



ドリームキャスト プラットフォーム用  
ASR1600/C V2 ローレベル API 仕様書

## 1 目次

ASR1600/C Low Level API Specification for Sega's DreamCast platform .....	1	CasrGetSNR .....	37
1 目次 .....	2	CasrGetSPIVersion .....	38
2 はじめに .....	4	CasrGetState .....	39
2.1 基本機能 .....	4	CasrImportContext .....	40
2.2 注意事項 .....	4	CasrImportData .....	41
3 API 操作の基本 .....	5	CasrOpen .....	42
3.1 定義 .....	5	CasrSetActiveWords .....	43
3.2 API の内部構造 .....	6	CasrSetParam .....	45
3.3 エンジンの初期化 .....	6	CasrSetParamList .....	46
3.4 コンテキスト操作 .....	7	CasrStart .....	47
3.5 クラス .....	7	CasrStop .....	48
3.6 音声認識 .....	7	5.2 追加機能 .....	49
3.7 適応データの継承 .....	9	5.2.1 ユーザー ワード トレーニング関連 .....	49
3.8 自動ゲイン制御 .....	9	CasrAcknowledge .....	49
3.9 メモリ .....	9	CasrAddContextUserWord .....	51
3.10 ユーザー ワード トレーニング .....	9	CasrDeleteContextUserWord .....	53
3.11 スペリング .....	10	CasrStartUserWordTraining .....	54
3.12 単語の登録と削除 .....	10	5.2.2 コンテキスト関連 .....	56
3.13 ユーザー ワードの登録と削除 .....	11	CasrExportContext .....	56
3.14 エクスポートされたシンボルのア クティブ化 .....	12	5.2.3 クラス関連 .....	57
3.15 複数のコンテキストの同時使用 .....	12	CasrCloseClasses .....	57
3.16 音声認識オペレーティングモード と状態遷移図 .....	12	CasrImportClasses .....	58
4 ASR1600 ローレベル API 関数 .....	14	5.2.4 コンテキストとクラスのマージ .....	59
4.1 エンジン関連関数 .....	14	CasrMergeContextsAndClasses .....	59
4.2 コンテキスト関連関数 .....	15	6 コールバック関数 .....	61
4.3 クラス関連関数 .....	15	CBABNORM .....	61
4.4 トレーニング関連関数 .....	16	CBAGC .....	62
4.5 エンジン状態関連関数 .....	16	CBASKCURRENTGAIN .....	63
4.6 その他の関数 .....	16	CBRESULT .....	64
5 ASR1600 関数仕様 .....	17	CBSTATE .....	65
5.1 基本機能 .....	17	CBTRAIN .....	66
CasrAcquisition .....	17	7 OS コールバック関数 .....	67
CasrActivateContext .....	19	CBCREATECRITICALSECTION .....	67
CasrActivateData .....	21	CBDELETECRITICALSECTION .....	68
CasrAPIClose .....	23	CBENTERCRITICALSECTION .....	69
CasrAPIInit .....	24	CBFREE .....	70
CasrClose .....	25	CBGETCURTASK .....	71
CasrCloseContext .....	26	CBGETCURTHREAD .....	72
CasrCloseData .....	27	CBLEAVECRITICALSECTION .....	73
CasrExportData .....	28	CBMALLOC .....	74
CasrGainSet .....	29	CBREALLOC .....	75
CasrGetActivatedContext .....	30	8 データ型、構造体、列挙型定数 .....	76
CasrGetActivatedData .....	31	8.1 データ型 .....	76
CasrGetActiveWords .....	32	BOOL .....	76
CasrGetAPIVersion .....	33	CASRLEVEL .....	76
CasrGetParam .....	34	CPARAMID .....	76
CasrGetParamList .....	35	CPARAMVALUE .....	79
CasrGetSignalLevel .....	36	CWORDID .....	79
		DWORD .....	79
		ERRORID .....	79
		HASR .....	81
		HCONT .....	81
		HCLASSES .....	81
		HCRITSECTION .....	81

HDATA.....	81	PWORD .....	82
PALTERNATIVE.....	81	PWORDBUF .....	82
PCASRLEVEL .....	81	PUSERDATA .....	82
PCASRRESULT .....	81	USERWORDBUF.....	82
PCASRSTATE.....	81	WORD.....	83
PDATA.....	81	8.2 構造体.....	84
PCLASSES.....	81	ALTERNATIVE.....	84
PCOSCALLBACKS .....	81	CASRRESULT .....	84
PCPARAMLIST .....	81	COSCALLBACKS .....	84
PCPARAMVALUE .....	81	CPARAMLIST .....	85
PCRECOGCALLBACKS .....	81	CRECOGCALLBACKS .....	86
PCONT .....	81	DATA.....	86
PCWORDID.....	82	RECWORD .....	88
PDWORD .....	82	SENTENCE.....	88
PHASR .....	82	SPEECHUNITBUF .....	88
PHCONT .....	82	TIMEINFO .....	89
PHCLASSES .....	82	USERDATA.....	89
PHDATA .....	82	WORDBUF.....	89
PRECWORD .....	82	8.3 列举型変数 .....	90
PSENTENCE.....	82	ABNORMCOND .....	90
PSPEECHUNITBUF .....	82	CASRSTATE.....	90
PTIMEINFO .....	82	CSAMPLEFORMAT .....	91
PUSERWORDBUF.....	82	PROMPTYPE.....	91
PVOID.....	82		

## 2 はじめに

ASR1600/C API は、音声認識エンジンを使用するコンシューマ製品を対象に設計された移植可能な音声認識 API です。実際の移植に向けこの API を使用して作業を行う場合は、本書を読むことをお勧めします。この API には、音声認識エンジンの基本的な機能だけを備えたインターフェイスである SPI (サービス プロバイダ インターフェイス) がカプセル化されています。さらにこの SPI には、ユーザー ワードの登録、削除といった機能のほか、コンテキストのエクスポートやクラス の概念<sup>1</sup>等の機能も追加されています。

以下に ASR 1600/C API の機能、および操作面での注意事項を示します。

### 2.1 基本機能

- 音声認識エンジンで同時にアクティブにできるコンテキストは 1 つです。より高水準な API あるいはアプリケーションで、同時に複数のコンテキストをアクティブにする場合は、あらかじめ、これらのコンテキストをマージして、1 つのコンテキストにする必要があります。
- 複数のエンジンを起動可能です。
- 言語の切り替えが可能です。
- 1 つのタスクまたはスレッドが 1 つの音声認識エンジンを制御します。
- 単語のアクティブ化を高速に行うことができます。また、シンボル<sup>1</sup>のアクティブ化を行う場合、単語のアクティブ化によって行なう必要があります。
- 実行時にユーザー ワード トレーニングを行うことができます。
- トレーニングされたユーザー ワードを、異なるコンテキスト間で共有したり、API で、保存することが可能のため、異なるコンテキストにそのユーザー ワードを追加登録することが可能です。
- N-Best 候補認識結果が出力されます。
- 各認識候補結果ごとにその信頼の度合いを示すコンフィデンス値も出力されます。
- スペリング機能は使用できません。

### 2.2 注意事項

- この API ではシステム・コールを直接呼び出していません。必要な OS 機能のすべては一連のコールバック関数の中にカプセル化されており、API を利用する側で用意する必要があります。
- この API では、入力デバイスに依存しません。また、音声信号の取り込みも行ないません。サンプル音声データは API のユーザーが提供します。
- この API では、変更可能なグローバル変数を使用しないので、リンカー生成ファイルを ROM 上に置くことができます。
- すべてのコールバック関数は、アプリケーションで定義したユーザーデータを渡します。これにより、アプリケーションでは特定のヒープ領域に固有の割り当て領域を確保したり、API やエンジンセッションに付随したデータを保存することができます。

---

<sup>1</sup> ASR の概念の詳細については、「音声認識アプリケーション開発ガイド」を参照してください。

### 3 API の基本操作

#### 3.1 定義

##### **API**

アプリケーション プログラマ インターフェイス。アプリケーション、またはより高水準な **API** と音声認識エンジン間のインターフェイスの役割を果たす一連の関数呼び出し。クライアントが利用可能なサービスを提供する。本書で表記している **API** とは **ASR1600/C** とも呼ばれる **ASR1600** ローレベル コンシューマ **API** のことを指します。

##### **SPI**

サービス プロバイダ インターフェイス。基本的な音声認識の機能を実行する一連の関数呼び出し。クライアント、すなわちアプリケーションや高水準な **API** にレコグナイザ等の機能を提供します。

##### コールバック関数

**API** から呼ばれるアプリケーションの関数。**API** 初期化時、アプリケーション関数のポインタが **API** に渡されます。**API** では、メモリ再割り当て要求、新しい認識結果の通知等の必要時にこの関数を呼び出します。

##### グラマー

認識対象のすべての文 (構文 + ボキャブラリ) のテキスト形式の記述文

##### コンテキスト

コンパイルされた文法 (エンジンで認識可能な形式)

##### エンジン

音声認識を実際に行うプログラム

##### ユーザー ワード

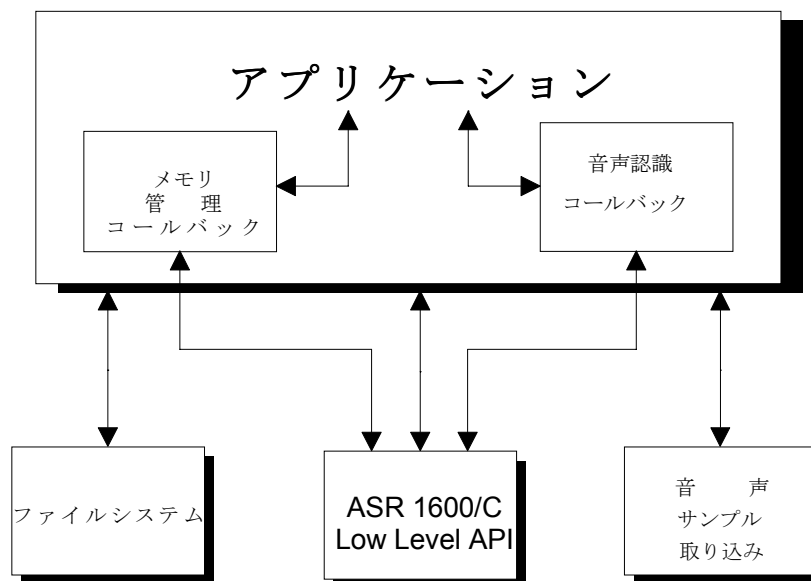
ユーザーが何回かマイクに向け発話すること (トレーニング プロシージャ) によってクラスに登録可能な単語。

### 3.2 API の内部構造

ASR 1600/C ローレベル API は、音声認識エンジンを使用するコンシューマ製品を対象に設計された移植可能な音声認識 API です。この API では、高レベルのコンテキスト/アプリケーション管理をサポートしていないため、異なるアプリケーション間でコンテキストを関連づけることはできません。また、音声の認識結果は、ひとつのアプリケーションだけに戻ってきます。

以下の機能により、ASR1600 ローレベル API の移植性が実現しています。

- この API ではファイルシステム構造を利用していません。すなわち、すべてのデータ (コンテキスト、言語データ、およびユーザーモデル) はデータバッファとしてやり取りされるため、ユーザー側で保存フォーマットを意識する必要はありません。アプリケーション側で管理する必要のあるものは、データのサイズとメモリ割り当てだけになります。
- この API はメモリの確保や開放等のメモリマネジメントを内部では行いません。
- この API では、ほとんどの OS で利用可能な限定された基本的なカプセル化された関数を使用してシステム コールを呼び出します。また、マルチタスク/マルチスレッドをサポートしていないオペレーティング システムでは、マルチタスク/マルチスレッドで定義した関数はすべてダミー関数に置き換えることができます。これらの関数の使用目的は、API タスクやスレッドセーフな処理の生成に限定されます。
- この API では変更可能なグローバル変数を使用しません。(定数変数のみ)



### 3.3 音声認識エンジンの初期化

この API では、ユーザーや言語の概念はありません。レコグナイザに必要なすべてのデータのインポートは、**CasrImportData** 関数呼び出しで、エンジン上のアクティブ化は、**CasrActivateData** 関数呼び出しで行うことができます。

**CasrImportData** 関数 (インポート) に渡す、データバッファは固定にしたり、変更可能にすることができます。固定データの場合は、この変更不可能なデータのポインタが確保され、メモリ上にコピーは生成されません。変更可能なデータの場合は、**Casr Export Data** 関数によって保存することができます。

### 3.4 コンテキスト操作

各コンテキストは、API 管理外のバッファに格納されます。また、このバッファをインポートするには **CasrImportContext** 関数を呼び出します。音声認識を行うには、**CasrActivateContext** 関数を呼び出して、このコンテキストをアクティブ化する必要があります。

ひとつのコンテキストは、別のエンジン上でもアクティブにすることができます。

コンテキストを保存するのに使用されるコンテキストバッファは、アドレスに依存しません。すなわちアプリケーションでこのバッファをメモリ中で移動したり、保存したり、リロードすることができます。

コンテキストは **CasrExportContext** 関数呼び出しにより、後からの再利用のために保存することができます。

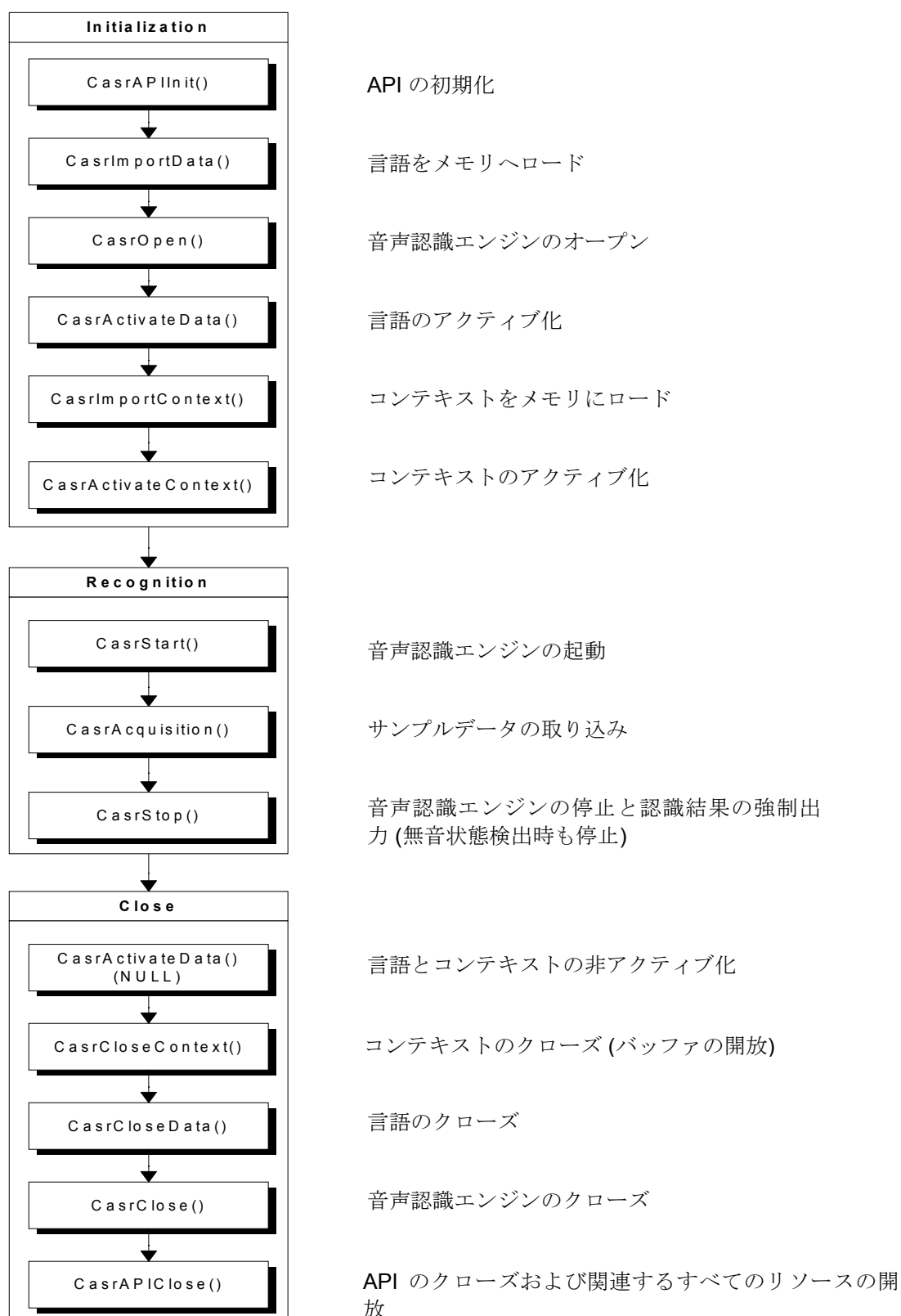
### 3.5 クラス

クラス名データバッファは、API 管理外で保存され、PC ツールにより取得することが可能です。またこのバッファをインポートするには **CasrImportClasses** 関数を呼び出します。

