Motif 対応 XRT/field™

プログラマーズ・ガイド&リファレンス・マニュアル

UNIX Edition ■ Version 2.0



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使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)を参照に従います。

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/field について 1
XRT/field 2.0 の新しい機能 2
前提知識 3
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/field の使用法

1　XRT/field の 簡単なプログラムの開発 9

1.1　環境変数 9
1.2　基本的なテキスト・フィールド 10
1.3　プログラムのコンパイルと許可 11
1.4　基本的な検証 12
1.5　Combo Box と Spin Box 13
1.5.1　Combo Box の作成 14
1.5.2　Spin Box の作成 15
1.6　XrtCalendar ウィジェット 17
1.7　ここから先に進むには 18

2　XRT/field の基本操作 19

2.1　Motif との統合 19
2.2　XrtField サブクラス 21
2.3　デフォルトのユーザ・インタラクション 21
2.4　フィールドに値を記憶させる 23
2.5　編集と表示のフォーマット 24
2.6　変換、補完、検証 24
2.7　編集のプロセス 24
2.7.1　Enter Field Action プロセス 25
2.7.2　Keystroke Action プロセス 25
2.7.3　Paste Action プロセス 25
2.7.4　User Exit Action プロセス 26
2.7.5　Programmatic Change Action プロセス 27
2.7.6　編集プロセスのまとめ 28

2.8 ポインタ・リソース 29
2.9 浮動小数点リソース 29
2.10 色 29
2.11 リソース・ファイルと GUI ビルダ 30

3 XRT/field のプログラミング 31

3.1 リソース 31
3.2 編集フォーマットの指定 36
3.3 日付編集フォーマットの指定 38
3.4 検証 40
3.4.1 行数 41
3.4.2 行の長さ 41
3.4.3 有効・無効な文字 41
3.4.4 検証マスク 42
3.4.5 データ型 46
3.4.6 値の範囲 46
3.4.7 有効な値のリスト 46
3.4.8 検証コールバック 48
3.5 無効フィールドの処理 49
3.6 変換 51
3.7 補完 52
3.8 表示フォーマットの指定 53
3.9 Exit コールバック 56
3.10 フィールドの色 57
3.11 複数行のテキスト 58

4 XRT/field の 高度なプログラミング 59

4.1 デフォルトのフィールド値 59
4.2 入力プロンプトとコールバック 61
4.3 XRT/table と一緒に使用する 61
4.4 自動マッチングによる補完 64
4.5 ドロップ・ダウン・リスト 65
4.6 XrtComboBox のプログラミング 65
4.7 XrtSpinBox のプログラミング 70
4.8 XrtCalendar のプログラミング 72
4.9 日付エディタのポップアップ 75
4.10 キーにアクションを割り当てる 76
4.10.1 Calendar Action 76
4.10.2 Cancel Action 77
4.10.3 Complete Action 77
4.10.4 Menu Action 78

4.10.5 Overstrike アクション....79
4.10.6 Scrollbars アクション....79
4.10.7 Spin アクション....80
4.11 複数のフィールドのチェック....81
4.12 日付の変換....82
4.13 スクロールのコントロール....84
4.14 マスク文字の定義....85
4.15 XrtField と XmText コールバック....85
4.16 XRT/field で C++を使用する....87
4.17 UIL の使用....87
4.18 リソース値をストリングに変換....87
4.19 国際化....88

Part II リファレンス

A リソースの一覧....93

A.1 ウィジェットの概要....93
A.2 XrtField リソース....95
A.3 XmText リソース....117
A.4 継承された XrtField リソース....120
A.5 XrtComboBox リソース....121
A.6 XrtSpinBox リソース....122
A.7 継承された XrtComboBox と XrtSpinBox リソース....124
A.8 XrtCalendar リソース....125

B 手続きと方法....129

C マクロ....139

D トランスレーションとアクション....141

D.1 デフォルトのトランスレーション....141
D.2 アクション･ルーチン....142

E データ型....145

F リソースの型とクラス....153

F.1 XrtField のサブクラスに共通なリソース....153
F.2 XrtStringField リソース....155
F.3 XrtIntField リソース....155
F.4 XrtFloatField リソース....156

F.5 XrtDateField リソース....156
F.6 XrtComboBox リソース....157
F.7 XrtSpinBox リソース....157
F.8 XrtCalendar リソース....157

G リソースファイルと GUI ビルダーの使用....159

G.1 構文....159
G.2 リソース・ファイルの例....162

H UIL 引数....167

H.1 XrtField のサブクラスに共通なリソース....167
H.2 XrtIntField のリソース....168
H.3 XrtFloatField のリソース....168
H.4 XrtDateField のリソース....168
H.5 XrtComboBox のリソース....169
H.6 XrtSpinBox のリソース....169
H.7 XrtCalendar Resources....169
H.8 Reasons....170

I サンプル・コード....171

I.1 例....171
I.2 デモ....172

索引....173

はじめに

XRT/field について ■ XRT/field 20.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/field について

XRT/fieldウィジェット・ファミリ

XRT/field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。ウィジェットが増えた点を除けば、XRT/field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ほとんど違いはありません。

XRT/fieldウィジェットは、記憶されたテキストを変換、補完、検証することができます。データベース関連等、データ入力が中心となるアプリケーションを作成する場合、XRT/fieldを使用することにより、データ入力や検証機能を簡単に効率よくアプリケーションに追加することができます。また、XRT/fieldには、spin boxやcombo boxを作成するウィジェットも提供されています。各ウィジェットは、リソースを使用して表示や動作をコントロールすることができます。

XRT/fieldは、次の三つのウィジェット・クラスから構成されています。

- XrtField とそのサブクラス(ユーザが入力したテキストを記憶、操作する)
- XrtComboBox(ユーザが pick list から値を選択する)
- XrtSpinBox(ユーザが数値をインクレメントしたり、pick list をサイクルさせて値を選択する)
- XrtCalendar(フィールドなどに日付のポップアップ機能を追加する)

XrtFieldウィジェット・クラスは、MotifのXmTextウィジェットのサブクラスとして定義されていますので、テキスト・ウィジェットやXRT/table(作表ウィジェット)を使用するアプリケーションに直接挿入することができます。また、XrtComboBox、XrtSpinBox、XrtCalendarは、XmManagerのサブクラスであるXmXrtFldManagerのサブクラスとして定義されています。

XRT/field の機能

XrtField ウィジェット･クラスには、下記の機能が提供されています。

- 検証: データは、マスク、データ型、値の範囲、pick list を使用して検証されます。
- 変換: 値は、コールバック･ルーチンを使用して変換されます。
- 補完: 値は、コールバック･ルーチンまたは pick list を使用して補完されます。
- 書式指定: 値の編集･表示のフォーマットを指定できます。
- アプリケーションは、入力、exit、検証、エラー検知のコールバック･ルーチンを使用できます。
- 入力文字の有効･無効を指定できます。
- 必要時には、スクロールバーが表示されます。

XrtField は、その他に XmtText ウィジェットの機能を継承しています。

XrtComboBox は、子の arrow(矢印)ウィジェットの配置やウィジェット･クラスをコントロールします。

XRT/field 2.0 の新しい機能

XRT/field 2.0 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。

- XrtCalendar という新しいウィジェットが追加されました。このウィジェットは、XrtDateField に対する日付のポップアップ等に使用できます。XrtCalendar の簡単な紹介が P17、このウィジェットのプログラミングに関しては、P72 に解説があります。
- XrtSpinBox の日付/時間の機能が、サブ･フィールドをサポートするようになりました。詳細は、P72 を参照してください。
- 自動マッチング機能が追加され、ユーザがデータの一部を入力するだけで、データが補完されるようになりました。詳細は、P64 を参照してください。

XrtComboBox では、ドロップ･ダウン･リストの作成を combo box の作成時と同時に行うことができるようになりました。この機能に関する解説は P65 にあります。

マスク付きのフィールドに対するペーストやドラッグ&ドロップで構成される新しいアクション、Paste が追加されました。詳細は、P25 をご覧ください。

新しいデモや例プログラム

既存の機能の改良やバグの修正等が行われています。(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/field 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/field 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/field 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/field 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/index.htm
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/field 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/field

1

XRT/field の 簡単なプログラムの開発

環境変数 ■ 基本的なテキスト・フィールド
プログラムのコンパイルと許可 ■ 基本的な検証
Combo Box と Spin Box ■ XrtCalendar の作成
ここから先に進むには

この章では、次の各ステップに沿ってXRT/fieldを紹介していきます。

■ ストリング・フィールドにテキストを表示するプログラムをコンパイル、実行する
■ ウィジェットを整数フィールドに変更し、フィールドの内容の簡単な検証を行う
始めに combo box、次に spin box を作成するようにプログラムを編集する
簡単な calendar ウィジェットを作成する

1.1 環境変数

開始する前にUNIXの環境をご確認ください。*XRTHOME*という環境変数が、XRT/field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、XRT/fieldのリリース&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/field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。

1.2 基本的なテキスト･フィールド

図 1 は、XRT/field の使用例です。この *simple.c* プログラムは、テキスト･ストリングを記憶する XrtStringField クラスのフィールドを作成します。(フィールドのウィジェット･クラスは、2.2 に詳しく解説されています。)

simple.c のコードは、一般的な Motif プログラミングの形です。13 行目から 15 行目では、Intrinsics をイニシャライズし、標準の Xt Intrinsics の方法で toplevel のシェル･ウィジェットを作成しています。18 行目から 21 行目では、標準の Xt Intrinsics の手続き *XtVaCreateManagedWidget()*を使用して、フィールドを生成しています。

`XmNxrtFldInternalValue` リソースがこのフィールドの値を記憶します。この場合、値は、`"this is my text"`の文字ストリングです。

標準のXtの手続き(*XtSetValues()*、*XtGetValues()*、*XtVaSetValues()*)をフィールドの操作に使用することができます。

このプログラムは、*$XRTHOME/src/field/examples/simple.c* にあります。

```
1 #include <X11/Intrinsic.h>
2 #include <Xm/Xm.h>
3 #include <Xm/Text.h>
4 #include <Xm/XrtField.h>
5
6 int
7 main (int argc, char *argv[])
8 {
9   XtAppContext app_context;
10  Widget field,    toplevel;
11
12  /* Initialize Intrinsics */
13  toplevel = XtVaAppInitialize(&app_context,
14      "Simple", NULL, 0,
15      &argc, argv, NULL, NULL);
16
17  /* Create field */
18  field = XtVaCreateManagedWidget("field",
19      xmXrtStringFieldWidgetClass, toplevel,
20      XmNxrtFldInternalValue, "this is my text",
21      NULL);
22
23  XtRealizeWidget(toplevel);
24  XtAppMainLoop(app_context);
25 }
```

図 1　簡単なフィールドを作成する simple.c

1.3 プログラムのコンパイルと許可

次のようにして、*simple.c*をコンパイル、リンクできます。[1]

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

$XRTHOME/bin/xrt_auth simple
```

*simple*と入力して、このプログラムを実行します。図2は、このプログラムで作成されたフィールドです。



図2　simple.cによって作成されたフィールド

ライブラリとインクルード･ファイル

上記のコマンドは、X、Motif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るとみなしています。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/field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/field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/field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。

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、空白のXRT/fieldウィジェットが表示され、次のメッセージが*stdout*に出力されます。

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

```
This application is built with a disabled or an expired
time-limited version of XRT/field for Motif. Please contact
KL Group Inc. or your XRT/field 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/field 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3　期限切れまたは未許可の場合のメッセージ

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/field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。

1.4　基本的な検証

XRT/field を使用して、フィールドに含まれる値が有効かどうかを簡単に検証することができます。

図4は、フィールドを作成しその値を検証する簡単なプログラムです。このプログラムと *simple.c* の違いは、太字で示されています。

simple.c とは、次の二つが異なっています。

- 作成されたフィールドは、XrtIntField(整数)ウィジェット・クラスに属しています。これらのウィジェットは整数だけを受け入れ、それ以外の値には無効のフラグを立てます。(フィールドのウィジェット・クラスは、2.2 に解説されています。)
- `XmNXrtFldMin` と `XmNxrtFldMax` リソースが、フィールドの最小値と最大値に設定されています。1 より小さい値及び 10 より大きい値には、無効のフラグが立てられます。

```
1 #include <X11/Intrinsic.h>
2 #include <Xm/Xm.h>
3 #include <Xm/Text.h>
4 #include <Xm/XrtField.h>
5
6 int
7 main (int argc, char *argv[])
8 {
9   XtAppContext app_context;
10  Widget field,    toplevel;
11
12  /* Initialize Intrinsics */
13  toplevel = XtVaAppInitialize(&app_context,
14      "Simple", NULL, 0,
15      &argc, argv, NULL, NULL);
16
17  /* create field */
18  field = XtVaCreateManagedWidget("field",
19      xmXrtIntFieldWidgetClass,    toplevel,
20      XmNxrtFldMin,            1,
21      XmNxrtFldMax,            10,
22      NULL);
23
24  XtRealizeWidget(toplevel);
25  XtAppMainLoop(app_context);
26 }
```

図 4　検証の例を加えた simple.c

その他の検証法

XRT/field には、上記の例の他にも次のような検証法があります。

- フィールドの幅と長さ
- 有効･無効文字のリスト
- 検証マスク
- 有効な値のリスト(pick list)
- 検証コールバック

これらの方法の詳細は、3.4 に解説されています。

1.5　Combo Box と Spin Box

通常、フィールドは他のウィジェットの子として作成されます。例えば、XmForm ウィジェットは、複数のフィールド･ウィジェットを持つことができます。XRT/field には、フィールドを含むことができる、次の二つの composite ウィジェットが提供されています。

- フィールドと arrow ウィジェットを作成、操作する XrtComboBox
- フィールドと二つの arrow ウィジェットを作成、操作する XrtSpinBox

1.5.1 Combo Box の作成

図 5 は、simple.c が XrtComboBox ウィジェットを作成するように修正したプログラムです。変更された部分は太字で示してあります。

```
1 #include <X11/Intrinsic.h>
2 #include <Xm/Xm.h>
3 #include <Xm/Text.h>
4 #include <Xm/XrtField.h>
5
6 int
7 main (int argc, char *argv[])
8 {
9   XtAppContext         app_context;
10  Widget toplevel,     combobox;
11  static String picklist[] = {"John Jones","Harry Smith",
12      "Jean Jones","Mary Martin","Harry Hayes",NULL};
13
14  /* Initialize Intrinsics */
15  toplevel = XtVaAppInitialize(&app_context,
16      "Combo", NULL, 0,
17      &argc, argv, NULL, NULL);
18
19  /* Create combo box */
20  combobox = XtVaCreateManagedWidget("combobox",
21      xmXrtComboBoxWidgetClass,        toplevel,
22      XmNxrtComboFieldType,        XRTFLD_FIELDTYPE_STRING,
23      XmNxrtFldPickList,           picklist,
24      NULL);
25
26  XtRealizeWidget(toplevel);
27  XtAppMainLoop(app_context);
28 }
```

図 5　XrtComboBox ウィジェットを作成するように修正された simple.c

このプログラムがコンパイル、実行されると、次のウィジェットが表示されます。

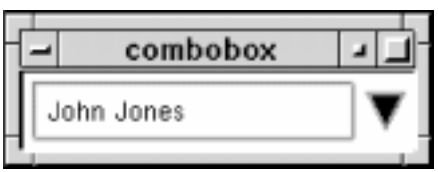


図 6　図 4 のプログラムによって作成された XrtComboBox

このプログラムでは、`XmNxrtFldPickList` がフィールドの有効な値のリストを定義しています。(フィールドのリソースは、combo box を定義する時にセットできます。また、*XtVaSetValues()*と *XtVaGetValues()*を使用して combo box 内からフィールドのリソースを設定、取得することもできます。)

フィールドのサブクラスは、XmNxrtComboFieldType リソースで指定し、XRTFLD_FIELDTYPE_STRING は、XrtStringField ウィジェットを指定します。

ユーザが arrow ウィジェットをクリックすると、XmNxrtFldPickList で定義された有効な値のリストを表示する、(XmList クラスの)ポップアップ・メニューが現れます。

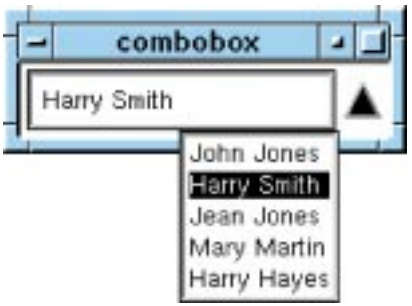


図 7　XrtComboBox ウィジェットに含まれるポップアップ・メニュー

ユーザは、このリストの中から値をクリックして選択します。ユーザが値を選択すると、ポップアップ・メニューが消え、選択された値がフィールド内に表示されます。フィールドの値を変更せずにポップアップ・メニューを消すには、arrow キーをもう一度クリックします。

XrtComboBox ウィジェットのプログラミングは、4.6 に詳しく解説されています。

1.5.2　Spin Box の作成

XrtSpinBox ウィジェットの作成法は、XrtComboBox ウィジェットの場合と似ています。図 9 は、*simple.c* を XrtSpinBox ウィジェットを作成するように変更してあります。このプログラムがコンパイル、実行されると、図 8 が表示されます。

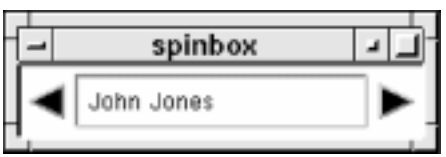


図 8　spin.c によって作成された XrtSpinBox

XrtSpinBox ウィジェットが作成される時、フィールドと二つの arrow ウィジェットの三つの子ウィジェットも自動的に作成されます。ユーザが右 arrow ウィジェットをクリックすると、フィールドは、pick list の次のエレメントを表示します。左 arrow をクリックすると、pick list を逆方向に見ることができます。

XrtSpinBox ウィジェットが作成される時、子フィールドに値を割り当てることができます。XrtComboBox の場合と同様に、*XtVaSetValues()*または *XtVaGetValues()*を使用して、spin box 内からフィールドのリソースを設定、取得することができます。

XrtSpinBox は、カウンタとして使用することもできます。この場合、ユーザが右 arrow をクリックするとウィジェットの値が増加し、左 arrow をクリックすると値が減少します。尚、カウンタやその他の XrtSpinBox の機能は、4.7 に詳しく解説されています。

```
1 #include <X11/Intrinsic.h>
2 #include <Xm/Xm.h>
3 #include <Xm/Text.h>
4 #include <Xm/XrtField.h>
5
6 int
7 main (int argc, char *argv[])
8 {
9   XtAppContext    app_context;
10  Widget      toplevel, spinbox;
11  static String picklist[] = {"John Jones","Harry Smith",
12      "Jean Jones","Mary Martin","Harry Hayes",NULL};
13
14  /* Initialize Intrinsics */
15  toplevel = XtVaAppInitialize(&app_context,
16      "Spin", NULL, 0,
17      &argc, argv, NULL, NULL);
18
19  /* Create spin box */
20  spinbox = XtVaCreateManagedWidget("spinbox",
21      xmXrtSpinBoxWidgetClass,     toplevel,
22      XmNxrtSpinFieldType,         XRTFLD_FIELDTYPE_STRING,
23      XmNxrtFldPickList,       picklist,
24      NULL);
25
26  XtRealizeWidget(toplevel);
27  XtAppMainLoop(app_context);
28 }
```

図 9　XrtSpinBox を作成するように修正された simple.c

1.6 XrtCalendar ウィジェット

XrtCalendar は、XrtSpinBox や XrtComboBox ウィジェットと同じようにして作成することができます。図 10 は、その下の図 11 にあるウィジェットを作成するプログラムです。

```
1 #include <Xm/Xm.h>
2 #include <Xm/XrtField.h>
3
4 int
5 #ifndef _NO_PROTO
6 main (int argc, char *argv[])
7    #else
8    main(argc, argv)
9    int argc;
10   char *argv[];
11   #endif
12   {
13         XtAppContext app_context;
14         Widget toplevel, calendar;
15
16         /* Initialize Intrinsics */
17         toplevel = XtVaAppInitialize(&app_context, "Calendar", NULL, 0,
18          &argc, argv, NULL, NULL, 0);
19
20         /* Create calendar */
21         calendar = XtVaCreateManagedWidget("calendar",
22          xmXrtCalendarWidgetClass, toplevel,
23          NULL);
24
25         XtRealizeWidget(toplevel);
26         XtAppMainLoop(app_context);
27 }
```

図 10　XrtCalendar ウィジェットを作成するコードの例



図 11　簡単な XrtCalendar ウィジェット

XrtCalendar ウィジェットは、オプションで、日付フィールドの spin box を持つことができる複合(compound)ウィジェットです。デフォルトでは、カレンダの下に配置される時間の spin box は表示されません。

時間や、月、年の spin box は、全て XrtSpinBox と同様な機能を持ちます。

日付グリッドでは、範囲を設定する最小及び最大値を指定できる他、`XmNxrtCalendarEnabled` を使用して、このウィジェットに対するユーザ・インタラクションをコントロールすることができます。

ユーザは、Tab キーを使用してフィールド間を移動したり、矢印キーで日付グリッドの日を移動することができます。また、時間と年のフィールドの値は、spin box から選択する他、直接入力することができますが、月のフィールドは、spin box のコントロールを使用した変更しか受け付けません。カレンダが日付のポップアップの場合は、`Calendar(Hide)`アクションを使用した ENTER キーを押すことによって、このカレンダを隠すことができます。ユーザ・インタラクションに関しては、Appendix D を参照してください。

XrtComboBox や XrtSpinBox と同様に、*XtVaSetValues()*と *XtVaGetValues()*を使用して、フィールドのリソースを設定したり取得したりすることができます。尚、XrtCalendar は、4.8 に解説がありますので、詳細はそちらを参照してください。

1.7 ここから先に進むには

短期間で XRT/field のプログラミングを習得するための二つの提案です。

- 第 2 章は、XRT/field の基本的な操作の解説です。この章を読むことにより、XRT/field の基礎知識を得ることができます。
- XRT/field には、例やデモプログラムが提供されています。これらを実行したり、そのソースコードを調べることにより、XRT/field プログラミングのテクニックを習得することができます。Appendix I に、これらのプログラムの簡単な解説があります。

2

XRT/field の基本操作

Motif との統合 ■ XrtField サブクラス
デフォルトのユーザ・インタラクション ■ フィールドに値を記憶させる
編集と表示のフォーマット ■ 変換、補完、検証
編集のプロセス ■ ポインタ・リソース
浮動小数点リソース ■ 色
リソース・ファイルと GUI ビルダ

XRT/fieldは、入力されたデータを検証、変換、補完するウィジェットを定義します。各ウィジェットは、それぞれ異なった型のデータを処理します。SpinBox、ComboBox、Calendarウィジェットは、データを操作するように定義することができます。

2.1 Motif との統合

XRT/fieldは、Motifツールキットに四つの新たなウィジェットを追加します。

- XrtField: XmText が変換、検証、表示機能を持つように拡張したサブクラスを定義する
- XrtComboBox: XrtField と一緒に作成される combo box

XrtSpinBox: XrtField と一緒に作成される spin box

XrtCalendar: カレンダ・ウィジェット

これらのウィジェットは、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。

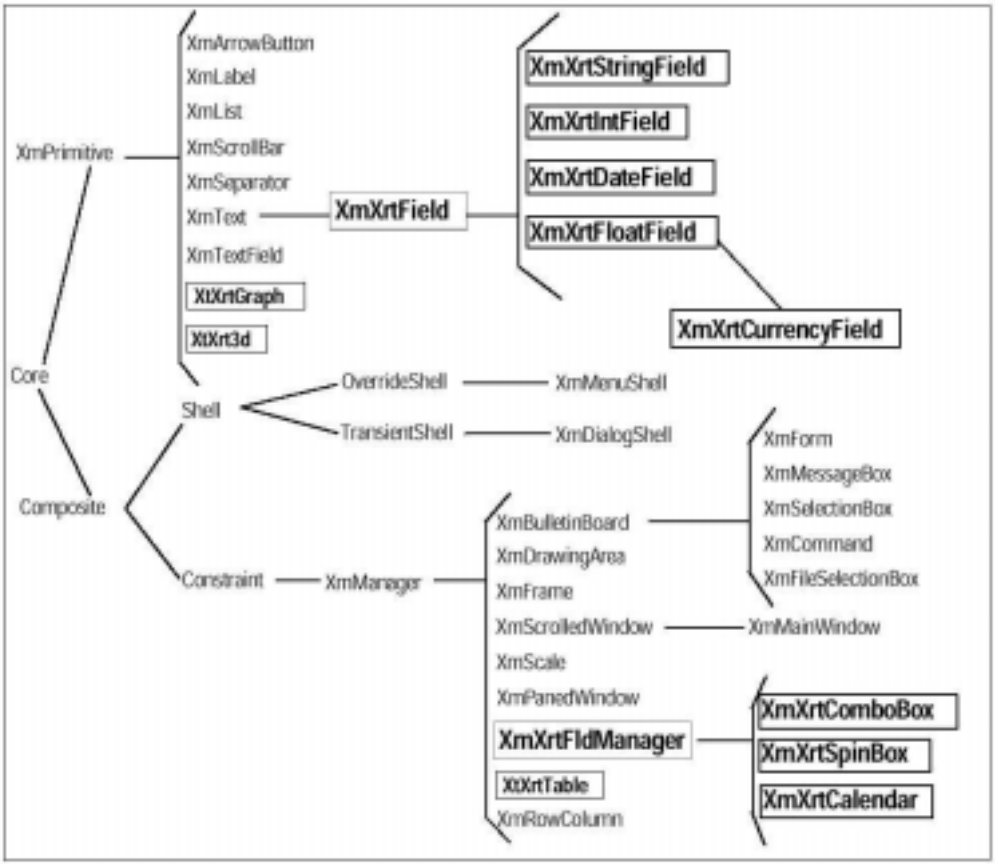


図12　Motifクラス階層内のXRTウィジェットの位置

Xt Intrinsicsは、ウィジェットをクラス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、XmFormウィジェットは、Core、Composite、Constraint、XmManagerのサブクラスのXmBulletinBoardのサブクラスです。

XrtFieldウィジェット・クラスは、XmPrimitiveのサブクラスであるXmTextのサブクラスとして定義されます。これは、XrtFieldウィジェットで定義されるリソースの他にXmText、XmPrimitiveとCoreのリソースも設定できることを意味します。例えば、`XmNbackground`リソースを用いてデフォルトの背景色を設定することができます。

また、図12にあるように、XrtComboBox、XrtSpinBox、XrtCalendarは、XmXrtFldManagerのサブクラスです。

XmText ウィジェットの置き換え

XrtFieldはXmTextのサブクラスですので、アプリケーションのどのXmTextウィジェットもXrtFieldに置き換えることができます。

XmTextField ウィジェットの置き換え

XmTextField は XmText と同じリソースを持っていますので、アプリケーションの動作を変更することなく、XmTextField ウィジェットを XrtField ウィジェットで置き換えることができます。

2.2 XrtField サブクラス

XrtField は、幾つかのサブクラスを定義する abstract クラス(基本となるクラス)です。XrtField クラスのインスタンスは生成できません。各サブクラス(図 13 参照)は、それぞれ異なる型の入力を処理します。

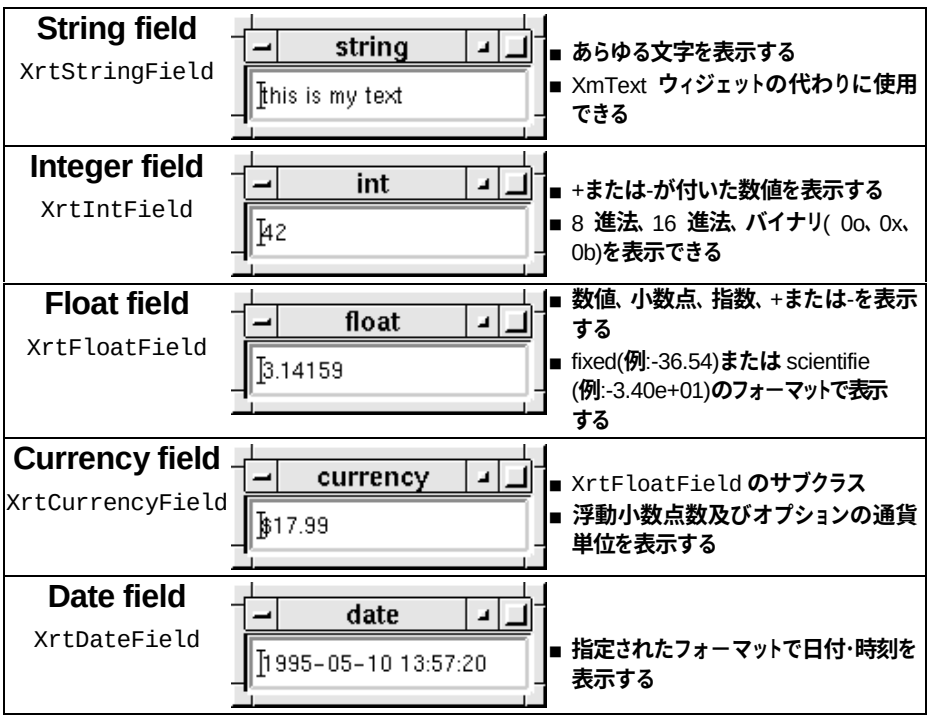


図 13　XrtField ウィジェットのサブクラス

このマニュアルでは、*フィールド(field)*という言葉は、XrtField のサブクラスに属するウィジェットを指します。

2.3 デフォルトのユーザ・インタラクション

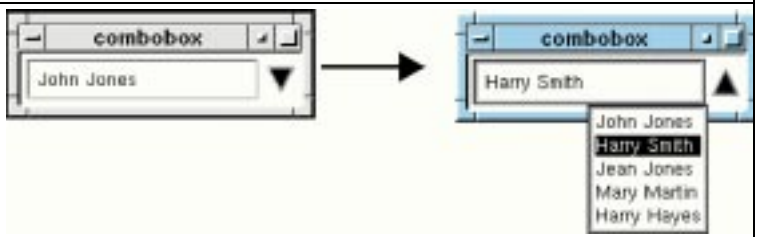
図 14 は、一般的な XRT/field とのインタラクションの例です。ユーザ・インタラクションの完全なリストは、Appendix D にありますので参照してください。

XrtField サブクラス



- 標準の XmText コントロールを使用してテキストを入力･修正
- Tab または Return を押してフィールドの外へトラバース

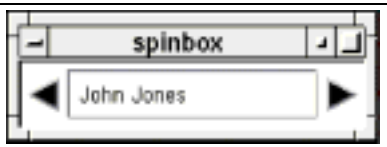
XrtComboBox



マウス:
- Arrow をクリックしてポップアップ･メニューを表示
- Arrow をもう一度クリック、または osfCancel を押すか、メニューの外をクリックしてメニューを消す
- ポップアップ･メニューのアイテムをクリックして、XrtField ウィジェットに記憶されている値を変更

キーボード:
- Ctrl-Down を押してポップアップ･メニューを表示 (Ctrl-Up でメニューを消去)
- Up または Down キーでメニュー内を上下に移動
- 選択するアイテムの最初の文字をタイプして、そのアイテムに移動
- Return でアイテムを選択

XrtSpinBox



マウス:
- 右 arrow をクリックして次のアイテムを表示
- 左 arrow をクリックして前のアイテムを表示

キーボード:
- Up キーで次のアイテムを表示
- Down キーで前のアイテムを表示

XrtCalendar



マウス:
- 上 arrow で現在値をインクレメント
- 下 arrow で現在値をデクレメント
- 日付グリッド上の数値をクリックして日付を選択

キーボード:
- Tab によってフィールド間を移動
- 日付グリッドにフォーカスがある場合は、矢印キーで移動
- 月、年、時間フィールドでは、上または右の矢印キーによって現在値をインクレメント
- 月、年、時間フィールドでは、下または左の矢印キーによって現在地をデクレメント

図 14　XRT/field のユーザ･インタラクション

2.4 フィールドに値を記憶させる

フィールドに値を記憶させるには、XmNvalue または XmNxrtFldInternalValue の二つのリソースの内どちらかをセットしてください。XRT の機能を有効に使用するには、XmNxrtFldInternalValue を使用してください。但し、フィールドに(何を入力するべきかをユーザに通知する *dd/mm/yy* などの)キューをセットする場合は、XmNvalue を使用してください。

XmNxrtFldInternalValue

XmNxrtFldInternalValue は、そのウィジェットのサブクラスに適するデータ型を用いて値を記憶します。

フィールドのサブクラス	内部の値の型
XrtStringField	string
XrtIntField	long
XrtFloatField	double
XrtCurrencyField	double
XrtDateField	struct tm

次の例では、integer フィールドを作成し、XmNxrtFldInternalValue を使用してその値をセットしています。

```
field = XtVaCreateManagedWidget("field",
  xmXrtIntFieldWidgetClass, toplevel,
  XmNxrtFldInternalValue, 42,
  NULL);
```

ユーザがそのフィールドを出ると、ユーザによって入力された値は、(必要時には型を変換して)XmNvalue から XmNxrtFldInternalValue へコピーされます。

XmNxrtFldInternalValueLong

ウィジェットが date フィールドである場合は、XmNxrtFldInternalValueLong は、XmNxrtFldInternalValue の値を *long* 型の整数として持ちます。(但し、これは日付がおよそ 1902 年から 2033 年の範囲にある場合です。)

XmNvalue

XmNvalue は、フィールドに記憶、表示されている実際のテキスト･ストリングを持ちます。次の例は、XmNvalue を使用して、integer フィールドの初期値をセットしています。

```
field = XtVaCreateManagedWidget("field",
  xmXrtIntFieldWidgetClass,   toplevel,
  XmNvalue,           "42",
  NULL);
```

XmNvalue が、ウィジェットのサブクラスに不適当な値(例: integer フィールドに“abc”という値)がセットされた場合は、値は表示されますが、無効というフラグが立てられます。

注意: XmNvalue は、マスク付きのフィールドにビジュアルなキューをセットする場合にのみ必要です。詳細は P45 を参照してください。

2.5 編集と表示のフォーマット

XRT/field では、フィールド内の値の表示をコントロールする二つのフォーマットを指定することができます。

- *編集フォーマット(edit format):* 編集されている時に値をどのように表示するかをコントロールする
- *表示フォーマット(display format):* 編集されていない時に値をどのように表示するかをコントロールする

編集フォーマットは 3.2 に、表示フォーマットは 3.8 にそれぞれ解説されています。

2.6 変換、補完、検証

XRT/field では、値の編集中に、次の三つの方法でフィールド値を操作することができます。

- 値の変換(*convert*)。例えば、値を小文字から大文字に変換することができます。
- 値の補完(*complete*)。例えば、ファイル名に.*c* という文字を追加することができます。
- 値の検証(*validate*)。

これらの操作は、編集プロセス中のあらゆる時点で行うことができます。

2.7 編集のプロセス

フィールドの編集プロセスは、次のステップに分けることができます。

- *enter field action:* ユーザがウィジェットに入る(フィールドに入力フォーカスを与える)。

keystroke action: ユーザがキーを押す。各キーの押下が keystroke action。

paste action: ユーザがペースト、ドラッグ&ドロップ操作を行う。

- *user exit action:* ユーザが、Return または Tab を押すか、マウス・ポインタをウィジェットの外に移動して、ウィジェットを出る。

アプリケーション自身も、フィールドに記憶されている値を編集することができます。この編集は、*programmatic change action* と呼ばれます。

XRT/field では、これらのどのアクションの後でも、プロセスを追加することができます。インタラクションの順序と可能なプロセスは、図 15 に示してあります。

2.7.1 Enter Field Action プロセス

enter field action が起きた時、次の事を提供することができます。

- enter field action に続いて開始されるコールバック・ルーチンのリスト。(enter コールバックの指定法は、4.2 に解説されています。)
- 編集中に、どのように値が表示されるかを指定する編集フォーマット。(編集フォーマットの指定法は、3.2 に解説されています。)

2.7.2 Keystroke Action プロセス

keystroke action が起きた時は、次の事を実行できます。

- (部分的に入力された)値のフォーマットを変換。(変換機能は、3.6 に解説されています。)
- 入力された値が有効であるかを検証。(検証機能は、3.4 に解説されています。)

2.7.3 Paste Action プロセス

paste action が起きた時は、次の事を実行できます。

- ペーストされた値を変換。(変換機能は、3.6 に解説されています。)
- ペーストされた値が有効か検証。(検証機能は、3.4 に解説されています。

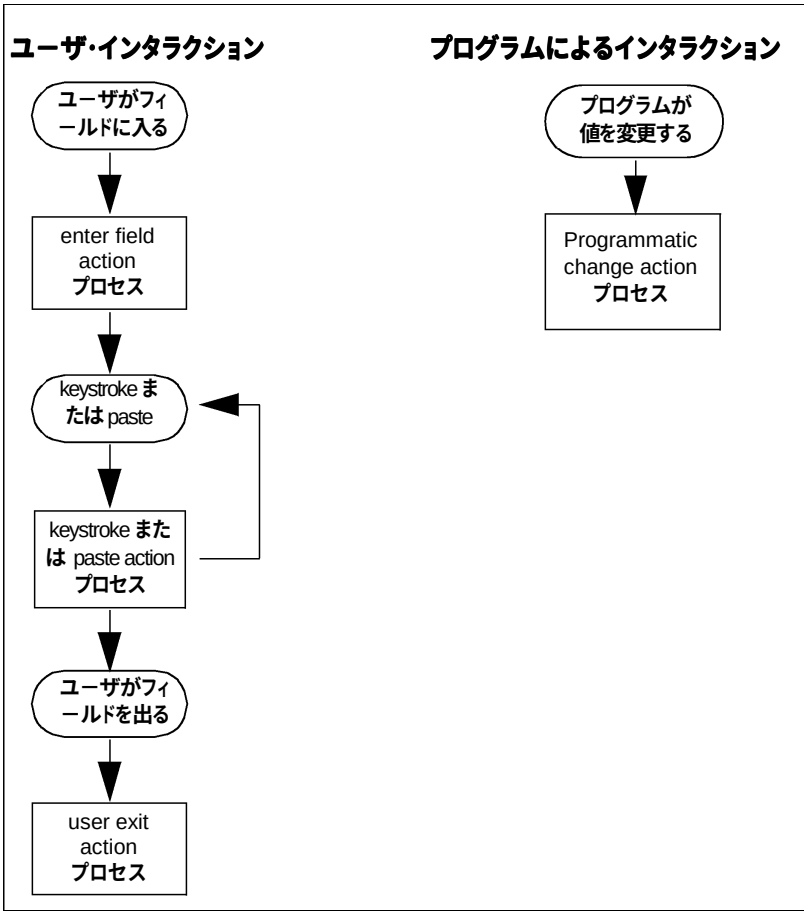


図 15　インタラクションとその結果のプロセス

2.7.4 User Exit Action プロセス

ユーザがフィールドを出ると、user exit action が起こります。”user exit”は、次のユーザによるアクションに定義することができます。

- `XmNfocusPolicy` が `XmPOINTER` にセットされている間に、マウス・ポインタを を他のウィンドウに移動する。
- Tab を押して次のウィジェットにトラバースする。
- Return を押す。(フィールドが複数行の入力を受付ない場合のみ。複数行の入力は、3.11 に解説されています。)

