Solaris 共通デスクトップ環境 上級ユーザ及びシステム管理者ガイド

Sun Microsystems, Inc.
901 San Antonio Road
Palo Alto, CA 94303
U.S.A. 650-960-1300

Part No: 805–5136–10
1998 年 11 月

原典: *Solaris Common Desktop Environment: Advanced User's and System Administrator's Guide*

Part No: 805–3900–10

Revision A

Adobe PostScript

目次

はじめに

このマニュアルは、Solaris™ 共通デスクトップ環境 (CDE) の外観と動作をカスタマイズする高度なタスクについて説明します。次の内容に関する章があります。

- システムの初期化、ログイン、およびセッションの初期化のカスタマイズ
- アプリケーションの追加と、アプリケーションとそのデータのインタフェース表現の提供
- デスクトップ・プロセス、アプリケーション、およびネットワーク上のデータの構成
- ウィンドウ管理、印刷、カラー、およびフォントなどのデスクトップ・サービスのカスタマイズ

注 - 「x86」という用語は、一般に Intel 8086 ファミリに属するマイクロプロセッサを意味します。これには、Pentium、Pentium Pro の各プロセッサ、および AMD と Cyrix が提供する互換マイクロプロセッサチップが含まれます。このマニュアルでは、このプラットフォームのアーキテクチャ全体を指すときに「x86」という用語を使用し、製品名では「Intel 版」という表記で統一しています。

対象読者

このマニュアルの対象読者は次のとおりです。

- システム管理者。本書のタスクの多くは、ルートのアクセス権を必要とします。

- デスクトップ・ユーザ・インタフェースを使用しても達成できないカスタマイズを実行したい上級ユーザ。デスクトップでは、大部分の構成ファイルに対してユーザ固有の設定ができます。

このマニュアルを読む前に

まず、次のマニュアルをお読みください。

- 『*Solaris* 共通デスクトップ環境 ユーザーズ・ガイド』
- 『*Solaris* 共通デスクトップ環境への移行』

このマニュアルの構成

このマニュアルは、次の章から構成されています。

第 1 章では、デスクトップ・ログイン・マネージャの外観と動作の構成方法について説明します。

第 2 章では、デスクトップがセッションを格納および取り出す方法と、セッションの起動をカスタマイズする方法について説明します。

第 3 章では、Solaris CDE 起動ファイルと Solaris CDE 起動時に起こる可能性のある問題を説明して、起動時の問題に対する解決策を提案します。

第 4 章では、アプリケーション・マネージャがアプリケーションを収集する方法と、アプリケーションの追加方法について説明します。

第 5 章では、アプリケーションの登録パッケージを作成する方法について説明します。

第 6 章では、カスタム・ログイン構成、複数画面の設定、ネットワーク環境のデスクトップと X 端末、ユーザのドット・ファイルの変更、メール印刷のカスタマイズ、デスクトップ環境の設定、およびエラー・ログのタイプなどの高度な構成について説明します。

第 7 章では、デスクトップ・サービス、アプリケーション、およびネットワーク上のデータを配信する方法について説明します。

第 8 章では、デスクトップ・プリンタを追加および削除する方法と、デフォルト・プリンタの指定方法について説明します。

第 9 章では、デスクトップがアプリケーション、ヘルプ・ファイル、アイコン、およびネットワーク上のその他のデスクトップ・データを検索する方法について説明します。

第 10 章では、アクションとデータ型の概念を紹介し、それらをアプリケーションに対してユーザ・インタフェースを提供するのに使用する方法について説明します。

第 11 章では、アクション作成アプリケーションを使用してアクションとデータ型を作成する方法について説明します。

第 12 章では、データベース構成ファイルを編集してアクション定義を作成する方法について説明します。

第 13 章では、データベース構成ファイルを編集してデータ型定義を作成する方法について説明します。

第 14 章では、デスクトップ・アイコンのためのアイコン・エディタ、命名規則、サイズ、および検索パスの使用方法について説明します。

第 15 章では、新しいシステム共通コントロールとサブパネルの作成、およびその他のパネルのカスタマイズ方法について説明します。

第 16 章では、ウィンドウ、マウス・ボタン割り当て、キーボード割り当て、およびワークスペース・マネージャ・メニューのカスタマイズについて説明します。

第 17 章では、アプリケーションのリソースの設定方法と、デスクトップがフォントとカラーを使用する方法について説明します。

第 18 章では、国際化対応セッションを実行中のシステムのシステム管理タスクについて説明します。

付録 A は、`dtconfig(1)` のマニュアルページです。

マニュアルの注文方法

SunDocs™ プログラムでは、米国 Sun Microsystems™, Inc. (以降、Sun™ とします) の 250 冊以上のマニュアルを扱っています。このプログラムを利用して、マニュアルのセットまたは個々のマニュアルをご注文いただけます。

マニュアルのリストと注文方法については、米国 SunExpress™, Inc. のインターネットホームページ `http://www.sun.com/sunexpress` にあるカタログセクションを参照してください。

表記上の規則

このマニュアルでは、次のような字体や記号を特別な意味を持つものとして使用します。

表 P–1 表記上の規則

字体または記号	意味	例
`AaBbCc123`	コマンド名、ファイル名、ディレクトリ名、画面上のコンピュータ出力、またはコード例を示します。	`.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表示します。 `system%`
`AaBbCc123`	ユーザが入力する文字を、画面上のコンピュータ出力とは区別して示します。	`system%` **`su`** `password:`
AaBbCc123	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` *filename* と入力します。
『 』	参照する書名を示します。	『コードマネージャ・ユーザーズガイド』を参照してください。
「 」	参照する章や節、または強調する単語を示します。	第 5 章「衝突の回避」を参照してください。 この操作ができるのは、「スーパーユーザ」だけです。

表 P–1 表記上の規則 続く

字体または記号	意味	例
[ ]	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
\	枠で囲まれたコード例で、テキストがページ行幅を越える場合、バックスラッシュは継続を示します。	`sun% grep '^#define \ XV_VERSION_STRING'`

ただし AnswerBook2™ では、ユーザが入力する文字と画面上のコンピュータ出力は区別して表示されません。

コード例は次のように表示されます。

- C シェルプロンプト

```
system% command y|n [filename]
```

- Bourne シェルおよび Korn シェルのプロンプト

```
system$ command y|n [filename]
```

- スーパーユーザのプロンプト

```
system# command y|n [filename]
```

[ ] は省略可能な項目を示します。上記の場合、*filename* は省略してもよいことを示します。

| は区切り文字 (セパレータ) です。この文字で分割されている引き数のうち 1 つだけを指定します。

キーボードのキー名は英文で、頭文字を大文字で示します (例: [Shift] キーを押します)。ただし、キーボードによっては [Enter] キーが [Return] キーの動作をします。

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。

第 1 章

ログイン・マネージャの構成

ログイン・マネージャは、ログイン画面の表示、ユーザの認証、ユーザのセッションの起動を行うサーバです。グラフィカル・ログインは、従来のビットマップ・ディスプレイ用のキャラクタ・モードでのログインに代わる、魅力的な方法です。ログイン・サーバが管理するログイン画面は、ログイン・サーバのディスプレイに直接表示することも、ネットワーク上の X 端末またはワークステーションのディスプレイに表示することもできます。

注 - ログイン・サーバを起動、停止、カスタマイズする場合は、必ず root ユーザで行なってください。

この章では、次の内容について説明します。

ログイン・サーバには、次の機能があります。

- 指定しなければビットマップ・ディスプレイに、指定すればローカルなビットマップ・ディスプレイおよびネットワーク上のビットマップ・ディスプレイにログイン画面を表示できます。
- キャラクタ・コンソール・ディスプレイに直接接続できます。
- ネットワーク上の他のログイン・サーバからユーザがログイン画面を表示できるようにする選択画面を表示できます。
- ログイン・サーバへのアクセスをコントロールできます。
- 従来のキャラクタ・モード・ログインにアクセスできます。

ログイン・マネージャが管理するディスプレイは、ログイン・マネージャ・サーバに接続することも、ネットワーク上の X 端末またはワークステーションに接続することもできます。ローカル・ディスプレイでは、ログイン・サーバが自動的に X サーバを起動し、ログイン画面を表示します。X 端末などのネットワーク・ディスプレイでは、ログイン・サーバは X ディスプレイ・マネージャ・プロトコル (XDMCP) 1.0 をサポートします。このプロトコルにより、ディスプレイは、ログイン・サーバがディスプレイにログイン画面を表示するよう要求できます。

ログイン・サーバの起動

ログイン・サーバは通常、システムのブート時に起動されます。コマンド行からもログイン・サーバを起動できます。

- システムのブート時にログイン・サーバが起動するよう設定するには、次のように入力します。

  ```
  /usr/dt/bin/dtconfig -e
  ```

  リブート時に自動的にログイン・サーバが起動します。

  デスクトップ構成ユーティリティ dtconfig の詳細は、付録 A を参照してください。この付録には、dtconfig(1) のマニュアルページがあります。

- コマンド行からログイン・サーバを起動するには、次のように入力します。

  ```
  /usr/dt/bin/dtlogin -daemon; exit
  ```

注 - 一時的に構成をテストするためにログイン・サーバをコマンド行から起動できますが、通常はシステムのブート時にログイン・サーバを起動してください。

ローカル・ディスプレイとネットワーク・ディスプレイの管理

図 1-1 にログイン・サーバの構成を示します。

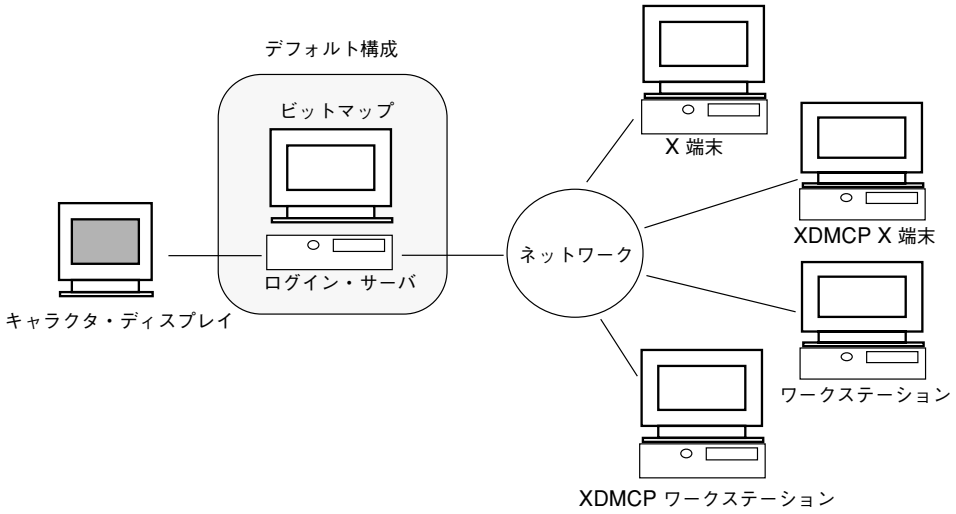


図 1–1　ログイン・サーバの構成例

ログイン・サーバのプロセス ID の検出

デフォルトでは、ログイン・サーバはプロセス ID を /var/dt/Xpid に格納します。

これを変更するため、Dtlogin.pidFile リソースを Xconfig ファイルに設定できます。変更した場合は、ログイン・サーバの起動時に指定したディレクトリが存在しなければなりません。

Xconfig を変更するには、Xconfig を /usr/dt/config から /etc/dt/config へコピーします。/etc/dt/config/Xconfig を変更した後で、次のように入力し、Xconfig をもう一度読み込むようログイン・サーバに通知します。

```
/usr/dt/bin/dtconfig -reset
```

上記は、コマンド kill –HUP *login_server_process_ID* を発行します。

たとえば、ログイン・サーバのプロセス ID を /var/myservers/Dtpid に格納するには、Xconfig ファイルで次のように設定してください。

```
Dtlogin.pidFile: /var/myservers/Dtpid
```

ログイン・サーバを再起動すると、ログイン・サーバはプロセス ID を /var/myservers/Dtpid に格納します。/var/myservers ディレクトリは、ログイン・サーバの起動時に存在しなければなりません。

ローカル・ディスプレイでのログイン画面の表示

起動時、ログイン・サーバは Xservers ファイルを検査して、X サーバを起動する必要があるかどうか、ログイン画面をローカル・ディスプレイまたはネットワーク・ディスプレイに表示するかどうかとその方法を決定します。

Xservers を変更するには、Xservers を /usr/dt/config から /etc/dt/config へコピーします。/etc/dt/config/Xservers を変更した後で、次のように入力し、Xservers をもう一度読み込むようログイン・サーバに通知します。

```
/usr/dt/bin/dtconfig -reset
```

上記は、コマンド kill –HUP *login_server_process_ID* を発行します。

Xservers 行の書式は次のとおりです。

display_name display_class display_type X_server_command

display_name — X サーバに接続するときに使用する接続名 (下記の例では :0) をログイン・サーバに通知します。* (アスタリスク) の値は、*host name*:0 に展開されます。指定した番号は、*X_server_command* 接続番号で指定した番号と一致しなければなりません。

display_class — このディスプレイに固有のリソース (以下の例では Local) を識別します。

display_type — ディスプレイがローカル・ディスプレイとネットワーク・ディスプレイのどちらであるかと、ログイン画面の [コマンド行ログイン] オプションを管理する方法 (以下の例では local@console) を、ログイン・サーバに通知します。

X_server_command — X サーバの起動にログイン・サーバが使用するコマンド行、接続番号、他のオプション (下記の例では /usr/bin/X11/X: 0) を識別します。指定した接続番号は *display_name* で指定した番号と一致しなければなりません。

デフォルトの Xservers 行は、次のようになります。

```
:0 Local local@console /usr/bin/X11/X :0
```

ローカル・ディスプレイなしでのログイン・サーバの実行

ログイン・サーバ・システムにビットマップ・ディスプレイがない場合は、# (ポンド記号) を使用してローカル・ディスプレイの Xservers を注釈行にすることにより、ローカル・ディスプレイなしでログイン・サーバを実行してください。

```
# :0 Local local@console /usr/bin/X11/X :0
```

ログイン・サーバが起動すると、ネットワーク・ディスプレイからの要求を待っているバックグラウンドで実行されます。

ローカル・ディスプレイでのコマンド行ログインへのアクセス

ログイン画面で [コマンド行ログイン] をユーザが選択すると、ログイン・サーバが一時的に X サーバを終了させ、ビットマップ・ディスプレイ端末デバイスで実行中の、従来のコマンド行ログインにアクセスできるようになります。ユーザがログインしてからログアウトした後、または指定したタイムアウト時間が経過した後、ログイン・サーバは X サーバを再起動します。

注 - ネットワーク・ディスプレイでは、[コマンド行ログイン] オプションは使用できません。

---

display_type は、コマンド行ログインの動作をコントロールします。*display_type* の書式は次のとおりです。

- local@*display_ terminal_device*
- local
- foreign

local@*display_terminal_device* を指定すると、ログイン・サーバは、X サーバおよび /dev/*display_terminal_device* が同じ物理デバイス上に存在し、コマンド行ログイン (通常は getty) がそのデバイスで実行されているものと見なします。ユーザが [コマンド行ログイン] を選択すると、X サーバが終了し、/dev/*display_terminal_device* で実行されているコマンド行ログイン (getty) にアクセスできるようになります。

ディスプレイの [コマンド行ログイン] オプションを使用しないようにするには、*display_terminal_device* に none を指定します。デフォルトの *display_terminal_device* は console です。local を指定すると、*display_terminal_device* はデフォルトの console になります。foreign を指定すると、[コマンド行ログイン] オプションは使用できなくなります。

注 - ログイン・サーバをコマンド行から起動した場合は、ローカル・ディスプレイの [コマンド行ログイン] オプションは使用できません。

キャラクタ・ディスプレイ・コンソールの適用

ログイン・サーバ・システムに、直接接続されているキャラクタ・ディスプレイがコンソールとして備えられている場合、*display_terminal_device* に none を設定して、ビットマップ・ディスプレイ・ログイン画面の [コマンド行ログイン] オプションを使用できないようにすることもできます。

コマンド行ログイン (getty) を、キャラクタ・ディスプレイ・コンソールおよびビットマップ・ディスプレイの両方で実行している場合、*display_terminal_device* をビットマップ・ディスプレイのコマンド行ログイン (getty) デバイスに変更できます。

たとえば、ビットマップ・ディスプレイ・コマンド行ログイン (getty) がデバイス /dev/tty01 にある場合、*display_type* を local@tty01 に変更します。

ネットワーク・ディスプレイでのログイン画面の表示

ログイン・サーバは、ネットワーク・ディスプレイからの要求でログイン画面を特定のディスプレイに表示できます。ネットワーク・ディスプレイは通常 X 端末ですが、ワークステーションの場合もあります。

ネットワーク・ディスプレイからの要求を管理するため、ログイン・サーバは X ディスプレイ・マネージャ・プロトコル (XDMCP) 1.0 をサポートします。このプロトコルは、ログイン・サーバがネットワーク・ディスプレイからの要求を受け入れたり拒否したりできるようにします。ほとんどの X 端末に XDMCP が組み込まれています。

ネットワーク・ディスプレイからの XDMCP の直接要求

XDMCP 直接モード (照会モード) を使用するように X 端末を構成する場合は、ログイン・サーバのホスト名を X 端末に通知します。X 端末を起動すると、自動的にログイン・サーバに通信し、ログイン・サーバが X 端末にログイン画面を表示します。XDMCP 直接モード用に X 端末を構成する方法については、X 端末のマニュアルを参照してください。

ほとんどの X サーバは –query オプションもサポートしています。このモードでは、X サーバは X 端末のように動作して、ログイン・サーバ・ホストに直接通信し、X サーバにログイン画面を表示するよう要求します。たとえば、X サーバをワークステーション bridget のビットマップ・ディスプレイで起動すると、ログイン・サーバ anita が X サーバにログイン画面を表示します。

```
X -query anita
```

ネットワーク・ディスプレイからの XDMCP の間接要求

XDMCP 間接モードを使用するように X 端末を構成する場合は、ログイン・サーバのホスト名を X 端末に通知します。X 端末を起動すると、ログイン・サーバに通信し、ログイン・サーバが、ネットワークの他のログイン・サーバ・ホストのリストを選択画面で表示します。このリストからユーザはホストを選択できます。そのホストがユーザの X 端末にログイン画面を表示します。XDMCP 間接モード用に X 端末を構成する方法については、X 端末のマニュアルを参照してください。

直接モード同様、ほとんどの X サーバが –indirect オプションをサポートしています。このオプションを指定すると X サーバは XDMCP 間接モードでログイン・サーバと通信します。

非 XDMCP ネットワーク・ディスプレイの管理

古い X 端末は XDMCP をサポートしていない可能性があります。このような X 端末にログイン・サーバがログイン画面を表示するには、Xservers ファイルに X 端末名を記入します。

ディスプレイはネットワーク上にあるので、*display_name* はホスト名を名前の一部として取り込みます。*display class* は、特定のクラスの X 端末に固有のリソースを指定するのに使用します (X 端末のマニュアルに、X 端末のディスプレイ・クラスが記載されています)。foreign の *display_type* はログイン・サーバに、ログイン・サーバ自身を起動するのではなく、既存の X サーバに接続するよう通知します。この場合、*X_server_command* は指定されません。

例

次に示す Xservers ファイルの行は、ruby と wolfie という 2 つの非 XDMCP X 端末に、ログイン・サーバがログイン画面を表示します。

```
ruby.blackdog.com:0 AcmeXsta foreign
wolfie:0 PandaCo foreign
```

ログイン・サーバへのアクセスのコントロール

デフォルトでは、ログイン・サーバ・ホストにアクセスするネットワークのホストはすべて、ログイン画面を表示するよう要求できます。Xaccess ファイルを変更すると、ログイン・サーバへのアクセスを制限できます。

Xaccess を変更するには、Xaccess を /usr/dt/config から /etc/dt/config へコピーします。/etc/dt/config/Xaccess を変更したら、次のように入力して Xaccess をもう一度読み込むようログイン・サーバに通知します。

```
/usr/dt/bin/dtconfig -reset
```

上記は、コマンド kill –HUP *login_server_process_ID* を発行します。

XDMCP 直接モード

ホストが XDMCP 直接モードによってログイン・サーバに接続しようとすると、ホスト名が Xaccess エントリと比較されて、ホストがログイン・サーバにアクセスできるかどうか決定されます。Xaccess の各エントリは、ワイルドカード * (アスタリスク) と ? (クエスチョン・マーク) を含むホスト名です。* (アスタリスク) は 0 以上の文字に、? (クエスチョン・マーク) は任意の 1 文字に一致します。! (エクスクラメーション・マーク) がエントリの前に付くとアクセスできません。

たとえば、Xaccess に次の 3 つのエントリが含まれているとします。

```
amazon.waterloo.com
 *.dept5.waterloo.com
 !*
```

1 番目のエントリはホスト amazon.waterloo.com から、2 番目のエントリはフルドメイン名が dept5.waterloo.com で終わっている任意のホストからログイン・サーバがアクセスできるようにし、3 番目のエントリはいずれのホストからもアクセスできないようにします。

XDMCP 間接モード

ホストが XDMCP 間接モードによってログイン・サーバに接続しようとすると、ホスト名が Xaccess エントリと比較されて、ホストがログイン・サーバにアクセスできるかどうか決定されます。Xaccess の各エントリは XDMCP 直接モードと同様で、ワイルドカードを含んでいますが、各エントリに CHOOSER 文字列がマークされているところが異なります。次に例を示します。

```
amazon.waterloo.com   CHOOSER BROADCAST
 *.dept5.waterloo.com  CHOOSER BROADCAST
 !*  CHOOSER BROADCAST
```

1 番目のエントリはホスト amazon.waterloo.com から、2 番目のエントリはフルドメイン名が dept5.waterloo.com で終わっている任意のホストからログイン・サーバがアクセスできるようにし、3 番目のエントリはいずれのホストからもアクセスできないようにします。

CHOOSER の後は次のいずれかが続きます。

- BROADCAST
- ホスト名のリスト

BROADCAST は、ログイン・サーバにログイン・サーバ・サブネットワークへ同報通信させて、使用可能なログイン・サーバ・ホストのリストを生成します。ホスト名のリストは、使用可能なログイン・ホストのリストとしてそのリストを使用するように、ログイン・サーバに通知します。次に例を示します。

```
amazon.waterloo.com   CHOOSER shoal.waterloo.com alum.waterloo.com
 *.dept5.waterloo.com  CHOOSER BROADCAST
 !*  CHOOSER BROADCAST
```

amazon.waterloo.com が XDMCP 間接モードによって接続する場合は、shoal と alum を含むリストが表示されます。alice.dept5.waterloo.com を接続する場合は、ログイン・サーバ・サブネットワークで使用可能な全ログイン・サーバ・ホストのリストが表示されます。他の XDMCP 間接モードは否定されます。

ホスト名のリストを指定するもう一つの方法は、ホスト名のリストを含む 1 つ以上のマクロを定義することです。次に例を示します。

```
%list1   shoal.waterloo.com alum.waterloo.com
 amazon.waterloo.com  CHOOSER %list1
```

エラー検査

デフォルトでは、ログイン・サーバは /var/dt/Xerrors ファイルにエラーを記録します。この設定を変更するため、Dtlogin.errorLogFile リソースを Xconfig ファイルに設定できます。指定したディレクトリは、ログイン・サーバの起動時に必ず存在しなければなりません。

たとえば、ログイン・サーバが /var/mylogs/Dterrors ファイルにエラーを記録するには、Xconfig ファイルで次のように設定します。

```
Dtlogin.errorLogFile: /var/mylogs/Dterrors
```

ログイン・サーバが再起動すると、ログイン・サーバは /var/mylogs/Dterrors ファイルにエラーを記録します。/var/mylogs ディレクトリは、ログイン・サーバの起動時に必ず存在しなければなりません。

ログイン・サーバの停止

- システムのブート時にログイン・サーバが起動しないようにするには、次のように入力します。

  ```
  /usr/dt/bin/dtconfig -d
  ```

  この設定により、次回のリブート時にシステムはログイン・サーバを起動しません。

- プロセス ID を強制終了してログイン・サーバを停止するには、次のように入力します。

  ```
  /usr/dt/bin/dtconfig -kill
  ```

  上記は、コマンド kill *login_server_process_ID* を発行します。

注 - ログイン・サーバのプロセスを終了すると、ログイン・サーバが管理するユーザ・セッションは、すべて終了します。

プロセス ID を強制終了して、ログイン・サーバを停止することもできます。ログイン・サーバのプロセス ID は /var/dt/Xpid または Dtlogin.pidFile リソースによって Xconfig に指定したファイルに格納されます。

ログイン・サーバを強制終了したときにデスクトップにログインすると、デスクトップ・セッションはすぐに終了します。

ログイン画面

ログイン・サーバが表示するログイン画面は、従来のキャラクタ・モードでのログイン画面に代わるもので、キャラクタ・モードでのログインを超える能力を備えています。

図 1–2　デスクトップ・ログイン画面

キャラクタ・モードでのログイン同様、ユーザ名の後にパスワードを入力します。認証されれば、ログイン・サーバはデスクトップ・セッションを起動します。デスクトップ・セッションを終了すると、ログイン・サーバは新しいログイン画面を表示し、処理が再開されます。

次のようなログイン画面のカスタマイズができます。

- ログイン画面表示の変更
- X サーバ権限の設定
- デフォルト言語の変更
- ログイン画面の表示前のコマンドの発行
- ログイン画面の [言語] メニューの内容変更
- ユーザのセッションを起動するコマンドの指定
- ユーザのデスクトップ・セッションの起動前のコマンドの発行

■ ユーザセッションの終了後のコマンドの発行

上記はいずれも、全ディスプレイまたはディスプレイごとに実行できます。

ログイン画面表示の変更

ログイン画面表示をカスタマイズするため、ロゴまたはグラフィック、ウェルカム・メッセージ、およびフォントを変更できます。

Xresources を変更するには、Xresources を /usr/dt/config/*language* から /etc/dt/config/*language* にコピーします。次回からは、すべての変更を反映したログイン画面が表示されます。ログイン画面を強制的に再表示させるには、ログイン画面の [オプション] メニューから [ログイン画面のリセット] を選択します。

Xresources ファイルのリソース指定によって決定されるログイン画面の属性は次のとおりです。

Dtlogin*logo*bitmapFile — ロゴ・イメージとして表示するビットマップ・ファイルまたはピックスマップ・ファイル

Dtlogin*greeting*persLabelString — 個人用ウェルカム・メッセージ

Dtlogin*greeting*labelString — ウェルカム・メッセージ

Dtlogin*greeting*fontList — ウェルカム・メッセージ用フォント

Dtlogin*labelFont — プッシュ・ボタンおよびラベル用フォント

Dtlogin*textFont — ヘルプ・メッセージおよびエラー・メッセージ用フォント

Dtlogin**language**languageName — ロケール名 *language* 用の選択テキスト

▼ ロゴを変更するには

♦ Dtlogin*logo*bitmapFile リソースを Xresources に設定します。

ロゴは、カラー・ピックスマップ・ファイルまたはビットマップ・ファイルです。

次の例は、ロゴに Mylogo ビットマップを使用します。

```
Dtlogin*logo*bitmapFile: /usr/local/lib/X11/dt/bitmaps/Mylogo.bm
```

▼ ウェルカム・メッセージを変更するには

デフォルトでは、ログイン・サーバはログイン画面にメッセージ Welcome to *hostname* を表示します。このメッセージは、次の方法で変更できます。

- Dtlogin*greeting*labelString リソースを Xresources に設定します。

  labelString リソースの値には、ログイン・サーバ・ホスト名に置き換える %*LocalHost*% と、X サーバ・ディスプレイ名に置き換える %*DisplayName*% を指定できます。

次の例は、ウェルカム・メッセージを Here's *host name*! に変更します。

```
Dtlogin*greeting*labelString: Here's %LocalHost%!
```

ユーザ名を一度入力すると、ログイン・サーバはデフォルトで Welcome *username* というメッセージを表示します。このメッセージは Dtlogin*greeting*persLabelString リソースを Xresources に設定すると変更できます。persLabelString の値には、*username* に置き換える %s を指定できます。

次の例は、個人用ウェルカム・メッセージを Hello *username* に変更します。

```
Dtlogin*greeting*persLabelString: Hello %s
```

▼ フォントを変更するには

ログイン画面で使用するフォントを、次のフォント・リソースのいずれかを Xresources に設定することにより変更できます。

Dtlogin*greeting*fontList — ウェルカム・メッセージ用フォント

Dtlogin*labelFont — プッシュ・ボタンとラベル用フォント

Dtlogin*textFont — ヘルプ・メッセージとエラー・メッセージ用フォント

使用可能なフォントを一覧表示するには、次のように入力します。

```
xlsfonts [-options] [–fn pattern]
```

次の例は、ウェルカム・メッセージを大きいフォントで表示します (指定する値は、必ず一行に入れてください)。

```
Dtlogin*greeting*fontList: -dt-interface \
 system-medium-r-normal-xxl*-*-*-*-*-*-*-*-*:
```

▼ 各言語を表示するためのテキストを提供するには

ログイン画面の [言語] メニューに、ロケール名のデフォルト・ディスプレイではなく、ロケールごとのテキストを表示するには、Dtlogin*_language_*languageName リソースを Xresources に設定します。

```
Dtlogin*En_US*languageName: American
```

ロケール名 En_US ではなく、テキスト American が表示されます。

ログイン画面動作の変更

ログイン画面の動作をカスタマイズするには、Xconfig ファイルに指定したリソースを変更します。

Xconfig を変更するには、Xconfig を /usr/dt/config から /etc/dt/config にコピーします。/etc/dt/config/Xconfig を変更したら、次のように入力して Xconfig をもう一度読み込むようログイン・サーバに通知します。

```
/usr/dt/bin/dtconfig -reset
```

上記は、コマンド kill –HUP *login_server_process_ID* を発行します。

Xconfig ファイルに指定するリソースは次のとおりです。

Dtlogin*authorize — Xaccess ファイル仕様

Dtlogin*environment — X サーバ環境

Dtlogin*language — デフォルト言語

Dtlogin*languageList — ログイン画面の [言語] メニューの言語リスト

Dtlogin*resources — Xresources 仕様

Dtlogin*setup — Xsetup ファイル仕様

Dtlogin*startup — Xstartup ファイル仕様

Dtlogin*session — Xsession ファイル仕様

Dtlogin*failsafeClient — Xfailsafe スクリプト仕様

Dtlogin*reset — Xreset スクリプト仕様

Dtlogin*userPath — Xsession および Xfailsafe 用 PATH 環境変数

Dtlogin*systemPath — Xsetup、Xstartup、Xfailsafe 用 PATH 環境変数

Dtlogin*systemShell — Xsetup、Xstartup、Xfailsafe 用 SHELL 環境変数

Dtlogin.timeZone — 全スクリプト用タイムゾーン

ディスプレイごとのログイン画面動作の変更

次の例では、Xconfig リソースを変更して、全ディスプレイのログイン画面動作を変更します。リストで * (アスタリスク) の付いたリソースは、ディスプレイごとに指定できます。これにより、あるディスプレイのログイン画面動作のカスタマイズを指定できます。特定のディスプレイのリソースを指定するには、リソースを Dtlogin**displayName***resource* と指定します。たとえば、ディスプレイ expo:0 のユーザによるアクセス・コントロールをオフにして、他のディスプレイはそのままオンにする場合は、次のように指定します。

```
Dtlogin*expo_0*authorize: False
```

注 - ディスプレイ名の : (コロン) や . (ピリオド) などの特殊文字は、_ (下線) に置き換えることができます。

X サーバ・アクセスの変更

デフォルトでは、ログイン・サーバが、ユーザごとに X サーバのアクセスをコントロールできるようにします。ログイン・サーバは、*HomeDirectory*/.Xauthority ファイルに格納され保護されている権限データに基づきます。このファイルを読み込めるユーザだけが X サーバに接続できます。通常、これが望ましい X サーバ・アクセス・コントロールの方法です。

ユーザベースのアクセス・コントロールの代わりに、ホストベースのアクセス・コントロールも可能です。この方法を使用すると、ホストが X サーバへのアクセスを与えられている場合、そのホストのすべてのユーザが X サーバに接続できます。ホストベースのコントロールを使用するのは、次のような理由からです。

- 古い R2 および R3 の X のクライアントは、ユーザベースのアクセス・コントロールでは X サーバに接続できない
- 安全性が確立されていないネットワークでは、ネットワークの X クライアントと X サーバとの間で渡される認証データを盗まれる可能性がある

Xconfig Dtlogin*authorize リソースは、ユーザベースの X サーバ・アクセス・コントロールを使用することをログイン・サーバに通知します。ホストベース

のアクセス・コントロールを使用する場合は、次のように承認リソース値を False に変更します。

```
Dtlogin*authorize: False
```

X サーバ環境を変更するには

ログイン・サーバによって起動されたときに 1 つ以上の環境変数と値を X サーバに指定する場合は、Xconfig の Dtlogin*environment リソースを使用して指定できます。

```
Dtlogin*environment: VAR1=foo VAR2=bar
```

たとえば上記は、ローカルな X サーバ・プロセスで変数 *VAR1* および *VAR2* を使用できるようにします。これらの変数も、Xsession および Xfailsafe スクリプトへエクスポートされます。

デフォルト言語を変更するには

ログイン画面からデスクトップにログインする場合、ユーザ・セッションは、[オプション] メニューの [言語] サブメニューから選択されたロケールで実行されます。言語を選択しない場合、ログイン・サーバはデフォルト言語を使用します。Xconfig の Dtlogin*language リソースを次のように設定することにより、デフォルト言語の値をコントロールできます。

```
Dtlogin*language: Ja_JP
```

システムのマニュアルを調べて、システムにインストールされている言語を判定してください。

ログイン画面の［言語］メニューの内容を変更するには

デフォルトでは、ログイン・サーバは、システムにインストールされたすべてのロケールのリストが入っているログイン画面の [言語] メニューを作成します。そのリストからロケールを選択すると、ログイン・サーバは選択されたロケールでログイン画面を再表示します。その後ログインすると、ログイン・サーバはそのロケールでデスクトップ・セッションを起動します。

Xconfig の Dtlogin*languageList リソースを変更することにより、独自の言語のリストを指定できます。

```
Dtlogin*languageList: En_US De_DE
```

上記のように指定すると、ログイン・サーバは En_US および De_DE だけをログイン画面の [言語] メニューに表示します。

ログイン画面が表示される前にコマンドを発行する

X サーバの起動後でログイン画面が表示される前に、ログイン・サーバは Xsetup スクリプトを実行します。Xsetup は root の権限に合わせて実行され、ログイン画面を表示する前に実行する必要のあるコマンドを発行します。

Xsetup を変更するには、Xsetup を /usr/dt/config から /etc/dt/config へコピーします。次にログイン画面が表示されるときは、変更された Xsetup が実行されます。

ユーザ・セッション起動前のコマンドの発行

ユーザ名とパスワードを入力し、それが認証された後で、ユーザ・セッションが起動される前に、ログイン・サーバは Xstartup スクリプトを実行します。Xstartup は root の権限に合わせて実行され、ユーザ・セッションを起動する前に root として実行する必要のあるコマンドを発行します。

Xstartup を変更するには、Xstartup を /usr/dt/config から /etc/dt/config へコピーします。次にログイン画面が表示されるときは、変更された Xstartup が実行されます。

デスクトップ・セッションの起動

デフォルトでは、ログイン・サーバは Xsession スクリプトを実行することによりユーザ・セッションを起動します。Xsession はユーザの権限に合わせて実行され、デスクトップの起動に必要なコマンドを発行します。

注 - Xsession スクリプトは、直接変更しないでください。

ユーザのデスクトップ・セッション起動のカスタマイズの方法については、第 2 章を参照してください。

復旧セッションの起動

ログイン画面の [オプション] メニューの [セッション] サブメニューから [復旧セッション] を選択する場合、ログイン・サーバは Xfailsafe スクリプトを実行します。Xfailsafe はユーザの権限に合わせて実行され、アイコン化されたウィンドウ (通常は [端末] ウィンドウとオプションのウィンドウ・マネージャ) 環境の起動に必要なコマンドを発行します。

Xfailsafe を変更するには、Xfailsafe を /usr/dt/config から /etc/dt/config へコピーします。次にログイン画面が表示されるときは、変更された Xfailsafe が実行されます。

ユーザのセッションが終了した後で

ユーザがデスクトップまたは復旧セッションを終了した後、ログイン・サーバは Xreset スクリプトを実行します。Xreset は root の権限に合わせて実行され、ユーザセッションの終了後に root として実行する必要のあるコマンドを発行します。

Xreset を変更するには、Xreset を /usr/dt/config から /etc/dt/config へコピーします。次にログイン画面が表示されるときは、変更された Xreset が実行されます。

ログイン・サーバ環境

ログイン・サーバは、Xsetup、Xstartup、Xsession、Xfailsafe、Xreset スクリプトへエクスポートする環境を提供します。この環境は表 1–1 で説明します。これら以外の変数も、ログイン・サーバによってエクスポートされることがあります。

表 1–1 ログイン・サーバ環境

環境変数	Xsetup	Xstartup	Xsession Xfailsafe	Xreset	説明
LANG	X	X	X	X	デフォルト言語または選択された言語
XAUTHORITY	X	X	X	X	(省略可能) 代替 X 許可ファイル

表 1–1 ログイン・サーバ環境　続く

環境変数	Xsetup	Xstartup	Xsession Xfailsafe	Xreset	説明
PATH	X	X	X	X	Dtlogin*userPath リソース (Xsession、Xfailsafe) または Dtlogin*systemPath リソース (Xsetup、Xstartup、Xreset) の値
DISPLAY	X	X	X	X	X サーバ接続番号
SHELL	X	X	X	X	/etc/passwd に指定されたシェル (Xsession、Xfailsafe) または Dtlogin*systemShell リソース (Xsetup、Xstartup、Xreset)
TZ	X	X	X	X	Dtlogin.timeZone リソースまたはシステムで決められたタイムゾーンの値
USER		X	X	X	ユーザ名
HOME		X	X	X	/etc/passwd に指定したホーム・ディレクトリ
LOGNAME		X	X	X	ユーザ名

ユーザ・パスまたはシステム・パスの変更

ログイン・サーバは、Xsession スクリプトと Xfailsafe スクリプトの実行時に PATH 環境変数を設定します。これらのスクリプトの代替パスを指定できます。

ユーザ・パスを変更するには

♦ Xconfig に Dtlogin*userPath リソースを設定します。次に例を示します。

```
Dtlogin*userPath:/usr/bin:/etc:/usr/sbin:/usr/ucb:/usr/bin/X11
```

システム・パスを変更するには

- Xconfig に Dtlogin*systemPath リソースを設定します。次に例を示します。

```
Dtlogin*systemPath: /usr/bin/X11:/etc:/bin:/usr/bin:/usr/ucb
```

▼ システム・シェルを変更するには

ログイン・サーバは、Xsetup、Xstartup、Xfailsafe スクリプトの実行時に SHELL 環境変数を設定します。デフォルトは /bin/sh です。これらのスクリプトに別のシェルを指定する場合は、次のように Xconfig に Dtlogin*systemShell リソースを設定します。

```
Dtlogin*systemShell: /bin/ksh
```

▼ タイムゾーンを変更するには

ログイン・サーバは、Xsetup、Xstartup、Xsession、Xfailsafe、Xreset スクリプトの実行時に TZ 環境変数を設定します。デフォルト値はシステムから派生するので、通常はこの動作を変更する必要はありません。これらのスクリプトに別のタイムゾーンを指定する場合は、次のように Xconfig に Dtlogin.timeZone リソースを設定します。

```
Dtlogin.timeZone: CST6CDT
```

ログイン・マネージャの管理

ログイン・サーバが起動すると、1 つの dtlogin プロセスが起動します。dtlogin プロセスは Xconfig ファイルを読み込んで、最初のログイン・サーバ構成を判定し、他のログイン・サーバ構成ファイルを配置します。それから Xservers ファイルを読み込んで、明示的に管理するディスプレイがあるかどうかを調べ、Xaccess ファイルを読み込んでログイン・サーバへのアクセスをコントロールします。

ログイン・サーバが、ローカル・ディスプレイを管理する必要があることを Xservers で知ると、Xservers ファイルで指定したように X サーバを起動し、そのディスプレイにログイン画面を表示します。

ログイン・サーバが、ネットワーク・ディスプレイを管理する必要があることを Xservers で知ると、X サーバがすでに指定したディスプレイ名で実行されているものと見なし、そのディスプレイにログイン画面を表示します。

その後、ログイン・サーバはネットワークからの XDMCP 要求を待ちます。

管理されている各ディスプレイに対して、ログイン・サーバはそのディスプレイ用の新しい dtlogin プロセスを最初に作成します。つまり、ログイン・サーバが *n* 個のディスプレイを管理している場合、*n+1* 個の dtlogin プロセスが存在します。ログイン・サーバは Xsetup スクリプトを実行し、Xresources ファイルを読み込んでから、dtgreet を実行してログイン画面を表示します。ユーザ名とパスワードを入力して認証されると、ログイン・サーバは Xstartup スクリプトを実行してから Xsession または Xfailsafe スクリプトを実行します。セッションを終了すると、ログイン・サーバは Xreset スクリプトを実行します。

ログイン・サーバが XDMCP 間接モード要求を獲得すると、dtchooser を実行してディスプレイのログイン・サーバ・ホストのリストを表示します。リストからホストを選択すると、そのホストのログイン・サーバがディスプレイを管理します。

Xaccess、Xconfig、Xfailsafe、Xreset、*language*/Xresources、Xservers、Xsetup、Xstartup 構成ファイルについては、デフォルトではログイン・サーバが最初に /etc/dt/config を調べ、次に /usr/dt/config を調べて、最初に見つけたファイルを使用します。

ログイン・マネージャ・ファイル

ログイン・マネージャ・ファイルのデフォルトの位置は次のとおりです。

/usr/dt/bin/dtlogin — ログイン・サーバおよびディスプレイ・マネージャ

/usr/dt/bin/dtgreet — ディスプレイ用ログイン画面の表示

/usr/dt/bin/dtchooser — ディスプレイ用選択画面の表示

/usr/dt/bin/Xsession — デスクトップ・セッションの起動

/usr/dt/config/Xfailsafe — 復旧セッションの起動

/usr/dt/config/Xconfig — ログイン・サーバ構成ファイル

/usr/dt/config/Xservers — ログイン・サーバ・ディスプレイ記述ファイル

/usr/dt/config/Xaccess — ログイン・サーバ・アクセス記述ファイル

/usr/dt/config/*language*/Xresources — レイアウト・リソースの表示

/usr/dt/config/Xsetup — セットアップ・ファイルの表示

/usr/dt/config/Xstartup — セッション起動ファイル

/usr/dt/config/Xreset — セッション開始後のリセット・ファイル

/var/dt/Xpid — ログイン・サーバのプロセス ID

/var/dt/Xerrors — ログイン・サーバのエラー記録ファイル

第 2 章

セッション・マネージャの構成

セッション・マネージャは、デスクトップを起動し、実行中のアプリケーション、カラー、フォント、マウス動作、音量、およびキーボード・クリックを自動的に保存および復元します。この章の構成は次のとおりです。

セッション・マネージャを使用して、次の作業を実行できます。

- すべてのデスクトップ・ユーザの初期セッションをカスタマイズする
- すべてのデスクトップ・ユーザの環境とリソースをカスタマイズする
- セッション起動メッセージを変更する
- セッション起動ツールとデーモンのパラメータを変更する
- すべてのユーザのデスクトップ・カラーの使用法をカスタマイズする

セッションとは

セッションとは、ユーザのデスクトップに存在するアプリケーション、設定、およびリソースのコレクションです。セッションの管理は、セッション・マネージャによるセッションの保存および復元を可能にする規約とプロトコルのセットです。

システムにログインすると、前回ログオフした時に提供されていたのと同じ実行中のアプリケーション、設定、およびリソースのセットを得ることができます。デスクトップに最初にログインした時は、デフォルトの初期セッションが読み込まれます。その後、セッション・マネージャは現在のセッションとホーム・セッションの概念をサポートします。

初期セッション

デスクトップに最初にログインしたときに、セッション・マネージャはシステムのデフォルト値を使用して初期セッションを作成します。デフォルトでは、ファイル・マネージャと、ヘルプ・ボリュームのデスクトップの紹介が起動されます。

現在のセッション

保存済みのホーム・セッション、保存済みの現在のセッション、またはシステム・デフォルトの初期セッションからログイン時に復元されたものかどうかにかかわらず、実行中のセッションは常に現在のセッションと見なされます。スタイル・マネージャの [起動] 設定に基づきセッションを終了すると、セッション・マネージャは現在のセッションを自動的に保存します。もう一度デスクトップにログインすると、セッション・マネージャは前に保存した現在のセッションを再起動します。これは、最後にログアウトしたときと同じ状態にデスクトップが復元されることを意味します。

ホーム・セッション

ログアウトしたときの状態に関係なく、ログインするたびにデスクトップを同じ状態に復元させることもできます。現在のセッションの状態を保存し、その後スタイル・マネージャの [起動] 設定を使用して、ユーザがログインするたびにセッション・マネージャにそのセッションを起動させることができます。

ディスプレイに固有のセッション

特定のディスプレイに対して固有のセッションを実行するために、ディスプレイに固有のセッションを作成できます。作成するために、ユーザは *HomeDirectory*/.dt/sessions ディレクトリを *HomeDirectory*/.dt/*display* にコピーできます。この場合、*display* は実際に存在する修飾されていないホスト名です (た

とえば、pablo:0 は有効で、pablo.gato.com:0 や unix:0 は無効です)。ユーザがディスプレイ pablo:0 にログインすると、セッション・マネージャはそのディスプレイに固有のセッションを起動します。

セッションの起動

セッション・マネージャは、/usr/dt/bin/Xsession によって起動されます。ログイン・マネージャを使用してログインすると、Xsession がデフォルトとして起動されます。

オプションとして、従来のキャラクタ・モード (getty)・ログインを使用してログインし、xinit などの X サーバを起動するツールを使用して、セッション・マネージャを手動で起動できます。たとえば、xinit /usr/dt/bin/Xsession のように指定します。

セッションの起動方法

セッション・マネージャは起動すると、次の手順に従ってセッションを起動します。

1. ***HomeDirectory***/.dtprofile スクリプトを参照します。
2. Xsession.d スクリプトを参照します。
3. ウェルカム・メッセージを表示します。
4. デスクトップ検索パスを設定します。
5. 使用可能なアプリケーションを集めます。
6. **(省略可能)** ***HomeDirectory***/.profile または ***HomeDirectory***/.login を参照します。
7. **ToolTalk**™ メッセージ・デーモンを起動します。
8. セッション・リソースを読み込みます。

9. カラー・サーバを起動します。

10. ワークスペース・マネージャを起動します。

11. セッション・アプリケーションを起動します。

次の節では、上記の手順について説明します。

.dtprofile スクリプトの参照

セッションの起動時に、Xsession スクリプトは、*HomeDirectory*/.dtprofile スクリプトを参照します。*HomeDirectory*/.dtprofile スクリプトは、セッションに対して環境変数を設定できる /bin/sh または /bin/ksh スクリプトです。環境変数の設定の詳細は、33ページの「追加セッション起動のカスタマイズ」を参照してください。

デスクトップに最初にログインしたときなど *HomeDirectory*/.dtprofile スクリプトが存在しない場合、Xsession はデスクトップのデフォルトの sys.dtprofile を *HomeDirectory*/.dtprofile にコピーします。

デスクトップのデフォルトは、/usr/dt/config/sys.dtprofile です。sys.dtprofile スクリプトをカスタマイズするには、sys.dtprofile を /usr/dt/config から /etc/dt/config にコピーし、新規ファイルを編集します。

Xsession.d スクリプトの参照

HomeDirectory/.dtprofile スクリプトを参照した後で、Xsession スクリプトは Xsession.d スクリプトを参照します。これらのスクリプトは追加する環境変数を設定し、ユーザのセッションに対して任意のデーモンを起動するために使用されます。デフォルトの Xsession.d スクリプトは次のとおりです。

0010.dtpaths — カスタマイズ可能なデスクトップ検索パスを文書化します。

0020.dtims — 任意の入力方式サーバを起動します。

0030.dttmpdir — ユーザごと、セッションごとに一時ディレクトリを作成します。

0040.xmbind — デスクトップ・デフォルトに $XMBINDDIR を設定します。

Xsession.d には、追加されたベンダ固有のスクリプトがあることもあります。

Xsession は最初に、/etc/dt/config/Xsession.d ディレクトリにあるすべてのファイルを参照し、続いて /usr/dt/config/Xsession.d ディレクトリにあるファイルを参照します。

デスクトップのデフォルトの Xsession.d スクリプトは、/usr/dt/config/Xsession.d ディレクトリに位置付けられます。Xsession.d スクリプトをカスタマイズするには、スクリプトを /usr/dt/config/Xsession.d から /etc/dt/config/Xsession.d にコピーし、新規ファイルを編集します。このタスクを実行するには、実行権を持っていなければなりません。

また、Xsession がユーザ独自のスクリプトを自動的に参照するには、そのスクリプトを /etc/dt/config/Xsession.d にコピーします。

注 - Xsession.d スクリプトを変更または作成する場合、コマンドの所要時間はセッションの起動時間に直接影響を与えるため、発行したフォアグラウンド・コマンドが短期のものであることを確認します。フォアグラウンド・コマンドが終了していないセッションの起動はハングアップします。セッションの継続中に実行を続行したい Xsession.d スクリプトで実行されるコマンドは、バックグラウンドで実行されます。

ウェルカム・メッセージの表示

HomeDirectory/.dtprofile スクリプトと Xsession.d スクリプトを参照した後、Xsession は画面をカバーするウェルカム・メッセージを表示します。表示されるウェルカム・メッセージは、カスタマイズしたり、メッセージを完全にオフにしたりできます。dthello クライアントはメッセージを表示するのに使用します。

メッセージ・テキストを変更するには、dtstart_hello[0] 変数を変更することにより dthello オプションを変更します。

dtstart_hello[0] を変更するには、新しい値を設定する /etc/dt/config/Xsession.d スクリプトを作成します。すべてのユーザにその日のメッセージを表示するには、実行可能な sh または ksh スクリプト (たとえば、/etc/dt/config/Xsession.d/myvars) を作成し、dtstart_hello[0] を次のように設定します。

```
dtstart_hello[0]="/usr/dt/bin/dthello -file /etc/motd &"
```

同様に、ユーザは *HomeDirectory*/.dtprofile に dtstart_hello[0] を設定することにより、それらのセッションのウェルカム・メッセージを変更できます。

ウェルカム・メッセージをオフにするには、dtstart_hello[0]="" を設定します。

dthello の詳細は、dthello(1X) のマニュアル・ページを参照してください。

デスクトップ検索パスの設定

デスクトップ検索パスは、dtsearchpath によるログイン時に作成されます。dtsearchpath によって使用される環境変数のカテゴリは 2 種類あります。

入力変数 — 値がシステム管理者かエンド・ユーザによって設定されるシステム共通環境変数と個人用環境変数

出力変数 — dtsearchpath によって作成され、値が割り当てられた変数。各変数の値はデスクトップ・セッションの検索パスです。

dtsearchpath のコマンド行オプションを変更するには、dtstart_searchpath 変数を変更します。すべてのユーザの dtstart_searchpath 変数を変更するには、実行可能な sh または ksh スクリプト (たとえば、/etc/dt/config/Xsession.d/myvars) を作成し、dtstart_searchpath を次のように設定します。

```
dtstart_searchpath="/usr/dt/bin/dtsearchpath"
```

同様に、*HomeDirectory*/.dtprofile に dtstart_searchpath を設定することによってのみ、ユーザのセッションの dtsearchpath オプションを変更できます。

dtsearchpath の詳細は、第 9 章を参照してください。dtsearchpath オプションの詳細は、dtsearchpath(1) のマニュアル・ページを参照してください。

使用可能なアプリケーションの収集

デスクトップ検索パスの設定の次の手順は、dtappgather を使用して使用可能なアプリケーションを収集することです。dtappgather のコマンド行オプションを変更するには、dtstart_appgather 変数を変更します。すべてのユーザの dtstart_appgather 変数を変更するには、実行可能な sh または ksh スクリプト (たとえば、/etc/dt/config/Xsession.d/myvars) を作成し、dtstart_appgather を次のように設定します。

```
dtstart_appgather="/usr/dt/bin/dtappgather &"
```

同様に、*HomeDirectory*/.dtprofile に dtstart_appgather を設定することによって、ユーザのセッションのみの dtappgather オプションを変更できます。

dtappgather オプションの詳細は、dtappgather(4) のマニュアル・ページを参照してください。

オプションとしての .profile または .login スクリプトの参照

Xsession は、従来の *HomeDirectory*/.profile スクリプトまたは *HomeDirectory*/.login スクリプトを参照できます。デフォルトではこの機能は使用できません。Xsession に .profile スクリプトか .login スクリプトを参照するように通知するには、DTSOURCEPROFILE に true を設定します。

すべてのユーザの DTSOURCEPROFILE を変更するには、新しい値を設定する /etc/dt/config/Xsession.d スクリプトを作成します。すべてのユーザに対して DTSOURCEPROFILE に true を設定するには、実行可能な sh または ksh スクリプト (たとえば、/etc/dt/config/Xsession.d/myvars) を作成し、DTSOURCEPROFILE を次のように設定します。

```
DTSOURCEPROFILE=true
```

同様に、*HomeDirectory*/.dtprofile で DTSOURCEPROFILE に true を設定することによって、ユーザのセッションの DTSOURCEPROFILE を変更できます。

ToolTalk メッセージ・デーモンの起動

ToolTalk メッセージ・デーモンの ttsession により、互いに依存しないアプリケーションは、お互いについて直接認識していなくても交信できます。アプリケーションは、お互いに交信できるように ToolTalk メッセージを作成して送信します。ttsession はネットワーク上で交信し、メッセージを配信します。

ttsession のコマンド行オプションを変更するには、dtstart_ttsession 変数を変更します。すべてのユーザの dtstart_ttsession 変数を変更するには、実行可能な sh または ksh スクリプト (たとえば、/etc/dt/config/Xsession.d/*myvars*) を作成し、dtstart_ttsession を次のように設定します。

```
dtstart_ttsession="/usr/dt/bin/ttsession -s"
```

同様に、 *HomeDirectory*/.dtprofile に dtstart_ttsession を設定することによって、ユーザのセッションの ttsession オプションを変更できます。

ttsession オプションの詳細は、ttsession(1) のマニュアル・ページを参照してください。ttsession の詳細は、『共通デスクトップ環境 *ToolTalk* メッセージの概要』を参照してください。

セッション・マネージャ・クライアントの起動

この時点で、Xsession は /usr/dt/bin/dtsession を起動し、セッション起動プロセスを続行します。

セッション・リソースの読み込み

セッション・マネージャは X サーバの RESOURCE_MANAGER 属性を使用して、デスクトップ・リソースをすべてのアプリケーションに対して使用可能にします。次の手順を実行することにより、セッション・マネージャは RESOURCE_MANAGER を読み込みます。

- システムのデフォルト・リソースを読み込む
- システム管理者によって指定されたシステム共通リソースをマージする
- ユーザ指定のリソースをマージする

デスクトップのデフォルト・リソースは /usr/dt/config/*language*/sys.resources にあります。これらのリソースは、RESOURCE_MANAGER 属性を介して各ユーザ・セッションに対して使用可能にされます。このファイルは、その後のデスクトップのインストール時に上書きされてしまうので、編集しないでください。

/etc/dt/config/*language*/sys.resources を作成することによって、システムのデフォルト・リソースを引き数にできます。このファイルでは、デフォルト・リソースを無効にしたり、すべてのデスクトップのユーザに対して追加のリソースを指定したりできます。このファイルは、セッションの起動中にデスクトップのデフォルト・リソースにマージされるため、新規または更新されたリソースの指定だけをこのファイルに格納してください。このファイルに指定されたリソースは、RESOURCE_MANAGER 属性を介して各ユーザのセッションに対して使用可能にされます。このファイルに指定されたリソースは、デスクトップのデフォルト・リソース・ファイルで指定されたものよりも優先されます。

HomeDirectory/.Xdefaults ファイルを使用して、デスクトップのデフォルト・リソースとシステム共通リソースを増やすことができます。このファイルに指定されたリソースは、RESOURCE_MANAGER 属性を介してユーザのセッションに対して使用可能にされます。このファイルに指定されたリソースは、デスクトップのデフォルト・リソース・ファイルまたはシステム管理者のリソース・ファイルで指定されたものよりも優先されます。

注 - X ツールキット・イントリンシクス・ユーティリティは、RESOURCE_MANAGER か *HomeDirectory*/.Xdefaults のどちらかからアプリケーションのリソースを読み込むように指定します。通常、これはユーザの *HomeDirectory*/.Xdefault ファイルが無視されることを意味します。しかし、セッション・マネージャは上述のように、セッションの起動時に *HomeDirectory*/.Xdefaults を RESOURCE_MANAGER にマージすることにより、*HomeDirectory*/.Xdefaults を格納します。

HomeDirectory/.Xdefaults を変更する場合、[リソースの再読込み] アクションを起動するまで新規アプリケーションはこの変更を表示できません。[リソースの再読込み] アクションは、デフォルト・リソース、システム共通リソース、およびユーザ指定のリソースで RESOURCE_MANAGER を再読み込みするようにセッション・マネージャに通知します。これにより、システム共通リソース・ファイルと個人用リソース・ファイルをアプリケーションが使用できるように変更されます。

詳細は、次の項目を参照してください。

- 293ページの「アプリケーション・リソースの設定」
- dtresourcesfile(4) のマニュアル・ページ

カラー・サーバの起動

セッション・マネージャは、デスクトップのカラー・サーバとして機能し、そのサーバをを構成するのに使用できる次のような dtsession リソースのセットを提供します。

foregroundColor — フォアグラウンド・カラーにピクセルを割り当てるかどうかを制御する。

dynamicColor — 読み込み専用カラーを割り当てるかどうか指定する。

shadowPixmaps — トップ・シャドウまたはボトム・シャドウにカラーを割り当てるかどうかを指定する。

colorUse — カラーの割り当てを制限する。

writeXrdbColors — *background リソースと *foreground リソースをリソース・データベースに格納するかどうか指定する。

/etc/dt/config/*language*/sys.resources を作成し、そのファイルの中にカラー・サーバを指定して、すべてのユーザに対してカラー・サーバのリソースを設定できます。

同様に、*HomeDirectory*/.Xdefaults にカラー・サーバのリソースを指定することによって、ユーザのセッションに対してのカラー・サーバのリソースを設定できます。

カラー・サーバのリソースの設定の詳細は、306ページの「カラーの管理」を参照してください。

ワークスペース・マネージャの起動

セッション・マネージャは、ワークスペース・マネージャを起動します。デフォルトでは、/usr/dt/bin/dtwm が起動されます。wmStartupCommand リソースを使用すると、代わりのウィンドウ・マネージャを指定できます。

/etc/dt/config/*language*/sys.resources を作成し、そのファイルにある Dtsession*wmStartupCommand リソースで絶対パス名とウィンドウ・マネージャのオプションを指定して、すべてのユーザの dtwm に代わるウィンドウ・マネージャを指定できます。

同様に、*HomeDirectory*/.Xdefaults に Dtsession*wmStartupCommand リソースを指定することによって、ユーザのセッションの代わりのウィンドウ・マネージャを指定できます。

