



---

## Hyperion(R) Enterprise(R)

リリース 6.5.1

---

APIリファレンス

**ORACLE®**  
ENTERPRISE PERFORMANCE  
MANAGEMENT SYSTEM

Hyperion Enterprise API リファレンス, 6.5.1

Copyright © 1991, 2009, Oracle and/or its affiliates. All rights reserved.

著者: Enterprise Information Development Team

このソフトウェアおよび関連ドキュメントの使用と開示は、ライセンス契約の制約条件に従うものとし、知的財産に関する法律により保護されています。ライセンス契約で明示的に許諾されている場合もしくは法律によって認められている場合を除き、形式、手段に関係なく、いかなる部分も使用、複写、複製、翻訳、放送、修正、ライセンス供与、送信、配布、発表、実行、公開または表示することはできません。このソフトウェアのリバース・エンジニアリング、逆アセンブル、逆コンパイルは互換性のために法律によって規定されている場合を除き、禁止されています。ここに記載された情報は予告なしに変更される場合があります。また、誤りが無いことの保証はいたしかねます。誤りを見つけた場合は、オラクル社までご連絡ください。

このソフトウェアまたは関連ドキュメントが、米国政府機関もしくは米国政府機関に代わってこのソフトウェアまたは関連ドキュメントをライセンスされた者に提供される場合は、次の通知が適用されます。

#### U.S. GOVERNMENT RIGHTS:

Programs, software, databases, and related documentation and technical data delivered to U.S. Government customers are "commercial computer software" or "commercial technical data" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, the use, duplication, disclosure, modification, and adaptation shall be subject to the restrictions and license terms set forth in the applicable Government contract, and, to the extent applicable by the terms of the Government contract, the additional rights set forth in FAR 52.227-19, Commercial Computer Software License (December 2007). Oracle USA, Inc., 500 Oracle Parkway, Redwood City, CA 94065.

このソフトウェアは様々な情報管理アプリケーションでの一般的な使用のために開発されたものです。このソフトウェアは、危険が伴うアプリケーション(人的傷害を発生させる可能性があるアプリケーションを含む)への用途を目的として開発されていません。このソフトウェアを危険が伴うアプリケーションで使用する際、このソフトウェアを安全に使用するために、適切な安全装置、バックアップ、冗長性、その他の対策を講じることは使用者の責任となります。このソフトウェアを危険が伴うアプリケーションで使用したことに起因して損害が発生しても、オラクル社およびその関連会社は一切の責任を負いかねます。

Oracle は、Oracle Corporation またはその関連会社、あるいはその両方の登録商標です。他の名称は、それぞれの所有者の商標である可能性があります。

このソフトウェアおよびドキュメントは、第三者のコンテンツ、製品、サービスへのアクセス、あるいはそれらに関する情報を提供することがあります。オラクル社およびその関連会社は、第三者のコンテンツ、製品、サービスに関して一切の責任を負わず、いかなる保証もいたしません。オラクル社およびその関連会社は、第三者のコンテンツ、製品、サービスへのアクセスまたは使用によって損失、費用、あるいは損害が発生しても、一切の責任を負いかねます。

---

# 目次

---

<b>第 1 章 Hyperion Enterprise API について</b>	15
Hyperion Enterprise API の概要	15
このガイドの使用方法	15
サポートされている言語	16
Visual Basic プログラミング上の注意	16
関数の宣言	17
定数	17
文字列	17
アドレスの渡し方	18
Visual Basic におけるコールバック関数	18
Visual Basic における apiStruct 構造体の使用方法	19
Visual Basic 以外のプログラムの変更	19
インクルードファイル	19
プログラムのインポートライブラリへのリンク	20
戻りコード	20
期間単位と表示形式	20
Hyperion Enterprise 期間単位	20
レポートの表示形式	21
<b>第 2 章 スプレッドシートのアドイン関数</b>	23
スプレッドシートのアドイン関数の概要	23
単一アプリケーションの管理関数	23
複数アプリケーションの管理関数	24
選択ダイアログボックス関数	24
データ取得関数	25
スプレッドシートのアドイン関数との連結詳細値	25
データ更新関数	26
アルファベット順の関数リファレンス	27
HypAcctAsk() - 勘定科目の選択	27
HypAcctListAskEx() - 勘定科目一覧の選択	28
HypCatAskEx() - データ種別の選択	28
HypConstruct() - アプリケーションを開く	29

HypConstructEx( ) - アプリケーションを開く	30
HypDestruct( ) - アプリケーションを閉じる	31
HypFreqAsk( ) - 期間単位の選択	32
HypGetDefJour( ) - デフォルト仕訳帳の取得	33
HypGetDefJourCat( ) - デフォルト仕訳帳データ種別の取得	34
HypGetDefJourPer( ) - デフォルト仕訳帳期間の取得	34
HypHPACC( ) - 勘定科目 ID またはサブ勘定科目 ID の取得	35
HypHPBET( ) - 優劣値の算出	36
HypHPCAD( ) - 通貨の説明の取得	36
HypHPCAL( ) - 算出勘定の判別	37
HypHPCDE( ) - データ種別説明の取得	38
HypHPCommit2( ) - データバッファの書き込み	39
HypHPCUREx( ) - コンポーネントの通貨の取得	40
HypHPDRV( ) - 派生値の算出	41
HypHPECO( ) - エンティティコードの取得	41
HypHPFlush( ) - データバッファのフラッシュ	42
HypHPFLW( ) - フロー勘定科目の判別	43
HypHPFRE( ) - データ種別期間単位の取得	44
HypHPFUL( ) - コンポーネントのエンティティに関する完全な説明の取得	45
HypHPHEA( ) - 勘定科目見出しの取得	46
HypHPINC( ) - 収益勘定科目の判別	47
HypHPINP( ) - 入力勘定科目の判別	47
HypHPJOUR( ) - 仕訳帳詳細情報の取得	49
HypHPKEYEx( ) - デフォルトのエンティティ、データ種別、勘定科目、または期間の取得	50
HypHPLNK( ) - データ値の保存	51
HypHPNAM( ) - エンティティ ID の取得	52
HypHPOWN( ) - 所有株数の取得	53
HypHPPBE( ) - 優劣比率の算出	55
HypHPPCH( ) - 差異率の算出	56
HypHPSCA( ) - 単位の取得	57
HypHPVAL( ) - 通常の勘定科目値の取得	58
HypHPVAL2( ) - 通常の勘定科目値の取得	59
HypHPVALEx( ) - 特殊な勘定科目値の取得	61
HypJourAsk( ) - 仕訳帳の選択	62
HypJourDetAsk( ) - 仕訳帳詳細の選択	63
HypMultiAsk( ) - デフォルトのアプリケーションの選択	63
HypMultiDefault( ) - デフォルトのハンドルの取得	64

HypMultiDeinit( ) - 複数アプリケーションからのログオフ	65
HypMultiEnum( ) - 開いているアプリケーションの列挙	65
HypMultiGet( ) - アプリケーションハンドルの取得	66
HypMultiInit( ) - 複数アプリケーションへのログオン	67
HypNamAskEx( ) - エンティティの選択	67
HypNameListAskEx( ) - エンティティ一覧の選択	68
HypPerAsk( ) - 期間の選択	69
HypSetDefJour( ) - デフォルト仕訳帳の設定	70
HypSetDefJourCat( ) - デフォルト仕訳帳データ種別の設定	70
HypSetDefJourPer - デフォルト仕訳帳期間の設定	71
HypSetMVMode( ) - HypVal 戻り値動作の設定	72
HypSetProgramName( ) - ログインユーザレポートのテキストの設定	72
HypValidateLnkParams( ) - リンクパラメータのチェック	73
HypValidateParams( ) - パラメータのチェック	74
HypValidateParamsEx( ) - パラメータのチェック	75
HypValReturn( ) - 最後の値取得のステータスを返す	76
<b>第 3 章 表関数</b>	79
表関数の概要	79
Hyp...( )関数を使用した表の選択と選択解除	79
Hyp...( )関数での表の使用	80
関連する表と Hyp...( )関数	80
組織構造体と Hyp...( )関数	81
アルファベット順の表関数リファレンス	82
DateConv( ) - 日付の変換	82
HypCatGetNumPeriodsEx( ) - データ種別に含まれる期間数の算出	83
HypCatGetPerShortEx( ) - データ種別期間単位内の期間 ID の取得	84
HypCatMapPeriodEx( ) - 期間の期間単位へのマッピング	84
HypEnumAcctListEntriesEx( ) - 勘定科目一覧の勘定科目の列挙	85
HypEnumApplications( ) - アプリケーションの列挙	87
HypEnum( ) - 表内のレコードの列挙	88
HypEnumNameListEntriesEx( ) - エンティティ一覧内のエンティティの列挙	90
HypEnumOrgNames( ) - 組織内のノードの列挙	91
HypEnumSubAcctSig( ) - Enumerate サブ勘定科目記号の列挙	93
HypFindEx( ) - 記号の検索	94
HypFindNameInOrgEx( ) - ノード ID の検索	95
HypGetCatPerFreq( ) - データ種別内の期間番号と期間単位の取得	96
HypGetChild( ) - 子ノードの取得	97
HypGetNameSig( ) - エンティティ記号の取得	98

HypGetOrgLevel( ) - 組織内レベルの取得	99
HypGetPerViewEx( ) - 期間単位と表示形式の取得	100
HypGetSibling( ) - 兄弟の取得	101
HypGetTopNodeEx( ) - トップノードの取得	101
HypIsNameParentEx( ) - エンティティが親であることの確認	102
HypLockEx( ) - 表の選択	103
HypQryRptFreqEx( ) - レポート期間単位のクエリ	105
HypQueryEx( ) - Hyperion Enterprise の表のクエリ	105
HypUnLockEx( ) - 表の選択解除	106
<b>第 4 章 高度な関数</b>	<b>109</b>
高度な関数の概要	109
アプリケーションを開く	110
表と高度な関数	110
apiStruct 構造体の使用	110
高度な関数を使用した表の選択と選択解除	111
選択のコールバック	112
表での高度な関数の使用	112
関連する表と高度な関数	113
スプレッドシートのアドイン関数と高度な関数の組み合わせ	113
選択ダイアログボックス関数	114
データの操作	114
データの取得	115
データの更新	117
連結詳細値	119
組織構造体と高度な関数	120
データの視点の設定	121
サーバのタスク	121
Hyperion Enterprise の INI ファイル関数	125
セキュリティ	125
Hyperion Enterprise のセキュリティを保護するための API 関数	126
使用されなくなった Ent...( )関数から新しい Ent...( )関数への置き換え	127
アルファベット順の高度な関数のリファレンス	127
EntAcctAsk( ) - 勘定科目の選択	127
EntAcctListAsk( ) - 勘定科目一覧の選択	128
EntAcctSplit( ) - 勘定科目記号の分割	129
EntAppendToCBChain( ) - コールバックチェーンへの追加	130
EntAppExtract( ) - アプリケーションの抽出	130
EntAppExtractVB( ) - アプリケーションの抽出	132

EntAppLoad( ) - アプリケーションの読み込み	134
EntAppLoadVB( ) - アプリケーションの読み込み	135
EntAsciiToDouble( ) - string から double への変換	139
EntCatAsk( ) - データ種別の変更	140
EntCatGetNumPeriods( ) - データ種別の期間数の計算	140
EntCatGetPerShort( ) - 期間ラベルの取得	141
EntCatMapPeriod( ) - 期間から期間単位へのマッピング	142
EntCloseApplication( ) - アプリケーションの終了	143
EntConsolidate( ) - 連結	143
EntCreateApplication( ) - アプリケーションの作成	145
EntDataExtract( ) - データの抽出	149
EntDataExtractVB2( ) - データの抽出	151
EntDataFileOpen( ) - データファイルを開く	155
EntDateConv( ) - 日付の変換と期間番号の計算	157
EntDeleteFromCBChain( ) - コールバックチェーンからの削除	157
EntDiscardChanges( ) - 変更の破棄	158
EntDiscardDefault( ) - デフォルトの破棄	159
EntDSMDDataExtract( ) - DSM データの抽出	159
EntDSMDDataExtractVB( ) - DSM データの抽出	162
EntEntityAsk( ) - エンティティの選択	166
EntEntityListAsk( ) - エンティティ一覧の選択	167
EntEntityListLock( ) - データの保護	167
EntEntityListUnLock( ) - データの保護解除	170
EntEnum( ) - 表内のレコードの列挙	171
EntEnumApplications( ) - アプリケーションの列挙	182
EntEnumOrgEntities( ) - 組織内のノードの列挙	183
EntEnumSubAcctSig( ) - サブ勘定科目の列挙	184
EntEnumUsersOnSystem( ) - システム上のユーザの列挙	186
EntFind( ) - 記号の検索	186
EntFindEntityInOrg( ) - ノード ID の検索	190
EntFormatNumber2( ) - 数値の書式設定	192
EntFreqAsk( ) - 期間単位の選択	193
EntGetAccountsInputType( ) - 勘定科目種別の取得	193
EntGetActiveModule( ) - プログラム名の取得	196
EntGetAppProfileLong( ) - アプリケーションプロファイル long の取得	196
EntGetAppProfileString( ) - アプリケーションプロファイル文字列の取得	197
EntGetCatPerFreq( ) - データ種別内の期間番号と期間単位の取得	198
EntGetChild( ) - 子ノードの取得	199

EntGetEntitySig( ) - エンティティ記号の取得 .....	200
EntGetHappFromSelect( ) - アプリケーションハンドルの取得 .....	201
EntGetLastError( ) - 最後のエラーの取得 .....	202
EntGetLastErrorByHApp( ) - 最後のエラーの取得 .....	203
EntGetOrgLevel( ) - 組織内のレベルの取得 .....	204
EntGetPerView( ) - 期間単位と表示形式の取得 .....	204
EntGetProfileLong( ) - Hyperion プロファイル long の取得 .....	205
EntGetProfileString( ) - Hyperion プロファイル文字列の取得 .....	206
EntGetRightsToTask( ) - タスクのアクセス権の取得 .....	207
EntGetSibling( ) - 兄弟ノードの取得 .....	207
EntGetVarAddr( ) - 変数のアドレスの取得 .....	208
EntInitApiStruct( ) - API 構造体の初期化 .....	209
EntIsAccountInput( ) - 勘定科目が入力勘定科目であるかどうかの確認 ...	210
EntIsEntityParent( ) - エンティティが親であるかどうかの確認 .....	210
EntIsModified( ) - 表が変更されているかどうかの確認 .....	211
EntIsSelected( ) - 表が選択されているかどうかの確認 .....	212
EntJournalExtract( ) - 仕訳帳の抽出 .....	213
EntJournalExtractVB( ) - 仕訳帳の抽出 .....	216
EntJournalLoad( ) - 仕訳帳の読み込み .....	220
EntJournalLoad( ) - 仕訳帳の読み込み .....	222
EntLogicAttach( ) - 計算式の追加 .....	225
EntLogicCompile( ) - ロジックのコンパイル .....	226
EntLogicDetach( ) - 計算式の分離 .....	227
EntLogicDiscard( ) - ロジックブロックの破棄 .....	228
EntLogicExport( ) - 計算式のエクスポート .....	228
EntLogicExportVB( ) - 計算式のエクスポート .....	230
EntLogicImport( ) - 計算式のインポート .....	233
EntLogicImportVB( ) - 計算式のインポート .....	234
EntMessage( ) - メッセージボックスの表示とエラーログへのメッセージの 書き込み .....	237
EntMessageVB( ) - メッセージボックスの表示とエラーログへのメッセージ の書き込み .....	238
EntMsgLogTaskEndTime( ) - エラーログへの終了時刻とタスクの書き込 み .....	239
EntMsgLogTaskStartTime( ) - エラーログへの開始時刻とタスクの書き込 み .....	240
EntMsgLogTimeVB( ) - ログファイルへの時刻とイベントの書き込み .....	241
EntMsgLogTimeVB( ) - ログファイルへの時刻とイベントの書き込み .....	241
EntOpenApplication( ) - アプリケーションを開く .....	242
EntOpenServerApplication( ) - サーバアプリケーションを開く .....	243

EntPgeExtract( ) - ページ書式および設定情報の抽出 .....	244
EntPgeLoad( ) - ページ書式および設定情報の読み込み .....	245
EntPerAsk( ) - 期間の選択 .....	245
EntQryRptFreq( ) - クエリのレポート期間単位 .....	246
EntQueryEx( ) - クエリ情報 .....	247
EntQueryDefault( ) - デフォルトのクエリ .....	248
EntRegisterMsgBox( ) - 代替 MessageBox( )ルーチンの登録 .....	249
EntRunRollover( ) - 期別替の実行 .....	250
EntSave( ) - 表の保存 .....	250
EntSaveDefault( ) - デフォルトの保存 .....	251
EntSecurityExtract( ) - セキュリティの抽出 .....	252
EntSecurityExtractVB( ) - セキュリティの抽出 .....	253
EntSecurityLoad( ) - セキュリティの読み込み .....	255
EntSecurityLoadVB( ) - セキュリティの読み込み .....	256
EntSelect( ) - 使用する表の選択 .....	258
EntSelectAdd( ) - 使用する追加の表の選択 .....	260
EntSelectTable( ) - 使用する表の選択 .....	261
EntSelectTableAdd( ) - 使用する追加の表の選択 .....	262
EntSetActiveModule( ) - アクティブモジュールの設定 .....	264
EntSharesExtract( ) - 株式の抽出 .....	264
EntSharesExtractVB( ) - 株式の抽出 .....	266
EntSharesLoad( ) - 株式の読み込み .....	268
EntSharesLoadVB( ) - 株式の読み込み .....	269
EntUNCDDataLoad( ) - データの読み込み .....	270
EntUNCDDataLoadVB( ) - データの読み込み .....	272
EntUnSelect( ) - 表の選択解除 .....	276
EntUpdate( ) - レコードの更新 .....	277
EntUpdateDefault( ) - デフォルトの更新 .....	278
EntWriteAppProfileLong( ) - アプリケーションプロファイルの long 整数の書き込み .....	279
EntWriteAppProfileString( ) - アプリケーションプロファイルの string の書き込み .....	280
EntWriteProfileLong( ) - Hyperion プロファイルの long 整数の書き込み .....	281
EntWriteProfileString( ) - Hyperion プロファイルの string の書き込み .....	281
EntVBGetCStrLen( ) - C 文字列の長さの取得 .....	282
EntVBCopyData( ) - データのコピー .....	283
EntVBCopyStr( ) - 文字列のコピー .....	283
HypGethApp( ) - アプリケーションハンドルの取得 .....	284
HypGethSelect( ) - 表選択ハンドルの取得 .....	284

HypGethSelect() - 表選択ハンドルの設定	285
<b>第 5 章 Hyperion Enterprise SE からの変換</b>	287
Hyperion Enterprise SE 変換の概要	287
Hyperion Enterprise のインクルードファイル	287
関数の分類	288
サポートされているスプレッドシートアドイン関数	288
サポートされているその他の関数	289
新しい Ent...()関数	289
サポートされていない関数	290
Hyperion Enterprise SE のコードのアップグレード	291
アプリケーションハンドルの再定義	292
16 ビットプログラムから 32 ビットプログラムへの変換	292
古い関数から新しい Hyp...()関数への置き換え	293
古い関数から新しい Ent...()関数への置き換え	295
関数のアップグレード	297
Upgrade HypLock() と HypUnLock()	298
HypFind() のアップグレード	298
HypQuery() のアップグレード	298
HypQuerySig() のアップグレード	302
HypQryNode() のアップグレード	303
HypQryAcctListEntry() と HypQryNameListEntry() のアップグレード	304
32 ビットの分割勘定科目とエンティティ記号の置き換え	306
Hyperion Enterprise の表の変更	306
名前が変更された表	306
会社間詳細表の変更	307
ノード表の変更	308
期別替表の変更	309
使用されなくなったセキュリティ表	311
セキュリティグループ表の変更	315
セキュリティユーザ表の変更	315
<b>第 6 章 コールバック関数</b>	317
コールバック関数の概要	317
コールバック関数の構文	318
CALLBACK12	318
CALLBACKAPI	319
CALLBACKAPP (または CALLBACK6)	320
CALLBACKCREATE	320
CALLBACKDBENUM	321

CALLBACKDBLOAD - EntAppExtract( ) の場合	322
CALLBACKDBLOAD - EntAppLoad( ) の場合	323
CALLBACKDBLOAD - EntDataExtract() の場合	324
CALLBACKDBLOAD - EntUNCDataLoad() の場合	324
CALLBACKJOUREXTRACT	325
CALLBACKJOURLOAD	326
CALLBACKLOGIC	326
CALLBACKSECLOAD	327
CALLBACKSEL	328
CALLBACKSHARES	329
CALLBACKSTR	330
CALLBACKUSERS	330
CALLBACKVB	331
DWCALLBACK	332
<b>付録 A. 表 ID</b>	<b>333</b>
表の情報	333
デフォルト設定の表	338
関連付けられている表	339
関連する表	340
<b>付録 B. クエリ属性</b>	<b>343</b>
クエリ属性の概要	343
デフォルトのクエリ属性	344
デフォルト以外のクエリ属性	346
ID_ACCOUNTS (勘定科目表) クエリ属性	346
ID_ACCTCONVERT (勘定科目変換表) クエリ属性	349
ID_ACCTCVTLIST (勘定科目変換一覧表) クエリ属性	349
ID_ACCTLIST (勘定科目一覧表) クエリ属性	350
ID_ACCTLISTENTRY (勘定科目一覧入力表) クエリ属性	350
ID_APPDEFAULT (アプリケーションデフォルト表) クエリ属性	352
ID_BOOK_ENTRIES (パッケージ入力表) クエリ属性	355
ID_BOOK_SETS (パッケージセット表) クエリ属性	355
ID_BOOKS (パッケージ表) クエリ属性	356
ID_CATEGORY (データ種別表) クエリ属性	356
ID_CATEGORY_LINKS (データ種別リンク表) クエリ属性	359
ID_CODES (コード表) クエリ属性	359
ID_CURRENCY (通貨表) クエリ属性	359
ID_DATAFILE (データファイル表) クエリ属性	359
ID_FORMATS (書式表) クエリ属性	361

ID_FORMULAS (計算式表) クエリ属性 .....	362
ID_FREQUENCY (期間単位表) クエリ属性 .....	363
ID_GROUP (勘定科目グループ表) クエリ属性 .....	364
ID_HAPP (アプリケーション情報表) クエリ属性 .....	365
ID_ICSET (会社間照合セット表) クエリ属性 .....	365
ID_INTCODET (会社間詳細表) クエリ属性 .....	366
ID_JOURNAL_DETAIL (仕訳帳詳細表) クエリ属性 .....	366
ID_JOURNAL_HISTORY (仕訳帳記録表) クエリ属性 .....	367
ID_JOURNAL_HISTORY_DETAIL (仕訳帳の記録の詳細表) クエリ属性 .....	369
ID_JOURNAL_PERIOD_INFO (仕訳帳の期間情報表) クエリ属性 .....	370
ID_JOURNAL_TEMPLATES (仕訳帳テンプレート表) クエリ属性 .....	371
ID_JOURNAL_TEMPLATES_DETAIL (仕訳帳テンプレートの詳細表) クエリ属性 .....	373
ID_JOURNALS (仕訳帳表) クエリ属性 .....	374
ID_LOGIC (ロジック表) クエリ属性 .....	377
ID_LOGIC_CAT_ATTRIB (ロジックデータ種別属性表) クエリ属性 ....	377
ID_NAMECONVERT (エンティティ変換表) クエリ属性 .....	378
ID_NAMECVTLIST (エンティティ変換表) クエリ属性 .....	378
ID_NAMELIST (エンティティ表) クエリ属性 .....	378
ID_NAMELISTENTRY (エンティティ一覧入力表) クエリ属性 .....	379
ID_NAMES (エンティティ表) クエリ属性 .....	380
ID_NODES (ノード表) クエリ属性 .....	384
ID_ORGANIZATION (組織表) クエリ属性 .....	386
ID_PERIOD (期間表) クエリ属性 .....	386
ID_PRINT (印刷表) クエリ属性 .....	387
ID_PSFDATA (PSF データ表) クエリ属性 .....	387
ID_REPORT_ENTRIES (レポート入力表) クエリ属性 .....	388
ID_REPORT_SETS (レポートセット表) クエリ属性 .....	389
ID_REPORTS (レポート表) クエリ属性 .....	389
ID_ROLLOVER (期別替表) クエリ属性 .....	389
ID_ROLLSET (期別替セット表) クエリ属性 .....	389
ID_RPTFREQ (レポート期間単位表) クエリ属性 .....	390
ID_RPTVIEW (レポート表示形式表) クエリ属性 .....	390
ID_RULES (更新ルール表) クエリ属性 .....	391
ID_RULESEXP (更新数式規則表) クエリ属性 .....	391
ID_RULESVAR (変数規則表) クエリ属性 .....	392
ID_SCHEDULES (データ入力表) クエリ属性 .....	393
ID_SECCLASS (セキュリティクラス表) クエリ属性 .....	394

ID_SECGRPTAB (セキュリティグループ表) クエリ属性 .....	394
ID_SECRIGHTS (セキュリティ権表) クエリ属性 .....	394
ID_SECTASK (セキュリティタスク表) クエリ属性 .....	395
ID_SECTASKFILTER (セキュリティタスクフィルタ表) クエリ属性 ....	395
ID_SECUSERTAB (セキュリティユーザ表) クエリ属性 .....	395
ID_SERVER (サーバ表) クエリ属性 .....	396
ID_SHARES (株式表) クエリ属性 .....	396
ID_SUBACCTDET (サブ勘定科目詳細表) クエリ属性 .....	398
ID_SUBACCTHDR (サブ勘定科目ヘッダー表) クエリ属性 .....	399
ID_SUBNAME (サブエンティティ表) クエリ属性 .....	399
ID_SUBSTRUCTURE (下位構造表) クエリ属性 .....	400
ID_SUGGEST_OWN (連結ロジックと連結比率の提案表) クエリ属性 ...	401
ID_USE_METHODS (使用ロジック表) クエリ属性 .....	401
ID_USERDEFAULT (ユーザデフォルト表) クエリ属性 .....	402
ID_USERDEFFUNC (カスタム関数表) クエリ属性 .....	405
<b>索引</b> .....	<b>407</b>



# 1

# Hyperion Enterprise APIについて

## この章の内容

Hyperion Enterprise API の概要 .....	15
このガイドの使用方法 .....	15
サポートされている言語 .....	16
Visual Basic プログラミング上の注意 .....	16
Visual Basic 以外のプログラムの変更 .....	19
インクルードファイル .....	19
プログラムのインポートライブラリへのリンク .....	20
戻りコード .....	20
期間単位と表示形式 .....	20

この章では、オラクル社の Hyperion(R) Enterprise(R) Application Programming Interface (API) とその使用方法についての全般的な情報を提供します。

## Hyperion Enterprise API の概要

Hyperion Enterprise API 関数を使用すると、どのプログラムからでも Hyperion Enterprise データへの高レベルのアクセスが可能です。Hyperion Enterprise API 関数は HEACCESS.DLL ファイルに収められています。

このガイドは主に Visual Basic プログラマを対象としています。Hyperion Enterprise で使用できる API 関数の使用方法を説明する中で、関数ごとのサンプルコードと C 言語コードも示します。Hyperion Enterprise SE HEACCESS プログラムを変換して Hyperion Enterprise で使用する方法も説明します。

## このガイドの使用方法

このガイドでは、Hyperion Enterprise API 関数を次のグループに分けて説明します。

- スプレッドシートアドイン関数。スプレッドシートなどのアプリケーション用に最適化されており、最も使用の容易な関数です。これらの関数はすべて Hyp...() という形式を使用します。詳しくは、[第2章「スプレッドシートのアドイン関数」](#)を参照してください。
- 表関数。スプレッドシートなどのアプリケーション用に最適化されており、最も使用の容易な関数です。これらの関数はすべて Hyp...() という形式を使用

します。詳しくは、[第2章「スプレッドシートのアドイン関数」](#)を参照してください。これらの関数は高レベルの関数に代わるもので、すべて `Hyp...()` という形式を使用します。詳しくは、[第3章「表関数」](#)を参照してください。

- 高レベル関数。基本となる Hyperion Enterprise API 関数の効力を最大限に活用します。これらの関数は通常、Hyperion Enterprise API 関数を直接呼び出します。これらの関数はすべて `Ent...()` という形式を使用します。詳しくは、[第4章「高度な関数」](#)を参照してください。
- Hyperion Enterprise SE 関数。Hyperion Enterprise で使用するにはアップデートする必要のある関数です。これらの関数を変換する方法については、[第5章「Hyperion Enterprise SE からの変換」](#)を参照してください。

通常は、簡単に使用できて、システムデータへのインパクトが比較的小さい、スプレッドシートアドイン関数と表での作業に使用する基本的な関数から始めてください。

関数に加えて、このガイドの付録には表 ID とクエリ属性に関する情報が記載されています。

## サポートされている言語

Hyperion Enterprise API は Windows の DLL ファイルの利用を可能にするプログラム言語をサポートします。次の言語がサポートされています。

- Visual Basic
- C
- C++
- Basic
- Fortran
- Pascal

当社では Microsoft Visual Basic Release 5.0 および Microsoft Visual C++ Version 5.0 で使用するインクルードファイルを提供しています。他社の Basic および C コンパイラも使用できますが、インクルードファイルやサンプルコードに変更を加える必要がある場合があります。その他の言語では独自にインクルードファイルを作成する必要があります。

## Visual Basic プログラミング上の注意

各章は基本となるトピックから構成されています。Visual Basic に適用されるトピックも含まれています。Visual Basic でのプログラミングには、各章の次のトピックを参照してください。

- 関数の宣言
- 定数
- 文字列
- アドレスの渡し方

- Visual Basic におけるコールバック関数
- Visual Basic における apiStruct 構造体の使用方法

## 関数の宣言

TOOLKIT.H ファイルではなく、TOOLKIT.BAS ファイルをプログラムにインクルードします。ファイル全体をインクルードするのではなく、必要な宣言のみをコピーすることによってメモリ問題を避けることができます。

**注：** TOOLKIT.BAS ファイルには 32 ビットおよび 16 ビットの関数宣言が含まれています。16 ビット関数は現在サポートされていません。32 ビットの関数宣言のみを使用してください。32 ビット宣言は Lib HACCESS32.DLL を使用します。古い 16 ビットの関数は Lib HACESS.DLL を使用します。

## 定数

TOOLKIT.BAS ファイルでは、定数の多くが HYP\_ で始まります。たとえば、NONE は HYP\_NONE のようになります。TOOLKIT.BAS の中で見つからない定数は、TOOLKIT.H または TOOLINC.H ファイルで見つけて、その定義のクローンを作成します。

表 ID はこのガイドでは、ID\_TABLENAME のように C 形式で識別します。Visual Basic を使用している場合は、すべての表 ID の先頭に HYP\_ を挿入します。TOOLKIT.BAS ファイルに示されているように、表 ID の正しい Visual Basic 形式は、HYP\_ID\_TABLENAME です。

このマニュアルでは、定数の TRUE および FALSE は C 標準の 1 と 0 を意味します。Visual Basic での定義の -1 と 0 ではありません。

**ヒント：** TOOLKIT.BAS ファイル全体をインクルードするのではなく、必要な定義のみをコピーすることによってメモリ問題を避けることができます。

## 文字列

string は C スタイルの文字列として返されます。Visual Basic では最初に返される値の大きさに合わせて string を割り当て、その string のアドレスを API 関数に渡します。返された string の中で数字のゼロを検索します。ゼロが string の終わりを示すマーカーとして使用されています。ゼロから後の string は切り捨てます。これは頻繁に行う操作なので、数字のゼロを検索する関数を書くこともできます。

次にその例を示します。

```
'CのstringをVisual Basicのstringに変換する関数
Private Function CToBStr(szStr$) As String
Dim p%
p% = InStr (szStr$, Chr$(0))
If p% Then szStr$ = Left (szStr$, p% -1)
CToBStr$ = szStr$
End Function
```

関数の中には引数にどんなデータタイプでも受け入れるものがあります。そのような関数は ANY として宣言されています。引数が string の場合は、使用する宣言のカスタムクローンを作成するのが最も簡単な方法です。Visual Basic で string のアドレスを渡すには、ByVal を使用して string の引数を値によって渡します。string を参照で渡すのがデフォルトですが、その場合 Visual Basic では string 自体のアドレスではなく、「BString」とも呼ばれる BASIC の string 構造体のアドレスが渡されます。

Visual Basic で定義されている定数 vbNullString は NULL を string のアドレスとして渡す必要があるときに使用できます。

## アドレスの渡し方

関数宣言は通常、アドレス渡しを自動的に処理します。詳しくは、[208 ページの「EntGetVarAddr\(\) - 変数のアドレスの取得」](#)を参照してください。構造体の中のフィールドにアドレスを値として設定する必要がある場合、この関数が役に立ちます。

## Visual Basic におけるコールバック関数

Hyperion Enterprise API 関数の中にはコールバック関数を使用するものがあります。API 関数の引数の 1 つは、その API から呼び出す関数のアドレスです。呼び出す関数を用意してそのアドレスを Hyperion Enterprise API 関数に渡します。列挙関数はコールバック関数を使用します。たとえば、列挙されるレコードごとにユーザが必要とする処理を行うために、HypEnumEx()関数はユーザが指定した関数を呼び出します。

Visual Basic バージョン 5.0 はコールバック関数をサポートします。Visual Basic の [関数のアドレス] を利用して Hyperion Enterprise API 関数にコールバック関数のアドレスを渡します。コールバック関数の中の引数の 1 つが C スタイルの string である場合、C スタイルの string を典型的な Visual Basic の string としてコピーするために EntVBGetCStrLen()および EntVBCopyData()を使用できます。

Visual Basic バージョン 4.0 ではコールバック関数を使用できません。関数のアドレスを取得することも別の関数にアドレスを渡すこともできません。Visual Basic でのコールバック関数の使用を可能にする製品があります。Desaware, Inc.製の SpyWorks もそのような製品の 1 つです。

SpyWorks はコールバック関数のコントロールを提供します。そのようなコントロールの 1 つをプログラムに追加するだけで使用できます。このコントロールのプロパティの 1 つは関数のタイプです。このプロパティに適切な関数のタイプを設定します。これはコールバック関数で必要とされる引数や戻り値のタイプに依存します。提供されているそのタイプの関数をカスタマイズして使用します。もう 1 つのプロパティ、ProcAddress プロパティは Hyperion Enterprise API 関数に渡す関数のアドレスを提供します。

関数 EntVBCopyData()および EntVBGetCStrLen()は SpyWorks のコールバック関数の引数を適切なタイプに変換するために使用できます。詳しくは、[151 ページの「EntDataExtractVB2\(\) - データの抽出」](#)を参照してください。

## Visual Basic における apiStruct 構造体の使用方法

表を操作する高度な関数の多くは、引数を持っています。この引数は、apiStruct と呼ばれる構造体です。この構造体は TOOLKIT.BAS ファイルで定義されているもので、残りの引数で扱いきれない情報を関数に渡すために使用されます。通常は apiStruct を使用する必要はありません。その場合は apiStruct 引数でゼロ（または NULL）を渡します。ただし、Visual Basic では apiStruct 引数でゼロ（または NULL）を渡すことはできません。この問題は関数宣言のクローンを作成して apiStruct 引数を定義し直すことによって回避できます。ByVal nullStruct As Long のようにこの引数を long として定義すると、ゼロを渡すことができます。ByVal nullStruct As String のように string として定義すると、定数 vbNullString を渡すことができます。引数が実際には apiStruct ではないときに、その引数にゼロまたは NULL を強制するように関数を書くことができます。

apiStruct を引数として受け取るコールバック関数を使用するときに、そのコールバック関数がその引数を APISTRUCT ではなく long として宣言している場合、EntVBCopyData 関数を使用して引数を apiStruct に変換できます。EntVBCopyData を呼び出す前にその引数がゼロになっていないことを確認します。

## Visual Basic 以外のプログラムの変更

Hyperion Enterprise API 関数呼び出す前に、プログラムの始めで COM を初期化するために、Visual Basic (VB) 以外のプログラムを変更する必要があります。これは Microsoft の OLE32.DLL の中にある CoInitialize( )関数呼び出すことによって行います。さらに、プログラムの終わりに CoUninitialize( )を呼び出します。

## インクルードファイル

Hyperion Enterprise では 2 つのインクルードファイルを提供しています。

- TOOLKIT.BAS (Microsoft Visual Basic での作業用)
- TOOLKIT.H (C および C++での作業用)

TOOLKIT.H は Microsoft Visual C++用に開発され使用されてきました。TOOLKIT.H は Hyperion Enterprise の構築に使用した TOOLINC.H を含んでいます。

Hyperion Enterprise API 関数では、シンボリック定数がマスク、パラメータ、表 ID、戻りコードなどとして使用されています。たとえば、関数 HypGetRptFreq( ) ではシンボリック定数 FREQ\_DAY、FREQ\_WEEK、FREQ\_MONTH をシステムの期間単位表現に使用します。

インクルードファイルの TOOLKIT.BAS、TOOLKIT.H および TOOLINC.H には、すべてのシンボリック定数とその値および記述とともに記載されています。TOOLKIT.BAS に記載されている情報の例を下に示します。

```
Global Const HYP_ID_ORGANIZATION = 0 Organization
Global Const HYP_ID_CATEGORY = 1 Category
Global Const HYP_ID_FREQUENCY = 2 Frequency
Global Const HYP_ID_ACCOUNTS = 3 Accounts
```

TOOLKIT.H および TOOLINC.H に記載されている情報の例を下に示します。

```
#define ORGANIZATION 0 //Organization
#define ID_CATEGORY 1 //Category
#define ID_FREQUENCY 2 //Frequency
#define ID_ACCOUNTS 3 //Accounts
```

インクルードファイルには関数宣言も含まれています。

## プログラムのインポートライブラリへのリンク

Hyperion Enterprise 関数を使用するには、プログラムを HEACCESS.LIB インポートライブラリファイルにリンクする必要があります。

## 戻りコード

通常、Hyperion Enterprise API 関数は正常に機能した場合はゼロを、エラーがあった場合は非ゼロ (NONE) を返します。API の中の関数のパラメータは変わりません。ライブラリはパラメータのローカルコピーを使用します。唯一の例外は、戻り値のためのポインタが設定されている場合です。関数の最後のパラメータが通常これに該当します。

**注：** NONE は TOOLINC.H で -1 と定義されています。Visual Basic では HYP\_NONE を使用します。これは TOOLKIT.BAS で -1 と定義されています。詳細エラーコードを返す関数もあります。エラーコードの定義も TOOLKIT.BAS および TOOLKIT.H ファイルまたは TOOLINC.H ファイルに含まれています。

## 期間単位と表示形式

このガイドで説明する関数の多くで Hyperion Enterprise 期間単位、レポート期間単位およびレポート表示形式が入力または出力として使用されます。

### Hyperion Enterprise 期間単位

表 1 に示されている期間単位は、Hyperion Enterprise にアプリケーションをインストールしてセットアップするときに使用する LOAD.PER ファイルに含まれています。詳しくは、『Hyperion Enterprise 管理者用ガイド』の「アプリケーションの設定」の章を参照してください。

表 1 Hyperion Enterprise 期間単位

期間単位	説明
D	日次
W	週次
M	月次

期間単位	説明
Q	四半期
T	三半期
H	半期
Y	年次

表 2 に示されているレポート期間単位は Hyperion Enterprise API 関数で利用できるものです。

**表 2** レポート期間単位

期間単位	説明
DAI	日次
DAY	日次
DYDT	日次データ種別累計
WEE	週次
WYTD	週次データ種別累計
MON	月次
YTD	月次データ種別累計
QUA	四半期
QYTD	四半期データ種別累計
TRI	三半期
TYTD	三半期データ種別累計
HAL	半期
HYTD	半期データ種別累計
YEA	年次

## レポートの表示形式

表 3 に示されているレポートの表示形式は Hyperion Enterprise API 関数で利用できるものです。

**表 3** レポートの表示形式

表示形式	説明
PER	期別
WTD	週次累計

表示形式	説明
MTD	月次累計
QTD	四半期累計
TTD	三半期累計
HTD	半期累計
CTD	データ種別累計
YTD	年次累計

# 2

## スプレッドシートのアドイン関数

### この章の内容

スプレッドシートのアドイン関数の概要.....	23
単一アプリケーションの管理関数.....	23
複数アプリケーションの管理関数.....	24
選択ダイアログボックス関数.....	24
データ取得関数.....	25
データ更新関数.....	26
アルファベット順の関数リファレンス.....	27

この章には、構文や例など、スプレッドシートのアドイン関数を使用するために必要な情報が記載されています。

## スプレッドシートのアドイン関数の概要

スプレッドシートのアドイン関数は、Hyperion Enterprise API システムで最も一般的に使用される関数です。これらの関数は、Hyperion Enterprise Retrieve などのスプレッドシートアプリケーション用に最適化されているという点で、基本となる Hyperion Enterprise API 関数より付加価値があります。アドイン関数のほとんどは読み取り専用ですが、データ値を書き込むこともできます。また、システムデータにもたらす影響が比較的小さいので、最も安全な関数と言えます。形式は常に Hyp...() になります。

実質的にすべてのスプレッドシートアドイン関数に対して、Hyperion Enterprise Retrieve アプリケーションハンドルである引数を使用する必要があります (C 言語のプログラマの場合はタイプ HRETRIEVEAPP)。Hyperion Enterprise Retrieve は、Hyperion Enterprise のスプレッドシートアドイン製品です。このハンドルを取得するには、HypConstruct() を呼び出してアプリケーションを起動および初期化するか、HypMultiGet() または HypMultiDefault() を呼び出します。

## 単一アプリケーションの管理関数

個々の Hyperion Enterprise アプリケーションを開いたり閉じたりするには、単一アプリケーションの管理関数を使用します。

表 4 に、単一アプリケーションの管理関数を示します。

表 4 単一アプリケーションの管理関数

関数	説明
HypConstruct( )	単一のアプリケーションを開きます。これにより、そのアプリケーションの他の関数を呼び出せるようになります。
HypDestruct( )	使用し終えた単一のアプリケーションをプログラム終了前に閉じます。

HypConstruct( )からは、現在のセッションのアプリケーションを一意に識別する値が取得されます。取得した値は、他の任意の関数に渡すことができます。C 言語では、この値は、Hyperion Enterprise Retrieve アプリケーションへのハンドルであるタイプ HRETRIEVEAPP として定義されます。一度に最大 20 (MAX\_OPEN\_APPS) 個のアプリケーションを開くことができます。

## 複数アプリケーションの管理関数

複数のアプリケーションを対象としてログオンやアプリケーションハンドルの取得などのタスクを実行するには、複数アプリケーションの管理関数を使用します。表 5 に、複数アプリケーションの管理関数を示します。

表 5 複数アプリケーションの管理関数

関数	説明
HypMultiAsk( )	開いているアプリケーションの一覧からアプリケーションを選択し、それをデフォルトのアプリケーションに設定するために使用されます。
HypMultiDefault( )	デフォルトのアプリケーションのハンドルを取得します。
HypMultiDeinit( )	使用し終えた複数のアプリケーションをプログラム終了前に閉じます。
HypMultiEnum( )	アプリケーションを列挙します。
HypMultiGet( )	Hyperion Enterprise Retrieve アプリケーションハンドルを取得します。
HypMultiInit( )	ファイル内にリストされた複数のアプリケーションを同時に開きます。

## 選択ダイアログボックス関数

API の標準の選択ダイアログボックスにアクセスするには、選択ダイアログボックス関数を使用します。選択ダイアログボックスは、勘定科目、勘定科目一覧、データ種別、期間単位、エンティティおよび期間を選択するために使用します。選択ダイアログボックス関数を使用するには、HypConstruct( )または HypMultiInit( )を使用してアプリケーションを開く必要があります。表 6 に、選択ダイアログボックス関数を示します。

表 6 選択ダイアログボックス関数

関数	説明
HypAcctAsk( )	勘定科目の選択に使用します。

関数	説明
HypAcctListAskEx( )	勘定科目一覧の選択に使用します。
HypCatAskEx( )	データ種別の選択に使用します。
HypFreqAsk( )	期間単位の選択に使用します。
HypJourAsk( )	仕訳帳の選択に使用します。
HypJourDetAsk( )	仕訳帳の詳細の選択に使用します。
HypNamAskEx( )	エンティティの選択に使用します。
HypNameListAskEx( )	エンティティ一覧の選択に使用します。
HypPerAsk( )	期間の選択に使用します。

## データ取得関数

Hyperion Enterprise データベースからデータ値を取得するには、データ取得関数を使用します。表 7 にデータ取得関数を示します。

**注：** 通常 of の値を取得する場合は HypHPVAL( )を使用し、調整後の値、比率値、調整値、通貨換算値などの特殊な値を取得する場合は HypHPVALEx( )を使用します。HypHPVAL2 は、いずれの場合にも使用できます。

表 7 データ取得関数

関数	説明
HypHPVAL( )	Hyperion Enterprise データベースからデータ値を取得します。
HypHPVAL2( )	Hyperion Enterprise データベースからデータ値を取得します。
HypHPVALEx( )	Hyperion Enterprise データベースから特定の値を取得します。
HypSetMVMode( )	HypValReturn( )の動作を設定します。
HypValidateParams( )	HypHPVAL( )のパラメータを検証します。
HypValidateParamsEx( )	HypHPVALEx( )のパラメータを検証します。
HypValReturn( )	最後に行ったデータ取得のステータスを判別します。

## スプレッドシートのアドイン関数との連結詳細値

Hyperion Enterprise アプリケーションでは、消去、調整、通貨換算値、エンティティが連結中にその親に搬出した値などの連結詳細を格納できます。この機能は、ヘッダーファイルでは詳細記憶モデル（DSM）と呼ばれています。

ID\_TRANSLATION\_FORCE は ID\_TRANSLATION と似ていますが、親および子のエンティティが同じ通貨を使用している場合でも有効です。通貨が同じ場合、データは換算されず、換算データファイル表は存在しません。この場合に ID\_TRANSLATION を使用すると、換算データファイル表が存在しないのでエラーコードが返されます。代わりに ID\_TRANSLATION\_FORCE を使用すると、親と子が同じ通貨を使用している場合に、エラーコードが返される代わりに通常のデータ（タイプ ID\_REGULAR）にアクセスされます。

警告：指定されているデータ種別とエンティティに対して ID\_REGULAR データファイル表が既に選択されている可能性がある場合、ID\_TRANSLATION\_FORCE タイプのデータファイル表は選択しないでください。同様に、ID\_TRANSLATION\_FORCE タイプが既に選択されている可能性がある場合、ID\_REGULAR タイプのデータファイル表は選択しないでください。

これらの連結詳細値を読み取るには、表 8 に示す例外を除いて 25 ページの「データ取得関数」の手順に従ってください。

表 8 wType コード

コード	説明
ID_ADJUSTMENT	親の調整値
ID_CONTRIBUTION	調整後の値
ID_ELIMINATION	消去値
ID_PROPORTIONAL	比率値
ID_TRANSLATION	換算値
ID_TRANSLATION_FORCE	通貨が同じ場合は換算または子

**注：** 連結詳細値の更新は行わないでください。これらの値は読み取り専用です。これらの値は、連結値の計算方法を示す記録として Hyperion Enterprise により連結中に書き込まれます。

## データ更新関数

Hyperion Enterprise のデータを更新するには、データ更新関数を使用します。表 9 にデータ更新関数を示します。

表 9 データ更新関数

関数	説明
HypHPCommit2( )	計算式を計算せずにデータバッファを書き込んでディスクに保存します。
HypHPFlush( )	データバッファをクリアします。
HypHPLNK( )	データ値をメモリ内のバッファに保存します。
HypValidateLnkParams( )	HypHPLNK( ) パラメータをチェックします。

通常、Hyperion Enterprise のデータを更新するには、次のタスクを実行します。

- HypHPFlush( )を呼び出してデータバッファをクリアします。
- 書き込むデータごとに HypHPLNK( )を呼び出します。HypHPLNK( )によってデータがバッファのみに書き込まれます。
- HypHPCommit2( )を呼び出して、処理が完了したらバッファ内のデータをディスクに書き込みます。ディスクに書き込まれる前のデータが HypHPCommit2( )によってソートされます。

HypHPLNK( )を呼び出してデータ値を更新した後で、ディスクに保存しないことに決めた場合は、HypHPFlush( )を呼び出してバッファをクリアします。バッファをクリアすると、後で HypHPCommit2( )を呼び出したときにデータが間違っ保存されるのを防ぐことができます。

## アルファベット順の関数リファレンス

ここでは、すべてのスプレッドシートアドイン関数をアルファベット順に説明します。

### HypAcctAsk( ) - 勘定科目の選択

この関数は、Hyperion Enterprise の「勘定科目の選択」ダイアログボックスを呼び出して、選択された勘定科目を取得します。「キャンセル」を選択した場合、勘定科目は取得されません。

次の形式を使用します。

Declare Function HypAcctAsk Lib "heaccess.dll" Alias "\_HypAcctAsk@8" (ByVal hRApp As Integer, ByVal szAcct As String) As Long

#### 変数 説明

hRApp HypMultiGet( )または HypConstruct( )からのゼロ以外の Hyperion Enterprise Retrieve アプリケーションハンドル

szAcct 勘定科目 ID を返すためのバッファを指すポインタ。バッファのサイズは、少なくとも HYP\_SIZEFULLACCT+1 文字（または C 言語では SIZEFULLACCT+1）にする必要があります。

#### 戻りコード

コード	説明
0	成功
なし	エラー発生

例：

```
szAcct$ = SPACE (HYP_SIZEFULLACCT+1)
ret& = HypAcctAsk(hRApp%, szAcct$)
If ret& = 0 Then szAcct$ = CToStr(szAcct$)
```

C 言語では次の形式を使用します。

short WINAPI HypAcctAsk( HRETRIEVEAPP hRApp, LPSTR szAcct )

## HypAcctListAskEx( ) - 勘定科目一覧の選択

この関数を実行すると、勘定科目一覧の選択に使用するダイアログボックスが表示されます。HypAcctListAskEx()では、選択した勘定科目一覧の ID が戻りバッファ (szRetbuf) で返されるか、[キャンセル] を選択した場合は空の文字列が返されます。この関数では、勘定科目一覧表が一時的に保護または保護解除されます (ID\_ACCTLIST)。

次の形式を使用します。

Declare Function HypAcctListAskEx Lib "heaccess.dll" Alias  
" \_HypAcctListAskEx@8" (ByVal hRApp As Integer, ByVal szAcctList As String) As Integer

### 変数 説明

hRApp HypConstruct( )からのアプリケーションハンドル

szAcctList 選択した勘定科目一覧を返すためのバッファを指すポインタ。バッファの長さは、少なくとも HYP\_SIZELABEL+1 文字 (C 言語では SIZENAME+1 文字) にする必要があります。

戻りコード：

コード	説明
0	成功
NONE	エラー発生

例：

```
szAcctList$ = SPACE (HYP_SIZELABEL+1)
ret$ = HypAcctListAskEx(hRApp%, szAcctList$)
If ret$ = 0 Then szAcctList$ = CToBStr(szAcctList$)
```

C 言語では次の形式を使用します。

short WINAPI HypAcctListAskEx(HRETRIEVEAPP hRApp, LPSTR szAcctList)

## HypCatAskEx( ) - データ種別の選択

この関数は、Hyperion Enterprise の [データ種別の選択] ダイアログボックスを呼び出して、選択されたデータ種別を取得します。[キャンセル] を選択した場合、データ種別は取得されません。

次の形式を使用します。

Declare Function HypCatAskEx Lib "heaccess.dll" Alias " \_HypCatAskEx@8" (ByVal  
hRApp As Integer, ByVal szCat As String) As Integer

## 変数 説明

hRApp HypMultiGet( )、HypConstruct( )または HypMultiDefault( )からのゼロ以外の Hyperion Enterprise Retrieve アプリケーションハンドル

szCat データ種別 ID を返すためのバッファを指すポインタ。バッファのサイズは、少なくとも HYP\_SIZELABEL+1 文字（または C 言語では SIZECAT+1）にする必要があります。

戻りコード：

コード	説明
0	成功
NONE	エラー発生

例：

```
szCat$ = SPACE (HYP_SIZELABEL+1)
ret$ = HypCatAskEx(hRApp%, szCat$)
If ret$ = 0 Then szCat$ = CToBStr(szCat$)
```

C 言語では次の形式を使用します。

```
short WINAPI HypCatAskEx( HRETRIEVEAPP hRApp,LPSTR szCat)
```

## HypConstruct( ) - アプリケーションを開く

この関数を実行すると、指定した Hyperion Enterprise アプリケーションが開きます。この Hyperion Enterprise アプリケーションの ID は、HYPENT.INI ファイルに表示されている必要があります。ゼロが返される場合は、hRApp にゼロ以外の Hyperion Enterprise Retrieve アプリケーションハンドルが含まれています。このハンドルを使用すると、他のスプレッドシート関数を呼び出してアプリケーション情報にアクセスできます。一度に最大 20 (MAX\_OPEN\_APPS) 個のアプリケーションを開くことができます。アプリケーションの使用が終わったら、プログラムの終了前に HypDestruct( )を呼び出してアプリケーションを終了します。

HypConstruct( )を使用して開くアプリケーションが HYPENT.INI でパラメータ AppID =によって指定されたアプリケーションと一致する場合、デフォルトはこのアプリケーションに設定されます。HypMultiDefault( )では、HypMultiAsk( )で別のアプリケーションを選択しない限り、このアプリケーションが返されます。

次の形式を使用します。

```
Declare Function HypConstruct Lib "heaccess.dll" Alias "_HypConstruct@16"
(hRApp As Integer, ByVal szApp As String, ByVal szUser As String, ByVal szPassword As String) As Long
```

## 変数 説明

hRApp Hyperion Enterprise Retrieve アプリケーションハンドルのアドレス。関数によってこのハンドルがここに配置されます。

szApp 開くアプリケーションの名前

## 変数 説明

szUser アプリケーションにログオンするために使用するユーザ ID  
szPassword アプリケーションにログオンするために使用するパスワード

戻りコード：

コード	説明
0	成功
NONE	エラー発生
HACC_BAD_CODE	無効な引数
HACC_BAD_PASSWORD	無効なパスワード
HACC_BAD_USERID	無効なユーザ ID
HACC_ERROR_USER_LOCKED	ユーザ ID が保護されている
HACC_ERROR_SELECT_FAILED	表を選択する際のエラー
HACC_BAD_OPENAPP (16)	アプリケーションを開く際のエラー
HACC_TOOMANY_APPS (15)	開いているアプリケーションが多すぎる
HACC_BAD_CODE	無効な引数

例：

```
'TAXアプリケーションを開く
ret& = HypConstruct (hRApp%, "TAX", "MARY","secret")
If ret& = 0 Then ret2% = MsgBox("Successful logon", vbOKOnly, "")
```

C 言語では次の形式を使用します。

```
int WINAPI HypConstruct(HRETRIEVEAPP * pRApp, LPSTR szApp, LPSTR szUser,
LPSTR szPassword )
```

## HypConstructEx( ) - アプリケーションを開く

この関数を実行すると、指定した Hyperion Enterprise アプリケーションが開きます。この Hyperion Enterprise アプリケーションの名前は、HYPENT.INI ファイルに表示されている必要があります。HypConstructEx( )は、HypConstruct( )の拡張で、プログラムがサーバとクライアントコンピュータのどちらで実行されているかを示す追加の引数を含んでいます。

次の形式を使用します。

```
Declare Function HypConstructEx Lib "heaccess.dll" Alias "_HypConstructEx@20"
(hRApp As Integer, ByVal szApp As String, ByVal szUser As String, ByVal szPassword As
String, szServerFlag As Long) As Long
```

変数	説明
hRApp	Hyperion Enterprise Retrieve アプリケーションハンドルのアドレス。関数によってこのハンドルがここに配置されます。
szApp	開くアプリケーションの名前
szUser	アプリケーションにログオンするために使用するユーザ ID
szPassword	アプリケーションにログオンするために使用するパスワード
fxServerFlag	プログラムがサーバ (True) またはクライアント (False) のどちらで実行されているかを識別するフラグ

戻りコード：

コード	説明
0	成功
NONE	エラー発生
HACC_BAD_CODE	無効な引数
HACC_BAD_OPENAPP	アプリケーションを開く際のエラー
HACC_BAD_PASSWORD	無効なパスワード
HACC_BAD_USERID	無効なユーザ ID
HACC_ERROR_USER_LOCKED	ユーザ ID が保護されている
HACC_ERROR_SELECT_FAILED	表を選択する際のエラー
HACC_TOOMANY_APPS	開いているアプリケーションが多すぎる

例：

```
'TAXアプリケーションを開く
ret& = HypConstructEx(hRApp%, "TAX", "MARY","secret",False)
If ret& = 0 Then ret2% = MsgBox("Successful logon", vbOKOnly, "")
```

C 言語では次の形式を使用します。

```
int WINAPI HypConstructEx(HRETREIVEAPP FAR * phRApp, LPSTR szApp, LPSTR
szUser, LPSTR szPassword, BOOL fServerFlag)
```

## HypDestruct( ) - アプリケーションを閉じる

この関数を実行すると、HypConstruct( )またはHypMultiInit( )を使用して開いたすべてのアプリケーションが閉じられます。

次の形式を使用します。

```
Declare Function HypDestruct Lib "heaccess.dll" Alias "_HypDestruct@4" (ByVal
hRApp As Integer) As Long
```

ここで、hRApp は HypConstruct( )からのゼロ以外の Hyperion Enterprise Retrieve アプリケーションハンドルです。

戻りコード：

コード	説明
0	成功

例：

```
Public Sub DestructApp( )
Dim rc&
If hRApp% <> 0 Then
    rc& = HypDestruct(hRApp%)
If rc& <> 0 Then ret% = MsgBox("Logoff failed", vbOKOnly, "Error")
hRApp% = 0
End If
End Sub
```

C 言語では次の形式を使用します。

```
int WINAPI HypDestruct( HRETREIVEAPP hRApp)
```

## HypFreqAsk( ) - 期間単位の選択

この関数は、Hyperion Enterprise の「期間単位の選択」ダイアログボックスを呼び出し、選択された期間単位を取得します。「キャンセル」を選択した場合、期間単位は取得されません。

期間単位は freq.view として返されます。ここで、freq は期間単位 ID、view は表示形式です。例えば、M.YTD は年次累計表示形式を使用した月次期間単位です。表示形式はレポートであり、Hyperion Enterprise ではありません。レポートの表示形式は、PER（期別）、WTD（週次累計）、MTD（月次累計）、QTD（四半期累計）、TTD（三半期累計）、HTD（半期累計）、CTD（データ種別累計）および YTD（年次累計）です。期間単位と表示形式については、[20 ページの「期間単位と表示形式」](#)を参照してください。

次の形式を使用します。

```
Declare Function HypFreqAsk Lib "heaccess.dll" Alias "_HypFreqAsk@8" (ByVal hRApp As Integer, ByVal szFreq As String) As Long
```

### 変数 説明

hRApp HypMultiGet( )または HypConstruct( )からのゼロ以外の Hyperion Enterprise Retrieve アプリケーションハンドル

szFreq 期間単位コードを返すためのバッファを指すポインタ。バッファの長さは少なくとも 18 文字（C 言語では 2x(OLDSIZENAME+1)文字）にする必要があります。

戻りコード：

コード	説明
0	成功
NONE	エラー発生

例：

```
szFreqView$ = SPACE(18)
ret& = HypFreqAsk(hRApp%, szFreqView$)
If ret$ = 0 Then szFreqView$ = CToBStr(szFreqView$)
```

C 言語では次の形式を使用します。

```
int WINAPI HypFreqAsk( HRETRIEVEAPP hRApp, LPSTR szFreq)
```

## HypGetDefJour( ) - デフォルト仕訳帳の取得

この関数を実行すると、デフォルト仕訳帳の記号が取得されます。通常、このデフォルト値は、「仕訳帳の選択」ダイアログボックスで選択されます。

HypGetDefJour()によって返される実際の仕訳帳記号は、仕訳帳記号、OR 演算子、そして仕訳帳がどの仕訳帳表に含まれているかを示すビットフラグの組み合わせです。ビットフラグが設定されているかどうかをテストするには、JOUR\_ENTRIES 定数を使用します。JOUR\_ENTRIES ビットフラグが設定されている場合、仕訳帳は通常の仕訳帳表に含まれ、それ以外の場合は仕訳帳テンプレート表に含まれています。詳しくは、[62 ページの「HypJourAsk\( \) - 仕訳帳の選択」](#)を参照してください。

例：

```
sigJour& = HypGetDefJour(hRApp%)
If (sigJour& And JOUR_ENTRIES) <> 0 Then
wTableID% = HYP_ID_JOURNAL_ENTRIES
sigJour& = sigJour& And (Not JOUR_ENTRIES)
Else
wTableID% = HYP_ID_JOURNAL_TEMPLATES
End If
```

次の形式を使用します。

```
Declare Function HypGetDefJour Lib "heaccess.dll" Alias "_HypGetDefJour@4"
(ByVal hRApp As Integer) As Long
```

ここで、hRApp は HypConstruct( )からのゼロ以外の Hyperion Enterprise Retrieve アプリケーションハンドルです。

戻りコード：

コード	説明
デフォルト仕訳帳の記号	成功
NONE	エラー発生

C 言語では次の形式を使用します。

```
SIGNA WINAPI HypGetDefJour( HRETRIEVEAPP hRApp)
```

## HypGetDefJourCat( ) - デフォルト仕訳帳データ種別の取得

この関数を実行すると、デフォルト仕訳帳データ種別の記号が取得されます。通常、このデフォルト値は、[仕訳帳の選択] ダイアログボックスで選択されます。詳しくは、[62 ページの「HypJourAsk\( \) - 仕訳帳の選択」](#)を参照してください。

次の形式を使用します。

**Declare Function HypGetDefJourCat Lib “heaccess.dll” Alias  
“\_HypGetDefJourCat@4” (ByVal hRApp As Integer) As Long**

ここで、hRApp は HypConstruct( )からのゼロ以外の Hyperion Enterprise Retrieve アプリケーションハンドルです。

戻りコード：

コード	説明
デフォルト仕訳帳データ種別の記号	成功
NONE	エラー発生

C 言語では次の形式を使用します。

**SIGNA WINAPI HypGetDefJourCat( HRETRIEVEAPP hRApp)**

## HypGetDefJourPer( ) - デフォルト仕訳帳期間の取得

この関数を実行すると、デフォルト仕訳帳期間の記号が取得されます。通常、このデフォルト値は、[仕訳帳の選択] ダイアログボックスで選択されます。詳しくは、[62 ページの「HypJourAsk\( \) - 仕訳帳の選択」](#)を参照してください。

次の形式を使用します。

**Declare Function HypGetDefJourPer Lib “heaccess.dll” Alias  
“\_HypGetDefJourPer@4” (ByVal hRApp As Integer) As Long**

ここで、hRApp は HypConstruct( )からのゼロ以外の Hyperion Enterprise Retrieve アプリケーションハンドルです。

戻りコード：

コード	説明
デフォルト仕訳帳期間の記号	成功
NONE	エラー発生

C 言語では次の形式を使用します。

**SIGNA WINAPI HypGetDefJourPer( HRETRIEVEAPP hRApp)**

# HypHPACC( ) - 勘定科目 ID またはサブ勘定科目 ID の取得

この関数を実行すると、表 10 のコードに基づいて、勘定科目またはサブ勘定科目の ID が取得されます。

表 10 HypHPACC( ) 勘定科目ラベルおよびサブ勘定科目ラベルの戻りコード

コード	返される ID
ACC_ACC	完全な勘定科目 ID。
ACC_AMJ	主要勘定科目 ID。
ACC_AS1	第 1 レベルサブ勘定科目 ID。
ACC_AS2	第 2 レベルサブ勘定科目ラベル見出し。

次の形式を使用します。

```
Declare Function HypHPACC Lib "heaccess.dll" Alias "_HypHPACC@16" (ByVal hRApp As Integer, ByVal szAcct As String, ByVal code As Long, ByVal retbuf As String) As Long
```

## 変数 説明

hRApp HypConstruct( )からの Hyperion Enterprise Retrieve アプリケーションハンドル

szAcct 勘定科目の ID

code 次のいずれかの取得コード：

- ACC\_ACC
- ACC\_AMJ
- ACC\_AS1
- ACC\_AS2

retbuf 勘定科目 ID を返すためのバッファを指すポインタ。バッファの長さは、少なくとも HYP\_SIZEDESC 文字（C 言語では SIZEDESC）にする必要があります。

戻りコード：

コード	説明
0	成功
HACC_BAD_ACCOUNT	エラー発生、無効な szAcct
HACC_BAD_SELECT	エラー発生

例：

```
szAcct$ = SPACE (HYP_SIZEDESC)
ret& = HypHPACC(hRApp, "SALES", ACC_ACC, szAcct$)
If ret& = 0 Then szAcct$ = CToBStr(szAcct$)
```

C 言語では次の形式を使用します。

```
int WINAPI HypHPACC(HRETRIEVEAPP hRApp, LPCSTR szAcct, int iCode, LPSTR retbuf)
```

## HypHPBET( ) - 優劣値の算出

ある勘定科目に 2 つの値がある場合にこの関数を実行すると、2 つの値の差異が算出され、一方の値がもう一方の値より優れているか劣っているかが結果として返されます。優劣値を表す記号は、指定した勘定科目が費用、資産、収益または負債のいずれであるかによって決まります。

次の形式を使用します。

```
Declare Function HypHPBET Lib "heaccess.dll" Alias "_HypHPBET@20" (ByVal hRApp As Integer, ByVal szAcct As String, value2 As Double, value1 As Double, dretval As Double) As Long
```

### 変数 説明

hRApp HypConstruct( )からの Hyperion Enterprise Retrieve アプリケーションハンドル

szAcct 使用する勘定科目の ID

value2 遅い方の期間値

value1 早い方の期間値

dretval 結果が返されるバッファ

戻りコード :

コード	説明
0	成功
HACC_BAD_ACCOUNT	エラー発生、無効な szAcct
HACC_BAD_SELECT	エラー発生

例 :

```
ret& = HypHPBET(hRApp%, "INCOME", 200.0, 100.0, amtBetter#)
```

C 言語では次の形式を使用します。

```
int WINAPI HypHPBET( HRETRIEVEAPP hRApp, LPCSTR szAcct, const double FAR * pdValue2, const double FAR * pdValue1, double FAR * pdRetVal)
```

## HypHPCAD( ) - 通貨の説明の取得

この関数を実行すると、指定したエンティティの通貨の説明が取得されます。

次の形式を使用します。

Declare Function HypHPCAD Lib "heaccess.dll" Alias "\_HypHPCAD@12" (ByVal hRApp As Integer, ByVal szEntity As String, ByVal szRetbuf As String) As Long

#### 変数 説明

hRApp HypConstruct( )からの Hyperion Enterprise Retrieve アプリケーションハンドル

szEntity エンティティ ID

szRetbuf 通貨の説明を返すためのバッファ。バッファの長さは、少なくとも HYP\_SIZEDESC+1 文字（C 言語の場合は SIZEDESC+1 文字）にする必要があります。

戻りコード：

コード	説明
0	成功
HACC_BAD_NAME	エラー発生、無効な szEntity
HACC_BAD_SELECT	エラー発生

例：

```
szCurrency$ = SPACE (HYP_SIZEDESC+1)
ret& = HypHPCAD(hRApp%, "CANADA", szCurrency$)
If ret& = 0 Then szCurrency$ = CToBStr(szCurrency$)
```

C 言語では次の形式を使用します。

int WINAPI HypHPCAD( HRETREIVEAPP hRApp, LPCSTR szEntity, LPSTR szRetbuf)

## HypHPCAL( ) - 算出勘定の判別

この関数を実行すると、指定した勘定科目が算出勘定であるかどうか判別されます。

次の形式を使用します。

Declare Function HypHPCAL Lib "heaccess.dll" Alias "\_HypHPCAL@20" (ByVal hRApp As Integer, ByVal szEntity As String, ByVal szCat As String, ByVal szAcct As String, wCal As Integer) As Long

#### 変数 説明

hRApp HypConstruct( )からの Hyperion Enterprise Retrieve アプリケーションハンドル

szEntity エンティティ ID

szCat データ種別 ID

szAcct 勘定科目 ID

wCal 結果が返されるバッファ

戻りコード：

コード	説明
0	成功
HACC_BAD_NAME	エラー発生、無効な szEntity
HACC_BAD_CATEGORY	エラー発生、無効な szCat
HACC_BAD_ACCOUNT	エラー発生、無効な szAcct
HACC_BAD_SELECT	エラー発生
NONE	データファイルを開く際のエラー

戻り値では、wCal の値は表 11 に示す値になります。

**表 11** wCal に指定される可能性がある値

値	説明
HPCAL_CALC_CALC_ACCT (1)	勘定科目が算出勘定であるかどうかを指定します。
HPCAL_NOT_CALC_ACCT (-1)	勘定科目が算出勘定でないかどうかを指定します。
0	エラーが発生したかどうかを指定します。

例：

```
ret& = HypHPCal(hRApp%, "France", "Actual", "Sales", wTync%)
If ret&=0 And wType% =HPCAL_CALC_CALC_ACCT Then
    ret%=MsgBox ("Calculated Account")
```

C 言語では次の形式を使用します。

```
int WINAPI HypHPCAL( HRETRIEVEAPP hRApp, LPCSTR szEntity, LPCSTR szCat,
LPCSTR szAcct, short FAR * pwCal)
```

## HypHPCDE( ) - データ種別の説明の取得

この関数を実行すると、指定したデータ種別の完全な説明が取得されます。

次の形式を使用します。

```
Declare Function HypHPCDE Lib "heaccess.dll" Alias "_HypHPCDE@12" (ByVal
hRApp As Integer, ByVal szCat As String, ByVal szRetbuf As String) As Long
```

### 値 説明

hRApp HypConstruct( )からの Hyperion Enterprise Retrieve アプリケーションハンドル

szCat データ種別の ID

szRetbuf データ種別の説明を返すためのバッファ。バッファの長さは、少なくとも HYP\_SIZEDESC 文字（C 言語の場合は SIZEDESC 文字）にする必要があります。

戻りコード：

コード	説明
0	成功
HACC_BAD_CATEGORY	エラー発生、無効な szCat
HACC_BAD_SELECT	エラー発生

例：

```
szCat$ = SPACE (HYP_SIZEDESC)
ret& = HypHPCDE(hRApp%, "ACTUAL", szCat$)
If ret& = 0 Then szCat$ = CToBStr(szCat$)
```

C 言語では次の形式を使用します。

```
short WINAPI HypHPCDE( HRETRIEVEAPP hRApp, LPCSTR szCat, LPSTR szRetbuf)
```

## HypHPCommit2( ) - データバッファの書き込み

この関数を実行すると、書き込みバッファを前回コミット（保存）またはフラッシュした後で HypHPLNK( ) 関数呼び出しを使用して加えたすべての変更がコミットされます。HypHPCommit2( ) では、HypHPLNK( ) 呼び出しを使用して送信されたデータベースにデータが書き込まれ、ロジックの実行とセキュリティのチェックのための引数が含まれます。HypHPCommit2( ) 関数では、引数 fCheckSecurity を TRUE (1) に設定すると、現在のユーザが Hyperion Enterprise Retrieve からデータをエクスポートするためのセキュリティ権を持っている場合に限り変更がコミットされます。コミットされたデータは、すべてバッファからクリアされます。

次の形式を使用します。

```
Declare Function HypHPCommit2 Lib "heaccess.dll" Alias "_HypHPCommit2@12"
(ByVal hRApp As Integer, ByVal fDoLogic As Long, ByVal fCheckSecurity As Long) As Integer
```

### 値 説明

**hRApp** HypConstruct( ) または HypMultiGet( ) からのゼロ以外の Hyperion Enterprise Retrieve アプリケーションハンドル

**fDoLogic** ロジックを実行する場合は TRUE (1)、それ以外の場合は FALSE (0)

**fCheckSecurity** セキュリティをチェックする場合は TRUE (1)、それ以外の場合は FALSE (0)

戻りコード：

コード	説明
0	成功
NONE	エラー発生

**注：** エラーが発生した場合は、HypHPLNK( ) によって送信されたデータの一部がデータベースに書き込まれることがあります。

例：

```
If SetHypValue$ <> "" Then Exit Function
rc& = HypHPCCommit2(hRApp%, 1, 1)
If rc& <> 0 Then
    SetHypValue$ = "Error in HypHPCCommit2 - " & Str$(rc&)
    Exit Function
End If
End Function
```

C 言語では次の形式を使用します。

```
int WINAPI HypHPCCommit2(HRETRIEVEAPP hRApp, BOOL fDoLogic, BOOL
fCheckSecurity)
```

## HypHPCUREx( ) - コンポーネントの通貨の取得

この関数を実行すると、指定したエンティティの通貨の ID が取得されます。

次の形式を使用します。

```
Declare Function HypHPCUREx Lib "heaccess.dll" Alias "_HypHPCUREx@12"
(ByVal hRApp As Integer, ByVal szEntity As String, ByVal szRetbuf As String) As Integer
```

### 値 説明

hRApp HypConstruct( )からの Hyperion Enterprise Retrieve アプリケーションハンドル

szEntity エンティティの ID

szRetbuf 通貨 ID を返すためのバッファ。バッファのサイズは、少なくとも HYP\_SIZELABEL+1 文字（または C 言語では SIZENAME+1）にする必要があります。

戻りコード：

コード	説明
0	成功
HACC_BAD_NAME	エラー発生、無効な szEntity
HACC_BAD_SELECT	エラー発生

例：

```
Public Function GetEntityCurrency(Entity As String) As String
Dim rc As Integer
Dim RetBuffer As String
RetBuffer$ = SPACE (HYP_SIZELABEL+1)
rc% = HypHPCUREx(hRApp%, Entity$, RetBuffer$)
If rc% = -1 Then
    GetEntityCurrency$ = "Error in HypHPCUREx"
Else
    GetEntityCurrency$ = CToBStr(RetBuffer$)
End If
End Function
```

C 言語では次の形式を使用します。

**short** WINAPI HypHPCUREx( HRETRIEVEAPP hRApp, LPCSTR szEntity, LPSTR szRetbuf)

## HypHPDRV( ) - 派生値の算出

ある勘定科目の値が 2 つある場合にこの関数を実行すると、派生値が返されます。フロー勘定科目の場合は、2 つの値が足し合わされます。残高勘定科目の場合は、遅い方の期間値が返されます。

次の形式を使用します。

**Declare Function** HypHPDRV Lib “heaccess.dll” Alias “\_HypHPDRV@20” (ByVal hRApp As Integer, ByVal szAcct As String, value2 As Double, value1 As Double, dretval As Double) As Long

### 値 説明

hRApp HypConstruct( )からの Hyperion Enterprise Retrieve アプリケーションハンドル

szAcct value1 と value2 が表す勘定科目の ID

value2 遅い方の期間値

value1 早い方の期間値

dretval 浮動小数点値を返すためのバッファ

戻りコード：

コード	説明
0	成功
HACC_BAD_ACCOUNT	エラー発生、無効な szAcct
HACC_BAD_SELECT	エラー発生

例：

ret& = **HypHPDRV**(hRApp%, "INCOME", 200.0, 100.0, retVal#)

C 言語では次の形式を使用します。

**int** WINAPI HypHPDRV( HRETRIEVEAPP hRApp, LPCSTR szAcct, const double FAR \* pdValue, const double FAR \* pdValue, double FAR \* pdRetval)

## HypHPECO( ) - エンティティコードの取得

この関数を実行すると、指定したデータ種別と期間のエンティティに関連付けられたコードが返されます。

次の形式を使用します。

Declare Function HypHPECO Lib “heaccess.dll” Alias “\_HypHPECO@24” (ByVal hRApp As Integer, ByVal szEntity As String, ByVal szCat As String, ByVal szPerName As String, ByVal wLen As Integer, ByVal szCode As String) As Integer

#### 変数 説明

hRApp HypConstruct( )からの Hyperion Enterprise Retrieve アプリケーションハンドル

szEntity エンティティ ID

szCat データ種別 ID

szPerNam 期間の日付 (“02/28/03”)、ID (“Feb 03”) または番号 (“2”)

wLen szCode バッファの長さ

szCode コードを返すためのバッファ

戻りコード :

コード	説明
0	成功
HACC_BAD_NAME	エラー発生、無効な szEntity
HACC_BAD_CATEGORY	エラー発生、無効な szCat
HACC_BAD_SELECT	エラー発生

C 言語では次の形式を使用します。

short pascal HypHPECO( HRETRIEVEAPP hRApp, LPCSTR szEntity, LPCSTR szCat, LPCSTR szPerNam, WORD wLen, LPSTR szCode)

## HypHPFlush( ) - データバッファのフラッシュ

この関数を実行すると、内部 Hyperion Enterprise の読み取りアクセスバッファまたは書き込みアクセスバッファがクリアされます。HypHPFlush( )では、HypHPLNK( )によって送信されたが、まだコミットされていないデータがすべて破棄されます。この関数を使用すると、読み取りアクセスデータバッファをディスクから強制的に再度読み込むこともできます。どちらのバッファセットも自動的にフラッシュされるので、コミットの後でフラッシュを実行する必要はありません。

次の形式を使用します。

Declare Function HypHPFlush Lib “heaccess.dll” Alias “\_HypHPFlush@8” (ByVal hRApp As Integer, ByVal wMask As Integer) As Long

#### 値 説明

hRApp HypConstruct( )からのアプリケーションハンドル

## 値 説明

wMask フラッシュするバッファ。HACC\_FLUSH\_READ（読み取りバッファをフラッシュする場合）と HACC\_FLUSH\_WRITE（書き込みバッファをフラッシュする場合）の組み合わせになる場合もあります。

戻りコード：

コード	説明
0	成功
NONE	エラー発生

例：

```
wMask% = HACC_FLUSH_WRITE  
rc& = HypHPFlush(hRApp%, wMask%)
```

C 言語では次の形式を使用します。

```
int WINAPI HypHPFlush(HRETRIEVEAPP hRApp, WORD wMask)
```

## HypHPFLW( ) - フロー勘定科目の判別

この関数を実行すると、指定した勘定科目がフロー勘定科目であるかどうか判別されます。指定した勘定科目がフロー勘定科目でない場合は、それが残高勘定科目であることを意味します。

次の形式を使用します。

```
Declare Function HypHPFLW Lib "heaccess.dll" Alias "_HypHPFLW@12" (ByVal  
hRApp As Integer, ByVal szAcct As String, iFlag As Long) As Long
```

## 値 説明

hRApp HypConstruct( )からの Hyperion Enterprise Retrieve アプリケーションハンドル

szAcct 勘定科目 ID

iFlag 結果を返すためのバッファ

戻りコード：

コード	説明
0	成功
HACC_BAD_ACCOUNT	エラー発生、無効な szAcct
HACC_BAD_SELECT	エラー発生

戻り値では、iFlag の値は表 12 に示す値になります。

表 12 iFlag に指定される可能性がある値

値	説明
1	勘定科目がフロー勘定科目であるかどうかを指定します。
-1	勘定科目がフロー勘定科目でないかどうかを指定します。
0	エラーが発生したかどうかを指定します。

例：

```
ret& = HypHPFLW(hRApp%, "SALES", isFlow&)
```

C 言語では次の形式を使用します。

```
int WINAPI HypHPFLW( HRETRIEVEAPP hRApp, LPCSTR szAcct, int FAR * piFlag)
```

## HypHPFRE( ) - データ種別期間単位の取得

この関数を実行すると、指定したデータ種別のデフォルトのレポート期間単位が取得されます。レポート期間単位の一覧は、[20 ページの「期間単位と表示形式」](#)を参照してください。

次の形式を使用します。

```
Declare Function HypHPFRE Lib "heaccess.dll" Alias "_HypHPFRE@12" (ByVal hRApp As Integer, ByVal szCat As String, ByVal szRetbuf As String) As Long
```

### 変数 説明

hRApp HypConstruct( )からの Hyperion Enterprise Retrieve アプリケーションハンドル

szCat データ種別 ID

szRetbuf レポート期間単位を返すためのバッファ。バッファの長さは、少なくとも HYP\_SIZERPTFREQ+1 文字（C 言語の場合は SIZERPTFREQ+1 文字）にする必要があります。

戻りコード：

コード	説明
0	成功
HACC_BAD_CATEGORY	エラー発生、無効な szCat
HACC_BAD_SELECT	エラー発生

例：

```
szRptFreq$ = SPACE (HYP_SIZERPTFREQ+1)
ret& = HypHPFRE(hRApp%, "BUDGET", szRptFreq$)
If ret& = 0 Then szRptFreq$ = CToBStr(szRptFreq$)
```

C 言語では次の形式を使用します。

```
int WINAPI HypHPFRE( HRETRIEVEAPP hRApp, LPCSTR szCat, LPSTR szRetbuf)
```

# HypHPFUL( ) - コンポーネントのエンティティに関する完全な説明の取得

この関数を実行すると、指定したエンティティの説明が取得されます。次のコードを指定すると、エンティティの説明をさまざまな形式で取得できます。

- FUL\_FUL：entity.sub-entity の完全修飾形式の説明を返します。
- FUL\_FNA：主要エンティティの説明のみを返します。
- FUL\_FSN：サブエンティティの説明のみを返します。

次の形式を使用します。

Declare Function HypHPFUL Lib “heaccess.dll” Alias “\_HypHPFUL@16” (ByVal hRApp As Integer, ByVal szEntity As String, ByVal iCode As Long, ByVal szRetbuf As String) As Long

## 変数 説明

hRApp HypConstruct( )からの Hyperion Enterprise Retrieve アプリケーションハンドル

szEntity エンティティ ID

iCode 次のいずれかの取得コード：

FUL\_FUL

FUL\_FNA

FUL\_FSN

szRetbuf エンティティの完全な説明を返すためのバッファ。バッファの長さは、少なくとも HYP\_SIZEDESC 文字（C 言語の場合は SIZEDESC 文字）にする必要があります。

戻りコード：

コード	説明
0	成功
HACC_BAD_NAME	エラー発生、無効な szEntity
HACC_BAD_SELECT	エラー発生

例：

```
szDesc$ = SPACE (HYP_SIZEDESC+1)
ret& = HypHPFUL(hRApp%, "CANADA", FUL_FUL, szDesc$)
If ret& = 0 Then szDesc$ = CToBStr(szDesc$)
```

C 言語では次の形式を使用します。

int WINAPI HypHPFUL( HRETRIEVEAPP hRApp, LPCSTR szEntity, int iCode, LPSTR szRetbuf)

## HypHPHEA( ) - 勘定科目見出しの取得

この関数を実行すると、指定した勘定科目の説明が取得されます。表 13 のいずれかの取得コードを指定できます。

表 13 HypHPHEA( ) 勘定科目コード

コード	説明
HEA_HEA	第 1 レベルおよび第 2 レベルのサブ勘定科目も含め、完全な説明を取得します。
HEA_HMJ	主要勘定科目の見出しのみを取得します。
HEA_HS1	第 1 レベルサブ勘定科目の見出しのみを取得します。
HEA_HS2	第 2 レベルサブ勘定科目の見出しのみを取得します。

次の形式を使用します。

```
Declare Function HypHPHEA Lib "heaccess.dll" Alias "_HypHPHEA@16" (ByVal  
hRApp As Integer, ByVal szAcct As String, ByVal iCode As Long, ByVal szRetbuf As String)  
As Long
```

### 変数 説明

hRApp HypConstruct( )からの Hyperion Enterprise Retrieve アプリケーションハンドル

szAcct 勘定科目 ID

iCode 次のいずれかの取得コード：

HEA\_HEA

HEA\_HMJ

HEA\_HS1

HEA\_HS2

szRetbuf 勘定科目の見出しを返すためのバッファ。バッファの長さは、少なくとも HYP\_SIZEDESC  
+1 文字（C 言語の場合は SIZEDESC+1 文字）にする必要があります。

戻りコード：

コード	説明
0	成功
HACC_BAD_ACCOUNT	エラー発生、無効な szAcct
HACC_BAD_SELECT	エラー発生

例：

```
szDesc$ = SPACE (HYP_SIZEDESC+1)  
ret& = HypHPHEA(hRApp%, "SALES", HEA_HEA, szDesc$)  
If ret& = 0 Then szDesc$ = CToBStr(szDesc$)
```

C 言語では次の形式を使用します。

int WINAPI HypHPHEA( HRETRIEVEAPP hRApp, LPCSTR szAcct, int iCode, LPSTR szRetbuf)

## HypHPINC( ) - 収益勘定科目の判別

この関数を実行すると、指定した勘定科目が収益勘定科目であるかどうか判別されます。

次の形式を使用します。

Declare Function HypHPINC Lib "heaccess.dll" Alias "\_HypHPINC@12" (ByVal hRApp As Integer, ByVal szAcct As String, iFlag As Long) As Long

### 変数 説明

hRApp HypConstruct( )からの Hyperion Enterprise Retrieve アプリケーションハンドル

szAcct 勘定科目 ID

iFlag 結果を返すためのバッファ

戻りコード：

コード	説明
0	成功
HACC_BAD_ACCOUNT	エラー発生、無効な szAcct
HACC_BAD_SELECT	エラー発生

戻り値では、iFlag の値は表 14 に示す値になります。

表 14 iFlag に指定される可能性がある値

値	説明
1	勘定科目が収益勘定科目であるかどうかを指定します。
-1	勘定科目が収益勘定科目でないかどうかを指定します。
0	エラーが発生したかどうかを指定します。

例：

ret& = **HypHPINC**(hRApp%, "SALES", isIncome&)

C 言語では次の形式を使用します。

int WINAPI HypHPINC( HRETRIEVEAPP hRApp, LPCSTR szAcct, int FAR \* piFlag)

## HypHPINP( ) - 入力勘定科目の判別

この関数を実行すると、指定した勘定科目が入力勘定科目であるかどうか判別されます。

次の形式を使用します。

```
Declare Function HypHPINP Lib "heaccess.dll" Alias "_HypHPINP@20" (ByVal  
hRApp As Integer, ByVal szEntity As String, ByVal szCat As String, ByVal szAcct As  
String, wFlag As Integer) As Long
```

#### 変数 説明

hRApp HypConstruct( )からの Hyperion Enterprise Retrieve アプリケーションハンドル

szEntity エンティティ ID を指すポインタ

szCat データ種別 ID を指すポインタ

szAcct 勘定科目 ID を指すポインタ

wFlag 結果を返すためのバッファを指すポインタ

戻りコード：

コード	説明
0	成功
NONE	エラー発生
HACC_BAD_ACCOUNT	エラー発生、無効な szAcc
HACC_BAD_CATEGORY	エラー発生、無効な szCat
HACC_BAD_NAME	エラー発生、無効な szEntity
HACC_BAD_SELECT	エラー発生

戻り値では、wFlag の値は表 15 に示す値になります。

表 15 wFlag に指定される可能性がある値

値	説明
HPINP_INPUT_ACCT (1)	勘定科目が入力勘定科目であるかどうかを指定します。
HPINP_NOT_INPUT_ACCT (-1)	勘定科目が入力勘定科目でないかどうかを指定します。
0	エラーが発生したかどうかを指定します。

例：

```
'HypHPINP ( ) で使用するコード  
Global Const HPINP_INPUT_ACCT = 1  
Global Const HPINP_NOT_INPUT_ACCT = -1
```

C 言語では次の形式を使用します。

```
int WINAPI HypHPINP( HRETRIEVEAPP hRApp, LPCSTR szEntity, LPCSTR szCat,  
LPCSTR szAcct, short FAR * pwFlag)
```

## HypHPJOUR( ) - 仕訳帳詳細情報の取得

この関数を実行すると、仕訳帳の詳細レコードに関する情報が取得されます。  
次の形式を使用します。

Declare Function HypHPJOUR Lib "heaccess.dll" Alias "\_HypHPJOUR@36" (ByVal hRApp As Integer, ByVal wAttr As Integer, ByVal szJourName As String, ByVal sigJourDet As Long, ByVal sTableID As Integer, ByVal szCat As String, ByVal szPerName As String, ByVal wLen As Integer, pzBuf As Any) As Integer

変数	説明
hRApp	ゼロ以外の Hyperion Enterprise Retrieve アプリケーションハンドル
wAttr	どの仕訳帳属性を返すかを指定する short 整数
szJourName	仕訳帳 ID
sigJourDet	HypJourDetAsk( )から取得される仕訳帳詳細行の記号
sTableID	ID_JOURNAL_ENTRIES または ID_JOURNAL_TEMPLATES から取得される表データの ID
szCat	データ種別 ID
szPerName	期間文字列 (mm/dd/yy)、期間番号 ("2"など) または期間 ID ("Feb 0"など)
wLen	pzBuf バッファの長さ
pzBuf	要求した値を返すためのバッファ

HypHPJOUR( )を呼び出す前に、バッファ (pzBuf 引数) を正しいサイズに割り当てる必要があります。表 16 では、有効な仕訳帳属性と各属性の正しいタイプとサイズを示します。

表 16 仕訳帳属性

属性	説明	タイプ/文字列サイズ
JOUR_NO	仕訳帳番号。仕訳帳番号がない場合は NONE (-1)。	Long
JOUR_ST	仕訳帳のステータス	整数 (C 言語では short)
JOUR_DS	仕訳帳の説明	文字列 SIZEJOURDESC+1
JOUR_EN	仕訳帳詳細エンティティ	文字列 SIZEFULLNAME+1
JOUR_AC	仕訳帳詳細勘定科目	文字列 SIZEFULLACT+1
JOUR_DB	仕訳帳詳細借方	Double
JOUR_CR	仕訳帳詳細貸方	Double

戻りコード：

コード	説明
0	成功
NONE	エラー発生
HACC_BAD_CATEGORY	エラー発生、無効な szCat
HACC_BAD_CODE	エラー発生、無効な wAttr
HACC_BAD_JOURNAL	エラー発生、無効な szJourName
HACC_BAD_PERIOD	エラー発生、無効な szPerName
HACC_BAD_SELECT	エラー発生
HACC_BAD_TABLE	エラー発生、無効な sTableID

C 言語では次の形式を使用します。

```
short pascal HypHPJOUR( HRETRIEVEAPP hRApp, short wAttr, LPSTR szJourName,
SIGNA sigJourDet, short sTableId, LPCSTR szCat, LPCSTR szPerName, short wLen,
void * pzBuf)
```

## HypHPKEYEx( ) - デフォルトのエンティティ、データ種別、勘定科目、または期間の取得

この関数を実行すると、指定したアプリケーションからデフォルトのエンティティ、データ種別、勘定科目または期間の ID が返されます。

次の形式を使用します。

```
Declare Function HypHPKEYEx Lib "heaccess.dll" Alias "_HypHPKEYEx@12"
(ByVal hRApp As Integer, ByVal szKey As String, ByVal szRetbuf As String) As Integer
```

### 変数 説明

hRApp HypConstruct( )からの Hyperion Enterprise Retrieve アプリケーションハンドル

szKey 取得する値を示す次のいずれかのディメンション：

“ENTITY”

“NAME”

“CATEGORY”

“ACCOUNT”

“PERIOD”

“ORGANIZATION”

szRetbuf デフォルトのディメンション値を返すためのバッファを指すポインタ

表 17 に示すように、バッファのサイズは取得するディメンションによって異なります。

表 17 キーとバッファサイズ

キー	バッファサイズ
"ENTITY"または"NAME"	HYP_SIZEFULLNAME+1 または C 言語では SIZEFULLNAME+1
"CATEGORY"	HYP_SIZELABEL+1 または C 言語では SIZECAT+1
"ACCOUNT"	HYP_SIZEFULLACCT+1 または C 言語では SIZEFULLACCT+1
"PERIOD"	HYP_SIZEDESC または C 言語では SIZEDESC
"ORGANIZATION"	long または C 言語では SIZEOF(SIGNA)

戻りコード：

コード	説明
0	成功
NONE	エラー発生

例：

```
Public Sub GetHyperionDefaults(Entity As String, Category As String,
Account As String, Period As String)
Dim rc%
Dim RetBuffer As String * 127
rc% = HypHPKEYEx(hRApp%, "ENTITY", RetBuffer$)
Entity$ = CToBStr(RetBuffer$)
rc% = HypHPKEYEx(hRApp%, "CATEGORY", RetBuffer$)
Category$ = CToBStr(RetBuffer$)
rc% = HypHPKEYEx(hRApp%, "ACCOUNT", RetBuffer$)
Account$ = CToBStr(RetBuffer$)
rc% = HypHPKEYEx(hRApp%, "PERIOD", RetBuffer$)
Period$ = CToBStr(RetBuffer$)
End Sub
```

C 言語では次の形式を使用します。

```
short WINAPI HypHPKEYEx( HRETRIEVEAPP hRApp, LPCSTR szKey, LPSTR
szRetbuf)
```

## HypHPLNK( ) - データ値の保存

この関数を実行すると、指定した入力基準に関するデータ値が保存されます。データ値は、HypHPCommit2( )が呼び出されるまで、実際のデータファイルではなく内部データバッファに保存されます。そのため、同じ値に関する HypHPVAL( )を後で呼び出すと、バッファの値ではなくデータファイル内の値が返されます。

次の形式を使用します。

```
Declare Function HypHPLNK Lib "heaccess.dll" Alias "_HypHPLNK@36" (ByVal
hRApp As Integer, ByVal szEntity As String, ByVal szCat As String, ByVal szAcct As String,
ByVal szPeriod As String, ByVal szFreq As String, ByVal fScale As Long, ByVal dp As
Double) As Long
```

## 変数 説明

hRApp HypConstruct( )からの Hyperion Enterprise Retrieve アプリケーションハンドル

szEntity エンティティ ID

szCat データ種別 ID

szAcct 勘定科目 ID

szPeriod 期間の日付 ("02/08/03")、ラベル ("Feb 03") または期間番号 ("2")

szFreq レポート期間単位または文字列 freq.view。ここで、freq は Hyperion Enterprise の期間単位 ("M"、"Q" など)、view はレポート表示形式です。期間単位については、[20 ページの「期間単位と表示形式」](#)を参照してください。

fScale 不使用（ゼロを渡します）

dp 送信される浮動小数点数

戻りコード：

コード	説明
0	成功
NONE	エラー発生
HACC_BAD_NAME	エラー発生、無効な szEntity
HACC_BAD_CATEGORY	エラー発生、無効な szCat
HACC_BAD_FREQUENCY	エラー発生、無効な szFreq
HACC_BAD_ACCOUNT	エラー発生、無効な szAcct
HACC_BAD_SELECT	エラー発生
DFERR_BADPERIOD	エラー発生

例：

```
rc& = HypHPLNK(hRApp%, Entity$, Cat$, Acct$, Per$, "M.PER" , 0,
    TheValue#)
If rc& <> 0 Then
SetHypValue = "Error in HypHPLNK - " & Str$(rc&)
Exit Function
End If
```

C 言語では次の形式を使用します。

```
int WINAPI HypHPLNK( HRETRIEVEAPP hRApp, LPCSTR szEntity, LPCSTR szCat,
LPCSTR szAcct, LPCSTR szPeriod, LPCSTR szFreq, BOOL fScale, double dp)
```

## HypHPNAM( ) - エンティティ ID の取得

この関数を実行すると、[表 18](#) に示す取得コードに基づいて、指定したエンティティの ID が取得されます。

表 18 HypHPNAM( )エンティティ取得コード

コード	取得されるエンティティ
NAM_NAM	主要エンティティとサブエンティティ (Entity.Subentity)
NAM_NNA	主要エンティティのみ
NAM_NSN	サブエンティティのみ

次の形式を使用します。

Declare Function HypHPNAM Lib “heaccess.dll” Alias “\_HypHPNAM@16” (ByVal hRApp As Integer, ByVal szEntity As String, ByVal iCode As Long, ByVal szRetbuf As String) As Long

**変数 説明**

hRApp HypConstruct( )からの Hyperion Enterprise Retrieve アプリケーションハンドル

szEntity エンティティ ID を指すポインタ

iCode 次のいずれかの取得コード : NAM\_NAM NAM\_NNA NAM\_NSN

szRetbuf エンティティ ID を返すためのバッファ。バッファの長さは、少なくとも HYP\_SIZEDESC 文字 (C 言語の場合は SIZEDESC 文字) にする必要があります。

戻りコード :

コード	説明
0	成功
HACC_BAD_NAME	エラー発生、無効な szEntity
HACC_BAD_SELECT	エラー発生

例 :

```
szMajor$ = SPACE (HYP_SIZEDESC)
ret& = HypHPNAM(hRApp%, szEntity$, NAM_NNA, szMajor$)
If ret& = 0 Then szMajor$ = CToBStr(szMajor$)
```

C 言語では次の形式を使用します。

int WINAPI HypHPNAM( HRETRIEVEAPP hRApp, LPCSTR szEntity, int iCode, LPSTR szRetbuf)

## HypHPOWN( ) - 所有株数の取得

この関数を実行すると、発行済み株式の総数、会社が別の会社を直接的または間接的に所有している比率など、会社の株式や出資に関する情報が取得されます。

次の形式を使用します。

Declare Function HypHPOWN Lib "heaccess.dll" Alias "\_HypHPOWN@28" (ByVal hRApp As Integer, ByVal szParent As String, ByVal szChild As String, ByVal szCat As String, ByVal szPeriod As String, ByVal wCode As Integer, dretval As Double) As Integer

## 変数 説明

hRApp	HypConstruct( )によって返されるゼロ以外の Hyperion Enterprise Retrieve アプリケーション ハンドル
szParent	親エンティティ
szChild	子エンティティ
szCat	データ種別 ID
szPeriod	期間の日付 ("02/08/03")、期間 ID ("Feb 03") または期間番号 ("2")
wCode	取得する情報のコード。詳しくは、「wCode 最終出資比率値」の表を参照してください。
dretval	情報を返すためのバッファを指すポインタ

指定した子エンティティに関する親エンティティの最終出資比率を取得するには、[表 19](#) に示す wCode 値を使用します。これには、出資比率を問わず、親の出資先会社によって直接的および間接的に所有される株式が含まれます。

**表 19 wCode 最終出資比率値**

値	取得される情報
SHARES_ULTPCTOWNED	最終出資比率
SHARES_ULTPCTCTL	最終支配比率
SHARES_ULTPCTCONSOL	最終連結比率

指定した子エンティティに関する親エンティティの直接出資比率を取得するには、[表 20](#) に示す wCode 値を使用します。戻り値は、Hyperion Enterprise の株の入力方法として単位または比率のどちらのオプションが設定されているかに応じて、株式数または出資比率になります。

**表 20 wCode 直接出資比率値**

値	取得される情報
SHARES_NONVOTING	直接出資比率（所有株数）
SHARES_VOTING	直接支配比率（所有議決権株数）

比率算出のためのさまざまな合計値を取得するには、[表 21](#) に示す wCode 値を使用します。

表 21 wCode 比率計算値

値	取得される情報
SHARES_TOT_ISSUED_NONVOT	発行済株式の総数
SHARES_TOT_ISSUED_VOT	発行済議決権株の総数
SHARES_TOT_OWNED_NONVOT	他エンティティによって所有される株式の総数
SHARES_TOT_OWNED_VOT	他エンティティによって所有される議決権株の総数

戻りコード：

コード	説明
0	成功
NONE	エラー発生
HACC_BAD_CATEGORY	エラー発生、無効な szCat
HACC_BAD_CODE	エラー発生、無効な wCode
HACC_BAD_SELECT	エラー発生

C 言語では次の形式を使用します。

**short pascal HypHPOWN( HRETRIEVEAPP hRApp, LPCSTR szParent, LPCSTR szChild, LPCSTR szCat, LPCSTR szPeriod, WORD wCode, double \* pdretval)**

## HypHPPBE( ) - 優劣比率の算出

ある勘定科目に 2 つの値がある場合にこの関数を実行すると、遅い方の期間値が早い方の期間値より優れているか劣っているかを示す比率が結果として返されます。結果を表す記号は、指定した勘定科目が費用、資産、収益または負債のいずれであるかによって決まります。

次の形式を使用します。

**Declare Function HypHPPBE Lib “heaccess.dll” Alias “\_HypHPPBE@20” (ByVal hRApp As Integer, ByVal szAcct As String, value2 As Double, value1 As Double, dretval As Double) As Long**

### 変数 説明

hRApp HypConstruct( )からの Hyperion Enterprise Retrieve アプリケーションハンドル

szAcct 勘定科目

value2 遅い方の期間値

value1 早い方の期間値

dretval 浮動小数点数を返すためのバッファ

戻りコード：

コード	説明
0	成功
HACC_BAD_ACCOUNT	エラー発生、無効な szAcct
HACC_BAD_SELECT	エラー発生

例：

```
ret& = HypHPPBE(hRApp%, "INCOME", 200.0, 100.0, pctDif#)
```

C 言語では次の形式を使用します。

```
int WINAPI HypHPPBE( HRETRIEVEAPP hRApp, LPCSTR szAcct, const double FAR *  
pdValue2, const double FAR * pdValue1, double FAR * pdRetVal)
```

## HypHPPCH( ) - 差異率の算出

この関数を実行すると、2 つの値の差異率が返されます。

次の形式を使用します。

```
Declare Function HypHPPCH Lib "heaccess.dll" Alias "_HypHPPCH@16" (ByVal  
hRApp As Integer, value1 As Double, value2 As Double, dretval As Double) As Long
```

### 変数 説明

hRApp HypConstruct( )からの Hyperion Enterprise Retrieve アプリケーションハンドル

value1 一方の値

value2 もう一方の値

dretval 浮動小数点数を返すためのバッファ

戻りコード：

コード	説明
0	成功

例：

```
ret& = HypHPPCH(hRApp%, 200.0, 100.0, pctDif#)
```

C 言語では次の形式を使用します。

```
int WINAPI HypHPPCH( HRETRIEVEAPP hRApp, const double FAR * pdValue1, const  
double FAR * pdValue2 double FAR * pdRetVal)
```

## HypHPSCA( ) - 単位の取得

この関数を実行すると、エンティティ、データ種別および勘定科目の単位比率が取得されます。

次の形式を使用します。

Declare Function HypHPSCA Lib "heaccess.dll" Alias "\_HypHPSCA@20" (ByVal hRApp As Integer, ByVal szEntity As String, ByVal szCat As String, ByVal szAcct As String, dretval As Double) As Long

### 変数 説明

hRApp HypConstruct( )からの Hyperion Enterprise Retrieve アプリケーションハンドル

szEntity エンティティ ID

szCat データ種別 ID

szAcct 勘定科目 ID

dretval 単位比率を返すためのバッファ

勘定科目の単位設定が不可能な場合、この関数の引数 dretval は 1 になります。データ種別に単位比率が設定されている場合、dretval はデータ種別の単位比率になります。それ以外の場合、dretval はエンティティの単位比率になります。たとえば、データ種別の単位が 3 の場合、dretval で参照される単位比率は 1000 になります。単位が設定されていない場合またはエラーが発生した場合、dRetVal で返される単位比率は 1 になります。

戻りコード：

コード	説明
0	成功
NONE	エラー発生
HACC_BAD_ACCOUNT	エラー発生、無効な szAcct
HACC_BAD_CATEGORY	エラー発生、無効な szCat
HACC_BAD_NAME	エラー発生、無効な szEntity
HACC_BAD_SELECT	エラー発生

例：

```
ret& = HypHPSCA(hRApp%, "EuroDiv", "ACTUAL", "SALES", scale#)
```

C 言語では次の形式を使用します。

```
int WINAPI HypHPSCA( HRETRIEVEAPP hRApp, LPCSTR szEntity, LPCSTR szCat, LPCSTR szAcct, double FAR * pdRetVal)
```

## HypHPVAL( ) - 通常の勘定科目値の取得

この関数を実行すると、指定したエンティティ、データ種別、勘定科目、期間および期間単位に基づいて、Hyperion Enterprise データベースからデータ値が取得されます。DateConv( )関数を使用すると、HypHPVAL( )で使用する正しい期間番号を算出できます。この関数によってエラーが返される場合は、HypValReturn( )を呼び出して、エラーの詳細情報を取得することができます。HypHPVAL( )では、データがない期間に対して、システム生成の派生値が単位として設定されます。

**注：** データ値を取得する場合は HypHPVAL( )を使用し、調整後の値、比率値、調整値、通貨換算値などの特殊な値を取得する場合は HypHPVALEx( )を使用します。HypHPVAL2 は、いずれの場合にも使用できます。

次の形式を使用します。

**Declare Function HypHPVAL Lib "heaccess.dll" Alias "\_HypHPVAL@32" (ByVal hRApp As Integer, ByVal szEntity As String, ByVal szCat As String, ByVal szAcct As String, ByVal szPeriod As String, ByVal szFreq As String, ByVal fScale As Long, dretval As Double) As Long**

### 変数 説明

hRApp HypConstruct( )からの Hyperion Enterprise Retrieve アプリケーションハンドル

szEntity エンティティ ID

szCat データ種別 ID

szAcct 勘定科目 ID

szPeriod 期間の日付 ("02/08/03")、期間ラベル ("Feb 03") または期間番号 ("2")

szFreq レポート期間単位コード (DAY、DYTD、WEE、WYTD、MON、YTD、QUA、QYTD、HAL、HYTD、YEA、DAI、TRI または TYTD) のいずれかを指すポインタ。期間単位については、[20 ページの「期間単位と表示形式」](#)を参照してください。

fScale データの単位を設定する場合は TRUE (ゼロ以外)、それ以外の場合は FALSE (ゼロ)

dretval 浮動小数点数を返すためのバッファ

戻りコード：

コード	説明
0	成功
HACC_BAD_ACCOUNT	エラー発生、無効な szAcct
HACC_BAD_CATEGORY	エラー発生、無効な szCat
HACC_BAD_NAME	エラー発生、無効な szEntity
HACC_BAD_SELECT	エラー発生

コード	説明
HACC_BAD_FREQUENCY	エラー発生、無効な szFreq
RET_NA	データが使用不可（データなし）
HACC_DATAFILE_ERR	データを読み取る際のエラー

例：

```
ret& = HypHPVAL(hRApp%, "CANADA", "ACTUAL", "SALES", "2", "MON", 0,
    TheValue#)
```

C 言語では次の形式を使用します。

```
int WINAPI HypHPVAL( HRETRIEVEAPP hRApp, LPCSTR szEntity, LPCSTR szCat,
    LPCSTR szAcct, LPCSTR szPeriod, LPCSTR szFreq, BOOL fScale, double FAR * pdRetVal)
```

## HypHPVAL2( ) - 通常の勘定科目値の取得

この関数は、Hyperion Enterprise の効率性を向上させるために特別にカスタマイズされた HypHPVAL( ) および HypVALEX のクローンです。HypHPVAL2( ) に含まれているオプションを使用すると、HypValReturn( ) を内部で呼び出して、パラメータが有効な場合に限り、値なし設定（MV モード）に基づいて戻りコードを調整することができます。

**注：** 通常のデータ値を取得する場合は HypHPVAL( ) を使用し、調整後の値、比率値、調整値、通貨換算値などの特殊な値を取得する場合は HypHPVALEX( ) を使用します。HypHPVAL2 は、いずれの場合にも使用できます。

次の形式を使用します。

```
Declare Function HypHPVAL2 Lib "heaccess.dll" Alias "_HypHPVAL@44" (ByVal
hRApp As Integer, ByVal szEntity As String, ByVal szCat As String, ByVal szAcct As String,
ByVal szPeriod As String, ByVal szFreq As String, ByVal szParent As String, ByVal wType
As Integer, ByVal fScale As Long, ByVal fUseMVMode As Long, dVal As Double) As
Integer
```