XmText ウィジェットとは異なり、フィールド・ウィジェットは、Return が押されると次のウィジェットにトラバースされます。この動作を不可にする方法は、3.9 に解説されています。

user exit action が起こった時は、次の事が実行できます。

- 値のフォーマットを変換する。(変換機能は 3.6 に解説されています。)
- 不完全に入力されている値を補完する。(補完は、3.7 に解説されています。)
- 値が有効であるか検証する。(検証は、3.4 に解説されています。)
- 値を表示する時に使用するフォーマットを指定する。(表示フォーマットの指定法は、3.8 に解説されています。
- exit コールバックを指定する。(コールバックは、3.9 に解説されています。)

2.7.5 Programmatic Change Action プロセス

フィールドに記憶されている値がプログラム自身によって変更された時は、programmatic change action が起こります。以下は、このアクションの例です。

- プログラムが `XmNvalue` または `XmNxrtFldInternalValue` の値を変更する。
- XmText 手続きが呼び出される。
- *Menu()*アクション・ルーチンで表示されるポップアップ・メニューから値が選択される。(*Menu()*は、4.10.4 に解説されています。)
- `XmNvalue` または `XmNxrtFldInternalValue` が、ウィジェットの生成時にセットされる。(フィールドの初期値が検証されます。)

programmatic change action が起こった時は、次の事が実行できます。

- フィールドの現在値のフォーマットを変換する。(変換機能は、3.6 に解説されています。)
- 値が有効であるか検証する。(検証は、3.4 に解説されています。)
- 値を表示する時に使用するフォーマットを指定する。(表示フォーマットの指定法は、3.8 に解説されています。

2.7.6 編集プロセスのまとめ

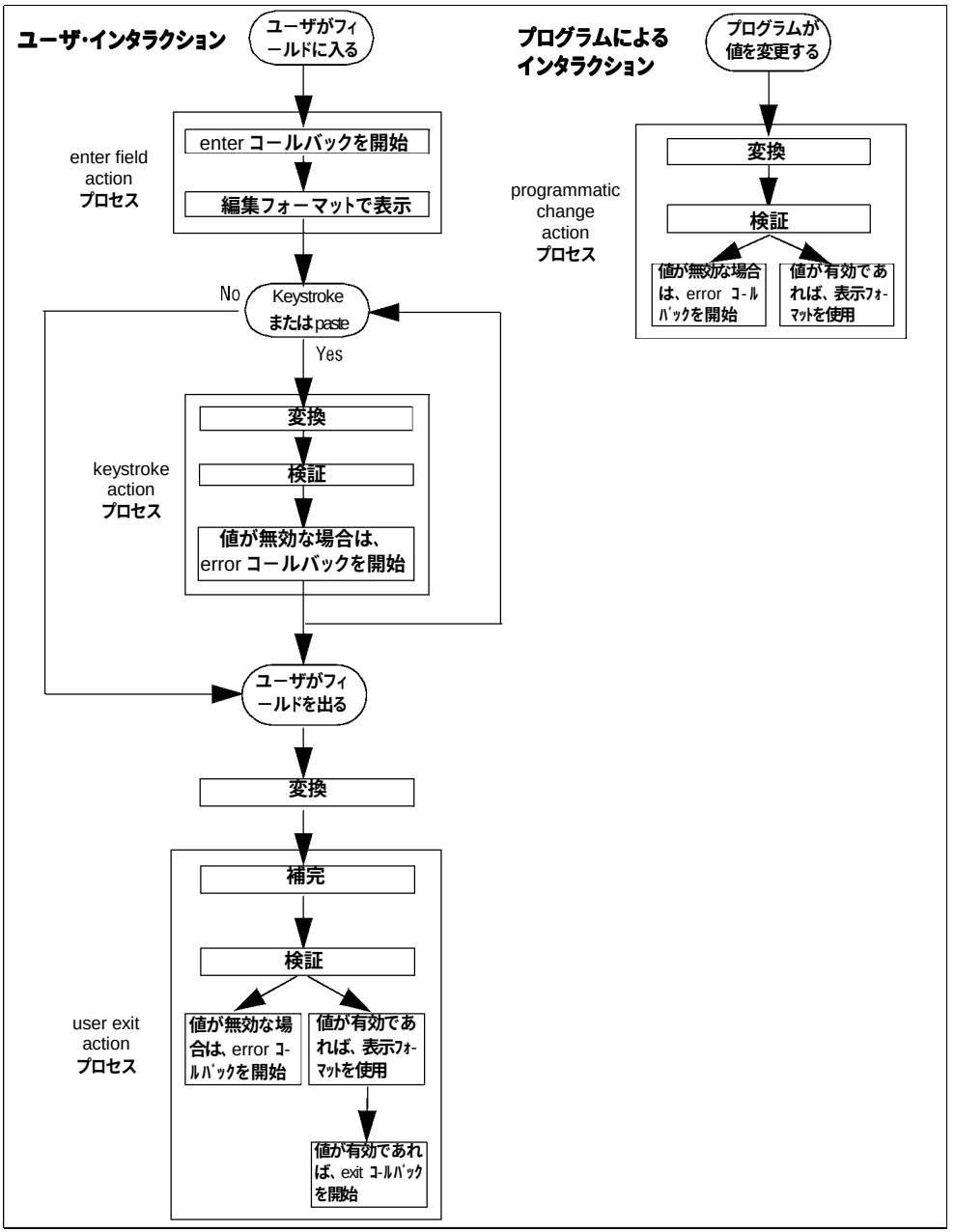


図 16　XrtField の編集プロセス

図 16 は、フィールド編集プロセスの全様を示しています。このダイアグラムは、一見複雑そうに見えますが、実際は、上記で説明された機能やその順序が図 15 に追加されているだけです。

2.8 ポインタ･リソース

幾つかのXRT/fieldのリソースは、*XtGetValues()*で使用されると、構造体かストリングのポインタを返します。これはウィジェットの内部記憶のポインタですので、このポインタを使用してメモリ内のデータを変更したり、このポインタを変更または解放しないでください。このポインタが指すものは全て"read only"で"temporary"です。(コピーされる、`XmNvalue`は例外です。このリソースの値は、使用後必ず解放してください。)

XRT/fieldは、ポインタ･リソースを通して渡された全てのデータの内部コピーを作成します。

Xtの手続き、*XtNewString()*はデータのコピー作成、*XtFree()*はこれらのコピーの解放に使用することができます。

2.9 浮動小数点リソース

float または currency フィールドで浮動小数点リソースを設定する時は、*XrtFldDoubleToArgVal()*を使用してください。例えば、フィールドに記憶できる最大浮動小数点数を設定する時は、次のようにします。

```
double maxval;
maxval = 125.5;
XtVaSetValues(myfield,
  XmNxrtFldMax, XrtFldDoubleToArgVal(maxval),
  NULL);
```

2.10 色

XRT/field で作成されたアプリケーションをモノクロの X サーバで使用する場合は、次のように表示されます。

- 背景色では、黒以外の色は白
- その他の白以外の色は黒

2.11 リソース･ファイルと GUI ビルダ

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 - 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 - エンドユーザがニーズにあわせてリソースの値を変更できます。

リソース･ファイルを使用して、(コールバック･リソースを除く)XRT/fieldの全てのリソースをセットすることができます。リソース･ファイルによって異なったリソース設定をプロトタイプしたり、その結果を直ぐに確認することができます。Appendix Gにリソース設定の構文の解説があります。

GUI ビルダの使用

XRT/field は、多くの主要な GUI 構築ツールと統合して使用することができます。これらのツールから XRT/field のリソースをセットする場合は、リソース･ファイルの場合と同じ構文を使用してください。この構文の詳細は、Appendix G にあります。

3

XRT/field のプログラミング

リソース ■ 編集フォーマットの指定
日付編集フォーマットの指定 ■ 検証
無効フィールドの処理 ■ 変換
補完 ■ 表示フォーマットの指定
Exit コールバック ■ フィールドの色
複数行のテキスト

3.1 リソース

以下は、XRT/field のリソースの一覧ですが、全て覚えていただく必要はありません。ほとんどのリソースは、デフォルトの設定のままにしておくことができます。

Resource	Type
XmNxrtFldAllowNull	Boolean
XmNxrtFldAutoComplete	Boolean
XmNxrtFldCase	XrtFldCase
XmNxrtFldCompleteCallback	XtCallbackList
XmNxrtFldConvertCallback	XtCallbackList
XmNxrtFldCreatePopups	Boolean
XmNxrtFldDefaultBackground	Pixel
XmNxrtFldDefaultForeground	Pixel
XmNxrtFldDisplayFormat	XrtFldFormat
XmNxrtFldDisplayHorizScrollBar	XrtFldDispScrollBar
XmNxrtFldDisplayVertScrollBar	XrtFldDispScrollBar
XmNxrtFldEditFormat	XrtFldFormat
XmNxrtFldEnterCallback	XtCallbackList

次ページに続く

Resource	Type
XmNxrtFldErrorBackground	Pixel
XmNxrtFldErrorCallback	XtCallbackList
XmNxrtFldErrorForeground	Pixel
XmNxrtFldErrorString	XmString
XmNxrtFldExitCallback	XtCallbackList
XmNxrtFldInvalidChars	String
XmNxrtFldIsDefault	Boolean
XmNxrtFldIsModified	Boolean
XmNxrtFldIsValid	Boolean
XmNxrtFldLastError	XrtFldError
XmNxrtFldMask	String
XmNxrtFldMaskChars	String
XmNxrtFldMatchMask	XrtFldMatchMask
XmNxrtFldMatchPickList	Boolean
XmNxrtFldMaxLineLength	int
XmNxrtFldMaxLines	int
XmNxrtFldMenuAlignment	XrtFldMenuAlignment
XmNxrtFldMenuCallback	XtCallbackList
XmNxrtFldMenuList	Widget
XmNxrtFldOverstrike	Boolean
XmNxrtFldPickListIndex	int
XmNxrtFldPopupType	XrtFldPopupType
XmNxrtFldPromptString	XmString
XmNxrtFldRange	XrtFldRange
XmNxrtFldSearchPolicy	XrtFldSearchPolicy
XmNxrtFldSpinCallback	XtCallbackList
XmNxrtFldValidateCallback	XtCallbackList
XmNxrtFldValidChars	String
XmNxrtFldVisibleItemCount	int

図 17　XrtField のリソース

Resource	Type
XmNxrtFldDefaultValue	String
XmNxrtFldInternalValue	String
XmNxrtFldMax	String
XmNxrtFldMin	String
XmNxrtFldPickList	String *

図 18　XrtStringField のリソース

Resource	Type
XmNxrtFldCounterIncrement	long
XmNxrtFldDefaultValue	long
XmNxrtFldDisplayFormatString	String
XmNxrtFldInternalValue	long
XmNxrtFldMax	long
XmNxrtFldMin	long
XmNxrtFldNegativeFormat	XrtFldNegFormat
XmNxrtFldPickList	long *
XmNxrtFldRange	XrtFldRange

図 19　XrtIntField のリソース

Resource	Type
XmNxrtFldCounterIncrement	double
XmNxrtFldDecimalPlaces	int
XmNxrtFldDefaultValue	double
XmNxrtFldDisplayFormatString	string
XmNxrtFldInternalValue	double
XmNxrtFldLeadingZero	Boolean
XmNxrtFldMax	double
XmNxrtFldMin	double
XmNxrtFldNegativeFormat	XrtFldNegFormat
XmNxrtFldPickList	double *
XmNxrtFldRange	XrtFldRange

図 20　XrtFloatField のリソース

Resource	Type
XmNxrtFldCalendar	Widget
XmNxrtFldCalendarCallback	XtCallbackList
XmNxrtFldCounterIncrement	struct tm *
XmNxrtFldDefaultValue	struct tm *
XmNxrtFldDisplayFormatString	String
XmNxrtFldEnterFormatList	String *
XmNxrtFldInternalValue	struct tm *
XmNxrtFldInternalValueLong	long
XmNxrtFldMax	struct tm *
XmNxrtFldMilleniumThreshold	int
XmNxrtFldMin	struct tm *
XmNxrtFldSpinPolicy	XrtFldSpinPolicy
XmNxrtFldPickList	struct tm **

図 21　XrtDateField のリソース

Resource	Type
XmNxrtComboArrow	Widget
XmNxrtComboArrowClass	WidgetClass
XmNxrtComboArrowType	XrtFldArrowType
XmNxrtComboField	Widget
XmNxrtComboFieldClass	WidgetClass
XmNxrtComboFieldType	XrtFldFieldType
XmNxrtComboSpacing	Dimension

図 22　XrtComboBox のリソース

Resource	Type
XmNxrtSpinArrowLayout	XrtSpinArrowLayout
XmNxrtSpinArrowOrientation	unsigned char
XmNxrtSpinArrowType	XrtFldArrowType
XmNxrtSpinDecrArrow	Widget
XmNxrtSpinDecrArrowClass	WidgetClass
XmNxrtSpinField	Widget
XmNxrtSpinFieldClass	WidgetClass
XmNxrtSpinFieldName	String
XmNxrtSpinFieldType	XrtFldFieldType
XmNxrtSpinIncrArrow	Widget
XmNxrtSpinIncrArrowClass	WidgetClass
XmNxrtSpinSpacing	Dimension
XmNxrtSpinType	XrtSpinType

図 23　XrtSpinBox のリソース

Resource	Type
XmNxrtCalendarCellMargin	Dimension
XmNxrtCalendarCellHighlightThickness	Dimension
XmNxrtCalendarCreateSpinMonth	Boolean
XmNxrtCalendarCreateSpinTime	Boolean
XmNxrtCalendarCreateSpinYear	Boolean
XmNxrtCalendarModifyCallback	XtCallbackList
XmNxrtCalendarEnabled	Boolean
XmNxrtCalendarFirstDayOfWeek	int
XmNxrtCalendarFullDayNames	Boolean
XmNxrtCalendarHeaderNumChars	int
XmNxrtCalendarHeaderBackground	Pixel
XmNxrtCalendarHeaderFontList	XmFontList
XmNxrtCalendarHeaderForeground	Pixel
XmNxrtCalendarMarginBottom	Dimension
XmNxrtCalendarMarginLeft	Dimension
XmNxrtCalendarMarginRight	Dimension
XmNxrtCalendarMarginTop	Dimension
XmNxrtCalendarMaxRollDate	struct tm*
XmNxrtCalendarMinRollDate	struct tm*
XmNxrtCalendarSpecialDayBackground	Pixel

次ページに続く

Resource	Type
XmNxrtCalendarSpecialDayForeground	Pixel
XmNxrtCalendarSpecialDays	XrtCalendarSpecialDay **
XmNxrtCalendarSelectedBackground	Pixel
XmNxrtCalendarSelectedForeground	Pixel
XmNxrtCalendarValue	struct tm*
XmNxrtCalendarValueLong	long

図 24　XrtCalendar のリソース

3.2　編集フォーマットの指定

ユーザがフィールドに入ると、(XmNxrtFldInternalValue リソースに記憶されている)フィールドの値は、編集することができるようになります。フィールドの値が編集中にどのように表示されるかは、XmNxrtFldEditFormat リソースでコントロールすることができます。

XmNxrtFldEditFormat の有効な値は、フィールドのサブクラスに依存します。

integer フィールド

integer フィールドでは、XmNxrtFldEditFormat は次のようにセットできます。

XRTFLD_FORMAT_BINARY	1 と 0 の数列として表示
XRTFLD_FORMAT_GENERAL	*printf* %d(long 10 進数)フォーマットで表示 (デフォルト)
XRTFLD_FORMAT_HEX	16 進数で表示
XRTFLD_FORMAT_OCTAL	8 進数で表示
XRTFLD_FORMAT_THOUSANDS	現在のロケールの区切り文字で 3 桁づつ区切って表示(例: 200,000)

上記に示した編集フォーマットは、フィールドに記憶される値の前に接頭文字をタイプして置き換えることが出来ます。“0b”はバイナリ、“0o”は 8 進数、“0x”は 16 進数を指定します。

float と currency フィールド

float フィールドでは、XmNxrtFldEditFormat は次のようにセットできます。

XRTFLD_FORMAT_FIXED	固定の精度(*printf* %f と同等)
XRTFLD_FORMAT_SCIENTIFIC	指数表記(*printf* %e と同等)

XRTLD_FORMAT_GENERAL	*printf* %g フォーマット(デフォルトでは、%e か%f の短い方を使用)
XRTFLD_FORMAT_PERCENT	パーセント表記(例: .75 は、75%)
XRTFLD_FORMAT_THOUSANDS	現在のロケールの区切り文字で 3 桁づつ区切って表示(例: 2,453.61)

currency フィールドでは、XmNxrtFldEditFormat は、上記の float フィールド値の他に下記にセットできます。

XRTFLD_FORMAT_CURRENCY	現在のロケールの標準の通貨表記(例: $1,234.00)
XRTFLD_FORMAT_CURRENCY_INTL	国際通貨表記(例: USDO 1,234.00)

通貨フィールドのデフォルト値は、XRTFLD_FORMAT_FIXED です。

下の例は、値が編集中に指数表記で表示されるフィールドを定義しています。

```
field = XtVaCreateManagedWidget("field",
  xmXrtFloatFieldWidgetClass, toplevel,
  XmNxrtFldInternalValue,
    XrtFldDoubleToArgVal(16857.0),
  XmNxrtFldEditFormat,    XRTFLD_FORMAT_SCIENTIFIC,
  XmNxrtFldDecimalPlaces, 4,
  NULL);
```

次にフィールドが編集された時は、値は、1.6857e+04 と表示されます。

小数点の指定

XmNxrtFldDecimalPlaces リソースは、小数点以下の桁数を指定します。float フィールドの場合のデフォルトは 2、通貨フィールドの場合のデフォルトは XRTFLD_LOCALE です。編集フォーマットが XRTFLD_FORMAT_GENERAL の場合は XmNxrtFldDecimalPlaces は無視されます。

先の例で XmNxrtFldDecimalPlaces が設定されていない場合、値 1.69e+04(小数点第三位以下が四捨五入)が表示されます。

その他のサブクラスの編集

date フィールドは、XmNxrtFldEnterFormatList リソースの最初のエレメントを編集フォーマットとして使用します。XmNxrtFldEnterFormatList は、3.3 に解説されています。

string フィールドでは、(文字ストリングは常にそのまま表示されるので)XmNxrtFldEditFormat リソースは意味を持ちません。

編集と表示フォーマット

XmNxrtFldEditFormat は、フィールドがフォーカスを得た時に値が最初にどのように表示されるかをコントロールします。XmNxrtFldDisplayFormat は、ユーザが編集を終え、そのウィジェットのフォーカスが移動された時に、値がどのように表示されるかをコントロールします。

XmNxrtFldDisplayFormat の詳細は、3.8 を参照してください。

read-only フィールドの作成

デフォルトでは、どのフィールドもユーザによる編集が可能です。read-only のフィールドを定義するには、XmNeditable と XmNcursorPositionVisible を False にセットしてください。

3.3 日付編集フォーマットの指定

日付や時刻を入力する際、不明瞭なフォーマットが問題となることがあります。例えば、"01/10/94"は、月と日のどちらが最初に来るかによって 1994 年 1 月 10 日とも 1994 年 10 月 1 日ともなります。"01/10/94"だけでは、正しい日付がわかりません。

XmNxrtFldEnterFormatList を使用すると、date フィールドに日付フォーマットのリストを定義できるので、上記のような混乱を防ぐことができます。ユーザによって値が入力されると、そのフォーマットは、リストのフォーマットと比較して決定されます。

date フォーマットは、以下の幾つかまたは全てを含む文字ストリングです。

D	日(1-31)
DD	二桁の日表記、日が一桁の場合は先頭に 0 を置く(例: 01、02)
W	省略された曜日
WW	曜日
M	月(1-12)
MM	二桁の月表記、月が一桁の場合は 0 を先頭に置く(例: 01、02)
MMM	省略された月名
MMMM	月名
Y、YY	今世紀での年(00-99)
YYYY	世紀を含む年
h	時間(0-23)
hh	二桁の時間表記、一桁の場合は 0 を先頭に置く(例: 01、02)
m	分(0-59)
mm	二桁の分表記、一桁の場合は 0 を先頭に置く(例: 01、02)
s	秒(0-59)
ss	二桁の秒表記、一桁の場合は 0 を先頭に置く(例: 01、02)
A、P	AM/PM 表記
Z	タイムゾーンの略
\\	エスケープ文字(次の文字は、その文字通りに解釈する)

例えば、01/10/94 を 1994 年 1 月 10 日とするには、MM/DD/YY というフォーマットを使用します。1994 年 10 月 1 日とするには、DD/MM/YY とします。

上記リスト以外の文字はスキップされます。このリストにある文字をその文字通りの意味に使用するには、その文字の前にバックスラッシュを二つ置きます。(\\M は、月ではなく文字の M として解釈されます。)

以下は、XmNxrtFldEnterFormatList を使用するフィールドの例です。

```
static String datelist[] = {"DD/MM/YY hh:mm:ss",
  "YYYY/MM/DD hh:mm:ss", NULL};
XtVaSetValues(field,
  XmNxrtFldEnterFormatList, datelist,
  NULL);
```

デフォルトの日付フォーマットのリスト

XmNxrtFldEnterFormatList がセットされていない場合は、次のデフォルトの日付フォーマットが(上から下の順に)使用されます。

YYYY-MM-DD hh:mm:ss
YYYY/MM/DD hh:mm:ss
DD/MM/YY hh:mm:ss
DD/MM/YYYY hh:mm:ss
DD-MM-YY hh:mm:ss
DD-MM-YYYY hh:mm:ss
MMM DD, YYYY hh:mm:ss
hh:mm:ss
hh:mm A
(現在のロケールの)date/timeフォーマット
(現在のロケールの)time フォーマット

未指定の日付エレメント

ユーザが日付を指定する際に不完全な情報があった場合は、次のデフォルト値が使用されます。

日、月	01
年	現在の年
時間	現在の時間
秒、分	00

時間、分、秒が全て提供されていない場合は、現在の時刻が使用されます。年と月の両方がない場合は、現在の年と月が使用されます。

編集フォーマットと表示フォーマット

XmNxrtFldDisplayFormatString で指定される表示フォーマットは、XmNxrtFldEnterFormatList で指定されるリストの最後に自動的に追加されます。これにより、date フィールドに表示される値は、編集中に日付として認識されます。

XmNxrtFldEnterFormatList で指定されるリストの最初のフォーマットは、編集中の日付の表示に使用されます。XmNxrtFldEnterFormatList が定義されていなく、表示フォーマットが XmNxrtFldDisplayFormatString で指定されている場合は、表示フォーマットが編集フォーマットとして使用されます。(XmNxrtFldEditFormat リソースは、date フィールドには使用されません。)

XmNxrtFldDisplayFormatString の詳細は、3.8 に解説されています。

3.4 検証

XRT/field は、ユーザがキー入力した時(keystroke action)、paste 操作を行った時(paste acton)、ユーザがフィールドを出る時(user exit action)、プログラムが値を変更する時(programmatic change action)に、フィールドに記憶されている値を検証します。検証が XRT/field の編集プロセスのどの時点で行われるかは、図 16(P30)で確認してください。

フィールドで以下の検証が行われるように指定することができます。

- フィールドの最大行
- フィールドの行の最大の長さ
- フィールドの有効･無効文字
- 検証マスク
- データ型
- フィールドの値の範囲
- フィールドの有効な値のリスト(pick list)
- 検証コールバック

検証は、上記のリストの順に実行されます。

値に無効のフラグが立てられると、XRT/field はエラー処理(3.5 参照)を行います。

検証リソースの変更

検証プロセスをコントロールするリソース(XmNxrtFldMin 等)を変更すると、フィールドは自動的に再検証を行い、フィールド値が有効であるか確認します。

再検証された値が無効になっても、XmNxrtFldInternalValue の値は変わりません。

検証結果の取得

XmNxrtFldIsValid リソースは、現在表示されている値が有効であるかどうかを示します。

無効フィールドの内部値

フィールド値が無効の場合、XmNvalue は XmNxrtFldInternalValue にコピーされます。XmNvalue がコピーできない場合(データ型が違っている場合等)、XmNxrtFldInternalValue のデフォルト値が使用されます。このデフォルトは、ウィジェットのサブクラスに依存します。

- date と string フィールドでは、NULL
- integer フィールドでは、XRTFLD_MAXLONG
- float と currency フィールドでは、XRTFLD_MAXDOUBLE

3.4.1 行数

フィールドが string フィールドで、XmNeditMode リソースが XmMULTI_LINE_EDIT にセットしてある場合、ユーザはフィールドに複数行のテキストを記憶させることができます。テキストの行数の最大数を指定するには、XmNxrtFldMaxLines リソースを使用します。

デフォルトでは、XmNxrtFldMaxLines は XRTFLD_MAXINT にセットされており、フィールドの行数の制限はありません。

XmNeditMode の詳細は、3.11 に解説されています。

3.4.2 行の長さ

フィールドの各行の最大の長さを指定するには、XmNxrtFldMaxLineLength を使用します。

デフォルトでは、XmNxrtFldMaxLineLength は XRTFLD_MAXINT にセットされており、フィールドの行の長さに制限はありません。

無効な行の長さが検知された場合、XmNwordWrap リソースが False(デフォルト)の時にのみ、XmNxrtFldErrorCallback で指定されたコールバックが呼び出されます。

XmNxrtFldMaxLineLength の動作は、XmText リソースの XmNmaxLenght の場合と異なります。XmNmaxLength リソースは、ある行ではなく XmText ウィジェットまたはフィールド全体分の newline も含む最大文字数を指定します。XmNxrtFldMaxLineLength がセットされると、XmNmaxLength は無視されます。

3.4.3 有効･無効な文字

フィールドに記憶される値が、指定された文字で構成される場合にのみ有効であるように指定することができます。これには、次の二つの方法があります。

- 有効な値に使用される文字のリストには、XmNxrtFldValidChars を使用
- 有効な値に使用できない文字のリストには、XmNxrtFldInvalidChars を使用

例えば、あるフィールドに数字のみが受け入れられるようにするには、次のように XmNxrtFldValidChars をセットします。

```
XtVaSetValues(field,
  XmNxrtFldValidChars, "0123456789",
  NULL);
```

次の例は、XmNxrtFldInvalidChars を使用して、#または%が含まれる値を無効にしています。

```
XtVaSetValues(field,
  XmNxrtFldInvalidChars, "#%",
  NULL);
```

XmNxrtFldValidChars と XmNxrtFldInvalidChars の両方がセットされている時は、XmNxrtFldInvalidChars で指定されているリストが最初に調べられます。

デフォルトでは、XmNxrtFldValidChars と XmNxrtFldInvalidChars は NULL にセットされ、値は調べられません。

3.4.4 検証マスク

3.4.3 にある文字リストでは、どの文字が有効または無効であるかを指定できますが、フィールド内のどの位置にどの文字が来るかは指定できません。この場合、XmNxrtFldMask リソースを使用して、*検証マスク(validation mask)*を指定します。この検証マスクは、string や date フィールドのみで使用できます。

検証マスクはテキスト・ストリングです。検証マスクの各文字は、フィールドの値に対応し、そのフィールドの位置でどの文字が有効であるか指定します。

次の文字が、検証マスクの一部として使用できます。

#	このフィールド位置では、数字、-と+、カンマ、小数点が有効。
@	このフィールド位置では、数字が有効。
H	このフィールド位置では、16 進数(数字と A-F)が有効。
U	このフィールド位置では、アルファベットが有効。小文字は大文字に変換される。
L	このフィールド位置では、アルファベットが有効。大文字は小文字に変換される。
A	このフィールド位置では、アルファベットが有効。大文字と小文字変換は行われない。
*	このフィールド位置では、全ての文字が有効
?	このフィールド位置では、XmNxrtFldValidChars リソースで指定されているリストの文字が有効
\\	このバックスラッシュの後に来る文字は、上記のリストにある文字でも、その文字として処理される。

マスクによる検証は、ユーザが入力した時、ウィジェットを出た時、プログラムがフィールドの値を変更した時に行われます。

次は、マスクを定義する例です。

```
XtVaSetValues(field,
  XmNxrtFldMask, "@@@*AAA",
  NULL);
```

この例の“@@@*AAA”は、フィールドは全体が 7 文字で、最初の 3 文字は数字、4 番目はどの文字も使用可能、最後の 3 文字はアルファベットという構成を指定します。

注意: ドラッグ&ドロップ操作が行われ、ペーストされた値がマスクの長さより長い場合は、その値は途中で切り取られます。

文字(literal)をマスクに使用する

マスク内に使用されている文字で、上記のリスト以外の文字は、*literal*(その文字どおり)として処理されます。literal はフィールド内に表示され、入力の際のガイドの役割を果たします。この literal はユーザが削除することはできません。

例えば、“@@@-@@@”がフィールドに指定されている場合、ダッシュ(-)は literal として処理され、図 25 のようにフィールド内に表示されます。



図 25　literal が表示されている空のフィールド

最初の 3 文字が入力され検証された後、カーソルは、ダッシュの後ろにジャンプします。(図 26)



図 26　部分的に入力された literal のあるフィールド

ユーザはダッシュをタイプする必要はありません。(このダッシュを消すことはできません。)

プロポーショナル・フォントが使用されている場合、フィールドに表示されている literal は、ユーザが文字を入力すると自動的にシフト(移動)します。

literal と記憶された値

マスクの一部の literal はフィールド内に表示はされますが、XmNxrtFldInternalValue リソースに記憶される値の一部にはなりません。literal は記憶されませんので、表示されるマスクを変更する際は、プログラムを変更する必要がありません。図 27 は、異なった局番のマスクを使用した電話番号のウィジェットです。

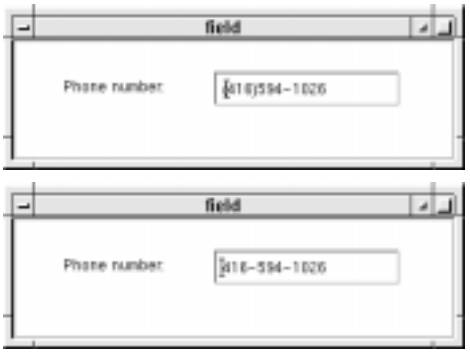


図 27　電話番号ウィジェット

上記の両方とも、XmNxrtFldInternalValue の値は、“4165941026”です。

フィールドに記憶されている、(literal も含む)実際の値にアクセスするには、XmNvalue リソースの値を取り出します。この例の XmNvalue は、上は“(416)594-1026”、下は“416-594-1026”です。

マスクと上書きモード

マスクが XmNxrtFldMask で指定される時は、常に、上書きモードにセットされます。XmNxrtFldOverstrike(4.14 参照)の値は無視されます。

部分的なマスクとの適合

デフォルトでは、マスクが定義されている時、XRT/field は下記の場合にのみフィールド値が有効であると処理します。

- フィールド内の全ての文字がマスクの対応する文字に適合する
- 値の長さがマスクの長さと同じ

値が、上記の一番目には従うが二番目の条件には従わない場合は、部分的に適合(partial match)している状態です。XRT/field に、部分的な適合も有効であると処理させるには、XmNxrtFldMatchMask リソースを使用します。

XmNxrtFldMatchMask は、次の値にセットできます。

XRTFLD_MATCH_PARTIAL_L_TO_R	部分的な適合を受け付ける: 左から右に向かって適合するか調べる

XRTFLD_MATCH_PARTIAL_R_TO_L	部分的な適合を受け付ける:右から左に向かって適合するか調べる
XRTFLD_MATCH_ALL	完全な適合のみ受け付ける(デフォルト)

以下は、電話番号にこのリソースを適用している例です。

```
XtVaSetValues(field,
  XmNxrtFldMask, "@@@-@@@@ ext @@@",
  XmNxrtFldMatchMask, XRTFLD_MATCH_PARTIAL_L_TO_R,
  NULL);
```

この場合、左から右の部分的な適合が指定されているので、内線番号が提供されていない電話番号でも有効とされます。

適合しない部分がスペースである場合も、部分的な適合として処理されます。

```
XtVaSetValues(field,
XmNxrtFldMask, "@@@-@@@@ ext @@@",
XmNxrtFldMatchMask, XRTFLD_MATCH_PARTIAL_L_TO_R,
XmNxrtFldInternalValue, "@@@@@@@    ",
NULL);
```

上の例では、電話番号が適合し、スペースは無視されます。

視覚的なキューの指定

マスクを使用したフィールドが最初に表示された時、視覚的なキュー(指示)を指定しておくと、何を入力したらよいかが明確になります。例えば、日付を“DD/MM/YY”のフォーマットで入力する場合、次のようにしてキューを指定できます。

```
XtVaSetValues(field,
  XmNvalue, "DD/MM/YY",
  XmNxrtFldMask, "@@/@@/@@",
  NULL);
```

図 28 は、上で指定したフィールドです。



図 28　DD/MM/YY テンプレートが含まれた date フィールド

XmNvalue リソースは、フィールドに表示されるテキスト・ストリングを持ちます。このストリング“DD/MM/YY”は、date フィールドでの有効な値ではありません。このストリングは、そこに何が入力されるべきかを示すテンプレートの役割を持ちます。

テンプレートは有効な値ではありませんので、ウィジェットは XmNxrtFldErrorBackground と XmNxrtFldErrorForeground リソースで指定される色を使用して表示されます。

ペーストされた値の検証

マスクがターゲットとなるフィールドにセットされている時、ペーストまたはドラッグ&ドロップされた値は、下記の条件を満たした場合にのみ有効です。

- ペーストされた値は、マスクで定義されている有効な文字のみを持つ
- ペーストされた値は、マスクで指定されている位置に literal を持ち、その他の位置にも有効な文字のみを持つ

どちらの場合も、左から右方向への部分的な適合(partial match)が有効とされ、マスクより長い値は、切り取られます。

3.4.5 データ型

XmNxrtFldMask によってマスクが指定されていない場合は、XRT/field が自動的にデータ型の検証を行います。例えば、integer フィールドの値を“3.14”に変更しようとすると、“3.14”は整数でないので、無効のフラグが立てられます。

date フィールドに記憶される日付は、正しい日付のみが有効とされます。編集フォーマットが DD/MM/YY の場合、“31/04/95”は、4 月 31 日は存在しないので無効とされます。

3.4.6 値の範囲

XmNxrtFldMin と XmNxrtFldMax リソースで値の最小と最大を指定して、値の検証を行うこともできます。

string フィールドでは、XmNxrtFldMin と XmNxrtFldMax の値はテキスト･ストリングです。フィールドに割り当てられた値は、*strcoll()*ルーチンを使用して、アルファベット順に XmNxrtFldMin と XmNxrtFldMax に比較されます。例えば、XmNxrtFldMin が“alpha”で、XmNxrtFldMax が“beta”であれば、値“bat”は、“alpha”と“beta”の間にあるので、有効とされます。

date フィールドでは、XmNxrtFldMin と XmNxrtFldMax は、*struct tm*(ANSI time structure)型の構造体のポインタです。日付は、XmNxrtFldMin 以後、XmNxrtFldMax 以前であれば有効です。

他のフィールドでは、値が XmNxrtFldMin 以上 XmNxrtFldMax 以下であれば有効です。

XmNxrtFldMax がセットされ、XmNxrtFldMin がセットされていない場合、フィールド内の値は最大値を持ちますが、最小値は持ちません。同様にして、XmNxrtFldMin がセットされ XmNxrtFldMax がセットされていなければ、値は、最小値は持ちますが最大値は持ちません。

keystroke のイベントが起こった後は、（ユーザが値を入力し終えていないので)範囲の検証は行われません。

3.4.7 有効な値のリスト

XmNxrtFldPickList は、フィールドの有効な値のリスト(pick list)を指定します。このリソースが定義され、XmNxrtFldMatchPickList が True にセットされている場合は、このリストのエレメントにマッチした値のみが有効です。

string フィールドでは、XmNxrtFldPickList の値はテキスト・ストリングの NULL 終端配列です。

```
String stringlist[] = {
  "valid1",
  "valid2",
  NULL };

XtVaSetValues(field,
  XmNxrtFldPickList, stringlist,
  XmNxrtFldMatchPickList, True,
  NULL);
```

ここでは、pick list は valid1 と valid2 というストリングで構成されています。フィールドに pick list が指定され、フィールドが空の場合は、pick list の最初のエレメントがフィールドの初期値になります。上の例では、ウィジェットの初期値は valid1 です。integer フィールドでは、XmNxrtFldPickList の値は *long* 型の値の配列です。この配列の最初のエレメントは、pick list のエレメントの数を指します。

```
long intlist[] = { 3,
  40,
  57,
  118 };

XtVaSetValues(field,
  XmNxrtFldPickList, intlist,
  XmNxrtFldMatchPickList, True,
  NULL);
```

ここでは、pick list は整数 40、57、118 で構成されています。

float と currency フィールドでは、XmNxrtFldPickList の値は、*double* 型の値の配列です。配列の最初の値は、pick list のエレメントの数を指します。

date フィールドでは、XmNxrtFldPickList の値は、*struct tm*(ANSI time structure)型の構造体のポインタの NULL 終端配列です。

keystroke のイベントが起こった後は、(ユーザが値を入力し終えていないので)pick list の検証は行われません。

pick list とマスク

XmNxrtFldMask を使用してフィールドのマスクが定義されている場合、pick list の各エレメントが、マスクの literal(文字)を含まないようにしてください。

```
String phonenumbers[] = {
  "4165551212",
  "4165941026",
  NULL };
XtVaSetValues(field,
  XmNxrtFldPickList,       phonenumbers,
  XmNxrtFldMatchPickList, True,
  XmNxrtFldMask,       “(@@@)@@@-@@@@”,
  NULL);
```

