Motif 対応 XRT/gear™

プログラマーズ・ガイド&リファレンス・マニュアル

UNIX Edition ■ Version 3.0



株式会社タンジェント・コンピューティング
〒150-0001 東京都渋谷区神宮前 6-6-11
tel:(03)3409-5364
fax: (03)3409-0147
info@tan.co.jp
http://www.tan.co.jp

1999 年 5 月

本書は著作権法により保護されています。(株)タンジェント・コンピューティングによる事前の許可なく、本書のいかなる部分も無断転載、複製、複写を禁じます。
Copyright©1999 (株)タンジェント・コンピューティング

XRT、XRT/3d、XRT/care、XRT/field、XRT/gear、XRT/graph、XRT/table、XRT PDSはKL Group Inc.の登録商標です。
その他製品名、名称、サービス等は該当する会社または組織の商標または登録商標です。

XRT製品ライセンスについて

この度はXRT製品をご購入いただきまして誠にありがとうございます。

本製品のご利用に際しましては、次ページに記載の「XRTバイナリ使用許諾契約書」をご一読くださるようお願いいたします。下記に掲げる事項は、この契約の理解を補助するもので、契約の一部ではありません。

- XRT 製品(XRT/graph、XRT/3d、XRT/table 等)のライセンスは、機種(SUN SPARC、IBM RS/6000、HP 9000 等)につき、またツールキット(Motif)につき許諾されます。
- XRT 製品のライセンスにはシングルとネットワークの二種類があります。
- シングル・ライセンスでは、一つの CPUID で識別されるマシンにおいてエンドユーザ・アプリケーションの開発に XRT 製品をご使用いただくことができます。XRT 製品を使用する開発には下記の内容が含まれます。
  - XRT 製品のライブラリをエンドユーザ・アプリケーションのバイナリにコンパイル、リンクする。
  - XRT 製品のビルダーを実行する。
  - XRT 製品を GUI ビルダーまたは UIMS ツールに統合する。XRT 製品が統合されているツールは XRT 製品が使用許諾されているマシンでのみ使用できます。
- ネットワーク・ライセンスでは、ネットワークに接続されている同ビル内の同機種のマシンにおいてエンドユーザ・アプリケーションの開発に XRT 製品をご使用いただくことができます。
- 静的にリンクされている XRT 製品のライブラリを含むエンドユーザ・アプリケーションのバイナリは、ロイヤリティーやランタイム費用なしで提供していただくことができます。

XRTバイナリ使用許諾契約書

重要: ソフトウェア・パッケージを開封する前、またはソフトウェアをインストール、使用する前に必ずお読みください。

XRT製品の開封、インストール、使用、またはいずれかの左記の行為により、以下の規定につきお客様が承認されたものとさせていただきます。本契約書はお客様(個人または法人)とKL Group Inc.(以下、「KL Group」という。)の間で締結される法的な契約書です。もし、以下の規定に御同意頂けない場合には、ダウンロードやインストールされたファイルを直ちに破棄し、開封前のパッケージをライセンス費用の御支払いを示す書類と一緒に御返却ください。お客様がお支払になった料金は返却されます。

ソフトウェア・ライセンス

ライセンスの許諾 この使用許諾契約(以下、「ライセンス」という。)は、御指定のCPUと一致する一台のコンピュータにおいて、お客様が御購入されたXRTソフトウェア(以下、「ソフトウェア」という。)を御使用いただく権利をお客様に許諾するものです。ネットワーク・ライセンスを御購入された場合は、御指定のCPUと一致するコンピュータとローカル・エリア・ネットワークで接続された、同機種及び同ビル内に設置されているコンピュータにおいて、御購入いただいたライセンスで許可される御用台数を上限として「ソフトウェア」を御使用いただくことができます。また、共有ライブラリを御購入された場合、「ソフトウェア」のユーザ数は、御購入されたライセンスで許可された数を超えることはできません。「ソフトウェア」は、テンポラリ・メモリ(RAM等)にロードされた時、パーマネント・メモリ(ハード・ディスク、CD-ROM、その他の記憶装置等)にインストールされた時、エンドユーザ・アプリケーションにリンクされた時、開発ツール(コンパイラ、リンカー、GUIビルダー等)によってアクセスされた時に「使用」されたとします。他のコンピュータへ「ソフトウェア」のファイルを配布する目的のみで、ネットワークのファイル・サーバにインストールされている場合は、「使用」されているとはみなされません。

アップデート 本許諾契約書は、KL Groupから提供される「ソフトウェア」の全てのアップグレード版に適用されます。

著作権 「ソフトウェア」とそれに関連するイメージ、写真、テキスト等は、KL Groupとその供給者が所有するもので、その著作権は、カナダと米国の著作権法及び国際条約の規定で保護されています。従って、「ソフトウェア」は、書籍や音楽録音等、著作権で保護される他の製品と同じように扱ってください。「ソフトウェア」は、バックアップ目的で複製する以外は、一切複製作業を行うことはできません。また、「ソフトウェア」に付属する関連資料も同様に扱ってください。

その他の制限 「ソフトウェア」は、貸与、リース、譲渡、リバース・エンジニア、逆コンパイル、逆アセンブル、パスワードの作成をすることはできません。但し、これらの制限が司法管轄区域によって許可されていない場合は、この限りではありません。

「ソフトウェア」で構築されたアプリケーションの配布 「ソフトウェア」の"lib"または"lib-shared"ディレクトリにある、オブジェクト・コードのアプリケーション・ファイルにリンク可能な形態のファイル(以下、「XRTライブラリ・ファイル」という。)は、次の条件に従う場合にのみ、ロイヤリティなしで複製、配布することができます。(1)再リンクが不可能な方法でソフトウェアにリンクされて「XRTライブラリ・ファイル」が配布される。(2)配布されるソフトウェアは、「ソフトウェア」と同等または類似した機能を持つ開発ツールではない。(3)配布するソフトウェアの販売活動に、許可なくKL Groupの社名、ロゴ、トレード・マーク等を使用しない。(4)配布するソフトウェアには、著作権に関する注意書きを明記する。(5)配布したお客様のソフトウェアに対して第三者からなされた損害賠償請求に基づく損害等については、KL Group及びその供給者は一切その責任を負わない。それ以外の場合は、「ソフトウェア」のヘッダー・ファイルや(APIのドキュメンテーションを含む)ドキュメンテーション類を配布することはいかなる方法であってもできません。

U.S. Government Restricted Rights 「ソフトウェア」とその関連資料は、RESTRICTED RIGHTSに基づいて提供されています。本契約書で許諾される「ソフトウェア」とその関連資料は、Federal Acquisition Regulations (FAR) Part 12と Defense FAR Supplement (DFARS) 252.227-7014 (JUN 1995)で定義される "Commercial computer software"です。FAR 12.212 とDFARS 252.227.7014に基づき、この「ソフトウェア」の使用、複写、開示は、本契約書の条項に従います。製造者: KL Group Inc., 260 King Street East, Toronto, Ontario, Canada, M5A 1K3.

保証規定

品質保証 KL Groupは、「ソフトウェア」購入日より60日間、「ソフトウェア」がその関連資料記載の仕様に従って実質的に動作することを保証します。但し、品質保証の期間の制限が司法管轄区域によって許可されていない場合は、適用法例の範囲内でのみ有効です。尚、本品質保証は、第三者に譲渡することはできません。KL Groupによる故意、重大な過失により生じた損害、または製品に対する責任は、KL Groupは除外いたしません。

補償 KL Groupがその判断に基づいて行う補償は、次の事項に限定されます。(1)御支払いになられた金額を払い戻す。(2)上記記載の「品質保証」を満たさない「ソフトウェア」に対して補修、交換する。但し、(1)、(2)いずれの対応も、「ソフトウェア」と共ににその購入を証する書類を返却された場合にのみ提供されます。また、かかる事態が事故、故意若しくは過失、誤用等、KL Groupの責に帰さない理由により生じた場合は、本品質保証は無効になります。尚、交換された「ソフトウェア」の保証期間は、その受領より30日か、または交換の対象になった「ソフトウェア」の保証期間の残りか、どちらか長い方が適用されます。

権利侵害に対する補償 「ソフトウェア」が、第三者の特許、業務上の秘密、著作権等の侵害の対象となった場合、またはKL Groupがそのように判断した場合は、KL Groupは、次の補償を行うことがあります。(1)「ソフトウェア」を継続して使用できる権利を取得する。(2)「ソフトウェア」を権利侵害の対象にならないように交換、修復する。但し、交換または修復された場合も、以前と同じ機能は保つようにする。上記(1)及び(2)が適正でない場合は、(3)「ソフトウェア」のライセンス許諾を中止し、削除する。その場合、ライセンスに対して支払われた金額の内、定額法を適用した5年間の償却期間に基づいた相当金額を返却する。

保証の制限 司法管轄区域の適用法令の範囲内で、KL Group及びその供給者は、「ソフトウェア」とその関連資料の商品性、特定目的適合性の保証を含め、本保証規定に規定されていないその他の保証を、明示たると黙示たるとを問わず一切いたしません。この有限保証は、お客様に特定な法的権利を与えるものですが、州、管轄区域によっては、その他の権利が与えられることもあります。。

免責 司法管轄区域の適用法令の範囲内で、KL Group及びその供給者は、「ソフトウェア」の使用または使用不能から生ずる損害(事業利益の損失、事業の中断、事業情報の損失または金銭的損害を含むがこれらに限定されない)に関して、一切責任を負わないものとします。例え、KL Groupがかかる損害の可能性について知らされていた場合も同様です。但し、州や管轄区域によっては、必然的または偶発的な損害に対する責任の除外や制限を認めていない場合があり、その場合、前述の制限は適応されないことがあります。

適用法令 お客様が"European Union"に属する地域に居住されている場合、本契約書はその国の法律に準拠し、契約当事者双方は、その国の司法に従います。それ以外の場合は、本契約書は、カナダ・オンタリオ州の法律に準拠し、契約当事者双方は、オンタリオ集の司法に従います。

パブリック・ドメイン・ソフトウェア 本契約書に記載されている条項は、パブリック・ドメインと明記されているソフトウェアには適用されません。

本契約書は、KL Groupより提供される"XRT FOR MOTIF BINARY LICENSE AGREEMENT"を(株)タンジェント・コンピューティングが日本語訳したものです。表現等に不一致がある場合は、"XRT FOR MOTIF BINARY LICENSE AGREEMENT"次ページ参照に従います。

XRT FOR MOTIF BINARY LICENSE AGREEMENT

Important — Read Carefully Before Opening Software Package(s), Installing or Using Software On Your Computer

By opening a sealed software package, or installing, or using this XRT product, you indicate your acceptance of the following XRT for Motif Binary License Agreement. This is a legal agreement between you (either an individual or an entity) and KL Group Inc. If you do not accept and agree with all of the terms of this agreement, promptly destroy any downloaded or installed files, and return the unopened software package(s) and all other materials with proof of payment to your place of purchase, and your license fee will be refunded.

SOFTWARE LICENSE

Please Note: This is a license agreement and not an agreement for sale.

Grant of License. This License Agreement ("License") permits you to use one copy of the software you purchased (the "SOFTWARE"), on a single computer after registration of that computer's CPU identification with us. If you have purchased a network site license you may also use the SOFTWARE on computers in the same building having the same architecture as a registered computer, and connected by the same local area network, provided that the total number of such computers, on which the SOFTWARE is ever used, does not exceed the maximum number permitted by the network site license which you ordered.

If you have purchased shared library versions of the SOFTWARE, the number of users of the SOFTWARE cannot exceed the maximum number permitted by the license which you ordered.

The SOFTWARE is "in use" on a computer when it is loaded into temporary memory (i.e. RAM); when it is installed into permanent memory (e.g. hard disk, CD-ROM, or other storage device); when it is being linked into an end-user application; or when it is accessed from a software development tool such as a compiler, a linker, or a GUI builder. The SOFTWARE is not considered "in use" when installed on a network file server for the sole purpose of file distribution to other computers.

Updates. This agreement will govern the terms of any SOFTWARE updates provided to you by KL Group.

Copyright. The SOFTWARE (including any images, photographs and text incorporated into the SOFTWARE) is owned by KL Group or its suppliers and is protected by Canadian and United States copyright laws and international treaty provisions. Therefore you must treat the SOFTWARE like any other copyrighted material (e.g., a book or musical recording) and not reproduce it except that you may make a backup for archival purposes. You may not copy the printed materials accompanying the SOFTWARE.

Other Restrictions. You may not rent, lease, transfer, reverse engineer, decompile, disassemble, create passwords for or translate the Software, except to the extent such foregoing restriction is expressly prohibited by applicable law.

Distribution of applications you build with the SOFTWARE. You have a royalty-free right to reproduce and distribute files in the SOFTWARE that are found in the "lib" or "lib-shared" directories and which are in a form suitable for linking into object code application files (the "XRT Library Files") provided that (a) you distribute the XRT Library Files only when linked into your software product in such a way as to prohibit further relinking; (b) your software product is not an application development tool that provides similar functionality to the SOFTWARE licensed hereunder; (c) you do not use KL Group's name, logo or trademark to market your software product; (d) you include a valid copyright notice on your software product; and (e) you agree to indemnify, hold harmless, and defend KL Group and its suppliers from and against any claims or lawsuits, including attorney's fees, that arise or result from the use or distribution of your software product. Unless otherwise specified, you may not in any manner distribute any header files or documentation (including API documentation) provided as part of the SOFTWARE.

U.S. Government Restricted Rights. The SOFTWARE and documentation are provided with RESTRICTED RIGHTS. The SOFTWARE (including documentation) licensed in this agreement is "Commercial computer software" as defined in Federal Acquisition Regulations (FAR) Part 12 and Defense FAR Supplement (DFARS) 252.227-7014 (JUN 1995). In accordance with FAR 12.212 and DFARS 252.227.7014, notwithstanding any provision to the contrary, use, duplication and disclosure of this software is governed by the terms of this agreement. Manufacturer is KL Group Inc., 260 King Street East, Toronto, Ontario, Canada, M5A 1K3.

LIMITED WARRANTY

Limited Warranty. KL Group warrants that the SOFTWARE will perform substantially in accordance with the accompanying written materials for a period of sixty (60) days from the date of receipt. Some states/jurisdictions do not allow limitations on duration of an implied warranty, so the above limitation may not apply to you. This warranty may not be assigned. KL Group does not exclude product liability or liability for damages resulting out of intent or gross negligence on the part of KL Group or its leading personnel.

Customer Remedies. KL Group's and its suppliers' entire liability and your exclusive remedy shall be, at KL Group's option, either (a) return of the price paid, or (b) repair or replacement of SOFTWARE that does not meet KL Group's Limited Warranty and which is returned to KL Group with a copy of your receipt. This Limited Warranty is void if failure of the SOFTWARE has resulted from accident, abuse, or misapplication. Any replacement SOFTWARE will be warranted for the remainder of the original warranty period or thirty (30) days, whichever is longer.

Customer Remedies for Infringement. Should the SOFTWARE become or in KL Group's opinion be likely to become the subject of a claim for infringement of patent, trade secret or copyright, KL Group may (a) procure for you the right to continue to use the SOFTWARE; or (b) replace or modify the SOFTWARE to make it non-infringing, provided that the same function is performed by the replacement or modified SOFTWARE; or if (a) or (b) are not commercially reasonable, (c) terminate the license to use the SOFTWARE, remove the SOFTWARE, and grant you a credit for the license fee as depreciated on a straight-line five (5) year basis.

No Other Warranties or Obligations.. To the maximum extent permitted by applicable law, KL Group and its suppliers shall have no other obligations and disclaim all other warranties, either express or implied, including, but not limited to, implied warranties of merchantability and fitness for a particular purpose, with regard to the SOFTWARE and the accompanying printed materials. This limited warranty gives you specific legal rights. You may have others which vary from state/jurisdiction to state/jurisdiction.

No Liability for Consequential Damages. To the maximum extent permitted by applicable laws, in no event shall KL Group or its suppliers be liable for any damages whatsoever (including without limitation, damages for loss of business profits, business interruption, loss of information, or other pecuniary loss) arising out of the use of or inability to use this KL Group product, even if KL Group has been advised of the possibility of such damages. Because some states/jurisdictions do not allow the exclusion or limitation of liability for consequential or incidental damages, the above limitation may not apply to you.

Governing Law. If you are a resident of one of the member states of the European Union, this Agreement is governed by the laws of your country of residence, and each of the parties hereto irrevocably attorns to the jurisdiction of the courts of your country of residence. Otherwise, this Agreement is governed by the laws of the Province of Ontario, Canada. Each of the parties hereto irrevocably attorns to the jurisdiction of the courts of the Province of Ontario.

Public Domain Software. This license agreement and the warranties, remedies, limitations, conditions and restrictions contained herein do not apply to any software clearly marked as being in the public domain.

BINLIC-M-11/98

目次

はじめに 1

XRT/gear について 1
XRT/gear 3.0 の新しい機能 2
前提知識 2
表記上の規約 3
関連資料 3
テクニカル・サポート 3
お客様のご意見 4

Part I XRT/gear の使用法

1 XRT/gear について 7

1.1 XRT/gear とは 7
1.2 XRT/gear と Motif クラス階層 8
1.3 環境変数 8
1.4 プログラムのコンパイルと許可 10
1.5 リソース・ファイルの使用 11
1.6 C++との使用 11
1.7 XRT/gear の国際化 12

2 Enhanced Label Widget と Pushbutton Widget 15

2.1 サンプル・プログラム 15
2.2 ウィジェットの概要 16
2.3 ボタンの型の指定 17
2.4 Pixmap の指定 17
2.5 テキストと pixmap の回転 18
2.6 Enhanced Label の枠 19
2.7 テキストと pixmap の位置 19
2.8 Enhanced Pushbutton - arm pixmap の指定 20
2.9 リソースの一覧 21
2.9.1 Enhanced Label のリソース 21
2.9.2 Enhanced Pushbutton のリソース 23
2.9.3 Enhanced Label によって継承されるリソース 26
2.9.4 Enhanced Pushbutton によって継承されるリソース 27

2.10 手続きと方法 28

3 Enhanced Toggle Widget 31

3.1 サンプル・プログラム 31
3.2 ウィジェットの概要 32
3.3 ボタンの状態 32
3.4 ボタンの色の設定 35
3.5 リソースの一覧 35
3.5.1 Enhanced Toggle によって継承されるリソース 39
3.6 手続きと方法 41

4 Tabbed Dialog 43

4.1 タブ・ダイアログとは 43
4.2 サンプル・プログラム 44
4.3 ウィジェットの概要 45
4.4 アクティブな tab の変更 46
4.5 page を Tab Button に関連させる 47
4.6 Tab 表示のコントロール 48
4.7 constraint リソース 52
4.8 Tab Button リソース 52
4.8.1 Button のタイプ 52
4.8.2 Tab のアンカー 53
4.8.3 pixmap の設定 53
4.8.4 Button の形 54
4.8.5 テキストと pixmap の配置 55
4.8.6 テキストと pixmap の回転 56
4.8.7 Tab Button と Page の影の色 57
4.9 角のサイズの指定 57
4.10 リサイズのコントロール 58
4.11 リソースの一覧 59
4.11.1 Tab Button のリソース 59
4.11.2 Tab Manager のリソース 61
4.11.3 Tab Manager の Constraint リソース 65
4.11.4 Tab Button によって継承されるリソース 66
4.11.5 Tab Manager によって継承されるリソース 68
4.12 手続きと方法 68
4.13 トランスレーションとアクション 71
4.13.1 デフォルトの Tab Button のトランスレーション 71
4.13.2 Tab Button のアクション 71

5 Toolbar Widget 73

5.1 サンプル・プログラム 73
5.2 ウィジェットの概要 74
5.3 方向の指定 74
5.4 マージンとスペース 75
5.5 Constraint リソース 75
5.6 ヘルプ・バルーンの追加 76
5.7 リソースの一覧 76
5.7.1 Toolbar の Constraint リソース 77
5.7.2 Toolbar によって継承されるリソース 78
5.8 手続きと方法 78

6 Aligner Widget 81

6.1 Aligner とは 81
6.2 サンプル・プログラム 82
6.3 ウィジェットの概要 82
6.4 マージンとスペース 83
6.5 Aligner と XmSeparator 83
6.6 ラベルの位置 84
6.7 Constraint リソース 85
6.8 リソースの一覧 87
6.8.1 Aligner の Constraint リソース 87
6.8.2 Aligner に継承されるリソース 89
6.9 手続きと方法 89

7 Outliner 91

7.1 Outliner について 91
7.1.1 データの階層と表示 92
7.1.2 その他のユーザ・インタラクション 92
7.1.3 ツリーの作成 93
7.1.4 ここから先に進むには 94
7.2 Outliner の基本概念 95
7.2.1 Outliner ウィジェットとツリー 95
7.2.2 XrtString オブジェクト 96
7.2.3 XrtList オブジェクト 96
7.3 ウィジェットと Xt Object の概要 97
7.4 ツリーの作成 98
7.4.1 データ・ファイルのフォーマット 98
7.4.2 データ・ファイルからツリーを作成する 99
7.4.3 文字ストリングからツリーを作成 101

7.4.4 ツリーを直接作成する 101
7.4.5 ツリーの保管 102
7.5 ツリーのトラバース 103
7.5.1 ノードの子のリスト 104
7.5.2 カレント・ノード 104
7.5.3 トラバースの方法 105
7.6 ツリーの再編成 107
7.6.1 ノードの移動 107
7.6.2 ノードの削除 108
7.6.3 ノードのソート 108
7.7 ツリー表示のコントロール 110
7.7.1 高さと幅 110
7.7.2 マージン 112
7.7.3 色とフォント 113
7.7.4 スペース 113
7.7.5 表示・非表示(visibility) 114
7.7.6 テキストの切り取り 115
7.7.7 ノードの高さ 115
7.7.8 フォーカスのハイライト 115
7.7.9 リサイズ 116
7.7.10 カーソルの表示 117
7.8 複数の列 118
7.8.1 複数ラベルの定義 118
7.8.2 新しい列の作成 120
7.8.3 列の順序 121
7.8.4 選択された列 122
7.8.5 列ラベル 123
7.8.6 列のインデックス 123
7.8.7 列の表示のコントロール 124
7.9 ノードのカスタマイズ 129
7.9.1 ノード・ラベル 129
7.9.2 ノード・スタイル 129
7.9.3 アイコン 131
7.9.4 オブジェクトを結ぶ線とショートカット・ボタン 133
7.9.5 ユーザ定義のデータ 135
7.10 ユーザ・インタラクション 135
7.10.1 選択 136
7.10.2 フォーカスと Selected(選択されている状態)の違い 139
7.10.3 フォルダの状態の指定 139
7.10.4 選択されているノード・リストの取得 140
7.10.5 選択コールバック 140

7.10.6 スクロールバー....141
7.10.7 ドラッグ&ドロップ....141
7.10.8 activate コールバック....144
7.10.9 イベントの位置....144
7.10.10 列の再編成....145
7.11 Outliner ウィジェットの印刷....146
7.11.1 PostScript 出力の生成....146
7.11.2 PostScript フォントのカスタマイズ....147
7.11.3 print コールバック....148
7.12 パフォーマンス....149
7.12.1 ダブル･バッファリング....149
7.12.2 再描画のコントロール....150
7.12.3 列幅を固定する....150
7.12.4 Lightweight ノードを使用する....150
7.12.5 XmNxrtGearNodeCheckForDuplicates の使用....150
7.12.6 完全な階層を指定しない....150
7.13 リソースの一覧....151
7.13.1 Outliner のリソース....151
7.13.2 XrtNodeObject リソース....170
7.13.3 XrtNodeFolder オブジェクトのリソース....171
7.13.4 XrtNodeStyle オブジェクトのリソース....171
7.13.5 XrtColumn Object....173
7.13.6 Outliner によって継承されるリソース....177
7.13.7 Objects によって継承されるリソース....177
7.14 手続きと方法....178
7.14.1 Outliner の手続きと方法....178
7.14.2 XrtNode と XrtNodeFolder の手続きと方法....185
7.14.3 XrtNodeStyle の手続きと方法....196
7.14.4 XrtColun の手続きと方法....197
7.15 トランスレーションとアクション....198
7.15.1 Node 領域の Clip Child のトランスレーション....198
7.15.2 列ラベルの Clip Child のトランスレーション....208
7.15.3 XmText ウィジェットのトランスレーション....208
7.15.4 アクション....209
7.16 Lightweight フォルダとアイテム･ノード....212
7.16.1 ウィジェットとして cast される Lightweight Node....213
7.16.2 Lightweight ノードのリソースの取得と設定....213
7.16.3 Lightweight ノードの破棄....214
7.16.4 Lightweight フォルダのデモ....214
7.16.5 Lightweight リソースの一覧....214
7.16.6 手続きと方法....215

7.16.7 Lightweight フォルダ・ノードのリソースの一覧 218
7.16.8 手続きと方法 218

8 Progress ウィジェット 223

8.1 サンプル・プログラム 223
8.2 ウィジェットの概要 225
8.3 ラベルと範囲 225
8.4 タイムアウトと TimeOut コールバック 226
8.5 ラベルの表示 228
8.6 バーの表示 231
8.7 リソースの一覧 233
8.7.1 Progress ウィジェットによって継承されるリソース 238
8.8 手続きと方法 238

9 Combo Box ウィジェット 241

9.1 サンプル・プログラム 241
9.2 ウィジェットの概要 242
9.3 メニューのコントロール 243
9.3.1 pick list の指定 243
9.3.2 メニューのコールバック 245
9.4 表示の属性のコントロール 245
9.4.1 矢印(arrow) 245
9.4.2 メニュー・アイテムの配置 248
9.4.3 間隔 248
9.4.4 表示されるアイテムの数 248
9.5 ユーザ・インタラクション 248
9.6 自動マッチング 250
9.7 リソースの一覧 250
9.7.1 Combo Box によって継承されるリソース 253
9.8 手続きと方法 254
9.9 トランスレーションとアクション 255
9.9.1 トランスレーション 255
9.9.2 アクション 255

10 PanedWindow 259

10.1 サンプル・プログラム 259
10.2 ウィジェットの概要 261
10.3 ユーザ・インタラクション 261
10.4 リソースの一覧 262
10.4.1 PanedWindow のリソース 262
10.4.2 PanedWindow の constraint リソース 264

10.4.3 PanedWindow によって継承されるリソース....265
10.5 手続きと方法....265
10.6 トランスレーションとアクション....267
10.6.1 トランスレーション....267
10.6.2 アクション....267

11 Widget Tip....269

11.1 Widget Tip について....269
11.2 Widget Tip の使用....270
11.3 Widget Tip をウィジェットに追加....270
11.4 Widget List の取得....271
11.5 ヘルプ・バルーン....271
11.6 表示コールバック....272
11.7 表示のタイミング....273
11.8 widget tip を不可に設定....274
11.9 ヘルプ・テキストの表示と削除....275
11.10 トランスレーションの定義....276
11.11 リソースの一覧....276
11.12 手続きと方法....278

12 Widget Snapshot....283

12.1 Widget Snapshot について....283
12.2 Pixmap Snapshot....283
12.3 Pixmap の書き込みと印刷....284
12.4 XImage スナップショット....285
12.5 リソースの一覧....285
12.6 手続きと方法....286

13 Lists, Strings, Colors, Icons....289

13.1 XrtList オブジェクト....289
13.1.1 サンプル・コード....290
13.1.2 リストの作成と破棄....290
13.1.3 リスト・アイテムの追加と削除....290
13.1.4 リスト・アイテムの検索と取得....292
13.1.5 リストの比較ルーチン....295
13.1.6 リストのソート....296
13.1.7 ユーザが定義したデータ....297
13.2 XrtString オブジェクト....297
13.2.1 ストリングの作成と破棄....297
13.2.2 ストリングの型....298
13.2.3 ストリング値の操作....299

13.2.4 二つのストリングの比較 299
13.3 XrtColorEditor オブジェクト 300
13.3.1 サンプル・コード 300
13.4 アイコン 300
13.5 手続きと方法 301
13.5.1 XrtList オブジェクトの方法 301
13.5.2 XrtString オブジェクトの方法 310
13.5.3 XrtColorEditor の方法 313
13.5.4 Icon 操作の方法 319

Part II リファレンス

A Data 型 323

B リソースの型とクラス 339

B.1 Aligner 339
B.1.1 Aligner Constraint リソース 339
B.2 Combo Box 340
B.3 Enhanced Label 340
B.4 Enhanced Pushbutton 340
B.5 Enhanced Toggle 341
B.6 Outliner 341
B.6.1 列のリソース 344
B.6.2 ノードのリソース 344
B.6.3 ノード・フォルダのリソース 344
B.6.4 ノード・スタイルのリソース 345
B.7 Progress 346
B.8 Tab Button 346
B.9 Tab Manager 348
B.9.1 Tab Manager Constraint リソース 348
B.10 Toolbar 349
B.10.1 Toolbar Constraint リソース 349
B.11 Widget Snapshot 349
B.12 Widget Tips 349
B.13 Paned Window 350
B.13.1 Paned Window constraint リソース 350

C リソースファイルと GUI ビルダの使用 351

C.1 構文 351

D UIL 引数 353

D.1 Aligner のリソース 353

D.2 Combo Box のリソース 354
D.3 Enhanced Label のリソース 354
D.4 Enhanced Pushbutton のリソース 355
D.5 Enhanced Toggle のリソース 355
D.6 Outliner のリソース 355
D.6.1 XrtNode のリソース 357
D.6.2 XrtFolderNode のリソース 357
D.6.3 XrtColumn のリソース 357
D.6.4 XrtNodeStyle のリソース 359
D.7 Progress のリソース 359
D.8 Tab Button のリソース 360
D.9 Tab Manager のリソース 360
D.10 Toolbar のリソース 361
D.11 定数の定義 361
D.12 Paned Window のリソース 362

E サンプル･コード 363

E.1 例 363
E.2 デモ 364

F Windows 95 のシミュレーション 365

はじめに

XRT/gear について ■ XRT/gear3.0 の新しい機能
前提知識 ■ 表記上の規約
関連資料■ テクニカル・サポート
お客様のご意見

XRT/gear について

XRT/gear は、ボタン、スライダ、メニュー等のウィジェットから構成されている Motif に新たなウィジェットを追加します。XRT/gear は次のように構成されています。

- *Enhanced Pushbutton* と *Enhanced Label* は、Motif の PushButton と Label に機能を追加したウィジェットです。
- *Enhanced Toggle* は、Motif の ToggleButton に機能を追加したウィジェットです。
- *Tab Manager* と *Tab Button* は、ファイル・フォルダの形式にアプリケーションのウィジェットを記憶、表示します。
- *Toolbar* は、ボタンまたはウィジェットを表示し、Motif のメニュー・バーの代りに使用することができます。
- *Aligner* は、ラベルとウィジェットのペアを列に揃えて表示します。
- *Outliner* は、階層に従ってデータをツリー構造に表示します。
- *Progress* は、タスクの状態をエンドユーザに示します。
- *Combo Box* は、エンドユーザにリストからアイテムを選択させる時に使用できます。
- *PanedWindow* は、子ウィジェットを縦または横方向に並べる、composite ウィジェットです。

XRT/gear には、次のようなユーティリティが提供されています。

- *Widget Tips* は、ウィジェットにポップアップするヘルプ・テキストを表示します。
- *Widget Snapshot* は、Motif ウィジェットの内容を PostScript に出力したり、pixmap または XImage フォーマットに変換します。
- *Icon Library* は、パブリック・ドメインのアイコンを 3000 個以上集めました。これらのアイコンをカテゴリ別に見ることができるビューワも付いています。

■ *Color Editor* は、色の編集に利用できるオブジェクトです。

XRT/gear には、ウィジェットの表示や動作をコントロールするリソースが提供されています。Motif に XRT/gear のウィジェットが追加された点を除けば、XRT/gear を使用したプログラミングは、他の Motif のプログラミングと全く同様に行えます。

XRT/gear 3.0 の新しい機能

XRT/gear 3.0 の主な新しい機能とそれが解説されているマニュアルのページです。

- Outliner のノード・ラベルに編集機能が追加されました。(P91、7.0)
- アプリケーション内で色の編集を簡単に行うことができるオブジェクト、Color Editor が追加されました。(P300、13.3)
- Paned Window を利用して、子ウィジェットを縦方向や横方向に簡単に並べることができるようになりました。(P259、10.0)
- その他、このバージョンでは、新しいデモや例プログラムが追加され、既存の機能の改良やバグの修正等が行われています。尚、修正されたバグに関してはリリースノートを参照してください。

前提知識

このマニュアルは、C 言語及び Motif ツールキットに精通していることを前提として書かれています。Motif や XRT/gear アプリケーションをプログラミングするには、C 言語の"char *の配列"、"構造体のポインタ"等を理解している必要があります。またリソース値の取得や設定等の Motif の基本的な概念も理解していなければなりません。Motif に関する資料については 関連資料の項(P3)をご覧ください。

表記上の規約

`Typewriter Font`

- C コードやファイル内容の例
- XRT/gear や Motif リソース名、定数
- 端末で入力するコマンド

斜体

- パス、ファイル名、プログラム、手続き、パラメータ
- 新しい用語や重要な用語
- 図やテーブル
- このマニュアルで参照されている参考資料名(Motif Programming Manual 等)

関連資料

- *Motif Programming Manual* と *Motif Reference Manual*(*O'Reilly & Associates,Inc* 出版)
- *X Toolkit Intrinsics Reference Manual*、*Xlib Programming Manual*、*Xlib Reference Manual*(*O'Reilly & Associates,Inc.*出版) これらのマニュアルは XRT/gear や Motif の開発には直接必要ありませんが、Xlib の知識は X Window アプリケーションの開発に重要です。

テクニカル･サポート

初期の問題の多くは、インストールやセットアップに関する問題です。リリースノート&インストールガイドには、インストールやセットアップの際の問題の回避法がありますので参照してください。

XRT テクニカル･サポートへお問い合わせされる場合は、次の事項をご連絡ください。

- XRT/gear ライセンスのシリアル番号
- X と Motif のバージョン
- OS のバージョン

尚、テクニカル･サポートを御受けいただくには、XRT/care と呼ばれる、保守サポート･サービスを御購入されていることが前提となります。XRT/care のお申し込み法やサービスに関する御質問等がございましたら、(株)タンジェント･コンピューティングまでご連絡ください。

オンライン･サポート

(株)タンジェント･コンピューティング及び、開発元の KL Group Inc 社の web サイトには、技術的な情報が掲載されていますので参照してください。

技術情報及び FAQ **(日本語)**	http://www.tan.co.jp/xrt/support/
技術情報(英語)	http://www.klgroup.com/cs/tech/xrt/
Knowledge Base(**英語**)	http://www.klgroup.com/cgi-bin/xrt_search.cgi
FAQ(**英語**)	http://www.klgroup.com/cs/tech/xrt/faq/

お客様のご意見

XRT 製品のアップグレードで追加される新しい機能の大半は、お客様のリクエストに基づいています。XRT/gear に関するお客様のご意見、ご使用例、或は、あれば便利と思われる機能等がございましたら下記までご連絡ください。

株式会社タンジェント･コンピューティング
〒150-0001 東京都渋谷区神宮前 6-6-11
電話 03-3409-5364
Fax 03-3409-0147
Email info@tan.co.jp
WWW http://www.tan.co.jp/

Part I

Using XRT/gear

1

XRT/gear について

XRT/gear とは ■ XRT/gear と Motif クラス階層

環境変数 ■ プログラムのコンパイルと許可

リソース・ファイルの使用 ■ C++ との使用

XRT/gear の国際化

1.1 XRT/gear とは

XRT/gear は、Motif アプリケーションの開発に便利なウィジェットやユーティリティ集です。

XRT/gear のウィジェット

XRT/gear には、次のウィジェットが提供されています。

- *Enhanced Pushbutton* と *Enhanced Label* は、標準の Motif の PushButton と Label ウィジェットの機能の他に、pixmap とテキストが同時に表示できるようになっています。スペース、レイアウト、回転、枠線、切り取り等をコントロールするリソースが提供されています。
- *Enhanced Toggle* では、標準の Motif の ToggleButton の機能の他に、複数の状態を持つトグルボタンが作成できます。状態を示す pixmap は、XRT/gear から提供されているものの他、新たに定義したものが使用できます。
- *Tab Manager* と *Tab Button* は、ファイル・フォルダ形式にアプリケーション・ウィジェットを記憶、表示します。位置、方向、スタイル、スペース、タブの回転等をコントロールするリソースが提供されています。
- *Toolbar* は、ボタンやウィジェットを並べて表示します。エンドユーザはアイコンをクリックするだけ、Motif のメニュー・バーからアイテムを探す必要はありません。スペース、マージン、方向等をコントロールするリソースが提供されています。Primitive Widget ならどれでも Toolbar に表示できます。
- *Aligner* は、ラベルとウィジェットがペアになって表示されている列を作成し、見やすいインターフェースを作ります。リソースは、マージン、スペース、方向をコントロールします。
- *Outliner* は、データをツリー構造に表示します。アイテムの関係性を一目で見ることができます。

- *Progress* は、タスクの現在の状態をエンドユーザに知らせます。
- *Combo Box* は、エンドユーザにリストから選択させる時に使用します。
- *Paned Window* は、ウィジェットを縦または横方向に配置します。

XRT/gear のユーティリティ

XRT/gear には、次のユーティリティが提供されています。

- *Widget Tip* は、ウィジェットにポップアップするヘルプを提供します。アプリケーションでは、ヘルプをいつ表示するかを指定でき、色、行揃え、フォント、タイミング、ウィジェットのグループ化等もコントロールできます。
- *Widget Snapshot* は、(Manager ウィジェットも含む)あらゆる Motif ウィジェットのイメージを PostScript フォーマットにプリントします。マージン、方向、用紙サイズ、スケーリング等はパラメータでコントロールできます。さらに、Motif ウィジェットの pixmap や XImage を取得するルーチンも提供されています。
- *Icon Library* は、パブリック・ドメインのアイコンを 3000 個以上集めたライブラリです。これらのアイコンは、機能、サイズ、色等に区分けされています。アイコンのビューワは、*$XRTHOME/src/gear/demos/iconviewer* ディレクトリにあります。
- *Color Editor* は、簡単にアプリケーションに組み込むことができる色の編集用オブジェクトです。ユーザは、色を色名、16 進数、RGB、HSB(Hue[色調]、Saturation[彩度]、Brightness[明るさ])で指定することができます。Color Editor は、ユーザ・インタラクションを始め、色の割り当てや選択された色のプレビューを行います。

1.2 XRT/gear と Motif クラス階層

XRT/gear は Motif ツールキットに新たなウィジェットを追加します。このウィジェットは、他の Motif プログラミングと同様、Xt Intrinsics で操作することができます。
図 1 は、XRT/gear が Motif クラス階層のどこに位置するか示しています。

Xt Intrinsics は、ウィジェットをクラスに編成するオブジェクト指向アーキテクチャを定義します。例えば、XmPushButton ウィジェットは、Xt Core ウィジェットのサブクラスの Xm Primitive のサブクラスの XmLabel のサブクラスです。各 XRT/gear のウィジェットは、その親のウィジェット・クラスで定義されているリソースを使用することができます。(Enhanced Toggle ウィジェットは、XmToggleButton、XmLabel、XmPrimitive、Core ウィジェット・クラスのリソースを使用することができます。)

1.3 環境変数

最初に UNIX の環境をご確認ください。*XRTHOME* という環境変数が、XRT/gear のファイルのディレクトリ名に定義されていなければなりません。

次のように入力して、XRTHOME の現在の値を確認できます。

```
echo $XRTHOME
```

もし XRTHOME 環境変数が正しく設定されていない場合は、リリース&インストレーション･ガイドを参照するか、システム管理者の方にお問い合わせください。

また Motif のアプリケーションがコンパイルできることもご確認ください。ご使用になっている Motif のマニュアルで、Motif のライブラリとインクルード･ファイルの位置を調べることができます。XRT/gear を使用される前に、簡単な Motif のアプリケーションがコンパイルでき稼動することも必ずご確認ください。

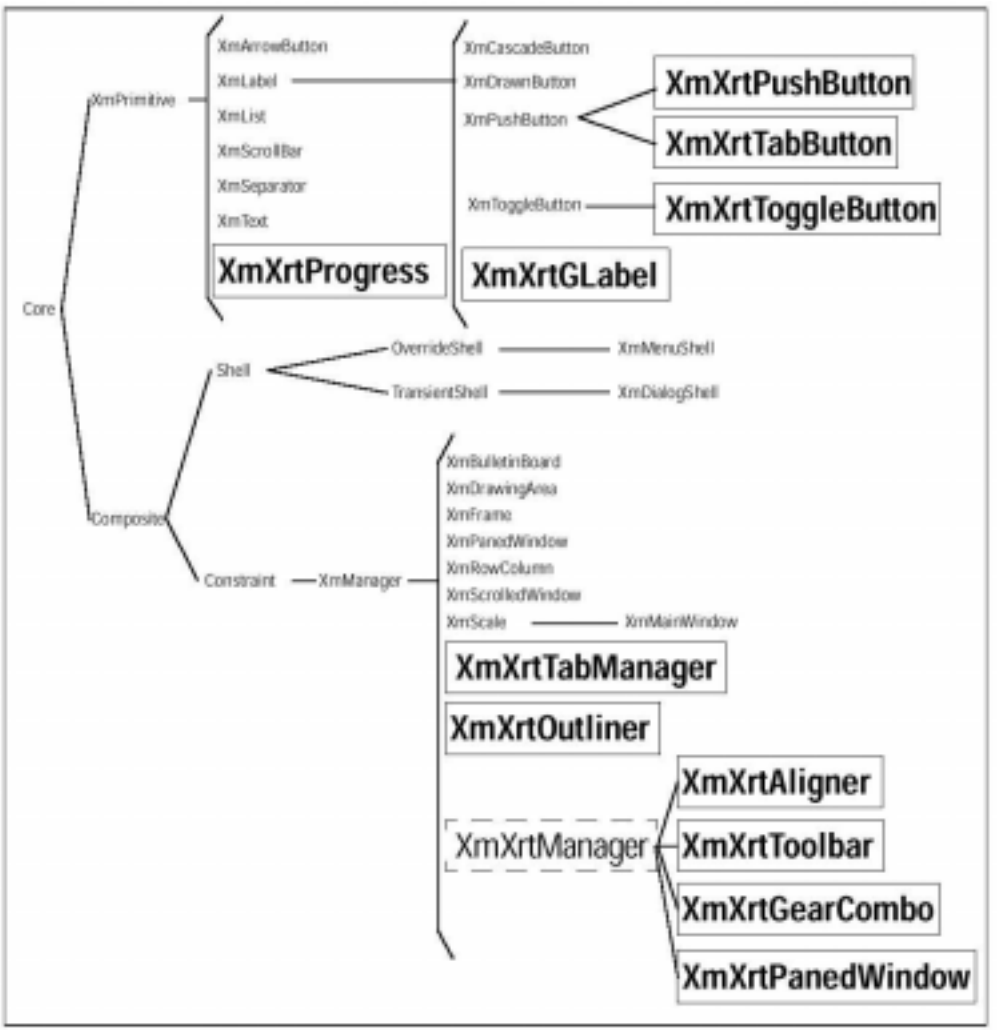


図 1　Motif クラス階層での XRT/gear の位置

1.4 プログラムのコンパイルと許可

X や Motif のライブラリとインクルード・ファイルが標準の場所(*/usr/lib*、*/usr/include/X11*、*/usr/include/Xm*)にインストールされていれば、次のようにして、simple.c をコンパイルし、許可を与えることができます。[1]

```
cc   simple.c -I$(XRTHOME)/include \
  -L$(XRTHOME)/lib -o simple \
  -lxrtgear -lXm -lXpm -lXt -lX11 -Xext -lm

  $XRTHOME/bin/xrt_auth simple
```

もし、ライブラリとインクルード・ファイルが標準の位置になければ、`-I` と `-L` のコンパイラ・フラグが必要です。

xrt_auth

XRT/gear のライブラリが実行可能アプリケーションにリンクされたら、*xrt_auth* を使用して許可を与えます。*xrt_auth* は、ご使用のマシンに XRT/gear のライセンスがあるかを調べ、アプリケーションに許可を与えます。*xrt_auth* には、`-v`(verbose message;詳細メッセージ)と`-q`(quiet mode;メッセージ出力なし)の二つのフラグのオプションがあります。

```
xrt_auth [-q | -v] executable_file
```

エンドユーザ・アプリケーションは *xrt_auth* により許可が与えられると、どのマシンでも実行することができます。XRT/gear で作成されたアプリケーションにはランタイム費用はかかりません。

もし、アプリケーションが *xrt_auth* によって許可が与えられないと、XRT/gear ウィジェットはブランクのまま表示され、次のメッセージが *stdout* に書かれます。

```
This application is built with a disabled or an expired
Time-limited version of XRT/gear for Motif. Please contact
KL Group Inc. or your XRT/gear for Motif reseller for
Further information or to purchase a permanent license.
If you have a license, please check:
1) That you have installed your license using the program
      $XRTHOME/bin/xrtpasswd or with the install script.
2) That you have authorized your executable by running
      "xrt_auth". (For more information refer to
      "Compiling and Authorizing a Program" in Chapter 1 of the
      XRT/gear for Motif Programmer's Guide & Reference Manual.)
3) That you have the $XRTHOME environment variable defined.
4) That you have permission to read the license file in
$XRTHOME/etc/licenses.
5) If you have a temporary license, make sure that it has
      not expired.
```

図2　期限切れまたは許可されていないアプリケーションで表示されるメッセージ

[1] コンパイルのオプションは機種に依存します。御使用のマシンでのコンパイル例が、*$XRTHOME/src/gear/examples* の Makefile にありますので参照してください。

注意: 共用ライブラリのリンクや許可に関しては、リリースノートを参照してください。

C++と ANSI C

XRT/gear のインクルード・ファイルは、C++、ANSI C、"Kernighan and Ritchie" C と互換性があります。

1.5 リソース・ファイルの使用

リソースの値を設定するには、C コードより、リソース・ファイルを使用した方がよい場合があります。リソースファイルには次の利点があります。

- スピード ー コンパイルしないので、変更が瞬時に確認できます。
- 柔軟性 ー エンドユーザがニーズに合わせてリソース値を変更できます。

XRT/gear のほとんどのリソースは、リソース・ファイルで設定できますので、複数のリソース設定の結果を直ぐに確認できます。Appendix C には、リソース設定の構文が解説されています。

GUI ビルダとの使用

XRT/gear は、主要な GUI ビルダと統合して使用することができます。GUI ビルダから XRT/gear のリソースをセットするには、リソース・ファイルの場合と同じ構文が使用できます。詳細は Appendix C に解説されていますので参照してください。

1.6 C++との使用

XRT/gear では C++ を使用することができます。この場合、*$XRTHOME/include/Xm/XrtGear.hxx* を C++プログラムにインクルードしてください。このファイルは、次のように各 XRT/gear ウィジェットのクラスと constructor(コンストラクタ)を定義します。

Widget	Class	ConstructorFunction
Aligner	XrtAligner	XmCreateXrtAligner()
Enhanced Label	XrtGLabel	XmCreateXrtGLabel()
Enhanced Pushbutton	XrtPushButton	XmCreateXrtPushButton()
Tab Button	XrtTabButton	XmCreateXrtTabButton()
Tab Manager	XrtTabManager	XmCreateXrtTabManager()
Enhanced Toggle	XrtToggleButton	XmCreateXrtToggleButton()
Toolbar	XrtToolbar	XmCreateXrtToolbar()
Outliner	XrtOutliner	XmCreateXrtOutliner()
Progress	XrtProgress	XmCreateXrtProgress()

Widget	Class	ConstructorFunction
Combo Box	XrtGearCombo	XmCreateXrtGearCombo()
PanedWindow	XrtPanedWindow	XmCreateXrtPanedWindow()

次の表は、Outliner ウィジェットに使用されるオブジェクトのクラスと constructor です。

Object	Class	ConstructorFunction
XrtNode	XrtNode	XmCreateXrtNode()
XrtNodeFolder	XrtNodeFolder	XmCreateXrtNodeFolder()
XrtColumn	XrtColumn	XmCreateXrtColumn()
XrtNodeStyle	XrtNodeStyle	XmCreateXrtNodeStyle()

上記に加えて、*XRTGear.hxx* には下記が含まれています。

- *XtDestroyWidget()*を呼び出す、ウィジェットの destructor。
- 各リソースをセットする method(方法)。(例: `XmNxrtGearBorderType` をセットする method は *setBorderType()*)
- 各リソースを取り出す method。(例: `XmNxrtGearBorderType` の値を取得する method は、*getBorderType()*)
- 通常の Xt を実行する method; *addCallback()*、*destroy()*、*getDisplay()*、*getName()*、*getParent()*、*isManaged()*、*isRealized()*、*manageChild()*。

C++の例は、*$XRTHOME/src/gear/examples/simple_cpp.cxx* と *$XRTHOME/src/gear/examples/complex_cpp.cxx* にあります。

1.7 XRT/gear の国際化

ISO Latin1 文字セット

御使用のXサーバでサポートされている X フォントなら、どれでも XRT/gear で使用できます。ISO Latin1 は、北米・西欧の言語の ISO 標準文字セットです。Xフォントのうち、名前の最後に *iso8859-1* と付くフォントが ISO Latin1 フォントです。

2-byte・フォント

漢字等の 2-byte フォントは、(String ではなく)XmString 型のテキスト・ストリングであれば表示できます。

印刷

*XrtGearOutlinerVaDrawPS()*と *XrtGearOutlinerDrawPS()*は、ISO Latin1 文字セットまたは日本語文字セットを使用した PostScript を出力します。(但し、PostScript のデバイスによって、それがサポートされている場合)

2

Enhanced Label Widget と Enhanced Pushbutton Widget

サンプル･プログラム ■ ウィジェットの概要

ボタンの型の指定 ■ Pixmap の指定

テキストと pixmap の回転 ■ Enhanced Label の枠

テキストと pixmap の位置 ■ Enhanced Pushbutton - arm pixmap の指定

ソースの一覧 ■ 手続きと方法

ここでは、Enhanced Label と Enhanced Pushbutton ウィジェットの一般的な機能及び、それぞれのウィジェット特有の機能を解説しています。

2.1 サンプル･プログラム

次のコードは、Enhanced Label を作成します。

```
#include <Xm/XrtGLabel.h>

Widget    label, form;
Pixmap    memo, memo_mask;

label = XtVaCreateManagedWidget("EnhLabel",
  xmXrtGLabelWidgetClass, form,
  XmNlabelPixmap,     memo,
  XmNxrtGearMask,     memo_mask,
  XmNxrtGearLabelType,    XRTGEAR_TYPE_STRING_PIXMAP,
  XmNxrtGearStringLayout, XRTGEAR_STRING_BOTTOM,
  XmNlabelString,     XmStringCreateSimple("Memos"),
  NULL);
```

次ページの図は、このコードで作成されたウィジェットです。



図 3　サンプルの Enhanced Label

Enhanced Label で定義されているリソースのほとんどは、Enhanced Pushbutton でも定義されています。次のコードは、Enhanced Pushbutton ウィジェットを作成します。

```
#include <Xm/XrtPushB.h>

Widget button, form;
Pixmap memo, memo_mask;

button = XtVaCreateManagedWidget("EnhButton",
  xmXrtPushButtonWidgetClass, form,
  XmNlabelPixmap,     memo,
  XmNxrtGearMask,     memo_mask,
  XmNxrtGearLabelType,    XRTGEAR_TYPE_STRING_PIXMAP,
  XmNxrtGearStringLayout, XRTGEAR_STRING_BOTTOM,
  XmNlabelString,     XmStringCreateSimple("Memos"),
  NULL);
```

図 4 は、このコードで作成されたウィジェットです。



図 4　サンプルの Enhanced Pushbutton

2.2　ウィジェットの概要

Enhanced Label	Include File:	*<Xm/XrtGLabel.h>*
	Class Name:	XmXrtGLabel
	Class Hierarchy:	Core → XmPrimitive → XmLabel → XmXrtGLabel
	Class Pointer:	xmXrtGLabelWidgetClass
Enhanced Pushbutton	Include File:	*<Xm/XrtPushB.h>*
	Class Name:	XmXrtPushButton
	Class Hierarchy:	Core → Primitive → XmLabel → XmPushButton → XmXrtPushButton
	Class Pointer:	xmXrtPushButtonWidgetClass

Enhanced Pushbutton は XmPushButton のサブクラスなので、Enhanced Pushbutton は、Motif の PushButton を使用しているアプリケーションに直接挿入することができます。同様にして、Enhanced Label も Motif の Label ウィジェットを使用しているアプリケーションに挿入することができます。

Motif ウィジェットの階層と Enhanced Label 及び Enhanced Pushbutton が階層のどこに配置されるかは、1.2(P8)で確認してください。

2.3 ボタンの型の指定

XmNxrtGearLabelType リソースは、Enhanced Label または Enhanced Pushbutton に使用される型をコントロールします。このリソースは、次のどれかにセットできます。

XRTGEAR_TYPE_STRING	ラベルは XmString 型
XRTGEAR_TYPE_PIXMAP	ラベルは pixmap
XRTGEAR_TYPE_STRING_PIXMAP	ラベルは XmString と pixmap 両方を含む

XmNxrtGearLabelType のデフォルト値は XRTGEAR_TYPE_STRING です。

2.4 Pixmap の指定

Enhanced Label または Enhanced Pushbutton ウィジェットに pixmap を指定するには、XmNxrtGearLabelPixmap を使用します。このリソースは Pixmap 型です。

この pixmap は、XmNxrtGearLabelType が XRTGEAR_TYPE_PIXMAP または XRTGEAR_TYPE_STRING_PIXMAP の時に表示されます。XmNxrtGearLabelPixmap が指定されていない場合は、XmNlabelPixmap が使用されます。

XmNxrtGearLabelPixmap はリソースファイルでセットでき、XPM または XBM フォーマットをサポートしています。[1]

Clip マスクの使用

XmNxrtGearMask リソースは、Enhanced Label または Enhanced Pushbutton ウィジェットに表示されている pixmap に対する clip マスクを定義します。この clip マスクは、ウィジェットに pixmap をどれだけ表示するかコントロールします。マスクの 0 以外の値が描画されます。

[1] XmNlabelPixmap は、XPM フォーマットの pixmap をサポートしていません。

次のコードは clip マスクを使用して、イメージの表示をコントロールしています。

```
Pixmap image, mask;
Widget label;

label = XtVaCreateManagedWidget("EnhLabel",
  xmXrtGLabelWidgetClass, form,
  XmNxrtGearLabelType,    XRTGEAR_TYPE_PIXMAP,
  XmNxrtGearLabelPixmap,  image,
  XmNxrtGearMask,     mask,
  NULL);
```

図 5 は、clip マスクがどのように表示をコントロールするかを示しています。



図 5　clip マスクとその結果

XmNxrtGearMask には Pixmap 型の値が必要です。このリソースのデフォルト値は XmUNSPECIFIED_PIXMAP です。clip マスクはモノクロです(bit depth=1)。一つのファイルで pixmap と clip マスクを両方指定できます。

2.5　テキストと pixmap の回転

Enhanced Label と Enhanced Pushbutton のテキストと pixmap は個別に回転させることができます。テキストを回転させるには、XmNxrtGearStringRotation を図 6 にある値の一つにセットします。



図 6 テキストの回転

回転は反時計回りに行われます。デフォルトは XRTGEAR_ROTATE_NONE です。

pixmap を回転させるには、XmNxrtGearPixmapRotation をセットします。このリソースは、XmNxrtGearStringRotation と同じ値にセットすることもできます。



図 7　pixmap の回転

XmNxrtGearPixmapRotation のデフォルトは XRTGEAR_ROTATE_NONE です。

2.6　Enhanced Label の枠

Enhanced Label の枠は、XmNxrtGearBorderType で指定します。有効な値は図 8 の通りです。

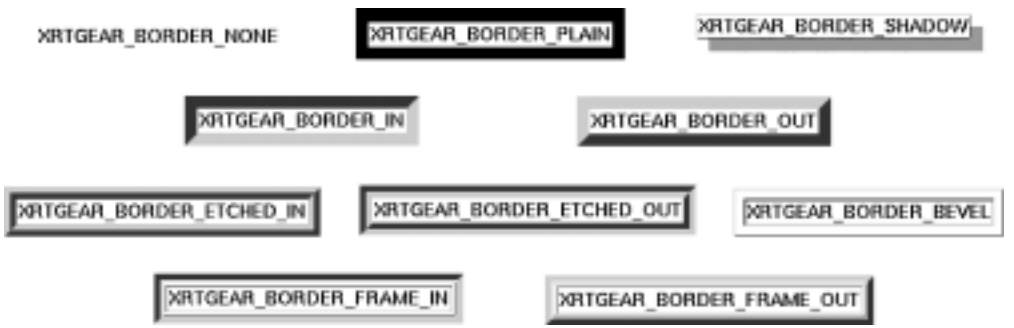


図 8 Enhanced Label の枠

枠線の幅は XmNshadowThickness で指定します。デフォルトでは、このリソースは 5 にセットされています(Core のリソースでセットされるデフォルト値 2 を上書きします)。

XmNxrtGearBorderType は、Enhanced Pushbutton ウィジェットでは定義されていません。

2.7　テキストと pixmap の位置

XmNxrtGearLabelType が XRTGEAR_TYPE_STRING_PIXMAP にセットされている場合、テキスト･ストリングと pixmap の両方が、Enhanced Label または Enhanced Pushbutton に表示されます。テキストと pixmap の配置をコントロールするリソースは、次の二つです。

- テキスト･ストリングと pixmap の間の距離を pixel で指定する XmNxrtGearLayoutSpacing
- pixmap に対するテキスト･ストリングの位置を相対的に指定する XmNxrtGearStringLayout

XmNxrtGearStringLayout は図 9 の値にセットできます。

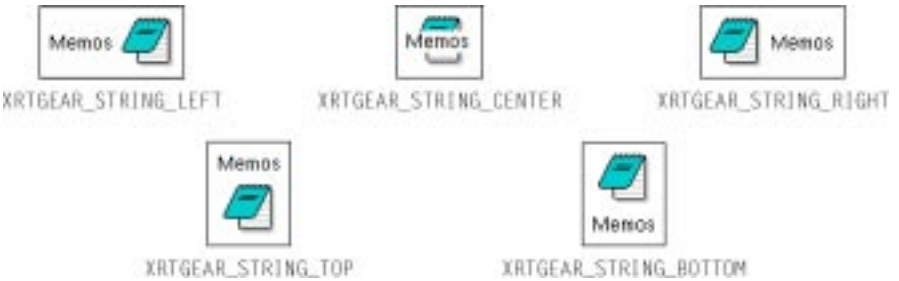


図 9　ストリングと pixmap の配置

XmNxrtGearStringLayout のデフォルト値は XRTGEAR_STRING_RIGHT です。

XmNxrtGearLayoutSpacing と XmNxrtGearStringLayout は、XmNxrtGearLabelType が XRTGEAR_TYPE_STRING_PIXMAP にセットされていなければ無視されます。

XmNxrtGearActive は、ボタンの周囲に線を描画するかどうかを指定します。デフォルトは False で、線が描画されます。True の場合には、この線は描画されませんが、マウス･ポインタがボタン上に来た時には、線が表示されます。

2.8　Enhanced Pushbutton - arm pixmap の指定

Enhanced Pushbutton では、エンドユーザがボタンを押したままの状態の時に表示される pixmap を定義することができます。この pixmap は、*arm pixmap* と呼ばれます。

arm pixmap を定義するには、XmNxrtGearArmPixmap をセットします。この pixmap は、XmNxrtGearLabelType が XRTGEAR_TYPE_PIXMAP または XRTGEAR_TYPE_STRING_PIXMAP の場合に表示されます。次は、XmNxrtGearArmPixmap を使用した例です。

```
button = XtVaCreateManagedWidget("EnhPushB",
  xmXrtPushButtonWidgetClass, form,
  XmNxrtGearArmPixmap,    timer,
  XmNxrtGearLabelType,    XRTGEAR_TYPE_STRING_PIXMAP,
  XmNxrtGearLabelPixmap,  smile,
  XmNlabelString,
    XmStringCreateSimple("XRT/gear's Enhanced Pushbutton"),
  NULL);
```

図 10 は、このコードで作成されたボタンの表示です。



ボタンが押されていない状態　　ボタンが押されている状態

図 10　arm pixmap

XmNxrtGearArmPixmap はリソースファイルでセットでき、XPM または XBM フォーマットがサポートされています。[1] (XmNarmPixmap は XPM をサポートしていません。)

XmNxrtGearArmPixmap が指定されていない場合は、XmNarmPixmap が使用されます。

arm pixmap に clip マスクを定義する場合は、XmNxrtGearArmMask をセットします。clip マスクはモノクロ(bit depth=1)でなければなりません。XmNxrtGearArmPixmap と XmNxrtGearArmMask には同じファイルが使用できます。

XmNxrtGearArmPixmap と XmNxrtGearArmMask は、Enhanced Label ウィジェットでは定義されていません。

2.9　リソースの一覧

Enhanced Label と Enhanced Pushbutton のリソースと、よく使用される継承されたリソースの一覧です。リソース名の後は、データ型、デフォルト値、そしてリソースと共に使用される手続きです。**C** は、そのリソースが create(例: *XtCreateWidget()*)で定義でき、**S** は set(例; *XtSetValues()*)でき、**G** は get(例: *XtGetValues()*)で使用できることを意味します。

2.9.1　Enhanced Label のリソース

XmNbackground　Pixel　dynamic　CSG

ラベルの背景色を指定します。

XmNxrtGearBorderType　XrtGearBorderType　XRTGEAR_BORDER_NONE　CSG

ラベルの枠に使用されるスタイルを指定します。有効な値は次の通りです。

XRTGEAR_BORDER_PLAIN	前景色で描かれる装飾無しの線
XRTGEAR_BORDER_SHADOW	影付き
XRTGEAR_BORDER_IN	ラベルが押されているように見える線
XRTGEAR_BORDER_OUT	ラベルが持ち上がっているように見える線
XRTGEAR_BORDER_ETCHED_IN	ラベルが押されているように見える二重線
XRTGEAR_BORDER_ETCHED_OUT	ラベルが持ち上がっているように見える二重線

[1] XmNxrtGearLabelPixmap と XmNxrtGearArmPixmap が両方セットされている場合は、ボタンが押されていない時は label pixmap、ボタンが押されている時は arm pixmap が表示されます。これらの二つの pixmap は同時には表示されません。

XRTGEAR_BORDER_BEVEL	1pixel の枠、影はその 1/2 の太さで描画
XRTGEAR_BORDER_FRAME_IN	ラベルが押されているように見える 1 pixel の影付きの枠
XRTGEAR_BORDER_FRAME_OUT	ラベルが持ち上がっているように見える 1 pixel の影付きの枠
XRTGEAR_BORDER_NONE	枠線なし

枠線の幅は XmNshadowThickness で指定します。

XmNfontList　XmFontList　dynamic　CSG

ウィジェットのテキストに使用されるフォント・リストを指定します。

XmNforeground　Pixel　dynamic　CSG

ラベルの前景色を指定します。

XmNheight　Dimension　dynamic　CSG

ウィンドウ高さを pixel で指定します。

XmNxrtGearLabelPixmap　Pixmap　XmUNSPECIFIED_PIXMAP　CSG

XmNxrtGearLabelType が XRTGEAR_TYPE_PIXMAP または XRTGEAR_TYPE_STRING_PIXMAP の時に表示される pixmap を指定します。XmNxrtGearLabelPixmap と XmNxrtGearMask は同じ XPM ファイルを使用できます。

XmNlabelString　XmString　dynamic　CSG

XmNxrtGearLabelType が XRTGEAR_TYPE_STRING または XRTGEAR_TYPE_STRING_PIXMAP の時に表示されるストリングを指定します。

XmNxrtGearLabelType　XrtGearType　XRTGEAR_TYPE_STRING　CSG

テキスト・ストリング、pixmap、または両方を表示するかどうか指定します。有効な値は、XRTGEAR_TYPE_STRING(XmString) 、 XRTGEAR_TYPE_PIXMAP(Pixmap) または XRTGEAR_TYPE_STRING_PIXMAP です。XmNlabelType が XmSTRING または XmPIXMAP にセットされている時は、対応する XmNxrtGearLabelType リソースをセットします。XmNlabelType と XmNxrtGearLabelType が両方とも同時にセットされる場合は、XmNxrtGearLabelType が優先されます。

XmNxrtGearLayoutSpacing　Dimension　2　CSG

テキストと pixmap の間の距離を pixel で指定します。このリソースは、XmNxrtGearLabelType が XRTGEAR_TYPE_STRING_PIXMAP の時にのみ使用されます。

XmNmarginHeight　Dimension　2　CSG

ラベルの上部と下部の端とその影の間の距離を指定します。

XmNmarginWidth　Dimension　2　CSG

ラベルの左と右の端とその影の間の距離を指定します。

XmNxrtGearMask Pixmap XmUNSPECIFIED_PIXMAP CSG

pixmap を clip(切り取り)するマスクを指定します。マスクのゼロ以外の値部分が描画されます。clip マスクはモノクロ(bit depth=1)でなければなりません。XmNxrtGearLabelPixmap と XmNxrtGearMask は同じ XPM ファイルを使用できます。

XmNxrtGearPixmapRotation XrtGearRotation XRTGEAR_ROTATE_NONE CSG

pixmap の回転を指定します。有効な値は、XRTGEAR_ROTATE_NONE、XRTGEAR_ROTATE_90、XRTGEAR_ROTATE_180、XRTGEAR_ROTATE_270 のどれかです。回転は反計回りに行われます。

XmNshadowThickness Dimension 5 CSG

Primitive から継承されている、このリソースのデフォルト値(2)を置き換えます。

XmNxrtGearStringLayout XrtGearStringLayout XRTGEAR_STRING_RIGHT CSG

pixmap に対するテキストの相対的な位置を指定します。有効な値は、XRTGEAR_STRING_CENTER、XRTGEAR_STRING_TOP、XRTGEAR_STRING_LEFT、XRTGEAR_STRING_BOTTOM、XRTGEAR_STRING_RIGHT です。

このリソースは、XmNxrtGearLabelType が XRTGEAR_TYPE_STRING_PIXMAP でない場合は無視されます。

XmNxrtGearStringRotation XrtGearRotation XRTGEAR_ROTATE_NONE CSG

テキストの回転を指定します。有効な値は、XRTGEAR_ROTATE_NONE、XRTGEAR_ROTATE_90、XRTGEAR_ROTATE_180、XRTGEAR_ROTATE_270 です。回転は反時計回りに行われます。

XmNuserData XtPointer NULL CSG

アプリケーションで定義されているデータのポインタを指定します。

XmNwidth Dimension dynamic CSG

ウィンドウの幅を pixel で指定します。

XmNx Position 0 CSG

親ウィジェットの内側の左上角に対応する、ウィジェットの外側の左上角の X 座標。

XmNy Position 0 CSG

親ウィジェットの内側の左上角に対応する、ウィジェットの外側の左上角の Y 座標。

2.9.2 Enhanced Pushbutton のリソース

XmNxrtGearActive boolean False CSG

True にセットされている場合、マウス・ポインタがボタンの上にない時は、そのボタンの周囲に線は表示されません。

XmNxrtGearArmMask Pixmap XmUNSPECIFIED_PIXMAP CSG

XmNxrtGearArmPixmap で定義された arm pixmap に使用される clip マスクを指定します。マスクのゼロ以外の値部分が描画されます。clip マスクはモノクロ(bit depth=1)でなければなりません。XmNxrtGearArmPixmap と XmNxrtGearArmMask は同じ XPM ファイルを使用することができます。

XmNxrtGearArmPixmap Pixmap XmUNSPECIFIED_PIXMAP CSG

ボタンが押されている状態を示す pixmap を指定します。この pixmap は、XmNxrtGearLabelType が XRTGEAR_TYPE_PIXMAP または XRTGEAR_TYPE_STRING_PIXMAP の時にのみ表示されます。XmNxrtGearArmPixmap と XmNxrtGearArmMask は同じ XPM ファイルを使用できます。

XmNbackground Pixel dynamic CSG

ボタンの背景色を指定します。

XmNfontList XmFontList dynamic CSG

ウィジェットのテキストに使用されるフォント・リストを指定します。

XmNforeground Pixel dynamic CSG

ボタンの前景色を指定します。

XmNheight Dimension dynamic CSG

ウィンドウの高さを pixel で指定します。

XmNxrtGearLabelPixmap Pixmap XmUNSPECIFIED_PIXMAP CSG

XmNxrtGearLabelType が XRTGEAR_TYPE_PIXMAP または XRTGEAR_TYPE_STRING_PIXMAP の時に表示されるラベルの pixmap を指定します。XmNxrtGearLabelPixmap と XmNxrtGearMask は同じ XPM ファイルを使用できます。

XmNlabelString XmString dynamic CSG

XmNxrtGearLabelType が XRTGEAR_TYPE_STRING または XRTGEAR_TYPE_STRING_PIXMAP の時に表示されるラベルのストリングを指定します。

XmNxrtGearLabelType XrtGearType XRTGEAR_TYPE_STRING CSG

ラベルの型を指定します。値は、次のうちのどれかです。

XRTGEAR_TYPE_STRING	XmNlabelString が表示される
XRTGEAR_TYPE_PIXMAP	XmNxrtGearLabelPixmap が表示される
XRTGEAR_TYPE_STRING_PIXMAP	XmNlabelString と XmNxrtGearLabelPixmap の両方が表示される

XmNlabelType が XmSTRING または XmPIXMAP にセットされている時は、対応する XmNxrtGearLabelType リソースをセットします。XmNlabelType と XmNxrtGearLabelType の両方が同時にセットされる時は、XmNxrtGearLabelType が優先されます。

XmNxrtGearLayoutSpacing Dimension 2 CSG

テキストと pixmap の間の距離を pixel で指定します。このリソースは、XmNxrtGearLabelType が XRTGEAR_TYPE_STRING_PIXMAP の時にのみ使用されます。

XmNmarginHeight Dimension 2 CSG

ボタンの上部と下部の端とその影の間の距離を指定します。

XmNmarginWidth Dimension 2 CSG

ボタンの左と右の端とその影の間の距離を指定します。

XmNxrtGearMask Pixmap XmUNSPECIFIED_PIXMAP CSG

XmNxrtGearLabelPixmap で定義された pixmap に使用される clip マスクを指定します。マスクのゼロ以外の値部分が表示されます。clip マスクはモノクロ(bit depth=1)でなければなりません。XmNxrtGearLabelPixmap と XmNxrtGearMask は同じ XPM ファイルを使用することができます。

XmNxrtGearPixmapRotation XrtGearRotation XRTGEAR_ROTATE_NONE CSG

pixmap の回転を指定します。有効な値は、XRTGEAR_ROTATE_NONE、XRTGEAR_ROTATE_90、XRTGEAR_ROTATE_180、XRTGEAR_ROTATE_270 です。回転は反時計回りに行われます。

XmNxrtGearStringLayout XrtGearStringLayout XRTGEAR_STRING_RIGHT CSG

pixmap に対するテキストの相対的な位置を指定します。有効な値は、XRTGEAR_STRING_CENTER 、 XRTGEAR_STRING_TOP 、 XRTGEAR_STRING_LEFT 、 XRTGEAR_STRING_BOTTOM、XRTGEAR_STRING_RIGHT です。

このリソースは、XmNxrtGearLabelType が XRTGEAR_TYPE_STRING_PIXMAP 以外の場合は無視されます。

XmNxrtGearStringRotation XrtGearRotation XRTGEAR_ROTATE_NONE CSG

テキストの回転を指定します。有効な値は、XRTGEAR_ROTATE_NONE 、XRTGEAR_ROTATE_90、XRTGEAR_ROTATE_180、XRTGEAR_ROTATE_270 です。回転は反時計回りに行われます。

XmNuserData XtPointer NULL CSG

アプリケーションで定義されているデータのポインタを指定します。

XmNwidth Dimension dynamic CSG

ウィンドウの幅を pixel で指定します。

XmNx Position 0 CSG

親ウィジェットの内側の左上角に対応する、ウィジェットの外側の左上角の X 座標

XmNy Position 0 CSG

親ウィジェットの内側の左上角に対応する、ウィジェットの外側の左上角の Y 座標

2.9.3 Enhanced Label によって継承されるリソース

Resource	Inherited From	Resource	Inherited From
XmNaccelerator	XmLabel	XmNlabelString	XmLabel
XmNacceleratorS	Core	XmNmappedWhenManaged	Core
XmNalignment	XmLabel	XmNmarginBottom	XmLabel
XmNancestorSensitive	Core	XmNmarginHeight	XmLabel
XmNbackground	Core	XmNmarginLeft	XmLabel
XmNbackgroundPixmap	Core	XmNmarginRight	XmLabel
XmNborderColor	Core	XmNmarginWidth	XmLabel
XmNborderPixmap	Core	XmNmarginTop	XmLabel
XmNborderWidth	Core	XmNmnemonic[a]	XmLabel
XmNbottomShadowColor	Primitive	XmNmnemonicCharSet[a]	XmLabel
XmNbottomShadowPixmap	Primitive	XmNnavigationType	Primitive
XmNcolormap	Core	XmNrecomputeSize	XmLabel
XmNdepth	Core	XmNscreen	Core
XmNdestroyCallback	Core	XmNsensitive	Core
XmNfontList	XmLabel	XmNshadowThickness	Primitive
XmNforeground	Primitive	XmNstringDirection	XmLabel
XmNheight	Core	XmNtopShadowColor	Primitive
XmNhelpCallback	Primitive	XmNtopShadowPixmap	Primitive
XmNhighlightColor	Primitive	XmNtranslations	Core
XmNhighlightOnEnter	Primitive	XmNtraversalOn	Primitive
XmNhighlightPixmap	Primitive	XmNuserData	Primitive
XmNhighlightThickness	Primitive	XmNwidth	Core
XmNinitialResourcesPersistent	Core	XmNx	Core
XmNlabelPixmap	XmLabel	XmNy	Core

a: `XmNxrtGearStringRotation` が `XRTGEAR_ROTATE_NONE` の場合にのみサポートされる

2.9.4 Enhanced Pushbutton によって継承されるリソース

Resource	Inherited From
XmNaccelerator	XmLabel
XmNacceleratorS	Core
XmNactivateCallback	XmPushButton
XmNalignment	XmLabel
XmNancestorSensitive	Core
XmNarmCallback	XmPushButton
XmNarmColor	XmPushButton
XmNarmPixmap	XmPushButton
XmNbackground	Core
XmNbackgroundPixmap	Core
XmNborderColor	Core
XmNborderPixmap	Core
XmNborderWidth	Core
XmNbottomShadowColor	Primitive
XmNbottomShadowPixmap	Primitive
XmNcolormap	Core
XmNdefaultButtonShadowThickness	XmPushButton
XmNdepth	Core
XmNdestroyCallback	Core
XmNdisarmCallback	XmPushButton
XmNfillOnArm	XmPushButton
XmNfontList	XmLabel
XmNforeground	Primitive
XmNheight	Core
XmNhelpCallback	Primitive
XmNhighlightColor	Primitive
XmNhighlightOnEnter	Primitive
XmNhighlightPixmap	Primitive
XmNhighlightThickness	Primitive

Resource	Inherited From
XmNinitialResourcesPersistent	Core
XmNlabelPixmap	XmLabel
XmNlabelString	XmLabel
XmNmappedWhenManaged	Core
XmNmarginBottom	XmLabel
XmNmarginHeight	XmLabel
XmNmarginLeft	XmLabel
XmNmarginRight	XmLabel
XmNmarginTop	XmLabel
XmNmarginWidth	XmLabel
XmNmnemonic[a]	XmLabel
XmNmnemonicCharSet[a]	XmLabel
XmNmultiClick	XmPushButton
XmNnavigationType	Primitive
XmNrecomputeSize	XmLabel
XmNscreen	Core
XmNsensitive	Core
XmNshadowThickness	Primitive
XmNshowAsDefault	XmPushButton
XmNstringDirection	XmLabel
XmNtopShadowColor	Primitive
XmNtopShadowPixmap	Primitive
XmNtranslations	Core
XmNtraversalOn	Primitive
XmNuserData	Primitive
XmNwidth	Core
XmNx	Core
XmNy	Core

a: `XmNxrtGearStringRotation` が `XRTGEAR_ROTATE_NONE` の場合にのみサポートされる

2.10 手続きと方法

Enhanced Label と Enhanced Pushbutton の手続きと方法の一覧です。

XmCreateXrtGLabel()

Enhanced Label ウィジェットを作成します。

```
Widget
XmCreateXrtGLabel(
  Widget  parent,
  String  name,
  Arg     *arglist,
  int     argcount
  )
```

parent は親、*name* は作成されるウィジェット名、*arglist* は、*argcount* 引数のリストです。

XmCreateXrtPushButton()

Enhanced Pushbutton ウィジェットを作成します。

```
Widget
XmCreateXrtPushButton(
  Widget  parent,
  String  name,
  Arg     *arglist,
  int     argcount
  )
```

parent は親、*name* は作成されるウィジェット名、*arglist* は *argcount* 引数のリストです。

XmIsxrtGLabel()

ウィジェットが Enhanced Label であれば `True` を返します。

```
Boolean
XmIsXrtGLabel(
  Widget  widget
  )
```

XmIsXrtPushButton()

ウィジェットが Enhanced Pushbutton であれば `True` を返します。

```
Boolean
XmIsXrtPushButton(
  Widget  widget
  )
```

XrtGearMrmInitialize()

UIL アプリケーションに使用するために、Mrm に XRT/gear ウィジェットを登録します。この手続きは、*MrmInitialize()*の後に呼び出して下さい。

```
void
XrtGearMrmInitialize(void)
```

XrtGearSetXmStringConverter()

XmRXmString リソース変換プログラムをセットし、XmString 値を作成します。

```
void
XrtGearSetXmStringConverter(void)
```

変換は、エンベッドされたフォント(タグ)を指定するために、次のシンタックスを使用します。

@font@　　フォントのタグは `XmNfontList`

@P@　　前のフォント

\@　　"@"とする

例:

```
For a @bold@free evaluation@P@ of any product
```

フォント・リストを設定

```
fixed,-*-times-bold-r-normal-*-140-*=bold
```

表示結果

For a **free evaluation** of any product

3

Enhanced Toggle Widget

サンプル･プログラム ■ ウィジェットの概要

ボタンの状態 ■ ボタンの色の設定

リソースの一覧 ■ 手続きと方法

3.1 サンプル･プログラム

次のプログラムは、三つの Enhanced Toggle ウィジェットを含む RowColumn ウィジェットを作成します。このプログラムは、幾つかの項目から選択を行う、最も一般的の Enhanced Toggle の使用法を示しています。

```
#include <stdio.h>
#include <stdlib.h>
#include <Xm/RowColumn.h>
#include <Xm/XrtToggleB.h>

int
main(int argc, char *argv[])
{
  XtAppContext        app_context;
  Widget      toplevel;
  Widget      menu, button1, button2, button3;

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

  menu = XmCreateRowColumn(toplevel, "laundry", NULL, 0);
  XtVaSetValues(menu,
    XmNorientation,   XmVERTICAL,
    NULL);
  button1 = XtVaCreateManagedWidget("button1",
    xmXrtToggleButtonWidgetClass, menu,
    XmNlabelString,        XmStringCreateSimple("Wash"),
    XmNxrtGearIndicator,   XRTGEAR_INDICATOR_CROSS_BOX,
    NULL);
  button2 = XtVaCreateManagedWidget("button2",
    xmXrtToggleButtonWidgetClass, menu,
    XmNlabelString,        XmStringCreateSimple("Dry"),
    XmNxrtGearIndicator,   XRTGEAR_INDICATOR_CROSS_BOX,
    NULL);
```

```
    button3 = XtVaCreateManagedWidget("button3",
      xmXrtToggleButtonWidgetClass, menu,
      XmNlabelString,        XmStringCreateSimple("Fold"),
      XmNxrtGearIndicator,   XRTGEAR_INDICATOR_CROSS_BOX,
      NULL);
    XtManageChild(menu);
    XtRealizeWidget(toplevel);
    XtAppMainLoop(app_context);
  }
```

図 11 は、このコードによって作成されたウィジェットです。この例では、エンドユーザが"Wash"ボタンをクリックしています。



図 11　サンプル・プログラムで表示される Toggle

3.2　ウィジェットの概要

Include File:　*<Xm/XrtToggleB.h>*
Class Name:　XmXrtToggleButton
Class Hierarchy:　Core → XmPrimitive → XmLabel → XmToggleButton → XmXrtToggleButton
Class Pointer:　xmXrtToggleButtonWidgetClass

Enhanced Toggle は XmToggleButton ウィジェットのサブクラスです。

使用されない、継承されたリソース

Enhanced Toggle は XmToggleButton のサブクラスなので、XmToggleButton のほとんどの機能を継承しています。但し、`XmNvalueChangedCallback` と `XmNset` リソースの代りに、`XmNxrtGearStateChangedCallback` と `XmNxrtGearState` リソースを使用しています。

3.3　ボタンの状態

デフォルトでは、Enhanced Toggle は、オン(選択状態)またはオフ(非選択状態)です。`XmNxrtGearState` リソースは、ウィジェットの現在の状態を指定します。このリソースが `XRTGEAR_STATE_ON` であれば選択、`XRTGEAR_STATE_OFF` であれば非選択状態です。エンドユーザがトグル・ボタンを押すと、このリソースは `XRTGEAR_STATE_ON` または `XRTGEAR_STATE_OFF` になります。

状態を示す表示

各 Enhanced Toggle ウィジェットには、トグル・ボタンの現在の状態を示すインディケータが提供されています。XmNxrtGearIndicator リソースがこのインディケータのスタイルをコントロールし、値は次の表の通りです。

XmNxrtGearIndicator の値	インディケータのスタイル	Off の状態	On の状態
XRTGEAR_INDICATOR_NONE	インディケータは表示されない	インディケータは表示されない	
XRTGEAR_INDICATOR_FILL	角ボタンの影で押下を示す(デフォルト)		
XRTGEAR_INDICATOR_CHECK	前景色を使用したチェックマーク		
XRTGEAR_INDICATOR_CHECK_BOX	箱の中に(前景色を使用した)チェックマーク		
XRTGEAR_INDICATOR_CIRCLE	丸ボタンの影で押下を示す		
XRTGEAR_INDICATOR_CIRCLE_BOX	(前景色を使用した)箱の中に丸ボタン、影で押下を示す		
XRTGEAR_INDICATOR_CROSS	(前景色を使用した)クロス		
XRTGEAR_INDICATOR_CROSS_BOX	箱の中に(前景色を使用した)クロス		
XRTGEAR_INDICATOR_PIXMAP	pixmap	下の解説参照	

XmNxrtGearIndicator が XRTGEAR_INDICATOR_PIXMAP にセットされている場合、インディケータの pixmap のリストは、XmNxrtGearIndicatorPixmapList で指定されます。このリストの終端は NULL としてください。[1]

[1] リストの pixmap が XmUNSPECIFIED_PIXMAP の場合は、リスト上の最後の有効な pixmap が点描で表示されます。

clip pixmap

インディケータ pixmap を切り取る場合は、XmNxrtGearIndicatorMaskList リソースをセットすることによって clip pixmap のリストを定義してください。このリストの終端は NULL としてください。各 clip マスクはモノクロ(bit depth=1)で、このリストのエレメントの数は、XmNxrtGearIndicatorPixmapList リソースのエレメントの数と同じにしてください。インディケータ pixmap に clip pixmap が指定されない時は、リストのそのエントリに対しては、UNSPECIFIED_PIXMAP としてください。

複数の状態

XmNxrtGearNumStates リソースを設定して、ウィジェットには複数の状態を指定することができます。このリソースが 2 より大きい値にセットされると、XmNxrtGearState の値が 0 から始まり、エンドユーザがトグル・ボタンを押す度に、この値が増えていきます。最大値(選択可能な状態の数より 1 少ない数)に達すると、その後に押されるボタンのイベントは、XmNxrtGearState の値を 0(XRTGEAR_STATE_OFF の状態)からまた増加させていきます。

複数の状態は、アプリケーションで定義されたインディケータのスタイルと一緒に使用することができます。次のコードは、トグル・ボタンの四つの状態を定義し、四つのそれぞれの状態のインディケータ pixmap のリストと clip pixmap リストを指定しています。

```
Pixmap timerlist[5], masklist[5];

button = XtVaCreateManagedWidget("EnhToggleB",
  xmXrtToggleButtonWidgetClass,  form,
  XmNlabelString,                XmStringCreateSimple("Time"),
  XmNxrtGearIndicatorPixmapList, timerlist,
  XmNxrtGearIndicatorMaskList,   masklist,
  XmNxrtGearIndicator,           XRTGEAR_INDICATOR_PIXMAP,
  XmNxrtGearNumStates,           4,
  NULL);
```

状態の変化を告げるコールバック

Enhanced Toggle の状態が変化する前または後に呼び出されるコールバックを指定することができます。このコールバックのリストは、XmNxrtGearStateChangedCallback リソースで指定できます。[1]

コールバックが開始されると、GearStateChanged コールバックには次の構造体が渡されます。

```
typedef struct {
  int     reason; // Read-only
  XEvent  *event; // Read-only
  int     state;
} XrtGearStateChangedCallbackStruct;
```

state メンバは、トグル・ボタンの新しい状態を指定します。*reason* は XRTGEAR_REASON_STATE_BEGIN(状態が変化する前) または XRTGEAR_REASON_STATE_END(状態が変化した後)のどちらかです。

[1] Enhanced Toggle では、XmNvalueChangedCallback は使用しないでください。

状態の変化の置き換えに GearStateChanged コールバックを使用するには、*reason* が XRTGEAR_REASON_STATE_BEGIN の時に *state* をセットしてください。

次のコードは、Enhanced Toggle が押される回数を数える、GearStateChanged コールバックの例です。

```
int click_count = 0;

void
stateCB(Widget w, XtPointer client_data,
  XrtGearStateChangedCallbackStruct *call_data)
{
  if (call_data->reason == XRTGEAR_REASON_STATE_BEGIN) {
    click_count++;
}
```

状態の変化の方法

*XrtGearToggleButtonGetState()*は、Enhanced Toggle ウィジェットの現在の状態を取得します。

```
state = XrtGearToggleButtonGetState(widget);
```

Enhanced Toggle ウィジェットの現在の状態をセットするには、*XrtGearToggleButtonSetState()*を呼び出してください。

```
XrtGearToggleButtonSetState(widget, XRTGEAR_STATE_OFF, False);
```

三番目の引数は、XmNxrtGearStateChangedCallback リストのコールバックが呼び出されたかどうかを示す Boolean です。

3.4 ボタンの色の設定

Enhanced Toggle では、インディケータの背景の色を設定することができます。トグルがセットされた時(XmNxrtGearState が XRTGEAR_STATE_ON)の背景色をセットするには、XmNxrtGearSelectColor を設定します。[1] トグルがセットされていない時(XmNxrtGearState が XRTGEAR_STATE_ON 以外)の背景色をセットするには、XmNxrtGearUnselectColor を設定します。XmNxrtGearSelectColor と XmNxrtGearUnselectColor には、Pixel 型の値が必要です。

Enhanced Toggle は XmPushButton、Primitive、Core の色のリソースを継承します。

3.5 リソースの一覧

Enhanced Toggle のリソースと、よく使用される継承されたリソースの一覧です。リソース名の後は、データ型、デフォルト値、そしてリソースと共に使用される手続きです。**C** は、そのリソースが create(例: *XtCreateWidget()*)で定義でき、**S** は set(例; *XtSetValues()*)でき、**G** は get(例: *XtGetValues()*)で使用できることを意味します。

[1] XmNxrtGearSelectColor リソースは、XmNselectColor リソースと同等です。

XmNarmCallback XtCallbackList dynamic CSG

ボタンが押された時に呼び出されるコールバックのリストを指定します。各ルーチンには、XmToggleButtonCallbackStrucnt のポインタが渡されます。コールバックは、*XtAddCallback()*を使用してリストに追加して下さい。コールバックは次のように構成されます。

```
void
CallbackProcedure(
  Widget  widget,
  XtPointer   client_data,
  XtPointer   call_data
    /* points to an XmToggleButtonCallbackStruct */
)
```

XmToggleButtonCallbackStruct は次のように定義されます。

```
typedef struct {
  int     reason; // Read-only
  XEvent  *event; // Read-only
  int     set;
} XmToggleButtonCallbackStruct;
```

reason は XmCR_ARM。*event* は、コールバックを呼び出したイベント構造体を指します。*set* は True(ボタンが押されている)または False(ボタンは押されていない)のどちらかです。

XmNbackground Pixel dynamic CSG

ボタンの背景色を指定します。

XmNdisarmCallback XtCallbackList dynamic CSG

ボタンが離された時に呼び出されるコールバックのリストを指定します。各ルーチンには、XmToggleButtonCallbackStruct のポインタが渡されます。コールバックは、*XtAddCallback()*を使用してリストに追加してください。コールバックは次のように構成されます。

```
void
CallbackProcedure(
  Widgetwidget,
  XtPointer client_data,
  XtPointer call_data
    /* points to an XmToggleButtonCallbackStruct */
)
```

XmToggleButtonCallbackStruct は次のように定義されます。

```
typedef struct {
int     reason; // Read-only
XEvent  *event; // Read-only
int     set;
} XmToggleButtonCallbackStruct;
```

reason は XmCR_DISARM。*event* は、コールバックを呼び出したイベント構造体を指します。*set* は True(ボタンが押されている)または False(ボタンは押されていない)のどちらかです。

XmNfontList XmFontList dynamic CSG

ウィジェットのテキストに使用されるフォント・リストを指定します。

XmNforeground Pixel dynamic CSG

ボタンの前景色を指定します。

XmNheight Dimension dynamic CSG

ウィンドウの高さを pixel で指定します。

XmNxrtGearIndicator XrtGearIndicator XRTGEAR_INDICATOR_FILL CSG

ボタンのインディケータのスタイルを指定します。有効な値は、次の通りです。

XRTGEAR_INDICATOR_NONE	インディケータは表示されない
XRTGEAR_INDICATOR_FILL	インディケータが表示され、影で状態が示される
XRTGEAR_INDICATOR_CHECK	チェックマークが前景色で描画される
XRTGEAR_INDICATOR_CHECK_BOX	箱の中にチェックマークが前景色で描画される
XRTGEAR_INDICATOR_CIRCLE	選択されている色で丸ボタンが表示され、影で状態が示される
XRTGEAR_INDICATOR_CIRCLE_BOX	箱の中に選択されている色で丸ボタンが表示され、影で状態が表示される
XRTGEAR_INDICATOR_CROSS	前景色でクロス(X 印)が表示される
XRTGEAR_INDICATOR_CROSS_BOX	箱の中に前景色でクロスが表示される
XRTGEAR_INDICATOR_PIXMAP	XmNxrtGearIndicatorPixmapList で指定された Pixmap

XmNxrtGearState が XRTGEAR_STATE_INDETERMINATE またはそれ以上で、XmNxrtGearIndiator が XRTGEAR_INDICATOR_PIXMAP または XRTGEAR_INDICATOR_PIXMAP_NONE でなければ、インディケータは点描画で表示されます。XmNxrtGearStage が XRTGEAR_STATE_INDETERMINATE またはそれ以上で、XmNxrtGearIndicator が XRTGEAR_INDICATOR_PIXMAP_NONE であれば、ラベルとボタンは、XmNxrtGearSelectColor と XmNxrtGearUnselectColor の色を使用して点描画で表示されます。

