IBM SPSS Modeler 18.1.1
CLEF 開発者ガイド

注記

本書および本書で紹介する製品をご使用になる前に、 351 ページの『特記事項』に記載されている情報をお読みください。

本書は、IBM SPSS Modeler バージョン 18 リリース 1 モディフィケーション 1 および新しい版で明記されない限り、以降のすべてのリリースおよびモディフィケーションに適用されます。

お客様の環境によっては、資料中の円記号がバックスラッシュと表示されたり、バックスラッシュが円記号と表示されたりする場合があります。

原典： IBM SPSS Modeler 18.1.1 CLEF Developer's Guide

発行： 日本アイ・ビー・エム株式会社

担当： トランスレーション・サービス・センター

目次

前書き

IBM® SPSS® Modeler は、IBM Corp. が開発した企業強化用のデータ・マイニング・ワークベンチです。SPSS Modeler を使用すると、企業はデータを詳しく調べることにより顧客および一般市民とのリレーションシップを強化することができます。企業は、SPSS Modeler を使用して得られた情報に基づいて利益をもたらす顧客を獲得し、抱き合わせ販売の機会を見つけ、新規顧客を引き付け、不正を発見し、リスクを減少させ、政府機関へのサービスの提供を改善することができます。

SPSS Modeler の視覚的インターフェースを使用すると、特定ビジネスの専門知識を適用し、より強力な予測モデルを実現し、解決までの時間を短縮します。SPSS Modeler では、予測、分類、セグメント化、および関連性検出アルゴリズムなど、さまざまなモデリング手法を提供しています。モデルを作成した後は、IBM SPSS Modeler Solution Publisher により、企業全体の意思決定者やデータベースにモデルを配布することが可能になります。

IBM Business Analytics について

IBM Business Analytics ソフトウェアは、意思決定者がビジネス・パフォーマンスの改善のために使用可能な完全で整合性があり、正確な情報を提供します。ビジネス・インテリジェンス、予測分析、財務実績および戦略管理、分析アプリケーション の包括的なポートフォリオを利用することによって、現在の実績を明確、迅速に理解し、将来の結果を予測することができます。豊富な業界のソリューション、証明された実践法、それに専門家によるサービスを組み合せることにより、あらゆる規模の組織が、最高の生産性を実現し、信頼できる意志決定を自動化して、よりよい結果を得ることができます。

このポートフォリオの一部として、IBM SPSS Predictive Analytics ソフトウェアを使用する組織は、将来のイベントを予測し、その洞察に基づいて積極的に行動し、より優れた業績を実現することができます。全世界の企業、政府、学術分野のお客様が IBM SPSS の技術を活用し、不正行為を減少させ、リスクを軽減させながら、顧客の獲得、保持、成長において、競争優位を高めることができます。IBM SPSS ソフトウェアを日々の業務に取り入れることによって、組織は業務目標を達成し、大きな競争的優位を獲得することができるよう、意思決定を方向付け、自動化することができるようになります。詳細な情報、または営業担当者へのお問い合わせ方法については、http://www.ibm.com/spss を参照してください。

技術サポート

お客様はテクニカル・サポートをご利用いただけます。IBM Corp. 製品の使用方法、または対応するハードウェア環境へのインストールについてサポートが必要な場合は、テクニカル・サポートにご連絡ください。テクニカル・サポートの詳細は、IBM Corp. Web ページ http://www.ibm.com/support を参照してください。支援を要請される場合は、ご本人、組織、サポートの同意を確認できるものをご用意ください。

第 1 章 概要

CLEF の概要

Component-Level Extension Framework (CLEF) は、ユーザーが提供した拡張を IBM SPSS Modeler の標準機能に追加できるメカニズムです。通常、拡張には、データ処理ルーチンやモデル作成アルゴリズムなどの共有ライブラリーが含まれていて、メニューの新規エントリーから、またはノード・パレットの新規ノードとして IBM SPSS Modeler に追加されて使用可能になります。

これを可能にするために、IBM SPSS Modeler はカスタム・プログラムについての詳細を必要とします。それには、カスタム・プログラムの名前、それに渡す必要のあるコマンド・パラメーター、 IBM SPSS Modeler からプログラムへのオプションの提示方法およびユーザーへの結果の表示方法などが含まれます。この情報を提供するために、仕様ファイルと呼ばれる XML 形式のファイルを用意してください。IBM SPSS Modeler は、このファイルの情報をメニューの新規エントリーまたはノード定義に変換します。

以下に、CLEF 使用の利点を示します。

- エンジニア、コンサルタント、エンド・ユーザーが新機能を IBM SPSS Modeler に統合できる、使いやすく、柔軟性の高い強力な環境が提供されます。
- 拡張モジュールがネイティブの IBM SPSS Modeler モジュールと同じ外観で同じ動作ができるようにします。
- 拡張ノードを、ネイティブの IBM SPSS Modeler ノードとできる限り同じスピードおよび効率で実行できるようにします。

システムのアーキテクチャー

IBM SPSS Modeler 自体と同様、 CLEF では 2 層のクライアント/サーバー・アーキテクチャーを使用し、それらの層は同じマシンまたは 2 の異なるマシンに存在します。

クライアント側コンポーネント

クライアント層のコンポーネントを次に示します。

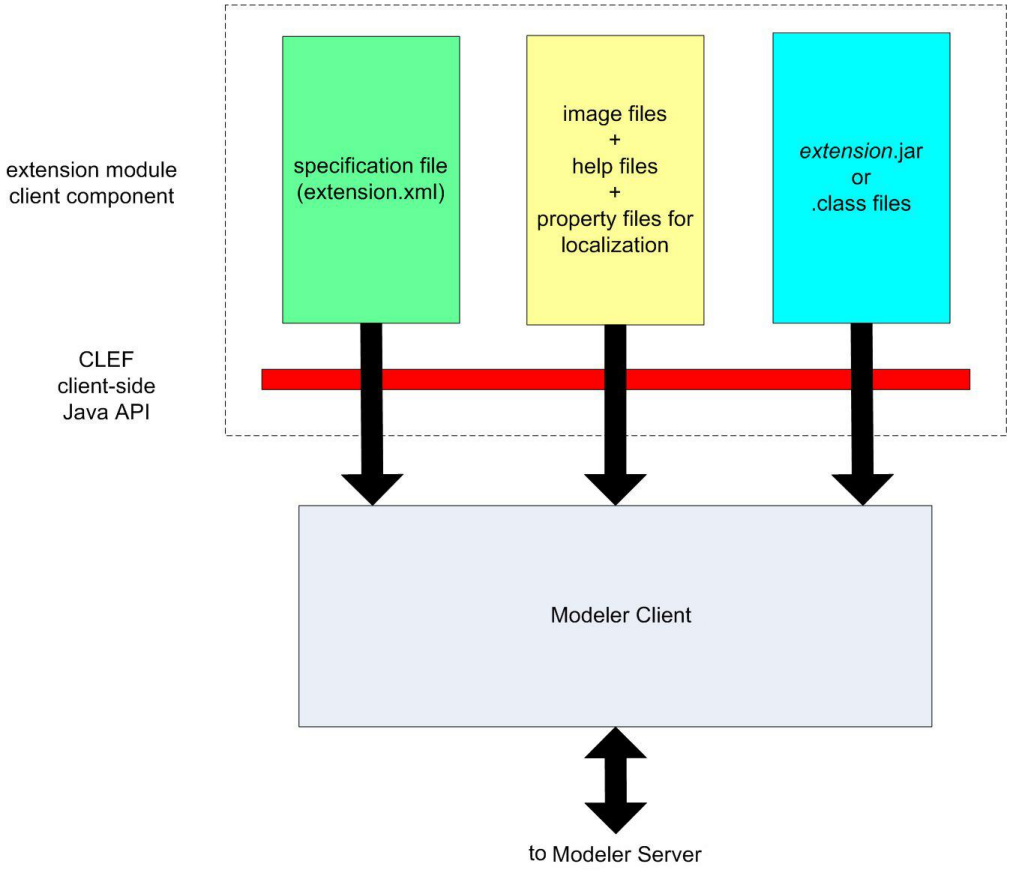

図 *1.* クライアント側コンポーネント

- 仕様ファイル**:** 拡張で定義されたプロパティー、形式、データ・モデルの変更、コントロール、その他の特性を列挙します。
- 画像ファイル**:** 拡張内でノードを識別するために使用される画像が含まれます。
- ヘルプ・ファイル**:** 拡張に関するヘルプ情報の表示に使用されます。
- プロパティー・ファイル**:** 拡張によって画面に表示される名前、ラベル、メッセージで構成されるテキスト文字列が含まれます。
- **Java** の **.jar** または **.class** ファイル**:** 拡張で使用された Java リソースが含まれます。
- **Java** アプリケーション・プログラミング・インターフェース **(API):** 仕様ファイルによって直接提供されない、追加のコントロールやユーザー・インターフェース・コンポーネントや双方向性を必要とする拡張によって使用できます。

サーバー側コンポーネント

サーバー層のコンポーネントを次に示します。

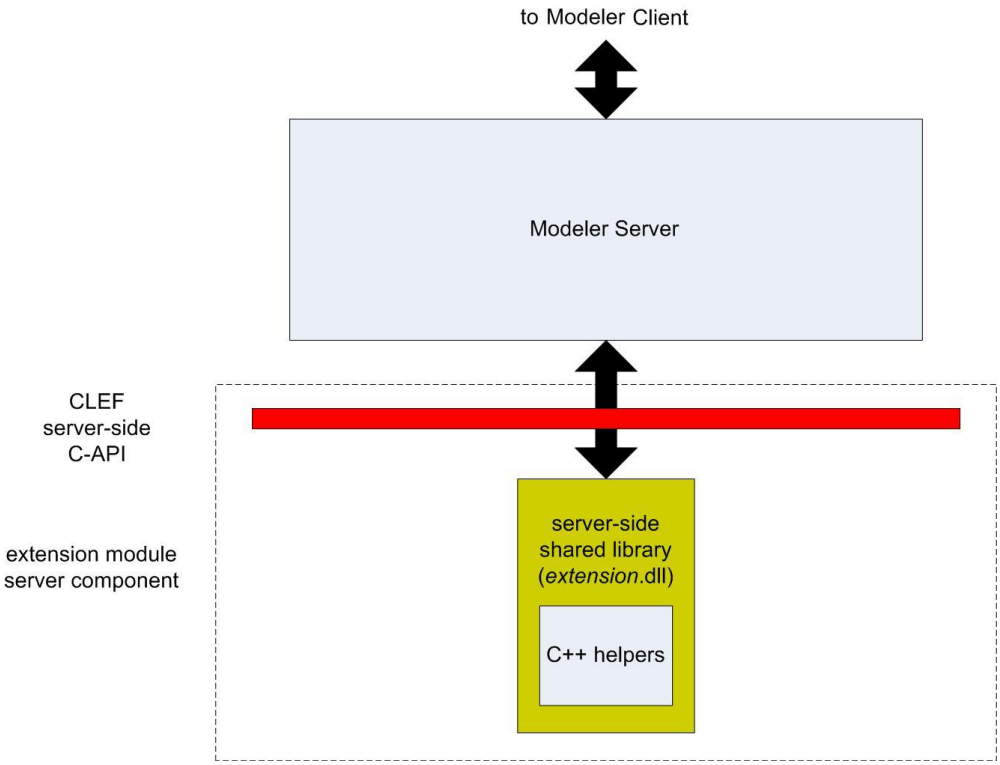

図 2. サーバー側コンポーネント

- 共有ライブラリーの **C** ベース **API :** 実行設定の設定と取得、それらの設定の持続、実行フィードバック、ジョブ制御 (実行の中断など)、SQL 生成、返されるオブジェクトなどの側面をカバーします。
- サーバー側共有ライブラリー **:** ノードの実行をサポートするダイナミック・リンク・ライブラリー (DLL)。C++ ヘルパーは、ソース・コードとして提供され、C++ CLEF モジュールに容易にコンパイルできる、一部の C ベース API のラッパーです。

CLEF の機能

以下の項で、CLEF の多くの主要な機能について説明します。

- 仕様ファイル
- ノード
- データ・モデル
- 入力ファイルと出力ファイル
- アプリケーション・プログラミング・インターフェース (API)

仕様ファイル

CLEF 仕様ファイルは、新しい拡張の動作を記述した構造仕様を格納する XML ファイルです。仕様ファイルは、以下について説明しています。

- 拡張に必要な共有リソース (ローカライズされたテキスト・バンドルおよびサーバー側の共有ライブラリーなど)。
- ファイル形式またはプロパティーの種類など、一般的な定義。
- ノードや出力モデルなど、エンド・ユーザーが使用できる新規オブジェクト。

IBM SPSS Modeler の起動時、仕様ファイルは格納場所からロードされ、ファイルに定義された機能はすぐに使用できます。

詳しくは、 31 ページの『第 4 章 仕様ファイル』を参照してください。

ノード

新規ノードを実装する IBM SPSS Modeler に拡張を追加する場合、まず作成するノードの種類を決定する必要があります (例えば、ノードがモデルを生成するかデータを転送するのみか)。 詳しくは、トピック 9 ページの『ノードの概要』を参照してください。

仕様ファイルと、必要なJava クラスおよび共有ライブラリーを作成した後、それらのファイルを IBM SPSS Modeler が読み取りできる特定の場所にコピーします。次回 IBM SPSS Modeler を起動すると、新規ノードが適切なパレットに追加され、使用する準備が整います。

データ・モデル

データ・モデルは、IBM SPSS Modeler ストリーム内を流れるデータの構造を表しています。ストリーム中の特定の位置のデータを記述するモデルは、データ型ノードに表示される情報に対応しています。ストリーム内にある特定のポイントの既存フィールドの名前を表示し、それらのデータ型を説明します。

IBM SPSS Modeler にノードを追加する場合、ノードに渡されるデータ・モデルがそのノードの動作にどのように影響するかを検討してください。例えば、フィールド作成ノードは、入力データ・モデルを取得し、そのデータ・モデルに新規フィールドを追加し、出力データ・モデルを生成して、IBM SPSS Modeler ストリーム内の次のノードに渡します。対照的に、グラフ・ノードは、データが他の後続ノードへ渡されないので、入力データ・モデルを取得しますが出力データ・モデルを生成しません。後続のノードが、どのフィールドが使用可能であるかという正確な情報を示すために、IBM SPSS Modeler はデータ・モデルに何が起こるのかを知る必要があります。仕様ファイル内にあるデータ・モデル情報は、ストリーム全体にわたってデータ・モデルの一貫性を保つために必要な情報を IBM SPSS Modeler に提供しています。

データが入力されるのか、出力されるのか、またはノードを通過するのかに応じて、入力、出力、または両方のデータ・モデルを仕様ファイルに記述する必要があります。CLEF ノードは、ノードに渡されるフィールドに新規フィールドを追加するか、またはノードに入力されるフィールドを、プログラム自体が生成したノードと置換することによって、データ・モデルに影響を与えることができます。仕様ファイルの `OutputDataModel` 要素は、 データモデルの CLEF ノードの影響を説明します。 詳しくは、トピック 60 ページの『出力データ・モデル』を参照してください。

入力ファイルと出力ファイル

CLEF ノードが実行される前に生成される一時ファイルを指定できます。サーバーのノード実行への入力であるため、これらのファイルは入力ファイルと呼ばれます。例えば、モデル・ビルダー・ノードには、内容がノードの実行時に指定の入力ファイルに転送されるモデル・コンテナーがあります。 詳しくは、トピック 56 ページの『入力ファイル』を参照してください。

モデル・ビルダーまたはドキュメント・ビルダー・ノード実行の結果など、その他の一時ファイルは、サーバーでのノードの実行中に生成されます。これらは出力ファイルと呼ばれ、ノード実行の後にクライアントに返されます。 詳しくは、トピック 57 ページの『出力ファイル』を参照してください。

アプリケーション・プログラミング・インターフェース (API)

拡張で何をするのかに応じて、アプリケーション・プログラミング・インターフェース(API) の利用を必要とする場合があります。単純なデータ転送であれば、必要な処理すべてを仕様ファイルで定義できる場合があります。一方、要件がより高度な場合には、以下の使用可能な API とのインターフェースが必要です。

- CLEF クライアント側 API
- CLEF サーバー側 API
- 予測サーバー API (PSAPI)

CLEF クライアント側 **API** は、追加のコントロール、ユーザー・インターフェース・コンポーネント、または仕様ファイルで直接提供されていない双方向性が必要な拡張によって使用できる Java API です。

CLEF サーバー側 **API** は、設定および実行設定の取得、これらの設定の持続、実行フィードバック、ジョブ制御 (実行の中止など)、SQL 生成、返されたオブジェクトなどの側面をカバーする C ベース API です。

予測サーバー **API** は、データ・マイニングおよび予測分析機能を必要とするアプリケーションが使用するための IBM SPSS Modeler 機能を公開する Java API です。

詳しくは、 179 ページの『第 9 章 プログラミング』を参照してください。

ファイル構造

CLEF の拡張は、2 つのコンポーネントのセットで構成されています。

- クライアント側コンポーネント
- サーバー側コンポーネント

クライアント側コンポーネントは、拡張の仕様ファイル、Java クラスおよび .jar ファイル、ローカライズ可能なリソースを含むプロパティー・バンドル、画像ファイルおよびヘルプ・ファイルで構成されています。

サーバー側コンポーネントは、拡張ノードが実行される場合に必要な共有ライブラリーおよび DLL です。

クライアント側コンポーネント

クライアント側コンポーネントは、IBM SPSS Modeler インストール・ディレクトリーの `¥ext¥lib` フォルダーにインストールされます。クライアント側コンポーネントの内容は、次のとおりです。

- 仕様ファイル
- Java クラスおよび `.jar` ファイル
- プロパティー・ファイル
- イメージ・ファイル
- ヘルプ・ファイル

拡張フォルダー

それぞれの拡張は、`¥ext¥lib` 直下の拡張フォルダーに配置されます。

拡張フォルダー名の表記方法は、以下をお勧めします。

`providerTag.id`

この場合、*providerTag* は、仕様ファイルの ExtensionDetails 要素のプロバイダ識別子で、*id* は同じ要素の拡張子です。

例としてExtensionDetails が次のように始まる場合には、

```
<ExtensionDetails providerTag="myco" id="sorter" ... />
```

拡張フォルダー名 myco.sorter が使用されます。

仕様ファイル

仕様ファイル自体の名前は extension.xml となり、拡張サブフォルダーの最上部に格納される必要があります。このため前述の例では、仕様ファイルへのパスは IBM SPSS Modeler インストール・ディレクトリーの下が次のようになります。

```
¥ext¥lib¥myco.sorter¥extension.xml
```

Java クラスおよび **.jar** ファイル

クライアント側 Java API を使用する拡張には、コンパイルされた Java コードが含まれます。このコードは、.class ファイルのセットとして残し、または .jar ファイルとしてパッケージ化できます。

Java .class ファイルは、 最上レベルの拡張フォルダーの相対パスに配置されます。例えば、ActionHandler インターフェースを実装するクラスのパスは、次のとおりです。

```
com.my_example.my_extension.MyActionHandler
```

この場合、.class ファイルは IBM SPSS Modeler インストール・ディレクトリーの下の次の場所にあります。

```
¥extension_folder¥com¥my_example¥my_extension¥MyActionHandler.class
```

.jar ファイルは、拡張フォルダーの下の任意の場所に置くことができます。仕様ファイルの JarFile 要素によって、.jar ファイルの実際の場所を指定します。例えば、拡張で以下のパスを持つ .jar ファイルを使用する場合

```
¥extension_folder¥lib¥common-utilities.jar
```

仕様ファイルでは、Resources 要素に次のエントリーがあります。

```
<Resources>
    <JarFile id="util" path="lib¥common-utilities.jar"/>
     ...
</Resources>
```

詳しくは、トピック 35 ページの『Jar ファイル』を参照してください。

プロパティー・ファイル

ローカライズされたリソース (画面のテキスト、エラー・メッセージ、およびそれらの外国語翻訳) は拡張子 .properties を持つファイルに保存され、拡張フォルダーの任意の場所に配置できます。 詳しくは、トピック 172 ページの『プロパティー・ファイル』を参照してください。

画像およびヘルプ・ファイル

アイコン表示用のグラフィック・イメージを含むファイル、ヘルプ・システムを含むファイルは、拡張フォルダーの下の任意の場所に配置できます。画像ファイルとヘルプ・ファイルを、それぞれのサブフォルダーに分けておくと便利です。

仕様ファイルの Icon 要素の imagePath 属性によって、画像ファイルの場所を宣言します。 詳しくは、トピック 110 ページの『アイコン』を参照してください。

同様に、仕様ファイルの HelpInfo 要素の path 属性を使用して、ヘルプ・システムの場所を宣言します。詳しくは、トピック 168 ページの『ヘルプ・システムの場所および種類の定義』を参照してください。

例

これらのコンポーネントに基づいたクライアント側のファイル構造は、次のようになります。

```
¥ext¥lib¥myco.sorter
¥ext¥lib¥myco.sorter¥extension.xml
¥ext¥lib¥myco.sorter¥sorter_en.properties
¥ext¥lib¥myco.sorter¥sorter_fr.properties
¥ext¥lib¥myco.sorter¥sorter_it.properties
¥ext¥lib¥myco.sorter¥com¥my_example¥my_extension¥MyActionHandler.class
¥ext¥lib¥myco.sorter¥help¥sorter.chm
¥ext¥lib¥myco.sorter¥images¥lg_sorter.gif
¥ext¥lib¥myco.sorter¥images¥sm_sorter.gif
¥ext¥lib¥myco.sorter¥lib¥common-utilities.jar
```

サーバー側コンポーネント

実行に必要な共有ライブラリーは、IBM SPSS Modeler インストール・ディレクトリーの ¥ext¥bin フォルダーの下にあるフォルダーに置く必要があります。次に例を示します。

installation_directory¥ext¥bin¥myco.sorter¥my_lib.dll

共有ライブラリーは、¥ext¥bin フォルダーに直接置くことはできません。

実行時に直接 IBM SPSS Modeler が起動する共有ライブラリーの場合、仕様ファイルの SharedLibrary 要素で場所を宣言します。 詳しくは、トピック 36 ページの『共有ライブラリー』を参照してください。

メインの共有ライブラリーが、他のライブラリーの使用を必要とする場合があります。また、メインの共有ライブラリーと同じ場所に従属共有ライブラリーを配置して、メイン・ライブラリーが従属ライブラリーを検出できるようにする必要があります。

例

サーバー側のファイル構造の例を以下に示します。

```
¥ext¥bin¥myco.sorter¥my_lib.dll
¥ext¥bin¥myco.sorter¥my_lib2.dll
```

第 2 章 ノード

ノードの概要

新規ノードを実装する拡張を作成する場合、IBM SPSS Modeler ノードの特性を理解する必要があります。ノードの特性を理解することによって、仕様ファイルの中で、ノードを正しく定義できます。

IBM SPSS Modeler ノードはその機能によって、入力、プロセス、出力、およびモデル作成ノードに分類されます。CLEF において、ノードは少し異なる方法で分類されます。2 つのシステム間のマッピングを以下の表に示します。

表 *1. CLEF* のノードの種類：

IBM SPSS Modeler の分類	パレット	**CLEF** ノードの種類
入力ノード	入力	データ・リーダー
プロセス・ノード	レコード設定(R)	データ変換
	フィールド設定(D)	
出力ノード	グラフ作成	ドキュメント・ビルダー
	出力 (レポート・ノード)	
	エクスポート	データ・ライター
モデル作成ノード	モデリング	モデル・ビルダー

CLEF ノードを新規作成する場合、CLEF ノードの種類の 1 つとして定義します。選択するノードの種類は、ノードの主要な機能によって異なります。

表 *2.* ノードの種類および機能：

CLEF ノードの種類	説明	対応するノード・パレット	アイコンの形状
データ・リーダー	異なる形式のデータを IBM SPSS Modeler にインポートします。	入力	 図 *3.* 入力ノードの形状 *(*丸*)*
データ変換	IBM SPSS Modeler からデータを取得し、データを何らかの方法で変更し、変更されたデータを IBM SPSS Modeler ストリームに戻します。	レコード設定、フィールド設定	 図 *4.* 設定ノードの形状 *(*六角形*)*
モデル・ビルダー	モデルを IBM SPSS Modeler のデータから生成します。	モデリング	 図 *5.* モデル・ビルダー・ノードの形状

表 2. ノードの種類および機能 (続き) :

CLEF ノードの種類	説明	対応するノード・パレット	アイコンの形状
ドキュメント・ビルダー	グラフまたはレポートを IBM SPSS Modeler のデータから生成します。	グラフ作成	 図 6. グラフ・ノードの形状 (三角形)
		出力 (レポート・ノード)	 図 7. 出力ノードの形状 (四角形)
モデル・アプライヤー (「モデル・ナゲット」)	IBM SPSS Modeler 領域に戻される生成されたモデルのコンテナーを定義します。	–	 図 8. モデル・アプライヤー・ノードの形状 (金のダイヤモンド)
データ・ライター	IBM SPSS Modeler 形式のデータを他のアプリケーションでの使用に適した形式にエクスポートします。	エクスポート	 図 9. エクスポート・ノードの形状 (四角形)

ノードの種類を、他の属性とともに仕様ファイルの `Node` 要素で定義します。以下に例を示します。

```
<Node name="sort_process" type="dataTransformer"
      palette="recordOp" ... >
     -- node elements --
</Node>
```

`palette` 属性は、ユーザーがノードにアクセスできる IBM SPSS Modeler メイン・ウィンドウにパレットを定義します。 この場合は、「レコード設定」パレットです。この属性を省略すると、ノードは「フィールド設定」パレットに表示されます。

多くのサンプル・ノードが IBM SPSS Modeler で提供されています。 詳しくは、トピック 25 ページの『サンプルについて』を参照してください。

データ・リーダー・ノード

データ・リーダー・ノードを使用すると、外部ソースのデータを IBM SPSS Modeler ストリームに読み込むことができます。IBM SPSS Modeler の「入力」パレットのノードは、データ・リーダー・ノードと同等のもので、 丸いアイコンの形状によって識別されます。

データ・リーダー・ノードの設定には、次の詳細が含まれます。

- データ・ソース (ファイルまたはデータベースなど)

- レコードの前処理 (前後のスペースまたはレコードの区切りとして使用する文字の処理など)
- レコード・フィールドの除外の有無
- 各フィールドに関連するデータの種類 (範囲型、セット型、フラグ型) およびストレージのデータ型 (文字列、整数、実数)
- 入力データ・モデルの変更の有無

データ・リーダー・ノードは、ロジックを使用して、ソース・データ・レコードを読み込むことができます。また、IBM SPSS Modeler のデータ型ノードを使用してより下流で実行することもできます。

サンプルのデータ・リーダー・ノードは IBM SPSS Modeler で提供されています。 詳しくは、トピック 25 ページの『サンプルについて』を参照してください。

データ変換ノード

データ変換ノードは、IBM SPSS Modeler ストリームからデータを取得、何らかの方法でデータを変更し、変更したデータをストリームに戻します。IBM SPSS Modeler の「レコード設定」および「フィールド設定」パレットのノードはデータ変換ノードで、六角形のアイコンの形状で識別されます。

データ変換ノードの設定には、次の詳細が含まれます。

- レコードまたフィールドのどちらを変換するか
- データの変更方法

サンプルのデータ変換ノードは IBM SPSS Modeler で提供されています。 詳しくは、トピック 25 ページの『サンプルについて』を参照してください。

モデル・ビルダー・ノード

IBM SPSS Modeler のモデル構築の概要は、『*IBM SPSS Modeler 18.1.1* アプリケーション ガイド』の「モデル作成の概要」を参照してください。

モデル・ビルダー・ノードは、IBM SPSS Modeler メイン・ウィンドウのマネージャ領域の「モデル」または「出力」タブに表示されるオブジェクトを生成します。

IBM SPSS Modeler の「モデル」パレットのノードは、モデル・ビルダー・ノードの例で、 五角形のアイコンの形状によって識別されます。

実行すると、モデル・ビルダー・ノードは「モデル」タブにモデル出力オブジェクト (モデル・ナゲット) を生成します。

生成されたモデルが領域に追加されると、モデル・アプライヤー・ノードの形式となります。

モデル・ビルダー・ノードの仕様には、次の内容を含めます。

- モデル構築の詳細。モデルの生成に使用されるアルゴリズムや、モデルによるデータのスコアリングにどの入力および出力フィールドが使用されるかなどです。
- モデルに使用されるプロパティー
- 出力プロジェクトの保有に使用されるコンテナー
- ノードのダイアログ・ボックスのユーザー・インターフェース
- ノードの実行時に使用されるプロパティーおよびファイル
- 入力データ・モデルがノードの実行によって受ける影響

- ノードの実行によって作成されるモデル出力オブジェクト (およびその他のオブジェクト) の識別子
- モデル・アプライヤー・ノードの識別子 (『モデル・アプライヤー・ノード』を参照)

注 : モデル・ビルダー・ノードを定義する場合、同じ設定ファイルの他の場所に実際のモデル出力オブジェクトおよびモデル・アプライヤー・ノードを定義します。

サンプルのモデル・ビルダー・ノードは IBM SPSS Modeler で提供されています。 詳しくは、トピック 25 ページの『サンプルについて』を参照してください。

ドキュメント・ビルダー・ノード

ドキュメント・ビルダー・ノードは、IBM SPSS Modeler メイン・ウィンドウのマネージャ領域の「出力」タブに表示されるオブジェクトを生成します。「グラフ」パレットのノードは、ドキュメント・ビルダー・ノードの例で、 三角形のアイコンの形状によって識別されます。

実行すると、ドキュメント・ビルダー・ノードはマネージャ領域の「出力」タブにドキュメント出力オブジェクトを生成します。

モデル出力オブジェクトとは対照的に、ドキュメント出力オブジェクトを、IBM SPSS Modeler 領域に戻すことはできません。

ドキュメント・ビルダー・ノードの設定には、次が含まれています。

- ドキュメント生成コントロールを含むノードのダイアログ・ボックスなど、ドキュメント構築の詳細
- ドキュメントに使用されるプロパティー
- 出力プロジェクトの保有に使用されるコンテナー
- ノードのダイアログ・ボックスのユーザー・インターフェース
- ノードの実行時に使用されるプロパティーおよびファイル
- ノードの実行によって作成されるドキュメント出力オブジェクトの識別子、その他のオブジェクト

注 : ドキュメント・ビルダー・ノードを定義する場合は、同じ仕様ファイルの別の場所で、実際のドキュメント出力オブジェクトを定義してください。

モデル・アプライヤー・ノード

モデル・アプライヤー・ノードは、モデルがマネージャ領域の「モデル」タブから IBM SPSS Modeler 領域に追加される場合に使用する生成されたモデルのコンテナーを定義します。

モデル・アプライヤー・ノードの設定には、次の詳細が含まれます。

- モデルのコンテナー (テキストおよび HTML など、モデル出力を複数の形式で作成できる場合、コンテナーは複数)
- ユーザーが「モデル」タブのアプライヤー・ノードを参照し、領域上で開く場合に表示されるダイアログ・ボックスのユーザー・インターフェースの詳細
- 出力データのモデル
- ノードを含むストリームが実行される場合に行う処理
- ノードを含むストリームが実行される場合に作成されるオブジェクトを処理するコンストラクター

データ・ライター・ノード

データ・ライター・ノードは、IBM SPSS Modeler 形式のデータを他のアプリケーションでの使用に適した形式にエクスポートします。IBM SPSS Modeler の「エクスポート」パレットのノードはデータ・ライター・ノードで、四角形のアイコンの形状で識別されます。

データ・ライター・ノードの設定には、次が含まれています。

- ストリーム・データが書き込まれるファイルまたはデータベースの詳細
- 外部アプリケーションに組み込むことができるよう、ストリーム全体を公開するかどうか (オプション)

メニュー、ツールバーおよびパレット

ユーザーは、IBM SPSS Modeler メニュー、ツールバーまたはパレットから拡張子にアクセスできます。拡張子はノードを実装または指定された動作を実行できます。

明示的に指定されたメニューからアクセスできる拡張子 (ノードまたは動作) は、ツールバーからもアクセス可能にすることができ、またツールバーからアクセス可能な拡張子をメニューからアクセスできるようにすることもできます。

パレットからアクセスできるノードは、自動的に「挿入」メニューの対応する項目からアクセスできます。

メニューおよびサブメニュー

ユーザーは、「挿入」メニューから標準 IBM SPSS Modeler ノードにアクセスできます。「挿入」メニューの最後のグループの各項目には (「モデル」とは別に)、一連の関連するノードへのアクセスを提供するサブメニューがあります。

これらの項目は、ノード・パレットのエントリーに直接対応します。パレットにノードを追加すると、ノードを「挿入」メニューの対応するグループに自動的に追加します。

拡張子がノード全体にアクセスできない動作を定義する場合、次のいずれかまたは複数を追加して拡張子を使用できるようにします。

- 新しい項目をシステム メニューまたはサブメニューに追加
- 新しいメニューを IBM SPSS Modeler に追加
- 新しい項目をツールバーに追加 (『ツールバー』 を参照)

新しいメニューまたはメニュー項目は、オプションで拡張子に関連するアイコンを「挿入」メニュー項目に表示できます。

詳しくは、 112 ページの『メニュー』および 113 ページの『メニュー項目』を参照してください。

ツールバー

拡張子がノード全体にアクセスできない動作を定義する場合、メイン IBM SPSS Modeler ツールバー追加して拡張子を使用できるようにします。

この場合、動作を表すラベルを隠すことをお勧めします。

ノードのダイアログ・ボックスまたは出力ウィンドウのツールバーに項目を追加することもできます。項目ラベルの表示と非表示を切り替えることができます。

詳しくは、トピック 114 ページの『ツールバー項目』を参照してください。

パレットおよびサブパレット

拡張子が新規ノードを定義する場合、任意の場所のノードを標準 IBM SPSS Modeler パレットまたはサブパレットのいずれかに配置できます。

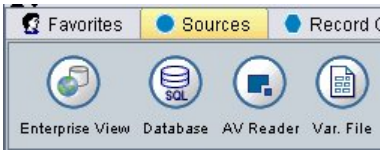

図 *10.* 標準パレットの新規ノード

エントリーを標準サブパレットに追加し、ノードをサブパレットからアクセスできるようにします。

図 *11.* 新規ノードの標準パレットへのカスタム追加

カスタム・パレットを定義し、新規ノードを配置できます。

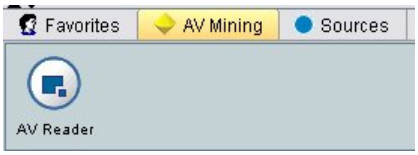

図 *12.* カスタムパレットの新規ノード

カスタム・パレットにはカスタム・サブパレットを配置できます。

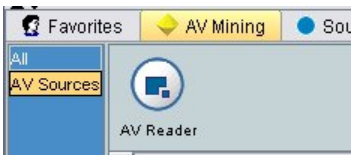

図 *13.* カスタム・パレットのカスタム・サブパレットの新規ノード

詳しくは、 48 ページの『ノード』および 43 ページの『User Interface (Palettes) セクション』を参照してください。

ノード・アイコンのデザイン

CLEF で作成するそれぞれの新規ノードについて、画面上のノードを識別するアイコンの中央の画像を提供できます。

注 : IBM SPSS Modeler では指定がない場合に表示されるデフォルトを提供しているため、画像を提供する必要はありません (ノード開発の開始時に役に立ちます)。

図 *14. CLEF* アイコンのデフォルト・イメージ

標準的な IBM SPSS Modeler アイコンは、次の 3 つのレイヤーで作成されます。

- 枠線
- 背景
- 中央の画像

新規ノードの場合、中央の画像 (グリフ) のみ提供する必要があります。IBM SPSS Modeler では、枠線および背景の処理を行います。アイコンの背景レイヤーが見えにくいことのないよう、グリフ イメージの背景は透明である必要があります。このセクションでは、グリフの表示では色の付いた背景が透過度を示しています。

このようにして、通常の IBM SPSS Modeler モデル作成アイコンが作成されます。

表 *3.* ノード・アイコンと生成されたモデル・アイコンの構成

	ノード・アイコン	生成されたモデル・アイコン
枠線	 図 *15.* アイコンの枠線	なし
背景	 図 *16.* アイコンの背景	 図 *17.* 生成されたモデルの背景
グリフ	 図 *18.* アイコンのグリフ	 図 *19.* 生成されたモデルのグリフ
表示される画像	 図 *20.* 表示されるアイコン	 図 *21.* 表示される生成されたモデルのアイコン

枠線

ノードの機能は、アイコンの枠線の形状で識別されます。 詳しくは、トピック 9 ページの『ノードの概要』を参照してください。

ノードがキャッシュを有効化している場合、枠線の形状に縮小された文書のシンボルが追加されています。ノードに表示されている文書アイコンが白い場合、キャッシュが空であることを表しています。キャッシュがいっぱいになると、文書アイコンは緑色で塗られます。

表 4. ノードの枠線およびキャッシュ ステータス

キャッシュ ステータス	例
キャッシュ なし	 図 22. キャッシュのないノード
有効なキャッシュ	 図 23. キャッシュが有効なノード
いっぱいになったキャッシュ	 図 24. キャッシュがいっぱいのノード

さまざまな枠線のシンボルがシステムに提供され、IBM SPSS Modeler は必要な処理を行い、適切なときに適切なシンボルを表示します。

背景

生成されたモデルおよびモデル・アプライヤー・ノードのノード・アイコン以外のノードアイコンの場合、背景の色が変わり、状態を表します。

表 5. ノードの背景

状態	色	例
選択解除状態	灰色	 図 25. アイコンの背景 (灰色)
選択	青	 図 26. アイコンの背景 (青)

表 5. ノードの背景 (続き)

状態	色	例
エラー	赤	 図 27. アイコンの背景 (赤)
データベースで実行する動作	紫	 図 28. アイコンの背景 (紫)

再度、背景の画像がシステムによって提供され、IBM SPSS Modeler で必要な処理を実行し、それぞれの状況に適切な背景を表示します。

グラフィックの必要条件

それぞれの新規 CLEF ノードに対し、次のバージョンのグリフ レイヤー画像を作成します。

- ストリーム領域のノードの場合、大きいサイズ (49 x 49 ピクセル)
- 画面下部のパレット・マネージャ内のノードの場合、小さいサイズ (38 x 38 ピクセル)

メニューまたはツールバー、ブラウザーまたは出力ウィンドウのタイトル・バーにアイコンを表示する場合、次を作成する必要もあります。

- 縮小サイズ (16 x 16 ピクセル)

ノードがモデルを生成する場合、次を作成する必要もあります。

- 生成されたモデルのアイコン (金色のナゲット) にオーバーレイする場合、左下に移動したデザインの小さいサイズ (38 x 38 ピクセル)

注 : IBM SPSS Modeler に表示する場合、これらのサイズを超える画像は一部が切り取られます。

詳しくは、トピック 110 ページの『アイコン』を参照してください。

カスタム画像の作成

ノードに作成する画像は、ノードの主要な機能を示す必要があります。海外のユーザーのために、1 つの国に特有のものにせず、他の国のユーザーに誤解されないような画像を使用する必要があります。

CLEF で使用する画像を作成する手順は、次のとおりです。

1. 透過度をサポートするグラフィック パッケージを使用して、描画領域を適切なサイズに設定し、イメージ バージョンを描画します。
2. それぞれのバージョン (大、小など) を、次の特性を持つ個々の *.gif* ファイルで保存します。
   - 透明な背景
   - 16 色 (4 ビット) 以上の色数

画像の背景透過度の設定方法は、使用するグラフィック パッケージによって異なります。例えば、背景色を透過度に直接設定できる場合もあれば、透過色を指定してから、画像の背景をこの色で「塗る」必要がある場合もあります。

画像ファイルの場合、IBM SPSS Modeler が内部的に使用するファイル命名規則に従うことをお勧めします。以下の表にその命名規則を示します。

表 *6.* 画像ファイル名の表記方法

画像の種類	ファイル名
大(L)	lg_*node*.gif
小	sm_*node*.gif
縮小	*node*16.gif
生成されたモデル	sm_gm_*node*.gif

3. 仕様ファイルの画像ファイルを参照し (『画像ファイルをノード設定に追加』を参照)、新規ノードを IBM SPSS Modeler に追加することにより (205 ページの『CLEF 拡張のテスト』を参照)、画像の外観をテストします。

画像ファイルをノード設定に追加

画像ファイルを作成した後、IBM SPSS Modeler を実行するコンピューターのフォルダーに画像をコピーします。設定ファイルにおいて、IBM SPSS Modeler インストール・ディレクトリーの ¥ext¥lib¥*provider.nodename* フォルダーに関連する画像パスを宣言し、アクセスしやすいフォルダーにファイルを展開する必要があります、 詳しくは、トピック 110 ページの『アイコン』を参照してください。

仕様ファイルで、Node 仕様の UserInterface セクションの Icons 要素を使用して、大きなアイコン・グラフィック・ファイルと小さなアイコン・グラフィック・ファイルをカスタム・ノードに関連付けます。以下に例を示します。

```
<Icons>
     <Icon type="standardNode" imagePath="images/lg_mynode.gif" />
     <Icon type="smallNode" imagePath="images/sm_mynode.gif" />
</Icons>
```

モデル・ビルダー・ノードまたはドキュメント・ビルダー・ノードの場合、ModelOutput 仕様 (モデル・ビルダー・ノードの場合) または DocumentOutput 仕様 (ドキュメント・ビルダー・ノードの場合) の UserInterface セクションの縮小バージョン (16 x 16 ピクセル) も参照する必要があります。以下に例を示します。

```
<Icons>
     <Icon type="standardWindow" imagePath="images/mynode16.gif" />
</Icons>
```

モデル・アプライヤー・ノードの場合、Node 仕様の UserInterface セクションの生成されたモデルのバージョンも参照する必要があります。以下に例を示します。

```
<Icons>
     <Icon type="standardNode" imagePath="images/lg_gm_mynode.gif" />
     <Icon type="smallNode" imagePath="images/sm_gm_mynode.gif" />
</Icons>
```

ダイアログ・ボックスのデザイン

この項では、CLEF の一貫したダイアログ・ボックスのデザインを支援する、標準 IBM SPSS Modeler ノード・ダイアログ・ボックスの特性を説明しています。

ノードのダイアログ・ボックスについて

ノードのダイアログ・ボックスは、エンド・ユーザーが実行に関する設定の変更を可能にするインターフェースを提供しています。ダイアログ・ボックスでノードの動作を修正、変更できるため、ダイアログ・ボックスの外観は非常に大切になります。このインターフェースには必要なすべての情報を記載し、しかも使いやすくなければなりません。

ノードの動作は、ユーザーが対話処理できるユーザー・インターフェースの要素である、さまざまなダイアログ・ボックス・ベースの コントロール の使用によって変更されます。ダイアログ・ボックスには、ラジオ・ボタン、チェック・ボックス、テキスト・ボックス、およびメニューなど多くのコントロールで構成されています。CLEF では、ダイアログ・ボックスにデザインできるさまざまなコントロールが提供されています。 詳しくは、トピック 125 ページの『プロパティー・コントロール設定』を参照してください。

変更されたパラメーターの種類によって、どのコントロールがダイアログ・ボックスに表示されるかが決まります。種類によっては、代替コントロールがあるものもあります。設定ファイルの `Tab` 要素によって、新しいタブのオプションをグループ化できます。 詳しくは、トピック 22 ページの『タブ領域』を参照してください。

注：拡張が実行する処理を指定していない場合でも、拡張のユーザー・インターフェースの外観をテストできます。 詳しくは、トピック 205 ページの『CLEF 拡張のテスト』を参照してください。

ダイアログ・ボックス・デザインのガイドライン

ダイアログ・ボックスのコントロールを定義する際は、次のガイドラインを参考にしてください。

- コントロールに表示ラベルがある場合、使用するテキストを慎重に考慮してください。テキストは、簡潔で正しく情報を伝達できるものでなければなりません。国際マーケット向けにデザインする場合、翻訳されたテキストの長さが元のテキストとは大きく異なる場合があることに留意してください。
- パラメーターに対して正しいコントロールを使用してください。例えば、チェック・ボックスは 2 つの値のみをとるパラメーターに対して最善の選択とは限りません。IBM SPSS Modeler C5.0 ノードのダイアログ・ボックスでは、ユーザーが出力タイプを選択できるラジオ・ボタンが使用されます。このボタンは、「デシジョン ツリー」と「ルール セット」 のいずれかの値を使用します。

  この設定は、「デシジョン ツリー」 というラベルを持つチェック・ボックスとして表すことができます。このオプションが選択されると、出力タイプはデシジョン・ツリーになります。選択が解除されると、出力はルール・セットになります。結果は同じでも、ラジオ・ボタンを使った方が、ユーザーにとってはわかりやすいものとなります。
- 通常ファイル名のコントロールは、ダイアログ・ボックスの上部に配置するようにしてください。
- ノードのフォーカスを形成するコントロールは、ダイアログ・ボックスの上の方に配置してください。例えば、グラフ・ノードはデータからフィールドを表示します。これらのフィールドを選択することがダイアログ・ボックスの主な機能になるため、この場合はフィールド・パラメーターを上部に配置します。
- さらに詳細な情報が必要なため、ユーザーにオプションを選択させる場合、チェック・ボックスやラジオ・ボタンを使用します。例えば、C5.0 ダイアログ・ボックスで 「ブースティングを使用」 を選択する場合、分析に 「繰り返し回数」 を示す数字を入れる必要があります。

このような付加情報は、常にオプションの選択の後、右側か直下に配置します。

CLEF のダイアログ・ボックスでは、標準的な IBM SPSS Modeler のダイアログ・ボックスと同じ方法で IBM SPSS Modeler のコミット編集を使用します。「**OK**」、「適用」、またはターミナル・ノードの場合は 「実行」 をクリックするまで、ダイアログ・ボックスに表示される値はノードにコピーされません。同様に、ユーザーがキャンセルしてダイアログ・ボックスを再表示するか、または 「リフレッシュ」 ボタンをクリックしない限り、ダイアログ・ボックスに表示される情報が更新されることはありません (例えば、現在のノードの上流における操作によりノードの入力フィールドが変更された場合)。

ダイアログ・ボックス・コンポーネント

ダイアログ・ボックスには、次のコンポーネントがあります。

- タイトル・バー
- アイコン領域
- 以下を組み込むツールバーおよびメニュー領域
  - 「ファイル」、「生成」、「表示」、「プレビュー」、「リフレッシュ」などのボタン (ノードによって異なります)
  - 最大/通常サイズ・ボタン
  - ヘルプ・ボタン
- ステータス領域
- パネル領域
- タブ領域
- ボタン領域

それぞれのカスタム・ノードには、ユーザーがノードを開いたときに表示されるダイアログ・ボックスが必要です。設定ファイルに `Node` 要素を持つ `UserInterface` セクションを含む `Tabs` 要素がある場合、ノードを開くと前述のダイアログ・ボックス・コンポーネントがすべて表示されます。ノードの種類に応じたタブ領域とボタン領域の最小限の内容を以下の表に示します。

表 7. ノードの種類別のタブ領域およびボタン領域の最小限の内容

ノード・タイプ	タブ	ボタン
データ・リーダー	「注釈」 (ツールバー領域の「リフレッシュ」ボタンを含む)	「OK」、「キャンセル」、「適用」、「リセット」
データ変換	注釈	「OK」、「キャンセル」、「適用」、「リセット」
データ・ライター	「公開」、「注釈」	「OK」、「キャンセル」、「実行」、「適用」、「リセット」
モデル・ビルダー	注釈	「OK」、「キャンセル」、「実行」、「適用」、「リセット」
ドキュメント・ビルダー	注釈	「OK」、「キャンセル」、「実行」、「適用」、「リセット」
モデル・アプライヤー	「概要」、「注釈」	「OK」、「キャンセル」、「適用」、「リセット」

最初、ノードのダイアログ・ボックスはユーザーがノードを開いた場合に、ノードのアイコンが表示するノードに付加されるように生成されます。ユーザーはダイアログ・ボックスを移動できますが、新しい位置は次のノードを開くときには記憶されていません。ユーザーがダイアログ・ボックスを移動し、別のダイアログボックスに一部または全体が隠れている場合、領域内の元のノードをダブルクリックすると、最初のダイアログ・ボックスが再度前面に表示されます。ダイアログ・ボックスはモードレス (同じユーザーの入力は常に同じ動作を実行) で、サイズ変更可能です。

ダイアログ・ボックスの編集可能なすべてのフィールドで、以下の表に示すキーボード・ショートカットがサポートされています。

表 *8.* ダイアログ・ボックスの編集可能フィールドのキーボード・ショートカット

ショートカット	効果
Ctrl-C	コピー
Ctrl-V	貼り付け
Ctrl-X	切り取り

タイトル・バー

ノードのダイアログ・ボックスのタイトル・バーでは、モデル名の前に IBM SPSS Modeler のナゲットのアイコンの縮小バージョンが表示されます。テキストは、モデル名コントロールの設定から取得されます。デフォルトでは、右上に「閉じる」ボタン (X) が表示されます。

アイコン領域

ノード・アイコンは、ダイアログ・ボックスの左上部のアイコン領域に表示されます。これは、領域に表示される大きいバージョンではなく、メイン・ウィンドウの下部のノード・パレットにも使用されるアイコンの小さいサイズ (38 x 38 ピクセル) のバージョンです。

注 : タイトル・バーの左端に表示されるナゲットの縮小アイコンは、すべてのノードのダイアログ・ボックスにハードコードされています。

ツールバーおよびメニュー領域

ダイアログ・ボックスの最上部は、ツールバーおよびメニュー領域として使用されます。

データ・リーダーおよびデータ・トランスフォーマー・ノードのダイアログのこの領域には、入力データのサンプルを表示する「プレビュー」ボタンがあります。

データ・リーダー・ノードのダイアログには、ノードへの入力フィールドが変更された場合などにノードの表示される情報を更新する「リフレッシュ」ボタンがあります。

モデル・アプライヤー・ノードには「ファイル」、「ノードの生成」、「表示」のメニュー・ボタンがあり、ユーザーはモデルのエクスポートまたはノードの新規作成など、さまざまな操作を実行することができます。モデル・アプライヤー・ノードには「プレビュー」ボタンもあり、この場合、ノード適用時に作成された追加の列とともに入力データのサンプルを表示します。

すべてのノードのダイアログ・ボックス内の最上部領域の右側には、次の 2 つのボタンがあります。

- 「最大化/通常サイズ」ボタン
- 「ヘルプ」ボタン

「最大化/通常サイズ」ボタン: このボタンは、ダイアログ・ボックスをフル・スクリーン・サイズにサイズを変更します。再度ボタンを押すと、ダイアログ・ボックスを最大化する前のサイズに戻します。

「ヘルプ」ボタン: このボタンを使用すると、ノードのコンテキスト・ヘルプを開きます。タブで分類されたダイアログ・ボックスまたは出力ウィンドウの場合、該当するタブのヘルプが表示されます。「F1」キーを使用して、ヘルプにアクセスすることもできます。

ステータス領域

ダイアログ・ボックス上部の残りの領域は、情報、警告、エラー・テキストの表示に使用されます。入力ノードは、ソース・データ・ファイルのフル・パスおよびファイル名を表示します。個々のノードには、この領域に表示される他のノード固有の情報が含まれます。この領域に指定されたテキストは、2 行に制限されます。

パネル 領域

これはダイアログ・ボックスの主要な領域で、すべてのコントロールが含まれ、ノードの領域を表示します。それぞれのタブには、異なるパネル領域があります。パネルの種類はそれぞれ、次のいずれかになります。

- テキスト・ブラウザー
- 拡張オブジェクト
- プロパティー

新しいウィンドウで開き、パネルの操作ボタンで呼び出される個別のダイアログ・ボックスであるサブパネルを指定することもできます。

詳しくは、トピック 119 ページの『パネル設定』を参照してください。

タブ領域

ノードのダイアログ・ボックスには、次のタブが含まれています。

- ユーザーが提供したノード固有のタブ
- 「要約」タブ (モデル出力オブジェクトおよびモデル・アプライヤー・ノードのみ)
- 「注釈」タブ

ノード固有のタブは、CLEF 設定ファイルの `Tabs` セクションで定義されます。 詳しくは、トピック 116 ページの『タブ』を参照してください。

モデル出力オブジェクトおよびモデル・アプライヤー・ノードのダイアログ・ボックスには、システムで提供された「要約」タブがあります。ここには、フィールド、構築設定、使用されたモデル推定プロセスなど、生成されたモデルについての情報が表示されます。結果は、特定の項目をクリックすると開いたり閉じたりできるツリーで表示されます。

「注釈」タブは、すべてのノードのダイアログ・ボックスのシステムによって提供され、ユーザーはノードに関する情報を指定できます。この情報には、ノード名、ツールヒント テキスト、長いコメント・フィールドが含まれます。

名前: デフォルトのノード名、設定ファイルの Node 要素の `Label` 属性で指定されます (48 ページの『ノード』を参照)。ユーザーは、カスタム を選択し、「カスタム」編集フィールドに名前を入力し、「適用」 または 「**OK**」 をクリックして、ノードの名前を変更できます。新しい名前はセッション全体で保存されますが、デフォルト名は 「自動」 を選択して復元することができます。「注釈」タブで指定されたカスタム名は、ダイアログ・ボックスの他のタブで指定されたカスタム名を上書きします。

ツールチップ・テキスト: ここで指定されたテキストは、領域上のノードのツールヒントとして表示されます。ここでツールチップ テキストが指定されていない場合、ノード上にカーソルを移動してもツールヒントは表示されません。

キーワード: ユーザーはキーワードを指定して、プロジェクト・レポートでおよび IBM SPSS Collaboration and Deployment Services Repository に保存されたオブジェクトを検索または追跡する場合に使用できます。

コメント・パネル: この領域で、ユーザーはコメント テキストを入力できます。

作成および保存情報: これは、作成情報および名前、ファイルが保存された日時 (日時の形式はロケールによって異なります) を表示する編集不可能なテキスト領域です。保存が行われていない場合、このフィールドには「この項目は保存されていません」と表示されます。

ボタン領域

すべてのダイアログ・ボックスの下部に、「適用」、「リセット」、「**OK**」、および 「キャンセル」 が表示されます。ノードがターミナル・ノード (ストリーム・データを処理する実行可能なノード) の場合、「実行」 ボタンも表示されます。

OK。すべての設定を適用し、ダイアログ・ボックスを閉じます。ダイアログ・ボックスが最初にノードから開いた場合、このボタンにはフォーカス (ボタンの周りに青い四角形で表示) があり、また Enter キーを押しても「OK」の操作ができます。

キャンセル: ダイアログ・ボックスを閉じ、設定をダイアログ・ボックスを開く前、または最後の「適用」操作以降のままにします。ダイアログ・ボックス全体にフォーカスがある場合、Esc キーを押すと「キャンセル」の操作ができます。

実行: すべての設定を適用し、ダイアログ・ボックスを閉じてターミナル・ノードを実行します。

適用。ダイアログ・ボックスの設定を保存し、下流の操作で使用できるようにします。

リセット。ダイアログ・ボックス全体をダイアログ・ボックスを開いたときの値、または最後の「適用」操作以降の値にリセットします。

出力ウィンドウのデザイン

この項では、CLEF の一貫した出力ウィンドウのデザインを支援する標準 IBM SPSS Modeler 出力ウィンドウの特性を説明します。

出力ウィンドウを使用すると、以下の出力を表示することができます。

- モデル (一連のデータをスコアリングし、そのデータにモデルを適用した結果など)
- ドキュメント (グラフやレポートなど)

詳しくは、トピック 107 ページの『ユーザー・インターフェースについて』を参照してください。

出力ウィンドウは、ノードのダイアログボックスと類似していますが、次の点が異なります。

- タイトル・バーには、標準的な金色のナゲットアイコンではなく、ノード固有の縮小アイコンが表示されます。
- メイン・ノード・アイコンは省略されます。

- ツールバーおよびメニュー領域では、「最大化/通常」ボタンが省略されますが (ドキュメント出力ウィンドウでは、「閉じる」および「削除」ボタンと置き換えられる場合があります)、ウィンドウはマウスを使用するとサイズ変更できます。
- ステータス領域が省略されます。
- タブは通常、以下のとおりです。
  - このオプションがモデル・ノードで選択されている場合に予測値の重要度データを表示する (モデル出力ウィンドウの)「モデル」タブ。
  - 出力の単一タブ。
  - モデルについて要約の詳細を表示する (モデル出力ウィンドウの)「要約」タブ。
  - 「注釈」タブ (注釈の値は出力を生成したノードから取得)。
- ボタン領域には「OK」、「キャンセル」、「適用」および「リセット」ボタンがあります

CLEF では、前述のウィンドウに類似したデフォルトのモデル出力およびドキュメント出力ウィンドウが提供されます。通常、設定ファイルの `ModelOutput` または `DocumentOutput` 要素を使用する場合に表示されます。 詳しくは、トピック 48 ページの『オブジェクト識別子』を参照してください。

また、`ModelOutput` または `DocumentOutput` 要素を指定して、デフォルトの出力ウィンドウを独自にデザインしたカスタム ウィンドウを置き換えることができます。 詳しくは、トピック 165 ページの『カスタム出力ウィンドウ』を参照してください。

第 3 章 CLEF の例

サンプルについて

CLEF を理解するため、IBM SPSS Modeler インストールには、完全なソース・コードとともにサンプル・ノードのセットが含まれています。これらは、制限された機能を持つ基本的なノードで、CLEF の動作方法および使用方法を理解するために設計されています。これらのノードを今すぐに、またはいつでも使用できます。

サンプル・ノードは次のとおりです。

- データ・リーダー・ノード (Apache Log Reader)
- データ変換ノード (URL Parser)
- ドキュメント・ビルダー・ノード (Web Status Report)
- モデル・ビルダー・ノード (Interaction)

使用する前に、ノードの例を有効化する必要があります。

サンプルの有効化

サンプルは、IBM SPSS Modeler インストールの一部として、Demos ディレクトリーに圧縮された形式でインストールされています。次のようにファイルを適切な場所に抽出することでサンプルを有効化する必要があります。

IBM SPSS Modeler がインストールされたコンピューター上で次のように実行します。

1. IBM SPSS Modeler が実行中の場合は終了します。
2. ファイル clef_examples_ext_lib.zip を、IBM SPSS Modeler インストール・ディレクトリーの Demos フォルダーに置きます。
3. clef_examples_ext_lib.zip の内容を、IBM SPSS Modeler インストール ディレクトリーの¥ext¥lib フォルダーに解凍します。

IBM SPSS Modeler または IBM SPSS Modeler Server (あるいはその両方) がインストールされたコンピューター上で、以下の処理を実行します。

1. clef_examples_ext_bin.zip の内容を、IBM SPSS Modeler と IBM SPSS Modeler Server の両方のインストール ディレクトリーにある ¥ext¥bin フォルダーに解凍します。
2. Windows 以外のシステムの場合は、clef_examples_ext_bin.zip で提供されている Make ファイルを使用して、目的のサンプルのソース コードをコンパイルします。詳しくは、トピック 29 ページの『ソース・コードの検査』を参照してください。

   または

   Windows の場合は、以下の手順に従って、目的のサンプルのソース コードをコンパイルします (Visual Studio 2008 が必要になります)。

   a. ステップ 1 で clef_examples_ext_bin.zip を解凍した ¥ext¥bin ディレクトリーで、コンパイルする CLEF 拡張のサンプルが格納されているサブディレクトリーに移動します (例: spss.apachelogreader)。

b. src サブディレクトリーで、.sln ファイルをダブルクリックして、CLEF 拡張のソリューションを Visual Studio で開きます (例えば、¥ext¥bin¥spss.apachelogreader¥src¥apachelogreader.sln をダブルクリックします)。

c. Visual Studio で、「ビルド」 > 「構成マネージャー」に移動します。

d. 「アクティブ ソリューション構成」として、「リリース」を選択します。

e. 「アクティブ ソリューション プラットフォーム」として、「**x64**」を選択します。

f. 「閉じる」をクリックします。

g. プロジェクトをビルドするために、「ビルド」 > 「ソリューションのビルド」に移動します (または F7 をクリックします)。

結果として生成された 64 ビットの DLL が、以下の場所 (src フォルダーからの相対的な場所) に書き込まれます。

```
..¥bin¥win64¥release¥spss.<extension-name>¥<extension-name>.dll
```

(例: ..¥bin¥win64¥release¥spss.apachelogreader¥apachelogreader.dll)。

いずれの場合も、最後に IBM SPSS Modeler を起動し、以下の表に示すノードがノード・パレットに表示されることを確認します。

表 *9.* ノード・パレットに表示されるノード:

「パレット」タブ	ノード
入力	Apache Log Reader
フィールド設定(D)	URL Parser
モデリング	交互作用
出力	Web Status Report

データ・リーダー・ノード (Apache Log Reader)

データ・リーダー・ノードのサンプルは、Apache HTTP Web サーバーのアクセス・ログ・ファイルからデータを読み込む入力ノードです。アクセス・ログには、Web サーバーが処理したすべての要求の詳細が含まれています。ログ・レコードは、Combined Log Format と呼ばれる形式です。以下に例を示します。

```
IP_address - - [09/Jul/2007:07:57:38 +0000] "GET /lsearch.php?county_id=3 HTTP/1.1" 200 16348
"http://www.google.co.uk/search?q=thunderbirds+cliveden&hl=en&start=10&sa=N"
"Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 5.1; .NET CLR 1.1.4322)"
```

サンプル・ノードを使用して、ログ・レコードを読みやすいテーブル形式に変換できます。

Apache Log Reader ノードを使用する手順は、次のとおりです。

1. CLEF サンプルがまだ有効化されていない場合は、有効化します。 詳しくは、トピック 25 ページの『サンプルの有効化』を参照してください。
2. IBM SPSS Modeler を開きます。
3. ノード・パレットの「入力」タブで、「**Apache Log Reader**」 を選択してストリーム領域に追加します。
4. ノードを編集します。「オプション」タブの「Apache ログ・ファイル」フィールドで、次を入力します。

demos_folder¥combined_log_format.txt

この場合、*demos_folder* は、IBM SPSS Modeler インストール・ディレクトリーの *Demos* フォルダーです (*$CLEO_DEMOS* 形式は使用しないでください)。

5. 「**OK**」をクリックします。
6. タイプ ノードをストリームに追加します。
7. データ型ノードを編集します。「値の読み込み」 をクリックしてデータを読み込み、「**OK**」 をクリックします。
8. テーブルノードをデータ型ノードに接続してストリームを実行します。ログ・ファイルの内容は、テーブル形式で表示されます。
9. 次の 2 つのサンプルと使用するためにストリームを保存します。

データ変換ノード (URL Parser)

データ変換ノード・サンプルは、前述のサンプルで戻されたデータにより高度な処理を実行します。ID フィールド (各行に一意の値を表示) および URL が表示された入力フィールドを選択します。ノードでは、これら 2 つのフィールドで構成される出力データを生成し、生成された各フィールドに URL データを解析します。例えば、次の URL レコードにクエリー文字列

```
http://www.dummydomain.co.uk/resource.php?res_id=89
```

レコードは、以下の表に示すように解析されます。

表 *10. URL* レコードの解析例：

生成されたフィールド	内容
*URLfield*_server	*http://www.dummydomain.co.uk*
*URLfield*_path	*/resource.php*
*URLfield*_field	*res_id*
*URLfield*_value	*89*

URL Parser ノードを使用する手順は、次のとおりです。

1. 前述のサンプルからのストリームが閉じられている場合は、開きます。ストリームには、Apache Log Reader とデータ型ノードが含まれています。
2. ノード・パレットの「フィールド設定」タブで、URL Parser ノードをデータ型ノードに接続します。
3. URL Parser ノードを編集します。「ID フィールド」ドロップダウン・リストで、「**ReturnedContentSize**」 を選択します。「URL フィールド」ドロップダウン・リストで、「**ReferralURL**」 を選択します。「**OK**」をクリックします。
4. テーブル・ノードを URL Parser ノードに接続してストリームを実行します。**ReturnedContentSize** および **ReferralURL** フィールドは、次の 4つの生成されたフィールドへ追加で解析した **ReferralURL** と一緒に表示されます。**ReferralURL_server**、**ReferralURL_path**、**ReferralURL_field** および **ReferralURL_value**です。

ドキュメント・ビルダー・ノード (Web Status Report)

ドキュメント・ビルダー・ノードのサンプルは、Web サーバー・ログから渡されたデータを読み込み、HTML ファイルの形式でレポートを生成します。レポートは、さまざまな HTTP ステータス・コード (200、302、404 など) を戻すログ・レコードの割合を示すテーブルで構成されています。.

Web Status Report ノードを使用する手順は、次のとおりです。

1. 最初のサンプルからのストリームが閉じられている場合は、開きます。これは、Apache Log Reader とデータ型ノードが含まれているストリームです。ストリームに 2 番目のサンプルの URL Parser がある場合、そのノードはこのサンプルでは無視されます。
2. ノード・パレットの「出力」タブで、Web Status Report ノードをデータ型ノードに接続します。
3. Web Status Report ノードを編集します。「ステータス・コード」ドロップダウン・リストで「**StatusCode**」を選択します。「実行」をクリックします。「実行」 をクリックします。出力ウィンドウで、レポートの内容が表示されます。

モデル・ビルダー・ノード (Interaction)

モデル・ビルダー・ノードのサンプルでは、他のサンプルとは独立して動作し、標準的 (非対話式) な方法で単純なモデルを構築、または生成される前にモデルを対話処理できます。このモデルは、通信会社の顧客離れの予想を試みます。

注**:** このサンプルでは、Windows 固有の API 呼び出しを使用して、スレッドの作成と管理を行います。したがって、Windows 以外のプラットフォームではサポートされません。

Interaction ノードを使用するには

1. CLEF サンプルがまだ有効化されていない場合は、有効化します。 詳しくは、 25 ページの『サンプルの有効化』を参照してください。
2. IBM SPSS Modeler の新規ストリームを作成します。
3. *Demos* ディレクトリーからファイル *telco.sav* をインポートする Statistics ファイル入力ノードを追加します。
4. 「データ型」タブで、「値の読み込み」 をクリックした後、メッセージ・ボックスの 「**OK**」 をクリックして確定します。
5. 「解約」 フィールド (リスト内の最後のフィールド) の役割を 「対象」 に設定し、「**OK**」 をクリックします。
6. ノード・パレットの「モデル」タブで、Interaction ノードを入力ノードに接続します。

標準 (非対話式) モデル構築をテストするには

1. ストリームを実行してストリーム内、そして画面右上の「モデル」パレットにモデル・ナゲットを作成します。
2. テーブルノードをモデル・ナゲットに接続します。
3. テーブル・ノードを実行します。テーブル出力ウィンドウの右側にスクロールし、解約予測を表示します。フィールド `$I-churn` には予測値が表示されますが、`$IP-churn` では予測の確信度 (0.0 ～ 1.0 が表示されます。

対話式モデル構築をテストするには

1. Interaction モデル・ビルダーのダイアログ・ボックスの「モデル」タブで、「インタラクティブ セッションを起動する」 を選択します。

2. 「実行」 をクリックして、「対話検定」ダイアログ・ボックスを表示します。
3. 「対話検定」ダイアログ・ボックスで、「構築タスクの開始」 をクリックしてモデル構築の進捗状況を表示します。
4. モデル構築の操作が終了した場合、ダイアログ・ボックスの「構築タスク」テーブルに追加された行を選択します。
5. ダイアログ・ボックス上部のツールバー領域で、黄色のダイアモンド型アイコンのボタンをクリックします。これにより、モデル出力オブジェクト (**model_1**) が画面右上の「モデル」パレットに生成されます。

対話的に生成されたモデルは、最初のモデルと同じですが、名前は異なります。「構築タスクの開始」 からのプロセスを繰り返すと、**model_2** などの同じモデルをさらに生成します。

使用ファイルの検査

CLEF の動作方法を理解するには、提供されたサンプルの設定ファイルを検証することが良い方法です。設定ファイルは、次の場所にあります。

install_dir¥ext¥lib¥*extension_folder*¥extension.xml

この場合、*install_dir* は IBM SPSS Modeler インストール・ディレクトリーで、*extension_folder* が、次のいずれかになります。

- *spss.apachelogreader*
- *spss.interaction*
- *spss.urlparser*
- *spss.webstatusreport*

¥ext¥lib の下に他の拡張フォルダーが表示される場合がありますが、これらは、CLEF を使用して作成されたシステム提供の IBM SPSS Modeler ノードに関連します。これらのノードがインストールに含まれるかどうかは、ライセンス許可された IBM SPSS Modeler モジュールによって異なります。設定ファイル全体で強調表示されている場合がありますが、これらのファイルを決して変更しないでください。変更した場合、これらのノードは正しく機能しない場合があります。その場合には、関連する IBM SPSS Modeler 製品を再インストールする必要があります。システムに提供されたファイルへの変更は、IBM Corp. ではサポートされていません。

ソース・コードの検査

参照用に、サンプル・ノードの完全なソース・コードも提供されています。すべてのサンプル・ノードでは、C++ のサーバー側ライブラリーを使用しますが、Interaction ノードのみ、クライアント側の Java クラスも使用します。

ソース・コード・ファイルは、サンプルを有効にすると自動的に抽出され、以下の表に示すようにインストールされます。

表 *11.* ソース・コード・ファイルのインストール

場所	内容
...¥ext¥lib¥spss.interaction¥src	親フォルダーの `ui.jar` ファイル内の `.class` ファイルの Java ソース・コード

表 11. ソース・コード・ファイルのインストール (続き)

場所	内容
...¥ext¥bin¥spss.apachelogreader¥src *...¥ext¥bin¥spss.interaction¥src* *...¥ext¥bin¥spss.urlparser¥src* *...¥ext¥bin¥spss.webstatusreport¥src*	親フォルダー内の DLL の C++ ソースおよびプロジェクト・ファイル

サンプルの削除

IBM SPSS Modeler のサンプル・ノードを今後参照しない場合、次の手順で削除できます。

1. IBM SPSS Modeler を終了します。
2. IBM SPSS Modeler インストール・ディレクトリー内の *¥ext¥bin* および *¥ext¥lib* ディレクトリーからサンプル・フォルダーを削除します。標準 IBM SPSS Modeler フォルダーを誤って削除しないようにしてください。削除すると、関連する IBM SPSS Modeler 製品を再インストールする必要があります。削除するフォルダーは次のとおりです。

- *spss.apachelogreader*
- *spss.urlparser*
- *spss.webstatusreport*
- *spss.interaction*

変更は、次回 IBM SPSS Modeler を起動した場合に有効になります。

第 4 章 仕様ファイル

仕様ファイルの概要

すべての CLEF の拡張子には、すべての拡張子の特性を定義する XML ファイルが含まれている必要があります。このファイルは仕様ファイルと呼ばれ、名前は常に `extension.xml` です。仕様ファイルは、次の項で構成されています。

- **XML** 宣言:XML バージョンおよびそのほかの情報のオプションの宣言。
- **Extension** 要素:ファイルの主要部分で、後続の項をすべて含みます。
- **Extension Details** セクション:拡張子に関する基本情報を指定します。
- **Resources** セクション:リソース・バンドル、JAR ファイル、共有ライブラリーなど、拡張子が機能するために必要が外部リソースを指定します。
- **Common Objects** セクション:(オプション) モデル、ドキュメント、プロパティー・タイプなど、拡張子の他のオブジェクトで使用または参照される項目を定義します。
- **User Interface (Palettes)** セクション:(オプション) ノードが表示されるカスタム・パレットまたはサブパレットを定義します。
- **Object Definition** セクション:ノード、モデル出力、ドキュメント出力など、拡張子で定義されるオブジェクトを識別します。

各セクションには、(要素のコンポーネントなど) 静的な宣言、または (ノードの出力データモデルの計算など) シンプルな動的プロセスのいずれか、または両方が含まれています。CLEF 仕様ファイルの全体の形式は、次のようになります。

```
<?xml version="1.0" encoding="UTF-8" ?>
<Extension ... >
   <ExtensionDetails ... />
   <Resources
      Resources section
   </Resources>
   <CommonObjects>
      Common Objects section
   </CommonObjects>
   <UserInterface>
      User Interface (Palettes) section
   </UserInterface>
   Object Definition section
     object definition
     object definition
     object definition
    ...
</Extension>
```

コメント行

仕様ファイルの任意のポイントで、次の形式のコメント行を含むことができます。

```
<!- comment text ->
```

必須かオプションか

後続のセクションの要素の定義 (通常は「フォーマット」という見出しが付いています) では、「(必須)」と記載されていない限り、要素の属性と子要素はオプションです。要素の詳細な構文については、 209 ページの『CLEF XML スキーマ』を参照してください。

仕様ファイルの例

ここに単純なデータ変換ノードの場合の CLEF 仕様ファイルの完全な例を示します。

```
<?xml version="1.0" encoding="UTF-8"?>
<Extension version="1.0" debug="true">
     <ExtensionDetails id="urlparser" providerTag="spss" label="URL CLEF Module" version="1.0"
provider="IBM Corp." copyright="(c) 2005-2011 IBM Corp." description="A Url Transform CLEF Extension"/>
     <Resources>
          <SharedLibrary id="urlparser_library" path="spss.urlparser/urlparser" />
     </Resources>
     <Node id="urlparser_node" type="dataTransformer" palette="fieldOp" label="URL Parser">
          <Properties>
               <Property name="id_fieldname" valueType="integer" label="ID field" />
               <Property name="url_fieldname" valueType="string" label="URL field" />
          </Properties>
          <UserInterface>
               <Icons />
               <Tabs>
                    <Tab label="Types" labelKey="optionsTab.LABEL">
                         <PropertiesPanel>
                              <SingleFieldChooserControl property="id_fieldname"  storage="integer" />
                              <SingleFieldChooserControl property="url_fieldname"  storage="string" />
                         </PropertiesPanel>
                    </Tab>
               </Tabs>
               <Controls />
          </UserInterface>
          <Execution>
               <Module libraryId="urlparser_library" name="">
                    <StatusCodes>
                         <StatusCode code="0" status="error" message="Cannot initialise a peer" />
                         <StatusCode code="1" status="error" message="error reading input data" />
                         <StatusCode code="2" status="error" message="Internal Error" />
                         <StatusCode code="3" status="error" message="Input Field Does Not Exist" />
                    </StatusCodes>
               </Module>
          </Execution>
          <OutputDataModel mode="replace">
               <AddField name="${id_fieldname}" fieldRef="${id_fieldname}"/>
               <AddField name="${url_fieldname}" fieldRef="${url_fieldname}"/>
               <AddField name="${url_fieldname}_server" storage="string" />
               <AddField name="${url_fieldname}_path" storage="string" />
               <AddField name="${url_fieldname}_field" storage="string" />
               <AddField name="${url_fieldname}_value" storage="string" />
          </OutputDataModel>
     </Node>
</Extension>
```

ExtensionDetails 要素は、IBM SPSS Modeler によって内部で使用される拡張子についての基本情報を提供します。

Resources 要素は、ファイルで後で参照されるサーバー側のライブラリーの場所を指定します。パスの指定により、ライブラリーが IBM SPSS Modeler インストール・ディレクトリーの ¥ext¥bin¥spss.urlparser¥urlparser.dll に存在することが指定されます。

この仕様ファイルには CommonObjects 要素は含まれません。

Node 要素はノード自体のすべての情報を指定します。

- Properties では、ノードのダイアログのタブで後で使用する 2 つのプロパティーが最初に宣言されます。
- UserInterface 要素は、この拡張子に固有のノード・ダイアログのタブの外観およびレイアウトを定義します (他のタブは IBM SPSS Modeler で提供されます)。
- Execution 要素は、ノード実行時に使用される項目を定義します。この場合、これらの項目はファイル内で最初に宣言されたサーバー側のライブラリーで、実行によって特定のステータス・コードを返す場合に表示される一連のメッセージです。
- OutputDataModel 要素は、このノードが実行するデータ変換を定義します。入力データ・モデル (このノードに入力されるフィールドのセット) がここで定義されるフィールドのセットと置き換えられるよう定義され、出力データ・モデル (モデルが継続して変更されない場合、ここより下流のすべてのノードに渡されるフィールドのセット) を構築します。この例のノードは、2 つの元のフィールド (id_fieldname と url_fieldname) を変更せずに通過しますが、url_fieldname から取得した名前を持つ 4 つのフィールドを追加します。

この仕様ファイルは、IBM SPSS Modeler インストールの一部として提供されたノードの例の 1 つから取得されます。 詳しくは、トピック 27 ページの『データ変換ノード (URL Parser)』を参照してください。

XML 宣言

XML 宣言はオプションで、文字エンコードの形式の詳細と共に、使用されている XML のバージョンを指定します。

例

```
<?xml version="1.0" encoding="UTF-8" ?>
```

Extension 要素

Extension 要素は、ファイルの主要な部分を構築し、その他のすべてのセクションが含まれます。形式は次のとおりです。

```
<Extension version="version_number" debug="true_false">
     Extension Details section
     Resources section
     Common Objects section
     User Interface (Palettes) section
     Object Definition section
</Extension>
```

ここで、

version は拡張子のバージョン番号です。

debug はオプションです、true に設定されている場合、「デバッグ」タブをダイアログまたは CLEF ノードまたは出力に関連するフレームに追加し、オブジェクトに定義されたプロパティーおよびコンテナーにアクセスすることができます。デフォルト値は false です。詳しくは、トピック 206 ページの『「デバッグ」タブの使用』を参照してください。

Extension Details セクション

Extension Details セクションでは、拡張子に関する基本情報を提供します。

フォーマット

```
<ExtensionDetails providerTag="extension_provider_tag"
     id="extension_unique_identifier"
     label="display_name" version="extension_version_number"
     provider="extension_provider" copyright="copyright_notice"
     description="extension_description"/>
```

ここで、

providerTag (必須) は、拡張子のプロバイダを一意に識別する名前です。値は、文字列 spss を含むことはできません。内部での使用に保持されます。

id (必須) はこの拡張子を一意に識別する名前で、それに関するシステム メッセージに使用されます。規約により、拡張ファイルは IBM SPSS Modeler インストール・ディレクトリーの ¥ext¥lib¥*providerTag.id* というフォルダーに置かれます。

label (必須) は、拡張子の表示名です。このテキストは、ノードが追加された場合にパレット・マネージャの「名前」フィールドに表示されます。 詳しくは、トピック 205 ページの『CLEF 拡張のテスト』を参照してください。

version は拡張子のバージョン番号です。

provider は、この拡張子のプロバイダを識別します。このテキストは、ノードが追加された場合にパレット・マネージャの「プロバイダ」フィールドに表示されます。デフォルトは、文字列 (unknown) です。

copyright は、拡張子の著作権の通知です。このテキストは、ノードが追加された場合にパレット・マネージャの「著作権」フィールドに表示されます。

description は、拡張の目的の簡単な説明です。このテキストは、ノードが追加された場合にパレット・マネージャの「説明」フィールドに表示されます。

例

```
<ExtensionDetails providerTag="myco" id="sorter" name="Sort Data" version="1.2"
     provider="My Company Inc." copyright="(c) 2005-2006 My Company Inc."
     description="An example extension that sorts data using built-in OS commands."/>
```

Resources セクション

このセクションは、拡張子が機能するために必要な外部リソースを定義します。

フォーマット

```
<Resources>
     <Bundle .../>
      ...
     <JarFile .../>
      ...
     <SharedLibrary .../>
      ...
     <HelpInfo .../>
</Resources>
```

ここで、

Bundle は、一連のクライアント側のローカライズされたリソースを識別します。 詳しくは、トピック『バンドル』を参照してください。

JarFile は、クライアント側の Java jar ファイルを識別します。 詳しくは、トピック『Jar ファイル』を参照してください。

SharedLibrary は、サーバー側のライブラリーまたは DLL を識別します。 詳しくは、トピック 36 ページの『共有ライブラリー』を参照してください。

HelpInfo は、拡張子のヘルプ情報があれば、その種類を指定します。 詳しくは、トピック 167 ページの『ヘルプ・システムの実装』を参照してください。

例

```
<Resources>
     <SharedLibrary id="discriminantnode" path="spss.xd/Discriminant"/>
     <Bundle id="translations.discrim" type="properties" path="messages"/>
     <JarFile id="java" path="discriminant.jar"/>
     <HelpInfo id="help" type="native"/>
</Resources>
```

バンドル

Bundle 要素は、.properties ファイルまたは Java .class ファイルのいずれかとして実装される、クライアント側のリソース・バンドルを指定します (ローカライズのメッセージ テキストなど)。 詳しくは、トピック 171 ページの『ローカライゼーション』を参照してください。

フォーマット

```
<Bundle id="identifier" path="path"/>
```

ここで、

id (必須) は、このバンドルの一意の識別子です。

path (必須) は、この仕様ファイルの親フォルダーに関連するバンドル・ファイルの場所を指定します。バンドルが .properties ファイルを参照する場合、パスには言語の拡張子または .properties 接尾辞が含まれる必要があります。

例

```
<Bundle id="translations.discrim" path="messages"/>
```

これは、リソース・バンドルが仕様ファイルと同じフォルダーの messages.properties というファイルにあることを示します。

Jar ファイル

JarFile 要素は、この拡張の Java クラスや他のクライアント側のリソースを提供するクライアント側の Java アーカイブ (.jar) ファイルを指定します。

フォーマット

```
<JarFile id="identifier" path="path"/>
```

ここで、

id (必須) は、この .jar ファイルの一意の識別子です。

path (必須) は、.jar ファイルの親フォルダーに関連するバンドル・ファイルの場所を指定します。

例

```
<JarFile id="java" path="coxreg_model_terms.jar"/>
```

これは、この拡張の .jar ファイルが仕様ファイルと同じフォルダーにあることを示します。

共有ライブラリー

SharedLibrary は、サーバー側の共有ライブラリーまたは DLL を識別します。これは通常、ノード実行のサポートにのみ必要です。ライブラリーが複数のモジュールを実装する場合、ノード設定の Execution セクションの Module 要素は、ライブラリー内の特定のモジュールを識別します。

フォーマット

```
<SharedLibrary id="identifier" path="path"/>
```

ここで、

id (必須) は、この共有ライブラリーの一意の識別子です。

path (必須) は、サーバー側のインストール・ディレクトリーの ¥ext¥bin フォルダーに関連する共有ライブラリーの場所を指定します。パスは、共有ライブラリー・ファイルの拡張子 (.dll など) を含むことはできません。

例

次の共有ライブラリーの宣言

```
<SharedLibrary id="binning" path="spss.binning/Binning" />
```

は、共有ライブラリーが次の場所からロードされることを指定します。

install_dir¥ext¥bin¥spss.binning¥Binning.dll

この場合、*install_dir* はサーバー側の CLEF コンポーネントがインストールされているディレクトリーを示します。このライブラリーは複数のモジュールを実装しているため、必要な特定のモジュール (supervisedBinning) は構築ノードの設定内の Module 要素によって識別され、次のライブラリーの識別子を参照します。

```
<Execution>
     <Module libraryId="binning" name="supervisedBinning" .../>
 ...
</Execution>
```

ヘルプ情報

オプションの HelpInfo 要素は、この拡張に提供することができるヘルプの種類を示します。 詳しくは、トピック 167 ページの『ヘルプ・システムの実装』を参照してください。

Common Objects セクション

オプションの Common Objects セクションは、設定ファイルの別の場所で定義された要素間で共有できるオブジェクトを定義しますこのセクションのオブジェクト (プロパティーの列挙など) の中には、必要な場所でローカルに定義される場合もあれば、(モデルやドキュメントなど) ここで定義される場合もあります。

フォーマット

```
<CommonObjects>
     <PropertyTypes .../>
     <PropertySets .../>
     <ContainerTypes .../>
     <Actions .../>
     <Catalogs .../>
</CommonObjects>
```

ここで、

PropertyTypes を使用すると、共通のプロパティー定義をオブジェクト間で共有することができます。 詳しくは、トピック『プロパティー・タイプ』を参照してください。

PropertySets は通常、ビルダー・ノード、モデル出力オブジェクトおよびモデル・アプライヤー・ノードに同じセットのプロパティーが含まれている場合に使用されます。 詳しくは、トピック 38 ページの『プロパティー・セット』を参照してください。

ContainerTypes は、複雑なデータ構造をラップすることができるオブジェクトである、コンテナー・タイプを定義します。 詳しくは、トピック 39 ページの『コンテナー・タイプ』を参照してください。

Actions は、メニューまたはツールバーによってなど、ユーザーとの対話に関する基本情報を定義します。詳しくは、トピック 41 ページの『アクション』を参照してください。

Catalogs は、サーバーが動的に生成する値のリストから 1 つまたは複数のオプションを選択できるコントロールを実装します。詳しくは、トピック 42 ページの『カタログ』を参照してください。

例

```
<CommonObjects>
     <ContainerTypes>
          <ModelType id="discriminant_model" format="utf8" />
          <DocumentType id="html_output" />
          <DocumentType id="zip_outputType" format="binary"/>
     </ContainerTypes>
</CommonObjects>
```

プロパティー・タイプ

オプションの Property Types セクションによって、共通するプロパティー定義をオブジェクト間で共有することができます。これにより、例えばプロパティーの定義を複数の場所で複製するのではなく、1 つの場所に表示することができるなど、メンテナンスしやすくなる場合があります。定義を共有すると、オブジェクトの新しいインスタンスが作成された場合に値がコピーされるさまざまなオブジェクトのプロパティー間の互換性を実現することもできます。

プロパティー・タイプは、Common Objects セクションでのみ定義することができます。

フォーマット

```
<PropertyTypes>
     <PropertyType id="identifier" isKeyed="true_false" isList="true_false" max="max_value"
       min="min_value" valueType="value_type">
          <Enumeration ... />
          <Structure ... />
          <DefaultValue ... />
     </PropertyType>
     <PropertyType ... />
      ...
</PropertyTypes>
```

PropertyType 属性は以下のとおりです。

id (必須) は、プロパティー・タイプの一意の識別子です。

isKeyed が true に設定されている場合、プロパティー・タイプがキーであることを示します。キー・プロパティーは、ユーザー定義の制御によって、一連の操作をフィールドを関連付けます （ 143 ページの『プロパティー・コントロール』を参照)。isKeyed が true に設定されているときに、valueType 属性は structure に設定されていなければなりません。構造化プロパティーの詳細は、 63 ページの『構造化プロパティー』を参照してください。

isList は、プロパティーが指定された値の種類の値のリスト (true) であるか、または単一の値 (false) であるかを指定します。

max および min は、範囲の最大値および最小値を示します。

valueType は、次のいずれかになります。

- string
- encryptedString
- fieldName
- integer
- double
- boolean
- date
- enum (62 ページの『列挙型プロパティー』を参照)
- structure (63 ページの『構造化プロパティー』を参照)
- databaseConnection

子要素の Enumeration と Structure を同時に指定することはできません。子要素の Enumeration、Structure、DefaultValue は、特定の場合に使用します。 62 ページの『列挙型プロパティー』、 63 ページの『構造化プロパティー』、 64 ページの『デフォルト値(D)』を参照してください。

プロパティー・セット

Property sets は通常、ビルダー・ノード、モデル出力オブジェクトおよびモデル・アプライヤー・ノードに同じセットのプロパティーが含まれている場合に使用されます。例えば、モデル・ビルダー・ノードはビルダーで設定できるプロパティーのデフォルト・セットを定義することができますが、実際はモデル・アプリケーションまで使用されません。自動的に転送するために、モデル出力にも含まれる必要があります。

フォーマット

```
<PropertySets>
     <PropertySet id="identifier">
          <Property ... />
          <Property ... />
           ...
     </PropertySet>
       ...
</PropertySets>
```

id は、このプロパティー・セットの一意の識別子です。

Property 要素の詳細は、 52 ページの『Properties』を参照してください。

例

この例では、2 つのプロパティー (作成する予測の数、確率を含めるかどうか) の定義を示します。Common Objects セクションで、次のように定義します。

```
<PropertySets>
     <PropertySet id="common_model_properties">
          <Property name="prediction_count" valueType="integer" min="1" max="10"/>
          <Property name="include_probabilities" valueType="boolean" defaultValue="false"/>
     </PropertySet>
       ...
</PropertySets>
```

モデル・ビルダー・ノード、モデル出力オブジェクト、モデル・アプライヤー・ノードのそれぞれの定義に、次のような includePropertySets 属性が含まれます (モデル・ビルダー・ノードのみの定義を示します)。

```
<Node id="my_builder" type="modelBuilder" ... >
     <Properties includePropertySets="[common_model_properties]">
      ...
     </Properties>
      ...
</Node>
```

コンテナー・タイプ

コンテナーは、モデルやドキュメントなどの複雑なデータ構造のプレースホルダとして動作するオブジェクトです。コンテナーは特定の種類として定義され、コンテナー・タイプはここで定義されます。次のコンテナー・タイプを定義することができます。

- モデルの種類
- ドキュメント・タイプ

コンテナー・タイプはクライアントとサーバー間で転送、複製、ファイルまたはコンテンツ・リポジトリーに保存することができます。モデルは、モデル・アプライヤー・ノードがモデル出力オブジェクトから生成された場合に複製されます。

それぞれのコンテナー・タイプには事前定義されたプロパティーのセットが含まれますが、カスタム・プロパティーを追加することができます。コンテナー・タイプは、Common Objects セクションでのみ定義することができます。

フォーマット

Container Types セクションの形式は次のとおりです。

```
<ContainerTypes>
     <ModelType ... />
      ...
     <DocumentType ... />
      ...
</ContainerTypes>
```

ここで、

ModelType は、特定のモデル タイプの形式を指定します。 詳しくは、トピック『モデル・タイプ』を参照してください。

DocumentType は、特定のドキュメント・タイプの形式を指定します。 詳しくは、トピック『ドキュメント・タイプ』を参照してください。

例

```
<ContainerTypes>
     <ModelType id="discriminant_model" format="utf8">
     <DocumentType id="html_output" />
     <DocumentType id="zip_outputType" format="binary"/>
</ContainerTypes>
```

モデル・タイプ

モデルは、アルゴリズム名、モデル タイプ、入力データ・モデルおよび出力データ・モデルなどの情報を提供する必要があります。モデル タイプの定義は、特定のモデル タイプの形式を指定します。

モデル タイプの情報は、仕様ファイルのこの場所で静的に、またはモデルがモデル・ビルダー・ノードで構築されている場合は動的に指定されます。

フォーマット

```
<ModelType id="identifier" format="model_type_format" />
```

ここで、

- id (必須) は、モデル タイプの一意の識別子です。
- format (必須) はモデルの形式を示し、utf8 (テキスト) または binary のいずれかとなります。モデル形式は、静的情報の一部として指定する必要があります。

例

```
<ModelType id="my_model" format="utf8" />
```

ドキュメント・タイプ

document は、グラフまたはレポートなどの出力オブジェクトです。ドキュメント・タイプの定義は、特定のドキュメント・タイプの形式を指定します。

フォーマット

```
<DocumentType id="identifier" format="document_type_format" />
```

ここで、

- id (必須) は、ドキュメント・タイプの一意の識別子です。
- format (必須) はドキュメント・タイプの形式で、utf8 (テキスト) または binary のいずれかです。

例

```
<DocumentType id="html_output" format="utf8" />
<DocumentType id="zip_outputType" format="binary"/>
```

アクション

アクションは、メニューまたはツールバーによってなど、ユーザーとの対話に関する基本情報を定義します。各アクションは、ラベル、ツールヒントまたはアイコンなど、ユーザー・インターフェースにどのように表示されるかを定義します。一連のアクションは、アクションの各グループに定義されるクライアント側の Java クラスに処理されます。また、アクションは特定のオブジェクト内で定義されます。

フォーマット

```
<Actions>
     <Action id="identifier" label="display_label" labelKey="label_key"
     description="action_description" descriptionKey="description_key" imagePath="image_path"
     imagePathKey="image_path_key" mnemonic="mnemonic_char" mnemonicKey="mnemonic_key"
     shortcut="shortcut_string" shortcutKey="shortcut_key" />
      ...
</Actions>
```

ここで、

id (必須) は、アクションの一意の識別子です。

label (必須) は、ユーザー・インターフェースに表示される、アクションの表示名です。

labelKey は、ローカライズ用のラベルを識別します。

description はアクションの説明です。例えば、カスタム メニュー項目またはツールバーのアイコンのアクション・ボタンなど、メニュー項目またはボタンのツールヒントのテキストとなります。

descriptionKey は、ローカライズ用の説明を識別します。

imagePath はアイコン・イメージなど、グラフィック ファイルの場所です。仕様ファイルがインストールされているディレクトリーに関連した場所が指定されます。

imagePathKey は、ローカライズ用のイメージ パスを識別します。

mnemonic は、このコントロールを有効化するために Alt キーと組み合わせて使用するアルファベットの文字です (例えば、値 S を指定すると、ユーザーは Alt + S キーを使用してこのコントロールを有効化できます)。

mnemonicKey は、ローカライズ用の mnemonic を識別します。mnemonic も mnemonicKey も使用されない場合、mnemonic はこのコントロールに使用できません。 詳しくは、トピック 117 ページの『アクセス キーとキーボード・ショートカット』を参照してください。

shortcut は、このアクションを開始するのに使用できるキーボードのショートカットを示す文字列です (例えば、CTRL+SHIFT+A)。

shortcutKey は、ローカライズ用のショートカットを指定します。shortcut も shortcutKey も使用しない場合は、このアクションでショートカットを使用することはできません。詳しくは、トピック 117 ページの『アクセス キーとキーボード・ショートカット』を参照してください。