### 3.6 音声認識

音声認識エンジンを起動するには、**CasrStart** 関数を呼び出します。音声入力を待たずにリターンします。エンジンを起動したのち、API 利用者は **Wave** データを保存したバッファを API に渡します。**CasrAcquisition** 関数を呼び出して、API に **Wave** データを含んだバッファを渡します。エンジン側で認識結果取得準備が完了すると、**CBRESULT** コールバック関数が呼ばれます。

## 音声認識時の関数呼び出しの基本的な流れ





### 3.7 適応データの継承

適応データとは、伝走路特性と話者特性に関するエンジンの設定です。これらの特性の値は、レコグナイザ開始時にデフォルトで設定されますが、入力信号を検出すると同時に、再計算(再適応)が自動的に行われます。ASR 1600 の適応データは、**CasrActivateData** 関数にイニシャライズ データを渡してイニシャライズすることができます。その設定可能な適応データを以下に示します。

1. プラットフォーム依存の適応データ。ハードウェアごとに最適化されています。
2. アプリケーションのマイクロフォン適応処理 (プラットフォームでこの処理を行えることが条件) により出力されたデータ。ダミー認識セッションの後、**CasrExportData** 関数を呼び出すと取得できます。
3. 以前の認識セッションのデータ (直前の **CasrExportData** 関数呼び出しで保存されたデータ)
4. デフォルト データ (工場出荷時設定)。**CasrStart** 関数呼び出しの前に **CasrActivateData** 関数呼び出しを行わない場合は、このデフォルト値を使用します。

注 : **CasrActivateData** 関数呼び出し時に指定する場合は、1 番の適応データを使用することをお勧めします。

### 3.8 自動ゲイン制御

ASR1600 エンジンでは、アナログ ゲイン値の調整を行う自動ゲイン制御機能をサポートしています。この機能は、**CPARAM\_AGCON** パラメータに値を設定することで有効になりますが、プラットフォームでアナログ ゲイン値が設定可能であることが条件です。エンジンは、アナログ ゲイン値の調整を行う必要がある場合、**CBAGC** コールバック関数を呼び出します。実際にこの値にゲインを設定するかどうかはアプリケーションに任せられます。アナログ ゲインの新しい値は、**CasrGainSet** 関数を呼び出して API に通知します。

注 : エンジンの起動時には、**CBASKCURRENTGAIN** コールバック関数が、エンジンのアナログ ゲイン値の初期化のために使用されます。

### 3.9 メモリ

API では、内部的なメモリ管理は行わないため、API の使用するメモリはアプリケーション側で準備する必要があります。コンテキストやサンプル バッファを含むバッファ以外に、API の作業領域や音声認識の中間結果を格納するバッファも対象になります。コールバック関数 **CBMALLOC/CBFREE** は、メモリの割り当てや開放時に使用されます。

### 3.10 ユーザー ワード トレーニング

ユーザー ワード トレーニングはコンテキストとは独立して行われます。ユーザー ワード トレーニングの結果はバッファに出力されます。このバッファは、このユーザー ワードをコンテキストに追加したり、**CasrAddContextUserWord** 関数の後で利用するためにアプリケーションで保存できます。

注 : このバッファは、後の格納や再ロード時に何度でも使用できます。

トレーニングを開始するには、**CasrStartUserWordTraining** 関数を呼び出します。エンジン側では、**CBTRAIN (PROMPTTYPE\_START)** コールバック関数を呼び出します。**CasrAcknowledge** 関数呼び出しによって実行されたこのコールバック関数の確認応答が返ってくるまで、エンジンは発話待機状態になります。発話が終了すると (後続無音状態を検出した、あるいは **CasrStop** 関数が呼び出された状態)、**CBTRAIN (PROMPTTYPE\_ACCEPT)** コールバック関数が呼び出され、この発話の受理を依頼します。この受理が受け付けられると、

次の発話依頼、あるいはそれが受理された最後の発話だった場合は **CBTRAIN (PROMPTTYPE\_CONFIRM)** コールバック関数の呼び出しが行われます。**CBTRAIN** コールバック関数の確認応答が **PROMPT\_OK** の場合は、ユーザー ワード モデルの計算処理が行われます。最後に、**CBTRAIN (PROMPTTYPE\_TRAINEND)** コールバック関数が呼び出され、使用可能なユーザー ワード バッファが出力されます。

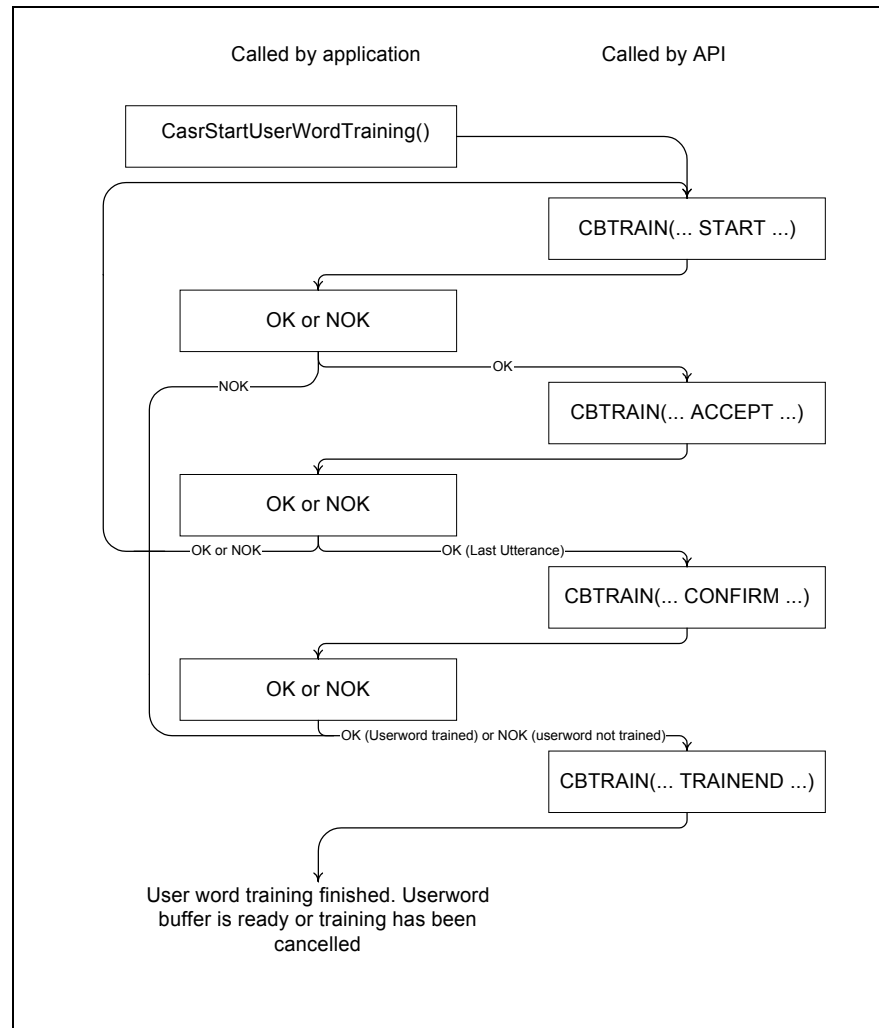


図 1: ユーザー ワード トレーニングの関数呼び出しの流れ

### 3.11 スペリング

スペリング機能は **DreamCast** では使用できません。

### 3.12 単語の登録と削除

この **API** では、コンテキストに含まれる単語の登録/削除をサポートしていません。すなわち、動的に追加できるのはユーザー ワードのみで、話者の依存ではない単語はすべてあらかじめ登録されている必要があります。

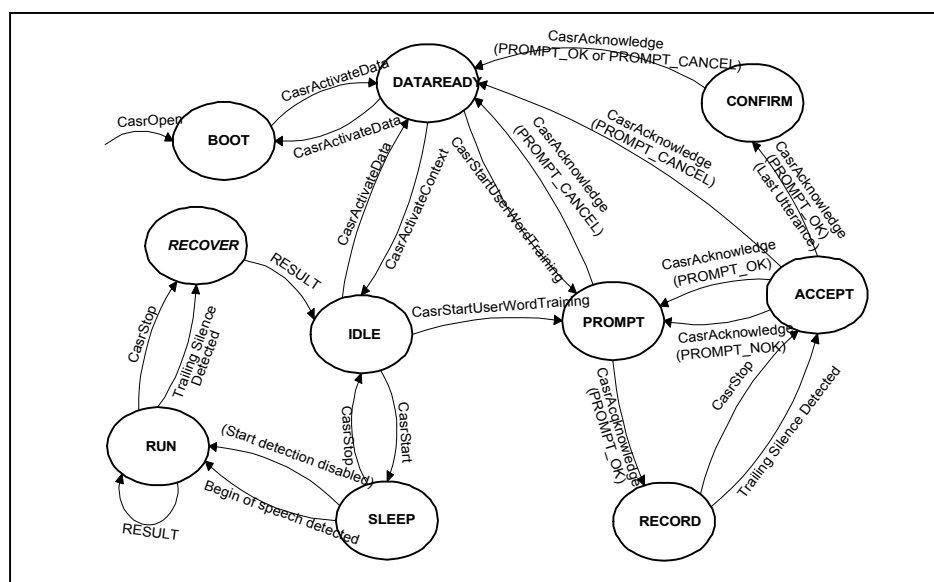
### 3.13 ユーザー ワードの登録と削除

この API では、コンテキストへのユーザー ワードの登録/削除をサポートしています。  
**CasrAddContextUserWord/CasrDeleteContextUserWord** 関数を呼び出すとそれらの登録/削除を行うことができます。また、ユーザー ワードを登録したコンテキストをエクスポートするには **CasrExportContext** 関数を呼び出します。

この API では、BNF グラマーにエクスポートされたシンボルのアクティブ化、および非アクティブ化はサポートしていません。これをアプリケーションで行うには、`CasrSetActiveWords/CasrGetActiveWords` 関数を使用してインプリメントする必要があります。

エンジン上では、同時にアクティブ可能なコンテキストは 1 つに制約されています。より高水準の API あるいはアプリケーションで同時に複数のコンテキストをアクティブにする場合は、`CasrMergeContextsAndClasses` 関数を呼び出して、複数のコンテキストを 1 つのコンテキストにマージする必要があります。

ASR1600 エンジンには、1つのオペレーティングモードがあり、これは起こりうる状態と状態遷移を表わす有限状態図で示すことができます。以下の図では、エンジンのAPI関数呼び出し、あるいは内部的イベントにより発生する状態遷移を示しています。



オペレーティングモードの状態は、以下の 10 個に分類することができます。

**DATAREADY:** エンジン上で必要なデータすべてがアクティブ化された状態。  
コンテキストのアクティブ化、ユーザー ワード トレーニングの開始はこの状態からスタートすることができます。

**SLEEP:** 入力音声を検出されると自動的にアクティブになる CPU 消費量の少ないプロセス。音声開始の検出を無効にすると、この状態は自動的に RUN 状態に遷移します。

**RECOVER**: 音声認識完了状態。認識結果を計算出力します。

**PROMPT:** 次の発話確認依頼状態 (ユーザートレーニング時)。

**RECORD:** 音声取り込み中 (録音状態)。

**ACCEPT:** 直前までの発話をユーザーが受理/棄却するかどうかの確認待ち状態。

**CONFIRM:** ユーザーが行った 3 回の発話を受理するかどうかの確認待ち状態

エンジンは、**BOOT** (初期状態) でオープンします。エンジン上で必要なデータがすべてアクティブになると (**CasrActivateData**)、**DATAREADY** 状態に遷移します。この状態から、音声を認識できるようコンテキストをアクティブ化する (**IDLE** 状態へ遷移) あるいはユーザー ワード トレーニングを開始する (**PROMPT** 状態へ遷移) ことができます。

**CasrStart** 関数を呼び出すと、**IDLE** 状態から **SLEEP** 状態へ遷移します。

**CPARAM\_START\_ENABLE** パラメータの設定によっては、**SLEEP** 状態ではなく、**RUN** 状態に遷移します。**CasrStop** 関数呼び出しが呼び出された、あるいは設定した後続無音状態 (**TS\_ENABLE** が有効になっているのが条件) が検出されると、認識結果を決定する **RECOVER** 状態に遷移します。この状態のとき認識結果がアプリケーション側に通知され、最初の **IDLE** 状態に戻ります。

ユーザー ワード トレーニングのスタート地点である **PROMPT** 状態に遷移するには、

**DATAREADY**/**IDLE** 状態で **CasrStartUserwordTraining** 関数を呼び出します。

**PROMPT** 状態でユーザーが発話を開始可能になると、アプリケーション側にその旨が通知されます。この通知を受けたアプリケーション側ではこのトレーニングを続行するために、**CasrAcknowledge** 関数を呼び出して、この通知に対する確認応答を返す必要があります。アプリケーション側で、**PROMPT** 通知として、**PROMPT\_OK** の確認応答を返すと、現在の **PROMPT** 状態から **RECORD** 状態に遷移します。実際の音声の取り込み、すなわちユーザー ワード の録音はこの **RECORD** 状態になって初めて行われることになります。

ユーザー ワード の録音時に、設定した後続無音状態が検出された、あるいは **CasrStop** 関数が呼び出されると、現在の **RECORD** 状態から **ACCEPT** 状態 (確認待ち状態) に遷移します。この **ACCEPT** 状態では、直前の **RECORD** 状態で録音した音声の受理、棄却 (リジェクト) をユーザーが判断します。どちらを選択しても **PROMPT** 状態に戻りますが、棄却を選択した場合、再度ユーザー ワード の発話をするよう要求されます。一方、受理を選択した場合は、その発話がユーザー ワード 登録の 1 回目の録音と認識され、2 回目の発話を行うよう要求されます。3 回目の録音が終了するとステージは、**ACCEPT** 状態から **CONFIRM** 状態に遷移し、3 回発話したその単語をユーザー ワード として受理するかどうかの問い合わせが行われます。

一連のトレーニング手続きが完了すると、**DATAREADY** 状態に遷移しトレーニング完了を示す **PROMPTTYPE\_TRAINEND** がアプリケーション側に通知されます。このとき初めてアプリケーションから、このユーザー ワード バッファを使用することができます。

**CPARAM\_TS** パラメータの値は発話完了の基準である後続無音状態の継続時間を示しますが、このパラメータ値は内部的に値が固定されていないため、アプリケーション側で設定します。離散単語音声認識やキーワード スポットティング構文の機能を利用しているコマンド制御アプリケーションでは、通常 300 ミリ秒、連続数字構文、または一般的な連続音声グラマーを使用しているアプリケーションでは、500 ~ 1000 ミリ秒の後続無音状態継続時間を必要とします。これは、発話中に、短い息継ぎをしたり、口ごもった場合に発生する無音状態継続時間を、エンジン側で後続無音状態継続時間と誤認するのを防ぐためです。

**CPARAM\_TIMEOUT** パラメータの値は、後続無音状態の最大継続許容時間を示します。騒音の激しい、背景雑音の甚だしい環境、あるいは入力音声の競合が発生している環境では、エンジン側で後続無音状態の識別が困難なため、音声認識時以外は、ユーザー側でエンジンを強制的に終了させる必要があります。