XmNxrtGearIndicatorMaskList XrtGearIndicatorMaskList NULL CSG

XmNxrtGearIndicatorPixmapList で指定されるインディケータ pixmap のリストと一緒に使用される、clip pixmap(clip マスク)のリストを提供します。各 clip pixmap は対応するインディケータ pixmap をコントロールします。pixmap が XmUNSPECIFIED_PIXMAP であれば、対応するインディケータ pixmap は切り取られません。各 clip pixmap は bit depth=1 としてください。clip pixmap とそれに対応するインディケータ pixmap は、同じ XPM ファイルを共有することができます。

XmNxrtGearIndicatorPixmapList XrtGearIndicatorPixmapList NULL CSG

XmNxrtGearIndicator が XRTGEAR_INDICATOR_PIXMAP の時に使用されるインディケータ pixmap のリストを提供します。このリソースが設定されていないと、インディケータは表示されません。リスト内の pixmap の位置は、XmNxrtGearState 値に指定されます。pixmap は、各 XmNxrtGearState 値に対して必要です。pixmap が XmUNSPECIFIED_PIXMAP であれば、リストの最後の pixmap が点描画で表示されます。

XmNlabelString XmString dynamic CSG

ラベルのストリングを指定します。

XmNmarginHeight Dimension 2 CSG

ボタンの上部と下部の端とその影との間の距離を指定します。

XmNmarginWidth Dimension 2 CSG

ボタンの左と右の端とその影との間の距離を指定します。

XmNxrtGearNumStates int 2 CSG

セットできる状態の最大数を指定します。デフォルトでは、ボタンは XRTGEAR_STATE_OFF と XRTGEAR_STATE_ON のどちらかにセットできます。XmNradioBehavior が XmONE_OF_MANY に設定されている XmRowColumn 内にトグルがあれば、このリソースの値は強制的に 2 になります。

XmNxrtGearSelectColor Pixel dynamic CSG

トグルがセットされた時のインディケータの背景色をセットする色を指定します。XmNselectColor と同じ役割を持ちます。

XmNxrtGearState int dynamic CSG

ボタンの現在の状態を指定します。次の enum が提供されています。XRTGEAR_STATE_OFF、XRTGEAR_STATE_ON、XRTGEAR_STATE_INDETERMINATE。

状態の数は XmNxrtGearNumStates で指定できます。エンドユーザがボタンをクリックすると XmNxrtGearState の値が増加し、最大値に達すると 0(XRTGEAR_STATE_OFF)からまた開始します。

XmNxrtGearStateChangedCallback XtCallbackList dynamic CSG

トグル・ボタンの状態が変更される時に呼び出されるコールバックのリストを指定します。各ルーチンには、XrtGearStateChangedCallbackStruct のポインタが渡されます。コールバックは、*XtAddCallback()*を使用してリストに追加してください。コールバックは次のように構成されます。

```
void
CallbackProcedure(
  Widget  widget,
  XtPointer   client_data,
  XtPointer   call_data
    /* points to an XrtGearStateChangedCallbackStruct */
)
```

XrtGearStateChangedCallbackStruct は次のように定義されます。

```
typedef struct {
  int     reason; // Read-only
  XEvent  *event; // Read-only
  int     state;
} XrtGearStateChangedCallbackStruct;
```

reason は XRTGEAR_REASON_STATE_BEGIN(状態が変化する前) または XRTGEAR_REASON_STATE_END(状態が変化した後)のどちらかです。*event* は、コールバックを呼び出したイベント構造体を指します。*state* はトグル・ボタンの現在の状態です。

XmNxrtGearUnselectColor Pixel dynamic CSG

トグルがセットされていない(XmNxrtGearState が XRTGEAR_STATE_ON でない)時のインディケータの背景色を指定します。カラーで表示される場合、デフォルトの色は背景色と

上部の影の色で、モノクロの場合のデフォルトは背景色です。

XmNuserData　XtPointer　NULL　CSG

アプリケーションで定義されているデータのポインタを指定します。

XmNwidth　Dimension　dynamic　CSG

ウィンドウの幅を pixel で指定します。

XmNx　Position　0　CSG

親ウィジェットの内側の左上角に対応する、ウィジェットの外側の左上角の X 座標。

XmNy　Position　0　CSG

親ウィジェットの内側の左上角に対応する、ウィジェットの外側の左上角の Y 座標。

3.5.1 Enhanced Toggle によって継承されるリソース

Resource	Inherited From	Resource	Inherited From
XmNaccelerator	XmLabel	XmNlabelString	XmLabel
XmNaccelerators	Core	XmNlabelType	XmLabel
XmNalignment	XmLabel	XmNmappedWhenManaged	Core
XmNancestorSensitive	Core	XmNmarginBottom	XmLabel
XmNarmCallback	XmToggleButton	XmNmarginHeight	XmLabel
XmNbackground	Core	XmNmarginLeft	XmLabel
XmNbackgroundPixmap	Core	XmNmarginRight	XmLabel
XmNborderColor	Core	XmNmarginTop	XmLabel
XmNborderPixmap	Core	XmNmarginWidth	XmLabel
XmNborderWidth	Core	XmNmnemonic	XmLabel
XmNbottomShadowColor	Primitive	XmNmnemonicCharSet	XmLabel
XmNbottomShadowPixmap	Primitive	XmNnavigationType	Primitive
XmNcolormap	Core	XmNrecomputeSize	XmLabel
XmNdepth	Core	XmNscreen	Core
XmNdestroyCallback	Core	XmNsensitive	Core
XmNdisarmCallback	XmToggleButton	XmNspacing	XmToggleButton
XmNfontList	XmLabel	XmNstringDirection	XmLabel
XmNforeground	Primitive	XmNtopShadowColor	Primitive

Resource	Inherited From
XmNheight	Core
XmNhelpCallback	Primitive
XmNhighlightColor	Primitive
XmNhighlightOnEnter	Primitive
XmNhighlightPixmap	Primitive
XmNindicatorOn	XmToggleButton
XmNindicatorSize	XmToggleButton
XmNindicatorType	XmToggleButton
XmNinitialResourcesPersistent	Core
XmNlabelPixmap	XmLabel

Resource	Inherited From
XmNtopShadowPixmap	Primitive
XmNtranslations	Core
XmNtraversalOn	Primitive
XmNuserData	Primitive
XmNvalueChangedCallback	XmToggleButton
XmNvisibleWhenOff	XmToggleButton
XmNwidth	Core
XmNx	Core
XmNy	Core

3.6 手続きと方法

Enhanced Toggle の手続きと方法の一覧です。

XmCreateXrtToggleButton()

Enhanced Toggle ウィジェットを作成します。

```
Widget
XmCreateXrtToggleButton(
  Widget  parent,
  String  name,
  Arg     *arglist,
  int     argcount
  )
```

parent は親。*name* は作成されるウィジェット名。*arglist* は *argcount* 引数のリストです。

XmIsXrtToggleButton()

ウィジェットが Enhanced Toggle であれば `True` を返します。

```
Boolean
XmIsXrtToggleButton(
  Widgetwidget
  )
```

XrtGearMrmInitialize()

UIL アプリケーションに使用するために、Mrm に XRT/gear ウィジェットを登録します。この手続きは、*MrmInitialize()*の後に呼び出して下さい。

```
void
XrtGearMrmInitialize(void)
```

XrtGearSetXmStringConverter()

XmRXmString リソース変換プログラムをセットし、XmString 値を作成します。

```
void
XrtGearSetXmStringConverter(void)
```

変換は、エンベッドされたフォント(タグ)を指定するために、次のシンタックスを使用します。

@font@	フォントのタグは `XmNfontList`
@P@	前のフォント
\@	"@"とする

例:

```
For a @bold@free evaluation@P@ of any product
```

フォント･リストを設定

```
fixed,-*-times-bold-r-normal-*-140-*=bold
```

表示結果

For a **free evaluation** of any product

XrtGearToggleButtonGetState()

Enhanced Toggle ウィジェットの現在の状態を取得します。

```
int
XrtGearToggleButtonGetState(
  Widget  widget
  )
```

widget は Enhanced Toggle ウィジェットです。

XrtGearToggleButtonSetState()

Enhanced Toggle ウィジェットの現在の状態を設定します。

```
int
XrtGearToggleButtonGetState(
  Widgetwidget,
  int   state,
  Boolean   notify
  )
```

widget は Enhanced Toggle ウィジェットです。*state* はウィジェットの新しい状態。*notify* は `XmNxrtGearStateChangedCallback` リストのコールバックを呼び出すかどうかを示します。

4

Tabbed Dialog

タブ･ダイアログとは ■ サンプル･プログラム
ウィジェットの概要 ■ アクティブな tab の変更
page を Tab Button に関連させる ■ Tab 表示のコントロール
constraint リソース ■ Tab Button リソース
角のサイズの指定 ■ リサイズのコントロール
リソースの一覧 ■ 手続きと方法
トランスレーションとアクション

4.1 タブ･ダイアログとは

タブ･ダイアログをプログラミングするには、下記を作成してください。

- tab の表示と配置をコントロールする Tab Manager ウィジェット
- ダイアログを表示する Tab Button ウィジェット
- タブ･ダイアログに記憶される page ウィジェット

Tab Manager と Tab Button を組み合わせると、並べられたファイル･フォルダのようなフォーマットが作成できます。page ウィジェットは Tab Manager の子として定義され、Tab Button との関連付けには constraint リソースが使用されます。page ウィジェットは、(Manager ウィジェット･クラスも含む)どのクラスにもなることができます。

図 12 は、一般的なタブ･ダイアログの例です。

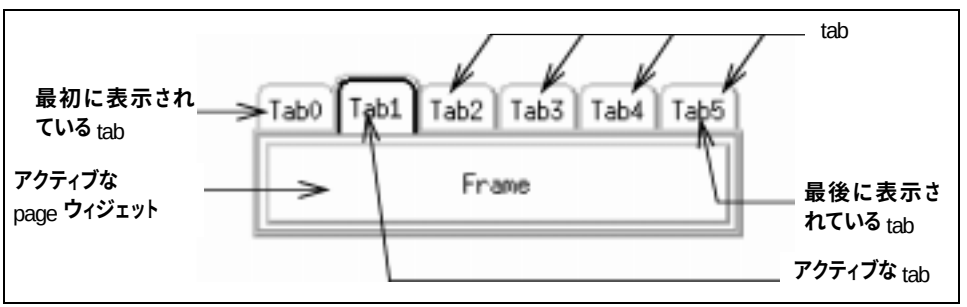


図 12　タブ･ダイアログの構成

4.2 サンプル･プログラム

次のコードはTab Managerウィジェットを作成し、Tab Buttonとpageウィジェットを子として定義しています。

```
#include <X11/Intrinsic.h>
#include <Xm/Xm.h>
#include <Xm/XrtGLabel.h>
#include <Xm/Frame.h>
#include <Xm/XrtTabB.h>
#include <Xm/XrtTabM.h>
typedef struct my_widgets {
  Position offset;
  String name;
  Widget w;
} MyWidgets;
MyWidgets tab_table[] = {
   10, "Memos", NULL,
   60, "Faxes", NULL,
   260, "Postponed", NULL,
};
Widget page_table[3];
int
main(int argc, char **argv)
{
  XtAppContext app_context;
  Widget toplevel, manager, page;
  int i;
  XmString xms;
  toplevel = XtVaAppInitialize(
    &app_context, "Button",
    NULL, 0, (Cardinal *)&argc, argv, NULL);
  manager = XtVaCreateWidget( "tab_manager",
    xmXrtTabManagerWidgetClass, toplevel, NULL);
  xms = XmStringCreateSimple("Page widget");
  for (i = 0; i < XtNumber(tab_table); i++) {
    tab_table[i].w = XtVaCreateManagedWidget(tab_table[i].name,
      xmXrtTabButtonWidgetClass, manager,
      XmNxrtGearChildType, XRTGEAR_CHILD_TYPE_TAB,
      NULL);
    page_table[i] = XtVaCreateManagedWidget(tab_table[i].name,
      xmFrameWidgetClass, manager,
      XmNxrtGearChildType, XRTGEAR_CHILD_TYPE_PAGE,
      NULL);
    XtVaSetValues( tab_table[i].w,
      XmNxrtGearPageWidget, page_table[i],
      NULL);
    XtVaCreateManagedWidget("pb1",
      xmXrtGLabelWidgetClass, page_table[i],
      XmNlabelString, xms,
      NULL);
  }
```

```
    XmStringFree(xms);
    XtManageChild(manager);
    XtRealizeWidget(toplevel);
    XtAppMainLoop(app_context);
  }
```

このプログラムがコンパイル、実行されると、次のような tab が表示されます。



図 13　Tab Manager と Tab Button

このアプリケーションは、一つの Tab Manager とその子として定義される三つの Tab Button で構成されています。各 Tab Button には page が関連付けられています。各 page には、その子としてラベル･ウィジェットが定義されています。実際のアプリケーションでは、page の子として container ウィジェットがさらに定義されることもあります。

複数のレベルを持つファイル･フォルダを作成するには、page ウィジェットとして別の Tab Manager を作成してください。

4.3　ウィジェットの概要

Tab Button	Include File:	*<Xm/XrtTabB.h>*
	Class Name:	XmXrtTabButton
	Class Hierarchy:	Core → XmPrimitive → XmLabel → XmPushButton → XmXrtTabButton
	Class Pointer:	xmXrtTabButtonWidgetClass
Tab Manager	Include File:	*<Xm/XrtTabM.h>*
	Class Name:	XmXrtTabManager
	Class Hierarchy:	Core → Composite → Constraint → XmManager → XmXrtTabManager
	Class Pointer:	xmXrtTabManagerWidgetClass

Tab Manager は XmManager のサブクラスで、Tab Button は XmPushButton のサブクラスです。

4.4 アクティブな tab の変更

Tab Manager には、Tab Manager とその子の表示をコントロールするリソースが三つ提供されています。

- 現在アクティブな page ウィジェットを含む XmNxrtGearActivePageWidget
- アクティブな page 番号を含む XmNxrtGearActivePageNumber
- 現在アクティブな tab ウィジェットを含む XmNxrtGearActiveTabWidget

page 変更によるコールバック

XmNxrtGearPageCallback は、XmNxrtGearPageNumber の値がインタラクションまたはプログラムによってセットされた時に開始されるコールバックを定義します。[1] このリソースによって page の表示をコントロールできます。

各 XmNxrtGearPageCallback リソースには、次の構造体が渡されます。

```
typedef struct {
  int         reason;          // Read-only
  XEvent      *event;          // Read-only
  XrtGearAction   action;          // Read-only
  int         prev_page_number;    // Read-only
  Widget      prev_page_widget;    // Read-only
  Widget      prev_tab;        // Read-only
  Widget      tab;
  int         page_number;
  Widget      page_widget;
  Boolean     doit;
} XrtGearPageCallbackStruct;
```

各コールバックは、page 変更が起こる前と後に呼び出されます。*reason* どのイベントが起こっているかを示し、XRTGEAR_REASON_PAGE_BEGIN(page 変更前) と XRTGEAR_REASON_PAGE_END(page 変更後)のどちらかです。

prev_page_widget と *prev_page_number* メンバによって、以前に表示されていた page ウィジェットとその番号にアクセスすることができます。以前に表示されていたページに関連する Tab Button は *prev_tab* で指定できます。Tab Manager が最初に表示される時、page 変更のコールバックが開始され、*prev_page_number* は-1 になります。

新しい page ウィジェットとその番号は *page_number* と *page_widget* で指定できます。これらのメンバは修正可能ですので、Tab Manager で page 変更をコントロールできます。*tab* は、新しい page ウィジェットに関連している Tab Button です。

doit フィールドを利用して、page 変更を制限することができます。*reason* が XRTGEAR_REASON_PAGE_BEGIN の時に *doit* が False にセットされれば、page は変更されません。

---

[1] XmNxrtGearPageNumber は、XmNxrtGearActivePageWidget と XmNxrtGearActiveTabWidget がセットされると、変更されます。

次の例は、エンドユーザが最初の page から離れた後はそのページに戻れないようにするコールバックの例です。

```
void
pageCB(Widget w, XtPointer client_data,
XrtGearPageCallbackStruct *call_data)
{
    if (call_data->reason == XRTGEAR_REASON_PAGE_BEGIN &&
       call_data->page_number == 0 &&
       call_data->prev_page_number != -1) {
      call_data->doit = False;
    }
}
```

4.5 page を Tab Button に関連させる

デフォルトでは、Tab Button が作成され page が明示的に指定されていなければ、Tab Manager が tab を適当なページに振り分けます。これは、プログラムの最初に全ての Tab Button と page を作成する時に便利です。

但し、アプリケーションが page と Tab Button をダイナミックに作成、破棄していく場合は、Tab Manager が自動的に Tab Button を page に関連されない方が便利です。アプリケーションによって、Tab Button と page の関係を明示的に指定する時は、XmNxrtGearPageAttachPolicy を XRTGEAR_ATTACH_NONE にセットしてください。

XmNxrtGearPageAttachPolicy のデフォルト値は XRTGEAR_ATTACH_ALWAYS で、必要時に自動的に Tab Button と page を振り分けます。

XmNxrtGearAttachPagePolicy が、Tab Manager が作成された後にセットされた場合は、次に Tab Button または page が作成、破棄されるまで、その設定は有効になりません。

下記の、(XmNxrtGearPageCallback で指定されるリストに追加された)コールバックでは、ページをダイナミックに作成しています。

```
void pageChange(Widget w, XtPointer client_data,
  XtPointer call_data)
{
  XrtGearPageCallbackStruct *cbstruct =
      (XrtGearPageCallbackStruct *) call_data;
  int     page_number;
  char    buff[80];
  Widget  page;
  if (cbstruct->reason == XRTGEAR_REASON_PAGE_END) {
    return;
  }
  if (cbstruct->page_widget == NULL) {
    XtVaGetValues(cbstruct->tab,
      XmNuserData, &page_number,
      NULL);
    sprintf( buff, "Page%02d", page_number);
    page = XtVaCreateManagedWidget(buff,
      xmLabelWidgetClass, w,
```

```
        XmNxrtGearChildType, XRTGEAR_CHILD_TYPE_PAGE,
        XmNxrtGearPageNumber, page_number,
        NULL);
      XtVaSetValues(cbstruct->tab,
        XmNxrtGearPageWidget, page,
        NULL);
      cbstruct->page_widget = page;
    }
  }
```

コールバック構造体の *page_widget* エレメントをセットすることにより、Tab Manager に新しい page を表示させることができます。

4.6 Tab 表示のコントロール

Tab Manager では、次のような表示をコントロールすることができます。

- tab が表示される位置
- どの tab を visible(表示)にするか
- tab 間のスペース
- tab の回転
- Tab Manager のハイライトの幅
- Tab Manager のマージン
- Tab のストレッチ

page arrow button(矢印ボタン)

Tab の位置

`XmNxrtGearTabSide` は、tab が表示される位置を指定します。このリソースは、次の値にセットできます。

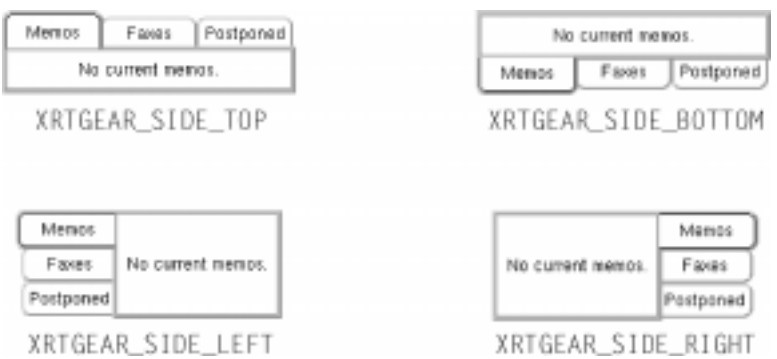


図 14　tab の位置

デフォルト値は、`XRTGEAR_SIDE_TOP` です。

Tab の visibility(表示)

XmNxrtGearFirstVisibleTab は、現在(一番上または左に)表示されている最初の tab を設定します。このリソースがセットされると、tab は必要に応じてスクロールします。

XmNxrtGearLastVisibleTab は、現在(一番下または右に)表示されている最後の tab を設定します。このリソースがセットされると、tab は必要に応じてスクロールされます。

特定な tab が表示されるようにするには、*XrtGearTabManagerMakeTabVisitble()*を呼び出してください。次の例では、二番目の tab 表示されるようにしています。(0 は最初の tab、1 はその次の tab のインデックス、以下同様に続きます。)

```
Widget tabmgr;
Boolean ok;

ok = XrtGearTabManagerMakeTabVisible(tabmgr, 1);
```

指定された tab が表示領域の左側にあれば、スクロール後、一番左に表示されます。指定された tab が表示領域の右側にあれば、スクロール後、一番右に表示されます。

*XrtGearTabManagerMakeTabVisible()*は、tab を表示する操作が成功したかどうかを示す Boolean 値を返します。(例: False であれば、tab インデックスは無効)

Tab のスペース

tab 間の(pixel で測定される)スペースは、XmNxrtGearTabSpacing によって指定できます。



図 15　tab のスペース

このリソースが負の値にセットされると、tab は重なり合います。

Tab のリサイズ

XmNxrtGearTabResize は、全ての tab がその行にある一番大きい tab の幅/高さと同じサイズにリサイズするかどうかを指定します。XmNxrtGearTabResize が True(デフォルト値)であれば、リサイズが実行されますが、False であれば tab はリサイズされません。XmNxrtGearTabSide が XRTGEAR_SIDE_TOP または XRTGEAR_SIDE_BOTTOM であれば、幅が使用され、そうでなければ高さが使用されます。



図 16　tab のリサイズ

tab が全て同じ大きさで、表示領域一杯に表示されるようにするには、XmNxrtGearTabResize と XmNxrtGearTabStretch を True にしてください。

Tab の回転

XmNxrtGearTabRotation は tab の回転を指定します。このリソースは、次の値にセットできます。



図 17　tab の回転

または、Tab Button のリソース、XmNxrtGearPixmapRotation、XmNxrtGearStringRotation、XmNxrtGearStringLayout を利用して、各 Tab Button の表示をコントロールすることができます。詳細は 4.8 を参照してください。

Tab Manager ハイライトの幅

XmNxrtGearHighlightThickness は、Tab Manager のハイライトの幅を指定します。デフォルトは 2 pixel 幅です。

Tab Manger のマージン

XmNxrtGearMarginWidth は、Tab Manager の左と右のマージンを決定します。同様にして、XmNxrtGearMarginHeight は上部と下部のマージンを決定します。



図 18　Tab Manager とマージンの定義

どちらのリソースも、デフォルト値は 2 pixel です。

Tab のストレッチ

XmNxrtGearTabStretch は、全ての tab が表示可能領域一杯に表示されるようにリサイズするかどうかを指定します。



図 19　tab のストレッチ

tab が全て同じ大きさで、表示領域一杯に表示されるようにするには、XmNxrtGearTabResize と XmNxrtGearTabStretch を True にしてください。

Page Arrow Button(矢印ボタン)の表示

デフォルトでは、tab ボタンが全て表示されない時に、矢印ボタンが表示されます。この矢印ボタンの位置は、図 20 のように XmNxrtGearTabSide の値に依存します。[1]

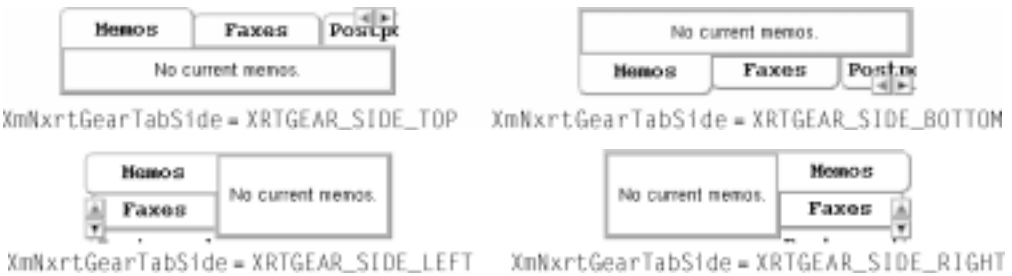


図 20　page 矢印ボタン

エンドユーザは、この矢印ボタンを押して、tab ボタンの表示を前または後ろ方向にスクロールすることができます。

XmNxrtGearArrowDisplayPolicy は、矢印ボタンを表示するかどうか指定します。有効な値は、次の通りです。

XRTGEAR_DISPLAY_AS_NEEDED	必要時に矢印ボタンを表示する(デフォルト)
XRTGEAR_DISPLAY_ALWAYS	矢印ボタンを常に表示する
XRTGEAR_DISPLAY_NEVER	矢印ボタンを表示しない

矢印ボタンの高さと幅 (pixel) は、XmNxrtGearArrowHeight と XmNxrtGearArrowWidth リソースで設定します。デフォルトの高さと幅は次のように決定されます。

[1] XmNstringDirection が XmSTRING_DIRECTION_R_TO_L にセットされている場合は、矢印ボタンは、反対側に表示されます。

- どちらのリソースも設定されていない場合、幅高さ共に tab ボタンの半分のサイズにセットされる
- XmNxrtGearArrowHeight がセットされ XmNxrtGearArrowWidth がセットされていない場合は、XmNxrtGearArrowHeight の値が幅にも適用される
- XmNxrtGearArrowWidth がセットされ XmNxrtGearArrowHeight がセットされていない場合は、XmNxrtGearArrowWidth の値が高さにも適用される

矢印ボタンの高さは、tab ボタンの矢印より高く設定することはできません。

4.7 constraint リソース

Tab Manager の子ウィジェットには、次の constraint リソースが定義されます。

- XmNxrtGearChildType は、子が page であるか tab であるか示します。子が Tab Button であれば、tab として使用され、XmNxrtGearChildType 値は XRTGEAR_CHILD_TYPE_TAB になります。他のウィジェットは全て page とみなされ、XmNxrtGearChildType 値は XRTGEAR_CHILD_TYPE_PAGE になります。[1] (このバージョンでは、Tab Button のみが tab としてサポートされています。)
- XmNxrtGearPageNumber は、page 子ウィジェットの page 番号を指定します。エンドユーザによってこの数がセットされていない場合は、page が作成された順にこのデフォルト値が計算されます。(最初の page は番号 0 です。)

  子 tab ウィジェットの場合は、XmNxrtGearPageNumber がその tab に関連付けられている page の番号を指定します。XmNxrtGearPageNumber が-1 であれば、XmNxrtGearPageWidget によって指定されている page ウィジェットが使用されます。
- XmNxrtGearPageWidget は、Tab Button ウィジェットに関連付けられている page を指定します。複数の tab は一つの page ウィジェットを共有することができます。この constraint リソースは、Tab Button ウィジェットにしか使用できません。

4.8 Tab Button リソース

ここでは、Tab Button ウィジェットの使用法を解説します。

4.8.1 Button のタイプ

XmNxrtGearLabelType リソースは、Tab Button に使用されるラベルのタイプを設定します。このリソースは次の値のどれかに設定できます。

XRTGEAR_TYPE_STRING	ラベルのタイプは XmString 、XmNlabelString が表示される

[1] private で internal な子ウィジェットの場合、XmNxrtGearChildType は XRTGEAR_CHILD_TYPE_INTERNAL です。この値は、アプリケーションではセットできません。

XRTGEAR_TYPE_PIXMAP	ラベルのタイプは pixmap 、XmNxrtGearLabelPixmap が表示される
XRTGEAR_TYPE_STRING_PIXMAP	ラベルのタイプは XmString と pixmap 両方、XmNlabelString と XmNxrtGearLabelPixmap が表示される

XmNxrtGearLabelType のデフォルト値は XRTGEAR_TYPE_STRING です。

4.8.2 Tab のアンカー

Tab Button が Tab Manager の子として指定されている場合、その配置は Tab Manager の XmNxrtGearTabSide リソースによってコントロールされます。Tab Button が他のウィジェットの子として定義されている場合は、親ウィジェットに対する配置は、XmNxrtGearAnchorSide によって指定されます。このリソースは次の値にセットできます。

XRTGEAR_ANCHOR_SIDE_TOP	Tab Button の上部が親ウィジェットに付く (デフォルト)
XRTGEAR_ANCHOR_SIDE_BOTTOM	Tab Button の下部が親ウィジェットに付く
XRTGEAR_ANCHOR_SIDE_LEFT	Tab Button の左端が親ウィジェットに付く
XRTGEAR_ANCHOR_SIDE_RIGHT	Tab Button の右端が親ウィジェットに付く

4.8.3 pixmap の設定

Tab Button ウィジェットに pixmap を指定するには、XmNxrtGearLabelPixmap をセットします。このリソースは Pixmap 型です。

XmNxrtGearLabelPixmap で指定される pixmap は、XmNxrtGearLabelType が XRTGEAR_TYPE_PIXMAP または XRTGEAR_TYPE_STRING_PIAMAP の時に表示されます。

XPM と XBM の pixmap フォーマットがサポートされています。

Clip マスクの使用

XmNxrtGearMask は、Tab Button ウィジェットに表示されている pixmap の clip マスクを定義します。この clip マスクは、ラベルに pixmap をどれだけ表示するかどうかをコントロールします。マスクの 0 以外の部分が実際に描画されます。次のコードは、clip マスクを使用する Tab Button ウィジェットを作成します。

```
Pixmap image, mask;
Widget button;

button = XtVaCreateManagedWidget("EnhLabel",
  xmXrtTabButtonWidgetClass,   tabmanager,
  XmNxrtGearChildType,     XRTGEAR_CHILD_TYPE_TAB,
  XmNxrtGearLabelType,     XRTGEAR_TYPE_PIXMAP,
  XmNxrtGearLabelPixmap,   image,
  XmNxrtGearMask,      mask,
  NULL);
```

XmNxrtGearMask には、Pixmap 型の値が必要です。このリソースのデフォルト値は XmUNSPECIFIED_PIXMAP です。clip マスクはモノクロ(bit depth=1)です。pixmap と clip マスクの指定には同じファイルが使用できます。

4.8.4 Button の形

Tab Button の形は、次の二つのリソースでコントロールします。

- ボタンのスタイルを指定する XmNxrtGearStyle
- ボタンの角のサイズを指定する XmNxrtGearCornerSize

XmNxrtGearStyle は次の値にセットできます。

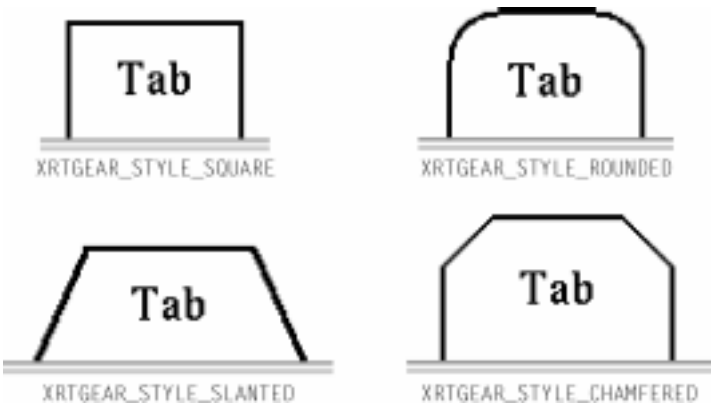


図 21　Tab Button の形

デフォルトは XRTGEAR_STYLE_SQUARE です。XmNxrtGearStyle が XRTGEAR_STYLE_CHAMFERED、角は面取りされた角になります。

図 22 は、各スタイルと XmNxrtGearCornerSize の組み合わせ例です。



図 22　角のサイズ

XmNxrtGearStyle が XRTGEAR_STYLE_SQUARE の場合は、XmNxrtGearCornerSize は意味を持ちません。

4.8.5 テキストと pixmap の配置

XmNxrtGearLabelType が XRTGEAR_TYPE_STRING_PIXMAP にセットされている時、テキスト･ストリングと pixmap の両方が Tab Button に表示されます。テキストと pixmap の配置をコントロールするには、次の二つのリソースがあります。

- テキスト・ストリングと pixmap の間の距離を pixel で指定する XmNxrtGearLayoutSpacing(デフォルト値は 2)
- テキスト・ストリングと pixmap の相対的な位置を指定する XmNxrtGearStringLayout

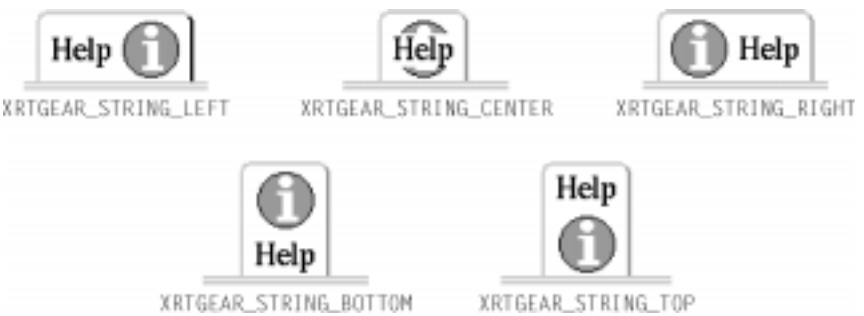


図 23　Tab Button のテキストと pixmap の配置

デフォルトは XRTGEAR_STRING_RIGHT です。

XmNxrtGearLayoutSpacing と XmNxrtGearStringLayout は、XmNxrtGearLabelType が XRTGEAR_TYPE_STRING_PIXMAP にセットされていなければ無視されます。

Tab Button が、XmNxrtGearTabRotation が XRTGEAR_ROTATE_UNSPECIFIED 以外に設定されている Tab Manager の子であると、XmNxrtGearStringLayout は無視されます。

マージンのコントロール

(XmLabel から継承される)XmNmarginLeft、XmNmarginRight、XmNmarginTop、XmNmarginBottom、XmNmarginHeight、XmNmarginWidth のリソースは、Tab Button の端とストリング/pixmap の間の距離の計算に使用されます。



図 24　Tab Button のマージンのコントロール

4.8.6　テキストと pixmap の回転

Tab Button のテキストと pixmap はそれぞれ別々に回転できます。Tab Button のテキストを回転するには XmNxrtGearStringRotation をセットします。このリソースは図 25 の値にセットできます。

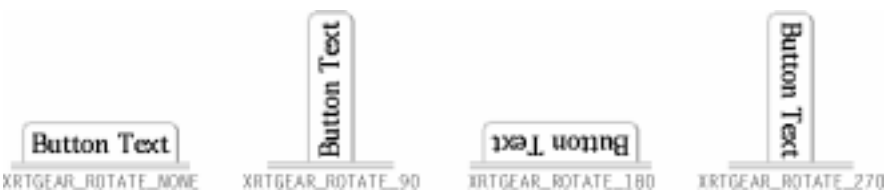


図 25　Tab Button のテキストの回転

デフォルトは XRTGEAR_ROTATE_NONE です。回転は反時計方向に行われます。

Tab Button が、XmNxrtGearTabRotation が XRTGEAR_ROTATE_UNSPECIFIED 以外に設定されている Tab Manager の子であると、XmNxrtGearStringRotation は無視されます。

Tab Button の pixmap を回転するには、XmNxrtGearPixmapRotation をセットします。このリソースは、XmNxrtGearStringRotation と同じ値にセットすることができます。



図 26　Tab Button の pixmap の回転

XmNxrtGearPixmapRotation のデフォルトは XRTGEAR_ROTATE_NONE です。

Tab Button が、XmNxrtGearTabRotation がセットされている Tab Manager の子であると、XmNxrtGearPixmapRotation は無視されます。

4.8.7　Tab Button と Page の影の色

各 Tab Button は、それと関連している(XmNxrtGearPageWidget で指定される)page ウィジェットから色の指定を取得します。Tab Button に独自の色を指定するには、Tab Button と page ウィジェットの関連付けを終えた後に、該当する色のリソース(XmNforeground 等)を変更してください。

各 page ウィジェットの影の色は、Tab Button の色の指定から取得されます。

4.9　角のサイズの指定

XmNxrtGearStyle が XRTGEAR_STYLE_SQUARE 以外の値にセットされると、Tab Button の角の大きさや形は、Tab Manager の XmNxrtGearCornerSize によって指定されます。XmNxrtGearStyle が XRTGEAR_STYLE_CHAMFERED または XRTGEAR_STYLE_ROUNDED であれば、XmNxrtGearCornerSize は角の幅と高さを指定します。

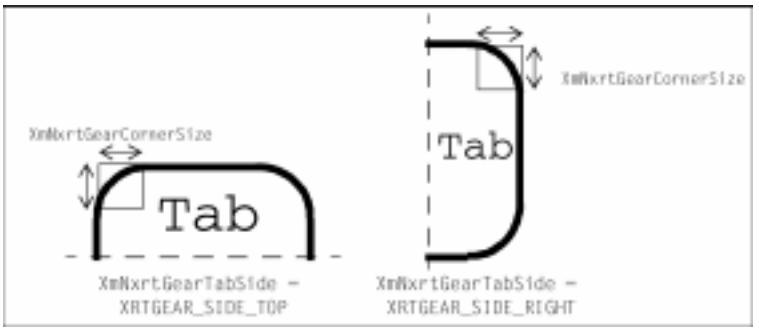


図 27　丸角のサイズ

tab の角が斜め(XmNxrtGearSytle が XRTGEAR_STYLE_SLANTED)であれば、XmNxrtGearCornerSize がその傾きを指定します。

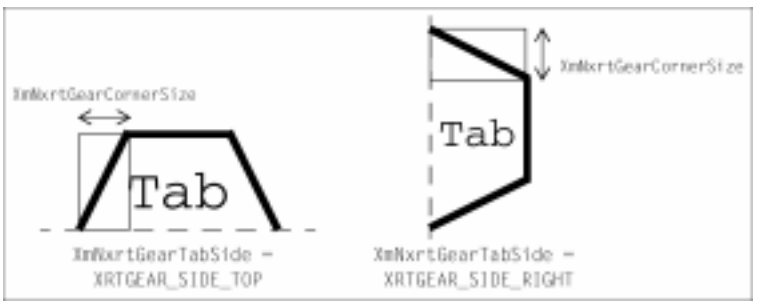


図 28　斜め角のサイズ

XmNxrtGearCornerSize が tab の幅/高さより大きい値にセットされると、0 が適用されます。

XmNxrtGearCornerSize は、Tab Button が Tab Manager の子でない時は、Tab Button のリソースとしても定義されます。Tab Button が Tab Manager の子である時は、Tab Button の XmNxrtGearCornerSize の値は、Tab Manager が manage された後にセットされた場合にのみ有効になります。

4.10　リサイズのコントロール

Tab Manager の XmNxrtGearResizeCallback は、Tab Manager がリサイズされた時に開始されるコールバックを提供します。

各 Resize コールバックには、次の構造体が渡されます。

```
typedef struct {
  int     reason; /* read-only */
  XEvent  *event; /* read-only */
  Dimension   width;
  Dimension   height;
} XrtGearResizeCallbackStruct;
```

構造体の *width* と *height* メンバは、Tab Manager の新しい幅と高さを指定します。

4.11 リソースの一覧

Tab Button と Tab Manager のリソースと、よく使用される継承されたリソースの一覧です。リソース名の後は、データ型、デフォルト値、そしてリソースと共に使用される手続きです。**C** は、そのリソースが create(例: *XtCreateWidget()*)で定義でき、**S** は set(例; *XtSetValues()*)でき、**G** は get(例: *XtGetValues()*)で使用できることを意味します。

4.11.1 Tab Button のリソース

XmNxrtGearAnchorSide　XrtGearAnchorSide　XRTGEAR_ANCHOR_SIDE_TOP　CSG

ウィジェットのどの側が親に接続するか指定します。有効な値は、XRTGEAR_ANCHOR_SIDE_LEFT 、 XRTGEAR_ANCHOR_SIDE_RIGHT 、 XRTGEAR_ANCHOR_SIDE_TOP または XRTGEAR_ANCHOR_SIDE_BOTTOM です。このリソースは、Tab Button の親が Tab Manager でない時にセットしてください。

XmNbackground　Pixel　dynamic　CSG

ボタンの背景色を指定します。

XmNxrtGearCornerSize　Dimension　dynamic　CSG

tab ボタンの角の大きさや形を指定します。XmNxrtGearStyle が XRTGEAR_STYLE_CHAMFERED(面取り角) または XRTGEAR_STYLE_ROUNDED の場合、XmNxrtGearCornerSize は、角の幅と高さを pixel で指定します。XmNxrtGearStyle が XRTGEAR_STYLE_SLANTED の場合は、XmNxrtGearCornerSize は傾きを pixel で指定します。

XmNxrtGearStyle が XRTGEAR_STYLE_SQUARE であると、XmNxrtGearCornerSize は無視されます。

Tab Button が Tab Manager の子である時は、Tab Manager の XmNxrtGearCornerSize が使用されます。(但し、Tab Button のリソースが、Tab Manager が manage された後にセットされていない場合)

XmNfontList　XmFontList　dynamic　CSG

ウィジェットのテキストに使用されるフォント・リストを指定します。

XmNforeground　Pixel　dynamic　CSG

ボタンの前景色を指定します。

XmNheight Dimension dynamic CSG

(枠を除く)ウィンドウの高さを pixel で指定します。

XmNxrtGearLabelPixmap Pixmap XmUNSPECIFIED_PIXMAP CSG

XmNxrtGearLabelType が XRTGEAR_TYPE_PIXMAP または XRTGEAR_TYPE_STRING_PIXMAP の時に表示されるラベルの pixmap を指定します。XmNxrtGearLabelPixmap と XmNxrtGearMask は同じ XPM ファイルを共有できます。

XmNlabelString XmString dynamic CSG

XmNxrtGearLabelType が XRTGEAR_TYPE_STRING または XRTGEAR_TYPE_STRING_PIXMAP の時に表示されるラベルのストリングを指定します。

XmNxrtGearLabelType XrtGearType XRTGEAR_TYPE_STRING CSG

ラベルにストリング、pixmap または両方を表示するかどうか指定します。有効な値は XRTGEAR_TYPE_STRING(XmString) 、 XRTGEAR_TYPE_PIXMAP(Pixmap) または XRTGEAR_TYPE_STRING_PIXMAP です。XmNlabelType が XmSTRING または XmPIXMAP にセットされている時は、対応する XmNxrtGearLabelType リソースもセットされます。XmNlabelType と XmNxrtGearLabelType の両方が同時にセットされた場合は、XmNxrtGearLabelType が優先されます。

XmNxrtGearLayoutSpacing Dimension 2 CSG

テキストと pixmap の間の距離を pixel で指定します。このリソースは、XmNxrtGearLabelType が XRTGEAR_TYPE_STRING_PIXMAP の時にのみ使用されます。

XmNmarginHeight Dimension 2 CSG

ボタンの上部/下部とその影の間の距離を指定します。

XmNmarginWidth Dimension 2 CSG

ボタンの左端/右端とその影の距離を指定します。

XmNXrtGearMask Pixmap XmUNSPECIFIED_PIXMAP CSG

clip マスクを指定します。マスクの 0 以外の部分が描画されます。clip マスクはモノクロ(bit depth=1)です。XmNxrtGearLabelPixmap と XmNxrtGearMask は、同じ XPM ファイルを使用することができます。

XmNxrtGearPixmapRotation XrtGearRotation XRTGEAR_ROTATE_NONE CSG

pixmap の回転を指定します。有効な値は、XRTGEAR_ROTATE_NONE、XRTGEAR_ROTATE_90、XRTGEAR_ROTATE_180、XRTGEAR_ROTATE_270 です。回転は反時計方向に行われます。

Tab Button が Tab Manager の子の場合は、Tab Manager の XmNxrtGearTabRotation がこのリソースの値をコントロールします。

XmNxrtGearStringLayout　XrtGearStringLayout　XRTGEAR_STRING_RIGHT　CSG

pixmap に対するテキストの相対的な位置を指定します。有効な値は XRTGEAR_STRING_CENTER 、 XRTGEAR_STRING_TOP 、 XRTGEAR_STRING_LEFT 、 XRTGEAR_STRING_BOTTOM、XRTGEAR_STRING_RIGHT です。

Tab Button が Tab Manager の子の場合は、Tab Manager の XmNxrtGearTabRotation がこのリソースの値をコントロールします。

XmNxrtGearLabelType が XRTGEAR_TYPE_STRING_PIXMAP でない時は、このリソースは無視されます。

XmNxrtGearStringRotation　XrtGearRotation　XRTGEAR_ROTATE_NONE　CSG

テキストの回転を指定します。有効な値は、 XRTGEAR_ROTATE_NONE 、 XRTGEAR_ROTATE_90、XRTGEAR_ROTATE_180、XRTGEAR_ROTATE_270 です。回転は反時計方向に行われます。

Tab Button が Tab Manager の子である時は、Tab Manager の XmNxrtGearTabRotation がこのリソースをコントロールします。

XmNxrtGearStyle　XrtGearStyle　XRTGEAR_STYLE_ROUNDED　CSG

tab の角のスタイルを指定します。有効な値は、 XRTGEAR_STYLE_ROUNDED 、 XRTGEAR_STYLE_SQUARE、XRTGEAR_STYLE_SLANTED、XRTGEAR)STYLE_CHAMFERED です。

XmNuserData　XtPointer　NULL　CSG

アプリケーションで定義されているデータのポインタを指定します。

XmNwidth　Dimension　dynamic　CSG

(枠を除く)ウィンドウの幅を pixel で指定します。

XmNx　Position　0　CSG

親ウィジェットの内側の左上角に対応する、ウィジェットの外側の左上角の X 座標。

XmNy　Position　0　CSG

親ウィジェットの内側の左上角に対応する、ウィジェットの外側の左上角の Y 座標。

4.11.2 Tab Manager のリソース

XmNxrtGearActivePageNumber　int　dynamic　CSG

アクティブな page 番号を指定します。存在しない page 番号がセットされると、空白の page が表示されます。このリソースは、XRTGEAR_PAGE_FIRST または XRTGEAR_PAGE_LAST にもセットできます。 XmNxrtGearActivePageWidget と XmNxrtGearActivePageNumber の両方が同時にセットされると、XmNxrtGearActivePageWidget の値が優先されます。

XmNxrtGearActivePageWidget　Widget　dynamic　CSG

現在の page ウィジェットを指定します。NULL がセットされると、空白の page が表示されます。XmNxrtGearActivePageWidget と XmNxrtGearActivePageNumber が両方同時にセットされると、XmNxrtGearActivePageWidget の値が優先されます。

XmNxrtGearActiveTabWidget Widget dynamic CSG

現在アクテイブな tab ウィジェットを指定します。NULL がセットされると、空白の page が表示されます。XmNxrtGearActivePageWidget と XmNxrtGearActivePageNumber が両方同時にセットされると、XmNxrtGearActivePageWidget の値が優先されます。

XmNxrtGearArrowDisplayPolicy XrtGearDisplayPolicy XRTGEAR_DISPLAY_AS_NEEDED CSG

矢印ボタンを表示するかどうか指定します。有効な値は次の通りです。

XRTGEAR_DISPLAY_AS_NEEDED	矢印ボタンを必要時に表示
XRTGEAR_DISPLAY_ALWAYS	矢印ボタンを常時表示
XRTGEAR_DISPLAY_NEVER	矢印ボタンは表示しない

XmNxrtGearArrowHeight Dimension 11 CSG

矢印ボタンの高さを指定します。

このリソースがセットされず、XmNxrtGearArrowWidth がセットされると、高さは XmNxrtGearArrowWidth の値にセットされます。どちらのリソースもセットされていなければ、矢印ボタンの幅と高さは、tab ボタンの高さの半分のサイズにセットされます。

XmNxrtGearArrowWidth Dimension 11 CSG

矢印ボタンの幅を指定します。

このリソースがセットされず、XmNxrtGearArrowHeight がセットされると、幅は XmNxrtGearArrowHeight の値にセットされます。どちらのリソースもセットされていなければ、矢印ボタンの幅と高さは、tab ボタンの高さの半分のサイズにセットされます。

XmNxrtGearPageAttachPolicy XrtGearPageAttachPolicy XRTGEAR_ATTACH_ALWAYS CSG

Tab Button を自動的に page に割り振るかどうか指定します。有効な値は次の通りです。

XRTGEAR_ATTACH_ALWAYS	必要時には常に自動的に割り振る
XRTGEAR_ATTACH_NONE	Tab Button は決して page に自動的に割り振られない。必ず、明示的に指定される必要がある

XmNxrtGearPageAttachPolicy が、Tab Manager が作成された後にセットされた場合は、次に Tab Button または page が作成または破棄されるまで、この設定は有効になりません。

XmNxrtGearCornerSize Dimension 5 CSG

Tab Button の角のサイズや形を指定します。XmNxrtGearStyle が XRTGEAR_STYLE_CHAMFERED(面取り角) または XRTGEAR_STYLE_ROUNDED の時は、XmNxrtGearCornerSize は角の幅と高さを pixel で指定します。XmNxrtGearStyle が XRTGEAR_STYLE_SLANTED の場合は、XmNxrtGearCornerSize は傾きを pixel で指定します。

XmNxrtGearStyle が XRTGEAR_STYLE_SQUARE である時は、XmNxrtGearCornerSize は無視されます。

XmNxrtGearFirstVisibleTab int dynamic CSG

現在(一番左または上に)表示されている最初の tab を指定します。このリソースを設定すると、その tab が全て表示されるように、必要に応じてスクロールします。

XmNhighlightThickness Dimension 2 CSG

ハイライトの幅を指定します。

XmNxrtGearLastVisibleTab int dynamic CSG

現在(一番右または下に)表示されている最後の tab を指定します。このリソースを設定すると、その tab が全て表示されるように、必要に応じてスクロールします。

XmNxrtGearMarginHeight Dimension 2 CSG

Tab Manager の上と下のマージン(Tab Manager の上部と下部のスペース)を指定します。

XmNxrtGearMarginWidth Dimension 2 CSG

Tab Manager の左と右のマージン(Tab Manager の右端と左端のスペース)を指定します。

XmNxrtGearPageCallback XtCallbackList NULL CSG

XmNxrtGearPageNumber が、インタラクションまたはプログラムによって変更された時に呼び出されるコールバックのリストを指定します。各ルーチンには、XrtGearPageCallbackStruct のポインタが渡されます。コールバックは *XtAddCallback()* を使用してリストに追加してください。コールバックは次のように構成されます。

```
void
CallbackProcedure(
  Widget  widget,
  XtPointer   client_data,
  XtPointer   call_data /* points to an XrtGearPageCallbackStruct */
)
```

XrtGearPageCallbackStruct は次のように定義されます。

```
typedef struct {
  int         reason;          // Read-only
  XEvent      *event;          // Read-only
  XrtGearAction   action;          // Read-only
  int         prev_page_number;    // Read-only
  Widget      prev_page_widget;    // Read-only
  Widget      prev_tab;        // Read-only
  Widget      tab;
  int         page_number;
  Widget      page_widget;
  Boolean     doit;
} XrtGearPageCallbackStruct;
```

reason は XRTGEAR_REASON_PAGE_BEGIN または XRTGEAR_REASON_PAGE_END です。*event* はコールバックを呼び出した XEvent で、プログラムによって呼び出された場合は NULL です。*action* は XRTGEAR_ACTION_SET_VALUE。*prev_page_number* は前の page 番号(空白 page が表示されている場合は XRTGEAR_NOVALUE)。*prev_page_widget* は前の page ウィジェット(イニシャライズ時は NULL)。*prev_tab* は前の tab ウィジェット(イニシャライズ時は NULL)。*tab* は新しい tab ウィジェット。*page_number* と *page_widget* は表示される page 番号とウィジェットです。これらは編集可能で、両方が変更された場合は *page_widget* が優先されます。

page が変更されないようにするには、*reason* が_BEGIN の時に *doit* を False にセットしてください。

XmNxrtGearResizeCallback　XtCallbackList　NULL　CSG

Tab Manager がリサイズされた時に呼び出されるコールバックのリストを指定します。各ルーチンには、XrtGearResizeCallbackStruct のポインタが渡されます。コールバックは、*XtAddCallback()*を使用してリストに追加してください。コールバックは次のように構成されます。

```
void
CallbackProcedure(
  Widget  widget,
  XtPointer   client_data,
  XtPointer   call_data /* points to an XrtGearResizeCallbackStruct */
)
```

XrtGearResizeCallbackStruct は次のように定義されます。

```
typedef struct {
  int     reason; // Read-only
  XEvent  *event; // Read-only
  Dimension   width;
  Dimension   height;
} XrtGearResizeCallbackStruct;
```

reason は XRTGEAR_REASON_RESIZE。*event* はコールバックを呼び出した XEvent で、コールバックがプログラムによって呼び出された場合はNULLです。*width* と *height* は、Tab Managerの新しい幅と高さです。

XmNxrtGearShadowThickenss　Dimension　2　CSG

枠の影の幅を指定します。

XmNxrtGearTabResize　Boolean　True　CSG

XmNxrtGearTabSide の値に応じて、全ての tab がその行にある一番大きな tab の幅または高さに合わせてリサイズされるかどうかを指定します。True であれば、このリソースは、XmNxrtGearTabStretch の前に適用されます。

XmNxrtGearTabRotation XrtGearTabRotation XRTGEAR_ROTATE_UNSPECIFIED CSG

Tab Button の XmNxrtGearPixmapRotation 、 XmNxrtGearStringRotation 、 XmNxrtGearStringLayout の値を指定します。このリソースの有効な値は次の通りです。

XRTGEAR_ROTATE_UNSPECIFIED	回転は指定されない(アプリケーションではこの値にセットできません。)
XRTGEAR_ROTATE_NONE	回転しない
XRTGEAR_ROTATE_90	反時計方向に 90 度回転
XRTGEAR_ROTATE_180	反時計方向に 180 度回転
XRTGEAR_ROTATE_270	反時計方向に 270 度回転

次の表は、上記の XmNxrtGearTabRotation の値とそれに影響される Tab Button リソースとの関係を示しています。

XmNxrtGearTabRotation の値	XmNxrtGearPixmapRotation と XmNxrtGearStringRotation がこの値にセットされる	そして XmNxrtGearStringLayout がこの値にセットされる
XRTGEAR_ROTATE_NONE	XRTGEAR_ROTATE_NONE	XRTGEAR_STRING_RIGHT
XRTGEAR_ROTATE_90	XRTGEAR_ROTATE_90	XRTGEAR_STRING_TOP
XRTGEAR_ROTATE_180	XRTGEAR_ROTATE_180	XRTGEAR_STRING_LEFT
XRTGEAR_ROTATE_270	XRTGEAR_ROTATE_270	XRTGEAR_STRING_BOTTOM

XmNxrtGearTabRotation がデフォルト値の XRTGEAR_ROTATE_UNSPECIFIED にセットされると、Tab Button のリソースは変更されません。

XmNxrtGearTabSide XrtGearTabSide XRTGEAR_SIDE_TOP CSG

tab が表示される側を指定します。有効な値は、 XRTGEAR_SIDE_TOP 、 XRTGEAR_SIDE_BOTTOM、XRTGEAR_SIDE_LEFT、XRTGEAR_SIDE_RIGHT です。

XmNxrtGearTabSpacing int 0 CSG

tab 間の距離を pixel で指定します。負の値がセットされると tab は重なり合います。

XmNxrtGearTabStretch Boolean False CSG

全ての tab が表示領域一杯に表示されるようにストレッチ(伸ばす)するかどうか指定します。スクロールする tab の場合は無視されます。

4.11.3 Tab Manager の Constraint リソース

XmNxrtGearChildType XrtGearChildType XRTGEAR_CHILD_TYPE_PAGE G

Tab Manager ウィジェットの子の型を指定します。Tab Button ウィジェットである子が tab として使用されます。その他のウィジェット・クラスは page です。

XRTGEAR_CHILD_TYPE_PAGE	子が Tab Button でない

XRTGEAR_CHILD_TYPE_TAB　子が Tab Button

XRTGEAR_CHILD_INTERNAL　子が private、internal ウィジェット

XmNxrtGearPageNumber　int　dynamic　CSG

page ウィジェットでは、このリソースは page ウィジェットに振り分けられた page 番号を指定します。デフォルトの番号は、page が作成される順に 0 から計算されます。このウィジェットの XmNxrtGearChildType が XRTGEAR_TYPE_PAGE でない時は、このリソースは無視されます。

tab ウィジェットでは、このリソースは tab に振り分けられた page 番号を指定します。このリソースが-1 にセットされると、XmNxrtGearPageWidget で指定されている page が使用されます。

XmNxrtGearPageWidget　Widget　NULL　CSG

tab ウィジェットに振り分けられた page を指定します。複数の tab が同じ page ウィジェットを共有することができます。このウィジェットの XmNxrtGearChildType が XRTGEAR_TYPE_PAGE である時は、このリソースは無視されます。

XmNxrtGearWidgetPosition　int　dynamic　G

Tab Manager の子が作成される順序を示します。最初に作成された子の XmNxrtGearWidgetPosition 値は 0、二番目は 1 というように続きます。この値は、アプリケーションでセットすることはできません。

4.11.4 Tab Button によって継承されるリソース

Resource	Inherited From
XmNaccelerator	XmLabel
XmNaccelerators	Core
XmNactivateCallback	XmPushButton
XmNalignment	XmLabel
XmNancestorSensitive	Core
XmNarmCallback	XmPushButton
XmNarmColor[a]	XmPushButton
XmNarmPixmap	XmPushButton
XmNbackground	Core
XmNbackgroundPixmap	Core
XmNborderColor[a]	Core
XmNborderPixmap	Core
XmNborderWidth	Core
XmNbottomShadowColor[a]	Primitive
XmNbottomShadowPixmap	Primitive

Resource	Inherited From
XmNinitialResourcesPersistent	Core
XmNlabelPixmap	XmLabel
XmNlabelString	XmLabel
XmNmappedWhenManaged	Core
XmNmarginBottom	XmLabel
XmNmarginHeight	XmLabel
XmNmarginLeft	XmLabel
XmNmarginRight	XmLabel
XmNmarginTop	XmLabel
XmNmarginWidth	XmLabel
XmNmnemonic[b]	XmLabel
XmNmnemonicCharSet[b]	XmLabel
XmNmultiClick	XmPushButton
XmNnavigationType	Primitive
XmNrecomputeSize	XmLabel

Resource	Inherited From
XmNcolormap	Core
XmNdefaultButtonShadowThickness	XmPushButton
XmNdepth	Core
XmNdestroyCallback	Core
XmNdisarmCallback	XmPushButton
XmNfillOnArm	XmPushButton
XmNfontList	XmLabel
XmNforeground	Primitive
XmNheight	Core
XmNhelpCallback	Primitive
XmNhighlightColor[a]	Primitive
XmNhighlightOnEnter	Primitive
XmNhighlightPixmap	Primitive
XmNhighlightThickness	Primitive

Resource	Inherited From
XmNscreen	Core
XmNsensitive	Core
XmNshadowThickness[c]	Primitive
XmNshowAsDefault	XmPushButton
XmNstringDirection	XmLabel
XmNtopShadowColor[a]	Primitive
XmNtopShadowPixmap	Primitive
XmNtranslations	Core
XmNtraversalOn	Primitive
XmNuserData	Primitive
XmNwidth	Core
XmNx	Core
XmNy	Core

a: Tab Button は、色のリソースを関連する page ウィジェットから継承する
b: `XmNxrtGearStringRotation` が `XRTGEAR_ROTATE_NONE` の場合にのみサポートされる
c: Tab Button が Tab Manager の子である場合は、Tab Manager の `XmNshadowThickness` の設定が使用される

4.11.5 Tab Manager によって継承されるリソース

Resource	Inherited From
XmNaccelerators	Core
XmNancestorSensitive	Core
XmNbackground	Core
XmNbackgroundPixmap	Core
XmNborderColor	Core
XmNborderPixmap	Core
XmNborderWidth	Core
XmNbottomShadowColor	XmManager
XmNbottomShadowPixmap	XmManager
XmNchildren	Composite
XmNcolormap	Core
XmNdepth	Core
XmNdestroyCallback	Core
XmNforeground	XmManager
XmNheight	Core
XmNhelpCallback	XmManager
XmNhighlightColor	XmManager
XmNhighlightPixmap	XmManager

Resource	Inherited From
XmNinitialFocus	XmManager
XmNinitialResourcesPersistent	Core
XmNinsertPosition	Composite
XmNmappedWhenManaged	Core
XmNnavigationType	XmManager
XmNnumChildren	Composite
XmNscreen	Core
XmNsensitive	Core
XmNshadowThickness	XmManager
XmNstringDirection	XmManager
XmNtopShadowColor	XmManager
XmNtopShadowPixmap	XmManager
XmNtranslations	Core
XmNtraversalOn	XmManager
XmNuserData	XmManager
XmNwidth	Core
XmNx	Core
XmNy	Core

4.12 手続きと方法

この項では、Tab Button と Tab Manager の手続きと方法をアルファベット順に解説します。

XmCreateXrtTabButton()

Tab Button ウィジェットを作成します。

```
Widget
XmCreateXrtTabButton(
  Widget  parent,
  String  name,
  Arg     *arglist,
  int     argcount
  )
```

parent は親。*name* は作成されるウィジェット名。*arglist* は *argcount* 引数のリストです。

XmCreateXrtTabManager()

Tab Manager ウィジェットを作成します。

```
Widget
XmCreateXrtTabManager(
  Widget  parent,
  String  name,
  Arg     *arglist,
  int     argcount
  )
```

parent は親。*name* は作成されるウィジェット名。*arglist* は *argcount* 引数のリストです。

XmIsXrtTabButton()

ウィジェットが Tab Button であれば `True` を返します。

```
Boolean
XmIsXrtTabButton(
  Widget  widget
  )
```

XmIsXrtTabManager()

ウィジェットが Tab Manager であれば `True` を返します。

```
Boolean
XmIsXrtTabManager(
  Widget  widget
  )
```

XrtGearMrmInitialize()

UIL アプリケーションに使用するために、Mrm に XRT/gear ウィジェットを登録します。この手続きは、*MrmInitialize()*の後に呼び出して下さい。

```
void
XrtGearMrmInitialize(void)
```

XrtGearSetXmStringConverter()

XmRXmString リソース変換プログラムをセットし、XmString 値を作成します。

```
void
XrtGearSetXmStringConverter(void)
```

変換は、エンベッドされたフォント(タグ)を指定するために、次のシンタックスを使用します。

@font@　　フォントのタグは `XmNfontList`

@P@　　前のフォント

\@　　"@"とする

例:

```
For a @bold@free evaluation@P@ of any product
```

フォント・リストを設定

```
fixed,-*-times-bold-r-normal-*-140-*=bold
```

表示結果

For a **free evaluation** of any product

XrtGearTabManagerMakeTabVisible()

Tab Manager ウィジェットの tab を(必要であれば)スクロールし、指定した tab が表示(visible)されるようにします。その tab が表示領域の左にあれば、スクロール後、その tab は一番左に表示されます。その tab が表示領域の右にあれば、スクロール後、その tab は一番右に表示されます。スクロールが成功すると、`True` を返します。

```
Boolean
XrtGearTabManagerMakeTabVisible(
  Widget  widget,
  int     tab_num
  )
```

widget は Tab Manager ウィジェット。*tab_num* は指定した tab のインデックス(0 は一番目、1 は二番目というように続きます)。

4.13 トランスレーションとアクション

4.13.1 デフォルトの Tab Button のトランスレーション

イベント	アクション
~Alt ~Ctrl ~Meta ~Shift <Btn1Down>	XrtArm()
~Alt ~Ctrl ~Meta ~Shift<Btn1Down>(2+)	MultiArm()
~Alt ~Ctrl ~Meta ~Shift <Btn1Up>	Activate() Disarm()
~Alt ~Ctrl ~Meta ~Shift <Btn1Up>(2+)	MultiActivate()
<Key>osfSelect	ArmAndActivate()
<Key>osfActivate	ArmAndActivate()
<Key>osfHelp	Help()
Ctrl ~Alt ~Meta ~Shift <Key>Return	XrtArm() ArmAndActivate()
Ctrl ~Alt ~Meta ~Shift <Key>Space	XrtArm() ArmAndActivate()

4.13.2 Tab Button のアクション

Activate()

`XmNactivateCallback` で指定されているコールバックのリストを呼び出します。

ArmAndActivate()

`XmNarmCallback`、`XmNactivateCallback`、`XmNdisarmCallback` で指定されているコールバックのリストを呼び出します。

Disarm()

`XmNdisarmCallback` で指定されているコールバックのリストを呼び出します。

Help()

前のキーボード･フォーカスを復元し、`XmNhelpCallback` で指定されているコールバックを呼び出します。

MultiActivate()

`XmPushButtonCallbackStruct` の *click_count* メンバをインクレメント(増加)させ、`XmNactivateCallback` と `XmNdisarmCallback` で指定されているコールバックを呼び出します。

このアクション･ルーチンは、`XmNmultiClick` リソースが `XmMULTICLICK_KEEP` にセットされている時にのみ有効です。

MultiArm()

XmNarmCallback で指定されているコールバックのリストを呼び出します。

このアクション・ルーチンは、XmNmultiClick リソースが XmMULTICLICK_KEEP にセットされている時にのみ有効です。

XrtArm()

XmNarmCallback で指定されているコールバックのリストを呼び出します。

5

Toolbar Widget

サンプル･プログラム ■ ウィジェットの概要
方向の指定 ■ マージンとスペース
Constraint リソース ■ ヘルプ･バルーンの追加
リソースの一覧 ■ 手続きと方法

5.1 サンプル･プログラム

Toolbar は、一つの行または列にボタンやウィジェットを表示するシンプルな Manager です。メニューバーの代りとして使用できます。

次のコードは、Toolbar ウィジェットを作成しています。

```
Pixmap cut_pixmap, cut_mask, copy_pixmap, copy_mask,
  paste_pixmap, paste_mask;
Widget column, toolbar, button1, button2, button3;

column = XtVaCreateWidget("column",
  xmRowColumnWidgetClass, toplevel,
  XmNorientation,     XmVERTICAL,
  NULL);
toolbar = XtVaCreateWidget("toolbar",
  xmXrtToolbarWidgetClass,column,
  NULL);
button1 = XtVaCreateManagedWidget("Cut",
  xmXrtPushButtonWidgetClass, toolbar,
  XmNlabelPixmap,     cut_pixmap,
  XmNxrtGearMask,     cut_mask,
  XmNxrtGearLabelType,XRTGEAR_TYPE_PIXMAP,
  NULL);
button2 = XtVaCreateManagedWidget("Copy",
  xmXrtPushButtonWidgetClass, toolbar,
  XmNlabelPixmap,     copy_pixmap,
  XmNxrtGearMask,     copy_mask,
  XmNxrtGearLabelType,    XRTGEAR_TYPE_PIXMAP,
  NULL);
```

```
button3 = XtVaCreateManagedWidget("Paste",
  xmXrtPushButtonWidgetClass, toolbar,
  XmNlabelPixmap, paste_pixmap,
  XmNxrtGearMask, paste_mask,
  XmNxrtGearLabelType,XRTGEAR_TYPE_PIXMAP,
  NULL);
XtManageChild(toolbar);
XtManageChild(column);
```

子ウィジェットが追加される度に、Toolbar の左から右に配置されていきます。

図 29 は、先のコードで作成された Toolbar です。



図 29　サンプルの Toolbar

5.2　ウィジェットの概要

Include File:	*<Xm/XrtToolbar.h>*
Class Name:	XmXrtToolbar
Class Hierarchy:	Core → Composite → Constraint → XmManager → XmXrtManager → XmXrtToolbar
Class Pointer:	xmXrtToolbarWidgetClass

Toolbar ウィジェットは、XmManager のサブクラスである XmXrtManager ウィジェットのサブクラスです。

5.3　方向の指定

デフォルトでは、Toolbar はウィジェットやアイコンを行(横方向)の形式に表示します。これを列として表示するには、`XmNxrtGearOrientation` を `XmVERTICAL` にセットしてください。



図 30　Toolbar の方向

5.4　マージンとスペース

`XmNxrtGearMarginHeight` は、Toolbar の上部/下部と子ウィジェットの間の距離を指定します。`XmNxrtGearMarginWidth` は、Toolbar の右端/左端と子ウィジェットの間の距離を指定します。



図 31　Toolbar のマージン

どちらのリソースも、デフォルト値は 2 です。

5.5　Constraint リソース

次の Toolbar の constraint リソースは、子ウィジェットとその親の Toolbar の間の距離を指定します。

- ウィジェットとアイテムの下のスペースを指定する `XmNxrtGearBottomSpace`
- ウィジェットとアイテムの上のスペースを指定する `XmNxrtGearTopSpace`
- ウィジェットとアイテムの左のスペースを指定する `XmNxrtGearLeftSpace`
- ウィジェットとアイテムの右のスペースを指定する `XmNxrtGearRightSpace`

図 32 は、最初の子ウィジェットに上記の全てのリソースが 20 にセットされている Toolbar です。

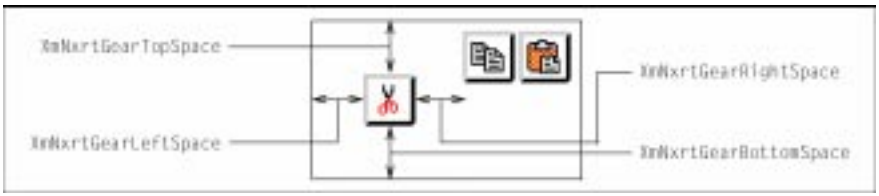


図 32　Toolbar の constraint リソース

マージンのリソースの方向は、Toolbar の向きが変わっても変更されません。(XmNxrtGearLeftSpace は、常に子ウィジェットとその左側の間の距離を指定します。)

また、Toolbar の constraint リソースは、加算されていきます。例えば、一つの子ウィジェットの XmNxrtGearRightSpace が 20 にセットされ、その右隣のウィジェットの XmNxrtGearleftSpace が 10 にセットされていれば、その二つのウィジェットの間の距離は 30pixel になります。

子ウィジェットの位置の指定

XmNxrtGearPosition は、子ウィジェットのリストの中の子の位置を指定します。このリソースの値を変更すると、ウィジェットが manage された場合、そのウィジェットの視覚的な位置も変更されます。XmNxrtGearPosition を 0 または XmLAST_POSITION にセットすると、そのウィジェットはリストの最初または最後に移動します。

5.6　ヘルプ･バルーンの追加

Toolbar にヘルプ･バルーン(マウスを置くとポップアップする注釈)を追加するには Widget Tip を使用してください。これは、Widget Tip の章に解説されています。(Widget Tip は、(Manager も含む)どのウィジェットにも付けることができます。)

5.7　リソースの一覧

Toolbar のリソースの一覧です。リソース名の後は、データ型、デフォルト値、そしてリソースと共に使用される手続きです。**C** は、そのリソースが create(例: *XtCreateWidget()*)で定義でき、**S** は set(例; *XtSetValues()*)でき、**G** は get(例: *XtGetValues()*)で使用できることを意味します。

XmNxrtGearMarginHeight　Dimension　2　CSG

Toolbar の上部/下部と子ウィジェットの間の距離を指定します。

XmNxrtGearMarginWidth　Dimension　2　CSG

Toolbar の左端/右端と子ウィジェットの間の距離を指定します。

XmNxrtGearOrientation　unsigned char　XmHORIZONTAL　CSG

Toolbar を行(XmHORIZONTAL)または列(XmVERTICAL)のどちらに表示するか指定します。

5.7.1 Toolbar の Constraint リソース

XmNxrtGearBottomSpace　Dimension　0　CSG

ウィジェットとアイテムの下の間の距離を指定します。

XmNxrtGearLeftSpace　Dimension　0　CSG

ウィジェットとその隣のウィジェット間の距離を指定します。

XmNxrtGearPosition　int　XmLAST_POSITION　CSG

リスト内で、ウィジェットの位置(0 から開始)を指定します。このリソースの値を変更すると、ウィジェットが manage された場合、そのウィジェットの視覚的な位置も変更されます。XmNxrtGearPosition を 0 または XmLAST_POSITION にセットすると、そのウィジェットはリストの最初または最後に移動します。

XmNxrtGearRightSpace　Dimension　0　CSG

ウィジェットとその隣のウィジェットの間の距離を指定します。

XmNxrtGearTopSpace　Dimension　0　CSG

ウィジェットと前の行との間の距離を指定します。

5.7.2 Toolbar によって継承されるリソース

Resource	Inherited From
XmNaccelerators	Core
XmNancestorSensitive	Core
XmNbackground	Core
XmNbackgroundPixmap	Core
XmNborderColor	Core
XmNborderPixmap	Core
XmNborderWidth	Core
XmNbottomShadowColor	XmManager
XmNbottomShadowPixmap	XmManager
XmNchildren	Composite
XmNcolormap	Core
XmNdepth	Core
XmNdestroyCallback	Core
XmNforeground	XmManager
XmNheight	Core
XmNhelpCallback	XmManager
XmNhighlightColor	XmManager
XmNhighlightPixmap	XmManager

Resource	Inherited From
XmNinitialFocus	XmManager
XmNinitialResourcesPersistent	Core
XmNinsertPosition	Composite
XmNmappedWhenManaged	Core
XmNnavigationType	XmManager
XmNnumChildren	Composite
XmNscreen	Core
XmNsensitive	Core
XmNshadowThickness	XmManager
XmNstringDirection	XmManager
XmNtopShadowColor	XmManager
XmNtopShadowPixmap	XmManager
XmNtranslations	Core
XmNtraversalOn	XmManager
XmNuserData	XmManager
XmNwidth	Core
XmNx	Core
XmNy	Core

5.8 手続きと方法

Toolbar の手続きと方法をアルファベット順に解説します。

XmCreateXrtToolbar()

Toolbar ウィジェットを作成します。

```
Widget
XmCreateXrtToolbar(
  Widget  parent,
  String  name,
  Arg     *arglist,
  int     argcount
  )
```

parent は親。*name* は作成されるウィジェット名。*arglist* は *argcount* 引数のリストです。

XmIsXrtToolbar()

ウィジェットが Toolbar であれば `True` を返します。

```
Boolean
XmIsXrtToolbar(
  Widget  widget
  )
```

XrtGearMrmInitialize()

UIL アプリケーションに使用するために、Mrm に XRT/gear ウィジェットを登録します。この手続きは、*MrmInitialize()*の後に呼び出して下さい。

```
void
XrtGearMrmInitialize(void)
```

XrtGearSetXmStringConverter()

XmRXmString リソース変換プログラムをセットし、XmString 値を作成します。

```
void
XrtGearSetXmStringConverter(void)
```

変換は、エンベッドされたフォント(タグ)を指定するために、次のシンタックスを使用します。

@font@　　フォントのタグは `XmNfontList`

@P@　　前のフォント

\@　　"@"とする

例:

```
For a @bold@free evaluation@P@ of any product
```

フォント･リストを設定

```
fixed,-*-times-bold-r-normal-*-140-*=bold
```

表示結果

For a **free evaluation** of any product

6

Aligner Widget

Aligner とは ■ サンプル・プログラム

ウィジェットの概要 ■ マージンとスペース

Aligner と XmSeparator ■ ラベルの位置

Constraint リソース ■ リソースの一覧

手続きと方法

6.1 Aligner とは

Aligner は、control ウィジェットとそれに関連するラベル(または他の)ウィジェットのグループを縦に並べる Manager ウィジェットです。control ウィジェットのテキストは、このラベルのベースラインに揃えることができます。

図 33 は、一般的な Aligner ウィジェットです。

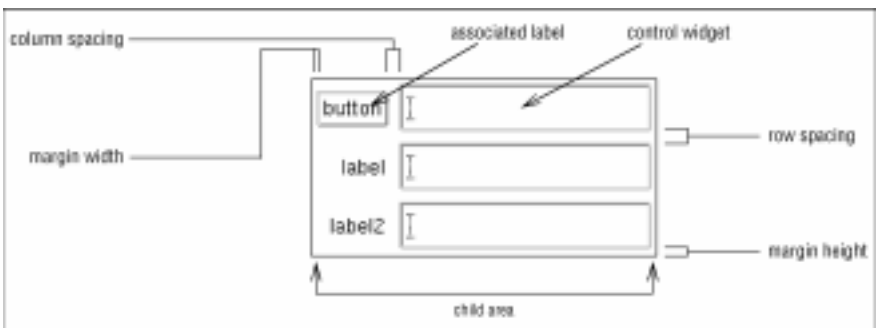


図 33　Aligner ウィジェット

6.2 サンプル･プログラム

次のコードは、Aligner ウィジェットとその子を作成します。

```
form = XtVaCreateWidget("form",
  xmFormWidgetClass,      parent,
  NULL);
aligner = XtVaCreateManagedWidget("aligner",
  xmXrtAlignerWidgetClass,form,
  NULL);
XtVaCreateManagedWidget("button",
  xmPushButtonWidgetClass,aligner,
  NULL);
XtVaCreateManagedWidget("text",
  xmTextWidgetClass,      aligner,
  NULL);
XtVaCreateManagedWidget("label",
  xmLabelWidgetClass,     aligner,
  NULL);
XtVaCreateManagedWidget("text2",
  xmTextWidgetClass,      aligner,
  NULL);
XtVaCreateManagedWidget("label2",
  xmLabelWidgetClass,     aligner,
  NULL);
XtVaCreateManagedWidget("text3",
  xmTextWidgetClass,      aligner,
  NULL);
XtManageChild(form);
```

図 34 は、このコードによって作成された Aligner です。

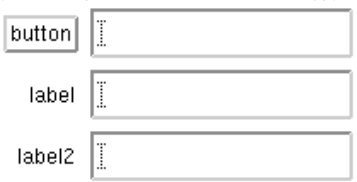


図 34　サンプルの Aligner

子ウィジェットは、作成された順に並べられます。各ラベルは、それに関連する control ウィジェットの左または上部に配置できます。Aligner ウィジェットがリサイズすると、control ウィジェットもそれに従って縦または横方向に伸びる(ストレッチ)するように設定できます。

6.3 ウィジェットの概要

Include File:	*<Xm/XrtAligner.h>*
Class Name:	XmXrtAligner
Class Hierarchy:	Core → Composite → Constraint → XmManager → XmXrtManager → XmXrtAligner
Class Pointer:	xmXrtAlignerWidgetClass

Aligner ウィジェットは、XmManager ウィジェットのサブクラスの XmXrtManager ウィジェットのサブクラスです。

6.4 マージンとスペース

各ラベルと control ウィジェットの間の距離、子ウィジェットの周囲、行間等はそれぞれ指定することができます。

`XmNxrtGearColumnSpacing` は、各ラベルと control の間の距離を pixel で指定します。デフォルトは 5 です。

`XmNxrtGearMarginHeight` は子ウィジェットの上部と下部、`XmNxrtGearMarginWidth` は子ウィジェットの右と左のスペースをそれぞれ指定します。どちらのリソースもデフォルト値は 2 です。

`XmNxrtGearRowSpacing` は、行(ラベルと control のペア)間を pixel で指定します。デフォルト値は 5 です。

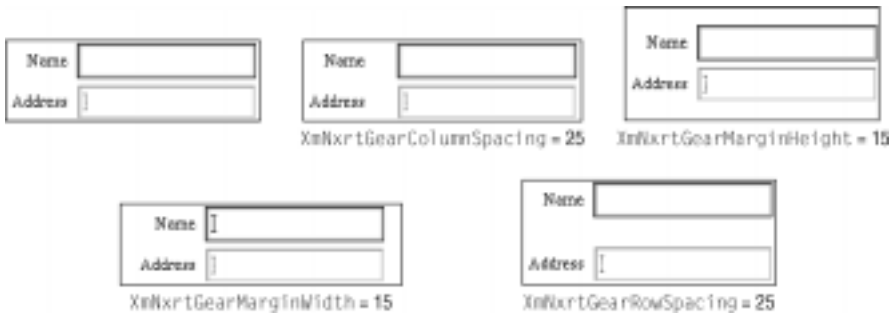


図 35　Aligner マージンとスペースのリソース

6.5 Aligner と XmSeparator

XmSeparator が Aligner ウィジェットの子として定義されている時は、それが(ラベル/control のペアの一部としてではなく)一つの行として処理されます。これを利用して、Aligner ウィジェットを分割することができます。

例として、次のコードでは、二つのラベル/contorol のペアを作成し分割しています。

```
form = XtVaCreateWidget("form",
  xmFormWidgetClass,      parent,
  NULL);
aligner = XtVaCreateManagedWidget("aligner",
  xmXrtAlignerWidgetClass, form,
  NULL);
XtVaCreateManagedWidget("button",
  xmPushButtonWidgetClass, aligner,
  NULL);
XtVaCreateManagedWidget("text",
  xmTextWidgetClass,      aligner,
  NULL);
XtVaCreateManagedWidget("separator",
  xmSeparatorWidgetClass, aligner,
  NULL);
XtVaCreateManagedWidget("label",
  xmLabelWidgetClass,     aligner,
  NULL);
XtVaCreateManagedWidget("text2",
  xmTextWidgetClass,      aligner,
  NULL);
```

図 36 は、このコードによって作成された Aligner です。



図 36　Separator を含む Aligner

6.6　ラベルの位置

XmNxrtGearOrientation は、ラベルとそれに関連する control ウィジェットの相対的な位置を指定します。このリソースは、XmHORIZONTAL(control の左にラベルを表示)または XmVERTICAL(control の上にラベルを表示)にセットできます。デフォルトは、XmHORIZONTAL です。[1]

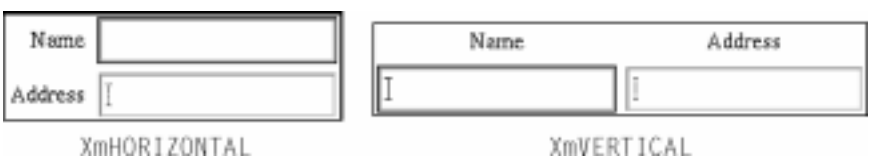


図 37　Aligner ラベルの位置

[1] Aligner ウィジェットが縦方向(vertical)に配置されている場合でも、XmNxrtGearColumnSpacing はラベルと control の間のスペースを指定し、XmNxrtGearRowSpacing はラベル/コントロールのペア間のスペースを指定します。

6.7 Constraint リソース

Aligner の constraint リソースを利用して、control に対応させてラベルを配置し、リサイズや子ウィジェットの位置を指定することができます。

ベースラインに揃える

XmNxrtGearAlignBaseLine は、ラベルのベースラインを control のベースラインと揃えるかどうかを指定します。デフォルトでは、このリソースは True です。

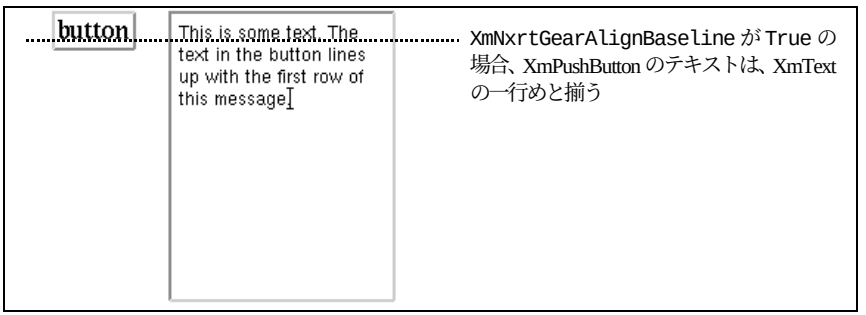


図 38　Aligner のベースライン

XmNxrtGearAlignBaseline は、ラベルと control が XmLabel または XmText ウィジェットのサブクラスである時にのみ使用できます。XmNxrtGearAlignBaseline が True であれば、XmNxrtGearAlignerAlignment を置き換えます。

ラベルの位置

XmNxrtGearAlignerAlignment は、XmNxrtGearAlignBaseline が True にセットされていない場合に、control に対応するラベルの位置を指定します。XmNxrtGearAlignerAlignment は、次の値に設定できます。

XRTGEAR_ALIGNER_TOPRIGHT
XRTGEAR_ALIGNER_MIDDLERIGHT
XRTGEAR_ALIGNER_BOTTOMRIGHT
XRTGEAR_ALIGNER_TOPCENTER
XRTGEAR_ALIGNER_MIDDLECENTER
XRTGEAR_ALIGNER_BOTTOMCENTER
XRTGEAR_ALIGNER_TOPLEFT
XRTGEAR_ALIGNER_MIDDLELEFT
XRTGEAR_ALIGNER_BOTTOMLEFT

Topleft	Topcenter	Topright
Middleleft	Middlecenter	Middleright
Bottomleft	Bottomcenter	Bottomright

上記設定は、XmVERTICAL と XmHORIZONTAL 両方向に適用されます。

リサイズのコントロール

XmNxrtGearResizeHeight と XmNxrtGearResizeWidth は、行または列がリサイズされた時に、それに従って control 子ウィジェットも縦または横方向にリサイズするかどうかを指定します。デフォルトでは、どちらのリソースも False で、control 子ウィジェットはリサイズしません。

子ウィジェットの位置

XmNxrtGearPosition は、子ウィジェット・リストの子の位置を指定します。子ウィジェットが作成されると、0 から順番に番号付けされます。下記のコードでは、

子ウィジェットが図 39 のように 0 から 5 の番号が付いています。

```
button = XtVaCreateManagedWidget("button",
  xmPushButtonWidgetClass,aligner,
  NULL);
text1 = XtVaCreateManagedWidget("text",
  xmTextWidgetClass,      aligner,
  NULL);
label1 = XtVaCreateManagedWidget("label",
  xmLabelWidgetClass,     aligner,
  NULL);
text2 = XtVaCreateManagedWidget("text2",
  xmTextWidgetClass,      aligner,
  NULL);
label2 = XtVaCreateManagedWidget("label2",
  xmLabelWidgetClass,     aligner,
  NULL);
text3 = XtVaCreateManagedWidget("text3",
  xmTextWidgetClass,      aligner,
  NULL);
```



図 39　子ウィジェットの位置

XmNxrtGearPosition の値を変更すると、ウィジェットが managed された場合、そのウィジェットの視覚的な位置も変更します。XmNxrtGearPosition が一つのウィジェットで変更されると、他の子ウィジェットの XmNxrtGearPosition は必要に応じて調整されます。例えば、次のように二番目のテキスト・フィールドが position を変更するとします。

```
XtVaSetValues(text2,
  XmNxrtGearPosition, 2,
  NULL);
```

それに従って、子ウィジェットは図 40 のように配置されます。



図 40　position 変更後の子ウィジェット

二番目のテキスト・ウィジェットが position 2 に変更すると、既に position 2 にあるウィジェットが position 3 に移動します。

XmNxrtGearPosition を 0 または XmLAST_POSITION にセットすると、そのウィジェットはリストの最初または最後に移動します。

6.8 リソースの一覧

Aligner のリソースの一覧です。リソース名の後は、データ型、デフォルト値、そしてリソースと共に使用される手続きです。**C** は、そのリソースが create(例: *XtCreateWidget()*)で定義でき、**S** は set(例; *XtSetValues()*)でき、**G** は get(例: *XtGetValues()*)で使用できることを意味します。

XmNxrtGearColumnSpacing Dimension 5 CSG

各ラベルと control ウィジェットの間の距離を pixel で指定します。

XmNxrtGearMarginHeight Dimension 2 CSG

子ウィジェットの上部と下部のスペースを pixel で指定します。

XmNxrtGearMarginWidth Dimension 2 CSG

子ウィジェットの左と右のスペースを pixel で指定します。

XmNxrtGearOrientation unsigned char XmHORIZONTAL CSG

control ウィジェットに対応するラベルの相対的な位置を指定します。有効な値は XmHORIZONTAL(control の左にラベルを表示)、XmVERTICAL(control の上にラベルを表示)です。

XmNxrtGearRowSpacing Dimension 5 CSG

行間を pixel で指定します。

6.8.1 Aligner の Constraint リソース

XmNxrtGearAlignerAlignment XrtGearAlignerAlignment XRTGEAR_ALIGNER_CENTER CSG

control ウィジェットに対応するラベルの位置を指定します。下記の設定は XmHORIZONTAL と XmVERTICAL の両方に適用されます。

XRTGEAR_ALIGNER_TOPRIGHT	上右
XRTGEAR_ALIGNER_MIDDLERIGHT	中央右
XRTGEAR_ALIGNER_BOTTOMRIGHT	下右
XRTGEAR_ALIGNER_TOPCENTER	上中央
XRTGEAR_ALIGNER_MIDDLECENTER	中央
XRTGEAR_ALIGNER_BOTTOMCENTE	下中央
XRTGEAR_ALIGNER_TOPLEFT	上左
XRTGEAR_ALIGNER_MIDDLELEFT	中央左
XRTGEAR_ALIGNER_BOTTOMLEFT	下左

XmNxrtGearAlignBaseline が True の場合は、XmNxrtGearAlignerAlignment を上書きします。

XmNxrtGearAlignBaseline　Boolean　True　CSG

ラベルのベースラインをcontrolウィジェットのベースラインと揃えるかどうか指定します。このリソースは、ラベルと control が XmLabel、XmText、XmXrtGearCombo のサブクラスでなければ無視されます。

`XmNxrtGearAlignBaseline` が `True` の場合は、`XmNxrtGearAlignerAlignment` を置き換えます。

XmNxrtGearPosition　int　XmLAST_POSITION　CSG

リスト内でウィジェットの(0 から開始する)位置を指定します。この値を変更すると、ウィジェットが managed された場合、そのウィジェットの視覚的な位置も変更します。0 または `XmLAST_POSITION` にセットすると、ウィジェットはリストの最初または最後に移動します。

XmNxrtGearResizeHeight　Boolean　False　CSG

行の高さが変更された時、control も縦方向にリサイズされるかどうか指定します。複数の control でこのリソースがセットされている場合、それらはサイズに応じて縦方向にストレッチされます。

XmNxrtGearResizeWidth　Boolean　False　CSG

列幅が変更された時、control も横方向にリサイズされるかどうか指定します。

6.8.2 Aligner に継承されるリソース

Resource	Inherited From
XmNaccelerators	Core
XmNancestorSensitive	Core
XmNbackground	Core
XmNbackgroundPixmap	Core
XmNborderColor	Core
XmNborderPixmap	Core
XmNborderWidth	Core
XmNbottomShadowColor	XmManager
XmNbottomShadowPixmap	XmManager
XmNchildren	Composite
XmNcolormap	Core
XmNdepth	Core
XmNdestroyCallback	Core
XmNforeground	XmManager
XmNheight	Core
XmNhelpCallback	XmManager
XmNhighlightColor	XmManager
XmNhighlightPixmap	XmManager

Resource	Inherited From
XmNinitialFocus	XmManager
XmNinitialResourcesPersistent	Core
XmNinsertPosition	Composite
XmNmappedWhenManaged	Core
XmNnavigationType	XmManager
XmNnumChildren	Composite
XmNscreen	Core
XmNsensitive	Core
XmNshadowThickness	XmManager
XmNstringDirection	XmManager
XmNtopShadowColor	XmManager
XmNtopShadowPixmap	XmManager
XmNtranslations	Core
XmNtraversalOn	XmManager
XmNuserData	XmManager
XmNwidth	Core
XmNx	Core
XmNy	Core

6.9 手続きと方法

Aligner の手続きと方法をアルファベット順に解説します。

XmCreateXrtAligner()

Aligner ウィジェットを作成します。

```
Widget
XmCreateXrtAligner(
  Widget  parent,
  String  name,
  Arg     *arglist,
  int     argcount
)
```

parent は親。*name* は作成されるウィジェット名。*arglist* は *argcount* 引数のリストです。

XmIsXrtAligner()

ウィジェットが Aligner であれば `True` を返します。

```
Boolean
XmIsXrtAligner(
  Widget  widget
  )
```

XrtGearMrmInitialize()

UIL アプリケーションに使用するために、Mrm に XRT/gear ウィジェットを登録します。この手続きは、*MrmInitialize()*の後に呼び出して下さい。

```
void
XrtGearMrmInitialize(void)
```