この例では、マスクに指定されている括弧とダッシュは、pick list のエレメントに含まれていません。

3.4.8 検証コールバック

XmNxrtFldValidateCallback リソースを使用して検証コールバックを定義することにより、フィールドに独自の検証方法を提供することができます。

各検証コールバックには、次の構造体のポインタが渡されます。

```
typedef struct {
  XrtFldReason    reason;      /* read-only */
  XEvent      *event;      /* read-only */
  XrtFldAction    action;      /* read-only */
  int         num_calls;   /* read-only */
  XmTextPosition  startPos, endPos; /* read-only */
  XmTextPosition  currInsert; /* read-only */
  XmTextPosition  newInsert;
  XmTextBlock         text;
  Boolean         doit;        /* initially True */
  String      value;
} XrtFldValidateCallbackStruct;
```

この構造体のエレメントでよく使用されるものは、*action*、*text*、*value* と *doit* です。(この構造体のその他のエレメントに関しては、Appendix A を参照してください。)

action は、現在処理されているアクションです。XRTFLD_ACTION_KEYSTROKE (keystroke action)、XRTFLD_ACTION_EXIT(user exit action)、XRTFLD_ACTION_PASTE(paste action)、XRTFLD_ACTION_SETVALUE (programmatic change action)のどれかです。(これらのアクションと XRT/field の編集プロセスは、2.7 を参照してください。)

text は検証されるテキストを持ちます。

value は、フィールドに記憶される値全体を持ちます。この値は、その前に指定された変換や補完も含みます。

doit は、*text* に記憶されている値が有効かどうかを示します。指定されたコールバック・ルーチンが、値が無効であると決定した場合は、ルーチンが *doit* を False にセットするようにしてください。

次のコールバック・ルーチンは、"MM/DD/YY"フォーマットの日付が一年の最初の 6 ヶ月に該当するかを調べます。後半の月であれば、無効のフラグが立てられます。

```
void
validateCB(Widget widget, XtPointer client_data,
  XrtFldValidateCallbackStruct *cs)
{
  int value;

  if (cs->action == XRTFLD_ACTION_KEYSTROKE) return;
  if (cs->text->ptr == NULL) return;
  value = 10 * (cs->text->ptr[0] - '0') +
    (cs->text->ptr[1] - '0');
  if (value > 6) {
    cs->doit = False;
  }
}
```

検証コールバックは、XmNvalue と XmNxrtFldInternalValue リソースをセットすることができません。(セットしようとしても無視されます。)フィールドに記憶されている値は、XrtFldValidateCallbackStruct 構造体からのみアクセスできます。

3.5 無効フィールドの処理

フィールドが検証によって無効と判断されると、次の事が起こります。

- XRT/field は、定義されているエラー・コールバックを開始する
- フィールドは、エラーの場合の背景色と前景色を使用して表示される

エラーが検知されフィールド値が表示される場合は、値はそのまま表示されます。XmNxrtFldEditFormat で指定されている編集フォーマットと XmNxrtFldDisplayFormat で指定されている表示フォーマットは使用されません。

エラー・コールバック

フォーカスを失うフィールドに、無効な値が検知された時に開始されるエラー・コールバックを指定するには、XmNxrtFldErrorCallback を使用します。このリソースは、標準の XtCallbackList を持っています。

各エラー・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 {
  XrtFldReason    reason;      /* read-only */
  XEvent      *event;      /* read-only */
  int         text_reason;     /* read-only */
  XrtFldAction    action;      /* read-only */
  XrtFldError         error;
  Boolean         is_modified;
  XmTextPosition  startPos, endPos; /* read-only */
  XmTextPosition  currInsert; /* read-only */
  XmTextPosition  newInsert;
  XmTextBlock         text;
  XmString        error_string;    /* read-only */
  Boolean         display;     /* initially False */
  Boolean         traverse;    /* initially True */
} XrtFldErrorCallbackStruct;
```

この構造体で最もよく使用されるエレメントは、*action*、*error*、*text*、*error_string* です。(この構造体のその他のエレメントは、Appendix A を参照してください。)

action は、現在処理されているアクションです。XRTFLD_ACTION_KEYSTROKE (keystroke action)、XRTFLD_ACTION_PASTE(paste action)、XRTFLD_ACTION_EXIT(user exit action)、XRTFLD_ACTION_SETVALUE (programmatic change action)のどれかです。(これらのアクションと XRT/field の編集プロセスは、2.7 を参照してください。)

error は、どの検証が無効な値を検知したかを示し、次の内のどれかです。

XRTFLD_ERROR_EXCEED_LENGTH	**値の行の長さが XmNxrtFldMaxLineLength より長い**
XRTFLD_ERROR_EXCEED_LINES	**値の行数が XmNxrtFldMaxLines より多い**
XRTFLD_ERROR_IS_NULL	**フィールドは空で、XmNxrtFldAllowNULL が False にセットされている**
XRTFLD_ERROR_INVALID_CHAR	**値の文字が XmNxrtFldInvalidChars リストにあるか、XmNxrtFldValidChars リストにない**
XRTFLD_ERROR_INVALID_DATATYPE	**値の文字が、フィールドのデータ型に不適当** (integer **フィールドにアルファベット文字等**)
XRTFLD_ERROR_NOMATCH_MASK	**値が、XmNxrtFldMask で指定されているマスクと適合しない**
XRTFLD_ERROR_NOMATCH_PICKLIST	**値が、XmNxrtFldPickList で指定されている** pick list **にない**
XRTFLD_ERROR_OUT_OF_RANGE	**値が、XmNxrtFldMax で指定されている上限より大きいか、XmNxrtFldMin で指定されている下限より小さい**
XRTFLD_ERROR_VALIDATECB	**値が、検証コールバックにより無効というフラグが立てられている**(*doit* **構造体メンバが False にセットされている**)

XRTFLD_ERROR_INVALID_DATATYPE 以外のエラーを置き換えるには、*error* を XRTFLD_ERROR_NONE にセットします。これは、XRT/field に値を有効と処理するように指示します。

error_string は、エラーが起こった時にアプリケーションによって表示されるメッセージで、XmNxrtFldErrorString リソースの値です。このメッセージは、XmString(複合ストリング)型のメッセージです。次のエラー・コールバックは、XmLabel ウィジェットにエラー・メッセージを表示します。

```
void
errorCB(Widget widget, XtPointer client_data,
  XrtFldErrorCallbackStruct *cs)
{
  XtVaSetValues(error_label,
  XmNlabelString, cs->error_string,
  NULL);
}
```

エラー表示の色

エラーを表示する色は、3.10 に解説されていますので参照してください。

3.6 変換

フィールドに記憶されている値は変換することができます。変換は、ユーザがフィールドを出た時(user exit action)、キーをタイプした時(keystroke action)、ペースト操作が行われた時(paste action)、プログラムによって値が変更された時(programmatic change action)に指定することができます。

変換は、値が検証がされる前に行われます。(図 16 に、変換が編集プロセスのどの段階に来るかが示されています。)

リソースを使用して、次の変換を指定することができます。

- 大文字・小文字変換は、XmNxrtFldCase リソースを使用して指定します。XRTFLD_CASE_UPPER にセットされると、フィールドに記憶されている値は、大文字に変換されます。XRTFLD_CASE_LOWER の場合は小文字に変換されます。デフォルトでは、XmNxrtFldCase は、XRTFLD_CASE_AS_IS にセットされ、大文字・小文字変換は行われません。
- ストリングの各文字の大文字・小文字変換は、検証マスクで指定します。マスクと検証に関しては、3.4.4 を参照してください。(マスクによる大文字・小文字変換は、XmNxrtFldCase で指定される変換より優先されます。)

その他の変換は、変換コールバックによって指定してください。

変換コールバック

変換コールバックを登録するには、XmNxrtFldConvertCallback で指定されるリストに新しいコールバックを追加してください。このリソースは、標準の XtCallbackList を持っています。

各変換コールバック・ルーチンには、次の構造体のポインタが渡されます。

```
typedef struct {
  XrtFldReason    reason;      /* read-only */
  XEvent      *event;      /* read-only */
  XrtFldAction    action;      /* read-only */
  XmTextPosition  startPos, endPos; /* read-only */
  XmTextPosition  currInsert; /* read-only */
  XmTextPosition  newInsert;
  XmTextBlock         text;
  String      value;
} XrtFldConvertCallbackStruct;
```

action は、現在処理されているアクションで、XRTFLD_ACTION_KEYSTROKE (keystroke action)、XRTFLD_ACTION_EXIT(user exit action)、ペースト操作が行われた時(paste action)、XRTFLD_ACTION_SETVALUE (programmatic change action)のどれかです。

text は、変換されるテキストです。その内容は、*action* の値に依存します。*action* が XRTFLD_ACTION_KEYSTROKE であれば最後にタイプされた文字、XRTFLD_ACTION_PASTE であれば最後にペーストかドラッグ、ドロップされた文字です。*action* が XRTFLD_ACTION_EXIT または XRTFLD_ACTION_SETVALUE であれば、*text* はフィールドの内容全てを指します。

startPos は、フィールドの開始点と *text* の最初の文字の間にある文字数です。*endPos* は、フィールドの開始点と *text* の最後の文字の間にある文字数です。

text が一文字であれば、*startPos* と *endPos* は同じです。

currInsert は、カーソルの現在の位置を指定します。*newInsert* はカーソルを移動させた時の新しい位置を指定します。(デフォルトでは、*newInsert* は *currInsert* と同じです。)

value メンバは値全体です。このフォーマットは *action* に依存します。変換コールバックが、ユーザがフィールドを出るアクション(user exit action)または、キーをタイプしたアクション(keystroke action)によって開始される場合は、*value* は、(`XmNxrtFldEditFormat` で指定される)フィールドの編集フォーマットで提供されます。ペースト操作のアクション(paste action)の場合は、*value* は、キーボード・フォーカスに従って、編集フォーマットまたは表示フォーマットで表示されます。変換コールバックが、プログラムによる変更(programmatic change action)によって開始される場合は、*value* は、(`XmNxrtFldDisplayFormat` で指定される)フィールドの編集フォーマットで提供されます。

次の例は、string フィールドの変換コールバックです。ユーザによる各キー入力が調べられ、始めにタイプされたキーは大文字に変換されます。それ以外の場合は小文字に変換されます。

```
void
convertCB(Widget widget, XtPointer client_data,
  XrtFldConvertCallbackStruct *cs)
{
  if (cs->action != XRTFLD_ACTION_KEYSTROKE) return;
  if (cs->text->ptr == NULL) return;
  if (cs->currInsert == 0) {
    cs->text->ptr[0] = toupper(cs->text->ptr[0]);
  } else {
    cs->text->ptr[0] = tolower(cs->text->ptr[0]);
  }
}
```

コールバック・ルーチンは、`XmNvalue` または `XmNxrtFldInternalvalue` をセットすることはできません。(セットしようとしても無視されます。)コールバックからフィールド値にアクセスまたは変更するには、コールバック構造体の *text* または *value* メンバを操作します。(*text* と *value* の両方が修正された場合は、*value* が使用されます。)フィールド値を変更するには、古い値で使用される記憶域を解放し、新しい値に記憶域を割り当て、その領域を *text* または *value* で指定されるポインタに割り当てます。

3.7 補完

ユーザがウィジェットを出る時、フィールドに記憶されている不完全な値に情報を追加することができます。この補完は、変換が行われた後、そして検証が行われる前に実施されます。(補完が、編集プロセスのどの段階に来るかは、図 16 で確認してください。)

値を補完するには、`XmNxrtFldCompleteCallback` リソースをセットして補完コールバックを指定してください。(このリソースは、標準の `XtCallbackList` を持っています。)

各補完コールバックには、次の構造体のポインタが渡されます。

```
typedef struct {
  XrtFldReason    reason; /* read-only */
  XEvent      *event; /* read-only */
  String      value;
} XrtFldCompleteCallbackStruct;
```

value はフィールドの新しい値です。(補完は、この値を修正することにより行われます。)値は、それを解放し、新しい値を割り当て、セットすることによっても変更することができます。*value* は、フィールドの編集フォーマットで提供されます。(編集フォーマットは 3.2 に解説されています。)

次は、補完コールバックの例です。このコールバックは、ファイル名に(接尾時が提供されていない場合)“.c”の接尾辞を追加します。

```
void
completeCB(Widget widget, XtPointer client_data,
  XrtFldCompleteCallbackStruct *cbs)
{
  String value;

  if (cbs->value && !strstr(cbs->value, ".c")) {
    value = XtMalloc(strlen(cbs->value) + 3);
    strcpy(value, cbs->value);
    strcat(value, ".c");
    XtFree(cbs->value);
    cbs->value = value;
  }
}
```

値の長さはこのコールバックによって変更されるので、古い値を含むストリングは解放され、新しい値のストリングが割り当てられます。

補完コールバックは、XmNvalue や XmNxrtFldInternalValue をセットすることはできません。(セットしようとしても無視されます。)フィールドに記憶されている値は、XrtFldCompleteCallbackStruct 構造体からのみアクセスできます。この値を変更するには、*value* で指されているストリングを解放した後、新しい値を持ったストリングを作成し、*value* がそれを指すようにセットします。

3.8 表示フォーマットの指定

integer、float、currency フィールドでは、XmNxrtFldDisplayFormat を設定して、値が編集されていない時の表示法を指定することができます。

XmNxrtFldDisplayFormat は、XmNxrtFldEditFormat と同じ値を持つことができます。これらの値については、3.2 の編集フォーマットに関する解説を参照してください。

XmNxrtFldDisplayFormat によって指定されるフォーマットは、値が有効な場合にのみ使用されます。値が無効な場合は、ユーザによってタイプされた通りに表示されます。

XmNxrtFldDisplayFormat が指定されていなかったり、XRTFLD_FORMAT_DEFAULT に定義されていると、値は、編集の時に使用されたフォーマットで表示されます。

次の例では、編集と表示のそれぞれの場合のフォーマットを指定しています。

```
XtVaSetValues(field,
  XmNxrtFldInternalValue, 255,
  XmNxrtFldEditFormat,    XRTFLD_FORMAT_GENERAL,
  XmNxrtFldDisplayFormat, XRTFLD_FORMAT_OCTAL,
  NULL);
```

ここでは、ウィジェットに記憶されている値が 8 進数表記で表示されていますが、編集中は 10 進数表記です。ユーザがフィールドの編集を開始すると、その値は 10 進数数表記になり、表示される前に自動的に 10 進数から 8 進数に変換されます。

値が 8 進数で表示されると、数値が 8 進数であるということを示す `0o` が表示されます。

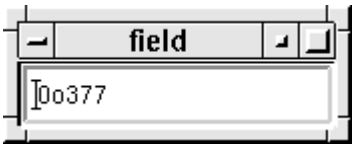


図 29　8 進数で表示されている値

同様に、`0b` はバイナリ、`0x` は 16 進数を示します。

表示フォーマットのカスタマイズ

値を integer、float、currency フィールドで表示する時、値が編集されていない場合の表示法を指定する *sprintf* フォーマット・ストリングを定義することができます。`XmNxrtFldDisplayFormat` を `XRTFLD_FORMAT_GENERAL` にセットし、`XmNxrtFldDisplayFormatString` を使用する *sprintf* ストリングにセットしてください。

例えば、整数値を 10 文字フィールドに右揃えに表示するには、次のようにします。

```
XtVaSetValues(field,
  XmNxrtFldDisplayFormat, XRTFLD_FORMAT_GENERAL,
  XmNxrtFldDisplayFormatString, "%10ld",
  NULL);
```

`XmNxrtFldDisplayFormatString` に使用される *sprintf* ストリングは、ウィジェットが integer フィールドの場合には、整数フォーマット指定子(%d、%o 等)を持っていなければなりません。float と currency フィールドには、浮動小数点フォーマット指定子(%e、%f、%g)が必要です。フラグ、フィールド幅、精度、サイズ等の変換の指定は、フォーマット指定子と一緒に提供することができます。*sprintf* ストリングは literal(マスクの文字)を含むこともできます。

date フィールドのフォーマット

XmNxrtFldDisplayFormatString リソースは、date フィールドの表示をコントロールします。この場合、リソースは、次の文字(またはその組み合わせ)から構成される文字ストリングを持ちます。

D	日(1-31)
DD	二桁の日表記、日が一桁の場合は 0 を先頭に置く(例: 01、02)
W	省略された曜日
WW	曜日
J	Julian(ユリウス暦の)日(1-366)
JJ	二桁の Julian 日、一桁の場合は 0 を先頭に置く
M	月(1-12)
MM	二桁の月表記、月が一桁の場合は 0 を先頭に置く(例: 01、02)
MMM	省略された月名
MMMM	月名
Y、YY	今世紀での年(00-99)
YYYY	世紀を含む年
H	時間(0-12)
HH	二桁の時間(0-12)、一桁の場合は 0 を先頭に置く
h	時間(0-23)
hh	二桁の時間表記、一桁の場合は 0 を先頭に置く(例: 01、02)
m	分(0-59)
mm	二桁の分表記、一桁の場合は 0 を先頭に置く(例: 01、02)
s	秒(0-59)
ss	二桁の秒表記、一桁の場合は 0 を先頭に置く(例: 01、02)
A、P	AM/PM 表記
Z	タイムゾーンの略
\\	エスケープ文字(次の文字は、その文字通りに解釈する)

その他の文字は literal(文字どおり)として処理され、そのまま表示されます。

タイム・ゾーンは使用しているマシンに依存します。XrtDataField ウィジェットで変更することはできません。

このリソースが指定されていない場合は、XmNxrtFldEnterFormatList に記憶されているリストの最初のアイテムが使用されます。

3.9 Exit コールバック

値が検証されたフィールドをユーザが出ようとする時、exit コールバックを指定することにより、検証後のプロセスを追加することができます。これらのコールバックは、標準の XtCallbackList を持つ、XmNxrtFldExitCallback リソースによって参照されます。

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 {
  XrtFldReason    reason;      /* read-only */
  XEvent      *event;      /* read-only */
  int         text_reason;     /* read-only */
  Boolean         is_modified;
  String      value;       /* read-only */
  Boolean         traverse;    /* initially True */
} XrtFldExitCallbackStruct;
```

value はフィールドの新しい値(この値が表示されます)。この値は、XmNxrtFldDisplayFormat で指定されたフォーマットを使用します。

text_reason は、どの XmText コールバックがこの exit コールバックを開始させたかを示し、次のうちのどちらかです。

XmCR_LOSING_FOCUS	XmNlosingFocusCallback によって開始
XmCR_ACTIVATE	XmNactivateCallback によって開始

text_reason は、ユーザが Return キーを押してフィールドを出ようとする時に XmCR_ACTIVATE にセットされます。その他の方法によってフィールドを出る場合は、*text_reason* は XmCR_LOSING_FOCUS にセットされます。(XmText コールバックとフィールドのコールバックとの関係は 4.15 に解説されています。)

is_modified は、XmNxrtFldIsModified の現在値を持ち、値が最後にユーザまたはプログラムによって変更されたかを示します。*is_modified* を変更すると、このリソースの値が変更されます。

traverse は、ユーザが Return または Tab キーを押すことによりフィールドの外へトラバースできるかどうかを示します。デフォルトでは、*traverse* は True になっています。

traverse と *text_reason* を一緒に使用して、ユーザが Return キーを押してもトラバースできないようにすることができます。

```
if (cs->text_reason == XmCR_ACTIVATE) {
  cs->traverse = False;
}
```

これにより、フィールド・ウィジェットは、XmText ウィジェットと同様な動作を持つことができます。(XmText では、ユーザが Return を押してもトラバースできません。)

exit コールバックは、フィールドをグループとしてまとめて調べることができないので、検証後のプロセスを行うには適さない場合があります。幾つかのフィールドを一度に調べる場合は、*XrtFldCheckFields()*手続きを使用してください。*XrtFldCheckFields()* は、4.11 に解説されています。

3.10 フィールドの色

背景色

フィールドの背景の色は、XmNbackground リソースで指定されます。背景色を変更するには、このリソースを変更してください。

前景色

フィールドの前景の色は、XmNforeground リソースで指定されます。前景色を変更するには、このリソースを変更してください。

エラー表示の色

無効な値が検知されると、フィールドの背景と前景色は、XmNxrtFldErrorBackground と XmNxrtFldErrorForeground で指定される色にセットされます。デフォルトでは、これらの色は、赤と白です。(モノクロのモニタでは、黒と白に表示されます。)エラーが起きてもフィールドの色が変更されないようにするには、XmNxrtFldErrorBackground と XmNxrtFldErrorForeground を XRTFLD_UNCHANGED_COLOR にします。

デフォルト値の色

XRT/field では、デフォルト値を持つフィールドに異なった背景と前景色を指定することができます。デフォルト値のフィールドの色を指定するには、XmNxrtFldDefaultBackground と XmNxrtFldDefaultForeground のリソースをセットしてください。デフォルト値のフィールドの色を変えることにより、ユーザは、どのフィールドがまだセットされていないかを一目で判断することができます。

XmNxrtFldDefaultBackground と XmNxrtFldDefaultForeground がセットされていない場合は、デフォルト値を持つウィジェットは、通常のフィールドの背景と前景色を使用して表示されます。

ウィジェットのデフォルト値は、4.1 に詳細が解説されています。

pixel データ型

フィールドによって認識される色のリソースは、Pixel 型の値を持っています。

色名(“red”等)を Pixel 型の値に変換するには、XtVaTypedArg を使用します。下の例では、エラーの背景色を green に指定します。

```
XtVaSetValues(field,
  XtVaTypedArg, XmNxrtFldErrorBackground,
  XmRString, "green", strlen("green")+1,
  NULL);
```

XtVaTypedArg または Pixel データ型に関する詳細は、Motif 関連の資料を参照してください。

3.11 複数行のテキスト

デフォルトでは、フィールドは 1 行分のテキストを扱います。newline 文字がユーザによって入力されると、newline 文字とその後の入力は無視されます。

string フィールドでは、複数行のテキストを入力することができます。`XmNeditMode` リソースを `XmMULTI_LINE_EDIT` にセットしてください。1 行のテキスト入力に戻すには、`XmNeditMode` をデフォルト値の `XmSINGLE_LINE_EDIT` にセットします。

注意: 複数行のテキスト・フィールドを使用する場合、Motif 1.2 のプログラムでは、必ず `XmNrows` を 1 より大きい数にセットしてください。Motif 1.2 のバグにより、複数行をサポートしても 1 行しか表示しないウィジェットは、正しく動作しません。

4

XRT/field の高度なプログラミング

デフォルトのフィールド値 ■ 入力プロンプトとコールバック■ XRT/table と一緒に使用する自動マッチング ■ ドロップ・ダウン・リスト ■ XrtComboBox のプログラミング XrtSpinBox のプログラミング ■ XrtCalendar のプログラミング■ 日付エディタ キーにアクションを割り当てる ■ 複数のフィールドのチェック ■ 日付の変換 スクロールのコントロール■マスク文字の定義 ■ XrtField と XmText コールバック XRT/field で C++ を使用する■ UIL の使用 ■ リソース値をストリングに変換 国際化

ここでは、XRT/field の基本操作を習得した方へ、更に進んだ使用法を紹介します。

4.1 デフォルトのフィールド値

XRT/field では、フィールドが空の時に表示される*デフォルト値(default value)*を設定することができます。

string フィールドは、`NULL` または空のストリングを持つ時は、”空“です。他のサブクラスは、`NULL` や白文字(スペース、タブ、newline 文字等)を持つ時は”空“です。

`XmNxrtFldDefaultValue` リソースは、フィールドのデフォルト値を指定します。このリソースのデータ型は、`XmNxrtFldInternalValue` リソースのデータ型と同じです。

次のコードは、integer フィールドを作成し、そのデフォルト値を 42 にセットしています。

```
field = XtVaCreateManagedWidget("field",
  xmXrtIntFieldWidgetClass,    toplevel,
  XmNxrtFldDefaultValue,       42,
  NULL);
```

XmNxrtFldDefaultValue が定義されていない場合は、次のデフォルト値が使用されます。

integer フィールド	0
float と currency フィールド	0
string フィールド	NULL
date フィールド	ウィジェットが作成された時間

注意: デフォルト値を持つフィールドが空にセットされた場合は、そのフィールドはアプデートされるとデフォルト値にセットされます。

デフォルト値を表示しない

フィールドがセットされていない時、デフォルト値が表示されないようにするには、XmNxrtFldDefaultValue を次のようにセットしてください。

- integer フィールドでは、XmNxrtFldDefaultValue を XRTFLD_NOVALUE にセットする
- float と currency フィールドでは、XmNxrtFldDefaultValue を XRTFLD_NOVALUE_FLOAT にセットする
- string と date フィールドでは、XmNxrtFldDefaultValue を NULL にセットする

デフォルト値は、ユーザによってセットされる値と同様、変換、補完、検証が行えます。デフォルト値が有効であれば、XmNvalue と XmNxrtFldInternalValue はこの値にセットされます。

デフォルト値の検査

XmNxrtFldIsDefault リソースを使用して、デフォルト値と割り当てられた値を区別することができます。

XmNxrtFldIsDefault が True であれば、フィールドは空で、フィールドに記憶されている値はデフォルト値です。XmNxrtFldIsDefault が False であれば、フィールドの値は、割り当てられた値です。(このリソースは read-only です。プログラムによってセットすることはできません。)

XmNxrtFldIsDefault は、ユーザまたはプログラムによって割り当てられた値が XmNxrtFldDefaultValue で指定される値と同じでも False になります。

デフォルト値の色

XmNxrtFldDefaultBackground と XmNxrtFldDefaultForeground を使用して、デフォルト値を含むフィールドの前景色と背景色をセットすることができます。

デフォルトでは、XmNxrtFldDefaultBackground と XmNxrtFldDefaultForeground は XRTFLD_UNCHANGED_COLOR にセットされ、デフォルト値を含むフィールドは、(XmNbackground と XmNforeground で指定される)通常のフィールドと同じ色で表示されます。XRT/field の色は、3.10 に詳細が解説されています。

空のフィールド

空のフィールドを扱うことができないアプリケーションの場合、XRT/field では、空のフィールドが有効であるかどうかを指定させることができます。空のフィールドを無効にするには、XmNxrtFldAllowNull を False にセットしてください。

デフォルトでは、XmNxrtFldAllowNull は True で、空のフィールドは有効です。

4.2 入力プロンプトとコールバック

XRT/field では、フィールドがキーボードのフォーカスを得た時に、アプリケーションによって表示されるメッセージを指定することができます。このメッセージは、フィールドの*入力プロンプト(enter prompt)*と呼ばれます。

入力プロンプトを定義するには、`XmNxrtFldPromptString` リソースをプロンプトとして使用するテキスト・ストリングにセットします。(このストリングは、Motif の複合ストリング型である、XmString 型にしてください。)その後、入力コールバックを定義し、`XmNxrtFldEnterCallback` リソースで指定されるリストに追加します。

各入力コールバックには、次の構造体のポインタが渡されます。

```
typedef struct {
  XrtFldReason    reason;      /* read-only */
  XEvent          *event;      /* read-only */
  String          value;       /* read-only */
  XmString        prompt_string;  /* read-only */
} XrtFldEnterCallbackStruct;
```

prompt_string は、`XmNxrtFldPromptString` リソースの現在の値です。*value* は、ユーザがフィールドを編集した時に表示される値です。これは、編集フォーマットに変換された `XmNxrtFldInternalValue` の内容です。

下の例は、`XmNxrtFldPromptString` を使用して入力プロンプトを定義し、そのプロンプトのストリングをアプリケーションの MainWindow ウィジェットのメッセージ領域に書いています。

```
string = XmStringCreate("Enter your login userid",
  XmFONTLIST_DEFAULT_TAG);
field = XtVaCreateManagedWidget("field",
  xmXrtStringFieldWidgetClass,    toplevel,
  XmNxrtFldPromptString,      string,
  NULL);

void
enterCB(Widget widget, XtPointer client_data,
  XrtFldEnterCallbackStruct *cs)
{
  XtVaSetValues(MessageArea, XmNlabelString,
    cs->prompt_string, NULL);
}
```

4.3 XRT/table と一緒に使用する

XRT/field を XRT/table(作表ウィジェット)と一緒に使用するには、次の四つのステップに従ってください。

1 コンパイルされたプログラムをリンクする際、XRT/field のライブラリ(`libxrtfield.a`)を XRT/table のライブラリ(`libxrttable.a`)の前に置いてください。

2 表を作成する前に、*XrtTblEnableXrtField()*を呼び出してください。

```
XrtTblEnableXrtField();
...
table = XtVaCreateManagedWidget("table",
  xtXrtTableWidgetClass, toplevel,
  NULL);
```

XmNxrtTblDataTypeContext を使用して、XRT/field の型を指定します。次の型が使用できます。

XRTTBL_TYPE_FIELD_CURRENCY	currency(通貨)フィールド
XRTTBL_TYPE_FIELD_DATE	date/time(日付/時間フィールド
XRTTBL_TYPE_FIELD_FLOAT	floating-point(浮動小数点)フィールド
XRTTBL_TYPE_FIELD_INT	integer(整数)フィールド
XRTTBL_TYPE_FIELD_NONE	
XRTTBL_TYPE_FIELD_STRING	string(文字列)フィールド
XRTTBL_TYPE_FIELD_COMBO_CURRENCY	currency(通貨) combo box
XRTTBL_TYPE_FIELD_COMBO_DATE	date (日付)combo box
XRTTBL_TYPE_FIELD_COMBO_FLOAT	floating-point(浮動小数点) combo box
XRTTBL_TYPE_FIELD_COMBO_INT	integer(整数) combo box
XRTTBL_TYPE_FIELD_COMBO_STRING	string(文字列) combo box
XRTTBL_TYPE_FIELD_SPIN_CURRENCY	currency (通貨)spin box
XRTTBL_TYPE_FIELD_SPIN_DATE	date(日付) spin box
XRTTBL_TYPE_FIELD_SPIN_FLOAT	floating-point(浮動小数点) spin box
XRTTBL_TYPE_FIELD_SPIN_INT	integer (整数)spin box
XRTTBL_TYPE_FIELD_SPIN_STRING	string(文字列) spin box

該当する型の XRT/field は、必要時に自動的に作成されます。フィールドの各型が一つだけ作成され、ユーザのトラバースによりそれが再利用されます。

4 XRT/field ウィジェットのリソースをセットするには、XmNxrtTblFieldArgListContext リソースを使用します。この値は、スタティック(静的)な Args のリストで、このリストの最後のエレメントは NULL になります。

```
static Arg args[3];

XtSetArg(args[0], XmNxrtFldMask, "###-####");
XtSetArg(args[1], XmNxrtFldAllowNull, False);
XtSetArg(args[2], NULL, NULL);
XtVaSetValues(table,
  XmNxrtTblContext, XrtTblSetContext(5, 5),
  XmNxrtTblFieldArgListContext, args,
  NULL);
```

編集が完了した後は、編集された値は、XrtTblCellValues 構造体に記憶されます。

table_entry デモ(*$XRTHOME/src/table/demos/table_entry*)は、XRT/field を XRT/table と一緒に使用する例ですので、参照してください。

注意: `XmNxrtTblFieldArgList` は内部的にコピーされませんので、これのメモリ管理はユーザの役割です。

リソース･ファイルのフィールド

作成された XRT/field はその型を反映しています。

ウィジェットの型	field名	子field名	子arrow名	子arrow名
XRTTBL_TYPE_FIELD_CURRENCY	“currency_field”			
XRTTBL_TYPE_FIELD_DATE	“date_field”			
XRTTBL_TYPE_FIELD_FLOAT	“float_field”			
XRTTBL_TYPE_FIELD_INT	“int_field”			
XRTTBL_TYPE_FIELD_STRING	“string_field”			
XRTTBL_TYPE_FIELD_COMBO_CURRENCY	“combo_currency_field”	“field”	“arrow”	
XRTTBL_TYPE_FIELD_COMBO_DATE	“combo_date_field”	“field”	“arrow”	
XRTTBL_TYPE_FIELD_COMBO_FLOAT	“combo_float_field”	“field”	“arrow”	
XRTTBL_TYPE_FIELD_COMBO_INT	“combo_int_field”	“field”	“arrow”	
XRTTBL_TYPE_FIELD_COMBO_STRING	“combo_string_field”	“field”	“arrow”	
XRTTBL_TYPE_FIELD_SPIN_CURRENCY	“spin_currency_field”	“field”	“decr_arrow”	“incr_arrow”
XRTTBL_TYPE_FIELD_SPIN_DATE	“spin_date_field”	“field”	“decr_arrow”	“incr_arrow”
XRTTBL_TYPE_FIELD_SPIN_FLOAT	“spin_float_field”	“field”	“decr_arrow”	“incr_arrow”
XRTTBL_TYPE_FIELD_SPIN_INT	“spin_int_field”	“field”	“decr_arrow”	“incr_arrow”
XRTTBL_TYPE_FIELD_SPIN_STRING	“spin_string_field”	“field”	“decr_arrow”	“incr_arrow”

このフィールドのリソースは、リソース･ファイルでセットできます。

フィールドの取得

特定な型の XRT/field ウィジェットを取得するには、*XrtTblGetXrtFieldByType()*を呼び出します。

```
field = XrtTblGetXrtFieldByType(table,
  XRTTBL_TYPE_FIELD_CURRENCY):
```

もし、その型のフィールドが存在しない場合は、それが(unmanaged で)作成されます。

フィールドを一度取得したら、セル値のフォーマットやアップデートに使用することができます。例えば、次のコードは、整数値を 16 進数フォーマットに変換しセルに入れています。

```
field = XrtTblGetXrtFieldByType(table,
  XRTTBL_TYPE_FIELD_INT);
XtVaSetValues(field,
  XmNxrtFldDisplayFormat, XRTFLD_FORMAT_HEX,
  XmNxrtFldInternalValue, 53,
  NULL);
value = XrtFldGetDisplayValue(field);
XtVaSetValues(table,
  XmNxrtTblContext,XrtTblSetContext(4, 3),
  XmNxrtTblCellValueContext,value,
  NULL);
XtFree(value);
```

XRT/field の *XrtFldGetDisplayValue()*は、フィールドが表示するストリングを取得します。このストリングは、`XmNxrtTblCellValueContext` リソースをセットすることにより、セルに記憶されます。

但し、次の XRT/field リソースは、リソースファイルや *XtSetValues()*の呼び出しではセットできません。

```
XmNxrtFldMask
XmNxrtFldPickList
XmNxrtFldMatchPickList
XmNxrtFldPickListIndex
```

これらのリソースは、ユーザがセル間を移動する際にテーブルによって内部的にリセットされますので、`XmNxrtTblFieldArgListContext` を使用してセットしてください。新しいセルでこの設定を有効にするには、`XmNxrtFldArgListContext` からリロードしてください。但し、`XmNxrtFldPickList` にはこの方法は使用できません。テーブルは(`XmNxrtTblStringPickListContext` 等の)独自の pick list リソースを使用し、ユーザが新しいセルに移動した時に、その pick list リソースがフィールドに適用されます。セルに XRT/table の pick list が与えられていない場合は、`XmNxrtFldArgListContext` の pick list が使用されます。pick list がテーブルと `XmNxrtTblFieldArgListContext` の両方に存在する場合は、テーブルの pick list が優先されます。

pick list の指定には、`XmNxrtTblFieldArgListContext` を使用せずに、(`XmNxrtTbl...PickList` 等を使用して)できるだけテーブルに pick list を管理させるようにしてください。もし、`XmNxrtTblFieldArgListContext` を使用する場合、これらの pick list に必要なメモリの割り当て、管理、割り当て解除は、XRT/table では行われません。

XRT/table とその機能の詳細に関しては、XRT/table のマニュアルをご覧ください。

4.4 自動マッチングによる補完

`XmNxrtFldAutoComplete` は、pick list がセットされている時に自動マッチングを行うかどうかを指定します。例えば、`XmNxrtFldAutoComplete` が `True` に

セットされていると、pick list に *red*、*green*、*blue* が含まれている場合、'g'が入力されると、値 green にジャンプします。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で表示されますが、自動的に補完された部分は、その逆の色で表示されます。部分的に入力された値が pick list のメンバでない場合は、無効な値が XmNxrtFldDefaultValue に置き換えられます。

XmNxrtFldMhPickList が False にセットされている場合は、pick list に存在しない値は、そのまま受け入れられ、XmNxrtFldDefaultValue によって置き換えられません。

4.5 ドロップ･ダウン･リスト

通常、ドロップ･ダウン･メニューまたは XrtCalendar ポップアップは、表示される段階まで作成されませんが、XmNxrtFldCreatePopups によってポップアップを直ぐに作成するか、または最初に表示される時に作成するかをコントロールすることができます。これにより、例えば、フィールド･ウィジェットが作成される時にポップアップのプロパティを指定することができます。XmNxrtFldCreatePopups が True の場合は、ポップアップは直ちに作成されます。一度作成されたポップアップは、このリソースが False にセットされても破棄されません。

ドロップ･ダウン･メニューのウィジェット ID は XmNxrtFldMenuList で、XrtCalendar ウィジェットの ID は XmNxrtFldCalendar でアクセスすることができます。

4.6 XrtComboBox のプログラミング

ここでは、XrtComboBox ウィジェットの動作のカスタマイズについて解説します。

デフォルトの設定

XrtComboBox が作成されると、その子フィールドは、デフォルトで次のようにセットされるリソースを持ちます。

```
XmNcursorPositionVisible    False
XmNeditable                 False
XmNeditMode                 XmSINGLE_LINE_EDIT
XmNmarginHeight             2
```

これらのリソースは、リソース･ファイルからセットすることはできませんが、プログラムによってセットすることができます。

```
XtVaSetValues(combo,
  XmNeditable,                True,
  XmNcursorPositionVisible,   True,
NULL);
```

ユーザが arrow ボタンを押した時に作られるポップアップ･メニューは、デフォルトで次のようにセットされる XmList リソースを持ちます。

```
XmNautomaticSelection       False
XmNborderWidth              0
```

```
XmNscrollBarDisplayPolicy   XmAS_NEEDED
XmNselectionPolicy          XmBROWSE_SELECT
XmNselectedItems            NULL
XmNselectedItemCount        0
XmNspacing                  0
```

これらのリソースは、アプリケーションではセットできません。その他の XmList リソースは、リソース・ファイルからセットすることができます。下の例では、ポップアップ・メニューの背景色を変更しています。

```
combo.*.picklist.background: green
```

Arrow クラスの指定

XmNxrtComboArrowClass リソースを使用して、arrow ボタンのウィジェット・クラスを指定できます。デフォルトでは、このウィジェット・クラスは、XmArrowButtonWidgetClass です。XmNxrtComboArrowClass は、XrtComboBox が作成される時にのみセットできます。

