このマニュアルについて

CLIP STUDIO PAINT で使用できるフィルタプラグインを、ユーザーの皆様に作成していただけるよう、プラグインの概要や、作成方法を記したものになります。
このマニュアルは、CLIP STUDIO PAINT Ver1.10.5 時点での情報です。
C++プログラミング、および開発ツールの使用経験があることを前提にしています。

概要

・プラグインモジュールのインターフェイス、サンプルによる説明
　フィルタプラグイン処理の作成方法を、サンプルを交えて説明します。
・基本型、定数、提供される関数群
　プラグインを作成するに当たり、定義されている関数群を記載します。

また、このマニュアルでは、プラグイン自体をプラグインモジュール、プラグインの呼び出し側(CLIP STUDIO PAINT)をプラグインホストと記載します。

プラグインについて

開発環境
Windows ：VisualStudio2019
Mac OS X：xcode12.2 以上

ビルドして出力するプラグインファイルは、Windows、Mac OS X ともに cpm ファイルになります。
Mac OS X 版は必ずプラグインごとに適切な BundleIdentifier を設定してください。

このファイルは、「マイドキュメント」または「ドキュメント」フォルダ中にある、CELSYS¥CLIPStudioModule¥PlugIn¥PAINT フォルダ(Windows)/「書類」フォルダ中にある、CELSYS/CLIPStudioModule/PlugIn/PAINT フォルダ （Mac OS X） に移動することで使用できます。
作成したプラグインは、CLIP STUDIO PAINT EX でしか使用できませんのでご注意ください。
また、EX の場合でも機能制限状態では使用できません。

アクティベーションへの対応について

アクティベーションに対応させる場合は、プリプロセッサの定義に TRIGLAV_PLUGIN_ACTIVATION を追加します。
アクティベーション用のプラグインはそのままでは使用できません。プラグインストアへの登録が必要です。

プラグインモジュールのインターフェイス

プラグインホストから、プラグインモジュールを呼び出すための機構について、説明を行います。

プラグインホストは、フィルタを実行するために TriglavPluginCall 関数を呼び出します。

```
void TRIGLAV_PLUGIN_API TriglavPluginCall( TriglavPlugInInt* result, TriglavPlugInPtr* data, TriglavPlugInInt selector, TriglavPlugInServer* pluginServer, TriglavPlugInPtr reserved )
```

この関数がプラグインモジュールのエントリーポイントとなります。

以下に、TriglavPlugInCall 関数の引数のリストを示します。

引数	result	out	処理が成功したらkTriglavPlugInCallResultSuccess 処理が失敗したらkTriglavPlugInCallResultFailed が代入されるように設定します。
	data	in/out	ユーザーによって任意の値を設定できます。 プラグインが終了するまで値が保持されます。
	selector	in	セレクター(後述)
	plugInServer	in	プラグインサーバー(後述)
	reserved	-	将来使用するため、予約されている引数です。

関数名は TriglavPlugInCall である必要があります。

■セレクター

ホストから要求された操作タイプを示します。このパラメータに対して場合分けでそれぞれの処理を実装するようにします。

セレクターには以下のいずれかの定数が代入されています。

kTriglavPlugInSelectorModuleInitialize	プラグインモジュールの初期化を実装します。
kTriglavPlugInSelectorModuleTerminate	プラグインモジュールの終了処理を実装します。
kTriglavPlugInSelectorFilterInitialize	フィルタの初期化を実装します。
kTriglavPlugInSelectorFilterTerminate	フィルタの終了処理を実装します。
kTriglavPlugInSelectorFilterRun	フィルタの計算処理を実装します。

■プラグインサーバー

プラグインで使用できる関数群、ホストオブジェクトなど、プラグインで処理を行うに当たって必要なデータを提供します。

以下に、プラグインサーバー構造体を示します。

TriglavPlugInHostObject	hostObject	ホストオブジェクト
TriglavPlugInRecordSuite	recordSuite	レコードスイート
TriglavPlugInServiceSuite	serviceSuite	サービススイート

・ホストオブジェクト

　プラグインホスト側のデータを参照するための構造体です。下記レコードスイート、サービススイートで提供される関数の引数として使用します。

・レコードスイート

　selecter の値に対する、特別な処理を行う関数群を提供します。

下記に、selecter の値にして取得できるレコードを示します。取得できなかった場合、NULL が代入されます。

kTriglavPlugInSelectorModuleInitialize	モジュール初期化レコード
kTriglavPlugInSelectorModuleTerminate	現在対応するレコードはありません。
kTriglavPlugInSelectorFilterInitialize	フィルタ初期化レコード
kTriglavPlugInSelectorFilterTerminate	現在対応するレコードはありません。
kTriglavPlugInSelectorFilterRun	フィルタ実行レコード

関数の詳細は 39～46 ページを参照してください。

・サービススイート

selecter がどの値であっても使用可能な関数群を提供します。

関数の詳細は 19～38 ページを参照してください。

サンプルによる説明

サンプルプロジェクトの HSV フィルタを例に、処理の流れについてを主に説明します。

HSV フィルタの計算自体の説明は行いません。

■プラグインモジュールの初期化

プラグインモジュールの初期化部分のコードです。

引数selectorがkTriglavPlugInSelectorModuleInitializeである時、この処理が実行されます。

モジュール初期化レコードを取得します。

```
TriglavPlugInModuleInitializeRecord* pModuleInitializeRecord =
                                        (*pluginServer).recordSuite.moduleInitializeRecord;
```

このレコードに、必要な情報を設定していきます。

文字列を扱う関数を使うために、文字列サービスを取得します。

```
TriglavPlugInStringService*    pStringService   = (*pluginServer).serviceSuite.stringService;
```

ホストのバージョンを取得し、プラグインモジュールが実行できる環境かどうかを判定します。

```
TriglavPlugInInt   hostVersion;
(*pModuleInitializeRecord).getHostVersionProc(&hostVersion,(*pluginServer).hostObject);
if( hostVersion >= kTriglavPlugInNeedHostVersion )
  {
    //   後述
  }
```

getHostVersionProc 関数を使用し、ホストのバージョンを取得します。ホストのバージョンと、このプラグインが必要としているバージョンを比較し、前者が小さければ result に kTriglavPlugInCallResultFailed を代入するようにして下さい。

基本的には、kTriglavPlugInNeedHostVersion と比較します。

モジュール初期化レコードに moduleID を設定します。

```
TriglavPlugInStringObject   moduleID  = NULL;
const char* moduleIDString = "A9EE0802-84E7-4847-87D8-9EBAC916EEE4";
(*pStringService).createWithAsciiStringProc(&moduleID,moduleIDString,static_cast<TriglavPlugInInt>(::strlen(moduleID
String)));
(*pModuleInitializeRecord).setModuleIDProc((*pluginServer).hostObject,moduleID);
```

setModuleIDProc関数を使用し、moduleIDを設定します。モジュールIDは、各プラグインで唯一のIDである必要があります。モジュールIDにはGUIDを使用してください。

変数 moduleIDはもう必要がないため解放します。

```
(*pStringService).releaseProc(moduleID);
```

モジュール初期化レコードにプラグインの種類を設定します。

```
(*pModuleInitializeRecord).setModuleKindProc((*pluginServer).hostObject,kTriglavPlugInModuleKindFilter);
```

setModuleKindProc関数を使用し、プラグインの種類を設定します。

フィルタのプラグインを作成する場合は、kTriglavPlugInModuleSwitchKindFilterを設定します。

プラグインモジュールで常に使用するデータを設定します。

```
HSVFilterInfo* pFilterInfo = new HSVFilterInfo;
*data = pFilterInfo;
```

フィルタ実行に必要な変数hue,saturation,valueと、UIの情報を扱うためのプロパティサービスを構造体にしたHSVFilterInfoの領域をdataに確保しておきます。後の処理で、この領域にデータを代入したり、参照したりします。

HSVFilterInfo構造体

```
struct   HSVFilterInfo
{
  Int hue;
  Int saturation;
  Int value;

  TriglavPlugInPropertyService*  pPropertyService;
};
```

処理が成功したら、変数resultにkTriglavPlugInCallResultSuccessを代入します。

```
*result  = kTriglavPlugInCallResultSuccess;
```

この処理が行われない場合、プラグインホスト側にフィルタが登録されません。

プラグインモジュールの終了処理

プラグインモジュールの終了部分のコードです。

引数selectorがkTriglavPlugInSelectorModuleTerminateである時、この処理が実行されます。

```
//        プラグインの終了処理
HSVFilterInfo* pFilterInfo = static_cast<HSVFilterInfo*>(*data);
delete pFilterInfo;
*data    = NULL;
*result  = kTriglavPlugInCallResultSuccess;
```

上述「プラグインの初期化」で確保した領域を解放します。

フィルタの初期化

フィルタの初期化のコード例を以下に示します。

引数selectorがkTriglavPlugInSelectorFilterInitializeである時、この処理が実行されます。

フィルタの初期化レコードへ設定を行うために、レコードスイートのポインタとホストオブジェクトを取得します。

```
TriglavPlugInRecordSuite* pRecordSuite = &(*pluginServer).recordSuite;
TriglavPlugInHostObject hostObject = (*pluginServer).hostObject;
```

文字列を扱う関数を使うために、文字列サービスを取得します。

```
TriglavPlugInStringService* pStringService = (*pluginServer).serviceSuite.stringService;
TriglavPlugInPropertyService* pPropertyService = (*pluginServer).serviceSuite.propertyService;
```

また、プロパティを作成するために、プロパティサービスを取得します。

フィルタ名の設定を行います。フィルタ名は、メニューと、ダイアログボックスのタイトルとして表示されます。

```
//        フィルタカテゴリ名とフィルタ名の設定
```

```
TriglavPlugInStringObject filterCategoryName = NULL;
TriglavPlugInStringObject filterName = NULL;
(*pStringService).createWithStringIDProc(&filterCategoryName,kStringIDFilterCategoryName,(*pluginServer).hostObj
ect);
(*pStringService).createWithStringIDProc(&filterName,kStringIDFilterName,(*pluginServer).hostObject);
TriglavPlugInFilterInitializeSetFilterCategoryName(pRecordSuite,hostObject,filterCategoryName,'c');
TriglavPlugInFilterInitializeSetFilterName(pRecordSuite,hostObject,filterName,'h');
(*pStringService).releaseProc(filterCategoryName);
(*pStringService).releaseProc(filterName);
```

1. **createWithStringIDProc** 関数を使用し、リソースの文字列テーブルで定義した文字列を取得します。
2. **TriglavPlugInFilterInitializeSetFilterCategoryName** 関数で、フィルタ初期化レコードにフィルタカテゴリ名を設定します。また、アクセスキーに[c]を設定しています。
3. **TriglavPlugInFilterInitializeSetFilterName** 関数で、フィルタ初期化レコードにフィルタ名を設定します。また、アクセスキーに[h]を設定しています。
4. 設定した文字列オブジェクトはもう必要がないため、**releaseProc** 関数で解放します。

フィルタを実行するレイヤーのターゲットを設定します。

```
//        ターゲット
TriglavPlugInInt target[]={kTriglavPlugInFilterTargetKindRasterLayerRGBAlpha};
TriglavPlugInFilterInitializeSetTargetKinds(pRecordSuite,hostObject,target,1);
```

この場合、表現色がカラーのラスターレイヤーのみターゲットとしています。

それ以外のレイヤーが編集中である場合は、コマンド自体が無効になり、ユーザーが実行できないようになります。

ここから先は、ダイアログボックスに表示する要素を設定する部分を説明します。

最終的に、以下のダイアログボックスが表示されるようにします。

プレビューのチェックボックスを表示させます。

```
//        プレビュー
TriglavPlugInFilterInitializeSetCanPreview(pRecordSuite,hostObject,true);
```

TriglavPlugInFilterInitializeSetCanPreview 関数で true を設定することで、プレビューのチェックボックスを表示することが出来ます。

プロパティオブジェクトを作成します。

```
//        プロパティの作成
TriglavPlugInPropertyObject propertyObject;
```

```
(*pPropertyService).createProc(&propertyObject);
```

