

MAID 3.1

Revision17J

目次

1	著者	5
2	コンセプト	エラー! ブックマークが定義されていません。
2.1	大きな画像	5
2.2	概要	5
2.3	オブジェクトの寿命	6
2.4	2つの視野	6
3	使用法	7
3.1	モジュールのロード	7
3.2	モジュールの初期化	7
3.3	capability の列挙	8
3.4	配列 capability の読み込み	8
3.5	範囲 capability の使用	10
3.6	capability グループ	10
3.7	ベンダー固有 capability の使用	10
3.8	ソースおよびアイテムのオープン	10
3.9	データオブジェクトを開く	11
3.10	データ転送	11
3.11	状態の保存と読み込み	12
3.12	イベントの通知	13
3.13	ユーザインターフェイスの要求	13
3.14	非同期によるモジュールの呼び出し	13
3.15	モジュールの終了	13
3.16	モジュールの解放	14
4	一覧表	14
4.1	戻り値	14
4.2	データオブジェクト型	14
4.3	データ型	15
4.4	配列型	15
4.5	機能型	15
4.6	機能操作	16
4.7	機能の可視性	16
4.8	オブジェクト型	16
4.9	イベント	16
4.10	ユーザインターフェイス要求型	17
4.11	ユーザインターフェイス値	17
4.12	フィルタ	17
4.13	コマンド	18
4.14	Capabilities	19
4.15	色空間	20
4.16	Boolean Default	20
4.17	Module Types	20
4.18	File Data Types	20
4.19	Flash Modes	21
5	構造体と型	21
5.1	Word 値	21
5.2	Unsigned Long 値	21
5.3	パラメータ値	21
5.4	ポインタ値	21
5.5	参照値	21
5.6	MAID Entry Point 関数ポインタ	21
5.7	MAID 完了関数ポインタ	22
5.8	MAID データ引き渡し関数ポインタ値	22
5.9	MAID イベント通知関数ポインタ	22
5.10	MAID 処理通知関数ポインタ	22
5.11	MAID ユーザインターフェイス要求関数ポインタ	22
5.12	コールバック定義構造体	22
5.13	Date/Time 構造体	22

5.14	ポイント構造体	22
5.15	Size 構造体	23
5.16	矩形構造体	23
5.17	文字列構造体	23
5.18	配列構造体	23
5.19	範囲構造体	24
5.20	機能情報構造体	24
5.21	オブジェクト構造体	24
5.22	ユーザインターフェイス要求構造体	25
5.23	標準データ配送構造体	26
5.24	イメージデータ配送構造体	26
5.25	音声データ配送構造体	27
5.26	列挙構造体	27
5.27	ファイルデータ配送構造体	28
6	戻り値	28
6.1	kNkMAIDResult_NotSupported	29
6.2	kNkMAIDResult_UnexpectedDataType	29
6.3	kNkMAIDResult_ValueOutOfBounds	29
6.4	kNkMAIDResult_BufferSize	29
6.5	kNkMAIDResult_Aborted	29
6.6	kNkMAIDResult_NoMedia	29
6.7	kNkMAIDResult_NoEventProc	29
6.8	kNkMAIDResult_ZombieObject	29
6.9	kNkMAIDResult_NoError	29
6.10	kNkMAIDResult_Pending	30
6.11	kNkMAIDResult_OrphanedChildren	30
6.12	kNkMAIDResult_NoDataProc	30
6.13	kNkMAIDResult_OutOfMemory	30
6.14	kNkMAIDResult_UnexpectedError	30
6.15	kNkMAIDResult_HardwareError	30
6.16	kNkMAIDResult_MissingComponent	30
7	イベント	30
7.1	kNkMAIDEvent_AddChild	30
7.2	kNkMAIDEvent_RemoveChild	31
7.3	kNkMAIDEvent_WarmingUp	31
7.4	kNkMAIDEvent_WarmedUp	31
7.5	kNkMAIDEvent_CapChange	31
7.6	kNkMAIDEvent_OrphanedChildren	32
7.7	kNkMAIDEvent_CapChangeValueOnly	32
8	コマンド	32
8.1	kNkMAIDCommand_Async	33
8.2	kNkMAIDCommand_Open	33
8.3	kNkMAIDCommand_Close	34
8.4	kNkMAIDCommand_GetCapCount	34
8.5	kNkMAIDCommand_GetCapInfo	34
8.6	kNkMAIDCommand_CapStart	35
8.7	kNkMAIDCommand_CapSet	35
8.8	kNkMAIDCommand_CapGet	35
8.9	kNkMAIDCommand_CapGetDefault	36
8.10	kNkMAIDCommand_CapGetArray	37
8.11	kNkMAIDCommand_Mark	37
8.12	kNkMAIDCommand_AbortToMark	37
8.13	kNkMAIDCommand_Abort	38
8.14	kNkMAIDCommand_EnumChildren	38
8.15	kNkMAIDCommand_GetParent	38
8.16	kNkMAIDCommand_ResetToDefault	38
9	Capabilities	39
9.1	kNkMAIDCapability_AsyncRate	39
9.2	kNkMAIDCapability_ProgressProc	39
9.3	kNkMAIDCapability_EventProc	40
9.4	kNkMAIDCapability_DataProc	40
9.5	kNkMAIDCapability_UIRequestProc	40
9.6	kNkMADCapability_IsAlive	41

9.7	kNkMAIDCapability_Children	41
9.8	kNkMAIDCapability_State	41
9.9	kNkMAIDCapability_Name	41
9.10	kNkMAIDCapability_Description	41
9.11	kNkMAIDCapability_Interface	42
9.12	kNkMAIDCapability_DataTypes	42
9.13	kNkMAIDCapability_DateTime	42
9.14	kNkMAIDCapability_StoredBytes	42
9.15	kNkMAIDCapability_Eject	42
9.16	kNkMAIDCapability_Feed	43
9.17	kNkMAIDCapability_Capture	43
9.18	kNkMAIDCapability_Mode	43
9.19	kNkMAIDCapability_Acquire	43
9.20	kNkMAIDCapability_Start	43
9.21	kNkMAIDCapability_Length	44
9.22	kNkMAIDCapability_SampleRate	44
9.23	kNkMAIDCapability_Stereo	44
9.24	kNkMAIDCapability_Samples	44
9.25	kNkMAIDCapability_Filter	44
9.26	kNkMAIDCapability_Prescan	44
9.27	kNkMAIDCapability_AutoFocus	45
9.28	kNkMAIDCapability_AutoFocusPt	45
9.29	kNkMAIDCapability_Focus	45
9.30	kNkMAIDCapability_Coords	45
9.31	kNkMAIDCapability_Resolution	45
9.32	kNkMAIDCapability_Preview	45
9.33	kNkMAIDCapability_Negative	46
9.34	kNkMAIDCapability_ColorSpace	46
9.35	kNkMAIDCapability_Bits	46
9.36	kNkMAIDCapability_Planar	46
9.37	kNkMAIDCapability_Lut	46
9.38	kNkMAIDCapability_Transparency	47
9.39	kNkMAIDCapability_Threshold	47
9.40	kNkMAIDCapability_Pixels	47
9.41	kNkMAIDCapability_ForceScan	47
9.42	kNkMAIDCapability_ForcePrescan	47
9.43	kNkMAIDCapability_ForceAutoFocus	48
9.44	kNkMAIDCapability_NegativeDefault	48
9.45	kNkMAIDCapability_Firmware	48
9.46	kNkMAIDCapability_CommunicationLevel1	48
9.47	kNkMAIDCapability_CommunicationLevel2	49
9.48	kNkMAIDCapability_BatteryLevel	49
9.49	kNkMAIDCapability_FreeBytes	49
9.50	kNkMAIDCapability_FreeItems	49
9.51	kNkMAIDCapability_Remove	50
9.52	kNkMAIDCapability_FlashMode	50
9.53	kNkMAIDCapability_ModuleType	50
9.54	kNkMAIDCapability_AcquireStreamStart	50
9.55	kNkMAIDCapability_AcquireStreamStop	50
9.56	kNkMAIDCapability_AcceptDiskAcquisition	51
9.57	kNkMAIDCapability_Version	51
9.58	kNkMAIDCapability_FilmFormat	51
9.59	kNkMAIDCapability_TotalBytes	52
10	関数定義	52
10.1	MAID エントリポイント関数	52
10.2	MAID 完了関数	52
10.3	MAID データ転送関数	52
10.4	MAID イベント通知関数	53
10.5	MAID 進行通知関数	53
10.6	MAID ユーザインターフェイス要求関数	53
11	変更履歴	53

1 著者

この日本語訳は参考のためのものであり、正式にはかならず英文のものを使用すること。

2 コンセプト

この章では、MAID における考え方を説明する。この章では、強調文字はこのドキュメントを通じて使用される用語の導入を意味している。

2.1 大きな画像

MAID 規約は、イメージ、サウンドもしくはビデオを持つデバイスのためのデバイスドライバとアプリケーション間のインターフェイス層を提供するために作成された。MAID においては、アプリケーション側は**クライアント**として参照され、デバイスドライバ側は**モジュール**として参照される。クライアントは、すべてのユーザインターフェイスを提供し、モジュールはデバイスとの通信のすべてを提供する。

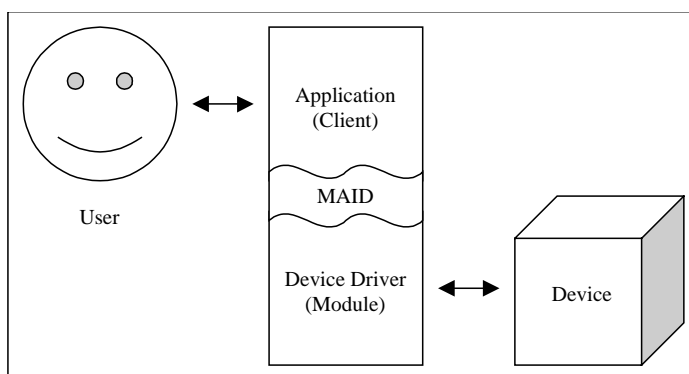


Figure 1

2.2 概要

MAID はデバイスとの通信を層に抽象化する。それぞれの層では、クライアントはモジュールに対してビューを開く。このビューは**オブジェクト**と呼ばれる。これはモジュール自身を意味する。次は**ソースオブジェクト**であり、これは物理的なデバイスを意味する。次は**アイテムオブジェクト**であり、もっとも深いレベルのオブジェクト、すなわち**データオブジェクト**、の集まりを意味する。データオブジェクトは画像、音もしくはビデオである。アイテム中には、データオブジェクトのそれぞれの型のうちの一種類のみが存在しうる。

このドキュメントを通じて、“モジュール”はデバイスドライバを指し、“モジュールオブジェクト”はデバイスドライバの MAID の抽象化されたものとして開かれた通信路を指す。

例えば、モジュールによってサポートされている、2つの使用可能なデバイスがあるとする。1つ目のデバイスは、画像と関連付けられていない音声が入部に保存されている。2つ目のデバイスは、音声を伴った画像と、音声と関連付けられていないビデオが入部に保存されている。クライアントがモジュールオブジェクトと、これらの物理的な対象物のそれぞれに対するオブジェクトを開いたとすると、オブジェクトの階層構造は図2のようになる。

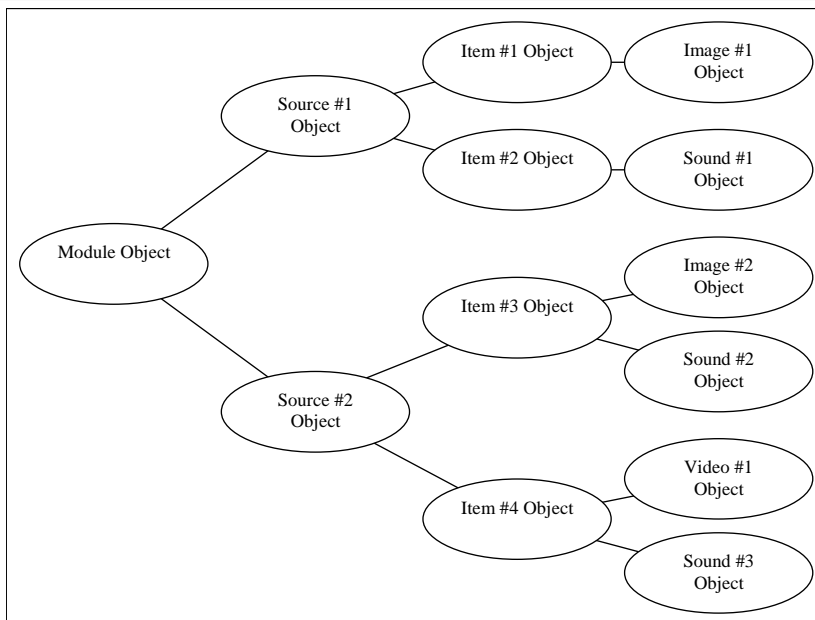


Figure 2

2.3 オブジェクトの寿命

オブジェクトは、クライアントがそれを閉じるまでの間、開かれた状態となる。クライアントはモジュールを解放する前に閉じなければならない。ソース、アイテムもしくはデータオブジェクトが開かれるときは、親オブジェクトが提供されなければならない。開かれるオブジェクトは子オブジェクトとなる。**capabilities**として知られる、すべての設定は各自のオブジェクトと関連付けられる。親オブジェクトの **capability** 設定は、すべてのその子オブジェクトに適用される。

通常、オブジェクトが開かれている間は、オブジェクトは **alive** である。クライアントがその親を閉じた場合、もしくはオブジェクトの物理的な対象物が消滅した場合は、もはや **alive** でなくなる。この **zombie** 状態でも、クライアントはオブジェクトに関するいくつかの **capabilities** を読み込むことができる。どの **capabilities** が有効かは、モジュールが親オブジェクトもしくは物理的な対象物なしで提供できる情報に依存する。

2.4 2つの視野

オブジェクトは単に物理的な対象物のビューにすぎないため、1つの物理的な対象物に対して1つを超えるオブジェクトを開くことができる。これは、MAID オブジェクト階層構造のそれぞれのレベルにおいて、真実である。例えば、クライアントがソースを2度開いたとすると、両方のソースオブジェクトは、同じ親モジュールオブジェクトを持ち、図3に示すように同じ物理デバイスに対してアクセスする。

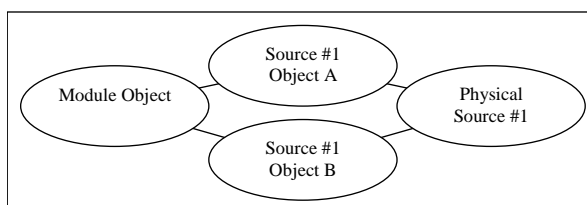


Figure 3

これらのソースオブジェクトのそれぞれは、物理オブジェクトに対する固有の設定を持つ。モジュールは、同時に物理ソースにアクセスを持つソースオブジェクトを、一つに制限する。2つのソースオブジェクトが、重要な **capability** を矛盾する値に設定した場合、モジュールのみが、もっとも良い方法でこの問題を解決する方法を、決定することができる。

この特徴は、データオブジェクトレベルにおいて最も便利である。画像型のデータにおいて、1つのオブジェクトをプレビュー要求用に使用する一方、他を最終的な要求に使用することができる。他の例としては、スキャナに複数の原点が存在するギャングスキャンがある。例として、クライアントはすべての領域のプレビューオブジェクト、3つの拡大したプレビューオブジェクトおよび3つの最終的なオブジェクトをもつことができる。これらのオブジェクトのそれぞれは、同じ物理的なスキャナから得られる、同じ物理的な画像データを参照している。もしクライアントが7つのオブジェクトを同時に要求した場合には、モジュールは明らかにそれぞれを継承してスキャンする。

3 使用法

ここで示す例については、簡単にするため全てのモジュールに対する呼び出しは同期とする。非同期操作については後に述べる。

3.1 モジュールのロード

モジュールのロードの処理はシステムに依存する。

Windows 環境では、モジュールは DLL として生成される。拡張子は MD3 であり、Common Files\MaidMods ディレクトリの下にいずれかのディレクトリに置かれる。Common Files ディレクトリの実際の名前と位置はマシンに依存しており、レジストリを参照して得ることができる。モジュールは MAID エントリーポイント用の1つの関数を用意する。クライアントはモジュールをロードするために LoadLibrary()を、MAID Entry Point のアドレスを得るために GetProcAddress()を使用する。32bit で生成する場合には、TWAIN クライアントは 0x11000000 に、モジュールは 0x12000000 もしくはそれ以上を基底アドレスとする。この推奨基底アドレスの設定により、ロード時間を短縮することができる。

Macintosh 環境では、モジュールは Fat ファイルとして構成される。そこには PowerMac で使用するコードフラグメントを構成するデータフォークが含まれる。また 68k Macintosh の動作に必要なオブジェクトコードは、リソースタイプ 'MAID' としてリソースフォークに含まれる。PowerMac においてクライアントは、GetDiskFragment()を使用し、その戻り値を MAID エントリーポイントの UPP ディスクリプタに代入する。68k Macintosh においてクライアントは、コードリソースをロードし、ハンドルをロックした後そのアドレスを MAID エントリーポイントとしてキャストする。

3.2 モジュールの初期化

クライアントにとっての最初のステップは、モジュールの構造体を持つことである。これは、構造体としての集合、オブジェクト、ヒープからの確保のいずれでもかまわない。注意すべき点は、この例では一度に一つのモジュールしか開くことはできない。これはモジュールの構造体を1つのグローバル変数としているからである。(Line1) 実際のクライアントは一度に一つ以上のモジュールを持つようになる。

クライアントは、refClient メンバを設定することにより、構造体を初期化する。(Line9) 次に object として NULL を、data としてモジュール構造体のポインタをセットして、モジュールを呼び出す。(Line12) コマンドが成功すれば、モジュールは開いている。(Line16)

```

1  NkMAIDObject objModule;
2
3  // open the module synchronously
4  BOOL InitializeMAIDModuleSync( NKREF ref, LPMAIDEntryPointProc pMAIDEntryPoint )
5  {

```

```

6     LONG nResult;
7
8     // set the reference
9     objModule.refClient = ref;
10
11    // call the module to open the module
12    nResult = (*pMAIDEntryPoint)( NULL, kNkMAIDCommand_Open, 0, kNkMAIDDataType_ObjectPtr,
13                                (NKPARAM)&objModule, NULL, 0 );
14
15    // return TRUE if the module successfully opened
16    return (nResult == kNkMAIDResult_NoError);
17 }

```

3.3 capability の列挙

一度オブジェクトが開かれると、機能が列挙される。クライアントはオブジェクトの機能の数を得るためにモジュールを呼び出す。(Line10) このコマンドが成功した場合には、クライアントは機能情報を保持するための領域を確保する。(Line16) メモリが正常に確保できた場合には、クライアントは機能情報を引き出すためにモジュールを呼び出す。(Line21) モジュールに対する 2 つの呼び出しの間に機能数が変化した場合 (Line24)、メモリは解放され (Line26) 処理が再度実行される。(Line33) コマンドが正常に終了していれば、機能は列挙されている。(Line36)

```

1  // enumerate the capabilities of an object
2  BOOL EnumerateMAIDObjectCapabilities( LPMAIDEntryPointProc pMAIDEntryPoint, LPNkMAIDObject pObject,
3  ULONG FAR *pulCapCount, LPNkMAIDCapInfo FAR * ppCapArray )
4  {
5      LONG nResult;
6
7      do
8      {
9          // call the module to get the size of the capability array
10         nResult = (*pMAIDEntryPoint)( pObject, kNkMAIDCommand_GetCapCount, 0,
11                                       kNkMAIDDataType_UnsignedPtr, pulCapCount, NULL, 0 );
12
13         if (nResult == kNkMAIDResult_NoError)
14         {
15             // allocate memory for the array
16             *ppCapArray = (LPNkMAIDCapInfo)malloc( *pulCapCount * sizeof( NkMAIDCapInfo ) );
17
18             if (*ppCapArray != NULL)
19             {
20                 // call the module to get the capability array
21                 nResult = (*pMAIDEntryPoint)( pObject, kNkMAIDCommand_GetCapInfo, *pulCapCount,
22                                               kNkMAIDDataType_CapInfoPtr, (NKPARAM)*ppCapArray, NULL, 0 );
23
24                 if (nResult == kNkMAIDResult_BufferSize)
25                 {
26                     free( *ppCapArray );
27                     *ppCapArray = NULL;
28                 }
29             }
30         }
31     }
32     // repeat the process if the number of capabilities changed between the two calls to the module
33     while (nResult == kNkMAIDResult_BufferSize);
34
35     // return TRUE if the capabilities were successfully enumerated
36     return (nResult == kNkMAIDResult_NoError);
37 }