フィールドに対する arrow ボタンの位置は、XmNstringDirection の値によって決定されます。

子ウィジェット ID の取得

XrtComboBox の子ウィジェット ID は取得することができます。フィールドのウィジェット ID を取得するには、XmNxrtComboField リソースの値を取り出してください。arrow ウィジェット ID の場合は、XmNxrtComboArrow リソースの値を取り出します。

XmNxrtComboField と XmNxrtComboArrow リソースはセットすることができません。

pick list の検索の決まり

pick list 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。XmNxrtFldSearchPolicy は、この検索の動作をコントロールし、XRTFLD_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 XRTFLD_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 30 は、XmNxrtFldSearchPolicy の動作の例です。

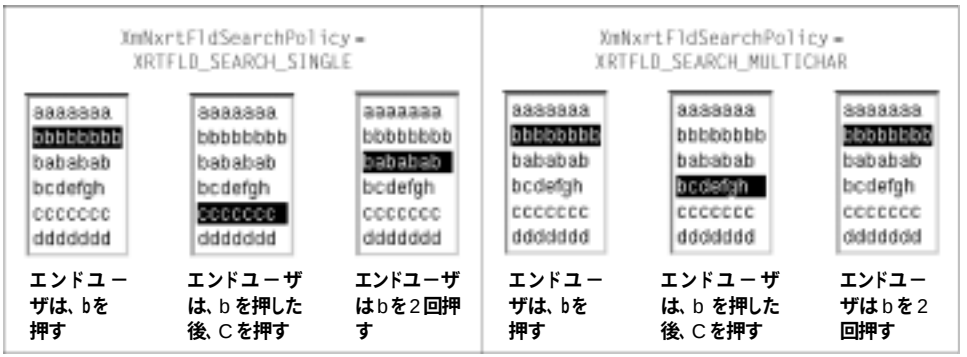


図30　ポップアップ･メニューの検索

pick list の変更

pick list のウィジェット ID を取得するには、XmNxrtFldMenuList 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。但し、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,
  XmNxrtFldMenuList, &menulist,
  NULL);
XtVaSetValues(menulist,
  XtVaTypedArg, XmNfontList, XmRString,
  fontlist, strlen(fontlist)+1,
  NULL);
```

XmNxrtFldSearchPolicy が XRTFLD_SEARCH_MULTICHAR にセットされると、図 31 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、Appendix A の XmNxrtFldSearchPolicy の解説を参照して下さい。)

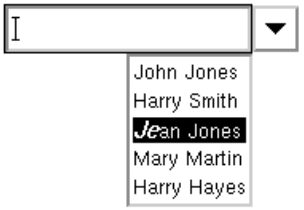


図 31　ハイライト・フォントを使用した pick list

尚、Combobox のユーザ・インタラクションに関する解説は、2.3 を参照してください。

Text Field の変更

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。

XmNautoShowCursorPosition	XmNblinkRate
XmNborderWidth	XmNcolumns
XmNcursorPosition	XmNcursorPositionVisible
XmNeditable	XmNeditMode
XmNfontList	XmNhighlightThickness
XmNmarginHeight	XmNmarginWidth
XmNmaxLength	XmNpendingDelete
XmNresizeWidth	XmNrows
XmNselectThreshold	XmNselectionArray
XmNselectionArrayCount	XmNsensitive
XmNsource	XmNshadowThickness
XmNtopCharacter	XmNvalue
XmNverifyBell	XmNwordWrap

他の text リソースをセットしたり、コールバックを追加するには、XmNxrtComboField の値を取得し、text フィールドのウィジェット 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。

メニューの配置

XmNxrtFldMenuAlignment は、Combo Box の text フィールドに対するポップアップ・メニューの位置を指定します。有効な値は図 32 の通りです。

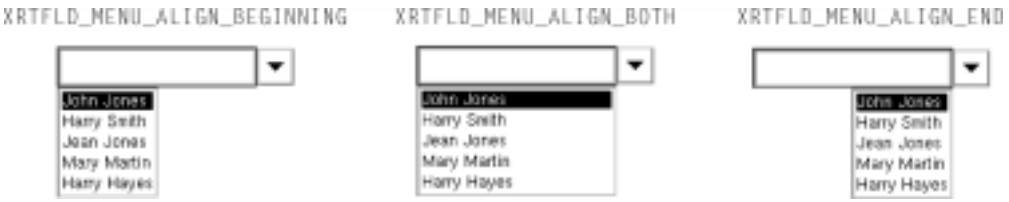


図 32　Combo Box のメニューの配置

デフォルトでは、XmNxrtFldMenuAlignment は XRTFLD_MENU_ALIGN_BOTH にセットされています。

子ウィジェットの間隔

デフォルトでは、text フィールドと arrow(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 33 のように XmNxrtComboSpacing をセットしてください。

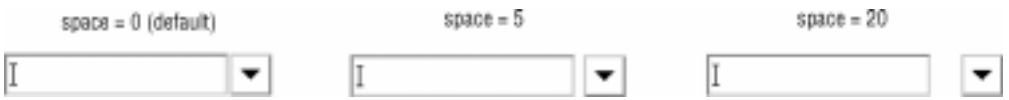


図 33　フィールドと arrow(矢印)の間隔

間隔は、pixel で指定されます。

Arrow の型の指定

表示される arrow(矢印)の型を指定するには、XmNxrtComboArrowType リソースをセットしてください。このリソースは、ウィジェットが作成される時にのみセットできます。有効な値は図 34 の通りです。



図 34　XrtComboBox の arrow の型

デフォルト値は、XRTFLD_ARROW_WEDGE です。

text フィールドに対する arrow ボタンの位置は、XmNstringDirection の値によって決定されます。

上記の型の他に、XmNxrtComboArrowType を XRTFLD_ARROW_CUSTOM にセットして、独自の型を指定することができます。

4.7 XrtSpinBox のプログラミング

ここでは、XrtSpinBox ウィジェットの動作のカスタマイズについて解説します。

デフォルトの設定

XrtSpinBox が作成されると、子フィールドは、次のようにデフォルトにセットされたリソースを持ちます。

```
XmNcursorPositionVisible    False
XmNeditable                 False
XmNeditMode                 XmSINGLE_LINE_EDIT
```

これらのリソースはリソース・ファイルからセットすることはできませんが、プログラミングによってセットすることができます。

子ウィジェットの追加

XrtSpinBox ウィジェットには、フィールドを始め、他のウィジェットを追加することができます。標準の Motif の *XtVaCreateManagedWidget()*手続きを使用し、XrtSpinBox を親ウィジェットとして指定してください。

spin box が、二つ以上のフィールド・ウィジェットを持っている場合は、arrow ボタンは、入力フォーカスがあるフィールドでアクティブになります。

Arrow の位置指定

`XmNxrtSpinArrowLayout` リソースを使用して、XrtSpinbox 内の arrow ボタンの位置を指定できます。このリソースは、次の値にセットできます。

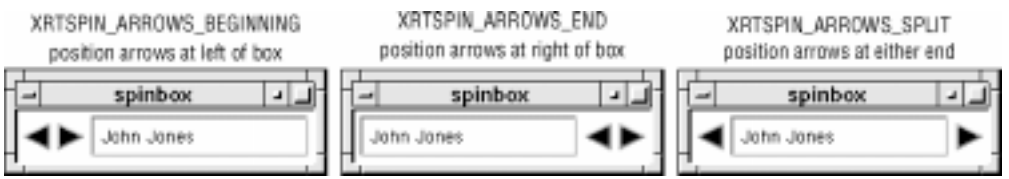


図 35 `XmNxrtSpinArrowLayout` の値

デフォルトの `XmNxrtSpinArrowLayout` は、`XRTSPIN_ARROWS_SPLIT` です。

親ウィジェットが `XmNstringDirection` リソースを定義し、`XmSTRING_DIRECTION_R_TO_L` にセットされていると、`XRTSPIN_ARROWS_BEGINNING` は、arrow をボックスの右に配置し、`XRTSPIN_ARROWS_END` は arrow をボックスの左に配置します。(`XmNxrtSpinArrowLayout` が `XRTSPIN_ARROWS_SPLIT` にセットされている場合は、`XmNstringDirection` は、arrow の位置をコントロールしません。)

Arrow の方向の指定

`XmNxrtSpinArrowOrientation` リソースは、XrtSpinBox 内の arrow の方向を指定します。このリソースは、次の値に設定できます。

`XmHORIZONTAL`	左と右の arrow を使用
`XmVERTICAL`	上と下の arrow を使用

XmNxrtSpinArrowLayout が XRTSPIN_ARROWS_BEGINNING または XRTSPIN_ARROWS_END にセットされ、XmNxrtSpinArrowOrientation が Xm_VERTICAL にセットされると、縦方向の arrow が上下に配置されて表示されます。

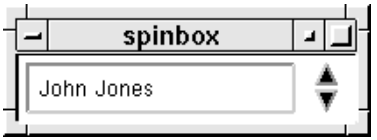


図 36　XrtSpinBox で上下に配置された arrow

Spin データ型の指定

XmNxrtSpinType リソースは、XrtSpinBox に表示されるデータ型を示します。このリソースは、次の値に設定できます。

XRTSPIN_TYPE_PICKLIST	XmNxrtFldPickList で指定されているアイテムを循環する
XRTSPIN_TYPE_COUNTER	XmNxrtFldMin と XmNxrtFldMax で定義されている範囲を(XmNxrtFldCounterIncrement で指定される)インクレメントに従ってスピンする

デフォルトの XmNxrtSpinType は、XRTSPIN_TYPE_PICKLIST です。

Spin Policy の指定

XmNxrtFldSpinPolicy は、XrtDateField が spin box 内にある時に使用されるカウンタ・スピンの型をコントロールします。このリソースを使用する場合は、XmNxrtFldSpinType を XRTSPIN_TYPE_COUNTER にセットしてください。XmNxrtFldSpinPolicy は、XRTFLD_SPIN_FIELD、XRTFLD_SPIN_SUBFIELD、XRTFLD_SPIN_WRAP の三つの値を持ちます。

Spin policy が XRTFLD_SPIN_FIELD の場合は、フィールドは XmNxrtFldCounterIncrement を使用します。XRTFLD_SPIN_WRAP は XRTFLD_SPIN_FIELD と似ていますが、スピンする値をラップ(最後の値まで到達すると最初の値に戻って表示)させることができます。XRTFLD_SPIN_SUBFIELD は、現在カーソルがある日付のサブ・フィールドをスピンさせます。サブ・フィールドは、XrtDate フィールドにのみ使用でき、その他の型では無視されます。また、サブ・フィールドをスピンする時は、XmNxrtFldEnterFormatList の最初のエントリだけが使用され、その他は全て無視されます。

Arrow クラスの指定

二つの子 arrow ウィジェット両方またはどちらかのウィジェット・クラスを指定することができます。インクレメントする arrow(上または右方向の矢印、但し XmNxrtSpinArrowLayout に依存）のウィジェット・クラスを指定するには、XrtSpinBox を作成する時に、XmNxrtSpinIncrArrowClass の値をセットします。デクレメントする arrow(下または左方向、但し、XmNxrtSpinArrowLayout に依存)のウィジェット・クラスを指定する場合は、XmNxrtSpinDecrArrowClass リソースをセットします。

XmNxrtSpinIncrArrowClass と XmNxrtSpinDecrArrowClass は、XrtSpinBox が作成される時にのみセットできます。デフォルトでは、両方の arrow のウィジェット・クラスは xmArrowButtonWidgetClass です。

子ウィジェット ID の取得

XrtSpinBox の定義済みの子ウィジェット ID は、次の方法によって取得することができます。

- 一番めのフィールドのウィジェット ID を取得するには、XmNxrtSpinField リソースの値を取り出す。
- インクレメント arrow のウィジェット ID を取得するには、XmNxrtSpinIncrArrow リソースの値を取り出す。
- デクレメント arrow のウィジェット ID を取得するには、XmNxrtSpinDecrArrow リソースの値を取り出す。

これらのリソースはセットできません。

子ウィジェットの間隔

デフォルトでは、フィールドと arrow の間にスペースはありません。子ウィジェットの間にスペースを置くには、XmNxrtSpinSpacing をセットしてください。この間隔は pixel で指定され、デフォルト値は 0(スペースはなし)です。

Arrow の型の指定

表示される arrow の型は、XmNxrtSpinArrowType でセットします。このリソースは、ウィジェットが作成される時にのみセットできます。有効な値は、XrtComboBox の XmNxrtComboArrowType と同じで、デフォルト値は XRTFLD_ARROW_WEDGE です。

4.8 XrtCalendar のプログラミング

XrtCalendar には、XrtSpinBox のほとんどの機能や設定を使用することができます。

年、月、時間のスピン等のカレンダのアクションでは、XmNxrtModifyCalendarCallback が呼び出されます。

Daygrid ヘッダー

ヘッダーに表示される曜日はロケールに依存し、このヘッダーのフォントや背景色、前景色はセットすることができます。また、XmNxrtCalendarHeaderNumChars によってヘッダーの数をセットすることができます。このリソースはデフォルトで 1、曜日を *SMTWTFS* で表し、2 にセットすると *SuMoTuWeThFrSa* で表示されます。このリソースを一番長い曜日名と同じかそれより長くセットしたり、MAX_INT にセットすると、曜日の名前が完全に表示されます。XmNxrtCalendarFullDayNames は、*full name* を曜日の実際の名前にするか、またはロケールにセットされている省略された名前にするかをコントロールします。

通常、週の最初の日は Sunday(Day 0)ですが、XmNxrtCalendarFirstDayOfWeek によってこれを変更することができます。

カレンダの表示

カレンダ・ウィジェットは、XmNfontList と XmNxrtCalendarHeaderFontList の値に基づいて、全てを表示できるようにしますが、それができない場合は、表示領域に中央揃えされます。カレンダ・ウィジェットにはスクロールバーは付きません。

デフォルトの色は XmNforeground と XmNbackground によってセットされますが、ヘッダーや、選択された日、指定された日の前景色や背景色をセットすることができます。また、カレンダの周囲(top、bottom、left、right)のマージンや、daygrid 上の各セルの周囲のマージンもセットすることができます。

カレンダのセルのハイライトの太さは XmNxrtCalendarCellHighlightThickness でセットします。このハイライトは、daygrid がフォーカスを得た時にカレント・セルの周囲に描画されます。

Special Days

休日や祭日、締切日等の特別な日を、*special days* と指定して daygrid 上で目立たせることができます。XmNxrtCalendarSpecialDays を special day のリストを指すようにセットしてください。リストの各エントリには special day または special day のリストを指定します。

このリソースが XRTCALENDAR_SPECIAL_DAY_BY_DATE にセットされている場合は、*day*、*month*、*year* フィールドが special day を決定します。XRTCALENDAR_SPECIAL_DAY_BY_DAY_OF_WEEK の場合は、*day_of_week*、*week*、*month*、*year* フィールドが special day を決定します。フィールドのどれかが XRTCALENDAR_ALL にセットされていれば、XrtCalendarSpecialDay はそのフィールドでは全てが有効と認識されます。例えば、ある日を毎年特別な日とするには、型を XRTCALENDAR_SPECIAL_DAY_BY_DATE にし、*day* と *month* フィールドをその special day に、*year* を XRTCALENDAR_ALL にセットします。

次ページのサンプル・コードは、*calendar.c* を使用し、Christmas Day と New Year's Day を special day としています。

```
#include <stdio.h>
#include <Xm/Xm.h>
#include <Xm/XrtField.h>

static char *fallbacks[] = {
"*.XmXrtCalendar.background: white",
"*.XmXrtCalendar.background: grey",
"*.XmXrtCalendar.foreground: black",
"*.XmXrtCalendar.xrtCalendarSpecialDayBackground: yellow",
"*.XmXrtCalendar.xrtCalendarSpecialDayForeground: red",
NULL
};

/* Set Christmas and New Years Day as special days */
static XrtCalendarSpecialDay days[] = {
  { XRTCALENDAR_SPECIAL_DAY_BY_DATE, 25, 0, 0, 11, XRTCALENDAR_ALL},
  { XRTCALENDAR_SPECIAL_DAY_BY_DATE, 1, 0, 0, 0, XRTCALENDAR_ALL},
  };
static XrtCalendarSpecialDay *spec_days[] = {
  &days[0],
  &days[1],
  NULL
  };

{
XtAppContext app_context;
Widget calendar, toplevel;

/* Initialize Intrinsics */
toplevel = XtVaAppInitialize(&app_context, "toplevel", NULL, 0,
&argc, argv, fallbacks, NULL);

calendar = XtVaCreateManagedWidget("calendar",
  xmXrtCalendarWidgetClass, toplevel,
  XmNxrtCalendarSpecialDays, spec_days,
  XmNxrtCalendarCreateTime, TRUE,
  NULL);

XtRealizeWidget(toplevel);
XtAppMainLoop(app_context);
}
```

図 37　special day を指定するコード

XmNxrtCalendarSpecialDayForeground と XmNxrtCalendarSpecialDayBackground は指定された日の前景色と背景色を変更して、daygrid 上で special day を目立つようにしています。これらのリソースの内、どちらかがセットされていない場合は、前景色の場合は XmNforeground、背景色の場合は XmNbackground が使用されます。

special day はリソース・ファイルでセットすることもできます。次のエントリは、Christmas Day と New Year's Day を special day としています。

```
*.XmXrtCalendar.xrtCalendarSpecialDays: (SPECIAL_DAY_BY_DATE 25 0 0 11
XRTCALENDAR_ALL) (SPECIAL_DAY_BY_DATE 1 0 0 0 XRTCALENDAR_ALL)
```

4.9 日付エディタのポップアップ

XrtDateField と XrtComboBox ウィジェットでは、XrtCalendar ウィジェットをポップアップとして使用することができます。カーソルがフィールド・ウィジェット上にある時に、Ctrl を押しながら下方向の arrow キーを押すか、または combo box の矢印キーをクリックすると、カレンダ・ウィジェットがポップアップします。XRTFLD_POPUP_PICKLIST は、XrtDateField を除く全てのフィールドのタイプのデフォルトで、XrtDateField のデフォルトは XRTFLD_POPUP_CALENDAR です。XmNxrtFldPopupType をセットしてこれを pick list に変更することができます。



図 38　combo box からポップアップしたカレンダ・ウィジェット

日付エディタの位置は XmNxrtFldMenuAlignment の設定に従います。但し、XRTFLD_MENU_ALIGN_BOTH の時、カレンダがフィールドより大きいと、カレンダの一部が欠けてしまいます。

XmNxrtFldCalendar リソースは、XrtCalendar のウィジェット ID を指します。従って、親の XrtCalendar ウィジェットがリソースをコントロールしますので、XmNwidth は変更できません。また、カレンダのリソース、XmNxrtCalendarCreateMonth、XmNxrtCalendarCreateTime、XmNxrtCalendarCreateYear をセットする時はリソース・ファイルを使用してください。下記の例では、combo box のカレンダ・ポップアップの背景色を hot pink にセットしています。

```
combo.*.datepopup.background:hotpink
```

calendar コールバックを使用すると、ポップアップが表示される時または、消される時に実行されるコードを追加することができます。このコールバックの機能は、menu コールバックと同様です。calendar コールバックを使用するには、XmNxrtFldCalendarCallback 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。

4.10 キーにアクションを割り当てる

XRT/field では、次のフィールドのアクションをキーに割り当てることができます。

- *Calendar()*、ポップアップする日付エディタを表示する
- *Cancel()*、ユーザによって現在タイプされている値を元に戻す
- *Complete()*、有効リストを使用して現在の値を補完する
- *Menu()*、有効なリスト・アイテムのポップアップ・メニューを表示する
- *NoAction()*、何もしない(トランスレーションを不可にする時に使用)
- *Overstrike()*、上書きモードをセットする
- *Scrollbars()*、スクロールバーを表示する
- *Spin()*、有効リストの次のアイテムを表示または、現在のアイテムをインクレメントする

次の例では、*Complete()*アクションを Return キーにバインドしています。

```
String translations = "Key<Return>: Complete()";

XtOverrideTranslations(field,
  XtParseTranslationTable(translations));
```

次のセクションからは、これらのアクションを解説していますが、アクションとキーのバインディングに関しては、Motif 関連の資料等を参照してください。

4.10.1 Calendar Action

*Calendar()*は、ポップアップ日付エディタの表示をコントロールします。このカレンダは、今日の日付をデフォルトとして開き、ユーザは XrtDateField に表示されている日を変更することができます。カレンダは、表示を指定する一つの引数を取ります。

`Display`	日付ポップアップを表示
`Hide`	日付ポップアップを隠す

Ctrl キーと下方向の矢印キーは、カレンダ・ポップアップの表示を切り替えます。

Calendar コールバック

XRT/field では、ポップアップ・カレンダが表示される時または消される直前に実行されるコードを提供することができます。コールバックは、`XmNxrtFldCalendarCallback` で指定されるリストに追加してください。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 {
  XrtFldReason    reason; /* read-only */
  Xevent          *event; /* read-only */
  struct tm       *date;
  Boolean         doit; /* initially True */
} XrtFldCalendarCallbackStruct;
```

reason は XRTFLD_REASON_CALENDAR_BEGIN（カレンダ・ポップアップが表示される前）または XRTFLD_REASON_CALENDAR_END(カレンダ・ポップアップが消された後)のどちらかです。*date* は、選択された日と時間を記憶する構造体のポインタです。ポップアップに時間の spinbox がない場合は、時間のフィールドはゼロでセットされます。*date* は、*reason* が XRTFLD_REASON_CALENDAR_END の時にのみ変更できます。

doit の動作は *reason* の値に依存します。*reason* が XRTFLD_REASON_CALENDAR_END で、*doit* が False にセットされれば、選択された日付は許可されません。

4.10.2 Cancel Action

*Cancel()*は、現在タイプされている値を削除し、そのフィールドに最後に入力された値を持つ XmNxrtFldInternalValue リソースの値と置き換えます。

*Cancel()*を使用して、アプリケーションに undo の機能を追加することができます。

4.10.3 Complete Action

*Complete()*アクションが指定されると、XRT/field は、XmNxrtFldPickList リソースで指定されている pick list の各エレメントを調べます。フィールドに現在記憶されている値が、pick list のエレメントの一部に適合すると、その適合したエレメントがフィールドの値の補完に使用されます。

例えば、pick list が次の値で構成され、フィールドの値が Apr とします。

```
January
April
July
October
```

この場合、*Complete()*が呼び出され、April がフィールドの値になります。

pick list が指定されていて、フィールドに記憶されている値がリストのどのエレメントにもマッチしない場合は、ビープ音が鳴り、フィールドの値はそのまま残ります。

補完コールバック

*Complete()*アクションが指定されると、XmNxrtFldCompleteCallback リソースで指定されているコールバックが開始されます。これらのコールバックに渡される構造体は、3.7 に解説されています。

*Complete()*は、XmNxrtFldPickList が定義されていても補完コールバックを呼び出しますので、あらゆる方法で値の補完を行うことができます。

4.10.4 Menu Action

*Menu()*は、XmNxrtFldPickList リソースによって指定された pick list を表示するポップアップ・メニューをコントロールします。フィールドの値が pick list のエレメントに適合する場合は、そのエレメントが選択され、そこまでスクロールされます。

*Menu()*は、表示アクションを指定する一つの引数を取ります。

Display	pick list を含むポップアップ・メニューが表示される。リストの各エレメントは、フィールドの編集フォーマットを使用して表示される。
Hide	ポップアップ・メニューは、非表示になる。

以下は、ポップアップ・メニューを使用した簡単な例です。

```
String picklist[] = {"Monday", "Tuesday", "Wednesday",
   NULL};

int
main(int argc, char *argv[])
{
  Widget toplevel, field;

toplevel = XtInitialize(argv[0], "Example", NULL, 0,
  &argc, argv);
field = XtVaCreateManagedWidget("field",
   xmXrtStringFieldWidgetClass, toplevel,
   XmNxrtFldPickList, picklist,
   NULL);
XtRealizeWidget(toplevel);
XtMainLoop();
}
```

図 39 は、Control と Down-arrow キーが押された時のポップアップ・メニューです。

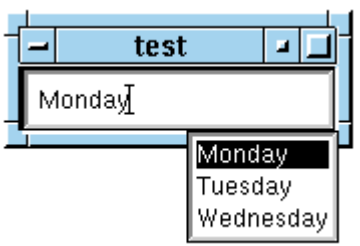


図 39　*Menu()*アクションによって表示されるポップアップ・メニュー

ポップアップ・メニューが作成されると、表示されるアイテムの数が XmNxrtFldVisibleItemCount リソースによって指定されます。必要時には、スクロールバーが表示されます。

メニュー・コールバック

XRT/field では、ポップアップ・メニューが表示された時と消えた時に実行されるコードを提供することができます。XmNxrtFldMenuCallback 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。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 {
  XrtFldReason    reason;     /* read-only */
  XEvent          *event;     /* read-only */
  int             position;   /* read-only */
  String          value;
  Boolean         doit;       /* initially True */
} XrtFldMenuCallbackStruct;
```

reason は XRTFLD_REASON_MENU_BEGIN(メニューが表示される前)か、XRTFLD_REASON_MENU_END(メニューが消えた後)のどちらかです。*position* は、pick list から選択されたアイテムのインデックス(1 から開始)です。ユーザがアイテムを選択しなかった場合、*position* は XRTFLD_NOVALUE にセットされます。*value* は、フィールドに表示される値です。*value* は、*reason* が XRTFLD_REASON_MENU_END の場合にのみ変更できます。*doit* の動作は、*reason* の値に依存します。*reason* が XRTFLD_REASON_MENU_BEGIN で、*doit* が False にセットされると、メニューは表示されません。(この場合に、アプリケーションは、任意のオペレーションを実行できます。)*reason* が XRTFLD_REASON_MENU_END で、*doit* が False であれば、選択された値は許可されません。

pick list とマスク

XmNxrtFldMask リソースがフィールドにマスクを定義すると、*Menu()*によって表示されるメニューのエレメントは、マスクによって指定されているフォーマットを使用します。

4.10.5 Overstrike アクション

*Overstrike()*アクションは、XmNxrtFldOverstrike リソースの動作をコントロールします。結果は、引数によって異なります。

- *Overstrike()*に True が渡されると、XmNxrtFldOverstrike は True にセットされる。
- *Overstrike()*に False が渡されると、XmNxrtFldOverstrike は False にセットされる。
- *Overstrike()*に Toggle が渡されると、XmNxrtFldOverstrike は現在の値の反対にセットされる。

XmNxrtFldMask がセットされると、XmNxrtFldOverstrike の値は無視されます。フィールドのマスクと検証は 3.4.4 に解説されていますので参照してください。

4.10.6 Scrollbars アクション

Scrollbars() アクションは、XmNxrtFldDisplayHorizScrollBar と XmNxrtFldDisplayVertScrollBar リソースの値を変更して、スクロールバーの表示をオンとオフにします。

Scrollbars() の一回目の呼び出しは、XmNxrtFldDisplayHorizScrollBar と XmNxrtFldDisplayVertScrollBar を XRTFLD_DISPSB_NEVER にセットして、スクロールバーの表示をオフにします。*Scrollbars()* の二回目の呼び出しは、XmNxrtFldDisplayHorizScrollBar を XRTFLD_DISPSB_ALWAYS に、XmNxrtFldDisplayVertScrollBar を XRTFLD_DISPSB_AS_NEEDED にセットして、スクロールバーの表示をオンにします。

それ以降の *Scrollbars()*の呼び出しも、同様にスクロールバーの表示をオンとオフにします。

4.10.7 Spin アクション

*Spin()*アクション・ルーチンは、XrtSpinBox ウィジェットに使用します。引数に従って、次または前の値を表示します。

- *Spin()*に `Next` が渡されると、次に表示可能な値がフィールドに記憶される。
- *Spin()*に `Prev` が渡されると、前に表示可能な値がフィールドに記憶される。

XrtSpinBox でユーザが右 arrow(矢印)ボタンを押すと、*Spin()*は引数 `Next` と一緒に呼び出されます。左 arrow ボタンが押されると、*Spin()*は引数 `Prev` と一緒に呼び出されます。

*Spin()*の動作は、`XmNxrtSpinType` リソースに依存します。

- `XmNxrtSpinType` が `XRTSPIN_TYPE_PICKLIST` にセットされると、*Spin()*は、`XmNxrtFldPickList` リソースによって指定された値をサイクルする。*Spin()*が最初に呼び出された時は、`XmNxrtFldPickList` によって指定されたリストの一番目の値が子フィールドに記憶される。次の Spin 操作は、pick list から次のエレメントを選択し、フィールドに記憶する。必要時には、wraparound します。(リストの最初のエレメントが最後のエレメントの次に来る、または、最後のエレメントが最初のエレメントの前に来るというように、サイクルします。)
- `XmNxrtSpinType` が `XRTSPIN_TYPE_COUNTER` にセットされると、*Spin()*は、`XmNxrtFldMin` と `XmNxrtFldMax` リソースで指定された値の範囲をサイクルする。*Spin()*が呼び出される度に、`XmNxrtFldCounterIncrement` の値が、フィールドに記憶されている値を増加/減少する。

デフォルトでは、`XmNxrtSpinType` は `XRTSPIN_TYPE_PICKLIST` にセットされています。

Spin コールバック

XRT/field では、*Spin()*アクションが実行される時に実行されるコードを提供することができます。`XmNxrtFldSpinCallback` 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。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 {
  XrtFldReason    reason;     /* read-only */
  XEvent          *event;     /* read-only */
  XrtFldDirection direction;  /* read-only */
  Boolean         at_boundary;    /* read-only */
  Boolean         doit;       /* initially True */
} XrtFldSpinCallbackStruct;
```

reason は、`XRTFLD_REASON_SPIN_BEGIN` または `XRTFLD_REASON_SPIN_END` です。*direction* は、`XRTFLD_DIRECTION_PREVIOUS` または `XRTFLD_DIRECTION_NEXT` です。`XmNxrtFldSpinType` リソースが `XRTFLD_SPINTYPE_PICKLIST` にセットされ、現在の値が pick list の最初または最後の値であれば、*at_boundary* は `True` です。*reason* が `XRTFLD_REASON_SPIN_BEGIN` の時に、値をサイクルさせないようにするには、*doit* を `False` にセットしてください。

pick list のインデックス

XmNxrtPickListIndex リソースは、pick list で現在選択されているエレメントの位置を示します。

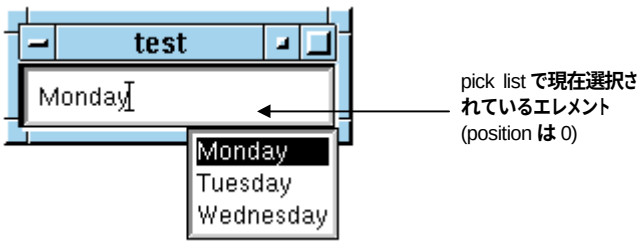


図 40　XmNxrtPickListIndex の値

リストの最初のエレメントが現在のエレメントであれば、XmNxrtPickListIndex は 0、二番目のエレメントが現在のエレメントであれば、XmNxrtPickListIndex の値は 1 で、それ以降は同様に続きます。

XmNxrtPickListIndex をセットして、異なったエレメントを選択することができます。このリソースにエレメントの数より大きい値をセットすると、リストの最後のエレメントが選択されます。

pick list のエレメントが何も表示されていない時は、XmNxrtPickListIndex の値は XRTFLD_NOVALUE です。

4.11　複数のフィールドのチェック

一つのフィールドの値が有効であるかは、他のフィールドがセットされるまで判断できないことがあります。(複数のフィールドが、ユーザによって入力されるフォームの一部である場合等)

複数のフィールドの検証を行うには、次のようにしてください。

- 必要なフィールドを manager ウィジェットの子孫として作成する
- ユーザがデータの入力を終えたら、*XrtFldCheckFields()*を呼び出して、全ての子ウィジェットを一度に調べる。

*XrtFldCheckFields()*は、次のように定義されます。

```
Boolean
XrtFldCheckFields(
  Widget          parent,
  XrtFldFieldData *data
)
```

parent は manager ウィジェットです。*data* は、manager ウィジェットの子孫であるフィールドの返されたリストです。このリストは、XrtFldFieldData 構造体のポインタとして返されます。この構造体は、次のように定義されます。

```
typedef struct {
  Widget *field_list;
  int num_field_list;
  Widget *invalid_list;
  int num_invalid_list;
  Widget *default_list;
  int num_default_list;
} XrtFldFieldData;
```

field_list は、manager ウィジェットの子孫であるフィールドのリストです。*num_field_list* は、これらのウィジェットが幾つあるか示します。

field_list にあるウィジェットで無効な値を持つものは、別のリスト、*invalid_list* に記憶されます。*num_invalid_list* は、無効な値を持つウィジェットの数を示します。同様にして、*field_list* にあるウィジェットでデフォルト値を持つものは、*default_list* に含まれ、このリストのウィジェットの数は *num_default_list* に記憶されます。(デフォルト値の解説は、4.1 にあります。)

*XrtFldCheckFields()*によってリストが返された後は、各フィールドまたは、(フィールドが相互に依存している場合は)それらの組み合わせを自由に調べることができます。例えば、次の push button コールバック(`XmNactivateCallback`)は、ユーザによって入力されていないフィールドがあった場合は、エラー・ボックスを表示します。

```
Widget form, errorbox;

void
pushCB(Widget widget, XtPointer client_data,
  XmPushButtonCallbackStruct *cs)
{
  XrtFldFieldData data;

  XrtFldCheckFields(form, &data);
  if (data.num_default_list > 0) {
    XtManageChild(errorbox);
  } else {
  /*
  * here is where the form processing goes
  */
  }
  XrtFldFieldDataFree(&data);
}
```

コールバックが、*XrtFldCheckFields()*によって返されたウィジェットのリストの使用を終了したら、*XrtFldFieldDataFree()*を呼び出して、リストに使用されていたメモリを解放してください。

4.12 日付の変換

date フィールドの値は、*struct tm* 型(ANSI の時間構造体)でなければなりません。*XrtFldCvtStringToTime()*を使用すると、簡単に日付を *struct tm* 構造体に変換することができます。

例えば、date フィールドの値を現在の値にセットするには、次のようにして、ストリング"NOW"を *XrtFldCvtStringToTime()*に渡します。

```
struct tm currtime;

XrtFldCvtStringToTime("NOW", NULL, &currtime);
  XtVaSetValues(field,
  XmNxrtFldInternalValue, &currtime,
  NULL);
```

*XrtFldCvtStringToTime()*を使用して、指定した時間を *struct tm* に変換することができます。

```
struct tm currtime;

XrtFldCvtStringToTime("Nov 11 1918 11:00",
  "MMM DD YYYY hh:mm", &currtime);
XtVaSetValues(field,
  XmNxrtFldInternalValue, &currtime,
  NULL);
```

上の例は、date フィールドを作成し、日時を November 11, 1918, 11:00 am にセットしています。この場合、*XrtFldCvtStringToTime()*には、変換される日時と一緒にフォーマット指定子が渡されます。このフォーマット指定子は、XmNxrtFldEnterFormatList リソースに使用されるものと同じです。(フォーマット指定子の詳細は、3.3 を参照してください。)

tm struct を使用して XmNxrtFldCounterIncrement をセットする

*XrtFldCvtStringToTime()*は、日付フィールドをセットする場合以外に、例えば、スピンボックスとカウンタと一緒に使用することもできます。

```
/*
** Get the current time and set up the tm structure to
** increment the minutes by one
*/

XrtFldCvtStringToTime("NOW", "hh:mm:ss", &currtime);
XrtFldCvtStringToTime("00:01:00", "hh:mm:ss", &incrementtime);

spinbox = XtVaCreateWidget("spinbox",
          xmXrtSpinBoxWidgetClass, rc,
          XmNxrtSpinType, XRTSPIN_TYPE_COUNTER,
          XmNxrtSpinFieldType, XRTFLD_FIELDTYPE_DATE,
          XmNxrtFldEnterFormatList, formatList,
          XmNxrtFldInternalValue, &currtime,
          NULL);
XtVaGetValues(spinbox, XmNxrtSpinField, &field, NULL);

XtVaSetValues(field,
  XmNxrtFldCounterIncrement, &incrementtime,
  NULL);
```

ここでは、*XrtFldCvtStringToTime()*が二回呼ばれています。(現在時刻で *strunct tm* をセットする時と、インクレメントするために *struct tm* をセットする時) 現在時刻の構造体は、スピンボックスの内部値に渡されます。incrementtime *tm struct* で表される一分間のインクレメントが、カウンタのインクレメント･リソースに渡されます。

その他の変換手続き

XRT/field は、long の整数を *struct tm* に、またはその逆に変換する手続きを提供しています。*XrtFldCvtLongToTime()*は long の整数を *struct tm* に、*XrtFldCvtTimeToLong()*は、*struct tm* を long の整数に戻します。

これらの手続きは、*struct tm* に記憶されている日付が long の整数に合う場合にのみ使用できます。(おおよそ 1902 から 2033 の間)

日付変換の手続きについての詳細は、Appendix B を参照してください。

4.13 スクロールのコントロール

図 41 にあるように、フィールドでは、スクロールバー機能が組み込まれています。

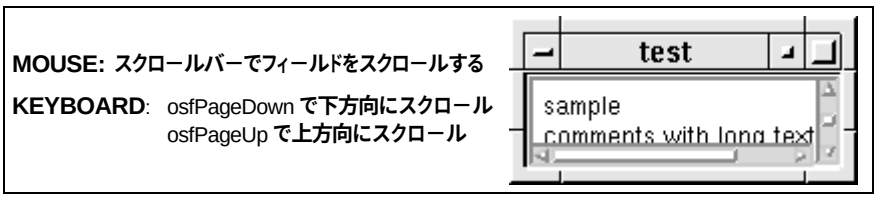


図 41　フィールドのスクロール

XmNxrtFldDisplayHorizScrollBar と XmNxrtFldDisplayVertScrollBar は、縦と横のスクロールバーをいつ表示するか指定します。これらのリソースは、次の値にセットしてください。

XRTFLD_DISPSB_ALWAYS	スクロールバーを常に表示する
XRTFLD_DISPSB_NEVER	スクロールバーは表示しない
XRTFLD_DISPSB_AS_NEEDED	スクロールバーは必要時に表示する

XmNxrtFldDisplayHorizScrollBar のデフォルト値は XRTFLD_DISPSB_AS_NEEDED、XmNxrtFldDisplayVertScrollBar のデフォルト値は XRTFLD_DISPSB_NEVER です。

これらのリソースは、*Scrollbars()*を使用してもコントロールできます。(4.10.6 参照)

スクロールバーのプログラミング

XRT/field で作成されるスクロールバーは XmScrollBar ウィジェットです。縦のスクロールバーは“VertScrollBar”、横のスクロールバーは“HorizScrollBar”です。縦と横のスクロールバーは、必要に応じて自動的に作成･破棄されます。

リソース・ファイルを使用することにより、XmScrollBar ウィジェットの属性を簡単に変更することができます。例えば、“Myfield”という名のウィジェットの横スクロールバーのインクレメントを設定するには、次のエントリをリソース・ファイルに追加してください。

```
*.Myfield.HorizScrollBar.increment:  3
```

4.14 マスク文字の定義

マスクの文字の一部または全てを、任意の文字と置き換えるには、XmNxrtFldMaskChars リソースをセットしてください。このリソースのデフォルト値は、“#@HULA*?\\”で、ストリングの各文字がマスクの文字と対応しています。

文字	マスクの値
#	数字、+と-の記号、カンマまたは小数点を表す特別文字
@	数字を表す特別文字
H	16 進数を表す特別文字
U	アルファベット大文字を表す特別文字
L	アルファベット小文字を表す特別文字
A	アルファベットを表す特別文字
*	一般的な文字を表す特別文字(どの文字も使用可能)
?	XmNxrtFldValidChars の文字を指定する特別文字
\\	エスケープ文字

独自の特別文字を指定するには、(リソース値の)ストリングの特別文字を指定する文字と置き換えてください。例えば、“!”を数字の特別文字として使用するには、デフォルト・ストリング“#@HULA*?\”の“@”と“!”を置き換えます。

```
XtVaSetValues(field,
  XmNxrtFldMaskChars, "#!HULA?\\",
  NULL);
```

XmNxrtFldMaskChars に割り当てられるストリングは“#!HULA?\\”で、“#!HULA?\”ではありません。C 及び C++では、バックススラッシュ文字“\”はエスケープ文字を表しますので、バックスラッシュをストリングに含ませるには、バックスラッシュが二つ必要です。

4.15 XrtField と XmText コールバック

XrtField は XmText ウィジェットのサブクラスですので、XRT/field は、XmText のコールバックを内部で使用します。多くの XmText コールバックは、XRT/field のコールバックを開始させます。

以下の XmText コールバックがフィールドに使用されます。

- XmNfocusCallback、フィールドがフォーカスを得る時に開始される
- XmNmodifyVerifyCallback、プログラミングまたはキーボードによるアクションが起こった時(但し、ウィジェットの値が変更される前)に開始される
- XmNmotionVerifyCallback、マスクの literal がスキップされた時に開始される (literal は、マスクの特別文字でない文字、変換されずに文字通りに解釈される)
- XmNvalueChangedCallback、ウィジェットの値が変更された後に開始される
- XmNactivateCallback、ユーザが Return キーを押した時に開始される
- XmNlosingFocusCallback、ユーザが Tab キーを使用して他のフィールドに移動した時や、他のウィンドウに移動した時に開始される

これらの XmText コールバックの一つが開始されると、登録されているフィールドのコールバックが開始されます。次の表は、XmText コールバックと関連するフィールドのコールバックの一覧です。

	XmNactivateCallback	XmNfocusCallback	XmNlosingFocusCallback	XmNmodifyVerifyCallback	XmNvalueChangedCallback
XmNxrtFldCompleteCallback	●		●	●	
XmNxrtFldConvertCallback	●		●	●	●
XmNxrtFldEnterCallback		●			
XmNxrtFldErrorCallback	●		●	●	
XmNxrtFldExitCallback	●		●		
XmNxrtFldValidateCallback	●		●	●	●

この表にある XmText コールバックは、フィールドのコールバックを開始します。但し、XmNxrtFldErrorCallback リストは、エラーが起こった場合にのみ開始されます。同様にして、XmNxrtFldExitCallback リストは、エラーが起きなかった場合にのみ開始されます。

コールバックの reason を取り出す

XmNxrtFldErrorCallback または XmNxrtFldExitCallback 手続きが開始された時、どの XmText コールバックがそれを開始させたか調べることができます。この機能により、例えば、ユーザが Return キーを押した時にのみ行われるプロセスを定義することができます。エラー・コールバックの詳細については 3.5、exit コールバックの詳細は 3.9 をそれぞれ参照してください。

XmText コールバックの指定

独自の XmText コールバックを指定した場合は、既に指定されたフィールドのコールバックの後にそれらが追加されます。例えば、ウィジェットに `XmNfocusCallback` を指定した時は、`XmNxrtFldEnterCallback` によって指定されているコールバックが開始された後に、このコールバックが開始されます。

通常は、フィールドのコールバックだけで充分な場合が多く、XmText コールバックをフィールドに指定する必要はありません。

4.16 XRT/field で C++を使用する

XRT/field は、C++インターフェースを提供しています。これを使用する場合は、ヘッダー・ファイル、*$XRTHOME/include/Xm/XrtField.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、次の方法で XrtField クラスを定義します。

- *XmCreateXrtField()*を呼び出す constructor。
- *XtDestroyWidget()*を呼び出す destructor。
- XRT/field ウィジェットを最初の引数として取る C の各手続きに対応する方法。例えば、*XrtFldGetField()*の方法は、*getField()*。
- 各設定可能なリソースをセットする方法。例えば、`XmNxrtFldEditFormat` をセットする方法は、*setEditFormat()*。
- 各リソースを取り出す方法。例えば、`XmNxrtFldEditFormat` を取り出す方法は、*getEditFormat()*。
- 標準の Xt の手続きを実行する方法。*addCallback()*、*destroy()*、*getDisplay()*、*getName()*、*getParent()*、*isManaged()*、*isRealized()*、*manageChild()*.

