grass valley

MEDIAEDGE LEB Series

Development Kit

Version 1.00

F2431012163
December 2010

www.grassvalley.com

■ご購入製品を使用される際の注意事項

ここではご購入製品を使用されるときにご注意いただきたい事項について説明しています。ご使用方法や、本注意事項の内容についてご不明な点、疑問点などございましたら、トムソン・カノープス株式会社カスタマーサポート部までお問い合わせください。

トムソン・カノープス株式会社
〒651-2241
神戸市西区室谷 1-2-2
カスタマーサポート部 TEL.078-992-8374
（10:00～12:00、13:00～17:00）
※土、日、祝日および当社指定休日を除く

●ご利用目的に関する注意事項

医療機器や人命に関るシステムでは、絶対にご利用にならないでください。製品の性質上、これらのシステムへの導入は適しません。

●他社製品との併用に関する注意事項

他社製品と併用されるとご購入製品は正常に動作しないことがあり、そのためにシステムが本来の目的を達成することができないこともあります。あらかじめ、製品単体の環境でご購入製品が正常に動作することをご確認ください。また、他社製品との併用でご購入製品が正常に動作しないのであれば、その他社製品とご購入製品との併用はお止めください。

■健康上のご注意

ごくまれに、コンピュータのモニタおよびテレビ画面に表示される強い光の刺激や点滅によって、一時的にてんかん・意識の喪失などが引き起こされる場合があります。こうした経験をこれまでにされたことがない方でも、それが起こる体質をもっていることも考えられます。こうした経験をお持ちの方や、経験をお持ちの方の血縁にあたられる方は、本製品を使用される前に必ず医師と相談してください。

■警告

● 電源コードを傷つけない

電源コードを傷つけると、火災や感電の原因となります。コードの上に重いものをのせたり、熱器具に近づけたりしないでください。また、コードを折り曲げたり、加工しないでください。AC アダプタを抜くときは、プラグ部分を持ってください。コードが傷んだら、お買い上げの販売店まで交換をご依頼ください。

● キャビネットを開けない

キャビネットを開けたり改造したりすると、火災や感電の原因となります。内部の点検、修理はお買い上げの販売店までご依頼ください。

● ほこりや湿気の多い場所では使用しない

ショートや発熱が起こり、火災や感電の原因となります。

● 内部に水や異物を入れない

水や異物が入ると、火災や感電の原因となります。万一、水や異物が入った場合は、電源コードをコンセントから抜いて、お買い上げの販売店までご連絡ください。

● 雷が鳴り出したら使わない

本体やプラグには触れないでください。感電の原因となります。

● ぬれた手で AC アダプタを触らない

ぬれた手で AC アダプタを抜き差ししないでください。感電の原因となります。

● 直射日光の当たる場所に置かない

日光の当たる場所や熱器具のそばに置かないでください。火災や製品の故障の原因となります。

● 煙が出た状態で使用しない

煙が出る、異臭がするなどの異常状態で使用しないでください。火災や製品の故障の原因となります。異常が発生したら、本体の電源を切り、電源コードを抜いて、煙が消えたのを確認してから、お買い上げの販売店までご連絡ください。

● 製品が破損した状態で使用しない

本製品を落としたり、カバーを破損した状態のまま使用しないでください。火災や製品の故障の原因となります。製品が破損した場合は、本体の電源を切り、電源コードをコンセントから抜いて、お買い上げの販売店までご連絡ください。

● 不安定な場所に置かない

不安定な台の上や傾いたところに置かないでください。落下するおそれがあり、けがをしたり、製品の故障の原因となります。

● お手入れの際は電源を切る

接続するときやお手入れの際は、電源プラグを抜いてください。感電や製品の故障の原因となります。お手入れの際は、シンナーなどの揮発性の溶剤を使用しないでください。

● 付属の AC アダプタ
会社名 : UNIFIVE TECHNOLOGY(ShenZhen) Co., Ltd.
型 : UIA336-12

● コード類は正しく配置する
電源コードや AV ケーブルは整理して配置してください。足にひっかけると、けがや製品の故障の原因となります。

● 本体を布などで覆わない
風通しの悪い場所や布などで覆った状態で使用しないでください。通風孔がふさがれると内部に熱がこもって、火災や製品の故障の原因となります。

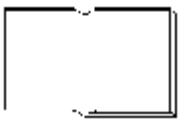
● 長時間使わないときは AC アダプタを外す
使用しないときは、安全のため AC アダプタをコンセントから外してください。

■個人情報の取扱いについて
当社では、原則として①ご記入いただいたお客様の個人情報は下記目的以外では使用せず、②下記以外の目的で使用する場合は事前に当該サービスにてお知らせいたします。
当社ではご記入いただいた情報を適切に管理し、特段の事情が無い限りお客様の承諾無く第三者に開示・提供することはございません。

1. ご利用の当社製品のサポートの実施
2. 当社製品の使用状況調査、製品改良、製品開発、サービス向上を目的としたアンケートの実施
   *調査結果を当社のビジネスパートナーに参考資料として提供することがありますが、匿名性を確保した状態で提供いたします。
3. 銀行口座やクレジットカードの正当性、有効性の確認
4. ソフトウェアのバージョンアップや新製品の案内等の情報提供
5. 懸賞企画等で当選された方やお客様への賞品の発送
   *お客様の個人情報の取扱いに関するご意見、お問い合わせは
   http://www.thomson-canopus.jp/info/までご連絡ください。

ご注意

(1)本製品の一部または全部を無断で複製することを禁止します。
(2)本製品の内容や仕様は将来予告無しに変更することがあります。
(3)本製品は内容について万全を期して作成いたしましたが、万一ご不審な点や誤り、記載漏れなどお気付きの事がございましたら、当社までご連絡ください。
(4)運用した結果については、(3)項にかかわらず責任を負いかねますのでご了承ください。
(5)ご使用上の過失の有無を問わず、本製品の運用において発生した逸失利益を含む特別、付随的、または派生的損害に対するいかなる請求があったとしても、当社はその責任を負わないものとします。
(6)本製品付属のソフトウェア、マニュアル、その他添付物を含めたすべての関連品に関して、解析、リバースエンジニアリング、デコンパイル、ディスアッセンブリを禁じます。
(7)MEDIAEDGE はトムソン・カノープス株式会社の登録商標です。
(8)Microsoft、Windows は米国マイクロソフト・コーポレーションの登録商標です。またその他の商品名やそれに類するものは各社の商標または登録商標です。