ウィンドウ・マネージャの詳細は、第 16 章を参照してください。

セッション・アプリケーションの起動

セッションの起動時に、セッション・マネージャはセッションの一部として保存されたアプリケーションを再起動します。ユーザの初期セッションの一部として復元されるアプリケーションシステムのデフォルト・セット
は、/usr/dt/config/*language*/sys.session にあります。このファイルは、その後のデスクトップのインストール時に必ず上書きされますので、編集しないでください。

詳細は、dtsessionfile(4) のマニュアル・ページを参照してください。

システム管理者は /usr/dt/config/*language*/sys.session を /etc/dt/config/*language*/sys.session にコピーし、コピーしたファイルを変更することにより、ユーザの初期セッションの一部として起動されるアプリケーションのセットを置き換えることができます。リソース・ファイルとは違い、このファイルはデスクトップのデフォルト・ファイルを完全に置き換えたものとして使用されますので、システムのデフォルト・ファイルのコピーを作成し、必要に応じて変更できます。

追加セッション起動のカスタマイズ

この節では、次の作業について説明します。

- 環境変数の設定
- リソースの設定
- ディスプレイに依存するセッションの使用
- ログイン時のスクリプトの実行
- バックアップ・セッションの復元

▼ 環境変数を設定するには

- **システム共通環境変数を設定するには、変数を設定してエクスポートする /etc/dt/config/Xsession.d ディレクトリにファイルを作成します。**

たとえば、実行可能な sh または ksh スクリプトである /etc/dt/config/Xsession.d/*myvars* を作成すると、次の行が含まれています。

```
export MYVARIABLE="value"
```

変数 MYVARIABLE は、次のログイン時に各ユーザの環境に設定されます。

- **個人用環境変数を設定するには、*HomeDirectory*/.dtprofile に変数を設定します。**

たとえば次の行により、変数 MYVARIABLE は、次のログイン時に各ユーザの環境に設定されます。

```
export MYVARIABLE="value"
```

注 - セッション・マネージャは .profile または .login ファイルを自動的に読み込みません。しかし、これらのファイルを使用するために構成することはできます。詳細は、29ページの「オプションとしての .profile または .login スクリプトの参照」を参照してください。

▼ リソースを設定するには

- システム共通リソースを設定するには、/etc/dt/config/*language*/sys.resources ファイルにリソースを追加します **(**ファイルを作成する必要がある場合があります**)**。

注 - .dtprofile は、/bin/sh または /bin/ksh 構文だけをサポートします。

たとえば、/etc/dt/config/C/sys.resources に下記の行を指定すると、リソース AnApplication*resource は、次のログイン時に各ユーザの RESOURCE_MANAGER 属性に設定されます。

```
AnApplication*resource: value
```

- 個人用リソースを設定するには、*HomeDirectory*/.Xdefaults ファイルにリソースを追加します。

▼ ディスプレイに固有のリソースを設定するには

システム上のすべてのデスクトップ・ユーザに対してディスプレイに固有のリソースを設定できます。また、ユーザのセッションに制限されたディスプレイに固有のリソースを設定できます。この設定により、ユーザがデスクトップにログインするディスプレイに応じて、リソースを指定できるようになります。

- システム上のすべてのデスクトップ・ユーザのディスプレイに固有のリソースを設定するには、ディスプレイに固有のリソースを指定する /etc/dt/config/*language*/sys.resources ファイルを作成します。

- 個人用ディスプレイに固有のリソースを設定するには、*HomeDirectory*/.Xdefaults ファイルにリソースを指定します。

cpp 条件文でこれらのリソースを囲むことにより、リソースを区切ります。DISPLAY_displayname マクロが $DISPLAY 変数の値に応じて定義されます。これは、すべての . (ピリオド) と : (コロン) 文字を _ (下線文字) に変換し、画面の指定を取り除き、最後に DISPLAY_ という接頭辞をその結果に付けます。

たとえば、:0 の $DISPLAY は DISPLAY_0 になり、blanco.gato.com:0.0 の $DISPLAY は DISPLAY_blanco_gato_com_0 になります。結果の値は、セッションのリソース・ファイルの cpp テストの一部として使用できます。たとえば、/etc/dt/config/C/sys.resources では、次のように指定します。

```
Myapp*resource: value


#ifdef DISPLAY_blanco_gato_com_0
    Myapp*resource: specialvalue1
    #endif


#ifdef DISPLAY_pablo_gato_com_0
    Myapp*resource: specialvalue2
    #endif
```

この場合、リソース MyApp*resource は、ディスプレイ blanco.gato.com:0 にログインするときは specialvalue1 に対して、pablo.gato.com:0; にログインするときは specialvalue2 に対して、別のディスプレイにログインするときは value に対して、それぞれ RESOURCE_MANAGER に設定されます。

▼ 初期セッションのアプリケーションを変更するには

ユーザの初期セッションの一部として起動する代わりのアプリケーションを指定できます。

1. /usr/dt/config/***language***/sys.session を /etc/dt/config/***language***/sys.session にコピーします。

2. 新規の sys.session ファイルを変更します。

   sys.session にある各エントリは次のように表示されます。

   ```
   dtsmcmd -cmd command_and_options
   ```

   ユーザの初期セッションの一部として追加のアプリケーションを起動するには、絶対パス名で新しい sys.session エントリを指定します。たとえば、ユーザの初期セッションの一部として /usr/bin/X11/xclock を起動するには、xclock エントリを /etc/dt/config/C/sys.session に追加します。

```
#
# Start up xclock...
#
dtsmcmd -cmd "/usr/bin/X11/xclock -digital"
```

▼ ディスプレイに固有のセッションを設定するには

特定のディスプレイに合わせてセッションを調節するように、ディスプレイに、固有のセッションを設定できます。

- ***HomeDirectory***/.dt/sessions ディレクトリを ***HomeDirectory***/.dt/***display*** にコピーします。この場合 ***display*** は実際に存在する修飾していないホスト名です (pablo:0 は有効で、pablo.gato.com:0 や unix:0 は無効です)。

たとえば、ディスプレイ pablo.gato.com:0 のディスプレイに固有のセッションを作成するには、次のように指定します。

```
cp -r HomeDirectory/.dt/sessions HomeDirectory/.dt/pablo:0
```

ディスプレイ pablo.gato.com:0 に次にログインしたときには、セッション・マネージャはそのディスプレイに固有のセッションを起動します。

セッションの起動時とログアウト時の追加コマンドの実行

デスクトップ・セッションにログインしたときに、追加コマンドが起動されるように指定できます。これは、セッション・マネージャが保存しない X の設定を行う場合に有用です。たとえば、ユーザは xsetroot を使用してルート (ワークスペース)・ポインタをカスタマイズできます。もう 1 つの使用方法は、セッション・マネージャによって保存および復元できないアプリケーションを起動することです。セッションが復元されたときにアプリケーションが再起動しない場合、ユーザはこの方法を使用して、クライアントを起動できます。

▼ セッションの起動時に追加コマンドを実行するには

- コマンドが入っている ***HomeDirectory***/.dt/sessions/sessionetc ファイルを作成します。

通常、このファイルはスクリプトで、実行権を持っていなければなりません。sessionetc で起動されるプロセスはバックグラウンドで実行されなければなりません。

注 - セッション・マネージャによって自動的に復元されるクライアントを起動するために、sessionetc を使用しないでください。使用すると、複数のアプリケーションのコピーが起動されてしまいます。ウィンドウは、もう 1 つのウィンドウの上部に重なることがあるので、コピーをすぐに見つけられない可能性があります。

▼ ログアウト時に追加コマンドを実行するには

sessionetc に付属したファイルは sessionexit です。セッション・マネージャが処理しないセッション終了時のオペレーションのいくつかを実行するには、sessionexit を使用します。

- ***HomeDirectory***/.dt/sessions/sessionexit ファイルを作成します。

  sessionetc と同様に、このファイルは通常は実行権を持っているスクリプトです。

▼ バックアップからセッションを復元するには

セッション・マネージャがセッションを保存すると、ディスプレイに固有のセッションを使用している場合は、セッション情報が *HomeDirectory*/.dt/sessions ディレクトリまたは *HomeDirectory*/.dt/display ディレクトリに保存されます。現在のセッションまたはホーム・セッションそれぞれに関する情報を格納するために、セッション・マネージャは current または home という名前のサブディレクトリをこれらのディレクトリに作成します。セッション情報が格納される前に、セッション・マネージャはその名前で以前のセッションのバックアップを作成し、current.old か home.old に格納します。

1. ログイン画面から **[**復旧セッション**]** か **[**コマンド行ログイン**]** を使用してログインします。

2. バックアップのセッション・ディレクトリを有効な名前にコピーします。たとえば、バックアップのホーム・セッションを復元するには、次のようにします。

   ```
   cp -r HomeDirectory/.dt/sessions/home.old  HomeDirectory/.dt/sessions/home
   ```

ディスプレイに固有のセッションも同じ方法で復元できます。

▼ セッションの起動に関する問題を調べるには

- ***HomeDirectory***/.dt/startlog ファイルをチェックします。

  セッション・マネージャは、このファイルに各ユーザのセッション起動の経過状況を記録します。

セッション・マネージャのファイルとディレクトリ

- /usr/dt/bin/Xsession
- /usr/dt/config/Xsession.d/*
- /usr/dt/bin/dtsession
- /usr/dt/bin/dtsession_res
- *HomeDirectory*/.dt/sessions/current
- *HomeDirectory*/.dt/sessions/home
- *HomeDirectory*/.dt/*display*/current
- *HomeDirectory*/.dt/*display*/home

第 3 章

ログイン時とセッション起動時の問題解決の方法

この章では、Solaris CDE 起動ファイルと Solaris CDE 起動時に考えられる問題について説明し、起動時の問題に対する解決策を提案します。

ログイン起動ファイル

Solaris CDE ログイン・マネージャは、ユーザを認証するとき、次のスクリプトを呼び出してデスクトップを起動します。

```
/usr/dt/bin/Xsession
```

`Xsession` が最初に呼び出すユーザ固有のファイルは、*HomeDirectory*`/.dtprofile` です。

新しいユーザが Solaris CDE に初めてログインした際に、`.dtprofile` ファイルは、そのユーザのホーム・ディレクトリにコピーされます。デフォルトでは、このファイルは何も実行しません。しかし、このファイルには、このファイルの編集方法について、多くのコメントが入っています。ユーザは、このファイルを編集して、ユーザ固有の環境変数を追加できます。

注 - .dtprofile を呼び出す Xsession スクリプトのように、このファイルは、ksh 構文を使用します。

便利な編集方法の 1 つは、次に示す .dtprofile ファイルの最後の行のコメントを解除して有効にすることです。

```
DTSOURCEPROFILE=true
```

この行によって、ユーザの *HomeDirectory*/.login (csh ユーザの場合) または *HomeDirectory*/.profile (他のシェルのユーザの場合) が、起動プロセスの一部として有効になります。

エラー・ログの位置

Solaris CDE ログイン画面上で、[オプション] メニューから [復旧セッション] を選択すると、通常の X セッション起動をバイパスして、エラー・ログが表示されるので、ユーザのドット・ファイルで発生する可能性のある問題を修正できます。表 3–1 に、エラー・ログとその位置を示します。

表 3–1 エラー・ログの位置

位置	エラー・ログ
/var/dt/Xerrors	ユーザ・ログイン以前の Solaris CDE ログイン・ウィンドウ・システムのエラー
HomeDirectory/.dt/startlog	X セッション、.dtprofile、.login、.profile 中の Solaris CDE 起動エラー
HomeDirectory/.dt/errorlog	X セッション起動後の Solaris CDE エラー
HomeDirectory/.dt/sessionlogs	セッション・マネージャとウィンドウ・マネージャのエラー用のセッション・ログのディレクトリ

システム・コンソール・ウィンドウに表示されるエラーもあります。システム・コンソール・ウィンドウが動作していない場合、代替コンソール・ログ・ファイル名は、時間と表示コードが連結された wscon です。たとえば、次のようになります。

```
/usr/tmp/wsconAAAa004EE:0.0
```

ユーザ起動ファイル

.login (csh ユーザの場合) または .profile (sh か ksh ユーザの場合) を起動するには、次のように .dtprofile ファイルの最後の行のコメントを解除し有効にします。

```
DTSOURCEPROFILE=true
```

ほとんどの場合、この作業だけで十分ですが、.login ファイルまたは .profile ファイルを変更しなければならない場合もあります。それらのファイルには、Solaris CDE ログイン・マネージャで動作しないコマンドが入っている可能性があるためです。コマンドの 1 つに問題がある場合、その問題は通常、stty や tset「入力待ち」コマンドなどを使って、端末キーボードからの入力を受け付けるファイルに関連します。

注意 - .dtprofile で起動するように設定した .login ファイルまたは .profile ファイルに、シェルをクラッシュさせるようなコマンドが入っている場合は、デスクトップの起動は失敗します。その結果、デスクトップは表示されません。その代わりに、Solaris CDE ログイン画面が再び表示されます。.login または .profile が原因の起動エラーは、通常、*HomeDirectory*/.dt/startlog に記録されます。.login または .profile でコマンドをデバッグするには、復旧ログイン・セッションかコマンド行ログインを使用してください。

問題とその解決策の詳細は、.dtprofile ファイルを参照してください。通常、問題のコマンドは端末の情報と制御に関係があります。

Solaris CDE 起動例

この節では、次のユーザ起動ファイルの編集例を示します。

- .login (csh ユーザの場合)
- .profile (sh または ksh ユーザの場合)
- .Xdefaults

Solaris CDE 起動プロセスは、Solaris CDE 起動プロセス中にチェックできるように、.login スクリプトまたは .profile スクリプトで、シェル変数 DT を定義します。これによってtty、や stty などの端末に関連するコマンドが実行されなくなります。次のスクリプトの例を参照してください。

.login (C シェル)

```
if ( ! ${?DT} ) then

 stty erase `^h`

endif
```

.profile (sh または ksh)

```
if [ ! ``$DT'' ]; then

 stty erase `^h`

fi
```

ユーザが従来のテキスト・ベースのコンソール・ログイン・プロンプトからログインしたときには DT は定義されませんが、このような場合でもシェルから警告が出ることはありません。ドット・ファイルの設定の詳細は、*HomeDirectory*/.dtprofile ファイルを参照してください。

注 - Solaris CDE の tty 設定は .Xdefaults に指定します。

ユーザの .Xdefaults ファイルは、Solaris CDE 起動中に、ユーザ固有のリソースの設定のために起動されます。たとえば、ttyModes は、dtterm や xterm などの端末エミュレータ・ウィンドウにおいて、ユーザの好みの tty 設定を指定します。次の行は、.Xdefaults ファイルの典型的な ttyModes 設定です。

```
*ttyModes: erase ^H intr ^C kill ^U start ^Q stop ^S susp ^Z`
```

注 - Solaris CDE リソースは、デフォルトの設定と異なる場合があります。

デフォルトの設定例として、デスクトップ・ウィンドウ・マネージャのアイコンの配置を考えます。この場合、.Xdefaults ファイルの次の行は、デフォルトのアイコン配置の設定を示します。

```
Dtwm*iconPlacement: right top
```

第 4 章

アプリケーションの追加と管理

アプリケーション・マネージャは、ユーザが使用できるアプリケーションのデスクトップ・コンテナです。

この章では、次の内容について説明します。

アプリケーション・マネージャの構造

通常、アプリケーション・マネージャのトップレベルにはディレクトリがあります。そのような各ディレクトリとその内容を「アプリケーション・グループ」と言います。

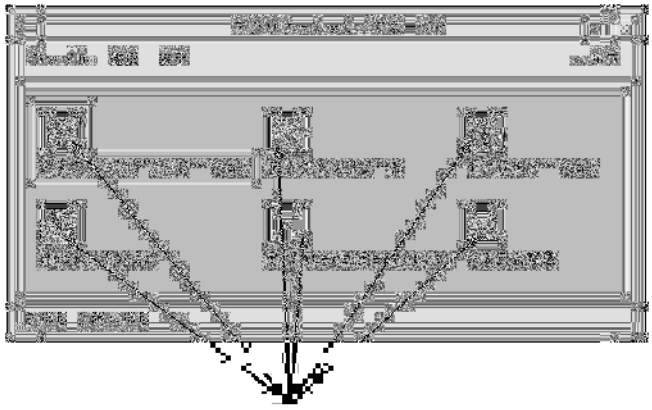


図 4–1　アプリケーション・マネージャのアプリケーション・グループ

アプリケーション・グループとその内容は、ローカルおよびネットワーク全体の複数の場所から収集されます。

アプリケーション・マネージャのディレクトリの位置

ファイル・システムにおいて、アプリケーション・マネージャはディレクトリ `/var/dt/appconfig/appmanager/`*login-hostname-display* です。ディレクトリは、ユーザがログインするたびに動的に作成されます。

たとえば、ユーザ `ronv` がディスプレイ `wxyz:0` からログインする場合、アプリケーション・マネージャのディレクトリ `/var/dt/appconfig/appmanager/ronv-wxyz-0` が作成されます。

アプリケーション・マネージャのアプリケーションの検索および収集方法

アプリケーション・マネージャは、ローカルなアプリケーション・グループとリモートのアプリケーション・グループを集めて構築されます。アプリケーション・グループは、アプリケーション検索パス上に位置するディレクトリから収集されます。

デフォルトのアプリケーション検索パスは、表 4–1 のようになります。

表 4–1 デフォルトのアプリケーション検索パスの位置

範囲	位置
組み込み	/usr/dt/appconfig/appmanager/*language*
システム共通	/etc/dt/appconfig/appmanager/*language*
個人用	*HomeDirectory*/.dt/appmanager

アプリケーション・マネージャのトップレベルを作成するために、ログイン時にアプリケーション検索パス上のディレクトリにあるアプリケーション・グループ (ディレクトリ) から、アプリケーション・マネージャのディレクトリ /var/dt/appconfig/appmanager/*login-hostname-display* へのリンクが作成されます。収集オペレーションは、デスクトップ・ユーティリティ dtappgather によって行われます。dtappgather は、ユーザがログインに成功した後に、ログイン・マネージャによって自動的に実行されます。

たとえば、デスクトップは次の組み込みアプリケーション・グループを提供します。

/usr/dt/appconfig/appmanager/*language*/Desktop_Tools

ログイン時に、次のディレクトリへのシンボリック・リンクが作成されます。

/var/dt/appconfig/appmanager/*login-hostname-display*/Desktop_Tools

アプリケーション検索パスには、リモートのディレクトリも指定できます。このため、ネットワーク全体に位置するシステムからアプリケーション・グループを収集できます。詳細は、55ページの「アプリケーション検索パスへのアプリケーション・サーバの追加」を参照してください。

アプリケーション収集の優先規則

検索パス上で重複したアプリケーションが存在する場合、個人用アプリケーション・グループはシステム共通グループに優先し、システム共通グループは組み込みグループに優先します。たとえば、/usr/dt/appconfig/appmanager/C/Desktop_Tools と /etc/dt/appconfig/appmanager/C/Desktop_Tools が存在する場合は、/etc ディレクトリにあるアプリケーション・グループが使用されます。

デフォルト・デスクトップとともに提供されるアプリケーション・グループ

カスタマイズされていないデスクトップは、次のアプリケーション・グループを提供します。

- デスクトップアプリケーション
- デスクトップツール
- デスクトップ・コントロール
- インフォメーション
- システム管理

アプリケーション・グループ収集方法の例

図 4–2 に、さまざまなアプリケーション・グループを含むアプリケーション・マネージャのウィンドウを示します。表 4–2 に、アプリケーション・グループが収集されるディレクトリを示します。

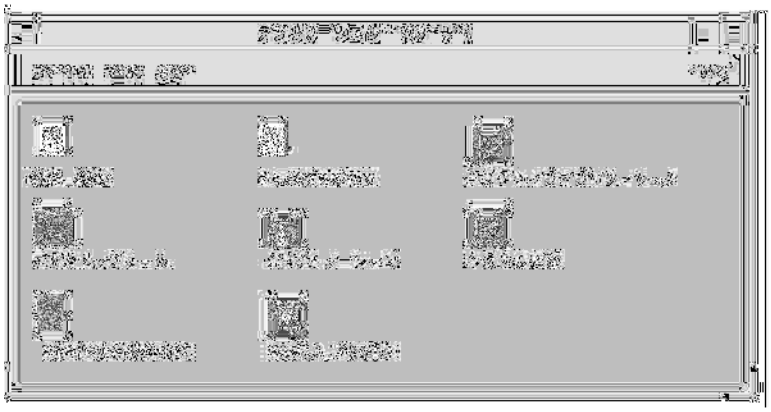


図 4–2　典型的なアプリケーション・マネージャのウィンドウ

表 4–2　図 4–2 のアプリケーション・グループのソース

名前	収集されるディレクトリ
CAD_App	`/net/ApServA/etc/dt/appconfig/appmanager/C/CAD_App`
DrawingApp	`/etc/dt/appconfig/appmanager/C/DrawingApp`

表 4–2　図 4–2 のアプリケーション・グループのソース　続く

名前	収集されるディレクトリ
デスクトップアプリケーション	/usr/dt/appconfig/appmanager/C/Desktop_Apps
デスクトップツール	/usr/dt/appconfig/appmanager/C/Desktop_Tools
インフォメーション	/usr/dt/appconfig/appmanager/C/Information
OpenWindows	/usr/dt/appconfig/appmanager/C/OpenWindows
システム管理	/etc/dt/appconfig/appmanager/C/System_Admin
MySpreadSheet	/users/anna/.dt/appmanager/MySpreadSheet
Media_Tools	/etc/dt/appconfig/appmanager/C/Media_Tools

[インフォメーション] アプリケーション・グループまたは [システム管理] アプリケーション・グループがカスタマイズされている場合、代わりに /etc/dt/appconfig/appmanager/C から収集されます。

ApServA という名前のシステムがアプリケーション検索パスに追加されたので、CAD_App グループが収集されます (詳細は、55ページの「アプリケーション検索パスへのアプリケーション・サーバの追加」を参照してください)。MySpreadSheet [自分用スプレッドシート] は、ユーザ anna だけが使用できる個人アプリケーション・グループです。

アプリケーション・マネージャへのアプリケーションの追加

アプリケーションがアプリケーション・マネージャに追加された場合、そのアプリケーションを起動するアプリケーション・グループの中にアイコンができます。

多くのアプリケーションはアプリケーション・グループを提供します。アプリケーション・グループは、アプリケーション・アイコンとアプリケーションに関連する

その他のファイルを含むアプリケーション・マネージャの、トップレベルのディレクトリです。

一部のアプリケーションには、独自のアプリケーション・グループがない可能性があります。その代わりに、アプリケーションを起動するアイコンが一般アプリケーション・グループにあります。たとえば、システム上にインストールしたすべてのゲームのコンテナとして使用する「Games」という名前の空のアプリケーション・グループを作成できます。

アプリケーションをアプリケーション・マネージャに追加する方法

アプリケーションをアプリケーション・マネージャに追加するには、次の 2 つの方法があります。

- アプリケーションを登録する
- アプリケーションを登録せずにアプリケーション・アイコンを追加する

アプリケーションの登録

アプリケーション登録により、アプリケーションの完全な統合が提供されます。

登録済みアプリケーションの特徴は次のとおりです。

- 独自のアプリケーション・グループがあります。
- 1 つの位置から収集されたデスクトップ構成ファイルがあります。このデスクトップ構成ファイルのグループを「登録パッケージ」と言います。
- 登録済みヘルプ・ボリュームがある場合があります。

アプリケーションを登録するには、次の 2 つの方法があります。

- デスクトップ化アプリケーションをインストールすると、自動的に登録されます。詳細は、51ページの「デスクトップ化アプリケーションをアプリケーション・マネージャに追加するには」を参照してください。
- 既存のアプリケーションは、登録パッケージを作成することによって登録できます。詳細は、51ページの「既存または非デスクトップ化アプリケーションを登録するには」を参照してください。

登録パッケージを使用すると、デスクトップでのアプリケーションの管理が簡単になります。登録パッケージは、ファイル・システムの、デスクトップ構成ファイルに使用された位置以外のどこかで作成されます。

登録パッケージを使用しないアプリケーションの追加

これは、アプリケーションを起動するためのアイコンだけをアプリケーション・マネージャに入れる場合に、アプリケーションを追加するのに望ましい方法です。

登録パッケージを使用せずに追加したアプリケーションの特徴は次のとおりです。

- 独自のアプリケーション・グループがある場合もありますが、通常はアイコンを既存のアプリケーション・グループに置きます。
- デスクトップ構成ファイルを、直接デスクトップの検索パス上の位置に置きます。

詳細は、52ページの「アプリケーション・アイコンを既存のアプリケーション・グループに追加するには」を参照してください。

▼ デスクトップ化アプリケーションをアプリケーション・マネージャに追加するには

デスクトップ化アプリケーションは、アプリケーションのインストール時に自動的にアプリケーション・マネージャに登録されるアプリケーションです。このアプリケーションのファイルセットには、デスクトップに必要な登録パッケージが入っています。

1. アプリケーションの指示に従って、アプリケーションをインストールします。
2. インストールが完了したら、**[デスクトップツール]** アプリケーション・グループの **[アプリケーションの再読込み]** をダブルクリックします。
3. インストールが完了したか次のように確認します。
   a. アプリケーション・マネージャを開き、新しいアプリケーション・グループがあるかチェックします。
   b. アプリケーションを開くには、アプリケーション・グループを開き、そのアプリケーションのアイコンをダブルクリックします。

▼ 既存または非デスクトップ化アプリケーションを登録するには

これは、アプリケーションをデスクトップに完全に統合するのに望ましい方法です。

デスクトップは、登録パッケージ・ファイルとデスクトップ検索パス上のディレクトリとの間にリンクを作成する dtappintegrate というツールを提供します。

デスクトップ登録については、第 5 章で説明します。

▼ アプリケーション・アイコンを既存のアプリケーション・グループに追加するには

この手順では、アプリケーション・アイコンを既存のアプリケーション・グループに追加する方法を説明します。

たとえば、デスクトップは [システム管理] という名前のアプリケーション・グループを提供しています。[システム管理] は、システム管理に関係するさまざまなアプリケーションとスクリプトのために確保されています。頻繁に実行するスクリプトがある場合は、[システム管理] アプリケーション・グループのアイコンをダブルクリックすることによってスクリプトを実行できるようにしてください。

1. アプリケーションにアクション定義を作成するために、アクション作成ツールを使用します。

   アクション作成ツールの詳細は、第 11 章を参照してください。

2. 実行可能ファイルを、アクション名と同じ名前でアプリケーション・グループのディレクトリに作成します。ファイルの内容は関係ありません。

   たとえば、システム管理ツールを実行する「Cleanup」というアクションを作成した場合、実行可能ファイル /etc/dt/appconfig/appmanager/*language*/System_Admin/Cleanup を作成します。

一般アプリケーション・グループの作成と管理

一般アプリケーションは、1 つの特定のアプリケーション・プロダクトに関連付けられていないアプリケーション・グループ (ディレクトリ) です。たとえば、組み込みの [デスクトップツール] アプリケーション・グループは、関連しているが 1 つのプロダクトの一部ではない多数のアプリケーション用アイコンを含む、一般グループです。

追加の一般アプリケーション・グループを作成することもできます。たとえば、システム上で使用可能なさまざまなゲームをグループ化するための Games というグループを作成できます。

一般アプリケーション・グループの範囲は、システム共通または個人用です。

▼ システム共通の一般アプリケーション・グループを作成するには

1. **root** でログインします。
2. `/etc/dt/appconfig/appmanager/`***language*** にディレクトリを作成します。
   このディレクトリ名がアプリケーション・グループ名になります。
3. **[**デスクトップツール**]** アプリケーション・グループの **[**アプリケーションの再読込み**]** をダブルクリックします。

▼ 個人用一般アプリケーション・グループを作成するには

1. ***HomeDirectory***`/.dt/appmanager` にディレクトリを作成します。
   このディレクトリ名がアプリケーション・グループ名になります。
2. **[**デスクトップツール**]** アプリケーション・グループの **[**アプリケーションの再読込み**]** をダブルクリックします。

▼ 組み込みアプリケーション・グループをカスタマイズするには

1. **root** でログインします。
2. アプリケーション・グループが `/usr/dt/appconfig/appmanager/`***language*** にある場合は、アプリケーション・グループを `/etc/dt/appconfig/appmanager/`***language*** にコピーします。
   たとえば、次のコマンドは [デスクトップツール] アプリケーション・グループをコピーします。

```
cp -r /usr/dt/appconfig/appmanager/C/Desktop_Tools \
   /etc/dt/appconfig/appmanager/C
```

アプリケーション・グループの新しいコピーは、組み込みアプリケーション・グループより優先されます。

3. アプリケーション・グループのコピーを変更します。たとえば、新しいアクション・ファイル (アクションと同じ名前の実行可能ファイル) を追加できます。

4. 変更を見るには、ログアウトしてからログインし直します。

アプリケーションの検索に使用される検索パスの変更

アプリケーション検索パスを変更する主な理由は、アプリケーション・サーバの追加です。アプリケーション・サーバを検索パスに追加すると、アプリケーション・マネージャはすべてのサーバのシステム共通のアプリケーション・グループを収集します。

アプリケーション検索パスの詳細は、147ページの「アプリケーション検索パス」を参照してください。

デフォルト検索パス

デフォルトのアプリケーション検索パスには、表 4–3 のディレクトリがあります。

表 4–3 デフォルトのアプリケーション検索パスのディレクトリ

範囲	検索パスディレクトリ
個人用	*HomeDirectory*/.dt/appmanager
システム共通	/etc/dt/appconfig/appmanager/*language*
組み込み	/usr/dt/appconfig/appmanager/*language*

表 4–3　デフォルトのアプリケーション検索パスのディレクトリ　続く

アプリケーション検索パスへのアプリケーション・サーバの追加

アプリケーション検索パスを変更するほかに、アプリケーション・サーバと通信できるようにするために、追加の構成タスクを実行する必要がある場合があります。詳細は、128ページの「アプリケーション・サービスの管理」を参照してください。

システム共通アプリケーション検索パスを設定するには

1. **root** でログインします。
2. /etc/dt/config/Xsession.d/0010.dtpaths ファイルが存在しない場合は、/usr/dt/config/Xsession.d/0010.dtpaths をコピーして作成します。
3. /etc/dt/Xsession.d/0010.paths を開いて編集します。DTSPSYSAPPHOSTS 変数を設定して行を追加または編集します。

   ```
   export DTSPSYSAPPHOSTS=hostname:[,hostname]
   ```

   たとえば、次の行はシステム ApServA をアプリケーション検索パスに追加します。

   ```
   export DTSPSYSAPPHOSTS=ApServA:
   ```

4. システム上のすべてのユーザに、変更を有効にするためにはログアウトしてからもう一度ログインするよう通知します。

個人アプリケーション検索パスを設定するには

1. *HomeDirectory*/.dtprofile を開いて編集します。
2. DTSPUSERAPPHOSTS 変数を設定して行を追加または編集します。

   ```
   export DTSPUSERAPPHOSTS=hostname:[,hostname]
   ```

たとえば、次の行はシステム ApServB および ApServC をアプリケーション検索パスに追加します。

```
export DTSPUSERAPPHOSTS=ApServB:,ApServC:
```

3. ログアウトしてからログインし直します。

一般アプリケーション・マネージャ管理

一般アプリケーション・マネージャの管理タスクは次の 2 つです。

- アプリケーションを削除する
- セッション中にアプリケーションのデータベースを再読み込みする

▼ アプリケーションを削除するには

dtappintegrate ツールを使用してアプリケーションを登録した場合は、同じ dtappintegrate ツールを使用して逆のプロセスを実行できます。アプリケーションの登録を解除すると、アプリケーション・グループはアプリケーション・マネージャから削除され、アクション、データ型、アイコン、ヘルプを使用できなくなります。

1. **root** でログインします。
2. 次のコマンドを実行します。

```
dtappintegrate -s app_root -u
```

▼ セッション中にアプリケーション・マネージャを更新するには

アプリケーションを追加した場合、変更を直ちに有効にするには、アプリケーション・マネージャを再構築しなければなりません。

- **[デスクトップツール] アプリケーション・グループを開き、[アプリケーションの再読込み] をダブルクリックします。**

[アプリケーションの再読込み] は、アプリケーションがアプリケーション・サーバに追加されたときにアプリケーション・マネージャを更新するのに便利です。しかし、[アプリケーションの再読込み] は、アプリケーション・サーバから削除されたアプリケーションや、別の場所に移動したアプリケーションを検出しません。

テキスト・エディタと端末エミュレータの変更

テキスト・エディタと端末エミュレータの両方のアプリケーションは、フロントパネルでコントロールを選択するか、アプリケーション・マネージャでアイコンをダブルクリックすると起動できます。

このようなアプリケーションは、その他のデスクトップ・アクティビティでも起動できます。

- ファイル・マネージャでテキスト・ファイルを選択し、[選択] メニューから [開く] を選択すると、テキスト・エディタが開きます。デフォルトのテキスト・エディタは `dtpad` です。
- ファイル・マネージャの [ファイル] メニューから [端末エミュレータを開く] を選択するか、アクションによって端末エミュレータ・ウィンドウを開くと、端末エミュレータが実行されます。デフォルトの端末エミュレータは `dtterm` です。

異なるテキスト・エディタや端末エミュレータのアプリケーションを使用するために、デスクトップを構成できます。

▼ デフォルトのテキスト・エディタまたは端末エミュレータを変更するには

1. システム共通に変更する場合は、**root** でログインします。
2. 次のいずれかの方法で、新しいテキスト・エディタまたは端末エミュレータのアプリケーションのためのアクションを作成します。

- アクション作成ツールを使用します。図 4–3 は、TextPad というアプリケーションのための [アクション作成] ウィンドウです。アクション作成ツールの詳細は、第 11 章を参照してください。

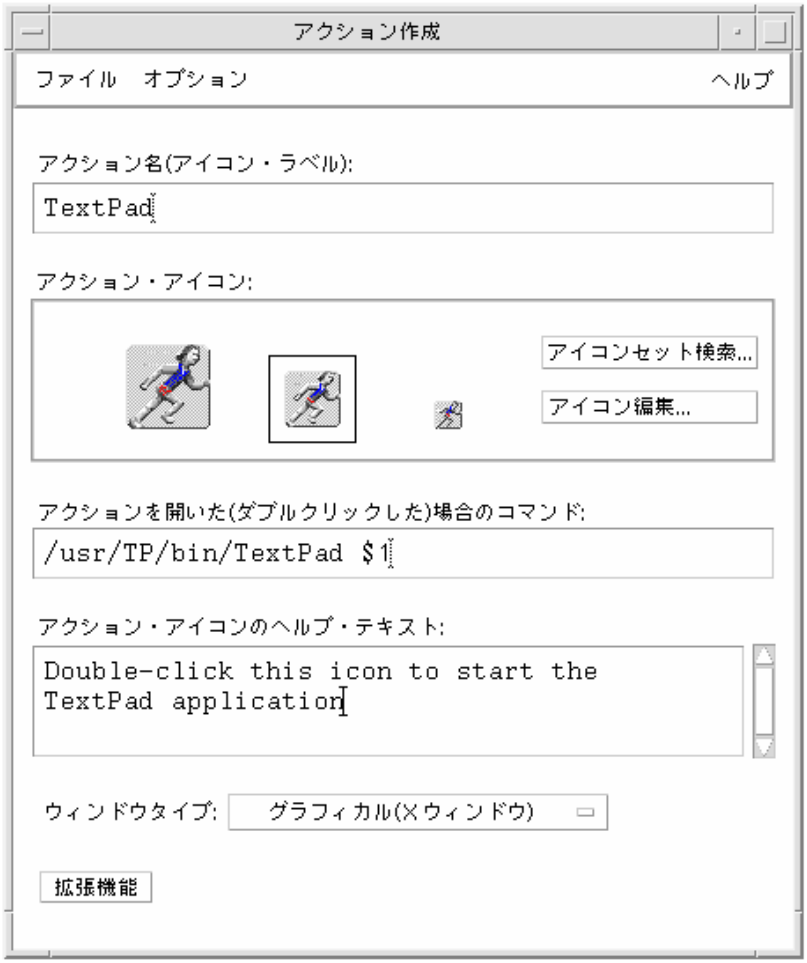


図 4–3　[アクション作成] ウィンドウ

- 次の例のように手入力でもアクション定義を作成できます。

```
ACTION TextPad
   {
      LABEL       TextPad
      TYPE     COMMAND
      WINDOW_TYPE  NO_STDIO
      EXEC_STRING  /usr/TP/bin/TextPad %(File)Arg_1%
      DESCRIPTION  Double-click this icon to start the TextPad application.
   }
```

手入力でのアクション定義の作成については、第 12 章を参照してください。

3. 新しいアクションを格納している構成ファイルを、適切なディレクトリに置きます。
   - システム共通: /etc/dt/appconfig/types/*language*
   - 個人用: *HomeDirectory*/.dt/types

4. 適切な user-prefs.dt ファイルが存在しない場合は、/usr/dt/appconfig/types/***language***/user-prefs.dt を次のディレクトリにコピーして作成します。
   - システム共通: /etc/dt/appconfig/types/*language* ディレクトリ
   - 個人用: *HomeDirectory*/.dt/types ディレクトリ

5. テキスト・エディタまたは端末のアクションを、システム共通または個人用の user-prefs.dt ファイルで編集します。アクションを新しいアクションに対応付けるために、MAP_ACTION 行を変更します。

   たとえば、次の行を

   ```
   MAP_ACTION Dtpad
   ```

   次のように変更します。

   ```
   MAP_ACTION TxtPd
   ```

6. user-prefs.dt ファイルを保存します。

7. アクション・データベースを再読み込みするために、**[**デスクトップツール**]** アプリケーション・グループで **[**アプリケーションの再読込み**]** をダブルクリックします。

第 5 章

アプリケーションの登録

この章では、アプリケーションの登録パッケージの作成方法と、デスクトップへのアプリケーションの登録方法について説明します。

アプリケーションを完全にデスクトップへ登録するには、次の内容を備えていることが必要です。

- アプリケーション・マネージャのトップレベルにある、独自のアプリケーション・グループ
- アプリケーションを起動するアクション。アプリケーション・グループではアクションはアイコンで表示されます。
- (省略可能) データ・ファイルのデータ型

アプリケーションの登録は、次のように設定するとアプリケーションのオペレーションを妨げません。

- アプリケーションの実行可能ファイル自身の変更を含まないこと。したがって、システムにすでに存在するアプリケーションも登録できます。
- アプリケーションが任意に提供するファイル (実行可能ファイルおよび `app-defaults` など) を他のファイルの位置へ移動させる必要がないこと
- 簡単に元に戻せること。`dtappintegrate` ツールはアプリケーションの登録に使用しますが、プロセスを逆にたどれるようなコマンド行オプションも提供します。

次のユーザは、登録パッケージの作成が必要です。

- 既存アプリケーションをデスクトップへ登録するシステム管理者
- デスクトップ化アプリケーションのインストール・パッケージを作成するソフトウェア・プログラマ

アプリケーション登録の概要

この節では、次の項目を説明します。

- アプリケーション登録の目的
- アプリケーション登録によって提供されるアプリケーションの機能

注 - 既存アプリケーションの登録方法の詳細は、85ページの「登録パッケージの作成例」を参照してください。

アプリケーション登録によって備わる機能

アプリケーションを登録すると、次のような作業をグラフィカルに実行できます。

- アプリケーションの配置

  インストール時、アプリケーションはアプリケーション・マネージャに「登録」され、独自のアプリケーション・グループを持ちます。

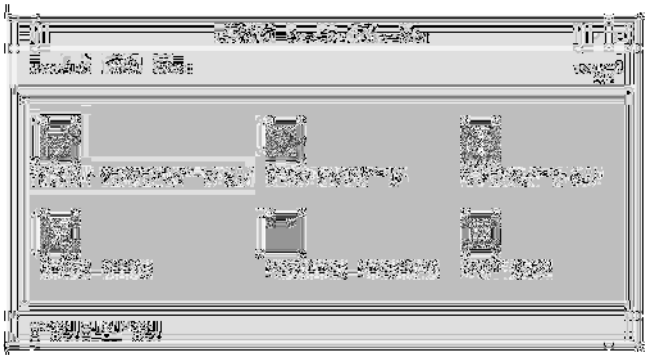


図 5–1　アプリケーション・マネージャのトップレベルにあるアプリケーション・グループ

- アプリケーションの起動

  アプリケーションのアプリケーション・グループには、ダブルクリックするとアプリケーションを起動できるアイコンがあります。

図 5–2　アプリケーションを起動するためのアイコンを含むアプリケーション・グループ

- データ・ファイルの識別および処理

  アプリケーションのデータ・ファイルにはファイル・マネージャで一意のアイコンがあります。

  ユーザはデータ・ファイル・アイコンを次の作業に使用できます。

  - アプリケーションの起動 (開く)
  - データ・ファイルの印刷

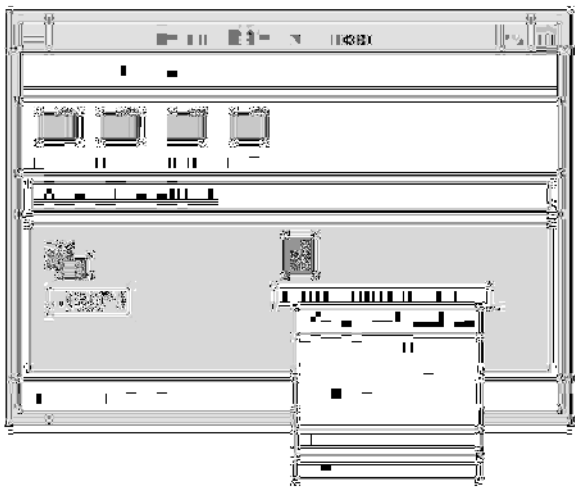


図 5–3　[開く] と [印刷] が表示されているデータ・ファイルのポップアップ・メニュー

- (オーディオ) データのメール送信、圧縮、表示、再生など、他のオペレーションの実行

アプリケーション登録の目的

登録されたデスクトップ・アプリケーションには、デスクトップがアプリケーションのユーザ・インタフェースを提供するのに使用する次のような構成ファイルがあります。

- アクションおよびデータ型定義ファイル
- アイコン・イメージ (ピックスマップまたはビットマップ)・ファイル
- アプリケーション・グループを作成するディレクトリおよびファイル
- (省略可能) デスクトップ・ヘルプ・ファイルおよびフロントパネル定義ファイル

これらのファイルをデスクトップが認識および使用するには、デスクトップの検索パスで指定したディレクトリに入っていなければなりません。

アプリケーションの構成ファイルが無数のディレクトリに分散されている場合、アプリケーションを管理するのは難しくなります。したがってデスクトップは、アプリケーションが 1 つのディレクトリの下にすべてのデスクトップ構成ファイルを集められるようにします。このグループ化されたファイルを「登録パッケージ」と呼びます。

アプリケーションがデスクトップ化されている場合、登録パッケージはインストール・パッケージの一部として提供されます。自分で構成ファイルを作成するシステム管理者は、自分で登録パッケージを作成できます。

登録パッケージの構成ファイルは正しい検索パス・ディレクトリにないため、デスクトップで使用できません。これらのファイルを正しい場所に置くプロセスを、アプリケーションの「登録」または「統合」と呼びます。

デスクトップは、正しい検索パス・ディレクトリにあるファイルのシンボリック・リンク表示を作成することにより、登録を行うツール `dtappintegrate` を提供します。

多くのデスクトップ化アプリケーションは、インストール・プロセス中に自動的に `dtappintegrate` を実行します。既存アプリケーションを統合するシステム管理者は、登録パッケージを作成した後で自分で実行できます。

アプリケーションをシステムのデスクトップに一度登録すると、システムの全ユーザがアプリケーションを使用できるようになります。システムがデスクトップ・アプリケーション・サーバとして設定されている場合、ネットワークの他のシステムもアプリケーションを使用できます。

`dtappintegrate` ツールには、リンクを切ることによってプロセスを逆に行うコマンド行オプションがあります。このオプションを使うと、アプリケーション・マネージャからアプリケーションを削除するのが容易になるため、アプリケーションを別のアプリケーション・サーバに移動または更新できます。

アプリケーション登録の一般的な手順

注 - これらの手順によってアプリケーション・パッケージを作成方法の詳細な例については、85ページの「登録パッケージの作成例」を参照してください。

1. **フォントとカラーを設定するアプリケーションのリソースを変更します。変更しないと、デスクトップのダイナミック・フォントおよびダイナミック・カラーが正しく動作しません。**

   詳細は、66ページの「手順 1: フォント・リソースとカラー・リソースの変更」を参照してください。

2. **アプリケーションの root 位置を作成します。**

詳細は、68ページの「手順 2: デスクトップ・アプリケーション root の作成」を参照してください。

3. アプリケーションの **root** の下にディレクトリ構造を作成します。

   詳細は、68ページの「手順 3: 登録パッケージ・ディレクトリの作成」を参照してください。

4. アプリケーションのアクションとデータ型を作成します。

   詳細は、70ページの「手順 4: アプリケーションのアクションとデータ型の作成」を参照してください。

5. 適切なディレクトリにヘルプ・ファイルを入れます。

   詳細は、74ページの「手順 5: 登録パッケージへのヘルプ・ファイルの組み込み」を参照してください。

6. アプリケーションのアイコンを作成します。

   詳細は、75ページの「手順 6: アプリケーション用アイコンの作成 」を参照してください。

7. アプリケーションのアプリケーション・グループを作成します。

   詳細は、76ページの「手順 7: アプリケーション・グループの作成 」を参照してください。

8. `dtappintegrate` を使用してアプリケーションを登録します。

   詳細は、82ページの「手順 8: `dtappintegrate` を使用したアプリケーションの登録」を参照してください。

手順 1: フォント・リソースとカラー・リソースの変更

注 - アプリケーションのリソースの変更例については、85ページの「登録パッケージの作成例」の手順 1 を参照してください。

デスクトップは、インタフェース・フォントとウィンドウ・カラーを設定および処理するための機能を提供します。アプリケーションでこれらの機能を正しく使用するには、アプリケーションの `app-defaults` ファイルを変更してください。

フォント・リソースの変更

注 - 本節の内容は、OSF/Motif 1.2 (またはそれ以降のバージョン) を使用して作成したアプリケーションに適用されます。スタイル・マネージャは、OSF/Motif の初期のバージョンを使用して書いたアプリケーションのインタフェース・フォントは設定できません。

デスクトップ・スタイル・マネージャは、アプリケーションがアプリケーション固有のインタフェース・フォントを作成しない場合、OSF/Motif 1.2 (またはそれ以降のバージョン) を使用して作成したアプリケーションのインタフェース・フォントを設定します。

スタイル・マネージャは、次の 2 つのフォントを提供します。

システム・フォント — ラベル、メニュー、ボタンなどのシステム領域で使用します。

ユーザ・フォント — テキスト・フィールドなど編集可能領域で使用します。

それぞれのフォントは、[フォント] ダイアログ・ボックスの 1 から 7 までの 7 種類のサイズで指定します。スタイル・マネージャ・フォントは、`/usr/dt/app-defaults/`*language*`/Dtstyle` に設定されたスタイル・マネージャ・リソースによって、システムの実際のフォントに接続されます。

アプリケーションにスタイル・マネージャ・フォントを使用する場合は、インタフェース・フォントを使用しているすべてのアプリケーションのリソースを削除してください。デスクトップは自動的にアプリケーションのリソースを適切に設定します。

`FontList` — システム・フォントに設定します。

`XmText*FontList` — ユーザ・フォントに設定します。

`XmTextField*FontList` — ユーザ・フォントに設定します。

カラー・リソースの変更

スタイル・マネージャは、アプリケーション・カラーをダイナミックに変更する機能を提供します。アプリケーションは OSF/Motif 1.1 または 1.2 クライアントでなければなりません。他のツールキットで書かれたクライアントは、カラーをダイナミックに変更できません。カラーの変更は、クライアントの再起動時に有効になります。

デスクトップが提供するダイナミック・カラーを最も簡単に使用する方法は、バックグラウンド・カラーおよびフォアグラウンド・カラーのアプリケーションのカラー・リソースを削除することです。

手順 2: デスクトップ・アプリケーション root の作成

注 - アプリケーションのデスクトップ・アプリケーションの root ディレクトリの作成例については、85ページの「登録パッケージの作成例」の手順 2 を参照してください。

アプリケーションの登録パッケージ・ファイルは、アプリケーションの root または *app_root* と呼ばれるディレクトリの下でグループ化されます。デスクトップの構成ファイル用に使用される *app_root* ディレクトリは、アプリケーションのインストール *app_root* と同じディレクトリか、他の場所でもかまいません。

たとえば、アプリケーションが `/usr/BTE` ディレクトリの下にあるとします。これと同じディレクトリを、デスクトップの構成ファイルの *app_root* として使用できます。ただし、既存のデスクトップ化されていないアプリケーションを統合する場合は、異なるデスクトップ *app_root* ディレクトリを作成してください。そうすれば、アプリケーションの更新時に、作成した構成ファイルは上書きされません。

たとえば、システム管理者がデスクトップ *app_root* ディレクトリとして、ディレクトリ `/etc/desktop_approots/BTE` を作成することもできます。

手順 3: 登録パッケージ・ディレクトリの作成

注 - アプリケーションの登録パッケージ・ディレクトリの作成例については、85ページの「登録パッケージの作成例」の手順 3 を参照してください。

登録パッケージは、デスクトップアプリケーションにグラフィカル・インタフェースを提供するために使用するデスクトップ構成ファイルのグループです。

登録パッケージの内容

デスクトップ構成ファイルには、次のものがあります。

- アクションおよびデータ型定義ファイル
- アイコン・イメージ・ファイル

- アプリケーション・グループ・ディレクトリとその内容
- (省略可能) ヘルプ・データ・ファイルおよびフロントパネル構成ファイル

登録パッケージは、アプリケーションの root または *app_root* と呼ばれるトップレベルのディレクトリの下に集められます。

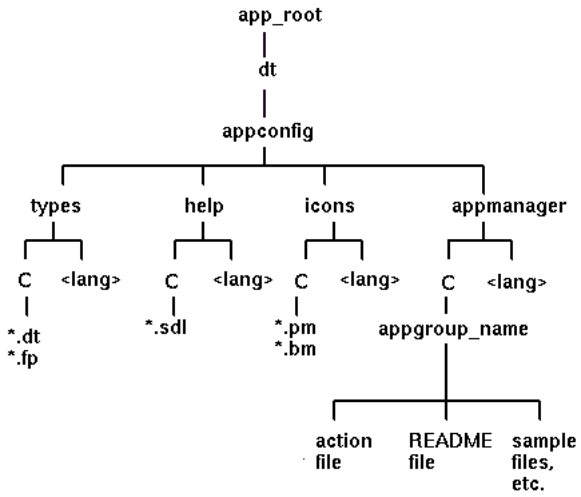


図 5–4　アプリケーションの root ディレクトリの下にある登録パッケージ

app_root/dt/appconfig ディレクトリの下にある構成フィールドの主なカテゴリは、表 5–1 のとおりです。

表 5–1　構成フィールドの主なカテゴリ

サブディレクトリ	内容
types	アクションおよびデータ型定義ファイル
help	デスクトップ・ヘルプ・ファイル
icons	アプリケーションのアクションとデータ型が使用するビットマップおよびピックスマップ・イメージ・ファイル
appmanager	アプリケーション・グループを作成するディレクトリとその内容

主なカテゴリはそれぞれ、言語依存ファイルのサブディレクトリです。デフォルトの言語ファイルは C ディレクトリにあります。

登録パッケージを作成するには

- 次のようなディレクトリを作成します。言語依存構成ファイルを指定する場合は、各言語について別のディレクトリを作成します。**1** つの言語だけを指定する場合は、ファイルを C ディレクトリに入れます。
  - *app_root*/dt/appconfig/types/*language*
  - *app_root*/dt/appconfig/help/*language*
  - *app_root*/dt/appconfig/icons/*language*
  - *app_root*/dt/appconfig/appmanager/*language*/*appgroup_name*
    appgroup_name は、アプリケーション・グループ名です。

たとえば、図 5–5 は、*appgroup_name* が「Media_Tools」であるグループを含むアプリケーション・マネージャを示します。

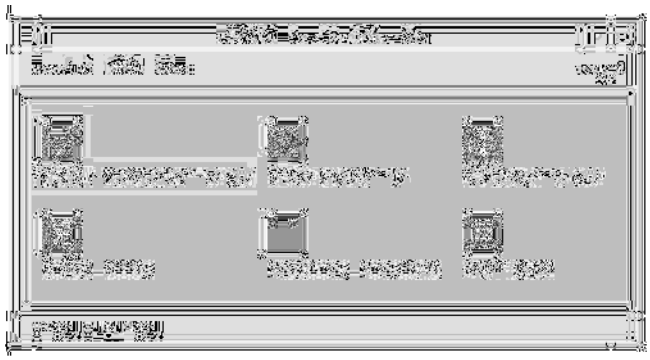


図 5–5　アプリケーション・マネージャのトップレベルにあるアプリケーション・グループ

dtappintegrate ツールは、types、help、icons、および appmanager ディレクトリのデスクトップ構成ファイルのみで動作します。アプリケーションのバイナリ実行可能ファイル、app-defaults、およびメッセージ・カタログ・ファイルは、別々に管理されます。

手順 4: アプリケーションのアクションとデータ型の作成

注 - アプリケーションのアクションとデータ型の作成例については、85ページの「登録パッケージの作成例」の手順 4 を参照してください。

アクションとデータ型は、アプリケーションのユーザ・インタフェースを提供します。

- アクションは、アプリケーションを起動するためのコマンドのユーザ・インタフェースを提供します。
- データ型は、アプリケーションのデータ・ファイル用にカスタマイズされた外観と動作を提供します。

アプリケーションが要求するアクションとデータ型

典型的なアプリケーションは、次のようなアクションとデータ型定義が必要です。

- アプリケーションを開くアクション
- アプリケーションのデータ・ファイルのデータ型 データ型を作成する場合は、次のアクションも作成してください。
  - アプリケーションのデータ・ファイルの [開く] アクション
  - アプリケーションのデータ・ファイルの [印刷] アクション
- アプリケーション・グループのデータ型 (詳細は、78ページの「固有のアイコンを使用するアプリケーション・グループの設定」を参照してください)。

アクションとデータ型がどのようにデスクトップで使用されるかについては、第 10 章を参照してください。

アクションおよびデータ型定義構成ファイルの位置

アクションとデータ型は、構成ファイルに定義されます。アクションとデータ型定義が入っているファイル名の取り決めとしては、必ず拡張子 `.dt` を付けるということだけです。取り決めに従って、*action_name*`.dt` または *application_name*`.dt` というファイル名が付けられます。

ディレクトリ *app_root*`/dt/appconfig/types/`*language* にあるアプリケーションの root の下に、アクションおよびデータ型の入っているファイルを置きます。デフォルトの *language* は `C` です。

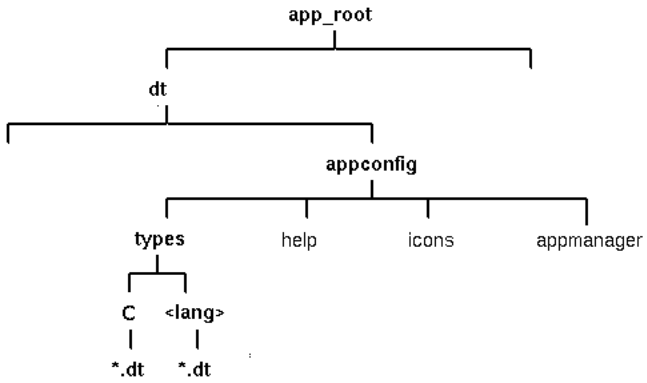


図 5–6　アクションおよびデータ型定義ファイル

アクションとデータ型の作成方法

次のいずれかの方法で、アクションとデータ型を作成できます。

- [アクション作成] ツールを使用する

  [アクション作成] には、入力するためのテキスト・フィールドがあり、使いやすいインタフェースを提供します。ただし、このツールには一定の制限があります。

- 定義を手入力で作成する

  この方法で作成する場合、定義を作成するための構文を知っていることが必要ですが、機能のすべてにアクセスできます。

[アクション作成] を使用してアクションとデータ型を作成するには

この手順は、[アクション作成] ユーティリティを使用して、アプリケーションのアクションとデータ型を作成します。

[アクション作成] の詳細は、オンライン・ヘルプまたは第 11 章を参照してください。

1. **[デスクトップアプリケーション]** アプリケーション・グループを開き、**[アクション作成]** をダブルクリックします。

2. **[アクション作成]** を使用して、アプリケーションとそのデータ型用のアクションとデータ型定義を作成します。

[アクション作成] で作成した構成ファイル
は、*HomeDirectory*/.dt/type/*action_name*.dt に書かれます。アクション・ファイル (アクションと同じ名前を持つ実行可能ファイル) は、ホーム・ディレクトリに置かれます。

3. ホーム・ディレクトリに作成されたアクション・ファイルを使用してアクションをテストします。

4. アクション定義ファイル ***HomeDirectory***/.dt/type/***action_name***.dt を ***app_root***/dt/appconfig/types/***language*** ディレクトリにコピーします。

5. アプリケーション・グループ・ディレクトリが作成された後 **(**詳細は、76ページの「手順 7: アプリケーション・グループの作成 」を参照してください**)**、アクション・ファイル ***HomeDirectory*/*action_name*** を ***app_root***/dt/appconfig/appmanager/***language*/*appgroup_name*** ディレクトリにコピーします。

アクションとデータ型を手入力で作成するには

- アプリケーションのアクションとデータ型が入っている構成ファイルを作成します。

アクションとデータ型定義ファイルは、*name*.dt というファイル名の命名規則に必ず従わなければなりません。

アクションとデータ型定義は、1 つまたは複数のファイルに分けて入れることができます。各ファイルは、システム管理者が簡単にアプリケーションへ接続できるファイル名を使用します。

アクション名とデータ型名は、必ず一語にしてください (埋め込みスペースも使用しないでください)。下線文字を使用することもできます。規則により、アクション名またはデータ型名の最初の文字は大文字にします。既存のアクション名またはファイル名は使用しないでください。上級ユーザおよびシステム管理者が、簡単にアプリケーションに接続できる名前を使用します。

アクション名と異なる名前のラベルが付いたアプリケーションのアイコンを使用する場合は、アクション定義に LABEL フィールドを取り込みます。

アクションとデータ型の作成方法については、次の章を参照してください。

- 第 10 章

- 第 11 章
- 第 12 章
- 第 13 章

手順 5: 登録パッケージへのヘルプ・ファイルの組み込み

注 - 登録パッケージへのヘルプ・ファイルの追加例については、85ページの「登録パッケージの作成例」の手順 5 を参照してください。

アプリケーションがデスクトップのヘルプ・ボリューム (デスクトップのヘルプ開発者キットで作成されたヘルプ・ボリューム) を取り込む場合、ヘルプ・ボリュームのマスタ・ファイル (`*.sdl`) をディレクトリ *app_root*`/appconfig/help/`*language* に置いてください。

ヘルプ・ファイルが使用するグラフィックは、通常 `graphics` サブディレクトリに置きます。グラフィックは、ヘルプ・ボリュームが作成されたときと同じ、マスタ・ヘルプ・ボリューム (`*.sdl`) に関連するディレクトリになければなりません。