このオブジェクトに、ダイアログに表示するパラメータ「色相、彩度、明度」の情報を設定していきます。

色相パラメータの情報を設定します。

```
//		色相
TriglavPlugInStringObject hueCaption = NULL;
(*pStringService).createWithStringIDProc(&hueCaption,kStringIDItemCaptionHue,(*pluginServer).hostObject);

(*pPropertyService).addItemProc(propertyObject,kItemKeyHue,kTriglavPlugInPropertyValueTypeInteger,kTriglavPlugInPropertyValueKindDefault,kTriglavPlugInPropertyInputKindDefault,hueCaption,'h');
(*pPropertyService).setIntegerValueProc(propertyObject,kItemKeyHue,0);
(*pPropertyService).setIntegerDefaultValueProc(propertyObject,kItemKeyHue,0);
(*pPropertyService).setIntegerMinValueProc(propertyObject,kItemKeyHue,-180);
(*pPropertyService).setIntegerMaxValueProc(propertyObject,kItemKeyHue,180);
(*pStringService).releaseProc(hueCaption);
```

1. **createWithStringIDProc** 関数を使用し、文字列テーブルで定義した「色相」の文字列を取得します。
2. **addItemProc**関数を使用し、アイテムを追加します。

   色相はkTriglavPlugInInt型の値を取るためkTriglavPlugInPropertyValueTypeIntegerを設定します。

   また、UIをスライダーにするためkTriglavPlugInPropertyInputKindDefaultを設定します。

   キャプションに先ほど取得した「色相」の文字列を設定し、アクセスキーに[h]を設定します。
3. **setIntegerValueProc** 関数を使用し、色相に 0 の値を設定します。これにより、フィルタ実行直後の色相の値は 0 になります。
4. **setIntegetDefaultValueProc** 関数を使用し、色相のデフォルト値に 0 の値を設定します。
5. **setIntegerMinValueProc** 関数を使用し、色相の最小値に-180 を設定します。この設定よって、ユーザーは-181 以下の数値が設定できないようになります。
6. **setIntegerMaxValueProc** 関数を使用し、色相の最大値に 180 を設定します。この設定よって、ユーザーは 181 以上の数値が設定できないようになります。
7. 不要な文字列を解放します。

同様に、彩度・明度の設定を行いますが、ここでは省略します。

フィルタ初期化レコードにダイアログの詳細を設定します。

また、プロパティコールバック関数を設定し、アプリケーションからコールバック関数が呼ばれるようにします。

```
//		プロパティの設定
TriglavPlugInFilterInitializeSetProperty(pRecordSuite,hostObject,propertyObject);
TriglavPlugInFilterInitializeSetPropertyCallBack(pRecordSuite,hostObject,TriglavPlugInFilterPropertyCallBack,*data);
```

フィルタの終了処理

フィルタの終了処理のコードを以下に示します。

引数selectorがkTriglavPlugInSelectorFilterTerminateである時、この処理が呼ばれます。

```
//		フィルタの終了処理
*result = kTriglavPlugInCallResultSuccess;
```

このサンプルの場合、フィルタ終了時には特に処理を行いません。

フィルタの実行

フィルタの実行のコードを以下に示します。

引数selectorがkTriglavPlugInSelectorFilterRunである時、この処理が呼ばれます。

まず、処理に必要なデータを取得します。

```
TriglavPlugInPropertyObject propertyObject;
TriglavPlugInFilterRunGetProperty(pRecordSuite,&propertyObject,(*pluginServer).hostObject);

TriglavPlugInOffscreenObject sourceOffscreenObject;
TriglavPlugInFilterRunGetSourceOffscreen(pRecordSuite,&sourceOffscreenObject,(*pluginServer).hostObject);

TriglavPlugInOffscreenObject destinationOffscreenObject;
TriglavPlugInFilterRunGetDestinationOffscreen(pRecordSuite,&destinationOffscreenObject,(*pluginServer).hostObject);

TriglavPlugInRect selectAreaRect;
TriglavPlugInFilterRunGetSelectAreaRect(pRecordSuite,&selectAreaRect,(*pluginServer).hostObject);

TriglavPlugInOffscreenObject selectAreaOffscreenObject;
TriglavPlugInFilterRunGetSelectAreaOffscreen(pRecordSuite,&selectAreaOffscreenObject,(*pluginServer).hostObject);

TriglavPlugInInt r,g,b;
(*pOffscreenService).getRGBChannelIndexProc(&r,&g,&b,destinationOffscreenObject);

TriglavPlugInInt blockRectCount;
(*pOffscreenService).getBlockRectCountProc(&blockRectCount,destinationOffscreenObject,&selectAreaRect);
```

1. **TriglavPlugInFilterRunGetProperty** 関数でプロパティオブジェクトを取得します。
2. **TriglavPlugInFilterRunGetSourceOffscreen** 関数で処理実行元のオフスクリーンを取得します。
3. **TriglavPlugInFilterRunGetDestinationOffscreen** 関数で処理実行先のオフスクリーンを取得します。HSV の計算結果はこのオフスクリーンに反映されます。
4. **TriglavPlugInFilterRunGetSelectAreaRect** 関数で選択範囲の領域を取得、getSelectAreaOffscreenProc 関数で選択範囲のオフスクリーンを取得します。選択範囲を考慮した処理にするために必要です。
5. RGB インデックスを取得します。
6. 描画先のブロック数を取得します。

描画先オフスクリーンのブロック領域をすべて取得し、vector 配列に保持しておきます。この配列は後で使用します。

```
std::vector<TriglavPlugInRect> blockRects;
blockRects.resize(blockRectCount);
for( TriglavPlugInInt i=0; i<blockRectCount; ++i )
{
  (*pOffscreenService).getBlockRectProc(&blockRects[i],i,destinationOffscreenObject,&selectAreaRect);
}
```

処理の進捗をホスト側に知らせるための準備を行います。

```
TriglavPlugInFilterRunSetProgressTotal(pRecordSuite,(*pluginServer).hostObject,blockRectCount);
```

処理の進捗率をプラグインモジュール側で順次設定し、ホスト側に知らせます。

TriglavPlugInFilterRunSetProgressTotal 関数は、進捗率の全体を指定します。このサンプルの場合、1 ブロックを処理するにつれて進捗率を 1 増加させる処理にしています。そのため進捗率の全体は、総ブロック数になります。

while 文から処理のメインループに入ります。フィルタ処理が終了するまでこのループを抜けません。

この中で HSV の計算処理を行いますが、計算自体の説明は省略します。

```
while( true )
{
  if( restart )
  {
    restart = false;
    TriglavPlugInInt processResult;
    TriglavPlugInFilterRunProcess(pRecordSuite,&processResult,(*pluginServer).hostObject,kTriglavPlugInFilterRunProcessStateStart);
    if( processResult == kTriglavPlugInFilterRunProcessResultExit ){ break; }
    if( (*pFilterInfo).hue != 0 || (*pFilterInfo).saturation != 0 || (*pFilterInfo).value != 0 )
    {
      // 省略
    }
    else
    {
      // 何もしない
      blockIndex = blockRectCount;
      TriglavPlugInFilterRunUpdateDestinationOffscreenRect(pRecordSuite,(*pluginServer).hostObject,&selectAreaRect);
    }
  }
```

hue、saturation、value がすべて 0 の場合、画像処理が必要ないためなにも行いません。

TriglavPlugInFilterRunUpdateDestinationOffscreenRect 関数を呼び出し、描画先オフスクリーンの更新だけ行います。この場合、元画像と同じ画像に更新されます。

計算処理のメイン部分です。ブロックごとに処理を行っていきます。

```
if( blockIndex < blockRectCount )
{
  (*pFilterRunRecord).setProgressDoneProc((*pluginServer).hostObject,blockIndex);

  TriglavPlugInRect blockRect = blockRects[blockIndex];
  TriglavPlugInPoint pos;
  pos.x = blockRect.left;
  pos.y = blockRect.top;
  TriglavPlugInRect tempRect;

  TriglavPlugInPtr dstImageAddress;
  TriglavPlugInInt dstImageRowBytes;
  TriglavPlugInInt dstImagePixelBytes;

  (*pOffscreenService).getBlockImageProc(&dstImageAddress,&dstImageRowBytes,&dstImagePixelBytes,&tempRect,destinationOffscreenObject,&pos);

  TriglavPlugInPtr dstAlphaAddress;
  TriglavPlugInInt dstAlphaRowBytes;
  TriglavPlugInInt dstAlphaPixelBytes;

  (*pOffscreenService).getBlockAlphaProc(&dstAlphaAddress,&dstAlphaRowBytes,&dstAlphaPixelBytes,&tempRect,destinationOffscreenObject,&pos);
```

1. **setProgressDoneProc** 関数で、進捗を 1 つ進めます。
2. 変数 pos に、計算対象ブロックの左上の座標を代入します。

3. **getBlockImageProc** 関数でブロックの左上の座標のアドレスを取得します。このアドレスには、RGB のデータが保存されています。
4. **getBlockAlphaProc** 関数でブロックの左上の座標のアドレスを取得します。このアドレスには、アルファのデータが保存されています。

ブロックの中を、左上から 1 ピクセルずつ更新していきます。

```
if( dstImageAddress != NULL && dstAlphaAddress != NULL )
{
  if( selectAreaOffscreenObject == NULL )
  {
      //後述
  }
  else
  {
      //後述
  }
}
```

selectAreaOffscreenObject が NULL の場合は選択範囲がありません。この時は画像全体に更新をかけます。

選択範囲がある場合は、選択範囲を考慮して更新するようにします。この時、**TriglavPlugInFilterRunGetSelectAreaRect** 関数で取得できる選択範囲の領域外の画像を更新すると、取り消し動作が正常に機能しなくなるなどの問題が出ますので避けるようにして下さい。

選択範囲がない場合の処理です。ブロックの中すべてのピクセルに処理を行います。

```
BYTE* pDstImageAddressY = static_cast<BYTE*>(dstImageAddress);
BYTE* pDstAlphaAddressY = static_cast<BYTE*>(dstAlphaAddress);
for( int y=blockRect.top; y<blockRect.bottom; ++y )
{
  BYTE* pDstImageAddressX = pDstImageAddressY;
  BYTE* pDstAlphaAddressX = pDstAlphaAddressY;
  for( int x=blockRect.left; x<blockRect.right; ++x )
  {
    if( *pDstAlphaAddressX > 0 )
    {
      PIHSVFilter::SetHSV8(pDstImageAddressX[r],pDstImageAddressX[g],pDstImageAddressX[b],hue,saturation,value);
    }

    pDstImageAddressX += dstImagePixelBytes;
    pDstAlphaAddressX += dstAlphaPixelBytes;
  }
  pDstImageAddressY += dstImageRowBytes;
  pDstAlphaAddressY += dstAlphaRowBytes;
}
```

dstImageAddress と dstAlphaAddress を 1 ピクセル分ずつ行方向に進めていき、それぞれのピクセルに対して処理を行っていきます。一行の処理が終わり次第、処理対象を次の列に移動させます。

*pDstAlphaAddressX > 0 の場合のみ、HSV の計算を行います。*pDstAlphaAddress = 0 は、何も描画されていないピクセルですので、処理の必要がありません。

選択範囲がある場合の処理です。

```
TriglavPlugInPtr selectAddress;
TriglavPlugInInt selectRowBytes;
```

```
TriglavPlugInInt selectPixelBytes;

(*pOffscreenService).getBlockSelectAreaProc(&selectAddress,&selectRowBytes,&selectPixelBytes,&tempRect,selectAreaOff
screenObject,&pos);
if( selectAddress != NULL )
{
  BYTE* pDstImageAddressY = static_cast<BYTE*>(dstImageAddress);
  BYTE* pDstAlphaAddressY = static_cast<BYTE*>(dstAlphaAddress);
  const BYTE* pSelectAddressY = static_cast<const BYTE*>(selectAddress);
  for( int y=blockRect.top; y<blockRect.bottom; ++y )
  {
    BYTE* pDstImageAddressX = pDstImageAddressY;
    BYTE* pDstAlphaAddressX = pDstAlphaAddressY;
    const BYTE* pSelectAddressX = pSelectAddressY;
    for( int x=blockRect.left; x<blockRect.right; ++x )
    {
      if( *pDstAlphaAddressX > 0 )
      {
        if( *pSelectAddressX == 255 )
        {
          PIHSVFilter::SetHSV8(pDstImageAddressX[r],pDstImageAddressX[g],pDstImageAddressX[b],hue,saturation,value);
        }
        else if( *pSelectAddressX != 0 )
        {
          PIHSVFilter::SetHSV8Mask(pDstImageAddressX[r],pDstImageAddressX[g],pDstImageAddressX[b],*pSelectAddressX,h
ue,saturation,value);
        }
      }

      pDstImageAddressX += dstImagePixelBytes;
      pDstAlphaAddressX += dstAlphaPixelBytes;
      pSelectAddressX += selectPixelBytes;
    }
    pDstImageAddressY += dstImageRowBytes;
    pDstAlphaAddressY += dstAlphaRowBytes;
    pSelectAddressY += selectRowBytes;
  }
}
```

getBlockSelectAreaProc 関数を使用して、計算対象ブロックに対する選択範囲データを取得します。

ブロックが完全に選択範囲外の場合、selectAddress に NULL が入ります。

dstImageAddress と dstAlphaAddress を 1 ピクセル分ずつ行方向に進めていき、それぞれのピクセルに対して処理を行っていきます。一行の処理が終わり次第、処理対象を次の列に移動させます。

