

共通デスクトップ環境 Dtksh ユーザーズ・ガイド

Many of the designations used by manufacturers and sellers to distinguish their products are claimed as trademarks. Where those designations appear in this book, and Addison-Wesley was aware of a trademark claim, the designations have been printed in initial capital letters or all capital letters.

RESTRICTED RIGHTS LEGEND: Use, duplication, or disclosure by the United States Government is subject to the restrictions set forth in DFARS 252.227-7013 (c)(1)(ii) and FAR 52.227-19.

THIS PUBLICATION IS PROVIDED "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NON-INFRINGEMENT.

The code and documentation for the DtComboBox and DtSpinBox widgets were contributed by Interleaf, Inc. Copyright (c) 1993 Interleaf, Inc.

Copyright (c) 1993, 1994, 1995 Hewlett-Packard

Copyright (c) 1993, 1994, 1995 International Business Machines Corp.

Copyright (c) 1993, 1994, 1995 Novell, Inc.

Copyright (c) 1993, 1994, 1995 Sun Microsystems, Inc.

All rights reserved. This product and related documentation are protected by copyright and distributed under licenses restricting its use, copying, distribution, and decompilation. No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, electronic, mechanical, photocopying, recording, or otherwise, without the prior written permission of the publisher.



Please

Recycle

制作会社および販売会社が自らの製品を識別するために本書の中で使用している名称の多くは、商標となっております。本書の中で使用される名称のうち、Addison-Wesley が商標として使用されていると認識した名称は、頭文字を大文字とするか、またはその名称全体を大文字にして記載しております。

RESTRICTED RIGHTS LEGEND: Use, duplication, or disclosure by the United States Government is subject to the restrictions set forth in DFARS 252.227-7013 (c) (1) (ii) and FAR 52.227-19.

本書は現状有姿で頒布され、その商品性、特定目的への適合性または第三者の権利の非侵害に対する黙示の保証を含め、明示的であるか黙示的であるかを問わず、いかなる保証も行いうものではありません。

DtComboBox ウィジェットと DtSpinBox ウィジェットのプログラムおよびドキュメントは、Interleaf, Inc. から提供されたものです (Copyright (c) 1993 Interleaf, Inc.)。

本製品および関連するドキュメントは著作権法により保護されており、その使用、複製、頒布および逆コンパイルを制限するライセンスのもとに頒布されております。

本書のいかなる部分も出版社の許可なく、電子的、機械的、フォトコピー、記録またはその他いかなる形式・方法によっても、複製、読み取り可能なシステムへの保存、または転送することを禁止します。

Copyright (c) 1993, 1994, 1995 Hewlett-Packard

Copyright (c) 1993, 1994, 1995 International Business Machines Corp.

Copyright (c) 1993, 1994, 1995 Novell, Inc.

Copyright (c) 1993, 1994, 1995 Sun Microsystems, Inc.

All rights reserved.



Please
Recycle

目次

はじめに	ix
第 1 章 デスクトップ Korn シェルの紹介	1
デスクトップ Korn シェルを使用して Motif アプリケーションを作成する	1
リソース	2
サポートしていないリソース	3
dtksh app-defaults ファイル	4
変数の値	6
戻り値	6
即時戻り値	8
Xt インタロインシックスの初期化	9
ウィジェットの作成	9
コールバックの使い方	11
コールバックの登録	11
データをコールバックへ渡す	12
第 2 章 サンプル・スクリプト	15

スクリプトの記述.....	15
コールバックの追加.....	18
第 3 章 上級トピック	19
コンテキスト変数の使い方.....	19
イベント・ハンドラ・コンテキスト変数	19
トランスレーション・コンテキスト変数	20
ワークスペース・コールバック・コンテキスト変数.....	20
入力コンテキスト変数.....	21
イベント・サブフィールドへのアクセス.....	22
ウィンドウ・マネージャのクローズ通知への応答.....	23
セッション・マネージャの保存状態通知への応答.....	24
ワークスペース・マネージャとの協調	27
ローカライズされたシェル・スクリプトの作成.....	28
dtksh を使用して X 描画関数 へアクセスする.....	28
ウィジェット・トランスレーションの設定.....	29
第 4 章 複雑なスクリプト	31
script_find の使い方	31
script_find の解析	33
関数とコールバック.....	34
メイン・スクリプト.....	36
付録 A dtksh コマンド.....	43
組み込み Xlib コマンド	43
組み込み Xt イントリンシクス・コマンド	45
組み込み Motif コマンド	51

組み込み共通デスクトップ環境 アプリケーション・ヘルプ・コマンド	63
組み込みローカル化コマンド	64
組み込み libDt セッション管理コマンド	64
組み込み libDt ワークスペース管理コマンド	65
組み込み libDt アクション・コマンド	67
組み込み libDt データ型作成コマンド	68
その他の組み込み libDt コマンド	70
組み込みデスクトップ・サービス・ メッセージ・セット・コマンド	70
付録 B dtksh 簡易関数	79
DtkshAddButtons	80
DtkshSetReturnKeyControls.	81
DtkshUnder、DtkshOver、DtkshRightOf、DtkshLeftOf	82
DtkshFloatRight、DtkshFloatLeft、 DtkshFloatTop、DtkshFloatBottom	83
DtkshAnchorRight、DtkshAnchorLeft、 DtkshAnchorTop、DtkshAnchorBottom	84
DtkshSpanWidth および DtkshSpanHeight	85
DtkshDisplayInformationDialog、 DtkshDisplayQuestionDialog、 DtkshDisplayWarningDialog、 DtkshDisplayWorkingDialog、 DtkshDisplayErrorDialog.	86
DtkshDisplayQuickHelpDialog および DtkshDisplayHelpDialog	87
付録 C script_find スクリプト	89

script_find の全リスト.....	89
Find.sticky	98
Find.help.....	98
索引	99

はじめに

この『共通デスクトップ環境 Dtksh ユーザーズ・ガイド』は、Korn シェル (K シェル) スクリプトで Motif アプリケーションを作成するのに必要な情報を提供します。最初に必要とされる基本的情報に加えて、複雑さを伴ういくつかのスクリプト例についても説明します。このマニュアルの全体を通して使用される用語 dtksh は、デスクトップ Korn シェルのことです。

対象読者

このマニュアルは、Motif アプリケーションを早くかつ簡単に作成したいが、C プログラミング言語の使用にあたって時間や知識がない、またはこの言語をあまり使用したくないプログラマー向けです。K シェル・プログラミング、Motif、Xt インタプリタ、および少なくとも Xlib に関して、よく理解していることが望まれます。C 言語に関する理解も役立ちます。

このマニュアルの構成

第 1 章「デスクトップ Korn シェルの紹介」

dtksh スクリプトによる Motif アプリケーションの記述を開始する時に必要とされる基本的な情報について説明します。

第 2 章「サンプル・スクリプト」

2 つの簡単な dtksh スクリプトについて説明します。最初のスクリプトは、ブリテン・ボード・ウィジェット内にプッシュ・ボタン・ウィジェットを作成します。2 番目のスクリプトは、プッシュ・ボタン用のコールバックを追加して、最初のスクリプトを拡張します。

第 3 章「上級トピック」

dtksh スクリプトに関する、より高度なトピックについて説明します。

第 4 章「複雑なスクリプト」

第 2 章「サンプル・スクリプト」で説明しているものに比べてかなり複雑な構造を持つスクリプトについて説明します。このスクリプトは、find コマンドに対するグラフィック・インタフェースを作成します。

付録 A「dtksh コマンド」

すべての dtksh コマンドのリストです。

付録 B「dtksh 簡易関数」

他のマニュアルに載っていないコマンドや関数のマニュアル・ページを載せています。

付録 C「script_find スクリプト」

第 4 章「複雑なスクリプト」で説明している複雑なスクリプトのリストをすべて載せています。

関連文書

次のマニュアルに、K シェル・プログラミング、Motif、Xt イントリンスクス、および Xlib に関する情報が載っています。

- 『KornShell Programming Tutorial』 by Barry Rosenberg, published by Addison-Wesley, Reading, MA 01867.
- 『OSF/Motif Programmer's Guide』 Open Software Foundation, 11 Cambridge Center, Cambridge, MA 02142, published by Prentice-Hall, Englewood Cliffs, NJ 07632.
- 『OSF/Motif Programmer's Reference』 Open Software Foundation, 11 Cambridge Center, Cambridge, MA 02142, published by Prentice-Hall, Englewood Cliffs, NJ 07632.

-
- 『OSF/Motif Reference Guide』 by Douglas A. Young, published by Prentice-Hall, Englewood Cliffs, NJ 07632.
 - 『Mastering OSF/Motif Widgets』 (Second Edition), by Donald L. McMinds, published by Addison-Wesley, Reading, MA 01867
 - 『The X Window System Programming and Applications with Xt OSF/Motif Edition』 by Douglas A. Young, published by Prentice-Hall, Englewood Cliffs, NJ 07632.
 - 『The Definitive Guides to the X Window System, Volume 1: Xlib Programming Manual』 by Adrian Nye, published by O'Reilly and Associates, Sebastopol, CA 95472.
 - 『The Definitive Guides to the X Window System, Volume 2: Xlib Reference Manual』 edited by Adrian Nye, published by O'Reilly and Associates, Sebastopol, CA 95472.
 - 『The Definitive Guides to the X Window System, Volume 3: X Window System User's Guide』 by Valerie Quercia and Tim O'Reilly, published by O'Reilly and Associates, Sebastopol, CA 95472.
 - 『The Definitive Guides to the X Window System, Volume 4: X Toolkit Intrinsic Programming Manual』 by Adrian Nye and Tim O'Reilly, published by O'Reilly and Associates, Sebastopol, CA 95472.
 - 『The Definitive Guides to the X Window System, Volume 5: X Toolkit Intrinsic Reference Manual』 edited by Tim O'Reilly, published by O'Reilly and Associates, Sebastopol, CA 95472.
 - 『The Definitive Guides to the X Window System, Volume 6: Motif Programming Manual』 by Dan Heller, published by O'Reilly and Associates, Sebastopol, CA 95472.

表記上の規則

次の表に、このマニュアルで使用する表記上の規則を示します。

表 P-1 表記上の規則

文字または記号	意味	使用例
AaBbCc123	コマンド、ファイル、ディレクトリ の名前、画面の出力表示	login ファイルを編集します。 ls -a を使用してすべてのファイルをリ ストします。 system% you have mail.
<i>AaBbCc123</i>	変数を示します。実際に使用する特 定の名前または値で置き換えます。	ファイルを削除するには、 rm <i>filename</i> を入力します。
AaBbCc123	強調する語句を示します。	アクション・アイコンをクリックします。
『 』	参照する書名を示します。	『ユーザーズ・ガイド』を参照してく ださい。
「 」	参照する章、節を示します。	第 1 章「基本スキル」を参照してくだ さい。
[]	アイコン、ボタン、メニューなどの ラベル名に使用します。	[了解] ボタン
コード例は次のように表示されます。		
%	UNIX C シェル・プロンプト	system%
\$	UNIX Bourne シェル・プロンプト および Korn シェル・プロンプト	system\$
#	スーパーユーザ・プロンプト (すべての シェル)	system#

注 - \ (バックスラッシュ) は、デバイスによって ¥ (円記号) で表示されるものがあります。

デスクトップ Korn シェルの紹介

1 

デスクトップ Korn シェル (dtksh) は、既存の Xt および MotifTM のほとんどの関数に容易にアクセスできる K シェル・スクリプトを提供します。dtksh は、ksh-93 が基になっています。ksh-93 は、シェル・プログラマ用の強力なツールおよびコマンドのセットを提供し、K シェル・プログラミング・コマンドの標準セットをサポートするものです。

dtksh は、ksh-93 が提供する機能およびコマンドのすべてをサポートします。さらに dtksh は、選り抜かれた数多くの libDt 関数、ウィジェット関連の Motif 関数の大部分、Xt インタリンシクス関数の大きなサブセット、および Xlib 関数の小さなサブセットもサポートします。サポートされるすべての関数のリストが付録 A「dtksh コマンド」にあります。

デスクトップ Korn シェルを使用して Motif アプリケーションを作成する

この節では、dtksh を使用して Motif アプリケーションを作成する方法について説明します。dtksh を使いこなすには、Xlib、Xt インタリンシクス、Motif のウィジェット、および Korn シェル・プログラミングに対する経験が必要です。また、C プログラミング言語の知識も役に立ちます。これらのどれにも馴染みがない場合は、必要と思われる適切なマニュアルを参照してください。これらのシステムに精通している場合でも、適切なマニュアル・ページにアクセスして、関連内容を参照してください。

システムには、次のライブラリがあります。

- libDtHelp
- libDtSvc
- libX11

- libXm
- libXt
- libtt

リソース

リソースはウィジェットの変数のことで、サイズ、位置、またはカラーといった属性の定義に使用します。各ウィジェットが持つリソースは通常、自身のリソースと、より高いレベルのウィジェットから引き継ぐリソースとが組み合わさったものです。Xt イントリンシクスおよび Motif のリソース名は接頭辞 (XtN または XmN) を持ち、その後にベース名が続きます。ベース名の最初の文字は常に小文字で、その後のベース名内の単語の最初の文字は常に大文字です。dtksh スクリプトでは、接頭辞を削除して、残ったベース名がリソース名になります。したがって、リソース XmNtopShadowColor は topShadowColor になります。

Xt および Motif のコマンドの中には、シェル・スクリプトがリソースと、リソース値のペアを表すパラメータの変数を渡せるものがあります。これは、引き数が関連する Xt または Motif の C 関数に渡されるのと似ています。この例としては、ウィジェットの作成に使用するコマンドと XtSetValues コマンドがあります。dtksh において、リソースは、次の構文で指定されます。

resource:value

この *resource* はリソース名で、*value* はリソースに割り当てられる値です。dtksh は、自動的に *value* 文字列を適切な内部表現に変換します。たとえば、次のとおりです。

```
XtSetValues $WIDGET height:100 width:200 resizePolicy:RESIZE_ANY
XmCreateLabel LABEL $PARENT myLabel labelString:"Close Dialog"
```

XtGetValues を使用してウィジェット・リソース値を読み取る場合、戻り値は環境変数に設定されます。したがって、Xt イントリンシクスとは異なり、dtksh 版の XtGetValues は、名前:値ペアの形式ではなく、名前:(環境)変数ペアの形式を使用します。たとえば、次のとおりです。

```
XtGetValues $WIDGET height:HEIGHT resizePolicy:POLICY
             sensitive:SENSITIVE
echo $HEIGHT
echo $POLICY
echo $SENSITIVE
```

上記の dtksh セグメントは、次の出力を行います。

100
RESIZE ANY
TRUE

ある種のリソース値（文字列テーブルやビット・マスクなどを含む）は、特殊な表現になります。たとえば、リスト・ウィジェットでは、文字列テーブルに `items` と `selectedItems` の両方のリソースの指定が可能です。dtksh では、文字列テーブルは、カンマで区切られた文字列リストで表現され、これは Motif の場合と似ています。文字列テーブルを返すリソースが `XtGetValues` による問い合わせを受けた場合、その結果としての値は、カンマで区切られた文字列セットになります。

ビット・マスク値が渡されるリソースは、そのマスクが、| (バー) で区切られたさまざまなマスク値から成る文字列として指定されていることを要求します。ビット・マスクを返すリソースが問い合わせを受けた場合、戻り値は、ビットを表す文字列（カンマで区切られている）になります。たとえば、次のコマンドを使用して、VendorShell ウィジェット・クラスの `mwmFunctions` リソースを設定できます。

`XtSetValues mwmFunctions: MWM_FUNC_ALL|MWM_FUNC_RESIZE`

サポートしていないリソース

dtksh は、Motif リソースのほとんどをサポートしています。次のリストは、サポートしていないリソースのリストです。* (アスタリスク) の付いているリソースは、ウィジェット作成時に `XtSetValues` を使用して指定できるが、`XtGetValues` を使用しての取得はできません。

- すべてのウィジェットおよびガジェットのクラス
 - 任意の `fontlist` リソース *
 - 任意の `pixmap` リソース *
- コンボジット
 - `insertPosition`
 - `children`
- コア
 - `accelerators`
 - `translations` *
 - `colormap`

- XmText
 - selectionArray
 - selectionArrayCount
- ApplicationShell
 - argv
- WMShell
 - iconWindow
 - WindowGroup
- Shell
 - createPopupChildrenProc
- XmSelectionBox
 - textAccelerators
- Manager、Primitive、および Gadget サブクラス
 - userData
- XmFileSelectionBox
 - dirSearchProc
 - fileSearchProc
 - qualifySearchDataProc

dtksh app-defaults ファイル

Dtksh という名の dtksh app-defaults ファイルは、次のパスで示した場所にあります。

`/usr/dt/app-defaults/<LANG>`

この app-defaults ファイルに納められている唯一の情報は、標準 Dt ベースの app-defaults ファイルに納められているものです。dtksh app-defaults ファイルの中身は、次のとおりです。

```
#include "Dt"
```


Dt ファイルもまた、/usr/dt/app-defaults/<LANG> にあり、次のとおりです。

```
*foregroundThreshold:70

!###
!#
!#  Help system specific resources
!#
!###

!#
!#  Display Area Colors
!#
!#  These resources set the colors for the display area (where
!#  actual help text is displayed).  The resources are complex
!#  because they have to override the standard color resources
!#  in all cases.
!#
*XmDialogShell.DtHelpDialog*DisplayArea.background: White
*XmDialogShell*XmDialogShell.DtHelpDialog*DisplayArea.background: White
*XmDialogShell.DtHelpDialog*DisplayArea.foreground: Black
*XmDialogShell*XmDialogShell.DtHelpDialog*DisplayArea.foreground: Black

!#
!#  Menu Accelerators
!#
!#  The following resources establish keyboard accelerators
!#  for the most frequently accessed menu commands.
!#

*DtHelpDialogWidget*searchMenu.keyword.acceleratorText: Ctrl+I
*DtHelpDialogWidget*searchMenu.keyword.accelerator:  Ctrl<Key>i
*DtHelpDialogWidget*navigateMenu.backTrack.acceleratorText: Ctrl+B
*DtHelpDialogWidget*navigateMenu.backTrack.accelerator:  Ctrl<Key>b
*DtHelpDialogWidget*navigateMenu.homeTopic.acceleratorText: Ctrl+H
*DtHelpDialogWidget*navigateMenu.homeTopic.accelerator:  Ctrl<Key>h
*DtHelpDialogWidget*fileMenu.close.acceleratorText:  Alt+F4
*DtHelpDialogWidget*fileMenu.close.accelerator:  Alt<Key>f4
```

変数の値

この節では、dtksh app-defaults ファイル内のいくつかの変数の値の型について説明します。

定義値

X、Xt および Motif のインタフェースの C バインディングにおいて、文字列でない値（ヘッダ・ファイルで定義されている）が多く見られます。このような値の一般的な形式は、Xt または Xm の接頭辞の後に説明的な名前が続く形式になっています。たとえば、フォーム・ウィジェットの子の制約値の 1 つとして、XmATTACH_FORM があります。dtksh では、接頭辞を取り去ったものが同等の値になり、Motif のデフォルト・ファイルで指定する値と全く同じです。

- XmDIALOG_COMMAND_TEXT は DIALOG_COMMAND_TEXT になります。
- XtATTACH_FORM は ATTACH_FORM になります。

ブール値

dtksh コマンドへのパラメータとして、True または False の語（大文字小文字は関係ありません）を使用してブール値を指定できます。結果としてのブール値は、true または false のいずれか（すべて小文字を使用）が返されます。

戻り値

dtksh のグラフィカル・コマンドは、対応する C 関数の定義に基づいて、次の 4 つのカテゴリに分類されます。

1. 関数は Void で、値を返しません。例: XtMapWidget()
2. 関数は Void ですが、関連するパラメータを介して 1 つ以上の値を返します。
例: XmGetColors()
3. 関数は、非ブール値を返します。例: XtCreateManagedWidget()
4. 関数は、ブール値を返します。例: XtIsSensitive()

