LS_EnumDiscPrintersHandle* enumHandle)**

This function returns an enumeration of all of the LightScribe disc printers on the system using the specified LS_DiscPrintMgr. If there are no suitable disc printers available, this function will return an empty enumeration.

The resulting enumeration should be used and immediately destroyed using LS_EnumDiscPrinters_Destroy. The list of available LightScribe disc printers may change due Plug and Play events. The enumeration does not update itself in this circumstance. Instead, the enumeration is a snapshot of the supported devices at the time of the enumeration.

If a disc printer is removed from the local machine, handles to LS_DiscPrintMgrHandle remain valid even though the physical device has been removed. Subsequent function calls using this handle will return error codes such as LS_E_DriveChanged.

Parameters

<i>mgrHandle</i>	The LS_DiscPrintMgrHandle to use to retrieve the disc printers enumeration
<i>enumHandle</i>	A LS_DiscPrintMgrHandle to store the returned LS_EnumDiscPrinters

Returns

Global error group

3.1.2. LS_EnumDiscPrinters Functions

This virtual object represents the LightScribe disc printer enumeration as returned from LS_DiscPrintMgr_EnumDiscPrinters(). This enumeration contains a list of all valid LightScribe disc printers on the system at the time that the enumeration was created.

Functions

- LSError LS_EnumDiscPrinters_Count(LS_EnumDiscPrintersHandle enumHandle, unsigned long* count)
- LSError LS_EnumDiscPrinters_Destroy(LS_EnumDiscPrintersHandle* enumHandle)
- LSError LS_EnumDiscPrinters_Item(LS_EnumDiscPrintersHandle enumHandle, unsigned long index, LS_DiscPrinterHandle* printerHandle)

**3.1.2.1. LSError LS_EnumDiscPrinters_Count(LS_EnumDiscPrintersHandle enumHandle
unsigned long* count)**

This function returns the number of LightScribe disc printers contained in the enumeration pointed to by the EnumDiscPrintersHandle.

Parameters

<i>enumHandle</i>	The EnumDiscPrintersHandle to use
<i>count</i>	The returned number of disc printers in the enumeration

Returns

Global error group

3.1.2.2. LSError LS_EnumDiscPrinters_Destroy(LS_EnumDiscPrintersHandle* enumHandle)

This function destroys and deallocates the LS_EnumDiscPrinters object pointed to by the passed in handle.

Parameters

mgrHandle A LS_EnumDiscPrintersHandle of the LS_EnumDiscPrinters to destroy

Returns

Global error group

**3.1.2.3. LSError LS_EnumDiscPrinters_Item(LS_EnumDiscPrintersHandle enumHandle
unsigned long index
LS_DiscPrinterHandle* printerHandle)**

This function retrieves the specified LS_DiscPrinter object from the enumeration.

Note that this is a 0-based enumeration where the first object in the enumeration is at index 0

Parameters

enumHandle The LS_EnumDiscPrintersHandle to use
index The 0-based index number of the object to retrieve
printerHandle A LS_DiscPrinterHandle to store the returned LS_DiscPrinter

Returns

Global error group

3.1.3. LS_DiscPrinter Functions

This virtual object offers access to a single LightScribe-enabled drive. This object can be used to inquire the device capabilities as well as information about the media it currently contains. This object may also be used to begin a printing session on the drive by calling.

Please refer to the Object Lifecycles and Drive Exclusivity sections for proper use of LS_DiscPrinter functions.

Functions

- LSError LS_DiscPrinter_AddExclusiveUse(LS_DiscPrinterHandle printerHandle)
- LSError LS_DiscPrinter_CloseDriveTray(LS_DiscPrinterHandle printerHandle)
- LSError LS_DiscPrinter_Destroy(LS_DiscPrinterHandle* printerHandle)
- LSError LS_DiscPrinter_GetCurrentMedia(LS_DiscPrinterHandle printerHandle, LS_MediaOptimizationLevel optimizationLevel, LS_MediaInformation* currentMedia)

- `LSError LS_DiscPrinter_GetDriveInnerRadius(LS_DiscPrinterHandle printerHandle, long* innerRadius)`
- `LSError LS_DiscPrinter_GetDriveOuterRadius(LS_DiscPrinterHandle printerHandle, long* outerRadius)`
- `LSError LS_DiscPrinter_GetLabelRegionInnerRadius(LS_DiscPrinterHandle printerHandle, LS_LabelMode mode, LS_MediaOptimizationLevel optimizationLevel, long* innerRadius)`
- `LSError LS_DiscPrinter_GetLabelRegionOuterRadius(LS_DiscPrinterHandle printerHandle, LS_LabelMode mode, LS_MediaOptimizationLevel optimizationLevel, long* outerRadius)`
- `LSError LS_DiscPrinter_GetPrinterCapabilities(LS_DiscPrinterHandle printerHandle, LS_DeviceCapabilitiesEnum* capabilities)`
- `LSError LS_DiscPrinter_GetPrinterDisplayNameSize(LS_DiscPrinterHandle printerHandle, size_t* stringSize)`
- `LSError LS_DiscPrinter_GetPrinterDisplayName(LS_DiscPrinterHandle printerHandle, char* displayName)`
- `LSError LS_DiscPrinter_GetPrinterPathSize(LS_DiscPrinterHandle printerHandle, size_t* stringSize)`
- `LSError LS_DiscPrinter_GetPrinterPath(LS_DiscPrinterHandle printerHandle, char* path)`
- `LSError LS_DiscPrinter_GetPrinterProductNameSize(LS_DiscPrinterHandle printerHandle, size_t* stringSize)`
- `LSError LS_DiscPrinter_GetPrinterProductName(LS_DiscPrinterHandle printerHandle, char* productName)`
- `LSError LS_DiscPrinter_GetPrinterVendorNameSize(LS_DiscPrinterHandle printerHandle, size_t* stringSize)`
- `LSError LS_DiscPrinter_GetPrinterVendorName(LS_DiscPrinterHandle printerHandle, char* vendorName)`
- `LSError LS_DiscPrinter_GetPrintResolution(LS_DiscPrinterHandle printerHandle, LS_PrintQuality quality, LS_MediaOptimizationLevel optimizationLevel, unsigned long* resolution)`
- `LSError LS_DiscPrinter_LockDriveTray(LS_DiscPrinterHandle printerHandle)`
- `LSError LS_DiscPrinter_OpenDriveTray(LS_DiscPrinterHandle printerHandle)`
- `LSError LS_DiscPrinter_OpenPrintSession(LS_DiscPrinterHandle printerHandle, LS_DiscPrintSessionHandle* sessionHandle)`
- `LSError LS_DiscPrinter_ReleaseExclusiveUse(LS_DiscPrinterHandle printerHandle)`
- `LSError LS_DiscPrinter_UnlockDriveTray(LS_DiscPrinterHandle printerHandle)`

