

日本語 hp DECwindows Motif for hp OpenVMS

DEC 入力サーバ・ライブラリ

2003 年 3 月

本書では、弊社のプラットフォーム上で、X11R6 XIM ライブラリを使用してクライアントと通信するための X11R6 入力サーバを作成する際に必要な情報を記述します。

オペレーティング・システム: 日本語 HP OpenVMS Alpha V7.3-1
ソフトウェア・バージョン: 日本語 HP DECwindows Motif for HP
OpenVMS V1.3

日本ヒューレット・パッカード株式会社

2003 年 3 月

本書の著作権は日本ヒューレット・パッカード株式会社が保有しており，本書中の解説および図，表は日本ヒューレット・パッカードの文書による許可なしに，その全体または一部を，いかなる場合にも再版あるいは複製することを禁じます。

また，本書に記載されている事項は，予告なく変更されることがありますので，あらかじめご承知おきください。万一，本書の記述に誤りがあった場合でも，日本ヒューレット・パッカードは一切その責任を負いかねます。

本書で解説するソフトウェア (対象ソフトウェア) は，所定のライセンス契約が締結された場合に限り，その使用あるいは複製が許可されます。

© 2003 日本ヒューレット・パッカード株式会社

Motif，OSF/1 および UNIX は The Open Group の商標です。

本書に記載のあるその他すべての製品名は，それぞれの会社の商標または登録商標です。

本書は，日本語 VAX DOCUMENT V 2.1を用いて作成しています。

目次

まえがき	vii
1 DEC 入力サーバ・ライブラリ	
1.1 用語	1-1
1.2 背景	1-1
1.3 アーキテクチャ	1-2
1.3.1 IMSSL の構造	1-2
1.3.2 イベント処理モデル	1-2
1.3.3 イベント・フロー制御	1-3
1.3.4 サーバの命名規則	1-4
1.3.5 IMSSL の機能	1-4
1.4 DIMs クラスのリソース	1-5
1.4.1 DIMsNlocale	1-6
1.4.2 DIMsNserverName	1-7
1.4.3 DIMsNversion	1-8
1.4.4 DIMsNextIMAttr	1-8
1.4.5 DIMsNextICAttr	1-9
1.4.6 DIMsNsupportStyles	1-10
1.4.7 DIMsNsupportEncodings	1-10
1.4.8 DIMsNtriggerKeys	1-11
1.4.9 DIMsNgetExtIMValuesCb	1-12
1.4.10 DIMsNcreateICCb	1-13
1.4.11 DIMsNdestroyICCb	1-13
1.4.12 DIMsNsetExtICValuesCb	1-13
1.4.13 DIMsNgetExtICValuesCb	1-13
1.4.14 DIMsNsetICFocusCb	1-13
1.4.15 DIMsNunsetICFocusCb	1-14
1.4.16 DIMsNresetICCb	1-14
1.4.17 DIMsNprocessEventCb	1-14
1.4.18 DIMsNsetAreaCb	1-14
1.4.19 DIMsNsetAreaNeededCb	1-14
1.4.20 DIMsNsetSpotLocationCb	1-14
1.4.21 DIMsNsetColormapCb	1-15
1.4.22 DIMsNsetStdColormapCb	1-15
1.4.23 DIMsNsetForegroundCb	1-15
1.4.24 DIMsNsetBackgroundCb	1-15
1.4.25 DIMsNsetBgPixmapCb	1-15
1.4.26 DIMsNsetFontSetCb	1-16
1.4.27 DIMsNsetLineSpaceCb	1-16
1.4.28 DIMsNsetCursorCb	1-16
1.4.29 DIMsNgetFilterEventsCb	1-16
1.4.30 DIMsNgetAreaCb	1-16
1.4.31 DIMsNgetAreaNeededCb	1-17

1.4.32	DIMsNgetSpotLocationCb	1-17
1.4.33	DIMsNgetColormapCb	1-17
1.4.34	DIMsNgetStdColormapCb	1-17
1.4.35	DIMsNgetForegroundCb	1-17
1.4.36	DIMsNgetBackgroundCb	1-18
1.4.37	DIMsNgetBgPixmapCb	1-18
1.4.38	DIMsNgetFontSetCb	1-18
1.4.39	DIMsNgetLineSpaceCb	1-18
1.4.40	DIMsNgetCursorCb	1-18
1.4.41	DIMsNtriggerNotifyCb	1-19
1.4.42	DIMsNtransport	1-19
1.4.43	DIMsNsetStringConversionCb	1-20
1.4.44	DIMsNsetStringConversionCallbackCb	1-20
1.4.45	DIMsNsetResetStateCb	1-20
1.4.46	DIMsNgetResetStateCb	1-20
1.4.47	DIMsNsetHotKeyCb	1-20
1.4.48	DIMsNgetHotKeyCb	1-21
1.4.49	DIMsNsetHotKeyStateCb	1-21
1.4.50	DIMsNgetHotKeyStateCb	1-21
1.4.51	DIMsNsetPreeditStateCb	1-21
1.4.52	DIMsNgetPreeditStateCb	1-21
1.4.53	DIMsNsetPreeditStateCallbackCb	1-22
1.4.54	DIMsNclientdata	1-22
1.5	パブリック・ルーチン	1-22
	DIMsClassCtxCreate	1-23
	DIMsGetClassValues	1-24
	DIMsClassCtxGetValues	1-25
	DIMsSLInitialize	1-26
	DIMsProtoInit	1-27
	DIMsPreEditStart	1-28
	DIMsPreEditDone	1-29
	DIMsPreEditStateNotify	1-30
	DIMsIsIState	1-31
	DIMsPreEditDraw	1-32
	DIMsPreEditCaret	1-34
	DIMsStatusStart	1-35
	DIMsStatusDone	1-36
	DIMsStatusDraw	1-37
	DIMsGeometryNegotiation	1-39
	DIMsSetEventsForward	1-40
	DIMsIsEventsForwarded	1-41
	DIMsCommit	1-42
	DIMsEncoding	1-43
	DIMslocale	1-44
	DIMsInputStyle	1-45
	DIMsClientWindow	1-46
	DIMsFocusWindow	1-47
	DIMsUserData	1-48
	DIMsSetTriggerKeys	1-49
	getExtIMValuesCb	1-50

	createICCb	1-51
	destroyICCb	1-53
	setExtICValuesCb	1-54
	getExtICValuesCb	1-56
	setICFocusCb	1-58
	unsetICFocusCb	1-59
	resetICCb	1-60
	processEventCb	1-61
	setAreaCb	1-63
	setAreaNeededCb	1-65
	setSpotLocationCb	1-67
	setColormapCb	1-69
	setStdColormapCb	1-71
	setForegroundCb	1-73
	setBackgroundCb	1-75
	setBgPixmapCb	1-77
	setFontSetCb	1-79
	setLineSpaceCb	1-81
	setCursorCb	1-83
	getFilterEventsCb	1-85
	getAreaCb	1-87
	getAreaNeededCb	1-88
	getSpotLocationCb	1-89
	getColormapCb	1-90
	getStdColormapCb	1-91
	getForegroundCb	1-92
	getBackgroundCb	1-93
	getBgPixmapCb	1-94
	setFontSetCb	1-95
	getLineSpaceCb	1-96
	getCursorCb	1-97
	triggerNotifyCb	1-98
	setStringConversionCb	1-99
	setStringConversionCallbackCb	1-100
	setResetStateCb	1-101
	getResetStateCb	1-102
	setHotKeyCb	1-103
	getHotKeyCb	1-104
	setHotKeyStateCb	1-105
	getHotKeyStateCb	1-106
	setPreeditStateCb	1-107
	getPreeditStateCb	1-108
	setPreeditStateCallbackCb	1-109
1.6	サンプル・ファイル	1-110
1.6.1	ビルド・プロシージャ	1-110
1.6.2	ソース・プログラム	1-110



1-1	IMSSL の構造	1-3
-----	-----------------	-----

本書の目的

このドキュメントでは、弊社のプラットフォーム上で、X11R6 XIM ライブラリを使用してクライアントと通信するための X11R6 入力サーバを作成する際に必要な情報を記述します。

本書の対象読者

本書は、日本語 HP DECwindows Motif for HP OpenVMS を使用して日本語アプリケーションの開発を行うプログラマを対象にしています。

本書で使用する表記法

本書では、以下の表記法を使用します。

Ctrl/x

Ctrl/x は、Ctrl キーを押しながら、別のキーまたはポインティング装置のボタンを押すことを示します。

Return

例中の四角で囲まれたキー名は、キーボード上の対応するキーを押すことを示します（文中ではキー名は四角で囲まれていません）。

...

例中の水平の反復記号は、以下のいずれかを示します。

- 文中で省略可能な追加の引数が省略されていること
- 前の項目を何度か繰り返すことができること
- 追加パラメータ、値、または他の情報を入力できること

.
.
.

垂直の反復記号は、コード例やコマンド形式で項目が省略されていることを示します。

()

形式の説明で、括弧は、複数のオプションを選択するとき、選択したオプション全体を括弧で囲まなければならないことを示します。

[]

形式の説明で、大括弧で囲まれた項目は、省略可能な項目を示します。何も選択しないか、1つの項目を選択するか、またはすべての項目を選択します（ただし、ファイル指定におけるディレクトリ名、および代入文における部分文字列指定では、大括弧を省略することはできません）。

太字	<p>太字のテキストは，新しい用語の紹介，引数名，属性または理由を示します。</p> <p>太字のテキストは，オンライン版ドキュメントでのユーザ入力を示すのにも使用します。</p>
数字	<p>特に指定しない限り，文中のすべての数字は 10 進数で示します。他の基数 — 2 進数，8 進数，または 16 進数 — の場合は明確に示します。</p>

DEC 入力サーバ・ライブラリ

OpenVMS オペレーティング・システムでは、通信メカニズムに関する詳しい知識がなくても入力サーバを作成できるようなアプリケーション・プログラミング・インタフェース (API) を提供しています。

1.1 用語

名前	説明
IMS (Input Method Server)	入力サーバ。XIM ライブラリからの入力キー・イベントおよびプロトコルを受信および処理し、前編集と文字列の確定を行うプロセスです。
IMSSL (Input Method Server Service Layer)	IMS の開発に必要なすべての API が含まれている層です。
DECXim プロトコル	弊社の入力メソッド・プロトコルです。
IMS プログラマ	IMS の開発者です。
DIMs ウィジェット (Digital Input Method Server widget)	クライアントが XCreateIC() を実行するときに作成されるウィジェットです。DIMs ウィジェットは、クライアントの XIC オブジェクトを処理する基本単位です。
DIM クラス	DIMs ウィジェットの特性と動作を決定するオブジェクトです。IMS プログラマはこの DIMs クラスにリソースを設定します。

1.2 背景

日本語、中国語、韓国語といったアジア言語環境では文字入力メソッドが複雑なため、文字入力メソッド部分は通常、アプリケーションから独立したサーバ・プロセスとして構成されます。ウィンドウ環境においては、入力サーバ (IMS) がその役割を受け持ちます。

OpenVMS では、入力サーバとクライアント間の通信は R6 XIM プロトコルで行われます。クライアント側では、X ライブラリ (Xlib)、X ツールキット・ライブラリ (Xt) および Motif ライブラリ (Xm) を使用することによって R6 XIM プロトコルを扱うことができます。しかし、入力サーバ側にはそのような上位レベルのプログラミング・インタフェースは提供されていませんでした。

1.3 アーキテクチャ

1.3.1 IMSSL の構造

クライアントが XOpenIM() を呼び出すと、IM ライブラリと IMSSL との間の接続が行われます。IMSSL は接続を行い、IM ライブラリ・コンテキストを作成し、クライアントに対する XIMID を生成します。クライアントは、XCreateIC() を使用して各入力テキスト・フィールドに対応する入力コンテキスト (IC) を作成します。

クライアントが複数の入力フィールドを持っている場合、IMSSL は個々の入力フィールドを管理するためにフィールドと同数の IC を作成し、有効な XICID を生成します。さらに、個々の IC に対応する DIMs ウィジェットが作成されます。この DIMs ウィジェットは、IMS プログラマによってコールバック・ルーチンなどのリソースで設定される DIMs クラスの動作を継承します。このように、IMSSL の IM および IC のコンテキストが IMS プログラマに対して透過的になっています。このため、IMS プログラマはそれぞれの DIMs ウィジェットを扱い、IMS のスタートアップ時に DIMs クラスのリソースを設定するだけでよいのです。

図 1-1 に、IMSSL の構造と各構成要素の関係を示します。

1.3.2 イベント処理モデル

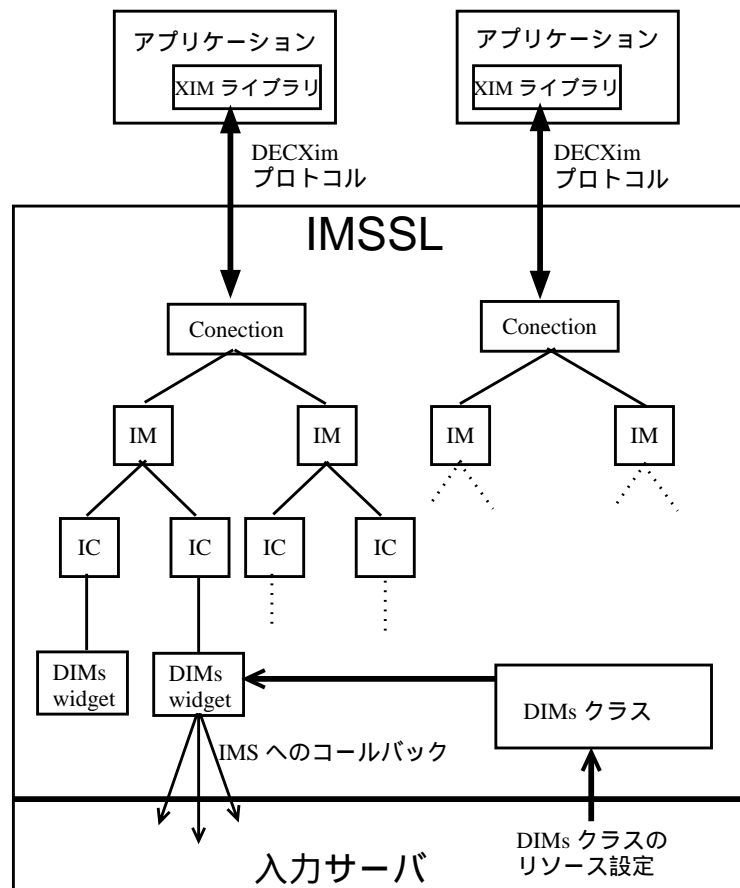
イベント処理モデルには、FrontEnd メソッドおよび BackEnd メソッドの 2 種類があります。

FrontEnd メソッドでは、クライアント・ウィンドウの入力イベントは、X サーバによって IMS ライブラリと XIM ライブラリの両方に直接渡されます。FrontEnd メソッドは、対話式の前編集において優れた性能を提供します。ただし FrontEnd モデルでは、IMS が処理するキー・イベントとクライアントが処理する他のイベントとの間で、同期化の問題が発生します。このため、FrontEnd メソッドではキー・イベントの損失または重複が発生することがあります。

BackEnd メソッドでは、クライアント・ウィンドウの入力イベントは、XIM ライブラリに渡したあと IMS に渡されます。各イベントは渡された順でシリアルに処理されるため、XIM ライブラリと IMS ライブラリとの間で同期化の問題は発生しません。BackEnd メソッドを使用すると、IM ライブラリは、すべての KeyPress と KeyRelease、およびイベント・フロー制御で必要とするその他のイベントを IMS に転送し、IMS と同期をとります。

R6 XIM および R5 DECxim プロトコルは、BackEnd モデルのみをサポートしています。

図 1-1 IMSSL の構造



1.3.3 イベント・フロー制御

R6 XIM および R5 DECXim プロトコルは、XIM ライブラリと IMS 間の通信のために静的イベント・フローおよび動的イベント・フローの 2 種類のイベント・フロー・モデルをサポートしています。

静的イベント・フローでは、入力キー・イベントがクライアントから IMS に常に送信されます。

動的イベント・フローでは、処理が必要なキー・イベントだけがクライアントから IMS に送信されます。たとえば、ASCII 文字と漢字を組み合わせで入力する場合、ASCII 文字は IMS で処理する必要はないのでキー・イベントを IMS に送信する必要はありませんが、漢字を形成するために必要なキー・イベントは IMS に送信しなければなりません。

動的イベント・フロー・モデルをサポートするには、IMS プログラマはトリガ・オン・キー・リストをクライアントに送信する必要があります (詳細は、第 1.4.8 項を参照)。クライアントは、トリガ・オン・キー・リストに含まれるキーを受信して初めて、キー・イベントを IMS に送信します。IMS プログラマがトリガ・オフ・キー・リストを提供している場合、トリガ・オフ・キー・リストに含まれるキーを受信すると、クライアントは IMS へのイベントの送信を停止します。IMS がトリガ・オフ・キー・リストを提供していない場合、IMS はトリガ・オフ・キーをチェックするために API を呼び出し、イベント・フローを停止します。詳細は、第 1.5 節の DIMsSetEventsForward の項を参照してください。

1.3.4 サーバの命名規則

IMS は、クライアントからの接続のために 1 つまたは複数のアトムを登録します。クライアントは、ロケール情報と環境変数 XMODIFIERS から特定の IMS と接続しようとします。たとえば、XMODIFIERS が @im=wnn でロケールが ja_JP.SJIS の場合、クライアントは _XIM_ja_JP.SJIS@wnn というアトムを検索します。このアトムが検出できない場合、クライアントは _XIM_ja_JP@wnn というアトムを探します。このアトムも検出できない場合は、接続が失敗したとみなされます。XMODIFIERS 環境変数が設定されていない場合、クライアントは _XIM_ja_JP.SJIS@DEC および _XIM_ja_JP@DEC を検索します。これらは弊社が提供する IMS のサーバ名です。サーバ名の設定方法についての詳細は、第 1.4.2 項を参照してください。

1.3.5 IMSSL の機能

OpenVMS の IMSSL は、IMS の開発者に以下のような機能を提供します。

- ネットワーク透過性
R6 XIM および R5 DECxim プロトコルは X プロトコルに基づいており、下位層のネットワーク・プロトコルから独立しています。このため、DEC 入力サーバ・ライブラリによって開発された IMS は、TCP/IP、DECnet など、どのようなネットワーク上でも動作します。
- プロトコルと転送メカニズムの透過性
IMS プログラマには、R6 XIM および R5 DECxim プロトコルのフォーマットや、転送メカニズムの動作方法についての知識は不要です。
- 信頼性
R6 XIM および R5 DECxim プロトコルは、より信頼性の高い BackEnd メソッドをサポートしています。詳細については、第 1.3.2 項を参照してください。
- イベント・フロー制御
R6 XIM および R5 DECxim プロトコルは、静的フロー・モデルおよび動的フロー・モデルの両方をサポートしています。詳細は、第 1.3.3 項を参照してください。