例

```
<Actions>
     <Action id="generateSelect" label="Select Node..." labelKey="generate.selectNode.LABEL"
      imagePath="images/generate.gif" description="Generates a select node"
      descriptionKey="generate.selectNode.TOOLTIP"/>
     <Action id="generateDerive" label="Derive Node..." labelKey="generate.deriveNode.LABEL"
      imagePath="images/generate.gif" description="Generates a derive node"
      descriptionKey="generate.deriveNode.TOOLTIP"/>
</Actions>
```

カタログ

カタログを使用して、プロパティーと、サーバーで動的に生成された値のリストから複数のオプションを選択できるコントロールを関連付けることができます。

<Select> エントリーをクリックすると、値がコントロールにポップアップ・リストで表示されます。

ユーザーがリスト内の 1 行を選択すると、Catalog で指定された列の行値がコントロールに投入されます。

フォーマット

```
<CommonObjects>
     <Catalogs>
          <Catalog id="identifier" valueColumn="integer">
               <Attribute label="display_name" />
                ...
          </Catalog>
           ...
     </Catalogs>
</CommonObjects>
```

ここで、

id (必須) は、カタログの一意の識別子です。

valueColumn (必須) は、ユーザーが行を選択した場合にコントロールに投入される値を持つ列の数値です。列のナンバリングは 1 から始まります。

下の例のように、列ごとに Attribute 要素を、列の順に使用します。

ユーザーがカタログに関連するコントロールを有効にすると、 getCatalogInformation 関数への呼び出しによって値リストを含むカタログがサーバーから取得されます。この関数は、値リストを含む XML ドキュメントを返します。 詳しくは、トピック 183 ページの『ピア関数』を参照してください。

例

この例では、カタログ・コントロールを定義されるために使用するコードの一部について説明します。3 つのカタログが定義され、ダイアログ・タブの 3 つのさまざまな異なるコントロールと関連します。

まず、カタログは Common Objects セクションで定義されます。

```
<CommonObjects>
     <Catalogs>
          <Catalog id="cat1" valueColumn="1">
               <Attribute label="col1" />
               <Attribute label="col2" />
          </Catalog>
          <Catalog id="cat2" valueColumn="2">
```

```
            <Attribute label="col1" />
            <Attribute label="col2" />
            <Attribute label="col3" />
        </Catalog>
        <Catalog id="cat3" valueColumn="1">
            <Attribute label="col1" />
        </Catalog>
    </Catalogs>
</CommonObjects>
```

次は、コントロールと関連するプロパティーが、ノード定義の Properties セクションで定義されます。

```
<Node id="catalognode" type="dataReader" palette="import" label="Catalog">
    <Properties>
        <Property name="sometext" valueType="string" label="Some Text" />
        <Property name="selection1" valueType="string" label="Selection 1" />
        <Property name="selection2" valueType="string" isList="true" label="Selection 2" />
        <Property name="selection3" valueType="string" label="Selection 3" />
    </Properties>
```

ノード定義の User Interface セクションで、コントロールが定義され、プロパティーへの参照によってカタログの定義と関連付けられます。

```
<UserInterface>
    <Tabs>
        <Tab label="Catalog Controls" labelKey="Catalog.LABEL" >
            <PropertiesPanel>
                <TextBoxControl property="sometext" />
                <SingleItemChooserControl property="selection1" catalog="cat1" />
                <MultiItemChooserControl property="selection2" catalog="cat2" />
                <SingleItemChooserControl property="selection3" catalog="cat3" />
            </PropertiesPanel>
        </Tab>
```

User Interface (Palettes) セクション

これはオプションのセクションで、この拡張子でノードが表示されるカスタム・パレットまたはサブパレットを定義する場合にのみ使用します。

拡張子によってカスタム・パレットまたはサブパレットが定義されている場合、同じパレットまたはサブパレットに含めるノードを定義する、後でロードされる拡張子では、User Interface (Palettes) セクションを使用する必要はありません。パレットを参照する customPalette 属性を Node 要素に含めるだけで構いません。拡張子は、*providerTag.id* 値のアルファベット順にロードされます。これらは、この拡張子の ExtensionDetails 要素の providerTag 属性と id 属性の値です (34 ページの『Extension Details セクション』を参照)。例えば、拡張子 myco.abc は拡張子 myco.def の前にロードされます。

注 : User Interface (Palettes) セクションは、メインの User Interface セクションとは異なり、個々のオブジェクトの定義の一部として表示されます。詳細については、 107 ページの『第 6 章 ユーザー・インターフェースの構築』を参照してください。

フォーマット

User Interface (Palettes) セクションの形式は次のとおりです。

```
<UserInterface>
    <Palettes>
        <Palette id="name" systemPalette="palette_name" customPalette="palette_name"
           relativePosition="position" relativeTo="palette" label="display_label"
```

```
            labelKey="label_key" description="description" descriptionKey="description_key"
            imagePath="image_path" />
        <Palette ... />
    ...
    </Palettes>
</UserInterface>
```

表 12. パレットの属性：

属性	説明
id	(必須) 定義するパレットまたはサブパレットの一意の識別子。
systemPalette	システム・パレットにサブパレットを追加する場合にのみ使用され、サブパレットが表示されるシステム・パレットを識別します。 import - ソース recordOp - レコード操作 fieldOp - フィールド操作 graph - グラフ modeling - モデル作成 (下記参照) dbModeling - データベース・モデリング output - 出力 export - エクスポート
customPalette	カスタム・パレットにサブパレットを追加する場合にのみ使用され、サブパレットが表示されるカスタム・パレットを識別します。カスタム・パレットを定義する Palette 要素の id 属性の値です。
relativePosition	カスタム・パレットを定義する場合のみ使用され、画面下部のパレットのストリップの場所を指定します。 指定できる値は以下のとおりです。 first last before after 値が before または after である場合、relativeTo 属性も必須です (下記参照)。 relativePosition が省略されている場合、パレットはストリップの最後に配置されます。
relativeTo	relativePosition の値が before または after の場合、relativeTo を使用して、このカスタム・パレットが先行するまたは後続するパレットの識別子を指定します。パレットの識別子は、Node 要素のパレットの属性値として一覧表示されます (48 ページの『ノード』を参照)。
label	(必須) ユーザー・インターフェースに表示されるパレットまたはサブパレットの表示名。
labelKey	ローカライズ用のラベルを識別します。
description	ツールヒントのテキストは、カーソルがパレット・タブの上にある場合に表示されます (サブパレットには使用されません)。また、この値はコントロールのアクセス可能な詳細な説明としても動作します。 詳しくは、トピック 177 ページの『アクセシビリティー』を参照してください。

表 12. パレットの属性 (続き):

属性	説明
descriptionKey	ローカライズ用の説明を識別します。
imagePath	パレット・タブで使用されるイメージの場所を識別します (サブパレットには使用されません)。仕様ファイルがインストールされているディレクトリーに関連した場所が指定されます。この属性を省略すると、イメージは使用されません。

例 - システム・パレットへのノードの追加

会社がオーディオ・データやビデオ・データをマイニングする新しいアルゴリズムを開発し、アルゴリズムを IBM SPSS Modeler に統合したいと考えているとします。まずオーディオ・ファイルおよびビデオ・ファイルから入力を読み込むカスタム・データ・リーダー・ノードを定義します。

最初に、入力システムパレットに新しいデータ・リーダー・ノードを追加します。必要なことは、Node 要素の palette 属性によって入力パレットを識別することだけです。 詳しくは、トピック 48 ページの『ノード』を参照してください。

入力パレットのデータベース・ノードの後にノードを追加するには、次を使用します。

```
<Node id="AVreader" type="dataReader" palette="import" relativePosition="after"
   relativeTo="database" label="AV Reader">
```

例 - カスタム・パレットの追加

標準 IBM SPSS Modeler パレットの使用は問題ありませんが、新しいノードにより高い重要度を与える必要があるとします。カスタム・パレットを定義し、「お気に入り」パレットの後、「入力」パレットの前に配置します。まず、次のような、カスタム・パレットを定義する User Interface (Palettes) セクションを追加する必要があります。

```
<UserInterface>
     <Palettes>
          <Palette id="AV_mining" label="AV Mining" relativePosition="before"
      relativeTo="import" description="Audio video mining"  />
     </Palettes>
</UserInterface>
```

relativeTo 属性で、「入力」パレットの内部識別子 import を使用する必要があります。

その後、次のように Node 定義を変更します。

```
<Node id="AVreader" type="dataReader" customPalette="AV_mining" label="AV Reader">
```

これにより、「お気に入り」システム・パレットと「入力」システム・パレットの間に 「**AV** マイニング」 パレットを配置します。

例 - カスタム・パレットへのカスタム・サブパレットの追加

前述の例に続き、データ・リーダー・ノードを 「**AV** マイニング」 パレットの 「**AV** ソース」 サブパレットで動作させるとします。これをアーカイブするには、まず User Interface (Palettes) セクションに 2 番目の Palette 要素を追加してサブパレットを指定する必要があります。

```
<UserInterface>
     <Palettes>
          <Palette id="AV_mining" label="AV Mining" description="Audio video mining" />
          <Palette id="AV_mining.sources" customPalette="AV_mining" label="AV Sources" />
     </Palettes>
<UserInterface>
```

その後、サブパレットの識別子を参照する Node 要素を変更します。

```
<Node id="AVreader" type="dataReader" customPalette="AV_mining.sources" label="AV Reader">
```

「**AV** マイニング」 タブをクリックすると、1 つは 「すべて」、もう 1 つは 「**AV** ソース」 というラベルの付いた 2 つのサブパレットが表示されます。AV リーダー・ノードが両方のサブパレットに表示されます。

「**AV** マイニング」 の追加の新しいサブパレットにもう 1 つ新しいノードを追加する場合、新しいノードは 「すべて」 サブパレットと新しいサブパレットの両方に表示されますが、「**AV** ソース」 サブパレットには表示されません。

例 - システム・サブ パレットへのノードの追加

オーディオおよびビデオ ソース・データを処理するには、モデル・ビルダー・ノードを定義します。多くの標準的なサブパレットを含む標準「モデル作成」パレットにモデル・ビルダー・ノードを追加します。次のように指定して、「監視」サブパレットにモデル・ビルダー・ノードを追加し、ニューラル・ノードの直前に配置します。

```
<Node id="AVmodeler" type="modelBuilder" palette="modeling.classification"
   relativePosition="before" relativeTo="neuralnet" label="AV Modeler">
```

ノードは、「モデル作成」パレットの「すべて」サブパレットの同じ関連位置にも追加されます。

例 - システム・パレットへのカスタム・サブパレットの追加

「監視」サブパレットのモデル・ビルダー・ノードの数について考えた場合、ユーザーは新しいノードに簡単に気が付かない場合があると考えました。ノードをより目立つようにする 1 つの方法は、「モデル作成」パレットに独自のサブパレットを追加し、そこにノードを配置することです。

そして次のように User Interface (Palettes) セクションをファイルに追加して、カスタム・サブパレットを定義する必要があります。

```
<UserInterface>
     <Palettes>
          <Palette id="modeling.av_modeling" systemPalette="modeling" label="AV Modeling"
             labelKey="av_modeling.LABEL" description="Contains AV mining-related modeling
             nodes" descriptionKey="av_modeling.TOOLTIP"/>
     </Palettes>
</UserInterface>
```

拡張するシステム・パレットを識別するためには、明示的に systemPalette を指定する必要があります。

ノードのメインの User Interface セクションで、このサブパレットに表示されるよう指定します。

```
<Node id="my.avmodeler" type="modelBuilder" customPalette="modeling.av_modeling"
   label="AV Modeler">
```

カスタム・サブパレットは、常にシステム・サブパレットの後に配置されます。

注: 「AV モデル作成」サブパレットにさらにノードを追加するには、AV Modeler 拡張子が最初にロードされている場合、設定ファイルに User Interface (Palettes) セクションは必要ありません。

カスタム・パレットまたはサブパレットの非表示または削除

カスタム・パレットまたはサブパレットを表示しない場合、IBM SPSS Modeler パレット・マネージャーを使用して、非表示または削除することができます。

非表示の操作は IBM SPSS Modeler セッション全体で反映されますが、チェック・ボックスで制御するように元に戻すことができます。削除の操作は同じセッションで元に戻すことはできませんが、IBM SPSS Modeler を再起動すると、仕様ファイルまたは拡張子全体から削除しない限りは、再度表示されます。 詳しくは、トピック 208 ページの『CLEF 拡張のアンインストール』を参照してください。

パレットを非表示または削除する手順は次のとおりです。

1. メイン IBM SPSS Modeler メニューから次の各項目を選択します。

   「ツール」 > 「パレット管理」
2. 「パレット名」フィールドでパレットを選択し、次の作業を行います。
   - パレットを非表示にするには、対応する「表示 ?」のチェックを解除します。チェック・ボックス
   - パレットを削除するには、「削除選択」ボタンをクリックします。
3. 「OK」 をクリックします。

サブパレットを非表示または削除する手順は次のとおりです。

1. メイン IBM SPSS Modeler メニューから次の各項目を選択します。

   「ツール」 > 「パレット管理」
2. 「パレット名」フィールドでパレットを選択し、次の作業を行います。
3. 「サブパレット」ボタンをクリックします。
4. 「サブパレット名」フィールドでサブパレットを選択し、次の作業を行います。
   - サブパレットを非表示にするには、対応する「表示 ?」のチェックを解除します。チェック・ボックス
   - サブパレットを削除するには、「削除選択」ボタンをクリックします。
5. 「OK」 をクリックします。

Object Definition セクション

この要素は、拡張子で最も目立つ部分です。Object Definition セクションは CLEF 仕様ファイルの残りを構成し、拡張子のさまざまなオブジェクトを定義するために使用されます。次の種類のオブジェクトを定義することができます。

- ノード
- モデル出力オブジェクト
- ドキュメント出力オブジェクト
- インタラクティブ出力オブジェクト

ノードは、ストリームで表示されるオブジェクトです。モデル出力オブジェクトはモデル・ビルダー・ノードで生成され、メイン・ウィンドウのマネージャ領域の「モデル」タブに表示されます。同様に、ドキュメ

ント出力オブジェクトはドキュメント・ビルダー・ノードで生成され、同じ領域の「出力」タブに表示されます。インタラクティブ出力オブジェクトはインタラクティブ・モデル・ビルダー・ノードで生成され、マネージャ領域の「出力」タブに表示されます。

Object Definition セクションは、3 つのオブジェクト定義の 1 つまたは複数で構成されています。

さまざまな種類のオブジェクトに定義することができる要素は、次のセクションで説明されています。これらの要素の中にはすべてのオブジェクト・タイプに共通なものがあれば、ノードまたはモデル出力定義に固有のものもあります。オブジェクト固有の要素は、テキスト内でそのようなものとして示されています。

- オブジェクト識別子
- モデル・ビルダー
- ドキュメント・ビルダー
- プロパティー
- コンテナー
- ユーザーインターフェース
- 実行
- 出力データモデル
- コンストラクター

オブジェクト識別子

オブジェクト識別子は、オブジェクトの種類を示し、次のいずれかが使用されます。

```
<Node .../>

<ModelOutput .../>

<DocumentOutput .../>

<InteractiveModelBuilder .../>
```

また、オブジェクト識別子はスクリプトでオブジェクトがどのように公開されるかについての情報も提供します。scriptName 属性は、オブジェクトの一意の名前を示します。スクリプトでは、この属性を使用して特定のオブジェクトを指定することができます (ストリーム内のノード、「出力」タブの出力など)

ノード

ノード定義は、ストリームに表示されるオブジェクトを説明します。

形式

```
<Node id="identifier" type="node_type" palette="palette" customPalette="custom_palette"
     relativePosition="position" relativeTo="node" label="display_label" labelKey="label_key"
     scriptName="script_name" helpLink="topic_id" description="description"
     descriptionKey="description_key" delegate="Java_class">
     <ModelBuilder ... >
      ...
     </ModelBuilder>
     <DocumentBuilder ... >
      ...
     </DocumentBuilder>
     <ModelProvider ... />
     <Properties>
```

```
      ...
    </Properties>
    <Containers>
      ...
    </Containers>
    <UserInterface>
      ...
    </UserInterface>
    <Execution>
      ...
    </Execution>
    <OutputDataModel ...>
      ...
    </OutputDataModel>
    <Constructors>
      ...
    </Constructors>
</Node>
```

ノード定義内で許可された要素は、 52 ページの『Properties』で始まるセクションで説明されています。

表 *13.* ノード属性：

属性	説明
id	(必須) このノードの識別子。テキスト文字列。
type	(必須) ノードの種類。 dataReader - データを読み込むノード (入力パレット・ノードなど)。 dataWriter - データを書き込むノード (エクスポート・パレット・ノードなど)。 dataTransformer - データを変換するノード (レコード/フィールド設定ノードなど)。 modelBuilder - モデル・ビルダー・ノード (モデル作成パレット・ノード)。 documentBuilder - グラフまたはレポートを作成するノード。 modelApplier - 生成されたモデルを含むノード ノード・タイプは、パレットおよび領域上のノードのアイコンの形状を指定します。詳しくは、トピック 9 ページの『ノードの概要』を参照してください。 ノード・タイプが modelBuilder の場合、ノードの定義には ModelBuilder 要素が含まれている必要があります。 51 ページの『モデル・ビルダー』 を参照してください。 ノード・タイプが documentBuilder の場合、ノードの定義には DocumentBuilder 要素が含まれている必要があります。 51 ページの『ドキュメント・ビルダー』 を参照してください。

表 13. ノード属性 (続き):

属性	説明
palette	ノードが表示される標準 IBM SPSS Modeler パレットまたはサブパレットのいずれかの識別子は、次のとおりです。 import - ソース recordOp - レコード操作 fieldOp - フィールド操作 graph - グラフ modeling - モデル作成 (下記参照) dbModeling - データベース・モデリング output - 出力 export - エクスポート 「モデル作成」パレットには、次のようにさまざまな標準サブパレットがあります。 modeling.classification - 監視 modeling.association - アソシエーション modeling.segmentation - セグメンテーション modeling.auto - 自動 palette 属性を省略すると、ノードは「フィールド設定」パレットに表示されます。 注 :palette 属性は、モデル・ビルダー・ノードにのみ使用されます。
customPalette	ノードが表示されるカスタム・パレットまたはサブパレットの識別子。id 要素の Palette 属性の値で、ファイルのユーザー・インターフェース (パレット) セクションで指定されます。 詳しくは、トピック 43 ページの『User Interface (Palettes) セクション』を参照してください。
relativePosition	パレット内のノードの位置を指定します。指定できる値は以下のとおりです。 first last before after 値が before または after である場合、relativeTo 属性も必須です (下記参照)。 relativePosition を省略した場合、ノードはパレットの最後に配置されます。
relativeTo	relativePosition の値が before または after の場合は、relativeTo を使用して、このノードに先行するパレットまたはこのノードの後続のパレットのノードが指定されます。relativeTo の値は、ノードのスクリプト名です。 標準の IBM SPSS Modeler ノードの場合のスクリプト名は、「*IBM SPSS Modeler* スクリプトとオートメーション・ガイド」の『プロパティー・リファレンス』セクションに記載されています。ただし、...node 接尾辞は使用しません (例えば、データベース・ノードの場合は databasenode ではなく database を使用します)。 CLEF ノードの場合、これはそのノードの scriptName 属性の値です。
label	(必須) パレット、領域、ダイアログに表示されるノードの表示名。
labelKey	ローカライズ用のラベルを識別します。

表 13. ノード属性 (続き) :

属性	説明
scriptName	スクリプトで参照された場合にノードを一意に識別する場合に使用します。 詳しくは、トピック 77 ページの『スクリプトの CLEF ノードの使用』を参照してください。
helpLink	ヘルプ・システムがあれば起動する場合に表示されるヘルプ トピックのオプション識別子です。識別子の形式は、以下のようにヘルプ・システムの種類によって異なります (167 ページの『第 7 章 ヘルプ・システムの追加』を参照)。 HTML Help - ヘルプ トピックの URL JavaHelp - トピック ID
description	ノードのテキストによる説明
descriptionKey	ローカライズ用の説明を識別します。
delegate	指定した場合、NodeDelegate インターフェースを実行する Java クラスの名前が定義されます。指定されたクラスのインスタンスが、関連するノードの各インスタンスに対して構築されます。

ノード定義内で含まれる要素は、『モデル・ビルダー』で始まるセクションで説明されています。

例

ノード定義の例については、 32 ページの『仕様ファイルの例』を参照してください。

モデル出力

モデル出力定義は生成されたモデル (ストリームの実行の後マネージャ領域の「モデル」タブに表示されるオブジェクト) を説明します。

ファイルのこの部分のコード化の詳細は、「 90 ページの『モデル出力』」を参照してください。

ドキュメント出力

ドキュメント出力定義は、ストリームの実行後マネージャ領域の「出力」タブに表示される生成されたテーブルまたはグラフなどのオブジェクトについて説明します。

ファイルのこの部分のコード化の詳細は、「 102 ページの『ドキュメント出力』」を参照してください。

インタラクティブ・モデル・ビルダー

ファイルのこの部分のコード化の詳細は、「 91 ページの『インタラクティブ・モデルの構築』」を参照してください。

モデル・ビルダー

この要素は、*Node* 要素の定義でのみ使用されます。

ファイルのこの部分のコード化の詳細は、「 81 ページの『第 5 章 モデルおよびドキュメントの作成』」を参照してください。

ドキュメント・ビルダー

この要素は、*Node* 要素の定義でのみ使用されます。

ファイルのこの部分のコード化の詳細は、「 81 ページの『第 5 章 モデルおよびドキュメントの作成』」を参照してください。

モデル・プロバイダ

この要素は、*Node* 要素の定義でのみ使用されます。

モデル出力オブジェクトおよびモデル・アプライヤー・ノードを定義する場合、ModelProvider 要素を使用して、モデルを保持するコンテナーを指定することができます。モデルが PMML 形式で保存されるかどうかも指定することができます。PMML モデルは、カスタム ビューアー、または ModelViewerPanel 要素で提供される標準 IBM SPSS Modeler モデル出力ビューアーのいずれかを使用して表示することができます。 詳しくは、トピック 124 ページの『モデル・ビューアー・パネル』を参照してください。

フォーマット

```
<ModelProvider container="container_name" isPMML="true_false" />
```

ここで、

container は、モデルを保持するコンテナーの名前です。

isPMML は、モデルが PMML 形式で保存されるかどうかを指定します。

例

```
<ModelProvider container="model" isPMML="true" />
```

モデル・アプライヤー・ノードのコンテキスト内での 124 ページの『モデル・ビューアー・パネル』 の使用例は、ModelProvider の例を参照してください。

Properties

プロパティーの定義は、名前と値のペアのセットで構成されています。数が多くなることが考えられるそれぞれのプロパティーの定義は、1 つの Properties セクションに含まれています。

注 : プロパティーが Properties セクション内で定義される場合、Properties セクションの定義が優先されるため、個々のプロパティー・コントロールにプロパティーを定義する必要はありません。このため、Properties セクション内でプロパティーを定義することをお勧めします。

この規則の 1 つの例外は、label 属性に関連します。label 属性がプロパティー・コントロールに対して定義されている場合は、(label 定義だけでなく) プロパティー・コントロールの宣言内にあるすべて のプロパティーの定義が、Properties セクション内の対応する定義よりも優先されます。この例外はプロパティー・コントロールのみに適用され、メニュー、メニュー項目、ツールバー項目などほかの種類のコントロールには適用されません。これらは直接的に (メニュー)、または Action 要素を使用して間接的に (メニュー項目およびツールバー項目)、ラベルを明示的に定義する必要があります。

フォーマット

```
<Properties>
     <Property name="name" scriptName="script_name" valueType="value_type" isList="true_false"
          defaultValue="default_value" label="display_label" labelKey="label_key"
          description="description" descriptionKey="description_key" />
          <Enumeration ... />
```

```
        <Structure ... />
        <DefaultValue ... />
    ...
</Properties>
```

Enumeration、Structure および DefaultValue 要素は特定のケースで使用されます。 詳しくは、トピック 61 ページの『値の種類』を参照してください。

Property 要素の属性を以下の表に示します。

表 *14. Property* 属性：

属性	説明
name	(必須) プロパティーの一意の名前。
scriptName	プロパティーがスクリプト内で参照する名前。 詳しくは、トピック 77 ページの『スクリプトの CLEF ノードの使用』を参照してください。
valueType	プロパティーが取得することができる値の種類を指定します。次のいずれかを指定します。 string encryptedString fieldName integer double boolean date enum structure databaseConnection 詳しくは、トピック 61 ページの『値の種類』を参照してください。
isList	プロパティーが、指定された種類の値のリスト (true) か、単一の値 (false) かを指定します。
defaultValue	このプロパティーのデフォルト値です。単一の値の属性または複合要素として示され、指定された有効な値と一貫している必要があります。
label	ユーザー・インターフェースに表示される、プロパティーの値の表示名。
labelKey	ローカライズ用のラベルを識別します。
description	プロパティーの説明。
descriptionKey	ローカライズ用の説明を識別します。

プロパティーは、オプションで有効な値がどのように指定されるのかを宣言します。

- 数値の場合、最小値または最大値となります。
- 文字列の場合、通常はフィールド選択となります (すべてのフィールド、すべての数値型フィールド、すべての離散型フィールドなど) がファイル選択となる場合もあります。
- 列挙する場合、有効な値のセットとなります。

また、キー・プロパティーは有効なキーがどのように指定されるかも宣言する必要があります。キー・プロパティーのキー・タイプは文字列または列挙のいずれかです。 詳しくは、トピック 37 ページの『プロパティー・タイプ』を参照してください。

プロパティーに関連するオプションのデフォルト値は、関連するオブジェクトが作成される場合に評価されます。例えば、ノードプロパティーのデフォルトは、ノードの新しいインスタンスが作成されるごとに評価され、実行プロパティーはノードが実行されるごとに評価されます。評価は、プロパティーが宣言された順番に行われます。

プロパティー定義は、Common Objects セクションで宣言されたプロパティー・タイプを参照する場合があります。

コンテナー

コンテナーは、生成が Constructors セクションで定義される出力オブジェクトのプレースホルダです。

フォーマット

```
<Containers>
     <Container name="container_name" />
      ...
</Containers>
```

ここで、

name は、CreateModel 要素または CreateDocument 要素の target 属性の値に対応し (103 ページの『コンストラクターの使用』を参照)、Common Objects セクションで宣言されたコンテナー・タイプのいずれかに間接的にコンテナーを関連付けます。

例

まず、コンテナー・タイプは Common Objects セクションで宣言されます。モデルについてテキスト形式のコンテナーが 1 つ、ドキュメント出力オブジェクトについて 2 つのコンテナー・タイプ、1 つは HTML 出力のデフォルト (テキスト) 形式、もう 1 つは zip 出力に対するバイナリ形式があります。

```
<CommonObjects>
     <ContainerTypes>
          <ModelType id="my_model" format="utf8" />
          <DocumentType id="html_output" />
          <DocumentType id="zip_outputType" format="binary" />
     </ContainerTypes>
</CommonObjects>
```

ノード定義の Execution セクションで、出力ファイルは Common Objects セクションで指定された識別子に対応するコンテナー・タイプのコンテナー・ファイルとして定義されます。

```
<Node id="mynode" ... >
 ...
     <Execution>
      ...
          <OutputFiles>
               <ContainerFile id="pmml" path="${tempfile}.pmml" containerType="my_model" />
               <ContainerFile id="htmloutput" path="${tempfile}.html" containerType="html_
               output" />
               <ContainerFile id="zipoutput" path="${tempfile}.zip" containerType="zip_
               outputType" />
          </OutputFiles>
```

この後で、Constructors セクションは、ノード実行時に生成される出力オブジェクトを定義します。ここで、CreateModel 要素および CreateDocument 要素には Output Files セクションで指定されたコンテナー・ファイルに対応する sourceFile 属性があります。

```
          <Constructors>
               <CreateModelOutput type="myoutput">
                    <CreateModel target="model" sourceFile="pmml" />
                    <CreateDocument target="advanced_output" sourceFile="htmloutput" />
                    <CreateDocument target="zip_output" sourceFile="zipoutput" />
               </CreateModelOutput>
          </Constructors>
     </Execution>
</Node>
```

最後に Model Output セクションで、コンテナーをモデル出力オブジェクトまたはドキュメント出力オブジェクトと関連付けます。Container 要素の name 属性は、指定された CreateModel 要素と CreateDocument 要素の target 属性に対応します。

```
<ModelOutput id="myoutput" label="My Model">
     <Containers>
          <Container name="model" />
          <Container name="advanced_output" />
          <Container name="zip_output" />
     </Containers>
      ...
</ModelOutput>
```

ユーザー・インターフェース

仕様ファイルは、さまざまなユーザー・インターフェース・コンポーネントをサポートし、オブジェクトを表示したり、コントロールやプロパティーを変更することができます。コンポーネントのレイアウトやサイズ変更の操作、ほかのコントロールが変更された場合にコンポーネントを有効化するかまたは表示するかを指定する機能が提供されています。

User Interface セクションは、オブジェクトの外観を指定します。設定を使用して、ノード・プロパティーのダイアログまたは出力ウィンドウなど、基本のユーザー・インターフェース・コンポーネントをカスタマイズすることができます。

User Interface セクションは、Node 要素設定の必要な部分です。

ファイルのこの部分のコード化の詳細は、「 107 ページの『第 6 章 ユーザー・インターフェースの構築』」を参照してください。

実行

この要素は、*Node* 要素の定義でのみ使用されます。

Execution セクションは、ノード実行時に使用されるプロパティーおよびファイルを定義します。

フォーマット

```
<Execution>
     <Properties>
      ...
     </Properties>
     <InputFiles>
          <ContainerFile ... />
           ...
     </InputFiles>
     <OutputFiles>
          <ContainerFile ... />
           ...
```

```
    </OutputFiles>
    <Module ... >
        <StatusCodes ... />
    </Module>
    <Constructors ... />
</Execution>
```

Execution セクションには、ノードが実行されるごとに再度作成されるプロパティーおよびノード実行中にのみ使用できるプロパティーのセットの定義が含まれます。

実行の情報は、ノードの実行前に生成される入力ファイルのセット、実行中に生成される出力ファイルを定義することもできます。

いくつもの入力ファイルおよび出力ファイルを指定することができます。各入力ファイルは、ノードで定義されたコンテナーと関連しています。各出力ファイルは、通常生成されたオブジェクトのコンテナーを構築するために使用されます。入力ファイルまたは出力ファイルの形式は、Common Objects セクションのコンテナーの宣言によって決まります。

例

Execution セクションの例については、 32 ページの『仕様ファイルの例』を参照してください。

Properties (ランタイム)

このセクションは、ノード実行時にのみ使用できるランタイム・プロパティーのセットを定義します。

フォーマット

形式は、要素定義の主要部分にある Properties セクションの定義に類似しています。 詳しくは、トピック 52 ページの『Properties』を参照してください。

モデル・ビルダー・ノードまたはドキュメント・ビルダー・ノードの実行中、サーバー一時ファイル が作成され、モデル出力オブジェクトまたはドキュメント出力オブジェクトを保存します。サーバーは一時ファイルにアクセスし、オブジェクトをコンテナーにラップされるクライアントに取り込みます。一時ファイルはここで指定する必要があります。

例

この例では、サーバー一時ファイルの指定方法を説明します。

```
<Properties>
    <Property name="tempfile" valueType="string">
        <DefaultValue>
            <ServerTempFile basename="datatmp"/>
        </DefaultValue>
    </Property>
</Properties>
```

入力ファイル

このセクションでは、ノードの実行前に生成される入力ファイルのセットを定義します。このコンテキストの入力ファイルは、サーバー上でのノード実行時に入力されるファイルです。例えば、モデル・アプライヤー・ノードには、ノードの実行時に指定の入力ファイルに転送されるモデル・コンテナーがあります。

フォーマット

```
<InputFiles>
     <ContainerFile id="identifier" path="path" container="container">
      ...
</InputFiles>
```

入力ファイルの ContainerFile 要素の属性は次の表に示されているとおりです。

表 *15.* コンテナー・ファイルの属性 - 入力ファイル :

属性	説明
id	コンテナー・ファイルの一意の識別子。
path	入力ファイルを生成するサーバー上の場所 (例えば、サーバー一時ファイルの場所 - 56 ページの『Properties (ランタイム)』を参照)。
container	サーバーに入力として送信されるオブジェクトを保持するコンテナーの識別子。

例

```
<InputFiles>
     <ContainerFile id="pmml" path="${tempfile}.pmml" container="model"/>
</InputFiles>
```

出力ファイル

このセクションでは、サーバー上のノードの実行中に生成される出力ファイルを指定します。(モデル・ビルダー・ノードまたはドキュメント・ビルダー・ノードの結果など) 出力ファイルは、実行後にクライアントに転送されます。

フォーマット

```
<OutputFiles>
     <ContainerFile id="identifier" path="path" containerType="container">
      ...
</OutputFiles>
```

ContainerFile 要素の属性は次の表示示されているとおりです。

表 *16.* コンテナー・ファイルの属性 - 出力ファイル :

属性	説明
id	コンテナー・ファイルの一意の識別子。
path	クライアントに転送されるオブジェクトのサーバー上の場所 (例えば、サーバー一時ファイルの場所 - 56 ページの『Properties (ランタイム)』を参照)。
containerType	オブジェクトのコンテナー・タイプ (モデル タイプまたはドキュメント・タイプの ID など) の識別子。オブジェクトを適切な形式で転送することができます。詳しくは、トピック 39 ページの『コンテナー・タイプ』を参照してください。

例

```
<OutputFiles>
     <ContainerFile id="pmml" path="${tempfile}.pmml" containerType="mynode_model" />
     <ContainerFile id="htmloutput" path="${tempfile}.html" containerType="html_output" />
     <ContainerFile id="zipoutput" path="${tempfile}.zip" containerType="zip_outputType" />
</OutputFiles>
```

モジュール

このセクションは、ノード実行中に使用されるサーバー側の共有ライブラリーを指定します (例えば、メモリーにロードされる DLL など)。

フォーマット

```
<Module libraryId="shared_library_identifier" name="node_name">
     <StatusCodes ... />
</Module>
```

ここで、

libraryId は、Resources セクションの Shared Library 要素で宣言されるライブラリーの識別子です。詳しくは、トピック 36 ページの『共有ライブラリー』を参照してください。

name は、ライブラリーが複数のノードで共有されている場合に使用され、実行される特定のノードを識別します。ライブラリーが 1 つのノードにのみ使用される場合、name は空白にします。

例

```
<Module libraryId="mynode1" name="mynode">
     <StatusCodes>
          <StatusCode code="0" status="error" message="An exception occurred" />
          <StatusCode code="1" status="error" message="Error reading input data" />
           ...
     </StatusCodes>
</Module>
```

ステータス・コード

ほとんどのプログラムは何らかのエラー・チェックを実行してユーザーに必要なメッセージを表示します。通常、成功またはそのほかのステータスであることを示す整数を返します。サーバー側の API は、ノードを含むストリームの実行後にステータス・コードを返すことができます。 詳しくは、トピック 198 ページの『状況詳細ドキュメント』を参照してください。

Status Codes セクションでは、メッセージを特定のステータス・コードと関連付け、ユーザーに表示することができます。

フォーマット

```
<StatusCodes>
     <StatusCode code="codenum" status="status" message="message_text"
       messageKey="message_key" />
      ...
</StatusCodes>
```

ステータス・コードの属性は、次の表のとおりです。

表 17. ステータス・コードの属性：

属性	説明
code	メッセージが関連するステータス・コード (整数値)。
status	次のような、ステータスの分類です。 success - ノードが正常に実行 warning - ノードの実行に警告 error - ノードの実行が失敗
メッセージ	ステータス・コードが返される場合に表示されるメッセージ

表 17. ステータス・コードの属性 (続き):

属性	説明
messageKey	ローカライズ用のメッセージを識別します。

例

この例では、エラー・メッセージのテキストに StatusCode 要素が含まれています。

```
<StatusCodes>
     <StatusCode code="0" status="error" message="Cannot initialise a peer" />
     <StatusCode code="1" status="error" message="Error reading input data" />
     <StatusCode code="2" status="error" message="Internal Error" />
     <StatusCode code="3" status="error" message="Input Field Does Not Exist" />
</StatusCodes>
```

実行時にサーバー側の API がステータス・コード 3 を返した場合は、ユーザーに対して以下のメッセージが表示されます。

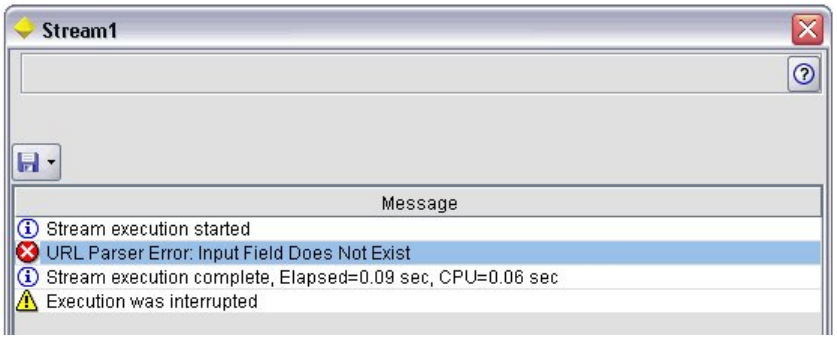

図 29. エラー・メッセージの表示

次の例では、エラー・メッセージのテキストは messageKey 属性に参照されます。

```
<StatusCodes>
     <StatusCode code="0" status="error" messageKey="initErrMsg.LABEL"/>
     <StatusCode code="1" status="error" messageKey="inputErrMsg.LABEL"/>
     <StatusCode code="2" status="error" messageKey="internalErrMsg.LABEL"/>
     <StatusCode code="3" status="error" messageKey="invalidMetadataErrMsg.LABEL"/>
      ...
</StatusCodes>
```

仕様ファイルと同じフォルダーにあるプロパティー・ファイル (messages.properties など) には、実際のメッセージ テキストとその他の表示テキストが含まれています。

```
 ...
initErrMsg.LABEL=Initialisation failed.
inputErrMsg.LABEL=Error when reading input data.
internalErrMsg.LABEL=Internal error.
invalidMetadataErrMsg.LABEL=Metadata (on input/output fields) not valid.
 ...
```

この方法は、ローカライズ用のすべてのテキストが 1 つのファイルにあるため、表示テキストを海外の市場向けにローカライズする必要がある場合に役立ちます。 詳しくは、トピック 171 ページの『ローカライゼーション』を参照してください。

出力データ・モデル

この要素は、*Node* 要素の定義でのみ使用されます。

Output Data Model セクションは、データ・モデルが特定のプロパティーにどのように影響されるかを指定します。

出力データ・モデルは、次の 3 つの方法のいずれかで指定することができます。

- 仕様ファイルのフィールド・セット定義機能を使用する。 詳しくは、トピック 71 ページの『フィールド・セット』を参照してください。
- プロパティーのセットおよび入力データ・モデルを受け取り、データ・モデルのインスタンスを返すデータ・モデル・プロバイダのインターフェースを実装するクライアント側の Java クラスを使用する。
- プロパティーのセットおよび入力データ・モデルを受け取り、メタデータ・ドキュメントを返す、サーバー側の共有ライブラリー・コンポーネントを使用する。

Output Data Model セクションは、ノードのプロパティーがノード全体のフィールドにどのように影響するかを定義します。出力データ・モデルによって実行できるのは次のとおりです。

- 入力データ・モデルを変更しないままにする
- 入力データ・モデルを変更する
- 入力データ・モデルを別のデータ・モデルを置き換える

例えば、入力ノードはプロパティー自体に影響を与えませんが並べ替え、フィールド作成ノードは新規フィールドを追加してデータ・モデルを変更し、レコード集計ノードはデータ・モデルを完全に置き換えます。

入力データ・モデルを変更した場合、定義では新規フィールドを追加、または既存のフィールドを変更または削除することができます。データ・モデルを置き換える場合は、新規フィールドのみ追加することができます。仕様ファイルは、基本的な操作 (型が入力フィールドに基づく新しいフィールドを作成する機能など) のほかに、入力フィールド・セットや、入力フィールド・セットのフィールドのグループを表すキーまたはリストのプロパティー全体にわたって反復する機能をサポートしています。

フォーマット

Output Data Model セクションの一般的な形式は以下のとおりですが、これらのケースの具体的な形式については、 139 ページの『複数フィールド設定コントロール』のセクションと 147 ページの『単一フィールド設定コントロール』のセクションを参照してください。

```
<OutputDataModel mode="mode" libraryId="container_name">
     -- data model operations --
</OutputDataModel>
```

出力データ・モデルの属性は、次の表に示すとおりです。

表 *18.* 出力データ・モデルの属性：

属性	説明
mode	データ・モデルへの影響。 extend - 新しいフィールドを既存のモデルに追加 fixed - 変更なし modify - 既存のフィールドを変更 (削除または名前を変更) replace - 既存のモデルを置換
libraryId	データ・モデルを取得するサーバー側のコンテナーの名前。

データ・モデルの操作は、新しいフィールドを追加、または既存のフィールドを変更または削除する操作です。 詳しくは、トピック 65 ページの『データ・モデルの操作』を参照してください。

例

OutputDataModel 要素が、仕様ファイルの例に含まれています。 詳しくは、トピック 32 ページの『仕様ファイルの例』を参照してください。

コンストラクター

コンストラクターは、ストリーム内のノードの実行またはストリームへのオブジェクトの生成の結果として作成されるオブジェクトを定義します。

ファイルのこの部分のコード化の詳細は、「 103 ページの『コンストラクターの使用』」で始まるセクションを参照してください。

共通機能

仕様ファイルの複数のセクションで使用できる機能もあります。

- 値の種類
- 評価文字列
- 操作
- フィールドおよびフィールド・メタデータ
- フィールド・セット
- 役割
- 論理演算子
- 条件

値の種類

値の種類の宣言は、列、プロパティーまたはプロパティー・タイプがとることができる値の種類を指定します。

文字列および暗号化された文字列

valueType="string" の形式は、値がテキスト文字列であることを指定します。
valueType="encryptedString" 宣言は、パスワードのフィールドなどユーザーが入力した場合に内容を非表示にする必要があるフィールドに関連するプロパティーに使用されます。

フィールド名

値をフィールド名の形式から取得する必要がある場合、valueType="fieldName" の形式を使用します。

数学式、論理式、および日付式

値が数学的 (整数または倍精度)、論理 (true/false) または日付式である場合、valueType を integer、double、boolean または date にそれぞれ設定します。

列挙型プロパティー

列挙型プロパティーは valueType="enum" 宣言の直後に続く Enumeration セクション内に含まれています。 詳しくは、トピック『列挙型プロパティー』を参照してください。

構造の宣言

valueType="structure" 宣言は、指定された他の属性を含む複合値を指定します。属性はプロパティーと似ていますが、構造化またはキー化されません。 詳しくは、トピック 63 ページの『構造化プロパティー』を参照してください。

- キー指標 : プロパティーが単一の値であるか、またはテーブルの各値が指定された値の種類であるハッシュ テーブルであるかを指定します。
- 値セット : 使用可能な値のセットがどのように決定されるかを指定します。
- キー・セット : キー・プロパティーの場合、使用可能なキーのセットがどのように決定されるかを指定します。この情報は、使用する最も適切な種類のコントローラに関するヒントをユーザー・インターフェースに提供するためにも使用します。

データベース接続

user1@testdb など、ユーザーがデータベースにログオンすることができる接続文字列です。ログオンの詳細は、データベースに定義されている必要があります。 詳しくは、トピック 137 ページの『データベース接続設定コントロール』を参照してください。

列挙型プロパティー

列挙型プロパティーは、値の事前定義されたリストから値を取得することができるプロパティーです。

フォーマット

列挙型プロパティーでは、次のように値のリストが定義されている Enumeration セクションを使用します。

```
<PropertyTypes>
     <PropertyType id="identifier" valueType="enum">
          <Enumeration>
               <Enum value="value" label="display_label" labelKey="label_key"
                    description="description" descriptionKey="description_key" />
           ...
          </Enumeration>
     </PropertyType>
<PropertyTypes>
```

ここで PropertyType 属性は次のとおりです。

- id は、プロパティー・タイプの一意の識別子です。
- valueType は、プロパティー・タイプが列挙型であることを示します。

Enum 属性は次のとおりです。

- value (必須) は、値のリストに表示されるプロパティー値です。
- label (必須) は、ユーザー・インターフェースに表示される、プロパティー値の表示名です。
- labelKey は、ローカライズ用のラベルを識別します。
- description は、列挙された値の説明です。
- descriptionKey は、ローカライズ用の説明を識別します。

例

```
<PropertyTypes>
     <PropertyType id="shared_enum1" valueType="enum">
          <Enumeration>
               <Enum value="value1" label="Value 5.1" labelKey="enum5.value1.LABEL" />
               <Enum value="value2" label="Value 5.2" labelKey="enum5.value2.LABEL" />
               <Enum value="value3" label="Value 5.3" labelKey="enum5.value3.LABEL" />
          </Enumeration>
     </PropertyType>
<PropertyTypes>
```

構造化プロパティー

構造化プロパティーは、ダイアログのテーブル・コントロールなど、グリッドのような構造で使用されるプロパティーです。

フォーマット

構造化プロパティーの形式では、次のように構造が定義されている Structure セクションを使用し、多くの Attribute 要素で構成されています。

```
<PropertyTypes>
     <PropertyType id="identifier" valueType="structure" isList="true_false">
          <Structure>
               <Attribute name="column_ID" valueType="value_type" isList="true_false"
                    label="column_label" labelKey="label_key" defaultValue="value"
                    description="description" descriptionKey="description_key" />
           ...
          </Structure>
     </PropertyType>
</PropertyTypes>
```

ここで PropertyType 属性は次のとおりです。

- id は、プロパティー・タイプの一意の識別子です。
- valueType は、プロパティー・タイプが構造型であることを示します。
- isList は、プロパティーが指定された値の種類の値のリスト (true) であるか、または単一の値 (false) であるかを指定します。

Attribute 要素の属性は次のとおりです。

- name (必須) は、列の識別子です。
- valueType は、列の内容がとることができる値の種類を指定します。次のいずれかを指定します。

  string

  encryptedString

  integer

  double

  boolean

  date

  enum

- isList は、この属性が、指定された種類の値のリスト (true) か、単一の値 (false) かを示します。このようにして、キー・プロパティーを既知の属性の固定セット (特定のフィールドで実行されるさまざまなデータ集計操作を示す Boolean 属性など) または値のリスト (その他の同じフィールド名と関連するフィールド名のリストなど) と関連付けることができます。
- label (必須) は、ユーザー・インターフェース上で表示する場合の、列の表示名です。
- labelKey は、ローカライズ用のラベルを識別します。
- defaultValue は、表示時に列に表示される値です。
- description は、列の説明です。
- descriptionKey は、ローカライズ用の説明を識別します。

例 - テーブル・コントロール

テーブル・コントロールでの構造化プロパティーの使用方法については、 150 ページの『テーブル・コントロール』を参照してください。

例 - キー・プロパティー・タイプ

最初の例では、関連する各値が固定されたセットの操作のフィールドに適用する集計操作を示す構造であるキー・プロパティー・タイプを示します。

```
<PropertyType id="aggregateOps" isKeyed="true" valueType="structure">
  <Structure>
    <Attribute name="MIN" valueType="boolean" label="Min" />
    <Attribute name="MAX" valueType="boolean" label="Max" defaultValue="true"/>
    <Attribute name="SUM" valueType="boolean" label="Sum" defaultValue="false"/>
    <Attribute name="MEAN" valueType="boolean" label="Mean" defaultValue="false"/>
    <Attribute name="SDEV" valueType="boolean" label="SDev" defaultValue="false"/>
  </Structure>
</PropertyType>
```

aggregateOps プロパティー・タイプを使用するよう宣言したプロパティーは、次のようになります。

```
<Property name="aggregationSettings" scriptName="aggregation_settings" type="aggregateOps"/>
```

ここで、プロパティーは異なるキーを持つ複数の値で構成されています。例えば、キー name はフィールドの名前です (MIN、MAX など)。

キー・プロパティー・タイプの次の例では、関連する各値は単一の属性を含む構造です。この場合、属性はフィールドに適用される乗数を示す倍精度の式の一覧です。

```
<PropertyType id="multiplierOps" isKeyed="true" valueType="structure">
  <Structure>
    <Attribute name="multipliers" valueType="double" isList="true"/>
  </Structure>
</PropertyType>
```

multiplierOps プロパティー・タイプを使用するよう宣言したプロパティーは、次のようになります。

```
<Property name="multiplierSettings" scriptName="multiplier_settings" type="multiplierOps"/>
```

デフォルト値(D)

DefaultValue 要素を使用して、サーバー一時ディレクトリー、ファイル、またはその両方を指定します。モデル出力オブジェクトまたはドキュメント出力オブジェクトを保存するために作成されます。

フォーマット

```
<DefaultValue>
     <ServerTempDir basename="name"/>
     <ServerTempFile basename="name"/>
</DefaultValue>
```

ここで basename (必須) は、一時ディレクトリーまたはファイルの名前です。

例

```
<DefaultValue>
     <ServerTempFile basename="datatmp"/>
</DefaultValue>
```

評価文字列

仕様ファイルで宣言された文字列の中には、プロパティー名の参照が含まれるものもあります。これらの文字列は、評価文字列とも呼ばれます。

プロパティー参照の構文は、次のとおりです。

"${*property_name*}"

評価文字列にアクセスした場合、プロパティー参照は参照されたプロパティーの値と置き換えられます。プロパティーが存在しない場合、エラーが発生します。例えば、新規フィールドを追加する場合、ノード定義に my_new_field というプロパティーが存在し、このプロパティーの値の編集を可能にするコントロールが User Interface セクションに含まれている場合があります。

例

```
<AddField name="${my_new_field}" ... >
```

操作

仕様ファイルの特定のセクションは、フィールドの追加、コンポーネントの作成、プロパティーの初期化などの操作をサポートしています。操作をサポートするセクションは次のとおりです。

- 出力データ・モデル (入力ノードおよびプロセス・ノード)
- 入力データ・モデルおよび出力データ・モデル (コンポーネント)
- 出力オブジェクト作成 (モデル・ビルダー・ノードおよびドキュメント・ビルダー・ノード)
- モデル・アプライヤー作成 (モデル出力)

操作は、次のタイプに分けられます。

- データ・モデル操作: AddField、ChangeField、RemoveField
- 反復: ForEach

データ・モデルの操作

データ・モデルで実行できる操作は次のとおりです。

- 新規フィールドを既存のデータ・モデルに追加する
- データ・モデルの既存のフィールドを変更する
- データ・モデルのフィールドを削除する

フィールドの追加: AddField 要素によって、新規フィールドを既存のデータ・モデルに追加することができます。

フォーマット

```
<AddField prefix="prefix" name="name" direction="field_role" directionRef="field_role_ref"
     fieldRef="field_ref" group="group_id" label="label" missingValuesRef="mval_ref"
     storage="storage_type" storageRef="storage_ref" targetField="target_field"
     type="data_type" typeRef="type_ref" role="role" tag="propensity_type" value="value"
depth="integer" valueStorage="storage_type">
     <Range min="min_value" max="max_value" />
</AddField>
```

AddField の属性は、次のとおりです。

表 *19. AddField* 属性：

属性	説明
prefix	モデル出力フィールドを示すなど、フィールド名に追加される接頭辞。
name	(必須) 追加されるフィールドの名前。ハードコード化された文字列 (field8 など) またはフィールドを参照する評価された文字列 (${target} など) のいずれかです。詳しくは、トピック 65 ページの『評価文字列』を参照してください。
direction	フィールドが入力となるか対象となるかなど、フィールドの役割。in、out、both、partition、または none のいずれかです。
directionRef	フィールドの方向がフィールドを参照する評価された文字列 (${field1} など) で識別されるフィールドの方向から取得されるよう指定します。 詳しくは、トピック 65 ページの『評価文字列』を参照してください。
fieldRef	フィールドを参照する評価対象文字列 (${field1} など) によって識別されるフィールドの対応する値からすべての参照値 (directionRef、missingValuesRef、storageRef、および typeRef) を取得するように指定します。詳しくは、トピック 65 ページの『評価文字列』を参照してください。
group	フィールドがフィールド グループのメンバーとなるよう指定します。 詳しくは、トピック 88 ページの『モデル・フィールド』を参照してください。
label	追加されるフィールドのラベル。
missingValuesRef	フィールドを参照する評価された文字列 (${field1} など) で識別されるフィールドの欠損値特性から取得するよう、欠損値を処理する方法を指定します。 詳しくは、トピック 65 ページの『評価文字列』を参照してください。
storage	フィールド値のデータ ストレージ タイプ - integer、real、string、date、time、timestamp、list、または unknown のいずれかです。
storageRef	ストレージ・タイプがフィールドを参照する評価された文字列 (${field1} など) で識別されるフィールドのストレージ・タイプから取得されるよう指定します。 詳しくは、トピック 65 ページの『評価文字列』を参照してください。

表 19. *AddField* 属性 (続き):

属性	説明
targetField	モデル出力フィールドについて、この新しいフィールドのデータが取得される対象フィールドを指定します。ハードコード化された文字列 (field8 など) またはフィールドを参照する評価された文字列 (${target} など) のいずれかです。 詳しくは、トピック 65 ページの『評価文字列』を参照してください。
type	フィールド値のデータ型 - auto、range、discrete、set、orderedSet、flag、collection、geospatial、または typeless のいずれかです。
typeRef	データ型がフィールドを参照する評価された文字列 (${field1} など) で識別されるフィールドのデータ型から取得されるよう指定します。 詳しくは、トピック 65 ページの『評価文字列』を参照してください。
role	モデル出力フィールドにあるデータの種類 - unknown、predictedValue、predictedDisplayValue、probability、residual、standardError、entityId、entityAffinity、 upperConfidenceLimit、lowerConfidenceLimit、propensity、value または supplementary のいずれか。詳しくは、トピック 72 ページの『役割』を参照してください。
tag	role の値が propensity の場合のみ使用。傾向タイプを示し、RAW または ADJUSTED となります。
value	新規フィールドが含む値がフィールドを参照する評価された文字列 (${field1} など) で識別されるフィールドから取得されるよう指定します。詳しくは、トピック 65 ページの『評価文字列』を参照してください。
depth	これは、storage の値が list の場合にのみ使用され、リストの深さ (-1 以上でなければならない) を定義します
valueStorage	これは、storage の値が list の場合のみ使用され、リスト値のストレージ タイプ (integer、real、string、date、time、または timestamp のいずれか) を定義します。

Range の属性は、次のとおりです。

表 20. 範囲の属性

属性	説明
最小	フィールドが受け入れることができる最小値。
最大	フィールドが受け入れることができる最大値。

例

次の例では、field8 という文字列を追加します。

```
<AddField name="field8" storage="string" />
```

次の例では、フィールド追加時にどのように参照をプロパティー名に追加することができるかを示します。ここで、フィールドは以前定義されたプロパティー prop1 の値に一致する名前で追加されます。

```
<AddField name="${prop1}" ... />
```

次の例では、対象フィールドの名前が field1 の場合、モデルは $S-field1 という出力フィールドを作成し、field1 の予測された値を保持します。

```
<AddField prefix="$S" name="${target}" role="predictedValue" targetField="${target}"/>
```

次の例では、0.0 ～ 1.0 の確率スコアを保持するモデル出力フィールドを追加します。

```
<AddField prefix="$SC" name="${target}" storage="real" role="probability" targetField=
"${target}">
     <Range min="0.0" max="1.0"/>
</AddField>
```

最後の例では、各モデル出力フィールドについて、0.0 ～ 1.0 の確率スコアを保持する出力フィールドが追加され、変数 fieldValue の値から値を取得します。

```
<ForEach var="fieldValue" inFieldValues="${field}">
     <AddField prefix="$SP" name="${fieldValue}" storage="real" role="probability" targetField=
     "${field}" value="${fieldValue}">
          <Range min="0.0" max="1.0"/>
     </AddField>
</ForEach>
```

詳しくは、トピック 65 ページの『評価文字列』を参照してください。

フィールドの変更: ChangeField 要素によって、データ・モデルの既存のフィールドを変更することができます。

フォーマット

```
<ChangeField name="name" fieldRef="field_reference" direction="field_role" storage="storage_
type" type="data_type" >
     <Range min="min_value" max="max_value" />
</ChangeField>
```

ChangeField の属性は、次のとおりです。

表 21. *ChangeField* 属性:

属性	説明
name	(必須) 変更されるフィールドの名前。
fieldRef	フィールドの参照値。
direction	フィールドが入力となるか対象となるかなど、フィールドの役割。in、out、both、partition、または none のいずれかです。
storage	フィールド値のデータ・ストレージ・タイプ - integer、real、 string、date、time、timestamp または unknown のいずれか。
type	フィールド値のデータ型 - auto、range、discrete、 set、orderedSet、flag または typeless のいずれか。

Range の属性は、次のとおりです。

表 22. 範囲の属性

属性	説明
最小	フィールドが受け入れることができる最小値。
最大	フィールドが受け入れることができる最大値。

フィールドの削除: RemoveField 要素によって、データ・モデルのフィールドを削除することができます。

フォーマット

```
<RemoveField fieldRef="field_reference" />
```

fieldRef はフィールドの参照値です。

ForEach 要素による反復

同じ操作を繰り返し実行してそれぞれの値のセットを処理すると役に立つ場合があります。仕様ファイルでは、順番に提供されたセットの各値に一時プロパティーを結びつける単純な ForEach の反復をサポートしています。ForEach ループを設定して、次のいずれかの方法で反復することができます。

- オプションのステップ・サイズで 2 つの整数間
- リスト・プロパティーの値全体
- キー・プロパティーのキー全体
- フィールド グループのフィールド全体

フォーマット

```
<ForEach var="field_name" from="integer_exp" to="integer_exp" step="integer_exp"
inFields="fields" inFieldValues="field_name" inProperty="property_name" >
   -- data model operation --
</ForEach>
```

ここで、

var (必須) は、反復が適用される値を含むフィールドを指定します。

from および to は、オプションの属性 step で整数のステップ・サイズを指示して、反復の下限および上限を示す整数 (または整数を計算する式) を指定します。

inFields、inFieldValues、inProperty は、from/to/step 形式の代替として使用します。以下を参照してください。

- inFields は、次の中から反復を実行するフィールド・セットを指定します。

  inputs - ノードの入力フィールド

  outputs - ノードの出力フィールド

  modelInput - モデル署名で指定された入力フィールド

  modelOutput - モデル署名で指定された入力フィールド
- inFieldValues は、フィールド名 (またはフィールド名を示すプロパティー) およびフィールドのメタデータの値の反復を指定します。
- inProperty は、反復を実行するプロパティーの名前を指定します。

ForEach 要素内で指定できるデータ・モデル操作は、AddField 要素、ChangeField 要素または RemoveField 要素のいずれかです。 詳しくは、トピック 65 ページの『データ・モデルの操作』を参照してください。ForEach 要素は入れ子にすることもできます。

例

次は、操作を 10 回実行します。

```
<ForEach var="val" from="1" to="10">
 ...
</ForEach>
```

次は、整数プロパティーで指定された回数だけ、操作を実行します。

```
<ForEach var="val" from="1" to="${history_count}">
 ...
</ForEach>
```

次の例で、処理はノードの出力フィールドの値だけ反復します。

```
<ForEach var="field" inFields="outputs">
 ...
</ForEach>
```

次の例では、${field} で識別されたフィールドのメタデータ内の値だけ反復します。

```
<ForEach var="fieldValue" inFieldValues="${field}">
 ...
</ForEach>
```

次の例では、リスト・プロパティーの値だけ反復します。

```
<ForEach var="val" inProperty="my_list_property">
 ...
</ForEach>
```

次の例では、キー・プロパティーのキー値の分反復します。

```
<ForEach var="key" inProperty="my_keyed_property">
 ...
</ForEach>
```

フィールドおよびフィールド メタデータ

ノード、モデル、およびデータ・ソースはデータ・モデル・プロバイダとして動作します。これらは他のオブジェクトからアクセスできるフィールド メタデータを定義することができます。

データ・モデル・プロバイダには、入力データ・モデルおよび出力データ・モデルが含まれます。例えば、フィールドを追加して入力モデルを超える場合や既存のモデルを変更する場合など、出力データ・モデルは入力データ・モデルについて定義することができます。

これらのオブジェクトはそれぞれ、要件が異なります。

ノード : 入力データ・モデルを参照することができますが、変更することはできません。出力データ・モデルは入力データ・モデルに基づいたり、置き換えたりすることができます。ノード・プロパティーまたは入力データ・モデルが変更される場合は、出力データ・モデルを再度計算します。モデル・アプライヤー・ノードの出力データ・モデルは、モデル・コンポーネントの出力データ・モデルを参照することもできます。

モデル: デフォルトでは、入力データ・モデルおよび出力データ・モデル (モデル署名) は、モデルの作成時に使用された入力フィールドおよび出力フィールドの設定に基づきます。モデル構築プロセスは、必要な入力フィールドおよび生成された出力フィールドを定義するメタデータ・ファイルを返すことが理想的です。モデル署名をいったん定義すると、変更することはできません。ただし、モデル・アプライヤー・ノードのプロパティーはアプライヤー・ノードのデータ・モデル出力を変更することができます。例えば、これらのプロパティーはクラスター ID が文字列として返されるのか整数として返されるのか、生成される一連の ID の数を定義します。また、モデル署名は通常、フィールドの役割 (direction) が "out" に設定されているものとして、出力を指定します。これに対し、ノードは多くの場合、このフィールドの役割が "in" である出力を生成します。

データ・ソース: データ・リーダー・ノードで使用されるデータ・ソースは出力データ・モデルを指定することができます。入力データ・モデルは常に空です。

フィールド・セット

フィールド・セットを多くの場所で使用して、データ・モデル・プロバイダのフィールドのサブセットを選択することができます。データ・モデル・プロバイダは、囲むオブジェクトまたは囲むオブジェクトのコンテナーの場合があります。フィールド・フィルターの最初の状態は、使用可能なすべてのフィールドを含めて特定の種類のフィールドを除外するか、空のフィールドのセットで開始し、必要なフィールドを含めたり新規フィールドを追加する場合があります。

次の例では、拡張ノードがどのように出力データ・モデルを指定することができるかを示します。キー・フィールドは、keys というリスト・プロパティーで指定され、名前のプロパティーで指定される生成可能なオプションのレコード度数フィールドが後に続きます。

```
<OutputDataModel mode="replace">
     <ForEach var="field" inProperty="keys">
          <AddField name="${field}" fieldRef="${field}"/>
     </ForEach>
     <AddField name="${record_count_name}" storage="integer">
          <Condition property="include_record_count" op="equals" value="true"/>
     </AddField>
</OutputDataModel>
```

フィールドセットおよびモデル構築

次の例では、モデル・アプライヤーがどのようにして以前作成されたモデル・コンポーネントの情報を使用して出力フィールドを生成できるかを示します。

```
<OutputDataModel mode="modify">
     <AddField provider="model" dataModel="output">
</OutputDataModel>
```

AddField および ForEach は、使用される入力データ・モデルまたは出力データ・モデルを指定すると共にデータ・モデル・プロバイダを指定します。データ・モデル・プロバイダのフィールドのセット (またはサブセット) を指定するメカニズムを提供します。デフォルトのプロバイダは、デフォルトで使用される入力フィールド・セットを持つ囲み要素 (例えば、作成されるオブジェクト以外) を示す this です。指定されるフィールド・セットがない場合、使用可能なフィールドはすべて使用されます。

フィールド・セットはストレージ、データ型、フィールドの役割または名前に基づきます。名前に基づく場合、リスト・プロパティーへの参照が必要です。フィールド・セットは完全 (デフォルト) または空の場合があります。完全の場合はフィールドを除外することができ、空の場合はフィールドを含むことができます。複数の値を個々のフィルターそれぞれに指定することができ、これらの値は「交差」または「and」の演算子として動作します。例を次に示します。

```
<FieldSet include="none">
     <Include direction="in" storage="string"/>
</FieldSet>
```

(include="none" で指定された) 空のフィールド・セットで開始し、フィールドの役割 (direction) が「"in"」の文字列ストレージのフィールドが含まれます。

次にもう 1 つの例を示します。

```
<FieldSet include="all">
     <Exclude type="typeless"/>
</FieldSet>
```

ここでは、使用可能なすべてのフィールド (include="all" 属性で指定され、デフォルトの動作) が含まれ、typeless のデータ型は除外されます。そのため、direction が "in" または "both" に設定されたフィールドが組み込まれます。

複数のフィルターを指定することもでき、これらは「統合」または「or」の演算子として動作します。

```
<FieldSet include="all">
     <Exclude type="discrete" storage="real"/>
     <Exclude type="discrete" storage="integer"/>
</FieldSet>
```

実数のストレージを持つ離散型のフィールドまたは整数のストレージを持つ離散型のフィールドは除外されます。

最初空のフィールド・セットにフィールドが含まれる場合、include 文の順序は通常、フィールドが含まれる順序に影響を与えません。フィールド・セット プロバイダのフィールドは、各条件に対して普通の順序で評価され、フィールド・セットに含むかどうかを指定します。

役割

役割は、データ・モデル出力フィールドに保持されるデータの種類を説明します。役割は AddField 要素で指定し、ondition 要素でテストすることができます。

役割は次の通りです。

表 23. モデル出力の役割

役割	意味
unknown	役割を指定されていません (指定することはできません)。
predictedValue	このフィールドには、対象フィールドの予測値が含まれます。
probability	予測の確率または確信度。
residual	残差の値。
standardError	予測の標準偏差。
entityId	エンティティー ID。通常、クラスター・モデルのクラスター ID を示します。
entityAffinity	エンティティーの類似性。通常、モデルのクラスターの中心からの距離を示します。
upperConfidenceLimit	予測の確信度の上限。
lowerConfidenceLimit	予測の確信度の下限。

表 23. モデル出力の役割 (続き)

役割	意味
propensity	傾向スコア。追加の tag 属性は未調整傾向を参照するか調整済み傾向を参照するかを指定します。
value	別の出力に関連するモデルの値を示します (下記参照)。
supplementary	他の役割でカバーされていないモデルで生成された情報。

value の例として、異常値検出モデルで、1 つのフィールドがフィールド名を表し、もう 1 つがフィールドの異常度を示す尺度を指定する、各グループが 2 つのフィールドで構成されるフィールドのグループを生成します。この場合、value はフィールド名となります。

論理演算子

多くの要素で、複合条件を設定する場合などに、論理演算子の And、Or、Not を使用してさまざまな処理を指定することができます (77 ページの『複合条件』を参照)。

フォーマット

And 要素の形式は次のとおりです。Or 要素と Not 要素の形式はほぼ同じですが、囲むタグがそれぞれ <Or>...</Or> と <Not>...</Not> である点だけが異なります。

```
<And>
     <Condition .../>
     <And ... />
     <Or ... />
     <Not ... />
</And>
```

Condition 要素は、テストされる条件を指定します。 詳しくは、トピック『条件』を参照してください。

子要素の And、Or、Not は入れ子にすることができます。

条件

いくつかのオブジェクトの動作を条件を使用して変更することができます。Condition 要素 (IF 文と同等) で指定します。例えば、コマンドで実行されるノードは、プロパティーに特定の値がある場合特定のオプションのみを含むなど、実行情報に条件を追加することができます。同様に、ユーザー・インターフェースのプロパティー・コントロールを、別のコントロールに特定の値が含まれている場合にのみ有効化または表示することができます。

条件は単純にすることも、複合にすることもできます。単純な条件は、次の内容で構成されています。

- 値のソース (プロパティーまたはコントロールのいずれか)
- テスト
- オプションのテスト値

複合条件を使用すると、他の条件と結合して複雑な論理条件を形成することができます。複合条件で使用されるのは次のとおりです。

- And
- Or
- Not

フォーマット

```
<Condition container="container_name" control="prop_name" property="name" op="operator"
     value="value" />
```

ここで、

container は、値が条件でテストされる特定のコンテナーの名前を指定します。

control は、条件で値がテストされるプロパティー・コントロールを指定します。*prop_name* は、コントロールが定義される要素の property 属性の値です (ダイアログ・タブのプロパティー・パネルなど)。

property は、条件で値がテストされるプロパティーを指定します。*name* は、コントロールが定義される Property 要素の name 属性の値です。

op は、条件の演算子です。詳しくは、トピック『条件演算子』を参照してください。

value は、条件でテストされる特定の値です。

例

条件設定の例は、 76 ページの『単純な条件』および 77 ページの『複合条件』を参照してください。

条件演算子

一連の演算子は、ほとんどの条件を処理するために使用できます。

表 *24.* 値でサポートされるテスト

演算子	値	説明
equals	*value*	プロパティーが提供された値と等しい場合は真です (文字列値の場合は大文字/小文字を区別します)。
notEquals	*value*	プロパティーが提供された値と等くない場合は真です (文字列値の場合は大文字/小文字を区別します)。
in	*list of values*	プロパティーは提供された値のリスト内になる場合は真です。

表 *25.* 数値でサポートされるテスト

演算子	値	説明
lessThan	*number*	プロパティーは提供された数値を下回る場合は真です。
lessOrEquals	*number*	プロパティーは提供された数値以下の場合は真です。
greaterThan	*number*	プロパティーは提供された数値を上回る場合は真です。
greaterOrEquals	*number*	プロパティーは提供された数値以上の場合は真です。

表 *26.* 文字列値でサポートされるテスト

演算子	値	説明
isEmpty	-	プロパティーに 0 の長さの文字列が含まれる場合は真です。
isNotEmpty	-	プロパティーに 0 以外の長さの文字列が含まれる場合は真です。

表 26. 文字列値でサポートされるテスト (続き)

演算子	値	説明
startsWith	*string*	プロパティーが提供された文字列で開始する場合は真です (大文字/小文字を区別)。
startsWithIgnoreCase	*string*	プロパティーが提供された文字列で開始する場合は真です (大文字/小文字の区別はなし)。
endsWith	*string*	プロパティーが提供された文字列で終了する場合は真です (大文字/小文字を区別)。
endsWithIgnoreCase	*string*	プロパティーが提供された文字列で終了する場合は真です (大文字/小文字の区別はなし)。
equalsIgnoreCase	*string*	プロパティーが提供された文字列と等しい場合は真です (大文字/小文字の区別はなし)。
hasSubstring	*string*	プロパティーが提供された文字列を含む場合は真です (大文字/小文字を区別)。
hasSubstringIgnoreCase	*string*	プロパティーが提供された文字列を含む場合は真です (大文字/小文字の区別はなし)。
isSubstring	*string*	プロパティーが提供された文字列のサブ文字列である場合は真です (大文字/小文字を区別)。
isSubstringIgnoreCase	*string*	プロパティーが提供された文字列のサブ文字列である場合は真です (大文字/小文字の区別はなし)。

表 27. リスト・プロパティーでサポートされたテスト

演算子	値	説明
isEmpty	-	リスト内の項目数が 0 の場合は真です。
isNotEmpty	-	リスト内の項目数が 0 でない場合は真です。
countEquals	*number*	リスト内の項目数が提供された値と等しい場合は真です。
countLessThan	*number*	リスト内の項目数が提供された値を下回る場合は真です。
countLessOrEquals	*number*	リスト内の項目数が提供された値以下の場合は真です。
countGreaterThan	*number*	リスト内の項目数が提供された値を上回る場合は真です。
countGreaterOrEquals	*number*	リスト内の項目数が提供された値以上の場合は真です。
contains	*value*	提供された項目がリスト内にある場合は真です。

表 28. フィールド・プロパティーでサポートされたテスト

演算子	説明
storageEquals	ストレージが提供された値と等しい場合は真です。
typeEquals	種類が提供された値と等しい場合は真です。
directionEquals	フィールドの役割 (direction) が提供された値と等しい場合は真です。
isMeasureDiscrete	フィールドのデータ型が discrete の場合は真です (set、flag または orderedSet のいずれかのみ)。

表 28. フィールド・プロパティーでサポートされたテスト (続き)

演算子	説明
isMeasureContinuous	フィールドのデータ型が range の場合は真です。
isMeasureTypeless	フィールドのデータ型が typeless の場合は真です。
isMeasureUnknown	フィールドのデータ型が unknown の場合は真です。
isStorageString	フィールドのストレージ・タイプが string の場合は真です。
isStorageNumeric	フィールドのストレージ・タイプが numeric の場合は真です。
isStorageDatetime	フィールドのストレージ・タイプが datetime の場合は真です。
isStorageUnknown	フィールドのストレージ・タイプが unknown の場合は真です。
isModelOutput	フィールドがモデル出力フィールドの場合は真です。
modelOutputRoleEquals	フィールドの役割が次の表で示される有効な役割のいずれかである場合は真です。
modelOutputTargetFieldEquals	対象フィールドが指定された値 (文字列) と等しい場合は真です。
modelOutputHasValue	フィールドはモデル出力フィールドで、それに関連する値を持つ場合は真です。
modelOutputTagEquals	タグが指定された値 (文字列) と等しい場合は真です。

キー・プロパティー をサポートする条件演算子は次のとおりです。

- isEmpty
- isNotEmpty
- countEquals
- countLessThan
- countLessOrEquals
- countGreaterThan
- countGreaterOrEquals
- contains

単純な条件

単純な条件は、テストされる最初の値のソース (プロパティー名またはコントローラ名のいずれか、または評価される式)、実行されるテスト、オプションでテストが実行される値で構成されます。

例

次の例では、values_grouped というブール・プロパティーが真である場合、真と評価します。

```
<Condition property="values_grouped" op="equals" value="true"/>
```

次の例では、ブール値を表示する values_grouped というコントロールがチェックされた場合に真と評価されます。

```
<Condition control="values_grouped" op="equals" value="true"/>
```

次の例は、plot_fields というリスト・プロパティーに少なくとも 1 つの値がある場合に真と評価します。

```
<Condition property="plot_fields" op="countGreaterThan" value="0"/>
```

次の例では、input_fields というリスト・プロパティーにインスタンス化された値だけが含まれている場合に真と評価します。

```
<Condition property="input_fields" op="instantiated" listMode="all"/>
```

最後の例では、input_fields というプロパティーにインスタンス化されていないフィールドを示す値を少なくとも 1 つ含む場合に真と評価します。

```
<Condition property="input_fields" op="uninstantiated" listMode="any "/>
```

複合条件

単純な条件のグループを、論理演算子を使用して結合することができます。

例

次の例では、ブール values_grouped プロパティーが真で、group_fields に少なくとも 1 つの値が含まれる場合に真と評価します。

```
<And>
     <Condition property="values_grouped" op="equals" value="true"/>
     <Condition property="group_fields" op="countGreaterThan" value="0"/>
</And>
```

次の例では、ブール values_grouped プロパティーが真、または group_fields に少なくとも 1 つの値が含まれる場合に真と評価します。

```
<Or>
     <Condition property="values_grouped" op="equals" value="true"/>
     <Condition property="group_fields" op="countGreaterThan" value="0"/>
</Or>
```

次の例では、group_fields に少なくとも 1 つの値が含まれている場合に真と評価します。

```
<Not>
     <Condition property="group_fields" op="equals" value="0"/>
</Not>
```

複合条件をネストして、条件の組み合わせを提供することができます。

スクリプトの CLEF ノードの使用

Node 要素の scriptName 属性を使用してスクリプトの CLEF ノードを参照できます。同じように、Property 要素の scriptName 属性を使用して、スクリプトのノードのプロパティーを参照できます。

いずれの場合も、scriptName 属性はオプションですが、拡張またはプロパティー間の名前の競合を回避するために属性を使用することをお勧めします。

ノード定義でスクリプト名を省略すると、スクリプトは拡張名が最初に付いている id 属性の値によってノードを参照できます。例えば、 ID import によってデータ・リーダー・ノードを定義する myext という名前の拡張子が指定されている場合、スクリプトはノードを myextimport として参照することができます。

プロパティー 定義でスクリプト名を省略すると、スクリプトは name 属性の値を使用してプロパティーを参照できます。

詳しくは、「*IBM SPSS Modeler* スクリプトとオートメーション・ガイド」を参照してください。

例 - ノードの編集および実行

次の例では、 26 ページの『データ・リーダー・ノード (Apache Log Reader)』で説明しているデータ・リーダー・ノードの例を編集および実行するタスクを自動化するスクリプトの使用方法について説明しています。

Apache Log Reader ノードの仕様ファイルでは、ノードの設定は次のように始まります。

```
<Node id="apachelogreader" type="dataReader" palette="import" labelKey="apacheLogReader.LABEL">
     <Properties>
          <Property name="log_filename" valueType="string" labelKey="logfileName.LABEL" />
     </Properties>
```

スクリプトでは、ノードとプロパティーを次のように参照します。

```
create apachelogreader
set :apachelogreader.log_filename='installation_directory¥Demos¥combined_log_format.txt'
create tablenode at 200 100
connect :apachelogreader to :tablenode
execute :tablenode
```

ここで、*installation_directory* は、IBM SPSS Modeler がインストールされているディレクトリーです。

このスクリプトを実行すると、次のようになります。

- データ・リーダー・ノードを作成する
- *combined_log_format.txt* を読み取る Apache ログ・ファイルとして指定する
- テーブル・ノードを作成する
- データ・リーダー・ノードをテーブル・ノードを接続する
- テーブル・ノードを実行する

例 - キー・プロパティー

キー・プロパティーは、標準的なスクリプト・シンタックスをサポートします。例えば、 63 ページの『構造化プロパティー』の最初の例のキー・プロパティー・タイプで示している構造は、スクリプトで次のように定義することができます。

```
set :mynode.aggregation_settings.Age = {true true false false false}
```

1 つの属性を次のように変更できました。

```
set :mynode.aggregation_settings.Age.MIN = true
```

下位互換性の保持

既存の拡張に更新を計画する場合、その拡張の以前配布されたバージョンとの互換性を保持してください。変更によっては不都合な影響のないもの、重大なリスクを伴うもの、また互換性を壊すため回避するべきものがあります。

リスクのない変更

次の変更は、下位互換性には影響ありません。

- 新しい Node 要素、ModelOutput 要素、DocumentOutput 要素、または InteractiveModelBuilder 要素の追加
- これらの要素への新しい Property 定義と関連する新しいコントロールの追加

- これらの要素への新しいコンテナーの追加*
- 既存の列挙型プロパティーへの新しい値の追加

* これらの新しいコンテナーを使用するコードは、拡張の以前のバージョンで作成されたオブジェクトの新しいコンテナーが空となるようにする必要があります。

重大なリスクのある変更

既存の宣言に対する変更は、互換性を壊す重大なリスクを引き起こします。配布する前に慎重にテストする必要があります。

回避するべき変更

次の変更は、互換性を壊すため、回避する必要があるとされています。

- `ExtensionDetail` 要素の `id` 属性または `providerTag` 属性の値の変更
- `Node` 要素、`ModelOutput` 要素、`DocumentOutput` 要素、または `InteractiveModelBuilder` 要素の `id` 属性の値の変更
- `Node` 要素、`ModelOutput` 要素、`DocumentOutput` 要素、または `InteractiveModelBuilder` 要素の拡張機能からの削除
- `Property` または `PropertyType` 要素の `valueType` 属性の値の変更

第 5 章 モデルおよびドキュメントの作成

モデルおよびドキュメント構築の概要

標準 IBM SPSS Modelerモジュールにはユーザーが様々なモデルおよびグラフを生成（または「構築」）できるノードが含まれています。CLEF で標準的に提供されていないその他のモデルやドキュメント（グラフおよびレポート）を構築する追加ノードを定義することができます。

モデル・ビルダおよびドキュメント・ビルダ ノードを定義する場合、これらのノードを実行する際に作成されるオブジェクトも定義する必要があります。「コンストラクター」と呼ばれる項目によって実行します。

次の項では、このプロセスについて詳細に説明します。

モデル

モデルは、一連の入力フィールドに基づいて結果を予測するために使用できるルールのセット、式、または方程式です。結果を予測する機能は、予測分析の中心となる目標です。IBM SPSS Modeler で結果を予測するために、次のことを行います。

- 既存データからモデルを生成
- 生成されたモデルをデータに適用し、予測を実行

モデル生成のプロセスは、モデルの「構築」とも呼ばれ、IBM SPSS Modeler ではモデル作成ノードを使用してモデルを構築します。CLEF で、モデル作成ノードはノードの定義に使用される XML 分の構文から取得された名前である モデル・ビルダー・ノードとして参照されます。 詳しくは、トピック 11 ページの『モデル・ビルダー・ノード』を参照してください。

モデルをデータに適用するプロセスは、「データのスコアリング」とも呼ばれています。データのスコアリングを行うと、モデル作成から取得した情報を使用して、新規レコードの予測を作成できます。IBM SPSS Modeler では、生成されたモデルのアイコンをストリーム領域に追加してデータのスコアリングを行います。アイコンは金色のナゲットの形をしているため、IBM SPSS Modeler の生成されたモデルは「モデル・ナゲット」として参照されます。CLEF では、マネージャ領域の「モデル」タブのモデル・ナゲットはモデル出力オブジェクトと呼ばれ、領域に追加されると モデル・アプライヤー・ノード として認識されます。 詳しくは、トピック 12 ページの『モデル・アプライヤー・ノード』を参照してください。

文書

グラフまたはレポート出力など、モデル以外のオブジェクトの生成が必要な場合があります。IBM SPSS Modeler では、モデル以外のオブジェクトはドキュメントとも呼ばれ、ドキュメント・ビルダー・ノードによって生成されます。 詳しくは、トピック 12 ページの『ドキュメント・ビルダー・ノード』を参照してください。

コンストラクター

コンストラクターは、ストリーム内のノードの実行またはストリームへのオブジェクトの生成の結果として作成されるオブジェクトを定義します。

コンストラクターは次のいずれかに定義できます。

- モデル・ビルダー・ノード
- ドキュメント・ビルダー・ノード
- モデル・アプライヤー・ノード
- モデル出力オブジェクト

モデル・ビルダー または ドキュメント・ビルダー ノードの場合、コンストラクターを使用すると、これらのノードによりノード実行時の出力オブジェクトの生成方法を定義できます。出力オブジェクトの定義には複数のプロパティーおよびコンポーネントが含まれる場合があり、Constructors セクションはこれらがどのように初期化されるか、または実行によって生成されたオブジェクトからどのように作成されるかを定義します。

モデル・アプライヤー ノードの場合、コンストラクターはノードがストリームまたは「モデル」タブに生成できるオブジェクトの種類を定義します。

モデル出力オブジェクトいコンストラクターが定義された場合、以下のことを実行できます。
- モデル出力オブジェクトがストリーム領域にある場合に作成するモデル・アプライヤー・ノードを指定
- モデル出力オブジェクトの作成に使用された設定でモデル・ビルダー・ノードを生成

詳しくは、トピック 103 ページの『コンストラクターの使用』を参照してください。

モデルの構築

モデルを生成できるノード (モデル・ビルダー・ノード) を指定する場合、モデルを実際に作成するプロセスである、IBM SPSS Modeler のモデル・ビルダー・コンポーネントとのノードの通信方法を定義する必要があります。設定ファイルの `Node` 要素の定義で行います。

ノードが指定されていない場合、エンド・ユーザーがモデル・ビルダー・ノードのダイアログ・ボックスの「実行」 ボタンをクリックするとすぐにモデル構築が開始されます。ただし、エンド・ユーザーがモデルが実際に構築される前に「実行」をクリックした後でデータ値を調整または変更できるよう、インタラクティブ・モデルを定義することもできます。さらにインタラクティブ・モデルの構築では、双方向性が定義される設定要素を使用する必要があります。 詳しくは、トピック 91 ページの『インタラクティブ・モデルの構築』を参照してください。

モデル・ビルダー・ノードを定義する場合、`Node` 要素に必要なものは次のとおりです。
- `type="modelBuilder"` 属性
- `ModelBuilder` 子要素
- `CreateModelOutput` 要素を含む `Constructors` 子要素 (103 ページの『コンストラクターの使用』を参照)

`Node` 要素の指定形式については、 48 ページの『ノード』を参照してください。

注 : 後続のセクションにおける要素の定義 (通常は「フォーマット」という見出しが付いています) では、「(必須)」と記載されていない限り、要素の属性と子要素はオプションです。要素の詳細な構文については、 209 ページの『CLEF XML スキーマ』を参照してください。

モデルを構築する場合、生成されたモデルを記述するために、拡張には `ModelOutput` 要素も必要になります (90 ページの『モデル出力』を参照してください)。`ModelOutput` 要素には、`CreateModelApplier` 定義を含む `Constructors` 子要素が必要です。 詳しくは、トピック 105 ページの『モデル・アプライヤーの作成』を参照してください。

モデル・ビルダー

ModelBuilder 要素は、モデル・ビルダー・ノードの動作を定義します。要素の属性および子要素によって定義します。

フォーマット

```
<ModelBuilder allowNoInputs="true_false" allowNoOutputs="true_false" nullifyBlanks="true_false"
     miningFunctions="[function1 function2 ... ]" >
     <Algorithm ... />
     <ModelingFields ... />
     <ModelGeneration ... />
     <ModelFields ... />
     <AutoModeling ... />
</ModelBuilder>
```

ここで、

- 入力フィールドがないまたは出力フィールドがないモデルをそれぞれ構築する場合、allowNoInputs および allowNoOutputs を明示的に使用する必要があります。
- false に設定されている場合、nullifyBlanks は、IBM SPSS Modeler のモデル・ビルダー・コンポーネントに渡されるデータの空白値をヌル値 ($null$) に置き換える機能をオフにします。デフォルトでは、空白値をヌル値に置き換えますが、例えばアルゴリズムがヌル値とは別に空白値を処理する必要がある場合に、この機能を無効化する必要があります。
- miningFunctions (必須) は、データ・マイニング機能またはモデルを実行する機能を識別します。

表 29. データ・マイニング機能：

関数	説明
classification	既知の対象値を持つレコードから、不明な対象属性の離散値 (set、flag、または orderedSet データ型の値) を予測します。
approximation	不明な対象値を持つレコードから、不明な対象属性の連続した (range データ型) 値を予測します。
clustering	類似したレコードのグループを識別し、それに応じてラベルを付けます。
association	データ内の関連したイベントまたは属性を識別します。
sequence	時間構造データのシーケンス・パターンを検索します。
reduction	例えば、下のデータ・フィールドの内容を要約する派生フィールドを使用するなど、データの複雑さを軽減します。
conceptExtraction	テキスト・マイニングに使用されます。
categorize	テキスト・マイニングに使用されます。
timeSeries	過去のデータのパターンから将来の値を予測します。
anomalyDetection	クラスター・グループの平均からの偏差に基づいて、例外的なケースを検索します。
attributeImportance	対象属性に最も大きな影響を持つ属性を識別します。
supervisedMultiTarget	さまざまな確率の 1 つに対する (「はい」または「いいえ」) 結果の尤度を推定します。

モデルが複数の機能を実行する場合、機能名は次の例のように大カッコ内でスペースで区切られます。

```
<ModelBuilder miningFunctions="[classification approximation]">
 ...
</ModelBuilder>
```

子要素

ModelBuilder 要素の子要素を以下の表に示します。

表 *30.* モデル・ビルダーの宣言の子要素：

子要素	説明	参照...
Algorithm	(必須) モデルの生成に使用されるアルゴリズムを指定します。	『アルゴリズム』
ModelingFields	User Interface セクションで一貫して使用される識別子を指定して、モデルの入力フィールドおよび出力フィールドのコントロールの場所を定義します。コントロール自体は、ModelingFields の InputFields 子要素と OutputFields 子要素で定義されます。	『モデル作成フィールド』
ModelGeneration	User Interface セクションで一貫して使用される識別子を指定して、生成されたモデルのモデル名コントロールの場所を定義します。	88 ページの『モデル生成』
ModelFields	モデルを持つデータのスコアリングに使用される入力フィールドおよび出力フィールドのセットを指定します。	88 ページの『モデル・フィールド』
AutoModeling	自動分類、自動クラスターまたは自動数値など、アンサンブル・モデル作成ノードによってモデルを使用できるようにします。	95 ページの『自動化されたモデル作成』

アルゴリズム

Algorithm 要素は、モデルの生成に使用されるアルゴリズムの詳細を定義します。

```
<Algorithm value="model_output_id" label="display_label" labelKey="label_key"/>
```

ここで、

- value (必須) は、アルゴリズムの内部名です。設定ファイルのさまざまな場所で参照されます。 詳しくは、トピック 88 ページの『モデル・ビルダーの例』を参照してください。
- label (必須) は、アルゴリズムの説明です。
- labelKey は、ローカライズ用のラベルを識別します。

モデル作成フィールド

データ型ノードを使用してノードモデルの入力フィールドおよび出力フィールドを指定するのが、標準的な方法です。必要に応じて、フィールドの役割を 「入力」 または 「出力」 に設定します。モデル・ビルダー・ノードの場合、必要に応じて、上流のデータ型ノードの設定の上書きとカスタム設定の使用をユーザーに対して許可することができます。

ModelingFields 要素によって行われます。この要素は、モデル・ビルダー・ノードの宣言の User Interface セクションで一貫して使用される識別子を指定し、モデルの入力フィールドおよび出力フィールドのコントロールを定義します。コントロール自体は、InputFields および OutputFields 子要素によって定義されます。

フォーマット

```
<ModelingFields controlsId="control_identifier" ignoreBOTH="true_false" >
      <InputFields ... />
      <OutputFields ... />
</ModelingFields>
```

ここで、

- controlsId (必須) は、これ以降に、モデル・ビルダー・ノード宣言の User Interface セクションの SystemControls 要素で使用される識別子です。モデルの入力フィールドおよび出力フィールドのコントロールを含むノードのダイアログ・ボックスのタブを識別します。
- ignoreBOTH に設定されている場合 (デフォルト)、true は、両方に方向が設定されたフィールドがモデルに無視されるよう指定します。

InputFields 要素と OutputFields 要素については、『入力フィールド』以降のセクションで説明しています。

例

この例では、モデル・ビルダー・ノードのダイアログ・ボックスの「フィールド」タブのモデル・フィールド・コントロールの使用について説明します。まず、識別子はコントロールのセットに指定されます。

```
<ModelBuilder miningFunctions="[classification]">
 ...
     <ModelingFields controlsId="modelingFields">
          <InputFields property="inputs" onlyNumeric="true" multiple="true" label="Inputs"
            labelKey="inputFields.LABEL"/>
          <OutputFields property="target" multiple="false" types="[set flag]" label="Target"
            labelKey="targetField.LABEL"/>
     </ModelingFields>
 ...
</ModelBuilder>
```

modelingFields 識別子は、「フィールド」タブが定義された時点で、ノードのダイアログ・ボックスの User Interface セクションで一貫して参照されます。

```
<UserInterface ... >
     <Tabs defaultTab="1">
          <Tab label="Fields" labelKey="Fields.LABEL" helpLink="modeling_fieldstab.htm">
               <PropertiesPanel>
                    <SystemControls controlsId="modelingFields">
                    </SystemControls>
               </PropertiesPanel>
          </Tab>
 ...
</UserInterface>
```

入力フィールド: InputFields 要素は、ユーザーがモデルの入力フィールド (予測フィールド) を選択できるフィールドのセットを定義します。

セットは、このノードで表示されるすべてのフィールドで構成されています。フィールドがこのノードの上流で詳細にフィルタリングされている場合、フィルターを通過したフィールドのみが表示されます。特定のストレージ・タイプおよびデータ型のフィールドのみが選択できるよう指定することによって、リストをより詳細に制限することができます。

```
<InputFields storage="storage_types" onlyNumeric="true_false" onlySymbolic="true_false"
     onlyDatetime="true_false" types="data_types" onlyRanges="true_false"
     onlyDiscrete="true_false" property="property_name" multiple="true_false" label="label"
     labelKey="label_key"/>
```

2 つの属性を指定して、入力フィールドとして使用されるフィールドのリストを制限できます。次のリストから選択する必要があります。

- storage は、リストに表示できるフィールドのストレージ・タイプを指定するリスト・プロパティーです。例えば storage="[integer real]" は、これらのストレージ・タイプのフィールドのみ一覧表示されることを意味します。可能なストレージ・タイプのセットについては、 188 ページの『データおよびストレージ・タイプ』 のテーブルを参照してください。
- onlyNumeric が true に設定されている場合、数値型ストレージのフィールドのみが一覧表示されます。
- onlySymbolic が true に設定されている場合、シンボル値のストレージ・タイプのフィールド (文字列) のみが一覧表示されます。
- onlyDatetime が true に設定されている場合、日付と時間のストレージ・タイプのみが一覧表示されます。