- `LSError LS_DiscPrinter_Validate(LS_DiscPrinterHandle printerHandle)`

3.1.3.1. **LSError LS_DiscPrinter_AddExclusiveUse(LS_DiscPrinterHandle printerHandle)**

This function acquires an exclusive lock to the LightScribe drive. This function can be called repeatedly, provided that the number of times it is called matches the number of times the lock is released, with `LS_DiscPrinter_ReleaseExclusiveUse`.

All functions that communicate with the drive will themselves add and release the exclusive lock. This function allows exclusivity beyond a single function.

Parameters

printerHandle The `LS_DiscPrinterHandle` to use

Returns

Global error group

`LS_E_DriveInUse`

3.1.3.2. **LSError LS_DiscPrinter_CloseDriveTray(LS_DiscPrinterHandle printerHandle)**

This function closes the LightScribe drive tray. Closure does not work on slimline drives or other systems with non-motorized drives.

This function requires exclusive use of the drive.

Parameters

printerHandle The `LS_DiscPrinterHandle` to use

Returns

Global error group

`LS_E_DriveInUse`

3.1.3.3. **LSError LS_DiscPrinter_Destroy(LS_DiscPrinterHandle* printerHandle)**

This function destroys and deallocates the `LS_DiscPrinter` object pointed to by the passed in handle.

Parameters

printerHandle A `LS_DiscPrinterHandle` of the `LS_DiscPrinter` to destroy

Returns

Global error group

**3.1.3.4. LSError LS_DiscPrinter_GetCurrentMedia(LS_DiscPrinterHandle printerHandle
LS_MediaOptimizationLevel optimizationLevel
LS_MediaInformation* currentMedia)**

This function will retrieve information about the current media in the LightScribe drive, if any. This function can potentially be a time consuming function as the drive will attempt to read the media identification data around the center of the disc if it has not already done so.

This function will always fill in the passed in LS_MediaInformation structure. The application must check the mediaPresent field to determine if media is actually present.

Please note that there is no way to cancel this call and it will block until the media identification sequence is complete.

This function requires exclusive use of the drive.

Parameters

<i>printerHandle</i>	The LS_DiscPrinterHandle to use
<i>optimizationLevel</i>	The LS_MediaOptimizationLevel to use. See Generic Labeling
<i>currentMedia</i>	A pointer to a LS_MediaInformation to be filled out

Returns

Global error group
Drive access error group
Media error group
LS_E_DriveInUse

**3.1.3.5. LSError LS_DiscPrinter_GetDriveInnerRadius(LS_DiscPrinterHandle printerHandle
long* innerRadius)**

This function retrieves the drive's inner printing radius in microns. This is the minimum possible radius supported by the drive.

Parameters

<i>printerHandle</i>	The LS_DiscPrinterHandle to use
<i>innerRadius</i>	A long* to receive the drive's inner radius

Returns

Global error group
Drive access error group

3.1.3.6. LSError LS_DiscPrinter_GetDriveOuterRadius(LS_DiscPrinterHandle printerHandle long* outerRadius)

This function retrieves the label region inner radius for the current media and specified label mode. This function can potentially be a time consuming function as the drive will attempt to read the media identification data around the center of the disc if it has not already done so. If there is no media present, innerRadius will be set to 0.

This function requires exclusive use of the drive.

Parameters

<i>printerHandle</i>	The LS_DiscPrinterHandle to use
<i>outerRadius</i>	A long* to receive the drive's outer radius

Returns

Global error group

Drive access error group

3.1.3.7. LSError LS_DiscPrinter_GetLabelRegionInnerRadius(LS_DiscPrinterHandle printerHandle LS_LabelMode mode LS_MediaOptimizationLevel optimizationLevel long* innerRadius)

This function retrieves the label region inner radius for the current media and specified label mode. This function can potentially be a time consuming function as the drive will attempt to read the media identification data around the center of the disc if it has not already done so. If there is no media present, innerRadius will be set to 0.

This function requires exclusive use of the drive.

Parameters

<i>printerHandle</i>	The LS_DiscPrinterHandle to use
<i>mode</i>	A LS_LabelMode specifying which label mode to use
<i>optimizationLevel</i>	The LS_MediaOptimizationLevel to use. See Generic Labeling
<i>innerRadius</i>	A long* to receive the label region inner radius

Returns

Global error group

Drive access error group

Media error group

LS_E_DriveInUse

**3.1.3.8. LSError LS_DiscPrinter_GetLabelRegionOuterRadius(LS_DiscPrinterHandle printerHandle
LS_LabelMode mode
LS_MediaOptimizationLevel optimizationLevel
long* outerRadius)**

This function retrieves the label region outer radius for the current media and specified label mode. This function can potentially be a time consuming function as the drive will attempt to read the media identification data around the center of the disc if it has not already done so. If there is no media present, outerRadius will be set to 0.

This function requires exclusive use of the drive.

Parameters

<i>printerHandle</i>	The LS_DiscPrinterHandle to use
<i>mode</i>	A LS_LabelMode specifying which label mode to use
<i>optimizationLevel</i>	The LS_MediaOptimizationLevel to use. See Generic Labeling
<i>outerRadius</i>	A long* to receive the label region outer radius

Returns

Global error group
Drive access error group
Media error group
LS_E_DriveInUse

**3.1.3.9. LSError LS_DiscPrinter_GetPrinterCapabilities(LS_DiscPrinterHandle printerHandle
LS_DeviceCapabilitiesEnum* capabilities)**

This function will retrieve the printer capabilities of the specified LS_DiscPrinter.