4.17 UIL の使用

XRT/field 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、OSF の Motif ウィジェットで使用されているデータ型のみサポートしているため、XRT/field のリソースの幾つかは UIL を使用してセットすることができませんが、`MrmNcreateCallback` を用いてアプリケーションでセットすることができます。Appendix H に UIL の引数とデータ型のリストがありますので、参照してください。

UIL でウィジェットを作成するには、次のようにしてください。

- アプリケーションに、UIL ヘッダーファイル、*XrtField.uil* をインクルードする
- *MrmInitialize()*のコールの直後に *XrtFldMrmInitialize()*を呼び出す

4.18 リソース値をストリングに変換

リソース変換プログラムは、リソース値のフォーマットを変換します。リソース・ファイルに記憶されている値は、ストリングからウィジェットが理解できるフォーマットに変換されます。(Boolean、integer 等) XRT/field は組み込まれているリソース変換プログラムを自動的に登録し、ストリングや XRT/field 特有のリソースを変換します。

XRT/field には、XRT/field 特有のリソースをストリングに変換する逆変換プログラムも提供されています。これは、リソース値をリソース・ファイルに書いたり、ウィジェットの状態を記憶する時に便利です。

変換プログラムを使用するには、*XrtFldSetReverseConverters()*を呼び出してください。この手続きは、逆変換プログラムを登録します。各逆変換プログラムは、変換の結果の記憶に必要なスペースを割り当てます。

次の例は、*XrtFldSetReverseConverters()*が呼ばれた後に、どのようにリソース・ストリングを取得するかを示しています。

```
Boolean
getStringValue(Widget w, String resource_name,
  String resource_type, String *string)
{
  XtArgVal value = 0;
  XrmValue from, to;

  XtVaGetValues(w, resource_name, &value, NULL);
  from.addr = (XtPointer) &value;
  from.size = sizeof(XtArgVal);
  to.addr = NULL;
  if (!XtConvertAndStore(w, resource_type, &from,
    XtRString, &to))
    return (False);
  *string = to.addr;
  return (True);
}
```

返されたストリングは、使用後必ず解放してください。

4.19 国際化

XRT/field は、標準の POSIX のロケールに準拠していますので、使用している言語やシンボルを XRT/field で使用することができます。

ロケールの設定

ロケールをセットするには、Xt の他の呼び出しの前に、必ず、*XtSetLanguageProc()*を呼び出してください。

```
int main(int arg, char** argv)
{
  ...
  XtSetLanguageProc(NULL, NULL, NULL);
  top=AtAppInitialize(...);
  ...
}
```

ロケール情報の印刷

XRT/field には、現在のロケールに関する情報をプリントするデモ・プログラム *printlocale*(*$XRTHOME/src/field/demos/locale*)が提供されています。現在のロケールに、

ロケール情報で定義されていないものがある場合は、無効のフラグが立てられます。

printlocale を使用して、必要なロケール情報が環境に含まれているか確認することができます。ロケール固有のデータを直接設定するには、*XrtFldSetDefaultLocaleStruct()*を使用してください。(この手続きの解説は、Appendix B にあります。)

関連資料

ロケールと国際化は、PTR Prentice Hall 出版の Sandra Martin O'Donnell 著、*Programming for the World* に詳しく解説されています。

Part II

Reference Appendices

A

リソースの一覧

ウィジェットの概要 ■ *XrtField リソース*
XmText リソース ■ *継承された XrtField リソース*
XrtComboBox リソース ■ *XrtSpinBox リソース*
継承された XrtComboBox と XrtSpinBox リソース
XrtCalendar リソース

XRT/fieldの全てのリソースのアルファベット順の一覧です。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。

A.1 ウィジェットの概要

各ウィジェットのインクルード・ファイルは、<*Xm/XrtField.h*>です。

XrtComboBox
クラス名: XmXrtComboBox
クラス階層: Core→Composite→Constraint→XmManager →XmXrtFldManager→XmXrtComboBox
クラス・ポインタ: xmXrtComboBoxWidgetClass

XrtCurrencyField
クラス名: XmXrtCurrencyField
クラス階層: Core→Primitive→XmText→XmXrtField →XmXrtFloatFied→XmXrtcurrencyField
クラス・ポインタ: xmXrtCurrencyFieldWidgetClass

XrtDateField

クラス名: XmXrtDateField
クラス階層: Core→Primitive→XmText→XmXrtField →XmXrtDateField
クラス・ポインタ: xmXrtDateFieldWidgetClass

XrtFloatField

クラス名: XmXrtFloatField
クラス階層: Core→Primitive→XmText→XmXrtField →XmXrtFloatField
クラス・ポインタ: xmXrtFloatFieldWidgetClass

XrtIntField

クラス名: XmXrtIntField
クラス階層: Core→Primitive→XmText→XmXrtField →XmXrtIntField
クラス・ポインタ: XmXrtIntFieldWidgetClass

XrtSpinBox

クラス名: XmXrtSpinBox
クラス階層: Core→Composite→Constraint→XmManager →XmXrtFldManager→XmXrtSpinBox
クラス・ポインタ: xmXrtSpinBoxWidgetClass

XrtStringField

クラス名: XmXrtStringField
クラス階層: Core→Primitive→XmText→XmXrtField →XmXrtStringField
クラス・ポインタ: XmXrtStringFieldWidgetClass

XrtCalendar

クラス名: XmXrtCalendar
クラス階層: Core→Composite→Constraint →XmManager →XmXrtFldManager→XmXrtCalendar
クラス・ポインタ: xmXrtCalendarWidgetClass

A.2 XrtField リソース

XrtField ウィジェット・クラスとそのサブクラスのリソースの一覧です。特定なフィールドのサブクラスに定義されるリソースには、リソース名の前に、サブクラスを表すシンボルが付いています。

C currency フィールド(XrtCurrencyField)

D date フィールド(XrtDateField)

F float フィールド(XrtFloatField)

I integer フィールド(XrtIntField)

S string フィールド(XrtStringField)

XmNxrtFldAllowNull Boolean True CSG

フィールドが空の時、有効であるか示します。`False` にセットされると、空のフィールドは無効と処理されます。string フィールドは、`NULL` または空のストリングを含む時は空です。他の型のフィールドは、`NULL` または白文字(whitespace character)を含む時は空とみなされます。

`XmNxrtFldDefaultValue` がセットされると、フィールドは空とはみなされません。

XmNxrtFldAutoComplete Boolean False CSG

pick list がセットされた時、自動マッチング機能を使用するかどうかを指定します。複数の文字による検索がサポートされています。キーが続いて押されると、pick list のカレント・アイテムは、キー入力と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。各キーの押下が検索を再定義します。

D XmNxrtFldCalendar Widget dynamic G

`XmNxrtFldPopupType` が `XRTFLD_POPUP_TYPE_CALENDAR` 型であると、このリソースは XrtCalendar 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。

D XmNxrtFldCalendarCallback XtCallbackList NULL CSG

*Calendar()*アクションが開始されるか、または XrtComboBox の矢印ボタンが押されると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次の構造体のポインタが渡されます。

```
typedef struct {
     XrtFldReason reason; /* read-only */
     Xevent       *event; /* read-only */
     struct tm    *date;
     Boolean      doit; /* initially True */
} XrtFldCalendarCallbackStruct;
```

reason は、`XRTFLD_REASON_CALENDAR_BEGIN`(カレンダ・ポップアップの表示前)または `XRTFLD_REASON_CALENDAR_END`(カレンダ・ポップが消された後)のどちらかです。*date* は、選択された日と時間を記憶する構造体のポインタです。ポップアップが時間の spinbox を持たない場合は、時間のフィールドはゼロでセットされます。*date* は、*reason* が `XRTFLD_REASON_CALENDAR_END` の時にのみ変更できます。*doit* の動作は、*reason* の値に依存します。`XRTFLD_REASON_CALENDAR_END` の時に *doit* が `False` にセットされると、選択されている日は許可されません。

XmNxrtFldCase　XrtFldCase　XRTFLD_CASE_AS_IS　CSG

フィールドのアルファベットの大文字/小文字を指定します。有効な値は次の通りです。

XRTFLD_CASE AS_IS	変換なし
XRTFLD_CASE_UPPER	大文字に変換される
XRTFLD_CASE_LOWER	小文字に変換される

XRT/field は、文字の変換に ANSI マクロの toupper と tolower を使用します。これらのマクロは、ロケールによって動作が異なる場合があります。

XmNxrtFldCompleteCallback　XtCallbackList　NULL　CSG

ユーザがフィールドを出た後で、値を完成させるコールバックのリストを指定します。XrtFldCompleteCallbackStruct 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
CallbackProcedure(
  Widget  field;
  XtPointer   client_data,
  XtPointer   call_data /* points to an
     XrtFldCompleteCallbackStruct */
  )
```

XrtFldCompleteCallbackStruct は次のように定義されています。

```
typedef struct {
  XrtFldReason    reason; /* read-only */
  XEvent      *event; /* read-only */
  String      value;
} XrtFldCompleteCallbackStruct;
```

reason は XRTFLD_REASON_COMPLETE です。*value* は、(XmNxrtFldEditFormat で指定される)編集フォーマットを使用したフィールド値です。補完はこの値を変更することにより実行されます。この値を解放し、新しい値を割り当ててセットすることによっても、値を変更することができます。

このリストにコールバックを追加するには、*XtAddCallback()*を使用してください。

XmNxrtFldConvertCallback XtCallbackList NULL CSG

検証される前に、フィールドに記憶されている値を変換するコールバックのリストを指定します。XrtFldConvertCallbackStruct 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
CallbackProcedure(
  Widget  field;
  XtPointer   client_data,
  XtPointer   call_data /* points to an
     XrtFldConvertCallbackStruct */
)
```

XrtFldConvertCallbackStruct は次のように定義されています。

```
typedef struct {
  XrtFldReason    reason; /* read-only */
  XEvent      *event; /* read-only */
  XrtFldAction    action; /* read-only */
  XmTextPosition  startPos, endPos; /* read-only */
  XmTextPosition  currInsert; /* read-only */
  XmTextPosition  newInsert;
  XmTextBlock     text;
  String      value;
} XrtFldConvertCallbackStruct;
```

reason は常に XRTFLD_REASON_CONVERT です。*action* は現在処理中のアクションで、XRTFLD_ACTION_KEYSTROKE(keystroke アクション)、XRTFLD_ACTION_PASTE(paste アクション)、XRTFLD_ACTION_EXIT(user exit アクション)、XRTFLD_ACTION_SETVALUE(programmatic change アクション)のどれかです。

text は変換されるテキストで、その内容は *action* の値に依存します。*action* が XRTFLD_ACTION_KEYSTROKE であれば、*text* はユーザによって最後に入力された文字です。*action* が XRTFLD_ACTION_PASTE であれば、*text* はペースト/ドラッグ/ドロップによって生成されたテキストです。XRTFLD_ACTION_EXIT または XRTFLD_ACTION_SETVALUE であれば、*text* はフィールドの内容全体を含みます。

startPos は、フィールドの開始点と *text* の最初の文字の間にある文字の数です。*endPos* は、フィールドの開始点と *text* の最後の文字の間にある文字の数です。*text* が一文字だけか、*action* が XRTFLD_ACTION_PASTE の場合は、*startPos* と *endPos* の値は同じです。

currInsert は、カーソルの現在の位置を指定し、*newInsert* は、新たに指定したいカーソルの位置です。(デフォルトでは、この値は *currInsert* と同じです。)

value は値全体を含み、フォーマットは *action* に依存します。ユーザの exit、ペースト、またはキーボードによるアクションがコールバックを開始した場合は、*value* は、(XmNxrtFldEditFormat で指定される)編集フォーマットを使用します。プログラミングによる変更のアクションがコールバックを開始した場合は、*value* は、(XmNxrtFldDisplayFormat で指定される)表示フォーマットを使用します。

このリストにコールバックを追加する場合は、*XtAddCallback()*を使用してください。

CXmNxrtFldCounterIncrement	double	1	CSG
DXmNxrtFldCounterIncrement	struct tm*	NULL	CSG
FXmNxrtFldCounterIncrement	double	1	CSG
IXmNxrtFldCounterIncrement	long	1	CSG

フィールドが、(XmNxrtFldSpinType が XRTSPIN_TYPE_COUNTER にセットされた)XrtSpinBox に含まれている時の、矢印ボタンの増減(インクレメントとデクリメント)をコントロールします。

date フィールドでは、XmNxrtFldCounterIncrement は NULL にセットされ、インクレメントは、現在表示されている時間の最小単位になります。(例: 日付が hh:mm のフォーマットで表示されている場合、インクレメントは 1 分です。)date フィールドでカウンタをセットするには、*struct tm* 構造体の該当するエレメントをセットし、他のエレメントは全て 0 にしてください。

double 型のリソースを設定する場合は、*XrtFldDoubleToArgVal()*を使用してください。

XmNxrtFldCreatePopups	Boolean	False	C

ポップアップを直ちに作成するか、それとも要求時に作成するかを XRT/field に指示します。要求時にポップアップが作成される場合は、必要時までメモリが使用されません。

CXmNxrtFldDecimalPlaces	int	XRTFLD_LOCALE	CSG
FXmNxrtFldDecimalPlaces	int	2	CSG

小数点以下何桁まで表示するか指定します。このリソースは、XmNxrtFldDisplayFormat が XRTFLD_FORMAT_GENERAL の時は無視されます。このリソースが XRTFLD_LOCALE にセットされている場合は、ロケールの値が使用されます。

小数点以下の桁数は、0 から 99 までの範囲で選択できます。99 より大きい値が指定されると、99 が使用されます。

XmNxrtFldDefaultBackground	Pixel	XRTFLD_UNCHANGED_COLOR	CSG

有効なデフォルト値(XmNxrtFldIsDefault は True)を表示するフィールドの背景色を指定します。XRTFLD_UNCHANGED_COLOR にセットされている場合、フィールドの背景色は、XmNbackground と同じです。モノクロのサーバでは、この色は黒に表示されます。

指定したデフォルト値が無効であると、XmNxrtFldErrorBackground の色が使用されます。

XmNxrtFldDefaultForeground	Pixel	XRTFLD_UNCHANGED_COLOR	CSG

有効なデフォルト値(XmNxrtFldIsDefault は True)を表示するフィールドの前景色を指定します。XRTFLD_UNCHANGED_COLOR にセットされている場合、フィールドの前景色は、XmNforeground と同じです。モノクロのサーバでは、この色は白に表示されます。

指定したデフォルト値が無効であると、XmNxrtFldErrorForeground の色が使用されます。

CXmNxrtFldDefaultValue	double	0	CSG
DXmNxrtFldDefaultValue	struct tm*	**現在時刻**	CSG
FXmNxrtFldDefaultValue	double	0	CSG
IXmNxrtFldDefaultValue	long	0	CSG
SXmNxrtFldDefaultValue	String	NULL	CSG

XmNvalue が、(プログラミングによって、またはユーザが全ての文字を削除して)空のフィールドにセットされている時、フィールドに記憶される値を指定します。この値は、プログラムによってセットされたかのように処理、検証されます。

以下のようなフィールドでは、デフォルト値は記憶されません。

- XmNxrtFldDefaultValue が NULL にセットされた date フィールド
- XmNxrtFldDefaultValue が XRTFLD_NOVALUE_FLOAT にセットされた float または currency フィールド
- XmNxrtFldDefaultValue が XRTFLD_NOVALUE にセットされた ingeter フィールド
- XmNxrtFldDefaultValue が NULL にセットされた string フィールド

これらの場合、値が削除された時は、フィールドの値は空の状態のままです。(XmNxrtFldAllowNull で、空のフィールドは無効というフラグを立てることができます。)

double 型のリソースを設定する場合は、*XrtFldDoubleToArgVal()*を使用してください。

CXmNxrtFldDisplayFormat	XrtFldFormat	XRTFLD_FORMAT_CURRENCY	CSG
FXmNxrtFldDisplayFormat	XrtFldFormat	XRTFLD_FORMAT_GENERAL	CSG
IXmNxrtFldDisplayFormat	XrtFldFormat	XRTFLD_FORMAT_GENERAL	CSG

ロケール値を使用して、マスクがないフィールドの表示フォーマットを指定します。このリソースが XRTFLD_FORMAT_DEFAULT にセットされている場合は、編集フォーマットが使用されます。編集フォーマットも表示フォーマットも指定されていない場合は、ユーザがタイプした通りに表示されます。

integer フィールドでは、次のフォーマットが使用できます。

XRTFLD_FORMAT_BINARY	バイナリ値(0b が先頭に追加される。 例: 0b10110)
XRTFLD_FORMAT_GENERAL	XmNxrtFldDisplayFormatString 値を使用
XRTFLD_FORMAT_HEX	16 進数値(0x が先頭に追加される。 例: 0x52ab3)
XRTFLD_FORMAT_OCTAL	8 進数値(0o が先頭に追加される。 例: 0o377)
XRTFLD_FORMAT_THOUSANDS	1、234

float フィールドでは、次のフォーマットが使用できます。

XRTFLD_FORMAT_FIXED	例: 12.34
XRTFLD_FORAMT_GENERAL	XmNxrtFldDisplayFormatString 値を使用
XRTFLD_FORMAT_PERCENT	12.34%
XRTFLD_FORMAT_SCIENTIFIC	1.23e+02
XRTFLD_FORMAT_THOUSANDS	1,234.00

currency フィールドでは、上の float のフォーマットと以下が使用できます。

XRTFLD_FORMAT_CURRENCY	$1,234.00
XRTFLD_FORMAT_CURRENCY_INTL	USDO 1,234.00

float と currency フィールドでは、XmNxrtFldDecimalPlaces リソースは、小数点以下の桁数を指定します。

CXmNxrtFldDisplayFormatString	String	NULL	CSG
DXmNxrtFldDisplayFormatString	String	NULL	CSG
FXmNxrtFldDisplayFormatString	String	NULL	CSG
IXmNxrtFldDisplayFormatString	String	NULL	CSG

date フィールドでは、このリソースは日付や時間を表示する際に使用されるフォーマットを指定します。date フォーマットは、次の組み合わせで構成されます。

D	日(1-31)
DD	二桁の日表記、一桁の場合は先頭に 0 を置く(例: 01、02)
W	省略された曜日
WW	曜日
J	Julian(ユリウス暦の)日(1-366)
JJ	二桁の Julian 日、一桁の場合は 0 を先頭に置く
M	月(1-12)
MM	二桁の月表記、一桁の場合は 0 を先頭に置く(例: 01、02)
MMM	省略された月名
MMMM	月名
Y、YY	今世紀での年(00-99)
YYYY	世紀を含む年
H	時間(0-12)
HH	二桁の時間(0-12)、一桁の場合は 0 を先頭に置く
h	時間(0-23)
hh	二桁の時間表記、一桁の場合は 0 を先頭に置く(例: 01、02)
m	分(0-59)
mm	二桁の分表記、一桁の場合は 0 を先頭に置く(例: 01、02)
s	秒(0-59)
ss	二桁の秒表記、一桁の場合は 0 を先頭に置く(例: 01、02)
A、P	AM/PM 表記
Z	タイムゾーンの略
\\	エスケープ文字(次の文字は、その文字通りに解釈する)

その他の文字は、その文字通りに表示されます。

このリソースが NULL であると、デフォルトの XmNxrtFldEnterFormatList の最初のストリングが XmNxrtFldDisplayFormatString として使用されます。(XmNxrtFldEnterFormatList は P103 に解説があります。)

float と integer フィールドでは、XmNxrtFldDisplayFormat が XRTFLD_FORMAT_GENERAL の時に表示される値のフォーマットを指定します。NULL でない場合は、*sprintf* フォーマット・ストリングとして使用されます。それ以外の場合は、"%g"が float フィールドに、"%d"が integer フィールドに使用されます。

XmNxrtFldDisplayHorizScrollBar　XrtFldDispScrollBar　XRTFLD_DISPSB_NEVER　CSG

いつ横スクロールバーを表示するか指定します。有効な値は以下の通りです。

XRTFLD_DISPSB_ALWAYS	スクロールバーは常に表示される
XRTFLD_DISPSB_NEVER	スクロールバーは表示されない
XRTFLD_DISPSB_AS_NEEDED	スクロールバーは必要に応じて表示される

XmNxrtFldDisplayVertScrollBar XrtFldDispScrollBar XRTFLD_DISPSB_AS_NEEDED CSG

いつ縦スクロールバーを表示するか指定します。有効な値は以下の通りです。

XRTFLD_DISPSB_ALWAYS	スクロールバーは常に表示される
XRTFLD_DISPSB_NEVER	スクロールバーは表示されない
XRTFLD_DISPSB_AS_NEEDED	スクロールバーは必要に応じて表示される

CXmNxrtFldEditFormat XrtFldFormat XRTFLD_FORMAT_FIXED CSG
FXmNxrtFldEditFormat XrtFldFormat XRTFLD_FORMAT_GENERAL CSG
IXmNxrtFldEditFormat XrtFldFormat XRTFLD_FORMAT_GENERAL CSG

編集中のフィールド値を表示する際に使用されるフォーマットを指定します。このリソースを XRTFLD_FORMAT_DEFUALT にしても、何の変化も起こりません。マスクがセットされている場合は、このリソースは無視されます。

integer フィールドでは、次のフォーマットが使用できます。

XRTFLD_FORMAT_BINARY	バイナリ値(0b が先頭に追加される。例: 0b10110)
XRTFLD_FORMAT_GENERAL	printf %d フォーマットを使用
XRTFLD_FORMAT_HEX	16 進数値(0x が先頭に追加される。例: 0x52ab3)
XRTFLD_FORMAT_OCTAL	8 進数値(0o が先頭に追加される。例: 0o377)
XRTFLD_FORMAT_THOUSANDS	1,234

float フィールドでは、次のフォーマットが使用できます。

XRTFLD_FORMAT_FIXED	例:12.34
XRTFLD_FORMAT_GENERAL	printf %g フォーマットを使用
XRTFLD_FORMAT_PERCENT	12.34%
XRTFLD_FORMAT_SCIENTIFIC	1.23e+02
XRTFLD_FORMAT_THOUSANDS	1、234.00

currency フィールドでは、上の float のフォーマットと以下が使用できます。

XRTFLD_FORMAT_CURRENCY	$1,234.00
XRTFLD_FORMAT_CURRENCY_INTL	USDO 1,234.00

ユーザがフィールドを出ると、値は、XmNxrtFldDisplayFormat の指定に従って表示されます。

XmNxrtFldEnterCallback　XtCallbackList　NULL　CSG

フィールドが入力フォーカスを得た時に開始されるコールバックのリストを指定します。XrtFldEnterCallbackStruct 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ています。

```
void
CallbackProcedure(
  Widget  field;
  XtPointer   client_data,
  XtPointer   call_data /* points to an
      XrtFldEnterCallbackStruct */
)
```

XrtFldEnterCallbackStruct は、次のように定義されています。

```
typedef struct {
  XrtFldReason    reason; /* read-only */
  XEvent      *event; /* read-only */
  String      value; /* read-only */
  XmString prompt_string; /* read-only */
} XrtFldEnterCallbackStruct;
```

reason は常に XRTFLD_REASON_ENTER です。*prompt_string* は、XmNxrtFldPromptString リソースの現在の値です。*value* は、ユーザがフィールドを編集した時に表示される値で、編集フォーマットに変換された XmNxrtFldInternalValue です。

このリストにコールバックを追加するには、*XtAddCallback()*を使用してください。

DXmNxrtFldEnterFormatList　String*　**下記参照**　CSG

有効な日付フォーマットの NULL 終端リストを指定します。このリストの最初の項目が、XmNxrtFldEditFormat として使用されます。表示された出力が、(XRT によって)理解されるように、XmNxrtFldDisplayFormatString がこのリストの最後に追加されます。

日付フォーマットは以下の組み合わせで構成されています。

D	日(1-31)
DD	二桁の日表記、一桁の場合は先頭に 0 を置く(例: 01、02)
W	省略された曜日
WW	曜日
M	月(1-12)
MM	二桁の月表記、一桁の場合は 0 を先頭に置く(例: 01、02)
MMM	省略された月名
MMMM	月名
Y、YY	今世紀での年(00-99)
YYYY	世紀を含む年
H	時間(0-12)
HH	0 から 12 の二桁の時間表記、一桁の場合は 0 を先頭に置く
h	時間(0-23)
hh	二桁の時間表記、一桁の場合は 0 を先頭に置く(例: 01、02)

m	分(0-59)
mm	二桁の分表記、一桁の場合は 0 を先頭に置く(例: 01、02)
s	秒(0-59)
ss	二桁の秒表記、一桁の場合は 0 を先頭に置く(例: 01、02)
A、P	AM/PM 表記
Z	タイムゾーンの略
\\	エスケープ文字(次の文字は、その文字通りに解釈する)

その他の文字は文字通りに解釈され、入力されたストリングを処理する際はスキップされます。

以下は、日付フォーマットのデフォルトのリストです。

YYYY-MM-DD hh:mm:ss
YYYY/MM/DD hh:mm:ss
DD/MM/YY hh:mm:ss
DD/MM/YYYY hh:mm:ss
DD-MM-YY hh:mm:ss
DD-MM-YYYY hh:mm:ss
MMM DD, YYYY hh:mm:ss
hh:mm:ss
hh:mm A
<ロケールの date/time フォーマット>
<ロケールの time フォーマット>

XmNxrtFldErrorBackground　Pixel　XRTFLD_ERROR_COLOR　CSG

`XmNxrtFldIsValid` が `Flase` であるフィールドの背景色を指定します。色が `XRTFLD_ERROR_COLOR` であれば、背景色は赤です。モノクロの端末では、常に黒に表示されます。このリソースが `XRTFLD_UNCHANGED_COLOR` にセットされている場合は、エラーが検知されても、ウィジェットの背景色は変更されません。

XmNxrtFldErrorCallback　XtCallbackList　NULL　CSG

検証中にエラーが検知された時、または無効なフィールドがフォーカスを失った時に開始されるコールバックのリストを指定します。`XrtFldErrorCallbackStruct` 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
CallbackProcedure(
  Widget  field;
  XtPointer   client_data,
  XtPointer   call_data /* points to an
     XrtFldErrorCallbackStruct */
)
```

`XrtFldErrorCallbackStruct` は次のように定義されています。

```
typedef struct {
  XrtFldReason    reason; /* read-only */
  XEvent      *event; /* read-only */
  int         text_reason;    /* read-only */
  XrtFldAction    action; /* read-only */
  XrtFldError     error;
  Boolean         is_modified;
  XmTextPosition  startPos, endPos; /* read-only */
  XmTextPosition  currInsert; /* read-only */
  XmTextPosition  newInsert;
  XmTextBlock     text;
  XmString        error_string;   /* read-only */
  Boolean         display;    /* initially False */
  Boolean         traverse;   /* initially True */
} XrtFldErrorCallbackStruct;
```

reason は XRTFLD_REASON_ERROR です。*action* は現在処理中のアクションで、XRTFLD_ACTION_KEYSTROKE(キーボードによるアクション)、XRTFLD_ACTION_EXIT(ユーザがフィールドを出るアクション)、XRTFLD_ACTION_SETVALUE(プログラミングによる変更のアクション)のどれかです。

text_reason は、どの XmText コールバックがこのコールバックを開始させたかを示し、次のどれかです。

XmCR_ACTIVATE	XmNactivateCallback によって開始
XmCR_LOSING_FOCUS	XmNlosingFocusCallback によって開始
XmCR_MODIFYING_TEXT_VALUE	XmNmodifyVerifyCallback によって開始
XmCR_VALUE_CHANGED	XmNvalueChangedCallback によって開始

error は、どの検証オペレーションが無効な値を検知したかを示します。有効な値は次の通りです。

XRTFLD_ERROR_EXCEED_LENGTH	値の行の長さが、XmNxrtFldMaxLineLength の値より大きい
XRTFLD_ERROR_EXCEED_LINES	値の行数が、XmNxrtFldMaxLines の値より大きい
XRTFLD_ERROR_IS_NULL	XmNxrtFldAllowNull が False にセットされている時にフィールドが空である
XRTFLD_ERROR_INVALID_CHAR	値に使用されている文字が、XmNxrtFldInvalidChars リストにあるか、または XmNxrtFldValidChars リストにない
XRTFLD_ERROR_INVALID_DATATYPE	値に使用されている文字が、フィールドのデータ型に適当ではない(例:integer フィールドに、アルファベット文字がある)

XRTFLD_ERROR_NOMATCH_MASK	値が、XmNxrtFldMask によって指定されているマスクに適合しない
XRTFLD_ERROR_NOMATCH_PICKLIST	値が、XmNxrtFldPickList で指定されている選択リストに含まれていない
XRTFLD_ERROR_OUT_OF_RANGE	値が、XmNxrtFldMax で指定されている上限より大きいか、XmNxrtFldMin で指定されている下限より小さい
XRTFLD_ERROR_VALIDATECB	値が、検証コールバックによって無効というフラグが立てられている(検証コールバックの *doit* 構造体メンバが False にセットされている)

XRTFLD_ERROR_INVALID_DATATYPE を除くエラーを置き換えるには、*error* を XRTFLD_ERROR_NONE にセットしてください。

is_modified は、XmNxrtFldIsModified の現在の値を持ちます。*is_modified* を変更すると、このリソースの値が変更されます。

currInsert、*newInsert*、*startPos*、*endPos* と *text* メンバは XrtFldConvertCallbackStruct 構造体の対応するメンバと同じです。

error_string は、エラーが起こった時にアプリケーションによって表示されるメッセージで、XmNxrtFldErrorString リソースの値です。メッセージは XmString 型(複合ストリング)です。

display メンバは、*action* が XRTFLD_ACTION_KEYSTROKE の場合にのみ使用されます。このメンバは、エラーである値をフィールドに表示するかどうか示します。*display* を True にすると、XRT/field はその値を表示します。デフォルトでは、*display* は False で、値は表示されません(XmNverifyBell が True であればビープ音が鳴ります)。

traverse メンバは、*action* が XRTFLD_ACTION_EXIT の場合に定義されます。このメンバによって、フィールドがエラーを含む時に、ユーザが Return または Tab キーによってフィールドの外へトラバースできるかどうかをコントロールすることができます。デフォルトでは *traverse* は True で、トラバースを実行できます。*traverse* を False にすると、Return または Tab キーが押されても、ユーザはそのフィールドに残ったままです。

このリストにコールバックを追加するには、*XtAddCallback()*を使用してください。

XmNxrtFldErrorForeground　Pixel　XRTFLD_ERROR_COLOR　CSG

XmNxrtFldIsValid が False であるフィールドの前景色を指定します。色が XRTFLD_ERROR_COLOR の場合は、前景色は白です。モノクロの端末では、この色は常に白です。このリソースが XRTFLD_UNCHANGED_COLOR にセットされると、ウィジェットの前景色は、エラーが検知されても変更されません。

XmNxrtFldErrorString　XmString　NULL　CSG

エラーが起きた時に表示されるメッセージを指定します。このリソースはアプリケーションのエラー・コールバックで使用され、XRT/field によっては使用されません。

XmNxrtFldExitCallback XtCallbackList NULL CSG

ユーザが有効なフィールドを出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。XrtFldExitCallbackStruct 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
CallbackProcedure(
  Widget  field;
  XtPointer   client_data,
  XtPointer   call_data /* points to an
     XrtFldExitCallbackStruct */
)
```

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

```
typedef struct {
  XrtFldReason    reason; /* read-only */
  XEvent      *event; /* read-only */
  int         text_reason;    /* read-only */
  Boolean         is_modified;
  String      value;      /* read-only */
  Boolean         traverse;   /* initially True */
} XrtFldExitCallbackStruct;
```

reason は XRTFLD_REASON_EXIT です。*event* はコールバックの原因となった X イベントです。*value* はフィールドの将来の値(表示される値)です。この値は、(XmNxrtFldDisplayFormat で指定される)表示フォーマットを使用して表示されます。

text_reason はどの XmText コールバックが exit コールバックを開始させたかを示します。有効な値は次の通りです。

XmCR_LOSING_FOCUS	XmNlosingFocusCallback によって開始
XmCR_ACTIVATE	XmNactivateCallbac によって開始

text_reason は、ユーザが Return キーを押す事によってフィールドを出ようした時に、XmCR_ACTIVATE にセットされます。その他の方法でフィールドを出る時は、*text_reason* は XmCR_LOSING_FOCUS にセットされます。

is_modified は、XmNxrtFldIsModified の現在の値を持ちます。*is_modified* を変更すると、このリソースの値が変更されます。

traverse は、ユーザが Return または Tab キーによってフィールドの外へトラバースできるかどうかコントロールします。デフォルトでは、*traverse* は True でトラバースを実行できます。

このリストにコールバックを追加するには、*XtAddCallback()*を使用してください。

CXmNxrtFldInternalValue	double	XRTFLD_MAXDOUBLE	CSG
DXmNxrtFldInternalValue	struct tm*	NULL	CSG
FXmNxrtFldInternalValue	double	XRTFLD_MAXDOUBLE	CSG
IXmNxrtFldInternalValue	long	XRTFLD_MAXLONG	CSG
SXmNxrtFldInternalValue	String	NULl	CSG