- エラー検出
IMSSL は、クライアントで発生したエラーをチェックし、クライアントにエラーを返します。たとえば、クライアントが無効な XIC 値を IMS に送信した場合や、XNClientWindow のような必須の属性を省略して XCreateIC() を呼び出した場合などです。
- データ変換
IMSSL は、すべてのデータ型変換を実行します。また、IMS の要求に応じてプロトコル・フォーマットからデータ構造への、あるいはその逆の変換を行います。
- 拡張 XIC 属性および XIM 属性
R6 XIM および R5 DECxim は、拡張 XIC 属性と XIM 属性のメカニズムをサポートしています。詳細については、第 1.4.4 項および第 1.4.5 項を参照してください。
- 簡単なプログラム・インタフェース
IMSSL は、便利な関数とコールバック・ルーチンを IMS プログラマに提供します。IMS プログラマは、IM コンテキストや IC コンテキストを処理および管理する代わりに、DIMs ウィジェットを処理するだけでよいのです。

1.4 DIMs クラスのリソース

各 DIMs ウィジェットは、ウィジェット自身の動作を決定する一連のデータを持っています。これを DIMs クラスと呼びます。この節では、DIMs クラスの各リソースの意味とその使用方法を説明します。

次の表に、プログラマがデータを指定するために使用する DIMs クラスのリソースを示します。各リソースは、DIMsSetTriggerKeys() を使用して設定および変更を行う DIMsNtriggerKeys を除いて、DIMsClassCtxCreate() 使用時にのみ設定できます。DIMsClassCtxCreate() および DIMsSetTriggerKeys() については、第 1.5 節を参照してください。

次の表の備考の欄は、DIMs クラスの作成時にリソースの設定が必須 (M) であるか、あるいは任意 (O) であるかを示しています。

名前	省略時の値	型	備考
DIMsNlocale	NULL	DIMsLocaleInfo	M
DIMsNserverName	NULL	DIMsStringList	M
DIMsNversion	動的	DIMsVersion	O
DIMsNextIMAttr	NULL	DIMsExtAttr	O
DIMsNextICAttr	NULL	DIMsExtAttr	O
DIMsNsupportStyles	NULL	DIMsSupportStyles	M
DIMsNsupportEncodings	Compound Text	DIMsStringList	O
DIMsNtriggerKeys	NULL	DIMsTriggerKeys	O
DIMsNgetExtIMValuesCb	NULL	XPointer	O

名前	省略時の値	型	備考
DIMsNcreateICCb	NULL	XPointer	O
DIMsNdestroyICCb	NULL	XPointer	O
DIMsNsetExtICValuesCb	NULL	XPointer	O
DIMsNgetExtICValuesCb	NULL	XPointer	O
DIMsNsetICFocusCb	NULL	XPointer	O
DIMsNunsetICFocusCb	NULL	XPointer	O
DIMsNresetICCb	NULL	XPointer	O
DIMsNprocessEventCb	NULL	XPointer	M
DIMsNsetAreaCb	NULL	XPointer	O
DIMsNsetAreaNeededCb	NULL	XPointer	O
DIMsNsetSpotLocationCb	NULL	XPointer	O
DIMsNsetColormapCb	NULL	XPointer	O
DIMsNsetStdColormapCb	NULL	XPointer	O
DIMsNsetForegroundCb	NULL	XPointer	O
DIMsNsetBackgroundCb	NULL	XPointer	O
DIMsNsetBgPixmapCb	NULL	XPointer	O
DIMsNsetFontSetCb	NULL	XPointer	O
DIMsNsetLineSpaceCb	NULL	XPointer	O
DIMsNsetCursorCb	NULL	XPointer	O
DIMsNgetFilterEventsCb	動的	XPointer	O
DIMsNgetAreaCb	NULL	XPointer	O
DIMsNgetAreaNeededCb	NULL	XPointer	O
DIMsNgetSpotLocationCb	NULL	XPointer	O
DIMsNgetColormapCb	NULL	XPointer	O
DIMsNgetStdColormapCb	NULL	XPointer	O
DIMsNgetForegroundCb	NULL	XPointer	O
DIMsNgetBackgroundCb	NULL	XPointer	O
DIMsNgetBgPixmapCb	NULL	XPointer	O
DIMsNgetFontSetCb	NULL	XPointer	O
DIMsNgetLineSpaceCb	NULL	XPointer	O
DIMsNgetCursorCb	NULL	XPointer	O
DIMsNclientdata	NULL	XPointer	O
DIMsNtriggerNotifyCb	NULL	XPointer	O
DIMsNtransport	X/	char	O
DIMsNsetStringConversionCb	NULL	XPointer	O
DIMsNsetStringConversionCallbackCb	NULL	XPointer	O
DIMsNsetResetStateCb	NULL	XPointer	O
DIMsNgetResetStateCb	NULL	XPointer	O
DIMsNsetHotKeyCb	NULL	XPointer	O
DIMsNgetHotKeyCb	NULL	XPointer	O
DIMsNsetHotKeyStateCb	NULL	XPointer	O
DIMsNgetHotKeyStateCb	NULL	XPointer	O
DIMsNsetPreeditStateCb	NULL	XPointer	O
DIMsNgetPreeditStateCb	NULL	XPointer	O
DIMsNsetPreeditStateCallbackCb	NULL	XPointer	O

以下の項で、これらのリソースについて説明します。

1.4.1 DIMsNlocale

DIMs ウィジェットがサポートするロケールを指定します。このリソースの値はDIMsLocaleInfo型で、次の構造体へのポインタです。

```
typedef struct_DIMsLocaleInfo {
    char    *language;
    short   num_support_codesets;
    char    **support_codesets;
} DIMsLocaleInfoRec, *DIMsLocaleInfo;
```

languageフィールドには言語とテリトリを指定します。指定する値は、たとえば ja_JP のように、XNLS に準拠しているものでなければなりません。DIMs ウィジェットが複数のコードセットをサポートしている場合は、ヌル終了文字列の配列としてsupport_codesetsフィールドを指定してください。たとえば日本語 DIMs ウィジェットの場合、support_codesetsには eucJP, deckanji, sdeckanji を指定することができます。DIMs ウィジェットがすべてのコード・セットをサポートする場合は、num_support_encodingsに 0 を、support_codesetsに NULL を指定してください。クライアントが IMS に接続された後、クライアントのロケールは DIMsLocale(w) を使用して得ることができます。DIMsLocale についての詳細は、第 1.5 節を参照してください。

このリソースは DIMs クラスの作成に必須です。

1.4.2 DIMsNserverName

接続に使用される一連のアトム名を指定します。このリソースの値は、DIMsStringList型で、次の構造体へのポインタです。

```
typedef struct_DIMsStringList {
    short   num_of_strings;
    char    **string_names;
} DIMsStringListRec, *DIMsStringList;
```

num_of_stringsフィールドには、接続に使用される別名数を指定します。string_namesフィールドには、接続に使用されるアトムの名前を指定するヌル終了文字列の配列を指定します。たとえば OpenVMS では、日本語 IMS の省略時の名前は DEC であり、次のようなアトムが作成されます。

```
_XIM_ja_JP@DEC
```

@DEC で終わるアトムは弊社専用ですので、同じ名前でアトムを作成しないでください。OpenVMS XIM と IMS との接続方法については、第 1.3.2 項および第 1.3.3 項を参照してください。

このリソースは DIMs クラスの作成に必須です。

1.4.3 DIMsNversion

DIMs ウィジェットがサポートする IM プロトコルのバージョンを範囲で指定します。このリソースの値はDIMsVersion型で、次の構造体へのポインタです。

```
typedef struct_DIMsVersion {
    short    highest_major_version;
    short    highest_minor_version;
    short    lowest_major_version;
    short    lowest_minor_version;
} DIMsVersionRec, *DIMsVersion;
```

highest_major_versionフィールドには、DIMs クラスがサポートする IM プロトコルの最高メジャー・バージョンを指定します。

highest_minor_versionフィールドには、DIMs クラスがサポートする IM プロトコルの最高マイナー・バージョンを指定します。

lowest_major_versionフィールドには、DIMs クラスがサポートする IM プロトコルの最低メジャー・バージョンを指定します。

lowest_minor_versionフィールドには、DIMs クラスがサポートする IM プロトコルの最低マイナー・バージョンを指定します。

本バージョンでは通常次のように設定します。

```
highest_major_version = 3
highest_minor_version = 0
lowest_major_version = 3
lowest_minor_version = 0
```

このリソースは任意です。

1.4.4 DIMsNextIMAttr

DIMs ウィジェットがサポートする拡張 XIM 属性 (XIM 値) を指定します。このリソースが指定されると、クライアントは、これらの拡張 XIM 値に対して XGetIMValues()を使用することができます。このリソースの値はDIMsExtAttr型で、次の構造体へのポインタです。

```
typedef struct_DIMsExtAttr {
    short    num_of_ext_attr;
    Attr     ext_attr;
} DIMsExtAttrRec, *DIMsExtAttr;
```

num_of_ext_attrフィールドには、拡張 XIM 属性の数を指定します。ext_attrフィールドには、拡張 XIM の属性の配列を指定します。このフィールドの値はAttr型で、次の構造体へのポインタです。


```
typedef struct Attr {
    unsigned short type;
    char *attributes;
} AttrRec, *Attr;
```

typeフィールドには、属性の型を定義する次のいずれかの値を指定します。

```
#define byte_t          1      /* 8 ビット符号なし整数 */
#define word_t          2      /* 16 ビット符号なし整数 */
#define long_t          3      /* 32 ビット符号なし整数 */
#define string_t        4      /* char の配列 */
#define window_t        5      /* Window フォーマット */
#define ximstyles_t     10     /* XIMStyles フォーマット */
#define xrectangle_t    11     /* XRectangle フォーマット */
#define xpoint_t        12     /* XPoint フォーマット */
#define xfontset_t       13     /* XFontSet フォーマット */
#define ximhotkeytriggers_t 15  /* HotKeyTriggers フォーマット */
#define ximhotkeystate_t 16    /* HotKeyState フォーマット */
#define ximstringconversion_t 17 /* StringConversion フォーマット */
#define ximpreeditstate 18     /* Preedit State フォーマット */
#define ximresetstate   19     /* Reset State フォーマット */
#define nestlist_t      0x7fff /* Nested List フォーマット */
```

attributesフィールドには、拡張 XIM 値の名前を定義します。この値はヌル終了文字列でなければなりません。

このリソースは任意です。

1.4.5 DIMsNextICAttr

DIMs ウィジェットがサポートする拡張 XIC 属性 (XIC 値) を指定します。このリソースが指定されると、クライアントはこれらの拡張 XIC 値に対して XGetICValues() および XSetICValues() を使用することができます。このリソースの値は DIMsExtAttr 型で、次の構造体へのポインタです。

```
typedef struct DIMsExtAttr {
    short num_of_ext_attr;
    Attr ext_attr;
} DIMsExtAttrRec, *DIMsExtAttr;
```

num_of_ext_attrフィールドには、拡張 XIC 属性の数を指定します。ext_attrフィールドには、拡張 XIC の配列を指定します。このフィールドの値はAttr型で、次の構造体へのポインタです。

```
typedef struct Attr {
    unsigned short type;
    char *attributes;
} AttrRec, *Attr;
```

typeフィールドには、属性の型を定義する次のいずれかの値を指定します。

```
#define byte_t          1      /* 8 ビット符号なし整数 */
#define word_t          2      /* 16 ビット符号なし整数 */
#define long_t          3      /* 32 ビット符号なし整数 */
#define string_t        4      /* char の配列 */
#define window_t        5      /* Window フォーマット */
#define ximstyles_t     10     /* XIMStyles フォーマット */
#define xrectangle_t    11     /* XRectangle フォーマット */
#define xpoint_t        12     /* XPoint フォーマット */
#define xfontset_t      13     /* XFontSet フォーマット */
#define ximhotkeytriggers_t 15  /* HotKeyTriggers フォーマット */
#define ximhotkeystate_t 16    /* HotKeyState フォーマット */
#define ximstringconversion_t 17 /* StringConversion フォーマット */
#define ximpreeditstate 18     /* Preedit State フォーマット */
#define ximresetstate   19     /* Reset State フォーマット */
#define nestlist_t      0x7fff /* Nested List フォーマット */
```

attributesフィールドには、拡張 XIC 値の名前を定義します。この値はヌル終了文字列でなければなりません。

このリソースは任意です。

1.4.6 DIMsNsupportStyles

DIMs ウィジェットがサポートする入力スタイルを指定します。このリソースの値はDIMsSupportStyles型で、次の構造体へのポインタです。

```
typedef struct _DIMsSupportStyles {
    short    num_support_styles;
    XIMStyle *support_styles;
} DIMsSupportStylesRec, *DIMsSupportStyles;
```

num_support_stylesフィールドには、サポートする入力スタイルの数を指定します。support_stylesフィールドには、XIMStyle の配列を指定します。

クライアントがIMS と接続したら、クライアントが選択した入力スタイルをDIMsInputStyle(w) を使用して得ることができます。DIMsInputStyle についての詳細は第 1.5 節を参照してください。

このリソースは必須です。

1.4.7 DIMsNsupportEncodings

DIMs ウィジェットがサポートするエンコーディングを指定します。XIM ライブラリはIMS に対して、確定した文字列または前編集の文字列に使用するエンコーディングを決定するよう要求します。IMS は省略時のエンコーディングとして少

なくとも COMPOUND_TEXT をサポートする必要があります。このリソースの値は DIMsStringList 型で、次の構造体へのポインタです。

```
typedef struct_DIMsStringList {
    short    num_of_strings;
    char     **string_names;
} DIMsStringListRec, *DIMsStringList;
```

num_of_strings フィールドには、サポートするエンコーディングの数を指定します。

string_names フィールドには、DIMs ウィジェットのエンコーディングを優先順位の高い順に定義するヌル終了文字列の配列を指定します。たとえば、通信のためのエンコーディングとして STRING を使用したい場合、string_names フィールドに STRING、COMPOUND_TEXT の順でエンコーディングを指定します。XIM ライブラリが STRING エンコーディングをサポートする場合は STRING が通信に使用され、STRING エンコーディングをサポートしない場合は COMPOUND_TEXT が使用されます。

これらの2つのフィールドが指定されていない場合、XIM ライブラリはエンコーディングとして COMPOUND_TEXT を使用します。クライアントが IMS と接続したら、サポートされているエンコーディングは DIMsEncoding(w) を使用して得ることができます。DIMsEncoding についての詳細は、第 1.5 節を参照してください。

1.4.8 DIMsNtriggerKeys

IMS のトリガ・キーのオン/オフを指定します。このリソースの値は DIMsTriggerKeys 型で、次の構造体へのポインタです。

```
typedef struct_DIMsTriggerKeys {
    short    num_of_on_keys;
    XIMTriggerkey on_keys;
    short    num_of_off_keys;
    XIMTriggerkey off_keys;
} DIMsTriggerKeysRec, *DIMsTriggerKeys;
```

on_keys には、入力メソッドをトリガ・オンできるキーのリストを指定します。num_of_on_keys には、リストに含まれるキーの数を指定します。

off_keys には、入力メソッドをトリガ・オフさせるキーのリストを指定します。num_of_off_keys には、リストに含まれるキーの数を指定します。

XIMTriggerkey は次の構造体へのポインタです。

```
typedef struct _XIMTriggerkey
{
    KeySym keysym;
    unsigned long modifier;
    unsigned long modifier_mask;
} XIMTriggerkeyRec, *XIMTriggerkey;
```

keysymフィールドには、たとえば XK_SPACE のような、入力メソッドをトリガ・オン/オフさせるためのキー・シンボルを指定します。

modifierフィールドには、トリガ・キー・シーケンスのキー・マスクを指定します。ShiftMask、ControlMask、Mod1Mask (コンポーズ・マスク)、あるいはこれらの組み合わせのいずれかです。

modifier_maskは、KeyPress イベントがトリガ・キーであるかどうかの評価に使用されるマスクです。たとえば、トリガー・オン (開始変換) キーが Ctrl ~ Shift<key>スペースである場合、有効なトリガ・オン・キーの修飾子の状態は、ControlMask に対しては on、ShiftMask に対しては off です。したがって、modifier_maskフィールドを ControlMask | ShiftMask と設定するのに対して、modifierフィールドは ControlMask と設定します。

入力メソッドのトリガ・オン/オフのテストのため、受け入れるどのキー・イベントの修飾子も、最初にmodifier_maskによってマスキングされます。この処理の結果は、修飾子の値と等しくなければなりません。

このリソースは、DIMsSetTriggerKeys()を呼び出すことによりいつでも変更することができます。DIMsSetTriggerKeys についての詳細は、第 1.5 節を参照してください。

このリソースは任意です。このリソースを指定しない場合は静的イベント・フローが使用されるため、すべてのイベントがクライアントから IMS に転送されます。イベント・フロー・モデルについての詳細は、第 1.3.3 項を参照してください。

1.4.9 DIMsNgetExtIMValuesCb

拡張 XIM 属性の値を得るために、クライアントが XGetIMValues を呼び出すときに呼び出されるコールバックを指定します。詳細については、第 1.5 節を参照してください。

このリソースは任意です。

1.4.10 DIMsNcreateICCb

クライアントが XCreateIC を呼び出すときに呼び出されるコールバックを指定します。このコールバックを呼び出す前に DIMs ウィジェットが作成されます。IMS プログラマは、すべての初期化作業を行い、通常は DIMs ウィジェットに対応する入力メソッド特定データを作成する必要があります。詳細については、第 1.5 節を参照してください。

このリソースは任意です。

1.4.11 DIMsNdestroyICCb

クライアントが XDestroyIC を呼び出すときに呼び出されるコールバックを指定します。IMS プログラマは、DIMs ウィジェット作成時に作成したすべてのデータを解放しなければなりません。詳細については、第 1.5 節を参照してください。

このリソースは任意です。

1.4.12 DIMsNsetExtICValuesCb

拡張 XIC 属性の値を設定するために、クライアントが XSetICValues を呼び出すときに呼び出されるコールバックを指定します。詳細は、第 1.5 節を参照してください。

このリソースは任意です。

1.4.13 DIMsNgetExtICValuesCb

拡張 XIC 属性の値を取得するために、クライアントが XGetICValues を呼び出すときに呼び出されるコールバックを指定します。詳細は、第 1.5 節を参照してください。

このリソースは任意です。

1.4.14 DIMsNsetICFocusCb

クライアントが XSetICFocus を呼び出すときに呼び出されるコールバックを指定します。詳細は、第 1.5 節を参照してください。

このリソースは任意です。

1.4.15 DIMsNunsetICFocusCb

クライアントが XUnsetICFocus を呼び出すときに呼び出されるコールバックを指定します。詳細は、第 1.5 節を参照してください。

このリソースは任意です。

1.4.16 DIMsNresetICCb

クライアントが XmbResetIC または XwcResetIC を呼び出すときに呼び出されるコールバックを指定します。詳細は、第 1.5 節を参照してください。

