

Guide du développeur CLEF de IBM  
SPSS Modeler 14.2



*Remarque* : avant d'utiliser ces informations et le produit dont elles traitent, consultez les informations générales dans la rubrique Avis sur p. .

Ce document contient des informations sur la propriété de SPSS Inc., an IBM Company. Il est fourni sous accord de licence et est protégé par la loi sur le copyright. Les informations contenues dans cette publication n'incluent pas les garanties de produit et aucune des provisions de ce manuel ne devra être interprétée comme tel.

Lorsque vous envoyez des informations à IBM ou SPSS, vous octroyez à IBM et SPSS le droit non exclusif d'utiliser ou de distribuer ces informations de la façon dont IBM ou SPSS le juge approprié sans aucune obligation envers vous.

© **Copyright IBM Corporation 1994, 2011..**

---

# Préface

IBM® SPSS® Modeler est le puissant utilitaire de Data mining de IBM Corp.. SPSS Modeler aide les entreprises et les organismes à améliorer leurs relations avec les clients et les citoyens grâce à une compréhension approfondie des données. A l'aide des connaissances plus précises obtenues par le biais de SPSS Modeler, les entreprises et les organismes peuvent conserver les clients rentables, identifier les opportunités de vente croisée, attirer de nouveaux clients, détecter les éventuelles fraudes, réduire les risques et améliorer les services gouvernementaux.

L'interface visuelle de SPSS Modeler met à contribution les compétences professionnelles de l'utilisateur, ce qui permet d'obtenir des modèles prédictifs plus efficaces et de trouver des solutions plus rapidement. SPSS Modeler dispose de nombreuses techniques de modélisation, telles que les algorithmes de prévision, de classification, de segmentation et de détection d'association. Une fois les modèles créés, l'utilisateur peut utiliser IBM® SPSS® Modeler Solution Publisher pour les remettre aux responsables, où qu'ils se trouvent dans l'entreprise, ou pour les transférer vers une base de données.

## ***A propos de IBM Business Analytics***

Le logiciel IBM Business Analytics fournit des informations complètes, cohérentes et précises que les preneurs de décision utilisent avec confiance pour améliorer la performance du marché. Un portefeuille étendu d'outils de [business intelligence](#), d'[analyses prédictives](#), de [performance financière et de gestion de stratégie](#), et des [applications analytiques](#) offre des connaissances claires, immédiates et applicables pour améliorer l'efficacité actuelle ainsi que la capacité de prévoir les résultats futurs. Combinées avec de riches solutions industrielles, des pratiques éprouvées et des services professionnels, les organisations de toutes tailles peuvent atteindre la productivité la plus élevée, automatiser des décisions en toute tranquillité et fournir de meilleurs résultats.

Dans le cadre de ce portefeuille, le logiciel IBM SPSS Predictive Analytics aide les organisations à prévoir des événements futurs et à agir en conséquence pour mener à de meilleurs résultats. Des clients dans le domaine commercial, gouvernemental et académique à travers le monde font confiance à la technologie IBM SPSS et considèrent qu'elle représente un avantage compétitif pour attirer, retenir et ajouter des clients, tout en réduisant la fraude et en atténuant les risques. En incorporant le logiciel IBM SPSS dans leur opérations quotidiennes, les organisations deviennent des entreprises prédictives – capables de diriger et d'automatiser les décisions pour atteindre les buts qu'ils se sont fixés et obtenir des avantages compétitifs sensibles. Pour informations supplémentaires ou pour joindre un revendeur, visitez le site <http://www.ibm.com/spss>.

## ***Assistance technique***

L'assistance technique est à la disposition des clients pour la maintenance des produits. Les clients peuvent contacter l'assistance technique pour obtenir de l'aide concernant l'utilisation des produits IBM Corp. ou l'installation dans l'un des environnements matériels pris en charge. Pour joindre l'assistance technique, consultez le site Web de IBM Corp. à l'adresse <http://www.ibm.com/support>. Lorsque vous contactez l'assistance technique, n'oubliez pas de préparer vos identifiants, le nom de votre société et votre contrat d'assistance.

---

# Contenu

## 1 Aperçu 1

Introduction à CLEF . . . . .	1
Architecture du système . . . . .	1
Composants côté client . . . . .	1
Composants côté serveur . . . . .	2
Fonctions de CLEF . . . . .	3
Fichier de spécifications . . . . .	4
Noeuds . . . . .	4
Data Model . . . . .	4
Fichiers d'entrée et de sortie . . . . .	5
Interfaces de programmation d'application (API) . . . . .	5
Structure de fichier . . . . .	6
Composants côté client . . . . .	6
Composants côté serveur . . . . .	8

## 2 Noeuds 9

Présentation des noeuds . . . . .	9
Noeuds Lecteur de données . . . . .	10
Noeuds de Transformation de données . . . . .	11
Noeuds de création de modèles . . . . .	11
Noeuds de création de document . . . . .	12
Noeuds applicateurs de modèle . . . . .	13
Noeuds Rédacteur de données . . . . .	14
Menus, barres d'outils et palettes . . . . .	14
Menus et sous-menus . . . . .	14
Barres d'outils . . . . .	15
Palettes et sous-palettes . . . . .	16
Conception d'icônes de noeud . . . . .	17
Bordures . . . . .	18
Arrière-plans . . . . .	19
Exigences graphiques . . . . .	20
Création d'images personnalisées . . . . .	20
Ajouter les fichiers image à la Spécification du noeud . . . . .	21
Conception des boîtes de dialogue . . . . .	21
A propos des boîtes de dialogue de noeud . . . . .	22
Instructions de conception des boîtes de dialogue . . . . .	23
Composants de la boîte de dialogue . . . . .	24
Conception des fenêtres de sortie . . . . .	30

### **3 CLEF Exemples** **32**

Présentation des exemples . . . . .	32
Activation des exemples . . . . .	32
Noeud de lecture de données (Apache Log Reader) . . . . .	33
Noeud de transformation de données (URL Parser) . . . . .	34
Noeud de création de document (Web Status Report) . . . . .	34
Noeud de création de modèle (Interaction) . . . . .	35
Analyse des fichiers de spécifications . . . . .	36
Analyse du code source . . . . .	36
Suppression des exemples . . . . .	37

### **4 Fichier de spécifications** **38**

Présentation des fichiers de spécifications . . . . .	38
Exemple d'un fichier de spécifications . . . . .	39
Déclaration XML . . . . .	40
Élément Extension . . . . .	41
Section Extension Details . . . . .	41
Section Resources . . . . .	42
Groupes . . . . .	43
Fichiers Jar . . . . .	43
Bibliothèques partagées . . . . .	44
Informations d'aide . . . . .	45
Section Common Objects . . . . .	45
Types de propriétés . . . . .	46
Ensembles de propriétés . . . . .	47
Types de conteneurs . . . . .	48
Actions . . . . .	50
Catalogs . . . . .	51
Section User Interface (Palettes) . . . . .	54
Section Object Definition . . . . .	59
Identificateur d'objet . . . . .	60
Concepteur de modèle . . . . .	64
Créateur de document . . . . .	64
Fournisseur de modèle . . . . .	64
Propriétés . . . . .	65
Containers . . . . .	67
Interface utilisateur . . . . .	68