フィールドの現在の値(最後に入力された完成した値)を指定します。上記のデフォルト値は、XmNvalue が該当するデータ型に変換できない時に使用されます。フィールドが空の場合は、XmNxrtFldDefaultValue の値が使用されます。

このリソースをセットすることにより、プログラミングによる変更のアクションが起こります。(指定された値は検証され、必要に応じて変換されます。)

double 型のリソースを設定する時は、*XrtFldDoubleToArgVal()*を使用してください。

DXmNxrtFldInternalValueLong　long　XRTFLD_MAXLONG　CSG

date フィールドの現在の値を *long* として指定します。現在の値が *long* にできない(1902 から 2033 の範囲外である)時は、このリソースは `XRTFLD_MAXLONG` にセットされます。

XmNxrtFldInvalidChars　String　NULL　CSG

値の一部とならない文字ストリングを指定します。`XmNxrtFldValidChars` がセットされている時は、無効なストリングが最初に検索されます。

XmNxrtFldIsDefault　Boolean　dynamic　G

フィールドが現在空で(その結果デフォルト値を含む)かどうか決定します。

XmNxrtFldIsModified　Boolean　dynamic　CSG

値が最後にどのように修正されたかを示します。`True` であれば、ユーザはフィールドに記憶されている値を変更しています。`False` であれば、プログラムによって値が変更されたか、または全く変更されていません。

このリソースは、エラー・コールバックまたは exit コールバック構造体の *is_modified* フィールドを変更することによってセットできます。

XmNxrtFldIsValid　Boolean　dynamic　G

最後に行われた検証の結果を示します。`XmNxrtFldIsValid` が `True` であれば、最後のアクション(keystroke action[キーボード]、user exit aciton[ユーザがフィールドを出る]、programmatic change aciton[プログラムによる変更])が処理された時に、エラーが検知されなかったことを示します。

XmNxrtFldLastError　XrtFldError　dynamic　CSG

最後に起きたエラーを示します。フィールドが有効である場合は、このリソースは `XRTFLD_ERROR_NONE` にセットされます。

CXmNxrtFldLeadingZero　Boolean　True　CSG
FXmNxrtFldLeadingZero　Boolean　True　CSG

(-1, 1)の範囲にある数値の小数点の前に 0 をつけて表示するかどうか指定します。

XmNxrtFldMask　String　NULL　CSG

string と date フィールドでは、このリソースは、入力する文字とその場所を制限したり、編集可/不可の文字位置を示すマスクを指定します。このリソースは、他のサブクラスでは使用できません。マスクは、ユーザにどのような値を入力するかを示したり、入力または表示されるデータをフォーマットします。マスクの各文字は、placeholder(文字の入る場所)のような働きをします。

次のマスク文字が定義されています。

0-9、-と+、コンマ、小数点
@ 0-9
H16 進数(0-9、A-F)
Uアルファベット、大文字に変換される
L アルファベット、小文字に変換される
Aアルファベット、変換されない
* どの文字も可
? XmNxrtFldValidChars にある文字
\\ 次の文字は、文字通りに解釈される

その他の文字は文字通りに処理され、フィールドの値が空でも cue(入力の際のガイド)として表示されます。プロポーショナル・フォントが使用されている時は、ユーザの入力に従って literal(文字)の位置はシフトしていきます(但し、XmNvalue によって empty(空)の位置が前もってセットされていない場合)。

マスクがあるフィールドでは、overstrike(上書き)モードがデフォルトで設定されています。(XmNxrtFldOverstrike の値は無視されます。)ある文字を削除しても、他の文字は左にシフトされませんが、その前にある文字をスペースで置き換えます。

XmNvalue がマスクのあるフィールドにセットされると、データがマスクに全て適合しなくても、文字は左から右に読み込まれます。マスク内の literal(文字)はスキップされます。例えば、“(416)555-1212”という値は、マスクが“(###)###-####”というフィールドにセットすることができます。これにより、例えば、ユーザがデータベースから値をセットしたり、X コピー/ペーストを実行することができるようになります。

XmNxrtFldMask が指定されている場合は、データ型の検証は行われません。

XmNxrtFldMaskChars String “#@HULA*?\\” CSG

定義済みの XmNxrtFldMask を置き換える文字を指定します。文字は、必ず次の順序で指定されます。

数字、-と+、カンマ、小数点
@ 数字
H16 進数
U大文字のアルファベット
L 小文字のアルファベット
Aアルファベット
* 全ての文字
? XmNxrtFldValidChars で指定されている文字
\\ エスケープ文字

与えられた文字数が少ないと、残りはデフォルト値にセットされます。

XmNxrtFldMatchMask XrtFldMatchMask XRTFLD_MATCH_ALL CSG

値をマスク全体に適合させるかどうか指定します。有効な値は次の通りです。

XRTFLD_MATCH_ALL	マスク全体に適合
XRTFLD_MATCH_PARTIAL_L_TO_R	左から右へ部分的に適合
XRTFLD_MATCH_PARTIAL_R_TO_L	右から左へ部分的に適合

値の適合しない部分が全てスペースであれば、マスクは部分的に適合しているとみなされます。

XmNxrtFldMatchPickList Boolean True CSG

値を XmNxrtFldPickList のアイテムの一つに適合させるかどうか指定します。

CXmNxrtFldMax double dynamic CSG
DXmNxrtFldMax struct tm* NULL CSG
FXmNxrtFldMax double dynamic CSG
IXmNxrtFldMax long dynamic CSG
SXmNxrtFldMax String NULL CSG

フィールドの最大有効値を指定します。

float と integer フィールドでは、XmNxrtFldMax のデフォルト値は、XmNxrtFldRange で指定される範囲の上限です。

string フィールドでは、*strcoll()*手続きが、フィールドの値と XmNxrtFldMax で指定されている最大有効値を比較します。

double 型のリソースを設定する時は、*XrtFldDoubleToArgVal()*を使用してください。

XmNxrtFldMaxLineLength int XRTFLD_MAXINT CSG

1 行に入力できる最大文字数を指定します。XmNwordWrap が False であれば、この最大文字数以上が入力されると、エラー・コールバックが開始されます。

XmNxrtFldMaxLineLength は、XmText リソースの XmNmaxLength を置き換えます。

XmNxrtFldMaxLines int XRTFLD_MAXINT CSG

最大行数を指定します。このリソースは、XmNeditMode が XmMULTI_LINE_EDIT の場合にのみ意味があります。この最大行数以上が入力されるとエラー・コールバックが開始されます。

XmNxrtFldMenuAlignment XrtFldMenuAlignment XRTFLD_MENU_ALIGN_END CSG

ComboBox で、text 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。

XRTFLD_MENU_ALIGN_BEGINNING	メニューの左端と text フィールドの左端を揃える
XRTFLD_MENU_ALIGN_END	メニューの右端と text フィールドの右端を揃える
XRTFLD_MENU_ALIGN_BOTH	メニューと text フィールドの両端を揃える

XmNxrtFldMenuCallback XtCallbackList NULL CSG

ユーザが *Menu()*アクションを起こしたり、XrtComboBox の arrow(矢印)ボタンを押し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。`XrtFldMenuCallbackStruct` 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
CallbackProcedure(
  Widget  field;
  XtPointer   client_data,
  XtPointer   call_data /* points to an
     XrtFldMenuCallbackStruct */
  )
```

`XrtFldMenuCallbackStruct` は次のように定義されています。

```
typedef struct {
  XrtFldReason    reason; /* read-only */
  XEvent      *event; /* read-only */
  int         position;   /* read-only */
  String      value;
  Boolean         doit;       /* initially True */
} XrtFldMenuCallbackStruct;
```

reason は `XRTFLD_REASON_MENU_BEGIN`(メ ニ ュ ー 表 示 前) ま た は `XRTFLD_REASON_MENU_END`(メニュー表示後)です。*position* は、pick list 内のインデックス(1 から開始)です。ユーザがリストからアイテムを選択しなかった場合は、*position* は `XRTFLD_NOVALUE` にセットされます。*value* は、フィールドに表示される値です。この値は、*reason* が `XRTFLD_REASON_MENU_END` であれば変更することができます。

doit の動作は、*reason* の値に依存します。*reason* が `XRTFLD_REASON_MENU_BEGIN` で、*doit* が `False` にセットされていれば、メニューは表示されません。(この場合、アプリケーションが任意のオペレーションを実行できます。)*reason* が `XRTFLD_REASON_MENU_END` で、*doit* が `False` にセットされていれば、選択された値は許可されません。

このリストにコールバックを追加するには、*XtAddCallback()*を使用してください。

XmNxrtFldMenuList Widget dynamic C

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 クラスです。

DXmNxrtFldMilleniumThreshold int 0 CSG

2000 年に対して、どのように二桁の年を解釈するか指定します。有効な値は 0 から 100 です。0 の場合は、二桁の年の前に 1900 が付きます。その他の場合、このリソース値より小さい二桁の年の前には 2000 が付き、このリソース値より大きい二桁の年の前には 1900 が付けられます。(値が 70 の場合は、70 より小さい二桁の年は 2000 年代、70 と同じかそれより大きい年は 1900 年代と解釈されます。)

CXmNxrtFldMin	double	dynamic	CSG
DXmNxrtFldMin	struct tm*	NULL	CSG
FXmNxrtFldMin	double	dynamic	CSG
IXmNxrtFldMin	long	dynamic	CSG
SXmNxrtFldMin	String	NULL	CSG

フィールドの最小有効値を指定します。

float と integer フィールドでは、XmNxrtFldMin のデフォルト値は、XmNxrtFldRange で指定される範囲の下限です。

string フィールドでは、*strcoll()*手続きが、フィールドの値と XmNxrtFldMin で指定されている最小有効値を比較します。

double 型のリソースを設定する時は、*XrtFldDoubleToArgVal()*を使用してください。

CXmNxrtFldNegativeFormat	XrtFldNegFormat	XRTFLD_NEG_LOCALE	CSG
FXmNxrtFldNegativeFormat	XrtFldNegFormat	XRTFLD_NEG_AS_IS	CSG
IXmNxrtFldNegativeFormat	XrtFldNegFormat	XRTFLD_NEG_AS_IS	CSG

どのように負の値を表示するか指定します。以下のフォーマットが有効です。

XRTFLD_NEG_AS_IS	先頭にダッシュ(-)を付けて表示
XRTFLD_NEG_PARENTHESES	ダッシュを取り去り括弧で表示

currency フィールドでは、ロケール設定を使用する XRTFLD_NEG_LOCALE フォーマットも使用することができます。float フィールドまたは currency フィールドに記憶されている値が、パーセント(XRTFLD_FORMAT_PERCENT)で表示され、XRTFLD_NEG_PARENTHESES が負の表示フォーマットとして指定されていると、括弧はパーセントのシンボルを囲みます。

XmNxrtFldOverstrike	Boolean	dynamic	CSG

フィールドの overstrike(上書き)モードを指定します。この値は、マスクが指定されているフィールド(常に上書きモード)では無視されます。

CXmNxrtFldPickList	double*	NULL	CSG
DXmNxrtFldPickList	struct tm**	NULL	CSG
FXmNxrtFldPickList	double*	NULL	CSG
IXmNxrtFldPickList	int*	NULL	CSG
SXmNxrtFldPickList	String*	NULL	CSG

ユーザが選択できる値のリストを指定します。ウィジェットが string フィールドまたは date フィールドの場合は、値のリストは NULL で終了します。ウィジェットが integer または float フィールドの場合は、最初のエレメントの値は、リストのその他のエレメントの数です。

XmNvalue をアイテムの一つに適合させる場合は、XmNxrtFldMatchPickList を True にセットしてください。

XmNxrtFldPickListIndex	int	dynamic	CSG

XmNxrtFldPickList で指定される pick list 内の現在のアイテムの位置を指定します。リストの最初のアイテムの位置は 0 です。この値をアイテムの数より大きい数にセットすると、最後のアイテムが表示されます。このリストから何も表示されない場合は、値は XRTFLD_NOVALUE にセットされます。

XmNxrtFldPopupType　XrtFldPopupType　dynamic　CG

XRT/field ウィジェットで Ctrl と下方向の arrow が押されたり、または combo box で arrow が押された時のポップアップの種類を指定します。XrtDateField でのデフォルトは XRTFLD_POPUP_CALENDAR です。その他のフィールドでは、デフォルトは XRTFLD_POPUP_PICKLIST です。

このリソースは全ての XRT/field のサブクラスによって継承されますが、XrtDateField のみで使用してください。

XmNxrtFldPromptString　XmString　NULL　CSG

フィールドがフォーカスを得た時に表示されるメッセージを指定します。このリソースは、XmNxrtFldEnterCallback リストの手続きによって使用され、ウィジェットによっては使用されません。

CXmNxrtFldRange　XrtFldRange　XRTFLD_RANGE_DOUBLE　CSG
FXmNxrtFldRange　XrtFldRange　XRTFLD_RANGE_DOUBLE　CSG
IXmNxrtFldRange　XrtFldRange　XRTFLD_RANGE_INT　CSG

ウィジェットに記憶されるデフォルトの値の範囲を指定します(但し、XmNxrtFldMax または XmNxrtFldMin がセットされていない場合)。

float フィールドでは、次の範囲が指定できます。

XRTFLD_RANGE_FLOAT	単精度の浮動小数点数
XRTFLD_RANGE_DOUBLE	精度の浮動小数点数(デフォルト)

currency フィールドでは、記憶される値の範囲は、XmNxrtFldRange の値に関係なく、常に倍精度の浮動小数点数とみなされます。

integer フィールドでは、以下の範囲が指定できます。ここにあるリストは、32-bit マシンを想定しています。*/usr/include/limits.h* を調べて、御使用のマシンの正確な範囲を確認してください。

XRTFLD_RANGE_CHAR	signed character (範囲 -128 から 127)
XRTFLD_RANGE_SHORT	short signed integer (範囲 -32768 から 72767)
XRTFLD_RANGE_INT	standard-sized signed integer (デフォルト; 範囲 -2147483648 から 2147483647)
XRTFLD_RANGE_LONG	long signed integer (XRTFLD_RANGE_INT と同じ)
XRTFLD_RANGE_UNSIGNED_CHAR	unsigned character (範囲 0 から 255)
XRTFLD_RANGE_UNSIGNED_SHORT	short unsigned character (範囲 0 から 65535)
XRTFLD_RANGE_UNSIGNED_INT	standard-sized unsigned integer (範囲 0 から 4294967295)
XRTFLD_RANGE_UNSIGNED_LONG	long unsigned integer (XRTFLD_RANGE_UNSIGNED_INT と同じ)

XmNxrtFldSearchPolicy　XrtFldSearchPolicy　XRTFLD_SEARCH_SINGLE　CSG

エンドユーザが、キーを押してポップアップ・メニューから選択する場合に適用される検索方法を指定します。有効な値は、次の通りです。

XRTFLD_SEARCH_SINGLE	一つの文字を使用した検索。エンドユーザがキーを押す度に、最初の文字がそのキーにマッチする最初のエレメントがカレント・アイテムにセットされる。キーが押される度に、検索キーが新しくなる。
XRTFLD_SEARCH_MULTICHAR	複数の文字を使用した検索。エンドユーザがキーを続けて押すと、それらのキーにマッチするエレメントがカレント・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。

XmNxrtFldSpinCallback　XtCallbackList　NULL　CSG

ユーザが *Spin()*を開始した時、または XrtSpinBox の arrow(矢印)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。XrtFldSpinCallbackStruct 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ています。

```
void
CallbackProcedure(
  Widget  field;
  XtPointer   client_data,
  XtPointer   call_data /* points to an
     XrtFldSpinCallbackStruct */
  )
```

XrtFldSpinCallbackStruct は、次のように定義されています。

```
typedef struct {
  XrtFldReason    reason; /* read-only */
  XEvent      *event; /* read-only */
  XrtFldDirection direction;  /* read-only */
  Boolean         at_boundary;    /* read-only */
  Boolean         doit;       /* initially True */
} XrtFldSpinCallbackStruct;
```

spin コールバックが指定されていない場合は、XmNxrtFldPickList の次のアイテムが表示されます。

reason は、XRTFLD_REASON_SPIN_BEGIN または XRTFLD_REASON_SPIN_END です。*direction* は XRTFLD_DIRECTION_PREVIOUS または XRTFLD_DIRECTION_NEXT です。親の XmNxrtFldSpinType リソースが XRTSPIN_TYPE_PICKLIST にセットされ、現在の値が pick list の最初または最後にある時は、*at_boundary* は True です。*doit* は、*reason* が XRTFLD_REASON_SPIN_BEGIN でインクレメントが起こらない時は、False にセットしてください。このリストにコールバックを追加するには、*XtAddCallback()*を使用してください。

XmNxrtFldSpinPolicy　XrtFldSpinPolicy　XRTFLD_SPIN_FIELD　CSG

Spin の種類が XRTSPIN_TYPE_COUNTER の時の spin の決まりを指定します。有効な値は次の通りです。

XRTFLD_SPIN_FIELD	通常の spin
XRTFLD_SPIN_SUBFIELD	カーソルがある日付サブ・フィールドが spin
XRTFLD_SPIN_WRAP	max から min、min から max というようにラップする

XmNxrtFldValidateCallback XtCallbackList NULL CSG

検証プロセスの最後の段階で開始されるコールバックのリストを指定します。XrtFldValidateCallbackStruct 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
CallbackProcedure(
  Widget  field;
  XtPointer   client_data,
  XtPointer   call_data /* points to an
     XrtFldValidateCallbackStruct */
  )
```

XrtFldValidateCallbackStruct は、次のように定義されています。

```
typedef struct {
  XrtFldReason    reason; /* read-only */
  XEvent      *event; /* read-only */
  XrtFldAction    action; /* read-only */
  int         num_calls;  /* read-only */
  XmTextPosition  startPos, endPos; /* read-only */
  XmTextPosition  currInsert; /* read-only */
  XmTextPosition  newInsert;
  XmTextBlock         text;
  Boolean         doit;       /* initially True */
  String      value;
} XrtFldValidateCallbackStruct;
```

reason は XRTFLD_REASON_VALIDATE です。*event* はコールバックの原因となった X イベントです。*action* は現在処理されているアクションで、XRTFLD_ACTION_KEYSTROKE(keystroke[キーボード]アクション)、XRTFLD_ACTION_PASTE(ペースト操作のアクション)、XRTFLD_ACTION_EXIT(user exit[ユーザがフィールドを出る]アクション)、XRTFLD_ACTION_SETVALUE(programmatic change[プログラミングによる変更]アクション)のどれかです。

currInsert、*newInsert*、*startPos*、*endPos*、*text* メンバは、XrtFldConvertCallbackStruct 構造体の対応するメンバと同じです。

value は、フィールドに記憶される値全体を持ちます。この値は、それ以前に指定されている変換や補完も含みます。

value のフォーマットは、*action* に依存します。検証コールバックが、user exit アクション、paste または keystroke アクションで開始される場合は、*value* は、フィールドの編集フォーマット(XmNxrtFldEditFormat で指定)を使用します。検証コールバックが programmatic change アクションで開始される場合は、*value* はフィールドの表示フォーマット(XmNxrtFldDisplayFormat で指定)を使用します。

doit は、*text* に記憶されている値を有効と扱うかどうか示します。コールバック・ルーチンが、値は無効であると決定すると、ルーチンは *doit* を False にセットします。

コールバック・ルーチンは、*text* や *value* に記憶されている値を修正することができます。*text* または *value* によって指されている記憶領域を解放し、新しい値に記憶領域を割り当て、セットし、*text* または *value* を新しい記憶領域を指すようにしてください。

text または *value* が修正されると、編集プロセス全体(変換、補完、検証)が、この検証コールバックも含めてもう一度実行されます。*num_calls* は、検証コールバックが呼び出される度にインクレメントされます。

text と *value* の両方が修正された場合は、*value* が優先されます。

このリストにコールバックを追加するには、*XtAddCallback()*を使用してください。

XmNxrtFldValidChars　String　NULL　CSG

入力可能な文字ストリングを指定します。`XmNxrtFldInvalidChars` もセットされている場合は、提供された文字に対して、無効なストリングが先に検索されます。

XmNxrtFldVisibleItemCount　int　10　CSG

*Menu()*アクションによって現れるドロップダウン・メニュー上に表示できる最大アイテム数を指定します。アイテムの数がこの値より大きい場合は、スクロールバーが表示されます。

A.3 XmText リソース

フィールドは全ての XmText リソースを継承します。以下のリストは、これらのリストの一覧ですが、詳細は、Motif 関連の資料を参照してください。

XmNactivateCallback XtCallbackList NULL CSG

(XmNeditMode が XmSINGLE_LINE_EDIT に指定され、)ウィジェット内で Return キーが押された時に開始されるコールバックです。

XmNautoShowCursorPosition Boolean True CSG

カーソルを表示する場合、テキストをスクロールさせるかどうか指定します。

XmNblinkRate int 500 CSG

カーソルが点滅を始めるまでの表示、非表示の時間(ミリ秒)。

XmNcolumns short dynamic CSG

テキスト・ウィンドウに表示されるテキスト文字の数。

XmNcursorPosition XmTextPosition 0 CSG

XmText ウィジェット内のカーソルの位置。

XmNcursorPositionVisible Boolean True CSG

カーソルの表示、非表示を指定します。

XmNeditable Boolean True CSG

ユーザがテキスト・ストリングを編集できるかどうか指定します。

XmNeditMode int XmSINGLE_LINE_EDIT CSG

ユーザが複数行のテキストを入力できるかどうか指定します。

XmNfocusCallback XtCallbackList NULL CSG

XmText ウィジェットが入力フォーカスを得た時に開始されるコールバック。

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。

XmNgainPrimaryCallbak XtCallbackList NULL CSG

XmText ウィジェットが、一次選択の所有権を得た時に開始されるコールバック。

XmNlosePrimaryCallbak XtCallbackList NULL CSG

XmText ウィジェットが、一次選択の所有権を失った時に開始されるコールバック。

XmNlosingFocusCallback XtcallbackList NULL CSG

XmText ウィジェットが入力フォーカスを失った時に開始されるコールバック。

XmNmarginHeight	Dimension	5	CSG

ウィジェットの上下とテキストの間のスペース。

XmNmarginWidth	Dimension	5	CSG

ウィジェットの左右とテキストの間のスペース。

XmNmaxLength	int	largest integer	CSG

テキスト・ストリングの最大の長さ。

XmNmodifyVerifyCallback	XtCallbackList	NULL	CSG

テキスト・ストリングが変更される前に開始されるコールバック。

XmNmotionVerifyCallback	XtCallbackList	NULL	CSG

カーソルが移動される前に開始されるコールバック。

XmNpendingDelete	Boolean	True	CSG

新しいテキストが挿入されたら、直ちに選択されたテキストを削除するかどうか指定します。

XmNresizeHeight	Boolean	False	CSG

複数行のテキストを表示するために、テキスト・ウィジェットを縦に広げるかどうか指定します。

XmNresizeWidth	Boolean	False	CSG

長いテキストを表示するために、テキスト・ウィジェットを横に広げるかどうか指定します。

XmNrows	short	dynamic	CSG

現在表示されているテキストの行数。

XmNselectionArray	XtPointer	default array	CSG

複数のマウス・クリックによるアクションを表す配列。

XmNselectionArryCount	int	4	CSG

`XmNselectionArray` によって指定された配列の中のアイテムの数。

XmNselectThreshold	int	5	CSG

テキストの次の文字の選択に必要な、カーソルがドラッグされる pixel 数。

XmNsource	XmTextSource	Default source	CSG

テキストの表示に XmText ウィジェットが使用するソース。

XmNtopCharacter	XmTextPosition	0	CSG

ウィンドウの最上部に現在表示されているテキストの位置。

XmNvalue	String	“”	CSG

ウィジェットに表示するテキスト・ストリング。

XmNvalueWcs	wchar_t*	(wchar_t*)""	CSG

wchar_t のポインタとして定義された、ウィジェットに表示するテキスト・ストリング。(このリソースは、XRT/field サブクラスではサポートされていません。)

XmNvalueChangedCallback	XtCallbackList	NULL	CSG

テキスト・ストリングが変更された後に開始されるコールバック。

XmNverifyBell	Boolean	True	CSG

ストリングの検証が何もアクションを起こさない時にベルを鳴らすかどうか指定します。

XmNwordWrap	Boolean	False	CSG

行を単語で区切るかどうか指定します。

A.4 継承された XrtField リソース

次の表は、XRT/field が親のクラスから継承するリソースの一覧です。リソースは、それらを定義する親のクラスと一緒にアルファベット順に並べられています。各リソースの詳細は、Motif または Xt の関連資料を参照してください。

Resources	Inherited From
XmNaccelerators	Core
XmNancestorSensitive	Core
XmNbackground	Core
XmNborderColor	Core
XmNborderPixmap	Core
XmNborderWidth	Core
XmNbottomShadowColor	Primitive
XmNbottomShadowPixmap	Primitive
XmNcolormap	Core
XmNdepth	Core
XmNdestroyCallback	Core
XmNforeground	Primitive
XmNheight	Core
XmNhighlightColor	Primitive
XmNhighlightOnEnter	Primitive
XmNhighlightPixmap	Primitive

Resource	Inherited From
XmNhighlightThickness	Primitive
XmNinitialResourcesPersistent	Core
XmNmappedWhenManaged	Core
XmNnavigationType	Primitive
XmNscreen	Core
XmNsensitive	Core
XmNshadowThickness	Primitive
XmNtopShadowColor	Primitive
XmNtopShadowPixmap	Primitive
XmNtranslations	Core
XmNtraversalOn	Primitive
XmNunitType	Primitive
XmNuserData	Primitive
XmNwidth	Core
XmNx	Core
XmNy	Core

A.5 XrtComboBox リソース

XrtComboBox のアルファベット順のリストです。

XmNxrtComboArrow　Widget　dynamic　G

子 arrow(矢印)ボタンのウィジェット ID。

XmNxrtComboArrowClass　WidgetClass　xmArrowButtonWidgetClass　C

子arrow(矢印)ボタンのウィジェット・クラス。このリソースは、作成時にのみセットできます。

XmNxrtComboArrowType　XrtFldArrowType　XRTFLD_ARROW_WEDGE　C

子 arrow(矢印)ボタンの表示を指定します。このリソースは、XmNxrtComboArrowClass が xmDrawnButtonWidgetClass にセットされている場合にのみ使用されます。有効な値は、次の通りです。

XRTFLD_ARROW_NONE	pixmap なしで Arrow ボタンが表示される
XRTFLD_ARROW_WEDGE	Windows 95 で使用されている矢印と似た pixmap を持つ Arrow
XRTFLD_ARROW_WITH_UNDERBAR	Windows 3.1 で使用されている矢印と似た pixmap を持つ Arrow
XRTFLD_ARROW_CUSTOM	Arrow はアプリケーションによって異なる

XmNxrtComboField　Widget　dynamic　G

子フィールドのウィジェット ID。

XmNxrtComboFieldClass　WidgetClass　NULL　C

子フィールドのウィジェット・クラス。このリソースは、XrtComboBox が作成される時にのみセットできます。このリソースが NULL の場合は、XmNxrtComboFieldType の値が使用されます。

XmNxrtComboFieldType　XrtFldFieldType　XRTFLD_FIELDTYPE_STRING　C

子フィールドのクラスを指定します。有効な値は次の通りです。

XRTFLD_FIELDTYPE_CURRENCY

XRTFLD_FIELDTYPE_DATE

XRTFLD_FIELDTYPE_FLOAT

XRTFLD_FIELDTYPE_INT

XRTFLD_FIELDTYPE_STRING

このリソースは、XrtComboBox が作成される時にのみセットでき、XmNxrtComboFieldClass がセットされていると無視されます。

XmNxrtComboSpacing　Dimension　0　CSG

子フィールドと子 arrow ボタンとの間を pixel で指定します。

A.6 XrtSpinBox リソース

XrtSpinBox のアルファベット順のリストです。

XmNxrtSpinArrowLayout XrtSpinArrowLayout XRTSPIN_ARROWS_SPLIT CSG

arrow(矢印)の位置を指定します。有効な値は次の通りです。

XRTSPIN_ARROWS_BEGINNING ボックスの左側に上下の矢印

XRTSPIN_ARROWS_END ボックスの右側に上下の矢印

XRTSPIN_ARROWS_SPLIT ボックスの両側にそれぞれ左右の矢印

XmNxrtSpinArrowOrientation unsigned char Xm_HORIZONTAL CSG

arrow(矢印)の方向を指定します。有効な値は次の通りです。

XmHORIZONTAL 左と右方向の矢印

XmVERTICAL 上と下方向の矢印

XmNxrtSpinArrowLayout が XRTSPIN_ARROWS_BEGINNING または XRTSPIN_ARROWS_END にセットされ、XmNxrtSpinArrowOrientation が Xm_VERTICAL にセットされていると、縦方向の矢印が上下に配置されて表示されます。

XmNxrtSpinArrowType XrtFldArrowType XRTFLD_ARROW_WEDGE C

arrow(矢印)の表示を指定します。このリソースは、XmNxrtSpinDecrArrowClass と XmNxrtSpinIncrArrowClass が xmDrawnButtonWidgetClass にセットされている場合にのみ使用されます。有効な値は、次の通りです。

XRTFLD_ARROW_NONE	pixmap なしで Arrow ボタンが表示される
XRTFLD_ARROW_WEDGE	Windows 95 で使用されている矢印と似た pixmap を持つ Arrow
XRTFLD_ARROW_WITH_UNDERBAR	Windows 3.1 で使用されている矢印と似た pixmap を持つ Arrow
XRTFLD_ARROW_CUSTOM	Arrow はアプリケーションによって異なる

XmNxrtSpinDecrArrow Widget dynamic G

子デクレメント(左または下方向)矢印ボタンのウィジェット ID。

XmNxrtSpinDecrArrowClass WidgetClass xmArrowButtonWidgetClass C

子デクレメント(左または下方向)矢印ボタンのウィジェット・クラス。このリソースは、XrtSpinBox が作成される時にのみセットできます。

XmNxrtSpinField Widget dynamic G

子フィールドのウィジェット ID。

XmNxrtSpinFieldClass WidgetClass NULL C

子フィールドのウィジェット・クラス。このリソースは、SpinBox の作成時にのみセットできます。NULL の場合は、XmNxrtComboFieldType 値が使用されます。

XmNxrtSpinFieldName　String　“field”　C

子フィールドの名前。このリソースは、XrtSpinBox 作成時にのみセットできます。

XmNxrtSpinFieldType　XrtFldFieldType　XRTFLD_FIELDTYPE_STRING　C

子フィールドのクラスを指定します。有効な値は次の通りです。

```
XRTFLD_FIELDTYPE_CURRENCY
XRTFLD_FIELDTYPE_DATE
XRTFLD_FIELDTYPE_FLOAT
XRTFLD_FIELDTYPE_INT
XRTFLD_FIELDTYPE_STRING
```

このリソースは、XrtSpinBox ウィジェットの作成時にのみセットでき、`XmNxrtSpinFieldClass` がセットされている時は無視されます。

XmNxrtSpinIncrArrow　Widget　dynamic　G

子インクレメント(右か上方向)矢印ボタンのウィジェット ID。

XmNxrtSpinIncrArrowClass　WidgetClass　xmArrowButtonWidgetClass　C

子インクレメント(右か上方向)矢印ボタンのウィジェット・クラス。このリソースは、XrtSpinBox ウィジェットの作成時にのみセットできます。

XmNxrtSpinSpacing　Dimension　0　CSG

フィールドと arrow ボタン・ウィジェットの(横方向の)間隔を pixel で指定します。

XmNxrtSpinType　XrtSpinType　XRTSPIN_TYPE_PICKLIST　CSG

子フィールドのデータ型。有効な値は次の通りです。

`XRTSPIN_TYPE_COUNTER`	`XmNxrtFldMin` と `XmNxrtFldMax` で定義される範囲をスピンする。インクレメントは `XmNxrtFldCounterIncrement` が指定。
`XRTSPIN_TYPE_PICKLIST`	`XmNxrtFldPickList` のアイテムをスピンする。

A.7 継承された XrtComboBox と XrtSpinBox リソース

以下の表は、XrtComboBox と XrtSpinBox が親のクラスから継承するリソースの一覧です。リソースは、それらを定義する親のクラスと一緒にアルファベット順に並べられています。各リソースの詳細は、Motif または Xt の関連資料を参照してください。

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

Resource	Inherited From
XmNinsertPosition	Composite
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

A.8 XrtCalendar リソース

XrtCalendar のアルファベット順のリストです。

XmNxrtCalendarCellMargin Dimension 0 CSG

各カレンダのセルの周囲のマージンを指定します。

XmNxrtCalendarCellMargin Dimension 0 CSG

セルのハイライトの太さを指定します。ハイライトは選択されているカレンダ・セルの周囲に描画されます。

XmNxrtCalendarCreateMonth Boolean True C

このリソースが `True` の場合、カレンダの左上の部分に月の spin box を作成します。

XmNxrtCalendarCreateTime Boolean False C

このリソースが `True` の場合、カレンダの下の部分に時間の spin box を作成します。

XmNxrtCalendarCreateYear Boolean True C

このリソースが `True` の場合、カレンダの右上の部分に年の spin box を作成します。

XmNxrtCalendarEnabled Boolean True CSG

ユーザ・インタラクションを許可するかどうかを指定します。

XmNxrtCalendarFirstDayofWeek int 0 CSG

週の最初の日を指定します。通常は、Sunday(day 0)ですが、変更可能です。

XmNxrtCalendarFullDayNames Boolean True CSG

曜日名を略するかどうかを指定します。`True` の場合は、使用しているロケールの曜日のフル・ネームが使用されます。

XmNxrtCalendarHeaderNumChars int 1 CSG

各月を表す文字数を指定します。

XmNxrtCalendarHeaderBackground Pixel dynamic CSG

ヘッダーの背景色を指定します。変更されない場合は、このリソースは `XmNbackground` と同じです。

XmNxrtCalendarHeaderFontList XmFontList dynamic CSG

ヘッダーに使用されるフォント・リストを指定します。変更されない場合は、このリソースは `XmNfontList` と同じです。

XmNxrtCalendarHeaderForeground Pixel dynamic CSG

ヘッダーの前景色を指定します。変更されない場合は、このリソースは、XmNforeground と同じです。

XmNxrtCalendarMarginBottom Dimension 0 CSG

XrtCalendar ウィジェットの下部のマージンを指定します。

XmNxrtCalendarMarginLeft Dimension 0 CSG

XrtCalendar ウィジェットに左横のマージンを指定します。

XmNxrtCalendarMarginRight Dimension 0 CSG

XrtCalendar ウィジェットの右横のマージンを指定します。

XmNxrtCalendarMarginTop Dimension 0 CSG

XrtCalendar ウィジェットの上部のマージンを指定します。

XmNxrtCalendarMaxRollDate struct tm * NULL CSG

ユーザが選択できる最大の日を指定します。このリソースで指定された日より後の日は許可されません。

XmNxrtCalendarMinRollDate struct tm * NULL CSG

ユーザが選択できる最小の日を指定します。このリソースで指定された日より前の日は許可されません。

XmNxrtCalendarModifyCallback XtCallbackList NULL CSG

ユーザが日を変更し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。XrtCalendarModifyCallbackStruct 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
CallbackProcedure(
   Widget      field;
   XtPointer   client_data,
   XtPointer   call_data /* points to XrtCalendarModifyCallbackStruct */
)
```

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

```
typedef struct {
   XrtFldReason  reason; /* read-only */
   Xevent        *event; /* read-only */
   struct tm     *date;
   Boolean       doit; /* initially true */
} XrtCalendarModifyCallbackStruct;
```

reason は、どのようにカレンダ・ウィジェットが変更されるかに依存します。日は、spin month ウィジェット、spin day ウィジェット、spin time ウィジェット、day grid、値の設定によって変更でき、それぞれの場合、*reason* は、XRTFLD_REASON_CALENDAR_MONTH、XRTFLD_REASON_CALENDAR_YEAR、XRTFLD_REASON_CALENDAR_TIME、XRTFLD_REASON_CALENDAR_DAYGRID、XRTFLD_REASON_CALENDAR_SETVALUES となります。*date* はユーザが選択した日と時間を記憶する構造体のポインタです。*doit* が False であれば、選択された日は許可されません。

XmNxrtCalendarSpecialDayBackground Pixel dynamic CSG

special day の背景に使用される色を指定します。変更されない場合は、このリソースは XmNbackground と同じです。

XmNxrtCalendarSpecialDayForeground Pixel dynamic CSG

special day の前景に使用される色を指定します。変更されない場合は、このリソースは XmNforeground と同じです。

XmNxrtCalendarSpecialDays XrtCalendarSpecialDay **NULL CSG

このリソースは、special day のリストを指します。リストの各エントリは、special day または special day のリストをセットします。これらの special day は、上記記述リソースで指定される前景色と背景色で表示されます。尚、このリストの終端は NULL にしてください。

XmNxrtCalendarSelectedBackground Pixel dynamic CSG

選択された部分の背景色を指定します。変更されない場合は、このリソースは XmNforeground と同じです。(デフォルトでは、選択は他の部分と逆の色で表示されます。)

XmNxrtCalendarSelectedForeground Pixel dynamic CSG

選択された部分の前景色を指定します。変更されない場合は、このリソースは XmNbackground と同じです。(デフォルトでは、選択は他の部分と逆の色で表示されます。)

XmNxrtCalendarValue struct tm * the current time CSG

カレンダ表示の現在の値を指定します。この値は、ユーザが値を選択するとアップデートされます。このリソースを変更すると、副作用で XmNxrtCalendarValueLong が変更されます。

XmNxrtCalendarValueLong long the current time CSG

カレンダ表示の現在の値を long として指定します。カレント値が long として扱えない場合(おおよそ 1902 年から 2033 年の範囲外)、このリソースは XRTFLD_MAXLONG にセットされます。このリソースを変更すると、副作用で XmNxrtCalendarValue が変更されます。

B

手続きと方法

XRT/field の手続きと方法のアルファベット順のリストです。

XmCreateXrtCalendar()

XrtCalendar ウィジェットを作成します。

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

parent は親、*name* は作成されたウィジェットの名前です。*arglist* はウィジェットに適用される *argcount* 引数のリストです。全ての子供は manage されます。

XmCrateXrtComboBox()

フィールドと arrow(矢印)ボタンを子として、XrtComboBox ウィジェットを作成します。矢印ボタンの名前は `arrow`、フィールドの名前は `field` です。combo ボックスのウィジェット ID が返されます。

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

parent は親、*name* は作成されたフィールドの名前です。*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(managed)されます。

XmCreateXrtCurrencyField()

currency(通貨)フィールドを作成します。

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

parent は親、*name* は作成されたウィジェットの名前です。*arglist* は *argcount* 引数のリストです。

XmCreateXrtDateField()

date(日付)フィールドを作成します。

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

parent は親、*name* は作成されたウィジェットの名前です。*arglist* は *argcount* 引数のリストです。

XmCreateXrtFloatField()

float フィールドを作成します。

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

parent は親、*name* は作成されたウィジェットの名前です。*arglist* は *argcount* 引数のリストです。

XmCrateXrtIntField()

integer(整数)フィールドを作成します。

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

parent は親、*name* は作成されたウィジェットの名前です。*arglist* は *argcount* 引数のリストです。

XmCreateXrtSpinBox()

フィールドと二つの arrow(矢印)ボタンを子として XrtSpinBox ウィジェットを作成します。矢印の名前は `incr_arrow` と `decr_arrow` です。指定されていない場合は、フィールドの名は `field` です。追加のフィールドや他のウィジェットは、子として作成することにより、XrtSpinBox に追加することができます。spin ボックスのウィジェット ID が返されます。

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

parent は親、*name* は作成されたフィールドの名前です。*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(managed)されます。

XmCreateXrtStringField()

string フィールドを作成します。

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

parent は親、*name* は作成されたウィジェットの名前です。*arglist* は *argcount* 引数のリストです。

XmIsXrtCalendar()

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