```

3.4 配列 capability の読み込み

配列機能の読み込み処理は、機能情報の読み込みと似ている。クライアントは機能が配列型であり、*kNkMAIDCommand_CapGetArray* および *kNkMAIDCommand_CapGet* コマンドをサポートしていることを確認する (Line54)。クライアントはモジュールに対して配列機能の情報を得るための呼び出しを行う。(Line62) コマンドが正常に終了した場合には、クライアントは配列データを保持するためのメモリを確保する。(Line68) 確保に成功した場合には、クライアントは配列データを取り出すためにモジュールを呼び出す。(Line73) 配列データのサイズが、モジュールへの 2 つの呼び出しの間に変更された場合には (Line76)、メモリは解放され (Line78) 再び処理が行われる。(Line85) コマンドが正常に実行された場合は、配列は読み出されている。(Line88)

```

1  // find the capability
2  BOOL FindMAIDCapability( LPMAIDEntryPointProc pMAIDEntryPoint, LPNkMAIDObject pObject,
3  ULONG ulCapID, LPNkMAIDCapInfo pCapInfo )

```



```

4  {
5      LONG nResult;
6      BOOL fRet = FALSE;
7      ULONG ulCapCount;
8      LPNkMAIDCapInfo lpCapArray;
9
10     // make sure we don't free some memory we didn't allocate
11     lpCapArray = NULL;
12
13     // this function is in the example for capability enumeration
14     if (EnumerateMAIDObjectCapabilities( pMAIDEntryPoint, pObject, &ulCapCount, &lpCapArray))
15     {
16         // make sure we got an array
17         if (lpCapArray != NULL)
18         {
19             ULONG ulIndex;
20
21             // find the capability
22             for (ulIndex=0; ulIndex<ulCapCount; ++ulIndex)
23                 if (lpCapArray[ulIndex].ulID == ulCapID)
24                     break;
25
26             // did we find it?
27             if (ulIndex < ulCapCount)
28             {
29                 fRet = TRUE;
30                 *pCapInfo = lpCapArray[ulIndex];
31             }
32         }
33     }
34
35     // make sure to free memory allocated by EnumerateMAIDObjectCapabilities()
36     if (lpCapArray != NULL)
37         free( lpCapArray );
38
39     return fRet;
40 }
41
42 // enumerate the capabilities of an object
43 BOOL ReadMAIDArrayCapability( LPMAIDEntryPointProc pMAIDEntryPoint, LPNkMAIDObject pObject,
44     ULONG ulCapID, LPNkMAIDArray pArray )
45 {
46     LONG nResult;
47     NkMAIDCapInfo capInfo;
48
49     // get the capability information
50     if (!FindMAIDCapability( pMAIDEntryPoint, pObject, ulCapID, &capInfo ))
51         return FALSE;
52
53     // the capability must be an array, and must support the CapGetArray and CapGet commands
54     if (capInfo->ulType != kNkMAIDCapType_Array ||
55         ! (capInfo-> ulOperations & kNkMAIDCapOperation_GetArray) ||
56         ! (capInfo-> ulOperations & kNkMAIDCapOperation_Get))
57         return FALSE;
58
59     do
60     {
61         // call the module to get the size of the array data
62         nResult = (*pMAIDEntryPoint)( pObject, kNkMAIDCommand_CapGet, ulCapID,
63             kNkMAIDDataType_ArrayPtr, (NKPARAM)pArray, NULL, 0 );
64
65         if (nResult == kNkMAIDResult_NoError)
66         {
67             // allocate memory for the array
68             pArray->pData = malloc( pArray->ulElements * pArray->wPhysicalBytes );
69
70             if (pArray->pData != NULL)
71             {
72                 // call the module to get the array data
73                 nResult = (*pMAIDEntryPoint)( pObject, kNkMAIDCommand_CapGetArray, ulCapID,
74                     kNkMAIDDataType_ArrayPtr, (NKPARAM)pArray, NULL, 0 );
75
76                 if (nResult == kNkMAIDResult_BufferSize)
77                 {
78                     free( pArray->pData );
79                     pArray->pData = NULL;
80                 }
81             }
82         }
83     }
84     // repeat the process if the array data size changed between the two calls to the module
85     while (nResult == kNkMAIDResult_BufferSize);
86
87     // return TRUE if the array was successfully read
88     return (nResult == kNkMAIDResult_NoError);
89 }

```

3.5 範囲 capability の使用

3.6 capability グループ

3.7 ベンダー固有 capability の使用

3.8 ソースおよびアイテムのオープン

ソースもしくはアイテムオブジェクトを開く場合、まず最初にクライアントは親として使用するための、それぞれモジュールもしくはソースオブジェクトを必要とする。クライアントは親の *kNkMAIDCapability_Children* 機能を見つけ出す。(Line9) クライアントはこの配列機能を読み込み (Line14) 子オブジェクトの ID を選択する。クライアントは refClient メンバを設定することにより、子オブジェクトの構造体を初期化する。(Line47) そして子オブジェクトを開くために構造体へのポインタを渡してモジュールを呼び出す。(Line50) コマンドが正常に終了した場合には、オブジェクトは開かれている。(Line55)

例としてあげるために、子が開かれる度ごとに、機能が列挙され子 ID 配列が読み込まれている。実際のクライアントは、クライアントとモジュール間の対話を最小にし動作を高速化するために、これらの両方をキャッシュする。

```

1 // read the child IDs into an array structure
2 BOOL GetMAIDChildIDs(LPMAIDEntryPointProc pMAIDEntryPoint, LPNkMAIDObject pParentObject,
3 LPNkMAIDArray pChildIDArray )
4 {
5     LONG nResult;
6     NkMAIDCapInfo capInfo;
7
8     // this function is in the example for array capabilities
9     if (!FindMAIDCapability( pMAIDEntryPoint, pParentObject, kNkMAIDCapability_Children,
10 &capInfo ))
11         return FALSE;
12
13     // this function is in the example for array capabilities
14     if (ReadMAIDArrayCapability( pMAIDEntryPoint, pParentObject, kNkMAIDCapability_Children,
15 pChildIDArray ))
16     {
17         // the array must be 32 bit unsigned integers
18         if (pChildIDArray->ulType != kNkMAIDArrayType_Unsigned ||
19 pChildIDArray->wPhysicalSize != 4 || pChildIDArray->wLogicalBits != 32)
20             return FALSE;
21     }
22
23     return TRUE;
24 }
25
26 // open child object
27 BOOL OpenMAIDChild( LPMAIDEntryPointProc pMAIDEntryPoint, LPNkMAIDObject pParentObject,
28 ULONG ulChildIndex, NKREF refChild, LPNkMAIDObject pChildObject )
29 {
30     LONG nResult;
31     BOOL fRet = FALSE;
32     NkMAIDArray childIDArray;
33
34     // make sure we don't free some memory we didn't allocate
35     childIDArray.pData = NULL;
36
37     // get array of child IDs
38     if (GetMAIDChildIDs( pMAIDEntryPoint, pParentObject, &childIDArray ))
39     {
40         // ulChildIndex must be a valid index
41         if (childIDArray.ulElements > ulChildIndex && childIDArray.pData != NULL)
42         {
43             // get the ID of the child from the array
44             ULONG FAR *pulChildID = (ULONG FAR *)childIDArray.pData;
45
46             // set the reference
47             pChildObject->refClient = refChild;
48
49             // tell the module to open the child
50             nResult = (*pMAIDEntryPoint)( pParentObject, kNkMAIDCommand_Open,
51 pulChildID[ulChildIndex], kNkMAIDDataType_ObjectPtr,
52 (NKPARAM)pChildObject, NULL, 0 );
53
54             // return TRUE if the child was successfully opened

```

```

55         fRet = (nResult == kNkMAIDResult_NoError);
56     }
57 }
58
59 // make sure to free memory allocated by ReadMAIDArrayCapability()
60 if (childDArray.pData != NULL)
61     free( childDArray.pData );
62
63 return fRet;
64 }

```

3.9 データオブジェクトを開く

データオブジェクトを開くためには、クライアントは親として使用するアイテムオブジェクトを最初に必要とする。クライアントは利用できるデータ型をアイテムから得て(Line15)、必要とするデータ型が有効かどうかを調べる (Line15)。クライアントは refClient メンバを設定することによりデータオブジェクト構造体を初期化する。(Line18) そしてクライアントは、データオブジェクトを開くためにデータオブジェクト構造体へのポインタを引数にして、モジュールを呼び出す (Line21)。コマンドが正常に終了されれば、オブジェクトは開かれていることになる。(Line25)

```

1 // open data object
2 BOOL OpenMAIDDataObject( LPMAIDEntryPointProc pMAIDEntryPoint, LPNkMAIDObject pDataObject,
3     ULONG ulDataObjectType, NKREF refChild, LPNkMAIDObject pDataObject )
4 {
5     LONG nResult;
6     BOOL fRet = FALSE;
7     ULONG ulDataTypes;
8
9     // get the data types available for this item
10    nResult = (*pMAIDEntryPoint)( pDataObject, kNkMAIDCommand_CapGet,
11        kNkMAIDCapability_DataTypes, kNkMAIDDataType_UnsignedPtr,
12        (NKPARAM)( ULONG FAR *)&ulDataTypes, NULL, 0 );
13
14    // make sure we got an answer and that the data type requested is available
15    if (nResult == kNkMAIDResult_NoError && (ulDataTypes & ulDataObjectType) != 0)
16    {
17        // set the reference
18        pDataObject->refClient = refChild;
19
20        // tell the module to open the data object
21        nResult = (*pMAIDEntryPoint)( pDataObject, kNkMAIDCommand_Open,
22            ulDataObjectType, kNkMAIDDataType_ObjectPtr, (NKPARAM)pDataObject, NULL, 0 );
23
24        // return TRUE if the child was successfully opened
25        fRet = (nResult == kNkMAIDResult_NoError);
26    }
27
28    return fRet;
29 }

```

3.10 データ転送

どのようにして行うかを以下に示す。

```

1 // acquire a data object
2 BOOL AcquireMAIDDataObject( LPMAIDEntryPointProc pMAIDEntryPoint, LPNkMAIDObject pDataObject,
3     LPVOID FAR *ppData )
4 {
5     LONG nResult;
6     BOOL fRet = FALSE;
7     ULONG ulDataSize;
8     NKMAIDCallback cbDataProc;
9
10    // find out how large the data will be - this is different for images, sound and video
11    :
12    :
13
14    if (nResult == kNkMAIDResult_NoError)
15    {
16        // allocate the memory we need
17        *ppData = malloc( ulDataSize );
18
19        if (*ppData != NULL)
20        {
21            // make sure the data delivery callback function gets a pointer to the memory
22            cbDataProc.pProc = (LPNKFUNC)ReceiveMAIDData;
23            cbDataProc.ref = (NKREF)*ppData;
24
25            // set the data delivery callback function
26            nResult = (*pMAIDEntryPoint)( pDataObject, kNkMAIDCommand_CapSet,
27                kNkMAIDCapability_ProgressProc, kNkMAIDDataType_CallbackPtr,
28                (NKPARAM)(LPNkMAIDCallback)&cbDataProc, NULL, 0 );

```

```

29
30         if (nResult == kNkMAIDResult_NoError)
31         {
32             // start the acquire
33             nResult = (*pMAIDEntryPoint)( pDataObject, kNkMAIDCommand_CapStart,
34                 kNkMAIDCapability_Acquire, kNkMAIDDataType_Null, NULL, NULL, 0 );
35
36             // return TRUE if the acquire was successfully completed
37             fRet = (nResult == kNkMAIDResult_NoError);
38         }
39     }
40 }
41
42 return fRet;
43 }
44
45 // copy the delivered data
46 LONG ReceiveMAIDData( LPNkMAIDObject pObject, NKREF ref, LPVOID pDataInfo, LPVOID pData )
47 {
48     LPVOID pBuffer = (LPVOID)ref;    // reference value in callback structure
49
50     // interpret the structure pointed to by pDataInfo and copy the
51     // data in pData to a client allocated buffer
52     :
53     :
54
55     return kNkMAIDResult_NoError;
56 }

```

3.11 状態の保存と読み込み

それぞれのモジュール、ソース、アイテムおよびデータオブジェクトは、それぞれの固有の状態を持つことができる。

オブジェクトの現在の状態を得るには、配列性能の *kNkMAIDCapability_State* を読むだけで良い。この性能から取り出されるデータは、逐語的に保存されるのが望ましい。オブジェクトの状態を読み出すには、クライアントはオブジェクトから以前に読み込んだデータと共に、配列性能に *kNkMAIDCapability_State* をセットするだけで良い。

```

1 // get the object state
2 BOOL GetMAIDObjectState( LPMAIDEntryPointProc pMAIDEntryPoint, LPNkMAIDObject pObject,
3     LPNkMAIDArray pStateArray )
4 {
5     LONG nResult;
6     NKMAIDCapInfo capInfo;
7
8     // this function is in the example for array capabilities
9     if (!FindMAIDCapability( pMAIDEntryPoint, pParentObject, kNkMAIDCapability_State,
10         &capInfo ))
11         return FALSE;
12
13     // this function is in the example for array capabilities
14     if (ReadMAIDArrayCapability( pMAIDEntryPoint, pParentObject, kNkMAIDCapability_State,
15         pStateArray ))
16     {
17         // the array must be 32 bit unsigned integers
18         if (pStateArray->ulType != kNkMAIDArrayType_Unsigned ||
19             pStateArray->wPhysicalSize != 1 || pStateArray->wLogicalBits != 8)
20             return FALSE;
21     }
22
23     return TRUE;
24 }
25
26 // Set the object state
27 BOOL SetMAIDObjectState( LPMAIDEntryPointProc pMAIDEntryPoint, LPNkMAIDObject pObject,
28     LPNkMAIDArray pStateArray )
29 {
30     LONG nResult;
31     NKMAIDCapInfo capInfo;
32
33     // this function is in the example for array capabilities
34     if (!FindMAIDCapability( pMAIDEntryPoint, pParentObject, kNkMAIDCapability_State,
35         &capInfo ))
36         return FALSE;
37
38     // set the state
39     nResult = (*pMAIDEntryPoint)( pObject, kNkMAIDCommand_CapSet,
40         kNkMAIDCapability_State, kNkMAIDDataType_ArrayPtr,
41         (NKPARAM)pStateArray, NULL, 0 );
42
43     return (nResult == kNkMAIDResult_NoError);
44 }

```

3.12 イベントの通知

3.13 ユーザーインターフェースの要求

3.14 非同期によるモジュールの呼び出し

例として、オブジェクトを閉じるときを考える。クライアントは完了関数へのポインタを引数として、モジュールを呼び出す。(Line39) モジュールがコマンドを非同期で実行できる場合には、モジュールはすぐに *kNkMAIDResult_Pending* の戻り値を返す。この例では、クライアントはコマンドが正常に終了するのを待つため、*kNkMAIDCommand_Async* のコマンドでモジュールを繰り返し呼び出している。クライアントは、優先度を与えるために、このコマンドを直接オブジェクトに指示したり(Line20)、モジュールに対して何を行うべきか決定させることができる。(Line26) クライアントは完了コールバック関数によって設定される戻り値を確認する (Line10)。(Line50) モジュールがコマンドを同期として処理する場合、モジュールは完了関数を戻る前に呼び出し、終了待ちのループ(Line10-27)は実行されない。コマンドが正常に終了した場合には、オブジェクトは閉じられる。

```

1  BOOL CloseMAIDObject( LPMAIDEntryPointProc pMAIDEntryPoint, LPNkMAIDObject pObject )
2  {
3      LONG nResult;
4
5      // call the module asynchronously
6      if (CallMAIDAsync( pMAIDEntryPoint, pObject, kNkMAIDCommand_Close, 0, kNkMAIDDataType_NULL, 0,
7          &nResult ))
8      {
9          // loop while processing the command
10         while (nResult == kNkMAIDResult_Pending)
11         {
12             // respond to user interface items or perform other non-MAID operations
13             // ...
14
15             // give a single threaded module a chance to call the callback - the client can
16             // direct the async command at an object or let the module choose what object
17             // it should be directed to
18
19             // direct the module to process asynchronous commands only for this object
20             (*pMAIDEntryPoint)( pObject, kNkMAIDCommand_Async, 0, kNkMAIDDataType_Null, 0,
21                 NULL, 0 );
22
23             // .. OR ..
24
25             // let the module process asynchronous commands for any object
26             (*pMAIDEntryPoint)( NULL, kNkMAIDCommand_Async, 0, kNkMAIDDataType_Null, 0, NULL, 0 );
27         }
28     }
29
30     // return TRUE if the object was closed
31     return (nResult == kNkMAIDResult_NoError);
32 }
33
34 // call the module asynchronously
35 BOOL CallMAIDAsync( LPMAIDEntryPointProc pMAIDEntryPoint, LPNkMAIDObject pObject, ULONG ulCommand,
36     ULONG ulParam, ULONG ulDataType, NKPARAM data, LONG FAR *pnResult )
37 {
38     // call the module
39     *pnResult = (*pMAIDEntryPoint)( pObject, ulCommand, ulParam, ulDataType, data,
40         SetMAIDResult, (NKREF)pnResult );
41
42     // return TRUE if the command was started
43     return (nResult == kNkMAIDResult_NoError || nResult == kNkMAIDResult_Pending);
44 }
45
46 // save the result of the command in the reference
47 void SetMAIDResult( LPMAIDObject pObject, ULONG ulCommand, ULONG ulParam, ULONG ulDataType,
48     NKPARAM data, NKREF refComplete, LONG nResult )
49 {
50     *((LONG FAR *)refComplete) = nResult;
51 }

```

3.15 モジュールの終了

モジュールを終了させる場合には、クライアントはすべてのオブジェクトを閉じる。それぞれのオブジェクトを閉じる際に、モジュールは処理中のすべてのコマンドを中断し、コールバックが存在する場合には、

ulDone を ulTotal として処理コールバックを呼び出す。そして完了関数を呼び出し、 NkMAIDObject 構造体の refModule メンバを NULL に設定する。これによりオブジェクトはクライアントに間違いによって使用されることがなくなる。

3.16 モジュールの解放

モジュールの解放処理はシステムに依存する。

Windows では、クライアントはモジュール DLL を解放するのに FreeLibrary() を使用する。

PowerPC Macintosh では、クライアントは CloseConnection() を使用し、コードフラグメントとの接続を断つ。68k Macintosh では、クライアントは単にモジュールコードを保持しているメモリブロックのロックを解除し解放する。

4 一覧表

4.1 戻り値

```
enum eNkMAIDResult
{
    // these values are errors
    kNkMAIDResult_NotSupported = -127,
    kNkMAIDResult_UnexpectedDataType,
    kNkMAIDResult_ValueOutOfBounds,
    kNkMAIDResult_BufferSize,
    kNkMAIDResult_Aborted,
    kNkMAIDResult_NoMedia,
    kNkMAIDResult_NoEventProc,
    kNkMAIDResult_NoDataProc,
    kNkMAIDResult_ZombieObject,
    kNkMAIDResult_OutOfMemory,
    kNkMAIDResult_UnexpectedError,
    kNkMAIDResult_HardwareError,
    kNkMAIDResult_MissingComponent,

    kNkMAIDResult_NoError = 0,

    // these values are warnings
    kNkMAIDResult_Pending,
    kNkMAIDResult_OrphanedChildren,

    kNkMAIDResult_VendorBase = +127
};
```

モジュールは、クライアントの完了コールバック関数の nResult パラメータへこれらの値のうちの一つを渡す。もしくは MAID Entry Point の戻り値として同じ値を返す。エラーは負の値を持つ。

4.2 データオブジェクト型

```
enum eNkMAIDDataObjectType
{
    kNkMAIDDataObjectType_Image = 0x00000001,
    kNkMAIDDataObjectType_Sound = 0x00000002,
    kNkMAIDDataObjectType_Video = 0x00000004,
    kNkMAIDDataObjectType_Thumbnail = 0x00000008,
    kNkMAIDDataObjectType_File = 0x00000010
};
```

モジュールは、モジュールもしくはソースが作成できるデータ型を示すために、一つ以上の値を使用する。また特定のアイテムについてどのデータ型が使用できるかを示すためにも使用する。詳細については *kNkMAIDCapability_DataTypes* の記述を参照のこと

4.3 データ型

```
enum eNkMAIDDataType
{
    kNkMAIDDataType_Null = 0,
    kNkMAIDDataType_Boolean, // passed by value, set only
    kNkMAIDDataType_Integer, // signed 32 bit int, passed by value, set only
    kNkMAIDDataType_Unsigned, // unsigned 32 bit int, passed by value, set only
    kNkMAIDDataType_BooleanPtr, // pointer to single byte boolean value(s)
    kNkMAIDDataType_IntegerPtr, // pointer to signed 4 byte value(s)
    kNkMAIDDataType_UnsignedPtr, // pointer to unsigned 4 byte value(s)
    kNkMAIDDataType_FloatPtr, // pointer to DOUB_P value(s)
    kNkMAIDDataType_PointPtr, // pointer to NkMAIDPoint structure(s)
    kNkMAIDDataType_SizePtr, // pointer to NkMAIDSize structure(s)
    kNkMAIDDataType_RectPtr, // pointer to NkMAIDRect structure(s)
    kNkMAIDDataType_StringPtr, // pointer to NkMAIDString structure(s)
    kNkMAIDDataType_DateTimePtr, // pointer to NkMAIDDateTime structure(s)
    kNkMAIDDataType_CallbackPtr, // pointer to NkMAIDCallback structure(s)
    kNkMAIDDataType_RangePtr, // pointer to NkMAIDRange structure(s)
    kNkMAIDDataType_ArrayPtr, // pointer to NkMAIDArray structure(s)
    kNkMAIDDataType_EnumPtr, // pointer to NkMAIDEnum structure(s)
    kNkMAIDDataType_ObjectPtr, // pointer to NkMAIDObject structure(s)
    kNkMAIDDataType_CapInfoPtr, // pointer to NkMAIDCapInfo structure(s)
    kNkMAIDDataType_GenericPtr // pointer to some value
};
```