ただし、ブロックの中にも選択範囲がある部分とない部分が存在する場合があります。また、選択範囲をぼかしている場合など、薄い選択範囲がある場合があります。

そのため、処理対象のピクセルに対する選択範囲の値 *pSelectAddressX を確認し、その値によって処理を分けます。

*pSelectAddressX == 255 の時は選択範囲が存在します。選択範囲が薄くなってもいないため、選択範囲がない場合と同じ処理を行います。

0 < *pSelectAddressX < 255 の時は薄い選択範囲が存在します。この場合は、フィルタの効果も薄くなるように、別の計算を行います。

*pSelectAddressX == 0 の時は選択範囲外です。そのため、処理を行いません。

一つのブロックの計算処理が終わり次第、ブロックの計算結果を表示に反映させます。

```
TriglavPlugInFilterRunUpdateDestinationOffscreenRect(pRecordSuite,(*pluginServer).hostObject,&blockRect);
```

処理を続行するか、中断するかの判定を行います。

```
TriglavPlugInInt processResult;
if( blockIndex < blockRectCount )
{
  TriglavPlugInFilterRunProcess(pRecordSuite,&processResult,(*pluginServer).hostObject,kTriglavPlugInFilterRunProcessStateContinue);
}
else
{
  TriglavPlugInFilterRunSetProgressDone(pRecordSuite,(*pluginServer).hostObject,blockIndex);
  TriglavPlugInFilterRunProcess(pRecordSuite,&processResult,(*pluginServer).hostObject,kTriglavPlugInFilterRunProcessStateEnd);
}
```

計算途中でユーザーがキャンセルボタン等でフィルタを終了させようとしたり、スライダーでパラメータの値を変えたりする場合があります。

その場合を考慮し、フィルタ処理が途中であっても中断して終了または再計算出来るようにします。

TriglavPlugInFilterRunProcess 関数を実行すると、中断すべきか、処理を続行すべきか、等の情報が変数 processResult に代入されます。詳細は 45 ページ **TriglavPlugInFilterRunProcess** 関数の説明部分を参照してください。

変数 processResult に代入された結果から、プラグインモジュールの処理をどうするかを設定します。

```
if( processResult == kTriglavPlugInFilterRunProcessResultRestart )
{
  restart = true;
}
else if( processResult == kTriglavPlugInFilterRunProcessResultExit )
{
  break;
}
```

変数 processResult に kTriglavPlugInFilterRunProcessResultRestart が代入されていた場合、フィルタの処理を再実行する必要があります。この場合、変数 restart に true を代入し、計算処理の再実行を行うようにしています。

変数 processResult に kTriglavPlugInFilterRunProcessResultExit が代入されていた場合、フィルタ処理を終了する必要があります。この場合、処理のメインループから抜けるようにしています。

変数 processResult の詳細は 45 ページ **TriglavPlugInFilterRunProcess** 関数の説明部分を参照してください。

プラグインホストのコールバック

```
static void TRIGLAV_PLUGIN_CALLBACK TriglavPlugInFilterPropertyCallBack( TriglavPlugInInt* result,
TriglavPlugInPropertyObject propertyObject, const TriglavPlugInInt itemKey, const TriglavPlugInInt notify,
TriglavPlugInPtr data )
{
  (*result) = kTriglavPlugInPropertyCallBackResultNoModify;
  HSVFilterInfo* pFilterInfo = static_cast<HSVFilterInfo*>(data);
  if( pFilterInfo != NULL )
  {
    TriglavPlugInPropertyService* pPropertyService = (*pFilterInfo).pPropertyService;
    if( pPropertyService != NULL )
    {
      if( notify == kTriglavPlugInPropertyCallBackNotifyValueChanged )
```

```
      {
        TriglavPlugInInt value;
        (*(*pFilterInfo).pPropertyService).getIntegerValueProc(&value,propertyObject,itemKey);

        if( itemKey == kItemKeyHue )
        {
          if( (*pFilterInfo).hue != value )
          {
            (*pFilterInfo).hue = value;
            (*result) = kTriglavPlugInPropertyCallBackResultModify;
          }
        }
        else if( itemKey == kItemKeySaturation )
        {
          if( (*pFilterInfo).saturation != value )
          {
            (*pFilterInfo).saturation = value;
            (*result) = kTriglavPlugInPropertyCallBackResultModify;
          }
        }
        else if( itemKey == kItemKeyValue )
        {
          if( (*pFilterInfo).value != value )
          {
            (*pFilterInfo).value = value;
            (*result) = kTriglavPlugInPropertyCallBackResultModify;
          }
        }
      }
    }
  }
}
```

ダイアログの内容が変更された等で、ホスト側の変更があった場合は、ホスト側からコールバック関数が呼び出されます。

パラメータが変更された等の場合は、引数 notify に kTriglavPlugInPropertyCallBackNotifyValueChanged が代入されています。

この時、フィルタの再計算が必要になるため、引数 data の内容を変更し、引数 result に kTriglavPlugInPropertyCallBackResultModify を代入します。

kTriglavPlugInPropertyCallBackResultModify を代入すると、プラグインホスト側でフィルタの処理結果が一旦クリアされ、プラグインモジュール側に再計算をするよう通知がされます。

引数 notify には、定義された定数が代入されています。詳細は 18 ページ「プロパティコールバックの通知の戻り値」を参照してください

■フィルタの実装における注意点

サンプルのフィルタでは、レイヤーの描画データ（オフスクリーン）から直接ピクセルデータを参照する方法を取っています。

この方法は、ピクセルの位置関係を必要としない、ピクセル単位でのフィルタ処理が可能な場合のみ有効な方法です。

ぼかしフィルタなど、ピクセルの位置関係が必要になる場合、この方法は推奨されません。

この場合は、一度ビットマップにデータを転送し、処理を行うようにして下さい。

例として、以下のような流れになります。

1. **createProc** 関数を使用し、新規にビットマップ(データ１とします)を作成します。
2. **getBitmapProc** 関数を使用し、フィルタ処理に必要な領域のデータをデータ１に転送します。

3. createProc 関数を使用し、新規にビットマップ(データ２とします)を作成します。
4. データ１から情報を取得し、フィルタ処理を行います。その結果をデータ２に格納します。
5. setBitmapProc 関数を使用し、描画先のオフスクリーンにデータ２の画像を転送します。

※ビットマップとオフスクリーンで大きなサイズを作成すると、その分メモリの使用量は大きくなります。
そのため、必要最小限のサイズでビットマップを作成することを推奨します。
ビットマップのために必要な関数は、ビットマップサービスとオフスクリーンサービスで提供されています。詳しくは 21～28 ページを参照してください。

基本型

プラグインモジュールで使用する基本型のリストを以下に示します。

これらは TriglavPlugInType.h で宣言されています。

TriglavPlugInBool	符号なし 8 ビットの整数です。 kTriglavPlugInBoolTrue と kTriglavPlugInBoolFalse を代入し、フラグとして使用します。
TriglavPlugInChar	8 ビットの整数です。char 型と同義です。
TriglavPlugInUniChar	16 ビットの整数です。unsigned short 型と同義です。
TriglavPlugInUInt8	符号なし 8 ビットの整数です。unsigned char 型と同義です。
TriglavPlugInInt	32 ビットの整数です。long int 型と同義です。
TriglavPlugInInt64	64 ビットの整数です。long long int 型と同義です。
TriglavPlugInFloat	32 ビットの浮動小数点数です。float 型と同義です。
TriglavPlugInDouble	64 ビットの浮動小数点数です。double 型と同義です。
TriglavPlugInPtr	汎用ポインタです。void 型のポインタと同義です。
TriglavPlugInPoint	TriglavPlugInInt 型の座標を定義する構造体です。
TriglavPlugInSize	TriglavPlugInInt 型の幅と高さを定義する構造体です。
TriglavPlugInRect	四角形の左上隅と右下隅の座標を定義する構造体です。
TriglavPlugInRGBColor	TriglavPlugInUInt8 型の R,G,B 値を定義する構造体です。
TriglavPlugInCMYKColor	TriglavPlugInUInt8 型の C,M,Y,K 値を定義する構造体です。

定義済み定数

プラグインで既に定義済みの主な定数リストを以下に示します。これらは TriglavPlugInDefine.h で宣言されています。

■プラグインの真理値

kTriglavPlugInBoolTrue	1	真
kTriglavPlugInBoolFalse	0	偽

■プラグインメインの戻り値

kTriglavPlugInCallResultSuccess	0	成功
kTriglavPlugInCallResultFailed	-1	失敗

■プラグイン API の戻り値

kTriglavPlugInAPIResultSuccess	0	成功
kTriglavPlugInAPIResultFailed	-1	失敗

■プラグインフィルタ実行の処理の状態

kTriglavPlugInFilterRunProcessStateStart	0x0101	処理の開始
kTriglavPlugInFilterRunProcessStateContinue	0x0102	処理の継続
kTriglavPlugInFilterRunProcessStateEnd	0x0103	処理の終了
kTriglavPlugInFilterRunProcessStateAbort	0x0104	処理の中断

■プラグインフィルタ実行の処理の戻り値

kTriglavPlugInFilterRunProcessResultContinue	0x0201	処理の継続
kTriglavPlugInFilterRunProcessResultRestart	0x0202	処理の再開
kTriglavPlugInFilterRunProcessResultExit	0x0203	処理の終了

■ターゲットレイヤーの種類

kTriglavPlugInFilterTargetKindRasterLayerGrayAlpha	0x0101	ラスターグレーレイヤー(白黒)
kTriglavPlugInFilterTargetKindRasterLayerRGBAlpha	0x0102	ラスターカラーレイヤー(RGB)
kTriglavPlugInFilterTargetKindRasterLayerCMYKAlpha	0x0103	ラスターカラーレイヤー(CMYK)
kTriglavPlugInFilterTargetKindRasterLayerAlpha	0x0104	ラスターグレーレイヤー(黒)、レイヤーマスク、選択範囲レイヤー、クイックマスク
kTriglavPlugInFilterTargetKindRasterLayerBinarizationAlpha	0x0105	ラスターモノクロレイヤー(黒)
kTriglavPlugInFilterTargetKindRasterLayerBinarizationGrayAlpha	0x0106	ラスターモノクロレイヤー(白黒)

■ビットマップのスキャンラインの種類

kTriglavPlugInBitmapScanlineHorizontalLeftTop	0x10	左上から右下への水平スキャン
kTriglavPlugInBitmapScanlineHorizontalRightTop	0x11	右上から左下への水平スキャン
kTriglavPlugInBitmapScanlineHorizontalLeftBottom	0x12	左下から右上への水平スキャン
kTriglavPlugInBitmapScanlineHorizontalRightBottom	0x13	右下から左上への水平スキャン
kTriglavPlugInBitmapScanlineVerticalLeftTop	0x14	左上から右下への垂直スキャン
kTriglavPlugInBitmapScanlineVerticalRightTop	0x15	右上から左下への垂直スキャン
kTriglavPlugInBitmapScanlineVerticalLeftBottom	0x16	左下から右上への垂直スキャン
kTriglavPlugInBitmapScanlineVerticalRightBottom	0x17	右下から左上への垂直スキャン

■オフスクリーンのチャンネルオーダー

kTriglavPlugInOffscreenChannelOrderAlpha	0x01	Alpha チャンネルのみのオフスクリーン
kTriglavPlugInOffscreenChannelOrderGrayAlpha	0x02	Gray と Alpha チャンネルのオフスクリーン
kTriglavPlugInOffscreenChannelOrderRGBAlpha	0x03	RGB と Alpha チャンネルのオフスクリーン
kTriglavPlugInOffscreenChannelOrderCMYKAlpha	0x04	CMYK と Alpha チャンネルのオフスクリーン
kTriglavPlugInOffscreenChannelOrderBinarizationAlpha	0x05	Alpha チャンネルのみのオフスクリーン(モノクロ)
kTriglavPlugInOffscreenChannelOrderBinarizationGrayAlpha	0x06	Gray と Alpha チャンネルのオフスクリーン(モノクロ)
kTriglavPlugInOffscreenChannelOrderSelectArea	0x10	選択範囲オフスクリーン(Alpha チャンネルのみ)
kTriglavPlugInOffscreenChannelOrderPlane	0x20	明確なチャンネルという概念を持たないオフスクリーン

