SmartAR™ リファレンス

1.1 版

変更履歴

版数	区分	項目	変更内容	日付
1.0	新規	-	新規作成	2016/01/27
1.0.1	変更	全般	変更履歴を削除	2016/02/01
1.1	更新	全般	ドキュメントのバージョンを更新	2016/03/28

目次

共通 API

定数

定数一覧

名前	値	解説
SAR_OK	0	正常終了
SAR_ERROR_ALREADY_INITIALIZED	-2138308607	初期化済み。SmartAR™ SDK は既に初期化されています。 (0x808c0001)
SAR_ERROR_ALREADY_REGISTERED	-2138308601	登録済み。入力は既に登録済みです。 (0x808c0007)
SAR_ERROR_ALREADY_STARTED	-2138308599	開始済み。SmartAR™ SDK は既に開始しています。 (0x808c0009)
SAR_ERROR_BUSY	-2138308594	ロック中。他のスレッドからのコールにより SmartAR™ SDK がロックされています。 (0x808c000e)
SAR_ERROR_INVALID_POINTER	-2138308602	不正ポインタ。与えられたポインタが不正です。 (0x808c0006)
SAR_ERROR_INVALID_VALUE	-2138308603	不正値。与えられた値が不正です。 (0x808c0005)
SAR_ERROR_NO_DICTIONARY	-2138308595	認識対象未登録。認識対象が登録されていません。 (0x808c000d)
SAR_ERROR_NOT_EMPTY	-2138308604	リソース未開放。まだ開放されていないリソースがあります。 (0x808c0004)
SAR_ERROR_NOT_REGISTERED	-2138308600	未登録。入力はまだ登録されていません。 (0x808c0008)
SAR_ERROR_NOT_REQUIRED	-2138308597	準備未完了。SmartAR™ SDK はまだ要求を処理する準備ができていません。 (0x808c000b)
SAR_ERROR_NOT_STARTED	-2138308598	未開始。SmartAR™ SDK はまだ開始していません。 (0x808c000a)
SAR_ERROR_NOT_STOPPED	-2138308605	未停止。SmartAR™ SDK がまだ停止していません。 (0x808c0003)
SAR_ERROR_OUT_OF_MEMORY	-2138308606	メモリ不足。SmartAR™ SDK はメモリが確保できずに失敗しました。 (0x808c0002)
SAR_ERROR_UNINITIALIZED	-2138308608	未初期化。SmartAR™ SDK はまだ初期化されていません。 (0x808c0000)
SAR_ERROR_VERSION_MISSMATCH	-2138308596	バージョン不整合。与えられた値やファイルは正しいバージョンではありません。 (0x808c000c)

列挙値

SarFacing

カメラの搭載位置

定　義

```
#include <SarCommon.h>
enum SarFacing{
      SAR_FACING_BACK,
      SAR_FACING_FRONT,
};
```

列　挙　値

名前	解説
SAR_FACING_BACK	背面カメラ
SAR_FACING_FRONT	前面カメラ

解　説

カメラの搭載されている位置を示す列挙値。

SarRotation

回転角度

定　義

```
#include <SarCommon.h>
enum SarRotation{
       SAR_ROTATION_0,
       SAR_ROTATION_90,
       SAR_ROTATION_180,
       SAR_ROTATION_270,
};
```

列　挙　値

名前	解説
SAR_ROTATION_0	0 度
SAR_ROTATION_90	90 度
SAR_ROTATION_180	180 度
SAR_ROTATION_270	270 度

解　説

回転角度を表す列挙値。

SarImageFormat

画像フォーマット

定　義

```
#include <SarCommon.h>
enum SarImageFormat{
       SAR_IMAGE_FORMAT_L8,
       SAR_IMAGE_FORMAT_YCRCB420,
       SAR_IMAGE_FORMAT_YCBCR420,
       SAR_IMAGE_FORMAT_RGBA8888,
       SAR_IMAGE_FORMAT_JPEG
};
```

列　挙　値

名前	解説
SAR_IMAGE_FORMAT_L8	モノクロ 8bit
SAR_IMAGE_FORMAT_YCRCB420	YCrCb420 形式
SAR_IMAGE_FORMAT_YCBCR420	YCbCr420 形式
SAR_IMAGE_FORMAT_RGBA8888	RGBA 各 8bit
SAR_IMAGE_FORMAT_JPEG	JPEG

解　説

画像フォーマットを表す列挙値。

クラス

SarMemoryAllocator

メモリアロケータ

定　義

```
#include <SarCommon.h>
class SarMemoryAllocator {
public:
        virtual~ SarMemoryAllocator();
        virtual void* allocate(size_t size) = 0;
        virtual void deallocate(void* ptr) = 0;
};
```

解　説

メモリアロケータのインタフェース。

SarStreamIn

入力ストリーム

定　義

```
#include <SarCommon.h>
class SarStreamIn : SarNonCopyable {
public:
        virtual ~SarStreamIn() {}
        virtual size_t sarRead(void* buf, size_t size) = 0;
};
```

解　説

入力ストリームのインタフェース。

SarStreamOut

出力ストリーム

定　義

```
#include <SarCommon.h>
class SarStreamOut : SarNonCopyable {
public:
        virtual ~SarStreamOut() {}
        virtual size_t sarWrite(const void* buf, size_t size) = 0;
};
```

解　説

出力ストリームのインタフェース。

SarFileStreamIn

ファイルからの入力ストリーム

定　義

```
#include <SarCommon.h>
class SarFileStreamIn : public SarStreamIn {
public:
        SarFileStreamIn(SarSmart* smart, const char* filepath);
        virtual ~SarFileStreamIn();
        bool sarIsConstructorFailed();
        virtual size_t sarRead(void* buf, size_t size);
};
```

解　説

ファイルからの入力ストリームクラス。

SarFileStreamOut

ファイルからの出力ストリーム

定　義

```
#include <SarCommon.h>
class SarFileStreamOut : public SarStreamOut {
public:
        SarFileStreamOut(SarSmart* smart, const char* filepath);
        virtual ~SarFileStreamOut();
        bool sarIsConstructorFailed();
        virtual size_t sarWrite(const void* buf, size_t size);
};
```

解　説

ファイルへの出力ストリームクラス。

SarAssetStreamIn

リソースからの入力ストリーム

定　義

```
#include <SarCommon.h>
class SarAssetStreamIn : public SarStreamIn {
public:
        SarAssetStreamIn(SarSmart* smart, const char* filepath);
        virtual ~SarAssetStreamIn();
        bool sarIsConstructorFailed();
        virtual size_t sarRead(void* buf, size_t size);
};
```

解　説

リソースファイルからの入力ストリームクラス。

SarVector2

2 次元ベクトル

定　義

```
#include <SarCommon.h>
struct SarVector2 {
        float x_;
        float y_;

        SarVector2();
        SarVector2(float x, float y);

        void set(float x, float y);
};
```

メ　ン　バ

*x_*　X
*y_*　Y

解　説

float 型の要素を持つ 2 次元ベクトルクラス。

SarVector3

3 次元ベクトル

定　義

```
#include <SarCommon.h>
struct SarVector3 {
        float x_;
        float y_;
        float z_;

        SarVector3();
        SarVector3(float x, float y, float z);

        void set(float x, float y, float z);

        SarVector3& operator+=(const SarVector3& rhs);
        SarVector3& operator-=(const SarVector3& rhs);
        SarVector3& operator*=(float rhs);
        SarVector3& operator/=(float rhs);
        SarVector3 operator+(const SarVector3& rhs) const;
        SarVector3 operator-(const SarVector3& rhs) const;
        SarVector3 operator*(float rhs) const;
        SarVector3 operator/(float rhs) const;
};
```

メ　ン　バ

*x_*　X
*y_*　Y
*z_*　Z

解　説

float 型の要素を持つ 3 次元ベクトルクラス。

SarQuaternion

クォータニオン

定　義

```
#include <SarCommon.h>
struct SarQuaternion {
	float w_;
	float x_;
	float y_;
	float z_;

	SarQuaternion();
	SarQuaternion(float w, float x, float y, float z);

	void set(float w, float x, float y, float z);

	SarQuaternion operator-() const;
	SarQuaternion operator*(const SarQuaternion& rhs) const;
	SarQuaternion& operator*=(const SarQuaternion& rhs);
};
```

メ　ン　バ

*w_*	W
*x_*	X
*y_*	Y
*z_*	Z

解　説

4 つの float 型で定義されるクォータニオンの構造体。

SarMatrix44

4x4 行列

定　義

```
#include <SarCommon.h>
struct SarMatrix44 {
	static const int32_t NUM_VALUES = 16;

	float values_[NUM_VALUES];

	SarMatrix44();
	SarMatrix44(float values[16]);

	SarMatrix44 operator*(const SarMatrix44& rhs) const;
	SarVector3 operator*(const SarVector3& rhs) const;

	void sarSetRotation(const SarQuaternion& quat);
	void sarSetTranslation(float x, float y, float z);
	void sarSetScaling(float x, float y, float z);
};
```

メ　ン　バ

*values_*　行列の要素

解　説

16 個の float 型で定義される 4x4 行列の構造体。

SarTriangle2

三角形の頂点配列

定　義

```
#include <SarCommon.h>
struct SarTriangle2 {
        static const int32_t NUM_POINTS = 3;

        SarVector2 points_[NUM_POINTS];
};
```

メ　ン　バ

*points_*　三角形の頂点座標を格納する SarVector2 型配列

解　説

3 つの SarVector2 型で定義される三角形を表す構造体。

SarTriangle3

三角形の頂点座標

定　義

```
#include <SarCommon.h>
struct SarTriangle3 {
        static const int32_t NUM_POINTS = 3;

        SarVector3 points_[NUM_POINTS];
};
```

メ　ン　バ

*points_*　三角形の頂点座標を格納する SarVector3 型配列

解　説

3 つの SarVector3 型で定義される三角形を表す構造体。

SarSize

サイズ

定　義

```
#include <SarCommon.h>
struct SarSize {
        int32_t width_;
        int32_t height_;

        SarSize();
        SarSize(int32_t width, int32_t height);

        bool operator==(const SarSize& rhs);
};
```

メ　ン　バ

*width_*　幅
*Height_*　高さ

解　説

2 つの int32_t 型で定義されるサイズの構造体。

SarRect

長方形

定　義

```
#include <SarCommon.h>
struct SarRect {
        int32_t left_;
        int32_t top_;
        int32_t right_;
        int32_t bottom_;

        SarRect();
        SarRect(int32_t left, int32_t top, int32_t right, int32_t bottom);

        void set(int32_t left, int32_t top, int32_t right, int32_t bottom);
        int32_t width() const;
        int32_t height() const;

        bool operator!=(const SarRect& rhs) const
        bool operator==(const SarRect& rhs) const;
        SarRect operator/(int rhs) const;
};
```

メ　ン　バ

*left_*	矩形の左端の X 座標
*top_*	矩形の上端の Y 座標
*right_*	矩形の右端の X 座標
*bottom_*	矩形の下端の Y 座標

解　説

4 つの int32_t 型で定義される長方形の構造体。

SarImage

画像データを保持するクラス

定　義

```
#include <SarCommon.h>
class SarImage {
public:
        SarImage(SarSmart* smart);
        SarImage(const SarImage &other);
        ~SarImage();
        SarImage & operator = (const SarImage &other);

        void setData(unsigned char* pixels);
        unsigned char* getData();
        void setWidth(int32_t width);
        int32_t getWidth();
        void setHeight(int32_t height);
        int32_t getHeight();
        void setStride(int32_t stride);
        int32_t getStride();
        void setImageFormat(SarImageFormat format);
        SarImageFormat getImageFormat();
};
```

解　説

画像データを保持するクラス。

SarImage::SarImage

コンストラクタ

定　義

```
public SarImage(
        SarSmart* smart
        );

public SarImage(
        const SarImage &other
        );
```

引　数

[in] smart	SarImage のインスタンスへのポインタ
[in] other	他の SarImage クラスのインスタンス

解　説

SarImage クラスのコンストラクタ。

SarImage::~SarImage

デストラクタ

定　義

```
public ~SarImage();
```

解　説

SarImage クラスのデストラクタ。

SarImage::setData

ピクセルデータを設定する

定　義

```
public void setData(
        unsigned char* pixels
);
```

引　数

[in] pixels　ピクセルデータへのポインタ

解　説

ピクセルデータを設定します。
認証に成功している場合のみ設定できます。

SarImage::getData

ピクセルデータを取得する

定　義

```
public unsigned char* getData();
```

返　り　値

ピクセルデータへのポインタ

解　説

ピクセルデータを取得します。
認証に成功している場合のみ取得できます。
認証に失敗している場合は NULL が取得されます。

SarImage::setWidth

画像の幅を設定する

定　義

```
public void setWidth(
        int32_t width
);
```

引　数

[in] width　　画像の幅

解　説

画像の幅を設定します。

SarImage::getWidth

画像の幅を取得する

定　義

```
public int32_t getWidth();
```

返　り　値

画像の幅

解　説

画像の幅を取得します。

SarImage::setHeight

画像の高さを設定する

定　義

```
public void setHeight(
        int32_t height
);
```

引　数

[in] height　画像の高さ

解　説

画像の高さを設定します。

SarImage::getHeight

画像の高さを取得する

定　義

```
public int32_t getHeight();
```

返　り　値

画像の高さ

解　説

画像の高さを取得します。

SarImage::setStride

行間のオフセットを設定する

定　義

```
public void setStride(
        int32_t stride
);
```

引　数

[in] stride　　行間のオフセット。0 ならオフセットは画像の幅に一致

解　説

画像の行の終端から次の行の先頭までのオフセットを設定します。
0 の場合は、オフセットと画像の幅が一致していることを意味します。

SarImage::getStride

行間のオフセットを取得する

定　義

```
public int32_t getStride();
```

返　り　値

行間のオフセット。`0` ならオフセットは画像の幅に一致

解　説

画像の行の終端から次の行の先頭までのオフセットを取得します。
`0` の場合は、オフセットと画像の幅が一致していることを意味します。

SarImage::setImageFormat

画像のフォーマットを設定する

定　義

```
public void setImageFormat(
        SarImageFormat format
);
```

引　数

`[in] format`　画像のフォーマット

解　説

画像のフォーマットを設定します。
画像フォーマットについては `SarImageFormat` の記述を参照して下さい。

SarImage::getImageFormat

画像のフォーマットを取得する

定　義

```
public SarImageFormat getImageFormat();
```

返　り　値

画像のフォーマット

解　説

画像のフォーマットを取得します。
画像フォーマットについては `SarImageFormat` の記述を参照して下さい。

SarRecognizer

定数

定数一覧

名前	解説
SAR_MAX_NUM_INITIALIZATION_POINTS	SarRecognizer によって認識される初期化ポイントの最大数
SAR_MAX_NUM_LANDMARKS	SarRecognizer によって認識されるランドマークの最大数
SAR_MAX_PROPAGATION_DURATION	推定結果を伝播できる最長の期間 [usec]