クライアントは MAID entry point の ulDataType パラメータにこれらの値のうちの一つを渡す。これはデータパラメータがモジュールによって、どのように解釈されるべきかを示す。

4.4 配列型

```
enum eNkMAIDArrayType
{
    kNkMAIDArrayType_Boolean = 0, // 1 byte per element
    kNkMAIDArrayType_Integer, // signed value that is 1, 2 or 4 bytes per element
    kNkMAIDArrayType_Unsigned, // unsigned value that is 1, 2 or 4 bytes per element
    kNkMAIDArrayType_Float, // DOUB_P elements
    kNkMAIDArrayType_Point, // NkMAIDPoint structures
    kNkMAIDArrayType_Size, // NkMAIDSize structures
    kNkMAIDArrayType_Rect, // NkMAIDRect structures
    kNkMAIDArrayType_PackedString, // packed array of strings
    kNkMAIDArrayType_String, // NkMAIDString structures
    kNkMAIDArrayType_DateTime // NkMAIDDateTime structures
};
```

モジュールは NkMAIDArray 構造体の ulType メンバにこれらの値のうちの一つを設定する。これは配列のデータがどのように解釈されるべきかを示す。詳細については NkMAIDArray 構造体を参照のこと。

4.5 機能型

```
enum eNkMAIDCapType
{
    kNkMAIDCapType_Process = 0, // a process that can be started
    kNkMAIDCapType_Boolean, // single byte boolean value
    kNkMAIDCapType_Integer, // signed 4 byte value
    kNkMAIDCapType_Unsigned, // unsigned 4 byte value
    kNkMAIDCapType_Float, // DOUB_P value
    kNkMAIDCapType_Point, // NkMAIDPoint structure
    kNkMAIDCapType_Size, // NkMAIDSize structure
    kNkMAIDCapType_Rect, // NkMAIDRect structure
    kNkMAIDCapType_String, // NkMAIDString structure
    kNkMAIDCapType_DateTime, // NkMAIDDateTime structure
    kNkMAIDCapType_Callback, // NkMAIDCallback structure
    kNkMAIDCapType_Array, // NkMAIDArray structure
    kNkMAIDCapType_Enum, // NkMAIDEnum structure
    kNkMAIDCapType_Range, // NkMAIDRange structure
    kNkMAIDCapType_Generic, // generic pointer
    kNkMAIDCapType_BooleanDefault, // NkMAIDBooleanDefault structure
};
```

モジュールは NkMAIDCapInfo 構造体の ulType メンバにこれらの値のうちの一つをセットする。これはどのような型の情報が提供されているかを示す。詳細については、機能の章もしくは NkMAIDCapInfo 構造体

の記述を参照すること。

4.6 機能操作

```
enum eNkMAIDCapOperations
{
    kNkMAIDCapOperation_Start      = 0x0001,
    kNkMAIDCapOperation_Get        = 0x0002,
    kNkMAIDCapOperation_Set        = 0x0004,
    kNkMAIDCapOperation_GetArray,   = 0x0008,
    kNkMAIDCapOperation_GetDefault = 0x0010
};
```

モジュールは NkMAIDCapInfo 構造体の ulOperations メンバにこれらの値の 1 つ以上をセットする。これは特定の機能において、どの動作が許可されているのかを示す。詳細については機能の章と NkMAIDCapInfo 構造体の記述を参照のこと。

4.7 機能の可視性

```
enum eNkMAIDCapVisibility
{
    kNkMAIDCapVisibility_Hidden      = 0x0001,
    kNkMAIDCapVisibility_Advanced    = 0x0002,
    kNkMAIDCapVisibility_Vendor      = 0x0004,
    kNkMAIDCapVisibility_Group        = 0x0008,
    kNkMAIDCapVisibility_GroupMember = 0x0010,
    kNkMAIDCapVisibility_Invalid     = 0x0020
};
```

モジュールは NkMAIDCapInfo 構造体の ulVisibility メンバにこれらの値の 1 つ以上をセットする。これは特定の機能がどのレベルの可視性を持つかを示す。詳細については、機能の章と NkMAIDCapInfo 構造体の記述を参照のこと。

4.8 オブジェクト型

```
enum eNkMAIDObjectType
{
    kNkMAIDObjectType_Module = 1,
    kNkMAIDObjectType_Source,
    kNkMAIDObjectType_Item,
    kNkMAIDObjectType_DataObj
};
```

モジュールは NkMAIDObject 構造体の ulType メンバにこれらの値のうちの一つをセットする。これはどの型のオブジェクトが提供されているかを示す。

4.9 イベント

```
enum eNkMAIDEvent
{
    kNkMAIDEvent_AddChild,
    kNkMAIDEvent_RemoveChild,
    kNkMAIDEvent_WarmingUp,
    kNkMAIDEvent_WarmedUp,
    kNkMAIDEvent_CapChange,
    kNkMAIDEvent_OrphanedChildren,
    kNkMAIDEvent_CapChangeValueOnly
};
```

モジュールは、どのイベントが発生したのかを示すために、クライアントのイベントコールバック関数の ulEvent パラメータに、これらの値のうちの一つを渡す。

4.10 ユーザインターフェイス要求型

```
enum eNkMAIDUI RequestType
{
    kNkMAIDUI RequestType_Ok,
    kNkMAIDUI RequestType_OkCancel ,
    kNkMAIDUI RequestType_YesNo,
    kNkMAIDUI RequestType_YesNoCancel ,
};
```

モジュールがクライアントのユーザインターフェイスコールバック関数を呼び出すときには、NkMAIDUIRequetInfo 構造体の ulType メンバに、これらの値のうちの 하나가セットされる。ユーザには値によって特定された選択が提示される。

4.11 ユーザインターフェイス値

```
enum eNkMAIDUI RequestResul t
{
    kNkMAIDUI RequestResul t_None,
    kNkMAIDUI RequestResul t_Ok,
    kNkMAIDUI RequestResul t_Cancel ,
    kNkMAIDUI RequestResul t_Yes,
    kNkMAIDUI RequestResul t_No
};
```

モジュールがクライアントのユーザインターフェイスコールバック関数を呼ぶときに、NkMAIDUIRequestInfo 構造体の ulDefault メンバはこれらの値のうちの 하나にセットされる。この値は、デフォルトでどのボタンがハイライト表示されるべきかを示す。クライアントのユーザインターフェイスコールバック関数は、ユーザの押したボタンに関連した、これらの値のうちの 하나を返す。*kNkMAIDEventResult_None* 値は、NkMAIDUIRequestInfo 構造体の fSync メンバが FALSE の場合のみとりうる値である。

4.12 フィルタ

```
enum eNkMAIDFi lter
{
    kNkMAIDFi lter_Whi te,
    kNkMAIDFi lter_I nfrared,
    kNkMAIDFi lter_Red,
    kNkMAIDFi lter_Green,
    kNkMAIDFi lter_Bl ue,
    kNkMAIDFi lter_UI travi ol et
};
```

モジュールは kNkMAIDCapability_Filter 機能において、これらの値の 하나以上を使用する。詳細については、機能の章を参照のこと。

4.13 コマンド

```
enum eNkMAIDCommand
{
    kNkMAIDCommand_Async,           // process asynchronous operations
    kNkMAIDCommand_Open,            // opens a child object
    kNkMAIDCommand_Close,           // closes a previously opened object
    kNkMAIDCommand_GetCapCount,     // get number of capabilities of an object
    kNkMAIDCommand_GetCapInfo,      // get capabilities of an object
    kNkMAIDCommand_CapStart,        // starts capability
    kNkMAIDCommand_CapSet,          // set value of capability
    kNkMAIDCommand_CapGet,          // get value of capability
    kNkMAIDCommand_CapGetDefault,    // get default value of capability
    kNkMAIDCommand_CapGetArray,     // get data for array capability
    kNkMAIDCommand_Mark,            // insert mark in the command queue
    kNkMAIDCommand_AbortToMark,     // abort asynchronous commands to mark
    kNkMAIDCommand_Abort,           // abort current asynchronous command
    kNkMAIDCommand_EnumChildren,    // requests 'add' events for all child IDs
    kNkMAIDCommand_GetParent        // gets previously opened parent for object
};
```

クライアントは、MAID entry point に対する ulCommand パラメータにこれらの値のうちの一つを渡す。これはモジュールがどのような操作を行うべきかを示す。これらのコマンドについては、コマンドの章で詳細に述べる。

4.14 Capabilities

```
enum eNkMAIDCapability
{
    kNkMAIDCapability_AsyncRate = 1,           // milliseconds between idle async calls
    kNkMAIDCapability_ProgressProc,           // callback during lengthy operation
    kNkMAIDCapability_EventProc,             // callback when event occurs
    kNkMAIDCapability_DataProc,              // callback to deliver data
    kNkMAIDCapability_UIRequestProc,          // callback to show user interface

    kNkMAIDCapability_IsAlive,               // FALSE if object is removed or parent closed
    kNkMAIDCapability_Children,             // IDs of children objects
    kNkMAIDCapability_State,                 // current state of the object
    kNkMAIDCapability_Name,                  // a string representing the name of the object
    kNkMAIDCapability_Description,           // a string describing the object
    kNkMAIDCapability_Interface,             // a string describing the interface to a device
    kNkMAIDCapability_DataTypes,             // what data types are supported or available
    kNkMAIDCapability_DateTime,             // date associated with an object
    kNkMAIDCapability_StoredBytes,           // read only size of object as stored on device

    kNkMAIDCapability_Eject,                 // ejects media from a device
    kNkMAIDCapability_Feed,                  // feeds media into a device
    kNkMAIDCapability_Capture,               // captures new item from the source
    kNkMAIDCapability_MediaPresent,          // TRUE if item has physical media to acquire

    kNkMAIDCapability_Mode,                  // mode of this object
    kNkMAIDCapability_Acquire,               // begins the acquisition of the object
    kNkMAIDCapability_ForceScan,             // If FALSE, unnecessary scans can be eliminated

    kNkMAIDCapability_Start,                 // start offset (in seconds) of the object
    kNkMAIDCapability_Length,                // length (in seconds) of the object
    kNkMAIDCapability_SampleRate,            // sampling rate (in samples/sec) of the object
    kNkMAIDCapability_Stereo,                // mono or stereo
    kNkMAIDCapability_Samples,               // given current state, read only number of samples

    kNkMAIDCapability_Filter,                // selects the filter for the light source
    kNkMAIDCapability_Prescan,               // performs a prescan
    kNkMAIDCapability_ForcePrescan,          // If FALSE, unnecessary prescans can be eliminated
    kNkMAIDCapability_AutoFocus,             // sets the focus automatically
    kNkMAIDCapability_ForceAutoFocus,        // If FALSE, unnecessary autofocus can be eliminated
    kNkMAIDCapability_AutoFocusPt,           // sets the position to focus upon
    kNkMAIDCapability_Focus,                 // sets the focus
    kNkMAIDCapability_Coords,                // rectangle of object in device units
    kNkMAIDCapability_Resolution,            // resolution of object (in pixels/inch)
    kNkMAIDCapability_Preview,               // preview or final acquire
    kNkMAIDCapability_Negative,              // negative or positive original
    kNkMAIDCapability_ColorSpace,            // colorspace for image delivery
    kNkMAIDCapability_Bits,                  // bits per color
    kNkMAIDCapability_Planar,                // interleaved or planar data transfer
    kNkMAIDCapability_Lut,                   // LUT(s) for object
    kNkMAIDCapability_Transparency,           // light path of the original
    kNkMAIDCapability_Threshold,             // threshold level for linear images
    kNkMAIDCapability_Pixels,                // given current state, read only size of image

    kNkMAIDCapability_NegativeDefault,       // Default value for Negative capability
    kNkMAIDCapability_Firmware,              // Firmware version number

    kNkMAIDCapability_CommunicationLevel1,   // Communication method
    kNkMAIDCapability_CommunicationLevel2,   // Communication method
    kNkMAIDCapability_BatteryLevel,          // Battery Level in device
    kNkMAIDCapability_FreeBytes,              // Free bytes in device
    kNkMAIDCapability_FreeItems,             // Free items in device
    kNkMAIDCapability_Remove,                // Delete an object
    kNkMAIDCapability_FlashMode,             // Flash mode
    kNkMAIDCapability_ModuleType,            // Module type
    kNkMAIDCapability_AcquireStreamStart,     // Start a stream acquisition
    kNkMAIDCapability_AcquireStreamStop,     // Stop a stream acquisition
    kNkMAIDCapability_AcceptDiskAcquisition, // Allow acquisitions to use disk transfer
    kNkMAIDCapability_Version,               // MAID version

    kNkMAIDCapability_FilmFormat,            // Film format (35mm, 6x6 etc)
    kNkMAIDCapability_TotalBytes,            // Total bytes in device storage

    kNkMAIDCapability_VendorBase = 0x8000    // vendor supplied capabilities start here
};
```

モジュールは NkMAIDCapInfo 構造体の ulID メンバにこれらの値を使用する。これはどのような機能がクライアントに対して提供されるかを示す。クライアントは MAID entry point の ulParam パラメータへこれらの値のうちの一つを渡す。この値はコマンド上でどのような機能が処理されるかを示す。

モジュール製作者は、デバイス固有の機能を定義することができる。クライアントはユーザに対して一般的な規則のもとで、これらの機能と相互作用することを可能にする。

これらの機能については、機能の章で詳細に述べる。

4.15 色空間

```
enum eNkMAIDColorSpace
{
    kNkMAIDColorSpace_LineArt,
    kNkMAIDColorSpace_Grey,
    kNkMAIDColorSpace_RGB,
    kNkMAIDColorSpace_sRGB,
    kNkMAIDColorSpace_CMYK,
    kNkMAIDColorSpace_Lab,
    kNkMAIDColorSpace_LCH,
    kNkMAIDColorSpace_AppleRGB,
    kNkMAIDColorSpace_ColorMatchRGB,
    kNkMAIDColorSpace_NTSCRGB,
    kNkMAIDColorSpace_BruceRGB,
    kNkMAIDColorSpace_AdobeRGB,
    kNkMAIDColorSpace_CIE_RGB,
    kNkMAIDColorSpace_AdobeWideRGB,
    kNkMAIDColorSpace_AppleRGB_Compensated,

    kNkMAIDColorSpace_VendorBase = 0x8000 // vendor supplied color spaces start here
};
```

クライアントは、これらの値を `kNkMAIDCapability_ColorSpace_Capability` で使用する。詳細については、`Capability` の章を参照のこと。

4.16 Boolean Default

```
enum eNkMAIDBooleanDefault
{
    kNkMAIDBooleanDefault_True,
    kNkMAIDBooleanDefault_False
};
```

クライアントは、これらの値を `kNkMAIDCapability_NegativeDefault_Capability` で使用する。詳細については、`Capability` の章を参照のこと。

4.17 Module Types

```
enum eNkMAIDModuleTypes
{
    kNkMAIDModuleType_Scanner = 0x0001,
    kNkMAIDModuleType_Camera = 0x0002
};
```

モジュールは、`kNkCapability_ModuleType` の `Get` コマンドに対し `data` にこれらのうち一つ以上をセットして返す。これはクライアントが、このモジュールを使用するか、あるいはユーザーインターフェースを表示するかを判断する助けになる。

4.18 File Data Types

```
enum eNkMAIDFileDataTypes
{
    kNkMAIDFileDataType_NotSpecified,
    kNkMAIDFileDataType_JPEG,
    kNkMAIDFileDataType_TIFF,
    kNkMAIDFileDataType_FlashPix,
    kNkMAIDFileDataType_NIF,
    kNkMAIDFileDataType_QuickTime,
    kNkMAIDFileDataType_UserType = 0x100
};
```

モジュールは、`kNkMAIDCapability_Acquire` を受けファイルデータをクライアントに転送する時、何れかの値を `NkMAIDFileInfo` 構造体の `ulFileDataType` メンバにセットする。

4.19 Flash Modes

```
enum eNkMAIDFlashMode
{
    kNkMAIDFlashMode_FrontCurtain,
    kNkMAIDFlashMode_RearCurtain,
    kNkMAIDFlashMode_SlowSync,
    kNkMAIDFlashMode_RedEyeReduction,
    kNkMAIDFlashMode_SlowSyncRedEyeReduction,
    kNkMAIDFlashMode_SlowSyncRearCurtain
};
```

クライアントは、これらの値を `kNkMAIDCapability_FlashModeCapability` で使用する。詳細については、`Capability` の章を参照のこと。

5 構造体と型

5.1 Word 値

```
typedef unsigned short WORD;
```

この定義は、16bit の符号無し整数に相当する値の必要性によって記述される。

5.2 Unsigned Long 値

```
typedef unsigned long ULONG;
```

この定義は、32bit の符号無し整数に相当する値の必要性によって記述される。

5.3 パラメータ値

```
typedef ULONG NKPARAM;
```

この定義は、オブジェクトへのポインタもしくは 32bit 整数に相当する値の必要性によって記述される。

5.4 ポインタ値

```
typedef void FAR *LPVOID;
```

この定義は、オブジェクトへのポインタに相当する値の必要性によって記述される。

5.5 参照値

```
typedef LPVOID NKREF;
```

この定義は、オブジェクトへのポインタに相当する値の必要性によって記述される。この型は、MAID クライアントが構造体と、他の構造体もしくはオブジェクトとを関連付ける必要があるときに使用される。またコールバック関数においても使用される。

5.6 MAID Entry Point 関数ポインタ

```
typedef LONG (FAR *LPMAIDEntryPointProc)( LPNkMAIDObject, ULONG, ULONG, ULONG, NKPARAM,
LPMAIDCompletionProc, NKREF );
```

5.7 MAID 完了関数ポインタ

```
typedef void (FAR *LPMAIDCompletionProc)( LPNkMAIDObject, ULONG, ULONG, ULONG, NKPARAM, NKREF, LONG );
```

5.8 MAID データ引き渡し関数ポインタ値

```
typedef LONG (FAR *LPMAIDDataProc)( NKREF, LPVOID, LPVOID );
```

5.9 MAID イベント通知関数ポインタ

```
typedef void (FAR *LPMAIDEventProc)( NKREF, ULONG, NKPARAM );
```

5.10 MAID 処理通知関数ポインタ

```
typedef void (FAR *LPMAIDProgressProc)( ULONG, ULONG, NKREF, ULONG, ULONG );
```

5.11 MAID ユーザインターフェイス要求関数ポインタ

```
typedef ULONG (FAR *LPMAIDUIRequestProc)( NKREF, LPNkMAIDUIRequestInfo );
```

5.12 コールバック定義構造体

```
typedef struct tagNkMAIDCallback
{
    LPNKFUNC      pProc;
    NKREF         refProc;
} NkMAIDCallback, FAR* LPNkMAIDCallback;
```

この構造体は、コールバック関数を記述するのに使用される。pProc メンバは関数へのポインタで、refProc メンバはそれぞれの目的のために、コールバックで使用される。通常、refProc メンバはオブジェクトもしくは構造体へのポインタとして MAID クライアントによって使用される。

5.13 Date/Time 構造体

```
typedef struct tagNkMAIDDateTime
{
    WORD nYear;           // ie 1997, 1998
    WORD nMonth;          // 0-11 = Jan-Dec
    WORD nDay;            // 1-31
    WORD nHour;           // 0-23
    WORD nMinute;         // 0-59
    WORD nSecond;         // 0-59
    ULONG nSubsecond;     // Module dependent
} NkMAIDDateTime, FAR* LPNkMAIDDateTime;
```

個々のモジュールは、nSubsecond メンバをどのように解釈するかを決定することができる。例えば、同一秒内にいくつかの写真が撮影された場合、0 から始まる連続値を割り当てることができる。あるいはミリ秒として使用することもできる。

5.14 ポイント構造体

```
typedef struct tagNkMAIDPoint
{
    LONG x;
    LONG y;
} NkMAIDPoint, FAR* LPNkMAIDPoint;
```

5.15 Size 構造体

```
typedef struct tagNkMAIDSize
{
    LONG    w;
    LONG    h;
} NkMAIDSize, FAR* LPNkMAIDSize;
```

5.16 矩形構造体

```
typedef struct tagNkMAIDRect
{
    LONG    x;        // left coordinate
    LONG    y;        // top coordinate
    ULONG   w;        // width
    ULONG   h;        // height
} NkMAIDRect, FAR* LPNkMAIDRect;
```

5.17 文字列構造体

```
typedef struct tagNkMAIDString
{
    SCHAR    str[256];    // allows a 255 character null terminated string
} NkMAIDString, FAR* LPNkMAIDString;
```

文字列は NULL で終端されていなければならない。この構造体を使用するにあたって、転送可能な文字列の最大の長さは 255 文字である。長い文字列については、ulType メンバに *kNkMAIDArrayType_PackedString* をセットしての NkMAIDArray 構造体の使用を検討すること。