カテゴリ 1

dtksh のカテゴリ 1 のコマンドは、対応する C 関数の呼び出しシーケンスに従います。パラメータの数や順序に関しては、関数の標準マニュアルを参照してください。

例:

```
XtMapWidget $FORM
```

カテゴリ 2

dtksh のカテゴリ 2 のコマンドも、一般的に対応する C 関数の呼び出しシーケンスに従います。変数へのポインタを渡す代わりに、環境変数の値を返します。

例:

```
XmGetColors $FORM $BG FOREGROUND TOPSHADOW BOTTOMSHADOW SELECT  
echo "Foreground color = " $FOREGROUND
```

カテゴリ 3

dtksh のカテゴリ 3 のコマンドは、対応する C 関数とは多少異なります。C 関数がある値をプロシージャ・コールの値として返すのに対して、dtksh コマンドは追加のパラメータを要求します。このパラメータは、戻り値として置かれている環境変数の名前で、常に最初のパラメータになります。

例:

```
XmTextGetString TEXT_VALUE $TEXT_WIDGET  
echo "The value of the text field is " $TEXT_VALUE
```

カテゴリ 4

dtksh のカテゴリ 4 のコマンドは、C の場合と全く同様の条件付き式で利用できる値を返します。C 関数がリファレンス変数 (カテゴリ 2 と同じ) を介して値を返すと、dtksh コマンドもまた、対応するパラメータの変数名を使用します。

例:

```
if XmIsTraversable $PUSH_BUTTON; then
echo "The pushbutton is traversable"
else
echo "The pushbutton is not traversable"
fi
```

一般に、コマンドに渡されるパラメータの順序と型は、対応する C 関数に渡されるそれと一致します。ただし、カテゴリ 3 のコマンドのところで記述したものは例外です。

即時戻り値

カテゴリ 3 のコマンドの多くは、コマンドへの最初のパラメータとして指定する環境変数を使用して 1 つの値を返します (これらの特殊なコマンドに対する最初のパラメータは、*variable* という名前です)。この戻り値がすぐに式で使用されると、変数名の代わりに、特殊環境変数 "-" が使用される場合があります。dtksh が、戻り値が返される環境変数の名前として "-" を認識すると、代わりにコマンドの値としてその結果を返します。これによりシェル・スクリプトは、そのコマンド呼び出しを別のコマンド呼び出しに埋め込むことができます。この機能は、単一の値を返すコマンドにだけ働き、その値は最初のパラメータに返されます。たとえば、次のとおりです。

```
XtDisplay DISPLAY $FORM
XSync $DISPLAY true
```

上記は、次のように置換できます。

```
XSync $(XtDisplay "-" $FORM) true
```

\$DISPLAY のリファレンスは、XtDisplay の呼び出しが返す値で置換されます。

これは、次の例外を除くすべてのカテゴリ 3 のコマンドに有効です。例外となるのは、ウィジェットを作成するコマンド、複数の値を返すコマンド、および最初のパラメータが名前の付いていない変数であるコマンドです。 "-" を環境変数名として受け入れないコマンドには、次のようなものがあります。

- XtInitialize()
- XCreateApplicationShell()
- XCreatePopupShell()
- XCreateManagedWidget()
- XCreateWidget()

- 次の形式のすべてのコマンド
XmCreate...()
- 次の形式のほとんどのコマンド
tt_...()

Xt イントリンシクスの初期化

dtksh スクリプトは、Xlib、Xt、Motif または libDt コマンドのいずれかを呼び出す前に、まず Xt イントリンシクスを初期化しなければなりません。これは、XtInitialize コマンドを呼び出して行います。このコマンドは、アプリケーション・シェル・ウィジェットを返します。ウィジェット ID を返すすべての dtksh コマンドが真であるように、XtInitialize は、最初の引き数である環境変数にそのウィジェット ID を返します。

たとえば、

```
XtInitialize TOPLEVEL myShellName Dtksh $0 "$@"
```

とすると、ウィジェット ID は環境変数 TOPLEVEL に返されます。

dtksh はデフォルトの app-defaults ファイルを提供し、このファイルは、XtInitialize の呼び出しで Dtksh のアプリケーション・クラス名を指定した場合に使用されます。また、この app-defaults ファイルには、デスクトップ・アプリケーションのデフォルト値の標準セットが格納されているので、dtksh アプリケーションと他のデスクトップ・アプリケーションとの外見的な一貫性を保つことができます。

ウィジェットの作成

ウィジェット作成に使用できるいくつかのコマンドは、次のとおりです。

XtCreateWidget	管理されないウィジェットを作成します。
XtCreateManagedWidget	管理されるウィジェットを作成します。
XtCreateApplicationShell	アプリケーション・シェルを作成します。
XtCreatePopupShell	ポップアップ・シェルを作成します。
XmCreate<widgettypes>	管理されないウィジェットを作成します。

これらのコマンドにはそれぞれ、従わなければならない特定の形式があります。たとえば、管理されないプッシュ・ボタン・ウィジェットをトップレベル・ウィジェットの子として作成したいと仮定します。XtCreateWidget か XmCreatePushButton のどちらかを使用できます。これらのコマンドの形式は、次のとおりです。

- XtCreateWidget *variable name widgetclass \$parent [resource:value ...]*
- XmCreatePushButton *variable \$parent name [resource:value ...]*

プッシュ・ボタン・ウィジェットを作成する実際のコマンドは、次のとおりです。

```
XtCreateWidget BUTTON button XmPushButton $TOPLEVEL
XmCreatePushButton BUTTON $TOPLEVEL button
```

上記のそれぞれのコマンドは、全く同じ動作（管理されないプッシュ・ボタンの作成）をします。リソース値は設定されていないことに注意してください。プッシュ・ボタンのバックグラウンドの色を赤、フォアグラウンドの色を黒にしたいと仮定します。これらのリソース値を次のように設定できます。

```
XtCreateWidget BUTTON button XmPushButton $TOPLEVEL \
background:Red \
foreground:Black
XmCreatePushButton BUTTON $TOPLEVEL button\
background:Red \
foreground:Black
```

ウィジェットを作成する C 関数のすべてが、ウィジェット ID、または ID を返します。対応する dtksh コマンドは、ウィジェット ID と同じ環境変数を設定します。これらはカテゴリ 3 コマンドで、その最初の引き数は、ウィジェット ID が戻る環境変数の名前です。ウィジェット ID は、dtksh が実際のウィジェット・ポインタにアクセスするために使用する ASCII 文字です。次のどちらかのコマンドで、新規フォーム・ウィジェットを作成できます。ただし、どちらの場合も、新規フォーム・ウィジェットのウィジェット ID が環境変数 FORM に返されます。

- XtCreateManagedWidget FORM name XmForm \$PARENT
- XmCreateForm FORM \$PARENT name

どちらかのコマンドを実行した後、\$FORM を使用して新規フォーム・ウィジェットを参照できます。たとえば、次のコマンドを使用して、新規フォーム・ウィジェット内でラベル・ウィジェットを作成できます。

```
XmCreateLabel LABEL $FORM name \  
labelString:"Hi Mom" \  
CH_FORM \  
leftAttachment:ATTACH_FORM
```

注 – NULL と呼ばれる特別なウィジェット ID があります。これは、シェル・スクリプトに NULL のウィジェットの指定が必要である場合に使用します。たとえば、フォーム・ウィジェットの defaultButton リソースを無効にする場合には、コマンド XtSetValues \$FORM defaultButton:NULL を使用してください。

コールバックの使い方

コールバックは、1 つのイベント、またはいくつか組み合わさったイベントが発生した場合に実行される関数またはプロシージャです。たとえば、コールバックは、プッシュ・ボタンが押された時、それによって要求される結果を出すために使用されます。dtksh シェル・スクリプトにとって、特定のコールバックがウィジェットに呼び出される時に、必ず起動するようにコマンドを割り当てることは容易です。そのコマンドは、共にブロックされるコマンドの文字列、または呼び出すシェル関数の名前と同じくらい単純です。

コールバックの登録

アプリケーションは、関係のある条件を指定するウィジェット、およびその条件が発生した時に起こるアクションを指定するウィジェットでコールバックを登録します。コールバックは、XtAddCallback を使用して登録します。登録するアクションは、任意の有効な dtksh コマンドになります。たとえば、次のとおりです。

```
XtAddCallback $WIDGET activateCallback "ActivateProc"  
XtAddCallback $WIDGET activateCallback \  
    "XtSetSensitive $BUTTON false"
```

データをコールバックへ渡す

コールバックに渡されるのはコンテキスト情報で、呼び出しまでの条件を決定します。C プロシージャの場合、この情報は一般的に `callData` 構造体へ渡されます。たとえば、`valueChangedCallback` を呼び出すスケール・ウィジェットは、次のような構造を `callData` に渡します。

```
typedef struct {
    int reason;
    XEvent event;
    int value;
}XmScaleCallbackStruct;
```

C アプリケーションのコールバックは、次のように動作します。

```
if (scaleCallData->reason == XmCR_VALUE_CHANGED)
{
    eventType = scaleCallData->event->type;
    display = scaleCallData->event->xany.display;
}
```

同様に、コールバックが `dtksh` で呼び出された場合、それが実行する前に次の特別な環境変数が設定されます。

`CB_WIDGET`

これは、コールバックを呼び出しているウィジェットのウィジェット ID に対して設定されます。

`CB_CALL_DATA`

これは、ウィジェットがコールバックに渡す `callData` 構造体のアドレスに対して設定されます。

`CB_CALL_DATA` 環境変数は構造体へのポインタを表し、そのフィールドへのアクセスには、C のそれと同様の構文を使用します。入れ子形式で環境変数が定義され、構造体のフィールドと同じ名前が付けられます（ただし、すべて大文字です）。この時、構造体の要素の内容を示すためにドットが使用されます。こうして、スケール・ウィジェットによって提供される `callData` へアクセスする前述の C コードは、次のように翻訳されます。

```
if [ ${CB_CALL_DATA.REASON} = "CR_VALUE_CHANGED" ]; then
    eventType=${CB_CALL_DATA.EVENT.TYPE}
    display=${CB_CALL_DATA.EVENT.XANY.DISPLAY}
fi
```


同様のことが、callData 構造体内のイベント構造体の場合にも言えます。

ほとんどのコールバック構造体において、シェル・スクリプトは、特定のコールバック構造体に定義される任意のフィールドを先に記述した手法を使用して参照できます。たいていの場合、シェル・スクリプトは、これらの構造体内のフィールドを変更できません。この例外として XmTextVerifyCallbackStruct があり、テキスト・ウィジェットの losingFocusCallback、modifyVerifyCallback、および motionVerify Callback 中では変更できます。

dtksh は、Motif でサポートしている範囲内において、この構造体内のフィールドの変更をサポートしています。コールバック構造体内の次のフィールドは、変更ができます。

- CB_CALL_DATA.DOIT
- CB_CALL_DATA.STARTPOS
- CB_CALL_DATA.TEXT.PTR
- CB_CALL_DATA.TEXT.LENGTH
- CB_CALL_DATA.TEXT.FORMAT

次は、上記のフィールドの変更例です。

- CB_CALL_DATA.DOIT="false"
- CB_CALL_DATA.TEXT.PTR="*"
- CB_CALL_DATA.TEXT.LENGTH=1

サンプル・スクリプト

2 

この章では、第 1 章で学んだ dtksh の使用方法を説明します。ここで説明する 2 つの簡単なスクリプトは、独自のスクリプトを記述する際に参考に使ってください。

スクリプトの記述

このスクリプトは、内部にプッシュ・ボタン・ウィジェットを持つブリテン・ボード・ウィジェットを作成します。コールバックが 1 つもない単純なスクリプトです。2 番目のスクリプトはコールバックを取り込んでいます。

最初のスクリプトは、次のとおりです。

```
#!/usr/dt/bin/dtksh
XtInitialize TOPLEVEL dttest1 Dtksh $0
XtSetValues $TOPLEVEL title:"dttest1"
XtCreateManagedWidget BBOARD bboard XmBulletinBoard $TOPLEVEL \
    resizePolicy:RESIZE_NONE height:150 width:250 \
    background:SkyBlue
XtCreateManagedWidget BUTTON pushbutton XmPushButton $BBOARD \
    background:goldenrod \
    foreground:MidnightBlue \
    labelString:"Push Here" \
    height:30 width:100 x:75 y:60 shadowThickness:3
XtRealizeWidget $TOPLEVEL
XtMainLoop
```

最初のスクリプトによって作成されるウィンドウを図 2-1 に示します。



図 2-1 スクリプト dttest のウィンドウ

スクリプトの最初の行は次のとおりです。

```
#!/usr/dt/bin/dtksh
```

この行は、標準シェルではなく /usr/dt/bin/dtksh を使用してスクリプトを実行することをオペレーティング・システムに指示しています。

次の行で、Xt インタリックスを初期化します。

```
XtInitialize TOPLEVEL dttest1 Dtksh $0
```

環境変数 \$TOPLEVEL にトップレベルのウィジェット名が保存され、dttest1 はシェル・ウィジェット名、Dtksh はアプリケーション・クラス名で、dtksh 変数 \$0 によってアプリケーション名が指定される、ということを表しています。

次の行で、タイトル・リソースにスクリプト名を設定します。

```
XtSetValues $TOPLEVEL title:"dttest1"
```

リソース名 (title) の後のコロンとリソース値の間にはスペースは入れられないので注意してください。スペースを入れると、エラー・メッセージが表示されます。

次の 4 行で、ブリテン・ボード・ウィジェットを作成し、いくつかのリソースを設定します。

```
XtCreateManagedWidget BBOARD bboard XmBulletinBoard $TOPLEVEL \
    resizePolicy:RESIZE_NONE \
    background:SkyBlue \
    height:150 width:250
```

環境変数 \$BBOARD にブリテン・ボード・ウィジェットの ID が保存されます。bboard はウィジェット名です。この名前は、Xt イントリンシクスが、外部のリソース・ファイルでリソース値を設定するのに使います。ウィジェット・クラスは XmBulletinBoard です。ブリテン・ボードの親ウィジェットは、環境変数 \$TOPLEVEL に格納されているウィジェット ID です。

これは、一番最初の初期化コマンドで作成したトップレベルのウィジェットです。行の終わりの \ (バックスラッシュ) は、そのコマンドが次の行へも続いているということを dtksh に指示しています。

次の 6 行で、プッシュ・ボタン・ウィジェットをブリテン・ボードの子として作成し、いくつかのプッシュ・ボタンのリソースを設定します。

```
XtCreateManagedWidget BUTTON pushbutton XmPushButton $BBOARD \  
  background:goldenrod \  
  foreground:MidnightBlue \  
  labelString:"Push Here" \  
  height:30 width:100 x:75 y:60 \  
  shadowThickness:3
```

これは、変数、名前、クラス、および親が異なるという点を除いて、ブリテン・ボードを作成するのに使用するプロシージャと基本的に同じです。

次の行で、トップレベル・ウィジェットとそのすべての子を認識します。

```
XtRealizeWidget $TOPLEVEL
```

最後に、XtMainLoop コマンドがウィジェットのイベントのループ処理を開始します。

```
XtMainLoop
```

このスクリプトを実行すると、ディスプレイ上にウィンドウが表示されます。スクリプトを終了するまで表示されたままの状態です。終了する方法は、[ウィンドウ・マネージャ]メニューで [閉じる] を選択するか、スクリプトを実行した端末エミュレータ・ウィンドウ内で [CTRL] + [C] キーを押すかのどちらかです。

コールバックの追加

ボタンが押されると端末エミュレータにメッセージが表示され、スクリプトが終了するというプッシュ・ボタン関数を作成するには、コールバックを追加します。また、このコールバックの存在をプッシュ・ボタンに伝える必要もあります。新規コードが追加されたスクリプトは、次のとおりです。

```
#!/usr/dt/bin/dtksh

activateCB() {
    echo "Pushbutton activated; normal termination."
    exit 0
}

XtInitialize TOPLEVEL dttest2 Dtksh $0
XtSetValues $TOPLEVEL title:"dttest2"
XtCreateManagedWidget BBOARD bboard XmBulletinBoard $TOPLEVEL \
    resizePolicy:RESIZE_NONE \
    background:SkyBlue \
    height:150 width:250
XtCreateManagedWidget BUTTON pushbutton XmPushButton $BBOARD \
    background:goldenrod \
    foreground:MidnightBlue \
    labelString:"Push Here" \
    height:30 width:100 x:75 y:60 shadowThickness:3

XtAddCallback $BUTTON activateCallback activateCB
XtRealizeWidget $TOPLEVEL
XtMainLoop
```

関数 `activateCB()` がコールバックです。通常は、次のようにプッシュ・ボタンが作成された後に、コールバックをプッシュ・ボタンに追加します。

```
XtAddCallback $BUTTON activateCallback activateCB
```

これで、コールバックはプッシュ・ボタンに認識されました。プッシュ・ボタンをクリックすると、関数 `activateCB()` は実行され、スクリプトを実行した端末エミュレータにメッセージ「Pushbutton activated; normal termination.」が表示されます。スクリプトは、関数 `exit 0` を呼び出して終了します。

前章までで dtksh に関する基本的な知識を学びました。この章では、より高度なトピックを紹介します。

コンテキスト変数の使い方

dtksh には、アプリケーションのいろいろな状態に対するコンテキストを提供する数多くの変数があります。

イベント・ハンドラ・コンテキスト変数

アプリケーションは、指定したイベントの 1 つが発生した時に起こるアクションを指定するウィジェットにイベント・ハンドラを登録します。アクションには、任意の dtksh コマンド行を指定できます。たとえば、次のとおりです。

```
XtAddEventHandler $W "Button2MotionMask" false "ActivateProc"
XtAddEventHandler $W "ButtonPressMask|ButtonReleaseMask" \
    false "echo action"
```

2 つの環境変数が、イベント・ハンドラへのコンテキストを提供するために次のように定義されます。

EH_WIDGET	イベント・ハンドラが登録されるウィジェットの ID を設定します。
EH_EVENT	イベント・ハンドラを起動する XEvent のアドレスを設定します。

XEvent 構造体内のフィールドへのアクセスを次の例に示します。

```
if [ ${EH_EVENT.TYPE} = "ButtonPress" ]; then
    echo "X = ${EH_EVENT.XBUTTON.X}"
    echo "Y = ${EH_EVENT.XBUTTON.Y}"
elif [ ${EH_EVENT.TYPE} = "KeyPress" ]; then
    echo "X = ${EH_EVENT.XKEY.X}"
    echo "Y = ${EH_EVENT.XKEY.Y}"
fi
```

トランスレーション・コンテキスト変数

Xt インtrinsic は、ウィジェットに対して登録されるイベントのトランスレーションを提供します。イベントのトランスレーション・コンテキストは、イベント・ハンドラの場合と同じ方法で提供されます。トランスレーション・コマンドで定義される 2 つの変数は次のとおりです。

TRANSLATION_WIDGET	トランスレーションが登録されるウィジェットのウィジェット・ハンドルを設定します。
TRANSLATION_EVENT	トランスレーションを起動する XEvent のアドレスを設定します。

次のようなドット表記で、イベントのフィールドへアクセスします。

```
echo "Event type = ${TRANSLATION_EVENT.TYPE}"
echo "Display = ${TRANSLATION_EVENT.XANY.DISPLAY}"
```

ワークスペース・コールバック・コンテキスト変数

アプリケーションは、ユーザが新規ワークスペースへ移動するたびに呼び出せるコールバック関数を登録できます。コールバックが呼び出されると、次に挙げる 2 つの特別な環境変数が設定され、これらはシェル・コールバック・コードによってアクセス可能です。

CB_WIDGET	コールバックを呼び出すウィジェットの ID を設定します。
CB_CALL_DATA	新規ワークスペースを一意に識別する X アトムを設定します。これは、XmGetAtomName コマンドを使用して文字列表現に変換できます。