列挙値

SarRecognitionMode

認識処理モード

定　義

```
#include <SarRecognizer.h>
enum SarRecognitionMode {
      SAR_RECOGNITION_MODE_TARGET_TRACKING,
      SAR_RECOGNITION_MODE_SCENE_MAPPING
};
```

列　挙　値

名前	解説
SAR_RECOGNITION_MODE_TARGET_TRACKING	TargetTracking モード。 TargetTracking モードはカメラ画像内の平面物体を認識し、SmartAR™動作端末のポーズ（位置と姿勢）を推定するためのモードです。任意の自然画像を認識対象とすることができます。
SAR_RECOGNITION_MODE_SCENE_MAPPING	SceneMapping モード。 SceneMapping モードは、SmartAR™動作端末を空間中で動かすことで、カメラ画像の変化とセンサーから未知の環境の 3D 構造と、その 3D 構造内での SmartAR™動作端末のポーズ（位置と姿勢）を推定するモードです。SLAM (Simultaneous Localization and Mapping)と呼ばれる技術を使っています。

解　説

SarRecognizer が行う認識処理の各処理モードを表す列挙値。

SarSearchPolicy

認識処理の探索方針

定　義

```
#include <SarRecognizer.h>
enum SarSearchPolicy {
       SAR_SEARCH_POLICY_FAST,
       SAR_SEARCH_POLICY_PRECISIVE,
};
```

列　挙　値

名前	解説
SAR_SEARCH_POLICY_FAST	探索速度を重視した探索方針です。 １回の探索を高速に実行することを重視した探索方針です。１回の探索に必要な時間が少ないため同じ時間で多くの探索を実行できますが、認識対象が入力画像内にある程度の大きさで映る必要があります。小さく映った認識対象を探索したい場合は SAR_SEARCH_POLICY_PRECISIVE を利用してください。
SAR_SEARCH_POLICY_PRECISIVE	探索感度を重視した探索方針です。 SAR_SEARCH_POLICY_FAST に比べて１回の探索に必要な 時間は多く必要ですが、入力画像内に小さく映った認識対象も探索することが可能です。 SAR_SEARCH_POLICY_FAST と比較して探索時間は約４倍、 認識可能な対象の大きさは約４分の１となります。 認識対象が入力画像内にある程度以上の大きさで映っている場合、１回の探索時間の短い SAR_SEARCH_POLICY_FAST の方が高い体感性能を得られる 場合があります。

解　説

認識処理の探索方針を表す列挙値。TargetTracking モードでは探索状態で認識対象を探索し、対象を発見した後にトラッキング状態へと遷移します。SarSearchPolicy は、探索状態におけるライブラリの探索方針を表します。利用する方針によって、探索に必要な計算時間や、入力画像内での探索可能な対象物体の大きさが変わります。この設定は自然画像が認識対象である場合にのみ有効です。

SarSceneMappingInitMode

SLAM の初期化モード

定　義

```
#include <SarRecognizer.h>
enum SarSceneMappingInitMode {
	SAR_SCENE_MAPPING_INIT_MODE_TARGET,
	SAR_SCENE_MAPPING_INIT_MODE_HFG,
	SAR_SCENE_MAPPING_INIT_MODE_VFG,
	SAR_SCENE_MAPPING_INIT_MODE_SFM,
	SAR_SCENE_MAPPING_INIT_MODE_DRY_RUN,
};
```

列　挙　値

名前	解説
SAR_SCENE_MAPPING_INIT_MODE_TARGET	学習した自然画像で初期化します。シーン固定座標系の原点に学習した自然画像があると仮定し、自然画像を検出して初期化します。
SAR_SCENE_MAPPING_INIT_MODE_HFG	重力方向を用いた水平面推定(Horizontal From Gravity:HFG)初期化です。センサーを利用して水平面の推定を行い、SLAM を初期化します。カメラをテクスチャの豊富な水平面(テーブルに追いた写真など)に向けてください。 注意：このモードは、重力方向を知るためにセンサーからの入力を必要とします。
SAR_SCENE_MAPPING_INIT_MODE_VFG	重力方向を用いた鉛直面推定(Vertical From Gravity:VFG)初期化です。センサーを利用して鉛直面の推定を行い、SLAM を初期化します。カメラをテクスチャの豊富な鉛直面(ビルの壁面など）に向けて下さい。カメラと鉛直面は正対している必要があります。 注意：このモードは、重力方向を知るためにセンサーからの入力を必要とします。
SAR_SCENE_MAPPING_INIT_MODE_SFM	2 次元画像内での特徴点トラッキングを行い、トラッキング開始時と終了時の視差から 3 次元構造を推定して SLAM を初期化します（Structure From Motion：SFM）。ユーザには、カメラをテクスチャの豊富なシーンに向けてゆっくりと平行に動かすように指示して下さい。 注意：センサー情報を渡さない場合、被写体を平面と仮定して処理します。

名前	解説
SAR_SCENE_MAPPING_INIT_MODE_DRY_RUN	SFM や HFG、VFG のための予行演習モードです。SFM や HFG、VFG で使われる特徴点トラッキングを行ない、2 次元画像内にいくつの特徴点があるかを調べます。特徴点の個数と位置は、SarRecognizer::sarGetResult()または SarRecognizer::sarGetResults()で返却される SarRecognitionResult 構造体の landmarks_と initPoints_メンバで知ることができます。 このモードでは実際の初期化は行われません。初期化を行うには SFM か HFG、VFG 初期化モードを利用してください。

解　説

SceneMapping モードの SLAM の初期化モードを表す列挙値。

SarTargetTrackingState

TargetTracking モードにおける認識状態

定　義

```
#include <SarRecognizer.h>
enum SarTargetTrackingState {
       SAR_TARGET_TRACKING_STATE_IDLE,
       SAR_TARGET_TRACKING_STATE_SEARCH,
       SAR_TARGET_TRACKING_STATE_TRACKING,
};
```

列 挙 値

名前	解説
SAR_TARGET_TRACKING_STATE_IDLE	アイドル状態。 TargetTracking モードの初期状態です。
SAR_TARGET_TRACKING_STATE_SEARCH	探索状態。 登録された認識対象をカメラ画像から探索している状態です。
SAR_TARGET_TRACKING_STATE_TRACKING	トラッキング状態。 認識対象の探索に成功し、認識対象をトラッキングしている状態です。

解　説

TargetTracking モードにおける SarRecognizer の認識状態を表す列挙値。
以下に TargetTracking モードで動作する SarRecognizer の状態遷移図を示します。

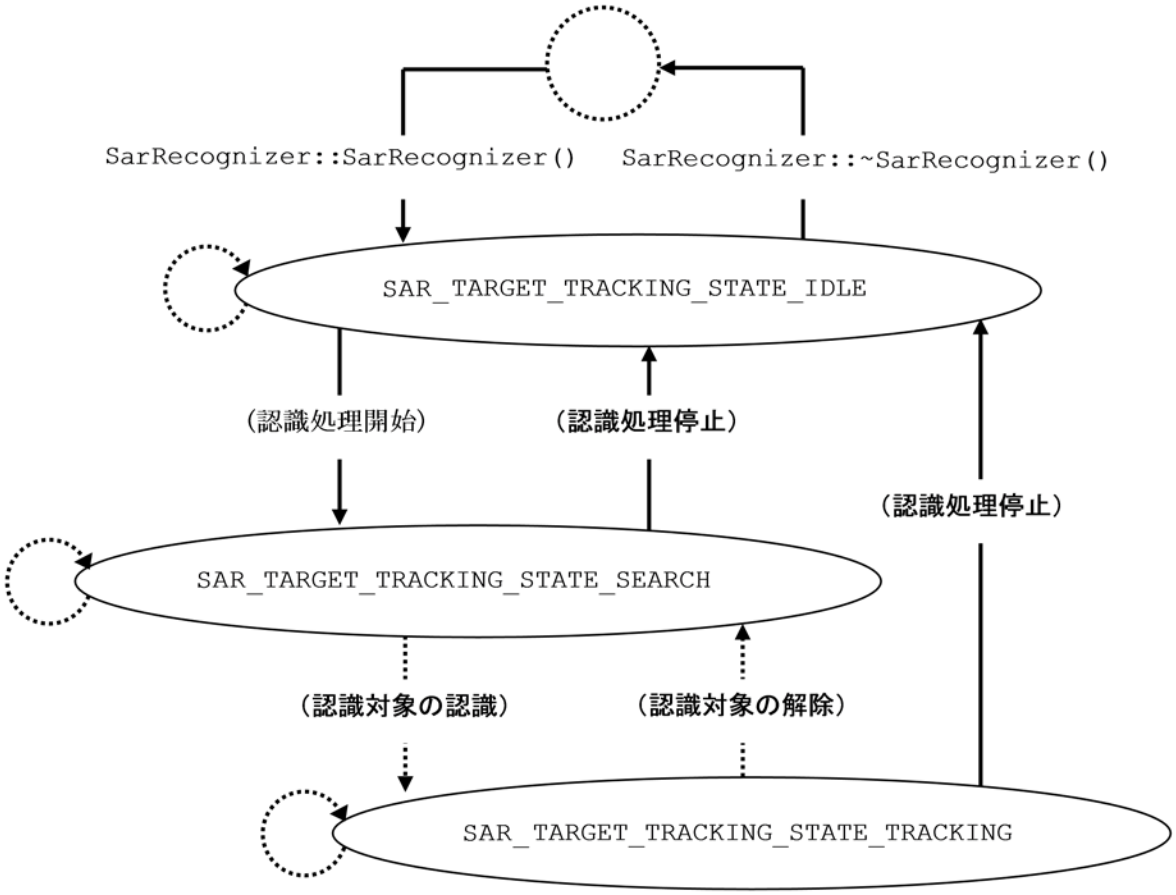

図 1　TargetTracking モードにおける SarRecognizer の状態遷移図

SarSceneMappingState

SceneMapping モードにおける認識状態

定　義

```
#include <SarRecognizer.h>
enum SarSceneMappingState {
	SAR_SCENE_MAPPING_STATE_IDLE,
	SAR_SCENE_MAPPING_STATE_SEARCH,
	SAR_SCENE_MAPPING_STATE_TRACKING,
	SAR_SCENE_MAPPING_STATE_LOCALIZE,
	SAR_SCENE_MAPPING_STATE_LOCALIZE_IMPOSSIBLE,
};
```

列　挙　値

名前	解説
SAR_SCENE_MAPPING_STATE_IDLE	アイドル状態。 この状態では認識処理を実行することはできません。認識処理を実行するには、SAR_SCENE_MAPPING_STATE_SEARCH 状態へ遷移し、カメラのポーズとシーンマップの初期化を行う必要があります。
SAR_SCENE_MAPPING_STATE_SEARCH	サーチ(初期化)状態。 この状態では SceneMapping モードは指定された初期化モードでカメラのポーズとシーンマップを初期化しようとします。初期化が成功すると、状態は次の状態 SAR_SCENE_MAPPING_STATE_TRACKING に自動的に遷移します。
SAR_SCENE_MAPPING_STATE_TRACKING	トラッキング状態。 カメラのポーズが正常に推定されており、シーンマップの拡大処理が正常に動作している状態です。 シーンマップはカメラがシーン内を移動していくにつれて徐々に作成されます。 カメラの動きが速過ぎる場合、カメラのポーズの推定が正常に行えなくなり、トラッキングに失敗します。 トラッキングに失敗すると SAR_SCENE_MAPPING_STATE_LOCALIZE に遷移します。
SAR_SCENE_MAPPING_STATE_LOCALIZE	ローカライズ状態。 一時的にカメラのポーズが推定できなくなった状態です。 SceneMapping モードは SAR_SCENE_MAPPING_STATE_TRACKING 状態の時に作成したシーンマップを使い、カメラのポーズを再度推定することを試みます。 ユーザが SAR_SCENE_MAPPING_STATE_TRACKING 状態の時にとったポーズにカメラを戻すように誘導してください。

名前	解説
SAR_SCENE_MAPPING_STATE_LOCALIZE_IMPOSSIBLE	ローカライズ不能な状態。 カメラのポーズを再推定するために必要となるシーンマップが不十分であり、初期化しなおす必要があります。 ユーザへ再度初期化するように誘導してください。

解　説

SceneMapping モードにおける SarRecognizer の認識状態を表した列挙値。
以下より SceneMapping モードで動作する SarRecognizer の状態遷移図を示します。

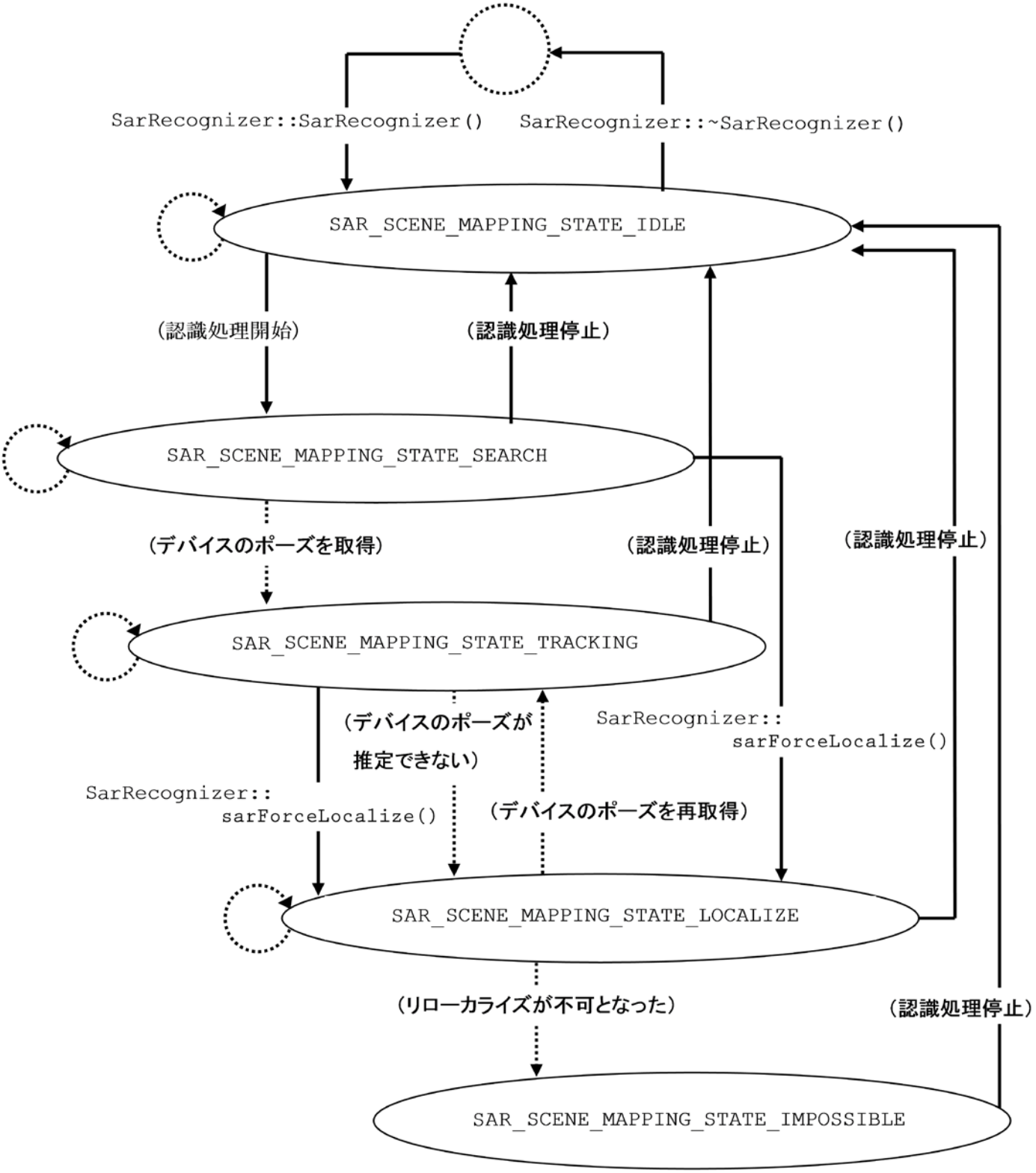

図 2　SceneMapping モードにおける SarRecognizer の状態遷移図

SarLandmarkState

ランドマークの認識状態

定　義

```
#include <SarRecognizer.h>
enum SarLandmarkState {
	SAR_LANDMARK_STATE_TRACKED,
	SAR_LANDMARK_STATE_LOST,
	SAR_LANDMARK_STATE_SUSPENDED,
	SAR_LANDMARK_STATE_MASKED,
};
```