変数	説明
hRApp	HypConstruct( ) からの Hyperion Enterprise Retrieve アプリケーションハンドル
szEntity	エンティティラベル
szCat	データ種別ラベル
szAcct	勘定科目ラベル
szPeriod	期間の日付 ("02/08/03")、期間ラベル ("Feb 03") または期間番号 ("2")
szFreq	レポート期間単位コード (DAY、DYTD、WEE、WYTD、MON、YTD、QUA、QYTD、HAL、HYTD、YEA、DAI、TRI または TYTD) のいずれか。期間単位については、 <a href="#">20 ページの「期間単位と表示形式」</a> を参照してください。

変数	説明
szParent	wType = ID_REGULAR の場合は NULL、それ以外の場合は親エンティティのラベル
wType	取得するデータタイプのコード。次のいずれかになります。 <ul style="list-style-type: none"> <li>● ID_REGULAR</li> <li>● ID_PROPORTIONAL</li> <li>● ID_ELIMINATION</li> <li>● ID_CONTRIBUTION</li> <li>● ID_ADJUSTMENT</li> <li>● ID_TRANSLATION</li> <li>● ID_TRANSLATION_FORCE</li> </ul>
fScale	データの単位を設定する場合は TRUE（ゼロ以外）、それ以外の場合は FALSE（ゼロ）
fUseMVMode	MV モードに基づいて戻りコードを調整する場合は TRUE、それ以外の場合は FALSE
dval	データ値を返す先の変数

戻りコード：

コード	説明
0	成功
HACC_BAD_ACCOUNT	エラー発生、無効な szAcct
HACC_BAD_CATEGORY	エラー発生、無効な szCat
HACC_BAD_NAME	エラー発生、無効な szEntity
HACC_BAD_SELECT	エラー発生
HACC_BAD_FREQUENCY	エラー発生、無効な szFreq
RET_NA	データが使用不可（データなし）
HACC_DATAFILE_ERR	データを読み取る際のエラー

例 1：

```
rc% = HypHPVAL2(hRApp%, "CANADA", "ACTUAL", "SALES", "2", "MON", "",
ID_REGULAR, 0, 1, TheValue#)
```

例 2：

```
rc% = HypHPVAL2(hRApp%, "Spain", "ACTUAL", "SALES", "2", "MON",
"EUROPE", ID_CONTRIBUTION, 0, 1, TheValue#)
```

C 言語では次の形式を使用します。

```
int WINAPI HypHPVAL2( HRETRIEVEAPP hRApp, LPCSTR szEntity, LPCSTR szCat,
LPCSTR szAcct, LPCSTR szPeriod, LPCSTR szFreq, LPCSTR szParent, WORD wType,
BOOL fScale, BOOL fUseMVMode, double FAR * pdVal)
```

## HypHPVALEx( ) - 特殊な勘定科目値の取得

この関数を実行すると、指定したエンティティ、データ種別、勘定科目、期間および期間単位に基づいて、Hyperion Enterprise データベースからデータ値が取得されます。DateConv()関数を使用すると、HypHPVALEx( )と HypHPVAL( )で使用する正しい期間番号を算出できます。

**注：** 通常 of データ値を取得する場合は HypHPVAL( )を使用し、調整後の値、比率値、調整値、通貨換算値などの特殊な値を取得する場合は HypHPVALEx( )を使用します。HypHPVAL2 は、いずれの場合にも使用できます。

次の形式を使用します。

```
Declare Function HypHPVALEx Lib "heaccess.dll" Alias "_HypHPVALEx@40"  
(ByVal hRApp As Integer, ByVal szEntity As String, ByVal szCat As String, ByVal szAcct  
As String, ByVal szPeriod As String, ByVal szFreq As String, ByVal szParent As String,  
ByVal wType As Integer, ByVal fScale As Integer, dretval As Double) As Integer
```

### 変数 説明

hRApp	HypConstruct( )からの Hyperion Enterprise Retrieve アプリケーションハンドル
szEntity	エンティティ ID
szCat	データ種別 ID
szAcct	勘定科目 ID
szPeriod	期間の日付 ("02/08/03")、期間 ID ("Feb 03") または期間番号 ("2") をへのポインタ
szFreq	レポート期間単位コード (DAY、DYTD、WEE、WYTD、MON、YTD、QUA、QYTD、HAL、HYTD、YEA、DAI、TRI または TYTD) のいずれか。期間単位については、 <a href="#">20 ページの「期間単位と表示形式」</a> を参照してください。
szParent	wType が ID_REGULAR でない場合は親エンティティ。
wType	要求されるデータのタイプ。次のコードのいずれかを使用します。 ID_REGULAR ID_PROPORTIONAL ID_ELIMINATION ID_CONTRIBUTION ID_ADJUSTMENT ID_TRANSLATION ID_TRANSLATION_FORCE
fScale	データの単位を設定する場合は TRUE (ゼロ以外)、それ以外の場合は FALSE (ゼロ)
dretval	浮動小数点数を返すためのバッファ

戻りコード：

コード	説明
0	成功
HACC_BAD_ACCOUNT	エラー発生、無効な szAcct
HACC_BAD_CATEGORY	エラー発生、無効な szCat
HACC_BAD_NAME	エラー発生、無効な szEntity
HACC_BAD_SELECT	エラー発生
HACC_BAD_FREQUENCY	エラー発生、無効な szFreq
RET_NA	データが使用不可（データなし）
HACC_DATAFILE_ERR	データを読み取る際のエラー

例 1 :

```
rc% = HypHPVAlEx(hRApp%, "CANADA", "ACTUAL", "SALES", "2", "MON", "",
    ID_REGULAR, 0, TheValue#)
```

例 2 :

```
rc% = HypHPVAlEx(hRApp%, "Spain", "ACTUAL", "SALES", "2", "MON",
    "EUROPE", ID_CONTRIBUTION, 0, TheValue#)
```

C 言語では次の形式を使用します。

```
short WINAPI HypHPVAlEx( HRETRIEVEAPP hRApp, LPCSTR szEntity, LPCSTR
szCat, LPCSTR szAcct, LPCSTR szPeriod, LPCSTR szFreq, LPCSTR szParent, WORD
wType, SBOOL fScale, double FAR * pdRetVal)
```

## HypJourAsk( ) - 仕訳帳の選択

この関数を実行すると、仕訳帳を選択するためのダイアログボックスが表示されます。ここで選択した仕訳帳は、[仕訳帳の詳細] ダイアログボックスで使用されるデフォルトの仕訳帳になります。これにより、[仕訳帳の詳細] ダイアログボックスのデフォルトのデータ種別と期間も設定されます。

次の形式を使用します。

```
Declare Function HypJourAsk Lib "heaccess.dll" Alias _HypJourAsk@8 (ByVal
hRApp As Integer, ByVal szRetbuf As String) As Integer
```

### 変数 説明

hRApp HypConstruct( )からの Hyperion Enterprise Retrieve アプリケーションハンドル

szRetbuf 仕訳帳 ID を返すためのバッファ

戻りコード :

コード	説明
0	成功
NONE	エラー発生

C 言語では次の形式を使用します。

```
short WINAPI HypJourAsk( HRETRIEVEAPP hRApp, LPSTR szRetbuf)
```

## HypJourDetAsk( ) - 仕訳帳詳細の選択

この関数を実行すると、[仕訳帳の詳細] ダイアログボックスが表示されます。HypJourAsk( )を使用して仕訳帳を選択してから、HypJourDetAsk( )を呼び出して、その仕訳帳から詳細レコードを選択します。HypJourDetAsk( )では、HypJourAsk( )によって設定されたデフォルトのデータ種別と期間が使用されます。詳しくは、[62 ページの「HypJourAsk\( \) - 仕訳帳の選択」](#)を参照してください。

次の形式を使用します。

```
Declare Function HypJourDetAsk Lib "heaccess.dll" Alias "_HypJourDetAsk@8"  
(ByVal hRApp As Integer, sig As Long) As Integer
```

### 変数 説明

hRApp HypConstruct( )からの Hyperion Enterprise Retrieve アプリケーションハンドル

sig 仕訳帳詳細レコードの記号を返すためのバッファ

戻りコード：

コード	説明
0	成功
NONE	エラー発生

C 言語では次の形式を使用します。

```
short WINAPI HypJourDetAsk (HRETRIEVEAPP hRApp, SIGNA * psig)
```

## HypMultiAsk( ) - デフォルトのアプリケーションの選択

この関数を実行すると、[アプリケーションの選択] ダイアログボックスが呼び出され、HypConstruct( )またはHypMultiInit( )によって開かれたアプリケーションが一覧表示されます。ここで選択したアプリケーションはデフォルトのアプリケーションになり、後でHypMultiDefault( )を呼び出したときに結果として返されます。詳しくは、[29 ページの「HypConstruct\( \) - アプリケーションを開く」](#)、[67 ページの「HypMultiInit\( \) - 複数アプリケーションへのログオン」](#)、または[64 ページの「HypMultiDefault\( \) - デフォルトのハンドルの取得」](#)を参照してください。

次の形式を使用します。

**Declare Function HypMultiAsk Lib “heaccess.dll” Alias “\_HypMultiAsk@0” ( ) As Long**

戻りコード：

コード	説明
0	成功

例：

```
ret& = HypMultiAsk()hRApp% = HypMultiDefault() '選択されたアプリケーションはhRApp%
```

C 言語では次の形式を使用します。

**int WINAPI HypMultiAsk( void )**

## HypMultiDefault( ) - デフォルトのハンドルの取得

この関数を実行すると、デフォルトのアプリケーションの Hyperion Enterprise Retrieve アプリケーションハンドルが返されます。アプリケーションが初期化されていない場合は、NULL が返されます。HypMultiDefault( )では、デフォルトのアプリケーションがない場合、最初に開かれたアプリケーションのハンドルが返されます。

次の形式を使用します。

**Declare Function HypMultiDefault Lib “heaccess.dll” Alias “\_HypMultiDefault@0” ( ) As Integer**

戻りコード：

コード	説明
Hyperion Enterprise Retrieve アプリケーションハンドル	成功
0	開いているアプリケーションがない
NONE	エラー発生

例：

```
hRApp% = HypMultiDefault()  
If hRApp% = 0 Then ret% = MsgBox("No applications are open",  
vbOKOnly, "Error")
```

C 言語では次の形式を使用します。

**HRETRIEVEAPP WINAPI HypMultiDefault( void )**

## HypMultiDeinit( ) - 複数アプリケーションからのログオフ

Hyperion Enterprise からログオフするには、この関数を使用します。この関数を実行すると、HypMultiInit( )または HypConstruct( )を使用して開いたすべての Hyperion Enterprise アプリケーションが閉じられ、初期化を解除されます。詳しくは、[67 ページの「HypMultiInit\( \) - 複数アプリケーションへのログオン」](#)または [29 ページの「HypConstruct\( \) - アプリケーションを開く」](#)を参照してください。

HypMultiInit( )の呼び出しが完了するたびに、各呼び出しを HypMultiDeinit( )の呼び出しと関連付ける必要があります。HypConstruct( )の呼び出しでは同じデータが共有されるので、HypMultiDeinit( )に対する最後の呼び出しが行われると、HypConstruct( )によって開かれたアプリケーションも閉じられます。

次の形式を使用します。

**Declare Function HypMultiDeinit Lib “heaccess.dll” Alias “\_HypMultiDeinit@0” ( ) As Long**

戻りコード：

コード	説明
0	成功

例：

```
hRApp% = HypMultiDeinit()If hRApp% <> 0 Then retz% = MsgBox("Log off failed", vbOKOnly, "Error")
```

C 言語では次の形式を使用します。

**int WINAPI HypMultiDeinit( void )**

## HypMultiEnum( ) - 開いているアプリケーションの列挙

この関数を実行すると、指定した関数のアドレスがシステム内で開かれているアプリケーションごとに一度ずつ呼び出され、各アプリケーションの Hyperion Enterprise Retrieve アプリケーションハンドルと ID が返されます。これにより、デフォルトのアプリケーションに設定しなくてもアプリケーションを選択できるようになります。

次の形式を使用します。

**Declare Sub HypMultiEnum Lib “heaccess.dll” Alias “\_HypMultiEnum@4” (ByVal lpCallback As Long)**

ここで、lpCallback は列挙関数のアドレスです。詳しくは、[330 ページの「CALLBACKSTR」](#)を参照してください。

16 ビットの Visual Basic では次の形式を使用します。

**Public Function MyFunc(ByVal hRApp As Integer, ByVal szApp As String) As Integer**

## 変数 説明

hRApp Hyperion Enterprise Retrieve アプリケーションハンドル

szApp アプリケーション ID

戻りコード：

この関数の戻りコードはありません。

C 言語では次の形式を使用します。

```
void WINAPI HypMultiEnum( CALLBACKSTR lpCallback)
```

## HypMultiGet( ) - アプリケーションハンドルの取得

この関数を実行すると、特定のアプリケーション名に対して使用する適切な Hyperion Enterprise Retrieve アプリケーションハンドルが取得されます。指定したアプリケーション名が見つからない場合は、NULL (0)が返されます。

**注：** HypMultiGet( )を使用すると、HypMultiInit( )またはHypConstruct( )を使用してアプリケーションを初期化した後でのみ、Hyperion Enterprise Retrieve アプリケーションハンドルを取得することができます。詳しくは、[67 ページの「HypMultiInit\( \) - 複数アプリケーションへのログオン」](#)または[29 ページの「HypConstruct\( \) - アプリケーションを開く」](#)を参照してください。

次の形式を使用します。

```
Declare Function HypMultiGet Lib "heaccess.dll" Alias "_HypMultiGet@4" (ByVal szApp As String) As Integer
```

ここで、szApp は、Hyperion Enterprise Retrieve アプリケーションハンドルが要求されているアプリケーションの名前です。ヌル文字の場合は、HypMultiDefault( ) からデフォルトの Hyperion Enterprise Retrieve アプリケーションハンドルが返されます。

戻りコード：

コード	説明
Hyperion Enterprise Retrieve アプリケーションハンドル	成功
NULL	エラー発生

例：

```
hRApp% = HypMultiGet("TAX")
If hRApp% = 0 Then ret% = MsgBox("TAX application is not active",
    vbOKOnly, "Error")
```

C 言語では次の形式を使用します。

```
HRETRIEVEAPP WINAPI HypMultiGet( LPCSTR szApp)
```

## HypMultiInit( ) - 複数アプリケーションへのログオン

この関数を実行すると、アプリケーションごとに HypConstruct( ) を内部で呼び出すことにより、アプリケーション一覧ファイルの内容に基づいて複数のアプリケーションが開きます。一度に最大 20 (MAX\_OPEN\_APPS) 個のアプリケーションを開くことができます。HypMultiInit( ) は一度だけ呼び出す必要があります。アプリケーションの使用を終えたら、プログラムを終了する前に HypMultiDeinit( ) を呼び出します。詳しくは、[29 ページの「HypConstruct\( \) - アプリケーションを開く」](#) または [65 ページの「HypMultiDeinit\( \) - 複数アプリケーションからのログオフ」](#) を参照してください。

アプリケーション一覧ファイルは、次の例に示すように、Hyperion Enterprise Retrieve によって使用される HPAPP.DAT ファイルと同じ形式にする必要があります。

```
app [, user [, password]]
```

次の形式を使用します。

**Declare Function HypMultiInit Lib “heaccess.dll” Alias “\_HypMultiInit@8” (ByVal szFName As String, ByVal ShowErr As Long) As Long**

### 変数 説明

szFName アプリケーション一覧ファイルの名前 (HPAPP.DAT など)

ShowErr ゼロまたはゼロ以外。TRUE (ゼロ以外) の場合はすべてのエラーが表示されます。

戻りコード：

コード	説明
0	成功
NONE	エラー発生

例：

```
'HPAPP.DATのアプリケーションを初期化
ret& = HypMultiInit("HPAPP.DAT", -1)
If ret& <> 0 Then ret2% = MsgBox("Logon failed", vbOKOnly, "Error")
```

C 言語では次の形式を使用します。

**int WINAPI HypMultiInit(LPCSTR szFName, int iShowErr)**

## HypNamAskEx( ) - エンティティの選択

この関数を実行すると、Hyperion Enterprise の [エンティティの選択] ダイアログボックスが呼び出され、ユーザが選択したエンティティが取得されます。ユーザが [キャンセル] を選択した場合、エンティティは取得されません。

次の形式を使用します。

**Declare Function HypNamAskEx Lib “heaccess.dll” Alias “\_HypNamAskEx@8” (ByVal hRApp As Integer, ByVal szRetbuf As String) As Integer**

## 変数 説明

hRApp HypMultiGet( )または HypConstruct( )からの Hyperion Enterprise Retrieve アプリケーションハンドル

szRetbuf エンティティ ID を返すためのバッファ。バッファの長さは、少なくとも HYP\_SIZEFULLNAME+1 文字（C 言語の場合は SIZEFULLNAME+1 文字）にする必要があります。

戻りコード：

コード	説明
0	成功
NONE	エラー発生

例：

```
szEntity$ = SPACE (HYP_SIZEFULLNAME+1)
ret% = HypNameAskEx(hRApp%, szEntity$)
If ret% = 0 Then szEntity$ = CToBStr(szEntity$)
```

C 言語では次の形式を使用します。

```
short WINAPI HypNamAskEx( HRETRIEVEAPP hRApp, LPSTR szRetbuf)
```

## HypNameListAskEx( ) - エンティティ一覧の選択

この関数を実行すると、Hyperion Enterprise の「エンティティ一覧の選択」ダイアログボックスが呼び出され、ユーザが選択したエンティティ一覧が取得されます。ユーザが「キャンセル」を選択した場合、エンティティ一覧は取得されません。

次の形式を使用します。

```
Declare Function HypNameListAskEx Lib "heaccess.dll" Alias
    "_HypNameListAskEx@8" (ByVal hRApp As Integer, ByVal szRetbuf As String) As
Integer
```

## 変数 説明

hRApp HypMultiGet( )からの Hyperion Enterprise Retrieve アプリケーションハンドル

szRetbuf エンティティ一覧を返すためのバッファ。バッファの長さは、少なくとも HYP\_SIZELABEL +1 文字（C 言語では SIZENAME+1 文字）にする必要があります。

戻りコード：

コード	説明
0	成功
NONE	エラー発生

例：

```
szEntityList$ = SPACE (HYP_SIZELABEL+1)
ret% = HypNameListAskEx(hRApp%, szEntityList$)
If ret% = 0 Then szEntityList$ = CToBStr(szEntityList$)
```

C 言語では次の形式を使用します。

```
short WINAPI HypNameListAskEx( HRETRIEVEAPP hRApp, LPSTR szRetbuf )
```

## HypPerAsk( ) - 期間の選択

この関数を実行すると、Hyperion Enterprise の「期間の選択」ダイアログボックスが呼び出され、ユーザが選択した期間が取得されます。ユーザが「キャンセル」を選択した場合、期間は取得されません。

次の形式を使用します。

```
Declare Function HypPerAsk Lib "heaccess.dll" Alias "_HypPerAsk@12" (ByVal
hRApp As Integer, fWantDate As Long, ByVal szRetbuf As String) As Long
```

### 変数 説明

hRApp	HypMultiGet( )または HypConstruct( )からの Hyperion Enterprise Retrieve アプリケーションハンドル
fWantDate	期間を期間番号または日付として返すいずれかのインジケータ（期間番号の場合は 0、日付の場合は 1）
szRetbuf	期間を返すためのバッファ。バッファは、少なくとも日付に 1 文字を足した長さにする必要があります。

fWantDate に 1 を指定した場合に利用可能な日付がないと、fWantDate の値は Hyperion Enterprise によってゼロに変更されます。たとえば、fWantDate に 1 を指定し、13 番目の会計年度の 13 番目の月を取得しようとする、fWantDate の値はゼロに変更され、szRetbuf の期間番号が返されます。

**注：** 日付（fWantDate = 1 の場合）は、月、日および年をカンマで区切った値として返されます（mm,dd,yy）。例えば、2003 年 3 月 31 日は"3,31,2003"として返されます。

戻りコード：

コード	説明
0	成功
NONE	エラー発生

例：

```
szPer$ = SPACE (HYP_SIZELABEL)
ret% = HypPerAsk(hRApp%, 0, szPer$)
If ret% = 0 Then szPer$ = CToBStr(szPer$)
```

C 言語では次の形式を使用します。

```
int WINAPI HypPerAsk( HRETRIEVEAPP hRApp, BOOL FAR * pfWantDate, LPSTR  
szRetbuf )
```

## HypSetDefJour( ) - デフォルト仕訳帳の設定

この関数を実行すると、アプリケーションに関連付けられたデフォルトの仕訳帳が変更されます。通常、このデフォルト値は、[仕訳帳の選択] ダイアログボックスで設定されます。詳しくは、[62 ページの「HypJourAsk\( \) - 仕訳帳の選択」](#)を参照してください。HypJourDetAsk( )を呼び出すには、まずデフォルトの仕訳帳を設定しておく必要があります。HypJourDetAsk 関数については、[63 ページの「HypJourDetAsk\( \) - 仕訳帳詳細の選択」](#)を参照してください。

次の形式を使用します。

```
Declare Function HypSetDefJour Lib "heaccess.dll" Alias "_HypSetDefJour@8"  
(ByVal hRApp As Integer, ByVal sigDefJour As Long) As Integer
```

### 変数 説明

hRApp HypConstruct( )からの Hyperion Enterprise Retrieve アプリケーションハンドル

sigDefJour 仕訳帳の記号

戻りコード：

コード	説明
0	成功
NONE	エラー発生

C 言語では次の形式を使用します。

```
SBOOL WINAPI HypSetDefJour(HRETRIEVEAPP hRApp, SIGNA sigDefJour)
```

## HypSetDefJourCat( ) - デフォルト仕訳帳データ種別の設定

この関数を実行すると、アプリケーションに関連付けられたデフォルトの仕訳帳データ種別が変更されます。通常、このデフォルト値は、[仕訳帳の選択] ダイアログボックス（HypJourAsk( )）で設定されます。HypJourDetAsk( )を呼び出すには、まずデフォルトの仕訳帳を設定しておく必要があります。HypJourDetAsk 関数については、[63 ページの「HypJourDetAsk\( \) - 仕訳帳詳細の選択」](#)を参照してください。

次の形式を使用します。

```
Declare Function HypSetDefJourCat Lib "heaccess.dll" Alias  
"_HypSetDefJourCat@8" (ByVal hRApp As Integer, ByVal sigDefJourCat As Long) As  
Integer
```

変数	説明
----	----

hRApp	HypConstruct( )からの Hyperion Enterprise Retrieve アプリケーションハンドル
sigDefJourCat	データ種別の記号

戻りコード：

コード	説明
0	成功
NONE	エラー発生

C 言語では次の形式を使用します。

SBOOL WINAPI HypSetDefJourCat(HRETREIVEAPP hRApp, SIGNA sigDefJourCat)

## HypSetDefJourPer - デフォルト仕訳帳期間の設定

この関数を実行すると、アプリケーションに関連付けられたデフォルトの仕訳帳期間が変更されます。通常、このデフォルト値は、[仕訳帳の選択] ダイアログボックス (HypJourAsk()) で設定されます。HypJourDetAsk() を呼び出すには、まずデフォルトの仕訳帳を設定しておく必要があります。HypJourDetAsk 関数については、[63 ページの「HypJourDetAsk\(\) - 仕訳帳詳細の選択」](#)を参照してください。

次の形式を使用します。

```
Declare Function HypSetDefJourPer Lib "heaccess.dll" Alias  
"_HypSetDefJourPer@8" (ByVal hRApp As Integer, ByVal sigDefJourPer As Long) As  
Integer
```

変数	説明
----	----

hRApp	HypConstruct( )からの Hyperion Enterprise Retrieve アプリケーションハンドル
sigDefJourPer	期間の記号

戻りコード：

コード	説明
0	成功
NONE	エラー発生

C 言語では次の形式を使用します。

SBOOL WINAPI HypSetDefJourPer(HRETREIVEAPP hRApp, SIGNA sigDefJourPer)

## HypSetMVMode( ) - HypVal 戻り値動作の設定

この関数を実行すると、HypValReturn( )関数の動作が制御されます。  
HypValReturn( )では、有効なデータ、データなしまたはエラーを表すコードが返されます。詳しくは、[76 ページの「HypValReturn\( \) - 最後の値取得のステータスを返す」](#)を参照してください。

次の形式を使用します。

**Declare Sub HypSetMVMode Lib “heaccess.dll” Alias “\_HypSetMVMode@4” (ByVal iMode As Long)**

ここで、iMode は次のいずれかのコードです。

0 = NA および ERR を許可

1 = NA の場合はゼロ

2 = ゼロのみ

戻りコード：

iMode 値	返される結果
0	エラーコード
1	データが使用不可の場合は 0、それ以外の場合はエラーコード
2	常に 0

例：

```
HypSetMVMode(1)
ret& = HypHPVAL(...)
If ret& <> 0 Then ret& = HypValReturn( )
```

C 言語では次の形式を使用します。

```
void WINAPI HypSetMVMode( int iMode )
```

## HypSetProgramName( ) - ログインユーザレポートのテキストの設定

Hyperion Enterprise のログインユーザレポートでプログラム名として表示するテキストを指定するには、この関数を使用します。この関数は、アプリケーションを開く前に呼び出す必要があります。HypSetProgramName( )を呼び出した後でHypConstruct( )またはHypMultiInit( )を使用して開いたすべてのアプリケーションでは、szProgram 引数で指定したテキストが使用されます。高度な関数EntOpenApplication( )を使用して開いたアプリケーションは、HypSetProgramName( )の影響を受けません。詳しくは、[67 ページの「HypMultiInit\( \) - 複数アプリケーションへのログオン」](#)、[29 ページの「HypConstruct\( \) - アプリケーションを開く」](#)または[242 ページの「EntOpenApplication\( \) - アプリケーションを開く」](#)を参照してください。

次の形式を使用します。

Declare Function HypSetProgramName Lib "heaccess.dll" Alias  
"\_HypSetProgramName@4" (ByVal szProgram As String) As Integer

ここで、szProgram は Hyperion Enterprise のログインユーザレポートでプログラム名として表示するテキストです。

戻りコード：

コード	説明
0	成功
NONE	エラー発生

例：

```
ret% = HypSetProgramName("Our Custom Program")  
ret2% = HypConstruct(hRApp%, application$, userID$, password$)
```

C 言語では次の形式を使用します。

short WINAPI HypSetProgramName(LPCSTR szProgram)

## HypValidateLnkParams( ) - リンクパラメータのチェック

この関数を実行すると、データを設定するために HypHPLNK( ) 呼び出しに指定しようとしているパラメータが有効であるかどうか判别されます。これにより、エンティティ、データ種別、期間および期間単位が有効であること、そして勘定科目が入力勘定科目であることが確認されます。詳しくは、[51 ページの「HypHPLNK\( \) - データ値の保存」](#)を参照してください。

次の形式を使用します。

Declare Function HypValidateLnkParams Lib "heaccess.dll" Alias  
"\_HypValidateLnkParams@24" (ByVal hRApp As Integer, ByVal szEntity As String,  
ByVal szCat As String, ByVal szAcct As String, ByVal szPer As String, ByVal szFreq As  
String) As Integer

### 変数 説明

hRApp HypConstruct( )からの Hyperion Enterprise Retrieve アプリケーションハンドル

szEntity エンティティ ID

szCat データ種別 ID

szAcct 勘定科目 ID

szPer 期間の日付 ("02/08/03")、期間 ID ("Feb 03") または期間番号 ("2")

szFreq レポート期間単位または文字列 freq.view。ここで、freq は Hyperion Enterprise の期間単位 ("M"、"Q" など)、view はレポート表示形式です。期間単位については、[20 ページの「期間単位と表示形式」](#)を参照してください。

戻りコード：

コード	説明
0	すべてのパラメータが有効
HACC_BAD_ACCOUNT	エラー発生、無効な szAcct
HACC_BAD_CATEGORY	エラー発生、無効な szCat
HACC_BAD_FREQUENCY	エラー発生、無効な szFreq
HACC_BAD_NAME	エラー発生、無効な szEntity
HACC_BAD_PERIOD	エラー発生、無効な szPer
HACC_BAD_SELECT	エラー発生

例：

```
ret% = HypValidateLnkParams(hRApp%, "US100", "ACTUAL", "INCOME",
    "JAN 03", "M.CTD")
```

C 言語では次の形式を使用します。

```
short WINAPI HypValidateLnkParams(HRETRIEVEAPP hRApp, LPCSTR szEntity,
LPCSTR szCat, LPCSTR szAcct, LPCSTR szPer, LPCSTR szFreq)
```

## HypValidateParams( ) - パラメータのチェック

この関数を実行すると、エンティティ、データ種別、勘定科目、期間および期間単位が有効であるかどうかチェックされます。この関数を使用すると、HypHPVAL()で使用する予定の引数をテストすることができます。

HypValidateParams( )は HypValidateLnkParams( )と似ていますが、HypValidateParams( )では勘定科目が入力勘定科目である必要がない点が異なります。詳しくは、[73 ページの「HypValidateLnkParams\( \) - リンクパラメータのチェック」](#)を参照してください。

次の形式を使用します。

```
Declare Function HypValidateParams Lib "heaccess.dll"
Alias " _HypValidateParams@24" (ByVal hRApp As Integer, ByVal szEntity As String,
ByVal szCat As String, ByVal szAcct As String, ByVal szPer As String, ByVal szFreq As
String) As Integer
```

### 変数 説明

hRApp HypConstruct( )からの Hyperion Enterprise Retrieve アプリケーションハンドル

szEntity エンティティ ID

szCat データ種別 ID

szAcct 勘定科目 ID

szPer 期間の日付 ("02/08/03")、期間ラベル ("Feb 03") または期間番号 ("2")

## 変数 説明

szFreq レポート期間単位または文字列 freq.view。ここで、freq は Hyperion Enterprise の期間単位 (“M”、“Q” など)、view はレポート表示形式です。期間単位については、[20 ページの「期間単位と表示形式」](#)を参照してください。

戻りコード：

コード	説明
0	すべてのパラメータが有効
HACC_BAD_ACCOUNT	エラー発生、無効な szAcct
HACC_BAD_CATEGORY	エラー発生、無効な szCat
HACC_BAD_FREQUENCY	エラー発生、無効な szFreq
HACC_BAD_NAME	エラー発生、無効な szEntity
HACC_BAD_PERIOD	エラー発生、無効な szPer
HACC_BAD_SELECT	エラー発生

例：

```
ret% = HypValidateParams(hRApp%, "US100", "ACTUAL", "INCOME", "JAN  
03", "M.PER")
```

C 言語では次の形式を使用します。

```
short WINAPI HypValidateParams(HRETRIEVEAPP hRApp, LPCSTR szEntity, LPCSTR  
szCat, LPCSTR szAcct, LPCSTR szPer, LPCSTR szFreq)
```

## HypValidateParamsEx( ) - パラメータのチェック

この関数を実行すると、エンティティ、データ種別、勘定科目、期間および期間単位が有効であるかどうかチェックされます。この関数を使用すると、HypHPVAEx( )で使用する予定の引数をテストすることができます。

次の形式を使用します。

```
Declare Function HypValidateParamsEx Lib "heaccess.dll"  
Alias "_HypValidateParams@32" (ByVal hRApp As Integer, ByVal szEntity As String,  
ByVal szCat As String, ByVal szAcct As String, ByVal szPer As String, ByVal szFreq As  
String, ByVal szParent As String, wType As Integer) As Integer
```

## 変数 説明

hRApp HypConstruct( )からの Hyperion Enterprise Retrieve アプリケーションハンドル

szEntity エンティティラベル

szCat データ種別ラベル

szAcct 勘定科目ラベル

## 変数 説明

- szPer 期間の日付 ("02/08/03")、期間ラベル ("Feb 03") または期間番号 ("2")
- szFreq レポート期間単位または文字列 freq.view。ここで、freq は Hyperion Enterprise の期間単位 ("M"、"Q" など)、view はレポート表示形式です。期間単位については、「Hyperion Enterprise 開発者用ツールキットについて」の章の「期間単位と表示形式」を参照してください。
- szParent 親エンティティのラベル。
- wType 要求されるデータのタイプ。次のコードのいずれかを使用します。
- ID\_REGULAR
  - ID\_PROPORTIONAL
  - ID\_ELIMINATION
  - ID\_CONTRIBUTION
  - ID\_ADJUSTMENT
  - ID\_TRANSLATION
  - ID\_TRANSLATION\_FORCE

戻りコード：

コード	説明
0	すべてのパラメータが有効
HACC_BAD_ACCOUNT	エラー発生、無効な szAcct
HACC_BAD_CATEGORY	エラー発生、無効な szCat
HACC_BAD_FREQUENCY	エラー発生、無効な szFreq
HACC_BAD_NAME	エラー発生、無効な szEntity
HACC_BAD_PERIOD	エラー発生、無効な szPer
HACC_BAD_SELECT	エラー発生

例：

```
ret% = HypValidateParamsEx(hRApp%, "US100", "ACTUAL", "INCOME",  
    "JAN 03", "M.PER", "EUROPE", ID_CONTRIBUTION)
```

C 言語では次の形式を使用します。

```
short WINAPI HypValidateParamsEx(HRETRIEVEAPP hRApp, LPCSTR szEntity,  
LPCSTR szCat, LPCSTR szAcct, LPCSTR szPer, LPCSTR szFreq, LPCSTR szParent, WORD  
wType)
```

## HypValReturn( ) - 最後の値取得のステータスを返す

この関数を実行すると、最後に実行したデータ取得関数 HypHPVAL( )または HypHPVALEx( )に加え、次の関数のステータスを示す値が返されます。

- HypHPACC( )
- HypHPBET( )
- HypHPCAD( )
- HypHPCAL( )
- HypHPCDE( )
- HypHPCUREx( )
- HypHPDRV( )
- HypHPECO( )
- HypHPFLW( )
- HypHPFRE( )
- HypHPFUL( )
- HypHPHEA( )
- HypHPINC( )
- HypHPINP( )
- HypHPJOUR( )
- HypHPKEYEx( )
- HypHPKEY( )
- HypHPOWN( )
- HypHPPBE( )
- HypHPPCH( )
- HypHPSCA( )

上記の関数によって返されたエラーの詳細情報を取得するには、HypValReturn( )を使用します。エラーが返されたら、一覧の他の関数を呼び出す前にHypValReturn( )を呼び出してください。

次の形式を使用します。

**Declare Function HypValReturn Lib “heaccess.dll” Alias “\_HypValReturn@0” ( ) As Long**

戻りコード：

コード	説明
RET_OK (0)	要求の成功。
RET_NA (-1)	値が使用不可。要求は有効ですが、データベースでデータが見つかりませんでした。
HACC_BAD_... ( >= 10 )	エラーの発生。要求がデータベースの現在の構造に対して有効ではありませんでした。特定のエラーコードは、TOOLKIT.BAS または TOOLKIT.H ファイルを参照するか、各関数のトピックを参照してください。

例：

```
ValRet& = HypValReturn( )  
Select Case ValRet&  
Case Ret_NA  
RetString$ = "Data not available"  
Case HACC_BAD_ACCOUNT  
RetString$ = "Bad Account"  
Case Ret_OK  
RetString$ = "OK"  
Case Else  
RetString$ = "Error"  
End Select
```

C 言語では次の形式を使用します。

```
int WINAPI HypValReturn( void )
```

## この章の内容

表関数の概要.....	79
Hyp...()関数を使用した表の選択と選択解除 .....	79
Hyp...()関数での表の使用.....	80
関連する表と Hyp...()関数.....	80
組織構造体と Hyp...()関数.....	81
アルファベット順の表関数リファレンス.....	82

この章では、Hyperion Enterprise API 関数を使用して、Hyperion Enterprise アプリケーションの情報が保存されている表から情報を取り出す方法を説明します。

## 表関数の概要

Hyperion Enterprise アプリケーションに関する情報は、表にまとめられます。例えば、勘定科目表には勘定科目とその属性が、データ種別表にはアプリケーションの中のデータ種別とその属性が表形式でまとめられています。この章では、Hyperion Enterprise API の中の関数を使用して表から情報を取り出す方法を説明します。

表関数はスプレッドシートアドイン関数とともに使用するように設計されています。スプレッドシートアドイン関数については、[第2章「スプレッドシートのアドイン関数」](#)で説明しています。表関数はより高度な高レベル関数の代わりに使用されます。表関数は Hyp...() という形式に従います。

大部分の表関数には、引数として Hyperion Enterprise Retrieve アプリケーションからのハンドル (hRApp) が必要です。このハンドルは HypConstruct() 関数を使用してアプリケーションを開くときに取得します。複数のアプリケーションを開くときは、ハンドルを HypMultiGet() または HypMultiDefault() から取得できます。詳しくは、[23 ページの「単一アプリケーションの管理関数」](#) または [24 ページの「複数アプリケーションの管理関数」](#) を参照してください。

## Hyp...() 関数を使用した表の選択と選択解除

表の中の情報にアクセスするには、まず HypLockEx() 関数を使用して表を選択する必要があります。この関数は Hyperion Enterprise API に使用する表を通知して表の使用を可能にします。一度に複数の表を選択できます。表の使用を終了したら、HypUnLockEx() 関数を使用して表を選択解除します。

多くの表には別の表が関連付けられています。Hyperion Enterprise SE では、メインの表を選択すると、関連付けられている表を使用できます。例えば、通貨とエンティティの表は組織表を選択すると使用できるようになります。Hyperion Enterprise では、関連付けられている表を個別に選択できます。

表を選択または選択解除するときは、HYP\_ID\_ASSOC (C 言語では ID\_ASSOC) で OR 演算および表 ID を使用して、関連付けられている表を自動的に選択または選択解除できます。表を個別に選択または選択解除するのではなく、関連付けられている表をすべて選択または選択解除するには、HYP\_ID\_ASSOC を必ず使用する必要があります。HYP\_ID\_ASSOC については、[339 ページの「関連付けられている表」](#)を参照してください。

HypConstruct( )関数または HypMultiInit( )関数を使用してアプリケーションを開くと、組織表 (ID\_ORGANIZATION)、データ種別表 (ID\_CATEGORY)、勘定科目表 (ID\_ACCOUNT) およびそれらに関連付けられている表が自動的に選択されます。HypConstruct( )はレポート期間単位とレポート表示形式の表 (ID\_RPTFREQ と ID\_RPTVIEW) も自動的に選択します。HypDestruct( )関数は以前に選択されていた表を自動的に選択解除します。

アプリケーションが連動組織で設定されている場合、データ種別を変更するときに、組織表 (ID\_ORGANIZATION) とその関連付けられている表を再選択する必要があります。組織表を選択するとき、新しいデータ種別の記号を sigkey 引数として渡します。詳しくは、[120 ページの「組織構造体と高度な関数」](#)を参照してください。

## Hyp...( )関数での表の使用

HypLockEx( )関数を使用して表を選択した後、[表 22](#)の中に記載されている関数を使用して選択されている表から情報を取り出すことができます。

**表 22** HypLockEx( )で表を選択した後に使用する関数

関数	説明
HypEnumEx( )	表内のレコードを列挙します。
HypFindEx( )	表内の特定のレコードを検索します。
HypQueryEx( )	特定のレコードに関する情報を取得します。

表内の各レコードは、記号と呼ばれる一意の値で識別されます。記号は、レコードに関する情報を要求するときに指定します。例えば、EntQueryEx( )関数を呼び出すときは、この関数に記号を渡し、必要な情報を含んでいるレコードを指定します。各レコードの記号は HypFindEx( )を呼び出すことによって取得できます。HypEnumEx( )を使用して表の中のレコードを列挙すると、各レコードの記号が返されます。

## 関連する表と Hyp...( )関数

他の表と関連している表があります。別の表が密接に関連している表の選択、アクセス、または選択解除を行うときには必ず、関連している表内のレコードを指

定する必要があります。例えば、勘定科目一覧入力表（ID\_ACCTLISTENTRY）には、勘定科目一覧内のレコードに関する情報が含まれています。勘定科目一覧の記号（勘定科目一覧表 ID\_ACCTLIST 内の）を渡して、勘定科目一覧を指定する必要があります。

次の関数を使用するときは、関連するレコードの記号を **sigKey** 引数として渡します。

- **HypEnum( )** - 表内のレコードの列挙
- **HypFindEx( )** - 記号の検索
- **HypLockEx( )** - 表の選択
- **HypQueryEx( )** - Hyperion Enterprise の表のクエリ
- **HypUnlockEx( )** - 表の選択解除

**注：** **sigKey** は **HypLockEx( )** の引数であるため、子表を選択する前に関連する表を選択する必要があります。

関連する Hyperion Enterprise の表の一覧については、[340 ページの「関連する表」](#)を参照してください。

## 組織構造体と Hyp...( )関数

表 23 は組織構造体に関する情報を取り出す関数のリストです。

**表 23** 組織情報を取り出すための関数

関数	説明
HypEnumOrgNames( )	組織内のノードを列挙します。
HypFindNameInOrgEx( )	組織内のノードを検索します。
HypGetChild( )	子ノードの記号を取得します。
HypGetOrgLevel( )	ノードの組織レベルを取得します。
HypGetSibling( )	組織内の次の兄弟ノードの記号を取得します。
HypGetTopNodeEx( )	組織内のトップノードの記号を取得します。
HypIsNameParentEx( )	指定されているエンティティが親エンティティであるかどうかを確認します。

Hyperion Enterprise では、組織構造体に関するほとんどの情報がノード表（ID\_NODES）に保存されています。ノード表内の各レコードは、ノードと呼ばれます。各ノードには、組織構造体内の特定のエンティティに関する情報が含まれています。この情報により、組織構造体内の親、直属の子または同じレベルの兄弟の検索が可能になります。

上の表の関数のほとんどは、エンティティの記号ではなく、ノード表内のノードの記号を返します。エンティティ記号を取得するには、**HypQueryEx( )**関数とクエリ属性 **NAMESIG** を使用してノード表のクエリを実行します。

アプリケーションが連動組織を使用するように設定されている場合、組織構造体はデータ種別と期間によって変化します。これらの関数ではデータ種別や期間を指定する方法がありません。現在のデータ種別と期間が使用されます。

アプリケーションが連動組織を使用する場合、Ent...() 形式の高レベル関数を使用します。詳しくは、[120 ページの「組織構造体と高度な関数」](#)を参照してください。

## アルファベット順の表関数リファレンス

ここでは、すべての表関数をアルファベット順に説明します。

### DateConv( ) - 日付の変換

1 つのデータ種別、例えば参照データ種別で 1 つの期間を与えられたとき、DateConv( ) は別のデータ種別とレポート期間単位、例えば最下位データ種別と期間単位での等価の期間を計算します。レポート期間単位については、[20 ページの「期間単位と表示形式」](#)を参照してください。

次の形式を使用します。

```
Declare Function DateConv Lib "heaccess.dll" Alias "_DateConv@28" (ByVal hRApp As Integer, ByVal iMethod As Integer, ByVal szRefCat As String, ByVal szRefPeriod As String, ByVal iOffset As Integer, ByVal szFreq As String, ByVal szBaseCat As String) As Integer
```

変数	説明
----	----

hRApp	HypConstruct( )からの Hyperion Enterprise Retrieve アプリケーションハンドル
-------	--

iMethod	次のコードのいずれか。 <ul style="list-style-type: none"><li>• DCONV_CMO は算出期間を毎年リセットします。</li><li>• DCONV_CUR は算出期間を毎年リセットしません。</li></ul>
---------	--

szRefCat	参照データ種別の ID。
----------	--------------

szRefPeriod	参照期間の ID。
-------------	-----------

iOffset	結果をオフセットする期間数。
---------	----------------

szFreq	最下位レポート期間単位の ID。
--------	------------------

szBaseCat	最下位データ種別の ID。
-----------	---------------

戻りコード：

コード	説明
期間番号	成功
BAD_PERIOD	エラー発生

例：

```
period% = DateConv(hRApp%, DCONV_CUR, "ACTUAL", "JAN 03", 0,
    "MON", "ACTUAL");
```

C 言語では次の形式を使用します。

```
short WINAPI DateConv(HRETRIEVEAPP hRApp, short iMethod, LPSTR szRefCat,
    LPSTR szRefPeriod, short iOffset, LPSTR szFreq, LPSTR szBaseCat);
```

## HypCatGetNumPeriodsEx( ) - データ種別に含まれる期間数の算出

この関数は、データ種別に含まれている期間数を任意の期間単位に対して計算します。1 つの期間単位のデータ種別と別の期間単位が与えられると、この関数は使用可能な期間数を返します。例えば、12 の期間を含む月次データ種別と四半期の期間単位が与えられた場合、HypCatGetNumPeriodsEx( )はそのデータ種別の期間数として 4（四半期）を返します。

次の形式を使用します。

```
Declare Function HypCatGetNumPeriodsEx Lib "heaccess.dll" Alias
    "HypCatGetNumPeriodsEx_@12" (ByVal hRApp As Integer, ByVal sigCat As Long,
    ByVal sigFreq As Long) As Integer
```

### 変数 説明

hRApp HypConstruct( )からの Hyperion Enterprise Retrieve アプリケーションハンドル

sigCat データ種別記号。

sigFreq 期間単位記号

戻りコード：

コード	説明
データ種別 sigCat が期間単位 sigFreq に対して持っている期間数	成功
NONE	エラー発生

例：

```
sigCat& = HypFindEx(hRApp% HYP_ID_CATEGORY, HYP_NONE, "ACTUAL")
sigFreq& = FREQ_QUARTER
numPer% = HypCatGetNumPeriodsEx(hRApp%, sigCat&, sigFreq&)
```

C 言語では次の形式を使用します。

```
short WINAPI HypCatGetNumPeriodsEx(HRETRIEVEAPP hRApp, SIGNA sigCat,
    SIGNA sigFreq);
```

## HypCatGetPerShortEx( ) - データ種別期間単位内の期間 ID の取得

この関数は、指定されたデータ種別での指定された期間単位に基づく期間 ID を返します。

次の形式を使用します。

```
Declare Function HypCatGetPerShortEx Lib "heaccess.dll"  
Alias " _HypCatGetPerShortEx@24" (ByVal hRApp As Integer, ByVal sigCatAs Long,  
ByVal sigPer As Long, ByVal sigFreq As Long, ByVal cchTag As Integer, ByVal pzTag As  
String) As Integer
```

### 変数 説明

hRApp HypConstruct( )からの Hyperion Enterprise Retrieve アプリケーションハンドル

sigCat データ種別記号

sigPer 指定されたデータ種別での期間番号

sigFreq 期間単位、または指定されたデータ種別の期間単位を使用するには HYP\_NONE (C 言語では NONE)

cchTag pszTag バッファの長さ

pzTag 期間 ID を返すためのバッファ

戻りコード：

コード	説明
結果の文字列の長さ	成功
0	エラー発生

例：

```
szPer$ = SPACE(10) '任意のサイズ  
ret% = HypCatGetPerShortEx(hRApp%, sigCat&, sigPer&, sigFreq&,  
Len(szPer$), szPer$)  
if ret% Then szPer$ = cToBStr (szPer$)
```

C 言語では次の形式を使用します。

```
short WINAPI HypCatGetPerShortEx(HRETRIEVEAPP hRApp, SIGNA sigCat, SIGNA  
sigPer, SIGNA sigFreq, short cchTag, char * pzTag);
```

## HypCatMapPeriodEx( ) - 期間の期間単位へのマッピング

この関数は、与えられた期間単位での期間を別の期間単位での対応する期間にマッピングします。

次の形式を使用します。

Declare Function HypCatMapPeriodEx Lib "heaccess.dll"  
Alias "\_HypCatMapPeriodEx@20" (ByVal hRApp As Integer, ByVal sigCat As  
Long,ByVal sigFreq As Long, ByVal sigPer As Long, ByVal sigAltFreq As Long) As Long

**変数      説明**

hRApp      HypConstruct( )からの Hyperion Enterprise Retrieve アプリケーションハンドル

sigCat      データ種別記号

sigFreq      sigPer の期間単位記号

sigPer      期間単位 sigFreq の期間記号

sigAltFreq      期間 sigPer をマッピングする先の期間の元になる期間単位の記号。そのデータ種別のデフォルト期間単位を使用するには HYP\_NONE (C 言語では NONE) を指定します。

戻りコード :

コード	説明
期間単位の期間記号	成功
BAD_PERIOD	エラー発生

例 :

データ種別内の四半期データの最初の期間が必要で、その週次期間番号を知る必要があるとします。この場合は、次の例を使用できます。

```
sigPerWeekly& = HypCatMapPeriodEx(hRApp%, sigCat&, FREQ_QUARTER, 0,  
FREQ_WEEK)
```

上の例では、データ種別が月次で 1 月から始まるとすると、Q1 は月次にマッピングされ、3 月 (3) が取得されます。この関数は次に月次 (3) を週にマッピングし、W13 を取得します。3 月の最終週は 13 週目であるためです。これは、週を基にしてレポートを作成する場合に役に立ちますが、ここでは期間単位が異なっている 2 つのデータ種別から対応する値を取得する必要があります。

C 言語では次の形式を使用します。

```
SIGNA WINAPI HypCatMapPeriodEx(HRETRIEVEAPP hRApp, SIGNA sigCat, SIGNA  
sigFreq, SIGNA sigPer, SIGNA sigAltFreq);
```

## HypEnumAcctListEntriesEx( ) - 勘定科目一覧の勘定科目の列挙

この関数は、勘定科目一覧の中の各勘定科目を対象にしたユーザ定義のコールバック関数を呼び出して、指定された勘定科目一覧の全勘定科目を列挙します。連動一覧では、HypEnumAcctListEntriesEx( )は勘定科目記号をコールバック関数に渡します。固定一覧では、コールバック関数に渡される記号は、勘定科目一覧の入力表 (ID\_ACCTLISTENTRY) 内の入力の記号です。この表に対して勘定科目記号のクエリを行うことができます。

**注：** この関数を呼び出す前に、勘定科目一覧表（ID\_ACCTLIST）と勘定科目一覧の入力表（ID\_ACCTLISTENTRY）を選択する必要があります。

次の形式を使用します。

**Declare Function** HypEnumAcctListEntriesEx Lib “heaccess.dll”  
**Alias”** \_HypEnumAcctListEntriesEx@16” (ByVal hRApp As Integer,ByVal sigList As Long, ByVal lpCallBack As Long,ByVal lParam As Long) As Integer

# **変数      説明**

hRApp      HypConstruct( )からの Hyperion Enterprise Retrieve アプリケーションハンドル

sigList      列挙する勘定科目一覧の記号。

lpCallBack ユーザ定義のコールバック関数のアドレス。詳しくは、[318 ページの「CALLBACK12」](#)を参照してください。

lParam      ユーザ定義のコールバック関数に返されるパラメータ

戻りコード：

コード	説明
0	成功
非ゼロ値	列挙を打ち切るためにコールバック関数から返される値
NONE	エラー発生

例：

```
sigList& = HypFindEx(hRApp%, HYP_ID_ACCTLIST, HYP_NONE,
    szAcctList$)
If sigList& = HYP_NONE Then Return 'エラー
'固定一覧か連動一覧かをチェック。listType%を取得する
ret% = HypQueryEx(hRApp%, HYP_ID_ACCTLIST,LIST_TYPE,
sigList&, HYP_NONE, 2, listType%)
If ret% Then Return 'エラー
ret% = HypLockEx(hRApp%, HYP_ID_ACCTLISTENTRY,sigList&,
APILOCK_READONLY)
If ret% <> APILOCK_READONLY Then Return 'エラー
lParam& = listType%
ret% = HypEnumAcctListEntriesEx(hRApp%, sigList&,
AddressOf EnumAcctListEntriesCB, lParam&)
ret% = HypUnLockEx(hRApp%, HYP_ID_ACCTLISTENTRY, sigList&, 0)
.
.
.
'ユーザ定義の名前を持つコールバック関数
Public Function EnumAcctListEntriesCB(ByVal hRApp%, ByVal sigList&,
ByVal sigEntry&, ByVal lParam&) As Integer
'この例では一覧タイプをlParamで渡した
If lParam& = Asc("F") Then '固定一覧
'勘定科目記号を取得
```

```

        ret% = HypQueryEx(hRApp%, HYP_ID_ACCTLISTENTRY,
        ENTRY_SIG, sigEntry&, sigList&, 4, sigAcct&)
Else
    ' 連結一覧
    sigAcct& = sigEntry&
End If
' 勘定科目IDを表示
szAcct$ = Space(HYP_SIZEFULLACCT+1)
ret% = HypQueryEx(hRApp%, HYP_ID_ACCOUNTS, NAME, sigAcct&,
    Len(szAcct$), szAcct$)
If ret% = 0 Then szAcct$ = CToBStr(szAcct$)
ret% = MsgBox("Account: " & szAcct$)
' 列挙を続けるために0を返す
EnumAcctListEntriesCB% = 0
End Function

```

C 言語では次の形式を使用します。

```

short WINAPI HypEnumAcctListEntriesEx(HRETRIEVEAPP hRApp, SIGNA sigList,
CALLBACK12 lpCallBack, LONG lParam);

```

## HypEnumApplications( ) - アプリケーションの列挙

この関数は、Hyperion Enterprise の [新規アプリケーション] または [アプリケーションの追加] を使用して現在のマシンに設定されているすべての Hyperion Enterprise アプリケーションを列挙します。HypEnumApplications( )は HYPENT.INI ファイルを読み取り、コールバック関数を使用してアプリケーションを列挙します。

次の形式を使用します。

```

Declare Function HypEnumApplications Lib "heaccess.dll"
Alias "_HypEnumApplications@8" (ByVal lpfnCallBack As Long, ByVal lParam As
Long) As Integer

```

### 変数 説明

lpfnCallBack コールバック関数。詳しくは、[320 ページの「CALLBACKAPP（または CALLBACK6）」](#)を参照してください。

lParam コールバック関数に渡されるパラメータ。

戻りコード：

コード	説明
0	成功
1	エラーが発生したかアプリケーションが不在
非ゼロ値	列挙を打ち切るためにコールバック関数から返される値

例：

```

ret% = HypEnumApplications(AddressOf EnumAppsCallback, lParam&)
.

```

```

・
・
'コールバック関数
Public Function EnumAppsCallback(laddr as Long, ByVal lParam&) As Integer
    'C文字列をB文字列にコピー
    wLen% = EntVBGetCStrLen( lAddr&)
    szApp$=Space (wLen%)
    ret%=EntVBCopyStr(szApp$, lAddr&, wLen%)
    If UCASE(szApp$) <> "DEFAULT" Then
        ret% = MsgBox("Application: " & szApp$)
    End If
    EnumAppsCallback% = 0    "列挙を続ける
End Function

```

C 言語では次の形式を使用します。

```
short WINAPI HypEnumApplications(CALLBACKAPP lpCallBack, LONG lParam);
```

## HypEnum( ) - 表内のレコードの列挙

この関数はユーザ定義のコールバック関数を使用して、指定された表内のすべてのレコードの記号を列挙します。表 wTabId 内の各レコードごとに、HypEnumEx( ) はその記号を指定されたコールバック関数に渡します。コールバック関数から非ゼロ値を返すことによって列挙を打ち切ることができます。

次の形式を使用します。

```
Declare Function HypEnumEx Lib "heaccess.dll" Alias "_HypEnumEx@20" (ByVal
hRApp As Integer, ByVal wTabId As Integer, ByVal sigKey As Long, ByVal lpCallback As
Long, ByVal lParam As Long) As Integer
```

変数	説明
hRApp	HypConstruct( )からの Hyperion Enterprise Retrieve アプリケーションハンドル
wTabId	記号を列挙する表。表 ID の完全なリストについては、 <a href="#">付録 A「表 ID」</a> を参照してください。
sigKey	オプションのキー（通常は NONE）。wTabId が <a href="#">340 ページの「関連する表」</a> のリストに含まれている表の場合、このキーは必須です。
lpCallback	ユーザ定義のコールバック関数へのハンドル。詳しくは、 <a href="#">332 ページの「DWCALLBACK」</a> を参照してください。
lParam	ユーザ定義のコールバック関数に返されるパラメータ

**注：** 表からの列挙については、[171 ページの「EntEnum\( \) - 表内のレコードの列挙」](#)から始まる各表に関するトピックを参照してください。

[表 24](#) は、HypEnumEx( )関数を使用できない Hyperion Enterprise 表のリストです。

表 24 HypEnumEx( )では使用できない表

表	使用する関数
ID_ACCTLISTENTRY	HypEnumAcctListEntriesEx( )
ID_NODES	HypEnumOrgNames( )
ID_SUBACCTDET	HypEnumSubAcctSig( )
ID_NAMELISTENTRY	HypEnumNameListEntriesEx( )
ID_SECGRPTAB	EntEnum( )
ID_SECUSERTAB	EntEnum( )
ID_SECRIGHTS	EntEnum( )
ID_SHARES	EntEnum( )
ID_DATAFILE	HypHPVAL( )は、勘定科目の列挙後、各勘定科目のデータ値の取得に使用します。
ID_JOURNAL_DETAIL	EntEnum( )
ID_JOURNAL_HISTORY	EntEnum( )
ID_JOURNAL_HISTORY_DETAIL	EntEnum( )
ID_JOURNAL_TEMPLATES	EntEnum( )
ID_JOURNAL_TEMPLATES_DETAIL	EntEnum( )
ID_JOURNAL_PERIOD_INFO	EntEnum( )
ID_JOURNALS	EntEnum( )
ID_PSFDATA	NA

戻りコード：

コード	説明
0	成功
NONE	エラー発生
非ゼロ値	列挙を打ち切るためにコールバック関数から返される値

例：

この例では勘定科目一覧が列挙されます。勘定科目一覧内のエントリは列挙されません。

```
ret%= HypEnumEx(hRapp%, HYP_ID_ACCTLIST, HYP_NONE, AddressOf
    MyCallback, 0)
```

C 言語では次の形式を使用します。

```
short WINAPI HypEnumEx(HRETRIEVEAPP hRApp, short wTabId, SIGNA sigKey,
    DWCALLBACK lpCallBack, LONG lParam);
```

# HypEnumNameListEntriesEx( ) - エンティティ一覧内のエンティティの列挙

この関数は指定されたエンティティ一覧内のすべてのエンティティを列挙します。この関数は、ユーザの lParam とユーザ定義のコールバック関数を、エンティティ一覧内の入力とともに呼び出します。

**注：** この関数を呼び出す前に、エンティティ一覧表（ID\_NAMELIST）とエンティティ一覧の入力表（ID\_NAMELISTENTRY）を選択する必要があります。

次の形式を使用します。

Declare Function HypEnumNameListEntriesEx Lib “heaccess.dll”  
Alias” \_HypEnumNameListEntriesEx@16” (ByVal hRApp As Integer, ByVal sigList As Long, ByVal lpCallback As Long, ByVal lParam As Long) As Integer

変数	説明
hRApp	HypConstruct からの Hyperion Enterprise Retrieve アプリケーションハンドル
sigList	列挙する一覧
lpCallback	エントリに対するユーザ定義のコールバックルーチン。詳しくは、 <a href="#">318 ページの「CALLBACK12」</a> を参照してください。
lParam	コールバックのユーザ情報

戻りコード：

コード	説明
0	成功
NONE	エラー発生
非ゼロ値	列挙を打ち切るためにコールバック関数から返される値

例：

```
ret% = HypLockEx(hRApp%, HYP_ID_NAMELIST, HYP_NONE, APILOCK_READONLY)
If ret% <> APILOCK_READONLY Then Return 'エラー
sigList& = HypFindEx(hRApp%, HYP_ID_NAMELIST, HYP_NONE,szList$)
If sigList& <> HYP_NONE Then
ret% = HypLockEx(hRApp%, HYP_ID_NAMELISTENTRY, sigList&, APILOCK_READONLY)
If ret% = APILOCK_READONLY Then      'ok
ret% = HypEnumNameListEntriesEx(hRApp%, sigList&, AddressOf EnumEntityListEntriesCB, 0)
ret% = HypUnLockEx(hRApp%, HYP_ID_NAMELISTENTRY, sigList&, 0)
End If
End If
ret% = HypUnLockEx(hRApp%, HYP_ID_NAMELIST, HYP_NONE, 0)
.
```

```

.
.
'ユーザ定義の名前を持つコールバック関数
Public Function EnumEntityListEntriesCB(ByVal hRApp%,
ByVal sigList&, ByVal sigEntry&, ByVal lParam&) As Integer
ret% = HypQueryEx(hRApp%, HYP_ID_NAMELISTENTRY,
ENTRY_SIG, sigEntry&, sigList&, 4, sigEntity&)
'エンティティラベルを表示
szEntity$ = Space(HYP_SIZEFULLNAME+1)
ret% = HypQueryEx(hRApp%, HYP_ID_NAMES, NAME, sigEntity&,
Len(szEntity$), szEntity$)
If ret% = 0 Then szEntity$ = CToBStr(szEntity$)
ret% = MsgBox("Entity: " & szEntity$)
'列挙を続けるために0を返す
EnumEntityListEntriesCB% = 0
End Function

```

C 言語では次の形式を使用します。

```

short WINAPI HypEnumNameListEntriesEx(HRETRIEVEAPP hRApp, SIGNA sigList,
CALLBACK12 lpCallback, LONG lParam);

```

## HypEnumOrgNames( ) - 組織内のノードの列挙

組織情報はノード表 (ID\_NODES) に保存されています。ノード表内の各レコードは、ノードと呼ばれます。HypEnumOrgNames( )は組織内のすべてのノードを列挙します。ノードの列挙は指定されたノードから開始され、組織にリストされている順番に列挙されます。ノードについては、[81 ページの「組織構造体と Hyp... \( \)関数」](#)を参照してください。

**注：** 連動組織を使用するように設定されているアプリケーションでは、高レベル関数を使用するようお勧めします。詳しくは、[120 ページの「組織構造体と高度な関数」](#)を参照してください。

次の形式を使用します。

```

Declare Function HypEnumOrgNames Lib "heaccess.dll"
Alias "_HypEnumOrgNames@20" (ByVal hRApp As Integer, ByVal sigNode As Long,
ByVal lpCallBack As Long, ByVal lParam As Long, ByVal wMask As Integer) As Integer

```

### 変数 説明

hRApp	HypConstruct( )からの Hyperion Enterprise Retrieve アプリケーションハンドル
sigNode	開始ノードの記号
lpCallBack	ユーザ定義のコールバック関数。詳しくは、 <a href="#">332 ページの「DWCALLBACK」</a> を参照してください。
lParam	コールバック関数に渡されるユーザ定義のパラメータ
wMask	次の選択フラグの任意の組み合わせ。 <ul style="list-style-type: none"> <li>ENUM_DEP は直属の子のみを列挙します。ENUM_DEP は単独で使用します。</li> </ul>

## 変数 説明

- ENUM\_CON は指定されたノードより下のすべての親エンティティを列挙します。
- ENUM\_BAS は指定されたノードより下のすべての最下位エンティティを列挙します。
- ENUM\_UNOWNED はすべての親が未定義のエンティティを列挙します。
- ENUM\_SUBNAME はすべてのサブエンティティを列挙します。
- ENUM\_ALL は最下位と親エンティティおよび sigNode より下のサブエンティティを列挙します。

戻りコード：

コード	説明
0	成功
NONE	エラー発生
非ゼロ値	列挙を打ち切るためにコールバック関数から返される値

例：

```
'組織内のすべての最下位エンティティを列挙
'トップノードから開始
sigNode& = HypGetTopNodeEx(hRapp%, sigOrg&)
If sigNode& <> HYP_NONE Then
    ret% = HypEnumOrgNames(hRapp%, sigNode&, AddressOf EnumOrgCB,0,
        ENUM_BASE Or ENUM_SUBNAME)
End If
.
.
.
'ユーザ定義の名前を持つコールバック関数
Public Function EnumOrgCB(hSelect&, sigNode&, sigKey&, lParam&, apiS) As
Integer
'エンティティIDの取得と表示
ret% = EntQuery(ByVal hRapp%, ByVal sigNode&, ByVal lParam&)
szEntity$ = Space(HYP_SIZEFULLNAME+1)
ret% = HypQueryEx(hRapp%, HYP_ID_NAMES, HYP_NAME, sigEntity&,
    HYP_NONE, Len(szEntity$), szEntity$)
If ret% = 0 Then szEntity$ = CToBStr(szEntity$)
ret% = MsgBox("Entity: " & szEntity$)
'列挙を続けるために0を返す
EnumOrgCB% = 0
End Function
```

C 言語では次の形式を使用します。

```
short WINAPI HypEnumOrgNames(HRETRIEVEAPP hRapp, SIGNA sigNode, short
DWCALLBACK lpCallBack, LONG lParam, WORD wMask);
```

# HypEnumSubAcctSig( ) - Enumerate サブ勘定科目記号の列挙

この関数は、指定された主要勘定科目の一連のサブ勘定科目記号を列挙します。  
この関数は、サブ勘定科目表 (ID\_SUBACCTDET) を列挙するために、  
HypEnumEx( )関数の代わりに使用します。HypEnumSubAcctSig( )は勘定科目表  
(ID\_ACCOUNTS) 内の勘定科目記号を各サブ勘定科目ごとにコールバック関数に  
渡します。

次の形式を使用します。

Declare Function HypEnumSubAcctSig Lib “heaccess.dll” Alias  
“\_HypEnumSubAcctSig@16” (ByVal hRApp As Integer, ByVal sigAcct As Long, ByVal  
lpCallback As Long, ByVal lParam As Long) As Integer

変数	説明
hRApp	HypConstruct( )からの Hyperion Enterprise Retrieve アプリケーションハンドル
sigAcct	開始勘定科目記号。
lpCallback	ユーザ定義のコールバック関数。詳しくは、 <a href="#">332 ページの「DWCALLBACK」</a> を参照してください。
lParam	コールバック関数に渡されるユーザ定義のパラメータ。

戻りコード：

コード	説明
0	成功
なし	エラー発生
非ゼロ値	列挙を打ち切るためにコールバック関数から返される値

例：

```
ret% = HypEnumSubAcctSig(hRApp%, sigAcct%,  
    AddressOf EnumSubAcctCB, 0)  
.  
.  
.  
' コールバック関数  
Public Function EnumSubAcctCB(ByVal hRApp%, ByVal sigAcct%,  
    ByVal lParam%) As Integer  
    szAcct$ = Space(HYP_SIZEFULLACCT+1)  
    ret% = HypQueryEx(hRApp%, HYP_ID_ACCOUNTS, NAME, sigAcct%,  
        HYP_NONE, Len(szAcct$), szAcct$)  
    If ret% = 0 Then  
        szAcct$ = CToBStr(szAcct$)  
        ret2% = MsgBox(szAcct$)  
    End If  
    EnumSubAcctCB% = 0  
End Function
```

C 言語では次の形式を使用します。

```
short WINAPI HypEnumSubAcctSig(HRETRIEVEAPP hRApp, SIGNA sigAcct,  
DWCALLBACK lpCallback, SIGNA lParam);
```

## HypFindEx( ) - 記号の検索

この関数は、指定された表（wTabId）を与えられた検索文字列で検索し、検索文字列に一致するレコードの記号を返します。

表 25 は、HypFindEx()を使用できない Hyperion Enterprise の表のリストです。

表 25 HypFindEx( )では使用できない表

表	使用する関数
ID_ACCTCONVERT	EntFind( )
ID_DATAFILE	NA
ID_INTCODET	この表の後の注意書きを参照。
ID_JOURNAL_DETAIL	EntFind( )
ID_JOURNAL_HISTORY	EntFind( )
ID_JOURNAL_HISTORY_DETAIL	EntFind( )
ID_JOURNAL_TEMPLATES	EntFind( )
ID_JOURNAL_TEMPLATES_DETAIL	EntFind( )
ID_JOURNAL_PERIOD_INFO	EntFind( )
ID_JOURNALS	EntFind( )
ID_NAMECONVERT	EntFind( )
ID_NAMES	HypGetNameSig( )
ID_NODES	HypFindNameInOrgEx( )
ID_PSFDATA	NA
ID_SHARES	EntFind( )

**注：** 会社間詳細表（ID\_INTCODET）には HypFindEx()を使用しないでください。代わりに、会社間照合セット表（ID\_ICSET）に HypFindEx()を使用して、必要な会社間セットを検索し、その後クエリ属性 INTCO\_GROUPSIG（および表 ID\_ICSET）で HypQueryEx()を呼び出し、表 ID\_INTCODET 内の最初の詳細レコードの記号を取得します。残りの詳細レコードは、クエリ属性 INTCO\_NEXTDET と表 ID\_INTCODET を使用して、各詳細レコードごとに HypQueryEx()を呼び出して取得します。

次の表では、キーは `string` ではなく `TOOLINC.H` ファイルで定義されている特殊な `struct` です。

- `ID_SECRIGHTS` 表には `SECURRIGHTSKEY struct` を使用します。
- `ID_SECGRPTAB` 表には `SECURGROUPKEY struct` を使用します。

次の形式を使用します。

**Declare Function HypFindEx Lib “heaccess.dll” Alias “\_HypFindEx@16” (ByVal hRApp As Integer, ByVal wTabId As Integer, ByVal sigKey As Long, ByVal szKey As String) As Long**

**変数 説明**

`hRApp` `HypConstruct( )`からの Hyperion Enterprise Retrieve アプリケーションハンドル

`wTabId` 検索する表の ID。表 ID の完全なリストについては、[付録 A「表 ID」](#)を参照してください。

`sigKey` 2 次的なキー（通常は `NONE`）。`wTabId` が [80 ページの「関連する表と Hyp...\(\)関数」](#)のリストに含まれている表の場合、このキーは必須です。

`szKey` 検索文字列。

戻りコード：

コード	説明
記号	成功
NONE	エラー発生

例：

```
sigCat& = HypFindEx(hRApp%, HYP_ID_CATEGORY, HYP_NONE, "ACTUAL")
```

C 言語では次の形式を使用します。

```
SIGNA WINAPI HypFindEx(HRETRIEVEAPP hRApp, short wTabId, SIGNA sigKey, void FAR * pzKey);
```

## HypFindNameInOrgEx( ) - ノード ID の検索

この関数は、指定された組織内の特定のエンティティ（ノード）のノード記号を返します。ノードは組織内でのエンティティの位置を定義します。1 つのエンティティの事例が 1 つの組織内に複数存在できます。各事例は一意のノード記号を持っています。ノードについては、[81 ページの「組織構造体と Hyp...\(\)関数」](#)を参照してください。

**注：** 連動組織を使用するように設定されているアプリケーションでは、高レベル関数を使用するようお勧めします。詳しくは、[120 ページの「組織構造体と高度な関数」](#)を参照してください。

次の形式を使用します。

Declare Function HypFindNameInOrgEx Lib "heaccess.dll"  
Alias "\_HypFindNameInOrgEx@16" (ByVal hRApp As Integer, ByVal sigOrg As Long,  
ByVal sigNode As Long, ByVal sigEntity As Long) As Long

#### 変数 説明

hRApp HypConstruct( )からの Hyperion Enterprise Retrieve アプリケーションハンドル

sigOrg 検索する組織。

sigNode 検索を開始するノード。これが HYP\_NONE (C 言語では NONE) の場合、検索は組織の  
トップノードから始まります。

sigEntity 検索するエンティティの記号。

戻りコード：

コード	説明
指定された エンティティの記号	成功
NONE	名前が指定された組織に存在しないためにエラーが発生

例：

```
ret% = HypHPKEYEX(hRApp%, "ORGANIZATION", sigOrg&) 'Get
sigOrg&
If ret% = 0 Then
sigNode& = HypFindNameInOrgEx(hRApp%, sigOrg&, HYP_NONE,
sigEntity&)
End If
```

C 言語では次の形式を使用します。

SIGNA WINAPI HypFindNameInOrgEx(HRETREVEAPP hRApp, SIGNA sigOrg,  
SIGNA sigNode, SIGNA sigEntity);

## HypGetCatPerFreq( ) - データ種別内の期間番号と期間単位の取得

この関数は、特定期間のデータ種別内での期間番号を返します。

次の形式を使用します。

Declare Function HypGetCatPerFreq Lib "heaccess.dll"  
Alias "\_HypGetCatPerFreq@16" (ByVal hRApp As Integer, ByVal szPeriodAs String,  
ByVal sigCat As Long, sigFreq As Long) As Long

#### 変数 説明

hRApp HypConstruct( )からの Hyperion Enterprise Retrieve アプリケーションハンドル

szPeriod 期間 ID、例えば JAN 03

## 変数 説明

sigCat データ種別記号

sigFreq NULL、または期間単位記号を返す先のバッファ

戻りコード：

コード	説明
期間番号	成功
NONE	エラー発生

例：

```
sigPeriod& = HypGetCatPerFreq(hRApp%, "APR 03", sigCat,  
    sigFreq&)
```

上の例では、sigFreq&で戻る期間単位記号は szPeriod で使用されている期間単位で、この場合は月次です。Q1 03 を使用すると、sigFreq&は四半期の期間単位記号になります。この期間単位はデータ種別の実際の期間単位に一致する必要はありません。

C 言語では次の形式を使用します。

```
SIGNA WINAPI HypGetCatPerFreq(HRETRIEVEAPP hRApp, LPSTR szPeriod, SIGNA  
sigCat, SIGNA * psigFreq);
```

## HypGetChild( ) - 子ノードの取得

この関数は、子ノードの記号を返します。ノードについては、[81 ページの「組織構造体と Hyp...\(\)関数」](#)を参照してください。

**注：** 連動組織を使用するように設定されているアプリケーションでは、高レベル関数を使用するようお勧めします。詳しくは、[120 ページの「組織構造体と高度な関数」](#)を参照してください。

次の形式を使用します。

```
Declare Function HypGetChild Lib "heaccess.dll" Alias "_HypGetChild@12" (ByVal  
hRApp As Integer, ByVal sigNode As Long, ByVal fExpandSubentities As Long) As Long
```

## 変数 説明

hRApp HypConstruct( )からの Hyperion Enterprise Retrieve アプリケーションハンドル

sigNode 子ノードの記号

fExpandSubentities 次のいずれかの値。

- 1 = True。付属の下位構造体を取得します。
- 0 = False。付属の下位構造体を取得しません。

戻りコード：

コード	説明
子ノードの記号	成功
NONE	エラー発生

例：

```
sigChild& = HypGetChild(hRApp%, sigNode&, 1)
```

C 言語では次の形式を使用します。

```
SIGNA WINAPI HypGetChild(HRETRIEVEAPP hRApp, SIGNA sigNode, BOOL  
fExpandSubentities);
```

## HypGetNameSig( ) - エンティティ記号の取得

この関数は、与えられたアプリケーションハンドル、エンティティ、およびサブエンティティ条件コードに基づいてエンティティ記号を返します。表 26 に可能な 3 つのサブエンティティ条件コードを示します。

表 26 HypGetNameSig( )のサブエンティティ条件コード

コード	返される記号
NAME_ IMPLICIT	entity.subentity 組み合わせの記号、エンティティのみが指定されている場合でも同じ。
NAME_ EXPLICIT	明示エンティティのみの記号。サブエンティティ付きのエンティティでは、このコードはエラーを返します。明示エンティティは有効なエンティティとはみなされないためです。
NAME_ ADMIN	明示エンティティのみの記号。そのエンティティにサブエンティティが付いているかどうかは関係ありません。

例えば、エンティティ ITALY が下位構造体 TRAN を持っているとします。サブエンティティ条件コードを NAME\_IMPLICIT に設定すると、この関数は ITALY.TRAN の記号を返します。それがデータの保存場所であるからです。コードを NAME\_EXPLICIT に設定すると、この関数への呼び出しは失敗します。ITALY のみでは有効なエンティティとみなされないためです。NAME\_ADMIN を使用すると、ITALY の記号が戻ります。そのエンティティに関連付けられているデータがないためです。

次の形式を使用します。

```
Declare Function HypGetNameSig Lib "heaccess.dll" Alias "_HypGetNameSig@12"  
(ByVal hRApp As Integer, ByVal szEntity As String, ByVal fImpliedSubEntities As Integer)  
As Long
```

変数	説明
hRApp	HypConstruct( )からの Hyperion Enterprise Retrieve アプリケーションハンドル

変数	説明
szEntity	検索対象のエンティティ
fImpliedSubEntities	次のコードのいずれか。 <ul style="list-style-type: none"> <li>● NAME_IMPLICIT</li> <li>● NAME_EXPLICIT</li> <li>● NAME_ADMIN</li> </ul>

戻りコード：

コード	説明
エンティティ記号	成功
NONE	エラー発生

例：

```
'次の行はItaly.Tranの記号を返します
sigEntity& = HypGetNameSig(hRApp%, "Italy", NAME_IMPLICIT)
'次の行はHYP_NONEを返します
sigEntity& = HypGetNameSig(hRApp%, "Italy", NAME_EXPLICIT)
'次の行はItalyのみの記号を返します
sigEntity& = HypGetNameSig(hRApp%, "Italy", NAME_ADMIN)
```

C 言語では次の形式を使用します。

```
SIGNA WINAPI HypGetNameSig(HRETRIEVEAPP hRApp, LPSTR szEntity, short
fImpliedSubEntities);
```

## HypGetOrgLevel( ) - 組織内レベルの取得

この関数は、指定されたノードの組織内でのレベルを返します。ノードについては、[81 ページの「組織構造体と Hyp...\(\)関数」](#)を参照してください。

**注：** 連動組織を使用するように設定されているアプリケーションでは、高レベル関数を使用するようお勧めします。詳しくは、[120 ページの「組織構造体と高度な関数」](#)を参照してください。

次の形式を使用します。

```
Declare Function HypGetOrgLevel Lib "heaccess.dll" Alias "_HypGetOrgLevel@16"
(ByVal hRApp As Integer, ByVal sigTopNode As Long, ByVal sigTarget As Long, ByVal
nLevel As Integer) As Integer
```

変数	説明
hRApp	HypConstruct( )からの Hyperion Enterprise Retrieve アプリケーションハンドル
sigTopNode	組織内の最上位ノード、またはカウントを開始するノードの記号

## 変数 説明

sigTarget	組織レベルを調べる対象のノードの記号
nLevel	通常は 0 または sigTopNode のレベルより 1 つ少ない値

戻りコード：

コード	説明
組織レベル	成功
0	エラー発生

例：

```
ret% = HypHPKEYEx(hRApp%, "ORGANIZATION", sigOrg%) 'sigOrg&を取得
If ret% = 0 Then
sigTop% = HypGetTopNodeEx(hRApp%, sigOrg%)
If sigTop% = HYP_NONE Then ret% = HYP_NONE
End If
If ret% = 0 Then
nLevel% = HypGetOrgLevel(hRApp%, sigTop%, sigTarget%, 0)
If nLevel% = 0 Then ret% = HYP_NONE 'エラー
End If
```

C 言語では次の形式を使用します。

```
short WINAPI HypGetOrgLevel(HRETRIEVEAPP hRApp, SIGNA sigTopNode, SIGNA
sigTarget, short nLevel);
```

## HypGetPerViewEx( ) - 期間単位と表示形式の取得

この関数は、指定されたレポート期間単位の期間単位と表示形式を返します。  
次の形式を使用します。

```
Declare Function HypGetPerViewEx Lib "heaccess.dll" Alias
"_HypGetPerViewEx@16" (ByVal hRApp As Integer, ByVal szFreq As String, sigFreq As
Long, iView As Integer) As Integer
```

## 変数 説明

hRApp	HypConstruct( )からの Hyperion Enterprise Retrieve アプリケーションハンドル
szFreq	レポート期間単位 (DAI, DYPD, WEE, WYTD, MON, YTD, QUA, QYTD, TRI, TYTD, HAL、HYTD, YEA, DAY)。詳しくは、 <a href="#">20 ページの「期間単位と表示形式」</a> を参照してください。
sigFreq	指定されたレポート期間単位の期間単位を返す場所。
iView	指定されたレポート期間単位 (VIEW_YTD または VIEW_PERIODIC) の表示形式を返す場所。

戻りコード：

コード	説明
0	成功
NONE	エラー発生

例：

```
ret% = HypGetPerViewEx(hRApp%, " QTD" ,SigFreq&,iview%)
```

C 言語では次の形式を使用します。

```
short WINAPI HypGetPerViewEx(HRETRIEVEAPP hRApp, LPSTR szFreq, SIGNA *  
psigFreq, short * piView);
```

## HypGetSibling( ) - 兄弟の取得

この関数は、組織内の次の兄弟ノードの記号を返します。

次の形式を使用します。

```
Declare Function HypGetSibling Lib "heaccess.dll" Alias "_HypGetSibling@8"  
(ByVal hRApp As Integer, ByVal sigNode As Long) As Long
```

### 変数 説明

hRApp HypConstruct( )からの Hyperion Enterprise Retrieve アプリケーションハンドル

sigNode ノードの記号

戻りコード：

コード	説明
次の兄弟ノードの記号	成功
NONE	エラー発生

例：

```
sigSibling& = HypGetSibling(hRApp%, sigNode&)
```

C 言語では次の形式を使用します。

```
SIGNA WINAPI HypGetSibling(HRETRIEVEAPP hRApp, SIGNA sigNode);
```

## HypGetTopNodeEx( ) - トップノードの取得

この関数は、組織内のトップノードの記号を返します。返されるトップノード記号は、HypEnumOrgNames( )と組み合わせて、組織内のすべてのエンティティを含むリストの作成に使用できます。ノードについては、[81 ページの「組織構造体と Hyp...\(\)関数」](#)を参照してください。

**注：** 連動組織を使用するように設定されているアプリケーションでは、高レベル関数を使用するようお勧めします。詳しくは、[120 ページの「組織構造体](#)

[と高度な関数](#)」を参照してください。組織内のトップノードの記号を取得するには、組織表 (ID\_ORGANIZATION) とクエリ属性 TOPNODE を指定して EntQueryEx()を使用します。

次の形式を使用します。

```
Declare Function HypGetTopNodeEx Lib "heaccess.dll" Alias  
"_HypGetTopNodeEx@8" (ByVal hRApp As Integer, ByVal sigOrg As Long) As Long
```

#### 変数 説明

hRApp HypConstruct( )からの Hyperion Enterprise Retrieve アプリケーションハンドル

sigOrg 組織記号

戻りコード：

コード	説明
組織内の一番上のノードの記号	成功
NONE	エラー発生

例：

```
ret% = HypHPKEYEx(hRApp%, "ORGANIZATION", sigOrg&) 'sigOrg&を取得  
If ret% = 0 Then  
sigTop& = HypGetTopNodeEx(hRApp%, sigOrg&)  
End If
```

C 言語では次の形式を使用します。

```
SIGNA WINAPI HypGetSibling(HRETRIEVEAPP hRApp, SIGNA sigNode);
```

## HypIsNameParentEx( ) - エンティティが親であることの確認

この関数は、エンティティが親エンティティかどうかを確認します。ノードについては、[81 ページの「組織構造体と Hyp...\( \)関数](#)」を参照してください。

**注：** 連動組織を使用するように設定されているアプリケーションでは、高レベル関数を使用するようお勧めします。詳しくは、[120 ページの「組織構造体と高度な関数](#)」を参照してください。

次の形式を使用します。

```
Declare Function HypIsNameParentEx Lib "heaccess.dll"  
Alias "_HypIsNameParentEx@12" (ByVal hRApp As Integer, ByVal sigEntity As Long,  
ByVal sigOrg As Long) As Integer
```

**変数 説明**

hRApp    HypConstruct( )からの Hyperion Enterprise Retrieve アプリケーションハンドル

sigEntity    エンティティ記号

sigOrg    検索対象の組織

sigOrg が NONE に設定されている場合、Hyperion Enterprise はそのエンティティがシステムの中のどこかで親になっているかどうかをチェックします。sigOrg に組織記号が指定されている場合、Hyperion Enterprise はそのエンティティがその組織内で親になっているかどうかをチェックします。エンティティは組織表示で子が表示されなくても親になっている可能性があります。

戻りコード：

コード	説明
TRUE (1)	エンティティは親です。
FALSE (0)	エンティティは親ではありません。

例：

```
isParent% = HypIsNameParentEx(hRApp%, sigEntity&, HYP_NONE)
If isParent% Then
ret% = MsgBox("Unable to put data in parent company")
End If
```

C 言語では次の形式を使用します。

```
SBOOL WINAPI HypIsNameParentEx(HRETREVEAPP hRApp, SIGNA sigEntity,
SIGNA sigOrg);
```

## HypLockEx( ) - 表の選択

この関数は、指定された 1 つまたは複数の表を選択します。表のデータにアクセスするにはその前に表を選択する必要があります。HypLockEx( )は、リクエストされた表がすでにメモリに入っていない場合、ディスクから読み込みます。その表がすでにメモリに入っている場合、HypLockEx( )はその表の使用回数を増加します。表は使用後 HypUnlockEx( )を呼び出して選択解除します。

**注：** 表 ID と ID\_ASSOC 定数を OR 演算で結合して、表とそれに関連付けられている表を同時に選択解除できます。詳しくは、[339 ページの「関連付けられている表」](#)を参照してください。

[表 27](#) は HypLockEx( )で選択すべきではない表のリストです。

表 27 HypLockEx( )で選択できない表

表	使用する関数
ID_DATAFILE	EntDataFileOpen( )
ID_JOURNAL_DETAIL	EntSelectTableAdd( )
ID_JOURNAL_HISTORY_DETAIL	EntSelectTableAdd( )
ID_JOURNAL_TEMPLATE_DETAIL	EntSelectTableAdd( )
ID_JOURNAL_PERIOD_INFO	EntSelectTableAdd( )

次の形式を使用します。

**Declare Function HypLockEx Lib "heaccess.dll" Alias "\_HypLockEx@16" (ByVal hRApp As Integer, ByVal wTabId As Integer, ByVal sigKey As Long, ByVal wAttr As Integer) As Integer**

#### 変数 説明

**hRApp** HypConstruct( )からの Hyperion Enterprise Retrieve アプリケーションハンドル

**wTabId** 選択する表の表 ID。関連付けられているすべての表を選択するには、「表 ID」付録内の表 ID 関連トピックに記載されている表 ID の 1 つを、定数 ID\_ASSOC と OR 演算で結合します。

**sigKey** キーレコード（通常は NONE）。これが要求される場合、wTabId に指定された表を選択する前にキーを含む表を選択する必要があります。詳しくは、付録 A「表 ID」を参照してください。

**wAttr** APILOCK\_READONLY を指定すると表が読み取り専用で選択されます。

戻りコード：

コード	説明
NONE	表へのアクセス拒否
APILOCK_READONLY	読み取り専用アクセスで開いている表

例：

```
rc% = HypLockEx(hRApp%, HYP_ID_ACCTLIST Or HYP_ID_ASSOC,
    HYP_NONE, APILOCK_READONLY)
If rc% <> APILOCK_READONLY Then MsgBox("Error selecting account
    list table")
```

C 言語では次の形式を使用します。

**short WINAPI HypLockEx(HRETREVEAPP hRApp, short wTabId, SIGNA sigKey, short wAttr);**

## HypQryRptFreqEx( ) - レポート期間単位のクエリ

この関数は、指定された Hyperion Enterprise の期間単位と表示形式に対応するレポート期間単位を取得します。レポート期間単位の ID またはその説明を取得できます。

次の形式を使用します。

**Declare Function HypQryRptFreqEx Lib “heaccess.dll” Alias  
“\_HypQryRptFreqEx@24” (ByVal hRApp As Integer, ByVal sigFreq As Long, ByVal  
wView As Integer, ByVal wAttr As Integer, ByVal wMax As Integer, ByVal pzbuf As String)  
As Integer**

### 変数 説明

hRApp HypConstruct( )からの Hyperion Enterprise Retrieve アプリケーションハンドル

sigFreq FREQ\_MONTH や FREQ\_QUARTER などの期間単位

wView VIEW\_YTD や VIEW\_PERIODIC など、指定された期間単位に関連付けられている表示形式

wAttr レポート期間単位の ID の取得には NAME、説明の取得には DESC を使用します。

wMax wAttr バッファの長さ

pzBuf レポート期間単位の ID または説明を返す先のバッファ。バッファのサイズは、wAttr が NAME の場合は、少なくとも HYP\_SIZERPTFREQ+1 (C 言語では SIZERPTFREQ+1)、wAttr が DESC の場合は、少なくとも HYP\_SIZEDESC+1 (C 言語では SIZEDESC+1) に十分な大きさである必要があります。

戻りコード：

コード	説明
0	成功
NONE	エラー発生

例：

```
szRptFreq$ = space(HYP_SIZERPTFREQ+1)
ret% = HypQryRptFreqExRpt(hRApp%, FREQ_MONTH, VIEW_YTD, NAME,
Len(szRptFreq$) szRtpFreq$)
If ret% = 0 Then szRptFreq$ = CToBStr(szRptFreq$)
```

C 言語では次の形式を使用します。

```
short WINAPI HypQryRptFreqEx(HRETREIVEAPP hRApp, SIGNA sigFreq, short
wView, short wAttr, short wMax, void * pzBuf);
```

## HypQueryEx( ) - Hyperion Enterprise の表のクエリ

この関数は、Hyperion Enterprise の表のクエリとその表内の入力に関する情報の取得を可能にします。取得できる情報の種類については、[付録 B「クエリ属性」](#)を参照してください。Visual Basic プログラマは、文字列値のクエリを行う場合、

HypQueryExStr()関数を使用する必要があります。HypQueryExStr()は TOOLKIT.BAS ファイルで宣言されているもので、HypQueryEx()のクローンを string 用に変更したものです。

次の形式を使用します。

```
Declare Function HypQueryEx Lib "haccess.dll" (ByVal hRApp As Integer, ByVal wTabId As Integer, ByVal wAttr As Integer, ByVal sigRecd As Long, ByVal sigKey As Long, ByVal wLen As Integer, pzBuf As Any) As Integer
```

#### 変数 説明

hRApp	HypConstruct( )からの Hyperion Enterprise Retrieve アプリケーションハンドル
wTabId	クエリする表の ID。表 ID の完全なリストについては、 <a href="#">付録 A「表 ID」</a> を参照してください。
wAttr	レコードのクエリ対象のフィールド。wTabId に指定されている表のクエリ属性の 1 つで、apiStruct を必要としないもの。
sigRecd	クエリ対象のレコード番号
sigKey	NONE、または関連する表のキー。wTabId が <a href="#">80 ページの「関連する表と Hyp...()関数」</a> のリストに含まれている表の場合、このキーは必須です。
wLen	pzBuf の長さ
pzBuf	要求した情報を返す先のバッファ。この引数は、クエリ内容に対応したデータ型である必要があります。詳しくは、 <a href="#">付録 B「クエリ属性」</a> を参照してください。

戻りコード：

コード	説明
0	成功
NONE	エラー発生

例：

データ種別内の期間数を取得するには

```
ret% = HypQueryEx(hRApp%, HYP_ID_CATEGORY, HYP_CAT_NUMPERIODS, sigCat%, HYP_NONE, Len(numPeriods%), numPeriods%)
```

C 言語では次の形式を使用します。

```
short WINAPI HypQueryEx(HRETRIEVEAPP hRApp, short wTabId, short wAttr, SIGNA sigRecd, SIGNA sigKey, short wLen, void FAR * pzBuf);
```

## HypUnLockEx( ) - 表の選択解除

この関数は、HypLockEx( )関数で選択した表を選択解除します。HypMultiDeinit( ) または HypDestruct( )を呼び出してアプリケーションを閉じる前に、選択されていたすべての表を選択解除する必要があります。

**注：** 表 ID と ID\_ASSOC 定数を OR 演算で結合して、表とそれに関連付けられている表を同時に選択解除できます。詳しくは、[339 ページの「関連付けられている表」](#)を参照してください。

次の形式を使用します。

**Declare Function HypUnLockEx Lib “heaccess.dll” Alias “\_HypUnLockEx@16”  
(ByVal hRApp As Integer, ByVal wTabId As Integer, ByVal sigKey As Long, ByVal bDiscard  
As Integer) As Integer**

**変数 説明**

- hRApp** HypConstruct( )からの Hyperion Enterprise Retrieve アプリケーションハンドル
- wTabId** 選択解除する表の表 ID。付録 A「表 ID」にリストされている表 ID の 1 つ、またはすべての表を指定するために NONE を使用します。また、関連付けられているすべての表を選択解除するには、表 ID と ID\_ASSOC を OR 演算で結合します。
- sigKey** キーのレコード。これは NONE、または表を選択したときに sigKey を指定した場合は、sigKey の値です。
- bDiscard** 表がメモリから自動的に削除されるべきか否かを示すための値で、1 (True) または 0 (False) です。

戻りコード

コード	説明
0	成功
NONE	エラー発生

例：

すべての表を選択解除するには

```
ret% = HypUnLockEx(hRApp%, HYP_NONE, HYP_NONE, 0)
```

個々の表とそれに関連付けられているすべての表を選択解除するには

```
ret% = HypUnLockEx(hRApp%, wTabId% OR HYP_ID_ASSOC, HYP_NONE, 0)
```

C 言語では次の形式を使用します。

**short WINAPI HypUnLockEx(HRETRIEVEAPP hRApp, short wTabId, SIGNA sigKey,  
SBOOL bDiscard);**



## この章の内容

高度な関数の概要 .....	109
アプリケーションを開く .....	110
表と高度な関数 .....	110
スプレッドシートのアドイン関数と高度な関数の組み合わせ .....	113
選択ダイアログボックス関数 .....	114
データの操作 .....	114
組織構造体と高度な関数 .....	120
データの視点の設定 .....	121
サーバのタスク .....	121
Hyperion Enterprise の INI ファイル関数 .....	125
セキュリティ .....	125
使用されなくなった Ent...()関数から新しい Ent...()関数への置き換え .....	127
アルファベット順の高度な関数のリファレンス .....	127

この章では、Hyperion Enterprise API の複雑な関数について説明します。

## 高度な関数の概要

Hyperion Enterprise API のほとんどの高度な関数は、Hyperion Enterprise 製品自体によって使用される基本の Hyperion Enterprise API 関数を直接呼び出します。高度な関数は、Hyperion Enterprise の威力と能力を最大限に引き出します。

ほとんどの高度な関数には、選択した表のハンドルが必要です。これは、表の hSelect と呼ばれます。このハンドルを取得するには、EntSelect() を呼び出して表を選択します。

EntSelect() などの一部の関数は、hSelect ではなくアプリケーションのハンドルを必要とします。hApp と呼ばれるアプリケーションハンドルは、EntOpenApplication() を呼び出してアプリケーションを開いたときに取得します。

**注：** hApp は、Hyp...() の形式を取るスプレッドシートのアドイン関数によって使用される Hyperion Enterprise Retrieve アプリケーションハンドル (hRApp) とは異なります。

プログラムの一般的な流れは次のとおりです。

- アプリケーションを開きます。詳しくは、[110 ページの「アプリケーションを開く」](#)を参照してください。
- 表を選択します。詳しくは、[111 ページの「高度な関数を使用した表の選択と選択解除」](#)を参照してください。
- 必要な操作を実行します。
- 表の選択を解除します。
- アプリケーションを閉じます。

スプレッドシートのアドイン関数 `HypConstruct()` または `HypMultiInit()` を使用してアプリケーションを開いた場合は、[113 ページの「スプレッドシートのアドイン関数と高度な関数の組み合わせ」](#) で、`hApp` および `hSelect` を取得して高度な関数を使用する方法について参照してください。

## アプリケーションを開く

まず、`EntOpenApplication()` 関数を呼び出してアプリケーションを開く必要があります。そうしないと、そのアプリケーションの他の関数を呼び出すことはできません。`EntOpenApplication()` によって、アプリケーションのハンドルが提供されます。C 言語では、このハンドルはタイプ `HAPP` として定義されます。

アプリケーションを使用した後は、`EntCloseApplication()` 関数を呼び出してアプリケーションを閉じ、プログラムを終了する必要があります。

スプレッドシートのアドイン関数 `HypConstruct()` とは異なり、`EntOpenApplication()` では表は選択されません。アプリケーションを操作するには、1 つまたは複数の表を選択する必要があります。

## 表と高度な関数

Hyperion Enterprise アプリケーションに関する情報は、表にまとめられます。ここでは、表を操作して必要な情報を取得する方法について説明します。

### apiStruct 構造体の使用

表を操作する高度な関数の多くは、引数を持っています。この引数は、`apiStruct` と呼ばれる構造体です。この構造体は、`TOOLKIT.BAS` および `TOOLINC.H` ファイルで定義されています。残りの引数で扱いきれない情報を関数に渡すために使用されます。

ほとんどの表には `apiStruct` は必要ありません。`apiStruct` 引数には `NULL` を渡すことができます。Visual Basic で `NULL` を渡す方法については、[16 ページの「Visual Basic プログラミング上の注意」](#) を参照してください。

`apiStruct` を使用する必要がある場合は、タイプ `APISTRUCT` の変数を宣言して構造体を割り当て、`EntInitApiStruct()` 関数を呼び出して構造体を初期化し、行おうとしている操作に必要な特定のフィールドを設定する必要があります。

**注：** apiStruct 引数を持つコールバック関数と Spyworks の両方を使用する場合は、[283 ページの「EntVBCopyData\(\) - データのコピー」](#)を参照して SpyWorks の long 引数を apiStruct に変換してください。

例：

```
'データ種別の期間1のIDを取得（“Jan 03” など）
DIM apiS As APISTRUCT
Call EntInitApiStruct(hApp&, apiS)
apiS.lStartPeriod = 1
apiS.u_ApiQry.sPeriod_View = HYP_NONE 'データ種別の期間単位を使用
szPeriod$ = SPACE(HYP_SIZELABEL+1)
ret% = EntQueryEx(hSelect&, HYP_ID_CATEGORY, CAT_PER_SHORT, sigCat&,
HYP_NONE, Len(szPeriod$), szPeriod$, apiS)
If ret% = 0 Then szPeriod$ = CToBStr(szPeriod$)
```

## 高度な関数を使用した表の選択と選択解除

Hyperion Enterprise アプリケーションを開いた後は、アクセスする表を選択する必要があります。表を選択するときは、Hyperion Enterprise API によってアクセスする表が準備されます。次のガイドラインに従って、表を選択してください。

- 最初の表を選択するには、EntOpenApplication()を使用してアプリケーションを開き、EntSelect()を呼び出します。
- 表をさらに選択するには、EntSelectAdd()を呼び出します。EntSelect()を使用して最初の表を選択したときに受け取った、選択済みの表のハンドル hSelect を EntSelectAdd()に渡します。表は必要なだけ選択できます。
- 表を使用した後は、EntUnSelect()を呼び出して表の選択を解除します。1 回の呼び出しで、選択されているすべての表の選択を解除するように EntUnSelect()に指示することができます。選択されているすべての表は、プログラムの終了前に選択を解除する必要があります。

多くの表には別の表が関連付けられています。表を選択または選択解除するときは、HYP\_ID\_ASSOC（C 言語では ID\_ASSOC）で OR 演算および表 ID を使用して、関連付けられている表を自動的に選択または選択解除できます。表を個別に選択または選択解除するのではなく、関連付けられている表をすべて選択または選択解除するには、HYP\_ID\_ASSOC を必ず使用する必要があります。詳しくは、[339 ページの「関連付けられている表」](#)を参照してください。

**ヒント：** データ種別表、組織表および勘定科目表とこれらに関連付けられている表を選択します。多くの関数では、これらの表が使用可能であることが前提となっています。最適なパフォーマンスを得るには、データ種別表、組織表、勘定科目表の順番で選択します。

Hyperion Enterprise アプリケーションが連動組織を使用するように設定されている場合は、組織表を選択するときに、データ種別の記号を sigkey 引数として渡す必要があります。別のデータ種別に変更するときは、新しいデータ種別を sigKey 引数として指定し、組織表（および関連付けられている表）を選択し直します。詳しくは、[120 ページの「組織構造体と高度な関数」](#)を参照してください。

表の選択に使用する高度な関数には、コールバック関数を指定するための引数があります。この引数には NULL を渡すことができます。表が変更されるたびに表から読み取った表示またはその他の内容を更新する関数を Hyperion Enterprise API が呼び出すようにするには、コールバック関数を渡します。詳しくは、[112 ページの「選択のコールバック」](#)を参照してください。

## 選択のコールバック

表の選択に使用する高度な関数には、コールバック関数を指定するための引数があります。この引数には NULL を渡すことができます。表が変更されるたびに Hyperion Enterprise API が関数を呼び出すようにするか、表から読み取った表示またはその他の内容を更新する場合は、コールバック関数を渡します。

コールバック関数とその引数は、表の選択時に指定します。選択したすべての表のコールバック関数は、すべて hSelect に関連付けられます。hSelect は、選択された表のハンドルです。コールバック関数は、表の選択後に EntAppendToCBChain( ) を呼び出して指定することもできます。コールバック関数の使用を停止するように API に指示するには、EntDeleteFromCBChain( ) を呼び出します。

選択されている表が変更されると、Hyperion Enterprise API ではその hSelect ハンドルに関連付けられているコールバック関数をすべて呼び出します。コールバック関数には、変更された表の ID が渡されます。詳しくは、[328 ページの「CALLBACKSEL」](#)を参照してください。

## 表での高度な関数の使用

表から情報を取得するには、表を選択してから、[表 28](#) で説明されている関数を使用します。

**表 28** 表で使用する高度な関数

関数	説明
EntEnum( )	表内のレコードを列挙します。
EntFind( )	表内の特定のレコードを検索します。
EntQueryEx( )	特定のレコードに関する情報を取得します。

表内の各レコードは、記号と呼ばれる一意の値で識別されます。記号は、レコードに関する情報を要求するときに指定します。例えば、EntQueryEx( ) 関数を呼び出すときは、この関数に記号を渡し、必要な情報を含んでいるレコードを識別する必要があります。記号を取得するには、EntFind( ) 関数を呼び出して特定のレコードを検索します。EntEnum( ) を使用してレコードの記号を取得することもできます。EntEnum( ) は、表内のレコードの記号をすべて列挙して戻します。

これらの関数は、ほとんどの表で使用できます。特定の表から特定の情報を取得する場合は、その他多くの特殊な高度関数が用意されています。

## 関連する表と高度な関数

一部の表は、別の表に密接に関連付けられているので、表の選択、アクセス、または選択解除を行うときは、関連する表のレコードを指定する必要があります。例えば、勘定科目一覧入力表 (ID\_ACCTLISTENTRY) には、勘定科目一覧内のレコードに関する情報が含まれています。勘定科目一覧を指定するには、勘定科目一覧の記号 (勘定科目一覧表 ID\_ACCTLIST 内) を渡す必要があります。

次の API 関数のいずれかを呼び出すときは、関連するレコードの記号を sigKey 引数として渡します。

- EntDiscardChanges() - 変更の破棄
- EntEnum() - 表内のレコードの列挙
- EntFind() - 記号の検索
- EntIsModified() - 表が変更されているかどうかの確認
- EntIsSelected() - 表が選択されているかどうかの確認
- EntQueryEx() - クエリ情報
- EntSave() - 表の保存
- EntSelectAdd() - 使用する追加の表の選択
- EntSelectTable() - 使用する表の選択
- EntSelectTableAdd() - 使用する追加の表の選択
- EntUnSelect() - 表の選択解除

**注：** sigKey は EntSelectAdd() の引数であるので、子表を選択する前に関連する表を選択する必要があります。

関連する Hyperion Enterprise 表の一覧については、[340 ページの「関連する表」](#)を参照してください。

## スプレッドシートのアドイン関数と高度な関数の組み合わせ

高度な関数のほとんどは、引数に hApp または hSelect を使用します。hApp は、Hyperion Enterprise ソフトウェアアプリケーションを開いたときに返されます。hSelect は、表を選択したときに返されます。一方、スプレッドシートのアドイン関数は、引数に hRApp を使用します。HRApp は、HypConstruct() によって返される Hyperion Enterprise Retrieve アプリケーションのハンドルです。

スプレッドシートのアドイン関数である HypConstruct() または HypMultiInit() を使用してアプリケーションを開いた場合は、高度な関数を組み合わせて使用することもできます。[表 29](#) に、高度な関数と組み合わせて使用するスプレッドシートのアドイン関数を示します。

表 29 高度な関数とともに使用するスプレッドシートのアドイン関数

関数	説明
HypGethApp( )	Hyperion Enterprise Retrieve アプリケーションに関連付けられている hApp を取得します。
HypGethSelect( )	Hyperion Enterprise Retrieve アプリケーションに関連付けられている hSelect ハンドルを返します。
HypSethSelect( )	Hyperion Enterprise Retrieve アプリケーションに関連付けられている hSelect ハンドルを変更します。

EntUnSelect( )など、hSelect ハンドルを変更する可能性のある操作を実行するときは、HypSethSelect( )を呼び出す必要があります。

## 選択ダイアログボックス関数

API の標準の選択ダイアログボックスにアクセスするには、選択ダイアログボックス関数を使用します。選択ダイアログボックスは、勘定科目、勘定科目一覧、データ種別、期間単位、エンティティおよび期間を選択するために使用します。表 30 に、選択ダイアログボックス関数を示します。

表 30 選択ダイアログボックス関数

関数	説明
EntAcctAsk( )	勘定科目の選択に使用します。
EntAcctListAsk( )	勘定科目一覧の選択に使用します。
EntCatAsk( )	データ種別の選択に使用します。
EntEntityAsk( )	エンティティの選択に使用します。
EntEntityListAsk( )	エンティティ一覧の選択に使用します。
EntFreqAsk( )	期間単位の選択に使用します。
EntPerAsk( )	期間の選択に使用します。

## データの操作

Hyperion Enterprise のデータ値は、データファイル表 ID\_DATAFILE に格納されます。他の表とは異なり、この表を選択するには EntSelect( )ではなく EntDataFileOpen( )を呼び出します。データファイル表を選択するときは、データ種別とエンティティを指定します。

データファイル表を選択すると、EntQueryEx( )関数を使用してデータ値を取得できるようになります。EntUpdate( )関数は、データ値の書き込み、計算式の実行、既存のデータのクリア、または選択されているデータファイル表の消去に使用します。表を使用した作業が終わったら、EntUnSelect( )関数を使用してデータファイル表の選択を解除します。

データファイル表を選択した後は、他の API 関数で使用する apiStruct 構造体を設定する必要があります。表 31 に、EntUnSelect( )を含むすべての API 関数に対して apiStruct に設定する必要のあるフィールドを示します。

表 31 すべての API 関数用の apiStruct フィールド

フィールド	説明
sigCat	データ種別記号
sigName	エンティティ記号
sigParent	通常は NONE (wType が ID_REGULAR の場合)
wType	通常のデータの場合は ID_REGULAR。詳しくは、119 ページの「連結詳細値」を参照してください。

## データの取得

データファイル表を選択し、apiStruct 構造体を設定した後は、EntQueryEx( )関数でクエリ属性 DATAFILE\_GET を使用して、データ値を取得します。EntQueryEx( )のレコード記号には HYP\_NONE (C 言語では NONE) を使用します。

EntQueryEx( )を呼び出す前に、2 つのバッファを割り当てて、apiStruct にさらに多くのフィールドを設定する必要があります。2 つのバッファのサイズは、必要な期間数によって異なります。1 つのバッファはデータ値用で、これは各期間の 1 つのデータ値 (タイプ double) を格納するのに十分な大きさにする必要があります。もう 1 つのバッファはステータス情報用で、これは各期間の整数 (2 バイト) を格納する必要があります。