5.18 配列構造体

```
typedef struct tagNkMAIDArray
{
    ULONG    ulType;        // one of eNkMAIDArrayType
    ULONG    ulElements;    // total number of elements
    ULONG    ulDimSize1;    // size of first dimension
    ULONG    ulDimSize2;    // size of second dimension, zero for 1 dim
    ULONG    ulDimSize3;    // size of third dimension, zero for 1 or 2 dim
    WORD     wPhysicalBytes; // bytes per element
    WORD     wLogicalBits;   // must be <= wPhysicalBytes * 8
    LPVOID    pData;        // allocated by the client
} NkMAIDArray, FAR* LPNkMAIDArray;
```

NkMAIDArray 構造体は MAID インターフェイスを通じて配列を転送できるようにする。クライアントは常にメモリを確保する。データを適切に取り扱うのは、データの受け手側の責任となる。バイト指定による pData のサイズは、常に ulElements の wPhysicalBytes 倍と等しくなければならない。

2 もしくは 3 次元の配列が、ulDimSize1、ulDimSize2 および ulDimSize3 をセットすることにより転送されうる。各要素が 10 個の 20 列の 2 次元配列については、ulDimSize1 が 10、ulDimSize2 が 20、ulElements が 200 となる。もしこのような配列が 5 つあるとしたら、ulDimSize1 が 10、ulDimSize2 が 20、ulDimSize3 が 5 で、ulElements が 1000 となる。

ulType が *kNkMAIDArrayType_Integer* もしくは *kNkMAIDArrayType_Unsigned* のときは、送り側は次のことを指定している。pData における整数値は、それぞれ 2 バイト(wPhysicalBytes)であるが、受け側は、これらを 10bit 値(wLogicalBits)として解釈しなければならない。

ulType が *kNkMAIDArrayType_PackedString* の場合には、pData は NULL で終端された文字列の一塊のリストへのポインタである。ulElements は終端文字 NULL のバイトを含んだデータの合計バイト長である。ulDimSize1 は文字列の数を示す。ulDimSize2 は 0 で、wPhysicalBytes は 1 となる。

5.19 範囲構造体

```
typedef struct tagNkMAIDRange
{
    DOUB_P    lfValue;
    DOUB_P    lfDefault;
    ULONG     ulValueIndex;           // zero-based index
    ULONG     ulDefaultIndex;        // zero-based index
    DOUB_P    lfLower;
    DOUB_P    lfUpper;
    ULONG     ulSteps;                // zero for infinite range, otherwise must be >= 2
} NkMAIDRange, FAR* LPNkMAIDRange;
```

この構造体は、値の数的な範囲(0-100,-5.0 ~ +5.0 など)により機能を実現するためのものである。最小値と最大値は、それぞれ lfLower および lfUpper に入れられる。

lfLower から lfUpper までのすべての値が使用できる場合、lfValue は現在の値、lfDefault はデフォルト値、ulSteps は 0 となる。この場合、ulValueIndex および ulDefaultIndex は使用されない。

不連続な段階的な値しか使用できない場合、ulValueIndex は現在の段階のインデックス、ulDefaultIndex はデフォルトの段階のインデックス、ulSteps は最小値と最大値を含む等間隔に分割されたステップの数を示す。すくなくとも 2 つの異なった段階が存在しなければならない。この場合、lfValue と lfDefault は使用されない。

範囲の値の設定には 2 つの方法がある。クライアントは lfValue もしくは ulValueIndex に新しい値をいれた NkMAIDRange 構造体へのポインタを送ることができる。不連続な段階的な値の範囲については、クライアントは、モジュールが値のインデックスとして受け取ることになっている符号無し整数を送ることができる。この場合、0 が最小値となり、ulSteps-1 が最大値となる。

5.20 機能情報構造体

```
typedef struct tagNkMAIDCapInfo
{
    ULONG     ulID;                  // one of eNkMAIDCapability or vendor specified
    ULONG     ulType;               // one of eNkMAIDCapabilityType
    ULONG     ulVi si bi l i t y;    // eNkMAIDCapVi si bl i t y bi ts
    ULONG     ulOperations;         // eNkMAIDCapOperations bi ts
    SCHAR     szDescription[256];   // text describing the capability
} NkMAIDCapInfo, FAR* LPNkMAIDCapInfo;
```

モジュール、ソース、アイテム、イメージおよび音声のオブジェクトは、それぞれの固有の機能を持つ。これらの構造体の配列をとりだすには、*kNkMAIDCommand_GetCapInfo* を使用する。それぞれは、ulID にユニークな識別値を持っている。ID は、eNkMAIDCapability のうちの一つもしくはベンダー特有の値をとることができる。

5.21 オブジェクト構造体

```
typedef struct tagNkMAIDObject
{
    ULONG     ulType;               // one of eNkMAIDObjectType
    ULONG     ulID;
    NKREF     refClient;
    NKREF     refModule;
} NkMAIDObject, FAR* LPNkMAIDObject;
```

この構造体は、MAID クライアントとモジュール間で渡される、モジュール、ソース、アイテム、イメージおよび音声のオブジェクトを表すことができる。

オブジェクトを開くには、クライアントは必要とされるメモリを確保し、それが必要としているすべてを refClient に設定する。次にクライアントは、オブジェクトを開くようにモジュールを呼び出す。呼び出しにおいて、モジュールは ulType を適当な値に、ulID をオブジェクトの ID に、refModule に必要とされるすべ

ての値を設定する。オブジェクトが開かれている間は、refClient および refModule の値は変化しない。

モジュール、ソース、アイテム、イメージもしくは音声は2回目に開かれたときには、モジュールは新しいオブジェクトに対する2つ目の機能値を保持する。モジュールは、refModule メンバの値により、最初のインスタンスと2度目のインスタンスを区別することができる。クライアントはrefClient メンバによって2つを区別することができる。

5.22 ユーザーインターフェイス要求構造体

```
typedef struct tagNkMAI DUI RequestInfo
{
    ULONG                ulType; // one of eNkMAI DUI RequestType
    ULONG                ulDefault; // default value - one of eNkMAI DUI RequestResult
    BOOL                fSync; // TRUE if user must respond before returning
    char FAR *           lpPrompt; // NULL terminated text to show to user
    char FAR *           lpDetail; // NULL terminated text indicating more detail
    LPNkMAI DObject      pObject; // Target Object for data element
    NKPARAM              data; // Pointer to an NkMAI DArray structure
} NkMAI DUI RequestInfo, FAR* LPNkMAI DUI RequestInfo;
```

モジュールが、ユーザに対してなんらかの通知を行う場合、もしくはユーザからの入力を要求する場合、pUIRequest をこの構造体へのポインタに設定して、クライアントのユーザーインターフェイス関数を呼び出す。ulType メンバは、ユーザに対してどのボタンを有効にするかを示す。ulDefault メンバは、どのボタンが初期状態でハイライト表示になるかを示す。もし fSync メンバが TRUE ならば、クライアントはすぐにダイアログを表示して、ユーザの入力を待たなければならない。もし FALSE ならばクライアントは *kNkMAIDEventResult_None* 値を返した後に少し遅れてダイアログを表示するか、ユーザの反応を待つかのどちらかを行うことができる。

より詳細な情報が存在する場合、lpDetail メンバは、モジュールによって提供される NULL 終端の文字列へのポインタとなる。ポインタはユーザーインターフェイスコールバックの期間だけ有効である。クライアントは非同期でユーザーインターフェイスを表示するときには、文字列のコピーを作成しなければならない。詳細な情報が無い場合には、lpDetail メンバは NULL にセットされなければならない。

ユーザーインターフェイス要求が、Capability を持たない単なるメッセージの場合には、pObject および data メンバは NULL に設定されなければならない。Capability がユーザに表示される場合、data メンバはモジュールによって確保される NkMAIDArray 構造体を指す。配列構造体は、一つ以上の MAID Capability の ID (enkMAIDCapability で定められている) を含む。クライアントは、これらの Capability をユーザに表示しようと試みる。配列構造体は、以下に示すように設定されなければならない：

```
1  ulType = kNkMAI DArrayType_Unsigned
2  ulElements = <number of capabilities to be displayed>
3  ulDimSize1 = <same as ulElements>
4  ulDimSize2 = 0
5  ulDimSize3 = 0
6  wPhysicalBytes = 4
7  wLogicalBits = 32
8  pData = <array of capability IDs allocated by the module>
```

次の表は、ユーザーインターフェイス要求のそれぞれの型に対する、クライアント側の記述の目安である。

Capability Type	Likely UI
Process	Button
Boolean	Check Box
Integer, Unsigned, Float, String	Edit Control
Point, Size, Rect	Custom UI
DateTime	Edit Controls or Custom UI

Callback	Undefined
Array	Radio Button Group
Range	Slider or Spin Control

5.23 標準データ配送構造体

```
typedef struct tagNkMAIDDataInfo
{
    ULONG          ulType;          // one of eNkMAIDDataObjType
} NkMAIDDataInfo, FAR* LPNkMAIDDataInfo;
```

この構造体は、NkMAIDImageInfo、NkMAIDSoundInfo と NkMAIDFileInfo 構造体において使用される。これは、クライアントのデータ配送コールバック関数に対してどのようなデータ型が配送されようとしているのかを示す。kNkMAIDDataObjType_File が他の値と共に **ulType** に含まれる場合、一緒に転送されるデータはファイルとして構成されており、NkMAIDFileInfo を使用しなければならない。

5.24 イメージデータ配送構造体

```
typedef struct tagNkMAIDImageInfo
{
    NkMAIDDataInfo base;
    NkMAIDSize      szTotalPixels; // total size of image to be transferred
    ULONG          ulColorSpace;    // One of eNkMAIDColorSpace
    NkMAIDRect      rData;          // coords of data, (0,0) = top left
    ULONG          ulRowBytes;      // number of bytes per row of pixels
    WORD           wBits[4];        // number of bits per plane per pixel
    WORD           wPlane;          // see below for description
    BOOL           fRemoveObject;   // TRUE if the object should be removed
} NkMAIDImageInfo, FAR* LPNkMAIDImageInfo;
```

モジュールはイメージデータが配送されようとしていることを示すため、この構造体へのポインタをクライアントのデータ配送コールバック関数への pData パラメータへセットする。fColorSpace および wBits メンバは全体としてのイメージに適用される。カラー画像のうちの一つのプレーンのみが転送されようとしているときには、ulColorSpace は全イメージの色空間を示し、wBits の要素のすべてがセットされる。

データが、kNkMAIDColorSpace_LineArt もしくは kNkMAIDColorSpace_Gray で送られる場合、wPlane パラメータは無視される。データがカラーフォーマットのうちの一つで送られ、一度に一つのプレーンで送られる場合、wPlane パラメータは転送されるプレーンを示す。RGB および sRGB では R=1, G=2, B=3。CMYK では C=1, M=2, Y=3, K=4。Lab では L=1, A=2, B=3。LCH では L=1, C=2, H=3 である。データがカラーフォーマットのうちの一つで送られ、ひとかたまりのフォーマットで転送される場合には、wPlane パラメータは 0 となる。

ひとかたまりのカラーデータは常にインターリーブ、ulColorSpace(RGB, CMYK, LAB もしくは LCH 順)で指定された順序、LSB 整列およびバイト整列として配送される。これは、カラーあたり 10bit のデータは、1 ピクセルおよび 1 カラーあたり 2 バイトを占有し、有効なビットはそれぞれの 2 バイトのペアの中の下位 10bit になる。バイトオーダはシステムに固有である。Windows 環境では、下位バイトが最初となる。Macintosh 環境では上位バイトが最初になる。

モジュールは、fRemoveObject を TRUE にセットすることで、クライアントがデータの受信を完了後そのデータオブジェクトを削除するよう要求できる。クライアントは、この要求通りに動作する義務はない。分割してデータが転送される場合、最後の転送時のみこのフラグを TRUE にセットしなければならない。さもなければ、データが失われることになるかもしれない。クライアントは kNkMAIDCapability_Remove を使ってデータオブジェクトを削除してもよい。アイテムオブジェクトに含まれるデータオブジェクトがそれ一つだけの場合、モジュールはアイテムオブジェクトを削除しても良い。この場合、モジュールはソースオブジ

エクトに対し `kNkMAIDEvent_RemoveChild` イベントを送らなければならない。

5.25 音声データ配送構造体

```
typedef struct tagNkMAIDSoundInfo
{
    NkMAIDDataInfo base;
    ULONG           ulTotalSamples; // number of full samples to be transferred
    BOOL            fStereo;        // TRUE if stereo, FALSE if mono
    ULONG           ulStart;        // index of starting sample of data
    ULONG           ulLength;       // number of samples of data
    WORD            wBits;          // number of bits per channel
    WORD            wChannel;       // 0 = mono or L+R; 1, 2 = left, right
    BOOL            fRemoveObject;  // TRUE if the object should be removed
} NkMAIDSoundInfo, FAR* LPNkMAIDSoundInfo;
```

モジュールは音声データが配送されようとしていることを示すため、この構造体へのポインタをクライアントのデータ配送コールバック関数への `pData` パラメータへセットする。`fStereo` メンバは全体としてのサウンドに適用される。ステレオ音声のうちの 1 チャンネルのみが転送されようとしているときは、`TRUE` となる。

ステレオデータは、常にインターリーブ、LR 順、LSB 整列、バイト整列で配送される。これは、チャンネルあたり 10bit のデータは、1 サンプルおよび 1 チャンネルあたり 2 バイトを占有し、有効なビットはそれぞれの 2 バイトのペアの中の下位 10bit になる。バイトオーダはシステムに固有である。Windows 環境では、下位バイトが最初となる。Macintosh 環境では上位バイトが最初になる。

モジュールは、`fRemoveObject` を `true` にセットすることで、クライアントがデータの受信を完了後そのデータオブジェクトを削除するよう要求できる。クライアントは、この要求通りに動作する義務はない。分割してデータが転送される場合、最後の転送時のみこのフラグを `TRUE` にセットしなければならない。さもなければ、データが失われることになるかもしれない。クライアントは `kNkMAIDCapability_Remove` を使ってデータオブジェクトを削除してもよい。アイテムオブジェクトに含まれるデータオブジェクトがそれ一つだけの場合、モジュールはアイテムオブジェクトを削除しても良い。この場合、モジュールはソースオブジェクトに対し `kNkMAIDEvent_RemoveChild` イベントを送らなければならない。

5.26 列挙構造体

```
typedef struct tagNkMAIDEnum
{
    ULONG           ulType;          // one of eNkMAIDArrayType
    ULONG           ulElements;      // total number of elements
    ULONG           ulValue;         // current index (zero-based)
    ULONG           ulDefault;       // default index (zero-based)
    WORD            wPhysicalBytes;  // bytes per element
    LPVOID          pData;          // allocated by the client
} NkMAIDEnum, FAR* LPNkMAIDEnum;
```

`NkMAIDEnum` 構造体は、MAID インターフェイスを通じて列挙型が転送されるようにする。クライアントが常にメモリを確保する。データを適切に解釈するのは、データの受け側の責任である。`pData` のバイトサイズは、常に `ulElements` の `wPhysicalBytes` 倍でなければならない。

unsigned integer which will be interpreted as the index.

この構造体は、オプションを選択する機能を実装することができる。現在のインデックスは `ulValue` に、デフォルトのインデックスは `ulDefault` に設定される。もし `ulType` が `kNkMAIDArrayType_String` もしくは `kNkMAIDArrayType_PackedString` の場合は、文字列はユーザに提供されるテキスト表示である。

選択機能の値は、2 つの方法でセットすることができる。クライアントは新しいインデックスを `ulValue` に

設定して NkMAIDEnum 構造体へのポインタを送ることができる。もしくはインデックスとして解釈される符号無し整数によって送ることもできる。

ulType が kNkMAIDArrayType_PackedString の場合、pData は NULL で終端された文字列のパックされたリストへのポインタである。ulElements は終端の NULL バイトを含んだデータの全体バイト長である。wPhysicalBytes は 1 となる。

5.27 ファイルデータ配送構造体

```
typedef struct tagNkMAIDFileInfo
{
    NkMAIDDataInfo base;
    ULONG           ulFileType; // One of eNkMAIDFileDataTypes
    ULONG           ulTotalLength; // total number of bytes to be transferred
    ULONG           ulStart; // index of starting byte (0-based)
    ULONG           ulLength; // number of bytes in this delivery
    BOOL            fDiskFile; // TRUE if the file is delivered on disk
    BOOL            fRemoveObject; // TRUE if the object should be removed
} NkMAIDFileInfo, FAR* LPNkMAIDFileInfo;
```

モジュールは、クライアントのデータ転送コールバック関数の pDataInfo に、この構造体へのポインタをセットする。転送されるデータの情報はこの構造体によりクライアントへ伝達される。ulFileType および ulTotalLength メンバはファイル全体に対応するものである。ulStart メンバは、転送されるデータがファイル内のどこに位置するかを示すオフセット値である。ulLength メンバは、そのとき転送されるデータのサイズである。データが複数回に分けて転送される場合、それぞれの転送は先頭から終端まで正しい順番で行われなければならない。

ディスク上にあるファイルを転送する場合は、fDiskFile メンバを TRUE にセットし、データ転送コールバック関数の引数 pData には NkMAIDString 構造体へのポインタをセットする。この構造体には、転送しようとするファイルのフルパス（ファイル名を含む）をセットする。ディスク上のファイルを転送する場合、ulStart メンバに 0 を入れなければならない。モジュールにファイルサイズがわかっている場合、ulLength および ulTotalLength にはファイル全体のサイズを入れる。ファイルサイズがわからない場合は、両者共に 0 を入れる。モジュールは、予めクライアントが kNkMAIDCapability_AcceptDiskAcquisition キャパビリティで転送場所を設定していない限り、ディスクファイルとして転送してはならない。

モジュールは、fRemoveObject を TRUE にセットすることで、クライアントがデータの受信を完了後そのデータオブジェクトを削除するよう要求できる。クライアントは、この要求通りに動作する義務はない。分割してデータが転送される場合、最後の転送時のみこのフラグを TRUE にセットしなければならない。さもなければ、データが失われることになるかもしれない。クライアントは kNkMAIDCapability_Remove を使ってデータオブジェクトを削除してもよい。アイテムオブジェクトに含まれるデータオブジェクトがそれ一つだけの場合、モジュールはアイテムオブジェクトを削除しても良い。この場合、モジュールはソースオブジェクトに対し kNkMAIDEvent_RemoveChild イベントを送らなければならない。

6 戻り値

これらの値のうちのいずれかが、エントリポイント関数から戻される。完了コールバック関数へも同じ値が送られる。

6.1 kNkMAIDResult_NotSupported

モジュールがこの値を返すのは以下の場合である。クライアントが指定されたオブジェクトに対して、存在しない capability における操作をしようとした場合。もしくはクライアントが、capability に対してサポートされていない操作を行おうとした場合。

6.2 kNkMAIDResult_UnexpectedDataType

クライアントがエントリポイント関数へ渡す ulDataType パラメータを、コマンドもしくは capability に対して不適切な型で渡した場合に、モジュールはこの値を返す。

6.3 kNkMAIDResult_ValueOutOfBounds

クライアントが、capability を許されている範囲外に設定しようとした場合に、モジュールはこの値を返す。

6.4 kNkMAIDResult_BufferSize

モジュールがこの値を返すのは、以下の 2 つの場合のみである。クライアントが *kNkMAIDCommand_GetCapInfo* コマンドを送信して、カウントが capabilities の数と一致しなかったとき。クライアントが、配列 capability に対して *kNkMAIDCommand_CapGet* を送ったとき、配列構造体で指定されているサイズが、capability に対するデータサイズと一致しなかった場合。

6.5 kNkMAIDResult_Aborted

クライアントが、*kNkMAIDCommand_Abort*, *kNkMAIDCommand_AbortToMark* もしくは *kNkMAIDCommand_Close* コマンドを非同期のコマンドオブジェクトに対して送信したとき、モジュールは非同期コマンドに対してこの値を返す。

6.6 kNkMAIDResult_NoMedia

クライアントが、取得、オートフォーカス、イジェクトもしくはデバイス中にいくつかのメディアがなければならぬ他の capability 処理を開始しようとしたとき、モジュールはこの値を返す。

6.7 kNkMAIDResult_NoEventProc

クライアントが、*kNkMAIDCapability_EventProc* capability を NULL 以外の値で最初に設定せずに *kNkMAIDCommand_EnumChildren* コマンドを送った場合、モジュールはこの値を返す。

6.8 kNkMAIDResult_ZombieObject

クライアントが、オブジェクトがもはや有効でないために、完了させることができないコマンドを送ろうとした場合、モジュールはこの値を返す。

6.9 kNkMAIDResult_NoError

コマンドが正常に実行された場合に、モジュールはこの値を返す。

6.10 kNkMAIDResult_Pending

クライアントが、コマンドに対する完了コールバック関数を特定して、モジュールがコマンドが完了する前に、クライアントに制御を返す場合に、モジュールはこの値を返す。

6.11 kNkMAIDResult_OrphanedChildren

クライアントが、オブジェクトの子がまだ開いている間に、そのオブジェクトを閉じようとしたときに、モジュールはこの値を返す。

6.12 kNkMAIDResult_NoDataProc

クライアントがオブジェクトに対して DataProc を指定しないで、*kNkMAIDCapability_Acquire* により取り込みを開始した場合に、モジュールはこの値を返す。

6.13 kNkMAIDResult_OutOfMemory

メモリ不足のため、なんらかの動作が成功しなかった場合に、モジュールはこの値を返す。

6.14 kNkMAIDResult_UnexpectedError

予期せぬエラーにより、なんらかの動作が成功しなかった場合、モジュールはこの値を返す。

6.15 kNkMAIDResult_HardwareError

ハードウェアのエラーにより、なんらかの動作が成功しなかった場合、モジュールはこの値を返す。

6.16 kNkMAIDResult_MissingComponent

必要とされるファイルが見つからなかった、開くことができなかった、もしくはアクセスすることができなかったために、作業が完了しなかった場合、モジュールはこの値を返す。

7 イベント

イベントはクライアントにとってはオプションであるが、モジュールにとってはそうではない。イベントコールバック関数を通じて、クライアントが通知されるすべての状態は、can be deduced by polling various elements (?)