CPARAM\_ENABLEPREMRES パラメータは、音声認識を確認すると、随時その結果を通知する機能を有効にするのに使用します。これは、キーワード スポットティング構文やディクテーションのような構文を使用して音声認識を行うのに有効です。エンジン起動後キーワードが検出されると、認識結果が通知されエンジンは **RUN** 状態で処理を続行します。  
**CasrStop** 関数を呼び出すと、このプロセスは停止します。この機能が有効な場合、**CPARAM\_START\_ENABLE** パラメータの値は、無音状態が続いている間は、CPU 消費量の少ないプロセスとして待機し (**SLEEP** 状態)、音声入力を確認されると、フルに動作する (**RUN** 状態) 機能を使用するかどうかを示すために使用されます。

## 4 ASR1600 ローレベル API 関数

### 4.1 エンジン関連関数

#### **CasrAcquisition**

エンジンにサンプル データを提供します。この関数は、定期的呼び出す必要があります。

#### **CasrActivateData**

エンジンの特定のデータをアクティブ化します。

#### **CasrClose**

エンジンのクローズ

#### **CasrCloseData**

データのクローズ

#### **CasrExportData**

データの変更が発生した場合、そのデータをバッファにエクスポートします。

#### **CasrGainSet**

API にアナログ ゲインの設定値を渡します。

#### **CasrGetActivatedData**

アクティブ化されたデータを取得します。

#### **CasrGetSignalLevel**

現在のサンプル バッファから信号レベルを取得します。

#### **CasrGetSNR**

観測された音声信号の現在の信号対雑音比 (**S/N** 比) を取得します。

#### **CasrImportData**

メモリ上のデータをロードします。永久データへのポインタが渡されます。

#### **CasrOpen**

エンジンをオープンします。(特定のコールバック関数の初期化も含む)

#### **CasrStart**

音声認識エンジンをスタートします。

#### **CasrStop**

エンジンの停止、および認識結果の強制出力をおこないます。

## 4.2 コンテキスト関連関数

### **CasrActivateContext**

エンジン上のコンテキストをアクティブにします。直前までアクティブだったコンテキストは非アクティブになります。

### **CasrAddContextUserWord**

ユーザー ワードをコンテキストに追加します。

### **CasrCloseContext**

コンテキストをクローズします。

### **CasrDeleteContextUserWord**

コンテキストからユーザー ワードを削除します。

### **CasrExportContext**

コンテキストをバッファにエクスポートします。

### **CasrGetActivatedContext**

アクティブ化したコンテキストを取得します。

### **CasrGetActiveWords**

コンテキストのクラスで使用可能な単語を取得します。

### **CasrImportContext**

メモリ上のコンテキストをロードして、ハンドルと関連付けます。

### **CasrMergeContextsAndClasses**

2 つのコンテキスト バッファとそれらに対応するクラス バッファを 1 つのコンテキスト バッファとクラス バッファにマージします。

### **CasrSetActiveWords**

コンテキストの 1 つのクラスの特定の単語を有効にします。

## 4.3 クラス関連関数

### **CasrCloseClasses**

クラス名インスタンスをクローズします。

### **CasrImportClasses**

クラス名データバッファをインポートします。

#### 4.4 トレーニング関連関数

##### **CasrAcknowledge**

ユーザー ワードの発話を受け付け、エンジン側に確認応答信号を送信します。

##### **CasrStartUserWordTraining**

ユーザー ワード トレーニングを初期化し開始します。

#### 4.5 エンジン状態関連関数

##### **CasrGetParam**

エンジン パラメータの現在の値を取得します。

##### **CasrGetParamList**

エンジン パラメータの現在の値を収録したリストを取得します。

##### **CasrGetState**

エンジンの現在の状態を取得します。

##### **CasrSetParam**

特定のエンジン パラメータの値を設定します。

##### **CasrSetParamList**

エンジン パラメータの値すべてに新しい値を設定します。

#### 4.6 その他の関数

##### **CasrAPIClose**

関連したすべてのリソースを開放します。

##### **CasrAPIInit**

API を初期化します。

##### **CasrGetAPIVersion**

API バージョンを取得します。

##### **CasrGetSPIVersion**

SPI バージョンを取得します。



## 5 ASR1600 関数仕様

### 5.1 基本機能

---

#### CasrAcquisition

ERRORID CasrAcquisition(HASR hAsr, CSAMPLEFORMAT SampleFormat, PVOID pSampleBuf, DWORD dwBufSize)

アプリケーションからサンプル バッファを取得し、認識処理を実行します。通知コールバック関数（状態変更、認識結果など）はこの関数から呼び出されます。

#### 引数

**hAsr**

CasrOpen 関数呼び出しによりオープンされたエンジンのハンドル [入力]

**SampleFormat**

pSampleBuf で指定した音声サンプルのフォーマット [入力]

現在使用可能なフォーマットは PCM\_16\_11KHZ のみです。

**pSampleBuf**

サンプルバッファ (音声サンプルを含むバッファ) [入力]

**dwBufSize**

サンプルバッファのサイズ (バイト数) [入力]

#### 戻り値

#### エラーコード

ERR_WRONG_STATE	不適切な状態のエンジンで関数が呼び出されました。 この関数を実行する場合は、エンジンが CASR_RUN、CASR_RECORD、CASR_SLEEP のいずれかの状態である必要があります。
ERR_BADSAMFORMAT	サンプルのフォーマットが有効ではありません。
ERR_NULL_HANDLE	NULL ハンドルが渡されました。
ERR_INVALID_ENGINE	指定したエンジンのハンドルは有効ではありません。 すでにクローズされている可能性があります。
ERR_TASK_THREAD	CasrOpen 関数を呼び出したタスク/スレッドとは別のタスク/スレッドでエンジンのハンドルが使用されています。
ERR_INVALID_SPI	SPI データのハンドルは有効ではないか、そのハンドルを生成したタスクとは別のタスクで使用されています。

この関数から呼び出されるコールバック関数  
エンジンの状態に応じてすべてのコールバック関数が呼び出されます。

関連項目

CasrOpen()、CasrStart()

---

## CasrActivateContext

### ERRORID CasrActivateContext(HASR hAsr, HCONT hCont)

エンジン上の特定のコンテキストをアクティブ化して、そのコンテキスト中のすべての単語を使用可能にします。この関数呼び出しが成功して初めて、音声認識をスタートさせることができます。

#### 引数

##### hAsr

エンジン インスタンスのハンドル [入力]

##### hCont

**CasrImportContext** 関数呼び出しにより返されたコンテキストのハンドル。この **hCont** に **NULL** を指定した場合は、現在アクティブなコンテキストが非アクティブ化されます。  
[入力]

#### 戻り値

#### エラーコード

ERR_WRONG_STATE	不適切な状態のエンジンで関数が呼び出されました。 この関数を実行するには <b>ASR_IDLE</b> 、 <b>CASR_DATAREADY</b> いずれかの状態である必要があります。
ERR_NULL_HANDLE	<b>NULL</b> ハンドルが渡されました。
ERR_INVALID_CONTEXT	インポートされたコンテキストへのハンドルは有効ではありません。すでにクローズされている可能性があります。
ERR_TASK_THREAD_CONTEXT	別のタスク/スレッドで <b>CasrImportContext</b> 関数によって作成されたコンテキストのハンドルです。
ERR_MALLOC	メモリーアロケーションエラー メモリの確保あるいは開放のためにコールバック関数 <b>CBMALLOC/CBFREE</b> 関数が呼ばれましたが、メモリ上のヒープ領域が十分でない可能性があります。
ERR_INVALID_ENGINE	指定したエンジンのハンドルは有効ではありません。 すでにクローズされている可能性があります。
ERR_TASK_THREAD	<b>CasrOpen</b> 関数を呼び出したタスク/スレッドとは別のタスク/スレッドでエンジンのハンドルが使用されています。
ERR_INVALID_SPI	<b>SPI</b> データのハンドルは有効ではないか、そのハンドルを生成したタスクとは別のタスクで使用されています。

この関数から呼び出されるコールバック関数

#### **CBSTATE**

状態変更を通知するのに使用するコールバック関数

関連項目

CasrImportContext()、CasrGetActivatedContext()、CasrSetActiveWords()、  
CasrGetActiveWords()

**ERRORID CasrActivateData(HASR hAsr, HDATA hData)**

エンジン側で使用する特定のデータをアクティブ化します。**ASR 1600** では、言語バッファかセッション データ バッファがそのアクティブ化の対象になります。すでにアクティブ化されたコンテキストがエンジン上に存在する場合は、(別の) 言語バッファをアクティブ化するとそのコンテキストは自動的に非アクティブ化されます。**hData** を **0** に指定してこの関数を呼び出すと、現在アクティブな言語やコンテキストは非アクティブ化され、エンジンは、**BOOT** 状態に遷移します (戻ります)。

**引数****hAsr**

エンジン インスタンスへのハンドル [入力]

**hData****CasrImportData** 関数呼び出しで返されたデータのハンドル [入力]**戻り値****エラーコード****ERR\_WRONG\_STATE**

不適切な状態のエンジンで関数が呼び出されました。  
この関数を実行する場合は、エンジンが **CASR\_IDLE**、**CASR\_DATAREADY**、**CASR\_BOOT** のいずれかの状態である必要があります。

**ERR\_NULL\_HANDLE**

NULL ハンドルが渡されました。

**ERR\_INVALID\_CONTEXT**

インポートされたコンテキストへのハンドルは有効ではありません。すでにクローズされている可能性があります。

**ERR\_TASK\_THREAD\_CONTEXT**

**CasrImportContext** 関数を呼び出したタスク/スレッドとは別のタスク/スレッドでコンテキストのハンドルが使用されています。コールバック関数 **CBGETCURTASK** を呼び出すことで確認できます。

**ERR\_MALLOC**

メモリ アロケーション エラー  
メモリの確保あるいは開放のためにコールバック関数 **CBMALLOC/CBFREE** が呼ばれましたが、メモリ上のヒープ領域が十分でない可能性があります。

**ERR\_INVALID\_ENGINE**

指定したエンジンのハンドルは有効ではありません。  
すでにクローズされている可能性があります。

**ERR\_TASK\_THREAD**

**CasrOpen** 関数を呼び出したタスク/スレッドとは別のタスク/スレッドでエンジンのハンドルが使用されています。

**ERR\_INVALID\_SPI**

SPI データのハンドルは有効ではないか、そのハン

ドルを生成したタスクとは別のタスクで使用されています。

この関数から呼び出されるコールバック関数

#### **CBSTATE**

状態変更を通知するのに使用するコールバック関数

関連項目

CasrImportData()、CasrGetActivatedData()、CasrCloseData()、CasrExportData()

---

## CasrAPIClose

### ERRORID CasrAPIClose(HAPI hApi)

この API で使用したすべてのリソースを開放します。hApi で指定したハンドルはこの関数を実行すると無効になります。この関数呼出し後、他の関数を呼び出すには **CasrAPIInit** 関数を再度呼び出す必要があります。

#### 引数

**hApi**

API インスタンスのハンドル [入力]

#### 戻り値

#### エラーコード

**ERR\_NULL\_HANDLE**

NULL ハンドルが渡されました。

**ERR\_INVALID\_ENGINE**

指定したエンジンのハンドルは有効ではありません。  
すでにクローズされている可能性があります。

**ERR\_INVALID\_SPI**

SPI データのハンドルは有効ではないか、そのハンドルを生成したタスクとは別のタスクで使用されています。

#### 関連項目

**CasrAPIInit()**

---

## CasrAPIInit

ERRORID CasrAPIInit(PCOSCALLBACKS pOsCallbacks, DWORD dwApiUserData, PHAPI phApi)

API で使用される OS コールバックを初期化します。OS コールバックに登録された関数だけが唯一のシステム コールです。API ハンドルは、異なるスレッド間で共有できる唯一のハンドルです。

### 引数

#### pOsCallbacks

すべての OS コールバック関数を含む COSCALLBACKS 構造体へのポインタ。構造体はメモリにコピーされ、スタック上に常駐します。[入力]

#### dwApiUserData

API によって呼び出されるすべてのコールバック関数によって渡される値を示す DWORD 型変数 [入力]

#### phApi

API インスタンスのハンドルが格納されたアドレス[出力]

### 戻り値

#### エラーコード

##### ERR\_NULL\_HANDLE

NULL ハンドルが渡されました。

##### ERR\_MALLOC

メモリ アロケーション エラー  
メモリの確保あるいは開放のためにコールバック関数 CBMALLOC/CBFREE が呼ばれましたが、メモリ上のヒープ領域が十分でない可能性があります。

### 関連項目

CasrAPIClose()、CasrOpen()



---

## CasrClose

### ERRORID CasrClose(HASR hAsr)

エンジンをクローズして、指定した **hAsr** ハンドルを無効にします。また、このエンジンで使用したすべてのリソースを開放します。エンジンは、**BOOT**, **IDLE**, **DATAREADY** のいずれかの状態でなければなりません。この関数は **hAsr** ハンドルを無効にします。

#### 引数

**hAsr**

エンジン インスタンスのハンドル [入力]

#### 戻り値

#### エラーコード

<b>ERR_WRONG_STATE</b>	不適切な状態のエンジンで関数が呼び出されました。 この関数を実行するには、エンジンが <b>CASR_IDLE</b> 、 <b>CASR_DATAREADY</b> 、 <b>CASR_BOOT</b> のいずれかの状態である必要があります。
<b>ERR_NULL_HANDLE</b>	<b>NULL</b> ハンドルが渡されました。
<b>ERR_INVALID_ENGINE</b>	指定したエンジンのハンドルは有効ではありません。 すでにクローズされている可能性があります。
<b>ERR_TASK_THREAD</b>	<b>CasrOpen</b> 関数を呼び出したタスク/スレッドとは別のタスク/スレッドでエンジンのハンドルが使用されています。
<b>ERR_INVALID_SPI</b>	<b>SPI</b> データのハンドルは有効ではないか、そのハンドルを生成したタスクとは別のタスクで使用されています。

#### 関連項目

**CasrOpen()**

**ERRORID CasrCloseContext(HCONT hCont)**

コンテキストをクローズし、**hCont** に指定したハンドルを無効にします。また、使用したすべてのリソースを開放します。**hCont** は、どのエンジンにおいてもアクティブ状態であってはなりません。

## 引数

**hCont**

コンテキストのハンドル [入力]

## 戻り値

## エラーコード

<b>ERR_CONTEXT_IN_USE</b>	指定したコンテキストは別のエンジンで使用中です。
<b>ERR_NULL_HANDLE</b>	<b>NULL</b> ハンドルが渡されました。
<b>ERR_INVALID_ENGINE</b>	指定したエンジンのハンドルは有効ではありません。 すでにクローズされている可能性があります。
<b>ERR_INVALID_SPI</b>	<b>SPI</b> データのハンドルは有効ではないか、そのハンドルを生成したタスクとは別のタスクで使用されています。

## 関連項目

**CasrImportContext()**、**CasrCloseData()**、**CasrActivateContext()**

---

## CasrCloseData

### ERRORID CasrCloseData(HDATA hDATA)

**hData** に指定したデータをクローズし、無効にします  
また、使用したリソースを開放します。無効にした **hData** は、どのエンジンにおいてもアクティブ状態であってはなりません。

#### 引数

##### **hDATA**

データへのハンドル [入力]

#### 戻り値

#### エラーコード

ERR_DATA_IN_USE	指定したインポート データはいずれかのエンジンで使用中です。
ERR_NULL_HANDLE	NULL ハンドルが渡されました。
ERR_INVALID_ENGINE	指定したエンジンのハンドルは有効ではありません。 すでにクローズされている可能性があります。
ERR_INVALID_SPI	SPI データのハンドルは有効ではないか、そのハンドルを生成したタスクとは別のタスクで使用されています。

#### 関連項目

CasrCloseContext()、CasrActivateData()、CasrImportData()、CasrExportData()