2 番目の属性を以下から選択する必要があります。

- types は、リスト内に表示されるフィールドのデータ型を指定するリストのプロパティーです。例えば、types="[range flag]" は、これらのストレージ・タイプのフィールドのみ一覧表示されることを意味します。使用できるデータ型のセットは次のとおりです。

  range

  flag

  set

  orderedSet

  numeric

  discrete

  typeless
- onlyRanges が true に設定されている場合は、範囲データ型のフィールドのみ一覧表示されます。
- onlyDiscrete が true に設定されている場合は、離散的な (フラグ型、セット型、データ型不明の) データ型のみ一覧表示されます。

したがって、例えば storage="[integer]" と types="[flag]" を指定するコントロールでは、フラグ型である整数フィールドのみリストに表示されます。

残りの属性は次の通りです。

- property は、フィールド値を保存するために使用されるプロパティーの識別子です。
- multiple は、フィールド値が列挙リストである (true) か、そうでないか (false) を指定します。
- label は、コントロールの表示名です。
- labelKey は、ローカライズ用のラベルを識別します。

出力フィールド: OutputFields 要素は、ユーザーがモデルの出力フィールド (対象フィールド) を選択できるフィールドのセットを定義します。

セットは、このノードで表示されるすべてのフィールドで構成されています。フィールドがこのノードの上流で詳細にフィルタリングされている場合、フィルターを通過したフィールドのみが表示されます。特定のストレージ・タイプおよびデータ型のフィールドのみが選択できるよう指定することによって、リストをより詳細に制限することができます。

```
<OutputFields storage="storage_types" onlyNumeric="true_false" onlySymbolic="true_false"
     onlyDatetime="true_false" types="data_types" onlyRanges="true_false"
     onlyDiscrete="true_false" property="property_name" multiple="true_false" label="label"
     labelKey="label_key"/>
```

2 つの属性を指定して、出力フィールドとして使用されるフィールドのリストを制限できます。次のリストから選択する必要があります。

- storage は、リストに表示できるフィールドのストレージ・タイプを指定するリスト・プロパティーです。例えば storage="[integer real]" は、これらのストレージ・タイプのフィールドのみ一覧表示されることを意味します。可能なストレージ・タイプのセットについては、 188 ページの『データおよびストレージ・タイプ』 のテーブルを参照してください。
- onlyNumeric が true に設定されている場合、数値型ストレージのフィールドのみが一覧表示されます。
- onlySymbolic が true に設定されている場合、シンボル値のストレージ・タイプのフィールド (文字列) のみが一覧表示されます。
- onlyDatetime が true に設定されている場合、日付と時間のストレージ・タイプのみが一覧表示されます。

2 番目の属性を以下から選択する必要があります。

- types は、リスト内に表示されるフィールドのデータ型を指定するリストのプロパティーです。例えば、types="[range flag]" は、これらのストレージ・タイプのフィールドのみ一覧表示されることを意味します。使用できるデータ型のセットは次のとおりです。

  range

  flag

  set

  orderedSet

  numeric

  discrete

  typeless
- onlyRanges が true に設定されている場合は、範囲データ型のフィールドのみ一覧表示されます。
- onlyDiscrete が true に設定されている場合は、離散的な (フラグ型、セット型、データ型不明の) データ型のみ一覧表示されます。

したがって、例えば storage="[integer]" と types="[flag]" を指定するコントロールでは、フラグ型である整数フィールドのみリストに表示されます。

残りの属性は次の通りです。

- property は、フィールド値を保存するために使用されるプロパティーの識別子です。
- multiple は、フィールド値が列挙リストである (true) か、そうでないか (false) を指定します。

- label は、コントロールの表示名です。
- labelKey は、ローカライズ用のラベルを識別します。

モデル生成

ModelGeneration 要素は、ファイルの別の場所で使用される識別子を指定し、生成されたモデルのモデル名コントロールを含むモデル・ビルダー・ノードのダイアログ・ボックスのタブを定義します。

形式は次のとおりです。

```
<ModelGeneration controlsId="control_identifier" />
```

controlsId 属性は、モデル・ビルダー・ノードの設定の User Interface セクションの SystemControls 要素で一貫して使用される識別子です。設定に SystemControls 要素があるタブには、モデル名コントロールが含まれます。

モデル・フィールド

ModelFields 要素を使用して、モデルによってデータをスコアリングするために使用する入力フィールドおよび出力フィールドのセットであるモデル署名を構築します。

```
<ModelFields inputDirections="[in]" outputDirections="[out]">
     <AddField prefix="field_prefix" ... />
      ...
     <ForEach ... >
          <AddField prefix="field_prefix" ... />
     </ForEach>
      ...
</ModelFields>
```

inputDirections と outputDirections は、モデル署名の構築方法を指定します。有効な値は、in、out、both です。

フィールド自体は、AddField 要素で指定されます。prefix 属性は、フィールド名に追加される接頭辞を指定し、モデルで生成されたフィールドを示します。例えば、フィールド名が field1 で接頭辞が $S の場合、生成されたフィールドの名前は $S-field1 となります。 詳しくは、トピック 65 ページの『フィールドの追加』を参照してください。

ForEach 要素を使用すると、反復が可能です。 詳しくは、トピック 69 ページの『ForEach 要素による反復』を参照してください。

フィールド グループを使用して、モデルから複数の出力フィールドをグループ化して反復することができます。例えば、$S-field1-1、$S-field1-2 など、出力フィールドに反復を示す接尾辞を使用することができます。フィールド グループを 1 つ使用すると、同じセットのフィールドがモデル出力で複数回表示されます。 詳しくは、トピック 89 ページの『フィールド グループの例』を参照してください。

Automodeling

AutoModeling 要素を使用すると、自動分類、自動クラスターまたは自動数値など、アンサンブル・モデル作成ノードによってモデルを使用できるようにします。 詳しくは、トピック 95 ページの『自動化されたモデル作成』を参照してください。

モデル・ビルダーの例

次の例では、反復ノードの例の設定ファイルの完全な Model Builder セクションを示します (28 ページの『モデル・ビルダー・ノード (Interaction)』 を参照)。

```
<Node id="interaction.builder" type="modelBuilder" palette="modeling" label="Interaction">
     <ModelBuilder miningFunctions="[classification]">
          <Algorithm value="robd" label="Robert's Algorithm" />
          <ModelingFields controlsId="modellingFields">
               <InputFields property="inputs" multiple="true" label="Inputs"
                onlyDiscrete="true" />
               <OutputFields property="target" multiple="false" label="Target"
                onlyDiscrete="true" />
          </ModelingFields>
          <ModelFields inputDirections="[in]" outputDirections="[out]">
               <ForEach var="field" inFields="outputs">
                    <AddField prefix="$I" name="${field}" fieldRef="${field}" role=
                     "predictedValue" targetField="${field}" />
                    <AddField prefix="$IP" name="${field}" storage="real" role="probability"
                     targetField="${field}">
                         <Range min="0.0" max="1.0"/>
                    </AddField>
               </ForEach>
          </ModelFields>
     </ModelBuilder>
 ...
</Node>
```

フィールド グループの例

この例は SLRM ノードから取得し、 2 つの新規フィールドのグループをモデル署名に追加して、モデルのスコアリング時に生成されたデータを含みます。各入力レコードについて、新規フィールドのデータは、max_predictions プロパティーの値でユーザーが指定した回数スコアリングされます。

新しい 2 つのフィールドは次のとおりです。

- $S-*target* – 対象フィールドの予測値が格納されます
- $SC-*target* – この予測の確率値が格納されます

これら 2 つのフィールドをグループ化するために、ModelFields セクションで宣言された同じグループの識別子が割り当てられます。グループの識別子が AddField 要素の group 属性によって割り当てられます。

モデル・ビルダー・ノードの宣言には、以下が含まれています。

```
<Node ... type="modelBuilder" ... >
     <ModelBuilder ... >
     ...
          <ModelFields inputDirections="[in]" outputDirections="[out]">
               <AddField prefix="$S" name="${target}" fieldRef="${target}" role=
                "predictedValue" targetField="${target}" group="[1]"/>
               <AddField prefix="$SC" name="${target}" storage="real" role="probability"
                  targetField="${target}" group="[1]">
                    <Range min="0.0" max="1.0"/>
               </AddField>
          </ModelFields>
     </ModelBuilder>
</Node>
```

モデル・アプライヤー・ノードの宣言には、以下が含まれています。

```
<Node ... type="modelApplier" ... >
...
     <OutputDataModel mode="extend">
          <ForEach var="group" from="1" to="${max_predictions}">
```

```
            <ForEach var="field" inFields="modelOutputs" container="model">
                 <AddField name="${field}" group="[${group}]" fieldRef="${field}" />
            </ForEach>
       </ForEach>
  </OutputDataModel>
</Node>
```

対象フィールド名は **campaign** となり、ユーザーは max_predictions プロパティーに対応するフィールドに 2 を入力します。モデル・ビルダー・ノードを実行すると、次のフィールドがモデルに追加されます。

- $S-campaign-1
- $SC-campaign-1
- $S-campaign-2
- $SC-campaign-2

モデル出力

ModelOutput 要素は、ストリーム実行後にマネージャ領域の「モデル」タブに表示されるモデル出力オブジェクトを説明します。

フォーマット

```
<ModelOutput id="identifier" label="display_label" labelKey="label_key" delegate="Java_class" >
     <ModelProvider ... />
     <Properties>
          <Property ... />
           ...
     </Properties>
     <Containers ... />
     <UserInterface ... />
     <Constructors ... />
</ModelOutput>
```

ここで、

- id (必須) は、生成されるモデルの一意の識別子です。
- label (必須) は、「モデル」タブに表示される、生成されたモデルの表示名です。
- labelKey は、ローカライズ用のラベルを識別します。

ModelOutput 要素内に格納できる子要素を以下の表に示します。

表 *31.* モデル出力の宣言の子要素：

子要素	定義	参照...
ModelProvider	モデル出力を保持するコンテナー、および出力が PMML 形式かどうかを定義します。	52 ページの『モデル・プロバイダ』
Properties	生成されたモデルに使用されるプロパティー。	52 ページの『Properties』
Containers	生成されたモデルの出力が格納されるコンテナー。	54 ページの『コンテナー』
UserInterface	モデル出力オブジェクトの「モデル」タブおよび「設定」タブなど、生成されたモデルの出力を表示できるユーザー・インターフェース。	55 ページの『ユーザー・インターフェース』
Constructors	生成されたモデルによって作成されたオブジェクト。	103 ページの『コンストラクターの使用』

表 31. モデル出力の宣言の子要素 (続き) :

子要素	定義	参照...
delegate	指定した場合、OutputDelegate インターフェースを実行する Java クラスの名前が定義されます。指定されたクラスのインスタンスが、関連する出力の各インスタンスに対して構築されます。	

例

```
<ModelOutput id="interaction.model" label="Interaction Model">
     <Properties>
     </Properties>
     <Containers>
          <Container name="model" />
     </Containers>
     <UserInterface>
          <Tabs>
               <Tab label="Model">
                    <TextBrowserPanel container="model" textFormat="plainText" />
               </Tab>
          </Tabs>
     </UserInterface>
     <Constructors>
          <CreateModelApplier type="interaction.applier">
               <SetContainer target="model" source="model" />
          </CreateModelApplier>
     </Constructors>
</ModelOutput>
```

インタラクティブ・モデルの構築

インタラクティブ・モデルの構築は、モデル生成前にエンド・ユーザーが対話処理できる出力オブジェクトを作成できる機能です。このインタラクティブ 出力オブジェクトは、マネージャ領域の「出力」タブに配置され、中間データ・セットが含まれます。中間データ・セットを使用して、モデルが生成される前にモデルを生成または単純化できます。インタラクティブ・モデルの構築は、通常のモデル・ビルダー・ノードの設定に特別な要素を追加して実現されます。

- Node 定義の Constructors セクションには CreateInteractiveModelBuilder 要素が定義されています。
- 拡張には、専用の InteractiveModelBuilder 要素が含まれます。

対話ウィンドウというウィンドウを使用して、ユーザーは中間データ・セットと対話処理します。対話ウィンドウは出力オブジェクトが作成されるとすぐに表示されます。

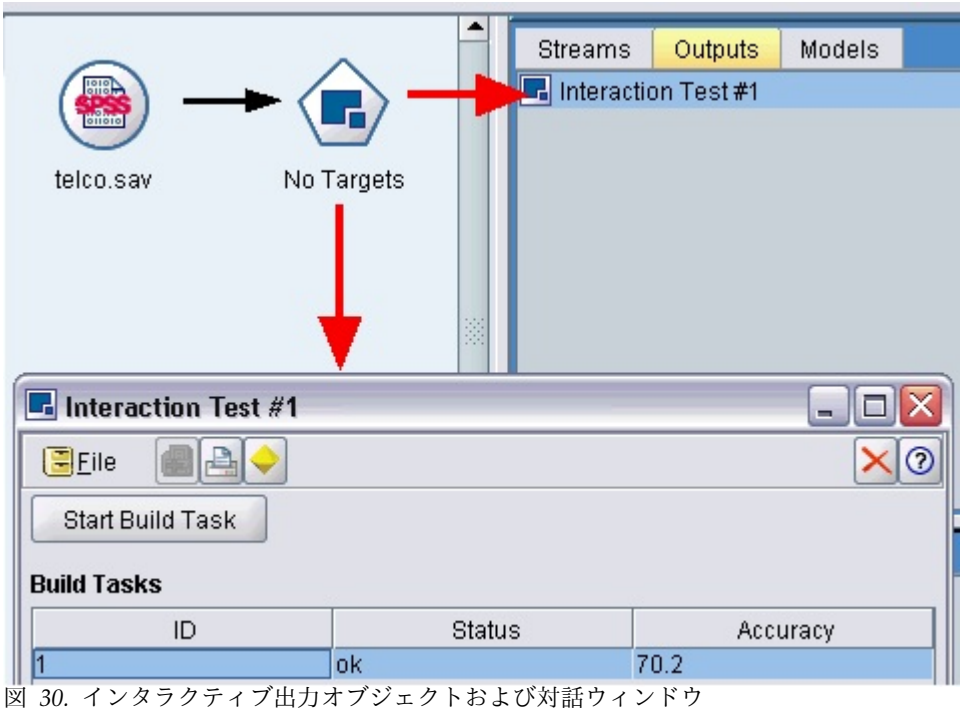

図 30. インタラクティブ出力オブジェクトおよび対話ウィンドウ

対話処理は、拡張によって使用され、実行されるアルゴリズムに固有のものです。対話ウィンドウは、InteractiveModelBuilder 要素の User Interface セクションに定義されます。次のいずれかを指定して、対話ウィンドウを定義できます。

- ウィンドウを完全に定義するフレーム・クラス (108 ページの『User Interface セクション』を参照)
- ウィンドウの各タブの、拡張オブジェクトパネルの属性として指定されるパネル・クラス (121 ページの『拡張オブジェクト・パネル』を参照)

ウィンドウを閉じた後、「出力」タブのオブジェクト名をダブルクリックして対話ウィンドウを再表示することができます。

対話ウィンドウの設定には、ユーザーが対話を終了した後モデルを生成するコードを使用する必要があります。説明されている例では、モデル生成の動作と関連する金色のナゲット・アイコンを含むツールバー・ボタンを使用して実行されています。このコードは 94 ページの『インタラクティブ・モデル構築の例』 の InteractiveModelBuilder セクションに表示されています。

インタラクティブ・モデル・ビルダーの作成

CreateInteractiveModelBuilder 要素は、ユーザーが対話処理する出力オブジェクトを説明します。 これは CreateModelOutput 要素のインタラクティブ・バージョンです。

フォーマット

この要素は、次のように、モデル・ビルダー・ノードの定義の Execution セクションで使用されます。

```
<Node ... type="modelBuilder" ... >
 ...
    <Execution>
     ...
        <Constructors>
         ...
            <CreateInteractiveModelBuilder ... >
              ...
```

```
            </CreateInteractiveModelBuilder>
        </Constructors>
    </Execution>
 ...
</Node>
```

要素自体の形式は次のとおりです。

```
<CreateInteractiveModelBuilder type="output_object_id">
    <Condition ... ./>
    <And>
    <Or>
    <Not>
    <CreateModel type="model_id" target="container_id" sourceFile="container_file_id" />
    <CreateDocument type="model_id" target="container_id" sourceFile="container_file_id" />
</CreateInteractiveModelBuilder>
```

ここで、type (必須) は InteractiveModelBuilder 要素で作成された出力オブジェクトの識別子です。

Condition セクションを使用すると、条件を指定できます。 詳しくは、トピック 73 ページの『条件』を参照してください。

And、Or および Not の演算子を含む複雑な条件を指定することもできます。 詳しくは、トピック 73 ページの『論理演算子』を参照してください。

CreateModel 要素と CreateDocument 要素には、以下の識別子があります。

- type は、定義対象のモデルまたはドキュメントの識別子です。
- target (必須) は、モデルのコンテナーの識別子です。このコンテナーは、モデル出力セクションで定義されます。詳しくは、トピック 90 ページの『モデル出力』を参照してください。
- sourceFile (必須) は、ノード実行中に生成される出力ファイルの識別子です。このファイルは、出力ファイル・セクションで定義されます。 詳しくは、トピック 57 ページの『出力ファイル』を参照してください。

例

```
<CreateInteractiveModelBuilder type="my.interaction">
    <Condition property="interactive" op="equals" value="true" />
</CreateInteractiveModelBuilder>
```

この例では、識別子 my.interaction を持つ出力オブジェクトが設定に InteractiveModelBuilder 要素が含まれるモデル・ビルダー・ノードの実行時に作成されるよう指定しています。出力オブジェクト自体は、この識別子を参照する InteractiveModelBuilder 要素により、仕様ファイルの別の場所で定義されます。以下に例を示します。

```
<InteractiveModelBuilder id="my.interaction" label=...>
 ...
</InteractiveModelBuilder>
```

インタラクティブ・モデル・ビルダー

この要素は、インタラクティブ出力オブジェクトを定義し、エンド・ユーザーは生成される前にモデルを調整または単純化できます。

InteractiveModelBuilder 要素は、 対応する CreateInteractiveModelBuilder 要素を含むモデル・ビルダー・ノードの定義に従います。 詳しくは、トピック 92 ページの『インタラクティブ・モデル・ビルダーの作成』を参照してください。

フォーマット

InteractiveModelBuilder 要素の形式は次のとおりです。

```
<Node ... type="modelBuilder" ... >
 ...
     -- Create Interactive Model Builder section --
 ...
</Node>
 ...
<InteractiveModelBuilder id="identifier" label="display_label" labelKey="label_key">
     <Properties>
          <Property name=... />
           ...
     </Properties>
     <Containers>
          <Container name="container_name"/>
     </Containers>
     <UserInterface ... />
     <Constructors ... />
</InteractiveModelBuilder>
```

ここで、

- id (必須) は、生成されるモデルの一意の識別子です。
- label (必須) は、「モデル」タブに表示される、生成されたモデルの表示名です。
- labelKey は、ローカライズ用のラベルを識別します。

Properties 要素、Containers 要素、UserInterface 要素、Constructors 要素の詳細については、52 ページの『Properties』、 54 ページの『コンテナー』、 108 ページの『User Interface セクション』、と 103 ページの『コンストラクターの使用』を参照してください。

インタラクティブ・モデル構築の例

この例は、モデルの生成前に対話処理を行うかどうかを単純なチェック・ボックスで選択できるようにモデル・ビルダー・ノードを定義する方法を示しています。

実際にこの動作を表示するには、このリリースで提供された Interaction のサンプルノードを使用してください。 詳しくは、トピック 28 ページの『モデル・ビルダー・ノード (Interaction)』を参照してください。

まず、モデル・ビルダー・ノードはブール・プロパティーを指定します。

```
<Node id="interaction.builder" type="modelBuilder" ... >
 ...
     <Properties>
      <Property name="interactive" valueType="boolean" />
    </Properties>
```

ノード設定の User Interface セクションでは、「モデル」タブを定義するセクションにはこのプロパティーへの参照が含まれます。

```
<Tab label="Model">
     <PropertiesPanel>
          <CheckBoxControl property="interactive" label="Start an interactive session" />
     </PropertiesPanel>
</Tab>
```

同じノードの CreateInteractiveModelBuilder セクションでは、プロパティーの設定が検定され、true の場合はインタラクティブ出力オブジェクトが次のように作成されます。

```
<CreateInteractiveModelBuilder type="my.interaction">
     <Condition property="interactive" op="equals" value="true" />
</CreateInteractiveModelBuilder>
```

参照される出力オブジェクトは、拡張の InteractiveModelBuilder セクションに定義されます。

```
<InteractiveModelBuilder id="my.interaction" label="Interaction Test">
     <Properties>
     </Properties>
     <Containers>
     </Containers>
     <UserInterface actionHandler="ui.InteractionHandler">
          <Controls>
               <ToolbarItem action="generateModel" showLabel="false" />
          </Controls>
          <Tabs>
               <Tab label="Model">
                    <ExtensionObjectPanel id="model.panel" panelClass=
                     "ui.SampleInteractionPanel" />
               </Tab>
               <Tab label="Generic">
                    <ExtensionObjectPanel id="generic.panel" panelClass=
                     "ui.GenericInteractionPanel" />
               </Tab>
          </Tabs>
     </UserInterface>
</InteractiveModelBuilder>
```

モデル生成の操作は、ToolbarItem 要素に定義されたツールバー・ボタンに制御されます。

ExtensionObjectPanel 要素の panelClass 属性を使用すると、対話ウィンドウのそれぞれのタブのユーザー・インターフェースを制御する Java クラスを指定します。

自動化されたモデル作成

IBM SPSS Modeler では、自動分類ノード、自動クラスター・ノード、または自動数値ノードなど、アンサンブル・モデル構築ノードのグループを標準として提供します。これらのノードは、あらゆるモデルの同時構築を自動化し、エンド・ユーザーは結果を比較し、データの最良のモデルを選択できます。CLEF では、ModelBuilder 要素に指定されたモデルをこれらのアンサンブルモードで使用できる AutoModeling 要素を提供しています。

AutoModeling 要素の形式は次のとおりです。

```
<AutoModeling enabledByDefault="true_false">
     <SimpleSettings ... />
     <ExpertSettings ... />
     <Constraint ... />
     <Constraint ... />
      ...
</AutoModeling>
```

この場合、enabledByDefault が、モデルをアンサンブル・モデル作成ノードのデフォルトで使用できるかどうかを指定します (つまり、「使用**?**」 列が特定モデルの場合デフォルトでチェックされています)。この属性が省略されている場合、値 true が想定されます。

アンサンブル・モデリング・ノードのダイアログ・ボックスの「エキスパート」タブに、ユーザーが選択および構築できるモデルが表示されます。

ユーザーがモデル タイプのオプションを選択できる場合、特定モデルの 「モデル・パラメーター」 フィールドの 「指定」 をクリックすると、「アルゴリズム 設定」ダイアログ・ボックスが表示されます。

「アルゴリズムの設定」ダイアログ・ボックスには「シンプル」タブ および「エキスパート」タブが含まれ、モデル構築ノードのシンプル実行モードおよびエキスパート実行モードに対応しています。「シンプル」タブおよび「エキスパート」タブの内容は、SimpleSettings 要素および ExpertSettings 要素に制御されます。これらの要素は後続のセクションで説明されています。

さらに、Constraint 要素を使用して条件を指定し、エンド・ユーザーが「アルゴリズムの設定」ダイアログ・ボックスのパラメーターを編集または何らかの方法で制限できるようにします。 詳しくは、トピック 99 ページの『制約』を参照してください。

「アルゴリズムの設定」ダイアログボックスのパラメーターには、複数の値をとるものもあります。複数の値が指定される場合、アンサンブル・ノードはパラメーター値の可能なすべての組み合わせに対するモデル構築を試みます。例えば一般化線型モデルユーザーが 2 つの分布 (正規分布およびガンマ分布) と 3 つのリンク関数 (恒等式、対数、べき乗) を指定する場合、自動数値ノードはこれらのパラメーターの可能な組み合わせに 1 つずつ、6 つの一般化線型モデルの構築を試みます。

シンプル設定

SimpleSettings 要素は、アンサンブル・モデリング・ノードのこのモデルの「アルゴリズムの設定」ダイアログ・ボックスの「シンプル」タブに表示されるパラメーターを決定します。詳しくは、トピック 95 ページの『自動化されたモデル作成』を参照してください。

フォーマット

```
<SimpleSettings>
     <PropertyGroup label="group_name" labelKey="resource_key" properties="[prop_name1
      prop_name2 ...]"/>
     <PropertyGroup ... />
</SimpleSettings>
```

PropertyGroup 要素 (最低 1 つは必須) では次のようになります。

label は、プロパティー・グループの表示ラベルで、グループの最初のパラメーターの前にある小見出しとしてダイアログ・ボックスに挿入されます。

labelKey は、ローカライズ用のラベルを識別します。label も labelKey も使用されていない場合、小見出しは挿入されません。

properties (必須) は、タブに表示されるプロパティーのリストです。*prop_name1*、*prop_name2* などの値は、このプロパティーが定義される Property 要素の name 属性の値です。 詳しくは、トピック 52 ページの『Properties』を参照してください。

例

```
<SimpleSettings>
     <PropertyGroup properties="[method]"/>
</SimpleSettings>
```

判別ノードのこの例では、関連するアンサンブル・モデル構築ノード (この場合、自動分類ノード) 内にある一般化線型モデルの「アルゴリズムの設定」ダイアログ・ボックスの「シンプル」タブに 方法パラメー

ターだけが表示されるようにします。label 属性または labelKey 属性が指定されていない場合、ダイアログ・ボックスにパラメーターの小見出しは表示されません。

エキスパート設定

ExpertSettings 要素は、アンサンブル・モデル構築ノードの一般化線型モデルの「アルゴリズムの設定」ダイアログ・ボックスの「エキスパート」タブに表示されるパラメーターを決定します。 詳しくは、トピック 95 ページの『自動化されたモデル作成』を参照してください。

フォーマット

```
<ExpertSettings>
     <Condition ... />
     <PropertyGroup label="group_name" labelKey="resource_key"
      properties="[property1 property2 ...]"/>
     <PropertyGroup ... />
      ...
</ExpertSettings>
```

true の場合、Condition 要素は、後続の PropertyGroup 要素または有効化される要素に識別されるパラメーターを使用する条件を指定します。 詳しくは、トピック 73 ページの『条件』を参照してください。

PropertyGroup 要素 (最低 1 つは必須) では次のようになります。

label は、プロパティー・グループの表示ラベルで、グループの最初のパラメーターの前にある小見出しとしてダイアログ・ボックスに挿入されます。

labelKey は、ローカライズ用のラベルを識別します。label も labelKey も使用されていない場合、小見出しは挿入されません。

properties (必須) は、タブに表示されるプロパティーのリストです。*prop_name1*、*prop_name2* などの値は、このプロパティーが定義される Property 要素の name 属性の値です。 詳しくは、トピック 52 ページの『Properties』を参照してください。

例

次の例では、「アルゴリズムの設定」ダイアログ・ボックスの「エキスパート」タブに、最初は「シンプル」に設定される モード パラメーターが含まれます。

図 *31.* エキスパート設定の無効化

以下は、「エキスパート」 に設定する モード パラメーターが変更される場合にのみ、他の「エキスパート」タブのパラメーターが有効化されるよう指定します。

```
<ExpertSettings>
     <Condition property="mode" op="equals" value="Expert"/>
     <PropertyGroup properties="[mode prior_probabilities covariance_matrix]"/>
      ...
</ExpertSettings>
```

「モード」の設定を「エキスパート」に変更すると、プロパティー・グループの 2 つのパラメーターが有効になります。

図 32. エキスパート設定の有効化

次の例では、プロパティー・グループのラベルの使用を説明します。

```
<ExpertSettings>
      ...
     <PropertyGroup labelKey="automodel.stepping_criteria_options"
      properties="[stepwise_method V_to_enter criteria]"/>
      ...
</ExpertSettings>
```

PropertyGroup 要素は、以下の図で強調表示されているパラメーターを制御します。

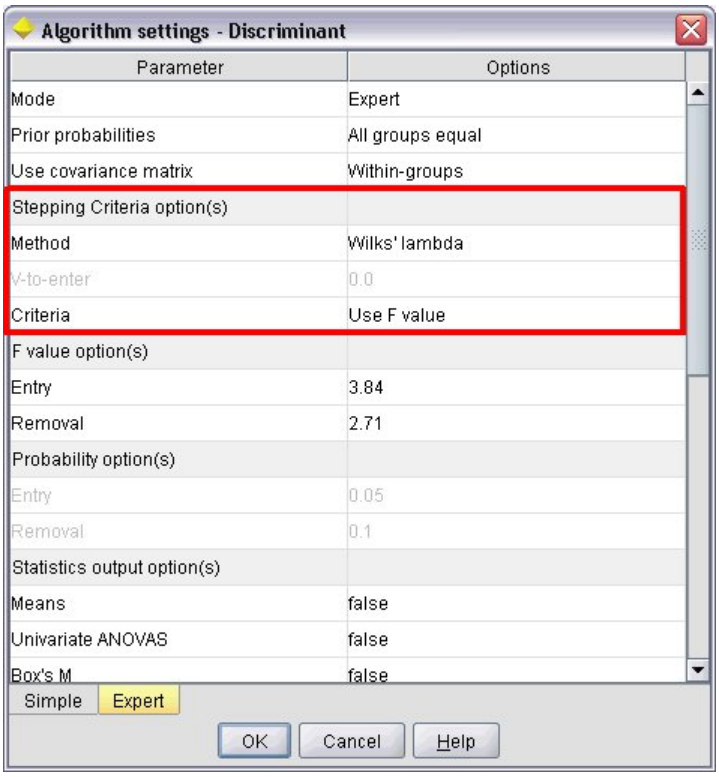

図 33. エキスパート設定の無効化

labelKey 属性を使用すると、CLEF は拡張のプロパティー・ファイル内の対応エントリーから、プロパティー・グループの小見出しに使用する表示テキストを取得できます。

```
automodel.stepping_criteria_options=Stepping Criteria option(s)
```

詳しくは、トピック 172 ページの『プロパティー・ファイル』を参照してください。

制約

Constraint 要素は、アンサンブル・モデル構築ノードのモデルの「アルゴリズムの設定」ダイアログ・ボックスに表示されたパラメーターの編集、またはなんらかの制約を許可する制約を指定します。例えば、エンド・ユーザーが現在パラメーターの変更を許可されていない場合、特定のパラメーターを無効化できます。

フォーマット

```
<Constraint property="prop_name" singleSelection="true_false">
     <Condition property="prop_name" op="operator" value="value"/>
     <And ... />
     <Or ... />
     <Not ... />
</Constraint>
```

ここで、

- property (必須) は編集または制限するパラメーターを識別します。*prop_name* はこのパラメーターに対応するプロパティーが定義されている Property 要素の name 属性の値です。 詳しくは、トピック 52 ページの『Properties』を参照してください。
- singleSelection は、エンド・ユーザーがパラメーターに使用可能な値を複数選択できるかどうかを制御します。true に設定されていると、複数の値が「アルゴリズムの設定」ダイアログ・ボックスの該当するパラメーターの「オプション」フィールドに表示されている場合であっても、選択できる値は 1 つだけです。false (デフォルト) に設定されていると、次の例で説明されているとおり、使用可能な値から 1 つまたは複数の値を選択できます。

Condition 要素は、実際の制約を指定します。 詳しくは、トピック 73 ページの『条件』を参照してください。

And、Or、および Not 要素を使用して、複合条件を指定できます。 詳しくは、トピック 73 ページの『論理演算子』を参照してください。

例

この例は、一般化線型ノードの設定ファイルから取得しています。エキスパート・モードの場合も、一部のパラメーターはデフォルトでは有効化されていません。

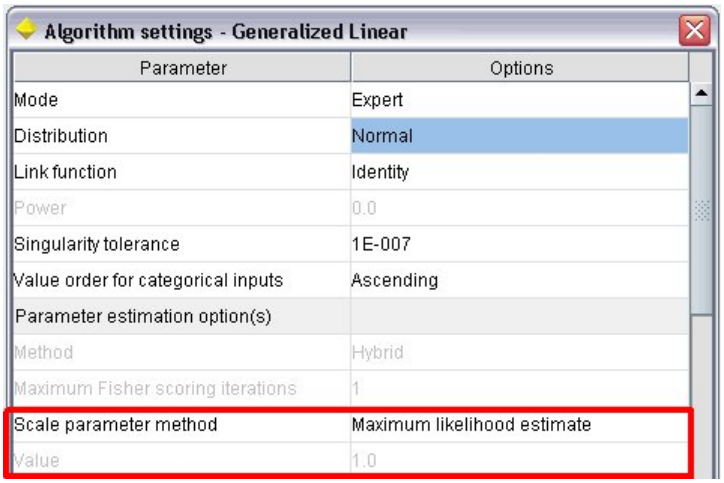

図 *34.* 制約の効果 — 無効になるパラメーター

この制約は、値パラメーターが有効化される条件を指定します。

```
<Constraint property="scale_value">
     <And>
          <Condition property="scale_method" op="equals" value="FixedValue"/>
          <Condition property="distribution" op="in" value="[IGAUSS GAMMA NORMAL]"/>
     </And>
</Constraint>
```

値パラメーターが有効化される前に、スケール パラメーター方法パラメーター (scale_method プロパティーで識別) は、「固定値」 に設定し、分布は、「正規分布」、「逆ガウス分布」 または 「ガンマ」 に設定する必要があります。

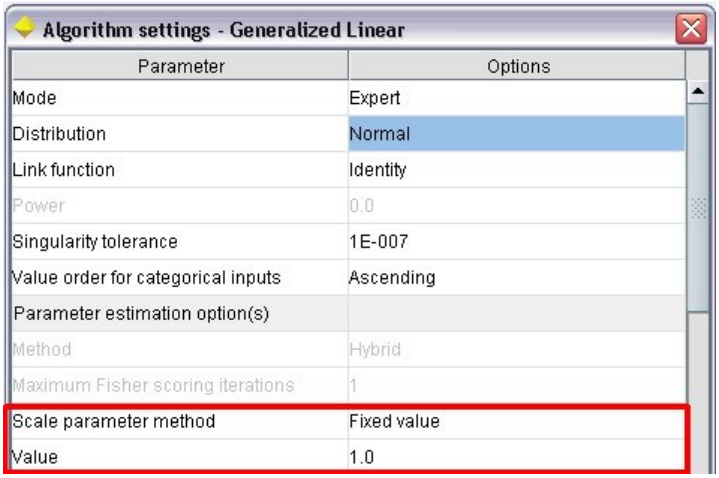

図 *35.* 制約の効果 — 有効になるパラメーター

モデルの適用

モデルの適用とは、生成されたモデルを使用してデータをスコアリングする、つまり新規レコードの予測を作成するためのモデル構築から取得した情報を使用することです。IBM SPSS Modeler では、モデル・アプライヤー・ノードを使用してモデルを適用します。 詳しくは、トピック 12 ページの『モデル・アプライヤー・ノード』を参照してください。

設定ファイルのモデル・アプライヤー・ノードを定義すると、生成されたモデルを適用するためのフレームワークを作成します。IBM SPSS Modeler では、モデル出力オブジェクトを示すアイコンをマネージャ領域の「モデル」タブからストリーム領域にドラッグして、モデル・アプライヤー・ノードのインスタンスを作成します。モデル・アプライヤー・ノードが定義されていない場合、モデル・ビルダー・ノードを実行すると未調整のモデルを生成します。このモデルをストリーム領域に追加することはできません。

モデル・アプライヤー・ノードを定義する場合、Node 要素には以下の情報を含める必要があります。

- type="modelApplier" 属性
- CreateModelOutput 要素を含む Constructors 子要素 (103 ページの『コンストラクターの使用』を参照)

Node 要素の指定形式については、 48 ページの『ノード』を参照してください。

ドキュメントの構築

ドキュメント・ビルダー・ノードを定義する場合、Node 要素に必要なものは次のとおりです。

- type="documentBuilder" 属性
- DocumentBuilder 子要素

ドキュメント構築において、拡張には生成されたモデルを説明するために DocumentOutput 要素も必要です。 詳しくは、トピック 90 ページの『モデル出力』を参照してください。

Node 要素の指定形式については、 48 ページの『ノード』を参照してください。

ドキュメント・ビルダー

DocumentBuilder 要素は、ドキュメント・ビルダー・ノードの動作を定義します。定義では DocumentGeneration 子要素を使用し、ドキュメント生成コントロールを含むドキュメント・ビルダー・ノードのダイアログ・ボックスのタブを指定する必要があります。コントロール自体は、User Interface セクションで定義されます (107 ページの『第 6 章 ユーザー・インターフェースの構築』 を参照)。

フォーマット

```
<DocumentBuilder>
     <DocumentGeneration controlsId="control_identifier />
</DocumentBuilder>
```

この場合、controlsId (必須) は、ドキュメント生成コントロールが表示される場所を指定するシステムコントロールによって使用される識別子です。

例

```
<DocumentBuilder>
     <DocumentGeneration controlsId="1"/>
</DocumentBuilder>
```

ドキュメント出力

DocumentOutput 要素は、ストリームの実行後にマネージャー・ペインの「出力」タブに表示されるドキュメント出力オブジェクトを記述します。

フォーマット

```
<DocumentOutput id="identifier" label="display_label" labelKey="label_key" delegate="Java_class" >
     <Properties>
          <Property ... />
           ...
     </Properties>
     <Containers>
          <Container ... />
           ...
     </Containers>
     <UserInterface ... />
     <Constructors ... />
</DocumentOutput>
```

ここで、

- id (必須) は、生成されるドキュメントの一意の識別子です。
- label (必須) は、「出力」タブに表示するときの、生成されたドキュメントの表示名です。
- labelKey は、ローカライズ用のラベルを識別します。

DocumentOutput 要素内に格納できる子要素を以下の表に示します。

表 32. ドキュメント出力の宣言の子要素：

子要素	定義	参照...
Properties	生成されたドキュメントに使用されるプロパティー。	52 ページの『Properties』
Containers	生成されたドキュメントの出力が格納されるコンテナー。	54 ページの『コンテナー』
UserInterface	生成されたドキュメント出力が表示できるユーザー・インターフェース。	55 ページの『ユーザー・インターフェース』
Constructors	生成されたドキュメントによって作成されたオブジェクト。	103 ページの『コンストラクターの使用』
delegate	指定した場合、OutputDelegate インターフェースを実行する Java クラスの名前が定義されます。指定されたクラスのインスタンスが、関連する出力の各インスタンスに対して構築されます。	

例

```
<DocumentOutput id="webstatusreport">
     <Containers>
          <Container name="webstatusreportdata" />
     </Containers>
     <UserInterface>
          <Tabs>
               <Tab label="Advanced" labelKey="advancedTab.LABEL" >
                    <TextBrowserPanel container="webstatusreportdata" textFormat="html" />
```

```
            </Tab>
        </Tabs>
    </UserInterface>
</DocumentOutput>
```

コンストラクターの使用

Constructors 要素は、設定ファイル内の次の 2 つの場所のいずれかに使用できます。

- ノード定義の Execution セクション (モデルまたはドキュメント出力オブジェクトの場合)
- モデル出力定義 (モデル・アプライヤー・ノードの場合)

1 つのノードで生成できる出力オブジェクトは 1 つだけです。この制限は、既存のノード既存のノードに対するスクリプトおよびクライアント API インターフェースとの一貫性については必要ありません。.

フォーマット

Execution セクションで使用される場合、Constructors 要素の形式は次のようになります。

```
<Constructors>
    <CreateModelOutput ... />
     ...
    <CreateDocumentOutput ... />
     ...
    <CreateInteractiveModelBuilder ... />
     ...
</Constructors>
```

モデル出力定義の場合、形式は次のようになります。

```
<Constructors>
    <CreateModelApplier ... />
</Constructors>
```

モデル出力の作成

この項では、「モデル」タブのモデル出力オブジェクトの作成方法、または「出力」タブのドキュメント出力オブジェクトの作成方法を定義します。

フォーマット

```
<CreateModelOutput type="output_object_id">
    <Condition ... ./>
    <And>
    <Or>
    <Not>
    <CreateModel type="model_id" target="container_id" sourceFile="container_file_id" />
     ...
    <CreateDocument type="document_id" target="container_id" sourceFile="container_file_id" />
     ...
</CreateModelOutput>
```

CreateModelOutput 要素では、以下のようになります。

- type (必須) は、Model Output セクションで定義されるモデル出力オブジェクトの識別子です。 詳しくは、トピック 51 ページの『モデル出力』を参照してください。

Condition セクションを使用すると、条件を指定できます。 詳しくは、トピック 73 ページの『条件』を参照してください。

And、Or、および Not の演算子を含む複雑な条件を指定することもできます。 詳しくは、トピック 73 ページの『論理演算子』を参照してください。

CreateModel 要素と CreateDocument 要素には、以下の識別子があります。

- type は、定義対象のモデルまたはドキュメントの識別子です。
- target (必須) は、モデルのコンテナーの識別子です。このコンテナーは、モデル出力セクションで定義されます。詳しくは、トピック 90 ページの『モデル出力』を参照してください。
- sourceFile (必須) は、ノード実行中に生成される出力ファイルの識別子です。このファイルは、出力ファイル・セクションで定義されます。 詳しくは、トピック 57 ページの『出力ファイル』を参照してください。

例

```
<Constructors>
     <CreateModelOutput type="naivebayes">
          <CreateModel type="naivebayes_model" target="model" sourceFile="pmml"/>
          <CreateDocument type="html_output" target="advanced_output" sourceFile="htmloutput" />
          <CreateDocument type="zip_outputType" target="zip_output" sourceFile="zipoutput" />
     </CreateModelOutput>
</Constructors>
```

ドキュメント出力の作成

この要素は、ドキュメント・ビルダー・ノードの定義の Execution セクションで使用され、作成されているドキュメント出力オブジェクトを識別します。

フォーマット

```
<CreateDocumentOutput type="output_object_id">
     <Condition ... ./>
     <And>
     <Or>
     <Not>
     <CreateModel type="model_id" target="container_id" sourceFile="container_file_id" />
      ...
     <CreateDocument type="document_id" target="container_id" sourceFile="container_file_id" />
      ...
</CreateModelOutput>
```

type (必須) は、Document Output セクションで定義されるドキュメント出力オブジェクトの識別子です。詳しくは、トピック 51 ページの『ドキュメント出力』を参照してください。

Condition セクションを使用すると、条件を指定できます。 詳しくは、トピック 73 ページの『条件』を参照してください。

And、Or、および Not の演算子を含む複雑な条件を指定することもできます。 詳しくは、トピック 73 ページの『論理演算子』を参照してください。

CreateModel 要素と CreateDocument 要素には、以下の識別子があります。

- type は、定義対象のモデルまたはドキュメントの識別子です。
- target (必須) は、モデルのコンテナーの識別子です。このコンテナーは、モデル出力セクションで定義されます。詳しくは、トピック 90 ページの『モデル出力』を参照してください。

- sourceFile (必須) は、ノード実行中に生成される出力ファイルの識別子です。このファイルは、出力ファイル・セクションで定義されます。 詳しくは、トピック 57 ページの『出力ファイル』を参照してください。

例

```
<Constructors>
     <CreateDocumentOutput type="webstatusreport">
          <CreateDocument type="webstatusreport" target="webstatusreportdata"
               sourceFile="webstatusreport_output_file" />
     </CreateDocumentOutput>
</Constructors>
```

インタラクティブ・モデル・ビルダーの作成

この要素は、インタラクティブ・モデル・ビルダー・ノードの定義の Execution セクションで使用され、ユーザーが対話処理する出力オブジェクトを識別します。 詳しくは、トピック 92 ページの『インタラクティブ・モデル・ビルダーの作成』を参照してください。

モデル・アプライヤーの作成

この要素は、モデル・ビルダー・ノードの定義の Model Output セクションの Constructors 要素内で使用されます (90 ページの『モデル出力』 を参照)。CreateModelApplier 要素は、モデル・ビルダー・ノードで生成されたモデル出力オブジェクトが領域内に配置された場合のモデル・アプライヤー・ノードの作成方法を定義します。

フォーマット

```
<CreateModelApplier type="apply_node_identifier">
     <Condition ... ./>
     <And>
     <Or>
     <Not>
     <CreateModel type="model_id" target="container_id" sourceFile="container_file_id" />
      ...
     <CreateDocument type="document_id" target="container_id" sourceFile="container_file_id" />
      ...
</CreateModelApplier>
```

CreateModelApplier 要素では、以下のようになります。

- type (必須) は、作成されるモデル・アプライヤー・ノードの識別子です。このノードは、実際には、ファイル内でこれより後に記述する Node ... type="modelApplier" 要素で定義されます。

Condition セクションを使用すると、条件を指定できます。 詳しくは、トピック 73 ページの『条件』を参照してください。

And、Or および Not の演算子を含む複雑な条件を指定することもできます。 詳しくは、トピック 73 ページの『論理演算子』を参照してください。

CreateModel 要素と CreateDocument 要素には、以下の識別子があります。

- type は、定義対象のモデルまたはドキュメントの識別子です。
- target (必須) は、モデルのコンテナーの識別子です。このコンテナーは、モデル出力セクションで定義されます。詳しくは、トピック 90 ページの『モデル出力』を参照してください。

- sourceFile (必須) は、ノード実行中に生成される出力ファイルの識別子です。このファイルは、出力ファイル・セクションで定義されます。 詳しくは、トピック 57 ページの『出力ファイル』を参照してください。

例

次の例では、CreateModelApplier 要素に myapplier という名前の付いたモデル・アプライヤー・ノードへの前方参照が含まれます。このノードは、後続の Node 要素に定義されます。

```
<ModelOutput>
     <Constructors>
          <CreateModelApplier type="myapplier"></CreateModelApplier>
     </Constructors>
</ModelOutput>
<Node id="myapplier" type="modelApplier">
 ...
</Node>
```

第 6 章 ユーザー・インターフェースの構築

ユーザー・インターフェースについて

新しい CLEF ノードを追加する場合、エンド・ユーザーが新しいノード、モデル出力、ドキュメント出力をどのように対話処理し、および拡張固有のノードをどのように適用するかを定義する必要があります。各オブジェクトのユーザー・インターフェースは、拡張の仕様ファイルの UserInterface ファイルで定義されます。この章では、UserInterface セクションについて詳細に説明しています。

注：ファイルに定義されているオブジェクトの種類によって、1 つの仕様ファイルに複数の UserInterface セクションがある場合があります。

CLEF オブジェクトのユーザー・インターフェースは、次のうち 1 つまたは複数で構成されています。

- メニュー・エントリー
- パレット・エントリー
- アイコン
- 1 つまたは複数のダイアログ ウィンドウ
- 1 つまたは複数の出力ウィンドウ

メニュー・エントリーおよびパレット・エントリーを使用すると、IBM SPSS Modeler メニュー・システムおよびノード・パレットからそれぞれアクセスできます。アイコンは、メニュー、描画領域、およびさまざまなノード・パレットのオブジェクトを識別します。ダイアログ ウィンドウには、ストリームが実行される前にさまざまなオプションを指定できる、またオプションで実行か完了した場合に作成される出力を指定できるタブおよびコントロールが含まれます。出力ウィンドウは、モデル出力 (モデルをデータのセットに適用した結果など) またはドキュメント出力 (グラフなど) を表示する場合に使用されます。

CLEF を使用すると、IBM SPSS Modeler で提供された標準的なオブジェクトに新しい種類のオブジェクトを追加し、これらの新しいオブジェクトのユーザー・インターフェースを定義する一貫した手法をサポートできます。 14 ページの『ノード・アイコンのデザイン』と 19 ページの『ダイアログ・ボックスのデザイン』で、CLEF でのアイコンとダイアログの作成に関するガイドラインについて説明しています。

アイコンは、特定のノードを識別するグラフィックの形式で、ノードの種類を識別する幾何学的図形の内部に表示されます。

ダイアログおよび出力ウィンドウの特性は次のとおりです。

- 縮小アイコンおよびオブジェクトのタイトルが表示されたタイトル・バー
- メニュー・バーに表示されるのは次のとおりです。
  - メニュー
  - オブジェクト固有のアクション・ボタン
  - 一般的な操作ボタン (「最大化」または「ヘルプ」)
- メイン・コンテンツ領域
- UI コンポーネントを論理グループに編成する複数のタブ
- サイズ変更機能

タブを使用して、ウィンドウのメイン・コンテンツ領域をさまざまな方法で変更します。ダイアログ ウィンドウの場合、さまざまなタブにオブジェクト・プロパティーの異なるセットのコントロールが表示されます。コントロールは変更でき、ユーザーが 「適用」 または 「**OK**」 ボタンをクリックするとその結果が基本オブジェクトに適用されます。

出力ウィンドウの場合、タブを使用するとユーザーは結果の要約を表示または注釈を結果に追加するなど、出力に関連するさまざまな操作を実行することができます。

User Interface セクション

オブジェクトのユーザー・インターフェースは仕様ファイルのオブジェクト定義内の `UserInterface` 要素で宣言されています。ファイルに設定されているオブジェクト (モデル・ビルダー・ノード、モデル出力オブジェクト、モデル・アプライヤー・ノード) の数によって、同一仕様ファイルには複数の `UserInterface` 要素が存在する場合があります。

それぞれの User Interface セクション内で定義できるのは次のとおりです。

- 領域またはパレットに表示するアイコン
- ダイアログまたは出力ウィンドウに表示するコントロール (カスタム メニューおよびツールバーの項目)
- プロパティー・コントロールのセットを定義するタブ (ダイアログまたは出力ウィンドウ)

注 : この後の要素の定義 (通常は「フォーマット」という見出しが付いています) では、「(必須)」と記載されていない限り、要素の属性と子要素はオプションです。すべての要素の詳細な構文については、 209 ページの『CLEF XML スキーマ』を参照してください。

各ユーザー・インターフェースについて、実行されるプロセスを指定します。オプションである操作ハンドラまたはフレーム・クラス属性のいずれかによって実行できます。操作ハンドラおよびフレーム・クラス属性のいずれも指定されていない場合、実行されるプロセスはファイル内の別の場所で指定されます。

フォーマット

`UserInterface` 要素の基本的な形式は次のとおりです。

```
<UserInterface>
     <Icons>
          <Icon ... />
           ...
     </Icons>
     <Controls>
          <Menu ... />
          <MenuItem ... />
           ...
          <ToolbarItem ... />
           ...
     </Controls>
     <Tabs>
          <Tab ... />
           ...
     </Tabs>
</UserInterface>
```

ノード ダイアログまたはモデル (またはドキュメント) 出力ウィンドウには、拡張オブジェクト **UI** デリゲートが関連付けられていて、ダイアログで呼び出されたカスタム操作を拡張機能が処理できるようにします。UI デリゲートによって、拡張により提供される Java クラスが指定されます。この Java クラスは、

IBM SPSS Modeler ウィンドウと同時に作成され、ExtensionObjectUIDelegate クラスを実装しています。詳しくは、トピック 180 ページの『クライアント側 API クラス』を参照してください。

UI デリゲートの形式は、次のような最初の行を除いて、基本形式と同じです。

```
<UserInterface uiDelegate="Java_class" >
      ...
</UserInterface>
```

ここで、uiDelegate は UI デリゲートの Java クラスの名前です。

拡張機能では、UI を呼び出し可能と想定しないでください。例えば、バッチ モードやアプリケーション サーバーなどのヘッドレス (UI のない) 環境内で拡張機能が実行される場合があります。

操作ハンドラは、カスタム操作を標準 IBM SPSS Modeler ウィンドウに追加する場合に使用されます。これは、拡張オブジェクト **UI** デリゲートに似ていますが、拡張ノードや出力ウィンドウがないとき (例えば、IBM SPSS Modeler のメイン ウィンドウから呼び出すことができる新しいツールの定義時) にも操作ハンドラを使用できます。操作ハンドラは、ユーザーがノード・ダイアログ、モデル出力ウィンドウ、またはドキュメント出力ウィンドウのカスタム メニュー・オプションまたはツールバー・ボタンを選択する場合に呼び出される Java クラスを指定します。ExtensionObjectFrame クラスまたは ActionHandler クラスのいずれかを実装します。いずれかの場合で、標準メニュー、タブ、およびツールバー・ボタンなどの標準ウィンドウ コンポーネントが自動的に使用されます。 詳しくは、トピック 180 ページの『クライアント側 API クラス』を参照してください。

操作ハンドラの形式も、基本形式と同じです。

```
<UserInterface actionHandler="Java_class" >
      ...
</UserInterface>
```

この場合、actionHandler は操作ハンドラの Java クラス名を表します。

IBM SPSS Modeler のメイン ウィンドウから直接呼び出し可能なツールを拡張機能で追加するときは操作ハンドラを使用し、拡張機能によって定義されるノードおよび出力に関連付けられたウィンドウの場合は拡張オブジェクト **UI** デリゲートを使用することをお勧めします。UI デリゲートの方が、基盤となるノートまたは出力オブジェクトへのアクセスが容易であるためです。拡張機能では、同じ UserInterface 要素に uiDelegate と actionHandler の両方を定義することはできません。

フレーム・クラスは、拡張によって標準 IBM SPSS Modeler ウィンドウをカスタマイズするのではなく、独自のウィンドウを提供する場合、モデル出力オブジェクトまたはドキュメント出力オブジェクトに対して使用されます。フレーム・クラスは、ウィンドウ全体およびプロセスを完全に指定する Java クラスです。自動的に使用される標準ウィンドウ コンポーネントはありません。クラスはこれらを個々に指定する必要があります。フレーム・クラスは、ノードに対してではなくモデル出力オブジェクトまたはドキュメント出力オブジェクトに対してのみ使用でき、常に IBM SPSS Modeler ダイアログを使用します。 詳しくは、トピック 165 ページの『カスタム出力ウィンドウ』を参照してください。

フレーム・クラスの形式は次のとおりです。

```
<UserInterface frameClass="Java_class" />
```

この場合 frameClass は、モデル出力オブジェクトまたはドキュメント出力オブジェクトのフレーム・クラス名を表します。アイコン、コントロール、およびタブはフレームクラス自体で指定されるため、それらの要素はこの形式では使用されません。

UserInterface 要素の子要素は、後続のセクションで記述されます。

例

最初の例では、UI デリゲートを定義するモデル・ビルダー・ノードのユーザー・インターフェースを示します。

```
<UserInterface uiDelegate="com.spss.myextension.MyNodeUIDelegate">
     <Icons>
          <Icon type="standardNode" imagePath="images/lg_discriminant.gif" />
          <Icon type="smallNode" imagePath="images/sm_discriminant.gif" />
     </Icons>
     <Tabs defaultTab="1">
      ...
     </Tabs>
</UserInterface>
```

モデル出力オブジェクトの対応するセクションは次のとおりです。

```
<UserInterface>
     <Icons>
          <Icon type="standardWindow" imagePath="images/browser_discriminant.gif" />
     </Icons>
     <Tabs>
          <Tab label="Advanced" labelKey="advancedTab.LABEL"
           helpLink="discriminant_output_advancedtab.htm">
               <ExtensionObjectPanel id="DiscriminantPanel"
                panelClass="com.spss.clef.discriminant.DiscriminantPanel"/>
          </Tab>
     </Tabs>
</UserInterface>
```

モデル・アプライヤー・ノードの User Interface セクションは次のようになります。

```
<UserInterface>
     <Icons>
          <Icon type="standardNode" imagePath="images/lg_gm_discriminant.gif" />
          <Icon type="smallNode" imagePath="images/sm_gm_discriminant.gif" />
     </Icons>
     <Tabs>
          <Tab label="Advanced" labelKey="advancedTab.LABEL"
           helpLink="discriminant_output_advancedtab.htm">
               <ExtensionObjectPanel id="DiscriminantPanel"
                panelClass="com.spss.clef.discriminant.DiscriminantPanel"/>
          </Tab>
     </Tabs>
</UserInterface>
```

アイコン

この項では、オブジェクトに関連するアイコンを定義します。

ノードの場合、この項では次の 2 つの Icon 要素を定義する必要があります。

- 領域に表示するための大きいバージョン
- 領域に表示するための小さいバージョン

モデル出力およびドキュメント出力の場合、ウィンドウ・フレームの左上部に表示される縮小アイコンが定義されます。

14 ページの『ノード・アイコンのデザイン』 では、CLEF にアイコンを作成するためのガイドラインが記載されています。

フォーマット

```
<Icons>
     <Icon type="icon_type" imagePath="image_path" />
      ...
</Icons>
```

ここで、

type (必須) は、以下の表に示すいずれかの種類のアイコンです。

表 33. アイコンの種類 :

データ型	例	説明
standardNode	 図 36. 標準的なノードのアイコン	領域内にノードの形状で表示される大きいサイズ (49 x 49 ピクセル) のアイコン。
smallNode	 図 37. 小さいノードのアイコン	領域内にノードの形状で表示される小さいサイズ (38 x 38 ピクセル) のアイコン。
window	図 38. ウィンドウ アイコン	ブラウザーまたは出力ウィンドウに表示される縮小 (16 x 16 ピクセル) アイコン。

imagePath (必須) は、このアイコンで使用されるイメージの場所を識別します。仕様ファイルがインストールされているディレクトリーに関連した場所が指定されます。

例

```
<Icon type="standardNode" imagePath="images/lg_mynode.gif" />
<Icon type="smallNode" imagePath="images/sm_mynode.gif" />
<Icon type="window" imagePath="images/mynode16.gif" />
```

コントロール

この項は、仕様ファイルの Common Objects セクションで宣言される操作を起動するために使用するカスタム メニューおよびツールバーの項目を定義します。 詳しくは、トピック 41 ページの『アクション』を参照してください。

フォーマット

```
<Controls>
     <Menu ... />
      ...
     <MenuItem ... />
      ...
     <ToolbarItem ... />
      ...
</Controls>
```

後続のセクションで、Menu、MenuItem、および ToolbarItem 要素が記述されます。

例

次の例では、「ノードの生成」メニュー、設定が行われるダイアログのツールバーに項目を追加します。いずれの項目も、フィールド作成ノードを生成する generateDerive という名前の以前定義された操作を実装します。

```
<Controls>
     <MenuItem action="generateDerive" systemMenu="generate"/>
     <ToolbarItem action="generateDerive" showLabel="false"/>
      ...
</Controls>
```

メニュー

カスタム メニューを定義して、標準メニューのいずれかを追加できます。

フォーマット

```
<Menu id="name" label="display_label" labelKey="label_key" systemMenu="menu_name"
showLabel="true_false" showIcon="true_false" separatorBefore="true_false" separatorAfter="true_
false" offset="integer" mnemonic="mnemonic_char" mnemonicKey="mnemonic_key"/>
```

ここで、

id (必須) は、追加されたメニューの一意の識別子です。

label (必須) ユーザー・インターフェースに表示されるメニューの表示名です (showLabel が true に設定されている場合)。

labelKey は、ローカライズ用のラベルを識別します。

systemMenu (必須) は、カスタム メニューが追加される標準メニューを識別します。*menu_name* の値は、次のいずれかです。

- file
- edit
- insert*
- view*
- tools*
- window*
- generate
- help*
- file.project

- file.outputs
- file.models
- edit.stream
- edit.node
- edit.outputs
- edit.models
- edit.project
- tools.repository
- tools.options
- tools.streamProperties

* メイン IBM SPSS Modeler ウィンドウに追加した場合にのみ有効

showLabel は、項目の表示ラベルをユーザー・インターフェースに表示するかどうかを指定します。

showIcon は、項目に関連するアイコンをユーザー・インターフェースに表示するかどうかを指定します。

separatorBefore は、区切り (例えば、メニュー項目の水平線またはツールバー・ボタンのスペースなど) を、新しい項目の前にメニューに追加するかどうかを指定します。

separatorAfter は、新しい項目の後に区切りをメニューに追加するかどうかを指定します。

offset は、新しい項目の位置を定義する、負ではない整数です。例えば、0 は新しい項目を最初の項目として追加します (デフォルトでは最後に追加します)。

mnemonic は、このコントロールを有効化するために Alt キーと組み合わせて使用するアルファベットの文字です (例えば、値 S を指定すると、ユーザーは Alt + S キーを使用してこのコントロールを有効化できます)。

mnemonicKey は、ローカライズ用の mnemonic を識別します。mnemonic も mnemonicKey も使用されない場合、mnemonic はこのコントロールに使用できません。 詳しくは、トピック 117 ページの『アクセスキーとキーボード・ショートカット』を参照してください。

メニュー項目

カスタム メニューの項目を定義して、標準メニューまたはカスタム メニューのいずれかを追加できます。

フォーマット

```
<MenuItem action="identifier" systemMenu="menu_name" customMenu="menu_name"
     showLabel="true_false" showIcon="true_false" separatorBefore="true_false"
     separatorAfter="true_false" offset="integer" />
```

ここで、

action (必須) は、Common Objects セクションで定義された操作の識別子で、このメニュー項目で実行される操作を指定します。

systemMenu は *menu_name* で識別される標準メニューに表示される次のいずれかの項目を指定します。

- file
- edit

- insert*
- view*
- tools*
- window*
- generate
- help*
- file.project
- file.outputs
- file.models
- edit.stream
- edit.node
- edit.outputs
- edit.models
- edit.project
- tools.repository
- tools.options
- tools.streamProperties

* メイン IBM SPSS Modeler ウィンドウに追加した場合にのみ有効

customMenu は Menu 要素の識別子でメニュー項目がカスタム メニューに表示されるように指定します。

showLabel は、項目の表示ラベルをユーザー・インターフェースに表示するかどうかを指定します。

showIcon は、項目に関連するアイコンをユーザー・インターフェースに表示するかどうかを指定します。

separatorBefore は、区切り (例えば、メニュー項目の水平線またはツールバー・ボタンのスペースなど) を、新しい項目の前にメニューに追加するかどうかを指定します。

separatorAfter は、新しい項目の後に区切りをメニューに追加するかどうかを指定します。

offset は、新しい項目の位置を定義する、負ではない整数です。例えば、0 は新しい項目を最初の項目として追加します (デフォルトでは最後に追加します)。

例

```
<MenuItem action="generateSelect" systemMenu="generate" showIcon="true"/>
```

ツールバー項目

ダイアログまたは出力ウィンドウ向けにカスタムのツールバー項目を定義できます。

フォーマット

```
<ToolbarItem action="action" showLabel="true_false" showIcon="true_false"
     separatorBefore="true_false" separatorAfter="true_false" offset="integer" />
```

ここで、

action (必須) は、Common Objects セクションで定義された操作の識別子で、このツールバー項目で実行される操作を指定します。

showLabel は、項目の表示ラベルをユーザー・インターフェースに表示するかどうかを指定します。

showIcon は、項目に関連するアイコンをユーザー・インターフェースに表示するかどうかを指定します。

separatorBefore は、区切り (例えば、メニュー項目の水平線またはツールバー・ボタンのスペースなど) を、新しい項目の前にメニューに追加するかどうかを指定します。

separatorAfter は、新しい項目の後に区切りをメニューに追加するかどうかを指定します。

offset は、新しい項目の位置を定義する、負ではない整数です。例えば、0 は新しい項目を最初の項目として追加します (デフォルトでは最後に追加します)。

例

```
<ToolbarItem action="generateDerive" showLabel="true"/>
```

例 :メイン・ウィンドウへの追加

これは、メイン・ウィンドウの「ツール」メニューの新規項目を追加する仕様ファイルの例です。標準オブジェクト (「モデル出力」ウィンドウまたは「ドキュメント出力」ウィンドウなど) は定義されませんが、「メイン・ウィンドウの変更」を意味する最上位の Extension に UserInterface 要素が含まれます。その他のすべての UserInterface セクションは、標準オブジェクト定義のいずれかに使用される必要があり、標準オブジェクトと関連するダイアログに影響を与えます。

```
<?xml version="1.0" encoding="UTF-8"?>
<Extension version="1.0">
     <ExtensionDetails providerTag="example" id="main_window" label="Main Window" version="1.0"
          provider="IBM Corp." copyright="(c) 2005-2006 IBM Corp."
          description="An example extension that plugs into the main window"/>
     <Resources/>
     <CommonObjects>
          <Actions>
               <Action id="customTool1" label="Custom Tool..." labelKey="customTool.LABEL"
                    imagePath="images/generate.gif" description="Invokes the custom tool"
                    descriptionKey="customTool.TOOLTIP"/>
               <Action id="customTool2" label="Custom Tool..." labelKey="customTool.LABEL"
                    imagePath="images/generate.gif" description="Invokes the custom tool"
                    descriptionKey="customTool.TOOLTIP"/>
          </Actions>
     </CommonObjects>
     <UserInterface actionHandler="com.spss.cleftest.MainWindowActionHandler">
          <Controls>
               <Menu systemMenu="tools" id ="toolsExtension" separatorBefore="true"
                    label="Extension Items" offset="3"/>
               <MenuItem action="customTool2" customMenu="toolsExtension" showIcon="true"/>
               <MenuItem action="customTool1" systemMenu="file.models" showIcon="true"/>
               <ToolbarItem action="customTool1" offset="0"/>
          </Controls>
     </UserInterface>
</Extension>
```

これにより、「ツール」メニューに新しいサブメニューの「拡張項目」が追加されます。この新しいサブメニューには、「カスタム・ツール」という単一のエントリーが表示されます。

XML コードを *extension.xml* というファイル名で保存し、拡張を CLEF に追加してこの例を使用することができます。 詳しくは、トピック 205 ページの『CLEF 拡張のテスト』を参照してください。

タブ

Tabs セクションでは、以下に表示されるタブを定義します。

- ユーザーが領域上のノードを開いた場合に表示されるダイアログ
- モデル出力ウィンドウ
- ドキュメント出力ウィンドウ

それぞれの Tabs セクションに複数の Tab 要素を定義し、表示するカスタム・タブをそれぞれの要素で宣言することができます。

```
<Tabs defaultTab="integer">
     <Tab ... />
     <Tab ... />
      ...
</Tabs>
```

この場合、defaultTab はノードのダイアログまたはウィンドウがストリーム内で初めて開いた場合に表示されるタブを指定する、負ではない整数です。ユーザーが異なるタブを選択する場合、ストリームが有効な間にダイアログまたはウィンドウを閉じて再び開くと、デフォルトのタブではなく、最近表示されたタブが表示されます。タブの番号は 0 から始まります。

その他のタブがダイアログまたはウィンドウに自動的に含まれる場合があります。例えば、すべてのダイアログおよび出力ウィンドウには「注釈」タブが含まれ、すべてのデータ・ソースのノードのダイアログには「フィルター」タブおよび「タイプ」タブが含まれます。

Tab 要素はタブ ラベル (タブ自体に表示されるテキスト) を宣言し、タブ ラベルが翻訳された場合に参照として使用されるラベル・キーも使用する必要があります。

各 Tab 要素内ではパネル設定が行われ、タブのメイン・コンテンツ領域がどのように配置されるかを定義します。パネル設定はテキスト・ブラウザー、拡張オブジェクト、またはプロパティーのいずれかになります。 詳しくは、トピック 119 ページの『パネル設定』を参照してください。

各 Tab 要素の形式は次のとおりです。

```
<Tab id="identifier" label="display_label" labelKey="label_key" helpLink="help_ID"
          mnemonic="mnemonic_char" mnemonicKey="mnemonic_key" >
     <TextBrowserPanel ... />
     <ExtensionObjectPanel ... />
     <PropertiesPanel ... />
     <ModelViewerPanel ... />
</Tab>
```

ここで、

id は、Java コードでタブの参照に使用できる一意の識別子です。

label (必須) は、ユーザー・インターフェース上に表示する場合のタブの表示名です。

labelKey は、ローカライズ用のラベルを識別します。

helpLink は、ヘルプ・システムを起動する場合に表示されるヘルプ トピックの識別子です。識別子の形式は、ヘルプ・システムの種類によって異なります (168 ページの『ヘルプ・システムの場所および種類の定義』 参照)。

HTML Help :ヘルプ トピックの URL

JavaHelp :トピック ID

mnemonic は、このコントロールを有効化するために Alt キーと組み合わせて使用するアルファベットの文字です (例えば、値 S を指定すると、ユーザーは Alt + S キーを使用してこのコントロールを有効化できます)。

mnemonicKey は、ローカライズ用の mnemonic を識別します。mnemonic も mnemonicKey も使用されない場合、mnemonic はこのコントロールに使用できません。 詳しくは、トピック『アクセス キーとキーボード・ショートカット』を参照してください。

例

Tab 要素の例については、要素内で定義できる各種パネルの指定に関するセクション (119 ページの『テキスト・ブラウザー・パネル』、 121 ページの『拡張オブジェクト・パネル』、 122 ページの『プロパティー・パネル』、 124 ページの『モデル・ビューアー・パネル』) を参照してください。

アクセス キーとキーボード・ショートカット

ユーザー・インターフェースの機能に対するマウス・アクセスの代わりとして、さまざまなキーの組み合わせを指定して、ユーザーはキーボードを使用して機能にアクセスできます。

アクセス・キー

アクセス キーは、Alt キーと組み合わせて使用できるキーです。メニュー項目、ダイアログ・タブ、さまざまなダイアログ・コントロールについて、次の要素の mnemonic 属性を使用してアクセス キーを指定できます。

表 *34.* ユーザー・インターフェースでの機能 :

機能	エレメント	参照...
画面操作 (例: メニュー項目)	action	41 ページの『アクション』
メニュー	menu	112 ページの『メニュー』
ダイアログ・タブ	タブ	116 ページの『タブ』
コントローラ	各種	131 ページの『コントローラ』

例えば、以下のメニューについて考えてみます。

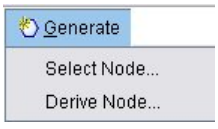

図 *39.* メニュー項目

このメニューのアクセス・キーを指定するには、以下のコードを使用します。

```
<Actions>
     <Action id="generateSelect" label="Select Node..." labelKey="generate.selectNode.LABEL"
          imagePath="images/generate.gif" description="Generates a select node"
          descriptionKey="generate.selectNode.TOOLTIP" mnemonic="S" />
     <Action id="generateDerive" label="Derive Node..." labelKey="generate.deriveNode.LABEL"
          imagePath="images/generate.gif" description="Generates a derive node"
          descriptionKey="generate.deriveNode.TOOLTIP"  mnemonic="D" />
   ...
</Actions>
```

これにより、メニュー項目に下線文字が表示されます。

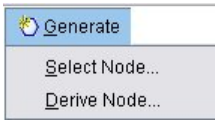

図 *40.* メニュー項目にアクセス キーを使用

ユーザーは、マウスでメニュー項目をクリックするほか、Alt + S キーや Alt + D キーを使用して、メニュー項目にアクセスできます。

ショートカット・キー

ショートカット・キーは、エンド・ユーザーがユーザー・インターフェースをナビゲートできるキーの組み合わせです。IBM SPSS Modeler は、標準としてさまざまなキーボード・ショートカットを提供しています。CLEF では、追加したカスタム メニュー項目にのみショートカットを追加できます。

例えば、アクセス キーの例で示されたメニューの項目にショートカットを指定するには、次のコードを使用します。

```
<Actions>
     <Action id="generateSelect" label="Select Node..." labelKey="generate.selectNode.LABEL"
          imagePath="images/generate.gif" description="Generates a select node"
          descriptionKey="generate.selectNode.TOOLTIP" mnemonic="S" shortcut="CTRL+SHIFT+S" />
     <Action id="generateDerive" label="Derive Node..." labelKey="generate.deriveNode.LABEL"
          imagePath="images/generate.gif" description="Generates a derive node"
          descriptionKey="generate.deriveNode.TOOLTIP"  mnemonic="D"  shortcut="CTRL+SHIFT+D" />
     ...
</Actions>
```

ショートカット・キーの組み合わせが、メニュー項目に追加されます。

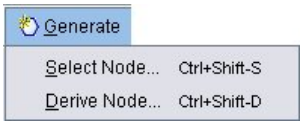

図 *41.* メニュー項目にショートカット・キーを使用

ユーザーは、マウスでメニュー項目をクリックするほか、キーボード・ショートカットを使用して、メニュー項目にアクセスできます。この例のように、アクセス キーと組み合わせてショートカット・キーを指定できます。

同じダイアログまたは標準の IBM SPSS Modeler ショートカットにすでに指定されたショートカットを使用しないように注意してください。

パネル設定

それぞれの Tab 要素には、以下の表に示すいずれかの種類の単一パネルを指定することができます。

表 35. パネル設定の種類

パネル	表示	参照...
テキスト・ブラウザー・パネル	指定されたコンテナーのテキスト・コンテンツ。	『テキスト・ブラウザー・パネル』
拡張オブジェクト・パネル	指定された Java クラスに定義されたコンテンツ。	121 ページの『拡張オブジェクト・パネル』
プロパティー・パネル	プロパティー・コントロール (ボタン、チェック・ボックス、入力フィールド)。	122 ページの『プロパティー・パネル』
モデル・ビューアー・パネル	指定されたコンテナーからの PMML 形式のモデル出力。	124 ページの『モデル・ビューアー・パネル』

テキスト・ブラウザー・パネル

テキスト・ブラウザー・パネルでは、拡張で指定されたコンテナーのテキスト・コンテンツが表示されます。サポートされた形式 (UTF-8 文字コード) は、通常のテキスト形式、HTML 形式、およびリッチ・テキスト形式 (RTF) です。

HTML 形式のテキスト・ブラウザー・パネルを持つモデル出力ウィンドウの例を以下に示します。

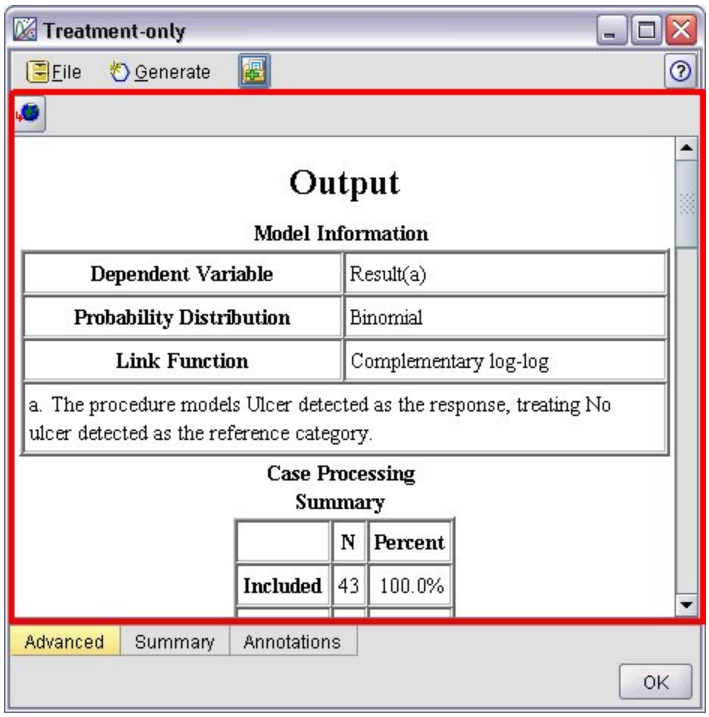

図 42. テキスト・ブラウザー・パネルが強調表示されたモデル出力ウィンドウ

フォーマット

```
<TextBrowserPanel container="name" textFormat="text_format" rows="integer"
     columns="integer" wrapLines="true_false" >
     -- advanced custom layout options --
</TextBrowserPanel>
```

ここで、

container (必須) は、パネルの内容を取得するコンテナーの名前です。

textFormat (必須) は、パネルに表示されるテキストの形式を指定します。次の 3 つの種類があります。

- plainText
- html
- rtf

rows および columns は、パネルを含むウィンドウが開くと表示されるテキスト行および列の数を指定します。

wrapLines は、長いテキスト行に折り返しを使用する (true) または長いテキスト行を読む場合に水平スクロールを使用する (false) かどうかを指定します。デフォルトは false です。

詳細なユーザー設定のレイアウト・オプションでは、画面コンポーネントの位置および表示について、正確なコントロールを提供します。 詳しくは、トピック 156 ページの『詳細なカスタム・レイアウト』を参照してください。

例

上記のテキスト・ブラウザー・パネルを定義する Tab セクションの例を以下に示します。

```
<Tab label="Advanced" labelKey="advancedTab.LABEL" helpLink="genlin_output_advancedtab.htm">
     <TextBrowserPanel container="advanced_output" textFormat="html" />
</Tab>
```

モデル出力が、タブ設定として同じ ModelOutput セクションに定義されたコンテナーに送られます。

```
<ModelOutput id="generalizedlinear" label="Generalized Linear">
     <Containers>
      ...
          <Container name="advanced_output"/>
     </Containers>
     <UserInterface>
      ...
          <Tabs>
               <Tab label="Advanced" ... >
                    <TextBrowserPanel container="advanced_output" ... />
               </Tab>
          </Tabs>
     </UserInterface>
</ModelOutput>
```

コンテナーは、関連する構築ノードの Execution セクションの CreateDocument 要素で参照されます。

```
<Execution>
 ...
     <Constructors>
          <CreateModelOutput type="generalizedlinear">
               <CreateModel type="generalizedlinear_model" target="model" sourceFile="pmml"/>
               <CreateDocument type="html_output" target="advanced_output" sourceFile=
```

```
                "htmloutput"/>
          </CreateModelOutput>
     </Constructors>
</Execution>
```

拡張オブジェクト・パネル

拡張オブジェクト・パネルはテキスト・ブラウザー・パネルと同様の方法で動作しますが、コンテナーのテキスト・コンテンツは表示せず、CLEF Java API で定義された ExtensionObjectPanel インターフェースを実装する指定された Java クラスのインスタンスを作成します。

拡張オブジェクト・パネルを持つモデル適用ノードのダイアログの例を以下に示します。

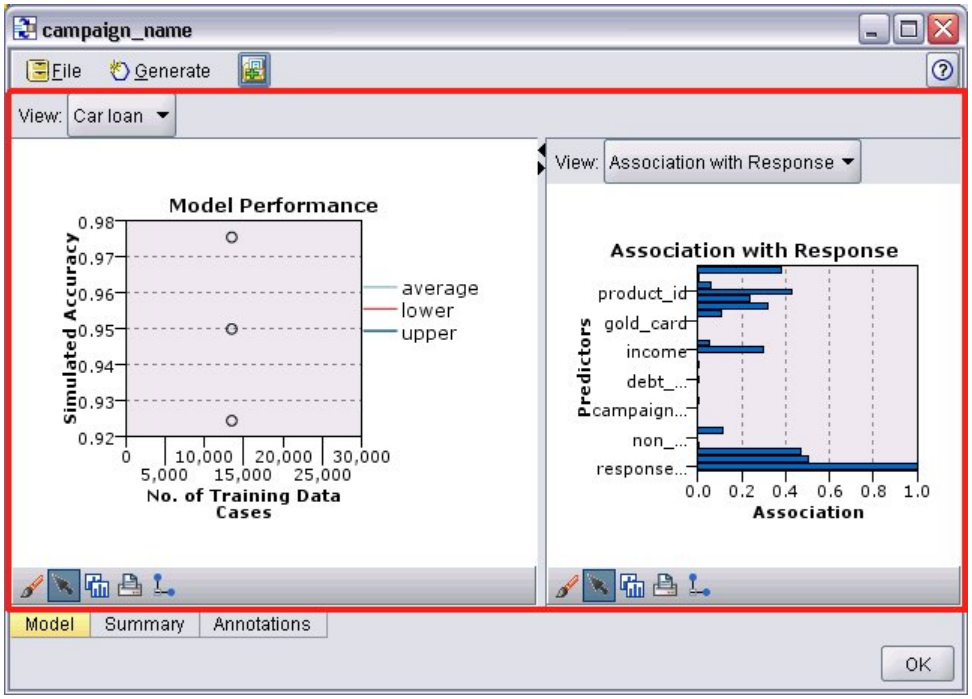

図 43. 拡張オブジェクト・パネルが強調表示されたモデル出力ウィンドウ

フォーマット

```
<ExtensionObjectPanel id="identifier" panelClass="Java_class" >
     -- advanced custom layout options --
</ExtensionObjectPanel>
```

ここで、

id は、Java コードのパネルを参照するために使用できる一意の識別子です。

panelClass (必須) は、パネルが実装する Java クラス名です。

詳細なユーザー設定のレイアウト・オプションでは、画面コンポーネントの位置および表示について、正確なコントロールを提供します。 詳しくは、トピック 156 ページの『詳細なカスタム・レイアウト』を参照してください。

例

上記の拡張オブジェクト・パネルを定義する Tab セクションの例を以下に示します。

```
<Tab label="Model" labelKey="Model.LABEL" helpLink="selflearnnode_output.htm">
     <ExtensionObjectPanel id="SelfLearningPanel"
          panelClass="com.spss.clef.selflearning.SelfLearningPanel"/>
</Tab>
```

プロパティー・パネル

プロパティー・パネルを使用すると、タブまたはプロパティー・サブパネル (129 ページの『プロパティー・サブパネル』を参照) にプロパティー・コントロールを表示することができます。プロパティー・コントロールは、画面に表示するオブジェクトのプロパティーを変更するために使用できる画面コンポーネント (ボタン、チェック・ボックス、入力フィールドなど) です。プロパティー・パネルでは、ユーザーが「**OK**」 または 「適用」 をクリックすると、プロパティー・コントロールで行われた変更が自動的に適用されます。ユーザーが 「キャンセル」 または 「リセット」 をクリックすると、最後に「適用」の操作が行われた以降に行われた変更が破棄されます。

プロパティー・パネルを持つノードのダイアログの例を以下に示します。

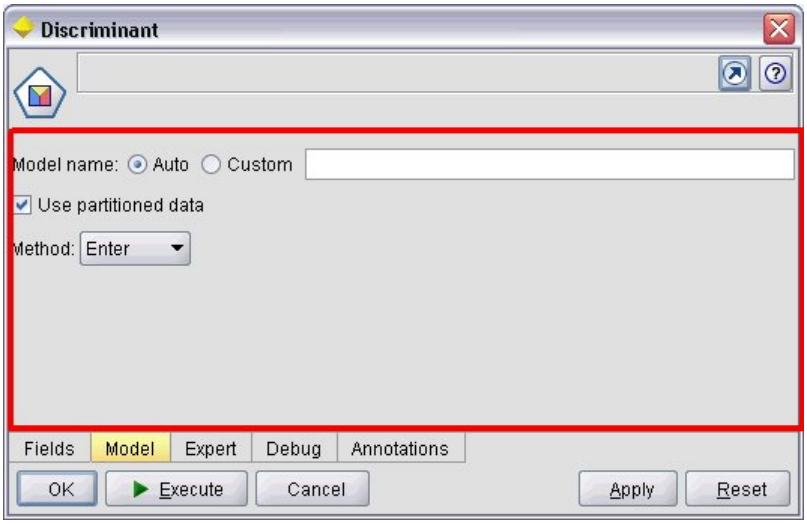

図 *44.* プロパティー・パネルが強調表示されたノードのダイアログ

3 つのプロパティー・パネルを持つプロパティーのサブパネルの例を以下に示します。

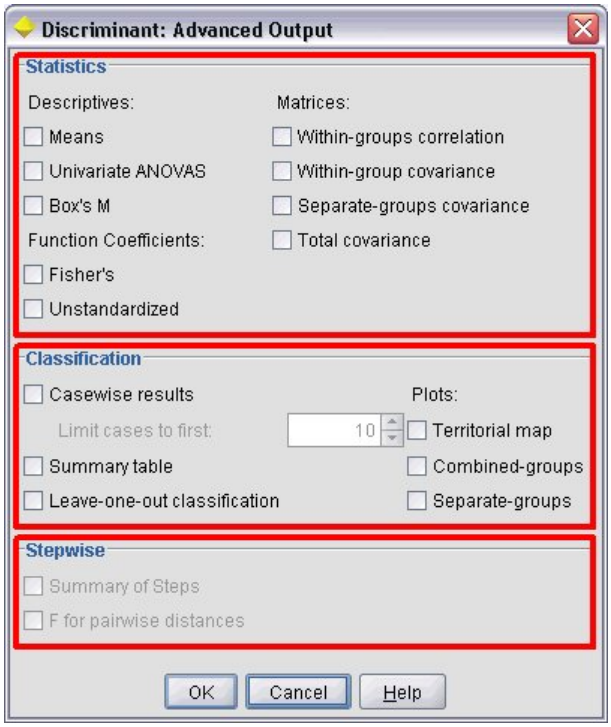

図 *45.* プロパティー・パネルが強調表示されたプロパティー・サブパネル

フォーマット

```
<PropertiesPanel id="identifier" label="display_label" labelKey="label_key">
     -- advanced custom layout options --
     -- property control specifications --
</PropertiesPanel>
```

ここで、

`id` は、Java コードのパネルを参照するために使用できる一意の識別子です。

`label` プロパティー・コントロールのグループに表示する見出しです (最後の例の 「統計」、「分類」 および 「ステップワイズ」 など)。

`labelKey` は、ローカライズ用のラベルを識別します。

詳細なユーザー設定のレイアウト・オプションでは、画面コンポーネントの位置および表示について、正確なコントロールを提供します。 詳しくは、トピック 156 ページの『詳細なカスタム・レイアウト』を参照してください。

個々のプロパティー・コントロール設定については、 125 ページの『プロパティー・コントロール設定』に記載されています。

例

```
<Tab label="Model" labelKey="Model.LABEL" helpLink="discriminant_node_model.htm">
     <PropertiesPanel>
          <SystemControls controlsId="ModelGeneration" />
          <ComboBoxControl property="method">
```

```
            <Layout fill="none" />
        </ComboBoxControl>
    </PropertiesPanel>
</Tab>
```

モデル・ビューアー・パネル

モデル・ビューアー・パネルには、拡張で指定されたコンテナーの PMML 形式モデル出力が表示されます。

以下に、モデル・ビューアー・モデルを含むモデル・アプライヤー・ノード ウィンドウの例を示します。

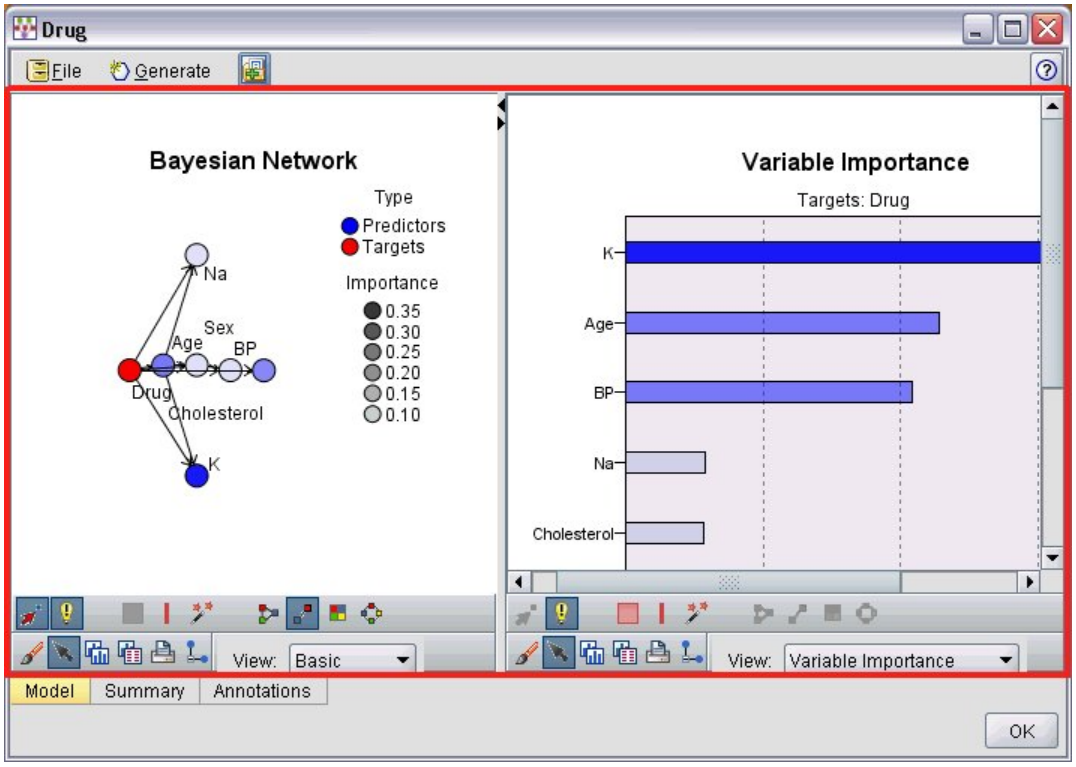

図 *46.* モデル・ビューアー・パネルが強調表示された出力ウィンドウ

フォーマット

```
<ModelViewerPanel container="container_name">
    -- advanced custom layout options --
</ModelViewerPanel>
```

container (必須) は、モデル出力が割り当てられるコンテナーの名前です。

詳細なユーザー設定のレイアウト・オプションでは、画面コンポーネントの位置および表示について、正確なコントロールを提供します。 詳しくは、トピック 156 ページの『詳細なカスタム・レイアウト』を参照してください。

例

この例では、モデル・アプライヤー 設定のモデル・ビューアー・パネルの使用について示しています。モデル出力は、model というコンテナーに割り当てられています。以下の例では、モデル・アプライヤー設定は model コンテナーを配置し、モデル・ビューアー・モデルと関連させます。

```
<Node id="applyBN" type="modelApplier">
     <ModelProvider container="model" isPMML="true" />
      ...
     <Containers>
          <Container name="model" />
     </Containers>
     <UserInterface>
      ...
     <Tabs>
          <Tab label="Model" labelKey="modelTab.LABEL" helpLink="BN_output_modeltab.htm">
               <ModelViewerPanel container="model"/>
          </Tab>
      ...
     </Tabs>
     </UserInterface>
      ...
</Node>
```

プロパティー・コントロール設定

プロパティー・コントロールは、画面上に表示されるオブジェクトのプロパティーを変更するために使用されるボタン、チェック・ボックス、入力フィールドなどの画面コンポーネントです。プロパティー・コントロール設定の形式は、次のようなプロパティー・コントロールの種類によって異なります。

- UI コンポーネント
- プロパティー・パネル
- コントローラー

UI コンポーネント コントロールは、操作ボタン、画面上の静的テキスト、システム コントロール (すべてのダイアログに共通するプロパティーを操作する一連のコントロール) です。

プロパティー・パネル コントロールは、プロパティー・パネル設定内の個々のパネルです。

コントローラは、プロパティー・コントロールの最大グループを形成し、チェック・ボックス、選択情報、スピン・コントロールなどの項目が含まれます。

UI コンポーネント・コントロール

UI コンポーネント・コントロールを以下の表に示します。

表 *36. UI* コンポーネント・コントロール

Control	説明
ActionButton	クリックすると事前設定された操作を実行する画面上のボタン。
StaticText	画面上に表示される非可変のテキスト文字列。
SystemControls	すべてのモデルに共通するプロパティーを操作するコントロールの標準セット。

操作ボタン

Common Objects セクションで指定された操作を実行するダイアログまたはツールバー・ボタンを定義します。ユーザーがこのボタンをクリックすると、新しい画面を表示するなどの操作が実行されます。

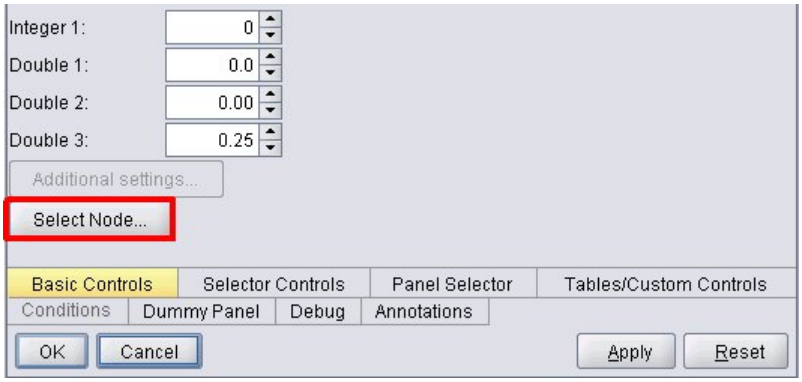

図 *47.* 操作ボタンが強調表示されたダイアログ

フォーマット

```
<ActionButton action="action" showLabel="true_false" showIcon="true_false" >
     -- advanced custom layout options --
</ActionButton>
```

ここで、

action (必須) は、実行される操作の識別子です。

showLabel は、ボタン・ラベルを表示する (true) かまたは非表示にする (false) かを指定します (例えば、ツールバー・ボタンの場合、ラベルではなくアイコンの表示を選択します)。デフォルトは true です。

showIcon はボタンに関連するアイコンを表示する (true) か、または非表示にする (false) かを選択します。デフォルトは false です。

詳細なユーザー設定のレイアウト・オプションでは、画面コンポーネントの位置および表示について、正確なコントロールを提供します。 詳しくは、トピック 156 ページの『詳細なカスタム・レイアウト』を参照してください。

例

上記の操作ボタンを作成するためのコードは、次のとおりです。

```
<ActionButton action="generateSelect"/>
```

Common Objects セクションで定義される操作は次のとおりです (ボタン・ラベルもここで定義されます)。

```
<CommonObjects extensionListenerClass="com.spss.cleftest.TestExtensionListener">
     ...
    <Actions>
      <Action id="generateSelect" label="Select Node..." labelKey="generate.selectNode.LABEL"
           imagePath="images/generate.gif" description="Generates a select node"
           descriptionKey="generate.selectNode.TOOLTIP"/>
        ...
    </Actions>
</CommonObjects>
```