Parameters

<i>printerHandle</i>	The LS_DiscPrinterHandle to use
<i>capabilities</i>	A pointer to a LS_DeviceCapabilitiesEnum to receive the drive's capabilities

Returns

Global error group
Drive access error group

**3.1.3.10. LSError LS_DiscPrinter_GetPrinterDisplayNameSize(LS_DiscPrinterHandle printerHandle
size_t* stringSize)**

This function retrieves the size in char including the null terminator of the LS_DiscPrinter display name

Parameters

<i>printerHandle</i>	The LS_DiscPrinterHandle to use
<i>stringSize</i>	A size_t* to receive the number of char required

Returns

Global error group

3.1.3.11. LSError LS_DiscPrinter_GetPrinterDisplayName(LS_DiscPrinterHandle printerHandle char* displayName)

This function retrieves the LS_DiscPrinter display name. The returned string is not internationalized or localized; it is the exact string the drive returns. Generally, this string contains the vendor name and product name of the drive.

The string must be pre-allocated to the size returned by LS_DiscPrinter_GetPrinterDisplayNameSize.

Parameters

<i>printerHandle</i>	The LS_DiscPrinterHandle to use
<i>displayName</i>	A char* to receive the display name string

Returns

Global error group

3.1.3.12. LSError LS_DiscPrinter_GetPrinterPathSize(LS_DiscPrinterHandle printerHandle size_t* stringSize)

This function retrieves the size in char including the null terminator of the LS_DiscPrinter path.

Parameters

<i>printerHandle</i>	The LS_DiscPrinterHandle to use
<i>stringSize</i>	A size_t* to receive the number of char required

Returns

Global error group

3.1.3.13. LSError LS_DiscPrinter_GetPrinterPath(LS_DiscPrinterHandle printerHandle char* path)

This function retrieves the LS_DiscPrinter path within the operating system. The returned string is not internationalized or localized. This path can be used in conjunction with the display name to completely and persistently identify a disc printer.

The string must be pre-allocated to the size returned by LS_DiscPrinter_GetPrinterPathSize.

Parameters

<i>printerHandle</i>	The LS_DiscPrinterHandle to use
<i>path</i>	A char* to receive the path string

Returns

Global error group

3.1.3.14. LSError LS_DiscPrinter_GetPrinterProductNameSize(LS_DiscPrinterHandle printerHandle size_t* stringSize)

This function retrieves the size in char including the null terminator of the LS_DiscPrinter product name.

Parameters

<i>printerHandle</i>	The LS_DiscPrinterHandle to use
<i>stringSize</i>	A size_t* to receive the number of char required

Returns

Global error group

3.1.3.15. LSError LS_DiscPrinter_GetPrinterProductName(LS_DiscPrinterHandle printerHandle char* productName)

This function retrieves the LS_DiscPrinter product name. The returned string is not internationalized or localized; it is the exact string the drive returns.

The string must be pre-allocated to the size returned by LS_DiscPrinter_GetPrinterProductNameSize.

Parameters

<i>printerHandle</i>	The LS_DiscPrinterHandle to use
<i>path</i>	A char* to receive the product name string

Returns

Global error group

3.1.3.16. LSError LS_DiscPrinter_GetPrinterVendorNameSize(LS_DiscPrinterHandle printerHandle size_t* stringSize)

This function retrieves the size in char including the null terminator of the LS_DiscPrinter vendor name.

Parameters

<i>printerHandle</i>	The LS_DiscPrinterHandle to use
<i>stringSize</i>	A size_t* to receive the number of char required

Returns

Global error group

3.1.3.17. LSError LS_DiscPrinter_GetPrinterVendorName(LS_DiscPrinterHandle printerHandle char* vendorName)

This function retrieves the LS_DiscPrinter vendor name. The returned string is not internationalized or localized; it is the exact string the drive returns.

The string must be pre-allocated to the size returned by LS_DiscPrinter_GetPrinterVendorNameSize.

Parameters

<i>printerHandle</i>	The LS_DiscPrinterHandle to use
<i>path</i>	A char* to receive the vendor name string

Returns

Global error group

3.1.3.18. LSError LS_DiscPrinter_GetPrintResolution(LS_DiscPrinterHandle printerHandle LS_PrintQuality quality LS_MediaOptimizationLevel optimizationLevel unsigned long* resolution)

This function retrieves the optimal input resolution for a given LS_PrintQuality in pixels per meter (ppm). Note that the optimal input resolution may vary depending upon the media currently in the drive.

This function can potentially be a time consuming function as the drive will attempt to read the media identification data around the center of the disc if it has not already done so.

This function requires exclusive use of the drive.

Parameters

<i>printerHandle</i>	The LS_DiscPrinterHandle to use
<i>quality</i>	The LS_PrintQuality to use
<i>optimizationLevel</i>	The LS_MediaOptimizationLevel to use. See Generic Labeling
<i>resolution</i>	A long* to receive the optimal resolution in ppm

Returns

Global error group

Drive access error group

Media error group

LS_E_UnsupportedPrintQuality

LS_E_DriveInUse

3.1.3.19. LSError LS_DiscPrinter_LockDriveTray(LS_DiscPrinterHandle printerHandle)

This function locks the drive tray. Locking the tray prevents end users from opening the drive tray and disables the OpenDriveTray function.

This function requires exclusive use of the drive.

Parameters

printerHandle The LS_DiscPrinterHandle to use

Returns

Global error group

Drive access error group

LS_E_DriveInUse

3.1.3.20. LSError LS_DiscPrinter_OpenDriveTray(LS_DiscPrinterHandle printerHandle)

This function opens the drive tray.

This function requires exclusive use of the drive.

Parameters

printerHandle The LS_DiscPrinterHandle to use

Returns

Global error group

LS_E_DriveInUse

**3.1.3.21. LSError LS_DiscPrinter_OpenPrintSession(LS_DiscPrinterHandle printerHandle
LS_DiscPrintSessionHandle* sessionHandle)**

This function begins a print session with the disc printer and returns a LS_DiscPrintSession object.