このリソースは任意です。

1.4.17 DIMsNprocessEventCb

クライアントが IMS に要求されたイベントを転送するときに呼び出されるコールバックを指定します。詳細は、第 1.5 節を参照してください。

このリソースは必須です。

1.4.18 DIMsNsetAreaCb

XIC 属性の 1 つとして XNArea を設定するために、クライアントが XSetICValues を呼び出すときに呼び出されるコールバックを指定します。詳細は、第 1.5 節を参照してください。

このリソースは任意です。

1.4.19 DIMsNsetAreaNeededCb

XIC 属性の 1 つとして XNAreaNeeded を設定するために、クライアントが XSetICValues を呼び出すときに呼び出されるコールバックを指定します。詳細は、第 1.5 節を参照してください。

このリソースは任意です。

1.4.20 DIMsNsetSpotLocationCb

XIC 属性の 1 つとして XNSpotLocation を設定するために、クライアントが XSetICValues を呼び出すときに呼び出されるコールバックを指定します。詳細は、第 1.5 節を参照してください。

このリソースは任意です。

1.4.21 DIMsNsetColormapCb

XIC 属性の 1 つとして XNColormap を設定するために、クライアントが XSetICValues を呼び出すときに呼び出されるコールバックを指定します。詳細は、第 1.5 節を参照してください。

このリソースは任意です。

1.4.22 DIMsNsetStdColormapCb

XIC 属性の 1 つとして XNStdColormap を設定するために、クライアントが XSetICValues を呼び出すときに呼び出されるコールバックを指定します。詳細は、第 1.5 節を参照してください。

このリソースは任意です。

1.4.23 DIMsNsetForegroundCb

XIC 属性の 1 つとして XNForeground を設定するために、クライアントが XSetICValues を呼び出すときに呼び出されるコールバックを指定します。詳細は、第 1.5 節を参照してください。

このリソースは任意です。

1.4.24 DIMsNsetBackgroundCb

XIC 属性の 1 つとして XNBackground を設定するために、クライアントが XSetICValues を呼び出すときに呼び出されるコールバックを指定します。詳細は、第 1.5 節を参照してください。

このリソースは任意です。

1.4.25 DIMsNsetBgPixmapCb

XIC 属性の 1 つとして XNBackgroundPixmap を設定するために、クライアントが XSetICValues を呼び出すときに呼び出されるコールバックを指定します。詳細は、第 1.5 節を参照してください。

このリソースは任意です。

1.4.26 DIMsNsetFontSetCb

XIC 属性の 1 つとして XNFontSet を設定するために、クライアントが XSetICValues を呼び出すときに呼び出されるコールバックを指定します。詳細は、第 1.5 節を参照してください。

このリソースは任意です。

1.4.27 DIMsNsetLineSpaceCb

XIC 属性の 1 つとして XNLineSpace を設定するために、クライアントが XSetICValues を呼び出すときに呼び出されるコールバックを指定します。詳細は、第 1.5 節を参照してください。

このリソースは任意です。

1.4.28 DIMsNsetCursorCb

XIC 属性の 1 つとして XNCursor を設定するために、クライアントが XSetICValues を呼び出すときに呼び出されるコールバックを指定します。詳細は、第 1.5 節を参照してください。

このリソースは任意です。

1.4.29 DIMsNgetFilterEventsCb

XIC 属性の 1 つとして XNFilterEvents を取得するために、クライアントが XGetICValues を呼び出すときに呼び出されるコールバックを指定します。詳細は、第 1.5 節を参照してください。

このリソースは任意です。このリソースが指定されない場合、IMSSL は KeyPressMask | KeyReleaseMask をクライアントに返します。

1.4.30 DIMsNgetAreaCb

XIC 属性の 1 つとして XNArea を取得するために、クライアントが XGetICValues を呼び出すときに呼び出されるコールバックを指定します。詳細は、第 1.5 節を参照してください。

このリソースは任意です。

1.4.31 DIMsNgetAreaNeededCb

XIC 属性の 1 つとして XNAreaNeeded を取得するために、クライアントが XGetICValues を呼び出すときに呼び出されるコールバックを指定します。詳細は、第 1.5 節を参照してください。

このリソースは任意です。

1.4.32 DIMsNgetSpotLocationCb

XIC 属性の 1 つとして XNSpotLocation を取得するために、クライアントが XGetICValues を呼び出すときに呼び出されるコールバックを指定します。詳細は、第 1.5 節を参照してください。

このリソースは任意です。

1.4.33 DIMsNgetColormapCb

XIC 属性の 1 つとして XNColormap を取得するために、クライアントが XGetICValues を呼び出すときに呼び出されるコールバックを指定します。詳細は、第 1.5 節を参照してください。

このリソースは任意です。

1.4.34 DIMsNgetStdColormapCb

XIC 属性の 1 つとして XNStdColormap を取得するために、クライアントが XGetICValues を呼び出すときに呼び出されるコールバックを指定します。詳細は、第 1.5 節を参照してください。

このリソースは任意です。

1.4.35 DIMsNgetForegroundCb

XIC 属性の 1 つとして XNForeground を取得するために、クライアントが XGetICValues を呼び出すときに呼び出されるコールバックを指定します。詳細は、第 1.5 節を参照してください。

このリソースは任意です。

1.4.36 DIMsNgetBackgroundCb

XIC 属性の 1 つとして XNBackground を設定するために、クライアントが XGetICValues を呼び出すときに呼び出されるコールバックを指定します。詳細は、第 1.5 節を参照してください。

このリソースは任意です。

1.4.37 DIMsNgetBgPixmapCb

XIC 属性の 1 つとして XNBackgroundPixmap を設定するために、クライアントが XGetICValues を呼び出すときに呼び出されるコールバックを指定します。詳細は、第 1.5 節を参照してください。

このリソースは任意です。

1.4.38 DIMsNgetFontSetCb

XIC 属性の 1 つとして XNFontSet を取得するために、クライアントが XGetICValues を呼び出すときに呼び出されるコールバックを指定します。詳細は、第 1.5 節を参照してください。

このリソースは任意です。

1.4.39 DIMsNgetLineSpaceCb

XIC 属性の 1 つとして XNLineSpace を設定するために、クライアントが XGetICValues を呼び出すときに呼び出されるコールバックを指定します。詳細は、第 1.5 節を参照してください。

このリソースは任意です。

1.4.40 DIMsNgetCursorCb

XIC 属性の 1 つとして XNCursor を設定するために、クライアントが XGetICValues を呼び出すときに呼び出されるコールバックを指定します。詳細は、第 1.5 節を参照してください。

このリソースは任意です。

1.4.41 DIMsNtriggerNotifyCb

DIMsNtriggerKeys で設定されたトリガ・キーを受け取ったときに呼び出されるコールバックを指定します。詳細は、第 1.5 節を参照してください。

このリソースは任意です。

1.4.42 DIMsNtransport

DIMs ウィジェットがサポートするトランスポート層を指定します。IMSSL は TCP/IP, DECnet, X の複数のトランスポート層をサポートし, IMS プログラマはそれらからサポートしたいトランスポート・メカニズムを選択できます。この値は優先順にカンマで区切られたリストからなるヌル終了文字列でなければなりません。X11R6 クライアントは XMODIFIER 環境変数を設定することにより使用したいトランスポート層を指定できます。各トランスポート層の書式は以下の通りです。

- 内部ドメイン

システム内部ドメイン名に対して使われる書式は以下の通りです。

"local/"ホスト名":"パス名

ここでパス名はソケット・アドレスのパス名, ホスト名はシンボリックあるいは 10 進表記のいずれかになります。

- TCP

インターネット・ドメイン名に対して使われる書式は以下の通りです。

"tcp/"ホスト名":"ipport番号

ここで ipport 番号は IM サーバが接続を待つポートです。

- DECnet

DECnet 名に対して使われる書式は以下の通りです。

"decnet/"ノード名 "::IMSERVER\$"オブジェクト名

ここでノード名は DECnet アドレスのシンボリックあるいは 10 進表記のいずれかになります。オブジェクト名は (大文字, 小文字を区別しない) DECnet オブジェクト名です。

- X

X 名に対して使われる書式は以下の通りです。

"X/"

IM サーバがこれら全てのトランスポートをサポートする場合, 以下の例のようにこれら全ての名前を優先順に指定する必要があります。

"local/expo.lcs.mit.edu:/tmp/.ximsocket,tcp/expo.lcs.mit.edu:8012,decnet/44.70::IMSERVER\$XIMOBJECT,X/"

このリソースは任意です。このリソースを指定しない場合，"X/"が設定されているものとします。

1.4.43 DIMsNsetStringConversionCb

XIC 属性の 1 つとして XNStringConversion を設定するために，クライアントが XSetICValues を呼び出すときに呼び出されるコールバックを指定します。詳細は，第 1.5 節を参照してください。

このリソースは任意です。

1.4.44 DIMsNsetStringConversionCallbackCb

XIC 属性の 1 つとして XNStringConversionCallback を設定するために，クライアントが XSetICValues を呼び出すときに呼び出されるコールバックを指定します。詳細は，第 1.5 節を参照してください。

このリソースは任意です。

1.4.45 DIMsNsetResetStateCb

XIC 属性の 1 つとして XNResetState を設定するために，クライアントが XSetICValues を呼び出すときに呼び出されるコールバックを指定します。詳細は，第 1.5 節を参照してください。

このリソースは任意です。

1.4.46 DIMsNgetResetStateCb

XIC 属性の 1 つとして XNResetState を取得するために，クライアントが XGetICValues を呼び出すときに呼び出されるコールバックを指定します。詳細は，第 1.5 節を参照してください。

このリソースは任意です。

1.4.47 DIMsNsetHotKeyCb

XIC 属性の 1 つとして XNHotKey を設定するために，クライアントが XSetICValues を呼び出すときに呼び出されるコールバックを指定します。詳細は，第 1.5 節を参照してください。

このリソースは任意です。

1.4.48 DIMsNgetHotKeyCb

XIC 属性の 1 つとして XNHotKey を取得するために、クライアントが XGetICValues を呼び出すときに呼び出されるコールバックを指定します。詳細は、第 1.5 節を参照してください。

このリソースは任意です。

1.4.49 DIMsNsetHotKeyStateCb

XIC 属性の 1 つとして XNHotKeyState を設定するために、クライアントが XSetICValues を呼び出すときに呼び出されるコールバックを指定します。詳細は、第 1.5 節を参照してください。

このリソースは任意です。

1.4.50 DIMsNgetHotKeyStateCb

XIC 属性の 1 つとして XNHotKeyState を取得するために、クライアントが XGetICValues を呼び出すときに呼び出されるコールバックを指定します。詳細は、第 1.5 節を参照してください。

このリソースは任意です。

1.4.51 DIMsNsetPreeditStateCb

XIC 属性の 1 つとして XNPreeditState を設定するために、クライアントが XSetICValues を呼び出すときに呼び出されるコールバックを指定します。詳細は、第 1.5 節を参照してください。

このリソースは任意です。

1.4.52 DIMsNgetPreeditStateCb

XIC 属性の 1 つとして XNPreeditState を取得するために、クライアントが XGetICValues を呼び出すときに呼び出されるコールバックを指定します。詳細は、第 1.5 節を参照してください。

このリソースは任意です。

1.4.53 DIMsNsetPreeditStateCallbackCb

XIC 属性の 1 つとして XNPreeditStateNotifyCallbackCb を設定するために、クライアントが XSetICValues を呼び出すときに呼び出されるコールバックを指定します。詳細は、第 1.5 節を参照してください。

このリソースは任意です。

1.4.54 DIMsNclientdata

上記のコールバック・ルーチンへの引数の 1 つとして渡されるクライアント・データを指定します。

このリソースは任意です。

1.5 パブリック・ルーチン

この節では、コールバック・ルーチンを含む IMSSL のすべてのパブリック・ルーチンの使用方法を記述します。

DIMsClassCtxCreate

DIMs ウィジェットで使用するクラス・コンテキストを作成する関数です。

フォーマット

DIMsClassCtx DIMsClassCtxCreate (*head*, *name*, *arglist*, *argcount*)

引数

DIMsClassCtx	<i>head</i> ;
char	<i>*name</i> ;
ArgList	<i>arglist</i> ;
Cardinal	<i>argcount</i> ;

戻り値

DIMs クラス・コンテキスト ID が返されます。

説明

DIMsClassCtxCreateは、DIMs ウィジェットのクラス・コンテキストのインスタンスを作成し、対応するクラス・コンテキスト ID を返します。この関数を呼び出すことによって、このクラス・コンテキストのリストを作成することができます。

<i>head</i>	DIMs クラス・コンテキスト・リストのヘッドを指定します。この関数をはじめて呼び出す場合は、NULL 値を渡すことができます。別のクラス・コンテキストを作成する場合には、最初の呼び出しに対する戻り値をこの関数に渡す必要があります。
<i>name</i>	DIMs クラス名を指定します。DIMs クラス名は呼び出し後に解放されるヌル終了文字列でなければなりません。
<i>arglist</i>	DIMs クラスのリソースの名称と値を含む引数リストを指定します。リソースに割り当てられたメモリは、呼び出した後に解放することができます。
<i>argcount</i>	引数リスト内の引数の数を指定します。

このルーチンは、DIMsClassInitialize()が呼び出される前に、最低 1 回呼び出す必要があります。

DIMs クラスのリソースの完全な定義については、第 1.4 節を参照してください。

DIMsGetClassValues

DEC 入力サーバ・サービス層を初期化する関数です。

フォーマット

```
char *DIMsGetClassValues (w, arglist, argcount)
```

引数

Widget	<i>w</i> ;
Arglist	<i>arglist</i> ;
Cardinal	<i>argcount</i> ;

戻り値

戻り値はありません。

説明

DIMsSLInitializeは、DIMs クラス・コンテキストを作成した後に 1 回だけ呼び出します。このルーチンは IMSSL を初期化します。

<i>toplevel</i>	(XtAppInitialize()などによって返される) アプリケーション・シェル・ウィジェットのウィジェット ID を指定します。
<i>ctx</i>	DIMs ウィジェット・クラス・リストのヘッドを指定します。

DIMsClassCtxGetValues

DEC 入力サーバ・サービス層を初期化する関数です。

フォーマット

```
char *DIMsClassCtxGetValues (ctx, arglist, argcount)
```

引数

DIMsClassCtx	<i>ctx</i> ;
Arglist	<i>arglist</i> ;
Cardinal	<i>argcount</i> ;

戻り値

戻り値はありません。

説明

DIMsSLInitializeは、DIMs クラス・コンテキストを作成した後に 1 回だけ呼び出します。このルーチンは IMSSL を初期化します。

<i>toplevel</i>	(XtAppInitialize())などによって返される) アプリケーション・シェル・ウィジェットのウィジェット ID を指定します。
<i>ctx</i>	DIMs ウィジェット・クラス・リストのヘッドを指定します。

DIMsSLInitialize

DEC 入力サーバ・サービス層を初期化する関数です。

フォーマット

```
void DIMsSLInitialize (toplevel, ctx)
```

引数

Widget	<i>toplevel</i> ;
DIMsClassCtx	<i>ctx</i> ;

戻り値

戻り値はありません。

説明

DIMsSLInitializeは、DIMs クラス・コンテキストを作成した後に 1 回だけ呼び出します。このルーチンは IMSSL を初期化します。

<i>toplevel</i>	(XtAppInitialize())などによって返される) アプリケーション・シェル・ウィジェットのウィジェット ID を指定します。
<i>ctx</i>	DIMs ウィジェット・クラス・リストのヘッドを指定します。

DIMsProtolnit

プロトコル層を初期化する関数です。

フォーマット

```
void DIMsProtolnit (toplevel, app_context)
```

引数

Widget	<i>toplevel</i> ;
XtAppContext	<i>app_context</i> ;

戻り値

戻り値はありません。

説明

DIMsProtolnitはプロトコル層を初期化します。このルーチンは、アプリケーション・シェル・ウィジェットが実現された後に1回だけ呼び出されなければなりません。

<i>toplevel</i>	(XtAppInitialize()などによって返される) アプリケーション・シェル・ウィジェットのウィジェット ID を指定します。 <i>toplevel</i> は、この関数が呼び出される前に実現されていなければなりません。
<i>app_context</i>	IMS のアプリケーション・コンテキストを指定します。IMS プログラマは、XtAppInitialize()またはIntrinsics アプリケーション・コンテキスト初期化ルーチンを使用することによって、IMS アプリケーション・コンテキストを得ることができます。

DIMsPreEditStart

DIMs ウィジェットに対応する入力コンテキストの前編集セッションを開始する関数です。

フォーマット

```
int DIMsPreEditStart (w)
```

引数

Widget *w*;

戻り値

正の戻り値は、クライアントが前編集文字列に使用できる最大バイト数です。
戻り値 -1 は、前編集文字列のサイズが無制限であることを示します。

説明

DIMsPreEditStartは前編集セッションを開始します。クライアント側では、PreeditStart コールバックが発行されると前編集バッファで使用できる最大値が返されます。この呼び出しは入力スタイルが on-the-spot の場合に使用され、KeyRelease イベントによってトリガされます。詳細については、processEventCb の項を参照してください。

この呼び出しは、createICCb(), または`reason`引数として DimCR_CreateIC を設定するようなコールバックでは、クライアントと IMS の接続がまだ確立されていないため使用できないことに注意してください。このようなコールバックの詳細については、setAreaCb の項およびそれ以降のいくつかの項を参照してください。

w 入力コンテキストに対応する DIMs ウィジェットを指定します。

DIMsPreEditDone

DIMs ウィジェットに対応する入力コンテキストの前編集セッションを終了させる関数です。

フォーマット

```
void DIMsPreEditDone (w)
```

引数

Widget *w*;

戻り値

戻り値はありません。

説明

DIMsPreEditDoneは前編集セッションを終了させます。クライアント側では、PreeditDone コールバックが発行されると前編集バッファが解放されます。この呼び出しはスタイルが on-the-spot の場合のみ使用できます。

この呼び出しは、createICCb(), または*reason*引数として DimCR_CreateIC を設定するようなコールバックでは、クライアントと IMS の接続がまだ確立されていないため使用できないことに注意してください。このようなコールバックの詳細については、setAreaCb の項およびそれ以降のいくつかの項を参照してください。

w 入力コンテキストに対応する DIMs ウィジェットを指定します。

DIMsPreEditStateNotify

DIMs ウィジェットに対応する入力コンテキストの前編集セッションが開始されているかどうかをチェックする関数です。

フォーマット

```
void DIMsPreEditStateNotify (w, state)
```

引数

Widget	<i>w</i> ;
XIMPreeditState	<i>state</i> ;

戻り値

DIMs ウィジェットの前編集セッションが開始されている場合は、True が返されます。DIMs ウィジェットの前編集セッションが開始されていない場合は、False が返されます。

説明

DIMsIsStateは、DIMs ウィジェットの前編集セッションが開始されているかどうかを示すブール値を返します。この呼び出しは入力スタイルが on-the-spot の場合のみ使用できます。

w 入力コンテキストに対応する DIMs ウィジェットを指定します。

DIMsIsIState

DIMs ウィジェットに対応する入力コンテキストの前編集セッションが開始されているかどうかをチェックする関数です。

フォーマット

Boolean DIMsIsIState (*w*)