Execution . . . . .	68
Modèle de données de sortie. . . . .	73
Constructors . . . . .	75
Fonctions communes . . . . .	75
Types de valeur . . . . .	75
Chaînes évaluées . . . . .	79
Operations. . . . .	80
Champs et métadonnées de champ . . . . .	85
Ensembles de champs. . . . .	86
Rôles . . . . .	87
Opérateurs logiques . . . . .	88
Conditions . . . . .	89
Utilisation de noeuds CLEF dans les scripts. . . . .	93
Maintenance de la compatibilité descendante . . . . .	94

## **5 Création de modèles et de documents 96**

Introduction à la création de modèles et de documents . . . . .	96
Modèles . . . . .	96
Documents . . . . .	97
Constructors . . . . .	97
Création de modèles. . . . .	97
Concepteur de modèle. . . . .	98
Sortie du modèle . . . . .	107
Création de modèles interactifs . . . . .	108
Modélisation automatisée . . . . .	113
Application de modèles . . . . .	121
Création de documents. . . . .	121
Créateur de document. . . . .	121
Sortie de document . . . . .	122
Utilisation des constructeurs . . . . .	123
Création de sortie de modèle . . . . .	123
Création de sortie de document. . . . .	124
Création de créateur de modèles interactif . . . . .	125
Création d'applicateur de modèle . . . . .	126

## **6 Création d'interfaces utilisateur 128**

A propos des interfaces utilisateur . . . . .	128
Section Interface utilisateur . . . . .	131

Icônes . . . . .	134
Controls . . . . .	135
Menus . . . . .	135
Éléments de menu . . . . .	137
Éléments de barre d'outils . . . . .	138
Exemple : Ajout à la fenêtre principale . . . . .	139
Onglets . . . . .	140
Touches d'accès et raccourcis clavier . . . . .	142
Spécifications de panneau . . . . .	143
Panneau Navigateur de texte . . . . .	144
Panneau Objet d'extension . . . . .	146
Panneau Propriétés . . . . .	147
Panneau Visualiseur de modèle . . . . .	149
Spécifications des contrôles de propriétés . . . . .	151
Contrôles du composant IU . . . . .	151
Contrôles du panneau Propriétés . . . . .	155
Contrôleurs . . . . .	158
Présentations du Contrôle Propriété . . . . .	185
Présentation du contrôle standard . . . . .	185
Présentation de contrôle personnalisée . . . . .	187
Fenêtres de sortie personnalisées . . . . .	201

## **7 Ajout d'un système d'aide 203**

Types de système d'aide . . . . .	203
Aide au format HTML . . . . .	203
JavaHelp . . . . .	204
Mise en oeuvre d'un système d'aide . . . . .	205
Définition de l'emplacement et du type du système d'aide . . . . .	205
Désignation d'une rubrique particulière à afficher . . . . .	206

## **8 Localisation et accessibilité 208**

Introduction . . . . .	208
Localisation . . . . .	208
Fichiers de propriétés . . . . .	209
Fichiers d'aide . . . . .	213
Test d'un noeud CLEF localisé . . . . .	214
Accessibilité . . . . .	215

## **9 Programmation** **216**

A propos de la programmation pour les noeuds CLEF . . . . .	216
CLEF Documentation de l'API . . . . .	216
API côté client . . . . .	217
Classes de l'API côté client . . . . .	217
Utilisation de l'API côté client . . . . .	217
API Predictive Server (API PS) . . . . .	218
API côté serveur. . . . .	218
Architecture . . . . .	218
Fonctions de service . . . . .	219
Fonctions de rappel . . . . .	221
Flux de processus . . . . .	223
Fonctions de l'API côté serveur . . . . .	225
Gestion des erreurs . . . . .	240
API d'analyse XML . . . . .	241
Utilisation de l'API côté serveur. . . . .	241
Conseils de programmation côté serveur . . . . .	242

## **10 Test et distribution** **246**

Test des extensions CLEF . . . . .	246
Test d'une extension CLEF . . . . .	246
Débogage d'une extension CLEF . . . . .	247
Distribution des extensions CLEF . . . . .	250
Installation des extensions CLEF . . . . .	251
Désinstallation des extensions CLEF . . . . .	251

## **Annexes**

### **A Schéma XML CLEF** **252**

Références des éléments CLEF . . . . .	252
Action Elément . . . . .	252
ActionButton Elément . . . . .	253
Actions Elément. . . . .	253
AddField Elément. . . . .	254
And Elément . . . . .	258



Attribute Élément . . . . .	258
BinaryFormat Élément . . . . .	259
Catalog Élément . . . . .	259
Catalogs Élément . . . . .	260
ChangeField Élément . . . . .	260
CheckBoxControl Élément . . . . .	264
CheckBoxGroupControl Élément . . . . .	265
ClientDirectoryChooserControl Élément . . . . .	266
ClientFileChooserControl Élément . . . . .	267
ComboBoxControl Élément . . . . .	268
Command Élément . . . . .	270
CommonObjects Élément . . . . .	270
Condition Élément . . . . .	271
Constraint Élément . . . . .	273
Constructors Élément . . . . .	274
Container Élément . . . . .	274
ContainerFile Élément . . . . .	275
ContainerTypes Élément . . . . .	275
Controls Élément . . . . .	275
CreateDocument Élément . . . . .	276
CreateDocumentOutput Élément . . . . .	277
CreateInteractiveDocumentBuilder Élément . . . . .	277
CreateInteractiveModelBuilder Élément . . . . .	278
CreateModel Élément . . . . .	279
CreateModelApplier Élément . . . . .	280
CreateModelOutput Élément . . . . .	281
DatabaseConnectionValue Élément . . . . .	282
DataFile Élément . . . . .	283
DataFormat Élément . . . . .	283
DataModel Élément . . . . .	283
DBConnectionChooserControl Élément . . . . .	290
DBTableChooserControl Élément . . . . .	291
DefaultValue Élément . . . . .	292
DelimitedDataFormat Élément . . . . .	294
DisplayLabel Élément . . . . .	295
DocumentBuilder Élément . . . . .	295
DocumentOutput Élément . . . . .	296
DocumentType Élément . . . . .	297
Enabled Élément . . . . .	298
Enumeration Élément . . . . .	299
ErrorDetail Élément . . . . .	299
Executable Élément . . . . .	301
Execution Élément . . . . .	302
Extension Élément . . . . .	302
ExtensionDetails Élément . . . . .	303