■オフスクリーンのコピーモード

kTriglavPlugInOffscreenCopyModeNormal	0x01	全てのチャンネルをコピー
kTriglavPlugInOffscreenCopyModeImage	0x02	Alpha 以外のチャンネルをコピー
kTriglavPlugInOffscreenCopyModeGray	0x03	Gray と Alpha チャンネルのみコピー
kTriglavPlugInOffscreenCopyModeRed	0x04	Red チャンネルのみコピー

kTriglavPlugInOffscreenCopyModeGreen	0x05	Green チャンネルのみコピー
kTriglavPlugInOffscreenCopyModeBlue	0x06	Blue チャンネルのみコピー
kTriglavPlugInOffscreenCopyModeCyan	0x07	Cyan チャンネルのみコピー
kTriglavPlugInOffscreenCopyModeMagenta	0x08	Magenta チャンネルのみコピー
kTriglavPlugInOffscreenCopyModeYellow	0x09	Yellow チャンネルのみコピー
kTriglavPlugInOffscreenCopyModeKeyPlate	0x10	KeyPlate チャンネルのみコピー
kTriglavPlugInOffscreenCopyModeAlpha	0x11	Alpha チャンネルのみコピー

■プロパティの値の型

kTriglavPlugInPropertyValueTypeVoid	0x00	値が存在しない
kTriglavPlugInPropertyValueTypeBoolean	0x01	真理値型
kTriglavPlugInPropertyValueTypeEnumeration	0x02	列挙型
kTriglavPlugInPropertyValueTypeInteger	0x11	整数型
kTriglavPlugInPropertyValueTypeDecimal	0x12	小数型
kTriglavPlugInPropertyValueTypePoint	0x21	座標型
kTriglavPlugInPropertyValueTypeString	0x31	文字列型

■プロパティの入力の種類

kTriglavPlugInPropertyInputKindDefault は、上記プロパティの型によって入力の UI が変化します(座標型は非対応)。

kTriglavPlugInPropertyInputKindCanvas は、座標型のみに対応しています。

kTriglavPlugInPropertyInputKindHide	0x10	入力 UI なし
kTriglavPlugInPropertyInputKindDefault	0x11	デフォルト
kTriglavPlugInPropertyInputKindPushButton	0x21	プッシュボタン
kTriglavPlugInPropertyInputKindCanvas	0x31	キャンバス

■プロパティの値の種類

kTriglavPlugInPropertyValueKindDefault	0x11	デフォルト
kTriglavPlugInPropertyValueKindPixel	0x21	環境設定の単位を使用

■プロパティの座標のデフォルト値の種類

kTriglavPlugInPropertyPointDefaultValueKindDefault	0x11	デフォルト
kTriglavPlugInPropertyPointDefaultValueKindCanvasLeftTop	0x21	キャンバスの左上
kTriglavPlugInPropertyPointDefaultValueKindCanvasRightTop	0x22	キャンバスの右上
kTriglavPlugInPropertyPointDefaultValueKindCanvasLeftBottom	0x23	キャンバスの左下
kTriglavPlugInPropertyPointDefaultValueKindCanvasRightBottom	0x24	キャンバスの右下
kTriglavPlugInPropertyPointDefaultValueKindCanvasCenter	0x25	キャンバスの中心
kTriglavPlugInPropertyPointDefaultValueKindSelectAreaLeftTop	0x31	選択範囲の左上
kTriglavPlugInPropertyPointDefaultValueKindSelectAreaRightTop	0x32	選択範囲の右上
kTriglavPlugInPropertyPointDefaultValueKindSelectAreaLeftBottom	0x33	選択範囲の左下
kTriglavPlugInPropertyPointDefaultValueKindSelectAreaRightBottom	0x34	選択範囲の右下

kTriglavPlugInPropertyPointDefaultValueKindSelectAreaCenter	0x35	選択範囲の中心

■プロパティの座標値の最小最大値の種類

kTriglavPlugInPropertyPointMinMaxValueKindDefault	0x11	デフォルト
kTriglavPlugInPropertyPointMinMaxValueKindNo	0x21	最小最大値なし

■プロパティコールバックの通知

kTriglavPlugInPropertyCallBackNotifyValueChanged	0x11	値が変更された
kTriglavPlugInPropertyCallBackNotifyButtonPushed	0x21	プッシュボタンが押された

■プロパティコールバックの通知の戻り値

kTriglavPlugInPropertyCallBackResultNoModify	0x01	フィルタ処理の再計算は不必要
kTriglavPlugInPropertyCallBackResultModify	0x02	フィルタ処理の再計算が必要

サービススイート

プラグインモジュール側で使用できる関数を提供します。

サービススイートの関数は、TriglavPlugInInt型の2種類の定数を戻します。

戻り値がkTriglavPlugInAPIResultSuccessの場合、処理が成功したことを示します。

戻り値がkTriglavPlugInAPIResultFailedの場合、処理が失敗したことを示します。

文字列サービス

このサービスは、主に文字列を扱う関数を提供します。

createWithAsciiStringProc()

引数	TriglavPlugInStringObject*	stringObject	out	文字列オブジェクト
	const TriglavPlugInChar*	ascii	in	ASCII文字列
	const TriglavPlugInInt	length	in	ASCII文字列のサイズ

指定されたASCII文字列から、文字列オブジェクトを作成します。失敗した場合は、引数stringObjectにNULLが代入されます。

作成された文字列は、不要になった場合**releaceProc**関数で破棄する必要があります。

createWithUnicodeStringProc()

引数	TriglavPlugInStringObject*	stringObject	out	文字列オブジェクト
	const TriglavPlugInUniChar*	unicode	in	Unicode文字列
	const TriglavPlugInInt	length	in	Unicode文字列のサイズ

指定されたUnicode文字列から、文字列オブジェクトを作成します。失敗した場合は、引数stringObjectにNULLが代入されます。

作成された文字列オブジェクトは、不要になった場合**releaceProc**関数で破棄する必要があります。

createWithLocalCodeStringProc()

引数	TriglavPlugInStringObject*	stringObject	out	文字列オブジェクト
	const TriglavPlugInChar*	localcode	in	ローカルコード(※)文字列
	const TriglavPlugInInt	length	in	ローカルコード文字列のサイズ

指定されたローカルコード文字列から、文字列オブジェクトを作成します。失敗した場合は、引数stringObjectにNULLが代入されます。

作成された文字列オブジェクトは、不要になった場合**releaceProc**関数で破棄する必要があります。

※ローカルコードは、日本語環境ではShift-JISです。

createWithStringIDProc()

引数	TriglavPlugInStringObject*	stringObject	out	文字列オブジェクト
	const TriglavPlugInInt	stringID	in	リソース識別子
	TriglavPlugInHostObject	hostObject	in	ホストオブジェクト

指定されたリソース識別子から、文字列オブジェクトを作成します。失敗した場合は、引数stringObjectにNULLが代入されます。

作成された文字列オブジェクトは、不要になった場合**releaceProc**関数で破棄する必要があります。

リソース識別子の設定方法について

■Windows

Visula StudioのString Tableを使用してください。ここで設定した値を引数stringIDに代入することで、それに対する文字列オブ

ジェクトが作成されます。IDは任意です。

■Mac OS X

Localizable.stringsを使用してください。ここで設定した値を引数stringIDに代入することで、それに対する文字列オブジェクトが作成されます。IDは任意です。

releaseProc()

引数	TriglavPlugInStringObject	stringObject	in	文字列オブジェクト

指定された文字列オブジェクトを破棄します。

getUnicodeCharsProc()

引数	const TriglavPlugInUniChar**	unicode	out	Unicode文字列
	TriglavPlugInStringObject	stringObject	in	文字列オブジェクト

指定された文字列オブジェクトから、Unicode文字列を取得します。

getUnicodeLengthProc()

引数	TriglavPlugInInt*	length	out	Unicode文字列のサイズ
	TriglavPlugInStringObject	stringObject	in	文字列オブジェクト

指定された文字列オブジェクトから、Unicode文字列のサイズを取得します。

getLocalCodeCharsProc()

引数	const TriglavPlugInChar**	localcode	out	ローカルコード文字列
	TriglavPlugInStringObject	stringObject	in	文字列オブジェクト

指定された文字列オブジェクトから、ローカルコード文字列を取得します。

getLocalCodeLengthProc()

引数	TriglavPlugInInt*	length	out	ローカルコード文字列のサイズ
	TriglavPlugInStringObject	stringObject	in	文字列オブジェクト

指定された文字列オブジェクトから、ローカルコード文字列のサイズを取得します。

ビットマップサービス

このサービスは、主にビットマップを扱う関数を提供します。

createProc()

引数	TriglavPlugInBitmapObject*	bitmapObject	out	ビットマップオブジェクト
	const TriglavPlugInInt	width	in	幅(pixel)
	const TriglavPlugInInt	height	in	高さ(pixel)
	const TriglavPlugInInt	depth	in	色深度(byte)
	const TriglavPlugInInt	scanLine	in	スキャンライン

ビットマップオブジェクトを作成します。作成に失敗した場合は、引数 bitmapObject に NULL が代入されます。

作成されたビットマップオブジェクトは、不要になった場合**releaceProc**関数で破棄する必要があります。

スキャンラインには、定義されている定数を使用します。16 ページ「ビットマップのスキャンラインの種類」を参照してください。

releaseProc()

引数	TriglavPlugInBitmapObject	bitmapObject	in	ビットマップオブジェクト

指定されたビットマップオブジェクトを破棄します。

getWidthProc()

引数	TriglavPlugInInt*	width	out	幅(pixel)
	TriglavPlugInBitmapObject	bitmapObject	in	ビットマップオブジェクト

指定されたビットマップオブジェクトの幅を取得します。取得に失敗した場合、引数 width に 0 が代入されます。

getHeightProc()

引数	TriglavPlugInInt*	height	out	高さ(pixel)
	TriglavPlugInBitmapObject	bitmapObject	in	ビットマップオブジェクト

指定されたビットマップオブジェクトの高さを取得します。取得に失敗した場合、引数 height に 0 が代入されます。

getDepthProc()

引数	TriglavPlugInInt*	depth	out	色深度(バイト数)
	TriglavPlugInBitmapObject	bitmapObject	in	ビットマップオブジェクト

指定されたビットマップオブジェクトの色深度を取得します。取得に失敗した場合、引数 depth に 0 が代入されます。

getScanlineProc()

引数	TriglavPlugInInt*	scanline	out	スキャンライン
	TriglavPlugInBitmapObject	bitmapObject	in	ビットマップオブジェクト

指定されたビットマップオブジェクトのスキャンラインを取得します。取得に失敗した場合、引数 scanline に 0 が代入されます。

getAddressProc()

引数	TriglavPlugInPtr*	address	out	アドレス
	TriglavPlugInBitmapObject	bitmapObject	in	ビットマップオブジェクト

	const TriglavPlugInPoint*	pos	in	座標

指定されたビットマップオブジェクトの引数 pos の座標におけるアドレスを取得します。取得に失敗した場合、引数 address に NULL が代入されます。

getRowBytesProc()

引数	TriglavPlugInInt*	rowBytes	out	次の列のピクセルまでのバイト数
	TriglavPlugInBitmapObject	bitmapObject	in	ビットマップオブジェクト

指定されたビットマップオブジェクトの、次の列のピクセルまでのバイト数を取得します。

getPixelBytesProc()

引数	TriglavPlugInInt*	pixelBytes	out	次の行のピクセルまでのバイト数
	TriglavPlugInBitmapObject	bitmapObject	in	ビットマップオブジェクト

指定されたビットマップオブジェクトの、次の行のピクセルまでのバイト数を取得します。

オフスクリーンサービス

このサービスでは、主にオフスクリーンを扱う関数を提供します。

createPlaneProc()

引数	TriglavPlugInOffscreenObject*	offscreenObject	out	オフスクリーンオブジェクト
	const TriglavPlugInInt	width	in	幅(pixel)
	const TriglavPlugInInt	height	in	高さ(pixel)
	const TriglavPlugInInt	depth	in	色深度(byte)