---

## CasrExportData

ERRORID CasrExportData(HASR hAsr, DWORD DataType, PVOID \*ppBuf, PDWORD pdwSize)

**DataType** 型のデータをメモリのバッファにエクスポートします。このメモリのバッファはあらかじめ **CBMALLOC** コールバック関数によって割り当てられている必要があります。バッファを開放するには、アプリケーション側から **CBFREE** 関数を呼び出してください。

### 引数

**hAsr**

エンジンインスタンスへのハンドル [入力]

**DataType**

エクスポートするバッファのデータ型。現在は **DATA\_SESSIONDATA** のみが使用可能です。[入力]

**ppBuf**

メモリバッファへのポインタが格納される変数のアドレス [出力]

**pdwSize**

メモリバッファの長さが格納される変数のアドレス [出力]

### 戻り値

#### エラーコード

<b>ERR_WRONG_STATE</b>	不適切な状態のエンジンでこの関数が呼ばれました。この関数を呼び出すには、エンジンが <b>DATAREADY</b> 、 <b>IDLE</b> 、 <b>SLEEP</b> 、 <b>RUN</b> のいずれかの状態である必要があります。
<b>ERR_WRONG_DATA</b>	指定した <b>DataType</b> が無効です。使用可能なデータ型は <b>DATA_SESSIONDATA</b> のみです。
<b>ERR_NULL_HANDLE</b>	<b>NULL</b> ハンドルが渡されました。
<b>ERR_INVALID_ENGINE</b>	指定したエンジンのハンドルは有効ではありません。すでにクローズされている可能性があります。
<b>ERR_TASK_THREAD</b>	<b>CasrOpen</b> 関数を呼び出したタスク/スレッドとは別のタスク/スレッドでエンジンのハンドルが使用されています。
<b>ERR_INVALID_SPI</b>	<b>SPI</b> データのハンドルは有効ではないか、そのハンドルを生成したタスクとは別のタスクで使用されています。

### 関連項目

**CasrImportData()**、**CasrExportData()**、**CasrActivateData()**、**CasrGetActivatedData()**

---

## CasrGainSet

### ERRORID CasrGainSet(HASR hAsr, WORD Gain)

アナログ ゲインの設定値を **API** に渡します。この関数は、**CBAGC/CBASKCURRENTGAIN** 関数に呼応する形で呼び出さなければいけません。音声取り込み時のアナログ ゲイン値を実際に設定するのはアプリケーションの責任です。この関数はエンジン側にアナログ ゲインの値を通知するためだけに使用します。

#### 引数

##### hAsr

エンジン インスタンスのハンドル [入力]

##### Gain

アナログ ゲインの現在の値。指定可能な値は、**1 (最小)** から **65535 (最大)** です。**0** を指定すると、入力した音声が無効になるため、**0** を指定してはいけません。[入力]

#### 戻り値

#### エラーコード

ERR_NULL_HANDLE	NULL ハンドルが渡されました。
ERR_INVALID_ENGINE	指定したエンジンのハンドルは有効ではありません。 すでにクローズされている可能性があります。
ERR_TASK_THREAD	<b>CasrOpen</b> 関数を呼び出したタスク/スレッドとは別のタスク/スレッドでエンジンのハンドルが使用されています。
ERR_INVALID_SPI	<b>SPI</b> データのハンドルは有効ではないか、そのハンドルを生成したタスクとは別のタスクで使用されています。

#### 関連項目

**CBAGC()**、**CBASKCURRENTGAIN()**

---

## CasrGetActivatedContext

ERRORID CasrGetActivatedContext(HASR hAsr, PHCONT phCont)

エンジン上でアクティブ化されたコンテキストのハンドルを返します。  
アクティブなコンテキストが存在しない場合は、\*phCont にハンドル 0 が返されます。

### 引数

**hAsr**

エンジン インスタンスのハンドル [入力]

**phCont**

コンテキスト ハンドルへのポインタ [出力]

### 戻り値

#### エラーコード

ERR_NULL_HANDLE	NULL ハンドルが渡されました。
ERR_WRONG_STATE	アクティブなコンテキストが存在しない状態でこの関数が呼ばれました。
ERR_INVALID_ENGINE	指定したエンジンのハンドルは有効ではありません。 すでにクローズされている可能性があります。
ERR_TASK_THREAD	CasrOpen 関数を呼び出したタスク/スレッドとは別のタスク/スレッドでエンジンのハンドルが使用されています。
ERR_INVALID_SPI	SPI データのハンドルは有効ではないか、そのハンドルを生成したタスクとは別のタスクで使用されています。

### 関連項目

CasrActivateContext()、CasrGetActivatedData()、CasrImportContext()、  
CasrCloseContext()

---

## CasrGetActivatedData

ERRORID CasrGetActivatedData(HASR hAsr, PHDATA \*ppData, PDWORD pdwSize)

アクティブ化されたデータ ハンドルの配列を返します。この関数は、2 つの HDATA ハンドルのメモリを割り当て、**ppData** にそのメモリ アドレスを格納します。(\***ppData**)[0] にはアクティブ化された言語が、(\***ppData**)[1] にはセッション データのハンドルが格納されます。割り当てられたメモリを開放するには、**CBFREE** 関数を呼び出してください。

### 引数

**hAsr**

エンジン インスタンスのハンドル [入力]

**ppData**

割り当てられたバッファへのポインタを受け取る変数のアドレス [出力]

**pdwSize**

配列のサイズが格納された変数のアドレス [出力]

### 戻り値

#### エラーコード

<b>ERR_NULL_HANDLE</b>	NULL ハンドルが渡されました。
<b>ERR_INVALID_ENGINE</b>	指定したエンジンのハンドルは有効ではありません。 すでにクローズされている可能性があります。
<b>ERR_TASK_THREAD</b>	<b>CasrOpen</b> 関数を呼び出したタスク/スレッドとは別のタスク/スレッドでエンジンのハンドルが使用されています。
<b>ERR_INVALID_SPI</b>	<b>SPI</b> データのハンドルは有効ではないか、そのハンドルを生成したタスクとは別のタスクで使用されています。
<b>ERR_MALLOC</b>	メモリ アロケーション エラー メモリの確保あるいは開放のためにコールバック関数 <b>CBMALLOC/CBFREE</b> が呼ばれましたが、メモリ上のヒープ領域が十分でない可能性があります。

### 関連項目

**CasrGetActivatedContext()**、**CasrImportData()**、**CasrActivateData()**、**CasrCloseData()**

---

## CasrGetActiveWords

ERRORID CasrGetActiveWords(HASR hAsr, PCWORDID \*ppActiveWords, PDWORD pdwSize)

エンジンでアクティブな単語を含む配列を返します。エンジンでは **CBMALLOC** コールバック関数を呼び出して単語 **ID** のバッファをメモリに割り当てます。この割り当てたメモリを開放するには、アプリケーション側で **CBFREE** 関数を呼び出してください。

### 引数

**hAsr**

エンジン インスタンスのハンドル [入力]

**ppActiveWords**

アクティブ化された単語の **ID** を含む配列へのポインタが格納された変数のアドレス [出力]

**pdwSize**

**ppActiveWords** の配列のサイズが格納された変数のアドレス [出力]

### 戻り値

#### エラーコード

<b>ERR_WRONG_STATE</b>	不適切な状態のエンジンでこの関数が呼び出されました。この関数を実行するにはエンジンが <b>IDLE</b> 、 <b>SLEEP</b> 、 <b>RUN</b> のいずれかの状態である必要があります。
<b>ERR_NULL_HANDLE</b>	<b>NULL</b> ハンドルが渡されました。
<b>ERR_INVALID_ENGINE</b>	指定したエンジンのハンドルは有効ではありません。すでにクローズされている可能性があります。
<b>ERR_TASK_THREAD</b>	<b>CasrOpen</b> 関数を呼び出したタスク/スレッドとは別のタスク/スレッドでエンジンのハンドルが使用されています。
<b>ERR_INVALID_SPI</b>	<b>SPI</b> データのハンドルは有効ではないか、そのハンドルを生成したタスクとは別のタスクで使用されています。

### 関連項目

**CasrSetActiveWords()**、**CasrGetActivatedContext()**、**CasrActivateContext()**



---

## CasrGetAPIVersion

ERRORID CasrGetAPIVersion(HAPI hApi, PDWORD pdwAPIVersion)

現在の API のバージョンを返します。

### 引数

**hApi**

API インスタンスのハンドル [入力]

**pdwAPIVersion**

API バージョン情報が格納される DWORD 型変数へのポインタ

DWORD 型変数の書式は以下の通りです。

0xMMmmbbbb

-> MM メジャー バージョン ナンバー

-> mm マイナー バージョン ナンバー

-> bbbb ビルド ナンバー

[出力]

### 戻り値

この関数は、常に **ERR\_SUCCES** の戻り値を返します。

### 関連項目

CasrGetSpiVersion()

---

## CasrGetParam

ERRORID CasrGetParam(HASR hAsr, CPARAMID ParamId, PCPARAMVALUE pParamValue)

エンジン パラメータの現在の値を取得します。

### 引数

**hAsr**

エンジン インスタンスのハンドル [入力]

**ParamId**

取得するエンジンパラメータの ID [入力]

**pParamValue**

エンジン パラメータが格納される変数へのポインタ [出力]

### 戻り値

### エラーコード

ERR_UNKNOW_PARAM	範囲外のパラメータが指定されました。
ERR_NULL_HANDLE	NULL ハンドルが渡されました。
ERR_INVALID_ENGINE	指定したエンジンのハンドルは有効ではありません。 すでにクローズされている可能性があります。
ERR_TASK_THREAD	<b>CasrOpen</b> 関数を呼び出したタスク/スレッドとは別のタスク/スレッドでエンジンのハンドルが使用されています。
ERR_INVALID_SPI	<b>SPI</b> データのハンドルは有効ではないか、そのハンドルを生成したタスクとは別のタスクで使用されています。

### 関連項目

CasrSetParam(), CasrGetParamList(), CasrSetParamList()

---

## CasrGetParamList

ERRORID CasrGetParamList(HASR hAsr, PCPARAMLIST pParamList)

エンジン パラメータの設定値のリストを取得します。

### 引数

**hAsr**

エンジン インスタンスのハンドル [入力]

**pParamList**

エンジン パラメータが格納される構造体へのポインタ [出力]

### 戻り値

#### エラーコード

ERR_NULL_HANDLE	NULL ハンドルが渡されました。
ERR_INVALID_ENGINE	指定したエンジンのハンドルは有効ではありません。 すでにクローズされている可能性があります。
ERR_TASK_THREAD	<b>CasrOpen</b> 関数を呼び出したタスク/スレッドとは別のタスク/スレッドでエンジンのハンドルが使用されています。
ERR_INVALID_SPI	SPI データのハンドルは有効ではないか、そのハンドルを生成したタスクとは別のタスクで使用されています。

### メモ

この関数を呼び出す前に、**pParamList** が指すメモリ領域を割り当てておく必要があります。

### 関連項目

**CasrSetParam()**、**CasrGetParam()**、**CasrSetParamList()**

---

## CasrGetSignalLevel

ERRORID CasrGetSignalLevel(HASR hAsr, PCASRLEVEL pCasrLevel)

現在の信号レベルの取得します。信号レベルの値の範囲は、7200 (-72dB) から 1800 (+18dB) です。

### 引数

**hAsr**

エンジン インスタンスのハンドル [入力]

**pCasrLevel**

現在の信号レベルが格納される変数へのポインタ [出力]

### 戻り値

#### エラーコード

**ERR\_NULL\_HANDLE**

NULL ハンドルが渡されました。

**ERR\_INVALID\_ENGINE**

指定したエンジンのハンドルは有効ではありません。  
すでにクローズされている可能性があります。

**ERR\_INVALID\_SPI**

SPI データのハンドルは有効ではないか、そのハンドルを生成したタスクとは別のタスクで使用されています。

### 関連項目

CasrGetSNR()

---

## CasrGetSNR

BOOL casrGetSNR(HASR hAsr, CASRLEVEL \*pSNR)

今まで観測された音声信号の現在の **SNR (S/N 比)** を取得します。このレベルは **0 (0dB)** から **5000 (50dB)** の間の値になります。

引数:

hAsr

エンジン インスタンスのハンドル [入力]

pSNR

SNR の値が格納される変数へのポインタ [出力]

戻り値:

エラーコード

ERR\_NULL\_HANDLE

NULL ハンドルが渡されました。

ERR\_INVALID\_ENGINE

指定したエンジンのハンドルは有効ではありません。  
すでにクローズされている可能性があります。

ERR\_TASK\_THREAD

**CasrOpen** 関数を呼び出したタスク/スレッドとは別のタスク/スレッドでエンジンのハンドルが使用されています。

ERR\_INVALID\_SPI

**SPI** データのハンドルは有効ではないか、そのハンドルを生成したタスクとは別のタスクで使用されています。

関連項目

GetSignalLevel()

---

## CasrGetSPIVersion

ERRORID CasrGetSPIVersion(HAPI hApi, PDWORD pdwSPIVersion)

現在の SPI のバージョンを返します。

### 引数

**hApi**

API インスタンスのハンドル [入力]

**pdwSPIVersion**

SPI バージョン情報が格納された DWORD 型変数へのポインタ

DWORD 型変数の書式は以下の通りです。

0xMMmmbbbb

-> MM メジャー バージョン ナンバー

-> mm マイナー バージョン ナンバー

-> bbbb ビルド ナンバー

[出力]

### 戻り値

エラーコード

### 関連項目

CasrGetAPIVersion()

---

## CasrGetState

ERRORID CasrGetState(HASR hAsr, PCASRSTATE pState)

現在のエンジンの状態を取得します。

### 引数

**hAsr**

エンジン インスタンスのハンドル [入力]

**pState**

現在のエンジン状態が格納される変数へのポインタ [出力]

### 戻り値

#### エラーコード

**ERR\_NULL\_HANDLE**

NULL ハンドルが渡されました。

**ERR\_INVALID\_ENGINE**

指定したエンジンのハンドルは有効ではありません。  
すでにクローズされている可能性があります。

**ERR\_INVALID\_SPI**

SPI データのハンドルは有効ではないか、そのハンドルを生成したタスクとは別のタスクで使用されています。

### 関連項目

CasrOpen()、CasrStart()、CBSTATE()

---

## CasrImportContext

ERRORID CasrImportContext(HAPI hApi, PCONT pCont, BOOL IsPermanent, PHCONT phCont)

コンテキストをインポートして、ハンドルと関連付けます。このコンテキストは異なるエンジン インスタンス間でアクティブ化することができます。この場合、少ないメモリ オーバーヘッドは最小限で済みます。同じコンテキストを 2 回インポートしてもデータは一度だけコピーされます。インポートしたコンテキストを以後使用しない場合は、**CasrCloseContext** 関数を呼び出してリソースを開放します。

### 引数

#### hApi

API インスタンスのハンドル [入力]

#### pCont

コンテキストを含む読取専用のバッファ。このデータは、異なるエンジン インスタンスで共有することが可能です。[入力]

#### IsPermanent

**pCont** に指定したバッファを「固定」(処理中はメモリ上に値が保持される) にする場合は **TRUE** を指定します。このバッファは、バッファ内のコンテキストの **HCONT** が有効である限り使用することができます。**FALSE** を指定すると、**CBMALLOC** 関数によってバッファが新たに確保され、指定されたデータ バッファはそこにコピーされます。[入力]

#### phCont

コンテキスト ハンドルのアドレス。このコンテキストは異なるエンジン上で使用可能です。[出力]

### 戻り値

### エラーコード

ERR_WRONG_DATA	指定したデータバッファは、コンテキストバッファではありません。
ERR_NULL_HANDLE	NULL ハンドルが渡されました。
ERR_INVALID_ENGINE	指定したエンジンのハンドルは有効ではありません。すでにクローズされている可能性があります。
ERR_TASK_THREAD	<b>CasrOpen</b> 関数を呼び出したタスク/スレッドとは別のタスク/スレッドでエンジンのハンドルが使用されています。
ERR_INVALID_SPI	<b>SPI</b> データのハンドルは有効ではないか、そのハンドルを生成したタスクとは別のタスクで使用されています。