Once a print session has been started on a disc printer, LSAPI will not permit any other print session to be started until the first session is complete.

This function requires exclusive use of the drive.

Parameters

printerHandle The LS_DiscPrinterHandle to use

sessionHandle A LS_DiscPrintSessionHandle to store the returned
LS_DiscPrintSession

Returns

Global error group

LS_E_DriveInUse

3.1.3.22. LSError LS_DiscPrinter_ReleaseExclusiveUse(LS_DiscPrinterHandle printerHandle)

This function releases and exclusive use lock. This function should be called for each call to `LS_DiscPrinter_AddExclusiveUse`.

Parameters

printerHandle The LS_DiscPrinterHandle to use

Returns

Global error group

3.1.3.23. LSError LS_DiscPrinter_UnlockDriveTray(LS_DiscPrinterHandle printerHandle)

This function unlocks the drive tray. If the tray has been locked with `LS_DiscPrinter_LockDriveTray`, this function must be called before the application exits or the tray will remain locked.

This function requires exclusive use of the drive.

Parameters

printerHandle The LS_DiscPrinterHandle to use

Returns

Global error group

Drive access error group

LS_E_DriveInUse

3.1.3.24. LSError LS_DiscPrinter_Validate(LS_DiscPrinterHandle printerHandle)

This function validates whether the disc printer may be used for printing. This method should be called immediately after acquiring a `LS_DiscPrinter` to verify that it may be used. If this drive can not be used, an appropriate error should be displayed. Depending upon the error return of this function, some of the `LS_DiscPrinter` may still be functional and may be used to create a more informative error message.

Parameters

printerHandle The LS_DiscPrinterHandle to use

Returns

Global error group

Drive access error group

3.1.4. LS_DiscPrintSession Functions

This virtual object represents a single disc printing session which is a printing object bound to a particular LightScribe disc printer. The intent is that the print session object will acquire exclusive access to the disc printer for the life of the printing session, preventing other applications from using the drive and potentially interrupting the printing process.

Functions

- `LS_Error LS_DiscPrintSession_Destroy(LS_DiscPrintSessionHandle* sessionHandle)`
- `LS_Error LS_DiscPrintSession_PrintDisc(LS_DiscPrintSessionHandle sessionHandle, LS_ImageType imageType, LS_LabelMode labelMode, LS_DrawOptions drawOptions, LS_PrintQuality quality, LS_MediaOptimizationLevel optimizationLevel, void* imageHeader, size_t headerSize, void* imageData, size_t dataSize)`
- `LS_Error LS_DiscPrintSession_PrintPreview(LS_DiscPrintSessionHandle sessionHandle, LS_ImageType imageType, LS_LabelMode labelMode, LS_DrawOptions drawOptions, LS_PrintQuality quality, LS_MediaOptimizationLevel optimizationLevel, void* imageHeader, size_t headerSize, void* imageData, size_t dataSize, char* previewFile, LS_ImageType previewImageType, LS_Size previewSize, bool ignoreMedia)`
- `LS_Error LS_DiscPrintSession_SetProgressCallback(LS_DiscPrintSessionHandle sessionHandle, LS_PrintCallbacks* callbacks)`

3.1.4.1. **LS_Error LS_DiscPrintSession_Destroy (LS_DiscPrintSessionHandle* sessionHandle)**

This function destroys and deallocates the `LS_DiscPrintSession` object pointed to by the passed in handle.

Parameters

<i>sessionHandle</i>	A <code>LS_DiscPrintSessionHandle</code> of the <code>LS_DiscPrintSession</code> to destroy
-----------------------------	---

Returns

Global error group

3.1.4.2. **LS_Error LS_DiscPrintSession_PrintDisc**, (LS_DiscPrintSessionHandle sessionHandle, LS_ImageType imageType, LS_LabelMode labelMode, LS_DrawOptions drawOptions, LS_PrintQuality quality, LS_MediaOptimizationLevel optimizationLevel, void* imageHeader, size_t headerSize, void* imageData, size_t dataSize)

This function prints an in-memory bitmap onto the drive to which this LS_DiscPrintSession is bound. This function will return at the end of the print.

The printing process may be a time consuming process. It is suggested to use the progress notification callbacks to provide updates to the application GUI and provide a method for cancellation of the print.

Warning

The supplied image data must stay in memory until the print is complete

Parameters

sessionHandle	The LS_DiscPrintSessionHandle to use
imageType	The LS_ImageType of the in-memory image
labelMode	The LS_LabelMode to use. Note that image data outside of the defined label mode will be truncated
drawOptions	The LS_DrawOptions to use.
quality	The LS_PrintQuality to use.
optimizationLevel	The LS_MediaOptimizationLevel to use. See Generic Labeling
imageHeader	The BITMAPINFOHEADER describing the image data
headerSize	The size in bytes of the supplied image header
imageData	A pointer to the actual image data
dataSize	The size in bytes of the supplied image data

Returns

Global error group
Drive access error group
Media error group
LS_E_DriveInUse
LS_E_UnsupportedSource
LS_E_InvalidSource
LS_E_Aborted
LS_E_CallbackException
LS_E_UnsupportedPrintQuality
LS_E_InvalidMediaOrientation

**3.1.4.3. LSError LS_DiscPrintSession_PrintPreview(LS_DiscPrintSessionHandle sessionHandle
LS_ImageType imageType, LS_LabelMode labelMode, LS_DrawOptions drawOptions
LS_PrintQuality quality, LS_MediaOptimizationLevel optimizationLevel
void* imageHeader, size_t headerSize, void* imageData, size_t dataSize
char* previewFile, LS_ImageType previewImageType, LS_Size previewSize, bool ignoreMedia)**

This function will create a print preview. This is not a fully accurate 1:1 mapping of bitmap pixels to label pixels. Instead, it is a transform by the print engine of the incoming bitmap image into a preview image with correct color map and potentially basic halftoning

The output will be saved to the named file. It is up to the application to delete the file after use.