アプリケーションがヘルプ・ボリュームを提供しない場合、ヘルプ開発者キットを使って作成できます。

ヘルプ・ボリュームの統合には、次の 2 つのレベルがあります。

- 完全統合

  デスクトップ・ヘルプを完全に統合すると、アイテムヘルプや [ヘルプ] メニューなどのアプリケーションからヘルプ・ボリュームにアクセスできます。この統合には、アプリケーションの実行可能ファイルの変更も含まれます。

- 部分統合

  デスクトップ・ヘルプを部分的に統合すると、ヘルプ・マネージャのトップレベルからデスクトップ・ヘルプを使用できます。ただし、アプリケーションのウィンドウからはヘルプ・ボリュームにアクセスできません。アプリケーション・グループからヘルプへアクセスできるアクションも提供されます。次の例にあるアクションは、ヘルプ・マスタ・ファイル `MyApp.sdl` にあるヘルプ・ボリュームを表示します。

```
ACTION OpenMyAppHelp
  {
    LABEL           MyAppHelp
    ARG_COUNT       0
    TYPE            COMMAND
    WINDOW_TYPE     NO_STDIO
```

```
    EXEC_STRING     /usr/dt/bin/dthelpview -helpVolume MyApp
    DESCRIPTION     Displays help for the MyApp application.
}
```

手順 6: アプリケーション用アイコンの作成

注 - アプリケーションのアイコン・ファイルの作成例については、85ページの「登録パッケージの作成例」の手順 6 を参照してください。

デスクトップは、アクション、データ型、アプリケーション・グループのデフォルト・アイコンを提供します。しかし、アプリケーション固有のアイコンを作成することもできます。

アイコンは、ディレクトリ *app_root*/dt/appconfig/icons/*language* にあります。

デスクトップに必要なアイコン

アプリケーションは、次のようなアイコン・イメージをデスクトップで使用します。

- アクション・アイコン — ダブルクリックするとアプリケーション (アクション) が起動するアイコンです。アプリケーションを起動するアクションの ICON フィールドで参照されます。

  サイズは、極小、中、大の 3 種類です。

- データ型アイコン — このアイコンは、ファイル・マネージャにあるアプリケーションのデータ・ファイルを表示するのに使用します。データ型定義の ICON フィールドで参照されます。

  アプリケーションが複数のデータ型をサポートする場合、各データ型ごとに異なるアイコンを指定してください。

  サイズは、極小と中の 2 種類です。

- アプリケーション・グループ・アイコン — アプリケーション・マネージャのトップレベルにあるディレクトリを示すアイコンです。アプリケーション・グループのためのデータ型定義の ICON フィールドで参照されます (詳細は、76 ページの「手順 7: アプリケーション・グループの作成 」を参照してください)。

  サイズは、極小と中の 2 種類です。

カラー (8 ビット以上) とモノクロ (8 ビット未満) ディスプレイをサポートするには、各アイコンでピックスマップとビットマップの両方のバージョンを用意する必要があります。

表 5–2　アイコン・ファイルの命名規則

サイズ	ピクセル・サイズ	ビットマップ名	ピックスマップ名
極小	16x16	*basename*.t.bm	*basename*.t.pm
中	32x32	*basename*.m.bm	*basename*.m.pm
大	48x48	*basename*.l.bm	*basename*.l.pm

ビットマップ・ファイルを提供しない場合、デスクトップはピックスマップ・ファイルのカラー指定を白黒にマップします。このマッピングでは、希望どおりに表示されないことがあります。

アイコンの詳細は、235ページの「アイコン・イメージ・ファイル」を参照してください。

手順 7: アプリケーション・グループの作成

注 - アプリケーション・グループの作成例については、85ページの「登録パッケージの作成例」の手順 7 を参照してください。

アプリケーションのアクションとデータ型定義を作成したら、ユーザが実際に見るアプリケーション・グループとその内容を作成するための構成ファイルを必ず作成してください。

アプリケーション・グループは、アプリケーション・マネージャのトップレベルにあるディレクトリです (詳細は、図 5–1 を参照してください)。

アプリケーション・グループの作成には、次の 3 つの手順があります。

- 登録パッケージにアプリケーション・グループ・ディレクトリを作成する
- (省略可能) アプリケーション・グループが一意のアイコンを使用するように設定する。これには、アプリケーション・グループ・ディレクトリのデータ型定義作成も含まれます。
- アプリケーション・グループの内容を作成する

アプリケーション・グループ・ディレクトリの作成

アプリケーション・グループを作成するには、図 5–7 のように、appmanager の下の登録パッケージにディレクトリを作成します。

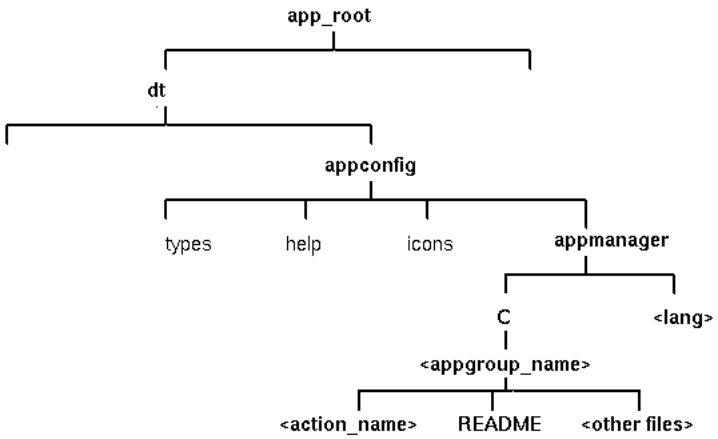


図 5–7　appmanager ディレクトリ

アプリケーション・グループ名

図 5–7 の <appgroup_name> は、アプリケーション・グループ名です。

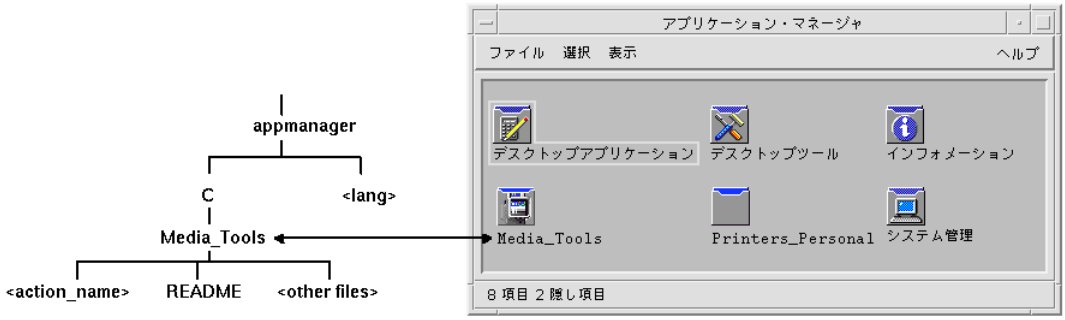


図 5–8　アプリケーション・グループ名 (<appgroup_name>)

アプリケーション・グループ名は、使用可能なファイル名 (ディレクトリ名) でかまいません。アプリケーションを説明する名前を使用します。

固有のアイコンを使用するアプリケーション・グループの設定

デスクトップはデフォルトのアプリケーション・グループ・アイコンを提供しますが、カスタム・アイコンが必要になる場合もあります。

アプリケーション・グループに固有のアイコンを指定する場合は、次のものを作成する必要があります。

- アプリケーション・マネージャのトップレベルに表示されるディレクトリのデータ型
- データ型の [開く] および [印刷] アクション

たとえば、Media_Tools という名前のアプリケーション・グループを作成するとします。ファイル *app_root*/dt/appconfig/types/*language*/*name*.dt にある次のデータ型定義が、アプリケーション・グループ・アイコンに一意のアイコンを割り当てます。

```
DATA_ATTRIBUTES  Media_ToolsAppgroup
 {
    ACTIONS       OpenInPlace,OpenNewView
    ICON          MediaTools
    DESCRIPTION   Double-click to open the Media_Tools \
                  application group
 }


DATA_CRITERIA  Media_ToolsAppgroupCriteria1
 {
    DATA_ATTRIBUTES_NAME Media_ToolsAppgroup
    MODE                 d
    PATH_PATTERN         */appmanager/*/Media_Tools
 }
```

定義の属性セクションが使用するアイコンを指定します。定義の条件セクションは、appmanager というディレクトリのサブディレクトリである Media_Tools というディレクトリにデータ型が定義されるように指定します。

図 5–9 に、アプリケーション・グループ名とデータ型定義との関係を示します。データ型定義の PATH_PATTERN フィールドは、アプリケーション・グループに固有のアイコンを結合します。

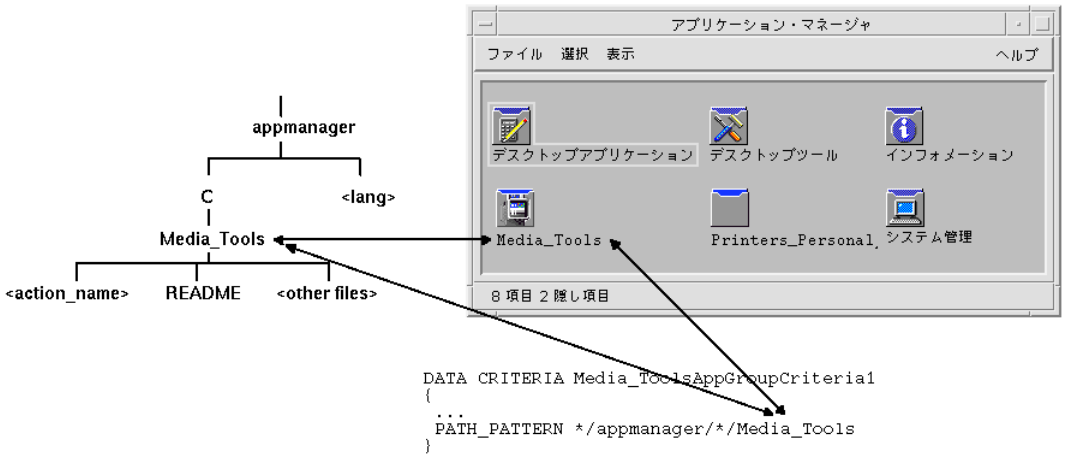


図 5–9　アプリケーション・グループが一意のアイコンを獲得する方法

アプリケーション・グループのデータ型の [開く] および [印刷] アクションも、次のように作成してください。

```
ACTION Open
 {
  ARG_TYPE   Media_ToolsAppGroup
  TYPE    MAP
  MAP_ACTION  OpenAppGroup
 }


ACTION Print
 {
  ARG_TYPE   Media_ToolsAppGroup
  TYPE    MAP
  MAP_ACTION  PrintAppGroup
 }
```

OpenAppGroup および PrintAppGroup アクションは、`/usr/dt/appconfig/types/`*language*`/dtappman.dt` に定義された組み込みアクションです。

アプリケーション・グループの内容の作成

アプリケーション・グループで最も重要な項目は、アプリケーションを起動するアイコン (アクション・アイコン) です。アプリケーション・グループに一連のアプリケーションが含まれている場合、通常は各アプリケーションのアイコンがあります。

次に、1 つ以上のアクション・アイコンの他に、アプリケーション・グループに含まれているものを示します。

- 1 つ以上の README ファイル
- 1 つ以上のサンプル・データ・ファイル
- テンプレート
- ダブルクリックしてヘルプ情報を表示するためのアイコン
- マニュアル・ページ
- 特殊なフロントパネル・コントロール

アプリケーション・グループには、サブディレクトリを含めることができます。

アクション・ファイル (アプリケーション・アイコン) の作成

アプリケーション・グループには、アプリケーションを起動するアイコンがあります。グループが一連のアプリケーションを提供する場合、各アプリケーションにアイコンがあります。これらのアイコンは基本のアクションを示すため、「アプリケーション・アイコン」または「アクション・アイコン」と呼びます。

アクション・アイコンは、次のように実行するアクションと同じ名前の実行可能ファイルを作成することによって作成します。

app_root/dt/appconfig/appmanager/*appgroup_name*/*action_name*

このファイルは、基本のアクションの視覚的な表示を作成することが目的であるため、「アクション・ファイル」と呼ばれます。

たとえば、BestTextEditor アプリケーションを実行する BestTextEditor というアクションを作成する場合、BestTextEditor という名前の実行可能ファイルを作成します。ファイル・マネージャとアプリケーション・マネージャでは、アクション・ファイルは、アクション定義で指定したアイコン・イメージを使用します。

図 5–10 は、アプリケーション・マネージャのウィンドウでのアクション定義、アクション・ファイル、実際の入力形式の関係を示します。

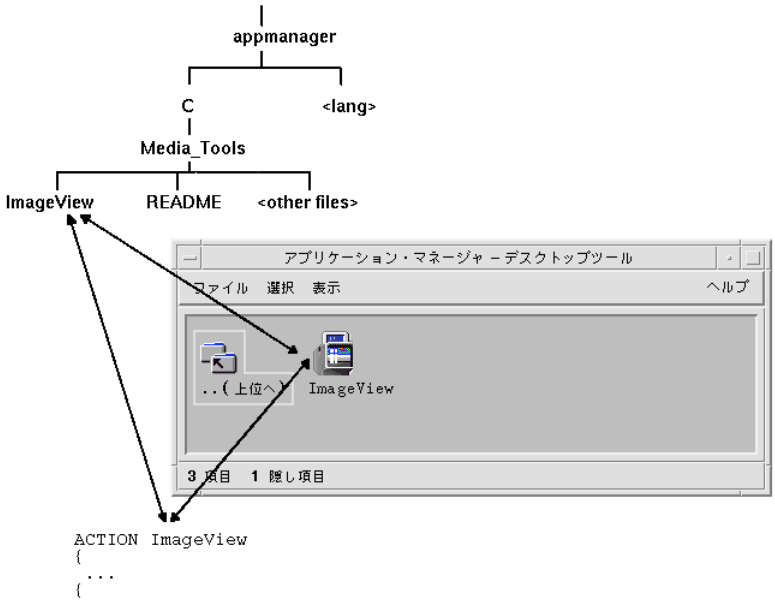


図 5–10　アプリケーション・グループのファイル

README ファイル

デスクトップは、アプリケーションの README ファイルに使用できる README データ型を提供します。次のいずれかの形式を使用します。

- `README`
- `readme`
- `README.*`
- `Read.*.Me`
- `read.*.me`
- `READ.*.ME`

特殊なフロントパネル・コントロールの作成

通常は、フロントパネル・コントロールの定義を指定する必要はありません。サブパネルの [アイコンのインストール] コントロールへアクション・アイコンをドロップすることにより、フロントパネルにアプリケーションを追加できます。

アクション・アイコンと異なる動作をするコントロールをユーザがインストールできるようにする場合、アプリケーションのコントロール定義を含むフロントパネル構成ファイルを作成する必要が生じます。たとえば、コントロールはファイルを監視し、そのファイルが変更されると表示を変更します。

フロントパネル構成ファイルは、*app_root*/dt/appconfig/types/*language* ディレクトリにあります。ファイル名の命名規則は、*name*.fp です。

コントロールを含む構成ファイルを提供すると、サブパネルの [アイコンのインストール] コントロールに *.fp ファイルをドロップすることにより、サブパネルにコントロールを追加できます。

たとえば、次の定義を、アプリケーション・グループのフロントパネル構成ファイルに指定できます。このファイルをサブパネルの [アイコンのインストール] コントロールへドロップする場合、BestTextEditor アプリケーションの 1 つのインスタンスを実行するサブパネルでコントロールが作成されます。BestTextEditor がすでに実行されている場合は、現在のワークスペースのウィンドウ・スタックの一番上にウィンドウが移動します。

```
CONTROL BestTextEditorControl
 {
  TYPE       icon
  ICON       BTEFPanel
  PUSH_RECALL     True
  CLIENT_NAME     BTEd
  PUSH_ACTION     BTEditor
  DROP_ACTION     BTEditor
  HELP_STRING     Starts the BestTextEditor application.
 }
```

フロントパネル構成ファイルの作成の詳細は、次を参照してください。

- 第 15 章
- dtfpfile(4) のマニュアル・ページ

手順 8: dtappintegrate を使用したアプリケーションの登録

注 - アプリケーションの登録例については、85ページの「登録パッケージの作成例」を参照してください。

アプリケーションの root の下に登録パッケージを作成した後、実際にアプリケーションを登録できます。

アプリケーションを登録すると、登録パッケージと、デスクトップ検索パスに沿って配置されているディレクトリがリンクされます (詳細は、84ページの「dtappintegrate によるアプリケーションの統合方法」を参照してください)。

アプリケーションを dtappintegrate で登録する

アプリケーションがデスクトップ化されている場合、dtappintegrate は通常、インストール・プロセスの最終手順として自動的に実行されます。自動的に実行されない場合、またはデスクトップ化されていないアプリケーションを統合するよう構成ファイルを作成した場合は、次のように手作業で dtappintegrate を実行できます。

1. **root** でログインします。
2. 次のコマンドを実行します。

   ```
   /usr/dt/bin/dtappintegrate -s  app_root
   ```

   app_root は、デスクトップ・アプリケーションの root ディレクトリです。詳細は、dtappintegrate(1) のマニュアルページを参照してください。
3. **[**デスクトップツール**]** アプリケーション・グループを開いて、**[**アプリケーションの再読込み**]** をダブルクリックします。
4. 次のようにアプリケーションが正しく登録されているか確認します。
   a. アプリケーション・マネージャのトップレベルを表示します。新規アプリケーション・グループがアプリケーション・マネージャに表示されているか確認します。
   b. アプリケーション・グループを開いて、アクション・アイコンをダブルクリックします。

dtappintegrate の構文とオプション

```
dtappintegrate -s app_root [-t target_path ] [-l language ] [-u]
```

-s *app_root*	必須パラメータ。アプリケーションをインストールするアプリケーションの root を指定します。
-t *target_path*	省略可能なパラメータ。システム上でのデフォルト位置は /etc/dt/appconfig です。デスクトップ構成ファイルをリンクする位置を指定します。必ずアプリケーション検索パスの位置を使用してください。
-l *language*	省略可能なパラメータ。デフォルトは全言語です。統合する言語依存デスクトップ構成ファイルを指定します。
-u	省略可能なパラメータ。アプリケーションを統合せず、統合中に設定されたリンクをすべて削除します。

dtappintegrate によるアプリケーションの統合方法

dtappintegrate は、インストールされたファイルと、デスクトップが構成ファイルを検索する位置とのリンクを設定します。

アクションとデータ型

dtappintegrate は、登録パッケージのアクションとデータ型定義ファイルから、アクション・データベースのヘルプ検索パスに沿ったシステム全体のディレクトリへシンボリック・リンクを作成します。これは

app_root/dt/appconfig/types/*language*/*.dt

を

/etc/dt/appconfig/types/*language*/*.dt

へリンクさせます。

ヘルプ情報ファイル

dtappintegrate は、登録パッケージのヘルプ・ファイルから、ヘルプ検索パスに沿ったシステム全体のディレクトリへシンボリック・リンクを作成します。これは

app_root/dt/appconfig/help/*language*/*help_file*.sdl

を

/etc/dt/appconfig/help/*language*/*help_file*.sdl

へリンクさせます。

アイコン・ファイル

dtappintegrate は、登録パッケージのアイコン・ファイルから、アイコン検索パスに沿ったシステム全体のディレクトリへシンボリック・リンクを作成します。これは

app_root/dt/appconfig/icons/*language*/*icon_files*

を

/etc/dt/appconfig/icons/*language*/*icon_files*

へリンクさせます。

アプリケーション・グループ

アプリケーションのアプリケーション・グループをアプリケーション・マネージャのトップレベルに置くため、dtappintegrate は、登録パッケージのアプリケーション・グループ・ディレクトリとアプリケーション検索パスに沿ったシステム全体の場所とをリンクします。これは

app_root/dt/appconfig/appmanager/*language*/*appgroup_name*

を

/etc/dt/appconfig/appmanager/*language*/*appgroup_name*

へリンクさせます。

登録パッケージの作成例

次の手順では、既存のデスクトップ化されていない BestTextEditor というアプリケーションに登録パッケージを作成します。

BestTextEditor について知っておくべき情報

この例では、BestTextEditor アプリケーションについて、次のことを想定しています。

- ディレクトリ /usr/BTE にインストールされています。
- ユーザのセッション言語はデフォルト値の C です。
- BestTextEditor を起動するコマンド行は次のとおりです。

  ```
  BTEd [filename]
  ```

  filename は、新規ウィンドウで開くデータファイル名です。BestTextEditor は、独自のウィンドウを作成します。つまり、端末エミュレータ・ウィンドウ内で実行できません。

- BestTextEditor は、次の 2 種類のデータ・ファイルを作成して使用します。

  - ドキュメンテーション・ファイル。ファイル名の命名規則は *.bte です。BestTextEditor は .bte データ・ファイルを印刷するためにコマンド行を提供します。コマンド構文は次のとおりです。

    ```
    BTEPrint [-d destination] [-s] filename
    ```

    –d *destination* — 宛先プリンタを指定します。

    –s — サイレント印刷を指定します。アプリケーションのプリント・ダイアログ・ボックスは表示されません。

    filename — 印刷するファイルを指定します。

  - テンプレート・ファイル。ファイル名の命名規則は *.tpl です。テンプレート・ファイルは印刷できません。

- BestTextEditor の、既存のデスクトップでない app-defaults ファイルには、インタフェース・フォントと、フォアグラウンド・カラーおよびバックグラウンド・カラーのリソースがあります。
- BestTextEditor のオンライン・ヘルプ・ボリュームは、デスクトップのヘルプ開発キットを使用して作成されます。オンライン・ヘルプ・ボリュームは組み込まれると、次のソース・ファイルを使用します。

  ```
  .../BTEHelp.htg
     .../graphics/BTE1.xwd
     .../graphics/BTE2.xwd
  ```

次に、ファイル .../BTEHelp.sdl を生成します。

BestTextEditor を登録するための手順

次の手順で BestTextEditor を登録します。

1. フォント・リソースとカラー・リソースを修正します。

   BestTextEditor の `app-defaults` ファイルでは、以下を設定するリソースを削除します。

   - テキストのフォント
   - フォアグラウンドおよびバックグラウンドのカラー

2. アプリケーションの **root** を作成します。

   次のディレクトリを作成します。

   ```
   /desktop_approots/BTE
   ```

   既存のアプリケーションを統合する場合、アプリケーションのインストール位置以外に、アプリケーションの root ディレクトリを作成してください。そうしないと、アプリケーションを更新したときに、作成した構成ファイルが削除されることがあります。

3. 登録パッケージ・ディレクトリを作成します。

   次のディレクトリを作成します。

   ```
   /desktop_approots/BTE/dt/appconfig/types/C
   /desktop_approots/BTE/dt/appconfig/help/C
   /desktop_approots/BTE/dt/appconfig/icons/C
   /desktop_approots/BTE/dt/appconfig/appmanager/C/BestTextEditor
   ```

4. アプリケーションのアクションとデータ型を作成します。

   a. アクションおよびデータ型定義の構成ファイルを作成します。

   ```
   /desktop_approots/BTE/dt/appconfig/types/C/BTE.dt
   ```

   b. **BestTextEditor** を実行するためのアクション定義を作成します。

   ```
   ACTION BTEditor
        {
          WINDOW_TYPE    NO_STDIO
          ICON       BTERun
          DESCRIPTION    Double-click this icon or drop \
                 a BTE data file on it to run \
                 BestTextEditor.
          EXEC_STRING    /usr/BTE/BTEd %Arg_1%
        }
   ```

c. *.bte ファイルのデータ型を作成します。

```
DATA_ATTRIBUTES BTEDataFile
     {
       DESCRIPTION    BestTextEditor data file.
       ICON      BTEData
       ACTIONS     Open,Print
     }

   DATA_CRITERIA BTEDataFileCriteria1
     {
       DATA_ATTRIBUTES_NAME   BTEDataFile
       NAME_PATTERN      *.bte
       MODE         f
     }
```

d. *.tpl ファイルのデータ型を作成します。

```
DATA_ATTRIBUTES BTETemplateFile
     {
       DESCRIPTION   BestTextEditor template file.
       ICON     BTETempl
       ACTIONS    Open
     }

   DATA_CRITERIAL BTETemplateFileCriteria1
     {
       DATA_ATTRIBUTES_NAME   BTETemplateFile
       NAME_PATTERN      *.tpl
       MODE         f
     }
```

e. *.bte ファイルの **[開く]** アクションを作成します。

```
ACTION Open
     {
       ARG_TYPE   BTEDataFile
       TYPE    MAP
       MAP_ACTION  BTEditor
     }
```

f. *.bte ファイルの **[印刷]** アクションを作成します。

次の例は、データ・ファイルを印刷する簡単な [印刷] アクションです。これらのアクションには、LPDEST 環境変数が必要で、–s 印刷オプションを無視します (LPDEST を設定しない場合、アクションは異常終了する可能性があります)。

```
ACTION Print
     {
       ARG_TYPE    BTEDataFile
       TYPE     MAP
       MAP_ACTION   BTEPrintData
```

```
    }


ACTION BTEPrintData
    {
      WINDOW_TYPE    NO_STDIO
      EXEC_STRING    BTEPrint -d $LPDEST %Arg_1%
    }
```

次は、BTEPrintData アクションと付随するスクリプトの別のバージョンを示します。アクションとスクリプトは、LPDEST が設定されていないか、サイレント印刷が要求されている状況を処理します。

```
ACTION BTEPrintData
    {
      WINDOW_TYPE    NO_STDIO
      EXEC_STRING    /usr/BTE/bin/BTEenvprint \
             %(File)Arg_1%
    }
```

/usr/BTE/bin/BTEenvprint スクリプトは次のとおりです。

```
# BTEenvprint
#!/bin/sh
DEST=''''
SILENT=''''
if [ $LPDEST ] ; then
DEST=''-d $LPDEST''
fi
BTEPrint $DEST SILENT $1
```

g. *.tpl ファイルの **[開く]** アクションを作成します。

```
ACTION Open
  {
    ARG_TYPE   BTETemplateFile
    TYPE    MAP
    MAP_ACTION  BTEditor
  }
```

h. *.tpl ファイルの **[印刷]** アクションを作成します。

```
ACTION Print
  {
    ARG_TYPES  BTETemplateFile
    TYPE    MAP
    MAP_ACTION  NoPrint
  }
```

NoPrint は、ファイルが印刷できないことをユーザに通知するダイアログ・ボックスを表示する組み込みアクションです。

5. ヘルプ・ファイルを登録パッケージに組み込みます。

   a. 次の位置に置きます。

```
/desktop_approots/BTE/dt/appconfig/help/C/BTEHelp.sdl
   /desktop_approots/BTE/dt/appconfig/help/C/graphics/BTE1.xwd
   /desktop_approots/BTE/dt/appconfig/help/C/graphics/BTE2.xwd
```

   b. 次のファイルを作成します。

```
/desktop_approots/BTE/dt/appconfig/types/C/BTEhelp.dt
```

   次のアクション定義をファイルに入れます。

```
ACTION BTEHelp
     {
       WINDOW_TYPE    NO_STDIO
       EXEC_STRING    /usr/dt/bin/dthelpview -helpVolume \
              BTEHelp.sdl
       DESCRIPTION    Opens the BestTextEditor help volume.
     }
```

6. アプリケーションのアイコンを作成します。

   [アイコン・エディタ] を使用して、アイコンを作成します。表 5–3 に示すサイズのガイドラインを使用します。

表 **5–3** アイコン・サイズのガイドライン

名前	サイズ
basename.t.pm	16x16
basename.m.pm	32x32
basename.l.pm	64x64

   以下のアイコン・ファイルを次のディレクトリに作成します。

```
/desktop_approots/BTE/dt/appconfig/icons/C
```

- アプリケーションを実行するアクションを示すアイコン:
  BTERun.t.pm、BTERun.m.pm、BTERun.l.pm
- *.bte ファイルを示すアイコン: BTEData.t.pm、BTEData.m.pm
- *.tpl ファイルを示すアイコン: BTETempl.t.pm、BTETempl.m.pm
- アプリケーション・グループ (手順 7 で使用) を示すアイコン:
  BTEApp.t.pm、BTEApp.m.pm

7. アプリケーション・グループを作成します。

a. まだ作成していない場合は、ディレクトリを作成します。

```
/desktop_approots/BTE/dt/appconfig/appmanager/C/BestTextEditor
```

b. **(省略可能)** アプリケーション・グループのデータ型と関連するアクションを作成して、アプリケーション・グループ・アイコンに一意のアイコンを作成します。この手順を省略すると、アプリケーション・グループはデフォルト・アイコンを使用します。

次のデータ型とアクション定義をファイル /desktop_approots/BTE/dt/appconfig/types/C/BTE.dt に追加します。データ型は、アイコンを BestTextEditor アプリケーション・グループが使用するように指定します。アクションは、組み込みアプリケーション・グループと同様の、[開く] と [印刷] の動作を提供します。

```
DATA_ATTRIBUTES BestTextEditorAppGroup
     {
       ACTIONS  OpenInPlace,OpenNewView
       ICON   BTEApp
     {


DATA_CRITERIA BestTextEditorAppGroupCriterial
     {
       DATA_ATTRIBUTES_NAME BestTextEditorAppGroup
       MODE       d
       PATH_PATTERN    */appmanager/*/BestTextEditor
     }


ACTION Open
     {
       ARG_TYPE    BestTextEditorAppGroup
       TYPE     MAP
       MAP_ACTION   OpenAppGroup
     }


ACTION Print
     {
       ARG_TYPE    BestTextEditorAppGroup
```

```
  TYPE     MAP
  MAP_ACTION   PrintAppGroup
}
```

c. アプリケーションを起動するアプリケーション・グループにアイコンを作成します。これを行うには、次のファイルを作成し、ファイルを実行可能にします。

```
/desktop_approots/BTE/dt/appconfig/appmanager/C/BestTextEditor/BTEditor
```

d. ヘルプ・ボリュームを開くアプリケーション・グループにアクション・ファイルを作成します。これを行うには、次のファイルを作成し、ファイルを実行可能にします。

```
/desktop_approots/BTE/dt/appconfig/appmanager/C/BestTextEditor/BTEHelp
```

e. アプリケーション・グループに、**README** ファイル、サンプル・データ、テンプレート・ファイルなどの他のファイルを置きます。

8. アプリケーションを登録します。

端末エミュレータ・ウィンドウで次の操作を実行します。

a. **root** でログインします。

b. 次のコマンドを実行します。

```
/usr/dt/bin/dtappintegrate -s /desktop_approots/BTE
```

c. **[**デスクトップツール**]** アプリケーション・グループを開き、**[**アプリケーションの再読込み**]** をダブルクリックします。

第 6 章

さまざまな構成

この章では、設定とシステム管理について説明します。

Solaris CDE ディレクトリ構造

この節では、ユーザのデスクトップ環境に含まれる主要なディレクトリを説明します。

/usr/dt

このディレクトリは、Solaris CDE のインストール場所です。遠隔ファイル・サーバからのマウント・ポイントの場合もあります。表 6–1 に、/usr/dt のサブディレクトリを説明します。

表 6–1 /usr/dt のサブディレクトリ

サブディレクトリ	説明
/bin	Solaris CDE アプリケーションとユーティリティ
/lib	Solaris CDE 実行時共有ライブラリ
/config	デフォルトのシステム構成ファイル
/man	(省略可能) マニュアル・ページ
/app-defaults	デフォルトのアプリケーションのリソース
/appconfig	デフォルトのアプリケーションのアイコン、タイプ、アクション
/examples	(省略可能) CDE コード/プログラム例
/include	(省略可能) 開発者用インクルード・ファイル
/palettes	カラー・パレット
/share	CDE AnswerBook™ とデフォルトの背景

/etc/dt

このディレクトリには、カスタマイズされた、ワークステーション固有の構成ファイルが入っています。これらのファイルを使用して、次のようにユーザの環境をカスタマイズできます。

- X サーバ構成オプションの設定
- 複数画面の使用
- ワークステーションのアクション・ファイル、データ・タイプ、アイコン、フォントのカスタマイズ

/var/dt

このディレクトリは、ログイン・マネージャやアプリケーション・マネージャなどの、Solaris CDE アプリケーション用の一時ファイルを格納するために使用されます。

$HomeDirectory

このディレクトリには、ユーザのデスクトップの設定に関連する、ユーザ固有のファイルが入っています。たとえば、アプリケーション、カラー・スキーム、ワークスペースのメニューやフロント・パネルの変更、エラー・ログなどがあります。

主要な構成ファイル

ユーザのデスクトップ環境のほとんどのカスタマイズは、多くのファイルを必要とします。例としては、次の 2 つのファイルがあります。

- /usr/dt/config/Xconfig
- /usr/dt/config/Xservers

Xconfig

Xconfig は、dtlogin で使用されるマスタ構成ファイルです。このファイルは、ログイン・リソースを設定して、dtlogin で必要な追加のファイルの位置を指定します。次の例は、デフォルトの Xconfig ファイル中の数行を示します。

```
Dtlogin.errorLogFile:       /var/dt/Xerrors

Dtlogin.servers:            /usr/dt/config/Xservers

Dtlogin*session:            /usr/dt/bin/Xsession
```

このファイルの詳細は、付録 A を参照してください。このファイル中にも詳しい説明があります。

Xservers

dtlogin は、ベースとなる XDM (X ディスプレイ・マネージャ) と同様に、Xservers ファイルを使用して、ローカルの X サーバの起動方法を指定します。デフォルトでは、このファイルの最後の行は、次のようになります。

```
:0 Local local@console /usr/openwin/bin/Xsun :0
```

- :0 は、X サーバ・ディスプレイが <localhost:0> であることを意味します。
- local は、新しい X11 サーバがローカルで起動することを示します。

注 - 実行中の X11 サーバに接続する場合は、local を foreign に置き換えます。

- console は、コマンド行ログインが /dev/console にエスケープすることを意味します。
- /usr/openwin/bin/Xsun は、X11 サーバへのパスです。

注 - Solaris CDE と OpenWindows 環境は、同じ X11 サーバを実行します。

このファイルの詳細は、dtlogin(1) のマニュアル・ページを参照してください。このファイル中にも詳しい説明があります。

ログイン・サーバの起動

ログイン・サーバは、通常、システムがブートしたときに自動的に起動します。また、コマンド行からもログイン・サーバを起動できます。この場合、ユーザは、まず root ユーザとしてログインしなければなりません。

システムが起動したときに、ログイン・サーバが起動するように設定するには、次のようにします。

◆ 次のコマンドを入力して、**Return** キーを押します。

```
# /usr/dt/bin/dtconfig -e
```

これにより、S99dtlogin ファイルが、ユーザの /etc/rc2.d ディレクトリに追加されます。ログイン・サーバは、リブート時に自動的に起動します。

システムが起動したときに、ログイン・サーバが自動的に起動しないように設定するには、次のようにします。

- 次のコマンドを入力して、**Return** キーを押します。

```
# /usr/dt/bin/dtconfig -d
```

コマンド行からログイン・サーバを起動するには、次のようにします。

- 次のコマンドを入力して、**Return** キーを押します。

```
# /usr/dt/bin/dtlogin -daemon; exit
```

注 - コマンド行からログイン・サーバを起動すると、一時的な構成テスト用に利用できますが、通常、ログイン・サーバは、システムが起動したときに起動するようにしてください。

ログイン・サーバ、X サーバ、および Solaris CDE デスクトップ全体を終了するには、次のようにします。

- 次のコマンドを入力して、**Return** キーを押します。

```
# /usr/dt/bin/dtconfig -kill
```

デスクトップ構成ユーティリティ `dtconfig` の詳細は、付録 A を参照してください。この付録には、`dtconfig(1)` のマニュアル・ページが掲載されています。

[コマンド行ログイン] オプションを使用して、Solaris CDE ログイン画面を終了するには、次のようにします。

- **Solaris CDE** ログイン画面上で、**[オプション]** メニューから **[コマンド行ログイン]** を選択します。画面が消去された後 **Return** キーを押すと、ログイン・プロンプトが表示されます。

注 - デスクトップ・ログイン・デーモンは、コマンド行ログイン終了後、自動的にデスクトップ・ログイン画面を起動します。

---

1 つの端末エミュレーション・ウィンドウを起動するには、次のようにします。

- **Solaris CDE** ログイン画面上で、**[**オプション**]** メニューの **[**セッション**]** サブメニューから **[**復旧セッション**]** を選択します。

X サーバを実行したままにする場合は、[セッション] サブメニューから [復旧セッション] を選択します。これによって、1 つの xterm ウィンドウが起動します。[コマンド行ログイン] オプションが利用できない場合でも、[復旧セッション] オプションはいつでも利用できます。

他のワークステーションまたはネットワーク・サーバのインストール場所からインストールされた CDE をマウントする

この手順は、ユーザのローカルのディスク容量を使用しないので、ユーザのワークステーションに必要なディスク容量がない場合は、他のワークステーションまたはネットワーク・サーバのインストール場所にすでにインストールされている CDE をマウントできます。

注 - /usr/dt ディレクトリ構造は、Solaris リリースとクライアント・ワークステーション間で異なります (SPARC™ のディレクトリ構造は x86 のディレクトリ構造とは異なります)。このため、クライアント・ワークステーションは、適切な NFS サーバに /usr/dt イメージをマウントしなければなりません。たとえば、Solaris 2.5 の SPARC システムは、/usr/dt に CDE をインストールしている他の Solaris 2.5 システムから、/usr/dt をマウントします。

▼ インストール済み CDE をマウントするには

1. すでにインストールし終わっているワークステーションまたはネットワーク・サーバの /usr/dt ディレクトリを、ユーザのワークステーションの /usr/dt ディレクトリにマウントします。

2. /usr/dt/bin/dtconfig –inetd と入力します。

3. 次のように入力して、**Solaris** デスクトップ・ログインを有効にします。

   ```
   /usr/dt/bin/dtconfig -e
   ```

4. ワークステーションをリブートします。

デスクトップ構成ユーティリティ dtconfig の詳細は、付録 A を参照してください。この付録には、dtconfig(1) のマニュアル・ページが掲載されています。

▼ マウントされた CDE ディレクトリをマウント解除するには

1. 次のように入力して、**Solaris** デスクトップ・ログインを無効にします。

```
/usr/dt/bin/dtconfig -d
```

2. /usr/dt/bin/dtconfig –inetd.ow と入力します。
3. /usr/dt をマウント解除します。
4. ワークステーションをリブートします。

デスクトップ構成ユーティリティ dtconfig の詳細は、付録 A を参照してください。この付録には、dtconfig(1) のマニュアル・ページが掲載されています。

複数の画面を使用するためのデスクトップの構成

標準的なログインでは、1 つの画面でデスクトップを起動します。Xconfig ファイルを編集すると、複数の画面でデスクトップを起動できます。このファイルを変更するには、root ユーザとしてログインしなければなりません。

注 - ログイン構成情報を迅速に編集するには、[復旧セッション] オプションを使用して、デスクトップ全体ではなく、1 つの xterm ウィンドウを実行してください。

▼ 複数の画面でデスクトップを起動するには

1. 次のコマンドを入力してから **Return** キーを押し、Xserver ファイルのコピーを作成します。

```
# cp /usr/dt/config/Xservers /etc/dt/config/Xservers
```

注 - /etc/dt/config/Xservers は、/usr/dt/config/Xservers の設定を無効にします。

---

2. /etc/dt/config/Xservers ファイルを編集して、**2** つのフレーム・バッファ **(**画面**)** を設定します。

   a. 次の行を探します。

   ```
   :0 Local local_uid@console root /usr/openwin/bin/Xsun :0
   ```

   b. この行の最後に、**2** つの –dev オプションを追加します。この例では、/dev/fb0 と /dev/fb1 という名前であると仮定します。スペースの後に、次のように追加します。

   ```
   -dev /dev/fb0 -dev /dev/fb1
   ```

3. **[**オプション**]** メニューから **[**ログイン画面のリセット**]** を選択します。

注 - Solaris CDE を複数の画面で実行する場合、フロント・パネルはそれぞれの画面に表示されます。構成ファイルは、フロントパネルごとに別々に設定できます。

ネットワーク接続されたデスクトップ

Solaris CDE ログイン・マネージャは、ネットワークを認識します。デフォルトでは、ログイン画面は、Solaris CDE チューザの照会に応答します。

リモートホストのリストを表示するには、次のようにします。

- **Solaris CDE ログイン画面上で、[リモートログイン] メニューから [リストからホストを選択] を選択します。**

ログイン画面を使用しないでチューザを実行するには、105ページの「X 端末としてのワークステーションの使用」を参照してください。次の画面は、通常、チューザで利用可能なサーバのリストを示します。

リストからアイドル状態のサーバを選択すると、そのリソースをチューザ経由で使用し、Solaris CDE にログインできます。

Solaris CDE ログイン・マネージャが動作している、すべてのネットワーク接続されたワークステーションは、ローカル・デスクトップ・ユーザと、ワークステーショ

ンを Solaris CDE デスクトップ・サーバとして使用している X 端末ユーザを含む複数のリモート・ユーザの両方をサポートできます (図 6–1 を参照)。

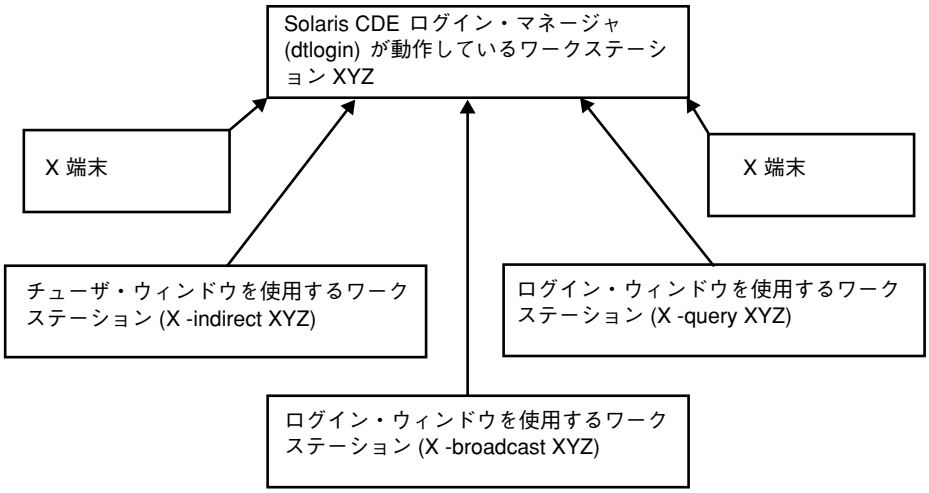


図 6–1　ネットワーク環境のデスクトップ

注 - 図 6–1 は、複数のハードウェアが混在する環境にも適用できます。

X 端末の使用

XDM プロトコルをサポートする X 端末は、チューザを使用して Solaris CDE にログインできます。チューザを実行するには、105ページの「X 端末としてのワークステーションの使用」を参照してください。SPARC Xterminal ソフトウェアのバージョン 2.0 以降は、Solaris CDE で正常に動作します。「Xterminal コントロール」ウィンドウの「端末の設定」をクリックし、「カテゴリ」から「セッション」を選択すると、図 6–2 のような設定画面が表示されます。

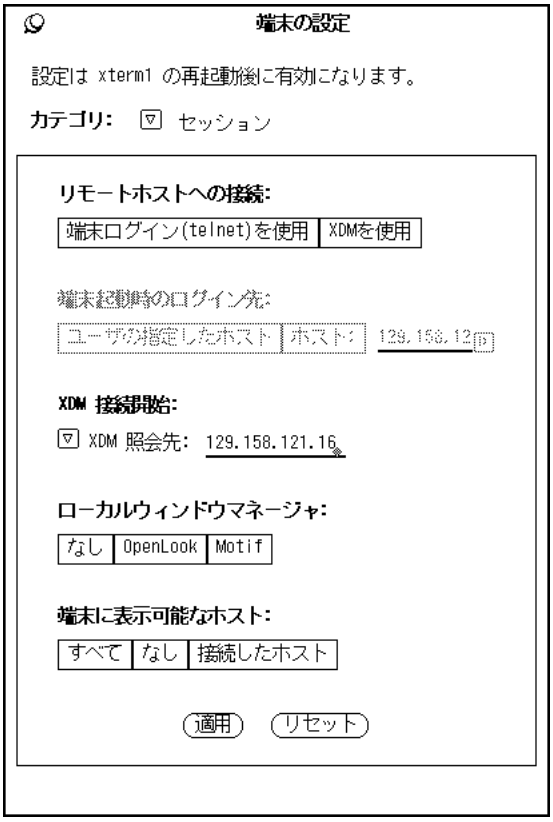
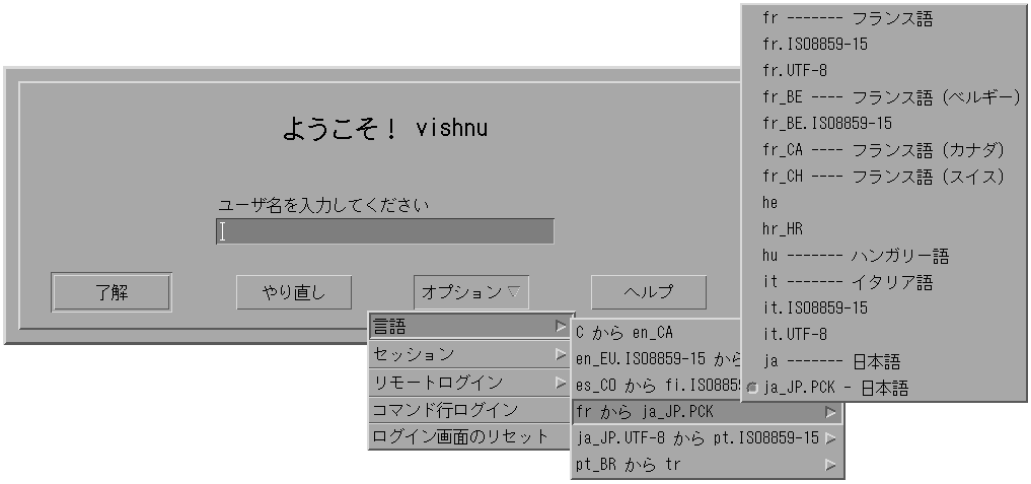


図 6–2　セッション設定画面

[ローカルウィンドウマネージャ] で [なし]、Solaris CDE が動作しているリモートシステムへの XDM 接続用に [XDM 接続開始] で [XDM 照会先] をそれぞれ選択します。

ログイン・ロケールとフォント・パス

Solaris CDE にログインする際には、言語を選択します。次の Solaris CDE ログイン画面は、[オプション] メニューの [言語] を選択しているところを示します。



ワークステーション上では、選択する言語とシステムのベースである「C」ロケールの両方に関連するフォント (と別名) を含むように、フォント・パスは自動的に設定されます。X 端末の場合、これらのフォントは、X11 フォント・サーバによって自動的に提供されます。

X 端末としてのワークステーションの使用

古いワークステーション、または 16M バイト以下のメモリを持つワークステーションを使用していて、Solaris 2.4 以降 (あるいは、X サーバ・ジャンボ・パッチをあてた Solaris 2.3) のバージョンを読み込んでいる場合は、ワークステーションを X 端末として使用できます。

注 - 次の手順を実行したのに、ワークステーションの画面でフォントが正しく表示されない場合は、Solaris CDE がホスト・ワークステーションにインストールされていない可能性があります。`pkgadd` ユーティリティを使用して、Solaris CDE 1.0 リリースから `SUNWdtft` Solaris CDE フォント・パッケージを追加してください。

▼ チューザを使用して、ホスト CDE ログインを選択するには

1. 実行中のウィンドウ・システムをすべて終了します。

2. チューザを起動します。コンソール・コマンド行から、次のコマンドを入力します (csh を使用している場合)。

```
setenv OPENWINHOME /usr/openwin
```

```
/usr/openwin/bin/X -indirect CDE_login_host
```

X サーバが起動して、ホストからチューザ・ウィンドウが表示されます。

3. チューザ・ウィンドウからログイン・ホストを選択します。

▼ 固有のホスト CDE ログインを使用するには

♦ 次のコマンドを入力して、**Return** キーを押します。

```
/usr/openwin/bin/X -once -query CDE_login_host
```

X 端末が起動して、ホストからログイン画面が表示されます。–once オプションを指定すると、サーバは、1 回のログインまたはログアウト・セッション後に終了します。–once オプションを指定していない場合は、Solaris CDE からログアウト後、ログイン画面がもう一度表示されます。

▼ 最初に利用可能なホスト・ログインを使用するには

♦ 次のコマンドを入力して、**Return** キーを押します。

```
/usr/openwin/bin/X -broadcast
```

X 端末が起動します。ローカルのサブネット上では、X 端末は、XDM (X ディスプレイ・マネージャ) ログイン・サービス用の要求を送信します。Solaris CDE ログイン・マネージャ (または他の XDM ベースのログイン・ウィンドウ) を実行しているシステムがサブネット上に存在する場合、最初に応答したホストがログイン・ウィンドウをユーザのデスクトップに表示します。

特別な CDE 構成

この節では、特別な構成について説明します。

メール印刷のカスタマイズ

CDE メール・プログラムは、DTMAIL_FILE データ型用に定義されている [印刷] アクションを使用して、メール・メッセージを出力します。OpenWindows のメール・ツールで指定した印刷スクリプトは使用しません。印刷の動作を変更するには、この [印刷] アクションを変更しなければなりません。

[印刷] アクションを変更するには

1. エディタを使用して、次のファイルを作成します。

   HomeDirectory/.dt/types/dtmail.dt

2. このファイルに次の行を入力します。

```
#
   # Override default Print action for mailboxes
   #
   ACTION Print
   {
   LABEL           Print
   ARG_TYPE        DTMAIL_FILE
   TYPE            COMMAND
   WINDOW_TYPE     NO_STDIO
   EXEC_STRING     sh -c ' \
                      dtmailpr -p -f %(File)Arg_1% | mp -m -l | \
                      dtlp -u %(File)Arg_1%;'
   }
```

注 - dtmailpr は、アタッチメントを削除して、プレーンテキストにする印刷フィルタです。dtlp は、lp に対する標準 CDE インタフェースです。%(File)Arg_1% は、印刷するファイルです。

---

3. 印刷コマンドを含むように EXEC_STRING を変更します。

4. メール・プログラムを起動し直します。

カレンダの新しいデータ形式への変換

バージョン 4 拡張可能なデータ形式は、CDE カレンダでサポートされている、新しい形式です。OpenWindows のカレンダ・マネージャでは、この形式を読むことができません。OpenWindows と CDE のプラットフォーム間を切り替えて使用する必要がある場合は、ユーザのカレンダをバージョン 4 データ形式に変換しないでください。ただし、`sdtcm_convert` スクリプトを使用すると、カレンダから余分な部分を削除できます。

注 - カレンダのデータ・バージョンを調べるには、カレンダの [ヘルプ] メニューから [カレンダについて] を選択します。

それ以外の場合、`sdtcm_convert` スクリプトを使用して、カレンダをバージョン 4 データ形式に変換します。

このファイルの詳細は、`sdtcm_convert(1)` のマニュアル・ページを参照してください。

注 - End User CDE パッケージだけをインストールした場合は、マニュアル・ページはインストールされません。マニュアル・ページを参照するには、`pkgadd` ユーティリティを使用して、マニュアル・ページをインストールしなければなりません。

ネットワークからの **AnswerBook** パッケージの追加

Solaris CDE をインストールする場合、インストール・メニューには、AnswerBook CDE パッケージをインストールするためのオプションが用意されています。[YES] を選択した場合、AnswerBook パッケージは、`/usr/dt/share/answerbooks/`*language* ディレクトリにインストールされます (*language* は、特定の AnswerBook の翻訳言語のローカル名)。

AnswerBook パッケージをインストールするには、117M バイトのディスク容量が必要です。AnswerBook パッケージをインストールすると、次の 2 つのオプションを利用できます。

- AnswerBook パッケージを `/usr/dt/share/answerbooks/`*language* にマウント (または `/net/. . .` からリンク) できます。

- AB_CARDCATALOG 環境変数を *HomeDirectory*/.dtprofile に追加できます。たとえば、ネットワーク上にエクスポートされたファイルがある場合は、次の行を *HomeDirectory*/.dtprofile に追加します。

```
export AB_CARDCATALOG=/net/hostname/usr/dt/share/answerbooks \
    /language/ab_cardcatalog
```

CDE デスクトップ外からの CDE 環境の設定

Solaris CDE アプリケーションが使用する環境変数は、CDE デスクトップ以外の場所からも設定できます。たとえば、リモート・ワークステーションにログインして、CDE アプリケーションをユーザのワークステーションに表示させたいとします。CDE ユーティリティ dtsearchpath を使用すると、さまざまな CDE シェル環境変数を設定できます。

Bourne シェルと Korn シェルの場合は、次のコマンドを入力します。

```
eval `/usr/dt/bin/dtsearchpath`
```

C シェルの場合は、次のコマンドを入力します。

```
eval `/usr/dt/bin/dtsearchpath -c`
```

このコマンドに続いて、DISPLAY をローカルのワークステーションに設定し、CDE アプリケーションをリモートから実行できます。結果は、ローカルのワークステーション上に表示されます。

注 - この例では、Bourne シェルと Korn シェルの dtsearchpath 構文は、CDE を実行しているすべてのプラットフォームで利用できる、CDE サンプル実装の一部です。しかし、C シェル (–c) オプションは、Sun のプラットフォームでしか利用できません。

デスクトップ環境ファイル

CDE デスクトップは、OpenWindows DeskSet™ アプリケーションに、実行時環境を提供します。この機能のために、特別な CDE 設定を行う必要はありません。この環境設定の一部は、次のファイルで提供されます。

```
/usr/dt/config/Xsession.d/0015.sun.env
```

特定のワークステーションに対して、この環境設定から追加または削除する必要がある場合は、このファイルを直接編集するか、最初に次の場所にコピーした後で編集できます。

```
/etc/dt/config/Xsession.d/0015.sun.env
```

追加の 1 つの例は、古い OpenWindows の仮想キーボードを起動して、OpenWindows DeskSet アプリケーションで使用することです。CDE デスクトップ上のほとんどのアプリケーション (および、ほとんどの Sun のユーザ) は、この仮想キーボード・ユーティリティ・プログラムを使用しないので、デスクトップ起動全体の性能を向上するために、デフォルトの起動シーケンスから除外されました。

オプションの OpenWindows 仮想キーボード (vkdb) の起動の詳細は、0015.sun.env ファイル中のコメントを参照してください。

Apple Macintosh アプリケーション環境でのフロッピーや CD メディアの使用

Apple Macintosh アプリケーション環境 (MAE) バージョン 1.0 をすでにインストールしていて、取り外し可能メディア (フロッピーディスクや CD-ROM) を OpenWindows のファイル・マネージャ・アプリケーションで使用する場合、/etc/rmmount.conf ファイルを編集しなければなりません。これによって、MAE は、ファイル・マネージャの取り外し可能メディアで正しく動作します。

/etc/rmmount.conf ファイルを変更するには

1. su と入力し、次にパスワードを入力して、**root** ユーザになります。
2. /etc ディレクトリに移動します。次のコマンドを入力して、**Return** キーを押します。

```
# cd /etc
```

3. エディタで、/etc/rmmount.conf ファイルを開きます。
4. #Actions の下にある次の行を、リストの最後に移動します。

```
action floppy action_macfs.so
```

たとえば、次のようになります。

```
# more rmmount.conf
   # @(#)rmmount.conf 1.2     92/09/23 SMI
   #
   # Removable Media Mounter configuration file.
```

```
#
# File system identification
ident hsfs ident_hsfs.so cdrom
ident ufs ident_ufs.so cdrom floppy
ident pcfs ident_pcfs.so floppy
ident macfs ident_macfs.so floppy
# Actions
action cdrom action_filemgr.so
action floppy action_filemgr.so
action floppy action_macfs.so
```

5. 保存して終了します。

   MAE をインストールすると、`rmmount.conf` ファイルに `macfs action` 行が追加されます。MAE のインストールを解除する場合、この行は削除されます。

   この変更によって、MAE で Macintosh のフロッピーを使用できるようになります。また、OpenWindows または CDE のファイル・マネージャ・アプリケーションでも取り外し可能メディアを使用できるようになります。しかし、MAE は、フォーマットしていないフロッピー、読めないフロッピー、および DOS フォーマットのフロッピーを認識できなくなります。このようなディスクはすべて、この設定手順に従って、OpenWindows のファイル・マネージャ・アプリケーションに渡されます。

   MAE で、フォーマットしていないフロッピー、読めないフロッピー、および DOS フォーマットのフロッピーを認識できるようにするには、`action floppy action action_macfs.so` 行を、`action floppy action_filemgr.so` 行の前に置かなければなりません。

第 7 章

ネットワークにおけるデスクトップの構成

デスクトップは、高度にネットワーク化された環境で十分動作するように設計されています。この章では、次の内容について説明します。

デスクトップのアーキテクチャにより、システム管理者はネットワーク全体にコンピューティング・リソースを分散させることができます。その中には次のものが含まれます。

- アプリケーション
- アプリケーションのデータ・ファイル
- デスクトップ・セッション・サービス (ログイン・マネージャやファイル・マネージャなどのデスクトップ・アプリケーション)
- ヘルプ・サービス。ヘルプのデータ・ファイルは中央のヘルプ・サーバに置くことができます。

デスクトップ・ネットワーキングの概要

オペレーティング・システムは、さまざまなネットワーキング・サービスを提供します。その中には、分散ファイル・システムとリモート実行も含まれます。X サーバは追加のネットワーキング機能を提供します。その中には、リモート・ディスプレイへのアクセスやセキュリティ・サービスも含まれます。

デスクトップは、これらのネットワーキング機能の最上部にユーザ・インタフェースを重ねます。このインタフェースとその基となるアーキテクチャの目的は、次のようなネットワーク・システムを構築することです。

- 簡単に使用できます。ネットワーク内でアプリケーションとデータの位置を気にすることなくアプリケーションを実行し、データ・ファイルにアクセスできます。
- 簡単に管理できます。デスクトップは、システムがリモート・データとアプリケーションを簡単に検出できるようアプリケーション統合ツールとネットワーク検索パスを提供します。さらに、デスクトップのファイル名のマッピング・プロセスは、サーバが多数含まれる複雑なネットワークの管理を簡単にします。
- 柔軟性があります。デスクトップの管理機能が共通ネットワーク環境に合うように設計されていると、デスクトップは他のカスタマイズされたネットワーク構成を多く取り入れることができます。

ネットワーク・デスクトップ・サービスの種類

ネットワークに接続されると、他のシステムに分散された、さまざまなコンピューティング・サービスにアクセスできるようになります。たとえば、次のようなサービスにアクセスできます。

- デスクトップ・セッションとそのアプリケーション (たとえば、ワークスペース・マネージャとファイル・マネージャ)
- 他のアプリケーション
- データ・ファイル

ネットワーク接続では、他の 1 つ以上のシステムにコンピューティング・サービスを提供するシステムを説明する用語として「サーバ」を使用します。システムがサーバからサービスを受ける場合は、そのサーバの「クライアント」と呼びます。

複雑なネットワークでは、システムはネットワーク全体の多数のシステム上にあるサービスを使用することもあります。さらに、システムは特殊なタイプのサーバ (たとえばセッション・サーバ) や、クライアント (たとえばアプリケーション・サーバのクライアント) として動作することもあります。