### 関連項目

**CasrImportData()**、**CasrExportContext()**、**CasrActivateContext()**



---

## CasrImportData

ERRORID CasrImportData(HAPI hApi, PDATA pData, BOOL IsPermanent, PHDATA phData)

データをメモリにロードし、ハンドルと関連付けます。この関数呼び出しが成功すると、ハンドルと関連付けられたこのデータは、異なるエンジン インスタンスで、少ないメモリ オーバーヘッドでアクティブ化することができます。同じデータを 2 回インポートしてもデータは一度だけコピーされます。インポートしたデータを以後使用しない場合は、**CasrCloseData** 関数を呼び出してリソースを開放します。

### 引数

#### hApi

API インスタンスへのハンドル [入力]

#### pData

データを含むバッファ。このデータは異なるエンジンのインスタンス間で共有することができます。ただしデータ型によっては共有できないデータもあります。[入力]

#### IsPermanent

**pData** に指定したバッファを「固定」(処理中はメモリ上に値が保持される)にする場合は **TRUE** を指定します。このバッファは **HDATA** が有効である限り使用することができます。**IsPermanent** を **FALSE** に指定すると、メモリが割り当てられ、そのバッファデータがコピーされます。エンジンで変更可能なデータ (ユーザー特定モデル、データ継承型) は、固定にはできません。**CasrExportData** 関数はその変更したデータを保存するのに使用します [入力]。

#### phData

**pData** に関連づけられたハンドルへのポインタ。このデータは、異なるエンジン上で使用することができます [出力]。

### 戻り値

#### エラーコード

ERR_NULL_HANDLE	NULL ハンドルが渡されました。
ERR_INVALID_ENGINE	指定したエンジンのハンドルは有効ではありません。 すでにクローズされている可能性があります。
ERR_TASK_THREAD	<b>CasrOpen</b> 関数を呼び出したタスク/スレッドとは別のタスク/スレッドでエンジンのハンドルが使用されています。
ERR_INVALID_SPI	<b>SPI</b> データのハンドルは有効ではないか、そのハンドルを生成したタスクとは別のタスクで使用されています。

### 関連項目

**CasrImportContext()**、**CasrExportData()**、**CasrActivateData()**

---

## CasrOpen

ERRORID CasrOpen(HAPI hApi, PCRECOGCALLBACKS pRecogCallbacks, DWORD dwUserData, PHASR phAsr)

エンジンのオープンと初期化を行います。

### 引数

#### hApi

API インスタンスのハンドル [入力]

#### pRecogCallbacks

このエンジンで使用可能な一連のコールバック関数を含む構造体へのポインタ。この構造体は、この **CasrOpen** 関数によってコピーされるので、メモリに常駐させる必要はありません [入力]。

#### dwUserData

アプリケーション側に情報を通知するために使用する、コールバック関数から返される **DWORD** 型変数。これはユーザー データを特定のエンジン インスタンスに割り当てるのに使用します [入力]。

#### phAsr

指定するエンジン インスタンスが格納されるハンドルへのポインタ [出力]

### 戻り値

### エラーコード

ERR_NULL_HANDLE	NULL ハンドルが渡されました。
ERR_INVALID_ENGINE	指定したエンジンのハンドルは有効ではありません。 すでにクローズされている可能性があります。
ERR_TASK_THREAD	<b>CasrOpen</b> 関数を呼び出したタスク/スレッドとは別のタスク/スレッドでエンジンのハンドルが使用されています。
ERR_INVALID_SPI	SPI データのハンドルは有効ではないか、そのハンドルを生成したタスクとは別のタスクで使用されています。

この関数から呼び出されるコールバック関数

#### CBSTATE

状態変更を通知するのに使用するコールバック関数。

#### CBASKCURRENTGAIN

アナログ ゲインの現在の設定値の取得依頼を行うコールバック関数。

### 関連項目

**CasrClose()**、**CasrAPIInit()**、**CasrAPIClose()**

---

## CasrSetActiveWords

ERRORID CasrSetActiveWords(HASR hAsr, PCWORDID pidWords, DWORD NbrWords)

1 つのコンテキストの特定の単語を使用可能または使用不可にします。現在アクティブなボキャブラリを別のボキャブラリに切り替える場合、この関数呼び出しを実行すると高速に行えます。

### 引数

#### hAsr

エンジン インスタンスのハンドル [入力]

#### pidWords

使用可能にする単語すべての ID を含む配列。ただし、この配列中に存在しない単語は、使用不可になります。[入力]

#### NbrWords

pidWords の単語 ID の数 [入力]

### 戻り値

### エラーコード

ERR_WRONG_STATE	不適切な状態のエンジンで関数が呼び出されました。 この関数を実行するには、エンジンが <b>CASR_IDLE</b> 、 <b>CASR_DATAREADY</b> のいずれかの状態である必要があります。
ERR_NULL_HANDLE	NULL ハンドルが渡されました。
ERR_INVALID_CONTEXT	インポートされたコンテキストへのハンドルは有効ではありません。すでにクローズされている可能性があります。
ERR_TASK_THREAD_CONTEXT	<b>CasrImportContext</b> 関数を呼び出したタスク/スレッドとは別のタスク/スレッドでこの関数がよばれました。
ERR_MALLOC	メモリ アロケーションエラー エンジンのダイナミック データのメモリへの格納あるいは開放のためにコールバック関数 <b>CBMALLOC/CBFREE</b> が呼ばれましたが、メモリ領域が十分でない可能性があります。
ERR_INVALID_ENGINE	指定したエンジンのハンドルは有効ではありません。 すでにクローズされている可能性があります。
ERR_TASK_THREAD	<b>CasrOpen</b> 関数を呼び出したタスク/スレッドとは別のタスク/スレッドでエンジンのハンドルが使用されています。

ERR\_INVALID\_SPI

SPI データのハンドルは有効ではないか、そのハンドルを生成したタスクとは別のタスクで使用されています。

関連項目

CasrActivateContext()、CasrImportContext()、CasrGetActiveWords()

---

## CasrSetParam

ERRORID CasrSetParam(HASR hAsr, CPARAMID ParamId, CPARAMVALUE ParamValue)

特定のエンジンパラメータの値を設定します。

### 引数

**hAsr**

エンジン インスタンスのハンドル [入力]

**ParamId**

設定するエンジン パラメータの ID [入力]

**ParamValue**

設定するエンジン パラメータの値 [入力]

### 戻り値

#### エラーコード

ERR_WRONG_STATE	不適切な状態のエンジンでこの関数が呼び出されました。この関数を実行するには、エンジンがCASR_IDLE、CASR_DATAREADY、CASR_BOOT、CASR_RECOVER のいずれかの状態である必要があります。
ERR_NULL_HANDLE	NULL ハンドルが渡されました。
ERR_INVALID_ENGINE	指定したエンジンのハンドルは有効ではありません。すでにクローズされている可能性があります。
ERR_TASK_THREAD	CasrOpen 関数を呼び出したタスク/スレッドとは別のタスク/スレッドでエンジンのハンドルが使用されています。
ERR_INVALID_SPI	SPI データのハンドルは有効ではないか、そのハンドルを生成したタスクとは別のタスクで使用されています。
ERR_UNKNOW_PARAM	範囲外のパラメータが指定されました。

### 関連項目

CasrOpen()、CasrSetParamList()、CasrGetParam()、CasrGetParamList()

---

## CasrSetParamList

ERRORID CasrSetParamList(HASR hAsr, PCPARAMLIST pParamList)

すべてのパラメータを初期化します。pParamList に指定する値は、以前のセッションにおける CasrGetParamList 関数呼び出しにより取得できます。

### 引数

hAsr

エンジン インスタンスのハンドル [入力]

pParamList

新しく設定するエンジンパラメータの値を含む構造体へのポインタ [入力]

### 戻り値

### エラーコード

ERR\_WRONG\_STATE

不適切な状態のエンジンでこの関数が呼ばれました。  
この関数を呼び出すにはエンジンが CASR\_IDLE、CASR\_DATAREADY、CASR\_BOOT のいずれかの状態である必要があります。

ERR\_NULL\_HANDLE

NULL ハンドルが渡されました。

ERR\_INVALID\_ENGINE

指定したエンジンのハンドルは有効ではありません。  
すでにクローズされている可能性があります。

ERR\_TASK\_THREAD

CasrOpen 関数を呼び出したタスク/スレッドとは別のタスク/スレッドでエンジンのハンドルが使用されています。

ERR\_INVALID\_SPI

SPI データのハンドルは有効ではないか、そのハンドルを生成したタスクとは別のタスクで使用されています。

### 関連項目

CasrGetParamList()、CasrGetParam()、CasrSetParam()

---

## CasrStart

### ERRORID CasrStart(HASR hAsr)

音声認識エンジンを起動します。この関数を呼び出して初めて音声認識可能状態になり、アプリケーションへの通知が開始されます。この時エンジンは **SLEEP** 状態、**RUN** 状態のいずれかに遷移しますが、それは **START\_ENABLE** パラメータの値により決まります。

#### 引数

**hAsr**

エンジン インスタンスのハンドル [入力]

#### 戻り値

エラーコード

<b>ERR_WRONG_STATE</b>	不適切な状態のエンジンでこの関数が呼ばれました。 この関数を実行するにはエンジンが <b>CASR_IDLE</b> の状態である必要があります。
<b>ERR_NULL_HANDLE</b>	<b>NULL</b> ハンドルが渡されました。
<b>ERR_INVALID_ENGINE</b>	指定したエンジンのハンドルは有効ではありません。 すでにクローズされている可能性があります。
<b>ERR_TASK_THREAD</b>	<b>CasrOpen</b> 関数を呼び出したタスク/スレッドとは別のタスク/スレッドでエンジンのハンドルが使用されています。
<b>ERR_INVALID_SPI</b>	<b>SPI</b> データのハンドルは有効ではないか、そのハンドルを生成したタスクとは別のタスクで使用されています。

この関数から呼び出されるコールバック関数

**CBSTATE**

状態変更を通知するのに使用されるコールバック関数

#### 関連項目

**CasrOpen()**、**CasrStop()**

---

## CasrStop

### ERRORID CasrStop(HASR hAsr)

音声認識エンジンを強制停止します。この時点までに入力されたすべての音声は認識結果に変換されます。

#### 引数

**hAsr**

エンジン インスタンスのハンドル [入力]

#### 戻り値

#### エラーコード

<b>ERR_NULL_HANDLE</b>	NULL ハンドルが渡されました。
<b>ERR_INVALID_ENGINE</b>	指定したエンジンのハンドルは有効ではありません。 すでにクローズされている可能性があります。
<b>ERR_TASK_THREAD</b>	<b>CasrOpen</b> 関数を呼び出したタスク/スレッドとは別のタスク/スレッドでエンジンのハンドルが使用されています。
<b>ERR_INVALID_SPI</b>	<b>SPI</b> データのハンドルは有効ではないか、そのハンドルを生成したタスクとは別のタスクで使用されています。

#### この関数から呼び出されるコールバック関数

##### **CBRESULT**

認識結果が得られたことを示すために使用されるコールバック関数

##### **CBSTATE**

状態変更を通知するのに使用されるコールバック関数

#### 関連項目

**CasrOpen()**、**CasrStart()**、**CasrAcquisition()**



## 5.2 追加機能

### 5.2.1 ユーザー ワード トレーニング関連

---

#### CasrAcknowledge

##### ERRORID CasrAcknowledge(HASR hAsr, DWORD Decision)

ユーザー ワード トレーニング続行依頼の確認応答 (および決定) をエンジン側に送ります。  
また、この関数は **CBSTATE** コールバック関数から呼び出すことができます。

#### 引数

##### hAsr

**CasrOpen** 関数呼び出しによりオープンされたエンジンのハンドル [入力]

##### Decision

アプリケーションからの確認応答 (**PROMPT\_OK**、**PROMPT\_NOK**、**PROMPT\_CANCEL**)  
[入力]

#### 戻り値

#### エラーコード

<b>ERR_WRONG_STATE</b>	不適切な状態のエンジンでこの関数が呼ばれました。 この関数を実行するには <b>CASR_PROMPT</b> 、 <b>CASR_ACCEPT</b> 、 <b>CASR_CONFIRM</b> のいずれかの状態で呼び出す必要があります。
<b>ERR_NULL_HANDLE</b>	<b>NULL</b> ハンドルが渡されました。
<b>ERR_INVALID_ENGINE</b>	指定したエンジンのハンドルは有効ではありません。 すでにクローズされている可能性があります。
<b>ERR_TASK_THREAD</b>	<b>CasrOpen</b> 関数を呼び出したタスク/スレッドとは別のタスク/スレッドでエンジンのハンドルが使用されています。
<b>ERR_INVALID_SPI</b>	<b>SPI</b> データのハンドルは有効ではないか、そのハンドルを生成したタスクとは別のタスクで使用されています。

この関数から呼び出されるコールバック関数

#### **CBSTATE**

状態変更を通知するのに使用されるコールバック関数

#### **CBTRAIN**

ユーザー ワード トレーニング中に使用するユーザー対話を通知するためコールバック関数

関連項目

CBTRAIN()

---

## CasrAddContextUserWord

ERRORID CasrAddContextUserWord(HCONT hCont, HCLASSES hClasses, char \* pszClass, USERWORDBUF userWordBuf, DWORD dwUserWordSize, PCWORDID pnFirstUserWordId, PDWORD pcUserWordIds)

この関数は、コンテキストにユーザー ワードを追加します。  
ユーザー ワードを特定のコンテキストのクラスに追加します。1 つのテキストに 1 つのクラスが複数回使用されるため、ID も 1 つ以上生成され戻されます。

### 引数

**hCont**

コンテキスト インスタンスのハンドル [入力]

**hClasses**

クラス名インスタンスのハンドル [入力]

**pszClass**

クラス名文字列 [入力]

**userWordBuf**

ユーザー ワード バッファ [入力]

**dwUserWordSize**

ユーザー ワード データ バッファのサイズ (バイト数) [入力]

**pnFirstUserWordId**

割り当てられた最初のユーザー ワード ID が格納される変数へのポインタ [出力]

**pcUserWordIds**

割り当てられたユーザー ワード ID の数が格納される変数へのポインタ [出力]

### 戻り値

#### エラーコード

**ERR\_CONTEXT\_IN\_USE**

指定したコンテキストは、別のエンジンで使用中です。

**ERR\_MALLOC**

メモリ アロケーション エラー  
エンジンのダイナミック データのメモリへの格納あるいは開放のためにコールバック関数 CBMALLOC/CBFREE が呼ばれましたが、メモリ上のヒープ領域が十分でない可能性があります。

## メモ

**CasrStartUserWordTraining** 関数を呼び出した後、ユーザー トレーニング プロセスが完了すると、パラメータ **userWordBuf** へのポインタが使用可能になります。この関数を呼ぶと、パラメータ **pnFirstUserWordId** と **pcUserWordIds** にそれぞれ最初の ID とその ID の数が戻るため、次に使用可能なユーザー ワード ID レンジが確定します。指定したクラスがそのグラマー中に何度か発生し、その度に新しいユーザー ワード ID が生成されるため、そのレンジはその範囲が変動可能なレンジとして定義されます。**CasrStartUserWordTraining** 関数呼び出しによって取得された **userWordBuf** を一旦保存しておけば、いつでもこの関数を呼び出して、最少のメモリ消費量で、このユーザー ワードをコンテキストに追加することができます。

## 関連項目

**CasrImportContext()**、**CasrCloseContext()**、**CasrStartUserWordTraining()**

---

## CasrDeleteContextUserWord

ERRORID CasrDeleteContextUserWord(HCONT hCont, HCLASSES hClasses, char \* pszClass, CWORDID nFirstUserWordId, DWORD cUserWordIds)