列　挙　値

名前	解説
SAR_LANDMARK_STATE_TRACKED	トラッキング中
SAR_LANDMARK_STATE_LOST	ロスト中
SAR_LANDMARK_STATE_MASKED	マスクされている
SAR_LANDMARK_STATE_SUSPENDED	その他の状態

解　説

ランドマークの認識状態を表す列挙値。ランドマークは、3 次元の位置が推定されたシーン中の点です。ランドマークの 3 次元位置の推定は、前の入力画像と今の入力画像に写っているランドマークをトラッキングし、その間の 2 次元位置の変化から計算しています。

SarDenseMapMode

デンスマップモード

定　義

```
#include <SarRecognizer.h>
enum SarDenseMapMode {
       SAR_DENSE_MAP_DISABLE,
       SAR_DENSE_MAP_SEMI_DENSE,
};
```

列　挙　値

名前	解説
SAR_DENSE_MAP_DISABLE	デンスマップモード無効
SAR_DENSE_MAP_SEMI_DENSE	デンスマップモード有効

解　説

シーンマップを構成するランドマークをより密に生成する際に使用するデンスマップモードに関する列挙値。

クラス

SarRecognizer

認識処理を実行するクラス

定　義

```
#include <SarRecognizer.h>
class SarRecognizer : SarNonCopyable {
public:
	SarRecognizer(SarSmart* smart,
	   SarRecognitionMode recogMode = SAR_RECOGNITION_MODE_TARGET_TRACKING,
	   SarSceneMappingInitMode initMode = SAR_SCENE_MAPPING_INIT_MODE_TARGET);
	~SarRecognizer();
	bool sarIsConstructorFailed();

	// setting
	int32_t sarSetCameraDeviceInfo(const SarCameraDeviceInfo& info);
	int32_t sarSetSensorDeviceInfo(const SarSensorDeviceInfo& info);
	int32_t sarSetTargets(const SarTarget* const* targets, int32_t numTargets);

	// start and stop
	int32_t sarReset();

	// run
	int32_t sarRun(const SarRecognitionRequest& request);
	int32_t sarDispatch(const SarRecognitionRequest& request);
	int32_t sarRunWorker();
	int32_t sarSetWorkDispatchedListener(SarWorkDispatchedListener*
listener);

	// get results
	int32_t sarGetNumResults() const;
	int32_t sarGetResults(SarRecognitionResult* results, int32_t maxResults)
const;
	int32_t sarGetResult(const SarTarget& target, SarRecognitionResult* result)
const;
	int32_t sarSetRecognizedListener(SarRecognizedListener* listener);

	// at present for target tracking only
	int32_t sarSetMaxTargetsPerFrame(int32_t maxTargets);
	int32_t sarSetSearchPolicy(SarSearchPolicy policy);

	// at present for scene mapping only
	int32_t sarPropagateResult(const SarRecognitionResult& fromResult,
	   SarRecognitionResult* toResult, uint64_t timestamp,
	   bool useVelocity = true) const;
	int32_t sarSetMaxTriangulateMasks(int32_t maxMasks);

	// for scene mapping
	int32_t sarSaveSceneMap(SarStreamOut* stream) const;
	int32_t sarFixSceneMap(bool isFix);
	int32_t sarForceLocalize();
	int32_t sarRemoveLandmark(const SarLandmark& landmark);
	int32_t sarSetDenseMapMode(SarDenseMapMode mode);
};
```

解　説

TargetTracking モードまたは SceneMapping モードを使用し認識処理を行います。
SarRecognizer による認識処理の概要と使用方法は「SmartSDK-Overview.doc」を参照してください。

SarRecognizer::SarRecognizer

コンストラクタ

定　義

```
public SarRecognizer(
       SarSmart* smart,
       SarRecognitionMode recogMode = SAR_RECOGNITION_MODE_TARGET_TRACKING,
       SarSceneMappingInitMode initMode = SAR_SCENE_MAPPING_INIT_MODE_TARGET
       );
```

引　数

[in] smart	SarSmart クラスのインスタンスへのポインタ
[in] recogMode	認識処理モード
[in] initMode	SceneMapping 使用時の初期化モード 注意: 認識処理モードとして SceneMapping モードが指定された場合のみ有効です。

解　説

SarRecognizer クラスのコンストラクタ。

initMode で指定される SceneMapping の初期化モードは、認識処理モードとして SceneMapping が選択された場合のみ利用されます。

SarRecognizer::~SarRecognizer

デストラクタ

定　義

```
public ~SarRecognizer();
```

解　説

SarRecognizer クラスのデストラクタ。

SarRecognizer::sarIsConstructorFailed

コンストラクタのエラー確認

定　義

```
public bool sarIsConstructorFailed();
```

返　り　値

コンストラクタでエラーが発生した場合は true を返し、それ以外は false を返します。

解　説

コンストラクタでエラーが発生したかを確認します。

SarRecognizer::sarSetCameraDeviceInfo

カメラデバイスの情報を設定する

定　義

```
public int32_t sarSetCameraDeviceInfo(
       const SarCameraDeviceInfo& info
       );
```

引　数

[in] info　　カメラデバイスの情報

返　り　値

エラーコード

解　説

カメラデバイスの情報を設定します。

SarRecognizer::sarSetSensorDeviceInfo

センサーデバイスの情報を設定する

定　義

```
public int32_t sarSetSensorDeviceInfo(
       const SarSensorDeviceInfo& info
       );
```

引　数

[in] info　　センサーデバイスの情報

返　り　値

エラーコード

解　説

センサーデバイスの情報を設定します。

SarRecognizer::sarSetTargets

認識対象の登録または登録解除を行う

定　義

```
public int32_t sarSetTargets(
       const SarTarget* const* targets,
       int32_t numTargets
       );
```

引　数

[in] targets	認識対象が格納される配列へのポインタ。 すべての認識対象の登録を解除するときは、NULL を指定してください。
[in] numTargets	targets の要素数。 すべての認識対象の登録を解除するときは、0 を指定してください。

返　り　値

エラーコード

解　説

認識対象を SarRecognizer に登録または登録解除を行います。
すべての認識対象の登録を解除する場合は、***targets*** に NULL、***numTargets*** に 0 を指定し sarSetTargets()を呼び出してください。

SarRecognizer::sarReset

認識状態をリセットする

定　義

```
public int32_t sarReset();
```

返　り　値

エラーコード

解　説

現在の認識状態をリセットします。

SarRecognizer::sarRun

認識処理を実行する

定　義

```
public int32_t sarRun(
       const SarRecognitionRequest& request
       );
```

引　数

[in] request　　認識処理に必要となる入力データ

返　り　値

エラーコード

解　説

指定された入力データを使用し認識処理を同期的に実行します。

SarRecognizer::sarDispatch

認識処理の非同期実行の要求

定　義

```
public int32_t sarDispatch(
       const SarRecognitionRequest& request
       );
```

引　数

[in] request　　認識処理に必要となる入力データ

返　り　値

エラーコード

解　説

指定された入力データによる認識処理の非同期実行を要求します。
実際の認識処理の実行は sarRunWorker()にて行われます。

SarRecognizer::sarRunWorker

認識処理の非同期実行

定　義

```
public int32_t sarRunWorker();
```

返　り　値

エラーコード

解　説

sarDispatch()で要求された認識処理を実行します。
本関数をアプリケーションが作成したワーカースレッド内で呼び出すことにより、認識処理の非同期実行を実現できます。

SarRecognizer::sarSetWorkDispatchedListener

sarRunWorker()の実行要求を受け取るリスナ

定　義

```
public int32 sarSetWorkDispatchedListener(
       SarWorkDispatchedListener* listener
       );
```

引　数

[in] listener　SarWorkDispatchedListener クラスのインスタンスへのポインタ

返　り　値

エラーコード

解　説

sarRunWorker()の実行要求を受け取るリスナを設定します。

SarRecognizer::sarGetNumResults

認識結果の数を取得する

定　義

```
public int32_t sarGetNumResults() const;
```

返　り　値

認識結果の数

解　説

SarRecognizer::sarGetResults()が返却する認識結果の数を返します。

SarRecognizer::sarGetResults

複数の認識結果を取得する

定　義

```
public int32_t sarGetResults(
       SarRecognitionResult* results,
       int32_t maxResults
       ) const;
```

引　数

[out] results	認識結果を格納する SarRecognitionResult 型配列へのポインタ
[in] maxResults	取得する認識結果の最大数

返　り　値

正常に終了した場合は取得した数を返し、異常終了した場合はエラーコードを返す。

解　説

maxResults で指定された数を上限とした複数の認識結果を取得します。

SarRecognizer::sarGetResult

特定の認識結果を取得する

定　義

```
public int32_t sarGetResult(
       const SarTarget* target,
       SarRecognitionResult* result
       ) const;
```

引　数

[in] target	取得する認識結果に対応する認識対象
[out] result	返却された認識結果を格納する SarRecognitionResult 構造体へのポインタ

返　り　値

エラーコード

解　説

特定の認識対象に対する認識結果を取得します。

SarRecognizer::sarSetRecognizedListener

認識結果更新通知を受け取るリスナを設定する

定　義

```
public int32_t sarSetRecognizedListener(
       SarRecognizedListener* listener
       );
```

引　数

[in] listener　SarRecognizedListener クラスのインスタンスへのポインタ

返　り　値

エラーコード

解　説

認識結果更新通知を受け取るリスナを設定します。

SarRecognizer::sarSetMaxTargetsPerFrame

TargetTracking で同時に認識する対象の最大数を指定する

定　義

```
public int32_t sarSetMaxTargetsPerFrame(
       int32_t maxTargets
       );
```

引　数

[in] maxTargets　　認識対象の最大数

返　り　値

エラーコード

解　説

TargetTracking で同時に認識する対象の最大数を設定します。

SarRecognizer::sarSetSearchPolicy

探索方針を設定する

定　義

```
public int32_t sarSetSearchPolicy(
       SarSearchPolicy policy
       );
```

引　数

[in] policy　　認識対象の探索方針

返　り　値

エラーコード

解　説

認識対象の探索方針を設定します。
設定した値は、SmartAR が認識処理を実行する際のヒントとして使用されます。

SarRecognizer::sarPropagateResult

認識結果を時間伝播する

定　義

```
public int32_t sarPropagateResult(
       const SarRecognitionResult& fromResult,
       SarRecognitionResult* toResult,
       uint64_t timestamp,
       bool useVelocity = true
       ) const;
```

引　数

[in] fromResult	伝播元の認識結果
[out] toResult	伝播された認識結果
[in] timestamp	伝播先の時刻
[in] useVelocity	速度を使用するかを示すフラグ値

返　り　値

エラーコード

解　説

認識結果を指定した時間へ時間伝播し、伝播された認識結果を受け取ります。時間伝播を行うことにより、計算の遅れをある程度補償することができます。
ただし、*timestamp* で指定した時刻が SAR_MAX_PROPAGATION_DURATION 以上の場合、fromResult に格納されている認識結果が toResult に格納されます。
長時間の伝播が必要な場合は伝播回数を複数回に分けて伝播してください。

SarRecognizer::sarSetMaxTriangulateMasks

三角パッチの最大数を設定する

定　義

```
public int32_t sarSetMaxTriangulateMasks(
       int32_t maxMasks
       );
```

引　数

[in] maxMasks　　三角パッチの最大数。0 を指定した場合、マスク機能は無効化します。

返　り　値

エラーコード

解　説

マスク機能を有効化し、マスク情報に必要なデータ領域を確保します。
マスク機能の詳細については SarRecognitionRequest::triangulateMasks_の項を参照してください。

SarRecognizer::sarSaveSceneMap

シーンマップを保存する

定　義

```
public int32_t sarSaveSceneMap(
       SarStreamOut* stream
       )const;
```

引　数

[in] stream　　出力ストリーム

返　り　値

エラーコード

解　説

指定した出力ストリームに現在のシーンマップを保存します。
注意:
この機能は SarRecognizer が SceneMapping モードの場合のみ使用することができます。

SarRecognizer::sarFixSceneMap

シーンマップの固定または固定を解除する

定　義

```
public int32_t sarFixSceneMap(
       bool isFix
       );
```

引　数

[in] isFix　　固定または固定解除を示すフラグ

返　り　値

エラーコード

解　説

ランドマークの探索を中止し、シーンマップを固定します。
シーンマップの固定を解除する場合は ***isFix*** に `false` を指定し、`sarFixSceneMap()`を呼び出してください。

注意:
この機能は SarRecognizer が SceneMapping モードの場合のみ使用することができます。

SarRecognizer::sarForceLocalize

強制的にローカライズ状態へ遷移させる

定　義

```
public int32 sarForceLocalize();
```

返　り　値

エラーコード

解　説

SarRecognizer の認識状態を強制的にローカライズ状態へ遷移させます。

注意:
この機能は SarRecognizer が SceneMapping モードの場合のみ使用することができます。

SarRecognizer::sarRemoveLandmark

指定したランドマークを削除する

定　義

```
public int32_t sarRemoveLandmark(
       const SarLandmark& landmark
       );
```