Warning

The supplied image data must stay in memory until the preview is complete

Parameters

<i>sessionHandle</i>	The LS_DiscPrintSessionHandle to use
<i>imageType</i>	The LS_ImageType of the in-memory image
<i>labelMode</i>	The LS_LabelMode to use. Note that image data outside of the defined label mode will be truncated
<i>drawOptions</i>	The LS_DrawOptions to use.
<i>quality</i>	The LS_PrintQuality to use.
<i>optimizationLevel</i>	The LS_MediaOptimizationLevel to use. See Generic Labeling
<i>imageHeader</i>	The BITMAPINFOHEADER describing the image data
<i>headerSize</i>	The size in bytes of the supplied image header
<i>imageData</i>	A pointer to the actual image data
<i>dataSize</i>	The size in bytes of the supplied image data
<i>previewFile</i>	The path to the file to which the preview image should be saved
<i>previewImageType</i>	The type of image that should be saved
<i>previewSize</i>	The dimension, in pixels, of the preview image to be generated
<i>ignoreMedia</i>	Whether to ignore potential media in the drive for creation of the preview. Note that setting this flag will cause LSAPI to ignore the optimizationLevel parameter. It is recommended to always generate an optimized preview, taking into account media, whenever possible

Returns

Global error group
Drive access error group
Media error group
LS_E_DriveInUse
LS_E_PrintPreviewFileError
LS_E_UnsupportedSource
LS_E_InvalidSource
LS_E_Aborted

LS_E_CallbackException

LS_E_UnsupportedPrintQuality

LS_E_InvalidMediaOrientation

3.1.4.4. LSError LS_DiscPrintSession_SetProgressCallback (LS_DiscPrintSessionHandle sessionHandle, LS_PrintCallbacks* callbacks)

This function will set the progress callback function pointers for the print process. Only one set of function pointers may be set per print session.

Parameters

<i>sessionHandle</i>	The LS_DiscPrintSessionHandle to use
<i>callbacks</i>	A LS_PrintCallbacks* containing the callbacks to use for this print session

Returns

Global error group

3.1.5. General Functions

There are several functions which are outside of the scope of the virtual objects provided above

Functions

- LSError LS_GetUpdateShellCommandSize(size_t* shellCommandSize)
- LSError LS_GetUpdateShellCommand(char* shellCommand)
- LSError LS_LastChanceCleanup()

3.1.5.1. LSError LS_GetUpdateShellCommandSize(size_t* shellCommandSize)

This function retrieves the size in char including the null terminator of the update shell command.

Parameters

<i>shellCommandSize</i>	A size_t* to receive the number of char required
--------------------------------	--

Returns

Global error group

3.1.5.2. LSError LS_GetUpdateShellCommand(char* shellCommand)

This function retrieves the update shell command. Please refer to the LSS Updates section for more information.

The string must be pre-allocated to the size returned by LS_GetUpdateShellCommandSize.

Parameters

shellCommand A char* to receive the display name string

Returns

Global error group

3.1.5.3. LSError LS_LastChanceCleanup()

This function provides cleanup of various internal LSAPI objects and the global semaphores used for drive exclusivity. This function should be called by the application in the case of a catastrophic failure or within the application's appropriate signal handlers, such as SIGKILL.

Returns

Global error group

3.2. Data Structures

This section outlines and describes all LSAPI data structures.

3.2.1. Typedefs

All *Handle types are defined as void*.

```
typedef void* LS_DiscPrintMgrHandle;  
typedef void* LS_EnumDiscPrintersHandle;  
typedef void* LS_DiscPrinterHandle;  
typedef void* LS_DiscPrintSessionHandle;
```

3.2.2. Enumerations

3.2.2.1. LS_DeviceCapabilitiesEnum

LS_DeviceCapabilitiesEnum is a bitwise enumeration of the disc printer capabilities

Enumeration values

LS_label_monochrome Device can print monochrome labels

LS_label_color Device can print color labels

3.2.2.2. **LS_DiscShapeAndSize**

This enumeration specifies the size and shape of the media in the drive by the media information.

Enumeration values

LS_label_unknown The dimensions of this media are unknown

LS_label_12cm_circle The media in the drive is a 12cm CD or DVD

LS_label_8cm_circle The media in the drive is a 8cm CD or DVD

3.2.2.3. **LS_DrawOptions**

LS_DrawOptions is an enumeration of the supported draw options. The specified dimension is scaled to the entire label area regardless of the selected print mode. Specifying LS_label_mode_title or LS_label_mode_content will crop the image to the specified mode after scaling has been applied.

If no scaling option is set, resolution set in the image's BITMAPINFOHEADER will be used. Please note that resolution in the BITMAPINFOHEADER is specified in pixels per meter.

Enumeration values

LS_draw_default Disable scaling of the input image, use the resolution in the BITMAPINFOHEADER

LS_draw_fit_height_to_label Fit the height of the image to the label size

LS_draw_fit_width_to_label Fit the width of the image to the label size

LS_draw_fit_smallest_to_label Fit the smallest dimension of the image to the label size

3.2.2.4. **LS_Error**

LS_Error is an enumeration of all of the valid, returnable codes from any LSAPI function. Please refer to the Error Code section for details and further information.

3.2.2.5. **LS_ImageType**

LS_ImageType is an enumeration of supported image types.

Enumeration values

windows_bitmap Only uncompressed windows bitmaps are supported

3.2.2.6. LS_LabelMode

LS_LabelMode is an enumeration of all supported label modes. Refer to the Label Modes section for further information and specification of each mode.

Enumeration values

LS_label_mode_full Specify to label the entire disc

LS_label_mode_title Specify to label only within the title mode dimensions

LS_label_mode_content Specify to label only within the content mode dimensions

3.2.2.7. LS_MediaOptimizationLevel

LS_MediaOptimizationLevel is an enumeration of all supported media optimization levels for all media types. Refer to the Generic Printing section for further information.

Enumeration values

LS_media_recognized Specify to require that the optimized printing parameters for this media are found and used.

LS_media_generic Specify to require that media is present but use un-optimized generic printing parameters

3.2.2.8. LS_PrintQuality

LS_PrintQuality is an enumeration of all supported print contrast levels.