XrtGearSetXmStringConverter()

XmRXmString リソース変換プログラムをセットし、XmString 値を作成します。

```
void
XrtGearSetXmStringConverter(void)
```

変換は、エンベッドされたフォント(タグ)を指定するために、次のシンタックスを使用します。

@font@　　フォントのタグは `XmNfontList`

@P@　　前のフォント

\@　　"@"とする

例:

```
For a @bold@free evaluation@P@ of any product
```

フォント・リストを設定

```
fixed,-*-times-bold-r-normal-*-140-*=bold
```

表示結果

For a **free evaluation** of any product

7

Outliner

Outliner について ■ Outliner の基本概念
ウィジェットと Xt Object の概要 ■ ツリーの作成
ツリーのトラバース ■ ツリーの再編成
ツリー表示のコントロール ■ 複数の列
ノードのカスタマイズ ■ ユーザ・インタラクション
Outliner ウィジェットの印刷 ■ パフォーマンス
リソースの一覧 ■ 手続きと方法
トランスレーションとアクション

7.1 Outliner について

Outliner ウィジェットを利用して、階層データを簡単にわかりやすく表示できます。図 41 は、Outliner を使用して、階層データを表示している例です。

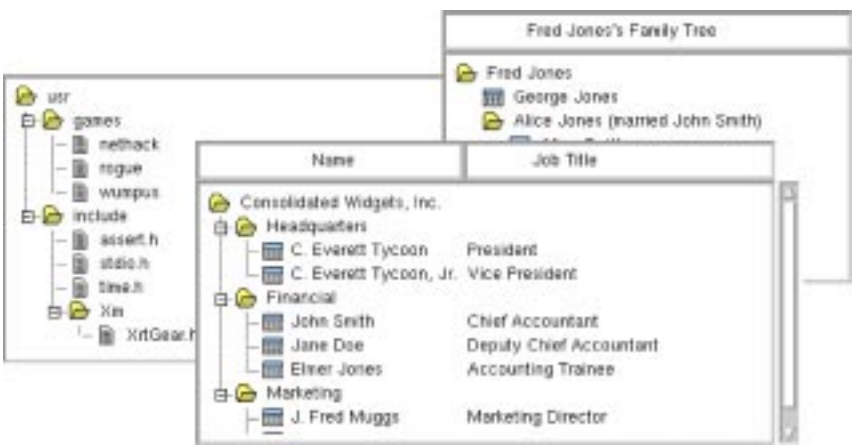


図 41　Outliner アプリケーションの例

7.1.1 データの階層と表示

Outliner ウィジェットで表示されるデータは、ツリー構造を持っています。各データは、フォルダ(*folder*)または、アイテム(*item*)のどちらかに相当します。

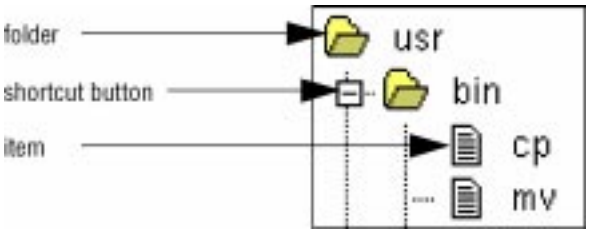
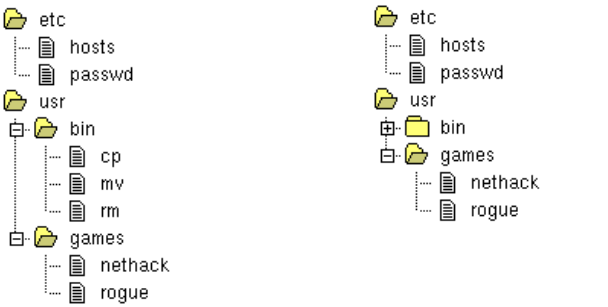


図 42　ツリー構造に表示されたデータ

エンドユーザは、フォルダに含まれているデータの表示･非表示をコントロールできます。データが表示されない場合は、フォルダは閉じられて(*collapsed されて*)います。フォルダをダブル･クリックまたは、ショートカット･ボタンをシングル･クリックして、このフォルダは開いたり閉じたりすることができます。

ショートカット･ボタンには、フォルダが閉じている状態かどうかを示す文字が表示されています。ボタンに + が表示されていたら、フォルダは閉じられています。 - が表示されていたら、フォルダ内のデータが表示されている状態です。



ショートカット･ボタンは -
データが表示されている

ショートカット･ボタンは +
フォルダは閉じられている

図 43　ショートカット･ボタン

7.1.2 その他のユーザ･インタラクション

Outliner は、その他に次のようなユーザ･インタラクション機能を備えています。

- ノードとフォルダの選択
- 列のリサイズ
- フォルダ･ノードの開閉
- ツリーのトラバース
- ノードのドラッグ&ドロップ
- 他のウィジェットからドラッグされたテキストの Outliner へのドロップ

尚、ユーザ･インタラクションは 7.10 に解説されています。

7.1.3 ツリーの作成

ツリーを作成するには、データ・ファイルからデータを読み込む方法が一番簡単です。次のコードは、データ・ファイルからツリーを作成する例です。

```
#include <stdio.h>
#include <stdlib.h>
#include <Xm/XrtGear.h>
#include <Xm/XrtOutliner.h>
#include <Xm/XrtNode.h>
#include <Xm/XrtNodeFolder.h>
#include <X11/Intrinsic.h>

int
main(int argc, char **argv)
{
  Widget      toplevel, outliner;
  XtAppContext        app_context;
  FILE        *infile;

  toplevel = XtAppInitialize(&app_context, "Sample", NULL, 0,
    &argc, argv, NULL, NULL, 0);
  outliner = XtVaCreateManagedWidget("outliner",
    xmXrtOutlinerWidgetClass, toplevel,
    NULL);
  infile = fopen("file.dat", "r");
  XrtGearNodeCreateTreeFromStream(outliner, infile, False,
    XRTGEAR_FOLDERSTATE_OPEN_ALL, NULL, NULL, NULL, NULL);
  XtRealizeWidget(toplevel);
  XtAppMainLoop(app_context);
}
```

この場合、データ・ファイルは、下記のようにアイテムに対してテキスト一行、階層レベルは<tab>という文字で指定されて構成されています。

```
etc
<tab>hosts
<tab>passwd
usr
<tab>bin
<tab><tab>cp
<tab><tab>mv
<tab><tab>rm
<tab>games
<tab><tab>nethack
<tab><tab>rogue
```

図 44 は、このデータ・ファイルから作成された Outliner です。

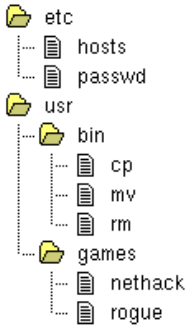


図 44　データ・ファイルから作成された Outliner

Outliner では、パス名のリストのデータ・ファイルからも簡単にツリーが作成できます。例えば、次のようなファイルからツリーを作成するとします。

```
etc/hosts
etc/passwd
usr/bin/cp
usr/bin/mv
usr/bin/rm
usr/games/nethack
usr/games/rogue
```

この場合は、*XrtGearNodeCreateTreeFromStream()*の呼び出しを下記のように変更します。

```
XrtGearNodeCreateTreeFromStream(outliner, infile, True,
XRTGEAR_FOLDERSTATE_OPEN_ALL, "/", NULL, NULL, NULL);
```

*XrtGearNodeCreateTreeFromStream()*に渡される三番目のパラメータを `True` にすると、パスの階層がデータの各行に含まれるようになります。五番目のパラメータは、インデントの文字を指定し、ここでは、スラッシュ(/)が使用されています。

作成されたツリーは、前のプログラムで作成された場合と同じで図 44 の通りです。

7.1.4　ここから先に進むには

Outliner ウィジェットを使用した、シンプルな階層データは、これらの基本事項を使用して簡単に表示できます。後に続く項では、複数の列の表示や、ツリーのソート/トラバース、ノード・スタイルのカスタマイズ等、更に高度な機能を解説します。

7.2 Outliner の基本概念

Outliner の高度な機能を利用するには、次の基本概念を理解しておいてください。

- Outliner ウィジェット、NodeFolder オブジェクト、Node オブジェクトを使用したツリーの作成法
- XrtString オブジェクトを使用してテキスト・ストリングや pixmap を記憶する方法
- XrtList オブジェクトを使用してエレメントのリストを記憶する方法

7.2.1 Outliner ウィジェットとツリー

Outliner ウィジェットは、階層のある情報をツリー構造に表示する Manager ウィジェットです。ツリー内のデータの各アイテムはノード(*node*)と呼ばれ、ノード・データ自身はノード・ラベル(*node label*)と呼ばれるテキスト・ストリングで構成されます。ノードが表示される領域は、ノード領域(*node area*)と呼ばれます。

ノードには、アイテム(*item*)とフォルダ・ノード(*folder node*)の二種類があります。これらの二つのノードの唯一の違いは、フォルダ・ノードは、複数の子を持つことができるという点です。XRT/gear は、ノードに二つの Xt Object を定義します。一つはアイテムである Node オブジェクト、もう一つはフォルダ・ノードである NodeFolder オブジェクトです。NodeFolder オブジェクトは Node オブジェクトのサブクラスですので、NodeFolder は、Node で定義されているリソースを使用することができます。

Outliner ウィジェットは、ツリーのルート・ノードとしての役割を果たし、NodeFolder オブジェクトのように処理されますが、それ自身は表示されません。

図 45 は、これらのウィジェットやオブジェクトを表示し、このツリーの考え方を示しています。Outliner は Manager ウィジェットですが、アイテムまたはフォルダ・ノード以外の子ウィジェットを持つことはできません。

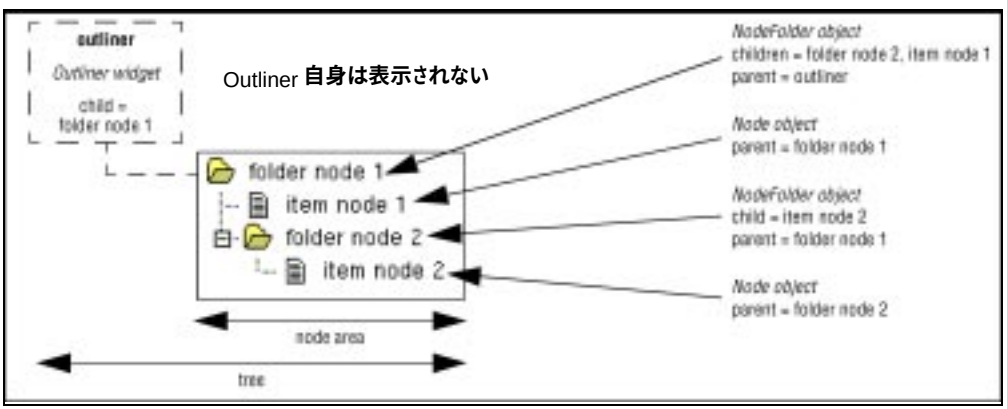


図 45　ツリーとノード

7.2.2 XrtString オブジェクト

XrtString オブジェクトは、データを String、XmString、XrtGearIcon の pixmap として記憶します。

String を記憶する XrtString オブジェクトを作成するには、*XrtGearStringCreateCharString()*を呼び出してください。

```
XrtGearObject mystring;

mystring = XrtGearStringCreateCharString("my string");
```

Outliner ウィジェットでは、XrtString は XmNxrtGearLabel に使用され、ノードの一つのラベルを指定します。尚、ノード・ラベルは、7.8.1 に解説されていますので参照してください。

XmString を記憶する XrtString オブジェクトを作成するには、*XrtGearStringCreateXmString()*を呼び出してください。

```
XrtGearObject mystring;
XmString xmstring;

xmstring = XmStringCreate("my string", XmFONTLIST_DEFAULT_TAG);
mystring = XrtGearStringCreateXmString(xmstring);
XmStringFre(xmstring);
```

XrtGearIcon の pixmap を記憶する XrtString オブジェクトを作成するには、*XrtGearStringCreateIconString()*を呼び出してください。

XRT/gear には、XrtString オブジェクトを操作する方法や手続きが提供されています。詳細は、13.2 を参照してください。

7.2.3 XrtList オブジェクト

XrtList オブジェクトを使用すると、同じサイズのエレメントのリストを簡単に記憶することができます。XrtList オブジェクトを作成するには、*XrtGearListCreate()*を作成してください。リストにエレメントを追加するには、*XrtGearListAppend()*を呼び出します。下記は、この方法を利用した XrtList オブジェクトの例です。

```
XrtGearObject list, label;
Widget newnode;
Widget folder;
...
list = XrtGearListCreate(sizeof(XrtGearObject));
label = XrtGearStringCreateCharString("label1");
XrtGearListAppend(list, (XtPointer)&label);
label = XrtGearStringCreateCharString("label2");
XrtGearListAppend(list, (XtPointer)&label);
label = XrtGearStringCreateCharString("label3");
XrtGearListAppend(list, (XtPointer)&label);
```

このコードは、XrtString オブジェクトのリストを作成し、このリストに三つのエレメントを追加しています。次のコードでは、リストの最初のオブジェクトを取り出しています。

```
myxrtstr = *(XrtGearObject *)XrtGearListGetItem(list, 0);
```

XmNxrtGearLabel リソースの値として使用される場合、XrtList オブジェクトを利用して同じノードに対する複数のラベルを指定することができます。この時、XrtList オブジェクトは必ず XrtStrings のリストとしてください。尚、ノード・ラベルは 7.8.1 に解説されています。

XRT/gear には、下記のように XrtList オブジェクトを操作する手続きや方法が提供されています。

XrtGearListDelete()	リストのエレメントを削除する
XrtGearListGetItemCount()	リストのエレメントの数を取得する
XrtGearListSort()	リストをソートする

XrtList の手続きと方法は、13.1 に解説されています。

7.3 ウィジェットと Xt Object の概要

Outliner widget	Include Files:	*<Xm/XrtOutliner.h>*
	Class Name:	XmXrtOutliner
	Class Hierarchy:	Core → Composite → Constraint → XmManager → XmXrtOutliner
	Class Pointer:	xmXrtOutlinerWidgetClass
Node object	Include Files:	*<Xm/XrtNode.h>*
	Class Name:	XmXrtNode
	Class Hierarchy:	Object → XmXrtNode
	Class Pointer:	xmXrtNodeObjectClass
NodeFolder object	Include Files:	*<Xm/XrtNodeFolder.h>*
	Class Name:	XmXrtNodeFolder
	Class Hierarchy:	Object → XmXrtNode → XmXrtNodeFolder
	Class Pointer:	xmXrtNodeFolderObjectClass
NodeStyle object	Include Files:	*<Xm/XrtNodeStyle.h>*
	Class Name:	XmXrtNodeStyle
	Class Hierarchy:	Object → XmXrtNodeStyle
	Class Pointer:	xmXrtNodeStyleObjectClass
Column object	Include Files:	*<Xm/XrtColumn.h>*
	Class Name:	XmXrtColumn
	Class Hierarchy:	Object → XmXrtColumn
	Class Pointer:	xmXrtColumnObjectClass

7.4 ツリーの作成

次の三つの方法でツリーを作成することができます。

- ツリーのデータ･ファイルから
- ストリング･バッファから
- 直接、*XtVaCreateWidget()*と *XtCreateWidget()*を使用して

7.4.1 データ･ファイルのフォーマット

データ･ファイルの各行は、ツリーの一つのノードを定義し、各行の始まりにくる＜tab>は、ツリーの階層を表します。下記は、ツリーのデータ･ファイルの一例です。

```
item1
folder
<tab>item2
<tab>item3
<tab>subfolder
<tab><tab>item4
item5
```

この例では、`item1`、`folder`、`item5` は先頭に<tab>がないので、ルートである Outliner ウィジェットの子となります。また、`item2`、`item3`、`subfolder` は、`folder` の子であり、`item4` は `subfolder` の子になります。

デフォルトでは、子がないノードはアイテムとして扱われます。子がないフォルダを作成する時は、ノード名に`(F)`というストリングを追加してください。

```
item1
folder1(F)
item2
```

その他、フォルダの状態を示す C、O、I、A を指定することができます。

```
C XRTGEAR_FOLDERSTATE_CLOSED
O XRTGEAR_FOLDERSTATE_OPEN_FOLDERS
I XRTGEAR_FOLDERSTATE_OPEN_ITEMS
A XRTGEAR_FOLDERSTATE_OPEN_ALL
```

例:

```
folder1(F,C)
```

データ･ファイルの特別文字

ツリーのデータ･ファイルでは、" ("は、*open escape character*(オープン･エスケープ文字)、") "は、*close escape character*(クローズ･エスケープ文字)として呼ばれ、この括弧内に囲まれる情報は特別な指示として処理されます。また、カンマも特別文字として扱われ、ノードの複数のラベルを分ける役割を果たします。

ノード名にこれらの特別文字を含ませる場合は、" \ "(バック･スラッシュ)を前に置いて下さい。

```
\(bracketed name)
```

上の例では、`(bracketed name)`がノードとして定義されています。

ノード名にバック・スラッシュを含ませるには、バック・スラッシュを二つ続けて下さい。特別文字でない文字にバック・スラッシュが付いている場合には、無視されます。

7.4.2 データ・ファイルからツリーを作成する

データ・ファイルからツリーを作成するには、*XrtGearNodeCreateTreeFromStream()*を呼び出します。

```
infile = fopen("file2.dat", "r");
XrtGearNodeCreateTreeFromStream(outliner, infile, False,
XRTGEAR_FOLDERSTATE_OPEN_ALL, "-", NULL, "{", "}");
```

*XrtGearNodeCreateTreeFromStream()*には、次の八個のパラメータが必要です。

- 親の Outliner ウィジェットを指す一番目のパラメータ
- データ・ファイルを指定し、既に開かれているファイル記述子を指す二番目のパラメータ
- 各ノードに完全な階層が提供されているかどうか指定する三番目のパラメータ
- ツリーのフォルダの初期状態を指定する四番目のパラメータ(フォルダの状態は 7.10.3 に解説されています。)
- (階層を表す)インデント文字を指定する五番目のパラメータ。このパラメータが `NULL` の場合は、tab 文字がインデントを表します。
- ノードの複数のラベルを区分する文字を指定する六番目のパラメータ。複数の列が定義されていなければ、複数のラベルの内、最初のラベルだけが表示されます。(列は、7.8 に解説されています。)このパラメータが `NULL` の場合は、カンマが代りに使用されます。
- open escape character(オープン・エスケープ文字)を指定する七番目のパラメータ。`NULL` の場合は、"(" が使用されます。
- close escape character(クローズ・エスケープ文字)を指定する八番目のパラメータ。`NULL` の場合は、")" が使用されます。

*XrtGearNodeCreateTreeFromStream()*の三番目のパラメータが `True` の場合は、下記のデータ・ファイルの例のように、各ノードに完全な階層が提供されます。

```
users/dave/bin
users/dave/lib
users/keith/bin
users/keith/lib
users/fang
```

インデントを表すストリング(五番目のパラメータ)がスラッシュとしてセットされると、*XrtGearNodeCreateFromStream()*によって、次のツリーが作成されます。

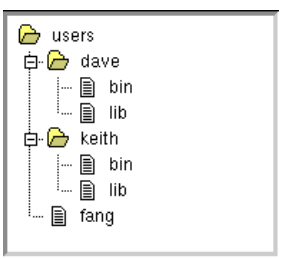


図 46　データ・ファイルから作成される Outliner

フォルダ･ノードは必要時に作成されます。例えば、次の行が読み込まれると、users と dave というフォルダ･ノードが、bin というアイテム･ノードと一緒に作成されます。

```
users/dave/bin
```

このフォルダ･ノードは　`users/dave/lib`

が読み込まれると、再度使用されます。

*XrtGearNodeCreateFromStream()*の三番目のパラメータが `False` の場合、ファイルのインデントのレベルから階層が推測されます。例えば、次のデータ･ファイルでは、七番目のパラメータが `False` にセットされると、(tab 文字がインデントのストリングとして認識され)図 46 のツリーを作成します。

```
users
<tab>dave
<tab><tab>bin
<tab><tab>lib
<tab>keith
<tab><tab>bin
<tab><tab>lib
<tab>fang
```

サブ･ツリーの作成

*XrtGearNodeCreateTreeFromStream()*は、既存のツリーに対するサブ･ツリーを作成することができます。既存のツリーのノードを一番目のパラメータとして渡して下さい。

```
XrtGearNodeCreateTreeFromStream(node, infile, False,
XRTGEAR_FOLDERSTATE_OPEN_ALL, NULL, NULL, NULL, NULL);
```

新しいサブ･ツリーは、*node* によって指されているノードの子として作成されます。

Outliner ウィジェットまたは NodeFolder オブジェクト以外が *XrtGearNodeCreateTreeFromStream()*に渡されると、"root"というフォルダが作成され、ファイルで指定されたツリーと、渡されたウィジェットの間に配置されます。

リソース･ファイルでデータ･ファイルを指定

リソース･ファイルまたはビルダーツール(GUI 構築ツール)からデータ･ファイルとそれに関連する特別文字を指定するには、下記のように `XmNxrtGearTreeData` リソースをセットします。

```
*.xrtGearTreeData: test.dat FOLDERSTATE_OPEN_ALL - DEFAULT { }
```

データ･ファイルのフォーマットは、*XrtGearNodeCreateFromStream()*に渡される場合と同じです。

`XmNxrtGearTreeData` のリソース･ストリングは、スペースで区切られる(一つまたは複数の)シンタックスで構成されます。次の順番に来るシンタックス･エレメントがサポートされています。

filename folderstate indent separator open_escape closed_escape

一番目のエレメント、*filename* は、ツリーのデータ･ファイル名です。(この例では、データ･ファイル名は `test.dat` です。) `XmNxrtGearTreeData` が、*XtSetValues()*の呼び出しやビルダー･ツールでセットされ、*filename* が `NULL` の場合は、既存のノードは破棄されます。

その後に続くエレメントには次のような役割があります。

- *indent* は、(階層を表す)インデント文字を指定します。DEFAULT にセットされると、tab 文字が使用されます。
- *separator* は、ノードの複数ラベルを区分する文字を指定します。複数の列が定義されていない場合は、一番目のラベルだけが表示されます。DEFAULT にセットされると、カンマが使用されます。
- *open_escape* は、オープン・エスケープ文字を指定します。DEFAULT の場合は、"(" が使用されます。
- *closed_escape* は、クローズ・エスケープ文字を指定します。DEFAULT の場合は、")" が使用されます。

これらのエレメントはオプションですが、一つのエレメントを指定する場合は、必ずそれに続くエレメントも全て指定してください。(例: *separator* を指定する場合は、*pathtype* と *indent* も必ず指定します。これらは DEFAULT にしておくこともできます。)

プログラムで、XmNxrtGearTreeData を使用することもできます。

```
XtVaSetValues(outliner,
XmNxrtGearTreeData, "test.dat - DEFAULT { }",
NULL);
```

このリソースの値は、リソース・ファイルで使用される場合と同じフォーマットを持つストリングです。

7.4.3 文字ストリングからツリーを作成

Outliner ウィジェットでは、*XrtGearNodeCreateTreeFromBuffer()*を呼び出して文字ストリングからツリーを作成することができます。

```
static char data[] =
  "item1\n\
  folder\n\
  \titem2\n\
  \titem3\n\
  \tsubfolder\n\
  \t\titem4\n\
  item5";

XrtGearNodeCreateTreeFromBuffer(outliner, data, False,
XRTGEAR_FOLDERSTATE_OPEN_ALL, "-", NULL, "{", "}");
```

*XrtGearNodeCreateTreeFromBuffer()*の二番目のパラメータは、ツリー・データを持つ文字ストリングです。このストリングは、*XrtGearNodeCreateTreeFromStream()*に使用される場合と同じフォーマットの複数行のストリングです。その他のパラメータは、全て *XrtGearNodeCreateTreeFromStream()*のパラメータと同様です。

7.4.4 ツリーを直接作成する

*XtVaCreateManagedWidget()*によって親の Outliner ウィジェット、*XtVaCreateWidget()*によってツリーのノードを作成し、ツリーを直接作成することができます。次の例では、簡単なツリーをこの方法で作成しています。

```
outliner = XtVaCreateManagedWidget("outliner",
  xmXrtOutlinerWidgetClass, toplevel,
  NULL);
folder = XtVaCreateWidget("folder",
  xmXrtNodeFolderObjectClass, outliner,
  XmNxrtGearLabel, XrtGearStringCreateCharString("Folder"),
  NULL);
item1 = XtVaCreateWidget("item",
  xmXrtNodeObjectClass, outliner,
  XmNxrtGearLabel, XrtGearStringCreateCharString("Item1"),
  NULL);
item2 = XtVaCreateWidget("item2",
  xmXrtNodeObjectClass, folder,
  XmNxrtGearLabel, XrtGearStringCreateCharString("Item2"),
  NULL);
```

上のコードで次のツリーが作成できます。



図 47　直接作成されたツリー

XmNxrtGearLabel リソースは、NodeFolder または Node オブジェクトに表示されるテキストを指定します。ここの例のテキストは、XrtString 型で、一つのラベルを定義しています。

7.4.5 ツリーの保管

Outliner ウィジェットには、ツリーの内容を保管する方法が二つあります。

- データ･ファイル
- バッファ(String 型)

どちらの方法でも、ツリーは、7.4.1 に解説されているデータ･ファイルのフォーマットで書かれます。

ツリーをファイルに書く

ツリーをファイルに保管するには、*XrtGearNodeWriteTreeToStream()*を呼び出してください。

```
Widget    outliner;
FILE      *outfile;
int       length;
Boolean   ok;

ok = XrtGearNodeWriteTreeToStream(outliner, outfile, False,
  False, "/", NULL, NULL, NULL, &length);
```

この方法に渡されるパラメータは、下記の点を除けば、*XrtGearNodeCreateTreeFromStream()*に渡されるパラメータと同じです。

- 二番目のパラメータは、ツリーが書き込まれる、開かれたファイルを指定。
- 四番目のパラメータは、書き込まれるツリーの一部として親ノードも含ませるかどうかを指定する Boolean。`True` であれば、親ノードが含まれます。(このパラメータが `True` で親ノードが Outliner ウィジェットであれば、*XtName()*で返される Outliner ウィジェット名がファイルに書き込まれます。)
- 最後のパラメータは、書かれたファイルの長さ(byte)を返す。(オプション)

書き込みが成功すると、この方法は `True` を返し、エラーが起きると `False` を返します。

ツリーをバッファに書く

ツリーをバッファに保管するには、*XrtGearNodeWriteTreeToBuffer()*を呼び出します。この方法は、Outliner データ・ファイルのフォーマットで、ツリーをテキスト・ストリングとして書きます。

```
Widget   outliner;
char     buffer[256];
int      length;

XrtGearNodeWriteTreeToBuffer(outliner, buffer, False,
  False, "/", NULL, NULL, NULL, &length);
```

この方法は、下記の点を除けば、*XrtGearNodeWriteTreeToStream()*と同じです。

- この方法は、テキスト・ストリングが記憶される内部メモリのポインタを返す(このメモリは、いつでも上書き可能)。
- 二番目のパラメータは、`NULL` でなければ、テキスト・ストリングが書かれるバッファを指定。独自のバッファを提供すると、テキスト・ストリングは、プログラムによって上書きされない。
- 最後のパラメータは、`NULL` でなければ、バッファの長さを byte で返す。

7.5 ツリーのトラバース

XRT/gear では、作成されたツリーをトラバースするリソースや手続きが幾つか提供されています。これらを利用して、ツリーの特定なラベルを検索したり、ノードを移動するために取得したりすることができます。(ツリーの再編成は、7.6 に解説されています。)

7.5.1 ノードの子のリスト

特定なノードや Outliner ウィジェット自身の子のリストを取り出すには、XmNxrtGearNodeChildList の値を取得します。その後、次の例のように、そのリストから特定な子を取り出します。

```
XrtGearObject     list;
Widget        folder, firstchildnode;

XtVaGetValues(folder,
  XmNxrtGearNodeChildList, &list,
  NULL);
firstchildnode = *(Widget *)XrtGearListGetItem(list, 0);
```

XmNxrtGearNodeChildList リソースをセットすることはできません。

XmNxrtGearNodeChildList の戻り値は、ノードの子が表示されている順に並べられた XrtList オブジェクトです。尚、XrtList オブジェクトは 7.2.3 に詳細が解説されています。

7.5.2 カレント・ノード

現在フォーカスがあるツリー・ノードは、カレント・ノード(*current node*)と呼ばれます。カレント・ノードは、Outliner ウィジェットの XmNxrtGearNodeCurrent 値を取得することにより知ることができます。

通常、カレント・ノードはエンドユーザによって指定されます。カレント・ノードを独自に指定するには、XmNxrtGearNodeCurrent をセットするかまたは、*XrtGearNodeTraverseTo()*を呼び出します。カレント・ノードが指定されていない場合は、XmNxrtGearNodeCurrent は NULL です。

ノード変更のコールバック

ユーザ・アクションや *XrtGearNodeTraverseTo()*によるカレント・ノードの変更があった時に呼び出されるコールバックを指定することができます。(但し、XmNxrtGearNodeCurrent の変更は、コールバックを呼び出しません。) XmNxrtGearNodeCurrentChangedCallback リソースで指定されるリストにコールバックを追加してください。

これらのコールバックには、次の構造体が渡されます。

```
typedef struct {
  XrtGearReason   reason;     /* read-only */
  XEvent      *event;     /* read-only */
  Widget      container;  /* read-only */
  Widget      previous_node;  /* read-only */
  Widget      new_node;
  Boolean     doit;
} XrtGearNodeCurrentChangedCallbackStruct;
```

previous_node と *new_node* は、前のカレント・ノードと新しいカレント・ノードを指定します。*container* は親の Outliner ウィジェットです。カレント・ノードが変更されないようにするには、*reason* が XRTGEAR_REASON_NODE_CHANGE_BEGIN の時に *doit* を False にします。

次は、NodeCurrentChanged コールバックの例です。このコールバックは、新しいカレント・ノードに定義されているラベルの数をプリントします。

```
void
changeCB(Widget outliner, XtPointer client_data,
XrtGearNodeCurrentChangedCallbackStruct *call_data)
{
  XrtGearObject label;

  if (call_data->reason != XRTGEAR_REASON_NODE_CHANGE_BEGIN) {
    return;
  }
  if (call_data->new_node == NULL) {
    return;
  }
  XtVaGetValues(call_data->new_node,
    XmNxrtGearLabel, &label,
    NULL);
  if (XrtGearListIsList(label)) {
    fprintf (stderr, "node has %d labels\n",
      XrtGearListGetItemCount(label));
  } else {
      fprintf (stderr, "node has one label\n");
  }
}
```

7.5.3 トラバースの方法

XRT/gear には、ツリーのトラバースに関する次の方法が提供されています。

XrtGearNodeGetChild(node, n)	ノードの n 番めの子を返します。
XrtGearNodeGetFirstInTree(root)	ツリー・ルートの最初の子を返します。
XrtGearNodeGetFirstNonCollapsedInTree(root)	ツリー・ルートの閉じられていない最初の子を返します。(閉じられているフォルダ・ノードの子孫の場合、そのノードは閉じられている(*collapsed*)状態です。この状態の解説は、7.10 にあります。)
XrtGearNodeGetLastInTree(root)	ノードの階層リストにある最後の子を返します。
XrtGearNodeGetLastNonCollapsedInTree(root)	ノードの階層リストにある、最後の閉じられていない子を返します。
XrtGearNodeGetLevel(root, node)	あるノードの、ルートに対応するレベル(世代)を返します。
XrtGearNodeGetNextInTree(root, node)	階層リストで、指定したノードの次に来るノードを返します。

XrtGearNodeGetNextNonCollapsedInTree(root, node)	階層リストで、指定したノードの次に来る、閉じられていないノードを返します。
XrtGearNodeGetNumChildren(node)	(`XmNxrtGearNodeChildList` で指定される)、ウィジェットまたはオブジェクトの子の数を返します。
XrtGearNodeGetPreviousInTree(root, node)	階層リストで、指定したノードの前に来るノードを返します。
XrtGearNodeGetPrviousNonCollapsedInTree(root, node)	階層リストで、指定したノードの前に来る、閉じられていないノードを返します。
XrtGearNodeGetVisibility(node)	ノードの現在の表示状態(visibility)を返します。
XrtGearNodeGetWidgetParent(node)	ウィジェット(通常 Outliner)に到達するまで階層を上がり、そのウィジェットを返します。
XrtGearNodeIsAncestor(root, ancestor, node)	指定されたノードが他のノードの先祖であれば `True` を返します。
XrtGearNodeIsInCollapsedBranch(root, node)	指定されたノードが、閉じられた枝(branch)にある場合に `True` を返します。

図 48 は、ノードの階層リストと上記の幾つかの方法の関係性を示します。

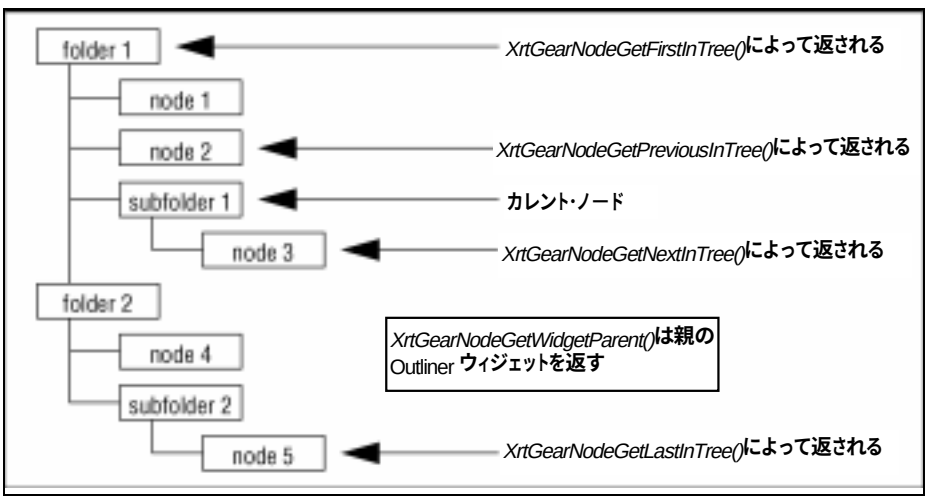


図 48　階層リストの方法

これらの方法は、7.13.2 に詳細が解説されています。

7.6 ツリーの再編成

XRT/gear では、ノードを移動したりソートすることにより、ツリーを再編成することができます。

7.6.1 ノードの移動

ノードは、*XrtGearNodeSetParent()*を呼び出すことにより、ツリー内で再編成させることができます。この方法は、ノードまたはサブ･ツリーの新しい親を定義します。

```
XrtGearNodeSetParent(subtree, newparent, sibling);
```

新しい親は、NodeFolder オブジェクトまたは Outliner ウィジェットです(新しい親が NULL の場合はサブ･ツリーはカットされます)。この親に他の子がある場合は、*sibling*(兄弟)で指定されるノードの直前にこのノードまたはサブ･ツリーが追加されます。図 49 は、*XrtGearNodeSetParent()*がどのようにノードを配置するか示しています。

sibling が指定されていない場合は、ノードまたはサブ･ツリーは、子のリストの最後に追加されます。(7.6.3 に解説されている)XmNxrtGearAutoSort が True にセットされた場合は、*sibling*(兄弟)は無視され、子は、ソートされた位置に挿入されます。

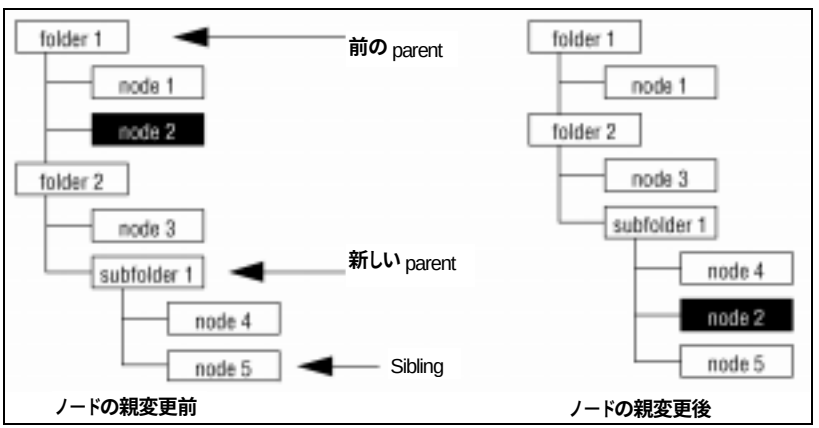


図 49 ノードの再編成

移動のコールバック

*XrtGearNodeSetParent()*が呼び出されたり、エンドユーザがノード/サブ･ツリーを移動/コピーする度に呼び出される、移動のコールバックを指定することができます。このコールバックを利用して、ツリーの動きを制限したり監視したりすることができます。

移動のコールバックを登録するには、XmNxrtGearNodeMoveCallback リソースで指定されるリストにコールバックを追加してください。これらのコールバックには、次の構造体が渡されます。

```
typedef struct {
  XrtGearReason   reason;      /* read-only */
  XEvent      *event;      /* read-only */
  unsigned char   operation;   /* read-only */
  Widget      node;        /* read-only */
  Widget      old_parent;  /* read-only */
  Widget      new_parent;  /* read-only */
  Widget      new_next;
  Boolean     doit;
} XrtGearNodeMoveCallbackStruct;
```

operation メンバは、XmDROP_MOVE(移動操作)または XmDROP_COPY(コピー操作)です。

*XrtGearNodeSetParent()*が呼び出されると、*old_parent* は移動されたノードの古い親、*new_parent* は新しい親になります。*new_next* は新しい sibling(兄弟)です。

new_next と *doit* は、*reason* が XRTGEAR_REASON_NODE_MOVE_BEGIN の時に修正することができます。移動の操作を中止するには、*doit* を False にします。*new_parent* の子リスト内の *node* の位置を変更するには、(そのノードの直後に来る sibling を指定する)*new_next* を変更して下さい。

7.6.2 ノードの削除

ノードを削除するには、*XtDestroyWidget(node)*または、*XrtGearNodeDestroy()*呼び出してください。

7.6.3 ノードのソート

Outliner ウィジェットの XmNxrtGearAutoSort リソースは、True にセットされると、sibling に対してノードが作成される順にソートされます。ソートには、XmNxrtGearNodeCompareProc でセットされる手続きが使用されます。

デフォルトでは、XmNxrtGearAutoSort は False でソートは自動的には行われません。

ツリーまたはサブ･ツリーを直接ソートするには、*XrtGearNodeSortTree()*を呼び出して下さい。

```
XrtGearNodeSortTree(tree);
```

デフォルトでは、アルファベット順に sibling のグループがソートされます。(図 50 参照)

ノードに複数のラベルがある場合は、それらが結びついて一つのストリングとして処理されます。(一つのノード内でラベルをソートするには、*XrtGearListSort()*を使用します。)

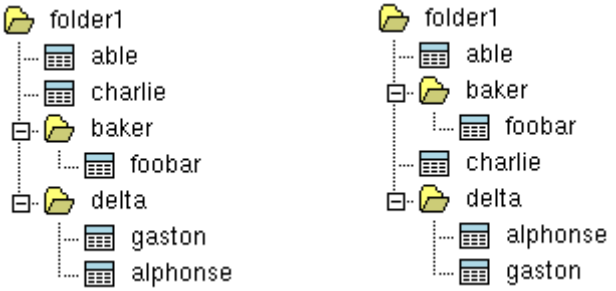


図 50　ツリーのソート

XmNxrtGearNodeMoveCallback で指定される移動のコールバックは、ツリーがソートされた時には呼び出されません。

ソートの方法

デフォルトでは、ソートはアルファベット順に行われます。これを変更するには、各ノードを比較する手続きを定義し、XmNxrtGearNodeCompareProc の値がその手続きを指すようにセットしてください。

ノード比較の手続きには、比較する二つのノードのポインタである XtPoinster 型のパラメータ二つが必要です。また、この手続きは、次のような整数を返します。

- 最初のアイテムが二番目より大きい場合は正の整数
- 二つのアイテムが同じ場合は 0(ゼロ)
- 最初のアイテムが二番目より小さい場合は負の整数

次の例は、アルファベット順に逆にノードをソートするようにセットしています。

```
int
my_node_compare_proc(XtPointer item1, XtPointer item2)
{
  char    buffer1[1024], buffer2[1024];
  Widget  node1 = *(Widget *)item1;
  Widget  node2 = *(Widget *)item2;

  assert(XmIsXrtNode(node1));
  assert(XmIsXrtNode(node2));
  XrtGearNodeCvtLabelToString(node1, ",", buffer1);
  XrtGearNodeCvtLabelToString(node2, ",", buffer2);
  return (strcmp(buffer2, buffer1));
}
...
XtVaSetValues(outliner,
  XmNxrtGearNodeCompareProc, my_node_compare_proc,
  NULL);
```

7.7 ツリー表示のコントロール

Outliner ウィジェットでは、表示に関する下記のような事項をコントロールすることができます。

- 高さと幅のサイズの決まり
- マージン
- 色とフォント
- ノードのインデントと縦方向の間隔
- 表示･非表示
- 各ノードの表示
- フォーカスのハイライト
- リサイズ

各ノードの表示は、ノード･スタイルを定義することによってもコントロールできます。(7.9 参照)

7.7.1 高さと幅

Outliner ウィジェットは、ウィジェットの高さと幅を定義する時に次の三つのコンセプトを使用します。

- 画面に表示されているウィジェットのウィンドウの実際の(*actual*)高さと幅
- ウィジェット全体を表示するのに必要な仮想の(*virtual*)高さと幅
- アプリケーションによって要求される、理想的な(*preferred*)高さと幅

実際(actual)の高さと幅

Outliner ウィジェットの実際の高さと幅は `XmNheight` と `XmNwidth` リソースで指定されます。この高さと幅は、図 51 にあるように、列ラベルやスクロールバーもウィジェットの一部として含んだウィジェット全体の高さと幅を指します。(列ラベルは 7.8.5、スクロールバーは 7.10.6 に解説されています。)

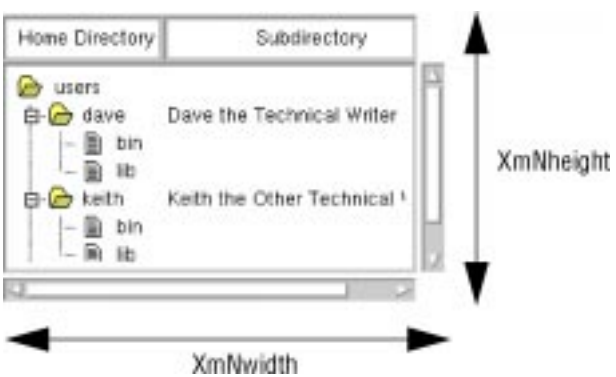


図 51 Outliner ウィジェットの `XmNheight`、`XmNwidth`

仮想(virtual)の高さと幅

Outliner ウィジェットの仮想の高さと幅は、XmNxrtGearHeightVirtual と XmNxrtGearWidthVirtual リソースによって指定されます。これらのリソースは、ウィジェット全体が表示されると仮定された場合の高さと幅を表し、値は read-only です。

仮想の高さと幅は、通常、縦と横のスクロールバーによって使用されます。尚、スクロールバーは、7.10.6 に解説されています。

要求される(preferred)高さ

Outliner ウィジェットに要求される高さは、次の三つを合計して計算されます。

- 列ラベルの高さ(列ラベルの詳細は 7.8.5 参照)
- 横向きのスクロールバーの高さ(但し、managed されている場合。スクロールバーの詳細は 7.10.6 参照)
- ノードの領域に必要な高さ

ノード領域に要求される高さは、次のようにして、XmNxrtGearHeightSizingPolicy と XmNxrtGearHeightPreferredCount リソースの値によって計算されます。

- XmNxrtGearHeightSizingPolicy が XRTGEAR_HEIGHT_PIXEL にセットされると、ノード領域に必要な高さは、XmNxrtGearHeightPreferredCount の値から得られます。
- XmNxrtGearHeightSizingPolicy が XRTGEAR_SIZE_CHAR にセットされると、
  ノード領域に必要な高さは、XmNfontList で指定されるリスト内の一番高いフォントの高さに XmNxrtGearHeightPreferredCount の値を乗じた値から計算されます。
- XmNxrtGearHeightSizingPolicy が XRTGEAR_SIZE_NODE にセットされた場合は、次のように計算されます。
  - XmNxrtGearHeightPreferredCount で指定されるノード数に XmNxrtGearNodeHeight の値を乗算
  - 次の値を加算
    (XmNxrtGearHeightPreferredCount-1)*XmNxrtGearNodeSpacing
  - 最後に、XmNhighlightThickness の値を 2 倍した数を加算

  上記により、ノードの表示に必要なサイズに、各ノードの間隔、フォーカス･ハイライトに必要なスペースを追加したサイズが計算されます。
- XmNxrtGearHeightSizingPolicy が XRTGEAR_SIZE_VARIABLE にセットされると、閉じられていないノードの高さが全て合計されます。(ノードは、閉じられているフォルダ･ノードの子孫である場合は、閉じられている(*collapsed*)状態です。)

Outliner ウィジェットは、表示される時は、(それが可能であれば)要求される高さになるようにサイズが決定されます。不可能な時は、ノード領域の高さが範囲に応じて小さくなります。ウィジェットの高さが、要求される高さより大きい場合は、それに従ってサイズが大きくなります。

要求される(preferred)幅

Outliner ウィジェットに要求される幅は、次の三つを合計して計算されます。

- 縦向きのスクロールバーの幅(但し、managed されている場合。スクロールバーの詳細は 7.10.6 参照)
- ノードの領域に必要な幅

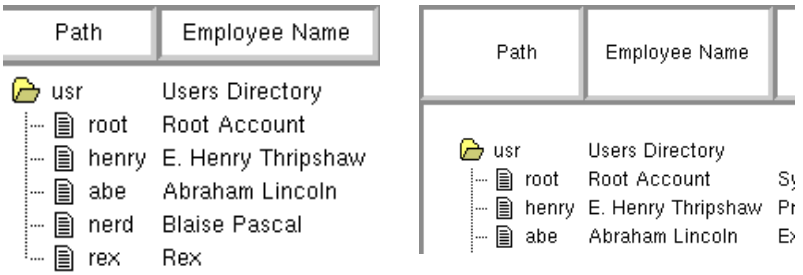
ノード領域に要求される幅は、次のようにして、XmNxrtGearWidthSizingPolicy と XmNxrtGearWidthPreferredCount リソースの値によって計算されます。

- XmNxrtGearWidthSizingPolicy が XRTGEAR_SIZE_PIXEL にセットされると、ノード領域に必要な幅は、XmNxrtGearWidthPreferredCount の値から得られます。
- XmNxrtGearWidthSizingPolicy が XRTGEAR_SIZE_CHAR にセットされると、XmNfontList で指定されるリストの一番始めのフォントを使用した、XmNxrtGearWidthChar で指定される文字が計測の基準の単位になります。ノード領域に必要な幅は、この単位に XmNxrtGearWidthPreferredCount の値を乗じた値から計算されます。(デフォルトでは、XmNxrtGearWidthChar は文字、です。)
- XmNxrtGearWidthSizingPolicy が XRTGEAR_SIZE_COLUMN にセットされた場合は、列幅の平均値に XmNxrtGearWidthPreferredCount の値を乗算します。(デフォルトでは、Outliner ウィジェットは一つの列しか持ちません。複数の列を作成する場合は、7.8 を参照してください。
- XmNxrtGearWidthSizingPolicy が XRTGEAR_SIZE_VARIABLE にセットされると、閉じられていないノードの幅が全て合計されます。
- XmNxrtGearWidthSizingPolicy が XRTGEAR_SIZE_MAXIMUM にセットされた場合は、全てのノードの幅の合計が計算されます。

Outliner ウィジェットは、表示される時は、(それが可能であれば)要求される幅になるようにサイズが決定されます。不可能な時は、ノード領域の幅が範囲に応じて小さくなります。ウィジェットの幅が、要求される幅より大きい場合は、それに従ってサイズが大きくなります。(複数の列が定義されている場合は、最後の列が大きくなります。尚、複数の列は、7.8 に解説されています。)

7.7.2 マージン

XmNxrtGearMarginHeight と XmNxrtGearMarginWidth は、ウィジェットとその枠、列ラベルとその枠の間の距離を pixel で指定します。図 52 は、このリソースによるマージンの大きさの例です。



高さと幅のマージン=5　　高さと幅のマージン=20

図 52　マージン幅と高さ

7.7.3　色とフォント

Outliner ウィジェットに使用される前景色とフォントは、XrtColumn と XrtNodeStyle オブジェクトによってコントロールされます。列ラベルの前景色とフォントの設定は、7.8.7 に解説されています。また、ノード・スタイルとノードの前景色及びフォントの設定は、7.9 に解説されています。

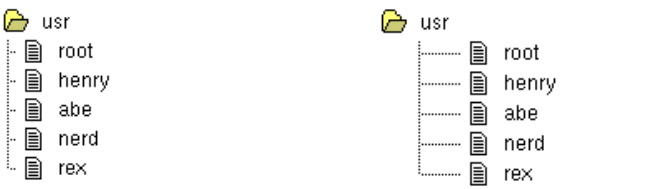
Outliner ウィジェットの背景色を変更するには、標準の Motif の `XmNbackground` リソースをセットしてください。

7.7.4　スペース

各ノードのレベルを表すインデント幅やノード間の縦の間隔をコントロールすることができます。

ノード・レベルのインデント

`XmNxrtGearNodeIndent` リソースは、フォルダ・ノードと子を区分するインデントの分量を pixel でコントロールします。デフォルトは 20pixel です。図 53 は、異なるインデントの場合です。



`XmNxrtGearNodeIndent`=10 pixel　　`XmNxrtGearNodeIndent`=50 pixel

図 53　ノードのインデント

ノード間の縦の間隔

Outliner ウィジェットの `XmNxrtGearNodeSpacing` は、上下に配置されるノードの上と下の間隔を指定します。この値は、フォーカス・ボックスを描画するのに充分なスペースがいつも確保されるように、必ず、`XmNhighlightThickness` と同じ値かそれ以上に設定してください。

デフォルトでは、XmNxrtGearNodeSpacing は XmNhighlightThickness の値の 2 倍です。

7.7.5 表示･非表示(visibility)

ノードの表示･非表示をコントロールする幾つかの機能を利用して、次の事が実行できます。

- 表示されているノードのリストを取り出す
- 一番上に表示されるノードをセットする
- あるノードを表示する
- あるノードの、現在の表示状態を取得する

表示ノードのリストの取得

表示されているノードの現在のリストを取り出すには、Outliner ウィジェットの XmNxrtGearNodeVisibleList リソースの値を取得します。このリストは、XrtList オブジェクトです。(XrtList オブジェクトは、7.2.3 に解説されています。)

一番上に表示されるノードをセット

あるノードをノード領域の一番上に配置するには､XmNxrtGearNodeTop リソースが、該当する Node または NodeFolder オブジェクトを指すようにセットします。

```
Node mytop;

XtVaSetValues(outliner,
  XmNxrtGearNodeTop,  mytop,
  NULL);
```

XmNxrtGearNodeTop を使用して、表示されているノード･リストの最初のノードを簡単に取得することができます。(XmNxrtGearNodeVisibleList の値を取得し、それの最初のエレメントを調べて取得するより簡単です。)

指定ノードを表示

あるツリー･ノードを表示するには、*XrtGearNodeMakeVisible()*を呼び出します。

```
Node node;
XrtGearNodeMakeVisible(node);
```

ノードが部分的または全体的に隠れていても、そのノードが完全に表示されるよう、そのツリーが再配置されます。

指定ノードの表示状態を取得

あるノードの現在の表示状態を取得するには、*XrtGearNodeGetVisibility()*を呼び出してください。

```
Node node;
XrtGearNodeGetVisibility(node);
```

返された XmVisiblity の値は、XmVISIBILITY_UNOBSCURED 、XmVISIBILITY_PARTIALLY_OBSCURED、XmVISIBILITY_FULLY_OBSCURED のどれかです。

7.7.6 テキストの切り取り

Outliner ウィジェットの XmNxrtGearTextHorizClipPolicy は、表示できるスペースに対してテキストが長すぎる場合に、どのようにテキストを切るか指定します。図 54 は、"this is a very long name"というストリングがこのリソースの値によってどのように表示されるか示しています。

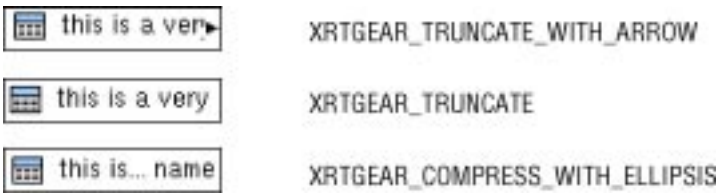


図 54 テキストの切り取り

デフォルトは、XRTGEAR_TRUNCATE_WITH_ARROW です。

7.7.7 ノードの高さ

ツリーの各ノードの高さは、Outliner ウィジェットの XmNxrtGearNodeHeight によって指定します。デフォルトでは、各ノードの高さは一番高いノード・スタイルの高さです。

7.7.8 フォーカスのハイライト

ノードがカレント・ノードになると、フォーカス・ハイライトがそのノードの周囲に描かれます。フォーカス・ハイライトのサイズは、XmNhighlightThickness によって pixel で指定できます。このリソースのデフォルト値は 2pixel です。

図 55 は、フォーカス・ハイライトの例です。

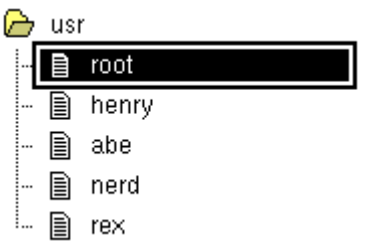


図 55 フォーカス・ハイライト

フォーカス・ハイライトは二色で描かれます。外側は XmNhighlightColor で指定された色、内側は、XmNbackground で指定される背景色で描画されます。XmNhighlightThickness の値が奇数の場合は、外側のハイライトが内側のハイライトより 1pixel 分太く描かれます。

次のコードは、ハイライトの幅と色をセットしています。

```
outliner = XtVaCreateManagedWidget("outliner",
  xmXrtOutlinerWidgetClass,    toplevel,
  XmNhighlightThickness,  4,
  XmNxrtGearNodeSpacing,  4,
  XtVaTypedArg, XmNhighlightColor,
    XmRString, "blue", strlen("blue")+1,
  NULL);
```

XmNhighlightThickness の値が XmNxrtGearNodeSpacing の値より大きい場合は、フォーカス･ハイライトは隣接するノードの上に重なります。

7.7.9 リサイズ

高さと幅のリサイズは、XmNxrtGearHeightResizePolicy と XmNxrtGearWidthResizePolicy によってコントロールされます。これらのリソースの値は、次の通りです。

XRTGEAR_WIDGET_RESIZE_VARIABLE	(親にリサイズが許可されている場合は)XmNxrtGearHeightSizingPolicy または XmNxrtGearWidthSizingPolicy によって決定された、要求される(preferred)サイズにリサイズされる。デフォルト値
XRTGEAR_WIDGET_RESIZE_GROW_ONLY	(親にリサイズが許可されている場合は)リサイズによってウィジェットのサイズが大きくなる場合にのみリサイズする。それ以外の場合は、リサイズしない。
XRTGEAR_WIDGET_RESIZE_FIXED	親からのリクエストがあった時以外は、リサイズしない。

リサイズのコールバック

Outliner ウィジェットがリサイズされる時に呼び出されるコールバックのリストを指定することができます。XmNxrtGearResizeCallback リソースにコールバック名を追加してください。リサイズのコールバックには次の構造体が渡されます。

```
typedef struct {
  XrtGearReason   reason; /* read-only */
  Xevent      *event; /* read-only */
  Dimension       width;
  Dimension       height;
} XrtGearResizeCallbackStruct;
```

width と *height* は新しい幅と高さで、編集可能です。

次のコードは、リサイズ･コールバックの例です。

```
void
resizeCB(Widget outliner, XtPointer client_data,
XrtGearResizeCallbackStruct *call_data)
{
  XrtGearDisplayPolicy hpolicy, vpolicy;

  if (call_data->width < 200) {
    vpolicy = XRTGEAR_DISPLAY_NEVER;
  } else {
    vpolicy = XRTGEAR_DISPLAY_AS_NEEDED;
  }
  if (call_data->height < 200) {
    hpolicy = XRTGEAR_DISPLAY_NEVER;
  } else {
    hpolicy = XRTGEAR_DISPLAY_AS_NEEDED;
  }
  XtVaSetValues(outliner,
    XmNxrtGearScrollBarHorizDisplayPolicy, hpolicy,
    NULL);
  XtVaSetValues(outliner,
    XmNxrtGearScrollBarVertDisplayPolicy, vpolicy,
    NULL);
}
```

このコールバックは、ウィジェットが 200pixel 以下にリサイズされた時は、スクロールバーを非表示にします。(スクロールバーは 7.10.6 に解説されています。)

7.7.10 カーソルの表示

Outliner ウィジェットの `XmNxrtGearCursor` は、ウィジェット上のカーソルの表示をコントロールします。このリソースの値は、Cursor 型で、リソースのデフォルト値は、矢印  です。

`XmNxrtGearCursorResizeWidth` は、列幅をリサイズする時に使用されるカーソルを指定します。このリソースの値も Cursor 型で、デフォルト値は、列のリサイズのポインタで です。

カーソルの追跡

次のようにして、カーソルがノードの端や内側、外側に移動した時に、形が変更されるようにプログラミングすることができます。

- カーソル追跡のコールバックを定義し、`XmNxrtGearCursorTrackingCallback` リソースで定義されるコールバックのリストに追加する。
- `XmNxrtGearCursorTracking` を `True` にセットして、カーソル追跡の開始を指示する。

XmNxrtGearCursorTrackingCallback で指定されるリストの各コールバックには、次の構造体が渡されます。

```
typedef struct {
  XrtGearReason   reason;     /* read-only */
  XEvent      *event;     /* read-only */
  Widget      object;     /* read-only */
  Cursor      old_cursor; /* read-only */
  Cursor      new_cursor;
} XrtGearCursorTrackingCallbackStruct;
```

reason は、コールバックを呼び出したマウスの動きを示し、次の内のどれかです。

XRTGEAR_REASON_CURSOR_EDGE_LEFT	カーソルは左端に移動
XRTGEAR_REASON_CURSOR_EDGE_RIGHT	カーソルは右端に移動
XRTGEAR_REASON_CURSOR_INSIDE	カーソルはノードまたはラベルの内側に移動
XRTGEAR_REASON_CURSOR_LEAVE	カーソルはノードまたはラベルを離れる

どの場合も、*new_cursor* は、カーソルの新しい形です。コールバック手続きでこの値を変更してカーソルの形を変更することができます。

7.8 複数の列

ツリーのどのノードにも複数のラベルを定義することができます。通常、ツリーが表示される時は、各ノードの最初のラベルだけが表示されます。

ツリーに複数のラベルを表示するには、親の Outliner ウィジェットに関連する列を新たに定義します。作成された各列が、ラベルを一つづつ表示します。

7.8.1 複数ラベルの定義

NodeFolder または Node オブジェクトに表示されるテキストは、XmNxrtGearLabel リソースで指定されます。このリソースがセットされた場合、その値は、XrtString オブジェクトまたは XrtList オブジェクトのどちらかです。XrtString オブジェクトは、ノードに一つのラベルが定義された時に使用され、(XrtString のリストも含む)XrtList オブジェクトは、複数のラベルが定義された時に使用されます。

次のコードは、XrtList オブジェクトを作成し、ノードの XmNxrtGearLabel リソースの値としてセットします。

```
XrtGearObject list, label;
Widget newnode;
Widget folder;
...
list = XrtGearListCreate(sizeof(XrtGearObject));
label = XrtGearStringCreateCharString("label1");
XrtGearListAppend(list, (XtPointer)&label);
label = XrtGearStringCreateCharString("label2");
XrtGearListAppend(list, (XtPointer)&label);
label = XrtGearStringCreateCharString("label3");
XrtGearListAppend(list, (XtPointer)&label);
newnode = XtVaCreateWidget("newnode",
xmXrtNodeObjectClass, folder,
XmNxrtGearLabel, list,
NULL);
```

このコードでは、*label1*、*label2*、*label3* という三つのラベルを定義しています。ツリーが表示されると、最初のラベル、*label1* のみが表示されます。(但し、複数の列が定義されていない時)

または *XrtGearNodeCutStringToLabel()*を使用することもできます。

```
XrtGearObject list, label;
Widget node;
list = XrtGearNodeCvtStringToLabel("label1,label2,label3", ",");
```

この方法は、ノード、ラベルに変換されるストリング、複数のラベルを区分する区切り文字の三つのパラメータを取ります。上の場合、*label1*、*label2*、*label3* が一つのノードに定義されています。

XmNxrtGearLabel は、*XrtGearNodeCreateTreeFromStream()* または *XrtGearNodeCreateTreeFromBuffer()*が呼び出されると、各ノードに自動的にセットされます。

XrtString または XrtList オブジェクトの詳細は 7.2 に解説されています。

XrtList オブジェクトとデータ･ファイル

*XrtGearNodeCreateTreeFromStream()*または *XrtGearNodeCreateTreeFromBuffer()*を使用してツリーを作成する時、ラベルをカンマで区切ることにより、一つのデータ･ファイルで複数のラベルを指定することができます。

```
item1
folder
<tab>item2
<tab>multilabel1,multilabel2,multilabel3
<tab>subfolder
<tab><tab>item4
item5
```

folder の二番目の子は、三つのラベルを持つノードです。これらのラベルには、XrtList オブジェクトが作成されており、ノードの XmNxrtGearLabel の値になっています。

カンマ以外の区切り文字を使用するには、*XrtGearNodeCreateTreeFromStream()*または*XrtGearNodeCreateTreeFromBuffer()*に渡される 6 番目のパラメータをセットします。

ラベルとリソースファイル

`XmNxrtGearLabel` の値がリソース･ファイルで指定される時は、ラベルとして使用する文字ストリングを提供してください。

```
*.node1.xrtGearLabel: mystring
```

複数のラベルを指定する場合は、それらをカンマで区切って下さい。

```
*.node1.xrtGearLabel: multilabel1,multilabel2,multilabel3
```

7.8.2 新しい列の作成

XRT/gear は、XrtColumn という特別なオブジェクトを定義し、列を作成します。列を作成するには、Outliner ウィジェットを親として XrtColumns オブジェクトを作成してください。

```
XtVaCreateWidget("column1", xmXrtColumnObjectClass, outliner, NULL);
```

これにより、ツリーが表示された時に自動的に列も表示されます。図 56 は、複数の列の表示例です。

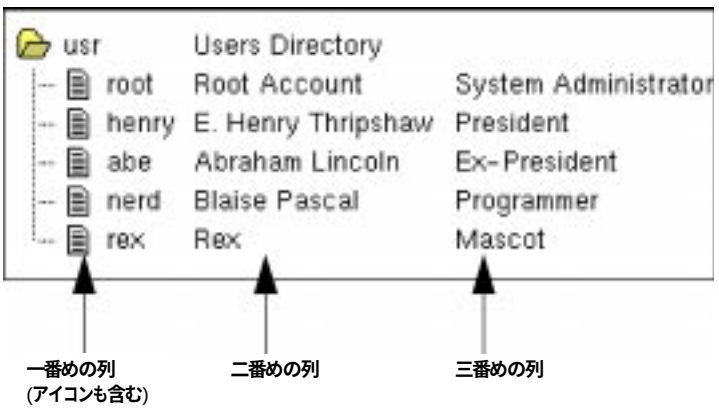


図 56 複数列のツリー表示

最初に作成される列はツリーの階層とそのノードの最初のラベルを表示します。次に続く列は、その次のラベルを表示します。列数よりラベルの数が多い場合は、あまっているラベルは表示されません。

Outliner ウィジェットは、自動的にデフォルトの列を作成しますが、独自の列を作成した場合は、デフォルトの列は破棄されます。従って、列を作成する場合は、必要な列は全て作成しなければなりません。(作成された列を全て破棄した場合は、新しくデフォルトの列が自動的に作成されます。)

Outliner ウィジェットに定義されている列のリストを取得するには、`XmNxrtGearColumnList` リソースの値を取り出して下さい。このリストは、XrtColumn オブジェクトのエレメントで構成される XrtList オブジェクトです。`XmNxrtGearColumnList` は read-only ですのでセットしたり作成することはできません。

7.8.3 列の順序

通常、列は作成された順に表示されますが、この順序を変更する場合は、*XrtGearOutlinerMoveColumns()*を呼び出してください。この方法は、指定された列を指定された位置に移動します。

移動される列は、次の決まりに従って移動先の列の左側または右側に配置されます。

- 移動される列が移動先の列の直ぐ左にある時は、移動先の列の右側に配置
- 上記以外の場合は、移動された列は全て、移動先の列の左側に配置

図 57 は、列がどのように移動されるか示しています。

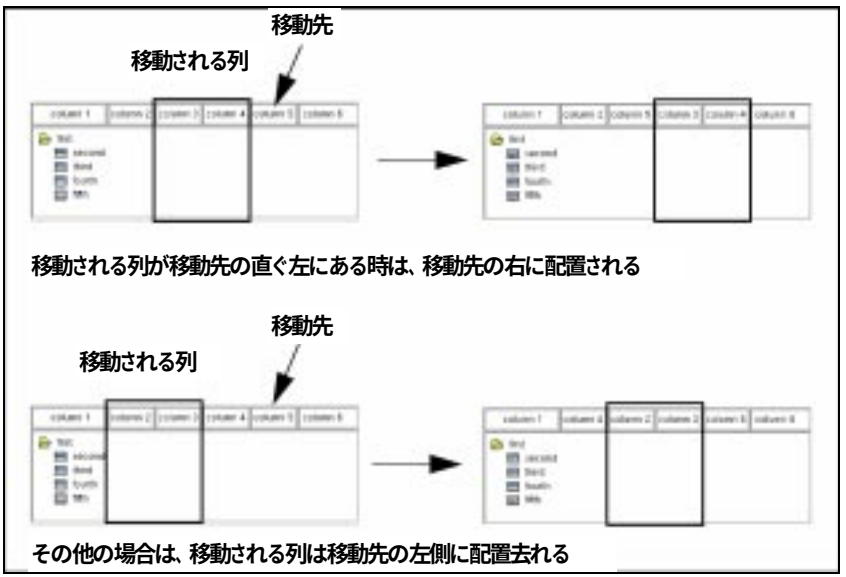


図 57　列の移動

次の例では、列 5 と列 6 を列 3 の前に移動しています。

```
int destination = 3;
int source = 5;
int num_to_move = 2;

XrtGearOutlinerMoveColumns(outliner, destination, source,
  num_to_move);
```

階層を示す一番左側の列のインデックスは 0 で、移動したり、移動先の列となることはできません。(この列は常に最初に表示されます。)

現在の列の順序

列の現在の順序は、XmNxrtGearColumnOrderList で指定します。このリソースは、表示される順に並んだ XrtColumn オブジェクトをエレメントとして持つ XrtList オブジェクトを返します。XmNxrtGearColumnOrderList は read-only ですので、作成したりセットすることはできません。(列の順序は、*XrtGearOutlinerMoveColumns()*で変更してください。)

エンドユーザも列が表示される順序を変更することができます。詳細は、7.10.10 を参照してください。

7.8.4 選択された列

エンドユーザによって現在選択されている列は、XmNxrtGearColumnSelected リソースの値を取得することにより得ることができます。

```
Widget outliner, currcolumn;

XtVaGetValues(outliner,
  XmNxrtGearColumnSelected, &currcolumn,
  NULL);
```

XmNxrtGearColumnSelected をセットして、現在選択されている列を変更することができます。

尚、選択等のユーザ･インタラクションの詳細は、7.10 を参照してください。

列の選択のコールバック

エンドユーザが列を選択した時に呼び出されるコールバックを指定できます。XmNxrtGearColumnSelectionCallback で定義されているリストにコールバックを追加して下さい。このリソースには次の構造体が渡されます。

```
typedef struct {
  XrtGearReason   reason;     /* read-only */
  XEvent      *event;     /* read-only */
  Widget      old_column; /* read-only */
  Widget      new_column;
  Boolean     doit;
} XrtGearColumnSelectionCallbackStruct;
```

old_column は前に選択されていた列、*new_column* はこれから選択されようとする列です。*reason* が XRTGEAR_REASON_SELECT_BEGIN の時に、この選択コールバックより次の事が行えます。

- *new_column* を変更して新しく選択される列を変更する
- *doit* を False にして、列選択を不可にする

7.8.5 列ラベル

各 XrtColumn オブジェクトには、ラベルを定義できます。ラベルを作成するには、XrtColumn オブジェクトの XmNxrtGearLabel リソースをセットしてください。

```
XtVaSetValues(column,
  XmNxrtGearLabel,
  XrtGearStringCreateCharString("col1label"),
  NULL);
```

Outliner ウィジェットの XmNxrtGearColumnLabelsShow リソースは、列ラベルの表示をコントロールします。デフォルトでは、このリソースは False にセットされており、ラベルは表示されません。

7.8.6 列のインデックス

列は整数値でインデックスが付けられます。次の例では、整数値から column ウィジェットを取得しています。

```
XrtGearObject     column_list;
int           column_index;
Widget        column, outliner;

XtVaGetValues (outliner,
      XmNxrtGearColumnList,&column_list,
      NULL);
column = *( Widget *)XrtGearListGetItem(column_list,
      column_index);
```

下記の例では、column ウィジェットのインデックスを取得します。

```
XrtGearObject     column_list;
int           column_index;
Widget        column, outliner;

XtVaGetValues(outliner,
      XmNxrtGearColumnList,&column_list,
      NULL);

column_index = XrtGearListFind(column_list,
(XtPointer)column);
```

7.8.7 列の表示のコントロール

Outliner ウィジェットでは、次のような列の表示に関する属性をコントロールできます。

- 列の位置
- 列ラベルの前景色と背景色
- 列ラベルのフォント
- ノードとラベルのテキストの行揃え
- 高さと幅
- 左と右のマージン
- 列とノードの領域の枠
- リサイズの動作とコールバック

列ラベルの前景色と背景色

列ラベルの前景色は XrtColumn オブジェクトの XmNforeground、背景色は XmNbackground をセットして変更することができます。デフォルトでは、この色は親の Outliner ウィジェットの前景/背景色より継承されます。

列ラベルのフォント

XrtColumn オブジェクトには、列ラベルに使用されるフォントをコントロールするリソースが幾つかあります。XrtColumn の XmNxrtGearFontListSelected は、列が選択された時に使用されるフォントを指定し、XmNfontList は、列が選択されていない時に使用されるフォントを指定します。デフォルトでは、これらのフォントは、Outliner ウィジェットの XmNfontList から継承されます。

ノードとラベルの行揃え

列内のノード・テキストの行揃えは、XrtColumn の XmNxrtGearNodeAlignment で指定されます。列ラベル内のテキストの行揃えは、XrtColumn の XmNxrtGearLabelAlignment で指定されます。どちらのリソースも、下記の値を取ります。

XRTGEAR_ALIGN_BEGINNING	テキスト領域の開始点に揃える
XRTGEAR_ALIGN_CENTER	テキスト領域の中央に揃える
XRTGEAR_ALIGN_END	テキスト領域の終了点に揃える

図 58 は、ノードとラベルの行揃えの変更例です。

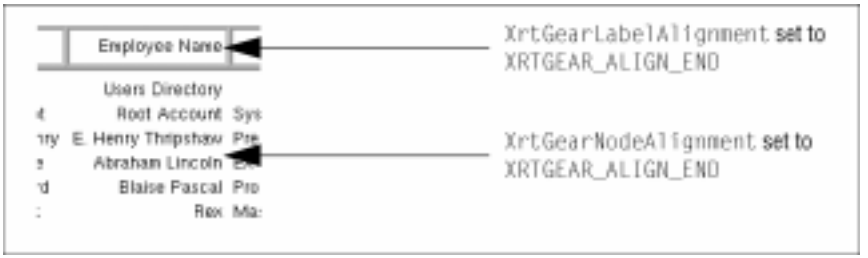


図 58　ノードとラベルの行揃え

XmNxrtGearNodeAlignment のデフォルト値は、XRTGEAR_ALIGN_BEGINNING、XmNxrtGearLabelAlignment のデフォルト値は XRTGEAR_ALIGN_CENTER です。

XmNxrtGearNodeAlignment は、階層を表す最初の列には影響せず、この列は常に左揃えで表示されます。

列の高さ

列の高さは XrtColumn オブジェクトの XmNheight で指定されます。このリソースは、列ラベルの表示に必要な高さに自動的にセットされ、通常はプログラムによって変更する必要はありません。

列幅

XrtColumn オブジェクトは、列幅を定義する際に次の二つの点を考慮します。

- 画面に表示される実際の(*actual*)幅
- アプリケーションによって要求される(*preferred*)幅

列の実際の幅は、XmNwidth によって指定されます。要求される(*preferred*)幅は、XrtColumn の XmNxrtGearWidthSizingPolicy と XmNxrtGearWidthPreferredCount の値から下記のようにして計算されます。

- XmNxrtGearWidthSizingPolicy が XRTGEAR_SIZE_PIXEL にセットされると、要求される(*preferred*)幅は、XmNwdith の値から取られます。
- XmNxrtGearWidthSizingPolicy が XRTGEAR_SIZE_CHAR にセットされると、XmNfontList で指定されるリストの一番始めのフォントを使用した、XmNxrtGearWidthChar で指定される文字が計測の基準の単位になります。この単位に XmNxrtGearWidthPreferredCount の値を乗じた値が計算されます。(デフォルトでは、XmNxrtGearWidthChar は M、XmNxrtGearWidthPreferredCount は 10 です。)

- XmNxrtGearWidthSizingPolicy が XRTGEAR_SIZE_LABEL_VARIABLE にセットされると、列ラベルの幅が、列に要求される(*preferred*)幅として使用されます。
- XmNxrtGearWidthSizingPolicy が XRTGEAR_SIZE_VARIABLE(デフォルト)にセットされると、列の一番広いラベルの幅が、要求される(*preferred*)幅として使用されます。

列が表示される時は、(それが可能であれば)要求される幅になるようにサイズが決定されます。不可能な時は、列幅が範囲に応じて小さくなります。ウィジェットの幅が、要求される幅より大きい場合は、それに従ってサイズが大きくなります。

XmNwidth が 0 の時は、列は非表示です。枠線も表示されず、エンドユーザがインタラクティブにリサイズすることもできません。

左と右のマージン

列ラベルの左端とラベル・テキストの開始点との間の距離は、XmNxrtGearMarginLeft によって指定されます。列ラベルの右端とラベル・テキストの終了点との間の距離は、XmNxrtGearMarginRight によって指定されます。図 59 は、これらのリソースを変更した例です。



図 59　列ラベルのマージン

デフォルトでは、これらのリソースは 2 にセットされています。

列ラベルの上下のマージンは、Outliner の XmNxrtGearMarginHeight と XmNxrtGearMarginWidth によってコントロールされます。(これらのリソースは 7.7.2 に解説されています。)

列とノード領域の枠

列ラベルとノード領域の枠は、XmNxrtGearBoarderLocation によって指定されます。このリソースによって、枠をどこに(列ラベル領域、ノード・ラベル領域、各列ラベルの周囲)描くか指定できます。XmNxrtGearBorderLocation の有効な値とその設定による表示の変更は図 60 を参照してください。

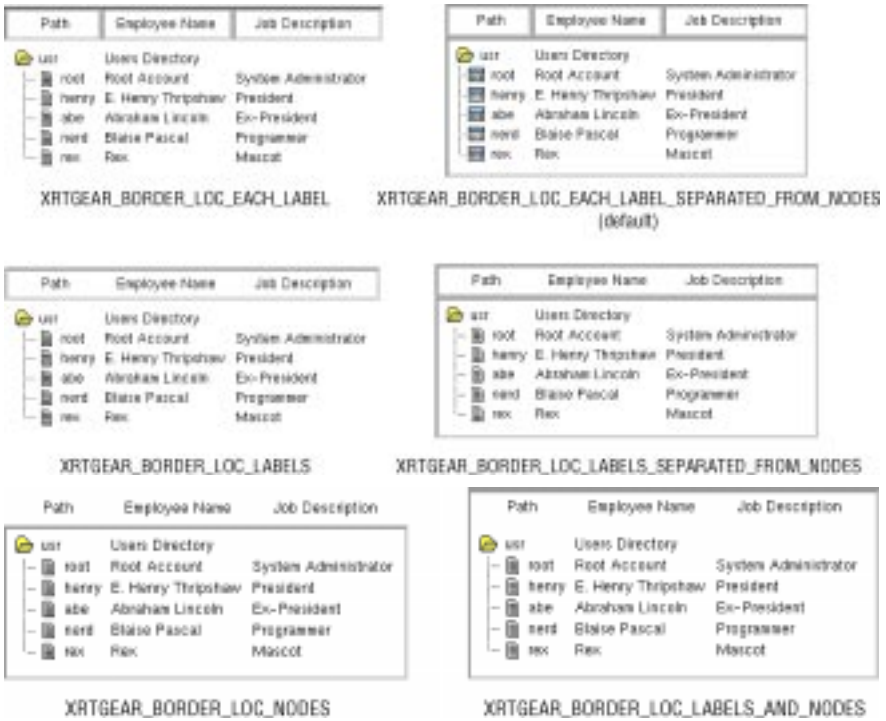


図 60　`XmNxrtGearBorderLocation` の値

枠線の種類は、Outliner の `XmNxrtGearBorderType` によって指定されます。有効な値は図 61 にある通りです。

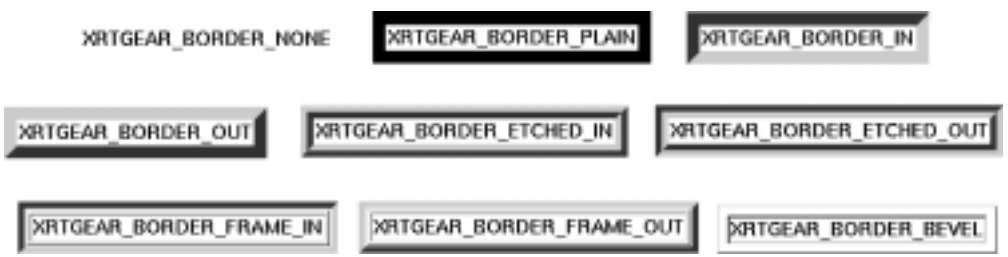


図 61　Outliner の枠線

枠線の幅は XmNxrtGearShadowThickness によって指定されます。

インタラクティブな列のリサイズ

XrtColumn には、エンドユーザによる列のリサイズをコントロールするリソースが三つあります。

- XmNxrtGearResizePolicy は、リサイズの許可･不許可を指定します。デフォルトでは True でリサイズが実行できます。
- XmNxrtGearWidthMinimum と XmNxrtGearWidthMaximum は、最小と最大のサイズを pixel で指定し、このサイズまでエンドユーザは列をリサイズできます。プログラムによって実行されるリサイズは、これらのリソース値の影響は受けません。これらのリソースのデフォルト値は、最小は 10、最大は 10000 です。

XmNxrtGearResizeCallback で定義されるリストにコールバックを追加することにより、列がリサイズされた時に呼び出されるコールバックを指定できます。各コールバックには次の構造体が渡されます。

```
typedef struct {
  XrtGearReason   reason;      /* read-only */
  XEvent      *event;      /* read-only */
  Dimension       width;
  Dimension       height;
  Dimension       old_width;  /* read-only */
  Dimension       old_height; /* read-only */
  Boolean     doit;
} XrtGearEnhancedSizeCallbackStruct;
```

列のリサイズは次のようにコントロールできます。

- 構造体の *height* と *width* メンバを変更し、新しい高さと幅の値を修正できます。*old_width* と *old_height* は、リサイズ前の幅と高さです。
- *doit* を False にセットすることにより、リサイズを制止できます。

XmNxrtGearShadowThickness が 0 であったり、XmNwidth が 0 であると、エンドユーザは列をリサイズできません。

また、上記のコントロールは *reason* が XRTGEAR_REASON_RESIZE_BEGIN の時にのみ有効です。

このコールバック構造体は、(Outliner の XmNxrtGearResizeCallback に使用される)XrtGearResizeCallbackStruct コールバック構造体と互換性があります。

7.9 ノードのカスタマイズ

Outliner ウィジェットには、ノードの表示をコントロールする幾つかのリソースと簡易手続きが提供されています。ノードの表示は、下記の事項が変更できます。

- ノード･ラベル
- 選択、非選択ノードの背景と前景色
- 選択、非選択ノードのフォント
- フォルダのアイコン
- フォルダと子を結ぶ線の種類

7.9.1 ノード･ラベル

ノードのテキストは、Node オブジェクトの `XmNxrtGearLabel` リソースによって定義できます。このリソースの値は、XrtString オブジェクト(一つのラベル)または XrtList オブジェクト(複数のラベル)のどちらかです。

尚、ノード･ラベルの操作は 7.8.1 に解説があります。

ノード･ラベルの編集

各列のノード･ラベルは個別に編集することができます。アプリケーションによってこれを検証するには、`XmNxrtGearNodeValidateCallback` を編集の前後に使用してください。または、特定のノードやフォルダが編集可能かどうかをマークし、ノードやフォルダの設定を上書きすることができます。さらに、outliner 全体に対して編集の可･不可をマークすることにより、ノードや列の設定、特定のノードやフォルダに付けられたマークを上書きすることができます。

セルが編集可とマークされていても、`XmNrtGearNodeEnterCellCallback` をトラバーサルの前後に使用して、そのセルに対するテキスト･ウィジェットのマッピングを許可したり不許可にすることができます。テキスト･ウィジェットにアクセスすることにより、色やフォント等をカスタマイズすることができます。

ノードの編集に使用されるウィジェットは、ノードの前景/背景色やフォントを使用します。

7.9.2 ノード･スタイル

Outliner ウィジェットでは、表示されるノードは、*node style* によってコントロールできます。特定なノードのスタイルは、NodeStyle オブジェクトによって指定できます。node style を使用すると、同じように表示されるノードのグループを簡単に作成することができます。

ノード･スタイルの作成

NodeStyle オブジェクトを作成するには、*XmCreateXrtNodeStyle()*を呼び出してください。

```
Widget outliner, nodestyle;

nodestyle = XmCreateXrtNodeStyle(outliner, "mystyle", NULL, NULL);
```

node style は、Outliner ウィジェットの子として作成してください。node style は、Outliner ウィジェットが破棄されると、一緒に破棄されます。

デフォルトの node style

Outliner ウィジェットが作成される時には、node style が自動的に作成され、そのツリーの *default node style* になります。ツリーの各ノードは、`XmNxrtGearNodeStyle` がセットされていなければ、自動的にこのノード・スタイルを使用します。

独自のデフォルトのノード・スタイルをセットするには、Outliner の `XmNxrtGearNodeStyleDefault` リソースを使用したいノード・スタイルにセットします。

```
Widget outliner, nodestyle;

XtVaSetValues(outliner,
  XmNxrtGearNodeStyleDefault, nodestyle,
  NULL);
```

既に定義されているデフォルトのノード・スタイルは、次の項で解説されています。

ノード・スタイルのリソース

下記の NodeStyle オブジェクトのリソースが、ノードの表示をコントロールします。

`XmNxrtGearBackgroundSelected`	選択されたノードの背景色
`XmNforeground`	非選択のノードの前景色
`XmNxrtGearForegroundSelected`	選択されたノードの前景色
`XmNfontList`	非選択のノードのフォント
`XmNxrtGearFontListSelected`	選択されたノードのフォント

他の NodeStyle リソースは、フォルダ・アイコン(7.9.3 参照)やフォルダと子を結ぶ線(7.9.4 参照)のスタイルをコントロールします。

次のコードは、これらのリソースをセットしている例です。

```
Widget outliner, nodestyle;

nodestyle = XmCreateXrtNodeStyle(outliner,
  "mystyle", NULL, NULL);
XtVaSetValues(nodestyle,
  XtVaTypedArg, XmNxrtGearBackgroundSelected,
  XmRString, "white", strlen("white")+1,
  NULL);
XtVaSetValues(nodestyle,
  XtVaTypedArg, XmNxrtGearFontListSelected,
  XmRString, "-*-terminal-bold-r-*-*-14-*-*-*-*_*-iso8859-*",
  strlen("-*-terminal-bold-r-*-*-14-*-*-*-*_*-iso8859-*")+1,
  NULL);
```

既に定義されているデフォルトのノード・スタイルは、これらのリソースに次の値を使用します。

- XmNbackground、XmNforeground、XmNfontList の値は親のウィジェットから継承される
- XmNxrtGearSelectedBackground の値は XmNbackground から継承される
- XmNxrtGearSelectedForeground の値は、XmNforeground から継承される
- XmNxrtGearFontListSelected の値は XmNfontList から継承される

ノード・スタイルをノードに定義する

ノード・スタイルを特定なノードに定義するには、ノードの XmNxrtGearNodeStyle をセットします。次の例では、mystyle という名のノード・スタイルを使用しています。

```
Widget outliner, node, nodestyle;

nodestyle = XrtGearNodeStyleFromName(outliner, "mystyle");
XtVaSetValues(node,
  XmNxrtGearNodeStyle, nodestyle,
  NULL);
```

この例では、*XrtGearNodeStyleFromName()*によって、NodeStyle オブジェクトが取り出されています。

アプリケーションに使用されるノード・スタイルのマスター・リストは、Outliner ウィジェットの XmNxrtGearNodeStyleList によって定義されます。

```
XrtGearObject stylelist;
XtVaGetValues(outliner,
  XmNxrtGearNodeStyleList, &stylelist,
  NULL);
```

返されたノード・スタイルのリストは XrtList オブジェクトで、このリストのエレメントは NodeStyle オブジェクトです。(XrtList オブジェクトは、7.2.3 に解説されています。)

7.9.3 アイコン

NodeStyle オブジェクトは、ツリーの描画時に画面に現れるフォルダ・アイコンを指定するリソースを定義します。フォルダ・アイコンには、下記のリソースがセットできます。

XmNxrtGearIconClosed	非選択状態の閉じられているフォルダを示すアイコン
XmNxrtGearIconClosedSelected	エンドユーザによって選択された閉じられているフォルダを示すアイコン
XmNxrtGearIconOpen	非選択状態の開いているフォルダを示すアイコン (XRTGEAR_FOLDERSTATE_CLOSED 以外の場合は、"開いている"状態です。)
XmNxrtGearIconOpenSelected	選択された開いているフォルダを示すアイコン
XmNxrtGearIconItem	非選択状態のアイテムを示すアイコン
XmNxrtGearIconItemSelected	選択されたアイテムを示すアイコン

各リソースは、XrtGearIcon 型の値のポインタを取ります。XrtGearIcon は次のように定義されます。

```
typedef struct {
  Pixmap icon;
  Pixmap icon_mask;
} XrtGearIcon;
```

icon は表示されるアイコンの pixmap、*icon_mask* はアイコンの図を切り取る(オプションの)マスクです。(clip マスクに関しては、Enhanced Label と Enhanced Pushbutton に定義される XmNxrtGearMask の解説(2.4)を参照してください。)

次のコードは、非選択状態の閉じたフォルダ・アイコンを定義しています。

```
Widget nodestyle;
Pixmap closed_pixmap, closed_pixmap_mask;
XrtGearIcon icon;

icon.icon = closed_pixmap;
icon.icon_mask = closed_pixmap_mask;
XtVaSetValues(nodestyle,
  XmNxrtGearIconClosed, &icon,
  NULL);
```

デフォルトのノード・スタイルの一部として定義されるアイコンは、図 62 の通りです。



図 62　デフォルトのフォルダ・アイコン

アイコンの間隔

各 NodeStyle オブジェクトは XmNxrtGearLayoutSpacing を定義し、ノードのアイコンとそのラベル・ストリングとの間の距離を pixel で指定します。



図 63 アイコンの間隔

XmNxrtGearLayoutSpacing のデフォルト値は 5 です。

XPM フォーマット

*XrtGearIconLoadFromXpmData()*と *XrtGearIconLoadFromXpmFile()*の簡易ルーチンは、XPM フォーマットのデータ・オブジェクトから XrtGearIcon 構造体を作成します。

*XrtGearIconLoadFromXpmData()*は、既存の XPM データ構造体からアイコンをロードします。

```
Widget    outliner;
XrtGearIcon   myicon;
String    my_xpm_data;
String    name;

XrtGearIconLoadFromXpmData(outliner, my_xpm_data, name,
     &myicon);
```

*XrtGearIconLoadFromXpmFile()*は、既存の XPM データ・ファイルからアイコンをロードします。

```
Widget    outliner;
XrtGearIcon   myicon;

XrtGearIconLoadFromXpmFile(outliner, "myicon.xpm", &myicon);
```

尚、XPM の詳細は、XPM の関連資料を参照してください。

7.9.4 オブジェクトを結ぶ線とショートカット・ボタン

NodeStyle オブジェクトには、フォルダとアイテムを結ぶ線やショートカット・ボタンの表示をコントロールするリソースがあります。

ショートカット・ボタンは、指定されたフォルダに付属する小さなボタンです。各ショートカット・ボタンには、フォルダが開いている(-)、閉じている(+)という状態を示す文字が付いています。

エンドユーザは、このショートカット・ボタンをクリックして、フォルダを閉じたり開いたりすることができます。図 64 は、このボタンの使用例です。

注意: 線やショートカット・ノードのリソースの変更は、そのノード自身ではなく、そのノードの子に影響します。

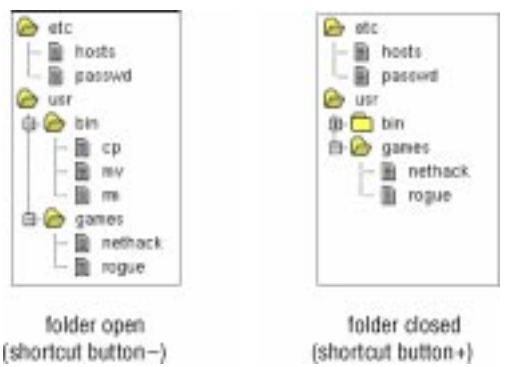


図 64　ショートカット・ボタン

接続線

下記は、フォルダとアイテムを結ぶ線の表示をコントロールする NodeStyle のリソースです。

XmNxrtGearLineColor	線の色を指定。デフォルトでは、XmNforeground から継承されます。

XmNxrtGearLineWidth	線幅を指定。デフォルト値は 1。
XmNxrtGearLineStyle	線のスタイルを指定。

XmNxrtGearLineStyle は、次の値にセットできます。

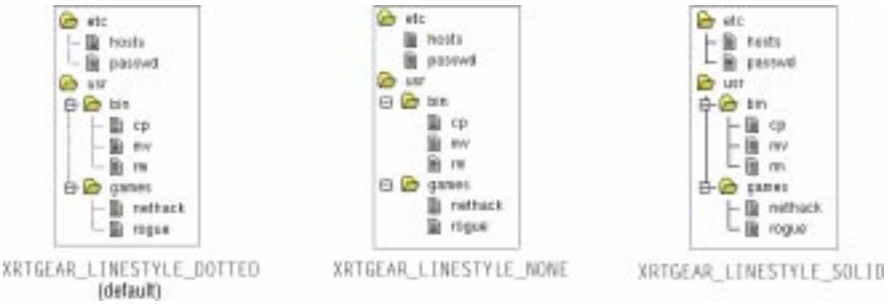


図 65　接続線のスタイル

ショートカット・ボタン

XmNxrtGearShortcutShow リソースは、ショートカット・ボタンを表示するかどうか指定します。デフォルトでは、このリソースは True で、ボタンは表示されます。