引数

Widget *w*;

戻り値

DIMs ウィジェットの前編集セッションが開始されている場合は、True が返されます。DIMs ウィジェットの前編集セッションが開始されていない場合は、False が返されます。

説明

DIMsIsIStateは、DIMs ウィジェットの前編集セッションが開始されているかどうかを示すブール値を返します。この呼び出しは入力スタイルが on-the-spot の場合のみ使用できます。

w 入力コンテキストに対応する DIMs ウィジェットを指定します。

DIMsPreEditDraw

前編集セッションを開始した入力コンテキストで、文字列を挿入/削除/置換する関数です。

フォーマット

```
void DIMsPreEditDraw (w, caret, chg_first, chg_length, string, string_length, renditions, rendition_length)
```

引数

Widget	<i>w;</i>
int	<i>caret;</i>
int	<i>chg_first;</i>
int	<i>chg_length;</i>
void	<i>*string;</i>
int	<i>string_length;</i>
XIMFeedback	<i>*renditions;</i>
int	<i>rendition_length;</i>

戻り値

戻り値はありません。

説明

DIMsPreEditDrawは、入力コンテキストの前編集文字列を挿入/削除/置換するために使用されます。クライアント側では、PreeditDraw コールバックが発行されると、このルーチンの引数に与えられた情報に従って編集バッファが挿入/削除/置換されます。この呼び出しは入力スタイルが on-the-spot の場合のみ使用できます。

この呼び出しは、createICCb(), または*reason*引数として DimCR_CreateIC を設定するようなコールバックでは、クライアントと IMS の接続がまだ確立されていないため使用できないことに注意してください。このようなコールバックの詳細については、setAreaCb の項およびそれ以降のいくつかの項を参照してください。

<i>w</i>	入力コンテキストに対応する DIMs ウィジェットを指定します。
<i>caret</i>	前編集文字列のカーソル・オフセット (文字単位) を指定します。

<i>chg_first</i>	変更部分の開始位置 (文字単位) を指定します。
<i>chg_length</i>	変更部分の長さ (文字単位) を指定します。
<i>string</i>	入力コンテキストに渡す文字列を指定します。エンコーディングは DIMsEncoding(w) の戻り値に一致しなければなりません。IMS プログラムは、このデータにメモリを割り当てたら呼び出し後に解放する必要があります。
<i>string_length</i>	<i>string</i> のバイト数を指定します。
<i>renditions</i>	文字ごとの表示情報を持つ XIMFeedback の配列へのポインタを指定します。
<i>redition_length</i>	<i>renditions</i> のメンバ数を指定します。

DIMsPreEditCaret

前編集の際にテキスト挿入位置を移動する関数です。

フォーマット

```
void DIMsPreEditCaret (w, position, direction, style)
```

引数

Widget	<i>w</i> ;	
int	<i>*position</i> ;	(呼び出し側へ返す)
XIMCaretDirection	<i>direction</i> ;	
XIMCaretStyle;	<i>style</i> ;	

戻り値

戻り値はありません。

説明

DIMsPreEditCaretは、前編集の際に、テキスト挿入位置を移動させる関数です。クライアント側では、PreeditCaret コールバックが発行されると新しいカレットの位置が返されます。この関数を呼び出す場合には、*position* 引数にカレットの位置を指定しなければなりません。この関数は、同じ引数で前編集文字列の新しいオフセットを返します。この呼び出しは入力スタイルが on-the-spot の場合のみ使用できます。

この呼び出しは、createICCb(), または *reason* 引数として DimCR_CreateIC を設定するようなコールバックでは、クライアントと IMS の接続がまだ確立されていないため使用できないことに注意してください。このようなコールバックの詳細については、setAreaCb の項およびそれ以降のいくつかの項を参照してください。

<i>w</i>	入力コンテキストに対応する DIMs ウィジェットを指定します。
<i>position</i>	前編集文字列内のカレット・オフセットを指定します。呼び出し後には、初期位置からの新しいオフセット値が返されます。
<i>direction</i>	カレットが移動する方向を指定します。
<i>style</i>	カレットのフィードバックを指定します。

DIMsStatusStart

DIMs ウィジェットに対応する入力コンテキストの状態更新セッションを開始する関数です。

フォーマット

```
void DIMsStatusStart (w)
```

引数

Widget *w*;

戻り値

戻り値はありません。

説明

DIMsStatusStartは、状態領域を更新する前に呼び出される必要があります。クライアント側では StatusStart が発行されます。この呼び出しは XIMStatusCallback にのみ使用できます。

この呼び出しは、createICCb()、または*reason*引数として DimCR_CreateIC を設定するようなコールバックでは、クライアントと IMS の接続がまだ確立されていないため使用できないことに注意してください。このようなコールバックの詳細については、setAreaCb の項およびそれ以降のいくつかの項を参照してください。

w 入力コンテキストに対応する DIMs ウィジェットを指定します。

DIMsStatusDone

DIMs ウィジェットに対応する入力コンテキストの状態更新セッションを終了させます。

フォーマット

```
void DIMsStatusDone (w)
```

引数

Widget *w*;

戻り値

戻り値はありません。

説明

DIMsStatusDoneは、状態領域を更新した後に呼び出す必要があります。クライアント側では、StatusDone コールバックが発行されます。この呼び出しは、XIMStatusCallback にのみ使用できます。

この呼び出しは、createICCb(), または*reason*引数として DimCR_CreateIC を設定するようなコールバックでは、クライアントと IMS の接続がまだ確立されていないため使用できないことに注意してください。このようなコールバックの詳細については、setAreaCb の項およびそれ以降のいくつかの項を参照してください。

w 入力コンテキストに対応する DIMs ウィジェットを指定します。

DIMsStatusDraw

DIMs ウィジェットに対応する入力コンテキストの状態領域を更新する関数です。

フォーマット

```
void DIMsStatusDraw (w, type, string, string_length, renditions, renditions_length, bitmap)
```

引数

Widget	<i>w</i> ;
XIMStatusDataType	<i>type</i> ;
void	<i>*string</i> ;
int	<i>string_length</i> ;
XIMFeedback	<i>*renditions</i> ;
int	<i>rendition_length</i> ;
Pixmap	<i>bitmap</i> ;

戻り値

戻り値はありません。

説明

DIMsStatusDrawは、入力コンテキストの状態領域を更新します。クライアント側では、StatusDraw コールバックが発行されます。この呼び出しは、XIMStatusCallback にのみ使用できます。

この呼び出しは、createICCb(), または*reason*引数として DimCR_CreateIC を設定するようなコールバックでは、クライアントと IMS の接続がまだ確立されていないため使用できないことに注意してください。このようなコールバックの詳細については、setAreaCb の項およびそれ以降のいくつかの項を参照してください。

<i>w</i>	入力コンテキストに対応する DIMs を指定します。
<i>type</i>	テキストまたはビットマップのいずれを使用するかを指定します。
<i>string</i>	状態領域を更新するためのテキストを指定します。エンコーディングは DIMsEncoding(w) の戻り値と一致しなければなりません。このデータにメモリを割り当てたら呼び出し後に解放する必要があります。

<i>string_length</i>	<i>string</i> のバイト数を指定します。
<i>renditions</i>	文字ごとの表示情報を持つ XIMFeedback の配列へのポインタを指定します。
<i>rendition_length</i>	<i>renditions</i> のメンバ数を指定します。
<i>bitmap</i>	<i>type</i> が XIMBitmapType の場合にビットマップを指定します。

DIMsGeometryNegotiation

クライアントと、ジオメトリについての折衝を開始する関数です。

フォーマット

Boolean DIMsGeometryNegotiation (*w*)

引数

Widget *w*;

戻り値

クライアントが GeometryNegotiation コールバックをサポートする場合は、True が返されます。クライアントが GeometryNegotiation コールバックをサポートしない場合は、False が返されます。

説明

DIMsGeometryNegotiationは、ジオメトリについてのクライアントとの折衝を開始します。クライアント側では、GeometryNegotiation コールバックが発行されます。この呼び出しは入力スタイルが off-the-spot の場合のみ使用できます。

この呼び出しは、createICCb()、または*reason*引数として DimCR_CreateIC を設定するようなコールバックでは、クライアントと IMS の接続がまだ確立されていないため使用できないことに注意してください。このようなコールバックの詳細については、setAreaCb の項およびそれ以降のいくつかの項を参照してください。

w 入力コンテキストに対応する DIMs ウィジェットを指定します。

DIMsSetEventsForward

DIMs ウィジェットに対応する入力コンテキストへの、クライアントからのイベント転送を有効または無効にします。

フォーマット

```
void DIMsSetEventsForward (w, state)
```

引数

Widget	<i>w</i> ;
Boolean	<i>state</i> ;

戻り値

戻り値はありません。

説明

DIMsSetEventsForwardは、XFilterEvents()による、クライアントから入力コンテキストへのイベント転送を有効、または無効にします。この関数は、IMS が動的イベント・フロー・モデルをサポートする場合にのみ有用です。イベント・フロー・モデルについての詳細は、第 1.3.3 項を参照してください。

この呼び出しは、createICCb()、または`reason`引数としてDimCR_CreateICを設定するようなコールバックでは、クライアントとIMSの接続がまだ確立されていないため使用できないことに注意してください。このようなコールバックの詳細については、setAreaCbの項およびそれ以降のいくつかの項を参照してください。

<i>w</i>	入力コンテキストに対応する DIMs ウィジェットを指定します。
<i>state</i>	イベント・フロー状態を指定します。クライアントによる入力コンテキストへのイベント転送を有効にする場合は True を指定し、クライアントがトリガオン・キーを受信するまで入力コンテキストへのイベント転送を無効にするには False を指定します。

DIMsIsEventsForwarded

クライアントから DIMs ウィジェットに対応する入力コンテキストにイベントが転送されているかどうかを問い合わせる関数です。

フォーマット

Boolean DIMsIsEventsForwarded (*w*)

引数

Widget *w*;

戻り値

クライアントからのイベント転送が有効になっている場合には、True が返されます。クライアントからのイベントの転送が無効になっている場合には、False が返されます。

説明

DIMsIsEventsForwardedは、入力コンテキストのイベント転送状態を返します。動的イベント・フロー・モデル使用時に、クライアントからのイベント転送を開始/停止する方法については、DIMsSetEventsForward の項を参照してください。イベント・フロー・モデルについての詳細は、第 1.3.3 項を参照してください。

w 入力コンテキストに対応する DIMs ウィジェットを指定します。

DIMsCommit

確定された文字列をクライアントに送信する関数です。

フォーマット

```
void DIMsCommit (w, string, string_length)
```

引数

Widget	<i>w</i> ;
void	<i>*string</i> ;
int	<i>string_length</i> ;

戻り値

戻り値はありません。

説明

DIMsCommitは、確定された文字列をクライアントの入力コンテキストに送信します。クライアントは、XmbLookupString()またはXwcLookupStringを使用して、確定された文字列を得ることができます。入力スタイルがon-the-spotの場合、IMSプログラマは、DIMsPreEditDone(w)を呼び出した後にこの関数を呼び出さなければならないことに注意してください。

この呼び出しは、createICCb()、または*reason*引数としてDimCR_CreateICを設定するようなコールバックでは、クライアントとIMSの接続がまだ確立されていないため使用できないことに注意してください。このようなコールバックの詳細については、setAreaCbの項およびそれ以降のいくつかの項を参照してください。

<i>w</i>	入力コンテキストに対応するDIMsウィジェットを指定します。
<i>string</i>	確定された文字列を指定します。エンコーディングについてはDIMsEncoding()によって返される値に一致する必要があります。詳細についてはDIMsEncodingの項を参照してください。IMSプログラマは、このデータにメモリを割り当てたら呼び出し後に解放する必要があります。
<i>string_length</i>	<i>string</i> のバイト数を指定します。

DIMsEncoding

クライアントと IMS とで同意したエンコーディングを返す関数です。

フォーマット

```
char *DIMsEncoding (w)
```

引数

Widget *w*;

戻り値

クライアントと IMS が同意したエンコーディングを指定するヌル終了文字列を返します。

説明

DIMsEncodingによってクライアントと IMS が同意したエンコーディングを取得することができます。クライアントが XOpenIM() を呼び出すと、XIM ライブラリは使用するエンコーディングについて IMS と折衝し、同意されたエンコーディングが DIMs ウィジェット・データとして IMSSL に保持されます。このエンコーディングは、前編集文字列 (on-the-spot の場合)、確定された文字列、および resetICCb に返される文字列に使用されます。

この関数が返す文字列は解放されてはなりません。

エンコーディングの設定方法についての詳細は、第 1.4.7 項を参照してください。

w 入力コンテキストに対応する DIMs ウィジェットを指定します。

DIMslocale

DIMs ウィジェットに対応する入力コンテキストのロケールを返す関数です。

フォーマット

```
char *DIMsLocale (w)
```

引数

Widget	<i>w</i> ;
--------	------------

戻り値

入力コンテキストのロケールを指定するヌル終了文字列が返されます。

説明

DIMsLocaleによって、入力コンテキストのローケル情報を取得することができます。クライアントが XOpenIM を呼び出すと、Xim に対応するロケールが IMS に送信されます。IMSSL は、このロケールを DIMs ウィジェット・データとして保持します。

返される文字列は解放されてはなりません。

w 入力コンテキストに対応する DIMs ウィジェットを指定します。

DIMsInputStyle

DIMs ウィジェットに対応する入力コンテキストの入力スタイルを返す関数です。

フォーマット

XIMStyle DIMsInputStyle (*W*)

引数

Widget	<i>w</i> ;
--------	------------

戻り値

入力コンテキストの入力スタイルを指定する XIMStyle 型の値を返します。

説明

DIMsInputStyleによって、入力コンテキストの入力スタイルが取得されます。クライアントが XCreateIC()を呼び出す場合には、入力スタイルを指定して、IMS に送信する必要があります。IMSSL は、この入力スタイルを DIMs ウィジェットのデータとして保持します。

w 入力コンテキストに対応する DIMs ウィジェットを指定します。

DIMsClientWindow

DIMs ウィジェットに対応する入力コンテキストの XNClientWindow の XIC 値を返す関数です。

フォーマット

Window DIMsClientWindow (*W*)

引数

Widget	<i>w</i> ;
--------	------------

戻り値

入力コンテキストのクライアント・ウィンドウを指定する Window 型の値を返します。

説明

DIMsClientWindowは、入力コンテキストのクライアント・ウィンドウを取得します。クライアントが XCreateIC()を呼び出す場合には、クライアント・ウィンドウを指定する必要があります。IMSSL は、このクライアント・ウィンドウをウィジェット・データとして保持します。

w 入力コンテキストに対応する DIMs ウィジェットを指定します。

DIMsFocusWindow

DIMs ウィジェットに対応する入力コンテキストの XNFocusWindow の XIC 値を返す関数です。

フォーマット

Window DIMsFocusWindow (w)

引数

Widget	w;
--------	----

戻り値

入力コンテキストのフォーカス・ウィンドウを指定する Window 型の値を返します。

説明

DIMsFocusWindowは、入力コンテキストのフォーカス・ウィンドウを取得します。クライアントが、XCreateIC()またはXSetICValues()を呼び出す場合には、フォーカス・ウィンドウを指定する必要があります。IMSSL は、このウィンドウをウィジェット・データとして保持します。フォーカス・ウィンドウが指定されない場合は、クライアント・ウィンドウが返されます。

w 入力コンテキストに対応する DIMs ウィジェットを指定します。

DIMsUserData

入力コンテキストの固有データを返す関数です。

フォーマット

Opaque *DIMsUserData (w)

引数

Widget	w;
--------	----

戻り値

入力コンテキストの固有データを指すポインタを返します。データ型は任意です。

説明

DIMsUserDataは、入力コンテキストの固有データを取得します。このデータは、クライアントが XCreateIC() を呼び出し、コールバックが呼び出される場合に、IMS プログラマによって作成されます。コールバックについての詳細は、createICCb の項を参照してください。

w 入力コンテキストに対応する DIMs ウィジェットを指定します。

DIMsSetTriggerKeys

すべてのクライアントのトリガ・キーを変更する関数です。

フォーマット

```
void DIMsSetTriggerKeys (new_keys)
```

引数

DIMsTriggerKeys	<i>new_keys;</i>
-----------------	------------------

戻り値

戻り値はありません。

説明

DIMsSetTriggerKeysは、IMS のトリガ・オン/オフのためのキーを変更します。この呼び出しによって、すべての DIMs クラスのDIMsNtriggerKeysリソースが変更され、新しいトリガ・キーがすべてのクライアントに送信されます。イベント・フロー・モデルにおけるトリガ・キーについては、第 1.3.3 項を参照してください。

この呼び出しは、createICCb()、または*reason*引数として DimCR_CreateIC を設定するようなコールバックでは、クライアントと IMS の接続がまだ確立されていないため使用できないことに注意してください。このようなコールバックの詳細については、setAreaCb の項およびそれ以降のいくつかの項を参照してください。

<i>new_keys</i>	IMS の新しいトリガ・キーを指定します。DIMsTriggerKeysのデータ型の定義は第 1.4.8 項を参照してください。IMS プログラマは、このポインタを後で解放しなければなりません。
-----------------	---

getExtIMValuesCb

このコールバック関数は、拡張 (非標準) XIM の属性値を取得するために呼び出されます。

フォーマット

```
int getExtIMValuesCb (client_data, attributes, value_len, value)
```

引数

Opaque	<i>*client_data;</i>	
char	<i>*attributes;</i>	
short	<i>*value_len;</i>	(呼び出し側へ返す)
Opaque	<i>**value;</i>	(呼び出し側へ返す)

戻り値

このコールバックが正常終了した場合、IMSSL に 1 が返されます。このコールバックが異常終了した場合、IMSSL に 0 が返されます。

説明

getExtIMValues は、リソース DIMsNgetExtIMValuesCb によって指定されるコールバック関数です。DIMsNgetExtIMValuesCb については、第 1.4.9 項を参照してください。IMS は、拡張 (非標準) XIM のいくつかの属性を定義することができ、クライアントは、XGetIMValues を呼び出してこれらの属性の値を検索することができます。IMS に拡張 XIM 属性を追加する方法については、第 1.4.4 項を参照してください。

<i>client_data</i>	リソース DIMsNclientdata によって指定されるデータのポインタを指定します。詳細については、第 1.4.54 項を参照してください。
<i>attributes</i>	XIM 属性名を指定します。
<i>value_len</i>	属性値のサイズ (バイト単位) を呼び出し元へ返します。 <i>value</i> のデータ型が <code>string_t</code> ではない場合、このフィールドは任意です。第 1.4.4 項を参照してください。
<i>value</i>	呼び出し側に属性値のポインタを返します。IMS プログラムは、このデータにメモリを割り当てなければなりません。これは後に IMSSL によって解放されます。

createICCb

このコールバック関数は、入力コンテキストおよびそれに対応する DIMs ウィジェットが作成される場合に呼び出されます。

フォーマット

```
int createICCb (w, client_data, widget_data)
```

引数

Widget	<i>w</i> ;	
Opaque	<i>*client_data</i> ;	
Opaque	<i>**widget_data</i> ;	呼び出し側へ返す