表 32 に、apiStruct に設定する必要のあるフィールドを示します。

表 32 データ取得用の apiStruct フィールド

フィールド	説明
sigCat	データ種別記号
sigName	エンティティ記号
sigParent	通常は NONE (wType が ID_REGULAR の場合)
wType	通常のデータの場合は ID_REGULAR。詳しくは、119 ページの「連結詳細値」を参照してください。
lStartPeriod	必要な最初の期間 (0 から始まる)
lEndPeriod	必要な最後の期間 (0 から始まる)
lpimrData	データバッファのアドレス
lpseStatus	ステータスバッファのアドレス
u_Dfa.sigAcct	勘定科目記号
u_Dfa.iFreq	デフォルトの期間単位の場合は NONE、または必要な期間単位の記号

フィールド	説明
u_Dfa.bAutoRecalc	データ取得時の計算式の自動再計算。1 (TRUE) または 0 (False) を指定する必要があります。
u_Dfa.bPeriodic	期別データの場合は 1 (TRUE)、あるいはデータ種別累計または他の累計データの場合は 0 (False)
u_Dfa.sViewType	NONE、または次のコードのいずれか データ種別累計の場合は VIEWTYPE_YTD 期別の場合は VIEWTYPE_PER 累計の場合は VIEWTYPE_TODATE (u_Dfa.sigAccumFreq を指定する場合)
u_Dfa.sigAccumFreq	累計期間単位。これは u_Dfa.sViewType が VIEWTYPE_TODATE である場合にのみ必要です。
u_Dfa.wRndFlag	データの四捨五入フラグ。次のコードのいずれかを指定する必要があります。 デフォルトの四捨五入の場合は DFPUT_RNDDEF 常に四捨五入する場合は DFPUT_ROUND 四捨五入しない場合は DFPUT_NOROUND

Microsoft Visual Basic のプログラマは、EntGetVarAddr()を使用して lpimrData および lpseStatus フィールドをバッファのアドレスに設定できます。u\_Dfa.sViewType、u\_Dfa.sigAccumFreq および u\_Dfa.wRndFlag のフィールドは、通常はレポート作成のみに使用されます。

EntQueryEx()は、実行に成功した場合は 0、エラーが発生した場合は NONE を返します。EntQueryEx()から結果が返された時点で、データバッファ (apiStruct.lpimrData) には要求した各期間のデータ値が含まれます。ステータスバッファ (apiStruct.lpseStatus) には、各期間のステータス値が含まれます。ステータス値 PS\_OK (TOOLINC.H ファイルを参照) は、その期間のデータがデータバッファに含まれていることを示し、PS\_NODATA はその勘定科目と期間のデータが存在しないことを示します。

例：

```

DIM apiS As APISTRUCT
ret% = EntDataFileOpen(hSelect&, ID_REGULAR, sigCat&,

sigEntity&, HYP_NONE, APILOCK_READONLY, 0)

If ret% = APILOCK_READONLY THEN          'ok

EntInitApiStruct (hApp&, apiS)
apiS.sigCat = sigCat&
apiS.sigName = sigEntity&
apiS.sigParent = HYP_NONE
apiS.wType = ID_REGULAR
apiS.lStartPeriod = 0
apiS.lEndPeriod = 0          '最初の期間を取得
apiS.lpimrData = EntGetVarAddr (dValue#)
apiS.lpseStatus = EntGetVarAddr (wStatus%)
apiS.u_Dfa.sigAcct = sigAcct&

```

```

apiS.u_Dfa.bPeriodic = 0
apiS.u_Dfa.iFreq = HYP_NONE
apiS.u_Dfa.sViewType = HYP_NONE
ret% = EntQueryStr(hSelect&, HYP_ID_DATAFILE,
'C言語のプログラマはEntQueryEx( )を使用して、vbNullStringの代わりに
    NULLを渡す必要あり
HYP_DATAFILE_GET, HYP_NONE, HYP_NONE, 0, vbNullString,apiS)
if ret% = 0 Then ret2% = MsgBox("The value is " & Str$(dValue#))
hSelect& = EntUnSelect(hSelect&, HYP_ID_DATAFILE,
HYP_NONE, 1, ret2%, apiS)      '完了後、表の選択を解除

End If

```

## データの更新

データを更新するには、データファイルテーブルの選択時に読み取り／書き込みアクセスを要求する必要があります。EntDataFileOpen( )関数を呼び出すときは、APILOCK\_READONLY ではなく APILOCK\_READWRITE を使用します。

また、データ用とステータス情報用の 2 つのバッファを割り当てる必要があります。これらのバッファのサイズは更新する期間数によって異なります。データバッファには、更新する期間ごとに 1 つのデータ値（タイプ double）を挿入します。ステータスバッファには定数 PS\_OK を挿入します。各期間については TOOLINC.H ファイルを参照してください。

次に、apiStruct を設定します。表 33 に、apiStruct に設定する必要があるフィールドを示します。

**表 33** データ更新用の apiStruct フィールド

フィールド	説明
sigCat	データ種別記号
sigName	エンティティ記号
sigParent	NONE
wType	ID_REGULAR
lStartPeriod	更新する最初の期間
lEndPeriod	更新する最後の期間
lpimrData	データバッファのアドレス
lpseStatus	ステータスバッファのアドレス
u_Dfa.sigAcct	勘定科目記号
u_Dfa.iFreq	デフォルトの期間単位の場合は NONE、または期間単位の記号
u_Dfa.bPeriodic	期別データの場合は 1 (TRUE)、またはデータ種別累計データの場合は 0 (False)
u_Dfa.sigAccumFreq	NONE、またはデータ種別のデフォルト期間単位の記号

フィールド	説明
u_Dfa.sViewType	NONE、または次のコードのいずれか データ種別累計の場合は VIEWTYPE_YTD 期別の場合は VIEWTYPE_PER

Visual Basic のプログラマは、EntGetVarAddr( )関数を使用して lpimrData および lpseStatus フィールドをバッファのアドレスに設定できます。

設定が完了したら、EntUpdate( )関数を呼び出してデータ値を更新します。EntUpdate( )を使用して適切な属性を渡すことで、計算式の実行、既存のデータのクリアまたは選択されているデータファイル表の消去を行うこともできます。データを更新した後は、EntSave( )関数を呼び出して変更を保存する必要があります。

表 34 に、EntUpdate( )の属性を示します。

**表 34** EntUpdate( )の属性

属性	説明
DATAFILE_CLEAR	既存のデータをクリアします。選択されているデータファイル表（エンティティおよびデータ種別）の勘定科目をすべてクリアするには、apiStruct の u_Dfa.sigAcct フィールドを HYP_NONE（C 言語の場合は NONE）に設定します。その勘定科目のみをクリアするには、特定の勘定科目記号に設定します。
DATAFILE_ERASE	選択されているデータファイル表を消去します。
DATAFILE_LOGICEXECUTE	計算式を実行します。データのある最後の期間まで実行するには、apiStruct の IEndPeriod フィールドを HYP_NONE（C 言語の場合は NONE）に設定します。IEndPeriod フィールドを開始点として使用するには、IStartPeriod フィールドを HYP_NONE に設定します。これらの両方のフィールドを HYP_NONE に設定すると、データのある期間すべての計算式が実行されます。apiStruct の u_Dfa.sigAcct フィールドは HYP_NONE に設定します。
DATAFILE_NODATAFLOW	算出勘定科目以外にゼロデータを生成します。
DATAFILE_PUT	データ値を更新します。
DATAFILE_SETPER_EJLCKED	期間の明示された仕訳帳を保護します。
DATAFILE_SETPER_EJUNLCK	期間の明示された仕訳帳の保護を解除します。
DATAFILE_SETPER_LCKED	期間を保護します。
DATAFILE_SETPER_UNLCK	期間の保護を解除します。

例：

```
Global Const PS_OK = 0
DIM apiS As APISTRUCT
ret% = EntDataFileOpen(hSelect&, ID_REGULAR, sigCat&,
```

```

        sigEntity&, HYP_NONE, APILOCK_READWRITE, 1)
If ret% = APILOCK_READWRITE THEN          'ok

dValue# = 100.35
wStatus% = PS_OK

EntInitApiStruct (hApp&, apiS)
apiS.sigCat = sigCat&
apiS.sigName = sigEntity&
apiS.wType = ID_REGULAR
apiS.lStartPeriod = 0
apiS.lEndPeriod = 0      '最初の期間を更新
apiS.lpimrData = EntGetVarAddr (dValue#)
apiS.lpseStatus = EntGetVarAddr (wStatus%)
apiS.u_Dfa.sigAcct = sigAcct&
apiS.u_Dfa.bAutoRecalc = 0;
apiS.u_Dfa.iFreq = HYP_NONE
apiS.u_Dfa.bPeriodic = 0
'C言語のプログラマはEntUpdate()を使用して、vbNullStringの代わりに
    NULLを渡す必要あり

ret% = EntUpdateStr (hSelect&, HYP_ID_DATAFILE, HYP_DATAFILE_PUT,
    HYP_NONE, HYP_NONE, vbNullString, apiS)

if ret% <> 0 Then ret2% = MsgBox("Error updating data")

'すべての期間と勘定科目の計算式を実行

apiS.u_Dfa.sigAcct = HYP_NONE
apiS.lStartPeriod = HYP_NONE
apiS.lEndPeriod = HYP_NONE

ret% = EntUpdateStr (hSelect&, HYP_ID_DATAFILE,
    HYP_DATAFILE_LOGICEXECUTE, HYP_NONE, HYP_NONE,
    vbNullString, apiS)

'完了後、表を保存して選択を解除

ret% = EntSave (hSelect&, HYP_NONE, HYP_NONE, 0);
hSelect& = EntUnSelect (hSelect&, HYP_ID_DATAFILE, HYP_NONE, 1,
    ret2%, apiS)

End If

```

## 連結詳細値

Hyperion Enterprise アプリケーションでは、消去、調整、通貨換算値、エンティティが連結中にその親に搬出した値などの連結詳細を格納できます。この機能はヘッダーファイルでは詳細記憶モデル（DSM）と呼ばれています。

ID\_TRANSLATION\_FORCE は ID\_TRANSLATION と似ていますが、親および子のエンティティが同じ通貨を使用している場合でも有効です。通貨が同じ場合、データは換算されず、換算データファイル表は存在しません。この場合に ID\_TRANSLATION を使用すると、換算データファイル表が存在しないのでエラーコードが返されます。代わりに ID\_TRANSLATION\_FORCE を使用すると、親と子

が同じ通貨を使用している場合に、エラーコードが返される代わりに通常のデータ（タイプ ID\_REGULAR）にアクセスされます。

警告：指定されているデータ種別とエンティティに対して ID\_REGULAR データファイル表が既に選択されている可能性がある場合、ID\_TRANSLATION\_FORCE タイプのデータファイル表は選択しないでください。同様に、ID\_TRANSLATION\_FORCE タイプが既に選択されている可能性がある場合、ID\_REGULAR タイプのデータファイル表は選択しないでください。

これらの連結詳細値を読み取るには、次の例外を除いて [115 ページの「データの取得」](#) の手順に従ってください。

- apiStruct の sigParent フィールドを親エンティティの記号に設定します。
- apiStruct の wType フィールドを [表 35](#) に示すいずれかのコードに設定します。
- 

表 35 wType コード

コード	説明
ID_ADJUSTMENT	親の調整値
ID_CONTRIBUTION	調整後の値
ID_ELIMINATION	消去値
ID_PROPORTIONAL	比率値
ID_TRANSLATION	換算値
ID_TRANSLATION_FORCE	通貨が同じ場合は換算または子

連結詳細値の更新は行わないでください。これらの値は読み取り専用です。これらの値は、連結値の計算方法を示す記録として Hyperion Enterprise により連結中に書き込まれます。

## 組織構造体と高度な関数

[表 36](#) に、組織構造体に関する情報の取得に使用する高度な関数を示します。

表 36 組織に使用する高度な関数

関数	説明
EntEnumOrgEntities( )	組織内のノードを列挙します。
EntFindEntityInOrg( )	組織内のノードを検索します。
EntGetChild( )	子ノードの記号を取得します。
EntGetOrgLevel( )	ノードの組織レベルを取得します。
EntGetSibling( )	組織内の次の兄弟ノードの記号を取得します。
EntIsEntityParent( )	指定されているエンティティが親エンティティであるかどうかを確認します。

これらの関数を呼び出す前に、データ種別表 (ID\_CATEGORY) を選択し、次に組織表 (ID\_ORGANIZATION) とそれ関連付けられているすべての表を選択する必要があります。連動組織を使用するように設定されているアプリケーションの場合は、組織表の選択時にデータ種別記号をキーとして渡す必要があります。表の選択については、[111 ページの「高度な関数を使用した表の選択と選択解除」](#)を参照してください。

Hyperion Enterprise では、組織構造体に関するほとんどの情報がノード表 (ID\_NODES) に保存されています。ノード表内の各レコードは、ノードと呼ばれます。各ノードには、組織構造体内の特定のエンティティに関する情報が含まれています。この情報により、組織構造体内の親、直属の子または同じレベルの兄弟の検索が可能になります。

上に一覧表示されている関数のほとんどは、エンティティ記号ではなく、ノード表内のノード記号を返します。エンティティ記号を取得するには、EntQueryEx() 関数とクエリ属性 NAMESIG を使用してノード表のクエリを実行します。

組織内の最上位ノードの記号を取得するには、組織表 (ID\_ORGANIZATION) とクエリ属性 TOPNODE を使用して EntQueryEx() を呼び出します。

アプリケーションが連動組織を使用するように設定されている場合、組織構造体はデータ種別と期間によって異なります。この場合は、上の表に示す関数の代わりに EntEnum()、EntFind() および EntQueryEx() を使用して、組織構造体に関する情報を取得します。

データ種別 (sigCat フィールド) と期間 (lStartPeriod および lEndPeriod フィールド) を指定する apiStruct を設定します。lEndPeriod フィールドを lStartPeriod と同じ値に設定します。この apiStruct を EntEnum()、EntFind()、および EntQueryEx() に渡して、この組織構造体に関する情報を取得します。apiStruct については、[110 ページの「apiStruct 構造体の使用」](#)を参照してください。

データ種別と期間によって異なる組織用に作成されたコードは、不変の組織でも使用できます。API では不要な情報は無視されます。

## データの視点の設定

ほとんどの API 関数を呼び出してタスクを実行する前に、データの視点を設定する必要があります。現在の組織、データ種別およびエンティティを設定します。次にその例を示します。

```
wRet = EntUpdateDefault(hApp, ID_USERDEFAULT, CURORSIG, sigOrg);  
wRet = EntUpdateDefault(hApp, ID_USERDEFAULT,  
    CURCATEGORYSIG, sigCat);  
wRet = EntUpdateDefault(hApp, ID_USERDEFAULT,  
    CURNAMESIG, sigEntity);
```

## サーバのタスク

Hyperion Enterprise アプリケーションサーバを使用して、クライアントワークステーションではなくアプリケーションサーバで実行できるプロセスがいくつかあ

ります。アプリケーションが Hyperion Enterprise アプリケーションサーバを使用している場合にサーバで実行できる関数を表 37 に示します。

**表 37** サーバベースの関数

サーバベースプロセス	関数	詳細情報
連結	EntConsolidate( )	<ul style="list-style-type: none"> <li>● 143 ページの「EntConsolidate( ) - 連結」</li> </ul>
データの読込	EntUNCDataLoad( ) EntUNCDataLoadVB( )	<ul style="list-style-type: none"> <li>● 270 ページの「EntUNCDataLoad( ) - データの読み込み」</li> <li>● 272 ページの「EntUNCDataLoadVB( ) - データの読み込み」</li> </ul>
データの抽出	EntDataExtract( ) EntDataExtractVB2( ) EntDSMDataExtract( ) EntDSMDataExtractVB( )	<ul style="list-style-type: none"> <li>● 149 ページの「EntDataExtract( ) - データの抽出」</li> <li>● 151 ページの「EntDataExtractVB2( ) - データの抽出」</li> <li>● 159 ページの「EntDSMDataExtract( ) - DSM データの抽出」</li> <li>● 162 ページの「EntDSMDataExtractVB( ) - DSM データの抽出」</li> </ul>
計算式の読み込み	EntLogicImport( ) EntLogicImportVB( )	<ul style="list-style-type: none"> <li>● 233 ページの「EntLogicImport( ) - 計算式のインポート」</li> <li>● 234 ページの「EntLogicImportVB( ) - 計算式のインポート」</li> </ul>
計算式の抽出	EntLogicExport( ) EntLogicExportVB( )	<ul style="list-style-type: none"> <li>● 228 ページの「EntLogicExport( ) - 計算式のエクスポート」</li> <li>● 230 ページの「EntLogicExportVB( ) - 計算式のエクスポート」</li> </ul>
仕訳帳の読み込み	EntJournalLoad( ) EntJournalLoadVB( )	<ul style="list-style-type: none"> <li>● 220 ページの「EntJournalLoad( ) - 仕訳帳の読み込み」</li> <li>● 222 ページの「EntJournalLoad( ) - 仕訳帳の読み込み」</li> </ul>
仕訳帳の抽出	EntJournalExtract( ) EntJournalExtractVB( )	<ul style="list-style-type: none"> <li>● 213 ページの「EntJournalExtract( ) - 仕訳帳の抽出」</li> </ul>

サーバベースプロセス	関数	詳細情報
		<ul style="list-style-type: none"> <li>● 216 ページの「EntJournalExtractVB( ) - 仕訳帳の抽出」</li> </ul>
アプリケーションの読込	EntAppLoad( ) EntAppLoadVB( )	<ul style="list-style-type: none"> <li>● 134 ページの「EntAppLoad( ) - アプリケーションの読み込み」</li> <li>● 135 ページの「EntAppLoadVB( ) - アプリケーションの読み込み」</li> </ul>
アプリケーションの抽出	EntAppExtract( ) EntAppExtractVB( )	<ul style="list-style-type: none"> <li>● 130 ページの「EntAppExtract( ) - アプリケーションの抽出」</li> <li>● 132 ページの「EntAppExtractVB( ) - アプリケーションの抽出」</li> </ul>
セキュリティの読み込み	EntSecurityLoad( ) EntSecurityLoadVB( )	<ul style="list-style-type: none"> <li>● 255 ページの「EntSecurityLoad( ) - セキュリティの読み込み」</li> <li>● 256 ページの「EntSecurityLoadVB( ) - セキュリティの読み込み」</li> </ul>
セキュリティの抽出	EntSecurityExtract( ) EntSecurityExtractVB( )	<ul style="list-style-type: none"> <li>● 252 ページの「EntSecurityExtract( ) - セキュリティの抽出」</li> <li>● 253 ページの「EntSecurityExtractVB( ) - セキュリティの抽出」</li> </ul>
データの保護	EntEntityListLock( )	<ul style="list-style-type: none"> <li>● 167 ページの「EntEntityListLock( ) - データの保護」</li> </ul>
データの保護解除	EntEntityListUnlock( )	<ul style="list-style-type: none"> <li>● 170 ページの「EntEntityListUnlock( ) - データの保護解除」</li> </ul>
株式の抽出	EntSharesExtract( )	<ul style="list-style-type: none"> <li>● 264 ページの「EntSharesExtract( ) - 株式の抽出」</li> <li>● 266 ページの「EntSharesExtractVB( ) - 株式の抽出」</li> </ul>
株式の読み込み	EntSharesLoad( ) EntSharesLoadVB( )	<ul style="list-style-type: none"> <li>● 268 ページの「EntSharesLoad( ) - 株式の読み込み」</li> <li>● 269 ページの「EntSharesLoadVB( ) - 株式の読み込み」</li> </ul>

サーバベースプロセス	関数	詳細情報
ロジックのコンパイル	EntLogicCompile( )	<ul style="list-style-type: none"> <li>● <a href="#">226 ページの「EntLogicCompile( ) - ロジックのコンパイル」</a></li> </ul>

これらの各タスクには、使用するオプションを示すために設定する引数があります。1つのオプションに、サーバ上でのタスクの実行があります。このオプションは通常、....\_REMOTE という定数で定義されます（定数の正確な名前は関数によって異なります）。

関数を呼び出す前に、データの視点を必ず設定してください。詳しくは、[121 ページの「データの視点の設定」](#)を参照してください。

これらの各関数では、コールバック関数（API が呼び出す関数の指定に使用）を使用して、タスクの実行中にステータス情報をユーザに渡します。ただし、サーバ上でタスクを実行することを選択した場合、コールバック関数は使用されません。コールバック関数のアドレスには NULL を渡す必要があります。API ではクライアントコンピュータで実行されている Hcommgr.exe のインスタンスを作成し、サーバ上で実行されているタスクに関するステータス情報を表示します。

サーバでタスク（連結、データの読み込み、データの抽出など）を実行するときは、タスク用に hcommgr.exe のインスタンスのハンドルを取得できます。Hcommgr.exe では、クライアントコンピュータにステータスウィンドウを表示します。ハンドルを取得するには、API 関数がハンドルを配置する変数のアドレスを渡す必要があります。これを Visual Basicで行うには、関数に応じて API 関数の宣言を変更するか、宣言のクローンを作成し、ByVal の代わりに引数 ByRef を渡すように宣言を変更する必要があります。例えば、EntConsolidate() を呼び出すときは、iParam 引数をハンドルの変数のアドレスに設定します。これを行うには、EntConsolidate() の宣言を変更し、ByVal の代わりに引数 ByRef を iParam に渡します。

ハンドルが用意できたら、Microsoft 関数 WaitForSingleObject() を使用して、hcommgr.exe のこのインスタンスが終了するのを待機します。次に、Microsoft 関数 GetExitCodeProcess() を使用して、hcommgr.exe の終了時に返された終了コードを取得します。

HComMgr.exe は、次の終了コードを返します。

0 - 正常完了

ゼロ以外 - エラー 6000 - サーバエラー（接続の問題）、6004 - キャンセル、6005 - エラーで完了（アプリケーションログ参照）

**注：** hcommgr.exe のこのインスタンスが終了しても、タスクがサーバ上でまだ実行されている場合があります。これは、ハンドルがクライアントコンピュータ上のプロセス用であって、サーバでタスクを実行しているプロセス用ではないためです。例えば、接続が切断された場合、タスクはサーバ上で引き続き実行されますが、クライアントはステータス情報を取得しなくなります（終了コード 6000 - サーバエラー）。また、タスクがサーバ上でまだ実行されている間にユーザがステータスウィンドウを閉じたか、途中で hcommgr.exe を中止したことも考えられます。

サーバでは、タスクに必要な表が選択されます。API で情報をサーバに渡すために特殊な表が必要でない限り、通常のデフォルトの表以外に特殊な表を選択する必要はありません。データの読み込みなどの一部のタスクでは、サーバは特定の表に対する読み取り／書き込みアクセスを必要とします。呼び出し中にはこれらの表は選択しないでください。選択すると、サーバが読み取り／書き込みアクセスを取得できなくなり、タスクの実行に失敗します。

## Hyperion Enterprise の INI ファイル関数

Hyperion Enterprise INI ファイル関数を使用すると、アプリケーションの.INI ファイルからの読み取りや.INI ファイルへの書き込みができます。Hyperion Enterprise の `hypent.ini` ファイル内の値の読み取りと書き込みには、表 38 に示す関数を使用します。

表 38 `Hypent.ini` ファイルに使用する高度な関数

関数	説明
<code>EntGetProfileLong</code>	整数を読み取ります。
<code>EntGetProfileString</code>	テキスト値を読み取ります。
<code>EntWriteProfileLong</code>	整数を書き込みます。
<code>EntWriteProfileString</code>	テキスト値を書き込みます。

表 39 に示す関数は、ファイルベースの Hyperion Enterprise アプリケーションの INI ファイル (`appl.ini`) の値の読み取りと書き込みに使用します。

表 39 アプリケーションの INI ファイルで使用する高度な関数

関数	説明
<code>EntGetAppProfileLong</code>	整数を読み取ります。
<code>EntGetAppProfileString</code>	テキスト値を読み取ります。
<code>EntWriteAppProfileLong</code>	整数を書き込みます。
<code>EntWriteAppProfileString</code>	テキスト値を書き込みます。

Hyperion Enterprise SQL アプリケーションにはアプリケーション INI ファイルがないので、SQL およびファイルベースの両方のアプリケーションに対して `EntQueryDefault()` 関数を使用して、Hyperion Enterprise によってファイルベースアプリケーションの.INI ファイルに格納された値を読み取ります。これらの値を更新するには、`EntUpdateDefault()` および `EntSaveDefault()` 関数を使用できます。

## セキュリティ

基本となる Hyperion Enterprise API では、データ種別、エンティティ、勘定科目などの項目にアクセスするユーザの権限をチェックします。ただし、連結、データの読み込み、データの抽出などのタスクを実行するユーザの権限はチェックされ

ません。タスクを実行するユーザの権限のチェックは、プログラマの責任になります。

タスクを実行する現在のユーザの権限をチェックするには、`EntGetRightsToTask()` 関数を呼び出します。

データ種別やエンティティなどの項目にアクセスする現在のユーザの権限をチェックするには、まず適切な表内にある項目（データ種別の場合は `ID_CATEGORY`、エンティティの場合は `ID_NAMES` など）を検索し、`USER_RIGHTS` クエリ属性を使用して `EntQueryEx()` 関数を呼び出します。返される値のバッファは整数（C 言語では `short *`）になっている必要があります。

詳しくは、次の項目を参照してください。

- [189 ページの「EntFind\(\)と ID\\_SECGRPTAB（セキュリティグループ表）」](#)
- [189 ページの「EntFind\(\)と ID\\_SECRIGHTS（セキュリティ権限表）」](#)
- [180 ページの「EntEnum\(\)と ID\\_SECTASK（セキュリティタスク表）」](#)
- [180 ページの「EntEnum\(\)と ID\\_SECUSERTAB（セキュリティユーザ表）」](#)
- [179 ページの「EntEnum\(\)と ID\\_SECGRPTAB（セキュリティグループ表）」](#)
- [180 ページの「EntEnum\(\)と ID\\_SECRIGHTS（セキュリティ権表）」](#)
- [344 ページの「デフォルトのクエリ属性」](#)
- [394 ページの「ID\\_SECCLASS（セキュリティクラス表）クエリ属性」](#)
- [394 ページの「ID\\_SECGRPTAB（セキュリティグループ表）クエリ属性」](#)
- [394 ページの「ID\\_SECRIGHTS（セキュリティ権表）クエリ属性」](#)
- [395 ページの「ID\\_SECTASK（セキュリティタスク表）クエリ属性」](#)
- [395 ページの「ID\\_SECTASKFILTER（セキュリティタスクフィルタ表）クエリ属性」](#)
- [395 ページの「ID\\_SECUSERTAB（セキュリティユーザ表）クエリ属性」](#)

## Hyperion Enterprise のセキュリティを保護するための API 関数

Hyperion Enterprise のセキュリティ機能では、その他すべての表に使用される一般的な API 関数を使用します。特殊な関数は一切ありません。ただし、特定のタスクに対する現在のユーザの権限を取得するために、`EntGetRightsToTask()` という関数が用意されています。セキュリティ情報に使用する、`EntGetRightsToTask()` 以外の主な API 関数に `EntQueryEx()` があります。Hyperion Enterprise には、任意の項目（データ種別、エンティティなど）に対するセキュリティクラスまたは現在のユーザの権限を取得するためのクエリ属性が用意されています。詳しくは、[344 ページの「デフォルトのクエリ属性」](#)を参照してください。Hyperion Enterprise 5 には、Hyperion Enterprise SE と大きく異なるセキュリティ表または新規のセキュリティ表が多く追加されていますが、これらにはその他すべての表に使用するものと同じ API 関数を使用してアクセスできます。さまざまなセキュリティ表の列挙のサブ項目については、[171 ページの「EntEnum\(\) - 表内のレコードの列挙」](#)を参照してください。

# 使用されなくなった Ent...( )関数から新しい Ent...( )関数への置き換え

表 40 に、この Hyperion Enterprise リリースの新しい Ent...( )関数に置き換えられている Ent...( )関数をすべて示します。古い関数を継続して使用することは可能ですが、これらの関数についてはこのガイドでは触れていません。

表 40 使用されなくなった Hyperion Enterprise Ent...( )関数および対応する新しい Ent...( )関数

古い関数	新しい関数
EntDataExtractVB( )	EntDataExtractVB2( )
EntDataLoad( )	EntUNCDataLoad( )
EntDataLoadVB( )	EntUNCDataLoadVB( )
EntFormatNumber( )	EntFormatNumber2( )
EntQuery( )	EntQueryEx( )
EntQueryStr( )	EntQueryExStr( )

## アルファベット順の高度な関数のリファレンス

ここでは、すべての高度な関数の一覧をアルファベット順に示します。

次の項目は、ほとんどの高度な関数に関する一般的な注意事項です。

- 関数にデフォルトの表を選択する必要がある場合は、ID\_CATEGORY、ID\_ORGANIZATION および ID\_ACCOUNTS とそれに関連付けられている表を選択する必要があります。詳しくは、[111 ページの「高度な関数を使用した表の選択と選択解除」](#)を参照してください。
- select ID\_APPDEFAULT を選択する必要はありません。これはアプリケーションハンドル（hApp 引数）が存在するときに常に利用できるためです。詳しくは、[338 ページの「デフォルト設定の表」](#)を参照してください。
- 関数に指定されている表を選択するには、APILOCK\_READWRITE のみを使用します。
- ID\_SECCLASS を選択する場合、EntSelect()は使用しないでください。代わりに EntSelectAdd()を使用します。EntSelect()は、最初に選択されている表のみに使用します。

### EntAcctAsk( ) - 勘定科目の選択

この関数は、Hyperion Enterprise の「勘定科目の選択」ダイアログボックスを呼び出して、選択された勘定科目を取得します。「キャンセル」を選択した場合、勘定科目は取得されません。

次の形式を使用します。

Declare Function EntAcctAsk Lib "heaccess.dll" Alias "\_EntAcctAsk@8" (ByVal hSelect As Long, ByVal szRetbuf As String) As Integer

#### 変数 説明

hSelect 選択された表のハンドル

szRetbuf 勘定科目 ID を返すためのバッファ。バッファのサイズは、少なくとも HYP\_SIZEFULLACCT+1（C 言語では SIZEFULLACCT+1）文字にする必要があります。

戻りコード：

コード	説明
0	成功
NONE	エラー発生

例：

```
szAcct$ = Space(HYP_SIZEFULLACCT+1)
ret% = EntAcctAsk(hSelect&, szAcct$)
If ret% = 0 Then szAcct$ = CToBStr(szAcct$)
```

C 言語では次の形式を使用します。

```
short WINAPI EntAcctAsk( HSELECT hSelect, LPSTR szRetbuf )
```

## EntAcctListAsk( ) - 勘定科目一覧の選択

この関数は、勘定科目一覧の選択に使用するダイアログボックスを表示します。EntAcctListAsk()は、選択された勘定科目一覧の名前を戻りバッファに戻します。または、[キャンセル] が選択された場合は、空の文字列を返します。この関数によって、勘定科目一覧表（ID\_ACCTLIST）が一時的に選択され、選択解除されます。

次の形式を使用します。

Declare Function EntAcctListAsk Lib "heaccess.dll" Alias "\_EntAcctListAsk@8" (ByVal hSelect As Long, ByVal szRetbuf As String) As Integer

#### 変数 説明

hSelect 選択された表のハンドル

szRetbuf 選択された勘定科目一覧を返すバッファ。このバッファのサイズは少なくとも HYP\_SIZELABEL+1（C 言語の場合は SIZENAME+1）文字にする必要があります。

戻りコード：

コード	説明
0	成功

コード	説明
NONE	エラー発生

例：

```
szAcctList$ = Space (HYP_SIZELABEL+1)
ret% = EntAcctAskListAsk(hSelect&, szAcctList$)
If ret% = 0 Then szAcctList$ = CToBStr(szAcctList$)
```

C 言語では次の形式を使用します。

```
short WINAPI EntAcctListAsk(HSELECT hSelect, LPSTR szRetbuf)
```

## EntAcctSplit( ) - 勘定科目記号の分割

この関数は、指定された勘定科目記号の主要勘定科目とサブ勘定科目の記号を返します。

次の形式を使用します。

```
Declare Function EntAcctSplit Lib "heaccess.dll" Alias "_EntAcctSplit@20" (ByVal hSelect
As Long, ByVal sigAcct As Long, sigMajor As Long, sigSub As Long, sigSubSub As Log) As
Integer
```

### 変数 説明

hSelect	選択された表のハンドル
sigAcct	主要勘定科目記号とサブ勘定科目記号の取得対象となる勘定科目記号
sigMajor	主要勘定科目記号を返すバッファ
sigSub	第 1 レベルサブ勘定科目記号を返すバッファ
sigSubSub	第 2 レベルサブ勘定科目記号を返すバッファ

戻りコード：

コード	説明
0	成功
NONE	エラー発生

例：

```
ret% = EntAcctSplit(hselect&, sigAcctIn&, sigMajor&,
sigSub1&, sigSub2&)
If ret% = 0 Then
Msg$ = "Major: " & sigMajor& & " sub1: " & sigSub1& & "sub2: " &
sigSub2&
Else
msg$ = "Error "
End If
ret2% = MsgBox(msg$)
```

C 言語では次の形式を使用します。

```
short WINAPI EntAcctSplit(HSELECT hSelect, SIGNA sigAcct, SIGNA * psigMajor,
SIGNA * psigSub, SIGNA * psigSubSub);
```

## EntAppendToCBChain( ) - コールバックチェーンへの追加

この関数は、コールバック関数を、hSelect 引数変数に関連付けられているコールバック関数のチェーンに追加します。このコールバック関数は、hSelect ブロック内の表が変更された場合に呼び出されます。詳しくは、[112 ページの「選択のコールバック」](#)を参照してください。

次の形式を使用します。

```
Declare Function EntAppendToCBChain Lib "heaccess.dll" Alias
"_EntAppendToCBChain@12" (ByVal hSelect As Long, ByVal SelCallback As Long, ByVal
lParam As Long) As Integer
```

変数	説明
----	----

hSelect	選択された表のハンドル
---------	-------------

SelCallback	コールバック関数のアドレス。詳しくは、 <a href="#">328 ページの「CALLBACKSEL」</a> を参照してください。
-------------	--

lParam	SelCallback 関数の引数
--------	-------------------

戻りコード：

コード	説明
0	成功
NONE	エラー発生

C 言語では次の形式を使用します。

```
short WINAPI EntAppendToCBChain(HSELECT hSelect, CALLBACKSEL SelCallback,
LONG lParam);
```

## EntAppExtract( ) - アプリケーションの抽出

この関数は、アプリケーションを ASCII ファイルに抽出します。この関数を呼び出す前に、表を選択しないでください。

ステータス情報用のコールバック関数。pArgs->fnStatusCB は、EntDataExtract( )に使用するコールバック関数のタイプと同じです。詳しくは、[322 ページの「CALLBACKDBLOAD - EntAppExtract\( \) の場合」](#)を参照してください。ただし、コールバック関数の szCategory 引数は、データ種別名ではなく、ステータスメッセージに使用されます。szEntity、iStart、iEnd、iTot、および iCur 引数は使用されません。

**注：** Visual Basic プログラマは、EntAppExtract( )の代わりに EntAppExtractVB( )を使用してください。

C 言語では次の形式を使用します。

**short WINAPI EntAppExtract(HAPP hApp, LPAPPLOADSTRUCT pArgs)**

変数	説明
hApp	アプリケーションハンドル
pArgs	アプリケーション抽出用の引数を含んでいる構造体
pArgs->dwSize	この構造体のサイズ
pArgs->cDelim	フィールドの区切り文字で、通常はカンマ (,)
pArgs->szFile	抽出ファイルのパスとファイル名
pArgs->dwOptions	オプションのビットフラグ。詳しくは、TOOLINC.H ファイルの APPL_...を参照してください。
pArgs->dwMoreOptions	0。将来使用するために予約済み。
pArgs->fnStatusCB	ステータスのコールバック関数または NULL。これは、サーバ上で実行していない場合にのみ使用します。
pArgs->lParamStatus	fnStatusCB の引数
pArgs->fnSelectCB	今後使用予定
pArgs->lParamSelect	今後使用予定
pArgs->hProcess	[タスクステータス] ウィンドウプロセスのハンドル。これは、サーバ上での実行時にプロセスハンドルを返します。
pArgs->bLoad	抽出されたものがある場合は TRUE、それ以外の場合は FALSE を返します。

戻りコード：

コード	説明
0	成功
LOAD_CHECK_ERRORLOG	エラーが発生し、そのエラーがエラーログに示されています。
NONE	エラー発生。具体的なエラーコードを取得するには、EntGetLastErrorByHApp( )関数を呼び出します。

[132 ページの「EntAppExtractVB\( \) - アプリケーションの抽出」](#)の例を参照してください。

## EntAppExtractVB( ) - アプリケーションの抽出

Visual Basic プログラマは、EntAppExtract( )の代わりにこの関数を使用してください。この関数は、アプリケーションを ASCII ファイルに抽出します。この関数を呼び出す前に、表を選択しないでください。

ステータス情報用のコールバック関数。pArgs.fnStatusCB は、EntDataExtract( )に使用するコールバック関数のタイプと同じです。詳しくは、[322 ページの「CALLBACKDBLOAD - EntAppExtract\( \) の場合」](#)を参照してください。ただし、コールバック関数の szCategory 引数は、データ種別名ではなく、ステータスメッセージに使用されます。szEntity、iStart、iEnd、iTot、および iCur 引数は使用されません。

次の形式を使用します。

```
Declare Function EntAppExtractVB Lib "heaccess.dll" Alias "_EntAppExtractVB@12"  
(ByVal hApp As Long, pArgs As APPLOADSTRUCT, ByVal szExtractFile As String) As  
Integer
```

変数	説明
hApp	アプリケーションハンドル
pArgs	アプリケーション抽出用の引数を含んでいる構造体
pArgs.dwSize	この構造体のサイズ
pArgs.cDelim	フィールドの区切り文字で、通常はカンマ (,)
pArgs.dwOptions	オプションのビットフラグ。詳しくは、TOOLINC.H ファイルの APPL_...を参照してください。
pArgs.dwMoreOptions	0。将来使用するために予約済み。
pArgs.fnStatusCB	ステータスのコールバック関数または NULL。これは、サーバ上で実行していない場合にのみ使用します。
pArgs.lParamStatus	fnStatusCB の引数
pArgs.fnSelectCB	将来使用するために予約済み
pArgs.lParamSelect	将来使用するために予約済み
pArgs.hProcess	[タスクステータス] ウィンドウプロセスのハンドル。これは、サーバ上での実行時にプロセスハンドルを返します。
pArgs.bLoad	抽出されたものがある場合は TRUE、それ以外の場合は FALSE を返します。
szExtractFile	抽出ファイルのパスとファイル名

戻りコード：

コード	説明
0	成功

コード	説明
LOAD_CHECK_ERRORLOG	エラーが発生し、そのエラーがエラーログに示されています。
NONE	エラー発生。具体的なエラーコードを取得するには、EntGetLastErrorByHApp( ) 関数を呼び出します。

**注：** この例の EntUnSelect0( )は、EntUnSelect( )と同じように宣言されますが、apiStruct の代わりに 0 を渡すことが許可されます。詳しくは、[16 ページの「Visual Basic プログラミング上の注意」](#)を参照してください。

例：

```
Private Sub ApplicationExtract(hSelect As Long,szFile As String)
Dim s As APPLDSTRUCT
Dim wRet%
Dim hApp&

'structを設定
s.dwSize = LenB(s)
s.cDelim = Asc("!")
s.szFile = 0
s.dwOptions = SetOptionFlags()
s.fnStatusCB = FuncPtrToLong(AddressOf MyCallback)
s.lParamStatus = 0
s.fnSelectCB = 0

hApp& = EntGetHappFromSelect(hSelect&)
'すべての表の選択を解除
hSelect& = EntUnSelect0(hSelect&,HYP_NONE,HYP_NONE,0,wRet%,0)

wRet% = EntAppExtractVB(hApp&, s, szFile$)
End Sub

'コールバック関数例
' 入力:addressOfFileNameCString = 抽出
ファイル名 (Cスタイル文字列) ;
' addressOfMessageCString = ステータス
メッセージ (Cスタイル文字列) ;
' lParam
' 出力: 0を返します
'
' =====
Public Function MyCallback(ByVal

addressOfFileNameCString As Long, ByVal addressOfMessageCString As Long,
ByVal unused1 As Long, ByVal unused2 As Long,ByVal unused3 As Long,
ByVal unused4 As Long, ByVal unused5 As Long, ByVal lParam As Long) As
Integer

Dim szFileName$, szMessage$
Dim wRet%, wLen%

On Error Resume Next      'Microsoftが勧めるコールバック関数;
```

```

'addressOfFileNameCStringからszFileName$を取得
wLen% = EntVBGetCStrLen (addressOfFileNameCString)
szFileName$ = Space(wLen%)
wRet% = EntVBCopyStr (szFileName$, addressOfFileNameCString,
    wLen%)

'addressOfMessageCStringからszMessage$を取得
wLen% = EntVBGetCStrLen (addressOfMessageCString)
szMessage$ = Space(wLen%)
wRet% = EntVBCopyStr (szMessage$, addressOfMessageCString, wLen%)

'メッセージを表示
szMessage$ = "File: [" & szFileName$ & "]" & Chr(13) & Chr(13)
    & "[" & szMessage$ & "]"
wRet% = MsgBox(szMessage$, vbOKOnly, "Now Extracting...")

MyCallback = 0      '0を返す
End Function

'AddressOf()の結果を (fnPtrとして) この関数に渡します。
'この関数はその結果をlongに変換するので、
As Anyではなくlongとして宣言されている'フィールドまたは引数に割り当てることができま
す。
Public Function FuncPtrToLong(ByVal fnPtr As Long) As Long

FuncPtrToLong = fnPtr

End Function

```

C 言語では次の形式を使用します。

```

short DllExport WINAPI EntAppExtractVB(HAPP hApp, LPAPPLOADSTRUCT pArgs,
LPCSTR szExtractFile);

```

## EntAppLoad( ) - アプリケーションの読み込み

この関数は、アプリケーションを ASCII ファイルから読み込みます。この関数を呼び出す前に、表を選択しないでください。

ステータス情報用のコールバック関数。pArgs->fnStatusCB は、EntDataLoad( )に使用するコールバック関数のタイプと同じです。詳しくは、[323 ページの「CALLBACKDBLOAD - EntAppLoad\( \)の場合」](#)を参照してください。ただし、コールバック関数の szCategory 引数は、データ種別名ではなく、ステータスメッセージに使用されます。szEntity、iStart、iEnd 引数は使用されません。進行状況バーに役立つ iTot および iCur 引数は、勘定科目の数ではなく、読み込みファイル内の合計バイト数（合計バイト数とそれまでに処理されたバイト数）を参照します。pArgs->fnSelectCB コールバック関数は、データ種別表への変更の保存時に通知を受ける必要がある場合に使用します。詳しくは、[328 ページの「CALLBACKSEL」](#)を参照してください。

**注：** Visual Basic プログラマは、EntAppLoad( )の代わりに EntAppLoadVB( )を使用してください。

C 言語では次の形式を使用します。

short WINAPI EntAppLoad(HAPP hApp, LPAPPLOADSTRUCT pArgs)

変数	説明
hApp	アプリケーションハンドル
pArgs	アプリケーション抽出用の引数を含んでいる構造体
pArgs->dwSize	この構造体のサイズ
pArgs->cDelim	フィールドの区切り文字で、通常はカンマ (,)
pArgs->szFile	読み込みファイルのパスとファイル名
pArgs->dwOptions	オプションのビットフラグ。詳しくは、TOOLINC.H ファイルの APPL_... を参照してください。
pArgs->dwMoreOptions	0。将来使用するために予約済み。
pArgs->fnStatusCB	ステータスのコールバック関数または NULL。これは、サーバ上で実行していない場合にのみ使用します。
pArgs->lParamStatus	fnStatusCB の引数
pArgs->fnSelectCB	データ種別表の変更時に呼び出す関数、または NULL。この引数はローカルでの実行時にのみ使用します（サーバでの実行時には使用しません）。
pArgs->lParamSelect	fnSelectCB の引数
pArgs->hProcess	[タスクステータス] ウィンドウプロセスのハンドル。これは、サーバ上での実行時にプロセスハンドルを返します。
pArgs->bLoad	読み込まれたものがある場合は TRUE、それ以外の場合は FALSE を返します。

戻りコード：

コード	説明
0	成功
LOAD_CHECK_ERRORLOG	エラーが発生し、そのエラーがエラーログに示されています。
NONE	エラー発生。具体的なエラーコードを取得するには、EntGetLastErrorByHApp() 関数を呼び出します。

## EntAppLoadVB( ) - アプリケーションの読み込み

Visual Basic プログラマは、EntAppLoad( )の代わりにこの関数を使用してください。この関数は、アプリケーションを ASCII ファイルから読み込みます。この関数を呼び出す前に、表を選択しないでください。

ステータス情報用のコールバック関数。pArgs.fnStatusCB は、EntDataLoad( )に使用するコールバック関数のタイプと同じです。詳しくは、[323 ページの「CALLBACKDBLOAD - EntAppLoad\( \)の場合」](#)を参照してください。ただし、コールバック関数の szCategory 引数は、データ種別名ではなく、ステータスメッセージ

ジに使用されます。szEntity、iStart、iEnd 引数は使用されません。進行状況バーに役立つ iTot および iCur 引数は、勘定科目の数ではなく、読み込みファイル内の合計バイト数（合計バイト数とそれまでに処理されたバイト数）を参照します。pArgs.fnSelectCB コールバック関数は、データ種別表への変更の保存時に通知を受ける必要がある場合に使用します。詳しくは、[328 ページの「CALLBACKSEL」](#)を参照してください。

次の形式を使用します。

**Declare Function EntAppLoadVB Lib "heaccess.dll" Alias "\_EntAppLoadVB@12" (ByVal hApp As Long, pArgs As APploadSTRUCT, ByVal szLoadFile As String) As Integer**

変数	説明
hApp	アプリケーションハンドル
pArgs	アプリケーション抽出用の引数を含んでいる構造体
pArgs.dwSize	この構造体のサイズ
pArgs.cDelim	フィールドの区切り文字で、通常はカンマ (,)
pArgs.szFile	NULL。代わりに szLoadFile を使用します。
pArgs.dwOptions	オプションのビットフラグ。詳しくは、TOOLINC.H ファイルの APPL_...を参照してください。
pArgs.dwMoreOptions	0。将来使用するために予約済み。
pArgs.fnStatusCB	ステータスのコールバック関数または NULL。これは、サーバ上で実行していない場合にのみ使用します。
pArgs.lParamStatus	fnStatusCB の引数
pArgs.fnSelectCB	データ種別表の変更時に呼び出す関数、または NULL。この引数はローカルでの実行時にのみ使用します。
pArgs.lParamSelect	fnSelectCB の引数
pArgs.hProcess	[タスクステータス] ウィンドウプロセスのハンドル。これは、サーバ上での実行時にプロセスハンドルを返します。
pArgs.bLoad	読み込まれたものがある場合は TRUE、それ以外の場合は FALSE を返します。
szLoadFile	読み込むファイルのパス名

戻りコード：

コード	説明
0	成功
LOAD_CHECK_ERRORLOG	エラーが発生し、そのエラーがエラーログに示されています。
NONE	エラー発生。具体的なエラーコードを取得するには、EntGetLastErrorByHApp() 関数を呼び出します。

C 言語では次の形式を使用します。

```
short WINAPI EntAppLoadVB(HAPP hApp, LPAPPLOADSTRUCT pArgs, LPCSTR szLoadFile);
```

例：

**注：** この例の EntUnSelect0() は、EntUnSelect() と同じように宣言されますが、apiStruct の代わりに 0 を渡すことが許可されます。詳しくは、[16 ページの「Visual Basic プログラミング上の注意」](#)を参照してください。

```
Private Sub ApplicationLoad(hSelect As Long, szFile As String)
Dim s As APPLOADSTRUCT
Dim wRet%
Dim hApp&

'structを設定
s.dwSize = LenB(s)
s.cDelim = Asc("!")
s.szFile = 0
s.dwOptions = SetOptionFlags()
s.fnStatusCB = FuncPtrToLong(AddressOf MyCallback)
s.lParamStatus = 0
s.fnSelectCB = 0

hApp& = EntGetHappFromSelect (hSelect&)
'すべての表の選択を解除
hSelect& = EntUnSelect0 (hSelect&, HYP_NONE, HYP_NONE, 0, wRet%, 0)

wRet% = EntAppLoadVB (hApp&, s, szFile$)

End Sub

'コールバック関数例
' 入力: addressOfFileNameCString = 読み込み
filename (C-style string);
' addressOfMessageCString = ステータスメッセージ (Cスタイル文字列) ;
' lTotalSize = 読み込みファイルの合計バイト数
' lCurPos = 読み込みファイル内の現在位置
' lParam
' 出力: 0を返します
'
' =====
Public Function MyCallback(ByVal
addressOfFileNameCString As Long,

ByVal addressOfMessageCString As Long, ByVal
unused1 As Long, ByVal unused2 As Long, ByVal unused3 As Long, ByVal
lTotalSize As Long, ByVal lCurPos As Long, ByVal lParam As Long) As
Integer

Dim szFileName$, szMessage$
Dim wRet%, wLen%

On Error Resume Next      'Microsoftが勧めるコールバック関数;

'addressOfFileNameCStringからszFileName$を取得
wLen% = EntVBGetCStrLen (addressOfFileNameCString)
```

```

szFileName$ = Space(wLen%)
wRet% = EntVBCopyStr(szFileName$, addressOfFileNameCString,
    wLen%)

'addressOfMessageCStringからszMessage$を取得
wLen% = EntVBGetCStrLen (addressOfMessageCString)
szMessage$ = Space(wLen%)
wRet% = EntVBCopyStr(szMessage$, addressOfMessageCString, wLen%)

'メッセージを表示
szMessage$ = "File: [" & szFileName$ & "]"

& Chr(13) & Chr(13)
& "[" & szMessage$ & "]"

szMessage$ = szMessage$ & Chr(13) & Chr(13)

& CStr(lCurPos&)
& " of " & CStr(lTotalSize&)
& " bytes."

wRet% = MsgBox(szMessage$, vbOKOnly, "読み込み中")

MyCallback = 0      '0を返す

End Function
'APPLLOADSTRUCT.dwOptionsにビットフラグを設定
'入力: フォーム上のチェックボックス
'出力: APPLLOADSTRUCT.dwOptionsに使用する値を返す
Private Function SetOptionFlags() As Long
Dim dwOptions&

dwOptions& = 0
If chkEntities.Value Then 'エンティティ、組織および通貨を処理

dwOptions& = dwOptions& Or APPL_ORGANIZATION
dwOptions& = dwOptions& Or APPL_ENTITY Or APPL_ENTITYLIST
dwOptions& = dwOptions& Or APPL_ENTITYCONVERSION Or
    APPL_SUBENTITY
dwOptions& = dwOptions& Or APPL_ENTOWNERSHIP Or
    APPL_SUBSTRUCTURE
dwOptions& = dwOptions& Or APPL_INTERCODEPEND
dwOptions& = dwOptions& Or APPL_CURRENCY
dwOptions& = dwOptions& Or APPL_ENTITYCODE
dwOptions& = dwOptions& Or APPL_SUBENTOWNERSHIP

End If
If chkAccts.Value Then '勘定科目を処理

dwOptions& = dwOptions& Or APPL_ACCOUNT Or APPL_ACCOUNTLIST
dwOptions& = dwOptions& Or APPL_SUBACCOUNT Or
    APPL_ACCOUNTCONVERSION
dwOptions& = dwOptions& Or APPL_INTERCOACCT

End If
If chkReports.Value Then 'レポートおよびパッケージの処理
    dwOptions& = dwOptions& Or APPL_REPORT Or APPL_BOOK
End If

```

```

If chkCat.Value Then      'データ種別と期別替えの処理
    dwOptions& = dwOptions& Or APPL_CATEGORY Or APPL_ROLLOVER
End If

If chkLogic.Value Then    'ロジック、カスタム関数および更新ルール
                           を処理
    dwOptions& = dwOptions& Or APPL_METHOD
End If

If chkCat.Value Then      'コード表の処理
    dwOptions& = dwOptions& Or APPL_CODE
End If

If chkFormats.Value Then  '書式表の処理
    dwOptions& = dwOptions& Or APPL_FORMAT
End If

If chkSchedules.Value Then 'データ入力表を処理
    dwOptions& = dwOptions& Or APPL_SCHEDULE
End If

If chkAppOptions.Value Then 'アプリケーションオプションと連結ロジック表
                           を処理
    dwOptions& = dwOptions& Or APPL_CONSOLASSIGN
    dwOptions& = dwOptions& Or APPL_APPDEFAULT
End If

SetOptionFlags = dwOptions&

End Function
'AddressOf()の結果を（fnPtrとして）この関数に渡します。
'この関数はその結果をlongに変換するので、
As Anyではなくlongとして宣言されている'フィールドまたは引数に割り当てることができます。
Public Function FuncPtrToLong(ByVal fnPtr As Long) As Long

FuncPtrToLong = fnPtr

End Function

```

## EntAsciiToDouble( ) - string から double への変換

この関数は、文字列を数値に変換します。文字列には、アプリケーション設定で指定されている小数点文字や千の区切り文字を含めることができます。

次の形式を使用します。

```

Declare Function EntAsciiToDouble Lib "heaccess.dll" Alias "_EntAsciiToDouble@12"
(ByVal hApp As Long, ByVal szBuf As String, ByVal pzRest As Long) As Double

```

### 変数 説明

hApp アプリケーションハンドル

szBuf 変換する文字列

pzRest szBuf 内の場所を指すポインタに続いて数値が返されます（この引数はC言語でプログラミングしている場合に役立ちますが、Microsoft Visual Basic には適用されません）。

戻りコード：

コード	説明
数値	成功
0	エラー発生

例：

```
szVal$ = "100.35"  
dVal# = EntAsciiToDouble(hApp&, szVal$, dummyVar&)
```

C 言語では次の形式を使用します。

```
double WINAPI EntAsciiToDouble(HAPP hApp, LPSTR szBuf, LPSTR * pzRest);
```

## EntCatAsk( ) - データ種別の変更

この関数は、[データ種別の選択] ダイアログボックスを呼び出して、選択されたデータ種別を取得します。[キャンセル] を選択した場合、データ種別は取得されません。

次の形式を使用します。

```
Declare Function EntCatAsk Lib "heaccess.dll" Alias "_EntCatAsk@8" (ByVal hSelect As Long, ByVal szRetbuf As String) As Integer
```

### 変数 説明

hSelect 選択された表のハンドル

szRetbuf データ種別名を返すためのバッファ。バッファの長さは、少なくとも HYP\_SIZELABEL+1 (C 言語の場合は SIZECAT+1) 文字にする必要があります。

戻りコード：

コード	説明
0	成功
NONE	無効な hSelect

例：

```
szCat$ = Space (HYP_SIZELABEL+1)  
ret% = EntCatAsk(hSelect&, szCat$)  
If ret% = 0 Then szCat$ = CToStr(szCat$)
```

C 言語では次の形式を使用します。

```
short WINAPI EntCatAsk( HSELECT hSelect, LPSTR szRetbuf);
```

## EntCatGetNumPeriods( ) - データ種別の期間数の計算

この関数は、データ種別に含まれている期間数を任意の期間単位に対して計算します。1 つの期間単位のデータ種別と別の期間単位が与えられると、この関数は

使用可能な期間数を返します。例えば、12 の期間を含む月次データ種別と四半期の期間単位が与えられた場合、EntCatGetNumPeriods( )はデータ種別の期間数として 4（四半期）を返します。

次の形式を使用します。

```
Declare Function EntCatGetNumPeriods Lib "heaccess.dll" Alias
"_EntCatGetNumPeriods@12" (ByVal hSelect As Long, ByVal sigCat As Long, ByVal
sigFreq As Long) As Integer
```

#### 変数 説明

hSelect 選択された表のハンドル

sigCat データ種別記号

sigFreq 期間単位記号

戻りコード：

コード	説明
データ種別 sigCat が期間単位 sigFreq に対して持っている期間数	成功
NONE	エラー発生

例：

```
sigCat& = EntFind(hSelect&, HYP_ID_CATEGORY, HYP_NONE, "ACTUAL",
0)
sigFreq& = FREQ_QUARTER
numPer% = EntCatGetNumPeriods(hSelect&, sigCat&, sigFreq&)
```

C 言語では次の形式を使用します。

```
short WINAPI EntCatGetNumPeriods(HSELECT hSelect, SIGNA sigCat, SIGNA sigFreq);
```

## EntCatGetPerShort( ) - 期間ラベルの取得

データ種別内の任意の期間に対して、この関数は指定された期間単位の期間 ID を返します。

次の形式を使用します。

```
Declare Function EntCatGetPerShort Lib "heaccess.dll" Alias "_EntCatGetPerShort@24"
(ByVal hSelect As Long, ByVal sigCat As Long, ByVal sigPer As Long, ByVal sigFreq As
Long, ByVal cchTag As Integer, ByVal pzPer As String) As Integer
```

#### 変数 説明

hSelect 選択された表のハンドル

sigCat データ種別記号

sigPer データ種別の期間数

## 変数 説明

sigFreq 期間単位記号、またはデータ種別の期間単位を使用する場合は HYP\_NONE（C 言語では NONE）

cchTag pzPer バッファの長さ

pzPer 期間 ID を返すためのバッファ

## 戻りコード

コード	説明
文字列の長さ	成功
0	エラー発生

例：

```
szPer$ = SPACE(10) '任意のサイズ
rc% = EntCatGetPerShort(hSelect&, sigCat&, sigPer&, sigFreq&,
    Len(szPer$), szPer$)
If rc% Then szPer$= CTobStr(szPer$)
```

C 言語では次の形式を使用します。

```
short WINAPI EntCatGetPerShort(HSELECT hSelect, SIGNA sigCat, SIGNA sigPer,
SIGNA sigFreq, short cchTag, char * pzPer);
```

## EntCatMapPeriod( ) - 期間から期間単位へのマッピング

この関数は、1 つの期間単位内の期間を別の期間単位内の対応する期間にマッピングします。

次の形式を使用します。

```
Declare Function EntCatMapPeriod Lib "heaccess.dll" Alias "_EntCatMapPeriod@20"
(ByVal hSelect As Long, ByVal sigCat As Long, ByVal sigFreq As Long, ByVal sigPer As
Long, ByVal sigAltFreq As Long) As Long
```

## 変数 説明

hSelect 選択された表のハンドル

sigCat データ種別記号

sigFreq sigPer の期間単位記号

sigPer 期間単位 sigFreq の期間記号

sigAltFreq 期間 sigPer のマッピング先の代替期間の期間単位記号、またはデータ種別のデフォルトの期間単位を使用する場合は HYP\_NONE（C 言語では NONE）

戻りコード：

コード	説明
期間単位の期間記号	成功
BAD_PERIOD	エラー発生

例：

データ種別内の四半期データの最初の期間が必要で、週次期間数を知る必要があります。この場合は、次の例を使用できます。

```
sigPerWeekly& = EntCatMapPeriod(hSelect&, sigCat&, FREQ_QUARTER, 0, FREQ_WEEK)
```

上の例では、データ種別が1月に月次になっているとすると、Q1は月次にマッピングされ、3月（3）が取得されます。この関数は次に月次（3）を週にマッピングし、W13を取得します。3月の最終週は13週目であるためです。これは、週を基にしてレポートを作成する場合に便利ですが、期間単位が異なっている可能性のある2つのデータ種別から対応する値を取得する必要があります。

C言語では次の形式を使用します。

```
SIGNA WINAPI EntCatMapPeriod(HSELECT hSelect, SIGNA sigCat, SIGNA sigFreq, SIGNA sigPer, SIGNA sigAltFreq);
```

## EntCloseApplication( ) - アプリケーションの終了

この関数は、アプリケーションを終了します。

次の形式を使用します。

```
Declare Sub EntCloseApplication Lib "heaccess.dll" Alias "_EntCloseApplication@4" (ByVal hApp As Long)
```

ここで、hApp はアプリケーションのハンドルです。

例：

```
Public Sub CloseApplication( )
    If hApp& <> 0 Then EntCloseApplication hApp&
        hApp& = 0
    End If
End Sub
```

次の形式を使用します。

```
VOID WINAPI EntCloseApplication(HAPP hApp);
```

## EntConsolidate( ) - 連結

この関数は、連結に使用します。EntConsolidate( )を呼び出す前に、データ種別表、組織表および勘定科目表と、これらに関連付けられている表を選択します。また、PSF表（ID\_PSFDATA）を選択してデータ種別をsigKey引数として渡し、APILOCK\_READWRITEをwAttr引数として指定します。

EntConsolidate()を呼び出してタスクを実行する前に、データの視点を必ず設定してください。現在の組織、データ種別およびエンティティを設定します。詳しくは、[121 ページの「データの視点の設定」](#)を参照してください。

次の形式を使用します。

```
Declare Function EntConsolidate Lib "heaccess.dll" Alias "_EntConsolidate@24" (ByVal hSelect As Long, ByVal cConsType As Integer, ByVal sigTopNode As Long, ByVal ConsolCB As Long, ByVal lParam As Long, apiS As APISTRUCT) As Integer
```

変数	説明
hSelect	選択された表のハンドル
cConsType	次の連結タイプのいずれか  CONSOL_IMPACTED  CONSOL_ALL  CONSOL_ALLWITHDATA  Hyperion Enterprise を実行しており、クライアント／サーバオプションがインストールされている場合は、サーバでの連結に選択した連結タイプにかかわらず、CONSOL_REMOTE 定数に OR 演算を使用できます。
sigTopNode	連結の最上位ノードの記号
ConsolCB	コールバック関数。CONSOL_REMOTE が cConstype に設定されている場合は NULL になります。詳しくは、 <a href="#">319 ページの「CALLBACKAPI」</a> を参照してください。
lParam	ConsolCB コールバック関数の引数。ただし、CONSOL_REMOTE が cConsType に設定されている場合、コールバック関数は使用されません。lParam が NULL 以外の場合は、それがプロセスハンドルのバッファとして想定されます。
apiS	詳細情報を含んでいる apiStruct を指すポインタ。sigCat はデータ種別記号に設定し、lStartPeriod および lEndPeriod は連結する範囲の開始期間と終了期間にそれぞれ設定します。

コールバック関数は、ステータス情報を提供します。sigRecd 引数は、現在処理中の親エンティティの記号です。sigKey 引数は、親エンティティに現在連結中の子エンティティの記号です。apiStruct 引数は、EntConsolidate()に渡した asiStruct と同じですが、lStartPeriod および lEndPeriod フィールドには現在処理中の期間が表示されます。続行するにはコールバック関数が 0 を返し、連結をキャンセルするには 1 を返す必要があります。

戻りコード：

コード	説明
0	成功
NONE	エラー発生

サーバ上で結している場合（CONSOL\_REMOTE フラグが引数変数 cConsType に設定されている場合）は、連結処理のプロセスハンドルを認識し、プログラムによって起動される連結処理を追跡および監視できるようにすると便利です。プロセスハンドルを取得するには、EntConsolidate()を呼び出す前に、引数変数 lParam を

HANDLE 変数に設定します。返されるハンドルは、ローカルクライアントコンピュータで実行されている [タスクステータス] ウィンドウプログラム HCOMMGR.EXE のインスタンスのプロセスハンドルです。詳しくは、[121 ページ](#)の「サーバのタスク」を参照してください。

例：

```
Dim lpAPIStruct As apiStruct
'現在のPOVを設定
wRet% = EntUpdateDefault(hApp%, HYP_ID_USERDEFAULT,
    HYP_CURCATEGORYSIG, sigCat%)
wRet% = EntUpdateDefault(hApp%, HYP_ID_USERDEFAULT,
    HYP_CURORGSIG, sigOrg%)
wRet% = EntUpdateDefault(hApp%, HYP_ID_USERDEFAULT,
    HYP_CURNAMESIG, sigEntity%)
'注：アプリケーションが期間ごとの組織である場合は、表ID_ORGANIZATIONおよびそれに関
連付けられている表を選択したときに、
'sigKey&引数に正しいデータ種別を指定したことを確認してください。
'PSF表の選択
wRet% = EntSelectAdd(HYP_ID_PSFDATA Or HYP_ID_ASSOC, sigCat%, hSelect%,
    APILOCK_READWRITE, 0, 0, 0, 0)
If wRet% <> APILOCK_READWRITE Then Exit Function
Call EntInitApiStruct(hApp%, lpAPIStruct)
lpAPIStruct.sigCat = sigCat%
lpAPIStruct.lStartPeriod = lStartPeriodSig
lpAPIStruct.lEndperiod = lEndPeriodSig
wRet% = EntConsolidate(hSelect%, iConsolidateFlags, sigTopNode%,
    AddressOf(MyCallBack), lParam, lpAPIStruct)
```

C 言語では次の形式を使用します。

```
short WINAPI EntConsolidate(HSELECT hSelect, HYPBOOL cConsType, SIGNA
sigTopNode, CALLBACKAPI ConsolCB, LONG lParam, LPAPISTRUCT pApiS);
```

## EntCreateApplication( ) - アプリケーションの作成

この関数は、アプリケーションを作成します。アプリケーションは自動的に起動するので、EntOpenApplication( )を呼び出す必要はありません。アプリケーションの使用が終わったら、プログラムの終了前に EntCloseApplication( )を呼び出してアプリケーションを終了します。

カレンダーファイルにパス (\'を含む) を指定しない場合、EntCreateApplication() では実行中のプログラム (のみ) のディレクトリを先頭に付加します。VB のデバッグ環境で EntCreateApplication() を実行している場合、これは VB.EXE を含むディレクトリになります。

**注：** EntCreateApplication() がカレンダーファイルを見つけられない場合、エラーメッセージを表示することはできません。これは、新しいアプリケーションがエラーメッセージルーチンに対して完全に初期化されていないためです。

次の形式を使用します。

Declare Function EntCreateApplication Lib "heaccess.dll" Alias  
"\_EntCreateApplication@40" (ByVal szAppId As String, ByVal szAppDesc As String,  
ByVal szDLL As String, ByVal szCal As String, ByVal szUserName As String, ByVal  
szPassword As String, ByVal lpCallback As Long, lParam As Long, wRet As Integer, ByVal  
sigSite As Long) As Long

変数	説明
szAppId	アプリケーション ID
szAppDesc	アプリケーションの説明
szDLL	ファイルドライバ (HEFILE または HESQL)
szCal	カレンダーファイル。NULL の場合は LOAD.PER が使用されます。
szUserName	アプリケーションを作成しているユーザの ID
szPassword	アプリケーションを作成しているユーザのパスワード
lpCallback	コールバック関数のポインタ。詳しくは、 <a href="#">320 ページの「CALLBACKCREATE」</a> を参照してください。
lParam	lpCallback コールバック関数の引数
wRet	エラーコードを返すためのバッファ
sigSite	NONE、またはサイトの記号

コールバック関数を使用して、作成プロセスにおける適切な時期にアプリケーション設定を定義します。

変換ファイルを使用してアプリケーションを作成するには、アプリケーションの説明の最初の文字を 0x01 ('1'の ASCII コード"1"ではなく数字の 1) に設定し、変換ファイル名を szCal 引数として渡します。

EntCreateApplication()の呼び出しに成功した後は、表 ID\_APPDEFAULT および表 ID\_USERDEFAULT に対して EntSaveDefault()を呼び出す必要があります。

戻りコード：

コード	説明
新しいアプリケーションのハンドル (hApp)	成功
NONE	エラー発生

EntCreateApplication()では、現在のところコールバック関数を呼び出す前に[表 41](#)に示すデフォルト値を設定します。

表 41 EntCreateApplication()によって設定されるデフォルト値

属性	値
APP_USE_MULTI_THREAD	ファイルベースのアプリケーションの場合は TRUE、SQL の場合は FALSE

属性	値
APP_BILLIONS	","
APP_MILLIONS	","
APP_THOUSANDS	","
APP_DECIMAL	"."
APP_NUM_DECIMALS	6
APP_USES_SYSTEM_NUM	0

EntCreateApplication()によって設定されるデフォルト値

APP\_ID

APP\_DRV\_DLL

APP\_DESC

APP\_SITE

APP\_CALENDAR

C 言語では次の形式を使用します。

**HAPP WINAPI EntCreateApplication(LPSTR szAppId, LPSTR szAppDesc, LPSTR szDLL, LPSTR szCal, LPSTR szUserName, LPSTR szPassword, CALLBACKCREATE lpCallback, LPARAM lParam, short \* wRet, SIGNA sigSite);**

例 :

```
Public gsDir As String
' _ _ _
Dim sName As String
Dim sDesc As String
Dim sUserName As String
Dim sUserPass As String
Dim sCal As String
Dim sDLL As String
Dim sigSite As Long
Dim lParam As Long
Dim hApp As Long
Dim nRC As Integer
    sName = Trim(txtName.Text)
    sDesc = Trim(txtDesc.Text)
    sDLL = "HEFILE"
    sCal = "c:\hypent\LOAD.PER"
    sUserName = Trim(txtID.Text)
    sUserPass = Trim(txtPW.Text)
    lParam = 0
    sigSite = HYP_NONE
    gsDir = Trim(frmMain.txtDir.Text)

    hApp = EntCreateApplication(sName, sDesc, sDLL, sCal, sUserName,
sUserPass,
        AddressOf .CreateAppCB, lParam, nRC, sigSite)
```

```

    If hApp <> 0 Then
        MsgBox "アプリケーション作成"
        EntCloseApplication hApp
    Else
        MsgBox "Failed to create app."
    End If
- - - - -
Public Function CreateAppCB(ByVal hApp As Long, ByVal lParam As Long) As
Long
Dim nRC As Integer
Dim sReport As String
Dim sOutbox As String
Dim sInbox As String
Dim sData As String
    sReport = "@APP\REPORT"
    sOutbox = "@APP\OUTBOX"
    sInbox = "@APP\INBOX"
    sData = "@APP\DATA"
' *****
' *****
' **** 文字列にはEntUpdateDefaultではなくEntUpdateDefaultStrを使用
' *****
' *****

    nRC = EntUpdateDefaultStr(hApp, HYP_ID_APPDEFAULT,
HYP_APP_PATH, gsDir)

nRC = EntUpdateDefaultStr(hApp, HYP_ID_APPDEFAULT, HYP_APP_DATADIR,
sData)
nRC = EntUpdateDefaultStr(hApp, HYP_ID_APPDEFAULT, HYP_APP_INBOXDIR,
sInbox)
nRC = EntUpdateDefaultStr(hApp, HYP_ID_APPDEFAULT, HYP_APP_OUTBOXDIR,
sOutbox)
nRC = EntUpdateDefaultStr(hApp, HYP_ID_APPDEFAULT, HYP_APP_REPORTDIR,
sReport)
nRC = EntUpdateDefault(hApp, HYP_ID_APPDEFAULT, HYP_APP_CONSOL_IS_PER,
1)
nRC = EntUpdateDefault(hApp, HYP_ID_APPDEFAULT, HYP_APP_IS_ORGBYPER, 0)
nRC = EntUpdateDefault(hApp, HYP_ID_APPDEFAULT,
HYP_APP_JOUR_AUTONUMBER, 0)
nRC = EntUpdateDefault(hApp, HYP_ID_APPDEFAULT,
HYP_APP_USE_CHILD_RATES, 0)
nRC = EntUpdateDefault(hApp, HYP_ID_APPDEFAULT, HYP_APP_SUBACCTSIG, 1)
nRC = EntUpdateDefault(hApp, HYP_ID_APPDEFAULT,
HYP_APP_STORETRANDETAIL, 0)
nRC = EntUpdateDefault(hApp, HYP_ID_APPDEFAULT, HYP_APP_STORECONDETAIL,
0)
nRC = EntUpdateDefault(hApp, HYP_ID_APPDEFAULT, HYP_APP_USETURBO, 0)
nRC = EntUpdateDefault(hApp, HYP_ID_APPDEFAULT, HYP_APP_DEVERASELOG, 0)
nRC = EntUpdateDefault(hApp, HYP_ID_APPDEFAULT, HYP_APP_JOUR_TOPLEVEL,
0)
nRC = EntUpdateDefault(hApp, HYP_ID_APPDEFAULT,
HYP_APP_IMPACTFUTURECAT, 0)
nRC = EntUpdateDefault(hApp, HYP_ID_APPDEFAULT,
HYP_APP_EXPECTED_ENTITIES, 0)

```

```

nRC = EntUpdateDefault(hApp, HYP_ID_APPDEFAULT,
HYP_APP_EXPECTED_REPORTS, 0)
nRC = EntUpdateDefault(hApp, HYP_ID_APPDEFAULT, HYP_APP_EXPECTED_ACCTS,
0)
nRC = EntUpdateDefault(hApp, HYP_ID_APPDEFAULT, HYP_APP_EXPECTED_CATS,
0)
nRC = EntSaveDefault(hApp, HYP_ID_APPDEFAULT)
' *****
' *****
' **** アプリケーションを作成するには"1"を返す必要あり
' *****
' *****

CreateAppCB = 1

End Function

```

## EntDataExtract( ) - データの抽出

この関数は、データの抽出を実行します。データを抽出する勘定科目を列挙する関数を提供する必要があります。一部の状況では、EntDataExtract( )を呼び出す前に、ID\_NAMELISTENTRY 表と ID\_PSFDATA 表を選択する必要があります。

**注：** この関数は、Visual Basic 以外のすべての言語で使用できます。Visual Basic アプリケーションには EntDataExtractVB2( )を使用してください。C 言語のプログラマは、EntDataExtractVB2( )を使用することもできます。

サーバ上でデータを抽出する場合（DBE\_REMOTE ビットフラグが pS->wExtractFlags に設定されている場合）は、抽出処理のプロセスハンドルを認識し、プログラムによって起動されるサーバ処理を追跡および監視できるようにすると便利です。pS -> hProcess に返されるプロセスハンドルは、ローカルクライアントコンピュータで実行されている [タスクステータス] ウィンドウプログラム HComMgr.exe のインスタンスのプロセスハンドルです。詳しくは、[121 ページの「サーバのタスク」](#)を参照してください。

C 言語では次の形式を使用します。

**short WINAPI EntDataExtract(HSELECT hSelect, LPDBEXTRACTSTRUCT pS)**

変数	説明
hSelect	選択された表のハンドル
pS	データ抽出用の引数を含んでいる構造体を指すポインタ。hProcess 以外のすべてのフィールドを指定する必要があります。
pS->dwSize	この構造体のサイズ
pS->cDelim	フィールドの区切り文字で、通常はカンマ (,)
pS->cNeg	負の区切り文字で、通常はダッシュまたはアポストロフィ (-または')
pS->cScale	データの単位

変数	説明
pS->cView	データ表示形式 (FORMVIEW_YTD、FORMVIEW_PER、FORMVIEW_CAT)。データ種別のデフォルトの表示形式を使用するには NONE を使用します。
pS->wDecimals	抽出する小数点以下桁数の数
pS->wDSMType	詳細なストレージモデルタイプ (ID_REGULAR、ID_PROPORTIONAL など)。ID_REGULAR 以外には sigParent が必要。
pS->wExtractFlags	データ抽出オプション (ToolInc.h の DBE_APPEND、DBE_REMOTE など参照)
pS->wOperation	FRMT_OP_MUL (乗算)、FRMT_OP_DIV (除算) または NONE
pS->dDataBy	wOperation に使用するデータ
pS->bDataFileOpened	データファイルが現在開いているかどうか (TRUE または FALSE)。入力および出力。
pS->hProcess	サーバ上での実行時に返されるプロセスハンドル (DBE_REMOTE)
pS->lStartPeriod	抽出する最初の期間 (0 から始まる)
pS->lEndPeriod	抽出する最後の期間 (0 から始まる)
pS->sigAcctConv	勘定科目変換表記号
pS->sigCat	データ種別記号
pS->sigName	エンティティ記号
pS->sigNameConv	組織単位変換表記号
pS->sigParent	wDSMType の親
pS->sigNameList	エンティティ一覧
pS->szExtractFile	抽出ファイルのパス名
pS->pApiDFA	エンティティ一覧の列挙 (pS->sigNameList != NONE の場合) およびデータファイルからのデータのクエリ (割り当てられているデータおよびステータスバッファ) に設定されている apiStruct。  詳しくは、 <a href="#">177 ページの「EntEnum() と ID_NAMELISTENTRY (エンティティ一覧入力表)」</a> および <a href="#">114 ページの「データの操作」</a> を参照してください。
pS->pApiDFA->sigName	エンティティ記号
pS->pApiDFA->u_Dfa.bAutoRecalc	詳しくは、 <a href="#">114 ページの「データの操作」</a> を参照してください。
pS->fnEnumAcct	勘定科目を列挙するためのコールバック関数。詳しくは、 <a href="#">321 ページの「CALLBACKDBENUM」</a> を参照してください。
pS->lParamEnumAcct	fnEnumAcct の引数
pS->fnStatusCB	ステータス情報のコールバック関数 (DBE_REMOTE ビットフラグが pS->wExtractFlags に設定されている場合は使用されません)。詳しくは、 <a href="#">324</a>

変数	説明
	ページの「CALLBACKDBLOAD - EntDataExtract()の場合」を参照してください。
pS->lParamStatus	fnStatusCB の引数

戻りコード：

コード	説明
0	成功
NONE	エラー発生
LOAD_CHECK_ERRORLOG (2)	少なくとも 1 つのエラーが発生しており、アプリケーションのエラーログに示されています。

EntDataExtract()を呼び出す前に、データの視点を必ず設定してください。次にその例を示します。

```
wRet = EntUpdateDefault(hApp, ID_USERDEFAULT, CURORSIG, sigOrg);
wRet = EntUpdateDefault(hApp, ID_USERDEFAULT,
    CURCATEGORYSIG, sigCat);
wRet = EntUpdateDefault(hApp, ID_USERDEFAULT,
    CURNAMEISIG, sigEntity);
```

**注：** pS->fnEnumAcct および pS->fnStatusCB フィールドの値を指定する必要があります。

## EntDataExtractVB2( ) - データの抽出

この関数は、勘定科目一覧内の勘定科目またはすべての勘定科目のデータを抽出します。一部の状況では、EntDataExtractVB2( )を呼び出す前に、ID\_NAMELISTENTRY 表と ID\_PSFDATA 表を選択する必要があります。

**注：** この関数は Visual Basic ユーザを対象としていますが、他のプログラムも使用できます。

次の形式を使用します。

```
Declare Function EntDataExtractVB2 Lib "heaccess.dll" Alias "_EntDataExtractVB@24"
_ (ByVal hSelect As Long, pArgs As DBEXTRACTSTRUCT, ByVal szExtractFile As String,
ByVal sigList As Long, ByVal lpfnCallback As Long, ByVal lParam As Long) As Integer
```

変数	説明
hSelect	選択された表のハンドル
pArgs	データ抽出用の引数を含んでいる構造体。hProcess 以外のすべてのフィールドを指定する必要があります。

変数	説明
pArgs.dwSize	この構造体のサイズ
pArgs.cDelim	フィールドの区切り文字で、通常はカンマ (,)
pArgs.cNeg	負の区切り文字で、通常はダッシュ (-) またはかっこ [()]
pArgs.cScale	データの単位
pArgs.cView	データ表示形式 (FORMVIEW_YTD、FORMVIEW_PER、FORMVIEW_CAT)。データ種別のデフォルトの表示形式を使用するには NONE を使用します。
pArgs.wDecimals	抽出する小数点以下桁数の数
pArgs.wDSMType	詳細なストレージモデルタイプ (ID_REGULAR、ID_PROPORTIONAL など)。ID_REGULAR 以外には sigParent が必要。
pArgs.wExtractFlags	データ抽出オプション (ToolInc.h の DBE_APPEND、DBE_REMOTEなどを参照)
pArgs.wOperation	演算: FRMT_OP_MUL (乗算)、FRMT_OP_DIV (除算) または NONE
pArgs.dDataBy	wOperation に使用するデータ
pArgs.bDataFileOpened	データファイルが現在開いているかどうか (TRUE または FALSE)。入力および出力。
pArgs.hProcess	サーバ上での実行時に返されるプロセスハンドル (DBE_REMOTE)
pArgs.lStartPeriod	抽出する最初の期間 (0 から始まる)
pArgs.lEndPeriod	抽出する最後の期間 (0 から始まる)
pArgs.sigAcctConv	勘定科目変換表記号
pArgs.sigCat	データ種別記号
pArgs.sigName	エンティティ記号
pArgs.sigNameConv	エンティティ変換表記号
pArgs.sigParent	wDSMType が ID_REGULAR 以外の場合は親
pArgs.sigNameList	エンティティ一覧
pArgs.pApiDFA	エンティティ一覧の列挙 (pArgs.sigNameList != NONE の場合) およびデータファイルからのデータのクエリ (割り当てられているデータおよびステータスバッファ) に設定されている apiStruct のアドレス  詳しくは、 <a href="#">177 ページの「EntEnum()と ID_NAMELISTENTRY (エンティティ一覧入力表)」</a> および <a href="#">114 ページの「データの操作」</a> を参照してください。
pArgs.pApiDFA.sigName	エンティティ記号
pArgs.pApiDFA.u_Dfa.bAutoRecalc	詳しくは、 <a href="#">114 ページの「データの操作」</a> を参照してください。
szExtractFile	抽出ファイルのパス名

変数	説明
sigList	勘定科目一覧の記号。すべての勘定科目の場合は NONE。
lpfnCallback	ステータス情報用のコールバック関数。詳しくは、 <a href="#">331 ページの「CALLBACKVB」</a> を参照してください。
IParam	コールバック関数の引数

EntDataExtractVB2( )は内部でファイル名文字列のローカルコピーを作成するので、Visual Basic によってコールバック関数の呼び出し中に元のコピーが移動されても問題はありません。

戻りコード：

コード	説明
0	成功
NONE	エラー発生
LOAD_CHECK_ERRORLOG (2)	少なくとも 1 つのエラーが発生しており、アプリケーションのエラーログに示されています。

例：

```

Dim s As DBEXTRACTSTRUCT
    Dim apiS As APISTRUCT
    lPer& = Val(per$)-1
    sigAcctList& = HYP_NONE
    '勘定科目一覧を使用している場合は、表を選択して記号を取得する
If LenB(acctList$) Then
    ret% = EntSelectAdd(HYP_ID_ACCTLIST Or HYP_ID_ASSOC, HYP_NONE,
hSelect&, APILOCK_READONLY, 0, 0, 0, 0)
    sigAcctList& = EntFind0(hSelect&, HYP_ID_ACCTLIST, HYP_NONE, acctList$,
0)
End If
'さまざまなチェックボックスに基づいてオプションを設定
    flag% = 0
If chkCalc.Value Then flag% = flag% Or DBE_CALC_ACCTS
If chkGlobal.Value Then flag% = flag% Or DBE_GLOBAL_ACCTS
If chkSupNoData.Value Then flag% = flag% Or DBE_SUPPRESS_ND
If chkSupZero.Value Then flag% = flag% Or DBE_SUPPRESS_ZERO
If chkDerived.Value Then flag% = flag% Or DBE_EXTRACT_DERIVED
'apiStructフィールドを設定
    hApp& = HypGethApp(frmMain.ghRApp%)
    Call EntInitApiStruct(hApp&, apiS)
    apiS.sigName = sigEntity&
    apiS.u_Dfa.bAutoRecalc = 0
'DBEXTRACTSTRUCTフィールドを設定
s.dwSize = LenB(s)
s.cDelim = Asc(",")
s.cNeg = Asc("-")
s.cScale = 255
s.cView = Asc(FORMVIEW_CAT)
s.wDecimals = HYP_NONE
s.wDSMTType = ID_REGULAR
'デフォルトにはNONEを使用

```

```

s.wExtractFlags = flag%
s.wOperation = HYP_NONE
s.dDataBy = -1 'wOperationで使用するデータ
s.bDataFileOpened = 0 'DataFileが現在開いているかどうか (TRUEまたは
FALSE)。入力および出力
s.hProcess = 0 'サーバ上での実行時に返されるプロセスハンドル
(DBE_REMOTE)
s.lStartPeriod = lPer% '抽出する最初の期間 (0から始まる)
s.lEndPeriod = lPer% '抽出する最後の期間 (0から始まる)
s.sigAcctConv = HYP_NONE '勘定科目変換表記号
s.sigCat = sigCat% 'データ種別記号
s.sigName = sigEntity% 'エンティティ記号
s.sigNameConv = HYP_NONE 'エンティティ変換表記号
s.sigParent = HYP_NONE '親 (wDSMTypeがID_REGULAR以外の場合)
s.sigNameList = HYP_NONE 'エンティティ一覧
s.fnEnumAcct = 0 '勘定科目を列挙する関数のアドレス (未使用のため0に設
定)
s.lParamEnumAcct = 0 'fnEnumAcct関数の引数
s.fnStatusCB = 0 'ステータス情報のコールバック関数のアドレス (未使用
のため代わりにlpfnCallbackを使用)
s.lParamStatus = 0 'fnStatusCB関数の引数
s.pApiDFA = EntGetVarAddr(apiS) 'APISTRUCTのアドレス。
apiStruct内の記号フィールドを設定。EntDataExtract()によってapiStructが変更され
る可能性あり。
s.szExtractFile = 0 '未使用のため代わりにszExtractFile引数を使用
'この例ではコールバック関数のSpyWorksを使用
ret% = EntDataExtractVB2(hSelect%, s, szFile$, sigAcctList%,
cbkExtract.ProcAddress, 0)
If LenB(acctList%) Then
hSelect% = EntUnSelect0(hSelect%, HYP_ID_ACCTLIST Or HYP_ID_ASSOC,
HYP_NONE, 0, ret%, 0)
End If
.
.
.
'SpyWorks関数のコールバック
Private Sub cbkExtract_cbxLLL(lval1 As Long, lval2 As Long, lval3 As
Long, retval As Long)
Dim ret%, wLen%
Dim szFile$
Dim args As CALLBACKSTRUCT
'lval1は'ByVal szFile As String'になる必要あり
'lval1からszFile$を取得
wLen% = EntVBGetCStrLen(lval1)
szFile$ = Space(wLen%)
ret% = EntVBCopyStr(szFile$, lval1, wLen%)
'lval2は実際には'args As CALLBACKSTRUCT'になる必要あり
'argsをlval2から取得
ret% = EntVBCopyData(args, lval2, LenB(args))
ret% = MyCallBack (szFile$, args, lval3)
retval% = ret%
EndSub
Function MyCallBack (szFile$, args As CALLBACKSTRUCT, ByVal lparam%) As
Integer
Dim szCat$, szEntity$, szMsg$
szMsg$ = "File: [" & szFile$
szCat$ = Space(HYP_SIZELABEL + 1)

```

```

        ret% = EntQueryExStr0(args.hSelect, HYP_ID_CATEGORY,
HYP_NAME, args.sigCat, HYP_NONE, Len(szCat$), szCat$, 0)
        If ret% = 0 Then szCat$ = CToBStr(szCat$)
        szMsg$ = szMsg$ & "], Category: [" & szCat$
        szEntity$ = Space(HYP_SIZEFULLNAME + 1)
        ret% = EntQueryExStr0(args.hSelect, HYP_ID_NAMES, HYP_NAME,
args.sigEntity, HYP_NONE, Len(szEntity$), szEntity$, 0)
        If ret% = 0 Then szEntity$ = CToBStr(szEntity$)
        szMsg$ = szMsg$ & "], Entity: [" & szEntity$
        szMsg$ = szMsg$ & "]" 期間 " & Str(args.lStart) & " 終了期間 "
& Str(args.lEnd)
        ret% = MsgBox(szMsg$, vbOKOnly, "抽出中...")
        MyCallBack% = 0
End Function

```

**注：** この例における **EntFind0 ( )**、**EntQueryExStr0 ( )**および **EntUnSelect0 ( )**の宣言は、**EntFind()**、**EntQueryExStr ( )**および **EntUnSelect()**に似ていますが、**apiStruct** の代わりに 0 を渡すことができるように調整されています。詳しくは、[16 ページの「Visual Basic プログラミング上の注意」](#)を参照してください。

C 言語では次の形式を使用します。

```

EntDataExtractVB2(HSELECT hSelect, LPDBEXTRACTSTRUCT pArgs, LPCSTR
szExtractFile, SIGNA sigList, CALLBACKVB lpfnCallback, LPARAM lParam);

```

## EntDataFileOpen( ) - データファイルを開く

この関数は、指定されたエンティティ、データ種別および親のデータファイル表を選択します。この関数は、データ種別とエンティティの各組み合わせに使用する必要があります。データファイル表には、指定されたデータ種別とエンティティに関する各期間の勘定科目値が格納されます。この表を使用した後は、**EntUnSelect ( )**を呼び出して表の選択を解除します。

警告：指定されているデータ種別とエンティティに対して **ID\_REGULAR** データファイル表が既に選択されている可能性がある場合、**ID\_TRANSLATION\_FORCE** タイプのデータファイル表は選択しないでください。同様に、**ID\_TRANSLATION\_FORCE** タイプが既に選択されている可能性がある場合、**ID\_REGULAR** タイプのデータファイル表は選択しないでください。

次の形式を使用します。

```

Declare Function EntDataFileOpen Lib "heaccess.dll" Alias "_EntDataFileOpen@28"
(ByVal hSelect As Long, ByVal wType As Integer, ByVal sigCat As Long, ByVal SigEntity
As Long, ByVal sigParent As Long, ByVal wAttr As Integer, ByVal bCreate As Integer) As
Integer

```

### 変数 説明

**hSelect** 選択された表のハンドル

**wType** 次の表を参照してください。

## 変数 説明

sigCat データ種別記号。

sigEntity エンティティ記号

sigParent wType = ID\_REGULAR の場合は NONE、それ以外の場合は親の記号

wAttr データファイルの保護方法。次のコードのいずれかを使用します。

- 表を読み取り専用で開く場合は APILOCK\_READONLY
- 表を更新するために開く場合は APILOCK\_READWRITE
- 読み取り／書き込みで開くのが望ましいが、読み取り専用も可能な場合は APILOCK\_DEFAULT

bCreate データファイルが存在しない場合に作成する場合は TRUE、それ以外の場合は FALSE

フィールド wType	説明
ID_ADJUSTMENT	調整
ID_CONTRIBUTION	調整後
ID_ELIMINATION	消去
ID_PROPORTIONAL	比率
ID_REGULAR	標準表
ID_ROLLOVERS	期別替
ID_TRANSLATION	換算
ID_TRANSLATION_FORCE	通貨が同じ場合は換算または子

戻りコード：

コード	説明
wAttr	付与されている実際の権限を返します。
NONE	エラー発生
APILOCK_MULTUSER	複数のユーザが使用しているため読み取り専用として選択されています。

例：

```
rc% = EntDataFileOpen(hSelect&, ID_REGULAR, sigCat&, SigEntity&, HYP_NONE, APILOCK_READONLY, False)
```

C 言語では次の形式を使用します。

```
short WINAPI EntDataFileOpen(HSELECT hSelect, WORD wType, SIGNA sigCat, SIGNA sigEntity, SIGNA sigParent, short wAttr, BOOL bCreate);
```

## EntDateConv( ) - 日付の変換と期間番号の計算

1 つのデータ種別（参照データ種別）の期間が与えられたとき、EntDateConv()では別のデータ種別と期間単位（最下位データ種別および期間単位）の対応する期間を計算します。この関数を呼び出す前に、レポート期間単位とレポート表示形式の表（ID\_RPTFREQ と ID\_RPTVIEW）を選択してください。

次の形式を使用します。

Declare Function EntDateConv Lib "heaccess.dll" Alias "\_EntDateConv@28" (ByVal hSelect As Long, ByVal wMethod As Integer, ByVal szRefCat As String, ByVal szRefPeriod As String, ByVal iOffset As Integer, ByVal szFreq As String, ByVal szBaseCat As String) As Long

変数	説明
hSelect	選択された表のハンドル
wMethod	年を無視した現在の期間の DCONV_CMO、または現在の期間の DCONV_CUR
szRefCat	参照データ種別 ID
szRefPeriod	参照期間（参照データ種別内の期間）
iOffset	参照期間からオフセットされた期間の番号
szFreq	最下位レポート期間単位の ID。レポート期間単位と表示形式については、 <a href="#">20 ページの「期間単位と表示形式」</a> を参照してください。
szBaseCat	最下位データ種別 ID

戻りコード：

コード	説明
期間番号	成功
BAD_PERIOD	エラー発生

例：

```
sigPeriod& = EntDateConv(hSelect&, DCONV_CUR, "ACTUAL", "JAN 03", 0, "MON", "LASTYR")
```

C 言語では次の形式を使用します。

```
SIGNA WINAPI EntDateConv(HSELECT hSelect, short wMethod, char * szRefCat, char * szRefPeriod, short iOffset, char * szFreq, char * szBaseCat);
```

## EntDeleteFromCBChain( ) - コールバックチェーンからの削除

この関数は、hSelect 引数変数に関連付けられているコールバック関数のチェーンからコールバック関数を削除します。詳しくは、[112 ページの「選択のコールバック」](#)を参照してください。

次の形式を使用します。

```
Declare Function EntDeleteFromCBChain Lib "heaccess.dll" Alias
"_EntDeleteFromCBChain@12" (ByVal hSelect As Long, ByVal lpCallback As Long,
ByVal lParam As Long) As Integer
```

#### 変数 説明

hSelect 選択された表のハンドル

lpCallback チェーンから削除するコールバック関数

lParam lpCallback 関数の引数。これは、コールバック関数をコールバック関数のチェーンに追加したときに指定された引数になっている必要があります。

戻りコード：

コード	説明
0	成功
NONE	エラー発生

C 言語では次の形式を使用します。

```
short WINAPI EntDeleteFromCBChain(HSELECT hSelect, CALLBACKSEL lpCallback,
LONG lParam);
```

## EntDiscardChanges( ) - 変更の破棄

この関数は、EntUpdate( )または EntUpdateDefault( )によって 1 つまたは複数の表に対して行われた変更を破棄します。EntSave( )または EntSaveDefault( )を呼び出す前に、この関数を呼び出してください。

次の形式を使用します。

```
Declare Function EntDiscardChanges Lib "heaccess.dll" Alias "_EntDiscardChanges@16"
(ByVal hSelect As Long, ByVal wTabId As Integer, ByVal sigKey As Long, apiS As
APISTRUCT) As Integer
```

#### 変数 説明

hSelect 選択された表のハンドル

wTabId 表 ID、または hSelect のすべての表を対象とする場合は NONE。付録 A「表 ID」に含まれている有効な表 ID のいずれかを指定できます。

sigKey NONE、または関連する表のキー。詳しくは、340 ページの「関連する表」を参照してください。

apiS NULL、または apiStruct のポインタ

戻りコード：

コード	説明
0	成功
NONE	エラー発生

C 言語では次の形式を使用します。

```
short WINAPI EntDiscardChanges(HSELECT hSelect, short wTabID, SIGNA sigKey,
LPAPISTRUCT pApiS);
```

## EntDiscardDefault( ) - デフォルトの破棄

この関数は、EntUpdateDefault( )によってデフォルト設定に対して行われた変更を破棄します。この関数は EntSaveDefault( )を呼び出す前に呼び出してください。そうしないと変更を破棄できません。

次の形式を使用します。

```
Declare Function EntDiscardDefault Lib "heaccess.dll" Alias "_EntDiscardDefault@8"
(ByVal hApp As Long, ByVal wTabId As Integer) As Integer
```

### 変数 説明

hApp アプリケーションハンドル

wTabId ID\_APPDEFAULT や ID\_USERDEFAULT などの表 ID。詳しくは、[338 ページの「デフォルト設定の表」](#)を参照してください。

戻りコード：

コード	説明
0	成功
NONE	エラー発生

C 言語では次の形式を使用します。

```
short WINAPI EntDiscardDefault(HAPP hApp, short wTabId);
```

## EntDSMDataExtract( ) - DSM データの抽出

この関数は、連結詳細を C から抽出します。選択される表は、ID\_NAMELISTENTRY と ID\_PSFDATA です。

**注：** VB プログラマは、EntDSMDataExtractVB( )を使用して連結詳細を抽出してください。

C 言語では次の形式を使用します。

**short WINAPI EntDSMDataExtract(HSELECT hSelect, LPDBEXTRACTSTRUCT pS, LPDSMEXTRACT pDsm)**

変数	説明
hSelect	選択された表のハンドル
pS	データ抽出用の引数を含んでいる構造体を指すポインタ。hProcess 以外のすべてのフィールドを指定する必要があります。
pS->dwSize	この構造体のサイズ
pS->cDelim	フィールドの区切り文字で、通常はカンマ (,)
pS->cNeg	負の区切り文字で、通常はダッシュまたはアポストロフィ (-または')
pS->cScale	データの単位
pS->cView	データ表示形式 (FORMVIEW_YTD、FORMVIEW_PER、FORMVIEW_CAT)。データ種別のデフォルトの表示形式を使用するには NONE を使用します。
pS->wDecimals	抽出する小数点以下桁数の数
pS->wDSMType	詳細なストレージモデルタイプ (ID_REGULAR、ID_PROPORTIONAL など)。ID_REGULAR 以外には sigParent が必要。
pS->wExtractFlags	データ抽出オプション (ToolInc.h の DBE_APPEND、DBE_REMOTEなどを参照)
pS->wOperation	FRMT_OP_MUL (乗算)、FRMT_OP_DIV (除算) または NONE
pS->dDataBy	wOperation に使用するデータ
pS->bDataFileOpened	データファイルが現在開いているかどうか (TRUE または FALSE)。入力および出力。
pS->hProcess	サーバ上での実行時に返されるプロセスハンドル (DBE_REMOTE)
pS->lStartPeriod	抽出する最初の期間 (0 から始まる)
pS->lEndPeriod	抽出する最後の期間 (0 から始まる)
pS->sigAcctConv	勘定科目変換表記号
pS->sigCat	データ種別記号
pS->sigName	エンティティ記号
pS->sigNameConv	組織単位変換表記号
pS->sigParent	wDSMType の親
pS->sigNameList	エンティティ一覧
pS->szExtractFile	抽出ファイルのパス名

変数	説明
pS->pApiDFA	エンティティ一覧の列挙（pS->sigNameList != NONE の場合）およびデータファイルからのデータのクエリ（割り当てられているデータおよびステータスバッファ）に設定されている apiStruct
pS->pApiDFA->sigName	エンティティ記号
pS->pApiDFA->u_ Dfa.bAutoRecalc	詳しくは、 <a href="#">114 ページの「データの操作」</a> を参照してください。
pS->fnEnumAcct	勘定科目を列挙するためのコールバック関数。詳しくは、 <a href="#">321 ページの「CALLBACKDBENUM」</a> を参照してください。
pS->IParamEnumAcct	fnEnumAcct の引数
pS->fnStatusCB	ステータス情報のコールバック関数（DBE_REMOTE ビットフラグが pS->wExtractFlags に設定されている場合は使用されません）。詳しくは、 <a href="#">324 ページの「CALLBACKDBLOAD - EntDataExtract()の場合」</a> を参照してください。
pS->IParamStatus	fnStatusCB の引数
pDsm	DSM 抽出用の引数を含んでいる構造体
pDsm->dwSize	この構造体のサイズ
pDsm->wExtractFlags	データ抽出オプションフラグ 次の値のいずれかまたはすべての組み合わせを使用します。
DBE_DSM_TRANS	変換データの抽出
DBE_DSM_PROP	比率データの抽出
DBE_DSM_CONTRIB	抽出調整後データ
DBE_DSM_ELIM	消去データの抽出
DBE_DSM_TOPADJ	親調整データの抽出
pDsm->cSeparator	連結詳細の抽出時に使用するフィールド区切り文字（デフォルトは' '）
pDsm->bXAFormat	形式（連結された 1 つのフィールド）または外部形式（3 つのフィールド）を使用した抽出
pDsm->bAllDeps	すべての子の連結詳細の抽出。すべての子を含めない場合は直属の子の一覧を使用します。
pDsm->bAllImmedDeps	すべての子の連結詳細の抽出。すべての子を含めない場合は直属の子の一覧を使用します。
pDsm->iSelectedDependents	選択されている子の数
pDsm->lpSelectedDepSigs	選択されている子の一覧

戻りコード：

コード	説明
0	成功
NONE	エラー発生
LOAD_CHECK_ERRORLOG (2)	少なくとも 1 つのエラーが発生しており、アプリケーションのエラーログに示されています。

## EntDSMDataExtractVB( ) - DSM データの抽出

この関数は、連結詳細を VB から抽出します。選択される表は、ID\_NAMELISTENTRY と ID\_PSFDATA です。

**注：** C 言語のプログラマは、EntDSMDataExtract( )を使用して連結詳細を抽出してください。

次の形式を使用します。

```
Declare Function EntDSMDataExtractVB Lib "heaccess.dll" Alias
    "_EntDSMDataExtractVB@28" _ (ByVal hSelect As Long, pArgs As
    DBEXTRACTSTRUCT, ByVal szExtractFile As String, ByVal sigList As Long, _ ByVal
    lpfnCallback As Long, ByVal lParam As Long, dsm As DSMDBEXTRACTSTRUCT) As
    Integer
```

変数	説明
hSelect	選択された表のハンドル
pArgs	データ抽出用の引数を含んでいる構造体。hProcess 以外のすべてのフィールドを指定する必要があります。
pArgs.dwSize	この構造体のサイズ
pArgs.cDelim	フィールドの区切り文字で、通常はカンマ (,)
pArgs.cNeg	負の区切り文字で、通常はダッシュ (-) またはかっこ []
pArgs.cScale	データの単位
pArgs.cView	データ表示形式 (FORMVIEW_YTD、FORMVIEW_PER、FORMVIEW_CAT)。データ種別のデフォルトの表示形式を使用するには NONE を使用します。
pArgs.wDecimals	抽出する小数点以下桁数の数
pArgs.wDSMType	詳細なストレージモデルタイプ (ID_REGULAR、ID_PROPORTIONAL など)。ID_REGULAR 以外には sigParent が必要。
pArgs.wExtractFlags	データ抽出オプション (ToolInc.h の DBE_APPEND、DBE_REMOTEなどを参照)
pArgs.wOperation	演算：FRMT_OP_MUL (乗算)、FRMT_OP_DIV (除算) または NONE

変数	説明
pArgs.dDataBy	wOperation に使用するデータ
pArgs.bDataFileOpened	データファイルが現在開いているかどうか (TRUE または FALSE)。入力および出力。
pArgs.hProcess	サーバ上での実行時に返されるプロセスハンドル (DBE_REMOTE)
pArgs.lStartPeriod	抽出する最初の期間 (0 から始まる)
pArgs.lEndPeriod	抽出する最後の期間 (0 から始まる)
pArgs.sigAcctConv	勘定科目変換表記号
pArgs.sigCat	データ種別記号
pArgs.sigName	エンティティ記号
pArgs.sigNameConv	エンティティ変換表記号
pArgs.sigParent	wDSMType が ID_REGULAR 以外の場合は親
pArgs.sigNameList	エンティティ一覧
pArgs.pApiDFA	エンティティ一覧の列挙 (pArgs.sigNameList != NONE の場合) およびデータファイルからのデータのクエリ (割り当てられているデータおよびステータスバッファ) に設定されている apiStruct のアドレス
pArgs.pApiDFA.sigName	エンティティ記号
pArgs.pApiDFA.u_Dfa.bAutoRecalc	詳しくは、 <a href="#">114 ページの「データの操作」</a> を参照してください。
szExtractFile	抽出ファイルのパス名
sigList	勘定科目一覧の記号。すべての勘定科目の場合は NONE。
lpfnCallback	ステータス情報用のコールバック関数。詳しくは、 <a href="#">331 ページの「CALLBACKVB」</a> を参照してください。
lParam	コールバック関数の引数
dsm	DSM 抽出用の引数を含んでいる構造体
dsm.dwSize	この構造体のサイズ
dsm.wExtractFlags	データ抽出オプションフラグ 次の値のいずれかまたはすべてを使用します。
DBE_DSM_TRANS	変換データの抽出
DBE_DSM_PROP	比率データの抽出
DBE_DSM_CONTRIB	抽出調整後データ
DBE_DSM_ELIM	消去データの抽出
DBE_DSM_TOPADJ	親調整データの抽出

変数	説明
dsm.cSeperator	連結詳細の抽出時に使用するフィールド区切り文字（デフォルトは' '）
dsm.bXAFormat	形式（連結された 1 つのフィールド）または外部形式（3 つのフィールド）を使用した抽出
dsm.bAllDeps	すべての子の連結詳細の抽出。すべての子を含めない場合は直属の子の一覧を使用します。
dsm.bAllImmedDeps	すべての直属の子の連結詳細の抽出。すべての子を含めない場合は直属の子の一覧を使用します。
dsm.iSelectedDependents	選択されている子の数。dsm.bAllDeps と dsm.bAllImmedDeps の両方が false の場合に使用します。
dsm.lpSelectedDepSigs	選択されている子の一覧

戻りコード：

コード	説明
0	成功
NONE	エラー発生
LOAD_CHECK_ERRORLOG (2)	少なくとも 1 つのエラーが発生しており、アプリケーションのエラーログに示されています。

例：

```
Private Sub DSMEExtract
Dim acctList$, entity$, cat$, per$, szFile$
Dim sigEntity&, sigCat&, sigAcctList&, lPer&, sigFreq&
Dim s As DBEXTRACTSTRUCT
Dim hApp&, hSelect&
Dim flag%, ret%
Dim apiS As apiStruct
Dim sigArr(4) As Long
Dim dsm As DSMDBEXTRACTSTRUCT
szFile$ = RTrim(txtFilename.Text)
cat$ = RTrim(frmMain.lblCat.Caption)
entity$ = RTrim(frmMain.lblEntity.Caption)
per$ = RTrim(frmMain.lblPer.Caption)
If frmMain.ghRApp% = 0 Or LenB(entity$) = 0 Or LenB(cat$) = 0 Or
    LenB(per$) = 0 Or LenB(szFile$) = 0 Then
ret% = MsgBox("Please select application, category,
entity,
    period, and filename first.", vbOKOnly, "Error!")
Exit Sub
End If
hSelect& = HypGethSelect(frmMain.ghRApp%)
sigCat& = EntFind0(hSelect&, HYP_ID_CATEGORY, HYP_NONE, cat$, 0)
sigEntity& = EntFind0(hSelect&, HYP_ID_NAMES, HYP_NONE, entity$, 0)

lPer& = Val(per$) - 1
```

```

acctList$ = RTrim(frmMain.lblAcctList.Caption)
sigAcctList& = HYP_NONE
If Len(acctList$) Then
    ret% = EntSelectAdd(HYP_ID_ACCTLIST Or HYP_ID_ASSOC,
        HYP_NONE, hSelect&, APILOCK_READONLY, _ 0, 0, 0, 0)
    sigAcctList& = EntFind0(hSelect&, HYP_ID_ACCTLIST, HYP_NONE,
        acctList$, 0)
End If
flag% = 0
If chkCalc.Value Then flag% = flag% Or DBE_CALC_ACCTS
If chkGlobal.Value Then flag% = flag% Or DBE_GLOBAL_ACCTS
If chkSupNoData.Value Then flag% = flag% Or DBE_SUPPRESS_ND
If chkSupZero.Value Then flag% = flag% Or DBE_SUPPRESS_ZERO
If chkDerived.Value Then flag% = flag% Or DBE_EXTRACT_DERIVED
hApp& = HypGethApp(frmMain.ghRApp%)
Call EntInitApiStruct(hApp&, apiS)
apiS.sigName = sigEntity&
apiS.u_Dfa.bAutoRecalc = 0
s.dwSize = LenB(s)
s.cDelim = Asc(",")
s.cNeg = Asc("-")
s.cScale = 255 'デフォルトにはNONEを使用
s.cView = Asc(FORMVIEW_CAT)
s.wDecimals = HYP_NONE
s.wDSMType = ID_REGULAR
s.wExtractFlags = flag%
s.wOperation = HYP_NONE
s.dDataBy = -1 'wOperationで使用するデータ
s.bDataFileOpened = 0 'DataFileが現在開いているかどうか (TRUEまたは
FALSE) 。 入力および出力

s.hProcess = 0 'サーバ上での実行時に返されるプロセスハンドル
(DBE_REMOTE)

s.lStartPeriod = lPer& '抽出する最初の期間 (0から始まる)
s.lEndPeriod = lPer& '抽出する最後の期間 (0から始まる)
s.sigAcctConv = HYP_NONE '勘定科目変換表記号
s.sigCat = sigCat& 'データ種別記号
s.sigName = sigEntity& 'エンティティ記号
s.sigNameConv = HYP_NONE 'エンティティ変換表記号
s.sigParent = HYP_NONE '親 (wDSMTypeがID_REGULAR以外の場合)

s.sigNameList = HYP_NONE 'エンティティ一覧
s.fnEnumAcct = 0 '勘定科目を列挙する関数のアドレス

s.lParamEnumAcct = 0 'fnEnumAcct関数の引数
s.fnStatusCB = 0 'ステータス情報のコールバック関数のアドレス

s.lParamStatus = 0 'fnStatusCB関数の引数
s.pApiDFA = EntGetVarAddr(apiS) 'APISTRUCTのアドレス。
apiStruct内の記号フィールドを設定。EntDataExtract()によってapiStructが変更され
る可能性あり。
dsm.dwSize = LenB(dsm)
dsm.wExtractFlags = 0
dsm.wExtractFlags = dsm.wExtractFlags Or DBE_DSM_TRANS
dsm.wExtractFlags = dsm.wExtractFlags Or DBE_DSM_PROP
dsm.wExtractFlags = dsm.wExtractFlags Or DBE_DSM_CONTRIB