静的テキスト

この要素を使用して、ダイアログまたは出力ウィンドウに非可変のテキスト文字列を使用することができます。静的テキストを持つプロパティー・パネルを以下に示します。

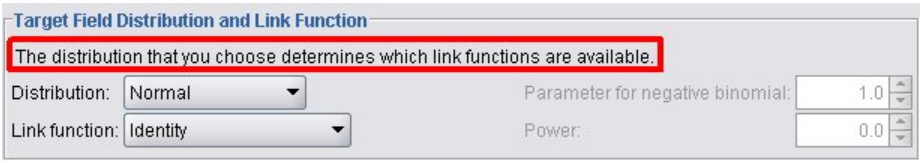

図 *48.* 静的テキストを強調表示したプロパティー・パネル

フォーマット

```
<StaticText text="static_text" textKey="text_key" >
     -- advanced custom layout options --
</StaticText>
```

ここで、

`text` は、使用するテキスト文字列です。

`textKey` は、ローカライズ用の静的テキストを識別します。

詳細なユーザー設定のレイアウト・オプションでは、画面コンポーネントの位置および表示について、正確なコントロールを提供します。 詳しくは、トピック 156 ページの『詳細なカスタム・レイアウト』を参照してください。

例

上記の静的テキストで使用される宣言の例を以下に示します。

```
<StaticText text="The distribution that you choose determines which link functions are
     available." textKey="Genlin_staticText1"/>
```

システム コントロール

一部のプロパティーは、すべてのモデルに共通です。モデル・ビルダー・ノードでは、システム コントロールは共通するプロパティーを操作するコントロールの標準セットです。

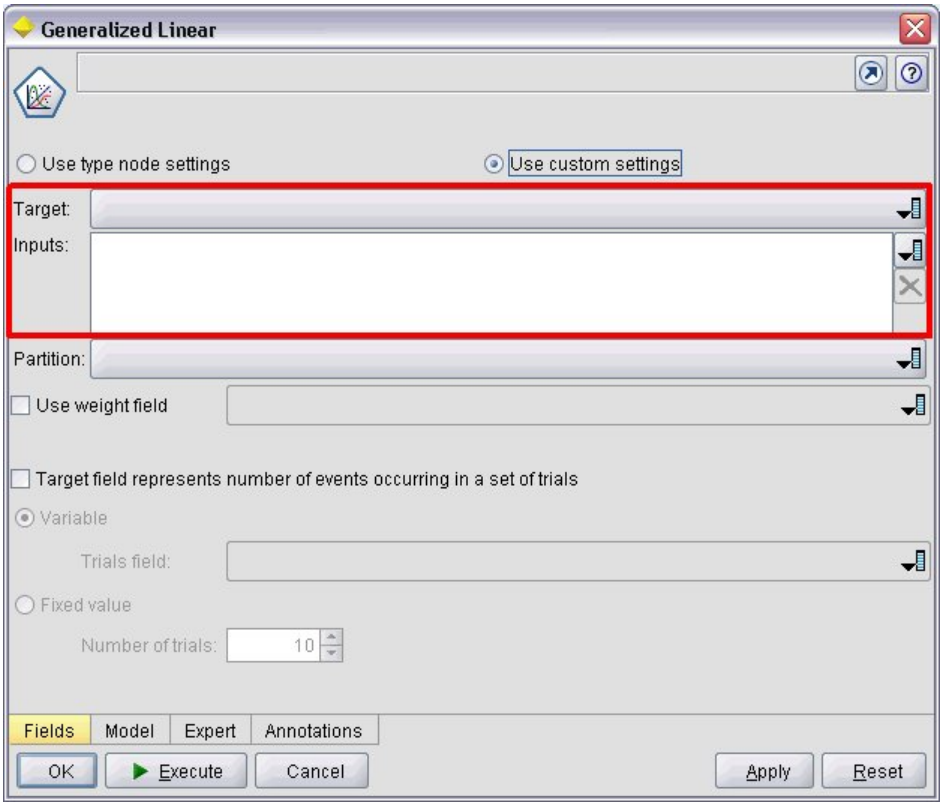

図 *49.* システム コントロールが強調表示されたダイアログの例

フォーマット

```
<SystemControls controlsID="identifier" >
     -- advanced custom layout options --
</SystemControls>
```

controlsID は、コントロールのセットの識別子です。この識別子は、モデル・ビルダーの宣言の ModelingFields 要素の controlsID 属性に指定されている識別子と同じものである必要があります (51 ページの『モデル・ビルダー』を参照)。

詳細なユーザー設定のレイアウト・オプションでは、画面コンポーネントの位置および表示について、正確なコントロールを提供します。 詳しくは、トピック 156 ページの『詳細なカスタム・レイアウト』を参照してください。

例

この例では、上記のシステム・コントロールで使用した宣言を示します。

ノード設定の Model Builder セクションでは、次で一連のシステム コントロールのセットが定義され、この場合モデルの入力フィールドおよび出力フィールドのフィールド・ピッカーを保持します。

```
<ModelBuilder ... >
     <ModelingFields controlsId="modelingFields">
          <InputFields property="inputs" multiple="true" label="Inputs" types="[range set
orderedSet flag]" labelKey="inputFields.LABEL"/>
          <OutputFields property="target" multiple="false" types="[range flag]"
```

```
label="Target" labelKey="targetField.LABEL"/>
     </ModelingFields>
      ...
</ModelBuilder>
```

その後、このセットのコントロールはそれらが表示されるモデル・ビルダー・ノードのダイアログのタブ定義で参照されます。

```
<Tab label="Fields" labelKey="Fields.LABEL" helpLink="genlin_node_fieldstab.htm">
     <PropertiesPanel>
          <SystemControls controlsId="modelingFields">
          </SystemControls>
      ...
     </PropertiesPanel>
</Tab>
```

プロパティー・パネル・コントロール

プロパティー・パネル・コントロールを以下の表に示します。

表 37. プロパティー・パネル・コントロール

Control	説明
PropertiesSubPanel	ユーザーがプロパティー・パネルのボタンをクリックすると表示される各ダイアログ。
PropertiesPanel	プロパティー・サブパネルの宣言にネストされた、または最上位のプロパティー・パネルの宣言にネストされたプロパティー。

プロパティー・サブパネル

ユーザーがプロパティー・パネルのボタンをクリックすると表示される各ダイアログを定義します。プロパティー・サブパネルの宣言は、タブのメインプロパティー・パネル設定の一部として作成されます。

フォーマット

```
<PropertiesSubPanel buttonLabel="display_label" buttonLabelKey="label_key"
     dialogTitle="display_title" dialogTitleKey="title_key" helplink="help_ID"
     mnemonic="mnemonic_char" mnemonicKey="mnemonic_key" >
     -- advanced custom layout options --
     -- property control specifications --
</PropertiesSubPanel>
```

ここで、

buttonLabel は、サブパネルへのアクセスを提供するボタンのラベルです。

buttonLabelKey ローカライズ用のボタンのラベルを識別します。

dialogTitle は、サブパネル・ダイアログのタイトル・バーに表示されるテキストです。

dialogTitleKey は、ローカライズ用のサブパネル・ダイアログのタイトルを識別します。

helpLink は、ヘルプ・システムを起動する場合に表示されるヘルプ トピックの識別子です。識別子の形式は、ヘルプ・システムの種類によって異なります (168 ページの『ヘルプ・システムの場所および種類の定義』 参照)。

HTML Help :ヘルプ トピックの URL

JavaHelp :トピック ID

mnemonic は、このコントロールを有効化するために Alt キーと組み合わせて使用するアルファベットの文字です (例えば、値 S を指定すると、ユーザーは Alt + S キーを使用してこのコントロールを有効化できます)。

mnemonicKey は、ローカライズ用の mnemonic を識別します。mnemonic も mnemonicKey も使用されない場合、mnemonic はこのコントロールに使用できません。 詳しくは、トピック 117 ページの『アクセス・キーとキーボード・ショートカット』を参照してください。

詳細なユーザー設定のレイアウト・オプションでは、画面コンポーネントの位置および表示について、正確なコントロールを提供します。 詳しくは、トピック 156 ページの『詳細なカスタム・レイアウト』を参照してください。

個々のプロパティー・コントロール設定については、 125 ページの『プロパティー・コントロール設定』に記載されています。

例

以下に、ユーザーがタブのメイン・プロパティー・パネルの「出力」ボタンをクリックした場合に表示されるプロパティー・サブパネルを示します。

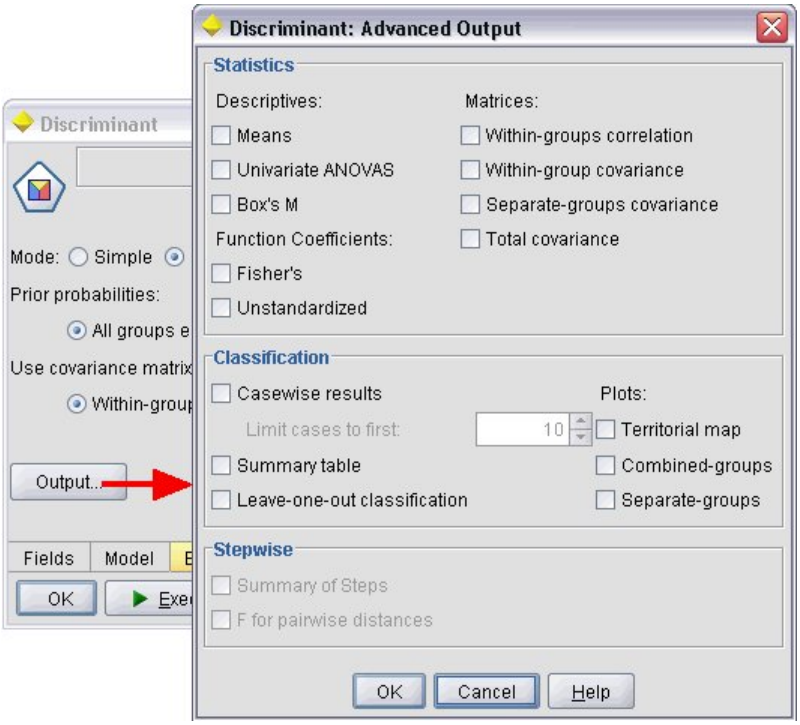

図 *50.* プロパティー・サブパネル

以下のコードは、上記のプロパティー・サブパネルを取得するために使用する宣言の大部分を示します。サブパネルの宣言内で、各フィールド グループ (統計、分類およびステップワイズ) は、独自のプロパティー・パネル設定を保持します。

```
<PropertiesSubPanel buttonLabel="Output..." buttonLabelKey="OutputSubPanel.LABEL"
     dialogTitle="Discriminant: Advanced Output" dialogTitleKey="AdvancedOutputSubDialog.LABEL"
     helpLink="discriminant_node_outputdlg.htm">
```

```
     ...
     <PropertiesPanel>
          <PropertiesPanel label="Statistics" ... >
           ...
          </PropertiesPanel>
          <PropertiesPanel label="Classification" ... >
           ...
          </PropertiesPanel>
          <PropertiesPanel label="Stepwise" ... >
           ...
          </PropertiesPanel>
     </PropertiesPanel>
</PropertiesSubPanel>
```

プロパティー・パネル (ネスト)

プロパティー・サブパネルの宣言内にプロパティー・パネル設定をネストし、サブパネルから表示されるダイアログの内容を定義することができます。 詳しくは、トピック 129 ページの『プロパティー・サブパネル』を参照してください。

また、最上位のプロパティー・パネルの宣言内にプロパティー・パネル設定をネストすることもできます。複数のプロパティー・パネルで構成されたタブ全体の内容を、タブの特定の特定のボタンが選択されているかどうかによって有効化または無効化する場合の例を示します。この場合、タブ設定は次のようになります。

```
<Tab .... >
     <PropertiesPanel>
      --- button specification ---
          <PropertiesPanel>
               <Enabled>
                --- condition involving button value ---
               </Enabled>
           ...
          </PropertiesPanel>
          <PropertiesPanel>
               <Enabled>
                --- condition involving button value ---
               </Enabled>
           ...
          </PropertiesPanel>
           ...
     </PropertiesPanel>
</Tab>
```

ネストされたプロパティー・パネル設定の形式は、最上位の要素の形式の同じです。 詳しくは、トピック 122 ページの『プロパティー・パネル』を参照してください。

コントローラ

コントローラーは、プロパティー・コントロールの最大のグループを形成しています。

表 38. コントローラ

Control	説明
CheckBoxControl	チェック・ボックス。
CheckBoxGroupControl	チェック・ボックスのセット。enum 値ごとに 1 つのチェック・ボックス。

表 38. コントローラ (続き)

Control	説明
ClientDirectoryChooserControl	ユーザーがクライアントのディレクトリーを選択できる単一行のテキスト・フィールドおよび関連ボタン。
ClientFileChooserControl	ユーザーがクライアントのファイルを選択できる単一行のテキスト・フィールドおよび関連ボタン。
ComboBoxControl	enum 値を含む選択情報のドロップダウン・リスト。
DBConnectionChooserControl	ユーザーはデータ・ソースおよびデータベースへの接続を選択可能。
DBTableChooserControl	データベース接続後、ユーザーはデータベース・テーブルを選択可能。
MultiFieldChooserControl	(ノードのみ) ユーザーがリストからフィールドを選択できる、フィールド名のリスト。
MultiItemChooserControl	値のリストから、1 つまたは複数の項目を選択できます。
PasswordBoxControl	入力文字が非表示になる単一行のテキスト・フィールド。
PropertyControl	ユーザー定義可能なプロパティーのコントロール。
RadioButtonGroupControl	1 回に 1 つのボタンのみ選択できるラジオ・ボタンのセット。enum プロパティーの場合、 enum 値ごとに 1 つのラジオ・ボタンが表示されます。ブール・プロパティーの場合、2 つのラジオ・ボタンが表示されます。
ServerDirectoryChooserControl	ユーザーがサーバーのディレクトリーを選択できる単一行のテキスト・フィールドおよび関連ボタン。
ServerFileChooserControl	ユーザーがサーバーのファイルを選択できる単一行のテキスト・フィールドおよび関連ボタン。
SingleFieldChooserControl	(ノードのみ) ユーザーがリストから 1 つのフィールドを選択できる、フィールド名のリスト。
SingleItemChooserControl	値のリストから、1 つの項目を選択できます。
SpinnerControl	スピン・コントロール (値を変更するための、上下の矢印の付いた数値型フィールド)。
TableControl	ダイアログまたはウィンドウにテーブルを追加。
TextAreaControl	複数行のテキストフィールド。
TextBoxControl	単一行のテキストフィールド。

コントローラの属性

コントローラの設定には、次の属性が含まれます。

property="*value*" showLabel="*true_false*" label="*display_label*" labelKey="*label_key*" labelWidth="*label_width*" labelAbove="*true_false*" description="*description*" descriptionKey="*description_key*" mnemonic="*mnemonic_char*" mnemonicKey="*mnemonic_key*"

ここで、

property (必須) は、プロパティー・コントロールの一意の識別子です。

showLabel はボタンに関連するプロパティー・コントロールの表示ラベルを表示する (true) か、または非表示にする (false) かを指定します。デフォルトは true です。

label は、ユーザー・インターフェースに表示される、プロパティー・コントロールの表示名です。また、この値はプロパティー・コントロールのアクセス可能な簡単な説明としても動作します。 詳しくは、トピック 177 ページの『アクセシビリティー』を参照してください。

labelKey は、ローカライズ用のラベルを識別します。

labelWidth は、ラベルが広がる表示グリッド列の数値です。デフォルトは 1 です。

labelAbove は、コントロールのラベルを上部に表示する (true) またはコントロールに隣接して表示する (false) かを指定します。デフォルトは false です。

description は、カーソルをコントロールに移動した場合に表示される、ツールヒントのテキストです。また、この値はプロパティー・コントロールのアクセス可能な詳細な説明としても動作します。 詳しくは、トピック 177 ページの『アクセシビリティー』を参照してください。

descriptionKey は、ローカライズ用の説明を識別します。

mnemonic は、このコントロールを有効化するために Alt キーと組み合わせて使用するアルファベットの文字です (例えば、値 S を指定すると、ユーザーは Alt + S キーを使用してこのコントロールを有効化できます)。

mnemonicKey は、ローカライズ用の mnemonic を識別します。mnemonic も mnemonicKey も使用されない場合、mnemonic はこのコントロールに使用できません。 詳しくは、トピック 117 ページの『アクセスキーとキーボード・ショートカット』を参照してください。

チェック・ボックス・コントロール

チェック・ボックスを定義します。

図 *51.* チェック・ボックス

フォーマット

```
<CheckBoxControl controller_attributes invert="true_false" >
     -- advanced custom layout options --
</CheckBoxControl>
```

ここで、

controller_attributes は、「 132 ページの『コントローラの属性』」で説明されているとおりです。

invert はあまり使用されることはありませんが、true に設定された場合、チェック・ボックスの選択および選択解除の影響を逆にします。デフォルトは false です。

詳細なユーザー設定のレイアウト・オプションでは、画面コンポーネントの位置および表示について、正確なコントロールを提供します。 詳しくは、トピック 156 ページの『詳細なカスタム・レイアウト』を参照してください。

例

上記のチェック・ボックスを配置するためのコード例を以下に示します (チェック・ボックスのラベルは、特性ファイル内の別の場所で定義されます)。Layout 要素については、 156 ページの『詳細なカスタム・レイアウト』を参照してください。

```
<CheckBoxControl property="means">
     <Layout rowIncrement="0" gridWidth="1" />
</CheckBoxControl>
<CheckBoxControl property="within_groups_correlation">
     <Layout gridColumn="2" />
</CheckBoxControl>
<CheckBoxControl property="univariate_anovas">
     <Layout gridWidth="1" rowIncrement="0" />
</CheckBoxControl>
<CheckBoxControl property="within_group_covariance">
     <Layout gridColumn="2" />
</CheckBoxControl>
<CheckBoxControl property="box_m">
     <Layout gridWidth="1" rowIncrement="0" />
</CheckBoxControl>
<CheckBoxControl property="separate_groups_covariance">
     <Layout gridColumn="2" />
</CheckBoxControl>
```

チェック・ボックス・グループ・コントロール

グループ化され、単一ユニットとして扱われるチェック・ボックスのセットを定義します。グループのメンバーを定義する列挙リストのプロパティーと組み合わせた場合にのみ使用できます。

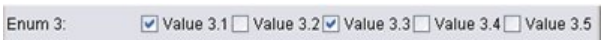

図 *52.* チェック・ボックス・グループ

フォーマット

```
<CheckBoxGroupControl controller_attributes rows="integer" layoutByRow="true_false"
     useSubPanel="true_false" >
     -- advanced custom layout options --
</CheckBoxGroupControl>
```

ここで、

controller_attributes は、「 132 ページの『コントローラの属性』」で説明されているとおりです。

rows は正の整数で、チェック・ボックス・グループは表示される画面の行数を指定します。デフォルトは 1 です。

layoutByRow は、チェック・ボックスを最初は行に沿って配置する (true) か、列に沿って配置する (false) かを指定します。デフォルトは true です。ラジオ・ボタン・グループを持つ layoutByRow の同様の使用法については、 155 ページの『コントロールの順序の変更』を参照してください。

useSubPanel は、チェック・ボックスをサブパネルとして配置する (true) か、配置しない (false) かを指定します。デフォルトは true です。

チェック・ボックス・グループは通常、グループのすべてのボックスを含むサブパネルとして配置されます。ただし、チェック・ボックス・グループが隣接するテキスト・フィールドと関連する場合、整列の問題が生じる場合があります。useSubPanel を false に設定すると、この問題を解決することができます。

詳細なユーザー設定のレイアウト・オプションでは、画面コンポーネントの位置および表示について、正確なコントロールを提供します。 詳しくは、トピック 156 ページの『詳細なカスタム・レイアウト』を参照してください。

例

上記のチェック・ボックス・グループを作成するためのコードは、次のとおりです。

```
<CheckBoxGroupControl property="enum3" label="Enum 3" labelKey="enum3.LABEL"/>
```

個々のチェック・ボックスに関連するラベルおよび値は、関連するノードの Properties セクションで次のように定義されます。

```
<Property name="enum3" valueType="enum" isList="true" defaultValue="[value1 value3]">
     <Enumeration>
          <Enum value="value1" label="Value 3.1" labelKey="enum3.value1.LABEL"/>
          <Enum value="value2" label="Value 3.2" labelKey="enum3.value2.LABEL"/>
          <Enum value="value3" label="Value 3.3" labelKey="enum3.value3.LABEL"/>
          <Enum value="value4" label="Value 3.4" labelKey="enum3.value4.LABEL"/>
          <Enum value="value5" label="Value 3.5" labelKey="enum3.value5.LABEL"/>
     </Enumeration>
</Property>
```

クライアント・ディレクトリー設定コントロール

ユーザーがクライアントのディレクトリーを選択できる単一行のテキスト・フィールドおよび関連ボタンを定義します。ディレクトリーは、常に存在する必要があります。ユーザーは、モード設定に応じてファイルをディレクトリーから開くか、ファイルをディレクトリーに保存することができます。

図 *53.* クライアント・ディレクトリー設定コントロール

ユーザーは、テキスト・フィールドにディレクトリー・パスまたはディレクトリー名を直接入力するか、隣接するボタンをクリックしてディレクトリーを選択できるダイアログを表示することができます。

フォーマット

```
<ClientDirectoryChooserControl controller_attributes mode="chooser_mode" >
     -- advanced custom layout options --
</ClientDirectoryChooserControl>
```

ここで、

controller_attributes は、「 132 ページの『コントローラの属性』」で説明されているとおりです。

mode は、ユーザーがディレクトリーを選択する、ダイアログに表示されるボタンを定義します。ボタンは次のいずれかです。

- open (デフォルト) は、「開く」ボタンを表示します。
- save は、「保存」ボタンを表示します。

詳細なユーザー設定のレイアウト・オプションでは、画面コンポーネントの位置および表示について、正確なコントロールを提供します。 詳しくは、トピック 156 ページの『詳細なカスタム・レイアウト』を参照してください。

例

```
<ClientDirectoryChooserControl property="directory2" label="Client Directory"
     labelKey="directory2.LABEL"/>
```

クライアント・ファイル設定コントロール

ユーザーがクライアントのファイルを選択できる単一行のテキスト・フィールドおよび関連ボタンを定義します。ディレクトリーは、すでに存在する必要があります。ユーザーはノード設定によって、ファイルを開くか保存することができます。

図 *54.* クライアント・ファイル設定コントロール

ユーザーは、テキスト・フィールドにファイル・パスまたはディレクトリー名を直接入力するか、隣接するボタンをクリックしてファイルを選択できるダイアログを表示することができます。

フォーマット

```
<ClientFileChooserControl controller_attributes mode="chooser_mode" >
     -- advanced custom layout options --
</ClientFileChooserControl>
```

ここで、

controller_attributes は、「 132 ページの『コントローラの属性』」で説明されているとおりです。

mode は、ユーザーがファイルを選択する、ダイアログに表示されるボタンを定義します。ボタンは次のいずれかです。

- open (デフォルト) は、「開く」ボタンを表示します。
- save は、「保存」ボタンを表示します。

詳細なユーザー設定のレイアウト・オプションでは、画面コンポーネントの位置および表示について、正確なコントロールを提供します。 詳しくは、トピック 156 ページの『詳細なカスタム・レイアウト』を参照してください。

例

```
<ClientFileChooserControl property="file2" label="Client File" labelKey="file2.LABEL"/>
```

コンボ・ボックス・コントロール

選択情報のドロップダウン・リストを定義します。

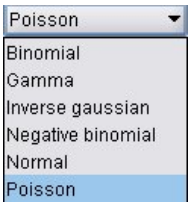

図 *55.* 選択情報

フォーマット

```
<ComboBoxControl controller_attributes >
     -- advanced custom layout options --
</ComboBoxControl>
```

ここで、

controller_attributes は、「 132 ページの『コントローラの属性』」で説明されているとおりです。

詳細なユーザー設定のレイアウト・オプションでは、画面コンポーネントの位置および表示について、正確なコントロールを提供します。 詳しくは、トピック 156 ページの『詳細なカスタム・レイアウト』を参照してください。

例

上記の選択情報のドロップダウン・リストを配置するためのコード例を以下に示します。

```
<ComboBoxControl property="distribution" >
     <Layout rowIncrement="0" gridWidth="1" fill="none"/>
</ComboBoxControl>
```

Layout 要素については、 156 ページの『詳細なカスタム・レイアウト』を参照してください。

注 : 実際のリストの項目は、関連ノードの Properties セクションで定義します。その場合は、以下のように、distribution プロパティーの宣言内で列挙リストとして定義します。

```
<Property name="distribution" valueType="enum" label="Distribution" labelKey="distribution.
LABEL" defaultValue="NORMAL">
     <Enumeration>
          <Enum value="BINOMIAL" label="Binomial" labelKey="distribution.BINOMIAL.LABEL"/>
          <Enum value="GAMMA" label="Gamma" labelKey="distribution.GAMMA.LABEL"/>
          <Enum value="IGAUSS" label="Inverse gaussian" labelKey="distribution.IGAUSS.LABEL"/>
          <Enum value="NEGBIN" label="Negative binomial" labelKey="distribution.NEGBIN.LABEL"/>
          <Enum value="NORMAL" label="Normal" labelKey="distribution.NORMAL.LABEL"/>
          <Enum value="POISSON" label="Poisson" labelKey="distribution.POISSON.LABEL"/>
     </Enumeration>
</Property>
```

データベース接続設定コントロール

ユーザーがデータ・ソースおよびデータベースへの接続を選択できるコントロールを定義します。

図 *56.* データベース接続設定コントロール

ユーザーはテキストをテキスト・フィールドに入力することはできませんが、ボタンをクリックして標準 IBM SPSS Modeler の「データベース接続」ダイアログを表示する必要があります。

正常に接続されると、接続の詳細情報が、データベース接続設定コントロールのテキスト・フィールドに表示されます。

フォーマット

```
<DBConnectionChooserControl controller_attributes >
     -- advanced custom layout options --
</DBConnectionChooserControl>
```

ここで、

controller_attributes は、「 132 ページの『コントローラの属性』」で説明されているとおりです。

詳細なユーザー設定のレイアウト・オプションでは、画面コンポーネントの位置および表示について、正確なコントロールを提供します。 詳しくは、トピック 156 ページの『詳細なカスタム・レイアウト』を参照してください。

例

次の例では、コントロールがどのように接続文字列に使用できる文字列プロパティーの定義を要求するかを示します。

```
<Node ... >
     <Properties>
      ...
          <Property name="dbconnect" valueType="databaseConnection" />
     </Properties>
      ...
     <UserInterface>
      ...
          <Tabs>
               <Tab label="Database">
                    <PropertiesPanel>
                         <DBConnectionChooserControl property="dbconnect" label="Database
                          Connection"/>
                     ...
                    </PropertiesPanel>
                     ...
     </UserInterface>
</Node>
```

データベース・テーブル設定コントロール

データベース接続後、ユーザーがデータベース・テーブルを選択できるテキスト・フィールドおよび関連するボタンを定義します。

図 *57.* データベース・テーブル設定コントロール

ユーザーはテキスト・フィールドに直接テーブル名を入力するか、ボタンをクリックして次のようなリストからテーブル名を選択することができます。

フォーマット

```
<DBTableChooserControl connectionProperty="DB_connection_property" controller_attributes >
     -- advanced custom layout options --
</DBTableChooserControl>
```

ここで、

`connectionProperty` は、すでに定義されているデータベース接続プロパティーの名前です。これは、ノードに以前定義されている `DBConnectionChooserControl` 要素の `property` 属性の値です。

controller_attributes は、「 132 ページの『コントローラの属性』」で説明されているとおりです。

詳細なユーザー設定のレイアウト・オプションでは、画面コンポーネントの位置および表示について、正確なコントロールを提供します。 詳しくは、トピック 156 ページの『詳細なカスタム・レイアウト』を参照してください。

例

この例は DBTableChooserControl の例に従い、データベース接続確率後どのように DBTableChooserControl 要素を使用してデータベース・テーブルを選択するかを示します。

```
<Node ... >
     <Properties>
      ...
          <Property name="dbconnect" valueType="databaseConnection" />
          <Property name="dbtable" valueType="string" />
     </Properties>
       ...
     <UserInterface>
      ...
          <Tabs>
               <Tab label="Database">
                    <PropertiesPanel>
                         <DBConnectionChooserControl property="dbconnect" label="Database
                          connection"/>
                         <DBTableChooserControl property="dbtable" connectionProperty=
                          "dbconnect" label="Database Table" />
                     ...
                    </PropertiesPanel>
      ...
     </UserInterface>
</Node>
```

複数フィールド設定コントロール

ユーザーがリストから 1 つまたは複数のフィールドを選択できるコントロールを定義します。

図 *58.* 複数フィールド設定コントロール

ユーザーがこのコントロールをクリックすると、ユーザーが 1 つまたは複数のフィールドを選択できるフィールドのリストが表示されます。

セットは、このノードで表示されるすべてのフィールドで構成されています。フィールドがこのノードの上流で詳細にフィルタリングされている場合、フィルターを通過したフィールドのみが表示されます。特定のストレージ・タイプおよびデータ型のフィールドのみが選択できるよう指定することによって、リストをより詳細に制限することができます。

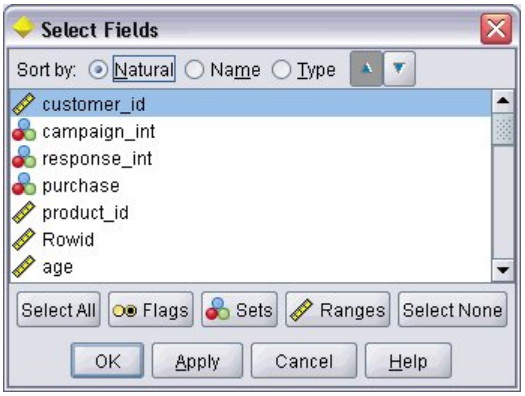

図 *59.* 複数フィールドのリスト

複数フィールド設定コントロールはプロパティー属性を指定します。プロパティー属性はファイル内の別の場所で宣言され、ノードのダイアログにリストがどのように表示されるかを定義します。

フォーマット

```
<MultiFieldChooserControl controller_attributes storage="storage_types" onlyNumeric="true_false"
     onlySymbolic="true_false" onlyDatetime="true_false" types="data_types"
     onlyRanges="true_false" onlyDiscrete="true_false" >
     -- advanced custom layout options --
</MultiFieldChooserControl>
```

ここで、

controller_attributes は、「 132 ページの『コントローラの属性』」で説明されているとおりです。

2 つの別の属性を指定してフィールドのリストをさらに制限することもできます。2 つの属性のいずれかは、次のリストから選択する必要があります。

- `storage` は、リストに表示できるフィールドのストレージ・タイプを指定するリスト・プロパティーです。例えば `storage="[integer real]"` は、これらのストレージ・タイプのフィールドのみ一覧表示されることを意味します。可能なストレージ・タイプのセットについては、 188 ページの『データおよびストレージ・タイプ』 のテーブルを参照してください。
- `onlyNumeric` が `true` に設定されている場合、数値型ストレージのフィールドのみが一覧表示されます。
- `onlySymbolic` が `true` に設定されている場合、シンボル値のストレージ・タイプのフィールド (文字列) のみが一覧表示されます。
- `onlyDatetime` が `true` に設定されている場合、日付と時間のストレージ・タイプのみが一覧表示されます。

2 番目の属性を以下から選択する必要があります。

- `types` は、リスト内に表示されるフィールドのデータ型を指定するリストのプロパティーです。例えば、`types="[range flag]"` は、これらのストレージ・タイプのフィールドのみ一覧表示されることを意味します。使用できるデータ型のセットは次のとおりです。

  ```
  range

  flag

  set
  ```

orderedSet

numeric

discrete

typeless

- onlyRanges が true に設定されている場合は、範囲データ型のフィールドのみ一覧表示されます。
- onlyDiscrete が true に設定されている場合は、離散的な (フラグ型、セット型、データ型不明の) データ型のみ一覧表示されます。

したがって、例えば storage="[integer]" と types="[flag]" を指定するコントロールでは、フラグ型である整数フィールドのみリストに表示されます。

詳細なユーザー設定のレイアウト・オプションでは、画面コンポーネントの位置および表示について、正確なコントロールを提供します。 詳しくは、トピック 156 ページの『詳細なカスタム・レイアウト』を参照してください。

注 : このコントロールは、Node 要素の定義にのみ使用されます。出力データモデルの定義内で複数フィールド設定を指定するには、次の形式を使用します。

```
<OutputDataModel mode="mode">
 ...
     <ForEach var="field" inProperty="prop_name">
          <AddField name="${field_name}_NEW" fieldRef="${field_name}" />
     </ForEach>
 ...
</OutputDataModel>
```

詳しくは、トピック 60 ページの『出力データ・モデル』を参照してください。ForEach 要素については、69 ページの『ForEach 要素による反復』を参照してください。AddField については、 65 ページの『フィールドの追加』を参照してください。

例

上記の複数フィールド設定コントロールを指定するためのコード例を以下に示します。

```
<MultiFieldChooserControl property="inputs" >
     <Enabled>
          <Condition control="custom_fields" op="equals" value="true"/>
     </Enabled>
</MultiFieldChooserControl>
```

custom_fields コントロールが選択されている場合のみ、Enabled セクションによってこのコントロールが有効になります。

注 : このリストの内容は、関連するノードの Properties セクション内の inputs プロパティーの宣言によって制御されます。

```
<Property name="inputs" valueType="string" isList="true" label="Inputs" labelKey="inputs.
LABEL"/>
```

複数項目設定コントロール

ユーザーが値のリストから 1 つまたは複数の項目を選択できるコントロールを定義します。プロパティーを、値のリストを持つカタログと関連付けます。 詳しくは、トピック 42 ページの『カタログ』を参照し

てください。

図 *60.* 複数項目設定コントロール

フォーマット

```
<MultiItemChooserControl controller_attributes catalog="catalog_name" >
     -- advanced custom layout options --
</MultiItemChooserControl>
```

ここで、

controller_attributes は、「 132 ページの『コントローラの属性』」で説明されているとおりです。

`catalog` (必須) は、関連するカタログの名前です。カタログが取得されるライブラリーは、Execution セクションの `Module` 要素で指定されているライブラリーです。 詳しくは、トピック 58 ページの『モジュール』を参照してください。

詳細なユーザー設定のレイアウト・オプションでは、画面コンポーネントの位置および表示について、正確なコントロールを提供します。 詳しくは、トピック 156 ページの『詳細なカスタム・レイアウト』を参照してください。

例

```
<MultiItemChooserControl property="selection2" catalog="cat2" />
```

`property` 属性で参照されるプロパティー (この場合は `selection2`) は、`isList="true"` 属性を持つプロパティーである必要があります。`MultiItemChooserControl` の使用に関する説明と例については、42 ページの『カタログ』を参照してください。

パスワード ボックス・コントロール

入力文字が入力されると非表示になる、単一行のテキスト・フィールドを定義します。

図 *61.* パスワード ボックス・コントロール

フォーマット

```
<PasswordBoxControl controller_attributes columns="integer" >
     -- advanced custom layout options --
</PasswordBoxControl>
```

ここで、

controller_attributes は、「 132 ページの『コントローラの属性』」で説明されているとおりです。

`columns` は正の整数で、パスワード ボックスが占める文字の列数を定義します。デフォルトは 20 です。

詳細なユーザー設定のレイアウト・オプションでは、画面コンポーネントの位置および表示について、正確なコントロールを提供します。 詳しくは、トピック 156 ページの『詳細なカスタム・レイアウト』を参照してください。

例

```
<PasswordBoxControl property="encrypted_string1" label="Encrypted string 1" labelKey=
"encryptedString1.LABEL"/>
```

テキスト・フィールドは、関連するノードの Properties セクション内で暗号化された文字列として定義されているプロパティーと関連させて暗号化します。

```
<Property name="encrypted_string1" valueType="encryptedString"/>
```

プロパティー・コントロール

プロパティー・コントロールは、ユーザーがノードのプロパティーを入力できる、完全にユーザー定義可能なコントロールです。プロセスは、ユーザーによって記述された Java クラスによって処理されます。以下の図に、プロパティー・コントロールの例を示します。

図 62. プロパティー・コントロールの例を強調表示したダイアログのセクション

フォーマット

```
<PropertyControl controller_attributes controlClass="Java_class" >
     -- advanced custom layout options --
</PropertyControl>
```

ここで、

controller_attributes は、「 132 ページの『コントローラの属性』」で説明されているとおりです。

controlClass (必須) は、プロパティー・コントロールを実装する Java クラスへの *.jar* ファイル内のパスです。(*Note* : *.jar* ファイルは、Resources セクションの JarFile 要素で宣言されます。 詳しくは、トピック 35 ページの『Jar ファイル』を参照してください。)

詳細なユーザー設定のレイアウト・オプションでは、画面コンポーネントの位置および表示について、正確なコントロールを提供します。 詳しくは、トピック 156 ページの『詳細なカスタム・レイアウト』を参照してください。

例

```
<PropertyControl property="target_field_values_specify" labelAbove="true"
     controlClass="com.spss.clef.selflearning.propertycontrols.list.CustomListControl" label=""
     labelKey="target_field_values_specify.LABEL">
     <Enabled>
```

```
          <Condition control="target_field_values" op="equals" value="specify"/>
     </Enabled>
     <Layout rowIncrement="2" />
</PropertyControl>
```

プロパティー・コントロールは、関連ノードの Properties セクションで定義されたプロパティーと関連します。

```
<Property name="target_field_values_specify" valueType="string" isList="true" label=""
     labelKey="target_field_values_specify.LABEL"/>
```

ラジオ・ボタン・グループ・コントロール

1 回に 1 つのボタンのみ選択できるラジオ・ボタンのセットを定義します。

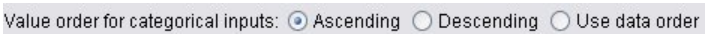

図 *63.* ラジオ・ボタン・グループ・コントロール

ラジオ・ボタン・グループ・コントロールには、それぞれグループを特定のプロパティーと関連付けるプロパティー属性があります。このプロパティーは、ファイル内の別の場所で定義され、グループを構成するボタンを指定します。

関連するプロパティーは、列挙リストまたはブール・プロパティーとなります。列挙リスト (プロパティー属性が `valueType="enum"`) の場合、各 `enum` 値にラジオ・ボタンが 1 つずつ表示されます。ブール・プロパティー (where `valueType="boolean"`) の場合、2 つのボタンが常に表示されます。

フォーマット

```
<RadioButtonGroupControl controller_attributes
     rows="integer" layoutByRow="true_false" useSubPanel="true_false"
     falseLabel="button_label" falseLabelKey="label_key" trueLabel="?button_label"
     trueLabelKey="label_key" trueFirst="true_false" >
     -- advanced custom layout options --
</RadioButtonGroupControl>
```

ここで、

controller_attributes は、「 132 ページの『コントローラの属性』」で説明されているとおりです。

`rows` は正の整数で、グループが表示される画面の行数を指定します。デフォルトは 1 です。

`layoutByRow` は、ラジオ・ボタンを最初は行に沿って配置する (`true`) か、列に沿って配置する (`false`) かを指定します。デフォルトは `true` です。ラジオ・ボタン・グループの `layoutByRow` の同様の使用例については、 155 ページの『コントロールの順序の変更』を参照してください。

`useSubPanel` は、ラジオ・ボタンをサブパネルとして配置する (`true`) か、配置しない (`false`) かを指定します。デフォルトは `true` です。

ラジオ・ボタン・グループは通常、グループのすべてのボタンを含むサブパネルとして配置されます。ただし、ラジオ・ボタン・グループが隣接するテキスト・フィールドと関連する場合、整列の問題が生じる場合があります。`useSubPanel` を `false` に設定すると、この問題を解決することができます。

`falseLabel` は、ブール・プロパティーの偽 (false) の値のラベルです (下の 2 番目の例を参照)。ブール・プロパティーでのみ使用され、その場合は必須です。

falseLabelKey は、ローカライズ用の偽 (false) のラベルを識別します。

trueLabel は、ブール・プロパティーの真 (true) の値のラベルです (下の 2 番目の例を参照)。ブール・プロパティーでのみ使用され、その場合は必須です。

trueLabelKey ローカライズ用の真 (true) のラベルを識別します。

true に設定されている場合、trueFirst はブール・プロパティーのボタンの表示順を逆にし、真 (true) の値を表すボタンが最初に表示されるようにします。デフォルトは false で、偽 (false) を表すボタンが最初に表示されます。

詳細なユーザー設定のレイアウト・オプションでは、画面コンポーネントの位置および表示について、正確なコントロールを提供します。 詳しくは、トピック 156 ページの『詳細なカスタム・レイアウト』を参照してください。

例

最初の例は、上記のラジオ・ボタン・グループで使用されるコードを示しています。

```
<RadioButtonGroupControl property="value_order" labelWidth="2">
     <Layout gridWidth="4"/>
</RadioButtonGroupControl>
```

Layout 要素については、 156 ページの『詳細なカスタム・レイアウト』を参照してください。

注 : ボタンとラベルの数は、関連ノードの Properties セクションで定義します。その場合は、以下のように、value_order プロパティーの宣言内で列挙リストとして定義します。宣言には、グループのラベルも含まれます。

```
<Property name="value_order" valueType="enum" label="Value order for categorical
   inputs" labelKey="value_order.LABEL">
     <Enumeration>
          <Enum value="Ascending" label="Ascending" labelKey="value_order.Ascending.LABEL"/>
          <Enum value="Descending" label="Descending" labelKey="value_order.Descending.
           LABEL"/>
          <Enum value="DataOrder" label="Use data order" labelKey="value_order.UseDataOrder.
           LABEL"/>
     </Enumeration>
</Property>
```

2 番目の例では、標準またはカスタム設定を有効化するかどうかを制御するなど、以下のようにブール・プロパティーを制御するラジオ・ボタン・グループの falseLabel および trueLabel の使用について示します。

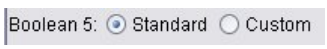

図 *64.* ブール・プロパティーを制御するラジオ・ボタン・グループ

実行するコードは次のとおりです。

```
<RadioButtonGroupControl property="boolean5" label="Boolean 5" labelKey="boolean5.LABEL"
     falseLabel="Standard" falseLabelKey="boolean5.false.LABEL" trueLabel="Custom"
     trueLabelKey="boolean5.true.LABEL" />
```

この場合、ボタンのラベルおよびグループのラベルを RadioButtonGroupControl 要素自体で定義されます。グループが関連するプロパティーは、次のようにノードの Properties セクションで定義されます。

```
<Property name="boolean5" valueType="boolean" defaultValue="false"/>
```

サーバー・ディレクトリー設定コントロール

ユーザーがサーバーのディレクトリーを選択できる単一行のテキスト・フィールドおよび関連ボタンを定義します。ディレクトリーは、常に存在する必要があります。ユーザーは、モード設定に応じてファイルをディレクトリーから開くか、ファイルをディレクトリーに保存することができます。

図 *65.* サーバー・ディレクトリー設定コントロール

ユーザーは、テキスト・フィールドにディレクトリー・パスまたはディレクトリー名を直接入力するか、隣接するボタンをクリックしてディレクトリーを選択できるダイアログを表示することができます。

フォーマット

```
<ServerDirectoryChooserControl controller_attributes mode="chooser_mode" >
     -- advanced custom layout options --
</ServerDirectoryChooserControl>
```

ここで、

controller_attributes は、「 132 ページの『コントローラの属性』」で説明されているとおりです。

mode は、ユーザーがディレクトリーを選択する、ダイアログに表示されるボタンを定義します。ボタンは次のいずれかです。

- open (デフォルト) は、「開く」ボタンを表示します。
- save は、「保存」ボタンを表示します。

詳細なユーザー設定のレイアウト・オプションでは、画面コンポーネントの位置および表示について、正確なコントロールを提供します。 詳しくは、トピック 156 ページの『詳細なカスタム・レイアウト』を参照してください。

例

```
<ServerDirectoryChooserControl property="directory1" label="Server Directory"
     labelKey="directory1.LABEL"/>
```

サーバー・ファイル設定コントロール

ユーザーがサーバーのファイルを選択できる単一行のテキスト・フィールドおよび関連ボタンを定義します。ディレクトリーは、すでに存在する必要があります。ユーザーはノード設定によって、ファイルを開くか保存することができます。

図 *66.* サーバー・ファイル設定コントロール

ユーザーは、テキスト・フィールドにファイル・パスまたはディレクトリー名を直接入力するか、隣接するボタンをクリックしてファイルを選択できるダイアログを表示することができます。

フォーマット

```
<ServerFileChooserControl controller_attributes mode="chooser_mode" >
     -- advanced custom layout options --
</ServerFileChooserControl>
```

ここで、

controller_attributes は、「 132 ページの『コントローラの属性』」で説明されているとおりです。

mode は、ユーザーがファイルを選択する、ダイアログに表示されるボタンを定義します。ボタンは次のいずれかです。

- open (デフォルト) は、「開く」ボタンを表示します。
- save は、「保存」ボタンを表示します。

詳細なユーザー設定のレイアウト・オプションでは、画面コンポーネントの位置および表示について、正確なコントロールを提供します。 詳しくは、トピック 156 ページの『詳細なカスタム・レイアウト』を参照してください。

例

```
<ServerFileChooserControl property="file1" label="Server File" labelKey="file1.LABEL"/>
```

単一フィールド設定コントロール

ユーザーがリストから単一フィールドを選択できるコントロールを定義します。

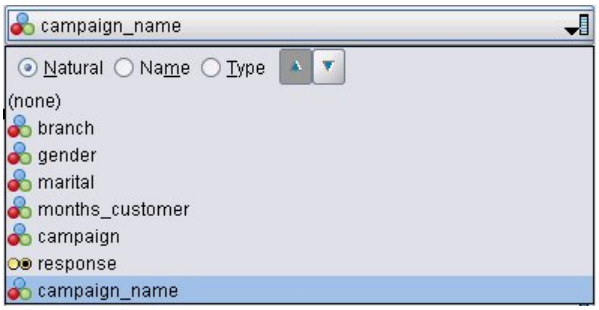

図 *67.* 単一フィールド設定コントロール

ユーザーがこのコントロールをクリックすると、単一のフィールドを選択できるフィールドのリストが表示されます。

セットは、このノードで表示されるすべてのフィールドで構成されています。フィールドがこのノードの上流で詳細にフィルタリングされている場合、フィルターを通過したフィールドのみが表示されます。特定のストレージ・タイプおよびデータ型のフィールドのみが選択できるよう指定することによって、リストをより詳細に制限することができます。

フォーマット

```
<SingleFieldChooserControl controller_attributes storage="storage_types" onlyNumeric="true_
     false" onlySymbolic="true_false" onlyDatetime="true_false" types="data_types"
     onlyRanges="true_false" onlyDiscrete="true_false" >
     -- advanced custom layout options --
</SingleFieldChooserControl>
```

ここで、

controller_attributes は、「 132 ページの『コントローラの属性』」で説明されているとおりです。

2 つの別の属性を指定してフィールドのリストをさらに制限することもできます。2 つの属性のいずれかは、次のリストから選択する必要があります。

- storage は、リストに表示できるフィールドのストレージ・タイプを指定するリスト・プロパティーです。例えば storage="[integer real]" は、これらのストレージ・タイプのフィールドのみ一覧表示されることを意味します。可能なストレージ・タイプのセットについては、188 ページの『データおよびストレージ・タイプ』 のテーブルを参照してください。
- onlyNumeric が true に設定されている場合、数値型ストレージのフィールドのみが一覧表示されます。
- onlySymbolic が true に設定されている場合、シンボル値のストレージ・タイプのフィールド (文字列) のみが一覧表示されます。
- onlyDatetime が true に設定されている場合、日付と時間のストレージ・タイプのみが一覧表示されます。

2 番目の属性を以下から選択する必要があります。

- types は、リスト内に表示されるフィールドのデータ型を指定するリストのプロパティーです。例えば、types="[range flag]" は、これらのストレージ・タイプのフィールドのみ一覧表示されることを意味します。使用できるデータ型のセットは次のとおりです。

  range

  flag

  set

  orderedSet

  numeric

  discrete

  typeless
- onlyRanges が true に設定されている場合は、範囲データ型のフィールドのみ一覧表示されます。
- onlyDiscrete が true に設定されている場合は、離散的な (フラグ型、セット型、データ型不明の) データ型のみ一覧表示されます。

したがって、例えば storage="[integer]" と types="[flag]" を指定するコントロールでは、フラグ型である整数フィールドのみリストに表示されます。

詳細なユーザー設定のレイアウト・オプションでは、画面コンポーネントの位置および表示について、正確なコントロールを提供します。 詳しくは、トピック 156 ページの『詳細なカスタム・レイアウト』を参照してください。

注 : このコントロールは、ノード定義についてのみ使用されます。出力データモデルの定義内で複数フィールド設定を指定するには、次の形式を使用します。

```
<OutputDataModel mode="mode">
 ...
    <ForEach var="field" from="1" to="${integer}">
```

```
        <AddField name="${string}_${field}" fieldRef="${field_ref}" />
    </ForEach>
 ...
</OutputDataModel>
```

詳しくは、トピック 60 ページの『出力データ・モデル』を参照してください。ForEach 要素については、 69 ページの『ForEach 要素による反復』を参照してください。AddField については、 65 ページの『フィールドの追加』を参照してください。

例

上記の単一フィールド設定コントロールを指定するためのコード例を以下に示します。

```
<SingleFieldChooserControl property="target"  storage="string" onlyDiscrete="true"/>
```

注 : 実際のリストの内容は、関連ノードの Properties セクションで定義されます。この場合、target プロパティーの宣言内で定義されます。

```
<Property name="target" valueType="string" label="Target field" labelKey="target.LABEL"/>
```

単一項目設定コントロール

ユーザーが値のリストから単一項目を選択できるコントロールを定義します。プロパティーを、値のリストを持つカタログと関連付けます。 詳しくは、トピック 42 ページの『カタログ』を参照してください。

図 *68.* 単一項目設定コントロール

フォーマット

```
<SingleItemChooserControl controller_attributes catalog="catalog_name" >
     -- advanced custom layout options --
</MultiItemChooserControl
```

ここで、

controller_attributes は、「 132 ページの『コントローラの属性』」で説明されているとおりです。

catalog (必須) は、関連するカタログの名前です。カタログが取得されるライブラリーは、Execution セクションの Module 要素で指定されているライブラリーです。 詳しくは、トピック 58 ページの『モジュール』を参照してください。

詳細なユーザー設定のレイアウト・オプションでは、画面コンポーネントの位置および表示について、正確なコントロールを提供します。 詳しくは、トピック 156 ページの『詳細なカスタム・レイアウト』を参照してください。

例

```
<SingleItemChooserControl property="selection1" catalog="cat1" />
```

このコントロールの使用に追加の説明および例は、 42 ページの『カタログ』を参照してください。

スピン・コントロール

スピン・コントロール (値を変更するための、上下の矢印の付いた数値型フィールド) を定義します。

図 *69.* スピン

フォーマット

```
<SpinnerControl controller_attributes columns="integer" stepSize="increment"
     minDecimalDigits="number" maxDecimalDigits="number" >
     -- advanced custom layout options --
</SpinnerControl>
```

ここで、

controller_attributes は、「 132 ページの『コントローラの属性』」で説明されているとおりです。

`columns` は正の整数で、コントロールが使用する文字の列数を指定します。デフォルトは 5 です。

`stepSize` は少数で、ユーザーが上下矢印のいずれかをクリックした場合にフィールド値が変更する数量を指定します。デフォルトは 1.0 です。

`minDecimalDigits` は、フィールド値に表示される小数部の桁数の最小値です。デフォルトは 1 です。

`maxDecimalDigits` は、フィールド値に表示される小数部の桁数の最大値です。

詳細なユーザー設定のレイアウト・オプションでは、画面コンポーネントの位置および表示について、正確なコントロールを提供します。 詳しくは、トピック 156 ページの『詳細なカスタム・レイアウト』を参照してください。

例

上記のスピン・コントロールを指定するためのコード例を以下に示します。

```
<SpinnerControl property="double1" label="Double 1" labelKey="double1.LABEL"/>
```

数値型フィールドの内容、精度および有効な範囲は関連ノードの Properties セクションに定義されます。この場合、`double1` プロパティーの宣言に定義されます。

```
<Property name="double1" valueType="double" min="0" max="100"/>
```

テーブル・コントロール

ノードのダイアログまたは出力ウィンドウに表示されるテーブルのレイアウト項目を定義します。

図 *70.* テーブル・コントロール

フォーマット

```
<TableControl controller_attributes rows="integer" columns="integer" columnWidths="list" >
     -- advanced custom layout options --
</TableControl>
```

ここで、

controller_attributes は、「 132 ページの『コントローラの属性』」で説明されているとおりです。

rows は正の整数で、画面上に表示されるテーブルの行数を指定します。デフォルトは 8 です。

columns は正の整数で、テーブルが占める文字の列数を指定します。デフォルトは 20 です。

columnwidths は、関連する列の幅を指定する値のリストを表示します。例えば [30 5 10] の値は、列 1 が列 3 の 3 倍の幅になるよう指定します。

詳細なユーザー設定のレイアウト・オプションでは、画面コンポーネントの位置および表示について、正確なコントロールを提供します。 詳しくは、トピック 156 ページの『詳細なカスタム・レイアウト』を参照してください。

次の例の *ColumnControl* 属性は、152 ページの『列コントロール』で説明しています。

例

上記のテーブル・コントロールを指定するためのコードを以下に示します。

```
<TableControl property="structure1" allowReorder="true" label="Structured"
  labelKey="structure1.LABEL" columnWidths="[20 6 10 10]">
     <ColumnControl column="0" editor="fieldValue" fieldControl="field1"/>
</TableControl>
```

テーブル・コントロールの構造は、次のように仕様ファイルの Common Objects セクションのプロパティーの種類として定義されます。

```
<PropertyType id="shared_structure1" valueType="structure" isList="true">
     <Structure>
          <Attribute name="id" valueType="string" label="Name" labelKey="structure1.id.LABEL"/>
          <Attribute name="yesno" valueType="boolean" label="Yes/No" labelKey="structure1.
           yesno.LABEL" defaultValue="true"/>
          <Attribute name="count" valueType="integer" label="Count" labelKey="structure1.
           count.LABEL" defaultValue="0"/>
          <Attribute name="limit" valueType="double" label="Limit" labelKey="structure1.
           limit.LABEL" defaultValue="0.0"/>
     </Structure>
</PropertyType>
```

ノード設定では、このプロパティーの種類の識別子はプロパティーの宣言によってテーブル・コントロールの識別子と関連付けられます。

```
<Property name="structure1" type="shared_structure1"/>
```

スクリプトのノードを参照する場合、リストに大カッコ [] および構造に中カッコ {} を使用して、値をプロパティーに設定することができます。例えば、次のように structure1 プロパティーの 2 つの構造のグリッドを設定できます。

```
set :node_ID.structure1 = [{"hello" true 4 0.21} {"bye" false 5 0.95}]
```

値の順番は、Attribute 定義が作成される順番と一貫している必要があります。

列コントロール: テーブル内の列のレイアウトを定義します。

テーブル・コントロールのすべての列は同じデータ型を共有します。そのため、すべての行の特定の列について、1 つのエディターを指定できます。したがって、すべての列は編集用に 1 つのエディターのみが必要になります。例えば、ユーザーに列 X に整数を入力させる必要がある場合、列 X に整数エディターを設定できます。

属性 *editor* は、列のエディター・タイプを指定します。エディターのタイプには、*default*、*field*、*fieldValue*、および *enumeration* の 4 つがあり、各タイプのエディターは編集可能なコンボ・ボックスになっています。

fieldValue タイプ・エディターの場合、ドロップダウン・リストに、*fieldControl* に指定したフィールドのすべての値が含まれています。したがって、以下の XML エレメントでは、列 0 を編集する際に、エディターはコンボ・ボックスであり、ドロップダウン・リストに field1 のすべての値が含まれることを定義しています。

```
<ColumnControl column="0" editor="fieldValue" fieldControl="field1" />
```

fieldControl は *fieldDirection* で置き換えることができます。例えば、*fieldDirecton="[in out]"* は、コンボ・ボックスのドロップダウン・リストに、方向が *in* または *out* である最初のフィールドのすべての値が含まれることを意味します。

field タイプ・エディターの場合、ドロップダウン・リストには、フィールド・フィルター条件に一致するすべてのフィールドが含まれます。以下の例では、列 0 にエディターとしてフィールド・コンボ・ボックスを使用し、ドロップダウン・リストにすべての実数フィールドと整数フィールドを含めるよう定義しています。

```
<ColumnControl column="0" editor="field" storage="[real integer]" />
```

さらに、属性 *types* を使用して、ドロップダウン・リスト内のフィールドが従うべき測定タイプを指定できます。このフィールドに使用できるブール属性は、*onlyRanges*、*onlyDiscrete*、*onlyNumeric*、*onlySymbolic*、および *onlyDatetime* です。

テキスト領域コントロール

複数行のテキスト・エントリー・フィールドを定義します。

図 *71.* テキスト領域コントロール

フォーマット

```
<TextAreaControl controller_attributes rows="integer" columns="integer" wrapLines="true_false" >
     -- advanced custom layout options --
</TextAreaControl>
```

ここで、

controller_attributes は、「 132 ページの『コントローラの属性』」で説明されているとおりです。

`rows` は正の整数で、テキスト領域が占める画面上の行数を指定します。デフォルトは 8 です。

columns は正の整数で、テキスト領域が占める文字の列数を指定します。デフォルトは 20 です。

wrapLines は、長いテキスト行に折り返しを使用する (true) または長いテキスト行を読む場合に水平スクロールを使用する (false) かどうかを指定します。デフォルトは true です。

詳細なユーザー設定のレイアウト・オプションでは、画面コンポーネントの位置および表示について、正確なコントロールを提供します。 詳しくは、トピック 156 ページの『詳細なカスタム・レイアウト』を参照してください。

例

上記の例を作成するためのコードは、次のとおりです。

```
<TextAreaControl property="string2" label="String 2" labelKey="string2.LABEL"/>
```

この場合、入力データ型は関連ノードの Properties で定義されますが、テキスト領域のラベルはテキスト領域コントロールの宣言 (string2 プロパティーの宣言) で定義されています。

```
<Property name="string2" valueType="string"/>
```

テキスト・ボックス・コントロール

単一行のテキスト・エントリー・フィールドを定義します。

図 72. テキスト・ボックス・コントロール

フォーマット

```
<TextBoxControl controller_attributes columns="integer" >
     -- advanced custom layout options --
</TextBoxControl>
```

ここで、

controller_attributes は、「 132 ページの『コントローラの属性』」で説明されているとおりです。

columns は正の整数で、テキスト・ボックスが占める文字の列数を指定します。デフォルトは 20 です。

詳細なユーザー設定のレイアウト・オプションでは、画面コンポーネントの位置および表示について、正確なコントロールを提供します。 詳しくは、トピック 156 ページの『詳細なカスタム・レイアウト』を参照してください。

例

上記のテキスト・ボックスを作成するためのコードは、次のとおりです。

```
<TextBoxControl property="string1" label="String 1" labelKey="string1.LABEL"/>
```

テキスト・ボックスのラベルおよび入力データ型は、関連ノードの Properties セクションに定義されます。この場合、string1 プロパティーの宣言に定義されます。

```
<Property name="string1" valueType="string"/>
```

プロパティー・コントロールのレイアウト

この項では、ダイアログおよびウィンドウに使用される標準的なレイアウト方法およびこのレイアウトを変更して独自のカスタム・レイアウトの取得方法について説明します。

標準的なコントロールのレイアウト

プロパティー・パネルは、セルの 2 次元グリッドとしてみなすことができます。それぞれの行は高さが異なり、それぞれの列は幅が異なります。UI コンポーネントを複数の連続セルに割り当てることができますが、通常 UI コンポーネントは 1 つのセルにのみ割り当てられます。

デフォルトでは、1 つのプロパティー・コントロールは 1 つの行に割り当てられ、各コントロールに 2 列ずつ割り当てられます。一方の列はラベル用、もう一方はコントロール・コンポーネントまたはコンポーネント用です。ラベルを含む列は、最も広いラベルの幅にまで広がります。例えば、仕様ファイルに次の要素が指定されている場合

```
<TextBoxControl property="string1" label="String 1"/>
<PasswordBoxControl property="encryptedString1" label="Encrypted string 1"/>
<TextAreaControl property="string2" label="String 2"/>
```

以下に示すパネルが表示されます。

図 *73.* 単純なプロパティー・パネル

「:」文字は、ラベルの最後に自動的に追加されます。

複数のユーザー・インターフェース・コンポーネントを含むプロパティー・コントロールは、これらのコンポーネントをレイアウトする表示されない四角形の領域を作成します。`RadioButtonGroupControl` と `CheckBoxGroupControl` 要素がこのようなコントロールの例となります。

コンポーネントがレイアウトされる四角形の領域の形は、プロパティー・コントロールによって異なります。そのため、各種コントロールのレイアウトは、常に厳密に調整されるわけではありません。

一部のプロパティー・コントロールには、コンポーネントの列を完全に満たすコンポーネントが含まれ、ウィンドウの幅が増減すると幅のサイズが変更されます。例えば `TextBoxControl` 要素、`PasswordBoxControl` 要素、`TextAreaControl` 要素によって指定されたコントロールがこれに該当します。ただし、すべてのコンポーネントで実行されるわけではありません。例えば、ウィンドウの幅が拡大されている場合でも、チェック・ボックスおよびスピン・コントロールは水平方向の領域の固定値のみをとります。

カスタム・コントロール・レイアウト

コントロールの標準レイアウトは、単純な方法または複雑な方法によって変更することができます。

単純なカスタム・レイアウト

コントロール レイアウトをカスタマイズする単純な方法は、つぎの 3 つです。

- コンポーネントの上にラベルを配置する
- コントロールが配置される行数を変更する

- コントロールが配置される順序を変更する

コンポーネントの上へのラベルの配置: コントロールの labelAbove 属性を true に設定してコンポーネントの上の各行にラベルを配置できます。以下に例を示します。

```
<TextBoxControl property="string0" label="String 0" labelAbove="true"/>
<TextBoxControl property="string1" label="String 1"/>
<PasswordBoxControl property="encryptedString1" label="Encrypted string 1"/>
```

コンポーネントの上にラベルを配置するとともに、実際の UI コンポーネントまたはコンポーネントは、画面のラベルの列に割り当てられます。これにより、対応するフィールドの上部にラベル「文字列 **0**」が表示される以下のパネルが作成されます。

図 *74.* 各行にフィールド・ラベルを持つパネル

行数の変更: デフォルトではラジオ・ボタン・グループとチェック・ボックス・グループは 1 行にレイアウトされ、ダイアログの幅はそれに合わせて調整されます。ラジオ・ボタン・グループまたはチェック・ボックス・グループには多数のオプションがあるため、ダイアログの幅が非常に広くなります。コントロールのレイアウトに使用される行数を変更して、これを回避できます。コントロール定義の rows 属性を該当する値に設定して行数を変更します。以下に例を示します。

```
<RadioButtonGroupControl property="enum1" label="Enum 1" rows="2"/>
```

これにより、2 行にラジオ・ボタン・グループが配置されたパネルが作成されます。

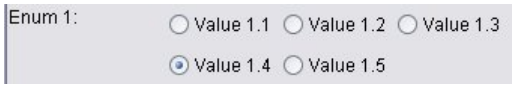

図 *75. 2* 行にラジオ・ボタン・グループが配置されたパネル

コントロールの順序の変更: ラジオ・ボタンおよびチェック・ボックス・グループの場合、enum 値のコントロールがパネルに追加される順序を変更することもできます。

デフォルトでは、前述の例のようにコントロールは行順に追加され、1 番目、2 番目、3 番目の値が最初の行に、4 番目および 5 番目の値が 2 番目の行に追加されます。代わりに、layoutByRow を false に設定して、指定された行数内でコントロールを列順に追加することができます。以下に例を示します。

```
<RadioButtonGroupControl property="enum1" label="Enum 1" rows="2" layoutByRow="false"/>
```

値は 2 行にわたって表示されますが、1 番目および 2 番目の値が最初の列に、3 番目および 4 番目の値が 2 番目の列に、そして 5 番目の値は 3 番目の列に追加されます。

図 *76.* 列順にラジオ・ボタン・グループが配置されたパネル

2 つのラジオ・ボタンとして表示されるブール・プロパティーの場合、デフォルトの順序によって、「真 (True)」のボタンの前に「偽 (False)」のボタンが表示されます。trueFirst 属性を true に設定して、この順序を逆にすることができます。

useSubPanel 属性を false に設定して、ラジオ・ボタンおよびチェック・ボックス・グループでサブパネルを使用できないようにすることもできます。ただし、Layout 要素と組み合わせて使用しない限り、意図しないレイアウトになる場合があります (『Layout 要素を使用したコントロール位置の正確な指定』を参照してください)。

詳細なカスタム・レイアウト

各コントロールの宣言内で、さまざまな要素を使用して複雑なコントロールのレイアウトを指定することができます。以下を行うことができます。

- Layout 要素を使用して、画面上のコントロールを正確に指定する
- Enabled 要素を使用して、表示の特性を制御する
- Visible 要素を使用して、画面コンポーネントが表示されるかどうかを制御する

Layout 要素を使用したコントロール位置の正確な指定: 次のように、明示的な Layout 要素を指定してコントロールと関連付けることによって、正確なレイアウトの位置を指定することができます。

フォーマット

```
<property_control ... >
     <Layout attributes
      --- cell specification ---
          ...
</property_control>
```

ここで、

property_control プロパティー・コントロールの 1 つです (125 ページの『プロパティー・コントロール設定』を参照）。

attributes は、以下の表に示す任意の属性です。

表 *39.* レイアウト属性 :

属性	値	説明
anchor	north northeast east southeast south southwest west northwest center	コントロールのアンカー・ポイントを定義します。
columnWeight	0.0–1.0	ウィンドウの水平方向のサイズ変更がコントロールの幅に与える影響を定義します。パネル内のすべての columnWeight 属性の合計は、1.0 を超えることはできません。

表 39. レイアウト属性 (続き):

属性	値	説明
fill	none horizontal vertical both	コントロールが割り当てられたセルに入力するかどうか、どのように入力するかを定義します。
gridColumn	整数または 0	コントロールのレイアウトが開始する最初の列を定義します。
gridHeight	integer	コントロールが配置される行数を定義します。0 の値 (デフォルト) を使用すると、残りの行すべてにコントロールを割り当てます。
gridRow	整数または 0	コントロールのレイアウトが開始する最初の行を定義します。デフォルトでは、グリッドの行のインデックスが自動的に増加します。
gridWidth	integer	コントロールが配置される列数を定義します。0 の値 (デフォルト) を使用すると、残りの列すべてにコントロールを割り当てます。
leftIndent	integer	デフォルトの位置からコントロールをインデントするピクセル数を定義します。
rowWeight	0.0–1.0	ウィンドウの垂直方向のサイズ変更がコントロールの高さに与える影響を定義します。パネル内のすべての rowWeight 属性の合計は、1.0 を超えることはできません。

cell specification を使用すると、画面上のコントロールの正確な位置を指定します。形式は次のとおりです。

```
<Cell row="integer" column="integer" width="integer" />
```

ここで、

row (必須) は負ではない整数で、コントロールが開始する行の位置を指定します。

column (必須) は負ではない整数で、コントロールが開始する列の位置を指定します。

width (必須) は負ではない整数で、コントロールが占める画面上のグリッド列の数を指定します。

例えば、画面のグリッドが 3 行 3 列であると仮定し、以下の形式のカスタム・コントロール・レイアウトを考えてみます。

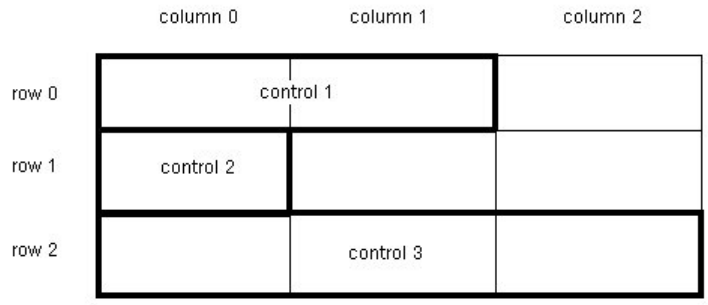

図 77. セルを使用したコントロールのレイアウトの例

この形式の場合は、以下のセルが指定された Layout 要素が必要になります。

```
<Layout ... >
     <Cell row="0" column="0" width="2">
     <Cell row="1" column="0" width="1">
     <Cell row="2" column="0" width="3">
</Layout>
```

次の例では、Layout 要素の使用方法について詳細に説明しています。

例 *:*テキスト・フィールドを有効化するチェック・ボックス*:* この例では、画面の同じ行にあるテキスト・フィールドを有効化するチェック・ボックスの使用について説明します。

チェック・ボックスを使用して同じ行の別のコントロールを有効にする場合は、コントロールが正しく表示されるように、単純な Layout 要素が必要です (注 : コントロールの有効と無効を切り替えるメカニズムについては、 163 ページの『Enabled 要素を使用した表示の特性の制御』を参照してください)。

画面に以下のパネルを表示する場合を考えてみます。

図 *78.* テキスト・フィールドを有効化するチェック・ボックス

以下の 2 つのコントロールがあります。

- テキスト・フィールドのラベルとしても動作するラベルを持つチェック・ボックス
- テキスト・フィールド

開始ポイントは、2 つのコントロールの通常の宣言です。

```
<CheckBoxControl property="boolean3" label="Check box 3"/>
<TextBoxControl property="string3" label="String 3"/>
```

これにより、以下のパネルが表示されます。

図 *79.* 各行のチェック・ボックスおよびテキスト・フィールド

まず、テキスト・フィールド・ラベル 「文字列 **3**」 を非表示にするとします。テキスト・フィールド・コントロールの showLabel 属性を false に設定して非表示にします。

```
<CheckBoxControl property="boolean3" label="Check box 3"/>
<TextBoxControl property="string3" label="String 3" showLabel="false"/>
```

テキスト・フィールドは、ラベルが以前占めていた領域を満たすまで拡大します。

図 *80.* ラベルのないチェック・ボックスおよびテキスト・フィールド

テキスト・フィールドをチェック・ボックスと同じ行に表示するとします。このためには、CheckBoxControl 要素内に Layout 要素を追加して、行の増分を 0 に設定します (デフォルトでは、行はコントロールごとに 1 ずつ増加します。)。

```
<CheckBoxControl property="boolean3" label="Check box 3">
     <Layout rowIncrement="0"/>
</CheckBoxControl>
<TextBoxControl property="string3" label="String 3" showLabel="false"/>
```

ただし、この場合は以下のように表示されます。

図 *81.* チェック・ボックスに重複するテキスト・フィールド

テキスト・フィールドは 1 行上に移動しますが、行全体に占められているため、チェック・ボックスに重複します。

注 : 以下のように表示される場合、チェック・ボックスはテキスト・フィールドの後に描画されます。

図 *82.* テキスト・フィールドに重複するチェック・ボックス

この場合、テキスト・フィールドの先頭から数文字は隠れてしまいます。

どちらのオブジェクトを最初に描画した場合でも、複数の UI コンポーネントを同じセルに割り当てると望まない動作または未定義の動作が生じるため、避ける必要があります。この問題を解決するには、2 番目の Layout 要素を追加する必要があり、次の例では TextBoxControl 要素内でテキスト・フィールドに画面の 2 番目の列から開始させます。

```
<CheckBoxControl property="boolean3" label="Check box 3">
     <Layout rowIncrement="0"/>
</CheckBoxControl>
<TextBoxControl property="string3" label="String 3" showLabel="false">
     <Layout gridColumn="1"/>
</TextBoxControl>
```

しかし、これは部分的な解決策にすぎません。両方のコントロールが正しく配置されますが、以下の表示のように、テキスト・フィールドが非常に短くなります。

図 *83.* 正しく配置されるが、テキスト・フィールドが非常に短い

問題は、カスタム・レイアウトがコントロールと関連すると、各種のコントロールに関連する「スマートな」デフォルトを上書きすることです。この場合、Layout 要素のセルを満たすデフォルトの操作 (コンポーネントが使用できるセルをどのように満たすか) は、使用可能なセルを満たすことではなく、画面上で可能な限り小さい領域になります。これを変更するには、テキスト・フィールドを水平方向の領域を満たすよう指示します。

```
<CheckBoxControl property="boolean3" label="Check box 3">
     <Layout rowIncrement="0"/>
</CheckBoxControl>
<TextBoxControl property="string3" label="String 3" showLabel="false">
     <Layout gridColumn="1" fill="horizontal" columnWeight="0.001"/>
</TextBoxControl>
```

Java によって入力領域を正しく割り振るには、小さな columnWeight 値を追加する必要があります。

これにより、意図したとおりのレイアウトになります。

図 *84.* テキスト・フィールドを有効化するチェック・ボックス

正しく表示されているように思われますが、処理する必要のある問題が 1 つあります。同じ行に別のコントロールを配置する場合でも、チェック・ボックスを正しく行全体に配置します。問題は、チェック・ボックスのラベルが比較的短いために正しく表示されず、重複しないようパネルの他のラベル (図では表示されていません) によって 2 番目の表示列を移動させていることです。チェック・ボックスのラベルをより長くすると、問題が明確になります。

```
<CheckBoxControl property="boolean3" label="Check box 3 with a much longer label than we had">
     <Layout rowIncrement="0"/>
</CheckBoxControl>
<TextBoxControl property="string3" label="String 3" showLabel="false">
     <Layout gridColumn="1" fill="horizontal" columnWeight="0.001"/>
</TextBoxControl>
```

これにより、以下のように表示されます。

図 *85.* チェック・ボックスの長いラベルにテキスト・フィールドが重複している

チェック・ボックスが 1 列あたりの可能な幅を制限するよう指示する必要があります。

```
<CheckBoxControl property="boolean3" label="Check box 3 with a much longer label than we had">
     <Layout rowIncrement="0" gridWidth="1"/>
</CheckBoxControl>
<TextBoxControl property="string3" label="String 3" showLabel="false">
     <Layout gridColumn="1" fill="horizontal" columnWeight="0.001"/>
</TextBoxControl>
```

これにより、意図したとおりの表示になります。

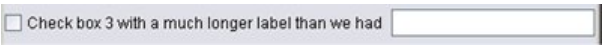

図 *86.* 完全に表示された長いチェック・ボックスのラベル

例 *:*ラジオ・ボタン・グループおよびテキスト・フィールド*:* この例では、各ラジオ・ボタン・グループをそれぞれのテキスト・フィールドと関連付ける方法について説明します。

以下のようなパネルを定義する場合を考えてみます。

図 *87.* テキスト・フィールドを含むラジオ・ボタン・グループ

この例では、次の 4 つのコントロールがあります。

- 3 つの値の列挙リストのラジオ・ボタン・グループ

- 各値に 1 つずつ、3 つのテキスト・フィールド

前述の例のように、コントロールの単純な宣言で開始されます。

```
<RadioButtonGroupControl property="enum4" label="Enum 4"/>
<TextBoxControl property="string4" label="String 4"/>
<TextBoxControl property="string5" label="String 5"/>
<TextBoxControl property="string6" label="String 6"/>
```

これにより、以下のように表示されます。

図 *88.* テキスト・フィールドおよびラベルを含むラジオ・ボタン・グループ

ラジオ・ボタンのラベルを使用してテキスト・フィールドを識別するには、まず 3 行で 1 列にラジオ・ボタンを配置し、テキスト・フィールドのラベルを非表示にします。

```
<RadioButtonGroupControl property="enum4" label="Enum 4" rows="3"/>
<TextBoxControl property="string4" label="String 4" showLabel="false"/>
<TextBoxControl property="string5" label="String 5" showLabel="false"/>
<TextBoxControl property="string6" label="String 6" showLabel="false"/>
```

これにより、以下のように表示されます。

図 *89. 1* 列に並ぶラジオ・ボタンおよびテキスト・フィールド

ここで 1 つの問題があります。ラジオ・ボタン・グループのラベルが最初のラジオ・ボタンに合わせて整列していません。この問題は後で解決します。テキスト・フィールドを対応するそれぞれのラジオ・ボタンの行に大まかに合わせます。

必要な手順は、例 1 と同様、以下のとおりです。

- ラジオ・ボタン・グループの行の増分を 0 に変更します。
- グリッド幅を制限して、テキスト・フィールドとラジオ・ボタンが重複しないようにします。
- ラジオ・ボタンと同じ行の各テキスト・フィールドを配置します。

前述の例と同様、Layout 要素を追加します。この場合、次のように仕様ファイルを変更します。

```
<RadioButtonGroupControl property="enum4" label="Enum 4" rows="3">
     <Layout rowIncrement="0" gridWidth="1"/>
</RadioButtonGroupControl>
<TextBoxControl property="string4" label="String 4" showLabel="false">
     <Layout gridColumn="1" fill="horizontal" columnWeight="0.001"/>
</TextBoxControl>
<TextBoxControl property="string5" label="String 5" showLabel="false">
```

```
     <Layout gridColumn="1" fill="horizontal" columnWeight="0.001"/>
</TextBoxControl>
<TextBoxControl property="string6" label="String 6" showLabel="false">
     <Layout gridColumn="1" fill="horizontal" columnWeight="0.001"/>
</TextBoxControl>
```

しかし、この場合は以下のような表示になります。

図 *90.* ラジオ・ボタンに重複するテキスト・フィールド

例 1 で使用したものとまったく同じ Layout 要素を使用しましたが、何が起こったのでしょう ?

それは、例 1 のチェック・ボックス・コントロールとは異なり、ラジオ・ボタン・グループ (多くのコントロールと同様) には実際のコントロールとともに個別のラベルがあるためです。つまりラジオ・ボタン・グループには追加の列が必要であるため、テキスト・フィールドが後の列、列 1 ではなく列 2 から開始するよう指示する必要があります。テキスト・フィールドの Layout 要素で、gridColumn 値を 2 に設定します。

```
<RadioButtonGroupControl property="enum4" label="Enum 4" rows="3">
     <Layout rowIncrement="0" gridWidth="1"/>
</RadioButtonGroupControl>
<TextBoxControl property="string4" label="String 4" showLabel="false">
     <Layout gridColumn="2" fill="horizontal" columnWeight="0.001"/>
</TextBoxControl>
<TextBoxControl property="string5" label="String 5" showLabel="false">
     <Layout gridColumn="2" fill="horizontal" columnWeight="0.001"/>
</TextBoxControl>
<TextBoxControl property="string6" label="String 6" showLabel="false">
     <Layout gridColumn="2" fill="horizontal" columnWeight="0.001"/>
</TextBoxControl>
```

テキスト・フィールドのグリッド列は 2 に増加しますが、ラジオ・ボタン・グループのグリッド幅は 1 から増加しません。これは、プロパティー・コントロールで、ほとんどの Layout 属性がコントロールのラベルではなく、コントロールの編集可能な部分を構成する UI コンポーネントにのみ影響を与えるためです。

これで、以下のように表示されます。

図 *91.* ラジオ・ボタンに重複していないテキスト・フィールド

希望のレイアウトにかなり近づきました。ただし、ラジオ・ボタンおよびテキスト・フィールド間の位置合わせの問題がいくつか残っています。

主な問題は、ラジオ・ボタンが各サブパネルに配置されているため、ラジオ・ボタンとテキスト・フィールドの間に実際のレイアウトの関係がないことです。サブパネルを使用して、次のようにラジオ・ボタン・グループを停止する必要があります。

```
<RadioButtonGroupControl property="enum4" label="Enum 4" rows="3" useSubPanel="false">
     <Layout rowIncrement="0" gridWidth="1"/>
</RadioButtonGroupControl>
<TextBoxControl property="string4" label="String 4" showLabel="false">
     <Layout gridColumn="2" fill="horizontal" columnWeight="0.001"/>
</TextBoxControl>
<TextBoxControl property="string5" label="String 5" showLabel="false">
     <Layout gridColumn="2" fill="horizontal" columnWeight="0.001"/>
</TextBoxControl>
<TextBoxControl property="string6" label="String 6" showLabel="false">
     <Layout gridColumn="2" fill="horizontal" columnWeight="0.001"/>
</TextBoxControl>
```

これにより、希望のレイアウトが設定されました。

図 *92.* テキスト・フィールドを含むラジオ・ボタン・グループ

Enabled 要素を使用した表示の特性の制御**:** Enabled 要素を使用して、通常は特定の条件を満たすかどうかによって、コントロールを有効化または無効化することができます。

パネルおよびプロパティー・コントロールには、さまざまな表示特性を決定するために関連付けられた条件がある場合があります。例えば、チェック・ボックスを使用して関連するテキスト・フィールド有効化することができます。またはラジオ・ボタンによって非表示フィールドのグループを表示されるようにすることもできます。

ユーザー・インターフェースの条件は通常、プロパティーではなく別のコントロールの値に基づきます。プロパティーに基づく条件は、変更が基本のオブジェクト (ノード、モデル出力、またはドキュメント 出力) に適用された場合にのみ有効です。ユーザー・インターフェースでは、関連するコントロールが変更された後すぐにコントロールを有効化する必要があります。

フォーマット

```
<Enabled>
     <Condition .../>
     <And ... />
     <Or ... />
     <Not ... />
</Enabled>
```

Condition 要素で、条件を指定してコントロールを有効化するかどうかを決定するために検定します。

And、Or、および Not 要素を使用して、複合条件を指定できます。

詳しくは、トピック 73 ページの『条件』を参照してください。

例 *:*単純な条件によるコントロールの有効化*:* 158 ページの『例 :テキスト・フィールドを有効化するチェック・ボックス』 で、ボックスが選択された場合にテキスト・フィールドを有効化できるチェック・ボックスを作成します。

チェック・ボックスが選択されるとすぐにテキスト・フィールドを有効化し、基本オブジェクトのプロパティーが変更されるとテキスト・フィールドを無効化する必要があるとします。これを実現するには、次のように Enabled 条件を追加する必要があります。

```
<CheckBoxControl property="boolean3" label="Check box 3 with a much longer label than we had">
     <Layout rowIncrement="0" gridWidth="1"/>
</CheckBoxControl>
<TextBoxControl property="string3" label="String 3" showLabel="false">
     <Layout gridColumn="1" fill="horizontal" columnWeight="0.001"/>
          <Enabled>
               <Condition control="boolean3" op="equals" value="true"/>
          </Enabled>
</TextBoxControl>
```

これにより、チェック・ボックスに関連するブール値が真の場合にのみ、テキスト・フィールドが有効になります。

例 *:*複雑な条件によるコントロールの有効化*:* 複雑な条件のコーディングを理解するために、CLEF を使用して作成された一般化線型ノードのダイアログ・タブの 1 つを参照します。

ノード・ダイアログには「エキスパート」タブがあり、ユーザーにモデルの詳細な知識を提供するオプションが含まれています。タブのすべてのオプションは、最初は無効です。

「モード」チェック・ボックスを「エキスパート」に設定すると、次のようにさまざまなオプションが有効になります。

ただし、ダイアログの下部にある「反復」コントロールなど、無効なオプションもあります。このコントロールは、次の条件の両方が真の場合にのみ無効です。

- 「分布」が「正規分布」に設定されている
- 「リンク関数」が「恒等式」 に設定されている

この組み合わせは実際エキスパート・モードの「エキスパート」タブのデフォルト設定で、これらの選択情報のいずれかの設定を変更すると、「反復」を有効にします。

上記を実行するコードは、次のように 「反復」 ボタンの PropertiesSubPanel 宣言に含まれています。

```
<PropertiesSubPanel buttonLabel="Iterations..." buttonLabelKey= ...
     <Enabled>
          <And>
               <Condition control="mode" op="equals" value="Expert"/>
                    <Not>
                         <And>
                              <Condition control="distribution" op="equals" value="NORMAL"/>
                              <Condition control="link_function" op="equals" value="IDENTITY"/>
                         </And>
                    </Not>
          </And>
     </Enabled>
      ...
</PropertiesSubPanel>
```

外部の And セクションの Condition 要素は、変更が行われる前に「モード」が「エキスパート」に設定されるよう指定します。この条件が真の場合、Not セクションにより、内側の And セクションの条件がどちらも 満たされた場合のみ、ボタンが無効になることが指定されます。エキスパート・モードでは、「分布」または「リンク関数」のいずれかがデフォルト値以外に設定されている場合、「反復」が有効になります。

Visible 要素を使用した表示の特性の制御**:** 条件を使用して、特性の環境に応じてコントロールを表示または非表示にすることができます。Visible 要素によって行われます。

フォーマット

```
<Visible>
     <Condition .../>
     <And ... />
     <Or ... />
     <Not ... />
</Visible>
```

Condition 要素で、条件を指定してコントロールを表示するかどうかを決定するために検定します。

And、Or、および Not 要素を使用して、複合条件を指定できます。

詳しくは、トピック 73 ページの『条件』を参照してください。

例

以下の例では、source_language 条件が満たされている場合のみ、指定されたプロパティー・パネルが表示されます。

```
<PropertiesPanel>
     <Visible>
          <Condition control="source_language" op="equals" value="eng" />
     </Visible>
       ...
</PropertiesPanel>
```

カスタム出力ウィンドウ

モデル出力、ドキュメント出力、インタラクティブな出力オブジェクト (ノード以外) の場合、拡張によってデフォルト出力ウィンドウをカスタムウィンドウを完全に置き換えることはできません。これは、標準の java.awt.Frame クラスとして実装されています。.