入力コンテキスト変数

Xt インタプリタは、XtAddInput 機能を提供します。これにより、アプリケーションは、特定のファイル記述子から利用可能なデータの配信対象を登録できます。C 言語でプログラミングする場合、アプリケーションはハンドラ関数を提供します。この関数は、入力がある場合に呼び出されます。入力ソースからデータを読み込んだり、エスケープ文字や継続行を処理するのは、ハンドラの役割です。

dtksh は XtAddInput 機能もサポートしますが、シェル・プログラマがさらに上達して、簡単に使用できるようにします。デフォルトでは、シェル・スクリプトがファイル記述子の配信対象を登録する場合、dtksh は、テキストの絶対行を受け取った時のみシェル・スクリプトの入力ハンドラを呼び出します。テキストの絶対行は、エスケープされていない改行文字またはファイルの終わりにによって終了している行と定義されます。入力ハンドラは、利用可能なデータが存在せず、ファイルの終わりに達した場合にも呼び出されます。この時、ハンドラは、XtRemoveInput を使用して入力ソースを削除し、ファイル記述子を閉じることができます。このデフォルト動作の利点は、入力ハンドラがエスケープ処理や継続行の処理にかかわらず動作する点です。不便な点は、すべての入力が行単位であり、バイナリ情報を含んでいないことを前提にしている点です。

dtksh はまた、入力ソースにバイナリ情報がある場合や、入力ハンドラがデータを入力ソースから直接読み込みたい場合に、「raw」入力モードをサポートします。raw モードでは、dtksh は、入力ソースからデータを全く読み込みません。dtksh は入力が入力ソース上で行えるようになると、シェル・スクリプトの入力ハンドラを呼び出します。この時、入力データを読み込んだり、バッファリング要求やエスケープ処理を行ったり、いつファイルの終わりに到達したかを検出したりするのは、ハンドラの責任において行われます（これにより、入力ソースが削除され、ファイル記述子を閉じます）。このモードは、dtksh スクリプトで使用されることはめったにありません。

入力ハンドラがデフォルト・モードまたは raw モードのどちらで動作するように構成されているかによって、シェル・スクリプトの入力ハンドラを呼び出す前に、dtksh がいくつかの環境変数を設定します。これらの環境変数は、入力データを処理するのに必要なすべてのものを入力ハンドラに提供します。その環境変数は次のとおりです。

INPUT_LINE	デフォルト・モードで動作する場合、この変数には入力ソースで利用可能な次の完全入力行が格納されます。INPUT_EOF が true の場合、このバッファにはデータはありません。raw モードで動作する場合、この変数には常に空の文字列が格納されます。
------------	--

- INPUT_EOF** デフォルト・モードで動作する場合、この変数は、INPUT_LINE にデータが格納されている限り false が設定され、ファイルの終わりに到達すると、true が設定されます。ファイルの終わりに到達すると、シェル・スクリプトの入力ハンドラは、入力ソースを登録解除し、ファイル記述子を閉じなければなりません。raw モードの場合、この変数は常に false が設定されます。
- INPUT_SOURCE** これは、入力可能なファイル記述子を示します。raw モードで動作する場合、このファイル記述子は、保留になっている入力を獲得するために使用されます。ファイル記述子はまた、必要がなくなった時に入力ソースを閉じるためにも使用されます。
- INPUT_ID** これは、入力ソースがもともと登録されている場合に XtAddInput が返す ID を示します。この情報は、XtRemoveInput で入力ソースを削除するのに必要です。

イベント・サブフィールドへのアクセス

XEvent 構造体には、イベントの型に基づいて異なる構成が数多くあります。dtksh は、もっとも頻繁に使用される XEvents へのアクセスのみを提供します。他の標準的なXEvents へは、次のサブフィールドが含まれているイベント型 XANY (XANY イベント構造体によって定義されるサブフィールドが続いている) を使用してアクセスできます。

- `$_TRANSLATION_EVENT.XANY.TYPE`
- `$_TRANSLATION_EVENT.XANY.SERIAL`
- `$_TRANSLATION_EVENT.XANY.SEND_EVENT`
- `$_TRANSLATION_EVENT.XANY.DISPLAY`
- `$_TRANSLATION_EVENT.XANY.WINDOW`

dtksh は、次のイベント型のすべてのイベント・フィールドへのアクセスを完璧にサポートします。

- XANY
- XBUTTON
- XEXPOSE
- XNOEXPOSE
- XGRAPHICSEXPOSE

- XKEY
- XMOTION

次の例は、前述のイベント型のサブフィールドへのアクセスの仕方を示しています。

```
{TRANSLATION_EVENT.XBUTTON.X}
{CB_CALL_DATA.EVENT.XKEY.STATE}
{EH_EVENT.XGRAPHICSEXPOSE.WIDTH}
```

ウィンドウ・マネージャのクローズ通知への応答

ユーザがアプリケーションに対して [ウィンドウ・マネージャ]メニューで [閉じる] を選択した場合、アプリケーションが、クローズ通知を「取り込む」準備ができていないと終了してしまいます。アプリケーションが通知を取り込まないと、アプリケーションが管理する複数のウィンドウはすべて消え、アプリケーション・データは、望ましくない状態で残る場合があります。これを避けるために、`dtksh` を使用して、クローズ通知を受け取り処理することができます。アプリケーションは、次のことを行わなければなりません。

- クローズ通知を処理するプロシージャを定義する。
- [閉じる] が選択された場合に通知を要求する。
- アプリケーションをシャットダウンしないように応答を無効にする。

次のコードは、この処理を説明しています。

```
# This is the 'callback' invoked when the user selects
# the 'Close' menu item
WMCallback()
{
echo "User has selected the Close menu item"
}
# Create the toplevel application shell
XtInitialize TOPLEVEL test Dtksh $0 "$@"
XtDisplay DISPLAY $TOPLEVEL

# Request notification when the user selects the 'Close'
# menu item
XmInternAtom DELETE_ATOM $DISPLAY "WM_DELETE_WINDOW" false
XmAddWMProtocolCallback $TOPLEVEL $DELETE_ATOM "WMCallback"

# Ask Motif to not automatically close down your
# application window
XtSetValues $TOPLEVEL deleteResponse:DO_NOTHING
```

セッション・マネージャの保存状態通知への応答

セッション・マネージャは、ユーザが現在のセッションを終了する時の状態をアプリケーションに保存させます。これによって、アプリケーションは、ユーザがセッションを再起動した時に、前回終了時の状態へ戻ることができます。

dtksh でこれを行うには、クローズ通知を処理する場合と同様の方法でハンドラを設定します。ハンドラを設定しないと、新規セッションでアプリケーションを手動で再起動しなければなりません。その場合、アプリケーションはいかなる状態も保持しません。

ハンドラを設定して現在の状態を保存するためには、アプリケーションで次のことを行います。

- セッションの終わりの状態を保存し、起動時にそれを復元するための関数を定義する。
- セッション・マネージャの通知の配信対象を登録する。
- 状態を保存する関数を登録する。
- 起動時に、保存した状態を復元するか否かを決定する。

次のコードは、この処理を説明しています。

```
#!/usr/bin/dtksh
# Function invoked when the session is being ended by the user
SessionCallback()
{
    # Get the name of the file into which we should save our
    # session information
    if DtSessionSavePath $TOPLEVEL PATH SAVEFILE; then
        exec 9>$PATH

        # Save off whether we are currently in an iconified state
        if DtShellIsIconified $TOPLEVEL ; then
            print -u9 'Iconified'
        else
            print -u9 'Deiconified'
        fi

        # Save off the list of workspaces we currently reside in
        if DtWsmGetWorkspacesOccupied $(XtDisplay "-" $TOPLEVEL) \
            $(XtWindow "-" $TOPLEVEL) \
            CURRENT_WS_LIST ;
        then
            # Map the comma-separated list of atoms into
            # their string representation
```

```
oldIFS=$IFS
IFS=","
for item in $CURRENT_WS_LIST;
do
    XmGetAtomName NAME $(XtDisplay "-" $TOPLEVEL) \
        $item
    print -u9 $NAME
done
IFS=$oldIFS
fi

exec 9<&-

# Let the session manager know how to invoke us when
# the session is restored
DtSetStartupCommand $TOPLEVEL \
    "/usr/dt/contrib/dtksh/SessionTest $SAVEFILE"
else
    echo "DtSessionSavePath FAILED!!"
    exit -3
fi
}

# Function invoked during a restore session; restores the
# application to its previous state
RestoreSession()
{
    # Retrieve the path where our session file resides
    if DtSessionRestorePath $TOPLEVEL PATH $1; then
        exec 9<$PATH
        read -u9 ICONIFY

        # Extract and restore our iconified state
        case $ICONIFY in
            Iconified) DtSetIconifyHint $TOPLEVEL True;;
            *) DtSetIconifyHint $TOPLEVEL False;
        esac

        # Extract the list of workspaces we belong in, convert
        # them to atoms, and ask the Workspace Manager to relocate
        # us to those workspaces
        WS_LIST=""
    fi
}
```

```

while read -u9 NAME
do
    XmInternAtom ATOM $(XtDisplay "-" $TOPLEVEL) \
        $NAME False
    if [ ${#WS_LIST} -gt 0 ]; then
        WS_LIST=$WS_LIST,$ATOM
    else
        WS_LIST=$ATOM
    fi
done

DtWsmSetWorkspacesOccupied $(XtDisplay "-" $TOPLEVEL) \
    $(XtWindow "-" $TOPLEVEL) $WS_LIST

exec 9<&-
else
    echo "DtSessionRestorePath FAILED!!"
    exit -3
fi
}
##### Create the Main UI #####
XtInitialize TOPLEVEL wmProtTest Dtksh $0 "$@"
XtCreateManagedWidget DA da XmDrawingArea $TOPLEVEL \
    height:200 width:200
XmInternAtom SAVE_SESSION_ATOM $(XtDisplay "-" $TOPLEVEL) \
    "WM_SAVE_YOURSELF" False

# If a command-line argument was supplied, then treat it as the
# name of the session file
if (( $# > 0 ))
then
    # Restore to the state specified in the passed-in session file
    XtSetValues TOPLEVEL mappedWhenManaged:False
    XtRealizeWidget TOPLEVEL
    XSync $(XtDisplay "-" $TOPLEVEL) False
    RestoreSession $1
    XtSetValues TOPLEVEL mappedWhenManaged:True
    XtPopup TOPLEVEL GrabNone
else
    # This is not a session restore, so come up in the default state
    XtRealizeWidget TOPLEVEL
    XSync $(XtDisplay "-" $TOPLEVEL) False

```

```

fi

# Register the fact that we are interested in participating in
# session management
XmAddWMProtocols $TOPLEVEL $SAVE_SESSION_ATOM
XmAddWMProtocolCallback $TOPLEVEL $SAVE_SESSION_ATOM \
    SessionCallback

XtMainLoop

```

ワークスペース・マネージャとの協調

dtksh は、Dt ライブラリの主要なワークスペース・マネージャ関数のすべてにアクセス可能です。たとえば、アプリケーションと関連のあるワークスペースのセットを問い合わせたり設定する関数、全ワークスペースのリストを問い合わせる関数、現在のワークスペースを問い合わせたり設定する関数、およびユーザが別のワークスペースへ変更する時は常にアプリケーションに通知を要求する関数などです。

ユーザの側から見ると、ワークスペースは名前のセットで識別されますが、ワークスペース・マネージャの側から見ると、ワークスペースは X アトムで識別されます。シェル・スクリプトがワークスペース識別子のリストを要求すると、常に X アトムの文字列が返ります。複数の X アトムがある場合、リストはカンマで区切られます。ワークスペース・マネージャは、そこへワークスペース識別子を戻す場合と同じ形式をシェル・スクリプトが使用することを期待します。指定されたセッションの間のシェル・スクリプトによる X アトムの処理は安全です。そのセッション中に、X アトムの値が変わるということはないからです。しかし、前節におけるセッション・マネージャのシェル・スクリプト例で示したように、シェル・スクリプトがワークスペース識別子の保存および復元を行おうとする場合、保存する前に識別子を X アトム表現から文字列へ変換しなければなりません。そして、セッションを復元する場合、シェル・スクリプトは、情報をワークスペース・マネージャに渡す前に、名前を X アトムに再マッピングしなければなりません。X アトムと文字列間のマッピング、文字列と X アトム間のマッピングは、次の 2 つのコマンドを使用して行われます。

- `XmInternAtom ATOM $DISPLAY $WORKSPACE_NAME false`
- `XmGetAtomName NAME $DISPLAY $ATOM`

ワークスペース管理を扱う特定の dtksh コマンドは、付録 A「dtksh コマンド」の「組み込み libDt セッション管理コマンド」で詳述しています。

ローカライズされたシェル・スクリプトの作成

dtksh スクリプトは、C アプリケーションに似たプロセスで、国際化対応とローカライズを行います。ユーザに表示されるすべての文字列は、スクリプト内で識別されます。ポストプロセッサは、スクリプトから文字列を抽出し、カタログを作成します。これは、要求されたロケールに翻訳できます。スクリプトを実行すると、現在のロケールは、表示文字列を捜すためのメッセージ・カタログを決定します。

文字列が表示される時、その文字列はセット内のメッセージ・セット ID (カタログに対応) とメッセージ番号で識別されます。これらの値は、ユーザに見えるテキストを決定します。次のコードで、そのプロセスを説明します。

```
# Attempt to open our message catalog
catopen MSG_CAT_ID "myCatalog.cat"

# The localized button label is in set 1, and is message #2
XtCreatePushButton OK $PARENT ok \
  labelString:$(catgets $MSG_CAT_ID 1 2 "OK")
# The localized button label is in set 1, and is message #3
XtCreatePushButton CANCEL $PARENT cancel \
  labelString:$(catgets $MSG_CAT_ID 1 3 "Cancel")

# Close the message catalog, when no longer needed
catclose $MSG_CAT_ID
```

catopen によって返されるファイル記述子は、kshell の exec コマンドではなく、catclose を使用して閉じなければならないということに注意してください。

dtksh を使用して X 描画関数 へアクセスする

dtksh コマンド群には、線、点、線分、矩形、弧、および多角形を描く標準的な Xlib 描画関数が含まれています。標準 C プログラミング環境では、これらの関数はグラフィックス・コンテキスト (GC) を描画データとしてだけでなく、引き数と見なします。dtksh の描画関数では、GC オプションの集合は、パラメータ・リスト内でコマンドに対して指定されます。

デフォルトでは、描画コマンドは、特定のコマンドに使用された後に捨てられる GC を作成します。スクリプトが -gc オプションを指定すると、グラフィックス・コンテキスト・オブジェクトの名前をコマンドへ渡すことができます。この GC は、コマンドの解釈に使用され、変数は、コマンドが行う GC の変更によって更新されます。

<code>-gc <GC></code>	<GC> は、初期化されていないか、または前の描画コマンドによるグラフィック・コンテキストを保持したままの環境変数の名前です。このオプションを指定する場合、まず GC オプションを指定しなければなりません。
<code>-foreground <color></code>	カラー名またはピクセル番号で行うフォアグラウンド・カラーの指定です。
<code>-background <color></code>	カラー名またはピクセル番号で行うバックグラウンド・カラーの指定です。
<code>-font </code>	使用されるフォント名の指定です。
<code>-line_width <number></code>	描画中に使用される行の幅の指定です。
<code>-function <drawing function></code>	描画関数、xor、or、clear、and、copy、noop、nor、nand、set、invert、equiv、andReverse、orReverse、または copyInverted などの指定です。
<code>-line_style <style></code>	線の形状、LineSolid、LineDoubleDash、または LineOnOffDash の指定です。

ウィジェット・トランスレーションの設定

dtksh は、C プログラミング環境において、ウィジェット・トランスレーションを増加、無効、削除する機能を提供します。C では、アプリケーションは、トランスレーションのアクション・プロシージャのセットをインストールします。このセットは、特定のイベント・シーケンスに接続できます (トランスレーションは、イベント・シーケンスと関連アクション・プロシージャで構成されます)。dtksh 内のトランスレーションは、単一のアクション・プロシージャだけでも使用できる点を除いて、同様の方法で処理されます。ksh_eval と呼ばれるこのアクション・プロシージャは、渡されるパラメータを dtksh コマンドとして解釈し、トランスレーション開始時に評価します。次のシェル・スクリプトでは、トランスレーションの使用法例を示しています。

```
BtnDownProcedure()
{
    echo "Button Down event occurred in button "$1
}
```

```
XtCreateManagedWidget BUTTON1 button1 XmPushButton $PARENT \  
    labelString:"Button 1" \  
    translations:#augment  
        <EnterNotify>:ksh_eval("echo Button1 entered")  
        <Btn1Down>:ksh_eval("BtnDownProcedure 1")'  
XtCreateManagedWidget BUTTON2 button2 XmPushButton $PARENT \  
    labelString:"Button 2"  
XtOverrideTranslations $BUTTON2 \  
    '#override  
    <Btn1Down>:ksh_eval("BtnDownProcedure 2")'
```

この章では、第 2 章で説明したものよりもさらに複雑なスクリプトについて説明します。非常に長いので、スクリプト全体の一覧は付録 C に掲載します。このマニュアルは、Korn シェル・プログラミングのチュートリアルではないので注意してください。Korn シェル・プログラミングについてあまり理解していない場合は、Korn シェル・プログラミングのマニュアルを取り寄せて、参照してください。

script_find の使い方

スクリプト `script_find` は、`dtksh` を使用して `find` コマンドに対するグラフィカル・インタフェースを提供する方法について図示します。`script_find` は、`find` コマンドのパラメータを指定できるウィンドウを作成します。スクリプトに関して完全に理解するには、`find` コマンドをよく理解し、マニュアル・ページを使用可能にしておく必要があります。`script_find` によって作成されたウィンドウにある多数のトグル・ボタン・メニュー選択には、`find` コマンドについての知識が多少必要です。

スクリプトのウィンドウで、検索ディレクトリとファイル名を指定できます。他のオプションを使用すると、検索用のファイル・システムの型とそれに一致するファイル型を制限できます。図 4-1 は、スクリプトのウィンドウを示します。

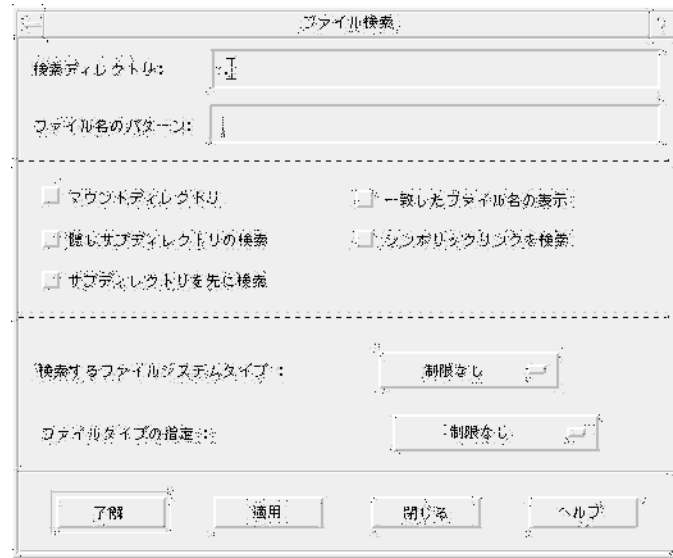


図 4-1 script_find のウィンドウ

捜したい検索ディレクトリとファイル名をウィンドウの上部のテキスト・フィールドに入力します。さらに、5 個のトグル・ボタンから適用可能な選択を選びます。オプション・メニュー内で検索をさらに制限することができます。必要な選択をすべて行ってから、[了解] をクリックします。すべてうまくいった場合は、その後すぐにウィンドウが現れ、find 処理の結果を表示します。検索ディレクトリやファイル名を指定しない場合、または指定された検索ディレクトリが無効な場合は、エラー・ダイアログが表示されます。たとえば、two_letter_calls という名前のファイルが、ディレクトリ /users/dlm 配下のどこにあるのか見つけようとしています。[検索ディレクトリ] テキスト・フィールドにディレクトリを入力する場合に、/users/dlm の代わりに誤って /users/dln を入力したとします。[了解] または [適用] をクリックすると、script_find はディレクトリ /users/dln を見つけることができないので、エラー・ダイアログを作成してユーザに通知します。