```

```

dsm.wExtractFlags = dsm.wExtractFlags Or DBE_DSM_ELIM
dsm.wExtractFlags = dsm.wExtractFlags Or DBE_DSM_TOPADJ
dsm.cSeperator = Asc("|")
dsm.bXAFormat = 1
dsm.bAllDeps = 0
dsm.bAllImmedDeps = 0
'エンティティ一覧記号を設定
dsm.iSelectedDependents = 4
sigArr(0) = 14
sigArr(1) = 15
sigArr(2) = 16
sigArr(3) = 18
dsm.lpSelectedDepSigs = EntGetVarAddr(sigArr(0))
ret% = EntDSMDataExtractVB(hSelect&, s, szFile$, sigAcctList&,
    AddressOf DataExtractCallback, 0, dsm)
ret% = MsgBox("Data extracted, return code = " & str(ret%),
    vbOKOnly, "データ抽出")
'
If Len(acctList$) Then
    hSelect& = EntUnSelect0(hSelect&, HYP_ID_ACCTLIST Or
        HYP_ID_ASSOC, HYP_NONE, 0, ret%, 0)
End If
Unload Me
End Sub

```

C 言語では次の形式を使用します。

```

short WINAPI EntDSMDataExtractVB(HSELECT hSelect, LPDBEXTRACTSTRUCT
pArgs, LPCSTR szExtractFile, SIGNA sigList, CALLBACKVB lpfnCallback, LPARAM
lParam, LPDSMDBEXTRACTSTRUCT lpDsm);

```

## EntEntityAsk( ) - エンティティの選択

この関数は、エンティティ選択ボックスを表示し、選択されたエンティティを取得するために使用されます。[キャンセル] を選択した場合、エンティティは取得されません。

次の形式を使用します。

```

Declare Function EntEntityAsk Lib "heaccess.dll" Alias "_EntEntityAsk@8" (ByVal hSelect
As Long, ByVal szRetbuf As String) As Integer

```

### 変数 説明

hSelect 選択された表のハンドル

szRetbuf 選択されたエンティティの ID を返すバッファ。バッファのサイズは、少なくとも HYP\_SIZEFULLNAME+1 (C 言語では SIZEFULLNAME+1) 文字にする必要があります。

戻りコード：

コード	説明
0	成功

コード	説明
NONE	エラー発生

例：

```
szEntity$ = Space(HYP_SIZEFULLNAME + 1)
ret% = EntEntityAsk(hSelect&, szEntity$)
If ret% = 0 Then szEntity = CToBStr(szEntity$)
```

C 言語では次の形式を使用します。

```
short WINAPI EntDSMDDataExtract(HSELECT hSelect, LPDBEXTRACTSTRUCT pArgs,
SIGNA sigList, CALLBACKVB lpfnCallback, LPARAM lParam,
LPDSMDBEXTRACTSTRUCT lpDsm);
```

## EntEntityListAsk( ) - エンティティ一覧の選択

この関数は、エンティティ一覧選択ボックスを表示し、選択されたエンティティ一覧を取得するために使用されます。[キャンセル] を選択した場合、エンティティ一覧は取得されません。

次の形式を使用します。

```
Declare Function EntEntityListAsk Lib "heaccess.dll" Alias "_EntEntityListAsk@8" (ByVal
hSelect As Long, ByVal szRetbuf As String) As Integer
```

### 変数 説明

hSelect 選択された表のハンドル

szRetbuf 選択されたエンティティ一覧の ID を返すバッファ。バッファの長さは、少なくとも HYP\_SIZELABEL+1 (C 言語の場合は SIZENAME+1) 文字にする必要があります。

戻りコード：

コード	説明
0	成功
NONE	エラー発生

例：

```
szEntityList$ = Space (HYP_SIZELABEL +1)
ret% = EntEntityListAsk(hSelect&, szEntityList$)
If ret% = 0 Then szEntityList$ = CToBStr(szEntityList$)
```

C 言語では次の形式を使用します。

```
short WINAPI EntEntityListAsk(HSELECT hSelect, LPSTR szRetbuf);
```

## EntEntityListLock( ) - データの保護

この関数は、エンティティ一覧のデータを保護します。EntEntityListLock() はオプションのコールバック関数を使用します。API ではこの関数を呼び出して、ルー

チンのステータスを報告します。EntEntityListLock()を呼び出す前に、デフォルトのいずれかの表を選択する必要があります。

次の形式を使用します。

**Declare Function EntEntityListLock Lib "heaccess.dll" Alias "\_EntEntityListLock@12" (ByVal hSelect As Long, lpLockData As EListLockStruct, ByVal lpApiStruct As ApiStruct) As Integer**

変数	説明
hSelect	選択された表のハンドル
lpLockData	ルーチンの引数を含んでいる構造体
lpLockData.dwSize	この構造体のサイズ
lpLockData.sigNameList	保護するエンティティ一覧の記号
lpLockData.dwFlags	オプションのビットフラグ。サーバの処理を示すのは現在 DB_REMOTE のみです。
lpLockData.sigCat	データ種別記号
lpLockData.lStartPeriod	ゼロベースの開始期間
lpLockData.lEndPeriod	ゼロベースの終了期間
lpLockData.sigNameOpenDatafile	データファイルが開いており、この記号では読み取り／書き込みになっていて、API ではこの記号のデータファイルを開く必要がないことを示します。
lpLockData.fnCallBack	ステータスのコールバック関数または NULL。これは、サーバ上で実行していない場合にのみ使用します。
lpLockData.lParam	fnCallBack の引数
lpLockData.lNthStatusCallback	API は、n 番目のエンティティごとに fnCallBack を呼び出します。
lpLockData.hProcess	[タスクステータス] ウィンドウプロセスのハンドル。これは、サーバ上での実行時にプロセスハンドルを返します。
lpApiStruct	NULL。この引数は API の互換性を提供するためのものです。

戻りコード：

コード	説明
0	成功
NONE	エラー発生
LOAD_CHECK_ERRORLOG (2)	エラーが発生しており、アプリケーションのエラーログに示されています。

C 言語では次の形式を使用します。

**Short WINAPI EntEntityListLock(HSELECT hSelect, LPELISTLOCKSTRUCT lpLockData, LPAPISTRUCT lpApiStruct);**

例：

```
Private Sub EntityListLock(sigCat&, lStartPeriod&, lEndPeriod&,
sigEntityList&, bServer As Boolean)
hSelect& = HypGethSelect(frmMain.ghRApp%)
Dim ELockStruct As EListLockStruct
    ELockStruct.dwSize = LenB(ELockStruct)
    ELockStruct.sigNameList = sigNameList&
    ELockStruct.dwFlags = 0
    ELockStruct.sigCat = sigCat&
    ELockStruct.lStartPeriod = lStartPeriod&
    ELockStruct.lEndperiod = lEndPeriod&
    ELockStruct.lNthStatusCallBack = 5 '5番目のエンティティごとにコールバック
ク
    ELockStruct.sigNameOpenDatafile = HYP_NONE
    ELockStruct.fnCallBack = FuncPtrToLong(AddressOf
cbEntityListLock)
    ELockStruct.lParam = 666 '表示用のダミーパラメータ
    ELockStruct.hProcess = 0
If bServer Then
    ELockStruct.dwFlags = ELockStruct.dwFlags Or DB_REMOTE
End If
If chkUnlock.Value Then
    rc% = EntEntityListUnlock0(hSelect&, ELockStruct, 0)
Else
    rc% = EntEntityListLock0(hSelect&, ELockStruct, 0)
End If

End Sub

Public Function cbEntityListLock(ByVal szCategory As Long, _
    ByVal szName As Long, ByVal lStartPer As Long, _
    ByVal lEndPer As Long, ByVal lCurrPer As Long, _
    ByVal lNumEntities As Long, _
    ByVal lCurEntity As Long, _
    ByVal lParam As Long) As Integer
    Dim szCaption$
    Dim ret%, wLen%
    Dim szMsg As String
    Dim strCat$, strName$
    wLen% = EntVGetCStrLen(szCategory)
    strCat$ = Space(wLen%)
    ret% = EntVBCopyStr(strCat$, szCategory, wLen%)
wLen% = EntVGetCStrLen(szName)
    strName$ = Space(wLen%)
    ret% = EntVBCopyStr(strName$, szName, wLen%)
    szCaption$ = "Now Processing at period " & str(lCurrPer) & "
    " & str(lCurEntity) _ & "th entity of " &
    str(lNumEntities) & " Lparam = " & str(lParam)
    frmEntityListLock.List1.AddItem (szCaption$)
    ret% = MsgBox(strName$, vbOKOnly, strCat$ & " from " &
    str(lStartPer) & " to " & str(lEndPer))
    ret% = 0
End Function
```

**注：** この例における EntEntityListLock0 () と EntEntityListUnlock0 () の宣言は、EntEntityListLock () と EntEntityListUnlock () に似ていますが、apiStruct の代わ

りに 0 を渡すことができるように調整されています。詳しくは、[16 ページ](#)の「[Visual Basic プログラミング上の注意](#)」を参照してください。

## EntEntityListUnlock( ) - データの保護解除

この関数は、エンティティ一覧のデータの保護を解除します。EntEntityListUnlock() はオプションのコールバック関数を使用します。API ではこの関数を呼び出して、ルーチンのステータスを報告します。EntEntityListUnlock()を呼び出す前に、デフォルトのいずれかの表を選択する必要があります。

次の形式を使用します。

```
Declare Function EntEntityListUnlock Lib "heaccess.dll" Alias
"_EntEntityListUnlock@12" (ByVal hSelect As Long, lpLockData As EListUnlockStruct,
ByVal lpApiStruct As ApiStruct) As Integer
```

変数	説明
hSelect	選択された表のハンドル
lpUnlockData	ルーチンの引数を含んでいる構造体
lpUnlockData.dwSize	この構造体のサイズ
lpUnlockData.sigNameList	保護を解除するエンティティ一覧の記号
lpUnlockData.dwFlags	オプションのビットフラグ。サーバの処理を示すのは現在のところ DB_REMOTE のみです。
lpUnlockData.sigCat	データ種別記号
lpUnlockData.lStartPeriod	ゼロベースの開始期間
lpUnlockData.lEndPeriod	ゼロベースの終了期間
lpUnlockData.sigNameOpenDatafile	データファイルが開いており、この記号では読み取り／書き込みになっていて、API ではこの記号のデータファイルを開く必要がないことを示します。
lpUnlockData.fnCallBack	サーバで実行していない場合はステータスを求めるコールバック関数、または NULL。これは、サーバ上で実行していない場合にのみ使用します。
lpUnlockData.lParam	fnCallBack の引数
lpUnlockData.lNthStatusCallback	API は、n 番目のエンティティごとに fnCallBack を呼び出します。
lpUnlockData.hProcess	[タスクステータス] ウィンドウプロセスのハンドル。これは、サーバ上での実行時にプロセスハンドルを返します。
lpApiStruct	NULL。この引数は使用されていません。

戻りコード：

コード	説明
0	成功
NONE	エラー発生
LOAD_CHECK_ERRORLOG (2)	エラーが発生しており、アプリケーションのエラーログに示されています。

C 言語では次の形式を使用します。

```
Short WINAPI EntEntityListUnLock(HSELECT hSelect, LPELISTLOCKSTRUCT
lpLockData, LPAPISTRUCT lpApiStruct);
```

[167 ページの「EntEntityListLock\(\) - データの保護」](#)の例を参照してください。

## EntEnum( ) - 表内のレコードの列挙

この関数はユーザ定義のコールバック関数を使用して、指定された表内のすべてのレコードの記号を列挙します。表 wTabId 内のレコードごとに、EntEnum( )は指定されたコールバック関数にレコード記号を渡します。コールバック関数から非ゼロ値を返すことによって列挙を打ち切ることができます。

多くの表では、API の u\_ApiEnum.dwMask を表に適した特定の定数に設定することで、列挙をフィルタにかけることができます。定数は、ToolInc.h 内を検索してください。

**ヒント：** 特定の表に apistruct が必要であるか、列挙をフィルタにかけ除き、API の引数には NULL を渡してください。そうしないと、列挙がフィルタにかけられており、誤って指定された基準を満たすレコードが見つからないものと API で認識される場合があります。

次の形式を使用します。

```
Declare Function EntEnum Lib "heaccess.dll" Alias "_EntEnum@24" (ByVal hSelect As
Long, ByVal wTabId As Integer, ByVal sigKey As Long, ByVal lpCallback As Long, ByVal
lParam As Long, apiS As APISTRUCT) As Integer
```

変数	説明
hSelect	選択された表のハンドル
wTabId	記号を列挙する表。欠けているリンクにリストされている有効な表 ID を含めることができます。
sigKey	NONE、または関連する表のキー。wTabId が関連する表の場合、これは必須のキーです。詳しくは、 <a href="#">340 ページの「関連する表」</a> を参照してください。
lpCallBack	ユーザ定義のコールバック関数のポインタ。詳しくは、 <a href="#">319 ページの「CALLBACKAPI」</a> を参照してください。
lParam	ユーザ定義のコールバック関数に返されるパラメータ (lpCallBack)

## 変数 説明

apiS      NULL、または apiStruct のポインタ

戻りコード :

コード	説明
0	成功
NONE	エラー発生
もう一方の値	コールバック関数から返される値

例 :

```
rc% = EntEnum0(hSelect&, HYP_ID_ACCTLISTENTRY, SigList&, AddressOf  
MyCallBackFunc, 0, 0)
```

EntEnum0()は、EntEnum()と同じように宣言されますが、apiStruct の代わりに 0 を渡すことができるように調整されます。詳しくは、[16 ページの「Visual Basic プログラミング上の注意」](#)を参照してください。

C 言語では次の形式を使用します。

```
short WINAPI EntEnum(HSELECT hSelect, short wTabId, SIGNA sigKey,  
CALLBACKAPI lpCallBack, LONG lParam, LPAPISTRUCT pApiS);
```

次の各項では、EntEnum()で特定の Hyperion Enterprise 表を使用する方法について説明します。

## EntEnum()および ID\_ACCTLISTENTRY（勘定科目一覧入力表）

一覧が連動一覧の場合、コールバック関数に渡される記号は勘定科目記号（table ID\_ACCOUNTS）になります。ただし、勘定科目一覧が固定一覧の場合、記号は勘定科目一覧入力表（ID\_ACCTLISTENTRY）内のレコードの記号になります。勘定科目記号を取得するには、クエリ属性 ENTRY\_SIG（表 ID\_ACCTLISTENTRY）を使用して EntQueryEx()を呼び出します。一覧が連動一覧であるか、固定一覧であるかを確認するには、クエリ属性 LIST\_TYPE を使用して勘定科目一覧表（ID\_ACCTLIST）の EntQueryEx()を呼び出します。[85 ページの「HypEnumAcctListEntriesEx\(\) - 勘定科目一覧の勘定科目の列挙」](#)の例を参照してください。

勘定科目一覧エントリを列挙するとき、コールバック関数の sigRecd および sigKey 引数は逆になります。最初の記号は一覧記号です。2 番目の記号は固定一覧のエントリ記号または連動一覧の勘定科目記号です。

EntEnum()を呼び出して連動勘定科目一覧の表 ID\_ACCTLISTENTRY を列挙した場合の結果は次のとおりです。

- コールバック関数によって ACCTLIST\_DONTEXPANDSUBS（Toollnc.h で定義）が返された場合、EntEnum()ではその勘定科目（コールバック関数に渡された勘定科目）のサブ勘定科目（存在する場合）は列挙しません。

- コールバック関数に渡された `apiStruct`（存在する場合）の `u_ApiEnum.dwMask` フィールドは、インデントレベルを示します。0 は主要勘定科目、1 は第 1 レベルサブ勘定科目、2 は第 2 レベル（サブサブ勘定科目）を表します。

## EntEnum( )と ID\_CODES

表 ID\_CODES を列挙するときは、勘定科目コードまたはエンティティコードをフィルタにかけることができます。このオプションを使用するには、`apiStruct` を初期化し、その `u_ApiEnum.dwMask` フィールドを `CODE_TYPE_ACCOUNT` または `CODE_TYPE_ENTITY` に設定する必要があります。

例：

```
EntInitApiStruct(hApp, apiS)
apiS.u_ApiEnum.dwMask =
CODE_TYPE_ACCOUNT                                     '他の選択肢は
CODE_TYPE_ENTITY。
nRet = EntEnum(hSelect, HYP_ID_CODES, HYP_NONE, AddressOf
EnumCodesCallBack, lParam, apiS)
```

結果をフィルタしない場合は、`apiStruct` を `EntEnum( )` に渡さないでください。  
`HypEnumEx( )` を使用して ID\_CODES 表を列挙することもできます。

## EntEnum()と ID\_DATAFILE

`EntEnum( )` を使用してデータファイル表内のレコードを列挙するときは、通常セキュリティがチェックされ、完全に空白のレコード（期間内にデータがないレコード）と、勘定科目が連動表示勘定科目に変更されている古いレコードがスキップされます。

`apiStruct` の `u_ApiEnum.dwMask` フィールドは、特定の種類のレコードをスキップするためのオプションのフィルタを指定します。このフィールドは 0 または以下の定数のいずれかに指定します（`ToolInc.h` の定義を参照）。ビット単位の OR 演算子を使用すると、表 42 のオプションを自由にいくつでも組み合わせることができます。

表 42 EntUnum()のオプション

オプション	説明
DF_FILTEROUT_INP	ロジックステートメントのない勘定科目をスキップします。ロジックによって勘定科目が入力勘定科目となるかどうかのロジックステートメントのテストは行われません。
DF_FILTEROUT_CALC	ロジックステートメントによって勘定科目が入力勘定科目とならない限り、ロジックステートメントのある勘定科目をスキップします。サブ勘定科目の合計である主要勘定科目や、エンティティがグローバルエンティティでない場合のグローバル勘定科目グループ内の勘定科目などの、他の種類の算出勘定科目はスキップされません。
DF_FILTEROUT_LINP	勘定科目を入力勘定科目にするロジックステートメントのある勘定科目をスキップします。

オプション	説明
DF_FILTEROUT_LOGIMP	サブ勘定科目の合計である主要勘定科目などの、暗黙的ロジックのある勘定科目をスキップします。
DF_FILTEROUT_NODATA	指定されている期間にデータがない勘定科目をスキップします。apiStruct の IStartPeriod および IEndPeriod フィールドを設定する必要があります。
DF_FILTEROUT_NODATA_FROM_PERO	最初の期間（期間 0）から apiStruct の IEndPeriod フィールドに指定されている期間までの間にデータがない勘定科目をスキップします。これは DF_FILTEROUT_NODATA と同じですが、apiStruct の IStartPeriod フィールドにどのようなデータが含まれているかにかかわらず、開始期間には 0 が使用されます。

EntEnum()は、EntEnum()に渡したものと同一 sigKey、IParam および apiStruct 引数をコールバック関数に渡します。

## EntEnum( )と ID\_INTCODET（会社間詳細表）

sigKey 引数を、表 ID\_ICSET に設定されている会社間照合記号に設定します。これによって、そのセット内の詳細レコードが列挙されます。sigKey を HYP\_NONE（C 言語では NONE）に設定し、すべてのセットの詳細レコードを列挙することもできます。

## EntEnum( )と ID\_JOURNALS（仕訳帳表）

sigKey 引数は HYP\_NONE（C 言語では NONE）に設定する必要があります。表 43 に示すフィールドを使用して apiStruct を設定してください。

表 43 EntEnum() apiStruct Fields - 仕訳帳表

フィールド	説明
sigCat	データ種別記号
IStartPeriod	期間番号
IEndPeriod	期間番号（IStartPeriod と同じ）
u_ApiEnum.dwMask	すべてを列挙する場合は 0

指定したデータ種別と期間の仕訳帳をすべて列挙するには、apiStruct のマスクフィールド（u\_ApiEnum.dwMask）をデフォルト値の 0 に設定します。特定の基準を満たす仕訳帳のみを列挙することもできます。その場合には、列挙に含める仕訳帳を示すマスクフィールドを設定します。マスクは、仕訳帳ステータス、仕訳帳種別、および仕訳帳属性の 3 つの部分で構成されています。列挙のマスクを指定する場合、マスクにはこれらの各部分を含める必要があります。例えば、仕訳帳のステータスをフィルタにかける場合、種別と属性のマスクも指定する必要があります。マスクが指定されていて、そのコンポーネントのいずれかがゼロの場合、列挙は実行されません。

例：

```

Dim apiS As APISTRUCT
Call EntInitApiStruct (hApp&, apiS)
apiS.sigCat = sigCat
apiS.lStartPeriod = PeriodNum&
apiS.lEndPeriod = PeriodNum&
'すべての仕訳帳を列挙（このデータ種別と期間）
apiS.u_ApiEnum.dwMask = 0
ret% = EntEnum(hSelect&, HYP_ID_JOURNALS, HYP_NONE,
    callback.ProcAddress, lParam&, apiS)
'転記され、貸借一致しているすべての通常仕訳帳を列挙
apiS.u_ApiEnum.dwMask = JOURMASK_STATUS_POSTED Or
JOURMASK_TYPE_REGULAR Or JOURMASK_ATTRIB_BALANCED
ret% = EntEnum(hSelect&, HYP_ID_JOURNALS, HYP_NONE,
    callback.ProcAddress, lParam&, apiS)
'指定されているデータ種別と期間のすべての自動逆仕訳帳を列挙
apiS.u_ApiEnum.dwMask = JOURMASK_TYPE_AUTOREVERSING Or
JOURMASK_ALL_STATUS Or JOURMASK_ALL_ATTRIB
ret% = EntEnum(hSelect&, HYP_ID_JOURNALS, HYP_NONE,
    callback.ProcAddress, lParam&, apiS)
'このマスクは機能しません。すべての仕訳帳に属性があり、

属性マスクがゼロに設定されているので、どの仕訳帳もテストに合格しません。
apiS.u_ApiEnum.dwMask = JOURMASK_TYPE_AUTOREVERSING or
JOURMASK_ALL_STATUS

```

### EntEnum( )と ID\_JOURNAL\_DETAIL（仕訳帳詳細表）

sigKey 引数を、詳細が適用される仕訳帳の記号に設定します。これによって、その特定の仕訳帳の詳細表にあるエントリがすべて列挙されます。[表 44](#) に示すフィールドを使用して apiStruct を設定してください。

**表 44** EntEnum() apiStruct Fields - 仕訳帳詳細表

フィールド	説明
sigCat	データ種別記号
u_ApiEnum.sigJournal	仕訳帳または仕訳帳の最初の詳細レコードの記号
u_ApiEnum.sJournalTableID	sigJournal の適用先の表 ID
u_ApiEnum.dwMask	未使用（フィルタなし）

### EntEnum() apiStruct Fields - 仕訳帳詳細表

例：

```

Dim apiS As APISTRUCT
Call EntInitApiStruct (hApp&, apiS)
apiS.sigCat = sigCat
apiS.u_ApiEnum.sigJournal = sigJournal
apiS.u_ApiEnum.sJournalTableID = HYP_ID_JOURNALS
ret% = EntEnum(hSelect&, HYP_ID_JOURNAL_DETAIL, sigJournal,
    callback.ProcAddress, lParam&, apiS)
'または...
apiS.u_ApiEnum.sigJournal = sigFirstJournalDetail

```

```
apiS.u_ApiEnum.sJournalTableID = HYP_ID_JOURNAL_DETAIL
ret% = EntEnum(hSelect&, HYP_ID_JOURNAL_DETAIL, sigJournal,
    callback.ProcAddress, lParam&, apiS)
```

## EntEnum( )と ID\_JOURNAL\_HISTORY\_DETAIL（仕訳帳履歴詳細表）

sigKey 引数を、履歴が適用される仕訳帳の記号に設定します。これによって、その特定の仕訳帳の仕訳帳履歴表にあるエントリがすべて列挙されます。表 45 に示すフィールドを使用して apiStruct を設定してください。

**表 45** EntEnum() apiStruct Fields - 仕訳帳履歴詳細表

フィールド	説明
sigCat	データ種別記号
u_ApiEnum.sigJournal	仕訳帳または最初の履歴の記号
u_ApiEnum.sJournalTableID	sigJournal の適用先の表 ID
u_ApiEnum.dwMask	未使用（フィルタなし）

詳しくは、110 ページの「apiStruct 構造体の使用」を参照してください。

任意の仕訳帳の記号がその表 ID を使用して apiStruct で指定されている場合は、その仕訳帳のエントリがすべて列挙されます。任意の仕訳帳の最初の履歴レコードの記号が apiStruct の u\_ApiEnum.sigJournal に指定されている場合は、その仕訳帳のエントリがすべて列挙されます。u\_ApiEnum.sigJournal が NONE に設定されている場合は、表全体が列挙されます。

## EntEnum( )と ID\_JOURNAL\_TEMPLATES（仕訳帳テンプレート表）

sigKey 引数は HYP\_NONE（C 言語では NONE）に設定する必要があります。以下の表に示すフィールドを使用して apiStruct を設定してください。

**表 46** EntEnum()の apiStruct フィールド - 仕訳帳テンプレート表

フィールド	説明
sigCat	データ種別記号
u_ApiEnum.dwMask	すべてを列挙する場合は 0

詳しくは、110 ページの「apiStruct 構造体の使用」を参照してください。

指定したデータ種別の仕訳帳テンプレートをすべて列挙するには、apiStruct の u\_ApiEnum.dwMask フィールドをデフォルト値の 0 に設定します。特定の基準を満たす仕訳帳テンプレートのみを列挙することもできます。この場合は、列挙に含める仕訳帳テンプレートを示すマスクフィールドを設定します。マスクは、テンプレート種別、仕訳帳種別および仕訳帳属性の 3 つの部分で構成されています。列挙のマスクを指定する場合は、マスクにこれらの各部分を含める必要があります。例えば、テンプレート種別をフィルタにかける場合、種別と属性のマスクも

指定する必要があります。マスクが指定されていて、そのコンポーネントのいずれかが 0 の場合、列挙は実行されません。

例：

```
Dim apiS As APISTRUCT
Call EntInitApiStruct (hApp&, apiS)
apiS.sigCat = sigCat
'データ種別のすべての仕訳帳テンプレートを列挙
apiS.u_ApiEnum.dwMask = 0
ret% = EntEnum(hSelect&, HYP_ID_JOURNAL_TEMPLATES,
    HYP_NONE, callback.ProcAddress, lParam&, apiS)
'種別が通常で、貸借一致しているすべての標準テンプレートを列挙
apiS.u_ApiEnum.dwMask = JOURMASK_TEMPTYPE_STANDARD or
    JOURMASK_TYPE_REGULAR or JOURMASK_ATTRIB_BALANCED
ret% = EntEnum(hSelect&, HYP_ID_JOURNAL_TEMPLATES, HYP_NONE,
    callback.ProcAddress, lParam&, apiS)
'指定されたデータ種別のすべての経常テンプレートを列挙
apiS.u_ApiEnum.dwMask = JOURMASK_TEMPTYPE_RECURRING or
    JOURMASK_ALL_TYPE or JOURMASK_ALL_ATTRIB
ret% = EntEnum(hSelect&, HYP_ID_JOURNAL_TEMPLATES, HYP_NONE,
    callback.ProcAddress, lParam&, apiS)
'このマスクは機能しません。すべてのテンプレートがテンプレート種別を持っており、
'テンプレート種別のマスクがゼロであるので、どのテンプレートもテストに合格しません。
apiS.u_ApiEnum.dwMask = JOURMASK_ATTRIB_BALANCED or
    JOURMASK_ALL_TYPE
```

## EntEnum( )と ID\_NAMELISTENTRY（エンティティ一覧入力表）

エンティティ一覧のエントリを列挙するとき、コールバック関数の sigRecd および sigKey 引数は逆になります。最初の記号は一覧記号です。2 番目の記号はエントリ記号です。

連動組織を使用するように設定されているアプリケーションの連動エンティティ一覧を列挙するには、次のようにします。apiStruct を EntEnum()に渡す場合は、apiStruct に、以下の表に示すフィールドを設定する必要があります。apiStruct を渡さない場合、API 関数では現在のデータ種別と現在の期間を使用します。

フィールド	説明
sigCat	データ種別記号。これが NONE の場合、API では現在のデータ種別を使用します。
lStartPeriod	開始期間（0 から始まる）
lEndPeriod	終了期間（0 から始まる）
u_ApiQry.bSupSubNames	サブエンティティを非表示にする場合は True（1）、サブエンティティを展開する場合は false（0）。  ヒント： 確かでない場合は 0 に設定してください。

固定エンティティ一覧では、apiStruct を EntEnum()に渡す場合、その u\_ApiEnum.uUseFixedListCriteria フィールドを ENUM\_FIXEDLIST\_CRITERIA（ToolInc.h に定義されている定数）に設定して、固定一覧内のエンティティを列挙する代わりに、固定一覧の作成基準に一致するエンティティを列挙できます。

## EntEnum( )と ID\_NAMES（エンティティ表）

すべてのエンティティを列挙する代わりに、エンティティの親（直属の親だけでなく、そのすべての親）を列挙できます。これを行うには、sigKey 引数を、列挙する親のエンティティ記号に設定します。これを行うには、u\_ApiEnum.dwMask フィールドが ENUM\_PARENTS\_ABOVE に設定されている apiStruct を設定します。

連動組織を使用するように設定されているアプリケーションには、以下の表に示すフィールドを使用してさらに apiStruct を設定します。

フィールド	説明
sigCat	データ種別記号（データ種別の期間数は、内部バッファのサイズを決定するために使用されます）。
lStartPeriod	開始期間
lEndPeriod	終了期間
lpseStatus	ステータス情報を返すためのバッファのアドレス。バッファはデータ種別の開始期間（lStartPeriod）から終了期間（lEndPeriod だけではない）までの各期間の整数（C 言語では short）を格納するのに十分な大きさにする必要があります。

これは、任意の範囲の期間の親をすべて列挙します。この列挙によって、コールバック関数に apiStruct を渡すときに、apiStruct 内の一部のフィールドが上書きされます。lStartPeriod フィールドは、エンティティが、指定されたエンティティの親である最初の期間を示します。子を持つ各期間のステータスバッファには 1 の値が含まれています（エンティティは 1 つの期間では親となり、別の期間では最下位エンティティとなることが可能なためです）。

詳しくは、[110 ページの「apiStruct 構造体の使用」](#)を参照してください。

## EntEnum( )と ID\_NODES（ノード表）

sigKey 引数を、開始点となる最上位ノードまたはエンティティの記号に設定します。これを NONE（Visual Basic では HYP\_NONE）に設定すると、指定した期間でアクティブになっているノードがすべて列挙されます。以下の表に示すフィールドを使用して apiStruct を設定してください。

フィールド	説明
u_ApiEnum.bNameSig	sigKey がエンティティ記号の場合は 1（TRUE）、sigKey がノード記号の場合は FALSE
u_ApiEnum.dwMask	次のコードを使用します。 <ul style="list-style-type: none"><li>● 下位の子をすべて列挙する場合は ENUM_ALL</li><li>● 直属の子のみを列挙する場合は ENUM_DEP（ENUM_IMMED_DEPS）</li><li>● sigKey 以下の親のみを列挙する場合は ENUM_CON（ENUM_PARENT）</li><li>● 下位の最下位エンティティのみを列挙する場合は ENUM_BAS（ENUM_BASE）</li><li>● 未定義のエンティティを列挙する場合は ENUM_UNOWNED</li></ul>

フィールド	説明
	<ul style="list-style-type: none"> <li>すべてのサブエンティティを含める場合は ENUM_SUBNAME</li> </ul>

連動組織を使用するようにアプリケーションが設定されている場合は、以下の表に示すフィールドを使用してさらに `apiStruct` を設定します。

フィールド	説明
<code>sigCat</code>	データ種別記号
<code>lStartPeriod</code>	開始期間
<code>lEndPeriod</code>	終了期間
<code>lpseStatus</code>	NULL、またはステータスバッファのアドレス。バッファは、1つの期間あたり1バイト（char）を格納するのに十分な大きさにする必要があります。

`lpseStatus` フィールドが NULL でない場合、`EntEnum()` はノードがアクティブになっている各期間のステータスバッファにゼロ以外の値（TRUE）を挿入し、ノードが非アクティブな各期間に 0（FALSE）を挿入します。

詳しくは、[110 ページの「apiStruct 構造体の使用」](#)を参照してください。

## EntEnum() と ID\_RPTVIEW（レポート表示形式表）

期間単位に適切なレポート表示形式のみを列挙するには、`sigKey` 引数をその期間単位の記号（`FREQ_MONTH` など）に設定します。すべてのレポート表示形式を列挙するには、`sigKey` を `HYP_NONE`（C 言語の場合は `NONE`）に設定します。

## EntEnum() と ID\_ROLLOVER（期別替表）

期別替セットの詳細レコードを列挙するには、表 `ID_ROLLSET` に設定されている期別替セットの記号に `sigKey` 引数を設定します。`sigKey` を `HYP_NONE`（C 言語の場合は `NONE`）に設定してすべての期別替セットの詳細レコードをすべて列挙することもできます。

## EntEnum() と ID\_SECGRPTAB（セキュリティグループ表）

`EntEnum()` で `ID_SECGRPTAB` を使用するには、`apiStruct` を設定します。詳しくは、[110 ページの「apiStruct 構造体の使用」](#)を参照してください。

ほとんどの表とは異なり、セキュリティグループ表のレコードをすべて列挙することはできません。次のいずれかの列挙オプションを選択します。

- ユーザが属しているグループを列挙するには、ユーザ記号（表 `ID_SECUSERTAB` 内）を `sigKey` 引数として渡します。`apiStruct` 内の `u_ApiEnum.bSecUserCode` フィールドを `SECURITY_USER`（`TOOLINC.H` ファイル内で定義）に設定します。
- ユーザグループのメンバを列挙するには、ユーザ表（`ID_SECUSERTAB`）内のグループの記号を `sigKey` 引数として渡します。`apiStruct` 内の

u\_ApiEnum.bSecUserGroupCode フィールドを SECURITY\_GROUP (TOOLINC.H ファイル内で定義) に設定します。

セキュリティグループ表を列挙するために EntEnum( ) を呼び出す前に、ユーザ表 (ID\_SECUSERTAB) を選択する必要があります。表を個別に選択する代わりに、OR 演算を使用して、表 ID\_SECURITY とそれに関連付けられているセキュリティ表をすべて選択することをお勧めします。

コールバック関数に渡す記号は、表 ID\_SECCRPTAB 内のセルの記号です。ユーザまたはグループ記号の取得に使用できるクエリ属性については、[394 ページの「ID\\_SECCRPTAB \(セキュリティグループ表\) クエリ属性」](#)を参照してください。

## EntEnum( ) と ID\_SECRIGHTS (セキュリティ権表)

EntEnum() で ID\_SECRIGHTS を使用するには、apiStruct を設定する必要があります。ユーザに明示権限があるクラスを列挙するには、ユーザ記号 (表 ID\_SECUSERTAB 内) を sigKey 引数として渡します。apiStruct の u\_ApiEnum.bSecUserGroupCode フィールドは SECURITY\_USER に設定します。

セキュリティクラス内のユーザを列挙するには、sigKey 引数をクラス記号 (表 ID\_SECCLASS) に設定します。apiStruct 内の u\_ApiEnum.bSecUserGroupCode フィールドは SECURITY\_CLASS に設定します。

apiStruct 内の u\_ApiEnum.bSecUserGroupCode フィールドを SECURITY\_ALLTYPES に設定することもできます。この場合、sigKey 引数は通常は HYP\_NONE (C 言語では NONE) にします。

## EntEnum( ) と ID\_SECTASK (セキュリティタスク表)

表 ID\_SECTASKFILTER 内の特定のセキュリティタスクフィルタによって指定されているタスクのみを列挙するには、sigKey 引数をセキュリティタスクフィルタの記号に設定します。すべてのタスクを列挙するには、sigKey 引数を HYP\_NONE (C 言語では NONE) に設定します。

## EntEnum( ) と ID\_SECUSERTAB (セキュリティユーザ表)

EntEnum( ) で ID\_SECUSERTAB を使用するには、apiStruct を設定します。apiStruct 内の u\_ApiEnum.bSecUserGroupCode フィールドは、次のいずれかのコードに設定します。

- (グループではなく) ユーザのみを列挙する場合は SECURITY\_USER
- グループのみを列挙する場合は SECURITY\_GROUP
- すべてのレコード (ユーザとグループ) を列挙する場合は SECURITY\_ALLTYPES

詳しくは、[110 ページの「apiStruct 構造体の使用」](#)を参照してください。

## EntEnum( )と ID\_SHARES (株式表)

連動組織を使用するようにアプリケーションが設定されている場合は、データ種別記号を sigKey として渡します。

以下の表に示すフィールドで apiStruct を設定してください。

フィールド	説明
sigName	エンティティ記号または NONE
u_ApiEnum.dwMask	次の定数のいずれかを使用します。 <ul style="list-style-type: none"><li>● 特定のエンティティが所有している株式を列挙する場合は SHARES_ WHOIOWN。sigName フィールドはエンティティの記号に設定します。</li><li>● 他のエンティティが所有している特定のエンティティの株式を列挙する場合は SHARES_ WHOOWNSME。sigName フィールドはエンティティの記号に設定します。</li><li>● グローバルエンティティが所有している株式を列挙する場合は SHARES_ OWNER_GLOBAL。sigName は NONE に設定します。</li><li>● いずれかのエンティティが所有している株式を列挙する場合は SHARES_ ALL_ WHOIOWN。sigName は NONE に設定します。</li></ul> <b>注：</b> これらの定数は Toolinc.h ファイルで定義されています。

オプションで、定数 SHARES\_INC\_NON\_INTCO とマスクフィールドで OR 演算を使用して、非会社間エンティティを含めることもできます。これは、SHARES\_ WHOIOWN、SHARES\_ WHOOWNSME および SHARES\_ ALL\_ WHOIOWN のみに関係があります。

連動組織を使用するようにアプリケーションが設定されている場合は、以下の表に示すフィールドを使用してさらに apiStruct を設定します。

フィールド	説明
sigCat	データ種別記号
lStartPeriod	開始期間
lEndPeriod	終了期間

例：

```
Dim apiS As APISTRUCT
Call EntInitApiStruct(hApp&, apiS)
apiS.sigCat& = sigCat&           'データ種別記号
apiS.lStartPeriod = 0             '最初の期間を使用
apiS.lEndPeriod& = apiS.lStartPeriod&
apiS.u_ApiEnum.dwMask = SHARES_ WHOOWNSME
apiS.sigName = sigEntityOwned&
ret% = EntEnum(hSelect&, HYP_ID_SHARES,
               sigCat&,callback.ProcAddress, 0, apiS)
```

詳しくは、[110 ページの「apiStruct 構造体の使用」](#)を参照してください。

## EntEnum( )と ID\_SUBACCTDET（サブ勘定科目詳細表）

sigKey 引数は、通常はサブ勘定科目表（表 ID\_SUBACCTHDR 内）の記号です。ただし、勘定科目記号（表 ID\_ACCOUNTS 内）を sigKey 引数として使用して、その勘定科目のサブ勘定科目を列挙することもできます。この場合は、apiStruct を設定する必要があります。

apiStruct の u\_ApiEnum.bSubAcctDet フィールドを ENUMSAD\_BYACCOUNT に設定し、sigKey 引数が勘定科目記号であることを示します。または、EntEnumSubAcctSig( )を使用します。

詳しくは、[110 ページの「apiStruct 構造体の使用」](#)を参照してください。

## EntEnumApplications( ) - アプリケーションの列挙

この関数は、Hyperion Enterprise の [新規アプリケーション] または [アプリケーションの追加] コマンドを使用するように設定されているすべての Hyperion Enterprise アプリケーションを列挙します。EntEnumApplications( )はコールバック関数を使用して HYPENT.INI ファイルを読み取り、アプリケーションを列挙します。この関数を呼び出す前に、アプリケーションを開く必要はありません。

次の形式を使用します。

```
Declare Function EntEnumApplications Lib "heaccess.dll" Alias
"_EntEnumApplications@8" (ByVal lpCallback As Long, ByVal lParam As Long) As Integer
```

### 変数 説明

lpCallback ユーザ定義のコールバック関数のポインタ。詳しくは、[320 ページの「CALLBACKAPP（または CALLBACK6）」](#)を参照してください。

lParam ユーザ定義のコールバック関数に返されるパラメータ

戻りコード：

コード	説明
0	成功
1	エラーが発生したかアプリケーションが不在
その他	コールバック関数から返される値

例：

```
ret% = EntEnumApplications(AddressOf EnumAppsCallback, lParam%)
.
.
.
' コールバック関数
Public Function EnumAppsCallback(ByVal lAddr As Long, ByVal lParam As Integer) As Integer
' Cの文字列をBの文字列にコピー
wLen% = EntVBGetCStrLen( lAddr&
```

```

szApp$=Space (wLen%)
ret%=EntVBCopyStr(szApp$, lAddr&, wLen%)
If UCASE(szApp$) <> "DEFAULT" Then
    ret% = MsgBox("Application:  " & szApp$)
End If
EnumAppsCallback% = 0    "列挙の続行
End Function

```

C 言語では次の形式を使用します。

```
short WINAPI EntEnumApplications(CALLBACKAPP lpCallback, LONG lParam);
```

## EntEnumOrgEntities( ) - 組織内のノードの列挙

この関数は、ユーザ定義のコールバック関数を使用して、任意の組織内のノードをすべて列挙します。列挙は組織内の任意のノードから開始されます。組織情報はノード表 ID\_NODES に保存されています。EntEnumOrgEntities( )は、組織内のノードを組織にリストされている順番で列挙します。EntQueryEx( )を使用すると、各ノードのエンティティのノード表をクエリできます。

ノードと組織構造体については、[120 ページの「組織構造体と高度な関数」](#)を参照してください。

次の形式を使用します。

```

Declare Function EntEnumOrgEntities Lib "heaccess.dll" Alias
    "_EntEnumOrgEntities@20" (ByVal hSelect As Long, ByVal sigNode As Long, ByVal
    lpCallback As Long, ByVal lParam As Long, ByVal wMask As Integer) As Integer

```

### 変数 説明

hSelect 選択された表のハンドル

sigNode 列挙を開始するノード

lpCallback ユーザ定義のコールバック関数のポインタ。詳しくは、[319 ページの「CALLBACKAPI」](#)を参照してください。

lParam ユーザ定義のコールバック関数に返されるパラメータ

wMask 次のコードを使用します。

- ENUM\_IMMED\_DEPS 直属の子のみを列挙する。
- ENUM\_PARENT sigNode の下位の親エンティティを列挙する。
- ENUM\_BASE 最下位エンティティを列挙する。
- ENUM\_ALL 最下位および親のエンティティおよびサブエンティティを列挙する。
- ENUM\_UNOWNED 未定義のエンティティを列挙する。
- ENUM\_SUBNAME すべてのサブエンティティを含める（これは、OR 演算および他の定数のいずれかとともに使用する必要があります）。

連動組織を使用するようにアプリケーションが設定されている場合、EntEnumOrgEntities()では現在のデータ種別と期間の組織構造体が列挙の対象とみなされます。この情報が必要でない場合は、EntEnumOrgEntities( )の代わりにEntEnum( )を使用してください。ID\_NODES を列挙し、apiStruct の sigCat、

lStartPeriod および lEndPeriod フィールドでデータ種別と期間を指定します。  
 apiStruct の u\_ApiEnu.dwMask フィールドは wMask 値に設定します。  
 u\_ApiEnum.bNameSig フィールドは FALSE (0) に設定します。

戻りコード：

コード	説明
0	成功
NONE	エラー発生
その他	コールバック関数から返される値

例：

```
'組織内のすべての最下位レベルエンティティを列挙
'最上位ノードから開始
ret% = EntQueryEx(hSelect&, HYP_ID_ORGANIZATION, TOPNODE,
    sigOrg&, HYP_NONE, 4, sigTopNode&, 0)
If sigTopNode& <> HYP_NONE Then
    ret% = EntEnumOrgEntities(hSelect&, sigTopNode&,
        AddressOf EnumOrgCB, 0, ENUM_BASE Or ENUM_SUBNAME)
End If
.
.
.
'コールバック関数
Public Function EnumOrgCB(ByVal hSelect&, ByVal sigNode&, ByVal sigKey&,
    ByVal lParam&, apis as ApiStruct) As Integer
    'エンティティIDの取得と表示
    ret% = EntQueryEx(hSelect&, HYP_ID_NODES, NAMESIG, sigNode&,
        HYP_NONE, 4, sigEntity&, 0)
    szEntity$ = Space(HYP_SIZEFULLNAME+1)
    ret% = EntQueryEx(hSelect&, HYP_ID_NAMES, NAME, sigEntity&,
        Len(szEntity$), szEntity$)
    If ret% = 0 Then szEntity$ = CToBStr(szEntity$)
    ret% = MsgBox("Entity: " & szEntity$)
    '列挙を続けるために0を返す
    EnumOrgCB% = 0
End Function
```

C 言語では次の形式を使用します。

```
short WINAPI EntEnumOrgEntities(HSELECT hSelect, SIGNA sigNode,
    CALLBACKAPI lpCallback, LONG lParam, WORD wMask);
```

## EntEnumSubAcctSig( ) - サブ勘定科目の列挙

この関数は、指定された主要勘定科目の一連のサブ勘定科目記号を列挙します。  
 サブ勘定科目詳細表 (ID\_SUBACCTDET) を列挙するには、この関数を EntEnum( )  
 の代わりに使用します。EntEnumSubAcctSig( )は、各サブ勘定科目に対し、勘定科  
 目表 (ID\_ACCOUNTS) 内の勘定科目記号をコールバック関数に渡します。

次の形式を使用します。

Declare Function EntEnumSubAcctSig Lib "heaccess.dll" Alias  
"\_EntEnumSubAcctSig@16" (ByVal hSelect As Long, ByVal sigAcct As Long, ByVal  
lpCallback As Long, ByVal lParam As Long) As Integer

変数	説明
hSelect	選択された表のハンドル
sigAcct	列挙するサブ勘定科目を含む勘定科目の記号
lpCallback	ユーザ定義のコールバック関数のポインタ。詳しくは、 <a href="#">319 ページの「CALLBACKAPI」</a> を参照してください。
lParam	ユーザ定義のコールバック関数に返されるパラメータ

コールバック関数に渡される記号は、ID\_ACCOUNTS 内の勘定科目記号です。  
戻りコード：

コード	説明
0	成功
NONE	エラー発生
その他	コールバック関数から返される値

例：

```
ret% = EntEnumSubAcctSig(hSelect&, sigAcct&, AddressOf  
EnumSubAcctCB, 0)  
.  
.  
.  
'ユーザ定義の名前を持つコールバック関数  
Public Function EnumSubAcctCB(ByVal hSelect&, ByVal sigAcct&, ByVal  
sigNone&, ByVal lParam&, apiS As APISTRUCT) As Integer  
szAcct$ = Space(HYP_SIZEFULLACCT+1)  
ret% = EntQueryEx(hSelect&, HYP_ID_ACCOUNTS, NAME, sigAcct&,  
HYP_NONE, Len(szAcct$), szAcct$, 0)  
If ret% = 0 Then  
szAcct$ = CToBStr(szAcct$)  
ret2% = MsgBox(szAcct$)  
End If  
EnumSubAcctCB% = 0  
End Function
```

C 言語では次の形式を使用します。

```
short WINAPI EntEnumSubAcctSig(HSELECT hSelect, SIGNA sigAcct, CALLBACKAPI  
lpCallback, LONG lParam);
```

## EntEnumUsersOnSystem( ) - システム上のユーザの列挙

この関数は、任意のアプリケーションを現在使用しているユーザとプログラムを列挙します。指定されたアプリケーションを現在使用している各ユーザに対し、この関数はコールバック関数を呼び出します。

次の形式を使用します。

```
Declare Function EntEnumUsersOnSystem Lib "heaccess.dll" Alias
    "_EntEnumUsersOnSystem@12" (ByVal hApp As Long, ByVal lpCallback As Long, ByVal
    lParam As Long) As Integer
```

### 変数 説明

hApp      アプリケーションハンドル

lpCallback      ユーザ定義のコールバック関数のポインタ。詳しくは、[330 ページの「CALLBACKUSERS」](#)を参照してください。

lParam      lpCallback の引数

戻りコード：

コード	説明
0	成功
NONE	エラー発生
その他	コールバック関数から返される値

例：

```
ret% = EntEnumUsersOnSystem(hApp&, AddressOf MyCallBackFunc,
    lParam&)
```

C 言語では次の形式を使用します。

```
short WINAPI EntEnumUsersOnSystem(HAPP hApp, CALLBACKUSERS lpCallback,
LONG lParam);
```

## EntFind( ) - 記号の検索

この関数は、指定された表（wTabId）を与えられた検索文字列で検索し、検索文字列に一致するレコードの記号を返します。検索文字列は、通常はレコード（データ種別、エンティティなど）の ID（短い名前）か、勘定科目の場合には勘定科目 ID になります。

次の形式を使用します。

```
Declare Function EntFind Lib "heaccess.dll" Alias "_EntFind@20" (ByVal hSelect As Long,
ByVal wTabId As Integer, ByVal sigKey As Long, ByVal szKey As String, apiS As
APISTRUCT) As Long
```

## 変数 説明

hSelect 選択された表のハンドル

wTabId 検索する表

sigKey NONE、または関連する表のキー。wTabId が密接に関連している表である場合は、このキーが必要です。詳しくは、[340 ページの「関連する表」](#)を参照してください。

szKey 検索するレコードの検索文字列

apiS NULL、または詳細情報を含んでいる apiStruct を指すポインタ

戻りコード：

コード	説明
記号	成功
NONE	エラー発生

例：

```
sigCat& = EntFind0(hSelect&, HYP_ID_CATEGORY, HYP_NONE, "ACTUAL", 0)
```

EntFind0()は EntFind()と同じように宣言されますが、apiStruct の代わりに 0 を渡すことができるように調整されます。詳しくは、[16 ページの「Visual Basic プログラミング上の注意」](#)を参照してください。

C 言語では次の形式を使用します。

```
SIGNA WINAPI EntFind(HSELECT hSelect, short wTabId, SIGNA sigKey, void FAR *  
szKey, LPAPISTRUCT pApiS)
```

次の各項は、EntFind()で特定の Hyperion Enterprise 表を使用する方法について説明します。

## EntFind()と ID\_ACCTCONVERT（勘定科目変換一覧表）

EntFind()で勘定科目変換一覧表を使用するには、sigKey 引数を勘定科目変換一覧の記号に設定します。以下の表に示すフィールドを使用して apiStruct を設定してください。

フィールド	説明
u_ApiFind.sltemToFindOne	次のコードのいずれかを使用します。 <ul style="list-style-type: none"><li>検索文字列が外部勘定科目名の場合は ACONV_FIND_ACONV_EXT</li><li>検索文字列が Hyperion Enterprise 勘定科目 ID の場合は ACONV_FIND_ACONV_HYP</li></ul>
u_ApiFind.uclIncludeDel	EXCLUDE_DELETED

詳しくは、[110 ページの「apiStruct 構造体の使用」](#)を参照してください。

## EntFind() と ID\_INTCODET（会社間詳細表）

EntFind()では会社間詳細表を使用しないでください。代わりに、表 ID\_ICSET に対して EntFind()を呼び出して必要な会社間照合セットを検索し、次にクエリ属性 INTCO\_GROUPSIG（および表 ID\_ICSET）を使用して EntQueryEx()を呼び出して、表 ID\_INTCODET 内の最初の詳細レコードの記号を取得します。残りの詳細レコードを取得するには、クエリ属性 INTCO\_NEXTDET（および表 ID\_INTCODET）を使用して、各詳細レコードごとに EntQueryEx()を呼び出します。

## EntFind() と仕訳帳表：ID\_JOURNALS

すべての仕訳帳表には、データ種別と期間を指定する apiStruct が必要です。以下の表に示すフィールドを使用して apiStruct を設定してください。

フィールド	説明
sigCat	データ種別記号
lStartPeriod	期間
lEndPeriod	期間（lStartPeriod と同じ）各仕訳帳は 1 つのみの期間を取り扱います。

EntFind()で仕訳帳表（ID\_JOURNALS）を使用するには、apiStruct に lpString フィールドをさらに設定する必要があります。このフィールドは、仕訳帳名文字列のアドレスに設定します（この設定は Visual Basic ではできません）。sigKey 引数は HYP\_NONE（C 言語では NONE）に設定し、szKey 引数は NULL（通常の引数の代わりに仕訳帳名を apiStruct に渡します）に設定します。

EntFind()で仕訳帳詳細表のいずれか（ID\_JOURNAL\_DETAIL、ID\_JOURNAL\_HISTORY\_DETAIL、ID\_JOURNAL\_TEMPLATES\_DETAIL）を使用するには、sigKey 引数を適切な表内の仕訳帳記号に設定します。他の表（ID\_JOURNAL\_HISTORY、ID\_JOURNAL\_TEMPLATES、ID\_JOURNAL\_PERIOD\_INFO）に対しては、sigKey を HYP\_NONE（C 言語では NONE）に設定します。

詳しくは、[110 ページの「apiStruct 構造体の使用」](#)を参照してください。

## EntFind() と ID\_NAMECONVERT（エンティティ変換表）

EntFind()でエンティティ変換表を使用するには、sigKey 引数をエンティティ変換一覧の記号に設定します。以下の表に示すフィールドで apiStruct を設定してください。

フィールド	説明
u_ApiFind.sltemToFindCode	次のコードのいずれかを使用します。 <ul style="list-style-type: none"><li>● 検索文字列が外部エンティティの場合は NCONV_FIND_NCONV_EXT</li><li>● 検索文字列が Hyperion Enterprise エンティティの場合は NCONV_FIND_NCONV_HYP</li></ul>

フィールド	説明
u_ApiFind.uclIncludeDeletedItems	EXCLUDE_DELETED

詳しくは、[110 ページの「apiStruct 構造体の使用」](#)を参照してください。

## EntFind( )と ID\_NAMES（エンティティ表）

apiStruct を作成し、u\_ApiFind.sItemToFindCode フィールドを次の表に示すいずれかのコードに設定します。

NAME_IMPLICIT	エンティティのみが szKey に指定されている場合でも、エンティティまたはサブエンティティの記号を取得します。
NAME_EXPLICIT	サブエンティティを含むエンティティのサブエンティティを指定しなかった場合はエラーが返されます。
NAME_ADMIN	要求されたものの記号を（前述のコードの代わりに）取得します。

フィールドが設定されていないか、apiStruct が渡されなかった場合のデフォルトは、NAME\_ADMIN コードが指定された場合と同じになります。

次の表に、PTADJ の最上位ノードが関連付けられている下位構造体が EUROFF に存在する場合に返される各オプションを示します。

	返される記号		
szKey	NAME_ADMIN	NAME_IMPLICIT	NAME_EXPLICIT
"EUROFF"	EUROFF	EUROFF.PTADJ	NONE（エラー）
"EUROFF.PTADJ"	EUROFF.PTADJ	EUROFF.PTADJ	EUROFF.PTADJ
"USDIV"	USDIV	USDIV	USDIV

## EntFind( )と ID\_SECGRPTAB（セキュリティグループ表）

レコードキー（szKey）は、文字列ではなく、TOOLINC.H ファイルで定義されている SECURGROUPKEY 構造体です。構造体を作成し、そのフィールドにデータを入力します。構造体内の sigGroup フィールドは、セキュリティユーザ表（ID\_SECUSERTAB）内のグループの記号です。sigUser フィールドは、グループのメンバのユーザ記号です。このメンバは個人である場合も、別のグループである場合もあります。

## EntFind( )と ID\_SECRIGHTS（セキュリティ権限表）

レコードキー（szKey）は、文字列ではなく、TOOLINC.H ファイルで定義されている SECURRIGHTSKEY 構造体です。構造体を作成し、そのフィールドにデータを入力します。

## EntFind( )と ID\_SHARES (株式表)

EntFind( )で株式表を使用するには、NULL をレコードキー (szKey) として渡します。レコードキーは代わりに apiStruct 内で与えられます。データ種別記号を sigKey として渡します。連動組織を使用するようにアプリケーションが設定されている場合はこれが必要です。

株式表の各レコードは、1つのエンティティの株のいくつが別のエンティティによって所有されているかを示します。特定のレコードを検索するには、所有者と所有エンティティの両方のエンティティ記号を指定します。他のレコードには、特定のエンティティの発行済み株式総数と別のエンティティが所有する株式の合計が入っています。これらの他のレコードは、所有者としてエンティティ記号を指定し、所有エンティティ記号として NONE (Visual Basic では Hyp\_NONE) を指定します。

特定のレコードを検索するには、次の表に示すフィールドを使用して apiStruct を設定します。

フィールド	説明
sigParent	株式を所有するエンティティの記号
sigName	所有エンティティの記号。合計発行済み株式数または他のエンティティが所有している合計株式数を持つレコードが必要な場合は HYP_NONE。

連動組織を使用するようにアプリケーションが設定されている場合は、以下の表に示すフィールドを使用してさらに apiStruct を設定します。

フィールド	説明
sigCat	データ種別記号
lStartPeriod	期間
lEndPeriod	期間 (lStartPeriod と同じ)

詳しくは、[110 ページの「apiStruct 構造体の使用」](#)を参照してください。

## EntFindEntityInOrg( ) - ノード ID の検索

この関数は、任意の組織内のエンティティのノード記号を返します。sigNode 引数変数が HYP\_NONE (C 言語の場合は NONE) の場合、検索は組織の最上位ノードから開始されます。それ以外の場合は、検索は sigNode から開始されます。ノードについては、[120 ページの「組織構造体と高度な関数」](#)を参照してください。

次の形式を使用します。

```
Declare Function EntFindEntityInOrg Lib "heaccess.dll" Alias
    "_EntFindEntityInOrg@16" (ByVal hSelect As Long, ByVal sigOrg As Long, ByVal sigNode
    As Long, ByVal sigEntity As Long) As Long
```

## 変数 説明

hSelect 選択された表のハンドル

sigOrg 組織記号

sigNode 検索を開始するノード、または組織の最上部から開始する場合は NONE

sigEntity 検索するエンティティの記号

戻りコード：

コード	説明
ノード記号	成功
NONE	エラー発生

例：

```
sigNode& = EntFindEntityInOrg(hSelect&, sigOrg&, HYP_NONE,  
    sigEntity&)
```

C 言語では次の形式を使用します。

```
SIGNA WINAPI EntFindEntityInOrg(HSELECT hSelect, SIGNA sigOrg, SIGNA sigNode,  
SIGNA sigEntity);
```

## EntFindEntityInOrg( ) - 連動組織

連動組織を使用するようにアプリケーションが設定されている場合、EntFindEntityInOrg()関数では現在のデータ種別と現在の期間の組織構造体が必要であるものとみなされます（「EntQueryDefault( )」を参照）。それが必要でない場合は、EntEnum()を使用してエンティティを検索してください。EntFinEntityOrg()の代わりに EntEnum()を使用する方法を次に示します。

```
Function FindEntityInOrg(ByVal hSelect As Long, ByVal sigOrg As Long,  
ByVal sigNode As Long, ByVal sigEntity As Long) As Long, Dim apiS As  
APISTRUCT Dim myStruct As MYSTRUCT_TYPE)  
    sigFound& = HYP_NONE ret% = 0  
    'apiStructを設定  
hApp& = EntGetHappFromSelect(hSelect&)  
Call EntInitApiStruct(hApp&, apiS)  
apiS.sigCat& = sigCat&          'データ種別記号  
apiS.lStartPeriod = 0           '最初の期間を使用  
apiS.lEndPeriod& = apiS.lStartPeriod&  
apiS.u_ApiEnum.bNameSig = 0  
apiS.u_ApiEnum.dwMsk = ENUM_ALL  
    'sigNode&が指定されていない場合は最上位ノードを取得  
If sigNode& = HYP_NONE Then  
ret% = EntQueryEx(hSelect&, HYP_ID_ORGANIZATION, TOPNODE,  
sigOrg&, HYP_NONE, Len(sigNode&), sigNode&, 0)  
End If  
    If sigEntity& = HYP_NONE Then ret% = 1          'エラー  
    '必要な対象は最上位ノードか?  
If ret% = 0 Then
```

```

ret% = EntQueryEx(hSelect&, HYP_ID_NODES, NAMESIG,
sigNode&, HYP_NONE, Len(sig&), sig&, apiS)
If sig& = sigEntity Then      'エラー
sigFound& = sigNode&
ret% = 1                      '残りの手順を省略
End If
End If
If ret% = 0 Then
myStruct.sigEntity& = sigEntity&
myStruct.sigFound& = HYP_NONE
lParam& = EntGetVarAddr(myStruct)
ret2% = EntEnum(hSelect&, HYP_ID_NODES, sigNode&,
callback.ProcAddress, lParam&, apiS)
sigFound& = myStruct.sigFound&
End If
EntFindEntityInOrg& = sigFound&      '戻り値End Function
'ユーザ定義の名前のコールバック関数
Function EnumOrgCB(ByVal hSelect As Long, ByVal sigNode&
As Long, ByVal sigKey& As Long,
myStruct As MYSTRUCT_TYPE, apiS As APISTRUCT) As Integer
'エンティティ記号の取得と比較
ret% = EntQueryEx(hSelect&, HYP_ID_NODES, NAMESIG, sigNode&,
HYP_NONE, Len(sigEntity&), sigEntity&, apiS)
If sigEntity& = myStruct.sigEntity& Then      '検索成功!
myStruct.sigFound& = sigNode&
EnumOrgCB% = 1                      '列挙を停止
Else
EnumOrgCB% = 0                      '列挙を続行
End If
End Function

```

## EntFormatNumber2( ) - 数値の書式設定

この関数は、数値を書式設定します。

次の形式を使用します。

```

Declare Function EntFormatNumber2 Lib "heaccess.dll" Alias
"_EntFormatNumber2@32" (ByVal hApp As Long, ByVal dVal As Double, ByVal szBuf As
String, ByVal wLen As Integer, ByVal wScale As Integer, ByVal wDecimals As Integer, ByVal
bRetZero As Integer) As Integer

```

変数	説明
hApp	アプリケーションハンドル
dVal	書式設定する値
szBuf	書式設定された数値を返すためのバッファ
wLen	pzBuf の長さ
wScale	適用する単位（存在する場合）
wDecimals	小数点以下桁数

## 変数 説明

bRetZero 文字列を"0.00"（またはゼロの場合は空）として書式設定するためのフラグ

戻りコード：

この関数は、常にゼロを返します。

C 言語では次の形式を使用します。

```
SIGNA WINAPI EntFormatNumber2(HAPP hApp, double dVal, LPSTR szBuf, short wLen, short wScale, short wDecimals, short bRetZero);
```

## EntFreqAsk( ) - 期間単位の選択

この関数は、[期間単位の選択] ダイアログボックスを呼び出し、選択された期間単位を取得します。[キャンセル] を選択した場合、期間単位は取得されません。この関数を呼び出す前に、レポート期間単位 (ID\_RPTFREQ) とレポート表示形式 (ID\_RPTVIEW) の表を選択する必要があります。

次の形式を使用します。

```
Declare Function EntFreqAsk Lib "heaccess.dll" Alias "_EntFreqAsk@8" (ByVal hSelect As Long, ByVal szRetbuf As String) As Integer
```

## 変数 説明

hSelect 選択された表のハンドル

szRetbuf 期間単位コードを返すためのバッファ。バッファの長さは少なくとも 2 \* (HYP\_SIZELABEL + 1) 文字にする必要があります。文字列は、freq.view の形式で返されます。freq は Hyperion Enterprise 期間単位で、view はレポート表示形式です。期間単位と表示形式については、[20 ページの「期間単位と表示形式」](#)を参照してください。

戻りコード：

コード	説明
0	成功
NONE	エラー発生

例：

```
szFreq$ = Space(2*(HYP_SIZELABEL+1))
ret% = EntFreqAsk(hSelect&, szFreqView$)
If ret% = 0 Then szFreqView$ = CToBStr(szFreqView$)
```

C 言語では次の形式を使用します。

```
short WINAPI EntFreqAsk( HSELECT hSelect, LPSTR szRetbuf );
```

## EntGetAccountsInputType( ) - 勘定科目種別の取得

この関数は、勘定科目の種別を返します。

次の形式を使用します。

```
Declare Function EntGetAccountsInputType Lib "heaccess.dll" Alias_  
"EntGetAccountsInputType@36" (ByVal hSelect As Long, ByVal sigCat As Long, ByVal  
sigEntity As Long, ByVal sigParentEntity As Long, ByVal sigAccount As Long, ByVal  
wDSMType As Integer, ByVal lStartPeriod As Long, ByVal lEndperiod As Long,  
lRetAcctType As Long) As Integer
```

変数	説明
ENTXA_ACCT_INPUT	入力勘定科目（エンティティは親ではありません）
ENTXA_ACCT_CALCULATED	算出勘定科目。ロジックによって、またはエンティティが親エンティティであるために算出されたもの。
ENTXA_ACCT_LOGICINPUT	ロジックステートメントによって、勘定科目は入力勘定科目になります。
ENTXA_ ACCTIMPLIEDSUBTOTAL	勘定科目は、サブ勘定科目表が関連付けられた主要勘定科目です。
ENTXA_ACCTCONSOLIDATED	エンティティは、指定された期間範囲のアクティブな親です。
ENTXA_ACCTNOINPUT	勘定科目は、通貨サブ表が関連付けられた主要勘定科目です。
ENTXA_ACCTBADGROUP	エンティティがグローバルエンティティであっても勘定科目がグローバル勘定科目でないか、勘定科目がグローバル勘定科目であってもエンティティがグローバルエンティティではありません。
ENTXA_ACCT_DYNAMICVIEW	連動表示勘定科目（その他のビットフラグは設定されていません）
ENTXA_ACCT_ERROR	エラーが発生しています（不正な hSelect、無効な sigCat、無効な sigAccount、無効な sigParentEntity、内部クエリの失敗、メモリー不足、無効な期間範囲、ロジック付加エラーなど）。

変数	説明
hSelect	選択された表のハンドル
sigCat	データ種別の記号
sigEntity	エンティティの記号または None。NONE の場合、関数はデフォルトの入力ロジックを使用します。また、エンティティが親エンティティではなく、最下位レベルであることが想定されます。
sigParentEntity	wDSMType が ID_REGULAR の場合は None、それ以外の場合は親エンティティの記号
sigAccount	勘定科目の記号。これは必須です。NONE を指定することはできません。
wDSMType	連結詳細タイプ。通常のデータには ID_REGULAR を使用します。次のタイプのいずれかを指定する必要があります。 <ul style="list-style-type: none"><li>● ID_REGULAR</li><li>● ID_PROPORTIONAL</li><li>● ID_ELIMINATION</li><li>● ID_CONTRIBUTION</li><li>● ID_ADJUSTMENT</li></ul>

変数	説明
	<ul style="list-style-type: none"> <li>● ID_TRANSLATION</li> </ul>
IStartPeriod	開始期間。最初の期間（期間 0）を使用する場合は NONE にできます。
IEndPeriod	開始期間。連動組織（期間ごとの組織）を含むアプリケーションでは、データ種別の最後の期間を使用する場合はこれを NONE にできます。アプリケーションが連動組織（期間ごとの組織）でない場合、IEndPeriod には IStartPeriod よりも大きい値（ただしデータ種別内の期間数以下）を指定する必要があります。 <b>ヒント</b> ：連動組織（期間ごとの組織）を使用しないアプリケーションでは、IStartPeriod を 0 に設定し、IEndPeriod を 1 に設定します。
IRetAcctType	<p>結果が返される変数。これは必須です。この関数からは、次の 1 つまたは複数の勘定科目種別が IRetAcctType に返されます。</p> <ul style="list-style-type: none"> <li>● ENTXA_ACCT_INPUT</li> <li>● ENTXA_ACCT_CALCULATED</li> <li>● ENTXA_ACCT_LOGICINPUT</li> <li>● ENTXA_ACCT_IMPLIEDSUBTOTAL</li> <li>● ENTXA_ACCT_CONSOLIDATED</li> <li>● ENTXA_ACCT_NOINPUT</li> <li>● ENTXA_ACCT_BADGROUP</li> <li>● ENTXA_ACCT_DYNAMICVIEW</li> <li>● ENTXA_ACCT_ERROR</li> </ul>

エラーがなく（ENTXA\_ACCT\_ERROR ビットフラグが設定されていない）、勘定科目が連動表示勘定科目でない（ENTXA\_ACCT\_DYNAMICVIEW ビットフラグが設定されていない）場合は、ENTXA\_ACCT\_INPUT、ENTXA\_ACCT\_CALCULATED または ENTXA\_ACCT\_LOGICINPUT ビットフラグのいずれか 1 つのみが設定されます。その他のビットフラグが、設定されるいずれかのビットフラグと組み合わせられる場合もあります。

**ヒント：** ENTXA\_ACCT\_INPUT はゼロとして定義されます。これをテストするには、ENTXA\_ACCT\_CALCULATED と ENTXA\_ACCT\_LOGICINPUT のどちらのビットフラグも設定されないことをテストします。

別の方法については、[210 ページの「EntIsAccountInput\( \) - 勘定科目が入力勘定科目であるかどうかの確認」](#)を参照してください。

戻りコード：

コード	説明
0	成功
NONE	エラー発生

C 言語では次の形式を使用します。

```
short WINAPI EntGetAccountsInputType(HSELECT hSelect, SIGNA sigCat, SIGNA sigEntity, SIGNA sigParentEntity, SIGNA sigAccount, WORD wDSMType, long IStartPeriod, long IEndPeriod, long* plRetAcctType);
```

## EntGetActiveModule( ) - プログラム名の取得

この関数は、引数変数 hApp によって指定されているアプリケーションを開いたプログラムまたはモジュールのコードを返します。Hyperion Enterprise モジュールを定義するコードについては、MF\_コードを示す、TOOLINC.H ファイル内のセクションを参照してください。プログラムでは MF\_HACCESS を使用する必要があります。

次の形式を使用します。

```
Declare Function EntGetActiveModule Lib "heaccess.dll" Alias
"_EntGetActiveModule@16" (ByVal hApp As Long, wParam As Integer, ByVal szUser As
String, ByVal ncbUser As Long) As Integer
```

変数	説明
----	----

hApp	アプリケーションハンドル
------	--------------

wModule	プログラムまたはモジュールを識別するコードを返すためのバッファ
---------	---------------------------------

szUser	未使用
--------	-----

ncbUser	未使用
---------	-----

戻りコード：

この関数は、常にゼロを返します。

例：

```
ret% = EntGetActiveModule(hApp&, progCode%, dummy$, 0)
If progCode% <> MF_HACCESS Then MsgBox("Something's Wrong")
```

C 言語では次の形式を使用します。

```
short WINAPI EntGetActiveModule(HAPP hApp, short * pwModule, LPSTR pszUser,
int ncbUser)
```

## EntGetAppProfileLong( ) - アプリケーションプロファイル long の取得

この関数は、指定されているアプリケーション.INI ファイルに格納されている数値を返します。

**注：** アプリケーション.INI ファイルから値を取得するには、SQL アプリケーションおよびファイルベースのアプリケーションのどちらに対しても、ID\_APPDEFAULT 表を指定して EntQueryDefault( )関数を使用する必要があります。SQL アプリケーションには.INI ファイルはありません。

次の形式を使用します。

Declare Function EntGetAppProfileLong Lib "heaccess.dll" Alias  
"\_EntGetAppProfileLong@12" (ByVal szAppName As String, ByVal szKeyName As String,  
ByVal lDefault As Long) As Long

変数	説明
----	----

szAppName	アプリケーション名
-----------	-----------

szKeyName	値のキー名
-----------	-------

lDefault	szKeyName が見つからない場合に割り当てられるデフォルト値
----------	-----------------------------------

戻りコード：

コード	説明
任意のキーの値	成功
NONE	エラー発生

例：

```
isServerEnabled& = EntGetAppProfileLong (szApp$,  
"APP_USE_SERVER", 0)
```

C 言語では次の形式を使用します。

```
long WINAPI EntGetAppProfileLong(LPSTR szAppName, LPCSTR szKeyName, long  
lDefault)
```

## EntGetAppProfileString( ) - アプリケーションプロファイル文字列の取得

この関数は、アプリケーション.INI ファイルからテキスト文字列を読み取ります。

**注：** アプリケーション.INI ファイルから値を取得するには、SQL アプリケーションおよびファイルベースのアプリケーションのどちらに対しても、ID\_APPDEFAULT 表を指定して EntQueryDefault( )関数を使用する必要があります。SQL アプリケーションには.INI ファイルはありません。

次の形式を使用します。

```
Declare Function EntGetAppProfileString Lib "heaccess.dll" Alias  
"_EntGetAppProfileString@20" (ByVal szAppName As String, ByVal szKeyName As  
String, ByVal szDefault As String, ByVal szResult As String, ByVal wSize As Integer) As  
Integer
```

変数	説明
----	----

szAppName	アプリケーション名
-----------	-----------

## 変数 説明

szKeyName	文字列のキー名
szDefault	szKeyName が見つからない場合に割り当てられるデフォルト値
szResult	文字列を返すためのバッファ
wSize	szResult の長さ（バイト数）

戻りコード：

コード	説明
szResult にコピーされた文字数	成功
NONE	エラー発生

例：

```
szPath$ = Space(128)
wLen% = EntGetAppProfileString(szApp$, "APP_OUTBOX", "@APP",
    szPath$, Len(szPath$))
If wLen% Then szPath$ = CToBStr(szPath$)
```

C 言語では次の形式を使用します。

```
short WINAPI EntGetAppProfileString(LPCSTR szAppName, LPCSTR szKeyName,
LPCSTR szDefault, LPSTR szResult, short wSize)
```

## EntGetCatPerFreq( ) - データ種別内の期間番号と期間単位の取得

この関数は、特定の期間に対して任意のデータ種別内の期間番号を取得します。また、引数変数 sigFreq が NULL でない場合に期間文字列に使用される期間単位も戻します。

次の形式を使用します。

```
Declare Function EntGetCatPerFreq Lib "heaccess.dll" Alias "_EntGetCatPerFreq@16"
(ByVal hSelect As Long, ByVal szPeriod As String, ByVal sigCat As Long, sigFreq As Long)
As Long
```

## 変数 説明

hSelect	選択された表のハンドル
szPeriod	JAN 03 などの期間のラベル
sigCat	データ種別記号
sigFreq	NULL、または期間単位記号を返すためのバッファ。これは、szPeriod で使用される期間単位になります。

戻りコード：

コード	説明
期間番号	成功
NONE	エラー発生

例：

```
sigPeriod& = EntGetCatPerFreq(hSelect&, "Q1 03", sigCat&,
    sigFreq&)
```

この例では、指定されたデータ種別の Q1 03 の期間が返されます。sigFreq には、四半期の期間単位の記号が返されます。この例は、四半期のデータ種別を使用しなくても機能します。

C 言語では次の形式を使用します。

```
SIGNA WINAPI EntGetCatPerFreq(HSELECT hSelect, LPSTR szPeriod, SIGNA sigCat,
    SIGNA * psigFreq)
```

## EntGetChild( ) - 子ノードの取得

この関数は、指定されたノードの子ノード記号を取得します。ノードについては、[120 ページの「組織構造体と高度な関数」](#)を参照してください。

次の形式を使用します。

```
Declare Function EntGetChild Lib "heaccess.dll" Alias "_EntGetChild@12" (ByVal hSelect
    As Long, ByVal sigNode As Long, ByVal fExpandSubnames As Integer) As Long
```

### 変数 説明

hSelect 選択された表のハンドル

sigNode 取得する子を持つノードの記号

fExpandSubnames サブエンティティを考慮する場合は 1 (TRUE)、それ以外の場合は 0 (False)

戻りコード：

コード	説明
子ノードの記号	成功
NONE	エラー発生

例：

```
sigChild& = EntGetChild(hSelect&, sigNode&, 1)
```

C 言語では次の形式を使用します。

```
SIGNA WINAPI EntGetChild(HSELECT hSelect, SIGNA sigNode, SBOOL
    fExpandSubnames)
```

## EntGetChild( ) - 連動組織

連動組織を使用するようにアプリケーションが設定されている場合、EntGetChild() 関数では現在のデータ種別と現在の期間の組織構造体が必要であるものとみなされます（「EntQueryDefault( )」を参照）。それが必要でない場合は、EntQueryEx( ) を使用して子ノードの記号を取得できます。表 ID、ID\_NODES（Visual Basic では HYP\_ID\_NODES）と、クエリ属性 CHILD を使用します。次に、EntGetChild( ) の代わりに EntQueryEx( ) を使用する方法例を示します。

```
Function GetChild(hSelect&, sigNode&, fExpandSubentities%)
As Long
Dim apiS As APISTRUCT
'apiStructを設定
hApp& = EntGetHappFromSelect (hSelect&)
Call EntInitApiStruct (hApp&, apiS)
apiS.sigCat& = sigCat&          'データ種別記号
apiS.lStartPeriod = 0          '最初の期間を使用
apiS.lEndPeriod& = apiS.lStartPeriod&
If fExpandSubentities% = 0 Then
apiS.u_ApiQry.bSupSubNames = 1 'サブエンティティを非表示
Else
apiS.u_ApiQry.bSupSubNames = 0
End If
'sigChild&を取得
sigChild& = HYP_NONE
ret% = EntQueryEx(hSelect&, HYP_ID_NODES, CHILD, sigNode&,
HYP_NONE, Len(sigChild&), sigChild&, apiS)
GetChild& = sigChild&
End Function
```

## EntGetEntitySig( ) - エンティティ記号の取得

この関数は、与えられたアプリケーションハンドル、エンティティおよびサブエンティティ条件コードに基づいてエンティティ記号を返します。次の表に、3つの可能なサブエンティティ条件コードを示します。

表 47 EntGetEntitySig( ) のサブエンティティ条件コード

コード	返される情報
NAME_IMPLICIT	エンティティが szKey に指定されている場合でも、entity.subentity の記号
NAME_EXPLICIT	サブエンティティを持たないエンティティのサブエンティティを指定した場合は、エラーメッセージ
NAME_ADMIN	（前のコードではなく）要求した項目の記号

フィールドが設定されていないか、apiStruct が渡されなかった場合のデフォルトは、NAME\_ADMIN と同じになります。

例として、EUROFF に最上位ノード PTADJ が関連付けられた下位構造体があり、USDIV には下位構造体がない場合に返される各オプションを次の表に示します。

	返される記号		
szKey	NAME_ADMIN	NAME_IMPLICIT	NAME_EXPLICIT
"EUROFF"	EUROFF	EUROFF.PTADJ	NONE（エラー）
"EUROFF.PTADJ"	EUROFF.PTADJ	EUROFF.PTADJ	EUROFF.PTADJ
"USDIV"	USDIV	USDIV	USDIV

次の形式を使用します。

```
Declare Function EntGetEntitySig Lib "heaccess.dll" Alias "_EntGetEntitySig@12" (ByVal hSelect As Long, ByVal szEntity As String, ByVal fImpliedSubEntities As Integer) As Long
```

変数	説明
hSelect	選択された表のハンドル
szEntity	検索対象のエンティティ
fImpliedSubEntities	次のコードのいずれか。 <ul style="list-style-type: none"> <li>● NAME_IMPLICIT</li> <li>● NAME_EXPLICIT</li> <li>● NAME_ADMIN</li> </ul>

戻りコード：

コード	説明
エンティティ記号	成功
NONE	エラー発生

例：

```
'次の行はItaly.Tranの記号を返します
sigEntity& = EntGetEntitySig(hSelect&, "Italy",NAME_IMPLICIT)
'次の行はHYP_NONEを返します
sigEntity& = EntGetEntitySig(hSelect&, "Italy",NAME_EXPLICIT)
'次の行はItalyのみの記号を返します
sigEntity& = EntGetEntitySig(hSelect&, "Italy",NAME_ADMIN)
```

C 言語では次の形式を使用します。

```
SIGNA WINAPI EntGetEntitySig(HSELECT hSelect, LPSTR szEntity, short fImpliedSubEntities)
```

## EntGetHappFromSelect( ) - アプリケーションハンドルの取得

この関数は、任意の表選択ハンドルに関連付けられているアプリケーションハンドルを返します。

次の形式を使用します。

Declare Function EntGetHappFromSelect Lib "heaccess.dll" Alias  
"\_EntGetHappFromSelect@4" (ByVal hSelect As Long) As Long

ここで、hSelect は選択された表のハンドルです。

戻りコード：

コード	説明
アプリケーションハンドル	成功
NONE	エラー発生

例：

```
hApp& = EntGetHappFromSelect (hSelect&)
```

C 言語では次の形式を使用します。

HAPP WINAPI EntGetHappFromSelect(HSELECT hSelect)

## EntGetLastError( ) - 最後のエラーの取得

この関数は、前の API 呼び出しによって設定された最後のエラーコードを取得します。他の API 関数によってエラーコード（通常は NONE）が返された場合は、この関数を呼び出して、そのエラーに関する詳細情報を取得します。この関数は、直前に呼び出された API 関数によって設定された最後の内部 API エラーコードを返します。エラーコードをすべて含むリストについては、TOOLONC.H ファイルを参照してください。詳しくは、[203 ページの「EntGetLastErrorByHApp\( \) - 最後のエラーの取得」](#)を参照してください。

次の形式を使用します。

Declare Function EntGetLastError Lib "heaccess.dll" Alias "\_EntGetLastError@4" (ByVal  
hSelect As Long) As Integer

ここで、hSelect は選択された表のハンドルです。

戻りコード：

コード	説明
エラーコード	成功
NONE	エラー発生

EntDataFileOpen( )の呼び出し後：

FILE_NOEXIST	要求されたデータファイル表が存在せず、bCreate 引数が FALSE であったので、表は作成されませんでした。データなし。
SECURITY_VIOLATION	要求されたデータファイル表および wAttr アクセスコードへのアクセス権がユーザにありません。
ERR_TABLE_NOT_SELECTED	必要な表が選択されていません。 <b>ヒント：</b> 連動組織を使用するアプリケーションの場合は、表 ID_ORGANIZATION およびそれに関連付けられている表（ID_ASSOC）を選

コード	説明
	扱ったときに、正しいデータ種別を指定したことを確認してください。

表 ID\_DATAFILE の EntQueryEx( )の呼び出し後：

DFER_NOERR	エラーなし
DFER_BADPERIOD	不正な期間
INSUFFICIENT_SECURITY RIGHTS	

例：

```
wRet% = EntAppLoadVB(hApp&, s, szFile$)
wErrCode% = EntGetLastErrorByHApp(hApp&)
If wErrCode% = MEMORY_ERROR Then MsgBox("Insufficient memory.",
    vbOKOnly, "Application Load")
```

C 言語では次の形式を使用します。

**short WINAPI EntGetLastError(HSELECT hSelect)**

## EntGetLastErrorByHApp( ) - 最後のエラーの取得

この関数は、前の API 呼び出しによって設定された最後のエラーコードを取得します。これは EntGetLastError( )と同じですが、EntGetLastErrorByHApp( )ではアプリケーションハンドルを引数として使用するので、表が選択されていなくても使用できます。

他の API 関数によってエラーコード（通常は NONE）が返された場合は、この関数を呼び出して、そのエラーに関する詳細情報を取得します。この関数は、Hyperion Enterprise アプリケーションに対して直前に呼び出された API 関数によって設定された最後の内部 API エラーコードを返します。すべてのエラーコードを含むリストについては、TOOLINC.H ファイルの EntGetLastError のセクションを参照してください。

次の形式を使用します。

**Declare Function EntGetLastErrorByHApp Lib "heaccess.dll" Alias  
 "\_EntGetLastErrorByHApp@4" (ByVal hApp As Long) As Integer**

ここで、hApp はアプリケーションハンドルです。

戻りコード：

コード	説明
エラーコード	成功
NONE	エラー発生

**注：** 一部の戻りコードを含むリストについては、[202 ページの「EntGetLastError\( \) - 最後のエラーの取得」](#)を参照してください。

C 言語では次の形式を使用します。

```
short WINAPI EntGetLastErrorByHApp(HAPP hApp)
```

## EntGetOrgLevel( ) - 組織内のレベルの取得

この関数は、指定されたノードの組織レベルを返します。連動組織を使用するようにアプリケーションが設定されている場合、この関数では現在の期間とデータ種別の組織構造体を使用します。現在の期間とデータ種別の組織構造体が必要でない場合は、EntUpdateDefault( )を使用して現在の期間とデータ構造体を一時的に変更してください。

次の形式を使用します。

```
Declare Function EntGetOrgLevel Lib "heaccess.dll" Alias "EntGetOrgLevel@16" (ByVal hSelect As Long, ByVal sigTopNode As Long, ByVal sigTarget As Long, ByVal wLevel As Integer) As Integer
```

変数	説明
----	----

hSelect	選択された表のハンドル
---------	-------------

sigTopNode	組織内の最上位ノード、またはカウントを開始するノードの記号
------------	-------------------------------

sigTarget	組織レベルを調べる対象のノードの記号
-----------	--------------------

wLevel	通常は 0 または sigTopNode のレベルより 1 つ少ない値
--------	-------------------------------------

戻りコード：

コード	説明
組織レベル	成功
0	エラー発生

例：

```
ret% = EntQueryEx(hSelect&, HYP_ID_ORGANIZATION, TOPNODE,  
    sigOrg&, HYP_NONE, Len(sigTop&), sigTop&, 0)  
If ret% = 0 Then  
    nLevel% = EntGetOrgLevel(hSelect&, sigTop&, sigTarget&, 0)  
End If
```

C 言語では次の形式を使用します。

```
short WINAPI EntGetOrgLevel(HSELECT hSelect, SIGNA sigTopNode, SIGNA  
sigTarget, short wLevel)
```

## EntGetPerView( ) - 期間単位と表示形式の取得

この関数は、指定されたレポート期間単位の期間単位と表示形式を取得します。この関数を呼び出す前に、レポート期間単位 (ID\_RPTFREQ) とレポート表示形式 (ID\_RPTVIEW) の表を選択する必要があります。

次の形式を使用します。

**Declare Function EntGetPerView Lib "heaccess.dll" Alias "\_EntGetPerView@16" (ByVal hSelect As Long, ByVal szFreq As String, sigFreq As Long, iView As Integer) As Integer**

**変数 説明**

hSelect 選択された表のハンドル

szFreq レポートの期間単位。DAI、DYTD、WEE、WYTD、MON、YTD、QUA、QYTD、TRI、TYTD、HAL、HYTD、YEA または DAY のいずれかでなければなりません。詳しくは、[20 ページの「期間単位と表示形式」](#)を参照してください。

sigFreq szFreq によって指定されたレポート期間単位からの期間単位を返す場所

iView szFreq によって指定されたレポート期間単位からの表示形式を返す場所 (VIEW\_YTD または VIEW\_PERIODIC のどちらか)

戻りコード：

コード	説明
0	成功
NONE	エラー発生

C 言語では次の形式を使用します。

**short WINAPI EntGetPerView(HSELECT hSelect, LPSTR szFreq, SIGNA \* psigFreq, short \* piView)**

## EntGetProfileLong( ) - Hyperion プロファイル long の取得

この関数は、HYPENT.INI に格納されている数値を読み取ります。

次の形式を使用します。

**Declare Function EntGetProfileLong Lib "heaccess.dll" Alias "\_EntGetProfileLong@12" (ByVal szAppName As String, ByVal szKeyName As String, ByVal lDefault As Long) As Long**

**変数 説明**

szAppName アプリケーション名、または [Default] セクションの場合は NULL

szKeyName 値のキー名

lDefault szKeyName が見つからない場合に割り当てられるデフォルト値

戻りコード：

コード	説明
任意のキーの値	成功

例：

```
isServerOpt& = EntGetProfileLong(vbNullString, "USE_SERVER", 0)
```

C 言語では次の形式を使用します。

```
long WINAPI EntGetProfileLong(LPCSTR szAppName, LPCSTR szKeyName, long
lDefault)
```

## EntGetProfileString( ) - Hyperion プロファイル文字列の取得

この関数は、HYPENT.INI ファイルに格納されている文字列を読み取ります。

次の形式を使用します。

```
Declare Function EntGetProfileString Lib "heaccess.dll" Alias "_EntGetProfileString@20"
(ByVal szAppName As String, ByVal szKeyName As String, ByVal szDefault As String,
ByVal szResult As String, ByVal wSize As Integer) As Integer
```

### 変数 説明

szAppName アプリケーション ID、または [Default] セクションの場合は NULL

szKeyName 文字列のキー名

szDefault szKeyName が見つからない場合に割り当てられるデフォルト値

szResult 文字列の結果を受け取るバッファ

wSize szResult バッファの長さ（バイト単位）

戻りコード：

コード	説明
szResult にコピーされた文字数	成功
0	エラー発生

例：

```
szDefaultApp$ = Space(HYP_SIZELABEL+1)
wLen% = EntGetProfileString(vbNullString, "APPID", "",
szDefaultApp$, Len(szDefaultApp$))
If wLen% Then szDefaultApp$ = CToBStr(szDefaultApp$)
```

C 言語では次の形式を使用します。

```
short WINAPI EntGetProfileString(LPCSTR szAppName, LPCSTR szKeyName, LPCSTR
szDefault, LPSTR szResult, short wSize)
```

## EntGetRightsToTask( ) - タスクのアクセス権の取得

この関数は、指定されたタスクへの現在のユーザのアクセス権を取得します。  
次の形式を使用します。

Declare Function EntGetRightsToTask Lib "heaccess.dll" Alias "\_EntGetRightsToTask"  
(ByVal hApp As Long, ByVal lTaskID As Long) As Integer

**変数 説明**

hApp アプリケーションハンドル

lTaskID 現在のユーザのアクセス権の取得対象となるタスク。これは、`TOOLINC.H` ファイルに  
`SECTASK_xxxx` の形式で定義されているセキュリティタスクコードのいずれかでなければ  
なりません。

戻りコード：

コード	説明
SECURITY_MODIFY	変更（書き込み）
SECURITY_NONE	なし
SECURITY_RESTRICTED	制限付き（部分的な表示）
SECURITY_VIEW	読み取り専用（表示）

例：

secRight% = **EntGetRightsToTask**(hApp&, SECTASK\_LOADDATA)

C 言語では次の形式を使用します。

short WINAPI EntGetRightsToTask(HAPP hApp, long lTaskID)

## EntGetSibling( ) - 兄弟ノードの取得

この関数は、組織内の次の兄弟ノードの記号を取得します。ノードについては、  
[120 ページの「組織構造体と高度な関数」](#)を参照してください。

次の形式を使用します。

Declare Function EntGetSibling Lib "heaccess.dll" Alias "\_EntGetSibling@8" (ByVal  
hSelect As Long, ByVal sigNode As Long) As Long

**変数 説明**

hSelect 選択された表のハンドル

sigNode ノード記号

戻りコード：

コード	説明
次の兄弟ノードの記号	成功
NONE	エラー発生

例：

```
sigSibling& = EntGetSibling(hSelect&, sigNode&)
```

C 言語では次の形式を使用します。

```
SIGNA WINAPI EntGetSibling(HSELECT hSelect, SIGNA sigNode)
```

## EntGetSibling( ) - 連動組織

連動組織を使用するようにアプリケーションが設定されている場合、EntGetSibling 関数では現在のデータ種別と現在の期間の組織構造体が必要であるものとみなされます。詳しくは、[248 ページの「EntQueryDefault\( \) - デフォルトのクエリ」](#)を参照してください。それが不要な場合は、EntQueryEx( )を使用して組織内の次の兄弟ノードの記号を取得できます。表 ID、ID\_NODES (Visual Basic では HYP\_ID\_NODES) と、クエリ属性 SIBLING を使用します。次に、EntGetSibling( )の代わりに EntQueryEx( )を使用する方法の例を示します。

```
Function EntGetSibling(hSelect&, sigNode&) As Long
Dim apiS As APISTRUCT
'apiStructを設定
hApp& = EntGetHappFromSelect (hSelect&)
Call EntInitApiStruct (hApp&, apiS)
apiS.sigCat& = sigCat&           'データ種別記号
apiS.lStartPeriod = 0           '最初の期間を使用
apiS.lEndPeriod& = apiS.lStartPeriod&
'sigSibling&を取得
sigSibling& = HYP_NONE
ret% = EntQueryEx(hSelect&, HYP_ID_NODES, SIBLING, sigNode&,
HYP_NONE, Len(sigSibling&), sigSibling&, apiS)
GetSibling& = sigSibling&
End Function
```

## EntGetVarAddr( ) - 変数のアドレスの取得

EntGetVarAddr( )を使用すると、Microsoft Visual Basic がアドレスを変数に割り当てるように強制できます。この関数は、渡された引数のアドレスを返します。引数は参照で渡すので、渡したものと同じアドレスが返されます。この関数は、apiStruct 構造体内のアドレスフィールドを設定するために使用できます。これは、一部の他の Hyperion Enterprise API 関数で必要になります。

**注：** 文字列のアドレスを取得する場合は EntGetVarAddr( )を使用しないでください。

次の形式を使用します。

**Declare Function EntGetVarAddr Lib "heaccess.dll" Alias "\_EntGetVarAddr@4" (pBuf As Any) As Long**

ここで、pBuf は関数に渡すアドレスです。

戻りコード：

コード	説明
関数に渡した変数のアドレス	成功
NONE	エラー発生

例：

```
apiS.lpimrData = EntGetVarAddr(dVal#)
```

**注意** Microsoft Visual Basic では、文字列の操作が行われたときに、メモリ内の内容が再配置される場合があります。再配置される内容には、文字列の割り当て、文字列の連結および文字列の関数があります。

背後で行われる文字列の操作によってもこの問題が発生する場合があります。例えば、固定長の文字列を関数の引数として渡した場合、Visual Basic ではその文字列の一時的なコピーを作成し、そのコピーを代わりに渡します。配列の次元を変更（ReDim）した場合にも、Visual Basic によってメモリ内の項目が再配置されることがあります。EntGetVarAddr() 関数を使用して動的な配列のアドレスを取得する場合には、十分注意してください。メモリ内でこの配列が移動する場合があります。この関数を使用する直前にアドレスを取得し、Visual Basic によるメモリ内の項目の再配置を発生させるような操作は実行しないようにしてください。そうしないと、一時的ではあっても、検出が非常に困難な重大なバグが発生する場合があります。

## EntInitApiStruct( ) - API 構造体の初期化

この関数は、apiStruct 構造体を初期化します。

次の形式を使用します。

**Declare Sub EntInitApiStruct Lib "heaccess.dll" Alias "\_EntInitApiStruct@8" (ByVal hApp As Long, apiS As APISTRUCT)**

### 変数 説明

hApp アプリケーションハンドル

apiS apiStruct 構造体のポインタ

戻りコード：

なし

例：

```
Dim apiS As APISTRUCT
```

**EntInitApiStruct** hApp&, apiS

C 言語では次の形式を使用します。

**void** WINAPI EntInitApiStruct(HAPP hApp, LPAPISTRUCT pApiS)

## EntIsAccountInput( ) - 勘定科目が入力勘定科目であるかどうかの確認

この関数は、勘定科目が特定のエンティティの入力勘定科目であるかどうかを判断するために使用します。これは計算式に依存するので、この関数を呼び出す前に、データ種別とエンティティのデータファイル表 (ID\_DATAFILE) を選択するか、データ種別とロジックの EntLogicAttach( ) を呼び出す必要があります。

次の形式を使用します。

**Declare Function** EntIsAccountInput Lib "heaccess.dll" Alias "\_EntIsAccountInput@20"  
(ByVal hSelect As Long, ByVal sigCat As Long, ByVal sigEntity As Long, ByVal  
sigAccount As Long, blsInput As Byte) As Integer

### 変数 説明

hSelect 選択された表のハンドル

sigCat データ種別記号

sigEntity エンティティ記号

sigAccount 勘定科目記号

blsInput 結果を返すためのバッファ。勘定科目が入力勘定科目である場合は 1 (TRUE) が返され、そうでない場合は 0 (False) が返されます。

戻りコード：

コード	説明
0	成功
NONE	エラー発生

C 言語では次の形式を使用します。

**short** WINAPI EntIsAccountInput(HSELECT hSelect, SIGNA sigCat, SIGNA sigEntity,  
SIGNA sigAccount, HYPBOOL\* pbIsInput);

## EntIsEntityParent( ) - エンティティが親であるかどうかの確認

この関数は、エンティティが親エンティティかどうかを確認します。sigOrg が NONE に設定されている場合、Hyperion Enterprise ではエンティティがアプリケーション内のどこかで親エンティティになっているかどうかを確認します。sigOrg

に組織記号が指定されている場合、Hyperion Enterprise ではエンティティが組織内で親エンティティになっているかどうかを確認します。組織については、[120 ページの「組織構造体と高度な関数」](#)を参照してください。

**注：** エンティティは、組織表示で子が表示されていなくても親になっている可能性があります。

連動組織を使用するようにアプリケーションが設定されている場合、この関数では現在のデータ種別と現在の期間の組織構造体が必要であるとみなされます。現在のデータ種別と現在の期間の組織構造体を使用する必要がない場合は、EntFindEntityInOrg( )を使用してノード記号を取得し、次に EntQueryEx( )を呼び出してその子ノードを取得し、子が存在するかどうかを確認できます。

次の形式を使用します。

**Declare Function EntIsEntityParent Lib "heaccess.dll" Alias "\_EntIsEntityParent@12" (ByVal hSelect As Long, ByVal sigEntity As Long, ByVal sigOrg As Long) As Integer**

**変数 説明**

hSelect 選択された表のハンドル

sigEntity エンティティ記号

sigOrg 検索する組織の記号

戻りコード：

コード	説明
1 (TRUE)	エンティティは親です。
0 (False)	エンティティは親ではありません。

例：

```
isParent% = EntIsEntityParent(hSelect&, sigEntity&,HYP_NONE)
If isParent% Then MsgBox("Unable to put data in parent company.")
```

C 言語では次の形式を使用します。

**SBOOL WINAPI EntIsEntityParent(HSELECT hSelect, SIGNA sigEntity, SIGNA sigOrg)**

## EntIsModified( ) - 表が変更されているかどうかの確認

この関数は、指定された hSelect 内の表が EntUpdate( )または EntUpdateDefault( )関数によって変更されているかどうかを判断します。

次の形式を使用します。

**Declare Function EntIsModified Lib "heaccess.dll" Alias "\_EntIsModified@16" (ByVal hSelect As Long, ByVal wTabId As Integer, ByVal sigKey As Long, apiS As APISTRUCT) As Integer**

## 変数 説明

hSelect 選択された表のハンドル

wTabId 確認する表 ID、または選択した表が変更されているかどうかを確認する場合は NONE。一覧に含まれている有効な表 ID のいずれかまたは ID\_APPDEFAULT を指定できます。

sigKey NONE、または関連する表のキー。詳しくは、[340 ページの「関連する表」](#)を参照してください。

apiS NULL、または詳細情報を含んでいる apiStruct 構造体を指すポインタ

戻りコード：

コード	説明
NON-ZERO	表が変更されています。
0 (False)	表は変更されていないか、エラーが発生しています。
-1 (NONE)	エラーが発生しています。hselect が無効です。

C 言語では次の形式を使用します。

```
short WINAPI EntIsModified(HSELECT hSelect, short wTabId, SIGNA sigKey,
LPAPISTRUCT pApiS)
```

表が変更されている場合、戻りコードは通常、1 になります。ただし、変更によってステータスがどのような影響を受けるかを示すために、他のビットも設定されている場合があります。詳しくは、TOOLINC.H に定義されている NMODIFY および関連する定数を参照してください。

## EntIsSelected( ) - 表が選択されているかどうかの確認

この関数は、指定されている表が既に選択されているかどうかを判断します。引数は、表を選択するときに使用するものと同様の引数を使用します。

次の形式を使用します。

```
Declare Function EntIsSelected Lib "heaccess.dll" Alias "_EntIsSelected@20" (ByVal
hSelect As Long, ByVal wTabId As Integer, ByVal sigKey As Long, wLockMode As Integer,
apiS As APISTRUCT) As Integer
```

## 変数 説明

hSelect 選択された表のハンドル

wTabId 確認する表

sigKey NONE（ほとんどの表）または関連する表のキー。詳しくは、[340 ページの「関連する表」](#)を参照してください。

wLockMode 現在の選択モードのバッファ（APILOCK\_READONLY など）。この変数には、現在の選択モードが返されるか、表が選択されていない場合には、NONE が返されます。

## 変数 説明

apiS      NULL、または詳細情報を含んでいる apiStruct 構造体を指すポインタ

戻りコード：

コード	説明
1 (TRUE)	表が選択されています。
0 (False)	表は選択されていません。

C 言語では次の形式を使用します。

**SIGNA** WINAPI EntIsSelected(**HSELECT** hSelect, **short** wTabId, **SIGNA** sigKey, **short** FAR \* pwLockMode, **LPAPISTRUCT** pApiS)

## EntJournalExtract( ) - 仕訳帳の抽出

この関数は、1 つまたは複数の仕訳帳を抽出します。この関数は、API が仕訳帳抽出の現在のステータスを送信するために呼び出すオプションのコールバック関数を使用します。この関数を呼び出す前に、次の表を読み取り専用形式で選択する必要があります。

- デフォルトの表 (ID\_CATEGORY、ID\_ORGANIZATION、ID\_ACCOUNTS および関連表)
- ID\_JOURNALS - 仕訳帳表
- ID\_SECURITY - セキュリティ表

**注：** Visual Basic プログラマは、EntJournalExtract( )の代わりに EntJournalExtractVB( )を使用する必要があります。

EntJournalExtract()は、Hyperion Enterprise が仕訳帳の抽出に使用する関数を呼び出します。この関数は、Hyperion Enterprise が仕訳帳の抽出に使用する 2 つの方法をサポートしています。1 つは 1 つの期間の仕訳帳の抽出で、もう 1 つはすべての期間の仕訳帳の抽出です。1 つの期間の仕訳帳を抽出するには、抽出する特定の仕訳帳の記号を (lpJourData.SelectedJourSigs 内で) 指定します。すべての期間の仕訳帳を抽出するには、条件を指定します。API 関数によって、その条件に一致する仕訳帳がすべて検索され、抽出されます。具体的には、lpJourData.sCheckedInclude に指定されている仕訳帳、仕訳帳テンプレートまたは経常仕訳帳テンプレートを API 関数が内部で列挙し、その中から lpJourData.lCheckedAttribTypeStatus に指定されている条件に一致する仕訳帳を抽出します。

C 言語では次の形式を使用します。

**short** WINAPI EntJournalExtract(**HSELECT** hSelect, **LPJOUREXTRSTRUCT** lpJourData, **CALLBACKJOURLOAD** fnCallBack, **LPARAM** lParam, **LPAPISTRUCT** lpApiStruct)

変数	説明
hSelect	選択された表のハンドル
lpJourData	仕訳帳抽出用の引数を含んでいる構造体
lpJourData->dwSize	この構造体のサイズ
lpJourData->hProcess	サーバで API 関数を実行すると、[タスクステータス] ウィンドウプロセスのハンドルがここに返されます。
lpJourData->szDelimiter	フィールドの区切り文字で、通常は感嘆符 (!) )
lpJourData->sCheckedInclude	仕訳帳 (LOADEXT_JOURNALS)、標準仕訳帳テンプレート (LOADEXT_STANDARD) または経常仕訳帳テンプレート (LOADEXT_RECTMP) を抽出するためのビットマスク。lpJourData.wOperation が JOUROP_EXTRACTALLPER の場合に検索する対象を API 関数に示します。
lpJourData->sCurrentInclude	API 関数によって内部で使用されます。これは 0 に設定します。
lpJourData->lCheckedAttribTypeStatus	<p>仕訳帳の属性、種別、およびステータスとその他の抽出操作フラグを含むビットマスク。lpJourData.wOperation が JOUROP_EXTRACTALLPER のときは、属性、種別およびステータスによって、抽出する仕訳帳が決定されます。このフィールドは、以下の項目を自由に組み合わせで設定できます。</p> <p>抽出する属性：</p> <ul style="list-style-type: none"> <li>● JOURMASK_ALL_ATTRIB (すべての属性を含む)</li> <li>● JOURMASK_ATTRIB_BALANCED (貸借一致仕訳帳を含む)</li> <li>● JOURMASK_ATTRIB_UNBALANCED (貸借不一致仕訳帳を含む)</li> <li>● JOURMASK_ATTRIB_BAL_BY_ENTITY (エンティティ単位の貸借一致仕訳帳を含む)</li> </ul> <p>抽出する仕訳帳の種別：</p> <ul style="list-style-type: none"> <li>● JOURMASK_ALL_TYPE (すべての仕訳帳種別を含む)</li> <li>● JOURMASK_TYPE_REGULAR (標準仕訳帳を含む)</li> <li>● JOURMASK_TYPE_AUTOREVERSING (自動逆仕訳仕訳帳を含む)</li> <li>● JOURMASK_TYPE_TOP_LEVEL (最上位仕訳帳含む)</li> </ul> <p>抽出する仕訳帳のステータス：</p> <ul style="list-style-type: none"> <li>● JOURMASK_ALL_STATUS (すべてのステータスを含む)</li> <li>● JOURMASK_STATUS_UNPOSTED (転記前仕訳帳を含む)</li> <li>● JOURMASK_STATUS_POSTED (転記済み仕訳帳を含む)</li> <li>● JOURMASK_STATUS_AUTOREVERSED (自動逆仕訳仕訳帳を含む)</li> <li>● JOURMASK_STATUS_REVIEWED (承認済み仕訳帳を含む)</li> <li>● JOURMASK_STATUS_REVERSED (逆仕訳仕訳帳を含む)</li> <li>● JOURMASK_STATUS_LOCKED (保護済み仕訳帳を含む)</li> </ul> <p>抽出するテンプレートの種別：</p> <ul style="list-style-type: none"> <li>● JOURMASK_ALL_EMPTYTYPE (すべてのテンプレート種別を含む)</li> <li>● JOURMASK_EMPTYTYPE_STANDARD (すべての標準テンプレートを含む)</li> <li>● JOURMASK_EMPTYTYPE_RECURRING (すべての経常テンプレートを含む)</li> </ul>