Enumeration values

LS_quality_best Specify the highest print contrast level as well as the longest print time

LS_quality_normal Specify normal print contrast level and moderate print time

LS_quality_draft Specify lightest print contrast level and fastest print time

3.2.3. Structures

3.2.3.1. LS_ColorRGB

LS_ColorRGB is a structure containing a RGB triple used to describe color values

Structure fields

unsigned char red

unsigned char green

unsigned char blue

3.2.3.2. LS_MediaInformation

LS_MediaInformation is a structure containing information about media in the drive.

Structure fields

int mediaPresent Boolean value set to 1 when media is present; set to 0 otherwise

int mediaOrientedForLabeling Boolean value set to 1 when the media is oriented label-side down and ready for labeling; set to 0 otherwise.

short manufacturer Manufacturer ID

long innerRadius The inner radius of the printable area on the media, in microns

long outerRadius The outer radius of the printable area on the media, in microns

LS_ColorRGB foreground The foreground color of the media

LS_ColorRGB background The background color of the media

LS_DiscShapeAndSize shapeAndSize The shape and size of the media

3.2.3.3. LS_PrintCallbacks

LS_PrintCallbacks is a C-style structure of function pointers. These functions will be called during the prepare image step and printing process to update printing status as well as provide a way to cancel the print.

Structure fields

bool (*AbortLabel)(void)

An application may call this function at any time during the prepare image step or printing process to cancel the print. Note that aborting after printing has commenced will yield a partially printed label.

void (*ReportPrepareProgress)(long current, long final)

This function will be called to update the progress of the prepare image step.

void (*ReportLabelProgress)(long current, long final)

This function will be called to update the progress of the printing step

void (*ReportFinished)(LS_Error status)

This function will be called at the conclusion of the printing step

bool (*ReportLabelTimeEstimate)(long time)

This function will be called at the conclusion of the prepare image step and before printing commences with an estimate of the time in seconds that will be required to print the label

3.2.3.4. LS_Size

LS_Size is a structure which defines sizes or dimensions.

Structure fields

- | | |
|---------------|---|
| <i>long x</i> | Size in the x dimension or inner radius |
| <i>long y</i> | Size in the y dimension or outer radius |

3.3. Error Codes

3.3.1. Groups of Errors

We group various error codes into certain classes so that a class of errors can be specified at once versus always specifying each and every error. This is simply for convenience in this reference manual.

In general, an application must always check the all LSError values returned by every LSAPI function.

3.3.1.1. Global Error Group

This group of errors may be returned by any LSAPI function:

- LS_E_MemoryError
- LS_E_DriveInUse
- LS_E_ReentrantCall
- LS_E_BadArgument
- LS_E_InternalError
- LS_E_DriveChanged
- LS_E_UpdateRequired
- LS_E_UnsupportedDriveCommunications
- LS_E_InstallationError
- LS_E_CommunicationsError
- LS_E_HardwareError

3.3.1.2. Drive Access Error Group

This group of errors may be returned by any LSAPI function which refers to the drive access error group:

- LS_E_UnsupportedDrive
- LS_E_CorruptResourceFile

3.3.1.3. Media Access Error Group

This group of errors may be returned by any LSAPI function which refers to the media access error group:

- `LS_E_UnableToReadMediaParams`
- `LS_E_InvalidMediaParams`
- `LS_E_MediaNotSupportedByDrive`
- `LS_E_UnsupportedMedia`
- `LS_E_UnsupportedMediaId`
- `LS_E_UnsupportedMediaNoUpdate`
- `LS_E_IncompatibleMedia`
- `LS_E_NonLightScribeMedia`
- `LS_E_NoMedia`

3.3.1.4. LSS Update Error Group

This group of errors may potentially be resolved by a LSS update:

- `LS_E_UnsupportedDrive`
- `LS_E_UnsupportedMedia`
- `LS_E_CorruptResourceFile`
- `LS_E_UpdateRequired`
- `LS_E_UnsupportedMediaId`
- `LS_E_UnsupportedDriveCommunications`

3.3.2. End User Messaging

This section contains recommended end user messaging for LSAPI errors. The application should provide the best possible descriptions and actionable information to the user. As such, it is strongly recommended that the application use this messaging as a minimum. The application can always provide more detailed information beyond this to provide an excellent labeling experience.

3.3.2.1. Terminal Errors

Errors from this group are critical error conditions such as a missing LightScribe library. These are all terminal error conditions where the LightScribe process can not continue

`LS_E_InstallationError`

“A critical internal LightScribe System Software (LSS) error has occurred. Please reinstall or update the LSS and try your label again.

Would you like to look for an update to the LSS now? (An Internet connection is required)”

3.3.2.2. Communication Errors

Errors from this group are communication errors such as failure to communicate with the LightScribe drive.

LS_E_CommunicationsError

“A communication error with the LightScribe drive has occurred. Please restart your labeling application and try your label again.

If this error occurs again, please restart your PC and try your label again.”

3.3.2.3. General Fault Errors

Errors from this group are general fault errors and usually point to a larger problem than the LightScribe system itself, such as memory corruption. It is suggested that the application use the same messaging that it would use in the case where this class of error is generated from the application itself.

LS_E_MemoryError

LS_E_PrintPreviewFileError

LS_E_InternalError

LS_E_HardwareError

“An internal LightScribe System Software (LSS) error has occurred. Please restart your labeling application and try your label again.

If this error occurs again, please restart your PC and try your label again.”

3.3.2.4. API Errors

Errors from this group are generated when the application calls LSAPI in an incorrect manner or uses an invalid parameter.

LS_E_UnsupportedSource

LS_E_InvalidSource

LS_E_CallbackException

LS_E_UnsupportedPreview

LS_E_ReentrantCall

LS_E_BadArgument

LS_E_UnsupportedPrintQuality

“Your labeling application has caused an internal LightScribe System Software (LSS) error. Please restart your labeling application and try your label again.

If this error occurs again, please contact your labeling application support.”

3.3.2.5. Generic Labeling

Errors in this group are errors which allow for generic printing.

LS_E_UnsupportedMedia

LS_E_UnsupportedMediaNoUpdate

“The LightScribe System Software (LSS) does not contain information about the LightScribe disc in the drive. For optimal LightScribe labeling, an update to the LSS is recommended before labeling to this disc. Or you may choose to use generic labeling.