引　数

[in] landmark　　ランドマーク

返　り　値

エラーコード

解　説

指定したランドマークを削除します。
注意 1: SmartAR™ SDK は、推定に悪影響を及ぼすランドマークを自動的に削除します。
注意 2: この機能は SarRecognizer が SceneMapping モードの場合のみ使用することができます。

SarRecognizer::sarSetDenseMapMode

デンスマップモードの設定

定　義

```
public int32_t sarSetDenseMapMode(
       SarDenseMapMode mode
       );
```

引　数

[in] mode　　デンスマップモード

返　り　値

エラーコード

解　説

デンスマップモードを設定します。

SarTarget

認識対象クラス

定　義

```
#include <SarRecognizer.h>
class SarTarget : SarNonCopyable {
public:
	virtual ~SarTarget() {}
	virtual int32_t sarGetPhysicalSize(SarVector2* size) const=0;
};
```

解　説

認識対象のインタフェース。

関　連　項　目

SarLearnedImageTarget クラス
SarCompoundTarget クラス
SarSceneMapTarget クラス

SarTarget::~SarTarget

SarTarget クラスのデストラクタ

定　義

```
public ~SarTarget();
```

解　説

SarTarget クラスのデストラクタ。

SarTarget::sarGetPhysicalSize

認識対象の物理サイズを取得する

定　義

```
public virtual int32_t sarGetPhysicalSize(
       SarVector2* size
       ) const=0;
```

引　数

[out] size　　認識対象の物理サイズ[m]

返　り　値

エラーコード

解　説

認識対象の物理サイズを取得します。

SarLearnedImageTarget

自然画像の情報を保持した認識対象クラス

定　義

```
#include <SarRecognizer.h>
class SarLearnedImageTarget : public SarTarget {
public:
	SarLearnedImageTarget(SarSmart*, SarStreamIn*, unsigned char *, unsigned
char *);
	virtual ~SarLearnedImageTarget();
	bool sarIsConstructorFailed();

	virtual int32_t sarGetPhysicalSize(SarVector2* size) const;
};
```

解　説

認識対象画像用辞書作成ツールを使用して作成した自然画像の情報を保持した認識対象クラスです。

関　連　項　目

SarTarget クラス

SarLearnedImageTarget::SarLearnedImageTarget

コンストラクタ

定　義

```
public SarLearnedImageTarget(
       SarSmart* smart,
       SarStreamIn* stream,
       unsigned char* customerID = NULL,
       unsigned char* customerKey = NULL
       );
```

引　数

[in] smart	SarSmart クラスのインスタンスへのポインタ
[in] stream	辞書ファイルへのストリーム
[in] customerID	Customer ID
[in] customerKey	Customer Key

解　説

認識対象画像用辞書作成ツールで作成した辞書ファイルをストリームから読み込み初期化を行います。
辞書ファイルの作成時に customerID と customerKey を指定した場合は、同じ値を本関数の *customerID* と *customerKey* に指定してください。

SarLearnedImageTarget::~SarLearnedImageTarget

デストラクタ

定　義

```
public ~SarLearnedImageTarget();
```

解　説

SarLearnedImageTarget クラスのデストラクタ。

SarLearnedImageTarget::sarIsConstructorFailed

コンストラクタのエラー確認

定　義

```
public bool sarIsConstructorFailed();
```

返　り　値

コンストラクタでエラーが発生した場合は true を返し、それ以外は false を返します。

解　説

コンストラクタでエラーが発生したかを確認します。

SarLearnedImageTarget::sarGetPhysicalSize

認識対象の物理サイズを取得する

定　義

```
public virtual int32_t sarGetPhysicalSize(
       SarVector2* size
       ) const=0;
```

引　数

[out] size　　認識対象の物理サイズ[m]

返　り　値

エラーコード

解　説

認識対象の物理サイズを取得します。

SarCompoundTarget

複数の認識対象で構成される認識対象クラス

定　義

```
#include <SarRecognizer.h>
class SarCompoundTarget : public SarTarget {
public:
        SarCompoundTarget(SarSmart*, const SarTarget* const*,
                        const SarChildTargetInfo*, int32_t);
        ~SarCompoundTarget();
        bool sarIsConstructorFailed();

        virtual int32_t sarGetPhysicalSize(SarVector2*) const;
};
```

解　説

複数の認識対象で構成される認識対象クラス。

関　連　項　目

SarTarget クラス

SarCompoundTarget::SarCompoundTarget

コンストラクタ

定　義

```
public SarCompoundTarget(
       SarSmart* smart,
       const SarTarget* const* childTargets,
       const SarChildTargetInfo* childTargetInfos,
       int32_t numChildTargets
       );
```

引　数

[in] smart	SarSmart クラスのインスタンスへのポインタ
[in] childTargets	構成要素となる認識対象を格納した配列へのポインタ
[in] childTargetInfos	各認識対象の情報
[in] numChildTargets	chlildTargets の要素数

解　説

SarCompoundTarget クラスのコンストラクタ。

SarCompoundTarget::~SarCompoundTarget

デストラクタ

定　義

```
public ~SarCompoundTarget();
```

解　説

SarCompoundTarget クラスのデストラクタ。

SarCompoundTarget::sarIsConstructorFailed

コンストラクタのエラー確認

定　義

```
public bool sarIsConstructorFailed();
```

返　り　値

コンストラクタでエラーが発生した場合は true を返し、それ以外は false を返します。

解　説

コンストラクタでエラーが発生したかを確認します。

SarCompoundTarget::sarGetPhysicalSize

認識対象の物理的なサイズを取得する

定　義

```
public virtual int32_t sarGetPhysicalSize(
       SarVector2* size
       ) const=0;
```

引　数

[out] size　　認識対象の物理的なサイズ[m]

返　り　値

エラーコード

解　説

認識対象の物理的なサイズを取得します。

注意:
現在のバージョンでは、このクラスの関数は縦と幅共に 0 のサイズを返却します。

SarSceneMapTarget

シーンマップの情報を保持した認識対象クラス

定　義

```
#include <SarRecognizer.h>
class SarSceneMapTarget : public SarTarget {
public:
       SarSceneMapTarget(SarSmart* smart, SarStreamIn* stream);
       ~SarSceneMapTarget();
        bool sarIsConstructorFailed();

       virtual int32_t sarGetPhysicalSize(SarVector2* size)const;
};
```

解　説

シーンマップの情報を保持した認識対象クラス。
注意:
この SarTarget は SarRecognizer が SceneMapping モードの場合のみ使用することができます。

関　連　項　目

SarTarget クラス

SarSceneMapTarget::SarSceneMapTarget

コンストラクタ

定　義

```
public SarSceneMapTarget(
       SarSmart* smart,
       SarStreamIn* stream
       );
```

引　数

[in] smart	SarSmart クラスのインスタンスへのポインタ
[in] stream	入力ストリーム

解　説

シーンマップファイルを入力ストリームから読み出し初期化を行います。

SarSceneMapTarget::~SarSceneMapTarget

デストラクタ

定　義

```
public ~SarSceneMapTarget();
```

解　説

SarSceneMapTarget クラスのデストラクタ。

SarSceneMapTarget::sarIsConstructorFailed

コンストラクタのエラー確認

定　義

```
public int32_t sarIsConstructorFailed();
```

返　り　値

コンストラクタでエラーが発生した場合は true を返し、それ以外は false を返します。

解　説

コンストラクタでエラーが発生したかを確認します。

SarSceneMapTarget::sarGetPhysicalSize

認識対象の物理サイズを取得する

定　義

```
public virtual int32_t sarGetPhysicalSize(
       SarVector2* size
       ) const=0;
```

引　数

[out] size　　認識対象の物理サイズ[m]

返　り　値

エラーコード

解　説

認識対象の物理サイズを取得します。

注意:
現在のバージョンでは、このクラスの関数は縦と幅共に 0 のサイズを返却します。

SarWorkDispatchedListener

sarRunWorker()実行要求通知を受け取るリスナ

定　義

```
#include <SarRecognizer.h>
class SarWorkDispatchedListener {
public:
	virtual ~WorkdispatchedListener();
	virtual void sarOnWorkDispatched() = 0;
};
```

解　説

sarRunWorker()実行要求通知を受け取るリスナです。

SarWorkDispatchedListener::~SarWorkDispatchedListener

デストラクタ

定　義

```
public ~WorkDispatchedListener();
```

解　説

SarWorkDispatchedListener クラスのデストラクタ。

SarWorkDispatchedListener::sarOnWorkDispatched

sarRunWorker()実行要求通知を受け取る

定　義

```
public virtual void sarOnWorkDispatched() = 0;
```

解　説

sarRunWorker()実行要求通知を受け取ります。

SarRecognitionResultHolder

認識結果を保持するクラス

定　義

```
#include <SarRecognizer.h>
class SarRecognitionResultHolder : SarNonCopyable {
public:
	virtual int32_t sarGetNumResults() const;
	virtual int32_t sarGetResults(SarRecognitionResult*, int32_t) const;
	virtual int32_t sarGetResult(const SarTarget*, SarRecognitionResult*)
const;
};
```

解　説

認識結果を保持するクラス。

SarRecognitionResultHolder::sarGetNumResults

認識結果の数を取得する

定　義

```
public virtual int32_t sarGetNumResults() const = 0;
```

返　り　値

認識結果の数

解　説

SarRecognitionResultHolder が保持する認識結果の数を返します。

SarRecognitionResultHolder::sarGetResults

複数の認識結果を取得する

定　義

```
public virtual int32_t sarGetResults(
       SarRecognitionResult* results,
       Int32_t maxResults
       ) const = 0;
```

引　数

[out] results	認識結果を格納する SarRecognitionResult 型配列へのポインタ
[in] maxResults	取得する認識結果の最大数

返　り　値

正常に終了した場合は取得した数を返し、異常終了した場合はエラーコードを返す。

解　説

複数の認識結果を取得します。

SarRecognitionResultHolder::sarGetResult

特定の認識結果を取得する

定　義

```
public virtual int32_t sarGetResult(
       const SarTarget* target,
       SarRecognitionResult* result
       ) const = 0;
```

引　数

[in] target	取得する認識結果に対応する認識対象
[out] result	返却された認識結果を格納する SarRecognitionResult 構造体へのポインタ

返　り　値

エラーコード

解　説

特定の認識対象に対する認識結果を取得します。

SarRecognizedListener

認識結果更新通知を受け取るリスナ

定　義

```
#include <SarRecognizer.h>
class SarRecognizedListener {
public:
	virtual ~SarRecognizedListener();
	virtual void sarOnRecognized(const SarRecognitionResultHolder&) = 0;
};
```

解　説

認識結果更新通知を受け取るリスナです。

SarRecognizedListener::~SarRecognizedListener

デストラクタ

定　義

```
public virtual ~SarRecognizedListener();
```

解　説

SarRecognizedListener クラスのデストラクタ。

SarRecognizedListener::sarOnRecognized

認識結果更新通知を受け取る

定　義

```
public void sarOnRecognized(
       const SarRecognitionResultHolder& resultHolder
       )=0;
```

引　数

[in] results　　認識結果を保持した SarRecognitionResultHolder

解　説

認識結果更新通知を受け取ります。

SarRecognitionRequest

認識処理の入力データ

定　義

```
#include <SarRecognizer.h>
struct SarRecognitionRequest {
       SarImage image_;
       int64_t timestamp_;
       int32_t numSensorStates_;
       SarSensorState* sensorStates_;

       int32_t numTriangulateMasks_;
       const SarTriangle2* triangulateMasks_;

       SarRecognitionRequest();
};
```

メ　ン　バ

*image_*	認識対象の探索が行われるカメラ画像。 現在のバージョンで指定可能なフォーマットは SAR_IMAGE_FORMAT_L8, SAR_IMAGE_FORMAT_YCRCB420, SAR_IMAGE_FORMAT_YCBCR420 のいずれかです。 また、stride は画像の幅にマッチしている必要があります。
*timestamp_*	カメラ画像のタイムスタンプ
*numSensorStates_*	sensorStates_の要素数
*sensorStates_*	センサー情報が格納された配列へのポインタ。
*numTriangulateMasks_*	triangulateMasks_ の要素数
*triangulateMasks_*	マスク情報として使用する三角パッチの配列。 三角パッチによって指定された領域はマスクされ、認識対象外となります。 三角パッチには画像の幅と高さをそれぞれ1.0とした場合の座標を指定します。 現在のバージョンでは SceneMapping モードでのみ有効です。

解　説

認識処理に必要となる入力データを保持する構造体。
カメラ画像、タイムスタンプ、センサー情報を保持します。
マスクを使用する場合はマスク情報として、三角パッチの配列を設定してください。

SarLandmark

ランドマークに関する情報

定　義

```
#include <SarRecognizer.h>
struct SarLandmark{
	uint32_t id_;
	SarLandmarkState state_;
	SarVector3 position_;
};
```

メ　ン　バ

*id_*	ランドマークを一意に識別する ID
*state_*	ランドマークの状態
*position_*	ランドマークの座標

解　説

ランドマークに関する情報を保持する構造体。
ランドマークは 3 次元の位置が推定されたシーンマップ中の特徴点です。

SarInitPoint

初期化点に関する情報

定　義

```
#include <SarRecognizer.h>
struct SarInitPoint {
	uint32_t id_;
	SarVector2 position_;
};
```

メ　ン　バ

*id_*	初期化点を一意に識別する ID
*position_*	初期化点の座標

解　説

初期化点に関する情報を保持する構造体。
初期化点は 2 次元画像内でトラッキングされる点で、SAR_SCENE_MAPPING_INIT_MODE_SFM と SAR_SCENE_MAPPING_INIT_MODE_HFG、SAR_SCENE_MAPPING_INIT_MODE_VFG の初期化モードで使用されます。

SarRecognitionResult

認識処理の認識結果

定　義

```
#include <SarRecognizer.h>
struct SarRecognitionResult {

        const SarTarget* target_;
        bool isRecognized_;
        SarVector3 position_;
        SarQuaternion rotation_;

        uint64_t timestamp_;

        SarVector3 velocity_;
        SarVector3 angularVelocity_;

        SarTargetTrackingState targetTrackingState_;

        SarSceneMappingState sceneMappingState_;

        int32_t numLandmarks_;
        int32_t maxLandmarks_;
        SarLandmark* landmarks_;

        int32_t numInitPoints_;
        int32_t maxInitPoints_;
        SarInitPoint* initPoints_;

        SarRecognitionResult();
};
```