表記について

■本書では Microsoft® Windows® operating system を『Windows』と表記します。
■本書では MEDIAEDGE LEB Pro/Express を『LEB Pro/Express』と表記します。
■本書は Windows 上でプログラムを作成することができる方を対象に書かれています。
■本書での説明と実際の運用方法とで相違点がある場合には、実際の運用方法を優先するものとします。プログラム上の基本的な事項、プログラムの方法などについては記載していません。
■本書に記載されていない情報が記載される場合がありますので、ディスクに添付のテキストファイルも必ずお読みください。

目次

CHAPTER 1　はじめに

1. 開発キットに含まれるソースコードの取り扱いについて

本開発キットに含まれるサンプルプログラムはドキュメントを補うための資料となっています。サンプルプログラムのソースコード（以下、「ソースコード」といいます）自体を改変したり、その一部をお客様のアプリケーションに組み込んでご利用いただけます。

トムソン・カノープス株式会社はソースコードの使用と変更に関して完全に自由な権利をお客様に許諾いたします。

また、最終的に作成されたアプリケーションの運用結果および目的への適合性につき、トムソン・カノープス株式会社では一切の責任を負いかねますので予めご了承ください。

CHAPTER 2　　LEB API 概要

LEB API は **LEB Pro/Express** の検出、設定値の取得・変更、ストリームの受信、再生、録画、静止画保存を行う機能を **C** 言語の **API** として提供する **DLL** です。

LEB API は **LEBEnum**、**LEBConf**、**LEBCoder** の 3種類に大別されます。

LEB SDK は以下の **OS** で動作確認を行っています。

- **Microsoft Windows XP SP3**（日本語版）
- **Microsoft Windows 7**（日本語版）
- **Microsoft Windows Server 2008 SP2**（日本語版）
- **Microsoft Windows Server 2008 R2**（日本語版）

※ただし、**Microsoft Windows Server** 環境では **MEDIAEDGE-SWT4** を利用した再生・録画を行うことはできません。

CHAPTER 3　インストール・アンインストール

1. インストール

製品付属の **CD-ROM** を **PC** のディスクドライブにセットして開き、[**LEBSDK**]フォルダ→[**LEBSDKSetup.msi**]をダブルクリックし、画面にしたがってインストールしてください。

2. アンインストール

[スタート]メニュー→[すべてのプログラム]→**[MEDIAEDGE]**→**[MEDIAEDGE LEB]**→**[SDK]**→[**uninstall**]を選択し、画面にしたがってアンインストールしてください。

CHAPTER 4　チュートリアル

Microsoft Visutal Studio 2008 (VC9) と Microsoft Foundation Class (MFC)を用いて LEB SDK を使用するプログラムの例を示します。

1. LEB API の使用準備

LEB API を使用するには "LEBAPIDef.h" をインクルードし、LEBAPI.lib をリンカの入力として指定します。

```
#include <windows.h>
#include "../include/LEBAPIDef.h"
#pragma comment(lib, "../lib/LEBAPI.lib")
```

Visual C++ を用いる場合は "LEBAPIWrapper.h" をインクルードすることでより簡単に LEB API を使用できます。

```
#include "../include/LEBAPIWrapper.h"
```

これらのファイルは LEB SDK をインストールしたフォルダに含まれています。

2. LEBEnum の使用

LEB Pro/Express の探索の開始

LEBEnum を用いて LEB Pro/Express の探索を開始します。

```
// 変数
HANDLE handle;
LEBRET ret;

// コールバック関数
VOID CALLBACK EnumEventCallback(
	LONG_PTR param,
	LEBENUM_EVENT enumEvent,
	LPCWSTR uniqueName,
	LPCWSTR address)
{
	// コールバック関数はワーカースレッドから呼ばれるため、処理内容によっては
	// 排他制御を行う必要があります。
	switch (enumEvent)
	{
	case LEBENUM_ADDED:
		// LEB Pro/Express が見つかったときの処理
		// uniqueName にデバイスの固有名称
		// address にデバイスの IP アドレスが格納されています
		break;
	case LEBENUM_REMOVED:
		// LEB Pro/Express が正常に停止されたときの処理
		// uniqueName にデバイスの固有名称が格納されています
		break;
	case LEBENUM_COMPLETE:
		// 現時点でネットワーク上に存在している LEB Pro/Express の探索が
		// 完了したときの処理
		// これ以降は新たに LEB Pro/Express がネットワーク上で起動 /
```

```
                // 正常終了した場合にイベントが発生します
                break;
        }
}

ret = LEBEnum_CreateHandle(&handle);
if (ret != LEBRET_SUCCESS)
{
        // エラー処理
        return;
}
ret = LEBEnum_Start(handle, EnumEventCallback, NULL);
if (ret != LEBRET_SUCCESS)
{
        // エラー処理
        return;
}
```

LEB Pro/Express の探索の終了

LEB Pro/Express を探索する処理を終了し、リソースを開放します。

```
// 変数
HANDLE handle = <LEBEnum_CreateHandle で取得したハンドル>;

LEBEnum_Stop(handle);
LEBEnum_CloseHandle(handle);
```

3. LEBConf の使用

LEB Pro/Express の設定値の取得（同期）

LEBConf を用いて LEB Pro/Express から設定値を取得します。

```
// 変数
LONG            timeout      = <タイムアウト時間>;      // ミリ秒
LPCWSTR         target       = <LEB Pro/Express の IP アドレス>;
LPCWSTR         user         = <LEB Pro/Express のユーザー名>;
LPCWSTR         password     = <LEB Pro/Express のパスワード>;
LPCWSTR         name         = <取得するパラメータ名>;  //「設定値」の項目を参照

HANDLE          handle;
LEBRET          ret;

const LONG LEN               = 256;
WCHAR           value[LEN]   = { L'¥0' };
LONG            length       = LEN;

ret = LEBConf_CreateHandle(&handle);
if (ret != LEBRET_SUCCESS)
{
       // エラー処理
       return;
}
ret = LEBConf_Prepare(handle, timeout, target, user, password);
if (ret != LEBRET_SUCCESS)
{
       // エラー処理
       return;
}
ret = LEBConf_Get(handle, name, value, &length);
if (ret != LEBRET_SUCCESS)
```

```
{
        // エラー処理
        return;
}
```