図 4-2 script_find エラー・ダイアログ

ミスを訂正すると、script_find はその後適切に実行し、ファイルが見つかった場合に要求したファイルの絶対パスを表示する dtterm ウィンドウを作成します。



図 4-3 絶対パスが表示されるウィンドウ

script_find が指定のディレクトリにファイルを見つけることができない場合、dtterm ウィンドウには何も表示されません。

script_find の解析

script_find の構造は、C プログラムに類似しています。つまり、いくつかの関数とコールバックが最初に表示され、次にメイン・スクリプトが表示されます。

スクリプトの最初の 2 行が重要で、全ての dtksh スクリプトに取り込む必要があります。

```
#!/usr/dt/bin/dtksh
. /usr/dt/lib/dtksh/DtFuncs.dtksh
```

1 行目は dtksh システムを実行し、2 行目は dtksh 簡易関数をロードします。2 行目は第 2 章で説明されているスクリプトでは使用されていませんが、これはそれらのスクリプトでは dtksh 簡易関数を使用しないためです。

関数とコールバック

script_find には、次のような関数とコールバックがあります。

- PostErrorDialog()
- OkCallback()
- LoadStickyValues()
- EvalCmd()
- RetrieveAndSaveCurrentValues()

PostErrorDialog()

この関数は、ユーザが無効なディレクトリを入力するなどのエラー検出時に呼び出されます。この関数は簡易関数 DtkshDisplayErrorDialog() を呼び出しますが、この簡易関数はタイトルが「検索エラー」で、呼び出し位置から渡される変数 \$1 にメッセージが格納されるダイアログ・ボックスを表示します。

```
dialogPostErrorDialog()
{
    DtDisplayErrorDialog "Find Error" "$1" \
        DIALOG_PRIMARY_APPLICATION_MODAL
}
```

最後のパラメータ DIALOG_PRIMARY_APPLICATION_MODAL は、他の対話が発生する前に応答しなければならないダイアログを作成するように dtksh に通知します。

OkCallback()

OkCallback() は、script_find メイン・ウィンドウの「了解」か「適用」ボタンが押されたときに呼び出されます。「了解」ボタンが押されると、script_find ウィンドウは管理から除外されます。「適用」か「了解」のどちらかに対して、入力検索ディレクトリは妥当性検査をされますが、それが無効な場合、OkCallback() は PostErrorDialog() を呼び

出します。有効な場合は、`script_find` ウィンドウのトグル・ボタンのステータスについて検査が行われ、そのステータスに対応した調整が変数 `$CMD` に対して行われます。この変数には、最後に実行されるコマンド全体が含まれています。

LoadStickyValues()

この関数は、ウィンドウが作成され管理されるようになった後でメイン・プログラムから呼び出されます。スクリプトの最新の実行結果からすべての値をロードします。これらの値は、関数 `RetrieveAndSaveCurrentValues()` によって `Find.sticky` と呼ばれるファイルに保存されます。

EvalCmd()

`EvalCmd()` は `LoadStickyValues()` によって使用され、`dtksh` コマンドとして `Find.sticky` にある各行の評価をします。`Find.sticky` ファイルの内容を次に示します。

```
XmTextSetString $SD "/users/dlm"
XmTextFieldSetInsertionPosition $SD 10
XmTextSetString $FNP "two_letter_calls"
XmTextFieldSetInsertionPosition $FNP 16
XtSetValues $FSTYPE menuHistory:$NODIR
XtSetValues $FILETYPE menuHistory:$NOTYPE
XmToggleButtonSetState $T2 true false
XmToggleButtonSetState $T4 true false
```

RetrieveAndSaveCurrentValues()

`RetrieveAndSaveCurrentValues()` は `script_find` ウィンドウにあるウィジェットの現在の設定と値を検索し、それらをファイル `Find.sticky` に保存します。`Find.sticky` は、スクリプトが実行された後に引き続き `LoadStickyValues()` によって使用されます。

メイン・スクリプト

スクリプトの残りは、C プログラムの `Main()` と同等なものです。`Xt` イントリンシクスを初期化し、`script_find` ウィンドウで使用されているすべてのウィジェットを作成します。1 行目にある `set -f` は、パス名にあるワイルドカード文字の拡張を禁止するように `dtksh` に通知します。これは、`find` コマンドがこの拡張を実行できるようにするために必要です。

script_find ウィンドウ (図 4-4 参照) は、4 つの領域から成る Form ウィジェットです。領域は Separator ウィジェットによってマークされ、各領域にはいくつかのウィジェットがあり、それらはすべて Form の子になります。

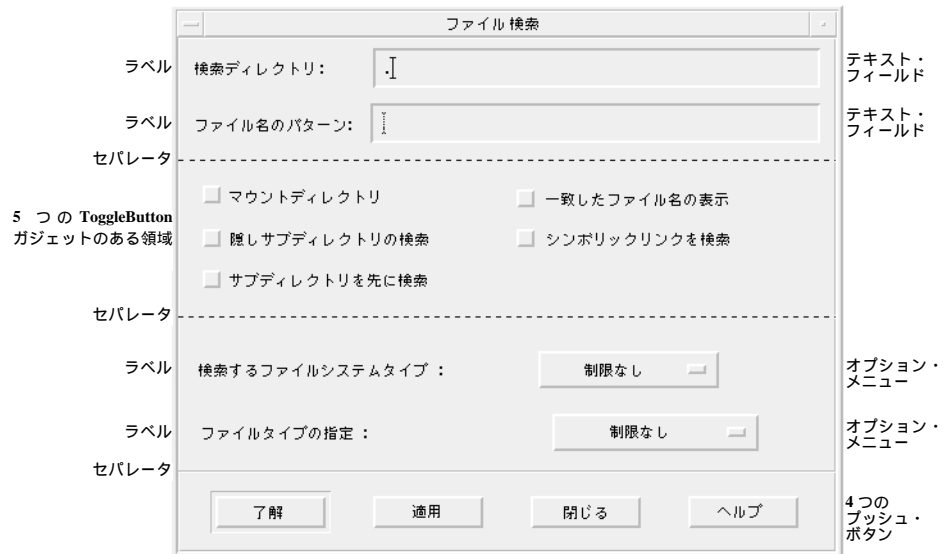


図 4-4 script_find ウィンドウにあるウィジェット

ウィジェットは、上部から下部に順番に領域ごとに作成されます。

初期化

初期化は、次のように Xt イントリンシクス関数 XtInitialize によって行われます。

```
XtInitialize TOPLEVEL find Dtksh $0 "${@:-}"
```

これにより、次に作成される Form ウィジェットの親として機能するトップレベルのシェルを作成します。

Form ウィジェットの作成

Form ウィジェットは、メインの親ウィジェットとして使用されます。Form は、ユーザがその子に制約を課することができるようにする Manager ウィジェットです。

script_find メイン・ウィンドウにあるほとんどのウィジェットは、Form の子です。ウィジェットの残りの部分の作成についての記述は、ウィンドウの 4 つの領域 (図 4-4 参照) ごとに分かれています。

1 番目の領域

1 番目の領域は、2 つの Label ウィジェットと 2 つの TextField ウィジェット、および 1 番目と 2 番目の領域を分割する Separator ウィジェットから成ります。

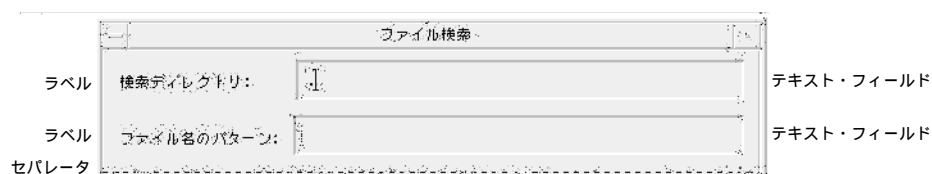


図 4-5 script_find ウィンドウの 1 番目の領域

次のコード・セグメントは、1 番目の Label ウィジェットを作成してから位置付け、DtkshAnchorTop 簡易関数と DtkshAnchorLeft 簡易関数を使用して Form 内にそのウィジェットを位置付けます。

```
XtCreateManagedWidget SDLABEL sdlabel XmLabel $FORM \
  labelString:"Search Directory:" \
  $(DtkshAnchorTop 12) \
  $(DtkshAnchorLeft 10)
```

次のコード・セグメントは、1 番目の TextField ウィジェットを作成してから位置付けます。このウィジェットは、Form ウィジェットと Label ウィジェットの両方に関連のある場所に位置付けられるので注意してください。

```
XtCreateManagedWidget SD sd XmText $FORM \
  columns:30 \
  value:"" \
  $(DtkshAnchorTop 6) \
  $(DtkshRightOf $SDLABEL 10) \
  $(DtkshAnchorRight 10)
```

```
navigationType:EXCLUSIVE_TAB_GROUP
XmTextFieldSetInsertionPosition $SD 1
```

残りの Label ウィジェットと TextField ウィジェットも同じ方法で作成されます。

Separator ウィジェットは、Form ウィジェットの子として作成され、2 番目の TextField ウィジェットの下に位置付けられます。

```
XtCreateManagedWidget SEP sep XmSeparator $FORM \
separatorType:SINGLE_DASHED_LINE \
$(DtkshUnder $FNP 10) \
$(DtkshSpanWidth)
```

2 番目の領域

2 番目の領域は、RowColumn ウィジェットと 5 つの ToggleButton ウィジェット、および 1 番目のとは別の Separator ウィジェットから成ります。

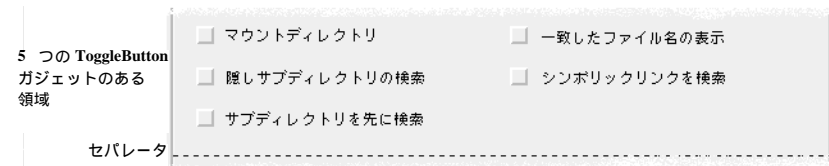


図 4-6 script_find ウィンドウの 2 番目の領域

ガジェットは、その属性の多くをその親に依存するウィジェットなので、メモリ・リソースを節約できます。

RowColumn ウィジェットは、Form ウィジェットの子として作成され、1 番目の領域で作成された Separator ウィジェットの直下に位置付けられます。

```
XtCreateManagedWidget RC rc XmRowColumn $FORM \
orientation:HORIZONTAL \
numColumns:3 \
packing:PACK_COLUMN \
$(DtkshUnder $SEP 10) \
$(DtkshSpanWidth 10 10) \
navigationType:EXCLUSIVE_TAB_GROUP
```

5 つの ToggleButton ガジェットは、次のように簡易関数 DtkshAddButtons を使用して RowColumn の子として作成されます。

```
DtkshAddButtons -w $RC XmToggleButtonGadget \
  T1 "Cross Mount Points" "" \
  T2 "Print Matching Filenames" "" \
  T3 "Search Hidden Subdirectories" "" \
  T4 "Follow Symbolic Links" "" \
  T5 "Descend Subdirectories First" ""
```

1 番目とは別の Separator が次に作成され、2 番目と 3 番目の領域を分割します。この Separator ウィジェット ID は SEP2 と呼ばれていますので注意してください。

```
XtCreateManagedWidget SEP2 sep XmSeparator $FORM \
  separatorType:SINGLE_DASHED_LINE \
  $(DtkshUnder $RC 10) \
  $(DtkshSpanWidth)
```

3 番目の領域

3 番目の領域は、2 つのオプション・メニューともう一つの Separator ウィジェットから成ります。



図 4-7 script_find ウィンドウの 3 番目の領域

オプション・メニューは、プルダウン・メニューです。オプション・メニュー・ボタンをクリックすると、多数の選択肢のあるメニュー区画が現れます。適切な選択肢へポインタをドラッグし、マウス・ボタンを離してください。メニュー区画が消え、オプション・メニュー・ボタン・ラベルが新規の選択肢を表示します。

最初のオプション・メニューのメニュー区画は、多数のプッシュ・ボタン・ガジェットから成り、find コマンドに強制するさまざまな制限を表示します。

```
XmCreatePulldownMenu PANE $FORM pane
DtkshAddButtons -w $PANE XmPushButtonGadget \
```

```
NODIR "no restrictions" "" \
NFS  "nfs"      "" \
CDFS "cdfs"     "" \
HFS  "hfs"      ""
```

Next, the Option Menu button itself is created and managed, with the menu pane just created (\$PANE) identified as a subMenuId:

```
XmCreateOptionMenu FSTYPE $FORM fstyle \
    labelString:"Restrict Search To File System Type:" \
    menuHistory:$NODIR \
    subMenuId:$PANE \
    $(DtkshUnder $SEP2 20) \
    $(DtkshSpanWidth 10 10) \
    navigationType:EXCLUSIVE_TAB_GROUP
XtManageChild $FSTYPE
```

2 番目のオプション・メニュー・ボタンも同じ方法で作成されます。このボタンは find コマンドにさらに制限を加えます。

3 番目のセパレータは、他のセパレータと同じ方法で作成されます。

4 番目の領域

4 番目の領域は、Form ウィジェットの子である 4 つのプッシュ・ボタンから成ります。



4 つのプッシュ・ボタンは次のように使用されます。

- [了解] は、script_find ウィンドウに入力されたパラメータで find コマンドを実行し、script_find ウィンドウを削除します。
- [適用] は、script_find ウィンドウに入力されたパラメータで find コマンドを実行しますが、script_find ウィンドウは削除しません。
- [閉じる] は、find コマンドを実行しないで script_find を終了します。
- [ヘルプ] は、script_find の使用に関する情報が入っているダイアログ・ボックスを作成します。

プッシュ・ボタンは、それぞれ別々にラベル付けされますが、他のウィジェットのボタンとほとんど同じ方法で作成され、位置付けられます。次のコード・セグメントは「了解」プッシュ・ボタンの作成方法を示します。

```
XtCreateManagedWidget OK ok XmPushButton $FORM \
    labelString:"Ok" \
    $(DtkshUnder $SEP3 10) \
    $(DtkshFloatLeft 4) \
    $(DtkshFloatRight 24) \
    $(DtkshAnchorBottom 10)
XtAddCallback $OK activateCallback "OkCallback"
```

オペレーティング・パラメータの設定

XtSetValues は、最初のオペレーティング・パラメータのいくつかを設定するために使用されます。

```
XtSetValues $FORM \
    initialFocus:$SD \
    defaultButton:$OK \
    cancelButton:$CLOSE \
    navigationType:EXCLUSIVE_TAB_GROUP
```

- 初期のフォーカスは、1 番目の領域の最初の TextField ウィジェットに設定されます。
- デフォルトのボタンには、4 番目の領域の「了解」プッシュ・ボタンが設定されます。
- 取消しのボタンには、4 番目の領域の「閉じる」ボタンが設定されます。
- ナビゲーション型には、EXCLUSIVE_TAB_GROUP が設定されます。

次の行は、リターン・キーを押しても Form 内のデフォルト・ボタンを起動しないように TextField ウィジェットを構成しています。その使用の詳細については、付録 B の EXCLUSIVE_TAB_GROUP についての説明を参照してください。

```
DtkshSetReturnKeyControls $SD $FNP $FORM $OK
```

認識とループ

スクリプトの最後の 3 行は、script_find ウィンドウの前の値をロードし、トップレベルのウィジェットを認識し、次にユーザ入力を待っているループに入ります。

```
LoadStickyValues
```

```
XtRealizeWidget $TOPLEVEL
```

```
XtMainLoop
```

この付録は、dtksh がサポートするコマンドのリストです。これらのコマンドの多くは、Motif、Xt イントリンシクス、Xlib のコマンドとほとんど同等です。値を返すコマンドには、呼び出しの最初のパラメータである環境変数としての戻り変数が必ずあります。それ以外でも相違点があるコマンドもあります。

次のサブセクションでは、各 dtksh コマンドの形式を示します。一般に、パラメータの順序と型は対応する C プロシージャと同じです。例外については注記します。コマンドの機能およびパラメータに関する詳細については、対応する Xlib、Xt イントリンシクス、Motif プロシージャの標準ドキュメントを参照してください。

コマンド定義では、*var*、*var2*、*var3* などのパラメータは、値が返される環境変数の名前をシェル・スクリプトが指定することを示します。*variable* は戻り値を受け取る環境変数のことです。

ブール値を返すコマンド (*if* 文の一部として直接使用できます) については、その旨注記します。

[] で括られたパラメータはオプションです。

組み込み Xlib コマンド

XBell display volume

XClearArea display drawable [optional GC arguments] *x y width height exposures*

XCclearWindow display drawable

XCopyArea display src dest srcX srcY width height destX destY [optional GC arguments]

XDefineCursor display window cursor

XDrawArc display drawable [optional GC arguments] *x y width height angle1 angle2*

XDrawLine display drawable [optional GC arguments] *x1 y1 x2 y2*

XDrawLines display drawable [-*coordinateMode*] [optional GC arguments] *x1 y1 x2 y2 [x3 y3 ...]*

coordinateMode は *CoordModeOrigin* または *CoordModePrevious* のいずれかです。

XDrawPoint display drawable [optional GC arguments] *x y*

XDrawPoints display drawable [-*coordinateMode*] [optional GC arguments] *x1 y1 [x2 y2 x3 y3 ...]*

coordinateMode は *CoordModeOrigin* または *CoordModePrevious* のいずれかです。

XDrawRectangle display drawable [optional GC arguments] *x y width height*

XDrawSegments display drawable [optional GC arguments] *x1 y1 x2 y2 [x3 y3 x4 y4 ...]*

XDrawString display drawable [optional GC arguments] *x y string*

XDrawImageString display drawable [optional GC arguments] *x y string*

XFillArc display drawable [optional GC arguments] *x y width height angle1 angle2*

XFillPolygon display drawable [-*shape*] [-*coordinateMode*] [optional GC arguments] *x1 y1 x2 y2 ...*

shape は *Complex*、*Convex*、*Nonconvex* のいずれかであり、*coordinateMode* は *CoordModeOrigin* または *CoordModePrevious* のいずれかです。

XFillRectangle display drawable [optional GC arguments] *x y width height*

XFlush *display*

XHeightOfScreen *variable screen*

XRaiseWindow *display window*

XRootWindowOfScreen *variable screen*

XSync *display discard*

discard は true または false です。

XTextWidth *variable fontName string*

注 – XTextWidth コマンドは対応する Xlib プロシージャとは異なります。これはコマンドがフォント構造体へのポインタではなくフォントの名前を取るからです。

XUndefineCursor *display window*

XWidthOfScreen *variable screen*

組み込み Xt イントリンシクス・コマンド

新規ウィジェットを作成するための Xt イントリンシクス・コマンドを使用するときは、新規ウィジェットのウィジェット・クラスを指定してください。ウィジェット (またはガジェット) のクラス名は、Motif が提供する標準クラス名です。たとえば Motif のプッシュボタン・ウィジェットのクラス名は XmPushButton で、Motif のラベル・ガジェットのクラス名は XmLabelGadget です。

XtAddCallback *widgetHandle callbackName ksh-command*

callbackName は標準 Motif コールバックまたは標準 Xt コールバックの名前の 1 つで、Xt または Xm の接頭辞を取り除いたものです。たとえば activateCallback などです。

`XtAddEventHandler widgetHandle eventMask nonMaskableFlag ksh-command`

`eventMask` は `mask/mask/mask` の形式で、`mask` コンポーネントは X イベント・マスクの標準セットのいずれかであり、`nonMaskableFlag` は `true` または `false` のいずれかです。

`XtAddInput variable [-r] fileDescriptor ksh-command`

指定したファイル記述子を入力ソースとして X ツールキットで登録します。入力ソースが不要になったときにそれを登録解除してファイル記述子を閉じるのは、シェル・スクリプトの入力ハンドラが行います。

`-r` オプション (raw モード) を指定すると、入力ソースから使用できるデータを `dtksh` が自動的に読み取らず、指定した `kshell` コマンドがすべてのデータを読み取ります。このオプションを指定しない場合は、絶対行 (つまり、エスケープされていない改行文字またはファイルの終わりによって終了している行) が読み取られるか、ファイルの終わりに到達したときにのみ、`ksh-command` で指定したコマンドが起動されます。テキスト以外のデータを処理するハンドラや、データ行で `dtksh` を自動的に読み取りたくないハンドラの場合には raw モードが便利です。ファイルの終わりを検出したときに、必要に応じて `XtRemoveInput` を使用して入力ソースを削除してファイル記述子を閉じるのは、シェル・スクリプトの入力ハンドラです。

どのような場合も、ハンドラが使用するいくつかの環境変数が設定されています。次のような変数があります。

<code>INPUT_LINE</code>	raw モードの場合は空です。そうでない場合は次に処理する行が入ります。
<code>INPUT_EOF</code>	ファイルの終わりに到達すると <code>true</code> に設定されます。そうでない場合は、 <code>false</code> です。
<code>INPUT_SOURCE</code>	この入力ソースに関連したファイル記述子です。
<code>INPUT_ID</code>	この入力ハンドラに関連した ID です。 <code>XtAddInput()</code> が返します。

`XtAddTimeout variable interval ksh-command`

XtAddWorkProc variable ksh-command

dtksh では、kshell コマンドは通常 kshell 関数名です。通常の作業プロセスと同様に、この関数には、作業プロセスを再び起動するか、作業を完了して自動的に登録解除されることを示す値が返されます。dtksh 関数が 0 を返した場合、作業プロセスはまだ登録されています。他の値を返した場合は、作業プロセスは自動的に登録解除されます。

XtAugmentTranslations widgetHandle translations

XtCreateApplicationShell variable applicationName widgetClass
[resource:value ...]