戻り値

コールバックが正常終了した場合、IMSSL に 1 が返されます。コールバックが異常終了した場合、IMSSL に 0 が返されます。

説明

createICCbは、リソースDIMsNcreateICCbによって指定されるコールバック関数です。DIMsNcreateICCbについては、第 1.4.10 項を参照してください。この段階ではクライアントと IMS はまだ接続されていないため、このコールバックで次のルーチン呼び出すことはできません。

```
DIMsPreEditStart()
DIMsPreEditDone()
DIMsPreEditDraw()
DIMsPreEditCaret()
DIMsStatusStart()
DIMsStatusDone()
DIMsStatusDraw()
DIMsGeometryNegotiation()
DIMsSetEventsForward()
DIMsCommit()
```

DIMsSetTriggerKeys()

<i>w</i>	入力コンテキストに対応する DIMs ウィジェットを指定します。
<i>client_data</i>	リソースDIMsNclientdataによって指定されるデータのポインタを指定します。DIMsNclientdataについての詳細は、第 1.4.54 項を参照してください。
<i>widget_data</i>	ウィジェットに対応する入力コンテキスト固有データのポインタを呼び出し側に返します。このデータは、DIMsUserData(w)を呼び出して後で検索することができます。IMS プログラムは、このデータにメモリを割り当て、destroyICCbを使用して解放する必要があります。

destroyICCb

このコールバック関数は、入力コンテキストおよびそれに対応する DIMs ウィジェットを破棄する場合に呼び出されます。

フォーマット

```
int destroyICCb (w, client_data)
```

引数

Widget	<i>w</i> ;
Opaque	<i>*client_data</i> ;

戻り値

コールバックが正常終了した場合、IMSSL に 1 が返されます。コールバックが異常終了した場合、IMSSL に 0 が返されます。

説明

destroyICCbは、リソースDIMsNdestroyICCbによって指定されるコールバック関数です。DIMsNdestroyICCbについては、第 1.4.11 項を参照してください。

<i>w</i>	入力コンテキストに対応する DIMs ウィジェットを指定します。
<i>client_data</i>	リソースDIMsNclientdataによって指定されるデータのポインタを指定します。DIMsNclientdataについては、第 1.4.54 項を参照してください。

setExtICValuesCb

このコールバック関数は、拡張 (非標準) XIC 属性の値を設定する場合に呼び出されます。

フォーマット

```
int setExtICValuesCb (w, client_data, attributes, value_len, value, attribute_flag, reason)
```

引数

Widget	<i>w</i> ;
Opaque	<i>*client_data</i> ;
char	<i>*attributes</i> ;
short	<i>value_len</i> ;
Opaque	<i>*value</i> ;
DIMsAttributesType	<i>attribute_flag</i> ;
int	<i>reason</i> ;

戻り値

コールバックが正常終了した場合、IMSSL に 1 が返されます。コールバックが異常終了した場合、IMSSL に 0 が返されます。

説明

setExtICValuesCbは、リソースDIMsNsetExtICValuesCbによって指定されるコールバック関数です。DIMsNsetExtICValuesCbについては、第 1.4.12 項を参照してください。IMS はいくつかの拡張 (非標準) XIC 属性を定義でき、クライアントは、XSetICValues を呼び出してこれらの属性の値を設定することができます。IMS に拡張 XIC 属性を追加する方法については、第 1.4.5 項を参照してください。

<i>w</i>	入力コンテキストに対応する DIMs ウィジェットを指定します。
<i>client_data</i>	リソースDIMsNclientdataによって指定されるデータのポインタを指定します。DIMsNclientdataについては、第 1.4.54 項を参照してください。
<i>attributes</i>	拡張 XIC 値の名称を指定します。
<i>value_len</i>	<i>value</i> フィールドのサイズ (バイト単位) を指定します。この属性のデータ型が <code>string_t</code> でない場合、このフィールドは無視できます。第 1.4.4 項を参照してください。

<i>value</i>	属性値のポインタを指定します。属性値のデータ型は任意ですが、IMS プログラマは、実際のデータ型へのポインタを属性に従ってキャストする必要があります。
<i>attribute_flag</i>	<p>XIC の属性の型を指定します。DIMsAttributeType のデータ型は次のとおりです。</p> <pre>typedef enum { GenericAttributesType, PreeditAttributesType, StatusAttributesType }DIMsAttributesType;</pre> <p>GenericAttributesType は、XIC の値が PreeditAttribute および StatusAttribute のどちらにも属さないことを示します。</p> <p>PreeditAttributesType は、XIC の値が PreeditAttribute に属することを示します。</p> <p>StatusAttributesType は、XIC の値が StatusAttribute に属することを示します。</p>
<i>reason</i>	<p>コールバックの理由を指定します。DimCR_CreateIC または DimCR_SetICValues のいずれかを指定します。DimCR_CreateIC はクライアントの XCreateIC() によってコールバックが開始されることを意味し、DimCR_SetICValues はクライアントの XSetICValues() によってコールバックが開始されることを意味します。</p>

getExtICValuesCb

このコールバック関数は、拡張 (非標準) XIC 属性の値を取得するために呼び出されます。

フォーマット

```
int getExtICValuesCb (w, client_data, attribute, value_len, value, attr_flag)
```

引数

Widget	<i>w</i> ;	
Opaque	<i>*client_data</i> ;	
char	<i>*attributes</i> ;	
short	<i>*value_len</i> ;	(呼び出し側へ返す)
Opaque	<i>**value</i> ;	(呼び出し側へ返す)
DIMsAttributesType	<i>attr_flag</i> ;	

戻り値

コールバックが正常終了した場合、IMSSL に 1 が返されます。コールバックが異常終了した場合、IMSSL に 0 が返されます。

説明

getExtICValuesCbは、リソースDIMsNgetExtICValuesCbによって指定されるコールバック関数です。DIMsNgetExtICValuesCbについては第 1.4.13 項を参照してください。IMS はいくつかの拡張 (非標準) XIC の属性を定義することができ、クライアントはXGetICValues()を呼び出してこれらの属性の値を検索することができます。IMS に拡張 XIC 属性を追加する方法については、第 1.4.5 項を参照してください。

<i>w</i>	入力コンテキストに対応する DIMs ウィジェットを指定します。
<i>client_data</i>	リソースDIMsNclientdataによって指定されるデータのポインタを指定します。DIMsNclientdataについては第 1.4.54 項を参照してください。
<i>attributes</i>	拡張 XIC 値の名称を指定します。
<i>value_len</i>	<i>value</i> フィールドのサイズ (バイト単位) を呼び出し側に返します。この属性のデータ型が <i>string_t</i> でない場合、このフィールドは無視できます。第 1.4.4 項を参照してください。

<i>value</i>	属性値のポインタを呼び出し側に返します。IMS プログラムは、このデータにメモリを割り当てなければなりません。このメモリは、IMSSL によって後で解放されます。
<i>attr_flag</i>	属性の型を指定します。データ型の定義については setExtICValuesCb の項を参照してください。

setICFocusCb

このコールバック関数は、入力コンテキストがフォーカスを取得した場合に呼び出されます。

フォーマット

```
int setICFocusCb (w, client_data)
```

引数

Widget	<i>w</i> ;
Opaque	<i>*client_data</i> ;

戻り値

コールバックが正常終了した場合、IMSSL に 1 が返されます。コールバックが異常終了した場合、IMSSL に 0 が返されます。

説明

setICFocusCbは、リソースDIMsNsetICFocusCbによって指定されるコールバック関数です。DIMsNsetICFocusCbについては、第 1.4.14 項を参照してください。

<i>w</i>	入力コンテキストに対応する DIMs ウィジェットを指定します。
<i>client_data</i>	リソースDIMsNclientdataによって指定されるデータのポインタを指定します。DIMsNclientdataについては第 1.4.54 項を参照してください。

unsetICFocusCb

このコールバック関数は、入力コンテキストがフォーカスを失った場合に呼び出されます。

フォーマット

```
int unsetICFocusCb (w, client_data)
```

引数

Widget	<i>w</i> ;
Opaque	<i>*client_data</i> ;

戻り値

コールバックが正常終了した場合、IMSSL に 1 が返されます。コールバックが異常終了した場合、IMSSL に 0 が返されます。

説明

setICFocusCbは、リソースDIMsNunsetICFocusCbによって指定されるコールバック関数です。DIMsNunsetICFocusCbについては、第 1.4.15 項を参照してください。

<i>w</i>	入力コンテキストに対応する DIMs ウィジェットを指定します。
<i>client_data</i>	リソースDIMsNclientdataによって指定されるデータのポインタを指定します。DIMsNclientdataについては第 1.4.54 項を参照してください。

resetlCCb

このコールバック関数は、入力コンテキストが再設定される場合に呼び出されます。

フォーマット

```
int resetlCCb (w, client_data, string, string_length)
```

引数

Widget	<i>w</i> ;	
Opaque	<i>*client_data</i> ;	
void	<i>**string</i> ;	(呼び出し側へ返す)
int	<i>*string_length</i> ;	(呼び出し側へ返す)

戻り値

コールバックが正常終了した場合、IMSSL に 1 が返されます。コールバックが異常終了した場合、IMSSL に 0 が返されます。

説明

resetlCCbは、リソースDIMsNresetlCCbによって指定されるコールバック関数です。DIMsNresetlCCbについては第 1.4.16 項を参照してください。現在の前編集セッションは、このコールバックが実行されてから終了します。

<i>w</i>	入力コンテキストに対応する DIMs ウィジェットを指定します。
<i>client_data</i>	リソースDIMsNclientdataによって指定されるデータのポインタを指定します。DIMsNclientdataについては、第 1.4.54 項を参照してください。
<i>string</i>	DIMsEncoding(w) によって与えられたエンコーディングで現在の前編集文字列を返します。IMS プログラマは、このデータにメモリを割り当てなければなりません。このメモリは IMSSL によって解放されます。
<i>string_length</i>	<i>string</i> のバイト数を返します。

processEventCb

このコールバック関数は、入力コンテキストが IMS にイベントを転送する場合に呼び出されます。

フォーマット

```
int processEventCb (w, client_data, event, send_back)
```

引数

Widget	<i>w</i> ;	
Opaque	<i>*client_data</i> ;	
XEvent	<i>*event</i> ;	
Boolean	<i>*send_back</i> ;	(呼び出し側へ返す)

戻り値

コールバックが正常終了した場合、IMSSL に 1 が返されます。コールバックが異常終了した場合、IMSSL に 0 が返されます。

説明

processEventCb は、リソース DIMsNprocessEventCb によって指定されるコールバック関数です。DIMsNprocessEventCb については、第 1.4.17 項を参照してください。

クライアントが XFilterEvent() を呼び出すと、XNFilterEvent の値を取得する XGetICValues 関数で IMS によって選択されたイベントが IMS に転送され、このコールバックを呼び出します (IMS に必要なイベントの選択方法については getFilterEventsCb の項を参照してください)。IMS によっては、これらのイベントは、文字列の確定、前編集セッションの開始または終了、あるいはその他の状況を引き起こします。

ただし、入力スタイルが on-the-spot の場合、1 つの KeyPress イベントで DIMsCommit、DIMsPreEditDone、および DIMsPreEditStart を同時に呼び出すことはできません。同時に呼び出すと、クライアントは、確定文字列およびコールバックを正しい順序で得ることができません。このため、KeyPress イベントで DIMsPreEditDone と DIMsCommit を呼び出したあとで、対応する KeyRelease イ

ベントで DIMsPreEditStart を呼び出して次の前編集セッションを開始してください。

<i>w</i>	入力コンテキストに対応する DIMs ウィジェットを指定します。
<i>client_data</i>	リソースDIMsNclientdataによって指定されるデータのポインタを指定します。DIMsNclientdataについては、第 1.4.54 項を参照してください。
<i>event</i>	クライアントから転送するイベントを指定します。
<i>send_back</i>	イベントをクライアントに返送するかどうかを指定します。True が返されると XFilterEvent はクライアントに False を返し、False の場合、イベントはクライアントによって破棄されます。

setAreaCb

このコールバック関数は、XIC 属性 XNArea の値を設定するために呼び出されます。

フォーマット

```
int setAreaCb (w, client_data, reason, attribut_flag)
```

引数

Widget	<i>w</i> ;
Opaque	<i>*client_data</i> ;
int	<i>reason</i> ;
XRectangle	<i>*value</i> ;
DIMsAttributesType	<i>attribute_flag</i> ;

戻り値

このコールバックが正常終了した場合、IMSSL に 1 が返されます。このコールバックが異常終了した場合、IMSSL に 0 が返されます。

説明

setAreaCbは、リソースDIMsNsetAreaCbによって指定されるコールバック関数です。DIMsNsetAreaCbについては、第 1.4.18 項を参照してください。

<i>w</i>	入力コンテキストに対応する DIMs ウィジェットを指定します。
<i>client_data</i>	リソースDIMsNclientdataによって指定されるデータへのポインタを指定します。DIMsNclientdataについては、第 1.4.54 項を参照してください。
<i>reason</i>	コールバックの理由を指定します。DimCR_CreateICまたはDimCR_SetICValuesのいずれかを指定します。DimCR_CreateICはクライアントの XCreateIC()によってコールバックが開始されることを意味し、DimCR_SetICValuesはクライアントの XSetICValues()によってコールバックが開始されることを意味します。
<i>value</i>	XRectangle 型の属性値へのポインタを指定します。

setAreaCb

attribute_flag

PreeditAttribute または StatusAttribute のどちらに属するか、XIC 属性の型を指定します。DIMsAttributesTypeのデータ型については、setExtICValuesCb の項を参照してください。

setAreaNeededCb

このコールバック関数は、XIC 属性 XNAreaNeeded の値を設定するために呼び出されます。

フォーマット

```
int setAreaNeededCb (w, client_data, reason, value, attribute_flag)
```

引数

Widget	<i>w</i> ;
Opaque	<i>*client_data</i> ;
int	<i>reason</i> ;
XRectangle	<i>*value</i> ;
DIMsAttributesType	<i>attribute_flag</i> ;

戻り値

このコールバックが正常終了した場合、IMSSL に 1 が返されます。このコールバックが異常終了した場合、IMSSL に 0 が返されます。

説明

setAreaNeededCbは、リソースDIMsNsetAreaNeededCbによって指定されるコールバック関数です。DIMsNsetAreaNeededCbについては、第 1.4.19 項を参照してください。

<i>w</i>	入力コンテキストに対応する DIMs ウィジェットを指定します。
<i>client_data</i>	リソースDIMsNclientdataによって指定されるデータへのポインタを指定します。DIMsNclientdataについては、第 1.4.54 項を参照してください。
<i>reason</i>	コールバックの理由を指定します。DimCR_CreateICまたはDimCR_SetICValuesのいずれかを指定します。DimCR_CreateICはクライアントの XCreateIC()によってコールバックが開始されることを意味し、DimCR_SetICValuesはクライアントの XSetICValues()によってコールバックが開始されることを意味します。
<i>value</i>	XRectangle 型の属性値へのポインタを指定します。

setAreaNeededCb

attribute_flag

PreeditAttribute または StatusAttribute のどちらに属するか、XIC 属性の型を指定します。DIMsAttributesTypeのデータ型については、setExtICValuesCb の項を参照してください。

setSpotLocationCb

このコールバック関数は、XIC 属性 XNSpotLocation の値を設定するために呼び出されます。

フォーマット

```
int setSpotLocationCb (w, client_data, reason, value, attr_flag)
```

引数

Widget	<i>w</i> ;
Opaque	<i>*client_data</i> ;
int	<i>reason</i> ;
XPoint	<i>*value</i> ;
DIMsAttributesType	<i>attr_flag</i> ;

戻り値

このコールバックが正常終了した場合、IMSSL に 1 が返されます。このコールバックが異常終了した場合、IMSSL に 0 が返されます。

説明

setSpotLocationCbは、リソースDIMsNsetSpotLocationCbによって指定されるコールバック関数です。DIMsNsetSpotLocationCbについては、第 1.4.20 項を参照してください。

<i>w</i>	入力コンテキストに対応する DIMs ウィジェットを指定します。
<i>client_data</i>	リソースDIMsNclientdataによって指定されるデータへのポインタを指定します。DIMsNclientdataについての詳細は、第 1.4.54 項を参照してください。
<i>reason</i>	コールバックの理由を指定します。DimCR_CreateICまたはDimCR_SetICValuesのいずれかを指定します。DimCR_CreateICはクライアントのXCreateIC()によってコールバックが開始されることを意味し、DimCR_SetICValuesはクライアントのXSetICValues()によってコールバックが開始されることを意味します。
<i>value</i>	XPoint 型の属性値へのポインタを指定します。

setSpotLocationCb

attri_flag

PreeditAttribute または StatusAttribute のどちらに属するか、XIC 属性の型を指定します。DIMsAttributesTypeのデータ型については、setExtICValuesCb の項を参照してください。

setColormapCb

このコールバック関数は、XIC 属性 XNColormap の値を設定するために呼び出されます。

フォーマット

```
int setColormapCb (w, client_data, reason, value, attribute_flag)
```

引数

Widget	<i>w</i> ;
Opaque	<i>*client_data</i> ;
int	<i>reason</i> ;
Colormap	<i>*value</i> ;
DIMsAttributesType	<i>attribute_flag</i> ;

戻り値

このコールバックが正常終了した場合、IMSSL に 1 が返されます。このコールバックが異常終了した場合、IMSSL に 0 が返されます。

説明

setColormapCbは、リソースDIMsNsetColormapCbによって指定されるコールバック関数です。DIMsNsetColormapCbについては、第 1.4.21 項を参照してください。

<i>w</i>	入力コンテキストに対応する DIMs ウィジェットを指定します。
<i>client_data</i>	リソースDIMsNclientdataによって指定されるデータへのポインタを指定します。DIMsNclientdataについての詳細は、第 1.4.54 項を参照してください。
<i>reason</i>	コールバックの理由を指定します。DimCR_CreateICまたはDimCR_SetICValuesのいずれかを指定します。DimCR_CreateICはクライアントの XCreateIC()によってコールバックが開始されることを意味し、DimCR_SetICValuesはクライアントの XSetICValues()によってコールバックが開始されることを意味します。
<i>value</i>	Colormap 型の属性値へのポインタを指定します。

setColormapCb

attribute_flag

PreeditAttribute または StatusAttribute のどちらに属するか、XIC 属性の型を指定します。DIMsAttributesTypeのデータ型については、setExtICValuesCb の項を参照してください。

setStdColormapCb

このコールバック関数は、XIC 属性 XNStdColormap の値を設定するために呼び出されます。

フォーマット

```
int setStdColormapCb (w, client_data, reason, value, attr_flag)
```

引数

Widget	<i>w</i> ;
Opaque	<i>*client_data</i> ;
int	<i>reason</i> ;
Atom	<i>*value</i> ;
DIMsAttributesType	<i>attr_flag</i> ;