指定されたオフスクリーンオブジェクトに、空のオフスクリーンを作成します。

作成されたオフスクリーンオブジェクトは、不要になった場合**releaceProc**関数で破棄する必要があります。

releaseProc()

引数	TriglavPlugInOffscreenObject	offscreenObject	in	オフスクリーンオブジェクト

指定されたオフスクリーンオブジェクトを破棄します。

getWidthProc()

引数	TriglavPlugInInt*	width	out	幅(pixel)
	TriglavPlugInOffscreenObject	offscreenObject	in	オフスクリーンオブジェクト

指定されたオフスクリーンオブジェクトの幅を取得します。取得に失敗した場合、引数 width に 0 が代入されます。

getHeightProc()

引数	TriglavPlugInInt*	height	out	高さ(pixel)
	TriglavPlugInOffscreenObject	offscreenObject	in	オフスクリーンオブジェクト

指定されたオフスクリーンオブジェクトの高さを取得します。取得に失敗した場合、引数 height に 0 が代入されます。

getRectProc()

引数	TriglavPlugInRect*	rect	out	オフスクリーンのサイズ(pixel)
	TriglavPlugInOffscreenObject	offscreenObject	in	オフスクリーンオブジェクト

指定されたオフスクリーンオブジェクトのサイズを取得します。取得に失敗した場合、引数 rect に 0 に空の矩形が代入されます。

getExtentRectProc()

引数	TriglavPlugInRect*	rect	out	描画されている領域の外接矩形(pixel)
	TriglavPlugInOffscreenObject	offscreenObject	in	オフスクリーンオブジェクト

指定されたオフスクリーンオブジェクトの描画されている領域の外接矩形を取得します。取得に失敗した場合、引数 rect に空の矩形が代入されます。

getChannelOrderProc()

引数	TriglavPlugInInt*	channelOrder	out	チャンネルの並びの情報
	TriglavPlugInOffscreenObject	offscreenObject	in	オフスクリーンオブジェクト

指定されたオフスクリーンオブジェクトのチャンネルの並びの情報を取得します。取得した情報には、定義されている定数が代入

されます。詳しくは 16 ページ「オフスクリーンのチャンネルオーダー」を参照してください。

取得に失敗した場合、引数 channelOrder に 0 が代入されます。

getRGBChannelIndexProc()

引数	TriglavPlugInInt*	redChannelIndex	out	Redのインデックス
	TriglavPlugInInt*	blueChannelIndex	out	Blueのインデックス
	TriglavPlugInInt*	greenChannelIndex	out	Greenのインデックス
	TriglavPlugInOffscreenObject	offscreenObject	in	オフスクリーンオブジェクト

指定されたオフスクリーンオブジェクトの RGB インデックスをそれぞれ取得します。オフスクリーンのチャンネルに RGB が存在しない場合、引数の RGB インデックスすべてに 0 が代入されます。

取得に失敗した場合、引数の RGB インデックスすべてに 0 が代入されます。

getCMYKChannelIndexProc()

引数	TriglavPlugInInt*	cyanChannelIndex	out	Cyanのインデックス
	TriglavPlugInInt*	magentaChannelIndex	out	Magentaのインデックス
	TriglavPlugInInt*	yellowChannelIndex	out	Yellowのインデックス
	TriglavPlugInInt*	keytoneChannelIndex	out	Keytoneのインデックス
	TriglavPlugInOffscreenObject	offscreenObject	in	オフスクリーンオブジェクト

指定されたオフスクリーンオブジェクトの CMYK インデックスをそれぞれ取得します。オフスクリーンのチャンネルに CMYK が存在しない場合、引数の CMYK インデックスすべてに 0 が代入されます。

取得に失敗した場合、引数の CMYK インデックスすべてに 0 が代入されます。

getBlockRectCountProc()

引数	TriglavPlugInInt*	blockRectCount	out	ブロックの数
	TriglavPlugInOffscreenObject	offscreenObject	in	オフスクリーンオブジェクト
	TriglavPlugInRect*	bounds	in	検索領域

指定された領域において、指定されたオフスクリーンオブジェクトのブロックの数を取得します。ただし、**setUseBlankImageProc** 関数にtrueを設定しなかった場合、何も描画されていないブロックはカウントされません。

失敗した場合、引数 blockRectCount に 0 が代入されます。

getBlockRectProc()

引数	TriglavPlugInRect*	blockRect	out	ブロックの領域
	TriglavPlugInInt	blockIndex	in	ブロックのインデックス
	TriglavPlugInOffscreenObject	offscreenObject	in	オフスクリーンオブジェクト
	TriglavPlugInRect*	bounds	in	検索領域

指定された領域において、指定されたオフスクリーンオブジェクトのブロックの領域を取得します。引数 blockIndex は、**getBlockRectCountProc** 関数で取得できる数値未満である必要があります。

失敗した場合、引数 blockRect に空の矩形が代入されます。

getBlockImageProc()

引数	TriglavPlugInPtr*	address	out	アドレス
	TriglavPlugInInt*	rowBytes	out	次の列のピクセルまでのバイト数
	TriglavPlugInInt*	pixelBytes	out	次の行のピクセルまでのバイト数
	TriglavPlugInRect*	blockRect	out	ブロックのサイズ
	TriglavPlugInOffscreenObject	offscreenObject	in	オフスクリーンオブジェクト
	TriglavPlugInPoint*	pos	in	座標

指定されたオフスクリーンオブジェクトにおいて、指定された座標の画像のアドレスと、その座標が含まれるブロックの、次の列のピクセルまでのバイト数、次の行のピクセルまでのバイト数、ブロックのサイズを取得します。
失敗した場合、引数 address に NULL、引数 rowBytes に 0、引数 pixelBytes に 0、引数 blockRect に空の矩形が代入されます。
※一度この関数で取得したアドレスは、再度この関数でアドレスを取得するまで有効です。この場合、以前取得したアドレスへのアクセスは保証されません。

getBlockAlphaProc()

引数	TriglavPlugInPtr*	address	out	アドレス
	TriglavPlugInInt*	rowBytes	out	次の列のピクセルまでのバイト数
	TriglavPlugInInt*	pixelBytes	out	次の行のピクセルまでのバイト数
	TriglavPlugInRect*	blockRect	out	ブロックのサイズ
	TriglavPlugInOffscreenObject	offscreenObject	in	オフスクリーンオブジェクト
	TriglavPlugInPoint*	pos	in	座標

指定されたオフスクリーンオブジェクトにおいて、指定された座標のアルファのアドレスと、その座標が含まれるブロックの、次の列のピクセルまでのバイト数、次の行のピクセルまでのバイト数、ブロックのサイズを取得します。
失敗した場合、引数 address に NULL、引数 rowBytes に 0、引数 pixelBytes に 0、引数 blockRect に空の矩形が代入されます。
※一度この関数で取得したアドレスは、再度この関数でアドレスを取得するまで有効です。この場合、以前取得したアドレスへのアクセスは保証されません。

getBlockSelectAreaProc()

引数	TriglavPlugInPtr*	address	out	アドレス
	TriglavPlugInInt*	rowBytes	out	次の列のピクセルまでのバイト数
	TriglavPlugInInt*	pixelBytes	out	次の行のピクセルまでのバイト数
	TriglavPlugInRect*	blockRect	out	ブロックのサイズ
	TriglavPlugInOffscreenObject	offscreenObject	in	オフスクリーンオブジェクト
	TriglavPlugInPoint*	pos	in	座標

指定されたオフスクリーンオブジェクトにおいて、指定された座標の選択範囲画像のアドレスと、その座標が含まれるブロックの、次の列のピクセルまでのバイト数、次の行のピクセルまでのバイト数、ブロックのサイズを取得します。
失敗した場合、引数 address に NULL、引数 rowBytes に 0、引数 pixelBytes に 0、引数 blockRect に空の矩形が代入されます。
※一度この関数で取得したアドレスは、再度この関数でアドレスを取得するまで有効です。この場合、以前取得したアドレスへのアクセスは保証されません。

getBlockPlaneProc()

引数	TriglavPlugInPtr*	address	out	アドレス

	TriglavPlugInInt*	rowBytes	out	次の列のピクセルまでのバイト数
	TriglavPlugInInt*	pixelBytes	out	次の行のピクセルまでのバイト数
	TriglavPlugInRect*	blockRect	out	ブロックのサイズ
	TriglavPlugInOffscreenObject	offscreenObject	in	オフスクリーンオブジェクト
	TriglavPlugInPoint*	pos	in	座標

指定されたオフスクリーンオブジェクトにおいて、指定された座標のプレーンオフスクリーンのアドレスと、その座標が含まれるブロックの、次の列のピクセルまでのバイト数、次の行のピクセルまでのバイト数、ブロックのサイズを取得します。
失敗した場合、引数 address に NULL、引数 rowBytes に 0、引数 pixelBytes に 0、引数 blockRect に空の矩形が代入されます。
※一度この関数で取得したアドレスは、再度この関数でアドレスを取得するまで有効です。この場合、以前取得したアドレスへのアクセスは保証されません。

getTileWidthProc()

引数	TriglavPlugInInt*	tileWidth	out	タイルの幅
	TriglavPlugInOffscreenObject	offscreenObject	in	オフスクリーンオブジェクト

指定されたオフスクリーンオブジェクトの、ホストで設定されたタイルの幅を取得します。

getTileHeightProc()

引数	TriglavPlugInInt*	tileHeight	out	タイルの高さ
	TriglavPlugInOffscreenObject	offscreenObject	in	オフスクリーンオブジェクト

指定されたオフスクリーンオブジェクトの、ホストで設定されたタイルの高さを取得します。

getBitmapProc()

引数	TriglavPlugInBitmapObject	bitmapObject	out	ビットマップオブジェクト
	const TriglavPlugInPoint*	bitmapPos	in	ビットマップの左上座標
	TriglavPlugInOffscreenObject	offscreenObject	in	オフスクリーンオブジェクト
	const TriglavPlugInPoint*	offscreenPos	in	オフスクリーン左上座標
	const TriglavPlugInInt	copyWidth	in	コピーする幅
	const TriglavPlugInInt	copyHeight	in	コピーする高さ
	const TriglavPlugInInt	copyMode	in	コピーモード

指定されたオフスクリーンから、ビットマップ画像を作成・取得します。失敗した場合、引数 bitmapObject に正しいデータが代入されません。
コピーモードには定義された定数を使用します。詳しくは 16 ページ「オフスクリーンのコピーモード」を参照してください。
指定されたコピーモードに対し、オフスクリーンに対象のチャンネルが存在しない、またはコピー先の bitmap の depth が不十分である場合、コピーが正常に行われません。

setBitmapProc()

引数	TriglavPlugInOffscreenObject	offscreenObject	out	オフスクリーンオブジェクト
	const TriglavPlugInPoint*	offscreenPos	out	オフスクリーンの左上座標
	TriglavPlugInBitmapObject	bitmapObject	in	ビットマップオブジェクト
	const TriglavPlugInPoint*	bitmapPos	in	ビットマップの左上座標

	const TriglavPlugInInt	copyWidth	in	コピーする幅
	const TriglavPlugInInt	copyHeight	in	コピーする高さ
	const TriglavPlugInInt	copyMode	in	コピーモード

指定されたオフスクリーンに、ビットマップ画像を設定します。失敗した場合、引数 offscreenObject に正しいデータが代入されません。

コピーモードには定義された定数を使用します。詳しくは 16 ページ「オフスクリーンのコピーモード」を参照してください。

指定されたコピーモードに対し、オフスクリーンに対象のチャンネルが存在しない、またはコピー先のビットマップの depth が不十分である場合、コピーが正常に行われません。

オフスクリーンサービス 2

このサービスでは、主にオフスクリーンを扱う関数を提供します。

getBitmapNormalAlphaChannelIndexProc()

引数	TriglavPlugInInt*	alphaChannelIndex	out	Alphaチャンネルのインデックス
	TriglavPlugInOffscreenObject	offscreenObject	in	オフスクリーンオブジェクト

コピーモードが kTriglavPlugInOffscreenCopyModeNormal である場合の、Alpha チャンネルのインデックスを取得します。

プロパティサービス

このサービスでは、主にフィルタで使用するパラメータを扱う関数を提供します。

createProc()