変数	説明
	その他のフラグ <ul style="list-style-type: none"> <li>● DBE_REMOTE (サーバベースの抽出)</li> <li>● JOURE_APPEND (既存の抽出ファイルへの追加)</li> </ul>
IpJourData->ICurrentAttrib	0. 内部での使用。
IpJourData->wOperation	操作を示します。JOUROP_EXTRACTSINGLEPER または JOUROP_EXTRACTALLPER のどちらかである必要があります。
IpJourData->wFlags	内部で使用されます。これは 0 に設定します。
IpJourData->ISelectedJourCount	IpJourData.SelectedJourSigs 配列内の記号の数。IpJourData.wOperation が JOUROP_EXTRACTSINGLEPER である場合は 0 を使用します。
IpJourData->SelectedJourSigs	選択された仕訳帳の記号配列のアドレス。仕訳帳を仕訳帳テンプレートと区別するには、"or"演算子を使用して、JOUR_ENTRIES 定数で仕訳帳 (テンプレートではない) の記号を組み合わせます。
IpJourData->bExtractPasU	転記済み仕訳帳を抽出する場合は True (1)、そうでない場合は False
IpJourData->wExtFlag	将来使用するために予約済み。0 を使用します。
IpJourData->IpcExtFileName	仕訳帳抽出ファイルのパスとファイル名
fnCallBack	ステータス情報用のコールバック関数。DBE_REMOTE フラグが IpJourData->ICheckedAttribTypeStatus に設定されている場合、これは使用されません。
IParam	コールバック関数の引数、または DBE_REMOTE フラグが IpJourData->ICheckedAttribTypeStatus に設定されている場合は NULL。詳しくは、 <a href="#">325 ページの「CALLBACKJOUREXTRACT」</a> を参照してください。
IpApiStruct	sigCat、IStartPeriod および IEndPeriod のフィールドが設定されている apiStruct
IpApiStruct->sigCat	データ種別の記号
IpApiStruct->IStartPeriod	期間番号 (最初の期間は 0 で開始) IpJourData.wOperation が JOUROP_EXTRACTALLPER の場合は、これを 0 に設定します。
IpApiStruct->IEndPeriod	IpApiStruct.IStartPeriod と同じです。

戻りコード :

コード	説明
EXTRACT_NOERR (0)	成功
EXTRACT_CANCELLED	ユーザによってキャンセルされました。
EXTRACT_CHECK_ERRORLOG	エラーが発生し、そのエラーが ERROR.LOG に示されています。
EXTRACT_CHECKBOTH	エラーが発生し、そのエラーが ERROR.LOG と userid.ERR ファイルに示されています。

例 :

詳しくは、216 ページの「EntJournalExtractVB() - 仕訳帳の抽出」を参照してください。

## EntJournalExtractVB() - 仕訳帳の抽出

Visual Basic プログラマは、EntJournalExtract()の代わりにこの関数を使用してください。この関数は、1 つまたは複数の仕訳帳を抽出します。この関数は、API が仕訳帳抽出の現在のステータスを送信するために呼び出すオプションのコールバック関数を使用します。この関数を呼び出す前に、次の表を読み取り専用形式で選択する必要があります。

- デフォルトの表 (ID\_CATEGORY、ID\_ORGANIZATION、ID\_ACCOUNTS および関連表)
- ID\_JOURNALS - 仕訳帳表
- ID\_SECURITY - セキュリティ表

EntJournalExtractVB()は、Hyperion Enterprise が仕訳帳の抽出に使用する関数を呼び出します。この関数は、Hyperion Enterprise が仕訳帳の抽出に使用する 2 つの方法をサポートしています。1 つは 1 つの期間の仕訳帳の抽出で、もう 1 つはすべての期間の仕訳帳の抽出です。1 つの期間の仕訳帳を抽出するには、抽出する特定の仕訳帳の記号を (lpJourData.SelectedJourSigs 内で) 指定します。すべての期間の仕訳帳を抽出するには、条件を指定します。API 関数によって、その条件に一致する仕訳帳がすべて検索され、抽出されます。具体的には、lpJourData.sCheckedInclude に指定されている仕訳帳、仕訳帳テンプレートまたは経常仕訳帳テンプレートを API 関数が内部で列挙し、その中から lpJourData.lCheckedAttribTypeStatus に指定されている条件に一致する仕訳帳を抽出します。

次の形式を使用します。

```
Declare Function EntJournalExtractVB Lib "heaccess.dll" Alias
"_EntJournalExtractVB@24" (ByVal hSelect As Long, lpJourData As JourExtrStruct, ByVal
szExtFileName As String, ByVal lpfnCallBack As Any, ByVal lParam As Long, lpApiStruct
As apiStruct) As Integer
```

変数	説明
hSelect	選択された表のハンドル
lpJourData	仕訳帳抽出用の引数を含んでいる構造体
lpJourData.dwSize	この構造体のサイズ
lpJourData.hProcess	サーバで API 関数を実行すると、[タスクステータス] ウィンドウプロセスのハンドルがここに返されます。
lpJourData.szDelimiter	フィールドの区切り文字で、通常はアポストロフィ (')
lpJourData.sCheckedInclude	仕訳帳 (LOADEXT_JOURNALS)、標準仕訳帳テンプレート (LOADEXT_STANDARD) または経常仕訳帳テンプレート (LOADEXT_RECTMP) を抽出するためのビットマスク。

変数	説明
	lpJourData.wOperation が JOUROP_EXTRACTALLPER の場合に検索する対象を API 関数に示します。
lpJourData.sCurrentInclude	API 関数によって内部で使用されます。これは 0 に設定します。
lpJourData.lCheckedAttribTypeStatus	<p>仕訳帳の属性、種別、およびステータスと抽出操作の 2 つのフラグを含むビットマスク。lpJourData.wOperation が JOUROP_EXTRACTALLPER のときは、属性、種別およびステータスによって、抽出する仕訳帳が決定されます。このフィールドは、以下の項目を自由に組み合わせて設定できます。</p> <p>抽出する属性：</p> <ul style="list-style-type: none"> <li>● JOURMASK_ALL_ATTRIB（すべての属性を含む）</li> <li>● JOURMASK_ATTRIB_BALANCED（貸借一致仕訳帳を含む）</li> <li>● JOURMASK_ATTRIB_UNBALANCED（貸借不一致仕訳帳を含む）</li> <li>● JOURMASK_ATTRIB_BAL_BY_ENTITY（エンティティ単位の貸借一致仕訳帳を含む）</li> </ul> <p>抽出する仕訳帳の種別：</p> <ul style="list-style-type: none"> <li>● JOURMASK_ALL_TYPE（すべての仕訳帳種別を含む）</li> <li>● JOURMASK_TYPE_REGULAR（標準仕訳帳を含む）</li> <li>● JOURMASK_TYPE_AUTOREVERSING（自動逆仕訳仕訳帳を含む）</li> <li>● JOURMASK_TYPE_TOP_LEVEL（親仕訳帳を含む）</li> </ul> <p>抽出する仕訳帳のステータス：</p> <ul style="list-style-type: none"> <li>● JOURMASK_ALL_STATUS（すべてのステータスを含む）</li> <li>● JOURMASK_STATUS_UNPOSTED（転記前仕訳帳を含む）</li> <li>● JOURMASK_STATUS_POSTED（転記済み仕訳帳を含む）</li> <li>● JOURMASK_STATUS_AUTOREVERSED（自動逆仕訳仕訳帳を含む）</li> <li>● JOURMASK_STATUS_REVIEWED（承認済み仕訳帳を含む）</li> </ul>
	<ul style="list-style-type: none"> <li>● JOURMASK_STATUS_REVERSED（逆仕訳仕訳帳を含む）</li> <li>● JOURMASK_STATUS_LOCKED（保護済み仕訳帳を含む）</li> </ul> <p>抽出するテンプレートの種別：</p> <ul style="list-style-type: none"> <li>● JOURMASK_ALL_EMPTYTYPE（すべてのテンプレート種別を含む）</li> <li>● JOURMASK_EMPTYTYPE_STANDARD（すべての標準テンプレートをを含む）</li> <li>● JOURMASK_EMPTYTYPE_RECURRING（すべての経常テンプレートをを含む）</li> </ul> <p>その他のフラグ</p> <ul style="list-style-type: none"> <li>● DBE_REMOTE（サーバベースの抽出）</li> <li>● JOURE_APPEND（既存の抽出ファイルへの追加）</li> </ul>
lpJourData.lCurrentAttrib	0. 内部での使用。

変数	説明
IpJourData.wOperation	操作を示します。JOUROP_EXTRACTSINGLEPER または JOUROP_EXTRACTALLPER のどちらかである必要があります。
IpJourData.wFlags	内部で使用されます。これは 0 に設定します。
IpJourData.lSelectedJourCount	IpJourData.SelectedJourSigs 配列内の記号の数。 IpJourData.wOperation が JOUROP_EXTRACTALLPER である場合は 0 を使用します。IpJourData.wOperation が JOUROP_EXTRACTSINGLEPER である場合は、これが 1 以上になっている必要があります。
IpJourData.SelectedJourSigs	選択された仕訳帳の記号配列のアドレス。仕訳帳を仕訳帳テンプレートと区別するには、"or" 演算子を使用して、JOUR_ENTRIES 定数で仕訳帳（テンプレートではない）の記号を組み合わせる必要があります。
IpJourData.bExtractPasU	転記済み仕訳帳を抽出する場合は True (1)、そうでない場合は False
IpJourData.wExtFlag	将来使用するために予約済み。0 を使用します。
lpszExtFileName	仕訳帳抽出ファイルのパスとファイル名
lpfnCallBack	ステータス情報を取得するコールバック関数、または NULL。DBE_REMOTE フラグが IpJourData.lCheckedAttribTypeStatus に設定されている場合、これは使用されません。
lParam	コールバック関数の引数、または DBE_REMOTE フラグが IpJourData.lCheckedAttribTypeStatus に設定されている場合は NULL。詳しくは、 <a href="#">325 ページの「CALLBACKJOUREXTRACT」</a> を参照してください。
IpApiStruct	sigCat、IStartPeriod および IEndPeriod のフィールドが設定されている apiStruct
IpApiStruct->sigCat	データ種別の記号
IpApiStruct->IStartPeriod	期間番号（最初の期間は 0 で開始）IpJourData.wOperation が JOUROP_EXTRACTALLPER の場合は、これを 0 に設定します。
IpApiStruct->IEndPeriod	IpApiStruct.IStartPeriod と同じです。

戻りコード：

コード	説明
Extract_NOERR (0)	成功
Extract_CANCELLED	ユーザによってキャンセルされました。
Extract_CHECK_ERRORLOG	エラーが発生し、そのエラーが ERROR.LOG に示されています。
Extract_CHECKBOTH	エラーが発生し、そのエラーが ERROR.LOG と userid.ERR ファイルに示されています。

C 言語では次の形式を使用します。

**short WINAPI EntJournalExtractVB(HSELECT hSelect, LPJOURNEXTRSTRUCT lpJourData, LPCSTR lpszExtFileName, CALLBACKJOURLOAD fnCallBack, LPARAM lParam, LPAPISTRUCT lpApiStruct)**

例 :

```
Private Sub JournalExtract(hSelect As Long, : sigCat As Long, szFile As String)
Dim apiStruct As apiStruct
Dim jourData As JourExtrStruct
Dim hApp&
Dim wRet As Integer
Const MAXJOURNALS = 100 '配列のサイズを100の仕訳帳用に設定
Dim sigArr(MAXJOURNALS+1) As Long
'配列は連動ではなく固定であるため、G.C.中にVBによって再配置されることはありません
'apiStructを設定
hApp& = EntGetHappFromSelect(hSelect&)
Call EntInitApiStruct(hApp&, apiStruct)
apiStruct.sigCat = sigCat&
apiStruct.lStartPeriod = 0 'apiStructの最初の期間。lEndPeriod = 0
jourData.dwSize = LenB(jourData)
jourData.hProcess = 0
jourData.szDelimiter = Asc("!")
'標準仕訳帳、標準テンプレート、および
'経常テンプレートを抽出
jourData.sCheckedInclude = 0
jourData.sCheckedInclude =
jourData.sCheckedInclude Or LOADEXT_JOURNALS
jourData.sCheckedInclude =
jourData.sCheckedInclude Or LOADEXT_STANDTMP
jourData.sCheckedInclude =
jourData.sCheckedInclude Or LOADEXT_RECTMP
jourData.sCurrentInclude = 0
'属性、種別およびステータスを設定
jourData.lCheckedAttribTypeStatus = 0
jourData.lCheckedAttribTypeStatus =
jourData.lCheckedAttribTypeStatus Or JOURMASK_ALL_ATTRIB
jourData.lCheckedAttribTypeStatus =
jourData.lCheckedAttribTypeStatus Or JOURMASK_ALL_TYPE
jourData.lCheckedAttribTypeStatus =
jourData.lCheckedAttribTypeStatus Or JOURMASK_ALL_STATUS
jourData.bExtractPasU = 0
'転記済み仕訳帳は転記前仕訳帳として抽出しない
jourData.lCurrentAttrib = 0
jourData.wFlags = 0
jourData.wExtFlag = 0
jourData.cExtFileName = 0 'szFile$を引数として関数に渡す
'表を選択
wRet% = EntSelectAdd(HYP_ID_JOURNALS, sigCat, hSelect&,
APILOCK_READONLY, 0, 0, 0, 0)
wRet% = EntSelectAdd(HYP_ID_SECURITY or HYP_ID_ASSOC, sigCat,
hSelect&, APILOCK_READONLY, 0, 0, 0, 0)
If List1.SelCount>0 Then '選択した仕訳帳の抽出 jourData.wOperation =
JOUROP_EXTRACTSINGLEPER
'apiStruct.lStartPeriod = apiStruct.lEndPeriodが正しい期間であることを確認
lCount& = List1.SelCount
If lCount& > MAXJOURNALS Then lCount& = MAXJOURNALS
```

```

    jourData.iSelectedJourCount = lCount&
    For i = 0 To lCount& - 1
        'この例では、List1.List(i) = sigJournalまたはJOUR_ENTRIES
        'それ以外はList1.List(i) = sigTemplate
        sigArr(i) = List1.List(i)
    Next i
    jourData.lpSelectedJourSigs =
        EntGetVarAddr(sigArr(0)) '連動配列は使用しない
Else
    '以前に設定した条件
    'jourData.sCheckedIncludeおよび
    jourData.ICheckedAttribTypeStatusに基づいてすべての仕訳帳を抽出
    jourData.wOperation = JOUROP_EXTRACTALLPER
    jourData.iSelectedJourCount = 0
    jourData.lpSelectedJourSigs = 0
End If
wRet% = EntJournalExtractVB(hSelect&, jourData, szFile$, 0, 0,
    apiStruct)
hSelect& = EntUnSelect0(hSelect&, HYP_ID_SECCLASS, sigCat&, 0, wRet%, 0)
hSelect& = EntUnSelect0(hSelect&, HYP_ID_JOURNALS, sigCat&, 0, wRet%, 0)
End Sub

```

## EntJournalLoad( ) - 仕訳帳の読み込み

この関数は、仕訳帳の読み込みを実行します。この関数には、仕訳帳読み込みの現在のステータスを送信するために API が呼び出すコールバック関数が必要です。この関数を呼び出す前に、デフォルトの表の他に、セキュリティ表、仕訳帳の期間情報表および仕訳帳入力表を選択する必要があります。

**注：** Visual Basic プログラマは、EntJournalLoad( )の代わりに EntJournalLoadVB( )を使用する必要があります。

C 言語では次の形式を使用します。

**short** WINAPI **EntJournalLoad**(HSELECT hSelect, LPJOURLOADSTRUCT lpJourData, CALLBACKJOURLOAD fnCallBack, LONG lParam, LPAPISTRUCT lpApiStruct)

変数	説明
hSelect	選択された表のハンドル
lpJourData	仕訳帳読み込み用の引数を含んでいる構造体
lpJourData->dwSize	この構造体のサイズ
lpJourData->hProcess	[タスクステータス] ウィンドウプロセスのハンドル。これは、サーバ上での実行時にプロセスハンドルを返します。
lpJourData->szDelimiter	フィールドの区切り文字で、通常はカンマ (,)
lpJourData->sCheckedInclude	仕訳帳 (LOADEXT_JOURNALS) および標準または経常仕訳帳テンプレート (LOADEXT_STANDARD または LOADEXT_RECTMP) を読み込みます。

変数	説明
lpJourData->sCurrentInclude	読み込みに含める仕訳帳または仕訳帳テンプレートの種別を示します。内部で使用されます。
lpJourData->lCheckedAttribTypeStatus	<p>仕訳帳属性、種別およびステータスを格納するビットマスク。以下の項目を自由に組み合わせて設定できます。</p> <p>抽出する属性：</p> <ul style="list-style-type: none"> <li>● JOURMASK_ALL_ATTRIB（すべての属性を含む）</li> <li>● JOURMASK_ATTRIB_BALANCED（貸借一致仕訳帳を含む）</li> <li>● JOURMASK_ATTRIB_UNBALANCED（貸借不一致仕訳帳を含む）</li> <li>● JOURMASK_ATTRIB_BAL_BY_ENTITY（エンティティ単位の貸借一致仕訳帳を含む）</li> </ul> <p>抽出する仕訳帳の種別：</p> <ul style="list-style-type: none"> <li>● JOURMASK_ALL_TYPE（すべての仕訳帳種別を含む）</li> <li>● JOURMASK_TYPE_REGULAR（標準仕訳帳を含む）</li> <li>● JOURMASK_TYPE_AUTOREVERSING（自動逆仕訳仕訳帳を含む）</li> <li>● JOURMASK_TYPE_TOP_LEVEL（最上位仕訳帳含む）</li> </ul> <p>抽出する仕訳帳のステータス：</p> <ul style="list-style-type: none"> <li>● JOURMASK_ALL_STATUS（すべてのステータスを含む）</li> <li>● JOURMASK_STATUS_UNPOSTED（転記前仕訳帳を含む）</li> <li>● JOURMASK_STATUS_POSTED（転記済み仕訳帳を含む）</li> <li>● JOURMASK_STATUS_AUTOREVERSED（自動逆仕訳仕訳帳を含む）</li> <li>● JOURMASK_STATUS_REVIEWED（承認済み仕訳帳を含む）</li> </ul>
	<ul style="list-style-type: none"> <li>● JOURMASK_STATUS_REVERSED（逆仕訳仕訳帳を含む）</li> <li>● JOURMASK_STATUS_LOCKED（保護済み仕訳帳を含む）</li> </ul> <p>抽出するテンプレートの種別：</p> <ul style="list-style-type: none"> <li>● JOURMASK_ALL_EMPTYTYPE（すべてのテンプレート種別を含む）</li> <li>● JOURMASK_EMPTYTYPE_STANDARD（すべての標準テンプレートを含む）</li> <li>● JOURMASK_EMPTYTYPE_RECURRING（すべての経常テンプレートを含む）</li> <li>● DBE_REMOTE（サーバベースの抽出）</li> </ul>
lpJourData->lCurrentAttrib	属性を格納します。内部で使用されます。
lpJourData->lCurrentType	種別を格納します。内部で使用されます。
lpJourData->lCurrentStatus	ステータスを格納します。内部で使用されます。
lpJourData->buse18Format	リリース 1.8 の形式を使用する場合は TRUE、それ以外の場合は False
lpJourData->sAutoReversing	転記済みおよび転記前自動逆仕訳仕訳帳を確認するためのフラグ。NULL に設定します。これは内部で使用されます。
lpJourData->wExtFlag	将来使用するために予約済み。

変数	説明
lpJourData->lpLoadFileName	仕訳帳読み込みファイルのパスと名前
fnCallBack	ステータス情報用のコールバック関数。DBE_REMOTE フラグが lpJourData->lCheckedAttribTypeStatus に設定されている場合、これは使用されません。
lParam	コールバック関数の引数、または DBE_REMOTE フラグが lpJourData->lCheckedAttribTypeStatus に設定されている場合は NULL。
lpApiStruct	NULL に設定します。

戻りコード：

コード	説明
LOAD_NOERR (0)	成功
ゼロ以外	失敗

例：

詳しくは、[222 ページの「EntJournalLoad\( \) - 仕訳帳の読み込み」](#)を参照してください。

## EntJournalLoad( ) - 仕訳帳の読み込み

Visual Basic プログラマは、EntJournalLoad( )の代わりにこの関数を使用してください。この関数は、仕訳帳の読み込みを実行します。この関数には、仕訳帳読み込みの現在のステータスを送信するために API が呼び出すコールバック関数が必要です。この関数を呼び出す前に、デフォルトの表の他に、セキュリティ表、仕訳帳の期間情報表および仕訳帳入力表を選択する必要があります。

次の形式を使用します。

```
Declare Function EntJournalLoadVB Lib "heaccess.dll" Alias "_EntJournalLoadVB@24"
(ByVal hSelect As Long, lpJourData As JourLoadStruct, ByVal lpszLoadFileName As String,
ByVal lpfnCallBack As Any, ByVal lParam As Long, lpApiStruct As apiStruct) As Integer
```

変数	説明
hSelect	選択された表のハンドル
lpJourData	仕訳帳読み込み用の引数を含んでいる構造体
lpJourData.dwSize	この構造体のサイズ
lpJourData.hProcess	[タスクステータス] ウィンドウプロセスのハンドル。これは、サーバ上での実行時にプロセスハンドルを返します。
lpJourData.szDelimiter	フィールドの区切り文字で、通常はカンマ (,)

変数	説明
lpJourData.sCheckedInclude	仕訳帳および標準または経常仕訳帳テンプレートを読み込みます。
lpJourData.sCurrentInclude	仕訳帳または仕訳帳テンプレートの種別を示します。
lpJourData.lCheckedAttribTypeStatus	<p>仕訳帳属性、種別およびステータスを格納するビットマスク。以下の項目を自由に組み合わせて設定できます。</p> <p>抽出する属性</p> <ul style="list-style-type: none"> <li>● JOURMASK_ALL_ATTRIB (すべての属性を含む)</li> <li>● JOURMASK_ATTRIB_BALANCED (貸借一致仕訳帳を含む)</li> <li>● JOURMASK_ATTRIB_UNBALANCED (貸借不一致仕訳帳を含む)</li> <li>● JOURMASK_ATTRIB_BAL_BY_ENTITY (エンティティ単位の貸借一致仕訳帳を含む)</li> </ul> <p>抽出する仕訳帳の種別</p> <ul style="list-style-type: none"> <li>● JOURMASK_ALL_TYPE (すべての仕訳帳種別を含む)</li> <li>● JOURMASK_TYPE_REGULAR (標準仕訳帳を含む)</li> <li>● JOURMASK_TYPE_AUTOREVERSING (自動逆仕訳仕訳帳を含む)</li> <li>● JOURMASK_TYPE_TOP_LEVEL (最上位仕訳帳を含む)</li> </ul> <p>抽出する仕訳帳のステータス</p> <ul style="list-style-type: none"> <li>● JOURMASK_ALL_STATUS (すべてのステータスを含む)</li> <li>● JOURMASK_STATUS_UNPOSTED (転記前仕訳帳を含む)</li> <li>● JOURMASK_STATUS_POSTED (転記済み仕訳帳を含む)</li> <li>● JOURMASK_STATUS_AUTOREVERSED (自動逆仕訳仕訳帳を含む)</li> <li>● JOURMASK_STATUS_REVIEWED (承認済み仕訳帳を含む)</li> </ul>
	<ul style="list-style-type: none"> <li>● JOURMASK_STATUS_REVERSED (逆仕訳仕訳帳を含む)</li> <li>● JOURMASK_STATUS_LOCKED (保護済み仕訳帳を含む)</li> </ul> <p>抽出するテンプレートの種別</p> <ul style="list-style-type: none"> <li>● JOURMASK_ALL_TEMPTYPE (すべてのテンプレート種別を含む)</li> <li>● JOURMASK_TEMPTYPE_STANDARD (すべての標準テンプレートをを含む)</li> <li>● JOURMASK_TEMPTYPE_RECURRING (すべての経常テンプレートをを含む)</li> <li>● DBE_REMOTE (サーバベースの抽出)</li> </ul>
lpJourData.lCurrentAttrib	属性を格納します。
lpJourData.lCurrentType	種別を格納します。
lpJourData.lCurrentStatus	ステータスを格納します。
lpJourData.buse18Format	読み込みファイルがリリース 1.8 の形式の場合は TRUE、それ以外の場合は False

変数	説明
lpJourData.sAutoReversing	転記済みおよび転記前自動逆仕訳仕訳帳を確認するためのフラグ
lpJourData.wExtFlag	将来使用するために予約済み
szLoadFileName	仕訳帳読み込みファイルのパス名
fnCallBack	ステータス情報用のコールバック関数。DBE_REMOTE フラグが lpJourData.lCheckedAttribTypeStatus に設定されている場合、これは使用されません。
lParam	コールバック関数の引数、または DBE_REMOTE フラグが lpJourData.lCheckedAttribTypeStatus に設定されている場合は NULL。
lpApiStruct	仕訳帳読み込み引数の残りを含んでいる構造体。

戻りコード：

コード	説明
LOAD_NOERR (0)	成功
ゼロ以外	失敗

例：

```
Private Sub JournalLoad(hSelect As Long, sigCat As Long, szFile As String)
Dim jourData As JourLoadStruct
Dim wRet%
jourData.dwSize = LenB(jourData)      '44が指定されている必要があります
jourData.hProcess = 0
jourData.szDelimiter = 33
jourData.sCheckedInclude = LOAEXT_JOURNALS
'通常仕訳帳
jourData.sCurrentInclude = 0
'属性、種別およびステータスを設定
jourData.lCheckedAttribTypeStatus = 0
jourData.lCheckedAttribTypeStatus =
jourData.lCheckedAttribTypeStatus Or JOURMASK_ALL_ATTRIB
jourData.lCheckedAttribTypeStatus =
    jourData.lCheckedAttribTypeStatus Or JOURMASK_TYPE_REGULAR
jourData.lCheckedAttribTypeStatus =
    jourData.lCheckedAttribTypeStatus Or JOURMASK_STATUS_UNPOSTED
jourData.lCurrentAttrib = 0
jourData.lCurrentStatus = 0
jourData.buse18Format = 0
jourData.lpcLoadFileName = 0    '代わりにszFileを引数として関数に渡す
'仕訳帳表を選択
wRet% = EntSelectAdd(HYP_ID_JOURNALS, sigCat&, hSelect&,
    APILOCK_READWRITE, 0, 0, 0, 0)
wRet% = EntSelectAdd(HYP_ID_SECURITY or HYP_ID_ASSOC, sigCat&,
    hSelect&, APILOCK_READONLY, 0, 0, 0, 0)
wRet% = EntSelectAdd(HYP_ID_JOURNAL_PERIOD_INFO Or HYP_ID_ASSOC,
    sigCat&, hSelect&, APILOCK_READONLY, 0, 0, 0, 0)
```

```

wRet% = EntJournalLoadVB(hSelect&, jourData, szFile$, AddressOf
    MyCallBack, 0, apiStruct)
End Sub

Public Function MyCallBack(ByVal addressOfFileNameCString As Long, ByVal
addressOfJourNameCString As Long,ByVal fSize As Long,ByVal fPos As Long,
ByVal lParam As Long) As Integer
Dim wRet%, wLen%
Dim szFile$, szJourName$
Dim szMsg$
'Cスタイルの文字列をVisual Basicの文字列に変換
wLen% = EntVBGetCStrLen(addressOfFileNameCString)
szFile$ = Space(wLen%)
wRet% = EntVBCopyStr(szFile$, addressOfFileNameCString, wLen%)
wLen% = EntVBGetCStrLen(addressOfJourNameCString)
szJourName$ = Space(wLen%)
wRet% = EntVBCopyStr(szJourName$, addressOfJourNameCString,
    wLen%)
szMsg$ = "Now Loading " & szJourName$ & " From " & szFile$
    frmJourLoad.List1.AddItem (szMsg$)
MyCallBack = 0
End Function

```

C 言語では次の形式を使用します。

```

short WINAPI EntJournalLoadVB(HSELECT hSelect, LPJOURLOADSTRUCT
lpJourLoad, LPCSTR lpszLoadFileName, CALLBACKJOURLOAD fnCallBack, LONG
lParam, LPAPISTRUCT lpApiStruct)

```

## EntLogicAttach( ) - 計算式の追加

この関数は、ロジックブロックを構築してアプリケーションに追加します。この操作は、計算式を実行するか、計算式に依存する **EntIsAccountInput( )**などの関数を呼び出す前に行う必要があります。**EntLogicAttach( )**を呼び出す前に、ロジック表 (ID\_LOGIC) を選択する必要があります。ロジックブロックの使用が終わったら、**EntLogicDetach( )**を呼び出す必要があります。

**注：** ID\_DATAFILE 表を選択すると、ロジックブロックが自動的に構築され、追加されます。データファイル表を選択した場合には **EntLogicAttach( )**を呼び出す必要はありません。

ロジックブロックとは、特定のデータ種別およびロジックに適した計算式を結合したセットです。これは、指定されたロジック、入力ロジック、および'USE'ステートメントを使用して含められたロジックの pcode 表を結合します。また、すべてのデータ種別に適用される計算式を含む、指定されたデータ種別の計算式を結合します。

次の形式を使用します。

```

Declare Function EntLogicAttach Lib "heaccess.dll" Alias "_EntLogicAttach@12" (ByVal
hSelect As Long, ByVal sigCat As Long, ByVal sigMethod As Long) As Integer

```

変数	説明
hSelect	選択された表のハンドル
sigCat	データ種別の記号
sigMethod	ロジックの記号

戻りコード：

コード	説明
0	成功
NONE	エラー発生

C 言語では次の形式を使用します。

```
short WINAPI EntLogicAttach(HSELECT hSelect, SIGNA sigCat, SIGNA sigMethod)
```

## EntLogicCompile( ) - ロジックのコンパイル

EntLogicCompile( )は、選択された特定のロジックまたはすべての無効なロジックをコンパイルします。この関数を呼び出す前に、次の表を選択する必要があります。

- デフォルトの表
- ID\_ASSOC を含む ID\_SECURITY（読み取り専用）
- ID\_ASSOC を含む ID\_LOGIC（読み取り／書き込み）。サーバで実行する場合、ロジック表はマルチユーザの競合を避けるために、読み取り専用として選択されます。

次の形式を使用します。

```
Declare Function EntLogicCompile Lib "heaccess.dll" Alias "_EntLogicCompile@12"  
(ByVal hSelect As Long, lpLogicCompile As LOGICCOMPSTRUCTSTRUCT,  
lpApiStruct As apiStruct) As Integer
```

変数	説明
hSelect	選択された表のハンドル
lpLogicCompile	ロジックのコンパイル用引数を含んでいる構造体
lpLogicCompile.dwSize	この構造体のサイズ
lpLogicCompile.hProcess	[タスクステータス] ウィンドウプロセスのハンドル。サーバで実行する場合はプロセスハンドルを返し、ローカルで実行する場合は NULL を返します。
lpLogicCompile.bRemoteFlag	ロジックをローカルでコンパイルするか、サーバでコンパイルするかを示します。ローカルでコンパイルする場合は FALSE、サーバの場合は TRUE になります。

変数	説明
lpLogicCompile.fnCallBack	ステータス情報のコールバック関数のアドレス。コールバック関数を使用しない場合は NULL (0) を渡すことができます。
lpLogicCompile.lParam	コールバック関数の引数
lpLogicCompile.bAllMethods	これはすべてのロジックをコンパイルするように TRUE に設定されます。noMethods と lpMethodArr は上書き（無視）されます。
lpLogicCompile.wCurrMethod	内部での使用のために予約済み
lpLogicCompile.wExtendFlag	将来使用するために予約済み
lpLogicCompile.noMethods	lpMethodArr で選択されているロジックの数を渡すために使用します（bAllMethods が TRUE に設定されている場合は 0）。
lpLogicCompile.lpMethodArr	コンパイルするロジックの記号を渡すために使用します（bAllMethods が TRUE に設定されている場合は無視されます）。
lpApiStruct	NULL に設定できる API 構造体。PCODE のエラーログの記録が必要な場合は、lpApiStruct.u_ApiUpd.bPcodeOnlyLogErr を TRUE に設定します。

戻りコード：

コード	説明
0	成功
NONE	エラー発生

C 言語では次の形式を使用します。

```
short WINAPI EntLogicCompile(HSELECT hSelect, LPLOGICCOMPSTRUCT
lpLogicCompile, LPAPISTRUCT lpApiStruct)
```

## EntLogicDetach( ) - 計算式の分離

この関数は、ロジックブロックを分離します。EntLogicAttach( )で作成されたブロックの使用が完了したら、この関数を呼び出します。ロジックブロックとは、特定のデータ種別およびロジックに適した計算式を結合したセットです。これは、指定されたロジック、入力ロジック、および'USE'ステートメントを使用して含まれたロジックの pcode 表を結合します。また、すべてのデータ種別に適用される計算式を含む、指定されたデータ種別の計算式を結合します。

次の形式を使用します。

```
Declare Function EntLogicDetach Lib "heaccess.dll" Alias "_EntLogicDetach@12" (ByVal
hSelect As Long, ByVal sigCat As Long, ByVal sigMethod As Long) As Integer
```

変数	説明
hSelect	選択された表のハンドル
sigCat	データ種別の記号

## 変数 説明

sigMethod ロジックの記号

戻りコード：

コード	説明
0	成功
NONE	エラー発生

C 言語では次の形式を使用します。

```
short WINAPI EntLogicDetach(HSELECT hSelect, SIGNA sigCat, SIGNA sigMethod)
```

## EntLogicDiscard() - ロジックブロックの破棄

この関数は、破棄するようにフラグが付けられているロジックブロックをすべて破棄します。ロジックブロックとは、特定のデータ種別およびロジックに適した計算式を結合したセットです。これは、指定されたロジック、入力ロジック、および'USE'ステートメントを使用して含められたロジックの pcode 表を結合します。また、すべてのデータ種別に適用される計算式を含む、指定されたデータ種別の計算式を結合します。

次の形式を使用します。

```
Declare Function EntLogicDiscard Lib "heaccess.dll" Alias "_EntLogicDiscard@12" (ByVal hApp As Long, ByVal sigCat As Long, ByVal sigMethod As Long) As Integer
```

## 変数 説明

hApp アプリケーションハンドル

sigCat データ種別の記号または NONE。未使用。

sigMethod ロジックの記号または NONE。未使用。

戻りコード：

この関数は、常にゼロを返します。

C 言語では次の形式を使用します。

```
short WINAPI EntLogicDiscard(HAPP hApp, SIGNA sigCat, SIGNA sigMethod)
```

## EntLogicExport( ) - 計算式のエクスポート

この関数は、計算式をアプリケーションからエクスポートします。この関数を呼び出す前に、次の表を選択する必要があります。

- デフォルトの表
- ID\_SECURITY とそれに関連する表（読み取り専用）

- ID\_LOGIC - ロジック表（読み取り専用）

**注：** Visual Basic プログラマは、EntLogicExport( )の代わりに  
EntLogicExportVB( )を使用してください。

C 言語では次の形式を使用します。

**short WINAPI EntLogicExport(HSELECT hSelect, LPLOGICEXPORT lpLogicExport ,  
LPAPISTRUCT lpApiStruct)**

変数	説明
hSelect	選択された表のハンドル
lpLogicExport	ロジックのエクスポート用引数を含んでいる構造体
lpLogicExport->dsize	構造体のサイズ
lpLogicExport->hProcess	[タスクステータス] ウィンドウプロセスのハンドル。サーバで実行する場合はプロセスハンドルを返し、ローカルで実行する場合は NULL を返します。
lpLogicExport->delimiter	フィールドの区切り文字で、通常はカンマ (,)
lpLogicExport->sigEntry	現在のエクスポートロジックを選択するために使用します。
lpLogicExport->IFlag	次のいずれかになる必要があります。 <ul style="list-style-type: none"> <li>● LOGEXP_ISMETHOD</li> <li>● LOGEXP_ISMETHOD</li> <li>● LOGEXP_ISFUNCTION</li> <li>● LOGEXP_ISRULE</li> <li>● LOGEXP_EXPALL</li> <li>● LOGEXP_NOHEADER</li> <li>● LOGEXP_REMOTE</li> </ul> ロジック、関数またはルールが設定されている場合、LOGEXP_EXPALL は使用しないでください。
lpLogicExport->wExtFlags	将来使用するために予約済み
lpLogicExport->iSelectedMethCount	選択されているメソッド数
lpLogicExport->SelectedMethSigs	メソッド記号の配列
lpLogicExport->iSelectedRuleCount	選択されているルール数
lpLogicExport->SelectedRuleSigs	ルール記号の配列
lpLogicExport->iSelectFuncCount	選択されている関数の数

変数	説明
lpLogicExport->SelectedFuncSigs	関数記号の配列
lpLogicExport->fnCallback	ステータス情報用のコールバック関数
lpLogicExport->lParam	コールバック関数の引数
lpLogicExport->FileName	ファイル名。 注： 指定したファイルが既に存在する場合は、既存のファイルに追加されます。
lpApiStruct	NULL。API 互換性のための構造体

戻りコード：

コード	説明
0	成功
NONE	エラー発生

[230 ページの「EntLogicExportVB\( \) - 計算式のエクスポート」](#)の例を参照してください。

## EntLogicExportVB( ) - 計算式のエクスポート

Visual Basic プログラマは、EntLogicExport( )の代わりにこの関数を使用してください。この関数は、計算式をアプリケーションからエクスポートします。この関数を呼び出す前に、次の表を選択する必要があります。

- デフォルトの表
- ID\_SECURITY とそれに関連する表（読み取り専用）
- ID\_LOGIC - ロジック表（読み取り専用）

Microsoft Visual Basic では、文字列のアドレスを確実に lpLogicExport 構造体に渡すことはできません。EntLogicExportVB( )を使用すると、エクスポートファイルのパスとファイル名を、lpLogicExport.FileName の代わりに関数の引数として渡すことができます。

次の形式を使用します。

```
Declare Function EntLogicExportVB Lib "heaccess.dll" Alias "_EntLogicExportVB@16"
(ByVal hSelect As Long, lpLogicExport As LOGICEXPORTSTRUCT, lpApiStruct As
apiStruct, ByVal szFileName As String) As Integer
```

変数	説明
hSelect	選択された表のハンドル
lpLogicExport	ロジックのエクスポート用引数を含んでいる構造体

変数	説明
lpLogicExport.dwsize	構造体のサイズ
lpLogicExport.hProcess	[タスクステータス] ウィンドウプロセスのハンドル。サーバ上での実行時にプロセスハンドルを返します。
lpLogicExport.delimiter	フィールドの区切り文字で、通常はカンマ (,)
lpLogicExport.sigEntry	エクスポートされる現在のロジックを選択するために内部で使用されます。NONE に設定します。
lpLogicExport.IFlag	次のいずれかになる必要があります。 <ul style="list-style-type: none"> <li>● LOGEXP_ISMETHOD</li> <li>● LOGEXP_ISMETHOD</li> <li>● LOGEXP_ISFUNCTION</li> <li>● LOGEXP_ISRULE</li> <li>● LOGEXP_EXPALL</li> <li>● LOGEXP_NOHEADER</li> <li>● LOGEXP_REMOTE</li> </ul> ロジック、関数またはルールが設定されている場合、LOGEXP_EXPALL は使用しないでください。
lpLogicExport.wExtFlags	将来使用するために予約済み
lpLogicExport.iSelectedMethCount	選択されているメソッド数
lpLogicExport.SelectedMethSigs	メソッド記号の配列
lpLogicExport.iSelectedRuleCount	選択されているルール数
lpLogicExport.SelectedRuleSigs	ルール記号の配列
lpLogicExport.iSelectFuncCount	選択されている関数の数
lpLogicExport.SelectedFuncSigs	関数記号の配列
lpLogicExport.fnCallback	ステータス情報用のコールバック関数
lpLogicExport.IParam	コールバック関数の引数
szFileName	読み込みファイルの名前。 <b>注：</b> 指定したファイルが既に存在する場合は、既存のファイルに追加されます。
lpApiStruct	API 互換性のための構造体

戻りコード：

コード	説明
0	成功
NONE	エラー発生

例：

```

Private Sub ExportLogic(hSelect As Long, szFile As String)
Dim logicExport As LOGICEXPORTSTRUCT
Dim apiStruct As apiStruct
Dim hApp&
Dim wRet%
logicExport.dwSize = LenB(logicExport)
logicExport.hProcess = 0
logicExport.delimiter = Asc("!")
logicExport.sigEntry = 0
logicExport.lFlag = LOGEXP_EXPALL
logicExport.wExtFlags = 0
logicExport.iSelectedMethCount = 0
logicExport.lpSelectedMethSigs = 0
logicExport.iSelectedRuleCount = 0
logicExport.lpSelectedRuleSigs = 0
logicExport.iSelectedFuncCount = 0
logicExport.lpSelectedFuncSigs = 0
logicExport.fileName = 0
logicExport.fnCallBack = FuncPtrToLong (AddressOf MyCallBack)
logicExport.lParam = 0
hApp& = EntGetHappFromSelect (hSelect&)
Call EntInitApiStruct (hApp&, apiStruct)
wRet% = EntSelectAdd (HYP_ID_SECURITY or HYP_ID_ASSOC, HYP_NONE,
    hSelect&, APILOCK_READONLY, 0, 0, 0, 0)
wRet% = EntSelectAdd (HYP_ID_LOGIC, HYP_NONE, hSelect&,
    APILOCK_READONLY, 0, 0, 0, 0)
wRet% = EntLogicExportVB (hSelect&, logicExport, apiStruct,
    szFile$)
End Sub

'AddressOf()の結果を (fnPtrとして) この関数に渡します。
'この関数はその結果をlongに変換するので、
As Anyではなくlongとして宣言されているフィールドまたは引数に割り当てることができま
す。
Public Function FuncPtrToLong(ByVal fnPtr As Long) As Long
FuncPtrToLong = fnPtr
End Function

Public Function MyCallBack(ByVal addressOfFileNameCString As Long, ByVal
addressOfMethodCString As Long, ByVal unused1 As Long, ByVal unused2 As
Long, ByVal lParam As Long) As Integer
Dim wRet%, wLen%
Dim szFile$, szMethod$
Dim szMsg$
'Cスタイルの文字列をVisual Basicの文字列にコピーする
wLen% = EntVBGetCStrLen (addressOfFileNameCString)
szFile$ = Space(wLen%)
wRet% = EntVBCopyStr (szFile$, addressOfFileNameCString, wLen%)
wLen% = EntVBGetCStrLen (addressOfMethodCString)
szMethod$ = Space(wLen%)
wRet% = EntVBCopyStr (szMethod$, addressOfMethodCString, wLen%)
szMsg$ = "Now Extracting..." & szMethod$ & " To " & szFile$
List1.AddItem (szMsg$)
MyCallBack = 0 '0を返す
End Function

```

C 言語では次の形式を使用します。

short WINAPI EntLogicExportVB(HSELECT hSelect, LPLOGICEXPORT lpLogicExport, LPAPISTRUCT lpApiStruct, LPCSTR szFileName)

## EntLogicImport( ) - 計算式のインポート

この関数は、計算式をアプリケーションにインポートします。この関数を呼び出す前に、次の表を選択する必要があります。

- デフォルトの表
- ID\_SECURITY とそれに関連する表（読み取り専用）
- ID\_LOGIC - ロジック表（読み取り／書き込み）。計算式のインポートをサーバで実行する場合は、読み取り専用のロジック表を選択し、マルチユーザの競合問題を避けるようにしてください。

**注：** Visual Basic プログラマは、EntLogicImport( )の代わりに EntLogicImportVB( )を使用してください。

C 言語では次の形式を使用します。

short WINAPI EntLogicImport(HSELECT hSelect, LPLOGICIMPORT lpLogicImport, LPAPISTRUCT lpApiStruct)

変数	説明
hSelect	選択された表のハンドル
lpLogicImport	計算式インポート用の引数を含んでいる構造体
lpLogicImport->dwsz	構造体のサイズ
lpLogicImport->hProcess	[タスクステータス] ウィンドウプロセスのハンドル。サーバで実行する場合はプロセスハンドルを返し、ローカルで実行する場合は NULL を返します。
lpLogicImport->delimiter	フィールドの区切り文字で、通常はカンマ (,)
lpLogicImport->bReplace	計算式を交換する場合は TRUE
lpLogicImport->sigMethod	現在のロジックの記号*
lpLogicImport->iLogicType	LOGIC_CHART、LOGIC_CONSOL または LOGIC_TRANS のいずれかの種類
lpLogicImport->uMethod	ロジックをインポートする場合は TRUE
lpLogicImport->uRule	ルールをインポートする場合は TRUE
lpLogicImport->uCustomFunc	カスタム関数をインポートする場合は TRUE

変数	説明
lpLogicImport->uLogicComp	計算式をコンパイルする場合は TRUE
lpLogicImport->bNoHeader	ヘッダーなし
lpLogicImport->wImportFlag	計算式をローカルでインポートするか、サーバでインポートするかを示します。サーバで処理する場合は LOGIC_REMOTE に設定します。
lpLogicImport->wExtendFlag	将来使用するために予約済み
lpLogicImport->fnCallback	ステータス情報用のコールバック関数
lpLogicImport->lParam	コールバック関数の引数
lpLogicImport->iFileListCnt	ファイル一覧表 lpLogicImport->lpFileList の要素の数
lpLogicImport->lpFileList	選択されているファイル名の配列（各最大サイズは 260 バイト）。
lpApiStruct	API 互換性のための構造体

\*sigMethod フィールドは、現在選択されているロジックの記号に設定する必要があります。ID\_FORMULAS 表とそれに関連付けられている表（ID\_ASSOC）を選択し、ロジック記号 lpLogicImport.sigMethod を sigKey 引数として指定します。アクセスロジックを（APILOCK\_READONLY ではなく）APILOCK\_READWRITE として指定します。記号 0 は、デフォルトの入力ロジックの記号です。

戻りコード：

コード	説明
0	成功
NONE	エラー発生
LOGIC_COMPERR (1)	ロジックのインポート中にエラーが発生したロジックの確認

例：

詳しくは、[234 ページの「EntLogicImportVB\( \) - 計算式のインポート」](#)を参照してください。

## EntLogicImportVB( ) - 計算式のインポート

Visual Basic プログラマは、EntLogicImport( )の代わりにこの関数を使用してください。この関数は、計算式をアプリケーションにインポートします。この関数を呼び出す前に、次の表を選択する必要があります。

- デフォルトの表
- ID\_SECURITY とそれに関連する表（読み取り専用）

- ID\_LOGIC - ロジック表（読み取り／書き込み）。計算式のインポートをサーバで実行する場合は、読み取り専用のロジック表を選択し、マルチユーザの競合問題を避けるようにしてください。

Microsoft Visual Basic では、文字列のアドレスを確実に lpLogicImport 構造体に渡すことはできません。EntLogicImportVB() を使用すると、インポートファイルのパスとファイル名を、lpLogicImport.lpFileList の代わりに関数の引数として渡すことができます。

**注：** また、Len の代わりに VB の LenB 関数を使用して、LOGICIMPORTSTRUCT 引数内の dwSize フィールドを設定してください。

次の形式を使用します。

```
Declare Function EntLogicImportVB Lib "heaccess.dll" Alias "_EntLogicImportVB@16"
(ByVal hSelect As Long, lpLogicImport As LOGICIMPORTSTRUCT, lpApiStruct As
apiStruct, ByVal szFileList As String) As Integer
```

変数	説明
hSelect	選択された表のハンドル
lpLogicImport	計算式インポート用の引数を含んでいる構造体
lpLogicImport.dwsize	構造体のサイズ
lpLogicImport.hProcess	[タスクステータス] ウィンドウプロセスのハンドル。サーバで実行する場合はプロセスハンドルを返し、ローカルで実行する場合は NULL を返します。
lpLogicImport.delimiter	フィールドの区切り文字で、通常はカンマ (,)
lpLogicImport.bReplace	計算式を交換する場合は TRUE
lpLogicImport.sigMethod	現在のロジックの記号
lpLogicImport.iLogicType	LOGIC_CHART、LOGIC_CONSOL または LOGIC_TRANS のいずれかの種類
lpLogicImport.uMethod	ロジックをインポートする場合は TRUE
lpLogicImport.uRule	ルールをインポートする場合は TRUE
lpLogicImport.uCustomFunc	カスタム関数をインポートする場合は TRUE
lpLogicImport.uLogicComp	計算式をコンパイルする場合は TRUE
lpLogicImport.bNoHeader	ヘッダーなし
lpLogicImport.wlmporFlag	計算式をローカルでインポートするか、サーバでインポートするかを示します。サーバで処理する場合は LOGIC_REMOTE に設定します。
lpLogicImport.wExtendFlag	将来使用するために予約済み
lpLogicImport.fnCallback	ステータス情報用のコールバック関数
lpLogicImport.lParam	コールバック関数の引数

変数	説明
lpLogicImport.iFileListCnt	ファイル一覧表 lpLogicImport.lpFileList の要素の数
szFileList	選択されているファイル名の配列のアドレス（各 260 バイトで、NULL で終了）
lpApiStruct	0. API 互換性のための構造体

戻りコード：

コード	説明
0	成功
NONE	エラー発生
LOGIC_COMPERR (1)	計算式のインポート中にエラーが発生したロジックの確認

例：

```
Private Sub ImportLogic(hSelect As Long, szFile As String)
Dim logicImport As LOGICIMPORTSTRUCT
Dim apiStruct As apiStruct
Dim hApp&
Dim wRet%
logicImport.dwSize = LenB(logicImport)
logicImport.uMethod = 1
logicImport.uRule = 1
logicImport.uCustomFunc = 1
logicImport.delimiter = Asc(",")
logicImport.hProcess = 0
logicImport.iFileListCnt = 1
logicImport.wImportFlag = 0
logicImport.sigMethod = 0 ' 現在のデフォルト
logicImport.bReplace = 0
logicImport.fnCallBack = FuncPtrToLong(AddressOf MyCallBack)
logicImport.lParam = 0
hApp& = EntGetHappFromSelect(hSelect&)
Call EntInitApiStruct(hApp&, apiStruct)
wRet% = EntSelectAdd(HYP_ID_SECURITY or HYP_ID_ASSOC, HYP_NONE,
    hSelect&, APILOCK_READONLY, 0, 0, 0, 0)
wRet% = EntSelectAdd(HYP_ID_LOGIC, HYP_NONE, hSelect&,
    APILOCK_READWRITE, 0, 0, 0, 0)
wRet% = EntLogicImportVB(hSelect&, logicImport, apiStruct,
    szFile$)
End Sub

'AddressOf()の結果を（fnPtrとして）この関数に渡します。
'この関数はその結果をlongに変換するので、
'As Anyではなくlongとして宣言されている
'フィールドまたは引数に割り当てることができます。
Public Function FuncPtrToLong(ByVal fnPtr As Long) As Long
FuncPtrToLong = fnPtr
End Function

Public Function MyCallBack(ByVal addressOfFileNameCString As Long, ByVal
addressOfMethodCString As Long, ByVal lTotal As Long, By Val lCurrent As
Long, ByVal lParam As Long) As Integer
```

```

Dim wRet%, wLen%
Dim szFile$, szMethod$
Dim szMsg$
'Cスタイルの文字列をVisual Basicの文字列にコピーする
wLen% = EntVBGetCStrLen(addressOfFileNameCString)
szFile$ = Space(wLen%)
wRet% = EntVBCopyStr(szFile$, addressOfFileNameCString, wLen%)
wLen% = EntVBGetCStrLen(addressOfMethodCString)
szMethod$ = Space(wLen%)
wRet% = EntVBCopyStr(szMethod$, addressOfMethodCString, wLen%)
szMsg$ = "Now Loading..." & szMethod$ & " From " & szFile$
List1.AddItem (szMsg$)
MyCallBack = 0      '0を返す
End Function

```

C 言語では次の形式を使用します。

```

short WINAPI EntLogicImportVB(HSELECT hSelect,LPLOGICIMPORT lpLogicImport,
LPAPISTRUCT lpApiStruct, LPCSTR szFileList)

```

## EntMessage( ) - メッセージボックスの表示とエラーログへのメッセージの書き込み

EntMessage( )は、メッセージボックスを表示し、メッセージを Hyperion Enterprise エラーログに書き込みます。この関数は、文字列表からメッセージを読み取り、メッセージ内のプレースホルダを指定された文字列に置き換えて、メッセージを書式設定します。この関数は、MB\_OK および指定された標準の Hyperion Enterprise の見出しを使用して MessageBox() を呼び出すことでメッセージを表示します。hApp が 0 でない場合は、メッセージが Hyperion Enterprise のエラーログに書き込まれます。メッセージボックスは、Hyperion Enterprise アプリケーション（hApp が 0 以外の場合）がメッセージを非表示にするモード（サーバ上など）で実行されているか、呼び出し元によってメッセージボックスが非表示になるように指定されている場合には、表示されません。

**注：** Visual Basic のプログラマは、EntMessageVB()を使用する必要があります。  
Visual Basic では C スタイル文字列のアドレスの配列を渡すことができないためです。

次の形式を使用します。

```

short WINAPI EntMessage(HAPP hApp, short wMsgId, HINSTANCE hModule, short
wCaptionCode, short wArgCount, LPCSTR arrayOfStrings)

```

変数	説明
hApp	アプリケーションハンドル
wMsgId	メッセージの文字列 ID（hModule 内）
hModule	文字列 wMsgId を含んでいるモジュール（resource.DLL）のハンドル

変数	説明
wCaptionCode	メッセージボックスに使用する見出しとメッセージボックスを表示するかどうかを示すコード <ul style="list-style-type: none"> <li>● HACC_NODISPLAY (HEMSG_NODISPLAY) - メッセージボックスを表示しない</li> <li>● HACC_WARNING (HEMSG_WARNING) - 標準の Hyperion Enterprise 警告</li> <li>● HACC_ERROR (HEMSG_ERROR) - エラー</li> <li>● HACC_FATAL (HEMSG_FATAL) - 致命的</li> </ul> 詳しくは、TOOLKIT.H ファイルを参照してください。
wArgCount	arrayOfStrings[ ]内の要素の数
arrayOfStrings[ ]	メッセージ (wMsgID 文字列) 内のプレースホルダ (%1、%2 など) を置換する文字列の配列

戻りコード：

0 または MessageBox( )から返された値 (0 または IDOK) を返します。

## EntMessageVB( ) - メッセージボックスの表示とエラーログへのメッセージの書き込み

EntMessageVB( )は、メッセージボックスを表示し、メッセージを Hyperion Enterprise エラーログに書き込みます。この関数は、文字列表からメッセージを読み取り、指定された最高 3 つの文字列でメッセージ内の最初の 3 つのプレースホルダを置き換えて、メッセージを書式設定します。この関数は、MB\_OK および指定された標準の Hyperion Enterprise の見出しを使用して MessageBox( )を呼び出すことでメッセージを表示します。hApp が 0 でない場合は、メッセージが Hyperion Enterprise のエラーログに書き込まれます。メッセージボックスは、Hyperion Enterprise アプリケーション (hApp が 0 以外の場合) がメッセージを非表示にするモード (サーバ上など) で実行されているか、呼び出し元によってメッセージボックスが非表示になるように指定されている場合には、表示されません。次の形式を使用します。

```
Declare Function EntMessageVB Lib "heaccess.dll" Alias "_EntMessageVB@28" (ByVal hApp as Long, ByVal wMsgId As Integer, ByVal hModule As Long, ByVal wCaptionCode As Integer, ByVal sz1 As String, ByVal sz2 As String, ByVal sz3 As String) As Integer
```

変数	説明
hApp	アプリケーションハンドル
wMsgId	メッセージの文字列 ID (hModule 内)
hModule	文字列 wMsgID を含んでいるモジュール (リソース DLL) のハンドル
wCaptionCode	メッセージボックスに使用する見出しとメッセージボックスを表示するかどうかを示すコード <ul style="list-style-type: none"> <li>● HACC_NODISPLAY - メッセージボックスを表示しない</li> <li>● HACC_WARNING - 警告</li> </ul>

## 変数 説明

- HACC\_ERROR - エラー
- HACC\_FATAL - 致命的

詳しくは、`TOOLKIT.H` ファイルを参照してください。

sz1、sz2、sz3 メッセージ (wMsgID 文字列) 内の最初の 3 つのプレースホルダ (%1、%2、%3) を置き換える文字列。必要ない場合は NULL。

戻りコード：

コード	説明
IDOK	成功
0	エラーまたは MessageBox() が呼び出されていない

例：

```
hModule& = GetModuleHandle("MyStrings.dll")
'Win32関数
'IDS_MY_STRINGはMyStrings.dll内の"Error in function %1."の文字列IDです。
'プレースホルダは1つしかないので、他の2つにはNULLを渡します。
wRet% = EntMessageVB(hApp&, IDS_MY_STRING, hModule&, HACC_ERROR,
    _ "MyTestFunction", vbNullString, vbNullString)
```

C 言語では次の形式を使用します。

```
short WINAPI EntMessageVB(HAPP hApp, short wMsgId, HINSTANCE hModule,
short wCaptionCode, LPCSTR sz1, LPCSTR sz2, LPCSTR sz3)
```

## EntMsgLogTaskEndTime( ) - エラーログへの終了時刻とタスクの書き込み

この関数は、終了時刻とタスクをログファイルに書き込み、現在のタスク情報をアプリケーションレポートのユーザから削除します。POV の設定方法例については、[278 ページの「EntUpdateDefault\( \) - デフォルトの更新」](#)を参照してください。

次の形式を使用します。

```
Declare Function EntMsgLogTaskEndTime Lib "heaccess.dll" Alias
    "_EntMsgLogTaskEndTimeVB@12" (ByVal hApp As Long, ByVal szTask As String,
    ByVal bIncludePOV As Long) As Integer
```

## 変数 説明

hApp アプリケーションハンドル

szTask タスクの簡潔な説明

bIncludePOV 現在の POV をタスクに追加するかどうかを判断する場合は TRUE、それ以外の場合は FALSE

この引数は、POV の次元であるデータ種別、期間、組織およびエンティティのみを追加します。

戻りコード：

コード	説明
0	成功
NONE	エラー発生

例：

```
wRet% = EntMsgLogTaskStartTime(hApp&, "Custom Task", 1)
.
.
.
wRet% = EntMsgLogTaskEndTime(hApp&, "Custom Task", 1)
```

C 言語では次の形式を使用します。

```
short WINAPI EntMsgLogTaskEndTime(HAPP hApp, LPSTR szTask, BOOL
bIncludePOV)
```

## EntMsgLogTaskStartTime( ) - エラーログへの開始時刻とタスクの書き込み

この関数は、開始時刻とタスクをログファイルおよびアプリケーションレポートのユーザに書き込みます。POV の設定方法例については、[278 ページの「EntUpdateDefault\( \) - デフォルトの更新」](#)を参照してください。

次の形式を使用します。

```
Declare Function EntMsgLogTaskStartTime Lib "heaccess.dll" Alias
"_EntMsgLogTaskStartTimeVB@12" (ByVal hApp As Long, ByVal szTask As String, ByVal
bIncludePOV As Long) As Integer
```

変数	説明
----	----

hApp	アプリケーションハンドル
------	--------------

szTask	タスクの簡潔な説明
--------	-----------

bIncludePOV	現在の POV をタスクに追加するかどうかを判断する場合は TRUE、それ以外の場合は FALSE
-------------	---

この引数は、POV の次元であるデータ種別、期間、組織およびエンティティのみを追加します。

戻りコード：

コード	説明
MessageBox( )が呼び出された場合はその値	成功
NONE または 0	エラー発生

例：

```
wRet% = EntMsgLogTaskStartTime(hApp&, "Custom Task", 1)
.
.
.
wRet% = EntMsgLogTaskEndTime(hApp&, "Custom Task", 1)
```

C 言語では次の形式を使用します。

```
short WINAPI EntMsgLogTaskEndTime(HAPP hApp, LPSTR szTask, BOOL
bIncludePOV)
```

## EntMsgLogTimeVB( ) - ログファイルへの時刻とイベントの書き込み

EntMsgLogTime( )は、ログファイルに時刻とイベントを書き込みます。この関数は、文字列表からメッセージを読み取り、メッセージ内のプレースホルダを指定された文字列に置き換えて、メッセージを書式設定します。この関数は、メッセージ内の次のプレースホルダを現在の日時に置き換え、書式設定されたメッセージを Hyperion Enterprise のエラーログに書き込みます。

**注：** Visual Basic のプログラマは、EntMsgLogTimeVB( )を代わりに使用する必要があります。Visual Basic では C スタイル文字列のアドレスの配列を渡すことができないためです。

次の形式を使用します。

```
short WINAPI EntMsgLogTime(HAPP hApp, short wMsgId, HINSTANCE hModule,
short wArgCount, LPCSTR arrayOfStrings)
```

変数	説明
hApp	アプリケーションハンドル
wMsgId	メッセージの文字列 ID (hModule 内)
hModule	文字列 wMsgId を含んでいるモジュールのハンドル
arrayOfStrings[]	メッセージ (wMsgId 文字列) 内のプレースホルダ (%1、%2 など) を置換する文字列の配列
wArgCount	arrayOfStrings[] 内の要素の数

戻りコード：

0 を返します。

## EntMsgLogTimeVB( ) - ログファイルへの時刻とイベントの書き込み

EntMsgLogTimeVB( )は、ログファイルに時刻とイベントを書き込みます。この関数は、文字列表からメッセージを読み取り、指定された最高 3 つの文字列でメッ

セージ内の最初の 3 つのプレースホルダを置き換えて、メッセージを書式設定します。この関数は、メッセージ内の次のプレースホルダを現在の日時に置き換え、書式設定されたメッセージをHyperion Enterprise のエラーログに書き込みます。

次の形式を使用します。

```
Declare Function EntMsgLogTimeVB Lib "heaccess.dll" Alias "_EntMsgLogTimeVB@24"  
(ByVal hApp As Long, ByVal wParam As Integer, ByVal hModule As Long, ByVal sz1 As  
String, ByVal sz2 As String, ByVal sz3 As String) As Integer
```

変数	説明
hApp	アプリケーションハンドル
wMsgId	メッセージの文字列 ID (hModule 内)
hModule	文字列 wParam を含んでいるモジュール (リソース DLL) のハンドル
sz1、sz2、sz3	メッセージ (wParam 文字列) 内の最初の 3 つのプレースホルダ (%1、%2、%3) を置き換える文字列。必要ない場合は NULL。

戻りコード：

常に 0 を返します。

例：

```
hModule& = GetModuleHandle("MyStrings.dll") 'Win32関数  
'IDS_MY_STRINGはMyStrings.dll内の"Starting function %1 at %2."の文字列IDで  
す。  
'最後のプレースホルダ%2は時刻と日付用です。  
wRet% = EntMsgLogTimeVB(hApp&, IDS_MY_STRING, hModule&,  
"MyTestFunction", _vbNullString, vbNullString)
```

C 言語では次の形式を使用します。

```
short WINAPI EntMsgLogTimeVB(HAPP hApp, short wParam, HINSTANCE hModule,  
LPCSTR sz1, LPCSTR sz2, LPCSTR sz3)
```

## EntOpenApplication( ) - アプリケーションを開く

この関数は、アプリケーションを開きますが、表は選択しません。この関数を呼び出した後に、表を選択する必要があります。アプリケーションの使用が終わったら、アプリケーションの終了前に EntCloseApplication( ) を呼び出してアプリケーションを終了する必要があります。

次の形式を使用します。

```
Declare Function EntOpenApplication Lib "heaccess.dll" Alias  
"_EntOpenApplication@16" (ByVal szAppId As String, ByVal szUserId As String, ByVal  
szPassword As String, wRet As Integer) As Long
```

変数	説明
szAppId	アプリケーション ID

変数	説明
----	----

szUserId	ユーザ ID
----------	--------

szPassword	ユーザのパスワード
------------	-----------

wRet	エラーコードを返すためのバッファ。この変数は、実行に成功した場合は 0（ゼロ）、エラーが発生した場合は NONE を返します。
------	---

戻りコード：

コード	説明
アプリケーションハンドル (hApp)	成功
NONE または 0	エラー発生

例：

```
hApp& = EntOpenApplication("TAX", "MARY", "secret", ret%)  
If ret% = 0 Then MsgBox("Successful login to TAX application")
```

C 言語では次の形式を使用します。

**HAPP WINAPI EntOpenApplication(LPSTR szAppId, LPSTR szUserId, LPSTR szPassword, short far \* pwRet)**

## EntOpenServerApplication( ) - サーバアプリケーションを開く

この関数は EntOpenApplication( ) に似ていますが、エラーメッセージを非表示にするための内部フラグを設定します。この関数はサーバ上で使用するためのものです。

次の形式を使用します。

```
Declare Function EntOpenServerApplication Lib "heaccess.dll" Alias  
"EntOpenServerApplication@16" (ByVal szAppId As String, ByVal szUserId As String,  
ByVal szPassWord As String, wRet As Integer) As Long
```

変数	説明
----	----

szAppId	アプリケーション ID
---------	-------------

szUserId	ユーザ ID
----------	--------

szPassWord	ユーザのパスワード
------------	-----------

wRet	エラーコードを返すためのバッファ。この変数は、実行に成功した場合は 0（ゼロ）、エラーが発生した場合は NONE を返します。
------	---

戻りコード：

コード	説明
アプリケーションハンドル (hApp)	成功
NONE または 0	エラー発生

例：

```
hApp% = EntOpenServerApplication("TAX", "MARY", "secret", ret%)
```

C 言語では次の形式を使用します。

```
HAPP WINAPI EntOpenServerApplication(LPSTR szAppId, LPSTR szUserId, LPSTR szPassWord, short far * pwRet)
```

## EntPgeExtract( ) - ページ書式および設定情報の抽出

この関数は、ページ書式および設定情報を抽出します。これらは [タスク] メニューの [ページ書式と設定の抽出] で設定できます。

**注：** EntPgeExtract() と EntPgeLoad() は、バイナリファイルの読み込みと書き込みを行います。バイナリファイルにはユーザはアクセスできません。

次の形式を使用します。

```
Declare Function EntPgeExtract Lib "heaccess.dll" Alias "_EntPgeExtract@12" (ByVal hApp As Long, ByVal lpszFileName As String, ByVal wFlag As Integer) As Integer
```

変数	説明
hApp	アプリケーションハンドル
lpszFileName	ページ書式および設定読み込みファイル
wFlag	フラグ（現在は無視されています）

戻りコード：

コード	説明
0	成功
NONE	エラー発生

例：

```
wRet% = EntPgeExtract(hApp%, szFile$, 0)
```

C 言語では次の形式を使用します。

```
short WINAPI API_PgeExtract(HAPP hApp, LPCSTR szFileName, WORD wFlag)
```

## EntPgeLoad( ) - ページ書式および設定情報の読み込み

この関数は、ページ書式および設定情報を読み込みます。これらは [タスク] メニューの [ページ書式と設定の抽出] で設定できます。この関数を使用すると、ユーザは自分のアプリケーションを完全に再構築できます。

読み込みは、実際にはデフォルト以外の書式および設定情報を結合するルーチンになります。デフォルト以外の値を上書きしてデフォルトの書式または設定が再度読み込まれることはありません。

**注：** EntPgeExtract( ) と EntPgeLoad( ) は、バイナリファイルの読み込みと書き込みを行います。バイナリファイルにはユーザはアクセスできません。

次の形式を使用します。

```
Declare Function EntPgeLoad Lib "heaccess.dll" Alias "_EntPgeLoad@12" (ByVal hApp As Long, ByVal lpszFileName As String, ByVal wFlag As Integer) As Integer
```

変数	説明
----	----

hApp	アプリケーションハンドル
------	--------------

lpszFileName	ページ書式および設定読み込みファイル
--------------	--------------------

wFlag	フラグ（現在は無視されています）
-------	------------------

戻りコード：

コード	説明
0	成功
NONE	エラー発生

例：

```
wRet% = EntPgeLoad(hApp%, szFile$, 0)
```

C 言語では次の形式を使用します。

```
short WINAPI API_PgeLoad(HAPP hApp, LPCSTR szFile, WORD wFlag)
```

## EntPerAsk( ) - 期間の選択

この関数は、Hyperion Enterprise の [期間の選択] ダイアログボックスを呼び出して、選択された期間を取得します。[キャンセル] を選択した場合、期間は取得されません。

次の形式を使用します。

```
Declare Function EntPerAsk Lib "heaccess.dll" Alias "_EntPerAsk@12" (ByVal hSelect As Long, fWantDate As Integer, ByVal szRetbuf As String) As Integer
```

## 変数 説明

hSelect 選択された表のハンドル

fWantDate 期間を返すための次のフラグのいずれか

- 期間を期間番号として返す場合は 0
- 期間を日付として返す場合（可能な場合）は 1

szRetbuf 期間を返すためのバッファ。バッファは、少なくとも日付に 1 文字を足した長さにする必要があります。

**注：** fWantDate に 1 を指定した場合に利用可能な日付がないと、fWantDate の値は Hyperion Enterprise によってゼロに変更されます。例えば、fWantDate に 1 を指定し、13 番目の会計年度の 13 番目の月を取得しようとする、fWantDate の値はゼロに変更され、szRetbuf の期間番号が返されます。日付（fWantDate = 1 の場合）は、月、日および年をカンマで区切った値として返されます（mm,dd,yy）。例えば、2003 年 3 月 31 日は"3,31,2003"として返されます。

戻りコード：

コード	説明
0	成功
NONE	エラー発生

例：

```
Public Function SelectPeriod(iMethod As Integer) As String
Dim ret As Integer
Dim PerAsk As String * 20
If hSelect <> 0 Then
If iMethod% < 0 Or iMethod% > 1 Then iMethod% = 0
ret% = EntPerAsk(hSelect&, iMethod%, PerAsk$)
If ret% = 0 Then SelectPeriod = CToBStr(PerAsk$)
End If
End Function
```

C 言語では次の形式を使用します。

```
short WINAPI EntPerAsk( HSELECT hSelect, SBOOL FAR * pfWantDate, LPSTR
szRetbuf)
```

## EntQryRptFreq( ) - クエリのレポート期間単位

この関数は、任意の期間単位と表示形式に関連付けられているレポート期間単位の ID（短い名前）または説明を取得します。この関数を呼び出す前に、レポート期間単位（ID\_RPTFREQ）とレポート表示形式（ID\_RPTVIEW）の表を選択する必要があります。

次の形式を使用します。

Declare Function EntQryRptFreq Lib "heaccess.dll" Alias "\_EntQryRptFreq@24" (ByVal hSelect As Long, ByVal sigFreq As Long, ByVal wView As Integer, ByVal pzbuf As String, ByVal wMax As Integer, ByVal szBuf As String ) As Integer

**変数 説明**

- hSelect 選択された表のハンドル
- sigFreq 期間単位（FREQ\_MONTH、FREQ\_QUARTER など）の記号。詳しくは、[20 ページの「期間単位と表示形式」](#)を参照してください。
- wView 表示形式（VIEW\_YTD または VIEW\_PERIODIC）
- pzbuf クエリ属性（NAME または DESC）
- wMax pzBuf の長さ
- szBuf レポート期間単位の ID または説明を返す先のバッファ。szBuf が NAME の場合、バッファのサイズは少なくとも HYP\_SIZERPTFREQ+1（C 言語では SIZERPTFREQ+1）文字にする必要があります。szBuf が DESC の場合、バッファのサイズは少なくとも HYP\_SIZEDESC+1（C 言語では SIZEDESC+1）文字にする必要があります。

戻りコード：

コード	説明
0	成功
NONE	エラー発生

例：

```
szRptFreq$ = Space(HYP_SIZERPTFREQ+1)
ret% = EntQryRptFreq(hSelect&, FREQ_MONTH, VIEW_YTD, NAME,
    Len(szRptFreq$), szRptFreq$)
If ret% = 0 Then szRptFreq$ = CToBStr(szRptFreq$)
```

C 言語では次の形式を使用します。

**short** WINAPI EntQryRptFreq(HSELECT hSelect, SIGNA sigFreq, **short** iView, **short** iAttr, **short** iMax, **void** \* szBuf )

## EntQueryEx( ) - クエリ情報

この関数は、表内のレコードのクエリを行い、要求されたフィールド（属性）をレコードから取得します。Hyperion Enterprise 表で利用できるクエリ属性については、[付録 B「クエリ属性」](#)を参照してください。

Visual Basic プログラマは、文字列値のクエリを行う場合、EntQueryExStr( )関数を使用する必要があります。EntQueryExStr()は、EntQueryEx( )の宣言のクローンを文字列用に調整したもので、TOOLKIT.BAS ファイルで宣言されます。

次の形式を使用します。

Declare Function EntQueryEx Lib "heaccess.dll" Alias "\_EntQueryEx@32" (ByVal hSelect As Long, ByVal wTabId As Integer, ByVal wAttr As Integer, ByVal sigRecd As Long,

ByVal sigKey As Long, ByVal dwLen As Long, pzBuf As Any, apiS As APISTRUCT) As Integer

#### 変数 説明

hSelect 選択された表のハンドル

wTabId クエリ対象のレコードを含む表

wAttr クエリ対象のフィールドの属性

sigRecd クエリ対象のレコードの記号

sigKey NONE、または関連する表のキー。詳しくは、[340 ページの「関連する表」](#)を参照してください。

dwLen pzBuf の長さ

pzBuf 要求した情報が返されるバッファ

apiS NULL、または詳細情報を含んでいる apiStruct 構造体のアドレス

**注：** Microsoft Visual Basic でプログラミング作業を行っている場合は、apiStruct の代わりに NULL を渡す方法について [16 ページの「Visual Basic プログラミング上の注意」](#)を参照してください。

戻りコード：

コード	説明
0	成功
NONE	エラー発生

例：

```
dwLen& = Len(sigFreq&)
ret% = EntQueryEx0(hSelect&, HYP_ID_CATEGORY, HYP_CAT_FREQ,
    sigCat&, HYP_NONE, dwLen&, sigFreq&, 0)
```

**注：** この例の EntQueryEx0( )は、EntQueryEx( )と同じように宣言されますが、apiStruct の代わりに 0 を渡すことが許可されます。詳しくは、[16 ページの「Visual Basic プログラミング上の注意」](#)を参照してください。

C 言語では次の形式を使用します。

```
short WINAPI EntQueryEx(HSELECT hSelect, short wTabId, short wAttr, SIGNA
sigRecd, SIGNA sigKey,longd wLen, void FAR * pzBuf, LPAPISTRUCT pApiS)
```

## EntQueryDefault( ) - デフォルトのクエリ

EntQueryDefault( )は、アプリケーションまたはユーザのデフォルトのクエリを実行します。この関数を呼び出す前に表を選択する必要はありません。Hyperion

Enterprise 表で利用できるクエリ属性については、[付録 B「クエリ属性」](#)を参照してください。

Visual Basic プログラマは、文字列値のクエリを行う場合、EntQueryDefaultStr( )関数を使用する必要があります。EntQueryDefaultStr()は EntQueryDefault()の宣言のクローンを文字列用に調整したもので、ToolKit.bas ファイルで宣言されます。Hyperion Enterprise 表で利用できるクエリ属性については、[付録 B「クエリ属性」](#)を参照してください。

次の形式を使用します。

```
Declare Function EntQueryDefault Lib "heaccess.dll" Alias "_EntQueryDefault@20"  
(ByVal hApp As Long, ByVal wTabId As Integer, ByVal wAttr As Integer, ByVal wLen As  
Integer, pzBuf As Any) As Integer
```

**変数 説明**

- hApp   アプリケーションハンドル
- wTabId   次の表のいずれかの表 ID
- ID\_HAPP
  - ID\_APPDEFAULT
  - ID\_USERDEFAULT
- wAttr   クエリ対象のフィールドの属性
- wLen   pzBuf の長さ
- pzBuf   要求した情報が返されるバッファ

戻りコード：

コード	説明
0	成功
NONE	エラー発生

例：

```
Dim isServerAvail As Byte  
ret% = EntQueryDefault(hApp&, HYP_ID_HAPP, HYP_SERVER_AVAIL,  
Len(isServerAvail), isServerAvail)
```

C 言語では次の形式を使用します。

```
short WINAPI EntQueryDefault(HAPP hApp, short wTabId, short wAttr, short wLen, void  
FAR * pzBuf)
```

## EntRegisterMsgBox( ) - 代替 MessageBox( )ルーチンの登録

この関数は、Windows の MessageBox( )関数を直接呼び出す代わりに使用する、Hyperion Enterprise API 関数用関数を登録します。

次の形式を使用します。

```
Declare Sub EntRegisterMsgBox Lib "heaccess.dll" Alias "_EntRegisterMsgBox@4" (ByVal lpfnMsgBox)
```

ここで、lpfnMsgBox は関数のアドレスまたは NULL です。以前に登録した関数の登録を解除して、API が MessageBox( ) を再び使用できるようにするには、NULL を使用します。

戻りコード：

なし

C 言語では次の形式を使用します。

```
void WINAPI EntRegisterMsgBox(LPFNMSGBOX lpfnMsgBox)
```

## EntRunRollover( ) - 期別替の実行

この関数は、期別替を実行します。EntRunRollover( ) のコールバック関数は、ステータス情報を返します。

次の形式を使用します。

```
Declare Function EntRunRollover Lib "heaccess.dll" Alias "_EntRunRollover@12" (ByVal hSelect As Long, ByVal sigRolloverSet As Long, ByVal lpCallback As Long) As Integer
```

変数	説明
----	----

hSelect	選択された表のハンドル
---------	-------------

sigRolloverSet	期別替レコードの記号
----------------	------------

lpCallback	ユーザ定義のコールバック関数のアドレス。詳しくは、 <a href="#">319 ページの「CALLBACKAPI」</a> を参照してください。
------------	--

戻りコード：

コード	説明
0	成功
NONE	エラー発生

C 言語では次の形式を使用します。

```
short WINAPI EntRunRollover(HSELECT hSelect, SIGNA sigRolloverSet, CALLBACKAPI lpCallback)
```

## EntSave( ) - 表の保存

この関数は、EntUpdate( ) によって行われた Hyperion Enterprise 表への変更を保存します。選択されている表への変更は、すべて同時に保存してください。一度に 1 つの表への変更を保存していて、プログラムによって変更の残りの保存が中断されたか、保存できなかった場合には、各表が非同期になる可能性があります。

次の形式を使用します。

```
Declare Function EntSave Lib "heaccess.dll" Alias "_EntSave@16" (ByVal hSelect As Long,
ByVal wTabId As Integer, ByVal sigKey As Long, apiS As APISTRUCT) As Integer
```

#### 変数 説明

hSelect 選択された表のハンドル

wTabId 保存する表の ID、または hSelect によって選択されている表をすべて保存する場合は HYP\_NONE (C 言語の場合は NONE)

sigKey NONE、または関連する表のキー。詳しくは、[340 ページの「関連する表」](#)を参照してください。

apiS NULL、または詳細情報を含んでいる apiStruct 構造体を指すポインタ

**注：** 内部の表は別の表への変更によって変化する場合がありますので、wTabI 引数を HYP\_NONE (C 言語では NONE) に設定し、選択されているすべての表への変更を保存することをお勧めします。

戻りコード：

コード	説明
0	成功
NONE	エラー発生

例：

```
rc% = EntSave(hSelect&, HYP_NONE, HYP_NONE, 0)
```

C 言語では次の形式を使用します。

```
short WINAPI EntSave(HSELECT hSelect, short wTabId, SIGNA sigKey, LPAPISTRUCT
pApiS)
```

## EntSaveDefault( ) - デフォルトの保存

この関数は、EntUpdateDefault( )関数によって変更されたアプリケーションまたはユーザのデフォルトを保存します。デフォルト設定については、[338 ページの「デフォルト設定の表」](#)を参照してください。

次の形式を使用します。

```
Declare Function EntSaveDefault Lib "heaccess.dll" Alias "_EntSaveDefault@8" (ByVal
hApp As Long, ByVal wTabId As Integer) As Integer
```

#### 変数 説明

hApp アプリケーションハンドル

## 変数 説明

wTabId 次の表のいずれかの表 ID

- ID\_APPDEFAULT
- ID\_USERDEFAULT

戻りコード：

コード	説明
0	成功
NONE	エラー発生

C 言語では次の形式を使用します。

**short** WINAPI EntSaveDefault(HAPP hApp, **short** wTabId)

## EntSecurityExtract( ) - セキュリティの抽出

この関数は、セキュリティの抽出を実行します。この関数は、API が抽出の現在のステータスを送信するために呼び出すコールバック関数を必要とします。

サーバ上でセキュリティを抽出する場合（DBE\_REMOTE ビットフラグが lpSecLoad->wFlags に設定されている場合）は、抽出処理のプロセスハンドルを認識し、プログラムによって起動されるサーバ処理を追跡および監視できるようにすると便利です。lpSecLoad->hProcess に返されるプロセスハンドルは、ローカルクライアントコンピュータで実行されている [タスクステータス] ウィンドウプログラム HComMgr.exe のインスタンスのプロセスハンドルです。詳しくは、[121 ページの「サーバのタスク」](#)を参照してください。

**注：** Visual Basic プログラマは、EntSecurityExtract( )の代わりに EntSecurityExtractVB( )を使用する必要があります。

C 言語では次の形式を使用します。

**short** WINAPI EntSecurityExtract(HSELECT hSelect, LPSECLOADSTRUCT lpSecLoad, CALLBACKSECLOAD fnCallBack, LPARAM lParam)

変数	説明
hSelect	選択された表のハンドル
lpSecLoad	セキュリティ抽出用の引数を含んでいる構造体
lpSecLoad->dwsz	この構造体のサイズ
lpSecLoad->cDelimiter	フィールドの区切り文字で、通常はカンマ (,)
lpSecLoad->szFileName	抽出ファイルのパス

変数	説明
lpSecLoad->bSecClass	セキュリティクラスを抽出する場合は TRUE、それ以外の場合は False
lpSecLoad->bAcesRight	アクセス権を抽出する場合は TRUE、それ以外の場合は False
lpSecLoad->bUserAndGroup	ユーザおよびユーザグループを抽出する場合は TRUE、それ以外の場合は False
lpSecLoad->bTask	セキュリティタスクを抽出する場合は TRUE、それ以外の場合は False
lpSecLoad->wFlags	必要に応じてフラグ DBE_APPEND および DBE_REMOTE
lpSecLoad->phProcess	サーバ上での実行時に返されるプロセスハンドル
fnCallBack	ステータス情報用のコールバック関数。これは DBE_REMOTE フラグが lpSecLoad->wFlags に設定されている場合には使用されません。
lParam	コールバック関数の引数

戻りコード：

コード	説明
LOAD_NOERR (0)	成功
NONE	エラー発生
LOAD_CANCELLED	ユーザによってキャンセルされました。
LOAD_CHECK_ERRORLOG	エラーが発生し、そのエラーがエラーログに示されています。

253 ページの「[EntSecurityExtractVB\( \) - セキュリティの抽出](#)」の例を参照してください。

## EntSecurityExtractVB( ) - セキュリティの抽出

Visual Basic プログラマは、EntSecurityExtract( )の代わりにこの関数を使用してください。この関数は、セキュリティの抽出を実行します。この関数は、API が抽出の現在のステータスを送信するために呼び出すコールバック関数を必要とします。

サーバ上でセキュリティを抽出する場合（DBE\_REMOTE ビットフラグが lpSecData.wFlags に設定されている場合）は、抽出処理のプロセスハンドルを認識し、プログラムによって起動されるサーバ処理を追跡および監視できるようにすると便利です。lpSecData.hProcess に返されるプロセスハンドルは、ローカルクライアントコンピュータで実行されている [タスクステータス] ウィンドウプログラム HComMgr.exe のインスタンスのプロセスハンドルです。詳しくは、[121 ページの「サーバのタスク」](#)を参照してください。

次の形式を使用します。

```
Declare Function EntSecurityExtractVB Lib "heaccess.dll" Alias
"_EntSecurityExtractVB@20" _ (ByVal hSelect As Long, lpSecData As SecLoadStruct,
ByVal lpszFileName As String, ByVal lpfnCallBack As Any, ByVal lParam As Long) As
Integer
```

変数	説明
hSelect	選択された表のハンドル
lpSecData	セキュリティ抽出用の引数を含んでいる構造体
lpSecData.dwsize	この構造体のサイズ
lpSecData.cDelimiter	フィールドの区切り文字で、通常はカンマ (,)
lpSecData.szFileName	抽出ファイルのパス
lpSecData.bSecClass	セキュリティクラスを抽出する場合は TRUE、それ以外の場合は False
lpSecData.bAcesRight	アクセス権を抽出する場合は TRUE、それ以外の場合は False
lpSecData.bUserAndGroup	ユーザおよびユーザグループを抽出する場合は TRUE、それ以外の場合は False
lpSecData.bTask	セキュリティタスクを抽出する場合は TRUE、それ以外の場合は False
lpSecData.wFlags	必要に応じてフラグ DBE_APPEND および DBE_REMOTE
lpSecData.phProcess	サーバ上での実行時に返されるプロセスハンドル
szFileName	抽出ファイルのパスとファイル名
lpfnCallBack	ステータス情報用のコールバック関数。これは DBE_REMOTE フラグが lpSecData.wFlags に設定されている場合には使用されません。
lParam	コールバック関数の引数

例：

**注：** この例の EntUnSelect0( )は、EntUnSelect( )と同じように宣言されますが、apiStruct の代わりに 0 を渡すことが許可されます。詳しくは、[16 ページの「Visual Basic プログラミング上の注意」](#)を参照してください。

```
Private Sub SecurityExtract(hSelect As Long, szFile As String)
Dim s As SecLoadStruct
Dim wRet%
wRet% = EntSelectAdd(HYP_ID_SECURITY Or HYP_ID_ASSOC, HYP_NONE,
    hSelect&, APILOCK_READONLY, 0, 0, 0, 0)
'抽出する項目を選択
If chkClass.Value Then s.bSecClass = True
If chkRights.Value Then s.bAcesRight = True
If chkUsr.Value Then s.bUserAndGroup = True
If chkTask.Value Then s.bTask = True
s.dwSize = LenB(s)
s.cDelimiter = Asc(",")
s.wFlags = 0
s.szFileName = 0 '代わりにszFile$を引数として関数に渡す
wRet% = EntSecurityExtractVB(hSelect&, s, szFile$, AddressOf
    MyCallBack, 0)
hSelect& = EntUnSelect0(hSelect&, HYP_ID_SECURITY Or
    HYP_ID_ASSOC, HYP_NONE, 0, wRet%, 0)
End Sub
```

```

Public Function MyCallBack(ByVal addressOfFileNameCString As Long, ByVal
addressOfTaskCString As Long, ByVal lPos As Long, ByVal lStart As Long,
ByVal lEnd As Long, ByVal lParam As Long) As Integer
Dim wRet%, wLen%
Dim szFile$, szTaskName$
Dim szMsg$
'Cスタイルの文字列をVisual Basicの文字列にコピーする
wLen% = EntVBGetCStrLen(addressOfFileNameCString)
szFile$ = Space(wLen%)
wRet% = EntVBCopyStr(szFile$, addressOfFileNameCString, wLen%)
wLen% = EntVBGetCStrLen(addressOfTaskCString)
szTaskName$ = Space(wLen%)
wRet% = EntVBCopyStr(szTaskName$, addressOfTaskCString, wLen%)
szMsg$ = "Now Extracting " & szTaskName$ & " From " & szFile$
List1.AddItem (szMsg$)
MyCallBack = 0      '0を返す
End Function

```

C 言語では次の形式を使用します。

```

short WINAPI EntSecurityExtractVB(HSELECT hSelect, LPSECLOADSTRUCT
lpSecData, LPCSTR lpszExtFileName,CALLBACKSECLOAD fnCallBack, LPARAM
lParam)

```

## EntSecurityLoad( ) - セキュリティの読み込み

この関数は、セキュリティの読み込みを実行します。この関数は、API が読み込みの現在のステータスを送信するために呼び出すコールバック関数を必要とします。

サーバ上でセキュリティを読み込む場合（DB\_REMOTE ビットフラグが lpSecLoad->wFlags に設定されている場合）は、読み込み処理のプロセスハンドルを認識し、プログラムによって起動されるサーバ処理を追跡および監視できるようにすると便利です。lpSecLoad->hProcess に返されるプロセスハンドルは、ローカルクライアントコンピュータで実行されている [タスクステータス] ウィンドウプログラム HComMgr.exe のインスタンスのプロセスハンドルです。詳しくは、[121 ページ](#)の「サーバのタスク」を参照してください。

**注：** Visual Basic プログラマは、EntSecurityLoad( )の代わりに EntSecurityLoadVB( )を使用する必要があります。

C 言語では次の形式を使用します。

```

short WINAPI EntSecurityLoad(HSELECT hSelect, LPSECLOADSTRUCT lpSecLoad,
CALLBACKSECLOAD fnCallBack, LPARAM lParam)

```

変数	説明
hSelect	選択された表のハンドル
lpSecLoad	セキュリティ読み込み用の引数を含んでいる構造体

変数	説明
lpSecLoad->dsize	この構造体のサイズ
lpSecLoad->bSecClass	読み込むセキュリティクラス
lpSecLoad->bAcesRight	読み込むアクセス権
lpSecLoad->bUserAndGroup	読み込むユーザとグループ
lpSecLoad->bTask	読み込むセキュリティタスク
lpSecLoad->szFileName	読み込みファイルのパス
lpSecLoad->cDelimiter	フィールドの区切り文字で、通常はカンマ (,)
lpSecLoad->cPad	未使用
lpSecLoad->wFlags	0 または DB_REMOTE
lpSecLoad->phProcess	サーバ上での実行時に返されるプロセスハンドル
fnCallBack	ステータス情報用のコールバック関数。これは DBE_REMOTE フラグが lpSecLoad->wFlags に設定されている場合には使用されません。
lParam	コールバック関数の引数

戻りコード：

コード	説明
LOAD_NOERR (0)	成功
NONE	エラー発生
LOAD_CANCELLED	ユーザによってキャンセルされました。
LOAD_CHECK_ERRORLOG	エラーが発生し、そのエラーがエラーログに示されています。

256 ページの「EntSecurityLoadVB() - セキュリティの読み込み」の例を参照してください。

## EntSecurityLoadVB() - セキュリティの読み込み

Visual Basic プログラマは、EntSecurityLoad() の代わりにこの関数を使用してください。この関数は、セキュリティの読み込みを実行します。この関数は、API が読み込みの現在のステータスを送信するために呼び出すコールバック関数を必要とします。

サーバ上でセキュリティを読み込む場合（DB\_REMOTE ビットフラグが lpSecData.wFlags に設定されている場合）は、読み込み処理のプロセスハンドルを認識し、プログラムによって起動されるサーバ処理を追跡および監視できるようにすると便利です。lpSecData.hProcess に返されるプロセスハンドルは、ローカルクライアントコンピュータで実行されている [タスクステータス] ウィンドウプ

プログラム HCOMMGR.exe のインスタンスのプロセスハンドルです。詳しくは、[121 ページの「サーバのタスク」](#)を参照してください。

**注：** Visual Basic プログラマは、EntSecurityLoad()の代わりに EntSecurityLoadVB() を使用する必要があります。

次の形式を使用します。

**Declare Function EntSecurityLoadVB Lib "heaccess.dll" Alias "\_EntSecurityLoadVB@20" (ByVal hSelect As Long, lpSecData As SecLoadStruct, ByVal lpzFileName As String, ByVal lpfnCallBack As Any, ByVal lParam As Long) As Integer**

変数	説明
hSelect	選択された表のハンドル
lpSecData	セキュリティ読み込み用の引数を含んでいる構造体
lpSecData.dwsz	この構造体のサイズ
lpSecData.cDelimiter	フィールドの区切り文字で、通常はカンマ (,)
lpSecData.szFileName	読み込みファイルのパス
lpSecData.bSecClass	セキュリティクラスを読み込む場合は TRUE、それ以外の場合は False
lpSecData.bAcesRight	アクセス権を読み込む場合は TRUE、それ以外の場合は False
lpSecData.bUserAndGroup	ユーザおよびユーザグループを読み込む場合は TRUE、それ以外の場合は False
lpSecData.bTask	セキュリティタスクを読み込む場合は TRUE、それ以外の場合は False
lpSecData.wFlags	0 または DB_REMOTE
lpSecData.phProcess	サーバ上での実行時に返されるプロセスハンドル
szFileName	読み込むファイルのパスとファイル名
fnCallBack	ステータス情報用のコールバック関数。これは DB_REMOTE フラグが lpSecData.wFlags に設定されている場合には使用されません。
lParam	コールバック関数の引数

戻りコード：

コード	説明
LOAD_NOERR (0)	成功
NONE	エラー発生
LOAD_CANCELLED	ユーザによってキャンセルされました。
LOAD_CHECK_ERRORLOG	エラーが発生し、そのエラーがエラーログに示されています。

例：

```

Private Sub SecurityLoad(hSelect As Long, szFile As String)
Dim s As SecLoadStruct
Dim flag%, wRet%
wRet% = EntSelectAdd(HYP_ID_SECURITY Or HYP_ID_ASSOC, HYP_NONE,
    hSelect&, APILOCK_READWRITE, 0, 0, 0, 0)
'読み込む項目を選択
If chkClass.Value Then s.bSecClass = True
If chkRights.Value Then s.bAcesRight = True
If chkUsr.Value Then s.bUserAndGroup = True
If chkTask.Value Then s.bTask = True
s.dwSize = LenB(s)
s.cDelimiter = Asc("!")
s.wFlags = 0 'ローカルでの実行
s.szFileName = 0 '代わりにszFile$を引数として関数に渡す
wRet% = EntSecurityLoadVB(hSelect&, s, szFile$, AddressOf
    MyCallBack, 0)
hSelect& = EntUnSelect0(hSelect&, HYP_ID_SECURITY Or
    HYP_ID_ASSOC, HYP_NONE, 0, wRet%, 0)
End Sub

Public Function MyCallBack(ByVal addressOfFileNameCString As Long, ByVal
    addressOfTaskCString As Long, ByVal lPos As Long, ByVal lStart As Long,
    ByVal lEnd As Long, ByVal lParam As Long) As Integer
Dim wRet%, wLen%
Dim szFile$, szTask$
Dim szMsg$
'Cスタイルの文字列をVisual Basicの文字列にコピーする
wLen% = EntVBGetCStrLen(addressOfFileNameCString)
szFile$ = Space(wLen%)
wRet% = EntVBCopyStr(szFile$, addressOfFileNameCString, wLen%)
wLen% = EntVBGetCStrLen(addressOfTaskCString)
szTask$ = Space(wLen%)
wRet% = EntVBCopyStr(szTask$, addressOfTaskCString, wLen%)
szMsg$ = "Now Loading " & szTask$ & " From " & szFile$
List1.AddItem (szMsg$)
MyCallBack = 0 '0を返す
End Function

```

C 言語では次の形式を使用します。

```

short WINAPI EntSecurityLoadVB(HSELECT hSelect, LPSECLOADSTRUCT lpSecData,
LPCSTR lpszFileName, CALLBACKSECLOAD fnCallBack, LPARAM lParam)

```

## EntSelect( ) - 使用する表の選択

この関数は、表の選択に使用します。表を他の関数からアクセスできるようにするには、その前に表を選択する必要があります。EntSelect( )は、選択された表のハンドルを返します。表 ID と ID\_ASSOC 定数を OR 演算で結合すると、関連付けられている表をすべて同時に選択できます。追加の表を選択するには EntSelectAdd( )を使用します。表の使用が終わったら、EntUnSelect( )関数を呼び出して表の選択を必ず解除してください。

関連付けられている表は個別に選択するのではなく、ID\_ASSOC 定数を使用します。詳しくは、[339 ページの「関連付けられている表」](#)を参照してください。

**注：** sigKey 引数を必要とする表に EntSelect( )を使用することはできません。これは、sigKey を含む表をまず選択してから、EntSelectAdd( )を使用して必要な表を選択する必要があるためです。詳しくは、[260 ページの「EntSelectAdd\(\) - 使用する追加の表の選択」](#)を参照してください。

次の形式を使用します。

**Declare Function EntSelect Lib "heaccess.dll" Alias "\_EntSelect@36" (ByVal hApp As Long, ByVal wTabId As Integer, ByVal sigKey As Long, ByVal wAttr As Integer, wLockMode As Integer, ByVal lpPtar As Long, ByVal lpPtac As Long, ByVal SelCallback As Long, ByVal lParam As Long) As Long**

変数	説明
hApp	アプリケーションハンドル
wTabId	選択する表の ID
sigKe	NONE
wAttr	必要なアクセス権。読み取り専用アクセスで表を開くには、APILOCK_READONLY を使用します。
wLockMode	付与されたアクセス権を返すためのバッファ。表のアクセス権が付与された場合は APILOCK_READONLY が返され、拒否された場合は NONE が返されます。
lpPtar	NULL（未実装）
lpPtac	NULL（内部での API の使用のために予約済み）
SelCallback	NULL またはコールバック関数。詳しくは、 <a href="#">112 ページの「選択のコールバック」</a> を参照してください。
lParam	ユーザ定義のコールバック関数に返されるパラメータ

戻りコード：

コード	説明
選択された表のハンドル	成功
NULL	エラー発生

例：

```
hSelect& = EntSelect(hApp&, HYP_ID_CATEGORY OR HYP_ID_ASSOC,
    HYP_NONE, APILOCK_READONLY, ret%, 0, 0, 0, 0)
If ret% <> APILOCK_READONLY Then MsgBox("Error selecting table")
```

C 言語では次の形式を使用します。

**HSELECT WINAPI EntSelect(HAPP hApp, short wTabId, SIGNA sigKey, short wAttr, short FAR \* lpwLockMode, void FAR \* lpPtar, void FAR \* lpPtac, CALLBACKSEL SelCallback, LPARAM lParam)**

## EntSelectAdd( ) - 使用する追加の表の選択

この関数は、表を選択し、EntSelect( )からの既存の表選択ハンドルに追加します。関連付けられた表を個別に選択するのではなく、OR 演算と ID\_ASSOC 定数を使用して、表および関連付けられた表を同時に選択してください。詳しくは、[339 ページの「関連付けられている表」](#)を参照してください。

**注：** 表を使用した後は、EntUnSelect( )を呼び出して表の選択を解除してください。

データ種別が sigKey 引数として渡される表は、1 つのデータ種別に適用されます。複数のデータ種別用に表を選択する場合は、EntSelectAdd( )を呼び出してデータ種別ごとに表を選択してください。

**注：** ID\_JOURNAL\_DETAIL、ID\_JOURNAL\_HISTORY\_DETAIL、ID\_JOURNAL\_TEMPLATE\_DETAIL または ID\_JOURNAL\_PERIOD\_INFO を選択する必要がある場合は、EntSelectAdd( )の代わりに EntSelectTableAdd( )を使用します。apiStruct に structure 引数を渡す必要があるためです。

次の形式を使用します。

```
Declare Function EntSelectAdd Lib "heaccess.dll" Alias "_EntSelectAdd@32" (ByVal wTabId As Integer, ByVal sigKey As Long, ByVal hSelect As Long, ByVal wAttr As Integer, ByVal lpPtar As Long, ByVal lpPtac As Long, ByVal SelCallback As Long, lParam As Long) As Integer
```

変数	説明
wTabId	選択する表の ID
sigKey	NONE、または関連する表のキー。関連する表が既に選択されている必要があります。
hSelect	EntSelect( )からの現在選択されている表のハンドル
wAttr	必要なアクセス権。読み取り専用アクセスで表を開くには、APILOCK_READONLY を使用します。
lpPtar	NULL（未実装）
lpPtac	NULL（内部での API の使用のために予約済み）
SelCallback	NULL、またはユーザ定義のコールバック関数へのポインタ。詳しくは、 <a href="#">112 ページの「選択のコールバック」</a> を参照してください。
lParam	ユーザ定義のコールバック関数に返されるパラメータ

戻りコード：

コード	説明
付与されたアクセス権	成功
NONE	エラー発生

例：

```
ret% = EntSelectAdd(HYP_ID_ORGANIZATION OR HYP_ID_ASSOC,
    sigCat&, hSelect&, APILOCK_READONLY, 0,0,0,0)
If ret% <> APILOCK_READONLY Then MsgBox("Error selecting table")
```

C 言語では次の形式を使用します。

```
short WINAPI EntSelectAdd(short wTabId, SIGNA sigKey, HSELECT hSelect, short
wAttr, void FAR * lpPtar, void FAR * lpPtac, CALLBACKSEL SelCallback, LPARAM
lParam)
```

## EntSelectTable( ) - 使用する表の選択

この関数は、1 つまたは複数の表を選択します。この関数は EntSelect( )関数と同じように機能しますが、apiStruct を渡すことができます。関連付けられた表を個別に選択するのではなく、OR 演算と ID\_ASSOC 定数を使用して、表および表に関連付けられた表を同時に選択してください。詳しくは、[339 ページの「関連付けられている表」](#)を参照してください。

**注：** 表を使用した後は、EntUnSelect( )を呼び出して表の選択を解除してください。

次の形式を使用します。

```
Declare Function EntSelectTable Lib "heaccess.dll" Alias "_EntSelectTable@40" (ByVal
hApp As Long, ByVal wTabId As Integer, ByVal sigKey As Long, ByVal wAttr As Integer,
wLockMode As Integer, ByVal lpPtar As Long, ByVal lpPtac As Long, ByVal SelCallback
As Long, ByVal lParam As Long, apiS As APISTRUCT) As Long
```

変数	説明
hApp	アプリケーションハンドル
wTabId	選択する表の ID
sigKey	NONE
wAttr	必要なアクセス権。読み取り専用アクセスで表を開くには、APILOCK_READONLY を使用します。
wLockMode	付与されたアクセス権を返すためのバッファ。表のアクセス権が付与された場合は APILOCK_READONLY が返され、拒否された場合は NONE が返されます。
lpPtar	NULL (未実装)
lpPtac	NULL (内部での API の使用のために予約済み)

## 変数 説明

SelCallback	NULL、またはユーザ定義のコールバック関数へのポインタ。詳しくは、 <a href="#">112 ページの「選択のコールバック」</a> を参照してください。
lParam	ユーザ定義のコールバック関数に返されるパラメータ
apiS	NULL、または詳細情報を含んでいる apiStruct 構造体を指すポインタ

戻りコード：

コード	説明
選択された表のハンドル	成功
NONE	エラー発生

例：

```
hSelect& = EntSelectTable(hApp&, HYP_ID_CATEGORY OR
    HYP_ID_ASSOC, HYP_NONE, APILOCK_READONLY, ret%, 0, 0,
    0, 0, 0)
If ret% <> APILOCK_READONLY Then MsgBox("Error selecting table")
```

C 言語では次の形式を使用します。

```
HSELECT WINAPI EntSelectTable(HAPP hApp, short wTabId, SIGNA sigKey, short
wAttr, short FAR * lpwLockMode, void FAR * lpPtar, void FAR * lpPtac, CALLBACKSEL
SelCallback, LPARAM lParam, LPAPISTRUCT pApiS)
```

## EntSelectTableAdd( ) - 使用する追加の表の選択

この関数は、表を選択し、既存の表選択ハンドルに追加します。これは EntSelectAll( )関数と同じように機能しますが、apiStruct を渡すことができます。関連付けられた表を個別に選択するのではなく、OR 演算と ID\_ASSOC 定数を使用して、表および表に関連付けられた表を同時に選択してください。詳しくは、[339 ページの「関連付けられている表」](#)を参照してください。

**注：** 表を使用した後は、EntUnSelect( )を呼び出して表の選択を解除してください。

特定の表にこの関数を使用する方法については、[262 ページの「EntSelectTableAdd\( \) - 使用する追加の表の選択」](#)を参照してください。

次の形式を使用します。

```
Declare Function EntSelectTableAdd Lib "heaccess.dll" Alias "_EntSelectTableAdd@36"
(ByVal wTabId As Integer, ByVal sigKey As Long, ByVal hSelect As Long, ByVal wAttr As
Integer, ByVal lpPtar As Long, ByVal lpPtac As Long, ByVal SelCallback As Long, lParam
As Long, apiS As APISTRUCT) As Integer
```

変数	説明
wTabId	選択する表の ID
sigKey	NONE、または関連する表のキー。関連する表が既に選択されている必要があります。
hSelect	EntSelect( )からの現在選択されている表のハンドル
wAttr	必要なアクセス権。読み取り専用アクセスで表を開くには、APILOCK_READONLY を使用します。
lpPtar	NULL (未実装)
lpPtac	NULL (内部での API の使用のために予約済み)
SelCallback	NULL、またはユーザ定義のコールバック関数へのポインタ。詳しくは、 <a href="#">112 ページの「選択のコールバック」</a> を参照してください。
lParam	ユーザ定義のコールバック関数に返されるパラメータ
apiS	NULL、または詳細情報を含んでいる apiStruct 構造体を指すポインタ

戻りコード：

コード	説明
付与されたアクセス権	成功
NONE	エラー発生

C 言語では次の形式を使用します。

```
short WINAPI EntSelectTableAdd(short wTabId, SIGNA sigKey, HSELECT hSelect, short wAttr, void FAR * lpPtar, void FAR * lpPtac, CALLBACKSEL SelCallback, LPARAM lParam, LPAPISTRUCT pApiS)
```

## EntSelectTableAdd( ) - 特定の表に関する注意事項

仕訳帳詳細表 (ID\_JOURNAL\_DETAIL、ID\_JOURNAL\_HISTORY\_DETAIL および ID\_JOURNAL\_TEMPLATES\_DETAIL) には、apiStruct を設定する必要があります ([110 ページの「apiStruct 構造体の使用」](#)を参照)。以下の表に示すフィールドを使用して apiStruct を設定してください。

フィールド	説明
sigCat	データ種別記号。
lStartPeriod	期間
lEndPeriod	期間 (仕訳帳は 1 つの期間のみに適用されるため、lStartPeriod と同じ)

適切な仕訳帳の記号を sigKey 引数として渡します。詳しくは、[113 ページの「関連する表と高度な関数」](#)を参照してください。

表 ID\_JOURNAL\_PERIOD\_INFO には類似の apiStruct が必要です。次にその例を示します。

```
Dim apiS As APISTRUCT
Call EntInitApiStruct(hApp&, apiS)
apiS.sigCat = sigCat&
apiS.lStartPeriod = 0    '1番目の期間
apiS.lStartPeriod = 0    '開始時期と同じ
ret% = EntSelectTableAdd(HYP_ID_JOURNAL_DETAIL, sigJour&,
hSelect&, APILOCK_READONLY, 0, 0, 0, 0, apiS)
If ret% <> APILOCK_READONLY Then MsgBox("Error selecting
table")
```