XtCallCallbacks widgetHandle callbackName

callbackName は標準 Motif コールバックまたは標準 Xt コールバックの名前の 1 つで、Xt または Xm の接頭辞を取り除いたものです。たとえば activateCallback などです。

XtClass variable widgetHandle

渡される先のウィジェット・ハンドルに関連したウィジェット・クラスの名前を返します。

XtCreateManagedWidget variable widgetName widgetClass
parentWidgetHandle [resource:value ...]

XtCreatePopupShell variable widgetName widgetClass
parentWidgetHandle [resource:value ...]

XtCreateWidget variable widgetName widgetClass
parentWidgetHandle [resource:value ...]

XtDestroyWidget widgetHandle [widgetHandle ...]

XtDisplay variable widgetHandle

XtDisplayOfObject variable widgetHandle

XtGetValues widgetHandle resource:var1 [resource:var2 ...]

XtHasCallbacks variable widgetHandle callbackName

callbackName は標準 Motif コールバックまたは標準 Xt コールバックの名前の 1 つで、Xt または Xm の接頭辞を取り除いたものです。たとえば activateCallback などです。

variable は CallbackNoList、CallbackHasNone、CallbackHasSome のいずれかに設定されます。

XtInitialize variable shellName applicationClassName applicationName
[arguments]

applicationClassName に Dtksh を使用すると、アプリケーションはデフォルトの dtksh app-defaults ファイルを使用します。*arguments* パラメータは、シェル・スクリプトのユーザが指定する可能性のある任意のコマンド行の引き数のリファレンスに使用されます。これらは通常、「\$@」のシェル形式を使用して行われます。

条件文に使用する値を返します。

XtIsManaged widgetHandle

条件文に使用する値を返します。

XtIsSubclass widgetHandle widgetClass

widgetClass はウィジェット・クラスの名前です。条件文に使用する値を返します。

XtNameToWidget variable referenceWidget name

XtIsRealized widgetHandle

条件文に使用する値を返します。

XtIsSensitive widgetHandle

条件文に使用する値を返します。

XtIsShell widgetHandle

条件文に使用する値を返します。

XtLastTimestampProcessed variable display

XtMainLoop

XtManageChild widgetHandle

XtManageChildren widgetHandle [widgetHandle ...]

XtMapWidget widgetHandle

XtOverrideTranslations widgetHandle translations

XtParent variable widgetHandle

XtPopdown widgetHandle

XtPopup widgetHandle grabType

grabType は GrabNone、GrabNonexclusive、GrabExclusive のいずれかの文字列です。

XtRealizeWidget widgetHandle

XtRemoveAllCallbacks widgetHandle callbackName

callbackName は標準 Motif コールバックまたは標準 Xt コールバックの名前の 1 つで、Xt または Xm の接頭辞を取り除いたものです。たとえば activateCallback などです。

XtRemoveCallback widgetHandle callbackName ksh-command

callbackName は標準 Motif コールバックまたは標準 Xt コールバックの名前の 1 つで、Xt または Xm の接頭辞を取り除いたものです。たとえば activateCallback などです。従来の Xt コールバックを指定する場合、コールバックを削除するときは、コールバックを最初に登録したときに指定したのと同じ kshell コマンド文字列を指定してください。

*XtRemoveEventHandler widgetHandle eventMask nonMaskableFlag
ksh-command*

eventMask は *mask|mask|mask* の形式で、*mask* コンポーネントは X イベント・マスクの標準セットのいずれかです。*nonMaskableFlag* で指定する *ButtonPressMask* は true または false のいずれかです。

従来の Xt イベント・ハンドラで true の場合、イベント・ハンドラを削除するときは、イベント・ハンドラを最初に登録したときに指定したのと同じ *eventMask* および *nonMaskableFlag* の設定と、kshell コマンド文字列を指定してください。

XtRemoveInput inputId

inputId は、XtAddInput コマンドを使用して代替入力ソースを登録したときに、指定した環境変数に返されるハンドルです。

XtRemoveTimeOut *timeoutId*

timeoutId は、XtAddTimeOut コマンドを使用してタイムアウトを登録したときに、指定した環境変数に返されるハンドルです。

XtRemoveWorkProc *workprocID*

workprocId は、XtAddWorkProc コマンドを使用して作業プロシージャを登録したときに、指定した環境変数に返されるハンドルです。

XtScreen *variable widgetHandle*

XtSetSensitive *widgetHandle state*

state は true または false のいずれかです。

XtSetValues *widgetHandle resource:value* [*resource:value* ...]

XtUninstallTranslations *widgetHandle*

XtUnmanageChild *widgetHandle*

XtUnmanageChildren *widgetHandle* [*widgetHandle* ...]

XtUnmapWidget *widgetHandle*

XtUnrealizeWidget *widgetHandle*

XtWindow *variable widgetHandle*

組み込み Motif コマンド

`XmAddWMProtocolCallback` *widgetHandle protocolAtom ksh-command*

protocolAtom は通常、`XmInternAtom` コマンドを使用して獲得します。

`XmAddWMProtocols` *widgetHandle protocolAtom [protocolAtom ...]*

protocolAtom は通常、`XmInternAtom` コマンドを使用して獲得します。

`XmCommandAppendValue` *widgetHandle string*

`XmCommandError` *widgetHandle errorString*

`XmCommandGetChild` *variable widgetHandle childType*

childType は、`DIALOG_COMMAND_TEXT`、`DIALOG_PROMPT_LABEL`、`DIALOG_HISTORY_LIST`、`DIALOG_WORK_AREA` のいずれかの文字列です。

`XmCommandSetValue` *widgetHandle commandString*

`XmCreateArrowButton` *variable parentWidgetHandle name [resource:value ...]*

`XmCreateArrowButtonGadget` *variable parentWidgetHandle name*
[resource:value ...]

`XmCreateBulletinBoard` *variable parentWidgetHandle name [resource:value ...]*

`XmCreateBulletinBoardDialog` *variable parentWidgetHandle name*
[resource:value ...]

`XmCreateCascadeButton` *variable parentWidgetHandle name [resource:value ...]*

`XmCreateCascadeButtonGadget` *variable parentWidgetHandle name*
[resource:value ...]

`XmCreateCommand` *variable parentWidgetHandle name [resource:value ...]*

`XmCreateDialogShell` *variable parentWidgetHandle name [resource:value ...]*

`XmCreateDrawingArea` *variable parentWidgetHandle name [resource:value ...]*

`XmCreateDrawnButton` *variable parentWidgetHandle name [resource:value ...]*

XmCreateErrorDialog variable parentWidgetHandle name [resource:value ...]

*XmCreateFileSelectionBox variable parentWidgetHandle name
[resource:value ...]*

*XmCreateFileSelectionDialog variable parentWidgetHandle name
[resource:value ...]*

XmCreateForm variable parentWidgetHandle name [resource:value ...]

XmCreateFormDialog variable parentWidgetHandle name [resource:value ...]

XmCreateFrame variable parentWidgetHandle name [resource:value ...]

*XmCreateInformationDialog variable parentWidgetHandle name
[resource:value ...]*

XmCreateLabel variable parentWidgetHandle name [resource:value ...]

XmCreateLabelGadget variable parentWidgetHandle name [resource:value ...]

XmCreateList variable parentWidgetHandle name [resource:value ...]

XmCreateMainWindow variable parentWidgetHandle name [resource:value ...]

XmCreateMenuBar variable parentWidgetHandle name [resource:value ...]

XmCreateMenuShell variable parentWidgetHandle name [resource:value ...]

XmCreateMessageBox variable parentWidgetHandle name [resource:value ...]

*XmCreateMessageDialog variable parentWidgetHandle name
[resource:value ...]*

XmCreateOptionMenu variable parentWidgetHandle name [resource:value ...]

XmCreatePanedWindow variable parentWidgetHandle name [resource:value ...]

XmCreatePopupMenu variable parentWidgetHandle name [resource:value ...]

XmCreatePromptDialog variable parentWidgetHandle name [resource:value ...]

XmCreatePulldownMenu variable parentWidgetHandle name [resource:value ...]

XmCreatePushButton variable parentWidgetHandle name [resource:value ...]

*XmCreatePushButtonGadget variable parentWidgetHandle name
[resource:value ...]*

*XmCreateQuestionDialog variable parentWidgetHandle name
[resource:value ...]*

XmCreateRadioBox variable parentWidgetHandle name [resource:value ...]

XmCreateRowColumn variable parentWidgetHandle name [resource:value ...]

XmCreateScale variable parentWidgetHandle name [resource:value ...]

XmCreateScrollBar variable parentWidgetHandle name [resource:value ...]

XmCreateScrolledList variable parentWidgetHandle name [resource:value ...]

XmCreateScrolledText variable parentWidgetHandle name [resource:value ...]

*XmCreateScrolledWindow variable parentWidgetHandle name
[resource:value ...]*

XmCreateSelectionBox variable parentWidgetHandle name [resource:value ...]

*XmCreateSelectionDialog variable parentWidgetHandle name
[resource:value ...]*

XmCreateSeparator variable parentWidgetHandle name [resource:value ...]

*XmCreateSeparatorGadget variable parentWidgetHandle name
[resource:value ...]*

XmCreateText variable parentWidgetHandle name [resource:value ...]

XmCreateTextField variable parentWidgetHandle name [resource:value ...]

XmCreateToggleButton variable parentWidgetHandle name [resource:value ...]

*XmCreateToggleButtonGadget variable parentWidgetHandle name
[resource:value ...]*

XmCreateWarningDialog variable parentWidgetHandle name [resource:value ...]

XmCreateWorkArea variable parentWidgetHandle name [resource:value ...]

*XmCreateWorkingDialog variable parentWidgetHandle name
[resource:value ...]*

XmFileSelectionDoSearch widgetHandle directoryMask

XmFileSelectionBoxGetChild variable widgetHandle childType

childType は、DIALOG_APPLY_BUTTON、DIALOG_CANCEL_BUTTON、DIALOG_DEFAULT_BUTTON、DIALOG_DIR_LIST、DIALOG_DIR_LIST_LABEL、DIALOG_FILTER_LABEL、DIALOG_FILTER_TEXT、DIALOG_HELP_BUTTON、DIALOG_LIST、DIALOG_LIST_LABEL、DIALOG_OK_BUTTON、DIALOG_SEPARATOR、DIALOG_SELECTION_LABEL、DIALOG_TEXT、DIALOG_WORK_AREA のいずれかの文字列です。

XmGetAtomName variable display atom

XmGetColors widgetHandle background variable var2 var3 var4

XmGetColors コマンドは、画面ポインタとカラーマップではなく *widgetHandle* を取るという点で C プロシージャと異なります。

XmGetFocusWidget variable widgetHandle

XmGetPostedFromWidget variable widgetHandle

XmGetTabGroup variable widgetHandle

XmGetTearOffControl variable widgetHandle

XmGetVisibility variable widgetHandle

XmInternAtom variable display atomString onlyIfExistsFlag

onlyIfExistsFlag は true または false のいずれかに設定されます。

XmIsTraversable widgetHandle

条件文で使用する値を返します。

XmListAddItem widgetHandle position itemString

XmListAddItem コマンドのパラメータの順序は、対応する C プログラミングのパラメータの順序とは同じではありません。

XmListAddItems widgetHandle position itemString [itemString ...]

XmListAddItems コマンドのパラメータの順序は、対応する C プログラミングのパラメータの順序とは同じではありません。

XmListAddItemsUnselected widgetHandle position itemString [itemString ...]

XmListAddItemsUnselected コマンドのパラメータの順序は、対応する C プログラミングのパラメータの順序とは同じではありません。

XmListAddItemUnselected widgetHandle position itemString

XmListAddItemUnselected コマンドのパラメータの順序は、対応する C プログラミングのパラメータの順序とは同じではありません。

XmListDeleteAllItems widgetHandle

XmListDeleteItem widgetHandle itemString

XmListDeleteItems widgetHandle itemString [itemString ...]

XmListDeleteItemsPos widgetHandle itemCount position

XmListDeletePos widgetHandle position

XmListDeletePositions widgetHandle position [position ...]

XmListDeselectAllItems widgetHandle

XmListDeselectItem widgetHandle itemString

XmListDeselectPos widgetHandle position

XmListGetSelectedPos variable widgetHandle

カンマで区切られた *variable* の索引のリストを返します。条件文で使用する値を返します。

XmListGetKbdItemPos variable widgetHandle

XmListGetMatchPos variable widgetHandle itemString

カンマで区切られた *variable* の索引のリストを返します。条件文で使用する値を返します。

XmListItemExists widgetHandle itemString

条件文で使用する値を返します。

XmListItemPos variable widgetHandle itemString

XmListPosSelected widgetHandle position

条件文で使用する値を返します。

XmListPosToBounds widgetHandle position variable var2 var3 vari4

条件文で使用する値を返します。

XmListReplaceItemsPos widgetHandle position itemString [itemString ...]

XmListReplaceItemsPos コマンドのパラメータの順序は、対応する C プログラミングのパラメータの順序とは同じではありません。

*XmListReplaceItemsPosUnselected widgetHandle position itemString
[itemString ...]*

XmListReplaceItemsPosUnselected コマンドのパラメータの順序は、対応する C プログラミングのパラメータの順序とは同じではありません。

XmListSelectItem widgetHandle itemString notifyFlag

notifyFlag は true または false のいずれかに設定されます。

XmListSelectPos widgetHandle position notifyFlag

notifyFlag は true または false のいずれかに設定されます。

XmListSetAddMode widgetHandle state

state は true または false のいずれかに設定されます。

XmListSetBottomItem widgetHandle itemString

XmListSetBottomPos widgetHandle position

XmListSetHorizPos widgetHandle position

XmListSetItem widgetHandle itemString

XmListSetKbdItemPos widgetHandle position

条件文で使用する値を返します。

XmListSetPos widgetHandle position

XmListUpdateSelectedList widgetHandle

XmMainWindowSep1 variable widgetHandle

XmMainWindowSep2 variable widgetHandle

XmMainWindowSep3 variable widgetHandle

*XmMainWindowSetAreas widgetHandle menuWidgetHandle
commandWidgetHandle
horizontalScrollbarWidgetHandle
verticalScrollbarWidgetHandle
workRegionWidgetHandle*

XmMenuPosition widgetHandle eventHandle

eventHandle は X イベントに対応します。これは通常 `CB_CALL_DATA.EVENT`、`EH_EVENT`、`TRANSLATION_EVENT` という環境変数のいずれかにアクセスすると獲得できます。

XmMessageBoxGetChild variable widgetHandle childType

childType は、`DIALOG_CANCEL_BUTTON`、`DIALOG_DEFAULT_BUTTON`、`DIALOG_HELP_BUTTON`、`DIALOG_MESSAGE_LABEL`、`DIALOG_OK_BUTTON`、`DIALOG_SEPARATOR`、`DIALOG_SYMBOL_LABEL` のいずれかの文字列です。

XmOptionButtonGadget variable widgetHandle

XmOptionLabelGadget variable widgetHandle

`XmProcessTraversal widgetHandle direction`

direction は、`TRAVERSE_CURRENT`、`TRAVERSE_DOWN`、`TRAVERSE_HOME`、`TRAVERSE_LEFT`、`TRAVERSE_NEXT`、`TRAVERSE_NEXT_TAB_GROUP`、`TRAVERSE_PREV`、`TRAVERSE_PREV_TAB_GROUP`、`TRAVERSE_RIGHT`、`TRAVERSE_UP` のいずれかの文字列です。

条件文で使用する値を返します。

`XmRemoveWMProtocolCallback widgetHandle protocolAtom ksh-command`

protocolAtom は通常、`XmInternAtom` コマンドを使用して獲得されます。

従来のウィンドウ・マネージャ・コールバックを指定する場合、コールバックを削除するときは、コールバックを最初に登録したときに指定したのと同じ `kshell` コマンド文字列を指定してください。

`XmRemoveWMProtocols widgetHandle protocolAtom [protocolAtom ...]`

protocolAtom は通常、`XmInternAtom` コマンドを使用して獲得されます。

`XmScaleGetValue widgetHandle variable`

`XmScaleSetValue widgetHandle value`

`XmScrollBarGetValues widgetHandle variable var2 var3 var4`

`XmScrollBarSetValues widgetHandle value sliderSize increment pageIncrement
notifyFlag`

notifyFlag は `true` または `false` のいずれかに設定されます。

`XmScrollVisible widgetHandle widgetHandle leftRightMargin topBottomMargin`

`XmSelectionBoxGetChild variable widgetHandle childType`

childType は、`DIALOG_CANCEL_BUTTON`、`DIALOG_DEFAULT_BUTTON`、`DIALOG_HELP_BUTTON`、`DIALOG_APPLY_BUTTON`、`DIALOG_LIST`、`DIALOG_LIST_LABEL`、`DIALOG_OK_BUTTON`、`DIALOG_SELECTION_LABEL`、`DIALOG_SEPARATOR`、`DIALOG_TEXT`、`DIALOG_WORK_AREA` のいずれかの文字列です。

`XmTextClearSelection widgetHandle time`

time は通常、X イベント内から獲得され、`XtLastTimestampProcessed` コマンドの呼び出しによって照会されます。

`XmTextCopy widgetHandle time`

time は通常、X イベント内から獲得され、`XtLastTimestampProcessed` コマンドの呼び出しによって照会されます。

条件文で使用する値を返します。

`XmTextCut widgetHandle time`

time は通常、X イベント内から獲得され、`XtLastTimestampProcessed` コマンドの呼び出しによって照会されます。

条件文で使用する値を返します。

`XmTextDisableRedisplay widgetHandle`

`XmTextEnableDisplay widgetHandle`

`XmTextFindString widgetHandle startPosition string direction variable`

direction は `TEXT_FORWARD` または `TEXT_BACKWARD` のいずれかの文字列です。

条件文で使用する値を返します。

`XmTextGetBaseline variable widgetHandle`

`XmTextGetEditable widgetHandle`

条件文で使用する値を返します。

`XmTextGetInsertionPosition variable widgetHandle`

`XmTextGetLastPosition variable widgetHandle`

`XmTextGetMaxLength variable widgetHandle`

`XmTextGetSelection variable widgetHandle`

`XmTextGetSelectionPosition widgetHandle variable var2`

条件文で使用する値を返します。

`XmTextGetString variable widgetHandle`

`XmTextGetTopCharacter variable widgetHandle`

`XmTextInsert widgetHandle position string`

`XmTextPaste widgetHandle`

条件文で使用する値を返します。

`XmTextPosToXY widgetHandle position variable var2`

条件文で使用する値を返します。

`XmTextRemove widgetHandle`

条件文で使用する値を返します。

`XmTextReplace widgetHandle fromPosition toPosition string`

`XmTextScroll widgetHandle lines`

`XmTextSetAddMode widgetHandle state`

`state` は `true` または `false` のいずれかに設定されます。

`XmTextSetEditable widgetHandle editableFlag`

`editableFlag` は `true` または `false` のいずれかに設定されます。

`XmTextSetHighlight widgetHandle leftPosition rightPosition mode`

`mode` は、`HIGHLIGHT_NORMAL`、`HIGHLIGHT_SELECTED`、`HIGHLIGHT_SECONDARY_SELECTED` のいずれかの文字列です。