LEB Pro/Express の設定値の取得（非同期）

LEBConf を用いて LEB Pro/Express から設定値を取得します。

```
// 変数
LONG            timeout      = <タイムアウト時間>;      // ミリ秒
LPCWSTR         target       = <LEB Pro/Express の IP アドレス>;
LPCWSTR         user         = <LEB Pro/Express のユーザー名>;
LPCWSTR         password     = <LEB Pro/Express のパスワード>;
LPCWSTR         name         = <取得するパラメータ名>;  //「設定値」の項目を参照

HANDLE          handle;
LEBRET          ret;
LEBRET          retPrepare;
HANDLE          completed;

const LONG LEN               = 256;
WCHAR           value[LEN]   = { L'¥0' };
LONG            length       = LEN;

// コールバック関数
VOID CALLBACK PrepareCompleted(LONG_PTR param, LEBRET ret)
{
        retPrepare = ret;
        SetEvent(completed);
}

ret = LEBConf_CreateHandle(&handle);
```

```
if (ret != LEBRET_SUCCESS)
{
	// エラー処理
	return;
}
completed = CreateEvent(NULL, TRUE, FALSE, NULL);
ret = LEBConf_PrepareAsync(handle, timeout,
	target, user, password, PrepareCompleted, NULL);
if (ret != LEBRET_PENDING)
{
	// エラー処理
	return;
}
WaitForSingleObject(completed, INFINITE);
CloseHandle(completed);
if (retPrepare != LEBRET_SUCCESS)
{
	// エラー処理
	return;
}
ret = LEBConf_Get(handle, name, value, &length);
if (ret != LEBRET_SUCCESS)
{
	// エラー処理
}
```

LEB Pro/Express の設定値の変更（同期）

LEBConf を用いて LEB Pro/Express の設定値を変更します。
「LEB Pro/Express の設定値の取得（同期）」の項目または「LEB Pro/Express の設定値の取得（非同期）」の項目を参考に LEBConf_Prepare(Async) までを完了させておいてください。

```
// 変数
LONG        timeout    = <タイムアウト時間>;         // ミリ秒
LPCWSTR     name       = <設定するパラメータ名>;     //「設定値」の項目を参照
LPCWSTR     value      = <設定値>;

HANDLE      handle     = <LEBConf_CreateHandle で取得したハンドル>;
LEBRET      ret;

ret = LEBConf_Set(handle, name, value);
if (ret != LEBRET_SUCCESS)
{
        // エラー処理
        return;
}
ret = LEBConf_Apply(handle, timeout);
if (ret != LEBRET_SUCCESS)
{
        // エラー処理
        return;
}
```

LEB Pro/Express の設定値の変更（非同期）

LEBConf を用いて LEB Pro/Express の設定値を変更します。
「LEB Pro/Express の設定値の取得（同期）」の項目または「LEB Pro/Express の設定値の取得（非同期）」の項目を参考に LEBConf_Prepare(Async) までを完了させておいてください。

```
// 変数
LONG        timeout       = <タイムアウト時間>;       // ミリ秒
LPCWSTR     name          = <設定するパラメータ名>;   //「設定値」の項目を参照
LPCWSTR     value         = <設定値>;

HANDLE      handle        = <LEBConf_CreateHandle で取得したハンドル>;
```

```
LEBRET     ret;
LEBRET     retApply;
HANDLE     completed;

// コールバック関数
VOID CALLBACK ApplyCompleted(LONG_PTR param, LEBRET ret)
{
       retApply = ret;
       SetEvent(completed);
}

ret = LEBConf_Set(handle, name, value);
if (ret != LEBRET_SUCCESS)
{
       // エラー処理
       return;
}
completed = CreateEvent(NULL, TRUE, FALSE, NULL);
ret = LEBConf_ApplyAsync(handle, timeout, ApplyCompleted, NULL);
if (ret != LEBRET_PENDING)
{
       // エラー処理
       return;
}
WaitForSingleObject(completed, INFINITE);
CloseHandle(completed);
if (retApply != LEBRET_SUCCESS)
{
       // エラー処理
       return;
}
```

LEB Pro/Express の設定値の取得・変更の終了

LEB Pro/Express の設定値の取得・変更処理を終了し、リソースを開放します。

```
// 変数
HANDLE     handle   = <LEBConf_CreateHandle で取得したハンドル>;

LEBConf_CloseHandle(handle);
```

4. LEBCoder の使用

LEB Pro/Express に接続（プレビューあり）

LEBCoder を用いて LEB Pro/Express に接続し、ストリームの受信を開始します。
受信処理、プレビュー表示は MEDIAEDGE-SWT4 により行います。MEDIAEDGE-SWT4 がインストールされていない場合は LEBCoder_Prepare は失敗します。
また、MEDIAEDGE-SWT4 が認証されない場合は LEBCoder_Start は失敗します。

```
// 変数
LPCWSTR    target    = <LEB Pro/Express の IP アドレス>;
RECT       rect      = <プレビューウィンドウのサイズ>;
HWND       parent    = <プレビューウィンドウの親ウィンドウのハンドル>;
LONG       style     = WS_CHILD | WS_VISIBLE;

HANDLE     handle;
LEBRET     ret;

// コールバック関数
VOID CALLBACK CoderEventCallback(
       LONG_PTR lpParam,
       LEBCODER_EVENT nEvent,
       LONG nCode)
{
       // コールバック関数はワーカースレッドから呼ばれるため、処理内容によっては
       // 排他制御を行う必要があります。
       // nEvent、nCode については「LEBCODER_EVENT」の項目を参照
       switch (nEvent)
       {
       case LEBCODER_START:                    break;
       case LEBCODER_STOP:                     break;
       case LEBCODER_END_OF_CONTENT:           break;
       case LEBCODER_DONGLE_ERROR:             break;
       case LEBCODER_DECODE_ERROR:             break;
```

```
        case LEBCODER_STREAM_CONTROL_ERROR:     break;
        case LEBCODER_RECORDING_START:          break;
        case LEBCODER_RECORDING_STOP:           break;
        case LEBCODER_RECORDING_ERROR:          break;
        default:                                break;
        }
}

ret = LEBCoder_CreateHandle(&handle);
if (ret != LEBRET_SUCCESS)
{
        // エラー処理
        return;
}
ret = LEBCoder_Prepare(handle,
        style, &rect, parent, CoderEventCallback, NULL);
if (ret != LEBRET_SUCCESS)
{
        // エラー処理
        return;
}
ret = LEBCoder_Start(handle, target);
if (ret != LEBRET_SUCCESS)
{
        // エラー処理
        return;
}
```