ExtensionObjectPanel Elément . . . . .	303
Field Elément. . . . .	304
FieldFormats Elément. . . . .	307
FieldGroup Elément . . . . .	309
FieldGroups Elément . . . . .	310
FileFormatType Elément. . . . .	311
FileFormatTypes Elément . . . . .	312
ForEach Elément . . . . .	312
Icon Elément . . . . .	313
Icons Elément . . . . .	314
InputFiles Elément . . . . .	314
InteractiveDocumentBuilder Elément. . . . .	315
InteractiveModelBuilder Elément. . . . .	316
Layout Elément . . . . .	317
License Elément . . . . .	319
ListValue Elément . . . . .	319
MapValue Elément. . . . .	320
Menu Elément . . . . .	324
MenuItem Elément . . . . .	326
MissingValues Elément . . . . .	327
ModelBuilder Elément . . . . .	330
ModelOutput Elément . . . . .	339
ModelProvider Elément . . . . .	340
ModelType Elément . . . . .	340
ModelViewerPanel Elément. . . . .	341
Module Elément. . . . .	342
MultiFieldChooserControl Elément . . . . .	342
MultitemChooserControl Elément . . . . .	344
Node Elément . . . . .	345
Not Elément. . . . .	347
NumberFormat Elément . . . . .	348
NumericInfo Elément. . . . .	349
Option Elément . . . . .	349
OptionCode Elément . . . . .	350
Or Elément. . . . .	350
OutputDataModel Elément. . . . .	351
OutputFiles Elément. . . . .	352
Palette Elément . . . . .	352
Parameters Elément . . . . .	353
PasswordBoxControl Elément . . . . .	355
Properties Elément . . . . .	356
PropertiesPanel Elément . . . . .	356
PropertiesSubPanel Elément . . . . .	358
Property Elément. . . . .	359
PropertyControl Elément . . . . .	361

PropertyGroup Elément . . . . .	362
PropertySets Elément . . . . .	362
PropertyType Elément . . . . .	363
PropertyTypes Elément . . . . .	364
RadioButtonGroupControl Elément. . . . .	364
Range Elément . . . . .	366
Range Elément . . . . .	366
RemoveField Elément. . . . .	366
Resources Elément . . . . .	367
Run Elément . . . . .	369
SelectorPanel Elément . . . . .	370
ServerDirectoryChooserControl Elément . . . . .	371
ServerFileChooserControl Elément. . . . .	372
SetContainer Elément . . . . .	373
SetProperty Elément . . . . .	374
SingleFieldChooserControl Elément . . . . .	374
SingleFieldValueChooserControl Elément. . . . .	376
SingleItemChooserControl Elément . . . . .	377
SpinnerControl Elément . . . . .	378
SPSSDataFormat Elément . . . . .	379
StaticText Elément. . . . .	380
StatusCode Elément. . . . .	380
StatusCodes Elément. . . . .	381
StatusDetail Elément . . . . .	381
Structure Elément . . . . .	383
StructuredValue Elément . . . . .	384
SystemControls Elément . . . . .	386
Tab Elément. . . . .	387
TabbedPanel Elément . . . . .	387
TableControl Elément. . . . .	388
Tabs Elément. . . . .	389
TextAreaControl Elément . . . . .	390
TextBoxControl Elément. . . . .	391
TextBrowserPanel Elément . . . . .	392
ToolBarItem Elément . . . . .	393
UserInterface Elément. . . . .	394
UTF8Format Elément . . . . .	394
Value Elément . . . . .	394
Values Elément . . . . .	395
Values Elément . . . . .	396
Visible Elément . . . . .	397
Étendus Types . . . . .	398

***B Avis***

***399***

***Index***

***402***

# Aperçu

## Introduction à CLEF

**Component-Level Extension Framework (CLEF)** est un mécanisme qui permet d'ajouter des extensions fournies par l'utilisateur à la fonctionnalité standard de IBM® SPSS® Modeler. Une extension comprend habituellement une bibliothèque partagée, par exemple une routine de traitement des données ou un algorithme de modélisation qui est ajouté à SPSS Modeler et qui est rendu disponible à partir d'une nouvelle entrée dans un menu ou en tant que nouveau noeud dans la palette des noeuds.

Pour ce faire, SPSS Modeler requiert certains détails relatifs au programme personnalisé, notamment son nom, les paramètres de commande qui doivent lui être transmis, la manière dont SPSS Modeler doit présenter les options au programme et les résultats à l'utilisateur, etc. Pour fournir ces informations, vous utilisez un fichier au format XML appelé **fichier de spécifications**. SPSS Modeler traduit les informations figurant dans ce fichier dans une nouvelle entrée de menu ou une définition de noeud.

L'utilisation de CLEF présente différents avantages, notamment :

- Il fournit un environnement simple à utiliser, extrêmement flexible et robuste qui permet aux ingénieurs, aux consultants et aux utilisateurs finaux d'intégrer de nouvelles fonctions à SPSS Modeler.
- Il garantit que les modules d'extension peuvent se présenter et se comporter de la même manière que les modules SPSS Modeler natifs.
- Il permet aux noeuds d'extension de s'exécuter avec une vitesse et une efficacité aussi proches que possible de celles des noeuds SPSS Modeler natifs.

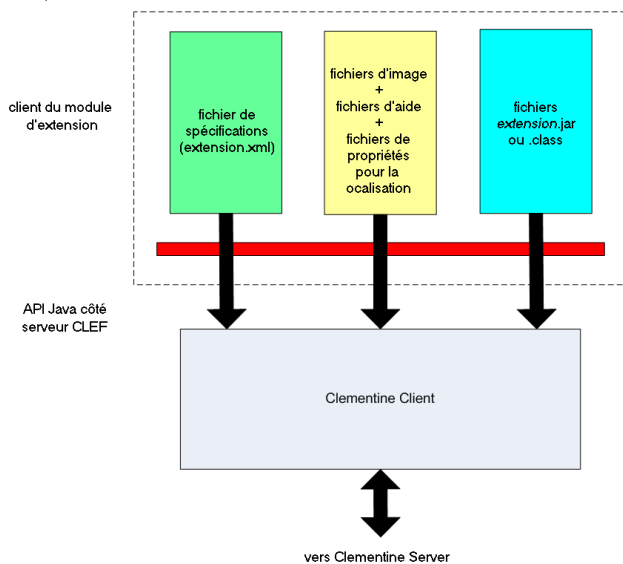
## Architecture du système

A l'instar de IBM® SPSS® Modeler lui-même, CLEF utilise une architecture client/serveur à deux niveaux, où les niveaux peuvent résider soit sur le même ordinateur, soit sur deux ordinateurs différents.

### Composants côté client

Les composants du niveau client sont présentés ici.