戻り値

このコールバックが正常終了した場合、IMSSL に 1 が返されます。このコールバックが異常終了した場合、IMSSL に 0 が返されます。

説明

setStdColormapCbは、リソースDIMsNsetStdColormapCbによって指定されるコールバック関数です。DIMsNsetStdColormapCbについては、第 1.4.22 項を参照してください。

<i>w</i>	入力コンテキストに対応する DIMs ウィジェットを指定します。
<i>client_data</i>	リソースDIMsNclientdataによって指定されるデータへのポインタを指定します。DIMsNclientdataについての詳細は、第 1.4.54 項を参照してください。
<i>reason</i>	コールバックの理由を指定します。DimCR_CreateICまたはDimCR_SetICValuesのいずれかを指定します。DimCR_CreateICはクライアントのXCreateIC()によってコールバックが開始されることを意味し、DimCR_SetICValuesはクライアントのXSetICValues()によってコールバックが開始されることを意味します。
<i>value</i>	Atom 型の属性値へのポインタを指定します。

setStdColormapCb

attri_flag

PreeditAttribute または StatusAttribute のどちらに属するか、XIC 属性の型を指定します。DIMsAttributesTypeのデータ型については、setExtICValuesCb の項を参照してください。

setForegroundCb

このコールバック関数は、XIC 属性 XNForeground の値を設定するために呼び出されます。

フォーマット

```
int setForegroundCb (w, client_data, reason, value, attribute_flag)
```

引数

Widget	<i>w</i> ;
Opaque	<i>*client_data</i> ;
int	<i>reason</i> ;
unsigned long	<i>*value</i> ;
DIMsAttributesType	<i>attribute_flag</i> ;

戻り値

このコールバックが正常終了した場合、IMSSL に 1 が返されます。このコールバックが異常終了した場合、IMSSL に 0 が返されます。

説明

setForegroundCbは、リソースDIMsNsetForegroundCbによって指定されるコールバック関数です。DIMsNsetForegroundCbについては、第 1.4.23 項を参照してください。

<i>w</i>	入力コンテキストに対応する DIMs ウィジェットを指定します。
<i>client_data</i>	リソースDIMsNclientdataによって指定されるデータへのポインタを指定します。DIMsNclientdataについての詳細は、第 1.4.54 項を参照してください。
<i>reason</i>	コールバックの理由を指定します。DimCR_CreateICまたはDimCR_SetICValuesのいずれかを指定します。DimCR_CreateICはクライアントのXCreateIC()によってコールバックが開始されることを意味し、DimCR_SetICValuesはクライアントのXSetICValues()によってコールバックが開始されることを意味します。
<i>value</i>	符号なし long 型の属性値へのポインタを指定します。

setForegroundCb

attribute_flag

PreeditAttribute または StatusAttribute のどちらに属するか、XIC 属性の型を指定します。DIMsAttributesTypeのデータ型については、setExtICValuesCb の項を参照してください。

setBackgroundCb

このコールバック関数は、XIC 属性 XNBackground の値を設定するために呼び出されます。

フォーマット

```
int setBackgroundCb (w, client_data, reason, value, attribute_flag)
```

引数

Widget	<i>w</i> ;
Opaque	<i>*client_data</i> ;
int	<i>reason</i> ;
unsigned long	<i>*value</i> ;
DIMsAttributesType	<i>attribute_flag</i> ;

戻り値

このコールバックが正常終了した場合、IMSSL に 1 が返されます。このコールバックが異常終了した場合、IMSSL に 0 が返されます。

説明

setBackgroundCbは、リソースDIMsNsetBackgroundCbによって指定されるコールバック関数です。DIMsNsetBackgroundCbについては、第 1.4.24 項を参照してください。

<i>w</i>	入力コンテキストに対応する DIMs ウィジェットを指定します。
<i>client_data</i>	リソースDIMsNclientdataによって指定されるデータへのポインタを指定します。DIMsNclientdataについての詳細は、第 1.4.54 項を参照してください。
<i>reason</i>	コールバックの理由を指定します。DimCR_CreateICまたはDimCR_SetICValuesのいずれかを指定します。DimCR_CreateICはクライアントのXCreateIC()によってコールバックが開始されることを意味し、DimCR_SetICValuesはクライアントのXSetICValues()によってコールバックが開始されることを意味します。
<i>value</i>	符号なし long 型の属性値へのポインタを指定します。

setBackgroundCb

attribute_flag

PreeditAttribute または StatusAttribute のどちらに属するか、XIC 属性の型を指定します。DIMsAttributesTypeのデータ型については、setExtICValuesCb の項を参照してください。

setBgPixmapCb

このコールバック関数は、XIC 属性 XNBackgroundPixmap の値を設定するために呼び出されます。

フォーマット

```
int setBgPixmapCb (w, client_data, reason, value, attribute_flag)
```

引数

Widget	<i>w</i> ;
Opaque	<i>*client_data</i> ;
int	<i>reason</i> ;
Pixmap	<i>*value</i> ;
DIMsAttributesType	<i>attribute_flag</i> ;

戻り値

このコールバックが正常終了した場合、IMSSL に 1 が返されます。このコールバックが異常終了した場合、IMSSL に 0 が返されます。

説明

setBgPixmapCbは、リソースDIMsNsetBgPixmapCbによって指定されるコールバック関数です。DIMsNsetBgPixmapCbについては、第 1.4.25 項を参照してください。

<i>w</i>	入力コンテキストに対応する DIMs ウィジェットを指定します。
<i>client_data</i>	リソースDIMsNclientdataによって指定されるデータへのポインタを指定します。DIMsNclientdataについての詳細は、第 1.4.54 項を参照してください。
<i>reason</i>	コールバックの理由を指定します。DimCR_CreateICまたはDimCR_SetICValuesのいずれかを指定します。DimCR_CreateICはクライアントの XCreateIC()によってコールバックが開始されることを意味し、DimCR_SetICValuesはクライアントの XSetICValues()によってコールバックが開始されることを意味します。
<i>value</i>	Pixmap 型の属性値へのポインタを指定します。

setBgPixmapCb

attribute_flag

PreeditAttribute または StatusAttribute のどちらに属するか、XIC 属性の型を指定します。DIMsAttributesTypeのデータ型については、setExtICValuesCb の項を参照してください。

setFontSetCb

このコールバック関数は、XIC 属性 XNFontSet の値を設定するために呼び出されます。

フォーマット

```
int setFontSetCb (w, client_data, reason, value, attribute_flag)
```

引数

Widget	<i>w</i> ;
Opaque	<i>*client_data</i> ;
int	<i>reason</i> ;
XFontSet	<i>*value</i> ;
DIMsAttributesType	<i>attribute_flag</i> ;

戻り値

このコールバックが正常終了した場合、IMSSL に 1 が返されます。このコールバックが異常終了した場合、IMSSL に 0 が返されます。

説明

setFontSetCbは、リソースDIMsNsetFontSetCbによって指定されるコールバック関数です。DIMsNsetFontSetCbについては、第 1.4.26 項を参照してください。

<i>w</i>	入力コンテキストに対応する DIMs ウィジェットを指定します。
<i>client_data</i>	リソースDIMsNclientdataによって指定されるデータへのポインタを指定します。DIMsNclientdataについての詳細は、第 1.4.54 項を参照してください。
<i>reason</i>	コールバックの理由を指定します。DimCR_CreateICまたはDimCR_SetICValuesのいずれかを指定します。DimCR_CreateICはクライアントの XCreateIC()によってコールバックが開始されることを意味し、DimCR_SetICValuesはクライアントの XSetICValues()によってコールバックが開始されることを意味します。
<i>value</i>	XFontSet 型の属性値へのポインタを指定します。

setFontSetCb

attribute_flag

PreeditAttribute または StatusAttribute のどちらに属するか、XIC 属性の型を指定します。DIMsAttributesTypeのデータ型については、setExtICValuesCb の項を参照してください。

setLineSpaceCb

このコールバック関数は、XIC 属性 XNLineSpace の値を設定するために呼び出されます。

フォーマット

```
int setLineSpaceCb (w, client_data, reason, value, attribute_flag)
```

引数

Widget	<i>w</i> ;
Opaque	<i>*client_data</i> ;
int	<i>reason</i> ;
int	<i>*value</i> ;
DIMsAttributesType	<i>attribute_flag</i> ;

戻り値

このコールバックが正常終了した場合、IMSSL に 1 が返されます。このコールバックが異常終了した場合、IMSSL に 0 が返されます。

説明

setLineSpaceCbは、リソースDIMsNsetLineSpaceCbによって指定されるコールバック関数です。DIMsNsetLineSpaceCbについては、第 1.4.27 項を参照してください。

<i>w</i>	入力コンテキストに対応する DIMs ウィジェットを指定します。
<i>client_data</i>	リソースDIMsNclientdataによって指定されるデータへのポインタを指定します。DIMsNclientdataについての詳細は、第 1.4.54 項を参照してください。
<i>reason</i>	コールバックの理由を指定します。DimCR_CreateICまたはDimCR_SetICValuesのいずれかを指定します。DimCR_CreateICはクライアントの XCreateIC()によってコールバックが開始されることを意味し、DimCR_SetICValuesはクライアントの XSetICValues()によってコールバックが開始されることを意味します。
<i>value</i>	integer 型の属性値へのポインタを指定します。

setLineSpaceCb

attribute_flag

PreeditAttribute または StatusAttribute のどちらに属するか、XIC 属性の型を指定します。DIMsAttributesTypeのデータ型については、setExtICValuesCb の項を参照してください。

setCursorCb

このコールバック関数は、XIC 属性 XNCursor の値を設定するために呼び出されます。

フォーマット

```
int setLineSpaceCb (w, client_data, reason, value, attribute_flag)
```

引数

Widget	<i>w</i> ;
Opaque	<i>*client_data</i> ;
int	<i>reason</i> ;
Cursor	<i>*value</i> ;
DIMsAttributesType	<i>attribute_flag</i> ;

戻り値

このコールバックが正常終了した場合、IMSSL に 1 が返されます。このコールバックが異常終了した場合、IMSSL に 0 が返されます。

説明

setCursorCbは、リソースDIMsNsetCursorCbによって指定されるコールバック関数です。DIMsNsetCursorCbについては、第 1.4.28 項を参照してください。

<i>w</i>	入力コンテキストに対応する DIMs ウィジェットを指定します。
<i>client_data</i>	リソースDIMsNclientdataによって指定されるデータへのポインタを指定します。DIMsNclientdataについての詳細は、第 1.4.54 項を参照してください。
<i>reason</i>	コールバックの理由を指定します。DimCR_CreateICまたはDimCR_SetICValuesのいずれかを指定します。DimCR_CreateICはクライアントの XCreateIC()によってコールバックが開始されることを意味し、DimCR_SetICValuesはクライアントの XSetICValues()によってコールバックが開始されることを意味します。
<i>value</i>	Cursor 型の属性値へのポインタを指定します。

setCursorCb

attribute_flag

PreeditAttribute または StatusAttribute のどちらに属するか、XIC 属性の型を指定します。DIMsAttributesTypeのデータ型については、setExtICValuesCb の項を参照してください。

getFilterEventsCb

このコールバック関数は、XIC 属性 XNFilterEvents の値を得るために呼び出されます。

フォーマット

```
int getFilterEventsCb (w, client_data, value)
```

引数

Widget	<i>w</i> ;	
Opaque	<i>*client_data</i> ;	
unsigned long	<i>**value</i> ;	(呼び出し側へ返す)

戻り値

このコールバックが正常終了した場合、IMSSL に 1 が返されます。このコールバックが異常終了した場合、IMSSL に 0 が返されます。

説明

getFilterEventsCb は、リソース DIMsNgetFilterEventsCb で指定されるコールバック関数です。DIMsNgetFilterEventsCb については、第 1.4.29 項を参照してください。

このコールバック・リソースが設定されていない場合、IMSSL は省略時のイベント・マスクとして KeyPressMask | KeyReleaseMask をクライアントへ返します。この値が返された場合は、クライアントから IMS へのイベント・フローが増えることによりネットワークのアクティビティが増加し、IMS の性能に直接影響するため注意が必要です。必要ないイベント・マスクは削除するようにしてください。たとえば、DIMs ウィジェットの入力スタイルが on-the-spot でない場合は、KeyReleaseMask の使用を削除することができます。

<i>w</i>	入力コンテキストに対応する DIMs ウィジェットを指定します。
<i>client_data</i>	DIMsNclientdata リソースで指定されるデータへのポインタを指定します。DIMsNclientdata についての詳細は、第 1.4.54 項を参照してください。

getFilterEventsCb

value

呼び出し側に、符号なし long 型の属性値へのポインタを返します。
IMS プログラムは、このデータにメモリを割り当てる必要があります。
このメモリ割り当ては IMSSL によって後で解放されます。

getAreaCb

このコールバック関数は、XIC 属性 XNArea の値を得るために呼び出されます。

フォーマット

```
int getAreaCb (w, client_data, value, attribute_flag)
```

引数

Widget	<i>w</i> ;	
Opaque	<i>*client_data</i> ;	
XRectangle	<i>**value</i> ;	(呼び出し側へ返す)
DIMsAttributesType	<i>attribute_flag</i> ;	

戻り値

このコールバックが正常終了した場合、IMSSL に 1 が返されます。このコールバックが異常終了した場合、IMSSL に 0 が返されます。

説明

getAreaCbは、リソースDIMsNgetAreaCbで指定されるコールバック関数です。DIMsNgetAreaCbについては、第 1.4.30 項を参照してください。

<i>w</i>	入力コンテキストに対応する DIMs ウィジェットを指定します。
<i>client_data</i>	DIMsNclientdataリソースで指定されるデータへのポインタを指定します。DIMsNclientdataについての詳細は、第 1.4.54 項を参照してください。
<i>value</i>	呼び出し側に、XRectangle 型の属性値へのポインタを返します。IMS プログラマは、このデータにメモリを割り当てる必要があります。このメモリ割り当ては IMSSL によって後で解放されます。
<i>attribute_flag</i>	PreeditAttribute または StatusAttribute のどちらに属するか、XIC 属性の型を指定します。DIMsAttributesTypeのデータ型については、setExtICValuesCb の項を参照してください。

getAreaNeededCb

このコールバック関数は、XIC 属性 XNAreaNeeded の値を得るために呼び出されます。

フォーマット

```
int getAreaNeededCb (w, client_data, value, attribute_flag)
```

引数

Widget	<i>w</i> ;	
Opaque	<i>*client_data</i> ;	
XRectangle	<i>**value</i> ;	(呼び出し側へ返す)
DIMsAttributesType	<i>attribute_flag</i> ;	

戻り値

このコールバックが正常終了した場合、IMSSL に 1 が返されます。このコールバックが異常終了した場合、IMSSL に 0 が返されます。

説明

getAreaNeededCbは、リソースDIMsNgetAreaNeededCbで指定されるコールバック関数です。DIMsNgetAreaNeededCbについては、第 1.4.31 項を参照してください。

<i>w</i>	入力コンテキストに対応する DIMs ウィジェットを指定します。
<i>client_data</i>	DIMsNclientdataリソースで指定されるデータへのポインタを指定します。DIMsNclientdataについての詳細は、第 1.4.54 項を参照してください。
<i>value</i>	呼び出し側に、XRectangle 型の属性値へのポインタを返します。IMS プログラマは、このデータにメモリを割り当てる必要があります。このメモリ割り当ては IMSSL によって後で解放されます。
<i>attribute_flag</i>	PreeditAttribute または StatusAttribute のどちらに属するか、XIC 属性の型を指定します。DIMsAttributesTypeのデータ型については、setExtICValuesCb の項を参照してください。

getSpotLocationCb

このコールバック関数は、XIC 属性 XNSpotLocation の値を得るために呼び出されます。

フォーマット

```
int getSpotLocationCb (w, client_data, value, attribute_flag)
```

引数

Widget	<i>w</i> ;	
Opaque	<i>*client_data</i> ;	
XPoint	<i>**value</i> ;	(呼び出し側へ返す)
DIMsAttributesType	<i>attribute_flag</i> ;	

戻り値

このコールバックが正常終了した場合、IMSSL に 1 が返されます。このコールバックが異常終了した場合、IMSSL に 0 が返されます。

説明

getSpotLocationCbは、リソースDIMsNgetSpotLocationCbリソースによって指定されるコールバック関数です。DIMsNgetSpotLocationCbについては、第 1.4.32 項を参照してください。

<i>w</i>	入力コンテキストに対応する DIMs ウィジェットを指定します。
<i>client_data</i>	DIMsNclientdataリソースで指定されるデータへのポインタを指定します。DIMsNclientdataについての詳細は、第 1.4.54 項を参照してください。
<i>value</i>	呼び出し側に、XPoint 型の属性値へのポインタを返します。IMS プログラマは、このデータにメモリを割り当てる必要があります。このメモリ割り当てはIMSSLによって後で解放されます。
<i>attribute_flag</i>	PreeditAttribute または StatusAttribute のどちらに属するか、XIC 属性の型を指定します。DIMsAttributesTypeのデータ型については、setExtICValuesCb の項を参照してください。

getColormapCb

このコールバック関数は、XIC 属性 XNColormap の値を得るために呼び出されます。

フォーマット

```
int getColormapCb (w, client_data, value, attribute_flag)
```

引数

Widget	<i>w</i> ;	
Opaque	<i>*client_data</i> ;	
Colormap	<i>**value</i> ;	(呼び出し側へ返す)
DIMsAttributesType	<i>attribute_flag</i> ;	

戻り値

このコールバックが正常終了した場合、IMSSL に 1 が返されます。このコールバックが異常終了した場合、IMSSL に 0 が返されます。

説明

getColormapCb は、リソース DIMsNgetColormapCb で指定されるコールバック関数です。DIMsNgetColormapCb については、第 1.4.33 項を参照してください。

<i>w</i>	入力コンテキストに対応する DIMs ウィジェットを指定します。
<i>client_data</i>	リソース DIMsNclientdata によって指定されるデータへのポインタを指定します。DIMsNclientdata についての詳細は、第 1.4.54 項を参照してください。
<i>value</i>	呼び出し側に、Colormap 型の属性値へのポインタを返します。IMS プログラマは、このデータにメモリを割り当てる必要があります。このメモリ割り当ては IMSSL によって後で解放されます。
<i>attribute_flag</i>	PreeditAttribute または StatusAttribute のどちらに属するか、XIC 属性の型を指定します。DIMsAttributesType のデータ型については、setExtICValuesCb の項を参照してください。

getStdColormapCb

このコールバック関数は、XIC 属性 XNStdColormap の値を得るために呼び出されます。

フォーマット

```
int getStdColormapCb (w, client_data, value, attribute_flag)
```

引数

Widget	<i>w</i> ;	
Opaque	<i>*client_data</i> ;	
Atom	<i>**value</i> ;	(呼び出し側へ返す)
DIMsAttributesType	<i>attribute_flag</i> ;	

戻り値

このコールバックが正常終了した場合、IMSSL に 1 が返されます。このコールバックが異常終了した場合、IMSSL に 0 が返されます。

説明

getStdColormapCb は、リソース DIMsNgetStdColormapCb によって指定されるコールバック関数です。DIMsNgetStdColormapCb については、第 1.4.34 項を参照してください。