カスタム ウィンドウを提供するには、次のように Java クラスを UserInterface 要素の frameClass 属性として指定します。

```
<DocumentOutput id="my_modelling_node" type="modelBuilder" ...>
     <Properties>
          <Property name="use_custom_type" valueType="boolean" .../>
           ...
     </Properties>
     <UserInterface frameClass="com.myextension.MyOutputFrame"/>
      ...
  </DocumentOutput>
```

指定されたクラスは、CLEF のクライアント側の API で定義された ExtensionObjectFrame インターフェースを実装する必要があります。これは次のようなウィンドウのライフ サイクルをカバーします。

- 基本の java.awt.Frame へのアクセス
- 出力オブジェクトおよびセッションへのアクセスなど、ウィンドウの初期化
- オブジェクトを保存または削除しようとする場合のウィンドウと基本オブジェクトの同期化
- ウィンドウの削除

詳しくは、トピック 180 ページの『クライアント側 API クラス』を参照してください。

第 7 章 ヘルプ・システムの追加

ヘルプ・システムの種類

CLEF 拡張子を作成する場合、通常オンラインの ヘルプ・システムを使用して、拡張子の使用方法を説明する必要があります。CLEF は、次のタイプのヘルプ・システムをサポートします。

- HTML Help
- JavaHelp

HTML Help

HTML Help は、Windows プラットフォーム上でのみ動作する、Microsoft が開発した独自の形式です。HTML Help システムは、`.chm` を持つ単一ファイルとして圧縮された形式にコンパイルされた `.htm` または `.html` ファイルで構成されています。IBM SPSS Modeler の独自のヘルプ・システムは、HTML Help 形式で提供されます。

HTML Help ファイルは、目次、インデックス、全文検索機能をサポートします (用語集をポップアップ・ウィンドウとして実装できます)。HTML エディターまたは商用のヘルプ・オーサリング・ツールを使用して、ソース `.htm` または `.html` トピック ファイルを作成できます。`.chm` ファイルを作成するには、Microsoft ダウンロード・センター Web サイトから無料ダウンロード可能な HTML Help Workshop を使用します (`.chm` ファイル作成の詳細は、HTML Help Workshop ヘルプ・システムを参照してください)。また、HTML Help 形式をサポートするヘルプ・オーサリング・ツールを使用して、`.chm` ファイルに使用されるトピック ファイルおよびグラフィック ファイルをコンパイルすることもできます。

JavaHelp

JavaHelp は、Java をサポートするすべてのプラットフォームで動作する、Sun Microsystems が開発したオープン ソースのヘルプ形式です。JavaHelp システムは、次のファイルで構成されています。

- ソース `.htm` または `.html` トピック・ファイル
- トピックで使用されるすべてのグラフィック ファイル
- ヘルプ・システムを制御するヘルプセット・ファイル (拡張子 `.hs`)
- トピック ID をトピック ファイルを関連付け、ヘルプ トピックを表示するウィンドウを定義するために使用する `map.xml` ファイル
- インデックス・エントリーを含む `index.xml` ファイル
- 目次のエントリーを含む `toc.xml` ファイル

JavaHelp は、目次、インデックス、全文検索、用語集機能をサポートします。HTML エディターまたは商用のヘルプ・オーサリング・ツールを使用して、ソース `.htm` または `.html` ファイルを作成できます。また、Sun Developer Network の Web サイトから無料ダウンロードできる JavaHelp ソフトウェアも必要です (詳細は、この Web サイトで入手可能な『*JavaHelp System User's Guide*』を参照してください)。

ヘルプ・システムの実装

この項では、設定ファイルのヘルプ・システムの関連するコンポーネントの定義方法について説明します。

ヘルプ・システムの場所および種類の定義

拡張子に使用されるヘルプ・システムの種類がある場合、拡張子に対する設定ファイルのリソース・セクションの HelpInfo の要素で定義されます。

フォーマット

```
<Resources>
 ...
     <HelpInfo id="name" type="help_type" path="help_path" helpset="helpset_loc"
      default="default_topicID" />
 ...
</Resources>
```

ここで、

id (必須) は、この拡張子に対するヘルプ情報の識別子です。

type (必須) は、ヘルプの種類を表し、次のいずれかを指定します。

- htmlhelp — path 属性で指定された .chm ファイルに含まれる HTML ヘルプ。
- javahelp — helpset 属性で指定されたヘルプセット・ファイル (.hs) を、ヘルプのソースと関連ファイルとともに使用する JavaHelp。

ヘルプの種類が htmlhelp である場合、次の追加の属性が必要です。

- path — ヘルプ・システムを含む .chm ファイルの場所 (仕様ファイルに対する相対位置) と名前。

ヘルプの種類が javahelp である場合、次の追加の属性が必要です。

- helpset — 使用する .hs ヘルプセット・ファイルの場所 (仕様ファイルに対する相対位置) と名前。
- default — 特定のタブに対するトピックが指定されていない場合に表示するデフォルトのトピックの識別子。

HelpInfo の要素が指定されていない場合、この拡張子に関連するヘルプはありません。

例

最初の例では、HTML Help の HelpInfo の要素を説明します。

```
<HelpInfo id="help" type="htmlhelp" path="help/mynode.chm" />
```

JavaHelp システムでは次のようになります。

```
<HelpInfo id="help" type="javahelp" helpset="help/mynode.hs"/>
```

JavaHelp の場合、関連ファイル (画像、マップ・ファイル、インデックス、コンテンツ ファイル) は、.hs ヘルプセット・ファイルと同じフォルダーに配置されている必要があります。

表示する特定のヘルプ トピックの指定

ユーザーがノードのダイアログ、特定のタブ、プロパティーのサブパネルでヘルプを起動する場合に表示する特定のヘルプ トピックを指定できます。ノード、タブ、またはプロパティーのサブパネルの指定の helpLink 属性によって実行されます。

helpLink 属性が指定されていない場合、ヘルプ・システムのデフォルト・トピックが、ユーザーがヘルプを起動する際に表示されます。

詳しくは、 48 ページの『ノード』、 116 ページの『タブ』、 129 ページの『プロパティー・サブパネル』で、helpLink 属性の説明を参照してください。

例

この例では、HTML Help を使用していると仮定し、ユーザーがヘルプを選択する場合にフォーカスがあるウィンドウによってさまざまなコンテキスト・トピックを表示できる方法を示します。

```
<Resources>
      ...
     <HelpInfo id="help" type="htmlhelp" path="help/mynode.chm"/>
      ...
</Resources>
 ...
<Node id="mynode" scriptName="my_node" type="dataTransformer" palette="recordOp"
  label="Sorter" description="Sorts a data file" >
  ...
     <Tabs defaultTab="1">
          <Tab label="Basic Controls" labelKey="basicControlsTab.LABEL"
            helpLink="basic_controls.htm">
               <PropertiesPanel>
                ...
                    <PropertiesSubPanel buttonLabel="Additional settings..."
                      buttonLabelKey="AdditionalOptions.LABEL" dialogTitle="Additional
                      Settings" dialogTitleKey="AdditionalOptionsDialog.LABEL" helpLink=
                      "addsettingsdlg.htm">
  ...
          </Tab>
          <Tab label="Selector Controls" labelKey="selectorControlsTab.LABEL"
           helpLink="selector_controls.htm">
  ...
          </Tab>
  ...
</Node>
```

この場合、「基本コントロール」タブにフォーカスが置かれている状態でユーザーがヘルプを選択すると、mynode.chm ヘルプ・ファイルの basic_controls.htm のトピックが表示されます。その後ユーザーが 「追加設定」 ボタンをクリックして「追加設定」ダイアログを開き、そのダイアログの 「ヘルプ」 を選択すると、addsettingsdlg.htm のトピックが表示されます。ユーザーが「追加設定」ダイアログを閉じる場合、「選択コントロール」タブを選択して 「ヘルプ」 を再度選択すると、selector_controls.htm のトピックが表示されます。

JavaHelp の場合、helpLink 属性の値は、map.xml ファイルの target 属性の値と一致する必要があります。例えば、map.xml ファイルに

```
<map version="1.0">
 ...
     <mapID target="basic_controls" url="basic_controls.htm"/>
 ...
</map>
```

が含まれる場合、対応する helpLink 属性に次の値を指定する必要があります。

```
helpLink="basic_controls"
```

これは、JavaHelp が呼び出される場合、target 属性の値を読み込み、関連する url 値にマップして表示する適切なファイルを検出するためです。

第 8 章 ローカライゼーションとアクセシビリティー

はじめに

ローカライゼーションは、特定のロケールのソフトウェア、ヘルプ、マニュアルを適合させるプロセスを参照しています。これには、ユーザー・インターフェース、ヘルプおよびマニュアルの翻訳および適切なロケールのシステムの検定が含まれます。自身の地域以外の地域に拡張子を配布する場合、その拡張子のローカライズされたバージョンを配布することができます。

ここでは、アクセシビリティーはユーザー・インターフェースの機能追加を参照し、視覚障害または手先の動作の制限など、特定の障害を持つユーザーにとってシステムへのアクセスが容易にできるようにします。

ローカライゼーション

IBM SPSS Modeler は、世界の多くの地域にローカライズされています。サポートされたいずれかの言語について、ユーザーは Windows の地域オプションを自身のロケールに設定すると、例えば次のような標準 IBM SPSS Modeler UI コンポーネントが該当する言語で表示されます。

- システム メニューとメニュー・エントリー
- システム ボタン (生成、OK、実行、キャンセル、適用、リセット)
- 標準的なダイアログ・ボックスのタブ (使用される場合、注釈およびデバッグ)
- エラーおよびシステム メッセージ (「このオブジェクトは保存されていません。」など)

拡張子がこれらの標準 IBM SPSS Modeler コンポーネントを使用している場合、選択した言語がサポートされていればその言語で自動的に表示されます。

拡張子の他のコンポーネントに対し、CLEF ではローカライズを支援する機能を提供しています。ローカライズできるのは次のとおりです。

- ノード名 (パレットおよびキャンバス上)
- モデル名 (マネージャ領域の「モデル」タブ)
- ドキュメント名 (マネージャ領域の「出力」タブ)
- アクションに関連するアイコン・イメージの場所
- ツールヒント テキスト
- ヘルプ・システム
- ノード・ダイアログ
  - タイトル・バー・テキスト
  - カスタム・メニューとメニュー項目
  - フィールド、プロパティー、ボタン、タブのラベル
  - 静的テキスト
- システムおよびエラー・メッセージ

テキスト文字列は、項目を翻訳した場合により長いテキストを使用できるよう、短くする必要があります。

システム メッセージおよびエラー・メッセージは、設定ファイル、プロパティー・ファイルおよびサーバー側 API を組み合わせてローカライズできます。 詳しくは、トピック 198 ページの『状況詳細ドキュメント』を参照してください。

プロパティー・ファイル

ローカライズできるテキスト文字列はプロパティー・ファイルと呼ばれるファイルにテキスト文字列として保存されます。ローカライゼーションのリソース・バンドルを保存するために標準 Java 形式を使用します。それぞれのプロパティー・ファイルは、一連のレコード、拡張子のローカライズされた各項目のレコードで構成されています。各レコードのフィールドは、設定ファイルの `labelKey` 属性に対応しており、CLEF はプロパティー・ファイルから対応するテキスト文字列を読み込み、適切な場所に表示することができます。

プロパティー・ファイルは拡張子 `.properties` を持つ必要があり、関連するノードの設定ファイルと同じディレクトリーにある必要があります。IBM SPSS Modeler は最初に次の名前を持つデフォルトのプロパティー・ファイルを検索します。

path`.properties`

path は、プロパティー・バンドルを定義する (Resources セクション内の) `Bundle` 要素の `path` 属性の値です。以下に例を示します。

```
<Bundle id="bundle" path="my_resources"/>
```

デフォルトのプロパティー・ファイルがない場合、IBM SPSS Modeler は設定ファイルの定義からテキスト文字列を読み込みます。

ローカライゼーションにサポートされた各言語のプロパティー・ファイルが必要です。デフォルト以外の言語のファイルは、ファイル名の接尾辞で区別されます。以下に例を示します。

```
my_resources.properties
my_resources_de.properties
my_resources_fr.properties
```

接尾辞は、言語コードの 2 つの文字の ISO 639-1 規格に従います。

プロパティー・ファイルの各レコードには、次の形式があります。

id=text_string

ここで、

id は、仕様ファイルの各属性 (`buttonLabelKey`、`descriptionKey`、`dialogTitleKey`、`falseLabelKey`、`imagePathKey`、`labelKey`、`messageKey`、`textKey`、`trueLabelKey`) の識別子です。設定ファイルがどのように表示されるかによって、任意の接尾辞を使用したり、まったく使用しない場合がありますが、この識別子には通常、接尾辞 `.LABEL` があり、容易に区別することができます。

text_string は、項目のテキストです。

例:「ダイアログ」タブのローカライズ

ノード・ダイアログのローカライズされたタブの例では、次の場所にある 2つのプロパティー・ファイル、デフォルト (英語) バージョンとフランス語のバージョンを使用します。

extension_folder¥**my_resources**.properties
extension_folder¥**my_resources**_fr.properties

この場合、*extension_folder* は、設定ファイルを含むフォルダーです。

設定ファイルで、プロパティー・ファイルは Resources セクションの Bundle 要素によって参照されます。

```
<Resources>
     <Bundle id="bundle" type="properties" path="my_resources"/>
</Resources>
```

path 属性には、言語拡張子または .properties 接尾辞を含むことはできません。

設定ファイルのその他の関連部分は次のとおりです。

```
<Node id="uiTest" scriptName="ui_test" type="dataTransformer" palette="recordOp" label=
"UI Test" ... >
     <Properties>
          <Property name="enum1" valueType="enum" defaultValue="value4">
               <Enumeration>
                    <Enum value="value1" label="Value 1.1" labelKey="enum1.value1.LABEL"/>
                    <Enum value="value2" label="Value 1.2" labelKey="enum1.value2.LABEL"/>
                    <Enum value="value3" label="Value 1.3" labelKey="enum1.value3.LABEL"/>
                    <Enum value="value4" label="Value 1.4" labelKey="enum1.value4.LABEL"/>
                    <Enum value="value5" label="Value 1.5" labelKey="enum1.value5.LABEL"/>
               </Enumeration>
          </Property>
     </Properties>
 ...
     <UserInterface ... >
          <Tabs defaultTab="1">
               <Tab label="Basic Controls" labelKey="basicControlsTab.LABEL" ... >
      ...
     </UserInterface>
 ...
</Node>
```

プロパティー・ファイルにおいて、プロパティー・ファイルの英語バージョンには、次のレコードが含まれています。

```
basicControlsTab.LABEL=Basic Controls
enum1.value1.LABEL=Value 1.1
enum1.value2.LABEL=Value 1.2
enum1.value3.LABEL=Value 1.3
enum1.value4.LABEL=Value 1.4
enum1.value5.LABEL=Value 1.5
```

以下の図では、これらのレコードの影響を受けるダイアログの箇所を強調表示しています。

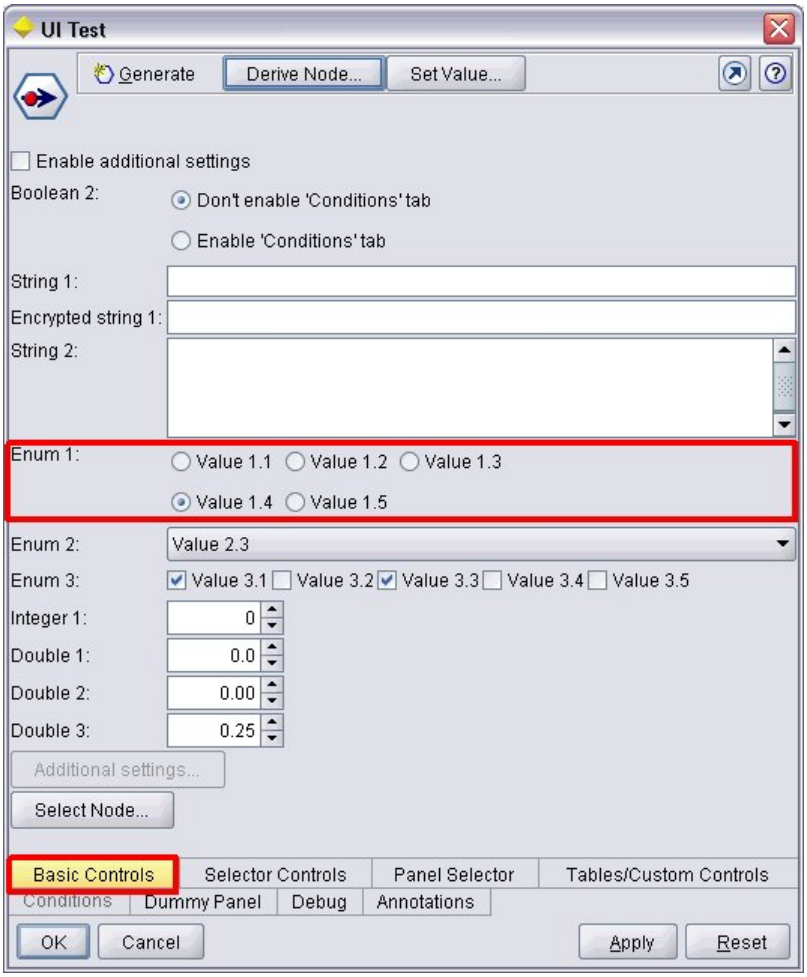

図 *93.* 「非ローカライズ」タブ

プロパティー・ファイルのフランス語バージョン (my_resources_fr.properties) 内の対応するセクションは以下のとおりです。

```
basicControlsTab.LABEL=Contrôles de Base
enum1.value1.LABEL=Valeur 1,1
enum1.value2.LABEL=Valeur 1,2
enum1.value3.LABEL=Valeur 1,3
enum1.value4.LABEL=Valeur 1,4
enum1.value5.LABEL=Valeur 1,5
```

これらのレコードにより、以下の図に示すように、画面の関連する箇所に、翻訳されたテキストが表示されます。

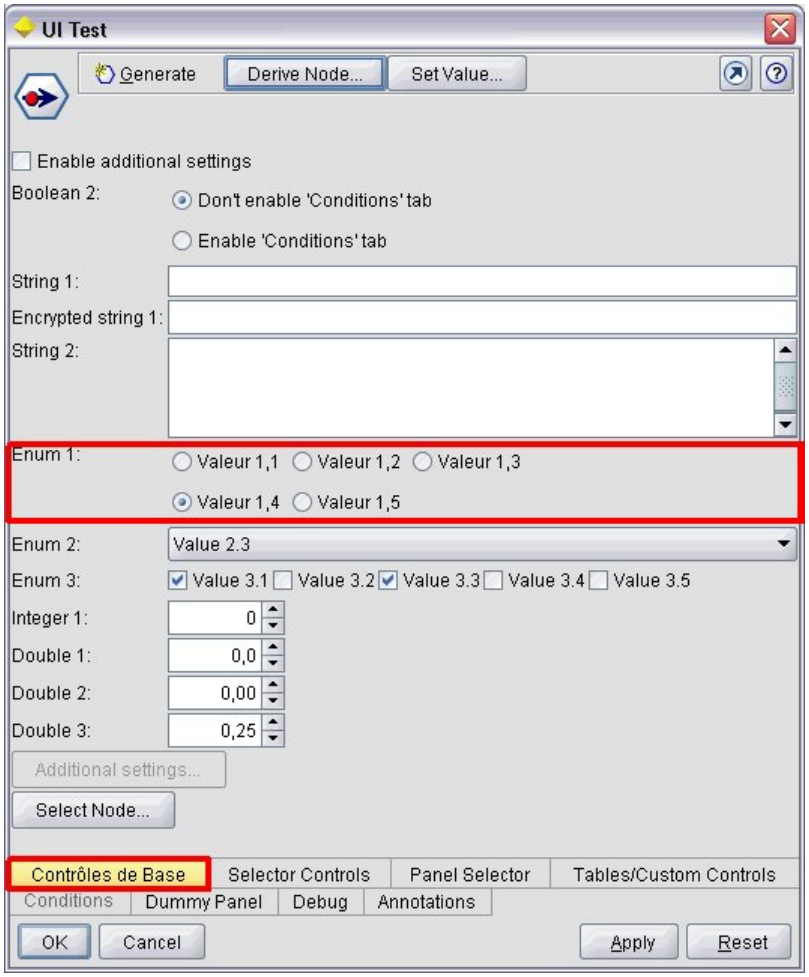

図 *94.* 「ローカライズ」タブ

画面の下部の 4 つのボタンのローカライゼーションは、CLEF ではなく標準 IBM SPSS Modeler 機能によって操作されます。

例:特殊文字の使用

プロパティー・ファイルでは、標準 ASCII テキスト文字列の特殊文字に Unicode 拡張シーケンスを使用する必要があります。次は、フランス語にローカライズされたプロパティー・ファイルの一部です。

```
Genlinnode.LABEL=Lin¥u00e9aire g¥u00e9n¥u00e9ralis¥u00e9

Fields.LABEL=Champs
Model.LABEL=Mod¥u00e8le
Expert.LABEL=Expert

inputFields.LABEL=Entr¥u00e9es
targetField.LABEL=Cible
 ...
```

日本語または中国語などラテン文字以外の文字を使用する言語の場合、テキスト文字列に Unicode 拡張シーケンスを使用する必要があります。次は、日本語にローカライズされたレコードの同じセットです。

```
Genlinnode.LABEL=¥u4e00¥u822c¥u5316¥u7dda¥u578b

Fields.LABEL=¥u30d5¥u30a3¥u30fc¥u30eb¥u30c9
```

```
Model.LABEL=¥u30e2¥u30c7¥u30eb
Expert.LABEL=¥u30a8¥u30ad¥u30b9¥u30d1¥u30fc¥u30c8

inputFields.LABEL=¥u5165¥u529b
targetField.LABEL=¥u5bfe¥u8c61
 ...
```

ヘルプ・ファイル

ヘルプ・システムを持つ拡張子をローカライズする場合、ヘルプ・システムのローカライズされたバージョンも提供する必要があります。それぞれのローカライズされた拡張子に対してローカライズされたヘルプ・システムを提供します。

HTML Help のローカライズ

ローカライズしている拡張子が HTML Help ファイル (接尾辞 .chm) を使用する場合、デフォルトの .chm ファイルをローカライズされたバージョンと置き換えることができます。HTML Help システムの詳細は、 167 ページの『HTML Help』を参照してください。

ローカライズされた .chm ファイルを作成する手順は、次のとおりです。

1. ヘルプ・システムを構成し、同じファイル名を持つ個々のヘルプ トピック (.htm または .html) ファイルの翻訳されたバージョンを作成します。
2. 必要であれば、ヘルプ・システムに含まれるグラフィック (スクリーン ショットなど) のローカライズされたバージョンを使用します。
3. Microsoft の HTML Help Workshop または他のヘルプ・オーサリング・ツールを使用して、ファイルをローカライズされた .chm ファイルにコンパイルします。
4. ローカライズされたノードを含むヘルプ・システムを検定します。 詳しくは、トピック『ローカライズされた CLEF ノードの検定』を参照してください。

JavaHelp のローカライズ

ローカライズしている拡張子が JavaHelp システムを使用する場合、サポートされた各言語のヘルプ ソース・ファイルのローカライズされたバージョンを提供する必要があります。JavaHelp では、存在する場合はローカライズされた適切なバージョンを表示できます。 詳しくは、トピック 167 ページの『JavaHelp』を参照してください。

ローカライズされた JavaHelp システムを作成する手順は、次のとおりです。

1. ヘルプ・システムを構成し、同じファイル名を持つ個々のヘルプ トピック (.htm または .html) ファイルの翻訳されたバージョンを作成します。
2. 必要であれば、ヘルプ・システムに含まれるグラフィック (スクリーン ショットなど) のローカライズされたバージョンを使用します。
3. ヘルプセットおよびその他の必要なファイル (マップ・ファイル、コンテンツ、インデックス・ファイル) を生成します。
4. ローカライズされたノードを含むヘルプ・システムを検定します。 詳しくは、トピック『ローカライズされた CLEF ノードの検定』を参照してください。

ローカライズされた CLEF ノードの検定

ローカライズされたノードおよびヘルプ・システムを検定する手順は次のとおりです。

1. ローカライズされたノードの設定ファイルの Resources セクションで、HelpInfo 要素の path 属性を変更し、ローカライズされた .chm または .hs ファイルを参照します。例えば HTML Help の場合、次のように使用します。

```
<Resources>
 ...
    <HelpInfo id="help" type="HTMLHelp" path="help/mynode_fr.chm "/>
</Resources>
```

   JavaHelp の場合は、次のように使用します。

```
<Resources>
 ...
    <HelpInfo id="help" type="javahelp" helpset="help/mynode_fr.hs "/>
</Resources>
```

2. ローカライズされた .chm または .jar ファイルを path 属性に示された場所にコピーします。
3. 必要なロケールの Windows の地域を設定します。

   「コントロール パネル」 > 「地域と言語のオプション」 > 「地域オプション」 > 「標準と形式」 > **<language>**
4. IBM SPSS Modeler を起動し、該当する言語で表示されるようにします。
5. ローカライズされたノードを IBM SPSS Modeler に追加します。 詳しくは、トピック 205 ページの『CLEF 拡張のテスト』を参照してください。
6. ノードのコピーを領域に配置します。

   ノードのダイアログを開き、該当する言語で正しく表示されていることを確認します。
7. ダイアログの「ヘルプ」ボタンをクリックし、適切なヘルプ トピックが該当する言語で表示されていることを確認します。

アクセシビリティー

CLEF ノードでは、 マウス操作に代わるキーボード操作、画面読み上げソフトウェアのサポートなど、すべての標準的な IBM SPSS Modeler アクセシビリティーを利用できます。

さらに、CLEF ノードでアクセスの目的でカスタマイズされたツールヒントのテキストを提供できます。

キーボードの組み合わせを指定して、エンド・ユーザーに CLEF に追加されたさまざまなユーザー・インターフェース機能への代替アクセスを提供することもできます。 詳しくは、トピック 117 ページの『アクセス キーとキーボード・ショートカット』を参照してください。

操作ボタン、およびコントローラとして分類されるそれぞれの画面コンポーネント (チェック・ボックスまたはラジオ・ボタンのグループ) について、次のように定義できます。

- label
- description

ラベルは、コンポーネントの名前として画面上に表示され、画面読み上げソフトウェアによって読み上げられる表示テキストです。視覚障害を持つユーザーの場合は、ラベルの表示フォント・サイズを以下から取得できる「ユーザー オプション」ダイアログの「表示」タブのコントロールによって変更できます。

「ツール」 > 「ユーザー オプション」

説明は、マウス・ポインタがコンポーネント上に移動した場合、表示されるツールヒントのテキストです。ツールヒントでは、名前だけで判断できないコンポーネントの詳細が提供されます。ツールヒントは、読み上げるように設定された画面読み上げソフトウェアによって読み上げられます。

ラベルおよび説明は、設定ファイルのコンポーネントを定義する label および description 属性によって定義されます。ラベルおよび説明のどちらも labelKey および descriptionKey 属性によってそれぞれローカライズできます。

例

この操作ボタンの例では、ラベルおよび説明機能のどちらの使用についても説明しています。

```
<Action id="setValue" label="Set Value..." labelKey="setValue.LABEL"
     description="Sets a value" descriptionKey="setValue.TOOLTIP"/>
```

第 9 章 プログラミング

CLEF ノードのプログラミングについて

設定ファイルで定義できない処理を実行するノードを有効にするために、CLEF はプログラムで呼び出せる次のアプリケーション・プログラミング・インターフェース (APIs) を提供します。

- クライアント側 **API:** 追加のコントロール、ユーザー・インターフェース・コンポーネント、または設定ファイルで直接提供されない双方向性を必要とする拡張子で使用できる Java API です。
- 予測サーバー **API (PSAPI):** データ・マイニングおよび予想分析機能を必要とするアプリケーションで使用するための IBM SPSS Modeler 機能を公開する Java API です。PSAPI および IBM SPSS Modeler データ・マイニング・ワークベンチは同じ基本テクノロジーを共有しています。
- サーバー側 **API:** 設定および実行設定の取得、これらの設定の持続、実行フィードバック、ジョブ制御(実行の中止など)、SQL 生成、返されたオブジェクトなどの側面をカバーする Cベースの APIです。

CLEF API ドキュメンテーション

次のセクションではクライアント側およびサーバー側の API の概要について説明します。さらに完全な API ドキュメントは IBM SPSS Modeler インストールにおける zip ファイルで含まれており、使用する前に抽出する必要があります。

API ドキュメントの抽出

1. 製品のダウンロードで *clef_apidoc.zip* ファイルを見つけます。
2. WinZip または同様のツールを用いて、zip ファイルの内容を適切なディレクトリーへ抽出します。それによって、すべての APIドキュメントを含んだディレクトリー内の *clef_apidoc* サブフォルダーが作成されます。

API ドキュメントの表示

1. *clef_apidoc* サブフォルダーへ移動して *clef_apidoc.htm* ファイルを開きます。
2. 必要に応じて PSAPI のクライアント側またはサーバー側を選択します。

クライアント側 API

CLEF はクライアント側の処理に使用できる方法を含めた多数の Java クラスを用意しています。例えば `DataModelProvider` クラスは、入力データ・モデルへの変更が複雑すぎて設定ファイルで提供される機能を使用できない場合、出力データの計算を行えるようにします。

クライアント側 API クラス

クライアント側のクラスは次のとおりです。

表 40. クライアント側 *API* クラス

クラス	説明
NodeDelegate	ノード デリゲートでサポートされるメソッドを定義します。ExtensionProcessor のインスタンスごとに、1 つのノード デリゲート インスタンスが作成されます。実装クラスのパスは、extension.xml ファイルにある、関連する Node 要素の「delegate」属性で指定されます。
OutputDelegate	出力デリゲートでサポートされるメソッドを定義します。ExtensionOutput のインスタンスごとに、1 つの出力デリゲート インスタンスが作成されます。実装クラスのパスは、extension.xml ファイルにある、関連する DocumentOutput 要素または ModelOutput 要素の「delegate」属性で指定されます。
ExtensionObjectUIDelegate	拡張オブジェクト UI デリゲートでサポートされるメソッドを定義します。拡張ノード ダイアログまたは出力ウィンドウのインスタンスごとに、1 つの UI デリゲート インスタンスが作成されます。実装クラスのパスは、extension.xml ファイルにある、関連する UserInterface 要素の「uiDelegate」属性で指定されます。
ExtensionDelegate	拡張デリゲートでサポートされるメソッドを定義します。1 つのプロセス内で共有される拡張機能ごとに、1 つの拡張デリゲート インスタンスが作成されます。単一のプロセスがさまざまなロケールのセッションをサポートしている可能性があるため、拡張デリゲートで使用可能なロケール情報はありません。実装クラスのパスは、extension.xml ファイルにある CommonObjects 要素の「extensionDelegate」属性で指定されます。
ActionHandler	拡張子でメニュー・オプションやツールバー・ボタンからユーザーがリクエストしたアクションを管理できます。
DataModelProvider	ノードで Java を使用してデータ・モデルへの複雑な変更を行い、出力データ・モデルを計算することができます。
ExtensionObjectFrame	モデルやドキュメント出力の表示に使用するウィンドウに関連した機能を定義します。
ExtensionObjectPanel	拡張オブジェクト・パネルに関連した機能を定義します。
PropertyControl	プロパティー・パネルのユーザー設定のプロパティー・コントロールに関連した機能を定義します。

これらのクラスの詳細はクライアント側 API ドキュメントに記載されています。 詳しくは、トピック 179 ページの『CLEF API ドキュメンテーション』を参照してください。

クライアント側 API の使用方法

CLEF ノードにクライアント側関数の呼び出しを含むには、

1. 関数呼び出しを含む *.java* ソース・ファイルを作成します。
2. ソース・ファイルを *.class* ファイルへコンパイルします。
3. 複数の *.class* ファイルを 1 つの *.jar* ファイルに結合し、その *.jar* ファイルへの参照を仕様ファイルで指定することができます。以下に例を示します。

```
<Resources>
 ...
     <JarFile id="selfjar" path="selflearning.jar"/>
 ...
</Resources>
```

CLEF 要素の中にはクラスを明示的に参照できるものもあります。例えば次に示すように、クラス参照を設定ファイルの PropertyControl 要素の controlClass 属性に含めることができます。

```
<PropertyControl property="target_field_values_specify" ...
     controlClass="com.spss.clef.selflearning.propertycontrols.list.CustomListControl" ... />
```

CustomListControl はプロパティー・コントロールを実装するクラスの名前、com.spss.clef.selflearning.propertycontrols.list は JarFile 要素で宣言した *.jar* ファイル内のクラスへのパスを示します。

詳しくは、トピック 34 ページの『Resources セクション』を参照してください。

このリリースで供給されているノード例のソース・コードを見ると便利です。 詳しくは、トピック 29 ページの『ソース・コードの検査』を参照してください。

予測サーバー API (PSAPI)

PSAPI は基本となる予測サーバー技術へのプログラミング インターフェースを提供します。PSAPI の主な要素は Java インターフェースとして指定されています。これらインターフェースの大半が PSAPI の提供する内部クラスを通じて実装されますが、それは PSAPI 特性の一部を形成するものではありません。このアプローチは、予測サーバー技術になされた変更から PSAPI ユーザーを保護することを目的としています（アーキテクチャー変更、プライベート・クライアントおよびサーバー・プロトコルの変更など）。

これらのクラスの詳細は PSAPI ドキュメントに記載されています。 詳しくは、トピック 179 ページの『CLEF API ドキュメンテーション』を参照してください。

サーバー側 API

サーバー側 API はC 言語 API として定義されていますが、C++ の実装に対応します。拡張モジュールの開発者は、C または C++ でプログラミングすることにより、言語 API を無視して直接プログラミングすることを選択できます。開発者が C API へバインドする方法を持っていれば、他の言語も使用できます。CLEF はまた、一部の C API へのラッパー機能を果たす複数の C++ ヘルパー・ソース・ファイルを提供します。

アーキテクチャー

クライアントの拡張 **node** はサーバーの拡張 **peer** で補完されます。ピアは、サーバーがホストする共有ライブラリーとして実装される、拡張モジュールによって定義されます。ノードとそのピア間のコミュニケーションは、サーバーが管理する拡張リソースによってメディエートされます。ピアがホストから情報がサービスをリクエストするためにコールバック関数を使用する一方で、リソースはそのピアを作成および操作するための拡張モジュールで定義されるサービス関数を呼び出します。

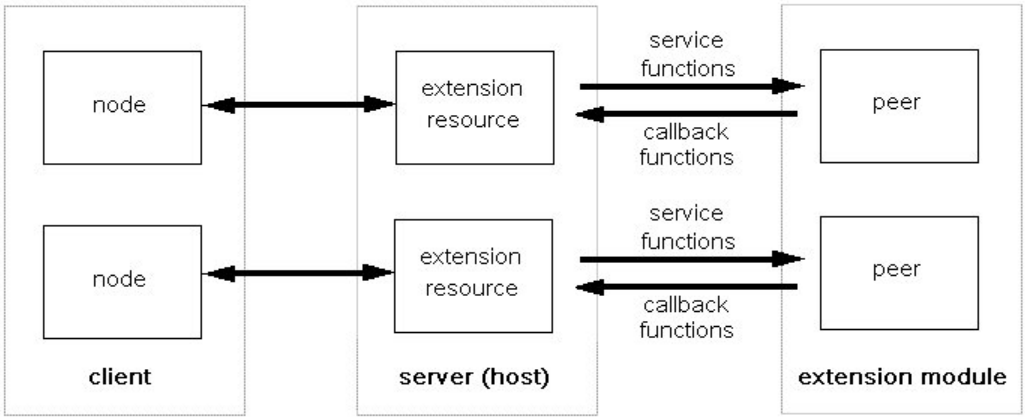

図 *95. CLEF API* アーキテクチャー

サービス関数

サービス関数は拡張モジュールによって実装されます。拡張モジュールは必要とされるすべての関数を実装する必要があり、またそれ以外の一部またはすべてを実装する場合もあります。

サービス関数には 2 種類あります。

- モジュール関数
- ピア関数

次のセクションでは、サービス関数の概要について説明します。これらの関数に関する詳細は、次のようにサーバー側 API ドキュメントに記載されています。

1. CLEF API ドキュメンテーション画面より、「サーバー側 **API** の概要」 を選択します。
2. 「モジュール」 タブをクリックします。
3. 「拡張モジュールで実装する **API** サービス関数」 を選択します。

CLEF API ドキュメントへのアクセスに関する情報は、 179 ページの『CLEF API ドキュメンテーション』を参照してください。

モジュール関数

モジュール関数はシングル・スレッドから呼び出されます。

表 *41.* モジュール関数

関数	必要ですか**?:**	説明
`clemext_initialise`	はい	拡張モジュールとして初期化します。
`clemext_cleanup`	はい	拡張モジュールで割り当てられたリソースの処理
`clemext_getModuleInformation`	いいえ	拡張モジュールに関する情報を取得します
`clemext_create_peer`	はい	拡張ノードのピア・インスタンスを作成します
`clemext_destroy_peer`	はい	ピア・インスタンスの処理

ピア関数

ピア関数は clemext_create_peer への前の呼び出しで戻されたピア・インスタンス ハンドルに適用されます。ピア関数は、ピア ハンドルが異なる場合のみ、別のスレッドから一斉に呼び出される場合があります。これに対する例外は、clemext_peer_cancelExecution 関数（定義される場合）が、別のスレッドでの長期間実行を中断するため常時どのスレッドからも呼び出される可能性があることです。

表 42. ピア関数

関数	必要ですか**?:**	説明
clemext_peer_configure	はい	ピア・インスタンスにそのパラメーターおよびデータ・モデルを処理するよう指示します
clemext_peer_getDataModel	はい	ピア・インスタンスから出力データ・モデルを取得します
clemext_peer_getCatalogueInformation	いいえ	モジュールからカタログ情報を取得します
clemext_peer_getExecutionRequirements	いいえ	ピア・インスタンスの実行要件を取得します
clemext_peer_beginExecution	はい	ピア・インスタンスの実行を開始します
clemext_peer_nextRecord	はい	ピアの結果セットで次のレコードに移動します
clemext_peer_getRecordValue	はい	最後にフェッチした結果レコード内の特定フィールドの値を戻します
clemext_peer_endExecution	はい	ピア・インスタンスに実行が完了したことを示します
clemext_peer_cancelExecution	いいえ	長時間の beginExecution または nextRecord の関数呼び出しを中断するための別のスレッドからの呼び出し
clemext_peer_getSQLGeneration	いいえ	ピア・インスタンスから SQL を生成します
clemext_peer_getErrorDetail	はい	最後のピアにおけるモジュール固有のエラーに関する補足情報を取得します

以下の表に示すピア関数は、インタラクティブ・モデル・ビルダーと併用するように設計されています。

表 43. インタラクティブ・モデル・ビルダーと併用するためのピア関数

関数	必要ですか**?:**	説明
clemext_peer_beginInteraction	いいえ	ピア・インスタンスとのインタラクションを開始します
clemext_peer_request	いいえ	ピアにおけるインタラクティブ・リクエストを実行します
clemext_peer_getRequestReply	いいえ	最後に正常実行されたリクエストからの応答を取得します
clemext_peer_endInteraction	いいえ	ピア・インスタンスにインタラクションが完了したことを示します

コールバック関数

拡張モジュールがホスト プロセスから情報またはサービスを要求するときは、コールバックを通じて行う必要があります。コールバックは、リクエストのターゲットを識別するポインターであるハンドルに適用されます。

コールバックは、呼び出しを指示する IBM SPSS Modeler オブジェクトのハンドルで渡すことにより起動します。ハンドルは、サービス関数のパラメーターとして拡張モジュールへ渡されます。

コールバック関数に失敗すると、関連したモジュール固有のエラー・コードで詳細を返す必要があります（CLEMEXTErrorCode に記載）。モジュールは代わりに、ホストが確認できるようコールバック エラーを戻しこの詳細を渡すことによって、これを管理できる場合があります。

次の種類の入力を利用することができます。

- ホスト関数
- ノード関数
- 反復関数
- 進行関数
- チャンネル関数（インタラクティブ・モデル専用）

次のセクションでは、コールバック関数の概要について説明します。これらの関数に関する詳細は、次のようにサーバー側 API ドキュメントに記載されています。

1. CLEF API ドキュメンテーション画面より、「サーバー側 **API** の概要」 を選択します。
2. 「モジュール」 タブをクリックします。
3. 「一般コールバック」 を選択します。

CLEF API ドキュメントへのアクセスに関する情報は、 179 ページの『CLEF API ドキュメンテーション』を参照してください。

ホスト関数

ホスト関数は clemext_initialise から渡されるホスト ハンドルで定義されます。

表 *44.* ホスト関数

関数	説明
clemext_host_getHostInformation	ホスト環境に関する統計情報を取得します

ノード関数

ノード関数は、clemext_create_peer から渡されるノード・ハンドルによって定義されます。

表 *45.* ノード関数

関数	説明
clemext_node_getNodeInformation	ノードに関する統計情報を取得します
clemext_node_getParameters	ノードからパラメーターを取得します
clemext_node_getDataModel	ノードの入力データ・モデルを取得します
clemext_node_getOutputDataModel	ノードの出力データ・モデルを取得します
clemext_node_getSQLGeneration	ノードの上流 SQL 生成情報を取得します
clemext_node_getPassword	パスワード識別子の平文を取得します
clemext_node_getFilePath	実行中にクライアントとサーバーの間で交換されるファイルのパスを取得します

反復関数

反復関数は、clemext_peer_beginExecution から渡される反復ハンドルによって定義されます。反復は入力データ・セットを拡張モジュールへ公開します

表 46. 反復関数

関数	説明
clemext_iterator_nextRecord	入力データ・セットの次のレコードへ移動します
clemext_iterator_getRecordValue	最後にフェッチした入力レコード内の特定フィールドの値を戻します
clemext_iterator_rewind	nextRecord への次の呼び出しがセットの最初のレコードに移動するよう、入力データ・セットのステートを復元します

進行関数

進行関数は、clemext_peer_beginExecution から渡される進行ハンドルによって定義されます。

表 47. 進行関数

関数	説明
clemext_progress_report	進行状況をホストへ報告します

チャンネル関数

チャンネル関数はインタラクティブ・モデルのみで使用し、clemext_peer_beginInteraction から渡されるチャンネル・ハンドルで定義されます。

表 48. チャンネル関数

関数	説明
clemext_channel_send	クライアントへ非同期メッセージを送り、ピア スレッドが進行状況を報告できるようにします。

プロセス・フロー

拡張モジュールは多数のサービスおよびコールバック関数を呼び出して、その処理を実行します。呼び出される実際の関数は、モジュールが実行するために必要とする処理によって異なります。

例

一般的なモジュール実行のシーケンス・ダイアグラムを次の図に示します。

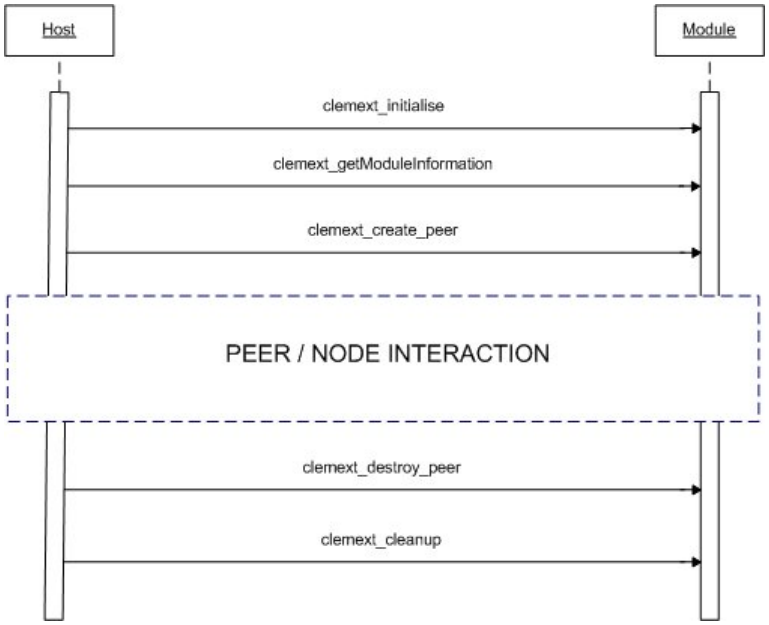

図 *96.* 一般的なプロセス・フロー

ピアとノードの相互作用のブロックを以下の図に示します。

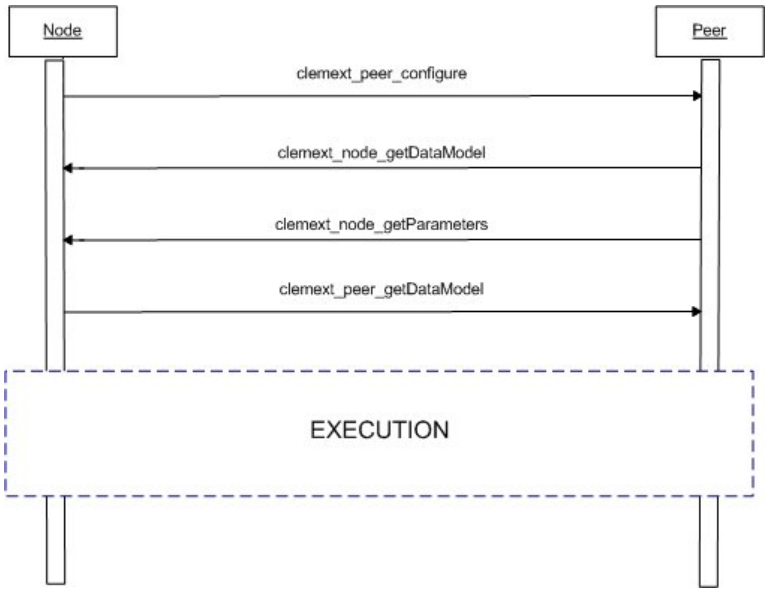

図 *97.* 一般的なピアとノードの相互作用のブロック

一般的な実行ブロックを以下の図に示します。

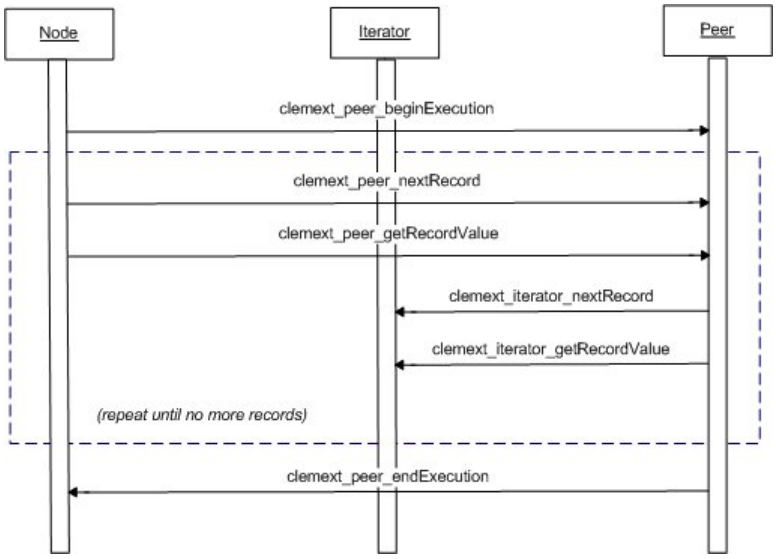

図 *98.* 一般的な実行ブロック

注:

- モジュールは、サーバー開始時に IBM SPSS Modeler サーバー・プロセスにロード、またはそのサービスが最初に要求されるときはオンデマンドで後にロードされる場合があります。
- モジュールがロードされると、ホストは サービス関数 `clemext_initialise` を一度起動します。
- モジュールが起動および初期化されると、ホストはサービス関数 `clemext_getModuleInformation` を使用してモジュールをクエリーする場合があります。
- モジュールがロードされた後、そのサービスはモジュールが提供するピア・オブジェクトによって起動されます。モジュール内で、ピア・オブジェクトは、ホストのノード・オブジェクトの対としてサービス関数 `clemext_create_peer` によって作成され、ホスト・アプリケーションによる指示に従ってタスクの実行を管理します。同じタイプの複数ピア・オブジェクトが存在し、それを一度のプロセスで同時に実行することが可能です。
- ピア・オブジェクトが作成されると、サービス関数 `clemext_peer_configure` で設定できます。
- この時点で、`clemext_node_getDataModel` や `clemext_node_getParameters` など、クライアントから情報を得るためにコールバック関数を実行することがあります。
- IBM SPSS Modelerは `clemext_peer_getDataModel` サービス関数を用いて、ピア・インスタンスから出力データ・モデルを取得します。
- ピア・インスタンスの実行は `clemext_peer_beginExecution` サービス関数で開始します。
- `clemext_peer_nextRecord` サービス関数は、ピアの結果セットにおいてフォーカスを次のレコードに移動します（あるいは関数が初めて呼び出される場合は、最初のレコードに移動）。この後に現在のレコード内で特定フィールドの値を返す `clemext_peer_getRecordValue` サービス関数が続きます。
- 反復コールバック関数 `clemext_iterator_nextRecord` および `clemext_iterator_getRecordValue` は CLEF モジュールで呼び出し、入力レコードで並べて特定のフィールド値を返すことができます。
- ピア・インスタンスの実行は `clemext_peer_endExecution` サービス関数で終了します。
- ピア・インスタンスは `clemext_destroy_peer` を呼び出すと削除されます。
- モジュールがアンロードされる前に、ホストはサービス関数 `clemext_cleanup` を起動します。
- モジュールはサーバー・プロセスがシャットダウンするとき、またはそれより早くサービスがもう必要なくなったときにアンロードされる場合があります。

サーバー側 API の機能

このセクションではサーバー側 API のいくつかの機能について取り上げます。

- ノード・タイプの情報
- 異なるタイプのデータ・ストレージを表すデータ・タイプ
- サーバー側共有ライブラリー
- ファイルスペースおよび一時ファイル
- データベースでSQL 指示を実行する SQL プッシュバック
- IBM SPSS Modeler および拡張間のデータ・モデル情報の交換
- 出力ドキュメント
- C++ ヘルパー

ノード・タイプ

設定ファイルにおいて、ノード定義は次の形式をとります。

```
<Node id="identifier" type="node_type" .../>
```

id 属性は、ノードを一意的に識別する文字列です。

type 属性は次のタイプの一つとしてノードを識別します。

- データ・リーダー
- データ・ライター
- データ変換
- モデル・ビルダー
- モデル・アプライヤー
- ドキュメント・ビルダー

詳しくは、トピック 9 ページの『ノードの概要』を参照してください。

clemext_create_peer 関数には、Node 要素の id 属性と type 属性の両方の値が引数として含まれています。

単一拡張モジュールは、ぞれぞれのタイプ内で、様々な機能を実行する様々なタイプのノードを実装します。例えば、モジュールは次を実装する場合があります。

- データ・ソースのためのデータ・リーダーおよびデータ・ライター
- 様々なモデル作成アルゴリズムのためのモデル・ビルダーおよびモデル・アプライヤー
- 様々なグラフ タイプに対応するドキュメント・ビルダー

データおよびストレージ・タイプ

ピア・インスタンスは、実行開始時に指定される反復子で clemext_iterator_getRecordValue を呼び出すことによって入力データを取得し、ホストからの clemext_peer_getRecordValue 要求に応じて出力データを提供します。データはメモリでバイナリ形式で転送され、ピアおよびホストはデータ・タイプを受け入れる必要があります。

バイナリ データ・タイプはデータ・モデルによって決定し、フィールドのストレージ属性に関連します。

次のテーブルでは、可能性のあるストレージ・タイプがそれらを表すために使用するデータ・タイプと一緒にリストされています。

表 49. ストレージ・タイプ

ストレージ・タイプ	次によって表示	IBM 注
string	char *	文字列は常に UTF-8 コード化されています
real	CLEMEXTReal	倍精度浮動小数点数
integer	CLEMEXTInteger	64 ビット符号付きの整数
date	CLEMEXTDate	1970 年 1 月 1 日からの日数を表す 64 ビット符号付きの整数
time	CLEMEXTTime	午前零時からの秒数を表す 64 ビット符号付きの整数
timestamp	CLEMEXTTimestamp	1970 年 1 月 1 日からの秒数を表す 64 ビット符号付きの整数
unknown	-	値を表現できない不明なデータ型を示します

ライブラリー

サーバー側の共有ライブラリーは、設定ファイルでノード実行への対応を宣言できます。共有ライブラリーのパスは、ホスト プロセスへ動的にロードされる共有ライブラリーの場所を指定するために使用されます。共有ライブラリーは必要な API 関数をすべて定義する必要があります。 詳しくは、トピック 36 ページの『共有ライブラリー』を参照してください。

モジュール名が仕様ファイルのノード定義の Execution セクションで指定されている場合、ピア・オブジェクトを作成するために、その名前がサービス関数 clemext_create_peer の nodeId パラメーターに渡されます。この場合、拡張は適切な種類のピア モジュールを作成できます。nodeType パラメーター値は作成されるピアの種類にも影響する場合があります。共有ライブラリーが各タイプで1 つ以上のモジュールを実装しない可能性があるため、モジュール名は空白でもかまいません。

拡張モジュールを実装する共有ライブラリーによって、依存ライブラリーが必要な場合があります。これらは拡張共有ライブラリーと同じディレクトリーに存在しなくてはいけません。

一時ファイル

クライアント設定ファイルおよびサーバー拡張モジュールは、ピアが実行中に使用するファイルを作成できる一時的なプライベート スペースの **filespace** に関連したパス名を指定できます。ファイルスペースはピアのために作成されたサーバーの一時ディレクトリーのサブディレクトリーです。これは必要に応じて作成され、ピアが解除されると削除されます。

ピアはファイアスペースを存在時に完全制御します。ファイルスペースの完全パス名はノード情報ドキュメントに記載されています。これは clemext_node_getNodeInformation コールバック関数の実行結果として返される XML 形式の情報です。 詳しくは、トピック 196 ページの『ノード情報ドキュメント』を参照してください。

SQL プッシュバック

IBM SPSS Modeler ストリームが SQL データベースからデータを読み込んでデータ処理を実行する際、アドバンス・ユーザーはデータベース自体で実行する SQL 指示をプッシュバックすることにより、この操作の効率性を向上できます。

いくつかの標準 IBM SPSS Modeler ノードは SQL プッシュバックに対応しており、サーバー側 API には CLEF ノードも同様に可能にする関数呼び出しが含まれています。

clemext_peer_getSQLGeneration サービス関数は、ピア・インスタンスから SQL を生成します。この関数を使用して、SQL の実行がデータベースにプッシュバックされます。データ・リーダー・ノードに関して

は、生成された SQL はそれ自体でピアの結果セットを作成するに十分である必要があります。他のタイプのノードの場合、作成された SQL は、ピアへの入力を提供する上流ノードのために生成された SQL に依存する可能性が高くなります。ピアは、関連するノード・ハンドルの clemext_node_getSQLGeneration コールバック関数を呼び出すことにより、上流 SQL を取得することができます。

データ・モデルの処理

いくつかのサーバー側 API 呼び出しは、IBM SPSS Modeler と拡張モジュール間のデータ・モデル情報の交換に関連しています。

- clemext_node_getDataModel はノードの入力データ・モデルを取得します
- clemext_peer_getDataModel はピア・インスタンスから出力データ・モデルを取得します
- clemext_node_getOutputDataModel はノードの入力データ・モデルを取得します

他の呼び出しはモジュール内外へのデータ渡しの方法と関連しています。データ・モデルは、最後にフェッチした入力レコード内で指定されたフィールドの値を返す、次の関数でフィールド値を調べるために使用する索引を判断します。

- clemext_peer_getRecordValue
- clemext_iterator_getRecordValue

IBM SPSS Modeler は clemext_node_getDataModel を呼び出して入力データ・モデルのフィールドに関する情報を取得します。情報は次のように XML 形式で返されます。

```
<DataModel>
     <Fields>
          <Field name="abc" storage="string" type="set" />
          <Field name="uvw" storage="integer" type="range" />
          <Field name="xyz" storage="real" type="range" />
     <Fields>
</DataModel>
```

モジュールはこの情報を使用して、clemext_iterator_getRecordValue 関数によって入力レコードから値を取得する際に、フィールド・インデックスを提供することができます。

モジュールが入力データ・モデルに影響する方法は、設定ファイルにおける OutputDataModel 要素の mode 属性の値によって制御されます。モジュールは次のことができます。

- 新規フィールドを追加してモデルを拡張する。
- 既存フィールドを削除または名前の変更をしてモデルを修正する。
- 既存モデルを新規フィールドと置き換える。
- モデルを変更せずそのままにしておく。

次の例はモデルの拡張および置換について説明しています。

例 — 入力データ・モデルの拡張: これは最もシンプルなケースで、モジュールで新規フィールドの追加や値の設定が可能になりますが、既存フィールドの値の削除や変更はできません。

設定ファイルにノード定義で次のような指示があると仮定します。

```
<OutputDataModel mode="extend">
     <AddField name="field1" storage="string" ... />
     <AddField name="field2" storage="real" ... />
     ...
</OutputDataModel>
```

ここでは、出力データ・モデルが入力データ・モデルにおけるすべてのフィールドを構成していると定義され、また 2 つの追加フィールドが OutputDataModel 要素で指定されています。したがって、出力データ・モデルは 5 つのフィールドで構成されます。

clemext_peer_getDataModel 関数は、次のように追加されたフィールドに関する情報のみを返します。

```
<DataModel>
     <Fields>
          <Field name="field1" storage="string" ... />
          <Field name="field2" storage="real" ... />
     <Fields>
</DataModel>
```

返されたこの情報は、仕様ファイルの <AddField> 要素のタイプと数値 (名前ではありません) に一致している必要があります。

モジュールはコールバック関数 clemext_node_getOutputDataModel を使用して、IBM SPSS Modeler が追加を予測するフィールドの詳細を取得できます。この情報は clemext_peer_getDataModel への呼び出しに応じて IBM SPSS Modeler へまっすぐ渡すことができます。これは、出力ファイルを作成および命名する設定ファイル・ロジックが複雑な状況に便利です。

IBM SPSS Modeler が clemext_peer_getRecordValue を呼び出す際、モジュールが各出力レコードの新しい値を提供します。新規フィールドのフィールド索引は入力フィールドの最後の索引の後に開始します。この例では、入力データ・モデル内に 3 つのフィールド (索引位置 0、1、2) が存在するため、2 つの出力フィールドにフィールド索引の 3 と 4が割り当てられます。モジュールはこれらのフィールドを変更できないため、IBM SPSS Modeler が、入力フィールドに対応するフィールド索引を使用して clemext_peer_getRecordValue を呼び出すことはありません。

例 — 入力データ・モデルの置き換え **(1):** この例では、拡張モジュールはその出力から入力データ・モデル・フィールドをすべて破棄し、新規フィールドに置き換えています。

設定ファイルには次が含まれています。

```
<OutputDataModel mode="modify">
     <AddField name="key" storage="integer" ... />
     <AddField name="field1" storage="real" ... />
     <AddField name="field2" storage="real" ... />
     ...
</OutputDataModel>
```

このとき、clemext_peer_getDataModel への呼び出しで返された XML データは、出力データ・モデル内のすべてのフィールドについて説明しています。

```
<DataModel>
     <Fields>
          <Field name="key" storage="integer" ... />
          <Field name="field1" storage="real" ... />
          <Field name="field2" storage="real" ... />
     <Fields>
</DataModel>
```

clemext_peer_getRecordValue の呼び出しに使用されるフィールド索引は、最初の出力フィールド (key) を 0 として始まり、次のフィールド (field1) が 1 になります (以下同様)。

例 — 入力データ・モデルの置き換え **(2):** この例では、拡張によって提供される出力データ・モデルもまた、前の例と同じく入力データ・モデルを置き換えます。ただしこの場合は、出力データ・モデルは設定ファイルで定義されておらず、代わりにサーバーの拡張モジュールによって実行時に計算されます。設定ファイルには次が含まれています。

```
<OutputDataModel mode="modify" method="sharedLibrary" libraryId="myLibraryId" />
```

出力データ・モデルを計算するために、IBM SPSS Modeler は最初に clemext_peer_configure を呼び出し、次に clemext_peer_getDataModel を呼び出します。前の例にあるように、入力データ・モデルのフィールドはどれも自動で出力データ・モデルに含まれるのではなく、clemext_peer_getDataModel からの回答で完全に定義されます。

注 : このような場合、拡張モジュールがサーバーの出力データ・モデルを定義するときは、「無効な操作」エラーをまねくため、モジュールは出力データ・モデルを取得するために clemext_node_getOutputDataModel を使用できません。

XML 出力ドキュメント

サービスおよびコールバック関数の中には、XML 出力ドキュメントの形式でホストおよび拡張モジュール間の情報を転送するものがあります。以下の表に示すさまざまなドキュメントを使用することができます。

表 *50. XML* 出力ドキュメント

XML 出力ドキュメント	**IBM** 注	次への呼び出しによる戻し
カタログ ドキュメント	カタログに関連するコントロールの値のリストを含みます。	clemext_peer_getCatalogInformation
データ・モデル ドキュメント	ノードに入るまたはノード外のフィールドのセットについて説明します。	clemext_peer_getDataModel clemext_node_getDataModel clemext_node_getOutputDataModel
エラー詳細ドキュメント	エラーまたはその他の条件に関する情報を提供します。	clemext_peer_getErrorDetail
実行要件ドキュメント	ピア・インスタンスに必要なデータ・キャッシュや必須の入力フィールドなどの実行サポートについて説明します。	clemext_peer_getExecutionRequirements
ホスト情報ドキュメント	次のようなホスト環境に関する情報を提供します。製品識別子、説明、バージョン、プロバイダー、著作権、プラットフォームの詳細など。	clemext_host_getHostInformation
モジュール情報ドキュメント	次のような拡張モジュールに関する情報を提供します。モジュール識別子、説明、バージョン、プロバイダー、著作権、ライセンスの詳細など。	clemext_getModuleInformation
ノード情報ドキュメント	次のようなピア・インスタンスに関するノードの情報を提供します。ノード識別子、タイプ、ファイルスペースの詳細など。	clemext_node_getNodeInformation
パラメーター・ドキュメント	クライアント・ノードからの設定パラメーターを含みます。コンテンツは拡張子によって決定します。	clemext_node_getParameters

表 *50. XML* 出力ドキュメント *(*続き*)*

XML 出力ドキュメント	**IBM** 注	次への呼び出しによる戻し
SQL 生成ドキュメント	ピアの実行がどのように SQL に翻訳できるかについて説明します。	clemext_peer_getSQLGeneration clemext_node_getSQLGeneration
状況詳細ドキュメント	実行中の進行および警告、または発生するその他の条件に関する補足情報を提供します。	clemext_progress_report

カタログ ドキュメント: カタログ ドキュメントは、UI コントロールから表示できる値のリストを含む、カタログの内容を説明します。

CLEF モジュールは、getCatalogInformation への呼び出しを次のように実行します。

```
CLEMEXTStatus
getCatalogInformation(
          const char *catalogId,
            char* buffer,
            size_t buffer_size,
            size_t* data_size,
            CLEMEXTErrorCode* errorCode) {
      ...
}
```

ここで、catalogId は、設定ファイルの Catalog 要素で定義されている、特定のカタログの識別子です。

この関数は、カタログ ドキュメントを返します。

例

```
<?xml version="1.0" encoding="UTF-8" standalone="no" ?>
<CatalogInformation>
     <CatalogEntry>
          <CatalogValue>apples</CatalogValue>
          <CatalogValue>0</CatalogValue>
     </CatalogEntry>
     <CatalogEntry>
          <CatalogValue>oranges</CatalogValue>
          <CatalogValue>1</CatalogValue>
     </CatalogEntry>
     <CatalogEntry>
          <CatalogValue>bananas</CatalogValue>
          <CatalogValue>2</CatalogValue>
     </CatalogEntry>
</CatalogInformation>
```

データ・モデル ドキュメント: データ・モデル・ドキュメントは、ノードに対する入力または出力となるデータ・モデル (名前、タイプ、関連情報を持つ一連のフィールド) を記述します。ータ型ノードで使用できる情報をカプセル化します。

入力 (入力ノード）を使わないピアは空の入力モデルを持っており、また出力 8ターミナル・ノード) を出さないピアは空の出力モデルを持っています。入力を使用して出力（プロセス・ノード）を作成するピアは、その入力から出力モデルを計算する方法を知る必要があります。

ピアは、関連するノード・ハンドルの clemext_node_getDataModel を呼び出すことにより、その入力データ・モデルを取得することができます。ピアは、ホストからの clemext_peer_getDataModel 要求に応じて、その出力データ・モデルを提供します。

どのデータ・モデルもそのプロパティーと一緒にデータ・モデル内のフィールドをすべて列挙するデータ・ディクショナリーとして直接表現できます。ノードによって提供されるピアに対する入力データ・モデルは常にこの形式です。ピアで作成された出力データ・モデルは同じ形式を持っている、または代わりに入力モデルに適用される操作 (フィールドの追加、フィールドの削除、フィールドの修正) のシーケンスとして表現される場合があります。これは同じノードの出力モデルを大幅に簡略化します。

データ・モデル ドキュメントのフィールドの順序は非常に重要で、対応する入力または出力データ・セットでどのデータを示すかという順番を決定します。

データ・モデルは完全でなくデータの一部特性のみを提供する場合があります。ピアが実行プランを計算できるよう十分に指定された入力モデルは、そのピアに対して **executable** と呼びます。実行可能なデータ・モデルは、入力および出力データが正確に配列できるよう、各フィールドに対するバイナリ タイプを常に含んでいる必要があります。

例

```
<?xml version="1.0" encoding="utf-8"?>
<DataModel>
  <Fields>
    <Field name="Age" type="range" storage="integer" direction="in">
      <Range minValue="15" maxValue="74"/>
    </Field>
    <Field name="Sex" type="flag" storage="string">
      <Values>
        <Value value="F" flagValue="false" displayLabel="Female"/>
        <Value value="M" flagValue="true" displayLabel="Male"/>
      </Values>
    </Field>
    <Field name="BP" type="orderedSet" storage="integer">
      <Values>
        <Value value="-1" />
        <Value value="0" />
        <Value value="1" />
      </Values>
    </Field>
    <Field name="Cholesterol" type="flag" storage="string">
      <Values>
        <Value value="NORMAL" flagValue="false"/>
        <Value value="HIGH" flagValue="true"/>
      </Values>
    </Field>
    <Field name="Na" type="range" storage="real" displayLabel="Blood sodium">
      <Range minValue="0.500517" maxValue="0.899774"/>
    </Field>
    <Field name="K" type="range" storage="real" displayLabel="Potassium concentration">
      <Range minValue="0.020152" maxValue="0.079925"/>
    </Field>
    <Field name="Drug" type="set" storage="string" direction="out">
      <Values>
        <Value value="drugA"/>
        <Value value="drugB"/>
        <Value value="drugC"/>
        <Value value="drugX"/>
        <Value value="drugY"/>
      </Values>
    </Field>
  </Fields>
</DataModel>
```

エラー詳細ドキュメント: エラー詳細ドキュメントは IBM SPSS Modeler にメッセージ (エラー、警告、情報) を送り返すために使用され、エラーやその他の条件に関する情報を提供します。拡張モジュールは、エラー詳細ドキュメントを提供することにより、ホストからの clemext_peer_getErrorDetail 要求に応じてモジュール固有のエラーを記述することができます。

エラー詳細とは、各診断に少なくとも 1 つのエラー・コード、メッセージ、そしてメッセージに挿入される詳細を含む 1 つ以上のパラメーターのセットを含んだ 1 つ以上の Diagnostic 要素のセットです。エラー・コードは設定ファイルの StatusCode 要素の値と一致します。

メッセージは異なる言語バリアントを持っている場合があり、そうでない場合はクライアントがエラー・コードを使用してリソース・バンドルからローカライズされたメッセージを選択できます。診断要素のシーケンスはエラーの因果連鎖を説明します。

例

```
<?xml version="1.0" encoding="utf-8"?>
<ErrorDetail>
  <Diagnostic code="123" severity="error">
    <Message>You can't do that ({0})</Message>
    <Parameter>Permission denied</Parameter>
  </Diagnostic>
  <Diagnostic code="456" severity="warning">
    <Message>That was silly!</Message>
    <Message lang="fr">Quel idiot!</Message>
  </Diagnostic>
</ErrorDetail>
```

実行要件ドキュメント: 実行要件ドキュメントはピア・インスタンスに必要な実行サポートについて説明します。ピア・インスタンスは、ホストからの clemext_peer_getExecutionRequirements リクエストに応じて実行要件ドキュメントを提供できます。正しい実行環境を提供するため、ホストはピアで clemext_peer_beginExecution を呼び出す前に、要件ドキュメントを参照します。

ホストは、モジュールが clemext_iterator_rewind 関数を使用することによって入力データに対して複数のパスができるよう、データ・キャッシュ・サービスを提供することができます。

例

```
<?xml version="1.0" encoding="utf-8"?>
<ExecutionRequirements>
  <Cache/><!-- this ensures that the CLEF module can make multiple passes over the input
    data -->
</ExecutionRequirements>
```

ホスト情報ドキュメント: ホスト情報ドキュメントはホスト環境について説明します。拡張モジュールは、ホスト・ハンドルの clemext_host_getHostInformation を呼び出すことにより、ホスト情報を取得することができます。

返される情報には製品識別子、詳細、バージョン、プロバイダー、著作権、プラットフォームなどが含まれます。

例

```
<?xml version="1.0" encoding="utf-8"?>
<HostInformation>
  <Host name="clemlocal" externalEncoding="cp1252" language="english_us"
        locale="English_United Kingdom.1252" provider="IBM Corp." version="18.1.1" platform=
        "Windows XP SP2" copyright="Copyright 1995-2011 IBM Corp. All rights reserved.">
```

```
    <VersionDetail major="12" minor="0"/>
    <PlatformDetail osType="windows" osName="WindowsNT" osMajor="5" osMinor="1"/>
    <LibraryDetail path="C:¥Program Files¥IBM¥SPSS¥Modeler¥18.1.1¥ext¥bin¥my.module¥myModule.dll"/>
  </Host>
</HostInformation>
```

モジュール情報ドキュメント: モジュール情報ドキュメントは拡張モジュールについて説明します。拡張モジュールは、ホストからの clemext_getModuleInformation リクエストに応じてモジュール情報ドキュメントを提供する必要があります。

返される情報にはモジュール識別子、詳細、バージョン、プロバイダー、著作権、ライセンスなどが含まれます。

例

```
<?xml version="1.0" encoding="utf-8"?>
<ModuleInformation>
  <Module name="MyModule" provider="My Company Inc." version="10.1.0.329"
          copyright="Copyright 2006 My Company Inc. All rights reserved.">
    <VersionDetail major="10" minor="1" release="0" build="329"/>
    <Licence code="1234" type="mandatory"/>
    <Description>Provides a thorough test of the new extensions framework.</Description>
  </Module>
</ModuleInformation>
```

ノード情報ドキュメント: ノード情報ドキュメントはピア・インスタンスに関連したノードについて説明します。ピア・インスタンスは、ノード・ハンドルの clemext_node_getNodeInformation を呼び出すことにより、ノード情報を取得することができます。ノード情報にはノード識別子、タイプ、ファイルスペースなどの詳細が含まれます。

例

```
<?xml version="1.0" encoding="utf-8"?>
<NodeInformation>
  <Node name="databaseImport" type="dataReader">
    <FileSpace path="C:¥Program Files¥IBM SPSS Modeler Server
18.1.1¥tmp¥ext-8005-6711-01"/>
  </Node>
</NodeInformation>
```

パラメーター・ドキュメント: パラメーター・ドキュメントには設定ファイルで定義された各 Property 要素の詳細が含まれます。詳細は、ピアがノードハンドルの clemext_node_getParameters を呼び出すことによって取得できる、設定パラメーターの形式で返されます。

パラメーターには名前と値があり、値は次のようにできます。

- 単純値 (文字列)
- キー値 (キーおよび値)
- 構造値 (名前付き値のセット)
- 値のリスト

パラメーター・ドキュメントの内容は拡張パッケージによって完全に決定します。パラメーターはクライアント設定ファイルで定義され、サーバー拡張モジュールで解釈されます。

例

```
<?xml version="1.0" encoding="utf-8"?>
<Parameters>
  <Parameter name="linesToScan" value="50"/>
  <Parameter name="useCaption" value="true"/>
  <Parameter name="caption" value="My Caption"/>
  <Parameter name="captionPosition" value="north"/>
  <Parameter name="defaultAggregation">
    <ListValue>
      <Value value="min"/>
      <Value value="max"/>
      <Value value="mean"/>
      <Value value="stddev"/>
    </ListValue>
  </Parameter>
</Parameters>
```

SQL 生成ドキュメント: SQL 生成ドキュメントは、ピアの実行がどのように SQL に翻訳できるかについて説明します。

ピアは、ホストからの clemext_peer_getSQLGeneration 要求に応じて、SQL 生成ドキュメントを提供することができます。ホストは内部でのピア実行に優先して、SQLの実行を試みます。

入力を使用するピアは、関連するノード・ハンドルで clemext_node_getSQLGeneration を呼び出すことにより、その入力 SQL を取得することができます。

SQL 生成ドキュメントの主なコンポーネントは、ノードまたはストリームのフラグメントの実行動作を複製する SQLステートメントです。データを作成するノードについては (つまり、データ・リーダーまたはデータ・トランスフォーマー・ノード）、文は SELECT となる必要があり、また SELECT 文でコラム名にデータ・モデルのフィールド名をマップするディクショナリーを伴う必要があります。

SQL 生成ドキュメントは、データ・ソース名や製品名など、ステートメントの実行に対するデータベース接続のプロパティーも含んでいる場合があります。ピアはこれらのプロパティーを使用して作成する SQL の決定を促すことができます。

例

```
<?xml version="1.0" encoding="utf-8"?>
<SqlGeneration>
  <Properties
    datasourceName="SQL Server"
    databaseName="DataMining"
    serverName="GB1-RDUNCAN1"
    passwordKey="PW0"
    userName="fred"
    dbmsName="Microsoft SQL Server"
    dbmsVersion="09.00.1399"/>
  <Statement>
    <Bindings>
      <Binding columnName="C0" fieldName="ID"/>
      <Binding columnName="C1" fieldName="START_DATE"/>
    </Bindings>
    <TableParameters>
 <TableParameter name="${TABLE26}" value="dbo.DRUG4N"/>
    </TableParameters>
    <Sql>
      SELECT
      T0.ID AS C0,T0."START_DATE" AS C1
      FROM ${TABLE26} T0
```

```
      WHERE (T0."START_DATE" > '2003-01-01')
      ORDER BY 2 ASC
    </Sql>
  </Statement>
</SqlGeneration>
```

状況詳細ドキュメント: 状況詳細ドキュメントは、進行状況や致命的でない警告、または実行中に発生するその他の条件に関する情報を提供します。拡張モジュールは、clemext_progress_report コールバック関数を使用して、状況詳細ドキュメントを非同期的にディスパッチすることができます。

状況詳細ドキュメントとは、各診断に少なくとも 1 つの条件 コード、メッセージ、そしてメッセージ (プロパティー・ファイルに提供されていない場合) に挿入される詳細を含む 1 つ以上のパラメーターのセットを含んだ 1 つ以上の Diagnostic 要素のセットで構成されています。StatusDetail 要素には、メッセージを次のいずれかに渡すよう指示するオプションの destination 属性もあります。

- IBM SPSS Modeler が管理するローカル トレース・ファイル
- クライアント (ユーザーへ向けたメッセージ)
- すべて (可能性のあるすべての行き先に送る)

Diagnostic 要素の形式は次のとおりです。

```
<Diagnostic code="integer" severity="severity_level">
     <Message>message_text</Message>
     <Parameter>value</Parameter>
</Diagnostic>
```

ここで、

code (必須) は、条件コードを示す整数です。

severity は、条件の重大度 (unknown、information、warning、error、または fatal) を示します。

例

```
<?xml version="1.0" encoding="utf-8"?>
<StatusDetail destination="client">
  <Diagnostic code="654" severity="information">
    <Message>Processed {0} records</Message>
    <Parameter>10000</Parameter>
  </Diagnostic>
</StatusDetail>
```

ローカライズされたメッセージの使用

プロパティー・ファイルのローカライズされたメッセージを使用する場合、状況詳細ドキュメントから Message 要素を省略し、次の例のように設定ファイルのメッセージ キーを使用します。

```
    ...
    <Execution ...>
    ...
        <StatusCodes>
            ...
            <StatusCode code="21" status="warning" messageKey="fieldIgnoredMsg.LABEL"/>
            ...
        </StatusCodes>
    </Execution>
    ...
```

messages.properties ファイルには、次のものが含まれます。

```
fieldIgnoredMsg.LABEL=Field "{0}" cannot be used for model building and was ignored
```

状況詳細ドキュメントでは、フィールド名などのパラメーターを送信して、次のようにローカライズされたメッセージに送信することができました。

```
<?xml version="1.0" encoding="utf-8"?>
<StatusDetail>
    <Diagnostic code="21">
        <Parameter>BP</Parameter>
    </Diagnostic>
</StatusDetail>
```

C++ ヘルパー

CLEF 例ノードのいくつかには、**helpers** として知られる事前定義済の C++ ソース・ファイルが多く含まれています。これらはいくつかの C ベースのサーバー側 API にはラッパーとして働き、容易に C++ CLEF へコンパイルできます。

表 *51. C++* ヘルパー

ヘルパー	説明
BufferHelper	C API に使用されるサイズ変更が可能なメモリ・バッファを管理します
DataHelper	データのラップ、読み書き操作に役立ちます。
HostHelper	LEMEXT HostHandle オブジェクトをラップします
XMLHelper	XML 処理をラップします

ヘルパーは、*.cpp* ファイルと *.h* ファイルのペアの形式をとります (*BufferHelper.cpp* や *BufferHelper.h* など)。

これらのヘルパーファイルに関する詳細は、 29 ページの『ソース・コードの検査』を参照してください。

これらのファイルに関する詳細は、次のようにサーバー側 API ドキュメントに記載されています。

1. CLEF API ドキュメンテーション画面より、「サーバー側 **API** の概要」 を選択します。
2. 「ファイル」タブをクリックします。
3. 必要な情報のヘルパーに対応する .h ファイルの名前をクリックします。
4. **Data Structures** の中で、対応するクラス名をクリックしてドキュメントを表示します。

CLEF API ドキュメントへのアクセスに関する情報は、 179 ページの『CLEF API ドキュメンテーション』を参照してください。

エラー処理

各関数呼び出しは、ステータス・コード (CLEMEXTStatus) およびオプションのモジュール固有エラー・コード (CLEMEXTErrorCode) を返します。ステータス・コードは success (エラーなし)、または API 関数用に列挙されたエラー・コードの 1 つとなります。これらはほとんど常に「モジュール固有のエラー」を含みます。モジュール固有のエラー・コードは、「モジュール固有エラーなし」を意味する 0 になる可能性もあります。

ステータス・コード メッセージは IBM SPSS Modeler によって提供されます。一般的なステータス・コードに関する詳細は、次のようにサーバー側 API ドキュメントに記載されています。

1. CLEF API ドキュメンテーション画面より、「サーバー側 **API** の概要」 を選択します。

2. 「モジュール」 タブをクリックします。
3. 「一般的なステータス・コード」 を選択します。

CLEF API ドキュメントへのアクセスに関する情報は、 179 ページの『CLEF API ドキュメンテーション』を参照してください。

モジュール固有エラー・メッセージは次のように提供されることがあります。

- 設定ファイルで (モジュール・セクションの `StatusCodes` で）
- 設定ファイルで参照されるリソース・バンドルで
- 拡張モジュールによって

モジュール固有のエラー・コードについては、モジュールが、デフォルトのエラー・メッセージ (設定ファイルまたはリソース・バンドルで説明されないエラー) およびメッセージに挿入されるパラメーターから成る追加のエラー詳細を提供できます。複数のエラーメッセージはエラーの因果連鎖を説明できます。

クライアントでのエラー報告には次の形式があります。

node_label:*message*

ここで、

- *node_label* は、モジュールが指定される `Node` 要素の `label` 属性の値です。
- *message* は、サーバーから提供される、または設定ファイル (またはローカリゼーション用の *.properties* ファイル) で定義されるメッセージのテキストです。

XML API の解析

IBM SPSS Modeler には Apache の Xerces-C XML parser を含まれており、モジュールで XML データの読み書きができる多数のコールバックを提供します。希望により自身の XML パーサーで代用することもできます。

サーバー側 API の使用方法

ノードにサーバー側関数の呼び出しを含むには、

1. 関数呼び出しを含む C++ *.cpp* and *.h* ソース・ファイルを作成します。
2. ソース・ファイルをダイナミック・リンク・ライブラリー (*.dll*) ファイルへコンパイルします。
3. 仕様ファイルから *.dll* ファイルへの参照を組み込みます。以下に例を示します。

```
<Resources>
.
     <SharedLibrary id="mylib1" path="mycorp.mynode/mylib" />
.
</Resources>
```

詳しくは、トピック 36 ページの『共有ライブラリー』を参照してください。

このリリースで供給されているノード例のソース・コードを見ると便利です。 詳しくは、トピック 29 ページの『ソース・コードの検査』を参照してください。

サーバー側のプログラミング・ガイドライン

CLEF モジュールのサーバー側ダイナミック・リンク・ライブラリー (DLL) の部分は、モジュールが適切に機能し、IBM SPSS Modeler の操作の影響を回避するよう、ガイドラインに従う必要があります。CLEF モジュールは、次のことが必要です。

- ピア実行が内蔵されている
- 単一プロセスの複数のピアインスタンスをサポートする
- スレッドが安全であること
- スレッドまたはプロセス環境の警告を回避する
- モジュール内のスレッドの使用を制限する
- 実行キャンセルの要求を正しく処理する
- 中断されたシステムの呼び出しを再開する (UNIX)
- CoInitialize または CoUninitialize の呼び出し時に注意する (Windows)
- モジュールがアンロードされた場合の仮説を立てないようにする
- サブプロセス開始時に注意する
- 標準出力または標準誤差への書き込みを回避する

次のセクションでは、これらの領域についてより詳細に説明します。

ピア実行が内蔵されている

ピア・インスタンスは IBM SPSS Modeler サーバー・プロセス内の他のピア・インスタンスの有無に関する推定はできません。IBM SPSS Modeler は、ストリーム内で隣接するノードに対応するピア・インスタンスが異なる段階で実際に実行されるよう、実行のスケジュールを設定できます。そのため、インスタンスの有無と実行は重複しません。

ピア・インスタンスは内蔵されており、パイプまたはソケットなどを使用してそのほかのインスタンスと直接通信することはありません。ピア・インスタンス間のすべての通信は、データを読み込んだり書き込んだりして、または外部エージェント (ピア間のデータ共有を管理するデータベース・サーバーなど) を使用して直接実行する必要があります。

単一プロセスの複数のピアインスタンスをサポートする

エンド・ユーザーは、ストリーム実行時のサーバー・プロセスで、特定の CLEF モジュールで複数のピア・インスタンス (同じタイプの複数のノードなど) を作成する場合があります。そのため、CLEF モジュールの静的データは複数のピア・インスタンス間で共有され、ピア・オブジェクト専用のデータを保存するために使用することはできません。静的データの例としては、C++ クラスの静的メンバー、C コンパイル・ユニットのグローバルまたは静的変数があります。

CLEF モジュールの API 関数を再度使用し、再使用しないシステム コールを行わないようにする必要があります。例えば、ピア・インスタンスが `clemext_iterator_nextRecord` を使用して入力反復子から入力データを取り出すときに、最初のピアの上流に存在し、最終的に最初のピアによって使用されるデータを生成する 2 番目のピア・インスタンスの `clemext_peer_nextRecord` が呼び出される可能性があります。

`strtok` のようなシステム コールは、再使用ではなく、また使用できません。再使用である代替の詳細は、ご使用のプラットフォームのオペレーティング・システム・マニュアルを参照してください。

スレッドを安全にする

IBM SPSS Modeler は、複数のピア・インスタンスの実行を異なる実行スレッドからインターリーブする場合があります。そのため、ミューテックス (相互排除オブジェクト) または同様のスレッド ライブラリー・サービスと同期するなどして、ピア・オブジェクト間で共有されたリソースへのアクセスを保護する必要があります。

CLEF モジュールは、スレッドセーフでないシステム コールを行わないようにする必要があります。詳細は、ご使用のオペレーティング・システムのマニュアル、または UNIX man を参照してください。

スレッドまたはプロセス環境の警告を回避する

スレッドまたはプロセスを呼び出す環境が変更されるシステム コールを使用しないようにします。

そのような呼び出しの例は、次のとおりです。ただし、このリストは完全ではありません。

- setlocale : ロケール情報の読み取りではなくロケールの変更に使用する場合
- SetCurrentDirectory (Windows) または chdir (UNIX)
- LogonUser (Windows) または seteuid (UNIX)
- putenv
- exit
- signal

注 :Windows の場合、CoInitialize は、スレッドの環境を変更しますが必要な場合があります。 詳しくは、トピック 203 ページの『CoInitialize または CoUninitialize の呼び出し時の注意 (Windows)』を参照してください。

モジュール内のスレッドの使用を制限する

通常、モジュールはスレッドを内部で自由に使用できます。ただし、IBM SPSS Modeler が CLEF モジュール関数 (clemext_peer_cancelExecution 以外) の呼び出しで使用したスレッドでのみ、コールバック関数を呼び出す必要があります。

次のコールバック関数はモジュール内で実行しているスレッドから非同期的に呼び出すことができます。

- clemext_progress_report
- clemext_channel_send

ピア・インスタンスは、複数のスレッドがこれらの呼び出しのそれぞれに同時に起動しないようにする必要があります。

実行キャンセルの要求を正しく処理する

エンド・ユーザーがピア・インスタンスの実行のキャンセルを要求した場合、IBM SPSS Modeler はモジュールの clemext_peer_cancelExecution 関数への非同期的な呼び出しを実行します。開発者は、この呼び出しを実行する必要があります。この関数は非同期でコールされ、別の CLEF API 関数コールが実行されている間にコールされます。

中断されたシステムの呼び出しを再開する (UNIX)

UNIX の場合、IBM SPSS Modeler アプリケーションは信号および信号ハンドラを使用します。一部の UNIX システム コールは、コール実行時に信号を受信した場合に EINTR コードを返す場合があります。特定の UNIX プラットフォームのシステム コールについては、man ページを参照してください。

このイベントが発生すると、コールしたコードは EINTR リターン・コードを確認し、コールを再開する必要があります。これを実行する方法の 1 つとして、簡単なラッパー関数 (open_safe) を作成して、アプリケーションが該当するラッパーをコールするようにします。

```
int
open_safe(const char* path, int oflag, mode_t mode) {
  int res;
  while ((res = ::open(path, oflag, mode)) == -1
                                    && errno == EINTR) {
  }
  return res;
}
```

CoInitialize または CoUninitialize の呼び出し時の注意 (Windows)

Windows の場合、Windows Component Object Model (COM) ライブラリー・サービスを使用する必要のあるスレッドは、COM サービス使用前にシステム API 関数 CoInitialize を呼び出し、完了時に CoUninitialize を呼び出す必要があります。IBM SPSS Modeler がモジュールに CLEF API を起動するスレッドは、CoInitialize を呼び出す場合もあれば、呼び出さない場合もあります。

これらのスレッドから COM サービスを使用する CLEF モジュールは、通常、clemext_create_peer 関数または clemext_peer_beginExecution 関数で CoInitialize を呼び出す必要があります。呼び出しが成功した場合、モジュールは、スレッドでの実行の完了時に、(通常は clemext_destroy_peer または clemext_peer_endExecution で) CoUninitialize も呼び出す必要があります。

CoInitiaize の呼び出しの詳細は、Microsoft Developer Network (MSDN) の Web サイト (http://msdn.microsoft.com) にあるマニュアルを参照してください。

モジュールがアンロードされた場合の仮説を立てないようにする

現在、CLEF モジュールはセッション終了までロードされたままになっています (モジュールを必要に応じてアンロードおよび再ロードすることはできません)。モジュールがロードされる IBM SPSS Modeler サーバー・プロセスからの終了時にも、関数 clemext_cleanup はコールされません。そのため、開発者はいかなる場合にも、モジュールがアンロードされ、リソースが解放されるという仮説を立てることはできません。

サブプロセス開始時に注意する

サブプロセスを開始すると、CreateProcess (Windows) または fork (UNIX) によって、親プロセスおよび子プロセスの対話、および親にオープンなリソースを子プロセスが継承する状況に多くの混乱をきたす場合があります。

CLEF モジュールがプロセス外で実行する必要がある場合、適切な代替のアーキテクチャーの使用を検討してください。例えば、CLEF モジュールは、必要なタスクを実行するアプリケーション・サーバーに提供されたサービスを使用する場合があります。

特に、Windows のプロセスでは、TRUE に設定された bInheritHandles パラメーターを含む CreateProcess 関数を使用してサブプロセスを開始しないようにする必要があります。そうすることによって、子プロセスは、親 (IBM SPSS Modeler サーバー) プロセスでオープンなすべてのファイル記述子を継承します。

標準出力または標準誤差への書き込みを回避する

CLEF モジュールがプロセスの標準出力または標準誤差のストリームに書き込むと (おそらくデバッグの目的)、通常エンド・ユーザーには表示されません。ただし、CLEF ノードを含むストリームが IBM SPSS

Modeler Solution Publisher を使用して表示され、コマンド・ライン シェルから実行されると (Windows または UNIX)、この出力が表示され、ユーザーは混乱する場合があります。

代わりに、CLEF モジュールは、ホスト・コールバック関数 `clemext_host_trace` をコールして文字列形式で表示するメッセージを渡して、トレース・サービスを起動することができます。IBM SPSS Modeler Server 構成オプション・ファイル (IBM SPSS Modeler インストール・ディレクトリーの */config/options.cfg*) で次の設定を使用し、IBM SPSS Modeler のインストールでトレースを有効にする必要もあります。

```
trace_extension, 1
```

トレースされたメッセージは、IBM SPSS Modeler インストール・ディレクトリーのファイル */log/trace-<process_ID>-<process_ID>.log* に出力されます。*process_ID* は、IBM SPSS Modeler Server プロセスの識別子です。セッションはすべて同じログ・ファイルを共有しているため、複数のセッションを同時にトレースしないようにしてください。

第 10 章 テストと配布

CLEF 拡張のテスト

他のユーザーに配布する前に新しい拡張をテストすることをお勧めします。

仕様ファイルおよび関連するリソース・バンドル、.jar ファイル、共有ライブラリーおよびユーザー・ヘルプ・ファイルを作成した後、ファイルを必須ファイル構造に配置し、それらをローカルの IBM SPSS Modeler のインストール・ディレクトリーにコピーすることによって拡張をテストできます。次回 IBM SPSS Modeler を起動するときに、新しい拡張が IBM SPSS Modeler ユーザー・インターフェースに表示されます。

CLEF 拡張のテスト

1. IBM SPSS Modeler が開いている場合は終了します。
2. 拡張が CLEF ノードまたは出力を定義する場合、拡張が正しく機能するまで、拡張ダイアログの「デバッグ」タブを有効にすることをお勧めします。 詳しくは、トピック 206 ページの『「デバッグ」タブの使用』を参照してください。
3. クライアント側ファイルおよびサーバー側ファイルを必須構造に配置します。仕様ファイルおよびノードで必要な関連したリソース (`.jar` または `.dll` ファイルなど) が適切な場所にコピーされます。 詳しくは、トピック 5 ページの『ファイル構造』を参照してください。
4. クライアント側のディレクトリーを IBM SPSS Modeler インストール・ディレクトリーの *¥ext¥lib* フォルダーにコピーします。
5. サーバー側のディレクトリーを IBM SPSS Modeler インストール・ディレクトリーの *¥ext¥bin* フォルダーにコピーします。
6. IBM SPSS Modelerを始動します。
7. 拡張でメニューまたはメニュー項目を定義する場合、メイン・メニュー・システムに正しく表示されていることを確認します。拡張が新しいノードを定義する場合、仕様ファイルに定義されているように、正しいノード・パレットの該当する位置にノードが表示されていることを確認します。
8. 拡張を徹底的にテストします。

   例えば次のことを確認します。
   - フィールドおよびレコードの数が増えてもノードのパフォーマンスが低下していない
   - ヌル値が一貫して処理されている
   - 必要に応じてさまざまなロケール (ヨーロッパ、極東) がサポートされている
9. 拡張を定義した後でも、仕様ファイルに変更を行うことができます。ただし、変更は IBM SPSS Modeler を再起動するまでは無効です。

CLEF 拡張のデバッグ

CLEF では、拡張のデバッグを支援する次の機能を定期要します。

- XML 構文エラーのメッセージ
- 外部実行
- 「デバッグ」タブ

XML 構文エラー

仕様ファイル内に正しくない XML 構文が存在する場合は、XML パーサーからのエラー・メッセージによってフラグが立てられます。

メッセージでは、エラーの内容とともに、エラーのおおよその行番号を表示します。

この状況を解決する手順は次のとおりです。

1. ファイルのエラーを修正します。
2. 205 ページの『CLEF 拡張のテスト』 の手順に従って、ファイルを再テストします。
3. 仕様ファイルに構文エラーがなくなるまで、この手順を繰り返します。

外部実行

通常、ユーザーによって記述された CLEF 拡張は、IBM SPSS Modeler プロセスとは別に独自のプロセスで実行します。これにより、デバッグ時、拡張プロセスが失敗した場合でも、IBM SPSS Modeler Server プロセス全体に取り込まれることがありません。

注 : デフォルト設定を無効にすることはできません。 詳しくは、トピック 207 ページの『実行オプションの変更』を参照してください。

「デバッグ」タブの使用

CLEF ノードまたは出力に関連するダイアログまたはフレームの場合、「デバッグ」タブを有効化してオブジェクトのプロパティー設定を調査できるようにします。また、拡張で定義されたコンテナーの内容を表示して、これらの内容をさらに調査するためにファイルに保存することもできます。 詳しくは、トピック 54 ページの『コンテナー』を参照してください。

「デバッグ」タブを有効にするには、仕様ファイルの `Extension` 要素の `debug` 属性の値を `true` に設定します。 詳しくは、トピック 33 ページの『Extension 要素』を参照してください。

タブのフィールドは、次のとおりです。

要素 **ID**。拡張の一意の識別子。仕様ファイルの `ExtensionDetails` 要素の `id` 属性の値です。

スクリプト名。スクリプト内で参照する場合のノードの一意の識別子。`Node` 要素の `scriptName` 属性の値です。

拡張 **ID**。拡張のファイルとディレクトリー・リソースが存在する拡張フォルダーの名前です。この値は、`ExtensionDetails` 要素の `providerTag` 属性と `id` 属性を「`.`」文字で区切って連結することによって生成されます。識別子の `providerTag` の部分には、値には文字列 `spss` が含まれ、内部の使用に指定されます。

プロパティー。この表では、ノードの `Property` 宣言の選択された詳細情報を示しています。

- プロパティー。プロパティーの一意の識別子。`Property` 要素の `name` フィールドの値です。
- スクリプト名。スクリプト内で参照する場合のプロパティーの一意の識別子。`Property` 要素の `scriptName` 属性の値です。
- 値のタイプ。このプロパティーがとることのできる値のタイプ。`Property` 要素の `valueType` 属性で定義されます。
- リスト**?** プロパティーが、指定されたタイプの値のリストであるかどうかを示します。`Property` 要素の `isList` 属性の値です。

- 共有しますか ? これを選択すると、このプロパティーが拡張内の複数の場所 (モデル・ビルダー・ノード、モデル出力、モデル・アプライヤー) で使用されます。
- 値。プロパティーのデフォルト値です (ある場合)。

コンテナー。選択されたコンテナーの内容 (モデル・データなど) を表示しますこのフィールドをクリックして、拡張に定義されたその他のコンテナーのリストを表示し、さまざまなコンテナーを選択して内容を表示します。隣接する「コンテナーの保存」ボタンをクリックして、より詳細に調査するために選択されたコンテナーの内容を XML 形式で保存します。

トレース。ノードが実行される場合にトレース 出力ができるようにするダイアログを表示します。

実行オプションの変更

デフォルトでは、ユーザーによって記述された CLEF 拡張モジュールは、IBM SPSS Modeler プロセスとは別のプロセスで実行します。このように、実行プロセスの失敗によって、IBM SPSS Modeler プロセスが失敗することはありません。これに対し、IBM Corp. が提供するモジュールは、デフォルトではメインのプロセスで実行します。

2 つのサーバー構成オプションによって、システム管理者は指定されたモジュールのこれらのケースを反対のケースに変更することができます。2 つのオプションは、モジュール識別子のカンマで区切られたリストで、変更の影響を受けているモジュールを示します。

注 : これらのオプションのいずれかの変更は、通常、カスタマ・サポート担当者の要求によってのみ実行できます。

オプションは次の通りです。

プロセス内実行オプション

このオプションで、通常外部プロセスにロードされる拡張モジュール (通常、ユーザーによって記述されたモジュール) を IBM SPSS Modeler に直接ロードすることができます。形式は次のとおりです。

```
clef_inprocess_execution, "moduleID1[,moduleID2[,...moduleIDn]]"
```

moduleID は、関連する仕様ファイルの `ExtensionDetails` 要素の `id` 属性の値です。次に例を挙げます。

```
clef_inprocess_execution, "test.example_filereader"
```

外部実行オプション

このオプションで、通常 IBM SPSS Modeler に直接ロードされる拡張モジュール (通常、IBM Corp. によって提供されたモジュール) を外部プロセスにロードすることができます。形式は次のとおりです。

```
clef_external_execution, "moduleID1[,moduleID2[,...moduleIDn]]"
```

ここで、*moduleID* は、`clef_inprocess_execution` の場合と同じです。架空の例を次に示します。

```
clef_external_execution, "spss.naivebayes,spss.terminator"
```

実行オプションの追加または変更

実行オプションを追加または変更するには、『 *IBM SPSS Modeler 18.1.1* サーバー管理およびパフォーマンス・ガイド』の「options.cfg ファイルの使用」に示された手順に従います。

CLEF 拡張の配布

新しい拡張がすべてテストされると、配布の準備ができます。

1. 「デバッグ」タブが有効な場合は、無効にします。 詳しくは、トピック 206 ページの『「デバッグ」タブの使用』を参照してください。
2. 拡張ファイルをインストールする方法に正確に反映されるファイル構造を作成します。 詳しくは、トピック 5 ページの『ファイル構造』を参照してください。
3. ファイル構造を .zip ファイルに圧縮します。クライアント側インストールおよびサーバー側インストールにそれぞれ .zip ファイルを作成するより簡単です。
4. .zip ファイルをエンド・ユーザーに配布します。

CLEF 拡張のインストール

CLEF 拡張をインストールする手順は、次のとおりです。

1. 拡張ファイル構造を含む .zip ファイルを受け取ったら、IBM SPSS Modeler インストール・ディレクトリーの ¥ext¥lib フォルダーにクライアント側のファイルを展開します。
2. IBM SPSS Modeler インストール・ディレクトリー (IBM SPSS Modeler Server を使用する場合同等のディレクトリー) の ¥ext¥bin フォルダーにサーバー側のファイルを展開します。
3. IBM SPSS Modeler 起動して、新しいノードがノード・パレットの該当する場所に表示されていることを確認します。
4. IBM SPSS Collaboration and Deployment Services または IBM SPSS Modeler Solution Publisher などの展開環境を使用している場合は、手順 1 および 2 を繰り返し、それらの製品の ¥ext¥lib フォルダーと ¥ext¥bin フォルダーにファイルを追加して、SPSS Modeler を再起動します。

CLEF 拡張のアンインストール

CLEF 拡張をアンインストールする手順は、次のとおりです。

1. IBM SPSS Modeler インストール・ディレクトリーの ¥ext¥lib ディレクトリーにある拡張フォルダーを検索します。

   拡張でサーバー側の拡張フォルダーもインストールしている場合、IBM SPSS Modeler または IBM SPSS Modeler Server のインストール・ディレクトリーの ¥ext¥bin ディレクトリーでこのフォルダーを検索します。
2. 拡張フォルダーを削除します。
3. IBM SPSS Collaboration and Deployment Services または IBM SPSS Modeler Solution Publisher などの展開環境を使用している場合は、手順 1 および 2 を繰り返し、それらの製品の ¥ext¥lib フォルダーと ¥ext¥bin フォルダーからファイルを削除します。