Figure 1-1  
Composants côté client

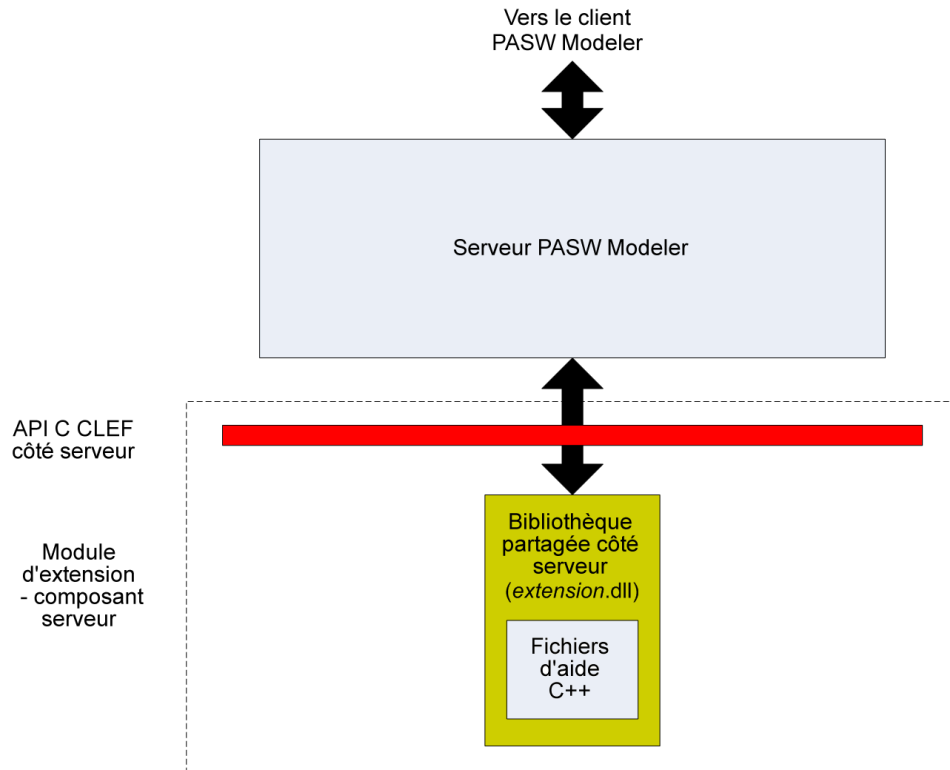


- **Fichier de spécifications.** Répertorie les propriétés, les formats, les modifications de modèle de données, les commandes et les autres caractéristiques qui sont définies par l'extension.
- **Fichiers image.** Ils contiennent les images utilisées pour identifier un nœud dans l'extension.
- **Fichiers d'aide.** Ils permettent d'afficher les informations d'aide à propos de l'extension.
- **Fichiers de propriétés.** Ils contiennent des chaînes de texte comprenant les noms, les étiquettes et les messages affichés à l'écran par l'extension.
- **Fichiers Java .jar ou .class.** Ils contiennent les éventuelles ressources Java utilisées par l'extension.
- **Interface de programmation d'application (API) Java.** Elle peut être utilisée par les extensions qui requièrent des commandes supplémentaires, des composants de l'interface utilisateur ou une interactivité non fournie directement par le fichier de spécifications.

## Composants côté serveur

Les composants du niveau serveur sont présentés ici.

Figure 1-2  
Composants côté serveur



- **API à base C pour les bibliothèques partagées.** Il couvre des aspects tels que la définition et l'obtention des paramètres d'exécution, la persistance de ces paramètres, les commentaires relatifs à l'exécution, le contrôle du travail (par exemple, l'interruption d'exécution), la génération SQL et les objets retournés.
- **Bibliothèque partagée côté serveur.** Une bibliothèque de liens dynamiques (DLL) prenant en charge l'exécution des nœuds. Les programmes d'aide C++ sont des wrappers de certaines API à base C qui sont fournies en tant que code source et peuvent facilement être compilées dans un module CLEF C++.

## Fonctions de CLEF

Les sections suivantes présentent un certain nombre de fonctions clés de CLEF :

- Fichier de spécifications
- Nœuds
- Modèle de données
- Fichiers d'entrée et de sortie
- Interfaces de programmation d'application (API)

## **Fichier de spécifications**

Le fichier de spécifications CLEF est un fichier XML contenant des spécifications structurées qui décrivent le comportement de la nouvelle extension. Un fichier de spécifications décrit :

- les ressources partagées requises par l'extension (par exemple, les groupes de texte localisé et les bibliothèques partagées côté serveur).
- les définitions courantes, telles que les types de fichier ou de propriétés.
- les nouveaux objets que l'utilisateur final peut employer, par exemple les nœuds et les modèles de sortie.

Au démarrage de IBM® SPSS® Modeler, les fichiers de spécifications sont chargés à partir de l'emplacement où ils résident, de sorte que les fonctions définies dans les fichiers sont immédiatement disponibles.

Pour plus d'informations, reportez-vous à [le chapitre 4, Fichier de spécifications sur p. 38](#).

## **Noeuds**

Lorsque vous ajoutez à IBM® SPSS® Modeler une extension qui met en oeuvre un nouveau noeud, vous devez d'abord choisir le type de noeud à créer (par exemple, si le noeud génère un modèle ou transforme simplement les données). [Pour plus d'informations, reportez-vous à la section Présentation des noeuds dans le chapitre 2 sur p. 9](#).

Après avoir créé le fichier de spécifications et les éventuelles classes Java et bibliothèques partagées nécessaires, vous copiez les fichiers à des emplacements spécifiques où SPSS Modeler peut les lire. La prochaine fois que vous démarrerez SPSS Modeler, le nouveau noeud sera ajouté à la palette appropriée et il sera prêt à l'emploi.

## **Data Model**

Le **modèle de données** représente la structure des données circulant dans le flux IBM® SPSS® Modeler. Si vous décrivez les données à ce stade du flux, le modèle correspond aux informations affichées dans le noeud Typer. Il répertorie les noms des champs existants à un emplacement particulier du flux et décrit leur type.

Lorsque vous ajoutez un noeud à SPSS Modeler, considérez la façon dont le modèle de données transmis dans le noeud affecte le comportement de ce noeud. Par exemple, un noeud Calculer prend un modèle de données d'entrée, il y ajoute un nouveau champ, puis il produit un modèle de données de sortie transmis au noeud suivant dans le flux SPSS Modeler. En revanche, un noeud Graphiques prend un modèle de données et ne produit aucun modèle de données de sortie car les données ne sont pas transmises aux noeuds suivants. SPSS Modeler doit savoir ce qui arrivera au modèle de données afin que les noeuds suivants puissent présenter les informations correctes à propos des champs disponibles. Les informations sur le modèle de données du fichier de spécifications apportent à SPSS Modeler les renseignements nécessaires à la cohérence du modèle de données dans le flux.

Selon que les données sont entrantes ou sortantes, ou qu'elles transitent par le noeud, le fichier de spécifications doit comporter la description du modèle de données pour l'entrée, la sortie ou les deux. Un noeud CLEF peut affecter le modèle de données en ajoutant de nouveaux champs aux



champs qui sont transmis au noeud ou en remplaçant les champs qui arrivent dans le noeud par de nouveaux champs générés par le programme lui-même. L'élément `OutputDataModel` du fichier de spécifications décrit les effets du noeud CLEF sur le modèle de données. [Pour plus d'informations, reportez-vous à la section Modèle de données de sortie dans le chapitre 4 sur p. 73.](#)

## ***Fichiers d'entrée et de sortie***

Il est possible de spécifier un ou plusieurs fichiers temporaires à générer avant l'exécution d'un noeud CLEF. On les appelle **fichiers d'entrée**, car ils sont entrés dans l'exécution du noeud sur le serveur. Par exemple, un noeud de création de modèle peut avoir un conteneur de modèle dont le contenu est transféré vers le fichier d'entrée spécifié lors de l'exécution du noeud. [Pour plus d'informations, reportez-vous à la section Fichiers d'entrée dans le chapitre 4 sur p. 70.](#)

D'autres fichiers temporaires sont générés au cours de l'exécution du noeud sur le serveur ; par exemple, les résultats de l'exécution d'un noeud de création de modèle ou de création de document. On les appelle **fichiers de sortie**, et ils sont restitués au client à la suite de l'exécution du noeud. [Pour plus d'informations, reportez-vous à la section Fichiers de résultats dans le chapitre 4 sur p. 70.](#)

## ***Interfaces de programmation d'application (API)***

Selon ce que vous souhaitez que l'extension fasse, vous pouvez être amené à utiliser une interface de programmation d'application (API). Pour une transformation simple de données, vous pouvez définir entièrement le traitement nécessaire dans le fichier de spécifications. Toutefois, pour les exigences plus avancées, vous devrez vous mettre en interface avec une ou plusieurs des API disponibles :

- CLEF API côté client
- CLEF API côté serveur
- API Predictive Server (API PS)

L'**CLEF API côté client** est une API Java qui peut être utilisée par les extensions qui requièrent des commandes supplémentaires, des composants de l'interface utilisateur ou une interactivité non fournie directement par le fichier de spécifications.