メ　ン　バ

*target_*	この構造体に格納される認識結果に対応した認識対象へのポインタ
*isRecognized_*	Tracking 中かを示すフラグ値
*position_*	SmartAR 動作デバイスの位置
*rotation_*	SmartAR 動作デバイスの回転
*timestamp_*	認識に使用した画像のタイムスタンプ
*velocity_*	SmartAR 動作デバイスの速度
*angularVelocity_*	SmartAR 動作デバイスの角速度
*targetTrackingState_*	現在の TargetTracking モードの状態
*sceneMappingState_*	現在の SceneMapping モードの状態
*numLandmarks_*	landmarks_に格納されいるランドマークの個数
*maxLandmarks_*	landmarks_に格納可能なランドマークの最大数
*landmarks_*	ランドマークが格納される配列へのポインタ
*numInitPoints_*	initPoint_に格納されている初期化点の個数
*maxInitPoints_*	initPoint_に格納可能な初期化点の最大数
*initPoints_*	初期化点が格納される配列へのポインタ

解　説

特定の認識対象に対する認識結果を保持するする構造体。

SarChildTargetInfo

認識対象の情報

定　義

```
#include <SarRecognizer.h>
struct SarChildTargetInfo{
        SarVector3 position_;
        SarQuaternion rotation_;
};
```

メ　ン　バ

*position_*　認識対象の位置情報
*rotation_*　認識対象の回転情報

解　説

SarCompoundTarget の構成要素となる認識対象の情報を保持する構造体です。
この構造体は SarCompoundTarget クラスを作成する際に使用します。

関　連　項　目

SarCompoundTarget クラス

SarCameraDevice

定数

定数一覧

定義	値	解説
SAR_INVALID_CAMERA_ID	-1	未定義の Camera ID を表す定数値。

列挙値

SarFocusMode

フォーカスモード

定　義

```
#include <SarCameraDevice.h>
enum SarFocusMode {
      SAR_FOCUS_MODE_MANUAL,
      SAR_FOCUS_MODE_CONTINUOUS_AUTO_PICTURE,
      SAR_FOCUS_MODE_CONTINUOUS_AUTO_VIDEO,
      SAR_FOCUS_MODE_EDOF,
      SAR_FOCUS_MODE_FIXED,
      SAR_FOCUS_MODE_INFINITY,
      SAR_FOCUS_MODE_MACRO,
};
```

列　挙　値

名前	解説
SAR_FOCUS_MODE_MANUAL	手動モード
SAR_FOCUS_MODE_CONTINUOUS_AUTO_PICTURE	継続的にフォーカスを合わせるモード。SAR_FOCUS_MODE_CONTINUOUS_AUTO_VIDEO よりも頻繁にオートフォーカスを実行する。
SAR_FOCUS_MODE_CONTINUOUS_AUTO_VIDEO	継続的にフォーカスを合わせるモード
SAR_FOCUS_MODE_EDOF	被写界深度拡張モード
SAR_FOCUS_MODE_FIXED	固定焦点モード
SAR_FOCUS_MODE_INFINITY	無限遠モード
SAR_FOCUS_MODE_MACRO	接写モード

解　説

カメラのフォーカスモードを表す列挙値。

SAR_FOCUS_MODE_MANUAL は、SarCameraDevice::sarRunAutoFocus()が呼び出された時にオートフォーカスを一回実行するモードです。SAR_FOCUS_MODE_CONTINUOUS_AUTO_PICTURE または SAR_FOCUS_MODE_CONTINUOUS_AUTO_VIDEO はカメラが継続的にオートフォーカスを実行し続けます。

SarFlashMode

フラッシュモード

定　義

```
#include <SarCameraDevice.h>
enum SarFlashMode {
       SAR_FLASH_MODE_AUTO,
       SAR_FLASH_MODE_OFF,
       SAR_FLASH_MODE_ON,
       SAR_FLASH_MODE_RED_EYE,
       SAR_FLASH_MODE_TORCH,
};
```

列　挙　値

名前	解説
SAR_FLASH_MODE_AUTO	フラッシュを使用するかを自動で判断する
SAR_FLASH_MODE_OFF	フラッシュを使用しない
SAR_FLASH_MODE_ON	フラッシュを使用する
SAR_FLASH_MODE_RED_EYE	赤目軽減フラッシュ
SAR_FLASH_MODE_TORCH	常にフラッシュを炊く

解　説

カメラのフラッシュモードを表す列挙値。

SarExposureMode

露出補正モード

定　義

```
#include <SarCameraDevice.h>
enum SarExposureMode{
       SAR_EXPOSURE_MODE_MANUAL,
       SAR_EXPOSURE_MODE_CONTINUOUS_AUTO,
};
```

列　挙　値

名前	解説
SAR_EXPOSURE_MODE_MANUAL	手動モード
SAR_EXPOSURE_MODE_CONTINUOUS_AUTO	自動モード

解　説

カメラの露出補正モードを表す列挙値。

SAR_EXPOSURE_MODE_MANUAL は SarCameraDevice::sarRunAutoExposure()が呼び出された時のみ露出補正を行います。SAR_EXPOSURE_MODE_CONTINUOUS_AUTO はカメラが継続的に露出補正を行います。

SarWhiteBalanceMode

ホワイトバランスモード

定　義

```
#include <SarCameraDevice.h>
enum SarWhiteBalanceMode{
       SAR_WHITE_BALANCE_MODE_CONTINUOUS_AUTO,
       SAR_WHITE_BALANCE_MODE_CLOUDY_DAYLIGHT,
       SAR_WHITE_BALANCE_MODE_DAYLIGHT,
       SAR_WHITE_BALANCE_MODE_FLUORESCENT,
       SAR_WHITE_BALANCE_MODE_INCANDESCENT,
       SAR_WHITE_BALANCE_MODE_SHADE,
       SAR_WHITE_BALANCE_MODE_TWILIGHT,
       SAR_WHITE_BALANCE_MODE_WARM_FLUORESCENT,
       SAR_WHITE_BALANCE_MODE_MANUAL,
};
```

列　挙　値

名前	解説
SAR_WHITE_BALANCE_MODE_CONTINUOUS_AUTO	自動モード
SAR_WHITE_BALANCE_MODE_CLOUDY_DAYLIGHT	くもりモード
SAR_WHITE_BALANCE_MODE_DAYLIGHT	太陽光モード
SAR_WHITE_BALANCE_MODE_FLUORESCENT	蛍光灯モード
SAR_WHITE_BALANCE_MODE_INCANDESCENT	白熱灯モード
SAR_WHITE_BALANCE_MODE_SHADE	日陰モード
SAR_WHITE_BALANCE_MODE_TWILIGHT	トワイライトモード
SAR_WHITE_BALANCE_MODE_WARM_FLUORESCENT	(暖色系) 蛍光灯モード
SAR_WHITE_BALANCE_MODE_MANUAL	手動モード

解　説

カメラのホワイトバランスモードを表す列挙値。

SAR_WHITE_BALANCE_MODE_MANUAL は SarCameraDevice::sarRunAutoWhiteBalance()が呼び出された時のみホワイトバランスの補正を行います。SAR_WHITE_BALANCE_MODE_CONTINUOUS_AUTO はカメラが継続的にホワイトバランスの補正を実行します。

SarSceneMode

シーンモード

定　義

```
#include <SarCameraDevice.h>
enum SarSceneMode{
       SAR_SCENE_MODE_ACTION,
       SAR_SCENE_MODE_AUTO,
       SAR_SCENE_MODE_BARCODE,
       SAR_SCENE_MODE_BEACH,
       SAR_SCENE_MODE_CANDLELIGHT,
       SAR_SCENE_MODE_FIREWORKS,
       SAR_SCENE_MODE_LANDSCAPE,
       SAR_SCENE_MODE_NIGHT,
       SAR_SCENE_MODE_NIGHT_PORTRAIT,
       SAR_SCENE_MODE_PARTY,
       SAR_SCENE_MODE_PORTRAIT,
       SAR_SCENE_MODE_SNOW,
       SAR_SCENE_MODE_SPORTS,
       SAR_SCENE_MODE_STEADYPHOTO,
       SAR_SCENE_MODE_SUNSET,
       SAR_SCENE_MODE_THEATRE,
};
```

列　挙　値

名前	解説
SAR_SCENE_MODE_ACTION	アクション
SAR_SCENE_MODE_AUTO	オート
SAR_SCENE_MODE_BARCODE	バーコード
SAR_SCENE_MODE_BEACH	ビーチ
SAR_SCENE_MODE_CANDLELIGHT	蝋燭の光
SAR_SCENE_MODE_FIREWORKS	花火
SAR_SCENE_MODE_LANDSCAPE	ランドスケープ
SAR_SCENE_MODE_NIGHT	夜景
SAR_SCENE_MODE_NIGHT_PORTRAIT	夜景ポートレイト
SAR_SCENE_MODE_PARTY	パーティー
SAR_SCENE_MODE_PORTRAIT	ポートレイト
SAR_SCENE_MODE_SNOW	雪
SAR_SCENE_MODE_SPORTS	スポーツ
SAR_SCENE_MODE_STEADYPHOTO	手ぶれ補正
SAR_SCENE_MODE_SUNSET	夕焼け
SAR_SCENE_MODE_THEATRE	劇場

解　説

カメラのシーンモードを表す列挙値。

クラス

SarCameraDevice

カメラデバイス

定　義

```
#include <SarCameraDevice.h>
class SarCameraDevice : SarNonCopyable {
public:
	static const int SAR_INVALID_CAMERA_ID = -1;

	SarCameraDevice(SarSmart* smart);
	SarCameraDevice(SarSmart* smart, int32_t cameraId, void* nativeDevice =
NULL);
	~SarCameraDevice();
	bool sarIsConstructorFailed();

	// setting
	int32_t sarSetNativeVideoOutput(void* nativeVideoOutput);
	int32_t sarSetVideoImageListener(SarCameraImageListener* listener);
	int32_t sarSetVideoImageSize(int32_t width, int32_t height);
	int32_t sarSetVideoImageFormat(SarImageFormat format);
	int32_t sarSetVideoImageFpsRange(float min, float max);
	int32_t sarSetStillImageListener(SarCameraImageListener* listener);
	int32_t sarSetStillImageSize(int32_t width, int32_t height);
	int32_t sarSetStillImageFormat(SarImageFormat format);
	int32_t sarSetShutterListener(SarCameraShutterListener* listener);
	int32_t sarSetFocusMode(SarFocusMode mode);
	int32_t sarSetFocusAreas(SarCameraArea* areas, int32_t numAreas);
	int32_t sarSetExposureMode(SarExposureMode mode);
	int32_t sarSetExposureAreas(SarCameraArea* areas, int32_t numAreas);
	int32_t sarSetFlashMode(SarFlashMode mode);
	int32_t sarSetWhiteBalanceMode(SarWhiteBalanceMode mode);
	int32_t sarSetSceneMode(SarSceneMode mode);
	int32_t sarSetAutoFocusListener(SarCameraAutoAdjustListener* listener);
	int32_t sarSetAutoExposureListener(SarCameraAutoAdjustListener*
listener);
	int32_t sarSetAutoWhiteBalanceListener(SarCameraAutoAdjustListener*
listener);
	int32_t sarSetErrorListener(SarCameraErrorListener* listener);
	int32_t sarSetOwningNativeDevice(bool isOwning);

	// get info
	static int32_t sarGetDefaultCameraId(SarSmart* smart, SarFacing facing,
	   int32_t* cameraId);
	int32_t sarGetSupportedVideoImageSize(SarSize* sizes, int32_t maxSizes)
const;
	int32_t sarGetSupportedVideoImageFormat(SarImageFormat* formats,
	   int32_t maxFormats) const;
	int32_t sarGetSupportedVideoImageFpsRange(SarCameraFpsRange* ranges,
	   int32_t maxRanges) const;
	int32_t sarGetSupportedStillImageSize(SarSize* sizes, int32_t maxSizes)
const;
	int32_t sarGetSupportedStillImageFormat(SarImageFormat* formats,
	   int32_t maxFormats) const;
```

```
        int32_t sarGetSupportedFocusMode(SarFocusMode* modes, int32_t maxModes)
const;
        int32_t sarGetMaxNumFocusAreas() const;
        int32_t sarGetSupportedFlashMode(SarFlashMode* modes, int32_t maxModes)
const;
        int32_t sarGetSupportedExposureMode(SarExposureMode* modes,
           int32_t maxModes) const;
        int32_t sarGetMaxNumExposureAreas() const;
        int32_t sarGetSupportedWhiteBalanceMode(SarWhiteBalanceMode* modes,
           int32_t maxModes) const;
        int32_t sarGetSupportedSceneMode(SarSceneMode* modes, int32_t maxModes)
const;

        int32_t sarGetDeviceInfo(SarCameraDeviceInfo* info) const;
        int32_t sarGetDeviceInfo(SarCameraDeviceInfo* info, int32_t scaledWidth,
           int32_t scaledHeight, bool isStillImage = false) const;
        int32_t sarGetFovY(float* fovy, float heightRatio = 1.0f,
           bool* calibrated = NULL) const;
        int32_t sarGetFovY(float* fovy, float heightRatio,
          bool* calibrated, bool isStillImage) const;
        int32_t sarGetFovY(float* fovy, int targetWidth, int targetHeight,
           bool* getfromapi = NULL, bool* calibrated = NULL) const;

        int32_t sarGetFacing(SarFacing* facing) const;
        int32_t sarGetRotation(SarRotation* rotation) const;
        int32_t sarGetNativeDevice(void** nativeDevice) const;

        // start and stop
        int32_t sarStart();
        int32_t sarStop();

        // misc
        int32_t sarCaptureStillImage();
        int32_t sarRunAutoFocus();
        int32_t sarRunAutoExposure();
        int32_t sarRunAutoWhiteBalance();
};
```

解　説

SarCameraDevice クラスは、カメラの設定値の変更、ビデオ画像、静止画像のキャプチャーなどを行うことが可能です。キャプチャーされた画像データは SarRecognizer の認識処理の入力データとして使用することができます。

注意:
Android 環境から SarCameraDevice クラスを使用するには以下に示すパーミッションを AndroidManifest.xml に設定する必要があります。

・　<uses-permission android:name="android.permission.CAMERA" />

SarCameraDevice::SarCameraDevice

コンストラクタ

定　義

```
public SarCameraDevice(
       SarSmart* smart
       );

pubiic SarCameraDevice(
       SarSmart* smart,
       int32_t cameraId,
       void* nativeDevice=NULL
       );
```

引　数

[in] smart	SarSmart クラスのインスタンスへのポインタ
[in] cameraId	使用するカメラの ID
[in] nativeDevice	プラットフォーム固有のカメラデバイスオブジェクトへのポインタ

解　説

SarCameraDevice クラスのコンストラクタ。
特定のカメラを使用する場合は ***cameraId*** にカメラ ID を指定してください。
アプリケーションが保持するプラットフォーム固有のカメラデバイスオブジェクトを SarCameraDevice クラスとして使用する場合は ***nativeDevice*** にカメラデバイスオブジェクトへのポインタを指定してください。
指定するカメラデバイスオブジェクトはアプリケーションのプラットフォームにより異なります。
Android 環境では android.hardware.Camera クラスのインスタンスを指定し、iOS 環境では AVCaptureSession クラスのインスタンスを指定してください。