`XmTextSetInsertionPosition widgetHandle position`

`XmTextSetMaxLength widgetHandle maxLength`

XmTextSetSelection widgetHandle firstPosition lastPosition time

time は通常、X イベント内から獲得され、XtLastTimestampProcessed コマンドの呼び出しによって照会されます。

XmTextSetString widgetHandle string

XmTextSetTopCharacter widgetHandle topCharacterPosition

XmTextShowPosition widgetHandle position

XmTextXYToPos variable widgetHandle x y

XmTextFieldClearSelection widgetHandle time

time は通常、X イベント内から獲得され、XtLastTimestampProcessed コマンドの呼び出しによって照会されます。

XmTextFieldGetBaseline variable widgetHandle

XmTextFieldGetEditable widgetHandle

条件文で使用する値を返します。

XmTextFieldGetInsertionPosition variable widgetHandle

XmTextFieldGetLastPosition variable widgetHandle

XmTextFieldGetMaxLength variable widgetHandle

XmTextFieldGetSelection variable widgetHandle

XmTextFieldGetSelectionPosition widgetHandle variable var2

条件文で使用する値を返します。

XmTextFieldGetString variable widgetHandle

XmTextFieldInsert widgetHandle position string

XmTextFieldPosToXY widgetHandle position variable var2

条件文で使用する値を返します。

XmTextFieldRemove widgetHandle

条件文で使用する値を返します。

XmTextFieldReplace widgetHandle fromPosition toPosition string

XmTextFieldSetEditable widgetHandle editableFlag

editableFlag は true または false のいずれかに設定されます。

XmTextFieldSetHighlight widgetHandle leftPosition rightPosition mode

mode は HIGHLIGHT_NORMAL、HIGHLIGHT_SELECTED、
HIGHLIGHT_SECONDARY_SELECTED のいずれかの文字列です。

XmTextFieldSetInsertionPosition widgetHandle position

XmTextFieldSetMaxLength widgetHandle maxLength

XmTextFieldSetSelection widgetHandle firstPosition lastPosition time

time は通常、X イベント内から獲得され、XtLastTimestampProcessed コマンドの呼び出しによって照会されます。

XmTextFieldSetString widgetHandle string

XmTextFieldShowPosition widgetHandle position

XmTextFieldXYToPos variable widgetHandle x y

XmTextFieldCopy widgetHandle time

time は通常、X イベント内から獲得され、XtLastTimestampProcessed コマンドの呼び出しによって照会されます。

条件文で使用する値を返します。

XmTextFieldCut widgetHandle time

time は通常、X イベント内から獲得され、XtLastTimestampProcessed コマンドの呼び出しによって照会されます。

条件文で使用する値を返します。

`XmTextFieldPaste widgetHandle`

条件文で使用する値を返します。

`XmTextFieldSetAddMode widgetHandle state`

`state` は true または false のいずれかに設定されます。

`XmToggleButtonGadgetGetState widgetHandle`

条件文で使用する値を返します。

`XmToggleButtonGadgetSetState widgetHandle state notifyFlag`

`state` は true または false のいずれかに設定され、`notifyFlag` は true または false のいずれかに設定されます。

`XmToggleButtonGetState widgetHandle`

条件文で使用する値を返します。

`XmToggleButtonSetState widgetHandle state notifyFlag`

`state` は true または false のいずれかに設定され、`notifyFlag` は true または false のいずれかに設定されます。

`XmUpdateDisplay widgetHandle`

組み込み共通デスクトップ環境アプリケーション・ヘルプ・コマンド

`DtCreateQuickHelpDialog variable parentWidgetHandle name`
[resource:value ...]

`DtCreateHelpDialog variable parentWidgetHandle name` [resource:value ...]

`DtHelpQuickDialogGetChild variable widgetHandle childType`

`childType` は、HELP_QUICK_OK_BUTTON、HELP_QUICK_PRINT_BUTTON、HELP_QUICK_HELP_BUTTON、HELP_QUICK_SEPARATOR、HELP_QUICK_MORE_BUTTON、HELP_QUICK_BACK_BUTTON のいずれかの文字列です。

`DtHelpReturnSelectedWidgetId` *variable widgetHandle var2*

variable は、HELP_SELECT_VALID、HELP_SELECT_INVALID、HELP_SELECT_ABORT、HELP_SELECT_ERROR のいずれかの文字列です。*var2* は選択したウィジェットの *widgetHandle* に設定されます。

`DtHelpSetCatalogName` *catalogName*

組み込みローカル化コマンド

`catopen` *variable catalogName*

指定したメッセージ・カタログを開き、*variable* で指定した環境変数にカタログ ID を返します。メッセージ・カタログに指定されたファイル記述子を閉じることをシェル・スクリプトが必要としている場合は、`catclose` コマンドを使用してカタログ ID を閉じてください。

`catgets` *variable catalogId setNumber messageNumber defaultMessageString*

要求したメッセージ文字列を *catalogId* パラメータで指定したメッセージ・カタログから取り出します。メッセージ文字列を取り出せない場合は、デフォルトのメッセージ文字列が返されます。いずれの場合も返されたメッセージ文字列は *variable* で指定した環境変数に入れられます。

`catclose` *catalogId*

指定した *catalogId* に関連したメッセージ・カタログを閉じます。

組み込み libDt セッション管理コマンド

`DtSessionRestorePath` *widgetHandle variable sessionFile*

(パス情報を含まない) セッション・ファイルのファイル名を指定します。このコマンドは *variable* で指定する環境変数にセッション・ファイルの絶対パスを返します。

正常終了の場合は 0、異常終了の場合は 1 を返します。

`DtSessionSavePath widgetHandle variable var2`

variable で指定する環境変数にセッション・ファイルの絶対パス名を返します。(パス情報を含まない) セッション・ファイルのファイル名部分は *var2* で指定される環境変数に返されます。

正常終了の場合は 0、異常終了の場合は 1 を返します。

`DtShellIsIconified widgetHandle`

シェル・スクリプトがシェル・ウィンドウのアイコン化状態を照会できるようにします。正常終了の場合は 0、異常終了の場合は 1 を返します。

`DtSetStartupCommand widgetHandle commandString`

セッション管理プロセスの部分が、ユーザがセッションを次に再開するときのアプリケーションの再起動方法をセッション・マネージャに通知します。このコマンドは指定したコマンド文字列をセッション・マネージャに渡します。ウィジェット・ハンドルはアプリケーション・シェルです。

`DtSetIconifyHint widgetHandle iconifyHint`

iconifyHint は true または false のいずれかに設定されます。

最初のアイコン化状態をシェル・ウィンドウが設定できるようにします。このコマンドは、ウィジェットに指定したウィンドウが認識されていて表示されていない場合のみ実行できます。

組み込み libDt ワークスペース管理コマンド

`DtWsmAddCurrentWorkspaceCallback variable widgetHandle ksh-command`

ユーザがワークスペースを変更するたびに、指定した kshell コマンドを評価します。このコールバックに指定したハンドルは *variable* で指定した環境変数に返されます。*widgetHandle* で指定するウィジェットはシェル・ウィジェットです。

`DtWsmRemoveWorkspaceCallback callbackHandle`

ワークスペース通知コールバックを削除します。削除するときは、`DtWsmAddCurrentWorkspaceCallback` でコールバックを登録したときに返されたコールバック・ハンドルを *callbackHandle* に指定してください。

`DtWsmGetCurrentWorkspace` *display rootWindow variable*

ユーザの現在のワークスペースを示す X アトムを *variable* で指定した環境変数に返します。
XmGetAtomName コマンドを使用して X アトムを文字列表現にマップしてください。

`DtWsmSetCurrentWorkspace` *widgetHandle workspaceNameAtom*

ユーザの現在のワークスペースを *workspaceNameAtom* で指定したワークスペースに変更します。

正常終了の場合は 0、異常終了の場合は 1 を返します。

`DtWsmGetWorkspaceList` *display rootWindow variable*

カンマで区切られた X アトムの文字列を返します。ユーザのために定義された現在のワークスペースのセットを *variable* で指定した環境変数に返します。

正常終了の場合は 0、異常終了の場合は 1 を返します。

`DtWsmGetWorkspacesOccupied` *display window variable*

カンマで区切られた X アトムの文字列を返します。*window* で指定したシェル・ウィンドウで占められた現在のワークスペースのセットを *variable* で指定した環境変数に返します。

正常終了の場合は 0、異常終了の場合は 1 を返します。

`DtWsmSetWorkspacesOccupied` *display window workspaceList*

window で指定したシェル・ウィンドウを *workspaceList* で指定したワークスペースのセットに移動します。*workspaceList* はカンマで区切られた X アトムのリストです。

`DtWsmAddWorkspaceFunctions` *display window*

ウィンドウ・マネージャ・メニューに、ウィンドウを他のワークスペースに移動させるのに使用する関数を取り込ませます。このコマンドは、ウィンドウが描画状態である場合のみ実行できます。

DtWsmRemoveWorkspaceFunctions display window

ウィンドウ・マネージャ・メニューに、ウィンドウを他のワークスペースに移動させるのに使用する関数を表示しないようにします。こうするとウィンドウが他のワークスペースに移動しません。このコマンドは、ウィンドウが描画状態である場合のみ実行できます。

DtWsmOccupyAllWorkspaces display window

ウィンドウが、作成した新規ワークスペースも含めてすべてのワークスペースを占有するよう要求します。

DtWsmGetCurrentBackdropWindows display rootWindow variable

カンマで区切られたウィンドウ ID の文字列を返します。この ID は一連のルート・バックドロップ・ウィンドウを示します。

組み込み libDt アクション・コマンド

この節で説明するコマンドは、アクション・データベースの読み込み、データベースで定義されたアクションに関する情報の照会、アクションの起動要求を行うためのツールを提供します。

DtDbLoad

アクションおよびデータ型データベースを読み込みます。複数回呼び出すと、古いデータベースは新しいデータベースを読み込む前に解放されます。このコマンドは、他の libDt アクション・コマンドまたは libDt データ型作成コマンドを起動する前に起動してください。シェル・スクリプトも DtDbReloadNotify コマンドを使用するので、新しいデータベースが読み込まれるとシェル・スクリプトに通知されます。

DtDbReloadNotify ksh-command

アクションまたはデータ型データベースの再読み込みが必要になったら必ず通知するよう要求します。指定した kshell コマンドは、通知を受け取ると実行されます。通常 kshell コマンドは、DtDbLoad コマンドの呼び出しを含みます。

DtActionExists actionName

actionName パラメータで指定した名前のデータベースにアクションが存在するかどうかをテストします。条件文に使用する値を返します。

`DtActionLabel` *variable actionName*

指定したアクションに関連したローカライズ可能な LABEL 属性を返します。アクションが存在しない場合は、空の文字列が返されます。

`DtActionDescription` *variable actionName*

指定したアクションに関連した DESCRIPTION 属性を返します。アクションが定義されていないか、DESCRIPTION 属性が指定されていない場合は、空の文字列が返されます。

組み込み libDt データ型作成コマンド

`DtLoadDataTypes`

データ型データベースを読み込みます。他のデータ型作成コマンドより先に起動されなければなりません。

`DtDtsFileToDataType` *variable filePath*

filePath 引き数で指定したファイルに関連したデータ型の名前を *variable* 引き数に返します。ファイルがない場合、*variable* 引き数には空の文字列が設定されます。

`DtDtsFileToAttributeValue` *variable filePath attrName*

filePath で指定したファイルに関連したデータ型のための *attrName* で指定した属性の値を表す文字列を返します。属性が定義されていない場合、またはファイルがない場合、*variable* 引き数には空の文字列が設定されます。

`DtDtsFileToAttributeList` *variable filePath*

filePath で指定したファイルに関連したデータ型に定義された属性名をスペースで区切られたリストで返します。シェル・スクリプトが属性の個々の値を照会するには、`DtDtsFileToAttributeValue` コマンドを使用します。ファイルがない場合、*variable* 引き数には空の文字列が設定されます。このコマンドは定義された属性の名前だけを返して値は返さないという点で、対応する C プログラミングのコマンドとは異なります。

`DtDtsDataTypeToAttributeValue` *variable dataType attrName optName*

dataType で指定したデータ型の *attrName* で指定した属性の値を表す文字列を返します。属性が定義されていない場合、または指定したデータ型が存在しない場合、*variable* 引き数には空の文字列が設定されます。

`DtDtsDataTypeToAttributeList` *variable dataType optName*

dataType で指定したデータ型に定義された属性名をスペースで区切られたリストで返します。シェル・スクリプトが属性の個々の値を照会するには、`DtDtsDataTypeToAttributeValue` コマンドを使用します。データ型が定義されていない場合、*variable* 引き数には空の文字列が設定されます。このコマンドは定義された属性の名前だけを返して値は返さないという点で、対応する C プログラミングのコマンドとは異なります。

`DtDtsFindAttribute` *variable name value*

name 引き数に指定した属性で *value* 引き数に指定した値を持つデータ型の名前をスペースで区切られたリストで返します。エラーが生じた場合、*variable* 引き数には空の文字列が設定されます。

`DtDtsDataTypeNames` *variable*

データ型データベースに現在定義されているすべてのデータ型をスペースで区切られたリストで返します。エラーが生じた場合、*variable* 引き数には空の文字列が設定されます。

`DtDtsSetDataType` *variable filePath dataType override*

指定したディレクトリのデータ型を設定します。*variable* 引き数にはディレクトリに保存されたデータ型が設定されます。

`DtDtsDataTypeIsAction` *dataType*

特定のデータ型がアクション・エントリを示すかどうかを判別します。条件文で使用する値を返します。

その他の組み込み libDt コマンド

`DtGetHourGlassCursor` *variable display*

標準 Dt 時計表示カーソルに関連した X カーソル ID を返します。

`DtTurnOnHourGlass` *widgetHandle*

指定したウィジェットの標準 Dt 時計表示カーソルをオンにします。

`DtTurnOffHourGlass` *widgetHandle*

指定したウィジェットの標準 Dt 時計表示カーソルをオフにします。

組み込みデスクトップ・サービス・メッセージ・セット・コマンド

次に示すコマンドは、シェル・スクリプトがデスクトップ・サービス・プロトコルの参入に必要なデスクトップ・サービス・メッセージ・セットの最小サブセットを実行します。ほとんどの ToolTalk コマンドは対応する C プログラミング・コールとわずかに異なっています。通常はポインタを返す ToolTalk コマンドでは、C アプリケーションが `tt_ptr_error()` 関数を呼び出すことによってそのポインタを有効にします。この関数呼び出しはポインタが有効であるかどうかを示す `Tt_status` 値を返します。有効でない場合は、その理由を示します。kshell コードの設計上の理由から、シェル・スクリプトが見る文字列ポインタは、通常は、基本の C コードが返す文字列ポインタと同じではありません。シェルのプログラミング中は、重要な情報が文字列ポインタではなく文字列値で示されるため、これは問題にはなりません。

シェル・スクリプトがポインタのステータスを獲得できるようにするため、通常はポインタを返すコマンドは、自動的にそのポインタに関連する `Tt_status` 値も返します。これによってシェル・スクリプトは元のポインタの有効性を検査するために呼び出しを追加する必要がなくなります。ポインタ・エラーが生じた場合は、`dtksh` はポインタ値として空の文字列を返し、`Tt_status` コードを設定します。

`Tt_status` 値は *status* 引き数に返されます。`Tt_status` 値はエラーを示す文字列で、次の値のいずれかになります。

```
TT_OK
TT_WRN_NOTFOUND
TT_WRN_STALE_OBJID
TT_WRN_STOPPED
```

TT_WRN_SAME_OBJID
TT_WRN_START_MESSAGE
TT_ERR_CLASS
TT_ERR_DBAVAIL
TT_ERR_DBEXIST
TT_ERR_FILE
TT_ERR_INVALID
TT_ERR_MODE
TT_ERR_ACCESS
TT_ERR_NOMP
TT_ERR_NOTHANDLER
TT_ERR_NUM
TT_ERR_OBJID
TT_ERR_OP
TT_ERR_OTYPE
TT_ERR_ADDRESS
TT_ERR_PATH
TT_ERR_POINTER
TT_ERR_PROCID
TT_ERR_PROPLEN
TT_ERR_PROPNAME
TT_ERR_PTYPE
TT_ERR_DISPOSITION
TT_ERR_SCOPE
TT_ERR_SESSION
TT_ERR_VTYPE
TT_ERR_NO_VALUE
TT_ERR_INTERNAL
TT_ERR_READONLY
TT_ERR_NO_MATCH
TT_ERR_UNIMP
TT_ERR_OVERFLOW
TT_ERR_PTPE_START
TT_ERR_CATEGORY
TT_ERR_DBUPDATE
TT_ERR_DBFULL

TT_ERR_DBCONSIST
TT_ERR_STATE
TT_ERR_NOMEM
TT_ERR_SLOTNAME
TT_ERR_XDR
TT_DESKTOP_EPERM
TT_DESKTOP_ENOENT
TT_DESKTOP_EINTR
TT_DESKTOP_EIO
TT_DESKTOP_EAGAIN
TT_DESKTOP_ENOMEM
TT_DESKTOP_EACCES
TT_DESKTOP_EFAULT
TT_DESKTOP_EEXIST
TT_DESKTOP_ENODEV
TT_DESKTOP_ENOTDIR
TT_DESKTOP_EISDIR
TT_DESKTOP_EINVAL
TT_DESKTOP_ENFILE
TT_DESKTOP_EMFILE
TT_DESKTOP_ETXBSY
TT_DESKTOP_EFBIG
TT_DESKTOP_ENOSPC
TT_DESKTOP_EROFS
TT_DESKTOP_EMLINK
TT_DESKTOP_EPIPE
TT_DESKTOP_ENOMSG
TT_DESKTOP_EDEADLK
TT_DESKTOP_ECANCELED
TT_DESKTOP_ENOTSUP
TT_DESKTOP_ENODATA
TT_DESKTOP_EPROTO
TT_DESKTOP_ENOTEMPTY
TT_DESKTOP_ETIMEOUT
TT_DESKTOP_EALREADY

TT_DESKTOP_UNMODIFIED
TT_MEDIA_ERR_SIZE
TT_MEDIA_ERR_FORMAT

一部のコマンドはパラメータとしてメッセージ・スコープを取ります。スコープは発信メッセージを受信する可能性のあるクライアントを示します。これらのコマンドでは、*scope* パラメータが次の値のいずれかに設定されます。

TT_SCOPE_NONE
TT_SESSION
TT_FILE
TT_BOTH
TT_FILE_IN_SESSION

tt_file_netfile variable status filename

指定した *filename* (ローカル・ホストで有効と見なされるファイル名) を、対応する *netfilename* 書式に変換します。*netfilename* はネットワーク上の他のホストに渡され、他のホストに関連したパスになるように *tt_netfile_file* コマンドによって変換されます。

tt_netfile_file variable status netfilename

指定した *netfilename* をローカル・ホストで有効なパス名に変換します。

tt_host_file_netfile variable status host filename

指定したファイル (指定したホストに存在すると見なされる) を、対応する *netfilename* 書式に変換します。

tt_host_netfile_file variable status host netfilename

指定した *netfilename* を指定したホストで有効なパスに変換します。

tttdt_open variable status var2 toolname vendor version sendStarted

ToolTalk 通信エンドポイントを開きます。この接続に関連した *procID* を *variable* 引き数に返します。この接続に関連したファイル記述子は *var2* に返します。ファイル記述子は代替 Xt 入力ハンドラを登録するのに使用します。*sendStarted* 引き数は、*true* が設定されると Started メッセージが自動的に送信されます。

ttdt_open によって返される任意の procID は埋め込みスペースを含みます。kshell が procID を複数のパラメータである (埋め込みスペースを伴う 1 つのパラメータではなく) と解釈しないように、procID を含む環境変数は次に示すように常に二重引用符で囲んでください。

```
ttdt_close STATUS "$PROC_ID" "" True
```

ttk_Xt_input_handler procID source id

ToolTalk メッセージを受信および処理するために、シェル・スクリプトは ttdt_open の呼び出しによって返されるファイル記述子に対して Xt 入力ハンドラを登録しなければなりません。Xt 入力ハンドラは XtAddInput コマンドを使用して登録しますが、raw 入力ハンドラとして登録してください。シェル・スクリプトが登録した入力ハンドラは ttk_Xt_input_handler を呼び出してメッセージの受信および処理を行います。次のコード・ブロックで実行方法を示します。

```
ttdt_open PROC_ID STATUS FID "Tool" "HP" "1.0" True XtAddInput
INPUT_ID -r $FID "ProcessTTInput \"${PROC_ID}\""
ProcessTTInput()
{
    ttk_Xt_input_handler $1 $INPUT_SOURCE $INPUT_ID
}
```