LEB Pro/Express に接続（プレビューなし）

LEBCoder を用いて LEB Pro/Express に接続し、ストリームの受信を開始します。
受信処理は LEB SDK 内部の機能により行います。
※この場合、LEB SDK を使用しているプロセスをファイアウォール例外に登録しておく必要があります。

```
// 変数
LPCWSTR    target    = <LEB Pro/Express の IP アドレス>;

HANDLE     handle;
LEBRET     ret;

// コールバック関数
VOID CALLBACK CoderEventCallback(
       LONG_PTR lpParam,
       LEBCODER_EVENT nEvent,
       LONG nCode)
{
       // コールバック関数はワーカースレッドから呼ばれるため、処理内容によっては
       // 排他制御を行う必要があります。
       // nEvent、nCode については「LEBCODER_EVENT」の項目を参照
       switch (nEvent)
       {
       case LEBCODER_START:                    break;
       case LEBCODER_STOP:                     break;
       case LEBCODER_END_OF_CONTENT:           break;
       case LEBCODER_DONGLE_ERROR:             break;
       case LEBCODER_DECODE_ERROR:             break;
       case LEBCODER_STREAM_CONTROL_ERROR:     break;
       case LEBCODER_RECORDING_START:          break;
       case LEBCODER_RECORDING_STOP:           break;
       case LEBCODER_RECORDING_ERROR:          break;
       default:                                break;
```

```
        }
}

ret = LEBCoder_CreateHandle(&handle);
if (ret != LEBRET_SUCCESS)
{
        // エラー処理
        return;
}
ret = LEBCoder_PrepareNoWindow(handle, CoderEventCallback, NULL);
if (ret != LEBRET_SUCCESS)
{
        // エラー処理
        return;
}
ret = LEBCoder_Start(handle, target);
if (ret != LEBRET_SUCCESS)
{
        // エラー処理
}
```

プレビューウィンドウのサイズ変更

親ウィンドウのサイズが変更された場合などに LEBCoder_Prepare で作成したプレビューウィンドウのサイズを変更します。「LEB Pro/Express に接続（プレビューあり）」の項目を参考に LEB Pro/Express に LEBCoder_Prepare を完了しておきます。

```
// 変数
RECT        rect     = <プレビューウィンドウのサイズ>;
HANDLE      handle   = <LEBCoder_CreateHandle で作成したハンドル>;
LEBRET      ret;

ret = LEBCoder_Resize(handle, &rect);
if (ret != LEBRET_SUCCESS)
```

```
{
        // エラー処理
        return;
}
```

ファイルへの録画

LEBCoder を用いて LEB Pro/Express から受信しているストリームをファイルに録画します。「LEB Pro/Express に接続（プレビューあり）」の項目または「LEB Pro/Express に接続（プレビューなし）」の項目を参考にストリームの受信を開始しておいてください。MEDIAEDGE-SWT4 が対応していないバージョンである場合は失敗します。（MEDIAEDGE-SWT4 Version 1.20 以降対応）

```
// 変数
LPCWSTR     file      = <録画ファイルのフルパス>;
HANDLE      handle    = <LEBCoder_CreateHandle で作成したハンドル>;
LEBRET      ret;

ret = LEBCoder_StartRecordFile(handle, file);
if (ret != LEBRET_SUCCESS)
{
        // エラー処理
        return;
}
```

メモリ転送

LEBCoder を用いて LEB Pro/Express から受信しているストリームをメモリ転送します。「LEB Pro/Express に接続（プレビューあり）」の項目または「LEB Pro/Express に接続（プレビューなし）」の項目を参考にストリームの受信を開始しておいてください。MEDIAEDGE-SWT4 が対応していないバージョンである場合は失敗します。（MEDIAEDGE-SWT4 Version 1.20 以降対応）

```
// 変数
LONG        size     = <メモリ転送のサイズ>;
HANDLE      handle   = <LEBCoder_CreateHandle で作成したハンドル>;
LEBRET      ret;

// コールバック関数
VOID CALLBACK CoderDataCallback(
        LONG_PTR lpParam,
        LPBYTE lpData,
        LONG nSize)
{
        // コールバック関数はワーカースレッドから呼ばれるため、処理内容によっては
        // 排他制御を行う必要があります。
        // ...
}

ret  =  LEBCoder_StartRecordMemory(handle,  size,  CoderDataCallback,
NULL);
if (ret != LEBRET_SUCCESS)
{
        // エラー処理
        return;
}
```

静止画キャプチャ

LEBCoder を用いて LEB Pro/Express から受信しているストリームを静止画としてファイルに保存します。「LEB Pro/Express に接続（プレビューあり）」の項目を参考にプレビューありの状態でストリームの受信を開始しておきます。プレビューなしで実行されている場合、または MEDIAEDGE-SWT4 が静止画キャプチャに対応していないバージョンである場合は失敗します。（MEDIAEDGE-SWT4 Version 1.20 以降対応）

```
// 変数
LPCWSTR    file     = <静止画ファイルのフルパス>;
HANDLE     handle   = <LEBCoder_CreateHandle で作成したハンドル>;
LEBRET     ret;

ret =  LEBCoder_Capture(handle, file);
if (ret != LEBRET_SUCCESS)
{
       // エラー処理
       return;
}
```

ファイルへの録画・メモリ転送の終了

ファイルへの録画またはメモリ転送を終了します。

```
// 変数
HANDLE     handle   = <LEBCoder_CreateHandle で作成したハンドル>;

LEBCoder_StopRecord(handle);
```

LEB Pro/Express への接続の終了

LEB Pro/Express からのストリームの受信を終了し、リソースを開放します。

```
// 変数
HANDLE     handle   = <LEBCoder_CreateHandle で作成したハンドル>;

LEBCoder_Stop(handle);
LEBCoder_Destroy(handle);
LEBCoder_CloseHandle(handle);
```

CHAPTER 5　　API リファレンス

1. LEBEnum

LEBEnum は LEB Pro/Express を UPnP (Universal Plug and Play) を用いて発見します。

この機能は **Windows SSDP Discovery Service (SSDPSRV)** に依存しているため、**SSDP Discovery Service** を開始しておく必要があります。また、対応するファイアウォールの規則を設定しておく必要があります。

SSDP Discovery Service を開始するには［ファイル名を指定して実行］から［**services.msc**］を実行し［サービス］の管理コンソールから［**SSDP Discovery**］を開始します。

ファイアウォールの規則を設定するには［ファイル名を指定して実行］から［**WF.msc**］を実行し、［セキュリティが強化された **Windows** ファイアウォール］管理コンソールから［ローカルコンピューターのセキュリティが強化された **Windows** ファイアウォール］→［受信の規則］の一覧にある［ネットワーク探索 (**SSDP** 受信)］を有効にします。

※開発環境では、上記の設定を **LEB SDK** のインストーラにより行いますが、実行環境では当設定を行う必要があります。

LEBEnum_CreateHandle

LEBEnum のハンドルを生成します。

書式

```
LEBRET LEBEnum_CreateHandle(
        LPHANDLE lphEnum
        );
```

引数

[out] lphEnum　　生成されたハンドルが返されます

戻り値

LEBRET_SUCCESS 成功
LEBRET_INTERNAL_ERROR 失敗
LEBRET_BAD_ARGUMENTS **lphEnum** が **NULL** の場合