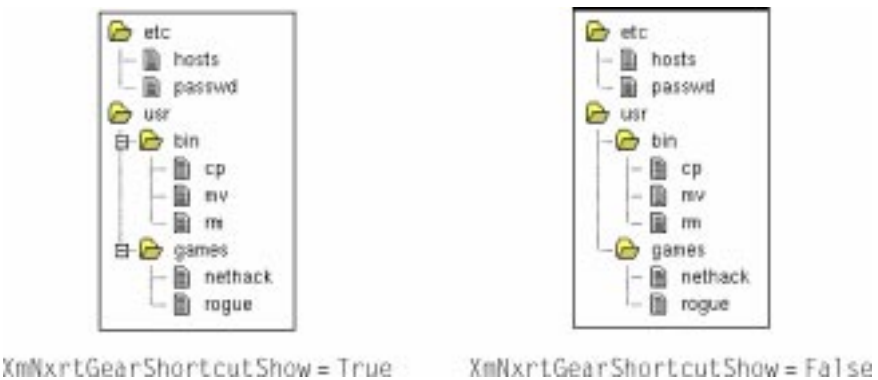


図 66　ショートカット・ボタン

ショートカット・ボタンの幅と高さは、XmNxrtGearShortcutSize リソースによって pixel で指定できます。デフォルト値は 9 pixel です。XmNxrtGearLineWidth が奇数の場合は XmNxrtGearShortcutSize も奇数値、XmNxrtGearLineWidth が偶数の場合は XmNxrtGearShortcutSize も偶数値にすると、良い結果が得られます。

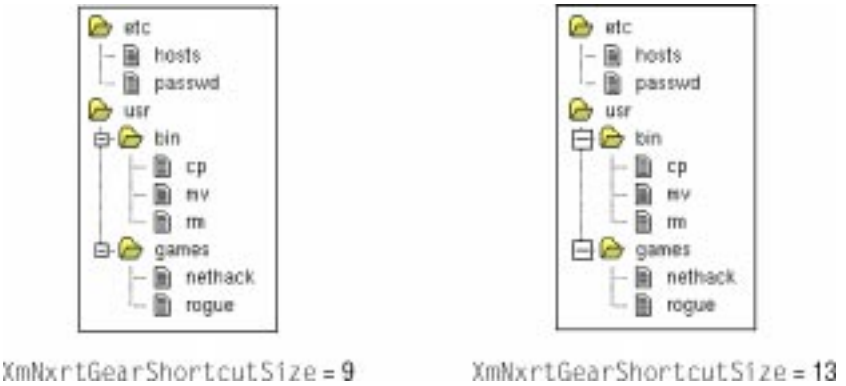


図 67　ショートカット・ボタンのサイズ

XmNxrtGearShortcutSize が表示するには大きすぎる(XmNxrtGearNodeHeight または XmNxrtGearNodeIndent リソース値より大きい)値にセットされた場合は、ショートカット・ボタンは表示されません。

7.9.5　ユーザ定義のデータ

ノードにデータを関連付けるには、ノードの XmNuserData リソースの値をセットします。

```
Node      node;
XtPointer mydata;

XtVaSetValues(node,
  XmNuserData, mydata,
  NULL);
```

Node(または NodeFolder)オブジェクトは、XmNuserData に指されているデータを解放しません。他の Motif ウィジェットの場合と同様に、データは必ず独自に解放してください。

7.10　ユーザ・インタラクション

XRT/gear では、次のようにエンドユーザが Outliner ウィジェットやそのオブジェクトとインタラクティブな操作を行うことができます。

- ノードやフォルダの選択
- 列のリサイズ
- フォルダ・ノードの開閉
- ツリーのトラバース
- ノードのドラッグ&ドロップ
- 列の再編成

図 68 は、デフォルトのユーザ・インタラクションとキー/マウスのイベントの一覧です。尚、Outliner ウィジェットとそのオブジェクトに定義されているトランスレーションとアクションの解説は、7.15 を参照してください。

7.10.1 選択

Select(Start)や Select(Add)アクションが実行された時、カレント・ノードには選択を示すマークが付きます。(デフォルトでは、エンドユーザがマウスの第一ボタンを押した時に起こります。)この選択の動作をコントロールするには、次の二つのリソースが使用できます。

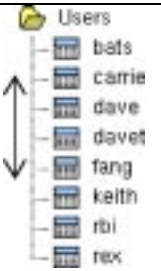
- 一度に幾つのノードを選択できるかどうかを指定する `XmNxrtGearSelectionPolicy`
- 選択できるノードの種類を制限する `XmNxrtGearSelectionRestrictions`

`XmNxrtGearSelectionPolicy` は、現在の選択モードを指定します。このリソースは、次の値にセットできます。

`XRTGEAR_SELECTION_POLICY_NONE`	選択不許可モード。ノードはドラッグすることはできますが、選択はできません。
`XRTGEAR_SELECTION_POLICY_SINGLE`	シングル・モード。一度にノードが一つしか選択できず、選択はトグルです(選択されているノードを再度選択すると、選択が解除されます)。選択されたノードだけがドラッグできます。

ツリー内の移動

PgUp、PgDn、Home、End、矢印キーを押す
スクロールバーをクリックする



ノードの選択

シングル・モード

MB1 を押してカレント・ノードを選択し、
他を選択解除する

範囲モード

- MB1 を押してカレント・ノードを選択し、他を選択解除する
- Shift+MB1 を押すか、MB1 をドラッグして範囲の一番下とカレント・ノードの間のノードを全て選択する



選択不許可モード

ノードは選択されない

シングル自動モード

カレント・ノードが常に選択される

マルチ・モード

- MB1 または Ctrl+MB1 を押して別の範囲の選択をする
- Shift+MB1 を押すか、MB1 をドラッグして範囲の一番下とカレント・ノードの間のノードを全て選択する



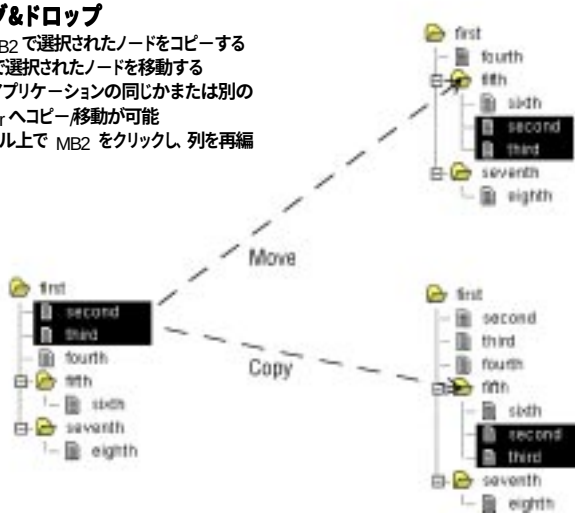
マルチ自動モード

- MB1 を押してカレント・ノードを選択し、他を選択解除する
- Shift+MB1 を押すか、MB1 をドラッグして範囲の一番下とカレント・ノードの間のノードを全て選択する
- Ctrl+MB1 を押して、別の範囲の選択を開始する



ドラッグ&ドロップ

- Ctrl+MB2 で選択されたノードをコピーする
- MB2 で選択されたノードを移動する
- 同一アプリケーションの同じかまたは別の Outliner へコピー/移動が可能
- 列ラベル上で MB2 をクリックし、列を再編成する



フォルダの開閉

MB1 またはショートカット・ボタンをクリックする
フォルダ上で MB1 をダブル・クリックする

列のリサイズ

列枠または列ラベルの枠で MB1 を押しながら、表示された点線をドラッグする



図 68 デフォルトのユーザ・インタラクション

XRTGEAR_SELECTION_POLICY_SINGLE_AUTO_SELECT
自動シングル・モード。一度にノードが一つしか選択できません。選択はトグルにはなりません(選択されているノードを再度選択しても、選択は解除されません)。選択されたノードだけがドラッグできます。

XRTGEAR_SELECTION_POLICY_RANGE 範囲モード。複数の(隣接している)ノードが選択できます。選択されているノードだけがドラッグできます。

XRTGEAR_SELECTION_POLICY_MULTIPLE マルチ・モード。ノードが隣接しているいないに関わらず複数のノードが選択できます。選択されたノードだけがドラッグできます。

XRTGEAR_SELECTION_POLICY_MULTIPLE_AUTO_UNSELECT
マルチ自動モード。カレント・ノードで選択/選択解除を行う前に、全ての選択されているノードをクリアします。選択されているノードだけがドラッグできます。

デフォルト値は、XRTGEAR_SELECTION_POLICY_SINGLE_AUTO_SELECT です。

XmNxrtGearSelectionPolicy は、選択できるノードの範囲を制限しますが、どのノードが選択できるかどうかは指定しません。選択できるノードの種類を指定するには、XmNxrtGearSelectionRestrictions をセットしてください。このリソースの値は、次の定数か(OR を使用した)その組み合わせです。

XRTGEAR_RESTRICT_NOTHING 制限なし。エンドユーザはどのノードも選択できます。

XRTGEAR_RESTRICT_ITEMS エンドユーザはフォルダのみ選択できます。アイテムは選択できません。

XRTGEAR_RESTRICT_FOLDERS エンドユーザはアイテムだけ選択できます。フォルダは選択できません。

XRTGEAR_RESTRICT_MULTI_LEVEL エンドユーザは、既に選択されているノードの兄弟(siblings)だけ選択できます。

XRTGEAR_RESTRICT_ALL 選択は全くできません。

デフォルトでは、XmNxrtGearSelectionRestrictions は XRTGEAR_RESTRICT_NOTHING にセットされています。

XmNxrtGearSelectionPolicy または XmNxrtGearSelectionRestrictions の値が変更されると、エンドユーザに選択されているノードは自動的に選択解除されます。

ノードをプログラムで選択する

通常、ノードはユーザ・インタラクションによって選択されますが、プログラム的にノードが選択された場合は、ノードは閉じられたフォルダに含まれることがありますので、アプリケーションによってそのフォルダを開き、ノードが見えるようにしてください。

7.10.2 フォーカスと Selected(選択されている状態)の違い

フォーカスがあるノードは、必ずしも選択されているとは限りません。フォーカスがあるノードは XmNxrtGearNodeCurrent に記憶され、ノードの周囲には枠が表示されます。一方、選択されているノードは XmNxrtGearSelectedNodeList に記憶され、その周囲にはハイライトが表示されます。

7.10.3 フォルダの状態の指定

通常、フォルダには"閉じている"と"開いている"の二つの状態が定義されます。フォルダは、子が表示されていない場合は閉じている状態で、子が表示されている場合は開いてる状態です。デフォルトでは、フォルダは、この二つの状態のどちらかです。(マウスの左ボタンのダブルクリック等により、このトグルを変更できます。)

その他に Outliner ウィジェットの XmNxrtGearFolderStateMask を使用して、"完全に閉じている"と"完全に開いている"間の状態を指定することができます。このリソースの値は次の定数か(OR を使用した)その組み合わせです。

XRTGEAR_FOLDERSTATE_CLOSED	フォルダは閉、子は表示されない
XRTGEAR_FOLDERSTATE_OPEN_FOLDERS	フォルダは開、フォルダの子だけ表示
XRTGEAR_FOLDERSTATE_OPEN_ITEMS	フォルダは開、子アイテムだけが表示
XRTGEAR_FOLDERSTATE_OPEN_ALL	フォルダは開、全ての子が表示

例えば、次の例では、開く(open)、フォルダだけ開く(open folders only)、閉じる(closed)の状態を指定しています。

```
XtVaSetValues(outliner,
  XmNxrtGearFolderStateMask, XRTGEAR_FOLDERSTATE_CLOSED |
  XRTGEAR_FOLDERSTATE_OPEN_FOLDERS | XRTGEAR_FOLDERSTATE_OPEN_ALL,
  NULL);
```

デフォルトは、XRTGEAR_FOLDERSTATE_CLOSED | XRTGEAR_FOLDERSTATE_OPEN です。

二つ以上の状態が定義されている場合、フォルダの状態の変更が繰り返されると、次のような順序でフォルダの状態が変化します。(定義されていない状態はスキップされます)。

XRTGEAR_FOLDERSTATE_CLOSED
XRTGEAR_FOLDERSTATE_OPEN_FOLDERS または
XRTGEAR_FOLDERSTATE_OPEN_ITEMS
XRTGEAR_FOLDERSTATE_OPEN_ALL

7.10.4 選択されているノード･リストの取得

エンドユーザによって現在選択されているノードのリストを取得するには、Outliner ウィジェットの XmNxrtGearSelectedNodeList リソースの値を取り出します。

```
Widget    outliner;
XrtGearObject nodelist;

XtVaSetValues(outliner,
XmNxrtGearSelectedNodeList, &nodelist,
NULL);
```

返されたリストは、エレメントが Node(または NodeFolder)オブジェクトで構成される XrtList オブジェクトです。XmNxrtGearSelectedNodeList は read-only です。

あるノードが選択されているかどうかを調べるには、そのノードの XmNxrtGearSelected の値を取り出します。このリソースが True であれば、ノードは選択されています。このリソースは設定可能なので、これを利用してプログラミングによるノードの選択を行うことができます。

7.10.5 選択コールバック

ノードが選択または選択解除された時に呼び出されるコールバックを指定することができます。Outliner ウィジェットの XmNxrtGearSelectionCallback で定義されるリストにコールバックを追加してください。各コールバックには、次の構造体が渡されます。

```
typedef struct {
  XrtGearReason   reason;     /* read-only */
  XEvent      *event;     /* read-only */
  Widget      container;  /* read-only */
  Widget      node;
  Widget      selected_node_list; /* read-only */
  Boolean     doit;
} XrtGearSelectCallbackStruct;
```

reason はコールバックの理由を指定し、値は次のうちのどれかです。

XRTGEAR_REASON_SELECT_BEGIN	選択の操作が開始
XRTGEAR_REASON_SELECT_END	選択の操作が終了
XRTGEAR_REASON_UNSELECT_BEGIN	選択解除の操作が開始
XRTGEAR_REASON_UNSELECT_END	選択解除の操作が終了

ノードが選択されると、選択のコールバックが次のように呼び出されます。

1. 新しく選択されたノードにコールバックが呼び出され、*reason* は XRTGEAR_REASON_SELECT_BEGIN にセットされます。
2. 選択操作の結果、選択解除されるノードにコールバックが呼び出され、*reason* は XRTGEAR_REASON_UNSELECT_BEGIN にセットされます。
3. 選択解除されるノードにコールバックが再び呼び出され、*reason* は XRTGEAR_REASON_UNSELECT_END にセットされます。
4. 新しく選択されたノードにコールバックが再び呼び出され、*reason* は XRTGEAR_REASON_SELECT_END にセットされます。

ノードの選択を制止するには、*reason* が XRTGEAR_REASON_SELECT_BEGIN の時に *doit* を False にセットします。

XmNxrtGearSelectionPolicy を XRTGEAR_SELECTION_POLICY_RANGE または XRTGEAR_SELECTION_POLICY_MULTIPLE にセットすることにより、ノードの選択解除を制止することができます。*reason* が XRTGEAR_REASON_UNSELECT の時に *doit* を False にセットしてください。

7.10.6 スクロールバー

ノード領域に対してツリーが大きすぎる場合は、スクロールバーが表示されます。エンドユーザはこのスクロールバーをクリックして、ノード領域にツリーのどの部分を表示するか指定できます。デフォルトでは、縦、横または両方のスクロールバーが必要時に表示されます。これらのスクロールバーは、Outliner の子として定義される別のウィジェットです。

次の Outliner リソースは、これらのスクロールバーをコントロールします。

- XmNxrtGearScrollBarHorizDisplayPolicy と XmNxrtGearScrollBarVertDisplayPolicy は、いつ縦と横のスクロールバーを表示するか指定します。有効な値は次の通りです。

XRTGEAR_DISPLAY_AS_NEEDED	必要時に表示(デフォルト)
XRTGEAR_DISPLAY_ALWAYS	常に表示
XRTGEAR_DISPLAY_NEVER	表示しない

- 縦と横のスクロールバーのウィジェット ID を取得するには、XmNxrtGearScrollBarHoriz と XmNxrtGearScrollBarVert リソースの値を取り出します。その後、スクロールバーのリソース(XmNbackground 等)をセットできます。

7.10.7 ドラッグ&ドロップ

Outliner ウィジェットは、次のようなドラッグ&ドロップが実行できます。

- サブ・ツリーを一つの Outliner ウィジェットから、同じアプリケーション内の別の Outliner に移動する
- 他のウィジェットからデータをインポートする

サブ・ツリーの移動は、XmNxrtGearNodeMovePolicy によってコントロールします。有効な値は次の通りです。

XRTGEAR_MOVE_NOWHERE	ノードの移動を禁止する(デフォルト)
XRTGEAR_MOVE_WITHIN_LEVEL	兄弟(siblings)間での移動のみを許可する
XRTGEAR_MOVE_WITHIN_APP	同じアプリケーション内の他の Outliner への移動を許可する
XRTGEAR_MOVE_WITHIN_WIDGET	Outliner ウィジェット内であればどこにでも移動可能

また、ノードはツリーの外にドロップすることもでき、その場合、ノード・ラベルはテキストに変換されます。

データをツリーへインポートするには、XmNxrtGearNodeImportPolicy をセットしてください。有効な値は次の通りです。

XRTGEAR_IMPORT_NOTHING	データのツリーへのインポートを許可しない(デフォルト)
XRTGEAR_IMPORT_ANYTHING	外部のテキスト等、データのインポートを許可する
XRTGEAR_IMPORT_WITHIN_APP	同じアプリケーション内の別の Outliner からのデータのインポートを許可する
XRTGEAR_IMPORT_WITHIN_WIDGET	同じウィジェットからのデータのインポートを許可する(コピー)

インポートされたデータは、次の決まりに従ってサブ・ツリーになります。

- newline がノードを区分
- Tab 文字がインデントのレベルを表す
- カンマが一つのノードの複数のラベルを区分

図 69 は、はこれらの決まりを使用して、どのように表示されるかを示しています。***<nl>*** は newline 文字、***<tab>*** は Tab 文字です。ここで使用されている特別文字は、*XrtGearNodeCreateTreeFromStream()*で使用されるデフォルトの特殊文字と同じです。

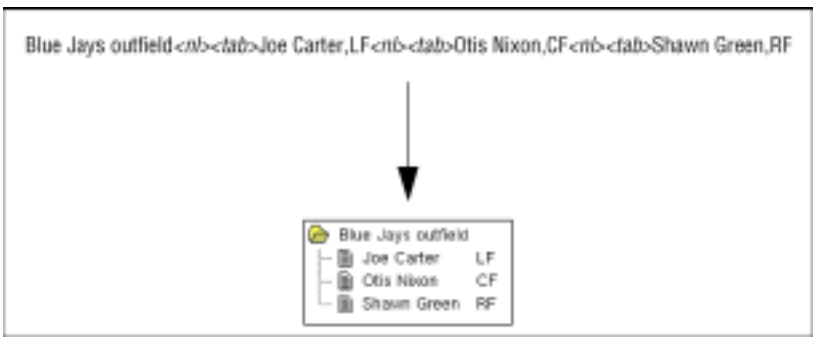


図 69　データをサブ・ツリーに変換

インポートされたデータからサブ・ツリーが作成されると、Outliner ツリーに挿入されます。エンドユーザによって現在指されているノードが、インポートの操作のターゲット(目的地)ノードとして扱われます。

実際のインポートの動作は、ターゲット・ノードに依存します。

- ターゲット・ノードがフォルダであれば、それが新しいサブ・ツリーの親になる
- ターゲット・ノードがアイテムで `XmNxrtGearAutoSort` が `False` であれば、ターゲット・ノードの親が、作成されたサブ・ツリーの親になる。ターゲット・ノードとサブ・ツリーのルートは兄弟(siblings)になり、ルートは、ターゲット・ノードの直前に配置される。
- ターゲット・ノードがアイテムで `XmNxrtGearAutoSort` が `True` であれば、ターゲット・ノードの親が、作成されたサブ・ツリーの親になる。ターゲット・ノードとサブ・ツリーのルートは兄弟(sibling)になり、兄弟は、ソートされた順に配置される。

図 70 は、インポートされたサブ・ツリーとターゲット・ノードの関係を示しています。

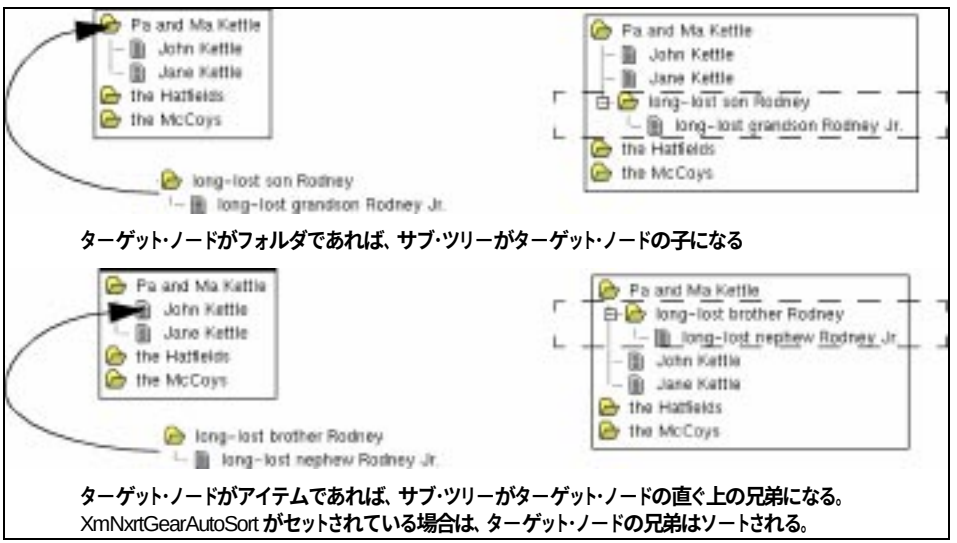


図 70　インポートされたデータとターゲット・ノード

インポートのコールバック

データがツリーにインポートされた時や移動された時に呼び出されるコールバックを指定できます。

XmNxrtGearImportCallback で定義されているリストにコールバックを追加して下さい。各コールバックには次の構造体が渡されます。

```
typedef struct {
  XrtGearReason   reason;     /* read-only */
  XEvent      *event;     /* read-only */
  String      target;     /* read-only */
  String      data;       /* read-only */
  unsigned long   data_length;/* read-only */
  Widget      object;     /* read-only */
  Boolean     doit;
} XrtGearOutlinerImportCallbackStruct;
```

サブ･ツリーは *object* がフォルダであれば、*object* の子になります。*object* がノードであれば、サブ･ツリーは、*object* の親の子になります。

XmNxrtGearImportCallback リストの各コールバックは、インポート操作開始前(*reason* は XRTGEAR_REASON_IMPORT_BEGIN)とインポート操作終了後(*reason* は XRTGEAR_REASON_IMPORT_END)に、二回呼び出されます。

特定なインポート操作を制止するには、*reason* が XRTGEAR_REASON_IMPORT_BEGIN の時に *doit* を False にしてください。

7.10.8 activate コールバック

デフォルトでは、*Activate()*は、エンドユーザがフォルダまたは列ラベルをダブル･クリックしたり、Space バーや Return キーを押した時に呼び出されます。activate コールバックは、*Activate()*アクションが呼び出されたときにはいつでも開始されるように指定できます。これにより、フォルダの開閉に従って自動的にアップデートするアプリケーションを書くことができます。

activate コールバックを定義するには、Outliner ウィジェットの XmNxrtGearActivateCallback で指定されるリストにコールバックを追加して下さい。このリソースには次の構造体が渡されます。

```
typedef struct {
  XrtGearReason   reason;     /* read-only */
  XEvent      *event;     /* read-only */
  Widget      container;  /* read-only */
  Widget      node;
  int         click_count;
} XrtGearNodeActivateCallbackStruct;
```

container は親の Outliner ウィジェット。*node* はアクティブになるノード。*click_count* はエンドユーザによるマウス･クリックの数

7.10.9 イベントの位置

アプリケーションが(Expose 等の)イベントを検知する場合、Outliner ウィジェットでは、イベントが起こったノードまたは列を知ることができます。

イベントが起こったノードを知るには、*XrtGearOutlinerNodeFromXEvent()*を呼び出して下さい。

```
Widget        outliner, node;
XEvent        *event;
XrtGearLocation    location;

node = XrtGearOutlinerNodeFromXEvent(outliner, event, &location);
```

この方法の戻り値は、イベントが起きた(または、近くに起きた)ノードです。この値が NULL であれば､イベントはノードでは起きていません｡三番目のパラメータが NULL 以外であれば、イベントの返された位置を指定します。このパラメータは、次の内のどれかです。

XRTGEAR_LOCATION_ON_NODE	イベントはノード上
XRTGEAR_LOCATION_ON_SHORTCUT	イベントはノードのショートカット･ボタン上
XRTGEAR_LOCATION_ON_LABEL	イベントは列ラベル上
XRTGEAR_LOCATION_ON_NOWHERE	イベントはノードでは起きていない

イベントが起きた列を知るには、*XrtGearColumnFromXEvent()*を呼び出して下さい。

```
Widget        outliner, column;
XEvent         *event;

column = XrtGearColumnFromXEvent(outliner, event);
```

この方法は、イベントが起きた列の XrtColumn オブジェクトを返します。イベントが列で起きなかった時は、NULL が戻ります。

7.10.10 列の再編成

デフォルトでは、エンドユーザは Outliner ウィジェット内で列を移動することができます。カーソルが移動したい列の上にある時にマウスの第二ボタンを押し、目的地でボタンを離してください。(但し、一番めの列は移動できません。)

エンドユーザが列の配置を変更できないようにするには、XmNxrtGearColumnMoveAllow を False にしてください。デフォルトでは、このリソースは True で、列の再編成が許可されています。

列移動のコールバック

エンドユーザによって列が移動された時に呼び出されるコールバックを指定できます。

XmNxrtGearColumnMoveCallback で定義されているリストにコールバックを追加してください。このリストの各コールバックには、次の構造体が渡されます。

```
typedef struct {
  XrtGearReason   reason;      /* read-only */
  XEvent      *event;      /* read-only */
  Widget      column;      /* read-only */
  Widget      target_sibling; /* read-only */
  Boolean     doit;
} XrtGearColumnMoveCallbackStruct;
```

column は移動される列。*target_sibling* は、移動が終了した時に移動された列の直前に位置する列。列の移動を制止するには、*reason* が XRTGEAR_REASON_COLUMN_MOVE_BEGIN の時に *doit* を False にセットしてください。

ColumnMove コールバックは、*XrtGearOutlinerMoveColumns()*が呼び出された時は、呼び出されません。

7.11 Outliner ウィジェットの印刷

XRT/gear では、Outliner ウィジェットの PostScript イメージを生成することができます。このイメージは、デバイスに依存しない PostScript フォントとオペレータを使用して作成されます。生成された PostScript ファイルは、PostScript プリンタで印刷したり他のドキュメントにインポートすることができます。

7.11.1 PostScript 出力の生成

XRT/gear には、Outliner ウィジェットの PostScript イメージ生成に、*XrtGearOutlinerDrawPS()*と *XrtGearOutlinerVaDrawPS()*の二つの方法が提供されています。これらの方法が生成する出力は同じですが、インターフェースが異なります。*XrtGearOutlinerVaDrawPS()*は、(*XtVaSetValues()*と同様に)*VarArgs*-style インターフェースを使用し、デフォルトから変更したい引数だけをセットできます。一方、

*XrtGearOutlinerDrawPS()*は *ArgList*-style を使用し、異なるプリントの要求に対する異なる引数のセットを保持することができます。

ヘッダーとフッターの定義

ウィジェットが印刷されるページにヘッダーとフッターを定義するには、*XrtGearOutlinerVaDrawPS()* または *XrtGearOutlinerDrawPS()* を呼び出す時に、XRTGEAR_PS_HEADER または XRTGEAR_PS_FOOTER をセットしてください。

これらのヘッダーとフッターに使用されるフォントは、XRTGEAR_PS_HEADER_FONT または XRTGEAR_PS_FOOTER_FONT でコントロールできます。フォントのサイズは、XRTGEAR_PS_HEADER_POINT_SIZE、XRTGEAR_PS_FOOTER_POINT_SIZE でセットしてください。指定されていない場合、デフォルトの 12.0 ポイントが使用されます。マージンは、XRTGEAR_PS_HEADER_MARGIN と XRTGEAR_PS_FOOTER_MARGIN でコントロールできます。

注意点

*XrtGearOutlinerVaDrawPS()*または *XrtGearOutlinerDrawPS()*を使用する時は、次の点に留意してください。

- XRTGEAR_PS_PAPERSIZE_と XRTGEAR_PS_MARGIN_の属性は、*image* ではなく、常に *paper* の上部に相対します。従って、ウィジェットが横方向(landscape)の時でも、XRTGEAR_PS_PAPERSIZE_HEIGHT は、イメージの高さではなく、ページの高さを計ります。
- 1 ページに複数のウィジェットを印刷するには、それらが重なり合わないように、XRTGEAR_PS_MARGIN_を使用して明示的に各ウィジェットを配置します。最後のイメージ以外は、XRTGEAR_PS_SHOWPAGE を False にセットしてください。

XrtGearOutlinerVaDrawPS()の例

次のコードは、landscape モードを使用してツリーの最初の三つのレベルをファイルに出力します。

```
FILE *fp;
fp = fopen("nodes.ps", "w");
if (fp) {
  numpages = XrtGearOutlinerVaDrawPS(outliner, fp,
    XRTGEAR_PS_ORIENTATION, XRTGEAR_PS_LANDSCAPE,
    XRTGEAR_PS_START_NODE, outliner,
    NULL);
  printf("%d page%s written.\n", numpages,
    (numpages==1)? "" : "s");
} else {
  printf("error opening file\n");
}
```

7.11.2 PostScript フォントのカスタマイズ

ほとんどの X サーバでは、PostScript 出力手続きが自動的に PostScript フォントを選択できるようになっています。または、XmNxrtGearPSFontMapList を使用して、X フォントに対応する PostScript のリストを定義することができます。このリストの各エレメントは XrtGearPSFontMap 型で、次のように定義されます。

```
typedef struct {
  String font_name;
  String tag;
  String PS_font;
  double point_size;
} XrtGearPSFontMap;
```

この構造体では、*PS_font* は PostScript フォント、*point-size* はポイント･サイズです。(図 71 は、標準の PostScript フォント名の一覧です。) X フォントは、次のどちらかの方法で指定されます。

- *font_name* を使用して、X Logical Font Description Conventions(XLFD)に準拠する X フォント名を指定。例:
- “-*-times-bold-r-normal-*-140-*”

tag を使用して、(bold 等の)XmNfontList フォント･タグを指定。

FontMap リストは、*PS_font* を NULL にセットしたエレメントで終了します。

point_size が 0.0 にセットされていると、PostScript フォントのポイント･サイズは、対応する X フォントのポイント･サイズから推定されます。

Times-Roman	Helvetica	NewCenturySchlbk-Roman
Times-Bold	Helvetica-Bold	NewCenturySchlbk-Bold
Times-Italic	Helvetica-Oblique	NewCenturySchlbk-Italic
Times-BoldItalic	Helvetica-BoldOblique	NewCenturySchlbk-BoldItalic
Palatino-Roman	Bookman-Light	Helvetica-Narrow
Palatino-Bold	Bookman-Demi	Helvetica-Narrow-Bold
Palatino-Italic	Bookman-LightItalic	Helvetica-Narrow-Oblique
Palatino-BoldItalic	Bookman-DemiItalic	Helvetica-Narrow-BoldOblique
Courier	AvantGarde-Book	Symbol
Courier-Bold	AvantGarde-Demi	ZapfChancery-MediumItalic
Courier-Oblique	AvantGarde-BookOblique	ZapfDingbats
Courier-BoldOblique	AvantGarde-DemiOblique	

図 71　35 の標準の LaserWriterII フォント名

次の例は、XmNxrtGearPSFontMapList を使用して、Helvetica Narrow PostScript フォントが Helvetica X フォントに対応するように指定しています。

```
XrtGearPSFontMap fontlist[] = {
  {"-*-helvetica-medium-r-*-*-*-*-*-*-*-*-iso8859-1",
    NULL, "Helvetica-Narrow", 0.0},
  {"-*-helvetica-bold-r-*-*-*-*-*-*-*-*-iso8859-1",
    NULL, "Helvetica-Narrow-Bold", 0.0},
  {"-*-helvetica-medium-o-*-*-*-*-*-*-*-*-iso8859-1",
    NULL, "Helvetica-Narrow-Oblique", 0.0},
  {"-*-helvetica-bold-o-*-*-*-*-*-*-*-*-iso8859-1",
    NULL, "Helvetica-Narrow-BoldOblique", 0.0},
  {NULL, NULL, NULL, 0.0}
};
...
XtVaSetValues(outliner,
  XmNxrtGearPSFontMapList, fontlist,
  NULL);
```

7.11.3 print コールバック

XmNxrtGearPrintCallback は、各ページが印刷される前後に呼び出されるコールバックのリストを指定します。

これらのコールバックを利用して、印刷のプロセスを監視したり、出力されるファイルを変更したりすることができます。このリストのコールバックには次の構造体が渡されます。

```
typedef struct {
  int     reason;      // Read-only
  XEvent  *event;      // Read-only
  int     current_page;    // Read-only
  int     total_pages; // Read-only
  int     across_pages;    // Read-only
  int     down_pages;  // Read-only
  int     page_number;
  String  header, footer;
  FILE    *fp;
  Boolean write_file_header;
  Boolean write_file_trailer;
  Boolean doit; // initially True
} XrtGearPrintCallbackStruct;
```

reason は、XRTGEAR_REASON_PAGE_PRINT_BEGIN(プリント前)かXRTGEAR_REASON_PAGE_PRINT_END(プリント後)のどちらかです。*current_page* はプリントされているページ、*total_page* はプリントされるページの合計数(*across_pages* x *down_pages*)です。

この構造体の次のメンバを変更することにより、プリントをコントロールできます。

- *header* と *footer* はページのヘッダーとフッター(*XrtGearOutlinerVaDrawPS()*のコールで指定される)
- *page_number* は、ヘッダーまたはフッターに表示されるページ番号
- *fp* は出力が書かれるファイル。(この値を新しいファイルに変更した場合、この新しいファイルに始めて書き込む時は、*write_file_header* を True にセットする必要があります。これにより、XRT/gear は PosfScript ファイルのヘッダーを再プリントします。) *fp* を変更することにより、例えば、奇数ページと偶数ページを別々のファイルに書くことができます。
- ファイルの最終ページを書いている時、このページが出力の最終ページではない場合、*write_file_trailer* を True にセットしてください。これにより、XRT/gear は、PostScript ファイルのトレイラー(trailer)を書きます。*write_file_trailer* は、*current_page* が *total_page*(プリントされているページが出力の最終ページ)であれば、自動的に True にセットされます。
- *doit* は、カレント・ページを印刷するかどうかコントロールします。ページをスキップするには、*reason* が XRTGEAR_REASON_PAGE_PRINT_BEGIN の時に doit を False にしてください。(*current_page* または *total_pages* では、*doit* は何も変更しません。)

7.12 パフォーマンス

7.12.1 ダブル・バッファリング

ダブル・バッファリングは、Outliner ウィジェットが変更またはスクロールされた時に、エンドユーザが受けるフラッシュ(ちらつき)を少なくするテクニックです。表示のアップデートは最初にオフ・スクリーンで描画され、その後画面にコピーされます。

デフォルトでは、Outliner ウィジェットはダブル･バッファリングを使用します。これをオフにするには、XmNxrtGearDoubleBuffer を False にセットしてください。

ダブル･バッファリングを使用すると、サーバへのメモリ要求が高くなります。

7.12.2 再描画のコントロール

デフォルトでは、Outliner は変更がある度に再描画されます。幾つかのリソースを変更する時は、変更を加える前に XmNxrtGearRepaint を False にし、ウィジェットの再描画を不可にしてください。リソースの変更後、XmNxrtGearRepaint をデフォルト値の True に戻します。この作業により、パフォーマンスが上がります。

7.12.3 列幅を固定する

デフォルトでは、XmNxrtGearWidthSizingPolicy は XRTGEAR_SIZE_VARIABLE にセットされており、一番幅が広いノード･ラベルに基づいて列幅が計算されます。この XmNxrtGearWidthSizingPolicy を XRTGEAR_SIZE_PIXEL または XRTGEAR_SIZE_CHAR にセットすることにより、列幅が固定され、パフォーマンスを上げることができます。

7.12.4 Lightweight ノードを使用する

Outliner のフォルダとアイテム･ノードは、それぞれ XtObjects です。そのため、何千ものフォルダやアイテム･ノードを使用した Outliner アプリケーションの場合、非常に多くの XtObject を扱うことになるので、パフォーマンスがかなり落ちてしまいます。そのような場合は、'lightweight'(軽量な)フォルダとアイテム･ノードを使用してください。Lightweight ノードは XtObject ではありませんが、Outliner のツリーに簡単に追加することができます。詳細は、"Lightweight フォルダとアイテム･ノード"(P212)の解説を参照してください。

7.12.5 XmNxrtGearNodeCheckForDuplicates の使用

XmNxrtGearNodeCheckForDuplicates は、Outliner がリストに重複しているノードがないかを調べるように指定します。これが True にセットされている場合、Outliner は、重複しているノードを調べますので、追加されるノードがリストのノードと重複しないことが確実な場合は、これを False にしてください。

7.12.6 完全な階層を指定しない

XrtGearNodeCreateTreeFromBuffer() と *XrtGearNodeCreateTreeFromStream()*では、ツリーの作成時に、各ノードに対して完全な階層が提供されるか(*full_path* は True)、または、インデントのレベルから推測させるか(*full_path* は False)どうかの選択ができます。パフォーマンスを重視する場合は、*full_path* を False にしてツリーを作成してください。

7.13 リソースの一覧

Outliner とそのオブジェクトのリソースと、よく使用される継承されたリソースの一覧です。リソース名の後は、データ型、デフォルト値、そしてリソースと共に使用される手続きです。**C** は、そのリソースが create(例: *XtCreateWidget()*)で定義でき、**S** は set(例: *XtSetValues()*)でき、**G** は get(例: *XtGetValues()*)で使用できることを意味します。

7.13.1 Outliner のリソース

XmNxrtGearActivateCallback XtCallbackList NULL CSG

*Activate()*アクションが呼び出された時に呼び出されるコールバックのリストを指定します。各ルーチンには、`XrtGearNodeActivateCallbackStruct` のポインタが渡されます。コールバックは、*XtAddCallback()*を使用してリストに追加してください。コールバックは次のように構成されます。

```
void
CallbackProcedure(
  Widget  widget,
  XtPointer   client_data,
  XtPointer   call_data /* points to
                an XrtGearNodeActivateCallbackStruct */
)
```

`XrtGearNodeActivateCallbackStruct` は次のように定義されます。

```
typedef struct {
  XrtGearReason   reason;     /* read-only */
  XEvent      *event;     /* read-only */
  Widget      container;  /* read-only */
  Widget      node;
  int         click_count;
} XrtGearNodeActivateCallbackStruct;
```

reason は XRTGEAR_REASON_ACTIVATE。*event* は *Activate()*アクションを起こした XEvent で、コールバックがプログラムによって呼び出されていれば、これは NULL になります。XEvent が起こったノードまたは列を取り出すには、*XrtGearNodeFromXEvent()*または *XrtGearColumnFromXEvent()*を呼び出して下さい。*container* は、親の Outliner ウィジェット、*node* はアクティブになるノードです。*click_count* は、エンドユーザによるマウス･クリックの数です。

このリソースはユーザ情報のコールバックだけで、Outliner ウィジェットに対するアクションを示しません。

XmNxrtGearAutoSort Boolean False CSG

ノードを兄弟(siblings)に対して自動的にソートするかどうか指定します。ノードは、このリソースが `True` に変更されると自動的にソートされます。

XmNbackground Pixel inherited CSG

ウィジェットの背景色を指定します。

XmNxrtGearBorderLocation XrtGearBorderLocation _EACH_LABEL_SEPARATED_FROM_NODES CSG

枠線を描く場所を指定します。有効な値は、次の通りです。

`XRTGEAR_BORDER_LOC_LABELS_SEPARATED_FROM_NODES`

列ラベルの領域とノード領域にそれぞれ枠線を描画

XRTGEAR_BORDER_LOC_EACH_LABEL_SEPARATED_FROM_NODES
列ラベルの領域とノード領域にそれぞれ枠線を描画し、各列ラベルの周囲にも枠を描く

XRTGEAR_BORDER_LOC_LABELS
列ラベルの領域にのみ枠線を描画

XRTGEAR_BORDER_LOC_EACH_LABEL
各列ラベルに個別に枠線を描画

XRTGEAR_BORDER_LOC_NODES
ノード領域の周囲に枠線を描画

XRTGEAR_BORDER_LOC_LABELS_AND_NODES
列ラベルとノードの両方を合わせた領域の周囲に枠線を描画

XmNxrtGearBorderType XrtGearBorderType XRTGEAR_BORDER_IN CSG

Outliner の枠線のスタイルを指定します。有効な値は次の通りです。

XRTGEAR_BORDER_PLAIN	前景色で描かれる装飾無しの線
XRTGEAR_BORDER_IN	ラベルが押されているように見える線
XRTGEAR_BORDER_OUT	ラベルが持ち上がっているように見える線
XRTGEAR_BORDER_ETCHED_IN	ラベルが押されているように見える二重線
XRTGEAR_BORDER_ETCHED_OUT	ラベルが持ち上がっているように見える二重線
XRTGEAR_BORDER_BEVEL	1pixel の枠、影はその 1/2 の太さで描画
XRTGEAR_BORDER_FRAME_IN	ラベルが押されているように見える 1 pixel の影付きの枠
XRTGEAR_BORDER_FRAME_OUT	ラベルが持ち上がっているように見える 1 pixel の影付きの枠
XRTGEAR_BORDER_NONE	枠線なし

枠線の幅は XmNxrtGearShadowThickness で指定されます。

XmNxrtGearColumnLabelsShow Boolean False CSG

列ラベルを表示するかどうか指定します。

XmNxrtGearColumnList XrtGearObject NULL G

Outliner ウィジェットに定義される、全ての column オブジェクトのリストを指定します。

XmNxrtGearColumnMoveAllow Boolean True CSG

列の移動を許可するかどうか指定します。最初の列は移動できません。

XmNxrtGearColumnMoveCallback XtCallbackList NULL CSG

列がエンドユーザによって移動された時に呼び出されるコールバックのリストを指定します。各ルーチンには、XrtGearColumnMoveCallbackStruct のポインタが渡されます。*XtAddCallback()*を使用してリストにコールバックを追加してください。コールバックは次のように構成されます。

```
void
CallbackProcedure(
  Widget  widget,
  XtPointer   client_data,
  XtPointer   call_data /* points to
              an XrtGearColumnMoveCallbackStruct */
)
```

XrtGearColumnMoveCallbackStruct は次のように定義されます。

```
typedef struct {
  XrtGearReason   reason;     /* read-only */
  XEvent      *event;     /* read-only */
  Widget      column;     /* read-only */
  Widget      target_sibling; /* read-only */
  Boolean doit;
} XrtGearColumnMoveCallbackStruct;
```

reason は XRTGEAR_REASON_COLUMN_MOVE_BEGIN(列 移 動 前) ま た は XRTGEAR_REASON_COLUMN_MOVE_END(列移動後)のどちらかです。*event* は NULL。*column* は移動される列です。*target_sibling* は、移動の操作が終了した時に、*column*(移動された列)の直前に位置する列です。*doit* は、列移動の操作を行うかどうかを示し、*reason* が XRTGEAR_REASON_COLUMN_MOVE_BEGIN の時に False にすることにより、列の移動を制止できます。

アプリケーションが *XrtGearOutlinerMoveColumns()*を呼び出した時は、ColumnMove コールバックは呼び出されません。

XmNxrtGearColumnOrderList XrtGearObject NULL G

列が表示される順に並んだ列のリストを指定します。表示の順序を変更するには、*XrtGearOutlinerMoveColumns()*を呼び出して下さい。

XmNxrtGearColumnSelected Widget NULL CSG

現在選択されている列のポインタを指定します。

XmNxrtGearColumnSelectionCallback Widget XtCallbackList CSG

列が選択された時に呼び出されるコールバックを指定します。各ルーチンには、XrtGearColumnSelectionCallbackStruct のポインタが渡されます。*XtAddCallback()*を使用してリストにコールバックを追加してください。コールバックは次のように構成されます。

```
void
CallbackProcedure(
  Widget  widget,
  XtPointer   client_data,
  XtPointer   call_data /* points to
            an XrtGearColumnSelectionCallbackStruct */
)
```

XrtGearColumnSelectionCallbackStruct は次のように定義されます。

```
typedef struct {
  XrtGearReason   reason;     /* read-only */
  XEvent      *event;     /* read-only */
  Widget      old_column; /* read-only */
  Widget      new_column;
  Boolean     doit;
} XrtGearColumnSelectionCallbackStruct;
```

reason は XRTGEAR_REASON_SELECT_BEGIN(選択前)、XRTGEAR_REASON_SELECT_END(選択後)、XRTGEAR_REASON_UNSELECT_BEGIN(選択解除前)、XRTGEAR_REASON_UNSELECT_END(選択解除後)のどれかです。*event* は、そのコールバックを呼び出した XEvent で、コールバックがプログラムによって呼び出されている場合は NULL です。*old_column* は前の列で、read-only です(NULL の場合もあります)。*new_column* は選択された列で、*reason* が XRTGEAR_REASON_SELECT_BEGIN の時に変更することもできます。*doit* は、列の選択を許可するかどうかを指定し、*reason* が XRTGEAR_REASON_SELECT_BEGIN の時に変更することができます。

XmNxrtGearColumnSelectionCallback で指定されるコールバックは、次の XrtGearColumnSelectionCallbackStruct 構造体の *reason* メンバと共に、次の順序で呼び出されます。

XRTGEAR_REASON_SELECT_BEGIN	新しく選択されたノード上
XRTGEAR_REASON_UNSELECT_BEGIN	前に選択されていたノード上
XRTGEAR_REASON_UNSELECT_END	前に選択されていたノード上
XRTGEAR_REASON_SELECT_END	新しく選択されたノード上

XmNxrtGearCursor　Cursor　XC_left_ptr　CSG

Outliner の通常のポイント&クリック操作に使用されるマウスのカーソルを指定します。

XmNxrtGearCursorResizeWidth　Cursor　XC_right_side　CSG

列幅がリサイズされる時に使用されるマウスのカーソルを指定します。

XmNxrtGearCursorTracking　Boolean　True　CSG

マウスが画面上を移動した時にカーソルの形を変更し、XmNxrtGearCursorTrackingCallback で指定されるコールバック・ルーチンを呼び出すかどうか指定します。True であれば、カーソルは変更され、コールバックが呼び出されます。

XmNxrtGearCursorTrackigCallback　XtCallbackList　NULL　CSG

XmNxrtGearCursorTracking が True にセットされ、カーソルが、マウスによって何らかの操作ができる領域へ移動した時に呼び出されるコールバックを指定します。各ルーチンには、XrtGearCursorTrackingCallbackStruct のポインタが渡されます。

コールバックは、*XtAddCallback()*を使用してリストに追加してください。

```
void
CallbackProcedure(
  Widget  widget,
  XtPointer   client_data,
  XtPointer   call_data /* points to an
              XrtGearCursorTrackingCallbackStruct */
)
```

XrtGearCursorTrackingCallbackStruct は次のように定義されます。

```
typedef struct {
  XrtGearReason   reason;     /* read-only */
  XEvent      *event;     /* read-only */
  Widget      object;     /* read-only */
  Cursor      old_cursor; /* read-only */
  Cursor      new_cursor;
} XrtGearCursorTrackingCallbackStruct;
```

reason は次の内のどれかです。

XRTGEAR_REASON_CURSOR_EDGE_LEFT	カーソルが左端へ移動
XRTGEAR_REASON_CURSOR_EDGE_RIGHT	カーソルが右端へ移動
XRTGEAR_REASON_CURSOR_INSIDE	カーソルがノードまたはラベルの内側へ移動
XRTGEAR_REASON_CURSOR_LEAVE	カーソルがノードまたはラベルを離れる

event はコールバックを呼び出す XEvent で、コールバックがプログラムによって呼び出された場合は NULL です。*object* は、現在カーソルがあるノードまたは列です。*old_cursor* は、マウスが画面上で操作可能な領域に移動しコールバックを呼び出す前の、カーソルの形です。*new_cursor* はカーソルの新しい形で、これは変更可能です。

XmNxrtGearDoubleBuffer Boolean True CSG

パフォーマンスを向上させるためにダブル・バッファリング機能を使用するかどうかを指定します。これを使用すると、メモリの要求が高まります。

XmNeditable Boolean True CSG

ノード・ラベルの編集をユーザに許可するかどうかを指定します。False であれば、ノード、フォルダ、列に対する編集の設定を上書きします。

XmNxrtGearEditColumn int XRTGEAR_NOVALUE G

編集されているノードを持つ列のインデックスを指定します。現在何も編集されていなければ、XRTGEAR_NOVALUE です。

XmNxrtGearEditNode Widget NULL G

編集されているノードのウィジェット ID を指定します。現在編集されているノードがなければ NULL です。

XmNxrtGearEditor Widget NULL G

セルの編集に使用されているカレント・ウィジェットの ID を指定します。このリソースは、read-only で、現在編集されているセルがなければ NULL です。このバージョンでは、セルの編集には、XmText のみがサポートされています。

XmNxrtGearFolderStateCallback XtCallbackList NULL CSG

フォルダ・ノードの状態が、*ChangeState()*アクションまたは *XrtGearNodeChangeFolderState()*によって変更された時に呼び出されます。各コールバックには、XrtGearNodeFolderStateCallbackStruct のポインタが渡されます。コールバックは、*XtAddCallback()*を使用してリストに追加してください。コールバックは次のように構成されます。

```
void
CallbackProcedure(
  Widget  widget,
  XtPointer   client_data,
  XtPointer   call_data /* points to an
          XrtGearNodeFolderStateCallbackStruct */
)
```

XrtGearNodeFolderStateCallbackStruct は次のように定義されます。

```
typedef struct {
  XrtGearReason   reason;     /* read-only */
  XEvent      *event;     /* read-only */
  Widget      container;  /* read-only */
  Widget      node;
  XrtGearFolderState  new_state;
  XrtGearFolderState  old_state;  /* read-only */
  Boolean doit;
} XrtGearNodeFolderStateCallbackStruct;
```

reason は XRTGEAR_REASON_FOLDER_CHANGE_BEGIN または XRTGEAR_REASON_FOLDER_CHANGE_END のどちらかです。*event* は、コールバックを呼び出したコールバックで、コールバックがプログラムによって呼び出された時は NULL です。*container* は親の Outliner ウィジェットです。*node* は状態が変更されるフォルダ・ノードです。*new_state* はフォルダの新しい状態、*old_state* はフォルダの古い状態です。*reason* が XRTGEAR_REASON_FOLDER_CHANGE_BEGIN の時に *doit* を False にすると、フォルダの状態の変更が不可になります。

XmNxrtGearFolderStateMask int XRTGEAR_FOLDERSTATE_OPEN_ALL|_CLOSED CSG

フォルダの状態をコントロールします。有効な値は、次の値またはその組み合わせ(OR)です。

XRTGEAR_FOLDERSTATE_CLOSED	フォルダは閉。子は表示されていない。
XRTGEAR_FOLDERSTATE_OPEN_FOLDER	フォルダは開。子のフォルダが表示される。
XRTGEAR_FOLDERSTATE_OPEN_ITEMS	フォルダは開。子のアイテムが表示される。
XRTGEAR_FOLDERSTATE_OPEN_ALL	フォルダは開。全ての子が表示される。

必ず一つの状態は指定してください。XRTGEAR_FOLDERSTATE_OPEN_FOLDERS と XRTGEAR_FOLDERSTATE_OPEN_ITEMS は相反する状態ですので、これらは同時に指定できません。フォルダ・ノードをクリックすることにより、次の状態を順に繰り返します。

```
XRTGEAR_FOLDERSTATE_CLOSED
XRTGEAR_FOLDERSTATE_OPEN_FOLDERS,XRTGEAR_FOLDERSTATE_OPEN_ITEMS
XRTGEAR_FOLDERSTATE_OPEN_ALL
```

XmNfontList XmFontList inherited CSG

全てのノードのデフォルトのフォントを指定します。XmNxrtGearHeightPreferredCount と XmNxrtGearWidthChar によって幅の計算にも使用されます。

XmNforeground Pixel inherited CSG

デフォルトの前景色を指定します。ショートカット・ボタンは常にこの色で描画されます。

XmNheight Dimension determined at run-time CSG

ウィジェットの高さを pixel で指定します。

XmNxrtGearHeightPreferredCount int 10 CSG

XmNxrtGearHeightSizingPolicy が XRTGEAR_SIZE_NODE または XRTGEAR_SIZE_CHAR にセットされている時に、ツリーの表示部分に要求される高さを計算する時に使用されるリソースです。

XmNxrtGearHeightSizingPolicy が XRTGEAR_SIZE_NODE にセットされている場合は、XmNxrtGearHeightPreferredCount の値に XmNxrtGearNodeHeight の値が乗算され、ツリーの表示部分の理想的な高さを計算します。

XmNxrtGearHeightSizingPolicy が XRTGEAR_SIZE_CHAR にセットされている場合は、XmNxrtGearHeightPreferredCount の値に XmNfontList の中の一番背の高いフォントの高さが乗算され、ツリーの表示部分の理想的な高さを計算します。

XmNxrtGearHeightResizePolicy XrtGearWidgetResizePolicy XRTGEAR_WIDGET_RESIZE_VARIABLE CSG

いつウィジェットが自分自身をリサイズするかどうか指定します。有効な値は次の通りです。

XRTGEAR_WIDGET_RESIZE_VARIABLE	XmNxrtGearHeightSizingPolicy で決定された高さにリサイズされる。
XRTGEAR_WIDGET_RESIZE_GROW_ONLY	ウィジェットが大きくなる場合にのみリサイズする。それ以外は変化なし。
XRTGEAR_WIDGET_RESIZE_FIXED	親が要求しない限りリサイズしない。

XmNxrtGearHeightSizingPolicy XrtGearSizingPolicy XRTGEAR_SIZE_NODE CSG

ツリーの表示部分の理想的な高さを計算する方法を指定します。有効な値は次の通りです。

XRTGEAR_SIZE_PIXEL	XmNxrtGearHeightPreferredCount の値を使用し、理想的な高さを pixel で指定する。
XRTGEAR_SIZE_CHAR	XmNfontList で指定されるリストの中の一番背の高いフォントの高さに XmNxrtGearHeightPreferredCount の値を乗算し、高さを計算する。
XRTGEAR_SIZE_VARIABLE	閉じられていない(non-collapsed)ノードの数を使用して高さを計算する。
XRTGEAR_SIZE_NODE	次の計算式を用いて高さを計算する。

```
XmNxrtGearHeightPreferredCount * XmNxrtGearNodeHeight +
(XmNxrtGearHeightPreferredCount - 1) * XmNxrtGearNodeSpacing +
XmNhighlightThickness * 2
```

XmNxrtGearheightVirtual　int　dynamic　G

ウィジェットの仮想的な高さを pixel で指定します。この高さは、縦のスクロールバーによって保持されます。高さが ULONG_MAX より大きい場合は、この値は ULONG_MAX にセットされます。

XmNhighlightColor　Pixel　inherited　CSG

(XmNxrtGearNodeCurrent で指定される)カレント・ノードの周囲に描画されるフォーカス・ハイライトの外側の部分の色を指定します。フォーカス・ハイライトの内側の部分は、XmNbackground で指定される背景色で描画されます。)

XmNhighlightThickness　Dimension　2　CSG

カレント・ノードの周囲に描画されるフォーカス・ハイライトの幅を指定します。このフォーカス・ハイライトは、二色で描画されます。(外側の部分は、XmNhighlightColor で指定される色、内側は XmNbackground で指定される色で描画されます。)XmNhighlightThickness が奇数の値にセットされると、外側の部分が内側より 1pixel 分大きく描画されます。

この値は、必ず 0 より大きい値にセットしてください。

XmNxrtGearImportCallback　XtCallbackList　NULL　CSG

データが Outliner ウィジェットに移動またはインポートされた時に呼び出されるコールバックを指定します。各ルーチンには、XrtGearOutlinerImportCallbackStruct のポインタが渡されます。コールバックは、*XtAddCallback()*を使用してリストに追加してください。コールバックは次のように構成されます。

```
void
CallbackProcedure(
  Widget  widget,
  XtPointer   client_data,
  XtPointer   call_data /* points to an
              XrtGearOutlinerImportCallbackStruct */
)
```

XrtGearOutlinerImportCallbackStruct は次のように定義されます。

```
typedef struct {
  XrtGearReason   reason;     /* read-only */
  XEvent      *event;     /* read-only */
  String      target;     /* read-only */
  String      data;       /* read-only */
  unsigned long   data_length;/* read-only */
  Widget      object;     /* read-only */
  Boolean     doit;
} XrtGearOutlinerImportCallbackStruct;
```

reason は XRTGEAR_REASON_IMPORT_BEGIN(インポート操作前)、XRTGEAR_REASON_IMPORT_END(インポート操作後)のどちらかです。*event* は NULL。*target* はインポートされるデータの型名、*data* はインポートされるデータです。newline は、インポートされたデータから作成されたサブツリーのノードを区分し、tab がインデントのレベル、カンマが複数ラベルを区分します。*data_length* はインポートされたデータの長さです。

object は、エンドユーザによって指定された列またはターゲット・ノードです。*object* がフォルダであれば、インポートされたサブ・ツリーの親になります。

object がノードであれば、サブ・ツリーのルートは、*object* の兄弟(sibling)になります。(XmNxrtGearAutoSort が True であれば、sibling のルートは、ソートされた順に配置されます。)インポート操作は、*reason* が XRTGEAR_REASON_IMPORT_BEGIN の時に、*doit* を False にセットすると、この操作を制止できます。

XmNxrtGearMarginHeight Dimension 5 CSG

ウィジェットと枠線の間の上下のマージンを指定します。

XmNxrtGearMarginWidth Dimension 5 CSG

ウィジェットと枠線の間の左右のマージンを指定します。

XmNxrtGearNodeCheckForDuplicates Boolean True CSG

子のリストに重複しているノードがないかどうかを Outliner に調べさせます。True の場合は、Outliner が重複しているノードを調べますので、追加されるノードが重複しないことが確実である場合は、これを False にセットしてください。

XmNxrtGearNodeChildList XrtGearObject NULL G

描画されるノードを持つ XrtList オブジェクトのポインタを指定します。(Outliner ウィジェットの子、ツリーのルート・ノードの役割を持つ)

XmNxrtGearNodeCompareProc XrtGearListItemCompareProc NULL CSG

二つのノードを比較してソートする時に呼び出される手続きを指定します。デフォルトの手続きは、ノードのラベルを数字とアルファベット順に比較します。XmNxrtGearColumnSortOrderList がセットされている時は、複数のキーを使用してソートが行えます。ソートの手続きは、次のように構成されます。

```
int compare(XtPointer node1, XtPointer node2)
```

手続きは、node1 < node2 であれば負の値、node1 = node2 であれば 0、node1 > node2 であれば正の値が返ります。

XmNxrtGearNodeCreateCallback XtCallbackList NULL CSG

Outliner ウィジェットがノードを作成する時に呼び出されるコールバックを指定します。コールバックは、通常 *Drag*(*Copy*) アクションの結果呼び出されます。各ルーチンには、XrtGearNodeCreateCallbackStruct のポインタが渡されます。コールバックは、*XtAddCallback()*を使用してリストに追加してください。コールバックは次のように構成されます。

```
void
CallbackProcedure(
  Widget  widget,
  XtPointer   client_data,
  XtPointer   call_data /* points to an
            XrtGearNodeCreateCallbackStruct */
)
```

XrtGearNodeCreateCallbackStruct は次のように定義されます。

```
typedef struct {
  XrtGearReason   reason;     /* read-only */
  XEvent      *event;     /* read-only */
  Widget      source_node;/* read-only */
  XrtGearObject   label;
  Widget      node_style;
  XtPointer       user_data;
  Widget      new_node;
} XrtGearNodeCreateCallbackStruct;
```

reason は XRTGEAR_REASON_NODE_CREATE_BEGIN(ノード作成前)または XRTGEAR_REASON_NODE_CREATE_END(ノード作成後)です。*event* は NULL です。新しいノードがコピーの結果作成される場合は、*source_node* はコピーされるノードです。

reason が XRTGEAR_REASON_NODE_CREATE_BEGIN であれば、*label*、*node_style*、*user_data*、*new_node* は NULL ですが、編集可能です。*label*、*node_style*、*user_data* をセットすることにより、ラベル、ノード・スタイル、ユーザ定義のデータを定義できます。*new_node* をセットすることにより、新しいノードを定義することができます。*new_node* がセットされた場合、*reason* が XRTGEAR_REASON_NODE_CREATE_END のコールバックは呼び出されません。

reason が XRTGEAR_REASON_NODE_CREATE_END であれば、*new_node* は新しく作成されたノードを指します。*label*、*node_style*、*user_data* は、新しく作成されたラベル、ノード・スタイル、ユーザが定義したデータを指します。これらの値は、この時点では変更できません。

これらのコールバックは、ノードが(*XrtGearNodeCreateTreeFromStream()*等により)プログラムによって作成された場合は、呼び出されません。

XmNxrtGearNodeCurrent　Widget　NULL　CSG

現在フォーカスがあるノードのポインタを指します。エンドユーザによってセットされた場合、新しくカレント・ノードになったノードが(非表示の場合は)表示されます。このリソースを変更しても、XmNxrtGearNodeCurrentChangeCallback によって指定されているコールバックは呼び出されません。

XmNxrtGearNodeCurrentChangedCallback　XtCallbackList　NULL　CSG

Outliner ウィジェットのカレント・ノードが、エンドユーザのアクションまたは *XrtGearNodeTraverseTo()*によって変更された時に呼び出されるコールバックを指定します。(XmNxrtGearNodeCurrent を変更した場合は、これらのコールバックは呼び出されません。)各ルーチンには、XrtGearNodeCurrentChangedCallbackStruct のポインタが渡されます。コールバックは、*XtAddCallback()*を使用してリストに追加してください。コールバックは次のように構成されます。

```
void
CallbackProcedure(
  Widget  widget,
  XtPointer   client_data,
  XtPointer   call_data /* points to an
            XrtGearNodeCurrentChangedCallbackStruct */
)
```

XrtGearNodeCurrentChangedCallbackStruct は次のように定義されます。

```
typedef struct {
  XrtGearReason   reason;      /* read-only */
  XEvent      *event;      /* read-only */
  Widget      container;   /* read-only */
  Widget      previous_node;   /* read-only */
  Widget      new_node;
  Boolean     doit;
} XrtGearNodeCurrentChangedCallbackStruct;
```

event はコールバックを呼び出した XEvent で、プログラムによって呼び出された場合は NULL。*reason* は XRTGEAR_REASON_NODE_CHANGE_BEGIN または XRTGEAR_REASON_NODE_CHANGE_END です。*container* は親の Outliner ウィジェットです。*previous_node* は前のカレント・ノードで、*new_node* は新しいカレント・ノードです。*reason* が XRTGEAR_REASON_NODE_CHANGE_BEGIN であれば、*new_node* を他のウィジェットを参照するようにセットしたり、*doit* を False にしてノード変更の操作を不可にすることができます。(デフォルトでは、*doit* は True。)*reason* が XRTGEAR_REASON_NODE_CHANGE_END であれば、*new_node* は新しいカレント・ノードを指し、これは変更できず、*doit* をセットしても何も変わりません。

XmNxrtGearNodeEnterCellCallback XtCallbackList NULL CSG

セル間のトラバースの前後に呼び出されるコールバックのリストを指定します。XrtGearNodeEnterCellCallbackStruct のポインタが各コールバックに渡されます。コールバックは次のように構成されます。

```
void
XrtGearNodeEnterCellCallback (
Widget    outliner,
XtPointer client_data
XtPointer call_data /* points to an
XrtGearNodeEnterCellCallbackStruct */
)
```

XrtGearNodeEnterCellCallbackStruct は次のように定義されます。

```
Typedef struct {
XrtGearReason reason; /* read only */
Xevent    *event; /* read only */
Widget    previous_node; /* read only */
Int       previous_column; /* read only */
Widget    editor; /* Widget used as the editor */
Widget    new_node;
Int       new_column;
Boolean   doit; /* initally True */
} XrtGearNodeEnterCellCallbackStruct;
```

reason は、XRTGEAR_REASON_NODE_ENTER_CELL_BEGIN（セルに入る前）か、XRTGEAR_REASON_NODE_ENTER_CELL_END（テキスト・ウィジェットが新しいセルにマップされた後、但し通知のみ）のどちらかです。*previous_node* と *previous_column* はどこからテキスト・ウィジェットが来たか(現在編集中のセルがなければ NULL か XRTGEAR_NOVALUE)を指します。*new_node* と *new_column* はトラバースされるセルです。*doit* は、セルへ入ることを許可するかどうかを決定するフラグです。ユーザがそのセルに入れないようにするには、*reason* が XRTGEAR_REASON_ENTER_CELL_BEGIN の時に *doit* を False にしてください。目的地のセルを変更する時は、*reason* が XRTGEAR_REASON_ENTER_CELL_BEGIN の時に *new_node* と *new_column* を変更してください。

XmNxrtGearNodeHeight Dimension 20 CSG

ノードの高さを指定します。デフォルトでは、ノード・スタイルの最大の高さです。

XmNxrtGearNodeHighlightOnDrag Boolean False CSG

True の時は、現在マウス・ポインタがあるノードが、ドラッグ&ドロップ操作中にハイライトされます。これにより、マウス・ボタンが放された時に、どのノードがターゲットになるかわかりやすくなります。

XmNxrtGearNodeImportPolicy XrtGearImportPolicy XRTGEAR_IMPORT_NOTHING CSG

Outliner ウィジェットにデータをインポートする時の制限事項を指定します。有効な値は次の通りです。

XRTGEAR_IMPORT_NOTHING	ツリーには、データはインポートできない (デフォルト)
XRTGEAR_IMPORT_ANYTHING	外部のテキスト等、データは全てインポート可能
XRTGEAR_IMPORT_WITHIN_APP	同じアプリケーション内の他の Outliner からのインポートは可能
XRTGEAR_IMPORT_WITHIN_WIDGET	同じウィジェット内からのインポートが可能

XmNxrtGearNodeIndent int 20 CSG

各階層レベルのインデント幅を pixel で指定します。これに負の値をセットすることはできません。

XmNxrtGearNodeMoveCallback XtCallbackList NULL CSG

ノードが移動されたり、親が変更された時に呼び出されるコールバックを指定します。これらのコールバックは、ノードがエンドユーザによってインタラクティブに移動された時に呼び出されます。各ルーチンには、XrtGearNodeMoveCallbackStruct のポインタが渡されます。コールバックは、*XtAddCallback()*を使用してリストに追加してください。コールバックは次のように構成されます。

```
void
CallbackProcedure(
  Widget  widget,
  XtPointer   client_data,
  XtPointer   call_data /* points to an
                XrtGearNodeMoveCallbackStruct */
)
```

XrtGearNodeMoveCallbackStruct は次のように定義されます。

```
typedef struct {
  XrtGearReason   reason;     /* read-only */
  XEvent      *event;     /* read-only */
  unsigned char   operation;  /* read-only */
  Widget      node;       /* read-only */
  Widget      old_parent; /* read-only */
  Widget      new_parent; /* read-only */
  Widget      new_next;
  Boolean     doit;
} XrtGearNodeMoveCallbackStruct;
```

reason は XRTGEAR_REASON_NODE_MOVE_BEGIN、XRTGEAR_REASON_NODE_MOVE_END、XRTGEAR_REASON_NODE_DRAG_BEGIN または XRTGEAR_REASON_NODE_DRAG_END です。*event* は NULL。*operation* は XmDROP_MOVE(移動操作)または XmDROP_COPY(コピー操作)です。*node* は移動されるノードです。(親が変更される場合は)*old_parent* は現在の親で、*new_parent* は新しい親です。*new_next* は *new_parent* の子で、NULL でなければ、ノード移動完了時に、*node* の一番若い兄弟(sibling)を表します。XmNxrtGearAutoSort が False であれば、*new_next* を修正して、*new_parent* の子のリスト上での *node* の位置を変更できます。

reason が XRTGEAR_REASON_NODE_MOVE_BEGIN の時に、*doit* を False にセットすることにより、移動を制止することができます。移動操作終了後、再描画される前に *reason* が XRTGEAR_REASON_NODE_MOVE_END にセットされコールバックが呼び出されます。

XmNxrtGearNodeMovePolicy XrtGearMovePolicy XRTGEAR_MOVE_NOWHERE CSG

Outliner ウィジェット内で、エンドユーザによるドラッグ&ドロップを許可するかどうか指定します。有効な値は次の通りです。

XRTGEAR_MOVE_NOWHERE	ノードの移動を許可しない
XRTGEAR_MOVE_WITHIN_LEVEL	兄弟(sibling)間で順序が変更可能
XRTGEAR_MOVE_WITHIN_WIDGET	同一 Outliner 内であれば、ノードはどこにでも移動可能
XRTGEAR_MOVE_WITHIN_APP	同一アプリケーション内であれば、他の Outliner にも移動可能

XmNxrtGearNodeSpacing Dimension 2 CSG

ノード間の上下のスペースを指定します。この値は、必ず XmNhighlightThickness の値と同じかそれ以上にしてください。それにより、フォーカス・ハイライトが描画されるスペースが確保されます。

XmNxrtGearNodeStyleDefault Widget default node style CSG

デフォルトのノード・スタイルとして使用される NodeStyle オブジェクトを指定します。このリソースがセットされていない場合は、XmNxrtGearNodeStyleList で指定されるリストの最初のエレメントがデフォルトのノード・スタイルとして使用されます。

XmNxrtGearNodeStyleList XrtGearObject contains default node style G

NodeStyle オブジェクトの XrtList のポインタを指定します。XmNxrtGearNodeStyleDefault がセットされていなければ、このリストの最初のエレメントがデフォルトのノード・スタイルとして扱われ、XmNxrtGearNodeStyle がセットされていない全てのノードに使用されます。

XmNxrtGearNodeTop Widget dynamic CSG

一番上に表示されるノードのポインタを指定します。この値は、XmNxrtGearNodeVisibleList で指定されるリストからも取得できます。

XmNxrtGearNodeValidateCallback XtCallbackList NULL CSG

ノード・ラベルの編集が確定される前後に呼び出されるコールバックのリストを指定します。このコールバックは、*CommitEdit()* アクションによって開始されます。`XrtGearValidateNodeCallbackStruct` のポインタが各コールバックに渡されます。コールバックは次のように構成されます。

```
void
ValidateCallback (
Widget outliner,
XtPointer client_data
XtPointer call_data // points to an
XrtGearNodeValidateCallbackStruct
)
```

`XrtGearNodeValidateCallbackStruct` は次のように定義されます。

```
typedef struct {
XrtGearReason     reason; /* read only */
Xevent        *event; /* read only, currently NULL */
Widget        node; /* read only */
XrtGearObject     old_label; /* read only */
int           column; /* read only */
String        value;
Boolean       doit; /* initially True */
Boolean       bell; /* initially False */
} XrtGearNodeValidateCallbackStruct;
```

reason は `XRTGEAR_REASON_NODE_VALIDATE_BEGIN`(編集が確定される前)または、`XRTGEAR_REASON_NODE_VALIDATE_END`(編集が確定された後)のどちらかです。*XrtGearOutlinerCommitEdit()*への呼び出しの結果であれば、*event* は `NULL` です。*old_label* は編集される前のラベル、*column* は編集されるノード・ラベルが属する列、*value* は(*doit* が `True` であれば)確定されるデータです。この値を確定させないようにするには、(*reason* が `XRTGEAR_REASON_NODE_VALIDATE_BEGIN` の時に)*doit* を `False` にセットしてください。*bell* は、編集されたセルが確定されていないことを示すベルを鳴らすかどうかを示します。

XmNxrtGearNodeVisibleList XrtGearObject dynamic G

現在表示されているノードのリストである、XrtList オブジェクトのポインタを指定します。

XmNxrtGearPrintCallback XtCallbackList NULL CSG

各ページが印刷される度に呼び出されるコールバックのリストを指定します。各ルーチンには、`XrtGearPrintCallbackStruct` のポインタが渡されます。コールバックには、*XtAddCallback()* を使用してリストに追加してください。コールバックは次のように構成されます。

```
void
CallbackProcedure(
  Widget  widget,
  XtPointer   client_data,
  XtPointer   call_data /* points to an
              XrtGearPrintCallbackStruct */
)
```

XrtGearPrintCallbackStruct は次のように定義されます。

```
typedef struct {
  int     ason;        /* read-only */
  XEvent  event;       /* read-only */
  int     current_page;    /* read-only */
  int     total_pages; /* read-only */
  int     across_pages;    /* read-only */
  int     down_pages;  /* read-only */
  int     page_number;
  String  header, footer;
  FILE    *fp;
  Boolean write_file_header;
  Boolean write_file_trailer;
  Boolean doit;
} XrtGearPrintCallbackStruct;
```

reason は XRTGEAR_REASON_PAGE_PRINT_BEGIN または XRTGEAR_REASON_PAGE_PRINT_END のどちらかです。*event* は常に NULL。*current_page* はプリントされているカレント・ページです。*across_pages* は、Outliner ウィジェットを印刷する時に横方向に必要なページ数です。*down_pages* は、Outliner ウィジェットを印刷する時に縦方向に必要なページ数です。*total_pages* はプリントされるページの合計数(*across_pages* x *down_pages*) です。*page_number* は現在プリントされているページの番号で、(*XrtGearOutlinerVaDrawPS()*で指定される)ページのヘッダーまたはフッター上で#というシンボルが指定されている個所に印刷されます。*header* と *footer* はカレント・ページのヘッダーとフッターです。*fp* は現在の出力のファイル・ポインタです。*write_file_header* は、PostScript ヘッダーを書くかどうかを指示します(*current_page* が 1 であれば、*write_file_header* は True)。*write_file_trailer* は、PostScript トレイラー (trailer) を書くかどうかを指定し、*current_page=total_pages* であれば自動的に True にセットされます。*doit* は、プリント時にそのページをスキップ(抜かす)かどうかを指定しますが、このパラメータは *current_page* または *total_pages* では、何も変更されません。

header または *footer* が変更される場合は、既存のヘッダーとフッターを解放し、新しい String を提供してください。ページは左から右、上から下の方向に印刷されます。

XmNxrtGearPSFontMapList　XrtGearPSFontMap *　NULL　CSG

PostScript フォントのリストを対応する X フォントに(場合によってはポイント・サイズと共に)定義します。リストの各エレメントは必ず XrtGearPSFontMap 型にしてください。

```
typedef struct {
  String  font_name, tag;
  String  PS_font;
  double  point_size;
} XrtGearPSFontMap;
```

X フォントは、フォント名(*font_name*)または XmNfontList タグ(*tag*)で表されます。*PS_font* は PostScript フォントです。*point_size* はポイント・サイズで、0.0 であれば、サイズは、フォントから推測されます。リストの一番最後のアイテムの *PS_font* メンバは NULL にセットされていなければなりません。

XmNxrtGearRepaint　Boolean　True　CSG

ウィジェットを再描画するかどうか指定します。このリソースが True にセットされている時は再描画されます。

XmNxrtGearResizeCallback　XtCallbackList　NULL　CSG

ウィジェットがリサイズされた時に呼び出されるコールバックを指定します。各ルーチンには、XrtGearResizeCallbackStruct のポインタが渡されます。コールバックは、*XtAddCallback()* を使用してリストに追加してください。コールバックは次のように構成されます。

```
void
CallbackProcedure(
  Widget  widget,
  XtPointer   client_data,
  XtPointer   call_data /* points to an
            XrtGearResizeCallbackStruct */
)
```

XrtGearResizeCallbackStruct は次のように定義されます。

```
typedef struct {
  XrtGearReason   reason; /* read-only */
  Xevent      *event; /* read-only */
  Dimension       width;
  Dimension       height;
} XrtGearResizeCallbackStruct;
```

reason は XRTGEAR_REASON_RESIZE です。*event* はコールバックを呼び出した XEvent で、コールバックがプログラムによって呼び出される時は NULL です。*width* は新しい幅、*height* は新しい高さです。

XmNxrtGearScrollBarHoriz　Widget　dynamic　CSG

横方向のスクロールバーのウィジェット ID を指定します。

XmNxrtGearScrollBarHorizDisplayPolicy　XrtGearDisplayPolicy　XRTGEAR_DISPLAY_AS_NEEDED　CSG

横方向のスクロールバーの表示をコントロールします。有効な値は次の通りです。

XRTGEAR_DISPLAY_AS_NEEDED	スクロールバーは必要時に表示する
XRTGEAR_DISPLAY_ALWAYS	スクロールバーを常に表示する
XRTGEAR_DISPLAY_NEVER	スクロールバーは表示しない

XmNxrtGearScrollBarVert　Widget　dynamic　CSG

縦方向のスクロールバーのウィジェット ID を指定します。

XmNxrtGearScrollBarVertDisplayPolicy　XrtGearDisplayPolicy　XRTGEAR_DISPLAY_AS_NEEDED　CSG

縦方向のスクロールバーの表示をコントロールします。有効な値は次の通りです。

XRTGEAR_DISPLAY_AS_NEEDED	スクロールバーは必要時に表示する
XRTGEAR_DISPLAY_ALWAYS	スクロールバーを常に表示する
XRTGEAR_DISPLAY_NEVER	スクロールバーは表示しない

XmNxrtGearSelectionCallback XtCallbackList NULL CSG

ノードが選択または選択解除された時に呼び出されるコールバックを指定します。各ルーチンには、XrtGearSelectCallbackStruct のポインタが渡されます。コールバックは、*XtAddCallback()*を使用してリストに追加して下さい。コールバックは次のように構成されます。

```
void
CallbackProcedure(
  Widget  widget,
  XtPointer   client_data,
  XtPointer   call_data /* points to an
            XrtGearSelectCallbackStruct */
)
```

XrtGearSelectCallbackStruct は次のように定義されます。

```
typedef struct {
  XrtGearReason   reason;     /* read-only */
  XEvent      *event;     /* read-only */
  Widget      container;  /* read-only */
  Widget      node;
  XrtGearObject   selected_node_list; /* read-only */
  Boolean     doit;
} XrtGearSelectCallbackStruct;
```

reason は XRTGEAR_REASON_SELECT_BEGIN(選択開始)、XRTGEAR_REASON_SELECT_END(選択終了)、XRTGEAR_REASON_UNSELECT_BEGIN(選択解除開始)、XRTGEAR_REASON_UNSELECT_END(選択解除終了)のどれかです。*event* はコールバックを呼び出した XEvent で、コールバックがプログラムによってよびだされた場合は NULL です。*container* は親の Outliner ウィジェットです。*node* は選択または選択解除されるノードです。*selected_node_list* は選択されたノード(複数のノードが選択された場合はノードのリスト)です。

reason が XRTGEAR_REASON_SELECT_BEGIN の時に *doit* を False にセットすると、選択の操作を不可にすることができます。XmNxrtGearSelectionPolicy が XRTGEAR_SELECTION_POLICY_RANGE または XRTGEAR_SELECTION_POLICY_MULTIPLE であれば、*reason* が XRTGEAR_REASON_UNSELECT_BEGIN の時に *doit* を False にして、選択解除の操作を不可にすることができます。デフォルトでは、*doit* は True で選択または選択解除が行われます。

XmNxrtGearSelectionPolicy が XRTGEAR_SELECTION_POLICY_SINGLE の場合は、XmNxrtGearSelectionCallback で指定されるコールバックは、次の XrtGearSelectCallbackStruct 構造体の *reason* メンバと共に次の順序で呼び出されます。

XRTGEAR_REASON_SELECT_BEGIN	新しく選択されたノード上
XRTGEAR_REASON_UNSELECT_BEGIN	前に選択されていたノード上
XRTGEAR_REASON_UNSELECT_END	前に選択されていたノード上
XRTGEAR_REASON_SELECT_END	新しく選択されたノード上

XmNxrtGearSelectedNodeList XrtGearObject NULL G

選択されたノードのリストのポインタを指定します。

XmNxrtGearSelectionPolicy XrtGearSelectionPolicy _SELECTION_POLICY_SINGLE_AUTO_SELECT CSG

ノードを選択する際のノードの範囲を指定します。有効な値は次の通りです。

XRTGEAR_SELECTION_POLICY_NONE	選択不許可モード。ノードはドラッグすることはできますが、選択はできません。

XRTGEAR_SELECTION_POLICY_SINGLE　シングル・モード。一度にノードが一つしか選択できず、選択はトグルです(選択されているノードを再度選択すると、選択が解除されます)。選択されたノードだけがドラッグできます。

XRTGEAR_SELECTION_POLICY_SINGLE_AUTO_SELECT　自動シングル・モード。一度にノードが一つしか選択できません。選択はトグルにはなりません(選択されているノードを再度選択しても、選択は解除されません)。選択されたノードだけがドラッグできます。

XRTGEAR_SELECTION_POLICY_RANGE　範囲モード。複数の(隣接している)ノードが選択できます。選択されているノードだけがドラッグできます。

XRTGEAR_SELECTION_POLICY_MULTIPLE　マルチ・モード。ノードが隣接しているいないに関わらず複数のノードが選択できます。選択されたノードだけがドラッグできます。

XRTGEAR_SELECITON_POLICY_MULTIPLE_AUTO_UNSELECT　マルチ自動モード。カレント・ノードで選択/選択解除を行う前に、全ての選択されているノードをクリアします。選択されているノードだけがドラッグできます。

このリソースは XmNxrtGearSelectionRestrictions リソースと共に使用され、エンドユーザが実行できる選択操作を指定します。このリソースの値が変更された時は、エンドユーザによって選択されている全てのノードは解除されます。

XmNxrtGearSelectionRestrictions XrtGearSelectionRestrictions XRTGEAR_RESTRICT_NOTHING CSG

何が選択可能かどうかを指定します。有効な値は、次の値の組み合わせ(OR)です。

XRTGEAR_RESTRICT_NOTHING　制限なし

XRTGEAR_RESTRICT_ITEMS　アイテムは選択できない

XRTGEAR_RESTRICT_FOLDERS　フォルダは選択できない

XRTGEAR_RESTRICT_MULTI_LEVEL　前に選択されたいるノードの兄弟(sibling)だけ選択できる

XRTGEAR_RESTRICT_ALL　何も選択できない

このリソースは XmNxrtGearSelectionPolicy と一緒に使用され、エンドユーザが実行できる選択操作を指定します。このリソースの値が変更された時は、エンドユーザによって選択されている全てのノードは解除されます。

XmNxrtGearShadowThickness Dimension 3 CSG

ウィジェットの周囲に描かれる罫線の幅を指定します。

XmNxrtGearTextEditor Widget dynamic G

セル編集に使用されるテキスト・ウィジェットの ID を指定します。このリソースは read-only です。このリソースの値は、セルが現在編集されているかどうかには依存せず、outliner が一度 realize されれば、常に有効なテキスト・ウィジェットを返します。

XmNxrtGearTextHorizClipPolicy XrtGearTextClipPolicy XRTGEAR_TRUNCATE_WITH_ARROW CSG

テキストを全て表示できるスペースがない場合、どのようにそのテキストを表示するか指定します。有効な値は次の通りです。

XRTGEAR_TRUNCATE	テキストを途中で切る
XRTGEAR_TRUNCATE_WITH_ARROW	テキストを途中で切り、矢印で示す
XRTGEAR_COMPRESS_WITH_ELLIPSIS	ストリングの最初と最後部分を表示し、省略部分は点で示す

XmNxrtGearTreeData String NULL CSG

ツリーにロードされるデータのファイル名を指定します(ファイルの内容に関する情報も一緒にロードすることができます)。このリソースは、主にリソース・ファイルや GUI ビルダに使用されます。

リソース・ファイルからロードされる時の XmNxrtGearTreeData の構文は次の通りです。

filename indent separator open_escape closed_escape

この構文の *filename* 以外のエレメントは全てオプションです。一つのエレメントが省略されると、それに続く他のエレメントも提供できません。

filename はデータを含むファイルの名前で、各行が各ノードを表します。*indent* はノードのインデント・レベルを指定するストリングです。*indent* が DEFAULT であったり、与えられていなかったりした場合は、 \t が使用されます。

separator はテキストを区分するストリングで、複数列を表示する時に使用されます。*separator* が DEFAULT であったり、与えられていなかったりした場合は、 , が使用されます。

open_escape は、ノード・テキストではない、指示にあたる部分の始まりを示すストリングです。*open_escape* が DEFAULT であったり、与えられていなかったりした場合は、 (が使用されます。

closed_escape は、ノード・テキストではない、指示にあたる部分の終わりを示すストリングです。*closed_escape* が DEFAULT であったり、与えられていなかったりした場合は、) が使用されます。

XmNxrtGearTreeData を NULL にセットすると、全ての既存のノードが破棄されます。

XmNwidth Dimension determined at runtime CSG

ウィジェットの幅を pixel で指定します。

XmNxrtGearWidthChar char 'M' CSG

XmNxrtGearWidthSizingPolicy が XRTGEAR_SIZE_CHAR にセットされている場合に、XmNxrtNodeIndent で文字幅を計算する時に使用される文字を指定します。文字幅の計算に使用されるフォントは、XmNfontList で指定されるリストの最初のフォントです。このリソースが '\0' にセットされていれば、そのフォントで一番幅の広い文字が使用されます。

XmNxrtGearWidthPreferredCount int 40 CSG

ツリーの表示部分の幅を計算する時に使用される数を指定します。このリソースは、XmNxrtGearWidthSizingPolicy が XRTGEAR_SIZE_COLUMN または XRTGEAR_SIZE_CHAR の場合にのみ使用されます。

XmNxrtGearWidthSizingPolicy が XRTGEAR_SIZE_COLUMN にセットされている場合は、XmNxrtGearWidthPreferredCount の値は、列幅の平均値に乗算されて、ツリーに要求される表示幅が計算されます。

XmNxrtGearWidthSizingPolicy が XRTGEAR_SIZE_CHAR にセットされている時は、XmNxrtGearWidthPreferredCount の値は、XmNxrtGearWidthChar の値に乗算されて、ツリーに要求される表示幅が計算されます。

XmNxrtGearWidthResizePolicy XrtGearWidgetResizePolicy XRTGEAR_WIDGET_RESIZE_VARIABLE CSG

ウィジェットのリサイズの決まりを指定します。有効な値は次の通りです。

XRTGEAR_WIDGET_RESIZE_VARIABLE	XmNxrtGearWidthSizingPolicy で決定されるサイズにリサイズする
XRTGEAR_WIDGET_RESIZE_GROW_ONLY	ウィジェットが大きくなる場合にだけリサイズする
XRTGEAR_WIDGET_RESIZE_FIXED	親がリクエストしない限りリサイズしない

XmNxrtGearWidthSizingPolicy XrtGearSizingPolicy XRTGEAR_SIZE_CHAR CSG

ツリーの表示部分の幅の計算方法を指定します。有効な値は次の通りです。

XRTGEAR_SIZE_PIXEL	XmNxrtGearWidthPreferredCount の値を使用して幅を指定
XRTGEAR_SIZE_CHAR	XmNxrtGearWidthChar の値に XmNxrtGearWidthPreferredCount の値を乗算して幅を計算
XRTGEAR_SIZE_COLUMN	列幅の平均値に XmNxrtGearWidthPreferredCount の値を乗算して幅を計算
XRTGEAR_SIZE_VARIABLE	閉じられていない(non-collapsed)ノードの幅を使用して幅を計算

XmNxrtGearWidthVirtual int dynamic G

ウィジェットの仮想的な幅を pixel で指定します。この値は、横方向のスクロールバーによって保持されます。幅が ULONG_MAX より大きい場合は、この値は ULONG_MAX にセットされます。

7.13.2 XrtNodeObject リソース

XmNeditable Boolean True CSG

このノードに属するノード・ラベルの編集をユーザに許可するかどうかをコントロールします。

XmNxrtGearLabel XrtGearObject NULL CSG

ノードに表示するテキストを指定します。これは、XrtString オブジェクト(String か XmString) または、XrtList オブジェクトです。テキストが XrtList オブジェクトとして指定されている場合は、このオブジェクトは XrtStrings のリストです。

XmNxrtGearNodeStyle Widget NULL CSG

ノード・スタイルを指定します。このリソースが XrtString オブジェクトにセットされた場合は、ノード・オブジェクトが、このリソースの値とマッチする名の NodeStyle オブジェクトを探します。

マッチするオブジェクトがみつからない場合や、このリソースが NULL にセットされた場合は、ノード・オブジェクトは Outliner の XmNxrtGearNodeStyleList で指定されるリストの最初の NodeStyle オブジェクトを使用します。ノード・スタイルが定義されていない場合は、Outliner によって定義されているデフォルトのノード・スタイルが使用されます。

XmNxrtGearSelected Boolean False CSG

このノードが選択されるかどうか指定します。

XmNuserData XtPointer NULL CSG

ノードに関連付けられた、ユーザ・データのポインタを提供します。ここにあるデータの解放は、ユーザの役割です。

7.13.3 XrtNodeFolder オブジェクトのリソース

XmNxrtGearFolderState XrtGearFolderState XRTGEAR_FOLDERSTATE_OPEN_ALL CSG

フォルダ・ノードの現在の状態を指定します。有効な値は次の通りです。

XRTGEAR_FOLDERSTATE_CLOSED	フォルダは閉、子は表示されない
XRTGEAR_FOLDERSTATE_OPEN_FOLDERS	フォルダは開、フォルダの子だけ表示
XRTGEAR_FOLDERSTATE_OPEN_ITEMS	フォルダは開、アイテムだけ表示
XRTGEAR_FOLDERSTATE_OPEN_ALL	フォルダは開、子は全て表示

この値は、プログラムまたはエンドユーザのインタラクションによってコントロールされます。Outliner ウィジェットの XmNxrtGearFolderStateMask リソースは、どの状態が有効であるかどうかをコントロールします。

XmNxrtGearNodeChildList XrtGearObject NULL G

ノードの全ての子のリストのポインタを指定します。

7.13.4 XrtNodeStyle オブジェクトのリソース

XmNbackground Pixel inherited CSG

このスタイルでの、非選択状態のノードの背景色を指定します。このリソースがセットされていない場合は、Outliner ウィジェットの XmNbackground の値が使用されます。

XmNxrtGearBackgroundSelected Pixel ***下記参照*** CSG

選択されているノードの背景色を指定します。このリソースがセットされていない場合は、既存の背景色の反対色が使用されます。

XmNxrtGearFolderShortcutShow　Boolean　True　CSG

このスタイルを持つフォルダ･ノードの子のショートカット･ボタンを表示するかどうか指定します。ショートカット･ボタンは、それが表示できるように充分なスペースが Outliner の XmNxrtGearNodeIndent によって設定されている時にのみ表示されます。False にセットされている場合は、このスタイルのフォルダ･ノードの子に対する XmNxrtGearFolderShortcutShow の設定を上書きします。

XmNfontList　XmFontList　inherited　CSG

非選択状態のノードを描画する時に使用するフォントを指定します。

XmNxrtGearFontListSelected　XmFontList　NULL　CSG

選択されたノードを描画する時に使用するフォントを指定します。このリソースがセットされていない時は、XmNfontList で指定されているフォントが使用されます。