L'**API côté serveur** CLEF est une API à base C qui couvre des aspects tels que la définition et l'obtention des paramètres d'exécution, la persistance de ces paramètres, les commentaires relatifs à l'exécution, le contrôle du travail (par exemple, l'interruption d'exécution), la génération SQL et les objets retournés.

L'**API Predictive Server** est une API Java qui expose la fonctionnalité de IBM® SPSS® Modeler à utiliser par des applications qui requièrent des capacités de Data mining et d'analyse prédictive.

Pour plus d'informations, reportez-vous à [le chapitre 9, Programmation sur p. 216.](#)

## **Structure de fichier**

Une extension CLEF comprend deux séries de composants :

- Composants côté client
- Composants côté serveur

Les **composants côté client** comprennent le fichier de spécifications de l'extension, les classes Java et les fichiers .jar, les groupes de propriétés contenant les ressources localisables et les fichiers image et d'aide.

Les **composants côté serveur** sont les bibliothèques partagées et les DLL requis lorsqu'un noeud d'extension est exécuté.

### **Composants côté client**

Les composants côté client sont installés dans le dossier \ext\lib dans le répertoire d'installation IBM® SPSS® Modeler. Les composants côté client sont :

- Fichier de spécifications
- Classes Java et fichiers .jar
- Fichiers de propriétés
- Fichiers image
- Fichiers d'aide

#### **Dossier d'extension**

Chaque extension se situe dans son propre **dossier d'extension**, immédiatement sous \ext\lib.

La convention de dénomination suggérée pour le dossier d'extension est :

*providerTag.id*

où *providerTag* est l'identifiant du fournisseur provenant de l'élément `ExtensionDetails` du fichier de spécifications et *id* est l'identifiant de l'extension provenant du même élément.

Ainsi par exemple, si l'élément `ExtensionDetails` commence comme suit :

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

le nom du dossier de l'extension `myco.sorter` serait utilisé.

#### **Fichier de spécifications**

Le fichier de spécifications lui-même doit être nommé `extension.xml` et doit résider au niveau supérieur du sous-dossier de l'extension. Ainsi, dans l'exemple précédent, le chemin d'accès au fichier de spécifications serait le suivant dans le répertoire d'installation du IBM® SPSS® Modeler :

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

### **Classes Java et fichiers .jar**

Les extensions utilisant l'API Java côté client comprennent le code Java compilé. Ce code peut être laissé sous la forme d'un ensemble de fichiers .class ou être groupé en tant que fichier .jar.

Les fichiers Java .class sont situés en fonction du dossier d'extension de niveau supérieur. Par exemple, une classe qui met en oeuvre l'interface `ActionHandler` pourrait avoir le chemin suivant :

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

Dans ce cas, le fichier .class se trouverait à l'emplacement suivant sous le répertoire d'installation du SPSS Modeler :

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

Un fichier .jar peut se trouver n'importe où sous le dossier de l'extension. Vous spécifiez l'emplacement réel d'un fichier .jar au moyen de l'élément `JarFile` dans le fichier de spécifications. Par exemple, si une extension utilise un fichier .jar possédant le chemin suivant :

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

le fichier de spécification devrait inclure l'entrée suivante dans l'élément `Resources` :

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

[Pour plus d'informations, reportez-vous à la section Fichiers Jar dans le chapitre 4 sur p. 43.](#)

### **Fichiers de propriétés**

Les ressources localisées (par exemple, le texte à l'écran et les messages d'erreur ainsi que leurs traductions en langues étrangères) peuvent être conservées dans des fichiers possédant l'extension .properties, lesquels peuvent se trouver à tout endroit sous le dossier de l'extension. [Pour plus d'informations, reportez-vous à la section Fichiers de propriétés dans le chapitre 8 sur p. 209.](#)

### **Fichiers image et d'aide**

Les fichiers contenant les images graphiques pour l'affichage des icônes et ceux qui contiennent les systèmes d'aide peuvent se trouver n'importe où sous le dossier de l'extension. Il pourrait vous être utile de séparer les fichiers image et d'aide dans leurs propres sous-dossiers.

Vous déclarez l'emplacement d'un fichier image au moyen de l'attribut `imagePath` d'un élément `Icon` dans le fichier de spécifications. [Pour plus d'informations, reportez-vous à la section Icônes dans le chapitre 6 sur p. 134.](#)

De la même manière, vous déclarez l'emplacement d'un système d'aide à l'aide de l'attribut `path` d'un élément `HelpInfo` dans le fichier de spécifications. [Pour plus d'informations, reportez-vous à la section Définition de l'emplacement et du type du système d'aide dans le chapitre 7 sur p. 205.](#)

**Exemple**

La structure de fichier côté client basée sur ces composants pourrait se présenter ainsi :

```
\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
```

**Composants côté serveur**

Les bibliothèques partagées requises pour l'exécution doivent se trouver dans un sous-dossier du dossier `\ext\bin` dans le répertoire d'installation de IBM® SPSS® Modeler, par exemple :

```
installation_directory\ext\bin\myco.sorter\my_lib.dll
```

Notez que les bibliothèques partagées ne doivent pas se trouver directement dans le dossier `\ext\bin`.

En ce qui concerne les bibliothèques partagées que SPSS Modeler appelle directement pendant l'exécution, vous déclarez leur emplacement dans un élément `SharedLibrary` du fichier de spécifications. [Pour plus d'informations, reportez-vous à la section Bibliothèques partagées dans le chapitre 4 sur p. 44.](#)

La bibliothèque partagée principale peut nécessiter l'utilisation d'autres bibliothèques. Vous devriez également placer les éventuelles bibliothèques partagées dépendantes au même emplacement que la bibliothèque partagée principale pour faciliter la localisation des bibliothèques dépendantes par la bibliothèque principale.

**Exemple**

Voici un exemple possible d'une structure de fichiers côté serveur :

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

# Noeuds

## Présentation des noeuds

Lorsque vous créez une extension qui met en oeuvre un nouveau noeud, vous devez vous familiariser avec les caractéristiques des noeuds IBM® SPSS® Modeler. Cela vous aidera à les définir correctement dans le fichier de spécifications.




Les noeuds SPSS Modeler sont classés par noeuds source, d'exécution, de sortie et de modélisation, selon leur fonction. Dans CLEF, les noeuds sont classés d'une manière un peu différente. Le tableau ci-dessous indique le mappage entre les deux systèmes :





Table 2-1  
*Types de noeuds CLEF*

classification SPSS Modeler	Palette	noeud Typer CLEF
Noeuds source	Sources	Lecteur de données
Noeuds d'exécution	Ops sur lignes	Transformateur de données
	Ops sur champs	
Noeuds de sortie	Graphes	Créateur de document
	Sortie (noeud Rapport)	
	Exporter	Rédacteur de données
noeuds de modélisation	Modélisation	Créateur de modèle

Lorsque vous créez un nouveau noeud CLEF, vous le définissez comme étant l'un des types de noeuds CLEF. Le type de noeud que vous choisissez dépend de sa fonction principale.