LEBEnum_CloseHandle

LEBEnum のハンドルを解放します。

書式

```
LEBRET LEBEnum_CloseHandle(
       HANDLE hEnum
       );
```

引数

[in] hEnum 解放するハンドル

戻り値

LEBRET_SUCCESS ハンドルの解放に成功
LEBRET_INVALID_HANDLE 不正なハンドル

LEBEnum_Start

SSDP を用いて **LEB Pro/Express** の検出を開始します。

まず現在ネットワークに接続されている **LEB Pro/Express** を検出し **LEBENUM_ADDED** を発行します。検出が完了すると **LEBENUM_COMPLETE** を発行します。

それ以降は新たにネットワーク上で起動された **LEB Pro/Express** を検出した時に **LEBENUM_ADDED** を、正常にシャットダウンされた **LEB Pro/Express** を検出した時には **LEBENUM_REMOVED** を発行します。

コールバック関数の引数 **lpszUniqueName** は **nEvent** が **LEBENUM_ADDED** または **LEBENUM_REMOVED** の場合に設定されます。**lpszAddrss** は **nEvent** が **LEBENUM_ADDED** の場合に設定されます。それ以外の場合は **NULL** になります。

書式

```
LEBRET LEBEnum_Start(
       HANDLE hEnum,
       LEBEnum_EventCallback lpfnCallback,
       LONG_PTR lpParam
       );
```

引数

[in] hEnum ハンドル

[in] lpfnCallback イベント発生時のコールバック関数

[in] lpParam コールバック関数の第一引数

戻り値

LEBRET_SUCCESS 成功

LEBRET_INVALID_HANDLE 不正なハンドル

LEBRET_INTERNAL_ERROR **UPnP** ライブラリでエラーが発生

コールバック関数の書式

```
VOID CALLBACK lpfnCallback(
       LONG_PTR lpParam,
       LEBENUM_EVENT nEvent,
       LPCWSTR lpszUniqueName,
       LPCWSTR lpszAddress
);
```

コールバック関数の引数

[in] lpParam 呼び出しで指定された **lpParam** の値

[in] nEvent イベントの種類

[in] lpszUniqueName **LEB Express/Pro** の固有名

[in] lpszAddress **LEB Express/Pro** の **IP** アドレス

nEvent の値

LEBENUM_ADDED	**LEB Express/Pro** を検出
LEBENUM_REMOVED	**LEB Express/Pro** が正常にシャットダウン
LEBENUM_COMPLETE	最初の検出処理が完了

LEBEnum_Stop

LEB Pro/Express の検出処理を停止します。

書式

```
LEBRET LEBEnum_Stop(
       HANDLE hEnum
       );
```

引数

[in] hEnum ハンドル

戻り値

LEBRET_SUCCESS	成功
LEBRET_INVALID_HANDLE	不正なハンドル

2. LEBConf

LEBConf は LEB Pro/Express の設定値の取得、変更を行います。現在の入力信号を検出し、それに対応する設定を行います。入力信号がない場合は利用できません。

LEBConf_CreateHandle

LEBConf のハンドルを生成します。

書式

```
LEBRET LEBConf_CreateHandle(
       LPHANDLE lphConf
       );
```

引数

[out] lphConf 生成されたハンドルが返されます

戻り値

LEBRET_SUCCESS 成功
LEBRET_INTERNAL_ERROR 失敗
LEBRET_BAD_ARGUMENTS **lphConf が NULL の場合**

LEBConf_CloseHandle

LEBConf のハンドルを解放します。

書式

```
LEBRET LEBConf_CloseHandle(
       HANDLE hConf
       );
```

引数

[in] hConf 解放するハンドル

戻り値

LEBRET_SUCCESS ハンドルの解放に成功

LEBRET_INVALID_HANDLE 不正なハンドル

LEBConf_Prepare

LEB Pro/Express に接続し設定値の一覧を（内部的に）取得し、完了してから制御を返します。

書式

```
LEBRET LEBConf_Prepare(
       HANDLE hConf,
       LONG nTimeout,
       LPCWSTR lpszHost,
       LPCWSTR lpszUser,
       LPCWSTR lpszPassword
       );
```

引数

[in] hConf ハンドル

[in] nTimeout タイムアウト時間［msec］

[in] lpszHost ホスト名または **IP** アドレス

[in] lpszUser ユーザー名

[in] lpszPassword パスワード

戻り値

LEBRET_SUCCESS 成功

LEBRET_INVAILD_HANDLE 不正なハンドル

LEBRET_INTERNAL_ERROR 内部処理でエラーが発生

LEBRET_TIMEOUT タイムアウト

LEBRET_FAILURE その他の失敗

LEBConf_PrepareAsync

LEB Pro/Express に接続し設定値の一覧を（内部的に）取得し、すぐに制御を返します。
完了すると **lpfnCompleted** で指定されたコールバック関数を呼び出します。

書式

```
LEBRET LEBConf_PrepareAsync(
        HANDLE hConf,
        LONG nTimeout,
        LPCWSTR lpszHost,
        LPCWSTR lpszUser,
        LPCWSTR lpszPassword,
        LEBConf_PrepareCompleted lpfnCompleted,
        LONG_PTR lpParam
        );
```

引数

[in] hConf ハンドル
[in] nTimeout タイムアウト時間［**msec**］
[in] lpszHost ホスト名または **IP** アドレス
[in] lpszUser ユーザー名
[in] lpszPassword パスワード
[in] lpfnCompleted 完了時のコールバック関数
[in] lpParam コールバック関数の第一引数

戻り値

LEBRET_PENDING 実行中
LEBRET_INVAILD_HANDLE 不正なハンドル

コールバック関数の書式

```
VOID CALLBACK lpfnCompleted(
        LONG_PTR lpParam,
        LEBRET nRet
);
```

コールバック関数の引数

[in] lpParam 呼び出しで指定された **lpParam** の値

[in] nRet 完了コード

nRet の値

LEBRET_SUCCESS	成功
LEBRET_INTERNAL_ERROR	内部処理でエラーが発生
LEBRET_TIMEOUT	タイムアウト
LEBRET_CANCELED	完了前にハンドルが閉じられた
LEBRET_FAILURE	その他の失敗

LEBConf_Get

LEBConf_Prepare(Async) で内部的に取得した設定値を返します。

書式

```
LEBRET LEBConf_Get(
       ANDLE hConf,
       PCWSTR lpszName,
       PWSTR lpszValue,
       PLONG lpnLength
       );
```

引数

[in] hConf ハンドル

[in] lpszName 設定値の名前（「設定値」の項目を参照）

[out] lpszValue 設定値を受け取るバッファ

[in,out] lpnLength バッファの文字数（**NULL** 文字を含む）
必要な文字数が返されます

戻り値

LEBRET_SUCCESS	成功
LEBRET_INVALID_HANDLE	不正なハンドル
LEBRET_BAD_STATE	**LEBConf_Prepare(Async)** が事前に呼び出されていない
LEBRET_NOT_FOUND	設定値の名前が不正
LEBRET_BAD_ARGUMENTS	**lpszName,lpnLength** が **NULL** の場合
LEBRET_INSUFFICIENT_BUFFER	バッファが不足