XmNforeground　Pixel　inherited　CSG

非選択状態のノードの前景色を指定します。このリソースがセットされていない場合は、Outliner ウィジェットの XmNforeground が使用されます。

XmNxrtGearForegroundSelected　Pixel　***下記参照***　CSG

選択されたノードの前景色を指定します。このリソースがセットされていない場合は、既存の前景色の反対色が使用されます。

XmNheight　Dimension　dynamic　G

ノードの高さを指定します。

XmNxrtGearIconClosed　XrtGearIcon　　CSG

フォルダの状態が XRTGEAR_FOLDERSTATE_CLOSED である NodeFolder オブジェクトが非選択状態であることを示すアイコンを指定します。

XmNxrtGearIconClosedSelected　XrtGearIcon　　CSG

フォルダの状態が XRTGEAR_FOLDERSTATE_CLOSED である NodeFolder オブジェクトが選択されていることを示すアイコンを指定します。

XmNxrtGearIconItem　XrtGearIcon　　CSG

非選択状態のアイテムに使用されるアイコンを指定します。

XmNxrtGearIconItemSelected　XrtGearIcon　　CSG

選択されているアイテムに使用されるアイコンを指定します。

XmNxrtGearIconOpen　XrtGearIcon　　CSG

フォルダの状態が XRTGEAR_FOLDERSTATE_CLOSED 以外である NodeFolder オブジェクトが非選択状態であることを示すアイコンを指定します。

XmNxrtGearIconOpenSelected XrtGearIcon CSG

フォルダの状態が XRTGEAR_FOLDERSTATE_CLOSED 以外である NodeFolder オブジェクトが選択されていることを示すアイコンを指定します。

XmNxrtGearLayoutSpacing Dimension 5 CSG

アイコンとラベル・ストリングの間の距離を pixel で指定します。

XmNxrtGearLineColor Pixel XmNforeground **から継承** CSG

フォルダ・ノードの子を結ぶ線の色を指定します。ショート・カットボタンの色と異なる色で線を描くことができます。

XmNxrtGearLineStyle XrtGearLineStyle XRTGEAR_LINESTYLE_DOTTED CSG

フォルダ・ノードの子を結ぶ線のスタイルを指定します。有効な値は次の通りです。

XRTGEAR_LINESTYLE_NONE	線は描かない
XRTGEAR_LINESTYLE_DOTTED	点線を描画
XRTGEAR_LINESTYLE_SOLID	実線を描画

XmNxrtGearLineWidth Dimension 1 CSG

フォルダ・ノードの子を結ぶ線の幅を指定します。

XmNxrtGearShortcutColor Pixel inherited CSG

このスタイルのフォルダ・ノードの全ての子のショートカット・ボタンの色を指定します。

XmNxrtGearShortcutShow Boolean True CSG

ショートカット・ボタンを表示するかどうか指定します。ショートカット・ボタンは、それが表示できるように充分なスペースが Outliner の XmNxrtGearNodeIndent によって設定されている時にのみ表示されます。

XmNxrtGearShortcutSize Dimension 9 CSG

ショートカット・ボタンの幅と高さを pixel で指定します。

7.13.5 XrtColumn Object

XmNbackground Pixel inherited CSG

ラベルの背景色を指定します。このリソースがセットされていない場合は、Outliner ウィジェットの XmNbackground の値が使用されます。

XmNeditable Boolean True CSG

この列にあるノード・ラベルの編集をユーザに許可するかどうかをコントロールします。False にセットされている場合は、その列のノードやフォルダの設定を上書きします。

XmNfontList XmFontList inherited CSG

列が選択されていない時のラベルに使用されるフォントを指定します。

XmNxrtGearFontListSelected XmFontList NULL CSG

列が選択されている時のラベルに使用されるフォントを指定します。このリソースがセットされていない場合は、`XmNfontList` で指定されるフォントが使用されます。

XmNforeground Pixel inherited CSG

ラベルの前景色を指定します。このリソースがセットされていない場合は、Outliner ウィジェットの `XmNforeground` リソースが使用されます。

XmNheight Dimension dynamic CSG

列の高さを pixel で指定します。

XmNxrtGearLabel XrtGearObject NULL CSG

列ラベルのテキストを指定します。

XmNxrtGearLabelAlignment XrtGearAlignment XRTGEAR_ALIGN_CENTER CSG

列ラベルのテキストの行揃えを指定します。有効な値は次の通りです。

`XRTGEAR_ALIGN_BEGINNING`	ラベルの開始点に配置
`XRTGEAR_ALIGN_CENTER`	ラベルの中央に配置
`XRTGEAR_ALIGN_END`	ラベルの終了点に配置

XmNxrtGearMarginLeft Dimension 2 CSG

列の左端と列データの開始点との間を pixel で指定します。

XmNxrtGearMarginRight Dimension 2 CSG

列の右端と列データの終了点との間を pixel で指定します。

XmNxrtGearNodeAlignment XrtGearAlignment XRTGEAR_ALIGN_BEGINNING CSG

列のノード・テキストの行揃えを指定します。有効な値は次の通りです。

`XRTGEAR_ALIGN_BEGINNING`	列の開始点に配置
`XRTGEAR_ALIGN_CENTER`	列の中央に配置
`XRTGEAR_ALIGN_END`	列の終了点に配置

このリソースは、(階層を示す)最初の列の表示には影響しません。

XmNxrtGearResizeCallback XtCallbackList NULL CSG

列がリサイズされた時に呼び出されるコールバックのリストを指定します。各ルーチンには、`XrtGearEnhancedSizeCallbackStruct` のポインタが渡されます。コールバックは、*XtAddCallback()*を使用してリストに追加してください。コールバックは次のように構成されます。

```
void
CallbackProcedure(
```

```
        Widget  widget,
        XtPointer   client_data,
        XtPointer   call_data /* points to an
                    XrtGearEnhancedSizeCallbackStruct */
      )
```

XrtGearEnhancedSizeCallbackStruct は次のように定義されます。

```
      typedef struct {
        XrtGearReason   reason;      /* read-only */
        Xevent      *event;      /* read-only */
        Dimension       width;
        Dimension       height;
        Dimension       old_width;  /* read-only */
        Dimension       old_height; /* read-only */
        Boolean     doit;
      } XrtGearEnhancedSizeCallbackStruct;
```

reason は XRTGEAR_REASON_RESIZE_BEGIN または XRTGEAR_REASON_RESIZE_END のどちらかです。*event* は NULL。*width* は新しい幅、*height* は新しい高さで、これらは *reason* が XRTGEAR_REASON_RESIZE_BEGIN の時に修正できます。*old_width* と *old_height* はリサイズ前の幅と高さです。

reason が XRTGEAR_REASON_RESIZE_BEGIN の時に、*doit* を False にセットしてリサイズの操作を不可にすることができます。

このコールバック構造体は、(Outliner の XmNxrtGearResizeCallback に使用される)XrtGearResizeCallbackStruct コールバック構造体と互換性があります。(XrtGearEnhancedResizeCallbackStruct 構造体に使用される)XRTGEAR_REASON_RESIZE_END は、XmNxrtGearResizeCallbackStruct 構造体に使用される XRTGEAR_REASON_RESIZE と同じです。

XmNxrtGearSelected　Boolean　False　CSG

この列が選択されているかどうか指定します。

XmNwidth　Dimension　100　CSG

列幅を pixel で指定します。

XmNxrtGearWidthMaximum　Dimension　10000　CSG

エンドユーザが列をリサイズできる最大幅を指定します。このリソースは、プログラムによるノードのリサイズには影響しません。

XmNxrtGearWidthMinimum　Dimension　10　CSG

エンドユーザが列をリサイズできる最小幅を指定します。このリソースは、プログラムによるノードのリサイズには影響しません。

XmNxrtGearWidthPreferredCount　int　10　CSG

XmNxrtGearWidthSizingPolicy が XRTGEAR_SIZE_CHAR にセットされている時、必要な列幅を文字で計算する時に使用される数を指定します。

XmNxrtGearWidthResizePolicy　XrtGearColumnResizePolicy　XRTGEAR_COLUMN_RESIZE_ALL　CSG

エンドユーザがどのように列をリサイズできるか指定します。有効な値は次の通りです。

XRTGEAR_COLUMN_RESIZE_ALL	列とラベルのリサイズが可能
XRTGEAR_COLUMN_RESIZE_LABELS_ONLY	列ラベルのリサイズだけ可能
XRTGEAR_COLUMN_RESIZE_NONE	リサイズはできない

XmNxrtGearWidthSizingPolicy XrtGearSizingPolicy XRTGEAR_SIZE_VARIABLE CSG

列幅を計算する方法を指定します。有効な値は次の通りです。

XRTGEAR_SIZE_PIXEL	XmNwidth の値を使用して列幅を指定する
XRTGEAR_SIZE_CHAR	XmNxrtGearWidthPreferredCount の値に親の Outliner ウィジェットの XmNxrtGearWidthChar の値を乗算し高さを計算する
XRTGEAR_SIZE_LABEL_VARIABLE	列ラベルの理想的な幅を使用して列幅を計算する
XRTGEAR_SIZE_VARIABLE	(その列の)最も幅が広いノード・ラベルの幅を使用して列幅を計算する

XmNx Position dynamic G

(Outliner ウィジェットの左に対応する)列の開始点の X-offset を指定します。Outliner は、このオフセットを使用して列ラベルとノード・ラベルを描画する場所を計算します。

7.13.6 Outliner によって継承されるリソース

Resource	Inherited From
XmNaccelerators	Core
XmNancestorSensitive	Core
XmNbackground	Core
XmNbackgroundPixmap	Core
XmNborderColor	Core
XmNborderPixmap	Core
XmNborderWidth	Core
XmNbottomShadowColor	XmManager
XmNbottomShadowPixmap	XmManager
XmNchildren	Composite
XmNcolormap	Core
XmNdepth	Core
XmNdestroyCallback	Core
XmNforeground	XmManager
XmNheight	Core
XmNhelpCallback	XmManager
XmNhighlightColor	XmManager
XmNhighlightPixmap	XmManager

Resource	Inherited From
XmNinitialFocus	XmManager
XmNinitialResourcesPersistent	Core
XmNinsertPosition	Composite
XmNmappedWhenManaged	Core
XmNnavigationType	XmManager
XmNnumChildren	Composite
XmNscreen	Core
XmNsensitive	Core
XmNshadowThickness	XmManager
XmNstringDirection	XmManager
XmNtopShadowColor	XmManager
XmNtopShadowPixmap	XmManager
XmNtranslations	Core
XmNtraversalOn	XmManager
XmNuserData	XmManager
XmNwidth	Core
XmNx	Core
XmNy	Core

7.13.7 Objects によって継承されるリソース

Resource	Inherited From
XtNdestroyCallback	Object

7.14 手続きと方法

7.14.1 Outliner の手続きと方法

XmCreateXrtOutliner()

Outliner ウィジェットを作成します。

```
Widget
XmCreateXrtOutliner(
  Widget  parent,
  String  name,
  Arg     *arglist,
  int     argcount
)
```

parent は親、*name* は作成されるウィジェット名、*arglist* は argcount 引数のリストです。

XmIsXrtOutliner()

ウィジェットが Outliner ウィジェットであれば `True` を返します。

```
Boolean
XmIsXrtOutliner(
  Widget  widget
)
```

XrtGearCvtStringToXmString()

String を XmString に変換します。

```
XmString
XrtGearCvtStringToXmString(
  String  string
)
```

XrtGearCvtXmStringToString()

XmString を String に変換します。ストリングは使用後解放してください。

```
String
XrtGearCvtXmStringToString(
  XmString    string
)
```

XrtGearMrmInitialize()

UIL アプリケーションで使用するために、Mrm に XRT/gear ウィジェットを登録します。この手続きは、*MrmInitialize()*の後に呼び出して下さい。

```
void
XrtGearMrmInitialize(void)
```

XrtGearOutlinerCancelEdit()

インタラクティブなセルの編集をキャンセルし、使用されているエディタをアンマップします。ユーザによって入力されたテキストは破棄されます。検証コールバックは呼び出されません。

*XrtGearOutlinerCancelEdit()*は、何も編集されていなかったり、引数が無効であると `False` を返します。

```
Boolean
XrtGearOutlinerCancelEdit(
      Widget  Outliner,
      )
```

outliner は XRT/gear outliner ウィジェットです。

XrtGearOutlinerCommitEdit()

ノードに対するインタラクティブな編集を確定し、ユーザによる変更を保管します。オプションで現在使用されているエディタをアンマップすることができます。`XmNxrtGearNodeValidateCallback` リストのコールバックが、編集が有効かどうかを検証するために呼び出されます。この方法は、引数が無効であると `False` を返します。

```
Boolean
XrtGearOutlinerCommitEdit(
      Widget  Outliner,
      Boolean unmap
      )
```

outliner は XRT/gear outliner ウィジェットです。*unmap* は、この方法が成功した時に、使用されているエディタをアンマップするかどうかを指定します。

XrtGearOutlinerDrawPS()

PostScript フォントとデバイスに依存しない PostScript オペレータを使用して、Outliner ツリーの encapsulated PostScript(EPSF-2.0)出力を生成します。この方法は、*arglist* を取る以外は、*XrtGearOutlinerVaDrawPS()*と同じです。*XrtGearOutlinerDrawPS()*は出力されるページ数を返します(0 ページの場合はエラー)。

```
int
XrtGearOutlinerDrawPS(
  Widget  outliner,
  FILE    *file,
  XrtGearPSarg    *arglist,
  int     argcount
)
```

outliner は Outliner ウィジェット。*file* は既に開かれ書き込み可能なファイルを指します。*arglist* は argcount 引数のリストです。

XrtGearOutlinerMoveColumns()

Outliner ウィジェット内で列を移動します。

```
Boolean
XrtGearOutlinerMoveColumns(
  Widget  outliner,
  int     destination,
  int     source,
  int     num_to_move
)
```

outliner は Outliner ウィジェット。*destination* は移動先の列、*source* は移動される列です。*num_to_move* は、移動される列数です。

移動された列は、下記の要因によって、移動先の列(destination column)の前か後に配置されます。

- 移動される列が移動先の列の直ぐ左にある場合は、この列は、右側に配置される
- その他の場合は、移動された列は全て移動先の列の左側に配置される

インデックスが0の列は階層を示す列で、移動したり、移動先の列にはできません。

XrtGearOutlinerNodeFromXEvent()

XEvent が起きたノードを返します。

```
Widget
XrtGearOutlinerNodeFromXEvent(
  Widget      outliner,
  XEvent      *event,
  XrtGearLocation *location
)
```

outliner はイベントが起きたノードを持つ Outliner ウィジェット。*event* は検知された XEvent です。*location* はイベントが起こった場所で、次の値です。

XRTGEAR_LOCATION_ON_NODE	イベントはノード上
XRTGEAR_LOCATION_ON_SHORTCUT	イベントはショートカット・ボタン上
XRTGEAR_LOCATION_ON_LABEL	イベントは列ラベル上
XRTGEAR_LOCATION_NOWHERE	イベントはノード内では起きていない

location が NULL の場合は、イベントの場所が返されません。

XrtGearOutlinerNodeFromXY()

(ルート・ウィンドウに相対する)XY 位置にあるノードを返します。

```
Widget
XrtGearOutlinerNodeFromXY(
  Widget      outliner,
  int         root_x,
  int         root_y,
  XrtGearLocation *location
)
```

outliner は、位置が確認されたノードを持つ Outliner ウィジェット。*root_x* は XY 位置の X 座標、*root_y* は Y 座標です。

location は返された位置で、次の値です。

XRTGEAR_LOCATION_ON_NODE	位置はノード上
XRTGEAR_LOCATION_ON_SHORTCUT	位置ショートカット・ボタン上
XRTGEAR_LOCATION_ON_LABEL	位置は列ラベル上
XRTGEAR_LOCATION_NOWHERE	位置はノード内ではない

location が NULL の場合は、位置は返されません。

XrtGearOutlinerTraverseToCell()

カレント・セルを新しい位置に移動します。*XrtGearOutlinerCommitEdit()*が最初に呼び出され、カレント・セルへの編集を確定します。*XrtGearOutlinerCommitEdit()*が `False` を返すと、*XrtGearOutlinerTraverseToCell()*も `False` を返し、XmText ウィジェットは移動しません。*XrtGearOutlinerCommitEdit()* が `True` を返すと、*XrtGearOutlinerTraverseToCell()*は、指定されたセルへトラバースしようとします。目的地のセルが編集可でない場合は、*XrtGearOutlinerTraverseToCell()*は `False` を返します。`XmNxrtGearNodeEnterCellCallback` がトラバースを許可しない時は、*XrtGearOutlinerTraverseToCell()*は `False` を返し、カレント・セルは変化しません。それ以外の場合は、エディタは目的地のセルにマップされます。

セルが outliner の表示されていない部分に存在する場合、そのセルが表示されるまでスクロールされます。また、セルが閉じている branch に存在する場合は、そのセルが表示されるまで、その branch が開かれます。`XmNxrtGearFolderStateMask` がこのアクションを許可しない場合は、*XrtGearOutlinerTraverseTopCell()*は `False` を返し、XmText ウィジェットは移動しません。

```
Boolean
XrtGearOutlinerTraverseToCell(
     Widget  outliner,
     Widget  node,
     Int column,
     )
```

ouliner は XRT/gear の outliner。*node* はトラバースするノードで、*column* は、エディタがマップされる、指定されたノードの列です。

XrtGearOutlinerVaDrawPS()

PostScript フォントとデバイスに依存しない PostScript オペレータを使用して、Outliner ツリーの encapsulated PostScript(EPSF-2.0) 出力を生成します。この方法は、*XtVaSetValues()* と同様に、*varargs*-style インターフェースを使用し、`XrtGearDrawPSarg` 型の属性と値のペアを使用します。*XrtGearOutlinerVaDrawPS()* は出力されるページ数を返します(0 ページの場合はエラー)。

```
int
XrtGearOutlinerVaDrawPS(
  Widget  outliner,
  FILE    *file,
  ...,
  )
```

outliner は Outliner ウィジェット。*file* は、既に開かれ書き込み可能なファイルを指します。下の表は、属性と値の一覧です。

Attribute	Values/Type	Default
XRTGEAR_PS_BORDER_TYPE	XRTGEAR_BORDER_NONE XRTGEAR_BORDER_PLAIN XRTGEAR_BORDER_IN XRTGEAR_BORDER_OUT XRTGEAR_BORDER_ETCHED_IN XRTGEAR_BORDER_ETCHED_OUT XRTGEAR_BORDER_FRAME_IN XRTGEAR_BORDER_FRAME_OUT XRTGEAR_BORDER_BEVEL	XmNxrtGearBorderTypeから継承される
XRTGEAR_PS_COLOR	XRTGEAR_PS_COLOR XRTGEAR_PS_MONO	XRTGEAR_PS_COLOR
XRTGEAR_PS_COLUMN_PAGE_BREAK_LIST	XrtGearObject	NULL
XRTGEAR_PS_FOOTER	String	NULL
XRTGEAR_PS_FOOTER_FONT	String	“Courier“
XRTGEAR_PS_FOOTER_MARGIN	double	本文解説参照
XRTGEAR_PS_FOOTER_POINT_SIZE	double	12.0
XRTGEAR_PS_HEADER	String	NULL
XRTGEAR_PS_HEADER_FONT	String	“Courier”
XRTGEAR_PS_HEADER_MARGIN	double	本文解説参照
XRTGEAR_PS_HEADER_POINT_SIZE	double	12.0
XRTGEAR_PS_MARGIN_BOTTOM	double	本文解説参照
XRTGEAR_PS_MARGIN_LEFT	double	本文解説参照
XRTGEAR_PS_MARGIN_RIGHT	double	本文解説参照
XRTGEAR_PS_MARGIN_TOP	double	本文解説参照
XRTGEAR_PS_NODE_PAGE_BREAK_LIST	XrtGearObject	NULL
XRTGEAR_PS_NUM_COPIES	int	1
XRTGEAR_PS_ORIENTATION	XRTGEAR_PS_PORTRAIT XRTGEAR_PS_LANDSCAPE	XRTGEAR_PS_PORTRAIT
XRTGEAR_PS_PAPERSIZE_HEIGHT	double	11.0
XRTGEAR_PS_PAPERSIZE_WIDTH	double	8.5
XRTGEAR_PS_SCALE	double XRTGEAR_FIT_TO_PAGE XRTGEAR_FIT_TO_PAGE_WIDTH XRTGEAR_FIT_TO_PAGE_HEIGHT	72.0
XRTGEAR_PS_SHOW_COLUMN_LABELS	XRTGEAR_PS_EDGE_PAGES XRTGEAR_PS_ALL XRTGEAR_PS_NONE	XRTGEAR_PS_EDGE_PAGES

Attribute	Values/Type	Default
XRTGEAR_PS_SHOWPAGE	Boolean	True
XRTGEAR_PS_SHOW_START_NODE	Boolean	True
XRTGEAR_PS_START_NODE	Widget	NULL
XRTGEAR_PS_UNITS	XRTGEAR_PS_INCHES XRTGEAR_PS_CM	XRTGEAR_PS_INCHES

XRTGEAR_PS_BORDER_TYPE は枠線を指定し、表示されている枠線とは異なる枠線を印刷する時に使用できます。有効な値は、Outliner の XmNxrtGearBorderType と同様です。デフォルトは、XmNxrtGearBorderType で指定される枠線です。

XRTGEAR_PS_COLOR は、色を指定し、XRTGEAR_PS_COLOR(カラー)か XRTGEAR_PS_MONO(モノクロ)のどちらかです。XRTGEAR_PS_MONO の場合は、全ての前景色は黒、全ての背景色は白、影は(印刷される場合は)灰色にセットされます。

XRTGEAR_PS_COLUMN_PAGE_BREAK_LIST は、XrtColumn オブジェクトをエレメントとして持つ XrtList オブジェクトです。横方向のページは、これらの列がプリントされる前に区切られます。列が指定されていない場合(デフォルト)は、ページ区切りは自動的に行われます。

XRTGEAR_PS_FOOTER は、出力される各ページにフッターとしてプリントされるストリングを指定します。同様に、XRTGEAR_PS_HEADER は、出力される各ページにヘッダーとしてプリントされるストリングを指定します。

XRTGEAR_PS_FOOTER_FONT と XRTGEAR_PS_HEADER_FONT は、フッターとヘッダーのフォントを使用します。デフォルトは Courier フォントです。

XRTGEAR_PS_FOOTER_POINT_SIZE と XRTGEAR_PS_HEADER_POINT_SIZE は、フッターとヘッダーのポイント･サイズを指定します。デフォルト値は 12.0 です。

XRTGEAR_PS_FOOTER と XRTGEAR_PS_HEADER は特別な文字を組み込むことができます。デフォルトでは、ヘッダーとフッターのテキストは左揃えですが、" | "文字を使用することにより、中央または右揃えにすることができます。始めの" | "は次に続くテキストを中央揃えにし、二番目の" | "はテキストを右揃えにします。その他に、次のシンボルを使用することができます。

#	現在のページ番号
^	ページの合計数
%<X>	現在の日付(*strftime()*フォーマット)
\	次に来る特別文字を文字として扱う

XRTGEAR_PS_FOOTER_MARGIN は、XRTGEAR_PS_FOOTER がセットされている場合に定義できます。これは、フッターとページの端の間のマージンを指定します。デフォルトは、0.25 units(XRTGEAR_PS_UNITS で指定)です。

XRTGEAR_PS_HEADER_MARGIN は、XRTGEAR_PS_HEADER がセットされている場合に定義できます。これは、ヘッダーとページの端の間のマージンを指定します。デフォルトは、0.25 units(XRTGEAR_PS_UNITS で指定)です。

XRTGEAR_PS_MARGIN_TOP と XRTGEAR_PS_MARGIN_BOTTOM は、(イメージではなく)用紙の上部と下部の印刷されない余白を指定します。デフォルトは、0.25 units(XRTGEAR_PS_UNITS で指定)です。

- XRTGEAR_PS_HEADER がセットされ、XRTGEAR_PS_ORIENTATION が XRTGEAR_PS_PORTRAIT になっている場合、XRTGEAR_PS_MARGIN_TOP のデフォルト値は、0.75 units です。
- XRTGEAR_PS_FOOTER がセットされ、XRTGEAR_PS_ORIENTATION が XRTGEAR_PS_PORTRAIT になっている場合、XRTGEAR_PS_MARGIN_BOTTOM のデフォルト値は、0.75 units です。

XRTGEAR_PS_MARGIN_LEFT と XRTGEAR_PS_MARGIN_RIGHT は、(イメージではなく)用紙の左端と右端の印刷されない余白を指定します。デフォルトは、0.25 units(XRTGEAR_PS_UNITS で指定)です。

- XRTGEAR_PS_HEADER がセットされ、XRTGEAR_PS_ORIENTATION が XRTGEAR_PS_LANDSCAPE になっている場合、XRTGEAR_PS_MARGIN_LEFT のデフォルト値は、0.75 units です。
- XRTGEAR_PS_FOOTER がセットされ、XRTGEAR_PS_ORIENTATION が XRTGEAR_PS_LANDSCAPE になっている場合、XRTGEAR_PS_MARGIN_RIGHT のデフォルト値は、0.75 units です。

XRTGEAR_PS_NODE_PAGE_BREAK_LIST は、XrtNode オブジェクトをエレメントとして持つ XrtList オブジェクトです。縦方向のページは、これらの各ノードがプリントされる前に区切られます。ノードが指定されていない場合(デフォルト)は、ページ区切りは自動的に行われます。

XRTGEAR_PS_NUM_COPIES は、印刷部数を指定します。この出力が、他のドキュメントへ挿入される場合は、1 のままにしておいてください。

XRTGEAR_PS_ORIENTATION は、イメージの向きを指定します。XRTGEAR_PS_PORTRAIT にセットされた場合は、画面に表示されているようにプリントされます。XRTGEAR_PS_LANDSCAPE は、Outliner を反時計方向に 90 度回転します。

XRTGEAR_PS_PAPERSIZE_WIDTH と XRTGEAR_PS_PAPERSIZE_HEIGHT は、使用される用紙の幅と高さを指定します。この幅と高さは、イメージの方向によって変化することはありません。用紙は常にプリンタから同じ方向でフィードされ、イメージの向きには関係ないので、用紙の幅と高さは、イメージではなく用紙の上部と左端に対応します。ここで指定される数値は、XRTGEAR_PS_UNITS で設定された単位を使用します。

一般的な用紙のサイズは以下の通りです。

- 8.5 x 11 インチ (レター)
- 8.5 x 14 インチ (リーガル)
- 21.0 x 29.7 センチメートル (A4)
- 14.8 x 21.0 センチメートル (A5)

XRTGEAR_PS_SCALE は、インチに対する pixel で、ウィジェットのイメージのスケーリングを指定します。この属性をスクリーンの解像度にセットしてある場合、スクリーンのイメージと同等の印刷イメージが得られます。(デフォルトの 72.0 は、最も一般的なスクリーン解像度です。)サイズを 200%に拡大する時は XRTGEAR_PS_SCALE を 36.0 に、50%に縮小する時は XRTGEAR_PS_SCALE を 144.0 にセットします。次の列挙された値は、その他のスケーリングを指定します。

- XRTGEAR_FIT_TO_PAGE は、イメージが 1 ページに収まるようにスケーリングします。
- XRTGEAR_FIT_TO_PAGE_WIDTH は、イメージがページ幅に収まるようにスケーリングします。
- XRTGEAR_FIT_TO_PAGE_HEIGHT は、イメージがページの高さに収まるようにスケーリングします。

XRTGEAR_PS_SHOW_COLUMN_LABELS は、ラベルを PostScript 出力させるか指定します。XRTGEAR_PS_ALL は、ラベルを各ページに置きます。XRTGEAR_PS_NONE は、全てのページからラベルを取り去ります。XRTGEAR_PS_EDGE_PAGES は、複数ページの内、端のページにだけラベルを置き、大きなイメージが複数のページに別れたようにプリントします。(1 ページしかない時は、結果は XRTGEAR_PS_ALL と同じになります。)

XRTGEAR_PS_SHOW_START_NODE は、フォルダ・ノードにだけ適用されます。True(デフォルト)の場合は、印刷されるツリーの最初のノード(start node)がプリントされます。最初のノードが Outliner ウィジェットの場合は、この属性は意味を持ちません。

XRTGEAR_PS_START_NODE は、印刷を開始するツリーの最初のノードを指定します。NULL(デフォルト)にセットしてある場合は、印刷はルートから開始されます。

XRTGEAR_PS_UNITS は、XRTGEAR_PAPERSIZE_と XRTGEAR_MARGIN_の属性の測定単位を指定します。

7.14.2 XrtNode と XrtNodeFolder の手続きと方法

XmCreateXrtNode()

ノードを作成します。

```
Widget
XmCreateXrtNode(
  Widget  parent,
  String  name,
  Arg     *arglist,
  int     argcount
)
```

parent は親ウィジェット。*name* は作成されるノード名。*arglist* は *argcount* 引数のリストです。

XmCreateXrtNodeFolder()

フォルダ･ノードを作成します。

```
Widget
XmCreateXrtNodeFolder(
  Widget  parent,
  String  name,
  Arg     *arglist,
  int     argcount
)
```

parent は親ウィジェット。*name* は作成されるフォルダ･ノード名。*arglist* は *argcount* 引数のリストです。

XmIsXrtNode()

引数がノードであれば True を返します。

```
Boolean
XmIsXrtNode(
  Widget  widget
)
```

XmIsXrtNodeFolder()

引数がフォルダ･ノードであれば True を返します。

```
Boolean
XmIsXrtNodeFolder(
  Widget  widget
)
```

XrtGearNodeChangeFolderState()

フォルダの状態をセットします。状態が変化すると、XmNxrtGearFolderStateCallback によって指定されるコールバックが呼び出されます。

```
void
XrtGearNodeChangeFolderState(
  Widget      folder,
  XrtGearFolderState  state
)
```

folder は状態が変更されるフォルダ(通常のフォルダまたは lightweight フォルダ)。*state* は新しい状態です。このパラメータの有効な値は、XmNxrtGearFolderStateMask の解説を参照してください。

XrtGearNodeCmpLabels()

二つのノードを取り、それらのラベルを比較する手続きです。(Outliner の XmNxrtGearNodeCompareProc リソースをセットする時に便利です)。一番めのアイテムが二番目のアイテムより大きい場合は 1、同じ場合は 0、小さい場合は-1 を返します。

```
int
XrtGearNodeCmpLabels(
  XtPointer   item1,
  XtPointer   item2
)
```

item1 と *item2* は、比較されるラベルを持つノードのポインタです。

XrtGearNodeCmpPositionsInTree()

ツリー内の二つのノードの位置を比較します。階層内で一番めのノードが二番めのノードの前にある時は負の値、同じ時は 0、後にある時は正の値を返します。

```
int
XrtGearNodeCmpPositionsInTree(
  Widget  root,
  Widget  node1,
  Widget  node2
)
```

root はツリーのルート・ノード。*node1* と *node2* は比較されるノードです。

XrtGearNodeConstructPath()

ツリーのルート・ノードから、指定したノードへ通じるパスを探し、そのパスにあるノードのラベルで構成される文字ストリングを形成します。形成されたストリングのポインタが返されます。

```
String
XrtGearNodeConstructPath(
  Widget  root,
  Widget  node,
  Boolean include_root,
  String  separator,
  String  label_separator,
  String  buffer
)
```

root はツリーのルート・ノードです。*node* はルートから開始したパスが到達するノードです。*include_root* は、パスにルード・ノード名も含ませるかどうか指定します。*root* が Outliner ウィジェット自身であれば、そのウィジェット名(*XtName()*で返される)がパスに含まれます。*separator* はパスのノードを区分する文字で、NULL であれば"/"が使用されます。*label_separator* は一つのノードの複数ラベルを分ける文字で、NULL であえば",が使用されます。*buffer* が NULL でなければ、形成されたパス名が含まれます。

buffer が NULL であれば、*XrtGearNodeConstructPath()*によって返されたパス名は内部メモリで、いつでも上書き可能です。パス名のコピーを保存するには、*buffer* を使用してください。

XrtGearNodeCreateTreeFromBuffer()

(データ・ファイルのフォーマットになっている)複数行のテキスト・ストリングからツリーまたはサブ・ツリーを作成します。戻り値は、*parent* の値に依存します。

```
Widget
XrtGearNodeCreateTreeFromBuffer(
  Widget      parent,
  char        *buffer,
  Boolean     full_path,
  XrtGearFolderState  initial_state,
  String      indent,
  String      separator,
  String      open_escape,
  String      close_escape,
)
```

parent は新しく作成されるツリーの親です。*buffer* は複数行のテキスト・ストリングで、各行が作成されるツリーの各ノードを表します。(これが有効なウィジェットの下に追加されない場合は、*buffer* に含まれていた情報は失われます。)

*XrtGearNodeCreateTreeFromBuffer()*によって返される値は *parent* の値に依存します。*parent* が Outliner ウィジェットまたは NodeFolder オブジェクトであれば、値が返されます。それ以外の場合は、そのツリーを含むルートの NodeFolder オブジェクトが作成され、これが返されます。

full_path は、各ノードにツリー階層全体の情報が与えられているか(*full_path* が `True`)、またはインデント・レベルから推測されるか(*full_path* が `False`)を示します。*full_path* が `True` であれば、*indent* で指定されているストリングがツリーの階層をコンポーネントに分けます。パフォーマンスを重視する場合は、*full_path* を `False` にしてください。

initial_state は、フォルダの初期状態です。*indent* は、(ある行の)ノードのインデント・レベルを指定するストリングです。*indent* が `NULL` の場合は、"`\t`"(tab 文字)が使用されます。*separator* は、テキストをセクション別に区切るストリングで、複数列のアウトラインを表示する時に使用されます。デフォルトは","(カンマ)です。*open_escape* は、(ノード・テキストではない)指示の部分の開始を示す文字です。*open_escape* が `NULL` の場合は、"`(`"が使用されます。*closed_escape* は、(ノード・テキストではない)指示の部分の終了を示す文字です。*closed_escape* が `NULL` の場合は、"`)`"が使用されます。

XrtGearNodeCreateTreeFromStream()

データ・ファイルまたは他のインプット・ストリームからツリーを作成します。戻り値は、*parent* の値に依存します。

```
Widget
XrtGearNodeCreateTreeFromStream(
  Widget      parent,
  FILE        *stream,
  Boolean     full_path,
  XrtGearFolderState  initial_state,
  String      indent,
  String      separator,
  String      open_escape,
  String      close_escape,
)
```

parent は新しく作成されるツリーの親です。*stream* はデータ・ファイルで、ファイルの各行がツリーの各ノードを表します。(これが有効なウィジェットの下に追加されていない場合は、ファイルに含まれていた情報は失われます。)

*XrtGearNodeCreateTreeFromStream()*によって返される値は *parent* の値に依存します。*parent* が Outliner ウィジェットまたは NodeFolder オブジェクトであれば、値が返されます。それ以外の場合は、そのツリーを含むルートの NodeFolder オブジェクトが作成され、これが返されます。

full_path は、各ノードにツリー階層全体が与えられているか(*full_path* が `True`)、またはインデント・レベルから推測されるか(*full_path* が `False`)を示します。*full_path* が `True` であれば、*indent* で指定されているストリングがツリーの階層をコンポーネントに分けます。パフォーマンスを重視する場合は、*full_path* を `False` にしてください。

initial_state は、フォルダの初期状態です。*indent* は、(ある行の)ノードのインデント・レベルを指定するストリングです。*indent* が NULL の場合は、"\t"(tab 文字)が使用されます。*separator* は、テキストをセクション別に区切るストリングで、これは複数列のアウトラインを表示する時に使用されます。デフォルトは","(カンマ)です。*open_escape* は、(ノード・テキストではない)指示の部分の開始を示す文字です。*open_escape* が NULL の場合は、"("が使用されます。*closed_escape* は、(ノード・テキストではない)指示の部分の終了を示す文字です。*closed_escape* が NULL の場合は、")"が使用されます。

XrtGearNodeCvtLabelToString()

指定されたノードのラベルを印刷可能なストリングとして返します。ノード・ラベルがストリングのリストの場合は、separator 文字で区切られた、リスト内の全てのストリングで構成される印刷可能なストリングが返されます。ラベルがリストである場合もそうでない場合も、形成されたラベルのポインタが返されます。

```
String
XrtGearNodeCvtLabelToString(
  Widget  node,
  String  separator,
  String  buffer
)
```

node は、ラベルが取り出されるノードです。*separator* は区切りを示すストリング。*buffer* が NULL でなければ、形成されたラベルを持ちます。

buffer が NULL であれば、*XrtGearNodeCvtLabelToString()*で返されたラベルは内部メモリで、いつでも上書き可能です。ラベルのコピーを保存する場合は、この *buffer* を使用してください。

XrtGearNodeCvtStringToLabel()

印刷可能なストリングを XrtGearObject に変換します。通常、ノード・ラベルとして使用されます。

```
XrtGearObject
XrtGearNodeCvtStringToLabel(
  String  string,
  String  separator
)
```

string は変換されるストリングです。但し、ストリングの内容は変更されません。*separator* は区切りを示すストリングで、変換されたストリングを複数のラベルに区切ります。

複数のラベルが存在する場合は、返された XrtGearObject は、XrtStrings をエレメントとして持つ XrtList オブジェクトです。*string* によって指定されるストリング全体が一つのラベルであれば、返された XrtGearObject は XrtString です。どちらの場合も、返された XrtGearObject は、XmNxrtGearLabel の値として使用できます。

XrtGearNodeGetChild()

指定されたインデックスに対応する子を返します。該当する子が存在しなければ NULL を返します。

```
Widget
XrtGearNodeGetChild(
  Widget  node,
  int     child_index
)
```

node は Outliner ウィジェットかフォルダ・ノードのオブジェクトです。*child_index* は返される子のインデックスです。

XrtGearNodeGetFirstInTree()

ツリーのルートの最初の子を返します。子が存在しなければ NULL を返します。

```
Widget
XrtGearNodeGetFirstInTree(
  Widget  root
)
```

root はツリーのルート・ノードです。

XrtGearNodeGetFirstNonCollapsedInTree()

ツリー・ルートの最初の閉じられていない(non collapsed)子を返します。該当する子が存在しなければ NULL を返します。

```
Widget
XrtGearNodeGetFirstNonCollapsedInTree(
  Widget  root
)
```

root はツリーのルート・ノードです。

XrtGearNodeGetLastInTree()

ツリーの一番下に表示されるノードを返します。ルート・ノードに子が存在しない場合は、NULL を返します。

```
Widget
XrtGearNodeGetLastInTree(
  Widget  root
)
```

root は、一番下のノードを返すツリーのルート・ノードです。

XrtGearNodeGetLastNonCollapsedInTree()

ノード階層リストの閉じられていない(non collapsed)一番最後の子を返します。ルートに子が存在しなければ NULL を返します。

```
Widget
XrtGearNodeGetLastNonCollapsedInTree(
  Widget  root
)
```

root は、一番最後の閉じられていない子を返すツリーのルート・ノードです。

XrtGearNodeGetLevel()

指定ノードのルート・ノードに対するレベル(generation)を返します。失敗時には NULL を返します。

```
int
XrtGearNodeGetLevel(
  Widget  root,
  Widget  node
)
```

root はツリーのルート・ノードです。*node* は、レベルを取り出すノードです。

XrtGearNodeGetNextInTree()

階層リスト内で、指定されたノードの次に来るノードを返します。

```
Widget
XrtGearNodeGetNextInTree(
  Widget  root,
  Widget  node
)
```

root はツリーのルート・ノードです。*node* は、その次に来るノードが取り出されるノードです。この *node* がツリーの最後のノードであれば、手続きは NULL を返します。*root* が NULL であれば、*root* は、XmNxrtGearNodeChildList が定義される、最後の NULL でないノードと推測されます。

XrtGearNodeGetNextNonCollapsedInTree()

階層リスト内で、指定されたノードの次に来る閉じられていない(non-collapsed)ノードを返します。

```
Widget
XrtGearNodeGetNextNonCollapsedInTree(
  Widget  root,
  Widget  node
)
```

root はツリーのルート・ノードです。*node* は、その次に来る閉じられていないノードが取り出されるノードです。この *node* がツリーの最後の閉じられていないノードであれば、手続きは NULL を返します。*root* が NULL であれば、*root* は、XmNxrtGearNodeChildList が定義される、最後の NULL でないノードと推測されます。

XrtGearNodeGetNumChildren()

XmNxrtGearNodeChildList で指定される、オブジェクトまたはウィジェットの子の数を返します。

```
int
XrtGearNodeGetNumChildren(
  Widget  node
)
```

node は、子の数が返されるウィジェットまたはオブジェクトです。

XrtGearNodeGetPreviousInTree()

階層リスト内で、指定されたノードの前に来るノードを返します。

```
Widget
XrtGearNodeGetPreviousInTree(
  Widget  root,
  Widget  node
  )
```

root はツリーのルート・ノードです。*node* は、その前に来るノードが取り出されるノードです。この *node* がツリーの最初であれば、手続きは NULL を返します。*root* が NULL であれば、*root* は、XmNxrtGearNodeChildList が定義される、最後の NULL でないノードと推測されます。

XrtGearNodeGetPreviousNonCollapsedInTree()

階層リスト内で、指定されたノードの前に来る閉じられていない(non-collapsed)ノードを返します。

```
Widget
XrtGearNodeGetPreviousNonCollapsedInTree(
  Widget  root,
  Widget  node
  )
```

root はツリーのルート・ノードです。*node* は、その前に来る閉じられていないノードが取り出されるノードです。この *node* がツリーの最初の閉じられていないノードであれば、手続きは NULL を返します。*root* が NULL であれば、*root* は、XmNxrtGearNodeChildList が定義される、最後の NULL でないノードと推測されます。

XrtGearNodeGetVisibility()

ノードの現在の表示状態を返します。

```
XmVisibility
XrtGearNodeGetVisibility(
  Widget  node
  )
```

node は表示状態が調べられるノードです。返される XmVisibility 値は、次の値のどれかです。

```
XmVISIBILITY_UNOBSCURED
XmVISIBILITY_PARTIALLY_OBSCURED
XmVISIBILITY_FULLY_OBSCURED
```

XrtGearNodeGetWidgetParent()

ウィジェットに到達するまでノード階層を上がり、そのウィジェットを返します。

```
Widget
XrtGearNodeGetWidgetParent(
  Widget  node
  )
```

node は、親ウィジェットが取り出されるノードです。

XrtGearNodeIsAncestor()

指定されたノードが、他のノードの先祖(ancestor)であれば(=先祖のサブツリーに存在すれば)、True を返します。

```
Boolean
XrtGearNodeIsAncestor(
  Widget  root,
  Widget  ancestor,
  Widget  node
)
```

root はツリーのルートで、NULL であれば推測されます。*ancestor* は指定された先祖ノードです。*node* は指定された子孫(descendant)です。

XrtGearNodeIsInCollapsedBranch()

指定されたノードが閉じられた枝(=閉じられたフォルダのサブツリーに存在すれば)、True を返します。

```
Boolean
XrtGearNodeIsInCollapsedBranch(
  Widget  root,
  Widget  node
  )
```

root はツリーのルートで、NULL であれば推測されます。*node* はチェックされるノードです。

XrtGearNodeMakeVisible()

指定されたノードが画面上で表示されるようにします。

```
void
XrtGearNodeMakeVisible(
  Widget  node
)
```

node は表示されるノードです。既に表示されているノードの場合は変化はありません。

XrtGearNodeSetParent()

ノードの親を変更します。親の変更が作成すれば True、何かの理由で失敗すれば False を返します。

```
Boolean
XrtGearNodeSetParent(
  Widget  node,
  Widget  new_parent,
  Widget  sibling
)
```

node は親が変更されるノードです。*new_parent* は新しい親で必ず NodeFolder です。*sibling* が NULL でなければ、*node* は *sibling* の直前に配置されます。(この場合、*sibling* は *new_parent* の子です。) *sibling* が NULL で *new_parent* に他に子ある場合は、*node* は子リストの最後に配置されます。(XmNxrtGearAutoSort が True であれば、*sibling* の値に関わらず、*node* は子リストでソートされた順に配置されます。)

XrtGearNodeSetValuesOnTree()

指定されたルート・ノードの全ての子(またはルート・ノード自身)の arg リストをセットします。ルートは lightweight ノードとすることもできます。

```
void
XrtGearNodeSetValuesOnTree(
      Widget      root,
      Arg     *args,
      Cardinal    num_args,
      Boolean     Set_on_root
      )
```

XrtGearNodeSortTree()

ツリーをソートします。各兄弟(*sibling*)のグループは、別々にソートされます。

```
Boolean
XrtGearNodeSortTree(
  Widget  root
  )
```

root はソートされるツリーのルートです。

XrtGearNodeTraverseTo()

XmNxrtGearNodeCurrent をセットし、指定されたノードにフォーカスを移動します。指定されたノードが表示されていなければ、そのノードが表示されるようスクロールされます。もし、そのノードが開かれていなければ、表示されている親までスクロールされます。XrtGearNodeTraverseTo()は、フォーカスを移動するだけでノードを選択しません。

```
void
XrtGearNodeTraverseTo(
  Widget  node
)
```

node は、フォーカスを得るノードです。

XrtGearNodeWriteTreeToBuffer()

ツリーまたはサブツリーを複数行のテキスト・ストリングとしてバッファに書きます。このテキスト・ストリングを持つ内部バッファが返されます。(この内部メモリはいつでも上書き可能です。テキスト・ストリングのコピーを保存するには、バッファのポインタを下記のように提供してください。)

```
String
XrtGearNodeWriteTreeToBuffer(
  Widget  parent,
  char    *buffer,
  Boolean full_path,
  Boolean write_root,
  String  indent,
  String  separator,
  String  open_escape,
  String  close_escape,
  int     *length
)
```

parent は、書かれるツリーの親です。*buffer* は NULL でなければ、ツリーが書かれるテキスト・ストリングで、各行が各ノードを表します。*buffer* が NULL であれば、ツリーは内部メモリに書かれるだけですので、直ちにこれを使用するか、またはコピーしてください。

full_path は、各ノードにツリー階層全体が与えられているか(*full_path* が `True`)、またはインデント・レベルとノード・ラベルだけが提供されているか(*full_path* が `False`)を示します。*full_path* が `True` であれば、*indent* で指定されているストリングがツリーの階層をコンポーネントに分けます。

write_root は、親ノードがバッファに書かれるかどうかを示します。*write_root* が Outliner ウィジェットであれば、*XtName()*で返されるウィジェット名がバッファに書かれます。*indent* は、(ある行の)ノードのインデント・レベルを指定するストリングです。*indent* が `NULL` の場合は、"`\t`"(tab 文字)が使用されます。*separator* は、一つのノードの複数ラベルを区切るストリングです。デフォルトは","(カンマ)です。

open_escape は、(ノード・テキストではない)指示の部分の開始を示す文字です。*open_escape* が `NULL` の場合は、"`(`"が使用されます。*closed_escape* は、(ノード・テキストではない)指示の部分の終了を示す文字です。*closed_escape* が `NULL` の場合は、"`)`"が使用されます。*length* は `NULL` でなければ、テキスト・ストリングの長さを byte で返します。

XrtGearNodeWriteTreeToStream()

ツリーをデータ・ファイルに書きます。ファイルが書かれると `True` を返し、エラーが起きると `False` を返します。

```
Boolean
XrtGearNodeWriteTreeToStream(
  Widget  parent,
  FILE    *fp,
  Boolean full_path,
  Boolean write_root,
  String  indent,
  String  separator,
  String  open_escape,
  String  close_escape,
  int     *length
)
```

parent は、書かれるツリーの親です。*fp* は、ツリーが書かれるデータ・ファイルです。(ファイルの各行がツリーの各ノードを表します。)

full_path は、各ノードにツリー階層全体が与えられているか(*full_path* が `True`)、またはインデント・レベルとノード・ラベルだけが提供されているか(*full_path* が `False`)を示します。*full_path* が `True` であれば、*indent* で指定されているストリングがツリーの階層をコンポーネントに分けます。

write_root は、親ノードがデータ・ファイルに書かれるかどうかを示します。*write_root* が Outliner ウィジェットであれば、*XtName()*で返されるウィジェット名がファイルに書かれます。

indent は、(ある行の)ノードのインデント・レベルを指定するストリングです。*indent* が `NULL` の場合は、"`\t`"(tab 文字)が使用されます。

separator は、一つのノードの複数ラベルを区切るストリングです。デフォルトは","(カンマ)です。

open_escape は、(ノード・テキストではない)指示の部分の開始を示す文字です。*open_escape* が NULL の場合は、"("が使用されます。*closed_escape* は、(ノード・テキストではない)指示の部分の終了を示す文字です。*closed_escape* が NULL の場合は、")"が使用されます。

length は NULL でなければ、書かれたファイルの長さを byte で返します。

7.14.3 XrtNodeStyle の手続きと方法

XmCreateXrtNodeStyle()

NodeStyle を作成します。

```
Widget
XmCreateXrtNodeStyle(
  Widget  parent,
  String  name,
  Arg     *arglist,
  int     argcount
  )
```

parent は親。*name* は作成されるノード・スタイル名。*arglist* は *argcount* 引数のリストです。

XmIsXrtNodeStyle()

引数が Nodestyle であれば True を返します。

```
Boolean
XmIsXrtNodeStyle(
  Widget  widget
  )
```

XrtGearNodestyleFromName()

Outliner ウィジェットのノード・スタイルのリストから、指定された名前とマッチするスタイルを探します。

```
Widget
XrtGearNodeStyleFromName(
  Widget  outliner,
  String  name
  )
```

outiner は、検索されるノード・スタイルのリストを持つ Outliner ウィジェット。*name* は検索に使用されるスタイル名。リスト内でノード・スタイルがみつからなければ、手続きは NULL を返します。

7.14.4 XrtColun の手続きと方法

XmCreateXrtColumn()

列を作成します。

```
Widget
XmCreateXrtColumn(
  Widget  parent,
  String  name,
  Arg     *arglist,
  int     argcount
)
```

parent は親。*name* は作成される列の名前。*arglist* は *argcount* 引数のリストです。

XmIsXrtColumn()

引数が列であれば `True` を返します。

```
Boolean
XmIsXrtColumn(
  Widget  widget
  )
```

XrtGearColumnFromXEvent()

XEvent が起きた列を返します。イベントが列で起こっていない時は `NULL` を返します。

```
Widget
XrtGearColumnFromXEvent(
  Widget  outliner,
  XEvent  *event
  )
```

outliner は Outliner ウィジェット。*event* は起きたイベントです。

XrtGearColumnFromXY()

(ルート・ウィンドウに相対する)XY 位置にある列を返します。指定された位置が列でない場合は、`NULL` を返します。

```
Widget
XrtGearColumnFromXY(
  Widget  outliner,
  int     root_x,
  int     root_y,
  )
```

outliner は Outliner ウィジェットです。*root_x* は X 位置、*root_y* は Y 位置です。

7.15 トランスレーションとアクション

7.15.1 Node 領域の Clip Child のトランスレーション

clip child のトランスレーションは、XmNxrtGearSelectionPolicy の値に依存します。次の表は、このリソースの値に対するトランスレーションの一覧です。デフォルトのトランスレーションも各表の最後にあります。

Single Auto のトランスレーション

選択がシングル自動モード(XmNxrtGearSelectionPolicy が XRTGEAR_SELECTION_POLICY_SINGLE_AUTO_SELECT にセットされている)時に使用されるトランスレーションです。

Event	Action / Actions
~Alt ~Ctrl ~Meta ~Shift <Key>space	NoAction()
~Alt ~Ctrl ~Meta ~Shift <Btn1Down>	ResizeColumn(Start) Edit(Pointer) GoTo(Pointer) ChangeState(ShortCut) Select(Start) ChangeState(Node,2) Activate(Node,2)
Shift ~Alt ~Ctrl ~Meta <Btn1Down>	GoTo(Pointer) Select(Start)
Ctrl ~Alt ~Meta ~Shift <Btn1Down>	GoTo(Pointer) Select(Start)
~Alt ~Ctrl ~Meta <Btn1Motion>	ResizeColumn(Extend)
~Alt ~Ctrl ~Meta ~Shift <Btn1Up>	ResizeColumn(End)
<Key>osfActivate	ChangeState(Current) Activate(Current)
<Key>Return	ChangeState(Current) Activate(Current)
<Key>osfCancel	ResizeColumn(Cancel) CancelEdit()
Shift ~Alt ~Ctrl ~Meta <Btn2Down>	Drag(Selection, Move)
Ctrl ~Shift ~Alt ~Meta <Btn2Down>	Drag(Selection, Copy)
~Shift ~Alt ~Ctrl ~Meta <Btn2Down>	Drag(Selection)
~Alt ~Ctrl ~Meta ~Shift <Key>osfSelect	ChangeState(Current) Activate(Current)
~Alt ~Ctrl ~Meta ~Shift <Key>osfPageDown	GoTo(NextPage) Select(Current)

Event	Action / Actions
~Alt ~Ctrl ~Meta ~Shift <Key>osfPageUp	GoTo(PreviousPage) Select(Current)
~Alt ~Ctrl ~Meta ~Shift <Key>space	NoAction()
~Alt ~Ctrl ~Meta ~Shift <Key>osfDown	GoTo(Next) Select(Current)
~Alt ~Ctrl ~Meta ~Shift <Key>osfUp	GoTo(Previous) Select(Current)
~Alt ~Ctrl ~Meta ~Shift <Key>osfBeginLine	GoTo(Top) Select(Current)
~Alt ~Ctrl ~Meta ~Shift <Key>osfEndLine	GoTo(Bottom) Select(Current)
~Alt ~Ctrl ~Meta Shift <Key>osfPageDown	GoTo(NextPage) Select(Current)
~Alt ~Ctrl ~Meta Shift <Key>osfPageUp	GoTo(PreviousPage) Select(Current)
~Alt ~Ctrl ~Meta Shift <Key>osfDown	GoTo(Next) Select(Current)
~Alt ~Ctrl ~Meta Shift <Key>osfUp	GoTo(Previous) Select(Current)
~Alt ~Ctrl ~Meta Shift <Key>osfBeginLine	GoTo(Top) Select(Current)
~Alt ~Ctrl ~Meta Shift <Key>osfEndLine	GoTo(Bottom) Select(Current)

Single のトランスレーション

選択がシングル・モード(`XmNxrtGearSelectionPolicy` が `XRTGEAR_SELECTION_POLICY_SINGLE` にセットされている)時に使用されるトランスレーションです。

Event	Action / Actions
~Alt ~Ctrl ~Meta ~Shift <Key>space	Select(Start)
~Alt ~Ctrl ~Meta ~Shift <Btn1Down>	ResizeColumn(Start) GoTo(Pointer) Edit(Pointer) ChangeState(ShortCut) Select(Start) ChangeState(Node,2) Activate(Node,2)
Shift ~Alt ~Ctrl ~Meta <Btn1Down>	GoTo(Pointer) Select(Start)
Ctrl ~Alt ~Meta ~Shift <Btn1Down>	GoTo(Pointer) Select(Start)
~Alt ~Ctrl ~Meta <Btn1Motion>	ResizeColumn(Extend)

Event	Action / Actions
~Alt ~Ctrl ~Meta ~Shift <Btn1Up>	ResizeColumn(End)
Shift ~Alt ~Ctrl ~Meta <Btn2Down>	Drag(Selection, Move)
Ctrl ~Shift ~Alt ~Meta <Btn2Down>	Drag(Selection, Copy)
~Shift ~Alt ~Ctrl ~Meta <Btn2Down>	Drag(Selection)
~Alt ~Ctrl ~Meta ~Shift <Key>osfSelect	ChangeState(Current) Activate(Current)
<Key>osfActivate	ChangeState(Current) Activate(Current)
<Key>Return	ChangeState(Current) Activate(Current)
<Key>osfCancel	Select(Cancel) ResizeColumn(Cancel) CancelEdit()
~Alt ~Ctrl ~Meta ~Shift <Key>osfPageDown	GoTo(NextPage)
~Alt ~Ctrl ~Meta ~Shift <Key>osfPageUp	GoTo(PreviousPage)
~Alt ~Ctrl ~Meta ~Shift <Key>osfDown	GoTo(Next)
~Alt ~Ctrl ~Meta ~Shift <Key>osfUp	GoTo(Previous)
~Alt ~Ctrl ~Meta ~Shift <Key>osfBeginLine	GoTo(Top)
~Alt ~Ctrl ~Meta ~Shift <Key>osfEndLine	GoTo(Bottom)
~Alt ~Ctrl ~Meta Shift <Key>osfPageDown	GoTo(NextPage)
~Alt ~Ctrl ~Meta Shift <Key>osfPageUp	GoTo(PreviousPage)
~Alt ~Ctrl ~Meta Shift <Key>osfDown	GoTo(Next)
~Alt ~Ctrl ~Meta Shift <Key>osfUp	GoTo(Previous)
~Alt ~Ctrl ~Meta Shift <Key>osfBeginLine	GoTo(Top)
~Alt ~Ctrl ~Meta Shift <Key>osfEndLine	GoTo(Bottom)

Range のトランスレーション

選択が範囲モード(XmNxrtGearSelectionPolicy が XRTGEAR_SELECTION_POLICY_RANGE にセットされている)時に使用されるトランスレーションです。

Event	Action / Actions
~Alt ~Ctrl ~Meta ~Shift <Key>space	Select(Start)
~Alt ~Ctrl ~Meta ~Shift <Btn1Down>	ResizeColumn(Start) GoTo(Pointer) Edit(Pointer) ChangeState(ShortCut) Select(Start) ChangeState(Node,2) Activate(Node,2)
Shift ~Alt ~Ctrl ~Meta <Btn1Down>	GoTo(Pointer) Select(Extend)
Ctrl ~Alt ~Meta ~Shift <Btn1Down>	NoAction()
~Alt ~Ctrl ~Meta <Btn1Motion>	GoTo(Pointer) ResizeColumn(Extend) Select(Extend)
~Alt ~Ctrl ~Meta ~Shift <Btn1Up>	ResizeColumn(End)
Shift ~Alt ~Ctrl ~Meta <Btn2Down>	Drag(Selection, Move)
Ctrl ~Shift ~Alt ~Meta <Btn2Down>	Drag(Selection, Copy)
~Shift ~Alt ~Ctrl ~Meta <Btn2Down>	Drag(Selection)
~Alt ~Ctrl ~Meta ~Shift <Key>osfSelect	ChangeState(Current) Activate(Current)
<Key>osfActivate	ChangeState(Current) Activate(Current)
<Key>Return	ChangeState(Current) Activate(Current)
<Key>osfCancel	Select(Cancel) ResizeColumn(Cancel) CancelEdit()
~Alt ~Ctrl ~Meta ~Shift <Key>osfPageDown	GoTo(NextPage)
~Alt ~Ctrl ~Meta ~Shift <Key>osfPageUp	GoTo(PreviousPage)
~Alt ~Ctrl ~Meta ~Shift <Key>osfDown	GoTo(Next)
~Alt ~Ctrl ~Meta ~Shift <Key>osfUp	GoTo(Previous)
~Alt ~Ctrl ~Meta ~Shift <Key>osfBeginLine	GoTo(Top)
~Alt ~Ctrl ~Meta ~Shift <Key>osfEndLine	GoTo(Bottom)

Event	Action / Actions
~Alt ~Ctrl ~Meta Shift <Key>osfPageDown	GoTo(NextPage) Select(CurrentExtend)
~Alt ~Ctrl ~Meta Shift <Key>osfPageUp	GoTo(PreviousPage) Select(CurrentExtend)
~Alt ~Ctrl ~Meta Shift <Key>osfDown	GoTo(Next) Select(CurrentExtend)
~Alt ~Ctrl ~Meta Shift <Key>osfUp	GoTo(Previous) Select(CurrentExtend)
~Alt ~Ctrl ~Meta Shift <Key>osfBeginLine	GoTo(Top) Select(CurrentExtend)
~Alt ~Ctrl ~Meta Shift <Key>osfEndLine	GoTo(Bottom) Select(CurrentExtend)

Multiple のトランスレーション

選択がマルチ・モード(`XmNxrtGearSelectionPolicy` が `XRTGEAR_SELECTION_POLICY_MULTIPLE` にセットされている)時に使用されるトランスレーションです。

Event	Action / Actions
~Alt ~Ctrl ~Meta ~Shift <Key>space	Select(Add)
~Alt ~Ctrl ~Meta ~Shift <Btn1Down>	ResizeColumn(Start) GoTo(Pointer) Edit(Pointer) ChangeState(ShortCut) Select(Add) ChangeState(Node,2) Activate(Node,2)
Shift ~Alt ~Ctrl ~Meta <Btn1Down>	GoTo(Pointer) Select(Extend)
Ctrl ~Alt ~Meta ~Shift <Btn1Down>	GoTo(Pointer) Select(Add)
~Alt ~Ctrl ~Meta <Btn1Motion>	GoTo(Pointer) ResizeColumn(Extend) Select(Extend)
~Alt ~Ctrl ~Meta ~Shift <Btn1Up>	ResizeColumn(End)
Shift ~Alt ~Ctrl ~Meta <Btn2Down>	Drag(Selection, Move)
Ctrl ~Shift ~Alt ~Meta <Btn2Down>	Drag(Selection, Copy)
~Shift ~Alt ~Ctrl ~Meta <Btn2Down>	Drag(Selection)
~Alt ~Ctrl ~Meta ~Shift <Key>osfSelect	ChangeState(Current) Activate(Current)

Event	Action / Actions
<Key>osfActivate	ChangeState(Current) Activate(Current)
<Key>Return	ChangeState(Current) Activate(Current)
<Key>osfCancel	Select(Cancel) ResizeColumn(Cancel) CancelEdit()
~Alt ~Ctrl ~Meta ~Shift <Key>osfPageDown	GoTo(NextPage)
~Alt ~Ctrl ~Meta ~Shift <Key>osfPageUp	GoTo(PreviousPage)
~Alt ~Ctrl ~Meta ~Shift <Key>osfDown	GoTo(Next)
~Alt ~Ctrl ~Meta ~Shift <Key>osfUp	GoTo(Previous)
~Alt ~Ctrl ~Meta ~Shift <Key>osfBeginLine	GoTo(Top)
~Alt ~Ctrl ~Meta ~Shift <Key>osfEndLine	GoTo(Bottom)
~Alt ~Ctrl ~Meta Shift <Key>osfPageDown	GoTo(NextPage) Select(CurrentExtend)
~Alt ~Ctrl ~Meta Shift <Key>osfPageUp	GoTo(PreviousPage) Select(CurrentExtend)
~Alt ~Ctrl ~Meta Shift <Key>osfDown	GoTo(Next) Select(CurrentExtend)
~Alt ~Ctrl ~Meta Shift <Key>osfUp	GoTo(Previous) Select(CurrentExtend)
~Alt ~Ctrl ~Meta Shift <Key>osfBeginLine	GoTo(Top) Select(CurrentExtend)
~Alt ~Ctrl ~Meta Shift <Key>osfEndLine	GoTo(Bottom) Select(CurrentExtend)

Multiple Auto のトランスレーション

選択がマルチ自動モード(`XmNxrtGearSelectionPolicy` が `XRTGEAR_SELECTION_POLICY_MULTIPLE_AUTO_UNSELECT` にセットされている)時に使用されるトランスレーションです。

Event	Action / Actions
~Alt ~Ctrl ~Meta ~Shift <Key>space	Select(Start)

Event	Action / Actions
~Alt ~Ctrl ~Meta ~Shift <Btn1Down>	ResizeColumn(Start) GoTo(Pointer) Edit(Pointer) ChangeState(ShortCut) Select(Start) ChangeState(Node,2) Activate(Node,2)
Shift ~Alt ~Ctrl ~Meta <Btn1Down>	GoTo(Pointer) Select(Extend)
Ctrl ~Alt ~Meta ~Shift <Btn1Down>	GoTo(Pointer) Select(Add)
~Alt ~Ctrl ~Meta <Btn1Motion>	GoTo(Pointer) ResizeColumn(Extend) Select(Extend)
~Alt ~Ctrl ~Meta ~Shift <Btn1Up>	ResizeColumn(End)
Shift ~Alt ~Ctrl ~Meta <Btn2Down>	Drag(Selection, Move)
Ctrl ~Shift ~Alt ~Meta <Btn2Down>	Drag(Selection, Copy)
~Shift ~Alt ~Ctrl ~Meta <Btn2Down>	Drag(Selection)
~Alt ~Ctrl ~Meta ~Shift <Key>osfSelect	ChangeState(Current) Activate(Current)
<Key>osfActivate	ChangeState(Current) Activate(Current)
<Key>Return	ChangeState(Current) Activate(Current)
<Key>osfCancel	Select(Cancel) ResizeColumn(Cancel) CancelEdit()
~Alt ~Ctrl ~Meta ~Shif <Key>osfPageDown	GoTo(NextPage)
~Alt ~Ctrl ~Meta ~Shift <Key>osfPageUp	GoTo(PreviousPage)
~Alt ~Ctrl ~Meta ~Shift <Key>osfDown	GoTo(Next)
~Alt ~Ctrl ~Meta ~Shift <Key>osfUp	GoTo(Previous)
~Alt ~Ctrl ~Meta ~Shif t<Key>osfBeginLine	GoTo(Top)
~Alt ~Ctrl ~Meta ~Shift <Key>osfEndLine	GoTo(Bottom)
~Alt ~Ctrl ~Meta Shift <Key>osfPageDown	GoTo(NextPage) Select(CurrentExtend)
~Alt ~Ctrl ~Meta Shift <Key>osfPageUp	GoTo(PreviousPage) Select(CurrentExtend)

Event	Action / Actions
~Alt ~Ctrl ~Meta Shift <Key>osfDown	GoTo(Next) Select(CurrentExtend)
~Alt ~Ctrl ~Meta Shift <Key>osfUp	GoTo(Previous) Select(CurrentExtend)
~Alt ~Ctrl ~Meta Shift <Key>osfBeginLine	GoTo(Top) Select(CurrentExtend)
~Alt ~Ctrl ~Meta Shift <Key>osfEndLine	GoTo(Bottom) Select(CurrentExtend)

Disallowed のトランスレーション

選択不許可モード(`XmNxrtGearSelectionPolicy` が `XRTGEAR_SELECTION_POLICY_NONE` にセットされている)時に使用されるトランスレーションです。

Event	Action / Actions
~Alt ~Ctrl ~Meta ~Shift <Key>space	NoAction()
~Alt ~Ctrl ~Meta ~Shift <Btn1Down>	ResizeColumn(Start) GoTo(Pointer) ChangeState(ShortCut) ChangeState(Node,2) Activate(Node,2)
Shift ~Alt ~Ctrl ~Meta <Btn1Down>	GoTo(Pointer)
Ctrl ~Alt ~Meta ~Shift <Btn1Down>	GoTo(Pointer)
~Alt ~Ctrl ~Meta <Btn1Motion>	ResizeColumn(Extend)
~Alt ~Ctrl ~Meta ~Shift <Btn1Up>	ResizeColumn(End)
Shift ~Alt ~Ctrl ~Meta <Btn2Down>	Drag(Node, Move)
Ctrl ~Shift ~Alt ~Meta <Btn2Down>	Drag(Node, Copy)
~Shift ~Alt ~Ctrl ~Meta <Btn2Down>	Drag(Node)
~Alt ~Ctrl ~Meta ~Shift <Key>osfSelect	ChangeState(Current) Activate(Current)
<Key>osfActivate	ChangeState(Current) Activate(Current)
<Key>Return	ChangeState(Current) Activate(Current)
<Key>osfCancel	ResizeColumn(Cancel) CancelEdit()

Event	Action / Actions
~Alt ~Ctrl ~Meta ~Shift <Key>osfPageDown	GoTo(NextPage)
~Alt ~Ctrl ~Meta ~Shift <Key>osfPageUp	GoTo(PreviousPage)
~Alt ~Ctrl ~Meta ~Shift <Key>osfDown	GoTo(Next)
~Alt ~Ctrl ~Meta ~Shift <Key>osfUp	GoTo(Previous)
~Alt ~Ctrl ~Meta ~Shift <Key>osfBeginLine	GoTo(Top)
~Alt ~Ctrl ~Meta ~Shift <Key>osfEndLine	GoTo(Bottom)
~Alt ~Ctrl ~Meta Shift <Key>osfPageDown	GoTo(NextPage)
~Alt ~Ctrl ~Meta Shift <Key>osfPageUp	GoTo(PreviousPage)
~Alt ~Ctrl ~Meta Shift <Key>osfDown	GoTo(Next)
~Alt ~Ctrl ~Meta Shift <Key>osfUp	GoTo(Previous)
~Alt ~Ctrl ~Meta Shift <Key>osfBeginLine	GoTo(Top)
~Alt ~Ctrl ~Meta Shift <Key>osfEndLine	GoTo(Bottom)

デフォルトのトランスレーション

上記の表にあるトランスレーションによって上書きされるトランスレーションです。

Event	Action / Actions
~Alt ~Ctrl ~Meta ~Shift <Key>space	ChangeState(Current)Activate(Current)
~Alt ~Ctrl ~Meta ~Shift <Btn1Down>	NoAction()
Shift ~Alt ~Ctrl ~Meta <Btn1Down>	NoAction()
Ctrl ~Alt ~Meta ~Shift <Btn1Down>	NoAction()
~Alt ~Ctrl ~Meta <Btn1Motion>	NoAction()

Event	Action / Actions
~Alt ~Ctrl ~Meta ~Shift <Btn1Up>	NoAction()
~Alt ~Ctrl ~Meta ~Shift <Key>osfSelect	ChangeState(Current) Activate(Current)
<Key>osfActivate	ChangeState(Current) Activate(Current)
<Key>Return	ChangeState(Current) Activate(Current)
<Key>osfCancel	Select(Cancel) ResizeColumn(Cancel)
~Alt ~Ctrl ~Meta ~Shift <Key>osfPageDown	GoTo(NextPage)
~Alt ~Ctrl ~Meta ~Shift <Key>osfPageUp	GoTo(PreviousPage)
~Alt ~Ctrl ~Meta ~Shift <Key>osfDown	GoTo(Next)
~Alt ~Ctrl ~Meta ~Shift <Key>osfUp	GoTo(Previous)
~Alt ~Ctrl ~Meta ~Shift <Key>osfBeginLine	GoTo(Top)
~Alt ~Ctrl ~Meta ~Shif t<Key>osfEndLine	GoTo(Bottom)
~Alt ~Ctrl ~Meta Shift <Key>osfPageDown	GoTo(NextPage)
~Alt ~Ctrl ~Meta Shift <Key>osfPageUp	GoTo(PreviousPage)
~Alt ~Ctrl ~Meta Shift <Key>osfDown	GoTo(Next)
~Alt ~Ctrl ~Meta Shift <Key>osfUp	GoTo(Previous)
~Alt ~Ctrl ~Meta Shift <Key>osfBeginLine	GoTo(Top)
~Alt ~Ctrl ~Meta Shift <Key>osfEndLine	GoTo(Bottom)
Shift ~Ctrl ~Alt ~Meta <Btn2Down>	Drag(Move)
Ctrl ~Shift ~Alt ~Meta <Btn2Down>	Drag(Copy)
~Ctrl ~Alt ~Shift ~Meta <Btn2Down>	Drag(Move)
<Key>osfDelete	Delete()
<Key>osfHelp	PrimitiveHelp()
Shift ~Meta ~Alt <Key>Tab	PrimitivePrevTabGroup()

Event	Action / Actions
~Meta ~Alt <Key>Tab	PrimitiveNextTabGroup
<FocusIn>	PrimitiveFocusIn()
<FocusOut>	PrimitiveFocusOut()
<Unmap>	PrimitiveUnmap()

7.15.2 列ラベルの Clip Child のトランスレーション

Event	Action / Actions
~Alt ~Ctrl ~Meta ~Shift <Btn1Down>	ResizeColumn(Start) Select(Column)
~Alt ~Ctrl ~Meta ~Shift <Btn1Down>(2+)	Activate(Column)
~Alt ~Ctrl ~Meta ~Shift <Btn1Up>	ResizeColumn(End)
~Alt ~Meta <Btn2Down>	Drag(Column)

7.15.3 編集に使用される XmText ウィジェットのトランスレーション

Event	Action / Actions
<Key>osfCancel	CancelEdit()
<Key>osfActivate	CommitEdit(True)
<Key>Return	CommitEdit(True)
~Alt ~Ctrl ~Meta Shift <Key>Tab	Edit(Previous)
~Alt ~Ctrl ~Meta ~Shift <Key>Tab	Edit(Next)
~Alt ~Meta <Key>osfPageDown	GoTo(NextPage) Edit(Current) Select(Start)
~Alt ~Meta <Key>osfPageUp	GoTo(PreviousPage) Edit(Current) Select(Start)
~Alt ~Meta <Key>osfDown	GoTo(Next) Edit(Current) Select(Start)
~Alt ~Meta <Key>osfUp	GoTo(Previous) Edit(Current) Select(Start)
~Alt ~Ctrl ~Meta Shift <Key>osfPageDown	GoTo(NextPage) Edit(Current) Select(Start)
~Alt ~Ctrl ~Meta Shift <Key>osfPageUp	GoTo(PreviousPage) Edit(Current) Select(Start)
~Alt Ctrl ~Meta <Key>osfLeft	Edit(Previous)
~Alt Ctrl ~Meta <Key>osfRight	Edit(Next)

7.15.4 アクション

Activate()

activate コールバックを呼び出します。(アプリケーションに使用されます。) 二つのパラメータを取ります。最初のパラメータは次の値のどれかです。

Node	ノード上(フォルダ・アイコンも含む)でクリックされた時に、activate コールバックを呼ぶ
ShortCut	ショートカット・ボタンがクリックされた時に、activate コールバックを呼ぶ
Current	カレント・ノードで、activate コールバックを呼ぶ。キーボードのトランスレーションに使用される
Column	列ラベルがクリックされた時に、activate コールバックを呼ぶ

デフォルトの動作は、ノード上のダブル・クリック、デフォルトの activate キーは `space` と `osfActivate` です。

二番目のパラメータはオプションで、activate コールバックを呼び出すのに必要なクリック数を指定します。デフォルトは 1(シングル・クリック)です。

CancelEdit()

このアクションは *XrtGearOutlinerCancelEdit()*を呼び出します。キーは `osfCancel` です。

ChangeState()

これを利用して、状態が変化する条件を指定できます。二つのパラメータをとります。最初のパラメータは次の値のどれかです。

Node	ノード上(フォルダ・アイコンも含む)でクリックされた時に、状態が変化する
ShortCut	ショートカット・ボタンがクリックされた時に、状態が変化する
Current	カレント・ノードで状態が変化する。キーボードのトランスレーションに使用される

二番目のパラメータはオプションで、状態を変化させるのに必要なクリック数を指定します。デフォルトは 1(シングル・クリック)です。

デフォルトの動作は次の通りです。

- ノードがクリックされる場合は、ダブル・クリックで状態を変化させる
- ショートカット・ボタンがクリックされる場合は、シングル・クリックで状態を変化させる

デフォルトの、状態を変更させるキーは `space` と `osfActivate` です。

CommitEdit()

ユーザが行う確定では検証コールバックに渡されるデータが異なるので、*XrtGearOutlinerCommitEdit()*ではなく、このアクションを使用します。XmText をアンマップするかどうかを指定する `True` か `False` の一つのパラメータを取ります。キーは、`osfActivate` と `Return` です。

Delete()

カレント・ノードを消去します。これはノードを削除するために *XtDestroyWidget()*を呼び出します。あるノードが削除された時にその通知を受け取るようにするには、`XmNdestroyCallback` を使用して、ノード作成時に destroy コールバック・ルーチンを追加して下さい。Outliner ウィジェット自身によって追加されたノードを登録するには、`XmNxrtGearNodeCreateCallback` で指定されるリストに destroy コールバックを追加してください。デフォルトでは、*Delete()*アクションは `osfDelete` です。

Drag()

これを利用して、エンドユーザはノードを他の場所にインタラクティブにドラッグすることができます。このアクションは、Mouse イベントと一緒に使用してください。このアクションには二つのパラメータが渡されます。最初のパラメータは、ドラッグされるアイテムを指定し、次のうちのどれかです。

Node	カレント・ノードをドラッグする
Selection	現在選択されているアイテムをドラッグする
Column	現在選択されている列をドラッグする

二番目の引数はアクションを定義し、次の内のどれかです

Move	指定されたアイテムを新しい位置へ移動する
Copy	アイテムのコピーを新しい位置へ挿入する

二番目の引数が指定されていない場合は、Move として処理されます。最初の引数が Column の時は、二番目の引数は Copy にすることはできません。

ノードがコピーされる時は、新しいノードが作成され、`XmNxrtGearNodeCreateCallback` で指定されるコールバックが呼び出されます。

Edit()

指定された場所にエディタ・ウィジェットをマップします。パラメータは次のどれかです。

Pointer	マウス・ポインタが置かれているノードを編集する
Current	カレント・ノードを編集する
Next	カレント・ノードの次の兄弟(sibling)を編集する
Previous	カレント・ノードの前の兄弟(sibling)を編集する

GoTo()

キーやマウスのアクションにより、指定した位置にフォーカスを移動します。このアクションは、次の内の一つのパラメータを取ります。

Pointer	フォーカスをマウス・ポインタがあるノードへ移動する
Current	カレント・ノードが(非表示状態であれば)表示する
TopOfPage	フォーカスをカレント・ページの一番上に移動する
BottomOfPage	フォーカスをカレント・ページの一番下に移動する
NextPage	次のページにフォーカスを移動し表示する
PreviousPage	前のページにフォーカスを移動し表示する

ParentNode	フォーカスを親へ移動する
NextParent	フォーカスをノードの親の次の兄弟(sibling)に移動する
Next	フォーカスをノードの次の兄弟(sibling)に移動する
Previous	フォーカスをノードの前の兄弟(sibling)に移動する
TopOfTree	フォーカスをルート・ノードの最初の子に移動する
BottomOfTree	フォーカスをルート・ノードの最後に表示されている子に移動する

NoAction()

何もせず、コールバックも呼び出されません。デフォルトでは、このアクション・ルーチンはどのイベントにも割り当てられていません。このルーチンを呼ぶようにイベントをセットすることにより、トランスレーションを効率的に不可にすることができます。

ResizeColumn()

列の枠、列ラベルの枠上にカーソルが来ると、このアクション・ルーチンは、列のリサイズを開始または終了します。次の内の一つのパラメータが渡されます。

Start	列のリサイズを開始。マウス・カーソルに従って、ウィジェットは点線を描画し、`End` パラメータで *ResizeColumns()*が呼び出されると点線の描画を終了する
Extend	一度描画した点線をマウス・カーソルに従って延長する
End	`XmNxrtGearResizeCallback` で指定されるコールバックを呼び出し、列をリサイズする

`XmNxrtGearShadowThickness` が 0 の場合や、`XmNwidth` が 0 にセットされている時(列が非表示)は、列はインタラクティブにリサイズできません。

Select()

ノードの選択のオンかオフ(トグル)を実行します。次の内の一つのパラメータを取ります。

Start	フォーカスがあるノードが選択される。他のノードは全て選択解除される
Extend	最後に選択されているノードからカレント・ノードまでを選択する
CurrentExtend	Extend と同様。トラバース・キーに使用される
Add	前の選択を残したまま、新しい選択を開始する
Column	現在の列を選択する。`XmNxrtGearColumnSelected` リソースをセットする
Cancel	選択されているノードを全て選択解除する。(但し、`XmNxrtGearSelectionPolicy` が `XRTGEAR_SELECTION_POLICY_SINGLE_AUTO_SELECT` にセットされていない場合)

7.16 Lightweight フォルダとアイテム・ノード

Outliner のフォルダとアイテム・ノードは、それぞれ XtObject です。そのため、何千ものフォルダやアイテム・ノードを使用した Outliner アプリケーションの場合、非常に多くの XtObject を扱うことになるので、パフォーマンスがかなり落ちてしまいます。このような場合に、Outliner の'lightweight'(軽量な)フォルダとアイテム・ノードを使用することができます。lightweight ノードは XtObject ではありませんが、XtObject のノードとも互換性があり、必要であれば、ひとつのアプリケーションで通常のノードと混合して使用することができます。

Lightweight ノードを使用したツリーの作成

下記のコードは、lightweight フォルダとアイテムを使用してツリーを作成しています。

```
#include <stdio.h>
#include <stdlib.h>
#include <Xm/XrtGear.h>
#include <Xm/XrtNode.h>
#include <Xm/XrtLWNode.h>
#include <Xm/XrtLWFolder.h>
#include <Xm/XrtNodeFolder.h>
#include <Xm/XrtList.h>
#include <Xm/XrtGearString.h>
#include <Xm/XrtColumn.h>
#include <Xm/XrtOutliner.h>
#include <X11/Intrinsic.h>

int
main(int argc, char **argv)
{
 Widget toplevel, outliner;
 XtAppContext app_context;
 Widget node, folder;

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

 outliner = XtVaCreateManagedWidget("outliner",
     xmXrtOutlinerWidgetClass, toplevel, NULL);

 folder = XrtGearFolderCreateLW(outliner, "Folder"
    XrtGearStringCreateCharString("Folder"),
    XRTGEAR_FOLDERSTATE_OPEN_ALL);

 node = XrtGearNodeCreateLW(folder, "Node ・
     XrtGearStringCreateCharString("item1"));

 node = XrtGearNodeCreateLW(folder, "Node ・
     XrtGearStringCreateCharString("item2"));

 XtRealizeWidget(toplevel);
 XtAppMainLoop(app_context);
}
```

7.16.1 ウィジェットとして cast される Lightweight Node

lightweight ノードは XtObject ではありませんが、既存のコードとの互換性を保つために、lightweight ノードを作成する方法 (*XrtGearNodeCreateLW()* や *XrtGearFolderCreateLW()*)は、cast される lightweight ノードをウィジェットとして返します。

`XtIsObject()`または `XtIsWidget()`に lightweight ノードが渡されれば、`False` を戻します。

`XtIsObject()`または `XtIsWidget()`は、lightweight ノードを渡すことができる唯一の Xt マクロ/手続きです。lightweight ノードには、必ず、XRT/gear が提供している手続きを使用してください。(例えば、*XtParent()*ではなく、*XrtGearNodeGetParent()*を使用する。)但し、XtObject ノードは、XRT/gear ノード関数に渡すことができます。(*XrtGearNodeGetParent()*に XtObject ノードが渡されれば、そのノードの親を取得するために、*XtParent()*が呼び出されます。)

7.16.2 Lightweight ノードのリソースの取得と設定

lightweight ノードは XtObject ではないので、*XtVaGetValues()*や *XtVaSetValues()*でプロパティを操作することができません。lightweight ノードの作成/破棄や、リソースの設定/取得を始め、その他の関数を実行するには、*XrtLWFolder.h* と *XrtLWNode.h* をインクルードする必要があります。

次のコードでは、lightweight アイテム・ノードの `Style` リソースを設定しています。

```
Widget outliner, nodestyle;
Widget node, folder;

nodestyle = XmCreateXrtNodeStyle(outliner,
  "mystyle", NULL, NULL);

/* XtVaSetValues() used to set XtObject resources */

  XtVaSetValues(nodestyle,
    XtVaTypedArg, XmNxrtGearBackgroundSelected,
    XmRString, "white", strlen("white")+1,
    NULL);
  XtVaSetValues(nodestyle,
    XtVaTypedArg, XmNxrtGearFontListSelected,
    XmRString, "-*-terminal-bold-r-*-*-14-*-*-*-*_*-iso8859-*",
    strlen("-*-terminal-bold-r-*-*-14-*-*-*-*_*-iso8859-*")+1,
    NULL);
  node = XrtGearNodeCreateLW(folder, "Node・
      XrtGearStringCreateCharString("item1"));

/* XrtGearNodeSetNodeStyle() used to set lightweight
node resources */

XrtGearNodeSetNodeStyle(node, nodestyle);
```

7.16.3 Lightweight ノードの破棄

lightweight アイテムとフォルダは、*XtDestroyWidget()*では破棄することができませんので、*XrtGearNodeDestroy()*を使用してください。

```
XrtGearNodeDestroy(node);
XrtGearNodeDestroy(folder);
```

7.16.4 Lightweight フォルダのデモ

fileviewer というデモでは、lightweight ノードが使用されています。既に構築されているデモでは、通常のノードを使用していますが、新しい *fileviewer* のデモのソースコードでは、通常のノードと lightweight ノードの両方を使用しています。エンベッドされている#ifdefs を使用して、通常のノードまたは lightweight ノードをコンパイルすることができます。

```
#ifdef XRTGEAR_LW
...
#else
...
#endif
```

fileviewer デモの lightweight バージョンを構築するには、make の段階で `XRTGEAR_LW` シンボルを定義してください。(この方法は、使用している機種に依存します。)

例： SunOS の場合

```
make all "CC= cc -DXRTGEAR_LW"
```

7.16.5 Lightweight アイテムとフォルダ･ノードのリソースの一覧

lightweight アイテム･ノードのリソースとフォルダ･ノードによって継承されるリソースの一覧です。フォルダ･ノード固有のリソースに関しては、Lightweight フォルダ･ノードのリソースの一覧(P218)を参照してください。

リソース名の後は、データ型、デフォルト値、そしてリソースと共に使用される手続きです。**C** は、そのリソースが提供されている関数(例: *XrtGearNodeCreateLW()*)で定義でき、**S** は設定の手続き(例: *XrtGearNodeSetLabel()*)が提供され、**G** は取得の関数(例: *XrtGearNodeGetLabel()*)が提供されていること表します。

Include File: <Xm/XrtLWNode.h>

Editable	**Boolean**	**True**	**SG**

このノードに属するノード･ラベルをユーザが編集できるかどうか指定します。

Label	**XrtGearObject**	**NULL**	**CSG**

表示されるラベルを指定します。

Selected	**Boolean**	**False**	**SG**

このノードが選択されているか(`True`)、選択されていないか(`False`)を指定します。

Style **Widget** ***default node style*** **SG**

デフォルトのノード･スタイルとして使用される NodeStyle オブジェクトを指定します。このリソースがセットされていない場合は、`XmNxrtGearNodeStyleList` で指定されるリストの最初のエレメントがデフォルトのスタイルとして使用されます。

UserData **XtPointer** **NULL** **SG**

ノードと関連付けられている、ユーザ(によって管理される)データのポインタを提供します。ここにあるデータの解放は、エンドユーザの役割です。

7.16.6 手続きと方法

lightweight アイテム･ノードの手続きと方法及び、フォルダ･ノードによって継承される手続きと方法のリストです。フォルダ･ノード固有の手続きと方法に関しては、218 ページを参照してください。これらの手続きには、XtObject ノードと lightweight ノードの両方を渡すことができます。

XrtGearIsNode()

与えられたウィジェットがアイテム･ノードまたはフォルダ･ノードであれば、`True` を返します。

```
Boolean
XrtGearIsNode(
 Widget          node
 )
```

XrtGearNodeCreateLW()

与えられた *parent* の子として lightweight ノードを作成し、*name* と *label* を割り当てます。新しいノードが取得され、互換性を保つために Widget として cast されます。

```
Widget
XrtGearNodeCreateLW(
 Widget          parent,
 String          name,
 XrtGearObject   label
 )
```

parent はフォルダ･ウィジェットです。フォルダ･ウィジェットは lightweight フォルダ･オブジェクトか Outliner フォルダ･オブジェクト、または Outliner ウィジェット自身とすることができます。

XrtGearNodeDestroy()

アイテムまたはフォルダ･ノードを破棄します。

```
void
XrtGearNodeDestroy(
 Widget          node,
 )
```

XrtGearNodeGetHeight()

指定された *node* の高さを pixel で返します。

```
Dimension
XrtGearNodeGetHeight(
 Widget          node
 )
```

XrtGearNodeGetEditable()

指定されたノードが編集可かどうかを返します。

```
Boolean
XrtGearNodeGetEditable(
  Widget       node
  );
```

node は XtObject または lightweight ノードです。

XrtGearNodeGetLabel()

指定された *node* の `Label` リソースの値を返します。

```
XrtGearObject
XrtGearNodeGetLabel(
  Widget           node
  )
```

XrtGearNodeGetName()

指定された *node* の `Name` リソースの値を返します。

```
String
XrtGearNodeGetName(
  Widget           node
  )
```

XrtGearNodeGetNodeStyle()

指定された *node* の `Style` リソースの値を返します。

```
Widget
XrtGearNodeGetNodeStyle(
  Widget           node
  )
```

XrtGearNodeGetParent()

node の親を返します。親は、lightweight ノードか XtObject ノード、または Outliner ウィジェットとすることができます。

```
Widget
XrtGearNodeGetParent(
  Widget           node
  )
```

XrtGearNodeGetSelected()

指定された *node* の `Selected` リソースの値を返します。ノードが選択されていれば `True`、されていなければ `False` を返します。

```
Boolean
XrtGearNodeGetSelected(
  Widget           node
  )
```

XrtGearNodeGetUserData()

与えられた *node* に関連付けられているユーザ(によって管理されている)データのポインタを返します。

```
XtPointer
XrtGearNodeGetUserData(
  Widget           node
  )
```

XrtGearNodeGetWidth()

指定された *node* の幅を pixel で返します。

```
Dimension
XrtGearNodeGetWidth(
  Widget           node
  )
```

XrtGearNodeSetEditable()

指定されたノードの編集状態を *editable* にセットします。

```
Boolean
XrtGearNodeSetSelected(
  Widget           node,
  Boolean          editable
  )
```

node は XtObject か Lightweight ノードです。

XrtGearNodeSetLabel()

node に表示される *label* をセットします。

```
Boolean
XrtGearNodeSetLabel(
  Widget           node,
  XrtGearObject    label
  )
```

XrtGearNodeSetNodeStyle()

ノード･スタイルをセットします。

```
Boolean
XrtGearNodeSetNodeStyle(
  Widget           node,
  Widget           style
  )
```

node はアイテム･ノード、*style* はノード･スタイル･オブジェクトです。

XrtGearNodeSetSelected()

node の `Selected` リソースを `True`(選択されている状態)または `False`(選択されていない状態)にセットします。属性が正しくセットされると `True` を返します。

```
Boolean
XrtGearNodeSetSelected(
  Widget           node,
  Boolean          selected
  )
```

node はアイテム･ノード、*selected* は、`Selected` リソースの設定です。

XrtGearNodeSetUserData()

node の UserData を、ユーザによって管理されているデータのポインタにセットします。このデータの解放はエンドユーザの役割です。属性が正しくセットされると True を返します。

```
Boolean
XrtGearNodeSetUserData(
 Widget          node,
 XtPointer       user_data
 )
```

node はアイテム・ノード、*user_data* はユーザ・データのポインタです。

7.16.7 Lightweight フォルダ・ノードのリソースの一覧

lightweight フォルダ・ノードのリソースの一覧です。その他の lightweight フォルダ・ノードのリソースは、P151 を参照してください。

リソース名の後は、データ型、デフォルト値、そしてリソースと共に使用される手続きです。**C** は、そのリソースが提供されている関数(例: *XrtGearFolderCreateLW()*)で定義でき、**S** は設定の手続き(例: *XrtGearNodeChangeFolderState()*)が提供され、**G** は取得の関数(例: *XrtGearNodeGetFolderState()*)が提供されていること表します。