一般的なネットワーク環境

デスクトップ環境では、一般的なネットワーク構成に、次の主要コンポーネントのいくつかの組み合わせを含んでいます。

ディスプレイ — ここで X サーバを実行します。

ログイン/セッション・サーバ — ここでデスクトップ・アプリケーション (ログイン・マネージャやワークスペース・マネージャなど) を実行します。

アプリケーション・サーバ — ここで他のアプリケーションを実行します。

ファイル・サーバ — ここにアプリケーションが使用するデータが格納されています。

最も一般的なネットワーク構成の 1 つには、アプリケーション・サーバにアクセスするシステムがあります。図 7–1 は、アプリケーション・サーバを使用しているワークステーションを示します。X サーバとデスクトップ・セッションは、ワークステーション上で実行中です。

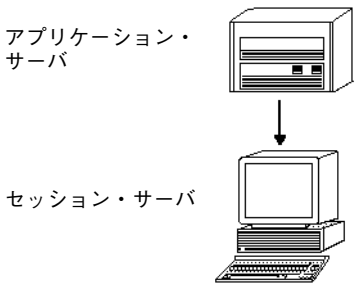


図 7–1　アプリケーションがデスクトップ・セッションにサービスを提供する

ネットワークはまた、ファイル・サーバを使用して大量のデータを保存します。このデータは、アプリケーション・サーバ上で実行中のアプリケーションや、デスクトップ・アプリケーションによって使用されることがあります (たとえば、ファイル・マネージャは [ファイル・マネージャ] ウィンドウでデータ・ファイルを表示するために、データ・ファイルにアクセスする必要があります)。

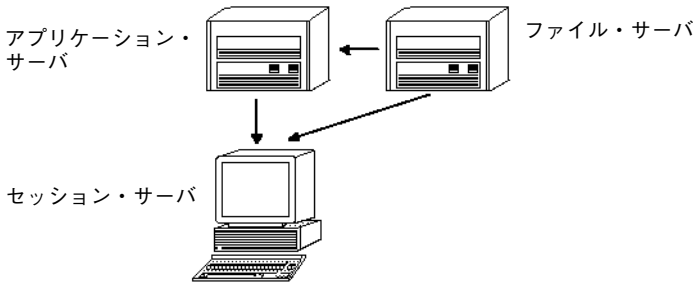


図 7–2　ファイル・サーバがアプリケーション・サーバとセッション・サーバにデータを提供する

X 端末は X サーバを実行し、別のシステムからデスクトップ・セッション・サービスを取得します。

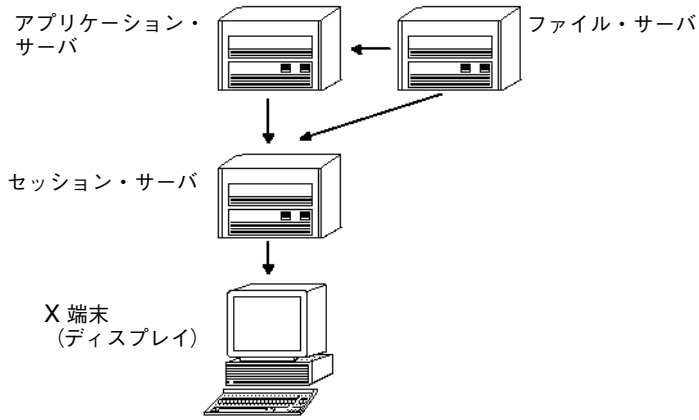


図 7–3　X 端末がセッション・サーバからセッション・サービスを取得する

他のネットワーキング環境

デスクトップには柔軟性があるので、もっと複雑なネットワーク構成をサポートできます。ファイル・サーバに加えて、アプリケーション・サーバで使用可能なさまざまなサービスも提供します。

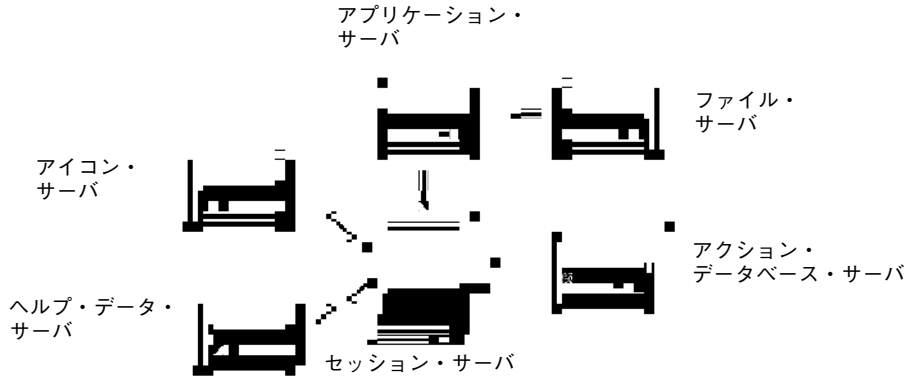


図 7–4　デスクトップ・アプリケーション・サーバが必要とするサービスを分散できる

まとめ — サーバの種類

ディスプレイ — X サーバを実行中のシステム

ログインとセッション・サーバ — デスクトップ・セッション (ログイン・マネージャ、セッション・マネージャ、ウィンドウ・マネージャ、ファイル・マネージャなど) を実行中のシステム

アプリケーション・サーバ — アプリケーションが実行されているシステム。「実行ホスト」とも呼ばれます。

ファイル・サーバ — アプリケーションのデータ・ファイルが格納されているシステム

ヘルプ・サーバ — ヘルプ・データ・ファイルが格納されているシステム

(アクション) データベース・サーバ — アクションとデータ型定義が入っているファイルが格納されているシステム

アイコン・サーバ — アイコン・ファイルが格納されているシステム

ネットワークには、パスワード・サーバ、メール・サーバ、ビデオ・サーバなどの追加のサーバが含まれている場合があります。

デスクトップ・ネットワーキングを構成するための一般的な手順

デスクトップ・ネットワーキングを構成するための一般的な手順としては、次の 3 つがあります。

1. 基本オペレーティング・システムのネットワーク・サービスを構成します。

   デスクトップが依存しているオペレーティング・システムによって提供されるネットワーク・サービスがあります。詳細は、118ページの「デスクトップ用の基本オペレーティング・システムのネットワーキング構成」を参照してください。

2. デスクトップ・ネットワーキングのソフトウェアとサービスをインストールし、設定します。

   設定中のクライアントやサーバのシステムの種類に関係なく、デスクトップが必要とするサービスがあります。詳細は、122ページの「デスクトップのクライアントとサーバの構成」を参照してください。

3. サーバまたはクライアントの特定の型を設定します。

   たとえば、アプリケーション・サーバを構成するには、ファイル・サーバを構成する場合とは異なる手順が必要です。詳細は、128ページの「アプリケーション・サービスの管理」を参照してください。

デスクトップ用の基本オペレーティング・システムのネットワーキング構成

デスクトップには、次の基本ネットワーキング構成が必要です。

- ユーザは、セッション・サーバ上と、デスクトップ・サービスをセッション・サーバに提供する各システム上にログイン・アカウントを持っていなければなりません。ユーザは、すべてのクライアントとサーバのシステムで同じユーザ ID とグループ ID を持っていなければなりません。
- システムは、セッションと他のアプリケーションによって使用されるデータが入っているリモート・ファイル・システムにアクセスできなければなりません。

- lp プリント・スプーラは、リモート・プリンタにアクセスできるように構成されていなければなりません。
- sendmail は、電子メール・サービス用に構成されていなければなりません。
- X 認証が設定されていなければなりません。

ユーザへのログイン・アカウントの提供

本節では、デスクトップ・ネットワーキングに必要なログイン・アカウントについて説明します。

ログイン・アカウントの提供

ユーザは、次のコンポーネントにログイン・アカウントを持っていなければなりません。

- デスクトップにサービスを提供しているすべてのシステム。この中には、アプリケーション・サーバ、ファイル・サーバ、およびネットワーク・プリンタを提供するシステムも含まれます。
- ユーザがアクセスできるすべてのセッション・サーバ。通常、セッション・サーバは X 端末で使用されます。

ユーザ ID とグループ ID の一貫性

UNIX ユーザは、ログイン名と数値ユーザ ID (UID) により識別されます。デスクトップ・ネットワークでは、ユーザはすべてのクライアントとサーバのシステム上に同じログイン名と UID を持っていなければなりません。

UNIX ユーザは、1 つ以上のログイン・グループにも割り当てられます。各グループはグループ名と数値グループ ID (GID) を持っています。デスクトップ・ネットワークでは、すべてのシステムは一貫したグループ名とグループ ID を使用しなければなりません。

詳細は、id(1) または id(1M) のマニュアル・ページを参照してください。

分散ファイル・システム・アクセスの構成

デスクトップは、システム間でファイルを共有するために NFS™ を使用します。共有ファイルが入っているネットワークのすべてのファイル・システムを識別し、確実に適切なシステムに正しくマウントしなければなりません。

通常は、次のリモート・ファイル・アクセスを提供しなければなりません。

- ユーザのホーム・ディレクトリは、すべてのデスクトップのクライアントとサーバのシステムによって共有されなければなりません。これは次の理由により必要です。
  - ホーム・ディレクトリには、リモート・システム上のアプリケーションによってアクセスしなければならないデータ・ファイルがあります。たとえば、データ・ファイルを使用するアプリケーションは、デフォルトのデータ・ファイルの位置としてホーム・ディレクトリを使用する場合が多いです。
  - ホーム・ディレクトリは、デフォルトの `dtspcd` 認証ディレクトリです。`dtspcd` の詳細は、126ページの「サブプロセス・コントロール・デーモンの構成」を参照してください。
- ホーム・ディレクトリにはないデータ・ファイルにアクセスする必要がある場合、これらのデータ・ファイルは、データ・ファイルで動作するデスクトップのクライアントとサーバのシステムによって共有されなければなりません。
- デスクトップのインストールディレクトリと構成ディレクトリ (`/usr/dt` と `/etc/dt`) は、アプリケーションのすべてが同じデスクトップ構成ファイルにアクセスするように、すべてのデスクトップのクライアントとサーバのシステムによって共有されなければなりません。

ネットワーク・ホーム・ディレクトリの提供

デスクトップ・ネットワークは、ネットワーク上のすべてのクライアントとサーバのシステム間で共有されている単一のホーム・ディレクトリを持っている場合に、最も効率的に動作します。

ネットワーク・ホーム・ディレクトリにより、ユーザは個人用のカスタマイズと構成を失うことなく、ネットワークで別のシステムを使用できます。これは、個人用のカスタマイズと、前のセッションを復元するのに必要な情報を、ホーム・ディレクトリのサブディレクトリに保存するからです。

次のものにも共通のホーム・ディレクトリが必要です。

- デフォルトの X 認証機構。詳細は、122ページの「X 認証の構成」を参照してください。
- デスクトップのサブプロセス・コントロール・デーモン。このデーモンは、リモート・アプリケーションの起動に含まれますが、ユーザのホーム・ディレクトリに書き込めなければなりません。

ファイル名の一貫性

同じ名前を使用して、すべてのシステムからデータ・ファイルにアクセスできるようにネットワークを構成しなければなりません。これは、「ファイル名の一貫性」を提供するものとして知られ、適切なシンボリック・リンクを作成することにより通常は達成されます。たとえば、ディレクトリの実際のマウント位置へのシンボリック・リンクを作成することにより、各ユーザのホーム・ディレクトリが /users/*login_name* として使用可能になるように各システムを構成できます。

リモート・プリンタへのアクセスの構成

デスクトップは、ローカル・プリンタまたはリモート・プリンタにアクセスするために lp プリント・スプーラを使用します。lp スプーラの構成の詳細は、lpadmin(1M) のマニュアル・ページを参照してください。

デスクトップのグラフィカル・インタフェースを使用して印刷する前に、lp コマンドを使用してすべてのプリンタに正しく印刷できることをテストしてください。

必ず一貫したプリンタ・デバイス名を使用してください。たとえば、直接接続されているシステム上で特定のプリンタが Postscript1 とされている場合、そのプリンタにリモート・アクセスしている他のすべてのシステムも Postscript1 という名前を使用してください。

電子メールの構成

デスクトップのメール・プログラムは、システム間でメールを配信するために sendmail を使用します。電子メールの接続性の構成方法の詳細は、sendmail(1M) のマニュアル・ページを参照してください。

デスクトップからメールを送信または受信する前に、mailx コマンドを使用してメールを正しく送受信できるかテストしてください。

X 認証の構成

デスクトップは、ローカル・ディスプレイにアクセスするためにリモート・アプリケーション (X クライアント) に認証を与えるのに、デフォルトの X 機構を使用します。X 機構を構成するのに最も簡単な方法は、各ユーザに対してネットワーク・ホーム・ディレクトリを提供することです。これにより、次の要件が確実に満たされます。

- ユーザは、*HomeDirectory*/.Xauthority ファイルへの書き込み権と読み取り権を持っていなければなりません。
- アプリケーション・サーバの .Xauthority ファイルには、アプリケーションが実行されるディスプレイの「マジック・クッキー」が入っていなければなりません。

  詳細は、X(1) または xauth(1) のマニュアル・ページを参照してください。

デスクトップのクライアントとサーバの構成

この節では、デスクトップに固有のネットワーク構成要件について説明します。これらの機能は、基本オペレーティング・システムではなくデスクトップによって提供されます。

この節は、次の 2 つの部分に分かれます。

- ログイン・サービスとセッション・サービスの構成
- アプリケーションとそのデータが必要とするサービスの構成。これには、アプリケーション、データベース、アイコン、ヘルプ・サーバとそのクライアントが含まれます。

ログイン・サービスとセッション・サービスの構成

ログインまたはセッション・サーバは、ディスプレイと X サーバにデスクトップ・サービス (ログイン・マネージャ、セッション・マネージャ、ファイル・マネージャ、ウィンドウ・マネージャなど) を提供するシステムです。

通常、セッション・サーバはサービスを X 端末に提供します。しかし、ネットワーク構成は、X 端末とワークステーションの両方によってアクセスされる 1 つ以上のサーバに、セッション・サービスを集中するように設定できます。

ログイン・マネージャは、ログイン・サービスを他のディスプレイに提供するデスクトップ・コンポーネントです。ユーザがログインすると、セッション・マネージャはユーザに対して起動されます。

ログインまたはセッション・サーバと X 端末の構成の詳細は、6ページの「ネットワーク・ディスプレイでのログイン画面の表示」を参照してください。

他のアプリケーション関連サービスの構成

この節では、デスクトップに共通のネットワーキングに必要な条件について説明します。

- アプリケーション・サーバ
- データベース・サーバ
- アイコン・サーバ
- ヘルプ・サーバ

デスクトップのクライアントとサーバを構成するには

1. **デスクトップが必要とするオペレーティング・システム・ネットワーク構成を提供します。**

   詳細は、118ページの「デスクトップ用の基本オペレーティング・システムのネットワーキング構成」を参照してください。

2. **デスクトップまたはファイルの最小セットをインストールします。**

   次のいずれかをインストールしなければなりません。

   - **共通デスクトップ環境の実行時のファイル・セット全体**
   - **CDE-MIN および CDE-TT のファイル・セット**

   注 - インストールとファイル・セットは、ベンダにより異なる場合があります。

   ---

3. **ToolTalk** ファイル名データベース・サーバ・デーモン `rpc.ttdbserver` **用にシステムを構成します。**

   デスクトップをインストールすると自動的に実行されます。詳細は、127ページの「ToolTalk データベース・サーバの構成」を参照してください。

4. サブプロセス・コントロール・デーモン (dtspcd) をインストールし構成します。

   デスクトップをインストールすると自動的に実行されます。詳細は、126ページの「サブプロセス・コントロール・デーモンの構成」を参照してください。

5. 必要なリモート・データをすべてマウントします。

   データを使用しているアプリケーションが実行中であるシステム以外のシステムにデータがある場合、データは「リモート」だと見なされます。

   たとえば、次のような場合があります。

   - アプリケーションがファイル・サーバにあるデータを使用する場合は、それらのファイルをマウントしなければなりません。
   - ファイル・マネージャのアイコンがアイコン・サーバにある場合、セッション・サーバはそれらのファイルをマウントしなければなりません。
   - ネットワークがデスクトップ・ヘルプ・ファイルのためにヘルプ・サーバを使用する場合、セッション・サーバとすべてのアプリケーション・サーバは、ヘルプ・データをマウントしなければなりません。

   マウント・ポイントの詳細は、124ページの「リモート・ファイル・システムのマウント・ポイントの構成」を参照してください。

リモート・ファイル・システムのマウント・ポイントの構成

デスクトップは 1 つのシステムから別のシステムにファイル名を渡す場合、そのファイル名を宛先システムにとって意味のある名前に変換、つまり「マップ」しなければなりません。このマッピングが必要なのは、ファイルが別のシステムの別の位置にマウントされ、そのため別の名前を使用してアクセスしなければならない場合があるためです。たとえば、sysA の /projects/big ファイルは、sysB の /net/sysA/projects/big としてアクセスされる可能性があります。

ファイル名マッピングのための要件

このファイル名マッピングを正しく実行するためには、次の条件のいずれか 1 つが True でなければなりません。

- mount コマンドをファイル・システムを静的にマウントするために使用する。これらの静的マウントは、通常、/etc/checklist、/etc/mnttab、/etc/filesystems などのファイルで構成されます。

システム間で正しく動作するファイル名マッピングの場合、ファイル・システムのマウントは一貫したホスト名を使用しなければなりません。ホストがいくつかの名前で認識される場合 (たとえば、別名または異なる名前で認識される 2 つ以上の LAN アドレスを持っている場合)、すべてのマウントに対して同じ名前と名前形式を使用しなければなりません。

- オートマウンタを、デフォルトの /net マウント・ポイントとしてファイル・システムをマウントするのに使用する。
- オートマウンタを、/net 以外の位置にファイル・システムをマウントするのに使用し、DTMOUNTPOINT 環境変数をマウント・ポイントを示すために設定する。詳細は、125ページの「DTMOUNTPOINT の値の設定」を参照してください。

オートマウンタの詳細は、automount(1M) のマニュアル・ページを参照してください。

DTMOUNTPOINT の値の設定

次の両方の条件が True の場合、DTMOUNTPOINT 環境変数を設定しなければなりません。

- オートマウンタを、ファイル・システムをマウントするのに使用する。
- リモート・ファイル・システムを /net 以外の位置にマウントする。

DTMOUNTPOINT は、次のようなプロセスに対して設定する必要があります。

- ワークスペース・マネージャ (dtwm) とファイル・マネージャ (dtfile) などにログインするときに自動的に起動されるユーザのデスクトップ・プロセス
- inetd などの機構によって起動される rpc.ttdbserver、dtspcd などのシステム・プロセス
- ローカル・システムまたはリモート・システム上のデスクトップによって起動されるアプリケーション
- シェル・コマンド行からユーザによって起動されるアプリケーション

これらのプロセスのすべてに対して DTMOUNTPOINT を設定するには、次のようにします。

1. /etc/inetd.conf ファイルを次のように編集します。
   a. dtspcd エントリを見つけ、次の行を追加します。

      -mount_point *mount_point*

b. rpc.ttdbserver エントリを見つけ、次の行を追加します。

```
-m mount_point
```

たとえば、オートマウンタが /nfs のマウント・ポイントで使用中の場合、/etc/inetd.conf のエントリは次のようになります。

```
dtspc stream tcp nowait root /usr/dt/bin/dtspcd \
   /usr/dt/bin/dtspcd -mount_point /nfs
   rpc stream tcp wait root /usr/dt/bin/rpc.ttdbserver \
   100083 1 rpc.ttdbserver -m /nfs
```

2. /etc/inetd.conf を再読み込みするシステム上の手続きを実行します。詳細は、inetd(1M) のマニュアル・ページを参照してください。

3. **DTMOUNTPOINT** の値がユーザのログインによって継承されるように設定します。

これは、/etc/dt/config/Xsession.d に変数を設定することによって実行できます。環境変数の設定の詳細は、33ページの「環境変数を設定するには」を参照してください。

サブプロセス・コントロール・デーモンの構成

デスクトップ・サブプロセス・コントロール (SPC) サービスは、クライアント/サーバのコマンド実行を提供します。

デスクトップ・サブプロセス・コントロール・デーモン (dtspcd) は、リモート・アプリケーションを起動するためにデスクトップによって使用されます。このデーモンは、コマンドを実行するためにリモート・クライアントから要求を受け取る inet デーモンです。inet デーモンの構成方法の詳細は、inetd.conf(1M) のマニュアル・ページを参照してください。

デスクトップ・アクション呼び出しライブラリは、リモート・アクションを呼び出すために SPC サービスを使用します。

dtspcd を構成するには

♦ dtspc が /etc/services と /etc/inetd.conf の両方に適切に登録されているか確認します。

詳細は、dtspcd(1M) のマニュアル・ページを参照してください。

SPC セキュリティ

サブプロセス・コントロール・サービスに対する認証は、ファイル・システム認証に基づきます。dtspcd は、すべての SPC クライアント・システムによってもマウントされる「認証ディレクトリ」にアクセスできなければなりません。

デフォルトでは、dtspcd 認証ディレクトリがユーザのホーム・ディレクトリです。しかし、/etc/inetd.conf ディレクトリに –auth_dir オプションを設定することにより、別の位置を使用するように dtspcd を構成できます。詳細は、dtspcd(1M) のマニュアル・ページを参照してください。

SPC 認証はファイル・システム認証に基づいているので、SPC サービスは分散ファイル・システムと同じ程度に安全です。分散ファイル・システムを信頼していないネットワーク上のデスクトップを使用している場合、dtspcd を使用できないようにするには、/etc/services の dtspc エントリをコメントにしてください。

リモート実行に対する環境変数の構成

リモート・システムでアプリケーションを起動するためにデスクトップがアクションを使用する場合、ユーザの環境変数はリモート・システムにコピーされ、アプリケーションの環境に配置されます。

デフォルトでは、環境変数のいくつかはリモート・システムにコピーされる前に変更されます。変数をアプリケーションの環境に位置付ける前に、追加のアプリケーション環境変数の処理を実行するように、アクション呼び出しコンポーネントとデスクトップのサブプロセス・コントロール・サービスの両方を構成できます。

デフォルトの構成とその変更方法の詳細は、dtactionfile(4) と dtspcdenv(4) のマニュアル・ページを参照してください。

ToolTalk データベース・サーバの構成

ToolTalk の 1 つのコンポーネントは、ToolTalk データベース・サーバ /usr/dt/bin/rpc.ttdbserver です。

ToolTalk データベース・サーバは、ToolTalk メッセージ・サービスによってファイル名マッピングのために使用されます。このサーバは、デスクトップがインストールされ、追加構成が必要ない場合、通常は /etc/inetd.conf に登録されます。

ToolTalk データベース・サーバとその構成オプションの詳細は、rpc.ttdbserver(1M) のマニュアル・ページを参照してください。

ToolTalk メッセージ・サーバの構成

ToolTalk メッセージ・サーバは ttsession です。デフォルトでは、このサーバには構成は必要ありません。ログイン中、このサーバは Xsession スクリプトによって起動されます。

ToolTalk メッセージ・サーバとその構成オプションの詳細は、ttsession(1) のマニュアル・ページを参照してください。

カレンダ・デーモンの構成

カレンダ・アプリケーションの 1 つのコンポーネントは、カレンダ・デーモン rpc.cmsd です。デスクトップがインストールされ、追加の構成が必要ない場合は、通常は /etc/inetd.conf に登録されます。

カレンダ・デーモンとその構成オプションの詳細は、rpc.cmsd(1) のマニュアル・ページを参照してください。

アプリケーション・サービスの管理

この節では、次のコンポーネントの特定の構成要件について説明します。

- アプリケーション・サーバとそのクライアント
- 特定のサービスを提供するデスクトップ・サーバ— データベース・サーバ、アイコン・サーバ、ヘルプ・サーバ

また、ネットワーク・アプリケーションの次の 2 つの特殊構成に対するネットワーキング要件についても説明します。

- リモート実行ホスト
- ファイル・システムをマウントして実行中のアプリケーション

検索パスの環境変数

アクション、データ型データベース、ヘルプ・ファイル、アイコン・ファイルなどのアプリケーション・デスクトップ構成ファイルを見つけるのに使用する検索パスを指定するために、デスクトップは環境変数セットを使用します。

検索パスの環境変数の使用方法の詳細は、第 9 章または dtenvvar(5) のマニュアル・ページを参照してください。

アプリケーション・サーバとそのクライアントの構成

標準アプリケーション・サーバ構成において、アプリケーション・サーバには、アプリケーションに関連付けられた次のようなバイナリ・ファイルと構成ファイルが入っています。

- アプリケーション実行可能ファイル
- app-defaults、メッセージ・カタログ、そのアプリケーションの共有ライブラリなどの標準アプリケーション構成ファイル
- デスクトップ構成ファイル
  - アクションとデータ型定義ファイル
  - アイコン・イメージ・ファイル
  - デスクトップ・ヘルプ・データ・ファイル

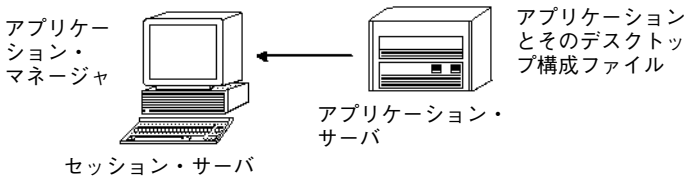


図 7–5　標準アプリケーション・サーバ構成

アプリケーション・サーバを構成するには

1. **デスクトップが必要とするオペレーティング・システムのネットワーク構成を提供します。**

   詳細は、118ページの「デスクトップ用の基本オペレーティング・システムのネットワーキング構成」を参照してください。

2. **サーバに対して必要な一般デスクトップ構成を提供します。**

   詳細は、123ページの「デスクトップのクライアントとサーバを構成するには」を参照してください。

3. **アプリケーションをインストールします。**

4. アプリケーションが自動的にそれ自身を登録しない場合は、登録手続きを実行しなければなりません。

   詳細は、第 5 章を参照してください。

アプリケーション・サーバのクライアントを構成するには

1. デスクトップが必要とするオペレーティング・システムのネットワーク構成を提供します。

   詳細は、118ページの「デスクトップ用の基本オペレーティング・システムのネットワーキング構成」を参照してください。

2. クライアントに対して必要な一般デスクトップ構成を提供します。

   詳細は、123ページの「デスクトップのクライアントとサーバを構成するには」を参照してください。

3. システム共通か個人用かによって、アプリケーション・サーバをアプリケーション検索パスに追加します。

   システム共通 — `DTSPSYSAPPHOSTS` 変数を `/etc/dt/config/Xsession.d/0010.dtpaths` に設定します。

   個人用 — `DTSPUSERAPPHOSTS` 変数を *HomeDirectory*`/.dtprofile` に設定します。

たとえば、`/etc/dt/config/Xsession.d/0010.dtpaths` にある次の行は、`SysAAA` と `SysBBB` というホスト名が付いているシステムをアプリケーション検索パスに追加します。

```
export DTSPSYSAPPHOSTS=SysAAA:,SysBBB:
```

アプリケーション検索パスの設定の詳細は、次の節を参照してください。

- 147ページの「アプリケーション検索パス」
- 145ページの「検索パスの値の設定」

データベース、アイコン、およびヘルプ・サービスの構成

通常は、アクションと、アプリケーションに関連付けられたデータ型定義、アイコン、ヘルプ・ファイルは、アプリケーションとして同じシステムにインストールされます。

たとえば、ヘルプのデータ・ファイルの一般的な構成について考えてみます。

- ファイル・マネージャのヘルプ・ファイルは通常、セッション・サーバにあります。ヘルプ検索パスはセッション・サーバで適切な位置を自動的に検索するため、デスクトップはそれらのファイルを見つけます。
- 他のアプリケーションのヘルプ・ファイルは通常、アプリケーションとして同じアプリケーション・サーバにあります。アプリケーション検索パスを変更するとヘルプ検索パスを自動的に変更するので、セッション・サーバはそれらのファイルを見つけます。

ネットワークのどこかにデータベース (アクションとデータ型)、ヘルプ、またはアイコン・データを置きたい場合があるでしょう。たとえば、ネットワークが複数のセッション・サーバを使用する場合、デスクトップ・アプリケーション (ファイル・マネージャ、スタイル・マネージャなど) のすべてのヘルプ・データ・ファイルが格納されているヘルプ・サーバを作成することをお勧めします。こうすると、ヘルプ・ファイルを各セッション・サーバで複製する必要がないので、ディスク・スペースを節約できます。

データベース・サーバ、ヘルプ・サーバ、またはアイコン・サーバを作成するには

1. **デスクトップが必要とするオペレーティング・システム・ネットワーク構成を提供します。**

   詳細は、118ページの「デスクトップ用の基本オペレーティング・システムのネットワーキング構成」**を参照してください。**

2. **クライアントに対して必要な一般デスクトップ構成を提供します。**

   詳細は、123ページの「デスクトップのクライアントとサーバを構成するには」**を参照してください。**

3. **データベース・ファイル、ヘルプ・ファイル、またはアイコン・ファイルをインストールします。**

ファイルは、システムのどこにでも置くことができます。しかし、システムがアプリケーション・サーバを指定すると自動的に検索されるディレクトリがあるので、次の位置を使用するとより簡単になります。

- データベース・ファイル: /etc/dt/appconfig/types/*language*
- ヘルプ・ファイル: /etc/dt/appconfig/help/*language*
- アイコン・ファイル: /etc/dt/appconfig/icons/*language*

データベース・サーバを設定する場合、コマンド (EXEC_STRING) を実行する位置を指定するためにアクションに書き込む必要があります。詳細は、133ページの「リモート実行ホストの指定」を参照してください。

データベース・サーバ、アイコン・サーバ、またはヘルプ・サーバを見つけるためにセッション・サーバを構成するには

1. デスクトップが必要とするオペレーティング・サーバ・ネットワーク構成を提供します。

   詳細は、118ページの「デスクトップ用の基本オペレーティング・システムのネットワーキング構成」を参照してください。

2. クライアントに対して必要な一般デスクトップ構成を提供します。

   詳細は、123ページの「デスクトップのクライアントとサーバを構成するには」を参照してください。

3. データベース・サーバ、アイコン・サーバ、またはヘルプ・サーバを適切な検索パスに追加します。

   - 131ページの「データベース・サーバ、ヘルプ・サーバ、またはアイコン・サーバを作成するには」の手順 3 で指定された位置にデータ・ファイルを格納した場合、アプリケーション検索パスを変更できます。
   - 他の位置にデータ・ファイルを格納した場合、特定の検索パスを変更しなければなりません。

   たとえば、ヘルプ・ファイルをシステム SysCCC にあるディレクトリ /etc/dt/help に格納した場合、次の行を /etc/dt/config/Xsession.d/0010.dtpaths に追加します。

   ```
   export DTSPSYSHELP=/net/SysCCC/etc/dt/help
   ```

検索パスの設定の詳細は、次の節を参照してください。

- 150ページの「データベース (アクションとデータ型) 検索パス」
- 152ページの「アイコン検索パス」
- 154ページの「ヘルプ検索パス」
- 145ページの「検索パスの値の設定」

特殊ネットワーク・アプリケーション構成

この節では、次のようなアプリケーションを実行するためにシステムを構成する方法について説明します。

- アクションが入っているシステム、つまりリモート実行ホスト以外の場所にあるアプリケーション
- ファイル・システム・マウントに対してローカルにあるアプリケーション

リモート実行ホストの指定

典型的なアプリケーション・サーバ構成では、アクション定義はアプリケーション実行可能ファイルと同じシステムにあります。しかし、アクションは他のシステムにあるコマンドを実行するために書き込むことができます。この構成では、アプリケーションが入っているシステムは「実行ホスト」と呼びます。

アクション定義は、セッション・サーバまたはセッション・サーバにアクションとデータ型のサービスを提供するシステムに置くことができます。このシステムを「データベース・サーバ」または「データベース・ホスト」と呼びます。

アクション定義は、コマンド (EXEC_STRING) を実行する位置を指定するために EXEC_HOST フィールドを使用します。たとえば次のアクション定義は、ホスト名が SysDDD であるシステムで xload クライアントが実行されるように指定します。

```
ACTION XloadSysDDD
 {
  TYPE   COMMAND
  EXEC_HOST  SysDDD
  EXEC_STRING  /usr/bin/X11/xload -label SysDDD
 }
```

EXEC_HOST フィールドが 2 つ以上のホスト名を指定する場合、デスクトップはアクションを実行できるホストを見つけるまで順番に各ホストで EXEC_STRING を実行しようとします。たとえば、次の EXEC_HOST フィールドはアクションが最初に

SysDDD、失敗した場合は SysEEE で EXEC_STRING を実行するように指定しています。

```
EXEC_HOST     SysDDD,SYSEEE
```

EXEC_HOST フィールドがアクション用に設定されていない場合、デフォルト値は %DatabaseHost% になります。%DatabaseHost% の値はデータ検索パスから取得されます。

たとえば、データベース検索パスは /etc/dt/config/Xsession.d/0010.dtpaths に次の行を追加することによって変更されたとします。

```
DTSPSYSDATABASEHOSTS=SysAAA:,/net/SysBBB/etc/dt/appconfig/types/C
```

SysAAA は、ホスト修飾子構文を使用して指定されます。つまり SysAAA: になります。検索パスのこの要素を使用して見つけられるアクション定義は、データベース・ホストを SysAAA に設定します。しかし、検索パスの /net/SysBBB... 部分を使用して見つけられるアクションは、構文にはホスト修飾子が含まれていないのでデータベース・ホストにローカル・システムを設定します。

リモート実行ホストを構成するには

1. デスクトップが必要とするオペレーティング・システムのネットワーク構成を提供します。

   詳細は、118ページの「デスクトップ用の基本オペレーティング・システムのネットワーキング構成」を参照してください。

2. サーバに対して必要な一般デスクトップ構成を提供します。

   詳細は、123ページの「デスクトップのクライアントとサーバを構成するには」を参照してください。

3. アプリケーションをローカル実行のために適切な方法で確実にインストールし構成します。

アクション定義が入っているシステムを構成するには

1. デスクトップが必要とするオペレーティング・システム・ネットワーク構成を提供します。

詳細は、118ページの「デスクトップ用の基本オペレーティング・システムのネットワーキング構成」を参照してください。

2. サーバに対して必要な一般デスクトップ構成を提供します。

   詳細は、123ページの「デスクトップのクライアントとサーバを構成するには」を参照してください。

3. アクション定義とアプリケーション・グループを作成しインストールします。

   詳細は、210ページの「リモート・システムでアプリケーションを実行するアクションの作成」と 52ページの「一般アプリケーション・グループの作成と管理」を参照してください。

セッション・サーバを構成するには

1. デスクトップが必要とするオペレーティング・システム・ネットワーク構成を提供します。

   詳細は、118ページの「デスクトップ用の基本オペレーティング・システムのネットワーキング構成」を参照してください。

2. クライアントに対して必要な一般デスクトップ構成を提供します。

   詳細は、123ページの「デスクトップのクライアントとサーバを構成するには」を参照してください。

3. データベース・ホストを組み込むためにアクション検索パスを変更します。

   詳細は、150ページの「データベース (アクションとデータ型) 検索パス」を参照してください。

4. 実行ホストを組み込むためにアプリケーション検索パスを変更します。

   詳細は、147ページの「アプリケーション検索パス」を参照してください。

アプリケーションをローカルに実行

標準アプリケーション・サーバ構成は、アプリケーション・サーバでアプリケーションを実行します。リモート・システムにアプリケーションをインストールし、セッション・サーバでローカルに実行する方が望ましいこともあります。

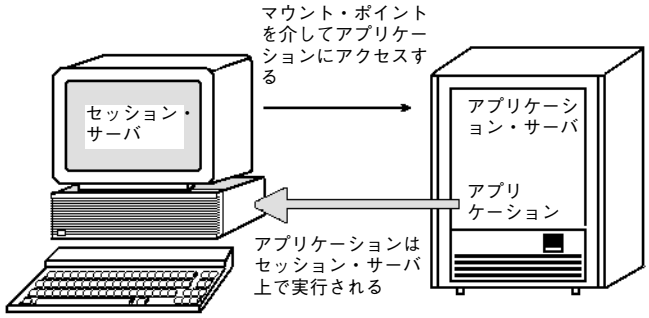


図 7–6　マウント・ポイントでの実行

アプリケーション・サーバを構成するには

特別な構成は必要ありません。

セッション・サーバを構成するには

- アプリケーション検索パスを変更します。アプリケーションへのローカル絶対パスを使用します。

たとえば、sysAAA に登録されたアプリケーションを見つけるには、次の変数定義を使用します。

```
DTSPSYSAPPHOSTS=/net/SysAAA/etc/dt/appconfig/appmanager/C
```

セッション・サーバは、app-defaults、メッセージ・カタログ、共有ライブラリなどのアプリケーションの構成ファイルにアクセスできなければなりません。

第 8 章

デスクトップからの印刷の構成と管理

デスクトップ・ユーザがファイルを印刷するには、さまざまな方法があります。主に、デスクトップからの印刷とアプリケーションからの印刷の 2 種類に分類されます。

デスクトップから印刷するには、次の方法があります。

- [ファイルマネージャ] でファイルを選択し、[選択] メニューまたはアイコンのポップアップ・メニューから [印刷] を選択する
- [ファイルマネージャ] からフロントパネルの [プリンタ] コントロールまたはサブパネルの [個人プリンタ] へファイルをドラッグする
- [ファイルマネージャ] から [印刷マネージャ] メイン・ウィンドウにあるプリンタにファイルをドラッグする

アプリケーションから印刷するには、[印刷] コマンドを使用します。このコマンドは通常、アプリケーションのウィンドウ内のメニューまたは他のコントロールでアクセスします。

この章では、次の内容について説明します。

プリンタの追加と削除

この節では、デスクトップからのプリンタの追加と削除の手順について説明します。

▼ プリンタをデスクトップに追加するには

1. プリンタをシステムの構成に追加します。

   お使いのオペレーティング・システムのシステム管理マニュアルの指示に従ってください。

2. 次のコマンドを実行します。

   ```
   env LANG=language /usr/dt/bin/dtprintinfo -populate
   ```

3. 印刷マネージャを再起動するか、アプリケーション・マネージャにある **[**デスクトップツール**]** アプリケーション・グループから **[**アクションの再読込み**]** をダブルクリックします。プリンタが表示されるか確認します。

4. メールをユーザに送信して、印刷マネージャを再起動するか、**[**アクションの再読込み**]** を実行するように通知します。

印刷マネージャは、起動されるたびにシステム・プリンタ構成リストを読み込みます。印刷マネージャが新規プリンタを検出すると、新しいデスクトップのプリンタ・アクションとそのプリンタのアイコンを自動的に作成します。プリンタをデスクトップに表示させること以外に、必要な作業はありません。

▼ プリンタをデスクトップから削除するには

1. システムの構成からプリンタを削除します。

   お使いのオペレーティング・システムのシステム管理マニュアルの指示に従ってください。

2. 印刷マネージャを再起動するか、アプリケーション・マネージャにある **[**デスクトップツール**]** アプリケーション・グループから **[**アクションの再読込み**]** をダブルクリックします。プリンタが削除されたか確認します。

3. メールをユーザに送信して、印刷マネージャを再起動するか、**[**アクションの再読込み**]** を実行するよう通知します。

印刷マネージャは、起動されるたびにシステム・プリンタ構成リストを読み込みます。プリンタがリストから削除されたことを確認すると、印刷マネージャとファイル・マネージャからプリンタのアクションとアイコンを自動的に削除します。必要な作業は、プリンタをデスクトップから削除することだけです。

注 - 印刷マネージャは、フロントパネルからプリンタを削除できません。したがって、構成からプリンタを削除する場合は、必ずシステム上のすべてのユーザにメールを送信して、削除したプリンタのアイコンをフロントパネルから削除するように通知してください。

ジョブ更新間隔の変更

印刷マネージャに表示された情報を更新する回数を変更するには、ジョブ更新間隔を変更します。デフォルトでは、印刷マネージャはプリント・ジョブの情報について、30 秒ごとにプリンタに問い合わせます。[オプションの設定] ダイアログ・ボックス ([表示] メニューから [オプションの設定] を選択すると表示されます) にある [更新] の間隔スライダを使用して、印刷マネージャがプリンタに問い合わせる間隔を変更できます。

プリンタ・アイコンのイメージ

プリンタを追加すると、そのプリンタにデフォルトのプリンタ・アイコンが自動的に割り当てられます。別のアイコンを使用する場合は、アイコン・ファイルを `/etc/dt/appconfig/icons/`*language*、またはアイコン検索パスに従って他のディレクトリに格納します。このアイコンを選択して、プリンタのデフォルトのアイコンを置き換えることができます。

必ずアイコンの完全なセット (大、中、または極小) を作成してください。そうしないと、印刷マネージャのアイコン・セレクタには表示されません。

アイコン検索パスの詳細は、152ページの「アイコン検索パス」を参照してください。

アイコン・ファイル名とサイズ

アイコンのファイル名の命名規則は次のとおりです。

base_name.size.*type*

size — l (大)、m (中)、t (極小) があります。アイコン・サイズの詳細は、237ページの「アイコン・サイズ規則」を参照してください。

type — pm (カラー・ピックスマップ) または bm (ビットマップ)

たとえば、カラー・プリンタのピックスマップの中型のアイコン・ファイル名は ColorPrinter.m.pm、極小型サイズのアイコン・ファイル名は ColorPrinter.t.pm になります。

アイコン作成の詳細は、第 14 章を参照してください。

▼ アイコン、プリンタ・ラベル、またはプリンタの記述をグローバルに変更するには

プリンタを追加した場合は、ユーザが印刷マネージャを使用してグローバル・プリンタ属性を変更する前に、すぐにその属性を変更してください。ユーザが印刷マネージャを使用してプリンタ属性を変更してしまうと、ユーザは変更内容を見ることはできません。

アイコン、プリンタ・ラベル、または説明などの情報を /etc/dt/appconfig/types/*language*/*printer_queue_name*.dt ファイルで編集します。

1. ICON フィールドで、***basename*** を新規アイコンのベース名に更新します。
2. LABEL フィールドで、***labelname*** をプリンタの新規ラベルに更新します。
3. DESCRIPTION フィールドでテキストを変更します。

   このフィールドに、プリンタの位置、プリンタの種類、およびプリンタの接続先を設定すると便利です。2 行以上追加するには、次の例のように、行の最後に \ または ¥ を入れます。

```
DESCRIPTION    This is a PostScript Printer in Building 1 \
      Room 123. Call 555-5555 for problems.
```

デフォルト・プリンタの構成

次の操作により、デフォルト・プリンタにアクセスします。

- フロントパネルの [プリンタ] コントロールでオブジェクトをドロップする
- ファイル・マネージャにあるオブジェクトを選択するか、[選択] メニューまたはアイコンのポップアップ・メニューから [印刷] を選択する
- デフォルト・プリンタを使用するアプリケーションから印刷する

▼ デフォルトの印刷の宛先を変更するには

すべてのユーザのデフォルト・プリンタを変更するには、次のようにします。

1. ファイル `/etc/dt/config/Xsession.d/0010.dtpaths` を開きます。

   `/etc/dt/config/Xsession.d/0010.dtpaths` が存在しない場合は、`/usr/dt/config/Xsession.d/0010.dtpaths` からコピーします。

2. `LPDEST=`***printer*** 行で、***printer*** をデフォルトの印刷の新しい宛先に変更します。

   `LPDEST=`*printer* 行が存在しない場合は追加します。*printer* は、デフォルト・プリンタにするプリンタ名です。

3. ユーザはログアウトしてからログインし直します。

1 人のユーザのデフォルト・プリンタを変更するには、そのユーザは次の作業を実行します。

- サブパネルの **[個人プリンタ]** からフロントパネルに別のプリンタをコピーします。

デフォルト・プリンタとして別のプリンタを指定するには、次のようにします。

1. ホーム・フォルダに入って、ファイル `.dtprofile` を開きます。

2. `LPDEST` 環境変数の値を設定する行を追加または編集します。

   ```
   LPDEST=printer_device; export LPDEST
   ```

   `csh` を使用している場合の構文は次のとおりです。

```
setenv LPDEST printer_device
```

たとえば、次の行はデフォルト・プリンタをデバイス名が laser3d であるプリンタに変更します。

```
LPDEST=laser3d; export LPDEST
```

csh を使用している場合の構文は次のとおりです。

```
setenv LPDEST laser3d
```

印刷の概念

プリンタ・コントロールにファイルをドロップすることにより印刷要求が起動されると、システムは次の作業を実行します。

1. システムは、ドロップされたオブジェクトの定義をデータ型データベースで検索します。

2. データ型用の一意の印刷アクション (印刷アクションの ARG_TYPE フィールドを使用して指定されます) がある場合は、そのアクション使用します。アクションがない場合は、デフォルトのプリント・アクション (dtlp) を使用します。たとえば、**PostScript**™ ファイルの場合、システムは **PostScript** ファイル用の [印刷] アクションを使用します (このアクションは /usr/dt/appconfig/types/*language*/dt.dt で定義されます)。このデータ型用のアクション作成ツールを使用した場合、入力した印刷コマンドは、このデータ型でファイルを印刷するために使用される固有の印刷アクションになります。

3. ファイルは、通常の **UNIX** lp 印刷サブシステムを使用してプリンタに配信されます。

第 9 章

デスクトップ検索パス

デスクトップは、アプリケーションとそれに関連付けられたデスクトップ・ファイルを検出するために検索パスを使用します。

この章では、次の内容について説明します。

デスクトップは、表 9–1 にある 4 つの検索パスを提供します。

表 9–1 デスクトップ検索パス

検索パス	説明
アプリケーション	アプリケーションを検出するのに使用します。アプリケーション・マネージャは、アプリケーション検索パスを使用して、ユーザのログイン時に動的にトップレベルを生成します。
データベース	アクションおよびデータ型の定義ファイル (`*.dt` ファイル) とフロントパネル・ファイル (`*.fp` ファイル) の追加位置を指定するのに使用します。

表 9–1　デスクトップ検索パス　続く

検索パス	説明
アイコン	アイコンの追加位置を指定するのに使用します。
ヘルプ・データ	デスクトップ・ヘルプ・データの追加位置を指定するのに使用します。

検索パスには、ローカル・ディレクトリとリモート・ディレクトリの両方を指定できます。したがって、デスクトップのネットワーク・アーキテクチャにおいて、検索パスは重要な役割を果たします。たとえば、アプリケーション・サーバ上でシステムがアプリケーションを見つけられるのは、そのアプリケーション・サーバがアプリケーション検索パスにリストされているからです。

検索パスにリモート位置を指定する場合は、その位置へのリモート・ファイル・アクセスを構成しなければなりません。詳細は、120ページの「分散ファイル・システム・アクセスの構成」を参照してください。

デスクトップ検索パスと環境変数

デスクトップ検索パスは、デスクトップ・ユーティリティ dtsearchpath によってログイン時に作成されます。dtsearchpath ユーティリティは、環境変数と組み込みの位置の組み合わせを使って検索パスを作成します。

dtsearchpath が読み取る環境変数を「入力変数」といいます。このような変数は、システム管理者またはエンド・ユーザによって設定されます。入力変数は命名規則 DTSP* を使用します。

ログイン時に dtsearchpath が実行されている場合、dtsearchpath は入力変数に割り当てられた値を組み込んで、組み込み位置を追加し、「出力変数」の値を作成します。検索パス 1 つにつき 1 つの出力変数があります。

表 9–2 デスクトップ検索パス環境変数

検索パスの種類	出力環境変数	システム共通の入力変数	個人用入力変数
アプリケーション	DTAPPSEARCHPATH	DTSPSYSAPPHOSTS	DTSPUSERAPPHOSTS
データベース [1]	DTDATABASESEARCHPATH	DTSPSYSDATABASEHOSTS	DTSPUSERDATABASEHOSTS
アイコン	XMICONSEARCHPATH、XMICONBMSEARCHPATH	DTSPSYSICON	DTSPUSERICON
ヘルプ・データ	DTHELPSEARCHPATH	DTSPSYSHELP	DTSPUSERHELP

1. アクション、データ型、フロントパネルの定義

コンポーネントは、出力変数の値を使用します。たとえば、アプリケーション・マネージャは、アプリケーション・グループを検出するために、アプリケーション検索パスの値 (DTAPPSEARCHPATH) を使用します。

検索パスの値の設定

システム共通または個人単位で検索パスを変更できます。変更は、システム共通または個人用入力変数に値を設定して実行します。変更は、すべて組み込み検索パス位置に追加されます。

▼ 検索パスの現在の値 (出力変数) を参照するには

- **検索パスの現在の値を表示するには、`dtsearchpath` コマンドを使用します。**
  - **現在の (ログイン) ユーザの値を取得するには、次のコマンドを入力します。**

    ```
    dtsearchpath -v
    ```

  - **別のユーザの値を取得するには、次のコマンドを入力します。**

    ```
    dtsearchpath -u user
    ```

検索パスの値には、次の変数が含まれます。

%H — DTHELPSEARCHPATH で使用します。ヘルプ・ボリューム名です。

%B — XMICONSEARCHPATH で使用します。アイコン・ファイルのベース名です。

%M — XMICONSEARCHPATH で使用します。アイコン・ファイル (.l、.m、.s、.t) のサイズです。

%L — LANG 環境変数の値です。

▼ 検索パスに個人用の変更を行うには

1. ***HomeDirectory***/.dtprofile を編集するために開きます。

2. 個人用の入力変数を定義する行を追加または編集します。

   たとえば、次の行はユーザの個人用アプリケーション検索パスに位置を追加します。

   ```
   export DTSPUSERAPPHOSTS=/projects1/editors
   ```

3. 変更を有効にするために、ログアウトしてからログインし直します。

▼ 検索パスにシステム共通の変更を行うには

1. **root** としてログインします。

2. /etc/dt/config/Xsession.d/0010.dtpaths ファイルが存在しない場合は、/usr/dt/config/Xsession.d/0010.dtpaths をコピーして作成します。

3. /etc/dt/config/Xsession.d/0010.paths を編集するために開きます。システム共通の入力変数を定義する行を追加または編集します。

   たとえば、次の行はシステム共通のヘルプ検索パスに位置を追加します。

   ```
   export DTSPSYSHELP=/applications/helpdata
   ```

4. システム上のすべてのユーザに、変更を有効にするためにログアウトしてからログインし直すよう通知します。

アプリケーション検索パス

アプリケーション検索パスは、デスクトップがローカル・システムとネットワーク上のアプリケーション・サーバのアプリケーションを検出するのに使用する一次検索パスです。

アプリケーション検索パスに位置が追加されると、別の検索パス (データベース、アイコン、ヘルプ) は、該当するデータの位置を反映させるために自動的に更新されます。したがって、アプリケーション検索パスによって、アプリケーションとデスクトップ構成ファイルの管理が比較的簡単になります。詳細は、149ページの「アプリケーション検索パスがデータベース、アイコン、およびヘルプの検索パスに与える影響」を参照してください。

デフォルトのアプリケーション検索パス

デフォルトのアプリケーション検索パスの位置には、個人用、システム共通、および組み込みがあります。デフォルトの *language* は C です。

個人用の位置 — *HomeDirectory*/.dt/appmanager

システム共通の位置 — /etc/dt/appconfig/appmanager/*language*

組み込みの位置 — /usr/dt/appconfig/appmanager/*language*

アプリケーション検索パス環境変数

アプリケーション検索パスは、組み込み位置と次の入力変数で構成されます。

DTSPSYSAPPHOSTS — システム共通のアプリケーション検索パス入力変数

DTSPUSERAPPHOSTS — 個人用のアプリケーション検索パス入力変数

構成された検索パスは、DTAPPSEARCHPATH 出力変数によって指定されます。

アプリケーション検索パス入力変数の構文

DTSPSYSAPPHOSTS 変数と DTSPUSERAPPHOSTS 変数の構文は次のとおりです。

VARIABLE=location [,*location*...]

location が取り得る構文は次のとおりです。

/path — ローカル (セッション・サーバ) システムのディレクトリを指定します。ローカル・ディレクトリを追加するにはこの構文を使用します。

hostname: — システム共通の /etc/dt/appconfig/appmanager/*language* ディレクトリをシステム *hostname* に指定します。アプリケーション・サーバを追加するにはこの構文を使用します。

hostname:*/path* — リモート・システム *hostname* にディレクトリを指定します。

localhost: — ローカル・システム共通の位置です。このキーワードは、ローカル・システム共通の位置の優先度を変えるのに使用されます。詳細は、148ページの「システム共通のローカル位置の優先度の変更」を参照してください。

アプリケーション検索パスの構成方法

アプリケーション検索パスの値 (DTAPPSEARCHPATH) は、次の位置を組み合わせて作成されます。位置は、上から優先度の高い順に並んでいます。

- DTSPUSERAPPHOSTS 変数を使用して指定した位置
- デフォルトの個人用の位置: *HomeDirectory*/.dt/appmanager
- デフォルト位置: /etc/dt/appconfig/appmanager/*language*
- DTSPSYSAPPHOSTS 変数を使用して指定した位置
- /usr/dt/appconfig/appmanager/*language*

構文 *hostname*: は、ディレクトリ /etc/dt/appconfig/appmanager をシステム *hostname* に指定するために展開されます。

システム共通のローカル位置の優先度の変更

デフォルトでは、ローカル・システム共通の位置 (/etc/dt/appconfig/appmanager/*language*) はリモート位置に優先します。したがって、ローカル・アプリケーション・グループは、同名のリモート・グループに優先します。たとえば、ローカル・システムとリモート・システムの両方に Printer アプリケーション・グループ (/etc/dt/appconfig/appmanager/*language*/Printers) がある場合は、ローカル・グループを使用します。

アプリケーション検索パス入力変数に、ローカル・システム共通のアプリケーション・グループの優先度を指定するには、次の構文を使用します。

```
localhost:
```

たとえば、システムがアプリケーション・サーバ SysA、SysB、SysC にアクセスしなければならず、SysB のシステム共通のアプリケーション・グループを同名の他のローカル・グループよりも優先したいとします。

そのような動作は、DTSPSYSAPPHOSTS の次の値で作成されます。

```
DTSPSYSAPPHOSTS=SysB:,localhost:,SysA:,SysC:
```

アプリケーション検索パスがデータベース、アイコン、およびヘルプの検索パスに与える影響

アプリケーション検索パスに追加されると、対応する位置が、データベース検索パス、アイコン検索パス、およびヘルプ検索パスに自動的に追加されます。この機能により、アプリケーション検索パス入力変数を設定するだけで、アプリケーション・サーバを検索パスに追加できます。

たとえば、DTSPSYSAPPHOSTS を次のように設定した場合は、表 9–3 の検索パスが影響を受けます。

```
export DTSPSYSAPPHOSTS=servera:
```

表 9–3 影響を受ける検索パス

検索パス	検索パスに追加されるディレクトリ
アプリケーション	servera:/etc/dt/appconfig/appmanager/*language*
データベース	servera:/etc/dt/appconfig/types/*language*
アイコン	servera:/etc/dt/appconfig/icons/*language*
ヘルプ	servera:/etc/dt/appconfig/help/*language*

同様に、DTSPSYSAPPHOSTS を次のように設定した場合は、表 9–4 の検索パスが影響を受けます。

```
export DTSPSYSAPPHOSTS=/projects1/apps
```

表 9–4　影響を受ける検索パス

検索パス	検索パスに追加されるディレクトリ
アプリケーション	/projects1/apps/appmanager/*language*
データベース	/projects1/apps/types/*language*
アイコン	/projects1/apps/icons/*language*
ヘルプ	/projects1/apps/help/*language*

データベース (アクションとデータ型) 検索パス

データベース検索パスは、次の定義を格納しているファイルを、指定された位置で検索するようデスクトップに指示します。

- アクションとデータ型定義 (*.dt ファイル)
- フロント・パネル定義 (*.fp ファイル)

データベース・サーバを作成した場合や、データベース・ファイルのローカル位置を追加した場合は、データベース検索パスを変更しなければならないことがあります。

デフォルトのデータベース検索パス

デフォルトのデータベース検索パスの位置には、個人用、システム共通、および組み込みがあります。デフォルトの *language* は C です。

個人用の位置 — *HomeDirectory*/.dt/types

システム共通の位置 — /etc/dt/appconfig/types/*language*

組み込みの位置 — /usr/dt/appconfig/types/*language*

アプリケーション検索パスがデータベース検索パスに与える影響

アプリケーション検索パスに位置が追加されると、適切なデータベースのサブディレクトリが、データベース検索パスに自動的に追加されます (詳細は、149ページの「アプリケーション検索パスがデータベース、アイコン、およびヘルプの検索パスに与える影響」を参照してください) 。

たとえば、アプリケーション・サーバ `hosta:` がアプリケーション検索パスに追加されると、ディレクトリ `hosta:/etc/dt/appconfig/types/`*language* がデータベース検索パスに自動的に追加されます。

データベース検索パス環境変数

データベース検索パスは、組み込み位置と次の入力変数で構成されます。

`DTSPSYSDATABASEHOSTS` — システム共通のデータベース検索パス入力変数

`DTSPUSERDATABASEHOSTS` — 個人用のデータベース検索パス入力変数

アプリケーション検索パス以外の位置を指定するには、これらの入力変数を使用します。

構成されたデータベース検索パスは、出力変数 `DTDATABASESEARCHPATH` によって指定されます。

データベース検索パス入力変数の構文

`DTSPSYSDATABASEHOSTS` 変数と `DTSPUSERDATABASEHOSTS` 変数の構文は次のとおりです。

VARIABLE=location `[,`*location*`...]`

location が取り得る構文は次のとおりです。

/path — ローカル (セッション・サーバ) システムのディレクトリを指定します。ローカル・ディレクトリを追加するにはこの構文を使用します。

hostname: — システム共通のディレクトリ `/etc/dt/appconfig/types/`*language* をシステム *hostname* に指定します。

hostname:*/path* — リモート・システム *hostname* にディレクトリを指定します。

データベース検索パスの構成方法

データベース検索パスの値 (DTDATABASESEARCHPATH) は、次の位置を組み合わせて作成されます。位置は、上から優先度の高い順に並んでいます。

- DTSPUSERDATABASEHOSTS 変数を使用して指定した位置
- DTSPUSERAPPHOSTS 変数から派生した位置
- デフォルトの個人用の位置: *HomeDirectory*/.dt/types
- デフォルト位置: /etc/dt/appconfig/types/*language*
- DTSPSYSDATABASEHOSTS 変数を使用して指定した位置
- DTSPSYSAPPHOSTS 変数から派生した位置
- /usr/dt/appconfig/types/*language*

  構文 *hostname*: は、ディレクトリ /etc/dt/appconfig/types をシステム *hostname* に指定するために展開されます。

アイコン検索パス

アイコン検索パスは、デスクトップが使用するビットマップとピックスマップのイメージ・ファイルを格納しているファイルを、指定された位置で検索するようデスクトップに指示します。

デフォルトのアイコン検索パス

デフォルトのアイコン検索パスの位置には、個人用、システム共通、および組み込みがあります。デフォルトの *language* は C です。

個人用の位置 — *HomeDirectory*/.dt/icons

システム共通の位置 — /etc/dt/appconfig/icons/*language*

組み込みの位置 — /usr/dt/appconfig/icons/*language*

アプリケーション検索パスがアイコン検索パスに与える影響

アプリケーション検索パスに位置が追加されると、適切なアイコンのサブディレクトリが、アイコン検索パスに自動的に追加されます (詳細は、149ページの「アプリケーション検索パスがデータベース、アイコン、およびヘルプの検索パスに与える影響」を参照してください)。

たとえば、アプリケーション・サーバ `hosta:` がアプリケーション検索パスに追加されると、ディレクトリ `hosta:/etc/dt/appconfig/icons/`*language* がアイコン検索パスに自動的に追加されます。