LEBConf_Set

新しい設定値を（内部的に）登録します。**Apply** を実行するまで新しい設定値は **LEB Pro/Express** に反映しません。

書式

```
LEBRET LEBConf_Set(
       ANDLE hConf,
       PCWSTR lpszName,
       PCWSTR lpszValue
       );
```

引数

[in] hConf	ハンドル
[in] lpszName	設定値の名前（「設定値」の項目を参照）
[in] lpszValue	設定値

戻り値

LEBRET_SUCCESS	成功
LEBRET_INVALID_HANDLE	不正なハンドル
LEBRET_BAD_STATE	**LEBConf_Prepare(Async)** が事前に呼び出されていない
LEBRET_NOT_FOUND	設定値の名前が不正
LEBRET_BAD_ARGUMENTS	設定値が不正

LEBConf_Apply

設定値を **LEB Pro/Express** に反映します。必要がある場合 **LEB Pro/Express** の再起動を行います。完了してから制御を返します。

書式

```
LEBRET LEBConf_Apply(
       HANDLE hConf,
       LONG nTimeout
       );
```

引数

[in] hConf ハンドル

[in] nTimeout タイムアウト時間［**msec**］

戻り値

LEBRET_SUCCESS	成功
LEBRET_INVALID_HANDLE	不正なハンドル
LEBRET_INTERNAL_ERROR	内部処理でエラーが発生
LEBRET_BAD_STATE	**LEBConf_Prepare(Async)** が事前に呼び出されていない
LEBRET_TIMEOUT	タイムアウト
LEBRET_FAILURE	その他の失敗

LEBConf_ApplyAsync

設定値を **LEB Pro/Express** に反映します。必要がある場合 **LEB Pro/Express** の再起動を行います。すぐに制御を返します。完了すると **lpfnCompleted** で指定されたコールバック関数を呼び出します。

書式

```
LEBRET LEBConf_ApplyAsync(
       HANDLE hConf,
       LONG nTimeout,
       LEBConf_ApplyCompleted lpfnCompleted,
       LONG_PTR lpParam
       );
```

引数

[in] hConf ハンドル

[in] nTimeout タイムアウト時間［msec］

[in] lpfnCompleted 完了時のコールバック関数

[in] lpParam コールバック関数の第一引数

戻り値

LEBRET_PENDING 実行中

LEBRET_INVALID_HANDLE 不正なハンドル

コールバック関数の書式

```
VOID CALLBACK lpfnCompleted(
       LONG_PTR lpParam,
       LEBRET nRET
);
```

コールバック関数の引数

[in] lpParam 呼び出しで指定された **lpParam** の値

[in] nRet 完了コード

nRet の値

LEBRET_SUCCESS	成功
LEBRET_INTERNAL_ERROR	内部処理でエラーが発生
LEBRET_BAD_STATE	**LEBConf_PrepareAsync** が事前に呼び出されていない
LEBRET_TIMEOUT	タイムアウト
LEBRET_CANCELED	完了前にハンドルが閉じられた
LEBRET_FAILURE	その他の失敗

LEBConf_GetErrorMessage

LEBConf_Prepare、**LEBConf_PrepareAsync**、**LEBConf_Get**、**LEBConf_Set**、**LEBConf_Apply**、**LEBConf_ApplyAsync** のうち直前の操作で発生したエラーメッセージを返します。

書式

```
LEBRET LEBConf_GetErrorMessage(
       HANDLE hConf,
       LPWSTR lpszMessage,
       LPLONG lpnLength
       );
```

引数

[in] hConf	ハンドル
[out] lpszMessage	エラーメッセージを受け取るバッファ
[in,out] lpnLength	バッファの文字数（**NULL** 文字を含む） 必要な文字数が返る

戻り値

LEBRET_SUCCESS	成功
LEBRET_INVALID_HANDLE	不正なハンドル
LEBRET_BAD_ARGUMENTS	**lpnLength** が **NULL** の場合
LEBRET_INSUFFICIENT_BUFFER	バッファが不足

設定値

LEBConf_Get、LEBConf_Set の lpszName 引数に渡す設定値の名称の一覧と値を示します。設定値は数値型のものも含めてすべてワイド文字列として入出力します。大文字小文字は区別しません。

設定値には依存関係を持っているものがあり、組み合わせによっては以下に記載されているものでも **Apply** が受けつけられない場合があります。その場合は **LEB_GetErrorMessage** を用いることで原因の手がかりが得られる場合があります。

distribution.fec_packet_count_and_interval

FEC の設定

"2,1"、"2,2"、"2,4"、"2,8"、"2,16"、"2,32"、"2,64"、"2,127"、
"3,1"、"3,2"、"3,4"、"3,8"、"3,16"、"3,32"、"3,63"、
"4,1"、"4,2"、"4,4"、"4,8"、"4,16"、"4,32"、"4,42"、
"5,1"、"5,2"、"5,4"、"5,8"、"5,16"、"5,25"、
"10,1"、"10,2"、"10,4"、"10,10"、"disabled"

distribution.path_mtu

MTU (Maximum Transmission Unit) の値

最小値 **576**

最大値 **1500**

distribution.mode

配信開始のタイミング

"auto" (起動時)

"on-demand" (オンデマンド)

distribution.ip_address

配信先アドレス

distribution.port_number

配信先ポート番号

最小値 **1**

最大値 **65533**

distribution.multicast_hop_limit

マルチキャスト時のホップリミットまたは **TTL (Time To Live)**

最小値 **0**

最大値 **255**

distribution.packet_count

ひとつの **RTP** パケットに格納する **TS** パケットの個数

最小値 **2**

最大値 **256**

distribution.max_unicast_recipients

同時に配信できるユニキャストストリームの数

最小値 **1**

最大値 **16**

distribution.bandwidth_control

通信帯域の上限**(kbps)**

最小値 **0**（無効）

最大値 **100000**

input.video_port

ビデオ入力の指定

"SDI1"、"SDI2"、"HDMI"、"CVBS"

input.audio_channels

オーディオ入力のチャネル指定

"mute"、"mono"、"stereo"

encoding.add_null_packets

NULL パケット挿入の設定

"false"、"true"

encoding.horizontal_resoluton_for_1080i

ビデオフォーマットの設定

※入力信号が 1080i の場合のみ"1440x1080"または"1920x1080"の指定が可能

"1440x1080"、"1920x1080"

encoding.video_bitrate

ビデオビットレート

最小値 500

最大値 24000

encoding.audio_format

オーディオフォーマットの設定

"Dolby Digital"、"MPEG-1 layer2"

encoding.audio_channels

オーディオ出力のチャネルの設定

"mono"、"stereo"

encoding.audio_bitrate

オーディオビットレート

"32", "64", "96", "128", "192", "256", "320", "384", "448"