<i>w</i>	入力コンテキストに対応する DIMs ウィジェットを指定します。
<i>client_data</i>	DIMsNclientdata リソースで指定されるデータへのポインタを指定します。DIMsNclientdata についての詳細は、第 1.4.54 項を参照してください。
<i>value</i>	呼び出し側に、Atom 型の属性値へのポインタを返します。IMS プログラマは、このデータにメモリを割り当てる必要があります。このメモリ割り当ては IMSSL によって後で解放されます。
<i>attribute_flag</i>	PreeditAttribute または StatusAttribute のどちらに属するか、XIC 属性の型を指定します。DIMsAttributesType のデータ型については、setExtICValuesCb の項を参照してください。

getForegroundCb

このコールバック関数は、XIC 属性 XNForeground の値を得るために呼び出されます。

フォーマット

```
int getForegroundCb (w, client_data, value, attribute_flag)
```

引数

Widget	<i>w</i> ;	
Opaque	<i>*client_data</i> ;	
unsigned long	<i>**value</i> ;	(呼び出し側へ返す)
DIMsAttributesType	<i>attribute_flag</i> ;	

戻り値

このコールバックが正常終了した場合、IMSSL に 1 が返されます。このコールバックが異常終了した場合、IMSSL に 0 が返されます。

説明

getForegroundCbは、リソースDIMsNgetForegroundCbで指定されるコールバック関数です。DIMsNgetForegroundCbについては、第 1.4.35 項を参照してください。

<i>w</i>	入力コンテキストに対応する DIMs ウィジェットを指定します。
<i>client_data</i>	DIMsNclientdataリソースで指定されるデータへのポインタを指定します。DIMsNclientdataについては、第 1.4.54 項を参照してください。
<i>value</i>	呼び出し側に、符号なし long 型の属性値へのポインタを返します。IMS プログラムは、このデータにメモリを割り当てる必要があります。このメモリ割り当ては IMSSL によって後で解放されます。
<i>attribute_flag</i>	PreeditAttribute または StatusAttribute のどちらに属するか、XIC 属性の型を指定します。DIMsAttributesTypeのデータ型については、setExtICValuesCb の項を参照してください。

getBackgroundCb

このコールバック関数は、XIC 属性 XNBackground の値を得るために呼び出されます。

フォーマット

```
int getBackgroundCb (w, client_data, value, attribute_flag)
```

引数

Widget	<i>w</i> ;	
Opaque	<i>*client_data</i> ;	
unsigned long	<i>**value</i> ;	(呼び出し側へ返す)
DIMsAttributesType	<i>attribute_flag</i> ;	

戻り値

このコールバックが正常終了した場合、IMSSL に 1 が返されます。このコールバックが異常終了した場合、IMSSL に 0 が返されます。

説明

getBackgroundCbは、リソースDIMsNgetBackgroundCbで指定されるコールバック関数です。DIMsNgetBackgroundCbについては、第 1.4.36 項を参照してください。

<i>w</i>	入力コンテキストに対応する DIMs ウィジェットを指定します。
<i>client_data</i>	DIMsNclientdataリソースで指定されるデータへのポインタを指定します。DIMsNclientdataについては、第 1.4.54 項を参照してください。
<i>value</i>	呼び出し側に、符号なし long 型の属性値へのポインタを返します。IMS プログラムは、このデータにメモリを割り当てる必要があります。このメモリ割り当ては IMSSL によって後で解放されます。
<i>attribute_flag</i>	PreeditAttribute または StatusAttribute のどちらに属するか、XIC 属性の型を指定します。DIMsAttributesTypeのデータ型については、setExtICValuesCb の項を参照してください。

getBgPixmapCb

このコールバック関数は、XIC 属性 XNBackgroundPixmap の値を得るために呼び出されます。

フォーマット

```
int getBgPixmapCb (w, client_data, value, attribute_flag)
```

引数

Widget	<i>w</i> ;	
Opaque	<i>*client_data</i> ;	
Pixmap	<i>**value</i> ;	(呼び出し側へ返す)
DIMsAttributesType	<i>attribute_flag</i> ;	

戻り値

このコールバックが正常終了した場合、IMSSL に 1 が返されます。このコールバックが異常終了した場合、IMSSL に 0 が返されます。

説明

getBgPixmapCb は、リソース DIMsNgetBgPixmapCb で指定されるコールバック関数です。DIMsNgetBgPixmapCb については、第 1.4.37 項を参照してください。

<i>w</i>	入力コンテキストに対応する DIMs ウィジェットを指定します。
<i>client_data</i>	DIMsNclientdata リソースで指定されるデータへのポインタを指定します。DIMsNclientdata については、第 1.4.54 項を参照してください。
<i>value</i>	呼び出し側に、Pixmap 型の属性値へのポインタを返します。IMS プログラムは、このデータにメモリを割り当てる必要があります。このメモリ割り当ては IMSSL によって後で解放されます。
<i>attribute_flag</i>	PreeditAttribute または StatusAttribute のどちらに属するか、XIC 属性の型を指定します。DIMsAttributesType のデータ型については、setExtICValuesCb の項を参照してください。

getFontSetCb

このコールバック関数は、XIC 属性 XNFontSet の値を得るために呼び出されます。

フォーマット

```
int getFontSetCb (w, client_data, value, attribute_flag)
```

引数

Widget	<i>w</i> ;	
Opaque	<i>*client_data</i> ;	
XFontSet	<i>**value</i> ;	(呼び出し側に返す)
DIMsAttributesType	<i>attribute_flag</i> ;	

戻り値

このコールバックが正常終了した場合、IMSSL に 1 が返されます。このコールバックが異常終了した場合、IMSSL に 0 が返されます。

説明

getFontSetCbは、リソースDIMsNgetFontSetCbで指定されるコールバック関数です。DIMsNgetFontSetCbについては、第 1.4.38 項を参照してください。

<i>w</i>	入力コンテキストに対応する DIMs ウィジェットを指定します。
<i>client_data</i>	DIMsNclientdataリソースで指定されるデータへのポインタを指定します。DIMsNclientdataについては、第 1.4.54 項を参照してください。
<i>value</i>	呼び出し側に、XFontSet 型の属性値へのポインタを返します。IMS プログラムは、このデータにメモリを割り当てる必要があります。このメモリ割り当ては IMSSL によって後で解放されます。
<i>attribute_flag</i>	PreeditAttribute または StatusAttribute のどちらに属するか、XIC 属性の型を指定します。DIMsAttributesTypeのデータ型については、setExtICValuesCb の項を参照してください。

getLineSpaceCb

このコールバック関数は、XIC 属性 XNLineSpace の値を得るために呼び出されます。

フォーマット

```
int getLineSpaceCb (w, client_data, value, attribute_flag)
```

引数

Widget	<i>w;</i>	
Opaque	<i>*client_data;</i>	
int	<i>**value;</i>	(呼び出し側へ返す)
DIMsAttributesType	<i>attribute_flag;</i>	

戻り値

このコールバックが正常終了した場合、IMSSL に 1 が返されます。このコールバックが異常終了した場合、IMSSL に 0 が返されます。

説明

getLineSpaceCbは、リソースDIMsNgetLineSpaceCbで指定されるコールバック関数です。DIMsNgetLineSpaceCbについては、第 1.4.39 項を参照してください。

<i>w</i>	入力コンテキストに対応する DIMs ウィジェットを指定します。
<i>client_data</i>	DIMsNclientdataリソースで指定されるデータへのポインタを指定します。DIMsNclientdataについては、第 1.4.54 項を参照してください。
<i>value</i>	呼び出し側に、integer 型の属性値へのポインタを返します。IMS プログラマは、このデータにメモリを割り当てる必要があります。このメモリ割り当てはIMSSLによって後で解放されます。
<i>attribute_flag</i>	PreeditAttribute または StatusAttribute のどちらに属するか、XIC 属性の型を指定します。DIMsAttributesTypeのデータ型については、setExtICValuesCb の項を参照してください。

getCursorCb

このコールバック関数は、XIC 属性 XNCursor の値を得るために呼び出されます。

フォーマット

int getCursorCb (w, client_data, value, attribute_flag)

引数

Widget	w;	
Opaque	*client_data;	
Cursor	**value;	(呼び出し側へ返す)
DIMsAttributesType	attribute_flag;	

戻り値

このコールバックが正常終了した場合、IMSSL に 1 が返されます。このコールバックが異常終了した場合、IMSSL に 0 が返されます。

説明

getCursorCbは、リソースDIMsNgetCursorCbで指定されるコールバック関数です。DIMsNgetCursorCbについては、第 1.4.40 項を参照してください。	
w	入力コンテキストに対応する DIMs ウィジェットを指定します。
client_data	DIMsNclientdataリソースで指定されるデータへのポインタを指定します。DIMsNclientdataについては、第 1.4.54 項を参照してください。
value	呼び出し側に、Cursor 型の属性値へのポインタを返します。IMS プログラムは、このデータにメモリを割り当てる必要があります。このメモリ割り当てはIMSSLによって後で解放されます。
attribute_flag	PreeditAttribute または StatusAttribute のどちらに属するか、XIC 属性の型を指定します。DIMsAttributesTypeのデータ型については、setExtICValuesCb の項を参照してください。

triggerNotifyCb

このコールバック関数は、アプリケーションが DIMsNtriggerKeys で設定されたトリガ・キーを受け取ったときに呼び出されます。

フォーマット

```
int triggerNotifyCb (w, client_data, flag, index)
```

引数

Widget	<i>w</i> ;
Opaque	<i>*client_data</i> ;
int	<i>flag</i> ;
int	<i>index</i> ;

戻り値

このコールバックが正常終了した場合、IMSSL に 1 が返されます。このコールバックが異常終了した場合、IMSSL に 0 が返されます。

説明

triggerNotifyCbは、リソースDIMsNtriggerNotifyCbで指定されるコールバック関数です。DIMsNtriggerNotifyCbについては、第 1.4.41 項を参照してください。

<i>w</i>	入力コンテキストに対応する DIMs ウィジェットを指定します。
<i>client_data</i>	DIMsNclientdataリソースで指定されるデータへのポインタを指定します。DIMsNclientdataについては、第 1.4.54 項を参照してください。
<i>flag</i>	トリガ・オンまたはトリガ・オフのどちらを受け取ったかを表すフラグです。 0 - トリガ・オン 1 - トリガ・オフ
<i>index</i>	DIMsNtriggerKeysリソースで指定されるトリガ・オン/オフのキー・リストのインデックスです。DIMsNtriggerKeysについては、第 1.4.8 項を参照してください。

setStringConversionCb

このコールバック関数は、XIC 属性 XNStringConversionCb の値を設定するために呼び出されます。

フォーマット

```
int setStringConversionCb (w, client_data, reason, value, attribute_flag)
```

引数

Widget	<i>w</i> ;
Opaque	<i>*client_data</i> ;
int	<i>reason</i> ;
XIMStringConversionText	<i>*value</i> ;
DIMsAttributesType	<i>attribute_flag</i> ;

戻り値

このコールバックが正常終了した場合、IMSSL に 1 が返されます。このコールバックが異常終了した場合、IMSSL に 0 が返されます。

説明

setStringConversionCbは、リソースDIMsNsetStringConversionCbで指定されるコールバック関数です。DIMsNsetStringConversionCbについては、第 1.4.43 項を参照してください。

<i>w</i>	入力コンテキストに対応する DIMs ウィジェットを指定します。
<i>client_data</i>	DIMsNclientdataリソースで指定されるデータへのポインタを指定します。DIMsNclientdataについては、第 1.4.54 項を参照してください。
<i>reason</i>	コールバックの理由を指定します。DimCR_CreateICまたはDimCR_SetICValuesのいずれかを指定します。DimCR_CreateICはクライアントの XCreateIC()によってコールバックが開始されることを意味し、DimCR_SetICValuesはクライアントの XSetICValues()によってコールバックが開始されることを意味します。
<i>value</i>	XIMStringConversionText 型の属性値へのポインタを指定します。
<i>attribute_flag</i>	PreeditAttribute または StatusAttribute のどちらに属するか、XIC 属性の型を指定します。DIMsAttributesTypeのデータ型については、setExtICValuesCb の項を参照してください。

setStringConversionCallbackCb

このコールバック関数は、XIC 属性 XNStringConversionCallbackCb の値を設定するために呼び出されます。

フォーマット

```
int setStringConversionCallbackCb (w, client_data, reason, attribute_flag)
```

引数

Widget	<i>w</i> ;
Opaque	<i>*client_data</i> ;
int	<i>reason</i> ;
DIMsAttributesType	<i>attribute_flag</i> ;

戻り値

このコールバックが正常終了した場合、IMSSL に 1 が返されます。このコールバックが異常終了した場合、IMSSL に 0 が返されます。

説明

setStringConversionCallbackCbは、リソースDIMsNsetStringConversionCallbackCbで指定されるコールバック関数です。DIMsNsetStringConversionCbについては、第 1.4.43 項を参照してください。

<i>w</i>	入力コンテキストに対応する DIMs ウィジェットを指定します。
<i>client_data</i>	DIMsNclientdataリソースで指定されるデータへのポインタを指定します。DIMsNclientdataについては、第 1.4.54 項を参照してください。
<i>reason</i>	コールバックの理由を指定します。DimCR_CreateICまたはDimCR_SetICValuesのいずれかを指定します。DimCR_CreateICはクライアントの XCreateIC()によってコールバックが開始されることを意味し、DimCR_SetICValuesはクライアントの XSetICValues()によってコールバックが開始されることを意味します。
<i>attribute_flag</i>	PreeditAttribute または StatusAttribute のどちらに属するか、XIC 属性の型を指定します。DIMsAttributesTypeのデータ型については、setExtICValuesCb の項を参照してください。

setResetStateCb

このコールバック関数は，XIC 属性 XNResetState の値を設定するために呼び出されます。

フォーマット

```
int setResetStateCb (w, client_data, reason, value, attribute_flag)
```

引数

Widget	<i>w</i> ;
Opaque	<i>*client_data</i> ;
int	<i>reason</i> ;
XIMResetState	<i>*value</i> ;
DIMsAttributesType	<i>attribute_flag</i> ;

戻り値

このコールバックが正常終了した場合，IMSSL に 1 が返されます。このコールバックが異常終了した場合，IMSSL に 0 が返されます。

説明

setResetStateCbは，リソースDIMsNsetResetStateCbで指定されるコールバック関数です。DIMsNsetResetStateCbについては，第 1.4.45 項を参照してください。

<i>w</i>	入力コンテキストに対応する DIMs ウィジェットを指定します。
<i>client_data</i>	DIMsNclientdataリソースで指定されるデータへのポインタを指定します。DIMsNclientdataについては，第 1.4.54 項を参照してください。
<i>reason</i>	コールバックの理由を指定します。DimCR_CreateICまたはDimCR_SetICValuesのいずれかを指定します。DimCR_CreateICはクライアントの XCreateIC()によってコールバックが開始されることを意味し，DimCR_SetICValuesはクライアントの XSetICValues()によってコールバックが開始されることを意味します。
<i>value</i>	XIMResetState 型の属性値へのポインタを指定します。
<i>attribute_flag</i>	PreeditAttribute または StatusAttribute のどちらに属するか，XIC 属性の型を指定します。DIMsAttributesTypeのデータ型については，setExtICValuesCb の項を参照してください。

getResetStateCb

このコールバック関数は、XIC 属性 XNResetStateCb の値を得るために呼び出されます。

フォーマット

```
int getResetStateCb (w, client_data, value, attribute_flag)
```

引数

Widget	<i>w</i> ;	
Opaque	<i>*client_data</i> ;	
XIMResetState	<i>**value</i> ;	(呼び出し側へ返す)
DIMsAttributesType	<i>attribute_flag</i> ;	

戻り値

このコールバックが正常終了した場合、IMSSL に 1 が返されます。このコールバックが異常終了した場合、IMSSL に 0 が返されます。

説明

getResetStateCbは、リソースDIMsNgetResetStateCbで指定されるコールバック関数です。DIMsNgetResetStateCbについては、第 1.4.46 項を参照してください。

<i>w</i>	入力コンテキストに対応する DIMs ウィジェットを指定します。
<i>client_data</i>	DIMsNclientdataリソースで指定されるデータへのポインタを指定します。DIMsNclientdataについては、第 1.4.54 項を参照してください。
<i>value</i>	呼び出し側に、XIMResetState 型の属性値へのポインタを返します。IMS プログラマは、このデータにメモリを割り当てる必要があります。このメモリ割り当ては IMSSL によって後で解放されます。
<i>attribute_flag</i>	PreeditAttribute または StatusAttribute のどちらに属するか、XIC 属性の型を指定します。DIMsAttributesTypeのデータ型については、setExtICValuesCb の項を参照してください。

setHotKeyCb

このコールバック関数は，XIC 属性 XNHotKeyCb の値を設定するために呼び出されます。

フォーマット

int setHotKeyCb (w, client_data, reason, value, attribute_flag)

引数

Widget	<i>w</i> ;
Opaque	<i>*client_data</i> ;
int	<i>reason</i> ;
XIMHotKeyTriggers	<i>*value</i> ;
DIMsAttributesType	<i>attribute_flag</i> ;

戻り値

このコールバックが正常終了した場合，IMSSL に 1 が返されます。このコールバックが異常終了した場合，IMSSL に 0 が返されます。

説明

setHotKeyCbは，リソースDIMsNsetHotKeyCbで指定されるコールバック関数です。DIMsNsetHotKeyCbについては，第 1.4.47 項を参照してください。

<i>w</i>	入力コンテキストに対応する DIMs ウィジェットを指定します。
<i>client_data</i>	DIMsNclientdataリソースで指定されるデータへのポインタを指定します。DIMsNclientdataについては，第 1.4.54 項を参照してください。
<i>reason</i>	コールバックの理由を指定します。DimCR_CreateICまたはDimCR_SetICValuesのいずれかを指定します。DimCR_CreateICはクライアントの XCreateIC()によってコールバックが開始されることを意味し，DimCR_SetICValuesはクライアントの XSetICValues()によってコールバックが開始されることを意味します。
<i>value</i>	XIMHotKeyTriggers 型の属性値へのポインタを指定します。
<i>attribute_flag</i>	PreeditAttribute または StatusAttribute のどちらに属するか，XIC 属性の型を指定します。DIMsAttributesTypeのデータ型については，setExtICValuesCb の項を参照してください。

getHotKeyCb

このコールバック関数は、XIC 属性 XNHotKeyCb の値を得るために呼び出されます。

フォーマット

```
int getHotKeyCb (w, client_data, value, attribute_flag)
```

引数

Widget	<i>w</i> ;	
Opaque	<i>*client_data</i> ;	
XIMHotKeyTriggers	<i>**value</i> ;	(呼び出し側へ返す)
DIMsAttributesType	<i>attribute_flag</i> ;	