アイコン検索パス環境変数

アイコン検索パスは、組み込み位置と次の入力変数で構成されます。

`DTSPSYSICON` — システム共通のアイコン検索パス入力変数

`DTSPUSERICON` — 個人用のアイコン検索パス入力変数

アプリケーション検索パス以外の位置を指定するには、これらの入力変数を使用します。

構成されたアイコン検索パスは、次の 2 つの出力変数によって指定されます。

`XMICONSEARCHPATH` — カラー・ディスプレイに使用します。

`XMICONBMSEARCHPATH` — モノクロ・ディスプレイに使用します。

アイコン検索パス入力変数の構文

`DTSPSYSICON` 変数と `DTSPUSERICON` 変数の構文は次のとおりです。

VARIABLE=location `[,`*location*`...]`

location が取り得る構文は次のとおりです。

/path — ローカル (セッション・サーバ) システムのディレクトリを指定します。ローカル・ディレクトリを追加するにはこの構文を使用します。

別のシステムの位置を指定するには、ネットワーク・ファイル名を使用します (例: `/nfs/servera/projects/icons`)。

アイコン検索パスの構成方法

アイコン検索パスの値 (XMICONSEARCHPATH と XMICONBMSEARCHPATH) は、次の位置を組み合わせて作成されます。位置は、上から優先度の高い順に並んでいます。

- DTSPUSERICON 変数を使用して指定した位置
- DTSPUSERAPPHOSTS 変数から派生した位置
- デフォルトの個人用の位置: *HomeDirectory*/.dt/icons
- デフォルト位置: /etc/dt/appconfig/icons/*language*
- DTSPSYSICON 変数を使用して指定した位置
- DTSPSYSAPPHOSTS 変数から派生した位置
- /usr/dt/appconfig/icons/*language*

カラーの検索パスとモノクロの検索パスは、ピックスマップとビットマップの優先度だけが異なります。XMICONSEARCHPATH 変数はビットマップよりピックスマップを優先し、XMICONBMSEARCPATH はピックスマップよりビットマップを優先します。

ヘルプ検索パス

ヘルプ検索パスは、システムに登録されるヘルプ情報を格納しているファイルを、指定された位置で検索するようデスクトップに指示します。

デフォルトのヘルプ検索パス

デフォルトのヘルプ検索パスの位置には、個人用、システム共通、および組み込みがあります。デフォルトの *language* は C です。

個人用の位置 — *HomeDirectory*/.dt/help

システム共通の位置 — /etc/dt/appconfig/help/*language*

組み込みの位置 — /usr/dt/appconfig/help/*language*

アプリケーション検索パスがヘルプ検索パスに与える影響

アプリケーション検索パスに位置が追加されると、適切なヘルプのサブディレクトリが、ヘルプ検索パスに自動的に追加されます (詳細は、149ページの「アプリケーション検索パスがデータベース、アイコン、およびヘルプの検索パスに与える影響」を参照してください)。

たとえば、アプリケーション・サーバ `hosta:` がアプリケーション検索パスに追加されると、ディレクトリ `hosta:/etc/dt/appconfig/help/`*language* がヘルプ検索パスに自動的に追加されます。

ヘルプ検索パス環境変数

ヘルプ検索パスは、組み込み位置と次の入力変数で構成されます。

`DTSPSYSHELP` — システム共通のヘルプ検索パス入力変数

`DTSPUSERHELP` — 個人用のヘルプ検索パス入力変数

アプリケーション検索パス以外の位置を指定するには、これらの入力変数を使用します。

構成されたヘルプ検索パスは、出力変数 `DTHELPSEARCHPATH` によって指定されます。

ヘルプ検索パス入力変数の構文

`DTSPSYSHELP` 変数と `DTSPUSERHELP` 変数の構文は次のとおりです。

VARIABLE=location `[,`*location*`...]`

location が取り得る構文は次のとおりです。

/path — ローカル (セッション・サーバ) システムのディレクトリを指定します。ローカル・ディレクトリを追加するにはこの構文を使用します。

別のシステム上での位置を指定するには、ネットワーク・ファイル名を使用します (例: `/nfs/servera/projects/help`)。

ヘルプ検索パスの構成方法

ヘルプ検索パスの値 (DTHELPSEARCHPATH) は、次の位置を組み合わせて作成されます。位置は、上から優先度の高い順に並んでいます。

- DTSPUSERHELP 変数を使用して指定した位置
- DTSPUSERAPPHOSTS 変数から派生した位置
- デフォルトの個人用の位置: *HomeDirectory*/.dt/help
- デフォルト位置: /etc/dt/appconfig/help/*language*
- DTSPSYSHELP 変数を使用して指定した位置
- DTSPSYSAPPHOSTS 変数から派生した位置
- /usr/dt/appconfig/help/*language*

ローカライズされた検索パス

出力変数には、ローカライズされた位置とデフォルト (C) 位置の両方のエントリがあります。

たとえば、デフォルトのアプリケーション検索パスは次のとおりです。

```
HomeDirectory/.dt/appmanager
   /etc/dt/appconfig/appmanager/language
   /etc/dt/appconfig/appmanager/C
   /usr/dt/appconfig/appmanager/language
   /usr/dt/appconfig/appmanager/C
```

language は LANG 環境変数の値です。

(システム共通と組み込みの) それぞれの範囲で、言語固有の位置はデフォルトの位置に優先します。

第 10 章

アクションとデータ型の概要

「アクション」と「データ型」は、アプリケーションをデスクトップへ統合するときに非常に役立つコンポーネントです。アプリケーションの起動とデータ・ファイルの処理を行うユーザ・インタフェースを作成する方法を提供します。

この章では、アクションとデータ型の概念を説明します。次の項目について説明します。

- アプリケーションのアクションとデータ型を作成する理由
- アクションとデータ型の関連性
- アクションおよびデータ型とデスクトップ印刷との関連性

アクションとデータ型を作成するときの手順と規則については、このマニュアルの次の 3 つの章で説明しています。

- 第 11 章では、デスクトップ・アプリケーションのアクション作成を使用してアクションとデータ型を作成する方法について説明します。

  多くのアプリケーションは、その定義の構文規則がわからなくても、アクション作成を使用してアクションとデータ型を作成できます。

- 第 12 章と第 13 章では、構成ファイルを作成および編集することにより、手入力でアクションとデータ型を作成する方法について説明します。

  アクション作成ツールでサポートしていない高度な機能を使用する場合は、手入力でアクションとデータ型を作成する必要があります。

アクションの概要

アクションは、アプリケーションの実行やデータ・ファイルを開くなど、自動化されたデスクトップのタスクを書いた命令です。アクションは、アプリケーション・マクロまたはプログラミング関数とよく似た動作をします。各アクションには、アクションを実行するのに使用するための名前があります。

一度アクションを定義すると、デスクトップ・ユーザ・インタフェースに適用されるので、タスクを実行しやすくなります。デスクトップは、アイコン、フロントパネル・コントロール、アクションに対するメニュー項目などのユーザ・インタフェース・コンポーネントに接続する機能を提供します。

たとえば、アプリケーション・マネージャの [デスクトップツール] アプリケーション・グループには、さまざまなユーティリティを起動するアイコンがあります。

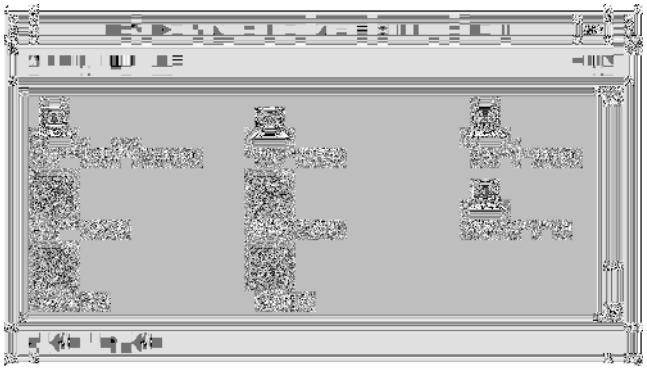


図 10–1　[デスクトップツール] アプリケーション・グループのアクション・アイコン

これらのアイコンをダブルクリックすると実行できます。次に例として、[Xwd 表示] というラベルの付いたアイコンをダブルクリックしたときにアクションを実行するという定義の一部を示します。アクションの定義は構成ファイル `/usr/dt/appconfig/types/`*language*`/xclients.dt` にあります。

```
ACTION Xwud
 {
  LABEL    Xwd Display
  TYPE    COMMAND
  EXEC_STRING  /usr/bin/X11/xwud -noclick -in \
        %(File)Arg_1"Xwd File To Display:"%
  …
 }
```

アイコンをダブルクリックすると、アクションの EXEC_STRING フィールドにあるコマンドが実行されます。

フロントパネルもアクションを使用します。次に例として、[個人アプリケーション] サブパネルの [端末エミュレータ] というラベルの付いたコントロールの定義の一部を示します。コントロールの定義は、構成ファイル /usr/dt/appconfig/types/*language*/dtwm.fp にあります。

```
CONTROL Term
 {
  ICON    Fpterm
  LABEL    Terminal
  PUSH_ACTION  Dtterm
  …
 }
```

PUSH_ACTION フィールドは、コントロールをクリックすると実行されるアクション (ここでは Dtterm という名前のアクション) を指定します。

他のアクションの使用方法としては、メニューで使用する方法があります。通常、データ・ファイルでは、ファイル・マネージャの [選択] メニューにアクションがあります。たとえば XWD ファイル (.xwd または .wd で終わる名前のファイル) には、Xwud アクションを実行することによって画面イメージを表示する [開く] アクションがあります。

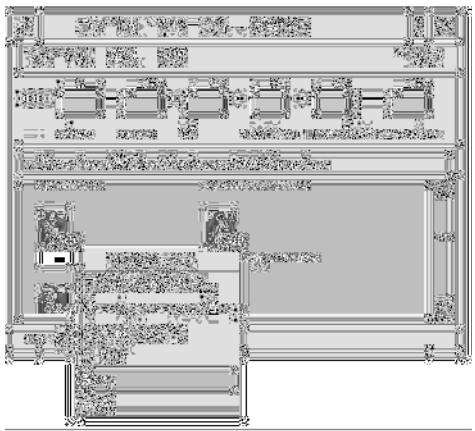


図 10–2　データ型 XWD のファイルの [開く] アクション

[選択] メニューのアクションは、XWD ファイルのデータ型定義で指定されます。定義は、構成ファイル /usr/dt/appconfig/types/*language*/xclients.dt にあります。

```
DATA_ATTRIBUTES XWD
 {
  ACTIONS    Open,Print
```

```
  ICON    Dtxwd
  …
 }
```

XWD データ型と、それに関連する [開く] および [印刷] アクションは、165ページの「データ型によるデータ・ファイルのアクションへの接続方法」で説明します。

アクションによるアプリケーション用アイコンの作成方法

[デスクトップツール] アプリケーション・グループには [Xwd 表示] があります。このアイコンをダブルクリックすると、X クライアントの xwud が実行されます。ただし、このアイコンは実際の xwud の実行可能ファイル /usr/bin/X11/xwud を示すわけではありません。

同じディレクトリに Xwud というファイルがあるので、アプリケーション・グループに [Xwd 表示] というラベルが付いたアイコンが表示されます (図 10–3 参照)。このファイルは、基本となるアクションを同じ名前、つまり Xwud で示します。アクション定義では、アクション名はキーワード ACTION に続く名前です。

```
ACTION Xwud
 {
  LABEL      Xwd Display
  TYPE      COMMAND
  WINDOW_TYPE    NO_STDIO
  EXEC_STRING    /usr/bin/X11/xwud -noclick -in \
         %(File)Arg_1"Xwd File To Display:"%
  DESCRIPTION    The Xwd Display (Xwud) XwdDisplay action \
         displays an xwd file that was created using the \
         Xwd Capture (Xwd) action. It uses \
         the xwud command.
 }
```

ファイルは、アクションを示すので「アクション・ファイル」と呼びます。ファイルがアクションと同じ名前の実行可能ファイルの場合は、アクション・ファイルです。アプリケーション・マネージャ (またはファイル・マネージャ) にあるアイコンは、「アクション・アイコン」、またはダブルクリックするとアプリケーションが起動するので「アプリケーション・アイコン」と呼びます。

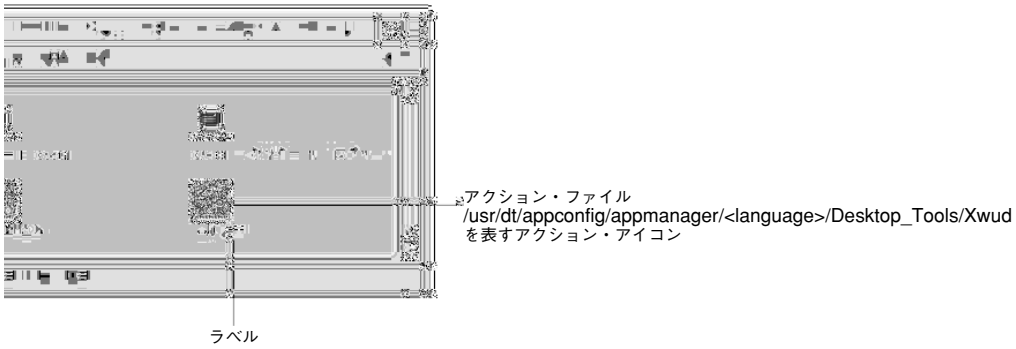


図 10–3　アクション・ファイルを示すアプリケーション (アクション)・アイコン

アプリケーション・マネージャは実行可能ファイルを検出すると、アクション・データベースを検索して、ファイル名と一致する名前のアクションがあるかどうか調べます。一致するファイルがある場合、アプリケーション・マネージャは、そのファイルがアクション・ファイルであると認識します。

アクション・ファイルの内容は関連ありません。通常、アクション・ファイルには、デスクトップ関数を説明するコメントがあります。

注 - 「アクション・ファイル」は「アクション定義ファイル」とは異なります。「アクション・ファイル」は、アクションと同じ名前を持つファイルです。ファイル・マネージャまたはアプリケーション・マネージャの「アプリケーション・アイコン」の作成に使用します。「アクション定義ファイル」は、アクションの定義が入った *name*.dt という名前のファイルです。

ファイルがアクション・ファイルであることをデスクトップが判別すると、基本となるアクション定義がアクション・ファイルの外観と動作の定義に使用されます。

- EXEC_STRING フィールドは、アプリケーション・アイコンの動作を指定します。[Xwd 表示] アイコンの場合、EXEC_STRING は、アクション・アイコンが正しいコマンド行引き数で X クライアントの xwud を実行することを指定します。
- LABEL フィールドは、アプリケーション・アイコンのラベルを指定します。
- DESCRIPTION フィールドは、ユーザが [アイテムヘルプ] を要求したときに表示するテキストを記述します。
- Xwud アプリケーション・アイコンは、アクション定義に別のイメージを指定する ICON フィールドがあるため、アクションのデフォルトのアイコン・イメージを使用します。

  反対に、[ファイルの圧縮] というラベルの付いたアイコンは、基本となるアクション定義に ICON フィールドがあるため、別のアイコン・イメージを使用します。

次に例を示します。

```
ACTION Compress
   {
    LABEL    Compress File
    ICON    Dtcmprs
    …
   }
```

図 10–4　アクション定義の ICON フィールドで指定したアイコン・イメージ

Xwud アクションは、実行するコマンド (EXEC_STRING) が定義に含まれているため、「コマンド」アクションと呼びます。アクション定義の TYPE フィールドは、アクション型を定義します。

最初に、[Xwd 表示] アイコンは [デスクトップツール] アプリケーション・グループに表示されます。ただし、書き込み権がある場合は、任意のディレクトリにアクション・アイコンのコピーを作成できます。Xwud アクション定義がデータベースの一部である間は、Xwud と名付けて作成した実行可能ファイルは、アクションを示すアクション・ファイルとなります。ファイル・マネージャまたはアプリケーション・マネージャのアイコンは、アクションを実行するのに使用されます。

アクションがデータ・ファイルを引き数として使用する方法

コマンドの「引き数」は、コマンドを動作させるためのもので、通常はファイルです。アクションは、ファイル引き数を受け取るように記述できます。

たとえば、Xwud アクションの EXEC_STRING は、ファイル引き数が必須であることを指定します。

```
EXEC_STRING     /usr/bin/X11/xwud -noclick -in \
        %(File)Arg_1"Xwd File To Display:"%
```

Arg という用語は「引き数」を意味します。構文 Arg_1 は最初の引き数であること、(File) はアクションが引き数をファイルとして処理することを意味します。

ファイル引き数を指定するのに最も簡単な方法は、データ・ファイルをアプリケーション・アイコンにドロップすることです。デスクトップはドロップされたファイルのパスを判別し、コマンド行の % 記号の間のテキストの位置 (%(File)Arg_1"Xwd File To Display:"%) にパスを置き換えます。したがって、実行されるコマンドは次のとおりです。

```
/usr/bin/X11/xwud -noclick -in file_path
```

アプリケーション・アイコンをダブルクリックすると、デスクトップは EXEC_STRING からファイル引き数が必須であることを判断し、ファイル名またはパスを入力するようダイアログ・ボックスでプロンプトします。Xwud アクションの場合、プロンプトは次のとおりです。

```
Xwd File To Display:
```

ユーザが指定するファイル名またはパスをファイル引き数として使用します。

アクションのその他の使い方

アプリケーションの起動の他に、アクションは次のような機能を作成するためにデスクトップで使用します。

- フロントパネル

  フロントパネル・コントロールの定義には、コントロールをクリックしたとき、またはファイルをドロップしたときに実行するアクションを指定するフィールドがあります。詳細は、260ページの「フロントパネル・コントロール定義」を参照してください。

- メニュー

  [ウィンドウ] メニューとワークスペース・メニューの定義の構文により、メニュー項目で実行するアクションを指定できます。詳細は、281ページの「ワークスペース・マネージャのメニュー」と dtwmrc(4) のマニュアル・ページを参照してください。

- アプリケーション間通信

  ToolTalk メッセージと呼ばれる特殊なアクション (TT_MSG) を使用して情報を送受信できるようにアプリケーションを設計できます。TT_MSG アクションは、デスクトップの開発者環境用マニュアルで説明します。

データ型の概要

新規データ・ファイルを作成した場合、ファイル・マネージャのファイルアイコン外観と動作は、作成したデータ・ファイルの型によって異なります。ファイルとディレクトリの外観および動作をカスタマイズするこの機能は、デスクトップのデータ型作成機能によって提供されます。

データ型とは何か

データ型は、デスクトップ・データベース内で定義される構造です。次に例として、XWD データ型の定義を示します。定義は、構成ファイル /usr/dt/appconfig/types/*language*/xclients.dt にあります。

```
DATA_ATTRIBUTES XWD
 {
  ACTIONS    Open,Print
  ICON     Dtxwd
  NAME_TEMPLATE  %s.xwd
  MIME_TYPE       application/octet-stream
  SUNV3_TYPE      xwd-file
  DESCRIPTION    This file contains a graphics image in the XWD \
        format. These files are typically created by \
        taking snapshots of windows using the XwdCapture \
        action. Its data type is named XWD. XWD files \
        have names ending with '.xwd' or '.wd'.
 }


DATA_CRITERIA XWD1
 {
  DATA_ATTRIBUTES_NAME   XWD
  MODE         f
  NAME_PATTERN      *.xwd
 }

DATA_CRITERIA XWD2
 {
  DATA_ATTRIBUTES_NAME   XWD
  MODE         f
  NAME_PATTERN      *.wd
 }
```

それぞれのデータ型定義には、次の 2 つの部分があります。

DATA_ATTRIBUTES — データ型の外観と動作を説明します。

DATA_CRITERIA — そのデータ型に属するファイルをカテゴリに分類するための (命名および内容に関する) 規則を指定します。

DATA_ATTRIBUTES_NAME フィールドは、条件を属性に接続します。

1 つの DATA_ATTRIBUTE に対して複数の DATA_CRITERIA が存在することも可能です。たとえば XWD データ型には、.xwd または .wd で終わる名前という 2 つの異なる命名条件 (NAME_PATTERN) を指定する 2 つの基準があります。

データ型によるデータ・ファイルのアクションへの接続方法

XWD データ型を想定してください。ファイルに 2 つのファイル名拡張子 .xwd または .wd のいずれかを指定することにより、XWD 型のファイルを作成します。デスクトップは、その型のファイルを設計するための「基準」としてファイル名を使用します。

XWD データ型は、次の内容を備えるデータ型の各ファイルを提供します。

- データ・ファイルを認識するのに役立つ一意のアイコン・イメージ
- データ型を通知する [アイテムヘルプ]
- [開く] および [印刷] アクションを含むファイル・マネージャでカスタマイズされた [選択] メニュー。XWD ファイルの [開く] アクションは、Xwud アクションを実行します。

[選択] メニューからのアクションの実行

ファイル・マネージャの [選択] メニューは、ファイルまたはディレクトリが選択された場合にのみ使用できます。[選択] メニューの下のコマンドは、データ型によって異なります。たとえば XWD ファイルを選択すると、[選択] メニューには [開く] と [印刷] という項目が含まれます。

データ型定義の ACTIONS フィールドは、データ型の [選択] メニューの下に追加されるコマンドを指定します。

```
DATA_ATTRIBUTES XWD
{
          ACTIONS         Open,Print
          …
}
```

[選択] メニューの内容はデータ型に依存しますが、[開く] アクションはほとんどのデータ型にあります。つまり、ファイル・マネージャの特定のデータ型のファイルを選択して、[選択] メニューを表示すると、[開く] コマンドが表示されます。

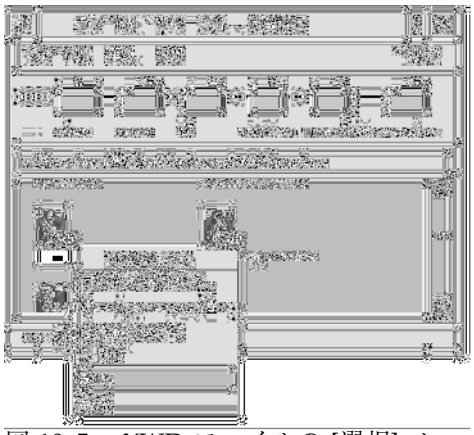


図 10–5　XWD ファイルの [選択] メニュー

[開く] アクションは、通常データ・ファイルが関連しているアプリケーションを実行します。たとえば、XWD ファイルを開くと Xwud アクションが実行されます。このアクションは、順番に xwud X クライアントを実行して画面イメージを表示します。つまり XWD データ型の場合、[開く] アクションは Xwud アクションと同じです。同様に、TEXTFILE データ型のファイルを開くとテキスト・エディタが実行され、BM (ビットマップ) ファイルまたは PM (ピックスマップ) ファイルを開くとアイコン・エディタが実行されます。

異なる動作を行うさまざまな [開く] アクションを作成する機能は、アクション定義の次の 2 つの機能によって実現されます。

- アクション・マッピング

  アクション・マッピングにより、直接コマンドを実行せずに、他のアクションを実行するアクションを作成できます。たとえば、Xwud アクションにマップ (して実行) する [開く] アクションを作成できます。

- アクションを制限するデータ型

  アクション定義には、あるデータ型にのみアクションを制限する ARG_TYPE フィールドを指定できます。たとえば、Xwud アクションにマップする [開く] アクションを XWD データ型のファイルだけに適用するよう指定できます。

[開く] アクションを XWD データ型の Xwud アクションにマップするアクションの定義を次に示します。この定義は、データベース構成ファイル /usr/dt/appconfig/types/C/xclients.dt にあります。

```
ACTION Open
 {
  LABEL    Open
  ARG_TYPE   XWD
  TYPE    MAP
```

```
  MAP_ACTION  Xwud
 }
```

TYPE フィールドは、このアクションがマップ・アクションであることを指定します。MAP_ACTION フィールドは、このアクションが Xwud アクションを実行することを指定します。ARG_TYPE フィールドは、このアクションが XWD データ型のファイルだけに適用されることを指定します。

上記の定義と対照的な定義を次に示します。これはデータベース・ファイル /usr/dt/appconfig/types/C/dt.dt にあります。

```
ACTION Open
 {
  LABEL     Open
  ARG_TYPE    BM
  TYPE     MAP
  MAP_ACTION   Dticon
 }
```

この定義は、ARG_TYPE データ型の BM (ビットマップ・ファイル) に適用されます。定義は [開く] アクションを、アイコン・エディタを実行する Dticon アクションにマップします。

データ型のダブルクリック動作の定義

データ型のダブルクリック動作は、ACTIONS フィールドの最初のエントリで定義します。たとえば XWD データ型をダブルクリックすると、Xwud アクションを順番に実行する [開く] アクションを実行します。

データ・ファイルのアクション・アイコンへのドロップ

データ・ファイルをアクション・アイコンへドロップすると、データ・ファイルをアクションの引き数としてシステムはアクションを実行します (詳細は、162ページの「アクションがデータ・ファイルを引き数として使用する方法」を参照してください)。

たとえば、XWD データ・ファイルを [Xwd 表示] アイコンへドロップすると、データ・ファイルを引き数として Xwud アクションが実行されます。これにより、そのデータ・ファイルとともに xwud X クライアントが実行されます。

データ型に応じたデスクトップ印刷の作成

デスクトップ印刷は、データ・ファイルを印刷する次の 2 つの方法を提供します。

- 使用可能な場合は、ファイル・マネージャの [選択] メニューの [印刷] コマンドを使用する
- データ・ファイルをデスクトップ・プリンタ・ドロップ領域(フロントパネル・プリンタ・コントロールまたは印刷マネージャのプリンタ・アイコン)にドロップする

デスクトップ印刷の他にも、たくさんのアプリケーションがアプリケーション内から印刷する方法を提供しています。

デスクトップ印刷は、[印刷] という名前のアクションを使用します。[印刷] は [開く] のように、さまざまなデータ型に使用されるアクション名です。したがって、[印刷] アクションは、アクション・マッピングと ARG_TYPE フィールドを使用し、各データ型の印刷をカスタマイズします。

次に例として、XWD データ型の [印刷] アクションを示します。定義は /usr/dt/appconfig/types/*language*/xclients.dt にあります。

```
ACTION Print
 {
  LABEL     Print
  ARG_TYPE    XWD
  TYPE     MAP
  MAP_ACTION   NoPrint
 }
```

この [印刷] アクションは XWD ファイルに固有で、NoPrint アクションにマップされます。NoPrint は、/usr/dt/appconfig/types/*language*/dt.dt で定義される特殊アクションです。NoPrint アクションは、このデータ型が印刷できないことをユーザに通知するダイアログ・ボックスを表示します。

XWD の [印刷] アクションを、次の PCL ファイルの [印刷] アクションと比較します。

```
ACTION Print
{
  LABEL    Print
  ARG_TYPE   PCL
  TYPE    MAP
  MAP_ACTION  PrintRaw
 }
```

PrintRaw アクションは、PCL ファイルを印刷するコマンド行が入っている構成ファイル /usr/dt/appconfig/types/*language*/print.dt で定義されています。

```
ACTION PrintRaw
 {
  TYPE      COMMAND
  WINDOW_TYPE    NO_STDIO
  EXEC_STRING    /usr/dt/bin/dtlp -w %(File)Arg_1%
 }
```

第 11 章

アクション作成ツールを使ったアクションとデータ型の作成

アクション作成ツールは、次のものを作成します。

- アプリケーションを起動するアクション
- アプリケーションのデータ・ファイルの 1 つ以上のデータ型
- アプリケーションのデータ・ファイルを開く、または印刷するためのアクション

アクション作成ツールは、オペレーティング・システムのコマンドとシェル・スクリプトを実行するための単純なアクションを作成するのにも便利です。

- 171ページの「アクション作成ツールの機能」
- 172ページの「アクション作成ツールの制限」
- 173ページの「アクション作成ツールを使ったアプリケーションのアクションとデータ型の作成」

詳細は、`dtcreate(1X)` のマニュアル・ページを参照してください。

アクション作成ツールの機能

アクション作成ツールには、アクションとアクションに関連するデータ型を作成するためのメイン・ウィンドウとダイアログ・ボックスのセットがあります。

アクション作成ツールには、次のような機能があります。

- コマンドを実行するアクション定義を作成します。

- ファイル *HomeDirectory*/.dt/types/*action_name*.dt を作成します。このファイルは、アプリケーション用に作成されたアクションとデータ型の定義を格納します。
- ホーム・ディレクトリに「アクション・ファイル」を作成します。アクション・ファイルは、アクションと同名の実行可能ファイルです。

  ファイル・マネージャのアクション・ファイル表示は、ダブルクリックするとアプリケーションが起動するので「アプリケーション・アイコン」と呼ばれます。

  アクションを作成するときにドロップ可能なデータ型を指定することにより、アクション・アイコンをドロップ領域にすることもできます。
- (省略可能) アプリケーションのデータ・ファイルに対して 1 つ以上のデータ型を作成します。
- 各データ型に対して [開く] アクションを作成します。
- (省略可能) 各データ型に対して [印刷] アクションを作成します。
- アクションとデータ型のデータベースを再読み込みします。これにより、アクションとデータ型はただちに有効になります。

アクション作成ツールの制限

アクション作成ツールは、アプリケーションを実行するためのアクションとデータ型を作成するよう設計されています。アクションとデータ型には非常に柔軟性がありますが、手作業で定義を作成した場合しかアクセスできない追加機能があります。

詳細は、次の章を参照してください。

- 第 12 章
- 第 13 章

アクションの制限

次の条件のいずれかが当てはまる場合は、アクション作成ツールを使用してアプリケーションのアクションを作成できません。

- コマンド行に、ファイルではない引き数 (パラメータ) が必要な場合

  たとえば、アクション作成ツールを使用して次のコマンドのアクションを記述できません。

```
lp -ddevice filename
```

このコマンドでは、コマンドを実行するたびに *device* の値を指定しなければなりません。

- アプリケーション・アイコンのラベルがアクション名と異なる場合

  たとえば、アクション作成ツールを使用して、既存のアクションのローカル言語バージョンを提供できません。

- アクションがアクション・データベースの拡張機能を必要とする場合

  拡張機能を必要とするアクションには次のようなものがあります。

  - アクション定義から、離れたリモートシステムにコマンドを発行するアクション
  - 他のアクションを起動するアクション
  - 別のユーザ (スーパーユーザなど) として実行しなければならないアクション
  - 「マップ」機能を広範囲に活用するアクション
  - 提供されるファイル引き数の数によって動作が非常に異なるアクション

データ型の制限

次の条件のいずれかが当てはまる場合は、アクション作成ツールを使用してアプリケーションのデータ型を作成できません。

- [開く] と [印刷] 以外のデータ型に関連付けられた追加のアクションが必要な場合
- データ型の [開く] アクションが、そのアクションのコマンドではない場合

  たとえば、アクション作成ツールを使用して、アプリケーションのアプリケーション・グループを表すディレクトリに一意のアイコンを提供するデータ型を作成できません。

アクション作成ツールを使ったアプリケーションのアクションとデータ型の作成

アクション作成ツールを実行する前に、アプリケーションについていくつか知っておく必要があることがあります。

- アプリケーションを起動するコマンド行

コマンド行に必要なファイル引き数が指定されているか、オプションのファイル引き数が指定されているか、あるいはファイル引き数が指定されていないかを知る必要があります。

アプリケーションにファイルではない引き数が必要な場合は、アクション作成ツールを使用してアクションを作成できません。

- アプリケーションが受け取れるデータ・ファイルの型

  アプリケーションによっては、1 種類のデータ型しか受け取れません。通常のアプリケーション (ASCII エディタやグラフィック・エディタなど) は複数のデータ型を受け取れます。

- アプリケーションがデータ・ファイルを識別する方法

  これは命名規則 (たとえば .doc で終わるファイル名など) で、ファイルの内容によります。アプリケーションがファイル命名規則を使用しない場合でも、アクション・アイコン用に命名規則を設定できます。

- (省略可能) ファイルを印刷するコマンド行

▼ アプリケーション用にアクションを作成するには

1. **[デスクトップアプリケーション] グループで [アクション作成] をダブルクリックします。**

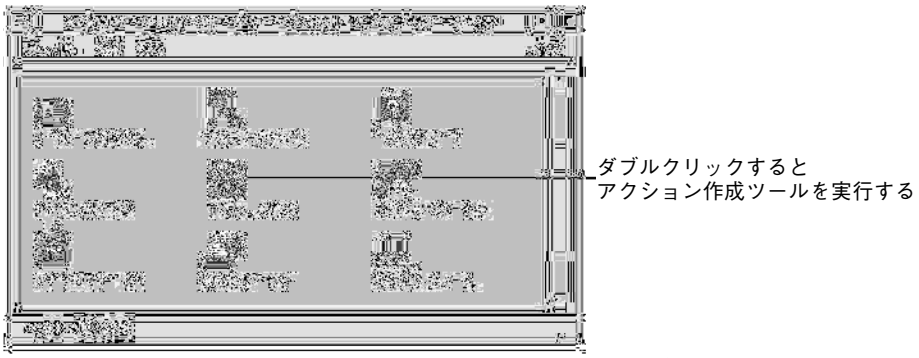


図 11–1　アプリケーション・マネージャの [アクション作成] アイコン

[アクション作成] メイン・ウィンドウが表示されます。

図 11–2　[アクション作成] メイン・ウィンドウ

2. **[アクション名]** テキスト・フィールドに、アクション・アイコンのラベル名を入力します。

3. **[アクション・アイコン]** コントロールを使用して、アイコンをアプリケーションに指定します。最初はデフォルト・アイコンが表示されます。

   - 別の既存のアイコンを選択するには、[アイコンセット検索] をクリックして [アイコンセット検索] ダイアログ・ボックスを表示します。詳細は、183ページの「アイコンを指定するための [アイコンセット検索] ダイアログ・ボックスの使用」を参照してください。
   - 新しいアイコンを作成するには、[アイコン編集] を選択してアイコン・エディタを実行します。

4. **[アクションを開いた (ダブルクリックした) 場合のコマンド]** テキスト・フィールドに、アプリケーションを起動するコマンドを入力します。

   ファイル引き数には、次のように構文 *$n* を使用します。

```
emacs
   bitmap $1
   diff $1 $2
   lp -oraw $1
```

コマンド行にファイル引き数 (*$n*) を指定した場合、アクション・アイコンはファイルのドロップ領域になります。

コマンド行は、シェルの使用を明示的に指定しない限り、シェルには渡されません。たとえば、次の行はシェル処理を行います。

```
/bin/sh -c 'ps | lp'
   /bin/sh -c 'spell $1 | more'
```

5. **[**アクション・アイコンのヘルプ・テキスト**]** テキスト・フィールドに、アクション・アイコンのためのアイテムヘルプテキストを入力します。

   テキストは、テキスト・フィールド内で自動的に折返されます。しかし、その改行はオンラインでは保持されません。強制改行を指定する場合は、`\n` を使用します。

6. アクションに必要なウィンドウ・サポートを **[**ウィンドウタイプ**]** オプション・メニューから選択します。

   グラフィカル (X Window System) — アプリケーションは独自のウィンドウを作成します。

   端末エミュレータ (自動的に閉じる) — アプリケーションを終了するときに自動的に閉じる端末エミュレータ・ウィンドウでアプリケーションを実行します。

   端末エミュレータ (手動で閉じる) — ユーザが明示的にウィンドウを閉じるまで開いている端末エミュレータ・ウィンドウでアプリケーションを実行します。

   出力なし — アプリケーションはディスプレイに出力しません。

7. 次の手順に従います。

   - アプリケーションにデータ・ファイルがあり、そのデータ・ファイル用に 1 つ以上のデータ型を作成する場合は、177ページの「アプリケーション用に 1 つ以上のデータ型を作成するには」を参照してください。

- データ型を作成する必要がない場合は、[ファイル] メニューから [保存] を選択してアクションを保存します。次に、ホーム・ディレクトリでアイコンをダブルクリックし、新しいアクションをテストします。

▼ アプリケーション用に 1 つ以上のデータ型を作成するには

1. 174ページの「アプリケーション用にアクションを作成するには」の手順に従って、アプリケーションのアクションを定義します。

2. ウィンドウを拡張するために、**[アクション作成]** ウィンドウで **[拡張機能]** ボタンをクリックします。

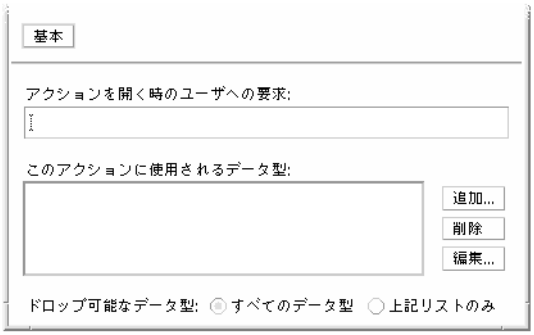


図 11–3　[アクション作成] メイン・ウィンドウの拡張機能

3. アイコンをダブルクリックしたときに、アプリケーション・アイコンがファイル引き数の指定を求めるプロンプトを表示するようにする場合は、プロンプト・テキストを **[アクションを開く時のユーザへの要求]** テキスト・フィールドに入力します。

   このテキスト・フィールドについては、次のガイドラインに従ってください。

   - アプリケーションのコマンド行に「必要な」ファイル引き数が指定されている場合は、このフィールドを使用しなければなりません。
   - コマンド行にファイル引き数が指定されていない場合は、このフィールドは空白にしておかなければなりません。
   - アプリケーションのコマンド行のファイル引き数がオプションの場合は、次のいずれかを選択できます。プロンプト・テキストを提供すると、アクション・アイコンをダブルクリックしたときにファイルの指定を求めるプロンプ

トが表示されます。プロンプト・テキストを提供しないと、アクション空文字列をファイル引き数として実行します。

4. アクションが引き数として受け取るファイル・タイプを指定します。

   - アクションがどんなデータ型でも受け取れる場合は、[すべてのデータ型] を選択します。
   - アクションが、あるアプリケーション用に作成したデータ型しか受け取れない場合は、[上記リストのみ] を選択します。

   最初は、[このアクションに使用されるデータ型] リストは空です。アプリケーション用にデータ型を作成するたびに、リストに追加されます。

5. **[**このアクションに使用されるデータ型**]** リスト・ボックスの横の **[**追加**]** ボタンをクリックして **[**データ型の追加**]** ダイアログ・ボックスを表示します。

図 11–4　[アクション作成] の [データ型の追加] ダイアログ・ボックス

6. **(省略可能)** デフォルトのデータ型名を使用しない場合は、**[データ型ファミリ名]** テキスト・フィールドに、データ型の新しい名前を入力します。

   データ型名には、スペースは使用できません。データ型名は、アプリケーション・ユーザには見えません。データ型名は、データ型定義を識別するためにアクションまたはデータ型データベースで使用します。

7. **[識別する特性]** ボックスの横の **[編集]** ボタンをクリックして、**[識別する特性]** ダイアログ・ボックスを表示します。

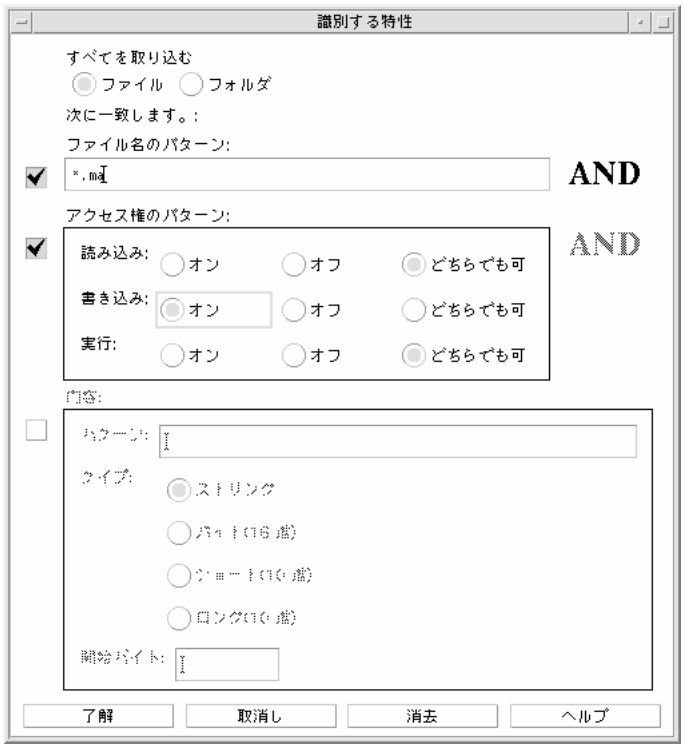


図 11–5　[アクション作成] の [識別する特性] ダイアログ・ボックス

データ型の特徴が、そのデータ型を他のデータ型と区別するために使用される基準になります。次の中から 1 つ以上の基準を選択できます。

ファイルまたはフォルダ — ファイルだけ、またはフォルダだけに適用されるデータ型

ファイル名のパターン — ファイル名に基づくデータ型の分類

アクセス権のパターン — 読み取り権、書き込み権、実行権

内容 — ファイルの指定された部分の内容

8. データ型が、ファイルとフォルダのどちらを表すか選択します。

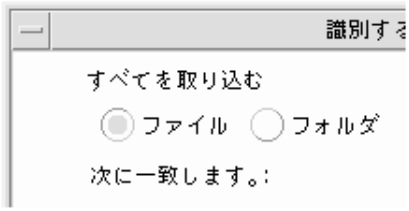


図 11–6　データ型にファイルまたはディレクトリの特徴を指定する

9. データ型の分類が名前に基づく場合は、**[**ファイル名のパターン**]** チェック・ボックスを選択して、テキスト・フィールドに入力します。

図 11–7　データ型にファイル名の特徴を指定する

ワイルドカードとして * と ? を使用できます。

* — すべての文字シーケンスに一致します。

? — 単一の文字すべてに一致します。

10. データ型の分類がアクセス権に基づく場合は、**[**アクセス権のパターン**]** チェック・ボックスを選択して、データ型のアクセス権を選択します。

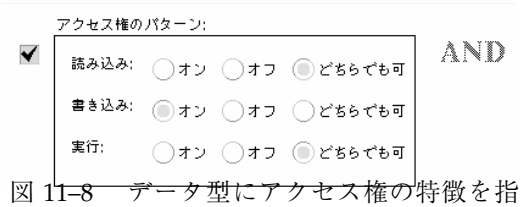


図 11–8　データ型にアクセス権の特徴を指定する

オン — 指定したアクセス権をファイルに持たせる

オフ — 指定したアクセス権をファイルに持たせない

どちらでも可 — 指定したアクセス権は関係ない

11. データ型の分類が内容に基づく場合は、**[**内容**]** チェック・ボックスを選択して、要求情報 **(**検索する **[**パターン**]** と内容の **[**タイプ**])** を提供します。オプションで、検索の開始バイトの位置を指定できます。

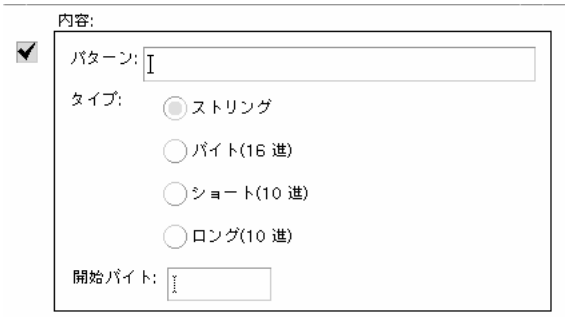


図 11–9　データ型に内容の特徴を指定する

注 - 内容に基づくデータ型の分類は、システム性能に影響を及ぼすことがあります。

---

12. [了解] をクリックして、[識別する特性] ダイアログ・ボックスを閉じます。

特徴は、次のコーディングを使用して [識別する特性] フィールドに表示されます。

d — ディレクトリ

r — 読み取り権を持つファイル

w — 書き込み権を持つファイル

x — 実行権を持つファイル

! — 論理演算子 NOT

& — 論理演算子 AND

13. [データ型アイコンのヘルプ・テキスト] テキスト・フィールドに、データ・ファイルのヘルプ・テキストを入力します。

14. [データ型アイコン] コントロールを使用して、アプリケーションにアイコンを指定します。最初は、デフォルト・アイコンが表示されます。

- 別の既存のアイコンを選択するには、[アイコンセット検索] をクリックして、[アイコンセット検索] ダイアログ・ボックスを表示します。詳細は、183ページの「アイコンを指定するための [アイコンセット検索] ダイアログ・ボックスの使用」を参照してください。

- 新しいアイコンを作成するには、[アイコン編集] をクリックして [アイコン・エディタ] を実行します。

15. **[**データ型を開くコマンド**]** テキスト・フィールドのコマンドを確認します。このコマンドは、データ・ファイルをダブルクリックしたときに実行されるコマンドです。

16. **(**省略可能**)** アプリケーションが、コマンド行からデータ・ファイルを印刷するための印刷コマンドを提供する場合は、そのコマンドを **[**データ型を印刷するコマンド**]** テキスト・フィールドに入力します。ファイル引き数に構文 *$n* を使用します。

17. 次のいずれかを実行して、データ型定義を保存します。
    - [了解] をクリックしてデータ型を保存し、[データ型追加] ダイアログ・ボックスを終了します。
    - [適用] をクリックして、[データ型追加] ダイアログ・ボックスを終了せずにデータ型を保存します。この場合、ただちにアクションの次のデータ型を定義できます。

アイコンを指定するための **[**アイコンセット検索**]** ダイアログ・ボックスの使用

[アイコンセット検索] ダイアログ・ボックスは、[アクション作成] メイン・ウィンドウまたは [データ型追加] ウィンドウで [アイコンセット検索] をクリックすると表示されます。アクションまたはデータ型に使用するアイコンを指定するために、このダイアログ・ボックスを使用します。

図 11–10　[アイコンセット検索] ダイアログ・ボックス

[アイコンセット検索] ダイアログ・ボックスにより、次の位置にあるアイコン・イメージ・ファイルのセットを指定できます。

- アイコン検索パスのディレクトリ。[アイコン・フォルダ] リストには、アイコン検索パスのすべてのディレクトリが載っています。
- `dtappintegrate` を使用してデスクトップに統合される登録パッケージ。これらのアイコンはアイコン検索パスのディレクトリにはありません が、`dtappintegrate` によって、そのディレクトリに配置されます。

注 - アクション作成ツールを使って作成されたアクションとデータ型の定義には、アイコン・ファイルのベース名が書いてあります (ファイル名から、サイズとファイルタイプを表すファイル名接尾辞を引いたもの)。アクション作成ツールで作成されたアクションとデータ型のアイコンは、最終的にはアイコン検索パスのディレクトリ上に配置されなければなりません。

アイコン検索パスにあるアイコンのセットを指定するには

1. **[アイコンセット検索]** ダイアログ・ボックスの **[アイコン・フォルダ]** リストで、アイコンを含むフォルダ・パスをダブルクリックします。

   [アイコン・ファイル] リストは、そのフォルダのすべてのアイコン・ファイルを表示します。

2. **[アイコン・ファイル]** リストで、使用するアイコンをクリックします。

   これで、アイコン・ファイルのベース名が [アイコン・ファイル名を入力] テキスト・フィールドに入ります。

3. **[了解]** をクリックします。

登録パッケージにアイコンを指定するには

システム管理者またはプログラマが登録パッケージを作成している場合、アイコン・イメージ・ファイルは当初、登録パッケージの次のディレクトリにあります。

app_root/dt/appconfig/icons/*language*

dtappintegrate で登録した後、アイコン・ファイルはアイコン検索パス上の /etc/dt/appconfig/icons/*language* にコピーされます。

次の手順で、登録パッケージの一部であるアイコンを指定します。

1. **[アイコンセット検索]** ダイアログ・ボックスの **[アイコン・ファイル名を入力]** テキスト・フィールドに、アイコン・ファイルのベース名を入力します。

2. **[了解]** をクリックします。

   アクション作成ツールは、それらのアイコンがアイコン検索パスのディレクトリに見つからないことを知らせるダイアログ・ボックスを表示します。

3. 表示される情報ダイアログ・ボックスで、**[名前を変更しない]** を選択します。

第 12 章

手入力によるアクションの作成

アクションを作成するには、次の 2 つの方法があります。

- アクション作成デスクトップ・アプリケーションを使用する
- 手入力でアクション定義を作成する

手入力でアクションを作成するには、データベース・ファイルを編集する必要があります。この章では、アクション定義を手入力で作成する方法について説明します。

- アクションの概要については、第 10 章を参照してください。
- アクション作成ツールの使用の詳細は、第 11 章を参照してください。

- アクション定義の参照情報については、dtactionfile(4) のマニュアル・ページを参照してください。

手入力でアクションを作成しなければならない理由

アクションには次の 3 つの基本型があります。

- COMMAND
- MAP
- TT_MSG

アクション作成ツールは、COMMAND アクションと MAP アクション型を作成するように設計されています。すべての TT_MSG アクションは、手動で作成しなければなりません。

詳細は、172ページの「アクション作成ツールの制限」を参照してください。

COMMAND アクション

「コマンド・アクション」は、アプリケーションやユーティリティを起動するコマンド、シェル・スクリプト、またはオペレーティング・システム・コマンドを実行します。アクションの定義には、実行されるコマンド (EXEC_STRING) も含まれます。

アクション作成ツールは、コマンド・アクションの最も一般的な型を作成するのに使用できます。しかし、アクションを手動で作成しなければならない状況のときもあります。たとえば、アクションが次のものを指定する場合には、手動で COMMAND アクションを作成しなければなりません。

- 各引き数に対して異なるプロンプトをもつ複数のファイル引き数
- アクション呼び出し — 他のアクションを呼び出すアクションの機能
- 引き数の数に依存する動作 — 異なる数のファイル引き数に対してさまざまな動作を行うアクションを作成する機能
- リモート実行ホスト — アクション定義を持っているシステム以外のシステムにあるアプリケーションを実行する機能
- ユーザの変更 — 異なるユーザとしてアクションを実行する機能 (たとえば、ルート・パスワードをプロンプトしてからルートとして実行する)

MAP アクション

「マップ・アクション」は、コマンドや ToolTalk メッセージを直接指定すること以外の、別のアクションに「マップされる」アクションです。

マッピングは、アクションの代替名を指定する機能を提供します。たとえば、[アイコンエディタ] という名前の組み込みコマンド・アクションは、アイコン・エディタを起動します。データベースにも [開く] アクションが含まれていますが、(ARG_TYPE フィールドによって) 定義はビットマップ・ファイルとピックスマップ・ファイルに制限され、 [アイコンエディタ] アクションにマップされます。これによりユーザは、ファイル・マネージャのビットマップ・ファイルまたはピックスマップ・ファイルを選択してから [選択] メニューより [開く] を選択することにより、アイコン・エディタを起動できます。

アクション作成ツールは、[開く] アクションと [印刷] アクションの制限されたマッピングを提供します。その他のすべてのマップ・アクションは、手動で作成しなければなりません。

TT_MSG (ToolTalk メッセージ) アクション

TT_MSG アクションは、ToolTalk メッセージを送信します。すべての TT_MSG アクションは、手動で作成しなければなりません。

手入力によるアクションの作成: 一般的な手順

この節では、アクション定義の構成ファイルを作成する方法について説明します。

アクションの構成ファイル

アクション定義が入っている構成ファイルは、次の要件を満たしていなければなりません。

- ファイルは、 *name*.dt という命名規則を使用しなければなりません。
- ファイルは、データベース (アクションとデータ型) 検索パス上になければなりません。デフォルトの検索パスは次のとおりです。

  個人用アクション — *HomeDirectory*/.dt/types

システム共通アクション — /etc/dt/appconfig/types/*language*

組み込みアクション — /usr/dt/appconfig/types/*language*。ただし、このディレクトリは使用しないでください。

アクションおよびデータ型検索パスの変更の詳細は、145ページの「検索パスの値の設定」を参照してください。

▼ 手入力でアクションを作成するには

1. 既存のデータベース・ファイルを開くか、新規のデータベース・ファイルを作成します。

   詳細は、189ページの「アクションの構成ファイル」を参照してください。

2. 次の構文を使用してアクション定義を作成します。

```
ACTION action_name
    {
      TYPE   action_type
      action_field
      ...
    }
```

   action_name — アクションを実行するのに使用する名前

   action_type — COMMAND (デフォルト)、MAP、または TT_MSG

   action_field — この型のアクションの必須または省略可能なフィールドの 1 つ。すべてのフィールドは、キーワードと値から成ります。ほとんどのアクション・フィールドについて、この章で説明します。

   詳細は、dtactionfile(4) のマニュアル・ページを参照してください。

3. ファイルを保存します。

4. アクション・アイコンが一意のイメージを持つようにするには、アクションのアイコンを作成します。アイコンのデフォルトの位置は次のとおりです。

   - 個人用アイコン: *HomeDirectory*/.dt/icons
   - システム共通アイコン: /etc/dt/appconfig/icons/*language*

     デフォルトの *language* は C です。

詳細は、195ページの「アクションが使用するアイコン・イメージの指定」を参照してください。

5. **[**デスクトップツール**]** アプリケーション・グループにある **[**アクションの再読込み**]** をダブルクリックします。

6. アクションのアクション・ファイルを作成します。アクション・ファイルは、アクションを表すアイコンをファイル・マネージャまたはアプリケーション・マネージャに作成します **(**アプリケーションを起動するようにアクションが書かれている場合、アイコンは「アプリケーション・アイコン」と呼ばれます**)**。

アクション・ファイルを作成するには、*action_name* と同じ名前の実行可能ファイルを作成します。書き込み権を持っているディレクトリに、そのファイルを置くことができます。アクション・ファイルは、好きな数だけ作成できます。

COMMAND アクションの作成例

次の手順は、リモート・システム AppServerA にある FAX アプリケーションを起動する個人用アクションを作成します。FAX アプリケーションを起動するコマンドは、次のとおりです。

`/usr/fax/bin/faxcompose` [*filename*]

1. ファイル ***HomeDirectory***`/.dt/types/Fax.dt` を作成します。

2. 次のアクション定義をファイルに記述します。

```
ACTION FaxComposer
    {
      TYPE      COMMAND
      ICON      fax
      WINDOW_TYPE   NO_STDIO
      EXEC_STRING   /usr/fax/bin/faxcompose -c %Arg_1%
      EXEC_HOST        AppServerA
      DESCRIPTION     Runs the fax composer
    }
```

`WINDOW_TYPE` フィールドと `EXEC_STRING` フィールドは、アクションの動作を説明します。

`WINDOW_TYPE` — `NO_STDIO` キーワードは、アクションが端末エミュレータ・ウィンドウで実行する必要がないように指定します。

詳細は、206ページの「アクションのウィンドウ・サポートの指定」を参照してください。

EXEC_STRING — 構文 %Arg_1% は、ドロップされたファイルを受け取ります。アクション・アイコンをダブルクリックすると、アクションは空の FAX 作成ウィンドウを開きます。

詳細は、199ページの「COMMAND アクションの実行文字列の作成」を参照してください。

3. ファイルを保存します。

4. アイコン・エディタを使用して、*HomeDirectory*/.dt/icons ディレクトリに、次のアイコン・イメージ・ファイルを作成します。

   - fax.m.pm、サイズ 32x32 ピクセル
   - fax.t.pm、サイズ 16x16 ピクセル

5. **[**デスクトップツール**]** アプリケーション・グループにある **[**アクションの再読込み**]** をダブルクリックします。

6. 書き込み権を持っているディレクトリ **(**たとえば、ホーム・ディレクトリ**)** に FaxComposer という名前の実行可能ファイルを作成します。

MAP アクションの作成例

FAX 送信するほとんどのファイルがテキスト・エディタで作成され、データ型 TEXTFILE (ファイル名は *.txt) であるものとします。

次の手順は、データ型の [選択] メニューに Fax メニュー項目を追加します。

1. 前節で作成したファイル *HomeDirectory*/.dt/types/Fax.dt を開きます。

2. 次のマップ・アクション定義をファイルに追加します。

```
ACTION Fax
    {
      ARG_TYPE  TEXTFILE
      TYPE    MAP
      MAP_ACTION  FaxComposer
    }
```

3. ファイルを保存します。

4. **TEXTFILE** のデータ属性定義を /usr/dt/appconfig/types/*language*/dtpad.dt から新規ファイル *HomeDirectory*/.dt/types/textfile.dt にコピーします。**Fax** アクションを ACTIONS フィールドに追加します。

```
DATA_ATTRIBUTES TEXTFILE
    {
      ACTIONS   Open,Print,Fax
      ICON    Dtpenpd
      …
    }
```

5. ファイルを保存します。

6. アプリケーション・マネージャを開き、**[**デスクトップツール**]** アプリケーション・グループにある **[**アクションの再読込み**]** をダブルクリックします。

▼ アクションとデータ型データベースを再読み込みするには

新規または編集されたアクション定義を有効にするには、デスクトップはデータベースを再読み込みしなければなりません。次のいずれかを実行します。

- **[**デスクトップツール**]** アプリケーション・グループを開き、**[**アクションの再読込み**]** をダブルクリックします。

- 次のコマンドを実行します。

  ```
  dtaction ReloadActions
  ```

  ReloadActions は、アイコンのラベルが [アクションの再読込み] であるアクション名です。

アクション・データベースは、次の操作を実行する場合にも再読み込みされます。

- ログインする
- ワークスペース・マネージャを再起動する
- [ファイル] メニューから [保存] を選択して [アクション作成] ウィンドウにあるアクションを保存する

アクションのアクション・ファイル (アイコン) の作成

「アクション・ファイル」は、ファイル・マネージャまたはアプリケーション・マネージャにあるアクションを視覚的に表現するために作成されるファイルです。

図 12–1　アプリケーション・マネージャにある「アクション・ファイル」

アクション・ファイルのアイコンはアクションを表現するので、「アクション・アイコン」と呼ぶこともあります。基本となるアクションがアプリケーションを起動する場合、アクション・ファイル・アイコンを「アプリケーション・アイコン」と呼びます。

アクション・アイコンをダブルクリックすると、アクションを実行します。アクション・アイコンはドロップ領域にもなります。

アクション・ファイル (アクション・アイコン) を作成するには

- アクション名と同じ名前の実行可能ファイルを作成します。ファイルの内容は関係ありません。

たとえば、次のようなアクション定義の場合、アクション・ファイルは MyFavoriteApp という名前の実行可能ファイルになります。

```
ACTION  MyFavoriteApp
 {
  EXEC_STRING    Mfa -file %Arg_1%
  DESCRIPTION    Runs MyFavoriteApp
  ICON      Mfapp
```

```
}
```

ファイル・マネージャとアプリケーション・マネージャでは、MyFavoriteApp ファイルはアイコン・イメージ Mfapp.*size*.*type* を使用します。MyFavoriteApp のアイコンをダブルクリックすると、アクションの実行文字列を実行し、アイコンのアイテムヘルプは DESCRIPTION フィールド (Runs MyFavoriteApp) の内容になります。

アクション・ラベル

アクション定義に LABEL フィールドが含まれている場合、ファイル・マネージャとアプリケーション・マネージャでは、ファイル名 (*action_name*) ではなく、このフィールドの内容によってアクション・ファイルにラベルが付けられます。たとえば、アクション定義に次のものが含まれる場合、アクション・アイコンには Favorite Application というラベルが付けられます。

```
ACTION  MyFavoriteApp
 {
  LABEL      Favorite Application
  …
 }
```

アクションが使用するアイコン・イメージの指定

ICON フィールドを使用して、アクション用に作成されたアクション・アイコン用にファイル・マネージャとアプリケーション・マネージャで使用するアイコンを指定します。