ユーザー ワードをコンテキストから削除します。

### 引数

**hCont**  
コンテキスト インスタンスのハンドル [入力]

**hClasses**  
クラス名インスタンスのハンドル [入力]

**pszClass**  
クラス名文字列 [入力]

**nFirstUserWordId**  
最初のユーザー ワードの ID [入力]

**cUserWordIds**  
ユーザー ワード ID の数 [入力]

### 戻り値

#### エラーコード

<b>ERR_CONTEXT_IN_USE</b>	指定したコンテキストは別のエンジンで使用中です。
<b>ERR_MALLOC</b>	メモリ アロケーション エラー エンジンのダイナミック データのメモリへの格納あるいは開放のためにコールバック関数 <b>CBMALLOC/CBFREE</b> が呼ばれましたが、メモリ上のヒープ領域が十分でない可能性があります。
<b>ERR_INVALID_RANGE</b>	無効なユーザ ワード ID のレンジが指定されました。

### メモ

この関数は、**CasrAddContextUserWord** 関数呼び出しによって取得したユーザー ワード ID のレンジを削除します。また、**nFirstUserWordId** よりも大きなユーザー ワード ID の値を **cUserWordIds** だけデクレメントします<sup>2</sup>。この場合、直前にインポートしたコンテキストに追加登録されたユーザー ワードだけが削除可能であることに注意してください。

### 関連項目

**CasrImportContext()**、**CasrCloseContext()**、**CasrAddContextUserWord()**

---

<sup>2</sup> ユーザー ワード ID は 0x80000000 から始まります。

---

## CasrStartUserWordTraining

ERRORID CasrStartUserWordTraining(HASR hAsr, PUSERWORDBUF pUserWordBuf, PDWORD pdwSize)

ユーザー ワード トレーニングを開始します。この関数を呼び出しても、パラメータの **UserWordBuf** と **dwSize** にすぐには実際のポインタは格納されないため、格納されたアドレスはユーザー ワード トレーニングが終了するまで有効にしておく必要があります。この関数を呼び出す前には、**START\_ENABLE** エンジン パラメータを無効にします (**FALSE** をセット)。ユーザ ワード処理には **SLEEP** 状態がなく、エンジンがいつもすぐに **START (PROMPT)** 状態から **RECORD** 状態へ遷移します。一連のユーザー ワード トレーニングが完了すると (ユーザー ワードとして登録するかどうかの確認依頼(**CONFIRM**) に **PROMPT\_OK** が返された時)、エンジン側では **CBMALLOC** コールバック関数呼び出しにより、ユーザー ワードのメモリを確保し、パラメータ **pUserWordBuf** にそのメモリへのポインタを返します。このユーザー ワードを使用しない場合は、アプリケーションから **CBFREE** コールバック関数を呼び出して、そのユーザー ワードに割り当てていたメモリを開放します。

### 引数

**hAsr**

エンジン インスタンスのハンドル [入力]

**pUserWordBuf**

ユーザー ワード データを含むバッファへのポインタを受け取る変数のアドレス。API のユーザーは、このデータを使用して、ユーザー ワードをコンテキストに追加することができます。ただし、このバッファが使用可能になるのは、一連のユーザー ワード トレーニングの完了時、すなわち通知 **PROMPTTYPE\_TRAINEND** を受け取った時からになります。[出力]

**pdwSize**

このバッファのサイズを受け取る変数のアドレス[出力]

### 戻り値

#### エラーコード

**ERR\_WRONG\_STATE**

不適切な状態のエンジンでこの関数が呼び出されました。この関数を呼び出すには、エンジンが **CASR\_IDLE**、**CASR\_DATAREADY** のいずれかの状態である必要があります。

**ERR\_NULL\_HANDLE**

**NULL** ハンドルが渡されました。

**ERR\_INVALID\_CONTEXT**

インポートされたコンテキストへのハンドルは有効ではありません。すでにクローズされている可能性があります。

**ERR\_TASK\_THREAD\_CONTEXT**

**CasrImportContext** 関数を呼び出したタスク/スレッドとは別のタスク/スレッドでコンテキストのハンドルが使用されています。

ERR_INVALID_ENGINE	指定したエンジンのハンドルは有効ではありません。 すでにクローズされている可能性があります。
ERR_TASK_THREAD	<b>CasrOpen</b> 関数を呼び出したタスク/スレッドとは別のタスク/スレッドでエンジンのハンドルが使用されています。
ERR_MALLOC	メモリ アロケーション エラー ユーザーデータのメモリへの格納あるいは開放のためにコールバック関数 <b>CBMALLOC/CBFREE</b> が呼ばれましたが、メモリ上のヒープ領域が十分でない可能性があります。
ERR_INVALID_SPI	<b>SPI</b> のハンドルが有効ではないか、このハンドルを生成したタスクとは別のタスクで使用されています。

この関数から呼び出されるコールバック関数

#### CBSTATE

状態変更を通知するために使用されるコールバック関数

関連項目

**CasrAddContextUserWord()**, **CasrDeleteContextUserWord()**

## 5.2.2 コンテキスト関連

---

### CasrExportContext

ERRORID CasrExportContext(HCONT hCont, PCONT pCont, PDWORD pdwLength)

コンテキストをデータ バッファにエクスポートします。まず **pCont** に **NULL** を指定してこの関数を呼び出すと、そのコンテキストの実際のバッファサイズが **pdwLength** に格納されます。次に、アプリケーション側ではこの取得したバッファサイズを基に **pcont** 用の十分なメモリ領域を確保し、再び **CasrExportContext** 関数を呼び出して、このコンテキストをそのメモリバッファにエクスポートします。

#### 引数

##### **hCont**

コンテキスト インスタンスのハンドル [入力]

##### **pCont**

コンテキスト データ バッファ。アプリケーション側では、この関数を呼び出す前に十分なメモリ領域を確保してから、**dwLength** にそのバッファサイズを指定してこの関数を呼び出します。[出力]

##### **pdwLength**

コンテキスト データ バッファの長さ (バイト数) が格納される変数へのポインタ [出力]  
--- メモリ情報取得時

コンテキスト データ バッファの長さ (バイト数) が渡される変数へのポインタ [入力] ---  
実際のエクスポート時

#### 戻り値

#### エラーコード

**ERR\_BUF\_TOO\_SMALL**                      出力バッファが小さすぎます。

#### メモ

**pCont** パラメータに **NULL** を指定した場合でも、常に指定したコンテキストの実際のデータバッファのサイズが返されます。

#### 関連項目

**CasrImportContext()**



### 5.2.3 クラス関連

---

#### CasrCloseClasses

ERRORID CasrCloseClasses(HCLASSES hClasses)

クラス名インスタンスをクローズします。

引数

**hClasses**

クラス名インスタンスのハンドル [入力]

戻り値

今のところ、関数呼び出しの処理結果にかかわらず、常に **ERR\_SUCCESS** が返ってきます。

メモ

この関数は関連したすべてのリソースを開放し、指定したクラス名インスタンスを無効にします。

関連項目

**CasrImportClasses()**

---

## CasrImportClasses

ERRORID CasrImportClasses(HAPI hApi, PCLASSES pClasses, PHCLASSES phClasses)

クラス名データ バッファをインポートします。

### 引数

**hApi**

API インスタンスのハンドル [入力]

**pClasses**

クラス名データ バッファ [入力]

**phClasses**

クラス名インスタンスのハンドル [出力]

### 戻り値

### エラーコード

**ERR\_MALLOC**

メモリ アロケーション エラー

エンジンのダイナミック データのメモリへの格納あるいは開放のためにコールバック関数 **CBMALLOC/CBFREE** が呼ばれましたが、メモリ上のヒープ領域が十分でない可能性があります。

### メモ

このクラス名データ バッファには、クラス名とクラス ID の対応関係が含まれており、これは **PC** ツールから得ることができます。クラス名インスタンス ハンドルはインポートされたデータと関連付けられています。

### 関連項目

**CasrCloseClasses()**

## 5.2.4 コンテキストとクラスのマージ

---

### CasrMergeContextsAndClasses

ERRORID CasrMergeContextsAndClasses(HCONT hCont1, HCONT hCont2, PHCLASSES phClasses1, PHCLASSES phClasses2, PHCONT phContResult, PHCLASSES phClassesResult)

この関数は、2 つのコンテキスト バッファとそれに該当するクラス バッファをマージし、新しいコンテキストとクラス バッファを生成します。これらのバッファはすでにインポートされ、ハンドルと関連付けられた状態になります。これらのハンドルはそれぞれ、**phContResult**、**phClassesResult** の指す変数に格納されます。マージされたコンテキストが不要になった場合は、**CasrCloseContext/CasrCloseClasses** 関数を呼び出して、これら 2 つのハンドルに関連するリソースを解放する必要があります。

#### 引数

**hCont1**

コンテキスト インスタンスのハンドル [入力]

**hCont2**

コンテキスト インスタンスのハンドル [入力]

**phClasses1**

クラス名インスタンスのハンドル [入力]

**phClasses2**

クラス名インスタンスのハンドル [入力]

**phContResult**

マージされたコンテキストのハンドルが格納される変数へのポインタ [出力]

**phClassesResult**

マージされたクラス名インスタンスのハンドルが格納される変数へのポインタ [出力]

#### 戻り値

#### エラーコード

**ERR\_CONTEXT\_IN\_USE**

指定したコンテキストは他のエンジンで使用されています。

**ERR\_MALLOC**

メモリ アロケーション エラー  
ユーザーデータのメモリへの格納あるいは開放のためにコールバック関数 **CBMALLOC/CBFREE** が呼ばれましたが、メモリ上のヒープ領域が十分でない可能性があります。

**ERR\_INCOMPATIBLE\_CONTEXTS** 指定した 2 つのコンテキストのデータ型が違います。

**ERR\_DOUBLE\_CLASSNAMES**

同じクラス名を含むコンテキストはマージすることはできません。

#### メモ

この関数は、同時に **2** つのコンテキストをエンジン上で使用可能にするために使用しますが、同じクラス名を持つ **2** つのコンテキストをマージすることはできません。マージされたコンテキストは **CasrActivateContext** 関数呼び出しによりアクティブ化することができます。

#### 関連項目

**CasrCloseContext()**、**CasrImportClasses()**、**CasrImportContext()**

## 6 コールバック関数

### CBABNORM

unsigned char ( \*CBABNORM )( ABNORMCOND AbnormalCondition, PUSERDATA pUserData )

異常状態を通知するのに使用するコールバック関数

引数

AbnormalCondition: 異常状態の種類

以下にその種類を示します。

**BADSNR**

S/N 比が低すぎます。

**OVERLOAD**

音量が大き過ぎます

**TOOQUIET**

音量が小さ過ぎます

**NOSIGNAL**

AGC 開始後、信号が確認できません。

**GARBLED SOUND**

非常に短い雑音が検出されました。

**POORMIC**

マイクの性能が良好ではありません。

pUserData: CasrOpen および CasrAPIInit 関数呼び出しで API に渡されたユーザーデータ

戻り値

異常終了時は 0 以外の値を返します。

## CBAGC

unsigned char ( \*CBAGC )( WORD NewGain, PUSERDATA pUserData )

アナログ ゲイン値変更依頼を行うコールバック関数。このコールバック関数呼び出しに呼応する形で、アプリケーション側からは、**CasrGainSet** 関数を呼び出して API にそのアナログ ゲインの新しい設定値を渡す必要があります。**CasrGainSet** 関数は、このコールバック関数内で呼び出すこともできますが、後から単独で呼び出すこともできます。ダミーの **CBAGC** コールバック関数、すなわちこの関数を呼び出し後 **CasrGainSet** 関数を呼ばないと、実際のアナログ ゲイン値の変更は行われません。

### 引数

**NewGain:** アナログ ゲインの新しい設定値 1 (最小) ~ 65535 (最大)。アナログ ゲインの値を変更する場合はその変更する値を、自動ゲイン制御の機能を無効にする場合は 0 を指定します。

**pUserData:** **CasrOpen** および **CasrAPIInit** 関数呼び出しで API に渡されたユーザー データ

### 戻り値

異常終了時は 0 以外の値を返します。

## CBASKCURRENTGAIN

unsigned char ( \*CBASKCURRENTGAIN )( PUSERDATA pUserData )

現在のアナログ ゲインの設定値取得を依頼するコールバック関数です。このコールバック関数呼び出しに呼応する形で、アプリケーション側からは、**CasrGainSet** 関数を呼び出して API にそのアナログ ゲインの新しい設定値を渡す必要があります。**CasrGainSet** 関数は、このコールバック関数内で呼び出すこともできますが、あとから単独で呼び出すこともできます。

### 引数

pUserData : CasrOpen および CasrAPIInit 関数で渡されたユーザー データ

### 戻り値

異常終了時は 0 以外の値を返します。

## CBRESULT

unsigned char ( \*CBRESULT )( PCASRRESULT pResult, PUSERDATA pUserData )

音声認識の結果が得られたことを示すのに使用するコールバック関数。

### 引数

**pResult** : 結果構造体へのポインタ。このポインタはアプリケーションの処理がコールバック関数で行われている間だけ有効になります。したがってこの結果構造体はこのコールバック関数が終了すると破壊されます (使用不可になります)。

結果構造体のメンバーは以下のとおりです。

**iNbr**

認識候補の数

**pSentences**

**N-best** 候補結果のすべてのセンテンスを含む配列。この配列は、コンフィデンス値 (信頼度) でソートされています。各センテンスはそのセンテンスが正しいものであるかどうかの確信の度合いを示す **iConf** メンバを持っています。

**pUserData** : **CasrOpen** および **CasrAPIInit** 関数呼び出しで **API** に渡されたユーザーデータ

### 戻り値

異常終了時は 0 以外の値を返します。



## CBSTATE

unsigned char ( \*CBSTATE )( CASRSTATE State, PUSERDATA pUserData )

エンジンの状態変更を通知するコールバック関数

引数

**State:** エンジン状態

エンジン状態には以下のものがあります。

### CASR\_BOOT

コンテキスト、言語、ユーザーがまだアクティブ化されていません。

### CASR\_DATAREADY

必要なデータはすべてアクティブ化されています。

### CASR\_IDLE

コンテキスト、言語、ユーザーがアクティブ化されています。

### CASR\_SLEEP

CPU 消費量の少ないプロセスで音声開始を検出。

### CASR\_RUN

音声認識中

### CASR\_RECOVER

後続無音を検出した、あるいは **CasrStop** 関数が呼び出された状態。認識結果の通知もこの状態から行なわれます。

### CASR\_PROMPT

ユーザー ワード トレーニングのための発話開始了解のユーザー フィードバック確認待ち状態

### CASR\_RECORD

ユーザー ワード トレーニングのための発話の録音中。

### CASR\_ACCEPT

ユーザー ワード トレーニングのための録音した発話を受理するかどうかのユーザー フィードバック確認待ち状態

### CASR\_CONFIRM

ユーザー ワード トレーニングのための録音した発話をユーザー ワードとして受理するかかどうかのユーザー フィードバック確認待ち状態

**pUserData** : **CasrOpen** および **CasrAPIInit** 関数呼び出しで **API** に渡されたユーザーデータ

戻り値

異常終了時は 0 以外の値を返します。

## CBTRAIN

unsigned char ( \*CBTRAIN )( PROMPTTYPE PromptType, PUSERDATA pUserData )

ユーザー ワード トレーニング中、ユーザーとの対話処理を通知するために使用するコールバック関数。

### 引数

**PromptType:** プロンプト通知の種類。指定可能な値については **PROMPTTYPE** を参照してください。

プロンプト通知には、以下の種類があります。

#### PROMPTTYPE\_START

発話開始の前に通知されます。この関数呼び出しが成功すると話者はユーザー ワードの発話を開始することができます。

##### 指定可能な確認応答

- PROMPT\_OK : 話者は発話を開始することが可能です。
- PROMPT\_CANCEL : 現在のトレーニングがキャンセルされ、  
PROMPTTYPE\_TRAINEND が通知されます。

#### PROMPTTYPE\_ACCEPT

録音したユーザー ワードを受理するかどうかを話者に問い合わせます。

##### 指定可能な確認応答

- PROMPT\_OK : 話者はこの発話を受理します。
- PROMPT\_NOK : 話者はこの発話を棄却します。
- PROMPT\_CANCEL : 現在のトレーニングがキャンセルされ、  
PROMPTTYPE\_TRAINED が通知されます。