Include File: <Xm/XrtLWFolder.h>

Children **XrtGearObject** **NULL** **G**

フォルダによって所有されているアイテムとフォルダ・ノードを含む XrtList オブジェクトのポインタを指定します。

State **XrtGearFolderState** **NULL** **CSG**

フォルダ・ノードの現在の状態を指定します。有効な値は次の通りです。

XRTGEAR_FOLDERSTATE_CLOSED	フォルダは閉、子は表示されない
XRTGEAR_FOLDERSTATE_OPEN_FOLDERS	フォルダは開、フォルダの子だけ表示
XRTGEAR_FOLDERSTATE_OPEN_ITEMS	フォルダは開、アイテムだけ表示
XRTGEAR_FOLDERSTATE_OPEN_ALL	フォルダは開、子は全て表示

この値は、プログラムまたはエンドユーザのインタラクションによってコントロールされます。Outliner の XmNxrtGearFolderStateMask リソースは、どの状態が有効であるかコントロールします。

7.16.8 手続きと方法

lightweight フォルダ・ノード固有の手続きと方法のリストです。その他の lightweight フォルダ・ノードの手続きに関しては、P178 を参照してください。

これらの手続きには、XtObject ノードと lightweight ノードの両方を渡すことができます。

XrtGearFolderCreateLW()

lightweight フォルダ・ノードを作成します。

```
Widget
XrtGearFolderCreateLW(
  Widget              parent,
  String              name,
  XrtGearObject       label,
  XrtGearFolderState  initial_state
  )
```

parent は outliner ウィジェット、*name* は作成されたウィジェットの名前です。*label* はフォルダの隣に表示されるラベル・テキストです。*initial_state* はフォルダの `State` リソースです(例: `XRTGEAR_FOLDERSTATE_CLOSED`)。

XrtGearIsNodeFolder()

node がフォルダであれば `True`、そうでなければ `False` を返します。

```
Boolean
XrtGearIsNodeFolder(
  Widget              node
  )
```

node はフォルダ・ノードです。

XrtGearNodeChangeFolderState()

フォルダの `state` リソースをセットします。この手続きは、一般の outliner に記述されている手続きと同じで、lightweight フォルダ・ノードにも使用できます。`XmNxrtGearFolderStateCallback` で指定されるコールバックは、状態(state)が変化すると呼び出されます。

```
void
XrtGearNodeChangeFolderState(
  Widget              folder,
  XrtGearFolderState  state
  )
```

folder は、状態が変更されるフォルダです。*state* は新しい `State` です。(このパラメータの値に関しては、`State` リソースの記述を参照してください。)

XrtGearNodeCreateLWTreeFromBuffer()

(データ・ファイル形式の)複数行のテキスト・ストリングから lightweight ツリーまたはサブ・ツリーを作成します。

```
Widget
XrtGearNodeCreateLWTreeFromBuffer(
  Widget              parent,
  char                *buffer,
  Boolean             full_path,
  XrtGearFolderState  initial_state,
  String              indent,
  String              separator,
  String              open_escape,
  String              close_escape
  )
```

parent は新しく作成されたツリーの親です。*buffer* は複数行のテキスト・ストリングで、各行は作成されるツリーのノードを表します。

*XrtGearNodeCreateLWTreeFromBuffer()*によって返される値は *parent* で、エラーが起こったときは `Null` が返ります。

full_path は、各ノードに対して完全なツリーの階層が提供されているか(*full_path* は `True`)、インデントのレベルから推測されるか(*full_path* は `False`)を示します。*full_path* が `True` の場合は、*indent* によって指定されたストリングが、ツリーの階層をコンポーネントに分けます。*full_path* を `True` でツリーを作成する場合は、outliner のパフォーマンスが落ちることがあります。

initial_state はフォルダの `State` リソースです(例: `XRTGEAR_FOLDERSTATE_CLOSED`)。

indent は、(ある行の)ノードのインデントのレベルを指定するストリングです。*indent* が `NULL` であれば、"\t"(tab 文字)が使用されます。

separator は、テキストをセクション別に区切るストリングで、複数列の outliner を表示する場合に使用されます。デフォルトは", "(カンマ)です。

open_escape は、(ノード・テキストではない)指示の部分の開始を示す文字です。*open_escape* が `NULL` の場合は、"("が使用されます。

closed_escape は、(ノード・テキストではない)指示の部分の終了を示す文字です。*closed_escape* が `NULL` の場合は、") "が使用されます。

XrtGearNodeCreateLWTreeFromStream()

データ・ファイルまたはその他の入力ストリームからツリーを作成します。

```
Widget
XrtGearNodeCreateLWTreeFromStream(
  Widget              parent,
  FILE                *stream,
  Boolean             full_path,
  XrtGearFolderState  initial_state,
  String              indent,
  String              separator,
  String              open_escape,
  String              close_escape
  )
```

parent は、新しく作成されたツリーの親です。*stream* はデータ・ファイルで、ファイルの各行は、ツリーのノードを表します。

*XrtGearNodeCreateLWTreeFromStream()*によって返される値は *parent* で、エラーが起きた場合は `Null` が返されます。

stream はデータ・ファイルまたは他の入力ストリームのポインタです。

full_path は、各ノードに対して完全なツリーの階層が提供されているか(*full_path* は `True`)、インデントのレベルから推測されるか(*full_path* は `False`)を示します。*full_path* が `True` の場合は、*indent* によって指定されたストリングが、ツリーの階層をコンポーネントに分けます。*initial_state* はツリーのフォルダの初期状態です。

indent は、(ある行の)ノードのインデントのレベルを指定するストリングです。*indent* が `NULL` であれば、"\t"(tab 文字)が使用されます。

separator は、テキストをセクション別に区切るストリングで、複数列の outliner を表示する場合に使用されます。デフォルトは", "(カンマ)です。

open_escape は、(ノード・テキストではない)指示の部分の開始を示す文字です。*open_escape* が NULL の場合は、”(“が使用されます。

closed_escape は、(ノード・テキストではない)指示の部分の終了を示す文字です。*closed_escape* が NULL の場合は、”)“が使用されます。

XrtGearNodeDestroy()

アイテムまたはフォルダ・ノードを破棄します。これはアイテム・ノードの API の一部です。

```
void
XrtGearNodeDestroy(
 Widget          node,
 )
```

XrtGearNodeGetChildren()

表示されるノードを持つ XrtList オブジェクトのポインタを返します。

```
XrtGearObject
XrtGearNodeGetChildren(
 Widget          node
 )
```

XrtGearNodeGetFolderState()

フォルダ・ノードの現在の状態を指定します。

```
XrtGearFolderState
XrtGearNodeGetFolderState(
 Widget              node
 )
```

node はフォルダ・ノードです。*XrtGearNodeGetFolderState()*は、次の値を返します。

XRTGEAR_FOLDERSTATE_CLOSED	フォルダは閉、子は表示されない
XRTGEAR_FOLDERSTATE_OPEN_FOLDERS	フォルダは開、フォルダの子だけ表示
XRTGEAR_FOLDERSTATE_OPEN_ITEMS	フォルダは開、アイテムだけ表示
XRTGEAR_FOLDERSTATE_OPEN_ALL	フォルダは開、全てのフォルダと子が表示

8

Progress ウィジェット

サンプル・プログラム ■ ウィジェットの概要
ラベルと範囲 ■ タイムアウトと TimeOut コールバック
ラベルの表示 ■ バーの表示
リソースの一覧 ■ 手続きと方法

8.1 サンプル・プログラム

Progress ウィジェットを使用して、タスクの状態をエンドユーザに知らせることができます。下にあるアプリケーションは、Progress ウィジェットの使用例です。Progress ウィジェットを作成、変更している部分は太字で示されています。

```
#include <stdio.h>
#include <stdlib.h>
#include <Xm/XrtGear.h>
#include <Xm/XrtProgress.h>
#include <Xm/PushB.h>
#include <Xm/RowColumn.h>
#include <Xm/Form.h>

Widget progress;
Display *display;

static void
buttonCB(Widget w, XtPointer client_data, XmAnyCallbackStruct *cb)
{
   int i = 0;

   for (i=1; i<=100; i++) {
     XtVaSetValues(progress,
       XmNvalue, i,
       NULL);
   XSync(display, True);
   usleep(100000);
  }
  exit(0);
}
```

```
void
main(int argc, char **argv)
{
  Widget toplevel, container, buttonArea, button;
  XtAppContext app;

  toplevel = XtVaAppInitialize(&app, "Progress1", NULL, 0,
              &argc, argv, NULL, NULL);
  container = XtVaCreateManagedWidget( "MyContainer",
    xmFormWidgetClass, toplevel,
    XmNorientation, XmVERTICAL,
    NULL);
  buttonArea = XtVaCreateManagedWidget( "MyButtonArea",
    xmRowColumnWidgetClass, container,
    XmNtopAttachment, XmATTACH_FORM,
    XmNleftAttachment, XmATTACH_FORM,
    XmNrightAttachment, XmATTACH_FORM,
    NULL);
  button = XtVaCreateManagedWidget("Press to start",
    xmPushButtonWidgetClass,buttonArea,
    NULL);
  XtAddCallback(button, XmNactivateCallback,
    (XtCallbackProc) buttonCB, NULL);
  progress = XtVaCreateWidget("prog",
    xmXrtProgressWidgetClass, container,
    XmNtopAttachment, XmATTACH_WIDGET,
    XmNtopWidget, buttonArea,
    XmNleftAttachment, XmATTACH_FORM,
    XmNrightAttachment, XmATTACH_FORM,
    XmNbottomAttachment, XmATTACH_FORM,
    NULL);
  display = XtDisplay(progress);
  XtManageChild(progress);
  XtRealizeWidget(toplevel);
  XtAppMainLoop(app);
}
```

このプログラムが実行されると、図 72 が表示されます。

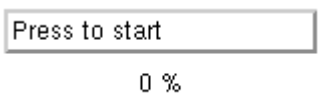


図 72　初期状態のサンプル Progres ウィジェット

エンドユーザが Press to start ボタンを押すと、Progress ウィジェットの XmNvalue リソース値がアップデートされ、ウィジェットは 1/10 秒毎に再描画されます。XmNvalue は Progress ウィジェットの現在値を 0 から 100 の数値のパーセントで表します。現在のパーセンテージは、図 73 のように数値と横向きのバーで示されます。

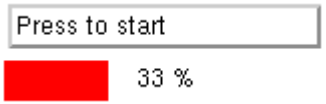


図 73　値を表示する Progress ウィジェット

実際のアプリケーションでは、Progres ウィジェットのポインタを他のアプリケーション･ウィンドウに渡すことができます。Progress ウィジェットを使用して、タスクの実行状況をモニタリングすることができます。

次のセクションからは、Progress ウィジェットのデフォルトの動作の変更方法や、TimeOut コールバックを使用したアップデート法等を解説します。

8.2　ウィジェットの概要

Include Files:　*<Xm/XrtProgress.h>*

Class Name:　XmXrtProgress

Class Hierarchy:　Core → XmPrimitive → XmXrtProgress

Class Pointer:　xmXrtProgressWidgetClass

8.3　ラベルと範囲

Progress ウィジェットによって表示される値は、次の三つのリソースによってコントロールされます。

- 値は `XmNvalue` に記憶される
- 最大値は `XmNmaximum` で指定される
- 最小値は `XmNminimum` で指定される

`XmNvalue` リソースは、`int` 型で Progress ウィジェットの現在の値を記憶します。この値は、表示する前にウィジェットによってフォーマットされます。(例えば、前ページの例では、値 43 は 43%というストリングにフォーマットされています。) フォーマットされたストリングは *label* と呼ばれ、`XmNxrtGearLabel` に記憶されます。*label* は読むことはできますが、セットできません。(値のフォーマットに関する詳細は 8.5 を参照してください。)

`XmNminimum` は `XmNvalue` に記憶できる最小整数を指定します。デフォルトでは、この値は 0 です。Progress ウィジェットが realize されると、`XmNminimum` の値は、`XmNvalue` の初期値になります。`XmNvalue` が `XmNminimum` より小さい値にセットされた場合は、`XmNminimum` の値が使用されます。

`XmNmaximum` は `XmNvalue` に記憶できる最大整数を指定します。デフォルトでは、この値は 100 です。アプリケーションが `XmNvalue` を `XmNmaximum` にセットすると、Progress ウィジェットによって監視されているタスクが完了したとみなされます。

この時点で、XmNxrtGearProgressCallback で指定されているコールバックが (*reason* は XRTGEAR_REASON_COMPLETE で) 呼び出されます。XmNvalue が XmNmaximum より大きい値にセットされた場合は、XmNmaximum の値が使用されます。

次のコードでは、8 ステップの処理を Progress ウィジェットが監視するようにセットしています。

```
XtVaSetValues(progress,
  XmNminimum, 1,
  XmNmaximum, 8,
  NULL);
```

XmNvalue は 1 から 8 の間の整数を持ちます。

通常、アプリケーションは、XmNvalue 値を最小値から最大値へ増加させていきます。

Progress Percentage

デフォルトの最小値 0 と最大値 100 を利用して、タスクが何パーセント進んだかを簡単に表示することができます。

XmNminimum または XmNmaximum がデフォルトより変更された場合は、タスクの進行状況を表すパーセンテージは、倍精度浮動小数点値として XmNxrtGearValuePercent に記憶されます。XmNxrtGearValuePercent は次の式を使用して、パーセントを計算します。

```
percentage = XmNvalue / (XmNmaximum - XmNminimum)
```

XmNxrtGearPercent の値は、XmNvalue、XmNmaximum、XmNminimum が変更されると再計算されます。

8.4 タイムアウトと TimeOut コールバック

セクション 8.1 では、Progress ウィジェットの値は XmNvalue の値を変更することによりアップデートされています。Progress ウィジェットの値は、次のようにタイムアウトのメカニズムを利用してアップデートすることもできます。

- 値を変更するコールバック・ルーチンを定義し、XmNxrtGearProgressCallback リソースで指定されるリストに追加する。
- 次に XmNxrtGearProgressTimeOut をセットして、タイムアウトのインターバルを定義する

XmNxrtGearProgressTimeOut は時間をミリ秒で指定し、タイムアウトに達すると、コールバック・ルーチンが呼び出されます。このリソースが 0 にセットされている場合 (デフォルト)は、タイムアウトは起こりません。

XmNxrtGearProgressCallback リストの各コールバックは、次の三つの時点で呼び出されます。

- タイムアウト処理が開始(XmNxrtGearProgressTimeOut がゼロ以外の値にセットされる)時。この時点では、コールバック構造体の *reason* メンバは XRTGEAR_REASON_PROGRESS_BEGIN にセットされます。
- タイムアウトが起こった時。*reason* は XRTGEAR_REASON_PROGRESS にセットされ、ウィジェットの値が変更されます。
- ウィジェットが最大値(XmNvalue=XmNmaximum)に達した時。*reason* は XRTGEAR_REASON_PROGRESS_COMPLETE にセットされます。XmNxrtGearProgressTimeOut の値が 0 でも、このコールは呼び出されます。

TimeOut コールバックが呼び出されると、次の構造体が渡されます。

```
typedef struct {
  XrtGearReason   reason;
  XEvent      *event;
  int         minimum;
  int         maximum;
  int         value_percent;
  XmString        label;
  int         value;
  Boolean continue_progression;
} XrtGearProgressCallbackStruct;
```

(*reason* が XRTGEAR_REASON_PROGRESS の時に変更できる)*label*、*value*、*continue_progression* を除くとこれらのフィールドは全て read-only です。*continue_progression* を False にセットすると、現在進行中の処理が止まります。

次は、TimeOut コールバックを使用して Progress ウィジェットの値をコントロールする例です。

```
static void
progressCB(Widget w, XtPointer client_data, XtPointer call_data)
{
    XrtGearProgressCallbackStruct *cbs =
        (XrtGearProgressCallbackStruct *) call_data;
    if (cbs->reason == XRTGEAR_REASON_PROGRESS) {
        cbs->value += 1;
    } else if (cbs->reason == XRTGEAR_REASON_PROGRESS_COMPLETE) {
       exit(0);
    }
}
static void
start_timerCB(Widget w, XtPointer client_data,
   XmAnyCallbackStruct *cb)
{
     if (progress != NULL) {
       XtVaSetValues(progress,
         XmNxrtGearProgressTimeOut, 100,
       NULL);
     }
     XtAddCallback(progress, XmNxrtGearProgressCallback,
       progressCB, NULL);
}
```

```
static void
stop_timerCB(Widget w, XtPointer client_data,
     XmAnyCallbackStruct *cb)
{
     if (progress != NULL) {
       XtVaSetValues(progress,
          XmNxrtGearProgressTimeOut, 0,
       NULL);
     }
}
...
     progress = XtVaCreateWidget("prog",
       xmXrtProgressWidgetClass, container,
       XmNtopAttachment,  XmATTACH_WIDGET,
       XmNtopWidget,      buttonArea,
       XmNleftAttachment, XmATTACH_FORM,
       XmNrightAttachment,XmATTACH_FORM,
       XmNbottomAttachment,   XmATTACH_FORM,
       NULL);
```

Progress ウィジェットに表示されるパーセンテージは、100 に達するまで 1/10 秒毎にインクレメントされます。(100 に達した時点までウィジェットが存在します。)値は、progressCB ルーチン内より、TimeOut コールバック構造体の *value* メンバを変更することにより、インクレメントできます。

ウィジェットの破棄

コールバック内からは、Progress ウィジェットを破棄することはできません。タスク終了時にウィジェットが自分自身を破棄する Progress ウィジェットを作成するには、*XmCreateXrtProgressShellSimple()*を使用してください。

```
Arg args[3];
int n = 0;

XtSetArg(args[n], XmNminimum, 1); n++;
XtSetArg(args[n], XmNmaximum, 10); n++;
progress = XmCreateXrtProgressShellSimple(parent,
  "myshelltitle", args, n);
```

この手続きは、"myshelltitle"という名のウィンドウに Progress ウィジェットを作成し、このウィンドウは、処理が完了する(XmNvalue=XmNmaximum)と自動的に破棄されます。

XtAppAddTimeOut() を呼び出して作成されたタイムアウト手続きや、*XtAppAddWorkProc()*を呼び出して作成された work 手続きから Progress ウィジェットを破棄することもできます。

8.5 ラベルの表示

Progress ウィジェットには、ラベル(ストリングとしてフォーマットされているカレント値)をコントロールするリソースが幾つかあります。これらのリソースを変更して、次の事項が指定できます。

- ラベルの表示、非表示

- ラベルを表示する際に使用するプリント･フォーマット
- 処理の特定な時点で表示される任意のテキスト(LabelFormat コールバックを使用)
- ラベルのフォント、行揃え、色
- ストリングや pixmap がフィットするようにサイズを変更するかどうか

ラベルの非表示

デフォルトでは、Progress ウィジェットが realize されたりアップデートされるとラベルが表示されます。表示させないようにするには、`XmNxrtGearLabelShow` を `False` にセットしてください。Progress ウィジェットは、処理状況を示すバーのみを表示します。

`XmNxrtGearLabelShow` を `False` にセットすることにより、Progress ウィジェット以外の場所にこのラベルを表示することができます。(詳細は、LabelFormat コールバックの解説を参照してください。)

ラベルのフォーマット

デフォルトでは、Progress ウィジェットのカレント値が表示用にラベルにフォーマットされると、パーセントを示す%がラベルに加えられます。ラベルのフォーマットのストリングは、`XmNxrtGearLabelFormat` リソースをセットすることにより指定できます。

このリソースは、下記の一つまたは複数のストリングを値として取り増す。

`%C`	終了までの推定時間(*hours:minutes:seconds* のフォーマット、*seconds* のみでも可)
`%E`	推定合計時間
`%M`	ラベルの最大値(`XmNmaximum` 値)
`%N`	ラベルの最小値(`XmNminimum` 値)
`%P`	パーセンテージ(`XmNxrtGearValuePercent` 値 x 100)
`%T`	経過時間
`%V`	カレント値(`XmNvalue` 値)
`%%`	一つのパーセント文字

下記は、フォーマット･ストリングの例です。

```
"processing part %V of %M"
"%C remaining"
"time: %T of %E"
```

`XmNxrtGearLabelFormat` のデフォルト値は、`"%P %%"`(現在のパーセンテージ、空白、そしてパーセントを示す%文字)です。

経過時間、合計時間、推定時間の値は、簡易手続きを使用して取得します。これらの方法は 8.8 を参照して下さい。(合計時間と推定時間は、タスクが進行するにつれてより正確な値になります。)

ラベル・フォーマットの Callback

Progress ウィジェットでは、表示されるラベルを変更したり、フォーマットされた現在のラベルを取得し他のウィジェットへ記憶させることができます。これらは、XmNxrtGearLabelFormatCallback を使用してコールバックを定義することにより実行できます。

リストの各コールバックには、XrtGearProgressCallbackStruct 構造体が渡されます。

```
typedef struct {
  XrtGearReason   reason;
  XEvent      *event;
  int         minimum;
  int         maximum;
  int         value_percent;
  XmString        label;
  int         value;
  Boolean     continue_progression;
} XrtGearProgressCallbackStruct;
```

各コールバックは、新しいラベル作成前と、作成後の二回呼び出されます。コールバック構造体の *reason* は、最初のコールで XRTGEAR_REASON_FORMAT_LABEL_BEGIN に、次のコールで XRTGEAR_REASON_FORMAT_LABEL_END にセットされます。*reason* が XRTGEAR_REASON_FORMAT_LABEL_BEGIN の時は、コールバック構造体の *label* を変更することにより、表示されるラベルを変更することができます。

フォーマットされたラベルを他のウィジェットに記憶させる

LabelFormat コールバックが、*reason* が XRTGEAR_REASON_FORMAT_LABEL_END にセットされて呼び出されると、このコールバックは、フォーマットされた現在のラベルを取得し、それを他のウィジェットに記憶させることができます。次のコードでは、カレント・ラベルを取り出し、XmLabel ウィジェットにラベル・ストリングとして記憶させています。

```
void
labelCB(Widget w, XtPointer client_data,
   XrtGearProgressCallbackStruct *call_data)
{
    if (call_data->reason == XRTGEAR_REASON_FORMAT_LABEL_END) {
      XtVaSetValues(label,
          XmNlabelString, call_data->label,
          NULL);
}
```

このコードは、指定された XmLabel ウィジェットでラベルを表示します。

ラベルのフォント

ラベル・テキストのフォントを指定するには、XmNfontList を次のようにセットしてください。

```
XtVaSetValues(progress,
  XtVaTypedArg, XmNfontList,
  XmRString, "-*-times-medium-*-*--12-*-iso8859-*",
    strlen("-*-times-medium-*-*--12-*-iso8859-*") + 1,
  NULL);
```

ラベルの行揃え

デフォルトでは、ラベル･テキストは、Progress ウィジェットの表示領域の中央に表示されます。位置を変更するには、図 74 のように XmNxrtGearLabelAlignment をセットして下さい。



図 74　Progress ウィジェットのラベルの行揃え

デフォルト値は XRTGEAR_ALIGN_CENTER です。

ラベルの色

ラベル･テキストの色は、テキストがバー上にあるかどうかによって異なります。バー上にあるテキストは、(XmNbackground で指定される)ウィジェットの背景色を使用します。その他のテキストは、(XmNforeground で指定される)ウィジェットの前景色を使用します。

ウィジェットのサイズの再計算

XmNrecomputeSize は、ラベル･ストリングがフィットするようにウィジェットのサイズを変更するかどうかコントロールします。デフォルトでは、このリソースは True にセットされており、サイズが再計算されます。

8.6　バーの表示

Progress ウィジェットでは、次のようなバー表示をコントロールできます。

- 表示されるバーの種類
- 高さと幅のマージン
- 表示されるバーの数とそれらの間隔(バーが分散(discrete)したコンポーネントの場合)
- 使用するアイコン(バーの種類がアイコンの場合)
- バーの色

バーの種類

デフォルトでは、Progress ウィジェットには、連続型(分散(discrete)型ではない)の横向きのバーが表示されます。このバーの種類は XmNxrtGearProgressType をセットすることにより変更できます。有効な値は、図 75 にある通りです。



図 75　バーの種類

デフォルト値は XRTGEAR_PROGRESS_BAR です。

バーの表示がスペースによる制限を受ける場合は、次の決まりに従います。

- XmNxrtGearProgressType が XRTGEAR_PROGRESS_DISCRETE_BAR で、バーが表示できない場合は、XRTGEAR_PROGRESS_BAR が使用されます。
- XmNxrtGearProgressType が XRTGEAR_PROGRESS_STACKING_ICONS で、アイコンが表示できない場合は、XRTGEAR_PROGRESS_ICON が使用されます。
- XmNxrtGearProgressType が XRTGEAR_PROGRESS_ICON で、アイコンが表示できない場合は、XRTGEAR_PROGRESS_BAR が使用されます。

高さと幅のマージン

XmNmarginHeight と XmNmarginWidth は、ウィジェットの枠線と表示されるバーとの間のスペースを指定します。デフォルトでは、どちらのリソースも 2pixel にセットされています。これらのリソースを変更すると図 76 のように表示されます。

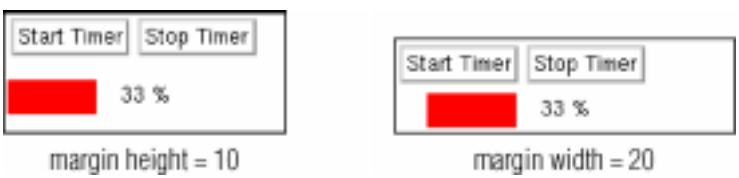


図 76　高さと幅のマージン

バーの数

XmNxrtGearProgressType が XRTGEAR_PROGRESS_DISCRETE_BAR にセットされている場合、分散型のバーの最大数を XmNxrtGearBarCount で指定することができます。図 77 は、このリソースを変更した例です。

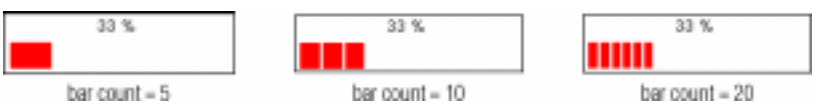


図 77　分散型のバーの数

デフォルトでは、XmNxrtGearBarCount は 10 で、バーまたはアイコンは 10 個まで表示されます。(実際に表示されるバー/アイコンの数は、タスクの進行状況に対応します。)

バーの間隔

XrtGearProgressType が XRTGEAR_PROGRESS_DISCRETE_BAR または XRTGEAR_PROGRESS_STACKING_ICONS の場合、デフォルトの各バー/アイコンの間隔は 2pixel です。この間隔を変更するには、図 78 のように XmNxrtGearBarSpacing をセットしてください。

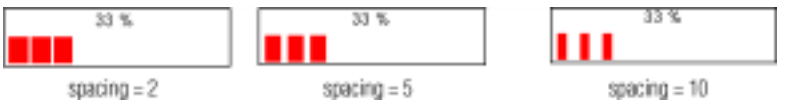


図 78 分散型のバーの間隔

バーの間隔やバーの数の設定によって、表示領域に分散型のバーが表示できなくなった場合は、一本の連続型のバーが表示されます。

バーのアイコン

バーの種類が XRTGEAR_PROGRESS_ICON または XRTGEAR_PROGRESS_STACKING_ICONS にセットされた時は、XmNxrtGearIcon で表示されるアイコンを指定できます。このリソースは、XrtGearIcon 型の値のポインタを取ります。

アイコンの作成法や操作法は、13.4 を参照して下さい。

バーの色

バーが描画される色は、XmNxrtGearBarColor で指定できます。このリソースは Pixel 型で、XmNbackgorund 等の標準の Motif の色のリソースと同じ方法でセットできます。

8.7 リソースの一覧

Progress ウィジェットのリソースと、よく使用される継承されたリソースの一覧です。リソース名の後は、データ型、デフォルト値、そしてリソースと共に使用される手続きです。**C** は、そのリソースが create(例: *XtCreateWidget()*)で定義でき、**S** は set(例; *XtSetValues()*)でき、**G** は get(例: *XtGetValues()*)で使用できることを意味します。

XmNxrtGearBarColor Pixel Red CSG

Progress ウィジェットによって表示されるバーの色を指定します。XmNxrtGearProgressType が XRTGEAR_PROGRESS_BAR または XRTGEAR_PROGRESS_DISCRETE_BAR の時に適用されます。

XmNxrtGearBarCount	int	10	CSG

XmNxrtGearProgressType が XRTGEAR_PROGRESS_DISCRETE_BAR の時に、表示される分散型のバーの最大数を指定します。

XmNxrtGearBarSpacing	Dimension	2	CSG

各ユニットの間隔を pixel で指定します。XmNxrtGearProgressType が XRTGEAR_PROGRESS_DISCRETE_BAR または XRTGEAR_PROGRESS_STACKING_ICONS の時に適用されます。

XmNxrtGearDoubleBuffer	Boolean	True	CSG

ウィジェットがダブル・バッファリング機能を利用するかどうか指定できます。この機能を使用すると、パフォーマンスは良くなりますが、サーバに対するメモリの要求が高くなります。

XmNfontList	XmFontList	dynamic	CSG

ラベル・テキストに使用されるフォント・リストを指定します。

XmNXrtGearIcon	XrtGearIcon	NULL	CSG

XmNxrtGearProgressType が XRTGEAR_PROGRESS_ICON の時に使用されるイメージを指定します。

XmNxrtGearLabel	XmString	NULL	G

表示されている、カレント・ラベルを指定します。

XmNxrtGearLabelAlignment	XrtGearAlignment	XRTGEAR_ALIGN_CENTER	CSG

バーに対するラベルの位置を指定します。有効な値は、XRTGEAR_ALIGN_BEGINNING、XRTGEAR_ALIGN_CENTER、XRTGEAR_ALIGN_END です。

XmNxrtGearLabelFormat	String	“%P %%”	CSG

値のフォーマットを指定します。次の特別文字が定義されています。

%C	終了までの推定時間(*hours:minutes:seconds* のフォーマット、*seconds* のみでも可)
%E	推定合計時間
%M	ラベルの最大値(XmNmaximum 値)
%N	ラベルの最小値(XmNminimum 値)
%P	パーセンテージ(XmNxrtGearValuePercent 値 x 100)
%T	経過時間
%V	カレント値(XmNvalue 値)
%%	一つのパーセント文字

XmNxrtGearLabelFormatCallback XtCallbackList NULL G

新しいフォーマットを作成する時に呼び出されるコールバックを指定します。これらのコールバックは、XmNvalue が変化すると呼び出されます。各ルーチンには、XrtGearProgressCallbackStruct のポインタが渡されます。コールバックは、*XtAddCallback()*を使用してリストに追加してください。コールバックは次の様に構成されます。

```
void
CallbackProcedure(
  Widget  widget;
  XtPointer   client_data;
  XtPointer   call_data /* points to an
                XrtGearProgressCallbackStruct */
)
```

XrtGearProgressCallbackStruct は次のように定義されます。

```
typedef struct {
  XrtGearReason   reason;
  XEvent      *event;
  int         minimum;
  int         maximum;
  int         value_percent;
  XmString        label;
  int         value;
  Boolean continue_progression;
} XrtGearProgressCallbackStruct;
```

reason は XRTGEAR_REASON_FORMAT_LABEL_BEGIN(新しいラベルの作成前)または XRTGEAR_REASON_FORMAT_LABEL_END(ラベル作成後)です。*reason* が XRTGEAR_REASON_FORMAT_LABEL_BEGIN の時は、コールバック構造体の *label* メンバを変更して他のラベルを指定できます。*event* はコールバックを呼び出した XEvent で、コールバックがプログラムによって呼び出された場合は、NULL です。(その他のフィールドは、Progress コールバックに使用されます。)

XmNxrtGearLabelShow Boolean True CSG

ラベルを表示するかどうか指定します(True の場合は表示)。ラベルの値は、XmNxrtGearLabelFormatCallback にコールバック手続きを提供することによって取得できます。

XmNmarginHeight Dimension 2 CSG

ウィジェットの枠線とバー/アイコンとの上下のスペースを指定します。

XmNmarginWidtht Dimension 2 CSG

ウィジェットの枠線とバー/アイコンとの左右のスペースを指定します。

XmNmaximum int 100 CSG

XmNvalue の最大値を指定します。XmNvalue が XmNmaximum と同じになると、XmNxrtGearProgressCallback で指定されるコールバックが、*reason* が XRTGEAR_REASON_COMPLETE で呼び出されます。

XmNminimum int 0 CSG

XmNvalue の最小値を指定します。

XmNxrtGearProgressCallback XtCallbackList NULL CSG

Progress の状態が変化した時に呼び出されるコールバックを指定します。各ルーチンには、XrtGearProgressCallbackStruct のポインタが渡されます。コールバックは、*XtAddCallback()*を使用してリストに追加して下さい。コールバックは次のように構成されます。

```
void
CallbackProcedure(
  Widget  widget;
  XtPointer   client_data;
  XtPointer   call_data /* points to an
               XrtGearProgressCallbackStruct */
)
```

XrtGearProgressCallbackStruct は次のように定義されます。

```
typedef struct {
  XrtGearReason   reason;     /* read-only */
  XEvent      *event;     /* read-only */
  int         minimum;    /* read-only */
  int         maximum;    /* read-only */
  int         value_percent;  /* read-only */
  XmString        label;
  int         value;
  Boolean     continue_progression;
} XrtGearProgressCallbackStruct;
```

これらのコールバックは、次の三つの時点で呼び出されます。最初は、ウィジェットが progress を開始した時です。この時、*reason* は XRTGEAR_REASON_PROGRESS_BEGIN にセットされます。

コールバックは、タイムアウトが起こった時にも呼び出されます。(インターバルは XmNxrtGearProgressTimeOut で指定できます。) この時、*reason* は XRTGEAR_REASON_PROGRESS です。*value* は編集可能で、現在の状態やウィジェットの進行をアップデートすることができます。

コールバックは XmNvalue が XmNmaximum 値にセットされた時にも呼び出されます。この時、*reason* は XRTGEAR_REASON_PROGRESS_COMPLETE です。

event はコールバックを呼び出した XEvent で、コールバックがプログラムによって呼び出された場合は NULL です。*minimum* と *maximum* は、進行状況を示す最小値と最大値です。*value_percent* は、現在何パーセントまで進行しているか示します。

進行を止めるには、*reason* が XRTGEAR_REASON_PROGRESS の時に *continume_progression* を False にセットしてください。

XmNxrtGearProgressTimeOut unsigned long 0 CSG

タイムアウトのインターバルをミリ秒で指定します。タイムアウトが起きると、XmNxrtGearProgressCallback で定義されている timeout コールバックが呼び出されます。

このリソースが 0(デフォルト値)にセットされている場合は、タイムアウトは起こりません。

XmNxrtGearProgressType　XrtGearProgressType　XRTGEAR_PROGRESS_BAR　CSG

バーの表示を指定します。有効な値は次の通りです。

XRTGEAR_PROGRESS_BAR	一本の横向きのバー
XRTGEAR_PROGRESS_DISCRETE_BAR	分割された横向きのバー
XRTGEAR_PROGRESS_ICON	進行状況によって移動する一つのアイコン
XRTGEAR_PROGRESS_STACKING_ICON	複数のアイコン

バーの表示がスペースによる制限を受ける場合は、次の決まりに従います。

- XmNxrtGearProgressType が XRTGEAR_PROGRESS_DISCRETE_BAR で、バーが表示できない場合は、XRTGEAR_PROGRESS_BAR が使用されます。
- XmNxrtGearProgressType が XRTGEAR_PROGRESS_STACKING_ICONS で、アイコンが表示できない場合は、XRTGEAR_PROGRESS_ICON が使用されます。
- XmNxrtGearProgressType が XRTGEAR_PROGRESS_ICON で、アイコンが表示できない場合は、XRTGEAR_PROGRESS_BAR が使用されます。

XmNrecomputeSize　Boolean　True　CSG

ストリングまたは pixmap がフィットするように、ウィジェットのサイズを変更するかどうか指定します。

XmNvalue　int　0　CSG

カレント値を指定します。この値は、必ず XmNminimum 以上、XmNmaximum 以下です。値がこの範囲以外の場合は、範囲内の一番近い値にセットされ、警告メッセージが出力されます。

XmNxrtGearValuePercent　double　0　G

進行状況を 0 から 100 のパーセントで表示します。パーセントは次の式で計算されます。

```
percentage = XmNvalue / (XmNmaximum - XmNminimum)
```

XmNvalue、XmNmaximum、XmNminimum が変化すると、このパーセントは再計算されます。

8.7.1 Progress ウィジェットによって継承されるリソース

Resource	Inherited From
XmNaccelerators	Core
XmNancestorSensitive	Core
XmNbackground	Core
XmNbackgroundPixmap	Core
XmNborderColor	Core
XmNborderPixmap	Core
XmNborderWidth	Core
XmNbottomShadowColor	Primitive
XmNbottomShadowPixmap	Primitive
XmNcolormap	Core
XmNdepth	Core
XmNdestroyCallback	Core
XmNforeground	Primitive
XmNheight	Core
XmNhelpCallback	Primitive
XmNhighlightColor	Primitive

Resource	Inherited From
XmNhighlightOnEnter	Primitive
XmNhighlightPixmap	Primitive
XmNinitialResourcesPersistent	Core
XmNmappedWhenManaged	Core
XmNnavigationType	Primitive
XmNscreen	Core
XmNsensitive	Core
XmNtopShadowColor	Primitive
XmNtopShadowPixmap	Primitive
XmNtranslations	Core
XmNtraversalOn	Primitive
XmNuserData	Primitive
XmNwidth	Core
XmNx	Core
XmNy	Core

8.8 手続きと方法

Progress の手続きと方法の一覧です。

XmCreateXrtProgress()

Progress ウィジェットを作成します。

```
Widget
XmCreateXrtProgress(
  Widget  parent,
  String  name,
  Arg     *arglist,
  int     argcount
)
```

parent は親、*name* は作成されるウィジェット名、*arglist* は *argcount* 引数のリストです。

XmCreateXrtProgressShellSimple()

Progress ウィジェットを含むシェルを作成し、タスクが終了するとそれを破棄します。

```
Widget
XmCreateXrtProgressShellSimple(
  Widget  parent,
  char    *title,
  Arg     *arglist,
  int     argcount
)
```

parent は親、*title* はシェルのタイトル、*arglist* は *argcount* 引数のリストです。

XmIsXrtProgress()

ウィジェットが Progress ウィジェットであれば `True` を返します。

```
Boolean
XmIsXrtProgress(
  Widget  widget
  )
```

XrtGearProgressCalcTimeElapsed()

`XmNvalue` が最後に `XmNminimum` 値にセットされてからの経過時間を計算します。

```
time_t
XrtGearProgressCalcTimeElapsed(
  Widget  progress
  )
```

progress は Progress ウィジェットです。

XrtGearProgressCalcTimeToCompletion()

現在進行しているタスクが、現時点から終了するまでにかかる推定時間を計算します。

```
time_t
XrtGearProgressCalcTimeToCompletion(
  Widget  progress
  )
```

progress は Progress ウィジェットです。推定時間は、タスクが進行するにつれて正確さが増します。

XrtGearProgressCalcTimeTotal()

現在進行しているタスクが終了するまでにかかかる合計時間を計算します。

```
time_t
XrtGearProgressCalcTimeTotal(
  Widget  progress
  )
```

progress は Progress ウィジェットです。計算される値は、タスクが進行するにつれて正確さが増します。

9

Combo Box ウィジェット

サンプル・プログラム ■ ウィジェットの概要

メニューのコントロール ■ 表示の属性のコントロール

ユーザ・インタラクション ■ 自動マッチング ■リソースの一覧

手続きと方法 ■ トランスレーションとアクション

9.1 サンプル・プログラム

Combo Box を使用して、エンドユーザは選択リストより項目を簡単に選択することができます。次のコードは、ComboBox を作成する簡単なアプリケーションです。

```
#include <X11/Intrinsic.h>
#include <Xm/Xm.h>
#include <Xm/Text.h>
#include <Xm/XrtGear.h>
#include <Xm/XrtGearCombo.h>

int
main (int argc, char *argv[])
{
    XtAppContext app_context;
    Widget toplevel, combobox;
    static String picklist[] = {"John Jones","Harry Smith",
      "Jean Jones","Mary Martin","Harry Hayes",NULL};

    toplevel = XtVaAppInitialize(&app_context,
      "Combo", NULL, 0,
      &argc, argv, NULL, NULL, 0);

    combobox = XtVaCreateManagedWidget("combobox",
      xmXrtGearComboWidgetClass, toplevel,
      XmNxrtGearPickList, picklist,
      NULL);
    XtRealizeWidget(toplevel);
    XtAppMainLoop(app_context);
}
```

このプログラムがコンパイル、実行されると、図 79 が表示されます。



図 79 サンプル・プログラムによって作成された Combo Box

Combo Box ウィジェットが作成される時に二つの子ウィジェットも同時に作成されます。一つは、*text field* と呼ばれる Text ウィジェットで現在選択されているアイテムを表示します。もう一つは、選択アイテムを変更する際にエンドユーザがクリックする Arrow ウィジェットです。Combo Box が最初に表示される時は、エンドユーザがまだ何も選択していない状態ですので、text field は空です。

このプログラムでは、`XmNxrtGearPickList` が Combo Box の有効な値を定義します。ユーザが Arrow ウィジェットをクリックすると、`XmNxrtGearPickList` で定義された値を持つポップアップ・メニュー(XmList クラス)が現れます。図 80 は、このリソースで定義される *pick list* です。

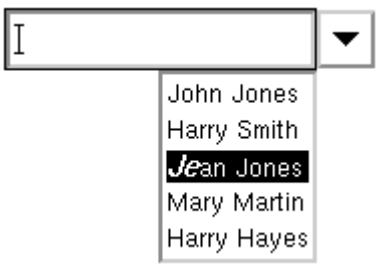


図 80 Combo Box ウィジェットに含まれるポップアップ・メニュー

エンドユーザは、pick list からアイテムをクリックして選択します。アイテムが一度選択されると、ポップアップ・メニューは画面から消え、text field に選択された値が表示されます。text field を変更せずに、このメニューを非表示にするには、もう一度 Arrow キーをクリックします。

9.2 ウィジェットの概要

Include Files: *<Xm/XrtGearCombo.h>*

Class Name: XmXrtGearCombo

Class Hierarchy: Core → Composite → Constraint → XmManager →
XmXrtManager → XmXrtComboBox

Class Pointer: xmXrtGearComboWidgetClass

9.3 メニューのコントロール

Combo Box では、次の pick list の属性をコントロールすることができます。

- pick list の内容
- pick list の選択されたアイテム
- pick list のウィジェット ID
- 選択されたアイテムを持つ text field のウィジェット ID

上記の他、メニューのアイテムが選択される時に呼び出されるコールバックを指定することができます。

9.3.1 pick list の指定

ポップアップ・メニューに表示される pick list を指定するには、String 配列を作成し、この配列を指すように XmNxrtGearPickList をセットしてください。このリストは、リソースがセットされた後でも変更できますので、外部イベントやエンドユーザのアクションに対応するようにポップアップ・メニューを変更することができます。

pick list のインデックス

ポップアップ・メニューからアイテムを選択するには、XmNxrtGearPickListIndex リソースをセットしてください。例えば、ポップアップ・メニューの最初のアイテムが選択される場合には、このリソースを 0 に、二番目のアイテムが選択される場合は、このリソースは 1 にセットしてください。エンドユーザがポップアップ・メニューからアイテムを選択すると、これに対応して XmNxrtGearPickListIndex も変化します。リストにアイテムが何も表示されない場合は、XmNxrtGearPickListIndex は XRTGEAR_NOVALUE にセットされています。

Pick List の変更

pick list のウィジェット ID を取得するには、XmNxrtGetGearMenuList リソースの値を取り出してください。このリソースは read-only でセットすることはできません。pick list は XmList ウィジェットで、このウィジェットの ID 取得後は、親の Combo Box にコントロールされていないリソースであれば、どれでもセットできます。XmNborderWidth 、 XmNheight 、 XmNitemCount 、 XmNitems 、 XmNselectionPolicy、XmNvisibleItemCount、XmNwidth は、親ウィジェットに管理され、プログラミングによってセットすることはできません。次の例では、pick list の XmNfontList リソースをセットして、アイテムを表示するフォントを指定しています。

```
static String fontlist =
"-adobe-helvetica-medium-r-normal--12-120-75-75-p-67-iso8859-1,\
-adobe-helvetica-bold-o-normal--14-140-75-75-p-82-iso8859\
-1=highlight";
XtVaGetValues(combobox,
  XmNxrtGearMenuList, &menulist,
  NULL);
XtVaSetValues(menulist,
  XtVaTypedArg, XmNfontList, XmRString,
  fontlist, strlen(fontlist)+1,
  NULL);
```

XmNxrtGearSearchPolicy が XRTGEAR_SEARCH_MULTICHAR にセットされると、図 81 のようにエンドユーザが pick list 上でアイテムを選択する時に入力するキーがハイライト・フォントで表示されるようになります。(詳細は、XmNxrtGearSearchPolicy の項を参照して下さい。)

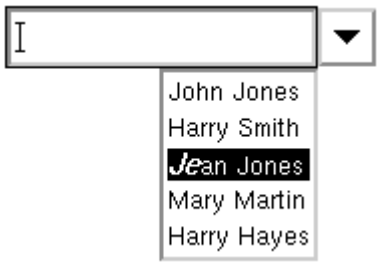


図 81　pick list でハイライト・フォントを使用

Combo Box のユーザ・インタラクションは、9.5 を参照してください。

Text Field の修正

Combo Box ウィジェットが作成されると、子の Text ウィジェットも自動的に作成され、現在選択されているアイテムを表示します。

Combo Box のウィジェット ID を *XtVaSetValues()* または *XtVaGetValues()* に渡すことにより、次のリソースを変更または取得できます。

XmNblinkRate	XmNcolumns
XmNcursorPosition	XmNcursorPositionVisible
XmNeditable	XmNfontList
XmNmarginHeight	XmNmarginWidth
XmNmaxLength	XmNpendingDelete
XmNresizeWidth	XmNselectThreshold
XmNselectionArray	XmNselectionArrayCount
XmNvalue	XmNvalueWcs
XmNverifyBell	

他のテキスト・リソースをセットしたり、コールバックを追加するには、XmNxrtGearTextField の値を取得し、text field のウィジェット ID を取り出してください。このリソースは read-only でセットすることはできません。

9.3.2 メニューのコールバック

ポップアップが表示されたり非表示になったりする時に呼び出されるコールバックのリストを指定することができます。このリストは、XmNxrtGearMenuCallback リソースに記憶されます。このリストの各コールバックには、次の構造体が渡されます。

```
typedef struct {
  int     reason; /* read-only */
  XEvent  *event; /* read-only */
  int     position; /* read-only */
  String  value;
  Boolean doit; /* initially True */
} XrtGearMenuCallbackStruct;
```

reason は、どの時点でコールバックが呼び出されたか示します。メニューが表示される時であれば *reason* は XRTGEAR_REASON_MENU_BEGIN、非表示になる時は XRTGEAR_REASON_MENU_END です。*position* は、現在選択されているアイテムを示し、*value* は、テキスト・フィールドに表示されている(またはこれから表示される)ストリングです。

このコールバックを利用して、ポップアップ・メニューの表示を制止したり、エンドユーザによるメニュー・アイテムの選択を不可にすることができます。メニューが表示されないようにするには、*reason* が XRTGEAR_REASON_MENU_BEGIN の時に *doit* を False にセットしてください。エンドユーザによるメニュー選択を無視するには、*reason* が XRTGEAR_REASON_MENU_END の時に *doit* を False にしてください。

9.4 表示の属性のコントロール

Combo Box ウィジェットでは、次の属性をコントロールできます。

- 表示される矢印(arrow)の種類
- 作成時の矢印のウィジェット・クラス
- テキスト・フィールドに対するポップアップ・メニューの配置

テキスト・フィールドと矢印ウィジェットの間隔

ドロップ・ダウン・リストに表示されるアイテムの数

9.4.1 矢印(arrow)

表示される矢印は、XmNxrtGearArrowType リソースで指定できます。このリソースは、ウィジェットの作成時にのみセットできます。有効な値は図 82 の通りです。



図 82　Combo Box の矢印

デフォルト値は XRTGEAR_ARROW_WEDGE です。

テキスト・フィールドに対する矢印ボタンの位置は、XmNstringDirection の値で指定されます。矢印の種類は、XmNxrtGearArrowType を XRTGEAR_ARROW_CUSTOM にセットして独自の矢印を使用することもできます。次のコードは、任意の矢印を指定する例です。

```
#include <stdio.h>
#include <Xm/Xm.h>
#include <Xm/ArrowB.h>
#include <Xm/XrtGearCombo.h>
  static void exposeArrow(Widget w, XtPointer client_data,
    XtPointer call_data);

int
main(int argc, char *argv[])
{
  XtAppContext app_context;
  Widget combo, toplevel, arrow = NULL;
  static String picklist[] =
    { "Joe Blow", "Harry Wong", "Sally Ann", NULL };
  Widget incr = NULL;

  toplevel = XtVaAppInitialize(&app_context, "Combo", NULL, 0,
    &argc, argv, NULL, NULL);
  combo = XtVaCreateManagedWidget("combo",
    xmXrtGearComboWidgetClass,toplevel,
    XmNxrtGearPickList,     picklist,
    XmNxrtGearArrowType,    XRTGEAR_ARROW_CUSTOM,
    NULL);
  XtVaGetValues(combo, XmNxrtGearArrow, &arrow, NULL);
  if (arrow) {
    XtAddCallback(arrow, XmNexposeCallback, exposeArrow, NULL);
    XtVaSetValues(arrow,
      XmNhighlightThickness,0,
      XmNshadowThickness,   0);
  }
  XtRealizeWidget(toplevel);
  XtAppMainLoop(app_context);
}
static void
exposeArrow(Widget w, XtPointer client_data, XtPointer call_data)
{
   static GC       gc = NULL;
   XGCValues       values;
   Xpoint     points[10];
   Dimension       width, height, shadow;
   Pixel      fg, bg;

   if (XtWindow(w) == NULL) {
     return;
   }
   XtVaGetValues(w, XmNshadowThickness, &shadow,
     XmNforeground,    &fg,
     XmNbackground,    &bg,
     XmNwidth,     &width,
     XmNheight,    &height,
```

```
      NULL);
    width -= (2 * shadow);
    height -= (2 * shadow);
    if (!gc) {
      values.fill_style = FillSolid;
      values.foreground = fg;
      values.background = bg;
      gc = XtGetGC(w,
        GCFillStyle | GCForeground | GCBackground,
        &values);
    }
    points[0].x = shadow;
    points[0].y = shadow;
    points[1].x = shadow + (width / 2);
    points[1].y = shadow + height;
    points[2].x = shadow + width;
    points[2].y = shadow;
    points[3].x = shadow;
    points[3].y = shadow;
    XFillPolygon(XtDisplay(w), XtWindow(w), gc, points, 4,
       Convex, CoordModeOrigin);
}
```

このプログラムが実行されると、次のウィジェットが作成されます。



図 83　任意の矢印を表示する Combo Box ウィジェット

矢印は，*exposeArrow()*によって作成され、矢印が描画される時に呼び出されます。

Arrow Class の指定

デフォルトでは、Combo Box によって作成される矢印ウィジェットは、XmArrowButton ウィジェット(クラスは `xmArrowButtonWidgetClass`)です。異なるクラスの矢印ウィジェットを作成するには、`XmNxrtGearArrowClass` を用いてクラスを指定して下さい。このリソースは、Combo Box が作成される時にのみセットできます。

矢印の変更

`XmNxrtGearArrow` リソースの値を取り出すことにより、矢印ウィジェットの ID を取得することができます。このリソースは、read-only でセットすることはできません。このウィジェット ID を得ることにより、矢印の表示や動きは、そのリソースによって変更できるようになります。

9.4.2 メニュー・アイテムの配置

XmNxrtGearAlignment は、テキスト・フィールドに相対したポップアップ・メニューの位置を指定します。有効な値は図 84 の通りです。



図 84　Combo Box のメニューの位置

デフォルトでは、XmNxrtGearAlignment は XRTGEAR_MENU_ALIGN_BOTH にセットされています。

9.4.3 間隔

デフォルトでは、テキスト・フィールドと矢印の間にはスペースはありません。これらの二つの子ウィジェットの間にスペースを置くには、図 85 のように XmNxrtGearSpacing をセットしてください。

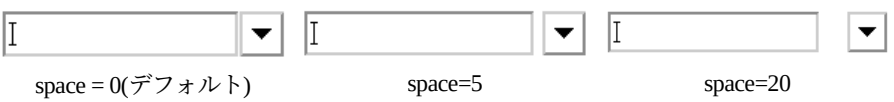


図 85 テキスト・フィールドと矢印のスペース

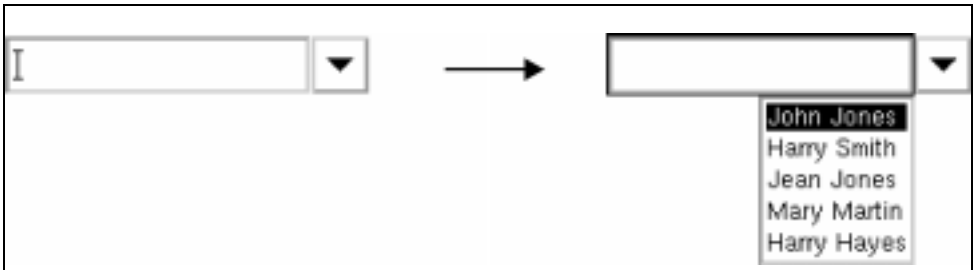
スペースは、pixel で測定されます。

9.4.4 表示されるアイテムの数

デフォルトでは、最大で 10 個のアイテムが、ドロップ・ダウン・リストに表示されます。この数は、XmNxrtGearVisibleItemCount によって変更することができます。但し、表示されるアイテムの数に関係なく、picklist のアイテム数がこのリソースで指定される数より大きい場合は、縦方向のスクロールバーが表示されます。

9.5 ユーザ・インタラクション

図 86 は、Combo Box の一般的なユーザ・インタラクションの例です。尚、トランスレーションとアクションのリストは、9.9 を参照してください。



マウス

- **矢印ボタンをクリックして、ポップアップ・メニューを表示する**
- **矢印ボタンをもう一度クリックする、または** osfCancel **押すかメニュー以外の場所をクリックして、メニューを非表示にする**
- **ポップアップ・メニューのアイテムをクリックして、テキスト・フィールドに記憶されている値を変更する**

キーボード

- Down **キーを押してポップアップ・メニューを表示する**
- Up **または** Down **キーを押してポップアップ・メニュー間を上下に移動する**
- **選択するアイテムの最初の文字のキーを押して、そのアイテムに移動する。次にキーを押すと、他のアイテムに移動するか、または検索にもう一つのキーが追加される(この動作は** `XmNxrtGearSearchPolicy` **に依存する)**
- **リターンキーを押して、ポップアップ・メニューからアイテムを選択する**
- Escape **または** Ctrl-UP **を押してメニューを非表示にし、フォーカスを** Combo Box **のテキスト・フィールドに戻す**

図 86　Combo Box のユーザ・インタラクション

pick list の検索

ポップアップ・メニューが表示されている時に、エンドユーザは、キー入力して、pick list のアイテムを検索することができます。`XmNxrtGearSearchPolicy` は、この検索の動作をコントロールし、`XRTGEAR_SEARCH_SINGLE`(各キー入力により一回毎にリセットされる検索、デフォルト)または `XRTGEAR_SEARCH_MULTICHAR`(キー入力が追加される、複数のキーによる検索)にセットできます。

図 87 は、`XmNxrtGearSearchPolicy` の例です。

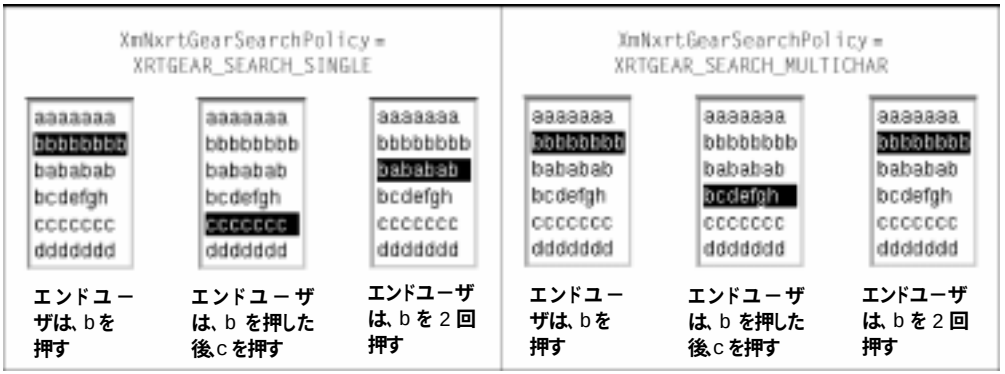


図 87　ポップアップ・メニューの検索

9.6　自動マッチング

XmNxrtGearAutoComplete が True の場合、picklist がセットされていれば、自動マッチングが行われます。例えば、picklist が *red*、*green*、*blue* という値を持っている場合、(XmNxrtGearAutoComplete が True であれば)、'g'を入力すると *green* というアイテムがフィールドに表示され、ユーザがそのフィールドを出ると field exit コールバックがアクティブになります。ユーザによってフィールドに入力された文字は、通常の前景色と背景色を使用しますが、picklist から自動的にマッチングされた部分はその逆の色で表示されます。ユーザがそのフィールドから出て自動マッチングによって選択された値が確定されると、XmNValue は変更されます。

9.7　リソースの一覧

Combo Box のリソースの一覧です。リソース名の後は、データ型、デフォルト値、そしてリソースと共に使用される手続きです。**C** は、そのリソースが create(例: *XtCreateWidget()*)で定義でき、**S** は set(例; *XtSetValues()*)でき、**G** は get(例: *XtGetValues()*)で使用できることを意味します。

XmNxrtGearAutoComplete　Boolean　False　CSG

picklist がセットされている時に、自動マッチングによって値の補完を行うかどうか指定します。複数の文字による検索がサポートされています。キーが連続して入力されると、picklist のカレント・アイテムが、そのキー入力にマッチする文字から始まるエレメントにセットされます。
キー入力(keypress)一回づつが新たな検索を行います。

XmNxrtGearAlignment　XrtGearMenuAlignment　XRTGEAR_MENU_ALIGN_BOTH　CSG

テキスト・フィールドに対するメニューの配置を指定します。有効な値は、次の通りです。

XRTGEAR_MENU_ALIGN_BEGINNING	メニューとテキスト・フィールドの左側を揃える
XRTGEAR_MENU_ALIGN_END	メニューとテキスト・フィールドの右側を揃える
XRTGEAR_MENU_ALIGN_BOTH	メニューとテキスト・フィールドの横幅を揃える

XmNxrtGearArrow　Widget　dynamic　G

子矢印ボタンのウィジェット ID を指定します。このリソースは、read-only です。

XmNxrtGearArrowClass　WidgetClass　xmArrowButtonWidgetClass　C

子矢印ボタンのウィジェット・クラスを指定します。このリソースは、作成時にのみセットできます。

XmNxrtGearArrowType　XrtGearArrowType　XRTGEAR_ARROW_WEDGE　CSG

矢印のタイプを指定します。有効な値は、次の通りです。

XRTGEAR_ARROW_NONE	pixmap は何も指定されていない
XRTGEAR_ARROW_WEDGE	Windows95 で使用されている矢印と類似した pixmap
XRTGEAR_ARROW_WITH_UNDERBAR	Windows 3.1 で使用されている矢印と類似した pixmap
XRTGEAR_ARROW_CUSTOM	任意の矢印を使用

XmNxrtGearMenuCallback　XtCallbackList　NULL　CSG

メニューが表示される前と、メニューが非表示になる前に呼び出されるコールバックのリストを指定します。各ルーチンには、XrtGearMenuCallbackStruct のポインタが渡されます。コールバックは、*XtAddCallback()*を使用してリストに追加してください。コールバックは次のように構成されます。

XrtGearMenuCallbackStruct は次のように定義されます。

```
void
CallbackProcedure(
  Widget  widget,
  XtPointer   client_data,
  XtPointer   call_data /* points to an
                XrtGearMenuCallbackStruct */
)
```

reason は XRTGEAR_REASON_MENU_BEGIN(メニュー表示前)または、XRTGEAR_REASON_MENU_END(メニュー非表示前)のどちらかです。*event* は、コールバックの原因となったイベントで、通常はキーまたはボタンのイベントです。コールバックがプログラムによって呼び出された場合は、*event* は NULL です。*position* は、現在選択されているメニュー・アイテムの位置で、1 は最初のアイテム、2 は二番目のアイテムを指します。*value* は、*reason* が XRTGEAR_REASON_MENU_BEGIN の時にのみアクセス可能な、テキストフィールドに表示されるストリングで、これは変更可能です。

doit メンバの動きは、*reason* の値に依存します。*reason* が XRTGEAR_REASON_MENU_BEGIN であれば、*doit* を False にセットすることによりメニューは表示されません。*reason* が XRTGEAR_REASON_MENU_END の時は、*doit* を False にすると、エンドユーザによるメニュー選択を無視するように指示します。

XmNxrtGearMenuList　Widget　dynamic　G

(ポップアップ・メニューに含まれる)pick list のウィジェット ID を指定します。このリソースは read-only です。

pick list は、XmList です。

XmNxrtGearPickList　String *　NULL　CSG

メニューに表示されるストリングの NULL 終端リストを指定します。

XmNxrtGearPickListIndex　int　XRTGEAR_NOVALUE　CSG

XmNxrtGearPickList で指定されるストリング・リストの、カレント・アイテムの位置を指定します。このリストの最初のアイテムは position 0、二番目は position 1 というように続きます。リストに何も表示されない時は、このリソースは XRTGEAR_NOVALUE にセットされます。

XmNxrtGearSearchPolicy　XrtGearSearchPolicy　XRTGEAR_SEARCH_SINGLE　CSG

エンドユーザが一つまたは複数のキーを押した時の検索の方法を指定します。有効な値は次の通りです。

XRTGEAR_SEARCH_SINGLE	一つの文字による検索。エンドユーザがキーを押すと、そのキーの文字から始まるアイテムがカレント・アイテムにセットされる。キーが押される度に検索が開始される。
XRTGEAR_SEARCH_MULTICHAR	複数の文字による検索。エンドユーザがキー続けて押すと、それらの文字から始まるアイテムがカレント・アイテムにセットされる。キーが押される度に検索キーが追加される。

XmNxrtGearSpacing　Dimension　0　CSG

テキスト・フィールドと矢印ウィジェットの間を pixel で指定します。

XmNxrtGearTextField　Widget　dynamic　G

子フィールドのウィジェット ID を指定します。このリソースは read-only です。

XmNxrtGearNodeVisibleItemCount　int　10　CSG

ドロップ・ダウン・リストに表示されるアイテムの数をコントロールします。指定された数よりアイテム数が多い場合は、縦方向のスクロールバーが表示されます。

9.7.1 Combo Box によって継承されるリソース

次の表は、Combo Box によって親のクラスから継承されるリソースとそれらを定義する親のクラスの一覧です。尚、詳細は、Motif と Xt に関する資料を参照してください。

Resource	Inherited From
XmNaccelerators	Core
XmNancestorSensitive	Core
XmNbackground	Core
XmNbackgroundPixmap	Core
XmNborderColor	Core
XmNborderPixmap	Core
XmNborderWidth	Core
XmNbottomShadowColor	XmManager
XmNbottomShadowPixmap	XmManager
XmNchildren	Composite
XmNcolormap	Core
XmNdepth	Core
XmNdestroyCallback	Core
XmNforeground	XmManager
XmNheight	Core
XmNhelpCallback	XmManager
XmNhighlightColor	XmManager
XmNhighlightPixmap	XmManager

Resource	Inherited From
XmNinitialFocus	XmManager
XmNinitialResourcesPersistent	Core
XmNinsertPosition	Composite
XmNmappedWhenManaged	Core
XmNnavigationType	XmManager
XmNnumChildren	Composite
XmNscreen	Core
XmNsensitive	Core
XmNshadowThickness	XmManager
XmNstringDirection	XmManager
XmNtopShadowColor	XmManager
XmNtopShadowPixmap	XmManager
XmNtranslations	Core
XmNtraversalOn	XmManager
XmNwidth	Core
XmNuserData	XmManager
XmNx	Core
XmNy	Core

9.8 手続きと方法

XmCreateXrtGearComboBox()

フィールドと矢印ボタンを子として Combo Box を作成します。矢印ボタンの名前は arrow、フィールドの名前は field です。Combo Box のウィジェット ID が返されます。

```
Widget
XmCreateXrtGearComboBox(
  Widget  parent,
  String  name,
  Arg     *arglist,
  int     argcount
)
```

parent は親、*name* は作成されたフィールド名です。*arglist* は、フィールドに適用される *argcount* 引数のリストです。全ての子は管理されます。

XmIsXrtGearComboBox()

ウィジェットが Combo Box ウィジェットであれば、True を返します。

```
int
XmIsXrtGearComboBox(
  Widget  widget
  )
```

9.9 トランスレーションとアクション

9.9.1 トランスレーション

Event	Action / Actions
<Key>Cancel	Cancel()
<Key>Escape	Cancel()
<Key>osfHelp	Menu(PrimitiveHelp)
<Key>Return	Menu(ListKbdActivate)
<Key>osfActivate	Menu(ListKbdActivate)
Ctrl<Key>osfDown	Menu(Hide) Menu(TraverseNextTabGroup)
Ctrl<Key>osfUp	Menu(Hide)
Ctrl<Key>osfRight	Cancel() Menu(TraverseRight)
Ctrl<Key>osfLeft	Cancel() Menu(TraverseLeft)
Ctrl<Key>a	Menu(ResetSearch)
Ctrl<Key>n	Menu(FindNext)
Ctrl<Key>p	Menu(FindPrevious)
<Key>osfDown[a]	Menu(Display)
<Key>osfDown[a]	Menu(ListNextItem)
<Key>osfUp	Menu(ListPrevItem)
<Key>osfPageDown	Menu(ListNextPage)
<Key>osfPageUp	Menu(ListPrevPage)
<Key>Tab	Menu(Tab) Menu(TraverseNextTabGroup)
<Key>osfBackSpace	Menu(BackSpace)
<Key>osfDelete	Menu(BackSpace)
<Key>	Menu(Navigate)

a: ピック・リストが非表示の時にosfDown を押すと、ピック・リストが表示される
ピック・リストが表示されている時にosfDown を押すと、ハイライトを次のアイテムに移動する

9.9.2 アクション

Cancel()

アイテムを選択せずに、pick list を非表示にする。

Menu()

メニュー操作を行う。このアクションは、次の一つのパラメータを取ります。

PrimitiveHelp	リスト・ウィジェットと関連している help コールバックを呼び出し、primitive ヘルプを表示する。
ListKbdActivate	メニューを非表示にする。`XmNxrtGearMenuCallback` リストのコールバックが呼び出される。ハイライトされているアイテムの値が、コールバック構造体の *value* メンバに渡され、それがフィールドに表示される。
Display	メニューを表示し、選択されているアイテムをハイライトする。`XmNfontList` で指定されるフォント・リストが、"highlight"というタグ付きのフォントを持っていたら、このフォントが、ハイライトされたテキストに使用される。
Hide	リストやフィールド・ウィジェットの状態を変更せずにリストを非表示にする。
ResetSearch	カレント・アイテムのハイライトをはずして、元のフォントに戻し、最初のアイテムをリストの最上部に移動してハイライトする。検索ストリングはクリアする。
FindPrevious	検索ストリングにマッチする、前のストリングを検索する。該当するアイテムがみつかった場合は、それがハイライトされて表示される。該当するアイテムがない場合は、何も変更されない。
FindNext	検索ストリングにマッチする、次のストリングを検索する。該当するアイテムがみつかった場合は、それがハイライトされて表示される。該当するアイテムがない場合は、何も変更されない。
ListPrevious	ハイライトを前のアイテムに移動する。
ListNext	ハイライトを次のアイテムに移動する。
ListNextPage	次のページを表示する。
ListPreviousPage	前のページを表示する。
Tab	フィールドやリスト・ウィジェットを変更せずに、メニューを非表示にし、tab グループ内の次のウィジェットにトラバースする。
Navigate	エンドユーザによって入力されたキーをつなげて検索ストリングを構成する。現在のキーにマッチするストリングが存在しなければ、何も変更されない。マッチするかどうかは、大文字小文字に関係なく調べられる。リストの最後に到達したら、検索は、リストの最初から開始される。
BackSpace	検索ストリングから、エンドユーザによる最後のキー入力を削除する。

TraverseLeft	ウィジェットの tab グループ内で、左側のアクティブなアイテムに移動する。必要であれば、循環する。
TraverseRight	ウィジェットの tab グループ内で、右側のアクティブなアイテムに移動する。必要であれば、循環する。
TraverseNextTabGroup	ウィジェット階層内の次のタブ・グループをアクティブなタブ・グループにする。

10

PanedWindow

サンプル・プログラム ■ ウィジェットの概要

ユーザ・インタラクション ■ リソースの一覧

手続きと方法 ■ トランスレーションとアクション

10.1 サンプル・プログラム

XrtPanedWindow は、子ウィジェットを縦または横方向にタイルのように配置する composite ウィジェットです。下記は、PanedWindow ウィジェットを作成する簡単なプログラムです。

```
#include <Xm/XrtGear.h>
#include <Xm/XrtPanedW.h>
#include <Xm/PushB.h>
#include <Xm/Form.h>

int
#ifndef _NO_PROTO
main(int argc, char *argv[])
#else
main(argc, argv)
int argc;
char *argv[];
#endif
{
 XtAppContext app_context;
 Widget toplevel, form, button, paned;

 /* Initialize Intrinsics */
 toplevel = XtVaAppInitialize(&app_context, "Paned", NULL, 0,
         &argc, argv, NULL, NULL);
 form = XtVaCreateManagedWidget("form",
         xmFormWidgetClass, toplevel,
         NULL);

 paned = XtVaCreateManagedWidget("paned",
         xmXrtPanedWindowWidgetClass, form,
         XmNleftAttachment, XmATTACH_FORM,
         XmNrightAttachment, XmATTACH_FORM,
```

```
            XmNtopAttachment, XmATTACH_FORM,
            XmNbottomAttachment, XmATTACH_FORM,
            NULL);

    button = XtVaCreateManagedWidget("Button1",
            xmPushButtonWidgetClass, paned,
            NULL);
    button = XtVaCreateManagedWidget("Button2",
            xmPushButtonWidgetClass, paned,
            NULL);
    button = XtVaCreateManagedWidget("Button3",
            xmPushButtonWidgetClass, paned,
            NULL);
    button = XtVaCreateManagedWidget("Button4",
            xmPushButtonWidgetClass, paned,
            NULL);
    XtRealizeWidget(toplevel);
    XtAppMainLoop(app_context);

    return(0);
  }
```

次の図は、このプログラムがコンパイルされ実行された結果です。



図 88　PanedWindow

子は、上から下または左から右の方向に表示されます(最初の子は上または左、最後の子は下または右)。PanedWindow は、縦方向の場合は一番大きい子の幅、横方向の場合は一番大きい子の高さに合わせてリサイズし、その他の子もそのサイズに合わせてリサイズされます。縦方向の場合は、PanedWindow の高さは、全ての子の高さとそれらの間隔、上下のマージンの合計と同じです。

pane のサイズをユーザがコントロールできるように、ほとんどの子には sash が作成されます。sash は、コントロールする子の下または右に表示される小さなボックスです。(図 88 ではボタン 1 から 3 の下に表示されています。) ユーザは、この sash をマウスやキーボードで操作することにより、pane のサイズを調整することができます。(ユーザ･インタラクションの詳細は P261 を参照してください。)

PanedWindow は constraint ウィジェットですので、各子供に対する constraint を作成、管理します。

各 pane の最小や最大のサイズを指定することにより、子供は最小値より小さくなったり最大値より大きくなったりすることができません。

最小値が最大値と同じ値に設定されている場合は、sash は表示されません。

10.2 ウィジェットの概要

PanedWindow

Include Files: <Xm/XrtPanedW.h>

Class Name: XmXrtPanedWindow

Class Hierarchy: Core → Composite → Constraint → XmManager → XmXrtManager → XmXrtPanedWindow

Class Pointer: XmXrtPanedWindowWidgetClass

Sash

Include Files: <Xm/XrtSash.h>

Class Name: XmXrtGearSash

Class Hierarchy: Core → Primitive → XmXrtGearSash

Class Pointer: XmXrtGearSashWidgetClass

sash ウィジェットはパブリック・ウィジェットではないので、エンド・ユーザがアクセスすることはできません。sash は、XrtPanedWindow の子としてのみ作成されます。

10.3 ユーザ・インタラクション

図 89 は、PanedWindow とのユーザ・インタラクションを示しています。尚、トランスレーションとアクションのリストは、10.6 にあります。



MOUSE:
- sash をクリックし、その pane にフォーカスを移動する
- sash をドラッグし、pane をリサイズする

KEYBOARD:
- Tab を押して、pane 間を移動する、Shift-Tab によって pane を逆方向に移動する
- osfUP/osfDown/osfLeft/osfRight を押して、pane を(任意の方向に)一行分リサイズする、Ctrl と一緒に使用して pane を表示領域分リサイズする
- スペースバーを押して、カレント pane の子をアクティブにする
- Up/Down/Left/Right を押して、アクティブな pane の子の間を移動する

図 89　PanedWindow とのユーザ･インタラクション

10.4　リソースの一覧

PanedWindow ウィジェットのリソースの一覧です。リソース名の後は、データ型、デフォルト値、そしてリソースと共に使用される手続きです。**C** は、そのリソースが create(例: *XtCreateWidget()*)で定義でき、**S** は set(例; *XtSetValues()*)でき、**G** は get(例: *XtGetValues()*)で使用できることを意味します。

10.4.1　PanedWindow のリソース

XmNxrtGearMarginHeight　Dimension　3　CSG

XrtPanedWindow の上下の端と子の間の距離を指定します。

XmNxrtGearMarginWidth　Dimension　3　CSG

XrtPanedWindow の左右の端と子の間の距離を指定します。

XmNxrtGearOrientation　unsigned char　XmVERTICAL　CSG

レイアウトを縦(XmVERTICAL)または、横(XmHORIZONTAL)に指定します。

XmNxrtGearPaneResizeCallback XtCallbackList NULL CSG

ユーザ・インタラクションによって行われた pane のリサイズの前後に呼び出されるコールバックのリストを指定します。各コールバックには、XrtGearPaneResizeCallbackStruct が渡されます。

```
typedef struct {
 XrtGearReason   reason; /* read-only */
 Xevent          *event; /* read-only */
 Widget          child; /* read-only */
 Int             position; /* read-only */
 Dimension       width;
 Dimension       height;
 Boolean doit;
} XrtGearPaneResizeCallbackStruct;
```

reason は XRTGEAR_REASON_RESIZE_BEGIN(pane がリサイズされる前) または、XRTGEAR_REASON_RESIZE_END(pane がリサイズされた後)のどちらかです。*event* は、コールバックの原因となったイベントで、通常はキーまたはボタンのイベントです。このコールバックがプログラムによって呼び出された場合は、*event* は NULL です。*child* は、リサイズされるウィジェット、*position* は paned ウィジェットの子のリスト内でのその子供の位置です。*width* と *height* は、ウィジェットの新しい幅と高さで、*reason* が XRTGEAR_REASON_RESIZE_BEGIN であれば変更することができます。また、*reason* が XRTGEAR_REASON_RESIZE_BEGIN の時に *doit* を False にすれば、リサイズを制止することができます。*reason* が XRTGEAR_REASON_RESIZE_BEGIN の時に *width* と *height* が変更されると、XmNxrtGearMinimum と XmNxrtGearMaximum の値に関係なく、その新しい値が使用されます。

XmNxrtGearRecomputeSize Boolean True CSG

このリソースが True の時は、XrtPanedWindow は、子供が追加、削除またはサイズ変更された時に XrtPanedWindow 自身とその子供のサイズを計算します。False の時には、サイズはアップデートされません。

XmNxrtGearSashHeight Dimension 10 CSG

sash の高さを指定します。この値は、sash が大きすぎる場合に XrtPanedWindow によって変更されることがあります。

XmNxrtGearSashIndent Position -10 CSG

sash の横方向のインデントを指定します。sash は、この値が正の数であると XrtPanedWindow の左/上からインデントされ、負の数であると XrtPanedWindow の下/右からインデントされます。オフセットが XrtPanedWindow の幅より大きい場合は、sash は端に表示されます。

XmNxrtGearSashShadowThickness Dimension dynamic CSG

sash の影の幅を指定します。

XmNxrtGearSashWidth Dimension 10 CSG

sash の幅を指定します。この値は、sash が大きすぎる場合は、XrtPanedWindow によって変更されることがあります。

XmNxrtGearSeparatorShow Boolean True CSG

各 pane の間にセパレータを作成するかどうかを指定します。

XmNxrtGearSpacing Dimension 8 CSG

pane 間の距離を指定します。

10.4.2 PanedWindow の constraint リソース

XmNxrtGearAllowResize Boolean False CSG

このリソースによって、アプリケーションは、XrtPanedWindow が pane のサイズ変更のリクエストを許可するかどうかを指定することができます。このフラグは、XrtPanedWindow とその子が realize された後にのみ効果があります。このフラグが `True` にセットされると、XrtPanedWindow は pane の幅/高さ変更のリクエストを許可します。`False` の場合は、pane のサイズ変更のリクエストは拒否されます。

XmNxrtGearMinimum Dimension 1 CSG

pane の最小サイズを指定します。この値は、必ず最大サイズの値以下にしてください。

XmNxrtGearMaximum Dimension 1000 CSG

pane の最大サイズを指定します。この値は、必ず最小サイズの値以上にしてください。

XmNxrtGearPosition int XmLAST_POSITION CSG

(sash は含まれない)pane の子のリスト内でのウィジェットの位置を指定します。値は、0 より大きく、(値が指定される時の)リストの子の数より小さい整数にしてください。値が 0 の場合は、子はリストの先頭に配置されます。デフォルトである `XmLAST_POSITION` は、子をリストの最後尾に配置します。その他の値は無視されます。`XtGetValues` は、(`XtGetValues` が呼び出された時の)子リスト内での、ウィジェットの位置を返します。

一つのウィジェットが子リストに挿入されると、そのウィジェットの `XmNxrtGearPosition` より大きい値を持つ既存の子の位置は一つづつ大きくなります。`XmNxrtGearPosition` で `XtSetValues` をセットすると、指定されたウィジェットが子リストより削除され、リスト内でそのウィジェットの位置より大きい値を持つ既存の子供の位置が一つづつ減少し、その後、ウィジェットがリストに挿入されます。

XmNxrtGearSkipAdjust Boolean False CSG

`True` にセットされた時は、アプリケーションは XrtPanedWindow が自動的にリサイズしないように指定できます。

10.4.3 PanedWindow によって継承されるリソース

PanedWindow は親のクラスからリソースを継承します。下記は、これらのリソース及びそれらを定義する親のクラスのアルファベット順のリストです。尚、詳細に関しては、Motif 及び Xt 関連の資料を参照してください。

Resource	Inherited From
XmNaccelerators	Core
XmNancestorSensitive	Core
XmNbackground	Core
XmNbackgroundPixmap	Core
XmNborderColor	Core
XmNborderPixmap	Core
XmNborderWidth	Core
XmNbottomShadowColor	XmManager
XmNbottomShadowPixmap	XmManager
XmNchildren	Composite
XmNcolormap	Core
XmNdepth	Core
XmNdestroyCallback	Core
XmNforeground	XmManager
XmNheight	Core
XmNhelpCallback	XmManager
XmNhighlightColor	XmManager
XmNhighlightPixmap	XmManager

Resource	Inherited From
XmNinitialFocus	XmManager
XmNinitialResourcesPersistent	Core
XmNmappedWhenManaged	Core
XmNnavigationType	XmManager
XmNnumChildren	Composite
XmNscreen	Core
XmNsensitive	Core
XmNshadowThickness	XmManager
XmNstringDirection	XmManager
XmNtopShadowColor	XmManager
XmNtopShadowPixmap	XmManager
XmNtranslations	Core
XmNtraversalOn	XmManager
XmNunitType	XmManager
XmNuserData	XmManager
XmNwidth	Core
XmNx	Core
XmNy	Core

10.5 手続きと方法

XmCreateXrtPanedWindow()

指定された親、名前、引数を使用して XrtPanedWindow ウィジェットを作成します。

```
Widget
XmCreateXrtPanedWindow(
 Widget  parent,
 String  name,
 Arg     *arglist
 int     argcount
 )
```

parent は親、*name* は作成される PanedWindow の名前です。*arglist* は PanedWindow に適用される *argcount* 引数のリストです。全ての子が管理されます。

XmIsXrtPanedWindow()

ウィジェットが PanedWindow または、XrtPanedWindow のサブクラスであれば `True` を返します。

```
Boolean
XmIsXrtPanedWindow(
 Widget  widget,
 )
```

XrtPanedWindowGetPaneList()

PanedWindow の(セパレータや sash は含まない)全ての子 pane に対するポインタ(`Widget*`型)を持つ XrtList を返します。このリストは `NULL` にすることはできませんが、ゼロのエレメントを持つことができます。

```
XrtGearObject
XrtPanedWindowGetPaneList(
 Widget  widget,
 )
```

10.6 トランスレーションとアクション

10.6.1 トランスレーション

Event	Action / Actions
~Ctrl ~Shift ~Meta ~Alt <Btn1Down>	SashAction(Start)
~Ctrl ~Shift ~Meta ~Alt <Btn1Motion>	SashAction(Move)
~Ctrl ~Shift ~Meta ~Alt <Btn1Up>	SashAction(Commit)
~Ctrl ~Shift ~Meta ~Alt <Btn2Down>	SashAction(Start)
~Ctrl ~Shift ~Meta ~Alt <Btn2Motion>	SashAction(Move)
~Ctrl ~Shift ~Meta ~Alt <Btn2Up>	SashAction(Commit)
<Key>osfActivate	PrimitiveParentActivate()
<Key>osfCancel	PrimitiveParentCancel()
<Key>osfHelp	Help()
Ctrl <Key>osfUp	SashAction(Key,Large,Up)
<Key>osfUp	SashAction(Key,Default,Up)
Ctrl <Key>osfRight	SashAction(Key,Large,RIght)
<Key>osfRight	SashAction(Key,Default,RIght)
Ctrl <Key>osfDown	SashAction(Key,Large,Down)
<Key>osfDown	SashAction(Key,Default,Down)
Ctrl <Key>osfLeft	SashAction(Key,Large,Left)
<Key>osfLeft	SashAction(Key,Default,Left)
~Shift ~Meta ~Alt <Key>Return	PrimitiveParentActivate()
Shift ~Meta ~Alt <Key>Tab	PrevTabGroup()
~Meta ~Alt <Key>osfTab	NextTabGroup()

10.6.2 アクション

Help()

`XmNhelpCallback` に対するコールバックを呼び出します。ウィジェットに対するコールバックが存在しない場合は、一番近い先祖のウィジェットの help コールバックを呼び出します。

NextTabGroup()

キーボード・フォーカスを次の tab グループに移動します。デフォルトでは、各 pane と sash で一つの tab グループを構成します。

PrevTabGroup()

キーボード・フォーカスを前の tab グループに移動します。デフォルトでは、各 pane と sash で一つの tab グループを構成します。

SashAction(start)

pane の枠線のインタラクティブな描画をアクティブにします。

SashAction(move)

sash がカーソルの位置をトラッキングできるようにします。一つの pane が最小または最大サイズに近づくと、サイズが調整されます。

SashAction(commit)

sash の動きを終了させます。

SashAction(key, increment, direction)

sash アクションの原因がキーボード・イベントの場合、キーボード・フォーカスがある sash は指定された *increment* と *direction* に従って移動します。デフォルトでは、一行分調整されます。Large は sash を表示領域分、調整します。*direction* は、`UP`、`Down`、`Left`、`Right` で指定できます。

*SashAction()*は XrtPanedWindow の直接的なアクションではなく、XrtPanedWindow によって作成された Sash コントロールのアクションです。

11

Widget Tip

Widget Tip について ■ Widget Tip の使用
Widget Tip をウィジェットに追加 ■ Widget List の取得
ヘルプ･バルーン ■ 表示コールバック
表示のタイミング ■ Widget tip を不可に設定
ヘルプ･テキストの表示と削除 ■ トランスレーションの定義
リソースの一覧 ■ 手続きと方法

11.1 Widget Tip について

Widget Tip ユーティリティは、エンドユーザがウィジェット上にカーソルを置いたとき、そのウィジェットに対するテンポラリなヘルプ･バルーンを表示します。ヘルプ･バルーンは、カーソルの近くに表示される長方形のウィンドウで、この中にヘルプ･テキストが表示されます。カーソルがウィンドウから離れたり、キー、ボタンが押されると、このウィンドウは画面から消えます。

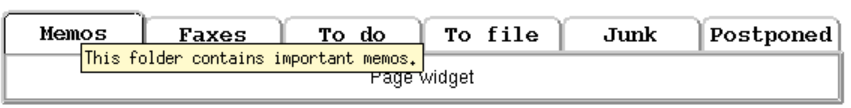


図 90　ヘルプ･テキストが表示されているウィジェット

ウィジェットにヘルプ･バルーンを表示させるには、そのウィジェットに Widget Tip のリソースを加えてください。Widget Tip のリソースは、リソース･ファイルまたは *XrtGearTipsVaSetValues()*または *XrtGearTipsSetValues()*を使用してセットすることができます。(*XtVaSetValues()*は使用できません。)

11.2 Widget Tip の使用

特定なシェル･ウィジェットとその子ウィジェットで Widget Tip を使用できるようにするには、次のように *XrtGearTipsEnable()*を呼び出してください。

```
#include <Xm/XrtTips.h>
Widget toplevel;
XrtGearTipsEnable(toplevel, True);
```

*XrtGearTipsEnable()*の最初の引数は、シェル･ウィジェット内のウィジェットです。二番目の引数は、Widget Tip の使用可、不可を示す Boolean です。不可にするには、この引数を `False` にセットして下さい。

また、Widget Tip を使用するには、必ず、*Xm/XrtTips.h* をインクルードしてください。

*XrtGearTipsEnable()*が呼び出された後に、子ウィジェットがシェルに追加された場合、Widget Tip は、この新しいウィジェットでは使用できません。(この新しい子ウィジェットで Widget Tip を使用するには、*XrtGearTipsEnable()*をもう一度呼び出して下さい。)

11.3 Widget Tip をウィジェットに追加

Widget Tip をウィジェットに追加するには、*XrtGearTipsVaSetValues()*を呼び出して `XmNxrtGearText` リソースをセットしてください。

```
XmString str;

str = XmStringCreateLtoR("my help text",
        XmSTRING_DEFAULT_CHARSET);
XrtGearTipsVaSetValues(widget,
  XmNxrtGearText, str,
  NULL);
```

または、*XrtGearTipsSetValues()*を使用することもできます。

```
ArgList args;
Cardinal num_args;

XrtGearTipsSetValues(widget, args, num_args);
```

*XtVaSetValues()*または *XtSetValues()*を使用して、`XmNxrtGearText` や Widget Tip の他のリソースはセットできません。

`XmNxrtGearText` には XmString 型の値が必要です。`XmNxrtGearText` が `NULL`(デフォルト値)にセットされている場合は、ヘルプ･バルーンは表示されません。

Widget Tip リソースの取得

Widget Tip のリソースを取り出すには、*XrtGearTipsVaGetValues()* または *XrtGearTipsGetValues()* を使用して下さい。これらの方法の動作は、Motif の *XtVaGetValues()*と *XtGetValues()*と同様です。

11.4 Widget List の取得

シェルの子ウィジェットのリストを取り出すには、*XrtGearTipsGetWidgetList()*を呼び出して下さい。

```
Widget toplevel, *list;
int num_widgets;
Boolean enabled_only = False;

Boolean retrieved;
retrieved = XrtGearTipsGetWidgetList(toplevel,
enabled_only, &list, &num_widgets);
```

*XrtGearTipsGetWidgetList()*は、子ウィジェットのリストのポインタとリスト内のウィジェットの数を取得します。このリストを使用して、アプリケーションのウィジェットに Widget Tip を使用することができます。

既に Widget Tip が使用可能になっているウィジェットのリストを返すには、二番目の引数、*enabled_only* を `True` にセットしてください。

*XrtGearTipsGetWidgetList()*は、リストの取得に成功すると `True` を返します。

11.5 ヘルプ･バルーン

次のリソースは、Widget Tip の表示をコントロールします。

- `XmNxrtGearForeground` と `XmNxrtGearBackground` リソースは、ヘルプ･バルーンの前景色と背景色を指定します。これらのリソースは、Pixel 型です。
- `XmNxrtGearFontList` は、ヘルプ･テキストに使用されるフォント･リストを指定します。

`XmNxrtGearTipsBorderType` は、ヘルプ･テキストの枠線を指定します。有効な値は図 91 の通りです。デフォルト値は、`XRTGEAR_BORDER_PLAIN` です。

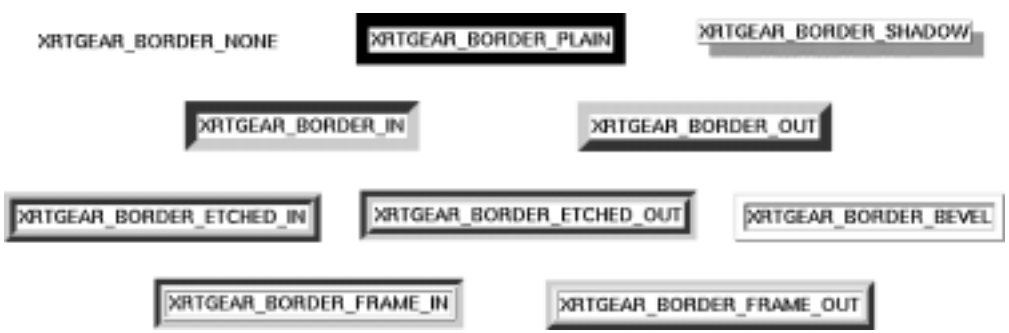


図 91　Widget Tip の枠線

各リソースは、*XtVaSetValues()*ではなく、*XrtGearTipsVaSetValues()*を呼び出してセットしてください。次のコードは、これらのリソースをセットしている例です。

```
XrtGearTipsVaSetValues(widget,
  XtVaTypedArg, XmNxrtGearForeground,
    XmRString, "red", strlen("red")+1,
  XtVaTypedArg, XmNxrtGearBackground,
    XmRString, "white", strlen("white")+1,
  XtVaTypedArg, XmNxrtGearFontList,
    XmRString, "-*-times-medium-r-*-*-*-120-*-*-*-*-iso8859-1",
    strlen("-*-times-medium-r-*-*-*-120-*-*-*-*-iso8859-1")+1,
  XmNxrtGearTipsBorderType, XRTGEAR_BORDER_ETCHED_IN,
  NULL);
```

11.6 表示コールバック

XmNxrtGearDisplayCallback リソースを使用して、ヘルプ・テキストを変更したり、ヘルプの表示を制止したりすることができます。リストの各コールバックは、ヘルプ・バルーンが表示される前と後に呼び出されます。

各コールバックには、次の構造体のポインタが渡されます。

```
typedef struct {
  int     reason; // Read-only
  XEvent  *event; // Read-only
  Widget  widget; // Read-only
  Boolean doit; // initially True
  XmString    text;
} XrtGearDisplayCallbackStruct;
```

reason は XRTGEAR_REASON_DISPLAY_BEGIN(ヘルプ・テキスト表示前)または XRTGEAR_REASON_DISPLAY_END(ヘルプ・テキスト表示後)です。コールバックが、テキストの表示前に呼び出された場合は、*text* を編集して表示テキストを変更したり[1]、*doit* を False にセットして表示を制止することができます。