引数	TriglavPlugInPropertyObject*	propertyObject	out	プロパティオブジェクト

プロパティオブジェクトを作成します。作成に失敗した場合、引数 propertyObject に正しい情報が代入されません。

作成されたプロパティオブジェクトは、不要になった場合**releaceProc**関数で破棄する必要があります。

releaceProc()

引数	TriglavPlugInPropertyObject*	propertyObject	out	プロパティオブジェクト

指定されたプロパティオブジェクトを破棄します。

addItemProc()

引数	TriglavPlugInPropertyObject*	propertyObject	in	プロパティオブジェクト
	const TriglavPlugInInt	itemKey	in	アイテムキー
	const TriglavPlugInInt	valueType	in	値の型
	const TriglavPlugInInt	valueKind	in	値の種類
	const TriglavPlugInInt	inputKind	in	入力の種類
	const TriglavPlugInStringObject	caption	in	キャプション
	const TriglavPlugInChar	accessKey	in	アクセスキー

指定されたプロパティオブジェクトに、アイテムを追加します。失敗した場合、引数 propatyObject にアイテムが追加されません。

引数 itemkey に代入される値は、一つのプロパティオブジェクト内で一意である必要があります。

値の型、値の種類、入力の種類には定義された定数を使用します。詳しくは17ページ「プロパティの値の型」「プロパティの入力の種類」「プロパティの値の種類」を参照してください。

setBooleanValueProc()

引数	TriglavPlugInPropertyObject*	propertyObject	in	プロパティオブジェクト
	const TriglavPlugInInt	itemKey	in	アイテムキー
	const TriglavPlugInBool	value	in	真理値

指定されたプロパティオブジェクトの、指定されたアイテムキーのアイテムに対して真理値を設定します。失敗した場合、引数 propertyObject に正しい情報が代入されません。

getBooleanValueProc()

引数	TriglavPlugInBool*	value	out	真理値
	TriglavPlugInPropertyObject*	propertyObject	in	プロパティオブジェクト
	const TriglavPlugInInt	itemKey	in	アイテムキー

指定されたプロパティオブジェクトの、指定されたアイテムキーのアイテムに対する真理値を取得します。

setBooleanValueProc()であらかじめ真理値を設定する必要があります。

setBooleanDefaultValueProc()

引数	TriglavPlugInPropertyObject*	propertyObject	in	プロパティオブジェクト
	const TriglavPlugInInt	itemKey	in	アイテムキー
	const TriglavPlugInBool	defaultValue	in	デフォルトの真理値

指定されたプロパティオブジェクトの、指定されたアイテムキーのアイテムに対してデフォルトの真理値を設定します。失敗した場合、引数 propertyObject に正しい情報が代入されません。

getBooleanDefaultValueProc()

引数	TriglavPlugInBool*	defaultValue	out	デフォルトの真理値
	TriglavPlugInPropertyObject*	propertyObject	in	プロパティオブジェクト
	const TriglavPlugInInt	itemKey	in	アイテムキー

指定されたプロパティオブジェクトの、指定されたアイテムキーのアイテムに対するデフォルトの真理値を取得します。

setBooleanDefaultValueProc 関数であらかじめデフォルトの真理値を設定する必要があります。

setIntegerValueProc()

引数	TriglavPlugInPropertyObject*	propertyObject	in	プロパティオブジェクト
	const TriglavPlugInInt	itemKey	in	アイテムキー
	const TriglavPlugInInt	value	in	整数値

指定されたプロパティオブジェクトの、指定されたアイテムキーのアイテムに対して整数値を設定します。失敗した場合、引数 propertyObject に正しい情報が代入されません。

getIntegerValueProc()

引数	TriglavPlugInInt*	value	out	整数値
	TriglavPlugInPropertyObject*	propertyObject	in	プロパティオブジェクト
	const TriglavPlugInInt	itemKey	in	アイテムキー

指定されたプロパティオブジェクトの、指定されたアイテムキーのアイテムに対する整数値を取得します。

setIntegerValueProc 関数であらかじめ整数値を設定する必要があります。

setIntegerDefaultValueProc()

引数	TriglavPlugInPropertyObject*	propertyObject	in	プロパティオブジェクト
	const TriglavPlugInInt	itemKey	in	アイテムキー
	const TriglavPlugInInt	defaultValue	in	デフォルトの整数値

指定されたプロパティオブジェクトの、指定されたアイテムキーのアイテムに対してデフォルトの整数値を設定します。失敗した場合、引数 propertyObject に正しい情報が代入されません。

getIntegerDefaultValueProc()

引数	TriglavPlugInInt*	defaultValue	out	デフォルトの整数値
	TriglavPlugInPropertyObject*	propertyObject	in	プロパティオブジェクト
	const TriglavPlugInInt	itemKey	in	アイテムキー

指定されたプロパティオブジェクトの、指定されたアイテムキーのアイテムに対するデフォルトの整数値を取得します。

setIntegerDefaultValueProc 関数であらかじめデフォルトの整数値を設定する必要があります。

setIntegerMinValueProc()

引数	TriglavPlugInPropertyObject*	propertyObject	in	プロパティオブジェクト
	const TriglavPlugInInt	itemKey	in	アイテムキー
	const TriglavPlugInInt	minValue	in	最小の整数値

指定されたプロパティオブジェクトの、指定されたアイテムキーのアイテムに対して最小の整数値を設定します。失敗した場合、引数 propertyObject に正しい情報が代入されません。

getIntegerMinValueProc()

引数	TriglavPlugInInt*	minValue	out	最小の整数値
	TriglavPlugInPropertyObject*	propertyObject	in	プロパティオブジェクト
	const TriglavPlugInInt	itemKey	in	アイテムキー

指定されたプロパティオブジェクトの、指定されたアイテムキーのアイテムに対する最小の整数値を取得します。

setIntegerMinValueProc 関数であらかじめ最小の整数値を設定する必要があります。

setIntegerMaxValueProc()

引数	TriglavPlugInPropertyObject*	propertyObject	in	プロパティオブジェクト
	const TriglavPlugInInt	itemKey	in	アイテムキー
	const TriglavPlugInInt	maxValue	in	最大の整数値

指定されたプロパティオブジェクトの、指定されたアイテムキーのアイテムに対して最大の整数値を設定します。失敗した場合、引数 propertyObject に正しい情報が代入されません。

getIntegerMaxValueProc()

引数	TriglavPlugInInt*	maxValue	out	最大の整数値
	TriglavPlugInPropertyObject*	propertyObject	in	プロパティオブジェクト
	const TriglavPlugInInt	itemKey	in	アイテムキー

指定されたプロパティオブジェクトの、指定されたアイテムキーのアイテムに対する最大の整数値を取得します。

getIntegerMaxValueProc 関数であらかじめ最大の整数値を設定する必要があります。

setDecimalValueProc()

引数	TriglavPlugInPropertyObject*	propertyObject	in	プロパティオブジェクト
	const TriglavPlugInInt	itemKey	in	アイテムキー
	const TriglavPlugInDouble	value	in	小数値

指定されたプロパティオブジェクトの、指定されたアイテムキーのアイテムに対して小数値を設定します。失敗した場合、引数 propertyObject に正しい情報が代入されません。

getDecimalValueProc()

引数	TriglavPlugInDouble*	value	out	小数値
	TriglavPlugInPropertyObject*	propertyObject	in	プロパティオブジェクト
	const TriglavPlugInInt	itemKey	in	アイテムキー

指定されたプロパティオブジェクトの、指定されたアイテムキーのアイテムに対する小数値を取得します。

setDecimalValueProc 関数であらかじめ小数値を設定する必要があります。

setDecimalDefaultValueProc()

引数	TriglavPlugInPropertyObject*	propertyObject	in	プロパティオブジェクト
	const TriglavPlugInInt	itemKey	in	アイテムキー
	const TriglavPlugInDouble	defaultValue	in	デフォルトの小数値

指定されたプロパティオブジェクトの、指定されたアイテムキーのアイテムに対してデフォルトの小数値を設定します。失敗した場合、引数 propertyObject に正しい情報が代入されません。

getDecimalDefaultValueProc()

引数	TriglavPlugInDouble*	defaultValue	out	デフォルトの小数値
	TriglavPlugInPropertyObject*	propertyObject	in	プロパティオブジェクト
	const TriglavPlugInInt	itemKey	in	アイテムキー

指定されたプロパティオブジェクトの、指定されたアイテムキーのアイテムに対するデフォルトの小数値を取得します。

setDecimalDefaultValueProc 関数であらかじめデフォルトの小数値を設定する必要があります。

setDecimalMinValueProc()

引数	TriglavPlugInPropertyObject*	propertyObject	in	プロパティオブジェクト
	const TriglavPlugInInt	itemKey	in	アイテムキー
	const TriglavPlugInDouble	minValue	in	最小の小数値

指定されたプロパティオブジェクトの、指定されたアイテムキーのアイテムに対して最小の小数値を設定します。失敗した場合、引数 propertyObject に正しい情報が代入されません。

getDecimalMinValueProc()

引数	TriglavPlugInDouble*	minValue	out	最小の小数値
	TriglavPlugInPropertyObject*	propertyObject	in	プロパティオブジェクト
	const TriglavPlugInInt	itemKey	in	アイテムキー

指定されたプロパティオブジェクトの、指定されたアイテムキーのアイテムに対する最小の小数値を取得します。

setDecimalMinValueProc 関数であらかじめ最小の小数値を設定する必要があります。

setDecimalMaxValueProc()

引数	TriglavPlugInPropertyObject*	propertyObject	in	プロパティオブジェクト
	const TriglavPlugInInt	itemKey	in	アイテムキー
	const TriglavPlugInDouble	maxValue	in	最大の小数値

指定されたプロパティオブジェクトの、指定されたアイテムキーのアイテムに対して最大の小数値を設定します。失敗した場合、引数 propertyObject に正しい情報が代入されません。

getDecimalMaxValueProc()

引数	TriglavPlugInDouble*	maxValue	out	最大の小数値

	TriglavPlugInPropertyObject*	propertyObject	in	プロパティオブジェクト
	const TriglavPlugInInt	itemKey	in	アイテムキー

指定されたプロパティオブジェクトの、指定されたアイテムキーのアイテムに対する最大の小数値を取得します。

getDecimalMaxValueProc 関数であらかじめ最大の小数値を設定する必要があります。

プロパティサービス 2

このサービスでは、主にフィルタで使用するパラメータを扱う関数を提供します。

setItemStoreValueProc()

引数	TriglavPlugInPropertyObject	propertyObject	in	プロパティオブジェクト
	const TriglavPlugInInt	itemKey	in	アイテムキー
	const TriglavPlugInBool	storeValue	in	アイテムの値を記憶するか

指定されたプロパティオブジェクトの、指定されたアイテムキーのアイテムの値を記憶するかを設定します。

フィルタ実行時の値を次回の初期値に設定したい場合に使用します。

setPointValueProc()

引数	TriglavPlugInPropertyObject	propertyObject	in	プロパティオブジェクト
	const TriglavPlugInInt	itemKey	in	アイテムキー
	const TriglavPlugInPoint*	value	in	座標

指定されたプロパティオブジェクトの、指定されたアイテムキーのアイテムに対して座標を設定します。

getPointValueProc()

引数	TriglavPlugInPoint*	value	out	座標
	TriglavPlugInPropertyObject	propertyObject	in	プロパティオブジェクト
	const TriglavPlugInInt	itemKey	in	アイテムキー

指定されたプロパティオブジェクトの、指定されたアイテムキーのアイテムに対する座標を取得します。

setPointDefaultValueKindProc()

引数	TriglavPlugInPropertyObject	propertyObject	in	プロパティオブジェクト
	const TriglavPlugInInt	itemKey	in	アイテムキー
	const TriglavPlugInInt	defaultValueKind	in	デフォルトの座標の種類

指定されたプロパティオブジェクトの、指定されたアイテムキーのアイテムに対してデフォルトの座標の種類を設定します。

デフォルトの座標の種類には、定義されている定数を使用します。17 ページ「プロパティの座標のデフォルト値の種類」を参照してください。

getPointDefaultValueKindProc()

引数	TriglavPlugInInt*	defaultValueKind	out	デフォルトの座標の種類
	TriglavPlugInPropertyObject	propertyObject	in	プロパティオブジェクト
	const TriglavPlugInInt	itemKey	in	アイテムキー

指定されたプロパティオブジェクトの、指定されたアイテムキーのアイテムに対するデフォルトの座標の種類を取得します。

取得した座標の種類には、定義されている定数が代入されます。17 ページ「プロパティの座標のデフォルト値の種類」を参照してください。

setPointDefaultValueProc()