7.1 kNkMAIDEvent_AddChild

クライアントは、*kNkMAIDCapability_Children* capability をポーリングすることにより、このイベントを推測することができる。

モジュールは、新しい子ソース、アイテムもしくはデータ型のオブジェクトの追加を検知した場合には、それぞれ親のモジュール、ソースもしくはアイテムオブジェクトに対して、このイベントを送る。イベントがモジュールもしくはソースに対して送られる場合には、data パラメータは新しい子の ID となる。イベントがアイテムに送られる場合には、data パラメータは eNkMAIDDataType のうちの一つとなる。

クライアントは、モジュールがオブジェクトのすべての子オブジェクトを列挙するように、要求するために

kNkMAIDCommand_EnumChildren コマンドを送ることができる。モジュールは、それぞれの子のオブジェクトに *kNkMAIDEvent_AddChild* を送る。イベントコールバック関数が存在しない場合、モジュールはコマンドに対して *kNkMAIDResult_NoEventProc* を返す。

7.2 kNkMAIDEvent_RemoveChild

クライアントは、*kNkMAIDCapability_Children* capability をポーリングすることにより、このイベントを推測することができる。

モジュールは、子ソース、アイテムもしくはデータ型のオブジェクトの削除を検知した場合には、それぞれ親のモジュール、ソースもしくはアイテムオブジェクトに対して、このイベントを送る。イベントがモジュールもしくはソースに対して送られる場合には、data パラメータは新しい(?)子の ID となる。イベントがアイテムに送られる場合には、data パラメータは *eNkMAIDDataType* のうちのひとつとなる。

このイベントを親オブジェクトに送る前に、最初にモジュールは、子オブジェクトに対して現在処理されている非同期コマンドをすべて中断する。そして、子オブジェクトの *kNkMAIDCapability_Alive* capability を FALSE に設定する。

7.3 kNkMAIDEvent_WarmingUp

クライアントは、*kNkMAIDCapability_WarmedUp* capability を吟味することにより、このイベントを推測することができる。

デバイスが最高の品質を保証できない状態にあるときに、モジュールはソースオブジェクトに対して、このイベントを送る。例えば光源が最初に電源投入されたときなど。

7.4 kNkMAIDEvent_WarmedUp

クライアントは、*kNkMAIDCapability_WarmedUp* capability を吟味することにより、このイベントを推測することができる。

デバイスが最高の品質を保証できない状態から抜け出したときに、モジュールはソースオブジェクトに対して、このイベントを送る。保証できない例としては、光源が最初に電源投入されたときなど。

7.5 kNkMAIDEvent_CapChange

クライアントは、capability の数と値を吟味することにより、このイベントを推測することができる。

モジュールは、Capability の数もしくは存在する Capability のどれかの値が変化した場合に、このイベントをモジュール、ソース、アイテムもしくはデータオブジェクトに対して送る。通常、クライアントが *kNkMAIDCommand_CapSet* コマンドによって単一の Capability の値を設定した場合には、このイベントは不要である。しかしながら、クライアントに指定された以外の Capability が *kNkMAIDCommand_CapSet* コマンドによって影響を受ける場合には、このイベントが送られなければならない。

このイベントが単一の Capability の変化を示す場合には、data パラメータは変化した Capability の ID となる。複数の Capability の変化もしくは有効な Capability の数が変化した場合には、data パラメータは NULL となる。いくつかの Capability の値が変化した場合に、モジュールはそれぞれの Capability に対して一つの CapChange イベントを発行するか、もしくは data を NULL にして単一の CapChange イベントを発行

することになる。

7.6 kNkMAIDEvent_OrphanedChildren

クライアントは、子オブジェクトの *kNkMAIDCapability_Alive* capability を吟味することにより、このイベントを推測することができる。

モジュールはこのイベントを、閉じられようとしているオブジェクトが、まだ開かれている子オブジェクトを持っていることを通知するために、このイベントを送る。

7.7 kNkMAIDEvent_CapChangeValueOnly

クライアントは、キャパビリティ ID と値からこのイベントを推定することができる。

キャパビリティの値が変わった時、モジュールはこのイベントをモジュール、ソース、アイテム、データオブジェクトに対して送る。このイベントは、値以外の性質（配列の要素数、列挙値、visibility・Invalid 属性等）は変わっていないことを暗に示している。値以外の性質が変わった場合は、*kNkMAIDEvent_CapChange* を送らなければならない。通常、クライアントが一つのキャパビリティに対して *kNkMAIDCommand_CapSet* コマンドで値をセットした場合、このイベントは不要である。しかし、その *kNkMAIDCommand_CapSet* コマンドが他のキャパビリティに影響する場合は、イベントを送らなければならない。

このイベントで一つのキャパビリティの変化を知らせる場合、**data** パラメータは、変化したキャパビリティの ID である。このイベントで多数のキャパビリティの変化を知らせる場合、**data** パラメータは NULL とする。幾つかのキャパビリティの値が変化した場合、モジュールは、それぞれのキャパビリティへ *CapChangeValueOnly* を送るか、**data** = NULL で一度だけ送るか選択することができる。

8 コマンド

ここで記述されるコマンドのそれぞれについて、MAID entry point へのパラメータが何になるかに関する説明をおこなう。

すべてのコマンドは、処理するためにはかなりの量の時間を必要とする。これはモジュールに任されている。モジュールに対するある一つの呼び出しについて、同期コマンドとして完了するか、速やかにタスクを戻し、別スレッドもしくは *kNkMAIDCommand_Async* コマンド中で非同期コマンドとして処理するかは、モジュールの判断に任される。コマンドが実行されたとき、コマンドによって提供されている完了関数は、同期もしくは非同期のどちらで実行されようとも呼び出される。クライアントは、完了関数のポインタを特定せずにコマンドが同期的に実行されるように要求することができる。クライアントのコールバック（完了通知もしくはイベント通知など）の処理中に発行された非同期コマンドが、クライアントがコールバックを終了するまでに、処理が完了するかどうかは状況による。これはモジュールの判断に任せられる。

クライアントは関連したいくつかの非同期コマンドを、前回のコマンドが完了するのを待たずに送信する場合がある。一連のコマンドの最後は、*kNkMAIDCommand_Mark* コマンドになる。キュー中のコマンドの一つが失敗した場合、クライアントは、*kNkMAIDCommand_Mark* コマンドを含む全ての他のコマンドを中断するために *kNkMAIDCommand_AbortToMark* コマンド送信することができる。モジュールは、それぞれのコマンドに対して、**nResult** パラメータに *kNkMAIDResult_Aborted* をセットして完了関数を呼び出

す。

問題は、あるモジュールがいくつかの同期コマンドを実行しているときに、他が非同期だった場合に発生する。もし同期コマンドよりも、非同期で処理されるコマンドが先に処理されてしまう場合、モジュールは新しいコマンドをすぐに処理することが安全かどうかを判断しなければならない。モジュールが新しいコマンドの処理を遅らせることにした場合、前のコマンドが完了するのを待った後に、新しいコマンドを同期的に処理するのか、非同期として処理される新しいコマンドを、キューの前のコマンドの後に追加するのかを判断しなければならない。

配列機能については、2つのコマンドがデータを取得するために送信されなければならない。どれだけのメモリを確保すればよいのかを知るには、クライアントは最初に *kNkMAIDCommand_CapGet* コマンドを送らなければならない。モジュールは *NkMAIDArray* 構造体のすべてのメンバをセットする。一度、クライアントが目盛りを確保して、*NkMAIDArray* 構造体の *pData* メンバをセットしたら、構造体は *kNkMAIDCommand_CapGetArray* コマンドとして送信される。これらの呼び出しの間にデータのサイズが変化した場合には、モジュールは *pData* メンバには何もデータを入力せずに *kNkMAIDResult_BufferSize* を返す。クライアントは別の *kNkMAIDCommand_CapGet* コマンドを送信することで、処理を再度開始しなければならない。

8.1 kNkMAIDCommand_Async

このコマンドは、シングルスレッドのモジュールにおける特定のオブジェクトに対する非同期コマンドを処理する。

pObject	NULLもしくはモジュール、ソース、アイテム、データオブジェクトへの参照となる。
ulParam	無効
ulDataType	無効
data	無効

マルチスレッドもしくは同期のモジュールは、このコマンドに対して単に *kNkMAIDResult_NoError* を返すことができる。

もしモジュールが、*kNkMAIDCapability_AsyncRate* 機能を持っていた場合、クライアントはこのコマンドをアイドル期間中に、特定された間隔で送信する。

モジュールはコマンドを処理中だが、*kNkMAIDCapability_AsyncRate* 機能による特定された間隔ではなくて、コマンドをできる限りすぐに受信したいことを示す場合には、*kNkMAIDResult_Pending* を返す。

8.2 kNkMAIDCommand_Open

このコマンドは、特定のオブジェクトの子を開く

pObject	NULLもしくはモジュール、ソース、アイテムへの参照
ulParam	NULL、ソースもしくはアイテムのID、もしくは開かれるべきデータオブジェクトの型
ulDataType	<i>kNkMAIDDataType_ObjectPtr</i> でなければならない。
data	<i>NkMAIDObject</i> 構造体へのポインタでなければならない。

このコマンドはモジュールがロードされた後に、最初に送られるコマンドである。pObject パラメータが NULL の場合、開かれたオブジェクトはモジュールオブジェクトで、まだ初期化されていなければ、自分自身で初期化を行う。pObject パラメータがモジュール、ソースもしくはアイテムを参照する場合、開かれたオブジェクトは、それぞれソース、アイテムもしくはデータオブジェクトである。

クライアントは、モジュールを呼び出す前に、data パラメータに渡される *NkMAIDObject* 構造体の *refClient*

メンバをセットする。モジュールは、状態、ポインタ、ハンドル、ID および他の識別子を維持するために、すべての内部的な構造体を、NkMAIDObject 構造体の refModule メンバに確保する。モジュールは、refClient を除いた NkMAIDObject の構造体メンバをすべてセットする。オブジェクトが開いている間、クライアントは refClient メンバの値を、モジュールは refModule の値を変更しない。2つのオブジェクトが refClient もしくは refModule に対して同じ値を持つことはできない。

このコマンドは、同じモジュール、ソース、アイテムもしくはデータオブジェクトを開くために、同じクライアントによって一度以上使用されることがある。モジュールは独立した内部構造体と状態を、それぞれについて保持する。

コマンドが正常に終了した場合は、クライアントはモジュールを解放する前に、オブジェクトを閉じなければならない。

8.3 kNkMAIDCommand_Close

このコマンドは、特定されたモジュール、ソース、アイテムもしくはデータオブジェクトとの接続を閉じる。

pObject	モジュール、ソース、アイテムもしくはデータオブジェクトへの参照
ulParam	無効
ulDataType	無効
data	無効

モジュールはオブジェクトに対して非同期で処理中のコマンドをすべて中断する。モジュールは、NkMAIDObject 構造体の refModule メンバを NULL にセットする。クライアントはオブジェクトを再度開くことなしに構造体を再度使用することはない。

閉じられようとしているオブジェクトの子オブジェクトについて、クライアントがまだすべてを閉じていない場合、モジュールは *kNkMAIDEvent_OrphanedChildren* イベントを送る。もしイベントコールバック関数が存在しないか、イベントコールバック関数がすべての子オブジェクトを閉じない場合、モジュールは *kNkMAIDResult_OrphanedChildren* を返す。

8.4 kNkMAIDCommand_GetCapCount

このコマンドは、指定されたモジュール、ソース、アイテムもしくはデータオブジェクトのおける使用可能な機能の数を取得する。

pObject	モジュール、ソース、アイテムもしくはデータオブジェクトへの参照
ulParam	無効
ulDataType	kNkMAIDDataType_UnsignedPtr でなければならない
data	32bitの符号無し整数へのポインタでなければならない。

8.5 kNkMAIDCommand_GetCapInfo

このコマンドは、指定されたモジュール、ソース、アイテムもしくはデータオブジェクトのおける使用可能なすべての機能の情報を取得する。

pObject	モジュール、ソース、アイテムもしくはデータオブジェクトへの参照
ulParam	保存することのできる NkMAIDCapInfo 構造体の数
ulDataType	kNkMAIDDataType_CapInfoPtr でなければならない
data	NkMAIDCapInfo 構造体の配列へのポインタでなければならない。

配列のサイズは、ulParam パラメータと一致していなければならない。もし異なっていた場合には、モジュ

ールは *kNkMAIDResult_BufferSize* を返す。

8.6 kNkMAIDCommand_CapStart

このコマンドは、指定されたモジュール、ソース、アイテムもしくはデータオブジェクトにおける、指定された機能を開始する。

pObject モジュール、ソース、アイテムもしくはデータオブジェクトへの参照
ulParam 開始される機能のID
ulDataType 無効
data 無効

機能の *NkMAIDCapInfo* 構造体において、*ulType* メンバは *kNkMAIDCapType_Process* でなければならず、*ulOperations* メンバは、ビットが *kNkMAIDCapOperation_Start* セットされていなければならない。

8.7 kNkMAIDCommand_CapSet

このコマンドは、指定されたモジュール、ソース、アイテムもしくはデータオブジェクトにおける、指定された機能の値をセットする。

pObject モジュール、ソース、アイテムもしくはデータオブジェクトへの参照
ulParam セットされる機能のID
ulDataType *eNkMAIDDataType* のうちのひとつ
data 値もしくはポインタ

機能の *NkMAIDCapInfo* 構造体において、*ulOperations* メンバは *kNkMAIDCapOperation_Set* ビットがセットされていなければならない。*ulDataType* に許される値は、以下の表に示すように、機能の *NkMAIDCapInfo* 構造体の *ulType* メンバに依存する。

<u>ulType value</u>	<u>ulDataType value</u>
<i>kNkMAIDCapType_Boolean</i>	<i>kNkMAIDDataType_Boolean</i> , <i>kNkMAIDDataType_BooleanPtr</i>
<i>kNkMAIDCapType_Integer</i>	<i>kNkMAIDDataType_Integer</i> , <i>kNkMAIDDataType_IntegerPtr</i>
<i>kNkMAIDCapType_Unsigned</i>	<i>kNkMAIDDataType_Unsigned</i> , <i>kNkMAIDDataType_UnsignedPtr</i>
<i>kNkMAIDCapType_Float</i>	<i>kNkMAIDDataType_FloatPtr</i>
<i>kNkMAIDCapType_Point</i>	<i>kNkMAIDDataType_PointPtr</i>
<i>kNkMAIDCapType_Size</i>	<i>kNkMAIDDataType_SizePtr</i>
<i>kNkMAIDCapType_Rect</i>	<i>kNkMAIDDataType_RectPtr</i>
<i>kNkMAIDCapType_String</i>	<i>kNkMAIDDataType_StringPtr</i>
<i>kNkMAIDCapType_DateTime</i>	<i>kNkMAIDDataType_DateTimePtr</i>
<i>kNkMAIDCapType_Callback</i>	<i>kNkMAIDDataType_CallbackPtr</i> , <i>kNkMAIDDataType_Null</i>
<i>kNkMAIDCapType_Array</i>	<i>kNkMAIDDataType_ArrayPtr</i> , <i>kNkMAIDDataType_Unsigned</i>
<i>kNkMAIDCapType_Enum</i>	<i>kNkMAIDDataType_EnumPtr</i> , <i>kNkMAIDDataType_Unsigned</i>
<i>kNkMAIDCapType_Range</i>	<i>kNkMAIDDataType_RangePtr</i> , <i>kNkMAIDDataType_Unsigned</i>
<i>kNkMAIDCapType_Generic</i>	<i>kNkMAIDDataType_GenericPtr</i>
<i>kNkMAIDCapType_BoolDefault</i>	<i>kNkMAIDDataType_UnsignedPtr</i>

kNkMAIDCapType_Process 型では、このコマンドは使用できない。

データ型がこの表と一致しない場合、モジュールは *kNkMAIDResult_UnexpectedDataType* を返す。もし機能がこのコマンドをサポートしなければ、モジュールは *kNkMAIDResult_NotSupported* を返す。

8.8 kNkMAIDCommand_CapGet

このコマンドは、指定されたモジュール、ソース、アイテムもしくはデータオブジェクトにおける、指定された機能の値を得る。

pObject	モジュール、ソース、アイテムもしくはデータオブジェクトへの参照
ulParam	受信する機能のID
ulDataType	eNkMAIDDataTypeのうちの一つ
data	ポインタ

機能の NkMAIDCapInfo 構造体において、ulOperations メンバは *kNkMAIDCapOperation_Get* ビットがセットされていなければならない。ulDataType に許される値は、以下の表に示すように、機能の NkMAIDCapInfo 構造体の ulType メンバに依存する。

<u>ulType value</u>	<u>ulDataType value</u>
<i>kNkMAIDCapType_Boolean</i>	<i>kNkMAIDDataType_BooleanPtr</i>
<i>kNkMAIDCapType_Integer</i>	<i>kNkMAIDDataType_IntegerPtr</i>
<i>kNkMAIDCapType_Unsigned</i>	<i>kNkMAIDDataType_UnsignedPtr</i>
<i>kNkMAIDCapType_Float</i>	<i>kNkMAIDDataType_FloatPtr</i>
<i>kNkMAIDCapType_Point</i>	<i>kNkMAIDDataType_PointPtr</i>
<i>kNkMAIDCapType_Size</i>	<i>kNkMAIDDataType_SizePtr</i>
<i>kNkMAIDCapType_Rect</i>	<i>kNkMAIDDataType_RectPtr</i>
<i>kNkMAIDCapType_String</i>	<i>kNkMAIDDataType_StringPtr</i>
<i>kNkMAIDCapType_DateTime</i>	<i>kNkMAIDDataType_DateTimePtr</i>
<i>kNkMAIDCapType_Callback</i>	<i>kNkMAIDDataType_CallbackPtr</i>
<i>kNkMAIDCapType_Array</i>	<i>kNkMAIDDataType_ArrayPtr</i>
<i>kNkMAIDCapType_Enum</i>	<i>kNkMAIDDataType_EnumPtr</i>
<i>kNkMAIDCapType_Range</i>	<i>kNkMAIDDataType_RangePtr</i>
<i>kNkMAIDCapType_Generic</i>	<i>kNkMAIDDataType_GenericPtr</i>
<i>kNkMAIDCapType_BoolDefault</i>	<i>kNkMAIDDataType_UnsignedPtr</i>

kNkMAIDCapType_Process 型では、このコマンドは使用できない。

データ型がこの表と一致しない場合、モジュールは *kNkMAIDResult_UnexpectedDataType* を返す。もし機能がこのコマンドをサポートしなければ、モジュールは *kNkMAIDResult_NotSupported* を返す。

8.9 kNkMAIDCommand_CapGetDefault

このコマンドは、指定されたモジュール、ソース、アイテムもしくはデータオブジェクトにおける、指定された機能のデフォルト値を得る。

pObject	モジュール、ソース、アイテムもしくはデータオブジェクトへの参照
ulParam	受信する機能のID
ulDataType	eNkMAIDDataTypeのうちの一つ
data	ポインタ

機能の NkMAIDCapInfo 構造体において、ulOperations メンバは *kNkMAIDCapOperation_GetDefault* ビットがセットされていなければならない。ulDataType に許される値は、以下の表に示すように、機能の NkMAIDCapInfo 構造体の ulType メンバに依存する。

<u>ulType value</u>	<u>ulDataType value</u>
<i>kNkMAIDCapType_Boolean</i>	<i>kNkMAIDDataType_BooleanPtr</i>
<i>kNkMAIDCapType_Integer</i>	<i>kNkMAIDDataType_IntegerPtr</i>
<i>kNkMAIDCapType_Unsigned</i>	<i>kNkMAIDDataType_UnsignedPtr</i>
<i>kNkMAIDCapType_Float</i>	<i>kNkMAIDDataType_FloatPtr</i>
<i>kNkMAIDCapType_Point</i>	<i>kNkMAIDDataType_PointPtr</i>
<i>kNkMAIDCapType_Size</i>	<i>kNkMAIDDataType_SizePtr</i>
<i>kNkMAIDCapType_Rect</i>	<i>kNkMAIDDataType_RectPtr</i>
<i>kNkMAIDCapType_Generic</i>	<i>kNkMAIDDataType_GenericPtr</i>

kNkMAIDCapType_Process, *kNkMAIDCapType_String*, *kNkMAIDCapType_DateTime*,
kNkMAIDCapType_Callback, *kNkMAIDCapType_Array*, *kNkMAIDCapType_Enum* および

kNkMAIDCapType_Range 型では、このコマンドは使用できない。

データ型がこの表と一致しない場合、モジュールは *kNkMAIDResult_UnexpectedDataType* を返す。もし機能がこのコマンドをサポートしなければ、モジュールは *kNkMAIDResult_NotSupported* を返す。