[1] テキストを変更する前に、*XmStringFree()*を使用して既存のテキストを解放してください。

次のコードでは、SuppressHelpBalloons という global 変数がセットされると、ヘルプ･バルーンが表示されなくなります。

```
int SuppressHelpBalloons;

void
displayCB(Widget w, XtPointer client_data,
XrtGearDisplayCallbackStruct *call_data)
{
    if (call_data->reason == XRTGEAR_REASON_DISPLAY_BEGIN) {
      if (SuppressHelpBalloons) {
          call_data->doit = False;
      }
    }
}

int
main(int argc, char **argv)
{
    Widget w;
    ...
    XrtGearTipsAddCallback(w, XmNxrtGearDisplayCallback,
      displayCB, NULL);
    ...
}
```

*XtAddCallback()*ではなく、必ず *XrtGearTipsAddCallback()*を使用して、コールバックを XrtGearDisplayCallbackStruct リストに追加してください。同様にして、リストからコールバックを削除するには、*XrtGearTipsRemoveCallback()*を使用してください。

11.7 表示のタイミング

XmNxrtGearDelay は、ヘルプ･バルーンが表示されるまで、どれくらいの時間カーソルがそのウィジェット上に置かれていなければならないかをミリ秒で指定します。下の例では、この時間を 1000 ミリ秒(1 秒)にセットしています。

```
XrtGearTipsVaSetValues(widget,
  XmNxrtGearDelay, 1000,
  NULL);
```

デフォルトでは、XmNxrtGearDelay は 500 ミリ秒です。

help display group

通常、フォーカスが一つのウィジェットから別のウィジェットに移動する時、XmNxrtGearDelay で指定された時間が、新しいウィジェットのヘルプ･バルーンが表示される前に経過していなければなりません。複数のウィジェットを一つの *help display group* として指定することにより、これを変更することができます。エンドユーザが、同じグループ内で一つのウィジェットから別のウィジェットに移動した場合、XmNxrtGearDelay が無視され、新しいウィジェットのヘルプ･バルーンが直ちに表示されるようになります。

help display group を定義するには、任意の正の整数値を選び、グループ内の各ウィジェットの XmNxrtGearGroup リソースをその数にセットします。下の例では、1、2と番号付けされた二つの help display group を定義しています。

```
XmString str, str2, str3;
str = XmStringCreateLocalized(
    "This folder contains important memos.");
XrtGearTipsVaSetValues(memos,
    XmNxrtGearText, str,
    XmNxrtGearGroup, 1,
    NULL);
str2 = XmStringCreateLocalized("This string appears immediately.");
XrtGearTipsVaSetValues(faxes,
    XmNxrtGearText, str2,
    XmNxrtGearGroup, 1,
    NULL);
str3 = XmStringCreateLocalized("This string appears later.");
XrtGearTipsVaSetValues(todo,
    XmNxrtGearText, str3,
    XmNxrtGearGroup, 2,
    NULL);
```

エンドユーザが *memos* ウィジェットから *faxes* ウィジェットに移動した時、これらのウィジェットは同じ display group にあるので、*faxes* のヘルプ・テキストが直ちに表示されます。エンドユーザが *todo* に移動した場合は、(これは別のグループに所属するので)XmNxrtGearDelay で指定された時間が経過しないと、ヘルプ・テキストが表示されません。XmNxrtGearGroup をセットする時は、次の値を使用できます。

XRTGEAR_PARENT	この help display group は、ある親の全ての子ウィジェットで構成される。(但し、display group の番号が明示的に他にセットされている子ウィジェットは含まれない。)
XRTGEAR_SELF	このウィジェットは、どの help display group にも所属していないことを示す。

デフォルトでは、XmNxrtGearGroup は XRTGEAR_PARENT です。

11.8 widget tip を不可に設定

次のコードは、あるウィジェットで widget tip が使用されないようにしています。

```
XrtGearTipsVaSetValues(widget,
  XmNxrtGearDelay, MAXINT,
  XmNxrtGearGroup, XRTGEAR_SELF,
  NULL);
```

XmNxrtGearDelay を非常に大きい数にセットすると、(エンドユーザはカーソルをそんなに長くそのウィジェット上に停止させて置く可能性がほとんどないので)ヘルプ・バルーンを表示させないようにセットできます。XmNxrtGearGroup を XRTGEAR_SELF にセットすると、このウィジェットは help display group に所属しなくなるので、XmNxrtGearDelay の値に関係なく直ちにヘルプ・テキストが表示されます。

11.9 ヘルプ･テキストの表示と削除

ヘルプの表示

*XrtGearTipsDisplayHelp()*を使用すると、(widget tip が使用可能になっているかどうかに関わらず)指定されたテキストを直接表示させることができます。このテキストは、画面上のどこにでも表示することができます。

次の Enhanced Toggle の(状態が変化した時に呼び出される)コールバック手続きは、*XrtGearTipsDisplayHelp()*を使用して、ボタンが押された回数を表示します。

```
int click_count = 0;
XmString str;

void
stateCB(Widget w, XtPointer client_data,
   XrtGearStateChangedCallbackStruct *call_data)
{
   char buf[80];
   if (call_data->reason != XRTGEAR_REASON_STATE_BEGIN) {
      return;
   }
   if (click_count != 0) {
      XmStringFree(str);
   }
   click_count++;
   sprintf(buf, "number of clicks: %d", click_count);
   str = XmStringCreateSimple(buf);
   XrtGearTipsDisplayHelp(w, False, 10, 10, str);
}
```

*XrtGearTipsDisplayHelp()*は、次の引数を取ります。

- ヘルプ･テキストが表示されるウィジェット
- ヘルプ･テキストがルート･ウィンドウに相対するか(`True`)またはウィジェットに相対するか(`False`)を示す Boolean
- ヘルプ･テキストの位置の X 座標
- ヘルプ･テキストの位置の Y 座標
- 表示するテキスト(`NULL` であれば、以前に `XmNxrtGearText` によってセットされているテキストが表示されます。)

*XrtGearTipsDisplayHelp()*によって表示されるヘルプ･テキストは、ポインタがそのウィジェットを離れても自動的には削除されません。*XrtGearTipsRemoveHelp()*を使用して直接削除してください。

ヘルプの削除

*XrtGearTipsDisplayHelp()*によって表示されるヘルプ･バルーンを削除するには、*XrtGearTipsRemoveHelp()*を呼び出して下さい。

```
XrtGearTipsRemoveHelp();
```

この手続きは、現在表示されている全てのヘルプ･バルーンを削除します。

11.10 トランスレーションの定義

エンドユーザがキーを押した時(または指定されたXイベントを起こした時)、widget tipのヘルプ・バルーンがアクティブになるように指定することができます。これを実現するには、イベントが *XrtGearTipsActionDisplay()*アクション・ルーチンを呼び出すようにしてください。

表示されているヘルプ・バルーンを削除するイベントを指定することもできます。この場合は、イベントが*XrtGearTipsActionRemove()*を呼び出すようにしてください。

下の例では、エンドユーザが Shift とマウスの中央ボタンを押した時にヘルプ・バルーンが表示され、ボタンを離すとこれが消えるように指定しています。

```
char translations[] = "\
   Shift<Btn2Down>: tips_display_action()\n\
   Shift<Btn2Up>: tips_remove_action()";

XtActionsRec actions[] = {
   {"tips_display_action",
     (XtActionProc) XrtGearTipsActionDisplay},
   {"tips_remove_action",
     (XtActionProc) XrtGearTipsActionRemove},
};

Widget    toplevel;
Widget    *widget_list;
int       num_widgets;
int       i;

XtAppAddActions(XtWidgetToApplicationContext(toplevel),
   actions, XtNumber(actions));
XrtGearTipsGetWidgetList(toplevel, False, &widget_list,
   &num_widgets);
for (i = 0; i < num_widgets; i++) {
   XtOverrideTranslations(widget_list[i],
     XtParseTranslationTable(translations));
}
```

*XrtGearTipsGetWidgetList()*を呼び出すことにより、トランスレーションが、アプリケーションの各ウィジェットに指定できます。

11.11 リソースの一覧

Widget Tip のリソースの一覧です。リソース名の後は、データ型、デフォルト値、そしてリソースと共に使用される手続きです。**C** は、そのリソースが create(例: *XtCreateWidget()*)で定義でき、**S** は set(例; *XtSetValues()*)でき、**G** は get(例: *XtGetValues()*)で使用できることを意味します。

XmNxrtGearAlignment　unsigned char　XmALIGNMENT_CENTER　CSG

ウィンドウ内のテキストの位置を指定します。値は、`XmALIGNMENT_BEGINNING`、`XmALIGNMENT_CENTER`、`XmALIGNMENT_END` のどれかです。

XmNxrtGearBackground Pixel ***下記参照*** CSG

ヘルプ・バルーンの背景色を指定します。デフォルトは、Lemon Chiffon に相当する Pixel 色です。

XmNxrtGearDelay int 500 CSG

ヘルプが表示されるまで、カーソルがウィジェット上に停止していなければならない時間をミリ秒で指定します。同じ XmNxrtGearGroup 内のウィジェットにカーソルが移動した時は、この時間は 0 になります。

XmNxrtGearDisplayCallback XtCallbackList NULL CSG

ヘルプが表示される前と後に呼び出されるコールバックのリストを指定します。ルーチンは、*XrtGearTipsAddCallback()*、*XrtGearTipsRemoveCallback()*を使用して追加、削除してください。各ルーチンには、XrtGearDisplayCallbackStruct のポインタが渡されます。コールバックは、*XtAddCallback()*を使用してリストに追加してください。コールバックは次のように構成されます。

```
void
CallbackProcedure(
  Widget   widget,
  XtPointerclient_data,
  XtPointercall_data /* points to an XrtGearDisplayCallbackStruct */
)
```

XrtGearDisplayCallbackStruct は次のように定義されます。

```
typedef struct {
  int     reason;      // Read-only
  XEvent  *event;      // Read-only
  Widget  widget;      // Read-only
  Boolean doit;        // Initially True
  XmString    text;
} XrtGearDisplayCallbackStruct;
```

reason は XRTGEAR_REASON_DISPLAY_BEGIN または XRTGEAR_REASON_DISPLAY_END のどちらかです。event は、ヘルプが XrtGearTipsDisplayHelp()を使用して直接表示される場合は NULL です。widget は、ヘルプが表示されるウィジェットです。ヘルプを表示させない時は、reason が XRTGEAR_REASON_DISPLAY_BEGIN の時に doit を False にセットしてください。text は表示されるテキストで、reason が XRTGEAR_REASON_DISPLAY_BEGIN であれば変更できます。(既存のテキストは、XmStringFree()を使用して最初に解放してください。)

XmNxrtGearFontList XmFontList fixed CSG

テキストの表示に使用される fontlist を指定します。

XmNxrtGearForeground Pixel Black CSG

ヘルプ・バルーンの前景色を指定します。

XmNxrtGearGroup int XRTGEAR_PARENT CSG

ウィジェットに割り当てられる任意の番号を指定します。これは XmNxrtGearDelay を処理する時に使用されます。この番号は、必ず正の整数にしてください。XmNxrtGearGroup には次の値が定義できます。

XRTGEAR_PARENT	この help display group は、ある親の全ての子ウィジェットで構成される。(但し、display group の番号が明示的に他にセットされている子ウィジェットは含まれない。) デフォルト。

XRTGEAR_SELF　このウィジェットは、どの help display group にも所属していないことを示す。

XmNxrtGearShadowThickness　Dimension　1　CSG

ヘルプ・バルーンの影の幅を指定します。

XmNxrtGearText　XmString　NULL　CSG

表示されるテキストを指定します。NULL であれば、ヘルプ・バルーンは表示されません。

XmNxrtGearTipsBorderType　XrtGearBorderType　XRTGEAR_BORDER_PLAIN　CSG

ウィンドウの枠に使用されるスタイルを指定します。詳細は、Enhanced Label の解説を参照してください。

11.12 手続きと方法

Widget Tip の手続きと方法のアルファベット順のリストです。

XrtGearTipsActionDisplay()

ウィジェットにヘルプ・バルーンを表示するアクション・ルーチンです。

```
void
XrtGearTipsActionDisplay(
  Widget  widget,
  XEvent  *event
  )
```

widget はヘルプが表示されるウィジェットです。*event* は、このアクション・ルーチンを呼び出した X イベントです。

XrtGearTipsActionRemove()

ウィジェットに表示されているヘルプ・バルーンを削除するアクション・ルーチンです。

```
void
XrtGearTipsActionRemove(
  Widget  widget,
  XEvent  *event
  )
```

widget はヘルプが表示されるウィジェットです。*event* は、このアクション・ルーチンを呼び出した X イベントです。

XrtGearTipsAddCallback()

XmNxrtGearDisplayCallback コールバックを追加する時に使用されます。

```
void
XrtGearTipsAddCallback(
  Widget      widget,
  String      callback_name,
  XtCallbackProc  proc,
  XtPointer       tag
  )
```

widget は、コールバックが定義されるウィジェットです。*callback_name* は XmNxrtGearDisplayCallback です。*proc* は追加された新しいコールバック。*tag* は、コールバックが開始された時に手続きに渡される引数のポインタです。

XrtGearTipsAddCallbacks()

XmNxrtGearDisplayCallback コールバックのリストを追加します。

```
void
XrtGearTipsAddCallbacks(
  Widget      widget,
  String      callback_name,
  XtCallbackList  callbacks
  )
```

widget は、新しいコールバックが定義されるウィジェットです。*callback_name* は XmNxrtGearDisplayCallback です。*callback* は、追加されるコールバックのリストです。

XrtGearTipsDisplayHelp()

ウィジェットにヘルプを表示または、画面上の任意の場所にテキストを表示します。

```
void
XrtGearTipsDisplayHelp(
  Widget      widget,
  Boolean     position_root,
  Position        x, y,
  XmString        text
  )
```

widget は、ヘルプが表示されるウィジェットです。*position_root* が True であれば、*x* と *y* はルート・ウィンドウに相対するヘルプの位置です。*position_root* が False であれば、*x* と *y* はウィジェットに相対するヘルプの位置です。*text* は、そのウィジェットにテキストがなかった場合に書かれるテキストです。

XrtGearTipsEnable()

シェル内の全てのウィジェットに対してヘルプの表示を可または不可にセットします。シェルのどのウィジェットも渡すことができます。

```
void
XrtGearTipsEnable(
  Widget      shell,
  Boolean     enable
  )
```

shell はシェル・ウィジェット(またはシェル内の任意のウィジェット)。*enable* は True(ヘルプの使用可)または False(ヘルプの使用不可)です。

*XrtGearTipsEnable()*が呼び出された後に子がシェルに追加された場合は、Widget Tip はその子ウィジェットでは使用できません。(新しいウィジェットで Widget Tip を使用するには、*XrtGearTipsEnable()*をもう一度呼び出して下さい。)

XrtGearTipsGetValues()

ウィジェットの Widget Tip リソースを取得します。

```
void
XrtGearTipsGetValues(
  Widget      widget,
  ArgList     args,
  Cardinal        num_args
  )
```

この方法の引数は、*XtGetValues()*に渡される引数と同様です。

XrtGearTipsGetWidgetList()

あるシェルの子である全てのウィジェットのリストまたは、Widget Tip が使用可能になっているウィジェットのリストを取り出します。取得に成功すると `True` を返します。

```
Boolean
XrtGearTipsGetWidgetList(
  Widget      shell,
  Boolean     enabled_only,
  Widget      **list,
  int         *num_widgets
  )
```

shell はシェル･ウィジェットです。*list* は返されたウィジェットのリストで、これは使用後解放して下さい。*enabled_only* が `True` であれば、*list* は Widget Tip が使用可能になっているウィジェットのリストです。*enabled_only* が `False` であれば、*list* は全ての子ウィジェットのリストです。*num_widgets* は、*list* のエレメント数を返します。

XrtGearTipsRemoveAllCallbacks()

全ての `XmNxrtGearDisplayCallback` コールバックを削除する時に、（*XtRemoveAllCallbacks()*の代りに）使用してください。

```
void
XrtGearTipsRemoveAllCallbacks(
  Widget      widget,
  String      callback_name
  )
```

widget は、コールバックが削除されるウィジェットです。*callback_name* は `XmNxrtGearDisplayCallback` です。

XrtGearTipsRemoveCallback()

`XmNxrtGearDisplayCallback` を削除する時に、(*XtRemoveCallback()*の代りに)使用してください。

```
void
XrtGearTipsRemoveCallback(
  Widget      widget,
  String      callback_name,
  XtCallbackProc  proc,
  XtPointer       tag
  )
```

widget は、削除されるコールバックを持つウィジェットです。*callback_name* は `XmNxrtGearDisplayCallback` です。*proc* は削除されるコールバック手続きです。

tag は、登録されたコールバック手続きにマッチするクライアント・データのポインタです。コールバックは、*proc* と *tag* がリストのコールバックにマッチした場合にのみ削除されます。

XrtGearTipsRemoveCallbacks()

XmNxrtGearDisplayCallback コールバックのリストを削除します。

```
void
XrtGearTipsRemoveCallbacks(
  Widget      widget;
  String      callback_name,
  XtCallbackList  callbacks
  )
```

widget は、削除されるコールバックを持つウィジェットです。*callback_name* は、XmNxrtGearDisplayCallback です。*callbacks* は削除されるコールバックのリストです。

XrtGearTipsRemoveHelp()

現在表示されているヘルプを削除します。

```
void
XrtGearTipsRemoveHelp(void)
```

XrtGearTipsSetValues()

ウィジェットに、（XmNxrtGearDisplayCallback も含む）Widget Tip リソースをセットします。

```
void
XrtGearTipsSetValues(
  Widget      widget,
  ArgList     args,
  Cardinal        num_args
  )
```

この方法の引数は、*XtSetValues()*に渡される引数と同様です。

XrtGearTipsTrackingEvent()

*XmTrackingEvent()*と同様、このルーチンは、エンドユーザによってクリックされた、シェル内のウィジェットに対してヘルプを表示します。選択されたウィジェットが返されます。

```
Widget
XrtGearTipsTrackingEvent(
  Widget      shell,
  Cursor      cursor,
  Boolean     confine_to,
  XEvent      *event_return
  )
```

shell はシェル・ウィジェットです。*cursor* は、ポインタとして使用されるカーソルを指定します。*confine_to* は、ポインタが *shell* に制限されるかどうかを指定します。*event_return* は、返された ButtonRelease または KeyRelease イベントです。

XrtGearTipsVaGetValues()

Widget Tip リソースを取得します。

```
void
XrtGearTipsVaGetValues(
  Widget  widget,
  ...
  NULL
  )
```

この方法の引数は、*XtVaGetValues()*に渡される引数と同様です。

XrtGearTipsVaSetValues()

(`XmNxrtGearDisplayCallback` を含む)Widget Tip リソースをセットします。

```
void
XrtGearTipsVaSetValues(
  Widget  widget,
  ...
  NULL
  )
```

この方法の引数は、*XtVaSetValues()*に渡される引数と同様です。

12

Widget Snapshot

Widget Snapshot について ■ Pixmap Snapshot
Pimap の書き込みと印刷 ■ XImage スナップショット
リソースの一覧 ■ 手続きと方法

12.1 Widget Snapshot について

Widget Snapshot ユーティリティを使用して、Manager ウィジェットも含む、Motif のウィジェットは全て PostScript フォーマットにプリントできます。これらのウィジェットの Pixmap または XImage を取得することもできます。

12.2 Pixmap Snapshot

ウィジェットの pixmap スナップショットを取得するには、*XrtGearSnapGetWidgetPixmap()*を呼び出して下さい。

```
Pixmap p;
p = XrtGearSnapGetWidgetPixmap(widget);
```

この方法で、どんな Motif ウィジェットのイメージも取得することができます。

*XrtGearSnapGetWidgetPixmap()*によって返される pixmap は、使用後、*XFreePixmap()*を呼び出して必ず解放してください。

12.3 Pixmap の書き込みと印刷

pixmap を PostScript フォーマットに変換し、ファイルに書くには、*XrtGearSnapVaPrintPixmap()*を呼び出してください。

```
Pixmap p;
FILE *file;

file = fopen("mypixmap.ps", "w");
XrtGearSnapVaPrintPixmap(widget, p, file, NULL);
fclose(file);
```

このファイルは、他の PostScript ファイルと同様にプリントすることができます。

*XrtGearSnapVaPrintPixmap()*に渡されるウィジェットは、表示されているどのウィジェットでも構いません。*XrtGearSnapVaPrintPixmap()*は、書き込みが成功したかどうかを示す Boolean 値を返します。ファイルは既に開いている状態にしておいてください。この方法には、*XrtGearSnapGetWidgetPixmap()*を呼び出して作成された pixmap を始め、どんな pixmap でも渡すことができます。

プリントに関する属性

*XrtGearSnapVaPrintPixmap()*を呼び出す時は、一つまたは複数の属性と値のペアを指定することができます。次の例では、pixmap の二つのコピーがファイルに書き込まれるように指定しています。

```
XrtGearSnapVaPrintPixmap(widget, p, file,
  XRTGEAR_SNAP_NUM_COPIES, 2,
  NULL);
```

*XrtGearSnapVaPrintPixmap()*を呼び出す時は、次の属性とその値を指定できます。

XRTGEAR_SNAP_COLOR	色を指定。XRTGEAR_SNAP_AS_IS(デフォルト)、XRTGEAR_SNAP_MONO、XRTGEAR_SNAP_COLOR のどれかです。
XRTGEAR_SNAP_MARGIN_BOTTOM	下のマージンを指定。値は *double* 型の浮動小数点値。デフォルトは 0.25 ユニット(このユニットは XRTGEAR_SNAP_UNITS で指定)。
XRTGEAR_SNAP_MARGIN_LEFT	左のマージンを指定。値は、*double* 型の浮動小数点値。デフォルトは 0.25 ユニット。
XRTGEAR_SNAP_MARGIN_RIGHT	右のマージンを指定。値は、*double* 型の浮動小数点値。デフォルトは 0.25 ユニット。
XRTGEAR_SNAP_MARGIN_TOP	上のマージンを指定。値は、*double* 型の浮動小数点値。デフォルトは 0.25 ユニット。
XRTGEAR_SNAP_PAPERSIZE_WIDTH	用紙の幅を指定。値は、*double* 型の浮動小数点値。デフォルトは 8.5 ユニット。
XRTGEAR_SNAP_PAPERSIZE_HEIGHT	用紙の高さを指定します。値は、*double* 型の浮動小数点値。デフォルトは 11.0 ユニット。

XRTGEAR_SNAP_UNITS	マージンや用紙サイズの基準となる単位。XRTGEAR_SNAP_INCHES(デフォルト)またはXRTGEAR_SNAP_CMのどちらかです。
XRTGEAR_SNAP_NUM_COPIES	プリントするコピー数。デフォルトは1。
XRTGEAR_SNAP_ORIENTATION	プリントされるウィジェットの方向。XRTGEAR_SNAP_PORTRAIT(デフォルト)、XRTGEAR_SNAP_LANDSCAPE、XRTGEAR_SNAP_BESTのどれかです。
XRTGEAR_SNAP_SCALE	ページのスケーリングを指定。XRTGEAR_SNAP_FIT_TO_PAGE、XRTGEAR_SNAP_FIT_TO_PAGE_WIDTH、XRTGEAR_SNAP_FIT_TO_PAGE_HEIGHT、または72.0(デフォルト)です。
XRTGEAR_SNAP_SHOWPAGE	デフォルトはTrue。

*XrtGearSnapPrintPixmap()*では、属性と値のペアをArgListとして提供することができます。

```
XrtGearSnapArg *arglist;
int numargs;

XrtGearSnapPrintPixmap(widget, p, file, arglist,
    numargs);
```

12.4 XImage スナップショット

*XrtGearSnapGetWidgetImage()*は、ウィジェットのXImgeを取得します。

```
XImage *ximage;

ximage = XrtGearSnapGetWidgetImage(widget);
```

この方法で、どんなMotifウィジェットも取得できます。

*XrtGearSnapGetWidgetImage()*によって返されるXImageは、使用後、*XDestroyImage()*を呼び出して必ず解放して下さい。

PDSのutilsライブラリとリンクし、このXImageを*XrtGearSnapSaveImage()*を使用してxpmまたはxwdフォーマットで保管することができます。または、独自のイメージを*XrtGearSnapSaveImage()*に渡して、任意のフォーマットに保管することもできます。

12.5 リソースの一覧

Widget Snapshotユーティリティのリソースの一覧です。リソース名の後は、データ型、デフォルト値、そしてリソースと共に使用される手続きです。**C**は、そのリソースがcreate(例: *XtCreateWidget()*)で定義でき、**S**はset(例; *XtSetValues()*)でき、**G**はget(例: *XtGetValues()*)で使用できることを意味します。

XmNxrtGearSnapNoShapeClip	Boolean	False	CSG

*XrtGearSnapGetWidgetPixmap()*の shape clipping 機能をオフにします。(御使用のサーバが shape clipping を正しく処理できない場合は、このリソースをリソース・ファイルまたは.Xdefaults ファイルでセットしてください。)

12.6 手続きと方法

Widget Snapshot の手続きと方法のアルファベット順のリストです。

XrtGearSnapGetWidgetImage()

ウィジェットの XImage を取得します。ウィジェットが manage または realize されていない場合は NULL を返します。イメージは、使用後、*XDestroyImage()*で解放してください。

```
XImage *
XrtGearSnapGetWidgetImage(
  Widget  widget
  )
```

widget はイメージが取得されるウィジェットです。

XrtGearSnapGetWidgetPixmap()

ウィジェットの pixmap を取得します。ウィジェットが manage または realize されていない場合は XmUNSPECIFIED_PIXMAP を返します。pixmap は、使用後、*XFreePixmap()* で解放してください。

```
Pixmap
XrtGearSnapGetWidgetPixmap(
  Widget  widget
  )
```

widget はイメージが取得されるウィジェットです。

XrtGearSnapPrintPixmap()

pixmap の encapsulated PostScript をファイルに書きます。ファイルに書き込みできない場合は、False を返します。(ファイルは既に開かれている状態にしておいてください。)

```
Boolean
XrtGearSnapVaPrintPixmap(
  Widget      widget,
  Pixmap      pixmap,
  FILE        *fp,
  XrtGearSnapArg  *args,
  int     num_args,
  NULL
  )
```

widget は、表示されているウィジェット。*pixmap* は、pixmap スナップショット。*fp* は PostScript 出力が書かれるファイル。*args* は属性と値のペアのリスト(詳細は、*XrtGearSnapVaPrintPixmap()*の解説を参照してください。)*num_args* は *args* によって指定されるリストの引数の数。

XrtGearSnapSaveImage()

pixmap のスナップショットをファイルに保管します。(ファイルは既に開かれているとみなしているので)ファイルに書き込みができない場合は False が戻ります。

```
Boolean
XrtGearSnapSaveImage(
     Widget      widget,
     XtPointer   funcs,
     String      filename,
     String      msg
);
```

widget は表示されているウィジェット、funcs は XrtPDSImageFunc のポインタです。filename はイメージを保管するファイルの名前で、msg はエラー・メッセージ用のバッファです。msg が NULL であれば、エラー・メッセージがコンソールにプリントされます。

XrtGearSnapVaPrintPixmap()

pixmap の encapsulated PostScript をファイルに書きます。ファイルに書き込みできない場合は、False を返します。(ファイルは既に開かれている状態にしておいてください。)

```
Boolean
XrtGearSnapVaPrintPixmap(
  Widget  widget,
  Pixmap  pixmap,
  FILE    *fp,
  ...
  NULL
  )
```

この方法には、三つの引数とオプションで属性と値のペアのリストが渡されます。*widget* は表示されているウィジェット。*pixmap* は、pixmap スナップショット。*fp* は PostScript 出力が書かれるファイル。

次の属性が指定できます。(デフォルト値はイタリックで書かれています。)

XRTGEAR_SNAP_COLOR	XRTGEAR_SNAP_AS_IS,XRTGEAR_SNAP_MONO, XRTGEAR_SNAP_COLOR
XRTGEAR_SNAP_MARGIN_BOTTOM	0.25
XRTGEAR_SNAP_MARGIN_LEFT	0.25
XRTGEAR_SNAP_MARGIN_RIGHT	0.25
XRTGEAR_SNAP_MARGIN_TOP	0.25
XRTGEAR_SNAP_NUM_COPIES	1
XRTGEAR_SNAP_ORIENTATION	*XRTGEAR_SNAP_PORTRAIT*, XRTGEAR_SNAP_LANDSCAPE, XRTGEAR_SNAP_BEST
XRTGEAR_SNAP_PAPERSIZE_WIDTH	8.5
XRTGEAR_SNAP_PAPERSIZE_HEIGHT	11.0
XRTGEAR_SNAP_SCALE	XRTGEAR_SNAP_FIT_TO_PAGE, XRTGEAR_SNAP_FIT_TO_PAGE_WIDTH, XRTGEAR_SNAP_FIT_TO_PAGE_HEIGHT,*72.0*

XRTGEAR_SNAP_SHOWPAGE	True
XRTGEAR_SNAP_UNITS	*XRTGEAR_SNAP_INCHES*, XRTGEAR_SNAP_CM

XRTGEAR_SNAP_MARGIN_LEFT 、 XRTGEAR_SNAP_MARGIN_RIGHT 、 XRTGEAR_SNAP_MARGIN_TOP 、 XRTGEAR_SNAP_MARGIN_BOTTOM 、 XRTGEAR_SNAP_PAPERSIZE_WIDTH、XRTGEAR_SNAP_PAPERSIZE_HEIGHT の値は、必ず浮動小数点値を指定してください。また、これらの値には、XRTGEAR_SNAP_UNITS で指定される単位が使用されます。

13

Lists, Strings, Colors, Icons

XrtList オブジェクト ■ XrtString オブジェクト

アイコン ■ XrtColorEditor オブジェクト

手続きと方法

XRT/gear には、下記のように、ウィジェットに使用されるオブジェクトやデータ型が提供されています。

■ 同じサイズのアイテムのグループを記憶する XrtList オブジェクト

Strings、XmStrings、アイコンを記憶する XrtString オブジェクト

■ 使いやすいカラー・エディタを提供する XrtColorEditor オブジェクト

■ XRT/gear ウィジェットに表示されるアイコンを指定する XrtGearIcon データ型

この章では、XrtList と XrtString を操作する手続きや方法と、XrtColorEditor オブジェクトと XrtGearIcon 構造体を作成する手続きや方法を解説します。

13.1 XrtList オブジェクト

XrtList オブジェクトを使用して、同じサイズのアイテムのグループを効率的に記憶させることができます。アイテムは、定義されている方法を使用して、追加、削除、ソート、取得が行えます。

XrtList オブジェクトで指定されるリストは、0 ベースで、最初のエレメントのインデックスは 0、二番目のエレメントのインデックスは 1 というように続きます。

13.1.1 サンプル･コード

次のコードは、リストの作成、アイテムのリストへの追加、アイテムの取得を行うサンプルの手続きです。

```
void
CreateListOfIntegers()
{

  XrtGearObject list;
  int i, *ptr;

  list = XrtGearListCreate(sizeof(int));
  i = 34;
  XrtGearListAppend(list, &i);
  i = 46;
  XrtGearListAppend(list, &i);
  ptr = XrtGearListGetItem(list, 1);
  fprintf (stderr, "item is %d\n", *ptr);
}
```

この手続きは、整数のリストを作成し、整数 34 と 46 をリストに追加しています。次にリストから二番目のエレメントを取り出し、それを表示しています。

13.1.2 リストの作成と破棄

リストを作成するには、*XrtGearListCreate()*を呼び出して下さい。

```
list = XrtGearListCreate(sizeof(int));
```

この方法には、リストの各エレメントのサイズを表す、一つのパラメータが渡されます。(異なるタイプのエレメントは、同じサイズであれば一つのリストに記憶させることができます。)この方法は、作成されたリストを返します。

*XrtGearListCreate()*で作成されたリストを破棄するには、*XrtGearListDestroy()*を呼び出して下さい。このリストに使用されていた全てのメモリは解放されます。

```
XrtGearListDestroy(list);
```

リストの複製

既存のリストのコピーを作成するには、*XrtGearListDuplicate()*を使用してください。

```
newlist = XrtGearListDuplicate(oldlist);
```

この方法はリストとそのエレメントのコピーを作成し、そのコピーのポインタを返します。元のリストは変更されません。

13.1.3 リスト･アイテムの追加と削除

リスト･アイテムの追加

アイテムは、既存リストの次の四つの場所に追加できます。

- リストの終わり
- 既存のアイテムの直前
- 既存のアイテムの直後
- ソートされた順での該当個所

既存リストの最後にアイテムを追加するには、*XrtGearListAppend()*を呼び出してください。

```
index = XrtGearListAppend(list, newitem);
```

この方法は、追加の対象となるリストと、追加されるアイテムの二つのパラメータが必要です。このアイテムのサイズは、リストが *XrtGearListCreate()*で作成された時に指定されたサイズと同じでなければなりません。この方法は、新しいアイテムのインデックスを返します。(0 はリストの最初のアイテム、1 の場合は二番目のアイテム(以下同様)です。)

既存リストのアイテムの前に追加するには、*XrtGearListInsertBefore()*を呼び出して下さい。

```
newindex = XrtGearListInsertBefore(list, oldindex, newitem);
```

*XrtGearListInsertBefore()*は、追加の対象となるリスト、既存リスト･アイテムのインデックス(0 ベース)、追加されるアイテムの三つのパラメータが必要です。新しいアイテムのインデックスが返されます。

同様に、*XrtGearListInsetAfter()*は、新しいアイテムを既存のリスト･アイテムの直後に追加します。

```
newindex = XrtGearListInsertAfter(list, oldindex, newitem);
```

この方法のパラメータと戻り値は、*XrtGearListInsertBefore()*と同じです。

リストがソートされている場合は、*XrtGearListAdd()*を呼び出し、アイテムを該当する位置に追加してください。

```
index = XrtGearListAdd(list, newitem);
```

この方法は、追加の対象となるリストと追加されるアイテムの二つのパラメータが必要です。新しいアイテムのインデックスが返されます。(詳細は 13.1.6 を参照してください。)

リスト･アイテムの削除と破棄

XrtList オブジェクトは、リストからアイテムを削除するルーチンと、削除しそのアイテムを破棄するルーチンを定義しています。

リストからアイテムを削除するには、*XrtGearListRemove()*を呼び出して下さい。

```
retpos = XrtGearListRemove(list, position);
```

この方法には、リストと削除されるアイテムの位置が渡されます。位置は、この方法によって返されます。

リストから全てのアイテムを削除するには、*XrtGearListRemoveAll()*を呼び出して下さい。

```
XrtGearListRemoveAll(list);
```

上記の二つの方法は、メモリ上のアイテムは破棄しません。アイテムを削除後に破棄するには、*XrtGearListDelete()*を呼び出して下さい。

```
retpos = XrtGearListDestroy(list, position);
```

この方法には、*XrtGearListRemove()*と同様、リストと削除されるアイテムの位置が渡され、位置を返します。

リストの全てのアイテムを削除するには、*XrtGearListDestroyAll()*を呼び出してください。

```
XrtGearListDestroyAll();
```

特定なアイテムと関連するデータ構造体が定義されている場合、アイテムが削除された時その構造体も破棄されるようにするには、*XrtGearListSetDestroyProc()*を呼び出して破棄の手続きを定義して下さい。

```
XrtGearListSetDestroyProc(list, my_destroy_proc);
```

破棄の手続きは、*XrtGearListDestroy()*または *XrtGearListDestroyAll()*が呼ばれる度に呼び出されます。破棄の手続きは、次の型になるよう定義してください。

```
void my_destroy_proc (
  XrtGearObject   object,
  XtPointer       item
  );
```

object はリスト、*item* は破棄されるアイテムです。

XrtList オブジェクトでは、*XrtGearListDestroyObject()*を呼び出すことによってもリストのオブジェクトを破棄することができます。

```
XrtGearObject     list, *objptr;
XrtGearListDestroyObject(list, objptr);
```

リストと破棄されるオブジェクトのポインタの二つのパラメータが必要です。

オブジェクトではないアイテムを破棄するには、*XrtGearListDestroyXtPointer()*を呼び出して下さい。

```
XrtGearObject     list;
XtPointer     *item;
XrtGearListDestroyXtPointer(list, item);
```

この方法には、リストと破棄されるアイテムのポインタが渡されます。

13.1.4 リスト･アイテムの検索と取得

XrtList オブジェクトには、検索や取得の操作を行う幾つかの方法が定義されています。

- XrtGearObject がリストであるかどうかを判断する
- リスト内でアイテムを検索する
- インデックス付きエレメントまたはエレメントの配列全体を取り出す
- リストを前方向または後ろ方向に検索する

その他、リストのアイテム数やサイズを取得する方法もあります。

リストのチェック

*XrtGearListIsList()*は、XrtGearObject が XrtList であるかどうかを調べます。

```
Boolean       is_list;
XrtGearObject     object;
is_list = XrtGearListIsList(object);
```

*XrtGearListIsList()*は、オブジェクトが XrtList オブジェクトであれば `True`、そうでなければ `False` を返します。

アイテムの検索

アイテムを検索するには、次の方法を使用します。

- *XrtGearListFind()* - XrtList の比較手続きを使用して、指定したアイテムとマッチするリスト･アイテムを検索する
- *XrtGearListLookupItem()* - 対応するメモリを比較して、指定したアイテムとマッチするリスト･アイテムを検索する

どちらの方法にも、リストと比較されるアイテムが渡されます。マッチしたアイテムが検索された場合は、そのインデックスを返し、該当するアイテムがリストになければ XRTGEAR_LIST_ITEM_NOT_FOUND を返します。

XrtGearListFind() に使用される比較の手続きを指定するには、*XrtGearListSetCompareProc()*を呼び出してください。

```
XrtGearListSetCompareProc(list, my_compare_proc);
```

この比較の手続きは、次の型になるように定義してください。

```
int my_compare_proc (
  XtPointer   item1,
  XtPointer   item2
  );
```

item1 と *item2* は、比較される二つのアイテムです。比較の手続きは、結果によって次の値を返します。

- *item1* が *item2* より大きい場合は正の整数
- 二つのアイテムが同じであれば 0
- *item1* が *item2* より小さい場合は負の整数

現在使用されている比較手続きを取り出すには、*XrtGearListGetCompareProc()*を呼び出してください。

```
XrtGearListItemCompareProc myproc;
myproc = XrtGearListGetCompareProc(list);
```

返された手続きは、XrtGearListItemCompareProc 型で、次のように定義されます。

```
typedef int (*XrtGearListItemCompareProc) (
   XtPointer item1, XtPointer item2
);
```

この比較手続きは、ソートにも使用されます。尚、ソートに関しては、13.1.6 を参照してください。

インデックスによってアイテムを取り出す

位置(position)によってアイテムを取り出すには、*XrtGearListGetItem()*を呼び出してください。

```
XtPointer     item;
XrtGearObject     list;
int           index;
item = XrtGearListGetItem(list, index);
```

*XrtGearListGetItem()*には、リストと返されるアイテムの位置(最初のエレメントは 0、二番目のエレメントは 1 と、続きます)が渡されます。指定された位置にあるアイテムが返されます。位置が無効な場合は、NULL が戻ります。

特定な位置でアイテムをセットするには、*XrtGearListSetItem()*を呼び出してください。

```
XrtGearListSetItem(list, index, item);
```

この方法には、リスト、新しいアイテムのインデックスと新しいアイテムの三つのパラメータが渡されます。セットの操作が成功すれば True、指定されたインデックスが無効な場合は False が返されます。

List 配列の取得

リスト･エレメントの配列全体を取り出すには、*XrtGearListGetItems()*を呼び出してください。

```
XrtGearObject     list;
XtPointer     array;

array = XrtGearListGetItems(list);
```

リストが空な場合は、NULL が返ります。

前/後方向の検索

アイテムをリストの前方向に検索する場合は、*XrtGearListSearchForward()*を呼び出します。

```
int           pos;
XrtGearObject     list;
int           start_position;
XtPointer     item;

pos = XrtGearListSearchForward(list, start_position, item);
```

この方法には、リスト、検索を開始する位置、検索するアイテムの三つのパラメータが渡されます。アイテムがマッチした場合は、その位置が返され、それ以外の場合は XRTGEAR_LIST_ITEM_NOT_FOUND が返されます。

リストを後ろ方向に検索するには、*XrtGearlistSearchBackward()*を呼び出します。

```
pos = XrtGearListSearchBackward(list, start_position, item);
```

パラメータと戻り値は、*XrtGearListSearchForward()*の場合と同じです。

リスト情報の取得

リストのアイテム数を取得するには、*XrtGearListGetItemCount()*を呼び出します。

```
int           numitems;
XrtGearObject     list;

numitems = XrtGearListGetItemCount(list);
```

この方法には、引数としてリストが渡されます。

アイテムのサイズを取得するには、*XrtGearListGetItemSize()*を呼び出します。

```
int           itemsize;
XrtGearObject     list;

itemsize = XrtGearListGetItemSize(list);
```

13.1.5 リストの比較ルーチン

XrtList には、倍精度浮動小数点数、単精度浮動小数点数、整数、ストリングを比較するルーチンが定義され、異なる型を持つ値を比較することができます。ここで定義されているルーチンは、*XrtGearListSetCompareProc()*へ比較手続きとして渡すことにより、*XrtGearListFind()*で使用することができます。

二つの倍精度浮動小数点数を比較するには、*XrtGearCmpDoubles()*を呼び出します。

```
XtPointer     double1, double2;
int           retval;

retval = XrtGearCmpDoubles(double1, double2);
```

このルーチンは、最初の数が二番目の数より大きければ 1、同じであれば 0、小さければ-1 を返します。

*XrtGearCmpFloats()*は、二つの単精度浮動小数点数を比較します。

```
XtPointer     float1, float2;
int           retval;

retval = XrtGearCmpFloats(float1, float2);
```

戻り値は、*XrtGearCompDoubles()*と同じです。

二つの整数を比較するには、*XrtGearCompInts()*を呼び出します。

```
XtPointer     int1, int2;
int           retval;

retval = XrtGearCmpInts(int1, int2);
```

戻り値は、*XrtGearCmpDoubles()*と同じです。

二つのストリングを比較するには、*XrtGearCmpStrings()*を呼び出します。

```
XtPointer     string1, string2;
int           retval;

retval = XrtGearCmpStrings(string1, string2);
```

このルーチンは、最初のストリングが二番目のストリングよりアルファベット順で後にあれば 1、同じであれば 0、前にあれば－１を返します。

*XrtGearCmpStrings()*は、大文字小文字を別の文字として扱います。大文字小文字関係無く二つのストリングを比較するには、*XrtGearCompStringsIgnoreCase()*を呼び出してください。

```
XtPointer string1, string2;
int retval;

retval = XrtGearCmpStringsIgnoreCase(string1, string2);
```

戻り値は、*XrtGearCmpStrings()*と同じです。

13.1.6 リストのソート

アイテムをソートするには、*XrtGearListSort()*を呼び出します。

```
XrtGearListSort(list);
```

*XrtGearListSetCompareProc()*で定義される比較の手続きは、アイテムをソートする時にも使用できます。比較手続きが指定されていなければ、次のデフォルトのソートが適用されます。

- アイテムのサイズが 8 以下のサイズであれば、アイテムは byte 順にソートされる
- アイテムのサイズが 8 より大きく、比較手続きが定義されていなければ、ソートは行われない

尚、*XrtGearListSetCompareProc()*に関しては、13.1.4 を参照してください。

リストがソートされているかどうかを調べるには、*XrtGearListIsSorted()*を呼び出してください。

```
int       is_sorted;
is_sorted = XrtGearListIsSorted(list);
```

この方法は、リストがソートされていれば 0 でない値、ソートされていなければ 0 を返します。

エレメントの再配置

リスト・エレメントをリスト内の一つの場所から他の場所に移動するには、*XrtGearListMoveItemsBefore()*を呼び出してください。

```
Boolean       ok;
XrtGearObject      list;
int           destination;
int           source;
int           num_items;
source = 3;         /* fourth element is the first to be moved */
num_items = 3; /* move three elements */
destination = 0;   /* place before first element */
ok = XrtGearListMoveItemsBefore(list, destination, source,
   num_items);
```

*XrtGearListMoveItemsBefore()*には、次の四つのパラメータが必要です。

- 移動されるアイテムを持つリスト
- 移動先のエレメントのインデックス(0 は最初のエレメントを指す)、エレメントのブロックは、このエレメントの直前に配置される
- 移動されるブロックの最初のエレメントのインデックス
- 移動されるエレメントの数

*XrtGearListMoveItemsBefore()*は、移動が成功すれば `True`、エラーが起きれば `False` を返します。

13.1.7 ユーザが定義したデータ

XrtList オブジェクトでは、リストと関連するユーザ・データを定義することができます。データを定義するには、ユーザ・データのポインタを *XrtGearListSetUserData()*に渡してください。

```
XrtGearObject     list;
XtPointer     user_data;
XrtGearListSetUserData(list, user_data);
```

ユーザ・データによって使用されるメモリは必ず解放してください。

リストに関連する現在のユーザ・データを取り出すには、*XrtGearListGetUserData()*を呼び出してください。

```
user_data = XrtGearListGetUserData(list);
```

リストにユーザ・データが定義されていなければ、*XrtGearListGetUserData()*は `NULL` を返します。

13.2 XrtString オブジェクト

XrtString オブジェクトでは、一つのデータ・オブジェクトを使用して文字列や XmStrings、アイコンを記憶することができます。

13.2.1 ストリングの作成と破棄

XrtString オブジェクトを作成するには、*XrtGearStringCreate()*を使用してください。

```
XrtGearStringObjectClass  class;
XtPointer         stringval;
XrtGearObject         newstring;
newstring = XrtGearStringCreate(class, stringval);
```

作成されたストリング(`XrtGearObject` 型)が返されます。

クラス・パラメータは、次のどれかです。

- `XrtGearCharStringObjectClass`(ストリングを作成)
- `XrtGearXmStringObjectClass`(XmString を作成)
- `XrtGearIconStringObjectClass`(アイコンを作成)

次の例では、`XrtGearCharStringObjectClass` の XrtString オブジェクトを作成しています。

```
XrtGearStringCreate(XrtGearCharStringObjectClass, “mystring”);
```

XrtGearStringCreateCharString() 、 *XrtGearStringCreateXmString()* 、 *XrtGearStringCreateIconString()*を使用して、どんなクラスの XrtStrings も簡単に作成できます。

```
XrtGearObject     charclass, xmclass;
XmString      xmstring;
XrtGearIcon       icon;

charclass = XrtGearStringCreateCharString("mystring");
xmclass = XrtGearStringCreateXmString(xmstring);
iconclass = XrtGearStringCreateIconString(&icon);
```

尚、XrtGearIcons の作成法は、13.4 を参照してください。

XrtString オブジェクトを破棄するには、*XrtGearStringDestroy()*を呼び出します。

```
XrtGearObject     mystring;

XrtGearStringDestroy(mystring);
```

このストリングに割り当てられているメモリは、ストリングが破棄される時に解放されます。

ストリングのコピー

既存のストリングをコピーするには、*XmNxrtGearStringCopy()*を呼び出します。

```
XrtGearObject     original,mycopy;

mycopy = XmNxrtGearStringCopy(original);
```

13.2.2 ストリングの型

XrtGearObject 型のオブジェクトが XrtString であるかどうか、ストリングであればそのクラスを調べるには、*XrtGearStringIsCharString()*、*XrtGearStringIsXmString()* または *XrtGearStringIsIconString()*を呼び出してください。

```
Boolean       is_string;
XrtGearObject     object;

is_string = XrtGearStringIsCharString(object);
is_string = XrtGearStringIsXmString(object);
is_string = XrtGearStringIsIconString(object);
```

これらの方法は、次のような値を返します。

- *XrtGearStringIsCharString()* は、渡されたオブジェクトが `XrtGearCharStringObjectClass` クラスの XrtString であれば `True` を返す
- *XrtGearStringIsXmString()*は、オブジェクトが `XrtGearXmStringObjectClass` クラスの XrtString であれば、`True` を返す
- *XrtGearStringIsIconString()*は、オブジェクトが `XrtGearIconStringObjectClass` クラスの XrtString であれば `True` を返す

XrtGearObject が XrtList オブジェクトであるかどうかを調べるには、*XrtGearListIsList()* を使用してください。

13.2.3 ストリング値の操作

XrtString オブジェクトの値を取り出すには、*XrtGearStringGetValue()*を呼び出してください。

```
XrtGearObject object;
XtPointer value;

value = XrtGearStringGetValue(object);
```

この方法は、次のような値を返します。

- オブジェクトが XrtGearCharStringObjectClass クラスであれば、戻り値は String 型。
- オブジェクトが XrtGearXmStringObjectClass クラスであれば、戻り値は XmString 型。
- オブジェクトが XrtGearIconStringObjectClass クラスであれば、戻り値は XrtGearIcon のポインタ。

XrtString オブジェクトの現在値をセットするには、*XrtGearStringSetValue()*を呼び出してください。

```
XrtGearStringSetValue(object, value);
```

ストリング値で XrtString をセットする場合は、値が正しい型であることを確認してください。XrtString が XrtGearCharStringObjectClass クラスであれば、新しい値は String 型です。XrtString が XrtGearXmStringObjectClass クラスであれば、新しい値は XmString 値です。次のコードは、その例です。

```
if (XrtGearStringIsCharString(object)) {
   XrtGearStringSetValue(object, "mynewvalue");
} else if (XrtGearStringIsXmString(object)) {
   XrtGearStringSetValue(object,
     XmStringCreateSimple("mynewvalue"));
}
```

13.2.4 二つのストリングの比較

二つのストリングの値を比較するには、*XrtGearStringCompare()*を呼び出してください。

```
int           result;
XrtGearObject     object1, object2;

result = XrtGearStringCompare(object1, object2);
```

戻り値は、次のどれかです。

- 最初の値が二番目の値よりアルファベット順で先にあれば 0 より大きい整数
- 二つの値が同じであれば 0
- 最初の値が二番目の値よりアルファベット順で後にあれば負の整数

各 XrtStrings は、XrtGearCharStringObjectClass または XrtGearXmStringObjectClass のどちらかのストリング･クラスです。異なるストリング･クラスのオブジェクトを比較することができます。

13.3 XrtColorEditor オブジェクト

XrtColorEditor オブジェクトは、アプリケーションに簡単に挿入できるカラー・エディタです。ユーザは、色を名前、16 進数、RGB、HSB(Hue[色調]、Saturation[彩度]、Brightness[明るさ])で選択することができます。このオブジェクトは、ユーザ・インタラクションを管理し、選択された色のプレビューや割り当てを行います。

13.3.1 サンプル・コード

1 色の作成

```
Widget parent;
XrtGearObject editor;
editor = XrtGearColorEditorCreate(parent, "Test Editor");
```

2 エディタのテスト

```
XrtGearObject editor;
if (XrtGearIsColorEditor(editor)) {
     printf("\"editor\" *is* an XrtColorEditor\n");
}
else {
     printf("\"editor\" is *not* an XrtColorEditor\n");
}
```

3 エディタの破棄

```
XrtGearObject editor:
XrtGearColorEditorDestroy(editor);
```

4 red、green、blue の値の取得

```
XrtGearObject editor;
unsigned int red, green, blue;
XrtGearColorEditorGetRGBValues(editor, &red, &green, &blue);
printf(Get RGB: red = %d, green = %d, blue = %d\n · red, green,
blue);
```

5 HSB 値による色の設定

```
XrtGearColorEditorSetHSBValues(editor, 300, 80, 80);
```

XrtColorEditor を操作する方法のリストは、13.5.3 にあります

13.4 アイコン

XrtGearIcon データ型を使用して、アイコンの pixmap と pixmap マスクを一つの構造体に記憶させることができます。これは、次のように定義されます。

```
typedef struct {
  Pixmap  icon;
  Pixmap  icon_mask;
} XrtGearIcon;
```

次のルーチンは、XrtGearIcon 構造体を作成、操作します。

- *XrtGearIconLoadFromXpmData()*は、XPM データ構造体から XrtGearIcon 構造体を読みます。
- *XrtGearIconLoadFromXpmFile()*は、XPM フォーマットのファイルから XrtGearIcon 構造体を読みます。

■ *XrtGearIconGetSize()*は、指定されたアイコンの幅と高さを返します。

これらの方法は、13.5.4 に解説されています。尚、XPM フォーマットに関する詳細は、XPM 関連の資料を参照してください。

13.5 手続きと方法

13.5.1 XrtList オブジェクトの方法

XrtGearCmpDoubles()

二つの倍精度浮動小数点数を比較します。一番めの値が二番めより大きい場合は 1、同じ場合は 0、小さい場合は-1 を返します。この手続きは、*XrtGearListSetCompareProc()* に渡すことができます。

```
int
XrtGearCmpDoubles(
  XtPointer       double1,
  XtPointer       double2
  )
```

double1 と *double2* は、比較される二つの数です。

XrtGearCmpFloats()

二つの単精度浮動小数点数を比較します。一番めの値が二番めより大きい場合は 1、同じ場合は 0、小さい場合は-1 を返します。この手続きは、*XrtGearListSetCompareProc()* に渡すことができます。

```
int
XrtGearCmpFloats(
  XtPointer       float1,
  XtPointer       float2
  )
```

float1 と *float2* は、比較される二つの数です。

XrtGearCmpInts()

二つの整数を比較します。一番めの値が二番めより大きい場合は 1、同じ場合は 0、小さい場合は-1 を返します。この手続きは、*XrtGearListSetCompareProc()*に渡すことができます。

```
int
XrtGearCmpInts(
  XtPointer       int1,
  XtPointer       int2
  )
```

int1 と *int2* は、比較される二つの数です。

XrtGearCmpStrings()

二つのストリングを比較します。一番めの値が二番めよりアルファベット順で先に来る場合は 1、同じ場合は 0、後に来る場合は-1 を返します。この手続きは、*XrtGearListSetCompareProc()*に渡すことができます。

```
int
XrtGearCmpStrings(
  XtPointer       string1,
  XtPointer       string2
  )
```

string1 と *string2* は、比較される二つのストリングです。

XrtGearCmpStringIgnoreCase()

二つのストリングを大文字小文字関係なく比較します。一番めの値が二番めよりアルファベット順で先に来る場合は 1、同じ場合は 0、後に来る場合は-1 を返します。この手続きは、*XrtGearListSetCompareProc()*に渡すことができます。

```
int
XrtGearCmpStringsIgnoreCase(
  XtPointer       string1,
  XtPointer       string2
  )
```

string1 と *string2* は、比較される二つのストリングです。

XrtGearListAdd()

アイテムをリストの(ソートされた)該当位置に追加します。ソートの順序は、*XrtGearListGetCompareProc()*によって返される手続きによって定義されます。新しいアイテムのインデックスまたは、XRTGEAR_LIST_INVALID_POSITION(失敗時)が返されます。

```
int
XrtGearListAdd(
  XrtGearObject   list,
  XtPointer       new
  )
```

list はリスト・オブジェクト。*new* は新しいアイテムのポインタで、これは*XrtGearListGetItemSize()*の戻り値と同じサイズでなければなりません。

XrtGearListAppend()

アイテムをリストの最後に追加します。新しいアイテムのインデックスまたは、XRTGEAR_LIST_INVALID_POSITION(失敗時)を返します。

```
int
XrtGearListAppend(
  XrtGearObject   list,
  XtPointer       new
  )
```

list はリスト・オブジェクト。*new* は新しいアイテムのポインタで、これは*XrtGearListGetItemSize()*の戻り値と同じサイズでなければなりません。

XrtGearListCreate()

0 の長さのリストを作成し、それを返します。

```
XrtGearObject
XrtGearListCreate(
  size_t      item_size
  )
```

item_size は、アイテムの予想されるサイズです。

XrtGearListDelete()

指定された位置のアイテムを削除します。削除されたアイテムの位置を返します。

```
int
XrtGearListDelete(
  XrtGearObject   list,
  int         position
  )
```

list はリスト・オブジェクト。*position* は、削除されるアイテムの位置です。

(*XrtGearSetDestroyProc()*によって)破棄の手続きが登録されている場合は、アイテムが削除される時に、この手続きが呼び出されます。(破棄の手続きを呼び出さずにアイテムを削除するには、*XrtGearListRemove()*を使用してください。)

XrtGearListDeleteAll()

リストから全てのアイテムを削除します。

```
void
XrtGearListDeleteAll(
  XrtGearObject   list,
  )
```

list は、アイテムが削除されるリスト・オブジェクトです。

(*XrtGearSetDestroyProc()*によって)破棄の手続きが登録されている場合は、アイテムが削除される時に、この手続きが呼び出されます。(破棄の手続きを呼び出さずにアイテムを削除するには、*XrtGearListRemoveAll()*を使用してください。)

XrtGearListDestroy()

指定されたリストを破棄し、割り当てられていたメモリを解放します。

```
void
XrtGearListDestroy(
  XrtGearObject   list
  )
```

list は、破棄されるリスト・オブジェクトです。

XrtGearListDestroyObject()

指定されたリストのオブジェクトを破棄します。

```
void
XrtGearListDestroyObject(
  XrtGearObject   list,
  XrtGearObject   *object
  )
```

list は、破棄されるオブジェクトを持つリスト・オブジェクト。*object* は破棄されるオブジェクトです。

XrtGearListDestroyXtPointer()

指定されたリストの、オブジェクトではないアイテムを破棄します。

```
void
XrtGearListDestroyXtPointer(
  XrtGearObject   list,
  XtPointer       *item
  )
```

list は、破棄されるアイテムを持つリスト・オブジェクト。*item* は破棄されるアイテムです。

XrtGearListDuplicate()

リストのコピーを作成し、そのコピーを返します。(注意: 新しいリストのアイテムの破棄手続きは、NULL にセットされます。これは、*XrtGearListSetDestroyProc()*を使用してセットできます。)

```
XrtGearObject
XrtGearListDuplicate(
  XrtGearObject   list
  )
```

list は、コピーされるリスト・オブジェクトです。

XrtGearListFind()

(*XrtGearListGetCompareProc()*によって返される手続きを使用して)指定されたアイテムにマッチするリスト・アイテムを探します。アイテムがみつかった場合はそのインデックス、みつからなかった場合は、XRTGEAR_LIST_ITEM_NOT_FOUND を返します。

```
int
XrtGearListFind(
  XrtGearObject   list,
  XtPointer       item
  )
```

list は、検索されるリスト・オブジェクト。*item* はマッチさせるアイテムのポインタ。リストがソートされる場合(XmNxrtGearSorted が True)は、アイテムの検索にバイナリ検索が使用されます。リストに重複したアイテムがある場合、返されたアイテムが重複するアイテムの内、先に来る方です。

XrtGearListGetCompareProc()

オブジェクトが検索やソートに使用する手続きを取り出します。

```
XrtGearListItemCompareProc
XrtGearListGetCompareProc(
  XrtGearObject   list
  )
```

list は、比較手続きが返されるリスト・オブジェクトです。

XrtGearListGetItem()

指定されたインデックスのアイテムのポインタを返します。インデックスが無効な場合は NULL を返します。返されたアイテムは、dereference してください。

```
XtPointer
XrtGearListGetItem(
  XrtGearObject   list,
  int         index
  )
```

list はリスト・オブジェクト。*index* は、指定されたアイテムのインデックスです(0 はリストの最初のエレメント、1 は二番目のエレメントと続きます。)

XrtGearListGetItemCount()

リストに記憶されているアイテムの数を取り出します。

```
int
XrtGearListGetItemCount(
  XrtGearObject   list
  )
```

list は、カウントされるアイテムを持つリスト・オブジェクトです。

XrtGearListGetItems()

アイテムの配列のポインタを返します。リストが空の場合は、NULL を返します。

```
XtPointer
XrtGearListGetItems(
  XrtGearObject   list
  )
```

list は、リスト・オブジェクトです。

XrtGearListGetItemSize()

各アイテムのサイズを取り出します。

```
int
XrtGearListGetItemSize(
  XrtGearObject   list
  )
```

list は、サイズが返されるアイテムを持つリスト・オブジェクトです。

XrtGearListGetUserData()

XrtList オブジェクトに関連しているユーザ・データを取得し、それを返します。

```
XtPointer
XrtGearListGetUserData(
  XrtGearObject   list
  )
```

list は、取得されるユーザ・データを持つリスト・オブジェクトです。

XrtGearListInsertAfter()

新しいアイテムをリストの指定位置の後に挿入します。新しいアイテムのインデックスまたは、XRTGEAR_LIST_INVALID_POSITION(失敗時)を返します。

```
int
XrtGearListInsertAfter(
  XrtGearObject   list,
  int         index,
  XtPointer       new
  )
```

list はリスト・オブジェクト。*index* は、新しいアイテムが挿入される位置の前のインデックス。*new* は新しいアイテムのポインタ。*index* がリストにあるエレメント数より大きい場合は、*new* は、リストの最後に追加されます。

尚、*new* は、*XrtGearListGetItemSize()*の戻り値と同じサイズでなければなりません。

XrtGearListInsertBefore()

新しいアイテムをリストの指定位置の前に挿入します。新しいアイテムのインデックスまたは、XRTGEAR_LIST_INVALID_POSITION(失敗時)を返します。

```
int
XrtGearListInsertBefore(
  XrtGearObject   list,
  int         index,
  XtPointer       new
  )
```

list はリスト・オブジェクト。*index* は、新しいアイテムが挿入される位置の後のインデックス。*new* は新しいアイテムのポインタ。*index* がリストにあるエレメント数より大きい場合は、*new* は、リストの最後に追加されます。

尚、*new* は、*XrtGearListGetItemSize()*の戻り値と同じサイズでなければなりません。

XrtGearListIsList()

XrtGearObject が XrtList であれば True を返します。

```
Boolean
XrtGearListIsList(
  XrtGearObject   object
  )
```

object は、調べられるオブジェクトです。

XrtGearListIsSorted()

リストが現在ソートされているかどうか示します。ソートされている場合は、0 以外の値を返します。

```
int
XrtGearListIsSorted(
  XrtGearObject   list
  )
```

list は、調べられるリスト・オブジェクトです。

XrtGearListLookupItem()

指定されたリストで、アイテムの場所を探します。このルーチンは、メモリを比較し、*XrtGearListGetCompareProc()*によって返された手続きを使用しません。マッチしたアイテムの位置または、XRTGEAR_LIST_ITEM_NOT_FOUND(失敗時)を返します。

```
int
XrtGearListLookupItem(
  XrtGearObject   list,
  XtPointer       item
  )
```

list は検索されるリスト。*item* は、位置が調べられるアイテムのポインタです。

XrtGearListMoveItemsAfter()

指定された数のアイテムを移動先の後に移動します。

```
Boolean
XrtGearListMoveItemsAfter(
  XrtGearObject   list,
  int         destination,
  int         source,
  int         num_items_to_move
  )
```

list はリスト・オブジェクト。*destination* は移動先で、移動されたアイテムはこの後に置かれます。*source* は、移動されるアイテムの内の最初のアイテムの位置。*num_items_to_move* は、移動されるアイテムの数です。

XrtGearListMoveItemsBefore()

指定された数のアイテムを移動先の前に移動します。

```
Boolean
XrtGearListMoveItemsBefore(
  XrtGearObject   list,
  int         destination,
  int         source,
  int         num_items_to_move
  )
```

list はリスト・オブジェクト。*destination* は移動先で、移動されたアイテムはこの前に置かれます。*source* は、移動されるアイテムの内の最初のアイテムの位置。*num_items_to_move* は、移動されるアイテムの数です。

XrtGearListRemove()

リストから指定された位置のアイテムを削除します。削除されたアイテムの位置を返します。

```
int
XrtGearListRemove(
  XrtGearObject   list,
  int         position
  )
```

list はリスト・オブジェクト。*position* は、削除されるアイテムの位置です。

(*XrtGearSetDestroyProc()*で登録される)破棄の手続きは、アイテムが削除される時に呼び出されません。(アイテムを削除し、破棄の手続きを呼び出すには、*XrtGearListDelete()*を呼び出してください。)

XrtGearListRemoveAll()

リストから全てのアイテムを削除します。

```
void
XrtGearListRemoveAll(
  XrtGearObject   list
  )
```

list は、削除されるアイテムを持つリスト・オブジェクトです。

(*XrtGearSetDestroyProc()*で登録される)破棄の手続きは、アイテムが削除される時に呼び出されません。(各アイテムを削除し、その度に破棄の手続きを呼び出すには、*XrtGearListDeleteAll()*を呼び出してください。)

XrtGearListSearchForward()

指定されたアイテムにマッチするアイテムを、リストの指定された位置から前方向に検索します。(*XrtGearListGetCompareProc()*によって返される手続きが、比較を行います。)マッチしたアイテムの位置または、XRTGEAR_LIST_ITEM_NOT_FOUND(失敗時)を返します。

```
int
XrtGearListSearchForward(
  XrtGearObject   list,
  int         start_position,
  XtPointer       item
  )
```

list は検索されるリスト。*start_position* は、最初に比較されるアイテムの位置です。*item* は、マッチさせるアイテムのポインタです。

XrtGearListSearchReverse()

指定されたアイテムにマッチするアイテムを、リストの指定された位置から後ろ方向に検索します。(*XrtGearListGetCompareProc()*によって返される手続きが、比較を行います。)マッチしたアイテムの位置または、XRTGEAR_LIST_ITEM_NOT_FOUND(失敗時)を返します。

```
int
XrtGearListSearchReverse(
  XrtGearObject   list,
  int         start_position,
  XtPointer       item
  )
```

list は検索されるリスト。*start_position* は、最初に比較されるアイテムの位置です。*item* は、マッチさせるアイテムのポインタです。

XrtGearListSetCompareProc()

リスト･オブジェクトがソートに使用する手続きを指定します。

```
void
XrtGearListSetCompareProc(
  XrtGearObject        list,
  XrtGearListItemCompareProc  compare_proc
  )
```

list は、比較手続きがセットされるリスト･オブジェクト。*compare_proc* は、呼び出される手続きです。

比較の手続きは、次の型になるように定義してください。

```
int (*XrtGearListItemCompareProc) (
  XtPointer        item1,
  XtPointer        item2
  );
```

item1 と *item2* は、比較される二つのアイテムです。比較の手続きは、次の値を返します。

- *item1* が *item2* より大きい場合は正の整数
- 二つのアイテムが同じ場合は 0
- *item1* が *item2* より小さい場合は負の整数

XrtGearListSetDestroyProc()

リスト･オブジェクトがアイテムを破棄する場合に使用する手続きを指定します。(リスト･アイテムにメモリ等が特別な方法で割り当てられている場合)

```
void
XrtGearListSetDestroyProc(
  XrtGearObject        list,
  XrtGearListItemDestroyProc  destroy_proc
  )
```

list は、アイテムを破棄する手続きがセットされるリスト･オブジェクト。*destroy_proc* は呼び出される手続きです。

アイテムを破棄する手続きは、次の型になるように定義してください。

```
void (*XrtGearListItemDestroyProc) (
  XrtGearObject   object,
  XtPointer        item,
  XtPointer        user_data
  );
```

object は破棄されるアイテムを持つリスト･オブジェクト。*item* は、破棄されるアイテム。*user_data* は(*XrtGearListSetUserData()*でセットされる)ユーザ･データです。

XrtGearListSetItem()

アイテムをセットします。指定されたインデックスが無効な場合は、`False` を返します。

```
Boolean
XrtGearListSetItem(
  XrtGearObject   list,
  int          index,
  XtPointer        item
  )
```

list はリスト・オブジェクト。*index* はセットされるアイテムの位置。*item* はセットされるアイテムのポインタです。

XrtGearListSetUserData()

XrtList オブジェクトに関連するユーザ・データをセットします。

```
void
XrtGearListSetUserData(
  XrtGearObject   list,
  XtPointer       user_data
  )
```

list はリスト・オブジェクト。*user_data* はユーザ・データです。

XrtGearListSort()

アイテムをソートします。リストが既にソートされている場合は、何も変化しません。

```
void
XrtGearListSort(
  XrtGearObject   list
  )
```

list はソートされるリストです。

13.5.2 XrtString オブジェクトの方法

XrtGearCvtStringToXmString()

String を XmString へ変換します。変換された XmString を返します。

```
XmString
XrtGearCvtStringToXmString(
  const String        string
  )
```

string は、変換される String です。

XrtGearCvtXmStringToString()

XmString を String へ変換します。変換された String を返します。

```
XmString
XrtGearCvtXmStringToString(
  const XmString  string
  )
```

string は変換される XmString です。

XrtGearStringCompare()

*strcmp()*を使用して二つのストリングを比較します。

```
int
XrtGearStringCompare(
  XrtGearObject   string1,
  XrtGearObject   string2
  )
```

string1 と *string2* は、比較される二つのストリングです。ストリング・オブジェクトが XmString 型であれば、String に変換されます。

XrtGearStringCopy()

ストリング･オブジェクトをコピーし、そのコピーのポインタを返します。

```
XrtGearObject
XrtGearStringCopy(
  XrtGearObject   string_object
  )
```

string_object はコピーされるストリング･オブジェクトです。

XrtGearStringCreate()

XrtString オブジェクトを作成し、そのポインタを返します。

```
XrtGearObject
XrtGearStringCreate(
  XrtGearObjectClass  class,
  XtPointer       value
  )
```

class はウィジェット･クラスで、`XrtGearCharStringObjectClass` または `XrtGearXmStringObjectClass` のどちらかです。*value* は作成されたストリングのテキストです。

XrtGearStringCreateCharString()

指定された文字ストリングを持つ、`XrtGearCharStringObjectClass` クラスの XrtString オブジェクトを作成し、そのポインタを返します。

```
XrtGearObject
XrtGearStringCreateCharString(
  String      char_string
  )
```

char_string は、オブジェクトに含まれる文字ストリングです。この手続きを呼び出すことは、次の場合と同様です。

```
XrtGearStringCreate(XrtGearCharStringObjectClass,
(XtPointer) char_string);
```

XrtGearStringCreateIconString()

指定された XrtGearIcon 構造体を持つ、`XrtGearIconStringObjectClass` クラスの XrtString オブジェクトを作成し、そのポインタを返します。

```
XrtGearObject
XrtGearStringCreateIconString(
  XrtGearIcon     *iconptr
  )
```

iconptr は、オブジェクトに含まれるアイコンのポインタです。この手続きを呼び出すことは、次の場合と同様です。

```
XrtGearStringCreate(XrtGearIconStringObjectClass,
(XtPointer) iconptr);
```

XrtGearStringCreateXmString()

指定された XmString を持つ、XrtGearXmStringObjectClass クラスの XmString オブジェクトを作成し、そのポインタを返します。

```
XrtGearObject
XrtGearStringCreateXmString(
  XmString        xm_string
  )
```

xm_string は、オブジェクトに含まれる XmString です。この手続きを呼び出すことは、次の場合と同様です。

```
XrtGearStringCreate(XrtGearXmStringObjectClass,
(XtPointer) char_string);
```

XrtGearStringDestroy()

渡されたストリング・オブジェクトを破棄し、割り当てられていたメモリを解放します。

```
void
XrtGearStringDestroy(
  XrtGearObject   string_object
  )
```

string_object は、破棄されるストリング・オブジェクトです。

XrtGearStringGetValue()

ストリング・オブジェクトの値を返します。オブジェクトのクラスによって、XmString または String のどちらかになります。

```
XtPointer
XrtGearStringGetValue(
  XrtGearObject   string_object
  )
```

string_object は、値が取り出されるストリング・オブジェクトです。

XrtGearStringIsCharString()

指定されたオブジェクトが XrtGearCharStringObjectClass クラスであれば、True を返します。

```
Boolean
XrtGearStringIsCharString(
  XrtGearObject string_object
)
```

string_object は、クラスが調べられるオブジェクトです。

XrtGearStringIsIconString()

指定されたオブジェクトが XrtGearIconStringObjectClass クラスであれば、True を返します。

```
Boolean
XrtGearStringIsIconString(
  XrtGearObject   string_object
  )
```

string_object は、クラスが調べられるオブジェクトです。

XrtGearStringIsXmString()

指定されたオブジェクトが XrtGearXmStringObjectClass クラスであれば、True を返します。

```
Boolean
XrtGearStringIsXmString(
  XrtGearObject   string_object
  )
```

string_object は、クラスが調べられるオブジェクトです。

XrtGearStringSetValue()

オブジェクトの値をセットします。

```
void
XrtGearStringSetValue(
  XrtGearObject   string_object,
  XtPointer       value
  )
```

string_object は値が変更されるオブジェクト、*value* は新しい値です。*value* は、*string_object* の前の値と同じ型でなければなりません。

13.5.3 XrtColorEditor の方法

XrtGearIsColorEditor()

オブジェクトが存在し、それが *XrtGearColorEditor* オブジェクトであれば True を返します。

```
Boolean
XrtGearIsColorEditor(
      XrtGearObject editor
      )
```

XrtGearColorEditorDestroy()

XrtGearColorEditor を破棄し、割り当てられていたメモリを解放します。エディタが表示されている場合は、破棄される前にポップ・ダウンされます。

```
void
XrtGearColorEditorDestroy(
      XrtGearObject   editor
      )
```

XrtGearColorEditorCreate()

指定されたウィジェットを親として、ポップアップ・シェルの中に *XrtGearColorEditor* を作成します。popup が True であれば指定された名前を使用し、False であれば、提供されている親の直接の子としてカラー・エディタが作成されます。XmForm などのリソースをエディタにセットするには、*XrtColorEditorGetChild()* を使用して XRTGEAR_COLOR_EDITOR_TOP_WIDGET を取得して、リソースをセットしてください。

```
XrtGearObject
XrtGearColorEditorCreate(
      Widget      parent,
      String      name,
      Boolean     popup
      )
```

parent はカラー・エディタの親、*name* はカラー・エディタの名前です。*popup* はどのようにカラー・エディタが作成されるかを決定します。

XrtGearColorEditorGetName()

指定された XRT/gear カラー・エディタの名前を取得します。名前は使用後に解放しないでください。

```
String
XrtGearColorEditorGetName(
      XrtGearObject    editor
      )
```

XrtGearColorEditorShow()

カラー・エディタのシェルをポップ・アップさせます。

```
Boolean
XrtGearColorEditorShow(
      XrtGearObject    editor,
      Position     x,
      Position     y
      )
```

XrtGearColorEditorHide()

カラー・エディタがポップ・アップとして作成されている場合は、エディタをポップ・ダウンします。現在の状態が保たれ、エディタは破棄されません。渡されたオブジェクトがカラー・エディタで、前に非表示(hide)になり、現在は表示されていれば `True` を返します。

```
Boolean
XrtGearColorEditorHide(
      XrtGearObject    editor
      )
```

XrtGearColorEditorIsShowing()

カラー・エディタのシェルがポップ・アップとして作成され、現在ポップ・アップされていれば `True` を返します。ポップ・アップでなければ、戻り値は undefine されます。

```
Boolean
XrtGearColorEditorIsShowing(
      XrtGearObject    editor
      )
```

XrtGearColorEditorGetChild()

指定されたカラー・エディタ・シェルの子を返します。この子供は破棄しないでください。この関数は、指定されたウィジェットの(簡易関数が提供されていない)プロパティを変更する時に使用できるよう設計されています。

```
Widget
XrtGearColorEditorGetChild(
      XrtGearObject        editor,
      XrtColorEditorChild  child
      )
```

child は *XrtColorEditorChild* 列挙値の一つです。

XrtGearColorEditorSetColorName()

色名によってカラー・エディタで現在選択されている色をセットします。

```
void
XrtGearColorEditorSetColorName(
      XrtGearObject    editor,
      String       color
      )
```

color は色の実際のテキスト名(例: 'Misty Rose')または Hex ストリング値(例: '#fff')です。ここでは、二つのフォーマットが使用されていますが、XParseColor でサポートされているストリング(例: 'TekHVC:0.0/100.0/0.0')であれば、ほとんどどれでも使用できます。このストリングは、表示される時には#RRGGBB フォーマットに変換されます。

XrtGearColorEditorSetColorList()

Combo Box で使用されているデフォルトの色の picklist をユーザによって提供されているリストと置き換えます。このリストは、必ず NULL 終端配列のストリングにしてください。NULL が色のリストとして渡されると、デフォルトのリストが使用されます。これらの色名は、表示時には *rgb.txt* カラー・データベースから検索されます。通常、このファイルは*/usr/lib/X11/rgb.txt* にあります。

```
void
XrtGearColorEditorSetColorList(
      XrtGearObject    editor,
      String       *color_list
      )
```

XrtGearColorEditorSetRGBLabels()

Red、Green、Blue のスライダに使用されるラベルをセットします。ストリングに NULL が渡された場合は、カレント・ラベルは変更されません。この関数の呼び出しの後は、ストリングをコピーし、解放することができます。

```
void
XrtGearColorEditorSetRGBLabels(
      XrtGearObject    editor,
      XmString     red,
      XmString     green,
      XmString     blue
      )
```

XrtGearColorEditorSetRGBValues()

提供された値によって、RGB スライダをアップデートします。範囲は、0 から 255 の間にしてください。カラー・テキスト・ウィジェットとスライダ・テキスト・ウィジェットがアップデートされます。

```
void
XrtGearColorEditorSetRGBValues(
      XrtGearObject    editor,
      unsigned int red,
      unsigned int green,
      unsigned int blue
      )
```

XrtGearColorEditorSetHSBLabels()

Hue、Saturation、Brightness のスライダに使用されるラベルをセットします。ストリングに NULL が渡された場合は、カレント・ラベルは変更されません。この関数の呼び出しの後は、ストリングをコピーし、解放することができます。

```
void
XrtGearColorEditorSetHSBLabels(
      XrtGearObject   editor,
      XmString    hue,
      XmString    saturation,
      XmString    brightness
      )
```

XrtGearColorEditorSetHSBValues()

提供された値によって、HSB スライダをアップデートします。範囲は、Saturation/Brightness を 0 から 100、Hue を 0 から 360 の間にしてください。HSB 値は RGB 値に変換され、RGB 値がアップデートされます。

```
void
XrtGearColorEditorSetHSBValues(
      XrtGearObject   editor,
      unsigned int hue,
      unsigned int saturation,
      unsigned int brightness
      )
```

XrtGearColorEditorGetColorName()

現在選択されている色を表す#RRGGBB ストリングを返します。このストリングにはメモリが割り当てられますが、これの解放は呼び出した側の役割です。match が True であれば、現在の色のリストから適合するエントリが検索され、みつかった場合は、その名前が返されます。リストのサイズによっては、X サーバへの多くの呼び出しが行われますので、パフォーマンスに影響を与えることがあります。

```
String
XrtGearColorEditorGetColorName(
      XrtGearObject   editor,
      Boolean     match
      )
```

XrtGearColorEditorGetRGBLabels()

red、green、blue のスライダに使用されている XmString を返します。これらの値は、コピーされず、この関数の呼び出しの後に解放することはできません。

```
void
XrtGearColorEditorGetRGBLabels(
      XrtGearObject   editor,
      XmString    *red,
      XmString    *green,
      XmString    *blue
      )
```

XrtGearColorEditorGetRGBValues()

範囲 0 から 255 の間の現在の色の red、green、blue の値を返します。

```
void
XrtGearColorEditorGetRGBValues(
      XrtGearObject   editor,
      unsigned int *red,
      unsigned int *green,
      unsigned int *blue
      )
```

XrtGearColorEditorGetHSBLabels()

hue、saturation、brightness スライダに使用されているラベルの XmString を返します。これらの値はコピーされず、この関数の呼び出しの後に解放することはできません。

```
void
XrtGearColorEditorGetHSBLabels(
      XrtGearObject   editor,
      XmString    *hue,
      XmString    *saturation,
      XmString    *brightness
      )
```

XrtGearColorEditorGetHSBValues()

現在の色の HSB 値を返します。

```
void
XrtGearColorEditorGetHSBValues(
      XrtGearObject   editor,
      unsigned int *hue,
      unsigned int *saturation,
      unsigned int *brightness
      )
```

XrtGearColorEditorSetEditTypes()

色の編集の型をセットします。渡される値は、下記の型で OR を使用してください。

```
XRTGEAR_COLOR_EDITOR_BYNAME
XRTGEAR_COLOR_EDITOR_BYRGB
XRTGEAR_COLOR_EDITOR_BYHSB
```

(toggle ボタンも含む)ウィジェットは、この関数によって manage または unmanage されます。“types” = = 0 は、許可されていません。

```
Boolean
XrtGearColorEditorSetEditTypes(
      XrtGearObject       editor,
      XrtColorEditorEditTypes  types
      )
```

XrtGearColorEditorGetEditTypes()

色編集の型を返します。この値は、OR を使用した次の型で構成されます。

```
XRTGEAR_COLOR_EDITOR_BYNAME
XRTGEAR_COLOR_EDITOR_BYRGB
XRTGEAR_COLOR_EDITOR_BYHSB

XrtColorEditorEditTypes
XrtGearColorEditorGetEditTypes(
      XrtGearObject    editor
      )
```

XrtGearColorEditorAddOKCallback()

カラー・エディタのダイアログにある OK ボタンに対して、コールバック手続きとクライアント・データを登録します。

```
void
XrtGearColorEditorAddOKCallback(
      XrtGearObject    editor,
      XtCallbackProc   callback,
      XtPointer    client_data
      )
```

XrtGearColorEditorAddCancelCallback()

カラー・エディタのダイアログにある Cancel ボタンに対して、コールバック手続きとクライアント・データを登録します。

```
void
XrtGearColorEditorAddCancelCallback(
      XrtGearObject    editor,
      XtCallbackProc   callback,
      XtPointer    client_data
      )
```

XrtGearColorEditorSetCloseColorProc()

colormap が一杯の時、要求された色に一番近い色を検索する関数をセットします。デフォルトの手続きは、*color_func.c* にある *XrtGearSampleColorFunction* です。このファイルは、*$XRTHOME/src/gear/utils* にあります。

```
void
XrtGearColorEditorSetCloseColorProc(
      XrtGearObject        editor,
      XrtColorEditorCloseProc  proc
      )
```

XrtGearColorEditorAllocateColor()

現在の値で新しいカラー・セルを割り当て、それを返します。セルは、*read_only* の値によって、read/write または *read_only* とすることができます。この色が必要でなくなった時に *XfreeColors()*で色を解放するのは、ユーザの役割です。

```
Pixel
XrtGearColorEditorAllocateColor(
      XrtGearObject    editor,
      Boolean      read_only
      )
```

13.5.4 Icon 操作の方法

XrtGearIconGetSize()

指定されたアイコンの幅と高さを返します。

```
Boolean
XrtGearIconGetSize(
  Display     *display,
  XrtGearIcon     *icon,
  Dimension       *width
  Dimension       *height,
  )
```

display は、アイコンが表示される画面に関する情報を持つ Display オブジェクトです。*icon* はサイズが調べられるアイコンです。*width* は返された幅、*height* は返された高さです。幅または高さのどちらかが必要なければ、*width* または *height* には、`NULL` を渡してください。

XrtGearIconLoadFromXpmData()

指定された XPM データ構造体から作成された pixmap を使用して、アイコンのイメージ・データをロードします。ファイルが既にロードされている場合は、キャッシュから取り出されます。エラーが起きた場合は `False`、成功時には `True` を返します。

```
Boolean
XrtGearIconLoadFromXpmData
  Widget  w,
  String  *data,
  String  name,
  XrtGearIcon *icon
  )
```

w は、アイコンのイメージ・データがロードされるウィジェット。*data* は、pixmap が作成される、XPM データ構造体です。*icon* はセットされるアイコン、*name* はアイコンの名前です。

XrtGearIconLoadFromXpmFile()

指定されたファイルから作成された pixmap を使用して、アイコンのイメージ・データをロードします。ファイルが既にロードされている場合は、キャッシュから取り出されます。エラーが起きた場合は `False`、成功時には `True` を返します。

```
Boolean
XrtGearIconLoadFromXpmFile(
  Widget  w,
  String  filename,
  XrtGearIcon *icon
  )
```

w は、アイコンのイメージ・データがロードされるウィジェット。*filename* は、pixmap が作成される、ファイル名。*icon* はセットされるアイコンです。

Part II

Reference Appendices

A

Data 型

XRT/gear のデータ型のアルファベット順のリスト。構造体の C の定義も提供されています。

XrtGearAction

TabManager の XmNxrtGearPageCallback によって使用される列挙法です。(各コールバックに渡される *action* メンバ)

```
XRTGEAR_ACTION_OTHERS
XRTGEAR_ACTION_SCROLL_INCR
XRTGEAR_ACTION_SCROLL_DECR
XRTGEAR_ACTION_SET_VALUES
XRTGEAR_ACTION_TAB
```

XrtGearAlignerAlignment

Aligner の XmNxrtGearAlignerAlignment によって、ラベルの位置指定に使用される列挙法です。

```
XRTGEAR_ALIGNER_BEGINNING
XRTGEAR_ALIGNER_CENTER
XRTGEAR_ALIGNER_END
```

XrtGearAlignment

Column オブジェクトの XmNxrtGearLabelAlignment と XmNxrtGearNodeAlignment によって、ラベルや列のテキストの行揃えに使用される列挙法です。

```
XRTGEAR_ALIGN_BEGINNING
XRTGEAR_ALIGN_CENTER
XRTGEAR_ALIGN_END
```

XrtGearAnchorSide

Tab Button の XmNxrtGearAnchorSide によって、ウィジェットのどの側が親に接するかを指定する時に使用される列挙法です。

```
XRTGEAR_ANCHOR_SIDE_BOTTOM
XRTGEAR_ANCHOR_SIDE_LEFT
XRTGEAR_ANCHOR_SIDE_RIGHT
XRTGEAR_ANCHOR_SIDE_TOP
```

XrtGearArrowType

ComboBox の XmNxrtGearArrowType によって、矢印の種類の指定に使用される列挙法です。

```
XRTGEAR_ARROW_CUSTOM
XRTGEAR_ARROW_NONE
XRTGEAR_ARROW_WEDGE
XRTGEAR_ARROW_WITH_UNDERBAR
```

XrtGearAttachPolicy

Tab Manager の XmNxrtGearAttachPolicy によって、Tab Button に自動的にページが割り当てられるようにするかどうかを指定する列挙法です。

```
XRTGEAR_ATTACH_ALWAYS
XRTGEAR_ATTACH_NONE
```

XrtGearBorderLocation

Outliner の XmNxrtGearBorderLocation によって、枠線を描画する場所の指定に使用される列挙法です。

```
XRTGEAR_BORDER_LOC_LABELS
XRTGEAR_BORDER_LOC_NODES
XRTGEAR_BORDER_LOC_LABELS_SEPARATED_FROM_NODES
XRTGEAR_BORDER_LOC_LABELS_AND_NODES
XRTGEAR_BORDER_LOC_EACH_LABEL_SEPARATED_FROM_NODES,
XRTGEAR_BORDER_LOC_LABELS_AND_NODES
```

XrtGearBorderType

Enhanced Label と Outliner の XmNxrtGearBoderType 及び Widget Tips の XmNxrtGearTipsBorderType によって、枠線の種類の指定に使用される列挙法です。

```
XRTGEAR_BORDER_NONE
XRTGEAR_BORDER_BEVEL
XRTGEAR_BORDER_ETCHED_IN
XRTGEAR_BORDER_ETCHED_OUT
XRTGEAR_BORDER_FRAME_IN
XRTGEAR_BORDER_FRAME_OUT
XRTGEAR_BORDER_IN
XRTGEAR_BORDER_OUT
XRTGEAR_BORDER_SHADOW
XRTGEAR_BORDER_PLAIN
```

XrtGearChildType

Tab Manager の XmNxrtGearChildType によって、子の型指定に使用される列挙法です。

```
XRTGEAR_CHILD_TYPE_INTERNAL
XRTGEAR_CHILD_TYPE_PAGE
XRTGEAR_CHILD_TYPE_TAB
```

XrtColorEditorChildType

Color Editor によって、子の型の指定に使用される列挙法です。

```
XRTGEAR_COLOR_EDITOR_TEXT_TOGGLE
XRTGEAR_COLOR_EDITOR_RGB_TOGGLE
XRTGEAR_COLOR_EDITOR_HSB_TOGGLE
XRTGEAR_COLOR_EDITOR_TOP_WIDGET
XRTGEAR_COLOR_EDITOR_COLOR_COMBO
XRTGEAR_COLOR_EDITOR_COLOR_LABEL
XRTGEAR_COLOR_EDITOR_COLOR_TEXT
XRTGEAR_COLOR_EDITOR_COLOR_PREVIEW
XRTGEAR_COLOR_EDITOR_OK_BUTTON
XRTGEAR_COLOR_EDITOR_CANCEL_BUTTON
```

XrtGearColumnMoveCallbackStruct

Outliner によって、XmNxrtGearColumnMoveCallback リストのコールバックに渡される情報の定義に使用される構造体です。

```
typedef struct {
  XrtGearReason   reason;      /* read-only */
  XEvent      *event;      /* read-only */
  Widget      column;      /* read-only */
  Widget      target_sibling; /* read-only */
  Boolean     doit;
} XrtGearColumnMoveCallbackStruct;
```

XrtGearColumnResizePolicy

Outliner の XrtColumn オブジェクトの XmNxrtGearWidthResizePolicy によって、列のリサイズの決まりの指定に使用される列挙法です。

```
XRTGEAR_COLUMN_RESIZE_ALL
XRTGEAR_COLUMN_RESIZE_LABELS_ONLY
XRTGEAR_COLUMN_RESIZE_NONE
```

XrtGearColumnSelectionCallbackStruct

Outliner によって、XmNxrtGearColumnSelectionCallback リストのコールバックに渡される情報の定義に使用される構造体です。

```
typedef struct {
  XrtGearReason   reason;      /* read-only */
  XEvent      *event;      /* read-only */
  Widget      old_column; /* read-only */
  Widget      new_column;
  Boolean     doit;
} XrtGearColumnSelectionCallbackStruct;
```

XrtGearCursorTrackingCallbackStruct

Outliner によって、XmNxrtGearCursorTrackingCallback リストのコールバックに渡される情報の定義に使用される構造体です。

```
typedef struct {
  XrtGearReason   reason;     /* read-only */
  XEvent      *event;     /* read-only */
  Widget      object;     /* read-only */
  Cursor      old_cursor; /* read-only */
  Cursor      new_cursor;
} XrtGearCursorTrackingCallbackStruct;
```

XrtGearDirection

Outliner によって、*XrtGearNodeSearchTree()*を呼び出す時の検索方向の指定に使用される列挙法です。

```
XRTGEAR_DIRECTION_FORWARD
XRTGEAR_DIRECTION_BACKWARD
```

XrtGearDisplayCallbackStruct

Widget Tips によって、XmNxrtGearDisplayCallback リストのコールバックに渡される情報の定義に使用される構造体です。

```
typedef struct {
  int     reason; /* read-only */
  XEvent  *event; /* read-only */
  Widget  widget; /* read-only */
  Boolean doit;
  XmString    text;
} XrtGearDisplayCallbackStruct;
```

XrtGearDisplayPolicy

Outliner の XmNxrtGearScrollBarHorizDisplayPolicy と XmNxrtGearScrollBarVertDisplayPolicy によるスクロールバーの表示や、Tab Manager の XmNxrtGearArrowDisplayPolicy による矢印の表示の指定に使用される列挙法です。

```
XRTGEAR_DISPLAY_AS_NEEDED
XRTGEAR_DISPLAY_ALWAYS
XRTGEAR_DISPLAY_NEVER
```

XrtGearEditType

Outliner によって、どのようにセルが編集されるかを決定する列挙法です。

```
XRTGEAR_EDIT_TYPE_WINDOWS/* デフォルト、一回目のクリックで選択、
                            次のクリックで編集。エディタの
                            テキストが最初に選択されている */
XRTGEAR_EDIT_TYPE_MOTIF /* クリックで選択と編集を行う。*/
```