変更は、次回 IBM SPSS Modeler を起動した場合に有効になります。

付録. CLEF XML スキーマ

CLEF 要素の参照

このセクションでは、CLEF のすべての要素の参照を提供しています。

各トピックには、要素および親要素や子要素の有効な属性を示しています。これらの要素は、親トピックの子としてではなく、このトピック (「CLEF 要素の参照」) の子として目次に表示されています。

要素

Action 要素

表 52. *Action* の属性

属性	使用	説明	有効な値
description	オプション		*string*
descriptionKey	オプション		*string*
ID	必須		*string*
imagePath	オプション		*string*
imagePathKey	オプション		*string*
label	必須		*string*
labelKey	オプション		*string*
mnemonic	オプション		*string*
mnemonicKey	オプション		*string*
resourceKey	オプション		*string*
shortcut	オプション		*string*
shortcutKey	オプション		*string*

XML 表記

```
<xs:element name="Action">
  <xs:attribute name="id" type="xs:string" use="required"/>
  <xs:attribute name="label" type="xs:string" use="required"/>
  <xs:attribute name="labelKey" type="xs:string" use="optional"/>
  <xs:attribute name="description" type="xs:string" use="optional"/>
  <xs:attribute name="descriptionKey" type="xs:string" use="optional"/>
  <xs:attribute name="imagePath" type="xs:string" use="optional"/>
  <xs:attribute name="imagePathKey" type="xs:string" use="optional"/>
  <xs:attribute name="mnemonic" type="xs:string" use="optional"/>
  <xs:attribute name="mnemonicKey" type="xs:string" use="optional"/>
  <xs:attribute name="shortcut" type="xs:string" use="optional"/>
  <xs:attribute name="shortcutKey" type="xs:string" use="optional"/>
  <xs:attribute name="resourceKey" type="xs:string" use="optional"/>
</xs:element>
```

親要素

アクション

ActionButton 要素

アクションを呼び出すために使用できるボタンを定義します。アクションは、通常、UI デリゲートまたはアクション リスナーによって実装されます。

表 53. *ActionButton* の属性

属性	使用	説明	有効な値
action	必須		*string*
showIcon	オプション		*boolean*
showLabel	オプション		*boolean*

XML 表記

```
<xs:element name="ActionButton">
  <xs:sequence>
    <xs:choice>
      <xs:element ref="Layout" minOccurs="0" maxOccurs="1"/>
      <xs:element ref="Enabled" minOccurs="0" maxOccurs="1"/>
      <xs:element ref="Visible" minOccurs="0" maxOccurs="1"/>
      <xs:element ref="Required" minOccurs="0" maxOccurs="1"/>
    </xs:choice>
  </xs:sequence>
  <xs:attribute name="action" type="xs:string" use="required"/>
  <xs:attribute name="showLabel" type="xs:boolean" use="optional" default="true"/>
  <xs:attribute name="showIcon" type="xs:boolean" use="optional" default="true"/>
</xs:element>
```

親要素

PropertiesPanel、PropertiesSubPanel

子要素

Enabled、Layout、Required、Visible

関連する要素

ComboBoxControl、ExtensionObjectPanel、FieldAllocationList、ModelViewerPanel、OutputViewerPanel、SelectorPanel、StaticText、SystemControls、TabbedPanel、TextBrowserPanel

Actions 要素

XML 表記

```
<xs:element name="Actions">
  <xs:sequence minOccurs="0" maxOccurs="unbounded">
    <xs:choice>
      <xs:element ref="Action"/>
    </xs:choice>
  </xs:sequence>
</xs:element>
```

親要素

CommonObjects

子要素

アクション

AddField 要素

表 54. *AddField* の属性

属性	使用	説明	有効な値
depth	オプション		*integer*
direction	オプション		**in** **out** **both** **none** **partition**
directionRef	オプション		
fieldRef	オプション		
group	オプション		*string*
label	オプション		*string*
missingValuesRef	オプション		
name	必須		
prefix	オプション		*string*
role	オプション		**unknown** **predictedValue** **predictedDisplayValue** **probability** **residual** **standardError** **entityId** **entityAffinity** **upperConfidenceLimit** **lowerConfidenceLimit** **propensity** **value** **supplementary**
storage	オプション		**unknown** **integer** **real** **string** **date** **time** **timestamp** **list**
storageRef	オプション		
tag	オプション		*string*
targetField	オプション		*string*

表 54. *AddField* の属性 (続き)

属性	使用	説明	有効な値
type	オプション		**auto** **range** **discrete** **set** **orderedSet** **flag** **typeless** **collection** **geospatial**
typeRef	オプション		
value	オプション		*string*
valueStorage	オプション		**unknown** **integer** **real** **string** **date** **time** **timestamp** **list**

XML 表記

```
<xs:element name="AddField">
  <xs:sequence>
    <xs:choice>
      <xs:element ref="Range" minOccurs="0"/>
      <xs:element ref="Values" minOccurs="0"/>
      <xs:element ref="NumericInfo" minOccurs="0"/>
      <xs:element name="MissingValues" minOccurs="0">
        <xs:sequence>
          <xs:element ref="Values" minOccurs="0" maxOccurs="unbounded"/>
          <xs:element ref="Range" minOccurs="0"/>
        </xs:sequence>
      </xs:element>
      <xs:element name="ModelField" type="MODEL-FIELD-INFORMATION" minOccurs="0">
      </xs:element>
    </xs:choice>
  </xs:sequence>
  <xs:attribute name="name" type="FIELD-NAME" use="required"/>
  <xs:attribute name="storage" type="FIELD-STORAGE">
    <xs:enumeration value="unknown"/>
    <xs:enumeration value="integer"/>
    <xs:enumeration value="real"/>
    <xs:enumeration value="string"/>
    <xs:enumeration value="date"/>
    <xs:enumeration value="time"/>
    <xs:enumeration value="timestamp"/>
    <xs:enumeration value="list"/>
  </xs:attribute>
  <xs:attribute name="type" type="FIELD-TYPE">
    <xs:enumeration value="auto"/>
    <xs:enumeration value="range"/>
    <xs:enumeration value="discrete"/>
    <xs:enumeration value="set"/>
    <xs:enumeration value="orderedSet"/>
    <xs:enumeration value="flag"/>
    <xs:enumeration value="typeless"/>
    <xs:enumeration value="collection"/>
    <xs:enumeration value="geospatial"/>
  </xs:attribute>
  <xs:attribute name="direction" type="FIELD-DIRECTION">
    <xs:enumeration value="in"/>
```

```
    <xs:enumeration value="out"/>
    <xs:enumeration value="both"/>
    <xs:enumeration value="none"/>
    <xs:enumeration value="partition"/>
  </xs:attribute>
  <xs:attribute name="label" type="xs:string"/>
  <xs:attribute name="depth" type="xs:integer" use="optional" default="-1"/>
  <xs:attribute name="valueStorage" type="FIELD-STORAGE" use="optional">
    <xs:enumeration value="unknown"/>
    <xs:enumeration value="integer"/>
    <xs:enumeration value="real"/>
    <xs:enumeration value="string"/>
    <xs:enumeration value="date"/>
    <xs:enumeration value="time"/>
    <xs:enumeration value="timestamp"/>
    <xs:enumeration value="list"/>
  </xs:attribute>
  <xs:attribute name="fieldRef" type="EVALUATED-STRING" use="optional"/>
  <xs:attribute name="storageRef" type="EVALUATED-STRING" use="optional"/>
  <xs:attribute name="typeRef" type="EVALUATED-STRING" use="optional"/>
  <xs:attribute name="directionRef" type="EVALUATED-STRING" use="optional"/>
  <xs:attribute name="missingValuesRef" type="EVALUATED-STRING" use="optional"/>
  <xs:attribute name="role" type="MODEL-FIELD-ROLE" use="optional">
    <xs:enumeration value="unknown"/>
    <xs:enumeration value="predictedValue"/>
    <xs:enumeration value="predictedDisplayValue"/>
    <xs:enumeration value="probability"/>
    <xs:enumeration value="residual"/>
    <xs:enumeration value="standardError"/>
    <xs:enumeration value="entityId"/>
    <xs:enumeration value="entityAffinity"/>
    <xs:enumeration value="upperConfidenceLimit"/>
    <xs:enumeration value="lowerConfidenceLimit"/>
    <xs:enumeration value="propensity"/>
    <xs:enumeration value="value"/>
    <xs:enumeration value="supplementary"/>
  </xs:attribute>
  <xs:attribute name="targetField" type="xs:string" use="optional"/>
  <xs:attribute name="value" type="xs:string" use="optional"/>
  <xs:attribute name="group" type="xs:string" use="optional"/>
  <xs:attribute name="tag" type="xs:string" use="optional"/>
  <xs:attribute name="prefix" type="xs:string" use="optional"/>
</xs:element>
```

親要素

ForEach、ModelFields

子要素

MissingValues、ModelField、NumericInfo、Range、Range、Values、Values

関連する要素

ChangeField

MissingValues 要素**:**

表 *55. MissingValues* の属性

属性	使用	説明	有効な値
treatNullAsMissing	オプション		*boolean*
treatWhitespaceAsMissing	オプション		*boolean*

XML 表記

```
<xs:element name="MissingValues" minOccurs="0">
  <xs:sequence>
    <xs:element ref="Values" minOccurs="0" maxOccurs="unbounded"/>
    <xs:element ref="Range" minOccurs="0"/>
```

```
  </xs:sequence>
  <xs:attribute name="treatWhitespaceAsMissing" type="xs:boolean" use="optional" default="true"/>
  <xs:attribute name="treatNullAsMissing" type="xs:boolean" use="optional" default="true"/>
</xs:element>
```

親要素

AddField

子要素

Range、Range、Values、Values

ModelField 要素**:**

表 *56. ModelField* の属性

属性	使用	説明	有効な値
group	オプション		
role	必須		**unknown** **predictedValue** **predictedDisplayValue** **probability** **residual** **standardError** **entityId** **entityAffinity** **upperConfidenceLimit** **lowerConfidenceLimit** **propensity** **value** **supplementary**
tag	オプション		*string*
targetField	オプション		*string*
value	オプション		*string*

XML 表記

```
<xs:element name="ModelField" type="MODEL-FIELD-INFORMATION" minOccurs="0">
  <xs:attribute name="role" type="MODEL-FIELD-ROLE" use="required">
    <xs:enumeration value="unknown"/>
    <xs:enumeration value="predictedValue"/>
    <xs:enumeration value="predictedDisplayValue"/>
    <xs:enumeration value="probability"/>
    <xs:enumeration value="residual"/>
    <xs:enumeration value="standardError"/>
    <xs:enumeration value="entityId"/>
    <xs:enumeration value="entityAffinity"/>
    <xs:enumeration value="upperConfidenceLimit"/>
    <xs:enumeration value="lowerConfidenceLimit"/>
    <xs:enumeration value="propensity"/>
    <xs:enumeration value="value"/>
    <xs:enumeration value="supplementary"/>
  </xs:attribute>
  <xs:attribute name="targetField" type="xs:string"/>
  <xs:attribute name="value" type="xs:string"/>
  <xs:attribute name="group" type="MODEL-FIELD-GROUP"/>
  <xs:attribute name="tag" type="xs:string"/>
</xs:element>
```

親要素

AddField

And 要素

XML 表記

```
<xs:element name="And">
  <xs:sequence minOccurs="2" maxOccurs="unbounded">
    <xs:group ref="CONDITION-EXPRESSION">
      <xs:choice>
        <xs:element ref="Condition"/>
        <xs:element ref="And"/>
        <xs:element ref="Or"/>
        <xs:element ref="Not"/>
      </xs:choice>
    </xs:group>
  </xs:sequence>
</xs:element>
```

親要素

And、Command、Constraint、CreateContainer、CreateDocument、CreateDocumentOutput、CreateInteractiveDocumentBuilder、CreateInteractiveModelBuilder、CreateModel、CreateModelApplier、CreateModelOutput、Enabled、Not、Option、Or、Required、Run、Validation、Visible

子要素

And、Condition、Not、Or

Arg 要素

表 *57. Arg* の属性

属性	使用	説明	有効な値
property	必須		*any*

XML 表記

```
<xs:element name="Arg">
  <xs:attribute name="property" use="required"/>
</xs:element>
```

親要素

Validation

Attribute 要素

表 *58. Attribute* の属性

属性	使用	説明	有効な値
defaultValue	オプション		
description	オプション		*string*
descriptionKey	オプション		*string*
isList	オプション		*boolean*
label	必須		*string*

表 58. *Attribute* の属性 (続き)

属性	使用	説明	有効な値
labelKey	オプション		*string*
name	必須		*string*
resourceKey	オプション		*string*
type	オプション		*string*
valueType	オプション		**string** **encryptedString** **integer** **double** **boolean** **date** **enum** **structure**

XML 表記

```
<xs:element name="Attribute">
  <xs:attribute name="name" type="xs:string" use="required"/>
  <xs:attribute name="label" type="xs:string" use="required"/>
  <xs:attribute name="labelKey" type="xs:string" use="optional"/>
  <xs:attribute name="description" type="xs:string" use="optional"/>
  <xs:attribute name="descriptionKey" type="xs:string" use="optional"/>
  <xs:attribute name="resourceKey" type="xs:string" use="optional"/>
  <xs:attribute name="type" type="xs:string" use="optional"/>
  <xs:attribute name="valueType" type="ATTRIBUTE-VALUE-TYPE">
    <xs:enumeration value="string"/>
    <xs:enumeration value="encryptedString"/>
    <xs:enumeration value="integer"/>
    <xs:enumeration value="double"/>
    <xs:enumeration value="boolean"/>
    <xs:enumeration value="date"/>
    <xs:enumeration value="enum"/>
    <xs:enumeration value="structure"/>
  </xs:attribute>
  <xs:attribute name="defaultValue" type="EVALUATED-STRING" use="optional"/>
  <xs:attribute name="isList" type="xs:boolean" use="optional" default="false"/>
</xs:element>
```

親要素

Catalog、Structure

BinaryFormat 要素

XML 表記

```
<xs:element name="BinaryFormat"/>
```

親要素

FileFormatType

Catalog 要素

表 59. *Catalog* の属性

属性	使用	説明	有効な値
ID	必須		*string*

表 59. *Catalog* の属性 (続き)

属性	使用	説明	有効な値
valueColumn	必須		*integer*

XML 表記

```
<xs:element name="Catalog">
  <xs:sequence minOccurs="1" maxOccurs="unbounded">
    <xs:element ref="Attribute"/>
  </xs:sequence>
  <xs:attribute name="id" type="xs:string" use="required"/>
  <xs:attribute name="valueColumn" type="xs:integer" use="required"/>
</xs:element>
```

親要素

カタログ

子要素

属性

Catalogs 要素

XML 表記

```
<xs:element name="Catalogs">
  <xs:sequence minOccurs="0" maxOccurs="unbounded">
    <xs:choice>
      <xs:element ref="Catalog"/>
    </xs:choice>
  </xs:sequence>
</xs:element>
```

親要素

CommonObjects

子要素

カタログ

ChangeField 要素

表 60. *ChangeField* の属性

属性	使用	説明	有効な値
depth	オプション		*integer*
direction	オプション		**in** **out** **both** **none** **partition**
directionRef	オプション		
fieldRef	必須		
label	オプション		*string*
missingValuesRef	オプション		

表 60. *ChangeField* の属性 (続き)

属性	使用	説明	有効な値
name	必須		
storage	オプション		**unknown** **integer** **real** **string** **date** **time** **timestamp** **list**
storageRef	オプション		
type	オプション		**auto** **range** **discrete** **set** **orderedSet** **flag** **typeless** **collection** **geospatial**
typeRef	オプション		
valueStorage	オプション		**unknown** **integer** **real** **string** **date** **time** **timestamp** **list**

XML 表記

```
<xs:element name="ChangeField">
  <xs:sequence>
    <xs:choice>
      <xs:element ref="Range" minOccurs="0"/>
      <xs:element ref="Values" minOccurs="0"/>
      <xs:element ref="NumericInfo" minOccurs="0"/>
      <xs:element name="MissingValues" minOccurs="0">
        <xs:sequence>
          <xs:element ref="Values" minOccurs="0" maxOccurs="unbounded"/>
          <xs:element ref="Range" minOccurs="0"/>
        </xs:sequence>
      </xs:element>
      <xs:element name="ModelField" type="MODEL-FIELD-INFORMATION" minOccurs="0">
      </xs:element>
    </xs:choice>
  </xs:sequence>
  <xs:attribute name="name" type="FIELD-NAME" use="required"/>
  <xs:attribute name="storage" type="FIELD-STORAGE">
    <xs:enumeration value="unknown"/>
    <xs:enumeration value="integer"/>
    <xs:enumeration value="real"/>
    <xs:enumeration value="string"/>
    <xs:enumeration value="date"/>
```

```
    <xs:enumeration value="time"/>
    <xs:enumeration value="timestamp"/>
    <xs:enumeration value="list"/>
  </xs:attribute>
  <xs:attribute name="type" type="FIELD-TYPE">
    <xs:enumeration value="auto"/>
    <xs:enumeration value="range"/>
    <xs:enumeration value="discrete"/>
    <xs:enumeration value="set"/>
    <xs:enumeration value="orderedSet"/>
    <xs:enumeration value="flag"/>
    <xs:enumeration value="typeless"/>
    <xs:enumeration value="collection"/>
    <xs:enumeration value="geospatial"/>
  </xs:attribute>
  <xs:attribute name="direction" type="FIELD-DIRECTION">
    <xs:enumeration value="in"/>
    <xs:enumeration value="out"/>
    <xs:enumeration value="both"/>
    <xs:enumeration value="none"/>
    <xs:enumeration value="partition"/>
  </xs:attribute>
  <xs:attribute name="label" type="xs:string"/>
  <xs:attribute name="depth" type="xs:integer" use="optional" default="-1"/>
  <xs:attribute name="valueStorage" type="FIELD-STORAGE" use="optional">
    <xs:enumeration value="unknown"/>
    <xs:enumeration value="integer"/>
    <xs:enumeration value="real"/>
    <xs:enumeration value="string"/>
    <xs:enumeration value="date"/>
    <xs:enumeration value="time"/>
    <xs:enumeration value="timestamp"/>
    <xs:enumeration value="list"/>
  </xs:attribute>
  <xs:attribute name="fieldRef" type="EVALUATED-STRING" use="required"/>
  <xs:attribute name="storageRef" type="EVALUATED-STRING" use="optional"/>
  <xs:attribute name="typeRef" type="EVALUATED-STRING" use="optional"/>
  <xs:attribute name="directionRef" type="EVALUATED-STRING" use="optional"/>
  <xs:attribute name="missingValuesRef" type="EVALUATED-STRING" use="optional"/>
</xs:element>
```

親要素

ForEach、ModelFields

子要素

MissingValues、ModelField、NumericInfo、Range、Range、Values、Values

関連する要素

AddField

MissingValues 要素**:**

表 *61. MissingValues* の属性

属性	使用	説明	有効な値
treatNullAsMissing	オプション		*boolean*
treatWhitespaceAsMissing	オプション		*boolean*

XML 表記

```
<xs:element name="MissingValues" minOccurs="0">
  <xs:sequence>
    <xs:element ref="Values" minOccurs="0" maxOccurs="unbounded"/>
    <xs:element ref="Range" minOccurs="0"/>
  </xs:sequence>
  <xs:attribute name="treatWhitespaceAsMissing" type="xs:boolean" use="optional" default="true"/>
  <xs:attribute name="treatNullAsMissing" type="xs:boolean" use="optional" default="true"/>
</xs:element>
```

親要素

AddField

子要素

Range、Range、Values、Values

ModelField 要素**:**

表 *62. ModelField* の属性

属性	使用	説明	有効な値
group	オプション		
role	必須		**unknown** **predictedValue** **predictedDisplayValue** **probability** **residual** **standardError** **entityId** **entityAffinity** **upperConfidenceLimit** **lowerConfidenceLimit** **propensity** **value** **supplementary**
tag	オプション		*string*
targetField	オプション		*string*
value	オプション		*string*

XML 表記

```
<xs:element name="ModelField" type="MODEL-FIELD-INFORMATION" minOccurs="0">
  <xs:attribute name="role" type="MODEL-FIELD-ROLE" use="required">
    <xs:enumeration value="unknown"/>
    <xs:enumeration value="predictedValue"/>
    <xs:enumeration value="predictedDisplayValue"/>
    <xs:enumeration value="probability"/>
    <xs:enumeration value="residual"/>
    <xs:enumeration value="standardError"/>
    <xs:enumeration value="entityId"/>
    <xs:enumeration value="entityAffinity"/>
    <xs:enumeration value="upperConfidenceLimit"/>
    <xs:enumeration value="lowerConfidenceLimit"/>
    <xs:enumeration value="propensity"/>
    <xs:enumeration value="value"/>
    <xs:enumeration value="supplementary"/>
  </xs:attribute>
  <xs:attribute name="targetField" type="xs:string"/>
  <xs:attribute name="value" type="xs:string"/>
  <xs:attribute name="group" type="MODEL-FIELD-GROUP"/>
  <xs:attribute name="tag" type="xs:string"/>
</xs:element>
```

親要素

AddField

CheckBoxControl 要素

ブール値を変更するために使用できるチェックボックス コントロールを定義します。

表 63. *CheckBoxControl* の属性

属性	使用	説明	有効な値
description	オプション		*string*
descriptionKey	オプション		*string*
invert	オプション		*boolean*
label	オプション		*string*
labelAbove	オプション		*boolean*
labelKey	オプション		*string*
labelWidth	オプション		*positiveInteger*
mnemonic	オプション		*string*
mnemonicKey	オプション		*string*
property	必須		*string*
resourceKey	オプション		*string*
showLabel	オプション		*boolean*

XML 表記

```
<xs:element name="CheckBoxControl">
  <xs:sequence>
    <xs:choice>
      <xs:element ref="Layout" minOccurs="0" maxOccurs="1"/>
      <xs:element ref="Enabled" minOccurs="0" maxOccurs="1"/>
      <xs:element ref="Visible" minOccurs="0" maxOccurs="1"/>
      <xs:element ref="Required" minOccurs="0" maxOccurs="1"/>
    </xs:choice>
  </xs:sequence>
  <xs:attribute name="property" type="xs:string" use="required"/>
  <xs:attribute name="showLabel" type="xs:boolean" use="optional" default="true"/>
  <xs:attribute name="resourceKey" type="xs:string" use="optional"/>
  <xs:attribute name="label" type="xs:string" use="optional"/>
  <xs:attribute name="labelKey" type="xs:string" use="optional"/>
  <xs:attribute name="mnemonic" type="xs:string" use="optional"/>
  <xs:attribute name="mnemonicKey" type="xs:string" use="optional"/>
  <xs:attribute name="labelWidth" type="xs:positiveInteger" use="optional" default="1"/>
  <xs:attribute name="labelAbove" type="xs:boolean" use="optional" default="false"/>
  <xs:attribute name="description" type="xs:string" use="optional"/>
  <xs:attribute name="descriptionKey" type="xs:string" use="optional"/>
  <xs:attribute name="invert" type="xs:boolean" use="optional" default="false"/>
</xs:element>
```

親要素

PropertiesPanel、PropertiesSubPanel

子要素

Enabled、Layout、Required、Visible

関連する要素

CheckBoxGroupControl、ClientDirectoryChooserControl、ClientFileChooserControl、DBConnectionChooserControl、DBTableChooserControl、MultiFieldAllocationControl、MultiFieldChooserControl、MultiFieldSelectionTableControl、PasswordBoxControl、PropertyControl、RadioButtonGroupControl、ServerDirectoryChooserControl、ServerFileChooserControl、SingleFieldAllocationControl、SingleFieldChooserControl、SingleFieldValueChooserControl、

SpinnerControl、TableControl、TextAreaControl、TextBoxControl

CheckBoxGroupControl 要素

列挙型リストから、選択する値を指定するために使用できるチェックボックス コントロールのグループを定義します。

表 64. *CheckBoxGroupControl* の属性

属性	使用	説明	有効な値
description	オプション		*string*
descriptionKey	オプション		*string*
label	オプション		*string*
labelAbove	オプション		*boolean*
labelKey	オプション		*string*
labelWidth	オプション		*positiveInteger*
layoutByRow	オプション		*boolean*
mnemonic	オプション		*string*
mnemonicKey	オプション		*string*
property	必須		*string*
resourceKey	オプション		*string*
rows	オプション		*positiveInteger*
showLabel	オプション		*boolean*
useSubPanel	オプション		*boolean*

XML 表記

```
<xs:element name="CheckBoxGroupControl">
  <xs:sequence>
    <xs:choice>
      <xs:element ref="Layout" minOccurs="0" maxOccurs="1"/>
      <xs:element ref="Enabled" minOccurs="0" maxOccurs="1"/>
      <xs:element ref="Visible" minOccurs="0" maxOccurs="1"/>
      <xs:element ref="Required" minOccurs="0" maxOccurs="1"/>
    </xs:choice>
  </xs:sequence>
  <xs:attribute name="property" type="xs:string" use="required"/>
  <xs:attribute name="showLabel" type="xs:boolean" use="optional" default="true"/>
  <xs:attribute name="resourceKey" type="xs:string" use="optional"/>
  <xs:attribute name="label" type="xs:string" use="optional"/>
  <xs:attribute name="labelKey" type="xs:string" use="optional"/>
  <xs:attribute name="mnemonic" type="xs:string" use="optional"/>
  <xs:attribute name="mnemonicKey" type="xs:string" use="optional"/>
  <xs:attribute name="labelWidth" type="xs:positiveInteger" use="optional" default="1"/>
  <xs:attribute name="labelAbove" type="xs:boolean" use="optional" default="false"/>
  <xs:attribute name="description" type="xs:string" use="optional"/>
  <xs:attribute name="descriptionKey" type="xs:string" use="optional"/>
  <xs:attribute name="rows" type="xs:positiveInteger" use="optional" default="1"/>
  <xs:attribute name="layoutByRow" type="xs:boolean" use="optional" default="true"/>
  <xs:attribute name="useSubPanel" type="xs:boolean" use="optional" default="true"/>
</xs:element>
```

親要素

PropertiesPanel、PropertiesSubPanel

子要素

Enabled、Layout、Required、Visible

関連する要素

CheckBoxControl、ClientDirectoryChooserControl、ClientFileChooserControl、DBConnectionChooserControl、DBTableChooserControl、MultiFieldAllocationControl、MultiFieldChooserControl、MultiFieldSelectionTableControl、PasswordBoxControl、PropertyControl、RadioButtonGroupControl、ServerDirectoryChooserControl、ServerFileChooserControl、SingleFieldAllocationControl、SingleFieldChooserControl、SingleFieldValueChooserControl、SpinnerControl、TableControl、TextAreaControl、TextBoxControl

ClientDirectoryChooserControl 要素

クライアント上のディレクトリーを選択するために使用できるコントロールを定義します。

表 *65. ClientDirectoryChooserControl* の属性

属性	使用	説明	有効な値
description	オプション		*string*
descriptionKey	オプション		*string*
label	オプション		*string*
labelAbove	オプション		*boolean*
labelKey	オプション		*string*
labelWidth	オプション		*positiveInteger*
mnemonic	オプション		*string*
mnemonicKey	オプション		*string*
最頻値	必須		**open** **save** **import** **export**
property	必須		*string*
resourceKey	オプション		*string*
showLabel	オプション		*boolean*

XML 表記

```
<xs:element name="ClientDirectoryChooserControl">
  <xs:sequence>
    <xs:choice>
      <xs:element ref="Layout" minOccurs="0" maxOccurs="1"/>
      <xs:element ref="Enabled" minOccurs="0" maxOccurs="1"/>
      <xs:element ref="Visible" minOccurs="0" maxOccurs="1"/>
      <xs:element ref="Required" minOccurs="0" maxOccurs="1"/>
    </xs:choice>
  </xs:sequence>
  <xs:attribute name="property" type="xs:string" use="required"/>
  <xs:attribute name="showLabel" type="xs:boolean" use="optional" default="true"/>
  <xs:attribute name="resourceKey" type="xs:string" use="optional"/>
  <xs:attribute name="label" type="xs:string" use="optional"/>
  <xs:attribute name="labelKey" type="xs:string" use="optional"/>
  <xs:attribute name="mnemonic" type="xs:string" use="optional"/>
  <xs:attribute name="mnemonicKey" type="xs:string" use="optional"/>
  <xs:attribute name="labelWidth" type="xs:positiveInteger" use="optional" default="1"/>
  <xs:attribute name="labelAbove" type="xs:boolean" use="optional" default="false"/>
  <xs:attribute name="description" type="xs:string" use="optional"/>
  <xs:attribute name="descriptionKey" type="xs:string" use="optional"/>
  <xs:attribute name="mode" type="FILE-CHOOSER-MODE" use="required">
    <xs:enumeration value="open"/>
    <xs:enumeration value="save"/>
```

```
    <xs:enumeration value="import"/>
    <xs:enumeration value="export"/>
  </xs:attribute>
</xs:element>
```

親要素

PropertiesPanel、PropertiesSubPanel

子要素

Enabled、Layout、Required、Visible

関連する要素

CheckBoxControl、CheckBoxGroupControl、ClientFileChooserControl、DBConnectionChooserControl、DBTableChooserControl、MultiFieldAllocationControl、MultiFieldChooserControl、MultiFieldSelectionTableControl、PasswordBoxControl、PropertyControl、RadioButtonGroupControl、ServerDirectoryChooserControl、ServerFileChooserControl、SingleFieldAllocationControl、SingleFieldChooserControl、SingleFieldValueChooserControl、SpinnerControl、TableControl、TextAreaControl、TextBoxControl

ClientFileChooserControl 要素

クライアント上のファイルを選択するために使用できるコントロールを定義します。

表 66. *ClientFileChooserControl* の属性

属性	使用	説明	有効な値
description	オプション		*string*
descriptionKey	オプション		*string*
label	オプション		*string*
labelAbove	オプション		*boolean*
labelKey	オプション		*string*
labelWidth	オプション		*positiveInteger*
mnemonic	オプション		*string*
mnemonicKey	オプション		*string*
最頻値	必須		**open** **save** **import** **export**
property	必須		*string*
resourceKey	オプション		*string*
showLabel	オプション		*boolean*

XML 表記

```
<xs:element name="ClientFileChooserControl">
  <xs:sequence>
    <xs:choice>
      <xs:element ref="Layout" minOccurs="0" maxOccurs="1"/>
      <xs:element ref="Enabled" minOccurs="0" maxOccurs="1"/>
      <xs:element ref="Visible" minOccurs="0" maxOccurs="1"/>
      <xs:element ref="Required" minOccurs="0" maxOccurs="1"/>
    </xs:choice>
```

```
  </xs:sequence>
  <xs:attribute name="property" type="xs:string" use="required"/>
  <xs:attribute name="showLabel" type="xs:boolean" use="optional" default="true"/>
  <xs:attribute name="resourceKey" type="xs:string" use="optional"/>
  <xs:attribute name="label" type="xs:string" use="optional"/>
  <xs:attribute name="labelKey" type="xs:string" use="optional"/>
  <xs:attribute name="mnemonic" type="xs:string" use="optional"/>
  <xs:attribute name="mnemonicKey" type="xs:string" use="optional"/>
  <xs:attribute name="labelWidth" type="xs:positiveInteger" use="optional" default="1"/>
  <xs:attribute name="labelAbove" type="xs:boolean" use="optional" default="false"/>
  <xs:attribute name="description" type="xs:string" use="optional"/>
  <xs:attribute name="descriptionKey" type="xs:string" use="optional"/>
  <xs:attribute name="mode" type="FILE-CHOOSER-MODE" use="required">
    <xs:enumeration value="open"/>
    <xs:enumeration value="save"/>
    <xs:enumeration value="import"/>
    <xs:enumeration value="export"/>
  </xs:attribute>
</xs:element>
```

親要素

PropertiesPanel、PropertiesSubPanel

子要素

Enabled、Layout、Required、Visible

関連する要素

CheckBoxControl、CheckBoxGroupControl、ClientDirectoryChooserControl、DBConnectionChooserControl、DBTableChooserControl、MultiFieldAllocationControl、MultiFieldChooserControl、MultiFieldSelectionTableControl、PasswordBoxControl、PropertyControl、RadioButtonGroupControl、ServerDirectoryChooserControl、ServerFileChooserControl、SingleFieldAllocationControl、SingleFieldChooserControl、SingleFieldValueChooserControl、SpinnerControl、TableControl、TextAreaControl、TextBoxControl

ComboBoxControl 要素

表 *67. ComboBoxControl* の属性

属性	使用	説明	有効な値
description	オプション		*string*
descriptionKey	オプション		*string*
label	オプション		*string*
labelAbove	オプション		*boolean*
labelKey	オプション		*string*
labelWidth	オプション		*positiveInteger*
mnemonic	オプション		*string*
mnemonicKey	オプション		*string*
property	必須		*string*
resourceKey	オプション		*string*
showLabel	オプション		*boolean*

XML 表記

```
<xs:element name="ComboBoxControl" type="CONTROLLER">
  <xs:sequence>
    <xs:choice>
```

```
      <xs:element ref="Layout" minOccurs="0" maxOccurs="1"/>
      <xs:element ref="Enabled" minOccurs="0" maxOccurs="1"/>
      <xs:element ref="Visible" minOccurs="0" maxOccurs="1"/>
      <xs:element ref="Required" minOccurs="0" maxOccurs="1"/>
    </xs:choice>
  </xs:sequence>
  <xs:attribute name="property" type="xs:string" use="required"/>
  <xs:attribute name="showLabel" type="xs:boolean" use="optional" default="true"/>
  <xs:attribute name="resourceKey" type="xs:string" use="optional"/>
  <xs:attribute name="label" type="xs:string" use="optional"/>
  <xs:attribute name="labelKey" type="xs:string" use="optional"/>
  <xs:attribute name="mnemonic" type="xs:string" use="optional"/>
  <xs:attribute name="mnemonicKey" type="xs:string" use="optional"/>
  <xs:attribute name="labelWidth" type="xs:positiveInteger" use="optional" default="1"/>
  <xs:attribute name="labelAbove" type="xs:boolean" use="optional" default="false"/>
  <xs:attribute name="description" type="xs:string" use="optional"/>
  <xs:attribute name="descriptionKey" type="xs:string" use="optional"/>
</xs:element>
```

表 *68.* 拡張タイプ

データ型	説明
ItemChooserControl	

親要素

PropertiesPanel、PropertiesSubPanel

子要素

Enabled、Layout、Required、Visible

関連する要素

ActionButton、ExtensionObjectPanel、FieldAllocationList、ModelViewerPanel、OutputViewerPanel、SelectorPanel、StaticText、SystemControls、TabbedPanel、TextBrowserPanel

Command 要素

表 *69. Command* の属性

属性	使用	説明	有効な値
path	必須		

XML 表記

```
<xs:element name="Command">
  <xs:sequence>
    <xs:group ref="CONDITION-EXPRESSION" minOccurs="0">
      <xs:choice>
        <xs:element ref="Condition"/>
        <xs:element ref="And"/>
        <xs:element ref="Or"/>
        <xs:element ref="Not"/>
      </xs:choice>
    </xs:group>
  </xs:sequence>
  <xs:attribute name="path" type="EVALUATED-STRING" use="required"/>
</xs:element>
```

親要素

Run

子要素

And、Condition、Not、Or

CommonObjects 要素

拡張に対してグローバルな定義の場所を提供します。

表 70. *CommonObjects* の属性

属性	使用	説明	有効な値
extensionDelegate	オプション		*string*
extensionListenerClass	オプション		*string*

XML 表記

```
<xs:element name="CommonObjects">
  <xs:all>
    <xs:element ref="PropertyTypes" minOccurs="0"/>
    <xs:element ref="PropertySets" minOccurs="0"/>
    <xs:element ref="FileFormatTypes" minOccurs="0"/>
    <xs:element ref="ContainerTypes" minOccurs="0"/>
    <xs:element ref="Actions" minOccurs="0"/>
    <xs:element ref="Catalogs" minOccurs="0"/>
  </xs:all>
  <xs:attribute name="extensionDelegate" type="xs:string" use="optional"/>
  <xs:attribute name="extensionListenerClass" type="xs:string" use="optional"/>
</xs:element>
```

親要素

Extension

子要素

Actions、Catalogs、ContainerTypes、FileFormatTypes、PropertySets、PropertyTypes

Condition 要素

表 71. *Condition* の属性

属性	使用	説明	有効な値
container	オプション		*string*
control	オプション		*string*
control2	オプション		*string*
expression	オプション		
listMode	オプション		**all** **any**

表 71. *Condition* の属性 (続き)

属性	使用	説明	有効な値
op	必須		**equals** **notEquals** **isEmpty** **isNotEmpty** **lessThan** **lessOrEquals** **greaterThan** **greaterOrEquals** **equalsIgnoreCase** **isSubstring** **startsWith** **endsWith** **startsWithIgnoreCase** **endsWithIgnoreCase** **isSubstring** **hasSubstring** **isSubstringIgnoreCase** **hasSubstringIgnoreCase** **in** **countEquals** **countLessThan** **countLessOrEquals** **countGreaterThan** **countGreaterOrEquals** **contains** **storageEquals** **typeEquals** **directionEquals** **isMeasureDiscrete** **isMeasureContinuous** **isMeasureCollection** **isMeasureGeospatial** **isMeasureTypeless** **isMeasureUnknown** **isStorageString** **isStorageNumeric** **isStorageDatetime** **isStorageList** **isStorageUnknown** **isModelOutput** **modelOutputRoleEquals** **modelOutputTargetFieldEquals** **modelOutputHasValue** **modelOutputTagEquals** **enumRestriction**
property	オプション		*string*
property2	オプション		*string*

表 71. *Condition* の属性 (続き)

属性	使用	説明	有効な値
value	オプション		

XML 表記

```
<xs:element name="Condition">
  <xs:attribute name="expression" type="EVALUATED-STRING" use="optional"/>
  <xs:attribute name="control" type="xs:string" use="optional"/>
  <xs:attribute name="property" type="xs:string" use="optional"/>
  <xs:attribute name="container" type="xs:string" use="optional"/>
  <xs:attribute name="op" type="CONDITION-TEST" use="required">
    <xs:enumeration value="equals"/>
    <xs:enumeration value="notEquals"/>
    <xs:enumeration value="isEmpty"/>
    <xs:enumeration value="isNotEmpty"/>
    <xs:enumeration value="lessThan"/>
    <xs:enumeration value="lessOrEquals"/>
    <xs:enumeration value="greaterThan"/>
    <xs:enumeration value="greaterOrEquals"/>
    <xs:enumeration value="equalsIgnoreCase"/>
    <xs:enumeration value="isSubstring"/>
    <xs:enumeration value="startsWith"/>
    <xs:enumeration value="endsWith"/>
    <xs:enumeration value="startsWithIgnoreCase"/>
    <xs:enumeration value="endsWithIgnoreCase"/>
    <xs:enumeration value="isSubstring"/>
    <xs:enumeration value="hasSubstring"/>
    <xs:enumeration value="isSubstringIgnoreCase"/>
    <xs:enumeration value="hasSubstringIgnoreCase"/>
    <xs:enumeration value="in"/>
    <xs:enumeration value="countEquals"/>
    <xs:enumeration value="countLessThan"/>
    <xs:enumeration value="countLessOrEquals"/>
    <xs:enumeration value="countGreaterThan"/>
    <xs:enumeration value="countGreaterOrEquals"/>
    <xs:enumeration value="contains"/>
    <xs:enumeration value="storageEquals"/>
    <xs:enumeration value="typeEquals"/>
    <xs:enumeration value="directionEquals"/>
    <xs:enumeration value="isMeasureDiscrete"/>
    <xs:enumeration value="isMeasureContinuous"/>
    <xs:enumeration value="isMeasureCollection"/>
    <xs:enumeration value="isMeasureGeospatial"/>
    <xs:enumeration value="isMeasureTypeless"/>
    <xs:enumeration value="isMeasureUnknown"/>
    <xs:enumeration value="isStorageString"/>
    <xs:enumeration value="isStorageNumeric"/>
    <xs:enumeration value="isStorageDatetime"/>
    <xs:enumeration value="isStorageList"/>
    <xs:enumeration value="isStorageUnknown"/>
    <xs:enumeration value="isModelOutput"/>
    <xs:enumeration value="modelOutputRoleEquals"/>
    <xs:enumeration value="modelOutputTargetFieldEquals"/>
    <xs:enumeration value="modelOutputHasValue"/>
    <xs:enumeration value="modelOutputTagEquals"/>
    <xs:enumeration value="enumRestriction"/>
  </xs:attribute>
  <xs:attribute name="control2" type="xs:string" use="optional"/>
  <xs:attribute name="property2" type="xs:string" use="optional"/>
  <xs:attribute name="value" type="EVALUATED-STRING" use="optional"/>
  <xs:attribute name="listMode" use="optional" default="all">
    <xs:simpleType>
      <xs:restriction base="xs:string">
        <xs:enumeration value="all"/>
        <xs:enumeration value="any"/>
      </xs:restriction>
    </xs:simpleType>
  </xs:attribute>
</xs:element>
```

親要素

And、Command、Constraint、CreateContainer、CreateDocument、CreateDocumentOutput、CreateInteractiveDocumentBuilder、CreateInteractiveModelBuilder、CreateModel、

CreateModelApplier、CreateModelOutput、Enabled、ExpertSettings、Not、Option、Or、Required、Run、Validation、Visible

Constraint 要素

表 72. *Constraint* の属性

属性	使用	説明	有効な値
property	必須		*string*
singleSelection	オプション		*boolean*

XML 表記

```
<xs:element name="Constraint">
  <xs:sequence>
    <xs:group ref="CONDITION-EXPRESSION">
      <xs:choice>
        <xs:element ref="Condition"/>
        <xs:element ref="And"/>
        <xs:element ref="Or"/>
        <xs:element ref="Not"/>
      </xs:choice>
    </xs:group>
  </xs:sequence>
  <xs:attribute name="property" type="xs:string" use="required"/>
  <xs:attribute name="singleSelection" type="xs:boolean" use="optional" default="false"/>
</xs:element>
```

親要素

AutoModeling

子要素

And、Condition、Not、Or

Constructors 要素

XML 表記

```
<xs:element name="Constructors">
  <xs:sequence minOccurs="0" maxOccurs="unbounded">
    <xs:choice>
      <xs:element ref="CreateModelOutput"/>
      <xs:element ref="CreateDocumentOutput"/>
      <xs:element ref="CreateInteractiveModelBuilder"/>
      <xs:element ref="CreateInteractiveDocumentBuilder"/>
      <xs:element ref="CreateModelApplier"/>
    </xs:choice>
  </xs:sequence>
</xs:element>
```

親要素

DocumentOutput、Execution、InteractiveDocumentBuilder、InteractiveModelBuilder、ModelOutput、Node

子要素

CreateDocumentOutput、CreateInteractiveDocumentBuilder、CreateInteractiveModelBuilder、CreateModelApplier、CreateModelOutput

Container 要素

表 73. *Container* の属性

属性	使用	説明	有効な値
content	オプション		*string*
format	オプション		**utf8** **binary**
name	必須		*string*
type	オプション		*string*

XML 表記

```
<xs:element name="Container">
  <xs:attribute name="name" type="xs:string" use="required"/>
  <xs:attribute name="format" use="optional">
    <xs:simpleType>
      <xs:restriction base="xs:string">
        <xs:enumeration value="utf8"/>
        <xs:enumeration value="binary"/>
      </xs:restriction>
    </xs:simpleType>
  </xs:attribute>
  <xs:attribute name="type" type="xs:string" use="optional"/>
  <xs:attribute name="content" type="xs:string" use="optional"/>
</xs:element>
```

親要素

Containers、Containers、Containers、Containers、Containers

ContainerFile 要素

表 74. *ContainerFile* の属性

属性	使用	説明	有効な値
container	オプション		
containerType	オプション		
path	必須		

XML 表記

```
<xs:element name="ContainerFile" type="SERVER-CONTAINER-FILE">
  <xs:attribute name="path" type="EVALUATED-STRING" use="required"/>
  <xs:attribute name="container" type="EVALUATED-STRING" use="optional"/>
  <xs:attribute name="containerType" type="EVALUATED-STRING" use="optional"/>
</xs:element>
```

親要素

InputFiles、OutputFiles

ContainerType 要素

新規コンテナー タイプを定義します。

表 75. *ContainerType* の属性

属性	使用	説明	有効な値
content	オプション		*string*

表 75. *ContainerType* の属性 (続き)

属性	使用	説明	有効な値
format	必須		**utf8** **binary**
ID	必須		*string*
type	オプション		*string*

XML 表記

```
<xs:element name="ContainerType">
  <xs:attribute name="id" type="xs:string" use="required"/>
  <xs:attribute name="format" use="required">
    <xs:simpleType>
      <xs:restriction base="xs:string">
        <xs:enumeration value="utf8"/>
        <xs:enumeration value="binary"/>
      </xs:restriction>
    </xs:simpleType>
  </xs:attribute>
  <xs:attribute name="content" type="xs:string" use="optional"/>
  <xs:attribute name="type" type="xs:string" use="optional"/>
</xs:element>
```

親要素

ContainerTypes

関連する要素

DocumentType、ModelType

ContainerTypes 要素

XML 表記

```
<xs:element name="ContainerTypes">
  <xs:sequence minOccurs="0" maxOccurs="unbounded">
    <xs:choice>
      <xs:element ref="DocumentType"/>
      <xs:element ref="ModelType"/>
      <xs:element ref="ContainerType"/>
    </xs:choice>
  </xs:sequence>
</xs:element>
```

親要素

CommonObjects

子要素

ContainerType、DocumentType、ModelType

Controls 要素

メニューやツールバー項目などの、ユーザー インターフェースに追加できる具体的なコントロールを定義します。

XML 表記

```
<xs:element name="Controls">
  <xs:sequence minOccurs="0" maxOccurs="unbounded">
    <xs:choice>
      <xs:element ref="Menu"/>
      <xs:element ref="MenuItem"/>
      <xs:element ref="ToolbarItem"/>
    </xs:choice>
  </xs:sequence>
</xs:element>
```

親要素

UserInterface

子要素

Menu、MenuItem、ToolbarItem

CreateContainer 要素

表 *76. CreateContainer* の属性

属性	使用	説明	有効な値
sourceFile	必須		*string*
target	必須		*string*
type	オプション		*string*

XML 表記

```
<xs:element name="CreateContainer">
  <xs:group ref="CONDITION-EXPRESSION" minOccurs="0">
    <xs:choice>
      <xs:element ref="Condition"/>
      <xs:element ref="And"/>
      <xs:element ref="Or"/>
      <xs:element ref="Not"/>
    </xs:choice>
  </xs:group>
  <xs:attribute name="sourceFile" type="xs:string" use="required"/>
  <xs:attribute name="target" type="xs:string" use="required"/>
  <xs:attribute name="type" type="xs:string" use="optional"/>
</xs:element>
```

親要素

CreateDocumentOutput、CreateInteractiveDocumentBuilder、CreateInteractiveModelBuilder、CreateModelApplier、CreateModelOutput

子要素

And、Condition、Not、Or

関連する要素

CreateDocument、CreateModel

CreateDocument 要素

表 77. *CreateDocument* の属性

属性	使用	説明	有効な値
sourceFile	必須		*string*
target	必須		*string*
type	オプション		*string*

XML 表記

```
<xs:element name="CreateDocument">
  <xs:group ref="CONDITION-EXPRESSION" minOccurs="0">
    <xs:choice>
      <xs:element ref="Condition"/>
      <xs:element ref="And"/>
      <xs:element ref="Or"/>
      <xs:element ref="Not"/>
    </xs:choice>
  </xs:group>
  <xs:attribute name="sourceFile" type="xs:string" use="required"/>
  <xs:attribute name="target" type="xs:string" use="required"/>
  <xs:attribute name="type" type="xs:string" use="optional"/>
</xs:element>
```

親要素

CreateDocumentOutput、CreateInteractiveDocumentBuilder、CreateInteractiveModelBuilder、CreateModelApplier、CreateModelOutput

子要素

And、Condition、Not、Or

関連する要素

CreateContainer、CreateModel

CreateDocumentOutput 要素

表 78. *CreateDocumentOutput* の属性

属性	使用	説明	有効な値
type	必須		*string*

XML 表記

```
<xs:element name="CreateDocumentOutput">
  <xs:sequence>
    <xs:group ref="CONDITION-EXPRESSION" minOccurs="0">
      <xs:choice>
        <xs:element ref="Condition"/>
        <xs:element ref="And"/>
        <xs:element ref="Or"/>
        <xs:element ref="Not"/>
      </xs:choice>
    </xs:group>
    <xs:sequence minOccurs="0" maxOccurs="unbounded">
      <xs:choice>
        <xs:element ref="SetProperty"/>
        <xs:element ref="SetContainer"/>
        <xs:element ref="CreateModel"/>
        <xs:element ref="CreateDocument"/>
        <xs:element ref="CreateContainer"/>
      </xs:choice>
```

```
    </xs:sequence>
  </xs:sequence>
  <xs:attribute name="type" type="xs:string" use="required"/>
</xs:element>
```

親要素

コンストラクター

子要素

And、Condition、CreateContainer、CreateDocument、CreateModel、Not、Or、SetContainer、SetProperty

関連する要素

CreateInteractiveDocumentBuilder、CreateInteractiveModelBuilder、CreateModelApplier、CreateModelOutput

CreateInteractiveDocumentBuilder 要素

表 79. *CreateInteractiveDocumentBuilder* の属性

属性	使用	説明	有効な値
type	必須		*string*

XML 表記

```
<xs:element name="CreateInteractiveDocumentBuilder">
  <xs:sequence>
    <xs:group ref="CONDITION-EXPRESSION" minOccurs="0">
      <xs:choice>
        <xs:element ref="Condition"/>
        <xs:element ref="And"/>
        <xs:element ref="Or"/>
        <xs:element ref="Not"/>
      </xs:choice>
    </xs:group>
    <xs:sequence minOccurs="0" maxOccurs="unbounded">
      <xs:choice>
        <xs:element ref="SetProperty"/>
        <xs:element ref="SetContainer"/>
        <xs:element ref="CreateModel"/>
        <xs:element ref="CreateDocument"/>
        <xs:element ref="CreateContainer"/>
      </xs:choice>
    </xs:sequence>
  </xs:sequence>
  <xs:attribute name="type" type="xs:string" use="required"/>
</xs:element>
```

親要素

コンストラクター

子要素

And、Condition、CreateContainer、CreateDocument、CreateModel、Not、Or、SetContainer、SetProperty

関連する要素

CreateDocumentOutput、CreateInteractiveModelBuilder、CreateModelApplier、CreateModelOutput

CreateInteractiveModelBuilder 要素

表 *80. CreateInteractiveModelBuilder* の属性

属性	使用	説明	有効な値
type	必須		*string*

XML 表記

```
<xs:element name="CreateInteractiveModelBuilder">
  <xs:sequence>
    <xs:group ref="CONDITION-EXPRESSION" minOccurs="0">
      <xs:choice>
        <xs:element ref="Condition"/>
        <xs:element ref="And"/>
        <xs:element ref="Or"/>
        <xs:element ref="Not"/>
      </xs:choice>
    </xs:group>
    <xs:sequence minOccurs="0" maxOccurs="unbounded">
      <xs:choice>
        <xs:element ref="SetProperty"/>
        <xs:element ref="SetContainer"/>
        <xs:element ref="CreateModel"/>
        <xs:element ref="CreateDocument"/>
        <xs:element ref="CreateContainer"/>
      </xs:choice>
    </xs:sequence>
  </xs:sequence>
  <xs:attribute name="type" type="xs:string" use="required"/>
</xs:element>
```

親要素

コンストラクター

子要素

And、Condition、CreateContainer、CreateDocument、CreateModel、Not、Or、SetContainer、SetProperty

関連する要素

CreateDocumentOutput、CreateInteractiveDocumentBuilder、CreateModelApplier、CreateModelOutput

CreateModel 要素

表 *81. CreateModel* の属性

属性	使用	説明	有効な値
signatureFile	オプション		*string*
sourceFile	必須		*string*
target	必須		*string*
type	オプション		*string*

XML 表記

```
<xs:element name="CreateModel">
  <xs:group ref="CONDITION-EXPRESSION" minOccurs="0">
    <xs:choice>
      <xs:element ref="Condition"/>
      <xs:element ref="And"/>
      <xs:element ref="Or"/>
      <xs:element ref="Not"/>
    </xs:choice>
```

```
  </xs:group>
  <xs:attribute name="sourceFile" type="xs:string" use="required"/>
  <xs:attribute name="target" type="xs:string" use="required"/>
  <xs:attribute name="type" type="xs:string" use="optional"/>
  <xs:sequence>
    <xs:element name="ModelDetail" maxOccurs="unbounded">
    </xs:element>
  </xs:sequence>
  <xs:attribute name="signatureFile" type="xs:string" use="optional"/>
</xs:element>
```

親要素

CreateDocumentOutput、CreateInteractiveDocumentBuilder、CreateInteractiveModelBuilder、CreateModelApplier、CreateModelOutput

子要素

And、Condition、ModelDetail、Not、Or

関連する要素

CreateContainer、CreateDocument

ModelDetail 要素**:**

表 *82. ModelDetail* の属性

属性	使用	説明	有効な値
アルゴリズム	必須		*string*

XML 表記

```
<xs:element name="ModelDetail" maxOccurs="unbounded">
  <xs:attribute name="algorithm" type="xs:string" use="required"/>
</xs:element>
```

親要素

CreateModel

CreateModelApplier 要素

表 *83. CreateModelApplier* の属性

属性	使用	説明	有効な値
type	必須		*string*

XML 表記

```
<xs:element name="CreateModelApplier">
  <xs:sequence>
    <xs:group ref="CONDITION-EXPRESSION" minOccurs="0">
      <xs:choice>
        <xs:element ref="Condition"/>
        <xs:element ref="And"/>
        <xs:element ref="Or"/>
        <xs:element ref="Not"/>
      </xs:choice>
    </xs:group>
    <xs:sequence minOccurs="0" maxOccurs="unbounded">
      <xs:choice>
        <xs:element ref="SetProperty"/>
        <xs:element ref="SetContainer"/>
        <xs:element ref="CreateModel"/>
        <xs:element ref="CreateDocument"/>
```

```
        <xs:element ref="CreateContainer"/>
      </xs:choice>
    </xs:sequence>
  </xs:sequence>
  <xs:attribute name="type" type="xs:string" use="required"/>
</xs:element>
```

親要素

コンストラクター

子要素

And、Condition、CreateContainer、CreateDocument、CreateModel、Not、Or、SetContainer、SetProperty

関連する要素

CreateDocumentOutput、CreateInteractiveDocumentBuilder、CreateInteractiveModelBuilder、CreateModelOutput

CreateModelOutput 要素

表 *84. CreateModelOutput* の属性

属性	使用	説明	有効な値
type	必須		*string*

XML 表記

```
<xs:element name="CreateModelOutput">
  <xs:sequence>
    <xs:group ref="CONDITION-EXPRESSION" minOccurs="0">
      <xs:choice>
        <xs:element ref="Condition"/>
        <xs:element ref="And"/>
        <xs:element ref="Or"/>
        <xs:element ref="Not"/>
      </xs:choice>
    </xs:group>
    <xs:sequence minOccurs="0" maxOccurs="unbounded">
      <xs:choice>
        <xs:element ref="SetProperty"/>
        <xs:element ref="SetContainer"/>
        <xs:element ref="CreateModel"/>
        <xs:element ref="CreateDocument"/>
        <xs:element ref="CreateContainer"/>
      </xs:choice>
    </xs:sequence>
  </xs:sequence>
  <xs:attribute name="type" type="xs:string" use="required"/>
</xs:element>
```

親要素

コンストラクター

子要素

And、Condition、CreateContainer、CreateDocument、CreateModel、Not、Or、SetContainer、SetProperty

関連する要素

CreateDocumentOutput、CreateInteractiveDocumentBuilder、CreateInteractiveModelBuilder、CreateModelApplier

DBConnectionChooserControl 要素

データベース接続を選択するために使用できるコントロールを定義します。

表 *85. DBConnectionChooserControl* の属性

属性	使用	説明	有効な値
description	オプション		*string*
descriptionKey	オプション		*string*
label	オプション		*string*
labelAbove	オプション		*boolean*
labelKey	オプション		*string*
labelWidth	オプション		*positiveInteger*
mnemonic	オプション		*string*
mnemonicKey	オプション		*string*
property	必須		*string*
resourceKey	オプション		*string*
showLabel	オプション		*boolean*

XML 表記

```
<xs:element name="DBConnectionChooserControl">
  <xs:sequence>
    <xs:choice>
      <xs:element ref="Layout" minOccurs="0" maxOccurs="1"/>
      <xs:element ref="Enabled" minOccurs="0" maxOccurs="1"/>
      <xs:element ref="Visible" minOccurs="0" maxOccurs="1"/>
      <xs:element ref="Required" minOccurs="0" maxOccurs="1"/>
    </xs:choice>
  </xs:sequence>
  <xs:attribute name="property" type="xs:string" use="required"/>
  <xs:attribute name="showLabel" type="xs:boolean" use="optional" default="true"/>
  <xs:attribute name="resourceKey" type="xs:string" use="optional"/>
  <xs:attribute name="label" type="xs:string" use="optional"/>
  <xs:attribute name="labelKey" type="xs:string" use="optional"/>
  <xs:attribute name="mnemonic" type="xs:string" use="optional"/>
  <xs:attribute name="mnemonicKey" type="xs:string" use="optional"/>
  <xs:attribute name="labelWidth" type="xs:positiveInteger" use="optional" default="1"/>
  <xs:attribute name="labelAbove" type="xs:boolean" use="optional" default="false"/>
  <xs:attribute name="description" type="xs:string" use="optional"/>
  <xs:attribute name="descriptionKey" type="xs:string" use="optional"/>
</xs:element>
```

親要素

PropertiesPanel、PropertiesSubPanel

子要素

Enabled、Layout、Required、Visible

関連する要素

CheckBoxControl、CheckBoxGroupControl、ClientDirectoryChooserControl、ClientFileChooserControl、DBTableChooserControl、MultiFieldAllocationControl、MultiFieldChooserControl、

MultiFieldSelectionTableControl、PasswordBoxControl、PropertyControl、RadioButtonGroupControl、ServerDirectoryChooserControl、ServerFileChooserControl、SingleFieldAllocationControl、SingleFieldChooserControl、SingleFieldValueChooserControl、SpinnerControl、TableControl、TextAreaControl、TextBoxControl

DBTableChooserControl 要素

データベース テーブルを選択するために使用できるコントロールを定義します。

表 *86. DBTableChooserControl* の属性

属性	使用	説明	有効な値
connectionProperty	必須		*string*
description	オプション		*string*
descriptionKey	オプション		*string*
label	オプション		*string*
labelAbove	オプション		*boolean*
labelKey	オプション		*string*
labelWidth	オプション		*positiveInteger*
mnemonic	オプション		*string*
mnemonicKey	オプション		*string*
property	必須		*string*
resourceKey	オプション		*string*
showLabel	オプション		*boolean*

XML 表記

```
<xs:element name="DBTableChooserControl">
  <xs:sequence>
    <xs:choice>
      <xs:element ref="Layout" minOccurs="0" maxOccurs="1"/>
      <xs:element ref="Enabled" minOccurs="0" maxOccurs="1"/>
      <xs:element ref="Visible" minOccurs="0" maxOccurs="1"/>
      <xs:element ref="Required" minOccurs="0" maxOccurs="1"/>
    </xs:choice>
  </xs:sequence>
  <xs:attribute name="property" type="xs:string" use="required"/>
  <xs:attribute name="showLabel" type="xs:boolean" use="optional" default="true"/>
  <xs:attribute name="resourceKey" type="xs:string" use="optional"/>
  <xs:attribute name="label" type="xs:string" use="optional"/>
  <xs:attribute name="labelKey" type="xs:string" use="optional"/>
  <xs:attribute name="mnemonic" type="xs:string" use="optional"/>
  <xs:attribute name="mnemonicKey" type="xs:string" use="optional"/>
  <xs:attribute name="labelWidth" type="xs:positiveInteger" use="optional" default="1"/>
  <xs:attribute name="labelAbove" type="xs:boolean" use="optional" default="false"/>
  <xs:attribute name="description" type="xs:string" use="optional"/>
  <xs:attribute name="descriptionKey" type="xs:string" use="optional"/>
  <xs:attribute name="connectionProperty" type="xs:string" use="required"/>
</xs:element>
```

親要素

PropertiesPanel、PropertiesSubPanel

子要素

Enabled、Layout、Required、Visible

関連する要素

CheckBoxControl、CheckBoxGroupControl、ClientDirectoryChooserControl、ClientFileChooserControl、DBConnectionChooserControl、MultiFieldAllocationControl、MultiFieldChooserControl、MultiFieldSelectionTableControl、PasswordBoxControl、PropertyControl、RadioButtonGroupControl、ServerDirectoryChooserControl、ServerFileChooserControl、SingleFieldAllocationControl、SingleFieldChooserControl、SingleFieldValueChooserControl、SpinnerControl、TableControl、TextAreaControl、TextBoxControl

DataFile 要素

表 87. *DataFile* の属性

属性	使用	説明	有効な値
path	必須		

XML 表記

```
<xs:element name="DataFile" type="SERVER-DATA-FILE">
  <xs:attribute name="path" type="EVALUATED-STRING" use="required"/>
  <xs:choice>
    <xs:element ref="DelimitedDataFormat"/>
  </xs:choice>
</xs:element>
```

親要素

InputFiles、OutputFiles

子要素

DelimitedDataFormat

DataFormat 要素

XML 表記

```
<xs:element name="DataFormat">
  <xs:group ref="DATA-FORMAT-TYPE">
    <xs:choice>
      <xs:element ref="DelimitedDataFormat"/>
      <xs:element ref="SPSSDataFormat"/>
    </xs:choice>
  </xs:group>
</xs:element>
```

親要素

FileFormatType

子要素

DelimitedDataFormat、SPSSDataFormat

DataModel 要素

ノードの入力または出力となるデータ・モデル。入力/出力データ・モデルはフィールドのセットです。

XML 表記

```
<xs:element name="DataModel" type="DATA-MODEL">
  <xs:sequence>
    <xs:element name="FieldFormats" type="FIELD-FORMATS" minOccurs="0">
      <xs:sequence>
        <xs:element name="NumberFormat" type="NUMBER-FORMAT-DECLARATION" minOccurs="0" maxOccurs="unbounded">
        </xs:element>
      </xs:sequence>
    </xs:element>
    <xs:element name="FieldGroups" type="FIELD-GROUPS" minOccurs="0">
      <xs:sequence>
        <xs:element name="FieldGroup" type="FIELD-GROUP-DECLARATION" minOccurs="0" maxOccurs="unbounded">
          <xs:sequence>
            <xs:element name="FieldName">
            </xs:element>
          </xs:sequence>
        </xs:element>
      </xs:sequence>
    </xs:element>
    <xs:element name="Fields" type="FIELDS">
      <xs:sequence>
        <xs:element name="Field" type="FIELD" minOccurs="0" maxOccurs="unbounded">
          <xs:group ref="FIELD-CONTENT">
            <xs:sequence>
              <xs:element ref="DisplayLabel"/>
              <xs:choice minOccurs="0">
                <xs:element ref="Range"/>
                <xs:element ref="Values"/>
              </xs:choice>
              <xs:element ref="MissingValues"/>
            </xs:sequence>
          </xs:group>
        </xs:element>
      </xs:sequence>
    </xs:element>
  </xs:sequence>
</xs:element>
```

子要素

FieldFormats、FieldGroups、Fields

FieldFormats 要素**:** デフォルトのフィールド・フォーマットを定義します。フィールド・フォーマットは、一般形式 (標準数値、科学的表記、または通貨形式)、表示する小数部の桁数、小数点記号などの出力で値を表示するときに使用されます。現在、フィールド・フォーマットは数値型フィールドにのみ使用されますが、これは将来のバージョンで変更される可能性があります。

表 *88. FieldFormats* の属性

属性	使用	説明	有効な値
count	オプション		*nonNegativeInteger*
defaultNumberFormat	必須		*string*

XML 表記

```
<xs:element name="FieldFormats" type="FIELD-FORMATS" minOccurs="0">
  <xs:sequence>
    <xs:element name="NumberFormat" type="NUMBER-FORMAT-DECLARATION" minOccurs="0" maxOccurs="unbounded">
    </xs:element>
  </xs:sequence>
  <xs:attribute name="defaultNumberFormat" type="xs:string" use="required"/>
  <xs:attribute name="count" type="xs:nonNegativeInteger"/>
</xs:element>
```

親要素

DataModel

子要素

NumberFormat

***NumberFormat* 要素:** 数値型フィールドの形式情報を定義します。

表 *89. NumberFormat* の属性

属性	使用	説明	有効な値
decimalPlaces	必須		*nonNegativeInteger*
decimalSymbol	必須		**period** **comma**
formatType	必須		**standard** **scientific** **currency**
groupingSymbol	必須		**none** **period** **comma** **space**
name	必須		*string*

XML 表記

```
<xs:element name="NumberFormat" type="NUMBER-FORMAT-DECLARATION" minOccurs="0" maxOccurs="unbounded">
  <xs:attribute name="name" type="xs:string" use="required"/>
  <xs:attribute name="formatType" type="NUMBER-FORMAT-TYPE" use="required">
    <xs:enumeration value="standard"/>
    <xs:enumeration value="scientific"/>
    <xs:enumeration value="currency"/>
  </xs:attribute>
  <xs:attribute name="decimalPlaces" type="xs:nonNegativeInteger" use="required"/>
  <xs:attribute name="decimalSymbol" type="DECIMAL-SYMBOL" use="required">
    <xs:enumeration value="period"/>
    <xs:enumeration value="comma"/>
  </xs:attribute>
  <xs:attribute name="groupingSymbol" type="NUMBER-GROUPING-SYMBOL" use="required">
    <xs:enumeration value="none"/>
    <xs:enumeration value="period"/>
    <xs:enumeration value="comma"/>
    <xs:enumeration value="space"/>
  </xs:attribute>
</xs:element>
```

親要素

FieldFormats

FieldGroups 要素: フィールド・グループを定義します。フィールド・グループは関連フィールドを関連付けるために使用されます。例えば、訪れたことがある場所をオプションのセットから選択するように回答者に求めるアンケートは、フラグ型フィールドのセットとして表示されます。フィールド・グループを使用して、そのアンケートに関連付けられたフィールドを特定できます。

表 *90. FieldGroups* の属性

属性	使用	説明	有効な値
count	オプション		*nonNegativeInteger*

XML 表記

```
<xs:element name="FieldGroups" type="FIELD-GROUPS" minOccurs="0">
  <xs:sequence>
    <xs:element name="FieldGroup" type="FIELD-GROUP-DECLARATION" minOccurs="0" maxOccurs="unbounded">
      <xs:sequence>
        <xs:element name="FieldName">
        </xs:element>
      </xs:sequence>
    </xs:element>
  </xs:sequence>
  <xs:attribute name="count" type="xs:nonNegativeInteger"/>
</xs:element>
```

親要素

DataModel

子要素

FieldGroup

FieldGroup 要素**:** フィールド・グループを定義します。フィールド・グループはフィールド名とフィールド・グループに関する情報 (例えば、グループ名、オプションのラベル、グループのタイプ、および複合二分グループの場合は「true」で表される値などのカウントされる値) のリストで構成されています。

表 *91. FieldGroup* の属性

属性	使用	説明	有効な値
count	オプション		*nonNegativeInteger*
countedValue	オプション		*string*
displayLabel	オプション		*string*
groupType	必須		**fieldGroup** **multiCategorySet** **multiDichotomySet**
name	必須		

XML 表記

```
<xs:element name="FieldGroup" type="FIELD-GROUP-DECLARATION" minOccurs="0" maxOccurs="unbounded">
  <xs:sequence>
    <xs:element name="FieldName">
    </xs:element>
  </xs:sequence>
  <xs:attribute name="name" type="FIELD-GROUP-NAME" use="required"/>
  <xs:attribute name="displayLabel" type="xs:string"/>
  <xs:attribute name="groupType" type="FIELD-GROUP-TYPE" use="required">
    <xs:enumeration value="fieldGroup"/>
    <xs:enumeration value="multiCategorySet"/>
    <xs:enumeration value="multiDichotomySet"/>
  </xs:attribute>
  <xs:attribute name="countedValue" type="xs:string"/>
  <xs:attribute name="count" type="xs:nonNegativeInteger"/>
</xs:element>
```

親要素

FieldGroups

子要素

FieldName

***FieldName* 要素:**

表 *92. FieldName* の属性

属性	使用	説明	有効な値
name	必須		

XML 表記

```
<xs:element name="FieldName">
  <xs:attribute name="name" type="FIELD-NAME" use="required"/>
</xs:element>
```

親要素

FieldGroup

Fields 要素:

表 *93. Fields* の属性

属性	使用	説明	有効な値
count	オプション		*nonNegativeInteger*

XML 表記

```
<xs:element name="Fields" type="FIELDS">
  <xs:sequence>
    <xs:element name="Field" type="FIELD" minOccurs="0" maxOccurs="unbounded">
      <xs:group ref="FIELD-CONTENT">
        <xs:sequence>
          <xs:element ref="DisplayLabel"/>
          <xs:choice minOccurs="0">
            <xs:element ref="Range"/>
            <xs:element ref="Values"/>
          </xs:choice>
          <xs:element ref="MissingValues"/>
        </xs:sequence>
      </xs:group>
    </xs:element>
  </xs:sequence>
  <xs:attribute name="count" type="xs:nonNegativeInteger"/>
</xs:element>
```

親要素

DataModel

子要素

フィールド

***Field* 要素:**

表 *94. Field* の属性

属性	使用	説明	有効な値
direction	オプション		**in** **out** **both** **none** **partition**
displayLabel	オプション		*string*
name	必須		*string*
storage	オプション		**unknown** **integer** **real** **string** **date** **time** **timestamp** **list**
type	オプション		**auto** **range** **discrete** **set** **orderedSet** **flag** **typeless** **collection** **geospatial**

XML 表記

```
<xs:element name="Field" type="FIELD" minOccurs="0" maxOccurs="unbounded">
  <xs:group ref="FIELD-CONTENT">
    <xs:sequence>
      <xs:element ref="DisplayLabel"/>
      <xs:choice minOccurs="0">
        <xs:element ref="Range"/>
        <xs:element ref="Values"/>
      </xs:choice>
      <xs:element ref="MissingValues"/>
    </xs:sequence>
  </xs:group>
  <xs:attribute name="name" type="xs:string" use="required"/>
  <xs:attribute name="type" type="FIELD-TYPE" default="auto">
    <xs:enumeration value="auto"/>
    <xs:enumeration value="range"/>
    <xs:enumeration value="discrete"/>
    <xs:enumeration value="set"/>
    <xs:enumeration value="orderedSet"/>
    <xs:enumeration value="flag"/>
    <xs:enumeration value="typeless"/>
    <xs:enumeration value="collection"/>
    <xs:enumeration value="geospatial"/>
  </xs:attribute>
  <xs:attribute name="storage" type="FIELD-STORAGE" default="unknown">
    <xs:enumeration value="unknown"/>
    <xs:enumeration value="integer"/>
    <xs:enumeration value="real"/>
    <xs:enumeration value="string"/>
    <xs:enumeration value="date"/>
    <xs:enumeration value="time"/>
```

```
    <xs:enumeration value="timestamp"/>
    <xs:enumeration value="list"/>
  </xs:attribute>
  <xs:attribute name="direction" type="FIELD-DIRECTION" default="in">
    <xs:enumeration value="in"/>
    <xs:enumeration value="out"/>
    <xs:enumeration value="both"/>
    <xs:enumeration value="none"/>
    <xs:enumeration value="partition"/>
  </xs:attribute>
  <xs:attribute name="displayLabel" type="xs:string"/>
</xs:element>
```

親要素

フィールド

子要素

DisplayLabel、MissingValues、Range、Range、Values、Values

DatabaseConnectionValue 要素

データベース接続の詳細を指定する値。

表 *95. DatabaseConnectionValue* の属性

属性	使用	説明	有効な値
connectionType	必須		*string*
datasourceName	必須		*string*
password	必須		*string*
userName	必須		*string*

XML 表記

```
<xs:element name="DatabaseConnectionValue" type="DATABASE-CONNECTION-VALUE">
  <xs:attribute name="connectionType" type="xs:string" use="required"/>
  <xs:attribute name="datasourceName" type="xs:string" use="required"/>
  <xs:attribute name="userName" type="xs:string" use="required"/>
  <xs:attribute name="password" type="xs:string" use="required"/>
</xs:element>
```

親要素

Attribute、Attribute、ListValue、ListValue、ListValue、Parameter

DefaultValue 要素

XML 表記

```
<xs:element name="DefaultValue">
  <xs:choice>
    <xs:element name="ServerTempFile">
    </xs:element>
    <xs:element name="ServerTempDir">
    </xs:element>
    <xs:element name="Identifier">
    </xs:element>
  </xs:choice>
</xs:element>
```

親要素

Property、PropertyType

子要素

Identifier、ServerTempDir、ServerTempFile

ServerTempFile 要素**:**

表 *96. ServerTempFile* の属性

属性	使用	説明	有効な値
basename	必須		

XML 表記

```
<xs:element name="ServerTempFile">
  <xs:attribute name="basename" type="EVALUATED-STRING" use="required"/>
</xs:element>
```

親要素

DefaultValue

ServerTempDir 要素**:**

表 *97. ServerTempDir* の属性

属性	使用	説明	有効な値
basename	必須		

XML 表記

```
<xs:element name="ServerTempDir">
  <xs:attribute name="basename" type="EVALUATED-STRING" use="required"/>
</xs:element>
```

親要素

DefaultValue

Identifier 要素**:**

表 *98. Identifier* の属性

属性	使用	説明	有効な値
basename	必須		

XML 表記

```
<xs:element name="Identifier">
  <xs:attribute name="basename" type="EVALUATED-STRING" use="required"/>
</xs:element>
```

親要素

DefaultValue

DelimitedDataFormat 要素

表 99. *DelimitedDataFormat* の属性

属性	使用	説明	有効な値
delimeter	オプション		**tab** **comma** **semicolon** **colon** **verticalBar** **other**
eol	オプション		**cr** **crlf** **lf** **other**
includeFieldNames	オプション		*boolean*
otherDelimiter	オプション		*string*
otherEol	オプション		*string*
quoteStrings	オプション		*boolean*
stringQuote	オプション		*string*

XML 表記

```
<xs:element name="DelimitedDataFormat">
  <xs:attribute name="delimeter" use="optional" default="tab">
    <xs:simpleType>
      <xs:restriction base="xs:string">
        <xs:enumeration value="tab"/>
        <xs:enumeration value="comma"/>
        <xs:enumeration value="semicolon"/>
        <xs:enumeration value="colon"/>
        <xs:enumeration value="verticalBar"/>
        <xs:enumeration value="other"/>
      </xs:restriction>
    </xs:simpleType>
  </xs:attribute>
  <xs:attribute name="otherDelimiter" type="xs:string" use="optional"/>
  <xs:attribute name="eol" use="optional" default="cr">
    <xs:simpleType>
      <xs:restriction base="xs:string">
        <xs:enumeration value="cr"/>
        <xs:enumeration value="crlf"/>
        <xs:enumeration value="lf"/>
        <xs:enumeration value="other"/>
      </xs:restriction>
    </xs:simpleType>
  </xs:attribute>
  <xs:attribute name="otherEol" type="xs:string" use="optional"/>
  <xs:attribute name="includeFieldNames" type="xs:boolean" use="optional" default="true"/>
  <xs:attribute name="quoteStrings" type="xs:boolean" use="optional" default="false"/>
  <xs:attribute name="stringQuote" type="xs:string" use="optional" default="""/>
</xs:element>
```

親要素

DataFile、DataFormat

DisplayLabel 要素

指定された言語でのフィールドまたは値の表示ラベル。この displayLabel 属性は、特定の言語のラベルがない場合に使用できます。

表 *100. DisplayLabel* の属性

属性	使用	説明	有効な値
lang	オプション		*NMTOKEN*
value	必須		*string*

XML 表記

```
<xs:element name="DisplayLabel" type="DISPLAY-LABEL" minOccurs="0" maxOccurs="unbounded">
  <xs:attribute name="value" type="xs:string" use="required"/>
  <xs:attribute name="lang" type="xs:NMTOKEN" default="en"/>
</xs:element>
```

親要素

フィールド

DocumentBuilder 要素

XML 表記

```
<xs:element name="DocumentBuilder">
  <xs:sequence>
    <xs:element name="DocumentGeneration">
    </xs:element>
  </xs:sequence>
</xs:element>
```

親要素

ノード

子要素

DocumentGeneration

DocumentGeneration 要素:

表 *101. DocumentGeneration* の属性

属性	使用	説明	有効な値
controlsId	必須		*string*

XML 表記

```
<xs:element name="DocumentGeneration">
  <xs:attribute name="controlsId" type="xs:string" use="required"/>
</xs:element>
```

親要素

DocumentBuilder

DocumentOutput 要素

表 *102. DocumentOutput* の属性

属性	使用	説明	有効な値
delegate	オプション		*string*
deprecatedScriptNames	オプション		*string*

表 *102. DocumentOutput* の属性 (続き)

属性	使用	説明	有効な値
ID	必須		*string*
resourceKey	オプション		*string*
scriptName	オプション		*string*

XML 表記

```
<xs:element name="DocumentOutput">
  <xs:sequence maxOccurs="unbounded">
    <xs:choice maxOccurs="unbounded">
      <xs:element ref="Properties"/>
      <xs:element name="Containers" minOccurs="0">
        <xs:sequence maxOccurs="unbounded">
          <xs:element ref="Container"/>
        </xs:sequence>
      </xs:element>
      <xs:element ref="UserInterface"/>
      <xs:element ref="Constructors" minOccurs="0"/>
      <xs:element ref="ModelProvider" minOccurs="0"/>
    </xs:choice>
  </xs:sequence>
  <xs:attribute name="id" type="xs:string" use="required"/>
  <xs:attribute name="scriptName" type="xs:string" use="optional"/>
  <xs:attribute name="deprecatedScriptNames" type="xs:string" use="optional"/>
  <xs:attribute name="delegate" type="xs:string" use="optional"/>
  <xs:attribute name="resourceKey" type="xs:string" use="optional"/>
</xs:element>
```

親要素

Extension

子要素

Constructors、Containers、ModelProvider、Properties、UserInterface

関連する要素

InteractiveDocumentBuilder、InteractiveModelBuilder、ModelOutput、Node

Containers 要素**:**

XML 表記

```
<xs:element name="Containers" minOccurs="0">
  <xs:sequence maxOccurs="unbounded">
    <xs:element ref="Container"/>
  </xs:sequence>
</xs:element>
```

親要素

ノード

子要素

Container

DocumentType 要素

新規ドキュメント・タイプを定義します。

表 *103. DocumentType* の属性

属性	使用	説明	有効な値
content	オプション		*string*
format	必須		**utf8** **binary**
ID	必須		*string*
type	オプション		**unknown** **rowSet** **report** **graph**

XML 表記

```
<xs:element name="DocumentType">
  <xs:attribute name="id" type="xs:string" use="required"/>
  <xs:attribute name="format" use="required">
    <xs:simpleType>
      <xs:restriction base="xs:string">
        <xs:enumeration value="utf8"/>
        <xs:enumeration value="binary"/>
      </xs:restriction>
    </xs:simpleType>
  </xs:attribute>
  <xs:attribute name="content" type="xs:string" use="optional"/>
  <xs:attribute name="type" type="DOCUMENT-TYPE" use="optional">
    <xs:enumeration value="unknown"/>
    <xs:enumeration value="rowSet"/>
    <xs:enumeration value="report"/>
    <xs:enumeration value="graph"/>
  </xs:attribute>
</xs:element>
```

親要素

ContainerTypes

関連する要素

ContainerType、ModelType

Enabled 要素

どのような場合に UI コンポーネントを有効または編集可能にするかの条件を定義します。

XML 表記

```
<xs:element name="Enabled">
  <xs:sequence>
    <xs:group ref="CONDITION-EXPRESSION" minOccurs="0">
      <xs:choice>
        <xs:element ref="Condition"/>
        <xs:element ref="And"/>
        <xs:element ref="Or"/>
        <xs:element ref="Not"/>
      </xs:choice>
    </xs:group>
  </xs:sequence>
</xs:element>
```

親要素

ActionButton、CheckBoxControl、CheckBoxGroupControl、ClientDirectoryChooserControl、ClientFileChooserControl、ComboBoxControl、DBConnectionChooserControl、DBTableChooserControl、ExtensionObjectPanel、FieldAllocationList、ItemChooserControl、ModelViewerPanel、MultiFieldAllocationControl、MultiFieldChooserControl、MultiFieldSelectionTableControl、MultiItemChooserControl、OutputViewerPanel、PasswordBoxControl、PropertiesPanel、PropertiesSubPanel、PropertyControl、RadioButtonGroupControl、SelectorPanel、ServerDirectoryChooserControl、ServerFileChooserControl、SingleFieldAllocationControl、SingleFieldChooserControl、SingleFieldValueChooserControl、SingleItemChooserControl、SpinnerControl、StaticText、SystemControls、TabbedPanel、TableControl、TextAreaControl、TextBoxControl、TextBrowserPanel

子要素

And、Condition、Not、Or

Enumeration 要素

XML 表記

```
<xs:element name="Enumeration">
  <xs:sequence>
    <xs:element name="Enum" maxOccurs="unbounded">
    </xs:element>
  </xs:sequence>
</xs:element>
```

親要素

PropertyType

子要素

Enum

Enum 要素:

表 *104. Enum* の属性

属性	使用	説明	有効な値
description	オプション		*string*
descriptionKey	オプション		*string*
imagePath	オプション		*string*
imagePathKey	オプション		*string*
label	必須		*string*
labelKey	オプション		*string*
value	必須		*string*

XML 表記

```
<xs:element name="Enum" maxOccurs="unbounded">
  <xs:attribute name="value" type="xs:string" use="required"/>
  <xs:attribute name="label" type="xs:string" use="required"/>
  <xs:attribute name="labelKey" type="xs:string" use="optional"/>
  <xs:attribute name="description" type="xs:string" use="optional"/>
```

```
  <xs:attribute name="descriptionKey" type="xs:string" use="optional"/>
  <xs:attribute name="imagePath" type="xs:string" use="optional"/>
  <xs:attribute name="imagePathKey" type="xs:string" use="optional"/>
</xs:element>
```

親要素

Enumeration

ErrorDetail 要素

エラーまたはその他の状況に関する補足情報。

XML 表記

```
<xs:element name="ErrorDetail" type="ERROR-DETAIL">
  <xs:sequence>
    <xs:element name="Diagnostic" type="DIAGNOSTIC" minOccurs="0" maxOccurs="unbounded">
      <xs:sequence>
        <xs:element name="Message" type="DIAGNOSTIC-MESSAGE" minOccurs="0">
        </xs:element>
        <xs:element name="Parameter" type="xs:string" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:element>
  </xs:sequence>
</xs:element>
```

子要素

Diagnostic

Diagnostic 要素**:**

表 *105. Diagnostic* の属性

属性	使用	説明	有効な値
code	必須		*integer*
severity	オプション		**unknown** **information** **warning** **error** **fatal**
source	オプション		*string*
subCode	オプション		*integer*

XML 表記

```
<xs:element name="Diagnostic" type="DIAGNOSTIC" minOccurs="0" maxOccurs="unbounded">
  <xs:sequence>
    <xs:element name="Message" type="DIAGNOSTIC-MESSAGE" minOccurs="0">
    </xs:element>
    <xs:element name="Parameter" type="xs:string" minOccurs="0" maxOccurs="unbounded"/>
  </xs:sequence>
  <xs:attribute name="code" type="xs:integer" use="required"/>
  <xs:attribute name="subCode" type="xs:integer" default="0"/>
  <xs:attribute name="severity" type="DIAGNOSTIC-SEVERITY" default="error">
    <xs:enumeration value="unknown"/>
    <xs:enumeration value="information"/>
    <xs:enumeration value="warning"/>
    <xs:enumeration value="error"/>
    <xs:enumeration value="fatal"/>
  </xs:attribute>
  <xs:attribute name="source" type="xs:string"/>
</xs:element>
```

親要素

ErrorDetail

子要素

Message、Parameter

Message 要素*:*

表 *106. Message* の属性

属性	使用	説明	有効な値
lang	オプション		*NMTOKEN*

XML 表記

```
<xs:element name="Message" type="DIAGNOSTIC-MESSAGE" minOccurs="0">
  <xs:attribute name="lang" type="xs:NMTOKEN"/>
</xs:element>
```

親要素

Diagnostic

Parameter 要素*:*

XML 表記

```
<xs:element name="Parameter" type="xs:string" minOccurs="0" maxOccurs="unbounded"/>
```

親要素

Diagnostic

Executable 要素

XML 表記

```
<xs:element name="Executable">
  <xs:sequence>
    <xs:element ref="Run" maxOccurs="unbounded"/>
  </xs:sequence>
</xs:element>
```

親要素

Execution

子要素

Run

Execution 要素

XML 表記

```
<xs:element name="Execution">
  <xs:sequence>
    <xs:element ref="Properties" minOccurs="0"/>
    <xs:element ref="InputFiles"/>
    <xs:element ref="OutputFiles"/>
```

```
    <xs:choice>
      <xs:element ref="Executable"/>
      <xs:element ref="Module"/>
    </xs:choice>
    <xs:element ref="Constructors" minOccurs="0"/>
  </xs:sequence>
</xs:element>
```

親要素

ノード

子要素

Constructors、Executable、InputFiles、Module、OutputFiles、Properties

Extension 要素

最上位の拡張コンテナーを定義します。

表 *107. Extension* の属性

属性	使用	説明	有効な値
debug	オプション		*boolean*
stable	オプション		*boolean*
version	必須		*string*

XML 表記

```
<xs:element name="Extension">
  <xs:sequence>
    <xs:element ref="ExtensionDetails"/>
    <xs:element ref="Resources"/>
    <xs:element ref="License" minOccurs="0"/>
    <xs:element ref="CommonObjects"/>
    <xs:element ref="UserInterface" minOccurs="0"/>
    <xs:sequence minOccurs="0" maxOccurs="unbounded">
      <xs:choice>
        <xs:element ref="Node"/>
        <xs:element ref="ModelOutput"/>
        <xs:element ref="DocumentOutput"/>
        <xs:element ref="InteractiveModelBuilder"/>
        <xs:element ref="InteractiveDocumentBuilder"/>
      </xs:choice>
    </xs:sequence>
  </xs:sequence>
  <xs:attribute name="version" type="xs:string" use="required"/>
  <xs:attribute name="debug" type="xs:boolean" use="optional" default="false"/>
  <xs:attribute name="stable" type="xs:boolean" use="optional" default="true"/>
</xs:element>
```

子要素

CommonObjects、DocumentOutput、ExtensionDetails、InteractiveDocumentBuilder、InteractiveModelBuilder、License、ModelOutput、Node、Resources、UserInterface

ExtensionDetails 要素

拡張 ID、拡張プロバイダー、およびバージョン情報などの拡張に関する情報を定義します。

表 *108. ExtensionDetails* の属性

属性	使用	説明	有効な値
copyright	オプション		*string*
description	オプション		*string*

表 108. *ExtensionDetails* の属性 (続き)

属性	使用	説明	有効な値
ID	必須		*string*
label	必須		*string*
provider	オプション		*string*
providerTag	必須		*string*
version	オプション		*string*

XML 表記

```
<xs:element name="ExtensionDetails">
  <xs:attribute name="providerTag" type="xs:string" use="required"/>
  <xs:attribute name="id" type="xs:string" use="required"/>
  <xs:attribute name="label" type="xs:string" use="required"/>
  <xs:attribute name="version" type="xs:string"/>
  <xs:attribute name="provider" type="xs:string" use="optional" default="(unknown)"/>
  <xs:attribute name="copyright" type="xs:string" use="optional"/>
  <xs:attribute name="description" type="xs:string" use="optional"/>
</xs:element>
```

親要素

Extension

ExtensionObjectPanel 要素

カスタム パネルを定義します。panelClass 属性で識別されるクラスは、ExtensionObjectPanel インターフェースを実装する必要があります。

表 109. *ExtensionObjectPanel* の属性

属性	使用	説明	有効な値
ID	オプション		*string*
panelClass	必須		*string*

XML 表記

```
<xs:element name="ExtensionObjectPanel">
  <xs:sequence>
    <xs:choice>
      <xs:element ref="Layout" minOccurs="0" maxOccurs="1"/>
      <xs:element ref="Enabled" minOccurs="0" maxOccurs="1"/>
      <xs:element ref="Visible" minOccurs="0" maxOccurs="1"/>
      <xs:element ref="Required" minOccurs="0" maxOccurs="1"/>
    </xs:choice>
  </xs:sequence>
  <xs:attribute name="panelClass" type="xs:string" use="required"/>
  <xs:attribute name="id" type="xs:string" use="optional"/>
</xs:element>
```

親要素

PropertiesPanel、PropertiesSubPanel、Tab

子要素

Enabled、Layout、Required、Visible

関連する要素

ActionButton、ComboBoxControl、FieldAllocationList、ModelViewerPanel、OutputViewerPanel、SelectorPanel、StaticText、SystemControls、TabbedPanel、TextBrowserPanel

Field 要素

表 *110. Field* の属性

属性	使用	説明	有効な値
depth	オプション		*integer*
direction	オプション		**in** **out** **both** **none** **partition**
label	オプション		*string*
name	必須		
storage	オプション		**unknown** **integer** **real** **string** **date** **time** **timestamp** **list**
type	オプション		**auto** **range** **discrete** **set** **orderedSet** **flag** **typeless** **collection** **geospatial**
valueStorage	オプション		**unknown** **integer** **real** **string** **date** **time** **timestamp** **list**

XML 表記

```
<xs:element name="Field" type="FIELD-DECLARATION">
  <xs:sequence>
    <xs:choice>
      <xs:element ref="Range" minOccurs="0"/>
      <xs:element ref="Values" minOccurs="0"/>
      <xs:element ref="NumericInfo" minOccurs="0"/>
      <xs:element name="MissingValues" minOccurs="0">
        <xs:sequence>
          <xs:element ref="Values" minOccurs="0" maxOccurs="unbounded"/>
          <xs:element ref="Range" minOccurs="0"/>
        </xs:sequence>
      </xs:element>
      <xs:element name="ModelField" type="MODEL-FIELD-INFORMATION" minOccurs="0">
      </xs:element>
    </xs:choice>
  </xs:sequence>
  <xs:attribute name="name" type="FIELD-NAME" use="required"/>
  <xs:attribute name="storage" type="FIELD-STORAGE">
    <xs:enumeration value="unknown"/>
    <xs:enumeration value="integer"/>
    <xs:enumeration value="real"/>
    <xs:enumeration value="string"/>
    <xs:enumeration value="date"/>
    <xs:enumeration value="time"/>
    <xs:enumeration value="timestamp"/>
    <xs:enumeration value="list"/>
  </xs:attribute>
  <xs:attribute name="type" type="FIELD-TYPE">
    <xs:enumeration value="auto"/>
    <xs:enumeration value="range"/>
    <xs:enumeration value="discrete"/>
    <xs:enumeration value="set"/>
    <xs:enumeration value="orderedSet"/>
    <xs:enumeration value="flag"/>
    <xs:enumeration value="typeless"/>
    <xs:enumeration value="collection"/>
    <xs:enumeration value="geospatial"/>
  </xs:attribute>
  <xs:attribute name="direction" type="FIELD-DIRECTION">
    <xs:enumeration value="in"/>
    <xs:enumeration value="out"/>
    <xs:enumeration value="both"/>
    <xs:enumeration value="none"/>
    <xs:enumeration value="partition"/>
  </xs:attribute>
  <xs:attribute name="label" type="xs:string"/>
  <xs:attribute name="depth" type="xs:integer" use="optional" default="-1"/>
  <xs:attribute name="valueStorage" type="FIELD-STORAGE" use="optional">
    <xs:enumeration value="unknown"/>
    <xs:enumeration value="integer"/>
    <xs:enumeration value="real"/>
    <xs:enumeration value="string"/>
    <xs:enumeration value="date"/>
    <xs:enumeration value="time"/>
    <xs:enumeration value="timestamp"/>
    <xs:enumeration value="list"/>
  </xs:attribute>
</xs:element>
```

子要素

MissingValues、ModelField、NumericInfo、Range、Range、Values、Values

MissingValues 要素:

表 *111. MissingValues* の属性

属性	使用	説明	有効な値
treatNullAsMissing	オプション		*boolean*
treatWhitespaceAsMissing	オプション		*boolean*