## EntSetActiveModule( ) - アクティブモジュールの設定

この関数は、Hyperion Enterprise アプリケーションモジュールのアプリケーションのユーザレポートに表示されるプログラム名を Hyperion Enterprise に伝えます。詳しくは、『Hyperion Enterprise 管理者用ガイド』の「アプリケーションの保守」の章を参照してください。

次の形式を使用します。

```
Declare Function EntSetActiveModule Lib "heaccess.dll" Alias
"_EntSetActiveModule@12" (ByVal hApp As Long, ByVal wModule As Integer, ByVal
szText As String) As Integer
```

### 変数 説明

hApp     アプリケーションハンドル

wModule   TOOLINC.H ファイルに定義される MF\_HACCESS

szText     システムレポートに表示されるテキスト

戻りコード：

コード	説明
0	成功
NONE	エラー発生

C 言語では次の形式を使用します。

```
short WINAPI EntSetActiveModule(HAPP hApp, short wModule, LPSTR szText)
```

## EntSharesExtract( ) - 株式の抽出

EntSharesExtract()は、株式を ASCII ファイルに抽出します。この関数を呼び出す前に、次の表を選択してください。

- デフォルトの表（データ種別、組織、勘定科目表。詳しくは、[111 ページの「高度な関数を使用した表の選択と選択解除」](#)を参照してください。）

- ID\_SHARES および関連付けられている表 - APILOCK\_READONLY
- ID\_SECURITY および関連付けられている表 - APILOCK\_READONLY

EntSharesExtract()を呼び出す前に、データの視点を適切なデータ種別に設定します。詳しくは、[121 ページの「データの視点の設定」](#)を参照してください。

**注：** Visual Basic プログラマは、EntSharesExtract()の代わりに EntSharesExtractVB()を使用する必要があります。

C 言語では次の形式を使用します。

**short WINAPI EntSharesExtract(HSELECT hSelect, LPSHRSEXTRACTSTRUCT pArgs, LPAPISTRUCT lpApiStruct)**

変数	説明
hSelect	選択された表のハンドル
pArgs	株式抽出用の引数を含んでいる構造体
pArgs->dwsz	株式の抽出構造体のサイズ
pArgs->hProcess	サーバ上で実行する場合は、ここに [タスクステータス] ウィンドウのプロセスハンドルが返されます。
pArgs->cDelim	フィールドの区切り文字で、通常はカンマ (,)
pArgs->nMode	すべてのエンティティ、選択したグループまたは選択したエンティティを抽出するために使用する各モード。 pArgs.nMode に使用する値は次のとおりです。 <ul style="list-style-type: none"> <li>● SHR_EXT_ALLENT - すべてのエンティティを抽出</li> <li>● SHR_EXT_SELGRP - グループ内のすべてのエンティティを抽出</li> <li>● SHR_EXT_SELENT - 選択したエンティティのみ (pSigs.sigNode) を抽出</li> </ul>
pArgs->sigNode	選択したエンティティの記号
pArgs->bRemoteFlag	リモートまたはローカルの抽出。リモートの抽出の場合は TRUE、ローカルの抽出の場合は FALSE。
pArgs->fnCallBack	未使用
pArgs->lParam	未使用
pArgs->szExtractFile	抽出先のファイルの名前
lpApiStruct	APISTRUCT 構造体のポインタ
lpApiStruct->1StartPeriod	最初の期間 (0 ベース)
lpApiStruct->1EndPeriod	株式を抽出する最後の期間 (0 から始まる)

戻りコード：

コード	説明
0	成功
NONE	エラー発生

例：

[266 ページの「EntSharesExtractVB\(\) - 株式の抽出」](#) の例を参照してください。

## EntSharesExtractVB( ) - 株式の抽出

Visual Basic プログラマは、EntSharesExtract( )の代わりにこの関数を使用してください。この関数は、株式を ASCII ファイルに抽出します。この関数を呼び出す前に、次の表を選択してください。

- デフォルトの表（データ種別、組織、勘定科目表。詳しくは、[111 ページの「高度な関数を使用した表の選択と選択解除」](#)を参照してください。）
- ID\_SHARES および関連付けられている表 - APILOCK\_READONLY
- ID\_SECURITY および関連付けられている表 - APILOCK\_READONLY

EntSharesExtract()を呼び出す前に、データの視点を適切なデータ種別に設定します。詳しくは、[121 ページの「データの視点の設定」](#)を参照してください。

この関数は EntSharesExtract()に似ていますが、抽出ファイルの名前が pArgs.lpszExtractFile の代わりに関数の引数として渡されます。

次の形式を使用します。

```
Declare Function Lib "heaccess.dll" Alias "_@16" (ByVal hSelect As Long, pArgs As SHRSEXTRACTSTRUCT, ByVal szExtractFile As String, apiS as ApiStruct) As Integer
```

変数	説明
hSelect	選択された表のハンドル
pArgs	株式抽出用の引数を含んでいる構造体。ToolKit.bas を参照してください。コールバック関数（NULL）と出力フィールド以外のすべてのフィールドに入力する必要があります。
pArgs.dwsize	株式の抽出構造体のサイズ
pArgs.hProcess	サーバ上で実行する場合は、ここに [タスクステータス] ウィンドウのプロセスハンドルが返されます。
pArgs.cDelim	フィールドの区切り文字で、通常はカンマ (,)
pArgs.nMode	すべてのエンティティ、選択したグループまたは選択したエンティティを抽出するために使用する各モード。 <ul style="list-style-type: none"> <li>● SHR_EXT_ALLENT - すべてのエンティティを抽出</li> <li>● SHR_EXT_SELGRP - グループ内のすべてのエンティティを抽出</li> <li>● SHR_EXT_SELENT - 選択したエンティティのみ（pArgs.sigNode）を抽出</li> </ul>
pArgs.sigNode	選択したエンティティの記号

変数	説明
pArgs.bRemoteFlag	リモートまたはローカルの抽出。リモートの抽出の場合は TRUE、ローカルの抽出の場合は FALSE。
pArgs.fnCallBack	未使用
pArgs.lParam	未使用
szExtractFile	株式抽出先のファイル名
apiS	ApiStruct 構造体（ユーザ定義タイプ）
apiS.lStartPeriod	最初の期間（0 ベース）
apiS.lEndPeriod	株式を抽出する最後の期間（0 から始まる）

戻りコード：

コード	説明
0	成功
NONE	エラー発生

例：

```

Dim shrsExt As ShrsExtractStruct
Dim apiStruct As apiStruct
Dim ret%
shrsExt.dwSize = LenB(shrsExt)
shrsExt.cDelim = Asc(",")
If optAllEnt.Value = True Then
    shrsExt.nMode = SHR_EXT_ALLENT
    'すべてのエンティティ
Else
    If optSelGrp.Value = True Then
        shrsLoad.nMode = SHR_EXT_SELGRP
        '選択されたグループ
    Else
        shrsExt.nMode = SHR_EXT_SELENT
        '選択されたエンティティ
    End If
End If
shrsExt.sigNode = EntGetEntitySig(hSelect&, szEntity, HYP_NAME_IMPLICIT)
shrsExt.bRemoteFlag = 0
shrsExt.fnCallBack = 0
shrsExt.lParam = 0
shrsExt.lpszFileName = 0
Call EntInitApiStruct(hApp&, apiStruct)
apiStruct.lStartPeriod = 0                                '1
番目の期間
apiStruct.lEndperiod = 11
'12番目の期間
ret% = EntSharesExtractVB(hSelect&, shrsExt, szFileName, apiStruct)

```

## EntSharesLoad( ) - 株式の読み込み

EntSharesLoad()は、ASCII ファイルから株式を読み込みます。この関数を呼び出す前に、次の表を選択してください。

- デフォルトの表（データ種別、組織、勘定科目表。詳しくは、[111 ページの「高度な関数を使用した表の選択と選択解除」](#)を参照してください。）
- ID\_SHARES および関連付けられている表 - APILOCK\_READWRITE
- ID\_SECURITY および関連付けられている表 - APILOCK\_READONLY

**注：** Visual Basic プログラマは、EntSharesLoad()の代わりに EntSharesLoadVB()を使用する必要があります。

C 言語では次の形式を使用します。

```
short WINAPI EntSharesLoad(HSELECT hSelect, LPSHRSLOADSTRUCT  
lpShrsLoadStruct, LPAPISTRUCT lpApiStruct)
```

変数	説明
hSelect	選択された表のハンドル
lpShrsLoadStruct	株式読み込み用の引数を含んでいる構造体
lpShrsLoadStruct->dwsz	株式の読み込み構造体のサイズ
lpShrsLoadStruct->hProcess	サーバ上で実行する場合は、ここに [タスクステータス] ウィンドウのプロセスハンドルが返されます。
lpShrsLoadStruct->cDelim	フィールドの区切り文字で、通常はカンマ (,)
lpShrsLoadStruct->nMode	株式の結合または置換モード： <ul style="list-style-type: none"><li>● SHR_MERGE - 結合</li><li>● SHR_REPLACE - 置換</li></ul>
lpShrsLoadStruct->bRemoteFlag	サーバの読み込みの場合は TRUE、ローカルの読み込みの場合は FALSE
lpShrsLoadStruct->wExtFlag	0。将来拡張予定。
lpShrsLoadStruct->fnCallBack	読み込み用のコールバック関数。 EntSharesLoad()用のコールバック関数および EntSharesLoadVB()は、CALLBACKSHARES として定義されているタイプになっている必要があります。詳しくは、 <a href="#">329 ページの「CALLBACKSHARES」</a> を参照してください。
lpShrsLoadStruct->lParam	コールバック関数の引数
lpShrsLoadStruct->pszFileName	読み込むファイル
lpApiStruct	NULL。API 互換性のための構造体

戻りコード：

コード	説明
0	成功
NONE	エラー発生

例：

[266 ページの「EntSharesExtractVB\(\) - 株式の抽出」](#) の例を参照してください。

## EntSharesLoadVB( ) - 株式の読み込み

Visual Basic プログラマは、EntSharesLoad()の代わりにこの関数を使用してください。この関数は、株式を ASCII ファイルから読み込みます。この関数を呼び出す前に、次の表を選択してください。

- デフォルトの表（データ種別、組織、勘定科目表。詳しくは、[111 ページの「高度な関数を使用した表の選択と選択解除」](#)を参照してください。）
- ID\_SHARES および関連付けられている表 - APILOCK\_READWRITE
- ID\_SECURITY および関連付けられている表 - APILOCK\_READONLY

この関数は EntSharesLoad()に似ていますが、抽出ファイルの名前が pArgs.lpszFileName の代わりに関数の引数として渡されます。

次の形式を使用します。

```
Declare Function EntSharesLoadVB Lib "heaccess.dll" Alias "_EntSharesLoadVB@16"  
(ByVal hSelect As Long, pArgs As SHRSLOADSTRUCT, ByVal szLoadFile As String, apiS  
as APIStruct) As Integer
```

変数	説明
hSelect	選択された表のハンドル
pArgs	株式読み込み用の引数を含んでいる構造体。ToolKit.bas を参照してください。コールバック関数（使用しない場合は NULL (0)）と出力フィールド以外のすべてのフィールドに入力する必要があります。コールバック関数（fnCallBack）は、サーバで実行するときではなく、ローカルで実行するときのみ使用されます。
pArgs.dwsz	株式の読み込み構造体のサイズ
pArgs.hProcess	サーバ上で実行する場合は、ここに [タスクステータス] ウィンドウのプロセスハンドルが返されます。
pArgs.cDelim	フィールドの区切り文字で、通常はカンマ (,)
pArgs.nMode	結合または置換モード： <ul style="list-style-type: none"><li>● SHR_MERGE - 結合</li><li>● SHR_REPLACE - 置換</li></ul>
pArgs.bRemoteFlag	サーバの読み込みの場合は TRUE、ローカルの読み込みの場合は FALSE

変数	説明
pArgs.wExtFlag	将来拡張予定。0
pArgs.fnCallBack	読み込み用のコールバック関数。詳しくは、 <a href="#">329 ページの「CALLBACKSHARES」</a> を参照してください。
pArgs.lParam	コールバック関数の引数
sxLoadFile	読み込むファイル
apiS	API 互換性のための構造体。NULL。

戻りコード：

コード	説明
0	成功
NONE	エラー発生

例：

**注：** 次の例の EntSharesLoadVB0( )は EntSharesLoadVB( )と同じようにして宣言されますが、apiStruct の代わりに 0 を渡すことができます。詳しくは、[16 ページの「Visual Basic プログラミング上の注意」](#)を参照してください。

```
Dim shrsLoad As ShrsLoadStruct Dim ret%

shrsLoad.dwSize = LenB(shrsLoad) shrsLoad.cDelim = Asc(",")

If optMerg.Value = True Then
shrsLoad.nMode = SHR_MERGE
Else
shrsLoad.nMode = SHR_REPLACE
End If

shrsLoad.bRemoteFlag = False
shrsLoad.wExtFlag = 0
shrsLoad.fnCallBack = 0
shrsLoad.lParam = 0
shrsLoad.lpszFileName = 0
ret% = EntSharesLoadVB0(hSelect&,shrsLoad,szFileName,0)
```

## EntUNCDataLoad( ) - データの読み込み

この関数は、データの読み込みを実行します。

**注：** この関数は、Visual Basic 以外のすべての言語で使用できます。Visual Basic アプリケーションには、EntUNCDataLoadVB( )を使用します。

サーバ上でデータを読み込む場合は、DB\_REMOTE ビットフラグが pArgs->wLoadFlag に設定されます。読み込み処理のプロセスハンドルを認識し、プログラムによって起動されるサーバプロセスを追跡および監視できるようにすると便利です。プロセスハンドルを取得するには、EntUNCDataLoad( )を呼び出す前に、lParam を HANDLE \*変数に設定します。返されるハンドルは、ローカルクライアントコンピュータで実行されている [タスクステータス] ウィンドウプログラム HCOMMGR.EXE のインスタンスのプロセスハンドルです。詳しくは、[121 ページ](#)の「[サーバのタスク](#)」を参照してください。

C 言語では次の形式を使用します。

short WINAPI EntUNCDataLoad(HSELECT hSelect, LPCDBLOADSTRUCT\_UNC pArgs, CALLBACKDBLOAD fnCallBack, LPARAM lParam)

変数	説明
hSelect	選択された表のハンドル
pArgs	データ読み込み用の引数を含んでいる構造体
pArgs->dwSize	pArg 構造体のサイズ
pArgs->cDelim	フィールドの区切り文字で、通常はカンマ (,)
pArgs->cNeg	負の区切り文字で、通常はダッシュ (-) またはかっこ([])
pArgs->cScale	データの単位
pArgs->cView	データ表示形式 (FORMVIEW_YTD、FORMVIEW_PER、FORMVIEW_CAT)、またはデータ種別のデフォルトの表示形式を使用する場合は NONE
pArgs->dDataBy	sOperation に使用するデータ
pArgs->sOperation	演算 (dDataBy による除算または乗算)
pArgs->iFileListCnt	ファイル一覧表内の要素の数
pArgs->lpFileList	選択されたパス\ファイル名の配列。これによって、ユーザは異なる場所から複数のデータファイルを同時に読み込むことができます。 <b>注：</b> この配列内の各ファイル名の長さは 260 (MAX_PATH) バイトで、ファイル名の最後の文字の後には NULL で終了している必要があります。ファイル名文字列のいずれかの長さが 260 (MAX_PATH) バイトでない場合、ファイルの一覧を適切に解析できなくなるので、関数は終了します。
pArgs->szLoadErrFileName	エラーファイル名
pArgs->sigAcctConv	勘定科目変換表記号
pArgs->sigCurCat	現在選択されているデータ種別
pArgs->sigFrequency	期間単位

変数	説明
pArgs->sigNameConv	エンティティ変換表記号
pArgs->wLoadFlag	データベース読み込みオプションフラグ
fnCallBack	ステータス情報用のコールバック関数（DB_REMOTE ビットフラグが pArgs->wLoadFlag に設定されている場合には使用されません）。詳しくは、 <a href="#">324 ページの「CALLBACKDBLOAD - EntUNCDataLoad()の場合」</a> を参照してください。
IParam	通常は、fnCallBack コールバック関数の引数。ただし、DB_REMOTE ビットフラグが pArgs->wLoadFlag に設定されている場合、コールバック関数は使用されないの、IParam が NULL でない場合には HANDLE *であるものとみなされます。

戻りコード：

コード	説明
0	成功
NONE	エラー発生
LOAD_CANCELLED	ユーザによるキャンセル
LOAD_CHECK_ERRORLOG	エラーがアプリケーションのエラーログに含まれています。
LOAD_CHECKBOTH	エラーがアプリケーションのエラーログと読み込みエラーファイルの両方に書き込まれています。

EntUNCDataLoad( )を呼び出してサーバ上でタスクを実行する前に、データの視点を必ず設定してください。現在の組織、データ種別およびエンティティを設定します。詳しくは、[121 ページの「データの視点の設定」](#)を参照してください。

例：

[272 ページの「EntUNCDataLoadVB\(\) - データの読み込み」](#)の例を参照してください。EntUNCDataLoadVB( )の追加の引数は、pArgs 内の適切なフィールドとして渡されます。

## EntUNCDataLoadVB( ) - データの読み込み

この関数は Visual Basic 用で、データの読み込みを実行します。

EntUNCDataLoadVB( )は文字列のローカルコピーを作成するので、コールバック関数の実行中、Visual Basic では文字列は移動せず、文字列のアドレスを DBLOADSTRUCT\_UNC 構造体のコピーに配置して、EntUNCDataLoad( )を呼び出します。

次の形式を使用します。

```
Declare Function EntUNCDataLoadVB Lib "heaccess.dll" Alias
    "_EntUNCDataLoadVB@24" (ByVal hSelect As Long, pArgs As DBLOADSTRUCT_UNC,
```

ByVal lpfnCallBack As Long, ByVal lParam As Long, ByVal szFileList As String, ByVal szLoadErrFileName As String) As Integer

変数	説明
hSelect	選択された表のハンドル
pArgs	データ読み込み用の引数を含んでいる構造体を指すポインタ
pArgs.dwSize	pS 構造体のサイズ
pArgs.cDelim	フィールドの区切り文字で、通常はカンマ (,)
pArgs.cNeg	負の区切り文字で、通常はダッシュ (-) またはアポストロフィ (')
pArgs.cScale	データの単位
pArgs.cView	データ表示形式 (FORMVIEW_YTD、FORMVIEW_PER、FORMVIEW_CAT)、またはデータ種別のデフォルトの表示形式を使用する場合は NONE
pArgs.dDataBy	sOperation に使用するデータ
pArgs.sOperatio	演算 (dDataBy による除算または乗算)
pArgs.sigAcctConv	勘定科目変換表記号
pArgs.sigCurCat	現在選択されているデータ種別
pArgs.sigFrequency	期間単位
pArgs.sigNameConv	エンティティ変換表記号
pArgs.iFileListCnt	szFileList 内のファイル名の数
pArgs.wLoadFlag	データベース読み込みオプションフラグ
fnCallBack	ステータス情報のコールバック関数 (DB_REMOTE ビットフラグが pArgs->wLoadFlag に設定されている場合は使用されません)。詳しくは、 <a href="#">331 ページの「CALLBACKVB」</a> を参照してください。
lParam	通常は、fnCallBack コールバック関数の引数。ただし、DB_REMOTE ビットフラグが pArgs.wLoadFlag に設定されている場合、コールバック関数は使用されないため、lParam が NULL 以外の場合にはプロセスハンドルを配置するためのバッファであるものとみなされます。
szFileList	<p>選択されたファイル名を含む配列のポインタ。各ファイル名のサイズは 260 (MAX_PATH) バイトで、NULL で終了しています。または、pArgs.lpFileList を既に設定している場合 (文字列の代わりにバイト配列または構造体配列を使用する場合は、NULL になります)。</p> <p><b>注：</b> この配列内の各ファイル名の長さは 260 (MAX_PATH) バイトで、ファイル名の最後の文字の後には NULL で終了している必要があります。ファイル名文字列のいずれかの長さが 260 (MAX_PATH) バイトでない場合、ファイルの一覧を適切に解析できなくなるので、関数は終了します。</p>
szLoadErrFileName	エラーファイル名

szFileList 引数に NULL (0 または vbNullString) を渡した場合、EntDataLoadVB() によって DBLOADSTRUCT 構造体内の lpFileList フィールドが上書きされることはな

くなります。このため、レコードの配列などのファイル名一覧に、文字列以外のデータタイプを使用してみることができます。

コールバック関数の引数は、EntUNCDataLoad( )のコールバック関数よりも少なくなっています。詳しくは、[331 ページの「CALLBACKVB」](#)を参照してください。

戻りコード：

コード	説明
0	成功
NONE	エラー発生
LOAD_CANCELLED	ユーザによるキャンセル
LOAD_CHECK_ERRORLOG	エラーがアプリケーションのエラーログに含まれています。
LOAD_CHECKBOTH	エラーがアプリケーションのエラーログと読み込みエラーファイルの両方に書き込まれています。

例：

```
Dim s As DBLOADSTRUCT_UNC
Dim szFile260 As String * 260
'ファイル名は260文字の文字列で、これらの260文字内でNULLで終了している必要あり
LSet szFile260$ = szFile$
Mid(szFile260$, Len(szFile$) + 1) = Chr(0)
szFile$ = szFile260$
'さまざまなラジオボタンやチェックボックスに基づいて読み込みオプションを設定
flag% = 0
If optAccum.Value Then
    flag% = flag% Or DB_ACCUM
Else
If optReplace.Value Then
    flag% = flag% Or DB_REPLACE
Else
If optMerge.Value Then
    flag% = flag% Or DB_MERGE
End If
If chkLogic.Value Then flag% = flag% Or DB_EXECLOG
If chkCalcAcct.Value Then flag% = flag% Or DB_REPCALCACCTS
If chkZero.Value Then flag% = flag% Or DB_ZEROFORNO
'DBLOADSTRUCT_UNC内のフィールドを設定
hSelect& = HypGethSelect(frmMain.ghRApp%)
s.dwSize = LenB(s)
s.cDelim = Asc(",")
s.cNeg = Asc("-")
s.cScale = 255 'デフォルトにはNONEを使用
s.cView = Asc(FORMVIEW_CAT)
s.dDataBy = -1 'sOperationに使用するデータ
s.sOperation = HYP_NONE
s.iFileListCnt = 1 'ファイル一覧表内の要素の数
s.sigAcctConv = HYP_NONE '勘定科目変換表記号
s.sigCurCat = sigCat& '現在選択されているデータ種別
s.sigFrequency = HYP_NONE '期間単位記号
s.sigNameConv = HYP_NONE 'エンティティ変換表記号
```

```

s.wLoadFlag = flag%           'データベース読み込みオプションフラグ
(DB_MERGEなど)
'この例ではコールバック関数にSpyworksを使用
ret% = EntUNCDataLoadVB(hSelect&, s, cbkLoad.ProcAddress, 0,
    szFile$, szErrFile$)
.
.
.
'SpyWorksコールバック関数
Private Sub cbkLoad_cbxLLL(lval1 As Long, lval2 As Long, lval3 As Long,
    retval As Long)
Dim ret%, wLen%
Dim szFile$
Dim args As CALLBACKSTRUCT
'lval1は'ByVal szFile As String'になる必要あり
'lval1からszFile$を取得
wLen% = EntVBGetCStrLen(lval1)
szFile$ = Space(wLen%)
ret% = EntVBCopyStr(szFile$, lval1, wLen%)
'lval2は実際には'args As CALLBACKSTRUCT'になる必要あり
'argsをlval2から取得
ret% = EntVBCopyData(args, lval2, LenB(args))
ret% = MyCallBack (szFile$, args, lval3)
retval& = ret%
EndSub
Function MyCallBack (szfile$, args As CALLBACKSTRUCT, ByVal lparam&) As
Integer
Dim szCat$, szEntity$, szMsg$
szMsg$ = "File: [" & szFile$
szCat$ = Space(HYP_SIZEELABEL + 1)
ret% = EntQueryExStr0(args.hSelect, HYP_ID_CATEGORY, HYP_NAME,
    args.sigCat, HYP_NONE, Len(szCat$), szCat$, 0)
If ret% = 0 Then szCat$ = CToBStr(szCat$)
szMsg$ = szMsg$ & "], Category: [" & szCat$
szEntity$ = Space(HYP_SIZEFULLNAME + 1)
ret% = EntQueryExStr0(args.hSelect, HYP_ID_NAMES, HYP_NAME,
    args.sigEntity, HYP_NONE, Len(szEntity$), szEntity$, 0)
If ret% = 0 Then szEntity$ = CToBStr(szEntity$)
szMsg$ = szMsg$ & "], Entity: [" & szEntity$
szMsg$ = szMsg$ & "]" Period " & Str(args.lStart) & " to Period "
    & Str(args.lEnd)
ret% = MsgBox(szMsg$, vbOKOnly, "Now Loading...")
MyCallBack% = 0
End Function

```

**注：** EntQueryExStr0( )は EntQueryExStr( )と同じようにして宣言されますが、apiStruct 引数の代わりに 0 を渡すことができるように調整されます。詳しくは、[16 ページの「Visual Basic プログラミング上の注意」](#)を参照してください。

C 言語では次の形式を使用します。

EntUNCDataLoadVB(HSELECT hSelect, LPCDBLOADSTRUCT\_UNC pS,  
CALLBACKVB fnCallBack, LPARAM lParam, LPCSTR szFileList, LPCSTR  
szLoadErrFileName)

## EntUnSelect( ) - 表の選択解除

この関数は、1 つまたは複数の表を選択します。この関数によって表選択ハンドルが変更される場合があるので、その後の API 呼び出しでは入力値ではなく戻り値を使用してください。指定されている表に関連付けられているすべての表の選択を解除するには、ID\_ASSOC 定数に OR 演算を使用する必要があります。詳しくは、[339 ページの「関連付けられている表」](#)を参照してください。

**注：** NONE を表 ID として使用して、選択されているすべての表の選択を解除することもできます。

ID\_DATAFILE 表の選択を解除するには、データ種別、エンティティ、種別および親エンティティ（種別が ID\_REGULAR 以外の場合）を含む apiStruct を渡します。各フィールドは、表の選択時に EntDataFileOpen( )に渡した値と同じ値を含んでいる必要があります。

**注：** スプレッドシートのアドイン関数 HypConstruct( )を使用してアプリケーションを開いた場合は、EntUnSelect( )を呼び出した後に HypSethSelect( )を呼び出してください。HypSethSelect( )では、Hyperion Enterprise Retrieve アプリケーションハンドルに内部で関連付けられている表選択ハンドルを更新します。

次の形式を使用します。

Declare Function EntUnSelect Lib "heaccess.dll" Alias "\_EntUnSelect@24" (ByVal hSelect As Long, ByVal wTabId As Integer, ByVal sigKey As Long, ByVal bDiscard As Integer, wRC As Integer, apiS As APISTRUCT) As Long

### 変数 説明

hSelect	選択された表のハンドル
wTabId	選択解除する表の ID。表 ID のいずれかを指定するか、すべてを選択解除する場合は NONE を指定します。
sigKey	NONE、または表の選択時にキーを指定した場合には関連する表のキー
bDiscard	表をメモリから自動的に削除する場合は 1 (TRUE)、表をメモリに保持する場合は 0 (False)
wRC	エラーコードを返すためのバッファ。実行に成功した場合は 0 (ゼロ) が返され、失敗した場合はゼロ以外または NONE が返されます。
apiS	NULL、または表の選択に使用したものと同様の apiStruct

戻りコード

この関数は、同じまたは新しい表選択ハンドルを返します。すべての表の選択が解除された場合は NULL を返します。

例：

```
hselect& = EntUnSelect(hselect&, HYP_ID_ACCTLISTENTRY or  
    HYP_ID_ASSOC, sigList&, False, ret%, 0)
```

C 言語では次の形式を使用します。

```
HSELECT WINAPI EntUnSelect(HSELECT hSelect, short wTabId, SIGNA sigKey, BOOL  
bDiscard, short FAR * lpwRC, LPAPISTRUCT pApiS)
```

## EntUpdate( ) - レコードの更新

この関数は、表のレコード内で指定されているフィールドの値を更新します。データの更新が終わったら、EntSave() を呼び出して変更を実際に保存する必要があります。Visual Basic プログラムは、文字列値の更新に EntUpdateStr() 関数を使用する必要があります。EntUpdateStr() は、EntUpdate() の宣言のクローンを文字列用に変更したもので、ToolKit.bas ファイルで宣言されます。

---

**注意** この関数は、データファイル表 (ID\_DATAFILE) のみに使用してください。この関数を使用する前に、この関数を使用したデータの更新方法について説明した [117 ページの「データの更新」](#) を参照してください。

---

次の形式を使用します。

```
Declare Function EntUpdate Lib "heaccess.dll" Alias "_EntUpdate@28" (ByVal hSelect As  
Long, ByVal wTabId As Integer, ByVal wAttr As Integer, ByVal sigRecd As Long, ByVal  
sigKey As Long, pzBuf As Any, apiS As APISTRUCT) As Integer
```

### 変数 説明

hSelect 選択された表のハンドル

wTabId データファイル表、ID\_DATAFILE

wAttr 更新するフィールドの属性。有効な属性については、[117 ページの「データの更新」](#) を参照してください。

sigRecd 更新するレコードの記号。多くの ID\_DATAFILE 更新属性には、NONE を使用できます。

sigKey 表 ID\_DATAFILE の場合は NONE

pzBuf 指定されているフィールドの新しい値。新しい値が apiS にある場合は表 ID\_DATAFILE を示す NULL。

apiS 詳細情報を含んでいる apiStruct 構造体。ほとんどの ID\_DATAFILE 更新属性に必要です。

戻りコード：

コード	説明
0	成功
NONE	エラー発生

例：

[117 ページの「データの更新」](#)の例を参照してください。

C 言語では次の形式を使用します。

```
short WINAPI EntUpdate(HSELECT hSelect, short wTabId, short wAttr, SIGNA sigRecd,
SIGNA sigKey, void FAR * pzBuf, LPAPISTRUCT pApiS)
```

## EntUpdateDefault( ) - デフォルトの更新

この関数は、アプリケーションまたはユーザのデフォルトを更新します。値の更新が終わったら、EntSaveDefault( )を呼び出して変更を保存します。Visual Basic プログラマは、文字列値のクエリに EntUpdateDefaultStr( )関数を使用する必要があります。EntUpdateDefaultStr()は、EntUpdateDefault()の宣言のクローンを文字列用に変更したもので、ToolKit.bas ファイルで宣言されます。

次の形式を使用します。

```
Declare Function EntUpdateDefault Lib "heaccess.dll" Alias "_EntUpdateDefault@16"
(ByVal hApp As Long, ByVal wTabId As Integer, ByVal wAttr As Integer, pzBuf As Any)
As Integer
```

### 変数 説明

hApp アプリケーションハンドル

wTabId 次の表のいずれかの表 ID

- ID\_HAPP
- ID\_APPDEFAULT
- ID\_USERDEFAULT

wAttr 更新する属性。表 ID\_APPDEFAULT および ID\_USERDEFAULT に有効な属性は、これらの表のクエリ属性と同じです。ID\_HAPP に有効な属性は HYP\_HAPP\_IS\_SILENT のみです。クエリ属性については、[付録 B「クエリ属性」](#)を参照してください。

pzBuf フィールドの新しい値

ID\_APPDEFAULT 表に有効な属性は、次の例外を除いた任意のクエリ属性です。

- APP\_NTDATADIR
- APP\_NTINBOXDIR
- APP\_NTOUTBOXDIR
- APP\_NTREPORTDIR
- APP\_SITE\_SHIFTED

さまざまなアプリケーションディレクトリ（APP\_INBOXDIR など）を更新するときは、パスの先頭の適切な位置で@APP マクロを使用します。クエリ属性 APP\_NTINBOXDIR を要求しない限り、EntQueryDefault( )によって@APP がアプリケーションパスに自動的に置換されます。詳しくは、『Hyperion Enterprise 管理者用ガイド』の「アプリケーションの作成」の章を参照してください。

**注意** 表 ID\_APPDEFAULT 内の多くのアプリケーション設定は、アプリケーション作成後に変更しないようにする必要があります。編集可能なメニュー項目やフィールドについては、Hyperion Enterprise アプリケーションモジュールを確認してください。変更不可のフィールドに対応しているアプリケーション設定は変更しないでください。変更すると、アプリケーションが破損するおそれがあります。

戻りコード：

コード	説明
0	成功
NONE	エラー発生

例：

```
wRet = EntUpdateDefault(hApp, ID_USERDEFAULT, CURORSIG,sigOrg);
wRet = EntUpdateDefault(hApp,ID_USERDEFAULT,
    CURCATEGORYSIG,sigCat);
wRet = EntUpdateDefault(hApp, ID_USERDEFAULT,
    CURNAMESIG,sigEntity);
```

C 言語では次の形式を使用します。

```
short WINAPI EntQueryDefault(HAPP hApp,short wTabId,short wAttr,short wLen,void
FAR * pzBuf)
```

## EntWriteAppProfileLong( ) - アプリケーションプロファイルの long 整数の書き込み

この関数は、指定されているアプリケーション.INI ファイルに数値を書き込みます。

**注：** 通常 Hyperion Enterprise では、引用符で囲まれた文字列として数値を書き込みます。詳しくは、[280 ページの「EntWriteAppProfileString\( \) - アプリケーションプロファイルの string の書き込み」](#)を参照してください。

アプリケーションの.INI ファイル（または SQL でそれに相当するもの）の値を更新するには、SQL アプリケーションおよびデータベースのアプリケーションのどちらに対しても、ID\_APPDEFAULT 表を指定して EntUpdateDefault( )関数を使用する必要があります。

次の形式を使用します。

```
Declare Function EntWriteAppProfileLong Lib "heaccess.dll" Alias
"_EntWriteAppProfileLong@12" (ByVal szAppName As String, ByVal szKeyName As
String, ByVal lValue As Long) As Long
```

変数	説明
----	----

szAppName	アプリケーション ID
-----------	-------------

szKeyName	値のキー名
-----------	-------

lValue	書き込む数値
--------	--------

戻りコード：

コード	説明
TRUE	成功
FALSE	エラー発生

C 言語では次の形式を使用します。

```
BOOL WINAPI EntWriteAppProfileLong(LPCSTR szAppName, LPCSTR szKeyName,
long lValue)
```

## EntWriteAppProfileString( ) - アプリケーションプロファイルの string の書き込み

この関数は、指定されているアプリケーション.INI ファイルに文字列を書き込みます。

アプリケーションの.INI ファイル（または SQL でそれに相当するもの）の値を更新するには、SQL アプリケーションおよびファイルベースのアプリケーションのどちらに対しても、ID\_APPDEFAULT 表を指定して EntUpdateDefault( )関数を使用する必要があります。

次の形式を使用します。

```
Declare Function EntWriteAppProfileString Lib "heaccess.dll" Alias
"_EntWriteAppProfileString@12" (ByVal szAppName As String, ByVal szKeyName As
String, ByVal szValue As String) As Long
```

変数	説明
----	----

szAppName	アプリケーション名
-----------	-----------

szKeyName	文字列のキー名
-----------	---------

szValue	書き込む文字列。引用符で囲んだ文字列を.INI ファイルにそのまま書き込む場合は、この文字列の適切な場所に引用符（"）を配置します。
---------	--

戻りコード：

コード	説明
TRUE	成功
FALSE	エラー発生

C 言語では次の形式を使用します。

```
BOOL WINAPI EntWriteAppProfileString(LPCSTR szAppName, LPCSTR szKeyName,
LPCSTR szValue)
```

## EntWriteProfileLong( ) - Hyperion プロファイルの long 整数の書き込み

この関数は、HYPENT.INI ファイルに数値を書き込みます。

次の形式を使用します。

```
Declare Function EntWriteProfileLong Lib "heaccess.dll" Alias
"_EntWriteProfileLong@12" (ByVal szAppName As String, ByVal szKeyName As String,
ByVal lValue As Long) As Long
```

### 変数 説明

szAppName アプリケーション名、または [Default] セクションの場合は NULL

szKeyName 数値のキー名

lValue 書き込む数値

戻りコード：

コード	説明
TRUE	成功
FALSE	エラー発生

例：

```
ret& = EntWriteProfileLong(vbNullString, "Organization", 0)
```

C 言語では次の形式を使用します。

```
BOOL WINAPI EntWriteProfileLong(LPCSTR szAppName, LPCSTR szKeyName, long
lValue)
```

## EntWriteProfileString( ) - Hyperion プロファイルの string の書き込み

この関数は、HYPENT.INI ファイルに文字列を書き込みます。

次の形式を使用します。

Declare Function EntWriteProfileString Lib "heaccess.dll" Alias  
"\_EntWriteProfileString@12" (ByVal szAppName As String, ByVal szKeyName As String,  
ByVal szValue As String) As Long

**変数 説明**

szAppNam アプリケーション名、または [Default] セクションの場合は NULL

szKeyName 文字列のキー名

szValue 書き込む文字列

戻りコード :

コード	説明
1	成功
0	エラー発生

例 :

```
ret& = EntWriteProfileString(vbNullString, "AppID", szApp$)
```

C 言語では次の形式を使用します。

BOOL WINAPI EntWriteProfileString(LPCSTR szAppName, LPCSTR szKeyName,  
LPCSTR szValue)

## EntVBCGetCStrLen( ) - C 文字列の長さの取得

この関数は Visual Basic 専用です。一部のコールバック関数には、C スタイル文字列のアドレスである引数が渡されます。Visual Basic では、コールバック関数の引数タイプは long になります。long 引数を Visual Basic の文字列に変換するには、EntVBCGetCStrLen( ) と EntVBCopyStr( ) を使用できます。

EntVBCGetCStrLen( ) を呼び出して文字列の長さを取得し、同じ長さの Visual Basic 文字列を割り当て、EntVBCopyStr( ) を呼び出して C スタイルの文字列を Visual Basic の文字列にコピーします。関数 EntVBCopyStr( ) は、EntVBCopyData のクローンです。)

次の形式を使用します。

Declare Function EntVBCGetCStrLen Lib "heaccess.dll" Alias "EntVBCGetCStrLen@4"  
(ByVal lAddr As Long) As Integer

ここで、lAddr は C スタイルの文字列のアドレスです。

戻りコード :

コード	説明
文字列の長さ	成功

例 :

```
wLen% = EntVBGetCStrLen(lval1&)
szFile$ = Space(wLen%)
ret% = EntVBCopyStr(szFile$, lval1&, wLen%)
```

## EntVBCopyData( ) - データのコピー

この関数は Visual Basic 専用です。一部のコールバック関数の引数は、実際には構造体（ユーザ定義タイプ）のアドレスになります。コールバック関数で引数が long タイプとして宣言される場合は、EntVBCopyData( )を使用してその構造体を正しいタイプの Visual Basic 変数にコピーし、使用可能にすることができます。

次の形式を使用します。

```
Declare Function EntVBCopyData Lib "heaccess.dll" Alias "_EntVBCopyData@12"
(Dest As Any, ByVal lAddr As Long, ByVal wLen As Integer) As Integer
```

### 変数 説明

Dest コピー先バッファ

lAddr コピー元のアドレス

wLen コピーする長さ

戻りコード：

コード	説明
0	成功
NONE	エラー発生

例：

```
Dim args As CALLBACKSTRUCT
ret% = EntVBCopyData( args, lval2&, LenB(args))
```

## EntVBCopyStr( ) - 文字列のコピー

EntVBCopyStr( )は、C スタイルの文字列引数を Visual Basic の文字列にコピーするために使用できます。EntVBCopyStr( )は、EntVBCopyData( )の代わりに宣言できます。詳しくは、[282 ページの「EntVBGetCStrLen\( \) - C 文字列の長さの取得」](#)を参照してください。

次の形式を使用します。

```
Declare Function EntVBCopyStr Lib "heaccess.dll" Alias "_EntVBCopyData@12" (ByVal
Dest As String, ByVal lAddr As Long, ByVal wLen As Integer) As Integer
```

### 変数 説明

Dest コピー先バッファ

## 変数 説明

lAddr コピー元のアドレス

wLen コピーする長さ

戻りコード：

コード	説明
0	成功
NONE	エラー発生

## HypGethApp( ) - アプリケーションハンドルの取得

ほとんどの高度な Hyperion Enterprise 関数では、アプリケーションハンドルを EntOpenApplication( ) から引数として取得するか、表選択ハンドルを EntSelect( ) から取得します。一方、スプレッドシートのアドイン関数では、Hyperion Enterprise Retrieve アプリケーションハンドルを HypConstruct( ) から引数として取得します。HypGethApp( ) は、Hyperion Enterprise Retrieve ハンドルに関連付けられているアプリケーションハンドルを返します。

次の形式を使用します。

```
Declare Function HypGethApp Lib "heaccess.dll" Alias "_HypGethApp@4"(ByVal hRApp As Integer) As Long
```

ここで、hRApp は HypConstruct( ) からの Hyperion Enterprise Retrieve アプリケーションハンドルです。

戻りコード：

コード	説明
アプリケーションハンドル	成功
NULL	エラー発生

例：

```
hApp% = HypGethApp(hRApp%)
```

C 言語では次の形式を使用します。

```
HAPP WINAPI HypGethApp(HRETREIVEAPP hRApp)
```

## HypGethSelect( ) - 表選択ハンドルの取得

ほとんどの高度な Hyperion Enterprise 関数では、表選択ハンドルを EntSelect( ) から引数として取得します。一方、スプレッドシートのアドイン関数では、Hyperion Enterprise Retrieve アプリケーションハンドルを HypConstruct( ) から引数として取得します。HypGethSelect( ) は、指定された Hyperion Enterprise Retrieve アプリケーションハンドルに関連付けられている表選択ハンドルを返します。

次の形式を使用します。

**Declare Function HypGethSelect Lib "heaccess.dll" Alias "\_HypGethSelect@4"(ByVal hRApp As Integer) As Long**

ここで、hRApp は HypConstruct( )からの Hyperion Enterprise Retrieve アプリケーションハンドルです。

戻りコード：

コード	説明
hselect アプリケーションハンドル	成功
NULL	エラー発生

例：

hSelect& = **HypGethSelect** (hRApp%)

C 言語では次の形式を使用します。

**HSELECT WINAPI HypGethSelect(HRETREIVEAPP hRApp)**

## HypGethSelect( ) - 表選択ハンドルの設定

HypSethSelect( )では、Hyperion Enterprise Retrieve アプリケーションハンドルに内部で関連付けられている表選択ハンドル (hSelect) を更新します。この関数は、HypConstruct( )または HypMultiInit( )を使用してアプリケーションを開き、その後 EntUnSelect( )を呼び出した場合にのみ呼び出してください。EntUnSelect( )を呼び出すと、表選択ハンドルが変更される場合があります。HypSethSelect( )は、表選択ハンドルが変更されていることをスプレッドシートのアドイン関数に伝えます。

次の形式を使用します。

**Declare Function HypSethSelect Lib "heaccess.dll" Alias "\_HypSethSelect@8"(ByVal hRApp As Integer, ByVal hSelect As Long) As Integer**

### 変数 説明

hRApp HypConstruct( )からの Hyperion Enterprise Retrieve アプリケーションハンドル

hSelect 新しい表選択ハンドル

戻りコード：

コード	説明
0	成功
NONE	エラー発生

例：

hSelect& = **EntUnSelect** (hSelect&, HYP\_ID\_SHARES or HYP\_ID\_ASSOC, sigCat&, 0, ret%, 0)  
ret% = **HypSethSelect** (hRApp%, hSelect&)

C 言語では次の形式を使用します。

```
SBOOL WINAPI HypSethSelect(HRETRIEVEAPP hRApp, HSELECT hSelect)
```

## この章の内容

Hyperion Enterprise SE 変換の概要 .....	287
Hyperion Enterprise のインクルードファイル .....	287
関数の分類.....	288
サポートされていない関数.....	290
Hyperion Enterprise SE のコードのアップグレード .....	291
関数のアップグレード.....	297
Hyperion Enterprise の表の変更 .....	306

この章では、Hyperion Enterprise SE のプログラムを変換して Hyperion Enterprise で使用する方法を説明します。

## Hyperion Enterprise SE 変換の概要

Hyperion Enterprise SE 用開発者ツールキットを使用して作成された 16 ビットプログラムは、少なくとも 32 ビットプログラムに変換する必要があります。32 ビットプログラムは調整と再コンパイルが必要です。詳しくは、[291 ページの「Hyperion Enterprise SE のコードのアップグレード」](#)の手順 1 と 2 を参照してください。

下位互換性のため、以前の機能の大半はまだサポートされていますが（「サポートされていない機能」のリストを参照）、Hyperion Enterprise の新機能をサポートするには、コードもアップグレードする必要があります。これらの機能には、大きいアプリケーションと大きい ID（以前は短い名前と呼ばれていました）が含まれます。詳しくは、[291 ページの「Hyperion Enterprise SE のコードのアップグレード」](#)の手順 3 を参照してください。

開発者用ツールキットの新機能については、[287 ページの「Hyperion Enterprise のインクルードファイル」](#)を参照してください。

## Hyperion Enterprise のインクルードファイル

Hyperion Enterprise では次の 2 つのインクルードファイルを提供しています。

- TOOLKIT.BAS（Microsoft Visual Basic での作業用）
- TOOLKIT.H（C および C++での作業用）

インクルードファイルについては、[287 ページの「Hyperion Enterprise のインクルードファイル」](#)を参照してください。

**注：** Hyperion Enterprise 提供のインクルードファイルを使用してプログラムを再コンパイルする場合は、アプリケーションハンドルを Hyperion Enterprise Retrieve アプリケーションハンドルとして定義し直す必要があります。詳しくは、[292 ページの「アプリケーションハンドルの再定義」](#)を参照してください。

## 関数の分類

このガイドでは、Hyperion Enterprise API 関数を次のグループに分けて説明します。

- スプレッドシートアドイン関数。アプリケーション用に最適化されています。これらの関数については、[第 2 章「スプレッドシートのアドイン関数」](#)で説明しています。
- 表を操作するためのその他のサポートされている関数。スプレッドシートアドイン関数と併用できるように設計されています。これらの関数については、[第 3 章「表関数」](#)で説明しています。
- 高レベル関数。基本となる Hyperion Enterprise API 関数の効力を最大限に活用します。これらの関数については、[第 4 章「高度な関数」](#)で説明しています。

Hyperion Enterprise SE 関数のすべて。最初の 2 グループでまだサポートされています。これらの関数については、[288 ページの「サポートされているスプレッドシートアドイン関数」](#)または[289 ページの「サポートされているその他の関数」](#)を参照してください。

高レベル関数は、Hyperion Enterprise に新しく加わった関数で、異なる引数を使用します。これらはすべて Ent...() の形式になります。これらの関数については、[289 ページの「新しい Ent...\(\) 関数」](#)を参照してください。

## サポートされているスプレッドシートアドイン関数

これらの関数は、Hyperion Enterprise Retrieve などのスプレッドシートアドイン製品で使用するために最適化されています。通常、最も安全で簡単に使用できる関数です。これらの関数は Hyperion Enterprise アプリケーション専用で、HypConstruct() または HypMultiInit() を使用して開きます。

以下の関数は、以前のリリースの Hyperion Enterprise と同じです。

- スプレッドシートアドイン関数は、読み取りを加速するために、最近使用したデータファイル表を自動的にバッファに入れます。これらは HypHPLNK() 関数を使用して、データの書き込みを自動的にバッファに入れます。HypHPCCommit2() 関数は、バッファ内のデータ書き込みを最適化してパフォーマンスを向上させます。

- `HypConstruct()` と `HypMultiInit()` は、さまざまな内部バッファを初期化し、このグループの他の関数で使用するデフォルトの表を選択します。
- これらの関数を使用すると、一度に最大 20 個の Hyperion Enterprise アプリケーションを開くことができます。`HypConstruct()`、`HypMultiDefault()` および `HypMultiGet()` は、他のスプレッドシートアドイン関数で使用するためのハンドルをアプリケーションに返します。

以下の関数は、以前のリリースの Hyperion Enterprise から変更されています。

- これらの関数の引数と戻り値は Hyperion Enterprise SE と同じですが、Hyperion Enterprise SE でタイプ HAPP のアプリケーションハンドルであったものは、Hyperion Enterprise Retrieve のタイプ HRETRIEVEAPP のアプリケーションハンドルになっています。詳しくは、[292 ページの「アプリケーションハンドルの再定義」](#)を参照してください。
- Hyperion Enterprise SE の一部の関数は、Hyperion Enterprise の大きいアプリケーションや ID を処理するためにクローンされています。これらの関数の新しいバージョンは `Hyp...Ex()` の形式になります。詳しくは、[第 2 章「スプレッドシートのアドイン関数」](#)で説明しています。これらの関数にアップグレードするには、[293 ページの「古い関数から新しい Hyp...\(\) 関数への置き換え」](#)を参照してください。使用されなくなった関数は記載されていません。
- Hyperion Enterprise と Hyperion Enterprise Retrieve の新しい関数をサポートするために、新しい `Hyp...()` 関数が追加されました。

## サポートされているその他の関数

下位互換のため、多くの Hyperion Enterprise SE 関数が Hyperion Enterprise でもまだサポートされています。これらの関数はスプレッドシートアドイン関数で簡単に使用できます。詳しくは、[第 3 章「表関数」](#)で説明しています。

これらの関数の一部は、Hyperion Enterprise リリースの大きいアプリケーションや ID を処理するためにクローンされています。これらの関数の新しいバージョンは、通常は `Hyp...Ex()` の形式になります。詳しくは、[第 3 章「表関数」](#)で説明しています。これらの関数にアップグレードするには、[293 ページの「古い関数から新しい Hyp...\(\) 関数への置き換え」](#)を参照してください。使用されなくなった関数は記載されていません。

これらの関数が Hyperion Enterprise Retrieve のアプリケーションハンドルを使用し、スプレッドシートアドイン関数と対で使用されることを強調するため、形式は `Hyp...()` のままです。サポートされているその他の `Hyp...()` 関数には、新しい `Ent...()` 関数が使用されます。これらの関数を対応する `Ent...()` 関数に置き換える方法については、[295 ページの「古い関数から新しい Ent...\(\) 関数への置き換え」](#)を参照してください。

## 新しい Ent...() 関数

Hyperion Enterprise には新しい関数が多数追加されています。これらの関数は Hyperion Enterprise の内部 API で使用するもので、`Ent...()` の形式になります。

新しい関数は、引数として表選択ハンドル `hSelect` またはアプリケーションハンドル `hApp` を使用します。Hyperion Enterprise Retrieve のアプリケーションハンドル `hRetrieveApp` は引数として使用しません。

これら新しい関数の多くには、`apiStruct` と定義される構造体の引数もあります。詳しくは、[110 ページの「apiStruct 構造体の使用」](#)を参照してください。

スプレッドシートアドイン関数およびその他のサポートしている `Hyp...` 関数を新しい `Ent...` 関数と混ぜることもできます。`hRetrieveApp` と内部で関連付けられている `hSelect` を使用するには、[113 ページの「スプレッドシートのアドイン関数と高度な関数の組み合わせ」](#)を参照してください。

## サポートされていない関数

Hyperion Enterprise および Hyperion Enterprise SE の関数グループはサポートされていません。通常は、新しい関数か、それと同等の結果が得られるメソッドがあります。以下の関数はサポートされていません。

- `HypAppendNode()`
- `HypConsolidate()`
- `HypCreateAcctTxt()`
- `HypCreateCatTxt()`
- `HypCreateNameTxt()`
- `HypDelete()`
- `HypDelNode()`
- `HypInsChildNode()`
- `HypInsert()`
- `HypInsParentNode()`
- `HypInsSiblingNode()`
- `HypJLockPeriodsSet()`
- `HypLockPeriodsSet()`
- `HypPeriodsSetEx()`
- `HypPsf_GetStatus()`
- `HypPsf_ImpactStatus()`
- `HypRetMultiAsk()`
- `HypSave()`
- `HypSetTopNode()`
- `HypUnJLockPeriodsSet()`
- `HypUnLinkName()`
- `HypUnlockPeriodsSet()`
- `HypUpdate()`

- HypUpdNode( )

## Hyperion Enterprise SE のコードのアップグレード

コンパイルできるように、Hyperion Enterprise SE のコードをアップグレードする必要があります。コードの C には新しい TOOLKIT.BAS インクルードファイルまたは TOOLKIT.H ファイルを使用してください。また、Hyperion Enterprise の大きいアプリケーション、レコード記号および ID で使用するためにもアップグレードが必要です。

**注：** 以下の手順の 1 と 2 は必須です。手順 3 は、Hyperion Enterprise の大きいアプリケーション、レコード記号および ID をサポートする場合にお勧めします。記述されていない関数のうち Hyperion Enterprise でサポートされているものは、可能な限り Hyperion Enterprise SE をエミュレートします。ただし、新しい Hyp...( )関数または新しい Ent...( )関数に置き換える場合は、Hyperion Enterprise で違いを調整する必要があります。詳しくは、[297 ページの「関数のアップグレード」](#) および [306 ページの「Hyperion Enterprise の表の変更」](#) を参照してください。

➤ Hyperion Enterprise SE コードをアップグレードするには、次の手順に従います。

- 1 アプリケーションハンドルをすべて Hyperion Enterprise Retrieve のアプリケーションハンドルに定義し直します。この手順を実行するまでは、コードを再コンパイルできません。詳しくは、[292 ページの「アプリケーションハンドルの再定義」](#) を参照してください。
- 2 16 ビットプログラムを 32 ビットプログラムに変換します。詳しくは、[292 ページの「16 ビットプログラムから 32 ビットプログラムへの変換」](#) を参照してください。

**注：** Hyperion Enterprise API の 32 ビットバージョンを使用した場合は、プログラムが Microsoft Visual Basic で作成されていれば、コードの関数宣言を新しい TOOLKIT.BAS ファイルの関数宣言に置き換える必要があります。

3 次のタスクを実行して関数をアップグレードします。

- Hyperion Enterprise SE の古い関数を、それぞれ対応する新しい Hyp...( )関数に置き換えます。これらの関数の一覧は、[293 ページの「古い関数から新しい Hyp...\( \)関数への置き換え」](#) を参照してください。または、Hyperion Enterprise SE の古い関数の多くは、対応する新しい Ent...( )関数に置き換えることができます。これらの関数一覧は、[295 ページの「古い関数から新しい Ent...\( \)関数への置き換え」](#) を参照してください。
- 必要に応じて、特定の関数をアップグレードします。詳しくは、[297 ページの「関数のアップグレード」](#) を参照してください。
- Hyperion Enterprise の表の変更に合わせてコードをアップグレードします。詳しくは、[306 ページの「Hyperion Enterprise の表の変更」](#) を参照してください。

## アプリケーションハンドルの再定義

これで、アプリケーションハンドルに使用した変数（C 言語ではタイプ `HAPP`）が Hyperion Enterprise Retrieve のアプリケーションハンドル（C ではタイプ `HRETRIEVEAPP`）になっているはずです。これは `HypConstruct()`、`HypMultiGet()` および `HypMultiDefault()` から返されたハンドルで、その他すべてのスプレッドシートアドイン関数と他のサポートされている関数によって使用されます。

## Visual Basic プログラミングに関する注意

Hyperion Enterprise SE で 16 ビットのコンパイラを使用している場合、ハンドルのタイプは両方とも `Integer` です。新しい `Ent...()` 関数で使用する `Long` 変数であるアプリケーションハンドルとの混乱を避けるために、これらの変数名を `hApp%` から `hRApp%` に変更することをお勧めします。

Hyperion Enterprise SE で 32 ビットのコンパイラを使用している場合は、これらの変数を `Long` から `Integer` に変更してください。

## C プログラミングに関する注意

プログラムを再コンパイルする前に、これらの変数をタイプ `HRETRIEVEAPP` として定義し直してください。Hyperion Enterprise SE で 32 ビットのコンパイラを使用していた場合は、基本タイプが `long` 整数（`DWORD`）から `short` 整数に変わったため、再コンパイルする必要があります。16 ビットコンパイラを使用していた場合は、タイプを `HRETRIEVEAPP` に変更し、プログラムを 32 ビットプログラムに変換する必要があります。

## 16 ビットプログラムから 32 ビットプログラムへの変換

16 ビットプログラムはサポートされなくなりました。既存のアプリケーションを 32 ビットプログラムに変換する必要があります。

- 16 ビットプログラムを 32 ビットプログラムに変換するには、次の手順に従います。
  - プログラムが Microsoft Visual Basic で作成されている場合は、コードの関数宣言を新しい `TOOLKIT.BAS` ファイルの 32 ビット関数宣言に置き換える必要があります。32 ビットの関数宣言は、ライブラリ名として `lib HACCESS.DLL` ではなく `lib HACCESS32.DLL` を使用します。また、後述するように、エイリアスや別の数値タイプも使用します。
  - プログラムが C または他の言語（Visual Basic 以外）で作成されている場合は、プログラムを `HACCESS32.LIB` ではなく `HACCESS32.LIB` にリンクする必要があります。Microsoft Visual Basic の場合は、32 ビットの関数宣言がリンクを処理します。
  - 16 ビットのプログラムで `short` 整数（16 ビット）であった関数引数と戻りコードは、32 ビットのプログラムでは `long` 整数（32 ビット）です。C 言語では、

これらはタイプ `int` として定義され、どちらの場合も正しくコンパイルされます。Microsoft Visual Basic では、これらをタイプ `integer` からタイプ `long` に変更する必要があります。どの変数を変更するかを判断するには、古い関数宣言と新しい関数宣言を比較します。

- 32 ビットバージョンのコンパイラでプログラムを再コンパイルします。詳しくは、[292 ページの「アプリケーションハンドルの再定義」](#)を参照してください。

**ヒント：** `TOOLKIT.BAS` ファイルには 32 ビットおよび 16 ビットの関数宣言が含まれています。32 ビットバージョンの Visual Basic は 32 ビットの宣言を使用します。16 ビットバージョンの Visual Basic は、サポートされなくなった 16 ビットの宣言を使用します。

## 古い関数から新しい `Hyp...()` 関数への置き換え

Hyperion Enterprise SE の一部の関数は、Hyperion Enterprise の大きいアプリケーションや ID を処理するためにクローンされています。以下は、Hyperion Enterprise SE の古い関数と、対応する Hyperion Enterprise の `Hyp...()` 関数の違いです。

- Hyperion Enterprise ではレコード記号はすべて 32 ビット (`long`) です。Hyperion Enterprise SE ではレコード記号はほとんどが 16 ビット (`short` 整数) です。
- 場合によっては、文字列を多く割り当てる必要があります。たとえば、Hyperion Enterprise ではほとんどの場合、ID（以前の短い名前）は 20 文字 (`SIZELABEL`) です。Hyperion Enterprise SE では、ID は通常 8 文字です。
- Hyperion Enterprise SE で 32 ビットのコンパイラを使用した場合は (`HEACCESS.DLL` ファイルヘリンク)、新しい関数のほとんどが `short` 整数を返すことに気付くはずですが、Hyperion Enterprise SE では、古い関数は通常 32 ビットコンパイラに対して `long` 整数を返しました。

[表 48](#) は、使用されなくなった関数と、それに代わる新しい `Hyp...()` 関数を示します。

**表 48** 使用されなくなった関数と新しい `Hyp...()` 関数

使用されなくなった関数	新しい関数
<code>AreValidLnkParams()</code>	<a href="#">73 ページの「HypValidateLnkParams() - リンクパラメータのチェック」</a> *
<code>EntFormatNumber()</code>	<a href="#">192 ページの「EntFormatNumber2() - 数値の書式設定」</a>
<code>HypAcctListAsk()</code>	<a href="#">28 ページの「HypAcctListAskEx() - 勘定科目一覧の選択」</a>
<code>HypCatAsk()</code>	<a href="#">28 ページの「HypCatAskEx() - データ種別の選択」</a>
<code>HypCatGetNumPeriods()</code>	<a href="#">83 ページの「HypCatGetNumPeriodsEx() - データ種別に含まれる期間数の算出」</a>
<code>HypCatGetPerShort()</code>	<a href="#">84 ページの「HypCatGetPerShortEx() - データ種別期間単位内の期間 ID の取得」</a>

使用されなくなった関数	新しい関数
HypCatMapPeriod( )	84 ページの「HypCatMapPeriodEx( ) - 期間の期間単位へのマッピング」
HypEnum( )	88 ページの「HypEnum( ) - 表内のレコードの列挙」
HypEnumAcctListEntries( )	85 ページの「HypEnumAcctListEntriesEx( ) - 勘定科目一覧の勘定科目の列挙」
HypEnumAcctSig( )	88 ページの「HypEnum( ) - 表内のレコードの列挙」
HypEnumNameListEntries( )	90 ページの「HypEnumNameListEntriesEx( ) - エンティティ一覧内のエンティティの列挙」
HypFind( )	94 ページの「HypFindEx( ) - 記号の検索」
HypFindNameInOrg( )	95 ページの「HypFindNameInOrgEx( ) - ノード ID の検索」
HypGetAcctSig( )	94 ページの「HypFindEx( ) - 記号の検索」 *
HypGetCatPeriod( )	96 ページの「HypGetCatPerFreq( ) - データ種別内の期間番号と期間単位の取得」
HypGetCatPeriodEx( )	96 ページの「HypGetCatPerFreq( ) - データ種別内の期間番号と期間単位の取得」
HypGetPerView( )	100 ページの「HypGetPerViewEx( ) - 期間単位と表示形式の取得」
HypGetRptFreq( )	105 ページの「HypQueryEx( ) - Hyperion Enterprise の表のクエリ」 *
HypGetTopNode( )	101 ページの「HypGetTopNodeEx( ) - トップノードの取得」
HypHPCommit( )	39 ページの「HypHPCommit2( ) - データバッファの書き込み」
HypHPCUR( )	40 ページの「HypHPCUREx( ) - コンポーネントの通貨の取得」
HypHPKEY( )	50 ページの「HypHPKEYEx( ) - デフォルトのエンティティ、データ種別、勘定科目、または期間の取得」
HypHPLogicCommit( )	39 ページの「HypHPCommit2( ) - データバッファの書き込み」
HypIsNameParent( )	102 ページの「HypIsNameParentEx( ) - エンティティが親であることの確認」
HypLock( )	103 ページの「HypLockEx( ) - 表の選択」
HypNamAsk( )	67 ページの「HypNamAskEx( ) - エンティティの選択」
HypNameListAsk( )	68 ページの「HypNameListAskEx( ) - エンティティ一覧の選択」
HypQryAcctListEntry()	105 ページの「HypQueryEx( ) - Hyperion Enterprise の表のクエリ」 *
HypQryNameListEntry()	105 ページの「HypQueryEx( ) - Hyperion Enterprise の表のクエリ」 *
HypQryNode()	105 ページの「HypQueryEx( ) - Hyperion Enterprise の表のクエリ」 *
HypQryRptFreq( )	105 ページの「HypQryRptFreqEx( ) - レポート期間単位のクエリ」
HypQuery( )	105 ページの「HypQueryEx( ) - Hyperion Enterprise の表のクエリ」

使用されなくなった関数	新しい関数
HypQuerySig()	<a href="#">105 ページの「HypQueryEx( ) - Hyperion Enterprise の表のクエリ」</a> *
HypUnLock()	<a href="#">106 ページの「HypUnLockEx( ) - 表の選択解除」</a>
ValidateHypParams( )	<a href="#">74 ページの「HypValidateParams( ) - パラメータのチェック」</a> *

\* オプション

**注：** 前の表では、一部の変更がオプションとなっていますが、使用されなくなった関数はこのガイドに記載されていません。

使用されなくなった関数は、可能な限り Hyperion Enterprise SE の動作をエミュレートします。ただし、新しい Hyp...()関数に置き換える場合は、Hyperion Enterprise でその違いを調整する必要があります。詳しくは、「関数のアップグレード」および[306 ページの「Hyperion Enterprise の表の変更」](#)を参照してください。

古い関数の多くは、新しい Ent...()関数に置き換えることができます。一覧を確認するは、以下のトピックを参照してください。

## 古い関数から新しい Ent...()関数への置き換え

[表 49](#) に、新しい Ent...()に置き換え可能な Hyperion Enterprise SE の関数すべてを示します。これらの新しい関数は、引数として HRETRIEVEAPP ではなく hSelect を使用します。詳しくは、[113 ページの「スプレッドシートのアドイン関数と高度な関数の組み合わせ」](#)を参照してください。

**表 49** Hyperion Enterprise SE の関数と対応する Ent...()関数

Hyperion Enterprise SE 関数	Ent...()関数
DateConv( ) *	<a href="#">157 ページの「EntDateConv( ) - 日付の変換と期間番号の計算」</a>
HypAcctAsk( ) *	<a href="#">127 ページの「EntAcctAsk( ) - 勘定科目の選択」</a>
HypAcctListAsk( )	<a href="#">128 ページの「EntAcctListAsk( ) - 勘定科目一覧の選択」</a>
HypCatAsk( )	<a href="#">140 ページの「EntCatAsk( ) - データ種別の変更」</a>
HypCatGetNumPeriods( )	<a href="#">140 ページの「EntCatGetNumPeriods( ) - データ種別の期間数の計算」</a>
HypCatGetPerShort( )	<a href="#">141 ページの「EntCatGetPerShort( ) - 期間ラベルの取得」</a>
HypCatMapPeriod( )	<a href="#">142 ページの「EntCatMapPeriod( ) - 期間から期間単位へのマッピング」</a>
HypConsolidate( )	<a href="#">143 ページの「EntConsolidate( ) - 連結」</a>
HypDBExtract( )	<a href="#">149 ページの「EntDataExtract( ) - データの抽出」</a>
HypDBLoad( )	<a href="#">270 ページの「EntUNCDataLoad( ) - データの読み込み」</a>

Hyperion Enterprise SE 関数	Ent...()関数
HypDFOpen( )	<a href="#">155 ページの「EntDataFileOpen( ) - データファイルを開く」</a>
HypDFClose( )	<a href="#">276 ページの「EntUnSelect( ) - 表の選択解除」</a>
HypDFSave( )	<a href="#">250 ページの「EntSave( ) - 表の保存」</a>
HypDFLogicExecute( )	<a href="#">277 ページの「EntUpdate( ) - レコードの更新」</a>
HypEnum( )	<a href="#">171 ページの「EntEnum( ) - 表内のレコードの列挙」</a>
HypEnumAcctListEntries( )	<a href="#">171 ページの「EntEnum( ) - 表内のレコードの列挙」</a>
HypEnumAcctSig( )	<a href="#">171 ページの「EntEnum( ) - 表内のレコードの列挙」</a>
HypEnumApplications( )*	<a href="#">182 ページの「EntEnumApplications( ) - アプリケーションの列挙」</a>
HypEnumNameListEntries( )	<a href="#">171 ページの「EntEnum( ) - 表内のレコードの列挙」</a>
HypEnumOrgNames( )*	<a href="#">183 ページの「EntEnumOrgEntities( ) - 組織内のノードの列挙」</a>
HypEnumSubAcctSig( )*	<a href="#">184 ページの「EntEnumSubAcctSig( ) - サブ勘定科目の列挙」</a>
HypFind( )	<a href="#">186 ページの「EntFind( ) - 記号の検索」</a>
HypFindNameInOrg( )	<a href="#">190 ページの「EntFindEntityInOrg( ) - ノード ID の検索」</a>
HypFreqAsk( )*	<a href="#">193 ページの「EntFreqAsk( ) - 期間単位の選択」</a>
HypGetAcctSig( )	<a href="#">186 ページの「EntFind( ) - 記号の検索」</a>
HypGetCatPeriod( )	<a href="#">198 ページの「EntGetCatPerFreq( ) - データ種別内の期間番号と期間単位の取得」</a>
HypGetCatPeriodEx( )	<a href="#">198 ページの「EntGetCatPerFreq( ) - データ種別内の期間番号と期間単位の取得」</a>
HypGetChild( )*	<a href="#">199 ページの「EntGetChild( ) - 子ノードの取得」</a>
HypGetNameSig( )*	<a href="#">200 ページの「EntGetEntitySig( ) - エンティティ記号の取得」</a>
HypGetOrgLevel( )*	<a href="#">204 ページの「EntGetOrgLevel( ) - 組織内のレベルの取得」</a>
HypGetPerView( )	<a href="#">204 ページの「EntGetPerView( ) - 期間単位と表示形式の取得」</a>
HypGetRptFreq( )	<a href="#">247 ページの「EntQueryEx( ) - クエリ情報」</a>
HypGetSibling( )*	<a href="#">207 ページの「EntGetSibling( ) - 兄弟ノードの取得」</a>
HypGetTopNode( )	<a href="#">247 ページの「EntQueryEx( ) - クエリ情報」</a>
HypIsNameParent( )	<a href="#">210 ページの「EntIsEntityParent( ) - エンティティが親であるかどうかの確認」</a>
HypLock()	<a href="#">258 ページの「EntSelect( ) - 使用する表の選択」</a> および <a href="#">260 ページの「EntSelectAdd( ) - 使用する追加の表の選択」</a>
HypNamAsk( )	<a href="#">166 ページの「EntEntityAsk( ) - エンティティの選択」</a>

Hyperion Enterprise SE 関数	Ent...()関数
HypNameListAsk( )	<a href="#">167 ページの「EntEntityListAsk( ) - エンティティー一覧の選択」</a>
HypPerAsk( )*	<a href="#">245 ページの「EntPerAsk( ) - 期間の選択」</a>
HypQryAcctListEntry()	<a href="#">247 ページの「EntQueryEx( ) - クエリ情報」</a>
HypQryNameListEntry()	<a href="#">247 ページの「EntQueryEx( ) - クエリ情報」</a>
HypQryNode()	<a href="#">247 ページの「EntQueryEx( ) - クエリ情報」</a>
HypQryRptFreq( )	<a href="#">246 ページの「EntQryRptFreq( ) - クエリのレポート期間単位」</a>
HypQuery( )	<a href="#">247 ページの「EntQueryEx( ) - クエリ情報」</a>
HypQuerySig()	<a href="#">247 ページの「EntQueryEx( ) - クエリ情報」</a>
HypUnLock()	<a href="#">276 ページの「EntUnSelect( ) - 表の選択解除」</a>

オプションです。これらの Hyperion Enterprise SE 関数はサポートされており、[第 2 章「スプレッドシートのアドイン関数」](#)と[第 3 章「表関数」](#)に記載されています。

使用されなくなった関数（オプションと記載されていないもの）は、可能な限り Hyperion Enterprise SE の動作をエミュレートします。ただし、新しい Ent...()関数に置き換える場合は、Hyperion Enterprise でその違いを調整する必要があります。詳しくは、[297 ページの「関数のアップグレード」](#)および[306 ページの「Hyperion Enterprise の表の変更」](#)を参照してください。

または、この一覧の古い関数を新しい Hyp...()関数に置き換えることもできます。詳しくは、[293 ページの「古い関数から新しい Hyp...\(\)関数への置き換え」](#)を参照してください。

**注：** 返された EntFreqAsk( )の期間単位名には、HypFreqAsk( ) よりも大きいバッファを割り当てる必要がありますが、実際にアプリケーションで大きい期間単位名が使用されるとは限りません。EntFreqAsk( )のバッファは、2(OLDSIZENAME+1)ではなく 2(SIZENAME+1)となるはずです。

## 関数のアップグレード

使用されなくなった関数を置き換える場合は、Hyperion Enterprise Release API でコードの違いを調整する必要があります。以下の関数は、置き換えるときに調整が必要になる可能性があります。

- HypLock( )と HypUnLock( )については、[298 ページの「Upgrade HypLock\( \)と HypUnLock\( \)」](#)を参照してください。
- HypFind( )については、[298 ページの「HypFind\( \)のアップグレード」](#)を参照してください。
- HypQuery( )については、[298 ページの「HypQuery\( \)のアップグレード」](#)を参照してください。

- HypQuerySig( )については、[302 ページの「HypQuerySig\( \)のアップグレード」](#)を参照してください。
- HypQryNode( )については、[303 ページの「HypQryNode\( \)のアップグレード」](#)を参照してください。
- HypQryAcctListEntry( )と HypQryNameListEntry( )。詳しくは、[304 ページの「HypQryAcctListEntry\( \)と HypQryNameListEntry\( \)のアップグレード」](#)を参照してください。

勘定科目記号を主要勘定科目記号とサブ勘定科目記号に分けた場合は、このコードを新しい関数 EntAcctSplit( )に置き換えます。詳しくは、[306 ページの「32 ビットの分割勘定科目とエンティティ記号の置き換え」](#)を参照してください。

## Upgrade HypLock( )と HypUnLock( )

HypLock( )と HypUnLock( )を HypLockEx( )と HypUnLockEx( )に置き換える場合は、関連付けられた表を個別に選択するのではなく、OR 演算を ID\_ASSOC 定数と一緒に使用します。これにより、Hyperion Enterprise SE と同様に、指定した表に関連付けられているすべての表がロックされます。詳しくは、[339 ページの「関連付けられている表」](#)を参照してください。

Hyperion Enterprise SE では、レポート表 (ID\_REPORTS) を選択するために HypLock( )を呼び出すたびに、ブック表 (ID\_BOOKS) が選択されます。アップグレードするには、HypLockEx( )を呼び出してブック表を選択します。

## HypFind( )のアップグレード

HypFind( )の呼び出しを HypFindEx( )または EntFind( )の呼び出しに置き換える場合は、ID\_FREQUENCY 表の期間単位を見つけるために HypFind( )を使用するかどうかをチェックします。Hyperion Enterprise の期間単位は長さが 1 文字です (M、D、W、Y など)。

Hyperion Enterprise SE では、渡す期間単位文字列の最初の文字のみが HypFind( )で使用されます。たとえば、期間単位 M または期間単位 M.PER を渡すと、どちらの場合も期間単位 M の記号が返されます。Hyperion Enterprise では、文字列全部が使用されます。たとえば、期間単位 M.PER を渡すと、そのような期間単位はないため、エラー (NONE) が返されます。

会社間照合セットに関連付けられているレコードを見つけるには、ID\_INTCODETではなく表 ID\_ICSETを使用します。詳しくは、[307 ページの「会社間詳細表の変更」](#)を参照してください。

## HypQuery( )のアップグレード

Hyperion Enterprise の大きい記号や ID を処理できるように、HypQuery( )の呼び出しを新しい HypQueryEx( )または EntQueryEx( )の呼び出しに置き換える場合は、以下の点に注意してください。

- Hyperion Enterprise SE で有効なクエリ属性の多くは、Hyperion Enterprise に存在しません。詳しくは、[299 ページの「使用されなくなったクエリ属性」](#)を参照してください。
- Hyperion Enterprise では、Hyperion Enterprise SE とは異なるサイズ（種類）の情報がクエリ属性によって返されます。詳しくは、[298 ページの「HypQuery\(\) のアップグレード」](#)を参照してください。
- Hyperion Enterprise SE では、多くのクエリで戻りバッファのサイズとして長さゼロを HypQuery() に渡すことが可能です。その理由は、Hyperion Enterprise SE の API でバッファが正しいサイズと想定され、長さ引数がチェックされない（特に short 整数の場合）からです。Hyperion Enterprise ではエラーメッセージが返されるため、必ず長さを渡す必要があります。

## 使用されなくなったクエリ属性

[表 50](#) に、使用されなくなったクエリ属性と、それに代わる新しいクエリ属性を示します。古いクエリ属性は HypQueryEx() または EntQueryEx() で使用できません。

**表 50** 使用されなくなったクエリ属性

表	使用されなくなったクエリ属性	新しいクエリ属性
ID_CATEGORY	CAT_DISPLAYPER	クエリ属性は元々機能しませんでした。
ID_GROUP	ACCT_GROUP	GRPACCT_CHARTORDER
ID_GROUP	ACCT_LEVEL	勘定科目にはレベルがないため該当しません。
ID_GROUP	QRYFIRST	GRPACCT_CHARTORDER
ID_INTCODET	すべて表内	詳しくは、 <a href="#">307 ページの「会社間詳細表の変更」</a> を参照してください。
ID_ROLLOVER	すべて表内	詳しくは、 <a href="#">309 ページの「期別替表の変更」</a> を参照してください。
ID_SECCATTAB	すべて表内	詳しくは、 <a href="#">311 ページの「使用されなくなったセキュリティ表」</a> を参照してください。
ID_SECCATDET	すべて表内	詳しくは、 <a href="#">311 ページの「使用されなくなったセキュリティ表」</a> を参照してください。
ID_SECNAMETAB	すべて表内	詳しくは、 <a href="#">311 ページの「使用されなくなったセキュリティ表」</a> を参照してください。
ID_SECNAMEDET	すべて表内	詳しくは、 <a href="#">311 ページの「使用されなくなったセキュリティ表」</a> を参照してください。

表	使用されなくなったクエリ属性	新しいクエリ属性
ID_SECHYPTAB	すべて表内	詳しくは、311 ページの「使用されなくなったセキュリティ表」を参照してください。
ID_SECHYPDET	すべて表内	詳しくは、311 ページの「使用されなくなったセキュリティ表」を参照してください。
ID_SECGRPTAB	すべて表内	詳しくは、315 ページの「セキュリティグループ表の変更」を参照してください。
ID_SECUSERTAB	USERTAB_GROUP USERTAB_PASSWORD USERTAB_USERID	詳しくは、315 ページの「セキュリティユーザ表の変更」を参照してください。
ID_SUBACCT (使用されなくなった表名 - ID_SUBACCTHDR)	SUBTABLE_LEVEL	該当しません。サブ勘定科目にはレベルがありません。
ID_SUBACCT (使用されなくなった表名 - ID_SUBACCTHDR)	SUBTABLE_MULTILEVEL	該当しません。サブ表はすべてマルチレベルの可能性あります。
ID_SUBACCTDETAIL (使用されなくなった表名 - ID_SUBACCTDET)	SIGNATURE	該当しません。詳細レコード記号はサブ勘定科目の記号です。
ID_SUBACCTDETAIL (使用されなくなった表名 - ID_SUBACCTDET)	QRYFIRST	表 ID_SUBACCTHDR の SUBTABLE_FIRSTDETAIL。
ID_SUBNAME	ELIMINATIONS INTERCOMPANY SUB-NAME	エラー。

## HypQuery( )クエリサイズのアップグレード

HypQuery( )では、変更されたクエリサイズが補正されますが、新しい関数 HypQueryEx( )と EntQueryEx( )ではクエリサイズの違いが補正されません。新しいクエリサイズを使用するには、コードを更新する必要があります。

以下のクエリ属性は Hyperion Enterprise SE では Integer (C 言語では short) ですが、Hyperion Enterprise ではすべてレコード記号であり long です。Hyperion Enterprise SE では Integer で Hyperion Enterprise では long の項目をクエリする際に Integer を渡すと、基本となる Hyperion Enterprise API でエラーメッセージが生成されます。

- ACCT\_GROUP
- ACCT\_SUBACCT\_TABLE
- BOTTOMLIST
- CAT\_FREQ
- CAT\_PRIORSIG

- CHARTMETHOD
- COUNT
- CURRENCY
- QRYFIRST
- QRYNEXT
- REPORT\_ID
- REPSET\_ID
- ROLLOVER\_CAT1 (ROLL\_SRCCAT)
- ROLLOVER\_CAT2 (ROLL\_DESTCAT)
- SUBACCT\_TABLE
- TOPLIST

以下のクエリ属性は、Hyperion Enterprise SE では Integer (C 言語では short) で 1 文字 (バイト) です。Byte フィールドに Integer を渡した場合、Hyperion Enterprise API では上位バイトの設定やクリアが行われなため、予期しない結果が出力される可能性があります。

- ACCT\_IS\_BALANCE
- ACCT\_IS\_INTERCO
- ACCT\_SUBACCT\_TYPE
- AUTORECALC
- CAT\_FLOWYTD
- INTERCOMPANY
- JOURNALS
- LIST\_DIRECTION
- SCALESCHED\_BOLD

現在は 1 バイトしか取得されませんが、Hyperion Enterprise では wLen = 2 の場合に 2 バイトに拡張されます。

- ACCT\_IS\_CONSOLIDATE
- ACCT\_IS\_CURRENCY
- ACCT\_IS\_SCALE
- ALLPERIODS
- CAT\_SCALE
- SCHED\_ITALIC
- SHOW\_LOGIC
- SUBACCT\_TYPE
- SUBTABLE\_IS\_INTERCO
- SUPPNODATA

- USELABEL

Hyperion Enterprise SE では、多くのクエリで戻りバッファのサイズとして長さゼロを HypQuery( ) に渡すことが可能です。これは Hyperion Enterprise では該当しません。エラーメッセージが表示されます。

Hyperion Enterprise SE ではクエリ属性 CODE で文字列を取得します。Hyperion Enterprise では関数 HypQueryEx( ) と EntQueryEx( ) でコードの記号を取得します。文字列を取得する場合は、コード表 ID\_CODES でコードの NAME をクエリする必要があります。

## HypQuerySig( ) のアップグレード

HypQuerySig( ) の呼び出しは HypQueryEx( ) または EntQuery に置き換えることができます。Hyperion Enterprise で属性のクエリサイズが Hyperion Enterprise SE と異なる場合は、HypQuerySig( ) によってこれらの変更が補正されます。HypQueryEx( ) または EntQueryEx( ) を使用する場合は、正しい長さの引数を使用する必要があります。

表 51 に、Hyperion Enterprise SE の HypQuerySig( ) に相当する Hyperion Enterprise のクエリ属性を示します。

表 51 HypQuerySig( ) クエリ属性のアップグレード

Hyperion Enterprise SE のクエリ属性	Hyperion Enterprise 相当
NAME 属性で完全な勘定科目 ID が返されます。	NAME 属性で主要勘定科目のみの ID が返されます。完全な勘定科目 ID を取得するには、ACCT_FULLNAME 属性を使用してください。
ACCT_LEVEL クエリ属性は勘定科目表 (ID_ACCOUNTS) に有効です。	勘定科目にはレベルがありません。勘定科目表 (ID_ACCOUNTS) の ACCT_LEVEL クエリ属性は該当しないため、エラーコードが返されます。
<p>以下のクエリ属性では Interger (C 言語では short) の値が取得されます。</p> <ul style="list-style-type: none"> <li>● ACCT_GROUP</li> <li>● ACCT_SUBACCT_TABLE</li> <li>● CHARTMETHOD</li> <li>● CURRENCY</li> <li>● SUB-NAME</li> </ul>	<p>以下のクエリ属性はレコード記号で、Long (C 言語では SIGNA) の値が取得されます。</p> <ul style="list-style-type: none"> <li>● ACCT_GROUP</li> <li>● ACCT_SUBACCT_TABLE</li> <li>● CHARTMETHOD</li> <li>● CURRENCY</li> <li>● SUB-NAME</li> </ul>
<p>以下のクエリ属性では Interger (C 言語では short) の値が取得されます。</p> <ul style="list-style-type: none"> <li>● ACCT_IS_BALANCE</li> <li>● ACCT_IS_CONSOLIDATE*</li> <li>● ACCT_IS_CURRENCY*</li> <li>● ACCT_IS_INTERCO</li> <li>● ACCT_IS_SCALE*</li> <li>● ACCT_SUBACCT_TYPE</li> <li>● INTERCOMPANY</li> </ul>	<p>以下のクエリ属性では 1 バイトが取得されます。</p> <ul style="list-style-type: none"> <li>● ACCT_IS_BALANCE</li> <li>● ACCT_IS_CONSOLIDATE*</li> <li>● ACCT_IS_CURRENCY*</li> <li>● ACCT_IS_INTERCO</li> <li>● ACCT_IS_SCALE*</li> <li>● ACCT_SUBACCT_TYPE</li> <li>● INTERCOMPANY</li> <li>● JOURNALS</li> </ul>

Hyperion Enterprise SE のクエリ属性	Hyperion Enterprise 相当
<ul style="list-style-type: none"> <li>● JOURNALS</li> <li>● SCALE</li> </ul>	<ul style="list-style-type: none"> <li>● SCALE</li> </ul>
クエリ属性 CODE では文字列が取得されます。	関数 HypQueryEx( ) と EntQueryEx( ) ではコードの記号が取得されず。文字列を取得する場合は、コード表 ID_CODES でコードの NAME をクエリする必要があります。

Hyperion Enterprise ではこれらは 1 バイトしか取得しませんが、wLen = 2 の場合は 2 バイト取得されます。

## HypQryNode( ) のアップグレード

HypQryNode( ) の代わりに HypQueryEx( ) または EntQueryEx( ) を呼び出すことができます。表 52 に、Hyperion Enterprise SE の HypQueryNode( ) に相当する Hyperion Enterprise のクエリ属性を示します。

表 52 HypQryNode( ) クエリ属性のアップグレード

Hyperion Enterprise SE のクエリ属性	Hyperion Enterprise 相当
ACQUIRED	該当しません。
CONSMETHOD	表 ID_NODES、属性 NODEBYPER_CONLOGIC を使用します。
DISPOSED	該当しません。
PCTOWNED	表 ID_NODES、属性 NODEBYPER_PCTOWNED を使用します。
ADDFLAG HIDEDEPS CHARTMETHOD CODE CURRENCY DESCRIPTION ELIMINATIONS INTERCOMPANY JOURNALS NAME NAMEID NAMESIG PARENT SCALE SUB-NAME	HypQueryEx( ) または EntQueryEx( ) を呼び出す場合は、新しいノード表 ID_NODES を使用します。これらのクエリ属性は Entities (ID_NAMES) 表を使用します。最初にクエリ属性 NAMEID を使用してノード表 (ID_NODES) をクエリしてエンティティの記号を取得し、次にそのエンティティ記号を使用して、目的のクエリ属性を使ってエンティティ表 (ID_NAMES) をクエリします。

## HypQryNode( ) クエリサイズのアップグレード

Hyperion Enterprise SE と Hyperion Enterprise では、クエリするフィールドのサイズが異なります。表 53 に、Hyperion Enterprise SE の HypQryNode( ) のクエリサイズと、それに相当する Hyperion Enterprise のクエリサイズを示します。

表 53 HypQryNode( ) のクエリサイズのアップグレード

Hyperion Enterprise SE のクエリサイズ	Hyperion Enterprise のクエリサイズ
<p>以下のクエリ属性では Interger (C 言語では short) の値が取得されます。</p> <ul style="list-style-type: none"> <li>● CHARTMETHOD</li> <li>● CURRENCY</li> </ul>	<p>以下のクエリ属性では Long (C 言語では SIGNA) が取得されます。Hyperion Enterprise SE の場合のように、HypQryNode( ) は値を整数 (C 言語では short) に切り捨てます。Hyperion Enterprise で完全に機能するには、</p>

Hyperion Enterprise SE のクエリサイズ	Hyperion Enterprise のクエリサイズ
<ul style="list-style-type: none"> <li>SUB-NAME</li> </ul>	<p>HypQueryEx( )または EntQueryEx( )を使用する必要があります。</p> <ul style="list-style-type: none"> <li>CHARTMETHOD</li> <li>CURRENCY</li> <li>SUB-NAME</li> </ul>
<p>以下のクエリ属性では Interger (C 言語では short) の値が取得されます。</p> <ul style="list-style-type: none"> <li>ELIMINATIONS*</li> <li>HIDEDEPS*</li> <li>INTERCOMPANY</li> <li>JOURNALS</li> <li>SCALE</li> </ul>	<p>以下のクエリ属性では 1 バイトが取得されます。</p> <ul style="list-style-type: none"> <li>ELIMINATIONS*</li> <li>HIDEDEPS*</li> <li>INTERCOMPANY</li> <li>JOURNALS</li> <li>SCALE</li> </ul>
<p>クエリ属性 CODE では文字列が取得されます。</p>	<p>関数 HypQueryEx( )と EntQueryEx( )ではコードの記号が取得されます。文字列を取得する場合は、コード表 ID_CODES でコードの NAME をクエリする必要があります。</p>

これらは 1 バイトしか取得しませんが、wLen = 2 の場合は 2 バイト取得されます。

## HypQryAcctListEntry( )と HypQryNameListEntry()のアップグレード

HypQryAcctListEntry( )および HypQryNameListEntry( )の呼び出しは HypQueryEx( )または EntQueryEx( )に置き換えることができます。表 ID\_ACCTLISTENTRY または ID\_NAMELISTENTRY を使用します。

Hyperion Enterprise でクエリ属性のサイズが Hyperion Enterprise SE と異なる場合は、HypQryAcctListEntry( )と HypQryNameListEntry( )によってこれらの変更が補正されます。HypQueryEx( )または EntQueryEx( )を使用する場合は、正しい長さの引数を使用する必要があります。

表 54 に、Hyperion Enterprise SE の HypQryAcctListEntry( )および HypQryNameListEntry( )のクエリ属性と、それに相当する Hyperion Enterprise のクエリ属性を示します。

**表 54** HypQryAcctListEntry( )および HypQryNameListEntry( )クエリ属性のアップグレード

Hyperion Enterprise SE のクエリ属性	Hyperion Enterprise 相当
<p>以下のクエリ属性では Interger (C 言語では short) の値が取得されます。</p> <ul style="list-style-type: none"> <li>NUM_CHART</li> </ul>	<p>以下のクエリ属性では Long (C 言語では SIGNA) が取得されます。</p> <ul style="list-style-type: none"> <li>NUM_CHART</li> </ul>
<ul style="list-style-type: none"> <li>NUM_CONSOL</li> </ul>	<ul style="list-style-type: none"> <li>NUM_CONSOL</li> </ul>
<ul style="list-style-type: none"> <li>NUM_CURRENCY</li> </ul>	<ul style="list-style-type: none"> <li>NUM_CURRENCY</li> </ul>
<ul style="list-style-type: none"> <li>NUM_GROUPS</li> </ul>	<ul style="list-style-type: none"> <li>NUM_GROUPS</li> </ul>

Hyperion Enterprise SE のクエリ属性	Hyperion Enterprise 相当
● NUM_NAME	● NUM_NAME
● NUM_ORGS	● NUM_ORGS
● NUM_SUBNAMES	● NUM_SUBNAMES
● NUM_SUBSTRUCTS	● NUM_SUBSTRUCTS
● NUM_TABLES	● NUM_TABLES
● QRYFIRST	● QRYFIRST
● QRYNEXT	● QRYNEXT
<p>以下のクエリ属性では Integer（C 言語では short）の値が取得されます。</p> <ul style="list-style-type: none"> <li>● ALL_DEPS</li> <li>● BASE_NAMES</li> <li>● CALC_ACCT*</li> <li>● ELIM_NAMES</li> <li>● IMMED_DEPS</li> <li>● INPUT_ACCT*</li> <li>● INTCO_NAMES</li> <li>● JOURNAL_NAMES</li> <li>● MULTI_ACCT*</li> <li>● PARENT_NAMES</li> <li>● SHOW_DETAIL*</li> <li>● SINGLE_ACCT*</li> <li>● SUBNAMES</li> <li>● SUB_MULT*</li> <li>● SUB_SINGLE*</li> </ul>	<p>以下のクエリ属性では 1 バイトが取得されます。</p> <ul style="list-style-type: none"> <li>● _DEPS</li> <li>● BASE_NAMES</li> <li>● CALC_ACCT*</li> <li>● ELIM_NAMES</li> <li>● IMMED_DEPS</li> <li>● INPUT_ACCT*</li> <li>● INTCO_NAMES</li> <li>● JOURNAL_NAMES</li> <li>● MULTI_ACCT*</li> <li>● PARENT_NAMES</li> <li>● SHOW_DETAIL*</li> <li>● SINGLE_ACCT*</li> <li>● SUBNAMES</li> <li>● SUB_MULT*</li> <li>● SUB_SINGLE*</li> </ul>
<p>以下のクエリ属性では Integer の配列（C 言語では short の配列）の値が取得されます。</p> <ul style="list-style-type: none"> <li>● CHART_DATA</li> <li>● CONSOL_DATA</li> <li>● ORG_DATA</li> <li>● SUBNAME_DATA</li> <li>● TABLE_DATA</li> <li>● SUBSTRUCT_DATA</li> </ul>	<p>以下のクエリ属性では Long の配列（C 言語では SIGNAs の配列）が取得されます。</p> <ul style="list-style-type: none"> <li>● CHART_DATA</li> <li>● CONSOL_DATA</li> <li>● ORG_DATA</li> <li>● SUBNAME_DATA</li> <li>● TABLE_DATA</li> <li>● SUBSTRUCT_DATA</li> </ul>
表 ID_NAMELISTENTRY のクエリ属性 CODE では、連動エンティティ一覧の条件として指定されたユーザ定義コードが取得されます。	連動エンティティ一覧には複数のコードを指定できます。
表 ID_ACCTLISTENTRY のクエリ属性 LEVELS	該当しません。代わりに、連動勘定科目一覧をコードでフィルタできます。

これらは 1 バイトしか取得しませんが、wLen = 2 の場合は 2 バイト取得されます。

HypQueryEx( )または EntQueryEx( )にアップグレードする場合は、エンティティ一覧に関連付けられたコードを以下のガイドラインに従って取得します。

- クエリ属性 NUM\_CODES のエンティティ一覧表をクエリして、エンティティ一覧に関連付けられたコードの数を取得します。取得した数は long 整数です。
- それだけ多くの記号（Visual Basic では Long、C 言語では SIGNA）の配列を格納できる大きさのバッファを割り当てます。
- クエリ属性 CODE\_DATA のエンティティ一覧表をクエリして、それにバッファのアドレスを渡します。バッファがコード記号の配列でいっぱいになります。
- 新しいコード表 ID\_CODES にコードの NAME をクエリして各コードの ID を取得し、それにコードの記号を渡します。

## 32 ビットの分割勘定科目とエンティティ記号の置き換え

Hyperion Enterprise SE では、32 ビットの勘定科目記号を分割して、主要勘定科目とサブ勘定科目の記号を抽出することができます。同様に、32 ビットのエンティティ記号を分割して、サブエンティティの記号を抽出することができます。

Hyperion Enterprise で主要勘定科目およびサブ勘定科目の記号を取得するには、EntAcctSplit( )を使用します。Hyperion Enterprise でサブエンティティの記号を取得するには、HypQueryEx( )を呼び出して、属性 SUB-NAME のエンティティ表 (ID\_NAME) をクエリします。

## Hyperion Enterprise の表の変更

使用されなくなった関数を新しい Hyp...()または Ent...()関数に置き換える場合は、Hyperion Enterprise の表へのコードの変更を調整する必要があります。以下の表は Hyperion Enterprise SE から変更されています。

- 会社間詳細表 (ID\_INTCODET)
- ノード表 (ID\_NODES)
- 期別替表 (ID\_ROLLOVER)
- セキュリティグループ表 (ID\_SECGRPTAB)
- セキュリティユーザ表 (ID\_NODES)

## 名前が変更された表

表 55 に示した表の表 ID は、名前が変更されています。これらの表の値は変更されています。

表 55 名前が変更された表

表名	Hyperion Enterprise SE	Hyperion Enterprise
サブ勘定科目ヘッダー	ID_SUBACCT	ID_SUBACCTHDR

表名	Hyperion Enterprise SE	Hyperion Enterprise
サブ勘定科目の詳細	ID_SUBACCTDETAIL	ID_SUBACCTDET
レポートセット	ID_SETS	ID_REPORT_SETS
レポートエントリ	ID_REPSETS	ID_REPORT_ENTRIES

## 会社間詳細表の変更

会社間詳細表 ID\_INTCODET は、Hyperion Enterprise では ID\_ICSET と ID\_INTCODET の 2 つの表に分割されました。

新しい表 ID\_ICSET には会社間照合セットが含まれています。この表の各レコードには、全セットに適用するセット名や他の情報が入っています（調整勘定科目など）。会社間詳細表 ID\_INTCODET には各セットの勘定科目ペアが含まれています。

Hyperion Enterprise SE の表にはこの 2 種類のレコードが混ざっています。

## 会社間表の HypEnum( )のアップグレード

HypEnum( )を HypEnumEx( )または EntEnum( )にアップグレードする場合は、表 ID\_INTCODET を使い続ける必要があります。

Hyperion Enterprise SE の場合と同様に、グループヘッダーレコード（現在は表 ID\_ICSET 内）の記号を渡して、そのセットのみの詳細レコードを列挙するか、NONE を渡して全セットの詳細レコードを列挙できます。さらに、表 ID\_ICSET の HypEnumEx( )または EntEnum( )を呼び出して、セットだけを列挙できるようになりました。

## 会社間表の HypLock( )と HypUnLock( )のアップグレード

HypLock( )と HypUnLock( )を HypLockEx( )と HypUnLockEx( )または EntSelect( )、EntSelectAdd( )および EntUnSelect( )に置き換える場合は、表 ID\_ICSET と OR 演算と ID\_ASSOC 定数を使用します。これで両方の表が選択されます。

## 会社間表の HypQuery( )のアップグレード

HypQuery( )を HypQueryEx( )または EntQueryEx( )にアップグレードする場合、取得しようとしている情報は、表 ID\_ICSET のグループヘッダーレコードにある場合と、表 ID\_INTCODET の詳細レコードにある場合があります。表 ID\_ICSET のヘッダーレコード記号を取得するには、HypFindEx( )または EntFind( )を使用できます。詳細レコード記号を取得するには、[308 ページの「会社間表の詳細レコードの取得」](#)を参照してください。

表 56 に、HypQuery()会社間詳細表の Hyperion Enterprise SE のクエリ属性、表 ID および Hyperion Enterprise のクエリ属性を一覧にします。

表 56 HypQuery( )会社間詳細表クエリ属性のアップグレード

Hyperion Enterprise SE のクエリ属性	表 ID	Hyperion Enterprise のクエリ属性
INTCO_GROUP_NAME	ID_ICSET	NAME
INTCO_PLUG	ID_ICSET	INTCO_PLUGSIG
INTCO_ACCT1	ID_INTCODET	INTCO_ACCTSIG1
INTCO_ACCT2	ID_INTCODET	INTCO_ACCTSIG2
QRYFIRST,	任意の表	QRYFIRST
QRYNEXT		QRYNEXT

古い関数は Hyperion Enterprise SE の動作をエミュレートしますが、完全ではありません。古い HypQuery( )関数を使い続ける場合は、古い HypFind( )関数から返された記号を使用して、現在は表 ID\_ICSET にある情報をクエリする必要があります。また、古い HypEnum( )関数で取得した記号（詳細レコード記号）を使用して、現在は表 ID\_INTCODET にある情報をクエリする必要があります。

## 会社間表の詳細レコードの取得

Hyperion Enterprise SE では、会社間詳細レコードは表のグループヘッダーレコード、セットのレコードに従います。Hyperion Enterprise では、ヘッダーレコードと詳細レコードは別の表にあります。

セットの最初の詳細レコードを取得するには、表 ID\_ICSET、セットのグループヘッダーレコード記号およびクエリ属性 INTCO\_GROUPSIG を使用して、HypQueryEx( )または EntQueryEx( )を呼び出します。これにより、表 ID\_INTCODET でセットの最初の詳細レコード記号が取得されます。

セットの次の詳細レコードを取得するには、表 ID\_INTCODET、現在の詳細レコード記号およびクエリ属性 INTCO\_NEXTDET を使用して、HypQueryEx( )または EntQueryEx( )を呼び出します。セットに詳細レコードがなくなった場合は、取得される記号は NONE です。

HypEnumEx( )または EntEnum( )を使用して詳細レコードの記号を取得することができます。

## ノード表の変更

Hyperion Enterprise の組織構造体情報はノードに格納されています。表 57 に、Hyperion Enterprise SE から Hyperion Enterprise へのノード表の変更を一覧にします。

表 57 ノード表の変更

Hyperion Enterprise SE	Hyperion Enterprise
ノード情報は名前表 ID_NAMES に格納されています。	ノード情報は新しい表 ID_NODES に格納されています。
HypFindNameInOrg( )はエンティティの記号を返します。	HypFindNameInOrg( )はノードの記号を返します。
HypEnumOrgNames( )はコールバック関数にエンティティの記号を渡します。	<p>HypEnumOrgNames( )はコールバック関数にノードの記号を渡します。HypQryNode( )はノード記号を予期しているため、コールバック関数は記号を使用して HypQryNode( )を呼び出すことができます。表 ID_NAMES はエンティティ記号ではないため、コールバック関数の記号を使用して表 ID_NAMES の HypQuerySig( )または HypQueryEx( )を呼び出すことはできません。</p> <p>ノード記号から適切なエンティティ記号を取得するには、クエリ属性 NAMEID または NAMESIG を使用してノード表 ID_NODES をクエリします。</p>

## 期別替表の変更

期別替表 ID\_ROLLOVER は ID\_ROLLOVER と ID\_ROLLSET の 2 つの表に分割されました。ID\_ROLLSET 表には、期別替グループ ID や説明など、期別替セットのヘッダー情報が含まれています。ID\_ROLLOVER には各期別替セットの詳細が含まれています。各レコードには配賦元と配賦先カテゴリが含まれています。

### HypLock( )と HypUnLock( )

HypLock( )と HypUnLock( )を HypLockEx( )と HypUnLockEx( )または EntSelect( )、EntSelectAdd( )および EntUnSelect( )にアップグレード場合は、表 ID\_ROLLSET と OR 演算を ID\_ASSOC 定数と一緒に使用します。これで両方の表が選択されます。

### HypEnum( )

HypEnum( )を HypEnumEx( )または EntEnum( )にアップグレードする場合は、表 ID\_ROLLOVER を使い続ける必要があります。

Hyperion Enterprise SE の場合と同様に、グループヘッダーレコード（現在は表 ID\_ROLLSET 内）の記号を渡して、その期別替セットの詳細レコードのみを列挙するか、NONE を渡して全期別替セットの詳細レコードを列挙できます。

さらに、ID\_ROLLSET 表の HypEnumEx( )または EntEnum( )を呼び出して、期別替セットだけを列挙できるようになりました。

### HypFind( )

HypFind( )の呼び出しを HypFindEx( )または Entfind( )に置き換えてグループ ID に関連付けられたレコードを見つける場合は、ID\_ROLLOVER ではなく表 ID\_ROLLSET を使用します。期別替 ID は現在は表 ID\_ROLLSET にあります。

## HypQuery( )

HypQuery( )を HypQueryEx( )または EntQuery()にアップグレードする場合、取得しようとしている情報は、表 ID\_ROLLSET のグループヘッダーレコードにある場合と、表 ID\_ROLLOVER の詳細レコードにある場合があります。

表 ID\_ROLLSET のヘッダーレコード記号を取得するには、HypFindEx()または EntFind()を使用できます。詳細レコード記号を取得する方法については、[311 ページの「期別替表の詳細レコードの取得」](#)を参照してください。

**表 58** HypQuery( )期別替表クエリ属性のアップグレード

Hyperion Enterprise SE のクエリ属性	表 ID	Hyperion Enterprise のクエリ属性
ROLLOVER_CAT1	ID_ROLLOVER	ROLL_SRCCAT
ROLLOVER_CAT2	ID_ROLLOVER	ROLL_DESTCAT
ROLLOVER_NAME	ID_ROLLSET	NAME
QRYFIRST QRYNEXT	任意の表	QRYFIRST QRYNEXT

古い関数は Hyperion Enterprise SE の動作をエミュレートしますが、完全ではありません。古い HypQuery()関数を使い続ける場合は、古い HypFind( )関数から返された記号（表 ID\_ROLLSET のグループヘッダーレコード記号）を使用して、現在は表 ID\_ROLLSET にある情報をクエリする必要があります。古い HypEnum( )関数で取得した記号（詳細レコード記号）を使用して、現在は表 ID\_ROLLOVER にある情報をクエリする必要があります。

**表 59** HypQuery( )期別替表クエリサイズのアップグレード

リリース 1.8 または 4SE	リリース 4
これらのクエリ属性では Integer（C 言語では short）の値が取得されます。	これらのクエリ属性では、現在は Long（C 言語では SIGNA）の値が取得されます。
ROLLOVER_CAT1 (または ROLL_SRCCAT) ROLLOVER_CAT2 (または ROLL_DESTCAT) QRYFIRST QRYNEXT	

Hyperion Enterprise SE では、以下のクエリ属性で Integer（C 言語では short）の値が取得されます。Hyperion Enterprise では、これらのクエリ属性で、現在は Long（C 言語では SIGNA）の値が取得されます。

- ROLLOVER\_CAT1（または ROLL\_SRCCAT）
- ROLLOVER\_CAT2（または ROLL\_DESTCAT）
- QRYFIRST

- QRYNEXT

## 期別替表の詳細レコードの取得

Hyperion Enterprise SE では、期別替詳細レコードは期別替表のグループヘッダーレコード（期別替セット名のレコード）に従います。Hyperion Enterprise では、ヘッダーレコードと詳細レコードは別の表にあります。

期別替セットの最初の詳細レコードを取得するには、表 ID\_ROLLSET、期別替セットのグループヘッダーレコード記号およびクエリ属性 ROLL\_GROUPSIG を使用して、HypQueryEx()または EntQueryEx()を呼び出します。これにより、表 ID\_ROLLOVER でセットの最初の詳細レコード記号が取得されます。

期別替セットの次の詳細レコードを取得するには、表 ID\_ROLLOVER、現在の詳細レコード記号およびクエリ属性 ROLL\_NEXTDET を使用して、HypQueryEx()または EntQueryEx()を呼び出します。セットに詳細レコードがなくなった場合は、取得される記号は NONE です。

## 使用されなくなったセキュリティ表

表 60 に、Hyperion Enterprise で使用されなくなったセキュリティ表を示します。以下のセキュリティ表は Hyperion Enterprise にはありません。

表 60 使用されなくなったセキュリティ表

表 ID	説明
ID_SECCATTAB	種別アクセスグループ
ID_SECCATDET	種別アクセス詳細
ID_SECNAMETAB	エンティティアksesグループ
ID_SECNAMEDET	エンティティアkses詳細
ID_SECHYPTAB	Hyperion Enterprise タスクアクセスグループ
ID_SECHYPDET	Hyperion Enterprise タスクアクセス詳細

## 使用されなくなったセキュリティアクセスグループ表

使用されなくなった HypEnum()、HypFind()、および HypQuery()関数は、使用されなくなったセキュリティアクセスグループ表（ID\_SECCATTAB、ID\_SECNAMETAB および ID\_SECHYPTAB）を Hyperion Enterprise SE のセキュリティグループ表（ID\_SECGRPTAB）と同様に扱います。これらの表のグループの記号は、現在はユーザ表 ID\_SECUSERTAB のグループの記号です。詳しくは、315 ページの「セキュリティグループ表の変更」を参照してください。

これらのテーブルの COUNT クエリ属性を使用する場合は、グループの番号を取得します。ただし、表 ID\_SECUSERTAB を使用して、HypQueryEx()または EntQueryEx()を使用するようにアップグレードした場合は、代わりに COUNT がユーザ表内のグループとユーザの総数を取得します。セキュリティユーザ表

(ID\_SECUSERTAB) のグループ数のみを取得するには、新しいクエリ属性 SECUSER\_NUM\_GROUPS を使用する必要があります。

## 使用されなくなったセキュリティアクセス詳細表

セキュリティアクセス詳細表 (ID\_SECCATDET、ID\_SECNAMEDET、および ID\_SECHYPDET) は Hyperion Enterprise にはありません。Hyperion Enterprise では、種別、エンティティ、タスクだけではなく、大半の項目を指定できます。

これらの古い表はほとんどの場合、指定した種別、エンティティ、またはタスクに対する現在のユーザの権限を問い合わせる場合に使用されます。この場合は、HypQuery( )に渡されるアクセスグループの記号は-1 または NONE です。

種別またはエンティティに対する現在のユーザの権限を取得するには、HypQueryEx( )または EntQueryEx( )を呼び出し、クエリ属性 USER\_RIGHTS を使用して種別表またはエンティティ表 (ID\_CATEGORY または ID\_NAMES) をクエリします。戻り値のバッファは整数 (C 言語では short\*) であることが必要です。