アイコンを指定しない場合、システムはデフォルトのアクション・アイコン・イメージ・ファイル /usr/dt/appconfig/icons/*language*/Dtactn.* を使用します。

図 12–2　デフォルトのアクション・アイコン・イメージ

デフォルトのアクション・アイコンは、次のリソースを使用して変更できます。

```
*actionIcon:      icon_file_name
```

icon_file_name は、ベース名または絶対パスを指定できます。

ICON フィールドの値は、次のようになります。

- ベース・ファイル名

ベース・ファイル名は、アイコン・イメージが入っているファイル名から、サイズ (m と t) とイメージ型 (bm と pm) を表すファイル拡張子を除いたものです。たとえば、ファイル名が GameIcon.m.pm と GameIcon.t.pm の場合は、GameIcon を使用します。

ベース・ファイル名を使用する場合、アイコン・ファイルは次のアイコン検索パスにあるディレクトリになければなりません。

- 個人用アイコン: *HomeDirectory*/.dt/icons
- システム共通アイコン: /etc/dt/appconfig/icons/*language*

- アイコン・ファイルへの絶対パス。完全なファイル名も含まれます。

  アイコン・ファイルがアイコン検索パス上にない場合だけ絶対パスを使用してください。たとえば、アイコン・ファイル GameIcon.m.pm がディレクトリ /doc/projects にある場合、アイコン検索パスにはないので、ICON フィールドの値は /doc/projects/GameIcon.m.pm になります。

  作成しなければならないアイコンのサイズと、それに対応するファイル名を表 12–1 に示します。

表 12–1 アイコン名とアクション・アイコンのサイズ

サイズ (ピクセル単位)	ビットマップ名	ピックスマップ名
48x48	*name*.l.bm	*name*.l.pm
32x32	*name*.m.bm	*name*.m.pm
16x16	*name*.t.bm	*name*.t.pm

▼ 既存のアクション定義を変更するには

システムのどの使用可能なアクション (組み込みアクションも含む) も変更できます。

注 - 組み込みアクション・データベースを変更する際には注意してください。組み込みアクションは、デスクトップ・アプリケーション上で動作するように設計されています。

---

1. **変更するアクション定義を検出します。**

アクション定義のデフォルトの位置は、次のとおりです。

- 組み込みアクション: /usr/dt/appconfig/types/*language*
- システム共通アクション: /etc/dt/appconfig/types/*language*
- 個人用アクション: *HomeDirectory*/.dt/types

システム位置が追加されている可能性もあります。アクション用にシステムが使用している位置のリストを参照するには、次のコマンドを実行します。

```
dtsearchpath -v
```

システムは、DTDATABASESEARCHPATH の下に表示されるディレクトリを使用します。

2. 必要に応じて、アクション定義のテキストを、次のいずれかのディレクトリにある新規または既存のファイルにコピーします。

   - システム共通アクション: /etc/dt/appconfig/types/*language*
   - 個人用アクション: *HomeDirectory*/.dt/types

   /usr/dt/appconfig/types/*language* ディレクトリにあるファイルは編集してはいけませんので、組み込みアクションをコピーしなければなりません。

3. アクション定義を編集します。

4. 編集が終わったら、ファイルを保存します。

5. **[**デスクトップツール**]** アプリケーション・グループにある **[**アクションの再読込み**]** をダブルクリックします。

アクション定義における優先順位

アクションを呼び出すと、システムはデータベースで一致するアクション名を検索します。その名前に対して 2 つ以上のアクションが存在する場合、システムはどちらを使用するかを決定するために優先規則を使用します。

- 他の優先規則が適用されない場合、優先順位は定義の位置に基づきます。次のリストは、優先度の高い順に並べてあります。

  - 個人用アクション (*HomeDirectory*/.dt/types)
  - システム共通のローカル・アクション (/etc/dt/appconfig/types/*language*)

- システム共通のリモート・アクション (*hostname*:/etc/dt/appconfig/types/*language*)。検索されたリモート・ホストは、アプリケーション検索パス上にあります。
- 組み込みアクション (/usr/dt/appconfig/types/*language*)

- 指定のディレクトリ内で、*.dt ファイルはアルファベット順に読み込まれます。
- ARG_CLASS、ARG_TYPE、ARG_MODE、または ARG_COUNT によって制限されたアクションは、制限されていないアクションよりも優先されます (これらの 4 つのフィールドのデフォルト値は * です)。

  2 つ以上の制限が適用される場合、優先順位の高い順に並べると次のようになります。

  - ARG_CLASS
  - ARG_TYPE
  - ARG_MODE
  - ARG_COUNT

  2 つ以上の制限された ARG_COUNT が存在する場合、優先順位の高い順に並べると次のようになります。

  - 特定の整数値 *n*
  - *<n*
  - *>n*
  - *

たとえば、次のアクション定義の一部分について考えてみます。

```
ACTION EditGraphics
 # EditGraphics-1
 {
  ARG_TYPE    XWD
  …
 }


ACTION EditGraphics
 # EditGraphics-2
 {
  ARG_COUNT      0
  …
 }


ACTION EditGraphics
 # EditGraphics-3
```

```
{
 ARG_TYPE    *
    …
}
```

EditGraphics アクション・アイコンをダブルクリックすると、引き数が指定されず ARG_COUNT 0 が優先されるので、EditGraphics-2 を起動します。XWD 型ファイル引き数を指定すると XWD ARG_TYPE が指定されるので、EditGraphics-1 が使用されます。EditGraphics-3 は、その他のファイル引き数すべてに対して使用されます。

COMMAND アクションの実行文字列の作成

COMMAND アクションには、必ず ACTION と EXEC_STRING の 2 つのフィールドが必要です。

```
ACTION action_name
 {
     EXEC_STRING execution_string
 }
```

実行文字列は、COMMAND アクション定義の最も重要な部分です。この文字列は、[端末エミュレータ] ウィンドウで実行するコマンド行と類似の構文を使用しますが、ファイルと文字列の引き数を処理するための追加構文も含みます。

実行文字列の一般的な機能

実行文字列には、次のものが含まれます。

- ファイル引き数と非ファイル引き数
- シェル構文
- 実行可能ファイルの絶対パスまたは名前

アクション引き数

引き数は、コマンドまたはアプリケーションを適切に実行するのに必要な情報です。たとえば、テキスト・エディタでファイルを開くのに使用できるコマンド行について考えてみます。

```
dtpad filename
```

このコマンドでは、*filename* は dtpad コマンドのファイル引き数です。

アプリケーションやコマンドのように、アクションは引き数を持つことができます。COMMAND アクションが使用できるデータ型は、次の 2 つです。

- ファイル
- 文字列データ

実行文字列でのシェルの使用

実行文字列は、シェルを介してではなく直接実行されます。しかし、実行文字列でシェルを明示的に呼び出すことができます。

たとえば、次のようになります。

```
EXEC_STRING    \
   /bin/sh -c \
   'tar -tvf %(File)Arg_1% 2>&1 | \${PAGER:-more};\
   echo "\\n*** Select Close from the Window menu to close ***"'
```

実行可能ファイルの名前または絶対パス

アプリケーションが PATH 変数にリストされているディレクトリにある場合は、単純な実行可能ファイル名を使用できます。アプリケーションが他にある場合は、実行可能ファイルへの絶対パスを使用しなければなりません。

引き数を使用しないアクションの作成

コマンド行からアプリケーションを起動するのに使用する EXEC_STRING に対しても同じ構文を使用します。

例

- 次の実行文字列は、X クライアント xcutsel を起動するアクションの一部です。

  ```
  EXEC_STRING xcutsel
  ```

- 次の実行文字列は、デジタル・クロックでクライアント xclock を起動します。コマンド行にはコマンド行オプションが含まれていますが、引き数は必要ありません。

  ```
  EXEC_STRING xclock -digital
  ```

ドロップされたファイルを受け取るアクションの作成

ファイル引き数に対して、次のいずれかの構文を使用します。

```
%Arg_n%

%(File)Arg_n%
```

Arg_*n* に指定された引き数は (デフォルトでは) ファイルと見なされるので、(File) はオプションになります。(%(String)Arg_*n*% 構文の使用方法については、202ページの「文字列としてのファイル引き数の解釈」を参照してください。)

この構文により、アクション・アイコンにデータ・ファイル・オブジェクトをドロップし、そのファイル引き数でアクションを起動できます。コマンド行で *n* 番目の引き数を置き換えます。ローカル・ファイルまたはリモート・ファイルのどちらも使用できます。

例

- 次の実行文字列は、-load パラメータとしてドロップされたファイルを使用し、次のように wc -w を実行します。

```
EXEC_STRING wc -w %Arg_1%
```

- 次の例は、ディレクトリ引き数によってのみ動作するアクションのための定義の一部を示します。ディレクトリがアクション・アイコンにドロップされると、アクションは読み取り権および書き込み権の両方を持つディレクトリにあるすべてのファイルのリストを表示します。

```
ACTION  List_Writable_Files
    {
      ARG_TYPE     FOLDER
      EXEC_STRING /bin/sh -c 's -l %Arg_1% | grep rw-'
      …
    }
```

ファイル引き数を要求するアクションの作成

ファイル引き数に対して、次の構文を使用します。

```
%(File)"prompt"%
```

この構文は、アクション・アイコンをダブルクリックしたときに、ファイル名の要求を表示するアクションを作成します。

たとえば次の実行文字列は、wc -w コマンドのファイル引き数を要求するダイアログ・ボックスを表示します。

```
EXEC_STRING wc -w %(File)"Count words in file:"%
```

ドロップされたファイルを受け取るかファイルを要求するアクションの作成

ファイル引き数に対して、次のいずれかの構文を使用します。

```
%Arg_n"prompt"%

%(File)Arg_n"prompt"%
```

この構文は、次の動作を行うアクションを作成します。

- ファイル引き数としてドロップされたファイルを受け取る
- アクション・アイコンをダブルクリックしたときに、ファイル名を要求するダイアログ・ボックスを表示する

たとえば次の実行文字列は、ドロップされたファイルで lp -oraw を実行します。アイコンをダブルクリックするとアクションが起動する場合、ダイアログ・ボックスが表示され、ファイル名を要求します。

```
EXEC_STRING lp -oraw %Arg_1"File to print:"%
```

非ファイル引き数を要求するアクションの作成

非ファイル・パラメータに対して、次のいずれかの構文を使用します。

```
%"prompt"%

%(String)"prompt"%
```

引用符で囲まれたテキストは、デフォルトでは文字列データとして解釈されるので、(String) はオプションになります。この構文は、非ファイル・データを要求するダイアログ・ボックスを表示します。ファイル名を要求するときは、この構文は使用しないでください。

たとえば、次の実行文字列は xwd コマンドを実行し、各ピクセルに追加される値を要求します。

```
EXEC_STRING xwd -add %"Add value:"% -out %Arg_1"Filename:"%
```

文字列としてのファイル引き数の解釈

引き数に対して次の構文を使用します。

```
%(String)Arg_n%
```

たとえば次の実行文字列は、コマンド lp –t*banner filename* を使用して、ファイル名が入っているバナーとファイルを一緒に印刷します。

```
EXEC_STRING lp -t%(String)Arg_1% %(File)Arg_1"File to print:"%
```

アクションへのシェル機能の提供

次のように実行文字列にシェルを指定します。

```
/bin/sh -c 'command'
/bin/ksh -c 'command'
/bin/csh -c 'command'
```

例

- 次の実行文字列は、シェルのパイプを使用するアクションを示します。

  ```
  EXEC_STRING /bin/sh -c 'ps | lp'
  ```

- さらに複雑な実行文字列で、シェルの処理を必要とし、ファイル引き数を受け取ります。

  ```
  EXEC_STRING /bin/sh -c 'tbl %Arg_1"Man Page:"% | troff -man'
  ```

- 次の実行文字列では、引き数は圧縮ファイルである必要があります。アクションはファイルを圧縮解除し、lp -oraw を使用してそのファイルを印刷します。

  ```
  EXEC_STRING    /bin/sh -c 'cat %Arg_1 "File to print:"% | \
       uncompress | lp -oraw'
  ```

- 次の実行文字列は、シェル・スクリプトを起動します。

  ```
  EXEC_STRING /usr/local/bin/StartGnuClient
  ```

複数のファイル引き数を処理する COMMAND アクションの作成

アクションが複数のファイル引き数を処理するには、次の 3 つの方法があります。

- アクションは繰り返し (各引き数に対しては 1 回ずつ) 実行できます。EXEC_STRING に単一のファイル引き数があり、アクション・アイコンに複数のファイルをドロップすることにより複数のファイル引き数が指定される場合、アクションはファイル引き数ごとに別々に実行されます。

  たとえば、複数のファイル引き数が次のアクション定義に対して指定される場合、DisplayScreenImage アクションは繰り返し実行されます。

```
ACTION DisplayScreenImage
     {
       EXEC_STRING     xwud -in %Arg_1%
       …
     }
```

- アクションは、2 つ以上の交換不可能なファイル引き数を使用できます。たとえば次の例では、特定の順番で並んでいる 2 つの一意のファイルが必要です。

  `xsetroot -cursor` *cursorfile maskfile*

- アクションは、各ファイル引き数で連続して同じコマンドを実行できます。たとえば次の例では、1 回の印刷ジョブで 1 つ以上のファイルを印刷します。

  `pr` *file* [*file* …]

交換不可能な引き数を処理するアクションの作成

次の構文規約のいずれか 1 つを使用します。

- アクションにファイル名をプロンプトさせる場合、ファイル引き数ごとに次の構文を使用します。

  `%(File)"`*prompt*`"%`

  引き数ごとに異なる *prompt* 文字列を使用します。

  たとえば、次の実行文字列は 2 つのファイルをプロンプトします。

```
EXEC_STRING    xsetroot -cursor %(File)"Cursor bitmap:"% \
         %(File)"Mask bitmap:"%
```

- ドロップされたファイルを受け取るには、各ファイル引き数に対して次の構文を使用します。

  `%Arg_`*n*`%`

  この場合、引き数ごとに異なる *n* 値を使用します。たとえば次のようになります。

```
EXEC_STRING    diff %Arg_1% %Arg_2%
```

交換可能なファイル引き数を処理するアクションの作成

次の構文規約のいずれか 1 つを使用します。

- ドロップされたファイルを受け取り、*command file 1 file 2 …* という形式でコマンドを発行するアクションを作成するには、ファイル引き数に対して次の構文を使用します。

  `%Args%`

- 複数のドロップされたファイルを受け取る、またはダブルクリックしたときに単一ファイルのプロンプトを表示するアクションを作成するには、ファイル引き数に対して次の構文を使用します。

  ```
  %Arg_1"prompt"% %Args%
  ```

  アクションは、*command file 1 file 2 …* という形式でコマンドを発行します。

例

- 次の実行文字列は、複数のファイル引き数をもつアクションを作成します。

  ```
  EXEC_STRING pr %Args%
  ```

  この実行文字列は次のコマンド行を実行します。

  ```
  pr file 1 file 2
  ```

- 次の実行文字列は、前の例と似たアクションを作成しますが、この例のアクションをダブルクリックするとプロンプトを表示します (ファイル引き数はありません)。

  ```
  EXEC_STRING  pr %Arg_1"File(s) to print:"% %Args%
  ```

複数のドロップされたファイルを処理するアクションの作成

複数のドロップされたファイル引き数を受け取り、*command file 1 file 2 …* という形式のコマンド行を実行するには、次の構文を使用します。

```
%Args%
```

例

- 次の実行文字列は、複数のファイルに対して Checkout という名前のスクリプトを実行します。

  ```
  EXEC_STRING    /usr/local/bin/Checkout \
          %Arg_1"Check out what file?"% %Args%
  ```

- 次の実行文字列は、複数のファイルに対して `lp -oraw` を実行します。

  ```
  EXEC_STRING    lp -oraw %Arg_1"File to print:"% %Args%
  ```

COMMAND アクションのウィンドウ・サポートと端末エミュレータ

COMMAND アクションがデスクトップ上でウィンドウをサポートするには、次の方法があります。

- アプリケーションがそれ自身のウィンドウを持っている場合、追加のウィンドウ・サポートを提供しないようにアクションに書き込むことができます。このオプションは、直接ユーザ入力を必要とせず、アクションが出力を持っていないコマンドを実行するときにも使用されます。
- アプリケーションを端末エミュレータウィンドウで実行しなければならない場合、ウィンドウを開いてからアプリケーションを実行するようにアクションに書き込むことができます。この場合、いくつかの端末オプションがあります。

アクションのウィンドウ・サポートの指定

WINDOW_TYPE フィールドを使用して、アクションが必要とするウィンドウ・サポートの型を表 12–2 のように指定します。

表 12–2 WINDOW_TYPE フィールドと提供されているウィンドウ・サポート

WINDOW_TYPE	指定されているウィンドウ・サポート
NO_STDIO	指定されません。アプリケーションがそれ自身のウィンドウを持っているか、コマンドが表示できる出力を持っていない場合は、NO_STDIO を使用します。
PERM_TERMINAL	常時端末エミュレータ・ウィンドウ。アクションは端末ウィンドウを開き、明示的に閉じるまで開いたままにします。ウィンドウにデータを入力できます。入力を行い、出力を作成し、その後終了するコマンド (たとえば `ls` *directory* など) で使用します。
TERMINAL	一時端末エミュレータ・ウィンドウ。アクションは端末ウィンドウを開き、コマンドが完了するとすぐに閉じます。全画面コマンド (たとえば `vi` など) で使用します。

端末エミュレータのコマンド行オプションの指定

アクション定義にある TERM_OPTS フィールドを使用して、端末エミュレータのコマンド行オプションを指定します。

たとえば、次のアクションは実行ホストをプロンプトします。

```
ACTION OpenTermOnSystemUserChooses
 {
  WINDOW_TYPE     PERM_TERMINAL
  EXEC_HOST        %(String)"Remote terminal on:"%
  TERM_OPTS        -title %(String)"Window title:"%
  EXEC_STRING      $SHELL
 }
```

他のデフォルト端末エミュレータの指定

アクションが使用するデフォルトの端末エミュレータは dtterm です。これを別の端末エミュレータに変更できます。デフォルトの端末エミュレータは、アクションが使用する端末エミュレータを明示的に指定しないときに使用されます。

アクションが使用する端末エミュレータは、次のコマンド行オプションを持っていなければなりません。

- –title *window_title*
- –e *command*

次の 2 つのリソースは、アクションが使用するデフォルトの端末エミュレータを決定します。

- localTerminal リソースは、ローカル・アプリケーションが使用する端末エミュレータを指定します。

  *localTerminal: *terminal*

  たとえば、次のようになります。

  *localTerminal: xterm

- remoteTerminal リソースは、リモート・アプリケーションが使用する端末エミュレータを指定します。

  *remoteTerminal: *host*:*terminal* [,*host*:*terminal*...]

  たとえば、次のようになります。

  *remoteTerminal: sysibm1:/usr/bin/xterm,syshp2:/usr/bin/yterm

特定の引き数へのアクションの制限

特定の型の引き数にアクションを制限すると、アクションが改良されます。たとえば、PostScript ファイル引き数に対してのみ PostScript ファイルのビューアを呼び出すようにアクションを制限すると便利です。つまり制限があると、PostScript ファイル以外を指定した場合、アクションはエラー・ダイアログを返します。

次のことに基づいてアクションを制限できます。

- ファイル引き数のデータ型
- ファイル引き数の数 — たとえば、引き数なしと 1 つ以上の引き数を指定する場合とでは、アクション・アイコンのドロップ&ダブルクリック動作が異なります。
- 引き数の読み取りまたは書き込みモード

指定されたデータ型へのアクションの制限

ARG_TYPE フィールドを使用して、アクションが有効であるデータ型を指定します。データ属性名を使用します。

コンマでエントリを区切って、データ型のリストを入力できます。

たとえば、次のアクション定義は Gif データ型が作成されている場合の例です。

```
ACTION Open_Gif
 {
  TYPE     COMMAND
  LABEL     "Display Gif"
  WINDOW_TYPE   NO_STDIO
  ARG_TYPE    Gif
  ICON     xgif
  DESCRIPTION   Displays gif files
  EXEC_STRING   xgif
 }
```

引き数の数に基づいたアクションの制限

ARG_COUNT フィールドを使用して、アクションが受け入れることができる引き数の数を指定します。有効な値は次のとおりです。

* (デフォルト) — 任意の数の引き数。その他の値は * よりも優先されます

n — 任意の負でない値 (0 を含む)

>*n* — *n* よりも大きい引き数

<n — *n* よりも小さい引き数

ARG_COUNT の使用方法の 1 つは、アイコンをダブルクリックするか、そのアイコンにファイルをドロップするかに応じて、異なるアクション・アイコン動作を提供することです。次の「異なるダブルクリック・アンド・ドロップ動作を提供するには」を参照してください。

▼ 異なるダブルクリック & ドロップ動作を提供するには

次の手順を使用して、ドロップされたファイルを受け取るアクションを作成しますが、アクション・アイコンをダブルクリックしてもファイルを要求しません。

1. ダブルクリック機能のためのアクション定義を作成します。

   ARG_COUNT フィールドを使用して 0 引き数を指定します。ドロップされた引き数を受け取らない EXEC_STRING の構文を使用します。

2. ドロップ機能のための 2 番目のアクション定義を作成します。

   ARG_COUNT フィールドを使用して >0 引き数を指定します。ドロップされたファイルを受け取る EXEC_STRING の構文を使用します。

   たとえば、次の 2 つのコマンド行は vedit という名前のエディタを起動するのに使用できるとします。

   - ファイル引き数を使用しないでエディタを起動するには、次のようにします。

     ```
     vedit
     ```

   - 読み専用ドキュメントとして開くファイル引き数でエディタを起動するには、次のようにします。

     ```
     vedit -R filename
     ```

次の 2 つのアクションは、Vedit という名前のアクションのためのドロップ & ダブルクリック機能を作成します。ARG_COUNT 0 は暗示的なドロップ機能定義の ARG_COUNT * よりも明確なので、データベースで一致するものを検索するときには、最初のアクションが優先されます。

```
# Double-click functionality
ACTION Vedit
 {
  TYPE       COMMAND
  ARG_COUNT     0
  WINDOW_TYPE     PERM_TERMINAL
```

```
  EXEC_STRING    vedit
 }

# Drop functionality
ACTION Vedit
 {
  TYPE      COMMAND
  WINDOW_TYPE    PERM_TERMINAL
  EXEC_STRING    vedit -R %Arg_1%
 }
```

引き数のモードに基づいたアクションの制限

ARG_MODE フィールドを使用して、引き数の読み取りまたは書き込みモードを指定します。有効な値は次のとおりです。

* (デフォルト) — 任意のモード

!w — 書き込み不可

w — 書き込み可能

リモート・システムでアプリケーションを実行するアクションの作成

アクションとリモート実行について説明する場合、次の 2 つの用語がよく使用されます。

データベース・ホスト — アクション定義が入っているシステム

実行ホスト — 実行可能ファイルを実行するシステム

ほとんどの場合、アクションとそのアプリケーションは同じシステム上にあります。つまり、アクションのデフォルトの実行ホストはデータベース・ホストであるため、特別な構文は必要ありません。

しかし、実行ホストがデータベース・ホストと異なる場合、アクション定義はどこで実行文字列を実行するか指定しなければなりません。

異なるシステム上にアクションとアプリケーションを配置するための機能は、デスクトップのクライアントサーバ・アーキテクチャの一部です。ネットワーク・アプリケーションの詳細は、128ページの「アプリケーション・サービスの管理」を参照してください。

リモート・アプリケーションを実行するアクションの作成

アクション定義 EXEC_HOST フィールドを使用して、アプリケーションの位置を指定します。

EXEC_HOST の有効な値は次のとおりです

%DatabaseHost% — アクションが定義されるホスト

%LocalHost% — アクションが呼び出されるホスト (「セッション・サーバ」)

%DisplayHost% — X サーバを実行中のホスト (X 端末では使用できません)

%SessionHost% — コントロールを行うログイン・マネージャが実行中のホスト

hostname — 名前付きホスト。アクションが 1 つの特定ホスト上に必ず呼び出される環境でこの値を使用します

%"*prompt*"% — アクションが呼び出されるたびにホスト名をプロンプトします。

デフォルト値は、%DatabaseHost% と %LocalHost% です。したがって、EXEC_HOST フィールドが削除されると、アクションはアクション定義が入っているホストで最初にコマンドの実行を試みます。これに失敗すると、アクションはセッション・サーバでコマンドの実行を試みます。

例

- このフィールドは、ホスト ddsyd を指定します。

```
EXEC_HOST  ddsyd
```

- このフィールドは、ホスト名をプロンプトします。

```
EXEC_HOST  %"Host containing application:"%
```

- このフィールドは、アクション定義が入っているホストでアクションがアプリケーションの実行を試みるように指定します。これに失敗すると、アクションはホスト ddsyd でアプリケーションの実行を試みます。

```
EXEC_HOST  %DatabaseHost%, ddsyd
```

アクションとデータ型定義での変数の使用

文字列変数と環境変数をアクションおよびデータ型定義ファイルに含むことができます。

アクションでの文字列変数の使用

文字列変数定義は、定義の位置からファイルの最後まで有効です。データベース用のグローバルな文字列変数はありません。

文字列変数と環境変数が同じ名前の場合は、文字列変数が優先されます。

文字列変数を定義するには

- 次の構文を使用します。

```
set variable_name=value
```

変数名に英数字と下線文字 (_) を使用できます。各変数定義は、別の行になければなりません。

たとえば次のようになります。

```
set Remote_Application_Server=sysapp
set Remote_File_Server=sysdata
```

文字列変数を参照するには

- 次の構文を使用します。

```
$[{]variable_name[}]
```

たとえば次のようになります。

```
EXEC-HOST   $Remote_Application_Server
CWD     /net/${Remote_File_Server}/doc/project
```

アクションとデータ型での環境変数の使用

- 次の構文を使用して環境変数を参照します。

```
$[{]variable[}]
```

データベースが読み込まれると、変数は拡張されます (その値に置き換わります)。文字列変数と環境変数が同じ名前の場合は、文字列変数が優先されます。

たとえば、次の実行文字列はログイン名が入っているバナーでファイルを印刷します。

```
EXEC-STRING lp -t$LOGNAME %(File)Arg_1%
```

コマンド行からのアクションの呼び出し

デスクトップは、コマンド行からアクションを実行するための dtaction コマンドを提供します。dtaction を使用して、次の部分からアクションを実行できます。

- スクリプト
- 他のアクション
- 端末エミュレータ・コマンド行

dtaction の構文

```
dtaction [-user user_name] [-execHost hostname] action_name [argument [argument]...]
```

–user *user_name* — 別のユーザとしてアクションを実行するための機能を提供します。dtaction が *user_name* 以外のユーザによって呼び出される場合、パスワードのプロンプトが表示されます。

–execHost *hostname* — COMMAND アクションの場合のみ。コマンドが実行されるホストを指定します。

argument — アクションに対する引き数。通常はファイル引き数です。

dtaction クライアントは、追加のコマンド行オプションを持っています。詳細は、dtaction(1) のマニュアル・ページを参照してください。

異なるアクションを実行するアクションの作成

アクションの EXEC_STRING にある dtaction を使用します。

たとえば、次のアクションは Spell という名前の組み込みアクション (アクションはアプリケーション・マネージャで「スペルチェック」というラベルが付けられます) を使用します。新規アクションは、テキスト・エディタと Spell アクションを実行し、別の端末エミュレータ・ウィンドウにスペルミスを表示します。

```
ACTION EditAndSpell
 {
  WINDOW_TYPE    NO_STDIO
  EXEC_STRING    /bin/sh -c 'dtaction Spell \
         %Arg_1"File:"%; dtpad %Arg_1%'
 }
```

別のユーザとして実行するアクションの作成

EXEC_STRING にある次の構文を使用します。

```
EXEC_STRING    dtaction -user user_name action_name [file_argument]
```

新規ユーザ (*user_name* は、次のいずれかの機構を介してディスプレイをシステムへアクセスしなければなりません。

- ログイン・ユーザの .Xauthority ファイルにある読み取り権
- xhost アクセス権

たとえば、次の 2 つのアクションは、root になるための機能と app-defaults ファイルを編集するための機能を提供します。

```
ACTION AppDefaults
 {
  WINDOW_TYPE    NO_STDIO
  EXEC_STRING    /usr/dt/bin/dtaction -user root \
         EditAppDefaults %Arg_1"File:"%
 }

ACTION EditAppDefaults
 {
  WINDOW_TYPE    TERMINAL
  EXEC_STRING    /bin/sh -c 'chmod +w %Arg_1%; \
         vi %Arg_1%; chmod -w %Arg_1%'
 }
```

ローカライズされたアクションの作成

データ型の検索パスには、言語に依存する位置も含まれています。デスクトップは、LANG の値を使用して、データ型定義が検索される位置を決定します。

ローカライズされたアクションの位置

ローカライズされたアクション定義は、アクション検索パスに応じて適切な言語に依存するディレクトリに置かなければなりません。

デフォルトの検索パスは次のとおりです。

- 個人用アクション: *HomeDirectory*/.dt/types
- システム共通アクション: /etc/dt/appconfig/types/*language*
- 組み込みアクション: /usr/dt/appconfig/types/*language*

▼ 既存のアクションをローカライズするには

1. 適切な言語に依存するディレクトリ (たとえば、/etc/dt/appconfig/types/japanese) にファイルを作成します。

2. アクション定義を言語に依存する構成ファイルにコピーします。

   たとえば、アクション定義を、

   app_root/dt/appconfig/types/C/*file*.dt

   から

   app_root/dt/appconfig/types/japanese/*newfile*.dt

   にコピーします。

3. LABEL フィールドを追加するか、既存の **LABEL** フィールドを変更します。

   LABEL *string*

   アプリケーション・マネージャとファイル・マネージャは、ラベル文字列を使用してアクションのアイコンを識別します。

4. アクション定義にある次のいずれかのフィールドをローカライズします。
   - ローカライズされたアイコンの場合: ICON
   - ローカライズされたアイテムヘルプの場合: DESCRIPTION

- ローカライズされたプロンプトの場合: EXEC_STRING にある引用符で囲まれたテキスト

ToolTalk アプリケーションのアクションの作成

注 - 次の情報は、ToolTalk メッセージをサポートするアプリケーションに対してのみ適用されます。

TT_MSG アクション型を使用して、ToolTalk メッセージを送信するアクションを作成します。

```
ACTION action_name
{
  TYPE  TT_MSG
  …
}
```

addressing フィールドと disposition フィールド

- ToolTalk addressing フィールドは、常に TT_PROCEDURE に設定されます。
- ToolTalk disposition フィールドは、静的メッセージ・パターンの指定がデフォルトの値です。

サポートされていないメッセージ

TT_MSG アクション型がサポートしていないものは、次のとおりです。

- ToolTalk オブジェクト指向メッセージ
- メッセージ内のコンテキスト引き数

TT_MSG アクションのキーワード

表 12–3 に、キーワードと TT_MSG アクションの使用法を示します。

表 12–3 TT_MSG アクション・キーワードと使用法

キーワード	使用法
TT_CLASS	ToolTalk class メッセージ・フィールドの値を定義します。
TT_SCOPE	ToolTalk scope メッセージ・フィールドの値を定義します。
TT_OPERATION	ToolTalk operation メッセージ・フィールドの値を定義します。
TT_FILE	ToolTalk file メッセージ・フィールドの値を定義します。
TT_ARGn_MODE	*n* 番目のメッセージ引き数の ToolTalk mode 属性の値を定義します。
TT_ARGn_VTYPE	*n* 番目のメッセージ引き数の ToolTalk vtype 属性の値を定義します。
TT_ARGn_VALUE	*n* 番目のメッセージ引き数の値を定義します。

第 13 章

手入力によるデータ型の作成

データ型定義は、次の 2 つの方法で作成できます。

- アクション作成ツールを使用する。

  アクション作成ツールの使用方法については、第 11 章で説明しています。

- 手入力でデータ型定義を作成する。

  手入力でデータ型を作成するには、データベース・ファイルを編集しなければなりません。

この章では、手入力によるデータ型定義の作成方法について説明します。

- データ型の概要については、第 10 章を参照してください。
- データ型定義のリファレンス情報については、`dtddsfile(4)` のマニュアル・ページを参照してください。

手入力でデータ型を作成しなければならない理由

データ型を手入力で作成すると、データ型定義の構文に構築されたすべての機能を使用できます。

次のデータ型の機能を使用する場合は、データ型を手入力で作成する必要があります。

- 位置 (パス) ベースのデータ型作成
- [開く] と [印刷] 以外のデータ型に関連するアクションを指定する機能
- 同じデータ型に対しての複数の名前、パターン、内容の基準。たとえば、`*.abc` または `*.def` という名前のファイルに基づくデータ型
- リンクベースのデータ型作成

データ型定義のコンポーネント: 基準と属性

データ型定義は、次の 2 つの異なるデータベース定義で構成されます。

- `DATA_ATTRIBUTES` 定義

  データ型の名前と、このデータ型のファイルの外観および動作を説明します。

- `DATA_CRITERIA` 定義

  型を作成するための基準を説明します。各基準定義は、基準を適用する `DATA_ATTRIBUTES` 定義を指定します。

各 `DATA_ATTRIBUTES` 定義に対して、少なくとも 1 つの `DATA_CRITERIA` 定義がなければなりません。`DATA_ATTRIBUTES` 定義は、関連する複数の `DATA_CRITERIA` を持つことができます。

たとえば、ファイル・マネージャでの PostScript ファイルの外観と動作を説明する PostScript ファイルの属性定義を作成できます。次に、PostScript データ型の 2 つの異なる基準を作成できます。1 つの基準はファイル名に基づき、もう 1 つはファイル内容に基づきます。

詳細は、227ページの「データ型のデータ基準の定義」を参照してください。

手入力によるデータ型の作成: 一般的な手順

この節では、データ型構成ファイルの作成方法を説明します。

データ型の構成ファイル

データ型定義を含む構成ファイルの要件は次のとおりです。

- ファイルが *name*.dt という命名規則を使用していること
- ファイルがデータベース検索パス上にあること。デフォルトの検索パスは次のとおりです。

  個人用データ型 — *HomeDirectory*/.dt/types

  システム共通データ型 — /etc/dt/appconfig/types/*language*

  組み込みデータ型 — /usr/dt/appconfig/types/*language*。このディレクトリは使用しないでください。

データベース検索パスの変更については、145ページの「検索パスの値の設定」を参照してください。

▼ データ型定義を作成するには

1. **既存のデータベース・ファイルを開くか、新しいファイルを作成します。**

   詳細は、221ページの「データ型の構成ファイル」を参照してください。

2. **次の構文を使って、データ型のデータ属性を定義します。**

```
DATA_ATTRIBUTES data_type_name
    {
      ICON         image_name
      DESCRIPTION  string
      attribute_field
      attribute_field
      ...
    }
```

   それぞれの意味は次のとおりです。

   data_type_name — このデータ型に指定する固有の名前

image_name — アイコン・ファイルのファイル名またはパス。ファイルのベース名を使用します。たとえば、myimage.m.pm と myimage.t.pm というアイコン・ファイルには、myimage を使用します。

attribute_field — データ型の外観と動作を定義するフィールド

string — 文字列。内容はデータ型のアイテムヘルプです。

詳細は、223ページの「パーソナル・アクションとデータ型の作成例」を参照してください。

3. 次の構文を使って、データ型のデータ基準を定義します。

```
DATA_CRITERIA criteria_name
    {
      DATA_ATTRIBUTES_NAME data_type_name
      criteria_field
      criteria_field
      ...
    }
```

それぞれの意味は次のとおりです。

criteria_name — この基準定義の固有の名前

data_type_name — DATA_ATTRIBUTES 定義で使用する名前

criteria_field — ファイルをこのデータ型に割り当てるための基準を定義するのに使用するフィールド

詳細は、227ページの「データ型のデータ基準の定義」を参照してください。

4. データベース・ファイルを保存します。

5. データ型のアイコンを作成します。

詳細は、224ページの「データ型に使用するアイコン・イメージの指定」を参照してください。

6. 必要に応じて、属性定義の ACTIONS フィールドにリストされたアクションを作成します。

7. **[**デスクトップツール**]** アプリケーション・グループの **[**アクションの再読込み**]** をダブルクリックし、データベースを再読み込みします。

パーソナル・アクションとデータ型の作成例

GIF 画像を表示する xgif というアプリケーションがシステムに含まれているとします。通常は、次のように入力してプログラムを実行します。

xgif *filename*

GIF 画像は、次の 2 つの方法で表示するものとします。

- GIF データ・ファイルをダブルクリックする
- データ・ファイルを選択し、[選択] メニューからアプリケーションを選択する

1. 新規ファイル ***HomeDirectory***/.dt/types/GifViewer.dt を開きます。

2. データ型定義を入力します。

```
DATA_ATTRIBUTES Gif
    {
      DESCRIPTION    Gif image file.
      ICON      GifIcon
      ACTIONS     View
    }

   DATA_CRITERIA Gif_Criteria
    {
      DATA_ATTRIBUTES_NAME   Gif
      NAME_PATTERN      *.gif
    }
```

3. **GifViewer** アクションのアクション定義を入力します。

```
ACTION GifViewer
    {
      EXEC_STRING     xgif %(File)Arg_1"Gif file to view:"
      WINDOW_TYPE     NO_STDIO
      DESCRIPTION     Double-click or drop a file to \
              start the Gif viewer.
    }
```

   定義に ICON フィールドは含まれないので、アクションはシステムのデフォルト・アイコンを使用します。

4. 次のマップ・アクションを入力し、**GifViewer** アクションを、データ型定義にリストされた **View** アクションに接続します。この表示アクションを **Gif** 型ファイルに制限するには ARG_TYPE フィールドを使用します。

```
ACTION View
    {
      ARG_TYPE    Gif
      TYPE     MAP
```

```
    MAP_ACTION   GifViewer
}
```

5. ファイルを保存します。

6. **[**デスクトップツール**]** アプリケーション・グループの **[**アクションの再読込み**]** をダブルクリックし、データベースを再読み込みします。

データ型のデータ属性の定義

DATA_ATTRIBUTES 定義は、データ型の外観と動作を定義します。データ型名を指定し、次の内容を指定する機能を提供します。

- ファイル・マネージャ・アイコン (ICON フィールド)
- [選択] メニューのダブルクリック動作と内容 (ACTIONS フィールド)
- データ型のアイテムヘルプ (DESCRIPTION フィールド)

データ型に使用するアイコン・イメージの指定

ICON フィールドを使用して、ファイル・マネージャで使用するアイコンを指定します。アイコン・イメージを指定しないと、ファイル・マネージャはラベルだけを表示します。

ICON フィールドの値は次のいずれかです。

- ベース・ファイル名

  ベース・ファイル名は、ファイル名からサイズに関する接尾辞拡張子 (m および t)と、イメージ型に関する接尾辞拡張子 (bm および pm) を除いたアイコン・イメージのファイル名です。たとえば、GameIcon.m.pm および GameIcon.t.pm という名前のファイルは、GameIcon を使用します。

  ベース・ファイル名を使用する場合、必ず次のアイコン検索パスのディレクトリにアイコン・ファイルを指定してください。

  - 個人用アイコン: *HomeDirectory*/.dt/icons
  - システム共通アイコン: /etc/dt/appconfig/icons/*language*

- 完全ファイル名を含むアイコン・ファイルの絶対パス

  アイコン・ファイルがアイコン検索パスにない場合のみ、絶対パスを使用してください。たとえば、アイコン・ファイル GameIcon.m.pm がアイコン検索パス上にないディレクトリ /doc/projects にある場合、ICON フィールドの値は /doc/projects/GameIcon.m.pm です。

作成するアイコンのサイズと、対応するファイル名を表 13–1 に示します。

表 13–1 データ型アイコンのアイコン名とサイズ

サイズ（ピクセル単位）	ビットマップ名	ピックスマップ名
32x32	*name*.m.bm	*name*.m.pm
16x16	*name*.t.bm	*name*.t.pm

データ型とアクションの関連付け

データ型をアクションに関連付けるには、次の 2 つの方法があります。

- DATA_ATTRIBUTES 定義の ACTIONS フィールドは、ファイル・マネージャの [選択] メニューに表示されるアクションをリストします。リストの最初のアクションは、デフォルトのダブルクリックアクションです。
- アクション定義の ARG_TYPE フィールドを使用して、指定したデータ型にアクションを制限できます。

たとえば次のデータ型定義は、*.rm という命名規則を使用して、システム管理者が作成した特殊な readme ファイルのデータ型を作成します。

```
DATA_ATTRIBUTES SysReadmeFile
 {
  ICON      SysReadMe
  ACTIONS     Open,Respond
 }

DATA_CRITERIA SysReadmeFileCriteria
 {
  NAME_PATTERN      *.rm
  DATA_ATTRIBUTES_NAME    SysReadmeFile
 }
```

特殊な Respond アクションは、ファイルに対して下記のように定義されます。このアクションは、テキスト・エディタで書き込み可能なファイルのコピーを開きます。ファイルを保存してテキスト・エディタを終了すると、ファイルはシステム管理者にメールで送信されます (メール・アドレスは sysadmin@utd です)。

```
ACTION Respond
 {
  ARG_TYPE    SysReadmeFile
  EXEC_STRING   /bin/sh -c 'cp %Arg_1% $HOME/readme.temp;\
        chmod +w $HOME/readme.temp;              \
        dtpad $HOME/readme.temp;                 \
        cat $HOME/readme.temp |                  \
        /usr/bin/mailx sysadmin@utd;             \
        rm $HOME/readme.temp'
  WINDOW_TYPE   NO_STDIO
 }
```

データ型に基づいてファイルを隠す

ファイルが非表示のデータ型の場合は、ファイル・マネージャには表示されません。

次のように DATA_ATTRIBUTES 定義の PROPERTIES フィールドを使用して、この型のオブジェクトを隠すよう指定します。

```
PROPERTIES    invisible
```

ファイルを処理するときの動作の指定

表 13–2 に示す DATA_ATTRIBUTES フィールドは、主にアプリケーション・プログラマが使用します。ユーザがさまざまなデスクトップ・アクティビティを実行したときのファイルの動作を指定します。

詳細は、開発者環境向けマニュアルの『*Solaris* 共通デスクトップ環境 プログラマーズ・ガイド』を参照してください。

表 13–2 DATA_ATTRIBUTES フィールドと説明

フィールド	説明
MOVE_TO_ACTION	ディレクトリなどのコンテナに対して使用します。このデータ型のコンテナにファイルを移動したときに実行するアクションを指定します。
COPY_TO_ACTION	ディレクトリなどのコンテナに対して使用します。このデータ型のコンテナにファイルをコピーしたときに実行するアクションを指定します。
LINK_TO_ACTION	このデータ型のファイルにファイルをリンクしたときに実行するアクションを指定します。

表 13–2　DATA_ATTRIBUTES フィールドと説明　続く

フィールド	説明
IS_TEXT	このデータ型のファイルがテキスト・ボックスに表示できるテキストを含むよう指定します。
MEDIA	対応する ToolTalk メディア型を指定します。
MIME_TYPE	対応する MIME 型を指定します。
X400_TYPE	対応する X400 型を指定します。

データ型のデータ基準の定義

DATA_CRITERIA 定義は、オブジェクト型をファイルまたはディレクトリに割り当てるのに使用する基準を定義します。

使用できるオブジェクト型の基準は、表 13–3 のとおりです。

表 13–3　DATA_CRITERIA の基準と説明

基準	説明
ファイル名	ファイル名は指定したパターンに一致しなければなりません。NAME_PATTERN フィールドを使用します。
ファイル位置	パスは指定したパターンに一致しなければなりません。PATH_PATTERN フィールドを使用します。
ファイル内容	ファイル内容の指定した部分が指定したデータに一致しなければなりません。CONTENT フィールドを使用します。
ファイル・モード	ファイルは指定したアクセス権 (読み取り、書き込み、実行、ディレクトリ) を所有しなければなりません。MODE フィールドを使用します。
シンボリック・リンク	リンク名はオブジェクトがリンクするファイルに基づきます。

1 つのデータ型に 2 つ以上の基準を使用できます。ただし、NAME_PATTERN と PATH_PATTERN を同じデータ型で使用しないでください。

名前に基づいたデータ型

NAME_PATTERN フィールドを使用して、命名要件を指定します。フィールド値には、次のワイルドカードを指定できます。

? — 任意の 1 つの文字を示します。

* — 任意の文字列 (空文字列を含む) を示します。

[*cc*...] — 角括弧で囲まれた任意の文字 (*c*) を示します。

[*c*–*c*] — *c* から *c* までの範囲の任意の文字を示します。

例

- 次のデータ型定義は、ファイル名に基づいたデータ型を作成します。ファイル名は必ず QS で始まり .doc で終わるようにしてください。

```
DATA_ATTRIBUTES QS_Doc
    {
      DESCRIPTION    This file contains a document for the QS \
             project.
      ICON      Word_Doc
      ACTIONS     Open
    }

   DATA_CRITERIA QS_Doc_Criteria
    {
      NAME_PATTERN    QS*.doc
      DATA_ATTRIBUTES_NAME  QS_Doc
    }
```

- 次の定義は、Demo_*n* (*n* は 0 から 9) という名前のディレクトリのデータ型を作成します。

```
DATA_ATTRIBUTES Demo_directory
    {
      DESCRIPTION    This is a directory. Double-click to open it.
      ICON      Demo
      ACTIONS     OpenInPlace,OpenNewView
    }

   DATA_CRITERIA Demo_directory_criteria
    {
      NAME_PATTERN      Demo_[0-9]
      MODE          d
      DATA_ATTRIBUTES_NAME   Demo_directory
    }
```

位置に基づいたデータ型

PATH_PATTERN フィールドを使用してパスを指定します。NAME_PATTERN と同じワイルドカードを使用できます。

たとえば、次のデータ型はパスに基づいた基準を使用します。

```
DATA_ATTRIBUTES Project_Graphics
 {
  DESCRIPTION   Graphics file for the QS project. Double-click the \
        icon to see the graphic.
  ICON     QSgraphics
 }

DATA_CRITERIA Project_Graphics_Criteria
 {
  DATA_ATTRIBUTES_NAME   Project_Graphics
  PATH_PATTERN      */projects/QS/graphics/*
 }
```

名前と位置に基づいたデータ型

ファイル名と位置の両方に基づいたデータ型を作成するには、PATH_PATTERN の値に名前を取り込みます。NAME_PATTERN と PATH_PATTERN の両方を同じ基準定義で使用できません。

例

- 次に定義する QS_Source_Files データ型は、*/projects/QS のサブディレクトリにある app*n*.c (*n* は 1 から 9) という名前のすべてのファイルに適用されます。

```
DATA_ATTRIBUTES QS_Source_Files
    {
      …
    }

   DATA_CRITERIA QS_Source_Files_Criteria
    {
      PATH_PATTERN      */projects/QS/*/app[1-9].c
      DATA_ATTRIBUTES_NAME   QS_Source_Files
    }
```

- 次のデータ型は、/doc/project1 ディレクトリの ch*nn.xxx* (*n* は 0 から 9、*xxx* は任意の 3 文字のファイル名の拡張子) という名前のすべてのファイルに適用されます。

```
DATA_ATTRIBUTES ChapterFiles
    {
      DESCRIPTION  Chapter file for the project document.
      ICON     chapter
      ACTIONS    Edit,Print
    }
```

```
DATA_CRITERIA Chapter_Criteria
 {
   PATH_PATTERN      /doc/project1/ch[0-9][0-9].???
   DATA_ATTRIBUTES_NAME   ChapterFiles
 }
```

データ型作成基準としてのファイル・モードの使用

MODE フィールドを使用して、必須アクセス権を指定します。

通常、モード基準は名前、位置、内容に基づいたデータ型作成の組み合わせで使用します。これらの基準によって、データ型をファイルまたはディレクトリに制限したり、必須の読み取り権、書き込み権、実行権を指定したりできます。

MODE フィールドには、論理演算子 (表 13–4) と文字 (表 13–5) を指定できます。

表 13–4 MODE フィールドの論理演算子と説明

演算子	説明
!	論理演算子 NOT
&	論理演算子 AND
|	論理 OR

表 13–5 MODE フィールドの文字と説明

文字	説明
f	ファイルだけに適用されるデータ型
d	ディレクトリだけに適用されるデータ型
r	任意のユーザが読み取れるファイル
w	任意のユーザが書き込めるファイル
x	任意のユーザが実行できるファイル
l	リンクであるファイル

特定モードのデフォルトは、モードには関係ありません。

例

- 次のモード・フィールドは、データ型を制限します。

  `f&!w` — 読み専用ファイル

  `!w` — 読み専用ファイルおよび読み専用ディレクトリ

  `f&x` — 実行可能ファイル

  `f&r&x` — 書き込み可能および実行可能ファイル

  `x|!w` — 実行可能ファイルまたは読み専用ファイル

- 次のデータ型定義は、読み専用で実行可能でないファイルのデータ型を作成します。ファイル名は `*.doc` という命名規則に従っています。View アクションはデータ型に対して定義されているものとします。

```
DATA_ATTRIBUTES ReadOnlyDocument
    {
      ICON      read_only
      DESCRIPTION   This document is not writable. Double- \
          clicking runs your editor with a \
          read-only copy of the file.
      ACTIONS     View
    }

   DATA_CRITERIA ReadOnlyDocument_Criteria
    {
      NAME_PATTERN      *.doc
      MODE          !d&!x&!w
      DATA_ATTRIBUTES_NAME   ReadOnlyDocument
    }
```

内容に基づいたデータ型

CONTENT フィールドを使用して、ファイル内容に基づいたデータ型を作成します。内容に基づいたデータ型の作成には、名前または位置に基づいたデータ型を組み合わせて使用できます。

データ型作成は、ファイルの文字または数字内容に基づいて行われます。ファイルの最初のバイトの番号は 0 です。

- ファイルの文字内容には、次の構文を使用します。

  `CONTENT` *starting_byte* `string` *string*

- ファイルの数字内容には、次の構文を使用します。

```
CONTENT starting_byte byte number
 CONTENT starting_byte short number
 CONTENT starting_byte long number
```

- ディレクトリの内容には、次の構文を使用します。

```
 CONTENT 0 filename "file_name"
```

  8 進数 (先頭が o) と 16 進数 (先頭が ox) には、標準 C 表記を使用します。

注 - 内容に基づいたデータ型を作成すると、システム性能が遅くなります。できるだけ、名前または位置に基づいたデータ型を使用してください。

たとえば、次のデータ型 Writable_Wingz は、ファイルの最初に WNGZ 文字列が入っていて、書き込み権を持つすべてのファイルに適用されます。

```
DATA_ATTRIBUTES Writable_Wingz
 {
  …
 }

DATA_CRITERIA Writable_Wingz_Criteria
 {
  CONTENT      0 string WNGZ
  MODE       w&!d
  DATA_ATTRIBUTES_NAME Writable_Wingz
 }
```

▼ 独自の基準を持つデータ型を作成するには

いくつかの独自の基準を持つデータ型を作成できます。つまり、基準のいずれか (または両方) に一致した場合、ファイルはデータ型に割り当てられます。

1. **データ型の** DATA_ATTRIBUTES **定義を作成します。**

2. **基準ごとに** DATA_CRITERIA **定義を作成します。**

   DATA_ATTRIBUTES_NAME **フィールドを使用して、各基準を同じ** DATA_ATTRIBUTES **定義に接続します。**

たとえば、次の定義は Mif データ型を作成します。データ型は名前または内容に基づき作成します。

```
DATA_ATTRIBUTES Mif
{
  ICON        Frame
  ACTION_LIST      Open,Print
 }
```

```
DATA_CRITERIA Mif_Name_Criteria
 {
  DATA_ATTRIBUTES_NAME    Mif
  NAME_PATTERN       *.mif
 }

DATA_CRITERIA Mif_Content_Criteria
 {
  DATA_ATTRIBUTES_NAME  Mif
  CONTENT       1 string MIFFile
 }
```

ローカライズされたデータ型の作成

データ型の検索パスには、言語に依存した位置を含みます。デスクトップは LANG の値を使用して、データ型定義を検索する場所を決定します。

ローカライズされたデータ型の位置

ローカライズされたデータ型定義は、アクション検索パス上にある正しい言語依存ディレクトリになければなりません。

デフォルト検索パスは次のとおりです。

- 個人用アクション: *HomeDirectory*/.dt/types
- システム共通アクション: /etc/dt/appconfig/types/*language*
- 組み込みアクション: /usr/dt/appconfig/types/*language*

▼ データ型をローカライズするには

1. **適切な言語依存ディレクトリ (**/etc/dt/appconfig/types/japanese **など) にファイルを作成します。**
2. **データ型定義を言語に依存した構成ファイルにコピーします。**
3. **データ型定義の 1 つ以上のフィールドをローカライズします。**

第 14 章

デスクトップのアイコンの作成

デスクトップ・アイコンは、次のものと関連しています。

- ファイル・マネージャとアプリケーション・マネージャのアクション・ファイルおよびデータ型
- フロントパネル・コントロール
- アイコン化されたアプリケーション・ウィンドウ
- アプリケーションが使用するグラフィック (パレットやツールバーなど)
- ワークスペースの背景

この章では、次の項目について説明します。

注 - 開発環境用のマニュアルには、デスクトップ・アイコンに関する補足情報が載っています。詳細は、『共通デスクトップ環境 スタイル・ガイド』の第 4 章「視覚的な設計」を参照してください。

アイコン・イメージ・ファイル

デスクトップでアイコン・イメージを使用するには、アイコン・イメージ・ファイルは次の条件を満たさなければなりません。

- フォーマットが適切であること
- 適切なファイル命名規則を使用していること
- デスクトップのサイズ規則を使用していること
- アイコン検索パス上のディレクトリに位置すること
- 適切な構文を使用したデスクトップ構造によって呼び出されること。たとえば、フロントパネルに新しいコントロールを作成する場合、フロントパネル定義の ICON フィールドを使用して、そのコントロールに使用するアイコン・イメージを指定します。

アイコン・ファイルの形式

カラー・ディスプレイの場合は、通常 .pm 拡張子が付いている X ピックスマップ (XPM) 形式のアイコン・ファイルを使用します。それ以外の場合は、通常 .bm 拡張子が付いている X ビットマップ (XBM) 形式のファイルを使用します。ピックスマップ・ファイルで透明色を使用する場合は、.bm ファイルを作成したときにマスク・ファイル (_m.bm) が生成されます。これらのファイルをデスクトップで検索する方法については、152ページの「アイコン検索パス」を参照してください。

アイコン・ファイル名

アイコンと背景のイメージは、それぞれ別のファイルに格納されます。通常、アイコンはファイル名のベース部分で指定されます。たとえば、実際には次のようにアイコン・ファイルが格納されていても、アイコンは mail という名前で参照されることがあります。

```
/usr/dt/appconfig/icons/language/mail.l.pm
```

拡張子を追加するファイル命名規則は、アイコンをサイズと型で分類するのに便利です。デスクトップ・コンポーネントのアイコン名は、通常は次のいずれかの形式になります。

basename.size.format

basename.format

basename — イメージを参照するのに使用する、イメージ・ベース名

size — サイズを示す文字 l (大)、m (中)、s (小)、t (極小)

format — ファイル形式: pm (ピックスマップ)、bm (ビットマップ)

アイコン・サイズ規則

表 14–1 に、デスクトップ・アイコン用として推奨するピクセル数を示します。

表 14–1 アイコン・サイズとファイル名

アイコン・サイズ	ビットマップ名	ピックスマップ名
16x16 (極小)	*name*.t.bm	*name*.t.pm
24x24 (小)	*name*.s.bm	*name*.s.pm
32x32 (中)	*name*.m.bm	*name*.m.pm
48x48 (大)	*name*.l.bm	*name*.l.pm

表 14–2 に、デスクトップ・コンポーネントで使用するアイコン・サイズを示します。使用するアイコンのサイズは、ディスプレイ解像度に依存する場合があります。

表 14–2 デスクトップ・コンポーネントとそのアイコン・サイズ

デスクトップ・コンポーネント	高解像度	中解像度	低解像度
ファイル・マネージャとアプリケーション・マネージャ (名前とアイコンによる表示)	中	中	中
ファイル・マネージャとアプリケーション・マネージャ (名前と小アイコンによる表示)	極小	極小	極小
メイン・フロントパネルのコントロール	大	大	中
フロントパネルのサブパネル	中	中	極小
フロントパネルのスイッチ・コントロール	小	小	極小
アイコン化されたウィンドウ	大	大	中

たとえば、データ型に `mail` というアイコンを指定してから、カラー・ディスプレイを使用して、ファイル・マネージャの設定を小アイコンに変更した場合、使用されるアイコン・イメージは `mail.t.pm` です。

アイコン検索パス

デスクトップは、アイコン・ファイル、すなわちイメージを、ディレクトリのリストからファイルを検索して見つけます。ディレクトリのリストは「アイコン検索パス」と呼ばれ、いくつかの環境変数の値によって決定されます。アイコン検索パスを作成するための変数の使用方法と組み合わせ方については、152ページの「アイコン検索パス」で説明します。

デフォルトの検索パスは次のとおりです。

- 組み込みアイコン: `/usr/dt/appconfig/icons/`*language*
- システム共通アイコン: `/etc/dt/appconfig/icons/`*language*
- 個人用アイコン: *HomeDirectory*`/.dt/icons`

ネットワークによるアイコンへのアクセス

デスクトップは、リモート・システムのアイコンにアクセスできます。アイコン・サーバの作成については、131ページの「データベース、アイコン、およびヘルプ・サービスの構成」を参照してください。

アイコンとの関連付け

オブジェクトをより速く認識するために、アイコンを次のものと関連付けることができます。

- アクションおよびデータ型
- フロントパネルとサブパネルのコントロール
- アイコン化されたアプリケーション・ウィンドウ

アイコン・ファイルの指定

アクション、データ型、フロントパネル、サブパネルで使用するアイコンは、ベース名だけを指定します (拡張子は付けません) 。正しい拡張子が、ディスプレイ解像度、カラー・サポート、ファイル・マネージャの表示オプション ([表示方法] など) に応じて自動的に付けられます。

検索パスを無効にするには、絶対パスとアイコン名を指定します。

▼ アイコンをアクションまたはデータ型に関連付けるには

1. ICON フィールドを使用してアイコンを指定します。

   アイコン・ファイルが適切な命名規則に従っている場合は、アイコンのベース名だけを指定します。ディスプレイの解像度とカラー・サポートに基づいて、正しいアイコンが表示されます。

2. 次のアイコン・サイズを作成します。

   - アクション: 大、中、極小
   - データ型: 中、極小

アクション定義の例

次の例は、Island Paint 描画ツールを起動するためのアクション定義です。アイコン Ipaint.l と Ipaint.s が、アクションに関連付けられます。

```
ACTION IslandPaintOpenDoc
{
  WINDOW_TYPE    NO-STDIO
  ICON      Ipaint
  EXEC_STRING      /usr/bin/IslandPaint %Arg_1"File to open:"%
 }
```

カラー・アイコンを使用している場合は、デスクトップは実際のアイコン・ファイルを探すときに、まず .pm を追加します。カラー・アイコンを使用していない場合 (または .pm で一致するファイルがない場合) は、デスクトップは .bm を追加します。

データ型定義の例

次のデータ型定義は、アイコン comprsd.l と comprsd.s を圧縮ファイルに関連付けます。

```
DATA_ATTRIBUTES  COMPRESSED
{
  ICON     comprsd
  ACTIONS    Uncompress
  DESCRIPTION   A COMPRESSED file has been compressed by the \
        'compress' command to take up less space.
}
```