代替 Xt 入力ハンドラの詳細については、XtAddInput コマンドの説明を参照してください。

procID 環境変数を示すには、前後に \ (バックスラッシュと二重引用符) を付けることが必須ですので注意してください。procID 環境変数の値は埋め込みスペースを含んでおり、誤って解釈される可能性があるからです。

ttdt_close status procID newProcId sendStopped

指定した通信接続を閉じ、*sendStopped* 引き数に true が設定されているとオプションで Stopped 通知を送信します。

ttdt_open の呼び出しによって返される procID には埋め込みスペースが入っているので、procID 環境変数を示すものは次のように二重引用符で囲んでください。

```
ttdt_close STATUS "$PROC_ID" "$NEW_PROC_ID" False
```

`ttdt_session_join variable status sessId shellWidgetHandle join`

多くの標準デスクトップ・メッセージ・インタフェースのパターン・コールバックとデフォルト・コールバックを登録することによって、適切なデスクトップとして *sessID* 引き数で指定したセッションを結合します。*sessID* 引き数が値を指定しない(つまり空の文字列である) 場合、デフォルト・セッションが結合されます。*shellWidgetHandle* 引き数がウィジェット・ハンドルを指定した(つまり空の文字列ではない) 場合は、mappedWhenManaged applicationShellWidget になります。*join* 引き数はブール値で、true または false が設定されます。このコマンドは *variable* 引き数に隠された pattern ハンドルを返します。このハンドルがなくなったときは、`ttdt_session_quit` コマンドで破棄できます。

`ttdt_session_quit status sessId sessPatterns quit`

sessPatterns 引き数で指定されるメッセージ・パターンを破棄します。*quit* 引き数に true が設定されている場合は、*sessId* 引き数が示セッションを終了します。*sessId* が空の場合はデフォルト・セッションを終了します。

`ttdt_file_join variable status pathName scope join ksh-command`

削除、変更、復帰、移動、保存されたメッセージの配信対象を、指定したスコープの指定したファイルに登録します。隠されたパターン・ハンドルが *variable* 引き数に返されます。指定したファイルのメッセージを監視する必要がなくなったときは、`ttdt_file_quit` コマンドで破棄できます。

要求した *ksh-command* は、指定したファイルがメッセージを受信すると常に評価されます。この *kshell* コマンドを評価すると、次に示す環境変数が定義され、受信したメッセージに関する追加情報が提供されます。

DT_TT_MSG	着信メッセージに隠されたハンドルが入っています。
DT_TT_OP	実行しなければならないオペレーションを示す文字列が入っています。文字列は、TTDT_DELETED、TTDT_MODIFIED、TTDT_REVERTED、TTDT_MOVED、TTDT_SAVED のいずれかです。
DT_TT_PATHNAME	メッセージが属するファイルのパス名が入っています。

DT_TT_SAME_EUID_EGID	このプロセスと同じ有効ユーザ ID (euid) および有効グループ ID (egid) で動作するアプリケーションによってメッセージが送信された場合は true に設定されます。
DT_TT_SAME_PROCID	(ttdt_open が返すのと) 同じ procID を持つアプリケーションによってメッセージが送信された場合は、true に設定されます。

コールバックが完了すると、渡される先のメッセージが「消費」(応答、破棄、拒否のいずれか) されたかどうか必ず示されます。コールバックがメッセージを返す場合 (DT_TT_MSG 環境変数に渡されます)、メッセージは消費されていないものと見なされます。メッセージが消費された場合は、コールバックは 0、あるいは tt_error_pointer コマンドが返す値の 1 つを返します。コールバックは値を次のように返します。

```
return $DT_TT_MSG (or) return 0
```

ttdt_file_quit status patterns quit

patterns 引き数で指定するメッセージ・パターンを破棄し、*quit* 引き数に true が設定されている場合は、ttdt_file_join コマンドに渡されたパス名の配信先を登録解除します。*patterns* 引き数は、ttdt_file_join コマンドの呼び出しによって返される値です。

ttdt_file_event status op patterns send

ファイルに関するイベントを通知する ToolTalk 通知を作成し、オプションで送信します。ファイルは、*patterns* の作成時に ttdt_file_join コマンドに渡されたパス名によって示されます。*op* 引き数は、指定したファイルに通知する内容を示すもので、TTDT_MODIFIED、TTDT_SAVED、TTDT_REVERTED のいずれかです。*op* 引き数に TTDT_MODIFIED が設定されている場合、このコマンドは Get_Modified、Save、Revert の各メッセージの処理を *patterns* の作成時に指定されたスコープに登録します。*op* 引き数に TTDT_SAVED または TTDT_REVERTED が設定されている場合は、このファイルの Get_Modified、Save、Revert の各メッセージの処理を登録解除します。*send* 引き数に true が設定されている場合、示されているメッセージが送信されます。

ttdt_Get_Modified pathName scope timeout

Get_Modified 要求を指定したスコープに送信し、応答があるか、指定したタイムアウト（ミリ秒単位）が経過するのを待ちます。Get_Modified 要求は、他の ToolTalk クライアントに、固定表示しようとして保留している *pathname* を変更したかどうかをたずねます。条件文で使用する値を返します。指定したタイムアウト内に肯定応答を受信した場合は値 `true` が返され、そうでない場合は `false` が返されます。

ttdt_Save status pathName scope timeout

Save 要求を指定したスコープに送信し、応答があるか、指定したタイムアウト（ミリ秒単位）が経過するのを待ちます。Save 要求は、処理中の ToolTalk クライアントに、*pathName* 引き数で指定したファイルで保留している変更を保存するかどうかをたずねます。指定したタイムアウト内に肯定応答を受信した場合はステータス `TT_OK` が返され、そうでない場合は標準の `Tt_status` エラー値のうちの 1 つが返されます。

ttdt_Revert status pathName scope timeout

Revert 要求を指定したスコープに送信し、応答があるか、指定したタイムアウト（ミリ秒単位）が経過するのを待ちます。Revert 要求は、処理中の ToolTalk クライアントに、*pathName* 引き数で指定したファイルで保留している変更を破棄するかどうかをたずねます。指定したタイムアウト内に肯定応答を受信した場合はステータス `TT_OK` が返され、そうでない場合は標準の `Tt_status` エラー値のうちの 1 つが返されます。

次のコマンドは通常、`ttdt_file_join` コマンドで登録されたコールバックが使用します。メッセージの消費および破棄を行う機能を提供します。メッセージは、拒否されるか、破棄されるか、応答されることによって消費されます。`tt_error_pointer` はコールバックが使用して、エラー条件を示す戻りポインタを獲得します。

tt_error_pointer variable ttStatus

無効なポインタを表すために ToolTalk が使用する「マジック値」を返します。マジック値は `ttStatus` 値が渡される先に依存して返されます。任意の有効な `Tt_status` 値が指定できます。

tttk_message_destroy status msg

msg 引き数が示すメッセージに格納されている任意のパターンを破棄し、メッセージを破棄します。

`tttk_message_reject status msg msgStatus msgStatusString destroy`

ステータスおよびステータス文字列を指定した要求メッセージに設定し、メッセージを拒否します。*destroy* 引き数に *true* が設定されている場合は、渡されるメッセージを破棄します。このコマンドは、`ttdt_file_join` コマンドで指定されたコールバックがメッセージを消費するための 1 つの方法です。通常、安全にメッセージを破棄するには、メッセージを拒否した後で `tttk_message_destroy` を使用してください。

`tttk_message_fail status msg msgStatus msgStatusString destroy`

ステータスおよびステータス文字列を指定した要求メッセージに設定し、メッセージを破棄します。*destroy* 引き数に *true* が設定されている場合は渡されるメッセージを破棄します。このコマンドは、`ttdt_file_join` コマンドで指定されたコールバックがメッセージを消費するための 1 つの方法です。通常、安全にメッセージを破棄するには、メッセージを拒否した後で `tttk_message_destroy` を使用してください。

`tt_message_reply status msg`

シェル・スクリプトがメッセージを処理し、すべて戻り値で満たされていることを ToolTalk サービスに通知します。ToolTalk サービスは、状態を `TT_HANDLED` に設定して送信プロセスに応答を送信します。通常、メッセージに応答した後で安全にメッセージを破棄するには、`tttk_message_destroy` コマンドを使用してください。

dtksh ユーティリティには、簡易関数のファイルがあります。このファイルは、シェル・プログラマにとって有益なシェル関数を含むシェル・スクリプトです。シェル関数は、dtksh プログラムが頻繁に行わなければならないオペレーションを実行します。これらには、(ヘルプ、エラー、警告などの) ダイアログの作成を簡易化する関数や、いくつかのボタンを容易に作成する関数やフォーム・ウィジェットの子に対する制約リソースの構成を容易にする関数が含まれます。シェル・スクリプトのライタはこれらの簡易関数を必ず使用しなければならないわけではありません。開発者がより短いステップで、よりわかりやすいシェル・スクリプトをより簡単に作成できるようにするために提供されているものです。

シェル・スクリプトがこれらの関数にアクセスする前に、簡易関数が入っているファイルを取り込んでください。簡易関数は `/usr/dt/scripts/DtFuncs.sh` ファイルにあります。これをシェル・スクリプトに取り込むには、次のように記述してください。

```
./usr/dt/lib/dtksh/DtFuncs.dtsh
```

DtkshAddButtons

DtkshAddButtons は、コンポジット・ウィジェットに同じ種類のボタンを 1 つ以上追加します。最もよく使用されるのは、いくつかのボタンをメニュー区画またはメニューバーに追加するときです。

次に使用方法を示します。

```
DtkshAddButtons parent widgetClass label1 callback1  
[label2 callback2 ...]
```

```
DtkshAddButtons [-w] parent widgetClass variable1 label1 callback1 \  
[variable2 label2 callback2 ...]
```

-w オプションは、作成する各ボタンのウィジェット・ハンドルを簡易関数が返すことを指定します。ウィジェット・ハンドルは指定した環境変数に返されます。widgetClass パラメータは次のいずれかに設定できますが、特に指定することがない場合のデフォルト値は XmPushButtonGadget です。

- XmPushButton
- XmPushButtonGadget
- XmToggleButton
- XmToggleButtonGadget
- XmCascadeButton
- XmCascadeButtonGadget

次に例を示します。

```
DtkshAddButtons $MENU XmPushButtonGadget Open do_Open Save do_Save  
Quit exit
```

```
DtkshAddButtons -w $MENU XmPushButtonGadget B1 Open do_Open B2 Save  
do_Save
```

DtkshSetReturnKeyControls

DtkshSetReturnKeyControls は、フォーム・ウィジェット内にテキスト・ウィジェットを構成し、そのフォーム内ではリターン・キーを押してもデフォルト・ボタンが動作せず、フォーム内の次のテキスト・ウィジェットにフォーカスが移動するようにします。一連のテキスト・ウィジェットを含むウィンドウがあり、フォーカスが最後のテキスト・ウィジェットにある間はユーザがリターン・キーを押すまでデフォルト・ボタンが動作しないようにする場合などに便利です。

次に使用方法を示します。

```
DtkshSetReturnKeyControls textWidget nextTextWidget formWidget  
                        defaultButton
```

textWidget パラメータは、リターン・キーが押されるとフォーカスが (*nextTextWidget* パラメータで指定したように) 次のテキスト・ウィジェットに移動するようにウィジェットの構成を設定します。*formWidget* パラメータは、デフォルト・ボタンを含み、2 つのテキスト・ウィジェットの親となるフォームを指定します。*defaultButton* パラメータは、フォーム・ウィジェット内でデフォルト・ボタンとして扱われるコンポーネントを指定します。

次に例を示します。

```
DtkshSetReturnKeyControls $TEXT1 $TEXT2 $FORM $OK  
DtkshSetReturnKeyControls $TEXT2 $TEXT3 $FORM $OK
```

DtkshUnder、DtkshOver、DtkshRightOf、DtkshLeftOf

これらの簡易関数は、フォーム制約条件のクラスの指定を簡易化します。コンポーネントを他のコンポーネントのエッジの 1 つに接続します。ウィジェットのリソース・リストを構築するときに使用します。これは ATTACH_WIDGET 制約条件を使用して実行されます。

次に使用方法を示します。

DtkshUnder widgetId [offset]

DtkshOver widgetId [offset]

DtkshRightOf widgetId [offset]

DtkshLeftOf widgetId [offset]

widgetId パラメータは、現在のコンポーネントを接続するウィジェットを指定します。
offset 値はオプションで、指定しない場合のデフォルト値は 0 です。

次に例を示します。

```
XtCreateManagedWidget BUTTON4 button4 XmPushButton $FORM \  
    labelString:"Exit" \  
    $(DtkshUnder $BUTTON2) \  
    $(DtkshRightOf $BUTTON3)
```

DtkshFloatRight、DtkshFloatLeft、DtkshFloatTop、DtkshFloatBottom

これらの簡易関数は、フォーム制約条件のクラスの指定を簡易化します。コンポーネントを、フォーム内の他のコンポーネントに依存せずに配置する方法を提供します。フォームが伸縮しても、コンポーネントはフォーム内の関連する場所を維持します。他のフォーム制約がコンポーネントに指定されると、コンポーネントも伸縮します。これは ATTACH_POSITION 制約条件を使用して実行されます。

次に使用方法を示します。

DtkshFloatRight [position]

DtkshFloatLeft [position]

DtkshFloatTop [position]

DtkshFloatBottom [position]

オプションの *position* パラメータは、コンポーネントの指定されたエッジが配置される相対位置を指定します。*position* 値はオプションで、指定しない場合のデフォルト値は 0 です。

次に例を示します。

```
XtCreateManagedWidget BUTTON1 button1 XmPushButton $FORM \
    labelString:"Ok" \
    $(DtkshUnder $SEPARATOR) \
    $(DtkshFloatLeft 10) \
    $(DtkshFloatRight 40)
```

DtkshAnchorRight、DtkshAnchorLeft、DtkshAnchorTop、DtkshAnchorBottom

これらの簡易関数は、フォーム制約条件のクラスの指定を簡易化します。フォームが伸縮してもコンポーネントの位置が変わらないように、フォーム・ウィジェットのエッジの1つをコンポーネントに接続する方法を提供します。ただしこのコンポーネントに設定されている他のフォーム制約条件によっては、サイズの伸縮がまだ行われる場合があります。これは ATTACH_FORM 制約条件を使用して実行されます。

次に使用方法を示します。

DtkshAnchorRight [offset]

DtkshAnchorLeft [offset]

DtkshAnchorTop [offset]

DtkshAnchorBottom [offset]

オプションの *offset* パラメータは、コンポーネントを配置するフォーム・ウィジェットのエッジからの距離を指定します。オフセットを指定しない場合は 0 が指定されます。

次に例を示します。

```
XtCreateManagedWidget BUTTON1 button1 XmPushButton $FORM \  
    labelString:"Ok" \  
    $(DtkshUnder $SEPARATOR) \  
    $(DtkshAnchorLeft 10) \  
    $(DtkshAnchorBottom 10)
```


DtkshSpanWidth および DtkshSpanHeight

これらの簡易関数は、フォーム制約条件のクラスの指定を簡易化します。コンポーネントをフォーム・ウィジェットの最大高または最大幅に拡大できるように構成する方法を提供します。この動作は、コンポーネントの 2 つのエッジを（上下は DtSpanHeight で、左右は DtSpanWidth で）フォーム・ウィジェットに接続することによって実行します。コンポーネントは通常、フォーム・ウィジェットのサイズを変更すると必ずサイズ変更されます。すべての接続に ATTACH_FORM 制約が使用されます。

次に使用方法を示します。

DtkshSpanWidth [leftOffset rightOffset]

DtkshSpanHeight [topOffset bottomOffset]

オプションの *offset* パラメータは、コンポーネントを配置するフォーム・ウィジェットのエッジからの距離を指定します。オフセットを指定しない場合は 0 が指定されます。

次に例を示します。

```
XtCreateManagedWidget SEP sep XmSeparator $FORM \  
    $(DtkshSpanWidth 1 1)
```

DtkshDisplayInformationDialog、 DtkshDisplayQuestionDialog、 DtkshDisplayWarningDialog、DtkshDisplayWorkingDialog、DtkshDisplayErrorDialog

これらの簡易関数は、Motif フィードバック・ダイアログのそれぞれのシングル・インスタンスを作成します。要求した型のダイアログのインスタンスがすでに存在している場合は、それが再利用されます。ダイアログの親は環境変数 \$TOPLEVEL から獲得されます。これはシェル・スクリプトによって設定され、その後は変更されません。要求したダイアログのハンドルは、次の環境変数のいずれかに返されます。

- _DTKSH_ERROR_DIALOG_HANDLE
- _DTKSH_QUESTION_DIALOG_HANDLE
- _DTKSH_WORKING_DIALOG_HANDLE
- _DTKSH_WARNING_DIALOG_HANDLE
- _DTKSH_INFORMATION_DIALOG_HANDLE

注 – 独自のコールバックをダイアログ・ボタンに接続する場合、終了するまでダイアログを破壊しないでください。ダイアログを管理しないと、後で再び使用されます。ダイアログの破壊が必要な場合は、関連する環境変数をクリアして、簡易関数がダイアログを再利用しないようにしてください。

次に使用方法を示します。

```
DtkshDisplay<name>Dialog title message [okCallback closeCallback
                                     helpCallback dialogStyle]
```

[了解] ボタンは常に管理され、デフォルト時にはダイアログを管理しません。[取消し] ボタンと [ヘルプ] ボタンは、コールバックが提供されたときだけ管理されます。*dialogStyle* パラメータは、関連するブリテン・ボード・リソースがサポートする標準リソース設定のいずれかを受け入れます。

次に例を示します。

```
DtkshDisplayErrorDialog "Read Error" "Unable to read the file"
    "OkCallback" \
        "CancelCallback" "" DIALOG_PRIMARY_APPLICATION_MODAL
```

DtkshDisplayQuickHelpDialog および DtkshDisplayHelpDialog

これらの簡易関数は、ヘルプ・ダイアログのそれぞれのシングル・インスタンスを作成します。ヘルプ・ダイアログの要求した型のインスタンスがすでに存在している場合は、それが再利用されます。ダイアログの親は環境変数 \$TOPLEVEL から獲得されます。これはシェル・スクリプトによって設定され、その後は変更されません。要求したダイアログのハンドルは、次の環境変数のいずれかに返されます。

- `_DTKSH_HELP_DIALOG_HANDLE`
- `_DTKSH_QUICK_HELP_DIALOG_HANDLE`

注 – ダイアログの破壊が必要な場合は、関連する環境変数をクリアして、簡易関数がダイアログを再利用しないようにしてください。

次に使用方法を示します。

`DtkshDisplay*HelpDialog title helpType helpInformation [locationId]`

パラメータの意味は *helpType* パラメータに指定した値に依存します。意味は次のとおりです。

- *helpType* = `HELP_TYPE_TOPIC`
 - *helpInformation* = ヘルプ・ボリューム名
 - *locationId* = ヘルプ・トピック位置 ID
- *helpType* = `HELP_TYPE_STRING`
 - *helpInformation* = ヘルプ文字列
 - *locationId* = <未使用>
- *helpType* = `HELP_TYPE_DYNAMIC_STRING`
 - *helpInformation* = ヘルプ文字列
 - *locationId* = <未使用>
- *helpType* = `HELP_TYPE_MAN_PAGE`
 - *helpInformation* = マニュアル・ページ名
 - *locationId* = <未使用>