タスクに対する現在のユーザの権限を取得する最も簡単な方法は、HYPACC\_...タスクのアクセスコードを、これに相当する SECTASK\_...セキュリティタスクのコードに変更してから、EntGetRightsToTask(hApp&, SECTASK\_..., ...)を呼び出します。これにより、セキュリティ権限 (SECURITY\_MODIFY、SECURITY\_VIEW、SECURITY\_NONE、または新しい SECURITY\_RESTRICTED) が返されます。なお、hApp&は、高レベル関数に使用される Hyperion Enterprise アプリケーションハンドルであり、HypConstruct()の Hyperion Enterprise Retrieve アプリケーションハンドルではありません。使用する正しい hApp&値を取得するには、HypGethApp( )を呼び出す必要があります。

古い HypLock( )および HypUnLock( )関数は、存在しなくなったセキュリティアクセス詳細表の選択や選択解除は行いません。ただし、レガシープログラムがエラー戻りコードをチェックした場合、Hyperion Enterprise SE で表を選択するのに必要なセキュリティ権限が現在のユーザにあれば、HypLock( )は APILOCK\_READONLY を返します。

## セキュリティタスクのコード

表 61 に、使用されなくなった表 ID\_SECHYPDET を使用して特定のタスクに対するユーザ権限のチェックに Hyperion Enterprise SE で使用された古い定数と、それに相当する Hyperion Enterprise のセキュリティタスクの定数を一覧にします。

表 61 セキュリティタスクコードのアップグレード

Hyperion Enterprise SE のセキュリティタスク	Hyperion Enterprise のセキュリティタスク
HYPACC_ACCOUNTS	SECTASK_CHARTOFACCTSMODULE
HYPACC_APPLICATION	SECTASK_APPLICATIONMODULE
HYPACC_BOOKS	SECTASK_BOOKSMODULE
HYPACC_CATEGORY	SECTASK_CATEGORYMODULE

Hyperion Enterprise SE のセキュリティタスク	Hyperion Enterprise のセキュリティタスク
HYPACC_CONSOLIDATION	SECTASK_CONSOLMODULE
HYPACC_DATABASE	SECTASK_DATABASEMODULE
HYPACC_JOURNALS	SECTASK_JOURNALSMODULE
HYPACC_LOGIC	SECTASK_FORMULASMODULE
HYPACC_MAIL	該当しません。
HYPACC_NAMES	SECTASK_ENTITIESMODULE
HYPACC_REPORTS	SECTASK_REPORTSMODULE
HYPACC_ROLLOVERS	SECTASK_ROLLOVERSETADMIN
HYPACC_SCHEDULES	SECTASK_SCHEDULESMODULE
HYPACC_SECURITY	SECTASK_SECURITY
HYPACC_SYSTEMS	該当しません。
HYPACC_COA	SECTASK_CHARTOFACCTSSETUP
HYPACC_SUBACCTAB	SECTASK_SUBACCTBLADMIN
HYPACC_ACCTCVTTAB	SECTASK_ACCTCNVADMIN
HYPACC_ACCTLIST	SECTASK_ACCTLISTADMIN
HYPACC_INTERCOMATCH	SECTASK_INTERCOADMIN
HYPACC_NEWAPPS	SECTASK_NEWAPPS
HYPACC_EDITAPPS	SECTASK_APPLICATIONADMIN
HYPACC_BOOKEDITOR	SECTASK_REPORTSCRIPTEDIT
HYPACC_BOOKCOMPILE	SECTASK_REPORTSCRIPTEDIT
HYPACC_BOOKRUN	SECTASK_RUNBOOKS
HYPACC_RUNINTERCO	SECTASK_RUNINTERCO
HYPACC_DATAENTRY	SECTASK_SCHEDDATAENTRY
HYPACC_LOADDATA	SECTASK_LOADDATA
HYPACC_EXTRACTDATA	SECTASK_EXTRACTDATA
HYPACC_UNLOCKDATA	SECTASK_UNLOCKDATA
HYPACC_DEFINEFORMAT	SECTASK_DATABASEADMIN
HYPACC_JOURCREATE	SECTASK_JOURADMIN
HYPACC_JOURPOST	SECTASK_POSTJOUR

Hyperion Enterprise SE のセキュリティタスク	Hyperion Enterprise のセキュリティタスク
HYPACC_JOUREDITPOST	SECTASK_EDITJOUR
HYPACC_JOURREVERSE	SECTASK_REVERSEJOUR
HYPACC_JOURUNPOST	SECTASK_UNPOSTJOUR
HYPACC_JOURPRINT	SECTASK_JOURREPORTS
HYPACC_LOGICADM	SECTASK_METHODADMIN
HYPACC_MAILADM	該当しません。
HYPACC_NORGS	SECTASK_ORGADMIN
HYPACC_NSUBSTRUCT	SECTASK_SUBSTRADMIN
HYPACC_NCURRENCIES	SECTASK_CURRENCYADMIN
HYPACC_NCONVTAB	SECTASK_ENTITYCNVADMIN
HYPACC_NAMELIST	SECTASK_ENTITYLISTADMIN
HYPACC_RSETMANAGER	SECTASK_REPORTADMIN
HYPACC_RIMPORT	SECTASK_REPORTSCRIPTEDIT
HYPACC_REDITOR	SECTASK_REPORTSCRIPTEDIT
HYPACC_RCOMPILE	SECTASK_REPORTSCRIPTEDIT
HYPACC_RRUN	SECTASK_RUNREPORTS
HYPACC_ROLLRUN	SECTASK_ROLLRUN
HYPACC_SDATAENTRY	SECTASK_SCHEDDATAENTRY
HYPACC_SDEFINESCHED	SECTASK_SCHEDADMIN
HYPACC_SECURITYADM	SECTASK_SECURITY
HYPACC_SYSLOAD	SECTASK_LOADAPP
HYPACC_SYSEXTRACT	SECTASK_EXTRACTAPP
HYPACC_BOOKDEFINE	SECTASK_REPORTSCRIPTEDIT
HYPACC_DEFROLLOVER	SECTASK_ROLLOVERSETADMIN
HYPACC_EDITDATA	SECTASK_SCHEDDATAENTRY
HYPACC_LOCKDATA	SECTASK_LOCKDATA
HYPACC_OPEN	SECTASK_OPENPERIOD
HYPACC_DEFREPORTS	SECTASK_REPORTSCRIPTEDIT
HYPACC_CATADM	SECTASK_CATADMIN

Hyperion Enterprise SE のセキュリティタスク	Hyperion Enterprise のセキュリティタスク
HYPACC_CONSOL	SECTASK_CONSOL_ALL
HYPACC_EXTRACTOLOCK	SECTASK_EXTRACTDATA
HYPACC_JOURREVIEW	SECTASK_REVIEWJOUR
HYPACC_JOURALLOWSS	SECTASK_UNBALANCEDJOUR
HYPACC_JOURALLOWUBWN	SECTASK_BALANCEINENTITY
HYPACC_DATAJOURLOCK	SECTASK_DATAJOURLOCK
HYPACC_DATAJOURUNLOCK	SECTASK_DATAJOURUNLOCK
HYPACC_GRW	SECTASK_REPORTSCRIPTEDIT

## セキュリティグループ表の変更

表 62 に、Hyperion Enterprise SE から Hyperion Enterprise へのセキュリティグループ表（ID\_SECGRPTAB）の変更を一覧にします。

表 62 セキュリティグループ表の変更

Hyperion Enterprise SE	Hyperion Enterprise
セキュリティグループはセキュリティグループ表 ID_SECGRPTAB にあります。	セキュリティグループはセキュリティユーザ表 ID_SECUSERTAB にあります。  注： HypEnum( )、HypFind( )および HypQuery( )をアップグレードする場合は、ID_SECGRPTAB の代わりに ID_SECUSERTAB を使用します。
各セキュリティグループには、種別アクセス、名前アクセスおよび Hyperion Enterprise タスクアクセスグループが割り当てられます。クエリ属性 GRPTAB_NAME、GRPTAB_CAT および GRPTAB_HYPE がこれらのグループを取得します。	アクセスグループとクエリ属性 GRPTAB_CAT、GRPTAB_NAME および GRPTAB_HYPE は Hyperion Enterprise にはありません。
HypEnum( )はグループのみをエミュレートします。	HypEnumEx( )は使用できません。代わりに、表 ID_SECUSERTAB で EntEnum( )を呼び出してください。

## セキュリティユーザ表の変更

表 63 に、Hyperion Enterprise SE から Hyperion Enterprise へのセキュリティユーザ表の変更を一覧にします。

**表 63** セキュリティユーザ表の変更

Hyperion Enterprise SE	Hyperion Enterprise
セキュリティユーザ表 ID_SECUSERTAB にはユーザのみが含まれています。	ユーザ表にはユーザとセキュリティグループの両方が含まれています。
HypEnum( )はユーザのみをエミュレートします。	HypEnumEx( )は使用できません。apiStruct で EntEnum( )を呼び出してください。
パスワードとユーザ ID を取得するには、HypQuery( )をクエリ属性 USERTAB_PASSWORD および USERTAB_USERID と一緒に使用します。	パスワードを取得するには SECUSER_PASSWORD、ユーザ ID を取得するには SHORTNAME を使用します。
USERTAB_GROUP クエリはユーザのセキュリティグループを返します。	Hyperion Enterprise ではユーザが複数のグループに属することができるため、クエリ属性 USERTAB_GROUP はありません。ユーザが属するグループをエミュレートするには、セキュリティグループ表 ID_SECGRPTAB を使用します。
COUNT クエリ属性は表内のユーザ数を取得します。	COUNT はセキュリティユーザ表のユーザとセキュリティグループの総数を取得します。ユーザの数を取得するには、新しいクエリ属性 SECUSER_NUM_GROUPS を使用して表内のグループ数を取得した後、COUNT から取得した合計からこの数を引きます。

## この章の内容

コールバック関数の概要.....	317
コールバック関数の構文.....	318

この章では、列挙関数に必要なコールバック関数について説明します。主な項目は以下のとおりです。

## コールバック関数の概要

データ列挙関数はすべて、ユーザ提供のコールバック関数を必要とします。列挙関数は、列挙する各項目についてコールバック関数を呼び出します。列挙する各項目についてコールバック関数を使用して処理します。

さらに、時間がかかりそうな処理の間、高レベル関数がコールバック関数を使用してステータス情報を提供する場合もあります。

以下は、コールバックを使用する `Hyp...()` データ列挙関数と、スプレッドシート関数、アドイン関数、または表関数に使用すべきコールバック関数です。

- CALLBACK12
- CALLBACKAPP（または CALLBACK6）
- CALLBACKSTR
- DWCALLBACK

以下は、`Ent...()` データ列挙関数と、コールバックを使用するその他の高レベル関数に使用すべきコールバック関数です。

- CALLBACKAPI
- CALLBACKAPP（または CALLBACK6）
- CALLBACKCREATE
- CALLBACKDBENUM
- CALLBACKDBLOAD
- CALLBACKJOUREXTRACT
- CALLBACKJOURLOAD
- CALLBACKLOGIC
- CALLBACKSECLOAD

- CALLBACKSEL
- CALLBACKSHARES
- CALLBACKUSERS
- CALLBACKVB

ほとんどのコールバック関数は通常は 0（ゼロ）を返すはずですが、ゼロ以外の値を返すと、列挙やその他の処理を破棄できます。ゼロ以外の値は戻りコードとして API 関数から渡されます。

## コールバック関数の構文

以下のトピックでは、個々のコールバック関数の構文を提供します。

### CALLBACK12

このコールバック関数は、勘定科目一覧のエン트리またはエンティティ一覧のエントリに使用されます。

関連する関数：

[85 ページの「HypEnumAcctListEntriesEx\( \) - 勘定科目一覧の勘定科目の列挙」](#)

[90 ページの「HypEnumNameListEntriesEx\( \) - エンティティ一覧内のエンティティの列挙」](#)

次の形式を使用します。

```
Public Function MyFunc(ByVal hRApp As Integer, ByVal sigList As Long, ByVal sigEntry
As Long, ByVal lParam As Long) As Integer
```

#### 変数 説明

MyFunc コールバック関数の名前

hRApp Retrieve アプリケーションハンドル

sigList 勘定科目一覧の記号またはエンティティ一覧の記号

sigEntry HypEnumAcctListEntriesEx( ) の場合：一覧が固定リストの場合は勘定科目一覧のエントリの記号、一覧が動的リストの場合は勘定科目の記号。

HypEnumNameListEntriesEx( ) の場合：エンティティ一覧のエントリの記号。これはエンティティの記号ではありません。クエリ属性 ENTRY\_SIG で HypQueryEx( ) を呼び出して、エンティティ一覧のエントリ表 (ID\_NAMELISTENTRY) からこのエントリのエンティティ記号を取得します。

lParam lParam 引数で渡されるユーザ定義のデータ

戻りコード：

コード	説明
0	処理を継続します。

コード	説明
ゼロ以外	処理をキャンセルします。

C 言語では次の形式を使用します。

```
short WINAPI MyFunc(HRETRIEVEAPP hRApp, SIGNA sigList, SIGNA sigEntry,
LONG lParam)
```

## CALLBACKAPI

このコールバック関数は列挙とステータス情報に使用されます。

関連する関数：

[143 ページの「EntConsolidate\(\) - 連結」](#)、[171 ページの「EntEnum\(\) - 表内のレコードの列挙」](#)、[183 ページの「EntEnumOrgEntities\(\) - 組織内のノードの列挙」](#)、[184 ページの「EntEnumSubAcctSig\(\) - サブ勘定科目の列挙」](#)、[250 ページの「EntRunRollover\(\) - 期別替の実行」](#)

次の形式を使用します。

```
Public Function MyFunc(ByVal hSelect As Long, ByVal sigRecd As Long, ByVal sigKey As Long, ByVal lParam As Long, apiS As APISTRUCT) as Integer
```

### 変数 説明

MyFunc コールバック関数の ID

hSelect 表選択ハンドル

sigRecd レコード記号

sigKey NONE、または関連する表のキー。詳しくは、[340 ページの「関連する表」](#)を参照してください。

lParam ユーザ定義データ

apiS apiStruct 構造体

戻りコード：

コード	説明
0	処理を継続します。
ゼロ以外	処理をキャンセルします。

C 言語では次の形式を使用します。

```
Short WINAPI MyFunc(HSELECT hSelect, SIGNA sigRecd, SIGNA sigKey, Long lParam,
LPAPISTRUCT pApiS)
```

## CALLBACKAPP（または CALLBACK6）

このコールバック関数は、EntEnumApplications()またはHypEnumApplications()で列挙される各アプリケーションについて呼び出されます。

次の形式を使用します。

**Public Function MyFunc(ByVal lAddr As Long, ByVal lParam As Long) As Integer**

### 変数 説明

MyFunc コールバック関数の ID

lAddr アプリケーション名である C スタイル文字列のアドレス

lParam ユーザ定義データ

戻りコード：

コード	説明
0	処理を継続します。
ゼロ以外	処理をキャンセルします。

C 言語では次の形式を使用します。

**short WINAPI MyFunc(LPSTR szApp, LONG lParam)**

## CALLBACKCREATE

このコールバック関数は、新しいアプリケーションのアプリケーションデフォルトを設定するために、EntCreateApplication()によって適宜呼び出されます。

EntCreateApplication()はこのコールバック関数に新しいアプリケーションのアプリケーションハンドル (hApp) を渡すので、EntUpdateDefault()を呼び出してアプリケーションのデフォルトを設定できます。アプリケーションは完全に作成や初期化がされているわけではないので、コールバック関数内でこの hApp を使用してその他の操作をしないでください。

表 ID\_APPDEFAULT を使用して EntUpdateDefault()を呼び出します。この表のクエリ属性については、[352 ページの「ID\\_APPDEFAULT（アプリケーションデフォルト表）クエリ属性」](#)を参照してください。

コールバック関数で属性を設定するには、Hyperion Enterprise の新規アプリケーションウィザードを実行して、必要な情報を確認します。ウィザードが要求する情報をすべて設定する必要があります。圧縮ファイルオプションなど、ウィザードが要求していない情報は設定しないでください。Hyperion Enterprise では、アプリケーションの作成時にコールバック関数を使用して以下の属性が設定されるので、これに従うことをお勧めします。

APP\_EXPECTED\_ACCTS APP\_EXPECTED\_ENTITIES APP\_EXPECTED\_CATS  
APP\_EXPECTED\_REPORTS APP\_DESC APP\_DEVERASELOG APP\_IS\_ORGBYPER  
APP\_JOUR\_AUTONUMBER APP\_STORECONDETAIL APP\_STORETRANDETAIL  
APP\_USETURBO APP\_JOUR\_TOPLEVEL APP\_IMPACTFUTURECAT

APP\_SUBACCTSIG APP\_INPUT\_PCT APP\_CONSOL\_IS\_PER  
APP\_USE\_CHILD\_RATES

ファイルベースのアプリケーションの場合は、Hyperion Enterprise のコールバック関数で以下の属性も設定されます。

APP\_PATH APP\_DATADIR APP\_INBOXDIR APP\_OUTBOXDIR APP\_REPORTDIR

SQL アプリケーションの場合は、Hyperion Enterprise のコールバック関数で以下の属性も設定されます。

APP\_PATH APP\_INBOXDIR APP\_OUTBOXDIR APP\_REPORTDIR APP\_SQL\_SERVER  
APP\_SQL\_DBNAME APP\_ORA\_LRG\_TBLSPACE APP\_ORA\_IDX\_TBLSPACE  
APP\_ORA\_TMP\_TBLSPACE コールバック関数からの戻り値では、  
EntCreateApplication()は次の属性を設定することを予期しています。

APP\_PATH APP\_USETURBO APP\_IS\_ORGBYPER

次の形式を使用します。

Public Function MyFunc(ByVal hAppl As Long, ByVal lParam As Long) As Long

**変数 説明**

MyFunc コールバック関数の ID

hAppl アプリケーションハンドル

lParam ユーザ定義データ

戻りコード：

コード	説明
1	True。処理を継続します。
0	False。処理をキャンセルします。

C 言語では次の形式を使用します。

BOOL WINAPI MyFunc(HAPP hAppl, LPARAM lParam)

## CALLBACKBENUM

コールバック関数は EntDataExtract( )に渡されて、勘定科目が列挙されます。勘定科目列挙関数は EntDataExtract( )によって数回呼び出されます。

ユーザが lKey 引数を入力します。EntDataExtract( )は専用の fnCallback と lParam を提供します。

C 言語では次の形式を使用します。

SHORT WINAPI MyFunc (LPARAM lKey, CALLBACKAPI fnCallback, LPARAM  
lParam);

変数	説明
----	----

MyFunc	コールバック関数の ID
IKey	ユーザの引数 (EntDataExtract( )に渡された pS->IParamEnumAcct)
fnCallback	各勘定科目について呼び出す引数
IParam	fnCallback 関数に渡す引数

列挙関数は、以下のように、抽出される勘定科目ごとに fnCallback を呼び出す必要があります。

```
wRet = (*fnCallback)(hSelect, sigAcct, (SIGNA) NONE, IParam, NULL);
```

勘定科目の記号である hSelect と IParam を渡す必要があります。2 次キー (NONE) と apiStruct (NULL) は EntDataExtract( )の fnCallback 関数では使用されません。fnCallback 関数は、成功の場合は 0 (ゼロ)、列挙を破棄すべきであればゼロ以外を返します。

戻りコード：

コード	説明
0	成功
ゼロ以外	処理をキャンセルします。まだ実装されていません。

## CALLBACKDBLOAD - EntAppExtract( ) の場合

このコールバック関数はアプリケーション抽出中、ステータス情報に使用されます。詳しくは、[130 ページの「EntAppExtract\( \) - アプリケーションの抽出」](#)および [132 ページの「EntAppExtractVB\( \) - アプリケーションの抽出」](#)を参照してください。

次の形式を使用します。

```
Public Function MyFunc(ByVal addressofFileNameCString As Long, ByVal  
addressofMessageCString As Long, ByVal szEntity As Long, ByVal lStart As Long, ByVal  
lEnd As Long, ByVal lTotal AsLong, ByVal lCurrent As Long, ByVal lParam As Long) As  
Integer
```

変数	説明
----	----

MyFunc	コールバック関数の ID
addressofFileNameCString	抽出ファイルの名前
addressofMessageCString	ステータステキスト
szEntity	未使用
lStart	未使用
lEnd	未使用

変数	説明
lTotal	未使用
lCurrent	未使用
lParam	コールバック関数に入力した引数

戻りコード：

コード	説明
0	処理を継続します。
ゼロ以外	処理をキャンセルします。(まだ実装されていません)

C 言語では次の形式を使用します。

```
short WINAPI MyFunc (LPCSTR szFileName, LPCSTR addressofMessageCString,
LPCSTR szEntity, long lStart, long lEnd, long lTotal, long lCurrent, long lParam);
```

## CALLBACKDBLOAD - EntAppLoad( )の場合

このコールバック関数は、アプリケーションの読み込み中、ステータス情報に使用されます。詳しくは、[134 ページの「EntAppLoad\( \) - アプリケーションの読み込み」](#) および [135 ページの「EntAppLoadVB\( \) - アプリケーションの読み込み」](#) を参照してください。

次の形式を使用します。

```
Public Function MyFunc (ByVal addressofFileNameCString As Long, ByVal
addressofMessageCString As Long, ByVal szEntity As Long, ByVal lStart As Long, ByVal
lEnd As Long, ByVal lTotal As Long, ByVal lCurrent As Long, ByVal lParam As Long) As
Integer;
```

変数	説明
MyFunc	コールバック関数の ID
addressofFileNameCString	読み込みファイルの名前
addressofMessageCString	ステータステキスト
szEntity	未使用
lStart	未使用
lEnd	未使用
lTotal	読み込みファイルのサイズ (バイト単位)
lCurrent	読み込みファイルの現在の位置
lParam	コールバック関数に入力した引数

戻りコード：

コード	説明
0	処理を継続します。
ゼロ以外	処理をキャンセルします。(まだ実装されていません)

C 言語では次の形式を使用します。

**short** WINAPI MyFunc (LPCSTR szFileName, LPCSTR szMsg, LPCSTR szEntity, long lStart, long lEnd, long lTotal, long lCurrent, long lParam)

## CALLBACKDBLOAD - EntDataExtract()の場合

このコールバック関数はデータの抽出中、ステータス情報に使用されます。詳しくは、[149 ページの「EntDataExtract\(\) - データの抽出」](#)を参照してください。

C 言語では次の形式を使用します。

**short** WINAPI MyFunc (LPCSTR szFileName, LPCSTR szCategory, LPCSTR szEntity, long lStart, long lEnd, long lTotal, long lCurrent, long lParam);

変数	説明
MyFunc	コールバック関数の ID
szFileName	抽出ファイルの名前
szCategory	抽出する種別
szEntity	抽出するエンティティ
lStart	最初の期間 (1 から始まる)
lEnd	最後の期間 (1 から始まる)
lTotal	抽出を選択した勘定科目の合計数
lCurrent	抽出された勘定科目の数
lParam	コールバック関数に入力した引数

戻りコード：

コード	説明
0	処理を継続します。
True	処理をキャンセルします。

## CALLBACKDBLOAD - EntUNCDataLoad()の場合

このコールバック関数はデータの読み込み中、ステータス情報に使用されます。詳しくは、[270 ページの「EntUNCDataLoad\(\) - データの読み込み」](#)を参照してください。

C 言語では次の形式を使用します。

**short** WINAPI MyFunc (LPCSTR szFileName, LPCSTR szCategory, LPCSTR szEntity,  
**long** lStart, **long** lEnd, **long** lTotal, **long** lCur, **long** lParam);

変数	説明
MyFunc	コールバック関数の ID
szFileName	読み込みファイルの名前
szCategory	読み込む種別
szEntity	読み込むエンティティ
lStart	最初の期間（1 から始まる）
lEnd	最後の期間（1 から始まる）
lTotal	読み込みファイルの合計行数
lCur	これまでに処理された行数
lParam	コールバック関数に入力した引数

## CALLBACKJOUREXTRACT

このコールバック関数は仕訳の抽出中、ステータス情報に使用されます。詳しくは、[213 ページの「EntJournalExtract\(\) - 仕訳帳の抽出」](#) および [216 ページの「EntJournalExtractVB\(\) - 仕訳帳の抽出」](#) を参照してください。

次の形式を使用します。

**Public Function** MyFunc(**ByVal** addressofFileNameCString **As Long**, **ByVal** addressofNameCString **As Long**, lTotal **As Long**, lCurrent **As Long**, lParam **As Long**) **As Integer**

変数	説明
MyFunc	コールバック関数の名前
addressofFileNameCString	抽出ファイルの名前
addressofNameCString	抽出する仕訳の名前
lTotal	未使用
lCurrent	未使用
lParam	コールバック関数に入力した引数

戻りコード：

コード	説明
0	処理を継続します。

コード	説明
TRUE	処理をキャンセルします。

C 言語では次の形式を使用します。

**short** WINAPI MyFunc (LPCSTR szFileName, LPCSTR szName, **long** lTotal, **long** lCurrent, **long** lParam)

## CALLBACKJOURLOAD

このコールバック関数は仕訳の読み込み中、ステータス情報に使用されます。詳しくは、[220 ページの「EntJournalLoad\( \) - 仕訳帳の読み込み」](#) および [222 ページの「EntJournalLoad\( \) - 仕訳帳の読み込み」](#) を参照してください。

次の形式を使用します。

**Public Function** MyFunc(**ByVal** addressofFileNameCString **As Long**, **ByVal** addressofNameCString **As Long**, lTotal **As Long**, lCurrent **As Long**, lParam **As Long**) **As Integer**

変数	説明
MyFunc	コールバック関数の名前
addressofFileNameCString	読み込みファイルの名前
addressofNameCString	読み込む仕訳の名前
lTotal	読み込みファイルのサイズ。この引数は、メーターバーコントロールの位置の計算に使用されます。
lCurrent	読み込みファイルの現在の位置。この引数は、メーターバーコントロールの位置の計算に使用されます。
lParam	コールバック関数に入力した引数

戻りコード：

コード	説明
0	処理を継続します。
TRUE	処理をキャンセルします。

C 言語では次の形式を使用します。

**short** WINAPI MyFunc (LPCSTR szFileName, LPCSTR szName, **long** lTotal, **long** lCurrent, **long** lParam)

## CALLBACKLOGIC

このコールバック関数は、ロジックのエクスポート時とインポート時のステータス情報に使用されます。ロジックエクスポートについては、[228 ページの](#)

「[EntLogicExport\( \) - 計算式のエクスポート](#)」を参照してください。ロジックインポートについては、[233 ページの「EntLogicImport\( \) - 計算式のインポート](#)」を参照してください。

次の形式を使用します。

```
Public Function MyFunc(ByVal addressOfFileNCString As Long, ByVal  
addressOfMethodCString As Long, ByVal lTotal As Long, ByVal lCurrent As Long, ByVal  
lParam As Long) As Integer
```

変数	説明
MyFunc	コールバック関数の名前
addressOfFileNCString	インポートまたはエクスポートするファイルの名前
addressOfMethodCString	インポートまたはエクスポートするメソッドの名前
lTotal	合計ファイル数。ロジックのエクスポートには使用されません。この引数は、メーターバーコントロールの位置の計算に使用されます。
lCurrent	これまでにインポートされたファイル数。ロジックのエクスポートには使用されません。この引数は、メーターバーコントロールの位置の計算に使用されます。
lParam	コールバック関数に入力した引数

戻りコード：

コード	説明
0	処理を継続します。
TRUE	処理をキャンセルします。

C 言語では次の形式を使用します。

```
short WINAPI MyFunc (LPCSTR szFileName, LPCSTR szMethod, long lTotal, long  
lCurrent, long lParam)
```

## CALLBACKSECLOAD

このコールバック関数は、セキュリティの読み込みまたは抽出時のステータス情報に使用されます。セキュリティ抽出については、[252 ページの「EntSecurityExtract\( \) - セキュリティの抽出](#)」を参照してください。セキュリティ読み込みについては、[255 ページの「EntSecurityLoad\( \) - セキュリティの読み込み](#)」を参照してください。

次の形式を使用します。

```
Public Function MyFunc(ByVal addressOfFileNCString As Long, ByVal  
addressOfTaskCString As Long, ByVal lCurrent As Long, ByVal lStart As Long, ByVal lEnd  
As Long, ByVal lParam As Long) As Integer
```

変数	説明
MyFunc	コールバック関数の名前
addressOfFileNameCString	読み込みまたは抽出するファイルの名前
addressOfTaskCString	処理するタスク
lCurrent	現在の位置。読み込みの場合、これは読み込みファイルの位置です。抽出の場合、これはタスク番号です。
lStart	読み込みまたは抽出ファイルの開始位置
lEnd	読み込みファイルの合計サイズ、または抽出するタスクの合計数
lParam	コールバック関数に入力した引数

抽出の場合は、タスクの数は選択したオプションの数です（SECLOADSTRUCT の bSecClass、bAcesRight、bUserAndGroup、bTask フィールド）。

戻りコード：

コード	説明
0	処理を継続します。
TRUE	処理をキャンセルします。

C 言語では次の形式を使用します。

**short** WINAPI MyFunc (LPCSTR szFileName, LPCSTR szTask, **long** lCurrent, **long** lStart, **long** lEnd, **long** lParam)

## CALLBACKSEL

このコールバック関数は、選択した表を変更する場合に呼び出されます。

関連する関数：

[134 ページの「EntAppLoad\( \) - アプリケーションの読み込み」](#)、[135 ページの「EntAppLoadVB\( \) - アプリケーションの読み込み」](#)、[130 ページの「EntAppendToCBChain\( \) - コールバックチェーンへの追加」](#)、[157 ページの「EntDeleteFromCBChain\( \) - コールバックチェーンからの削除」](#)、[258 ページの「EntSelect\( \) - 使用する表の選択」](#)、[260 ページの「EntSelectAdd\( \) - 使用する追加の表の選択」](#)、[261 ページの「EntSelectTable\( \) - 使用する表の選択」](#)

次の形式を使用します。

**Public Function** MyFunc(**ByVal** hSelect As **Long**, **ByVal** wTabId As **Integer**, **ByVal** sigKey As **Long**, **ByVal** lParam As **Long**, apiS As APISTRUCT) As **Integer**

### 変数 説明

MyFunc コールバック関数の ID

## 変数 説明

hSelect 表選択ハンドル

wTabId 変更された表の表 ID

sigKey NONE、または関連する表のキー。詳しくは、[340 ページの「関連する表」](#)を参照してください。

lParam ユーザ定義データ

apiS wTabId = ID\_DATAFILE でない場合は NULL ポインタ。この場合は、sigParent、wType、sigName および sigCat フィールドが設定された apiStruct 構造体なので、変更されたデータファイルがわかります。

この変更の hSelect に関連付けられたコールバック関数を API でこれ以上呼び出したい場合を除いて、入力した関数で 0（ゼロ）の値が返されるはずです。

戻りコード：

コード	説明
0	処理を継続します。
ゼロ以外	処理をキャンセルします。

C 言語では次の形式を使用します。

short WINAPI MyFunc (HSELECT hSelect, short wTabId, SIGNA sigKey, LONG lParam, LPAPISTRUCT pApiS)

## CALLBACKSHARES

このコールバック関数は、株式読み込み時のステータス情報に使用されます。株式読み込みについては、[269 ページの「EntSharesLoadVB\(\) - 株式の読み込み」](#)および [268 ページの「EntSharesLoad\(\) - 株式の読み込み」](#)を参照してください。

次の形式を使用します。

Public Function MyFunc(ByVal addressofFileNameCString As Long, ByVal addressofEntityCString As Long, ByVal lSize As Long, ByVal lPos As Long, ByVal lParam As Long) As Integer

## 変数 説明

MyFunc コールバック関数の ID

addressofFileNameCString 読み込みファイルまたは抽出ファイルの名前

addressofEntityCString 読み込みファイルで処理されたレコードの数。完了時に空の文字列("")

lSize 読み込みファイルのサイズ（バイト）

lPos 読み込みファイルの現在の位置（バイト数）

変数	説明
lParam	コールバック関数に入力した引数

C スタイルの文字列 `addressofFileNameCString` と `addressofEntityCString` を Visual Basic の文字列に変換する方法の説明と例は、[282 ページの「EntVBGetCStrLen\(\) - C 文字列の長さの取得」](#)を参照してください。

C 言語では次の形式を使用します。

```
short WINAPI MyFunc(LPCSTR szFileName, LPCSTR szEntity, long lSize, long lPos,
long lParam);
```

## CALLBACKSTR

このコールバック関数はアプリケーションの列挙に使用されます。詳しくは、[65 ページの「HypMultiEnum\(\) - 開いているアプリケーションの列挙」](#)を参照してください。

次の形式を使用します。

```
Public Function MyFunc(ByVal hRApp As Integer, ByVal lAddr As Long) As Long
```

### 変数 説明

MyFunc コールバック関数の ID

hRApp Hyperion Enterprise Retrieve アプリケーションハンドル

lAddr アプリケーション名である C スタイル文字列のアドレス。[282 ページの「EntVBGetCStrLen\(\) - C 文字列の長さの取得」](#)を参照してください。

戻りコード：

コード	説明
0	処理を継続します。
ゼロ以外	処理をキャンセルします。

C 言語では次の形式を使用します。

[318 ページの「コールバック関数の構文」](#) `int WINAPI MyFunc(HRETRIEVEAPP hRApp, LPSTR szApp)`

## CALLBACKUSERS

このコールバック関数は、特定のアプリケーションを使用しているユーザを列挙するために使用されます。詳しくは、[186 ページの「EntEnumUsersOnSystem\(\) - システム上のユーザの列挙」](#)を参照してください。

次の形式を使用します。

Public Function MyFunc(ByVal hApp As Long, UserInfo As Active\_User\_Info, ByVal lParam As Long) As Integer

**変数 説明**

MyFunc コールバック関数の ID

hApp アプリケーションハンドル

UserInfo ACTIVE\_USER\_INFO struct

lParam ユーザ定義データ

戻りコード :

コード	説明
0	処理を継続します。
ゼロ以外	処理をキャンセルします。

C 言語では次の形式を使用します。

short WINAPI MyFunc (HAPP hApp, LPACTIVE\_USER\_INFO pUserInfo, LONG lParam)

## CALLBACKVB

このコールバック関数は、タスクの実行中にステータス情報を提供するために EntDataExtractVB2( ) と EntUNCDataLoadVB( ) の両方で使用されます。

**注 :** 関数 EntVBCopyData( ) と EntVBGetCStrLen( ) は long コールバック関数の引数を適切な型に変換するために使用できます。

次の形式を使用します。

Public Function MyFunc(ByVal lAddrFileName As Long, argStruct As CALLBACKSTRUCT, ByVal lParam As Long) As Integer

**変数 説明**

MyFunc コールバック関数の名前

lAddrFileName 現在処理中の抽出ファイルまたは読み込みファイルのファイル名を含んでいる C スタイルの文字列のアドレス

argStruct 残りのステータス情報を含んでいる構造体 (ユーザ定義のレコードタイプ)。フィールドの説明は、TOOLKIT.BAS ファイルの CALLBACKSTRUCT を参照してください。

lParam 入力した引数

戻りコード :

コード	説明
0	処理を継続します。
ゼロ以外	処理をキャンセルします。

C 言語では次の形式を使用します。

```
short WINAPI MyFunc(LPCSTR szFileName, CALLBACKSTRUCT * pArgStruct,
LPARAM lParam)
```

## DWCALLBACK

このコールバック関数は列挙に使用されます。

関連する関数：

[88 ページの「HypEnum\( \) - 表内のレコードの列挙」](#)、[91 ページの「HypEnumOrgNames\( \) - 組織内のノードの列挙」](#)、[93 ページの「HypEnumSubAcctSig\( \) - Enumerate サブ勘定科目記号の列挙」](#)

次の形式を使用します。

```
Public Function MyFunc(ByVal hRApp As Integer, ByVal sigRecd As Long, ByVal lParam
As Long) As Integer
```

### 変数 説明

MyFunc コールバック関数の ID

hRApp Hyperion Enterprise Retrieve アプリケーションハンドル

sigRecd レコード記号

lParam ユーザ定義データ

戻りコード：

コード	説明
0	処理を継続します。
ゼロ以外	処理をキャンセルします。

C 言語では次の形式を使用します。

```
short WINAPI MyFunc(HRETRIEVEAPP hRApp, SIGNA sigRecd, LONG lParam)
```



# 表ID

## この付録の内容

表の情報.....	333
デフォルト設定の表.....	338
関連付けられている表.....	339
関連する表.....	340

この付録では、Hyperion Enterprise アプリケーションの情報が保存されている表に関する情報を提供します。

## 表の情報

表 64 は、すべての Hyperion Enterprise 表の ID、その説明、および関連付けられている表のリストです。表 ID は、`TOOLKIT.H` ファイルでの形式に従って、すべて C 言語形式の `ID_TABLENAME` で表されています。

Visual Basic を使用している場合は、すべての表 ID の先頭に `HYP_` を挿入します。`TOOLKIT.BAS` ファイルに示されているように、表 ID の Microsoft Visual Basic での正しい形式は、`HYP_ID_TABLENAME` です。記載された表の 1 つに対応する表 ID が `TOOLKIT.BAS` ファイルにない場合は、その表に対して独自の定数を定義できます。

**注：** `apiStruct` を使用する必要のある表では、`Ent...()` 形式の高レベル関数を使用する必要があります。

表 64 Hyperion Enterprise 表の ID

表 ID	名前	特記事項
ID_ACCOUNTS	勘定科目	該当しません。
ID_ACCTCONVERT	勘定科目変換	ID_ACCTCVTLIST 内の勘定科目変換一覧の記号をキーとして渡します。
ID_ACCTCVTLIST	勘定科目変換一覧	
ID_ACCTLIST	勘定科目一覧	
ID_ACCTLISTENTRY	勘定科目一覧入力	勘定科目一覧の記号をキーとして渡します。

表 ID	名前	特記事項
ID_BOOK_ENTRIES	パッケージ入力	パッケージセットの記号をキーとして渡します。
ID_BOOK_SETS	パッケージセット	
ID_BOOKS	パッケージ	
ID_CATEGORY	データ種別	
ID_CATEGORY_LINKS	データ種別リンク	ロジックの記号をキーとして渡します。
ID_CODES	コード	
ID_CURRENCY	通貨	
ID_DATAFILE	データファイル	詳しくは、 <a href="#">114 ページの「データの操作」</a> を参照してください。
ID_FORMATS	データ読み込みファイルとデータ抽出ファイルの書式	
ID_FORMULAS	計算式	ロジックの記号をキーとして渡します。
ID_FREQUENCY	期間単位	この表は、FREQ_MONTH、FREQ_QUARTER などの定数を期間単位の記号として使用します。これらの定数は TOOLINC.H ファイルで定義されています。
ID_GROUP	勘定グループ	
ID_ICSET	会社間照合セット	
ID_INTCODET	会社間詳細	ID_ICSET 内の会社間セットの記号をキーとして渡します。
ID_JOURNAL_DETAIL	仕訳帳の詳細	ID_JOURNALS 内の仕訳帳の記号をキーとして渡します。*
ID_JOURNAL_HISTORY	仕訳帳の記録	データ種別の記号をキーとして渡します。*
ID_JOURNAL_HISTORY_DETAIL	仕訳帳の記録の詳細	ID_JOURNAL_HISTORY 内の仕訳帳記録の記号をキーとして渡します。*
ID_JOURNAL_PERIOD_INFO	仕訳帳の期間情報	データ種別の記号をキーとして渡します。*
ID_JOURNAL_TEMPLATES	仕訳帳テンプレート	データ種別の記号をキーとして渡します。*
ID_JOURNAL_TEMPLATES_DETAIL	仕訳帳テンプレートの詳細	ID_JOURNAL_TEMPLATES 内の仕訳帳テンプレートの記号をキーとして渡します。*
ID_JOURNALS	仕訳帳	データ種別の記号をキーとして渡します。*

表 ID	名前	特記事項
ID_LOGIC	ロジック	DEFCHARTLOGICSIG にはデフォルトの入力ロジックの記号が入っています。さまざまなデフォルトロジックについては、TOOLINC.H ファイル内のレコード記号を参照してください。
ID_LOGIC_CAT_ATTRIB	ロジックデータ種別属性	ロジックの記号をキーとして渡します。
ID_NAMECONVERT	エンティティ変換	ID_NAMECVTLIST 内のエンティティ変換一覧の記号をキーとして渡します。
ID_NAMECVTLIST	エンティティ変換一覧	
ID_NAMELIST	エンティティ一覧	
ID_NAMELISTENTRY	エンティティ一覧入力	ID_NAMELIST 内のエンティティ一覧の記号をキーとして渡します。
ID_NAMES	エンティティ	この表を使用するに、ID_ORGANIZATION OR ID_ASSOC または ID_NAMES OR ID_ASSOC を選択して、それに関連付けられたすべての表を選択します。
ID_NODES	ノード	<p>連結組織を使用するように設定されているアプリケーションでは、データ種別をキーとして渡す必要があります。クエリによっては apiStruct 引数を渡す必要がある場合もあります。</p> <p>この表を使用するには、ID_ORGANIZATION OR ID_ASSOC を選択します。</p>
ID_ORGANIZATION	組織	連結組織を使用するように設定されているアプリケーションでは、この表を選択するときデータ種別をキーとして渡す必要があります。
ID_PERIOD	期間	
ID_PRINT	印刷	
ID_PSFDATA	PSF データ	この表を選択するときデータ種別をキーとして渡します。API 関数には apiStruct 引数を渡します。
ID_REPORT_ENTRIES	レポートセットのレポートエントリ	レポートセットの記号をキーとして渡します。
ID_REPORT_SETS	レポートセット	
ID_REPORTS	レポート	
ID_ROLLOVER	期別替	期別替セットの記号をキーとして渡します。

表 ID	名前	特記事項
ID_ROLLSET	期別替セット	
ID_RPTFREQ	レポート期間単位	<p>この表 ID は次の関数でのみ有効です。</p> <ul style="list-style-type: none"> <li>● EntSelect( )</li> <li>● EntUnSelect( )</li> <li>● EntEnum( )</li> <li>● EntQueryEx( )</li> <li>● EntFind( )</li> <li>● HypLockEx( )</li> <li>● HypUnLockEx( )</li> <li>● HypEnumEx( )</li> <li>● HypQueryEx( )</li> <li>● HypFindEx( )</li> </ul>
ID_RPTVIEW	レポートの表示形式	<p>この表 ID は次の関数でのみ有効です。</p> <ul style="list-style-type: none"> <li>● EntSelect( )</li> <li>● EntUnSelect( )</li> <li>● EntEnum( )</li> <li>● EntQueryEx( )</li> <li>● EntFind( )</li> <li>● HypLockEx( )</li> <li>● HypUnLockEx( )</li> <li>● HypEnumEx( )</li> <li>● HypQueryEx( )</li> <li>● HypFindEx( )</li> </ul>
ID_RULES	ルール	
ID_RULESEXP	数式規則	ルールの記号をキーとして渡します。
ID_RULESVAR	変数規則	ルールの記号をキーとして渡します。
ID_SCHEDULES	データ入力表	
ID_SECCLASS	セキュリティクラス	
ID_SECGRPTAB	セキュリティグループ	<p>この表は、ユーザ別のセキュリティグループのマトリックスです。マトリックス内の各セルは表内レコードです。この表を使用して各セキュリティグループのメンバを検索します。</p> <p>この表のキーフィールドは、データタイプ SECURGROUPKEY として定義されている構造体です。通常、</p>

表 ID	名前	特記事項
		ID として使用される文字列ではありません。
ID_SECRIGHTS	セキュリティ権	<p>この表は、セキュリティクラス別のユーザのマトリックスです。マトリックス内の各セルは表内レコードです。表には各セキュリティクラスに対するユーザの明示的な権限が入っています。ユーザは ID_SECUSERTAB 内の個人やグループです。</p> <p>特定のユーザとクラスのセルにレコードがない場合、そのユーザはそのクラスに明示的な権限を持っていません。ただし、所属グループの権限に基づく暗黙の権限を持っている可能性があります。</p> <p>この表のキーフィールドは、データタイプ SECURRIGHTSKEY として定義されている構造体です。通常、ID として使用される文字列ではありません。</p>
ID_SECTASK	セキュリティタスク	
ID_SECTASKFILTER	セキュリティタスクフィルタ	
ID_SECURITY	セキュリティ	<p>この表 ID は次の関数でのみ有効です。</p> <ul style="list-style-type: none"> <li>● EntSelect( )</li> <li>● EntUnSelect( )</li> <li>● HypLockEx( )</li> <li>● HypUnLockEx( )</li> </ul> <p>この表 ID は、すべてのセキュリティ表を選択または選択解除するために ID_ASSOC との組み合わせでのみ使用されます。OR 演算で ID_ASSOC を結合することを忘れた場合、ID_SECUSERTAB のみが選択または選択解除されます。</p>
ID_SECUSERTAB	セキュリティユーザ	
ID_SERVER	サーバ	†
ID_SHARES	所有率	<p>この表の各レコードは、1 つのエンティティの株のいくつか別のエンティティによって所有されているかどうかを示します。特殊なレコードとして、エンティティの発行済み株式総数と別のエンティティが所有する株式の合計が入っています。</p> <p>連結組織を使用するように設定されているアプリケーションでは、この表を選択するときデータ種別</p>

表 ID	名前	特記事項
		をキーとして渡します。この表のキーフィールドは、通常の ID とは違って、string ではありません。詳しくは、EntFind( )関数を参照してください。
ID_SUBACCTDET	サブ勘定科目の詳細	サブ勘定科目の見出しの記号をキーとして渡します。
ID_SUBACCTHDR	サブ勘定科目の見出し	
ID_SUBNAME	サブエンティティ	
ID_SUBSTRUCTURE	下位構造	
ID_SUGGEST_OWN	連結ロジックと連結比率を提案する表	
ID_USE_METHODS	使用ロジック	ロジックの記号をキーとして渡します。
ID_USERDEFFUNC	カスタム関数	

\* すべての仕訳帳表は、ほとんどの関数で apiStruct を要求します。また、仕訳帳表 ID\_JOURNAL\_DETAIL、ID\_JOURNAL\_HISTORY\_DETAIL、ID\_JOURNAL\_PERIOD\_INFO、ID\_JOURNAL\_TEMPLATES\_DETAIL を選択するときにも apiStruct を使用する必要があります。従って、これらの表を選択するときには EntSelectTableAdd( )関数しか使用できません。仕訳帳表の apiStruct には次のフィールドが必要です。sigCat、データ種別の記号。lStartPeriod、開始時間。lEndPeriod、lStartPeriod と同じ期間。仕訳帳が一度に扱うことができるのは 1 期間のみです。

† Hyperion Enterprise Server がインストールされている場合、サーバ表 (ID\_SERVER) も使用できます。この表 ID は HypQueryEx( )、HypFindEx( )、HypEnumEx( )、EntQueryEx( )、EntFind( )、EntEnum( )の関数でのみ有効です。ID\_SERVER 表は選択できません。Hyperion Enterprise リリースでクライアント／サーバ機能を使用している場合、上記の API 関数はリクエストを Hyperion Enterprise Server に転送して要求された情報を返します。

## デフォルト設定の表

表 65 に記載されている表は、デフォルト設定を含み、EntQueryDefault( )、EntUpdateDefault( )および EntSaveDefault( )などの関数で使用されます。これらの表は HypQueryEx( )、EntQueryEx( )、EntUpdate( )および EntSave( )でも使用できます。これらの表を選択する必要はありません。表はアプリケーションを開くと使用できます。

表 65 デフォルト設定表の表 ID

表 ID	表名
ID_HAPP	アプリケーション情報

表 ID	表名
ID_APPDEFAULT	アプリケーションデフォルト
ID_USERDEFAULT	ユーザデフォルト

**注：** これらの表 ID は toolinc.H ファイルで定義されています。

## 関連付けられている表

表の選択や選択解除を行うとき、表 ID と ID\_ASSOC 定数を OR 演算で結合して、表とそれに関連付けられている表を同時に選択できます。Visual Basic での例

```
HYP_ID_ORGANIZATION OR HYP_ID_ASSOC
```

C 言語での例

```
ID_ORGANIZATION | ID_ASSOC
```

表を個別に選択するのではなく、ID\_ASSOC 定数を使用します。ID\_ASSOC を使用して表の選択や選択解除を行うと、関連付けられている表も選択または選択解除されます。表 66 は、Hyperion Enterprise 表の関連付けられている表すべてのリストです。

表の多くは関連付けられている内部表を持っていますが、それらはこのリストに含まれていません。

**表 66** 関連付けられている表

表 ID	関連付けられている表
ID_ACCOUNTS	ID_GROUP ID_SUBACCTDET ID_SUBACCTHDR
ID_BOOKS	ID_BOOK_ENTRIES ID_BOOK_SETS
ID_CATEGORY	ID_FREQUENCY ID_PERIOD
ID_FORMULAS	ID_CATEGORY_LINKS ID_LOGIC_CAT_ATTRIB ID_USE_METHODS
ID_FREQUENCY	ID_PERIOD
ID_ICSET	ID_INTCODET
ID_JOURNAL_TEMPLATES	ID_JOURNAL_ENTRIES
ID_JOURNALS	ID_JOURNAL_TEMPLATES ID_JOURNAL_PERIOD_INFO ID_JOURNAL_ENTERIES ID_JOURNALS は ID_ASSOC を指定しなくても関連付けられている表を選択または選択解除します。
ID_LOGIC	ID_CODES
ID_NAMES	ID_CODES ID_CURRENCY ID_LOGIC ID_SUBNAME ID_SUBSTRUCTURE
ID_NODES	ID_SHARES,

表 ID	関連付けられている表
ID_ORGANIZATION	ID_CODES ID_CURRENCY ID_LOGIC ID_NAMES ID_NODES ID_SHARES ID_SUBNAME ID_SUBSTRUCTURE
ID_REPORTS	ID_REPORT_ENTRIES ID_REPORT_SETS
ID_ROLLSET	ID_ROLLOVER
ID_RULES	ID_RULESEXP、ID_RULESVAR
ID_SECCCLASS	ID_SECGRPTAB ID_SECRIGHTS ID_SECTASK、ID_SECTASKFILTER ID_SECUSERTAB
ID_SECGRPTAB	ID_SECCCLASS ID_SECTASK ID_SECTASKFILTER ID_SECUSERTAB
ID_SECRIGHTS	ID_SECCCLASS ID_SECGRPTAB ID_SECTASK ID_SECTASKFILTER ID_SECUSERTAB
ID_SECTASK	ID_SECCCLASS ID_SECGRPTAB ID_SECRIGHTS ID_SECTASKFILTER ID_SECUSERTAB
ID_SECTASKFILTER	ID_SECCCLASS ID_SECGRPTAB ID_SECRIGHTS ID_SECTASK ID_SECUSERTAB
ID_SECURITY	ID_SECCCLASS ID_SECGRPTAB ID_SECRIGHTS ID_SECTASK ID_SECTASKFILTER
ID_SECUSERTAB	ID_SECCCLASS ID_SECGRPTAB ID_SECRIGHTS ID_SECTASK ID_SECTASKFILTER

## 関連する表

表 67 は緊密に関連する Hyperion Enterprise の表のリストです。Visual Basic を使用している場合は、すべての表 ID の先頭に HYP\_ を挿入します。TOOLKIT.BAS ファイルにあるように、Visual Basic での表 ID の正しい形式は、HYP\_ID\_TABLENAME です。

表 67 関連する Hyperion Enterprise 表

表	必要なキー
ID_ACCTCONVERT	ID_ACCTCVTLIST
ID_ACCTLISTENTRY	ID_ACCTLIST
ID_BOOK_ENTRIES	ID_BOOK_SETS
ID_CATEGORY_LINKS	ID_LOGIC
ID_FORMULAS	ID_LOGIC
ID_INTCODET	ID_ICSET
ID_JOURNAL_DETAIL	ID_JOURNALS
ID_JOURNAL_HISTORY	ID_CATEGORY
ID_JOURNAL_HISTORY_DETAIL	ID_JOURNAL_HISTORY
ID_JOURNAL_PERIOD_INFO	ID_CATEGORY

表	必要なキー
ID_JOURNAL_TEMPLATES	ID_CATEGORY
ID_JOURNAL_TEMPLATES_DETAIL	ID_JOURNAL_TEMPLATES
ID_JOURNALS	ID_CATEGORY
ID_LOGIC_CAT_ATTRIB	ID_LOGIC
ID_NAMECONVERT	ID_NAMECVTLIST
ID_NAMELISTENTRY	ID_NAMELIST
ID_PSFDATA	ID_CATEGORY
ID_REPORT_ENTRIES	ID_REPORT_SETS
ID_ROLLOVER	ID_ROLLSET
ID_RULESEXP	ID_RULES
ID_RULESVAR	ID_RULES
ID_SHARES	ID_CATEGORY
ID_SUBACCTDET	ID_SUBACCTHDR
ID_USE_METHODS	ID_LOGIC





# クエリ属性

## この付録の内容

クエリ属性の概要 .....	343
デフォルトのクエリ属性 .....	344
デフォルト以外のクエリ属性 .....	346

この付録では、EntQueryEx()、または HypQueryEx()、および関連する関数を使用して Hyperion Enterprise のバージョン表をクエリするために使用できる Hyperion Enterprise のすべてのクエリ属性を示します。

## クエリ属性の概要

HypQueryEx()では apiStruct は処理されないため、特定のクエリに apiStruct を使用する必要がある場合は、EntQueryEx()を使用する必要があります。ほとんどのクエリでは apiStruct は必要ありませんが、代わりに EntQueryEx()で NULL が渡されます。この付録に記載されたクエリ属性には、apiStruct、およびその apiStruct で設定するフィールドを必要とするものが含まれます。apiStruct を使用する方法については、[110 ページの「apiStruct 構造体の使用」](#)を参照してください。

**注：** Visual Basic を使用している場合は、すべての表 ID の先頭に HYP\_を挿入します。TOOLKIT.BAS ファイルに示される表 ID の正しい Visual Basic 形式は、HYP\_ID\_TABLENAME です。記載された表の 1 つに対応する表 ID が TOOLKIT.BAS ファイルにない場合は、その表に対して独自の定数を定義できます。

この付録では、各クエリ属性によって返される情報の種類についても説明します。これはクエリ関数の pzbuf 引数では、この種類を使用する必要があります。例えば、クエリでタイプ LONG が返された場合、pzbuf もタイプ LONG である必要があります。C 言語でプログラミングを行っている場合は、タイプが本質的にポインタタイプでない限り、指定のタイプへのポインタを使用する必要があります。[表 68](#) には、さまざまなタイプが Visual Basic と C で示されています。

**表 68** Visual Basic タイプと C タイプ

Visual Basic タイプ	C タイプ
バイトまたは 1 文字の文字列	Char、UCHAR、または HYPBOOL。

Visual Basic タイプ	C タイプ
	HYPBOOL は、True (1) または False (0) であるブール値に使用されます。
Double	double
Long	long または LONG
Integer	short
Long	SIGNA。レコード記号である値に使用されます。
String	LPSTR

文字列を取得するクエリ属性の場合は、バッファを次のサイズに割り当てます。

- ID には HYP\_SIZELABEL+1 を使用します。
- 説明には HYP\_SIZEDESC+1 を使用します。
- エンティティ ID には HYP\_SIZEFULLNAME+1 を使用します。
- 勘定科目 ID には HYP\_SIZEFULLACCT+1 を使用します。

**注：** C 言語でプログラミングを行っている場合は、HYP\_を削除してください。

その他の文字列サイズについては、TOOLKIT.BAS ファイルと TOOLINC.H ファイルを参照してください。

## デフォルトのクエリ属性

表 69 に、一部を除き Hyperion Enterprise のほとんどの表に適用できるデフォルトのすべてのクエリ属性を示します。これらのクエリ属性が適用されない表は、この付録でそのように識別されています。

表 69 デフォルトのクエリ属性

クエリ	説明	種別
COUNT	表内のレコードの数。	VB: long C: long
DATETIME	time_t タイプは、1970 年 1 月 1 日零時からの秒数として定義されます。Microsoft の Visual C では、time_t 値を、日付と時刻のさまざまなコンポーネントのための個々のフィールド（日、月、年、時、分、その他）がある tm struct（別の Microsoft C タイプ）に変換するために localtime()関数を使用します。詳しくは、Microsoft のマニュアル（MSDN - Microsoft Developers' Network）を参照してください。	VB: Long C: time_t
DESC	表内のレコードの説明。	VB: String C: LPSTR

クエリ	説明	種別
DESCRIPTION	表内のレコードの説明。	VB: String C: LPSTR
LOCK_MODE	表のロックモード。詳しくは、 <a href="#">212 ページの「EntIsSelected() - 表が選択されているかどうかの確認」</a> を参照してください。	VB: Integer C: short
MODIFIED	表が更新されたかどうかを示します。詳しくは、 <a href="#">211 ページの「EntIsModified() - 表が変更されているかどうかの確認」</a> を参照してください。	VB: Integer C: short
NAME	ID。	VB: String C: LPSTR
NUM_ELEMENTS	要素の数。	VB: Long C: long
NUM_COLUMNS	列数。	VB: Long C: long
QRYFIRST	最初のレコード。	VB: Long C: SIGNA
QRYNEXT	次のレコード。	VB: Long C: SIGNA
SECURITYCLASS	項目のセキュリティクラス記号。	VB: Long C: SIGNA
SHORTNAME	ID。	VB: String C: LPSTR
TIME_STAMP	表のタイムスタンプ。	VB: Long C: Long
TIMESTAMP_CHK	タイムスタンプが異なる場合（別のユーザによって表が変更され、更新する必要がある場合）は TRUE（1）、それ以外の場合は FALSE（0）。	VB: Byte C: HYPBOOL
USECOUNT	表でのロックの数。	Integer short
USER_RIGHTS	SECURITY_MODIFY、SECURITY_VIEW、SECURITY_NONE、または SECURITY_RESTRICTED のいずれかの項目にアクセスするための現在のユーザ権限。	
VERIFYSIG	記号を確認するために使用します。記号が無効な場合は、NONE を返します。「注」を参照してください。	VB: 0 C: NULL

**注：** VERIFYSIG クエリ属性では、記号が有効な場合はゼロが返され、無効な場合は NONE が返されます。戻りコードとはクエリ関数 HypQueryEx() または EntQueryEx() で返された値のことで、pzBuf 引数の値ではありません。NULL を渡すことができます。このクエリ属性は、ユーザのアクセス権限のみを確認し、レコードが削除されていないかをチェックします。記号が有効な範囲内かどうかは確認しません。

## デフォルト以外のクエリ属性

次のトピックでは、デフォルトのクエリ属性を使用しない表のクエリ属性について説明しています。

### ID\_ACCOUNTS（勘定科目表）クエリ属性

デフォルトクエリ NAME では、主要勘定科目のみの ID が返されます。完全な勘定科目 ID を取得するには、クエリ属性 ACCT\_FULLNAME を使用してください。

表 70 に、勘定科目表のクエリ属性を示します。

表 70 ID\_ACCOUNTS 表クエリ属性

クエリ	説明	VB タイプ	C タイプ
ACCT_COA_SEQ_NUM	勘定科目表の連続番号。		
ACCT_COMBINE_VSIGS	主要勘定科目およびサブ勘定科目の記号を指定して、勘定科目記号を取得します。「注 1」を参照してください。	Long	SIGNA
ACCT_DECIMALS	表示する小数点以下の桁数。	Integer	short
ACCT_FIRST_SUB_SIG		Long	SIGNA
ACCT_FULLNAME	完全な勘定科目 ID major.sub.subsub を取得します。	String	LPSTR
ACCT_GROUP	勘定科目グループの記号。	Long	SIGNA
ACCT_IS_BALANCE	指定された勘定科目が残高勘定科目である場合は TRUE（1）、それ以外の場合は FALSE（0）。	BYTE	HYPBOOL
ACCT_IS_CONSOLIDATE	指定された勘定科目が連結勘定かどうかを指定します。	BYTE	HYPBOOL
ACCT_IS_CURRENCY	通貨換算を適用するかどうかを指定します。	BYTE	HYPBOOL
ACCT_IS_DYNAMIC	勘定科目が連動勘定科目かどうかを指定します。	BYTE	HYPBOOL
ACCT_IS_INTERCO	勘定科目が会社間勘定かどうかを指定します。	BYTE	HYPBOOL
ACCT_IS_MAJOR	勘定科目が主要勘定科目かどうかを指定します。	BYTE	HYPBOOL
ACCT_IS_SCALE	勘定科目が単位勘定かどうかを指定します。	BYTE	HYPBOOL

クエリ	説明	VB タイプ	C タイプ
ACCT_IS_SINGLELEVEL	サブ勘定科目を 1 つのレベル (TRUE (1) /FALSE (0)) に限定します。	BYTE	HYPBOOL
ACCT_IS_SUB	勘定科目が、第 2 レベルのサブ勘定科目を持たない第 1 レベルのサブ勘定科目 (major.sub) である場合は、TRUE (1)。	BYTE	HYPBOOL
ACCT_IS_SUBSUB	勘定科目が、第 2 レベルのサブ勘定科目 (major.sub.subsub) を持つ場合は TRUE (1)、それ以外の場合は FALSE (0)。	BYTE	HYPBOOL
ACCT_IS_VALIDATED	サブ勘定科目の種別が、固定またはなしではなく、チェック済みである場合は True (1)、それ以外の場合は FALSE (0)。	BYTE	HYPBOOL
ACCT_NEXTCHARTORDER	勘定科目表で次の勘定科目。	Long	SIGNA
ACCT_SPLIT_VSIGS	勘定科目記号から主要勘定科目、サブ勘定科目、サブサブ勘定科目の記号を取得します。「注 2」を参照してください。	Long	SIGNA
ACCT_SUBACCT_TABLE	サブ勘定科目表の記号。	Long	SIGNA
ACCT_SUBACCT_TYPE	固定、チェック済み、またはなし。TOOLINC.H ファイルの SUBACTYPE_*を参照してください。関連付けられたサブ勘定科目表の種別によって影響を受けます。	BYTE	CHAR
ACCT_SUBACCTDESC	最下位レベルのサブ勘定科目の説明 (それがあある場合)。	String	LPSTR
ACCT_SUBACCTNAME	最下位レベルのサブ勘定科目 ID (それがあある場合)。	String	LPSTR
ACCT_TYPE	資産や負債など、勘定科目の種別。TOOLINC.H ファイルの ACTTYPE_*の定義を参照してください。	Integer	short
CODE	ユーザ定義のコードの記号。	Long	SIGNA
INTERCO_NAMESIG	最下位レベルの詳細が会社間サブ勘定科目である	Long	SIGNA

クエリ	説明	VB タイプ	C タイプ
	場合に、エンティティ記号を取得します。		
LAST_ATTACHED_SUBTABLE	勘定科目の最下位レベルの詳細サブ勘定科目表を取得します。	Long	SIGNA

**注：** 主要勘定科目、サブ勘定科目、および第 2 レベルのサブ勘定科目を連結して、結果の勘定科目の記号を返すには、ACCT\_COMBINE\_VSIGS クエリ属性を使用します。そのためには、表 71 に示したフィールドで apiStruct を設定する必要があります。

**表 71** ApiStruct を設定するフィールド

フィールド	説明
u_ApiQry.sigSub	第 1 レベルのサブ勘定科目の記号、または HYP_NONE (C 言語では NONE)。
u_ApiQry.sigSubSub	第 2 レベルのサブ勘定科目の記号、または HYP_NONE (C 言語では NONE)。

EntQueryEx() を呼び出し、sigRecd 引数として主要勘定科目の記号を渡します。代わりに、第 2 レベルのサブ勘定科目を連結する 1 つのサブ勘定科目レベルを持つ勘定科目の記号を渡すこともできます。EntQueryEx() では pzBuf 引数で結果の勘定科目の記号が返されます。EntQueryEx() では次のコードのいずれかが返されます。

戻りコード：

コード	説明
ACCT_NOERR (0)	成功。
ACCT_FULL_OSUBSUSED (1)	入力勘定科目には第 1 レベルと第 2 レベルのサブ勘定科目が既に含まれているので、記号は返されません。
ACCT_FULL_1SUBUSED (2)	入力勘定科目には既に第 1 レベルのサブ勘定科目がありますが、それは apiStruct から連結されています。
ACCT_SUBSNOTVALID (3)	入力勘定科目には無効な組み合わせが含まれていたため、記号は返されませんでした。
NONE	表が未選択の場合や、無効な勘定科目である場合など、その他のエラー。

**注：** ACCT\_SPLIT\_VSIGS クエリ属性では、次のフィールドのある apiStruct が必要です。

フィールド	説明
u_ApiQry.sigSub	HYP_NONE (C 言語 NONE) に設定します。

フィールド	説明
u_ApiQry.sigSubSub	HYP_NONE (C 言語 NONE) に設定します。

EntQueryEx() を呼び出し、それに勘定科目の記号を渡します。このクエリにより、pzBuf 引数で主要勘定科目の記号が返されます。さらに EntQueryEx() では、表 ID\_SUBACCTDET の一部である第 1 レベルと第 2 レベルのサブ勘定科目記号が apiStruct の u\_ApiQry.sigSub フィールドおよび u\_ApiQry.sigSubSub フィールドで返されます。

**注：** 代わりに EntAcctSplit() 関数を使用することもできます。

## ID\_ACCTCONVERT（勘定科目変換表）クエリ属性

表 72 には、表 ID\_ACCTCVTLIST 内の各勘定科目変換一覧の個々の入力項目が含まれます。各入力項目には、Hyperion Enterprise 勘定科目と、対応する外部勘定科目名が含まれます。デフォルトのクエリ属性 NAME と SHORTNAME では、外部勘定科目名が返されます。

表 72 ID\_ACCTCONVERT 表クエリ属性

クエリ	説明	VB タイプ	C タイプ
ACONV_ACCTNAME	Hyperion Enterprise 勘定科目 ID。	String	LPSTR
ACONV_ACCTSIG	勘定科目記号。	Long	SIGNA
ACONV_ADD_SUB	加算または減算します。	Integer	short

## ID\_ACCTCVTLIST（勘定科目変換一覧表）クエリ属性

表 73 には、勘定科目変換一覧に関する情報が含まれています。各勘定科目変換一覧内の実際の入力項目は、表 ID\_ACCTCONVERT に含まれます。

表 73 ID\_ACCTCVTLIST 表クエリ属性

クエリ	説明	VB タイプ	C タイプ
FIRST_ENTRY	勘定科目変換表の最初の入力項目。	Long	SIGNA
LIST_DIRECTION	方向：Hyperion Enterprise から抽出する場合は E、Hyperion Enterprise に読み込む場合は L、両方向が有効な場合は B。	BYTE	CHAR

## ID\_ACCTLIST（勘定科目一覧表）クエリ属性

表 74 には、アプリケーションの勘定科目一覧に関する情報が含まれています。各勘定科目一覧内の勘定科目は、表 ID\_ACCTLISTENTRY に保存されます。

表 74 ID\_ACCTLIST 表クエリ属性

クエリ	説明	VB タイプ	C タイプ
FIRST_ENTRY	最初の詳細レコードの記号。	Long	SIGNA
LIST_TYPE	一覧の種別：固定（F）または連動（D）。	Integer	short

## ID\_ACCTLISTENTRY（勘定科目一覧入力表）クエリ属性

この表には、指定された勘定科目一覧の入力項目が含まれます。レコード記号（sigRecd）が HYP\_NONE（C 言語 NONE）で、勘定科目一覧が固定勘定科目一覧である場合、クエリ関数では、勘定科目一覧内の最初の入力項目がクエリされます。レコード記号が HYP\_NONE で、勘定科目一覧が連動である場合、クエリ関数では、勘定科目一覧の条件レコードがクエリされます。条件レコードにはレコード記号を使用します。

勘定科目一覧入力表には、各固定勘定科目一覧内の入力項目ごとに 1 つのレコードが、連動一覧または固定一覧ごとに 1 つの条件レコードが格納されています。条件レコードには、勘定科目一覧を作成するために使用した条件が含まれます。一部のクエリ属性は、固定一覧のみに適用されます。ほとんどのクエリ属性は、条件レコードのみに適用されます。詳しくは、[379 ページの「ID\\_NAMELISTENTRY（エンティティ一覧入力表）クエリ属性」](#)を参照してください。

表 75 に、固定一覧の入力項目に適用されるすべてのクエリ属性を示します。デフォルトのクエリ NAME、SHORTNAME、QRYNEXT、および QRYFIRST は、固定一覧のみに適用されます。

表 75 固定一覧の入力項目用 ID\_ACCTLISTENTRY クエリ属性

クエリ	説明	VB タイプ	C タイプ
ENTRY_SIG	一覧入力のエンティティ記号。	Long	SIGNA
FIRST_ENTRY	最初の入力項目の記号を取得します。	Long	SIGNA

表 76 に、条件レコードに適用されるすべてのクエリ属性を示します。

表 76 ID\_ACCTLISTENTRY 表クエリ属性

クエリ	説明	VB タイプ	C タイプ
CALC_ACCT	勘定科目が算出勘定かどうかを指定します。	BYTE	char、HYPBOOL、または short
CODE_DATA	選択されたコード。 注： 配列を割り当てるサイズを取得するには、NUM_CODES を使用します。	Long の配列	SIGNA の配列
CRITERIA_TYPES		String	LPSTR
DYNVIEW_ACCOUNT		BYTE	HYPBOOL
ENTRY_SIG	勘定科目の記号。	Long	SIGNA
FIRST_ENTRY	最初のレコードを取得します (QRYFIRST と同様です)。	Long	SIGNA
GROUP_DATA	選択した勘定科目グループ。 注： 配列を割り当てるサイズを取得するには、NUM_GROUPS を使用します。	Long の配列	SIGNA の配列
INPUT_ACCT	勘定科目が入力勘定かどうかを指定します。	BYTE	HYPBOOL
MULTI_ACCOUNT	勘定科目が複数レベルの勘定科目かどうかを指定します。	BYTE	HYPBOOL
NUM_CODES	条件として選択されたコードの数。	Long	long
NUM_GROUPS	条件として選択された定義済みグループの数。	Long	long
NUM_SUBACCTS	条件として選択されたサブ勘定科目の数。	Long	long
NUM_TABLES	条件として選択されたサブ勘定科目表の数。	Long	long
SHOW_DETAIL	この勘定科目のサブ勘定科目を表示するかどうかを TRUE (1) または FALSE (0) で指定します。	BYTE	HYPBOOL
SINGLE_ACCT	勘定科目が単一レベル勘定科目かどうかを指定します。	BYTE	HYPBOOL
SUB_MULT	サブ勘定科目が複数レベルのサブ勘定科目かどうかを指定します。	BYTE	HYPBOOL

クエリ	説明	VB タイプ	c タイプ
SUB_SINGLE	サブ勘定科目が単一レベルサブ勘定科目かどうかを指定します。	BYTE	HYPBOOL
SUBACCT_DATA	選択されたサブ勘定科目。 注： 配列を割り当てるサイズを取得するには、NUM_SUBACCTS を使用します。	Long の配列	SIGNA の配列
TABLE_DATA	選択されたサブ勘定科目表。 注： 配列を割り当てるサイズを取得するには、NUM_TABLES を使用します。	Long の配列	SIGNA の配列

## ID\_APPDEFAULT（アプリケーションデフォルト表） クエリ属性

表 77 のクエリ属性は、EntQueryDefault() で使用しますが、HypQueryEx() または EntQueryEx() でも使用できます。デフォルトのクエリはこの表には適用しません。

表 77 ID\_APPDEFAULT 表クエリ属性

クエリ	説明	VB タイプ	c タイプ
APP_ACCTFORLOCK	チェック勘定科目。	Long	SIGNA
APP_ALWAYS_EXEC_TRANS	換算ロジックを常に実行します。	BYTE	HYPBOOL
APP_BALACCT	残高レート勘定科目。	Long	SIGNA
APP_BILLIONS	十億の区切り文字。	String	LPSTR
APP_CALENDAR	アプリケーションカレンダーファイル名	String	LPSTR
APP_CONSOL_IS_PER	連結は期別です。	BYTE	HYPBOOL
APP_CURRENCY	アプリケーション基本通貨。	Long	SIGNA
APP_DATADIR	マクロ置換のあるデータディレクトリ。	String	LPSTR
APP_DECIMAL	小数の区切り文字。	String	LPSTR
APP_DESC	アプリケーションの説明。	String	LPSTR
APP_DRV_DLL	HEFILE.DLL などのデバイスドライバ DLL 名またはそれに SQL でそれに相当するもの。	String	LPSTR

クエリ	説明	VB タイプ	C タイプ
APP_EXPECTED_ENTITIES	予期されるエンティティの数。	Long	long
APP_EXPECTED_REPORTS	予期されるレポートの数。	Long	long
APP_EXPECTED_ACCTS	予期される勘定科目の数。	Long	long
APP_EXPECTED_CATS	予期されるデータ種別の数。	Long	long
APP_FLOACCT	フローレート勘定科目。	Long	SIGNA
APP_HOLDING_LOGIC	持株会社のロジック。	Long	SIGNA
APP_ID	アプリケーション ID。	String	LPSTR
APP_IMPACTFUTURECAT	将来のデータ種別に影響します。	BYTE	HYPBOOL
APP_INBOXDIR	マクロ置換のないインボックスディレクトリ。	String	LPSTR
APP_INPUT_PCT	株数ではなく、比率として親子関係を入力します。	BYTE	HYPBOOL
APP_IS_ORGBYPER	アプリケーションが、連動組織を使用するように設定されているかどうかを示します。	BYTE	HYPBOOL
APP_ISBALACCTPVA	残高勘定科目が PVA かどうかを指定します。	BYTE	HYPBOOL
APP_ISFLOACCTPVA	フロー勘定科目が PVA かどうかを指定します。	BYTE	HYPBOOL
APP_JOUR_AUTONUMBER	保存時に、仕訳帳に自動的に番号を付けます。	BYTE	HYPBOOL
APP_JOUR_REVIEW	承認する必要がある仕訳帳。	BYTE	HYPBOOL
APP_JOUR_RUN_LOGIC	仕訳帳の転記時に計算式を実行します。	BYTE	HYPBOOL
APP_JOUR_TOPLEVEL	最上位の仕訳帳の調整を許可します。	BYTE	HYPBOOL
APP_MILLIONS	百万の区切り文字。	String	LPSTR
APP_NODVIMPLIED	連動表示勘定科目の暗黙的計算はありません。	BYTE	HYPBOOL
APP_NTDATADIR	マクロ置換のないデータディレクトリ。	String	LPSTR
APP_NTINBOXDIR	マクロ置換のないインボックスディレクトリ。	String	LPSTR

クエリ	説明	VB タイプ	c タイプ
APP_NTOUTBOXDIR	マクロ置換のないアウトボックスディレクトリ。	String	LPSTR
APP_NTREPORTDIR	マクロ置換のないレポートディレクトリ。	String	LPSTR
APP_NUM_DECIMALS	表示する小数点以下の桁数。	Integer	short
APP_OUTBOXDIR	マクロ置換のあるアウトボックスディレクトリ。	String	LPSTR
APP_PATH	アプリケーションパス。	String	LPSTR
APP_REPORTDIR	マクロ置換のあるレポートディレクトリ。	String	LPSTR
APP_SERVER	クライアント／サーバオプションのサーバ名。	String	LPSTR
APP_SITE	サイト番号（0 ～ 14）。	Long	SIGNA
APP_SITE_SHIFTED	サイト番号が上位に 4 ビット、シフトされます。	Long	SIGNA
APP_STORECONDETAIL	調整後の詳細を保存します。	BYTE	HYPBOOL
APP_STORETRANDETAIL	換算の詳細を保存します。	BYTE	HYPBOOL
APP_SUBACCTSIG	共有されない一意のサブ勘定科目記号。	BYTE	HYPBOOL
APP_THOUSANDS	千の区切り文字。	String	LPSTR
APP_USE_CHILD_RATES	非グローバルレートが子のレートかどうかを指定します。	BYTE	HYPBOOL
APP_USE_MULTI_THREAD	マルチスレッドサポートを使用するかどうかを指定します（TRUE（1）または False（0））。	BYTE	HYPBOOL
APP_USE_SERVER	クライアント／サーバオプションを使用するかどうかを指定します。	BYTE	HYPBOOL
APP_USES_SYSTEM_NUM	数値書式にデフォルトのシステム設定を使用します。	BYTE	HYPBOOL
APP_USETURBO	Statutory Consolidation Engine（ACE）を使用するかどうかを指定します。	BYTE	HYPBOOL

表 78 に、ファイルベースのアプリケーションのクエリ属性を示します。これらのクエリ属性は現在使用されていませんが、これらの読み取り、更新、保存は可能です。

**表 78** ファイルベースのアプリケーションのクエリ属性

クエリ	説明	VB タイプ	C タイプ
APP_SWAPDRIVE	スワップドライブ。	String	LPSTR
APP_ISSWAPDRIVE	ユーザがスワップドライブを指定したかどうかを示します。	Long (32 ビット) Integer (16 ビット)	HYPBOOL
APP_ISCOMPRESS	ファイルの圧縮オプション。	Long (32 ビット) Integer (16 ビット)	HYPBOOL

表 79 に、SQL アプリケーションのクエリ属性を示します。

**表 79** SQL アプリケーションのクエリ属性

クエリ	説明	VB タイプ	C タイプ
APP_SQL_SERVER	SQL サーバ名。	String	LPSTR
APP_SQL_DBNAME	SQL データベース名。	String	LPSTR
APP_NTWRK_PCKT_SIZE	SQL ドライバの要求された最大パケットサイズ。	Long	long

## ID\_BOOK\_ENTRIES（パッケージ入力表）クエリ属性

表 80 に、表 ID\_BOOK-SETS の各パッケージセットに含まれるパッケージを示します。

**表 80** ID\_BOOK\_ENTRIES 表クエリ属性

クエリ	説明	VB タイプ	C タイプ
BOOK_ID	パッケージセット内のパッケージ記号。	Long	SIGNA
BOOKSET_ID	パッケージセットの記号。	Long	SIGNA
NEXT_BOOK_SIG	パッケージセット内の次のパッケージ記号。	Long	SIGNA

## ID\_BOOK\_SETS（パッケージセット表）クエリ属性

表 81 にはパッケージセットに関する情報が含まれます。各パッケージセット内の実際のパッケージは、表 ID\_BOOK-ENTRIES に保存されます。

**表 81** ID\_BOOK\_SETS 表クエリ属性

クエリ	説明	VB タイプ	C タイプ
BOOKSET_ID	パッケージセット内のレポートセットの ID。	Long	SIGNA

## ID\_BOOKS（パッケージ表）クエリ属性

表 82 に、パッケージ表のクエリ属性を示します。

表 82 ID\_BOOKS 表クエリ属性

クエリ	説明	VB タイプ	C タイプ
BOOK_COMPILED_BOOK	ヘルプの説明の長さ。	String	LPSTR
BOOK_SCRIPT	ヘルプの説明。	String	LPSTR
DOC_FILTER	マニュアルフィルタ。	Long	long
DOC_TYPE_CD	マニュアルフィルタコード。	BYTE	CHAR

## ID\_CATEGORY（データ種別表）クエリ属性

表 83 に、データ種別表のクエリ属性を示します。

表 83 ID\_CATEGORY 表クエリ属性

クエリ	説明	VB タイプ	C タイプ
CAT_CVTLITFREQTOPER	リテラルの期間単位「Q1 03」をデータ種別内の期間番号に変換します。「注 1」を参照してください。	Long	long
CAT_DATEFROMPER	データ種別の期間を、カレンダーの日付に変換します。	3 つの Integer の配列	3 つの short の配列。戻り値では、最初の要素は月、2 番目の要素は日、そして 3 番目の要素は年になります。
CAT_FISCALYEARSTART	期間番号に基づく、データ種別の会計年度の開始期間。	Long	long
CAT_FLOWYTD	期別またはデータ種別累計フロー。	BYTE	HYPBOOL
CAT_FREQ	期間単位インデックス。	Long	SIGNA
CAT_GETPERANDYEAR	IStartPeriod を期間と年にマッピングします。	CATMAP STRUCT	LPCATMAP STRUCT
CAT_GETPERIOD	期間単位、期間、および年に基づき期間を取得します。	Long	long。「注 2」を参照してください。
CAT_HASDATA		BYTE	HYPBOOL
CAT_HASORG	このデータ種別に組織が存在するかどうかを指定します。	BYTE	HYPBOOL

クエリ	説明	VB タイプ	C タイプ
CAT_JOUR_RESTARTNUMON	期間ごとに仕訳帳自動番号フラグをリセットします。	BYTE	HYPBOOL
CAT_JOUR_STARTNUMBER	仕訳帳の開始自動番号。	Long	long
CAT_MAPPERIOD	FreqA から FreqB に期間をマッピングします。「注 3」を参照してください。	Long	long
CAT_NUMPERIODS	期間数。	Integer	short
CAT_PATH	換算済み@APP のあるこのデータ種別のデータパス。	String	LPSTR
CAT_PATH_NOT	換算済み@APP なしのこのデータ種別のデータパス。	String	LPSTR
CAT_PER_LONG	「2003 年 1 月」の形式の日付を取得します。	String	LPSTR
CAT_PER_SHORT	期間ラベル（「Jan 03」）を取得します。「注 4」を参照してください。	String	LPSTR
CAT_PRIORSIG	前のデータ種別の記号。	Long	SIGNA
CAT_SCALE	単位。	BYTE	char または short
CAT_SCALE_FACTOR	1、2、3 などの単位比率。	Double	double
CAT_STARTPER	データ種別が開始される期間。	Integer	short
CAT_STARTYEAR	データ種別が開始される年。	Integer	short
CAT_STOREDET	連結詳細を保存します。	BYTE	HYPBOOL
CAT_VIEW	データの表示形式（VIEW_YTD または VIEW_PERIODIC）。	Integer	short
CODE	ユーザ定義のコード。	Long	SIGNA

関数 HypCatGetNumPeriodsEx( ) と EntCatGetNumPeriods( ) では、CAT\_NUMPERIODS クエリ属性と CAT\_MAPPERIOD クエリ属性の組み合わせを使用します。

**注：** CAT\_CVTLITFREQTOPER クエリ属性では、表 84 のフィールドのある apiStruct が必要です。

表 84 apiStruct のフィールド

フィールド	説明
lpString	リテラルの期間単位文字列のアドレス。例えば、「Q1 03」です。
u_ApiQry.bFreq_Not_CatFreq	データ単位の期間単位以外の期間単位を使用できるようにする場合は TRUE (1)、それ以外の場合は FALSE (0) です。

**注：** CAT\_GETPERIOD クエリ属性では、表 85 に示したフィールドのある apiStruct が必要です。

表 85 apiStruct のフィールド

フィールド	説明
lStartPeriod	月。
u_ApiQry.sigFreq	FREQ_MONTH などの期間単位記号。
u_ApiQry.sYear	年。例えば、2003 などです。

**注：** CAT\_MAPPERIOD クエリ属性では、表 86 のフィールドのある apiStruct が必要です。

表 86 apiStruct のフィールド

フィールド	説明
lStartPeriod	Freq A の期間番号。
u_ApiQry.sigFreq	lStartPeriod FREQ_MONTH などの期間単位。例えば、Freq A などです。データ種別のデフォルト期間単位には NONE を使用します。
u_ApiQry.sigAltFreq	例えば Freq B などの希望の期間単位。

戻り値では、pzBuf は Freq B の期間番号で、Freq A の指定の期間と同等です。

詳しくは、142 ページの「EntCatMapPeriod() - 期間から期間単位へのマッピング」または 84 ページの「HypCatMapPeriodEx() - 期間の期間単位へのマッピング」を参照してください。

**注：** CAT\_PER\_SHORT クエリ属性では、表 87 に示したフィールドのある apiStruct が必要です。

表 87 apiStruct のフィールド

フィールド	説明
lStartPeriod	期間番号。
u_ApiQry.sPeriod_View	期間単位。データ種別の期間単位を使用するには NONE に指定します。

## ID\_CATEGORY\_LINKS（データ種別リンク表）クエリ属性

表 88 に、データ種別リンク表のクエリ属性を示します。

表 88 ID\_CATEGORY\_LINKS クエリ属性

クエリ	説明	VB タイプ	C タイプ
CATLINK_LINKCAT	データ種別リンク	Long	SIGNA
CATLINK_MAINCAT	データ種別記号	Long	SIGNA

## ID\_CODES（コード表）クエリ属性

表 89 に、コード表のクエリ属性を示します。

表 89 ID\_CODES クエリ属性

クエリ	説明	VB タイプ	C タイプ
CODES_TYPE	toolinc.h ファイルの CODE_TYPE...の定義を参照してください。	Integer	short

## ID\_CURRENCY（通貨表）クエリ属性

表 90 に、通貨表のクエリ属性を示します。

表 90 ID\_CURRENCY 表クエリ属性

クエリ	説明	VB タイプ	C タイプ
CURR_SYMBOL	\$や¥などのこの通貨の記号。	String	LPSTR
MULFLAG	乗算／除算フラグの状態。通貨を乗算する場合は TRUE（1）です。	Integer	short

## ID\_DATAFILE（データファイル表）クエリ属性

データファイル表のほとんどのクエリ属性では、apiStruct が必要です。apiStruct を設定する手順は、110 ページの「apiStruct 構造体の使用」を参照してください。apiStruct で設定するフィールドについては、115 ページの「データの取得」を参照してください。さらに、一部のクエリでは apiStruct で u\_Dfa.hLogic フィールドを設定する必要がある場合があります。レコード記号（sigRecd 引数）は通常、HYP\_NONE（C 言語 NONE）に設定されます。クエリでは、指定に応じて apiStruct で設定したデータバッファとステータスバッファのデータ値とステータス値が返されます。表 91 に、データファイル表のクエリ属性を示します。

表 91 ID\_DATAFILE 表クエリ属性

クエリ	説明	VB タイプ	C タイプ
ACCT_TYPE	勘定科目が算定勘定 (LOGIC_CAL) か入力勘定 (LOGIC_INPUT または LOGIC_LINPUT) かを指定します。	Integer	short
DATAFILE_GETLOGICSTMT	勘定科目の計算式を取得します。	String	LPSTR
DATAFILE_ACCTHASDATA	要求された期間のうちデータを含んでいる期間の数を取得します。	Integer	short
DATAFILE_ACCTSIG	勘定科目の記号を取得します。	Long	SIGNA
DATAFILE_GET	データ値を取得します。	0	NULL
DATAFILE_GETACCTLIST	データファイル内から勘定科目の一覧を取得します。	Long の配列	SIGNA の配列
DATAFILE_GET_BAS	残高勘定科目の前年期末値を取得します。	0	NULL
DATAFILE_GET_BASFLO	フロー勘定科目の前年期末値を取得します。		
DATAFILE_GET_CUM	累積値を取得します。	0	NULL
DATAFILE_GET_OPE	期首残高を取得します。	0	NULL
DATAFILE_GET_PRE	前期の値を取得します。	0	NULL
DATAFILE_GETSTATUS	各期間のステータスを取得します。	0	NULL
DATAFILE_ISPER_EJLCKABLE	期間の仕訳帳を保護できるかどうかを指定します。	BYTE	HYPBOOL
DATAFILE_ISPER_EJUNLCKABLE	期間の仕訳帳を保護解除できるかどうかを指定します。	BYTE	HYPBOOL
DATAFILE_ISPER_LCKABLE	期間を保護できるかどうかを指定します。	BYTE	HYPBOOL
DATAFILE_ISPER_UNLCKABLE	期間を保護解除できるかどうかを指定します。	BYTE	HYPBOOL
PARENT	要求された期間について親／子のステータスを取得します。	Integer の配列	short の配列

**注：** 次のクエリ属性の結果は、データバッファおよびステータスバッファの両方、すなわち apiStruct、DATAFILE\_GET DATAFILE\_GET\_BAS DATAFILE\_GET\_BASFLO DATAFILE\_GET\_CUM DATAFILE\_GET\_OPE

DATAFILE\_GET\_OPEMUST DATAFILE\_GET\_PRE DATAFILE\_GETSTATUS の lpimrData アドレスおよび lpseStatus アドレスに返されます。各期間のステータスコードを解釈するには、TOOLINC.H ファイルの DF\_...ステータスタイプの定義を参照してください。

**注：** ACCT\_TYPE クエリ属性では、表 92 に示したフィールドのある apiStruct が必要です。

表 92 apiStruct のフィールド

フィールド	説明
sigCat	データ種別記号。
sigName	エンティティ記号。
u_Dfa.sigAcct	勘定科目記号。

pzBuf で返される値は、LOGIC\_CALC、LOGIC\_INPUT または LOGIC\_LINPUT です。

**注：** DATAFILE\_GETACCTLIST クエリ属性では、表 93 に示したフィールドのある apiStruct が必要です。

表 93 apiStruct のフィールド

フィールド	説明
u_ApiQry.pad	データファイル列挙フィルタ。0、DF_FILTEROUT_CALC、DF_FILTEROUT_NODATA、および toolkit.bas ファイルを参照してください。
u_ApiQry.pad2	バッファに格納できる入力項目数。

**注：** この呼び出しにより、バッファの最後の入力項目は NONE に設定されるため、 $((\#ofAccts + 1) * sizeof(SIGNA))$ によってバッファを割り当てます。DATAFILE\_GETACCTLIST クエリ属性は、Hyperion Enterprise におけるデータ抽出のパフォーマンスを改善することのみを目的として作成されました。EntQueryEx()に対する sigRecd、SigKey、dwLen などの引数は使用していません。このクエリは、バッファが十分なサイズであることを前提としているので、サイズは確認されません。レコード数に 1 を足したものを格納できるサイズのバッファを割り当てる必要があります。このようにしないと、メモリが上書きされてしまいます。

## ID\_FORMATS（書式表）クエリ属性

表 94 に、書式表のクエリ属性を示します。

表 94 ID\_FORMATS 表クエリ属性

クエリ	説明	VB タイプ	C タイプ
FORMAT_ACONVSIG	勘定科目変換表記号。	Long	SIGNA
FORMAT_DECIMALS	小数点以下の桁数。	Integer	short
FORMAT_DELIMITER	文字列の区切り文字。	BYTE	UCHAR
FORMAT_EXTRACT_DERIVED	ゼロ／データなしを非表示にします。	BYTE	CHAR
FORMAT_FILESPEC	ファイル指定子 (*.DAT)。	String	LPSTR
FORMAT_NCONVSIG	組織単位変換表記号。	Long	SIGNA
FORMAT_NEG	負の区切り文字。	BYTE	UCHAR
FORMAT_OPCODE	Opcode (乗算、除算、またはなし)。	BYTE	CHAR
FORMAT_OPERAND	オペランド (opcode で使用される数値)。	Double	double
FORMAT_SCALE	データの単位。	Integer	short
FORMAT_SUPPRESS	ゼロ／データなしを非表示にします。	BYTE	UCHAR
FORMAT_USE	書式では読み込み／抽出を使用します。	BYTE	CHAR
FORMAT_VIEW	YTD、PER、CATEGORY などのデータの表示形式。	BYTE	CHAR

## ID\_FORMULAS (計算式表) クエリ属性

この表には、ロジック表に保存された各ロジックの個別のロジックステートメントが含まれます。表 95 に、計算式表のクエリ属性を示します。

表 95 ID\_FORMULAS 表クエリ属性

クエリ	説明	VB タイプ	C タイプ
FORMULA_ACCOUNT	計算式の勘定科目記号。	Long	SIGNA
FORMULA_CAT_HAS_LOGIC	データ種別にロジックがあるかどうかをチェックします。		
FORMULA_CATEGORY	この計算式のデータ種別。	Long	SIGNA
FORMULA_COMMENT	このステートメントのコメント。	String	LPSTR
FORMULA_COMMENT_SZ	このコメントのサイズ。	Long	LONG

クエリ	説明	VB タイプ	c タイプ
FORMULA_ERROR_CODE	エラーコード（文字列リソース）。	Long	LONG
FORMULA_ERROR_SIG	エラーのあった計算式の記号。	Long	SIGNA
FORMULA_STATEMENT	実際のステートメント。	String	LPSTR
FORMULA_STATEMENT_SZ	このステートメントのサイズ。	Long	LONG

## ID\_FREQUENCY（期間単位表）クエリ属性

表 96 に、期間単位表のクエリ属性を示します。

表 96 ID\_FREQUENCY 表クエリ属性

クエリ	説明	VB タイプ	c タイプ
FREQ_BASEPERIOD		Long	SIGNA
FREQ_FISCALSAME		Integer	short
FREQ_FISCALSTART		Integer	short
FREQ_LITTOSIG	期間 ID から期間単位の記号を取得します。「注 1」を参照してください。	Long	SIGNA
FREQ_MAP_PERIODS	「注 3」を参照してください。	Integer	short
FREQ_NUMPERYEAR	年間の期間単位数。	Integer	short
FREQ_PERLONG	期間単位内の期間の説明。「注 4」を参照してください。	String	LPSTR
FREQ_PERSHORT	期間単位内の期間の ID。「注 4」を参照してください。	String	LPSTR
FREQ_RPTFREQ	関連するレポート期間単位を取得します。「注 2」を参照してください。	Long	SIGNA

**注：** FREQ\_LITTOSIG クエリ属性では、lpString フィールドが「Q1」などのリテラルな期間文字列のアドレスに設定されている apiStruct が必要です。

**注：** FREQ\_RPTFREQ クエリ属性により、Hyperion Enterprise 期間単位と表示形式が対応するレポート期間単位に変換されます。Hyperion Enterprise 期間単位の記号が、レコードの記号（sigRecd 引数）として渡され、Hyperion Enterprise 表示形式が sigKey 引数として渡されます。

**注：** `FREQ_MAP_PERIODS` クエリ属性により、1 つの期間単位内の期間が、別の期間単位内の対応する期間に変換されます。このクエリ属性では、`u_ApiQry.sPeriod_Base` フィールドがベース期間単位の記号に設定され、`lStartPeriod` フィールドが期間番号（ベース期間内の）に設定された `apiStruct` が必要です。`EntQueryEx()` または `HypQueryEx()` の `sigRecd` 引数は、期間のマッピング先の期間単位の記号です。戻り値では、`pzBuf` は期間番号または定数 `BAD_PERIOD` です。戻りコードは常に 0 です。また、`ID_CATEGORY` クエリ属性表の `CAT_MAPPERIOD` クエリ属性を参照してください。

**注：** `FREQ_PERLONG` および `FREQ_PERSHORT` クエリ属性の場合は、`sigKey` 引数として期間番号を渡します。

## ID\_GROUP（勘定科目グループ表）クエリ属性

この表のクエリ属性のほとんどは、新しい勘定科目をグループに追加するときに使用するデフォルト値です。表 97 に、勘定科目グループ表のクエリ属性を示します。

表 97 ID\_GROUPS 表クエリ属性

クエリ	説明	VB タイプ	C タイプ
ACCT_DECIMALS	表示する小数点以下の桁数。	Integer	short
ACCT_IS_BALANCE	勘定科目が残高勘定科目であるかどうかを TRUE (1) または FALSE (0) によって指定します。	BYTE	HYPBOOL
ACCT_IS_CONSOLIDATE	勘定科目が連結勘定科目であるかどうかを TRUE (1) または FALSE (0) によって指定します。	BYTE	HYPBOOL
ACCT_IS_CURRENCY	通貨換算を適用するかどうかを TRUE (1) または FALSE (0) によって指定します。	BYTE	HYPBOOL
ACCT_IS_DYNAMIC	勘定科目が連動勘定科目であるかどうかを TRUE (1) または FALSE (0) によって指定します。	BYTE	HYPBOOL
ACCT_IS_SCALE	勘定科目が単位勘定科目であるかどうかを TRUE (1) または FALSE (0) によって指定します。	BYTE	HYPBOOL
ACCT_IS_SINGLELEVEL	勘定科目を 1 つのレベルに限定します。TRUE (1) または FALSE (0)。	BYTE	HYPBOOL
ACCT_SUBACCT_TABLE	サブ勘定科目記号。	Long	SIGNA

クエリ	説明	VB タイプ	C タイプ
ACCT_SUBACCT_TYPE	サブ勘定科目種別を取得します。固定、チェック済み、なしのいずれかです。	BYTE	CHAR
ACCT_TYPE	勘定科目の種別。	Integer	short
CODE	ユーザ定義のコードの記号。	Long	SIGNA
GRPACCT_CHARTORDER	グループ内の最初の勘定科目の記号。	Long	SIGNA

## ID\_HAPP（アプリケーション情報表）クエリ属性

表 98 のクエリ属性は、EntQueryDefault() で使用します。さらに、HypQueryEx() または EntQueryEx() でも使用できます。デフォルトのクエリはこの表には適用しません。

表 98 ID\_HAPP 表クエリ属性

クエリ	説明	VB タイプ	C タイプ
HYP_HAPP_IS_SILENT	エラーメッセージが画面に表示されません。エラーメッセージを表示することが実用的ではないサーバアプリケーション用のクエリ属性です。	BYTE	HYPBOOL
HYP_PASSWORD	現在のユーザのパスワード。	String	LPSTR
HYP_SERVER_AVAIL	アプリケーションでクライアント／サーバオプションを使用するかどうか、クライアント／サーバ DLL が使用可能かどうかを指定します。TRUE (1) または FALSE (0)。	BYTE	HYPBOOL
HYP_USERID	現在のユーザのユーザ ID。	String	LPSTR

## ID\_ICSET（会社間照合セット表）クエリ属性

各セットの会社間照合勘定科目は、会社間詳細表（ID\_INTCODET）に含まれています。表 99 に、会社間照合セット表のクエリ属性を示します。

表 99 IC\_ICSET 表クエリ属性

クエリ	説明	VB タイプ	C タイプ
INTCO_GROUPSIG	表 ID_INTCODET 内の最初の詳細レコードの記号。	Long	SIGNA

クエリ	説明	VB タイプ	c タイプ
INTCO_PLUGSIG	調整勘定の記号。	Long	SIGNA

## ID\_INTCODET（会社間詳細表）クエリ属性

この表には、会社間照合セット表（ID\_ICSET）内の各会社間照合セットの会社間勘定のペアが含まれています。表 100 に、会社間詳細表のクエリ属性を示します。

表 100 ID\_INTCODET 表クエリ属性

クエリ	説明	VB タイプ	c タイプ
INTCO_ACCTSIG1	1 つの会社間勘定の記号。	Long	SIGNA
INTCO_ACCTSIG2	照合勘定科目の記号。	Long	SIGNA
INTCO_NEXTDET	セット内の次の詳細レコードの記号。詳細がなくなった場合は NONE です。	Long	SIGNA

## ID\_JOURNAL\_DETAIL（仕訳帳詳細表）クエリ属性

この表には、仕訳帳表（ID\_JOURNALS）内の各仕訳帳の詳細レコードが含まれています。デフォルトのクエリはこの表には適用しません。表 101 に、仕訳帳詳細表のクエリ属性を示します。

**注：** さまざまな仕訳帳表の HypQuery...() または EntQuery...() の呼び出しでは、その仕訳帳に有効でないクエリ属性を使用すると、エラーコード JOUERR\_INVATTR が返される場合があります。JOURERR\_INVATTR は TOOLINC.H ファイルで定義されていません。独自に定義する必要があります。値は 13 です。

表 101 ID\_JOURNAL\_DETAIL 表クエリ属性

クエリ	説明	VB タイプ	c タイプ
JOUR_ACCT_SIG	影響のある勘定科目。	Long	SIGNA
JOUR_CREDIT_AMOUNT	貸方列に入力した値。	Double	double
JOUR_DEBIT_AMOUNT	借方列に入力した値。	Double	double
JOUR_DETAIL_IS_COMPLETE		BYTE	HYPBOOL
JOUR_DETAIL_IS_DELETED	レコードが削除された場合は TRUE (1)。	BYTE	HYPBOOL
JOUR_ENTITY_SIG	この詳細で影響を受けたエンティティ。	Long	SIGNA

クエリ	説明	VB タイプ	C タイプ
JOUR_JOURNAL_SIG	この詳細が適用される仕訳帳。	Long	SIGNA
JOUR_NEXT_DET_SIG	表内の次の詳細。	Long	SIGNA
JOUR_NEXT_JRNL_DET_SIG	この仕訳帳の次の詳細。	Long	SIGNA
JOUR_OWNER_TABID	詳細の所有者の表 ID。	Integer	short
JOUR_PERIOD_NUM	仕訳帳が適用される期間番号。	Long	long

JOUR\_JOURNAL\_SIG、JOUR\_NEXT\_DET\_SIG、JOUR\_NEXT\_JRNL\_DET\_SIG、および JOUR\_OWNER\_TABID クエリ属性では、表 102 に示したフィールドのある apiStruct が必要です。

JOUR\_PERIOD\_NUM クエリ属性の場合は、apiStruct で IStartPeriod フィールドと IEndPeriod フィールドを設定する必要はありませんが、apiStruct で sigCat フィールドを指定する必要があります。

表 102 apiStruct のフィールド

フィールド	説明
sigCat	データ種別記号。
IStartPeriod	開始期間。
IEndPeriod	終了期間。

## ID\_JOURNAL\_HISTORY（仕訳帳記録表）クエリ属性

仕訳帳の記録の詳細表（ID\_JOURNAL\_HISTORY\_DETAIL）には、各仕訳帳記録の詳細レコードが含まれています。デフォルトのクエリはこの表には適用しません。表 103 に、仕訳帳記録表のクエリ属性を示します。

**注：** さまざまな仕訳帳表の HypQuery...() または EntQuery...() の呼び出しでは、その仕訳帳に有効でないクエリ属性を使用すると、エラーコード JOUERR\_INVATTR が返される場合があります。JOURERR\_INVATTR は TOOLINC.H ファイルで定義されていません。独自に定義する必要があります。値は 13 です。

表 103 ID\_JOURNAL\_HISTORY 表クエリ属性

クエリ	説明	VB タイプ	C タイプ
JOUR_AFFECTED_PARENT	仕訳帳が適用されるエンティティの親。	Long	SIGNA

クエリ	説明	VB タイプ	c タイプ
JOUR_ATTRIB	仕訳帳を貸借一致または貸借不一致として指定します。	BYTE	UCHAR 下記の「注 2」を参照してください。
JOUR_AUTONUMBER	自動的に生成された仕訳帳番号。	Long	long
JOUR_CODE_SIG	コード記号。	Long	SIGNA
JOUR_CREATE_DATETIME	ユーザの入力の現地日付と時刻。	Long	time_t 下記の「注 1」を参照してください。
JOUR_CREATED_BY_USER_ID	この仕訳帳を入力したユーザの ID。	String	LPSTR
JOUR_DETAIL_SIG	仕訳帳の最初の詳細の記号。	Long	SIGNA
JOUR_DETAIL_TABID	該当する詳細表の ID。	Integer	short
JOUR_ISBALANCED	仕訳帳が貸借一致の場合は TRUE (1)。	BYTE	HYPBOOL
JOUR_ISCOMPLETE		BYTE	HYPBOOL
JOUR_DIFFERENCE	仕訳帳内の借方合計と貸方合計の差異。	Double	double
JOUR_JOURNAL_NAME	仕訳帳の名前（データ種別と期間内で一意です）。	String	LPSTR
JOUR_JOURNAL_STATUS	仕訳帳のステータスを転記済み、転記前、承認済み、編集済み、または削除済みとして指定します。	Integer	short
JOUR_PERIOD_NUM	仕訳帳が適用される期間番号。	Long	long
JOUR_PERIODIC_FLAG	期別転記の場合は TRUE (1)、期間累計の場合は FALSE (0)。	BYTE	HYPBOOL
JOUR_POST_DATETIME	転記の現地日付と時刻。	Long	time_t 下記の「注 1」を参照してください。
JOUR_POSTED_BY_USER_ID	この仕訳帳を転記したユーザの ID。	String	LPSTR
JOUR_REVIEWED_BY_USER_ID	この仕訳帳を承認したユーザの ID。	String	LPSTR
JOUR_REVIEWED_DATETIME	承認の現地日付と時刻。	Long	time_t 下記の「注 1」を参照してください。
JOUR_TOTAL_CREDITS	すべての仕訳帳詳細の貸方合計。	Double	double
JOUR_TOTAL_DEBITS	すべての仕訳帳詳細の借方合計。	Double	double

クエリ	説明	VB タイプ	C タイプ
JOUR_TYPE	仕訳帳の種別を自動逆仕訳、最上位、または標準として指定します。	BYTE	UCHAR
JOURHIST_NEXT_HIST_SIG	記録の詳細表でのこの仕訳帳に関する最初の入力。	Long	SIGNA

**注：** time\_t タイプは、1970 年 1 月 1 日零時からの秒数として定義されます。Microsoft の Visual C では、time\_t 値を、日付と時刻のさまざまなコンポーネントのための個々のフィールド（日、月、年、時、分、その他）がある tm struct（別の Microsoft C タイプ）に変換するために localtime()関数を使用します。詳しくは、Microsoft のマニュアル（MSDN - Microsoft Developers' Network）を参照してください。

**注：** JOUR\_ATTRIB クエリ属性では、以下の定数が返されます。

- JOUR\_ATTRIB\_BALANCED
- JOUR\_ATTRIB\_UNBALANCED
- JOUR\_ATTRIB\_BAL\_BY\_ENTITY

この表のすべてのクエリ属性では、表 104 に示したフィールドのある apiStruct が必要です。

**表 104** apiStruct のフィールド

フィールド	説明
sigCat	データ種別記号。
lStartPeriod	開始期間。
lEndPeriod	終了期間。

## ID\_JOURNAL\_HISTORY\_DETAIL（仕訳帳の記録の詳細表）クエリ属性

この表には、仕訳帳記録表（ID\_JOURNAL\_HISTORY）の各仕訳帳記録の詳細レコードが含まれています。デフォルトのクエリはこの表には適用しません。表 105 に、仕訳帳の記録の詳細表のクエリ属性を示します。

**注：** さまざまな仕訳帳表の HypQuery...()または EntQuery...()の呼び出しでは、その仕訳帳に有効でないクエリ属性を使用すると、エラーコード JOUERR\_INVATTR が返される場合があります。JOUERR\_INVATTR は TOOLINC.H ファイルで定義されていません。独自に定義する必要があります。値は 13 です。

表 105 ID\_JOURNAL\_HISTORY\_DETAIL 表クエリ属性

クエリ	説明	VB タイプ	C タイプ
JOUR_ACCT_SIG	影響のある勘定科目。	Long	SIGNA
JOUR_CREDIT_AMOUNT	貸方列に入力した値。	Double	double
JOUR_DEBIT_AMOUNT	借方列に入力した値。	Double	double
JOUR_DETAIL_IS_COMPLETE	詳細が完全な場合は TRUE (1)。	BYTE	HYPBOOL
JOUR_DETAIL_IS_DELETED	レコードが削除された場合は TRUE (1)。	BYTE	HYPBOOL
JOUR_ENTITY_SIG	この詳細で影響を受けたエンティティ。	Long	SIGNA
JOUR_JOURNAL_SIG	この詳細が適用される仕訳帳。	Long	SIGNA
JOUR_NEXT_DET_SIG	表内の次の詳細。	Long	SIGNA
JOUR_NEXT_JRNL_DET_SIG	この仕訳帳の次の詳細。	Long	SIGNA
JOUR_OWNER_TABID	詳細の所有者の表 ID。	Integer	short
JOUR_PERIOD_NUM	仕訳帳が適用される期間番号。	Long	long