▼ アイコンをフロントパネル・コントロールに表示するには

1. ICON フィールドを使用して、イメージ名を指定します。

   コントロールがファイルを監視する場合 (MONITOR_TYPE がメールまたはファイルに設定されている場合) は、ALTERNATE_ICON フィールドを使用して、変更が見つかったときに使用するアイコンを指定します。

   ボタンとドロップ領域コントロールにアニメーションを使用することもできます。

2. 次のアイコン・サイズを作成します。
   - メイン・パネルとサブパネル: 大、中、極小
   - ワークスペース・スイッチ: 小

例

次のコントロールは、report という名前のファイルが /doc/ftp/pub/ ディレクトリにある場合に表示を変更します。ファイルがない場合は NoReport.pm アイコン、ファイルがある場合は Report.pm が表示されます。

```
CONTROL MonitorReport
 {
  CONTAINER_NAME     container_name
  TYPE        ICON
  MONITOR_TYPE    file
  FILE_NAME     /doc/ftp/pub/report
  ICON        NoReport
  ALTERNATE_ICON    Report
 }
```

▼ アイコンをアプリケーション・ウィンドウに関連付けるには

1. 次のように、ワークスペース・マネージャに iconImage リソースを設定します。

   Dtwm*_clientname_*iconImage: _icon_file_name_

   clientname の正しい値を決定するには、アプリケーション・マネージャを開いて、[デスクトップツール] アプリケーション・グループの [ウィンドウ属性] をダブルクリックします。ウィンドウを選択すると、その属性がリスト表示されます。WM_CLASS 属性は、ウィンドウのクラス名を引用符で囲んで表示します。

リソースの設定の詳細は、293ページの「アプリケーション・リソースの設定」を参照してください。

2. ワークスペース・メニューから **[ワークスペースマネージャの再起動]** を選択します。

   アイコンがワークスペース・マネージャに認識されたか確認するために、アイコンを変更しようとしているウィンドウをアイコン化します。

注 - 一部のアプリケーションでは、デフォルトのウィンドウ・アイコンを無効にできません。

▼ ファイル・マネージャをアイコン・ブラウザとして使用するには

1. `/usr/dt/examples/`***language***`/IconBrowse.dt` ファイルを ***HomeDirectory***`/.dt/types/Iconbrowse.dt` ディレクトリにコピーします。

2. アプリケーション・マネージャを開いて、**[デスクトップツール]** アプリケーション・グループの **[アクションの再読込み]** をダブルクリックします。

アイコン (`.bm` ファイルと `.pm` ファイル) が入っているディレクトリに変更する場合、各アイコンがアイコン名の隣に表示されます。たとえば、`/usr/dt/appconfig/icons/`*language* ディレクトリに変更すると、多くのデスクトップ・アイコンが表示されます。

注 - メモリの少ないシステムでアイコン・ブラウザを使用した場合、ファイル・マネージャがディレクトリを表示するのが遅くなることがあります。256x256 より大きいイメージは、デフォルトの構成では表示されません。

アイコン・ブラウザを使用不可にするには、次のようにします。

1. `IconBrowse.dt` ファイルの個人用コピーを削除します。

2. アプリケーション・マネージャを開いて、**[デスクトップツール]** アプリケーション・グループの **[アクションの再読込み]** をダブルクリックします。

アイコン設計についてのアドバイス

関連するアイコンの間では共通のテーマを使用します。たとえば、アプリケーションのアイコンを設計している場合は、アプリケーションのアイコンと、データ・ファイル用のアイコンの間に、意図的に類似性を持たせます。

設計するカラー・アイコンはすべて、2 色のバージョンも使用可能であるようにしてください。カラー・アイコンをモノクロ・ディスプレイかグレースケール・ディスプレイで表示する (または使用できる色数が少ない) 場合、そのアイコンは自動的に 2 色のバージョンで表示されます。

システムで使う色数を少なくするには、アイコンに使う色数を、デスクトップが提供する色数に限定してください。(アイコン・エディタを使って作成したアイコンは、デスクトップ・カラーのみ使用します。)

デスクトップ・コンポーネントが使用するサイズについては、表 14–1 を参照してください。

使用する色の数

デスクトップ・アイコンは、次の 22 色のパレットを使用します。

- グレー 8 色
- カラー 8 色 — 赤、青、緑、シアン、マゼンタ、黄、黒、白
- ダイナミックカラー 6 色 — フォアグラウンド、バックグラウンド、トップシャドウ、ボトムシャドウ、選択、透明

このパレットにより、他のアプリケーションが必要とするカラー・リソースを越えることなく、魅力ある読みやすいアイコンを作成できます。デスクトップで提供されるほとんどのアイコンはグレーを使用し、カラーでアクセントを付けています。

透明色は、矩形でない輪郭がぼんやりと見えて、アイコンの後ろの色が透けて見えるようなアイコンを作成するのに便利です。

第 15 章

フロントパネル拡張機能のカスタマイズ

フロントパネルのポップアップ・メニューと、サブパネルのアイコンのインストール・コントロールを使用して、フロントパネルをカスタマイズできます。

この章では、構成ファイルの作成および編集によるフロントパネルのカスタマイズについて説明します。

- フロントパネル・コントロールと構成については、`dtfpfile(4X)` のマニュアル・ページを参照してください。
- ワークスペース・マネージャについては、`dtwm(1)` と `dtwmrc(4)` のマニュアル・ページを参照してください。

フロントパネル構成ファイル

フロントパネルは、構成ファイルのデータベースで定義されます。

構成ファイルは、フロントパネルをカスタマイズする方法を提供します。次のような変更は、構成ファイルを編集しないと実行できません。

- 新しいコントロールの位置をメイン・パネルに追加する
- 特殊な型のコントロール (クライアントのウィンドウなど) を追加する
- 特定のデフォルト動作を変更する。たとえば、フロントパネル・コントロールがシングルクリックとダブルクリックのどちらに反応するかを変更する

パネル構成に最大限の柔軟性を提供するため、構成ファイルは個人用、システム共通、または他のシステムに配置できます。

フロントパネルは、ワークスペース・マネージャによって作成および管理されます。

デフォルトのフロントパネル構成ファイル

デフォルトのフロントパネル構成ファイルは、フロントパネル構成ファイル `/usr/dt/appconfig/types/`*language*`/dtwm.fp` で定義されます。

このファイルは変更しないでください。

フロントパネル構成ファイルの検索パス

フロントパネル定義は、ローカルに位置するファイルやリモート・システムのファイルに分散できます。

フロントパネルの定義に使用するファイルは、次の要求事項を満たさなければなりません。

- ファイル名が `.fp` で終わる。例: `mail.fp`
- ファイルがアクション・データベース検索パス上に位置する

デフォルトのアクション・データベース検索パスには、次のディレクトリがあります。次の順番で検索されます。

個人用カスタマイズ — *HomeDirectory*`/.dt/types`

システム共通カスタマイズ — `/etc/dt/appconfig/types/`*language*

組み込みパネルおよびコントロール — `/usr/dt/appconfig/types/`*language*

追加のディレクトリ *HomeDirectory*`/.dt/types/fp_dynamic` は、ユーザ・インタフェースで行われる個人用カスタマイズに使用します。このディレクトリは、手動のカスタマイズには使用しないでください。

アクション・データベース検索パスに、システムをネットワーク用に構成するためにディレクトリを追加することもあります。特に、システムがアプリケーション・サーバにアクセスするよう構成する場合は、リモート位置が追加されます。詳細は、150 ページの「データベース (アクションとデータ型) 検索パス」を参照してください。

フロントパネルの構成方法: 優先度規則

フロントパネルは、アクション・データベース検索パス上のすべての構成ファイルで作成されます。

定義のコンポーネントが競合する場合、どの定義を使用するかは優先度規則が決定します。次の場合、2 つのコンポーネントは競合します。

- CONTAINER_NAME と CONTAINER_TYPE が同じコントロール名である
- 2 つのコンポーネントが同じ位置にある (名前は異なるが、CONTAINER_NAME、CONTAINER_TYPE、および POSITION_HINTS が同じである)

フロントパネルは、次の優先度規則を使用します。

- コンポーネントのコントロール名、コンテナ名、コンテナ型がすべて同じ場合は、先に読み込まれたコンポーネントを使用します。

  たとえば、次のフィールドを持つという点以外は異なるシステム共通コントロールと組み込みコントロールの場合、システム共通コントロールが優先されます。

```
CONTROL TextEditor
    {
      CONTAINER_TYPE    BOX
      CONTAINER_NAME    Top
      …
    }
```

- 2 つのコンポーネントが同じ位置にある場合は、読み込まれた順番に配置します。

  たとえば、メインパネルに新しい個人用コントロール (CONTAINER_TYPE BOX と CONTAINER_NAME Top) を作成して POSITION_HINTS 5 を割り当てた場合、その個人用コントロールは、組み込みコントロールとその他の 5 より高い位置番号を持つすべてのコントロールを 1 つずつ右へずらします。

注 - 新規にシステム共通または個人用のコントロールを作成することによってコントロールを変更する場合は、新しいコントロール定義に同じコントロール名 CONTAINER_NAME と CONTAINER_TYPE を指定しなければなりません。指定しない場合は、既存のコントロールと新しいコントロールの両方が表示されます。

動的に作成されたフロントパネル・ファイル

[アイコンのインストール] コントロールとポップアップ・メニューを使用してフロントパネルをカスタマイズすると、ファイルは *HomeDirectory*/.dt/types/fp_dynamic ディレクトリに書き込まれます。

フロントパネルは、追加のファイル *HomeDirectory*/.dt/sessions/dtwmfp.session を作成します。このファイルは、カスタマイズしたフロントパネルの状態をセッションごとに保存および復元するのに使用します。

ユーザ・インタフェースのカスタマイズの管理

フロントパネルのコントロールのポップアップ・メニューと [アイコンのインストール] コントロールを使用して、フロントパネルを大規模にカスタマイズできます。

この節では次のことを説明します。

- 特定の個人用カスタマイズを回避する方法。たとえば、ユーザがコントロールを削除できないようにする方法など
- 個人用カスタマイズを元に戻す方法。たとえば、うっかり削除してしまった1つのコントロールを復元するよう他のユーザが要求してきた場合の方法など

▼ 個人用カスタマイズを回避するには

1. **コントロールが組み込みコントロールの場合は、定義を /usr/dt/appconfig/types/*language*/dtwm.fp から /etc/dt/appconfig/types/*language*/*name*.fp へコピーします。**

2. **次の行をコントロール定義に追加します。**

```
LOCKED    True
```

▼ 削除されたコントロールまたはサブパネルを復元するには

[デスクトップツール] アプリケーション・グループの [フロントパネルの復元] アクションは、ユーザ・インタフェースで行われたフロントパネルのカスタマイズをすべて削除します。このアクションを使用して、フロントパネルのポップアップ・メニューで行なった個人用カスタマイズをすべて削除できます。

個々のコントロールを復元するには、次の手順に従ってください。

- ***HomeDirectory***/.dt/types/fp_dynamic ディレクトリで、コントロールを削除したときに作成されたファイルを削除します。コントロールは、削除された元のコントロールと同じ名前になります。

たとえば、アイコン・エディタ・コントロールを削除した場合、fp_dynamic ディレクトリのファイルの内容は次のようになります。

```
CONTROL IconEditor
{
   …
  DELETE   True
 }
```

サブパネルを削除すると、そのサブパネルとサブパネルの各コントロールに対して、別の動的ファイルが作成されます。

フロントパネル定義の構成

フロントパネルは、フロントパネルのコンポーネントの定義を集めて構築されます。各コンポーネントは、フロントパネル上のコンポーネントの配置と、コンポーネントの外観および動作を定義する構文が必要です。

フロントパネル・コンポーネント

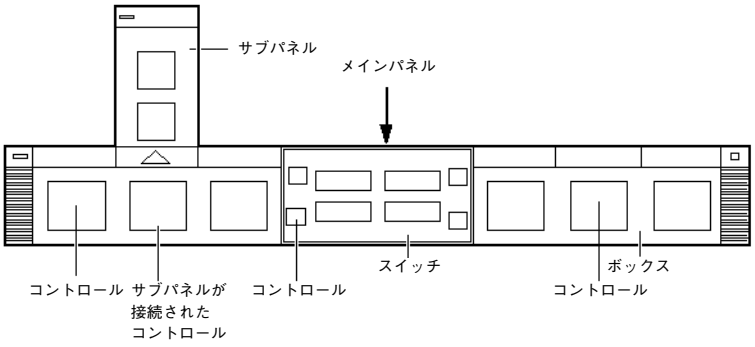


図 15–1　フロントパネルのコンポーネント

フロントパネルは、次のように構成されます。

- パネルは、フロントパネル全体のトップレベルのコンテナ (親) です。
- パネルは 1 つ以上のボックスのコンテナです。
- ボックスは 1 つ以上のコントロールのコンテナです。

特別なコンテナは 2 種類あります。

- サブパネルは、特定のコントロールと関連付けられています (このコントロールはサブパネルのコンテナです)。サブパネルは、関連付けられているコントロールから上方にスライドして表示されます。
- スイッチには、ワークスペースを変更するボタンと追加のコントロールが入っています。

フロントパネル定義の一般的な構文

フロントパネルの各コンポーネントは、次の構文を使用して別々に定義されます。

```
COMPONENT  name
{
  KEYWORD          value
  KEYWORD          value
  ...
 }
```

キーワードには、必須なものと省略可能なものがあります。詳細は、`dtfpfile(4X)` のマニュアル・ページを参照してください。

パネル定義

パネルはトップレベルのコンポーネントです。定義には次のものが含まれます。

- フロントパネル名
- フロントパネル全体の一般的な外観と動作

```
PANEL front_panel_name
   {
      KEYWORD      value
      KEYWORD      value
      ...
   }
```

front_panel_name は、フロントパネルに固有の名前です。デフォルトは「FrontPanel」です。

ボックス定義

ボックス定義には次のものが含まれます。

- ボックス名
- ボックスが入っているパネル (CONTAINER_NAME)
- パネル内のボックスの位置 (POSITION_HINTS)
- ボックス全体に適用する外観と動作を記述するフィールド

```
BOX box_name
   {
     CONTAINER_NAME       front_panel_name
     POSITION_HINTS       position
     KEYWORD      value
     KEYWORD      value
      ...
   }
```

コントロール定義

コントロール定義には次のものが含まれます。

- コントロール名
- コントロールがボックス、サブパネル、スイッチのどの中にあるのか (CONTAINER_TYPE)
- コントロールがどのボックス、サブパネル、スイッチに入っているのか (CONTAINER_NAME)
- ボックス内のコントロールの位置 (POSITION_HINTS)

- コントロールの外観と動作を記述するフィールド

```
CONTROL control_name
   {
     CONTAINER_TYPE      BOX or SUBPANEL or SWITCH
     CONTAINER_NAME      box_name or subpanel_name or switch_name
     TYPE          control_type
     POSITION_HINTS      position
     KEYWORD         value
     KEYWORD         value
    …
   }
```

サブパネル定義

サブパネル定義には次のものが含まれます。

- サブパネル名
- サブパネルを関連付けるコントロール名 (CONTAINER_NAME)
- サブパネルに固有の外観と動作を記述するフィールド

```
SUBPANEL subpanel_name
   {
    CONTAINER_NAME      control_name
    KEYWORD         value
    KEYWORD         value
      …
    }
```

スイッチ定義

スイッチ定義には次のものが含まれます。

- スイッチ名
- スイッチが入っているボックス (CONTAINER_NAME)
- ボックス内のスイッチの位置 (POSITION_HINTS)
- スイッチの外観と動作を記述するフィールド

```
SWITCH switch_name
   {
    CONTAINER_NAME      box_name
    POSITION_HINTS      position
    KEYWORD         value
    KEYWORD         value
    …
   }
```

メイン・パネルの変更

メイン・パネルは、サブパネルを除くフロントパネルのウィンドウです。

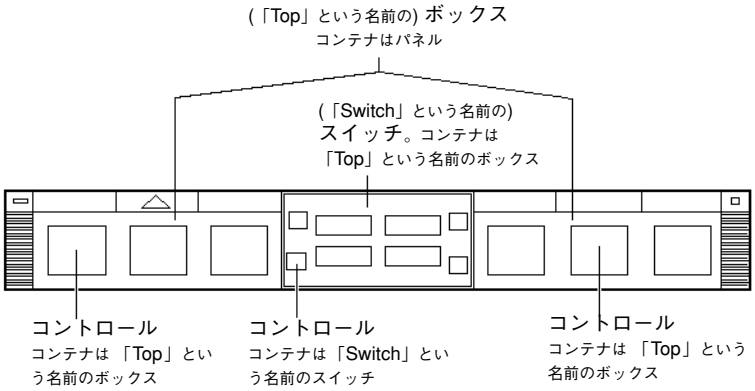


図 15–2　メイン・パネルのコンテナ

次の変更を実行できます。

- コントロールの追加または削除
- コントロールの位置の交換

▼ メイン・パネルにコントロールを追加するには

1. フロントパネル構成ファイルを作成します。
   - システム共通: `/etc/dt/appconfig/types/`*language*`/*.fp`
   - 個人用: *HomeDirectory*`/.dt/types/*.fp`
2. ファイルにコントロールを定義します。

   コントロールのコンテナを指定するために、CONTAINER_NAME フィールドと CONTAINER_TYPE フィールドを使用します。

```
CONTAINER_NAME  Top
CONTAINER_TYPE  BOX
```

   コントロールの配置を左から右へ指定するために POSITION_HINTS を使用します。カスタマイズは組み込みコントロールに優先するので、新しいコントロールが入ると、同じ位置にあった既存のコントロールは 1 つずつ右へずれます。

3. 構成ファイルを保存します。

4. フロントパネル・コントロールのアイコンを作成します。

   詳細は、255ページの「コントロールが使用するアイコンの指定」を参照してください。

5. ワークスペース・メニューから **[ワークスペースマネージャの再起動]** を選択します。

たとえば、ファイル /etc/dt/appconfig/types/*language*/audio.fp にある次のコントロール定義は、オーディオ・アプリケーションのコントロールを [時計] と [カレンダ] コントロールの間に挿入します。

```
CONTROL AudioApplication
 {
  TYPE              icon
  CONTAINER_NAME    Top
  CONTAINER_TYPE    BOX
  ICON              AudioApp
  POSITION_HINTS    2
  PUSH_ACTION       StartAudioApplication
  PUSH_RECALL       true
 }
```

▼ コントロールを削除するには

1. フロントパネル構成ファイルを作成します。

   - システム共通: /etc/dt/appconfig/types/*language*/*name*.fp
   - 個人用: *HomeDirectory*/.dt/types/*name*.fp

2. 削除するコントロールの定義を新しいファイルにコピーします。

   削除するコントロールが組み込みの場合、定義は次のファイルにあります。

   /usr/dt/appconfig/types/*language*/dtwm.fp

   定義全体をコピーする必要はありませんが、必ず CONTAINER_NAME と CONTAINER_TYPE のフィールドを含む範囲をコピーしてください。

3. 定義に DELETE フィールドを追加します。

```
DELETE    True
```

4. 構成ファイルを保存します。

5. ワークスペース・メニューから **[ワークスペースマネージャの再起動]** を選択します。

たとえば、ファイル /etc/dt/appconfig/types/*language*/TrashCan.fp にある次のコントロール定義は、[ごみ箱] コントロールを削除します。

```
CONTROL Trash
 {
  CONTAINER_NAME  Top
  CONTAINER_TYPE  BOX
  DELETE          True
 }
```

▼ コントロールを変更するには

コントロール定義を変更する必要がある場合 (たとえばアイコン・イメージを変更する場合など) は、次の手順を実行します。

1. コントロール定義全体を /usr/dt/appconfig/types/***language***/dtwm.fp から次のファイルにコピーします。

   - システム共通: /etc/dt/appconfig/types/*language*/*name*.fp
   - 個人用: *HomeDirectory*/.dt/types/*name*.fp

2. 変更するフィールドを編集します。フィールドを追加することもできます。

3. ファイルを保存します。

4. ワークスペース・メニューから **[ワークスペースマネージャの再起動]** を選択します。

▼ コントロールの位置を交換するには

1. 位置を変更するコントロールの定義を /usr/dt/appconfig/types/***language***/dtwm.fp から次のファイルにコピーします。

   - システム共通: /etc/dt/appconfig/types/*language*/*name*.fp

- 個人用: *HomeDirectory*/.dt/types/*name*.fp

移動するコントロールごとに、コントロール定義全体をコピーしなければなりません。

2. コントロール定義の POSITION_HINTS フィールドの値を交換します。
3. ファイルを保存します。
4. ワークスペース・メニューから **[ワークスペースマネージャの再起動]** を選択します。

たとえば、ファイル /etc/dt/appconfig/types/C/MailHelp.fp にある次のコントロール定義は、[メール] と [ヘルプ・マネージャ] のコントロールの位置を交換し、それらのコントロールを個人の変更に対してロックします。

```
CONTROL Mail
 {
  POSITION_HINTS     12
  LOCKED        True
  …その他のコントロール定義
 }


CONTROL Help
 {
  POSITION_HINTS     5
  LOCKED        True
  …その他のコントロール定義
 }
```

▼ フロントパネル・コントロールを交換するには

♦ 次のものを同じにして、別のコントロール定義を作成します。

- *control_name*
- CONTAINER_NAME value

たとえば、次の 2 つのコントロールは異なる 2 つの構成ファイルで定義されます。これらのコントロールはコントロール名とコンテナ名が同じなので、同一コントロールと見なされます。

- /etc/dt/appconfig/types/C/SysControls.fp にある定義

```
Control ImportantApplication
    {
      CONTAINER_NAME     Top
      CONTAINER_TYPE     BOX
      POSITION_HINTS     2
      …
    }
```

- *HomeDirectory*/.dt/types/MyControls.fp にある定義

```
Control ImportantApplication
    {
      CONTAINER_NAME     Top
      CONTAINER_TYPE     BOX
      POSITION_HINTS     6
      …
    }
```

個人用コントロールが優先するので、コントロールは位置 6 に配置されます。

コントロールが使用するアイコンの指定

コントロール定義の ICON フィールドは、コントロールが使用するアイコン・イメージを定義します。

ICON フィールドの値は、次のいずれかになります。

- ベース・ファイル名

  ベース・ファイル名は、アイコン・イメージを格納しているファイル名から、サイズの拡張子 (m と t) とイメージ型 (bm と pm) を除いたものです。たとえば、ファイル名が MyGame.l.pm と MyGame.m.pm の場合は、MyGame を使用します。

  ベース・ファイル名を使用する場合、アイコン・ファイルはアイコン検索パス上のディレクトリになければなりません。

  - 個人用アイコン: *HomeDirectory*/.dt/icons
  - システム共通アイコン: /etc/dt/appconfig/icons/*language*

- 完全なファイル名を含む、アイコン・ファイルへの絶対パス

  アイコン・ファイルへの絶対パスは、アイコン・ファイルがアイコン検索パス上にない場合のみ使用してください。

  必要なアイコンのサイズは、コントロールの位置により決まります。

- メイン・パネル — 48x48 ピクセル (*name*.l.pm または *name*.l.bm)
- サブパネル — 24x24 ピクセル (*name*.s.pm または *name*.s.bm)

アイコン・ファイルを次のいずれかに配置します。

- 個人用アイコン: *HomeDirectory*/.dt/icons
- システム共通アイコン: /etc/dt/appconfig/icons/*language*

サブパネルの作成と変更

フロントパネルのポップアップ・メニューを使用して、サブパネルを作成および変更できます。

この節では、システム共通のカスタマイズの方法を説明します。カスタマイズするには、フロントパネル構成ファイルを変更する必要があります。

サブパネルは、メイン・パネル内のコントロールに「接続」されています。

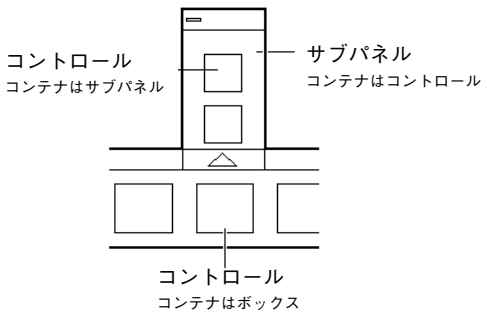


図 15–3　サブパネルのコンテナ (サブパネルが接続されているコントロール)

接続は、サブパネルの定義で行われます。CONTAINER_NAME フィールドは、サブパネルが接続されるコントロールを指定します。

```
CONTROL control_name
{
    …
 }

SUBPANEL subpanel_name
{
  CONTAINER_NAME        control_name
  …
 }
```

▼ 新しいシステム共通サブパネルを作成するには

1. メイン・パネル上の、サブパネルを接続するコントロールの ***control_name*** を検出します。

   コントロールが組み込みの場合、定義は次のファイルにあります。

   `/usr/dt/appconfig/types/`*language*`/dtwm.fp`

2. 新しいファイル `/etc/dt/appconfig/types/`***language***`/*.fp` を作成します。

3. サブパネルを定義します。

```
SUBPANEL subpanel_name
    {
      CONTAINER_NAME     control_name
      TITLE     value
      KEYWORD     value
      ...
    }
```

4. 新しい構成ファイルを保存します。

5. ワークスペース・メニューから **[**ワークスペースマネージャの再起動**]** を選択します。

組み込みサブパネルのカスタマイズ

組み込みサブパネルの一般的な属性 (タイトルなど) と内容を変更できます。

組み込みサブパネルの一般的な属性を変更するには

1. 新しいフロントパネル構成ファイルを作成します。
   - システム共通: `/etc/dt/appconfig/types/`*language*`/`*name*`.fp`
   - 個人用: *HomeDirectory*`/.dt/types/`*name*`.fp`

2. デフォルトのサブパネル定義全体を `/usr/dt/appconfig/types/`***language***`/dtwm.fp` から新しいファイルへコピーします。

```
SUBPANEL     subpanel_name
    {
      ...
```

```
}
```

3. サブパネル定義を変更します。
4. 新しい構成ファイルを保存します。
5. ワークスペース・メニューから **[ワークスペースマネージャの再起動]** を選択します。

たとえば、ファイル /users/janice/.dt/types/PerApps.fp にある次の定義は、[個人アプリケーション] サブパネル名を変更します。

```
SUBPANEL PersAppsSubpanel
 {
  CONTAINER_NAME     TextEditor
  TITLE        Janice's Applications
 }
```

組み込みサブパネルにシステム共通コントロールを追加するには

1. フロントパネル構成ファイル /etc/dt/appconfig/types/***language***/***name***.fp を作成します。
2. システム共通コントロールをファイルに定義します。

   コントロールのコンテナを指定するために、CONTAINER_NAME フィールドと CONTAINER_TYPE フィールドを使用します。

   ```
   CONTROL control_name
       {
         CONTAINER_NAME      subpanel_name
         CONTAINER_TYPE      SUBPANEL
         …
       }
   ```

   詳細は、260ページの「フロントパネル・コントロール定義」を参照してください。
3. 構成ファイルを保存します。
4. ワークスペース・メニューから **[ワークスペースマネージャの再起動]** を選択します。

たとえば、新しいファイル
/etc/dt/appconfig/types/*language*/DigitalClock.fp に定義された次のコントロールは、([デスクトップツール] アプリケーション・グループの) [デジタル時計] を、すべてのユーザの [個人アプリケーション] サブパネルに追加します。

```
CONTROL DigitalClockControl
 {
  TYPE      icon
  CONTAINER_NAME   PerAppsSubpanel
  CONTAINER_TYPE   SUBPANEL
  ICON      Dtdgclk
  PUSH_ACTION    DigitalClock
  PUSH_RECALL    True
 }
```

組み込みサブパネルからコントロールを削除するには

- メイン・パネルのコントロールを削除する方法と同じです。詳細は、252ページの「コントロールを削除するには」を参照してください。

アイコンのインストールのコントロールを削除するには

- 次のフィールドをサブパネル定義に追加します。

```
CONTROL_INSTALL     False
```

▼ サブパネルの自動的に閉じる動作を変更するには

デフォルトでは、サブパネルを元の位置から移動していない限り、コントロールを選択するとサブパネルも閉じます。

次の手順で、サブパネルを明示的に閉じるまでサブパネルを開いておくようにフロントパネルを構成できます。

1. フロントパネル構成ファイルを作成します。
   - システム共通: /etc/dt/appconfig/types/*language*/*.fp
   - 個人用: *HomeDirectory*/.dt/types/*.fp
2. デフォルトのパネル定義を /usr/dt/appconfig/types/***language***/dtwm.fp から新しいファイルへコピーします。

```
PANEL FrontPanel
    {
      …
```

```
}
```

3. 次のフィールドをパネル定義に追加します。

```
SUBPANEL_UNPOST     False
```

4. 新しい構成ファイルを保存します。

5. ワークスペース・メニューから **[ワークスペースマネージャの再起動]** を選択します。

フロントパネル・コントロール定義

アイコンを [アイコンのインストール] コントロールにドロップすることで、個人用のコントロールを作成できます。

この方法によって簡単にカスタマイズできますが、提供される機能はフロントパネル・コントロールの機能のサブセットになります。たとえば、[アイコンのインストール] コントロールを使用して作成されたコントロールでは、次の作業は実行できません。

- アニメーションの提供
- クライアント・ウィンドウの表示
- イベント発生時の外観の変更 (たとえば新しいメールを受け取った場合など)

この節では、手動でフロントパネル・コントロールを作成する方法を説明します。

フロントパネル・コントロールの構文については、`dtfpfile(4X)` のマニュアル・ページを参照してください。

フロントパネル・コントロール定義

フロントパネル定義の構造は次のとおりです。

```
CONTROL control_name
{
  TYPE              control_type
  CONTAINER_NAME    value
```

```
  CONTAINER_TYPE   value
  外観と動作を定義している他のフィールド
}
```

コントロールの型

コントロール定義の TYPE フィールドは、コントロールの基本的な動作を指定します。

TYPE フィールド	コントロールの動作
icon	(デフォルト) コントロールは、コントロールをクリックするかファイルをコントロール上にドロップすると、指定されたアクションを実行します。
blank	コントロールの間隔を調節するためのプレースホルダ
busy	ビジー・ライト。アクションが起動されると、コントロールが点滅します (イメージを切り替えます)。
client	フロントパネルのクライアント・ウィンドウ
clock	時計
date	現在の日付を表示します。
file	ファイルを表します。コントロールを選択すると、ファイルのデフォルト・アクションを実行します。

▼ 新しいコントロールを作成するには

この節では、コントロールを定義する一般的な手順と、さまざまな型のコントロールの作成方法を説明します。

1. コントロールに PUSH_ACTION と DROP_ACTION のいずれかが関連付けられている場合は、アクション定義を作成します。

   これらは、コントロールをクリック、またはファイルをコントロール上にドロップすると実行されるアクションです。

2. コントロールのアイコン・イメージ・ファイルを作成します。

   アイコンのサイズ、名前、および位置については、235ページの「アイコン・イメージ・ファイル」を参照してください。

3. 新しいフロントパネル構成ファイルを作成します。
   - システム共通: `/etc/dt/appconfig/types/`*language*`/*.fp`
   - 個人用: *HomeDirectory*`/.dt/types/*.fp`
4. ファイルにコントロール定義を追加します。
5. ファイルを保存します。
6. ワークスペース・メニューから **[ワークスペースマネージャの再起動]** を選択します。

クリックするとアクションを実行するコントロールの作成

コントロールの動作を定義するには、次のフィールドを使用します。

- `TYPE`: `icon` に設定します。
- `PUSH_ACTION`: 実行するアクション名を指定します。

たとえば、[個人アプリケーション] サブパネルに入る次のコントロールは、ユーザが獲得したゲームを実行します。

```
CONTROL Ball
 {
  TYPE       icon
  CONTAINER_NAME    PersAppsSubpanel
  CONTAINER_TYPE    SUBPANEL
  ICON       ball
  PUSH_ACTION     RunBallGame
  HELP_STRING     "Choose this control to play Ball."
 }
```

次のコントロールは、スイッチの左上隅に配置されます。このコントロールは、CutDisp という名前のアクションを起動します。

```
CONTROL StartCutDisp
 {
   TYPE       icon
   CONTAINER_NAME   Switch
   CONTAINER_TYPE   SWITCH
   POSITION_HINTS   first
   ICON       cutdisp
   HELP_STRING    "Choose this control to run cutdisp."
   PUSH_ACTION    CutDisp
 }
```

ファイルを開くコントロールの作成

コントロールの動作を定義するには、次のフィールドを使用します。

- TYPE — file に設定します。
- FILE_NAME — 開くファイルのパスを指定します。
- PUSH_ACTION — Open に設定します。

ファイルのデータ型に Open アクションが定義されていなければなりません。

たとえば、次のコントロールはメイン・パネルの右側奥に配置されます。このコントロールは、テキスト・エディタをデータファイル /users/ellen/PhoneList.txt で起動します。*.txt ファイルの Open アクションは、デフォルト・アクション・データベースの一部です。

```
CONTROL EditPhoneList
 {
  TYPE       file
  FILE_NAME     /users/ellen/PhoneList.txt
  CONTAINER_NAME    Top
  CONTAINER_TYPE    BOX
  POSITION_HINTS    last
  ICON       PhoneBook
  HELP_STRING     "This control displays Ellen's phone list."
  PUSH_ACTION     Open
 }
```

ドロップ領域として動作するコントロールの作成

ファイルをコントロール上にドロップしたときに実行されるアクションは、DROP_ACTION フィールドに指定します。このアクションは、ファイル引き数を受け取れなければなりません。

コントロール定義に PUSH_ACTION フィールドと DROP_ACTION フィールドの両方を含む場合が多いです。同じアクションをプッシュ&ドロップ・アクションに使用できます。

たとえば、[個人アプリケーション] サブパネルにある次のコントロールは、ファイル引き数を受け取る X クライアント xwud を実行します。

```
CONTROL Run_xwud
 {
  CONTAINER_NAME    PerAppsSubpanel
  CONTAINER_TYPE    SUBPANEL
  POSITION_HINTS    2
  ICON       XwudImage
  PUSH_ACTION     RunXwud
  DROP_ACTION     RunXwud
 }
```

ファイルを監視するコントロールの作成

コントロールの動作を定義するには、次のフィールドを使用します。

- TYPE — 次のいずれかの値を指定します。

  icon — コントロールに PUSH_ACTION、DROP_ACTION のいずれかを指定する場合は、この型を使用します。

  file — コントロールを選択したときに、ファイル・マネージャでファイル・アイコンをダブルクリックしたときのような動作を実行する場合は、この型を使用します。

- ICON と ALTERNATE_ICON — 監視するファイルの、変更なしの状態と変更ありの状態を示すイメージを記述します。

- MONITOR_TYPE — イメージを変化させる条件を記述します。次のいずれかの値を使用します。

  mail — コントロールは、ファイルに情報が追加されると外観が変わります。

  file — コントロールは、指定されたファイルが空でなくなると外観が変わります。

- FILE_NAME — ファイルを監視するように指定します。

たとえば、次のコントロールは、anonymous ftp を使用して自分のシステムに転送されることになっている、meetings という名前のファイルの有無を確認します。このコントロールは、[個人アプリケーション] サブパネルの一番上に配置されます。

```
CONTROL MonitorCalendar
 {
  TYPE       file
  CONTAINER_NAME    PersonalApps
  CONTAINER_TYPE    SUBPANEL
  POSITION_HINTS    first
  FILE_NAME     /users/ftp/meetings
  MONITOR_TYPE    file
  ICON       meetingsno
  ALTERNATE_ICON    meetingsyes
 }
```

1 インスタンス (切り替え) コントロール

1 インスタンス・コントロールは、PUSH_ACTION によって起動されたプロセスがすでに実行中かどうかをチェックします。プロセスが実行中でない場合は、PUSH_ACTION が実行されます。プロセスが実行中の場合は、ウィンドウが現在のワークスペースのウィンドウの重なりの一番上に移動します。

コントロールの動作を定義するには、次のフィールドを使用します。

- PUSH_RECALL — True に設定します。

- CLIENT_NAME — コントロールにクライアント名を指定します。

CLIENT_NAME の値は、アプリケーションのトップレベル・ウィンドウの WM_CLASS 属性の一番目の文字列 (*res_name*) に一致しなければなりません。詳細は、xprop(1) のマニュアル・ページを参照してください。

- PUSH_ACTION — コントロールをクリックしたときに実行されるアクションを記述します。

たとえば次のコントロールは、MyEditor という名前のアクションを持つアプリケーションのインスタンスを 1 つ実行します。

```
CONTROL MyEditor
 {
  TYPE       icon
  CONTAINER_NAME    Top
  CONTAINER_TYPE    BOX
  POSITION_HINTS    15
  PUSH_RECALL     True
  CLIENT_NAME     BestEditor
  PUSH_ACTION     StartMyEditor
  ICON       MyEd
 }
```

クライアントのウィンドウ・コントロールを作成するには

クライアント・ウィンドウ・コントロールは、フロントパネルにはめ込まれたアプリケーション・ウィンドウです。たとえば、xload クライアントのウィンドウ・コントロールを作成することで、システム負荷メータをフロントパネルに入れることができます。

1. コントロールを定義します。

   コントロールの動作を定義するには、次のフィールドを使用します。

   - TYPE — client に設定します。
   - CLIENT_NAME — 起動するクライアントを指定します。

     CLIENT_NAME の値は、アプリケーションのトップレベル・ウィンドウの WM_CLASS 属性の一番目の文字列 (*res_name*) に一致しなければなりません。詳細は、xprop(1) のマニュアル・ページを参照してください。
   - CLIENT_GEOMETRY — クライアントのフロントパネル・ウィンドウで必要なサイズをピクセル単位で指定します。

     xwininfo(1) のマニュアル・ページで、ピクセル単位のウィンドウ・サイズを調べる方法を説明しています。

2. ワークスペース・メニューから **[**ワークスペースマネージャの再起動**]** を選択します。

3. 端末エミュレータのコマンド行からクライアントを起動します。

たとえば、次のコントロールは 30x20 ピクセルの負荷メータを表示します。

```
CONTROL LoadMeter
 {
  TYPE       client
  CONTAINER_NAME    Top
  CONTAINER_TYPE    BOX
  CLIENT_NAME     xload
  CLIENT_GEOMETRY   30x20
 }
```

セッション中にクライアントが保存および復元されない場合、コントロールをクリックすると、そのコントロールがクライアントを起動するように構成するとします。たとえば、次の行を定義に追加すると LoadMeter コントロールが `xload` を起動するように構成できます。

```
PUSH_ACTION    StartXload
```

次のアクションを作成します。

```
ACTION StartXload
 {
  WINDOW_TYPE    NO_STDIO
  EXEC_STRING    /usr/contrib/bin/X11/xload
 }
```

コントロールをアニメーション化するには

アニメーション・シーケンスを接続してコントロールを選択するか、オブジェクトをコントロール上にドロップしたときに使用できます。

アニメーション・シーケンスを指定するには、コントロールは次の条件が必要です。

- 型が `icon` である
- `PUSH_ACTION` または `DROP_ACTION` を持っている

1. `ANIMATION` コンポーネントを使用して、アニメーション・シーケンスを指定します。

```
ANIMATION animation_name
     {
      ANIMATION icon1      [milisecond_delay]
      ANIMATION icon2      [milisecond_delay]
      …
       }
```

icon1 や *icon 2* は、アイコン名で、*milisecond_delay* は、アニメーション・アイコン間のミリ単位の遅延時間です。デフォルトの遅延時間は 200 ミリ秒です。

2. PUSH_ANIMATION フィールドと DROP_ANIMATION フィールド、またはそのどちらかをコントロール定義に追加します。値は ANIMATION シーケンス名です。

たとえば次の行は、BestEditor アプリケーションを起動するコントロールをアニメーション化します。アイコン間の遅延時間は 300 ミリ秒です。この例では、アイコン・ファイル frame1、frame2 などを作成してあると想定しています。

```
CONTROL BestEditor
 {
  …
  PUSH_ANIMATION BestEdAnimation
  …
 }


ANIMATION BestEdAnimation
 {
  frame1   300
  frame2
  …
 }
```

フロントパネル・コントロールのアイテムヘルプを提供する

コントロールにヘルプを提供するには、次の 2 つの方法があります。

- コントロール定義にヘルプ文字列を提供する

  コントロールのアイテムヘルプを起動すると、ヘルプ文字列がヘルプ・ビューアに表示されます。ヘルプ文字列にはフォーマット (ヘッダなど) やリンクを指定できません。

  ヘルプ文字列を表示するには、コントロール定義にヘルプ文字列を指定します。

  ```
  HELP_STRING        help_string
  ```

- 登録済みヘルプ・ボリュームにヘルプ・トピックを指定する

  ヘルプ・トピックは、ヘルプ・システムの全機能を使用して書かれた情報です。ヘルプ・トピックを記述するには、デスクトップのヘルプ開発者用キットを使用する必要があります。

  ヘルプ・トピックを表示するには、ヘルプ・ボリュームとトピック ID をコントロール定義に指定します。

  ```
  HELP_VOLUME     help_volume_name
  HELP_TOPIC      topic_id
  ```

ワークスペース・スイッチのカスタマイズ

ワークスペース・スイッチをカスタマイズするには、次のいくつかの方法があります。

- ワークスペースの数を変更する
- スイッチの配置を変更する
- スイッチのコントロールを変更する

▼ ワークスペースのデフォルト数を変更するには

♦ 次のワークスペース・マネージャ・リソースを変更します。

```
Dtwm*workspaceCount:      n
```

詳細は、279ページの「システム共通ベースのワークスペース数を変更するには」を参照してください。

▼ スイッチの列の数を変更するには

♦ SWITCH 定義の NUMBER_OF_ROWS フィールドを変更します。

たとえば、次の例は 3 列のスイッチを定義します。

```
SWITCH Switch
 {
  CONTAINER_NAME     box_name
  NUMBER_OF_ROWS     3
   …
 }
```

▼ ワークスペース・スイッチのコントロールを変更および追加するには

1. フロントパネル構成ファイルをコントロール定義とともに作成します。
   - コントロールをスイッチの内側に指定します。

```
CONTAINER_NAME    Switch
CONTAINER_TYPE    SWITCH
```

- スイッチ内での位置を指定します。

  ```
  POSITION_HINTS     n
  ```

  n は整数です。位置は左から右、上から下の順に番号が付けられています (デフォルトの 2 列スイッチの場合、位置は 1 ～ 4 です)。

2. コントロールのアイコンを作成します。**16x16** ピクセルのサイズを推奨します。

たとえば次のコントロールは、スイッチに端末エミュレータ・コントロールを入れます。

```
 CONTROL SwitchTerminal
{
 TYPE                   icon
 CONTAINER_NAME         Switch
 CONTAINER_TYPE         SWITCH
 POSITION_HINTS         3
 ICON                   Fpterm
 LABEL                  Terminal
 PUSH_ACTION            Dtterm
 HELP_TOPIC             FPOnItemTerm
 HELP_VOLUME            FPanel
}
```

このコントロールは、組み込みアイコンと、[個人アプリケーション] サブパネルの端末エミュレータ・コントロールが使用するのと同じヘルプ・トピックを使用します。

一般的なフロントパネルの構成

フロントパネルのパネル構文により、次の作業を実行できます。

- フロントパネルの位置を変更する
- ウィンドウ装飾を変更する
- コントロールの一般的な外観と動作を設定する

デフォルトのパネルの記述は、`/usr/dt/appconfig/types/`*language*`/dtwm.fp` にあります。

その他の詳しい情報については、`dtfpfile(4X)` のマニュアル・ページを参照してください。

一般的な手順

1. 新しいフロントパネル構成ファイルを /etc/dt/appconfig/types/*language* または *HomeDirectory*/.dt/types に作成します。

2. デフォルトのパネルの記述を /usr/dt/appconfig/types/*language*/dtwm.fp から新しいファイルにコピーします。

3. パネルの記述を編集します。

新しいパネルの記述はデフォルトに優先します。

▼ デフォルトのフロントパネル位置を変更するには

- 位置を指定するには、パネル定義の PANEL_GEOMETRY フィールドを使用します。

たとえば、次のパネルは右上隅にあります。

```
PANEL SpecialFrontPanel
 {
   PANEL_GEOMETRY       -1+1
   …
 }
```

▼ メイン・パネルのコントロールにラベルを付けるには

1. パネル定義に次の行を追加します。

```
DISPLAY_CONTROL_LABELS   True
```

2. 各コントロールに LABEL フィールドを追加します。

LABEL が指定されていない場合は、*control_name* を使用します。

▼ コントロールのクリック動作を変更するには

- コントロールの PUSH_ACTION を実行する方法を指定するには、パネル定義の CONTROL_BEHAVIOR フィールドを使用します。

  single_click — コントロールをクリックして PUSH_ACTION を実行します。

  double_click — コントロールをダブルクリックして PUSH_ACTION を実行します。

▼ 新しいフロントパネルを作成するには

フロントパネルを大幅に変更する場合は、新しく作成した方がよいでしょう。

組み込みのフロントパネル・コンポーネントとの競合を避けるために、新しいフロントパネルでは、パネルとその他のコンテナに新しい名前を付けてください。

1. 新しいフロントパネル用のパネル・コンポーネントを作成します。固有の名前を指定します。

```
PANEL front_panel_name
    {
      …
    }
```

2. 新しいコンテナ名を使用して、新しいボックスとコントロールを作成します。

   既存のコンポーネントを使用する場合は、それらの定義をコピーしてから CONTAINER_NAME の値を変更する必要があります。

3. ワークスペース・メニューから **[ワークスペースマネージャの再起動]** を選択します。

3 列の個人用フロントパネルの作成例

次の例では、デフォルトのフロントパネルを、コントロールが 3 列になるように変更します。

1. /usr/dt/appconfig/types/***language***/dtwm.fp を ***HomeDirectory***/.dt/types/MyFrontPanel.fp にコピーします。ファイルに書き込み権を与えます。

   このファイルを編集して、新しいフロントパネルを提供します。

2. フロントパネル名を変更します。

```
PANEL NewFrontPanel
```

3. **Top** という名前のボックス名を変更し、そのコンテナ名を編集します。

```
BOX NewFrontPanelTop
    {
      CONTAINER_NAME     NewFrontPanel
      POSITION_HINTS     first
      …
    }
```

4. 中段と下段のボックス定義を追加します。

```
BOX NewFrontPanelMiddle
    {
      CONTAINER_NAME      NewFrontPanel
      POSITION_HINTS      second
    }


BOX NewFrontPanelBottom
    {
      CONTAINER_NAME      NewFrontPanel
      POSITION_HINTS      second
    }
```

5. 次のコントロールの CONTAINER_NAME を NewFrontPanelTop に変更します。
   - Clock (時計)
   - Date (カレンダ)
   - Home (ホーム・フォルダ)
   - TextEditor (テキストエディタ)
   - Mail (メール)
6. 次のコントロールの CONTAINER_NAME を NewFrontPanelBottom に変更します。
   - Printer (デフォルト)
   - Style (デスクトップ・スタイル)
   - Applications (アプリケーション)
   - Help (ヘルプ・マネージャ)

- Trash (ごみ箱)

7. **Switch (スイッチ)** の `CONTAINER_NAME` を `NewFrontPanelMiddle` に変更します。

8. ワークスペース・メニューの **[ウィンドウ]** サブメニューから **[ワークスペースマネージャの再起動]** を選択します。

第 16 章

ワークスペース・マネージャのカスタマイズ

この章では、デスクトップ・ワークスペース・マネージャのカスタマイズの方法について説明します。

ワークスペース・マネージャは、デスクトップが提供するウィンドウ・マネージャです。他のウィンドウ・マネージャのように、次のものをコントロールします。

- ウィンドウ枠コンポーネントの外観
- ウィンドウの重なり順やフォーカス動作などのウィンドウの動作
- キー割り当てとボタン割り当て
- アイコン化されたウィンドウの外観
- ワークスペース・メニューとウィンドウ・メニュー

さらに、次のデスクトップ・コンポーネントをコントロールします。

- 「ワークスペース」。いくつものワークスペースをコントロールし、ワークスペースごとにどのウィンドウを開いているか監視します。
- 「ワークスペース背景」。スタイル・マネージャを使用して背景を変更しますが、背景の管理はワークスペース・マネージャの機能の 1 つです。

- 「フロントパネル」。フロントパネルは独自の構成ファイルを使用しますが、そのファイルはワークスペース・マネージャが作成および管理します。

これらのほとんどは、スタイル・マネージャで変更できます。スタイル・マネージャは、よく行われる変更を簡単に素早く実行できるようにします。他のリソースは手動で設定しなければなりません。

ワークスペース・マネージャは `dtwm` です。Motif ウィンドウ・マネージャに基づきます。

- ワークスペース・マネージャについては、`dtwm(1)` と `dtwmrc(4)` のマニュアル・ページを参照してください。
- ワークスペース・マネージャのリソースの設定については、293ページの「アプリケーション・リソースの設定」を参照してください。
- フロントパネル構成ファイルについては、第 15 章を参照してください。

リソース設定の追加情報については、293ページの「アプリケーション・リソースの設定」を参照してください。

ワークスペース・マネージャ構成ファイル

ワークスペース・マネージャは、ウィンドウ・メニュー、ワークスペース・メニュー、ボタン割り当て、および構成ファイルからのキー割り当てに関する情報を獲得します。

ワークスペース・マネージャは、次のいずれかのファイルを使用します。

- 個人用ファイル: *HomeDirectory*`/.dt/dtwmrc`
- システム・カスタム・ファイル: `/etc/dt/config/`*language*`/sys.dtwmrc`
- 組み込みファイル: `/usr/dt/config/`*language*`/sys.dtwmrc`

ワークスペース・マネージャは、上記に示した順で構成ファイルを検索し、最初に見つけたファイルを使用します。

2 つ以上のセッション言語を使用するユーザは、個人用の言語依存構成ファイル *HomeDirectory*`/.dt/`*language*`/dtwrmc` を *HomeDirectory*`/.dt/dtwmrc` より優先させて作成できます。

▼ 個人用構成ファイルを作成または変更するには

個人用ワークスペース・マネージャ構成ファイルは、*HomeDirectory*/.dt/dtwmrc です。このファイルが存在している場合は使用します。

1. **[デスクトップツール] アプリケーション・グループの [Dtwmrc の編集]** をダブルクリックします。

   すでに個人用 dtwmrc ファイルがある場合は、エディタに読み込まれます。ファイルがない場合は、sys.dtwmrc が *HomeDirectory*/.dt/dtwmrc にコピーされてからエディタに読み込まれます。

2. ファイルを編集します。

3. エディタを終了します。

   ファイルは、その元のソースに関わらず、個人用 dtwmrc として保存されます。

▼ システム共通構成ファイルを作成するには

システム共通のワークスペース・マネージャ構成ファイルは /etc/dt/config/*language*/sys.dtwmrc です。

- /usr/dt/config/***language***/sys.dtwmrc を /etc/dt/config/***language***/sys.dtwmrc へコピーします。

注 - *HomeDirectory*/.dt/dtwmrc が存在する場合は、このファイルを使用しません。

▼ 他のファイルを取り込む (参照する) には

- 次の構文を使用します。

```
include
 {
   path
   path
   ...
 }
```

たとえば次の行は、/users/ellen/mymenu ファイルを参照します。

```
include
  {
     /users/ellen/mymenu
  }
```

include 文は、構成ファイルをすべてコピーせずに機能を追加する場合に便利です。たとえば、すべての構成ファイルを管理せずに新しいキー割り当てを作成する場合、次の内容の *HomeDirectory*/.dt/dtwmrc ファイルを作成できます。

```
include
 {
  /etc/dt/config/C/sys.dtwmrc
 }
Keys DtKeyBindings
 {
   Alt<Key>F5 root  f.menu  Applications
 }
Menu Applications
 {
    "GraphicsApp" f.exec "/usr/bin/GraphicsApp/GApp"
    …
 }
```

▼ ワークスペース・マネージャを再起動するには

構成ファイルに対する変更内容を有効にするには、必ずワークスペース・マネージャを再起動してください。

- **ワークスペース・メニューから [ワークスペースマネージャの再起動] を選択します (ポインタが背景上にあるときにマウス・ボタン 3 を押します)。**

ワークスペースのカスタマイズ

ワークスペース名やワークスペース数の変更など、ワークスペースのカスタマイズの多くは、デスクトップのインタフェースを使用して実行できます。ただし、システム共通デフォルトを設定するリソースは、ワークスペース・マネージャが提供します。

▼ システム共通ベースのワークスペース数を変更するには

デフォルトのデスクトップ設定では、4 つのワークスペースが提供されます。ワークスペース・スイッチに関連付けられたポップアップ・メニューを使用して、ワークスペースの追加や削除ができます。

/usr/dt/app-defaults/C/Dtwm ファイルは、workspaceCount リソースを持っています。このリソースには、次のようにワークスペースのデフォルト数が設定されています。

```
Dtwm*0*workspaceCount: 4
Dtwm*workspaceCount:  1
```

画面 0 には、複数のワークスペースが指定されています。他の画面には、1 つのワークスペースが指定されています。

/etc/dt/config/C/sys.resources ファイルを作成して (あるいは、既に存在する場合は、そのファイルを変更して)、ワークステーション上のすべての新規ユーザのために、ワークスペースのデフォルト数を変更できます。

- **0*workspaceCount リソースを使用して、一次画面のシステム共通デフォルトを設定します。**

  ```
  Dtwm*0*workspaceCount: number
  ```

たとえば次のリソースは、一次画面のシステム共通ワークスペース数を 6 に設定します。

```
Dtwm*0*workspaceCount: 6
```

ワークスペース・マネージャのリソースの設定については、293ページの「アプリケーション・リソースの設定」を参照してください。

たとえば次のリソースは、ワークスペース数を 6 に設定します。

```
Dtwm*workspaceCount: 6
```

▼ システム共通ワークスペース名を指定するには

内部的には、ワークスペースは番号割り当て規則 ws*n* (*n* は 0、1、2 など) によって番号が付けられます。たとえば、4 つのデフォルト・ワークスペースは、内部的に ws0 から ws3 までの番号が付けられます。

♦ title リソースを使用して指定したワークスペース名を変更します。

```
Dtwm*wsn: name
```

ワークスペース・マネージャのリソースの設定については、293ページの「アプリケーション・リソースの設定」を参照してください。

たとえば次のリソースは、4 つのデフォルト・ワークスペースを指定した名前に設定します。

```
Dtwm*ws0*title:  Anna
Dtwm*ws1*title:  Don
Dtwm*ws2*title:  Julia
Dtwm*ws3*title:  Patti
```

▼ 追加背景を作成するには

1. 背景イメージを作成します。ビットマップ・ファイルまたはピックスマップ・ファイルにしてください。
2. 次のいずれかのディレクトリに背景を指定します (ディレクトリを作成しなければならない場合もあります)。
   - システム共通背景: /etc/dt/backdrops
   - 個人用背景: *HomeDirectory*/.dt/backdrops
3. ワークスペース・メニューから [ワークスペースマネージャの再起動] を選択します。

システム共通背景と個人用背景は、/usr/dt/backdrops の組み込み背景に追加されます。

既存の組み込み背景は、同じ名前の個人用背景またはシステム共通背景を作成することによって置き換えることができます。

▼ グラフィック・イメージで背景を置き換えるには

背景は、ディスプレイのルート・ウィンドウ全体を覆っています。スタイル・マネージャの [背景] ダイアログ・ボックスは、背景が透過的である NoBackdrop 設定を提供します。

すべてのワークスペース背景の背後には、1 つのルート・ウィンドウしかありません。したがって、ルート・ウィンドウにあるグラフィック・イメージは、すべてのワークスペースで存在します。どのワークスペースがルート・ウィンドウを背景で覆うかを指定できます。ただし、NoBackdrop が有効である場合、表示可能なイメージはすべてのワークスペースで同じになります。

1. グラフィック・イメージを作成します。

   これは、ルート・ウィンドウにイメージを表示するためのツールの形式でなければなりません。たとえば xsetroot を使用する場合は、ビットマップ・ファイルを作成しなければなりません。

2. ファイルが存在しない場合は、実行可能ファイル *HomeDirectory*/.dt/sessions/sessionetc を作成します。

   sessionetc ファイルは、ログインするたびに実行されます。

3. コマンドを入力して、sessionetc ファイルのイメージを表示します。

たとえば次のコマンドは、ルート・ウィンドウに指定したビットマップを表示します。

```
xsetroot -bitmap /users/ellen/.dt/icons/root.bm
```

ワークスペース・マネージャのメニュー

ワークスペース・マネージャには、次の 3 つのデフォルト・メニューがあります。

ワークスペース・メニュー — ルート・メニューとも呼ばれます。ポインタが背景にあるときに、マウス・ボタン 3 を押すと表示されます。このメニューは、ボタン割り当てによってマウス・ボタンに関連付けられています。

ウィンドウ・メニュー — ポインタがウィンドウ・メニュー・ボタン (ウィンドウ枠の左上隅) 上にあるときに、マウス・ボタン 1 または 3 を押すと表示されます。このメニューは、windowMenu リソースによってボタンに関連付けられています。

フロントパネル・メニュー — ポインタがフロントパネルのウィンドウ・メニュー・ボタン上にあるときに、マウス・ボタン 1 または 3 を押すと表示されます。

ワークスペース・マネージャのメニュー構文

ワークスペース・マネージャのメニュー構文は、次のとおりです。

```
Menu MenuName
 {
    selection1 [mnemonic] [accelerator] function [argument]
    selection2 [mnemonic] [accelerator] function [argument]
    ...
 }
```

- *selection*

  メニューに表示されるテキストまたはビットマップ。テキストにスペースを入れるときは、テキストを引用符で囲みます。ビットマップには、@*/path* 構文を使用します。

- *mnemonic*

  メニューが表示されたときに、キーボード・ショートカットとして動作する1つの文字。`form_`*character* で指定します。

- *accelerator*

  メニューが表示されているかどうかに関わらず使用可能なキーボード・ショートカット。アクセラレータの構文は *modifier*<Key> *Keyname* で、修飾子は `Ctrl`、`Shift`、`Alt` (拡張文字)、または `Lock` です。すべての可能なキー名のリストについては、システムの X11 include ディレクトリの `keysymdef.h` ファイルを参照してください。

- *function*

  選択したときに実行される関数。関数のリストについては、`dtwmrc(4)` のマニュアル・ページを参照してください。

- *argument*

  関数の引き数。詳細は、`dtwmc(4)` のマニュアル・ページを参照してください。

たとえば、次の Restore というラベルの付いたメニュー項目は、ウィンドウを元に戻します。メニューが表示されたときに R と入力してもウィンドウは復元されます。拡張文字 F5 を押しても同じです。

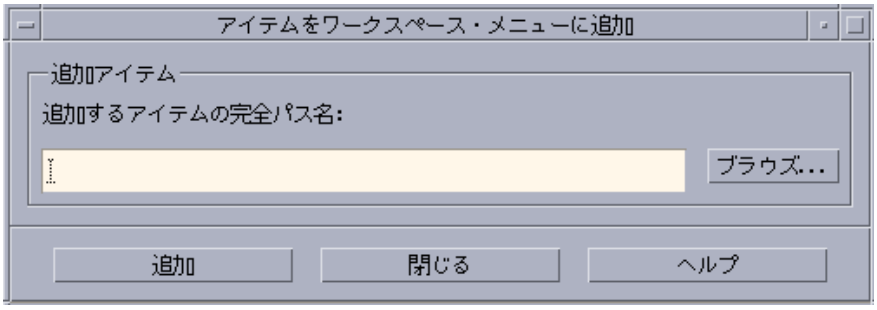
```
Restore   _R   Alt<Key> F5   f.normalize
```