- *helpType* = `HELP_TYPE_FILE`
 - *helpInformation* = ヘルプ・ファイル名
 - *locationId* = <未使用>

次に例を示します。

```
DtkshDisplayHelpDialog "Help On Dtksh" HELP_TYPE_FILE  
    "helpFileName"
```

script_find スクリプト



この付録は、第 4 章「複雑なスクリプト」で説明する script_find の全リストです。スクリプトは、script_find の後にリストされている Find.sticky という二次スクリプトを実行します。ユーザがメイン・スクリプト・ウィンドウの [ヘルプ] ボタンをクリックすると表示できる Find.help というテキスト・ファイルもあります。スクリプトの詳細については第 4 章「複雑なスクリプト」を参照してください。

script_find の全リスト

```
#!/usr/dt/bin/dtksh
set -u

./usr/dt/lib/dtksh/DtFuncs.dtsh

#
# This sample shell script provides a graphical interface to the
# 'find' command. Each time it is executed, it will attempt to
# restore the dialog to the last set of values entered by the user. # When the 'find' command is
# initiated, the output will be displayed # in a dtterm window.
#

#
# Post an# error dialog. The main application window is disabled
# until the error dialog is unposted. The message to be displayed # in the # error dialog is passed in
# as $1
#
```

```

PostErrorDialog()
{
    DtDisplayErrorDialog "Find Error" "$1" \
        DIALOG_PRIMARY_APPLICATION_MODAL
}

#
# This is both the 'Ok' and the 'Apply' callback; in the case of the
# 'Ok' callback, it unposts the main application window, and then
# exits, if the dialog contains valid information. For both 'Ok' and
# 'Apply', the set of search directories is first validated; if any
# of the paths are not valid, then an error dialog is posted.
# Otherwise, the 'find' process is started in a terminal window.
#
OkCallback()
{
    RetrieveAndSaveCurrentValues
    if [ "$SD_VAL" = "" ] ; then
        PostErrorDialog "You must specify a directory to search"
    else
        for i in $SD_VAL ; do
            if [ ! -d $i ] ; then
                MSG="The following search directory does not exist:

                $i"
                PostErrorDialog "$MSG"
                return 1
            fi
        done

        if [ $CB_WIDGET = $OK ] ; then
            XtPopdown $TOPLEVEL
            fi

        CMD="/bin/find $SD_VAL"
        if [ ! "$FNP_VAL" = "" ] ; then
            CMD="$CMD" -name $FNP_VAL"
        fi

        if ! $(XmToggleButtonGetState $T1); then
            CMD="$CMD" -xdev"
        fi
    }

```

```

if $(XmToggleButtonGetState $T3); then
    CMD=$CMD" -hidden"
fi

if $(XmToggleButtonGetState $T4); then
    CMD=$CMD" -follow"
fi

if $(XmToggleButtonGetState $T5); then
    CMD=$CMD" -depth"
fi

case $FSTYPE_VAL in
    $NFS) CMD=$CMD" -fonly nfs" ;;
    $CDFS) CMD=$CMD" -fonly cdfs" ;;
    $HFS) CMD=$CMD" -fonly hfs" ;;
    *) ;;
esac

case $FILETYPE_VAL in
    $REGULAR) CMD=$CMD" -type f" ;;
    $DIRECTORY) CMD=$CMD" -type d" ;;
    $BLOCK) CMD=$CMD" -type b" ;;
    $CHAR) CMD=$CMD" -type c" ;;
    $FIFO) CMD=$CMD" -type p" ;;
    $SYMLINK) CMD=$CMD" -type l" ;;
    $SOCKET) CMD=$CMD" -type s" ;;
    $NET) CMD=$CMD" -type n" ;;
    $MOUNT) CMD=$CMD" -type M" ;;
    $HIDDEN) CMD=$CMD" -type H" ;;
    *) ;;
esac

if $(XmToggleButtonGetState $T2); then
    CMD=$CMD" -print"
fi

/usr/dt/bin/dtterm -title "Find A File" -e /usr/dt/bin/dtexec
    -open -1 $CMD &

if [ $CB_WIDGET = $OK ] ; then

```

```

        exit 0
    fi
fi
}

#
# This function attempt to load in the previous dialog values.
# Each line read from the file is then interpreted as a ksh command.
#
LoadStickyValues()
{
    if [ -r “./Find.sticky” ]; then
        exec 6< “./Find.sticky”
        XtAddInput FID 6 “EvalCmd”
    fi
}

#
# This function is invoked for each line in the ‘sticky’ values file.
# It will evalutate each line as a dtksh command.
#
EvalCmd()
{
    if [ ${#INPUT_LINE} -gt 0 ]; then
        eval “$INPUT_LINE”
    fi

    if [ “$INPUT_EOF” = ‘true’ ]; then
        XtRemoveInput $INPUT_ID
        eval exec $INPUT_SOURCE’<&-’
    fi
}

#
# This function retrieves the current values, and then saves them
# off into a file, so that they can be restored the next time the
# dialog is displayed. It is called anytime the user selects either
# the “Ok” or “Apply” buttons.
#

```



```

RetrieveAndSaveCurrentValues()
{
    XmTextGetString SD_VAL $SD
    XmTextGetString FNP_VAL $FNP
    XtGetValues $FSTYPE menuHistory:FSTYPE_VAL
    XtGetValues $FILETYPE menuHistory:FILETYPE_VAL

    exec 3> “./Find.sticky”
    if [ ! “$SD_VAL” = “” ]; then
        print -u 3 “XmTextSetString \ $SD \”$SD_VAL\””
        print -u 3 “XmTextFieldSetInsertionPosition \ $SD ${#SD_VAL}”
    fi
    if [ ! “$FNP_VAL” = “” ]; then
        print -u 3 “XmTextSetString \ $FNP \”$FNP_VAL\””
        print -u 3 “XmTextFieldSetInsertionPosition \ $FNP ${#FNP_VAL}”
    fi

    case $FSTYPE_VAL in
        $NFS) FST=“\ $NFS” ;;
        $CDFS) FST=“\ $CDFS” ;;
        $HFS) FST=“\ $HFS” ;;
        *) FST=“\ $NODIR” ;;
    esac
    print -u 3 “XtSetValues \ $FSTYPE menuHistory:$FST”

    case $FILETYPE_VAL in
        $REGULAR) FT=“\ $REGULAR” ;;
        $DIRECTORY) FT=“\ $DIRECTORY” ;;
        $BLOCK) FT=“\ $BLOCK” ;;
        $CHAR) FT=“\ $CHAR” ;;
        $FIFO) FT=“\ $FIFO” ;;
        $SYMLINK) FT=“\ $SYMLINK” ;;
        $SOCKET) FT=“\ $SOCKET” ;;
        $NET) FT=“\ $NET” ;;
        $MOUNT) FT=“\ $MOUNT” ;;
        $HIDDEN) FT=“\ $HIDDEN” ;;
        *) FT=“\ $NOTYPE” ;;
    esac
    print -u 3 “XtSetValues \ $FILETYPE menuHistory:$FT”

    if $(XmToggleButtonGetState $T1); then
        print -u 3 “XmToggleButtonSetState \ $T1 true false”
    fi
}

```

```

fi

if $(XmToggleButtonGetState $T2); then
    print -u 3 "XmToggleButtonSetState \"$T2 true false"
fi

if $(XmToggleButtonGetState $T3); then
    print -u 3 "XmToggleButtonSetState \"$T3 true false"
fi

if $(XmToggleButtonGetState $T4); then
    print -u 3 "XmToggleButtonSetState \"$T4 true false"
fi

if $(XmToggleButtonGetState $T5); then
    print -u 3 "XmToggleButtonSetState \"$T5 true false"
fi

exec 3<&-
}

##### Create the Main UI #####

set -f
XtInitialize TOPLEVEL find Dtksh $0 "${ @:-}"
XtSetValues $TOPLEVEL title:"Find Files"

XtCreateManagedWidget FORM form XmForm $TOPLEVEL

XtCreateManagedWidget SDLABEL sdlabel XmLabel $FORM \
    labelString:"Search Directory:" \
    $(DtkshAnchorTop 12) \
    $(DtkshAnchorLeft 10)

XtCreateManagedWidget SD sd XmText $FORM \
    columns:30 \
    value:"." \
    $(DtkshAnchorTop 6) \
    $(DtkshRightOf $SDLABEL 10) \
    $(DtkshAnchorRight 10) \

```

```

navigationType:EXCLUSIVE_TAB_GROUP
XmTextFieldSetInsertionPosition $SD 1

XtCreateManagedWidget FNPLABEL fnpabel XmLabel $FORM \
labelString:"Filename Pattern:" \
$(DtkshUnder $SDLABEL 24) \
$(DtkshAnchorLeft 10)

XtCreateManagedWidget FNP fnp XmText $FORM \
columns:30 \
$(DtkshUnder $SD 8) \
$(DtkshRightOf $FNPLABEL 10) \
$(DtkshAnchorRight 10) \
navigationType:EXCLUSIVE_TAB_GROUP

XtCreateManagedWidget SEP sep XmSeparator $FORM \
separatorType:SINGLE_DASHED_LINE \
$(DtkshUnder $FNP 10) \
$(DtkshSpanWidth)

XtCreateManagedWidget RC rc XmRowColumn $FORM \
orientation:HORIZONTAL \
numColumns:3 \
packing:PACK_COLUMN \
$(DtkshUnder $SEP 10) \
$(DtkshSpanWidth 10 10) \
navigationType:EXCLUSIVE_TAB_GROUP

DtkshAddButtons -w $RC XmToggleButtonGadget \
T1 "Cross Mount Points"      "" \
T2 "Print Matching Filenames" "" \
T3 "Search Hidden Subdirectories" "" \
T4 "Follow Symbolic Links"   "" \
T5 "Descend Subdirectories First" ""

XtCreateManagedWidget SEP2 sep XmSeparator $FORM \
separatorType:SINGLE_DASHED_LINE \
$(DtkshUnder $RC 10) \
$(DtkshSpanWidth)

XmCreatePulldownMenu PANE $FORM pane
DtkshAddButtons -w $PANE XmPushButtonGadget \

```

```

NODIR "no restrictions" "" \
NFS  "nfs"      "" \
CDFS "cdfs"     "" \
HFS  "hfs"      ""

```

```

XmCreateOptionMenu FSTYPE $FORM fstype \
  labelString:"Restrict Search To File System Type:" \
  menuHistory:$NODIR \
  subMenuId:$PANE \
  $(DtkshUnder $SEP2 20) \
  $(DtkshSpanWidth 10 10) \
  navigationType:EXCLUSIVE_TAB_GROUP
XtManageChild $FSTYPE

```

```

XmCreatePulldownMenu PANE2 $FORM pane2
DtkshAddButtons -w $PANE2 XmPushButtonGadget \
  NOTYPE  "no restrictions" "" \
  REGULAR "regular"      "" \
  DIRECTORY "directory"  "" \
  BLOCK   "block special" "" \
  CHAR    "character special" "" \
  FIFO    "fifo"         "" \
  SYMLINK "symbolic link" "" \
  SOCKET  "socket"       "" \
  NET     "network special" "" \
  MOUNT   "mount point"  "" \
  HIDDEN  "hidden directory" ""

```

```

XmCreateOptionMenu FILETYPE $FORM filetype \
  labelString:"Match Only Files Of Type:" \
  menuHistory:$NOTYPE \
  subMenuId:$PANE2 \
  $(DtkshUnder $FSTYPE 10) \
  $(DtkshSpanWidth 10 10) \
  navigationType:EXCLUSIVE_TAB_GROUP
XtManageChild $FILETYPE
XtSetValues $FILETYPE spacing:90

```

```

XtCreateManagedWidget SEP3 sep3 XmSeparator $FORM \
  $(DtkshUnder $FILETYPE 10) \
  $(DtkshSpanWidth)

```

```
XtCreateManagedWidget OK ok XmPushButton $FORM \
    labelString:"Ok" \
    $(DtkshUnder $SEP3 10) \
    $(DtkshFloatLeft 4) \
    $(DtkshFloatRight 24) \
    $(DtkshAnchorBottom 10)
XtAddCallback $OK activateCallback "OkCallback"

XtCreateManagedWidget APPLY apply XmPushButton $FORM \
    labelString:"Apply" \
    $(DtkshUnder $SEP3 10) \
    $(DtkshFloatLeft 28) \
    $(DtkshFloatRight 48) \
    $(DtkshAnchorBottom 10)
XtAddCallback $APPLY activateCallback "OkCallback"

XtCreateManagedWidget CLOSE close XmPushButton $FORM \
    labelString:"Close" \
    $(DtkshUnder $SEP3 10) \
    $(DtkshFloatLeft 52) \
    $(DtkshFloatRight 72) \
    $(DtkshAnchorBottom 10)
XtAddCallback $CLOSE activateCallback "exit 1"

XtCreateManagedWidget HELP help XmPushButton $FORM \
    labelString:"Help" \
    $(DtkshUnder $SEP3 10) \
    $(DtkshFloatLeft 76) \
    $(DtkshFloatRight 96) \
    $(DtkshAnchorBottom 10)
XtAddCallback $HELP activateCallback \
    "DtkshDisplayQuickHelpDialog 'Using The Find Command' HELP_TYPE_FILE \
    './Find.help' "

XtSetValues $FORM \
    initialFocus:$SD \
    defaultButton:$OK \
    cancelButton:$CLOSE \
    navigationType:EXCLUSIVE_TAB_GROUP

DtkshSetReturnKeyControls $SD $FNP $FORM $OK
LoadStickyValues
```

```
XtRealizeWidget $TOPLEVEL
XtMainLoop
```

Find.sticky

次のスクリプト Find.sticky は script_find によって実行されます。Find.sticky は、最後に script_find を実行したときに使用したファイルとディレクトリの名前を記録します。

```
XmTextSetString $SD "/users/dlm"
XmTextFieldSetInsertionPosition $SD 10
XmTextSetString $FNP "elmbug"
XmTextFieldSetInsertionPosition $FNP 6
XtSetValues $FSTYPE menuHistory:$NODIR
XtSetValues $FILETYPE menuHistory:$DIRECTORY
XmToggleButtonSetState $T1 true false
XmToggleButtonSetState $T2 true false
```

Find.help

Find.help は、ユーザがメインの script_find ウィンドウの [ヘルプ] ボタンをクリックしたときに画面に表示されるテキスト・ファイルです。

このダイアログは UNIX の find コマンドのグラフィック・インタフェースを提供します。唯一必要とするフィールドは、検索するディレクトリの名前です。他のフィールドは、オプションです。フィールドに希望する値を設定して、[了解] または [適用] を使用して find 処理を開始させます。find 処理の結果は、dtterm の端末エミュレータに表示されます。

索引

A

app-defaults ファイル, 4

C

CB_CALL_DATA, 12

D

dtksh

 ksh-93 との関係, 1

 定義, 1

dtksh app-defaults ファイル, 4

DtkshAddButtons, 39, 80

DtkshAnchorBottom, 84

DtkshAnchorLeft, 84

DtkshAnchorRight, 84

DtkshAnchorTop, 84

DtkshDisplayErrorDialog, 34, 86

DtkshDisplayHelpDialog, 87

DtkshDisplayInformationDialog, 86

DtkshDisplayQuestionDialog, 86

DtkshDisplayQuickHelpDialog, 87

DtkshDisplayWarningDialog, 86

DtkshDisplayWorkingDialog, 86

DtkshFloatBottom, 83

DtkshFloatLeft, 83

DtkshFloatRight, 83

DtkshFloatTop, 83

DtkshLeftOf, 82

DtkshOver, 82

DtkshRightOf, 82

DtkshSetReturnKeyControls, 81

DtkshSpanHeight, 85

DtkshSpanWidth, 85

DtkshUnder, 82

F

Find.sticky, 98

Form ウィジェットの作成, 37

K

ksh-93, 1

L

libDt コマンド, 70
libDt セッション管理コマンド, 64

M

Motif アプリケーション, 1
Motif コマンド, 51
mwmFunctions, 3

S

script_find, 31, 89
Separator ウィジェットの作成, 38

T

topShadowColor, 2

V

VendorShell, 3

X

XmCreateForm, 10
XmCreateLabel, 11
XmCreateOptionMenu, 40
XmCreatePulldownMenu, 40
XmCreatePushButton, 10
XmNtopShadowColor, 2
XmTextFieldSetInsertionPosition, 35, 38
XmTextSetString, 35
XmToggleButtonSetState, 35
XtAddCallback, 11, 41, 45
XtAddEventHandler, 46
XtAddInput, 21, 46
XtCreateApplicationShell, 9

XtCreateManagedWidget, 9, 15, 37, 38, 39, 41, 47
XtCreatePopupShel, 9
XtCreateWidget, 9
XtDisplay, 47
XtGetValues, 2, 3
XtInitialize, 9, 15, 36
XtMainLoop, 15, 17, 42
XtManageChild, 40
XtRealizeWidget, 15, 17, 42
XtRemoveInput, 21
XtSetValues, 3, 15, 35, 41
Xt イントリンシクスの初期化, 9
Xt イントリンシクス・コマンド, 45

あ

アクション・コマンド, 67
アプリケーション・ヘルプ・コマンド, 63

い

イベント・サブフィールド, 22
イベント・ハンドラ, 19

う

ウィジェット

Form, 37
Separator, 38
設定, 10
トップレベル, 16
トランスレーション, 29
プッシュ・ボタン, 17
ブリテン・ボード, 16

ウィジェットの作成, 9

ウィンドウ・マネージャのクローズ通知, 23

か

- カテゴリ 1, 7
- カテゴリ 2, 7
- カテゴリ 3, 7
- カテゴリ 4, 7
- 簡易関数, 79
- 関数
 - サポートしている, 1

こ

- コマンド, 43
 - CDE アプリケーション・ヘルプ, 63
 - libDt, 70
 - libDt セッション管理, 64
 - Motif, 51
 - Xt インタリンシクス, 45
 - アクション, 67
 - データ型作成, 68
 - メッセージ・セット, 70
 - ローカル化, 64
 - ワークスペース管理, 65
- コンテキスト変数, 19
 - イベント・ハンドラ, 19
 - トランスレーション, 20
 - 入力, 21
 - ワークスペース・コールバック, 20
- コールバック, 11, 18
 - script_find, 34
 - データを渡す, 12
 - 登録, 11
 - ワークスペース, 20
- コールバックの登録, 11

さ

- サポートしていないリソース, 3
- サポートしている関数, 1
- サンプル・スクリプト, 15

し

- 初期化, 16

す

- スクリプト
 - 記述, 15
 - サンプル, 15
 - ローカライズ, 28

せ

- セッション・マネージャの保存状態通知, 24

そ

- 即時戻り値, 8

て

- 定義値, 6
- データ型作成コマンド, 68

と

- トップレベルのウィジェット, 16
- トランスレーション, 20, 29

に

- 入力コンテキスト変数, 21
- 入力モード, 21
- 任意の数のパラメータ, 2

ひ

必要なライブラリ, 1
描画関数, 28

ふ

複雑なスクリプト, 31
プッシュ・ボタン, 18
ブリテン・ボード, 16
ブール値, 6

へ

変数の値, 6

め

メッセージ・セット・コマンド, 70
メニューの作成, 39

も

戻り値, 6
 カテゴリ 1, 7
 カテゴリ 2, 7
 カテゴリ 3, 7
 カテゴリ 4, 7
 即時, 8

り

リソース, 2
 サポートしていない, 3

ろ

ローカライズされたシェル・スクリプト, 28
ローカル化コマンド, 64

わ

ワークスペース管理, 27
ワークスペース管理コマンド, 65
ワークスペース・コールバック, 20