Would you like to look for a LSS update now? (An Internet connection is required)”

Note that LS_E_UnsupportedMediaNoUpdate is included in both this section and the Unable to Label section. Generic labeling may still be possible with this error code so it belongs to this group, however, no LSS update will fix this error, only generic labeling is possible.

3.3.2.6. Update Conditions

This is the LSS Update Error group of errors.

LS_E_UnsupportedDrive

LS_E_CorruptResourceFile

LS_E_UpdateRequired

LS_E_UnsupportedMediaId

LS_E_UnsupportedDriveCommunications

“An update to the LightScribe System Software (LSS) is required to use your selected LightScribe drive.

Would you like to look for a LSS update now? (An Internet connection is required)”

3.3.2.7. Unable to Label

Errors from this group are generated in situations where the LightScribe system can not label the media in the drive and an update will not change the error condition. These errors are generally returned when the LightScribe media in the drive is explicitly incompatible with the drive itself.

LS_E_InvalidMediaParams

LS_E_MediaNotSupportedByDrive

LS_E_UnsupportedMediaNoUpdate

LS_E_IncompatibleMedia

“This disc is incompatible with the selected LightScribe drive and can not be labeled.”

3.3.2.8. General Error Conditions

Errors from this group cover a wide range of conditions which the user might experience such as aborting a label or the application was unable to get an exclusive lock on the LightScribe drive

LS_E_InvalidMediaOrientation

“To label your LightScribe disc, please insert the disc with the label side down.”

LS_E_NoMedia

“A LightScribe disc was not detected in the drive. Please insert a LightScribe disc with the label side down.”

LS_E_Aborted

“LightScribe labeling has been canceled”

LS_E_DriveInUse

LS_E_DriveTrayLocked

“The LightScribe drive is currently in use”

LS_E_UnableToReadMediaParams

LS_E_NonLightScribeMedia

“The disc in the LightScribe drive is not recognized

Please ensure that the disc in the drive is a LightScribe disc and is inserted into the drive label side down.

If your LightScribe disc is still not recognized, dirt or smudges may be on the disc. Please wipe both disc surfaces with a soft cloth and try again.”

LS_E_DriveChanged

“The LightScribe drive has changed. This may have been caused by a hardware change or a Plug and Play event.

Please re-select the LightScribe drive that you wish to use.”

3.3.3. Error Code Reference

Value	Name	Description
0x1	LS_OK	No Error
0x205	LS_E_InvalidMediaOrientation	The media is not oriented such that labeling can proceed. This typically means that the disc needs to be flipped so the label side is exposed to the labeling laser.
0x208	LS_E_UnableToReadMediaParams	The encoded media information parameters where not able to be read from the LightScribe disc. The cause could be a dirty disc or a disc with the wrong side up.

Value	Name	Description
0x209	LS_E_InvalidMediaParams	A media identification parameter is invalid. A LSS update will not fix this problem.
0x211	LS_E_UnsupportedSource	Unsupported source image format. Only those explicitly supported LS_ImageType are allowed.
0x212	LS_E_InvalidSource	Invalid source image format. This can include the resolution being unset or zero.
0x219	LS_E_Aborted	The printing process has been aborted because the AbortLabel callback has been called
0x21A	LS_E_MemoryError	Memory Error
0x21B	LS_E_CallbackException	The progress callback threw an exception, which triggered an emergency abort of the labeling process.
0x21D	LS_E_UnsupportedPreview	Unsupported preview image format. Only those explicitly supported LS_ImageType are allowed.
0x21E	LS_E_DriveInUse	An attempt to open a drive has been made with another drive already opened or this drive already in use
0x220	LS_E_MediaNotSupportedByDrive	The drive does not support this media type. A LSS update will not fix this problem.
0x223	LS_E_ReentrantCall	The application attempted a reentrant call to a busy object.
0x226	LS_E_BadArgument	One of the supplied arguments is erroneous, possibly out of range.
0x229	LS_E_PrintPreviewFileError	The print preview command failed as the output file could not be created.
0x22A	LS_E_DriveTrayLocked	The drive tray operation was blocked because the currently selected drive's tray is locked.
0x22B	LS_E_UnsupportedDrive	The print engine was unable to find the resources associated with this drive type. A LSS update may fix this.
0x22C	LS_E_UnsupportedMedia	The print engine was unable to find the optimized labeling parameters associated with this media type. A LSS update may fix this.
0x22D	LS_E_UnsupportedPrintQuality	The specified print quality is not supported by this drive and media type.
0x22E	LS_E_CorruptResourceFile	The labeling parameters associated with this drive are corrupt and this drive can not be used. A LSS update may fix this.

Value	Name	Description
0x22F	LS_E_InternalError	An internal error has occurred.
0x233	LS_E_DriveChanged	A change in the drive description indicates that the selected drive is no longer valid. This may have been caused by a Plug and Play event.
0x234	LS_E_UpdateRequired	A version mismatch has occurred between the print engine and the required labeling parameters. A LSS update may fix this.
0x237	LS_E_UnsupportedMediaId	A media parameter value is unsupported. A LSS update may fix this.
0x239	LS_E_UnsupportedDriveCommunications	This version of the LSS can not communicate with the selected drive. A LSS update may fix this.
0x23b	LS_E_UnsupportedMediaNoUpdate	The optimized labeling parameters for this media type were not found and no further updates for this drive are supported. A LSS update will not fix this.
0x23c	LS_E_IncompatibleMedia	The print engine indicates that this media is not compatible with the selected drive. A LSS update will not fix this.
0x23d	LS_E_NonLightScribeMedia	The drive indicates that non-LightScribe media is present in the drive.
0x23E	LS_E_InstallationError	Component(s) are missing or corrupt with this installation. Reinstallation or a LSS update may fix this.
0x1012	LS_E_NoMedia	The drive indicates that no media is present.
0x1080	LS_E_CommunicationsError	The drive has not communicated as expected or a request to the drive has failed.
0x1081	LS_E_HardwareError	A hardware error has occurred with the drive.