戻り値

このコールバックが正常終了した場合、IMSSL に 1 が返されます。このコールバックが異常終了した場合、IMSSL に 0 が返されます。

説明

getHotKeyCbは、リソースDIMsNgetHotKeyCbで指定されるコールバック関数です。DIMsNgetHotKeyCbについては、第 1.4.48 項を参照してください。

<i>w</i>	入力コンテキストに対応する DIMs ウィジェットを指定します。
<i>client_data</i>	DIMsNclientdataリソースで指定されるデータへのポインタを指定します。DIMsNclientdataについては、第 1.4.54 項を参照してください。
<i>value</i>	呼び出し側に、XIMHotKeyTriggers 型の属性値へのポインタを返します。IMS プログラマは、このデータにメモリを割り当てる必要があります。このメモリ割り当ては IMSSL によって後で解放されます。
<i>attribute_flag</i>	PreeditAttribute または StatusAttribute のどちらに属するか、XIC 属性の型を指定します。DIMsAttributesTypeのデータ型については、setExtICValuesCb の項を参照してください。

setHotKeyStateCb

このコールバック関数は，XIC 属性 XNHotKeyStateCb の値を設定するために呼び出されます。

フォーマット

```
int setHotKeyStateCb (w, client_data, reason, value, attribute_flag)
```

引数

Widget	<i>w</i> ;	
Opaque	<i>*client_data</i> ;	
int	<i>reason</i> ;	
XIMHotKeyState	<i>**value</i> ;	(呼び出し側へ返す)
DIMsAttributesType	<i>attribute_flag</i> ;	

戻り値

このコールバックが正常終了した場合，IMSSL に 1 が返されます。このコールバックが異常終了した場合，IMSSL に 0 が返されます。

説明

setHotKeyStateCbは，リソースDIMsNsetHotKeyStateCbで指定されるコールバック関数です。DIMsNsetHotKeyStateCbについては，第 1.4.49 項を参照してください。

<i>w</i>	入力コンテキストに対応する DIMs ウィジェットを指定します。
<i>client_data</i>	DIMsNclientdataリソースで指定されるデータへのポインタを指定します。DIMsNclientdataについては，第 1.4.54 項を参照してください。
<i>reason</i>	コールバックの理由を指定します。DimCR_CreateICまたはDimCR_SetICValuesのいずれかを指定します。DimCR_CreateICはクライアントの XCreateIC()によってコールバックが開始されることを意味し，DimCR_SetICValuesはクライアントの XSetICValues()によってコールバックが開始されることを意味します。
<i>value</i>	XIMHotKeyState 型の属性値へのポインタを指定します。
<i>attribute_flag</i>	PreeditAttribute または StatusAttribute のどちらに属するか，XIC 属性の型を指定します。DIMsAttributesTypeのデータ型については，setExtICValuesCb の項を参照してください。

getHotKeyStateCb

このコールバック関数は、XIC 属性 XNHotKeyStateCb の値を得るために呼び出されます。

フォーマット

```
int getHotKeyStateCb (w, client_data, value, attribute_flag)
```

引数

Widget	<i>w</i> ;	
Opaque	<i>*client_data</i> ;	
XIMHotKeyState	<i>**value</i> ;	(呼び出し側へ返す)
DIMsAttributesType	<i>attribute_flag</i> ;	

戻り値

このコールバックが正常終了した場合、IMSSL に 1 が返されます。このコールバックが異常終了した場合、IMSSL に 0 が返されます。

説明

getHotKeyStateCbは、リソースDIMsNgetHotKeyStateCbで指定されるコールバック関数です。DIMsNgetHotKeyStateCbについては、第 1.4.50 項を参照してください。

<i>w</i>	入力コンテキストに対応する DIMs ウィジェットを指定します。
<i>client_data</i>	DIMsNclientdataリソースで指定されるデータへのポインタを指定します。DIMsNclientdataについては、第 1.4.54 項を参照してください。
<i>value</i>	呼び出し側に、XIMHotKeyState 型の属性値へのポインタを返します。IMS プログラマは、このデータにメモリを割り当てる必要があります。このメモリ割り当ては IMSSL によって後で解放されます。
<i>attribute_flag</i>	PreeditAttribute または StatusAttribute のどちらに属するか、XIC 属性の型を指定します。DIMsAttributesTypeのデータ型については、setExtICValuesCb の項を参照してください。

setPreeditStateCb

このコールバック関数は、XIC 属性 XNPreeditStateCb の値を設定するために呼び出されます。

フォーマット

```
int setPreeditStateCb (w, client_data, reason, value, attribute_flag)
```

引数

Widget	<i>w</i> ;
Opaque	<i>*client_data</i> ;
int	<i>reason</i> ;
XIMPreeditState	<i>*value</i> ;
DIMsAttributesType	<i>attribute_flag</i> ;

戻り値

このコールバックが正常終了した場合、IMSSL に 1 が返されます。このコールバックが異常終了した場合、IMSSL に 0 が返されます。

説明

setPreeditStateCbは、リソースDIMsNsetPreeditStateCbで指定されるコールバック関数です。DIMsNsetPreeditStateCbについては、第 1.4.51 項を参照してください。

<i>w</i>	入力コンテキストに対応する DIMs ウィジェットを指定します。
<i>client_data</i>	DIMsNclientdataリソースで指定されるデータへのポインタを指定します。DIMsNclientdataについては、第 1.4.54 項を参照してください。
<i>reason</i>	コールバックの理由を指定します。DimCR_CreateICまたはDimCR_SetICValuesのいずれかを指定します。DimCR_CreateICはクライアントの XCreateIC()によってコールバックが開始されることを意味し、DimCR_SetICValuesはクライアントの XSetICValues()によってコールバックが開始されることを意味します。
<i>value</i>	XIMPreeditState 型の属性値へのポインタを指定します。
<i>attribute_flag</i>	PreeditAttribute または StatusAttribute のどちらに属するか、XIC 属性の型を指定します。DIMsAttributesTypeのデータ型については、setExtICValuesCb の項を参照してください。

getPreeditStateCb

このコールバック関数は、XIC 属性 XNPreeditStateCb の値を得るために呼び出されます。

フォーマット

```
int getPreeditStateCb (w, client_data, value, attribute_flag)
```

引数

Widget	w;	
Opaque	*client_data;	
XIMPreeditState	**value;	(呼び出し側へ返す)
DIMsAttributesType	attribute_flag;	

戻り値

このコールバックが正常終了した場合、IMSSL に 1 が返されます。このコールバックが異常終了した場合、IMSSL に 0 が返されます。

説明

getPreeditStateCbは、リソースDIMsNgetPreeditStateCbで指定されるコールバック関数です。DIMsNgetPreeditStateCbについては、第 1.4.52 項を参照してください。

w	入力コンテキストに対応する DIMs ウィジェットを指定します。
client_data	DIMsNclientdataリソースで指定されるデータへのポインタを指定します。DIMsNclientdataについては、第 1.4.54 項を参照してください。
value	呼び出し側に、XIMPreeditState 型の属性値へのポインタを返します。IMS プログラマは、このデータにメモリを割り当てる必要があります。このメモリ割り当ては IMSSL によって後で解放されます。
attribute_flag	PreeditAttribute または StatusAttribute のどちらに属するか、XIC 属性の型を指定します。DIMsAttributesTypeのデータ型については、setExtICValuesCb の項を参照してください。

setPreeditStateCallbackCb

このコールバック関数は、XIC 属性 XNPreeditStateCallbackCb の値を設定するために呼び出されます。

フォーマット

int setPreeditStateCallbackCb (*w, client_data, reason, attribute_flag*)

引数

Widget	<i>w</i> ;
Opaque	<i>*client_data</i> ;
int	<i>reason</i> ;
DIMsAttributesType	<i>attribute_flag</i> ;

戻り値

このコールバックが正常終了した場合、IMSSL に 1 が返されます。このコールバックが異常終了した場合、IMSSL に 0 が返されます。

説明

setPreeditStateCallbackCbは、リソースDIMsNsetPreeditStateCallbackCbで指定されるコールバック関数です。DIMsNsetPreeditStateCallbackCbについては、第 1.4.53 項を参照してください。

<i>w</i>	入力コンテキストに対応する DIMs ウィジェットを指定します。
<i>client_data</i>	DIMsNclientdataリソースで指定されるデータへのポインタを指定します。DIMsNclientdataについては、第 1.4.54 項を参照してください。
<i>reason</i>	コールバックの理由を指定します。DimCR_CreateICまたはDimCR_SetICValuesのいずれかを指定します。DimCR_CreateICはクライアントのXCreateIC()によってコールバックが開始されることを意味し、DimCR_SetICValuesはクライアントのXSetICValues()によってコールバックが開始されることを意味します。
<i>attribute_flag</i>	PreeditAttribute または StatusAttribute のどちらに属するか、XIC 属性の型を指定します。DIMsAttributesTypeのデータ型については、setExtICValuesCb の項を参照してください。

1.6 サンプル・ファイル

入力サーバに関するサンプル・ファイルが DECW\$EXAMPLES:IMS_EXAMPLE.C に用意されています。このサンプル・ファイルを参照することにより、API の使用方法をより明確に理解することができるでしょう。

1.6.1 ビルド・プロシージャ

ビルド用のコマンド・プロシージャとして DECW\$EXAMPLES:IMS_EXAMPLE.COM が用意されています。これにより、実行可能な入力サーバをビルドできます。IMS の開発に必要なすべての API およびデータ構造体を含むヘッダ・ファイルは、DECW\$INCLUDE:DIMSDEF.H にあります。IMSSL は共有ライブラリ SYSS\$SHARE:DECW\$DIMSLIBSHR.EXE として提供されます。

1.6.2 ソース・プログラム

```
*****
*
* IMS_example.c
*
* This is an example of how to write an Input Method Server for X11R6.
* The only protocol being used in here is X118n and the transport mechanism
* is X.
*
* In this example, there are only 26 composed characters, which are
* formed by 2 consecutive key inputs which are equal. These key inputs
* should be from a-z, case insensitive. The lookup string
* will be found from an array, which is some kind of small-scale dictionary.
*
* You should load Iso-latin 1 keymap files in order to work properly for
* this example.
*
* You can start this example using four of the languages:
* Korean, Traditional Chinese, Simplified Chinese and Japanese
*
* To start this IMServer, set the xnllanguage resource properly, for example,
* if you want to input Korean text, you should set the session language to
* Korean, or simply start as
* $IMS_example -xnllanguage "ko_KR.deckorean"
*
* To connect a client to this IMS, simply start up this IMS as a background
* job, then start the client with the correct language setting and XMODIFIERS
* environment. There are 4 modifiers that can be set according to locales :
* "zh_tw_exam", "zh_cn_exam", "ko_kr_exam", ja_jp_exam". For example,
*
* $define XMODIFIERS "@im=ko_kr_exam"
* $mc decw$cardfiler -xnllanguage "ko_KR.deckorean"
*/
```

```

#include
#include
#ifdef VMS
#include "DIMSDef.h"
/* #include */
#else
#include
#endif
static char my_dictionary[26][2] = {0xa1, 0xa1,
                                   0xa1, 0xa2,
                                   0xa1, 0xa3,
                                   0xa1, 0xa4,
                                   0xa1, 0xa5,
                                   0xa1, 0xa6,
                                   0xa1, 0xa7,
                                   0xa1, 0xa8,
                                   0xa1, 0xa9,
                                   0xa1, 0xaa,
                                   0xa1, 0xab,
                                   0xa1, 0xac,
                                   0xa1, 0xad,
                                   0xa1, 0xae,
                                   0xa1, 0xaf,
                                   0xa1, 0xb1,
                                   0xa1, 0xb2,
                                   0xa1, 0xb3,
                                   0xa1, 0xb4,
                                   0xa1, 0xb5,
                                   0xa1, 0xb6,
                                   0xa1, 0xb7,
                                   0xa1, 0xb8,
                                   0xa1, 0xb9,
                                   0xa1, 0xba,
                                   0xa1, 0xbb};

typedef struct {
    char data[1];
    int num_char;
} MyDataRec, *MyData; /* This is the input context specific data */

static int CreateICallback(w, client_data, widget_data)
Widget w;
Opaque *client_data;
Opaque **widget_data;
{
    MyData my_data = (MyData)XtMalloc(sizeof(MyDataRec));
    if (!my_data)
        return 0; /* fail */
    my_data->num_char = 0;
    *widget_data = (Opaque *)my_data;
    return 1; /* success */
}

```

1.6 サンプル・ファイル

```
static int DestroyICCallback(w, client_data, widget_data)
Widget w;
Opaque *client_data;
{
    MyData my_data = (MyData)DIMsUserData(w);
    if (my_data)
        XtFree((char *)my_data);
    return 1; /* success */
}

static int GetFilterEventsCallback(w, client_data, mask)
Widget w;
Opaque *client_data;
unsigned long **mask;
{
    *mask = (unsigned long *) XtMalloc(sizeof(unsigned long));
    /* Since this IMS supports root-window style only, so just KeyPress
       events are necessary */
    **mask = (unsigned long) (KeyPressMask);
    return 1;
}

static int ProcessEventCallback(w, client_data, event, send_back)
Widget w;
Opaque *client_data;
XEvent *event;
Boolean *send_back;
{
    MyData my_data = (MyData)DIMsUserData(w);
    Display *dpy = XtDisplay(w);
    *send_back = True;
    switch (event->type)
    {
        case KeyPress :
        {
            KeyCode keycode = event->xkey.keycode;
            KeySym keysym;
            char second_char;
            XTextProperty text_property;
            char *text_list[1];
            char text[3];
```



```

keysym = XKeycodeToKeysym( dpy, keycode, 0 );
if ((keysym >= XK_A && keysym <= XK_Z) ||
    (keysym >= XK_a && keysym <= XK_z))
{
    if (my_data->num_char)
    {
        if (keysym >= XK_a)
            second_char = keysym - XK_a;
        else
            second_char = keysym - XK_A;
        if (second_char == my_data->data[0])
        { /* two consecutive key are equal, send commit string */
            text[0] = my_dictionary[second_char][0];
            text[1] = my_dictionary[second_char][1];
            text[2] = '\0';
            text_list[0] = text;
            XmbTextListToTextProperty(dpy, text_list, 1,
                                       XCompoundTextStyle, &text_property);
            DIMSCommit(w, (char *)text_property.value,
                      strlen((char *)text_property.value)+1);
            XtFree((char *)text_property.value);
            *send_back = False; /* the key is used by IMS,
                                no need to return to client */
        }
        my_data->num_char = 0; /* reset to 0*/
    } else {
        my_data->num_char = 1;
        if (keysym >= XK_a)
            my_data->data[0] = keysym - XK_a;
        else
            my_data->data[0] = keysym - XK_A;
        *send_back = False;
    }
}
break;
}
default :
break;
}

return 1;
}

}

IgnoreXErrors(dpy, event)
Display *dpy;
XErrorEvent *event;
{
    return 0;
}

```

1.6 サンプル・ファイル

```
main(argc, argv)
int argc;
char **argv;
{
    Widget toplevel;
    XtAppContext app_context;
    DIMsClassCtx ctx;
    char *zh_tw_server_strings[] = {"_XIM_zh_TW@zh_tw_exam"};
    char *zh_cn_server_strings[] = {"_XIM_zh_CN@zh_cn_exam"};
    char *ko_kr_server_strings[] = {"_XIM_ko_KR@ko_kr_exam"};
    char      *ja_jp_server_strings[] = {"_XIM_ja_JP@ja_jp_exam"};
    XIMStyle support_styles[1] = {XIMPreeditNothing | XIMStatusNothing};
    DIMsLocaleInfoRec locale;
    DIMsStringListRec server_names;
    DIMsSupportStylesRec im_styles;
    Cardinal n;
    Arg al[40];
    DIMsVersionRec im_version;

    XtSetLanguageProc(NULL, NULL, NULL);

    toplevel = XtAppInitialize(&app_context, "example", NULL, 0,
        &argc, argv, NULL, NULL, 0);

    /* set up DIMs Class */
    /* This example targets to work for 4 locales: zh_TW, ja_JP, zh_CN, ko_KR.
       Therefore many DIMs class contexts are created */
    /* create contexts for locales zh_TW, ja_JP, zh_CN, ko_KR */

    locale.language = "zh_TW";
    locale.num_support_codesets = 0;
    locale.support_codesets = NULL;

    server_names.num_of_strings = 1;
    server_names.string_names = zh_tw_server_strings;

    im_styles.num_support_styles = 1;
    im_styles.support_styles = support_styles;

    n = 0;
    XtSetArg( al[n], DIMsNlocale, &locale);n++;
    XtSetArg( al[n], DIMsNserverName, &server_names);n++;
    /* use Xil8n protocol */
    im_version.highest_major_version = 3;
    im_version.highest_minor_version = 0;
    im_version.lowest_major_version = 3;
    im_version.lowest_minor_version = 0;
    XtSetArg( al[n], DIMsNversion, &im_version);n++;
    XtSetArg( al[n], DIMsNsupportStyles, &im_styles);n++;
    XtSetArg( al[n], DIMsNcreateICCb, CreateICCallback);n++;
    XtSetArg( al[n], DIMsNdestroyICCb, DestroyICCallback);n++;
    XtSetArg( al[n], DIMsNprocessEventCb, ProcessEventCallback);n++;
    XtSetArg( al[n], DIMsNgetFilterEventsCb, GetFilterEventsCallback);n++;
    XtSetArg( al[n], DIMsNtransport, "X/");n++;
    ctx = DIMsClassCtxCreate(NULL, "zh_tw_xil8n_exam", al, n);
```

```

    locale.language = "zh_CN";
    server_names.string_names = zh_cn_server_strings;
    XtSetArg( al[0], DIMsNlocale, &locale);
    XtSetArg( al[1], DIMsNserverName, &server_names);
    ctx = DIMsClassCtxCreate(ctx, "zh_cn_xi18n_exam", al, n);

    locale.language = "ko_KR";
    server_names.string_names = ko_kr_server_strings;
    XtSetArg( al[0], DIMsNlocale, &locale);
    XtSetArg( al[1], DIMsNserverName, &server_names);
    ctx = DIMsClassCtxCreate(ctx, "ko_kr_xi18n_exam", al, n);

    locale.language = "ja_JP";
    server_names.string_names = ja_jp_server_strings;
    XtSetArg( al[0], DIMsNlocale, &locale);
    XtSetArg( al[1], DIMsNserverName, &server_names);
    ctx = DIMsClassCtxCreate(ctx, "ja_jp_xi18n_exam", al, n);

    XSetErrorHandler(IgnorXErrors);

/* Initialize IMSSL */
    DIMsSSLInitialize(toplevel, ctx);

/* Realize widget hierrachy */
    XtRealizeWidget(toplevel);

/* Initialize protocol layer */
    DIMsProtoInit(toplevel, app_context);

    XtAppMainLoop(app_context);
}

```


日本語 hp DECwindows Motif for hp OpenVMS
DEC 入力サーバ・ライブラリ

2003 年 3 月 発行

日本ヒューレット・パッカート株式会社

〒140-8641 東京都品川区東品川 2-2-24 天王洲セントラルタワー

電話 (03)5463-6600 (大代表)