注 - ワークスペース・マネージャのメニュー構文の詳細は、`dtwmrc(4)` のマニュアル・ページを参照してください。

▼ 新規メニュー項目をワークスペース・メニューに追加するには

1. フロントパネルの **[ツール]** サブパネルにある、**[メニューに項目を追加]** をクリックします。

   [アイテムをワークスペース・メニューに追加] ダイアログが表示されます。



2. 絶対パスを入力するか **[ブラウズ]** をクリックし、新しく追加するファイルのパスを選択します。

   指定したファイルは、通常はこのホストで使用可能です。このファイルは、実行可能ファイル、またはホストに登録されているデータ型のファイル (たとえばオーディオやマニュアルページ) である必要があります。

   ファイル名を含んだ絶対パスが、テキストボックスに表示されます。

3. **[追加]** をクリックして、ファイルをメニューに追加します。

   ファイルが、ワークスペース・メニューの一番上に追加されます。デフォルトのアイコン (ある場合) とファイル名が、メニュー項目のアイコンとテキストとして使用されます。

注 - ワークスペース・メニューのメニュー項目の位置を変更するには、284ページの「ワークスペース・メニューを変更するには」を参照してください。

▼ ワークスペース・メニューを変更するには

1. フロントパネルの **[ツール]** サブパネルにある **[ワークスペース・メニューのカスタマイズ]** をクリックします。

   ワークスペース・メニューの階層構造がファイル・マネージャの構成フォルダとして表示されます。このフォルダ内のファイルは各メニュー項目を表し、サブフォルダは各サブメニューを表します。ワークスペース・メニューフォルダ内の配列を変更して、ワークスペース・メニューの配列を変更します。

2. 新しい位置に移動したいメニュー項目を、該当するサブフォルダにドラッグ＆ドロップします。

   ツリー表示モードでファイル・マネージャを使用する場合のヘルプについては、『*Solaris* 共通デスクトップ環境 ユーザーズ・ガイド』を参照してください。

3. 不要なメニュー項目を削除する場合は、該当するファイルをフロントパネルのごみ箱にドラッグします。

4. メニュー項目の名前を変更したい場合は、該当するファイル名かフォルダ名を編集します。

5. すべての変更が完了したら、ファイル・マネージャの **[ファイル]** メニューから **[ワークスペース・メニューの更新]** を選択してファイル・マネージャを終了します。

   ワークスペース・メニューに、ワークスペース・メニューのフォルダで変更した内容が反映されます。

▼ 新規ワークスペース (Root) メニューを作成するには

1. 次のいずれかのファイルを開きます。

   - 個人用: *HomeDirectory*/.dt/dtwmrc
   - システム共通: /etc/dt/config/*language*/sys.dtwmrc

   これらのファイルの作成方法の詳細は、276ページの「ワークスペース・マネージャ構成ファイル」を参照してください。

2. 新規メニューを作成します。

```
Menu menu_name
    {
      …
    }
```

   詳細は、282ページの「ワークスペース・マネージャのメニュー構文」を参照してください。

3. ボタン割り当てを作成または編集して新規メニューを表示します。

   既存のメニューを新規メニューに置き換える場合は、ワークスペース・メニューを表示するボタン割り当てを編集します。

```
<Btn3Down> root   f.menu   menu_name
```

   メニューが追加メニューの場合は、新しいマウス・ボタンを割り当てます。たとえば次のようにボタンを割り当てると、背景上で [Shift] キーとマウス・ボタン 3 を押したときにメニューが表示されます。

```
Shift<Btn3Down> root   f.menu   menu_name
```

4. ワークスペース・メニューから **[ワークスペースマネージャの再起動]** を選択します。

▼ 新規ウィンドウ・メニューを作成するには

注 - ウィンドウ・メニューはワークスペース・マネージャに組み込まれ、通常はカスタマイズされません。アプリケーション間でウィンドウの動作の一貫性を保つには、ウィンドウ・メニューを大幅に変更しないでください。

---

1. 次のいずれかのファイルを開きます。
   - 個人用: *HomeDirectory*/.dt/dtwmrc
   - システム共通: /etc/dt/config/*language*/sys.dtwmrc

   これらのファイルの作成方法の詳細は、276ページの「ワークスペース・マネージャ構成ファイル」を参照してください。

2. 新規メニューを作成します。

```
Menu menu_name
    {
      …
    }
```

3. windowMenu リソースを使用して新規メニューを指定します。

```
Dtwm*windowMenu: menu_name
```

4. ワークスペース・メニューから **[ワークスペースマネージャの再起動]** を選択します。

ボタン割り当てのカスタマイズ

ボタン割り当ては、ウィンドウ・マネージャ関数と、マウス・ボタンのオペレーションおよび可能なキーボード修飾キーとを関連付けることです。ボタン割り当ては、すべてのワークスペースに適用されます。

デスクトップのデフォルトのボタン割り当ては、DtButtonBindings というボタン割り当てセットのワークスペース・マネージャ構成ファイルに定義されています。

```
Buttons DtButtonBindings
 {
   …
 }
```

ボタン割り当て構文

ボタン割り当ての構文は次のとおりです。

```
Buttons ButtonBindingSetName
{
      [modifier] <button_nameMouse_action>     context   function   [argument]
      [modifier] <button_nameMouse_action>     context   function   [argument]
      …
}
```

- *button_name*

  Btn1 — 左ボタン

  Btn2 — 中央ボタン (3 ボタン・マウスの場合) または左右ボタン (2 ボタン・マウスの場合)

  Btn3 — 右ボタン

  Btn4 — 3 ボタン・マウスの場合のボタン 1 とボタン 2

  Btn5 — 3 ボタン・マウスの場合のボタン 2 とボタン 3

- *modifier*

  Ctrl、Shift、Alt、Lock

- *mouse_action*

  Down — マウス・ボタンを押し続ける

  Up — マウス・ボタンを離す

  Click — マウス・ボタンをダブルクリックする

  Click2 — マウス・ボタンを押して離す

  Drag — マウス・ボタンを押しながらドラッグする

- *context*

  割り当てを有効にするには、ポインタがどこにあればいいかを示します。必要に応じて、複数の内容は「|」文字で区切ります。

  root — ワークスペースウィンドウ

window — クライアント・ウィンドウまたはウィンドウ枠

frame — 内容を除くウィンドウ枠

icon — アイコン

title — タイトル・バー

app — クライアントのウィンドウ (枠を除く)

- *function*

  ウィンドウ・マネージャ関数の 1 つ。有効な関数のリストについては、dtwmrc(4) のマニュアル・ページを参照してください。

- *argument*

  任意のウィンドウ・マネージャ関数の必須引き数。詳細は、dtwmrc(4) のマニュアル・ページを参照してください。

たとえば次の行を入力すると、ポインタが (クライアントのウィンドウ内ではなく) ワークスペース・ウィンドウにあるときにマウス・ボタン 3 を押すと、DtRootMenu に記述されたメニューが表示されます。

```
<Btn3Down>       root      f.menu      DtRootMenu
```

注 - ボタン割り当て構文の詳細は、dtwmrc(4) のマニュアル・ページを参照してください。

▼ ボタン割り当てを追加するには

1. **次のいずれかのファイルを開きます。**

   - **個人用:** *HomeDirectory*/.dt/dtwmrc
   - **システム共通:** /etc/dt/config/*language*/sys.dtwmrc

   これらのファイルの作成方法の詳細は、276**ページの「ワークスペース・マネージャ構成ファイル」を参照してください。**

2. **ボタン割り当てを** DtButtonBindings **定義に追加します。**

   クリックおよび押す操作という異なる関数を同じボタンに割り当てないでください。また、2 つ以上の関数を同じボタンおよび内容に割り当てないでください。

3. ワークスペース・メニューから **[ワークスペースマネージャの再起動]** を選択します。

▼ 新規ボタン割り当てセットを作成するには

1. 次のいずれかのファイルを開きます。
   - 個人用: *HomeDirectory*/.dt/dtwmrc
   - システム共通: /etc/dt/config/*language*/sys.dtwmrc

   これらのファイルの作成方法については、276ページの「ワークスペース・マネージャ構成ファイル」を参照してください。

2. 新規ボタン割り当てセットを作成します。詳細は、287ページの「ボタン割り当て構文」を参照してください。

3. buttonBindings リソースに新しい名前を設定します。

   ```
   Dtwm*buttonBindings: ButtonBindingsSetName
   ```

4. ワークスペース・メニューから **[ワークスペースマネージャの再起動]** を選択します。

注 - 既存のボタン割り当てを新規のボタン割り当てに置き換えます。保持したいボタン割り当てがある場合は、DtButtonBindings からコピーします。

キー割り当てのカスタマイズ

「キー割り当て」とも呼ばれる「キーボード割り当て」は、キーの組み合わせとワークスペース・マネージャ関数とを関連付けます。キー割り当ては、すべてのワークスペースに適用されます。

注 - キーボード割り当てとして共通キーの組み合わせを使用するときは注意してください。たとえば、[Shift] キーと [A] キーを押すと、通常は現在のウィンドウに文字「A」が表示されます。[Shift] + [A] キーを関数に割り当てた場合は、通常の動作をしません。

デスクトップのデフォルト・キー割り当て

デスクトップのデフォルト・キー割り当ては、`DtKeyBindings` というキー割り当てセットのワークスペース・マネージャ構成ファイルに定義されています。

```
Keys DtKeyBindings
 {
  …
 }
```

キー割り当て構文

キー割り当ての構文は次のとおりです。

```
Keys KeyBindingSetName
 {
     [Modifiers] <Key>key_name     context     function    [argument]
     [Modifiers] <Key>key_name     context     function    [argument]
     …
 }
```

- *Modifiers*

  `Ctrl`、`Shift`、`Alt`、`Lock`。複数の修飾子を指定できますが、それぞれをスペースで区切ります。

- *key_name*

  関数をマップするキー。文字または数字のキーの場合、*key_name* 名は通常キーの上に刻印されます。たとえば「a」というキーは「a」、「2」というキーは「2」と名付けられます。同様に「Tab」キーは「Tab」、「F3」キーは「F3」です。

  他のキーについては、名前が省略されずに書かれます。たとえば、「+」キーは `plus` と表示されます。システム依存ディレクトリにある `keysymdef.h` ファイルには、キー名に関する追加情報があります。

- *context*

このアクションを有効にするためのキーボード・フォーカスを必ず持つ要素。割り当てを 2 つ以上の内容に適用すると、内容を連結できます。複数の内容は「|」文字で区切ります。

root — ワークスペース背景

window — クライアントのウィンドウ

icon — アイコン

- *function*

  ウィンドウ・マネージャ関数。有効な関数のリストについては、dtwmrc(4) のマニュアル・ページを参照してください。

- *argument*

  任意のウィンドウ・マネージャ関数の必須引き数。詳細は、dtwmrc(4) のマニュアル・ページを参照してください。

たとえば、次のようなキー割り当てをすると、[Alt] + [F6] キーを押すことにより、アプリケーションの次の一時ウィンドウへキーボード・フォーカスを切り替えることができます。

```
Alt<Key>F6      window      f.next_key      transient
```

注 - キー割り当て構文の詳細は、dtwmrc(4) のマニュアル・ページを参照してください。

▼ キー割り当てセットをカスタマイズするには

1. 次のいずれかのファイルを開きます。

   - 個人用: *HomeDirectory*/.dt/dtwmrc
   - システム共通: /etc/dt/config/*language*/sys.dtwmrc

   これらのファイルの作成方法の詳細は、276ページの「ワークスペース・マネージャ構成ファイル」を参照してください。

2. 一意の ***KeyBindingSetName*** で新規キー割り当てセットを作成します。デスクトップのデフォルト・キー割り当てセット DtKeyBindings をガイドとして使用します。

3. keyBindings リソースに新規セット名を設定します。

   ```
   Dtwm*keyBindings: KeyBindingSetName
   ```

4. ワークスペース・メニューから **[ワークスペースマネージャの再起動]** を選択します。

注 - 既存のキー割り当ては、新規のキー割り当てに置き換わります。保持したいキー割り当てがある場合は、`DtKeyBindings` から新規セットにコピーします。

デフォルト動作とカスタマイズ動作との切り替え

Motif 1.2 デフォルト動作と CDE デスクトップ・ウィンドウ動作を切り替えるには、次のようにします。

1. **[Alt] + [Shift] + [Ctrl] + [!]** キーを押します。
2. ダイアログ・ボックスの **[了解]** をクリックします。

デフォルト動作に切り替えると、フロントパネル、任意のカスタマイズ・キー、およびボタン割り当てが削除されます。

第 17 章

アプリケーションのリソース、フォント、およびカラーの処理

スタイル・マネージャを使用するか、追加のフォントやカラーのリソースをカスタマイズすることによって、ディスプレイに対していろいろなカラーやフォントを選択できます。この章では、フォントとカラーのリソースをカスタマイズする方法について説明します。

また、デスクトップ・テキスト・エディタ (dtpad) やメール・プログラム (dtmail) などの DtEditor ウィジェット・アプリケーションのスタイル変換方法と、これらの変換と競合する DtEditor ウィジェット・アプリケーションのメニュー・アクセラレータに代わるものを指定する方法についても説明します。

アプリケーション・リソースの設定

アプリケーションはリソースを使用して、外観と動作を設定します。たとえば、スタイル・マネージャ (dtstyle) が提供するリソースにより、システムがカラー・パレットに関する情報が入っているファイルを検索する場所を指定できます。

```
dtstyle*paletteDirectories: /usr/dt/palettes/C HomeDirectory/.dt/palettes
```

デスクトップ・アプリケーションのデフォルトのリソース・ファイルは、/usr/dt/app-defaults/*language* ディレクトリにあります。

▼ システム共通リソースを設定するには

- ♦ リソースをファイル /etc/dt/config/***language***/sys.resources に追加します **(**ファイルを作成しなければならない場合もあります**)**。

たとえば、/etc/dt/config/C/sys.resources に次のように指定します。

```
AnApplication*resource: value
```

この場合、次のログイン時にリソース AnApplication*resource が各ユーザの RESOURCE_MANAGER 属性に設定されます。

注 - スタイル・マネージャ・リソースの詳細は、dtstyle(1X) のマニュアルページを参照してください。メール・プログラム・リソースの詳細は、dtmail(1X) のマニュアルページを参照してください。

▼ 個人用リソースを設定するには

1. リソースをファイル ***HomeDirectory***/.Xdefaults に追加します。
2. **[**デスクトップツール**]** アプリケーション・グループにある **[**リソースの再読込み**]** をダブルクリックします。

デスクトップがリソースを読み込む方法

リソースは、セッション・マネージャによってセッションの起動時に読み込まれます。セッション・マネージャがリソースを RESOURCE_MANAGER に読み込む方法の詳細は、30ページの「セッション・リソースの読み込み」を参照してください。

プロセス・マネージャ・リソース

次のプロセス・マネージャ・リソースが使用可能です。

- sampleNowTR
- postPopupMenuTR

- selectNextProcessTR
- selectPrevProcessTR
- selectFirstProcessTR
- selectLastProcessTR
- killSelectedProcessTR

UNIX 割り当ての定義

デフォルトでは UNIX 割り当ては利用できません。

▼ EMACS スタイル変換を指定するには

次のように指定します。

- デスクトップのテキスト・エディタ (dtpad) やメール・プログラム (dtmail) などの DtEditor ウィジェット・アプリケーションの EMACS スタイル変換
- これらの変換と競合する DtEditor ウィジェット・アプリケーションのメニュー・アクセラレータに代わるもの

1. 次の行を ***HomeDirectory***/.Xdefaults ファイルに追加します。

   ```
   #include "/usr/dt/app-defaults/language/UNIXbindings"
   ```

   language は、LANG 環境変数の値です。

2. セッションを再起動します。

▼ EMACS スタイル変換を変更するには

1. ファイル /usr/dt/app-defaults/***language***/UNIXbindings の内容を ***HomeDirectory***/.Xdefaults に挿入します。
2. .Xdefaults ファイルの割り当てを編集します。
3. 編集を終了してから、セッションを再起動します。

UNIXbindings ファイルが提供する UNIX 割り当て

/usr/dt/app-defaults/*language*/UNIXbindings ファイルは、次の割り当てを行います。

注 - UNIX 割り当てが可能な場合、[Delete] キーは直前の文字を削除し、[Shift] + [Delete] キーは次の文字を削除します。

表 17–1 は、dtpad が無効にする UNIX 割り当てと競合するメニュー・アクセラレータとアクセラレータ・テキストのリストです。

表 17–1 dtpad が無効にするキー

メニュー・アクセラレータとアクセラレータ・テキスト	無効キー
Dtpad*fileMenu.print.acceleratorText:	
Dtpad*fileMenu.print.accelerator:	
Dtpad*editMenu.undo.acceleratorText:	Ctrl+_
Dtpad*editMenu.undo.accelerator:	Ctrl<Key>_
Dtpad*editMenu.paste.acceleratorText:	Shift+Insert
Dtpad*editMenu.paste.accelerator:	Shift<Key>osfInsert
Dtpad*editMenu.findChange.acceleratorText:	Ctrl+S
Dtpad*editMenu.findChange.accelerator:	Ctrl<Key>s

表 17–2 は、UNIX 割り当てと競合するメニュー・アクセラレータとアクセラレータ・テキストの dtmail メール作成ウィンドウの無効キーのリストです。

表 17–2 dtmail メール作成ウィンドウの無効キー

メニュー・アクセラレータとアクセラレータ・テキスト	無効キー
`Dtmail*ComposeDialog*menubar*Edit.Undo.acceleratorText:`	Ctrl+_
`Dtmail*ComposeDialog*menubar*Edit.Undo.accelerator:`	Ctrl<Key>_
`Dtmail*ComposeDialog*menubar*Edit.Paste.acceleratorText:`	Shift+Insert
`Dtmail*ComposeDialog*menubar*Edit.Paste.accelerator:`	Shift<Key>osfInsert
`Dtmail*ComposeDialog*menubar*Edit.Find/Change.acceleratorText:`	Ctrl+S
`Dtmail*ComposeDialog*menubar*Edit.Find/Change.accelerator:`	Ctrl<Key>s

次の変換は、(GNU スタイル) EMACS コントロールと [Meta] キー割り当て、および追加の割り当てを提供します。適切な場合は、[Shift] キーを通常の割り当てと組み合わせて使用して、オペレーションの方向を反対にすることもできます。たとえば、[Ctrl] + [F] キーは通常 1 文字前に移動するので、[Ctrl] + [Shift] + [F] キーは 1 文字後ろにカーソルを移動します。

追加の割り当ては次のとおりです。

Ctrl+comma — 1 語後ろへ (backward-word)

Ctrl+Shift+comma — 1 語前へ (forward-word)

Ctrl+period — 1 語前へ (forward-word)

Ctrl+Shift+period — 1 語後ろへ (backward-word)

Ctrl+Return — ファイルの最後へ (end-of-file)

Ctrl+Shift+Return — ファイルの最初へ (beginning-of-file)

GNU EMACS は、[Delete] キーに対して delete-next-character() ではなく delete-previous-character() を割り当てます。[Meta] + [F] キーは通常は [ファイル] メニューのニーモニックのため、forward-word() への割り当ては無視されます。1 語前 (forward-word) の割り当てのうち 1 つを使用します (たとえば、[Ctrl] + [period])。

表 17–3 は、`DtEditor.text` 変換のリストです。

表 17–3 DtEditor.text 変換

修飾キー	キー	アクション・ルーチン
c ~s	<Key>a:	beginning-of-line()\n\
c s	<Key>a:	end-of-line()\n\
c ~s	<Key>b:	backward-character()\n\
c s	<Key>b:	forward-character()\n\
c ~s	<Key>b:	backward-character()\n\
c s	<Key>b:	backward-word()\n\
m ~s	<Key>b:	backward-word()\n\
m s	<Key>b:	forward-word()\n\
c ~s	<Key>d:	delete-next-character()\n\
c s	<Key>d:	delete-previous-character()\n\
m ~s	<Key>d:	kill-next-word()\n\
m s	<Key>d:	kill-previous-word()\n\
c ~s	<Key>e:	end-of-line()\n\
c s	<Key>e:	beginning-of-line()\n\
c ~s	<Key>f:	forward-character()\n\
c s	<Key>f:	backward-character()\n\
m ~s	<Key>f:	forward-word()\n\
m s	<Key>f:	backward-word()\n\
c	<Key>j:	newline-and-indent()\n\
c ~s	<Key>k:	kill-to-end-of-line()\n\
c s	<Key>k:	kill-to-start-of-line()\n\
c	<Key>l:	redraw-display()\n\
c	<Key>m:	newline()\n\

表 17–3　DtEditor.text 変換　続く

修飾キー	キー	アクション・ルーチン
c s	<Key>n:	process-up()\n\
c ~s	<Key>n:	process-down()\n\
c	<Key>o:	newline-and-backup()\n\
c ~s	<Key>p:	process-up()\n\
c s	<Key>p:	process-down()\n\
c ~s	<Key>u:	kill-to-start-of-line()\n\
c s	<Key>u:	kill-to-end-of-line()\n\
c ~s	<Key>v:	next-page()\n\
c s	<Key>v:	previous-page()\n\
m ~s	<Key>v:	previous-page()\n\
m s	<Key>v:	next-page()\n\
c	<Key>w:	kill-selection()\n\
c ~s	<Key>y:	unkill()\n\
m	<Key>]:	forward-paragraph()\n\
m	<Key>[:	backward-paragraph()\n\
c ~s	<Key>comma:	backward-word()\n\
c s	<Key>comma:	forward-word()\n\
m	<Key>\\<:	beginning-of-file()\n\
c ~s	<Key>period:	forward-word()\n\
c s	<Key>period:	backward-word()\n\
m	<Key>\\>:	end-of-file()\n\
c ~s	<Key>Return:	end-of-file()\n\
c s	<Key>Return:	beginning-of-file()\n\

表 17–3　DtEditor.text 変換　続く

修飾キー	キー	アクション・ルーチン
~c ~s ~m ~a	<Key>osfDelete:	delete-previous-character()\n\
~c s ~m ~a	<Key>osfDelete:	delete-next-character()

フォントの処理

スタイル・マネージャの [フォント] ダイアログ・ボックスを使用して、すべてのアプリケーションに対してフォントグループとサイズを選択できます。コマンド行でフォントを指定するか、次を実行してリソースを使用することもできます。

- 個々のアプリケーションにフォントのグループとリソースを設定する
- [フォント] ダイアログ・ボックスによって異なるフォントを使用するように割り当てる

「フォント」は、テキスト文字が印刷または表示される型のスタイルです。デスクトップには、異なるスタイルやサイズのさまざまな種類のフォントが入っています。

「ビットマップ・フォント」は、ドットのマトリックスから成ります。(デフォルトでは、スタイル・マネージャはビットマップ・フォントだけを構成します。) フォントは、完全に 1 つのファイルに収められます。多くのファイルには、いろいろなサイズ、スラント、線の太さが入っている必要があります。

フォントは、リソースの値およびコマンドへのパラメータとして指定されます。XLFD 名 (論理フォント名) は、希望のフォントを要求するための方法です。システムは、指定された記述と最もよく一致するフォントを探し出します。

デスクトップ・フォント・リソースの設定

スタイル・マネージャの [フォント] ダイアログ・ボックスにより、テキストのエントリやラベルなどに対してフォントを (7 種類のサイズまで) 設定できます。また、フォント・グループを追加または削除できます。

[フォント] ダイアログ・ボックスが設定するリソース

フォントを選択すると、次のリソースが RESOURCE_MANAGER 属性に書き込まれます。

- SystemFont は、メニュー・バー、メニュー区画、プッシュ・ボタン、トグル・ボタン、ラベルなどのシステム領域に使用します。次のリソースは SystemFont によって設定されます。

  *FontList — デスクトップのクライアントと、OSF/Motif ツールキットを使用して作成された、その他のクライアントのシステム領域に表示されます。

- UserFont は、ウィンドウに入力するテキストに使用します。次のリソースが UserFont によって設定されます。

  *Font — X アプリケーションの以前のバージョンをサポートします。

  *FontSet — 一次設定です。

  *XmText*FontList — テキスト・エントリ・ボックスに表示されます。

  *XmTextField*FontList — テキスト・エントリ・ボックスに表示されます。

[フォント] ダイアログ・ボックスが使用するリソース

[フォント] ダイアログ・ボックスでの各選択に対して使用されるフォントは、/usr/dt/app-defaults/Dtstyle リソース・ファイルで指定します。最高 7 種類までのサイズを指定できます。

NumFonts — [フォント] ダイアログ・ボックスにあるフォント・サイズの数

SystemFont[1-7] — SystemFont の [フォント] ダイアログ・ボックス選択に特定フォントを割り当てる、最高 7 種類までのリソース

UserFont[1-7] — UserFont の [フォント] ダイアログ・ボックス選択に特定フォントを割り当てる、最高 7 種類までのリソース

注 - これらのリソースのデフォルト・フォントは、さまざまなディスプレイで読み込み可能なように選択されています。アプリケーション用に特定フォントを使用する場合は、これらのデスクトップ・フォントを変更するよりもアプリケーションのフォント・リソースでフォントを設定します。

アプリケーション・フォントの詳細は、DtStdAppFontNames(5) と DtStdInterfaceFontNames(5) のマニュアル・ページを参照してください。

▼ 使用可能なフォントを表示するには

1. 次の行を入力します。

   ```
   xlsfonts [-options] [-fn pattern]
   ```

   システムで使用できる XLFD 名とフォント別名が表示されます。ビットマップ・フォントは、14 個の XLFD フィールドの全部の値を示します。スケーラブル・タイプフェースは、*PixelSize*、 *PointSize*、*ResolutionX*、*ResolutionY* の位置にあるゼロを示します。

2. 特定フォントをチェックするには、xlsfonts のパターン一致機能を使用します。ワイルドカードを使用して、一致させるつもりがないパターンの一部を置き換えます。

3. xlsfonts が dt で始まるフォント名を表示しない場合は、フォント・パスにデスクトップ・フォントが入っていません。デスクトップ・フォントを使用可能なフォントに入れるには、次のコマンドを入力します。

   ```
   xset +fp directory name
   ```

   directory name は、デスクトップ・フォントが入っているディレクトリです。セッション起動によって設定されるデフォルトの位置は、/usr/dt/config/xfonts/*language* です。

追加情報については、次を参照してください。

- xset(1) と xlsfonts(1) のマニュアル・ページには、使用可能なオプションのリストがあります。
- 『*Using the X Window System*』では、フォント別名と xset クライアントについて説明しています。

▼ コマンド行でフォントを指定するには

- –xrm コマンド行オプションを使用して、特定クライアントのフォント・リソースを指定します。たとえば、次のようになります。

```
application name -xrm "*bitstream-charter-medium-r-normal-8-88-75-75-p-45-iso8859-1"
```

論理フォント名 (XLFD)

フォントは、ダッシュ (-) で区切られた 14 個の異なる特性をリストすることによって指定されます。これは、論理フォント名 (XLFD) と呼ばれます。いくつかの場合、リストにある属性は、ワイルドカード「*」に、属性内の文字はワイルドカード「?」にそれぞれ置き換えることができます。表 17–4 は、フォント属性文字列指定のリストです。

属性文字列指定の形式は次のとおりです。

"*-Foundry-FamilyName-WeightName- Slant-SetwidthName-AddStyleName-PixelSize-PointSize-ResolutionX-ResolutionY-Spacing-AverageWidth-CharSetRegistry-CharSetCoding*"

表 17–4 フォント属性文字列指定

属性文字列	定義
Foundry	フォント作成者を識別する文字列
FamilyName	フォントの商標登録された名前を識別する文字列
WeightName	ボールドなどの、フォントの相対的な線の太さを指定する文字列
Slant	スラントの方向を記述するコード R (ローマン – スラントなし) I (イタリック – 右スラント) O (オブリック – 右スラント) RI (リバース・イタリック – 左スラント) RO (リバース・オブリック – 左スラント)
SetwidthName	圧縮または拡張などの、幅について記述する文字列
AddStyleName	フォントを識別するためだけに必要な追加情報を提供する文字列
PixelSize	全角の M が入る正方形 (エム・スクウェア) のサイズをピクセル単位で指定する整数
PointSize	全角の M が入る正方形 (エム・スクウェア) のサイズを 0.1 ポイント単位で指定する整数
ResolutionX	水平解像度をピクセル単位で指定する整数

表 17–4 フォント属性文字列指定 続く

属性文字列	定義
ResolutionY	垂直解像度をピクセル単位で指定する整数
Spacing	ユニット間の間隔を指定するコード M (モノスペース – 固定幅) P (プロポーショナル・スペース – 可変幅) C (キャラクタ・セル)
AverageWidth	1/10 ピクセル単位で平均幅を指定する整数
CharSetRegistry	フォントのエンコーディングを登録した登録権限を識別する文字列
CharSetEncoding	指定した登録の文字セットを識別する文字列

例

次の論理フォント名は、ISO8859-1 標準エンコーディングをサポートする Bitstream によって作成された charter という名前のフォントについて説明します。

```
-bitstream-charter-medium-r-normal--8-80-75-75-p-45-iso8859-1
```

これは、ミディアム・ウェイトで、特別にスラントしていない、通常の幅です。フォントはプロポーショナルで、そのサイズは 8 ピクセルまたは 8.0 ポイントの全角の M が入る (エム・スクエア) です。水平解像度と垂直解像度は、両方とも 75 ピクセルです。文字幅の平均は 45 1/10 ピクセルまたは 4.5 ピクセルです。

これらの文字列の一部は、ワイルドカードで置き換えることができます。システムは指定した部分と一致する最初に見つけたフォントを使用します。

8 ピクセルの charter フォントを希望する場合、次の行を使用できます。

```
*-charter-*-*-*-*-8-*
```

選択したフォント・グループ属性の表示

スタイル・マネージャの [フォント] ダイアログ・ボックスから [属性] ボタンを選択することによって、次のフォント・グループ属性を表示できます。

- Font Group (フォント・グループ)

- Size (サイズ)
- Alias (別名)
- Alias XLFD (別名 XLFD)
- Alias Location (別名位置)
- Font (フォント)
- Font XLFD (フォント XLFD)

ユーザのフォント・グループ・ファイルシステムの格納

ユーザがフォント・グループを追加した場合、そのフォント・グループは次の位置に格納されます。

HomeDirectory/.dt/sdtfonts/*host*/*locale*/*typeface-nnnnnn*

host は、ローカル・ワークステーションのホスト名です。

locale は、ユーザの現在のロケールです (たとえば「C」や「ja」など)。

typeface-nnnnnn は、選択したフォントと固有の生成番号から決定される名前です。

このフォント・グループ・ディレクトリには、次の 3 つのファイルがあります。

- `fonts.alias`
- `fonts.dir`
- `sdtfonts.group`

`fonts.alias` ファイルと `fonts.dir` ファイルは、X サーバのフォント・パスへ追加するのに適切な、通常の X11 フォント・ファイルです。`sdtfonts.group` ファイルには、ユーザ指定のフォント・グループ名が入っています。

システム管理者のフォント・グループの作成

他のユーザがワークステーション上のフォント・グループにアクセスできるようにするために、システム管理者は、フォント・グループを `/etc/dt/sdtfonts/`*locale* ディレクトリか `/usr/openwin/lib/X11/stdfonts/`*locale* ディレクトリにコピーできます。セッション・マネージャは、まず *HomeDirectory*`/.dt/stdfonts/`*host*/*locale* を、次に `/etc/dt/sdtfonts/`*locale* を、最後に `/usr/openwin/lib/X11/stdfonts/`*locale* を探します。

カラーの管理

この節では次の内容を説明します。

- スタイル・マネージャがディスプレイ・カラーを設定する方法
- デスクトップ・カラーの使用をコントロールするためにスタイル・マネージャが使用するリソース

カラー・パレット

パレットは、カラー・セットのグループから成ります。現在のパレットのカラー・セットは、スタイル・マネージャの [カラー] ダイアログ・ボックスに表示されます。

各パレットに 1 つのファイルが存在します。paletteDirectories リソースは、パレット・ファイルが入っているディレクトリを指定します。デフォルトでは、このリソースには次のものが入っています。

- 組み込みパレット: /usr/dt/palettes
- システム共通パレット: /etc/dt/palettes
- 個人用パレット: *HomeDirectory*/.dt/palettes

カラー・セット

現在のパレットに設定されている各カラーは、スタイル・マネージャの [カラー] ダイアログ・ボックスにあるカラー・ボタンによって表されます。各カラーは、1 から 8 までのカラー・セット ID によって識別します。

図 17–1　HIGH_COLOR のためのカラー・セット ID 値

各カラー・セットは、最高 5 色までのカラーで構成されます。各カラー・ボタンは、カラー・セットのバックグラウンド・カラーを表示します。各カラー・セットにある 5 つのカラーは、次のディスプレイ・コンポーネント・リソースを表します。

`foreground` — アプリケーション・ウィンドウまたはウィンドウ枠のフォアグラウンドです。通常は黒か白です。一般的にウィンドウやタイトル内のテキストに使用します。

`background` — アプリケーション・ウィンドウまたはウィンドウ枠のバックグラウンドです。

`topShadowColor` — アプリケーション・コントロール (プッシュ・ボタンなど) とウィンドウ枠の上部と左部の射影のカラーです。

`bottomShadowColor` — アプリケーション・コントロールとウィンドウ枠の下部と右部の射影のカラーです。

selectColor — アクティブなトグルやボタンなど、特定コントロールのアクティブな状態を示すカラーです。

各パレットが使用するカラー・セットの数は、colorUse リソースによって決まりますが、スタイル・マネージャの [使用する色の数] ダイアログ・ボックスを使用して設定できます。

カラー値の指定

スタイル・マネージャは、そのパレット・ファイルにカラー情報を書き込むときに RGB 値を使用します。RGB 数の構文は次のとおりです。

#*RedGreenBlue*

Red、*Green*、*Blue* はそれぞれ 1 桁から 4 桁 の 16 進数で、使用するカラーの量を示します。カラーのそれぞれの桁数は同じでなければなりません。したがって、有効カラー値は、3、6、9、12 桁のいずれかの 16 進数から成ります。

たとえば、白は次のどの方法でも指定できます。

```
#fff
#ffffff
#fffffffff
#ffffffffffff
```

カラー・リソースを直接設定する場合、カラー名か RGB 値のどちらかを使用できます。ファイル /usr/lib/X11/rgb.txt には、名前が付いたカラーのすべてがリストされています。

カラー・セットをリソースにマップする方法

デスクトップはリソースを介してさまざまなディスプレイ要素にカラー・セットをマップし、表 17–5 のような割り当てを行います。

表 17–5 リソースにマップされているカラー・セット

リソース	ディスプレイ要素
activeColorSetId	アクティブなウィンドウ枠のカラー
inactiveColorSetId	非アクティブなウィンドウ枠のカラー

表 17–5　リソースにマップされているカラー・セット　続く

リソース	ディスプレイ要素
textColorSetId	テキスト・エントリ領域
primaryColorSetId	アプリケーションのメイン・バックグラウンド領域
secondaryColorSetId	アプリケーションのメニュー・バー、メニュー、およびダイアログ・ボックス

これらのリソースは、カラー・セット ID を値として取ります。カラー・セット ID でディスプレイ要素に配色を指定すると、新しいパレットがスタイル・マネージャで選択されるときに要素を新しい配色へと動的に変更できます。

個々のアプリケーションに対して、これらのリソースを使用できます。たとえば次の行は、原色に 8 カラー・セットを使用することにより、すべての dtterm ウィンドウを視覚的にグループ化する方法を示します。

```
dtterm*primaryColorSetId:   8
```

デフォルトのカラー・セット割り当て

ディスプレイ要素に使用されるカラー・セット ID は、スタイル・マネージャのカラーの数の設定に対応します。

表 17–6 に、スタイル・マネージャで [デスクトップにもっと多くの色数を割り当てる] を設定した場合の、High Color (8 カラー・セット) 用のカラー・セット ID を示します。

表 17–6　High Color

カラー・セット **ID**	ディスプレイ要素
1	アクティブなウィンドウ枠のカラー
2	非アクティブなウィンドウ枠のカラー
3	未使用 (デフォルト時)
4	テキスト・エントリ領域
5	アプリケーションのメイン・バックグラウンド領域

表 17–6 High Color 続く

カラー・セット ID	ディスプレイ要素
6	アプリケーションのメニュー・バー、メニュー、およびダイアログ・ボックス
7	デフォルト時には未使用
8	フロント・パネルのバックグラウンド

表 17–7 に、スタイル・マネージャで [アプリケーションにもっと多くの色数を割り当てる] に設定した場合の、Medium Color (4 カラー・セット) 用のカラー・セット ID を示します。

表 17–7 Medium Color

カラー・セット ID	ディスプレイ要素
1	アクティブなウィンドウ枠のカラー
2	非アクティブなウィンドウ枠のカラー
3	アプリケーションとフロント・パネルのバックグラウンドのカラー
4	テキスト・エントリ領域

表 17–8 に、スタイル・マネージャで [アプリケーションに最大限度の色数を割り当てる] に設定した場合の、Low Color (2 カラー・セット) 用のカラー・セット ID を示します。

表 17–8 Low Color

カラー・セット ID	ディスプレイ要素
1	アクティブなウィンドウ枠、ワークスペース選択ボタン
2	すべてのその他のディスプレイ要素

スタイル・マネージャによるカラーのコントロール

スタイル・マネージャを介して、デスクトップ・アプリケーションやその他の連携アプリケーションのカラーを動的に変更できます。スタイル・マネージャが設定するフォアグラウンド・カラーとバックグラウンド・カラーは、非連携アプリケーションに対して使用可能です。

スタイル・マネージャのカラー変更に反応するクライアントの場合、クライアントは必ずデスクトップの Motif ライブラリを使用してください。他のツールキットで書かれたクライアントは、スタイル・マネージャの変更に応じて動的にカラーを変更できません。これらのクライアントに対するカラーの変更は、クライアントが再起動されるまで有効ではありません。

クライアントに適用される特定のカラー・リソースは他にありません。これには、ユーザ指定のリソース、app-defaults、およびアプリケーションに組み込まれるリソースが含まれます。

クライアントは primaryColorSetId リソースと secondaryColorSetId リソースを指定して、デスクトップ・パレット内の特定のカラーを使用できます。

スタイル・マネージャが使用するカラーの数

スタイル・マネージャが使用するカラーの数は、次のリソースの値に依存します。

colorUse — デスクトップが使用するカラーの数を設定します。

shadowPixmaps — 2 つのシャドウ・カラーをピックスマップと置き換えるようにデスクトップに指示します。

foregroundColor — フォアグラウンド・カラーを動的に変更するか指定します。

dynamicColor — パレットを切り替えるときにアプリケーションがカラーを変更するかをコントロールします。

表 17–9 は、デスクトップが割り当てるカラーの最大数をリストします。

表 17–9 デスクトップ・カラーの数

ディスプレイ	カラーの最大数	内訳
B_W	2	黒と白
LOW_COLOR	12	2 カラー・セット x 5 カラー + (黒と白)

表 17–9　デスクトップ・カラーの数　続く

ディスプレイ	カラーの最大数	内訳
MEDIUM_COLOR	22	4 カラー・セット x 5 カラー + (黒と白)
HIGH_COLOR	42	8 カラー・セット x 5 カラー + (黒と白)

カラーの最大数を決定するには、次のようにします。

1. パレット内のカラー・セットの数と各カラー・セット内のカラーの数を掛けます。

2. 2 (黒と白の分) を加えます。

しかし、次の設定により、パレット内に 10 色のカラー (つまり、4 カラー・セット x 各セット (background と selectColor) 内の 2 色のカラー ＋ (黒と白)) だけを持つことになります。

```
*colorUse: MEDIUM_COLOR
*shadowPixmaps:  True
*foregroundColor: White
```

注 - マルチカラー・アイコンは、さらに 14 色のカラーを使用します。

colorUse リソース

colorUse リソースのデフォルトの値は、MEDIUM_COLOR です。このリソースの値は、パレットで使用するカラーの数に影響を与えます。他のリソースは、シャドウを作成するのに使用するカラーの数に影響を与えます。colorUse リソースの値は、マルチカラー・アイコンの使用にも影響を与えます。

値	説明
B_W	[白黒] スタイル・マネージャ設定。1 から 3 までのカラー区画で表示 カラー・セットの数: 2 カラーの最大数: 2 デフォルトのカラー数: 2 マルチカラー・アイコンなし
LOW_COLOR	[アプリケーションに最大限度の色数を割り当てる] スタイル・マネージャ設定。4 から 5 までのカラー区画で表示 カラー・セットの数: 2 カラーの最大数: 12 デフォルトのカラー数: 12 マルチカラー・アイコンなし
MEDIUM_COLOR	[アプリケーションにもっと多くの色数を割り当てる] スタイル・マネージャ設定。6 のカラー区画で表示 カラー・セットの数: 4 カラーの最大数: 22 デフォルトのカラー数: 22 マルチカラー・アイコンあり
HIGH_COLOR	[デスクトップにもっと多くの色数を割り当てる] スタイル・マネージャ設定。7 以上のカラー区画で表示 カラー・セットの数: 8 カラーの最大数: 42 デフォルトのカラー数: 42 マルチカラー・アイコンあり
デフォルト	デスクトップは、そのディスプレイに対して正確な値を選択します。(色数が多いディスプレイでデスクトップが使用するカラーの数を減らすには、デフォルトの `colorUse` リソースに `MEDIUM_COLOR` を設定します。)

shadowPixmaps リソース

`shadowPixmaps` リソースは、デスクトップに 2 つのシャドウ・カラーをピックスマップに置き換えるよう指示します。これらのピックスマップは、黒または白とバックグラウンド・カラーを混ぜ合わせて、上部か下部のシャドウ値のシミュレー

ションを行います。この方法は、カラー・セルをシャドウ・カラーに割り当てる必要がないため、必要なカラーの数から 2 が引かれます。

値	説明
True	デスクトップは topShadowPixmap と bottomShadowPixmap を作成して、シャドウ・カラーの代わりに使用します。
False	パレットの topShadowColor と bottomShadowColor を使用します。

shadowPixmaps のデフォルト値は、持っている colorUse リソースとディスプレイをサポートするハードウェアに依存します。

foregroundColor リソース

foregroundColor リソースは、パレットでフォアグラウンドを設定する方法を指定します。

設定	結果
White	フォアグラウンドに白を設定します。
Black	フォアグラウンドに黒を設定します。
Dynamic	(デフォルト) フォアグラウンドに、background の値に応じて、黒か白を動的に設定します。たとえば、黄色のバックグラウンドで白い文字を使用すると読みにくいので、システムは黒を選択します。

foregroundColor に Black か White を設定する場合、カラー・セット内のカラーの数が 1 減りますが、フォアグラウンドがバックグラウンド・カラーの変更に応じて変更されることはありません。

colorUse の値が B_W の場合以外、foregroundColor のデフォルト値は Dynamic です。

dynamicColor リソース

dynamicColor リソースは、アプリケーションがカラーを動的に変更するかどうか、つまりパレットを切り替えるときにクライアントがカラーを変更するかどうかをコントロールします。

値	説明
True	クライアントは新しいパレットが選択されるとカラーを動的に変更します。これがデフォルト値です。
False	クライアントはカラーを動的に変更しません。新しいパレットが選択されると、クライアントはセッションの再起動時に新しいカラーを使用します。

dynamicColor リソースの値が True の場合、同じカラーに見えたとしても、カラーを動的に変更できるクライアントではなく、カラーを動的に変更できないクライアント (非 Motif アプリケーション) が、カラー・マップに異なるセルを割り当てます。

注 - すべてのクライアントは同じカラー・セルを共有するので、dynamicColor に False を設定すると、デスクトップで使用したカラーの数だけ減少します。

アプリケーション・ウィンドウのシャドウの濃さの設定

デスクトップは、ボタンのシャドウやフォーカス強調表示などのように、アプリケーション・ウィンドウにあるコンポーネントの 1 ピクセルのデフォルトのシャドウの濃さを定義します。Motif 1.2 アプリケーションは、このリソース値を使用します。他のアプリケーションはこのリソース値を取得しないので、ディスプレイには異なる状態で表示されます。

シャドウの濃さを非 Motif 1.2 アプリケーションの 1 ピクセルに設定するには、次のようにします。

1. **root** でログインします。
2. /etc/dt/config/***language***/sys.resources ファイルを作成します。
3. /etc/dt/config/***language***/sys.resources にアプリケーションに固有のリソースを次のように指定します。

```
application_class_name*XmCascadeButton*shadowThickness: 1
```

システム・デフォルト・リソースを無効にする方法、およびすべてのデスクトップ・ユーザに追加のリソースを指定する方法の詳細は、30ページの「セッション・リソースの読み込み」を参照してください。

第 18 章

ローカライズされたデスクトップ・セッションの構成

ローカライズされたデスクトップ・セッションを構成するには、次の作業が必要です。

- LANG 環境変数とその他の NLS (National Language Support) 環境変数を設定する
- 言語に依存するメッセージ・カタログとリソース・ファイルにアクセスする
- 国際化対応システムを介してアプリケーションをリモートで実行する

この章では、次の内容について説明します。

LANG 環境変数の管理

オペレーティング・システムの言語依存ルーチンを使用するには、LANG 環境変数がデスクトップに設定されなければなりません。デスクトップは、次の言語をサポートします。

- 西欧、ラテン系言語
- 日本語
- 繁体字 (中国語)
- 簡体字 (中国語)
- 韓国語

注 - デスクトップ・ベンダによって、その他の言語のサポートが追加されている場合もあります。

LANG は、オペレーティング・システムでサポートされているどの値にも設定できます。ログイン画面の [オプション] メニューに、サポートされている言語と地域のリストが表示されます。

デスクトップに LANG を設定するには、次の 4 つの方法があります。

- Xconfig ファイルのリソースを編集する
- ログイン画面の [オプション] メニューを使用する
- 実行可能な sh または ksh Xsession.d スクリプトを作成する (Xsession.d スクリプトの使用方法については、26ページの「Xsession.d スクリプトの参照」を参照してください。)
- .dtprofile ファイルを編集する

LANG が設定されていると、デスクトップはローカライズされたインタフェースを決定するために、次の言語依存ファイルを使用します。

カラー — /usr/dt/palettes/desc.*language*

背景 — /usr/dt/backdrops/desc.*language*

複数のユーザの言語を設定する

Xconfig ファイルを使用して言語を設定する場合、ログイン画面がローカライズされ、すべてのユーザに対して LANG が設定されます。これは、マルチディスプレイ・システムで、すべてのディスプレイの LANG を変更する唯一の方法です。(Xconfig を変更するには、/usr/dt/config/Xconfig を /etc/dt/config/Xconfig にコピーします。)

言語は、次の行を /etc/dt/config/Xconfig に配置することで設定されます。

```
dtlogin.host_display.language: language
```

たとえば、次の行はディスプレイ my_host:0 の LANG を Swedish_locale に設定します。

```
dtlogin.my_host_0.language: Swedish_locale
```

dtlogin クライアントは、その言語の適切なメッセージ・カタログを読み込み、ローカライズされたログイン画面に表示します。次に dtlogin クライアントは、/etc/dt/config/Xresources リソース・ファイルの以下のリソースを使用して、ロケールのリストを判定します。

- dtlogin*language
- dtlogin*languageList
- dtlogin*languageName

Xconfig ファイルは、選択した言語のために NLSPATH 環境変数を適切に設定する必要がある場合があります。その必要がない場合、または NLSPATH 環境変数を自分で設定したい場合は、321ページの「NLSPATH 環境変数」を参照してください。

1 つのセッションに言語を設定する

1 つのセッションに言語を設定するには、ログイン画面の [オプション] メニューを使用します。ログイン画面はローカライズされ、LANG はユーザ用に設定されます。LANG はセッションの完了時に (dtlogin に設定されている) デフォルト値に戻ります。

1 人のユーザの言語を設定する

ログインの LANG 設定を *HomeDirectory*/.dtprofile ファイルの中で無効にできます。ログイン画面はローカライズされず、LANG はユーザ用に設定されます。

- sh または ksh を使用する場合

```
LANG=language
export LANG
```

- csh を使用する場合

```
setenv LANG language
```

LANG 環境変数とセッション構成

LANG 環境変数は、セッション構成ファイルを検索する際に使用するディレクトリ名を変更します。

ローカライズされたセッション構成ファイルは次のとおりです。

- /usr/dt/config/*language*/Xresources (ログイン・マネージャのリソース・ファイル)
- /usr/dt/config/*language*/sys.font (セッション・マネージャのリソース・ファイル)
- /usr/dt/config/*language*/sys.resources (セッション・マネージャのリソース・ファイル)
- /usr/dt/config/*language*/sys.session (セッション・マネージャ実行可能シェル)
- /usr/dt/config/*language*/sys.dtwmrc (ウィンドウ・マネージャのリソース・ファイル)
- /usr/dt/appconfig/types/*language*/dtwm.fp (ウィンドウ・マネージャ・フロントパネル)

その他の NLS 環境変数の設定

LANG の他に、LC_CTYPE や LC_ALL などの NLS 環境変数があります。これらの変数は、dtlogin 言語リソースや、ログイン画面の [オプション] メニューの影響を受けません。これらの変数は、次のファイルに設定しなければなりません。

- システム共通変数: /etc/dt/config/Xsession.d
- 個人用変数: *HomeDirectory*/.dtprofile

NLSPATH 環境変数

NLSPATH 環境変数は、アプリケーションがメッセージ・カタログの検索に使用するディレクトリ・パスを決定します。LANG と NLSPATH の両方でこれらのメッセージ・カタログを使用するように設定しなければなりません。ローカライズされたメッセージの位置については、324ページの「メッセージ・カタログのローカライズ」を参照してください。ほとんどのデスクトップ・クライアントは、起動時にパスを NLSPATH の先頭に付けます。

フォントの検索

デスクトップに含まれるフォントは /usr/lib/X11/fonts ディレクトリにあります。各ディレクトリには、ディレクトリ・ファイル fonts.dir と別名ファイル fonts.alias があります。fonts.dir ファイルと fonts.alias ファイルの作成方法については、mkfontdir(1) のマニュアル・ページを参照してください。

サーバで使用できるすべてのフォントをリストするには、xlsfonts コマンドを使用します。サーバにフォントを追加または削除するには、xset コマンドを使用します。

ローカライズされた app-defaults リソース・ファイル

デスクトップ・クライアント用の app-defaults ファイルのデフォルト位置は /usr/dt/app-defaults/*language* です。たとえば、LANG が Swedish_locale に設定されている場合、アプリケーションは app-defaults ファイルを /usr/dt/app-defaults/Swedish_locale で検索します。LANG が設定されていない場合、*language* は無視され、アプリケーションは app-defaults ファイルを /usr/app-defaults/C で検索します。

app-defaults の位置を変更するには、XFILESEARCHPATH 環境変数を使用します。たとえば、app-defaults を /users に移動するには、XFILESEARCHPATH を /usr/app-defaults/*language*/*classname* に設定します。

XFILESEARCHPATH を *HomeDirectory*/.dtprofile に設定した場合、その値は実行するすべてのデスクトップおよび X クライアントに適用されます。非クライアント

は、XFILESEARCHPATH によって指定されるディレクトリにリンクするかコピーしない限り、リソース・ファイルを見つけることはできません。

アクションとデータ型のローカライズ

注 - /usr/dt/appconfig ディレクトリ内のファイルをカスタマイズする場合は、カスタマイズする前に、ファイルを /etc/dt/appconfig ディレクトリにコピーしてください。

アクションおよびデータ型定義ファイルの検索パスには、言語に依存するディレクトリが含まれます。

- 個人用: *HomeDirectory*/dt/types
- システム共通: /etc/dt/appconfig/types/*language*
- 組み込み: /usr/dt/appconfig/types/*language*

アプリケーション・マネージャの構成ファイルの検索パスは次のとおりです。

- 個人用: *HomeDirectory*/dt/appmanager
- システム共通: /etc/dt/appconfig/appmanager/*language*
- 組み込み: /usr/dt/appconfig/appmanager/*language*

このディレクトリのファイル名とディレクトリ名はローカライズされています。

アイコンとビットマップのローカライズ

アイコンをローカライズするには、アイコン・エディタでアイコンを編集し、次のディレクトリに保存します。

/etc/dt/appconfig/icons/*language*

アイコンを別のディレクトリに保存する場合は、アイコンを保存したディレクトリを XMICONSEARCHPATH 環境変数に指定します。XMICONBMSEARCHPATH 環境変数は、アイコンの検索に使用するパスを制御します。

背景名のローカライズ

背景のローカライズは、記述ファイル (desc.*language* と desc.backdrops) を使用します。背景ファイルには、特定のローカライズされたディレクトリ (/usr/dt/backdrops/*language* など) は存在しません。すべてのロケールは同じ背景ファイルのセットを使用しますが、翻訳された背景名を格納しているロケール独自の desc.*language* ファイルを持っています。

記述ファイルには、翻訳された背景名のリソースが指定されています。

```
Backdrops*Corduroy.desc:      Velours
Backdrops*DarkPaper.desc:      PapierKraft
Backdrops*Foreground.desc:     AvantPlan
```

desc.*language* ファイルは、スタイル・マネージャに背景を表示するために、ロケール *language* の背景の記述を取り出すのに使用します。記述の指定がある場合は、スタイル・マネージャの背景リストに表示されます。指定がない場合は、背景ファイル名を使用します。

独自の背景記述を *HomeDirectory*/.dt/backdrops/desc.backdrops ファイルに追加できます。このファイルは、ロケールに関係なく、ユーザによって追加されたすべての背景の背景記述を取り出すのに使用します。

description ファイルの検索パスは次のとおりです。

- 個人用: *HomeDirectory*/.dt/backdrops/desc.backdrops
- システム共通: /etc/dt/backdrops/desc.*language*
- 組み込み: /usr/dt/backdrops/desc.*language*

パレット名のローカライズ

パレットのローカライズは、記述ファイル (desc.*language* と desc.palettes) を使用します。特定のローカライズされたディレクトリ (/usr/dt/palettes/*language* など) は存在しません。すべてのロケールは同じパレット・ファイルのセットを使用しますが、翻訳されたパレット名を格納している独自の desc.palettes ファイルを持っています。

記述ファイルには、翻訳されたパレット名のリソースが指定されています。

```
Palettes*Cardamon.desc:     Cardamone
Palettes*Cinnamon.desc:     Cannelle
Palettes*Clove.desc:      Brun
```

desc.*language* ファイルは、スタイル・マネージャ・リストにパレットを表示するために、ロケール *language* のパレットの記述を取り出すのに使用します。記述の指定がある場合は、スタイル・マネージャのパレット・リストに表示します。指定がない場合は、パレット・ファイル名を使用します。

独自のパレット記述を *HomeDirectory*/.dt/palettes/desc.palettes ファイルに追加できます。このファイルは、ロケールに関係なく、ユーザによって追加されたすべてのパレットのパレット記述を取り出すのに使用します。

記述ファイルの検索パスは次のとおりです。

- 個人用: *HomeDirectory*/.dt/palettes/desc.palettes
- システム共通: /etc/dt/palettes/desc.*language*
- 組み込み: /usr/dt/palettes/desc.*language*

ヘルプ・ボリュームのローカライズ

ローカライズされたヘルプ・ボリュームがある場合は、次のいずれかのディレクトリに格納しなければなりません。システムは最初に見つけたヘルプ・ボリュームを使用します。ディレクトリは次の順番に検索されます。

- 個人用: *HomeDirectory*/.dt/help
- システム共通: /etc/dt/appconfig/help/*language*
- 組み込み: /usr/dt/appconfig/help/*language*

メッセージ・カタログのローカライズ

メッセージ・カタログをローカライズした場合は、次のディレクトリに格納します。

/usr/dt/lib/nls/msg/*language*

このディレクトリに *.cat ファイルを格納します。

ローカライズされたデスクトップ・アプリケーションのリモート実行

ローカライズされたデスクトップ・アプリケーションは、同じようにローカライズされたデスクトップ・インストールがある場合、どのリモート実行ホストでも起動できます。アプリケーションを起動しているホストの NLS 関連の環境変数の値は、アプリケーションの起動時にリモート・ホストに渡されます。しかし、環境変数にはホスト情報は含まれません。

キーボード・マップのリセット

予期しない文字や動作に遭遇した場合、または文字の表示や入力ができない場合は、キーボード・マップをリセットしてインストールするか、入力メソッドを変更する必要がある場合があります。

入力メソッドは LC_CTYPE、LANG、LC_ALL 環境変数か、–lang オプションで指定された言語によって決定されます。

たとえば、POSIX シェル内で C ロケールで端末を開く場合は、次のように指定します。

```
LANG=C   dtterm
```

この新しい端末は、C の入力メソッドとフォントを含む C ロケールを使用します。言語固有のキーボードを使用している場合は、入力メソッドは拡張文字の入力を受け付けないことがあります。言語固有のキーボードで C ロケールを使用する場合、端末を起動する前に、LC_CTYPE、LANG、LC_ALL のいずれかの環境変数を適切な値に設定する必要があります。

たとえば、ドイツ語のキーボードで C ロケールを使用するには、次のように入力します。

```
LANG=C LC_CTYPE=DeDE dtterm
```

X サーバがリセットされてキーマップが初期化されている場合は、xmodmap コマンドを使用して、サーバで適切なキーボード・マップをリセットできます。

付録 **A**

dtconfig(1) のマニュアルページ

```
NAME
     dtconfig - desktop configuration utility
SYNOPSIS
     dtconfig [-d |-e |-kill |-reset |-p |-inetd |-inetd.ow ]
DESCRIPTION
     Desktop configuration utility.  Integrates CDE with the
     operating system of the underlying platform.  System root
     login privilege is required to use dtconfig.
OPTIONS
     -d   Disables desktop auto-start feature.  At end of boot
          cycle, platform's native text based login mechanism
          will be used.
     -e   Enable's desktop auto-start feature.  Desktop login
     -kill
          Kill desktop (window based) login process and any user
          sessions associated with it.  Return control to
          system's native text based console.
     -reset
          Tell desktop (window based) login process to reread its
          configuration file to incorporate any changes.

     -p   Printer actions for any printer known to platform will
          be created if such print actions do not already exist
          in the platform's actions database.  This option is
          executed automatically at boot time if desktop auto-
          start has been enabled.
     -inetd
          Adds /usr/dt/bin daemons to the /etc/inetd.conf file.
          Specific CDE background daemon setup includes
          rpc.ttdbserverd (ToolTalk), rpc.cmsd (Calendar
          Manager), and dtspcd (subprocess control).  This -inetd
          option is called automatically by Solaris CDE package
          installs.  This -inetd option is also useful for CDE
          daemon setup outside of normal Solaris CDE install,
          including system setup where /usr/dt has simply been
          mounted from some remote fileserver exporting the
          /usr/dt directory.
     -inetd.ow
          Switches the ToolTalk and Calendar Manager daemons
          (rpc.ttdbserverd & rpc.cmsd) start lines in
```

```
          /etc/inetd.conf back to the older /usr/openwin/bin
          area.  This option is called automatically by Solaris
          CDE package remove scripts when needed.  It is also
          useful outside of normal Solaris package remove opera-
          tions when /usr/dt is about to be manually removed or
          unmounted.
RETURN VALUES
     0    Successful completion
    >0    Error condition
FILES
     /usr/dt/bin/dtconfig       location of dtconfig utility
SEE ALSO
     dtlogin (1), dtprintinfo (1)
```

索引

E

F

G

H

I

K

L

T

U

V

W

X

あ

さ

し

す

せ

た

ち

て

と

な

に

ね

へ

ほ

わ