3.4. Label Modes Specification

3.4.1. Title Mode Label

The Title Mode label is defined by a circular band extending from 6.28 cm to 7.3 cm diameter, centered on the disc. The outer diameter (OD) of the band is located about 0.5 cm inward from the center of the radii of a 12 cm disc. This is located near the OD of an 8 cm disc. The 8 cm disc label may extend closer to the disc edge than the 12 cm disc label.

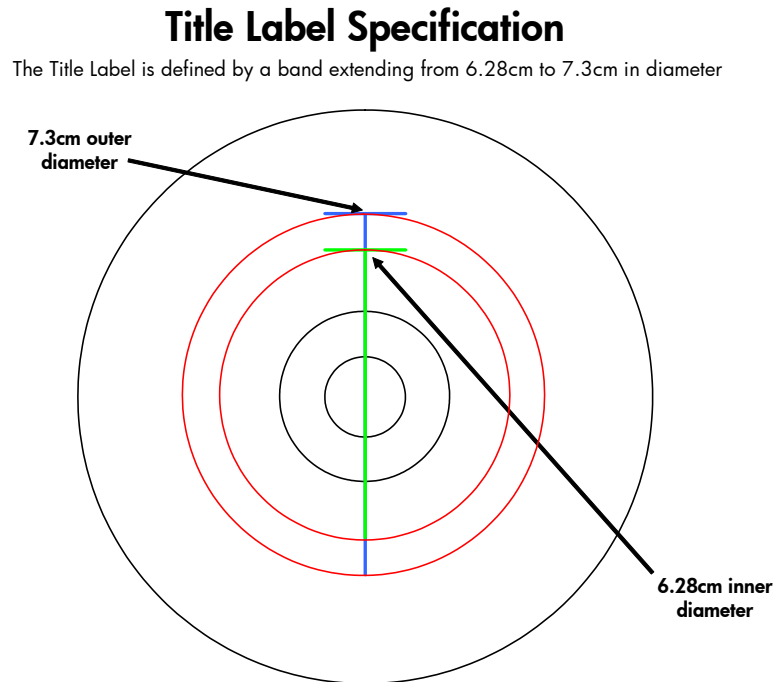


Figure 3.1.

3.4.2. Content Mode Label

The Content Mode label is defined by a circular band extending from 5.01 cm to 7.3 cm diameter, centered on the disc. The OD of the band is located at the same location as defined for the "Title Label" band OD. The band almost covers the entire label area for 8 cm discs. The outer diameter (OD) of the band is located about 0.5 cm inward from the center of the radii of a 12 cm disc. This is located near the label OD of an 8 cm disc. The 8 cm disc label may extend closer to the disc edge than the 12 cm disc label.

Content Label Specification

The Content Label is defined by a band extending from 5.01cm to 7.3cm in diameter

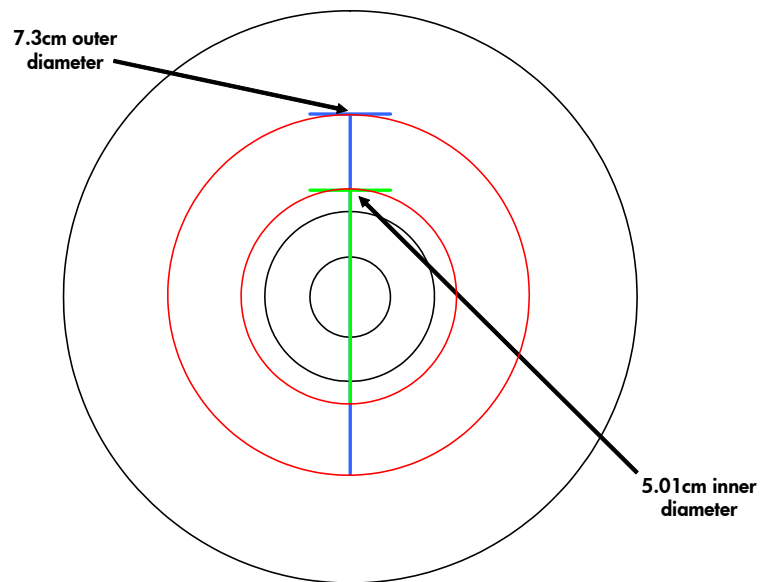


Figure 3.2.

3.4.3. Full Mode Label

The Full Mode label is defined as the labeling area from 4.76 cm to 11.74 cm diameter, centered on the disc.

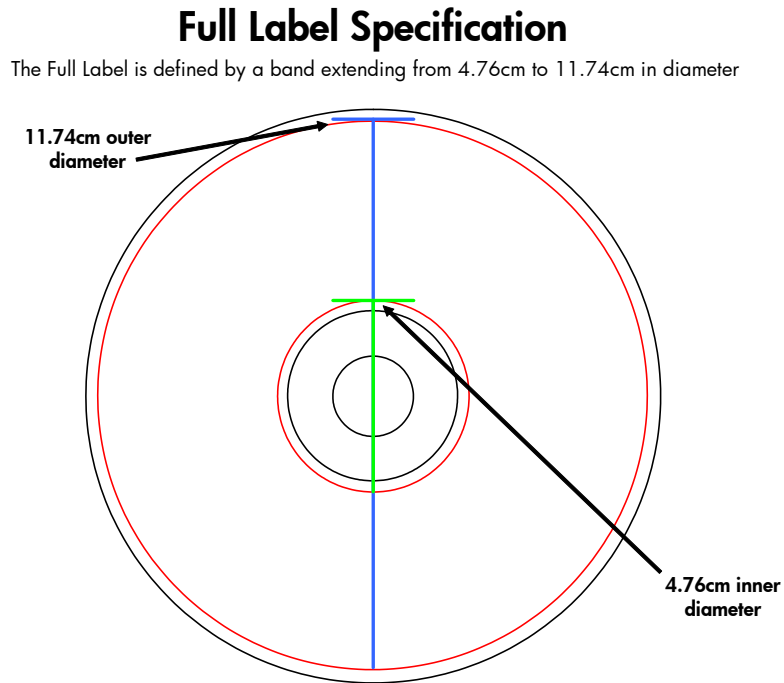


Figure 3.3.