Table 2-2  
*Types et fonctions du noeud*

noeud Typer CLEF	Description	Palette de noeuds correspondante	Forme de l'icône
Lecteur de données	Importe les données dans SPSS Modeler à partir d'un format différent.	Sources	
Transformateur de données	Prend des données de SPSS Modeler, les modifie d'une certaine manière et renvoie les données modifiées au flux SPSS Modeler.	Ops sur lignes, Ops sur champs	
Créateur de modèle	Génère des modèles à partir des données de SPSS Modeler.	Modélisation	

noeud Typier CLEF	Description	Palette de noeuds correspondante	Forme de l'icône
Créateur de document	Génère un graphique ou un rapport à partir des données de SPSS Modeler.	Graphes	
		Sortie (noeud Rapport)	
Applicateur de modèle (ou "nugget de modèle")	Définit un conteneur pour un modèle généré qui est renvoyé à l'espace de travail SPSS Modeler.	–	
Rédacteur de données	Exporte des données au format SPSS Modeler vers un format adapté à une utilisation dans une autre application.	Exporter	

Vous pouvez définir le type de noeud ainsi que d'autres attributs dans un élément Node dans le fichier de spécifications, par exemple :

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

L'attribut palette définit la palette dans la fenêtre principale SPSS Modeler par laquelle les utilisateurs peuvent accéder au noeud ; dans ce cas, la palette Ops sur lignes. Si vous omettez cet attribut, le noeud apparaît sur la palette Ops sur champs.

Divers exemples de noeuds sont fournis avec IBM® SPSS® Modeler. [Pour plus d'informations, reportez-vous à la section Présentation des exemples dans le chapitre 3 sur p. 32.](#)

## **Noeuds Lecteur de données**

Un noeud de lecture de données permet de lire les données d'une source externe dans un flux IBM® SPSS® Modeler. Les noeuds de la palette Sources de SPSS Modeler sont équivalents à des noeuds de lecture de données et se distinguent par une icône circulaire.

Dans les spécifications d'un noeud de lecture de données, vous indiquez les informations suivantes :

- La source de données (telle qu'un fichier ou une base de données)
- Tout pré-traitement des enregistrements (comme la gestion des espaces situés en début et en fin de chaîne ou le caractère à utiliser pour délimiter l'enregistrement)

- S'il faut filtrer tout champ d'enregistrement
- Le type de données (par exemple intervalle, ensemble, booléen) et le type de stockage (chaîne, entier, réel) à associer à chaque champ
- Si le modèle de données d'entrée est modifié

Le noeud de lecture de données peut inclure une logique pour lire les enregistrements de données source. Ceci peut aussi être effectué plus en aval à l'aide d'un noeud Typer dans SPSS Modeler.

Un exemple de noeud de lecture de données est fourni avec le client IBM® SPSS® Modeler.

Pour plus d'informations, reportez-vous à la section [Présentation des exemples dans le chapitre 3](#) sur p. 32.

## Noeuds de Transformation de données

Un noeud de Transformation de données prend les données d'un flux IBM® SPSS® Modeler, les modifie d'une certaine manière et renvoie les données modifiées vers le flux. Les noeuds des palettes Ops sur lignes et Ops sur champs de SPSS Modeler sont des noeuds de Transformation de données et se distinguent par une icône hexagonale.

Dans les spécifications d'un noeud de Transformation de données, vous indiquez les informations suivantes :

- Les enregistrements ou les champs transformés
- Comment les données doivent être modifiées

Un exemple de noeud de Transformation de données est fourni avec IBM® SPSS® Modeler. Pour plus d'informations, reportez-vous à la section [Présentation des exemples dans le chapitre 3](#) sur p. 32.

## Noeuds de création de modèles

Pour une présentation de la création de modèles dans IBM® SPSS® Modeler, consultez la rubrique "Introduction à la modélisation" dans le *Guide des applications SPSS Modeler 14.2*.

Les noeuds de création de modèles génèrent des objets apparaissant sur l'onglet Modèles ou Sorties du panneau du gestionnaire dans la fenêtre principale de SPSS Modeler.

Les noeuds de la palette Modélisation de SPSS Modeler sont des exemples de noeuds de création de modèles et se distinguent par une icône pentagonale.

Lors de son exécution, un noeud de création de modèles génère un **objet de sortie du modèle** (également appelé "nugget de modèle") sur l'onglet Modèles.

Figure 2-1

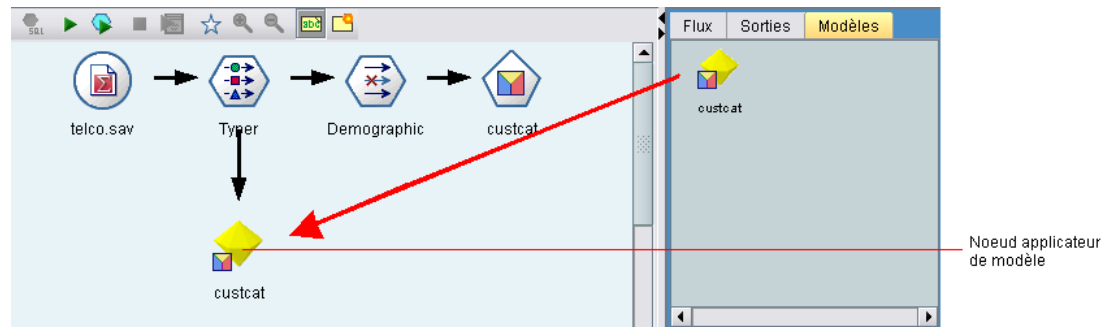
Le noeud de création de modèles génère un objet de sortie du modèle (nugget de modèle)



Quand un modèle généré est ajouté à l'espace de travail, il prend la forme d'un noeud applicateur de modèle.

Figure 2-2

Objet de sortie du modèle ajouté à l'espace de travail en tant que noeud applicateur de modèle



Dans les spécifications d'un noeud de création de modèles, vous spécifiez :

- Les détails de la création du modèle, tels que l'algorithme utilisé pour générer le modèle et les champs d'entrée et de sortie à utiliser pour évaluer les données avec le modèle
- Propriétés utilisées par le modèle
- Conteneurs utilisés pour stocker les objets de sortie
- Interface utilisateur de la boîte de dialogue du noeud
- Propriétés et fichiers utilisés lors de l'exécution du noeud
- Manière dont le modèle de données d'entrée est affecté par l'exécution du noeud
- Identificateur de l'objet de sortie du modèle et tout autre objet produit lors de l'exécution du noeud
- Identificateur du noeud applicateur du modèle (consultez la rubrique [Noeuds applicateurs de modèle](#) sur p. 13)

*Remarque* : Lors de la définition d'un noeud de création de modèles, vous incluez la définition de l'objet de sortie du modèle réel et du noeud applicateur de modèle ailleurs dans le même fichier de spécifications.