8.10 kNkMAIDCommand_CapGetArray

このコマンドは、指定されたモジュール、ソース、アイテムもしくはデータオブジェクトにおける、指定された配列機能に関連したデータを得る。

pObject	モジュール、ソース、アイテムもしくはデータオブジェクトへの参照
ulParam	データを得る配列機能のID
ulDataType	<i>kNkMAIDDataType_ArrayPtr</i> , <i>kNkMAIDDataType_EnumPtr</i> でなければならない。
data	NkMAIDArray もしくは NkMAIDEnum 構造体へのポインタでなければならない。

機能の NkMAIDCapInfo 構造体において、ulType メンバは *kNkMAIDCapType_Array* もしくは *kNkMAIDCapType_Enum* で、ulOperations メンバは *kNkMAIDCapOperation_GetArray* ビットがセットされていなければならない。NkMAIDArray もしくは NkMAIDEnum 構造体のすべてのメンバは変更されない。モジュールは pData メンバによって指定されているアドレスへデータを保存するだけである。機能がこのコマンドをサポートしていない場合には、モジュールは *kNkMAIDResult_NotSupported* を返す。モジュールが保存しようとするものと NkMAIDArray もしくは NkMAIDEnum 構造体のメンバが一致しない場合には、モジュールは *kNkMAIDResult_BufferSize* を返す。

8.11 kNkMAIDCommand_Mark

このコマンドは、指定されたモジュール、ソース、アイテムおよびデータオブジェクトに対するキューにマークを挿入する。

pObject	モジュール、ソース、アイテムもしくはデータオブジェクトへの参照
ulParam	無効
ulDataType	無効
data	無効

このコマンドは、モジュールによって非同期に実行されるコマンドをよりよくサポートするためのものである。モジュールは何の操作を実行する必要はないが、指定されたオブジェクトに対するすべての非同期コマンドが完了するまでは、このコマンドは完了しない。完了関数は、このコマンドに関しては提供されない。

8.12 kNkMAIDCommand_AbortToMark

このコマンドは、指定されたモジュール、ソース、アイテムおよびデータオブジェクトに対するキューにある非同期コマンドを中断する。これには次の *kNkMAIDCommand_Mark* コマンドも含む。

pObject	モジュール、ソース、アイテムもしくはデータオブジェクトへの参照
ulParam	無効
ulDataType	無効
data	無効

このコマンドはモジュールによって処理される非同期コマンドをよりよくサポートするためのものである。

クライアントは、いくつかの関連した非同期コマンドを、前の終了を待たずに発行することがある。一連のうちの最後は、*kNkMAIDCommand_Mark* コマンドになる。もしキュー内のコマンドが失敗した場合、クライアントは *kNkMAIDCommand_Mark* コマンドを含めた他のコマンドを中断するために、*kNkMAIDCommand_AbortToMark* コマンドを送ることができる。モジュールはそれぞれのコマンドについ

て `nResult` パラメータを `kNkMAIDResult_Aborted` にセットして完了関数を呼び出す。

もし `kNkMAIDCommand_Mark` コマンドがなければ、すべての非同期コマンドが中断される。

指定されたオブジェクトに送られたコマンドだけが中断される。

8.13 kNkMAIDCommand_Abort

このコマンドは、現在処理されている非同期コマンドを中断する。

pObject	モジュール、ソース、アイテムもしくはデータオブジェクトへの参照
ulParam	無効
ulDataType	無効
data	無効

このコマンドはモジュールによって処理される非同期コマンドをよりよくサポートするためのものである。モジュールはコマンドの完了関数を `kNkMAIDResult_Aborted` で呼び出す。

指定されたオブジェクトに送られたコマンドだけが中断される。

8.14 kNkMAIDCommand_EnumChildren

モジュールはオブジェクトに対して、そのすべての子のための `kNkMAIDEvent_AddChild` イベントを送る。

pObject	モジュール、ソース、アイテムもしくはデータオブジェクトへの参照
ulParam	無効
ulDataType	無効
data	無効

このコマンドは、クライアントが、`kNkMAIDCapability_EventProc` 機能を `NULL` 以外の値にセットしていない場合には、何も行わない。

8.15 kNkMAIDCommand_GetParent

モジュールはオブジェクトの親に関する情報を得る。

pObject	モジュール、ソース、アイテムもしくはデータオブジェクトへの参照
ulParam	無効
ulDataType	<code>kNkMAIDDataType_ObjectPtr</code> でなければならない
data	<code>NkMAIDObject</code> 構造体へのポインタでなければならない

モジュールは、オブジェクトの親の `NkMAIDObject` 構造体の値と一致させるために、`NkMAIDObject` 構造体のメンバを設定する。クライアントは、モジュールへの二次的な呼び出しにおいて、この構造体を使用する。しかしながら、すべての、親オブジェクトのために持つ持続的なデータが有効でありつづけることを保証するのは、クライアントの義務である。

8.16 kNkMAIDCommand_ResetToDefault

モジュールはオブジェクトをデフォルトに戻す。

pObject	モジュール、ソース、アイテムもしくはデータオブジェクトへの参照
ulParam	無効
ulDataType	無効
data	無効

モジュールは、オブジェクトが持つ全てのキャパビリティをリセットし、それぞれのデフォルト値に戻す。選択されたオブジェクトが子オブジェクトまたはデータオブジェクトをオープンしている場合、それらのキャパビリティも同様にリセットされる。

9 Capabilities

ここで記述される機能のそれぞれについて、NkMAIDCapInfo のメンバが何になるかに関する説明をおこなう。

ここでとりあげる機能のほとんどは、クライアントによって明示的に取り扱われる。残りの機能とベンダー機能については、一般的な方法で取り扱われる。クライアントはこれらをユーザに説明するために ulVisibility および szDescription メンバを使用する。

モジュールは、グループとして扱われるべき Capability のセットを指定することができる。グループに含まれるそれぞれの Capability は、ulVisibility メンバに *kNkMAIDCapVisibility_GroupMember* 値を含まなければならない。これらの Capability をグループ化するために、新しい group Capability が生成されなければならない。この ulVisibility メンバは *kNkMAIDCapVisibility_Group* 値を含まなければならない。この新しい Capability は *kNkMAIDCapType_Array* 型の Capability で、他の Capability の ID の配列を含む。クライアントが、*kNkMAIDCommand_CapGet* コマンドによりこの Capability を読み込んだ場合には、モジュールは以下に示すように NkMAIDArray 構造体をセットする。

```

1  ul Type = kNkMAIDArrayType_Unsigned
2  ul Elements = <number of member-capabilities in this group>
3  ul DimSize1 = <same as ul Elements>
4  ul DimSize2 = 0
5  ul DimSize3 = 0
6  wPhysicalBytes = 4
7  wLogicalBits = 32
8  pData = NULL

```

データを保持するのに十分なメモリを確保した後、クライアントはモジュールに対して *kNkMAIDCommand_CapGetArray* コマンドを呼び出す。戻り時には、pData メンバは ULONG 値の配列を含んでいなければならない。それぞれの値は、他の Capability の ID となる。

9.1 kNkMAIDCapability_AsyncRate

モジュールはこの機能を、クライアントがアイドルプロセス中に送信する *kNkMAIDCommand_Async* コマンドの周期についての推奨値を知らせるために使用する。これはコマンド中でミリ秒単位で表現される。

Object types モジュールのみ
ulType *kNkMAIDCapType_Unsigned*
ulOperations *kNkMAIDCapOperation_Get*

もしモジュールが周期的な *kNkMAIDCommand_Async* コマンドを必要としないときは、この機能を提供してはならない。これはマルチスレッドのモジュールの場合などが考えられる。

周期は、モジュールからクライアントへの提案に過ぎない。クライアントは、モジュールが望むほどに早くは、コマンドを送信することはできないかもしれない。

詳細については、*kNkMAIDCommand_Async* コマンドの記述を参照のこと。

9.2 kNkMAIDCapability_ProgressProc

モジュールは、長い処理の間にこのコールバックを呼び出すことができる。

Object types モジュール、ソース、アイテムもしくはデータオブジェクト
ulType *kNkMAIDCapType_Callback*
ulOperations *kNkMAIDCapOperation_Get, kNkMAIDCapOperation_Set*

オブジェクトに対して発行されたコマンドが、著しく時間がかかる場合には、クライアントがユーザに対して進行表示を提供できるようにするために、モジュールはこのコールバックを呼び出す。どのくらいの間隔

で、もしくはまったく呼び出さないかどうかは、モジュールに任されている。

初期値は NULL である。クライアントは、MAID entry point の `ulDataType` パラメータに `kNkMAIDDataType_Null` をセットして、この機能を設定することにより、進行情報を必要としないことを示すことができる。

詳細については、MAIDProgress コールバック関数の記述を参照すること。

9.3 kNkMAIDCapability_EventProc

モジュールはクライアントのイベントを通知するために、このコールバックルーチン呼び出す。

Object types モジュール、ソース、アイテムもしくはデータオブジェクト
ulType `kNkMAIDCapType_Callback`
ulOperations `kNkMAIDCapOperation_Get`, `kNkMAIDCapOperation_Set`

初期値は NULL である。クライアントは、MAID entry point の `ulDataType` パラメータに `kNkMAIDDataType_Null` をセットして、この機能を設定することにより、イベント通知を必要としないことを示すことができる。

詳細については、MAIDEvent コールバック関数の記述を参照すること。

9.4 kNkMAIDCapability_DataProc

モジュールは、クライアントに対してデータを渡すために、このコールバック呼び出す。

Object types データオブジェクトのみ
ulType `kNkMAIDCapType_Callback`
ulOperations `kNkMAIDCapOperation_Get`, `kNkMAIDCapOperation_Set`

モジュールは、データオブジェクトに対して、この機能を提供することを要求する。

初期値は NULL である。クライアントは、取得を開始する前に、この機能をセットしなければならない。一度、データが正常に取り交わされれば、クライアントは MAID entry point への `ulDataType` パラメータを `kNkMAIDDataType_Null` にセットすることにより、この機能を設定することができる。

詳細については、MAIDData コールバック関数の記述を参照すること。

9.5 kNkMAIDCapability_UIRequestProc

モジュールは、いくつかのユーザインターフェイスを表示させるように要求するために、このコマンドを使用できる。

Object types モジュールのみ
ulType `kNkMAIDCapType_Callback`
ulOperations `kNkMAIDCapOperation_Get`, `kNkMAIDCapOperation_Set`

モジュールは、モジュールオブジェクトに対して、この機能を提供するように要求される。クライアントは、モジュールが開かれた直後に、この機能を設定しなければならない。もし設定されていなければ、モジュールはユーザに対して通知したり、ユーザに対して問い合わせをすることはできない。

初期値は NULL である。

詳細については、MAIDUIRequest コールバック関数の記述を参照すること。

9.6 kNkMAIDCapability_IsAlive

これはオブジェクトの有効性の状態である。

Object types モジュール、ソース、アイテムもしくはデータオブジェクト
ulType *kNkMAIDCapType_Boolean*
ulOperations *kNkMAIDCapOperation_Get*

モジュールは、すべてのオブジェクトに対して、この capability を用意することを求められる。

この capability の値は通常 TRUE である。これが FALSE になるのは、オブジェクトがモジュールによって削除されているか、クライアントによってオブジェクトの親が閉じられているときである。

9.7 kNkMAIDCapability_Children

これは、子ソースのリストもしくはアイテムの ID である。

Object types モジュールもしくはソース
ulType *kNkMAIDCapType_Enum*
ulOperations *kNkMAIDCapOperation_Get, kNkMAIDCapOperation_GetArray*

モジュールは、モジュールもしくはソースオブジェクトに対して、この機能を提供するように要求される。

NkMAIDArray 構造体において、ulType は *kNkMAIDArrayType_Unsigned*、wPhysicalBytes は 4 となる。

9.8 kNkMAIDCapability_State

クライアントは、オブジェクトの状態を保存するために、この機能を使用することができる。

Object types モジュール、ソース、アイテムもしくはデータオブジェクト
ulType *kNkMAIDCapType_Array*
ulOperations *kNkMAIDCapOperation_GetArray, kNkMAIDCapOperation_Get, kNkMAIDCapOperation_Set*
 配列内のデータは、完全にモジュールに依存しており、クライアントは関与しない。データはクライアントによって、そのまま保存もしくは復元される。

NkMAIDArray 構造体内において、ulType は *kNkMAIDArrayType_Unsigned* で、wPhysicalBytes は 1 で、wLogicalBits は 8 である。

9.9 kNkMAIDCapability_Name

これはオブジェクトの名前である。

Object types モジュール、ソース、アイテムもしくはデータオブジェクト
ulType *kNkMAIDCapType_String*
ulOperations *kNkMAIDCapOperation_Get*

kNkMAIDCapability_Description 機能と異なり、この機能は設定することはできない。モジュールは、記述名を保存できて、ユーザに名前を編集できるようにしたい場合には、この機能を使用しなければならない。

9.10 kNkMAIDCapability_Description

これは、オブジェクトの説明である。

Object types モジュール、ソース、アイテムもしくはデータオブジェクト
ulType *kNkMAIDCapType_String* or *kNkMAIDCapType_Array*
ulOperations *kNkMAIDCapOperation_Get, kNkMAIDCapOperation_GetArray* for an array type, possibly *kNkMAIDCapOperation_Set*

モジュールは、オブジェクトを *kNkMAIDCapability_Name* 機能よりも詳細に説明できる場合には、この機

能を用意する。これによりユーザは、説明されたオブジェクトを区別しやすくなる。

モジュールは、この機能を配列として実現する。NkMAIDArray 構造体の ulType メンバは、*kNkMAIDArrayType_String* もしくは *kNkMAIDArrayType_PackedString* である。

9.11 kNkMAIDCapability_Interface

これは、ソースとの通信に使用されている物理インターフェイスに関する説明である。

Object types ソースのみ
ulType *kNkMAIDCapType_String*
ulOperations *kNkMAIDCapOperation_Get*

これによりユーザはソースを識別しやすくなる。

9.12 kNkMAIDCapability_DataTypes

これは、アイテムで有効なデータ型、もしくはソースが生成できるデータ型である。

Object types アイテムもしくはソース
ulType *kNkMAIDCapType_Unsigned*
ulOperations *kNkMAIDCapOperation_Get*

値は、eNkMAIDDataObjType 値とのビット方法による組み合わせである。kNkMAIDDataObjType_File をこのキャパビリティで使用してはならない。この値はデータ転送時のみ使用する。

9.13 kNkMAIDCapability_DateTime

これはアイテムの捕捉の日時である。

Object types アイテムのみ
ulType *kNkMAIDCapType_DateTime*
ulOperations *kNkMAIDCapOperation_Get*

この capability は、保存の capabilities とともにデバイスに対して、モジュールによって提供されるのみである。スキャナモジュールはこれを提供しない。

9.14 kNkMAIDCapability_StoredBytes

これは、デバイスに保存される際の、バイト表示によるオブジェクトのサイズである。

Object types アイテムもしくはデータオブジェクト
ulType *kNkMAIDCapType_Unsigned*
ulOperations *kNkMAIDCapOperation_Get*

この capability は、保存の capabilities とともにデバイスに対して、モジュールによって提供されるのみである。

9.15 kNkMAIDCapability_Eject

これはソースデバイスからメディアをイジェクトする。

Object types ソースもしくはアイテム
ulType *kNkMAIDCapType_Process*
ulOperations *kNkMAIDCapOperation_Start*

この capability がソースに対して開始された場合、すべてのメディアがイジェクトされる。アイテムに対して開始された場合は、そのアイテムに対するメディアのみがイジェクトされる。

9.16 kNkMAIDCapability_Feed

これはソースデバイス内にメディアを送り込む。

Object types ソースのみ
ulType *kNkMAIDCapType_Process*
ulOperations *kNkMAIDCapOperation_Start*

送り込まれるべきメディアがない場合には、モジュールは *kNkMAIDResult_NoMedia* を返す。

9.17 kNkMAIDCapability_Capture

これは、ソースデバイスの他のアイテムを捕捉する。

Object types ソースのみ
ulType *kNkMAIDCapType_Process*
ulOperations *kNkMAIDCapOperation_Start*

この処理が正常に終了した場合には、ソースは追加の子アイテムを持つことになる。ソースは再びそのアイテムを列挙しなければならない。

9.18 kNkMAIDCapability_Mode

これはデータオブジェクトに対する要求モードである。

Object types データオブジェクトのみ
ulType *kNkMAIDCapType_Enum*
ulOperations *kNkMAIDCapOperation_Get, kNkMAIDCapOperation_GetArray, kNkMAIDCapOperation_Set*

どのモードが有効で、どれを意味しているのかを決めるのは、モジュールに任されている。ユーザは文字列の配列の中から選択を行う。

NkMAIDEnum 構造体の ulType メンバは、 *kNkMAIDArrayType_String* もしくは *kNkMAIDArrayType_PackedString* である。

9.19 kNkMAIDCapability_Acquire

これは取得を開始する。

Object types データオブジェクトのみ
ulType *kNkMAIDCapType_Process*
ulOperations *kNkMAIDCapOperation_Start*

モジュールは、データとともにデータオブジェクトのデータ転送コールバックの呼び出しを開始する。モジュールはまた、取得にかなりの量の時間がかかりそうならば、データオブジェクトの進行コールバックを呼び出す場合がある。

9.20 kNkMAIDCapability_Start

取得についての、秒単位での開始位置

Object types 音声もしくはビデオ
ulType *kNkMAIDCapType_Float*
ulOperations *kNkMAIDCapOperation_Get, kNkMAIDCapOperation_Set*

これは、音声もしくはビデオオブジェクトの最初からのオフセットである。この capability は、保存の capabilities とともにデバイスに対して、モジュールによって提供されるのみである。

9.21 kNkMAIDCapability_Length

秒単位での、有効な長さもしくは取得された長さ

Object types 音声もしくはビデオ
ulType *kNkMAIDCapType_Float*
ulOperations *kNkMAIDCapOperation_Get*, possibly *kNkMAIDCapOperation_Set*, possibly *kNkMAIDCapOperation_GetDefault*

デフォルト値は、保存の capability をもつデバイスに対するモジュールからの、有効な合計の長さになる。

9.22 kNkMAIDCapability_SampleRate

取得における一秒あたりのサンプル数

Object types 音声もしくはビデオ
ulType *kNkMAIDCapType_Enum* or *kNkMAIDCapType_Range*
ulOperations *kNkMAIDCapOperation_Get*, possibly *kNkMAIDCapOperation_Set*

NkMAIDEnum 構造体における、ulType メンバは *kNkMAIDArrayType_Float* である。

9.23 kNkMAIDCapability_Stereo

これは、モノラルもしくはステレオのどちらかの型を選択する。

Object types 音声もしくはビデオ
ulType *kNkMAIDCapType_Boolean*
ulOperations *kNkMAIDCapOperation_Get*, *kNkMAIDCapOperation_Set*

モジュールがこの capability を用意していない場合、クライアントはデバイスがモノラルの取得のみが可能であるとする。

9.24 kNkMAIDCapability_Samples

データオブジェクトの現在の状態で予想される、取得されるであろうサンプルの数

Object types 音声もしくはビデオ
ulType *kNkMAIDCapType_Unsigned*
ulOperations *kNkMAIDCapOperation_Get*

9.25 kNkMAIDCapability_Filter

これは、デバイスのライトソースに対してフィルタを選択する。

Object types イメージもしくはサムネイル
ulType *kNkMAIDCapType_Enum*
ulOperations *kNkMAIDCapOperation_Get*, *kNkMAIDCapOperation_GetArray*, *kNkMAIDCapOperation_Set*

NkMAIDArray 構造体において、ulType メンバは *kNkMAIDArrayType_Unsigned* で、wPhysicalBytes は 4、wLogicalBits は 32 となる。配列は eNkMAIDFilter 列挙値を含む。

9.26 kNkMAIDCapability_Prescan

デバイスは固有のメディアに対して自動的に自分自身のセットアップを行う。

Object types イメージもしくはサムネイル
ulType *kNkMAIDCapType_Process*
ulOperations *kNkMAIDCapOperation_Start*

9.27 kNkMAIDCapability_AutoFocus

デバイスは、デバイスのフォーカスを自動的に設定する。

Object types イメージもしくはサムネイル
ulType *kNkMAIDCapType_Process*
ulOperations *kNkMAIDCapOperation_Start*

モジュールは、もし可能ならば、 *kNkMAIDCapability_Focus* の値を更新しなければならない。

9.28 kNkMAIDCapability_AutoFocusPt

これは、モジュールがフォーカス調整がおこなう場所である。

Object types イメージもしくはサムネイル
ulType *kNkMAIDCapType_Point*
ulOperations *kNkMAIDCapOperation_Get, kNkMAIDCapOperation_Set*