引数	TriglavPlugInPropertyObject	propertyObject	in	プロパティオブジェクト

	const TriglavPlugInInt	itemKey	in	アイテムキー
	const TriglavPlugInPoint*	defaultValue	in	デフォルトの座標

指定されたプロパティオブジェクトの、指定されたアイテムキーのアイテムに対してデフォルトの座標を設定します。

getPointDefaultValueProc()

引数	TriglavPlugInPoint*	defaultValue	out	デフォルトの座標
	TriglavPlugInPropertyObject	propertyObject	in	プロパティオブジェクト
	const TriglavPlugInInt	itemKey	in	アイテムキー

指定されたプロパティオブジェクトの、指定されたアイテムキーのアイテムに対するデフォルトの座標を取得します。

setPointMinMaxValueKindProc()

引数	TriglavPlugInPropertyObject	propertyObject	in	プロパティオブジェクト
	const TriglavPlugInInt	itemKey	in	アイテムキー
	const TriglavPlugInInt	minMaxValueKind	in	座標値の最小最大値の種類

指定されたプロパティオブジェクトの、指定されたアイテムキーのアイテムに対して最大最小の座標の種類を設定します。
座標値の最小最大値の種類には、定義されている定数を使用します。18 ページ「プロパティの座標値の最小最大値の種類」を参照してください。

getPointMinMaxValueKindProc()

引数	TriglavPlugInInt*	minMaxValueKind	out	座標値の最小最大値の種類
	TriglavPlugInPropertyObject	propertyObject	in	プロパティオブジェクト
	const TriglavPlugInInt	itemKey	in	アイテムキー

指定されたプロパティオブジェクトの、指定されたアイテムキーのアイテムに対する最大最小の座標の種類を取得します。
取得した座標値の最小最大値の種類には、定義されている定数が代入されます。18 ページ「プロパティの座標値の最小最大値の種類」を参照してください。

setPointMinValueProc()

引数	TriglavPlugInPropertyObject	propertyObject	in	プロパティオブジェクト
	const TriglavPlugInInt	itemKey	in	アイテムキー
	const TriglavPlugInPoint*	minValue	in	最小の座標

指定されたプロパティオブジェクトの、指定されたアイテムキーのアイテムに対して最小の座標を設定します。

getPointMinValueProc()

引数	TriglavPlugInPoint*	minValue	out	最小の座標
	TriglavPlugInPropertyObject	propertyObject	in	プロパティオブジェクト
	const TriglavPlugInInt	itemKey	in	アイテムキー

指定されたプロパティオブジェクトの、指定されたアイテムキーのアイテムに対する最小の座標を取得します。

setPointMaxValueProc()

引数	TriglavPlugInPropertyObject	propertyObject	in	プロパティオブジェクト

	const TriglavPlugInInt	itemKey	in	アイテムキー
	const TriglavPlugInPoint*	maxValue	in	最大の座標

指定されたプロパティオブジェクトの、指定されたアイテムキーのアイテムに対して最大の座標を設定します。

getPointMaxValueProc()

引数	TriglavPlugInPoint*	maxValue	out	最大の座標
	TriglavPlugInPropertyObject	propertyObject	in	プロパティオブジェクト
	const TriglavPlugInInt	itemKey	in	アイテムキー

指定されたプロパティオブジェクトの、指定されたアイテムキーのアイテムに対する最大の座標を取得します。

setEnumerationValueProc()

引数	TriglavPlugInPropertyObject	propertyObject	in	プロパティオブジェクト
	const TriglavPlugInInt	itemKey	in	アイテムキー
	const TriglavPlugInInt	value	in	カレントの列挙値

指定されたプロパティオブジェクトの、指定されたアイテムキーのアイテムに対してカレントの列挙値を設定します。

value には、**addEnumerationItemProc** 関数で設定した数値を設定します。

getEnumerationValueProc()

引数	TriglavPlugInInt*	value	out	カレントの列挙値
	TriglavPlugInPropertyObject	propertyObject	in	プロパティオブジェクト
	const TriglavPlugInInt	itemKey	in	アイテムキー

指定されたプロパティオブジェクトの、指定されたアイテムキーのアイテムに対するカレントの列挙値を取得します。

value には、**addEnumerationItemProc** 関数で設定した列挙値が代入されます。

setEnumerationDefaultValueProc()

引数	TriglavPlugInPropertyObject	propertyObject	in	プロパティオブジェクト
	const TriglavPlugInInt	itemKey	in	アイテムキー
	const TriglavPlugInInt	value	in	デフォルトの列挙値

指定されたプロパティオブジェクトの、指定されたアイテムキーのアイテムに対してデフォルトの列挙値を設定します。

value には、**addEnumerationItemProc** 関数で設定した列挙値を設定します。

getEnumerationDefaultValueProc()

引数	TriglavPlugInInt*	value	out	デフォルトの列挙値
	TriglavPlugInPropertyObject	propertyObject	in	プロパティオブジェクト
	const TriglavPlugInInt	itemKey	in	アイテムキー

指定されたプロパティオブジェクトの、指定されたアイテムキーのアイテムに対するデフォルトの列挙値を取得します。

value には、**addEnumerationItemProc** 関数で設定した列挙値が代入されます。

addEnumerationItemProc()

引数	TriglavPlugInPropertyObject	propertyObject	in	プロパティオブジェクト

	const TriglavPlugInInt	itemKey	in	アイテムキー
	const TriglavPlugInInt	value	in	列挙値
	TriglavPlugInStringObject	caption	in	文字列
	const TriglavPlugInChar	accessKey	in	アクセスキー

指定されたプロパティオブジェクトの、指定されたアイテムキーのアイテムに対して列挙するアイテムを追加します。

setStringValueProc()

引数	TriglavPlugInPropertyObject	propertyObject	in	プロパティオブジェクト
	const TriglavPlugInInt	itemKey	in	アイテムキー
	TriglavPlugInStringObject	value	in	文字列

指定されたプロパティオブジェクトの、指定されたアイテムキーのアイテムに対して文字列を設定します。

getStringValueProc()

引数	TriglavPlugInStringObject*	value	out	文字列
	TriglavPlugInPropertyObject	propertyObject	in	プロパティオブジェクト
	const TriglavPlugInInt	itemKey	in	アイテムキー

指定されたプロパティオブジェクトの、指定されたアイテムキーのアイテムに対する文字列を取得します。

setStringDefaultValueProc()

引数	TriglavPlugInPropertyObject	propertyObject	in	プロパティオブジェクト
	const TriglavPlugInInt	itemKey	in	アイテムキー
	TriglavPlugInStringObject	defaultValue	in	デフォルトの文字列

指定されたプロパティオブジェクトの、指定されたアイテムキーのアイテムに対してデフォルトの文字列を設定します。

getStringDefaultValueProc()

引数	TriglavPlugInStringObject*	defaultValue	out	デフォルトの文字列
	TriglavPlugInPropertyObject	propertyObject	in	プロパティオブジェクト
	const TriglavPlugInInt	itemKey	in	アイテムキー

指定されたプロパティオブジェクトの、指定されたアイテムキーのアイテムに対するデフォルトの文字列を取得します。

setStringMaxLengthProc()

引数	TriglavPlugInPropertyObject	propertyObject	in	プロパティオブジェクト
	const TriglavPlugInInt	itemKey	in	アイテムキー
	const TriglavPlugInInt	maxLength	in	文字列の最大文字数

指定されたプロパティオブジェクトの、指定されたアイテムキーのアイテムに対して文字列の最大文字数を設定します。

getStringMaxLengthProc()

引数	TriglavPlugInInt*	maxLength	out	文字列の最大文字数
	TriglavPlugInPropertyObject	propertyObject	in	プロパティオブジェクト
	const TriglavPlugInInt	itemKey	in	アイテムキー

指定されたプロパティオブジェクトの、指定されたアイテムキーのアイテムに対する文字列の最大文字数を取得します。

レコードスイート

プラグインモジュール側で、特定の条件下で使用できる関数を提供します。

レコードスイートの関数は、TriglavPlugInInt型の2種類の定数を戻します。

戻り値がkTriglavPlugInAPIResultSuccessの場合、処理が成功したことを示します。

戻り値がkTriglavPlugInAPIResultFailedの場合、処理が失敗したことを示します。

モジュール初期化レコード

getHostVersionProc()

引数	TriglavPlugInInt*	hostVersion	out	ホストバージョン
	TriglavPlugInHostObject	hostObject	in	ホストオブジェクト

指定されたホストオブジェクトのホストバージョンを取得します。失敗した場合、引数 hostVersion に 0 が代入されます。

setModuleIDProc()

引数	TriglavPlugInHostObject	hostObject	in	ホストオブジェクト
	TriglavPlugInStringObject	moduleID	in	モジュールID

指定されたホストオブジェクトにモジュール ID を設定します。失敗した場合、引数 hostObject にモジュール ID が設定されません。

モジュール ID は一意である必要があります。GUID を使用してください。

setModuleKindProc()

引数	TriglavPlugInHostObject	hostObject	in	ホストオブジェクト
	const TriglavPlugInInt	moduleKind	in	モジュールの種類

指定されたホストオブジェクトにモジュールの種類を設定します。失敗した場合、引数hostObjectにモジュールの種類が設定されません。

フィルタプラグインを作成する場合は、引数 moduleKind に kTriglavPlugInModuleKindFilter を設定してください。

フィルタ初期化レコード

TriglavPlugInGetFilterInitializeRecord()

引数	TriglavPlugInRecordSuite	record	in	レコードスイート

フィルタ初期化レコードを取得します。

TriglavPlugInFilterInitializeSetFilterCategoryName()

引数	TriglavPlugInRecordSuite	record	in	レコードスイート
	TriglavPlugInHostObject	hostObject	in	ホストオブジェクト
	TriglavPlugInStringObject	filterCategoryName	in	フィルタカテゴリ名
	const TriglavPlugInChar	accessKey	in	アクセスキー

指定されたホストオブジェクトにフィルタカテゴリ名と、アクセスキーを設定します。失敗した場合、引数hostObjectにフィルタカテゴリ名と、アクセスキーが設定されません。

TriglavPlugInFilterInitializeSetFilterName()

引数	TriglavPlugInRecordSuite	record	in	レコードスイート
	TriglavPlugInHostObject	hostObject	in	ホストオブジェクト
	TriglavPlugInStringObject	filterName	in	フィルタ名
	const TriglavPlugInChar	accessKey	in	アクセスキー

指定されたホストオブジェクトにフィルタ名と、アクセスキーを設定します。失敗した場合、引数hostObjectにフィルタ名と、アクセスキーが設定されません。

TriglavPlugInFilterInitializeSetCanPreview()

引数	TriglavPlugInRecordSuite	record	in	レコードスイート
	TriglavPlugInHostObject	hostObject	in	ホストオブジェクト
	const TriglavPlugInBool	canPreview	in	プレビュー表示を許可するかどうか

指定されたホストオブジェクトにプレビュー表示を許可するかどうかを設定します。失敗した場合、引数hostObjectにこの設定がされません。

TriglavPlugInFilterInitializeSetUseBlankImage()

引数	TriglavPlugInRecordSuite	record	in	レコードスイート
	TriglavPlugInHostObject	hostObject	in	ホストオブジェクト
	const TriglavPlugInBool	useBlankImage	in	何も描画されていない画像に対してもフィルタを実行するかどうか

指定されたホストオブジェクトに、何も描画されていない画像に対してもフィルタを実行するかどうかを設定します。失敗した場合、引数hostObjectにこの設定がされません。

TriglavPlugInFilterInitializeSetTargetKinds()

引数	TriglavPlugInRecordSuite	record	in	レコードスイート
	TriglavPlugInHostObject	hostObject	in	ホストオブジェクト

	const TriglavPlugInInt*	targetKinds	in	ターゲットレイヤーの種類の配列へのポインタ
	const TriglavPlugInInt	targetKindCount	in	ターゲットレイヤーの種類の数

指定されたホストオブジェクトにターゲットレイヤーの種類を設定します。複数のターゲットレイヤーを指定することが出来ます。失敗した場合、ターゲットレイヤーの種類が設定されません。

ターゲットレイヤーの種類には定義された定数を使用します。詳しくは 16 ページ「ターゲットレイヤーの種類」を参照してください。

TriglavPlugInFilterInitializeSetProperty()