SarCameraDevice::sarIsConstructorFailed

コンストラクタのエラー確認

定　義

```
public bool sarIsConstructorFailed();
```

返　り　値

コンストラクタでエラーが発生した場合は true を返し、それ以外は false を返します。

解　説

コンストラクタでエラーが発生したかを確認します。

SarCameraDevice::sarSetNativeVideoOutput

ビデオの出力先を指定する

定　義

```
public sarSetNativeVideoOutput(
       void* nativeVideoOutput
       );
```

引　数

[in] nativeVideoOutput	プラットフォーム固有のビデオの出力先を示すオブジェクトへのポインタ

返　り　値

エラーコード

解　説

特定の出力先へビデオを描画する場合は *nativeVideoOutput* にプラットフォーム固有なビデオ出力先を示すオブジェクトを指定してください。
Android 環境の場合は android.view.SurfaceView を指定し、iOS 環境の場合は AVCaptureOutput を指定してください。
Android 端末ではビデオの出力先を指定しない場合、正常に動作しないことがあります。
iOS 環境では特に必要がない限り設定する必要はありません。

SarCameraDevice::sarSetVideoImageListener

撮影されたビデオ画像を受け取るリスナを設定する

定　義

```
public int32_t sarSetVideoImageListener(
       SarCameraImageListener* listener
       );
```

引　数

[in] listener　SarCameraImageListener クラスのインスタンスへのポインタ

返　り　値

エラーコード

解　説

撮影されたビデオ画像を受け取るリスナを設定します。

SarCameraDevice::sarSetVideoImageSize

ビデオ画像のサイズを設定する

定　義

```
public int32_t sarSetVideoImageSize(
       int32_t width,
       int32_t height
       );
```

引　数

[in] width	画像の幅
[in] height	画像の高さ

返　り　値

エラーコード

解　説

ビデオ画像のサイズを設定します。

SarCameraDevice::sarSetVideoImageFormat

ビデオ画像のフォーマットを設定する

定　義

```
public int32_t sarSetVideoImageFormat(
       SarImageFormat format
       );
```

引　数

[in] format　　画像のフォーマット

返　り　値

エラーコード

解　説

ビデオ画像のフォーマットを設定します。

SarCameraDevice::sarSetVideoImageFpsRange

ビデオ画像の更新間隔を設定する

定　義

```
public int32_t sarSetVideoImageFpsRange(
       float min,
       float max
       );
```

引　数

[in] min	更新間隔の最小値
[in] max	更新間隔の最大値

返　り　値

エラーコード

解　説

ビデオ画像の更新間隔を設定します。

SarCameraDevice::sarSetStillImageListener

撮影された静止画像を受け取るリスナを設定する

定　義

```
public int32_t sarSetStillImageListener(
       SarCameraImageListener*  listener
       );
```

引　数

[in] listener　SarCameraImageListener クラスのインスタンスへのポインタ

返　り　値

エラーコード

解　説

撮影された静止画像を受け取るリスナを設定します。

SarCameraDevice::sarSetStillImageSize

静止画像のサイズを設定する

定　義

```
public int32_t sarSetStillImageSize(
       int32_t width,
       int32_t height
       );
```

引　数

[in] width	画像の幅
[in] height	画像の高さ

返　り　値

エラーコード

解　説

静止画像のサイズを設定します。

SarCameraDevice::sarSetStillImageFormat

静止画像の画像フォーマットを設定する

定　義

```
public int32_t sarSetStillImageFormat(
       SarImageFormat format
       );
```

引　数

[in] format　　画像フォーマット

返　り　値

エラーコード

解　説

静止画像の画像フォーマットを設定します。

SarCameraDevice::sarSetShutterListener

シャッターの完了通知を受け取るリスナを設定する

定　義

```
public int32_t sarSetShutterListener(
       SarCameraShutterListener* listener
       );
```

引　数

[in] listener　SarCameraShutterListener クラスのインスタンスへのポインタ

返　り　値

エラーコード

解　説

シャッターの完了通知を受け取るリスナを設定します。

SarCameraDevice::sarSetFocusMode

フォーカスモードを設定する

定　義

```
public int32_t sarSetFocusMode(
       SarFocusMode mode
       );
```

引　数

[in] mode　　フォーカスモード

返　り　値

エラーコード

解　説

カメラのフォーカスモードを設定します。

SarCameraDevice::sarSetFocusAreas

オートフォーカス対象エリアを設定する

定　義

```
public int32_t sarSetFocusAreas(
       SarCameraArea* areas,
       int32_t numAreas
       );
```

引　数

[in] areas	オートフォーカス対象エリアが格納される配列へのポインタ
[in] numAreas	*areas* の要素数

返　り　値

エラーコード

解　説

オートフォーカスの対象としたいエリアを設定します。
指定したエリアを解除したい場合、*numAreas* に `0` を指定してください。

SarCameraDevice::sarSetExposureMode

露出補正モードを設定する

定　義

```
public int32_t sarSetExposureMode(
       SarExposureMode mode
       );
```

引　数

[in] mode　　露出補正モード

返　り　値

エラーコード

解　説

カメラの露出補正モードを設定します。

SarCameraDevice::sarSetExposureAreas

露出補正対象エリアを設定する

定　義

```
public int32_t sarSetExposureAreas(
       SarCameraArea* areas,
       int32_t numAreas
       );
```

引　数

[in] areas	露出補正対象エリアが格納されている配列へのポインタ
[in] numAreas	*areas* の要素数

返　り　値

エラーコード

解　説

露出補正の対象としたいエリアを設定します。

SarCameraDevice::sarSetFlashMode

フラッシュモードを設定する

定　義

```
public int32_t sarSetFlashMode(
       SarFlashMode mode
       );
```

引　数

[in] mode　　フラッシュモード

返　り　値

エラーコード

解　説

カメラのフラッシュモードを設定します。

SarCameraDevice::sarSetWhiteBalanceMode

ホワイトバランスを設定する

定　義

```
public int32_t sarSetWhiteBalanceMode(
       SarWhiteBalanceMode mode
       );
```

引　数

[in] mode　　ホワイトバランスモード

返　り　値

エラーコード

解　説

カメラのホワイトバランスモードを設定します。

SarCameraDevice::sarSetSceneMode

シーンモードを設定する

定　義

```
public int32_t sarSetSceneMode(
       SarSceneMode mode
       );
```

引　数

[in] mode　　シーンモード

返　り　値

エラーコード

解　説

カメラのシーンモードを設定します。

SarCameraDevice::sarSetAutoFocusListener

オートフォーカスの完了通知を受け取るリスナを設定する

定　義

```
public int32_t sarSetAutoFocusListener(
       SarCameraAutoAdjustListener* listener
       );
```

引　数

[in] listener　`SarCameraAutoAdjustListener` クラスのインスタンスへのポインタ

返　り　値

エラーコード

解　説

`sarRunAutoFocus()` によるオートフォーカスの完了通知を受け取るリスナを設定します。

SarCameraDevice::sarSetAutoExposureListener

露出補正の完了通知を受け取るリスナを設定する

定　義

```
public int32_t sarSetAutoExposureListener(
       SarCameraAutoAdjustListener* listener
       );
```

引　数

[in] listener　`SarCameraAutoAdjustListener` クラスへのインスタンスへのポインタ

返　り　値

エラーコード

解　説

`sarRunAutoExposure()` による露出補正の完了通知を受け取るリスナを設定します。

SarCameraDevice::sarSetAutoWhiteBalanceListener

ホワイトバランスの補正完了通知を受け取るリスナを設定する

定　義

```
public int32_t sarSetAutoWhiteBalanceListener(
       SarCameraAutoAdjustListener* listener
       );
```

引　数

[in] listener　SarCameraAutoAdjustListener クラスのインスタンスへのポインタ

返　り　値

エラーコード

解　説

sarRunAutoWhiteBalance()による補正の完了通知を受け取るリスナを設定します。

SarCameraDevice::sarSetErrorListener

カメラのエラーを受け取るリスナを設定する

定　義

```
public int32_t sarSetErrorListener(
       SarCameraErrorListener*  listener
       );
```

引　数

[in] listener　SarCameraErrorListener クラスのインスタンスへのポインタ

返　り　値

エラーコード

解　説

カメラのエラーを受け取るリスナを設定します。

SarCameraDevice::sarSetOwningNativeDevice

NativeDevice オブジェクトの所有権を示すフラグを設定する

定　義

```
public int32_t sarSetOwningNativeDevice(
       bool isOwning
       );
```

引　数

[in] isOwning　　所有権を示すフラグ値

返　り　値

エラーコード

解　説

NativeDevice オブジェクトの所有権を示すフラグを設定します。
コンストラクタの引数に指定された NativeDevice オブジェクトへのポインタが NULL の場合、所有権を示すフラグ値はデフォルトで true が設定され、それ以外は false が設定されます。
isOwning に true が設定された場合、SarCameraDevice クラスのデストラクタが呼び出されると NativeDevice オブジェクトも同時に解放されます。

SarCameraDevice::sarGetDefaultCameraId

指定した位置に搭載されたカメラの ID を取得する

定　義

```
public static int32_t sarGetDefaultCameraId(
       SarSmart* smart,
       SarFacing facing,
       int32_t* cameraId
       );
```

引　数

[in] smart	SarSmart クラスのインスタンスへのポインタ
[in] facing	カメラの搭載位置
[out] cameraId	カメラの ID を格納する int32_t 型変数へのポインタ。 該当するカメラが存在しない場合は SAR_INVALID_CAMERA_ID を返す。

返　り　値

エラーコード

解　説

指定した位置に搭載されたカメラの ID を取得します。
該当するカメラが存在しない場合は SAR_INVALID_CAMERA_ID を返します。

SarCameraDevice::sarGetSupportedVideoImageSize

サポートするビデオ画像のサイズを取得する

定　義

```
public int32_t sarGetSupportedVideoImageSize(
       SarSize* sizes,
       int32_t maxSizes
       ) const;
```

引　数

[out] sizes	ビデオ画像のサイズが格納される配列へのポインタ
[in] maxSizes	*sizes* に格納することが可能な要素数

返　り　値

正常終了した場合は、sizes に格納された SarSize 構造体の要素数を返し、異常終了した場合はエラーコードを返す。

解　説

デバイスがサポートするビデオ画像サイズの一覧を取得します。

SarCameraDevice::sarGetSupportedVideoImageFormat

サポートするビデオ画像のフォーマットを取得する

定　義

```
public int32_t sarGetSupportedVideoImageFormat(
       SarImageFormat* formats,
       int32_t maxFormats
       ) const;
```

引　数

[out] formats	ビデオ画像のフォーマットが格納される配列へのポインタ
[in] maxFormats	*formats* に格納することが可能な要素数

返　り　値

正常終了した場合は、formats に格納された SarImageFormat の要素数を返し、異常終了した場合はエラーコードを返す。

解　説

デバイスがサポートする画像フォーマットの一覧を取得します。

SarCameraDevice::sarGetSupportedVideoImageFpsRange

サポートするビデオ画像の更新頻度を取得する

定　義

```
public int32_t sarGetSupportedVideoImageFpsRange(
       SarCameraFpsRange* ranges,
       int32_t maxRanges
       ) const;
```

引　数

[out] ranges	ビデオ画像の更新頻度が格納される配列へのポインタ
[in] maxRanges	*range* に格納することが可能な要素数

返　り　値

正常終了した場合は、ranges に格納された SarCameraFpsRange の要素数を返し、異常終了した場合はエラーコードを返す。

解　説

デバイスがサポートする更新頻度の一覧を取得します。

SarCameraDevice::sarGetSupportedStillImageSize

サポートする静止画像のサイズを取得する

定　義

```
public int32_t sarGetSupportedStillImageSize(
       SarSize* sizes,
       int32_t maxSizes
       ) const;
```

引　数

[out] sizes	静止画像のサイズが格納される配列へのポインタ
[in] maxSizes	*sizes* に格納することが可能な要素数

返　り　値

正常終了した場合は、sizes に格納された SarSize 構造体の要素数を返し、異常終了した場合はエラーコードを返す。

解　説

デバイスがサポートする画像サイズの一覧を取得します。

SarCameraDevice::sarGetSupportedStillImageFormat

サポートする静止画像のフォーマットを取得する

定　義

```
public int32_t sarGetSupportedStillImageFormat(
       SarImageFormat* formats,
       int32_t maxFormats
       ) const;
```

引　数

[out] formats	静止画像フォーマットが格納される配列へのポインタ
[in] maxFormats	*formats* に格納することが可能な要素数

返　り　値

正常終了した場合、formats に格納された SarImageFormat の要素数を返し、異常終了した場合はエラーコードを返す。

解　説

デバイスがサポートするフォーマットの一覧を取得します。

SarCameraDevice::sarGetSupportedFocusMode

サポートするフォーカスモードを取得する

定　義

```
public int32_t sarGetSupportedFocusMode(
       SarFocusMode* modes,
       int32_t maxModes
       ) const;
```

引　数

[out] modes	フォーカスモードが格納される配列へのポインタ
[in] maxModes	*modes* に格納することが可能な要素数

返　り　値

正常終了した場合は、modes に格納された SarFocusMode の要素数を返し、異常終了した場合はエラーコードを返す。

解　説

デバイスがサポートするフォーカスモードの一覧を取得します。

SarCameraDevice::sarGetMaxNumFocusAreas

設定可能なオーフォーカス対象エリアの最大数を取得する

定　義

```
public int32_t sarGetMaxNumFocusAreas() const;
```

返　り　値

正常終了した場合は、エリアの最大個数を返し、異常終了した場合はエラーコードを返す。

解　説

設定可能なオーフォーカス対象エリアの最大数を取得します。

SarCameraDevice::sarGetSupportedFlashMode

サポートするフラッシュモードを取得する

定　義

```
public int32_t sarGetSupportedFlashMode(
       SarFlashMode* modes,
       int32_t maxModes
       ) const;
```

引　数

[out] modes	フラッシュモードが格納される配列へのポインタ
[in] maxSizes	*modes* に格納することが可能な要素数

返　り　値

正常終了した場合は、modes に格納された SarFlashMode の要素数を返し、異常終了した場合はエラーコードを返す。

解　説

デバイスがサポートするフラッシュモードの一覧を取得します。

SarCameraDevice::sarGetSupportedExposureMode

サポートする露出補正モードを取得する

定　義

```
public int32_t sarGetSupportedExposureMode(
       SarExposureMode* modes,
       int32_t maxModes
       ) const;
```