```
Widget
XmIsXrtCalendar(
      Widget  widget
      )
```

XmIsXrtComboBox()

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

```
int
XmIsXrtComboBox(
  Widget  widget
  )
```

XmIsXrtCurrencyField()

ウィジェットが currency フィールドまたはサブクラスであれば `True` を返します。

```
int
XmIsXrtCurrencyField(
  Widget  widget
  )
```

XmIsXrtDateField()

ウィジェットが date フィールドまたはサブクラスであれば True を返します。

```
int
XmIsXrtDateField(
  Widget  widget
  )
```

XmIsXrtFloatField()

ウィジェットが float フィールドまたはサブクラスであれば True を返します。ウィジェットが currency フィールドの場合にも True が返されます。

```
int
XmIsXrtFloatField(
  Widget  widget
  )
```

XmIsXrtField()

ウィジェットが、XrtField のサブクラスまたは、サブクラスであれば True を返します。

```
int
XmIsXrtField(
  Widget  widget
  )
```

XmIsXrtIntField()

ウィジェットが integer フィールドまたはそのサブクラスであれば True を返します。

```
int
XmIsXrtIntField(
  Widget  widget
  )
```

XmIsXrtSpinBox()

ウィジェットが XrtSpinBox ウィジェットであれば True を返します。

```
int
XmIsXrtSpinBox(
  Widget  widget
  )
```

XmIsXrtStringField()

ウィジェットが string フィールドであれば True を返します。

```
int
XmIsXrtStringField(
  Widget  widget
  )
```

XrtCalendarAddSpecialDay()

リストの最後に special day を追加します。成功すると True を返します。

```
Boolean
XrtCalendarAddSpecialDay(
     Widget          widget,
     XrtCalendarSpecialDay    *special_day
     )
```

XrtCalendarDeleteSpecialDay()

リストから special day を削除します。ここで指定する special day は、必ずリストに含まれるアイテムで、構造体で使用されるパラメータにマッチしていなければなりません。成功すると True を返します。

```
Boolean
XrtCalendarDeleteSpecialDay(
     Widget          widget,
     XrtCalendarSpecialDay    *special_day
     )
```

XrtCalendarDupSpecialDays()

special day のリストを複写します。失敗すると NULL を返します。

```
XrtSpecialDay **
XrtCalendarDupSpecialDays(
XrtCalendarSpecialDay     **special_day_list
)
```

XrtCalendarFreeSpecialDays()

special day のリストを解放します。成功すると True を返します。

```
Boolean
XrtCalendarFreeSpecialDays(
XrtCalendarSpecialDay     **special_day_list
)
```

XrtFldCheckFields()

manager ウィジェットの子階層をトラバースし、三つのリスト(全てのフィールドのリスト、無効フィールドのリスト、デフォルト・フィールドのリスト)を持つ構造体を返します。フィールドが見つからなかった場合は、False を返します。

```
Boolean
XrtFldCheckFields(
  Widget  parent,
  XrtFldFieldData *data /* returned */
  )
```

data のリスト・メンバは、*XrtFldFieldDataFree()*を呼び出して、使用後必ず解放してください。

無効フィールドのリストは、XmNxrtFldValidateCallback リストのコールバックによって無効なステータスが置き換えられたフィールドを含みます。

XrtFldCvtLongToTime()

long 整数を *struct tm* 構造体に変換します。long 整数、*value* は *time_t* 構造体と同じで、1970 年 1 月 1 日からの秒数を持っています。(*value* の負の数は、1970 年以前の日とみなされます。)

```
void
XrtFldCvtLongToTime(
  long    value,
  struct tm   *date /* returned */
  )
```

XrtFldCvtStringToDate()

ストリングを *struct tm* 構造体に変換します。変換に失敗すると `False` を返します。

```
Boolean
XrtFldCvtStringToDate(
  String  input,
  String  mask,
  int     millenium_threshold,
  struct tm   *result /* returned */
  )
```

input は変換されるストリング、*mask* はストリングを読むときに使用される日付フォーマットで、このフォーマットの構成は、`XmNxrtFldEnterFormatList` リソースに使用されるものと同じです。*millenium_threshold* より小さい二桁の年号は、21 世紀の年号として解釈されます。(例: *millenium_threshold* が 70 で *input* が 69 であれば、*XrtFldCvtStringToDate()*は 69 は 2069 と解釈します。)

input は、"`TODAY`"として深夜を基準とする今日の日付や、"`NOW`"として現在時刻を指定することができます。

XrtFldCvtStringToTime()

ストリングを *struct tm* 構造体に変換します。変換に失敗すると `False` を返します。

```
Boolean
XrtFldCvtStringToTime(
  String  value,
  String  format,
  struct  tm *date /* returned */
  )
```

value は変換されるストリング。*format* は、ストリングを読む時に使用する日付フォーマットで、このフォーマットの構成は、`XmNxrtFldEnterFormatList` に使用されるものと同じです。

value は、深夜 12 時を基準とした今日の日付である"`TODAY`"、または現在時刻を指定する"`NOW`"にできます。

XrtFldCvtTimeToLong()

struct tm 構造体を、(1970 年 1 月 1 日からの秒数を持つ)*time_t* 値と同じである long 整数に変換します。*time_t* 型の値と異なり、この手続きによって返される値は、1970 年以前の日付を負の数として表現できます。*tm_gmtoff* と *tm_zone* 構造体メンバは無視されます。

```
Boolean
XrtFldCvtTimeToLong(
  struct tm   *date,
  long    *value /* returned */
  )
```

この手続きは、構造体メンバが無効であったり、日付が long 整数に合わない時は `False` を返します。(32-bit マシンでは、おおよそ 1902 年から 2033 年の間の年がこの手続きによって返されます。)

XrtFldDoubleToArgVal()

float または currency フィールドのリソース値を設定する時に使用できるよう、*double* を XtArgVal に変換します。

```
XtArgVal
XrtFldDoubleToArgVal(
  double  value
  )
```

XrtFldFieldDataFree()

*XrtFldCheckFields()*で割り当てられたデータを解放します。

```
Boolean
XrtFldFieldDataFree(
  XrtFldFieldData *data
  )
```

XrtFldGetCurrencyLocale()

現在使用されている、通貨のロケール値(*setlocale()*の category 値、LC_NUMERIC または LC_CURRENCY や *XrtFldSetDefaultLocaleStruct()*でセットされる)を取得します。

```
void
XrtFldGetCurrencyLocale(
  struct lconv    *data
  )
```

data は返さたデータで、read-only です。*data* は解放することはできません。

XrtFldGetDateLocale()

現在使用されている、日付/時間のロケールの値を取得します。

```
void
XrtFldGetDateLocale(
  XrtFldDTConv    *data
  )
```

data は返さたデータで、read-only です。*data* は解放することはできません。

XrtFldGetNumericLocale()

現在使用されている、数値のロケールの値(*setlocale()*の category 値 LC_NUMERIC や *XrtFldSetDefaultLocaleStruct()*でセットされる)を取得します。

```
void
XrtFldGetNumericLocale(
  struct lconv    *data
  )
```

data は返さたデータで、read-only です。*data* は解放することはできません。

XrtFldGetDisplayValue()

フィールドを出た時に表示されるストリングを取得します。返されたストリングは、使用後解放してください。

```
String
XrtFldGetDisplayValue(
  Widget  field
  )
```

XrtFldGetField()

キャッシュされたフィールドのリストから、指定されたクラスのフィールドを取得するか、または必要な場合はフィールドを作成します。この手続きは、一回に一つのテキスト・ウィジェットを表示する XRT/table と一緒に使用されるよう設計されています。テーブル毎のキャッシュを使用することにより、必要最低限のフィールドが作成されます。

```
Widget
XrtFldGetField(
  Widget      parent,
  String      name,
  WidgetClass         field_class,
  Arg         *arglist,
  int         argcount
  )
```

parent は親のテーブル・ウィジェット、*name* は作成されたフィールドの名前です。*arglist* は、フィールドに適用される *argcount* 引数のリストです。

XrtFldGetPassword()

`XmNxrtFldConvertCallback` の文字を、指定した *echo* 文字にセットします。ユーザがタイプした文字は、*string* パラメータに交換されます。ウィジェットが Xrtfield のサブクラスでない時は、`False` が返されます。通常、*XrtFldGetPassword()*は、アプリケーションのパスワードの入力フィールド用に Convert コールバックで使用されます。

```
Boolean
XrtFldGetPassword(
  Widget  field,
  String  *string, /*returned */
  char    echo,
  XrtFldConvertCallbackStruct *cbs
  )
```

string は、ユーザのキーストロークによってセットされる現在のパスワードです。*string* は `NULL` に初期化し、使用後は必ず解放してください。*echo* はキーストロークされた文字に代わって表示される文字です。*echo* が 0 であれば、キーストロークがそのままエコーされます。*cbs* は、渡される Convert コールバック構造体です。

XrtFldMrmInitialize()

UIL アプリケーションに使用できるよう、XRT/field ウィジェット・クラスを Mrm に登録します。この手続きは、*MrmInitialize()*の後に呼び出してください。

```
void
XrtFldMrmInitialize(void)
```

XrtFldSetDefaultLocaleStruct()

*setlocale()*が呼び出されていなかったり、無効なメンバが与えられている場合に、global ロケール・データをセットします。使用している OS がロケール情報をサポートしていない場合は、*dt_conv* を使用してロケール・データを提供できます。

この手続きは、データが無効な場合は `False` を返します。

```
Boolean
XrtFldSetDefaultLocaleStruct(
  struct lconv    *locale,
  XrtFldDTconv    dt_conv
  )
```

locale と *dt_conv* が `NULL` であると、US ロケールのデータが使用されます。

XrtFldSetReverseConverters()

*XtConvertAndStore()*と使用される XRT/field の逆変換プログラムを登録します。逆変換プログラムは、アプリケーションに、人間が読める形のリソース値を提供します。

```
void
XrtFldSetReverseConverters(void)
```

変換プログラムは、XRT/field のデータ型にのみ使用できます。

XrtFldValidateValue()

フィールドの現在の `XmNvalue` ストリングを検証し、値が無効であれば `False` を返します。有効な場合は、フィールドの内部値(`XmNxrtFldInternalValue`)がセットされます。検証は、フィールドを出る時に自動的に行われるので、この手続きは、通常キーストロークによるアクションによってのみ呼び出されます。

```
Boolean
XrtFldValidateValue(
  Widget field
  )
```

C

マクロ

次の表のマクロは、時間の値を変換する時に使用できます。これらは、*$XRTHOME/include/Xm/XrtField.h* に定義されています。

マクロ名	定義
MIN_TO_SEC(m) **分の値(*m*)を秒数に変換する**	**(60L * (m))**
HOUR_TO_SEC(h) **時間の値(*h*)を秒数に変換する**	**(3600L * (h))**
DAY_TO_SEC(d) **日(*d*、1 から 366 の値)を秒数に変換する**	**(86400L * (d))**
MONTH_TO_SEC(m) **月(*m*)を秒数に変換する**	**(2592000L * (m))**
YEAR_TO_SEC(y) **年(y)を秒数に変換する**	**(31536000L * (y))**

D

トランスレーションとアクション

デフォルトのトランスレーション

アクション･ルーチン

D.1 デフォルトのトランスレーション

XrtField ウィジェットは、XmText の全てのトランスレーションを継承し、独自に以下のようなトランスレーションを定義します。

	Event	Action
全てのフィールド	<Key>osfCancel	Cancel()
	Ctrl<Key>s	Scrollbars()
	Ctrl<Key>S	Scrollbars()
	Ctrl<Key>osfDown	Menu(Display)
	Ctrl<Key>osfUP	Menu(Hide)
	Ctrl <Key>osfDown	Calendar(Display)
	Ctrl <Key>osfUp	Calendar(Hide)
XrtSpinBox のフィールド	<Key>osfRight	Spin(Next)
	<Key>osfUp	Spin(Next)
	<Key>osfLeft	Spin(Previous)
	<Key>osfDown	Spin(Previous)

XmText のトランスレーションに関する詳細は、Motif 関連の資料を参照してください。

D.2 アクション･ルーチン

XRT/field は、以下のアクション･ルーチンを定義します。

Calendar()

XrtDateField ウィジェットの XrtCalendar ポップアップをコントロールします。このアクションは次のうちの一つのパラメータを取ります。

`Display`	ポップアップを表示
`Hide`	ポップアップを非表示

Ctrl を押しながら下方向の arrow キーを押すとカレンダ･ポップアップが表示/非表示されます。

Cancel()

フィールドに表示されている値(ユーザによって現在編集されている値)を `XmNxrtFldInternalValue` の値と置き換えます。この値は、通常、フィールドの編集フォーマット(サブクラスに定義されている場合は `XmNxrtEditFormat` で指定)で表示されます。

Complete()

(マッチする値があった場合に)`XmNxrtFldPickList` の内容を利用して値を補完します。数値フィールドにリストが指定されていない場合は、0 が使用されます。date フィールドにリストが指定されていない場合は、現在の日が使用されます。補完が実行できない時は、ビープ音が鳴ります。

Menu()

ポップアップ･メニューに `XmNxrtFldPickList` アイテムを表示します。各アイテムは、フィールドの編集フォーマットを利用して表示されます。このアクションは、一つのパラメータを取ります。

`Display`	ポップアップ･メニューを表示
`Hide`	ポップアップ･メニューを非表示

ユーザは、このポップアップ･メニューから選択することにより、フィールドに記憶されている値を変更することができます。その結果、プログラミングによる変更(programmatic change)のアクションが開始されます。

NoAction()

このアクションは何もしません。このルーチンを呼び出すイベントを設定することにより、トランスレーションを不可にすることができます。

Overstrike()

挿入/上書きモード(`XmNxrtFldOverstrike` リソースで指定)をセットします。このアクションは、一つのパラメータを取ります。

`False`	上書きモードをオフ
`True`	上書きモードをオン
`Toggle`	現在のモードとは反対のモードに切り替え

Scrollbars()

スクロールバーの表示をオン/オフに切り替えます。*Scrollbars()*の最初の呼び出しは、XmNxrtFldDisplayHorizScrollBar と XmNxrtFldDisplayVertScrollBar を両方とも XRTFLD_DISPSB_NEVER にセットし、スクロールバーの表示をオフにします。

Scrollbars() の次の呼び出しは、XmNxrtFldDisplayHorizScrollBar を XRTFLD_DISPSB_ALWAYS に、XmNxrtFldDisplayVertScrollBar を XRTFLD_DISPSB_AS_NEEDED にセットし、スクロールバーの表示をオンにします。

それ以降の *Scrollbars()*の呼び出しは、上記のようにスクロールバー表示をオンとオフに切り替えます。

Spin()

XrtSpinBox の XmNxrtSpinType リソースに従って、XmNxrtFldPickList の次のアイテムを表示したり、現在の値をインクレメントします。このアクションは、パラメータを一つ取ります。

Next	次のアイテムを表示するか値をインクレメントする
Prev	前のアイテムを表示するか値をデクレメントする

E

データ型

XRT/field のデータ型のアルファベット順のリストです。

XrtCalendarModifyCallbackStruct

XmNxrtCalendarModifyCallback リストのコールバックに渡される構造体です。

```
typedef struct {
      XrtFldReason reason;
      Xevent       *event;
      struct tm    *date;
      Boolean      doit;
      } XrtCalendarModifyCallbackStruct;
```

XrtCalendarReason

コールバックの reason を指定する列挙法です。

```
XRTFLD_REASON_CALENDAR_MONTH
XRTFLD_REASON_CALENDAR_YEAR
XRTFLD_REASON_CALENDAR_TIME
XRTFLD_REASON_CALENDAR_DAYGRID
XRTFLD_REASON_CALENDAR_SETVALUES
```

XrtCalendarSpecialDay

special day を定義する構造体です。

```
typedef struct {
      XrtCalendarSpecialDayType type;

/* the rest of the structure is interpreted based on the type */

      int day;
      int day_of_week; /* 0 == Sun && 6 == Sat */
      int week;        /* 1 based */
      int month;       /* 0 == Jan and 11 == Dec */
      int year;        /* Full year. 1998 not 98 */
      } XrtCalendarSpecialDay;
```

XrtCalendarSpecialDayType

追加または削除される special day の型を指定する列挙法です。

```
XRTCALENDAR_SPECIAL_DAY_BY_DATE
XRTCALENDAR_SPECIAL_DAY_BY_DAY_OF_WEEK
```

XrtFldAction

コールバックを呼び出すアクション(イベント)を指定する列挙法です。

```
XRTFLD_ACTION_EXIT
XRTFLD_ACTION_KEYSTROKE
XRTFLD_ACTION_SETVALUE
XRTFLD_ACTION_PASTE
```

XrtFldArrowType

XrtComboBox の XmNxrtComboArrowType に使用され、表示される arrow(矢印)の種類を指定する列挙法です。

```
XRTFLD_ARROW_CUSTOM
XRTFLD_ARROW_NONE
XRTFLD_ARROW_WEDGE
XRTFLD_ARROW_WITH_UNDERBAR
```

XrtFldCalendarCallbackStruct

Calendar コールバックに渡される構造体です。

```
typedef struct {
     XrtFldReason reason; /* read-only */
     Xevent       *event; /* read-only */
     struct tm    *date;
     Boolean      doit; /* initially true */
} XrtFldCalendarCallbackStruct;
```

XrtFldCase

XmNxrtFldCase で大文字･小文字変換の指定に使用される列挙法です。

```
XRTFLD_CASE_AS_IS
XRTFLD_CASE_LOWER
XRTFLD_CASE_UPPER
```

XrtFldCompleteCallbackStruct

XmNxrtFldCompleteCallback リストのコールバックに渡される構造体です。

```
typedef struct {
  XrtFldReason reason;
  XEvent *event;
  String value;
} XrtFldCompleteCallbackStruct;
```

XrtFldConvertCallbackStruct

XmNxrtFldConvertCallback リストのコールバックに渡される構造体です。

```
typedef struct {
  XrtFldReason    reason;
  XEvent      *event;
  XrtFldAction    action;
  XmTextPosition  startPos, endPos;
  XmTextPosition  currInsert;
  XmTextPosition  newInsert;
  XmTextBlock         text;
  String      value;
} XrtFldConvertCallbackStruct;
```

XrtFldDirection

スピンまたはカウンタの方向を指定する、XmNxrtFldspinCallback コールバックに渡される列挙法です。

```
XRTFLD_DIRECTION_NEXT
XRTFLD_DIRECTION_PREVIOUS
```

XrtFldDispScrollBar

XmNxrtFldDisplayHorizScrollBar と XmNxrtFldDisplayVertScrollBar に使用され、スクロールバーの表示を指定する列挙法です。

```
XRTFLD_DISPSB_ALWAYS
XRTFLD_DISPSB_AS_NEEDED
XRTFLD_DISPSB_NEVER
```

XrtFldDTconv

ロケールをサポートしていない環境で、*XrtFldSetDefaultLocaleStruct()*に渡される、日付/時間の情報を定義する構造体です。

```
typedef struct {
  String abbrev_month_names[12];
  String month_names[12];
  String abbrev_weekday_names[7];
  String weekday_names[7];
  String time_format;
  String sdate_format;
  String dtime_format;
  String am_string;
  String pm_string;
  String ldate_format;
  String abbrev_tz;
} XrtFldDTconv;
```

XrtFldEnterCallbackStruct

XmNxrtFldEnterCallback リストのコールバックに渡される構造体です。

```
typedef struct {
  XrtFldReason    reason;
  XEvent      *event;
  String      value;
  XmString        prompt_string;
} XrtFldEnterCallbackStruct;
```

XrtFldError

XmNxrtFldErrorCallback リストのコールバックによって使用され、エラーを指定する列挙法です。

```
XRTFLD_ERROR_EXCEED_LENGTH
XRTFLD_ERROR_EXCEED_LINES
XRTFLD_ERROR_INVALID_CHAR
XRTFLD_ERROR_INVALID_DATATYPE
XRTFLD_ERROR_IS_NULL
XRTFLD_ERROR_NOMATCH_MASK
XRTFLD_ERROR_NOMATCH_PICKLIST
XRTFLD_ERROR_OUT_OF_RANGE
XRTFLD_ERROR_VALIDATECB
XRTFLD_ERROR_NONE
```

XrtFldErrorCallbackStruct

XmNxrtFldErrorCallback リストのコールバックに渡される構造体です。

```
typedef struct {
  XrtFldReason    reason;
  XEvent      *event;
  int         text_reason;
  XrtFldAction    action;
  XrtFldError         error;
  Boolean         is_modified;
  XmTextPosition  startPos, endPos;
  XmTextPosition  currInsert;
  XmTextPosition  newInsert;
  XmTextBlock         text;
  XmString        error_string;
  Boolean         display;
  Boolean         traverse;
} XrtFldErrorCallbackStruct;
```

XrtFldExitCallbackStruct

XmNxrtFldExitCallback リストのコールバックに渡される構造体です。

```
typedef struct {
  XrtFldReason    reason;
  XEvent      *event;
  int         text_reason;
  Boolean         is_modified;
  String      value;
  Boolean         traverse;
} XrtFldExitCallbackStruct;
```

XrtFldFieldData

*XrtFldCheckFields()*に使用され、子孫のフィールドのリストを返す構造体です。

```
typedef struct {
  Widget  *field_list;
  int     num_field_list;
  Widget  *invalid_list;
  int     num_invalid_list;
  Widget  *default_list;
  int     num_default_list;
} XrtFldFieldData;
```

XrtFldFieldType

XmNxrtComboFieldType と XmNxrtSpinFieldType に使用され、子フィールドのサブクラスを指定する列挙法です。

```
XRTFLD_FIELDTYPE_CURRENCY
XRTFLD_FIELDTYPE_DATE
XRTFLD_FIELDTYPE_FLOAT
XRTFLD_FIELDTYPE_INT
XRTFLD_FIELDTYPE_NONE
XRTFLD_FIELDTYPE_STRING
```

XrtFldFormat

XmNxrtFldEditFormat と XmNxrtFldDisplayFormat に使用され、編集と表示フォーマットを指定する列挙法です。

```
XRTFLD_FORMAT_BINARY
XRTFLD_FORMAT_CURRENCY
XRTFLD_FORMAT_CURRENCY_INTL
XRTFLD_FORMAT_DEFAULT
XRTFLD_FORMAT_FIXED
XRTFLD_FORMAT_GENERAL
XRTFLD_FORMAT_HEX
XRTFLD_FORMAT_OCTAL
XRTFLD_FORMAT_PERCENT
XRTFLD_FORMAT_SCIENTIFIC
XRTFLD_FORMAT_THOUSANDS
```

XrtFldMatchMask

XmNxrtFldMatchMask に使用される列挙法で、部分的なマスクの適合が可能かどうか指定します。

```
XRTFLD_MATCH_ALL
XRTFLD_MATCH_PARTIAL_L_TO_R
XRTFLD_MATCH_PARTIAL_R_TO_L
```

XrtFldMenuAlignment

XmNxrtFldMenuAlignment に使用される列挙法です。

```
XRTFLD_MENU_ALIGN_BEGINNING
XRTFLD_MENU_ALIGN_END
XRTFLD_MENU_ALIGN_BOTH
```

XrtFldMenuCallbackStruct

XmNxrtFldMenuCallback リストのコールバックに渡される構造体です。

```
typedef struct {
  XrtFldReason    reason;
  XEvent      *event;
  int         position;
  String      value;
  Boolean         doit;
} XrtFldMenuCallbackStruct;
```

XrtFldNegFormat

XmNxrtFldNegativeFormat に使用される列挙法で、負の数の表示フォーマットを指定します。

```
XRTFLD_NEG_AS_IS
XRTFLD_NEG_LOCALE
XRTFLD_NEG_PARENTHESES
```

XrtFldPopupType

フィールドまたは combo box のポップアップの型の指定に使用される列挙法です。

```
XRTFLD_POPUP_PICKLIST
XRTFLD_POPUP_CALENDAR
```

XrtFldRange

XmNxrtFldRange に使用される列挙法で、フィールドの範囲を指定します。

```
XRTFLD_RANGE_CHAR
XRTFLD_RANGE_DEFAULT
XRTFLD_RANGE_DOUBLE
XRTFLD_RANGE_FLOAT
XRTFLD_RANGE_INT
XRTFLD_RANGE_LONG
XRTFLD_RANGE_SHORT
XRTFLD_RANGE_UNSIGNED_CHAR
XRTFLD_RANGE_UNSIGNED_INT
XRTFLD_RANGE_UNSIGNED_LONG
XRTFLD_RANGE_UNSIGNED_SHORT
```

XrtFldReason

コールバックに使用される列挙法で、コールバックの理由(reason)を指定します。

```
XRTFLD_REASON_COMPLETE
XRTFLD_REASON_CONVERT
XRTFLD_REASON_ENTER
XRTFLD_REASON_ERROR
XRTFLD_REASON_EXIT
XRTFLD_REASON_MENU_BEGIN
XRTFLD_REASON_MENU_END
XRTFLD_REASON_SPIN_BEGIN
XRTFLD_REASON_SPIN_END
XRTFLD_REASON_VALIDATE
XRTFLD_REASON_CALENDAR_BEGIN
XRTFLD_REASON_CALENDAR_END
```

XrtFldSearchPolicy

XmNxrtFldSearchPolicy に使用される列挙法で、pick list の検索法を指定します。

```
XRTFLD_SEARCH_SINGLE
XRTFLD_SEARCH_MULTICHAR
```

XrtFldSpinCallbackStruct

XmNxrtFldSpinCallback リストのコールバックに渡される構造体です。

```
typedef struct {
  XrtFldReason    reason;
  XEvent      *event;
  XrtFldDirection direction;
  Boolean         at_boundary;
  Boolean         doit;
} XrtFldSpinCallbackStruct;
```

XrtFldSpinPolicy

XRTSPIN_TYPE_COUNTER 型の spin box のスピンの種類を指定する列挙法です。

```
  XRTFLD_SPIN_FIELD
  XRTFLD_SPIN_SUBFIELD
  XRTFLD_SPIN_WRAP
```

XrtFldValidateCallbackStruct

XmNxrtFldValidateCallback リストのコールバックに渡される構造体です。

```
typedef struct {
  XrtFldReason    reason;
  XEvent      *event;
  XrtFldAction    action;
  int         num_calls;
  XmTextPosition  startPos, endPos;
  XmTextPosition  currInsert;
  XmTextPosition  newInsert;
  XmTextBlock         text;
  Boolean         doit;
  String      value;
} XrtFldValidateCallbackStruct;
```

XrtSpinArrowLayout

XrtSpinBox ウィジェットに使用される列挙法で、arrow(矢印)ボタンの位置を指定します。

```
XRTSPIN_ARROWS_BEGINNING
XRTSPIN_ARROWS_END
XRTSPIN_ARROWS_SPLIT
```

XrtSpinType

XrtSpinBox ウィジェットの種類を指定する列挙法です。

```
XRTSPIN_TYPE_COUNTER
XRTSPIN_TYPE_PICKLIST
```

F

リソースの型とクラス

XrtField のサブクラスに共通なリソース ■ XrtStringField リソース

XrtIntField リソース ■ XrtFloatField リソース

XrtDateField リソース ■ XrtComboBox リソース

XrtSpinBox リソース ■ XrtCalendar リソース

各 XRT/field リソースの型とクラスのリストです。一番目のリストは、全ての XrtField のサブクラスに共通なリソース、次のリストは各サブクラスに定義されているリソース、最後のリストは XrtComboBox、XrtSpinBox、XrtCalendar ウィジェットに定義されているリソースです。

F.1 XrtField のサブクラスに共通なリソース

Name	Class	Type
XmNxrtFldAllowNull	XmCBoolean	XmRBoolean
XmNxrtFldCase	XmCXrtFldCase	XmRxrtFldCase
XmNxrtFldCompleteCallback	XmCCallback	XmRCallback
XmNxrtFldConvertCallback	XmCCallback	XmRCallback
XmNxrtFldDefaultBackground	XmCXrtFldBackground	XmRxrtFldPixel
XmNxrtFldDefaultForeground	XmCXrtFldForeground	XmRxrtFldPixel
XmNxrtFldDisplayFormat	XmCXrtFldFormat	XmRxrtFldFormat
XmNxrtFldDisplayHorizScrollBar	XmCXrtFldDispScrollBar	XmRxrtFldDispScrollBar
XmNxrtFldDisplayVertScrollBar	XmCXrtFldDispScrollBar	XmRxrtFldDispScrollBar
XmNxrtFldEditFormat	XmCXrtFldFormat	XmRxrtFldFormat

Name	Class	Type
XmNxrtFldEnterCallback	XmCCallback	XmRCallback
XmNxrtFldErrorBackground	XmCXrtFldBackground	XmRxrtFldPixel
XmNxrtFldErrorCallback	XmCCallback	XmRCallback
XmNxrtFldErrorForeground	XmCXrtFldForeground	XmRxrtFldPixel
XmNxrtFldErrorString	XmCXrtFldMessageString	XmRXmString
XmNxrtFldExitCallback	XmCCallback	XmRCallback
XmNxrtFldInvalidChars	XmCXrtFldInvalidChars	XmRString
XmNxrtFldIsDefault	XmCReadOnly	XmRBoolean
XmNxrtFldIsModified	XmCReadOnly	XmRBoolean
XmNxrtFldIsValid	XmCReadOnly	XmRBoolean
XmNxrtFldLastError	XmCReadOnly	XmRxrtFldError
XmNxrtFldMask	XmCXrtFldMask	XmRString
XmNxrtFldMaskChars	XmCXrtFldMask	XmRString
XmNxrtFldMatchMask	XmCXrtFldMatchMask	XmRxrtFldMatchMask
XmNxrtFldMatchPickList	XmCBoolean	XmRBoolean
XmNxrtFldMaxLineLength	XmCMaxLength	XmRInt
XmNxrtFldMaxLines	XmCMaxLength	XmRInt
XmNxrtFldMenuAlignment	XmCXrtFldMenuAlignment	XmRXrtFldMenuAlignment
XmNxrtFldMenuCallback	XmCCallback	XmRCallback
XmNxrtFldMenuList	XmCReadOnly	XmRWidget
XmNxrtFldOverstrike	XmCBoolean	XmRBoolean
XmNxrtFldPickListIndex	XmCXrtFldPickListIndex	XmRInt
XmNxrtFldPromptString	XmCXrtFldMessageString	XmRXmString
XmNxrtFldPopupType	XmCXrtFldPopupType	XmRXrtFldPopupType
XmNxrtFldRange	XmCXrtFldRange	XmRxrtFldRange
XmNxrtFldSearchPolicy	XmCXrtFldSearchPolicy	XmRxrtFldSearchPolicy
XmNxrtFldSpinCallback	XmCCallback	XmRCallback
XmNxrtFldValidateCallback	XmCCallback	XmRCallback
XmNxrtFldValidChars	XmCXrtFldValidChars	XmRString
XmNxrtFldVisibleItemCount	XmCVisibleItemCount	XmRInt

F.2 XrtStringField リソース

Name	Class	Type
XmNxrtFldDefaultValue	XmCValue	XmRString
XmNxrtFldInternalValue	XmCString	XmRString
XmNxrtFldMax	XmCMaximum	XmRString
XmNxrtFldMin	XmCMinimum	XmRString
XmNxrtFldPickList	XmCString	XmRStringTable

F.3 XrtIntField リソース

Name	Class	Type
XmNxrtFldCounterIncrement	XmCValue	XmRxrtFldInt
XmNxrtFldDefaultValue	XmCValue	XmRxrtFldInt
XmNxrtFldDisplayFormatString	XmCXrtFldDisplayFormatString	XmRString
XmNxrtFldInternalValue	XmCValue	XmRxrtFldInt
XmNxrtFldMax	XmCMaximum	XmRxrtFldInt
XmNxrtFldMin	XmCMinimum	XmRxrtFldInt
XmNxrtFldNegativeFormat	XmCXrtFldNegativeFormat	XmRxrtFldNegativeFormat
XmNxrtFldPickList	XmCXrtFldIntList	XmRxrtFldIntList
XmNxrtFldRange	XmCXrtFldRange	XmRxrtFldRange

F.4 XrtFloatField リソース

Name	Class	Type
XmNxrtFldCounterIncrement	XmCValue	XmRxrtFldFloat
XmNxrtFldDecimalPlaces	XmCValue	XmRxrtFldDecimalPlaces
XmNxrtFldDefaultValue	XmCValue	XmRxrtFldFloat
XmNxrtFldDisplayFormatString	XmCXrtFldDisplayFormatString	XmRString
XmNxrtFldInternalValue	XmCValue	XmRxrtFldFloat
XmNxrtFldLeadingZero	XmCBoolean	XmRBoolean
XmNxrtFldMax	XmCMaximum	XmRxrtFldFloat
XmNxrtFldMin	XmCMinimum	XmRxrtFldFloat
XmNxrtFldNegativeFormat	XmCXrtFldNegativeFormat	XmRxrtFldNegativeFormat
XmNxrtFldPickList	XmCXrtFldFloatList	XmRxrtFldFloatList
XmNxrtFldRange	XmCXrtFldRange	XmRxrtFldRange

F.5 XrtDateField リソース

Name	Class	Type
XmNxrtFldCounterIncrement	XmCValue	XmRxrtFldDateIncrement
XmNxrtFldDefaultValue	XmCValue	XmRxrtFldDate
XmNxrtFldDisplayFormatString	XmCXrtFldDateFormat	XmRString
XmNxrtFldEnterFormatList	XmCXrtFldEnterFormatList	XmRStringTable
XmNxrtFldInternalValue	XmCValue	XmRxrtFldDate
XmNxrtFldInternalValueLong	XmCValue	XmRxrtFldInt
XmNxrtFldMax	XmCMaximum	XmRxrtFldDate
XmNxrtFldMilleniumThreshold	XmCXrtFldMilleniumThreshold	XmRInt
XmNxrtFldMin	XmCMinimum	XmRxrtFldDate
XmNxrtFldPickList	XmCXrtFldDateList	XmRxrtFldDateList

F.6 XrtComboBox リソース

Name	Class	Type
XmNxrtComboArrow	XmCReadOnly	XmRWidget
XmNxrtComboArrowClass	XmCReadOnly	XmRPointer
XmNxrtComboArrowType	XmCXrtFldArrowType	XmRxrtFldArrowType
XmNxrtComboField	XmCReadOnly	XmRWidget
XmNxrtComboFieldClass	XmCReadOnly	XmRPointer
XmNxrtComboFieldType	XmCXrtFldFieldType	XmRxrtFldFieldType
XmNxrtComboSpacing	XmCXrtFldSpacing	XmRHorizontalDimension

F.7 XrtSpinBox リソース

Name	Class	Type
XmNxrtSpinArrowLayout	XmCXrtSpinArrowLayout	XmRxrtSpinArrowLayout
XmNxrtSpinArrowOrientation	XmCXrtSpinArrowOrientation	XmRxrtSpinArrowOrientation
XmNxrtSpinArrowType	XmCXrtFldArrowType	XmRxrtFldArrowType
XmNxrtSpinDecrArrow	XmCReadOnly	XmRWidget
XmNxrtSpinDecrArrowClass	XmCReadOnly	XmRPointer
XmNxrtSpinField	XmCReadOnly	XmRWidget
XmNxrtSpinFieldClass	XmCReadOnly	XmRPointer
XmNxrtSpinFieldName	XmCString	XmRString
XmNxrtSpinFieldType	XmCXrtFldFieldType	XmRxrtFldFieldType
XmNxrtSpinIncrArrow	XmCReadOnly	XmRWidget
XmNxrtSpinIncrArrowClass	XmCReadOnly	XmRPointer
XmNxrtSpinSpacing	XmCXrtFldSpacing	XmRHorizontalDimension
XmNxrtSpinType	XmCXrtSpinType	XmRxrtSpinType

F.8 XrtCalendar リソース

Name	Class	Type
XmNxrtCalendarCellHighlightColor	XmCBackground	XmRxrtFldPixel
XmNxrtCalendarCellHighlightThickness	XmCHighlightThickness	XmRDimension

Name	Class	Type
XmNxrtCalendarCellMargin	XmCMargin	XmRDimension
XmNxrtCalendarCreateMonth	XmCBoolean	XmRBoolean
XmNxrtCalendarCreateTime	XmCBoolean	XmRBoolean
XmNxrtCalendarCreateYear	XmCBoolean	XmRBoolean
XmNxrtCalendarEnabled	XmCBoolean	XmRBoolean
XmNxrtCalendarFirstDayOfWeek	XmCXrtCalendarFirstDayOfWeek	XmRInt
XmNxrtCalendarFullWeekdayNames	XmCBoolean	XmRBoolean
XmNxrtCalendarHeaderBackground	XmCBackground	XmRxrtFldPixel
XmNxrtCalendarHeaderFontList	XmCFontList	XmRFontList
XmNxrtCalendarHeaderForeground	XmCForeground	XmRxrtFldPixel
XmNxrtCalendarHeaderNumChars	XmCXrtCalendarHeaderNumChars	XmRInt
XmNxrtCalendarMarginBottom	XmCMarginBottom	XmRDimension
XmNxrtCalendarMarginLeft	XmCMarginLeft	XmRDimension
XmNxrtCalendarMarginRight	XmCMarginRight	XmRDimension
XmNxrtCalendarMarginTop	XmCMarginTop	XmRDimension
XmNxrtCalendarMaxRollDate	XmCValue	XmRxrtFldDate
XmNxrtCalendarMinRollDate	XmCValue	XmRxrtFldDate
XmNxrtCalendarModifyCallback	XmCCallback	XmRCallback
XmNxrtCalendarSelectedBackground	XmCBackground	XmRxrtFldPixel
XmNxrtCalendarSelectedForeground	XmCForeground	XmRxrtFldPixel
XmNxrtCalendarSpecialDayBackground	XmCBackground	XmRxrtFldPixel
XmNxrtCalendarSpecialDayForeground	XmCForeground	XmRxrtFldPixel
XmNxrtCalendarSpecialDays	XmCXrtCalendarSpecialDays	XmRxrtCalendarSpecialDays
XmNxrtCalendarValue	XmCValue	XmRxrtFldDate
XmNxrtCalendarValueLong	XmCValue	XmRxrtFldInt

G

リソースファイルと GUI ビルダーの使用

構文

リソース・ファイルの例

リソース・ファイルや GUI ビルダーで XRT/field リソースを設定する時に使用される構文のリストです。この章の最後は、サンプルのリソース・ファイルがあります。

XRT/field は、Xt や Motif によって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。

全てのリソースは、コールバック・リソースを除けば、全てリソース・ファイルでセットできます。

G.1 構文

コメント

イクスクラメーション・マーク(!)の後のテキストは、無視されます。

継続

行の最後にバックスラッシュ(\)を指定すると、次の行に継続します。

Boolean

大文字、小文字、大小混合文字で、“`True`”または“`False`”。

列挙型

列挙型のリソースは、列挙された名前に基づく一貫した命名規則を使用します。全てのリソース型では、先頭の `XRTFLD_`または `XRTSPIN_`が取れます。リソース値は、

大文字、小文字、大小混合文字で入力できます。例えば、次の各リソースの指定は、全て、XrtFldArrowLayout に XRTFLD_ARROWS_BEGINNING という値を使用するように指定します。