3. LEBCoder

LEBCoder では **MEDIAEDGE-SWT4** または **LEB SDK** 内蔵の受信(再生)機能を利用することができます。

LEBCoder_Prepare を実行すると **MEDIAEDGE-SWT4** を使用します。この場合はストリームの受信、プレビューウィンドウの表示、録画、静止画キャプチャを行うことができます。
MEDIAEDGE-SWT4 がインストールされていない場合は **LEBCoder_Prepare** は失敗します。
MEDIAEDGE-SWT4 のドングルまたは **MEDIAEDGE-DAS4** がない場合は **LEBCoder_Start** を行った時に失敗します。 **MEDIAEDGE-SWT4** が録画、静止画キャプチャをサポートしていないバージョンだった場合、それぞれの操作が失敗します。(**MEDIAEDGE-SWT4 Version 1.20** 以降対応)

LEBCoder_PrepareNoWindow を実行すると **LEB SDK** 内蔵の受信機能を使用します。この場合はストリームの受信、録画を行うことができます。

LEBCoder_CreateHandle

LEBCoder のハンドルを生成します。

書式

```
LEBRET LEBCoder_CreateHandle(
       LPHANDLE lphCoder
       );
```

引数

[out] lphCoder　　生成されたハンドルが返されます

戻り値

LEBRET_SUCCESS　　成功
LEBRET_INTERNAL_ERROR　失敗
LEBRET_BAD_ARGUMENTS　**lphCoder** が **NULL** の場合

LEBCoder_CloseHandle

LEBCoder のハンドルを解放します。

書式

```
LEBRET LEBCoder_CloseHandle(
        HANDLE hCoder
        );
```

引数

[in] hCoder　　解放するハンドル

戻り値

LEBRET_SUCCESS　　ハンドルの解放に成功

LEBRET_INVALID_HANDLE　　不正なハンドル

LEBCoder_Prepare

LEBCoder の機能を提供するために **MEDIAEDGE-SWT4** コントロールウィンドウ（プレビューウィンドウ）を作成します。**MEDIAEDGE-SWT4** がインストールされていない場合は失敗します。

書式

```
LEBRET LEBCoder_Prepare(
        HANDLE hCoder,
        LONG nStyle,
        LPCRECT lpRect,
        HWND hParentWnd,
        LEBCoder_EventCallback lpfnCallback,
        LONG_PTR lpParam
        );
```

引数

[in] hCoder ハンドル

[in] nStyle プレビューウィンドウのウィンドウスタイル

[in] lpRect プレビューウィンドウのサイズ

[in] hParentWnd プレビューウィンドウの親ウィンドウのハンドル

[in] lpfnCallback イベント発生時のコールバック関数

[in] lpParam コールバック関数の第一引数

戻り値

LEBRET_SUCCESS 成功

LEBRET_INVALID_HANDLE 不正なハンドル

LEBRET_BAD_ARGUMENTS **lpRect** が **NULL** の場合

LEBRET_INTERNAL_ERROR **MEDIAEDGE-SWT4** コントロールの作成に失敗

コールバック関数の書式

```
VOID CALLBACK lpfnCallback(
       LONG_PTR lpParam,
       LEBCODER_EVENT nEvent,
       LONG nCode
);
```

コールバック関数の引数

[in] lpParam 呼び出しで指定された **lpParam** の値

[in] nEvent イベントの種類（「**LEBCODER_EVENT**」の項目を参照）

[in] nCode 種類ごとの詳細コード（「**LEBCODER_EVENT**」の項目を参照）

LEBCoder_PrepareNoWindow

LEBCoder の機能を提供するために **LEB SDK** 内蔵の受信機能を準備します。

書式

```
LEBRET LEBCoder_PrepareNoWindow(
        HANDLE hCoder,
        LEBCoder_EventCallback lpfnCallback,
        LONG_PTR lpParam
        );
```

引数

[in] hCoder ハンドル

[in] lpfnCallback イベント発生時のコールバック関数

[in] lpParam コールバック関数の第一引数

戻り値

LEBRET_SUCCESS 成功

LEBRET_INVALID_HANDLE 不正なハンドル

コールバック関数の書式

```
VOID CALLBACK lpfnCallback(
        LONG_PTR lpParam,
        LEBCODER_EVENT nEvent,
        LONG nCode
);
```

コールバック関数の引数

[in] lpParam 呼び出しで指定された **lpParam** の値

[in] nEvent イベントの種類（「**LEBCODER_EVENT**」の項目を参照）

[in] nCode 種類ごとの詳細コード（「**LEBCODER_EVENT**」の項目を参照）

LEBCoder_Destroy

作成済みの受信(再生)機能（プレビューウィンドウ）を破棄します。

書式

```
LEBRET LEBCoder_Destroy(
       HANDLE hCoder
       );
```

引数

[in] hCoder ハンドル

戻り値

LEBRET_SUCCESS 成功
LEBRET_INVALID_HANDLE 不正なハンドル

LEBCoder_GetHwnd

MEDIAEDGE-SWT4 コントロールウィンドウのウィンドウハンドルを取得します。

書式

```
LEBRET LEBCoder_GetHwnd(
       HANDLE hCoder,
       HWND* lphWnd
       );
```

引数

[in] hCoder ハンドル
[out] lphWnd **MEDIAEDGE-SWT4** コントロールウィンドウのハンドル

戻り値

LEBRET_SUCCESS 成功
LEBRET_INVALID_HANDLE 不正なハンドル
LEBRET_BAD_STATE **LEBCoder_Prepare** が事前に呼び出されていない
LEBRET_BAD_ARGUMENTS **lphWnd** が **NULL**
LEBRET_FAILURE その他の失敗

LEBCoder_Resize

MEDIAEDGE-SWT4 コントロールウィンドウのサイズを変更します。

書式

```
LEBRET LEBCoder_Resize(
       HANDLE hCoder,
       LPCRECT lpRect
       );
```

引数

[in] hCoder ハンドル

[in] lpRect サイズ

戻り値

LEBRET_SUCCESS	成功
LEBRET_INVALID_HANDLE	不正なハンドル
LEBRET_BAD_STATE	**LEBCoder_Prepare** が事前に呼び出されていない
LEBRET_BAD_ARGUMENTS	**lpRect** が **NULL**
LEBRET_FAILURE	その他の失敗