引数	TriglavPlugInRecordSuite	record	in	レコードスイート
	TriglavPlugInHostObject	hostObject	in	ホストオブジェクト
	TriglavPlugInPropertyObject	propertyObject	in	プロパティオブジェクト

指定されたホストオブジェクトにプロパティオブジェクトを設定します。失敗した場合、引数hostObjectにプロパティオブジェクトが設定されません。

TriglavPlugInFilterInitializeSetPropertyCallBack()

引数	TriglavPlugInRecordSuite	record	in	レコードスイート
	TriglavPlugInHostObject	hostObject	in	ホストオブジェクト
	TriglavPlugInPropertyCallBackProc	propertyCallBackProc	in	プロパティコールバック
	TriglavPlugInPtr	data	in	データ

指定されたホストオブジェクトにプロパティコールバック関数へのポインタと、プロパティコールバックで使用するデータへのポインタを設定します。失敗した場合、引数hostObjectにこの設定がされません。

フィルタ実行レコード

TriglavPlugInGetFilterRunRecord()

引数	TriglavPlugInRecordSuite	record	in	レコードスイート

フィルタ実行レコードを取得します。

TriglavPlugInFilterRunGetProperty()

引数	TriglavPlugInRecordSuite	record	in	レコードスイート
	TriglavPlugInPropertyObject*	propertyObject	in	プロパティオブジェクト
	TriglavPlugInHostObject	hostObject	in	ホストオブジェクト

指定されたホストオブジェクトからプロパティオブジェクトを取得します。失敗した場合、引数 propertyObject に NULL が代入されます。

TriglavPlugInFilterRunGetCanvasWidth()

引数	TriglavPlugInRecordSuite	record	in	レコードスイート
	TriglavPlugInInt*	width	out	キャンバスの幅
	TriglavPlugInHostObject	hostObject	in	ホストオブジェクト

指定されたホストオブジェクトからキャンバスの幅を取得します。失敗した場合、引数 width に 0 が代入されます。

TriglavPlugInFilterRunGetCanvasHeight()

引数	TriglavPlugInRecordSuite	record	in	レコードスイート
	TriglavPlugInInt*	height	out	キャンバスの高さ
	TriglavPlugInHostObject	hostObject	in	ホストオブジェクト

指定されたホストオブジェクトからキャンバスの高さを取得します。失敗した場合、引数 height に 0 が代入されます。

TriglavPlugInFilterRunGetCanvasResolution()

引数	TriglavPlugInRecordSuite	record	in	レコードスイート
	TriglavPlugInInt*	resolution	out	キャンバスの解像度(dpi)
	TriglavPlugInHostObject	hostObject	in	ホストオブジェクト

指定されたホストオブジェクトからキャンバスの解像度を取得します。失敗した場合、引数 resolution に 0 が代入されます。

TriglavPlugInFilterRunGetLayerOrigin()

引数	TriglavPlugInRecordSuite	record	in	レコードスイート
	TriglavPlugInPoint*	layerOrigin	out	編集レイヤーの原点
	TriglavPlugInHostObject	hostObject	in	ホストオブジェクト

指定されたホストオブジェクトから編集レイヤーの原点を取得します。失敗した場合、引数 layerOrigin に座標(0,0)が代入されます。

TriglavPlugInFilterRunIsLayerMaskSelected()

引数	TriglavPlugInRecordSuite	record	in	レコードスイート

	TriglavPlugInBool*	selected	out	レイヤーマスクが選択されているかどうか
	TriglavPlugInHostObject	hostObject	in	ホストオブジェクト

指定されたホストオブジェクトから、編集レイヤーのレイヤーマスクが選択されているかどうかを取得します。

レイヤーマスクが選択されている場合、引数selectedにkTriglavPlugInBoolTrueが代入されます。

レイヤーマスクが選択されていない場合、引数selectedにkTriglavPlugInBoolFalseが代入されます。

取得に失敗した場合、引数selectedにkTriglavPlugInBoolFalseが代入されます。

TriglavPlugInFilterRunIsAlphaLocked()

引数	TriglavPlugInRecordSuite	record	in	レコードスイート
	TriglavPlugInBool*	locked	out	透明ピクセルがロックされているかどうか
	TriglavPlugInHostObject	hostObject	in	ホストオブジェクト

指定されたホストオブジェクトから、編集レイヤーの透明ピクセルがロックされているかどうかを取得します。

透明ピクセルがロックされている場合、引数lockedにkTriglavPlugInBoolTrueが代入されます。

透明ピクセルがロックされていない場合、引数lockedにkTriglavPlugInBoolFalseが代入されます。

取得に失敗した場合、引数selectedにkTriglavPlugInBoolFalseが代入されます。

TriglavPlugInFilterRunGetSourceOffscreen()

引数	TriglavPlugInRecordSuite	record	in	レコードスイート
	TriglavPlugInOffscreenObject*	offscreenObject	out	オフスクリーンオブジェクト
	TriglavPlugInHostObject	hostObject	in	ホストオブジェクト

指定されたホストオブジェクトから、実行処理の元のオフスクリーンを取得します。失敗した場合、引数 offscreenObject に NULL が代入されます。

TriglavPlugInFilterRunGetDestinationOffscreen()

引数	TriglavPlugInRecordSuite	record	in	レコードスイート
	TriglavPlugInOffscreenObject*	offscreenObject	out	オフスクリーンオブジェクト
	TriglavPlugInHostObject	hostObject	in	ホストオブジェクト

指定されたホストオブジェクトから、実行処理の先のオフスクリーンを取得します。失敗した場合、引数 offscreenObject に NULL が代入されます。

TriglavPlugInFilterRunHasSelectAreaOffscreen()

引数	TriglavPlugInRecordSuite	record	in	レコードスイート
	TriglavPlugInBool*	hasSelectAreaOffscreen	out	選択範囲のオフスクリーンを持っているか
	TriglavPlugInHostObject	hostObject	in	ホストオブジェクト

指定されたホストオブジェクトから、編集レイヤーが選択範囲のオフスクリーンを持っているかを取得します。

選択範囲のオフスクリーンを持っている場合、引数hasSelectAreaOffscreenにkTriglavPlugInBoolTrueが代入されます。

選択範囲のオフスクリーンを持っていない場合、引数hasSelectAreaOffscreenにkTriglavPlugInBoolFalseが代入されます。

取得に失敗した場合、引数hasSelectAreaOffscreenにkTriglavPlugInBoolFalseが代入されます。

TriglavPlugInFilterRunGetSelectAreaRect()

引数	TriglavPlugInRecordSuite	record	in	レコードスイート
	TriglavPlugInRect*	rect	out	選択範囲領域の外接矩形(pixel)
	TriglavPlugInOffscreenObject	offscreenObject	in	オフスクリーンオブジェクト

指定されたホストオブジェクトから、選択範囲領域の外接矩形を取得します。選択範囲がない場合は、編集レイヤーのオフスクリーン領域を取得します。取得に失敗した場合、引数 rect に空の矩形が代入されます。

TriglavPlugInFilterRunGetSelectAreaOffscreen()

引数	TriglavPlugInRecordSuite	record	in	レコードスイート
	TriglavPlugInOffscreenObject*	offscreenObject	out	選択範囲のオフスクリーン
	TriglavPlugInHostObject	hostObject	in	ホストオブジェクト

指定されたホストオブジェクトから、選択範囲のオフスクリーンを取得します。失敗した場合、引数offscreenObjectにNULLが代入されます。

TriglavPlugInFilterRunHasSelectAreaOffscreen 関数を使用することで、選択範囲のオフスクリーンが存在するかどうかを調べることが出来ます。

TriglavPlugInFilterRunUpdateDestinationOffscreenRect()

引数	TriglavPlugInRecordSuite	record	in	レコードスイート
	TriglavPlugInHostObject	hostObject	in	ホストオブジェクト
	const TrigravPlugInRect*	updateRect	in	更新する範囲

指定されたホストオブジェクトの、処理実行先のオフスクリーンを更新します。

TriglavPlugInFilterRunGetMainColor()

引数	TriglavPlugInRecordSuite	record	in	レコードスイート
	TriglavPlugInRGBColor*	mainColor	in	メインカラー
	TriglavPlugInUInt8*	mainAlpha	in	メインアルファ
	TriglavPlugInHostObject	hostObject	in	ホストオブジェクト

指定されたホストオブジェクトから、現在のメインカラーを取得します。失敗した場合、引数mainColorとmainAlphaに値が代入されません。

TriglavPlugInFilterRunGetSubColor()

引数	TriglavPlugInRecordSuite	record	in	レコードスイート
	TriglavPlugInRGBColor*	subColor	in	サブカラー
	TriglavPlugInUInt8*	subAlpha	in	サブアルファ
	TriglavPlugInHostObject	hostObject	in	ホストオブジェクト

指定されたホストオブジェクトから、現在のサブカラーを取得します。失敗した場合、引数subColorとsubAlphaに値が代入されません。

TriglavPlugInFilterRunGetDrawColor()

引数	TriglavPlugInRecordSuite	record	in	レコードスイート

	TriglavPlugInRGBColor*	drawColor	in	描画色
	TriglavPlugInUInt8*	drawAlpha	in	描画色の不透明度
	TriglavPlugInHostObject	hostObject	in	ホストオブジェクト

指定されたホストオブジェクトから、現在の描画色を取得します。失敗した場合、引数drawColorとdrawAlphaに値が代入されません。

TriglavPlugInFilterRunProcess()

引数	TriglavPlugInRecordSuite	record	in	レコードスイート
	TriglavPlugInInt*	result	out	ホストオブジェクトのフィルタ実行の処理の戻り値
	TriglavPlugInHostObject	hostObject	in	ホストオブジェクト
	const TriglavPlugInInt	processState	in	プラグインモジュールのフィルタ実行の処理の状態

指定されたプラグインモジュールのフィルタ実行の処理の状態から、指定されたホストオブジェクトのフィルタ実行処理の状態を取得します。

引数processStateと、それに対する引数resultの状態を示します。

開発者は、引数resultの結果に応じて適切な処理を行う必要があります。

processState	result	
kTriglavPlugInFilterRunProcessStateStart	kTriglavPlugInFilterRunProcessResultRestart	処理の再計算が必要
	kTriglavPlugInFilterRunProcessResultContinue	処理の継続
	kTriglavPlugInFilterRunProcessResultExit	処理の終了が必要(キャンセルボタンや×ボタンが押された)
kTriglavPlugInFilterRunProcessStateContinue	kTriglavPlugInFilterRunProcessResultRestart	処理の再計算が必要
	kTriglavPlugInFilterRunProcessResultContinue	処理の継続
	kTriglavPlugInFilterRunProcessResultExit	処理の終了が必要(キャンセルボタンや×ボタンが押された)
kTriglavPlugInFilterRunProcessStateEnd	kTriglavPlugInFilterRunProcessResultRestart	処理の再計算が必要
	kTriglavPlugInFilterRunProcessResultExit	処理の終了が必要(OKボタンが押された)
kTriglavPlugInFilterRunProcessStateAbort	kTriglavPlugInFilterRunProcessResultContinue	処理の継続
	kTriglavPlugInFilterRunProcessResultExit	処理の終了が必要(キャンセルボタンや×ボタンが押された)

TriglavPlugInFilterRunSetProgressTotal()

引数	TriglavPlugInRecordSuite	record	in	レコードスイート
	TriglavPlugInHostObject	hostObject	in	ホストオブジェクト
	const TriglavPlugInInt	progressTotal	in	プログレスのトータル

指定されたホストオブジェクトにプログレスのトータルを設定します。失敗した場合、引数hostObjectにプログレスのトータルが設定されません。

TriglavPlugInFilterRunSetProgressDone()

引数	TriglavPlugInRecordSuite	record	in	レコードスイート
	TriglavPlugInHostObject	hostObject	in	ホストオブジェクト
	const TriglavPlugInInt	processState	in	プログレスの進捗

指定されたホストオブジェクトにプログレスの進捗を設定します。失敗した場合、引数hostObjectにプログレスの進捗が設定されません。

商標および著作権について

・CELSYS、CLIP STUDIO PAINT、CLIPは、株式会社セルシスの商標または登録商標です。
・Microsoft、Windows は、米国Microsoft Corporation の米国およびその他の国における登録商標または商標です。
・Apple、Macintosh の名称は、米国Apple Inc. の米国およびその他の国における商標または登録商標です。
・その他、記載されております会社名または製品名は、各社の商標または登録商標です。

発行者：株式会社セルシス