Un exemple de noeud de création de modèles est fourni avec IBM® SPSS® Modeler. [Pour plus d'informations, reportez-vous à la section Présentation des exemples dans le chapitre 3 sur p. 32.](#)

## **Noeuds de création de document**

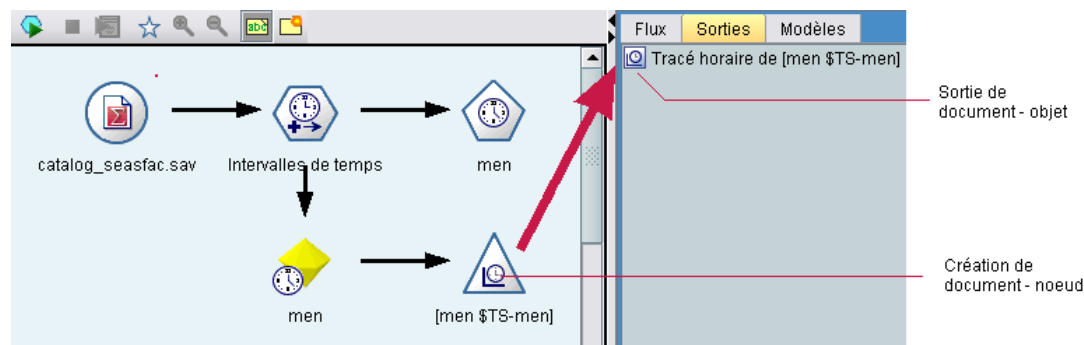
Les noeuds de création de document génèrent des objets apparaissant sur l'onglet Sorties du panneau du gestionnaire dans la fenêtre principale de IBM® SPSS® Modeler. Les noeuds de la palette Graphiques sont des exemples de noeuds de création de document et se distinguent par une icône triangulaire.

Lors de son exécution, un noeud de création de document génère un **objet de sortie de document** sur l'onglet Sorties du panneau du gestionnaire.



Figure 2-3

Noeud de création de document générant un objet de sortie de document



Contrairement à un objet de sortie du modèle, un objet de sortie de document ne peut pas être réintégré à l'espace de travail SPSS Modeler.

Dans les spécifications d'un noeud de création de document, vous spécifiez :

- Les détails de création du document, telles que l'onglet de la boîte de dialogue du noeud qui contiendra les commandes de création du document
- Propriétés utilisées par le document
- Conteneurs utilisés pour stocker les objets de sortie
- Interface utilisateur de la boîte de dialogue du noeud
- Propriétés et fichiers utilisés lors de l'exécution du noeud
- Identificateur de l'objet de sortie de document et tout autre objet produit lors de l'exécution du noeud

*Remarque* : Lors de la définition d'un noeud de création de document, vous incluez la définition de l'objet de sortie du document réel ailleurs dans le même fichier de spécifications.

## **Noeuds applicateurs de modèle**

Un noeud applicateur de modèle définit un conteneur pour un modèle généré à utiliser quand le modèle est ajouté à l'espace de travail IBM® SPSS® Modeler à partir de l'onglet Modèles du panneau du gestionnaire.

Dans les spécifications d'un noeud applicateur de modèle, vous indiquez les informations suivantes :

- Le conteneur du modèle (ou les conteneurs, si la sortie du modèle peut être produite dans plusieurs formats, par exemple texte et HTML)
- Les informations de l'interface utilisateur pour la boîte de dialogue affichées lorsque l'utilisateur parcourt le noeud applicateur sur l'onglet Modèles ou l'ouvre dans l'espace de travail.
- Le modèle de données de sortie

- Le traitement à effectuer lorsque le flux contenant le noeud est exécuté
- Les créateurs pour gérer les objets produits lorsque le flux contenant le noeud est exécuté

### ***Noeuds Rédacteur de données***

Un noeud Rédacteur de données exporte des données au format IBM® SPSS® Modeler vers un format adapté à une utilisation dans une autre application. Les noeuds de la palette Export de SPSS Modeler sont des noeuds Rédacteurs de données et se distinguent par une icône rectangulaire.

Dans les spécifications d'un noeud Rédacteur de données, vous spécifiez :

- Les informations du fichier ou de la base de données dans lequel le flux de données doit être écrit
- Facultativement, si le flux entier doit être publié afin de l'intégrer dans une application externe

### ***Menus, barres d'outils et palettes***

Les utilisateurs peuvent accéder à une extension à partir d'un menu IBM® SPSS® Modeler, de la barre d'outils ou d'une palette. Une extension peut mettre en œuvre un noeud ou effectuer une action spécifique.

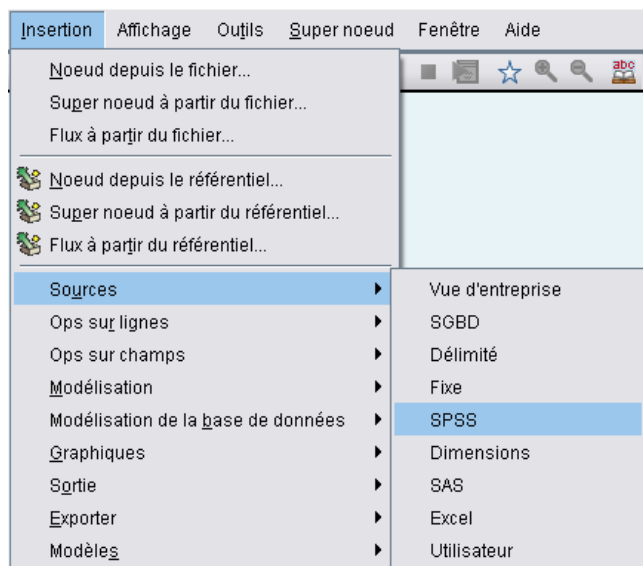
Une extension (noeud ou action) accessible à partir d'un menu explicitement spécifié peut aussi être rendue accessible à partir de la barre d'outils et vice versa.

Un noeud accessible à partir d'une palette est automatiquement accessible à partir d'un élément correspondant dans le menu Insertion.

### ***Menus et sous-menus***

Les utilisateurs peuvent accéder aux noeuds standard IBM® SPSS® Modeler à partir du menu Insertion. Chacun des éléments du dernier groupe de ce menu (à part les Modèles) dispose d'un sous-menu permettant d'accéder à un ensemble de noeuds liés.

**Figure 2-4**  
*Menu Insérer*



Ces éléments correspondent directement à des entrées sur les palettes de noeuds. L'ajout d'un noeud à une palette l'ajoute automatiquement au groupe correspondant dans le menu Insérer.

Si votre extension définit une action qui n'est pas accessible par un noeud, vous pouvez mettre l'extension à disposition en ajoutant un ou plusieurs des éléments suivants :

- Un nouvel élément dans un menu ou un sous-menu du système
- Un nouveau menu pour SPSS Modeler
- Un nouvel élément à la barre d'outils (consultez la rubrique [Barres d'outils sur p. 15](#))

Un nouveau menu ou élément de menu peut facultativement afficher l'icône associée à l'extension, comme par exemple sur certains éléments du menu Insérer.

Pour plus d'informations, reportez-vous à [Menus sur p. 135](#) et [Éléments de menu sur p. 137](#).