LEBCoder_Start

ストリームの受信を開始します。**MEDIAEDGE-SWT4** がドングルあるいは **MEDIAEDGE-DAS4** で認証されない場合は失敗します。

書式

```
LEBRET LEBCoder_Start(
       HANDLE hCoder,
       LPCWSTR lpszHost
       );
```

引数

[in] hCoder ハンドル

[in] lpszHost ホスト名または IP アドレス

戻り値

LEBRET_SUCCESS	成功
LEBRET_INVALID_HANDLE	不正なハンドル
LEBRET_BAD_STATE	**LEBCoder_Prepare(NoWindow)** が事前に呼び出されていない
LEBRET_INTERNAL_ERROR	内部処理でエラーが発生
LEBRET_FAILURE	その他の失敗

LEBCoder_Stop

ストリームの受信を停止します。録画していた場合録画も停止します。

書式

```
LEBRET LEBCoder_Stop(
       HANDLE hCoder
       );
```

引数

[in] hCoder ハンドル

戻り値

LEBRET_SUCCESS	成功
LEBRET_INVALID_HANDLE	不正なハンドル
LEBRET_BAD_STATE	**LEBCoder_Prepare(NoWindow)** が事前に呼び出されていない
LEBRET_INTERNAL_ERROR	内部処理でエラーが発生
LEBRET_FAILURE	その他の失敗

LEBCoder_Capture

静止画キャプチャを行います。**MEDIAEDGE-SWT4** が対応していないバージョンである場合失敗します。(**MEDIAEDGE-SWT4 Version 1.20** 以降対応)

書式

```
LEBRET LEBCoder_Capture(
       HANDLE hCoder,
       LPCWSTR lpszFilePath
       );
```

引数

[in] hCoder ハンドル

[in] lpszFilePath 静止画ファイルの保存先

戻り値

LEBRET_SUCCESS	成功
LEBRET_INVALID_HANDLE	不正なハンドル
LEBRET_BAD_STATE	ストリームの受信が行われていない
LEBRET_INTERNAL_ERROR	内部処理でエラーが発生
LEBRET_FAILURE	その他の失敗

LEBCoder_StartRecordFile

ファイルへの録画を開始します。**MEDIAEDGE-SWT4** が対応していないバージョンである場合失敗します。(**MEDIAEDGE-SWT4 Version 1.20** 以降対応)

書式

```
LEBRET LEBCoder_StartRecordFile(
       HANDLE hCoder,
       LPCWSTR lpszFilePath
       );
```

引数

[in] hCoder ハンドル

[in] lpszFilePath 静止画ファイルの保存先

戻り値

LEBRET_SUCCESS	成功
LEBRET_INVALID_HANDLE	不正なハンドル
LEBRET_BAD_STATE	ストリームの受信が行われていない
LEBRET_INTERNAL_ERROR	内部処理でエラーが発生
LEBRET_FAILURE	その他の失敗

LEBCoder_StartRecordMemory

メモリ転送を開始します。**MEDIAEDGE-SWT4** が対応していないバージョンである場合失敗します。
（**MEDIAEDGE-SWT4 Version 1.20** 以降対応）

書式

```
LEBRET LEBCoder_StartRecordMemory(
       HANDLE hCoder,
       LONG nSize,
       LEBCoder_DataCallback lpfnCallback,
       LONG_PTR lpParam
       );
```

引数

[in] hCoder ハンドル

[in] nSize 一回あたりに読み込む

[in] lpfnCallback 読み込んだデータのコールバック関数

[in] lpParam コールバック関数の第一引数

戻り値

LEBRET_SUCCESS	成功
LEBRET_INVALID_HANDLE	不正なハンドル
LEBRET_BAD_STATE	ストリームの受信が行われていない
LEBRET_INTERNAL_ERROR	内部処理でエラーが発生
LEBRET_FAILURE	その他の失敗

コールバック関数の書式

```
VOID CALLBACK lpfnCallback(
       LONG_PTR lpParam,
       LPBYTE lpData,
       LONG nSize
);
```

コールバック関数の引数

[in] lpParam	呼び出しで指定された **lpParam** の値
[in] lpData	読み込んだデータ
[in] nSize	データのサイズ 不連続があった場合はサイズ **0** のコールバックが行われます

LEBCoder_StopRecord

録画またはメモリ転送を停止します。

書式

```
LEBRET LEBCoder_StopRecord(
       HANDLE hCoder
       );
```

引数

[in] hCoder	ハンドル

戻り値

LEBRET_SUCCESS 成功

LEBRET_INVALID_HANDLE 不正なハンドル

LEBCODER_EVENT

LEBCODER_START

ストリーム受信の開始

LEBCODER_STOP

ストリーム受信の停止

LEBCODER_END_OF_CONTENT

ストリームの終端通知を受け取った場合に発生

LEBCODER_DONGLE_ERROR

受信開始時に **MEDIAEDGE-SWT4** のドングルエラーが発生

LEBCODER_DECODE_ERROR

MEDIAEDGE-SWT4 でデコードエラーが発生

LEBCODER_STREAM_CONTROL_ERROR

ストリームの再生制御に失敗

以下の **Code** を伴います

```
LEBCODER_STREAM_CONTROL_ERROR_MALFORMED_URL = 1,
```

URL が不正

```
LEBCODER_STREAM_CONTROL_ERROR_INTERNAL_ERROR = 2,
```

受信(再生)機能の内部エラー

```
LEBCODER_STREAM_CONTROL_ERROR_CONNECTION_FAILURE = 3,
```

LEB Pro/Express への接続失敗

```
LEBCODER_STREAM_CONTROL_ERROR_HOST_NOT_FOUND = 4,
```

LEB Pro/Express が見つからない

LEBCODER_STREAM_CONTROL_ERROR_INIT_ERROR = 5,

受信(再生)機能の初期化に失敗

LEBCODER_STREAM_CONTROL_ERROR_RTSP_SERVER_DOWN = 6,

サービス **"MeSwtSvr"**との通信に失敗

LEBCODER_STREAM_CONTROL_ERROR_KEEPALIVE_FAILURE = 7,

LEB Pro/Express の生存確認に失敗

LEBCODER_RECORDING_START

録画の開始

LEBCODER_RECORDING_STOP

録画の停止

LEBCODER_RECORDING_ERROR

録画時にエラーが発生

以下の **Code** を伴います

LEBCODER_RECORDING_ERROR_BUFFER_OVERFLOW = 2,

内部バッファでオーバーフロー発生

LEBCODER_RECORDING_ERROR_DISK_FULL = 3,

保存先のディスクに空き容量がない

LEBCODER_RECORDING_ERROR_FILE_ERROR = 4,

ファイル関連のエラー(保存先パスが不正等)

トムソン・カノープス株式会社
本社／〒651-2241 神戸市西区室谷 1-2-2