#### PROMPTTYPE\_CONFIRM

3 回の発話が終了したときに呼ばれ、その 3 回の発話を用いてユーザー ワードを作成してよいかどうかの確認を話者に求めます。

##### 指定可能な確認応答

- PROMPT\_OK : OK (ユーザー ワードとして作成します)
- PROMPT\_CANCEL : ユーザー ワードは作成されず、PROMPTTYPE\_TRAINED  
が通知されます。

#### PROMPTTYPE\_TRAINED

トレーニング終了。確認応答必要なし

**pUserData :** CasrOpen および CasrAPIInit 関数呼び出しで API に渡されたユーザーデータ

### 戻り値

異常終了時は 0 以外の値を返します。

## 7 OS コールバック関数

OS コールバック関数は、ASR1600\C API で必要なオペレーティング システム機能がカプセル化された一連のコールバック関数です。マルチタスクをサポートしていないオペレーティングシステムは、これらのコールバック関数はダミー関数として定義することができますが、CBMALLOC、CBFREE、および CBREALLOC の 3 つのコールバック関数は必要になります。上記の 3 つの関数以外は、マルチタスクをサポートしている環境でマルチタスクやマルチスレッドの機能を十分に活用するのに必要になります。

### CBCREATECRITICALSECTION

HCRITSECTION ( \*CBCREATECRITICALSECTION )( PUSERDATA pUserData )

クリティカル セクションの生成と初期化を行うコールバック関数。マルチスレッドをサポートしていないオペレーティングシステムでは、このコールバック関数はただ単に 1 を返すだけのダミー関数として定義することができます。クリティカル セクションは、異なるスレッドからアクセスされるリソースの同期を取るのに使用されます。

#### 戻り値

クリティカルセクションに関連するハンドル (HCRITSECTION 型)。この関数呼び出しが異常終了すると、NULL が返ってきます。

#### 引数

pUserData : CasrOpen および CasrAPIInit または CspellOpen および CspellAPIInit 関数呼び出しで API に渡されたユーザー データ

## CBDELETECRITICALSECTION

BOOL ( \*CBDELETECRITICALSECTION )( HCRITSECTION hCritSection, PUSERDATA pUserData )

クリティカル セクションの削除を行うコールバック関数。マルチスレッドをサポートしていないオペレーティング システムでは、このコールバック関数は、ただ単に **TRUE** を返すだけのダミー関数として定義することができます。クリティカル セクションは異なるスレッドからアクセスされるリソースの同期をとるために使用します。この関数はクリティカル セクションによって使用されたすべてのリソースを解放します。

### 引数

**hCritSection** : 削除するクリティカル セクションのハンドル

**pUserData** : **CasrOpen** および **CasrAPIInit** または **CspellOpen** および **CspellAPIInit** 関数呼び出しで API に渡されたユーザー データ

### 戻り値

異常終了時は **TRUE** が返されます。

## CBENTERCRITICALSECTION

BOOL ( \*CBENTERCRITICALSECTION )( HCRITSECTION hCritSection, PUSERDATA pUserData )

クリティカルセクションにエントリするためのコールバック関数。マルチスレッドをサポートしていないオペレーティング システムでは、このコールバック関数は、ただ単に TRUE を返すだけのダミー関数として定義することができます。クリティカルセクションは異なるスレッドからアクセスされるリソースの同期をとるために使用します。このコールバック関数を呼び出したスレッドは、このコールバック関数のリターン時にこのクリティカルセクションを所有します。他のスレッドがこの関数を呼んだ場合、最初のスレッドが CBLEAVECRITICALSECTION 関数を呼び出すまでは、このスレッドはロックされます。

### 引数

hCritSection : エントリするクリティカルセクションのハンドル

pUserData : CasrOpen および CasrAPIInit または CspellOpen および CspellAPIInit 関数呼び出しで API に渡されたユーザーデータ

### 戻り値

異常終了時は TRUE が返されます。

## CBFREE

void ( \*CBFREE )( PVOID pMem, PUSERDATA pUserData )

CBMALLOC 関数呼び出しで確保したメモリを開放するのに使用するコールバック関数。

### 引数

pMem : 開放するメモリへのポインタ

pUserData : CasrOpen および CasrAPIInit 関数呼び出しで API に渡されたユーザーデータ

### 戻り値

なし

## CBGETCURTASK

int ( \*CBGETCURTASK )( PUSERDATA pUserData)

タスク ID の取得依頼に使用するコールバック関数。マルチタスクをサポートしていないオペレーティング システムでは、戻り値が常に 0 でもかまいません。この関数は複数のタスク間でハンドルが使用されていないかどうかをチェックするのに使用します。

戻り値

現在のタスク ID

引数

pUserData : CasrOpen および CasrAPIInit または CspellOpen および CspellAPIInit 関数呼び出しで API に渡されたユーザーデータ

## CBGETCURTHREAD

int ( \*CBGETCURTHREAD )( PUSERDATA pUserData)

現在のスレッド ID の取得の依頼に使用するコールバック関数。マルチスレッドをサポートしていないオペレーティング システムでは、戻り値が常に **0** でもかまいません。この関数は複数のスレッド間でハンドルが使用されていないかどうかをチェックするのに使用します。

戻り値

現在のスレッドの ID

引数

**pUserData** : **CasrOpen** および **CasrAPIInit** 関数呼び出しで **API** に渡されたユーザーデータ



## CBLEAVECRITICALSECTION

BOOL ( \*CBLEAVECRITICALSECTION )( HCRITSECTION hCritSection, PUSERDATA pUserData )

クリティカル セクションから退出するためのコールバック関数。マルチスレッドをサポートしていないオペレーティング システムでは、このコールバック関数は、ただ単に TRUE を返すだけのダミー関数として定義することができます。クリティカル セクションは異なるスレッドからアクセスされるリソースの同期をとるために使用します。この関数はクリティカル セクションの所有権を解放します。すなわち(CBENTERCRITICALSECTION 関数の内部で)、そのクリティカル セクションの所有待ち状態にある他のスレッドのロックを解除します。

### 引数

hCritSection : 退出するクリティカル セクションのハンドル

pUserData : CasrOpen および CasrAPIInit 関数呼び出しで API に渡されたユーザーデータ

### 戻り値

異常終了時は TRUE が返されます。

## CBMALLOC

```
void * ( *CBMALLOC )( DWORD dwSize, PUSERDATA pUserData )
```

メモリの割り当てを行うコールバック関数

### 引数

**dwSize** : 割り当てるメモリ バッファのサイズ

**pUserData** : **CasrOpen** および **CasrAPIInit** 関数呼び出しで API に渡されたユーザー データ

### 戻り値

新しく割り当てられたメモリブロックへのポインタ。十分なメモリが確保できなかった場合は **NULL** を返します。

## CBREALLOC

```
void* (* CBREALLOC)(void *Ptr,DWORD Size, PUSERDATA pUserData);
```

**CBMALLOC** コールバック関数で以前に割り当てられたメモリ バッファのサイズを変更するのに使用します。このコールバックは、現在の **API** では使用されていませんが、スペリング機能を有効にする際この関数が必要になるため、現在のコールバック構造体の中にこの関数が指定されています。

### 引数

**Ptr:** サイズ変更対象のメモリ バッファへのポインタ

**Size:** 変更するメモリ バッファのサイズ

**pUserData:** **CasrOpen** および **CasrAPIInit** 関数呼び出しで **API** に渡されたユーザーデータ

### 戻り値:

新しく再割り当てされたメモリブロックへのポインタ。十分なメモリが確保できなかった場合、**NULL** を返します。戻り値のバッファ ポインタは、**Ptr** で指定した変更前のバッファ ポインタと同じであるとは限りません。

## 8 データ型、構造体、および列挙型定数

### 8.1 データ型

#### BOOL

ブール変数。TRUE (1) または FALSE (0)。

#### CASRLEVEL

現在のエンジンの信号レベルを表わす short 型変数。有効な値は 7200 (-72 dB) から 1800 (+18 dB) です。

#### CPARAMID

特定のエンジンパラメータを表わす DWORD 型変数。以下に示すパラメータがあります。

##### 音声認識エンジン (ハンドル タイプ HASR)

#### CPARAM\_TS\_ENABLE

後続無音状態検出の ON/OFF

レンジ: TRUE (1) または FALSE (0)  
デフォルト値: TRUE (1)

#### CPARAM\_START\_ENABLE

CPU 消費量の少ないプロセスでの音声開始検出の有効/無効を決定します。

レンジ: TRUE (1) または FALSE (0)  
デフォルト値: TRUE (1)

#### CPARAM\_ENABLEPREMRES

音声認識結果を即時出力します。認識エンジンが連続的に動作して、キーワードが認識される度にその結果が通知されるため、キーワード スポッティング モードで音声認識を行う際に有効です。

レンジ: TRUE (1) または FALSE (0)  
デフォルト値: FALSE (0)

#### CPARAM\_TS

発話終了の検出の基準である後続無音状態の最小継続時間

レンジ: 100 ミリ秒 ~ 2000 ミリ秒  
デフォルト値: 300 ミリ秒

#### CPARAM\_TIMEOUT

SLEEP、RUN、あるいは RECORD 状態のエンジンにおいて、音声認識エンジンが認識できる最大継続時間。

レンジ: 0 秒 (タイムアウト無し) ~ 30 秒  
デフォルト値: 0

#### CPARAM\_ACCURACY

音声認識の精度を示します。この値を指定する場合、CPU の負荷、必要メモリ消費量、およびレコグナイザの音声認識精度間のトレードオフ、すなわちこの値に小さい値を設定すると、CPU 占有時間やメモリ消費量が少なくて済みますが、認識精度は低下します。逆に高いと、CPU 占有時間やメモリ消費量が多くなります。ただし、音声認識の精度は向上します。

レンジ: 100 ~ 10000  
デフォルト値: 300

#### CPARAM\_REJECTION

このパラメータで、音声認識のコンフィデンス値を調整することができます。このパラメータ値が高ければ高いほど、リジェクトが起こりやすくなります（コンフィデンス値が低くなります）

レンジ: 0 ～ 100  
デフォルト値: 50

#### CPARAM\_GARBAGE

このパラメータで、「全音声モデル」<...>とのマッチ度を調整するのに使用します。この値が大きければ大きいほど、音声データのうち「全音声」モデル(<...>)にマッチする部分が大きくなります。

レンジ: 0 ～ 100  
デフォルト値: 50

#### CPARAM\_SENSITIVITY

低 CPU 負荷の音声検出システムの感度。この値が小さければ小さいほど、音声開始の検出 (SLEEP 状態から RUN 状態への遷移) が容易に行われることになります。

レンジ: 0 (0 dB) ～ 4000 (40 dB)  
デフォルト値: 2000 (20 dB)

#### CPARAM\_MINSPEECH

低 CPU 負荷の音声検出システムが、音声開始を検出するために必要な最小音声継続時間。(瞬間的な音を入力音声として認識するのを避ける)

レンジ: 10 ミリ秒 ～ 400 ミリ秒  
デフォルト値: 60 ミリ秒

#### CPARAM\_AGC

AGC 機能の ON/OFF

レンジ: TRUE (1) または FALSE (0)  
デフォルト値: FALSE (0)

#### CPARAM\_MAXNBEST

認識結果のセンテンスの最大候補数

レンジ: 1 ～ 1000  
デフォルト値: 10

#### CPARAM\_FARTALK

非接話型マイクか接話型マイクかの選択を AGC に通知

レンジ: TRUE (1) または FALSE (0)  
デフォルト値: TRUE (1)

#### CPARAM\_SELECTGENDER

エンジン側で認識する男女 (少年少女) 等のジェンダーを選択します。複数のジェンダーを選択した場合は、エンジン側で自動的にジェンダーを選択します。話者のジェンダーがあらかじめわかっている場合は、エンジンにそのジェンダーをセットすれば、非選択時 (GENDER\_UNKNOWN) と比較してメモリや CPU の消費量を削減することができ、いっぽう音声認識の性能にはほとんど影響を与えません。このジェンダーは、以下に示すパラメータを OR 結合することで定義します。

GENDER_MALE	(成人男性)
GENDER_FEMALE	(成人女性)
GENDER_CHILD	(子供)
GENDER_BOY	(少年)
GENDER_GIRL	(少女)
GENDER_UNKNOWN	(ジェンダー未知)

アクティブ化可能なジェンダーはアクティブ化された言語によって異なります。GENDER\_UNKNOWN を指定すると、その言語でカバーしているジェンダーすべてが有効になります。たいていの言語は 少なくとも GENDER\_MALE と GENDER\_FEMALE の 2 つのジェンダーをカバーしています。

レンジ: 以下に示すパラメータの 1 つ以上を OR 結合したビットフィールド  
GENDER\_MALE、GENDER\_FEMALE、GENDER\_CHILD、GENDER\_BOY、および  
GENDER\_GIRL  
デフォルト値: GENDER\_UNKNOWN

## CPARAMVALUE

特定のエンジン パラメータの値を示す DWORD 型変数。CPARAMID も参照のこと。

## CWORDID

コンテキスト中の単語を示す DWORD 型変数。

## DWORD

unsigned long 型変数 (32 ビット)

## ERRORID

エラーコードを含む DWORD 型変数。以下に示すエラーコードがあります。

基本エラーコード

0	ERR_SUCCESS	正常終了
1	ERR_MALLOC	メモリ アロケーション エラー
2	ERR_TASK_THREAD	<b>CasrOpen</b> 関数を呼び出したタスク/スレッドとは別のタスク/スレッドでエンジンのハンドルが使用されています。
3	ERR_APIINIT	<b>CasrAPIInit</b> 関数が 2 回以上呼ばれました。
4	ERR_CONTEXT_WRONG_HANDLE	指定されたコンテキスト ハンドルはインポートされたコンテキストのグローバルなリストに存在しません。
5	ERR_CONTEXT_IN_USE	<b>CasrCloseContext</b> で指定したコンテキストはいずれかのエンジンで使用中です。
6	ERR_NULL_HANDLE	NULL ハンドルが渡されました。
7	ERR_WRONG_STATE	不適切な状態にあるエンジンで関数が呼び出されました。
8	ERR_DATA_WRONG_HANDLE	指定したデータ ハンドルはインポートされたグローバルなリストに存在しません。
9	ERR_DATA_IN_USE	<b>CasrCloseData</b> で指定したデータはいずれかのエンジンで使用中です。
10	ERR_UNKNOWN_PARAM	<b>CasrSetParam/CasrGetParam</b> で範囲外のパラメータが指定されました。
11	ERR_INVALID_ENGINE	指定したエンジンのハンドルは有効ではありません。すでにクローズされている可能性があります。
12	ERR_LANGUAGE_MISMATCH	アクティブ化されている言語とはマッチしないコンテキストまたはセッションデータが指定されました。
13	ERR_WRONG_DATA	不適切な <b>DataType</b> でデータバッファをインポートしようとした。

14	ERR_INVALID_DATA	インポートされたデータのハンドルは有効ではありません。すでにクローズされている可能性があります。
15	ERR_TASK_THREAD_DATA	<b>CasrImportData</b> 関数を呼び出したタスク/スレッドとは別のタスク/スレッドで、インポートされたデータのハンドルが使用されています。
16	ERR_INVALID_CONTEXT	インポートされたコンテキストへのハンドルは有効ではありません。すでにクローズされている可能性があります。
17	ERR_TASK_THREAD_CONTEXT	<b>CasrImportContext</b> 関数を呼び出したタスク/スレッドとは別のタスク/スレッドでコンテキストのハンドルが使用されています。
18	ERR_INVALID_SPI	<b>SPI</b> データのハンドルは有効ではないか、そのハンドルを生成したタスクとは別のタスクで使用されています。
19	ERR_NULL_POINTER	<b>SPI</b> 関数に <b>NULL</b> ポインタが渡されました。
20	ERR_INVALID_GENDER	<b>CPARAM_SELECTGENDER</b> に不正な値が指定されました。 ( <b>CasrActivateData</b> 関数実行中でも発生する可能性があります)
20	ERR_NO_USERWORD_TRAINING	現在のアクティブな言語ではユーザー ワード トレーニングを行うことはできません。
22	ERR_NO_START_DETECTION	音声開始検出を有効にして、 <b>CasrStartUserWordTraining</b> 関数が呼び出されました。
23	ERR_WRONG_SIZE	不適切なサイズのバッファが渡されました。
24	ERR_UTTFORMAT_UNKNOWN	使用できない話者フォーマットです。
25	ERR_INVALID_WORDID	<b>CasrSetActiveWord</b> で範囲外の単語 ID が指定されました。
26	ERR_TIMEOUT	タイムアウトが発生しました。
追加エラーコード		
100	ERR_INCOMPATIBLE_HANDLES	互換性のないハンドルが指定されました。
110	ERR_BUF_TOO_SMALL	出力バッファが小さすぎます。
120	ERR_INVALID_CLASSNAME	クラス名が有効ではありません。
130	ERR_INCOMPATIBLE_CONTEXTS	指定した 2 つのコンテキストのデータ型が違います。
131	ERR_DOUBLE_CLASSNAMES	同じクラス名を含むコンテキストはマージすることはできません。
140	ERR_INVALID_RANGE	指定したレンジは無効です。