XrtGearEnhancedSizeCallbackStruct

Column オブジェクトによって、XmNxrtGearResizeCallback リストのコールバックに渡される情報の定義に使用される構造体です。

```
typedef struct {
  int     reason;      /* read-only */
  XEvent  *event;      /* read-only */
  Dimension   width;
  Dimension   height;
  Dimension   old_width;  /* read-only */
  Dimension   old_height; /* read-only */
  Boolean doit;
} XrtGearEnhancedSizeCallbackStruct;
```

XrtGearFolderState

FolderNode オブジェクトの XmNxrtGearFolderState によって、フォルダの状態の指定に使用される列挙法です。

```
XRTGEAR_FOLDERSTATE_CLOSED
XRTGEAR_FOLDERSTATE_OPEN_FOLDERS
XRTGEAR_FOLDERSTATE_OPEN_ITEMS
XRTGEAR_FOLDERSTATE_OPEN_ALL
```

XrtGearIcon

NodeStyle と Progress によってアイコンの指定に使用される構造体です。

```
typedef struct {
  Pixmap  icon;
  Pixmap  icon_mask;
} XrtGearIcon;
```

XrtGearImportPolicy

Outliner の XmNxrtGearNodeImportPolicy によって、ドラッグとドロップ操作のインポートの決まりの指定に使用される構造体です。

```
XRTGEAR_IMPORT_NOTHING
XRTGEAR_IMPORT_ANYTHING
XRTGEAR_IMPORT_WITHIN_APP
XRTGEAR_IMPORT_WITHIN_WIDGET
```

XrtGearIndentPolicy

Outliner の XmNxrtGearNodeIndentPolicy によって、インデント幅の指定に使用される列挙法です。

```
XRTGEAR_INDENT_PIXEL
XRTGEAR_INDENT_CHAR
XRTGEAR_INDENT_NONE
```

XrtGearIndicator

Enhanced Toggle の XmNxrtGearIndicator によって、インディケータのスタイルの指定に使用される列挙法です。

```
XRTGEAR_INDICATOR_NONE
XRTGEAR_INDICATOR_CHECK
XRTGEAR_INDICATOR_CHECK_BOX
XRTGEAR_INDICATOR_CIRCLE
XRTGEAR_INDICATOR_CIRCLE_BOX
XRTGEAR_INDICATOR_CROSS
XRTGEAR_INDICATOR_CROSS_BOX
XRTGEAR_INDICATOR_FILL
XRTGEAR_INDICATOR_PIXMAP
```

XrtGearLineStyle

NodeStyle オブジェクトの XmNxrtGearLineStyle によって、線のスタイルの指定に使用される列挙法です。

```
XRTGEAR_LINESTYLE_NONE
XRTGEAR_LINESTYLE_DOTTED
XRTGEAR_LINESTYLE_SOLID
```

XrtGearListChangedCallbackStruct

List オブジェクトの *XrtGearListSetChangedProc()*によって、リスト･オブジェクトが変更される時に呼び出される手続きの指定に使用される構造体です。

```
typedef struct {
  XrtGearReason   reason; /* read-only */
  XtPointer       user_data;
  XtPointer       modified_item;
  int         position;
} XrtGearListChangedCallbackStruct;
```

XrtGearLocation

Node オブジェクトの *XrtGearNodeFromXEvent()*と *XrtGearNodeFromXY()*によって、イベントが起こった(ノードに相対する)位置の指定に使用される列挙法です。

```
XRTGEAR_LOCATION_ON_NODE
XRTGEAR_LOCATION_BEFORE_NODE
XRTGEAR_LOCATION_ON_SHORTCUT
XRTGEAR_LOCATION_ON_LABEL
XRTGEAR_LOCATION_NOWHERE
```

XrtGearMenuAlignment

ComboBox の XmNxrtGearMenuAlignment によって、ポップアップ･メニューの(テキスト･フィールドに相対する)位置の指定に使用される列挙法です。

```
XRTGEAR_MENU_ALIGN_BEGINNING
XRTGEAR_MENU_ALIGN_BOTH
XRTGEAR_MENU_ALIGN_END
```

XrtGearMenuCallbackStruct

ComboBox によって、XmNxrtGearMenuCallback リストのコールバックに渡される情報の定義に使用される構造体です。

```
typedef struct {
  int     reason;      /* read-only */
  XEvent  *event;      /* read-only */
  int     position;    /* read-only */
  String  value;
  Boolean doit;        /* initially True */
} XrtGearMenuCallbackStruct;
```

XrtGearMovePolicy

Outliner の XmNxrtGearNodeMovePolicy によって、ウィジェット内のドラッグ&ドロップ操作のコントロールに使用される列挙法です。

```
XRTGEAR_MOVE_NOWHERE
XRTGEAR_MOVE_WITHIN_LEVEL
XRTGEAR_MOVE_WITHIN_WIDGET
XRTGEAR_MOVE_WITHIN_APP
```

XrtGearNodeActivateCallbackStruct

Outliner によって、XmNxrtGearActivateCallback リストのコールバックに渡される情報の定義に使用される構造体です。

```
typedef struct {
  XrtGearReason   reason;      /* read-only */
  XEvent      *event;      /* read-only */
  Widget      container;   /* read-only */
  Widget      node;
  int         click_count;
} XrtGearNodeActivateCallbackStruct;
```

XrtGearNodeCreateCallbackStruct

Outliner によって、XmNxrtGearNodeCreateCallback リストのコールバックに渡される情報の定義に使用される構造体です。

```
typedef struct {
  XrtGearReason   reason;      /* read-only */
  XEvent      *event;      /* read-only */
  Widget      source_node; /* read-only */
  XrtGearObject   label;
  Widget      node_style;
  XtPointer       user_data;
  Widget      new_node;
} XrtGearNodeCreateCallbackStruct;
```

XrtGearNodeCurrentChangedCallbackStruct

Outliner によって、XmNxrtGearNodeCurrentChangedCallback リストのコールバックに渡される情報の定義に使用される構造体です。

```
typedef struct {
  XrtGearReason   reason;     /* read-only */
  XEvent      *event;     /* read-only */
  Widget      container;  /* read-only */
  Widget      previous_node;  /* read-only */
  Widget      new_node;
  Boolean     doit;
} XrtGearNodeCurrentChangedCallbackStruct;
```

XrtGearNodeEnterCellCallbackStruct

Outliner によって、XmNxrtGearEnterCellCallback リストのコールバックに渡される情報の定義に使用される構造体です。

```
typedef struct {
     XrtGearReason   reason;
     Xevent      *event;
     Widget      previous_node;
     Int     previous_column;
     Widget      editor;
     Widget      new_node; /* read/write */
     Int     new_column; /* read/write */
     Boolean     doit; /* Initially True */
} XrtGearNodeEnterCellCallbackStruct;
```

XrtGearNodeFolderStateCallbackStruct

Outliner によって、XmNxrtGearFolderStateCallback リストのコールバックに渡される情報の定義に使用される構造体です。

```
typedef struct {
  XrtGearReason   reason;     /* read-only */
  XEvent      *event;     /* read-only */
  Widget      container;  /* read-only */
  Widget      node;
  XrtGearFolderState  new_state;
  XrtGearFolderState  old_state;  /* read-only */
  Boolean doit;
} XrtGearNodeFolderStateCallbackStruct;
```

XrtGearNodeMoveCallbackStruct

Outliner によって、XmNxrtGearNodeMoveCallback リストのコールバックに渡される情報の定義に使用される構造体です。

```
typedef struct {
  XrtGearReason   reason;     /* read-only */
  XEvent      *event;     /* read-only */
  unsigned char   operation;  /* read-only */
  Widget      node;       /* read-only */
  Widget      old_parent; /* read-only */
  Widget      new_parent; /* read-only */
  Widget      new_next;
  Boolean     doit;
} XrtGearNodeMoveCallbackStruct;
```

XrtGearNodeValidateCallbackStruct

Outliner によって、XmNxrtGearNodeValidateCallback リストのコールバックに渡される情報の定義に使用される構造体です。

```
typedef struct {
     XrtGearReason   reason;
     Xevent      *event;
     Widget      node;
     XrtGearObject   old_label;
     int     column;
     String      value; /* read/write */
     Boolean     doit; /* Initially True */
     Boolean     bell; /* Initially True */
} XrtGearNodeValidateCallbackStruct;
```

XrtGearOutlinerImportCallbackStruct

Widget Tips によって、XmNxrtGearNodeImportCallback リストのコールバックに渡される情報の定義に使用される構造体です。

```
typedef struct {
  XrtGearReason   reason;     /* read-only */
  XEvent      *event;     /* read-only */
  String      target;     /* read-only */
  String      data;       /* read-only */
  unsigned long   data_length;/* read-only */
  Widget      object;     /* read-only */
  Boolean     doit;
} XrtGearOutlinerImportCallbackStruct;
```

XrtGearPageCallbackStruct

Tab Manager によって、XmNxrtGearPageCallback リストのコールバックに渡される情報の定義に使用される構造体です。

```
typedef struct {
  int         reason;     /* read-only */
  XEvent      *event;     /* read-only */
  XrtGearAction   action;     /* read-only */
  int         prev_page_number;/* read-only */
  Widget      prev_page_widget;/* read-only */
  Widget      prev_tab;   /* read-only */
  Widget      tab;
  int         page_number;
  Widget      page_widget;
  Boolean     doit;
} XrtGearPageCallbackStruct;
```

XrtGearPaneResizeCallbackStruct

Paned Window によって、XmNxrtGearPaneResizeCallback リストのコールバックに渡される情報の定義に使用される構造体です。

```
typedef struct {
     XrtGearReason   reason; /* read-only */
     Xevent      *event; /* read-only */
     Widget      child; /* read-only */
     Int     position; /* read-only */
     Dimension   width;
     Dimension   height;
     Boolean     doit;
} XrtGearPaneResizeCallbackStruct;
```

XrtGearPrintCallbackStruct

Outliner によって、XmNxrtGearPrintCallback リストのコールバックに渡される情報の定義に使用される構造体です。

```
typedef struct {
  int         reason;     /* read-only */
  XEvent      *event;     /* read-only */
  int         current_page;   /* read-only */
  int         total_pages;/* read-only */
  int         across_pages;   /* read-only */
  int         down_pages; /* read-only */
  int         page_number;
  String      header, footer;
  FILE        *fp;
  Boolean     write_file_header;
  Boolean     write_file_trailer;
  Boolean     doit;
} XrtGearPrintCallbackStruct;
```

XrtGearProgressCallbackStruct

Progress ウィジェットによって、XmNxrtGearProgressCallback リストのコールバックに渡される情報の定義に使用される構造体です。

```
typedef struct {
  int     reason;     /* read-only */
  XEvent  *event;     /* read-only */
  int     maximum;    /* read-only */
  int     minimum;    /* read-only */
  double  value_percent;  /* read-only */
  XmString    label;
  int     value;
  Boolean continue_progression;
} XrtGearProgressCallbackStruct;
```

XrtGearProgressType

Progress ウィジェットの XmNxrtGearProgressType によって、ウィジェットの表示の指定に使用される列挙法です。

```
XRTGEAR_PROGRESS_BAR
XRTGEAR_PROGRESS_DISCRETE_BAR
XRTGEAR_PROGRESS_ICON
XRTGEAR_PROGRESS_STACKING_ICONS
```

XrtGearPSFontMap

Outliner の XmNxrtGearPSFontMapList によって、PostScript フォントに対応する X フォントの定義に使用される構造体です。

```
typedef struct {
  String  font_name, tag;
  String  PS_font;
  double  point_size;
} XrtGearPSFontMap;
```

XrtGearReason

XRT/gear によって、コールバックの理由(reason)の指定に使用される列挙法です。

```
XRTGEAR_REASON_ACTIVATE
XRTGEAR_REASON_COLUMN_MOVE_BEGIN
XRTGEAR_REASON_COLUMN_MOVE_END
XRTGEAR_REASON_COLUMN_SELECT_BEGIN
XRTGEAR_REASON_COLUMN_SELECT_END
XRTGEAR_REASON_DISPLAY_BEGIN
XRTGEAR_REASON_DISPLAY_END
XRTGEAR_REASON_FOCUS_IN_BEGIN
XRTGEAR_REASON_FOCUS_IN_END
XRTGEAR_REASON_FOCUS_OUT
XRTGEAR_REASON_FOLDER_CHANGE_BEGIN
XRTGEAR_REASON_FOLDER_CHANGE_END
XRTGEAR_REASON_FORMAT_LABEL_BEGIN
XRTGEAR_REASON_FORMAT_LABEL_END
XRTGEAR_REASON_IMPORT_BEGIN
XRTGEAR_REASON_IMPORT_END
XRTGEAR_REASON_LIST_ADD
XRTGEAR_REASON_LIST_DELETED
XRTGEAR_REASON_LIST_MOVE
XRTGEAR_REASON_LIST_SET
XRTGEAR_REASON_MENU_BEGIN
XRTGEAR_REASON_MENU_END
XRTGEAR_REASON_NODE_CHANGE_BEGIN
XRTGEAR_REASON_NODE_CHANGE_END
XRTGEAR_REASON_NODE_CREATE_BEGIN
XRTGEAR_REASON_NODE_CREATE_END
XRTGEAR_REASON_NODE_MOVE_BEGIN
XRTGEAR_REASON_NODE_MOVE_END
XRTGEAR_REASON_PAGE_BEGIN
XRTGEAR_REASON_PAGE_END
XRTGEAR_REASON_PAGE_PRINT_BEGIN
XRTGEAR_REASON_PAGE_PRINT_END
XRTGEAR_REASON_PROGRESS
XRTGEAR_REASON_PROGRESS_BEGIN
XRTGEAR_REASON_PROGRESS_COMPLETE
XRTGEAR_REASON_RESIZE_BEGIN
XRTGEAR_REASON_RESIZE_END
XRTGEAR_REASON_SELECT_BEGIN
XRTGEAR_REASON_SELECT_END
XRTGEAR_REASON_STATE_BEGIN
XRTGEAR_REASON_STATE_END
XRTGEAR_REASON_UNSELECT_BEGIN
XRTGEAR_REASON_UNSELECT_END
```

XrtGearResizeCallbackStruct

Tab Manager と Outliner によって、XmNxrtGearResizeCallback リストのコールバックに渡される情報の定義に使用される構造体です。

```
typedef struct {
  int     reason; /* read-only */
  XEvent  *event; /* read-only */
  Dimension   width;
  Dimension   height;
} XrtGearResizeCallbackStruct;
```

XrtGearRotation

Enhanced Label、Enhanced Pushbutton、Tab Button の XmNxrtGearPixmapRotation、XmNxrtGearStringRotation、XmNxrtGearTabRotation によって、回転の指定に使用される列挙法です。

```
XRTGEAR_ROTATE_NONE
XRTGEAR_ROTATE_90
XRTGEAR_ROTATE_180
XRTGEAR_ROTATE_270
XRTGEAR_ROTATE_UNSPECIFIED
```

XrtGearSearchPolicy

Combo Box の XmNxrtGearSearchPolicy によって、pick list の検索法の指定に使用される列挙法です。

```
XRTGEAR_SEARCH_SINGLE
XRTGEAR_SEARCH_MULTICHAR
```

XrtGearSelectCallbackStruct

Outliner によって、XmNxrtGearSelectionCallback リストのコールバックに渡される情報の定義に使用される構造体です。

```
typedef struct {
  XrtGearReason   reason;     /* read-only */
  XEvent      *event;     /* read-only */
  Widget      container;  /* read-only */
  Widget      node;
  XrtGearObject   selected_node_list; /* read-only */
  Boolean     doit;
} XrtGearSelectCallbackStruct;
```

XrtGearSelecitonPolicy

Outliner の XmNxrtGearSelectionPolicy によって、選択の決まりの指定に使用される列挙法です。

```
XRTGEAR_SELECTION_POLICY_NONE
XRTGEAR_SELECTION_POLICY_SINGLE
XRTGEAR_SELECTION_POLICY_SINGLE_AUTO_SELECT
XRTGEAR_SELECTION_POLICY_RANGE
XRTGEAR_SELECTION_POLICY_MULTIPLE
XRTGEAR_SELECTION_POLICY_MULTIPLE_AUTO_UNSELECT
```

XrtGearSelectionRestrictions

Outliner の XmNxrtGearSelectionRestriction によって、選択の制限の指定に使用される列挙法です。

```
XRTGEAR_RESTRICT_NOTHING
XRTGEAR_RESTRICT_ITEMS
XRTGEAR_RESTRICT_FOLDERS
XRTGEAR_RESTRICT_MULTI_LEVEL
XRTGEAR_RESTRICT_ALL
```

XrtGearSizingPolicy

Outlienr と Column オブジェクトの XmNxrtGearHeightSizingPolicy と XmNxrtGearWidthSizingPolicy によって、ツリーの理想的な表示サイズの計算法の指定に使用される列挙法です。

```
XRTGEAR_SIZE_PIXEL
XRTGEAR_SIZE_CHAR
XRTGEAR_SIZE_COLUMN
XRTGEAR_SIZE_NODE
XRTGEAR_SIZE_LABEL_VARIABLE
XRTGEAR_SIZE_VARIABLE
```

XrtGearSnapArg

Widget Snapshot によって、属性と値のペアを *XrtGearSnapPrintPixmap()*に渡す時に使用される構造体です。

```
typedef struct {
  XrtGearSnapDrawArg  name;
  XtArgVal        value;
} XrtGearSnapArg;
```

XrtGearSnapDrawArg

Widget Snapshot によって、*XrtGearSnapPrintPixmap()*へ渡される属性と値に使用される列挙法です。

```
XRTGEAR_SNAP_ALL
XRTGEAR_SNAP_AS_IS
XRTGEAR_SNAP_AUTO
XRTGEAR_SNAP_BEST
XRTGEAR_SNAP_CM
XRTGEAR_SNAP_COLOR
XRTGEAR_SNAP_INCHES
XRTGEAR_SNAP_LANDSCAPE
XRTGEAR_SNAP_MARGIN_BOTTOM
XRTGEAR_SNAP_MARGIN_LEFT
XRTGEAR_SNAP_MARGIN_RIGHT
XRTGEAR_SNAP_MARGIN_TOP
XRTGEAR_SNAP_MONO
XRTGEAR_SNAP_NONE
XRTGEAR_SNAP_ORIENTATION
XRTGEAR_SNAP_PAPERSIZE_HEIGHT
XRTGEAR_SNAP_PAPERSIZE_WIDTH
XRTGEAR_SNAP_PORTRAIT
XRTGEAR_SNAP_SCALE
XRTGEAR_SNAP_SHOWPAGE
XRTGEAR_SNAP_UNITS
XRTGEAR_SNAP_NUM_COPIES
```

XrtGearState

Enhanced Toggle の XmNxrtGearState によって、ボタンの状態の指定に使用される列挙法です。

```
XRTGEAR_STATE_OFF
XRTGEAR_STATE_ON
XRTGEAR_STATE_INDETERMINATE
```

XrtGearStateChangedCallbackStruct

Enhanced Toggle によって、XmNxrtGearStateChangedCallback リストのコールバックに渡される情報の定義に使用される構造体です。

```
typedef struct {
  int     reason; /* read-only */
  XEvent  *event; /* read-only */
  int     state;
} XrtGearStateChangedCallbackStruct;
```

XrtGearStringLayout

Enhanced Label、Enhanced Pushbutton、Tab Button の XmNxrtGearStringLayout によって、pixmap に対応するテキストの位置の指定に使用される列挙法です。

```
XRTGEAR_STRING_BOTTOM
XRTGEAR_STRING_CENTER
XRTGEAR_STRING_LEFT
XRTGEAR_STRING_RIGHT
XRTGEAR_STRING_TOP
```

XrtGearStyle

Tab Button の XmNxrtGearStyle によって、ボタンのスタイルの指定に使用される列挙法です。

```
XRTGEAR_STYLE_CHAMFERED
XRTGEAR_STYLE_ROUNDED
XRTGEAR_STYLE_SLANTED
XRTGEAR_STYLE_SQUARE
```

XrtGearTabSide

Tab Manager の XmNxrtGearTabSide によって、タブが表示される位置の指定に使用される列挙法です。

```
XRTGEAR_SIDE_BOTTOM
XRTGEAR_SIDE_LEFT
XRTGEAR_SIDE_RIGHT
XRTGEAR_SIDE_TOP
```

XrtGearTextClipPolicy

Outliner の XmNxrtGearTextHorizClipPolicy によって、長すぎるテキストの処理方法の指定に使用される列挙法です。

```
XRTGEAR_CLIP_TRUNCATE
XRTGEAR_CLIP_TRUNCATE_WITH_ARROW
XRTGEAR_CLIP_COMPRESS_WITH_ELLIPSIS
```

XrtGearType

Enhanced Label、Enhanced Pushbutton、Tab Button の XmNxrtGearLabelType によって、ラベルのタイプの指定に使用される列挙法です。

```
XRTGEAR_TYPE_UNSPECIFIED
XRTGEAR_TYPE_PIXMAP
XRTGEAR_TYPE_STRING
XRTGEAR_TYPE_STRING_PIXMAP
```

XrtGearWidgetResizePolicy

Outliner の XmNxrtGearHeightResizePolicy と XmNxrtGearWidthResizePolicy によって、ウィジェットのリサイズの指定に使用される列挙法です。

```
XRTGEAR_WIDGET_RESIZE_VARIABLE
XRTGEAR_WIDGET_RESIZE_GROW_ONLY
XRTGEAR_WIDGET_RESIZE_FIXED
```

B

リソースの型とクラス

各 XRT/gear リソースの型とクラスのリストです。

B.1 Aligner

Name	Class	Type
XmNxrtGearColumnSpacing	XmCXrtGearColumnSpacing	XmRHorizontalDimension
XmNxrtGearMarginHeight	XmCXrtGearMarginHeight	XmRHorizontalDimension
XmNxrtGearMarginWidth	XmCXrtGearMarginWidth	XmRHorizontalDimension
XmNxrtGearOrientation	XmCXrtGearOrientation	XmROrientation
XmNxrtGearRowSpacing	XmCXrtGearRowSpacing	XmRHorizontalDimension

B.1.1 Aligner Constraint リソース

Name	Class	Type
XmNxrtGearAlignerAlignment	XmCXrtGearAlignerAlignment	XmRXrtGearAlignerAlignment
XmNxrtGearAlignBaseline	XmCXrtGearAlignBaseline	XmRBoolean
XmNxrtGearPosition	XmCXrtGearPosition	XmRInt
XmNxrtGearResizeHeight	XmCXrtGearResizeHeight	XmRBoolean
XmNxrtGearResizeWidth	XmCXrtGearResizeWidth	XmRBoolean

B.2 Combo Box

Name	Class	Type
XmNxrtGearAlignment	XmCXrtGearAlignment	XmRXrtGearMenuAlignment
XmNxrtGearArrow	XmCReadOnly	XmRWidget
XmNxrtGearArrowClass	XmCReadOnly	XmRPointer
XmNxrtGearArrowType	XmCXrtGearArrowType	XmRxrtGearArrowType
XmNxrtGearAutoComplete	XmCBoolean	XmRBoolean
XmNxrtGearMenuCallback	XmCCallback	XmRCallback
XmNxrtGearMenuList	XmCReadOnly	XmRWidget
XmNxrtGearPickList	XmCXrtGearPickList	XmRStringTable
XmNxrtGearPickListIndex	XmCXrtGearPickListIndex	XmRInt
XmNxrtGearSearchPolicy	XmCXrtGearSearchPolicy	XmRXrtGearSearchPolicy
XmNxrtGearSpacing	XmCXrtGearSpacing	XmRHorizontalDimension
XmNxrtGearTextField	XmCReadOnly	XmRWidget

B.3 Enhanced Label

Name	Class	Type
XmNxrtGearBorderType	XmCXrtGearBorderType	XmRXrtGearBorderType
XmNxrtGearLabelPixmap	XmCXrtGearLabelPixmap	XmRXrtGearPixmap
XmNxrtGearLabelType	XmCXrtGearLabelType	XmRXrtGearType
XmNxrtGearLayoutSpacing	XmCXrtGearLayoutSpacing	XmRDimension
XmNxrtGearMask	XmCXrtGearMask	XmRXrtGearMask
XmNxrtGearPixmapRotation	XmCXrtGearPixmapRotation	XmRXrtGearRotation
XmNxrtGearStringLayout	XmCXrtGearStringLayout	XmRXrtGearStringLayout
XmNxrtGearStringRotation	XmCXrtGearStringRotation	XmRXrtGearRotation

B.4 Enhanced Pushbutton

Name	Class	Type
XmNxrtGearActive	XmCBoolean	XmRBoolean
XmNxrtGearArmMask	XmCXrtGearArmMask	XmRXrtGearMask

Name	Class	Type
XmNxrtGearArmPixmap	XmCXrtGearArmPixmap	XmRXrtGearPixmap
XmNxrtGearLabelPixmap	XmCXrtGearLabelPixmap	XmRXrtGearPixmap
XmNxrtGearLabelType	XmCXrtGearLabelType	XmRXrtGearType
XmNxrtGearLayoutSpacing	XmCXrtGearLayoutSpacing	XmRDimension
XmNxrtGearMask	XmCXrtGearMask	XmRXrtGearMask
XmNxrtGearPixmapRotation	XmCXrtGearPixmapRotation	XmRXrtGearRotation
XmNxrtGearStringLayout	XmCXrtGearStringLayout	XmRXrtGearStringLayout
XmNxrtGearStringRotation	XmCXrtGearStringRotation	XmRXrtGearRotation

B.5 Enhanced Toggle

Name	Class	Type
XmNxrtGearIndicator	XmCXrtGearIndicator	XmRXrtGearIndicator
XmNxrtGearIndicatorMaskList	XmCXrtGearIndicatorMaskList	XmRXrtGearIndicatorMaskList
XmNxrtGearIndicatorPixmapList	XmCXrtGearIndicatorPixmapList	XmRXrtGearIndicatorPixmapList
XmNxrtGearNumStates	XmCXrtGearNumStates	XmRInt
XmNxrtGearSelectColor	XmCXrtGearSelectColor	XmRPixel
XmNxrtGearState	XmCXrtGearState	XmRXrtGearState
XmNxrtGearStateChangedCallback	XmCCallback	XmRCallback
XmNxrtGearUnselectColor	XmCXrtGearUnselectColor	XmRPixel

B.6 Outliner

Name	Class	Type
XmNxrtGearActivateCallback	XmCCallback	XmRCallback
XmNxrtGearAutoSort	XmCXrtGearAutoSort	XmRBoolean
XmNxrtGearBorderLocation	XmCXrtGearBorderLocation	XmRXrtGearBorderLocation
XmNxrtGearBorderType	XmCXrtGearBorderType	XmRXrtGearBorderType
XmNxrtGearColumnLabelsShow	XmCXrtGearColumnLabelsShow	XmRBoolean
XmNxrtGearColumnList	XmCXrtGearColumnList	XmRXrtGearColumnList
XmNxrtGearColumnMoveAllow	XmCXrtGearColumnMoveAllow	XmRBoolean

Name	Class	Type
XmNxrtGearColumnMoveCallback	XmCXrtGearColumnMoveCallback	XmRCallback
XmNxrtGearColumnOrderList	XmCXrtGearColumnList	XmRXrtGearList
XmNxrtGearColumnSelected	XmCXrtGearColumnSelected	XmRXrtGearWidget
XmNxrtGearColumnSelection Callback	XmCXrtGearColumnSelection Callback	XmRCallback
XmNxrtGearCursor	XmCCursor	XmRCursor
XmNxrtGearCursorResizeWidth	XmCCursor	XmRCursor
XmNxrtGearCursorTracking	XmCXrtGearCursorTracking	XmRBoolean
XmNxrtGearCursorTracking Callback	XmCCallback	XmRCallback
XmNxrtGearDoubleBuffer	XmCXrtGearDoubleBuffer	XmRBoolean
XmNeditable	XmCBoolean	XmRBoolean
XmNxrtGearEditColumn	XmCXrtGearEditColumn	XmRInt
XmNxrtGearEditNode	XmCXrtGearEditNode	XmRWidget
XmNxrtGearEditor	XmCXrtGearEditor	XmRWidget
XmNxrtGearFolderShortcutShow	XmCBoolean	XmRBoolean
XmNxrtGearFolderStateCallback	XmCCallback	XmRCallback
XmNxrtGearFolderStateMask	XmCXrtGearFolderStateMask	XmRXrtGearFolderStateMask
XmNfontList	XmCFontList	XmRFontList
XmNxrtGearHeightPreferredCount	XmCXrtGearHeightPreferredCount	XmRInt
XmNxrtGearHeightResizePolicy	XmCXrtGearHeightResizePolicy	XmRXrtGearWidgetResizePolicy
XmNxrtGearHeightSizingPolicy	XmCXrtGearHeightSizingPolicy	XmRXrtGearSizingPolicy
XmNxrtGearHeightVirtual	XmCXrtGearHeightVirtual	XmRInt
XmNhighlightThickness	XmCHighlightThickness	XmRDimension
XmNxrtGearImportCallback	XmCCallback	XmRCallback
XmNxrtGearMarginHeight	XmCXrtGearMarginHeight	XmRDimension
XmNxrtGearMarginWidth	XmCXrtGearMarginWidth	XmRDimension
XmNxrtGearNodeChildList	XmCXrtGearNodeChildList	XmRXrtGearNodeChildList
XmNxrtGearNodeCompareProc	XmCXrtGearNodeCompareProc	XmRXrtGearNodeCompareProc
XmNxrtGearNodeCreateCallback	XmCCallback	XmRCallback
XmNxrtGearNodeCurrent	XmCXrtGearNodeCurrent	XmRXrtGearWidget
XmNxrtGearNodeCurrentChanged Callback	XmCCallback	XmRCallback
XmNxrtGearNodeEnterCellCallback	XmCCallback	XmRCallback
XmNxrtGearNodeHeight	XmCXrtGearNodeHeight	XmRDimension

Name	Class	Type
XmNxrtGearNodeImportPolicy	XmCXrtGearNodeImportPolicy	XmRXrtGearImportPolicy
XmNxrtGearNodeIndent	XmCXrtGearNodeIndent	XmRInt
XmNxrtGearNodeIndentPolicy	XmCXrtGearNodeIndentPolicy	XmRXrtGearIndentPolicy
XmNxrtGearNodeMoveCallback	XmCCallback	XmRCallback
XmNxrtGearNodeMovePolicy	XmCXrtGearNodeMovePolicy	XmRXrtGearNodeMovePolicy
XmNxrtGearNodeSpacing	XmCXrtGearNodeSpacing	XmRDimension
XmNxrtGearNodeStyleDefault	XmCXrtGearNodeStyle	XmRXrtGearNodeStyle
XmNxrtGearNodeStyleList	XmCXrtGearNodeStyleList	XmRXrtGearNodeStyleList
XmNxrtGearNodeTop	XmCXrtGearNodeTop	XmRXrtGearWidget
XmNxrtGearNodeValidateCallback	XmCCallback	XmRCallback
XmNxrtGearNodeVisibleList	XmCXrtGearNodeVisibleList	XmRXrtGearList
XmNxrtGearPrintCallback	XmCCallback	XmRCallback
XmNxrtGearPSFontMapList	XmCXrtGearPSFontMapList	XmRXrtGearPSFontMapList
XmNxrtGearRepaint	XmCXrtGearRepaint	XmRBoolean
XmNxrtGearResizeCallback	XmCCallback	XmRCallback
XmNxrtGearScrollBarHoriz	XmCXrtGearScrollBarHoriz	XmRXrtGearWidget
XmNxrtGearScrollBarHorizDisplay Policy	XmCXrtGearScrollBarHorizDisplay Policy	XmRXrtGearDisplayPolicy
XmNxrtGearScrollBarVert	XmCXrtGearScrollBarVert	XmRXrtGearWidget
XmNxrtGearScrollBarVertDisplay Policy	XmCXrtGearScrollBarVertDisplay Policy	XmRXrtGearDisplayPolicy
XmNxrtGearSelectionCallback	XmCCallback	XmRCallback
XmNxrtGearSelectedNodeList	XmCXrtGearSelectedNodeList	XmRXrtGearList
XmNxrtGearSelectionPolicy	XmCXrtGearSelectionPolicy	XmRXrtGearSelectionPolicy
XmNxrtGearSelectionRestrictions	XmCXrtGearSelectionRestrictions	XmRXrtGearSelectionRestrictions
XmNxrtGearShadowThickness	XmCXrtGearShadowThickness	XmRDimension
XmNxrtGearTextEditor	XmCXrtGearTextEditor	XmRWidget
XmNxrtGearTextHorizClipPolicy	XmCXrtGearTextHorizClipPolicy	XmRXrtGearTextClipPolicy
XmNxrtGearTreeData	XmCXrtGearTreeData	XmRXrtGearTreeData
XmNxrtGearWidthChar	XmCXrtGearWidthChar	XmRChar
XmNxrtGearWidthPreferredCount	XmCXrtGearWidthPreferredCount	XmRInt
XmNxrtGearWidthResizePolicy	XmCXrtGearWidthResizePolicy	XmRXrtGearWidgetResizePolicy
XmNxrtGearWidthSizingPolicy	XmCXrtGearWidthSizingPolicy	XmRXrtGearSizingPolicy
XmNxrtGearWidthVirtual	XmCXrtGearWidthVirtual	XmRInt

B.6.1 列のリソース

Name	Class	Type
XmNfontList	XmCFontList	XmRFontList
XmNxrtGearFontListSelected	XmCXrtGearFontListSelected	XmRFontList
XmNforeground	XmCForeground	XmRPixel
XmNheight	XmCDimension	XmRDimension
XmNxrtGearLabel	XmCXrtGearString	XmRXrtGearString
XmNxrtGearLabelAlignment	XmCXrtGearAlignment	XmRXrtGearAlignment
XmNxrtGearMarginLeft	XmCXrtGearMarginLeft	XmRDimension
XmNxrtGearMarginRight	XmCXrtGearMarginRight	XmRDimension
XmNxrtGearNodeAlignment	XmCXrtGearAlignment	XmRXrtGearAlignment
XmNxrtGearResizeCallback	XmCCallback	XmRCallback
XmNxrtGearSelected	XmCXrtGearSelected	XmRBoolean
XmNwidth	XmCDimension	XmRDimension
XmNxrtGearWidthMaximum	XmCXrtGearWidthMaximum	XmRDimension
XmNxrtGearWidthMinimum	XmCXrtGearWidthMinimum	XmRDimension
XmNxrtGearWidthPreferredCount	XmCXrtGearWidthPreferredCount	XmRInt
XmNxrtGearWidthResizePolicy	XmCXrtGearWidthResizePolicy	XmRXrtGearColumnResizePolicy
XmNxrtGearWidthSizingPolicy	XmCXrtGearWidthSizingPolicy	XmRXrtGearSizingPolicy
XmNx	NULL	XmRPosition

B.6.2 ノードのリソース

Name	Class	Type
XmNEditable	XmCBoolean	XmRBoolean
XmNxrtGearLabel	XmCXrtGearLabel	XmRXrtGearLabel
XmNxrtGearNodeStyle	XmCXrtGearNodeStyle	XmRXrtGearNodeStyle
XmNxrtGearSelected	XmCXrtGearSelected	XmRBoolean
XmNuserData	XmCUserData	XmRPointer

B.6.3 ノード･フォルダのリソース

Name	Class	Type
XmNEditable	XmCBoolean	XmRBoolean

Name	Class	Type
XmNxrtGearFolderState	XmCXrtGearFolderState	XmRXrtGearFolderState
XmNxrtGearNodeChildList	XmCXrtGearNodeChildList	XmRXrtGearNodeChildList

B.6.4 ノード･スタイルのリソース

Name	Class	Type
XmNBackground	XmCBackground	XmRPixel
XmNxrtGearBackgroundSelected	XmCXrtGearBackgroundSelected	XmRPixel
XmNxrtGearFolderShortcutShow	XmCBoolean	XmRBoolean
XmNfontList	XmCFontList	XmRFontList
XmNxrtGearFontListSelected	XmCXrtGearFontListSelected	XmRFontList
XmNforeground	XmCForeground	XmRPixel
XmNxrtGearForegroundSelected	XmCXrtGearForegroundSelected	XmRPixel
XmNheight	XmCDimension	XmRDimension
XmNxrtGearIconClosed	XmCXrtGearIcon	XmRXrtGearIcon
XmNxrtGearIconClosedSelected	XmCXrtGearIcon	XmRXrtGearIcon
XmNxrtGearIconItem	XmCXrtGearIcon	XmRXrtGearIcon
XmNxrtGearIconItemSelected	XmCXrtGearIcon	XmRXrtGearIcon
XmNxrtGearIconOpen	XmCXrtGearIcon	XmRXrtGearIcon
XmNxrtGearIconOpenSelected	XmCXrtGearIcon	XmRXrtGearIcon
XmNxrtGearLayoutSpacing	XmCDimension	XmRDimension
XmNxrtGearLineColor	XmCXrtGearLineColor	XmRPixel
XmNxrtGearLineStyle	XmCXrtGearLineStyle	XmRXrtGearLineStyle
XmNxrtGearLineWidth	XmCXrtGearLineWidth	XmRDimension
XmNxrtGearShortcutShow	XmCBoolean	XmRBoolean
XmNxrtGearShortcutSize	XmCDimension	XmRDimension

B.7 Progress

Name	Class	Type
XmNxrtGearBarColor	XmCForeground	XmRPixel
XmNxrtGearBarCount	XmCXrtGearBarCount	XmRInt
XmNxrtGearBarSpacing	XmCXrtGearBarSpacing	XmRDimension
XmNxrtGearDoubleBuffer	XmCXrtGearDoubleBuffer	XmRBoolean
XmNfontList	XmCFontList	XmRFontList
XmNxrtGearIcon	XmCXrtGearIcon	XmRXrtGearIcon
XmNxrtGearLabel	XmCXrtGearLabel	XmRXrtGearLabel
XmNxrtGearLabelAlignment	XmCXrtGearLabelAlignment	XmRXrtGearAlignment
XmNxrtGearLabelFormat	XmCXrtGearLabelFormat	XmRString
XmNxrtGearLabelFormatCallback	XmCXrtGearLabelFormatCallback	XmRCallback
XmNxrtGearLabelShow	XmCXrtGearLabelShow	XmRBoolean
XmNmarginHeight	XmCMarginHeight	XmRDimension
XmNmarginWidth	XmCMarginWidth	XmRDimension
XmNmaximum	XmCXrtGearProgressMaximum	XmRInt
XmNminimum	XmCXrtGearProgressMinimum	XmRInt
XmNxrtGearProgressCallback	XmCXrtGearProgressCallback	XmRCallback
XmNxrtGearProgressTimeOut	XmCXrtGearProgressTimeOut	XmRXrtGearUlong
XmNxrtGearProgressType	XmCXrtGearProgressType	XmRXrtGearProgressType
XmNrecomputeSize	XmCRecomputeSize	XmRBoolean
XmNvalue	XmCXrtGearProgressValue	XmRInt
XmNxrtGearValuePercent	XmCXrtGearValuePercent	XmRXrtGearFloat

B.8 Tab Button

Name	Class	Type
XmNxrtGearAnchorSide	XmCXrtGearAnchorSide	XmRXrtGearAnchorSide
XmNxrtGearCornerSize	XmCXrtGearCornerSize	XmRDimension
XmNxrtGearLabelPixmap	XmCXrtGearLabelPixmap	XmRXrtGearPixmap
XmNxrtGearLabelType	XmCXrtGearLabelType	XmRXrtGearType
XmNxrtGearLayoutSpacing	XmCXrtGearLayoutSpacing	XmRDimension

Name	Class	Type
XmNxrtGearMask	XmCXrtGearMask	XmRXrtGearMask
XmNxrtGearPixmapRotation	XmCXrtGearPixmapRotation	XmRXrtGearRotation
XmNxrtGearStringLayout	XmCXrtGearStringLayout	XmRXrtGearStringLayout
XmNxrtGearStringRotation	XmCXrtGearStringRotation	XmRXrtGearRotation
XmNxrtGearStyle	XmCXrtGearStyle	XmRXrtGearStyle

B.9 Tab Manager

Name	Class	Type
XmNxrtGearActivePageNumber	XmCXrtGearActivePageNumber	XmRXrtGearActivePageNumber
XmNxrtGearActivePageWidget	XmCXrtGearActivePageWidget	XmRXrtWidget
XmNxrtGearActiveTabWidget	XmCXrtGearActiveTabWidget	XmRXrtWidget
XmNxrtGearArrowDisplayPolicy	XmCXrtGearArrowDisplayPolicy	XmRXrtGearDisplayPolicy
XmNxrtGearArrowHeight	XmCXrtGearArrowHeight	XmRDimension
XmNxrtGearArrowWidth	XmCXrtGearArrowWidth	XmRDimension
XmNxrtGearAttachPolicy	XmCXrtGearAttachPolicy	XmRXrtGearAttachPolicy
XmNxrtGearCornerSize	XmCXrtGearCornerSize	XmRDimension
XmNxrtGearFirstVisibleTab	XmCXrtGearFirstVisibleTab	XmRInt
XmNxrtGearHighlightThickness	XmCXrtGearHighlightThickness	XmRDimension
XmNxrtGearLastVisibleTab	XmCXrtGearLastVisibleTab	XmRInt
XmNxrtGearMarginHeight	XmCXrtGearMarginHeight	XmRDimension
XmNxrtGearMarginWidth	XmCXrtGearMarginWidth	XmRDimension
XmNxrtGearPageCallback	XmCCallback	XmRCallback
XmNxrtGearResizeCallback	XmCCallback	XmRCallback
XmNxrtGearShadowThickness	XmCXrtGearShadowThickness	XmRDimension
XmNxrtGearTabResize	XmCXrtGearTabResize	XmRBoolean
XmNxrtGearTabRotation	XmCXrtGearTabRotation	XmRXrtGearTabRotation
XmNxrtGearTabSide	XmCXrtGearTabSide	XmRXrtGearTabSide
XmNxrtGearTabSpacing	XmCXrtGearTabSpacing	XmRInt
XmNxrtGearTabStretch	XmCXrtGearTabStretch	XmRBoolean

B.9.1 Tab Manager Constraint リソース

Name	Class	Type
XmNxrtGearChildType	XmCXrtGearChildType	XmRXrtGearChildType
XmNxrtGearPageNumber	XmCXrtGearPageNumber	XmRInt
XmNxrtGearPageWidget	XmCXrtGearPageWidget	XmRXrtWidget
XmNxrtGearWidgetPosition	XmCXrtGearWidgetPosition	XmRInt

B.10 Toolbar

Name	Class	Type
XmNxrtGearMarginHeight	XmCXrtGearMarginHeight	XmRHorizontalDimension
XmNxrtGearMarginWidth	XmCXrtGearMarginWidth	XmRHorizontalDimension
XmNxrtGearOrientation	XmCXrtGearOrientation	XmROrientation

B.10.1 Toolbar Constraint リソース

Name	Class	Type
XmNxrtGearBottomSpace	XmCXrtGearBottomSpace	XmRHorizontalDimension
XmNxrtGearLeftSpace	XmCXrtGearLeftSpace	XmRHorizontalDimension
XmNxrtGearPosition	XmCXrtGearPosition	XmRInt
XmNxrtGearRightSpace	XmCXrtGearRightSpace	XmRHorizontalDimension
XmNxrtGearTopSpace	XmCXrtGearTopSpace	XmRHorizontalDimension

B.11 Widget Snapshot

Name	Class	Type
XmNxrtGearSnapNoShapeClip	XmCXrtGearSnapNoShapeClip	XmRBoolean

B.12 Widget Tips

Name	Class	Type
XmNxrtGearAlignment	XmCXrtGearAlignment	XmRAlignment
XmNxrtGearBackground	XmCXrtGearBackground	XmRPixel
XmNxrtGearDelay	XmCXrtGearDelay	XmRInt
XmNxrtGearDisplayCallback	XmCXrtGearDisplayCallback	XmRPointer
XmNxrtGearFontList	XmCXrtGearFontList	XmRFontList
XmNxrtGearForeground	XmCXrtGearForeground	XmRPixel

Name	Class	Type
XmNxrtGearGroup	XmCXrtGearGroup	XmRXrtGearGroup
XmNxrtGearShadowThickness	XmCXrtGearShadowThickness	XmRDimension
XmNxrtGearText	XmCXrtGearText	XmRXmString
XmNxrtGearTipsBorderType	XmCXrtGearTipsBorderType	XmRXrtGearBorderType

B.13 Paned Window

Name	Class	Type
XmNxrtGearMarginHeight	XmCXrtGearMarginHeight	XmRDimension
XmNxrtGearMarginWidth	XmCXrtGearMarginWidth	XmRDimension
XmNxrtGearOrientation	XmCOrientation	XmROrientation
XmNxrtGearPaneResizeCallback	XmCXrtGearPaneResizeCallback	XmRCallback
XmNxrtGearRecomputeSize	XmCXrtGearRecomputeSize	XmRBoolean
XmNxrtGearSashHeight	XmCXrtGearSashHeight	XmRDimension
XmNxrtGearSashIndent	XmCXrtGearSashIndent	XmRPosition
XmNxrtGearSashShadowThickness	XmCXrtGearSashShadowThickness	XmRDimension
XmNxrtGearSashWidth	XmCXrtGearSashWidth	XmRDimension
XmNxrtGearSeparatorShow	XmCBoolean	XmRBoolean
XmNxrtGearSpacing	XmCXrtGearSpacing	XmRDimension

B.13.1 Paned Window constraint リソース

Name	Class	Type
XmNxrtGearAllowResize	XmCBoolean	XmRBoolean
XmNxrtGearMinimum	XmCXrtGearMinimum	XmRDimension
XmNxrtGearMaximum	XmCXrtGearMaximum	XmRDimension
XmNxrtGearPosition	XmCXrtGearPosition	XmRInt
XmNxrtGearSkipAdjust	XmCBoolean	XmRBoolean

C

リソースファイルと GUI ビルダの使用

XRT/gear のリソース変換プログラムが認識できる、リソースファイルのストリングのフォーマットです。XRT/gear は、Xt と Motif に定義されていないデータ型のリソース変換のみを定義します。

値をセットする手続きを使用したリソース以外は、すべてリソースファイルでセットできます。

C.1 構文

コメント

(!)の後のテキストは無視されます。

継続

行の一番最後をバック・スラッシュ(\)にすると、次の行に継続します。

Boolean

大文字、小文字、大小混合文字の"`True`"または"`False`"。

列挙型

列挙型を用いるリソースは、列挙された値に基づく一貫した命名規則を使用します。全てのリソースでは、先頭の `XRTGEAR_`がありません。リソース値は、大文字、小文字または大小混合の文字で入力できます。例えば、次の各リソースの設定は、どれも、Tab Button ウィジェットで `XRTGEAR_STYLE_SQUARE` を `XmNxrtGearStyle` に使用されるようにしています。

```
*.XmXrtTabButton.xrtGearStyle: STYLE_SQUARE
*.XmXrtTabButton.xrtGearStyle: style_square
*.XmXrtTabButton.xrtGearStyle: StyLe_SquArE
```

D

UIL 引数

XRT/gear では、OSF の User Interface Language(UIL)を使用してウィジェットを作成することができます。UIL は、標準の Motif ウィジェットのデータ型のみをサポートしており、XRT/gear リソースの中には UIL でセットできないものもあります。これらは、`MrmNcreateCallback` を使用してアプリケーションでセットしてください。

ここでは、UIL 引数とデータ型、及び reasons がアルファベット順の一覧表にまとめてあります。

UILを使用した例のプログラムは、
$XRTHOME/src/gear/examples/simple_cpp.c (*simple_cpp.uilを使用*)と
$XRTHOME/src/gear/examples/complex_cpp.c (*complex_cpp.uilを使用*)にありますので参照してください。

D.1 Aligner のリソース

UIL Argument	Argument Type
XmNxrtGearAlignBaseline	boolean
XmNxrtGearAlignerAlignment	integer
XmNxrtGearColumnSpacing	integer
XmNxrtGearMarginHeight	integer
XmNxrtGearMarginWidth	integer
XmNxrtGearOrientation	integer
XmNxrtGearResizeHeight	boolean
XmNxrtGearResizeWidth	boolean
XmNxrtGearRowSpacing	integer

D.2 Combo Box のリソース

UIL Argument	Argument Type
XmNxrtGearAlignment	Integer
XmNxrtGearArrow[a]	Widget_ref
XmNxrtGearArrowType	Integer
XmNxrtGearAutoComplete	Boolean
XmNxrtGearMenuList[a]	Widget_ref
XmNxrtGearPickListIndex	integer
XmNxrtGearSearchPolicy	integer
XmNxrtGearTextField[a]	widget_ref

a: Get only

Reasons
XmNxrtGearMenuCallback

D.3 Enhanced Label のリソース

UIL Argument	Argument Type
XmNxrtGearBorderType	integer
XmNxrtGearLabelPixmap	icon
XmNxrtGearLabelType	integer
XmNxrtGearLayoutSpacing	integer
XmNxrtGearMask	pixmap
XmNxrtGearPixmapRotation	integer
XmNxrtGearStringLayout	integer
XmNxrtGearStringRotation	integer

D.4 Enhanced Pushbutton のリソース

UIL Argument	Argument Type
XmNxrtGearActive	Boolean
XmNxrtGearArmMask	pixmap
XmNxrtGearLabelPixmap	icon
XmNxrtGearLabelType	integer
XmNxrtGearLayoutSpacing	integer
XmNxrtGearMask	pixmap
XmNxrtGearPixmapRotation	integer
XmNxrtGearStringLayout	integer
XmNxrtGearStringRotation	integer

D.5 Enhanced Toggle のリソース

UIL Argument	Argument Type
XmNxrtGearIndicator	integer
XmNxrtGearNumStates	integer
XmNxrtGearSelectColor	color
XmNxrtGearState	integer
XmNxrtGearUnselectColor	color

Reasons
XmNxrtGearStateChangedCallback

D.6 Outliner のリソース

UIL Argument	Argument Type
XmNxrtGearAutoSort	boolean
XmNxrtGearBorderLocation	integer
XmNxrtGearColumnLabelsShow	boolean
XmNxrtGearColumnMoveAllow	boolean
XmNxrtGearColumnSelected	widget_ref

UIL Argument	Argument Type
XmNxrtGearCursorTracking	boolean
XmNxrtGearDoubleBuffer	boolean
XmNxrtGearEditor	widget
XmNxrtGearEditNode	widget
XmNxrtGearEditColumn	integer
XmNxrtGearFolderStateMask	integer
XmNxrtGearHeightPreferredCount	integer
XmNxrtGearHeightResizePolicy	integer
XmNxrtGearHeightSizingPolicy	integer
XmNxrtGearHeightVirtual[a]	integer
XmNxrtGearHighlightOnDrag	boolean
XmNxrtGearNodeCheckForDuplicates	boolean
XmNxrtGearNodeCurrent	widget_ref
XmNxrtGearNodeHeight	integer
XmNxrtGearNodeImportPolicy	integer
XmNxrtGearNodeIndent	integer
XmNxrtGearNodeMovePolicy	integer
XmNxrtGearNodeSpacing	integer
XmNxrtGearNodeStyleDefault	widget_ref
XmNxrtGearNodeTop	widget_ref
XmNxrtGearRepaint	boolean
XmNxrtGearScrollBarHoriz	widget_ref
XmNxrtGearScrollBarVert	widget_ref
XmNxrtGearSelectionPolicy	integer
XmNxrtGearSelectionRestrictions	integer
XmNxrtGearTextEditor	widget
XmNxrtGearTextHorizClipPolicy	integer
XmNxrtGearTreeData	string
XmNxrtGearWidthPreferredCount	integer
XmNxrtGearWidthResizePolicy	integer
XmNxrtGearWidthSizingPolicy	integer
XmNxrtGearWidthVirtual[a]	integer

a: Get only

Reasons
XmNxrtGearActivateCallback
XmNxrtGearColumnMoveCallback
XmNxrtGearFolderStateCallback
XmNxrtGearNodeCreateCallback
XmNxrtGearNodeEnterCellCallback
XmNxrtGearNodeImportCallback
XmNxrtGearNodeMoveCallback
XmNxrtGearNodeValidateCallback
XmNxrtGearPrintCallback
XmNxrtGearSelectionCallback

D.6.1 XrtNode のリソース

UIL Argument	Argument Type
XmNeditable	boolean
XmNxrtGearNodeStyle[a]	widget_ref
XmNxrtGearSelecteda	boolean

a: Get only

D.6.2 XrtFolderNode のリソース

UIL Argument	Argument Type
XmNeditable	boolean
XmNxrtGearFolderState	integer

D.6.3 XrtColumn のリソース

UIL Argument	Argument Type
XmNbackground	pixel
XmNxrtGearFontListSelected	font_table
XmNxrtGearLabelAlignment	integer
XmNxrtGearMarginLeft	integer

UIL Argument	Argument Type
XmNxrtGearMarginRight	integer
XmNxrtGearNodeAlignment	integer
XmNxrtGearWidthMaximum	integer
XmNxrtGearWidthMinimum	integer

D.6.4 XrtNodeStyle のリソース

UIL Argument	Argument Type
XmNbackground	pixel
XmNxrtGearBackgroundSelected	color
XmNxrtGearFolderShortcutShow	boolean
XmNxrtGearForegroundSelected	color
XmNxrtGearIconClosed	icon
XmNxrtGearIconClosedSelected	icon
XmNxrtGearIconItem	icon
XmNxrtGearIconItemSelected	icon
XmNxrtGearIconOpen	icon
XmNxrtGearIconOpenSelected	icon
XmNxrtGearLineColor	color
XmNxrtGearLineStyle	integer
XmNxrtGearLineWidth	integer
XmNxrtGearShortcutColor	pixel
XmNxrtGearShortcutShow	boolean
XmNxrtGearShortcutSize	integer

D.7 Progress のリソース

UIL Argument	Argument Type
XmNxrtGearBarColor	color
XmNxrtGearIcon	icon
XmNxrtGearLabelFormat	string
XmNxrtGearLabelShow	boolean
XmNxrtGearProgressTimeOut	integer
XmNxrtGearProgressType	integer

Reasons
XmNxrtGearLabelFormatCallback
XmNxrtGearProgressCallback

D.8 Tab Button のリソース

UIL Argument	Argument Type
XmNxrtGearAnchorSide	integer
XmNxrtGearCornerSize	integer
XmNxrtGearLabelPixmap	icon
XmNxrtGearLabelType	integer
XmNxrtGearLayoutSpacing	integer
XmNxrtGearMask	pixmap
XmNxrtGearPixmapRotation	integer
XmNxrtGearStringLayout	integer
XmNxrtGearStringRotation	integer
XmNxrtGearStyle	integer

D.9 Tab Manager のリソース

UIL Argument	Argument Type
XmNxrtGearActivePageNumber	integer
XmNxrtGearActivePageWidget	integer
XmNxrtGearArrowDisplayPolicy	integer
XmNxrtGearArrowHeight	integer
XmNxrtGearArrowWidth	integer
XmNxrtGearChildType	integer
XmNxrtGearFirstVisibleTab	integer
XmNxrtGearLastVisibleTab	integer
XmNxrtGearPageAttachPolicy	integer
XmNxrtGearPageNumber	integer
XmNxrtGearPageWidget	widget_ref

UIL Argument	Argument Type
XmNxrtGearShadowThickness	integer
XmNxrtGearTabResize	boolean
XmNxrtGearTabRotation	integer
XmNxrtGearTabSide	integer
XmNxrtGearTabSpacing	integer
XmNxrtGearTabStretch	boolean
XmNxrtGearWidgetPosition	integer

Reasons
XmNxrtGearPageCallback
XmNxrtGearResizeCallback

D.10 Toolbar のリソース

UIL Argument	Argument Type
XmNxrtGearBottomSpace	integer
XmNxrtGearLeftSpace	integer
XmNxrtGearPosition	integer
XmNxrtGearRightSpace	integer
XmNxrtGearTopSpace	integer

D.11 定数の定義

下記の定数は、UIL の 31 文字の制限に合うように短くなっています。

- `XRTGEAR_UNSPECIFIED_ACTIVE_PAGE_NUMBER` は、`XRTGEAR_UNSPECIFIED_ACTIVE_PAGE`
- `XRTGEAR_UNSPECIFIED_PAGE_POSITION` は、`XRTGEAR_UNSPECIFIED_PAGE_POS`
- `XRTGEAR_GLABEL_LAYOUT_SPACING_DEFAULT` は、`XRTGEAR_GLABEL_LAYOUT_SPACING`

D.12 Paned Window のリソース

UIL Argument	Argument Type
XmNxrtGearMarginHeight	Dimension
XmNxrtGearMarginWidth	Dimension
XmNxrtGearOrientation	unsigned char
XmNxrtGearRecomputeSize	Boolean
XmNxrtGearSashHeight	Dimension
XmNxrtGearSashIndent	Position
XmNxrtGearSashShadowThickness	Dimension
XmNxrtGearSashWidth	Dimension
XmNxrtGearSeparatorShow	Boolean
XmNxrtGearSpacing	Dimension
XmNxrtGearAllowResize	Boolean
XmNxrtGearMinimum	Dimension
XmNxrtGearMaximum	Dimension
XmNxrtGearPositionIndex	short
XmNxrtGearSkipAdjust	Boolean

Reasons
XmNxrtGearPaneResizeCallback

E

サンプル・コード

XRT/gear には、幾つかの例やデモのプログラムが提供されています。例は、特定な機能を使用した短いプログラムです。デモは、複数の機能を使用した、アプリケーションです。XRT/gear の基本的なプログラミングの大半は、これらの例やデモプログラムで紹介されています。

E.1 例

例のプログラムは、*$XRTHOME/src/gear/examples* にあります。

aligner.c	Aligner ウィジェットを表示します。
combo.c	Combo Box ウィジェットを表示します。
complex_cpp.cxx	XRT/gear ウィジェットを作成し、リソースを操作します。C++ で書かれています。
label.c	Enhanced Label ウィジェットを表示します。
multicol.c	ラベル付きの複数の列を持った Outliner ウィジェットを表示します。
outliner.c	指定したファイルのデータを表示する Outliner ウィジェットを作成します。
paned.c	Paned window ウィジェットとカラー・エディタ・オブジェクトを表示します。
progress.c	Progress ウィジェットを表示します。
pushb.c	Enhanced Pushbutton ウィジェットを表示します。
racecar.c	アイコンを使用した Progress ウィジェットを表示します。
simple_cpp.cxx	XRT/gear ウィジェットを作成します。C++で書かれています。
tabs.c	Tab Manager と Tab Button ウィジェットを表示します。
toggle.c	Enhanced Toggle ウィジェットを表示します。
toolbar.c	Toolbar ウィジェットを表示します。

E.2 デモ

デモ・アプリケーションは、*$XRTHOME/src/gear/demos* にあります。各デモごとにディレクトリがあります。

Fileviewer	Outliner ウィジェットを使用して、御使用のシステムのファイル階層を表示し、変更することができます。
Sampler	複数の XRT/gear を使用したアプリケーションです。

F

Windows 95 のシミュレーション

XRT/gear ウィジェットを Microsoft Windows 95 のコンポーネントと同じように表示する場合のヒントを提供しています。

XRT/gear 全体

- MS Sans Serif フォントが使用できる場合は、このフォントを使用する。使用できない場合は、7-point Arial または 9-point Times New Roman のような、類似したフォントを使用する。

Combo Box

- XmNxrtGearArrowType をデフォルト値の XRTGEAR_ARROW_WEDGE にセットする。
- XmNxrtGearAlignment をデフォルト値の XRTGEAR_MENU_ALIGN_BOTH にセットする。

Enhanced Toggle

- XmNxrtGearIndicator を XRTGEAR_INDICATOR_CHECK にセットする。

Outliner

- XmNxrrGearShortcutSize を 7 にセットする。
- XmNxrtGearBorderType を XRTGEAR_BORDER_FRAME_OUT にセットする。
- XmNxrtGearShadowThickness を 1 にセットする。
- XmNxrtGearBorderLocation をデフォルト値の XRTGEAR_BORDER_EACH_LABEL_SEPARATED_FROM_NODES にセットする。
- XmNbottomShadowColor を black にセットする。この色と frame out の枠線によって Outliner の枠が Windows Explorer に使用されている枠と類似します。スクロールバーを変更するには、XmNxrtGearScrollBarHoriz と XmNxrtGearScrollBarVert の値を取得してウィジェット ID を取り出してください。
- NodeStyle オブジェクトの XmNxrtGearLineStyle は、デフォルト値 XRTGEAR_LINESTYLE_DOTTED を使用する。

- NodeStyle オブジェクトの XmNxrtGearIconOpen を、(XmNxrtGearIconClosed と XmNxrtGearIconClosedSelected で使用される)閉じたフォルダにセットする。Windows 95 では、子が表示されている、いないに関わらず、閉じたフォルダ・アイコンが非選択状態のフォルダを示します。

Progress

- XmNxrtGearProgressType を XRTGEAR_PROGRESS_DISCRETE_BAR にセットする。

Tab Manager

- XmNxrtGearStyle をデフォルト値の XRTGEAR_STYLE_ROUNDED にセットする。
- XmNxrtGearCornerSize を 2 にセットする。

索引

A

Activate(), 71, 209
Aligner, 7, 81
  constraint リソース, 85, 87
  XmSeparator, 83
  位置, 84
  継承されるリソース, 89
  子ウィジェットの位置, 85
  サンプル・プログラム, 82
  手続きと方法, 89
  マージン, 83
  ラベルの位置, 85
  リサイズ, 85
  リソースの一覧, 87
arm pixmap, 20
ArmAndActivate(), 71

C

C++, 11
Cancel(), 255
CancelEdit(), 209
ChangeState(), 209
clip マスク, 17, 34, 53, 132
Color Editor, 8
Combo Box, 8, 241
  pick list, 243
  pick list のインデックス, 243
  pick list の検索, 249
  text field, 242
  text field の修正, 244
  継承されるリソース, 253
  サンプル・プログラム, 241
  自動マッチング, 250
  手続きと方法, 254
  トランスレーション, 255
  メニューの位置, 248
  矢印, 245
  ユーザ・インタラクション, 248
  リソースの一覧, 250
CommitEdit(), 209

D

Delete(), 210
Disarm(), 71
Drag(), 210

E

Edit(), 210
Enhanced Label, 7, 15
  clip マスク, 17
  pixmap, 17
  回転, 18
  型, 17
  継承されるリソース, 26
  サンプル・プログラム, 15
  テキストと pixmap, 19
  手続きと方法, 28
  リソースの一覧, 21
  枠, 19
Enhanced Pushbutton, 7, 15
  arm pixmap, 20
  clip マスク, 17
  pixmap, 17
  回転, 18
  型, 17
  継承されるリソース, 27
  サンプル・プログラム, 16
  テキストと pixmap, 19
  手続きと方法, 28
  リソースの一覧, 23
Enhanced Toggle, 7, 31
  clip pixmap, 34
  継承されるリソース, 39
  サンプル・プログラム, 31
  手続きと方法, 41
  複数の状態, 34
  ボタンの色, 35
  ボタンの状態, 32
  リソースの一覧, 35

G

GoTo(), 210
GUI ビルダとの統合, 11

H

Help(), 71

I

Icon
　手続きと方法, 319
Icon Library, 8

M

Menu(), 255
MultiActivate(), 71
MultiArm(), 72

N

NoAction(), 211

O

Outliner, 7, 91
　Lightweight フォルダ, 212
　NodeFolder オブジェクト, 95
　Node オブジェクト, 95
　PostScript 出力, 146
　sibling, 107
　XrtColumn, 120
　　ラベル, 123
　XrtGearIcon, 96
　XrtList, 118, 122, 129
　XrtList オブジェクト, 96
　XrtString, 118, 129
　XrtString オブジェクト, 96
　アイテム, 92, 95
　移動のコールバック, 107
　イベントの位置, 144
　色とフォント, 113
　カーソル, 117
　カレント・ノード, 104
　基本概念, 95
　兄弟, 107
　子のリスト, 104
　サイズ, 110
　ショートカット・ボタン, 92, 134
　スクロールバー, 141
　接続線, 133
　選択のコールバック, 140
　ソート, 108
　ツリーの作成, 93, 98
　ツリーの直接作成, 101
　ツリーのトラバース, 105
　テキストの切り取り, 115
　データ・ファイル, 98
　データ・ファイル
　　特別文字, 98
　　リソース・ファイル, 100
　手続きと方法, 178
　ドラッグ&ドロップ, 141
　トランスレーション, 198
　ノード, 95
　ノード・スタイル, 129
　ノードの移動, 107
　ノードのインデント, 113
　ノードの間隔, 113
　ノードの選択, 136
　ノードの高さ, 115
　ノードの表示, 114
　ノード・ラベル, 95
　ノード領域, 95
　パフォーマンス, 150
　フォーカス・ハイライト, 115
　フォルダ, 92
　フォルダ・アイコン, 131
　フォルダの状態, 139
　フォルダ・ノード, 95
　複数の列, 118
　保管, 102
　　バッファ, 103
　　ファイル, 102
　マージン, 112
　ユーザ・インタラクション, 135
　リサイズ, 116
　リソースの一覧, 151
　列の移動, 145
　列の作成, 120
　列の順序, 121

列の選択, 122
列の表示のコントロール, 124
列のリサイズ, 128
枠, 126
手続きと方法, 178

P

PanedWindow, 8, 259
ユーザ・インタラクション, 261
PostScript, 146
Progress, 8, 223
ウィジェットの破棄, 228
継承されるリソース, 238
サイズの再計算, 231
サンプルプログラム, 223
タイムアウト, 226
手続きと方法, 238
バーのアイコン, 233
バーの色, 233
バーの間隔, 233
バーの種類, 231
マージン, 232
ラベル, 228
ラベルの色, 231
ラベルの行揃え, 231
ラベルの範囲, 225
ラベルの表示, 229
ラベルのフォーマット, 229
ラベルのフォント, 230
リソースの一覧, 233, 262

R

ResizeColumn(), 211

S

Select(), 211

T

Tab Button, 7, 43
clip マスク, 53
pixmap, 53
アクション, 71
色, 57
回転, 56
形, 54
角のサイズ, 57
継承されるリソース, 66
テキストと pixmap の配置, 55
手続きと方法, 68
トランスレーション, 71
ボタンの型, 52
マージン, 56
リソースの一覧, 59
Tab Manager, 7, 43
constraint リソース, 52, 65
継承されるリソース, 68
サンプル・プログラム, 44
タブ間のスペース, 49
タブの位置, 48
タブの回転, 50
タブのストレッチ, 50
タブの表示, 49
タブのリサイズ, 49
手続きと方法, 68
ハイライト, 50
ページの振り分け, 47
ページの変更, 46
マージン, 50
矢印ボタン, 51
リサイズ, 58
リソースの一覧, 61
Toolbar, 7, 73
constraint リソース, 75, 77
継承されるリソース, 78
サンプル・プログラム, 73
手続きと方法, 78
ヘルプ・バルーン, 76
方向, 74
マージン, 75
リソースの一覧, 76

U

UIL 引数, 353

W

Widget Snapshot, 8, 283
pixmap の取得, 283

PostScript への変換, 284
XImage の取得, 285
手続きと方法, 286
リソースの一覧, 285
Widget Tip, 8, 269
help display group, 273
ウィジェットへの追加, 270
手続きと方法, 278
トランスレーション, 276
表示のコールバック, 272
表示のタイミング, 273
不可に設定, 274
ヘルプの削除, 275
ヘルプの表示, 275
ヘルプ・バルーン, 271
リソースの一覧, 276

X

XmCreateXrtAligner(), 89
XmCreateXrtColumn(), 197
XmCreateXrtGearComboBox(), 254
XmCreateXrtGLabel(), 28
XmCreateXrtNode(), 185
XmCreateXrtNodeFolder(), 186
XmCreatexrtNodeStyle(), 196
XmCreateXrtNodeStyle(), 129
XmCreateXrtOutliner(), 178
XmCreateXrtPanedWindow(), 265
XmCreateXrtProgress(), 238
XmCreateXrtProgressShellSimple(), 228, 239
XmCreateXrtPushButton(), 28
XmCreateXrtTabButton(), 68
XmCreateXrtTabManager(), 69
XmCreateXrtToggleButton(), 41
XmCreateXrtToolbar(), 78
XmIsXrtAligner(), 90
XmIsXrtColumn(), 197
XmIsXrtGearComboBox(), 254
XmIsxrtGLabel(), 28
XmIsXrtNode(), 186
XmIsXrtNodeFolder(), 186
XmIsXrtNodeStyle(), 196
XmIsXrtOutliner(), 178
XmIsXrtPanedWindow(), 266
XmIsXrtProgress(), 239
XmIsXrtPushButton(), 28
XmIsXrtTabButton(), 69
XmIsXrtTabManager(), 69
XmIsXrtToggleButton(), 41
XmIsXrtToolbar(), 79
XmList, 243
XmNarmCallback, 36
XmNbackground, 21, 24, 36, 59, 151, 171, 173
XmNdisarmCallback, 36
XmNeditable, 155, 170, 173
XmNfontList, 22, 24, 36, 59, 157, 172, 174, 234
XmNforeground, 22, 24, 36, 59, 157, 172, 174
XmNheight, 22, 24, 37, 60, 157, 172, 174
XmNhighlightColor, 158
XmNhighlightThickness, 63, 115, 158
XmNlabelString, 22, 24, 37, 60
XmNmarginHeight, 22, 25, 38, 60, 235
XmNmarginWidth, 22, 25, 38, 60
XmNmarginWidtht, 235
XmNmaximum, 235
XmNminimum, 235
XmNrecomputeSize, 237
XmNshadowThickness, 19, 23
XmNuserData, 23, 25, 39, 61, 171
XmNvalue, 237
XmNwidth, 23, 25, 39, 61, 169, 175
XmNx, 23, 25, 39, 61, 176
XmNxrtGearActivateCallback, 144, 151
XmNxrtGearActive, 20, 23
XmNxrtGearActivePageNumber, 61
XmNxrtGearActivePageWidget, 61
XmNxrtGearActiveTabWidget, 62
XmNxrtGearAlignBaseline, 88
XmNxrtGearAlignBaseLine, 85
XmNxrtGearAlignerAlignment, 85, 87
XmNxrtGearAlignment, 248, 250, 276
XmNxrtGearAllowResize, 264
XmNxrtGearAnchorSide, 53, 59
XmNxrtGearArmMask, 24
XmNxrtGearArmPixmap, 20, 24
XmNxrtGearArrow, 251
XmNxrtGearArrowClass, 251
XmNxrtGearArrowDisplayPolicy, 51, 62
XmNxrtGearArrowHeight, 62
XmNxrtGearArrowType, 245, 246, 251
XmNxrtGearArrowWidth, 62
XmNxrtGearAttachPolicy, 62
XmNxrtGearAutoComplete, 250
XmNxrtGearAutoSort, 151
XmNxrtGearBackground, 277
XmNxrtGearBackgroundSelected, 171
XmNxrtGearBarColor, 233
XmNxrtGearBarCount, 232, 234
XmNxrtGearBarSpacing, 233, 234
XmNxrtGearBoarderLocation, 126
XmNxrtGearBorderLocation, 151

XmNxrtGearBorderType, 19, 21, 152
XmNxrtGearBottomSpace, 77
XmNxrtGearChildType, 65
XmNxrtGearColumnLabelsShow, 123, 152
XmNxrtGearColumnList, 120, 152
XmNxrtGearColumnMoveAllow, 152
XmNxrtGearColumnMoveCallback, 146, 153
XmNxrtGearColumnOrderList, 122, 153
XmNxrtGearColumnSelected, 122, 153
XmNxrtGearColumnSelectionCallback, 153
XmNxrtGearColumnSpacing, 83, 87
XmNxrtGearCornerSize, 54, 57, 59, 62
XmNxrtGearCursor, 154
XmNxrtGearCursorResizeWidth, 154
XmNxrtGearCursorTrackigCallback, 154
XmNxrtGearCursorTracking, 154
XmNxrtGearCursorTrackingCallback, 117
XmNxrtGearDelay, 273, 277
XmNxrtGearDisplayCallback, 272, 277
XmNxrtGearDoubleBuffer, 150, 155, 234
XmNxrtGearEditColumn, 155
XmNxrtGearEditNode, 155
XmNxrtGearEditor, 156
XmNxrtGearFirstVisibleTab, 49, 63
XmNxrtGearFolderShortcutShow, 172
XmNxrtGearFolderState, 171
XmNxrtGearFolderStateCallback, 156
XmNxrtGearFolderStateMask, 139, 156
XmNxrtGearFontList, 271, 277
XmNxrtGearFontListSelected, 124, 172, 174
XmNxrtGearForeground, 277
XmNxrtGearForegroundSelected, 172
XmNxrtGearGroup, 274, 277
XmNxrtGearHeightPreferredCount, 111, 157
XmNxrtGearHeightResizePolicy, 116, 157
XmNxrtGearHeightSizingPolicy, 111, 157
XmNxrtGearheightVirtual, 158
XmNxrtGearHeightVirtual, 111
XmNxrtGearHighlightThickness, 50
XmNxrtGearIcon, 233
XmNXrtGearIcon, 234
XmNxrtGearIconClosed, 172
XmNxrtGearIconClosedSelected, 172
XmNxrtGearIconItem, 172
XmNxrtGearIconItemSelected, 172
XmNxrtGearIconOpen, 172
XmNxrtGearIconOpenSelected, 173
XmNxrtGearImportCallback, 144, 158
XmNxrtGearIndicator, 33, 37
XmNxrtGearIndicatorMaskList, 37
XmNxrtGearIndicatorPixmapList, 33, 37
XmNxrtGearLabel, 96, 118, 129, 171, 174, 234
XmNxrtGearLabelAlignment, 174, 234
XmNxrtGearLabelFormat, 229, 234
XmNxrtGearlabelFormatCallback, 230
XmNxrtGearLabelFormatCallback, 235
XmNxrtGearLabelPixmap, 17, 22, 24, 60
XmNxrtGearlabelShow, 229
XmNxrtGearLabelShow, 235
XmNxrtGearLabelType, 17, 22, 24, 52, 60
XmNxrtGearLastVisibleTab, 63
XmNxrtGearLayoutSpacing, 19, 22, 24, 55, 60, 132, 173
XmNxrtGearLeftSpace, 77
XmNxrtGearLineColor, 173
XmNxrtGearLineStyle, 173
XmNxrtGearLineWidth, 173
XmNxrtGearMarginHeight, 63, 75, 77, 83, 87, 159, 262
XmNxrtGearMarginLeft, 126, 174
XmNxrtGearMarginRight, 126, 174
XmNxrtGearMarginWidth, 50, 63, 75, 77, 83, 87, 159, 262
XmNxrtGearMask, 17, 23, 25, 53
XmNXrtGearMask, 60
XmNxrtGearMaximum, 264
XmNxrtGearMenuCallback, 245, 251
XmNxrtGearMenuList, 252
XmNxrtGearMinimum, 264
XmNxrtGearNodeAlignment, 124, 174
XmNxrtGearNodeCheckForDuplicates, 159
XmNxrtGearNodeChildList, 104, 159, 171
XmNxrtGearNodeCompareProc, 108, 159
XmNxrtGearNodeCreateCallback, 159
XmNxrtGearNodeCurrent, 104, 160
XmNxrtGearNodeCurrentChangedCallback, 104, 160
XmNxrtGearNodeEnterCellCallback, 161
XmNxrtGearNodeHeight, 115, 162
XmNxrtGearNodeImportPolicy, 142, 162
XmNxrtGearNodeIndent, 113, 162
XmNxrtGearNodeMoveCallback, 162
XmNxrtGearNodeMovePolicy, 141, 163
XmNxrtGearNodeSpacing, 113, 163
XmNxrtGearNodeStyle, 130, 171
XmNxrtGearNodeStyleDefault, 163
XmNxrtGearNodeStyleList, 163
XmNxrtGearNodeTop, 114, 163
XmNxrtGearNodeValidateCallback, 164
XmNxrtGearNodeVisibleItemCount, 252
XmNxrtGearNodeVisibleList, 114, 164
XmNxrtGearNumStates, 34, 38
XmNxrtGearOrientation, 74, 77, 84, 87, 262
XmNxrtGearPageCallback, 46, 63

XmNxrtGearPageNumber, 46, 66
XmNxrtGearPageWidget, 66
XmNxrtGearPaneResizeCallback, 263
XmNxrtGearPickList, 242, 243, 252
XmNxrtGearPickListIndex, 243, 252
XmNxrtGearPixmapRotation, 18, 23, 25, 60
XmNxrtGearPosition, 76, 77, 85, 88, 264
XmNxrtGearPrintCallback, 148, 164
XmNxrtGearProgressCallback, 226, 236
XmNxrtGearProgressTimeOut, 226, 236
XmNxrtGearProgressType, 232, 237
XmNxrtGearPSFontMapList, 147, 165
XmNxrtGearRecomputeSize, 263
XmNxrtGearRepaint, 150, 165
XmNxrtGearResizeCallback, 64, 116, 128, 166, 174
XmNxrtGearResizeHeight, 88
XmNxrtGearResizePolicy, 128
XmNxrtGearResizeWidth, 88
XmNxrtGearRightSpace, 77
XmNxrtGearRowSpacing, 83, 87
XmNxrtGearSashHeight, 263
XmNxrtGearSashIndent, 263
XmNxrtGearSashShadowThickness, 263
XmNxrtGearSashWidth, 263
XmNxrtGearScrollBarHoriz, 166
XmNxrtGearScrollBarHorizDisplayPolicy, 141, 166
XmNxrtGearScrollBarVert, 166
XmNxrtGearScrollBarVertDisplayPolicy, 141, 166
XmNxrtGearSearchPolicy, 244, 249, 252
XmNxrtGearSelecitonPolicy, 167
XmNxrtGearSelectColor, 35, 38
XmNxrtGearSelected, 171, 175
XmNxrtGearSelectedNodeList, 140, 167
XmNxrtGearSelectionCallback, 140, 167
XmNxrtGearSelectionPolicy, 136, 141, 198
XmNxrtGearSelectionRestrictions, 136, 168
XmNxrtGearSeparatorShow, 263
XmNxrtGearShadowThickenss, 64
XmNxrtGearShadowThickness, 168, 278
XmNxrtGearShortcutShow, 134, 173
XmNxrtGearShortcutSize, 134, 173
XmNxrtGearSkipAdjust, 264
XmNxrtGearSnapNoShapeClip, 286
XmNxrtGearSpacing, 248, 252, 264
XmNxrtGearState, 32, 38
XmNxrtGearStateChangedCallback, 34, 38
XmNxrtGearStringLayout, 19, 23, 25, 55, 61
XmNxrtGearStringRotation, 18, 23, 25, 56, 61
XmNxrtGearStyle, 54, 61
XmNxrtGearTabResize, 64
XmNxrtGearTabRotation, 50, 65
XmNxrtGearTabSide, 48, 65
XmNxrtGearTabSpacing, 49
XmNxrtGearTabStretch, 50
XmNxrtGearText, 278
XmNxrtGearTextEditor, 168
XmNxrtGearTextField, 252
XmNxrtGearTextHorizClipPolicy, 115, 169
XmNxrtGearTipsBorderType, 271, 278
XmNxrtGearTopSpace, 77
XmNxrtGearTreeData, 100, 169
XmNxrtGearUnselectColor, 35, 38
XmNxrtGearValuePercent, 237
XmNxrtGearWidgetPosition, 66
XmNxrtGearWidthChar, 169
XmNxrtGearWidthMaximum, 128, 175
XmNxrtGearWidthMinimum, 128, 175
XmNxrtGearWidthPreferredCount, 125, 170, 175
XmNxrtGearWidthResizePolicy, 116, 170, 175
XmNxrtGearWidthSizingPolicy, 125, 170, 176
XmNxrtGearWidthVirtual, 111, 170
XmNxrtGetGearMenuList, 243
XmNy, 23, 25, 39, 61
XPM フォーマット, 132
xrt_auth, 10
XrtArm(), 72
XrtColorEditor, 300
XrtColorEditorChildType, 325
XrtGearAction, 323
XrtGearAlignerAlignment, 323
XrtGearAlignment, 323
XrtGearAnchorSide, 324
XrtGearArrowType, 324
XrtGearAttachPolicy, 324
XrtGearBorderLocation, 324
XrtGearBorderType, 324
XrtGearChildType, 325
XrtGearCmpDoubles(), 301
XrtGearCmpStringIgnoreCase(), 302
XrtGearCmpStrings(), 302
XrtGearColorEditorAddCancelCallback(), 318
XrtGearColorEditorAddOKCallback(), 318
XrtGearColorEditorAllocateColor(), 318
XrtGearColorEditorCreate(), 313
XrtGearColorEditorDestroy(), 313
XrtGearColorEditorGetChild(), 314
XrtGearColorEditorGetColorName(), 316
XrtGearColorEditorGetEditTypes(), 318
XrtGearColorEditorGetHSBLabels(), 317
XrtGearColorEditorGetHSBValues(), 317

XrtGearColorEditorGetName(), 314
XrtGearColorEditorGetRGBLabels(), 316
XrtGearColorEditorGetRGBValues(), 317
XrtGearColorEditorHide(), 314
XrtGearColorEditorIsShowing(), 314
XrtGearColorEditorSetCloseColorProc(), 318
XrtGearColorEditorSetColorList(), 315
XrtGearColorEditorSetColorName(), 315
XrtGearColorEditorSetEditTypes(), 317
XrtGearColorEditorSetHSBLabels(), 316
XrtGearColorEditorSetHSBValues(), 316
XrtGearColorEditorSetRGBLabels(), 315
XrtGearColorEditorSetRGBValues(), 315
XrtGearColorEditorShow(), 314
XrtGearColumnFromXEvent(), 145, 197
XrtGearColumnFromXY(), 197
XrtGearColumnMoveCallbackStruct, 325
XrtGearColumnResizePolicy, 325
XrtGearColumnSelectionCallbackStruct, 325
XrtGearCompFloats(), 301
XrtGearCompInts(), 301
XrtGearCursorTrackingCallbackStruct, 326
XrtGearCvtStringToXmString(), 178, 310
XrtGearCvtXmStringToString(), 310
XrtGearCvtXmStringToStrint(), 178
XrtGearDirection, 326
XrtGearDisplayCallbackStruct, 326
XrtGearDisplayPolicy, 326
XrtGearEditType, 326
XrtGearEnhancedSizeCallbackStruct, 327
XrtGearFolderCreateLW(), 219
XrtGearFolderState, 327
XrtGearIcon, 132, 300, 327
XrtGearIconGetSize(), 319
XrtGearIconLoadFromXpmData(), 132, 319
XrtGearIconLoadFromXpmFile(), 132, 319
XrtGearImportPolicy, 327
XrtGearIndentPolicy, 327
XrtGearIndicator, 328
XrtGearIsColorEditor(), 313
XrtGearIsNode(), 215
XrtGearIsNodeFolder(), 219
XrtGearLineStyle, 328
XrtGearListAdd(), 302
XrtGearListAppend(), 96, 291, 302
XrtGearListChangedCallbackStruct, 328
XrtGearListCreate(), 96, 303
XrtGearListDelete(), 303
XrtGearListDeleteAll(), 303
XrtGearListDestroy(), 303
XrtGearListDestroyObject(), 292, 303
XrtGearListDestroyXtPointer(), 304
XrtGearListDuplicate(), 304
XrtGearListFind(), 293, 304
XrtGearListGetCompareProc(), 304
XrtGearListGetItem(), 293, 305
XrtGearListGetItemCount(), 294, 305
XrtGearListGetItems(), 305
XrtGearListGetItemSize(), 305
XrtGearListGetUserData(), 297, 305
XrtGearListInsertAfter(), 306
XrtGearListInsertBefore(), 291, 306
XrtGearListIsList(), 306
XrtGearListIsSorted(), 306
XrtGearListLookupItem(), 293, 307
XrtGearListMoveItemsAfter(), 307
XrtGearListMoveItemsBefore(), 296
XrtGearListMOveItemsBefore(), 307
XrtGearListRemove(), 307
XrtGearListRemoveAll(), 308
XrtGearlistSearchBackward(), 294
XrtGearListSearchForward(), 294, 308
XrtGearListSearchReverse(), 308
XrtGearListSetCompareProc(), 296, 309
XrtGearListSetDestroyProc(), 309
XrtGearListSetItem(), 309
XrtGearListSetUserData(), 297, 310
XrtGearListSort(), 310
XrtGearLocation, 328
XrtGearMenuAlignment, 328
XrtGearMenuCallbackStruct, 329
XrtGearMovePolicy, 329
XrtGearMrmInitialize(), 28, 41, 69, 79, 90, 178
XrtGearNodeActivateCallbackStruct, 329
XrtGearNodeChangeFolderState(), 186, 219
XrtGearNodeCmpLabels(), 186
XrtGearNodeCmpPositionsInTree(), 187
XrtGearNodeConstructPath(), 187
XrtGearNodeCreateCallbackStruct, 329
XrtGearNodeCreateLW(), 215
XrtGearNodeCreateTreeFromBuffer(), 101, 187
XrtGearNodeCreateTreeFromStream(), 94, 99, 188
XrtGearNodeCurrentChangedCallbackStruct, 330
XrtGearNodeCvtLabelToString(), 189
XrtGearNodeCvtStringToLabel(), 189
XrtGearNodeDestroy(), 215
XrtGearNodeEnterCellCallbackStruct, 330
XrtGearNodeFolderStateCallbackStruct, 330
XrtGearNodeGetChild(), 190
XrtGearNodeGetChildren(), 221
XrtGearNodeGetEditable(), 216
XrtGearNodeGetFirstInTree(), 190

XrtGearNodeGetFirstNonCollapsedInTree(), 190
XrtGearNodeGetFolderState(), 221
XrtGearNodeGetHeight(), 215
XrtGearNodeGetLabel(), 216
XrtGearNodeGetLastInTree(), 190
XrtGearNodeGetLastNonCollapsedInTree(), 190
XrtGearNodeGetLevel(), 191
XrtGearNodeGetName(), 216
XrtGearNodeGetNextInTree(), 191
XrtGearNodeGetNextNonCollapsedInTree(), 191
XrtGearNodeGetNodeStyle(), 216
XrtGearNodeGetNumChildren(), 191
XrtGearNodeGetParent(), 216
XrtGearNodeGetPreviousInTree(), 192
XrtGearNodeGetPreviousNonCollapsedInTree(), 192
XrtGearNodeGetSelected(), 216
XrtGearNodeGetUserData(), 217
XrtGearNodeGetVisibility(), 114, 192
XrtGearNodeGetWidgetParent(), 192
XrtGearNodeGetWidth(), 217
XrtGearNodeIsAncestor(), 193
XrtGearNodeIsInCollapsedBranch(), 193
XrtGearNodeMakeVisible(), 114, 193
XrtGearNodeMoveCallbackStruct, 330
XrtGearNodeSetEditable(), 217
XrtGearNodeSetLabel(), 217
XrtGearNodeSetNodeStyle(), 217
XrtGearNodeSetParent(), 107, 193
XrtGearNodeSetSelected(), 217
XrtGearNodeSetUserData(), 218
XrtGearNodeSetValuesOnTree(), 194
XrtGearNodeSortTree(), 108, 194
XrtGearNodestyleFromName(), 196
XrtGearNodeTraverseTo(), 104, 194
XrtGearNodeValidateCallbackStruct, 331
XrtGearNodeWriteTreeToBuffer(), 103, 194
XrtGearNodeWriteTreeToStream(), 102, 195
XrtGearObject, 298
XrtGearOutlierNodeFromXEvent(), 180
XrtGearOutlinerCancelEdit(), 178
XrtGearOutlinerCommitEdit(), 179
XrtGearOutlinerDrawPS(), 146, 179
XrtGearOutlinerImportCallbackStruct, 331
XrtGearOutlinerMoveColumns(), 121, 146, 179
XrtGearOutlinerNodeFromXY(), 180
XrtGearOutlinerTraverseToCell(), 181
XrtGearOutlinerVaDrawPS(), 146, 181
XrtGearPageCallbackStruct, 331
XrtGearPaneResizeCallbackStruct, 332
XrtGearPrintCallbackStruct, 332
XrtGearProgresscalcTimeElapsed(), 239
XrtGearProgressCalcTimeToCompletion(), 239
XrtGearProgressCalcTimeTotal(), 239
XrtGearProgressCallbackStruct, 332
XrtGearProgressType, 332
XrtGearPSFontMap, 333
XrtGearReason, 333
XrtGearResizeCallbackStruct, 334
XrtGearRotation, 334
XrtGearSearchPolicy, 334
XrtGearSelecitonPolicy, 334
XrtGearSelectCallbackStruct, 334
XrtGearSelectionRestrictions, 335
XrtGearSetXmStringConverter(), 29, 41, 69, 79, 90
XrtGearSizingPolicy, 335
XrtGearSnapArg, 335
XrtGearSnapDrawArg, 335
XrtGearSnapGetWidgetImage(), 286
XrtGearSnapGetWidgetPixmap(), 283, 286
XrtGearSnapPrintPixmap(), 286
XrtGearSnapVaPrintPixmap(), 287
XrtGearState, 336
XrtGearStateChangedCallbackStruct, 336
XrtGearStrigCreate(), 311
XrtGearStringCompare(), 299, 310
XrtGearStringCopy(), 311
XrtGearStringCreate(), 297
XrtGearStringCreateCharString(), 96, 311
XrtGearStringCreateIconString(), 96, 311
XrtGearStringCreateXmString(), 312
XrtGearStringDestroy(), 298, 312
XrtGearStringGetValue(), 299, 312
XrtGearStringIsCharString(), 312
XrtGearStringIsIconString(), 312
XrtGearStringIsXmString(), 313
XrtGearStringLayout, 336
XrtGearStringSetValue(), 299, 313
XrtGearStyle, 336
XrtGearTabManagerMakeTabVisible(), 70
XrtGearTabManagerMakeTabVisitble(), 49
XrtGearTabSide, 336
XrtGearTextClipPolicy, 336
XrtGearTipsAcitonRemove(), 276, 278
XrtGearTipsActionDisplay(), 276, 278
XrtGearTipsAddCallback(), 278
XrtGearTipsAddCallbacks(), 279
XrtGearTipsDisplayHelp(), 275, 279
XrtGearTipsEnable(), 270, 279
XrtGearTipsGetValues(), 280
XrtGearTipsGetWidgetList(), 271, 280

XrtGearTipsRemoveAllCallbacks(), 280
XrtGearTipsRemoveCallback(), 280
XrtGearTipsRemoveCallbacks(), 281
XrtGearTipsRemoveHelp(), 275, 281
XrtGearTipsSetValues(), 269, 281
XrtGearTipsTrackingEvent(), 281
XrtGearTipsVaGetValues(), 282
XrtGearTipsVaSetValues(), 269, 270, 282
XrtGearToggleButtonGetState(), 35, 42
XrtGearToggleButtonSetState(), 35, 42
XrtGearType, 337
XrtGearWidgetResizePolicy, 337
XRTHOME, 8
XrtList, 289
　手続きと方法, 301
XrtPanedWindowGetPaneList(), 266
XrtString, 297
　手続きと方法, 310

あ

Icon Library, 8
印刷, 13

か

環境変数, 8
許可, 10
国際化, 12
コンパイル, 10

さ

ストリングのコピー, 298
ストリングの破棄, 298
ストリングの比較, 299

た

タブ, 43
ダブル･バッファリング, 149

ら

リスト･アイテムの検索, 292
リスト･アイテムの削除, 291
リスト･アイテムの追加, 290
リストとユーザ･データ, 297
リストの作成, 290
リストのソート, 296
リストの比較, 295
リソースの型とクラス, 339
リソースファイル, 11, 351