モジュールは、一点でのフォーカスをサポートしていなければ、この capability を用意しなくても良い。

9.29 kNkMAIDCapability_Focus

これは、デバイスのフォーカス位置である。

Object types イメージもしくはサムネイル
ulType *kNkMAIDCapType_Enum* or *kNkMAIDCapType_Range*
ulOperations *kNkMAIDCapOperation_Get, kNkMAIDCapOperation_Set*
 NkMAIDEnum 構造体において、ulType は *kNkMAIDArrayType_Float* となる。

9.30 kNkMAIDCapability_Coords

これは、最大解像度のピクセルで表現された、取得される対象領域である。

Object types イメージもしくはサムネイル
ulType *kNkMAIDCapType_Rect*
ulOperations *kNkMAIDCapOperation_Get, kNkMAIDCapOperation_Set, kNkMAIDCapOperation_GetDefault*
 デフォルト値は取得されうる領域のなかで、最も大きい領域となる。

9.31 kNkMAIDCapability_Resolution

これは、pixels/inch 単位での取得される解像度である。

Object types イメージもしくはサムネイル
ulType *kNkMAIDCapType_Enum* or *kNkMAIDCapType_Range*
ulOperations *kNkMAIDCapOperation_Get, kNkMAIDCapOperation_GetArray, kNkMAIDCapOperation_Set*
 NkMAIDEnum 構造体において、ulType メンバは *kNkMAIDArrayType_Float* となる。

9.32 kNkMAIDCapability_Preview

これはスピードもしくは品質の優先度を設定する。

Object types イメージもしくはサムネイル
ulType *kNkMAIDCapType_Boolean*
ulOperations *kNkMAIDCapOperation_Get, kNkMAIDCapOperation_Set*

クライアントが、この capability を TRUE に設定した場合、モジュールはできる限り速く取得するようにしなければならない。FALSE の場合、最高の品質で生成するようにしなければならない。

9.33 kNkMAIDCapability_Negative

これは、もとのメディアがネガもしくはポジのどちらの型かを設定する。

Object types イメージもしくはサムネイル
ulType *kNkMAIDCapType_Boolean*
ulOperations *kNkMAIDCapOperation_Get; kNkMAIDCapOperation_Set*

もしモジュールがこの capability を用意しない場合、クライアントはもとのメディアに対しては何の仮定もおこなわない。

9.34 kNkMAIDCapability_ColorSpace

これはクライアントに対して送られるデータの色空間を選択する。

Object types イメージもしくはサムネイル
ulType *kNkMAIDCapType_Enum*
ulOperations *kNkMAIDCapOperation_Get; kNkMAIDCapOperation_GetArray; kNkMAIDCapOperation_Set*

NkMAIDEnum 構造体において、ulType メンバは *kNkMAIDArrayType_Unsigned* となる。wPhysicalBytes は 4 となる。列挙値は、eNkMAIDColorSpace の enum 値のうちの一つ以上の値を含む。

9.35 kNkMAIDCapability_Bits

これは、カラーあたりの取得されるビット数を選ぶ。

Object types イメージもしくはサムネイル
ulType *kNkMAIDCapType_Enum*
ulOperations *kNkMAIDCapOperation_Get; kNkMAIDCapOperation_Set; kNkMAIDCapOperation_GetArray*

もしモジュールがこの capability を用意しない場合には、クライアントは、デバイスはカラーあたり 8 ビットが要求できると判断する。

9.36 kNkMAIDCapability_Planar

これは、オブジェクトによってサポートされる転送モードを報告するに過ぎない。

Object types イメージもしくはサムネイル
ulType *kNkMAIDCapType_Boolean*
ulOperations *kNkMAIDCapOperation_Get; possibly kNkMAIDCapOperation_Set*

モジュールがデータを planar もしくは interleaved のいずれかで転送したい場合、*kNkMAIDCommand_CapSet* コマンドはサポートしない。

9.37 kNkMAIDCapability_Lut

イメージデータがクライアントに転送される前に適用される、ルックアップテーブルを設定する。

Object types イメージもしくはサムネイル
ulType *kNkMAIDCapType_Array*
ulOperations *kNkMAIDCapOperation_Get; kNkMAIDCapOperation_Set; kNkMAIDCapOperation_GetArray*

NkMAIDArray 構造体において、ulType メンバは *kNkMAIDArrayType_Unsigned* である。カラーイメージについては、配列は 2 つ以上のルックアップテーブルを持ち、順番は現在の色空間に依存する。RGB の場合は 3 つのテーブルで、順番は赤、緑そして青となる。CMYK の場合は 4 つのテーブルで、順番はシアン、マゼンダ、黄色、黒色となる。モノクロイメージについては、ルックアップテーブルは一つのみ存在する。

9.38 kNkMAIDCapability_Transparency

これは、もとのメディアが、透過もしくは反射のいずれかの型かを選択する。

Object types イメージもしくはサムネイル
ulType *kNkMAIDCapType_Boolean*
ulOperations *kNkMAIDCapOperation_Get; kNkMAIDCapOperation_Set*

モジュールがこの capability を用意しない場合、クライアントはもとのメディアに対して何の仮定も行わない。

9.39 kNkMAIDCapability_Threshold

これは、二色イメージの閾値である。

Object types イメージもしくはサムネイル
ulType *kNkMAIDCapType_Range*
ulOperations *kNkMAIDCapOperation_Get; kNkMAIDCapOperation_Set*

9.40 kNkMAIDCapability_Pixels

データオブジェクトの現在の状態から取得できるであろうピクセルの数。

Object types イメージ、サムネイルもしくはビデオ
ulType *kNkMAIDCapType_Size*
ulOperations *kNkMAIDCapOperation_Get*

9.41 kNkMAIDCapability_ForceScan

デバイスによって実行される読み込み動作（モジュールによって決められる）が不要かどうかを決定する。

Object types Data object only
ulType *kNkMAIDCapType_Boolean*
ulOperations *kNkMAIDCapOperation_Get; kNkMAIDCapOperation_Set*

この Capability が TRUE にセットされている場合、kNkMAIDCapability_Acquire が開始されたときには、デバイスは常に物理的な読み込み動作を行う。この Capability が FALSE にセットされている場合、モジュールは現在の状態に基づいて、このような読み込み動作が必要かどうかを決定する。物理的な読み込み動作が必要ない場合には、通常の取得における動作と同じステップを踏まなければならない（データ転送や I/O 完了通知など）。異なるのは、データがデバイスからではなく内部のバッファから供給されるという点だけである。この Capability のデフォルト値は TRUE である。

9.42 kNkMAIDCapability_ForcePrescan

デバイスによって実行されるプリスキャン動作（モジュールによって決められる）が不要かどうかを決定する。

Object types Data object only
ulType *kNkMAIDCapType_Boolean*
ulOperations *kNkMAIDCapOperation_Get; kNkMAIDCapOperation_Set*

この Capability が TRUE にセットされている場合、kNkMAIDCapability_Prescan が開始されたときには、デバイスは常に物理的なプリスキャン動作を行う。この Capability が FALSE にセットされている場合、モジュールは現在の状態に基づいて、このような動作が必要かどうかを決定する。物理的なプリスキャン動作が必要ない場合には、通常のプリスキャンにおける動作と同じステップを踏まなければならない（I/O 完了通知など）。この Capability のデフォルト値は TRUE である。

9.43 kNkMAIDCapability_ForceAutoFocus

デバイスによって実行されるオートフォーカス動作（モジュールによって決められる）が不要かどうかを決定する。

Object types Data object only
ulType *kNkMAIDCapType_Boolean*
ulOperations *kNkMAIDCapOperation_Get; kNkMAIDCapOperation_Set*

この Capability が TRUE にセットされている場合、kNkMAIDCapability_AutoFocus が開始されたときには、デバイスは常に物理的なオートフォーカス動作を行う。この Capability が FALSE にセットされている場合、モジュールは現在の状態に基づいて、このような動作が必要かどうかを決定する。物理的なオートフォーカス動作が必要ない場合には、通常のオートフォーカスにおける動作と同じステップを踏まなければならない（I/O 完了通知など）。この Capability のデフォルト値は TRUE である。

9.44 kNkMAIDCapability_NegativeDefault

ソースオブジェクト用の Capability で、*kNkMAIDCapability_Negative* に設定されるデフォルト値を決定する。

Object types Source object only
ulType *kNkMAIDCapType_Unsigned*
ulOperations *kNkMAIDCapOperation_Get; kNkMAIDCapOperation_Set*

一度この Capability が設定された場合、それ以降はこのソースオブジェクトで生成される全ての Image オブジェクトの、*kNkMAIDCapability_Negative* のデフォルト値として、この Capability の値が使用される。この Capability がソースオブジェクトに存在しない、クライアントに設定されていない、もしくは *kNkMAIDBooleanDefault_None* に設定されている場合、モジュールは Item に対して適切なデフォルト値を使用することができる。

モジュールはハードウェアの状態の変化に応じて、この Capability の値を変更することができる。モジュールがこのような変更を行う場合には、*kNkMAIDEvent_CapChange* のイベントによって通知しなければならない。

9.45 kNkMAIDCapability_Firmware

ソースオブジェクト用の Capability で、デバイスのファームウェアのバージョンを保持する。

Object types Source object only
ulType *kNkMAIDCapType_String*
ulOperations *kNkMAIDCapOperation_Get*

この Capability によって、クライアントはデバイスのファームウェアのバージョンを知ることができる。

9.46 kNkMAIDCapability_CommunicationLevel1

ソースオブジェクト用の Capability で、デバイスとの間の通信方法を規定する。

Object types Source object only
ulType *kNkMAIDCapType_Enum*
ulOperations *kNkMAIDCapOperation_Get; kNkMAIDCapOperation_GetArray;
 kNkMAIDCapOperation_GetDefault; kNkMAIDCapOperation_Set*

モジュールはサポートする通信方式を決定する。ユーザーはその文字列から選択することになる。NkMAIDEnum 構造体の ulType メンバは、*kNkMAIDArrayType_String* または *kNkMAIDArrayType_PackedString* となる。例えばこのリストには"COM1"、"COM2"、"COM3"、"COM4"、

"SCSI"のような文字列が含まれる。任意に、モジュールはシステムを解析し、そのシステムがサポートしない通信方式を除外することができる。例に挙げた"COM3"、"COM4"は、システムがそれらのポートをサポートしていない場合削除されることもある。

9.47 kNkMAIDCapability_CommunicationLevel2

ソースオブジェクト用の Capability で、デバイスとの間の通信方法について詳細を規定する。

Object types Source object only
ulType *kNkMAIDCapType_Enum*
ulOperations *kNkMAIDCapOperation_Get; kNkMAIDCapOperation_GetArray;
 kNkMAIDCapOperation_GetDefault; kNkMAIDCapOperation_Set*

このキャパビリティは、通信方式について詳しく規定する文字列のリストを構成する。ユーザーはその文字列から選択することになる。NkMAIDEnum 構造体の ulType メンバは、*kNkMAIDArrayType_String* または *kNkMAIDArrayType_PackedString* となる。*kNkMAIDCapability_CommunicationLevel1* で通信方式として"COM1"が選択された場合、典型的なリストは、"Comm Speed 19,200"、"Comm Speed 38,400"、"Comm Speed 57,600"、"Comm Speed 115,200"のようになる。任意に、モジュールはシステムを解析し、そのシステムがサポートしない通信方式を除外することができる。通信方式が *kNkMAIDCapability_CommunicationLevel1* のみで十分規定され、詳しい情報が必要ない場合、NkMAIDEnum 構造体の ulElements メンバに 0 をセットしなければならない。

9.48 kNkMAIDCapability_BatteryLevel

ソースオブジェクト用の Capability で、バッテリーレベルを報告する。

Object types Source object only
ulType *kNkMAIDCapType_Integer*
ulOperations *kNkMAIDCapOperation_Get*

デバイスがバッテリー使用可能の場合、このキャパビリティをサポートしなければならない。バッテリー使用中このキャパビリティの問い合わせがあったら、モジュールは 0 から 100 までの整数を返さなければならない。この数値はバッテリー残量を百分率で表したものである。バッテリーを使用していない時(外部電源使用中等)、このキャパビリティの問い合わせに対してモジュールは-1 を返す。

9.49 kNkMAIDCapability_FreeBytes

ソースオブジェクト用の Capability で、デバイス内の記憶媒体の空きバイト数を報告する。

Object types Source object only
ulType *kNkMAIDCapType_Float*
ulOperations *kNkMAIDCapOperation_Get*

デバイスがコンパクトフラッシュ等の記憶装置を使用できる場合、このキャパビリティをサポートしなければならない。このキャパビリティは正の整数を使用するが、整数型変数の上限に制限されぬよう浮動小数点型を用いている。

9.50 kNkMAIDCapability_Freeltems

ソースオブジェクト用の Capability で、デバイスが現在の設定で内蔵記憶媒体に記録できるアイテム数を報告する。

Object types Source object only
ulType *kNkMAIDCapType_Unsigned*
ulOperations *kNkMAIDCapOperation_Get*

デバイスがコンパクトフラッシュ等の記憶装置を使用できる場合、このキャパビリティをサポートしなければならない。

9.51 kNkMAIDCapability_Remove

デバイスに記憶媒体からオブジェクトを削除するよう指示する。

Object types Source, item, data object
ulType *kNkMAIDCapType_Process*
ulOperations *kNkMAIDCapOperation_Start*

このキャパビリティがアイテムまたはデータオブジェクトに対して実行された場合、そのオブジェクトはデバイスから削除されなければならない。このキャパビリティがソースオブジェクトに対して実行された場合、全てのアイテムとその下にあるデータオブジェクトは、デバイスから削除されなければならない。

9.52 kNkMAIDCapability_FlashMode

スピードライトモードを設定する。

Object types Source, item, data object
ulType *kNkMAIDCapType_Enum*
ulOperations *kNkMAIDCapOperation_Get; kNkMAIDCapOperation_GetArray;
 kNkMAIDCapOperation_GetDefault; kNkMAIDCapOperation_Set*

NkMAIDEnum 構造体の ulType は *kNkMAIDArrayType_Unsigned*、wPhysicalBytes は 4 となる。この列挙値は eNkMAIDFlashMode の一つ以上を含む。

9.53 kNkMAIDCapability_ModuleType

このモジュールが用いられるデバイスのタイプを報告する。

Object types Module object only
ulType *kNkMAIDCapType_Unsigned*
ulOperations *kNkMAIDCapOperation_Get*

このキャパビリティの値は、ビットアサインで eNkMAIDModuleType で列挙される 1 つ以上のビットを含む。このキャパビリティは、クライアントがこのモジュールを使うかどうか、あるいはユーザーインターフェースを表示するかどうか決定するのに役立つ。

9.54 kNkMAIDCapability_AcquireStreamStart

ストリームデータの取得を開始する。

Object types Data object only
ulType *kNkMAIDCapType_Process*
ulOperations *kNkMAIDCapOperation_Start*

モジュールはデータオブジェクトのデータ転送コールバック関数を用いて転送を開始する。このプロセスはクライアントが停止の指示を出すまで繰り返される。そのため、進行通知関数と呼んではならない。

9.55 kNkMAIDCapability_AcquireStreamStop

ストリームデータの取得を停止する。

Object types Data object only
ulType *kNkMAIDCapType_Process*
ulOperations *kNkMAIDCapOperation_Start*

ストリームデータ所得中でない場合、モジュールは *kNkMAIDResult_UnexpectedError* を返す。

9.56 kNkMAIDCapability_AcceptDiskAcquisition

モジュールが、*kNkMAIDCapability_Acquire* に対してディスク上のファイルを渡すことを許可する。

Object types Source object only
ulType *kNkMAIDCapType_Generic*
ulOperations *kNkMAIDCapOperation_Get, kNkMAIDCapOperation_Set*

クライアントがこのキャパビリティに NULL 以外のパラメータをセットしていない限り、モジュールは画像データを要求された時、ディスクファイルとしてクライアントに渡してはならない。クライアントは保存先を指定するのにこのキャパビリティを使用する。一旦この指定がなされると、モジュールはメモリ上での転送に代えてディスク上のファイルとして渡すことができる。ファイルが書き込まれクローズした後、モジュールはデータオブジェクトのデータ転送コールバック関数を呼ばなければならない。Windows において pData は、ファイルが書き込まれるフォルダのフルパスを含む NkMAIDString 構造体へのポインタでなければならない。Macintosh において pData は、ファイルが書き込まれるフォルダを示す FSSpec 構造体へのポインタでなければならない。モジュールは指定されたフォルダ内で重複しないファイル名を選定する。データ転送コールバック関数をコールしたら、モジュールは如何なる理由があってもそのファイルにアクセスしてはならない。ディスクに書き込むのに長い時間がかかる場合、モジュールはデータオブジェクトの進行通知コールバック関数を呼んでも良い。

9.57 kNkMAIDCapability_Version

モジュールが、*kNkMAIDCapability_Acquire* に対してディスク上のファイルを渡すことを許可する。

Object types Module object only
ulType *kNkMAIDCapType_Unsigned*
ulOperations *kNkMAIDCapOperation_Get*

クライアントはこのキャパビリティにより、モジュールが準拠する MAID のバージョンを知ることができる。このキャパビリティは、MAID バージョン 3.1 より導入された。したがって MAID バージョン 3.1 以前に基づいて作成されたモジュールはサポートしない (*kNkMAIDResult_NotSupported* を返す)。

このキャパビリティは、4 バイトの符号なし整数を返す。MAID バージョン番号は4つの部分に分解され、最上位の数字は最上位のバイトに、最下位の数字は最下位のバイトに割り当てられる。例えば MAID バージョン 3.1 の場合、最上位バイトは3、次のバイトは1、その次のバイトは0、最下位バイトは0となる。

9.58 kNkMAIDCapability_FilmFormat

現在のフィルムフォーマットを表示・選択するためのキャパビリティである。

Object types Source object only
ulType *kNkMAIDCapType_Enum*
ulOperations *kNkMAIDCapOperation_Get, kNkMAIDCapOperation_GetArray, kNkMAIDCapOperation_Set*

NkMAIDEnum 構造体の ulType メンバは、*kNkMAIDArrayType_String* もしくは *kNkMAIDArrayType_PackedString* となる。要素数は1もしくはそれ以上となる。

クライアントは、このキャパビリティによってフィルムフォーマットを選択することができる。例えば、フィルムフォーマットには、“35mm”、“6x6”や“6x4.5”などが含まれる。要素数が1の場合には、フィ

フィルムフォーマットが選択できないことを意味し、サポートされているフィルムフォーマット名が要素として含まれる。

9.59 kNkMAIDCapability_TotalBytes

これはソースオブジェクトのキャパビリティで、装置の内部メモリの総メモリ量を返す。

Object types Source object only
ulType *kNkMAIDCapType_Float*
ulOperations *kNkMAIDCapOperation_Get*

装置がなんらかの記憶装置をもつ場合、このキャパビリティがサポートされる。（例：コンパクトフラッシュなど）。通常、正の整数値として総メモリ量が表される。ulType が浮動小数点となっているのは、巨大なメモリ容量を持つ装置に対応するためである。

10 関数定義

10.1 MAID エントリポイント関数

```
LONG MAIDEntryPoint(
    LPNkMAIDObject pObject,           // module, source, item or data object
    ULONG          ulCommand,          // one of eNkMAIDCommand
    ULONG          ulParam,            // parameter for the command
    ULONG          ulDataType,         // one of eNkMAIDDataType
    NKPARAM        data,               // pointer or long integer
    LPNKFUNC       pfnComplete,        // function to call when complete, may be null
    NKREF          refComplete        // passed to pfnComplete
);
```

戻り値は、eNkMAIDResult のうちのひとつとなる。

10.2 MAID 完了関数

```
void MAIDCompletion(
    LPNkMAIDObject pObject,           // module, source, item or data object
    ULONG          ulCommand,          // one of eNkMAIDCommand
    ULONG          ulParam,            // parameter for the command
    ULONG          ulDataType,         // one of eNkMAIDDataType
    NKPARAM        data,               // pointer or long integer
    NKREF          refComplete,        // passed to MAIDEntryPoint
    LONG           nResult             // one of eNkMAIDResult
);
```

これはクライアントによって提供されるコールバック関数のプレースホルダである。この関数は、コマンドを完了した後に、モジュールから呼び出される。パラメータは、MAID エントリポイントに渡されたのと同じパラメータである。

10.3 MAID データ転送関数

```
LONG MAIDData(
    NKREF          refProc,             // reference set by client
    LPNkMAIDDataInfo pDataInfo,        // cast to LPNkMAIDImageInfo or LPNkMAIDSoundInfo
    LPVOID         pData
);
```

これはクライアントによって提供されるコールバック関数のプレースホルダである。この関数は、データを転送するためにモジュールから呼び出される。戻り値は、eNkMAIDResult のうちのひとつになる。

10.4 MAID イベント通知関数

```
void MAIDEvent(
    NKREF          refProc,      // reference set by client
    ULONG          ulEvent,      // one of eNkMAIDEvent
    NKPARAM        data          // pointer or long integer
);
```