XML 表記

```
<xs:element name="MissingValues" minOccurs="0">
  <xs:sequence>
    <xs:element ref="Values" minOccurs="0" maxOccurs="unbounded"/>
    <xs:element ref="Range" minOccurs="0"/>
  </xs:sequence>
  <xs:attribute name="treatWhitespaceAsMissing" type="xs:boolean" use="optional" default="true"/>
  <xs:attribute name="treatNullAsMissing" type="xs:boolean" use="optional" default="true"/>
</xs:element>
```

親要素

AddField

子要素

Range、Range、Values、Values

ModelField 要素**:**

表 *112. ModelField* の属性

属性	使用	説明	有効な値
group	オプション		
role	必須		**unknown** **predictedValue** **predictedDisplayValue** **probability** **residual** **standardError** **entityId** **entityAffinity** **upperConfidenceLimit** **lowerConfidenceLimit** **propensity** **value** **supplementary**
tag	オプション		*string*
targetField	オプション		*string*
value	オプション		*string*

XML 表記

```
<xs:element name="ModelField" type="MODEL-FIELD-INFORMATION" minOccurs="0">
  <xs:attribute name="role" type="MODEL-FIELD-ROLE" use="required">
    <xs:enumeration value="unknown"/>
    <xs:enumeration value="predictedValue"/>
    <xs:enumeration value="predictedDisplayValue"/>
    <xs:enumeration value="probability"/>
    <xs:enumeration value="residual"/>
    <xs:enumeration value="standardError"/>
    <xs:enumeration value="entityId"/>
    <xs:enumeration value="entityAffinity"/>
    <xs:enumeration value="upperConfidenceLimit"/>
    <xs:enumeration value="lowerConfidenceLimit"/>
    <xs:enumeration value="propensity"/>
    <xs:enumeration value="value"/>
    <xs:enumeration value="supplementary"/>
  </xs:attribute>
  <xs:attribute name="targetField" type="xs:string"/>
```

```
  <xs:attribute name="value" type="xs:string"/>
  <xs:attribute name="group" type="MODEL-FIELD-GROUP"/>
  <xs:attribute name="tag" type="xs:string"/>
</xs:element>
```

親要素

AddField

FieldAllocationList 要素

利用可能なフィールドのリストを格納しているパネルを定義します。単一フィールドまたは複数フィールドの割り振りコントロールによって、このパネルからフィールドを割り振ることができます。

表 *113. FieldAllocationList* の属性

属性	使用	説明	有効な値
enabledProperty	オプション		*string*
enabledValue	オプション		*string*
excludes	オプション	コンテンツがブロック対象であるフィールド ターゲット プロパティー名のリスト。	*string*
ID	必須		*string*
includes	オプション	コンテンツがデータ ソースであるフィールド ターゲット プロパティー名のリスト。	*string*
onlyDatetime	オプション		*boolean*
onlyDiscrete	オプション		*boolean*
onlyNumeric	オプション		*boolean*
onlyRanges	オプション		*boolean*
onlySymbolic	オプション		*boolean*
storage	オプション		*string*
types	オプション		*string*

XML 表記

```
<xs:element name="FieldAllocationList">
  <xs:sequence>
    <xs:choice>
      <xs:element ref="Layout" minOccurs="0" maxOccurs="1"/>
      <xs:element ref="Enabled" minOccurs="0" maxOccurs="1"/>
      <xs:element ref="Visible" minOccurs="0" maxOccurs="1"/>
      <xs:element ref="Required" minOccurs="0" maxOccurs="1"/>
    </xs:choice>
  </xs:sequence>
  <xs:attribute name="id" type="xs:string" use="required"/>
  <xs:attribute name="enabledProperty" type="xs:string" use="optional"/>
  <xs:attribute name="enabledValue" type="xs:string" use="optional"/>
  <xs:attribute name="storage" type="xs:string" use="optional"/>
  <xs:attribute name="onlyNumeric" type="xs:boolean" use="optional"/>
  <xs:attribute name="onlySymbolic" type="xs:boolean" use="optional"/>
  <xs:attribute name="onlyDatetime" type="xs:boolean" use="optional"/>
  <xs:attribute name="types" type="xs:string" use="optional"/>
  <xs:attribute name="onlyRanges" type="xs:boolean" use="optional"/>
  <xs:attribute name="onlyDiscrete" type="xs:boolean" use="optional"/>
  <xs:attribute name="includes" type="xs:string" use="optional"/>
  <xs:attribute name="excludes" type="xs:string" use="optional"/>
</xs:element>
```

親要素

PropertiesPanel、PropertiesSubPanel

子要素

Enabled、Layout、Required、Visible

関連する要素

ActionButton、ComboBoxControl、ExtensionObjectPanel、ModelViewerPanel、OutputViewerPanel、SelectorPanel、StaticText、SystemControls、TabbedPanel、TextBrowserPanel

FieldFormats 要素

デフォルトのフィールド・フォーマットを定義します。フィールド・フォーマットは、一般形式 (標準数値、科学的表記、または通貨形式)、表示する小数部の桁数、小数点記号などの出力で値を表示するときに使用されます。現在、フィールド・フォーマットは数値型フィールドにのみ使用されますが、これは将来のバージョンで変更される可能性があります。

表 *114. FieldFormats* の属性

属性	使用	説明	有効な値
count	オプション		*nonNegativeInteger*
defaultNumberFormat	必須		*string*

XML 表記

```
<xs:element name="FieldFormats" type="FIELD-FORMATS">
  <xs:sequence>
    <xs:element name="NumberFormat" type="NUMBER-FORMAT-DECLARATION" minOccurs="0" maxOccurs="unbounded">
    </xs:element>
  </xs:sequence>
  <xs:attribute name="defaultNumberFormat" type="xs:string" use="required"/>
  <xs:attribute name="count" type="xs:nonNegativeInteger"/>
</xs:element>
```

子要素

NumberFormat

NumberFormat 要素: 数値型フィールドの形式情報を定義します。

表 *115. NumberFormat* の属性

属性	使用	説明	有効な値
decimalPlaces	必須		*nonNegativeInteger*
decimalSymbol	必須		**period** **comma**
formatType	必須		**standard** **scientific** **currency**

表 115. *NumberFormat* の属性 (続き)

属性	使用	説明	有効な値
groupingSymbol	必須		**none** **period** **comma** **space**
name	必須		*string*

XML 表記

```
<xs:element name="NumberFormat" type="NUMBER-FORMAT-DECLARATION" minOccurs="0" maxOccurs="unbounded">
  <xs:attribute name="name" type="xs:string" use="required"/>
  <xs:attribute name="formatType" type="NUMBER-FORMAT-TYPE" use="required">
    <xs:enumeration value="standard"/>
    <xs:enumeration value="scientific"/>
    <xs:enumeration value="currency"/>
  </xs:attribute>
  <xs:attribute name="decimalPlaces" type="xs:nonNegativeInteger" use="required"/>
  <xs:attribute name="decimalSymbol" type="DECIMAL-SYMBOL" use="required">
    <xs:enumeration value="period"/>
    <xs:enumeration value="comma"/>
  </xs:attribute>
  <xs:attribute name="groupingSymbol" type="NUMBER-GROUPING-SYMBOL" use="required">
    <xs:enumeration value="none"/>
    <xs:enumeration value="period"/>
    <xs:enumeration value="comma"/>
    <xs:enumeration value="space"/>
  </xs:attribute>
</xs:element>
```

親要素

FieldFormats

FieldGroup 要素

フィールド・グループを定義します。フィールド・グループはフィールド名とフィールド・グループに関する情報 (例えば、グループ名、オプションのラベル、グループのタイプ、および複合二分グループの場合は「true」で表される値などのカウントされる値) のリストで構成されています。

表 116. *FieldGroup* の属性

属性	使用	説明	有効な値
count	オプション		*nonNegativeInteger*
countedValue	オプション		*string*
displayLabel	オプション		*string*
groupType	必須		**fieldGroup** **multiCategorySet** **multiDichotomySet**
name	必須		

XML 表記

```
<xs:element name="FieldGroup" type="FIELD-GROUP-DECLARATION">
  <xs:sequence>
    <xs:element name="FieldName">
    </xs:element>
  </xs:sequence>
  <xs:attribute name="name" type="FIELD-GROUP-NAME" use="required"/>
```

```
  <xs:attribute name="displayLabel" type="xs:string"/>
  <xs:attribute name="groupType" type="FIELD-GROUP-TYPE" use="required">
    <xs:enumeration value="fieldGroup"/>
    <xs:enumeration value="multiCategorySet"/>
    <xs:enumeration value="multiDichotomySet"/>
  </xs:attribute>
  <xs:attribute name="countedValue" type="xs:string"/>
  <xs:attribute name="count" type="xs:nonNegativeInteger"/>
</xs:element>
```

子要素

FieldName

FieldName 要素:

表 *117. FieldName* の属性

属性	使用	説明	有効な値
name	必須		

XML 表記

```
<xs:element name="FieldName">
  <xs:attribute name="name" type="FIELD-NAME" use="required"/>
</xs:element>
```

親要素

FieldGroup

FieldGroups 要素

フィールド・グループを定義します。フィールド・グループは関連フィールドを関連付けるために使用されます。例えば、訪れたことがある場所をオプションのセットから選択するように回答者に求めるアンケートは、フラグ型フィールドのセットとして表示されます。フィールド・グループを使用して、そのアンケートに関連付けられたフィールドを特定できます。

表 *118. FieldGroups* の属性

属性	使用	説明	有効な値
count	オプション		*nonNegativeInteger*

XML 表記

```
<xs:element name="FieldGroups" type="FIELD-GROUPS">
  <xs:sequence>
    <xs:element name="FieldGroup" type="FIELD-GROUP-DECLARATION" minOccurs="0" maxOccurs="unbounded">
      <xs:sequence>
        <xs:element name="FieldName">
        </xs:element>
      </xs:sequence>
    </xs:element>
  </xs:sequence>
  <xs:attribute name="count" type="xs:nonNegativeInteger"/>
</xs:element>
```

子要素

FieldGroup

FieldGroup 要素: フィールド・グループを定義します。フィールド・グループはフィールド名とフィールド・グループに関する情報 (例えば、グループ名、オプションのラベル、グループのタイプ、および複合二分グループの場合は「true」で表される値などのカウントされる値) のリストで構成されています。

表 *119. FieldGroup* の属性

属性	使用	説明	有効な値
count	オプション		*nonNegativeInteger*
countedValue	オプション		*string*
displayLabel	オプション		*string*
groupType	必須		**fieldGroup** **multiCategorySet** **multiDichotomySet**
name	必須		

XML 表記

```
<xs:element name="FieldGroup" type="FIELD-GROUP-DECLARATION" minOccurs="0" maxOccurs="unbounded">
  <xs:sequence>
    <xs:element name="FieldName">
    </xs:element>
  </xs:sequence>
  <xs:attribute name="name" type="FIELD-GROUP-NAME" use="required"/>
  <xs:attribute name="displayLabel" type="xs:string"/>
  <xs:attribute name="groupType" type="FIELD-GROUP-TYPE" use="required">
    <xs:enumeration value="fieldGroup"/>
    <xs:enumeration value="multiCategorySet"/>
    <xs:enumeration value="multiDichotomySet"/>
  </xs:attribute>
  <xs:attribute name="countedValue" type="xs:string"/>
  <xs:attribute name="count" type="xs:nonNegativeInteger"/>
</xs:element>
```

親要素

FieldGroups

子要素

FieldName

FieldName 要素*:*

表 *120. FieldName* の属性

属性	使用	説明	有効な値
name	必須		

XML 表記

```
<xs:element name="FieldName">
  <xs:attribute name="name" type="FIELD-NAME" use="required"/>
</xs:element>
```

親要素

FieldGroup

FileFormatType 要素

表 121. *FileFormatType* の属性

属性	使用	説明	有効な値
name	オプション		

XML 表記

```
<xs:element name="FileFormatType">
  <xs:sequence>
    <xs:group ref="FILE-FORMAT">
      <xs:choice>
        <xs:element ref="UTF8Format"/>
        <xs:element ref="BinaryFormat"/>
        <xs:element ref="DataFormat"/>
      </xs:choice>
    </xs:group>
  </xs:sequence>
  <xs:attribute name="name" type="EVALUATED-STRING" use="optional"/>
</xs:element>
```

親要素

FileFormatTypes

子要素

BinaryFormat、DataFormat、UTF8Format

FileFormatTypes 要素

XML 表記

```
<xs:element name="FileFormatTypes">
  <xs:sequence>
    <xs:element ref="FileFormatType" minOccurs="0" maxOccurs="unbounded"/>
  </xs:sequence>
</xs:element>
```

親要素

CommonObjects

子要素

FileFormatType

ForEach 要素

表 122. *ForEach* の属性

属性	使用	説明	有効な値
container	オプション		*string*
from	オプション		*string*
inFieldValues	オプション		*string*
inFields	オプション		*string*
inProperty	オプション		*string*
step	オプション		*string*
～	オプション		*string*

表 122. *ForEach* の属性 (続き)

属性	使用	説明	有効な値
var	必須		*string*

XML 表記

```
<xs:element name="ForEach">
  <xs:sequence maxOccurs="unbounded">
    <xs:group ref="DATA-MODEL-EXPRESSION">
      <xs:choice>
        <xs:element ref="ForEach"/>
        <xs:element ref="AddField"/>
        <xs:element ref="ChangeField"/>
        <xs:element ref="RemoveField"/>
      </xs:choice>
    </xs:group>
  </xs:sequence>
  <xs:attribute name="var" type="xs:string" use="required"/>
  <xs:attribute name="inProperty" type="xs:string" use="optional"/>
  <xs:attribute name="inFields" type="xs:string" use="optional"/>
  <xs:attribute name="inFieldValues" type="xs:string" use="optional"/>
  <xs:attribute name="from" type="xs:string" use="optional"/>
  <xs:attribute name="to" type="xs:string" use="optional"/>
  <xs:attribute name="step" type="xs:string" use="optional"/>
  <xs:attribute name="container" type="xs:string" use="optional"/>
</xs:element>
```

親要素

ForEach、ModelFields

子要素

AddField、ChangeField、ForEach、RemoveField

Icon 要素

表 123. *Icon* の属性

属性	使用	説明	有効な値
imagePath	必須		*string*
resourceID	オプション		*string*
type	必須		**standardNode** **smallNode** **standardWindow**

XML 表記

```
<xs:element name="Icon">
  <xs:attribute name="type" use="required">
    <xs:simpleType>
      <xs:restriction base="xs:string">
        <xs:enumeration value="standardNode"/>
        <xs:enumeration value="smallNode"/>
        <xs:enumeration value="standardWindow"/>
      </xs:restriction>
    </xs:simpleType>
  </xs:attribute>
  <xs:attribute name="imagePath" type="xs:string" use="required"/>
  <xs:attribute name="resourceID" type="xs:string" use="optional"/>
</xs:element>
```

親要素

Icons、Palette

Icons 要素

XML 表記

```
<xs:element name="Icons">
  <xs:sequence>
    <xs:element ref="Icon" maxOccurs="unbounded"/>
  </xs:sequence>
</xs:element>
```

親要素

UserInterface

子要素

Icon

InputFiles 要素

XML 表記

```
<xs:element name="InputFiles">
  <xs:group ref="RUNTIME-FILES">
    <xs:sequence>
      <xs:element ref="DataFile"/>
      <xs:element ref="ContainerFile" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:group>
</xs:element>
```

親要素

Execution、Module

子要素

ContainerFile、DataFile

InteractiveDocumentBuilder 要素

表 *124. InteractiveDocumentBuilder* の属性

属性	使用	説明	有効な値
delegate	オプション		*string*
deprecatedScriptNames	オプション		*string*
ID	必須		*string*
resourceKey	オプション		*string*
scriptName	オプション		*string*

XML 表記

```
<xs:element name="InteractiveDocumentBuilder">
  <xs:sequence maxOccurs="unbounded">
    <xs:choice maxOccurs="unbounded">
      <xs:element ref="Properties"/>
      <xs:element name="Containers" minOccurs="0">
        <xs:sequence maxOccurs="unbounded">
```

```
          <xs:element ref="Container"/>
        </xs:sequence>
      </xs:element>
      <xs:element ref="UserInterface"/>
      <xs:element ref="Constructors" minOccurs="0"/>
      <xs:element ref="ModelProvider" minOccurs="0"/>
    </xs:choice>
  </xs:sequence>
  <xs:attribute name="id" type="xs:string" use="required"/>
  <xs:attribute name="scriptName" type="xs:string" use="optional"/>
  <xs:attribute name="deprecatedScriptNames" type="xs:string" use="optional"/>
  <xs:attribute name="delegate" type="xs:string" use="optional"/>
  <xs:attribute name="resourceKey" type="xs:string" use="optional"/>
</xs:element>
```

親要素

Extension

子要素

Constructors、Containers、ModelProvider、Properties、UserInterface

関連する要素

DocumentOutput、InteractiveModelBuilder、ModelOutput、Node

Containers 要素**:**

XML 表記

```
<xs:element name="Containers" minOccurs="0">
  <xs:sequence maxOccurs="unbounded">
    <xs:element ref="Container"/>
  </xs:sequence>
</xs:element>
```

親要素

ノード

子要素

Container

InteractiveModelBuilder 要素

表 *125. InteractiveModelBuilder* の属性

属性	使用	説明	有効な値
delegate	オプション		*string*
deprecatedScriptNames	オプション		*string*
ID	必須		*string*
resourceKey	オプション		*string*
scriptName	オプション		*string*

XML 表記

```
<xs:element name="InteractiveModelBuilder">
  <xs:sequence maxOccurs="unbounded">
    <xs:choice maxOccurs="unbounded">
      <xs:element ref="Properties"/>
      <xs:element name="Containers" minOccurs="0">
```

```
        <xs:sequence maxOccurs="unbounded">
          <xs:element ref="Container"/>
        </xs:sequence>
      </xs:element>
      <xs:element ref="UserInterface"/>
      <xs:element ref="Constructors" minOccurs="0"/>
      <xs:element ref="ModelProvider" minOccurs="0"/>
    </xs:choice>
  </xs:sequence>
  <xs:attribute name="id" type="xs:string" use="required"/>
  <xs:attribute name="scriptName" type="xs:string" use="optional"/>
  <xs:attribute name="deprecatedScriptNames" type="xs:string" use="optional"/>
  <xs:attribute name="delegate" type="xs:string" use="optional"/>
  <xs:attribute name="resourceKey" type="xs:string" use="optional"/>
</xs:element>
```

親要素

Extension

子要素

Constructors、Containers、ModelProvider、Properties、UserInterface

関連する要素

DocumentOutput、InteractiveDocumentBuilder、ModelOutput、Node

Containers 要素**:**

XML 表記

```
<xs:element name="Containers" minOccurs="0">
  <xs:sequence maxOccurs="unbounded">
    <xs:element ref="Container"/>
  </xs:sequence>
</xs:element>
```

親要素

ノード

子要素

Container

Layout 要素

UI コンポーネントのレイアウトを、パネル内でどのように配置するかを定義します。レイアウトはグリッド バッグ レイアウトに基づき、各コントロールはグリッド内の 1 つ以上のセルを占有します。このレイアウトでは、サイズ変更、塗りつぶし、アンカー、インデントの動作を指定できます。

表 *126. Layout* の属性

属性	使用	説明	有効な値
anchor	オプション		**north** **northeast** **east** **southeast** **south** **southwest** **west** **northwest** **center**
columnWeight	オプション		*double*
fill	オプション		**horizontal** **vertical** **both** **none**
gridColumn	オプション		*nonNegativeInteger*
gridHeight	オプション		*nonNegativeInteger*
gridRow	オプション		*nonNegativeInteger*
gridWidth	オプション		*nonNegativeInteger*
leftIndent	オプション		*nonNegativeInteger*
rowIncrement	オプション		*nonNegativeInteger*
rowWeight	オプション		*double*

XML 表記

```
<xs:element name="Layout">
  <xs:sequence minOccurs="0" maxOccurs="unbounded">
    <xs:element name="Cell">
    </xs:element>
  </xs:sequence>
  <xs:attribute name="gridRow" type="xs:nonNegativeInteger" use="optional"/>
  <xs:attribute name="gridColumn" type="xs:nonNegativeInteger" use="optional"/>
  <xs:attribute name="rowIncrement" type="xs:nonNegativeInteger" use="optional"/>
  <xs:attribute name="gridWidth" type="xs:nonNegativeInteger" use="optional" default="1"/>
  <xs:attribute name="gridHeight" type="xs:nonNegativeInteger" use="optional" default="1"/>
  <xs:attribute name="rowWeight" type="xs:double" use="optional"/>
  <xs:attribute name="columnWeight" type="xs:double" use="optional"/>
  <xs:attribute name="fill" type="UI-COMPONENT-FILL" use="optional" default="none">
    <xs:enumeration value="horizontal"/>
    <xs:enumeration value="vertical"/>
    <xs:enumeration value="both"/>
    <xs:enumeration value="none"/>
  </xs:attribute>
  <xs:attribute name="anchor" type="UI-COMPONENT-ANCHOR" use="optional" default="west">
    <xs:enumeration value="north"/>
    <xs:enumeration value="northeast"/>
    <xs:enumeration value="east"/>
    <xs:enumeration value="southeast"/>
    <xs:enumeration value="south"/>
    <xs:enumeration value="southwest"/>
    <xs:enumeration value="west"/>
    <xs:enumeration value="northwest"/>
    <xs:enumeration value="center"/>
  </xs:attribute>
  <xs:attribute name="leftIndent" type="xs:nonNegativeInteger" use="optional"/>
</xs:element>
```

親要素

ActionButton、CheckBoxControl、CheckBoxGroupControl、ClientDirectoryChooserControl、ClientFileChooserControl、ComboBoxControl、DBConnectionChooserControl、DBTableChooserControl、ExtensionObjectPanel、FieldAllocationList、ItemChooserControl、ModelViewerPanel、MultiFieldAllocationControl、MultiFieldChooserControl、MultiFieldSelectionTableControl、MultiItemChooserControl、OutputViewerPanel、PasswordBoxControl、PropertiesPanel、PropertiesSubPanel、PropertyControl、RadioButtonGroupControl、SelectorPanel、ServerDirectoryChooserControl、ServerFileChooserControl、SingleFieldAllocationControl、SingleFieldChooserControl、SingleFieldValueChooserControl、SingleItemChooserControl、SpinnerControl、StaticText、SystemControls、TabbedPanel、TableControl、TextAreaControl、TextBoxControl、TextBrowserPanel

子要素

Cell

Cell 要素**:**

表 *127. Cell* の属性

属性	使用	説明	有効な値
column	必須		*nonNegativeInteger*
row	必須		*nonNegativeInteger*
width	必須		*nonNegativeInteger*

XML 表記

```
<xs:element name="Cell">
  <xs:attribute name="row" type="xs:nonNegativeInteger" use="required"/>
  <xs:attribute name="column" type="xs:nonNegativeInteger" use="required"/>
  <xs:attribute name="width" type="xs:nonNegativeInteger" use="required"/>
</xs:element>
```

親要素

Layout

License 要素

システムが使用するために予約済みです。

XML 表記

```
<xs:element name="License">
  <xs:sequence minOccurs="0" maxOccurs="unbounded">
    <xs:element ref="OptionCode"/>
  </xs:sequence>
</xs:element>
```

親要素

Extension

子要素

OptionCode

ListValue 要素

値のシーケンス。すべての値は同じコンテンツ・タイプを持つ必要がありますが、これはチェックされません。

XML 表記

```
<xs:element name="ListValue" type="LIST-VALUE">
  <xs:group ref="PARAMETER-CONTENT" minOccurs="0" maxOccurs="unbounded">
    <xs:choice>
      <xs:element ref="MapValue"/>
      <xs:element ref="StructuredValue"/>
      <xs:element ref="ListValue"/>
      <xs:element ref="Value"/>
      <xs:element ref="DatabaseConnectionValue"/>
    </xs:choice>
  </xs:group>
</xs:element>
```

親要素

Attribute、Attribute、ListValue、ListValue、ListValue、Parameter

子要素

DatabaseConnectionValue、ListValue、MapValue、StructuredValue、Value

MapValue 要素

A set of map entries, each consisting if a key and a value.

XML 表記

```
<xs:element name="MapValue" type="MAP-VALUE">
  <xs:sequence>
    <xs:element name="MapEntry" type="MAP-ENTRY" maxOccurs="unbounded">
      <xs:sequence>
        <xs:element name="KeyValue" type="KEY-VALUE">
        </xs:element>
        <xs:element name="StructuredValue" type="STRUCTURED-VALUE">
          <xs:sequence>
            <xs:element name="Attribute" type="ATTRIBUTE" maxOccurs="unbounded">
              <xs:group ref="PARAMETER-CONTENT" minOccurs="0">
                <xs:choice>
                  <xs:element ref="MapValue"/>
                  <xs:element ref="StructuredValue"/>
                  <xs:element ref="ListValue"/>
                  <xs:element ref="Value"/>
                  <xs:element ref="DatabaseConnectionValue"/>
                </xs:choice>
              </xs:group>
              <xs:sequence>
                <xs:element name="ListValue" type="LIST-VALUE" minOccurs="0" maxOccurs="1">
                  <xs:group ref="PARAMETER-CONTENT" minOccurs="0" maxOccurs="unbounded">
                    <xs:choice>
                      <xs:element ref="MapValue"/>
                      <xs:element ref="StructuredValue"/>
                      <xs:element ref="ListValue"/>
                      <xs:element ref="Value"/>
                      <xs:element ref="DatabaseConnectionValue"/>
                    </xs:choice>
                  </xs:group>
                </xs:element>
              </xs:sequence>
            </xs:element>
          </xs:sequence>
        </xs:element>
      </xs:sequence>
    </xs:element>
  </xs:sequence>
</xs:element>
```

親要素

Attribute、Attribute、ListValue、ListValue、ListValue、Parameter

子要素

MapEntry

MapEntry 要素**:** キー・プロパティー・マップのエントリー各エントリーは、キーと関連する値で構成されています。

XML 表記

```
<xs:element name="MapEntry" type="MAP-ENTRY" maxOccurs="unbounded">
  <xs:sequence>
    <xs:element name="KeyValue" type="KEY-VALUE">
    </xs:element>
    <xs:element name="StructuredValue" type="STRUCTURED-VALUE">
      <xs:sequence>
        <xs:element name="Attribute" type="ATTRIBUTE" maxOccurs="unbounded">
          <xs:group ref="PARAMETER-CONTENT" minOccurs="0">
            <xs:choice>
              <xs:element ref="MapValue"/>
              <xs:element ref="StructuredValue"/>
              <xs:element ref="ListValue"/>
              <xs:element ref="Value"/>
              <xs:element ref="DatabaseConnectionValue"/>
            </xs:choice>
          </xs:group>
          <xs:sequence>
            <xs:element name="ListValue" type="LIST-VALUE" minOccurs="0" maxOccurs="1">
              <xs:group ref="PARAMETER-CONTENT" minOccurs="0" maxOccurs="unbounded">
                <xs:choice>
                  <xs:element ref="MapValue"/>
                  <xs:element ref="StructuredValue"/>
                  <xs:element ref="ListValue"/>
                  <xs:element ref="Value"/>
                  <xs:element ref="DatabaseConnectionValue"/>
                </xs:choice>
              </xs:group>
            </xs:element>
          </xs:sequence>
        </xs:element>
      </xs:sequence>
    </xs:element>
  </xs:sequence>
</xs:element>
```

親要素

MapValue

子要素

KeyValue、StructuredValue

KeyValue 要素**:** マップ・エントリー内のキー値。

表 *128. KeyValue* の属性

属性	使用	説明	有効な値
value	必須		*string*

XML 表記

```
<xs:element name="KeyValue" type="KEY-VALUE">
  <xs:attribute name="value" type="xs:string" use="required"/>
</xs:element>
```

親要素

MapEntry

StructuredValue 要素*:* 名前付きの値のシーケンス (「属性」)。

XML 表記

```
<xs:element name="StructuredValue" type="STRUCTURED-VALUE">
  <xs:sequence>
    <xs:element name="Attribute" type="ATTRIBUTE" maxOccurs="unbounded">
      <xs:group ref="PARAMETER-CONTENT" minOccurs="0">
        <xs:choice>
          <xs:element ref="MapValue"/>
          <xs:element ref="StructuredValue"/>
          <xs:element ref="ListValue"/>
          <xs:element ref="Value"/>
          <xs:element ref="DatabaseConnectionValue"/>
        </xs:choice>
      </xs:group>
      <xs:sequence>
        <xs:element name="ListValue" type="LIST-VALUE" minOccurs="0" maxOccurs="1">
          <xs:group ref="PARAMETER-CONTENT" minOccurs="0" maxOccurs="unbounded">
            <xs:choice>
              <xs:element ref="MapValue"/>
              <xs:element ref="StructuredValue"/>
              <xs:element ref="ListValue"/>
              <xs:element ref="Value"/>
              <xs:element ref="DatabaseConnectionValue"/>
            </xs:choice>
          </xs:group>
        </xs:element>
      </xs:sequence>
    </xs:element>
  </xs:sequence>
</xs:element>
```

親要素

MapEntry

子要素

属性

Attribute 要素*:*

表 *129. Attribute* の属性

属性	使用	説明	有効な値
name	必須		*string*
value	オプション		*string*

XML 表記

```
<xs:element name="Attribute" type="ATTRIBUTE" maxOccurs="unbounded">
  <xs:group ref="PARAMETER-CONTENT" minOccurs="0">
    <xs:choice>
      <xs:element ref="MapValue"/>
      <xs:element ref="StructuredValue"/>
      <xs:element ref="ListValue"/>
      <xs:element ref="Value"/>
      <xs:element ref="DatabaseConnectionValue"/>
    </xs:choice>
  </xs:group>
  <xs:sequence>
    <xs:element name="ListValue" type="LIST-VALUE" minOccurs="0" maxOccurs="1">
      <xs:group ref="PARAMETER-CONTENT" minOccurs="0" maxOccurs="unbounded">
        <xs:choice>
          <xs:element ref="MapValue"/>
          <xs:element ref="StructuredValue"/>
```

```
          <xs:element ref="ListValue"/>
          <xs:element ref="Value"/>
          <xs:element ref="DatabaseConnectionValue"/>
        </xs:choice>
      </xs:group>
    </xs:element>
  </xs:sequence>
  <xs:attribute name="name" type="xs:string" use="required"/>
  <xs:attribute name="value" type="xs:string"/>
</xs:element>
```

親要素

StructuredValue

子要素

DatabaseConnectionValue、ListValue、ListValue、MapValue、StructuredValue、Value

ListValue 要素*:* 値のシーケンス。すべての値は同じコンテンツ・タイプを持つ必要がありますが、これはチェックされません。

XML 表記

```
<xs:element name="ListValue" type="LIST-VALUE" minOccurs="0" maxOccurs="1">
  <xs:group ref="PARAMETER-CONTENT" minOccurs="0" maxOccurs="unbounded">
    <xs:choice>
      <xs:element ref="MapValue"/>
      <xs:element ref="StructuredValue"/>
      <xs:element ref="ListValue"/>
      <xs:element ref="Value"/>
      <xs:element ref="DatabaseConnectionValue"/>
    </xs:choice>
  </xs:group>
</xs:element>
```

親要素

属性

子要素

DatabaseConnectionValue、ListValue、MapValue、StructuredValue、Value

Menu 要素

メニューを定義します。メニューバー内の最上位の新規メニューである場合と、既存のメニューからのサブメニューである場合があります。

表 *130. Menu* の属性

属性	使用	説明	有効な値
ID	必須		*string*
label	必須		*string*
labelKey	オプション		*string*
mnemonic	オプション		*string*
mnemonicKey	オプション		*string*
offset	オプション		*nonNegativeInteger*
separatorAfter	オプション		*boolean*
separatorBefore	オプション		*boolean*
showIcon	オプション		*boolean*

表 *130. Menu* の属性 *(*続き*)*

属性	使用	説明	有効な値
showLabel	オプション		*boolean*
systemMenu	必須		**file** **edit** **insert** **view** **tools** **window** **help** **generate** **file.project** **file.outputs** **file.models** **edit.stream** **edit.node** **edit.outputs** **edit.models** **edit.project** **tools.repository** **tools.options** **tools.streamProperties**

XML 表記

```
<xs:element name="Menu">
  <xs:attribute name="id" type="xs:string" use="required"/>
  <xs:attribute name="label" type="xs:string" use="required"/>
  <xs:attribute name="labelKey" type="xs:string" use="optional"/>
  <xs:attribute name="mnemonic" type="xs:string" use="optional"/>
  <xs:attribute name="mnemonicKey" type="xs:string" use="optional"/>
  <xs:attribute name="systemMenu" type="STANDARD-MENU" use="required">
    <xs:enumeration value="file"/>
    <xs:enumeration value="edit"/>
    <xs:enumeration value="insert"/>
    <xs:enumeration value="view"/>
    <xs:enumeration value="tools"/>
    <xs:enumeration value="window"/>
    <xs:enumeration value="help"/>
    <xs:enumeration value="generate"/>
    <xs:enumeration value="file.project"/>
    <xs:enumeration value="file.outputs"/>
    <xs:enumeration value="file.models"/>
    <xs:enumeration value="edit.stream"/>
    <xs:enumeration value="edit.node"/>
    <xs:enumeration value="edit.outputs"/>
    <xs:enumeration value="edit.models"/>
    <xs:enumeration value="edit.project"/>
    <xs:enumeration value="tools.repository"/>
    <xs:enumeration value="tools.options"/>
    <xs:enumeration value="tools.streamProperties"/>
  </xs:attribute>
  <xs:attribute name="showLabel" type="xs:boolean" use="optional" default="true"/>
  <xs:attribute name="showIcon" type="xs:boolean" use="optional" default="false"/>
  <xs:attribute name="separatorBefore" type="xs:boolean" use="optional" default="false"/>
  <xs:attribute name="separatorAfter" type="xs:boolean" use="optional" default="false"/>
  <xs:attribute name="offset" type="xs:nonNegativeInteger" use="optional" default="0"/>
</xs:element>
```

親要素

コントロール

MenuItem 要素

メニューに追加できる項目を定義します。

表 *131. MenuItem* の属性

属性	使用	説明	有効な値
action	必須		*string*
customMenu	オプション		*string*
offset	オプション		*nonNegativeInteger*
separatorAfter	オプション		*boolean*
separatorBefore	オプション		*boolean*
showIcon	オプション		*boolean*
showLabel	オプション		*boolean*
systemMenu	オプション		**file** **edit** **insert** **view** **tools** **window** **help** **generate** **file.project** **file.outputs** **file.models** **edit.stream** **edit.node** **edit.outputs** **edit.models** **edit.project** **tools.repository** **tools.options** **tools.streamProperties**

XML 表記

```
<xs:element name="MenuItem">
  <xs:attribute name="action" type="xs:string" use="required"/>
  <xs:attribute name="systemMenu" type="STANDARD-MENU" use="optional">
    <xs:enumeration value="file"/>
    <xs:enumeration value="edit"/>
    <xs:enumeration value="insert"/>
    <xs:enumeration value="view"/>
    <xs:enumeration value="tools"/>
    <xs:enumeration value="window"/>
    <xs:enumeration value="help"/>
    <xs:enumeration value="generate"/>
    <xs:enumeration value="file.project"/>
    <xs:enumeration value="file.outputs"/>
    <xs:enumeration value="file.models"/>
    <xs:enumeration value="edit.stream"/>
    <xs:enumeration value="edit.node"/>
    <xs:enumeration value="edit.outputs"/>
    <xs:enumeration value="edit.models"/>
    <xs:enumeration value="edit.project"/>
    <xs:enumeration value="tools.repository"/>
    <xs:enumeration value="tools.options"/>
    <xs:enumeration value="tools.streamProperties"/>
  </xs:attribute>
  <xs:attribute name="customMenu" type="xs:string" use="optional"/>
```

```
  <xs:attribute name="showLabel" type="xs:boolean" use="optional" default="true"/>
  <xs:attribute name="showIcon" type="xs:boolean" use="optional" default="false"/>
  <xs:attribute name="separatorBefore" type="xs:boolean" use="optional" default="false"/>
  <xs:attribute name="separatorAfter" type="xs:boolean" use="optional" default="false"/>
  <xs:attribute name="offset" type="xs:nonNegativeInteger" use="optional" default="0"/>
</xs:element>
```

親要素

コントロール

MissingValues 要素

表 *132. MissingValues* の属性

属性	使用	説明	有効な値
treatNullAsMissing	オプション		*boolean*
treatWhitespaceAsMissing	オプション		*boolean*

XML 表記

```
<xs:element name="MissingValues" type="MISSING-VALUES" minOccurs="0">
  <xs:sequence>
    <xs:element name="Range" type="RANGE">
    </xs:element>
    <xs:element name="Values" type="FIELD-VALUES">
      <xs:sequence>
        <xs:element name="Value" type="FIELD-VALUE" minOccurs="0" maxOccurs="unbounded">
          <xs:sequence>
            <xs:element name="DisplayLabel" type="DISPLAY-LABEL" minOccurs="0" maxOccurs="unbounded">
            </xs:element>
          </xs:sequence>
        </xs:element>
      </xs:sequence>
    </xs:element>
  </xs:sequence>
  <xs:attribute name="treatNullAsMissing" type="xs:boolean" default="true"/>
  <xs:attribute name="treatWhitespaceAsMissing" type="xs:boolean" default="false"/>
</xs:element>
```

親要素

フィールド

子要素

Range、Values

Range 要素:

表 *133. Range* の属性

属性	使用	説明	有効な値
maxValue	必須		*string*
minValue	必須		*string*

XML 表記

```
<xs:element name="Range" type="RANGE">
  <xs:attribute name="minValue" type="xs:string" use="required"/>
  <xs:attribute name="maxValue" type="xs:string" use="required"/>
</xs:element>
```

親要素

MissingValues

Values 要素**:**

表 *134. Values* の属性

属性	使用	説明	有効な値
count	オプション		*nonNegativeInteger*

XML 表記

```
<xs:element name="Values" type="FIELD-VALUES">
  <xs:sequence>
    <xs:element name="Value" type="FIELD-VALUE" minOccurs="0" maxOccurs="unbounded">
      <xs:sequence>
        <xs:element name="DisplayLabel" type="DISPLAY-LABEL" minOccurs="0" maxOccurs="unbounded">
        </xs:element>
      </xs:sequence>
    </xs:element>
  </xs:sequence>
  <xs:attribute name="count" type="xs:nonNegativeInteger"/>
</xs:element>
```

親要素

MissingValues

子要素

Value

Value 要素***:***

表 *135. Value* の属性

属性	使用	説明	有効な値
code	必須		*integer*
displayLabel	オプション		*string*
flagValue	オプション		*boolean*
value	必須		*string*

XML 表記

```
<xs:element name="Value" type="FIELD-VALUE" minOccurs="0" maxOccurs="unbounded">
  <xs:sequence>
    <xs:element name="DisplayLabel" type="DISPLAY-LABEL" minOccurs="0" maxOccurs="unbounded">
    </xs:element>
  </xs:sequence>
  <xs:attribute name="value" type="xs:string" use="required"/>
  <xs:attribute name="code" type="xs:integer" use="required"/>
  <xs:attribute name="flagValue" type="xs:boolean"/>
  <xs:attribute name="displayLabel" type="xs:string"/>
</xs:element>
```

親要素

値

子要素

DisplayLabel

DisplayLabel *要素:* 指定された言語でのフィールドまたは値の表示ラベル。この displayLabel 属性は、特定の言語のラベルがない場合に使用できます。

表 *136. DisplayLabel* の属性

属性	使用	説明	有効な値
lang	オプション		*NMTOKEN*
value	必須		*string*

XML 表記

```
<xs:element name="DisplayLabel" type="DISPLAY-LABEL" minOccurs="0" maxOccurs="unbounded">
  <xs:attribute name="value" type="xs:string" use="required"/>
  <xs:attribute name="lang" type="xs:NMTOKEN" default="en"/>
</xs:element>
```

親要素

値

ModelBuilder 要素

表 *137. ModelBuilder* の属性

属性	使用	説明	有効な値
allowDateTimeInputs	オプション		*boolean*
allowNoInputs	オプション		*boolean*
allowNoOutputs	オプション		*boolean*
miningFunctions	必須		*any*
nullifyBlanks	オプション		*boolean*

XML 表記

```
<xs:element name="ModelBuilder">
  <xs:sequence>
    <xs:element name="Algorithm">
    </xs:element>
    <xs:element name="ModelingFields" minOccurs="0">
      <xs:sequence>
        <xs:element name="InputFields" minOccurs="0">
        </xs:element>
        <xs:element name="OutputFields" minOccurs="0">
        </xs:element>
      </xs:sequence>
    </xs:element>
    <xs:element name="ModelGeneration">
    </xs:element>
    <xs:element name="ModelFields">
      <xs:sequence minOccurs="0" maxOccurs="unbounded">
        <xs:group ref="DATA-MODEL-EXPRESSION">
          <xs:choice>
            <xs:element ref="ForEach"/>
            <xs:element ref="AddField"/>
            <xs:element ref="ChangeField"/>
            <xs:element ref="RemoveField"/>
          </xs:choice>
        </xs:group>
      </xs:sequence>
    </xs:element>
    <xs:element name="ModelEvaluation" minOccurs="0">
      <xs:sequence>
```

```
        <xs:element name="RawPropensity" minOccurs="0">
        </xs:element>
        <xs:element name="AdjustedPropensity" minOccurs="0">
        </xs:element>
        <xs:element name="VariableImportance" minOccurs="0">
        </xs:element>
      </xs:sequence>
    </xs:element>
    <xs:element name="AutoModeling" minOccurs="0">
      <xs:sequence>
        <xs:element name="SimpleSettings">
          <xs:sequence>
            <xs:element ref="PropertyGroup" maxOccurs="unbounded"/>
          </xs:sequence>
        </xs:element>
        <xs:element name="ExpertSettings" minOccurs="0">
          <xs:sequence>
            <xs:element ref="Condition" minOccurs="0"/>
            <xs:element ref="PropertyGroup" maxOccurs="unbounded"/>
          </xs:sequence>
        </xs:element>
        <xs:element name="PropertyMap" minOccurs="0">
          <xs:sequence>
            <xs:element name="PropertyMapping" maxOccurs="unbounded">
            </xs:element>
          </xs:sequence>
        </xs:element>
        <xs:element ref="Constraint" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:element>
  </xs:sequence>
  <xs:attribute name="miningFunctions" use="required"/>
  <xs:attribute name="allowNoInputs" type="xs:boolean" use="optional" default="false"/>
  <xs:attribute name="allowNoOutputs" type="xs:boolean" use="optional" default="false"/>
  <xs:attribute name="allowDateTimeInputs" type="xs:boolean" use="optional" default="false"/>
  <xs:attribute name="nullifyBlanks" type="xs:boolean" use="optional" default="true"/>
</xs:element>
```

親要素

ノード

子要素

Algorithm、AutoModeling、ModelEvaluation、ModelFields、ModelGeneration、ModelingFields

Algorithm 要素**:**

表 *138. Algorithm* の属性

属性	使用	説明	有効な値
label	オプション		*string*
labelKey	オプション		*string*
resourceKey	オプション		*string*
value	必須		*string*

XML 表記

```
<xs:element name="Algorithm">
  <xs:attribute name="value" type="xs:string" use="required"/>
  <xs:attribute name="label" type="xs:string" use="optional"/>
  <xs:attribute name="labelKey" type="xs:string" use="optional"/>
  <xs:attribute name="resourceKey" type="xs:string" use="optional"/>
</xs:element>
```

親要素

ModelBuilder

ModelingFields 要素:

表 *139. ModelingFields* の属性

属性	使用	説明	有効な値
controlsId	必須		*string*
ignoreBOTH	オプション		*boolean*

XML 表記

```
<xs:element name="ModelingFields" minOccurs="0">
  <xs:attribute name="controlsId" type="xs:string" use="required"/>
  <xs:sequence>
    <xs:element name="InputFields" minOccurs="0">
    </xs:element>
    <xs:element name="OutputFields" minOccurs="0">
    </xs:element>
  </xs:sequence>
  <xs:attribute name="ignoreBOTH" type="xs:boolean" use="optional" default="true"/>
</xs:element>
```

親要素

ModelBuilder

子要素

InputFields、OutputFields

InputFields* 要素*:

表 *140. InputFields* の属性

属性	使用	説明	有効な値
label	必須		*string*
labelKey	オプション		*string*
multiple	必須		*boolean*
onlyDatetime	オプション		*boolean*
onlyDiscrete	オプション		*boolean*
onlyNumeric	オプション		*boolean*
onlyRanges	オプション		*boolean*
onlySymbolic	オプション		*boolean*
property	必須		*string*
storage	オプション		*string*
types	オプション		*string*

XML 表記

```
<xs:element name="InputFields" minOccurs="0">
  <xs:attribute name="storage" type="xs:string" use="optional"/>
  <xs:attribute name="onlyNumeric" type="xs:boolean" use="optional"/>
  <xs:attribute name="onlySymbolic" type="xs:boolean" use="optional"/>
  <xs:attribute name="onlyDatetime" type="xs:boolean" use="optional"/>
  <xs:attribute name="types" type="xs:string" use="optional"/>
```

```
  <xs:attribute name="onlyRanges" type="xs:boolean" use="optional"/>
  <xs:attribute name="onlyDiscrete" type="xs:boolean" use="optional"/>
  <xs:attribute name="property" type="xs:string" use="required"/>
  <xs:attribute name="multiple" type="xs:boolean" use="required"/>
  <xs:attribute name="label" type="xs:string" use="required"/>
  <xs:attribute name="labelKey" type="xs:string" use="optional"/>
</xs:element>
```

親要素

ModelingFields

OutputFields 要素*:*

表 *141. OutputFields* の属性

属性	使用	説明	有効な値
label	必須		*string*
labelKey	オプション		*string*
multiple	必須		*boolean*
onlyDatetime	オプション		*boolean*
onlyDiscrete	オプション		*boolean*
onlyNumeric	オプション		*boolean*
onlyRanges	オプション		*boolean*
onlySymbolic	オプション		*boolean*
property	必須		*string*
storage	オプション		*string*
types	オプション		*string*

XML 表記

```
<xs:element name="OutputFields" minOccurs="0">
  <xs:attribute name="storage" type="xs:string" use="optional"/>
  <xs:attribute name="onlyNumeric" type="xs:boolean" use="optional"/>
  <xs:attribute name="onlySymbolic" type="xs:boolean" use="optional"/>
  <xs:attribute name="onlyDatetime" type="xs:boolean" use="optional"/>
  <xs:attribute name="types" type="xs:string" use="optional"/>
  <xs:attribute name="onlyRanges" type="xs:boolean" use="optional"/>
  <xs:attribute name="onlyDiscrete" type="xs:boolean" use="optional"/>
  <xs:attribute name="property" type="xs:string" use="required"/>
  <xs:attribute name="multiple" type="xs:boolean" use="required"/>
  <xs:attribute name="label" type="xs:string" use="required"/>
  <xs:attribute name="labelKey" type="xs:string" use="optional"/>
</xs:element>
```

親要素

ModelingFields

ModelGeneration 要素**:**

表 *142. ModelGeneration* の属性

属性	使用	説明	有効な値
controlsId	必須		*string*

XML 表記

```
<xs:element name="ModelGeneration">
  <xs:attribute name="controlsId" type="xs:string" use="required"/>
</xs:element>
```

親要素

ModelBuilder

ModelFields 要素**:**

XML 表記

```
<xs:element name="ModelFields">
  <xs:sequence minOccurs="0" maxOccurs="unbounded">
    <xs:group ref="DATA-MODEL-EXPRESSION">
      <xs:choice>
        <xs:element ref="ForEach"/>
        <xs:element ref="AddField"/>
        <xs:element ref="ChangeField"/>
        <xs:element ref="RemoveField"/>
      </xs:choice>
    </xs:group>
  </xs:sequence>
</xs:element>
```

親要素

ModelBuilder

子要素

AddField、ChangeField、ForEach、RemoveField

ModelEvaluation 要素**:**

表 *143. ModelEvaluation* の属性

属性	使用	説明	有効な値
container	必須		*any*
controlsId	必須		*string*
outputContainer	オプション		*any*

XML 表記

```
<xs:element name="ModelEvaluation" minOccurs="0">
  <xs:attribute name="controlsId" type="xs:string" use="required"/>
  <xs:sequence>
    <xs:element name="RawPropensity" minOccurs="0">
    </xs:element>
    <xs:element name="AdjustedPropensity" minOccurs="0">
    </xs:element>
    <xs:element name="VariableImportance" minOccurs="0">
    </xs:element>
  </xs:sequence>
  <xs:attribute name="container" use="required"/>
  <xs:attribute name="outputContainer" use="optional"/>
</xs:element>
```

親要素

ModelBuilder

子要素

AdjustedPropensity、RawPropensity、VariableImportance

RawPropensity 要素*:*

表 *144. RawPropensity* の属性

属性	使用	説明	有効な値
availabilityProperty	オプション		*string*
defaultValue	オプション		*boolean*
property	オプション		*string*

XML 表記

```
<xs:element name="RawPropensity" minOccurs="0">
  <xs:attribute name="property" type="xs:string" use="optional"/>
  <xs:attribute name="availabilityProperty" type="xs:string" use="optional"/>
  <xs:attribute name="defaultValue" type="xs:boolean" use="optional"/>
</xs:element>
```

親要素

ModelEvaluation

AdjustedPropensity 要素*:*

表 *145. AdjustedPropensity* の属性

属性	使用	説明	有効な値
availabilityProperty	オプション		*string*
defaultValue	オプション		*boolean*
property	オプション		*string*

XML 表記

```
<xs:element name="AdjustedPropensity" minOccurs="0">
  <xs:attribute name="property" type="xs:string" use="optional"/>
  <xs:attribute name="availabilityProperty" type="xs:string" use="optional"/>
  <xs:attribute name="defaultValue" type="xs:boolean" use="optional"/>
</xs:element>
```

親要素

ModelEvaluation

VariableImportance 要素*:*

表 *146. VariableImportance* の属性

属性	使用	説明	有効な値
availabilityProperty	オプション		*string*
defaultValue	オプション		*boolean*
property	オプション		*string*

XML 表記

```
<xs:element name="VariableImportance" minOccurs="0">
  <xs:attribute name="property" type="xs:string" use="optional"/>
  <xs:attribute name="availabilityProperty" type="xs:string" use="optional"/>
  <xs:attribute name="defaultValue" type="xs:boolean" use="optional"/>
</xs:element>
```

親要素

ModelEvaluation

AutoModeling 要素**:**

表 *147. AutoModeling* の属性

属性	使用	説明	有効な値
enabledByDefault	オプション		*any*
supportsAnalyticServer	オプション		*boolean*
supportsLargeData	オプション		*boolean*
supportsModelerServer	オプション		*boolean*

XML 表記

```
<xs:element name="AutoModeling" minOccurs="0">
  <xs:sequence>
    <xs:element name="SimpleSettings">
      <xs:sequence>
        <xs:element ref="PropertyGroup" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:element>
    <xs:element name="ExpertSettings" minOccurs="0">
      <xs:sequence>
        <xs:element ref="Condition" minOccurs="0"/>
        <xs:element ref="PropertyGroup" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:element>
    <xs:element name="PropertyMap" minOccurs="0">
      <xs:sequence>
        <xs:element name="PropertyMapping" maxOccurs="unbounded">
        </xs:element>
      </xs:sequence>
    </xs:element>
    <xs:element ref="Constraint" minOccurs="0" maxOccurs="unbounded"/>
  </xs:sequence>
  <xs:attribute name="enabledByDefault" use="optional"/>
  <xs:attribute name="supportsAnalyticServer" type="xs:boolean" use="optional"/>
  <xs:attribute name="supportsLargeData" type="xs:boolean" use="optional"/>
  <xs:attribute name="supportsModelerServer" type="xs:boolean" use="optional"/>
</xs:element>
```

親要素

ModelBuilder

子要素

Constraint、ExpertSettings、PropertyMap、SimpleSettings

SimpleSettings 要素***:***

XML 表記

```
<xs:element name="SimpleSettings">
  <xs:sequence>
    <xs:element ref="PropertyGroup" maxOccurs="unbounded"/>
  </xs:sequence>
</xs:element>
```

親要素

AutoModeling

子要素

PropertyGroup

ExpertSettings 要素*:*

XML 表記

```
<xs:element name="ExpertSettings" minOccurs="0">
  <xs:sequence>
    <xs:element ref="Condition" minOccurs="0"/>
    <xs:element ref="PropertyGroup" maxOccurs="unbounded"/>
  </xs:sequence>
</xs:element>
```

親要素

AutoModeling

子要素

Condition、PropertyGroup

PropertyMap 要素*:*

XML 表記

```
<xs:element name="PropertyMap" minOccurs="0">
  <xs:sequence>
    <xs:element name="PropertyMapping" maxOccurs="unbounded">
    </xs:element>
  </xs:sequence>
</xs:element>
```

親要素

AutoModeling

子要素

PropertyMapping

PropertyMapping 要素*:*

表 *148. PropertyMapping* の属性

属性	使用	説明	有効な値
property	必須		*string*
systemProperty	必須		*string*

XML 表記

```
<xs:element name="PropertyMapping" maxOccurs="unbounded">
  <xs:attribute name="property" type="xs:string" use="required"/>
  <xs:attribute name="systemProperty" type="xs:string" use="required"/>
</xs:element>
```

親要素

PropertyMap

ModelOutput 要素

表 *149. ModelOutput* の属性

属性	使用	説明	有効な値
delegate	オプション		*string*
deprecatedScriptNames	オプション		*string*
helpLink	オプション		*string*
ID	必須		*string*
label	オプション		*string*
labelKey	オプション		*string*
resourceKey	オプション		*string*
scriptName	オプション		*string*

XML 表記

```
<xs:element name="ModelOutput">
  <xs:sequence maxOccurs="unbounded">
    <xs:choice maxOccurs="unbounded">
      <xs:element ref="Properties"/>
      <xs:element name="Containers" minOccurs="0">
        <xs:sequence maxOccurs="unbounded">
          <xs:element ref="Container"/>
        </xs:sequence>
      </xs:element>
      <xs:element ref="UserInterface"/>
      <xs:element ref="Constructors" minOccurs="0"/>
      <xs:element ref="ModelProvider" minOccurs="0"/>
    </xs:choice>
  </xs:sequence>
  <xs:attribute name="id" type="xs:string" use="required"/>
  <xs:attribute name="scriptName" type="xs:string" use="optional"/>
  <xs:attribute name="deprecatedScriptNames" type="xs:string" use="optional"/>
  <xs:attribute name="delegate" type="xs:string" use="optional"/>
  <xs:attribute name="resourceKey" type="xs:string" use="optional"/>
  <xs:attribute name="label" type="xs:string" use="optional"/>
  <xs:attribute name="labelKey" type="xs:string" use="optional"/>
  <xs:attribute name="helpLink" type="xs:string" use="optional"/>
</xs:element>
```

親要素

Extension

子要素

Constructors、Containers、ModelProvider、Properties、UserInterface

関連する要素

DocumentOutput、InteractiveDocumentBuilder、InteractiveModelBuilder、Node

Containers 要素**:**

XML 表記

```
<xs:element name="Containers" minOccurs="0">
  <xs:sequence maxOccurs="unbounded">
    <xs:element ref="Container"/>
  </xs:sequence>
</xs:element>
```

親要素

ノード

子要素

Container

ModelProvider 要素

表 *150. ModelProvider* の属性

属性	使用	説明	有効な値
container	必須		*string*
isPMML	オプション		*boolean*

XML 表記

```
<xs:element name="ModelProvider">
  <xs:attribute name="container" type="xs:string" use="required"/>
  <xs:attribute name="isPMML" type="xs:boolean" use="optional" default="true"/>
</xs:element>
```

親要素

DocumentOutput、InteractiveDocumentBuilder、InteractiveModelBuilder、ModelOutput、Node

ModelType 要素

新規モデル・タイプを定義します。

表 *151. ModelType* の属性

属性	使用	説明	有効な値
content	オプション		*string*
format	必須		**utf8** **binary**
ID	必須		*string*
inputDirection	オプション		*string*
outputDirection	オプション		*string*
type	オプション		*string*

XML 表記

```
<xs:element name="ModelType">
  <xs:attribute name="id" type="xs:string" use="required"/>
  <xs:attribute name="format" use="required">
    <xs:simpleType>
      <xs:restriction base="xs:string">
        <xs:enumeration value="utf8"/>
        <xs:enumeration value="binary"/>
      </xs:restriction>
    </xs:simpleType>
  </xs:attribute>
  <xs:attribute name="content" type="xs:string" use="optional"/>
  <xs:attribute name="inputDirection" type="xs:string" use="optional" default="[in]"/>
  <xs:attribute name="outputDirection" type="xs:string" use="optional" default="[out]"/>
  <xs:attribute name="type" type="xs:string" use="optional"/>
</xs:element>
```

親要素

ContainerTypes

関連する要素

ContainerType、DocumentType

ModelViewerPanel 要素

表 *152. ModelViewerPanel* の属性

属性	使用	説明	有効な値
container	必須		*string*

XML 表記

```
<xs:element name="ModelViewerPanel">
  <xs:sequence>
    <xs:choice>
      <xs:element ref="Layout" minOccurs="0" maxOccurs="1"/>
      <xs:element ref="Enabled" minOccurs="0" maxOccurs="1"/>
      <xs:element ref="Visible" minOccurs="0" maxOccurs="1"/>
      <xs:element ref="Required" minOccurs="0" maxOccurs="1"/>
    </xs:choice>
  </xs:sequence>
  <xs:attribute name="container" type="xs:string" use="required"/>
</xs:element>
```

親要素

タブ

子要素

Enabled、Layout、Required、Visible

関連する要素

ActionButton、ComboBoxControl、ExtensionObjectPanel、FieldAllocationList、OutputViewerPanel、SelectorPanel、StaticText、SystemControls、TabbedPanel、TextBrowserPanel

Module 要素

表 *153. Module* の属性

属性	使用	説明	有効な値
libraryId	オプション		*string*
name	オプション		*string*
systemModule	オプション		**SmartScore** **TextReader** **TextWriter**

XML 表記

```
<xs:element name="Module">
  <xs:sequence>
    <xs:element ref="InputFiles"/>
    <xs:element ref="OutputFiles"/>
```

```
    <xs:element ref="StatusCodes" minOccurs="0" maxOccurs="1"/>
    <xs:group ref="SYSTEM-MODULE" minOccurs="0">
      <xs:choice>
        <xs:element ref="TextReader"/>
        <xs:element ref="TextWriter"/>
      </xs:choice>
    </xs:group>
  </xs:sequence>
  <xs:attribute name="systemModule" use="optional" default="SmartScore">
    <xs:simpleType>
      <xs:restriction base="xs:string">
        <xs:enumeration value="SmartScore"/>
        <xs:enumeration value="TextReader"/>
        <xs:enumeration value="TextWriter"/>
      </xs:restriction>
    </xs:simpleType>
  </xs:attribute>
  <xs:attribute name="libraryId" type="xs:string" use="optional"/>
  <xs:attribute name="name" type="xs:string" use="optional"/>
</xs:element>
```

親要素

Execution

子要素

InputFiles、OutputFiles、StatusCodes、TextReader、TextWriter

MultiFieldAllocationControl 要素

allocator 属性によって識別されるフィールド割り振りリスト コントロールから複数のフィールドを選択するために使用できるコントロールを定義します。

表 *154. MultiFieldAllocationControl* の属性

属性	使用	説明	有効な値
allocator	必須		*string*
buttonColumn	必須		*integer*
description	オプション		*string*
descriptionKey	オプション		*string*
label	オプション		*string*
labelAbove	オプション		*boolean*
labelKey	オプション		*string*
labelWidth	オプション		*positiveInteger*
mnemonic	オプション		*string*
mnemonicKey	オプション		*string*
onlyDatetime	オプション		*boolean*
onlyDiscrete	オプション		*boolean*
onlyNumeric	オプション		*boolean*
onlyRanges	オプション		*boolean*
onlySymbolic	オプション		*boolean*
property	必須		*string*
resourceKey	オプション		*string*
showLabel	オプション		*boolean*
storage	オプション		*string*
types	オプション		*string*

XML 表記

```
<xs:element name="MultiFieldAllocationControl">
  <xs:sequence>
    <xs:choice>
      <xs:element ref="Layout" minOccurs="0" maxOccurs="1"/>
      <xs:element ref="Enabled" minOccurs="0" maxOccurs="1"/>
      <xs:element ref="Visible" minOccurs="0" maxOccurs="1"/>
      <xs:element ref="Required" minOccurs="0" maxOccurs="1"/>
    </xs:choice>
  </xs:sequence>
  <xs:attribute name="property" type="xs:string" use="required"/>
  <xs:attribute name="showLabel" type="xs:boolean" use="optional" default="true"/>
  <xs:attribute name="resourceKey" type="xs:string" use="optional"/>
  <xs:attribute name="label" type="xs:string" use="optional"/>
  <xs:attribute name="labelKey" type="xs:string" use="optional"/>
  <xs:attribute name="mnemonic" type="xs:string" use="optional"/>
  <xs:attribute name="mnemonicKey" type="xs:string" use="optional"/>
  <xs:attribute name="labelWidth" type="xs:positiveInteger" use="optional" default="1"/>
  <xs:attribute name="labelAbove" type="xs:boolean" use="optional" default="false"/>
  <xs:attribute name="description" type="xs:string" use="optional"/>
  <xs:attribute name="descriptionKey" type="xs:string" use="optional"/>
  <xs:attribute name="allocator" type="xs:string" use="required"/>
  <xs:attribute name="buttonColumn" type="xs:integer" use="required"/>
  <xs:attribute name="storage" type="xs:string" use="optional"/>
  <xs:attribute name="onlyNumeric" type="xs:boolean" use="optional"/>
  <xs:attribute name="onlySymbolic" type="xs:boolean" use="optional"/>
  <xs:attribute name="onlyDatetime" type="xs:boolean" use="optional"/>
  <xs:attribute name="types" type="xs:string" use="optional"/>
  <xs:attribute name="onlyRanges" type="xs:boolean" use="optional"/>
  <xs:attribute name="onlyDiscrete" type="xs:boolean" use="optional"/>
</xs:element>
```

親要素

PropertiesPanel、PropertiesSubPanel

子要素

Enabled、Layout、Required、Visible

関連する要素

CheckBoxControl、CheckBoxGroupControl、ClientDirectoryChooserControl、ClientFileChooserControl、DBConnectionChooserControl、DBTableChooserControl、MultiFieldChooserControl、MultiFieldSelectionTableControl、PasswordBoxControl、PropertyControl、RadioButtonGroupControl、ServerDirectoryChooserControl、ServerFileChooserControl、SingleFieldAllocationControl、SingleFieldChooserControl、SingleFieldValueChooserControl、SpinnerControl、TableControl、TextAreaControl、TextBoxControl

MultiFieldChooserControl 要素

現在のデータ モデルの複数のフィールドを選択するために使用できるコントロールを定義します。

表 *155. MultiFieldChooserControl* の属性

属性	使用	説明	有効な値
description	オプション		*string*
descriptionKey	オプション		*string*
label	オプション		*string*
labelAbove	オプション		*boolean*
labelKey	オプション		*string*
labelWidth	オプション		*positiveInteger*

表 155. *MultiFieldChooserControl* の属性 (続き)

属性	使用	説明	有効な値
mnemonic	オプション		*string*
mnemonicKey	オプション		*string*
onlyDatetime	オプション		*boolean*
onlyDiscrete	オプション		*boolean*
onlyNumeric	オプション		*boolean*
onlyRanges	オプション		*boolean*
onlySymbolic	オプション		*boolean*
property	必須		*string*
resourceKey	オプション		*string*
showLabel	オプション		*boolean*
storage	オプション		*string*
types	オプション		*string*

XML 表記

```
<xs:element name="MultiFieldChooserControl">
  <xs:sequence>
    <xs:choice>
      <xs:element ref="Layout" minOccurs="0" maxOccurs="1"/>
      <xs:element ref="Enabled" minOccurs="0" maxOccurs="1"/>
      <xs:element ref="Visible" minOccurs="0" maxOccurs="1"/>
      <xs:element ref="Required" minOccurs="0" maxOccurs="1"/>
    </xs:choice>
  </xs:sequence>
  <xs:attribute name="property" type="xs:string" use="required"/>
  <xs:attribute name="showLabel" type="xs:boolean" use="optional" default="true"/>
  <xs:attribute name="resourceKey" type="xs:string" use="optional"/>
  <xs:attribute name="label" type="xs:string" use="optional"/>
  <xs:attribute name="labelKey" type="xs:string" use="optional"/>
  <xs:attribute name="mnemonic" type="xs:string" use="optional"/>
  <xs:attribute name="mnemonicKey" type="xs:string" use="optional"/>
  <xs:attribute name="labelWidth" type="xs:positiveInteger" use="optional" default="1"/>
  <xs:attribute name="labelAbove" type="xs:boolean" use="optional" default="false"/>
  <xs:attribute name="description" type="xs:string" use="optional"/>
  <xs:attribute name="descriptionKey" type="xs:string" use="optional"/>
  <xs:attribute name="storage" type="xs:string" use="optional"/>
  <xs:attribute name="onlyNumeric" type="xs:boolean" use="optional"/>
  <xs:attribute name="onlySymbolic" type="xs:boolean" use="optional"/>
  <xs:attribute name="onlyDatetime" type="xs:boolean" use="optional"/>
  <xs:attribute name="types" type="xs:string" use="optional"/>
  <xs:attribute name="onlyRanges" type="xs:boolean" use="optional"/>
  <xs:attribute name="onlyDiscrete" type="xs:boolean" use="optional"/>
</xs:element>
```

親要素

PropertiesPanel、PropertiesSubPanel

子要素

Enabled、Layout、Required、Visible

関連する要素

CheckBoxControl、CheckBoxGroupControl、ClientDirectoryChooserControl、ClientFileChooserControl、DBConnectionChooserControl、DBTableChooserControl、MultiFieldAllocationControl、MultiFieldSelectionTableControl、PasswordBoxControl、PropertyControl、RadioButtonGroupControl、ServerDirectoryChooserControl、ServerFileChooserControl、SingleFieldAllocationControl、

SingleFieldChooserControl、SingleFieldValueChooserControl、SpinnerControl、TableControl、TextAreaControl、TextBoxControl

MultiFieldSelectionTableControl 要素

現在のデータ モデルのフィールドを選択するためのボタンを提供するテーブル パネル コントロールを定義します。ユーザーは controlClass 属性を介して JTable 実装を指定する必要があります。JTable は PropertyControl と CustomFieldSelectionTable を実装する必要があり、テーブル モデルは CustomFieldSelectionTableModel を実装する必要があります。

表 *156. MultiFieldSelectionTableControl* の属性

属性	使用	説明	有効な値
controlClass	必須		*string*
description	オプション		*string*
descriptionKey	オプション		*string*
excludes	オプション	コンテンツがブロック対象であるフィールド ターゲット プロパティー名のリスト。	*string*
includes	オプション	コンテンツがデータ ソースであるフィールド ターゲット プロパティー名のリスト。	*string*
label	オプション		*string*
labelAbove	オプション		*boolean*
labelKey	オプション		*string*
labelWidth	オプション		*positiveInteger*
mnemonic	オプション		*string*
mnemonicKey	オプション		*string*
onlyDatetime	オプション		*boolean*
onlyDiscrete	オプション		*boolean*
onlyNumeric	オプション		*boolean*
onlyRanges	オプション		*boolean*
onlySymbolic	オプション		*boolean*
property	必須		*string*
resourceKey	オプション		*string*
showLabel	オプション		*boolean*
storage	オプション		*string*
types	オプション		*string*

XML 表記

```
<xs:element name="MultiFieldSelectionTableControl">
  <xs:sequence>
    <xs:choice>
      <xs:element ref="Layout" minOccurs="0" maxOccurs="1"/>
      <xs:element ref="Enabled" minOccurs="0" maxOccurs="1"/>
      <xs:element ref="Visible" minOccurs="0" maxOccurs="1"/>
      <xs:element ref="Required" minOccurs="0" maxOccurs="1"/>
    </xs:choice>
  </xs:sequence>
```

```
  <xs:attribute name="property" type="xs:string" use="required"/>
  <xs:attribute name="showLabel" type="xs:boolean" use="optional" default="true"/>
  <xs:attribute name="resourceKey" type="xs:string" use="optional"/>
  <xs:attribute name="label" type="xs:string" use="optional"/>
  <xs:attribute name="labelKey" type="xs:string" use="optional"/>
  <xs:attribute name="mnemonic" type="xs:string" use="optional"/>
  <xs:attribute name="mnemonicKey" type="xs:string" use="optional"/>
  <xs:attribute name="labelWidth" type="xs:positiveInteger" use="optional" default="1"/>
  <xs:attribute name="labelAbove" type="xs:boolean" use="optional" default="false"/>
  <xs:attribute name="description" type="xs:string" use="optional"/>
  <xs:attribute name="descriptionKey" type="xs:string" use="optional"/>
  <xs:attribute name="controlClass" type="xs:string" use="required"/>
  <xs:attribute name="storage" type="xs:string" use="optional"/>
  <xs:attribute name="onlyNumeric" type="xs:boolean" use="optional"/>
  <xs:attribute name="onlySymbolic" type="xs:boolean" use="optional"/>
  <xs:attribute name="onlyDatetime" type="xs:boolean" use="optional"/>
  <xs:attribute name="types" type="xs:string" use="optional"/>
  <xs:attribute name="onlyRanges" type="xs:boolean" use="optional"/>
  <xs:attribute name="onlyDiscrete" type="xs:boolean" use="optional"/>
  <xs:attribute name="includes" type="xs:string" use="optional"/>
  <xs:attribute name="excludes" type="xs:string" use="optional"/>
</xs:element>
```

親要素

PropertiesPanel、PropertiesSubPanel

子要素

Enabled、Layout、Required、Visible

関連する要素

CheckBoxControl、CheckBoxGroupControl、ClientDirectoryChooserControl、ClientFileChooserControl、DBConnectionChooserControl、DBTableChooserControl、MultiFieldAllocationControl、MultiFieldChooserControl、PasswordBoxControl、PropertyControl、RadioButtonGroupControl、ServerDirectoryChooserControl、ServerFileChooserControl、SingleFieldAllocationControl、SingleFieldChooserControl、SingleFieldValueChooserControl、SpinnerControl、TableControl、TextAreaControl、TextBoxControl

MultiItemChooserControl 要素

選択項目から複数の値を選択するために使用できるコントロールを定義します。

表 *157. MultiItemChooserControl* の属性

属性	使用	説明	有効な値
catalog	必須		*string*
description	オプション		*string*
descriptionKey	オプション		*string*
label	オプション		*string*
labelAbove	オプション		*boolean*
labelKey	オプション		*string*
labelWidth	オプション		*positiveInteger*
mnemonic	オプション		*string*
mnemonicKey	オプション		*string*
property	必須		*string*
resourceKey	オプション		*string*
showLabel	オプション		*boolean*

XML 表記

```
<xs:element name="MultiItemChooserControl">
  <xs:sequence>
    <xs:choice>
      <xs:element ref="Layout" minOccurs="0" maxOccurs="1"/>
      <xs:element ref="Enabled" minOccurs="0" maxOccurs="1"/>
      <xs:element ref="Visible" minOccurs="0" maxOccurs="1"/>
      <xs:element ref="Required" minOccurs="0" maxOccurs="1"/>
    </xs:choice>
  </xs:sequence>
  <xs:attribute name="property" type="xs:string" use="required"/>
  <xs:attribute name="showLabel" type="xs:boolean" use="optional" default="true"/>
  <xs:attribute name="resourceKey" type="xs:string" use="optional"/>
  <xs:attribute name="label" type="xs:string" use="optional"/>
  <xs:attribute name="labelKey" type="xs:string" use="optional"/>
  <xs:attribute name="mnemonic" type="xs:string" use="optional"/>
  <xs:attribute name="mnemonicKey" type="xs:string" use="optional"/>
  <xs:attribute name="labelWidth" type="xs:positiveInteger" use="optional" default="1"/>
  <xs:attribute name="labelAbove" type="xs:boolean" use="optional" default="false"/>
  <xs:attribute name="description" type="xs:string" use="optional"/>
  <xs:attribute name="descriptionKey" type="xs:string" use="optional"/>
  <xs:attribute name="catalog" type="xs:string" use="required"/>
</xs:element>
```

親要素

PropertiesPanel、PropertiesSubPanel

子要素

Enabled、Layout、Required、Visible

関連する要素

SingleItemChooserControl

Node 要素

表 *158. Node* の属性

属性	使用	説明	有効な値
delegate	オプション		*string*
deprecatedScriptNames	オプション		*string*
description	オプション		*string*
descriptionKey	オプション		*string*
helpLink	オプション		*string*
ID	必須		*string*
label	必須		*string*
labelKey	オプション		*string*

表 *158. Node* の属性 (続き)

属性	使用	説明	有効な値
palette	オプション		**import** **fieldOp** **recordOp** **modeling** **dbModeling** **graph** **output** **export** **modeling.classification** **modeling.association** **modeling.segmentation** **modeling.auto**
relativePosition	オプション		*string*
relativeTo	オプション		*string*
resourceKey	オプション		*string*
scriptName	オプション		*string*
type	必須		**dataReader** **dataWriter** **dataTransformer** **modelApplier** **modelBuilder** **documentBuilder**

XML 表記

```
<xs:element name="Node">
  <xs:sequence maxOccurs="unbounded">
    <xs:choice maxOccurs="unbounded">
      <xs:element ref="Properties"/>
      <xs:element name="Containers" minOccurs="0">
        <xs:sequence maxOccurs="unbounded">
          <xs:element ref="Container"/>
        </xs:sequence>
      </xs:element>
      <xs:element ref="UserInterface"/>
      <xs:element ref="Constructors" minOccurs="0"/>
      <xs:element ref="ModelProvider" minOccurs="0"/>
    </xs:choice>
  </xs:sequence>
  <xs:attribute name="id" type="xs:string" use="required"/>
  <xs:attribute name="scriptName" type="xs:string" use="optional"/>
  <xs:attribute name="deprecatedScriptNames" type="xs:string" use="optional"/>
  <xs:attribute name="delegate" type="xs:string" use="optional"/>
  <xs:attribute name="resourceKey" type="xs:string" use="optional"/>
  <xs:sequence>
    <xs:element ref="ModelBuilder" minOccurs="0"/>
    <xs:element ref="DocumentBuilder" minOccurs="0"/>
    <xs:element ref="Execution"/>
    <xs:element ref="OutputDataModel" minOccurs="0"/>
    <xs:element ref="Validations" minOccurs="0"/>
  </xs:sequence>
  <xs:attribute name="type" type="NODE-TYPE" use="required">
    <xs:enumeration value="dataReader"/>
    <xs:enumeration value="dataWriter"/>
    <xs:enumeration value="dataTransformer"/>
    <xs:enumeration value="modelApplier"/>
    <xs:enumeration value="modelBuilder"/>
    <xs:enumeration value="documentBuilder"/>
  </xs:attribute>
```

```
  <xs:attribute name="label" type="xs:string" use="required"/>
  <xs:attribute name="labelKey" type="xs:string" use="optional"/>
  <xs:attribute name="description" type="xs:string" use="optional"/>
  <xs:attribute name="descriptionKey" type="xs:string" use="optional"/>
  <xs:attribute name="palette" type="SYSTEM-PALETTE" use="optional">
    <xs:enumeration value="import"/>
    <xs:enumeration value="fieldOp"/>
    <xs:enumeration value="recordOp"/>
    <xs:enumeration value="modeling"/>
    <xs:enumeration value="dbModeling"/>
    <xs:enumeration value="graph"/>
    <xs:enumeration value="output"/>
    <xs:enumeration value="export"/>
    <xs:enumeration value="modeling.classification"/>
    <xs:enumeration value="modeling.association"/>
    <xs:enumeration value="modeling.segmentation"/>
    <xs:enumeration value="modeling.auto"/>
  </xs:attribute>
  <xs:attribute name="helpLink" type="xs:string" use="optional"/>
  <xs:attribute name="relativeTo" type="xs:string" use="optional"/>
  <xs:attribute name="relativePosition" type="xs:string" use="optional"/>
</xs:element>
```

親要素

Extension

子要素

Constructors、Containers、DocumentBuilder、Execution、ModelBuilder、ModelProvider、OutputDataModel、Properties、UserInterface、Validations

関連する要素

DocumentOutput、InteractiveDocumentBuilder、InteractiveModelBuilder、ModelOutput

Containers 要素**:**

XML 表記

```
<xs:element name="Containers" minOccurs="0">
  <xs:sequence maxOccurs="unbounded">
    <xs:element ref="Container"/>
  </xs:sequence>
</xs:element>
```

親要素

ノード

子要素

Container

Not 要素

XML 表記

```
<xs:element name="Not">
  <xs:sequence>
    <xs:group ref="CONDITION-EXPRESSION">
      <xs:choice>
        <xs:element ref="Condition"/>
        <xs:element ref="And"/>
        <xs:element ref="Or"/>
        <xs:element ref="Not"/>
```

```
      </xs:choice>
    </xs:group>
  </xs:sequence>
</xs:element>
```

親要素

And、Command、Constraint、CreateContainer、CreateDocument、CreateDocumentOutput、CreateInteractiveDocumentBuilder、CreateInteractiveModelBuilder、CreateModel、CreateModelApplier、CreateModelOutput、Enabled、Not、Option、Or、Required、Run、Validation、Visible

子要素

And、Condition、Not、Or

NumberFormat 要素

数値型フィールドの形式情報を定義します。

表 *159. NumberFormat* の属性

属性	使用	説明	有効な値
decimalPlaces	必須		*nonNegativeInteger*
decimalSymbol	必須		**period** **comma**
formatType	必須		**standard** **scientific** **currency**
groupingSymbol	必須		**none** **period** **comma** **space**
name	必須		*string*

XML 表記

```
<xs:element name="NumberFormat" type="NUMBER-FORMAT-DECLARATION">
  <xs:attribute name="name" type="xs:string" use="required"/>
  <xs:attribute name="formatType" type="NUMBER-FORMAT-TYPE" use="required">
    <xs:enumeration value="standard"/>
    <xs:enumeration value="scientific"/>
    <xs:enumeration value="currency"/>
  </xs:attribute>
  <xs:attribute name="decimalPlaces" type="xs:nonNegativeInteger" use="required"/>
  <xs:attribute name="decimalSymbol" type="DECIMAL-SYMBOL" use="required">
    <xs:enumeration value="period"/>
    <xs:enumeration value="comma"/>
  </xs:attribute>
  <xs:attribute name="groupingSymbol" type="NUMBER-GROUPING-SYMBOL" use="required">
    <xs:enumeration value="none"/>
    <xs:enumeration value="period"/>
    <xs:enumeration value="comma"/>
    <xs:enumeration value="space"/>
  </xs:attribute>
</xs:element>
```

NumericInfo 要素

表 *160. NumericInfo* の属性

属性	使用	説明	有効な値
平均	オプション		*double*
standardDeviation	オプション		*double*

XML 表記

```
<xs:element name="NumericInfo">
  <xs:attribute name="mean" type="xs:double"/>
  <xs:attribute name="standardDeviation" type="xs:double"/>
</xs:element>
```

親要素

AddField、ChangeField、Field

Option 要素

表 *161. Option* の属性

属性	使用	説明	有効な値
ifProperty	オプション		*string*
unlessProperty	オプション		*string*
value	必須		

XML 表記

```
<xs:element name="Option">
  <xs:sequence>
    <xs:group ref="CONDITION-EXPRESSION" minOccurs="0">
      <xs:choice>
        <xs:element ref="Condition"/>
        <xs:element ref="And"/>
        <xs:element ref="Or"/>
        <xs:element ref="Not"/>
      </xs:choice>
    </xs:group>
  </xs:sequence>
  <xs:attribute name="value" type="EVALUATED-STRING" use="required"/>
  <xs:attribute name="ifProperty" type="xs:string" use="optional"/>
  <xs:attribute name="unlessProperty" type="xs:string" use="optional"/>
</xs:element>
```

親要素

Run

子要素

And、Condition、Not、Or

OptionCode 要素

表 *162. OptionCode* の属性

属性	使用	説明	有効な値
code	オプション		*long*
description	オプション		*string*

表 *162. OptionCode* の属性 *(*続き*)*

属性	使用	説明	有効な値
type	オプション		必須 オプション

XML 表記

```
<xs:element name="OptionCode">
  <xs:attribute name="code" type="xs:long"/>
  <xs:attribute name="type" type="LicenseType">
    <xs:enumeration value="mandatory"/>
    <xs:enumeration value="optional"/>
  </xs:attribute>
  <xs:attribute name="description" type="xs:string"/>
</xs:element>
```

親要素

License

Or 要素

XML 表記

```
<xs:element name="Or">
  <xs:sequence minOccurs="2" maxOccurs="unbounded">
    <xs:group ref="CONDITION-EXPRESSION">
      <xs:choice>
        <xs:element ref="Condition"/>
        <xs:element ref="And"/>
        <xs:element ref="Or"/>
        <xs:element ref="Not"/>
      </xs:choice>
    </xs:group>
  </xs:sequence>
</xs:element>
```

親要素

And、Command、Constraint、CreateContainer、CreateDocument、CreateDocumentOutput、CreateInteractiveDocumentBuilder、CreateInteractiveModelBuilder、CreateModel、CreateModelApplier、CreateModelOutput、Enabled、Not、Option、Or、Required、Run、Validation、Visible

子要素

And、Condition、Not、Or

OutputDataModel 要素

表 *163. OutputDataModel* の属性

属性	使用	説明	有効な値
libraryId	オプション		*string*
method	オプション		**xml** **delegate** **dataModelProvider** **sharedLibrary**

表 163. *OutputDataModel* の属性 (続き)

属性	使用	説明	有効な値
最頻値	オプション		**fixed** **modify** **extend** **replace**
providerClass	オプション		*string*

XML 表記

```
<xs:element name="OutputDataModel">
  <xs:attribute name="mode" use="optional" default="fixed">
    <xs:simpleType>
      <xs:restriction base="xs:string">
        <xs:enumeration value="fixed"/>
        <xs:enumeration value="modify"/>
        <xs:enumeration value="extend"/>
        <xs:enumeration value="replace"/>
      </xs:restriction>
    </xs:simpleType>
  </xs:attribute>
  <xs:attribute name="method" use="optional" default="xml">
    <xs:simpleType>
      <xs:restriction base="xs:string">
        <xs:enumeration value="xml"/>
        <xs:enumeration value="delegate"/>
        <xs:enumeration value="dataModelProvider"/>
        <xs:enumeration value="sharedLibrary"/>
      </xs:restriction>
    </xs:simpleType>
  </xs:attribute>
  <xs:attribute name="providerClass" type="xs:string" use="optional"/>
  <xs:attribute name="libraryId" type="xs:string" use="optional"/>
</xs:element>
```

親要素

ノード

OutputFiles 要素

XML 表記

```
<xs:element name="OutputFiles">
  <xs:group ref="RUNTIME-FILES">
    <xs:sequence>
      <xs:element ref="DataFile"/>
      <xs:element ref="ContainerFile" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:group>
</xs:element>
```

親要素

Execution、Module

子要素

ContainerFile、DataFile

OutputViewerPanel 要素

表 *164. OutputViewerPanel* の属性

属性	使用	説明	有効な値
container	必須		*string*

XML 表記

```
<xs:element name="OutputViewerPanel">
  <xs:sequence>
    <xs:choice>
      <xs:element ref="Layout" minOccurs="0" maxOccurs="1"/>
      <xs:element ref="Enabled" minOccurs="0" maxOccurs="1"/>
      <xs:element ref="Visible" minOccurs="0" maxOccurs="1"/>
      <xs:element ref="Required" minOccurs="0" maxOccurs="1"/>
    </xs:choice>
  </xs:sequence>
  <xs:attribute name="container" type="xs:string" use="required"/>
</xs:element>
```

親要素

タブ

子要素

Enabled、Layout、Required、Visible

関連する要素

ActionButton、ComboBoxControl、ExtensionObjectPanel、FieldAllocationList、ModelViewerPanel、SelectorPanel、StaticText、SystemControls、TabbedPanel、TextBrowserPanel

Palette 要素

表 *165. Palette* の属性

属性	使用	説明	有効な値
description	オプション		*string*
descriptionKey	オプション		*string*
ID	必須		*string*
label	必須		*string*
labelKey	オプション		*string*
position	オプション		**atStart** **atEnd** **before** **after**
resourceKey	オプション		*string*

表 *165. Palette* の属性 (続き)

属性	使用	説明	有効な値
systemPalette	オプション		**import** **fieldOp** **recordOp** **modeling** **dbModeling** **graph** **output** **export** **modeling.classification** **modeling.association** **modeling.segmentation** **modeling.auto**

XML 表記

```
<xs:element name="Palette">
  <xs:sequence>
    <xs:element ref="Icon"/>
  </xs:sequence>
  <xs:attribute name="id" type="xs:string" use="required"/>
  <xs:attribute name="label" type="xs:string" use="required"/>
  <xs:attribute name="labelKey" type="xs:string" use="optional"/>
  <xs:attribute name="description" type="xs:string" use="optional"/>
  <xs:attribute name="descriptionKey" type="xs:string" use="optional"/>
  <xs:attribute name="resourceKey" type="xs:string" use="optional"/>
  <xs:attribute name="position" use="optional">
    <xs:simpleType>
      <xs:restriction base="xs:string">
        <xs:enumeration value="atStart"/>
        <xs:enumeration value="atEnd"/>
        <xs:enumeration value="before"/>
        <xs:enumeration value="after"/>
      </xs:restriction>
    </xs:simpleType>
  </xs:attribute>
  <xs:attribute name="systemPalette" type="SYSTEM-PALETTE" use="optional">
    <xs:enumeration value="import"/>
    <xs:enumeration value="fieldOp"/>
    <xs:enumeration value="recordOp"/>
    <xs:enumeration value="modeling"/>
    <xs:enumeration value="dbModeling"/>
    <xs:enumeration value="graph"/>
    <xs:enumeration value="output"/>
    <xs:enumeration value="export"/>
    <xs:enumeration value="modeling.classification"/>
    <xs:enumeration value="modeling.association"/>
    <xs:enumeration value="modeling.segmentation"/>
    <xs:enumeration value="modeling.auto"/>
  </xs:attribute>
</xs:element>
```

子要素

Icon

Parameters 要素

拡張ノードからの設定パラメーター。

表 *166. Parameters* の属性

属性	使用	説明	有効な値
count	オプション		*nonNegativeInteger*

XML 表記

```
<xs:element name="Parameters" type="PARAMETERS">
  <xs:sequence>
    <xs:element name="Parameter" type="PARAMETER" minOccurs="0" maxOccurs="unbounded">
      <xs:group ref="PARAMETER-CONTENT" minOccurs="0">
        <xs:choice>
          <xs:element ref="MapValue"/>
          <xs:element ref="StructuredValue"/>
          <xs:element ref="ListValue"/>
          <xs:element ref="Value"/>
          <xs:element ref="DatabaseConnectionValue"/>
        </xs:choice>
      </xs:group>
    </xs:element>
  </xs:sequence>
  <xs:attribute name="count" type="xs:nonNegativeInteger"/>
</xs:element>
```

子要素

パラメーター

Parameter 要素: パラメーターには、名前と値があります。単純値は値属性で表現できます。複合値は ParameterContent によって記述されるコンテンツ・モデルを使用します。この属性とコンテンツの組み合わせは、ネストされた値に対して繰り返されます。

表 *167. Parameter* の属性

属性	使用	説明	有効な値
name	必須		*string*
value	オプション		*string*

XML 表記

```
<xs:element name="Parameter" type="PARAMETER" minOccurs="0" maxOccurs="unbounded">
  <xs:group ref="PARAMETER-CONTENT" minOccurs="0">
    <xs:choice>
      <xs:element ref="MapValue"/>
      <xs:element ref="StructuredValue"/>
      <xs:element ref="ListValue"/>
      <xs:element ref="Value"/>
      <xs:element ref="DatabaseConnectionValue"/>
    </xs:choice>
  </xs:group>
  <xs:attribute name="name" type="xs:string" use="required"/>
  <xs:attribute name="value" type="xs:string"/>
</xs:element>
```

親要素

パラメーター

子要素

DatabaseConnectionValue、ListValue、MapValue、StructuredValue、Value

PasswordBoxControl 要素

文字列値を変更する (ただし、この値は表示されない) ために使用できる単一行パスワード コントロールを定義します。

表 *168. PasswordBoxControl* の属性

属性	使用	説明	有効な値
columns	オプション		*positiveInteger*

表 *168. PasswordBoxControl* の属性 (続き)

属性	使用	説明	有効な値
description	オプション		*string*
descriptionKey	オプション		*string*
label	オプション		*string*
labelAbove	オプション		*boolean*
labelKey	オプション		*string*
labelWidth	オプション		*positiveInteger*
mnemonic	オプション		*string*
mnemonicKey	オプション		*string*
property	必須		*string*
resourceKey	オプション		*string*
showLabel	オプション		*boolean*

XML 表記

```
<xs:element name="PasswordBoxControl">
  <xs:sequence>
    <xs:choice>
      <xs:element ref="Layout" minOccurs="0" maxOccurs="1"/>
      <xs:element ref="Enabled" minOccurs="0" maxOccurs="1"/>
      <xs:element ref="Visible" minOccurs="0" maxOccurs="1"/>
      <xs:element ref="Required" minOccurs="0" maxOccurs="1"/>
    </xs:choice>
  </xs:sequence>
  <xs:attribute name="property" type="xs:string" use="required"/>
  <xs:attribute name="showLabel" type="xs:boolean" use="optional" default="true"/>
  <xs:attribute name="resourceKey" type="xs:string" use="optional"/>
  <xs:attribute name="label" type="xs:string" use="optional"/>
  <xs:attribute name="labelKey" type="xs:string" use="optional"/>
  <xs:attribute name="mnemonic" type="xs:string" use="optional"/>
  <xs:attribute name="mnemonicKey" type="xs:string" use="optional"/>
  <xs:attribute name="labelWidth" type="xs:positiveInteger" use="optional" default="1"/>
  <xs:attribute name="labelAbove" type="xs:boolean" use="optional" default="false"/>
  <xs:attribute name="description" type="xs:string" use="optional"/>
  <xs:attribute name="descriptionKey" type="xs:string" use="optional"/>
  <xs:attribute name="columns" type="xs:positiveInteger" use="optional" default="20"/>
</xs:element>
```

親要素

PropertiesPanel、PropertiesSubPanel

子要素

Enabled、Layout、Required、Visible

関連する要素

CheckBoxControl、CheckBoxGroupControl、ClientDirectoryChooserControl、ClientFileChooserControl、DBConnectionChooserControl、DBTableChooserControl、MultiFieldAllocationControl、MultiFieldChooserControl、MultiFieldSelectionTableControl、PropertyControl、RadioButtonGroupControl、ServerDirectoryChooserControl、ServerFileChooserControl、SingleFieldAllocationControl、SingleFieldChooserControl、SingleFieldValueChooserControl、SpinnerControl、TableControl、TextAreaControl、TextBoxControl

Properties 要素

XML 表記

```
<xs:element name="Properties">
  <xs:sequence>
    <xs:element ref="Property" minOccurs="0" maxOccurs="unbounded"/>
  </xs:sequence>
</xs:element>
```

親要素

DocumentOutput、Execution、InteractiveDocumentBuilder、InteractiveModelBuilder、ModelOutput、Node

子要素

Property

PropertiesPanel 要素

最上位のプロパティー パネルを定義します。

表 *169. PropertiesPanel* の属性

属性	使用	説明	有効な値
ID	オプション		*string*
label	オプション		*string*
labelKey	オプション		*string*

XML 表記

```
<xs:element name="PropertiesPanel">
  <xs:sequence>
    <xs:choice>
      <xs:element ref="Layout" minOccurs="0" maxOccurs="1"/>
      <xs:element ref="Enabled" minOccurs="0" maxOccurs="1"/>
      <xs:element ref="Visible" minOccurs="0" maxOccurs="1"/>
      <xs:element ref="Required" minOccurs="0" maxOccurs="1"/>
    </xs:choice>
  </xs:sequence>
  <xs:sequence maxOccurs="unbounded">
    <xs:choice>
      <xs:element ref="CheckBoxControl"/>
      <xs:element ref="TextBoxControl"/>
      <xs:element ref="PasswordBoxControl"/>
      <xs:element ref="TextAreaControl"/>
      <xs:element ref="RadioButtonGroupControl"/>
      <xs:element ref="CheckBoxGroupControl"/>
      <xs:element ref="ComboBoxControl"/>
      <xs:element ref="SpinnerControl"/>
      <xs:element ref="ServerFileChooserControl"/>
      <xs:element ref="ServerDirectoryChooserControl"/>
      <xs:element ref="ClientFileChooserControl"/>
      <xs:element ref="ClientDirectoryChooserControl"/>
      <xs:element ref="TableControl"/>
      <xs:element ref="SingleFieldChooserControl"/>
      <xs:element ref="SingleFieldAllocationControl"/>
      <xs:element ref="MultiFieldChooserControl"/>
      <xs:element ref="MultiFieldAllocationControl"/>
      <xs:element ref="MultiFieldSelectionTableControl"/>
      <xs:element ref="SingleFieldValueChooserControl"/>
      <xs:element ref="SingleItemChooserControl"/>
      <xs:element ref="MultiItemChooserControl"/>
      <xs:element ref="DBConnectionChooserControl"/>
      <xs:element ref="DBTableChooserControl"/>
      <xs:element ref="PropertyControl"/>
      <xs:element ref="StaticText"/>
      <xs:element ref="SystemControls"/>
      <xs:element ref="ActionButton"/>
      <xs:element ref="FieldAllocationList"/>
```

```
      <xs:element ref="PropertiesPanel"/>
      <xs:element ref="PropertiesSubPanel"/>
      <xs:element ref="SelectorPanel"/>
      <xs:element ref="ExtensionObjectPanel"/>
      <xs:element ref="TabbedPanel"/>
    </xs:choice>
  </xs:sequence>
  <xs:attribute name="id" type="xs:string" use="optional"/>
  <xs:attribute name="label" type="xs:string" use="optional"/>
  <xs:attribute name="labelKey" type="xs:string" use="optional"/>
</xs:element>
```