引　数

[out] modes	露出補正モードが格納される配列へのポインタ
[in] maxSizes	*modes* に格納することが可能な要素数

返　り　値

正常終了した場合は、modes に格納された SarExposureMode の要素数を返し、異常終了した場合はエラーコードを返す。

解　説

デバイスがサポートする露出補正モードの一覧を取得します。

SarCameraDevice::sarGetMaxNumExposureAreas

設定可能な露出補正対象エリアの最大数を取得する

定　義

```
public int32_t sarGetMaxNumExposureAreas() const;
```

返　り　値

正常終了した場合は、エリアの最大個数を返し、異常終了した場合はエラーコードを返す。

解　説

設定可能な露出補正対象エリアの最大数を取得します。

SarCameraDevice::sarGetSupportedWhiteBalanceMode

サポートするホワイトバランスモードを取得する

定　義

```
public int32_t sarGetSupportedWhiteBalanceMode(
       SarWhiteBalanceMode* modes,
       int32_t maxModes
       ) const;
```

引　数

[out] modes	ホワイトバランスモードが格納される配列へのポインタ
[in] maxModes	*modes* に格納できる SarWhiteBalanceMode の要素数

返　り　値

正常終了した場合は、modes に格納された SarWhiteBalanceMode の要素数を返し、異常終了した場合はエラーコードを返す。

解　説

デバイスがサポートするホワイトバランスの一覧を取得します。

SarCameraDevice::sarGetSupportedSceneMode

サポートするシーンモードを取得する

定　義

```
public int32_t sarGetSupportedSceneMode(
       SarSceneMode* modes,
       int32_t maxModes
       ) const;
```

引　数

[out] modes	シーンモードが格納される配列へのポインタ
[in] maxModes	*modes* に格納できる SarSceneMode の要素数

返　り　値

正常終了した場合は、modes に格納された SarSceneMode の要素数を返し、異常終了した場合はエラーコードを返す。

解　説

デバイスがサポートするシーンモードの一覧を取得します。

SarCameraDevice::sarGetDeviceInfo

カメラデバイスの情報を取得する

定　義

```
public int32_t sarGetDeviceInfo(
       SarCameraDeviceInfo* info
       ) const;
```

引　数

[out] info　　カメラデバイスの情報

返　り　値

エラーコード

解　説

SarRecognizer が必要とするカメラデバイスの情報を取得します。

SarCameraDevice::sarGetDeviceInfo

カメラデバイスの情報を取得する

定　義

```
public int32_t sarGetDeviceInfo(
       SarCameraDeviceInfo* info,
       int32_t scaledWidth,
       int32_t scaledHeight,
       bool isStillImage = false
       ) const;
```

引　数

[out] info	カメラデバイスの情報
[in] scaledWidth	指定した値を幅として SmartAR で使用するカメラの内部パラメータを変換 -1 のときは現在設定されているサイズを使用
[in] scaledHeight	指定した値を高さとして SmartAR で使用するカメラの内部パラメータを変換 -1 のときは現在設定されているサイズを使用
[in] isStillImage	SmartAR で使用するカメラの内部パラメータの値のモードの指定 true:静止画用、false:動画用

返　り　値

エラーコード

解　説

SarRecognizer が必要とするカメラデバイスの情報を取得します。

SarCameraDevice::sarGetFovY

ディスプレイ表示上の垂直方向の画角を取得する

定　義

```
public int32_t sarGetFovY(
       float* fovy,
       int heightRatio = 1.0f,
       bool* calibrated = NULL
       ) const;
```

引　数

[out] fovy	画角が格納される float 型変数へのポインタ
[in] heightRatio	4:3→16:9 の場合など、カメラ画像の上下を切り詰めた場合の画角を取得したい場合などに切り詰めた高さと元の高さとの比を指定します。
[out] calibrated	SmartAR で保持するカメラキャリブレーション値を使用した結果かどうか

返　り　値

エラーコード

解　説

ディスプレイ表示上の垂直方向の画角を返します。

SarCameraDevice::sarGetFovY

ディスプレイ表示上の垂直方向の画角を取得する

定　義

```
public int32_t sarGetFovY(
       float* fovy,
       float heightRatio,
       bool* calibrated,
       bool isStillImage
) const;
```

引　数

[out] fovy	画角が格納される float 型変数へのポインタ
[in] heightRatio	4:3→16:9 の場合など、カメラ画像の上下を切り詰めた場合の画角を取得したい場合などに切り詰めた高さと元の高さとの比を指定します。
[out] calibrated	SmartAR で保持するカメラキャリブレーション値を使用した結果かどうか
[in] isStillImage	静止画用のカメラ内部パラメータを使うかどうかの指定 (false の場合は動画の値を使用)

返　り　値

エラーコード

解　説

ディスプレイ表示上の垂直方向の画角を返します。

SarCameraDevice::sarGetFovY

ディスプレイ表示上の垂直方向の画角を取得する

定　義

```
public int32_t sarGetFovY(
       float* fovy,
       int targetWidth,
       int targetHeight,
       bool* getfromapi = NULL,
       bool* calibrated = NULL
) const;
```

引　数

[out] fovy	画角が格納される float 型変数へのポインタ
[in] targetWidth	表示するディスプレイサイズの幅
[in] targetHeight	表示するディスプレイサイズの高さ
[out] getfromapi	Android API から取得した値を使用した結果かどうか
[out] calibrated	SmartAR で保持するカメラキャリブレーション値を使用した結果かどうか

返　り　値

エラーコード

解　説

ディスプレイ表示上の垂直方向の画角を返します。
ソニーモバイルコミュニケーションズ株式会社製の Xperia™ Z1 以降に発売された機種では、Android API から取得した値を使います。それ以外の Android 端末では、sarGetFovY(fovy, 1.0f, calibrated, false) として呼び出した結果を返します。
なお、Android 以外のプラットフォームでは常に SAR_ERROR_INVALID_VALUE を返します。

SarCameraDevice::sarGetFacing

カメラの搭載位置を取得する

定　義

```
public int32_t sarGetFacing(
       SarFacing* facing
       ) const;
```

引　数

[out] facing　　カメラの搭載位置を格納する SarFacing 型変数へのポインタ

返　り　値

エラーコード

解　説

SarCameraDevice が使用しているカメラの搭載位置を取得します。

SarCameraDevice::sarGetRotation

カメラ画像の回転角度を取得する

定　義

```
public int32_t sarGetRotation(
       SarRotation* rotation,
       ) const;
```

引　数

[out] rotation　回転角度が格納される SarRotation 型変数へのポインタ

返　り　値

エラーコード

解　説

デバイスに対するカメラ画像の回転角度(時計回り)を取得します。

SarCameraDevice::sarGetNativeDevice

NativeDevice オブジェクトへのポインタを取得する

定　義

```
public int32_t sarGetNativeDevice(
       void** nativeDevice,
       ) const;
```

引　数

[out] nativeDevice　NativeDevice オブジェクトへのポインタを格納するポインタ

返　り　値

エラーコード

解　説

プラットフォーム固有のカメラデバイスオブジェクトを表す NativeDevice オブジェクトへのポインタを取得します。

SarCameraDevice::sarStart

ビデオ画像のキャプチャを開始する

定　義

```
public int32_t  sarStart();
```

返　り　値

エラーコード

解　説

ビデオ画像のキャプチャを開始します。

SarCameraDevice::sarStop

ビデオ画像のキャプチャを停止する

定　義

```
public int32_t  sarStop();
```

返　り　値

エラーコード

解　説

ビデオ画像のキャプチャを停止します。

SarCameraDevice::sarCaptureStillImage

静止画像を撮影する

定　義

```
public int32_t  sarCaptureStillImage();
```

返　り　値

エラーコード

解　説

静止画像の撮影を行います。

SarCameraDevice::sarRunAutoFocus

オートフォーカスを開始する

定　義

```
public int32_t  sarRunAutoFocus();
```

返　り　値

エラーコード

解　説

被写体との距離を自動的に計測しピントを合わせる機能である、オートフォーカスを開始します。
このメソッドを使用しオートフォーカスを実行するにはフォーカスモードとして SAR_FOCUS_MODE_MANUAL が設定されている必要があります。
SAR_FOCUS_MODE_CONTINUOUS_AUTO_PICTURE または SAR_FOCUS_MODE_CONTINUOUS_AUTO_VIDEO が設定されている場合は、カメラによって自動的にオートフォーカスが行われるため、このメソッドを呼び出す必要はありません。

SarCameraDevice::sarRunAutoExposure

露出補正を開始する

定　義

```
public int32_t sarRunAutoExposure();
```

返　り　値

エラーコード

解　説

撮影される画像の明るさのずれを自動的に補正する機能である、自動露出補正を開始します。
このメソッドを使用し自動露出補正を実行するには露出モードとして SAR_EXPOSURE_MODE_MANUAL が設定されている必要があります。
SAR_EXPOSURE_MODE_CONTINUOUS_AUTO が設定されている場合は、カメラによって自動的に露出の補正がおこなわれるため、このメソッドを呼び出す必要はありません。

SarCameraDevice::sarRunAutoWhiteBalance

ホワイトバランスの補正を開始する

定　義

```
public int32_t  sarRunAutoWhiteBalance();
```

返　り　値

エラーコード

解　説

自動的に色調を調整する機能である、ホワイトバランスの自動補正を開始します。
このメソッドを使用しホワイトバランスの自動補正機能を実行するにはホワイトバランスモードとして SAR_WHITE_BALANCE_MODE_MANUAL が設定されている必要があります。
SAR_WHITE_BALANCE_MODE_CONTINUOUS_AUTO が設定されている場合は、カメラによって自動的にホワイトバランスの補正が行われるため、このメソッドを呼び出す必要はありません。

SarCameraDeviceInfo

カメラデバイス情報

定　義

```
#include <SarCameraDevice.h>
class SarCameraDeviceInfo : SarNonCopyable {
public:
	SarCameraDeviceInfo();
	~SarCameraDeviceInfo();

	SarCameraDeviceInfo(const SarCameraDeviceInfo& rhs);
	SarCameraDeviceInfo& operator=(const SarCameraDeviceInfo& rhs);
};
```

解　説

カメラデバイスに関する情報を保持するクラス。

SarImageHolder

画像データを保持するクラス

定　義

```
#include <SarCameraDevice.h>
class SarImageHolder : SarNonCopyable {
public:
	~SarImageHolder();
	int32_t sarGetImageSizeInBytes() const = 0;
	int32_t sarGetImage(SarImage*, int32_t)
};
```

解　説

画像データを保持するクラス。

SarImageHolder::~SarImageHolder

デストラクタ

定　義

```
public ~SarImageHolder();
```

解　説

SarImageHolder クラスのデストラクタ。

SarImageHolder::sarGetImageSizeInBytes

画像サイズの取得

定　義

```
public int32_t sarGetImageSizeInBytes() const;
```

返　り　値

バイト単位での画像サイズ

解　説

SarImageHolder が保持する画像データのサイズをバイト単位で返します。

SarImageHolder::sarGetImage

画像データの取得

定　義

```
public int32_t sarGetImage(
       SarImage* image,
       int32_t maxSizeInBytes
       ) const;
```

引　数

[out] image	取得した画像データを格納する SarImage クラスへのポインタ 注意: SarImage クラスの *ピクセルデータは* ***maxSizeInBytes*** で指定したサイズ以上の領域を確保しておく必要があります。
[in] maxSizeInBytes	返却される SarImage クラスのピクセルデータが指し示す領域のサイズ

返　り　値

エラーコード

解　説

SarImageHolder が保持する画像データを取得します。

SarCameraImageListener

カメラのビデオ画像や静止画像を受け取るリスナ

定　義

```
#include <SarCameraDevice.h>
class SarCameraImageListener {
public:
	virtual ~SarCameraImageListener();
	virtual void sarOnImage(const SarImageHolder&, uint64_t) = 0;
};
```

解　説

カメラのビデオ画像や静止画像を受け取るリスナです。

SarCameraImageListener::~SarCameraImageListener

デストラクタ

定　義

```
public ~SarCameraImageListener();
```

解　説

SarCameraImageListener クラスのデストラクタ。

SarCameraImageListener::sarOnImage

カメラのビデオ画像や静止画像を受け取る

定　義

```
public virtual void sarOnImage(
       const SarImageHolder& imageHolder,
       uint64_t timestamp
       ) = 0;
```

引　数

[in] imageHolder	画像データを保持する SarImageHolder
[in] timestamp	画像のタイムスタンプ

解　説

カメラのビデオ画像や静止画像を受け取ります。

SarCameraShutterListener

シャッターの完了通知を受け取るリスナ

定　義

```
#include <SarCameraDevice.h>
class SarCameraShutterListener {
public:
	virtual ~SarCameraShutterListener();
	virtual void sarOnShutter() = 0;
};
```

解　説

シャッターの完了通知を受け取るリスナです。

SarCameraShutterListener::~SarCameraShutterListener

デストラクタ

定　義

```
public virtual ~SarCameraShutterListener();
```

解　説

SarCameraShutterListener クラスのデストラクタ。

SarCameraShutterListener::sarOnShutter

シャッター終了通知を受け取る

定　義

```
public virtual void sarOnShutter() = 0;
```

解　説

シャッターの完了通知を受け取ります。

SarCameraAutoAdjustListener

自動調整の完了通知を受け取るリスナ

定　義

```
#include <SarCameraDevice.h>
class SarCameraAutoAdjustListener {
public:
	virtual ~SarCameraAutoAdjustListener();
	virtual void sarOnAutoAdjust(bool success) = 0;
};
```

解　説

フォーカスなどの自動調整が完了したことを通知するリスナです。

SarCameraAutoAdjustListener::~SarCameraAutoAdjustListener

デストラクタ

定　義

```
public virtual ~SarCameraAutoAdjustListener();
```

解　説

CameraAutoAdjustListener クラスのデストラクタ。

SarCameraAutoAdjustListener::sarOnAutoAdjust

自動調整の完了通知を受け取る

定　義

```
public virtual void sarOnAutoAdjust(
       bool success
       ) = 0;
```

引　数

[in] success	自動調整が正常に完了したかを示すフラグ値

解　説

フォーカスなどの自動調整処理の完了通知を受け取ります。

SarCameraErrorListener

カメラのエラーを受け取るリスナ

定　義

```
#include <SarCameraDevice.h>
class SarCameraErrorListener {
public:
	virtual ~SarCameraErrorListener();
	virtual void sarOnError(int32_t error) = 0;
};
```

解　説

カメラのエラーを受け取るリスナです。

SarCameraErrorListener::~SarCameraErrorListener

デストラクタ

定　義

```
public ~SarCameraErrorListener();
```

解　説

SarCameraErrorListener クラスのデストラクタ。

SarCameraErrorListener::sarOnError

カメラのエラーを受け取る

定　義

```
public virtual void sarOnError(
       int32_t error
       ) = 0;
```

引　数

[in] error　　エラーコード

解　説

カメラのエラーを受け取ります。

SarCameraFpsRange

カメラの更新間隔

定　義

```
#include <SarCameraDevice.h>
struct SarCameraFpsRange{
       float min_;
       float max_;
};
```