これはクライアントによって提供されるコールバック関数のプレースホルダである。この関数は、イベントをクライアントに通知するために、モジュールから呼び出される。

10.5 MAID 進行通知関数

```
void MAIDProgress(
    ULONG          ulCommand,    // one of eNkMAIDCommand
    ULONG          ulParam,      // parameter for the command
    NKREF          refProc,      // reference set by client
    ULONG          ulDone,       // the numerator
    ULONG          ulTotal       // the denominator
);
```

これはクライアントによって提供されるコールバック関数のプレースホルダである。この関数は、非同期コマンドの進行についてクライアントに通知するために、モジュールから呼び出される。

モジュールは、コマンドの開始時に、この関数の ulDone パラメータを 0 にセットして呼び出す。コマンドが完了したときに、モジュールは ulDone パラメータを ulTotal パラメータに等しくして、この関数を呼び出す。モジュールは、ulDone パラメータを 0 およびちょうど ulTotal にセットして、この関数を一度ずつ呼び出す。

コマンドの進行状態が計測できない場合、モジュールは ulDone を 1 に ulTotal を 0 にセットして、コマンドの開始の段階でこの関数を呼び出す。コマンドが完了した時、モジュールは ulDone と ulTotal を共に 0 にセットしてこの関数を呼び出す。

10.6 MAID ユーザインターフェイス要求関数

```
ULONG MAIDUIRequest(
    NKREF          refProc,      // reference set by client
    LPNkMAIDUIRequestInfo pUIRequest // information about the UI request
);
```

これはクライアントによって提供されるコールバック関数のプレースホルダである。この関数は、ユーザに対する通知もしくはユーザへの問いかけのために、モジュールによって呼び出される。pObject パラメータは、ユーザに対して表示するための Capability をユーザインターフェイス要求が含んでいる場合に使用される。表示するための Capability がない場合は、このパラメータは NULL となる。pUIRequest パラメータは、NkMAIDUIRequestInfo 構造体へのポインタである。この構造体は、メッセージ、ボタンについての情報を含んでいる。オプションとして、表示するための Capability についての情報も含む(?)。戻り値は、eNkMAIDUIRequestResult のうちのひとつとなる。

11 変更履歴

11.1 Version 3.0 Revision 2 からの変更

「使用法」の章を追加。

「変更履歴」の章を追加。

「Capability」の章に加筆。

「関数定義」の章で、関数の説明を補完。

NkMAIDPoint、NkMAIDRect 構造体を追加。

eNkMAIDFilter 列挙値を追加。

eNkMAIDResult において、kNkMAIDResult_Aborted、kNkMAIDResult_NoMedia を追加、kNkMAIDResult_NotLocked、kNkMAIDResult_Locked を削除。

eNkMAIDCommand において、kNkMAIDCommand_Abort を追加、kNkMAIDCommand_OpenModule、kNkMAIDCommand_GetChildCount、kNkMAIDCommand_GetChildIDs を削除、kNkMAIDCommand_OpenChild を kNkMAIDCommand_Open に、kNkMAIDCommand_ClearToMark を kNkMAIDCommand_AbortToMark に変更。

eNkMAIDCapability において、kNkMAIDCapability_Children、kNkMAIDCapability_Start、kNkMAIDCapability_Prescan、kNkMAIDCapability_AutoFocus、kNkMAIDCapability_AutoFocusPt、kNkMAIDCapability_Preview、kNkMAIDCapability_Transparency、kNkMAIDCapability_Threshold を追加、kNkMAIDCapability_Abort、kNkMAIDCapability_DataObj を削除、kNkMAIDCapability_DataAvailable を kNkMAIDCapability_DataTypes に、kNkMAIDCapability_Date を kNkMAIDCapability_DateTime に、kNkMAIDCapability_AcquireMode を kNkMAIDCapability_Mode に、kNkMAIDCapability_LightSource を kNkMAIDCapability_Filter に変更。

eNkMAIDDataType に kNkMAIDDataType_PointPtr、kNkMAIDDataType_RectPtr を追加。

eNkMAIDArrayType に kNkMAIDArrayType_Point、kNkMAIDArrayType_Rect を追加。

eNkMAIDCapType に kNkMAIDCapType_Point、kNkMAIDCapType_Rect を追加。

kNkMAIDCommand_Mark と kNkMAIDCommand_AbortToMark のコメントを変更。

NkMAIDObject 構造体の ulObjectType メンバを ulType に変更。

11.2 Version 3.0 Revision 3 からの変更

「関数定義」、「構造体と型」の章で、データ転送関数、イベント通知関数、進行通知関数の引数にリファレンスを追加した。

ULONG、NKPARAM、LPVOID、NKREF、LPMAIDEntryPointProc、LPMAIDCompletionProc、LPMAIDDataProc、LPMAIDEventProc の定義を追加。

eNkMAIDDataType において、kNkMAIDDataType_CharPtr、kNkMAIDDataType_ShortPtr、kNkMAIDDataType_BytePtr、kNkMAIDDataType_WordPtr を削除、kNkMAIDDataType_LongPtr を kNkMAIDDataType_IntegerPtr に、kNkMAIDDataType_DwordPtr を kNkMAIDDataType_UnsignedPtr に変更。

eNkMAIDCapType において、kNkMAIDCapType_Char、kNkMAIDCapType_Short、kNkMAIDCapType_Byte、kNkMAIDCapType_Word を削除、kNkMAIDCapType_Long を kNkMAIDCapType_Integer に、kNkMAIDCapType_Dword を kNkMAIDCapType_Unsigned に変更。

11.3 Version 3.0 Revision 4 からの変更

章節番号を付けた。

eNkMAIDCapability に kNkMAIDCapability_Pixels 、 kNkMAIDCapability_Stereo 、 kNkMAIDCapability_Samples を追加し、kNkMAIDCapability_Size を kNkMAIDCapability_StoredBytes に変更。

eNkMAIDEvent で kNkMAIDEvent_Add を kNkMAIDEvent_AddChild に、kNkMAIDEvent_Remove を kNkMAIDEvent_RemoveChild に変更。

eNkMAIDDataType に kNkMAIDDataType_SizePtr を追加。

eNkMAIDArrayType に kNkMAIDArrayType_Size を追加。

eNkMAIDCapType に kNkMAIDCapType_Size を追加。

NkMAIDArray 構造体に ulDimSize3 メンバを追加し、説明を改正。

NkMAIDUIEventInfo、NkMAIDDataInfo、NkMAIDImageInfo、NkMAIDSoundInfo 構造体を追加。

11.4 Version 3.0 Revision 5 からの変更

「使用法」の章において、「オブジェクトの追加と削除」を削除し、「イベントの通知」、「ユーザーインターフェイスの要求」を追加した。

eNkMAIDEvent に kNkMAIDEvent_NewMedia、kNkMAIDEvent_MediaRemoved を追加、kNkMAIDEvent_UserInterface を削除。

eNkMAIDUIEventType を eNkMAIDUIRequestType に変更。

eNkMAIDEventResult を eNkMAIDUIRequestResult に変更。

eNkMAIDCommand に kNkMAIDCommand_EnumChildren を追加。

eNkMAIDCapability に kNkMAIDCapability_UIRequestProc、kNkMAIDCapability_MediaPresent.を追加。

「コマンド」、「使用法」の章において、kNkMAIDCommand_Open の説明を変更した。

ユーザーインターフェイス要求関数のポインタ LPMAIDUIRequestProc を追加。

「構造体と型」の章にユーザーインターフェイス要求構造体を追加。

「コマンド」の章に kNkMAIDCommand_EnumChildren の説明を追加。

イベント通知コールバック関数の戻り値を ULONG から void に変更。

11.5 Version 3.0 Revision 6 からの変更

4.1 eNkMAIDResult に kNkMAIDResult_ZombieObject と kNkMAIDResult_OrphanedChildren を追加。

4.13 eNkMAIDCommand、8 章に kNkMAIDCommand_GetParent を追加。

4.14 eNkMAIDCapability、9 章に kNkMAIDCapability_Alive を追加。

4.9 eNkMAIDEvent、7 章に kNkMAIDEvent_WarmedUp、kNkMAIDEvent_CapChange、

kNkMAIDEvent_OrphanedChildren を追加し、kNkMAIDEvent_NewMedia、kNkMAIDEvent_MediaRemoved を削除。

4.10 eNkMAIDUIRequestType に kNkMAIDUIRequestType_CustomOkCancel を追加。

5.21 オブジェクト構造体にリファレンス (NKREF) の説明を追加。

5.22 ユーザーインターフェイス要求構造体に表示文字列の説明を追加。

8.3 kNkMAIDCommand_Close に説明を追加。

9.15 kNkMAIDCapability_Eject に説明を追加。

6 章 戻り値、7 章 イベントを追加。

5.16 NkMAIDRect を (x1, y1, x2, y2) から (x, y, w, h) に変更。

5.24 NkMAIDImageInfo 構造体の wBits メンバを 1 つの値から 4 要素の配列に変更し、用法説明を追加。

11.6 Version 3.0 Revision 7 からの変更

変更履歴の順番を、最新のものが最後になるよう並べ替えた。

5.6 MAID エントリポイント関数の引数の型を LPNKFUNC から LPMAIDCompletionProc に変更。

5.11 および 10.6 MAID ユーザーインターフェイス要求関数に pObject パラメータを追加。

5.22 NkMAIDUIRequestInfo 構造体に data パラメータを追加。

4.10 eNkMAIDUIRequestType から kNkMAIDUIRequestType_CustomOkCancel を削除。

4.15 色空間の項を追加。

4.14 kNkMAIDCapability_Color から NkMAIDCapability_ColorSpace に変更。

9.34 kNkMAIDCapability_Color から NkMAIDCapability_ColorSpace に変更。

9.6 kNkMAIDCapability_Alive から kNkMAIDCapability_IsAlive に変更。

5.24 NkMAIDImageInfo 構造体のメンバを ulColorSpace に変更。

11.7 Version 3.0 Revision 8 からの変更

変更履歴の順番を、最新のものが最後になるよう並べ替えた。

9.12 で参照する列挙値を eNkMAIDDataType から eNkMAIDDataObjType に変更。

5.19 NkMAIDRange 構造体に ulValueIndex と ulDefaultIndex を追加し、nSteps を ulSteps に変更。

5.22 NkMAIDUIRequestInfo 構造体に pObject を追加し、10.6 MAID ユーザーインターフェイス要求関数から pObject を削除した。

5.18 NkMAIDArray 構造体から ulValue と ulDefault を削除した。この構造体は、もはや列挙には使われない。

5.26 NkMAIDEnum 構造体を追加。

4.3 eNkMAIDDataType に kNkMAIDDataType_EnumPtr を追加。

- 4.5 eNkMAIDCapType に kNkMAIDCapType_Enum を追加。
- 8.7 CapSet、8.9 CapGetDefault、8.10 CapGetArray に kNkMAIDCapType_Enum を追加。
- 9.7 kNkMAIDCapability_Children を kNkMAIDCapType_Enum タイプに変更。
- 9.18 kNkMAIDCapability_Mode を kNkMAIDCapType_Enum タイプに変更。
- 9.22 kNkMAIDCapability_SampleRate を kNkMAIDCapType_Enum タイプに変更。
- 9.25 kNkMAIDCapability_Filter を kNkMAIDCapType_Enum タイプに変更。
- 9.29 kNkMAIDCapability_Focus を kNkMAIDCapType_Enum タイプに変更。
- 9.31 kNkMAIDCapability_Resolution を kNkMAIDCapType_Enum タイプに変更。
- 9.34 kNkMAIDCapability_ColorSpace を kNkMAIDCapType_Enum タイプに変更。

11.8 Version 3.0 Revision 9 からの変更

- 5.11 MAIDUIRequestProc から LPNkMAIDObject を削除。
- 9 章にグループキャパビリティの説明を追加。
- 9.37 kNkMAIDCapability_Lut を複数の色空間に対応させた。
- 5.22 ユーザインターフェイス要求構造体で配列構造体の使用法を明確にした。
- 3.4 項で kNkMAIDCommand_CapGet と kNkMAIDCommand_CapGetArray を説明するサンプルを統合した。
- 5.19 範囲構造体で 0 基点のインデックスを使うよう明記した。
- 5.26 列挙構造体で 0 基点のインデックスを使うよう明記した。
- 4.14 eNkMAIDCapability が 1 から始まるように変更。
- 7.5 kNkMAIDEvent_CapChange の使用法を明確にした。
- 9.3 kNkMAIDCapability_EventProc において、Object types に "data object" を追加。
- 4.7 eNkMAIDCapVisibility から kNkMAIDCapVisibility_Normal を削除。
- 4.1 eNkMAIDResult に kNkMAIDResult_NoDataProc 、 kNkMAIDResult_OutOfMemory 、 kNkMAIDResult_UnexpectedError、 kNkMAIDResult_HardwareError を追加。

Added descriptions for new result codes in sections 6.12, 6.13, 6.14, and 6.15.

Added "thumbnail" to the supported object types for the following capabilities in section 9: kNkMAIDCapability_Filter, kNkMAIDCapability_Prescan, kNkMAIDCapability_AutoFocus, kNkMAIDCapability_AutoFocusPt, kNkMAIDCapability_Focus, kNkMAIDCapability_Coords, kNkMAIDCapability_Resolution, kNkMAIDCapability_Preview, kNkMAIDCapability_Negative, kNkMAIDCapability_ColorSpace, kNkMAIDCapability_Bits, kNkMAIDCapability_Planar, kNkMAIDCapability_Lut, kNkMAIDCapability_Transparency, kNkMAIDCapability_Threshold, and kNkMAIDCapability_Pixels.

11.9 Version 3.0 Revision 10 からの変更

- 4.1 eNkMAIDResult に kNkMAIDResult_MissingComponent を追加。
- 6.16 kNkMAIDResult_MissingComponent を追加。
- 4.14 eNkMAIDCapability に kNkMAIDCapability_ForceScan、 kNkMAIDCapability_ForcePrescan、

kNkMAIDCapability_ForceAutoFocus を追加。

9.41 kNkMAIDCapability_ForceScan を追加。

9.42 kNkMAIDCapability_ForcePrescan を追加。

9.43 kNkMAIDCapability_ForceAutoFocus を追加。

5.22 NkMAIDUIRequestInfo 構造体に lpDetail メンバを追加。

4.3 eNkMAIDDataType に kNkMAIDDataType_GenericPtr を追加。

4.5 eNkMAIDCapType に kNkMAIDCapType_Generic を追加。

8.7 kNkMAIDCommand_CapSet に kNkMAIDCapType_Generic を追加。

8.8 kNkMAIDCommand_CapGet に kNkMAIDCapType_Generic を追加。

8.9 kNkMAIDCommand_CapGetDefault に kNkMAIDCapType_Generic を追加。

11.10 Version 3.0 Revision 11 からの変更

8 章にコールバック関数内から非同期コマンドを出す場合の制限事項を追加。

4.14 項に kNkMAIDCapability_NegativeDefault を追加。

4.3 eNkMAIDDataType から kNkMAIDDataType_BoolDefaultPtr を削除。

4.16 eNkMAIDBooleanDefault を追加。

4.3 eNkMAIDDataType に kNkMAIDDataType_BoolDefaultPtr を追加。

4.5 eNkMAIDCapType に kNkMAIDCapType_BoolDefault を追加。

9.44 kNkMAIDCapability_NegativeDefault を追加。

11.11 Version 3.0 Revision 12 からの変更

4.3 eNkMAIDDataType から kNkMAIDDataType_BoolDefaultPtr を削除。

10.5 項にコマンドの進行状態が計測できない場合の説明を追加。

11.12 Version 3.0 Revision 13 からの変更

4.17 eNkMAIDModuleTypes を追加。

4.18 eNkMAIDFileDataTypes を追加。

5.24 NkMAIDImageInfo および 5.25 NkMAIDSoundInfo 構造体に fRemoveObject を追加。

5.27 NkMAIDFileInfo 構造体を追加。

8.16 kNkMAIDCommand_ResetToDefault コマンドを追加。

9.45 kNkMAIDCapability_CommunicationLevel1 を追加。

9.46 kNkMAIDCapability_CommunicationLevel2 を追加。

9.47 kNkMAIDCapability_BatteryLevel を追加。

9.48 kNkMAIDCapability_FreeBytes を追加。

9.49 kNkMAIDCapability_FreeItems を追加。

9.50 kNkMAIDCapability_Remove を追加。

9.51 kNkMAIDCapability_FlashMode を追加。

9.52 kNkMAIDCapability_ModuleType を追加。

9.53 kNkMAIDCapability_AcquireStreamStart を追加。

9.54 kNkMAIDCapability_AcquireStreamStop を追加。

9.55 kNkMAIDCapability_AcceptDiskAcquisition を追加。

9.56 kNkMAIDCapability_Version を追加。

上記キャパビリティを 4.14 Capabilities 中の eNkMAIDCapability に追加。

3.1 項において、MAID モジュールファイルの置き場所を変更した。

11.13 Version 3.1 Revision 1 からの変更

9.44 kNkMAIDCapability_NegativeDefault のタイプを BooleanDefault から Unsigned に変更。

4.16 eNkMAIDBooleanDefault から kNkMAIDBooleanDefault_None を削除。

11.14 Version 3.1 Revision 2 からの変更

4.14 eNkMAIDCapability に kNkMAIDCapability_Firmware を追加。

セクション番号 9.45 から 9.56 を 9.46 から 9.57 に変更。

9.45 項へ kNkMAIDCapability_Firmware を挿入。

11.15 Version 3.1 Revision 3 からの変更

4.1 eNkMAIDResult に kNkMAIDResult_VendorBase を追加。

11.16 Version 3.1 Revision 4 からの変更

4.18 eNkMAIDFileDataTypes に kNkMAIDFileDataTypes_NIF を追加。

4.9 eNkMAIDEvent に kNkMAIDEvent_CapChangeValueOnly を追加。

7.7 kNkMAIDEvent_CapChangeValueOnly を追加。

11.17 Version 3.1 Revision 5 からの変更

9.56 kNkMAIDCapability_AcceptDiskAcquisition において、クライアントはファイル名でなく書き込み先フォルダを指定するように変更。

4.2 eNkMAIDDataObjType に kNkMAIDDataObjType_File を追加。

9.12 kNkMAIDCapability_DataTypes で kNkMAIDDataObjType_File の使用を禁止した。

5.23 項へ kNkMAIDDataObjType_File に関する説明を追加。

11.18 Version 3.1 Revision 6 からの変更

4.15 eNkMAIDColorSpace に kNkMAIDColorSpace_AppleRGB、kNkMAIDColorSpace_ColorMatchRGB、kNkMAIDColorSpace_NTSCRGB、kNkMAIDColorSpace_BruceRGB、kNkMAIDColorSpace_AdobeRGB、kNkMAIDColorSpace_CIERGB、kNkMAIDColorSpace_AdobeWideRGB、kNkMAIDColorSpace_NikonWideRGBg18、kNkMAIDColorSpace_NikonWideRGBg22 を追加。

11.19 Version 3.1 Revision 7 からの変更

9.56 kNkMAIDCapability_AcceptDiskAcquisition のキャパビリティタイプを kNkMAIDCapType_Generic に変更。

4.7 eNkMAIDCapVisibility に kNkMAIDCapVisibility_Valid を追加。

11.20 Version 3.1 Revision 8 からの変更

4.15 eNkMAIDColorSpace において、kNkMAIDColorSpace_NikonWideRGBg18 を kNkMAIDColorSpace_AppleRGB_Compensated に、kNkMAIDColorSpace_NikonWideRGBg22 を kNkMAIDColorSpace_AdobeWideRGB_Compensated にそれぞれ変更。

11.21 Version 3.1 Revision 9 からの変更

4.7 eNkMAIDCapVisibility において、kNkMAIDCapVisibility_Valid を kNkMAIDCapVisibility_Invalid に、その値を 0x0020 に変更。

11.22 Version 3.1 Revision 10 からの変更

4.15 eNkMAIDColorSpace から kNkMAIDColorSpace_AdobeWideRGB_Compensated を削除。

11.23 Version 3.1 Revision 11 からの変更

9.52 kNkMAIDCapability_FlashMode を文字列列挙から整数値列挙に変更。

4.19 eNkMAIDFlashMode を追加。

11.24 Version 3.1 Revision 12 からの変更

4.19 eNkMAIDFlashMode に kNkMAIDFlashMode_SlowSyncRearCurtain を追加。

11.25 Version 3.1 Revision 13 からの変更

4.15 eNkMAIDColorSpace に kNkMAIDColorSpace_VendorBase を追加。

11.26 Version 3.1 Revision 14 からの変更

4.18 eNkMAIDFileDataTypes に kNkMAIDFileType_QuickTime を追加。

11.27 Version 3.1 Revision 15 からの変更

4.14 eNkMAIDCapability_enum に kNkMAIDCapability_FilmFormat を追加

9.58 に kNkMAIDCapability_FilmFormat に関する記述を追加

11.28 Version 3.1 Revision 16 からの変更

4.14 eNkMAIDCapability_enum に kNkMAIDCapability_TotalBytes を追加

9.58 に kNkMAIDCapability_TotalBytes に関する記述を追加