この表のすべてのクエリ属性では、表 106 に示したフィールドのある apiStruct が必要です。

表 106 apiStruct のフィールド

フィールド	説明
sigCat	データ種別記号。
lStartPeriod	開始期間。
lEndPeriod	終了期間。

## ID\_JOURNAL\_PERIOD\_INFO（仕訳帳の期間情報表） クエリ属性

デフォルトのクエリはこの表には適用しません。レコード記号としては、HYP\_NONE（C 言語 NONE）を渡すことができます。表 107 に、仕訳帳の期間情報表のクエリ属性を示します。

**注：** さまざまな仕訳帳表の HypQuery...() または EntQuery...() の呼び出しでは、その仕訳帳に有効でないクエリ属性を使用すると、エラーコード JOUERR\_INVATTR が返される場合があります。JOURERR\_INVATTR は

TOOLINC.H ファイルで定義されていません。独自に定義する必要があります。値は 13 です。

表 107 ID\_JOURNAL\_PERIOD\_INFO 表クエリ属性

クエリ	説明	VB タイプ	C タイプ
JOURPER_IS_PERIOD_OPENED	期間が開いている場合は TRUE (1)。	BYTE	HYPBOOL
JOURPER_NEXT_PERIOD_TO_OPEN	開いていない次の期間数。	Long	long
JOURPER_PERIOD_NUMBER	仕訳帳が適用される期間番号。	Long	long

この表のすべてのクエリ属性では、表 108 に示したフィールドのある apiStruct が必要です。

表 108 apiStruct のフィールド

フィールド	説明
sigCat	データ種別記号。
IStartPeriod	開始期間。
IEndPeriod	終了期間。

## ID\_JOURNAL\_TEMPLATES（仕訳帳テンプレート表）クエリ属性

この表には、仕訳帳テンプレートが保存されます。各仕訳帳テンプレートの詳細レコードは、仕訳帳テンプレートの詳細表、ID\_JOURNAL\_TEMPLATES\_DETAIL に保存されます。デフォルトのクエリはこの表には適用しません。表 109 に、仕訳帳テンプレート表のクエリ属性を示します。

**注：** さまざまな仕訳帳表の HypQuery...() または EntQuery...() の呼び出しでは、その仕訳帳に有効でないクエリ属性を使用すると、エラーコード JOURERR\_INVATTR が返される場合があります。JOURERR\_INVATTR は toolinc.h ファイルで定義されていません。独自に定義する必要があります。値は 13 です。

表 109 ID\_JOURNAL\_TEMPLATES 表クエリ属性

クエリ	説明	VB タイプ	C タイプ
JOUR_AFFECTED_PARENT	仕訳帳が適用されるエンティティの親。	Long	SIGNA
JOUR_ATTRIB	仕訳帳を貸借一致または貸借不一致として指定し	BYTE	UCHAR

クエリ	説明	VB タイプ	C タイプ
	ます。下記の「注 2」を参照してください。		
JOUR_CODE_SIG	コード記号。	Long	SIGNA
JOUR_CREATE_DATETIME	ユーザの入力の現地日付と時刻。	Long	time_t 下記の「注 1」を参照してください。
JOUR_CREATED_BY_USER_ID	この仕訳帳を入力したユーザの ID。	String	LPSTR
JOUR_DETAIL_SIG	仕訳帳テンプレートの最初の詳細の記号。	Long	SIGNA
JOUR_DETAIL_TABID	該当する詳細表の ID。	Integer	short
JOUR_DIFFERENCE	仕訳帳内の借方合計と貸方合計の差異。	Double	double
JOUR_ISBALANCED	仕訳帳が貸借一致の場合は TRUE (1)。	BYTE	HYPBOOL
JOUR_ISCOMPLETE		BYTE	HYPBOOL
JOUR_JOURNAL_NAME	仕訳帳の名前（データ種別と期間内で一意です）。	String	LPSTR
JOUR_PERIODIC_FLAG	期別転記の場合は TRUE (1)、期間累計の場合は FALSE (0)。	BYTE	HYPBOOL
JOUR_TEMPLATE_TYPE	仕訳帳の種別（標準または経常）。下記の「注 3」を参照してください。	BYTE	UCHAR
JOUR_TOTAL_CREDITS	すべての仕訳帳詳細の貸方合計。	Double	double
JOUR_TOTAL_DEBITS	すべての仕訳帳詳細の借方合計。	Double	double
JOUR_TYPE	仕訳帳の種別を自動逆仕訳、最上位、または標準として指定します。	BYTE	UCHAR

**注：** time\_t タイプは、1970 年 1 月 1 日零時からの秒数として定義されます。Microsoft の Visual C では、time\_t 値を、日付と時刻のさまざまなコンポーネントのための個々のフィールド（日、月、年、時、分、その他）がある tm struct（別の Microsoft C タイプ）に変換するために localtime() 関数を使用します。詳しくは、Microsoft のマニュアル（MSDN - Microsoft Developers' Network）を参照してください。

**注：** JOUR\_ATTRIB クエリ属性では、以下の定数が返されます。

- JOUR\_ATTRIB\_BALANCED
- JOUR\_ATTRIB\_UNBALANCED
- JOUR\_ATTRIB\_BAL\_BY\_ENTITY

これらの定数については、374 ページの「ID\_JOURNALS（仕訳帳表）クエリ属性」を参照してください。

**注：** JOUR\_TEMPLATE\_TYPE クエリ属性では、以下の定数が返されます。

- JOUR\_TEMPLATE\_STANDARD (1)
- JOUR\_EMPTYTYPE\_RECURRING (2)

JOUR\_TEMPLATE\_TYPE クエリ属性の定数は、TOOLINC.H ファイルで定義されていないため、独自に定義する必要があります。

この表のすべてのクエリ属性では、表 110 に示したフィールドのある apiStruct が必要です。

**表 110** apiStruct のフィールド

フィールド	説明
sigCat	データ種別記号。
lStartPeriod	開始期間。
lEndPeriod	終了期間。

## ID\_JOURNAL\_TEMPLATES\_DETAIL（仕訳帳テンプレートの詳細表）クエリ属性

この表には、仕訳帳テンプレート表（ID\_JOURNAL\_TEMPLATES）内の各仕訳帳テンプレートの詳細レコードが含まれます。デフォルトのクエリ属性はこの表には適用しません。表 111 に、仕訳帳テンプレートの詳細表のクエリ属性を示します。

**注：** さまざまな仕訳帳表の HypQuery...()または EntQuery...()の呼び出しでは、その仕訳帳に有効でないクエリ属性を使用すると、エラーコード JOUERR\_INVATTR が返される場合があります。JOURERR\_INVATTR は toolinc.h ファイルで定義されていません。独自に定義する必要があります。値は 13 です。

**表 111** ID\_JOURNAL\_TEMPLATES\_DETAIL 表クエリ属性

クエリ	説明	VB タイプ	C タイプ
JOUR_ACCT_SIG	影響のある勘定科目。	Long	SIGNA
JOUR_CREDIT_AMOUNT	貸方列に入力した値。	Double	double

クエリ	説明	VB タイプ	C タイプ
JOUR_DEBIT_AMOUNT	借方列に入力した値。	Double	double
JOUR_DETAIL_IS_COMPLETE	詳細が完全な場合は TRUE (1)。	BYTE	HYPBOOL
JOUR_DETAIL_IS_DELETED	レコードが削除された場合は TRUE (1)。	BYTE	HYPBOOL
JOUR_ENTITY_SIG	影響を受けた詳細のエンティティ。	Long	SIGNA
JOUR_JOURNAL_SIG	この詳細が適用される仕訳帳。	Long	SIGNA
JOUR_NEXT_DET_SIG	表内の次の詳細。	Long	SIGNA
JOUR_NEXT_JRNL_DET_SIG	この仕訳帳の次の詳細。	Long	SIGNA
JOUR_OWNER_TABID	詳細の所有者の表 ID。	Integer	short
JOUR_PERIOD_NUM	この仕訳帳が適用される期間番号。	Long	long

この表のすべてのクエリ属性では、表 112 に示したフィールドのある `apiStruct` が必要です。

**表 112** `apiStruct` のフィールド

フィールド	説明
<code>sigCat</code>	データ種別記号。
<code>lStartPeriod</code>	開始期間。
<code>lEndPeriod</code>	終了期間。

## ID\_JOURNALS（仕訳帳表）クエリ属性

各仕訳帳の詳細レコードは、仕訳帳の詳細表（ID\_JOURNAL\_DETAIL）に保存されます。表 113 に、仕訳帳表のクエリ属性を示します。

**注：** さまざまな仕訳帳表の `HypQuery...()` または `EntQuery...()` の呼び出しでは、その仕訳帳に有効でないクエリ属性を使用すると、エラーコード `JOUERR_INVATTR` が返される場合があります。`JOURERR_INVATTR` は `toolinc.h` ファイルで定義されていません。独自に定義する必要があります。値は 13 です。

表 113 ID\_JOURNALS 表クエリ属性

クエリ	説明	VB タイプ	C タイプ
JOUR_AFFECTED_PARENT	親の調整のみに使用されます。	Long	SIGNA
JOUR_ATTRIB	仕訳帳を貸借一致または貸借不一致として指定します。		UCHAR
JOUR_AUTONUMBER	自動的に生成された仕訳帳番号。	Long	long
JOUR_CODE_SIG	コード記号。	Long	SIGNA
JOUR_CREATE_DATETIME	ユーザの入力の現地日付と時刻。	Long	time_t 下記の「注」を参照してください。
JOUR_CREATED_BY_USER_ID	この仕訳帳を入力したユーザの ID。	String	LPSTR
JOUR_DETAIL_SIG	仕訳帳の最初の詳細の記号。	Long	SIGNA
JOUR_DETAIL_TABID	該当する詳細表の ID。	Integer	short
JOUR_DIFFERENCE	仕訳帳内の借方合計と貸方合計の差異。	Double	double
JOUR_HIST_SIG	仕訳帳記録表 (ID_JOURNAL_HISTORY) でのこの仕訳帳に関する最初の入力。	Long	SIGNA
JOUR_ISBALANCED	この仕訳帳が貸借一致の場合は TRUE (1)。	BYTE	HYPBOOL
JOUR_ISBALANCEDBYNAME	この仕訳帳が貸借一致の場合は TRUE (1)。	BYTE	HYPBOOL
JOUR_ISCOMPLETE	この仕訳帳が完全な場合は TRUE (1)。	BYTE	HYPBOOL
JOUR_ISEDITABLE	この仕訳帳が編集可能な場合は TRUE (1)。	BYTE	HYPBOOL
JOUR_ISPOSTABLE	この仕訳帳が転記可能な場合は TRUE (1)。	BYTE	HYPBOOL
JOUR_ISREVERSABLE	この仕訳帳が逆仕訳可能な場合は TRUE (1)。	BYTE	HYPBOOL
JOUR_ISREVIEWABLE	この仕訳帳が承認可能な場合は TRUE (1)。	BYTE	HYPBOOL
JOUR_ISUNPOSTABLE	この仕訳帳が転記不可能な場合は TRUE (1)。	BYTE	HYPBOOL
JOUR_JOURNAL_NAME	データ種別と期間内で一意の仕訳帳の名前。	String	LPSTR

クエリ	説明	VB タイプ	C タイプ
JOUR_JOURNAL_STATUS	仕訳帳のステータスを転記済み、転記前、承認済み、編集済み、または削除済みとして指定します。	Integer	short
JOUR_PERIOD_NUM	仕訳帳が適用される期間番号。	Long	long
JOUR_PERIODIC_FLAG	期別転記の場合は TRUE (1)、期間累計の場合は FALSE (0)。	BYTE	HYPBOOL
JOUR_POST_DATETIME	転記の現地日付と時刻。	Long	time_t 下記の「注」を参照してください。
JOUR_POSTED_BY_USER_ID	この仕訳帳を転記したユーザの ID。	String	LPSTR
JOUR_REVIEWED_BY_USER_ID	この仕訳帳を承認したユーザの ID。	String	LPSTR
JOUR_REVIEWED_DATETIME	承認の現地日付と時刻。	Long	time_t 下記の「注」を参照してください。
JOUR_TOTAL_CREDITS	すべての仕訳帳詳細の貸方合計。	Double	double
JOUR_TOTAL_DEBITS	すべての仕訳帳詳細の借方合計。	Double	double
JOUR_TYPE	仕訳帳の種別を自動逆仕訳、親、または標準として指定します。	BYTE	UCHAR

**注：** time\_t タイプは、1970 年 1 月 1 日零時から秒数として定義されます。Microsoft の Visual C では、time\_t 値を、日付と時刻のさまざまなコンポーネントのための個々のフィールド（日、月、年、時、分、その他）がある tm struct（別の Microsoft C タイプ）に変換するために localtime() 関数を使用します。詳しくは、Microsoft のマニュアル（MSDN - Microsoft Developers' Network）を参照してください。すべての定数は、toolinc.h ファイルで定義されています。

この表のすべてのクエリ属性では、表 114 に示したフィールドのある apiStruct が必要です。

**表 114** apiStruct のフィールド

フィールド	説明
sigCat	データ種別記号。
IStartPeriod	開始期間。
IEndPeriod	終了期間。

## ID\_LOGIC（ロジック表）クエリ属性

この表には、システムで使用可能なロジックが含まれます。さまざまなデフォルトロジックについては、toolinc.h ファイル内のレコード記号を参照してください。

表 115 は、ロジック表のクエリ属性のリストです。

表 115 ID\_LOGIC 表クエリ属性

クエリ	説明	VB タイプ	C タイプ
CODE	ユーザ定義のコード。	Long	SIGNA
INVALID	ロジックの検証フラグ。	BYTE	CHAR
LOGIC_TYPE	ロジックの種類：入力または連結。	Integer	short
NUM_NAMES	表示されるエンティティの数。	Integer	short

## ID\_LOGIC\_CAT\_ATTRIB（ロジックデータ種別属性表）クエリ属性

表 116 に、ロジックデータ種別属性表のクエリ属性を示します。

表 116 ID\_LOGIC\_CAT\_ATTRIB 表クエリ属性

クエリ	説明	VB タイプ	C タイプ
CATATTR_ATTRIBUTE	データ種別属性。	Integer	Word
CATATTR_CONSOL	Consol オプション。	BYTE	Uchar
CATATTR_ELIM	Elim オプション。	BYTE	Uchar
CATATTR_MAINCAT	これらの属性が適用するデータ種別の記号。	Long	SIGNA
CATATTR_NOCONSOL	NoConsol オプション。	BYTE	Uchar
CATATTR_NOELIM	NoElim オプション。	BYTE	Uchar
CATATTR_NOROUND	四捨五入なしのオプション。	BYTE	Uchar
CATATTR_NOTRANS	Notrans オプション。	BYTE	Uchar
CATATTR_ROUND	四捨五入オプション。	BYTE	Uchar
CATATTR_TRANS	TRANS オプション。	BYTE	Uchar

## ID\_NAMECONVERT（エンティティ変換表）クエリ属性

この表には、エンティティ変換一覧表（ID\_NAMECVTLIST）の各エンティティ一覧の入力項目が含まれます。各入力項目には、Hyperion Enterprise エンティティと、変換に使用する外部組織単位が含まれます。デフォルトのクエリ属性 NAME と SHORTNAME では、外部組織単位が返されます。表 117 に、エンティティ変換表のクエリ属性を示します。

表 117 ID\_NAMECONVERT 表クエリ属性

クエリ	説明	VB タイプ	C タイプ
NCONV_ADD_SUB	加算または減算します。	Integer	short
NCONV_HYPNAME	Hyperion Enterprise エンティティ。	String	LPSTR
NCONV_SIG	エンティティ記号。	Long	SIGNA

## ID\_NAMECVTLIST（エンティティ変換表）クエリ属性

この表には、アプリケーションのエンティティ変換一覧に関する情報が含まれています。各エンティティ変換表の入力項目は、エンティティ変換表（ID\_NAMECONVERT）に保存されます。表 118 に、エンティティ変換表のクエリ属性を示します。

表 118 ID\_NAMECVTLIST 表クエリ属性

クエリ	説明	VB タイプ	C タイプ
FIRST_ENTRY	エンティティ変換表の最初の入力項目。	Long	SIGNA
LIST_DIRECTION	方向：Hyperion Enterprise から抽出する場合は E、Hyperion Enterprise に読み込む場合は L、両方向の場合は B。	BYTE	CHAR

## ID\_NAMELIST（エンティティ表）クエリ属性

この表には、アプリケーションのエンティティ一覧に関する情報が含まれています。各エンティティ一覧内のエンティティは、エンティティ一覧入力表（ID\_NAMELISTENTRY）に保存されます。表 119 に、エンティティ表のクエリ属性を示します。

表 119 ID\_NAMELIST 表クエリ属性

クエリ	説明	VB タイプ	C タイプ
FIRST_ENTRY	エンティティ一覧入力表 (ID_NAMELISTENTRY) の最初の入力項目。	Long	SIGNA
LIST_TYPE	一覧の種別：固定は F、連動は D。	Integer	short

## ID\_NAMELISTENTRY（エンティティ一覧入力表）クエリ属性

この表には、エンティティ一覧表 (ID\_NAMELIST) で指定されたエンティティ一覧が固定一覧の場合、その入力項目が保存されます。レコード記号 (sigRecd 引数) が HYP\_NONE (C 言語 NONE) で、エンティティ一覧が連動ではなく固定一覧である場合、クエリ属性では、一覧の最初の入力項目がクエリされます。

エンティティ一覧入力表には、固定一覧の各入力項目のレコードと、(固定か連動にかかわらず) 各エンティティ一覧の条件レコードが含まれます。連動エンティティ一覧には、この表の条件レコードのみが含まれます。条件レコードには、エンティティ一覧を作成するために使用した条件が含まれます。

表 120 に、固定一覧の入力項目に適用されるすべてのクエリ属性を示します。デフォルトのクエリ NAME、SHORTNAME、QRYNEXT、および QRYFIRST は、固定一覧のみに適用されます。

表 120 固定一覧の入力項目用 ID\_NAMELISTENTRY クエリ属性

クエリ	説明	VB タイプ	C タイプ
ENTRY_SIG	一覧入力のエンティティ記号。	Long	SIGNA
FIRST_ENTRY	最初の入力項目の記号を取得します。	Long	SIGNA

表 121 に、条件レコードに適用されるすべてのクエリ属性を示します。

表 121 条件レコードの ID\_NAMELISTENTRY クエリ属性

クエリ	説明	VB タイプ	C タイプ
ALL_DEPS	すべての子を含みます。	BYTE	HYPBOOL
BASE_NAMES	最下位エンティティを含みます。	BYTE	HYPBOOL
CHART_DATA	選択された入力ロジック。	Long の配列	SIGNA の配列
CODE_DATA	選択されたコード。	Long の配列	SIGNA の配列
CONSOL_DATA	連結ロジック。	Long の配列	SIGNA の配列
CURRENCY_DATA	選択された通貨。	Long の配列	SIGNA の配列

クエリ	説明	VB タイプ	c タイプ
DUP_NAMES	重複エンティティを許可します。	BYTE	HYPBOOL
ELIM_NAMES	消去名のみを含みます。	BYTE	HYPBOOL
IMMED_DEPS	直属の子のみを含みます。	BYTE	HYPBOOL
INTCO_NAMES	会社間エンティティのみを含みます。	BYTE	HYPBOOL
JOURNAL_NAMES	仕訳帳エンティティのみを含みます。	BYTE	HYPBOOL
NAME_DATA	選択されたエンティティ。	Long の配列	SIGNA の配列
NUM_CHART	入力ロジック数。	Long	long
NUM_CODES	コード数。	Long	long
NUM_CONSOL	連結ロジック数。	Long	long
NUM_CURRENCY	通貨の数。	Long	long
NUM_NAMES	指定されたエンティティの数。	Long	long
NUM_ORGS	表示される組織の数。	Long	long
NUM_SUBNAMES	サブエンティティの数。	Long	long
NUM_SUBSTRUCTS	下位構造の数。	Long	long
NUM_TRANS	換算ロジック数。	Long	long
ORG_DATA	組織データ。	Long の配列	SIGNA の配列
PARENT_NAMES	親エンティティを含みます。	BYTE	HYPBOOL
SUBNAMES	サブエンティティを表示します。	BYTE	HYPBOOL
SUBNAME_DATA	選択されたサブエンティティ。	Long の配列	SIGNA の配列
SUBSTRUCT_DATA	選択された下位構造。	Long の配列	SIGNA の配列
TRANS_DATA	換算ロジック。	Long の配列	SIGNA の配列

## ID\_NAMES（エンティティ表）クエリ属性

デフォルト以外のすべてのクエリ属性では、指定されたエンティティの指定された主要コンポーネントのみにに関する情報を取得するには、サブエンティティを非表示にします。このオプションでは、u\_ApiQry.bSupSubNames フィールドが TRUE (1) に設定された apiStruct が必要です。表 122 に、エンティティ表のクエリ属性を示します。

表 122 ID\_NAMES 表クエリ属性

クエリ	説明	VB タイプ	C タイプ
ALLOW_TOPADJ	このエンティティで親の調整が可能かどうかを指定します。TRUE (1) または FALSE (0)。	BYTE	HYPBOOL
CHART_TOPADJ	親の調整の入力ロジックの記号。	Long	SIGNA
CHARTMETHOD	入力ロジック記号。	Long	SIGNA
CODE	ユーザ定義のコード。	Long	SIGNA
CURRENCY	通貨。	Long	SIGNA
DESC	エンティティとサブエンティティの説明の組み合わせを取得します。	String	LPSTR
ELIMINATIONS	エンティティが消去エンティティかどうかを指定します。TRUE (1) または FALSE (0)。	BYTE	HYPBOOL
HOLDING_COMPANY	持株会社の記号。	Long	SIGNA
INTERCOMPANY	エンティティが会社間エンティティかどうかを指定します。TRUE (1) または FALSE (0)。	BYTE	HYPBOOL
JOURNALS	エンティティで仕訳帳を使用できるかどうかを指定します。TRUE (1) または FALSE (0)。	BYTE	HYPBOOL
NAME	連結した entity.subentity ID を取得します。	String	LPSTR
NAME_COMBINE_VSIGS	指定のエンティティとサブエンティティの組み合わせの記号を取得します。「注」を参照してください。	Long	SIGNA
NAME_ENT_CODES	エンティティにコードがある場合は、株式から期間別のエンティティコードを返します。「注 2」を参照してください。	Long または Long の配列	SIGNA または SIGNA の配列
NAME_IN_ORG	指定されたエンティティが指定された組織に存在するかどうかを示します。TRUE (1) または FALSE (0)。「注 3」を参照してください。	BYTE	HYPBOOL

クエリ	説明	VB タイプ	C タイプ
NAME_IS_HOLDING_COMPANY	エンティティが別のエンティティの持株会社かどうかを指定します。TRUE (1) または FALSE (0)。	BYTE	HYPBOOL
NAME_IS_JOURPOSTED	指定されたエンティティに転記された仕訳帳があるかどうかを指定します。TRUE (1) または FALSE (0)。	BYTE	HYPBOOL
NAME_IS_JOURPARPOSTED	指定されたエンティティに転記された親仕訳帳があるかどうかを指定します。TRUE (1) または FALSE (0)。	BYTE	HYPBOOL
NAME_IS_VALID_FOR_DATA	エンティティのデータを入力できるかどうかを指定します。TRUE (1) または FALSE (0)。「注 4」を参照してください。	BYTE	HYPBOOL
NAME_MAJOR	このエンティティの主要エンティティ記号のみを返します。	Long	SIGNA
NAME_PARENT_NODE	このエンティティと親の組み合わせのノードを返します。	Long	SIGNA
NAME_SPLIT_VSIGS	エンティティの主要エンティティとサブエンティティの記号を返します。「注 5」を参照してください。	Long	SIGNA
NAMEWITHSUB	エンティティをクエリします。EUROFF を送り、それに下位構造がある場合は、EUROFF.PTADJ を返します。	Long	SIGNA
PAR_CHILD_STAT	親、子、非アクティブなどのステータスを取得します。「注 6」を参照してください。	Integer	short
PARENT	親、子、非アクティブなどのステータスを取得します。「注 6」を参照してください。	Integer	short
PARENT_IN_ANY_CAT	エンティティが任意のデータ種別の親である場合は、PARENT_ACTIVE を返します。	Integer	short
SCALE	単位。	BYTE	CHAR

クエリ	説明	VB タイプ	C タイプ
SCALE_FACTOR	1000 などの単位比率。	Double	double
STOREAUDITDET	連結中に監査証跡詳細情報を保存するかどうかを指定します。TRUE (1) または FALSE (0)。	BYTE	HYPBOOL
SUB-NAME	サブエンティティ記号。	Long	SIGNA
SUBSTRUCT	関連する下位構造の記号。下位構造がない場合は NONE。	Long	SIGNA

**注：** NAME\_COMBINE\_VSIGS クエリ属性では、u\_ApiQry.sigSub フィールドがサブエンティティの記号に設定されている apiStruct が必要です。戻り値では、pzBuf に結果の記号または NONE が含まれます。クエリ関数では次のコードのいずれかが返されます。

戻りコード：

コード	説明
0	成功。
NAME_FULL_OSUBSUSED	エンティティ記号には既にサブエンティティがありました。
NAME_SUBSNOTVALID	サブエンティティがないか、無効なサブエンティティが指定されました。
NONE	apiStruct がないか、その他のエラーです。

**注：** エンティティがエンティティ表 (ID\_NAMES) のものか、株式表 (ID\_SHARES) のものかにかかわらず、そのエンティティに関連するコードの記号を取得するには、NAME\_ENT\_CODES クエリ属性を使用します。このクエリ属性では、期間別のエンティティコードを取得する場合にのみ、apiStruct が必要です。表 123 に示したフィールドを apiStruct で設定します。

**表 123** apiStruct のフィールド

フィールド	説明
sigCat	データ種別記号。
lStartPeriod	開始期間。
lEndPeriod	終了期間。

期間の範囲 (lStartPeriod から lEndPeriod) が指定されている場合は、pzBuf は各期間の記号を格納するのに十分な大きさである必要があります。アプリケーション

が連動組織を使用するように設定されていない場合は、期間を指定する `apiStruct` を渡すかどうかにかかわらず、`pzBuf` では 1 つのコード記号しか返されません。

**注：** `NAME_IN_ORG` クエリ属性では、`sigName` フィールドが組織の記号に設定されている `apiStruct` が必要です。

**注：** `NAME_IS_VALID_FOR_DATA` クエリ属性を使用する際、エンティティに下位構造が関連付けられている場合は、`entity.subentity` の組み合わせにのみデータを入力できます。例えば、`FRANCE.ADJ` にはデータを入力できますが、`FRANCE` には入力できません。`NAME_IS_VALID_FOR_DATA` は、エンティティがこのような組み合わせである場合、または下位構造が関連付けられていない場合は、`pzBuf` で `TRUE (1)` を返します。

**注：** `NAME_SPLIT_VSIGS` クエリ属性では、`pzBuf` で主要エンティティの記号が返され、`u_ApiQry.sigSub` でサブエンティティの記号が返される `apiStruct` が必要です。

**注：** `PAR_CHILD_STAT` クエリ属性を使用する際、エンティティが親エンティティである場合は、戻り値の `pzBuf` は `PARENT_ACTIVE` になります。エンティティが子エンティティである場合は、`pzBuf` で、`CHILD_ACTIVE` が返されます。指定されたすべての期間でエンティティがアクティブではない場合は、`pzBuf` で `NAME_INACTIVE` が返されます。

このクエリ属性では、アプリケーションが連動組織を使用するように設定されている場合は、`apiStruct` が必要です。表 124 に示したフィールドを `apiStruct` で設定します。

表 124 `apiStruct` のフィールド

フィールド	説明
<code>sigCat</code>	データ種別記号。
<code>IStartPeriod</code>	開始期間。
<code>IEndPeriod</code>	終了期間。
<code>lpseStatus</code>	NULL、またはステータスバッファのアドレス。指定された範囲の各期間について、フィールドが <code>integer (C 言語 short)</code> を格納するのに十分な大きさである必要があります。このフィールドが NULL でない場合は、値は <code>pzBuf</code> ではなく、ステータスバッファで期間別に返されます。

## ID\_NODES（ノード表）クエリ属性

デフォルトのクエリ属性はこの表には適用しません。期間の範囲を指定した場合は、バッファは範囲内のすべての期間の戻り値を格納するのに十分な大きさである必要があります。表 125 に、ノード表のクエリ属性を示します。

表 125 ID\_NODES 表クエリ属性

クエリ	説明	VB タイプ	C タイプ
ADDFLAG	ゼロの値は、親にデータを追加する必要があることを、ゼロ以外の値は、親からデータを減算する必要があることを意味します。	Integer	short
CHILD	子ノードの記号。「注」を参照してください。	Long	SIGNA
COUNT	表内の項目数。	Long	long
HIDEDEPS	子を非表示にします。	BYTE	HYPBOOL
NAME_MAJOR	このエンティティの主要エンティティ記号のみを返します。	Long	SIGNA
NAMEID	サブエンティティが存在する場合、暗黙的サブエンティティを持つエンティティ記号。	Long	SIGNA
NAMESIG	サブエンティティが存在する場合、暗黙的サブエンティティを持たないエンティティ記号。	Long	SIGNA
NEXTPARENT	次の親。	Long	SIGNA
NODE_CONSOLIDATABLE		BYTE	HYPBOOL
NODEBYPER_ACTIVE	値がゼロの場合は、指定された期間中ノードが非アクティブであることを意味し、それ以外の値の場合は、ノードはアクティブです。	BYTE	HYPBOOL
NODEBYPER_ACTIVE_OR		BYTE	HYPBOOL
NODEBYPER_CONLOGIC	子の連結を実行するときに使用する親の連結ロジック。	Long	SIGNA
NODEBYPER_ISACTIVE		BYTE	HYPBOOL
NODEBYPER_ISACTIVE_OR		BYTE	HYPBOOL
NODEBYPER_PCTCONSOL	子に対する親の連結比率。	Double	double
NODEBYPER_PCTCONTROL	子に対する親の支配比率。	Double	double
NODEBYPER_PCTOWNED	子に対する親の出資比率。	Double	double
PARENT	親ノード。	Long	SIGNA
SIBLING	兄弟ノードの記号。	Long	SIGNA

クエリ	説明	VB タイプ	C タイプ
TRANSMETHOD	子の連結の実行時に使用する親の換算ロジック。	Long	SIGNA

**注：** CHILD クエリ属性でサブエンティティを非表示にするには、apiStruct の u\_ApiQry.bSupSubNames フィールドを TRUE (1) に設定します。

アプリケーションが連動組織を使用するように設定されている場合は、この表のすべてのクエリ属性で、表 126 に示したフィールドのある apiStruct が必要です。

**表 126** apiStruct のフィールド

フィールド	説明
sigCat	データ種別記号。
lStartPeriod	開始期間。
lEndPeriod	終了期間。

## ID\_ORGANIZATION（組織表）クエリ属性

表 127 に、組織表のクエリ属性を示します。

**表 127** ID\_ORGANIZATION 表クエリ属性

クエリ	説明	VB タイプ	C タイプ
SUBSTRUCT	下位構造の記号。	Long	SIGNA
TOPNODE	組織内の最上位のノードの記号。	Long	SIGNA

TOPNODE クエリ属性については、101 ページの「HypGetTopNodeEx() - トップノードの取得」を参照してください。

## ID\_PERIOD（期間表）クエリ属性

表 128 に、期間表のクエリ属性を示します。

**表 128** ID\_PERIOD 表クエリ属性

クエリ	説明	VB タイプ	C タイプ
PER_DAYS	年度内の日の番号。	Integer	short
PER_HALFYEARS	年度内の半期の番号。	Integer	short
PER_MONTHS	年度内の月の番号。	Integer	short
PER_QUARTERS	年度内の四半期の番号。	Integer	short

クエリ	説明	VB タイプ	C タイプ
PER_TRIMESTERS	年度内の三半期の番号。	Integer	short
PER_WEEKS	年度内の週の番号。	Integer	short
PER_YEARS	年。	Integer	short

## ID\_PRINT（印刷表）クエリ属性

ID\_PRINT には、Hyperion Enterprise で提供される各サンプルレポートのレコードが含まれています。各レコードには、そのレポートのページ設定とページ書式の情報が含まれます。ページ設定とページ書式の情報、プリントエンジン関数で使用するためのものです。特定のレポートが表への書き込みアクセスのあるユーザによって実行されていない場合、そのレポートのレコードは必ずしも存在しません。表 129 に、印刷表のクエリ属性を示します。

表 129 ID\_PRINT 表クエリ属性

クエリ	説明	VB タイプ	C タイプ
PRINT_PAGEFORMAT	ページ書式。	BYTE	CHAR
PRINT_PAGESETUP	ページ設定構造体。	BYTE	CHAR

## ID\_PSFDATA（PSF データ表）クエリ属性

PSF（プログラムステータスフラグ）表クエリの場合、以下のいずれかを実行できます。

- レコード記号（sigRecd 引数）として NONE を渡し、apiStruct（sigName、sigParent、および wType フィールド）でエンティティ、親、種別（ID\_REGULAR）などを指定します。API 関数はこの情報を使用して、PSF 表で適切なレコードを検索します。
- PSF 表でレコードの記号を（sigRecd 引数）として渡します。EntFind()または EntEnum()への前の呼び出しからレコード記号がわかっている場合は、この方法が効率的です。

表 130 に、PSF データ表のクエリ属性を示します。

表 130 ID\_PSFDATA 表クエリ属性

クエリ	説明	VB タイプ	C タイプ
PSF_GETSTATUS	PSF ステータス情報を取得します。		
PSF_GETSTATUSTEXT	1 つの期間の PSF ステータスとテキストを取得します。		
PSF_NAME	エンティティ記号。	Long	SIGNA

クエリ	説明	VB タイプ	C タイプ
PSF_PARENT	親エンティティ記号。	Long	SIGNA
PSF_PUTSTATUS	PSF ステータス情報を設定します。		
PSF_TYPE	PSF 種別。「注」を参照してください。		

**注：** wType および PSF\_TYPE クエリ属性では、次の定数が返されます。

- ID\_REGULAR
- ID\_ELIMINATION
- ID\_TRANSLATION
- ID\_PROPORTIONAL
- ID\_CONTRIBUTION
- ID\_ADJUSTMENT

この表のすべてのクエリ属性では、表 131 のフィールドのある apiStruct が必要です。

**表 131** apiStruct のフィールド

フィールド	説明
IStartPeriod	開始期間。
IEndPeriod	終了期間。
sigName	エンティティ記号。
sigParent	sigType が ID_REGULAR ではない場合は、親エンティティ記号。
wType	種別。上記と同じ定数を使用します。
sigCat	データ種別記号。

## ID\_REPORT\_ENTRIES（レポート入力表）クエリ属性

この表には、レポートセット表（ID\_REPORT\_SETS）内の特定のレポートセットのレポートが含まれます。表 132 に、レポート入力表のクエリ属性に示します。

**表 132** ID\_REPORT\_ENTRIES 表クエリ属性

クエリ	説明	VB タイプ	C タイプ
NEXT_REPORT_SIG	レポートセット内の次のレポート記号。	Long	SIGNA

クエリ	説明	VB タイプ	C タイプ
REPORT_ID	レポートセット内のレポート ID。	Long	SIGNA

## ID\_REPORT\_SETS（レポートセット表）クエリ属性

レポートセット内レポートは、レポート入力表（ID\_REPORT\_ENTRIES）に保存されます。表 133 に、レポートセット表のクエリ属性に示します。

表 133 ID\_REPORT\_ENTRIES 表クエリ属性

クエリ	説明	VB タイプ	C タイプ
REPSET_ID	セット内の最初のレポートの記号。	Long	SIGNA

## ID\_REPORTS（レポート表）クエリ属性

表 134 に、レポート表のクエリ属性を示します。

表 134 ID\_REPORTS 表クエリ属性

クエリ	説明	VB タイプ	C タイプ
REPORT_COMPILED_RPT		BYTE	CHAR
REPORT_SCRIPT		BYTE	CHAR

## ID\_ROLLOVER（期別替表）クエリ属性

この表には、期別替セット表（ID\_ROLLSET）内の各期別替セットの入力項目が含まれます。表 135 に、期別替表のクエリ属性を示します。

表 135 ID\_ROLLOVER 表クエリ属性

クエリ	説明	VB タイプ	C タイプ
ROLL_DESTCAT	配賦先データ種別。	Long	SIGNA
ROLL_NEXTDET	次の詳細レコード。	Long	SIGNA
ROLL_SRCCAT	配賦元データ種別。	Long	SIGNA

## ID\_ROLLSET（期別替セット表）クエリ属性

この表には、各期別替セットに関する情報が含まれています。各期別替セットの入力項目は、期別替表（ID\_ROLLOVER）に保存されます。toolinc.h ファイルの ROLL\_... の定義を参照してください。表 136 に、期別替セット表のクエリ属性を示します。

表 136 ID\_ROLLSET 表クエリ属性

クエリ	説明	VB タイプ	C タイプ
ROLL_DESTDATAOPT	配賦先データオプションを取得します。	Integer	Short
ROLL_DESTOPT	配賦先オプションを取得します。[ソースと同じにする] または [年次] です。	Integer	Short
ROLL_GROUPSIG	期別替セットのグループ記号を取得します。	Long	SIGNA
ROLL_NUMPERCOPY	コピーする配賦元の期間数を取得します。	Integer	Short
ROLL_NUMPERINCR	増分する期間数を取得します。	Integer	Short
ROLL_SRCDATAOPT	配賦元データオプションを取得します。	Integer	Short
ROLL_TYPE	期別替の種別：年度末または期別。	Integer	Short

## ID\_RPTFREQ（レポート期間単位表）クエリ属性

デフォルトのクエリはこの表には適用しません。レポート期間単位については、[20 ページの「期間単位と表示形式」](#)を参照してください。[表 137](#) に、レポート期間単位表のクエリ属性を示します。

表 137 ID\_RPTFREQ 表クエリ属性

クエリ	説明	VB タイプ	C タイプ
DESC	説明。	String	LPSTR
NAME	ラベル。文字列のサイズは少なくとも SIZERPTFREQ + 1 文字である必要があります。	String	LPSTR
RFQ_FREQ	Hyperion Enterprise 期間単位を取得します。	Integer または Long	short または SIGNA
RFQ_VIEW	Hyperion Enterprise 表示形式を取得します。	Integer または Long	short または SIGNA
SHORTNAME	ID。	String	LPSTR

## ID\_RPTVIEW（レポート表示形式表）クエリ属性

デフォルトのクエリはこの表には適用しません。[表 138](#) に、レポート表示形式表のクエリ属性を示します。

表 138 ID\_RPTVIEW 表クエリ属性

クエリ	説明	VB タイプ	C タイプ
DESC	説明。	String	LPSTR
NAME	ラベル。	String	LPSTR
RFQ_FREQ	レポートの表示形式に関連する Hyperion Enterprise 期間単位を取得します。	Integer または Long	short または SIGNA
RFQ_VIEW	レポートの表示形式に関連する Hyperion Enterprise 表示形式の種別を取得します。	Integer または Long	short または SIGNA
SHORTNAME	ID。	String	LPSTR

## ID\_RULES（更新ルール表）クエリ属性

表 139 に、更新ルール表のクエリ属性を示します。

表 139 ID\_RULES 表クエリ属性

クエリ	説明	VB タイプ	C タイプ
RULES_CHKACCOUNTSIGN	更新ルールの実行中に勘定科目記号をチェックします。		
RULES_EXECUTEALWAYS	常に更新ルールを実行します。		
RULES_EXPRESSION	このルールの最初の数式の記号。	Long	SIGNA
RULES_INST_LENGTH	ヘルプの説明の長さ。		
RULES_INSTRUCTIONS	ヘルプの説明。「注」を参照してください。	String	LPSTR
RULES_VARIABLES	このルールの最初の変数の記号。	Long	SIGNA

**注：** RULES\_INSTRUCTIONS クエリ属性を使用する場合は、最初に項目の長さをクエリし、クエリ+1 を格納するために十分な大きさのバッファを割り当ててから、項目をクエリします。

## ID\_RULESEXP（更新数式規則表）クエリ属性

この表には、更新ルール表（ID\_RULES）内の特定の更新ルールに関連する数式のレコードが含まれます。表 140 に、更新数式規則表のクエリ属性を示します。

表 140 ID\_RULESEXP 表クエリ属性

クエリ	説明	VB タイプ	C タイプ
RULESEXP_DESTACCT	この更新ルールの配賦先勘定科目。「注」を参照してください。		
RULESEXP_DESTACCT_LENGTH	配賦先勘定科目数式の長さ。		
RULESEXP_ELIMEXP	この更新ルールの消去式。「注」を参照してください。	String	LPSTR
RULESEXP_ELIMEXP_LENGTH	消去式の長さ。		
RULESEXP_IS_DELETED	削除済みフラグ。	BYTE	HYPBOOL
RULESEXP_NEXT_EXP	次の数式の記号。	Long	SIGNA
RULESEXP_PELIMEXP	この更新ルールのパートナーの消去式。「注」を参照してください。	String	LPSTR
RULESEXP_PELIMEXP_LENGTH	パートナーの消去式の長さ。		
RULESEXP_PROPEXP	この更新ルールの比率式。「注」を参照してください。	String	LPSTR
RULESEXP_PROPEXP_LENGTH	パートナーの比率式の長さ。		
RULESEXP_RULEID	更新ルールの記号。	Long	SIGNA

**注：** RULESEXP\_ELIMEXP および RULESEXP\_PROPEXP を使用する場合は、最初に項目の長さをクエリし、クエリ+1 を格納するために十分な大きさのバッファを割り当ててから、項目をクエリします。

## ID\_RULESVAR（変数規則表）クエリ属性

この表には、更新ルール表（ID\_RULES）内の特定の更新ルールに関連する変数のレコードが含まれます。表 141 に、変数規則表のクエリ属性を示します。

表 141 ID\_RULESVAR 表クエリ属性

クエリ	説明	VB タイプ	C タイプ
RULESVAR_IS_DELETED	削除済みフラグ。	BYTE	HYPBOOL
RULESVAR_NEXT_VAR	次の変数の記号。	Long	SIGNA
RULESVAR_RULEID	この変数を使用するルールの記号。	Long	SIGNA

クエリ	説明	VB タイプ	C タイプ
RULESVAR_VAR_LENGTH	この変数の長さ。		
RULESVAR_VARIABLE	この変数のテキスト。 「注」を参照してください。	String	LPSTR

**注：** RULESVAR\_VARIABLE クエリ属性を使用する場合は、項目の長さをクエリし、クエリ+1 を格納するために十分な大きさのバッファを割り当ててから、項目をクエリします。

## ID\_SCHEDULES（データ入力表）クエリ属性

表 142 に、データ入力表のクエリ属性を示します。

表 142 ID\_SCHEDULES 表クエリ属性

クエリ	説明	VB タイプ	C タイプ
ACCTWIDTH	勘定科目のセルの幅。	Integer	short
ALLPERIODS	すべての期間／今期を表示します。すべての期間を表示する場合は、ゼロ以外の値、今期のみを表示する場合はゼロです。	BYTE	HYPBOOL
AUTORECALC	自動的に計算式を実行します。	BYTE	HYPBOOL
BOTTOMLIST	最下部の勘定科目一覧。	Long	SIGNA
PERIODWIDTH	データのセルの幅。	Integer	short
SCHED_BOLD	太字体を使用します。	BYTE	HYPBOOL
SCHED_FONTNAME	使用するフォントの名前。	String	LPSTR
SCHED_FONTSIZE	使用するフォントのサイズ。	Integer	short
SCHED_ITALIC	斜体を使用します。	BYTE	HYPBOOL
SHOWLOGIC	［ロジック］ウィンドウを表示します。	BYTE	HYPBOOL
SUPPNODATA	データなしの勘定科目を非表示にするかどうかを指定します。TRUE（1）または FALSE（0）。	BYTE	HYPBOOL
TOPLIST	最上部の勘定科目一覧。	Long	SIGNA
USELABEL	勘定科目の ID（1）または説明（0）を表示します。	BYTE	HYPBOOL

## ID\_SECCLASS（セキュリティクラス表）クエリ属性

エンティティ、データ種別、勘定科目、タスクなどは、それぞれセキュリティクラスに割り当てられます。SECURITYCLASS クエリ属性を使用して、エンティティの場合は ID\_NAMES、データ種別の場合には ID\_CATEGORY のように、適切な表をクエリします。このクエリ属性では、項目またはタスクのセキュリティクラスの記号が返されます。セキュリティクラス表でその項目またはタスクの現在のユーザの権限をクエリするには、この記号をクエリ属性 SECCLASS\_CURRENTRIGHTS で使用します。pzBuf 引数で返される可能性のある権限は、SECRIGHT\_VIEW、SECRIGHT\_RESTRICTED、または SECRIGHT\_NONE です。表 143 に、セキュリティクラス表のクエリ属性を示します。

表 143 ID\_SECCLASS 表クエリ属性

クエリ	説明	VB タイプ	C タイプ
SECCLASS_CURRENTRIGHTS	このクラスの現在のユーザ権限。	Integer	short

## ID\_SECGRPTAB（セキュリティグループ表）クエリ属性

この表は、ユーザ別のセキュリティグループのマトリックスです。マトリックス内の各セルは表内レコードです。セキュリティグループのフィールドは、セキュリティユーザ表 (ID\_SECUSERTAB) のグループの記号です。ユーザフィールドは、グループのメンバのユーザ記号です。このメンバは個人である場合も、別のグループである場合もあります。

セキュリティグループ表のセキュリティグループの最初のレコードは、セキュリティユーザ表 (ID\_SECUSERTAB) に対してクエリ属性 SECUSER\_FIRSTMEMBER を使用してクエリすることにより、見つけることができます。サポートされていないクエリの場合、このクエリ関数では、SECURERR\_NOERR (0) が返されます。表 144 に、セキュリティグループ表のクエリ属性を示します。

表 144 ID\_SECGRPTAB 表クエリ属性

クエリ	説明	VB タイプ	C タイプ
SECGRP_GROUPSIG	セキュリティグループ。	Long	SIGNA
SECGRP_NEXTSIG	グループ表内の次のグループメンバの記号。	Long	SIGNA
SECGRP_USERSIG	セキュリティユーザ表 (ID_SECUSERTAB) のグループメンバのメンバ ID。	Long	SIGNA

## ID\_SECRIGHTS（セキュリティ権表）クエリ属性

この表は、セキュリティクラス別のユーザのマトリックスです。マトリックス内の各セルは表内レコードです。ユーザは、セキュリティユーザ表

(ID\_SECUSERTAB) 内の個人ユーザである場合も、セキュリティグループである場合もあります。表 145 に、セキュリティ権表のクエリ属性を示します。

表 145 ID\_SECRIGHTS 表クエリ属性

クエリ	説明	VB タイプ	C タイプ
SECRIGHTS_NEXTCLASS	次のクラスの記号。	Long	SIGNA
SECRIGHTS_RIGHTS	割り当てられた権限。	Integer	short
SECRIGHTS_SECCLASS	セキュリティクラス。	Long	SIGNA
SECRIGHTS_USERSIG	権限ユーザの記号。	Long	SIGNA

## ID\_SECTASK（セキュリティタスク表）クエリ属性

表 146 に、セキュリティタスク表のクエリ属性を示します。

表 146 ID\_SECTASK 表クエリ属性

クエリ	説明	VB タイプ	C タイプ
SECTASK_SECCLASS	タスクセキュリティクラス。	Long	SIGNA
SECTASK_TASKFILTER	タスクフィルタ。	String	LPSTR

## ID\_SECTASKFILTER（セキュリティタスクフィルタ表）クエリ属性

表 147 に、セキュリティタスクフィルタ表のクエリ属性を示します。

表 147 ID\_SECTASKFILTER 表クエリ属性

クエリ	説明	VB タイプ	C タイプ
SECFILTER_TASKFILTER	タスク記号	Long	SIGNA

## ID\_SECUSERTAB（セキュリティユーザ表）クエリ属性

この表には、ユーザとセキュリティグループのレコードが含まれています。セキュリティグループのメンバを検索したり、セキュリティグループ表を列挙したりするには、セキュリティグループ表 ID\_SECGRPTAB のクエリ属性を使用します。サポートされていないクエリの場合、このクエリ関数では、SECURERR\_NOERR (0) が返されます。表 148 に、セキュリティユーザ表のクエリ属性を示します。

表 148 ID\_SECUSERTAB 表クエリ属性

クエリ	説明	VB タイプ	C タイプ
SECUSER_CLASSRIGHTS	現在のユーザのこのクラスに対する権限。		
SECUSER_FIRSTMEMBER	ID_SECGRPTAB 表の記号であるグループの最初のメンバを返します。	Long	SIGNA
SECUSER_NUM_GROUPS	セキュリティグループの数を取得します。	Long	long
SECUSER_PASSWORD	ユーザパスワード。	String	LPSTR
SECUSER_STATE	ユーザの状態。		
SECUSER_TYPECODE	ユーザ／グループフラグ。グループの場合は、SECURITY_GROUP。	BYTE	CHAR

## ID\_SERVER（サーバ表）クエリ属性

この表へのクエリは、Hyperion Enterprise Server がシステムにインストールされている場合にのみ使用できます。デフォルトのクエリ属性はこの表には適用しません。表 149 に、サーバ表のクエリ属性を示します。

表 149 ID\_SERVER 表クエリ属性

クエリ	説明	VB タイプ	C タイプ
NAME	サーバ名。	String	LPSTR
SHORTNAME	サーバ名。		
SVR_MACHINE	サーバのマシン名。	String	LPSTR
SVR_PORT	サーバ上の Hyperion Enterprise サービスのポート番号。	Integer	short

## ID\_SHARES（株式表）クエリ属性

この表の各レコードでは、1つのエンティティの株のいくつが別のエンティティによって所有されているかが指定されます。特別なレコードにより、エンティティによって発行された株式の合計、および他のエンティティによって所有されている株式の合計が示されます。表 150 に、株式表のクエリ属性を示します。

表 150 ID\_SHARES 表クエリ属性

クエリ	説明	VB タイプ	C タイプ
SHARES_ENT_CODES	ユーザ定義のエンティティコードを取得します。	Long または Long の配列	SIGNA または SIGNA の配列

クエリ	説明	VB タイプ	C タイプ
SHARES_NONVOTING	株式の数。	Double または Double の配列	double または double の配列
SHARES_PERCENT_NONVOT	株式の数、または未払株の合計。	Double または Double の配列	double または double の配列
SHARES_PERCENT_VOT	議決権株の数、または未払議決権株の合計。	Double または Double の配列	double または double の配列
SHARES_SIGOWNED	株式の所有エンティティのエンティティ記号。	Long	SIGNA
SHARES_SIGOWNER	株式の所有者エンティティのエンティティ記号。	Long	SIGNA
SHARES_TOT_ISSUED_NONVOT	エンティティの未払株の合計。「注」を参照してください。	Double または Double の配列	double または double の配列
SHARES_TOT_ISSUED_VOT	エンティティの発行済議決権株の合計。「注」を参照してください。	Double または Double の配列	double または double の配列
SHARES_TOT_OWNED_NONVOT	エンティティについて他のすべてのエンティティが所有している株式の合計。「注」を参照してください。	Double または Double の配列	double または double の配列
SHARES_TOT_OWNED_VOT	エンティティについて他のすべてのエンティティが所有している議決権株の合計。「注」を参照してください。	Double または Double の配列	double または double の配列
SHARES_ULTMETHOD	連結ロジック。	Long または Long の配列	SIGNA または SIGNA の配列
SHARES_ULTPCTCONSOL	最終連結比率。	Double または Double の配列	double または double の配列
SHARES_ULTPCTCTL	最終支配比率。	Double または Double の配列	double または double の配列
SHARES_ULTPCTOWNED	この所有者と所有エンティティのペアの最終出資比率。	Double または Double の配列	double または double の配列
SHARES_VOTING	議決権株の数。	Double または Double の配列	double または double の配列

次のクエリ属性では、アプリケーションが連動組織を使用するように設定されていない場合でも apiStruct が必要です。apiStruct の sigName フィールドをエンティティの記号に設定します。

- SHARES\_TOT\_ISSUED\_NONVOT
- SHARES\_TOT\_ISSUED\_VOT
- SHARES\_TOT\_OWNED\_NONVOT

- SHARES\_TOT\_OWNED\_VOT

アプリケーションが連動組織を使用するように設定されている場合は、データ種別記号を sigKey 引数として渡し、表 151 に示したフィールドのあるすべてのクエリについて apiStruct を設定します。

**表 151** apiStruct のフィールド

フィールド	説明
sigCat	データ種別記号。
IStartPeriod	開始期間。
IEndPeriod	終了期間。

期間の範囲を指定した場合は、バッファは範囲内のすべての期間の戻り値を格納するのに十分な大きさである必要があります。

## ID\_SUBACCTDET（サブ勘定科目詳細表）クエリ属性

この表には、サブ勘定科目ヘッダー表（ID\_SUBACCTHDR）内の各サブ勘定科目表のサブ勘定科目が含まれます。表 152 に、サブ勘定科目詳細表のクエリ属性を示します。

**表 152** ID\_SUBACCTDET 表クエリ属性

クエリ	説明	VB タイプ	C タイプ
SUBACCT_ALT_SIG	代替タイプが SUB_ALT_TYPE_NORMAL でない場合は、通貨またはエンティティの記号。	Long	SIGNA
SUBACCT_ALT_SIG_TYPE	代替タイプ。「注」を参照してください。	BYTE	CHAR
SUBACCT_NEXTDETAIL	サブ勘定科目表の次の詳細サブ勘定科目レコードの記号。	Long	SIGNA
SUBACCT_REVSIGN	このエントリの符号を逆にします。	BYTE	HYPBOOL
SUBACCT_TABLE	次のレベルのサブ勘定科目表がある場合はその記号。	Long	SIGNA
SUBACCT_TYPE	サブ勘定科目種別：固定、チェック済み、なしのいずれかです。	BYTE	CHAR
SUBACCT_VIA_ALTSIG	入力記号は通貨またはエンティティ記号です。詳細レコードの記号を出力します。	Long	SIGNA

**注：** SUBACCT\_ALT\_SIG\_TYPE クエリ属性では、以下の定数が返されます。

- SUB\_ALT\_TYPE\_NORMAL
- SUB\_ALT\_TYPE\_CURRENCY
- SUB\_ALT\_TYPE\_INTCO

## ID\_SUBACCTHDR（サブ勘定科目ヘッダー表）クエリ属性

この表には、サブ勘定科目表に関する情報が含まれています。各勘定科目表の入力項目は、サブ勘定科目詳細表（ID\_SUBACCTDET）保存されます。[表 153](#) に、サブ勘定科目ヘッダー表のクエリ属性を示します。

**表 153** ID\_SUBACCTHDR 表クエリ属性

クエリ	説明	VB タイプ	C タイプ
SUBTABLE_FIRSTDETAIL	サブ勘定科目表の最初の詳細レコード。	Long	SIGNA
SUBTABLE_IS_INTERCO	サブ勘定科目表が会社間かどうかを指定します。 TRUE (1) または FALSE (0)。	BYTE	HYPBOOL
SUBTABLE_TYPE	サブ勘定科目表タイプ。 詳しくは、TOOLINC.H ファイルを参照してください。	Integer	short

**注：** 記号 0 (Toolinc.h の SIG\_SUBHIDRCURRENCY 定数) を持つレコードが通貨サブ勘定科目表です。記号 1 (Toolinc.h の SIG\_SUBHIDRINTERCOMPANY 定数) を持つレコードが会社間サブ勘定科目表です。

## ID\_SUBNAME（サブエンティティ表）クエリ属性

この表には、サブエンティティ表に関する情報が含まれています。各サブエンティティ表の入力項目は、サブエンティティ表（ID\_SUBNAME）に含まれます。[表 154](#) に、サブエンティティ表のクエリ属性を示します。

**表 154** ID\_SUBNAME 表クエリ属性

クエリ	説明	VB タイプ	C タイプ
CHARTMETHOD	入力ロジック記号。	Long	SIGNA
CODE	ユーザ定義のコード。	Long	SIGNA
CURRENCY	通貨。	Long	SIGNA
JOURNALS	サブエンティティで仕訳帳を使用できるかどうかを	BYTE	HYPBOOL

クエリ	説明	VB タイプ	C タイプ
	指定します。TRUE (1) または FALSE (0)。		
PAR_CHILD_STAT	エンティティをクエリし、親、子、非アクティブなどのステータスを取得します。「注」を参照してください。	Integer	short
PARENT	親エンティティ。	Integer	short
SCALE	単位。	BYTE	CHAR
SUBSTRUCT	関係付けられた下位構造の記号。下位構造が関連付けられていない場合は NONE。	Long	SIGNA

PAR\_CHILD\_STAT クエリ属性を使用する際、エンティティが親エンティティである場合は、pzBuf で PARENT\_ACTIVE が返されます。エンティティが子エンティティである場合は、pzBuf で、CHILD\_ACTIVE が返されます。指定されたすべての期間でエンティティがアクティブではない場合は、pzBuf で NAME\_INACTIVE が返されます。

このクエリ属性では、アプリケーションが連動組織を使用するように設定されている場合は、apiStruct が必要です。表 155 に示したフィールドを apiStruct で設定します。

**表 155** apiStruct のフィールド

フィールド	説明
sigCat	データ種別記号。
IStartPeriod	開始期間。
IEndPeriod	終了期間。
lpseStatus	NULL、またはステータスバッファのアドレス。  その範囲の各期間について、フィールドが integer (C 言語 short) を格納するのに十分な大きさである必要があります。このフィールドが NULL でない場合は、値は pzBuf ではなく、ステータスバッファで期間別に返されます。

## ID\_SUBSTRUCTURE（下位構造表）クエリ属性

表 156 に、下位構造表のクエリ属性を示します。

**表 156** ID\_SUBSTRUCTURE 表クエリ属性

クエリ	説明	VB タイプ	C タイプ
SUBSTRUCT	下位構造の記号	Long	SIGNA
TOPNODE	TopNode フィールド	Long	SIGNA

## ID\_SUGGEST\_OWN（連結ロジックと連結比率の提案表）クエリ属性

表 157 に、連結ロジックと連結比率の提案表のクエリ属性を示します。

表 157 ID\_SUGGEST\_OWN 表クエリ属性

クエリ	説明	VB タイプ	C タイプ
SUGGEST_OWN_BOT_INCLUSIVE	範囲に下限値を含める場合は TRUE (1)。	BYTE	HYPBOOL
SUGGEST_OWN_BOT_RANGE	支配比率の下限値を取得します。	Double	double
SUGGEST_OWN_CONS_METHOD	支配比率が範囲内の場合は、この連結ロジックを使用します。	Long	SIGNA
SUGGEST_OWN_PCT_CONSOL	連結ロジックが一致する場合は、この連結比率を使用します。	Double	double
SUGGEST_OWN_PCT_TYPE	使用する連結比率の種類。「注」を参照してください。		
SUGGEST_OWN_TOP_INCLUSIVE	範囲に上限値を含める場合は TRUE (1)。	BYTE	HYPBOOL
SUGGEST_OWN_TOP_RANGE	支配比率の上限値を取得します。	Double	double

注： SUGGEST\_OWN\_PCT\_TYPE クエリ属性では、以下の定数が返されます。

- SUGGEST\_OWN\_USE\_NUMBER
- SUGGEST\_OWN\_USE\_PCONTROL
- SUGGEST\_OWN\_USE\_POWN
- SUGGEST\_OWN\_USE\_POWNMIN

## ID\_USE\_METHODS（使用ロジック表）クエリ属性

表 158 に、使用ロジック表のクエリ属性を示します。

表 158 ID\_USE\_METHODS 表クエリ属性

クエリ	説明	VB タイプ	C タイプ
USEMETHOD_CATSIG	USE ステートメントが適用されるデータ種別の記号	Long	SIGNA
USEMETHOD_METHODSIG	使用するロジックの記号	Long	SIGNA

## ID\_USERDEFAULT（ユーザデフォルト表）クエリ属性

この表のクエリ属性は、EntQueryDefault( )関数で使用しますが、HypQueryEx()またはEntQueryEx()でも使用できます。デフォルトのクエリはこの表には適用しません。表 159 に、ユーザデフォルト表のクエリ属性を示します。

表 159 ID\_USERDEFAULT 表クエリ属性

クエリ	説明	VB タイプ	C タイプ
CURACCOUNT	現在の勘定科目。	String	LPSTR
CURACCOUNTSIG	現在の勘定科目の記号。	Long	SIGNA
CURCATEGORY	現在のデータ種別。	String	LPSTR
CURCATEGORYSIG	現在のデータ種別記号。	Long	SIGNA
CURFREQ	現在の期間単位。	String	LPSTR
CURFREQSIG	現在の期間単位記号。	Long	SIGNA
CURNAME	現在のエンティティ。	String	LPSTR
CURNAMESIG	現在のエンティティ記号。	Long	SIGNA
CURORG	現在の組織。	String	LPSTR
CURORGSIG	現在の組織記号。	Long	SIGNA
CURPERIOD	現在の期間。	String	LPSTR
CURPERIODSIG	現在の期間の記号。	Long	SIGNA
CURSCALESIG	現在の単位の記号。	Long	SIGNA
CURSCHEDULE	データを入力する現在のデータ入力表。	String	LPSTR
CURSCHEDULESIG	現在のデータ入力表の記号。	Long	SIGNA
CURVIEWSIG	現在の表示形式。	Long	SIGNA
DELETEERRLOG	前のエラーログファイルを削除するかどうかを指定します。	Integer	short
ENTERKEY	Enter キーの動作。	Integer	short
ERROR_MODE	エラーモードを表示します。[Verbose（詳細）]、[Terse（簡易）]、または[すべて]のいずれかです。toolinc.h ファイルのERR_NO_...の定義を参照してください。	Integer	short

クエリ	説明	VB タイプ	C タイプ
SAVECHANGES	終了時に選択ボックスの変更を保存するかどうかを指定します。	Integer	short
SECURITYCLASS	現在のユーザのセキュリティパスワード。	Long	SIGNA
SHOWFOLDERS		Integer	short
SHOWPALETTE	パレットを表示するかどうかを指定します。	Integer	short
SHOWSTATBAR	ステータスバーを表示するかどうかを指定します。	Integer	short
SHOWTOOLBAR	ツールバーを表示するかどうかを指定します。	Integer	short
TRACE_FILE	トレースデータの出力ファイル。	String	LPSTR
TRACE_LEVEL	有効なトレースレベル。	Integer	short
USER_DEFAULTSECURITY	新しい要素のデフォルトのセキュリティクラス。	Long	SIGNA
USER_ERRORASZERO	Hyperion Enterprise Retrieve でエラーをゼロとして表示するかどうかを指定します。	BYTE	HYPBOOL
USER_EXECUTEEXCEL	ワークシートオプションの後で Microsoft Excel を実行するかどうかを指定します。	BYTE	HYPBOOL
USER_NODATAASZERO	Hyperion Enterprise Retrieve でデータなしをゼロとして表示するかどうかを指定します。	BYTE	HYPBOOL
USER_SCALEHPVAL	オラクル社の Hyperion(R) Enterprise(R) Retrieve で HPVAL 値を単位で表示するかどうかを指定します。	BYTE	HYPBOOL
USERPATH	エラーログファイルのユーザパス。	String	LPSTR

表 160 に色の設定のための ID\_USERDEFAULT の追加のクエリ属性を示します。

**表 160** 色の設定のための ID\_USERDEFAULT 表クエリ属性

クエリ	説明	VB タイプ	C タイプ
DESKTOPBORDER	デスクトップ境界線の色。	Long	long
BACKGROUNDCOLOR	モードなしのダイアログボックスの背景色。	Long	long

クエリ	説明	VB タイプ	c タイプ
DIALOGBACKCOLOR	ポップアップダイアログボックスの背景色。	Long	long
DIALOGTEXTCOLOR	ダイアログボックステキスト（静的）の色。	Long	long
DIALOGCAPTIONTEXTCOLOR	ダイアログボックス見出しの背景色。	Long	long
DIALOGCAPTIONBACKCOLOR	ダイアログボックス見出しテキストの色。	Long	long
TABLECAPTIONTEXTCOLOR	表の列見出しの背景色。	Long	long
TABLECAPTIONBACKCOLOR	表の列見出しテキストの色。	Long	long
SELECTTEXTCOLOR	選択したテキストの色。	Long	long
SELECTBACKCOLOR	選択した背景色。	Long	long
INPUTTEXTCOLOR	入力データのテキストの色。	Long	long
INPUTBACKCOLOR	入力データの背景色。	Long	long
CALCTEXTCOLOR	算出データのテキストの色。	Long	long
CALCBACKCOLOR	算出データの背景色。	Long	long
PARENTTEXTCOLOR	親データのテキストの色。	Long	long
PARENTBACKCOLOR	親データの背景色。	Long	long
DERIVEDTEXTCOLOR	派生データのテキストの色。	Long	long
DERIVEDBACKCOLOR	派生データの背景色。	Long	long
ADMINMODIFYTEXTCOLOR	システム変更テキストの色。	Long	long
ADMINMODIFYBACKCOLOR	システム変更の背景色。	Long	long
ADMINVIEWTEXTCOLOR	システム表示のテキストの色。	Long	long
ADMINVIEWBACKCOLOR	システム表示の背景色。	Long	long
NAMESMODIFYTEXTCOLOR	エンティティ変更テキストの色。	Long	long
NAMESMODIFYBACKCOLOR	エンティティ変更の背景色。	Long	long
NAMESVIEWTEXTCOLOR	エンティティ表示のテキストの色。	Long	long

クエリ	説明	VB タイプ	C タイプ
NAMESVIEWBACKCOLOR	エンティティ表示の背景色。	Long	long
NAMESSELECTTEXTCOLOR	選択したエンティティテキストの色。	Long	long
NAMESSELECTBACKCOLOR	選択したエンティティの背景色。	Long	long
NAMEISCONSOLTEXTCOLOR	有効な連結範囲テキストの色。	Long	long
NAMENOTCONSOLTEXTCOLOR	無効な連結範囲テキストの色。	Long	long
NAMECONSOLBACKCOLOR	連結範囲の背景色。	Long	long
CONSOLEKCOLOR	OK ステータステキストの色。	Long	long
CONSOLNOTOKCOLOR	影響ありステータステキストの色。	Long	long
NODATATEXTCOLOR	データなしステータステキストの色。	Long	long
OTHERDATATEXTCOLOR	データステータステキストの色。	Long	long
STATUSBACKCOLOR	ステータスの背景色。	Long	long
ADMINRESTRICTTEXTCOLOR	システム制限テキストの色。	Long	long
ADMINRESTRICTBACKCOLOR	システム制限の背景色。	Long	long
DYNVIEWTEXTCOLOR	連動表示のテキストの色。	Long	long
DYNVIEWBACKCOLOR	連動表示の背景色。	Long	long

## ID\_USERDEFFUNC（カスタム関数表）クエリ属性

表 161 に、カスタム関数表のクエリ属性を示します。

表 161 ID\_USERDEFFUNC 表クエリ属性

クエリ	説明	VB タイプ	C タイプ
UDF_CHART	関数が、入力ロジック関数として設定されるかどうかを指定します。	BYTE	HYPBOOL
UDF_CONSOL	関数が、連結ロジック関数として設定されるかどうかを指定します。	BYTE	HYPBOOL

クエリ	説明	VB タイプ	C タイプ
UDF_NOTRANS	関数が、換算関数として設定されるかどうかを指定します。	BYTE	HYPBOOL
UDF_INSTRUCTION_LENGTH	指示の長さを取得します。	Integer	short
UDF_EXPRESSION_LENGTH	数式の長さを取得します。	Integer	short
UDF_INSTRUCTIONS	指示を取得します。「注」を参照してください。	String	LPSTR
UDF_EXPRESSION	数式を取得します。「注」を参照してください。	String	LPSTR

**注：** クエリ属性 UDF\_INSTRUCTIONS および UDF\_EXPRESSION の場合は、項目の長さをクエリし、十分な大きさ（クエリされたサイズ+1）の文字列バッファを割り当ててから、項目をクエリします。

# 索引

## 記号

16 ビット関数の宣言

関数の宣言, [17](#)

16 ビットプログラム

32 ビットへの変換, [292](#)

## A - Z

apiStruct, [110](#)

Visual Basic での, [19](#)

初期化, [209](#)

追加の表の選択での使用, [262](#)

表の選択での使用, [261](#)

apiStruct 構造体の使用, [19](#), [110](#)

Basic, [16](#)

C, [16](#)

C++, [16](#)

CALLBACK12, [318](#)

CALLBACK6, [320](#)

CALLBACKAPI, [319](#)

CALLBACKAPP, [320](#)

CALLBACKCREATE, [320](#)

CALLBACKDBENUM, [321](#)

CALLBACKDBLOAD, [322](#), [323](#), [324](#)

CALLBACKJOUREXTRACT, [325](#)

CALLBACKJOURLOAD, [326](#)

CALLBACKLOGIC, [326](#)

CALLBACKSECLOAD, [327](#)

CALLBACKSEL, [328](#)

CALLBACKSTR, [330](#)

CALLBACKUSERS, [330](#)

CALLBACKVB, [331](#)

C 文字列

長さの取得, [282](#)

DATECONV( ), [82](#)

DWCALLBACK, [332](#)

Ent...(), [289](#)

Entacctask( ), [127](#)

Entacctlistask( ), [128](#)

Entacctsplitt( ), [129](#)

Entappendtochain( ), [130](#)

Entappextract( ), [130](#)

Entappextractvb( ), [132](#)

Entappload( ), [134](#)

Entapploadvb( ), [135](#)

Entasciitodouble( ), [139](#)

Entcatask( ), [140](#)

Entcatgetnumperiods( ), [140](#)

Entcatgetpershort( ), [141](#)

Entcatmapperiod( ), [142](#)

Entcloseapplication( ), [143](#)

Entconsolidate( ), [143](#)

Entcreateapplication( ), [145](#)

Entdataextract( ), [149](#)

Entdataextractvb( ), [151](#)

Entdatafileopen( ), [155](#)

Entdateconv( ), [157](#)

Entdeletefromchain( ), [157](#)

Entdiscardchanges( ), [158](#)

Entdiscarddefault( ), [159](#)

EntDSMDataExtractVB(), [162](#)

Ententityask( ), [166](#)

Ententitylistask( ), [167](#)

EntEnum( )

ID\_SECGRPTAB との使用, [179](#)

ID\_SECRIGHTS との使用, [180](#)

ID\_SECTASK との使用, [180](#)

ID\_SECUSERTAB との使用, [180](#)

ID\_SHARES との使用, [181](#)

ID\_SUBACCTDET との使用, [182](#)

Entenum( ), [171](#)

ID\_ACCTLISTENTRY との使用, [172](#)

ID\_INTCODET との使用, [174](#)

ID\_JOURNAL\_DETAIL との使用, [175](#)

ID\_JOURNAL\_HISTORY との使用, [176](#)

ID\_JOURNAL\_TEMPLATES との使用, [176](#)

ID\_JOURNALS との使用, [174](#)

- ID\_NAMELISTENTRY との使用, 177
- ID\_NAMES との使用, 178
- ID\_NODES との使用, 178
- ID\_ROLLOVER との使用, 179
- ID\_RPTVIEW との使用, 179
- EntEnumApplications( ), 182
- EntEnumOrgEntities( ), 183
- EntEnumSubAcctSig( ), 184
- EntEnumUsersOnSystem( ), 186
- EntFind( ), 186
  - ID\_ACCTCONVERT との使用, 187
  - ID\_INTCODET との使用, 188
  - ID\_NAMECONVERT との使用, 188
  - ID\_NAMES との使用, 189
  - ID\_SECGRPTAB との使用, 189
  - ID\_SECRIGHTS との使用, 189
  - ID\_SHARES との使用, 190
  - 仕訳帳表との使用, 188
- EntFindEntityInOrg( ), 190, 191
  - 追加例, 191
- EntFormatNumber( ), 192
- EntFreqAsk( ), 193
- EntGetAccountsInputType( ), 193
- EntGetActiveModule( ), 196
- EntGetAppProfileLong( ), 196
- EntGetAppProfileString( ), 197
- EntGetCatPerFreq( ), 198
- EntGetChild( ), 199, 200
- EntGetEntitySig( ), 200
- EntGetHappFromSelect( ), 201
- EntGetLastError( ), 202
- EntGetLastErrorByHapp( ), 203
- EntGetOrgLevel( ), 204
- EntGetPerView( ), 204
- EntGetProfileLong( ), 205
- EntGetProfileString( ), 206
- EntGetRightsToTask( ), 207
- ENTGETSIBLING( )
  - 追加例, 208
- EntGetSibling( ), 207, 208
- EntGetVarAddr( ), 208
- EntInitApiStruct( ), 209
- EntIsAccountInput( ), 210
- EntIsEntityParent( ), 210
- EntIsModified( ), 211
- EntIsSelected( ), 212
- EntJournalExtract( ), 213
- EntJournalExtractVB( ), 216
- EntJournalLoad( ), 220
- EntJournalLoadVB( ), 222
- EntLogicAttach( ), 225
- EntLogicDetach( ), 227
- EntLogicDiscard( ), 228
- EntLogicExport( ), 228
- EntLogicExportVB( ), 230
- EntLogicImport( ), 233
- EntLogicImportVB( ), 234
- EntMessageVB( ), 238
- EntMsgLogTaskEndTime( ), 239
- EntMsgLogTaskStartTime( ), 240
- EntMsgLogTime( ), 241
- EntOpenApplication( ), 242
- EntOpenServerApplication( ), 243
- EntPageExtract( ), 244
- EntPageLoad( ), 245
- EntPerAsk( ), 245
- EntQryRptFreq( ), 246
- EntQuery( ), 247
- EntQueryDefault( ), 248
- EntQueryEx( ), 247, 248
- EntQueryExStr( ), 248
- EntRegisterMsgBox( ), 249
- EntRunRollover( ), 250
- EntSave( ), 250
- EntSaveDefault( ), 251
- EntSecurityExtract( ), 252
- EntSecurityExtractVB( ), 253
- EntSecurityLoad( ), 255
- EntSecurityLoadVB( ), 256
- EntSelect( ), 258
- EntSelectAdd( ), 260
- EntSelectTable( ), 261
- EntSelectTableAdd( ), 262, 263
- EntSetActiveModule( ), 264
- EntUNCDDataLoad( ), 270
- EntUNCDDataLoad( ) - データの読み込み, 270
- EntUNCDDataLoadVB( ), 272
- EntUnSelect( ), 276
- EntUpdate( ), 277
- EntUpdateDefault( ), 278
- EntVBCopyData( ), 283
- EntVBCopyStr( ), 283
- EntVBGetCStrLen( ), 282
- EntWriteAppProfileLong( ), 279

- EntWriteAppProfileString( ), 280
- EntWriteProfileLong( ), 281
- EntWriteProfileString( ), 281
- FALSE
  - 定義, 17
- fortran, 16
- HACCESS32.LIB
  - インポートライブラリファイル, 20
- Hyp...( )
  - 表選択用の関数, 79
- Hyp...( )関数
  - Hyperion Enterprise でサポート, 288, 289
- HypAcctAsk( ), 27
- HypAcctListAskEx( ), 28
- HypCatAskEx( ), 28
- HypCatGetNumPeriodsEx( ), 83
- HypCatGetPerShortEx( ), 84
- HypCatMapPeriodEx( ), 84
- HypConstruct( ), 23, 29
- HypConstructEx( ), 30
- HypDestruct( ), 23, 31
- HYPENT.INI ファイル
  - 数値の書き込み, 281
  - 数値の取得, 205
  - 文字列の書き込み, 281
  - 文字列の取得, 206
- HypEnum( )
  - 会社間表のアップグレード, 307
- HypEnumAcctListEntriesEx( ), 85
- HypEnumApplications( ), 87
- HypEnumEx( ), 88
- HypEnumNameListEntriesEx( ), 90
- HypEnumOrgNames( ), 91
- HypEnumSubAcctSig( ), 93
- Hyperion Enterprise SE
  - Hyperion Enterprise へのコードのアップグレード, 291
- Hyperion Enterprise SE から Hyperion Enterprise への変換, 306
- Hyperion Enterprise 期間単位, 20
- Hyperion Enterprise の表の変更, 306
  - 使用されなくなったセキュリティ表, 311
- Hyperion Retrieve
  - 関数, 288
- HypFind( )関数
  - アップグレード, 298
- HypFindEx( ), 94
- HypFindNameInOrgEx( ), 95
- HypFreqAsk( ), 32
- HypGetCatPerFreq( ), 96
- HypGetChild( ), 97
- HypGetDefJour( ), 33
- HypGetDefJourCat( ), 34
- HypGetDefJourPer( ), 34
- HypGethApp( )
  - 関数, 284
- HypGethSelect( )
  - 関数, 284
- HypGetNameSig( ), 98
- HypGetOrgLevel( ), 99
- HypGetPerViewEx( ), 100
- HypGetSibling( ), 101
- HypGetTopNodeEx( ), 101
- HypHPACC( ), 35
- HypHPBET( ), 36
- HypHPCAD( ), 36
- HypHPCAL( ), 37
- HypHPCDE( ), 38
- HypHPCommit2( ), 39
- HypHPCUREx( ), 40
- HypHPDRV( ), 41
- HypHPECO( ), 41
- HypHPFlush( ), 42
- HypHPFLW( ), 43
- HypHPFRE( ), 44
- HypHPFUL( ), 45
- HypHPHEA( ), 46
- HypHPINC( ), 47
- HypHPINP( ), 47
- HypHPJOUR( ), 49
- HypHPKEY( ), 52
- HypHPKEYEx( ), 50
- HypHPLNK( ), 51
- HypHPOWN( ), 53
- HypHPPBE( ), 55
- HypHPPCH( ), 56
- HypHPSCA( ), 57
- HypHPVAL( ), 58
- HypHPVal( )
  - 戻り値動作の設定, 72
- HypHPVAL2( ), 59
- HypHPVALEx( ), 61
- HypHPVal の戻りステータス, 76
- HypIsNameParentEx( ), 102

HypJourAsk( ), 62  
 HypJourDetAsk( ), 63  
 HypLock()  
   アップグレード, 298, 307  
 HypLockEx( ), 103  
 HypMultiAsk( ), 63  
 HypMultiDefault( ), 64  
 HypMultiDeinit( ), 65  
 HypMultiEnum( ), 65  
 HypMultiGet( ), 66  
 HypMultiInit( ), 67  
 HypNamAskEx( ), 67  
 HypNameListAskEx( ), 68  
 HypPerAsk( ), 69  
 HypQryNode()  
   アップグレード, 303  
   クエリサイズのアップグレード, 303  
 HypQryRptFreqEx( ), 105  
 HypQuery( )  
   アップグレード, 298  
   会社間表の HypQuery()へのアップグレード, 307  
 HypQuery( )のアップグレード  
   使用されなくなったクエリ属性, 299  
 HypQueryEx( ), 105  
 HypQuerySig()  
   アップグレード, 302  
 HypSetDefJour( ), 70  
 HypSetDefJourCat( ), 70  
 HypSetDefJourPer( ), 71  
 HypSethSelect( ), 285  
 HypSetMVMode( ), 72  
 HypSetProgramName( ), 72  
 HypUnLock()  
   アップグレード, 298, 307  
 HypUnLockEx( ), 106  
 HypValidateLnkParams( ), 73  
 HypValidateParams( ), 74  
 HypValidateParamsEx( ), 75  
 HypValReturn( ), 76  
 ID\_ACCOUNTS, 346  
 ID\_ACCTCONVERT, 349  
 ID\_ACCTCVTLIST, 349  
 ID\_ACCTLIST, 350  
 ID\_ACCTLISTENTRY, 350  
 ID\_APPDEFAULT, 352  
 ID\_BOOK\_ENTRIES, 355  
 ID\_BOOK\_SETS, 355  
 ID\_BOOKS, 356  
 ID\_CATEGORY, 356  
 ID\_CODES, 359  
 ID\_CURRENCY, 359  
 ID\_DATAFILE, 359  
 ID\_FORMATS, 361  
 ID\_FREQUENCY, 363  
 ID\_GROUP, 364  
 ID\_HAPP, 365  
 ID\_ICSET, 365  
 ID\_INTCODET, 366  
 ID\_JOURNAL\_DETAIL, 366  
 ID\_JOURNAL\_HISTORY, 367  
 ID\_JOURNAL\_HISTORY\_DETAIL, 369  
 ID\_JOURNAL\_PERIOD\_INFO, 370  
 ID\_JOURNAL\_TEMPLATES, 371  
 ID\_JOURNAL\_TEMPLATES\_DETAIL, 373  
 ID\_JOURNALS, 374  
 ID\_LOGIC, 377  
 ID\_NAMECONVERT, 378  
 ID\_NAMECVTLIST, 378  
 ID\_NAMELIST, 378  
 ID\_NAMELISTENTRY, 379  
 ID\_NAMES, 380  
 ID\_NODES, 384  
 ID\_ORGANIZATION, 386  
 ID\_PERIOD, 386  
 ID\_PRINT, 387  
 ID\_PSFDATA, 387  
 ID\_REPORT\_ENTRIES, 388  
 ID\_REPORT\_SETS, 389  
 ID\_REPORTS, 389  
 ID\_ROLLOVER, 389  
 ID\_ROLLSET, 389  
 ID\_RPTFREQ, 390  
 ID\_RPTVIEW, 390  
 ID\_RULES, 391  
 ID\_RULESEXP, 391  
 ID\_RULESVAR, 392  
 ID\_SCHEDULES, 393  
 ID\_SECCLASS, 394  
 ID\_SECGRPTAB, 394  
 ID\_SECRIGHTS, 394  
 ID\_SECTASK, 395  
 ID\_SECTASKFILTER, 395  
 ID\_SECUSERTAB, 395

ID\_SERVER, 396  
 ID\_SHARES, 396  
 ID\_SUBACCTDET, 398  
 ID\_SUBACCTHDR, 399  
 ID\_SUBNAME, 399  
 ID\_SUBSTRUCTURE, 400  
 ID\_SUGGEST\_OWN, 401  
 ID\_USERDEFAULT, 402  
 ID\_USERDEFFUNC, 405  
 INI ファイル  
   高度な関数の使用, 125  
   数値の書き込み, 279  
   文字列の書き込み, 280  
 NONE  
   戻りコードに定義された, 20  
 Pascal, 16  
 PSF データ表, 387  
 SpyWorks, 18  
 string から double への変換, 139  
 TOOLINC.H  
   インクルードファイル, 19  
 TOOLKIT.BAS  
   インクルードファイル, 19  
 TOOLKIT.H  
   インクルードファイル, 19  
 TRUE  
   定義, 17  
 vbNULLString, 17  
 Visual Basic, 16  
   32 ビット関数の宣言, 17  
   プログラミング上の注意, 16

## あ行

アクティブモジュールの設定, 264  
 値  
   派生の算出, 41  
   保存, 51  
   優劣の算出, 36  
   優劣比率の算出, 55  
 アップグレード  
   Hyperion Enterprise SE から Hyperion Enterprise へ, 291  
   HypFind(), 298  
   HypLock(), 298  
   HypQryAcctListEntry()関数, 304  
   HypQryNameListEntry()関数, 304

HypQryNode()関数, 303  
 HypQryNode()クエリサイズ, 303  
 HypQuery()関数, 298  
 HypQuery()クエリサイズ, 300  
 HypQuerySig()関数, 302  
 HypUnlock(), 298  
 会社間表, 307  
 会社間表の HypQuery(), 307  
   関数, 297  
 アドレスの渡し方, 18  
 アプリケーション  
   ENTCLOSEAPPLICATION()での終了, 143  
   ENTCREATEAPPLICATION()での作成, 145  
   ENTENUMAPPLICATIONS()を使用した列挙, 182  
   ENTOPENAPPLICATION()で開く, 242  
   ENTSERVEROPENAPPLICATION()で開く, 243  
   HYPCONSTRUCT()で開く, 29  
   HYPDESTRUCT()での終了, 31  
   デフォルトの選択, 63  
   デフォルトのハンドルの取得, 64  
   開く, 110  
   複数を閉じる, 65  
   列挙に HYPENUMAPPLICATIONS()を使用, 87  
   ログオン, 67  
 アプリケーション.INI ファイル  
   数値の取得, 196  
   テキスト文字列の読み取り, 197  
 アプリケーション情報表, 365  
 アプリケーションデフォルト表, 352  
 アプリケーションのデフォルト  
   クエリ, 248  
   保存, 251  
 アプリケーションハンドル, 201  
   再定義, 292  
   取得, 284  
   複数の取得, 66  
 アルファベット順リファレンス  
   表関数, 82  
 インクルードファイル, 19, 287  
   TOOLKIT.BAS, 19  
   TOOLKIT.H, 19  
 印刷表, 387  
 インポートライブラリ  
   プログラムをリンク, 20

エンティティ  
 エンティティ一覧から列挙, 90  
 親かどうかをチェック, 102  
 親であるかどうかの確認, 210  
 記号の取得, 98, 200  
 選択, 67, 166  
 組織内から列挙, 91  
 デフォルトの取得, 50  
 エンティティ一覧  
 エンティティの列挙, 90  
 選択, 68, 167  
 エンティティ一覧入力表, 379  
 エンティティ一覧表, 378  
 エンティティ記号, 200  
 エンティティコード, 41  
 エンティティ表, 380  
 エンティティ変換一覧表, 378  
 エンティティ変換表, 378  
 エンティティラベル, 52  
 置き換え  
 32 ビット分割勘定科目記号, 306  
 親エンティティ  
 エンティティのチェック, 102

## か行

下位構造表, 400  
 会社間照合セット表, 365  
 会社間詳細表, 366  
 詳細レコードの取得, 308  
 変更, 307  
 会社間表の詳細レコードの取得, 308  
 書き込み  
 アプリケーション.INI ファイルの数値, 279  
 アプリケーション.INI ファイルの文字列,  
 280  
 アプリケーション HYPENT.INI ファイルの数  
 値, 281  
 アプリケーション NYPENT.INI ファイルの文  
 字列, 281  
 データバッファ, 39  
 カスタム関数表, 405  
 株式表, 396  
 勘定科目  
 ENTACCTASK( )での選択, 127  
 HYPACCTASK( )での選択, 27  
 勘定科目一覧から列挙, 85  
 デフォルト名の取得, 50

入力勘定科目であるかどうかの確認, 210  
 勘定科目一覧  
 ENTACCTLISTASK( )での選択, 128  
 HYPACCTLISTASKEX( )での選択, 28  
 勘定科目の列挙, 85  
 勘定科目一覧入力表, 350  
 勘定科目一覧表, 350  
 勘定科目記号  
 分割, 129  
 勘定科目グループ表, 364  
 勘定科目種別  
 EntGetAccountsInputType( )での取得, 193  
 勘定科目値  
 HypHPVAL2( )を使用した取得, 59  
 通常の取得, 58  
 特殊な取得, 61  
 [勘定科目の選択] ダイアログボックス, 27  
 勘定科目表, 346  
 勘定科目変換一覧表, 349  
 勘定科目変換表, 349  
 勘定科目見出し  
 取得, 46  
 勘定科目ラベル, 35  
 関数  
 Ent...(), 289  
 EntAcctAsk( ), 127  
 EntAcctListAsk( ), 128  
 EntAcctSplit( ), 129  
 EntAppendToCBChain( ), 130  
 EntAsciiToDouble( ), 139  
 EntCatAsk( ), 140  
 EntCatGetNumPeriods( ), 140  
 EntCatGetPerShort( ), 141  
 EntCatMapPeriod( ), 142  
 EntCloseApplication( ), 143  
 EntConsolidate( ), 143  
 EntCreateApplication( ), 145  
 EntDataExtract( ), 149  
 EntDataFileOpen( ), 155  
 EntDataLoad( ), 270  
 EntDataLoadVB( ), 272  
 EntDateConv( ), 157  
 EntDeleteFromCBChain( ), 157  
 EntDiscardChanges( ), 158  
 EntDiscardDefault( ), 159  
 EntEntityAsk( ), 166  
 EntEntityListAsk( ), 167

- EntEnum( ), 171
- EntEnumApplications( ), 182
- EntEnumOrgEntities( ), 183
- EntEnumSubAcctSig( ), 184
- EntEnumUsersOnSystem( ), 186
- EntFind( ), 186
- EntFindEntityInOrg( ), 190, 191
- EntFormatNumber( ), 192
- EntFreqAsk( ), 193
- EntGetActiveModule( ), 196
- EntGetAppProfileLong( ), 196
- EntGetAppProfileString( ), 197
- EntGetCatPerFreq( ), 198
- EntGetChild( ), 199, 200
- EntGetEntitySig( ), 200
- EntGetHappFromSelect( ), 201
- EntGetLastError( ), 202
- EntGetOrgLevel( ), 204
- EntGetPerView( ), 204
- EntGetProfileLong( ), 205
- EntGetProfileString( ), 206
- EntGetRightsToTask( ), 207
- EntGetSibling( ), 207, 208
- EntGetVarAddr( ), 208
- EntInitApiStruct( ), 209
- EntIsAccountInput( ), 210
- EntIsEntityParent( ), 210
- EntIsModified( ), 211
- EntIsSelected( ), 212
- EntLogicAttach( ), 225
- EntLogicDetach( ), 227
- EntLogicDiscard( ), 228
- EntOpenApplication( ), 242
- EntOpenServerApplication( ), 243
- EntPerAsk( ), 245
- EntQryRptFreq( ), 246
- EntQuery( ), 247
- EntQueryDefault( ), 248
- EntRegisterMsgBox( ), 249
- EntRunRollover( ), 250
- EntSave( ), 250
- EntSaveDefault( ), 251
- EntSelect( ), 258
- EntSelectAdd( ), 260
- EntSelectTable( ), 261
- EntSelectTableAdd( ), 262, 263
- EntSetActiveModule( ), 264
- EntUnSelect( ), 276
- EntUpdate( ), 277
- EntUpdateDefault( ), 278
- EntVBCopyData( ), 283
- EntVBCopyStr( ), 283
- EntVBGetCStrLen( ), 282
- EntWriteAppProfileLong( ), 279
- EntWriteAppProfileString( ), 280
- EntWriteProfileLong( ), 281
- EntWriteProfileString( ), 281
- HypEnum( ), 307
- HypFind( ), 298
- HypGethApp( ), 284
- HypGethSelect( ), 284
- HypQryAcctListEntry(), 304
- HypQryNameListEntry(), 304
- HypQryNode(), 303
- HypQuery( ), 298
- HypQuerySig(), 302
- HypSethSelect( ), 285
- HypUnlock(), 307
- アップグレード, 297
- 以前のバージョン, 289
- スプレッドシート用, 288
- 登録, 249
- 古い関数の置き換え, 295
- 分類, 288
- リリース 4 でサポートされていない, 290
- 関数の一覧, 24, 25, 26
- 関連する表
  - 高度な関数, 113
  - 高レベル関数と Hyp... ( )関数, 340
- 関連付けられている表, 339
- 期間
  - EntPerAsk( )での選択, 245
  - HypCatMapPeriodEx( )を使用して期間単位にマッピング, 84
  - HypPerAsk( )での選択, 69
  - 期間単位へのマッピング, 142
  - デフォルト名の取得, 50
- 期間ごとの組織, 120
- 期間単位
  - EntFreqAsk( )での選択, 193
  - EntGetPerView( )での取得, 204
  - HypFreqAsk( )での選択, 32
- 期間単位と表示形式, 20
- HypGetPerViewEx( )で取得, 100

- 期間単位表, 363
- 期間表, 386
- 期間ラベル
  - 取得, 141
  - データ種別期間単位内で取得, 84
- 期間を期間単位にマッピング, 84
- 記号
  - 検索, 94, 186
  - サブ勘定科目の列挙, 93
  - ノードの検索, 95
- 期別替
  - 実行, 250
- 期別替セット表, 389
- 期別替表, 389
  - 詳細レコードの取得, 311
  - 変更, 309
- 期別替表の詳細レコードの取得, 311
- 兄弟ノード, 101, 207
- クエリ
  - Hyperion Enterprise の表, 105
  - アプリケーションのデフォルト, 248
  - 表内のレコード, 247
  - ユーザのデフォルト, 248
  - レポートの期間単位, 105
- クエリ属性
  - ID\_ACCOUNTS, 346
  - ID\_ACCTCONVERT, 349
  - ID\_ACCTCVTLIST, 349
  - ID\_ACCTLIST, 350
  - ID\_ACCTLISTENTRY, 350
  - ID\_APPDEFAULT, 352
  - ID\_BOOK\_ENTRIES, 355
  - ID\_BOOK\_SETS, 355
  - ID\_BOOKS, 356
  - ID\_CATEGORY, 356
  - ID\_CODES, 359
  - ID\_CURRENCY, 359
  - ID\_DATAFILE, 359
  - ID\_FORMATS, 361
  - ID\_FREQUENCY, 363
  - ID\_GROUP, 364
  - ID\_HAPP, 365
  - ID\_ICSET, 365
  - ID\_INTCODET, 366
  - ID\_JOURNAL\_DETAIL, 366
  - ID\_JOURNAL\_HISTORY, 367
  - ID\_JOURNAL\_HISTORY\_DETAIL, 369
  - ID\_JOURNAL\_PERIOD\_INFO, 370
  - ID\_JOURNAL\_TEMPLATES, 371
  - ID\_JOURNAL\_TEMPLATES\_DETAIL, 373
  - ID\_JOURNALS, 374
  - ID\_LOGIC, 377
  - ID\_NAMECONVERT, 378
  - ID\_NAMECVTLIST, 378
  - ID\_NAMELIST, 378
  - ID\_NAMELISTENTRY, 379
  - ID\_NAMES, 380
  - ID\_NODES, 384
  - ID\_ORGANIZATION, 386
  - ID\_PERIOD, 386
  - ID\_PRINT, 387
  - ID\_PSFDATA, 387
  - ID\_REPORT\_ENTRIES, 388
  - ID\_REPORT\_SETS, 389
  - ID\_REPORTS, 389
  - ID\_ROLLOVER, 389
  - ID\_ROLLSET, 389
  - ID\_RPTFREQ, 390
  - ID\_RPTVIEW, 390
  - ID\_RULES, 391
  - ID\_RULESEXP, 391
  - ID\_SCHEDULES, 393
  - ID\_SECCLASS, 394
  - ID\_SECGRPTAB, 394
  - ID\_SECRIGHTS, 394
  - ID\_SECTASK, 395
  - ID\_SECTASKFILTER, 395
  - ID\_SECUSERTAB, 395
  - ID\_SERVER, 396
  - ID\_SHARES, 396
  - ID\_SUBACCTDET, 398
  - ID\_SUBACCTHDR, 399
  - ID\_SUBNAME, 399
  - ID\_SUBSTRUCTURE, 400
  - ID\_SUGGEST\_OWN, 401
  - ID\_USERDEFAULT, 402
  - ID\_USERDEFFUNC, 405
  - 計算式表, 362
  - 使用されなくなった, 299
  - 使用ロジック表, 401
  - デフォルト, 344
  - データ種別リンク表, 359
  - 変数規則表, 392
  - ロジックデータ種別属性表, 377

言語, サポートされている, 16

更新

デフォルト, 278

データ, 117

レコード, 277

高度な関数

Hyperion Enterprise INI ファイルの使用, 125

アプリケーションを開く, 110

アルファベット順リファレンス, 127

関連する関数, 113

スプレッドシートのアドイン関数との組み合わせ, 113

選択ダイアログボックス, 114

組織構造体, 120

データ操作, 114

データの更新, 117

データの取得, 115

表, 110

表の使用, 112

表の選択, 111

表の選択解除, 111

連結詳細値, 25, 119

子ノード, 199

HYPGETCHILD()での取得, 97

コピー

データ, 283

文字列, 283

コンポーネントのエンティティに関する完全な説明, 45

コンポーネントの通貨, 40

コード表, 359

コールバック関数, 18

CALLBACK12, 318

CALLBACK6, 320

CALLBACKAPI, 319

CALLBACKAPP, 320

CALLBACKCREATE, 320

CALLBACKDBENUM, 321

CALLBACKDBLOAD, 322, 323, 324

CALLBACKJOUREXTRACT, 325

CALLBACKJOURLOAD, 326

CALLBACKLOGIC, 326

CALLBACKSECLOAD, 327

CALLBACKSEL, 328

CALLBACKSTR, 330

CALLBACKUSERS, 330

CALLBACKVB, 331

Visual Basic での, 18

コールバックチェーン

削除, 157

追加, 130

## さ行

最後の値取得

戻りステータス, 76

最後のエラー

取得, 202

差異率

算出, 56

サブエンティティ表, 399

サブ勘定科目

EntEnumSubAcctSig()を使用した列挙, 184

HypEnumSubAcctSig()での記号の列挙, 93

サブ勘定科目詳細表, 398

サブ勘定科目ヘッダー表, 399

サブ勘定科目ラベル, 35

サポートされていない関数, 290

サポートされている言語, 16

算出

差異率, 56

データ種別内の期間数, 83

算出勘定の判別, 37

サーバアプリケーション

開く, 243

サーバ表, 396

収益勘定科目の判別, 47

出資比率

情報の取得, 53

取得

アプリケーション.INI ファイルの数値, 196

アプリケーションハンドル, 201, 284

エンティティ記号, 98

エンティティコード, 41

勘定科目見出し, 46

勘定科目ラベルまたはサブ勘定科目ラベル, 35

期間番号と期間単位, 198

兄弟ノード, 101

子ノード, 97

コンポーネントのエンティティに関する完全な説明, 45

コンポーネントの通貨, 40

単位, 57

通貨の説明, 36

- デフォルト仕訳帳, 33
- デフォルト仕訳帳期間, 34
- デフォルト仕訳帳データ種別, 34
- デフォルトのラベル, 50
- データ, 115
- データ種別期間単位, 44
- データ種別内の期間単位, 96
- データ種別内の期間番号, 96
- データ種別の説明, 38
- 特殊な勘定科目値, 61
- 表選択ハンドル, 284
- 詳細記憶モデル
  - 定義, 25, 119
- 初期化
  - apiStruct, 209
- 書式表, 361
- 所有株数
  - 情報の取得, 53
- 使用されなくなった Ent...()関数
  - 新しい Ent...()関数への置き換え, 127
- 使用されなくなったクエリ属性, 299
- 使用されなくなったセキュリティアクセスグループ表, 311
- 使用されなくなったセキュリティアクセス詳細表, 312
- 使用されなくなったセキュリティ表, 311
  - セキュリティアクセスグループ, 311
  - セキュリティアクセス詳細, 312
- 仕訳帳
  - 詳細情報の取得, 49
  - 詳細の選択, 63
  - 選択, 62
  - デフォルトの設定, 70
- 仕訳帳記録表, 367
- 仕訳帳詳細表, 366
- 仕訳帳テンプレートの詳細表, 373
- 仕訳帳テンプレート表, 371
- 仕訳帳の期間情報表, 370
- 仕訳帳の記録の詳細表, 369
- 仕訳帳の詳細情報, 49
- 仕訳帳表, 374
- 数式規則表, 391
- 数値
  - EntFormatNumber()による書式設定, 192
  - HYPENT.INI ファイルでの取得, 205
  - HYPENT.INI ファイルへの書き込み, 281
- アプリケーション.INI ファイルへの書き込み, 279
- スプレッドシート
  - 関数, 288
- スプレッドシートのアドイン関数
  - 高度な関数との組み合わせ, 113
  - 選択ダイアログボックスの使用, 24
  - 単一アプリケーションの管理, 23
  - データの更新, 26
  - データの取得, 25
  - 複数アプリケーションの管理, 24
- スプレッドシートのアドイン関数と高度な関数の組み合わせ, 113
- セキュリティクラス表, 394
- セキュリティグループ表, 394
  - 変更, 315
- セキュリティ権限
  - 現在のユーザに関する取得, 207
- セキュリティ権表, 394
- セキュリティタスクのコード, 312
- セキュリティタスク表, 395
- セキュリティタスクフィルタ表, 395
- セキュリティユーザ表, 395
  - 変更, 315
- 設定
  - HypHPVal()の戻り値動作, 72
  - デフォルト仕訳帳, 70
  - デフォルト仕訳帳期間, 71
  - デフォルト仕訳帳データ種別, 70
  - 表選択ハンドル, 285
- 選択
  - エンティティ, 67
  - エンティティ一覧, 68
  - 勘定科目, 27
  - 勘定科目一覧, 28
  - 期間, 69
  - 期間単位, 32
  - 仕訳帳, 62
  - 仕訳帳の詳細, 63
  - 追加の表, 260, 262
  - データ種別, 28
  - 表, 103, 258, 261
- 選択解除
  - 表, 276
- 選択ダイアログボックス
  - 高度な関数, 114
  - スプレッドシートのアドイン関数, 24

選択のコールバック, 112

組織

兄弟ノード, 207

子ノード, 199

トップノードの ID の取得, 101

ノードを列挙, 91

レベルの取得, 99

組織構造体

高度な関数, 120

と Hyp...()関数, 81

組織表, 386

## た行

単位

取得, 57

単一アプリケーション

スプレッドシートのアドイン関数によって開く, 23

スプレッドシートのアドイン関数による管理, 23

単一アプリケーションの管理, 23

チェック

パラメータ, 74

リンクパラメータ, 73

通貨

説明, 36

通貨表, 359

通貨ラベル, 40

定数

visual basic での, 17

デフォルト仕訳帳, 33

デフォルト仕訳帳期間, 34

デフォルト仕訳帳データ種別, 34

デフォルト設定

表, 338

変更の破棄, 159

デフォルトのアプリケーション

選択, 63

デフォルトのアプリケーションハンドル, 64

デフォルトのクエリ属性, 344

データ

高度な関数による取得, 115

高度な関数を使用した更新, 117

保存, 51

読み込み, 270, 272

データ更新関数, 26

データ取得関数, 25

データ種別

ENTCATASK()での選択, 140

HYPCATASKEX()での選択, 28

期間数の算出, 83

期間単位, 44

期間と期間単位の取得, 198

期間番号と期間単位の取得, 96

説明, 38

デフォルト名の取得, 50

データ種別の期間数の計算, 140

データ種別表, 356

データ値の保存, 51

データ入力表, 393

データの更新

スプレッドシートのアドイン関数, 26

データの取得

スプレッドシートのアドイン関数, 25

データの操作

高度な関数の使用, 114

データバッファ, 39

フラッシュ, 42

データバッファ, HYPPCOMMIT2()での書き込み, 39

データバッファのフラッシュ, 42

データファイル

ENTDATAFILEOPEN()で開く, 155

データファイル表, 359

閉じる

アプリケーション, 31

トップノード ID, 101

## な行

名前が変更された表, 306

入力勘定科目の判別, 47

ノード

EntEnumOrgEntities()を使用した列挙, 183

HypEnumOrgNames()で列挙, 91

ノード ID

EntFindEntityInOrg()での検索, 190

ノードの記号

HypFindNameInOrgEx()で取得, 95

ノード表, 384

変更, 308

## は行

破棄, ロジック, 228

派生値, 41

- ハンドル
  - 再定義, 292
- バッファ
  - データ, 39
- パッケージセット表, 355
- パッケージ入力表, 355
- パッケージ表, 356
- パラメータ
  - チェック, 74
- 非ゼロ
  - 定義, 20
- 日付の変換, 82
  - ENTDATECONV( ), 157
- 表
  - apiStruct の使用, 110
  - EntSave( )での保存, 250
  - EntSelect( )での選択, 258
  - EntSelectTable( )での選択, 261
  - EntUnselect( )での選択解除, 276
  - Hyp...()関数で使用, 80
  - Hyp...()関数を使用した選択, 79
  - Hyp...()関数を使用した選択解除, 79
  - Hyperion Enterprise SE から Hyperion Enterprise への変換の変更, 306
  - 関連付けられている, 339
  - 高度な関数, 110
  - 選択, 103, 111
  - 選択解除, 106, 111
  - 選択されているかどうかの確認, 212
  - 選択のコールバック, 112
  - 追加の選択, 260, 262
  - デフォルト設定の, 338
  - 変更されているかどうかの確認, 211
  - 変更の破棄, 158
  - リリース 4 で名前変更, 306
  - レコードのクエリ, 247
  - レコードの列挙, 88, 171
- 表関数
  - アルファベット順リファレンス, 82
  - スプレッドシートのアドイン関数の組み合わせ, 113
  - 選択, 79
  - 選択解除, 79
- 表示形式
  - EntGetPerView( )での取得, 204
- 表選択ハンドル
  - 取得, 284
- 設定, 285
- 表の使用
  - Hyp...()関数で, 80
  - 高度な関数, 112
- 表の選択
  - Hyp...()関数の使用, 79
  - 高度な関数の使用, 111
- 表の選択解除, 103, 106
- 開く
  - アプリケーション, 29
- 複数アプリケーション
  - 管理, 24
  - スプレッドシートのアドイン関数によって開く, 24
  - スプレッドシートのアドイン関数による管理, 24
  - 閉じる, 65
  - ログオン, 67
- 複数アプリケーションからのログオフ, 65
- 複数アプリケーションへのログオン, 67
- 古い関数の置き換え, 293
- フロー勘定科目, 43
- フロー勘定科目の判別, 43
- 分離, ロジック, 227
- 分類
  - 関数, 288
- プログラム
  - インポートライブラリへのリンク, 20
- プログラム名
  - EntGetActiveModule( )での取得, 196
- ヘッダーファイル, 287
- 変換
  - 16 ビットプログラムから 32 ビットプログラムへ, 292
- 変更の破棄
  - デフォルト設定, 159
  - 表内, 158
- 変数
  - アドレスの取得, 208
- 保存
  - デフォルト, 251
  - 表, 250
- ま行**
  - メッセージボックス
    - 登録, 249
  - 文字列, 17

double への変換, 139  
 HYPENT.INI ファイルでの取得, 206  
 HYPENT.INI ファイルへの書き込み, 281  
 Visual Basic での, 17  
 アプリケーション.INI ファイルへの書き込み, 280  
 戻りコードでの, 20

## や行

優劣

値, 36  
 比率, 55

優劣比率, 55

ユーザ

列挙に EntEnumUsersOnSystem( )を使用, 186

ユーザデフォルト表, 402

ユーザのデフォルト

クエリ, 248  
 保存, 251

クエリ, 246  
 レポートの表示形式, 20  
 レポート表, 389  
 レポート表示形式表, 390  
 連結  
     ENTCONSOLIDATE( )の使用, 143  
 連結詳細値, 25, 119  
 連結ロジックと連結比率の提案表, 401  
 ログインユーザレポート  
     プログラム名の設定, 72  
 ロジック  
     追加, 225  
     破棄, 228  
     分離, 227  
 ロジック表, 377

## ら行

リンクパラメータ

チェック, 73

ルール表, 391

レコード

列挙, 171

列挙

EntEnumApplications( )でのアプリケーション, 182

HypEnumAcctListEntriesEx( )での勘定科目, 85

アプリケーション, 87

エンティティを

HypEnumNameListEntriesEx( )で, 90

サブ勘定科目, 184

サブ勘定科目記号, 93

システム上のユーザ, 186

組織内のノード, 183

表内のレコード, 88, 171

開いているアプリケーション, 65

レベル

組織内での取得, 204

レポート期間単位表, 390

レポートセット表, 389

レポート入力表, 388

レポートの期間単位, 20

期間単位と表示形式, 204