## HASR

エンジンインスタンスのハンドルを示すポインタ

## HCONT

エンジン上でロード、アクティブ化して使用可能にできるコンテキスト データのハンドルを示すポインタ

## HCLASSES

クラス名インスタンスのハンドルを示すポインタ

## HCRTSECTION

クリティカル セクションに関連づけられたハンドルを示すポインタ

## HDATA

メモリにロードしたデータに関連づけられたハンドルを示すポインタ

## PALTERNATIVE

ALTERNATIVE 構造体へのポインタ

## PCASRLEVEL

CASRLEVEL 型変数へのポインタ

## PCASRRESULT

CASRRESULT 構造体へのポインタ

## PCASRSTATE

CASRSTATE 型変数へのポインタ

## PDATA

DATA 構造体へのポインタ

## PCLASSES

クラス名データバッファ

## PCOSCALLBACKS

COSCALLBACKS 構造体へのポインタ

## PCPARAMLIST

CPARAMLIST 構造体へのポインタ

## PCPARAMVALUE

CPARAMVALUE 型変数へのポインタ

## PCRECOGCALLBACKS

CRECOGCALLBACKS 構造体へのポインタ

## PCONT

コンテキストバッファへの PVOID 型ポインタ

PCWORDID

CWORDID 型変数へのポインタ

PDWORD

DWORD 型変数へのポインタ

PHASR

HASR 型変数へのポインタ

PHCONT

HCONT 型変数へのポインタ

PHCLASSES

HCLASSES 型変数へのポインタ

PHDATA

HDATA 型変数へのポインタ

PRECWORD

RECWORD 構造体へのポインタ

PSENTENCE

SENTENCE 構造体へのポインタ

PSPEECHUNITBUF

SPEECHUNITBUF 構造体へのポインタ

PTIMEINFO

TIMEINFO 構造体へのポインタ

PUSERWORDBUF

USERWORDBUF へのポインタ

PVOID

void 型ポインタ

PWORD

WORD 型変数へのポインタ

PWORDBUF

WORDBUF 構造体へのポインタ

PUSERDATA

USERDATA 型変数へのポインタ

USERWORDBUF

ユーザー ワード バッファを示す PVOID 型ポインタ

WORD

unsigned short 型変数 (16 ビット)

## 8.2 構造体

### ALTERNATIVE

```
typedef struct {  
    int iConf;  
    long lScore;  
    CWORDID idWord;  
} ALTERNATIVE;
```

部分認識結果のフォーマット

メンバ

**iConf**

この認識結果候補の信頼の度合いを示すコンフィデンス値

**lScore**

この認識結果候補のスコア値。このスコア値は低ければ低いほど、マッチ度がよいことを表します。

**idWord**

この認識結果候補の ID (ユーザーワードの場合には、オフセット **OFFSET\_USERWORDID** が使用されます)。

### CASRRESULT

```
typedef struct {  
    int iNbr;  
    PSENTENCE pSentences;  
} CASRRESULT;
```

認識結果のフォーマット

メンバ

**iNbr**

認識結果候補の数

**pSentences**

**N-Best** 認識結果のすべてのセンテンスを含む配列

### COSCALLBACKS

```
typedef struct {  
    CBMALLOC cbMalloc;  
    CBFREE cbFree;  
    CBREALLOC cbRealloc;  
    CBGETCURTASK cbGetCurThread;  
    CBGETCURTHREAD cbGetCurThread;  
    CBCREATECRITICALSECTION cbCreateCriticalSection;  
    CBDELETECRITICALSECTION cbDeleteCriticalSection;  
    CBENTERCRITICALSECTION cbEnterCriticalSection;  
    CBLEAVECRITICALSECTION cbLeaveCriticalSection;  
} COSCALLBACKS;
```

この API で必要なシステムコールをカプセル化した一連のコールバック関数を格納する構造体。 **CasrAPIInit** 関数を呼び出すとこの構造体が API に渡されます。

メンバ

**cbMalloc**

メモリ割り当て関数

**cbFree**

メモリ開放関数

**cbRealloc**

メモリ再割り当て関数

**cbGetCurTask**

現在のタスクの ID を取得します (マルチタスクをサポートしている環境でのみ有効)

**cbGetCurThread**

現在のスレッドの ID を取得します (マルチタスクをサポートしている環境でのみ有効)

**cbCreateCriticalSection**

クリティカル セクションの生成と初期化を行う関数 (マルチ スレッドをサポートしている環境下でのみ有効)

**cbDeleteCriticalSection**

クリティカル セクションの削除を行う関数 (マルチ スレッドをサポートしている環境下でのみ有効)

**cbEnterCriticalSection**

クリティカル セクションへのエントリを行う関数(マルチ スレッドをサポートしている環境下でのみ有効)

**cbLeaveCriticalSection**

クリティカル セクションから退出を行う関数(マルチ スレッドをサポートしている環境下でのみ有効)

## CPARAMLIST

```
typedef struct {
    DWORD cparam_ts_enable;
    DWORD cparam_start_enable;
    DWORD cparam_ts;
    DWORD cparam_timeout;
    DWORD cparam_accuracy;
    DWORD cparam_rejection;
    DWORD cparam_garbage;
    DWORD cparam_sensitivity;
    DWORD cparam_minspeech;
    DWORD cparam_agcon;
    DWORD cparam_maxnbest;
    DWORD cparam_enablepremlres;
    DWORD cparam_fartalk;
    DWORD cparam_selectgender;
} CPARAMLIST;
```

一連のエンジン パラメータを格納する構造体。CPARAMID および CPARAMVALUE を参照のこと。

メンバ

**cparam\_ts\_enable**

後続無音状態検出 ON/OFF

**cparam\_start\_enable**

低 CPU 負荷の音声開始検出の ON/OFF

**cparam\_ts**

後続無音状態の最小継続時間

**cparam\_timeout**

音声認識エンジンが認識できる最大継続時間

**cparam\_accuracy**

音声認識の精度

**cparam\_rejection**

音声認識時のリジェクト機能 (認識結果の受理/棄却) の調整値

**cparam\_garbage**

「全音声」モデル (<...>) とのマッチ度の調整値

**cparam\_sensitivity**

低 CPU 負荷の音声検出の感度。

**cparam\_minspeech**

低 CPU 負荷の音声検出システムが音声開始を検出するために必要な最小音声継続時間。

**cparam\_agcon**

AGC の ON/OFF

**cparam\_maxnbest**

認識結果候補の最大数

**cparam\_enablepremres**

音声認識結果の即時出力機能の ON/OFF

**cparam\_fartalk**

非接話型マイクか接話型マイクかの選択

**cparam\_selectgender**

自動ジェンダー選択範囲の設定

## CRECOGCALLBACKS

```
typedef struct {
    CBRESULT cbResult;
    CBSTATE cbState;
    CBTRAIN cbTrain;
    CBABNORM cbAbnorm;
    CBAGC cbAgc;
    CBASKCURRENTGAIN cbAskCurrentGain;
} CRECOGCALLBACKS;
```

エンジンが使用する一連のコールバック関数を格納する構造体。この構造体は、**CasrOpen** 関数呼び出しにより **API** に渡されます。

メンバ

**cbResult**

結果通知

**cbState**

状態通知

**cbTrain**

ユーザー ワード トレーニングの通知

**cbAbnorm**

入力音の異常状態通知

**cbAgc**

アナログ ゲイン値の変更依頼

**cbAskCurrentGain**

現在のゲイン設定値の取得依頼

## DATA

```
typedef struct {
    DWORD dwDataType;
    DWORD dwSize;
} DATA;
```

データ バッファ情報を格納した構造体

メンバ

**dwDataType**

バッファのデータのタイプ。DATA\_LANG、DATA\_LANG\_FIXED、DATA\_CONTEXT、  
DATA\_SESSIONDATA、DATA\_NAMETREE のいずれかのタイプになります。  
dwSize  
データバッファのサイズ

## RECWORD

```
typedef struct {  
    int iNbrAlternatives;  
    TIMEINFO TimeInfo;  
    ALTERNATIVE pAlternatives;  
} RECWORD;
```

部分認識結果のフォーマット

メンバ

**iNbrAlternatives**

この単語に対する認識結果候補の数

**TimeInfo**

この単語の開始と終了を示すタイム スタンプ情報を格納した構造体

**pAlternatives**

すべての認識結果候補の配列

## SENTENCE

```
typedef struct {  
    int iNbrWords;  
    int iConf;  
    long lScore;  
    TIMEINFO TimeInfo;  
    RECWORD pWords;  
} SENTENCE;
```

部分音声認識結果のフォーマット

メンバ

**iNbrWords**

この認識結果のセンテンスに含まれる単語数

**iConf**

このセンテンスの認識の信頼の度合いを示すコンフィデンス値

**lScore**

このセンテンスのスコア値。このスコア値は低ければ低いほど、マッチ度がよいことを表しますが、通常は上記の **iConf**、すなわちコンフィデンス値を使用して下さい。

**TimeInfo**

このセンテンスの開始と終了を表わすタイム スタンプ情報を含む構造体。

**pWords**

このセンテンスを構成する単語を含む配列。

## SPEECHUNITBUF

```
typedef struct {  
    WORD NbrSpeechUnits;  
    PWORD pSpeechUnits;  
} SPEECHUNITBUF;
```

単語の発音の音声ユニットを表わす構造体

メンバ

**NbrSpeechUnits**

配列中の音声ユニットの数

**pSpeechUnits**

音声ユニットの配列



## TIMEINFO

```
typedef struct {  
    DWORD TimeBegin;  
    DWORD TimeEnd;  
} TIMEINFO;
```

発話のタイムスタンプ情報が格納される構造体。TimeBegin、TimeEnd には CastStart 関数呼び出し時からの音声サンプルの個数が格納されます。TimeBegin TimeEnd が -1 (0xffffffff) の場合は、タイムスタンプ情報がないことを意味しています。

### メンバ

#### TimeBegin

該当する発話の開始時間

#### TimeEnd

該当する発話の終了時間

## USERDATA

```
typedef struct {  
    DWORD dwApiUserData;  
    DWORD dwEngineUserData;  
} USERDATA;
```

ユーザー データを格納した構造体

### メンバ

#### dwApiUserData

CasrAPIInit/CspellAPIInit 関数呼び出しにより API に渡されるユーザーデータ

#### dwEngineUserData

CasrOpen/CspellOpen 関数呼び出しにより API に渡されるユーザーデータ

## WORDBUF

```
typedef struct {  
    WORD NbrProns;  
    PSPEECHUNITBUF pSpeechUnitBufs;  
} WORDBUF;
```

特定の単語の考えられる発音の情報を格納した構造体

### メンバ

#### NbrProns

特定の単語の発音の数

#### pSpeechUnits

異なる発音を含む音声ユニットの構造体の配列

### 8.3 列挙型変数

#### ABNORMCOND

```
enum ABNORMCOND {  
    BADSNR,  
    OVERLOAD,  
    TOOQUIET,  
    NOSIGNAL,  
    GARBLEDSOUND,  
    POORMIC  
};
```

入力音声の異常状態の種類

メンバ

##### BADSNR

S/N 比が低すぎます。

##### OVERLOAD

音量が大き過ぎます。

##### TOOQUIET

音量が小さ過ぎます。

##### NOSIGNAL

AGC 開始後、信号が確認できません。

##### GARBLEDSOUND

非常に短い雑音が検出されました。

##### POORMIC

マイクの性能が良好ではありません。

#### CASRSTATE

```
enum CASRSTATE {  
    CASR_BOOT,  
    CASR_DATAREADY,  
    CASR_IDLE,  
    CASR_SLEEP,  
    CASR_RUN,  
    CASR_RECOVER,  
    CASR_PROMPT,  
    CASR_RECORD,  
    CASR_ACCEPT,  
    CASR_CONFIRM  
};
```

エンジンの現在の状態を表す列挙型変数

メンバ

##### CASR\_BOOT

コンテキスト、言語、ユーザー、いずれも非アクティブな初期状態

##### CASR\_DATAREADY

エンジンで必要なすべてのデータがアクティブ化されている状態

##### CASR\_IDLE

コンテキスト、言語、ユーザーがアクティブな状態

##### CASR\_SLEEP

低 CPU 負荷の音声検出システムが動作中の状態

##### CASR\_RUN

音声認識中

##### CASR\_RECOVER

後続無音が検出された、あるいは、**CasrStop** 関数が呼び出された状態。音声認識結果を出力し、その旨を通知します。

#### **CASR\_PROMPT**

ユーザー ワード トレーニング中の発話開始のユーザー フィードバック待ち状態

#### **CASR\_RECORD**

ユーザー ワードの発話を録音中

#### **CASR\_ACCEPT**

ユーザー ワードの発話の受理依頼待機状態

#### **CASR\_CONFIRM**

録音した 3 回の発話をユーザー ワードとして受理するかどうかの確認依頼待機状態

### **CSAMPLEFORMAT**

```
enum CSAMPLEFORMAT {  
    PCM_16_11KHZ  
};
```

**CasrAcquisition** 関数呼び出しにより渡される音声サンプルのフォーマット指定

メンバ

**PCM\_16\_11KHZ**

16 ビット PCM 11 kHz

### **PROMPTTYPE**

```
enum PROMPTTYPE {  
    PROMPTTYPE_START,  
    PROMPTTYPE_ACCEPT,  
    PROMPTTYPE_CONFIRM,  
    PROMPTTYPE_TRAINED  
};
```

ユーザー ワード トレーニング中に送られる一連のプロンプト通知の列挙型変数。  
これらのプロンプト通知への返答は、**CasrAcknowledge** 関数の引数で指定してエンジン側に送られます。

メンバ

#### **PROMPTTYPE\_START**

発話開始の前に通知します。この関数呼び出しが成功すると、ユーザー ワード録音を開始することができます。

これに対する可能な確認応答

- **PROMPT\_OK** : 発話を開始することができます。
- **PROMPT\_CANCEL** : トレーニングがキャンセルされ、プロンプト通知の **PROMPTTYPE\_TRAINED** が送られます。

#### **PROMPTTYPE\_ACCEPT**

ユーザー発話の受理依頼を通知します。

これに対する可能な確認応答

- **PROMPT\_OK** : 話者はこの発話を受理します。
- **PROMPT\_NOK** : 話者はこの発話を棄却します。
- **PROMPT\_CANCEL** : トレーニングがキャンセルされ、プロンプト通知の **PROMPTTYPE\_TRAINED** が送られます。

#### PROMPTTYPE\_CONFIRM

録音した 3 回の発話を使ってユーザー ワードとして受理するかどうかの確認依頼を通知します。

これに対する可能な確認応答

- PROMPT\_OK : OK (ユーザー ワードとして登録します)

- PROMPT\_CANCEL : ユーザー ワードは登録されず、PROMPTTYPE\_TRAINEND がプロンプト通知として送られます。

#### PROMPTTYPE\_TRAINED

トレーニング終了。確認応答を送る必要はありません。