メ　ン　バ

*min_*	更新間隔の下限
*max_*	更新間隔の上限

解　説

更新間隔の範囲を保持する構造体。

SarCameraArea

カメラの調整領域

定　義

```
#include <SarCameraDevice.h>
struct SarCameraArea {
	float left_;
	float top_;
	float right_;
	float bottom_;
	float weight_;
};
```

メ　ン　バ

*left_*	矩形の左端の X 座標
*top_*	矩形の上端の Y 座標
*right_*	矩形の右端の X 座標
*bottom_*	矩形の下端の Y 座標
*weight_*	エリアの重要度

解　説

カメラの調整領域の情報を保持する構造体。
SarCameraArea はオートフォーカスや露出補正を行う対象領域を指定する際に使用します。
*left_*と *top_*で矩形領域の左上の座標を指定し、*right_*と *bottom_*で矩形領域の左下の座標を指定します。指定する矩形領域の座標は 1.0 から-1.0 までの範囲を指定してください。
*weight_*に指定矩形領域へのウェイト値を指定することで、領域ごとに重みづけを行うことができます。
複数の指定矩形領域が重なりあっている場合は、各領域のウェイト値が加算されます。
*weight_*は 0.0 から 1.0 までの範囲を指定してください。

SarSensorDevice

列挙値

SarSensorType

センサータイプ

定　義

```
#include <SarPlatformDef.h>
enum SarSensorType {
      SAR_SENSOR_TYPE_ACCELEROMETER,
      SAR_SENSOR_TYPE_GYROSCOPE,
};
```

列　挙　値

名前	解説
SAR_SENSOR_TYPE_ACCELEROMETER	加速度センサー
SAR_SENSOR_TYPE_GYROSCOPE	ジャイロセンサー

解　説

センサータイプを表す列挙値。

クラス

SarSensorDevice

センサーデバイス

定　義

```
#include <SarSensorDevice.h>
class SarSensorDevice : SarNonCopyable {
public:
	SarSensorDevice(SarSmart* smart, void* nativeDevice = NULL);
	~SarSensorDevice();
	bool sarIsConstructorFailed();

	// setting
	int32_t sarSetSarSensorListener(SarSensorListener* listener);
	int32_t sarSetOwningNativeDevice(bool isOwning);

	// get info
	int32_t sarGetDeviceInfo(SarSensorDeviceInfo* info) const;
	int32_t sarGetNativeDevice(void** nativeDevice) const;

	// sarStart and sarStop
	int32_t sarStart();
	int32_t sarStop();
};
```

解　説

SarSensorDevice クラスはデバイスに搭載されるセンサーから値を取得するためのクラスです。
取得したセンサー情報は SarRecognizer の認識処理の入力データとして使用することが出来ます。

SarSensorDevice::SarSensorDevice

コンストラクタ

定　義

```
public SarSensorDevice(
       SarSmart* smart,
       void* nativeDevice = NULL
       );
```

引　数

[in] smart	SarSmart クラスのインスタンスへのポインタ
[in] nativeDevice	プラットフォーム固有のセンサデバイスオブジェクトへのポインタ

解　説

SarSensorDevice クラスのコンストラクタ。
アプリケーションが保持するセンサデバイスオブジェクトを SarSensorDevice クラスとして使用する場合は ***nativeDevice*** にプラットフォーム固有のセンサデバイスオブジェクトへのポインタを指定してください。
指定するセンサデバイスオブジェクトはアプリケーションのプラットフォームにより異なります。
Android 環境では android.hardware.SensorManager クラスのインスタンスを指定し、iOS 環境では CMMotionManager クラスのインスタンスを指定してください。

SarSensorDevice::~SarSensorDevice

デストラクタ

定　義

```
public ~SarSensorDevice();
```

解　説

SarSensorDevice クラスのデストラクタ。

SarSensorDevice::sarIsConstructorFailed

コンストラクタのエラー確認

定　義

```
public bool sarIsConstructorFailed();
```

返　り　値

コンストラクタでエラーが発生した場合は true を返し、それ以外は false を返します。

解　説

コンストラクタでエラーが発生したかを確認します。

SarSensorDevice::sarSetSarSensorListener

センサー情報を受け取るリスナを設定する

定　義

```
public int32_t sarSetSarSensorListener(
       SarSensorListener* listener
       );
```

引　数

[in] listener	SarSensorListener クラスのインスタンスへのポインタ

返　り　値

エラーコード

解　説

センサー情報を受け取るリスナを設定します。

SarSensorDevice::sarSetOwningNativeDevice

NativeDevice オブジェクトの所有権を示すフラグを設定する

定　義

```
public int32_t sarSetOwningNativeDevice(
       bool isOwning
       );
```

引　数

[in] isOwning	所有権を示すフラグ値

返　り　値

エラーコード

解　説

NativeDevice オブジェクトの所有権を示すフラグを設定します。
コンストラクタの引数に指定された NativeDevice オブジェクトへのポインタが NULL の場合、所有権を示すフラグ値はデフォルトで true が設定され、それ以外は false が設定されます。
isOwning に true が設定された場合、SarSensorDevice クラスのデストラクタが呼び出されると NativeDevice オブジェクトも同時に解放されます。

SarSensorDevice::sarGetDeviceInfo

センサーデバイスに関する情報を取得する

定　義

```
public int32_t sarGetDeviceInfo(
       SarSensorDeviceInfo* info
       ) const;
```

引　数

[out] info　　センサーデバイスの情報

返　り　値

エラーコード

解　説

SarRecognizer が使用するセンサーデバイスに関する情報を取得します。

SarSensorDevice::sarGetNativeDevice

NativeDevice オブジェクトへのポインタを取得する

定　義

```
public int32_t sarGetNativeDevice(
       void** nativeDevice,
       ) const;
```

引　数

[out] nativeDevice　NativeDevice オブジェクトへのポインタを格納するポインタ

返　り　値

エラーコード

解　説

プラットフォーム固有のセンサデバイスオブジェクトを表す NativeDevice オブジェクトへのポインタを取得します。

SarSensorDevice::sarStart

センサーからの値の取得を開始する

定　義

```
public int32_t sarStart();
```

返　り　値

エラーコード

解　説

センサーからの値の取得を開始します。

SarSensorDevice::sarStop

センサーからの値の取得を停止する

定　義

```
public int32_t sarStop();
```

返　り　値

エラーコード

解　説

センサーからの値の取得を停止します。

SarSensorDeviceInfo

センサーデバイス情報

定　義

```
#include <SarSensorDevice.h>
class SarSensorDeviceInfo : SarNonCopyable {
public:
	SarSensorDeviceInfo();
	~SarSensorDeviceInfo();

	SarSensorDeviceInfo(const SarSensorDeviceInfo& );
	SarSensorDeviceInfo& operator=(const SarSensorDeviceInfo& );
};
```

解　説

センサーデバイスに関する情報を保持するクラス。

SarSensorListener

センサー情報を受け取るリスナ

定　義

```
#include <SarSensorDevice.h>
class SarSensorListener {
public:
	virtual ~SarSensorListener();
	virtual void sarOnSensor(const SarSensorState& state) = 0;
};
```

解　説

センサー情報を受け取るリスナです。

SarSensorListener::~SarSensorListener

デストラクタ

定　義

```
public virtual ~SarSensorListener();
```

解　説

SarSensorListener クラスのデストラクタ。

SarSensorListener::sarOnSensor

センサー情報をを受け取る

定　義

```
public virtual void sarOnSensor(
       const SarSensorState& state
       ) = 0;
```

引　数

[in] state　　センサー状態が格納される SarSensorState クラスのインスタンス

解　説

センサー情報を受け取ります。

SarSensorState

センサー情報を保持するクラス

定　義

```
#include <SarSensorDevice.h>
class SarSensorState : SarNonCopyable {
        SarSensorState();
        SarSensorState(const SarSensorState &other);
        ~SarSensorState();
        SarSensorState & operator = (const SarSensorState &other);
};
```

解　説

センサー情報を保持するクラスです。

SarScreenDevice

クラス

SarScreenDevice

スクリーンデバイス

定　義

```
#include <SarScreenDevice.h>
class SarScreenDevice : SarNonCopyable {
public:
        SarScreenDevice (SarSmart* smart);
        ~ SarScreenDevice ();
        int32_t sarIsConstructorFailed();

        int32_t sarGetRotation(SarRotation* rotation);
};
```

解　説

プラットフォームのスクリーンデバイスを抽象化したクラスです。

SarScreenDevice:: SarScreenDevice

コンストラクタ

定　義

```
public SarScreenDevice (
       SarSmart* smart
       );
```

引　数

[in] smart　　SarSmart クラスのインスタンスへのポインタ

解　説

SarScreenDevice クラスのコンストラクタ。

SarScreenDevice::~ SarScreenDevice

デストラクタ

定　義

```
public ~ SarScreenDevice ();
```

解　説

SarScreenDevice クラスのデストラクタ。

SarScreenDevice::sarIsConstructorFailed

コンストラクタのエラー確認

定　義

```
public bool sarIsConstructorFailed();
```

返　り　値

コンストラクタでエラーが発生した場合は true を返し、それ以外は false を返します。

解　説

コンストラクタでエラーが発生したかを確認します。

SarScreenDevice:: sarGetRotation

画面の回転角度を取得する

定　義

```
public int32_t sarGetRotation (
       SarRotation* rotation
       );
```

引　数

[out] rotation　画面の回転角度

返　り　値

エラーコード

解　説

画面の回転角度を取得します。

SarCameraImageDrawer

クラス

SarCameraImageDrawer

カメラ画像の描画ユーティリティ

定　義

```
#include <SarCameraImageDrawer.h>
class SarCameraImageDrawer : SarNonCopyable {
public:
	SarCameraImageDrawer(SarSmart* smart);
	~SarCameraImageDrawer();
	bool sarIsConstructorFailed();

	int32_t sarSetDrawRange(float x1, float y1, float x2, float y2);
	int32_t sarSetRotation(SarRotation rotation);
	int32_t sarSetFlipX(bool flipX);
	int32_t sarSetFlipY(bool flipY);

	int32_t sarStart();
	int32_t sarStop();
	int32_t sarDraw(const SarImage& image);
	int32_t sarDraw(const SarImage& image, const SarRect& rect);
};
```

解　説

カメラ画像をスクリーンへ描画するクラスです。

SarCameraImageDrawer は OpenGLES1.0 と 2.0 の両バージョンに対応しています。

SarCameraImageDrawer::SarCameraImageDrawer

コンストラクタ

定　義

```
public SarCameraImageDrawer(
       SarSmart* smart
       );
```

引　数

[in] smart	SarSmart クラスのインスタンスへのポインタ

解　説

SarCameraImageDrawer クラスのコンストラクタ。

SarCameraImageDrawer::~SarCameraImageDrawer

デストラクタ

定　義

```
public ~SarCameraImageDrawer();
```

解　説

SarCameraImageDrawer クラスのデストラクタ。

SarCameraImageDrawer::sarIsConstructorFailed

コンストラクタのエラー確認

定　義

```
public bool sarIsConstructorFailed();
```

返　り　値

コンストラクタでエラーが発生した場合は true を返し、それ以外は false を返します。

解　説

コンストラクタでエラーが発生したかを確認します。

SarCameraImageDrawer::sarSetDrawRange

描画範囲を設定する

定　義

```
public int32_t sarSetDrawRange(
       float x1,
       float y1,
       float x2,
       float y2
       );
```

引　数

[in] x1	描画範囲の左端のＸ座標
[in] y1	描画範囲の上端のＹ座標
[in] x2	描画範囲の右端のＸ座標
[in] y2	描画範囲の下端のＹ座標

返　り　値

エラーコード

解　説

-1.0 から 1.0 までの範囲でスクリーンの描画領域を設定します。

注意:
描画領域に含まれない領域に対しては本クラスは何も行いません。
必要であれば glClear()を呼び出して、フレームバッファをクリアするなどしてください。

SarCameraImageDrawer::sarSetRotation

カメラ画像を描画する際の回転角度を指定する

定　義

```
public int32_t sarSetRotation(
       SarRotation rotation
       );
```

引　数

[in] rotation　　カメラ画像を描画する際の回転角度

返　り　値

エラーコード

解　説

カメラ画像を描画する際の回転角度を指定します。

SarCameraImageDrawer::sarSetFlipX

カメラ画像を描画する際に水平方向に反転するかどうかを指定する

定　義

```
public int32_t sarSetFlipX (
       bool flipX
       );
```

引　数

[in] flipX	true ならカメラ画像を描画する際に水平方向に反転する

返　り　値

エラーコード

解　説

カメラ画像を描画する際に水平方向に反転するかどうかを指定します。

SarCameraImageDrawer::sarSetFlipY

カメラ画像を描画する際に垂直方向に反転するかどうかを指定する

定　義

```
public int32_t sarSetFlipY (
       bool flipY
       );
```

引　数

[in] flipY　　true ならカメラ画像を描画する際に垂直方向に反転する

返　り　値

エラーコード

解　説

カメラ画像を描画する際に垂直方向に反転するかどうかを指定します。

SarCameraImageDrawer::sarStart

描画処理の初期化

定　義

```
public int32_t sarStart();
```

返　り　値

エラーコード

解　説

カメラ画像の描画を行うために必要となる初期化処理を行います。
注意：このメソッドは OpenGL の機能を使用するため、OpenGL の描画スレッドから呼び出してください。

SarCameraImageDrawer::sarStop

描画処理のリソース解放

定　義

```
public int32_t sarStop();
```

返　り　値

エラーコード

解　説

カメラ画像の描画処理に使用したリソースを解放します。
注意: このメソッドは OpenGL の機能を使用するため、OpenGL の描画スレッドから呼び出してください。

SarCameraImageDrawer::sarDraw

カメラ画像をスクリーンに描画する

定　義

```
public int32_t sarDraw(
       const SarImage& image
       );

public int32_t sarDraw(
       const SarImage& image,
       const SarRect& rect
       );
```

引　数

[in] image	描画する画像データを格納した SarImage クラス。 現在のバージョンで指定可能なフォーマットは SAR_IMAGE_FORMAT_YCRCB420, SAR_IMAGE_FORMAT_YCBCR420 のいずれかです。 また、stride は画像の幅にマッチしている必要があります。
[in] rect	指定領域を格納した SarRect 構造体

返　り　値

エラーコード

解　説

カメラ画像をスクリーンへ描画します。
カメラ画像は sarSetDrawRange()で指定された領域へ描画されます。
カメラ画像の一部のみ描画したい場合は、SarRect 構造体に指定領域を示す座標を格納し、sarDraw(const SarImage&, const SarRect&)を呼び出してください。