```
*.XtXrtField.xrtFldArrowLayout: ARROWS_BEGINNING
*.XtXrtField.xrtFldArrowLayout: arrows_beginning
*.XtXrtField.xrtFldArrowLayout: ArRowS-BeginninG
```

日付

XmRxrtFldDate 型のリソースは、次のフォーマットで入力できます。

```
YYYY-MM-DD hh:mm:ss
YYYY-MMM-DD hh:mm:ss
YYYY-MM-DD
YYYY-MMM-DD
YYYY-MMM-DD
hh:mm:ss
```

“Today”(深夜 12 時を始まりとした今日)と“NOW”(現在時刻)というストリングも使用できます。

日付リスト

XmRxrtFldDateList 型のリソースは、カンマで区切られた日付のリスト(上記のフォーマットを使用)で定義されます。

```
*.DateField.xrtFldPickList: now, 1980, 1850-jan-01\
12:00, today, 1955-3-10
```

日付カウンタのインクレメント

XmRxrtFldDateIncrement 型のリソースは、*year-month-date hour:minute:second* の構文を使用して定義されます。例えば、2-0-5 0:0:0 は、2 年と 5 ヶ月インクレメントし、0-0-0 0:30:0 は 30 分インクレメントします。

浮動小数点値

XmRxrtFldFloat 型のリソースは、以下を始め、float フィールドに入力できるストリングならどれでも使用できます。

MAXDOUBLE	最大倍精度浮動小数点数
-MAXDOUBLE	負の最大倍精度浮動小数点数

浮動小数点リスト

XmRxrtFldFloatList 型のリソースは、カンマで区切られた浮動小数点数のリストを使用できます。例:

```
*.FloatField.xrtFldPickList: -34, 1.09, 10000, 2e5, (100)
```

整数

XmRxrtFldInt 型のリソースは、以下を始め、integer フィールドに入力できるストリングならどれでも使用できます。

MAXLONG	*long* 型に定義されている変数に格納できる最大数
-MAXLONG	*long* 型に定義されている変数に格納できる負の最大数
MAXINT	*int* 型に定義されている変数に格納できる最大数
-MAXINT	*int* 型に定義されている変数に格納できる負の最大数

整数リスト

XmRxrtFldIntList 型のリソースは、カンマで区切られた整数のリストを使用します。例:

```
*.IntField.xrtFldPickList: -34, 34 0b11101, 0xA112
```

色の指定

色のリソース(XmRxrtFldPixel 型)は、以下を始め、色に変換できるストリングならどれでも使用できます。

UNCHANGED_COLOR	Core ウィジェット・クラスから継承された XmNbackground または XmNforeground を使用
ERROR_COLOR	無効な値を含むウィジェットには、デフォルトの背景･前景色を使用

G.2 リソース･ファイルの例

下記のリソース･ファイルは、*sampler* というデモ(Appendix I 参照)の動作をコントロールします。

```
! sampler.res - resource file for sampler demo
!! General
*.background: #D1C1AB
*.foreground: black
*.XmToggleButton.selectColor: Yellow
*.fontList: -*-helvetica-bold-r-normal--14-140-*-*-*-*-*-*
*.field.fontList: 9x15
*.XmXrtCurrencyField.background: #FFF7D4
*.XmXrtCurrencyField.foreground: black
*.XmXrtDateField.background: #FFF7D4
*.XmXrtDateField.foreground: black
*.XmXrtFloatField.background: #FFF7D4
*.XmXrtFloatField.foreground: black
*.XmXrtIntField.background: #FFF7D4
*.XmXrtIntField.foreground: black
*.XmXrtStringField.background: #FFF7D4
*.XmXrtStringField.foreground: black
*.XmList.foreground: black
*.XmList.background: #FFF7D4
*.selections.fontList: -*-new century schoolbook-medium-r-normal-*-140-*
! Don't display non-message panedWindow sashes
*.sashWidth: 0
*.sashHeight: 0
*.sashShadowThickness: 0
*.sash.traversalOn: false
*.msg_pane.sashWidth: 10
*.msg_pane.sashHeight: 10
*.msg_pane.sashShadowThickness: 5
!*.xrtFldDefaultBackground: yellow
!*.xrtFldDefaultForeground: black
*.XmLabel.alignment: alignment_end
*.prompt_label.labelString: \n \n \n
*.prompt_label.alignment: alignment_center
*.prompt_label.recomputeSize: false
*.prompt_frame.shadowThickness: 5
*.error_label.labelString: \n \n
*.error_label.foreground: red
*.error_label.alignment: alignment_center
*.error_label.recomputeSize: false
*.error_frame.shadowThickness: 5
*.Help.title: Field Help
*.Help.XmLabel.alignment: alignment_beginning
*.Help.*.labelString: \n
*.Help.*.width: 500
*.Help.*.height: 100
*.Help.x: 500
*.Help.y: 500
*.Help.*.fontList:
-*-new century schoolbook-medium-r-normal-*-140-*=roman,\
```

```
-*-new century schoolbook-medium-i-normal-*-140-*=italic,\
-*-new century schoolbook-bold-r-normal-*-140-*=bold,\
-*-symbol-medium-r-normal-*-140-*=symbol
*.aboutXRTfield.fontList:
-*-new century schoolbook-medium-r-normal-*-140-*=roman,\
-*-new century schoolbook-medium-i-normal-*-140-*=italic,\
-*-new century schoolbook-bold-r-normal-*-140-*=bold,\
-*-symbol-medium-r-normal-*-140-*=symbol
!! Edit and display formats
*.Edit_Format.labelString: Edit Format:
*.Display_Format.labelString: Display Format:
*.format_bb.*.xrtFldDefaultBackground: unchanged_color
*.format_bb.*.xrtFldMatchPickList: false
*.format_bb.*.columns: 25
!! Fields
*.File_field.value: sample
*.File_field.xrtFldComplete: true
*.File_field.translations: #override\n\
Ctrl <Key>c: FileComplete()
*.File_field.xrtFldInvalidChars: [ ]
*.File_field.xrtFldErrorString: Spaces and brackets are not allowed
*.File_field.xrtFldPromptString: Press @B@CTRL-c@P@ to complete the filename \
(add a @B@.c@P@ suffix)
*.File_field.fontList: 9x15
*.Phone1_field.xrtFldMask: (@@@) @@@-@@@@ Ext. @@@@
*.Phone1_field.xrtFldAllowNull: false
*.Phone1_field.xrtFldInternalValue: 4165551212
*.Phone1_field.xrtFldMatchMask: match_partial_l_to_r
*.Phone1_field.xrtFldErrorString: Only numbers may be entered
*.Phone1_field.xrtFldPromptString: Enter a phone number, with optional
extension
*.Phone1_field.fontList: 9x15
*.Phone1_field.columns: 25
*.Phone2_field.*.xrtFldMask: (@@@) @@@-@@@@
*.Phone2_field.*.fontList: 9x15
*.Phone2_field.*.xrtFldInternalValue: 4165551212
*.Phone2_field.*.xrtFldErrorString: Only numbers may be entered
*.Phone2_field.*.xrtFldPromptString: Select an international or local phone number\
mask from the picklist.
*.Phone2_Mask_combo.*.xrtFldPickList: Local 1,Local 2,International
*.Phone2_Mask_combo.*.xrtFldPromptString: Select an international or local
phone\
number mask from the picklist.
*.Phone2_Mask_combo.*.fontList: 9x15
*.Float_field.xrtFldErrorString: Only numbers may be entered
*.Float_field.xrtFldPromptString: Enter any number
*.Float_field.fontList: 9x15
*.Date_field.xrtFldMask: @@@@-@@-@@ @@:@@:@@
*.Date_field.value: yyyy-mm-dd hh:mm:ss
*.Date_field.xrtFldErrorString: Only numbers may be entered
*.Date_field.xrtFldPromptString: Enter a date/time
*.Date_field.fontList: 9x15
*.ZipCode.labelString: Zip Code:
*.ZipCode_field.xrtFldMask: @@@@@-@@@@
```

```
*.ZipCode_field.xrtFldDefaultValue: 000000000
*.ZipCode_field.xrtFldMatchMask: match_partial_l_to_r
*.ZipCode_field.xrtFldErrorString: Only 5 or 9 digits may be entered
*.ZipCode_field.xrtFldPromptString: Enter a 5 or 9-digit number
*.ZipCode_field.fontList: 9x15
*.PostalCode.labelString: Postal Code:
*.PostalCode_field.xrtFldMask: U@U @U@
*.PostalCode_field.xrtFldDefaultValue: A1A 1A1
*.PostalCode_field.xrtFldErrorString: Enter a letter or a number
*.PostalCode_field.xrtFldPromptString: Enter a 6-character code
*.PostalCode_field.fontList: 9x15
*.Name_field.xrtFldPromptString: A Convert callback uppercases the first \
letter\nin each word
*.Name_field.xrtFldDisplayHorizScrollBar: dispsb_as_needed
*.Name_field.fontList: 9x15
*.Name_field.height: 40
*.Password_field.xrtFldAllowNull: false
*.Password_field.xrtFldErrorString: A password must be entered
*.Password_field.xrtFldPromptString: Enter any string
*.Password_field.xrtFldDisplayHorizScrollBar: dispsb_as_needed
*.Password_field.fontList: 9x15
*.Comment1_field.editMode: multi_line_edit
*.Comment1_field.xrtFldMaxLineLength: 10
*.Comment1_field.rows: 2
*.Comment1_field.xrtFldMaxLines: 3
*.Comment1_field.value: A\nsample\ncomment
*.Comment1_field.xrtFldPromptString: Only 3 lines, max 10 characters/line,\n\
can be entered
*.Comment1_field.fontList: 9x15
!! Combo boxes
*.Date_combo.*.xrtFldMatchPickList: false
*.Date_combo.*.xrtFldPickList: now,1980,1850-jan-01 12:00,today,1955-mar-10 00,\
1988-jun-01,2001-02-27 15:00
*.Date_combo.*.xrtFldErrorString: A valid date must be entered
*.Salaries_combo.*.xrtFldPickList: 5000,1000,15000,20000
*.Salaries_combo.*.xrtFldMatchPickList: false
*.Salaries_combo.*.xrtFldPromptString: Try setting the Edit and Display Formats for\
this field.
*.Salaries_combo.*.xrtFldErrorString: A valid currency must be entered.
*.Names_combo.*.xrtFldPickList: John Jones,Harry Smith,Jean Jones,Mary Martin,\
Harry Hayes
*.Int_combo.*.xrtFldDisplayFormatString: Value: %7d
*.Int_combo.*.xrtFldPickList: -34,100,10000
*.Int_combo.*.xrtFldPromptString: This field uses an
XmNxrtFldDisplayFormatString \
\n\to customize the appearance.\\nThe entered value must be one of the picklist\
items.
*.Int_combo.*.xrtFldErrorString: A valid integer must be entered which \
matches one of the picklist items.
*.Float_combo.*.xrtFldPickList: -34,1.09,10000,2e5
*.Float_combo.*.xrtFldPromptString: Try setting the Edit and Display Formats for \
this field.
```

```
!! Spin boxes
*.Zodiac_spin.*.xrtFldPickList:
Aries,Taurus,Gemini,Cancer,Leo,Virgo,Libra,\
Scorpio,Sagittarius,Capricorn,Aquarius,Pisces
*.Zodiac_spin.xrtSpinArrowLayout: arrows_beginning
*.Titles_spin.*.xrtFldPickList: Mr.,Mrs.,Ms.,Miss,Dr.
*.Primes_spin.*.xrtFldPickList: 1,3,5,7,11,13,17,19,23,29,31
*.Primes_spin.xrtSpinArrowLayout: arrows_end
!! Counter boxes
*.Integer_counter.xrtSpinArrowLayout: arrows_end
*.Integer_counter.xrtSpinArrowOrientation: vertical
*.Integer_counter.*.xrtFldInternalValue: 0
*.Integer_counter.*.xrtFldMin: -10
*.Integer_counter.*.xrtFldMax: 10
*.Integer_counter.*.xrtFldPromptString: Try setting the Edit and Display Formats\
for this field.
*.Integer2_counter.xrtSpinArrowLayout: arrows_end
*.Integer2_counter.xrtSpinArrowOrientation: vertical
*.Integer2_counter.*.xrtFldMin: 0
*.Integer2_counter.*.xrtFldCounterIncrement: 10
*.Integer2_counter.*.xrtFldDisplayFormat: format_hex
*.Integer2_counter.*.xrtFldInternalValue: 1
*.Integer2_counter.*.xrtFldPromptString: Try setting the Edit and Display Formats\
for this field.
*.Float_counter.*.xrtFldDefaultValue: 1
*.Float_counter.xrtSpinArrowLayout: arrows_end
*.Float_counter.xrtSpinArrowOrientation: vertical
*.Float_counter.*.xrtFldPromptString: This demonstrates using a @B@Spin \
Callback@P@\nto change the value by a power of 10.\\nTry setting the Edit and \
Display Formats for this field.
*.Date_counter.xrtSpinArrowLayout: arrows_end
*.Date_counter.xrtSpinType: type_counter
*.Date_counter.*.xrtFldInternalValue: today
*.Date_counter.day.columns: 2
*.Date_counter.day.xrtFldCounterIncrement: 0-0-1
*.Date_counter.day.xrtFldDisplayFormatString: DD
*.Date_counter.day.xrtFldEnterFormatList: DD
*.Date_counter.day.xrtFldPromptString: Pressing the arrow buttons changes the\
@B@day@P@
*.Date_counter.month.columns: 10
*.Date_counter.month.xrtFldCounterIncrement: 0-1-0
*.Date_counter.month.xrtFldDisplayFormatString: MMMM
*.Date_counter.month.xrtFldEnterFormatList: MMMM
*.Date_counter.month.xrtFldPromptString: Pressing the arrow buttons changes the\
@B@month@P@
*.Date_counter.wday.columns: 10
*.Date_counter.wday.xrtFldCounterIncrement: 0-0-1
*.Date_counter.wday.xrtFldDisplayFormatString: WW
*.Date_counter.wday.xrtFldEnterFormatList: WW
*.Date_counter.wday.xrtFldPromptString: Pressing the arrow buttons changes the\
@B@weekday@P@
*.Date_counter.year.columns: 4
*.Date_counter.year.xrtFldCounterIncrement: 1-0-0
*.Date_counter.year.xrtFldDisplayFormatString: YYYY
*.Date_counter.year.xrtFldEnterFormatList: YYYY
```

```
*.Date_counter.year.xrtFldPromptString: Pressing the arrow buttons changes the\
@B@year@P@
*.Date_counter.*.fontList: 9x15

!! Currencies
*.currency_bb.*.xrtFldPickList: 0,1000,1000000,-100
*.currency_bb.*.xrtFldMatchPickList: false
*.currency_bb.*.xrtFldPromptString: This illustrates the use of\
@I@XrtFldSetDefaultLocaleStruct()@P@\\nto set the locale for individual fields.\
\nTry setting the Negative Format and number of decimal places for this field.
*.Negative_Format.labelString: Negative Format:
*.Decimal_Places.labelString: Decimal Places:

*.UK_combo.*.xrtFldPickList: 0,1000.00,1000000,-100
*.UK_Intl.labelString: UK Intl:
*.UK_Intl_combo.*.xrtFldPickList: 0,1000.50,1000000,-100
*.UK_Intl_combo.*.xrtFldDisplayFormat: format_currency_intl

*.US_combo.*.xrtFldPickList: 0,1000.50,1000000,-100
*.US_Intl_combo.*.xrtFldPickList: 0,1000.50,1000000,-100
*.US_Intl_combo.*.xrtFldDisplayFormat: format_currency_intl
*.US_Intl.labelString: US Intl:

*.Japan_combo.*.xrtFldPickList: 0,1000.50,1000000,-100
*.Japan_Intl_combo.*.xrtFldPickList: 0,1000,1000000,-100
*.Japan_Intl_combo.*.xrtFldDisplayFormat: format_currency_intl
*.Japan_Intl.labelString: Japan Intl:

*.Swiss_Intl_combo.*.xrtFldDisplayFormat: format_currency_intl
*.Swiss_combo.*.xrtFldPickList: 0,1000.50,1000000,-100
*.Swiss_Intl_combo.*.xrtFldPickList: 0,1000.50,1000000,-100
*.Swiss_Intl.labelString: Swiss Intl:

*.Decimal_Places_counter.xrtSpinArrowLayout: arrows_end
*.Decimal_Places_counter.xrtSpinArrowOrientation: vertical
*.Decimal_Places_counter.*.xrtFldMin: 0
*.Decimal_Places_counter.*.xrtFldMax: 10

! File Menu
*.File_cascade.mnemonic: F
*.Exit_button.mnemonic: X
*.Exit_button.accelerator: Ctrl<Key>X
*.Exit_button.acceleratorText: Ctrl+X

! Help Menu
*.Help_cascade.mnemonic: H

! Date Popup
*.XmXrtCalendar.xrtCalendarCreateTime: true
*.XmXrtCalendar.fontList: 9x15
*.XmXrtCalendar.xrtCalendarHeaderFontList: -*-helvetica-bold-r-normal--\
14-140-*-*-*-*-*-*
*.XmXrtCalendar.xrtCalendarSpecialDays: (SPECIAL_DAY_BY_DATE 25 0 0 11 \
XRTCALENDAR_ALL) (SPECIAL_DAY_BY_DATE 1 0 0 0 XRTCALENDAR_ALL) \
(SPECIAL_DAY_BY_DAY_OF_WEEK 0 0 XRTCALENDAR_ALL XRTCALENDAR_ALL XRTCALENDAR_ALL)
*.XmXrtCalendar.xrtCalendarSpecialDayBackground: #FFF7D4
```

H

UIL 引数

XrtField のサブクラスに共通なリソース ■ *XrtIntField のリソース*

XrtFloatField のリソース ■ *XrtDateField のリソース*

XrtComboBox のリソース ■ *XrtSpinBox のリソース*

XrtCalendar リソース ■ *Reasons*

XRT/field の UIL 引数とそのデータ型のアルファベット順のリストです。二番めの表は、XRT/field の UIL の reason です。UIL に使用については、4.17 を参照してください。

H.1 XrtField のサブクラスに共通なリソース

UIL Argument Name	Argument Type
XmNxrtFldAllowNull	boolean
XmNxrtFldCase	integer
XmNxrtFldDisplayFormat	integer
XmNxrtFldDisplayFormatString	string
XmNxrtFldDisplayHorizScrollBar	integer
XmNxrtFldDisplayVertScrollBar	integer
XmNxrtFldEditFormat	integer
XmNxrtFldInvalidChars	string
XmNxrtFldIsDefault	boolean
XmNxrtFldIsModified	boolean
XmNxrtFldIsValid	boolean
XmNxrtFldLastError	integer

UIL Argument Name	Argument Type
XmNxrtFldMask	string
XmNxrtFldMaskChars	string
XmNxrtFldMatchMask	integer
XmNxrtFldMatchPickList	boolean
XmNxrtFldMaxLineLength	integer
XmNxrtFldOverstrike	boolean
XmNxrtFldPickListIndex	integer
XmNxrtFldRange	integer
XmNxrtFldValidChars	string
XmNxrtFldVisibleItemCount	integer

H.2 XrtIntField のリソース

UIL Argument Name	Argument Type
XmNxrtFldCounterIncrement	integer

H.3 XrtFloatField のリソース

UIL Argument Name	Argument Type
XmNxrtFldDecimalPlaces	integer
XmNxrtFldLeadingZero	boolean
XmNxrtFldNegativeFormat	integer

H.4 XrtDateField のリソース

UIL Argument Name	Argument Type
XmNxrtFldInternalValueLong	integer

H.5 XrtComboBox のリソース

UIL Argument Name	Argument Type
XmNxrtComboFieldType	integer

H.6 XrtSpinBox のリソース

UIL Argument Name	Argument Type
XmNxrtSpinArrowLayout	integer
XmNxrtSpinFieldName	string
XmNxrtSpinFieldType	integer
XmNxrtSpinType	integer

H.7 XrtCalendar Resources

UIL Argument Name	Argument Type
XmNxrtCalendarCellMargin	integer
XmNxrtCalendarCellHighlightColor	color
XmNxrtCalendarCellHighlightThick	integer
XmNxrtCalendarCreateMonth	boolean
XmNxrtCalendarCreateTime	boolean
XmNxrtCalendarCreateYear	boolean
XmNxrtCalendarEnabled	boolean
XmNxrtCalendarFirstDayOfWeek	integer
XmNxrtCalendarFullWeekdayNames	boolean
XmNxrtCalendarHeaderNumChars	integer
XmNxrtCalendarHeaderBackground	color
XmNxrtCalendarHeaderFontList	font
XmNxrtCalendarHeaderForeground	color
XmNxrtCalendarMarginBottom	integer
XmNxrtCalendarMarginLeft	integer
XmNxrtCalendarMarginRight	integer
XmNxrtCalendarMarginTop	integer

UIL Argument Name	Argument Type
XmNxrtCalendarSpecialDayBackground	color
XmNxrtCalendarSpecialDayForeground	color
XmNxrtCalendarSelectedBackground	color
XmNxrtCalendarSelectedForeground	color
XmNxrtCalendarValueLong	integer

H.8 Reasons

Reasons
XmNxrtFldCompleteCallback
XmNxrtFldConvertCallback
XmNxrtFldEnterCallback
XmNxrtFldErrorCallback
XmNxrtFldExitCallback
XmNxrtFldMenuCallback
XmNxrtFldSpinCallback
XmNxrtFldValidateCallback

I

サンプル･コード

例

デモ

XRT/field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。これらの例やデモから、XRT/fieldに関する多くのプログラミング･タスクを習得することができます。

I.1 例

これらの例のプログラムは、*$XRTHOME/src/field/examples* にあります。

calendar.c	簡単な XrtCalendar ウィジェットを表示します。
combo.c	簡単な XrtComboBox ウィジェットを表示します。
counter.c	0 から 10 をカウントする XrtSpinBox を表示します。
datespin.c	四つのカウンタ(一週間の日、一ヶ月の日、月、年)を子として XrtSpinBox を作成します。これらのカウンタの内、どれかがインクレメントまたはデクレメントされると、それに従って、他のカウンタもアップデートされます。(例えば、一ヶ月の日がインクレメントされれば、一週間の日も自動的に増えていきます。)
datespin2.c	スピンするサブ･フィールドを持った、*datespin.c* よりもう少し複雑な例です。
field.c	XRT/field の五つのサブクラスの例を表示します。

password.c ユーザ・パスワードの入力にフィールドを使用します。ユーザによって入力されたパスワードは、*stdout* に書かれます。

phone.c 電話番号のフィールドを表示します。マスクを使用して、地域コードを表す括弧とダッシュが自動的に挿入されます。

simple.c “1 XRT/field の簡単なプログラムの開発”で使用された、プログラムです。

spin.c 簡単な XrtSpinBox ウィジェットを表示します。

I.2 デモ

デモ・アプリケーションは、*$XRTHOME/src/field/demos* にあります。各デモ毎にディレクトリがあります。

form XRT/field を使用して、従業員データの入力フォームを作成するデモです。combo box は従業員の役職、マスクを使用した date フィールドは誕生日と入社日、currency フィールドは給与というように、データによって異なる XRT/field ウィジェットを使い分けています。

locale このサブ・ディレクトリには、選択されたロケールまたはカレント・ロケールの情報を表示する、*printlocale* プログラムがあります。

sampler XRT/field の全ての機能を使用したデモです。(radio ボタンを使用してアクセスできる)別のウィンドウでは、複数のフィールド、combo box、spin box、カウンタ box 等を表示しています。

XRT/field が入力フォーカスを得ると、アプリケーションの下部にあるメッセージ・ウィンドウが、そのウィジェットの機能を表示します。

table_entry XRT/field と XRT/table を一緒に使用したデモです。テーブルでは、列の編集に幾つかのフィールドのサブクラスを使用しています。(例: “Phone Number”の列では integer フィールドを使用)

他の XRT 製品を使用しているので、このデモは、*$XRTHOME/src/common_demos* にあります。

索引

C

C++, 12, 87
Combo Box, 14, 65
  pick list, 66
  テキスト・フィールド, 68
  デフォルトの設定, 65
  ポップアップ・メニュー, 68
  矢印, 69

E

enter field action, 24, 25

K

keystroke action, 24, 25

M

Motif との統合, 19

P

paste action, 25
pick list, 47, 77, 79
  インデックス, 81
  検索法, 66
programmatic change action, 24, 27

S

Spin Box, 15, 70
  データ型, 71
  デフォルトの設定, 70
  矢印, 70, 72

U

UIL, 87, 167
user exit action, 24, 26

X

XmCrateXrtComboBox(), 129
XmCrateXrtIntField(), 130
XmCreateXrtCalendar(), 129
XmCreateXrtCurrencyField(), 130
XmCreateXrtDateField(), 130
XmCreateXrtFloatField(), 130
XmCreateXrtSpinBox(), 131
XmCreateXrtStringField(), 131
XmIsXrtCalendar(), 131
XmIsXrtComboBox(), 131
XmIsXrtCurrencyField(), 131
XmIsXrtDateField(), 132
XmIsXrtField(), 132
XmIsXrtFloatField(), 132
XmIsXrtIntField(), 132
XmIsXrtSpinBox(), 132
XmIsXrtStringField(), 132
XmNactivateCallback, 117
XmNautoShowCursorPosition, 117
XmNblinkRate, 117
XmNcolumns, 117
XmNcursorPosition, 117
XmNcursorPositionVisible, 117
XmNeditable, 117
XmNeditMode, 117
XmNfocusCallback, 117
XmNfontList, 117
XmNgainPrimaryCallbak, 117
XmNlosePrimaryCallbak, 117
XmNlosingFocusCallback, 117
XmNmarginHeight, 118
XmNmarginWidth, 118
XmNmaxLength, 118
XmNmodifyVerifyCallback, 118
XmNmotionVerifyCallback, 118
XmNpendingDelete, 118
XmNresizeHeight, 118
XmNresizeWidth, 118
XmNrows, 118
XmNselectionArray, 118
XmNselectionArryCount, 118

XmNselectThreshold, 118
XmNsource, 118
XmNtopCharacter, 118
XmNvalue, 23, 118
XmNvalueChangedCallback, 119
XmNvalueWcs, 119
XmNverifyBell, 119
XmNwordWrap, 119
XmNxrtCalendarCellHighlightThickness, 73
XmNxrtCalendarCellMargin, 125
XmNxrtCalendarCreateMonth, 75, 125
XmNxrtCalendarCreateTime, 75, 125
XmNxrtCalendarCreateYear, 75, 125
XmNxrtCalendarEnabled, 125
XmNxrtCalendarFirstDayofWeek, 125
XmNxrtCalendarFirstDayOfWeek, 72
XmNxrtCalendarFullDayNames, 72, 125
XmNxrtCalendarHeaderBackground, 125
XmNxrtCalendarHeaderFontList, 73, 125
XmNxrtCalendarHeaderForeground, 126
XmNxrtCalendarHeaderNumChars, 72, 125
XmNxrtCalendarMarginBottom, 126
XmNxrtCalendarMarginLeft, 126
XmNxrtCalendarMarginRight, 126
XmNxrtCalendarMarginTop, 126
XmNxrtCalendarMaxRollDate, 126
XmNxrtCalendarMinRollDate, 126
XmNxrtCalendarModifyCallback, 126
XmNxrtCalendarSelectedBackground, 127
XmNxrtCalendarSelectedForeground, 127
XmNxrtCalendarSpecialDayBackground, 74, 127
XmNxrtCalendarSpecialDayForeground, 74, 127
XmNxrtCalendarSpecialDays, 127
XmNxrtCalendarValue, 127
XmNxrtCalendarValueLong, 127
XmNxrtComboArrow, 66, 121
XmNxrtComboArrowClass, 66, 121
XmNxrtComboArrowType, 69, 121
XmNxrtComboField, 66, 121
XmNxrtComboFieldClass, 121
XmNxrtComboFieldType, 15, 121
XmNxrtComboSpacing, 69, 121
XmNxrtFldAlignment, 68
XmNxrtFldAllowNull, 60, 95
XmNxrtFldAutoComplete, 95
XmNxrtFldCalendar, 75, 95
XmNxrtFldCalendarCallback, 75, 95
XmNxrtFldCase, 51, 96
XmNxrtFldCompleteCallback, 52, 96
XmNxrtFldConvertCallback, 51, 97
XmNxrtFldCounterIncrement, 98
XmNxrtFldCreatePopups, 98
XmNxrtFldDecimalPlaces, 37, 98
XmNxrtFldDefaultBackground, 57, 60, 98
XmNxrtFldDefaultForeground, 57, 60, 98
XmNxrtFldDefaultValue, 60, 99
XmNxrtFldDisplayFormat, 38, 53, 100
XmNxrtFldDisplayFormatString, 39, 54, 55, 100
XmNxrtFldDisplayHorizScrollBar, 79, 84, 101
XmNxrtFldDisplayVertScrollBar, 79, 84, 102
XmNxrtFldEditFormat, 36, 102
XmNxrtFldEnterCallback, 61, 103
XmNxrtFldEnterFormatList, 37, 38, 39, 103
XmNxrtFldErrorBackground, 57, 104
XmNxrtFldErrorCallback, 49, 104
XmNxrtFldErrorForeground, 57, 106
XmNxrtFldErrorString, 106
XmNxrtFldExitCallback, 56, 107
XmNxrtFldInternalValue, 23, 41, 107
XmNxrtFldInternalValueLong, 23, 108
XmNxrtFldInvalidChars, 41, 108
XmNxrtFldIsDefault, 60, 108
XmNxrtFldIsModified, 108
XmNxrtFldIsValid, 40, 108
XmNxrtFldLastError, 108
XmNxrtFldLeadingZero, 108
XmNxrtFldMask, 42, 47, 79, 108
XmNxrtFldMaskChars, 85, 109
XmNxrtFldMatchMask, 44, 109
XmNxrtFldMatchPickList, 110
XmNxrtFldMax, 46, 110
XmNxrtFldMaxLineLenght, 41
XmNxrtFldMaxLineLength, 110
XmNxrtFldMaxLines, 41, 110
XmNxrtFldMenuAlignment, 110
XmNxrtFldMenuCallback, 78, 111
XmNxrtFldMenuList, 67, 111
XmNxrtFldMilleniumThreshold, 111
XmNxrtFldMin, 46, 112
XmNxrtFldNegativeFormat, 112
XmNxrtFldOverstrike, 44, 79, 112
XmNxrtFldPickList, 14, 46, 112
XmNxrtFldPickListIndex, 112
XmNxrtFldPopupType, 75, 113
XmNxrtFldPromptString, 61, 113
XmNxrtFldRange, 113
XmNxrtFldSearchPolicy, 66, 113
XmNxrtFldSpinCallback, 80, 114
XmNxrtFldSpinPolicy, 114
XmNxrtFldValidateCallback, 48, 115
XmNxrtFldValidChars, 41, 116
XmNxrtFldVisibleItemCount, 78, 116
XmNxrtPickListIndex, 81

XmNxrtSpinArrowLayout, 70, 122
XmNxrtSpinArrowOrientation, 70, 122
XmNxrtSpinArrowType, 72, 122
XmNxrtSpinDecrArrow, 122
XmNxrtSpinDecrArrowClass, 71, 122
XmNxrtSpinField, 122
XmNxrtSpinFieldClass, 122
XmNxrtSpinFieldName, 123
XmNxrtSpinFieldType, 123
XmNxrtSpinIncrArrow, 123
XmNxrtSpinIncrArrowClass, 71, 123
XmNxrtSpinSpacing, 123
XmNxrtSpinType, 71, 80, 123
XmText, 21, 85
XmTextField, 21
XRT/field
  read-only, 38
  XmText のコールバック, 85
  XRT/table との使用, 61
  空のフィールド, 60
  デフォルト値, 59
  複数のフィールド, 81
XRT/field 2.0
  新しい機能, 2
xrt_auth, 11
XrtCalendar, 17, 72, 75
XrtCalendarAddSpecialDay(), 132
XrtCalendarDeleteSpecialDay(), 133
XrtCalendarDupSpecialDays(), 133
XrtCalendarFreeSpecialDays(), 133
XrtCalendarModifyCallbackStruct, 145
XrtCalendarReason, 145
XrtCalendarSpecialDay, 145
XrtCalendarSpecialDayType, 146
XrtComboBox, 14, 65
XrtField
  サブクラス, 21
XrtFldAction, 146
XrtFldArrowType, 146
XrtFldCalendarCallbackStruct, 146
XrtFldCase, 146
XrtFldCheckFields(), 81, 133
XrtFldCompleteCallbackStruct, 146
XrtFldConvertCallbackStruct, 147
XrtFldCvtLongToTime(), 133
XrtFldCvtStringToDate(), 134
XrtFldCvtStringToTime(), 82, 134
XrtFldCvtTimeToLong(), 134
XrtFldDirection, 147
XrtFldDispScrollBar, 147
XrtFldDoubleToArgVal(), 135
XrtFldDTconv, 147
XrtFldEnterCallbackStruct, 147
XrtFldError, 148
XrtFldErrorCallbackStruct, 148
XrtFldExitCallbackStruct, 148
XrtFldFieldData, 148
XrtFldFieldDataFree(), 135
XrtFldFieldType, 149
XrtFldFormat, 149
XrtFldGetCurrencyLocale(), 135
XrtFldGetDateLocale(), 135
XrtFldGetDisplayValue(), 136
XrtFldGetField(), 136
XrtFldGetNumericLocale(), 135
XrtFldGetPassword(), 136
XrtFldMatchMask, 149
XrtFldMenuAlignment, 149
XrtFldMenuCallbackStruct, 149
XrtFldMrmInitialize(), 136
XrtFldNegFormat, 150
XrtFldPopupType, 150
XrtFldRange, 150
XrtFldReason, 150
XrtFldSearchPolicy, 150
XrtFldSetDefaultLocaleStruct(), 137
XrtFldSetReverseConverters(), 137
XrtFldSpinCallbackStruct, 151
XrtFldSpinPolicy, 151
XrtFldValidateCallbackStruct, 151
XrtFldValidateValue(), 137
XRTHOME, 9
XrtSpinArrowLayout, 151
XrtSpinBox, 15, 70
XrtSpinType, 151
XrtTblGetXrtFieldByType(), 63

あ

アクション・ルーチン
  Calendar(), 76, 142
  Cancel(), 77, 142
  Complete(), 77, 142
  Menu(), 78, 142
  NoAction(), 142
  Overstrike(), 79, 142
  Scrollbars(), 79, 143
  Spin(), 80, 143
アプリケーションの許可, 11
色, 29, 57
  pixel 型, 57
  エラー, 57

デフォルト値, 57, 60
上書きモード, 44

か

カレンダ, 17, 72
特別な日, 73
環境変数, 9
検証, 12, 24, 40
値の範囲, 46
値のリスト(pick list), 46
行数, 41
行の長さ, 41
検証マスク, 42
コールバック, 48
データ型, 46
複数のフィールド, 81
有効･無効文字, 41
国際化, 88
コールバック
exit, 27, 56
spin, 80
XmText, 85
エラー, 49
カレンダ, 76
検証, 48
入力, 61
変換, 51
補完, 52, 77
メニュー, 78
コンパイル, 11

さ

スクロールバー, 84

た

デフォルト値, 59
色, 57

な

入力プロンプト, 61

は

日付
フォーマット, 38
変換, 82
日付エディタ, 75
表示フォーマット, 24, 38, 39, 53
date フィールド, 55
カスタマイズ, 54
複数行のテキスト, 58
複数のフィールド, 81
変換, 24, 51
コールバック, 51
編集フォーマット, 24, 36, 40
編集プロセス, 24
まとめ, 28
補完, 24, 52
コールバック, 53
自動マッチング, 64

ま

マスク, 79
literal, 43
上書きモード, 44
検証, 42
視覚的なキュー, 45
部分的な適合, 44
文字の定義, 85
無効フィールド, 49
エラー･コールバック, 49
内部値, 41

や

ユーザ･インタラクション
デフォルト, 21
トランスレーション, 141

ら

リソース
一覧, 31, 93
型とクラス, 153
浮動小数点, 29
変換プログラム, 87

ポインタ, 29
リソース・ファイル, 30, 159
ロケール, 88