親要素

PropertiesPanel、PropertiesSubPanel、Tab

子要素

ActionButton、CheckBoxControl、CheckBoxGroupControl、ClientDirectoryChooserControl、ClientFileChooserControl、ComboBoxControl、DBConnectionChooserControl、DBTableChooserControl、Enabled、ExtensionObjectPanel、FieldAllocationList、Layout、MultiFieldAllocationControl、MultiFieldChooserControl、MultiFieldSelectionTableControl、MultiItemChooserControl、PasswordBoxControl、PropertiesPanel、PropertiesSubPanel、PropertyControl、RadioButtonGroupControl、Required、SelectorPanel、ServerDirectoryChooserControl、ServerFileChooserControl、SingleFieldAllocationControl、SingleFieldChooserControl、SingleFieldValueChooserControl、SingleItemChooserControl、SpinnerControl、StaticText, SystemControls、TabbedPanel、TableControl、TextAreaControl、TextBoxControl、Visible

関連する要素

PropertiesSubPanel

PropertiesSubPanel 要素

関連する UI コンポーネントおよびコントロールをまとめてグループ化するために使用できるサブパネルを定義します。

表 *170. PropertiesSubPanel* の属性

属性	使用	説明	有効な値
buttonLabel	オプション		*string*
buttonLabelKey	オプション		*string*
dialogTitle	オプション		*string*
dialogTitleKey	オプション		*string*
helpLink	オプション		*string*
mnemonic	オプション		*string*
mnemonicKey	オプション		*string*

XML 表記

```
<xs:element name="PropertiesSubPanel">
  <xs:sequence>
    <xs:choice>
      <xs:element ref="Layout" minOccurs="0" maxOccurs="1"/>
      <xs:element ref="Enabled" minOccurs="0" maxOccurs="1"/>
      <xs:element ref="Visible" minOccurs="0" maxOccurs="1"/>
      <xs:element ref="Required" minOccurs="0" maxOccurs="1"/>
    </xs:choice>
```

```
  </xs:sequence>
  <xs:sequence maxOccurs="unbounded">
    <xs:choice>
      <xs:element ref="CheckBoxControl"/>
      <xs:element ref="TextBoxControl"/>
      <xs:element ref="PasswordBoxControl"/>
      <xs:element ref="TextAreaControl"/>
      <xs:element ref="RadioButtonGroupControl"/>
      <xs:element ref="CheckBoxGroupControl"/>
      <xs:element ref="ComboBoxControl"/>
      <xs:element ref="SpinnerControl"/>
      <xs:element ref="ServerFileChooserControl"/>
      <xs:element ref="ServerDirectoryChooserControl"/>
      <xs:element ref="ClientFileChooserControl"/>
      <xs:element ref="ClientDirectoryChooserControl"/>
      <xs:element ref="TableControl"/>
      <xs:element ref="SingleFieldChooserControl"/>
      <xs:element ref="SingleFieldAllocationControl"/>
      <xs:element ref="MultiFieldChooserControl"/>
      <xs:element ref="MultiFieldAllocationControl"/>
      <xs:element ref="MultiFieldSelectionTableControl"/>
      <xs:element ref="SingleFieldValueChooserControl"/>
      <xs:element ref="SingleItemChooserControl"/>
      <xs:element ref="MultiItemChooserControl"/>
      <xs:element ref="DBConnectionChooserControl"/>
      <xs:element ref="DBTableChooserControl"/>
      <xs:element ref="PropertyControl"/>
      <xs:element ref="StaticText"/>
      <xs:element ref="SystemControls"/>
      <xs:element ref="ActionButton"/>
      <xs:element ref="FieldAllocationList"/>
      <xs:element ref="PropertiesPanel"/>
      <xs:element ref="PropertiesSubPanel"/>
      <xs:element ref="SelectorPanel"/>
      <xs:element ref="ExtensionObjectPanel"/>
      <xs:element ref="TabbedPanel"/>
    </xs:choice>
  </xs:sequence>
  <xs:attribute name="buttonLabel" type="xs:string" use="optional"/>
  <xs:attribute name="buttonLabelKey" type="xs:string" use="optional"/>
  <xs:attribute name="mnemonic" type="xs:string" use="optional"/>
  <xs:attribute name="mnemonicKey" type="xs:string" use="optional"/>
  <xs:attribute name="dialogTitle" type="xs:string" use="optional"/>
  <xs:attribute name="dialogTitleKey" type="xs:string" use="optional"/>
  <xs:attribute name="helpLink" type="xs:string" use="optional"/>
</xs:element>
```

親要素

PropertiesPanel、PropertiesSubPanel

子要素

ActionButton、CheckBoxControl、CheckBoxGroupControl、ClientDirectoryChooserControl、ClientFileChooserControl、ComboBoxControl、DBConnectionChooserControl、DBTableChooserControl、Enabled、ExtensionObjectPanel、FieldAllocationList、Layout、MultiFieldAllocationControl、MultiFieldChooserControl、MultiFieldSelectionTableControl、MultiItemChooserControl、PasswordBoxControl、PropertiesPanel、PropertiesSubPanel、PropertyControl、RadioButtonGroupControl、Required、SelectorPanel、ServerDirectoryChooserControl、ServerFileChooserControl、SingleFieldAllocationControl、SingleFieldChooserControl、SingleFieldValueChooserControl、SingleItemChooserControl、SpinnerControl、StaticText, SystemControls、TabbedPanel、TableControl、TextAreaControl、TextBoxControl、Visible

関連する要素

PropertiesPanel

Property 要素

表 171. *Property* の属性

属性	使用	説明	有効な値
defaultValue	オプション		
defaultValueKey	オプション		*string*
deprecatedScriptNames	オプション		*string*
description	オプション		*string*
descriptionKey	オプション		*string*
isList	オプション		*boolean*
label	オプション		*string*
labelKey	オプション		*string*
max	オプション		*string*
min	オプション		*string*
name	必須		*string*
required	オプション		*boolean*
resourceKey	オプション		*string*
scriptName	オプション		*string*
type	オプション		*string*
valueType	オプション		**string** **encryptedString** **fieldName** **integer** **double** **boolean** **date** **enum** **structure** **databaseConnection**

XML 表記

```
<xs:element name="Property">
  <xs:choice>
    <xs:element ref="DefaultValue" minOccurs="0"/>
  </xs:choice>
  <xs:attribute name="valueType" type="PROPERTY-VALUE-TYPE">
    <xs:enumeration value="string"/>
    <xs:enumeration value="encryptedString"/>
    <xs:enumeration value="fieldName"/>
    <xs:enumeration value="integer"/>
    <xs:enumeration value="double"/>
    <xs:enumeration value="boolean"/>
    <xs:enumeration value="date"/>
    <xs:enumeration value="enum"/>
    <xs:enumeration value="structure"/>
    <xs:enumeration value="databaseConnection"/>
  </xs:attribute>
  <xs:attribute name="isList" type="xs:boolean" use="optional" default="false"/>
  <xs:attribute name="min" type="xs:string" use="optional"/>
  <xs:attribute name="max" type="xs:string" use="optional"/>
  <xs:attribute name="name" type="xs:string" use="required"/>
  <xs:attribute name="scriptName" type="xs:string" use="optional"/>
  <xs:attribute name="deprecatedScriptNames" type="xs:string" use="optional"/>
  <xs:attribute name="type" type="xs:string" use="optional"/>
  <xs:attribute name="defaultValue" type="EVALUATED-STRING" use="optional"/>
```

```
  <xs:attribute name="defaultValueKey" type="xs:string" use="optional"/>
  <xs:attribute name="required" type="xs:boolean" use="optional" default="false"/>
  <xs:attribute name="label" type="xs:string" use="optional"/>
  <xs:attribute name="labelKey" type="xs:string" use="optional"/>
  <xs:attribute name="description" type="xs:string" use="optional"/>
  <xs:attribute name="descriptionKey" type="xs:string" use="optional"/>
  <xs:attribute name="resourceKey" type="xs:string" use="optional"/>
</xs:element>
```

親要素

Properties、PropertySets

子要素

DefaultValue

関連する要素

PropertyType

PropertyControl 要素

カスタム プロパティー コントロールを定義します。controlClass 属性で識別されるクラスは、PropertyControl インターフェースを実装する必要があります。

表 *172. PropertyControl* の属性

属性	使用	説明	有効な値
controlClass	必須		*string*
description	オプション		*string*
descriptionKey	オプション		*string*
label	オプション		*string*
labelAbove	オプション		*boolean*
labelKey	オプション		*string*
labelWidth	オプション		*positiveInteger*
mnemonic	オプション		*string*
mnemonicKey	オプション		*string*
property	必須		*string*
resourceKey	オプション		*string*
showLabel	オプション		*boolean*

XML 表記

```
<xs:element name="PropertyControl">
  <xs:sequence>
    <xs:choice>
      <xs:element ref="Layout" minOccurs="0" maxOccurs="1"/>
      <xs:element ref="Enabled" minOccurs="0" maxOccurs="1"/>
      <xs:element ref="Visible" minOccurs="0" maxOccurs="1"/>
      <xs:element ref="Required" minOccurs="0" maxOccurs="1"/>
    </xs:choice>
  </xs:sequence>
  <xs:attribute name="property" type="xs:string" use="required"/>
  <xs:attribute name="showLabel" type="xs:boolean" use="optional" default="true"/>
  <xs:attribute name="resourceKey" type="xs:string" use="optional"/>
  <xs:attribute name="label" type="xs:string" use="optional"/>
  <xs:attribute name="labelKey" type="xs:string" use="optional"/>
  <xs:attribute name="mnemonic" type="xs:string" use="optional"/>
  <xs:attribute name="mnemonicKey" type="xs:string" use="optional"/>
  <xs:attribute name="labelWidth" type="xs:positiveInteger" use="optional" default="1"/>
```

```
  <xs:attribute name="labelAbove" type="xs:boolean" use="optional" default="false"/>
  <xs:attribute name="description" type="xs:string" use="optional"/>
  <xs:attribute name="descriptionKey" type="xs:string" use="optional"/>
  <xs:attribute name="controlClass" type="xs:string" use="required"/>
</xs:element>
```

親要素

PropertiesPanel、PropertiesSubPanel

子要素

Enabled、Layout、Required、Visible

関連する要素

CheckBoxControl、CheckBoxGroupControl、ClientDirectoryChooserControl、ClientFileChooserControl、DBConnectionChooserControl、DBTableChooserControl、MultiFieldAllocationControl、MultiFieldChooserControl、MultiFieldSelectionTableControl、PasswordBoxControl、RadioButtonGroupControl、ServerDirectoryChooserControl、ServerFileChooserControl、SingleFieldAllocationControl、SingleFieldChooserControl、SingleFieldValueChooserControl、SpinnerControl、TableControl、TextAreaControl、TextBoxControl

PropertyGroup 要素

表 *173. PropertyGroup* の属性

属性	使用	説明	有効な値
label	オプション		*string*
labelKey	オプション		*string*
properties	必須		*string*

XML 表記

```
<xs:element name="PropertyGroup">
  <xs:attribute name="label" type="xs:string" use="optional"/>
  <xs:attribute name="labelKey" type="xs:string" use="optional"/>
  <xs:attribute name="properties" type="xs:string" use="required"/>
</xs:element>
```

親要素

ExpertSettings、SimpleSettings

PropertySets 要素

XML 表記

```
<xs:element name="PropertySets">
  <xs:sequence>
    <xs:element ref="Property" minOccurs="0" maxOccurs="unbounded"/>
  </xs:sequence>
</xs:element>
```

親要素

CommonObjects

子要素

Property

PropertyType 要素

表 *174. PropertyType* の属性

属性	使用	説明	有効な値
ID	必須		*string*
isKeyed	オプション		*boolean*
isList	オプション		*boolean*
max	オプション		*string*
min	オプション		*string*
valueType	オプション		**string** **encryptedString** **fieldName** **integer** **double** **boolean** **date** **enum** **structure** **databaseConnection**

XML 表記

```
<xs:element name="PropertyType">
  <xs:choice>
    <xs:element ref="DefaultValue" minOccurs="0"/>
  </xs:choice>
  <xs:attribute name="valueType" type="PROPERTY-VALUE-TYPE">
    <xs:enumeration value="string"/>
    <xs:enumeration value="encryptedString"/>
    <xs:enumeration value="fieldName"/>
    <xs:enumeration value="integer"/>
    <xs:enumeration value="double"/>
    <xs:enumeration value="boolean"/>
    <xs:enumeration value="date"/>
    <xs:enumeration value="enum"/>
    <xs:enumeration value="structure"/>
    <xs:enumeration value="databaseConnection"/>
  </xs:attribute>
  <xs:attribute name="isList" type="xs:boolean" use="optional" default="false"/>
  <xs:attribute name="min" type="xs:string" use="optional"/>
  <xs:attribute name="max" type="xs:string" use="optional"/>
  <xs:choice>
    <xs:element ref="Enumeration" minOccurs="0"/>
    <xs:element ref="Structure" minOccurs="0"/>
  </xs:choice>
  <xs:attribute name="id" type="xs:string" use="required"/>
  <xs:attribute name="isKeyed" type="xs:boolean" use="optional" default="false"/>
</xs:element>
```

親要素

PropertyTypes

子要素

DefaultValue、Enumeration、Structure

関連する要素

Property

PropertyTypes 要素

XML 表記

```
<xs:element name="PropertyTypes">
  <xs:sequence>
    <xs:element ref="PropertyType" minOccurs="0" maxOccurs="unbounded"/>
  </xs:sequence>
</xs:element>
```

親要素

CommonObjects

子要素

PropertyType

RadioButtonGroupControl 要素

真/偽のブール値、または列挙型の値を指定するために使用できる、ラジオ ボタン コントロールのグループを定義します。

表 *175. RadioButtonGroupControl* の属性

属性	使用	説明	有効な値
description	オプション		*string*
descriptionKey	オプション		*string*
falseLabel	オプション		*string*
falseLabelKey	オプション		*string*
label	オプション		*string*
labelAbove	オプション		*boolean*
labelKey	オプション		*string*
labelWidth	オプション		*positiveInteger*
layoutByRow	オプション		*boolean*
mnemonic	オプション		*string*
mnemonicKey	オプション		*string*
property	必須		*string*
resourceKey	オプション		*string*
rows	オプション		*positiveInteger*
showLabel	オプション		*boolean*
trueFirst	オプション		*boolean*
trueLabel	オプション		*string*
trueLabelKey	オプション		*string*
useSubPanel	オプション		*boolean*

XML 表記

```
<xs:element name="RadioButtonGroupControl">
  <xs:sequence>
    <xs:choice>
      <xs:element ref="Layout" minOccurs="0" maxOccurs="1"/>
      <xs:element ref="Enabled" minOccurs="0" maxOccurs="1"/>
      <xs:element ref="Visible" minOccurs="0" maxOccurs="1"/>
      <xs:element ref="Required" minOccurs="0" maxOccurs="1"/>
    </xs:choice>
  </xs:sequence>
  <xs:attribute name="property" type="xs:string" use="required"/>
  <xs:attribute name="showLabel" type="xs:boolean" use="optional" default="true"/>
  <xs:attribute name="resourceKey" type="xs:string" use="optional"/>
  <xs:attribute name="label" type="xs:string" use="optional"/>
  <xs:attribute name="labelKey" type="xs:string" use="optional"/>
  <xs:attribute name="mnemonic" type="xs:string" use="optional"/>
  <xs:attribute name="mnemonicKey" type="xs:string" use="optional"/>
  <xs:attribute name="labelWidth" type="xs:positiveInteger" use="optional" default="1"/>
  <xs:attribute name="labelAbove" type="xs:boolean" use="optional" default="false"/>
  <xs:attribute name="description" type="xs:string" use="optional"/>
  <xs:attribute name="descriptionKey" type="xs:string" use="optional"/>
  <xs:attribute name="rows" type="xs:positiveInteger" use="optional" default="1"/>
  <xs:attribute name="layoutByRow" type="xs:boolean" use="optional" default="true"/>
  <xs:attribute name="useSubPanel" type="xs:boolean" use="optional" default="true"/>
  <xs:attribute name="falseLabel" type="xs:string" use="optional"/>
  <xs:attribute name="falseLabelKey" type="xs:string" use="optional"/>
  <xs:attribute name="trueLabel" type="xs:string" use="optional"/>
  <xs:attribute name="trueLabelKey" type="xs:string" use="optional"/>
  <xs:attribute name="trueFirst" type="xs:boolean" use="optional" default="false"/>
</xs:element>
```

親要素

PropertiesPanel、PropertiesSubPanel

子要素

Enabled、Layout、Required、Visible

関連する要素

CheckBoxControl、CheckBoxGroupControl、ClientDirectoryChooserControl、ClientFileChooserControl、DBConnectionChooserControl、DBTableChooserControl、MultiFieldAllocationControl、MultiFieldChooserControl、MultiFieldSelectionTableControl、PasswordBoxControl、PropertyControl、ServerDirectoryChooserControl、ServerFileChooserControl、SingleFieldAllocationControl、SingleFieldChooserControl、SingleFieldValueChooserControl、SpinnerControl、TableControl、TextAreaControl、TextBoxControl

Range 要素

表 *176. Range* の属性

属性	使用	説明	有効な値
max	オプション		*string*
min	オプション		*string*

XML 表記

```
<xs:element name="Range">
  <xs:attribute name="min" type="xs:string"/>
  <xs:attribute name="max" type="xs:string"/>
</xs:element>
```

親要素

AddField、ChangeField、Field、Field、MissingValues、MissingValues、MissingValues

Range 要素

表 *177. Range* の属性

属性	使用	説明	有効な値
maxValue	必須		*string*
minValue	必須		*string*

XML 表記

```
<xs:element name="Range" type="RANGE">
  <xs:attribute name="minValue" type="xs:string" use="required"/>
  <xs:attribute name="maxValue" type="xs:string" use="required"/>
</xs:element>
```

親要素

AddField、ChangeField、Field、Field、MissingValues、MissingValues、MissingValues

RemoveField 要素

表 *178. RemoveField* の属性

属性	使用	説明	有効な値
fieldRef	必須		

XML 表記

```
<xs:element name="RemoveField">
  <xs:attribute name="fieldRef" type="EVALUATED-STRING" use="required"/>
</xs:element>
```

親要素

ForEach、ModelFields

Required 要素

どのような場合に UI コンポーネントで有効な値の指定を必須とするかの条件を定義します。

表 *179. Required* の属性

属性	使用	説明	有効な値
messageKey	オプション		*string*
propertyName	オプション		*string*

XML 表記

```
<xs:element name="Required">
  <xs:sequence>
    <xs:group ref="CONDITION-EXPRESSION" minOccurs="0">
      <xs:choice>
        <xs:element ref="Condition"/>
        <xs:element ref="And"/>
        <xs:element ref="Or"/>
        <xs:element ref="Not"/>
      </xs:choice>
    </xs:group>
```

```
  </xs:sequence>
  <xs:attribute name="propertyName" type="xs:string" use="optional"/>
  <xs:attribute name="messageKey" type="xs:string" use="optional"/>
</xs:element>
```

親要素

ActionButton、CheckBoxControl、CheckBoxGroupControl、ClientDirectoryChooserControl、ClientFileChooserControl、ComboBoxControl、DBConnectionChooserControl、DBTableChooserControl、ExtensionObjectPanel、FieldAllocationList、ItemChooserControl、ModelViewerPanel、MultiFieldAllocationControl、MultiFieldChooserControl、MultiFieldSelectionTableControl、MultiItemChooserControl、OutputViewerPanel、PasswordBoxControl、PropertiesPanel、PropertiesSubPanel、PropertyControl、RadioButtonGroupControl、SelectorPanel、ServerDirectoryChooserControl、ServerFileChooserControl、SingleFieldAllocationControl、SingleFieldChooserControl、SingleFieldValueChooserControl、SingleItemChooserControl、SpinnerControl、StaticText、SystemControls、TabbedPanel, TableControl, TextAreaControl、TextBoxControl、TextBrowserPanel

子要素

And、Condition、Not、Or

Resources 要素

クライアント側のライブラリー、リソース・バンドル、およびサーバー側のライブラリーなどの共通リソースを定義します。

XML 表記

```
<xs:element name="Resources">
  <xs:sequence minOccurs="0" maxOccurs="unbounded">
    <xs:choice>
      <xs:element name="Bundle" minOccurs="0">
      </xs:element>
      <xs:element name="JarFile" minOccurs="0">
      </xs:element>
      <xs:element name="SharedLibrary" minOccurs="0">
      </xs:element>
      <xs:element name="HelpInfo" minOccurs="0">
      </xs:element>
    </xs:choice>
  </xs:sequence>
</xs:element>
```

親要素

Extension

子要素

Bundle、HelpInfo、JarFile、SharedLibrary

Bundle 要素:

表 *180. Bundle* の属性

属性	使用	説明	有効な値
ID	必須		*string*
path	必須		

表 *180. Bundle* の属性 (続き)

属性	使用	説明	有効な値
type	必須		**list** **properties**

XML 表記

```
<xs:element name="Bundle" minOccurs="0">
  <xs:attribute name="id" type="xs:string" use="required"/>
  <xs:attribute name="type" use="required">
    <xs:simpleType>
      <xs:restriction base="xs:string">
        <xs:enumeration value="list"/>
        <xs:enumeration value="properties"/>
      </xs:restriction>
    </xs:simpleType>
  </xs:attribute>
  <xs:attribute name="path" type="EVALUATED-STRING" use="required"/>
</xs:element>
```

親要素

リソース

JarFile 要素:

表 *181. JarFile* の属性

属性	使用	説明	有効な値
ID	必須		*string*
path	必須		

XML 表記

```
<xs:element name="JarFile" minOccurs="0">
  <xs:attribute name="id" type="xs:string" use="required"/>
  <xs:attribute name="path" type="EVALUATED-STRING" use="required"/>
</xs:element>
```

親要素

リソース

SharedLibrary 要素:

表 *182. SharedLibrary* の属性

属性	使用	説明	有効な値
ID	必須		*string*
path	必須		

XML 表記

```
<xs:element name="SharedLibrary" minOccurs="0">
  <xs:attribute name="id" type="xs:string" use="required"/>
  <xs:attribute name="path" type="EVALUATED-STRING" use="required"/>
</xs:element>
```

親要素

リソース

HelpInfo 要素**:**

表 *183. HelpInfo* の属性

属性	使用	説明	有効な値
default	オプション		*string*
helpset	オプション		
ID	オプション		*string*
missing	オプション		*string*
path	オプション		
type	必須		**native** **javahelp** **html**

XML 表記

```
<xs:element name="HelpInfo" minOccurs="0">
  <xs:attribute name="id" type="xs:string" use="optional"/>
  <xs:attribute name="type" use="required">
    <xs:simpleType>
      <xs:restriction base="xs:string">
        <xs:enumeration value="native"/>
        <xs:enumeration value="javahelp"/>
        <xs:enumeration value="html"/>
      </xs:restriction>
    </xs:simpleType>
  </xs:attribute>
  <xs:attribute name="path" type="EVALUATED-STRING" use="optional"/>
  <xs:attribute name="helpset" type="EVALUATED-STRING" use="optional"/>
  <xs:attribute name="default" type="xs:string" use="optional"/>
  <xs:attribute name="missing" type="xs:string" use="optional"/>
</xs:element>
```

親要素

リソース

Run 要素

XML 表記

```
<xs:element name="Run">
  <xs:sequence>
    <xs:group ref="CONDITION-EXPRESSION" minOccurs="0">
      <xs:choice>
        <xs:element ref="Condition"/>
        <xs:element ref="And"/>
        <xs:element ref="Or"/>
        <xs:element ref="Not"/>
      </xs:choice>
    </xs:group>
    <xs:element ref="Command" minOccurs="0" maxOccurs="unbounded"/>
    <xs:element ref="Option" minOccurs="0" maxOccurs="unbounded"/>
    <xs:element ref="StatusCodes" minOccurs="0"/>
  </xs:sequence>
</xs:element>
```

親要素

Executable

子要素

And、Command、Condition、Not、Option、Or、StatusCodes

SPSSDataFormat 要素

XML 表記

```
<xs:element name="SPSSDataFormat"/>
```

親要素

DataFormat

SelectorPanel 要素

複数のサブパネルを格納できるが、このうち、一度に表示できるサブパネルは 1 つだけである UI コンポーネントを定義します。

表 *184. SelectorPanel* の属性

属性	使用	説明	有効な値
control	必須		*string*

XML 表記

```
<xs:element name="SelectorPanel">
  <xs:sequence>
    <xs:choice>
      <xs:element ref="Layout" minOccurs="0" maxOccurs="1"/>
      <xs:element ref="Enabled" minOccurs="0" maxOccurs="1"/>
      <xs:element ref="Visible" minOccurs="0" maxOccurs="1"/>
      <xs:element ref="Required" minOccurs="0" maxOccurs="1"/>
    </xs:choice>
  </xs:sequence>
  <xs:sequence minOccurs="0" maxOccurs="unbounded">
    <xs:element name="Selector">
    </xs:element>
  </xs:sequence>
  <xs:attribute name="control" type="xs:string" use="required"/>
</xs:element>
```

親要素

PropertiesPanel、PropertiesSubPanel

子要素

Enabled、Layout、Required、Selector、Visible

関連する要素

ActionButton、ComboBoxControl、ExtensionObjectPanel、FieldAllocationList、ModelViewerPanel、OutputViewerPanel、StaticText、SystemControls、TabbedPanel、TextBrowserPanel

Selector 要素:

表 *185. Selector* の属性

属性	使用	説明	有効な値
controlValue	必須		*string*
panelId	必須		*string*

XML 表記

```
<xs:element name="Selector">
  <xs:attribute name="panelId" type="xs:string" use="required"/>
  <xs:attribute name="controlValue" type="xs:string" use="required"/>
</xs:element>
```

親要素

SelectorPanel

ServerDirectoryChooserControl 要素

サーバー上のディレクトリーを選択するために使用できるコントロールを定義します。

表 *186. ServerDirectoryChooserControl* の属性

属性	使用	説明	有効な値
description	オプション		*string*
descriptionKey	オプション		*string*
label	オプション		*string*
labelAbove	オプション		*boolean*
labelKey	オプション		*string*
labelWidth	オプション		*positiveInteger*
mnemonic	オプション		*string*
mnemonicKey	オプション		*string*
最頻値	必須		**open** **save** **import** **export**
property	必須		*string*
resourceKey	オプション		*string*
showLabel	オプション		*boolean*

XML 表記

```
<xs:element name="ServerDirectoryChooserControl">
  <xs:sequence>
    <xs:choice>
      <xs:element ref="Layout" minOccurs="0" maxOccurs="1"/>
      <xs:element ref="Enabled" minOccurs="0" maxOccurs="1"/>
      <xs:element ref="Visible" minOccurs="0" maxOccurs="1"/>
      <xs:element ref="Required" minOccurs="0" maxOccurs="1"/>
    </xs:choice>
  </xs:sequence>
  <xs:attribute name="property" type="xs:string" use="required"/>
  <xs:attribute name="showLabel" type="xs:boolean" use="optional" default="true"/>
  <xs:attribute name="resourceKey" type="xs:string" use="optional"/>
  <xs:attribute name="label" type="xs:string" use="optional"/>
  <xs:attribute name="labelKey" type="xs:string" use="optional"/>
  <xs:attribute name="mnemonic" type="xs:string" use="optional"/>
  <xs:attribute name="mnemonicKey" type="xs:string" use="optional"/>
  <xs:attribute name="labelWidth" type="xs:positiveInteger" use="optional" default="1"/>
  <xs:attribute name="labelAbove" type="xs:boolean" use="optional" default="false"/>
  <xs:attribute name="description" type="xs:string" use="optional"/>
  <xs:attribute name="descriptionKey" type="xs:string" use="optional"/>
  <xs:attribute name="mode" type="FILE-CHOOSER-MODE" use="required">
    <xs:enumeration value="open"/>
    <xs:enumeration value="save"/>
```

```
    <xs:enumeration value="import"/>
    <xs:enumeration value="export"/>
  </xs:attribute>
</xs:element>
```

親要素

PropertiesPanel、PropertiesSubPanel

子要素

Enabled、Layout、Required、Visible

関連する要素

CheckBoxControl、CheckBoxGroupControl、ClientDirectoryChooserControl、ClientFileChooserControl、DBConnectionChooserControl、DBTableChooserControl、MultiFieldAllocationControl、MultiFieldChooserControl、MultiFieldSelectionTableControl、PasswordBoxControl、PropertyControl、RadioButtonGroupControl、ServerFileChooserControl、SingleFieldAllocationControl、SingleFieldChooserControl、SingleFieldValueChooserControl、SpinnerControl、TableControl、TextAreaControl、TextBoxControl

ServerFileChooserControl 要素

サーバー上のファイルを選択するために使用できるコントロールを定義します。

表 *187. ServerFileChooserControl* の属性

属性	使用	説明	有効な値
description	オプション		*string*
descriptionKey	オプション		*string*
label	オプション		*string*
labelAbove	オプション		*boolean*
labelKey	オプション		*string*
labelWidth	オプション		*positiveInteger*
mnemonic	オプション		*string*
mnemonicKey	オプション		*string*
最頻値	必須		**open** **save** **import** **export**
property	必須		*string*
resourceKey	オプション		*string*
showLabel	オプション		*boolean*

XML 表記

```
<xs:element name="ServerFileChooserControl">
  <xs:sequence>
    <xs:choice>
      <xs:element ref="Layout" minOccurs="0" maxOccurs="1"/>
      <xs:element ref="Enabled" minOccurs="0" maxOccurs="1"/>
      <xs:element ref="Visible" minOccurs="0" maxOccurs="1"/>
      <xs:element ref="Required" minOccurs="0" maxOccurs="1"/>
    </xs:choice>
```

```
  </xs:sequence>
  <xs:attribute name="property" type="xs:string" use="required"/>
  <xs:attribute name="showLabel" type="xs:boolean" use="optional" default="true"/>
  <xs:attribute name="resourceKey" type="xs:string" use="optional"/>
  <xs:attribute name="label" type="xs:string" use="optional"/>
  <xs:attribute name="labelKey" type="xs:string" use="optional"/>
  <xs:attribute name="mnemonic" type="xs:string" use="optional"/>
  <xs:attribute name="mnemonicKey" type="xs:string" use="optional"/>
  <xs:attribute name="labelWidth" type="xs:positiveInteger" use="optional" default="1"/>
  <xs:attribute name="labelAbove" type="xs:boolean" use="optional" default="false"/>
  <xs:attribute name="description" type="xs:string" use="optional"/>
  <xs:attribute name="descriptionKey" type="xs:string" use="optional"/>
  <xs:attribute name="mode" type="FILE-CHOOSER-MODE" use="required">
    <xs:enumeration value="open"/>
    <xs:enumeration value="save"/>
    <xs:enumeration value="import"/>
    <xs:enumeration value="export"/>
  </xs:attribute>
</xs:element>
```

親要素

PropertiesPanel、PropertiesSubPanel

子要素

Enabled、Layout、Required、Visible

関連する要素

CheckBoxControl、CheckBoxGroupControl、ClientDirectoryChooserControl、ClientFileChooserControl、DBConnectionChooserControl、DBTableChooserControl、MultiFieldAllocationControl、MultiFieldChooserControl、MultiFieldSelectionTableControl、PasswordBoxControl、PropertyControl、RadioButtonGroupControl、ServerDirectoryChooserControl、SingleFieldAllocationControl、SingleFieldChooserControl、SingleFieldValueChooserControl、SpinnerControl、TableControl、TextAreaControl、TextBoxControl

SetContainer 要素

表 *188. SetContainer* の属性

属性	使用	説明	有効な値
source	必須		*string*
target	必須		*string*

XML 表記

```
<xs:element name="SetContainer">
  <xs:attribute name="source" type="xs:string" use="required"/>
  <xs:attribute name="target" type="xs:string" use="required"/>
</xs:element>
```

親要素

CreateDocumentOutput、CreateInteractiveDocumentBuilder、CreateInteractiveModelBuilder、CreateModelApplier、CreateModelOutput

関連する要素

SetProperty

SetProperty 要素

表 189. *SetProperty* の属性

属性	使用	説明	有効な値
source	必須		*string*
target	必須		*string*

XML 表記

```
<xs:element name="SetProperty">
  <xs:attribute name="source" type="xs:string" use="required"/>
  <xs:attribute name="target" type="xs:string" use="required"/>
</xs:element>
```

親要素

CreateDocumentOutput、CreateInteractiveDocumentBuilder、CreateInteractiveModelBuilder、CreateModelApplier、CreateModelOutput

関連する要素

SetContainer

SingleFieldAllocationControl 要素

allocator 属性によって識別されるフィールド割り振りリスト コントロールから 1 つのフィールドを選択するために使用できるコントロールを定義します。

表 190. *SingleFieldAllocationControl* の属性

属性	使用	説明	有効な値
allocator	必須		*string*
buttonColumn	必須		*integer*
description	オプション		*string*
descriptionKey	オプション		*string*
label	オプション		*string*
labelAbove	オプション		*boolean*
labelKey	オプション		*string*
labelWidth	オプション		*positiveInteger*
mnemonic	オプション		*string*
mnemonicKey	オプション		*string*
onlyDatetime	オプション		*boolean*
onlyDiscrete	オプション		*boolean*
onlyNumeric	オプション		*boolean*
onlyRanges	オプション		*boolean*
onlySymbolic	オプション		*boolean*
property	必須		*string*
resourceKey	オプション		*string*
showLabel	オプション		*boolean*
storage	オプション		*string*

表 190. *SingleFieldAllocationControl* の属性 (続き)

属性	使用	説明	有効な値
types	オプション		*string*

XML 表記

```
<xs:element name="SingleFieldAllocationControl">
  <xs:sequence>
    <xs:choice>
      <xs:element ref="Layout" minOccurs="0" maxOccurs="1"/>
      <xs:element ref="Enabled" minOccurs="0" maxOccurs="1"/>
      <xs:element ref="Visible" minOccurs="0" maxOccurs="1"/>
      <xs:element ref="Required" minOccurs="0" maxOccurs="1"/>
    </xs:choice>
  </xs:sequence>
  <xs:attribute name="property" type="xs:string" use="required"/>
  <xs:attribute name="showLabel" type="xs:boolean" use="optional" default="true"/>
  <xs:attribute name="resourceKey" type="xs:string" use="optional"/>
  <xs:attribute name="label" type="xs:string" use="optional"/>
  <xs:attribute name="labelKey" type="xs:string" use="optional"/>
  <xs:attribute name="mnemonic" type="xs:string" use="optional"/>
  <xs:attribute name="mnemonicKey" type="xs:string" use="optional"/>
  <xs:attribute name="labelWidth" type="xs:positiveInteger" use="optional" default="1"/>
  <xs:attribute name="labelAbove" type="xs:boolean" use="optional" default="false"/>
  <xs:attribute name="description" type="xs:string" use="optional"/>
  <xs:attribute name="descriptionKey" type="xs:string" use="optional"/>
  <xs:attribute name="allocator" type="xs:string" use="required"/>
  <xs:attribute name="buttonColumn" type="xs:integer" use="required"/>
  <xs:attribute name="storage" type="xs:string" use="optional"/>
  <xs:attribute name="onlyNumeric" type="xs:boolean" use="optional"/>
  <xs:attribute name="onlySymbolic" type="xs:boolean" use="optional"/>
  <xs:attribute name="onlyDatetime" type="xs:boolean" use="optional"/>
  <xs:attribute name="types" type="xs:string" use="optional"/>
  <xs:attribute name="onlyRanges" type="xs:boolean" use="optional"/>
  <xs:attribute name="onlyDiscrete" type="xs:boolean" use="optional"/>
</xs:element>
```

親要素

PropertiesPanel、PropertiesSubPanel

子要素

Enabled、Layout、Required、Visible

関連する要素

CheckBoxControl、CheckBoxGroupControl、ClientDirectoryChooserControl、ClientFileChooserControl、DBConnectionChooserControl、DBTableChooserControl、MultiFieldAllocationControl、MultiFieldChooserControl、MultiFieldSelectionTableControl、PasswordBoxControl、PropertyControl、RadioButtonGroupControl、ServerDirectoryChooserControl、ServerFileChooserControl、SingleFieldChooserControl、SingleFieldValueChooserControl、SpinnerControl、TableControl、TextAreaControl、TextBoxControl

SingleFieldChooserControl 要素

現在のデータ モデルの 1 つのフィールドを選択するために使用できるコントロールを定義します。

表 191. *SingleFieldChooserControl* の属性

属性	使用	説明	有効な値
description	オプション		*string*
descriptionKey	オプション		*string*
label	オプション		*string*

表 191. *SingleFieldChooserControl* の属性 (続き)

属性	使用	説明	有効な値
labelAbove	オプション		*boolean*
labelKey	オプション		*string*
labelWidth	オプション		*positiveInteger*
mnemonic	オプション		*string*
mnemonicKey	オプション		*string*
onlyDatetime	オプション		*boolean*
onlyDiscrete	オプション		*boolean*
onlyNumeric	オプション		*boolean*
onlyRanges	オプション		*boolean*
onlySymbolic	オプション		*boolean*
property	必須		*string*
resourceKey	オプション		*string*
showLabel	オプション		*boolean*
storage	オプション		*string*
types	オプション		*string*

XML 表記

```
<xs:element name="SingleFieldChooserControl">
  <xs:sequence>
    <xs:choice>
      <xs:element ref="Layout" minOccurs="0" maxOccurs="1"/>
      <xs:element ref="Enabled" minOccurs="0" maxOccurs="1"/>
      <xs:element ref="Visible" minOccurs="0" maxOccurs="1"/>
      <xs:element ref="Required" minOccurs="0" maxOccurs="1"/>
    </xs:choice>
  </xs:sequence>
  <xs:attribute name="property" type="xs:string" use="required"/>
  <xs:attribute name="showLabel" type="xs:boolean" use="optional" default="true"/>
  <xs:attribute name="resourceKey" type="xs:string" use="optional"/>
  <xs:attribute name="label" type="xs:string" use="optional"/>
  <xs:attribute name="labelKey" type="xs:string" use="optional"/>
  <xs:attribute name="mnemonic" type="xs:string" use="optional"/>
  <xs:attribute name="mnemonicKey" type="xs:string" use="optional"/>
  <xs:attribute name="labelWidth" type="xs:positiveInteger" use="optional" default="1"/>
  <xs:attribute name="labelAbove" type="xs:boolean" use="optional" default="false"/>
  <xs:attribute name="description" type="xs:string" use="optional"/>
  <xs:attribute name="descriptionKey" type="xs:string" use="optional"/>
  <xs:attribute name="storage" type="xs:string" use="optional"/>
  <xs:attribute name="onlyNumeric" type="xs:boolean" use="optional"/>
  <xs:attribute name="onlySymbolic" type="xs:boolean" use="optional"/>
  <xs:attribute name="onlyDatetime" type="xs:boolean" use="optional"/>
  <xs:attribute name="types" type="xs:string" use="optional"/>
  <xs:attribute name="onlyRanges" type="xs:boolean" use="optional"/>
  <xs:attribute name="onlyDiscrete" type="xs:boolean" use="optional"/>
</xs:element>
```

親要素

PropertiesPanel、PropertiesSubPanel

子要素

Enabled、Layout、Required、Visible

関連する要素

CheckBoxControl、CheckBoxGroupControl、ClientDirectoryChooserControl、ClientFileChooserControl、DBConnectionChooserControl、DBTableChooserControl、MultiFieldAllocationControl、MultiFieldChooserControl、MultiFieldSelectionTableControl、PasswordBoxControl、PropertyControl、RadioButtonGroupControl、ServerDirectoryChooserControl、ServerFileChooserControl、SingleFieldAllocationControl、SingleFieldValueChooserControl、SpinnerControl、TableControl、TextAreaControl、TextBoxControl

SingleFieldValueChooserControl 要素

fieldControl で識別されるコントロールによって選択されたフィールドから 1 つのフィールド値を選択するために使用できるコントロールを定義します。

表 *192. SingleFieldValueChooserControl* の属性

属性	使用	説明	有効な値
description	オプション		*string*
descriptionKey	オプション		*string*
fieldControl	オプション		*string*
fieldDirection	オプション		**in** **out** **both** **none** **partition**
label	オプション		*string*
labelAbove	オプション		*boolean*
labelKey	オプション		*string*
labelWidth	オプション		*positiveInteger*
mnemonic	オプション		*string*
mnemonicKey	オプション		*string*
property	必須		*string*
resourceKey	オプション		*string*
showLabel	オプション		*boolean*

XML 表記

```
<xs:element name="SingleFieldValueChooserControl">
  <xs:sequence>
    <xs:choice>
      <xs:element ref="Layout" minOccurs="0" maxOccurs="1"/>
      <xs:element ref="Enabled" minOccurs="0" maxOccurs="1"/>
      <xs:element ref="Visible" minOccurs="0" maxOccurs="1"/>
      <xs:element ref="Required" minOccurs="0" maxOccurs="1"/>
    </xs:choice>
  </xs:sequence>
  <xs:attribute name="property" type="xs:string" use="required"/>
  <xs:attribute name="showLabel" type="xs:boolean" use="optional" default="true"/>
  <xs:attribute name="resourceKey" type="xs:string" use="optional"/>
  <xs:attribute name="label" type="xs:string" use="optional"/>
  <xs:attribute name="labelKey" type="xs:string" use="optional"/>
  <xs:attribute name="mnemonic" type="xs:string" use="optional"/>
  <xs:attribute name="mnemonicKey" type="xs:string" use="optional"/>
  <xs:attribute name="labelWidth" type="xs:positiveInteger" use="optional" default="1"/>
  <xs:attribute name="labelAbove" type="xs:boolean" use="optional" default="false"/>
  <xs:attribute name="description" type="xs:string" use="optional"/>
  <xs:attribute name="descriptionKey" type="xs:string" use="optional"/>
```

```
  <xs:attribute name="fieldControl" type="xs:string" use="optional"/>
  <xs:attribute name="fieldDirection" type="FIELD-DIRECTION" use="optional">
    <xs:enumeration value="in"/>
    <xs:enumeration value="out"/>
    <xs:enumeration value="both"/>
    <xs:enumeration value="none"/>
    <xs:enumeration value="partition"/>
  </xs:attribute>
</xs:element>
```

親要素

PropertiesPanel、PropertiesSubPanel

子要素

Enabled、Layout、Required、Visible

関連する要素

CheckBoxControl、CheckBoxGroupControl、ClientDirectoryChooserControl、ClientFileChooserControl、DBConnectionChooserControl、DBTableChooserControl、MultiFieldAllocationControl、MultiFieldChooserControl、MultiFieldSelectionTableControl、PasswordBoxControl、PropertyControl、RadioButtonGroupControl、ServerDirectoryChooserControl、ServerFileChooserControl、SingleFieldAllocationControl、SingleFieldChooserControl、SpinnerControl、TableControl、TextAreaControl、TextBoxControl

SingleItemChooserControl 要素

選択項目から 1 つの値を選択するために使用できるコントロールを定義します。

表 *193. SingleItemChooserControl* の属性

属性	使用	説明	有効な値
catalog	必須		*string*
description	オプション		*string*
descriptionKey	オプション		*string*
label	オプション		*string*
labelAbove	オプション		*boolean*
labelKey	オプション		*string*
labelWidth	オプション		*positiveInteger*
mnemonic	オプション		*string*
mnemonicKey	オプション		*string*
property	必須		*string*
resourceKey	オプション		*string*
showLabel	オプション		*boolean*

XML 表記

```
<xs:element name="SingleItemChooserControl">
  <xs:sequence>
    <xs:choice>
      <xs:element ref="Layout" minOccurs="0" maxOccurs="1"/>
      <xs:element ref="Enabled" minOccurs="0" maxOccurs="1"/>
      <xs:element ref="Visible" minOccurs="0" maxOccurs="1"/>
      <xs:element ref="Required" minOccurs="0" maxOccurs="1"/>
    </xs:choice>
  </xs:sequence>
```

```
  <xs:attribute name="property" type="xs:string" use="required"/>
  <xs:attribute name="showLabel" type="xs:boolean" use="optional" default="true"/>
  <xs:attribute name="resourceKey" type="xs:string" use="optional"/>
  <xs:attribute name="label" type="xs:string" use="optional"/>
  <xs:attribute name="labelKey" type="xs:string" use="optional"/>
  <xs:attribute name="mnemonic" type="xs:string" use="optional"/>
  <xs:attribute name="mnemonicKey" type="xs:string" use="optional"/>
  <xs:attribute name="labelWidth" type="xs:positiveInteger" use="optional" default="1"/>
  <xs:attribute name="labelAbove" type="xs:boolean" use="optional" default="false"/>
  <xs:attribute name="description" type="xs:string" use="optional"/>
  <xs:attribute name="descriptionKey" type="xs:string" use="optional"/>
  <xs:attribute name="catalog" type="xs:string" use="required"/>
</xs:element>
```

親要素

PropertiesPanel、PropertiesSubPanel

子要素

Enabled、Layout、Required、Visible

関連する要素

MultiItemChooserControl

SpinnerControl 要素

数値を指定するために使用できるスピン コントロールを定義します。

表 *194. SpinnerControl* の属性

属性	使用	説明	有効な値
columns	オプション		*positiveInteger*
description	オプション		*string*
descriptionKey	オプション		*string*
label	オプション		*string*
labelAbove	オプション		*boolean*
labelKey	オプション		*string*
labelWidth	オプション		*positiveInteger*
maxDecimalDigits	オプション		*positiveInteger*
minDecimalDigits	オプション		*positiveInteger*
mnemonic	オプション		*string*
mnemonicKey	オプション		*string*
property	必須		*string*
resourceKey	オプション		*string*
showLabel	オプション		*boolean*
stepSize	オプション		*10* 進数

XML 表記

```
<xs:element name="SpinnerControl">
  <xs:sequence>
    <xs:choice>
      <xs:element ref="Layout" minOccurs="0" maxOccurs="1"/>
      <xs:element ref="Enabled" minOccurs="0" maxOccurs="1"/>
      <xs:element ref="Visible" minOccurs="0" maxOccurs="1"/>
      <xs:element ref="Required" minOccurs="0" maxOccurs="1"/>
    </xs:choice>
```

```
  </xs:sequence>
  <xs:attribute name="property" type="xs:string" use="required"/>
  <xs:attribute name="showLabel" type="xs:boolean" use="optional" default="true"/>
  <xs:attribute name="resourceKey" type="xs:string" use="optional"/>
  <xs:attribute name="label" type="xs:string" use="optional"/>
  <xs:attribute name="labelKey" type="xs:string" use="optional"/>
  <xs:attribute name="mnemonic" type="xs:string" use="optional"/>
  <xs:attribute name="mnemonicKey" type="xs:string" use="optional"/>
  <xs:attribute name="labelWidth" type="xs:positiveInteger" use="optional" default="1"/>
  <xs:attribute name="labelAbove" type="xs:boolean" use="optional" default="false"/>
  <xs:attribute name="description" type="xs:string" use="optional"/>
  <xs:attribute name="descriptionKey" type="xs:string" use="optional"/>
  <xs:attribute name="columns" type="xs:positiveInteger" use="optional" default="5"/>
  <xs:attribute name="stepSize" type="xs:decimal" use="optional" default="1.0"/>
  <xs:attribute name="minDecimalDigits" type="xs:positiveInteger" use="optional" default="1"/>
  <xs:attribute name="maxDecimalDigits" type="xs:positiveInteger" use="optional"/>
</xs:element>
```

親要素

PropertiesPanel、PropertiesSubPanel

子要素

Enabled、Layout、Required、Visible

関連する要素

CheckBoxControl、CheckBoxGroupControl、ClientDirectoryChooserControl、ClientFileChooserControl、DBConnectionChooserControl、DBTableChooserControl、MultiFieldAllocationControl、MultiFieldChooserControl、MultiFieldSelectionTableControl、PasswordBoxControl、PropertyControl、RadioButtonGroupControl、ServerDirectoryChooserControl、ServerFileChooserControl、SingleFieldAllocationControl、SingleFieldChooserControl、SingleFieldValueChooserControl、TableControl、TextAreaControl、TextBoxControl

StaticText 要素

静的テキストを含む UI コンポーネントを定義します。これは、パネルの目的を記述するために、サブパネルでよく使用されます。

表 *195. StaticText* の属性

属性	使用	説明	有効な値
text	オプション		*string*
textKey	オプション		*string*

XML 表記

```
<xs:element name="StaticText">
  <xs:sequence>
    <xs:choice>
      <xs:element ref="Layout" minOccurs="0" maxOccurs="1"/>
      <xs:element ref="Enabled" minOccurs="0" maxOccurs="1"/>
      <xs:element ref="Visible" minOccurs="0" maxOccurs="1"/>
      <xs:element ref="Required" minOccurs="0" maxOccurs="1"/>
    </xs:choice>
  </xs:sequence>
  <xs:attribute name="text" type="xs:string" use="optional"/>
  <xs:attribute name="textKey" type="xs:string" use="optional"/>
</xs:element>
```

親要素

PropertiesPanel、PropertiesSubPanel

子要素

Enabled、Layout、Required、Visible

関連する要素

ActionButton、ComboBoxControl、ExtensionObjectPanel、FieldAllocationList、ModelViewerPanel、OutputViewerPanel、SelectorPanel、SystemControls、TabbedPanel、TextBrowserPanel

StatusCode 要素

表 *196. StatusCode* の属性

属性	使用	説明	有効な値
code	必須		*integer*
message	オプション		*string*
messageKey	オプション		*string*
status	オプション		**success** **warning** **error**

XML 表記

```
<xs:element name="StatusCode">
  <xs:attribute name="code" type="xs:integer" use="required"/>
  <xs:attribute name="status" use="optional" default="success">
    <xs:simpleType>
      <xs:restriction base="xs:string">
        <xs:enumeration value="success"/>
        <xs:enumeration value="warning"/>
        <xs:enumeration value="error"/>
      </xs:restriction>
    </xs:simpleType>
  </xs:attribute>
  <xs:attribute name="message" type="xs:string" use="optional"/>
  <xs:attribute name="messageKey" type="xs:string" use="optional"/>
</xs:element>
```

親要素

StatusCodes

StatusCodes 要素

表 *197. StatusCodes* の属性

属性	使用	説明	有効な値
defaultMessage	オプション		*string*
defaultMessageKey	オプション		*string*

XML 表記

```
<xs:element name="StatusCodes">
  <xs:sequence>
    <xs:element ref="StatusCode" maxOccurs="unbounded"/>
  </xs:sequence>
  <xs:attribute name="defaultMessage" type="xs:string" use="optional"/>
  <xs:attribute name="defaultMessageKey" type="xs:string" use="optional"/>
</xs:element>
```

親要素

Module、Run

子要素

StatusCode

StatusDetail 要素

進行状況またはその他の状況に関する補足情報。

表 198. *StatusDetail* の属性

属性	使用	説明	有効な値
destination	オプション		**client** **tracefile** **console**

XML 表記

```
<xs:element name="StatusDetail" type="STATUS-DETAIL">
  <xs:sequence>
    <xs:element name="Diagnostic" type="DIAGNOSTIC" minOccurs="0" maxOccurs="unbounded">
      <xs:sequence>
        <xs:element name="Message" type="DIAGNOSTIC-MESSAGE" minOccurs="0">
        </xs:element>
        <xs:element name="Parameter" type="xs:string" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:element>
  </xs:sequence>
  <xs:attribute name="destination" type="STATUS-DESTINATION" default="client">
    <xs:enumeration value="client"/>
    <xs:enumeration value="tracefile"/>
    <xs:enumeration value="console"/>
  </xs:attribute>
</xs:element>
```

子要素

Diagnostic

Diagnostic 要素:

表 199. *Diagnostic* の属性

属性	使用	説明	有効な値
code	必須		*integer*
severity	オプション		**unknown** **information** **warning** **error** **fatal**
source	オプション		*string*
subCode	オプション		*integer*

XML 表記

```
<xs:element name="Diagnostic" type="DIAGNOSTIC" minOccurs="0" maxOccurs="unbounded">
  <xs:sequence>
    <xs:element name="Message" type="DIAGNOSTIC-MESSAGE" minOccurs="0">
    </xs:element>
    <xs:element name="Parameter" type="xs:string" minOccurs="0" maxOccurs="unbounded"/>
  </xs:sequence>
  <xs:attribute name="code" type="xs:integer" use="required"/>
  <xs:attribute name="subCode" type="xs:integer" default="0"/>
  <xs:attribute name="severity" type="DIAGNOSTIC-SEVERITY" default="error">
    <xs:enumeration value="unknown"/>
    <xs:enumeration value="information"/>
    <xs:enumeration value="warning"/>
    <xs:enumeration value="error"/>
    <xs:enumeration value="fatal"/>
  </xs:attribute>
  <xs:attribute name="source" type="xs:string"/>
</xs:element>
```

親要素

StatusDetail

子要素

Message、Parameter

***Message* 要素:**

表 *200. Message* の属性

属性	使用	説明	有効な値
lang	オプション		*NMTOKEN*

XML 表記

```
<xs:element name="Message" type="DIAGNOSTIC-MESSAGE" minOccurs="0">
  <xs:attribute name="lang" type="xs:NMTOKEN"/>
</xs:element>
```

親要素

Diagnostic

***Parameter* 要素:**

XML 表記

```
<xs:element name="Parameter" type="xs:string" minOccurs="0" maxOccurs="unbounded"/>
```

親要素

Diagnostic

Structure 要素

XML 表記

```
<xs:element name="Structure">
  <xs:sequence>
    <xs:element ref="Attribute" minOccurs="0" maxOccurs="unbounded"/>
  </xs:sequence>
</xs:element>
```

親要素

PropertyType

子要素

属性

StructuredValue 要素

名前付きの値のシーケンス (「属性」)。

XML 表記

```
<xs:element name="StructuredValue" type="STRUCTURED-VALUE">
  <xs:sequence>
    <xs:element name="Attribute" type="ATTRIBUTE" maxOccurs="unbounded">
      <xs:group ref="PARAMETER-CONTENT" minOccurs="0">
        <xs:choice>
          <xs:element ref="MapValue"/>
          <xs:element ref="StructuredValue"/>
          <xs:element ref="ListValue"/>
          <xs:element ref="Value"/>
          <xs:element ref="DatabaseConnectionValue"/>
        </xs:choice>
      </xs:group>
      <xs:sequence>
        <xs:element name="ListValue" type="LIST-VALUE" minOccurs="0" maxOccurs="1">
          <xs:group ref="PARAMETER-CONTENT" minOccurs="0" maxOccurs="unbounded">
            <xs:choice>
              <xs:element ref="MapValue"/>
              <xs:element ref="StructuredValue"/>
              <xs:element ref="ListValue"/>
              <xs:element ref="Value"/>
              <xs:element ref="DatabaseConnectionValue"/>
            </xs:choice>
          </xs:group>
        </xs:element>
      </xs:sequence>
    </xs:element>
  </xs:sequence>
</xs:element>
```

親要素

Attribute、Attribute、ListValue、ListValue、ListValue、Parameter

子要素

属性

Attribute 要素:

表 *201. Attribute* の属性

属性	使用	説明	有効な値
name	必須		*string*
value	オプション		*string*

XML 表記

```
<xs:element name="Attribute" type="ATTRIBUTE" maxOccurs="unbounded">
  <xs:group ref="PARAMETER-CONTENT" minOccurs="0">
    <xs:choice>
      <xs:element ref="MapValue"/>
      <xs:element ref="StructuredValue"/>
      <xs:element ref="ListValue"/>
      <xs:element ref="Value"/>
      <xs:element ref="DatabaseConnectionValue"/>
```

```
    </xs:choice>
  </xs:group>
  <xs:sequence>
    <xs:element name="ListValue" type="LIST-VALUE" minOccurs="0" maxOccurs="1">
      <xs:group ref="PARAMETER-CONTENT" minOccurs="0" maxOccurs="unbounded">
        <xs:choice>
          <xs:element ref="MapValue"/>
          <xs:element ref="StructuredValue"/>
          <xs:element ref="ListValue"/>
          <xs:element ref="Value"/>
          <xs:element ref="DatabaseConnectionValue"/>
        </xs:choice>
      </xs:group>
    </xs:element>
  </xs:sequence>
  <xs:attribute name="name" type="xs:string" use="required"/>
  <xs:attribute name="value" type="xs:string"/>
</xs:element>
```

親要素

StructuredValue

子要素

DatabaseConnectionValue、ListValue、ListValue、MapValue、StructuredValue、Value

ListValue 要素*:* 値のシーケンス。すべての値は同じコンテンツ・タイプを持つ必要がありますが、これはチェックされません。

XML 表記

```
<xs:element name="ListValue" type="LIST-VALUE" minOccurs="0" maxOccurs="1">
  <xs:group ref="PARAMETER-CONTENT" minOccurs="0" maxOccurs="unbounded">
    <xs:choice>
      <xs:element ref="MapValue"/>
      <xs:element ref="StructuredValue"/>
      <xs:element ref="ListValue"/>
      <xs:element ref="Value"/>
      <xs:element ref="DatabaseConnectionValue"/>
    </xs:choice>
  </xs:group>
</xs:element>
```

親要素

属性

子要素

DatabaseConnectionValue、ListValue、MapValue、StructuredValue、Value

SystemControls 要素

表 *202. SystemControls* の属性

属性	使用	説明	有効な値
controlsId	必須		*string*
style	オプション		*string*

XML 表記

```
<xs:element name="SystemControls">
  <xs:sequence>
    <xs:choice>
      <xs:element ref="Layout" minOccurs="0" maxOccurs="1"/>
      <xs:element ref="Enabled" minOccurs="0" maxOccurs="1"/>
```

```
      <xs:element ref="Visible" minOccurs="0" maxOccurs="1"/>
      <xs:element ref="Required" minOccurs="0" maxOccurs="1"/>
    </xs:choice>
  </xs:sequence>
  <xs:attribute name="controlsId" type="xs:string" use="required"/>
  <xs:attribute name="style" type="xs:string" use="optional"/>
</xs:element>
```

親要素

PropertiesPanel、PropertiesSubPanel

子要素

Enabled、Layout、Required、Visible

関連する要素

ActionButton、ComboBoxControl、ExtensionObjectPanel、FieldAllocationList、ModelViewerPanel、OutputViewerPanel、SelectorPanel、StaticText、TabbedPanel、TextBrowserPanel

Tab 要素

タブ付きパネル内のタブを定義します。

表 *203. Tab* の属性

属性	使用	説明	有効な値
helpLink	オプション		*string*
ID	オプション		*string*
label	必須		*string*
labelKey	オプション		*string*
mnemonic	オプション		*string*
mnemonicKey	オプション		*string*
resourceKey	オプション		*string*

XML 表記

```
<xs:element name="Tab">
  <xs:sequence minOccurs="0" maxOccurs="unbounded">
    <xs:choice>
      <xs:element ref="PropertiesPanel"/>
      <xs:element ref="ExtensionObjectPanel"/>
      <xs:element ref="TextBrowserPanel"/>
      <xs:element ref="ModelViewerPanel"/>
      <xs:element ref="OutputViewerPanel"/>
      <xs:element ref="TabbedPanel"/>
    </xs:choice>
  </xs:sequence>
  <xs:attribute name="id" type="xs:string" use="optional"/>
  <xs:attribute name="label" type="xs:string" use="required"/>
  <xs:attribute name="labelKey" type="xs:string" use="optional"/>
  <xs:attribute name="mnemonic" type="xs:string" use="optional"/>
  <xs:attribute name="mnemonicKey" type="xs:string" use="optional"/>
  <xs:attribute name="resourceKey" type="xs:string" use="optional"/>
  <xs:attribute name="helpLink" type="xs:string" use="optional"/>
</xs:element>
```

親要素

タブ

子要素

ExtensionObjectPanel、ModelViewerPanel、OutputViewerPanel、PropertiesPanel、TabbedPanel、TextBrowserPanel

TabbedPanel 要素

タブ付きパネルを定義します。タブ付きパネルのタブに、別のパネルを追加することができます。

表 *204. TabbedPanel* の属性

属性	使用	説明	有効な値
style	オプション		**standard** **sidebar**

XML 表記

```
<xs:element name="TabbedPanel">
  <xs:sequence>
    <xs:choice>
      <xs:element ref="Layout" minOccurs="0" maxOccurs="1"/>
      <xs:element ref="Enabled" minOccurs="0" maxOccurs="1"/>
      <xs:element ref="Visible" minOccurs="0" maxOccurs="1"/>
      <xs:element ref="Required" minOccurs="0" maxOccurs="1"/>
    </xs:choice>
  </xs:sequence>
  <xs:sequence maxOccurs="unbounded">
    <xs:element ref="Tabs"/>
  </xs:sequence>
  <xs:attribute name="style" use="optional">
    <xs:simpleType>
      <xs:restriction base="xs:string">
        <xs:enumeration value="standard"/>
        <xs:enumeration value="sidebar"/>
      </xs:restriction>
    </xs:simpleType>
  </xs:attribute>
</xs:element>
```

親要素

PropertiesPanel、PropertiesSubPanel、Tab

子要素

Enabled、Layout、Required、Tabs、Visible

関連する要素

ActionButton、ComboBoxControl、ExtensionObjectPanel、FieldAllocationList、ModelViewerPanel、OutputViewerPanel、SelectorPanel、StaticText、SystemControls、TextBrowserPanel

TableControl 要素

構造のリストの値を追加、変更、削除するために使用できるテーブル コントロールを定義します。

表 *205. TableControl* の属性

属性	使用	説明	有効な値
columnWidths	オプション		*string*
columns	オプション		*positiveInteger*
description	オプション		*string*

表 205. *TableControl* の属性 (続き)

属性	使用	説明	有効な値
descriptionKey	オプション		*string*
label	オプション		*string*
labelAbove	オプション		*boolean*
labelKey	オプション		*string*
labelWidth	オプション		*positiveInteger*
mnemonic	オプション		*string*
mnemonicKey	オプション		*string*
property	必須		*string*
resourceKey	オプション		*string*
rows	オプション		*positiveInteger*
showLabel	オプション		*boolean*

XML 表記

```
<xs:element name="TableControl">
  <xs:sequence>
    <xs:choice>
      <xs:element ref="Layout" minOccurs="0" maxOccurs="1"/>
      <xs:element ref="Enabled" minOccurs="0" maxOccurs="1"/>
      <xs:element ref="Visible" minOccurs="0" maxOccurs="1"/>
      <xs:element ref="Required" minOccurs="0" maxOccurs="1"/>
    </xs:choice>
  </xs:sequence>
  <xs:attribute name="property" type="xs:string" use="required"/>
  <xs:attribute name="showLabel" type="xs:boolean" use="optional" default="true"/>
  <xs:attribute name="resourceKey" type="xs:string" use="optional"/>
  <xs:attribute name="label" type="xs:string" use="optional"/>
  <xs:attribute name="labelKey" type="xs:string" use="optional"/>
  <xs:attribute name="mnemonic" type="xs:string" use="optional"/>
  <xs:attribute name="mnemonicKey" type="xs:string" use="optional"/>
  <xs:attribute name="labelWidth" type="xs:positiveInteger" use="optional" default="1"/>
  <xs:attribute name="labelAbove" type="xs:boolean" use="optional" default="false"/>
  <xs:attribute name="description" type="xs:string" use="optional"/>
  <xs:attribute name="descriptionKey" type="xs:string" use="optional"/>
  <xs:attribute name="rows" type="xs:positiveInteger" use="optional" default="8"/>
  <xs:attribute name="columns" type="xs:positiveInteger" use="optional" default="20"/>
  <xs:attribute name="columnWidths" type="xs:string" use="optional"/>
</xs:element>
```

親要素

PropertiesPanel、PropertiesSubPanel

子要素

Enabled、Layout、Required、Visible

関連する要素

CheckBoxControl、CheckBoxGroupControl、ClientDirectoryChooserControl、ClientFileChooserControl、DBConnectionChooserControl、DBTableChooserControl、MultiFieldAllocationControl、MultiFieldChooserControl、MultiFieldSelectionTableControl、PasswordBoxControl、PropertyControl、RadioButtonGroupControl、ServerDirectoryChooserControl、ServerFileChooserControl、SingleFieldAllocationControl、SingleFieldChooserControl、SingleFieldValueChooserControl、SpinnerControl、TextAreaControl、TextBoxControl

Tabs 要素

タブ付きパネル内でのタブの順序を定義します。

表 206. *Tabs* の属性

属性	使用	説明	有効な値
defaultTab	オプション		*nonNegativeInteger*

XML 表記

```
<xs:element name="Tabs">
  <xs:sequence>
    <xs:element ref="Tab" minOccurs="0" maxOccurs="unbounded"/>
  </xs:sequence>
  <xs:attribute name="defaultTab" type="xs:nonNegativeInteger" use="optional" default="0"/>
</xs:element>
```

親要素

TabbedPanel、UserInterface

子要素

タブ

TextAreaControl 要素

文字列値を変更するために使用できる複数行テキスト域を定義します。

表 207. *TextAreaControl* の属性

属性	使用	説明	有効な値
columns	オプション		*positiveInteger*
description	オプション		*string*
descriptionKey	オプション		*string*
label	オプション		*string*
labelAbove	オプション		*boolean*
labelKey	オプション		*string*
labelWidth	オプション		*positiveInteger*
mnemonic	オプション		*string*
mnemonicKey	オプション		*string*
monospaced	オプション		*boolean*
property	必須		*string*
resourceKey	オプション		*string*
rows	オプション		*positiveInteger*
showLabel	オプション		*boolean*
wrapLines	オプション		*boolean*

XML 表記

```
<xs:element name="TextAreaControl">
  <xs:sequence>
    <xs:choice>
      <xs:element ref="Layout" minOccurs="0" maxOccurs="1"/>
      <xs:element ref="Enabled" minOccurs="0" maxOccurs="1"/>
      <xs:element ref="Visible" minOccurs="0" maxOccurs="1"/>
```

```
      <xs:element ref="Required" minOccurs="0" maxOccurs="1"/>
    </xs:choice>
  </xs:sequence>
  <xs:attribute name="property" type="xs:string" use="required"/>
  <xs:attribute name="showLabel" type="xs:boolean" use="optional" default="true"/>
  <xs:attribute name="resourceKey" type="xs:string" use="optional"/>
  <xs:attribute name="label" type="xs:string" use="optional"/>
  <xs:attribute name="labelKey" type="xs:string" use="optional"/>
  <xs:attribute name="mnemonic" type="xs:string" use="optional"/>
  <xs:attribute name="mnemonicKey" type="xs:string" use="optional"/>
  <xs:attribute name="labelWidth" type="xs:positiveInteger" use="optional" default="1"/>
  <xs:attribute name="labelAbove" type="xs:boolean" use="optional" default="false"/>
  <xs:attribute name="description" type="xs:string" use="optional"/>
  <xs:attribute name="descriptionKey" type="xs:string" use="optional"/>
  <xs:attribute name="rows" type="xs:positiveInteger" use="optional" default="8"/>
  <xs:attribute name="columns" type="xs:positiveInteger" use="optional" default="20"/>
  <xs:attribute name="wrapLines" type="xs:boolean" use="optional" default="true"/>
  <xs:attribute name="monospaced" type="xs:boolean" use="optional" default="false"/>
</xs:element>
```

親要素

PropertiesPanel、PropertiesSubPanel

子要素

Enabled、Layout、Required、Visible

関連する要素

CheckBoxControl、CheckBoxGroupControl、ClientDirectoryChooserControl、ClientFileChooserControl、DBConnectionChooserControl、DBTableChooserControl、MultiFieldAllocationControl、MultiFieldChooserControl、MultiFieldSelectionTableControl、PasswordBoxControl、PropertyControl、RadioButtonGroupControl、ServerDirectoryChooserControl、ServerFileChooserControl、SingleFieldAllocationControl、SingleFieldChooserControl、SingleFieldValueChooserControl、SpinnerControl、TableControl、TextBoxControl

TextBoxControl 要素

文字列値を変更するために使用できる単一行のテキスト コントロールを定義します。

表 *208. TextBoxControl* の属性

属性	使用	説明	有効な値
columns	オプション		*positiveInteger*
description	オプション		*string*
descriptionKey	オプション		*string*
label	オプション		*string*
labelAbove	オプション		*boolean*
labelKey	オプション		*string*
labelWidth	オプション		*positiveInteger*
mnemonic	オプション		*string*
mnemonicKey	オプション		*string*
property	必須		*string*
resourceKey	オプション		*string*
showLabel	オプション		*boolean*

XML 表記

```
<xs:element name="TextBoxControl">
  <xs:sequence>
    <xs:choice>
      <xs:element ref="Layout" minOccurs="0" maxOccurs="1"/>
      <xs:element ref="Enabled" minOccurs="0" maxOccurs="1"/>
      <xs:element ref="Visible" minOccurs="0" maxOccurs="1"/>
      <xs:element ref="Required" minOccurs="0" maxOccurs="1"/>
    </xs:choice>
  </xs:sequence>
  <xs:attribute name="property" type="xs:string" use="required"/>
  <xs:attribute name="showLabel" type="xs:boolean" use="optional" default="true"/>
  <xs:attribute name="resourceKey" type="xs:string" use="optional"/>
  <xs:attribute name="label" type="xs:string" use="optional"/>
  <xs:attribute name="labelKey" type="xs:string" use="optional"/>
  <xs:attribute name="mnemonic" type="xs:string" use="optional"/>
  <xs:attribute name="mnemonicKey" type="xs:string" use="optional"/>
  <xs:attribute name="labelWidth" type="xs:positiveInteger" use="optional" default="1"/>
  <xs:attribute name="labelAbove" type="xs:boolean" use="optional" default="false"/>
  <xs:attribute name="description" type="xs:string" use="optional"/>
  <xs:attribute name="descriptionKey" type="xs:string" use="optional"/>
  <xs:attribute name="columns" type="xs:positiveInteger" use="optional" default="20"/>
</xs:element>
```

親要素

PropertiesPanel、PropertiesSubPanel

子要素

Enabled、Layout、Required、Visible

関連する要素

CheckBoxControl、CheckBoxGroupControl、ClientDirectoryChooserControl、ClientFileChooserControl、DBConnectionChooserControl、DBTableChooserControl、MultiFieldAllocationControl、MultiFieldChooserControl、MultiFieldSelectionTableControl、PasswordBoxControl、PropertyControl、RadioButtonGroupControl、ServerDirectoryChooserControl、ServerFileChooserControl、SingleFieldAllocationControl、SingleFieldChooserControl、SingleFieldValueChooserControl、SpinnerControl、TableControl、TextAreaControl

TextBrowserPanel 要素

表 *209. TextBrowserPanel* の属性

属性	使用	説明	有効な値
columns	オプション		*string*
container	必須		*string*
rows	オプション		*string*
textFormat	必須		**plainText** **html** **rtf**
wrapLines	オプション		*boolean*

XML 表記

```
<xs:element name="TextBrowserPanel">
  <xs:sequence>
    <xs:choice>
      <xs:element ref="Layout" minOccurs="0" maxOccurs="1"/>
      <xs:element ref="Enabled" minOccurs="0" maxOccurs="1"/>
```

```
      <xs:element ref="Visible" minOccurs="0" maxOccurs="1"/>
      <xs:element ref="Required" minOccurs="0" maxOccurs="1"/>
    </xs:choice>
  </xs:sequence>
  <xs:attribute name="container" type="xs:string" use="required"/>
  <xs:attribute name="textFormat" use="required">
    <xs:simpleType>
      <xs:restriction base="xs:string">
        <xs:enumeration value="plainText"/>
        <xs:enumeration value="html"/>
        <xs:enumeration value="rtf"/>
      </xs:restriction>
    </xs:simpleType>
  </xs:attribute>
  <xs:attribute name="rows" type="xs:string" use="optional"/>
  <xs:attribute name="columns" type="xs:string" use="optional"/>
  <xs:attribute name="wrapLines" type="xs:boolean" use="optional" default="false"/>
</xs:element>
```

親要素

タブ

子要素

Enabled、Layout、Required、Visible

関連する要素

ActionButton、ComboBoxControl、ExtensionObjectPanel、FieldAllocationList、ModelViewerPanel、OutputViewerPanel、SelectorPanel、StaticText、SystemControls、TabbedPanel

TextReader 要素

表 *210. TextReader* の属性

属性	使用	説明	有効な値
delimiter	オプション		**tab** **comma** **semicolon** **colon** **bar**
delimiterChar	オプション		
fieldNamesIncluded	オプション		*boolean*
fileId	オプション		
path	オプション		
stringQuote	オプション		**doubleQuote** **singleQuote** **none**
stringQuoteChar	オプション		

XML 表記

```
<xs:element name="TextReader">
  <xs:attribute name="fileId" type="EVALUATED-STRING" use="optional"/>
  <xs:attribute name="path" type="EVALUATED-STRING" use="optional"/>
  <xs:attribute name="delimiter" use="optional" default="tab">
    <xs:simpleType>
      <xs:restriction base="xs:string">
```

```
        <xs:enumeration value="tab"/>
        <xs:enumeration value="comma"/>
        <xs:enumeration value="semicolon"/>
        <xs:enumeration value="colon"/>
        <xs:enumeration value="bar"/>
      </xs:restriction>
    </xs:simpleType>
  </xs:attribute>
  <xs:attribute name="delimiterChar" type="EVALUATED-STRING" use="optional"/>
  <xs:attribute name="stringQuote" use="optional" default="none">
    <xs:simpleType>
      <xs:restriction base="xs:string">
        <xs:enumeration value="doubleQuote"/>
        <xs:enumeration value="singleQuote"/>
        <xs:enumeration value="none"/>
      </xs:restriction>
    </xs:simpleType>
  </xs:attribute>
  <xs:attribute name="stringQuoteChar" type="EVALUATED-STRING" use="optional"/>
  <xs:attribute name="fieldNamesIncluded" type="xs:boolean" use="optional" default="true"/>
</xs:element>
```

親要素

Module

TextWriter 要素

表 *211. TextWriter* の属性

属性	使用	説明	有効な値
delimiter	オプション		**tab** **comma** **semicolon** **colon** **bar**
delimiterChar	オプション		
fields	オプション		
fileId	オプション		
includeFieldNames	オプション		*boolean*
stringQuote	オプション		**doubleQuote** **singleQuote** **none**
stringQuoteChar	オプション		

XML 表記

```
<xs:element name="TextWriter">
  <xs:attribute name="fileId" type="EVALUATED-STRING" use="optional"/>
  <xs:attribute name="fields" type="EVALUATED-STRING" use="optional"/>
  <xs:attribute name="delimiter" use="optional" default="tab">
    <xs:simpleType>
      <xs:restriction base="xs:string">
        <xs:enumeration value="tab"/>
        <xs:enumeration value="comma"/>
        <xs:enumeration value="semicolon"/>
        <xs:enumeration value="colon"/>
        <xs:enumeration value="bar"/>
      </xs:restriction>
    </xs:simpleType>
  </xs:attribute>
  <xs:attribute name="delimiterChar" type="EVALUATED-STRING" use="optional"/>
  <xs:attribute name="stringQuote" use="optional" default="none">
    <xs:simpleType>
```

```
      <xs:restriction base="xs:string">
        <xs:enumeration value="doubleQuote"/>
        <xs:enumeration value="singleQuote"/>
        <xs:enumeration value="none"/>
      </xs:restriction>
    </xs:simpleType>
  </xs:attribute>
  <xs:attribute name="stringQuoteChar" type="EVALUATED-STRING" use="optional"/>
  <xs:attribute name="includeFieldNames" type="xs:boolean" use="optional" default="true"/>
</xs:element>
```

親要素

Module

ToolbarItem 要素

ウィンドウ ツールバーに追加できる項目を定義します。

表 *212. ToolbarItem* の属性

属性	使用	説明	有効な値
action	必須		*string*
offset	オプション		*nonNegativeInteger*
separatorAfter	オプション		*boolean*
separatorBefore	オプション		*boolean*
showIcon	オプション		*boolean*
showLabel	オプション		*boolean*

XML 表記

```
<xs:element name="ToolbarItem">
  <xs:attribute name="action" type="xs:string" use="required"/>
  <xs:attribute name="showLabel" type="xs:boolean" use="optional" default="false"/>
  <xs:attribute name="showIcon" type="xs:boolean" use="optional" default="true"/>
  <xs:attribute name="separatorBefore" type="xs:boolean" use="optional" default="false"/>
  <xs:attribute name="separatorAfter" type="xs:boolean" use="optional" default="false"/>
  <xs:attribute name="offset" type="xs:nonNegativeInteger" use="optional" default="0"/>
</xs:element>
```

親要素

コントロール

UTF8Format 要素

XML 表記

```
<xs:element name="UTF8Format"/>
```

親要素

FileFormatType

UserInterface 要素

拡張オブジェクトまたはツールのユーザー インターフェースを定義します。

表 *213. UserInterface* の属性

属性	使用	説明	有効な値
actionHandler	オプション		*any*

表 213. *UserInterface* の属性 (続き)

属性	使用	説明	有効な値
frameClass	オプション		*any*
uiDelegate	オプション		*any*

XML 表記

```
<xs:element name="UserInterface">
  <xs:sequence>
    <xs:element ref="Icons" minOccurs="0"/>
    <xs:element ref="Controls" minOccurs="0"/>
    <xs:element ref="Tabs" minOccurs="0"/>
  </xs:sequence>
  <xs:attribute name="uiDelegate" use="optional"/>
  <xs:attribute name="frameClass" use="optional"/>
  <xs:attribute name="actionHandler" use="optional"/>
</xs:element>
```

親要素

DocumentOutput、Extension、InteractiveDocumentBuilder、InteractiveModelBuilder、ModelOutput、Node

子要素

Controls、Icons、Tabs

Validation 要素

プロパティー検証、メッセージの書式設定および表示の情報を定義します。

表 214. *Validation* の属性

属性	使用	説明	有効な値
severity	必須		**error** **warning** **information**
textKey	必須		*string*

XML 表記

```
<xs:element name="Validation">
  <xs:sequence>
    <xs:group ref="CONDITION-EXPRESSION">
      <xs:choice>
        <xs:element ref="Condition"/>
        <xs:element ref="And"/>
        <xs:element ref="Or"/>
        <xs:element ref="Not"/>
      </xs:choice>
    </xs:group>
    <xs:element ref="Arg" minOccurs="0" maxOccurs="unbounded"/>
  </xs:sequence>
  <xs:attribute name="textKey" type="xs:string" use="required"/>
  <xs:attribute name="severity" use="required">
    <xs:simpleType>
      <xs:restriction base="xs:string">
        <xs:enumeration value="error"/>
        <xs:enumeration value="warning"/>
        <xs:enumeration value="information"/>
      </xs:restriction>
    </xs:simpleType>
  </xs:attribute>
</xs:element>
```

親要素

Validations

子要素

And、Arg、Condition、Not、Or

Validations 要素

プロパティー検証の情報を定義します。

XML 表記

```
<xs:element name="Validations">
  <xs:sequence>
    <xs:element ref="Validation" minOccurs="1" maxOccurs="unbounded"/>
  </xs:sequence>
</xs:element>
```

親要素

ノード

子要素

Validation

Value 要素

単純な値。

表 *215. Value* の属性

属性	使用	説明	有効な値
value	必須		*string*

XML 表記

```
<xs:element name="Value" type="SIMPLE-VALUE">
  <xs:attribute name="value" type="xs:string" use="required"/>
</xs:element>
```

親要素

Attribute、Attribute、ListValue、ListValue、ListValue、Parameter

Values 要素

XML 表記

```
<xs:element name="Values">
  <xs:sequence>
    <xs:element name="Value" minOccurs="0" maxOccurs="unbounded">
    </xs:element>
  </xs:sequence>
</xs:element>
```

親要素

AddField、ChangeField、Field、Field、MissingValues、MissingValues、MissingValues

子要素

Value

Value 要素**:**

表 *216. Value* の属性

属性	使用	説明	有効な値
flagProperty	オプション		**trueValue** **falseValue**
value	必須		*string*
valueLabel	オプション		*string*

XML 表記

```
<xs:element name="Value" minOccurs="0" maxOccurs="unbounded">
  <xs:attribute name="value" type="xs:string" use="required"/>
  <xs:attribute name="valueLabel" type="xs:string" use="optional"/>
  <xs:attribute name="flagProperty">
    <xs:simpleType>
      <xs:restriction base="xs:string">
        <xs:enumeration value="trueValue"/>
        <xs:enumeration value="falseValue"/>
      </xs:restriction>
    </xs:simpleType>
  </xs:attribute>
</xs:element>
```

親要素

値

Values 要素

表 *217. Values* の属性

属性	使用	説明	有効な値
code	必須		*integer*
displayLabel	オプション		*string*
flagValue	オプション		*boolean*
value	必須		*string*

XML 表記

```
<xs:element name="Values" type="FIELD-VALUE">
  <xs:sequence>
    <xs:element name="DisplayLabel" type="DISPLAY-LABEL" minOccurs="0" maxOccurs="unbounded">
    </xs:element>
  </xs:sequence>
  <xs:attribute name="value" type="xs:string" use="required"/>
  <xs:attribute name="code" type="xs:integer" use="required"/>
  <xs:attribute name="flagValue" type="xs:boolean"/>
  <xs:attribute name="displayLabel" type="xs:string"/>
</xs:element>
```

親要素

AddField、ChangeField、Field、Field、MissingValues、MissingValues、MissingValues

子要素

DisplayLabel

DisplayLabel 要素: 指定された言語でのフィールドまたは値の表示ラベル。この displayLabel 属性は、特定の言語のラベルがない場合に使用できます。

表 *218. DisplayLabel* の属性

属性	使用	説明	有効な値
lang	オプション		*NMTOKEN*
value	必須		*string*

XML 表記

```
<xs:element name="DisplayLabel" type="DISPLAY-LABEL" minOccurs="0" maxOccurs="unbounded">
  <xs:attribute name="value" type="xs:string" use="required"/>
  <xs:attribute name="lang" type="xs:NMTOKEN" default="en"/>
</xs:element>
```

親要素

値

Visible 要素

どのような場合に UI コンポーネントを表示するかの条件を定義します。

XML 表記

```
<xs:element name="Visible">
  <xs:sequence>
    <xs:group ref="CONDITION-EXPRESSION" minOccurs="0">
      <xs:choice>
        <xs:element ref="Condition"/>
        <xs:element ref="And"/>
        <xs:element ref="Or"/>
        <xs:element ref="Not"/>
      </xs:choice>
    </xs:group>
  </xs:sequence>
</xs:element>
```

親要素

ActionButton、CheckBoxControl、CheckBoxGroupControl、ClientDirectoryChooserControl、ClientFileChooserControl、ComboBoxControl、DBConnectionChooserControl、DBTableChooserControl、ExtensionObjectPanel、FieldAllocationList、ItemChooserControl、ModelViewerPanel、MultiFieldAllocationControl、MultiFieldChooserControl、MultiFieldSelectionTableControl、MultiItemChooserControl、OutputViewerPanel、PasswordBoxControl、PropertiesPanel、PropertiesSubPanel、PropertyControl、RadioButtonGroupControl、SelectorPanel、ServerDirectoryChooserControl、ServerFileChooserControl、SingleFieldAllocationControl、SingleFieldChooserControl、SingleFieldValueChooserControl、SingleItemChooserControl、SpinnerControl、StaticText、SystemControls、TabbedPanel, TableControl, TextAreaControl、TextBoxControl、TextBrowserPanel

子要素

And、Condition、Not、Or

拡張タイプ

拡張タイプは、属性および子要素を追加することで、XML ドキュメントの要素を拡張します。 XML ドキュメントで拡張タイプを使用するには、要素の xsi:type 属性を使用して拡張タイプを指定します。これにより、拡張タイプによって定義された属性と要素を使用できます。

ItemChooserControl タイプ

表 *219. ItemChooserControl* の属性

属性	使用	説明	有効な値
catalog	必須		*string*
description	オプション		*string*
descriptionKey	オプション		*string*
label	オプション		*string*
labelAbove	オプション		*boolean*
labelKey	オプション		*string*
labelWidth	オプション		*positiveInteger*
mnemonic	オプション		*string*
mnemonicKey	オプション		*string*
property	必須		*string*
resourceKey	オプション		*string*
showLabel	オプション		*boolean*

XML 表記

```
<xs:complexType name="ItemChooserControl" mixed="false">
  <xs:sequence>
    <xs:choice>
      <xs:element ref="Layout" minOccurs="0" maxOccurs="1"/>
      <xs:element ref="Enabled" minOccurs="0" maxOccurs="1"/>
      <xs:element ref="Visible" minOccurs="0" maxOccurs="1"/>
      <xs:element ref="Required" minOccurs="0" maxOccurs="1"/>
    </xs:choice>
  </xs:sequence>
</xs:complexType>
```

拡張

ComboBoxControl

子要素

Enabled、Layout、Required、Visible

関連するタイプ

ItemChooserControl

特記事項

本書は米国 IBM が提供する製品およびサービスについて作成したものです。この資料の他の言語版を IBM から入手できる場合があります。ただし、これを入手するには、本製品または当該言語版製品を所有している必要がある場合があります。

本書に記載の製品、サービス、または機能が日本においては提供されていない場合があります。日本で利用可能な製品、サービス、および機能については、日本 IBM の営業担当員にお尋ねください。本書で IBM 製品、プログラム、またはサービスに言及していても、その IBM 製品、プログラム、またはサービスのみが使用可能であることを意味するものではありません。これらに代えて、IBM の知的所有権を侵害することのない、機能的に同等の製品、プログラム、またはサービスを使用することができます。ただし、IBM 以外の製品とプログラムの操作またはサービスの評価および検証は、お客様の責任で行っていただきます。

IBM は、本書に記載されている内容に関して特許権 (特許出願中のものを含む) を保有している場合があります。本書の提供は、お客様にこれらの特許権について実施権を許諾することを意味するものではありません。実施権についてのお問い合わせは、書面にて下記宛先にお送りください。

〒*103-8510*
東京都中央区日本橋箱崎町*19*番*21*号
日本アイ・ビー・エム株式会社
法務・知的財産
知的財産権ライセンス渉外

IBM およびその直接または間接の子会社は、本書を特定物として現存するままの状態で提供し、商品性の保証、特定目的適合性の保証および法律上の瑕疵担保責任を含むすべての明示もしくは黙示の保証責任を負わないものとします。国または地域によっては、法律の強行規定により、保証責任の制限が禁じられる場合、強行規定の制限を受けるものとします。

この情報には、技術的に不適切な記述や誤植を含む場合があります。本書は定期的に見直され、必要な変更は本書の次版に組み込まれます。IBM は予告なしに、随時、この文書に記載されている製品またはプログラムに対して、改良または変更を行うことがあります。

本書において IBM 以外の Web サイトに言及している場合がありますが、便宜のため記載しただけであり、決してそれらの Web サイトを推奨するものではありません。それらの Web サイトにある資料は、この IBM 製品の資料の一部ではありません。それらの Web サイトは、お客様の責任でご使用ください。

IBM は、お客様が提供するいかなる情報も、お客様に対してなんら義務も負うことのない、自ら適切と信ずる方法で、使用もしくは配布することができるものとします。

本プログラムのライセンス保持者で、(i) 独自に作成したプログラムとその他のプログラム (本プログラムを含む) との間での情報交換、および (ii) 交換された情報の相互利用を可能にすることを目的として、本プログラムに関する情報を必要とする方は、下記に連絡してください。

IBM Director of Licensing
IBM Corporation
North Castle Drive, MD-NC119
Armonk, NY 10504-1785
US

本プログラムに関する上記の情報は、適切な使用条件の下で使用することができますが、有償の場合もあります。

本書で説明されているライセンス・プログラムまたはその他のライセンス資料は、IBM 所定のプログラム契約の契約条項、IBM プログラムのご使用条件、またはそれと同等の条項に基づいて、IBM より提供されます。

記載されている性能データとお客様事例は、例として示す目的でのみ提供されています。実際の結果は特定の構成や稼働条件によって異なります。

IBM 以外の製品に関する情報は、その製品の供給者、出版物、もしくはその他の公に利用可能なソースから入手したものです。IBM は、それらの製品のテストは行っておりません。したがって、他社製品に関する実行性、互換性、またはその他の要求については確証できません。 IBM 以外の製品の性能に関する質問は、それらの製品の供給者にお願いします。

IBM の将来の方向または意向に関する記述については、予告なしに変更または撤回される場合があり、単に目標を示しているものです。

本書には、日常の業務処理で用いられるデータや報告書の例が含まれています。より具体性を与えるために、それらの例には、個人、企業、ブランド、あるいは製品などの名前が含まれている場合があります。これらの名前はすべて架空のものであり、類似する個人や企業が実在しているとしても、それは偶然にすぎません。

商標

IBM、IBM ロゴおよび ibm.com は、世界の多くの国で登録された International Business Machines Corporation の商標です。他の製品名およびサービス名等は、それぞれ IBM または各社の商標である場合があります。現時点での IBM の商標リストについては、http://www.ibm.com/legal/copytrade.shtml をご覧ください。

Adobe、Adobe ロゴ、PostScript、PostScript ロゴは、Adobe Systems Incorporated の米国およびその他の国における登録商標または商標です。

インテル、Intel、Intel ロゴ、Intel Inside、Intel Inside ロゴ、Centrino、Intel Centrino ロゴ、Celeron、Xeon、Intel SpeedStep、Itanium、および Pentium は、Intel Corporation または子会社の米国およびその他の国における商標または登録商標です。

Linux は、Linus Torvalds の米国およびその他の国における登録商標です。

Microsoft、Windows、Windows NT および Windows ロゴは、Microsoft Corporation の米国およびその他の国における商標です。

UNIX は The Open Group の米国およびその他の国における登録商標です。

Java およびすべての Java 関連の商標およびロゴは Oracle やその関連会社の米国およびその他の国における商標または登録商標です。

製品資料に関するご使用条件

これらの資料は、以下のご使用条件に同意していただける場合に限りご使用いただけます。

適用範囲

IBM Web サイトの「ご利用条件」に加えて、以下のご使用条件が適用されます。

個人使用

これらの資料は、すべての著作権表示その他の所有権表示をしていただくことを条件に、非商業的な個人による使用目的に限り複製することができます。ただし、IBM の明示的な承諾をえずに、これらの資料またはその一部について、二次的著作物を作成したり、配布 (頒布、送信を含む) または表示 (上映を含む) することはできません。

商業的使用

これらの資料は、すべての著作権表示その他の所有権表示をしていただくことを条件に、お客様の企業内に限り、複製、配布、および表示することができます。ただし、IBM の明示的な承諾をえずにこれらの資料の二次的著作物を作成したり、お客様の企業外で資料またはその一部を複製、配布、または表示することはできません。

権利

ここで明示的に許可されているもの以外に、資料や資料内に含まれる情報、データ、ソフトウェア、またはその他の知的所有権に対するいかなる許可、ライセンス、または権利を明示的にも黙示的にも付与するものではありません。

資料の使用が IBM の利益を損なうと判断された場合や、上記の条件が適切に守られていないと判断された場合、IBM はいつでも自らの判断により、ここで与えた許可を撤回できるものとさせていただきます。

お客様がこの情報をダウンロード、輸出、または再輸出する際には、米国のすべての輸出入 関連法規を含む、すべての関連法規を遵守するものとします。

IBM は、これらの資料の内容についていかなる保証もしません。これらの資料は、特定物として現存するままの状態で提供され、商品性の保証、特定目的適合性の保証および法律上の瑕疵担保責任を含むすべての明示もしくは黙示の保証責任なしで提供されます。

索引

日本語, 数字, 英字, 特殊文字の順に配列されています。なお, 濁音と半濁音は清音と同等に扱われています。

［ア行］

［カ行］

［サ行］

［タ行］

［ナ行］

［ハ行］

［マ行］

［ヤ行］

［ラ行］

［ワ行］

A

B

C

D

E

F

H

I

J

T

U

V

X

Printed in Japan

日本アイ・ビー・エム株式会社
〒103-8510 東京都中央区日本橋箱崎町19-21