## ***Barres d'outils***

Si votre extension définit une action qui n'est pas accessible par un noeud, vous pouvez mettre l'extension à disposition en l'ajoutant à la barre d'outils principale de IBM® SPSS® Modeler.

**Figure 2-5**  
*Ajouter un élément à la barre d'outils principale*

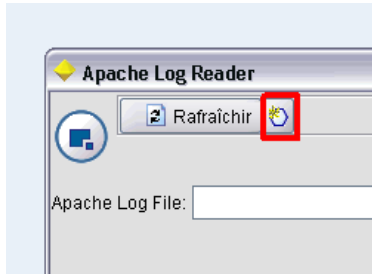


Dans ce cas, il est conseillé de cacher l'étiquette de l'action.

Vous pouvez également ajouter un élément à la barre d'outils d'une boîte de dialogue de noeud ou d'une fenêtre de sortie. Vous pouvez choisir d'afficher l'étiquette de l'élément ou de la cacher, comme dans cet exemple :

Figure 2-6

Ajouter un élément à une barre d'outils de boîte de dialogue



Pour plus d'informations, reportez-vous à la section [Eléments de barre d'outils](#) dans le chapitre 6 sur p. 138.

## Palettes et sous-palettes

Si votre extension définit un nouveau noeud, vous pouvez placer ce noeud dans n'importe quelle position sur l'une des palettes ou des sous-palettes IBM® SPSS® Modeler standard.

Figure 2-7

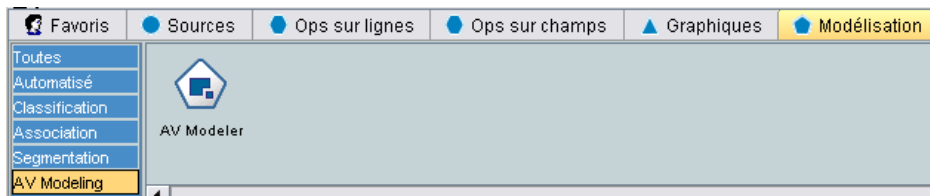
Nouveau noeud sur une palette standard



Vous pouvez ajouter une entrée à une sous-palette standard et rendre le noeud accessible à partir de là.

Figure 2-8

Nouveau noeud comme ajout personnalisé à une sous-palette standard



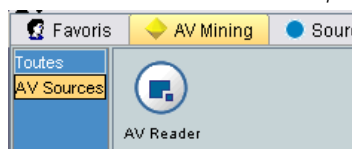
Vous pouvez définir une palette personnalisée et y placer le nouveau noeud.

Figure 2-9  
Nouveau noeud sur une palette personnalisée



Une palette personnalisée peut avoir des sous-palettes personnalisées.

Figure 2-10  
Nouveau noeud sur une sous-palette personnalisée d'une palette personnalisée



Pour plus d'informations, reportez-vous à [Noeud sur p. 61](#) et [Section User Interface \(Palettes\) sur p. 54](#).

## Conception d'icônes de noeud

Pour chaque nouveau noeud que vous créez dans CLEF, vous pouvez fournir une image centrale pour l'icône identifiant le noeud à l'écran.

*Remarque* : Vous n'avez pas besoin de fournir une image ; IBM® SPSS® Modeler fournit une image par défaut, qui est affichée si vous n'en spécifiez pas d'autre (ceci peut être utile lorsque vous commencez à développer un noeud).

Figure 2-11  
Image par défaut pour les icônes CLEF



Les icônes SPSS Modeler standard sont constituées de trois couches :

- Bordure
- Arrière-plan
- Image centrale




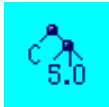



Pour un nouveau noeud, vous ne devez fournir que l'image centrale (ou **glyphe**). SPSS Modeler gère le traitement de la bordure et de l'arrière-plan. Le glyphe doit avoir un arrière-plan transparent afin de ne pas masquer la couche d'arrière-plan de l'icône. Dans cette section, les représentations des glyphes ont un arrière-plan coloré pour indiquer la transparence.

**Figure 2-12**  
*Couche de glyphe avec arrière-plan coloré indiquant la transparence*



Ceci montre comment une icône typique de modélisation SPSS Modeler est constituée.

**Table 2-3**  
*Composition des icônes de noeud et de modèle généré*




	<b>Icône de noeud</b>	<b>Icône de modèle généré</b>
Bordure		Aucun
Arrière-plan		
Glyphe		
Image affichée en tant que		

## **Bordures**

La fonction du noeud est indiquée par la forme de la bordure de l'icône. [Pour plus d'informations, reportez-vous à la section Présentation des noeuds sur p. 9.](#)

Si la mise en cache d'un noeud est activée, la forme de la bordure dispose d'un symbole de document miniature. Lorsque le cache d'un noeud est vide, une icône en forme de document blanc apparaît sur le noeud. Lorsque le cache est saturé, l'icône en forme de document devient entièrement verte.

Table 2-4  
Bordures du noeud et état de mise en cache





Etat de mise en cache	Exemple
Pas de mise en cache	
Mise en cache activée	
Cache plein	

Les différents symboles de bordures sont fournis par le système, et IBM® SPSS® Modeler gère le traitement nécessaire pour afficher le symbole adéquat au bon moment.

## Arrière-plans

Pour les icônes de noeuds autres que celles des modèles générés et des noeuds d'application de modèle, l'arrière-plan change de couleur pour indiquer l'état.

Table 2-5  
Arrière-plans de noeuds

Etat	Couleur	Exemple
Désactivé	Gris	
Activé	Bleu	
Erreur	Rouge	
Action effectuée dans une base de données	Violet	

Encore une fois, les images de l'arrière-plan sont fournies par le système, et IBM® SPSS® Modeler effectue le traitement nécessaire pour afficher le bon arrière-plan dans chaque situation.

## Exigences graphiques

Pour chaque nouveau noeud CLEF, créez les versions suivantes de l'image de la couche du glyphe :

- Grande taille (49 x 49 pixels) pour les noeuds de l'espace de travail de flux
- Petite taille (38 x 38 pixels) pour les noeuds du gestionnaire de palette au bas de l'écran

Si vous souhaitez afficher l'icône sur un menu ou une barre d'outil ou dans la barre de titre d'un navigateur ou d'une fenêtre de sortie, vous devrez aussi créer :

- Taille miniature (16 x 16 pixels)

Si le noeud génère un modèle, vous devrez aussi créer :

- Petite taille (38 x 38 pixels) avec la conception déplacée dans le coin inférieur gauche, pour une superposition sur l'icône de modèle généré (le nugget doré)

*Remarque* : Les images de taille supérieure à celles-ci sont tronquées lors de leur affichage dans IBM® SPSS® Modeler.

Pour plus d'informations, reportez-vous à la section [Icônes dans le chapitre 6 sur p. 134](#).

## Création d'images personnalisées

L'image que vous créez pour un noeud doit indiquer la fonction principale de ce noeud. Pour un public international, assurez-vous d'utiliser des images non spécifiques à un pays et non susceptibles d'être mal comprises par des utilisateurs d'autres pays.

Pour créer une image personnalisée à utiliser avec CLEF :

- ▶ A l'aide d'un logiciel graphique qui prend en charge la transparence, paramétrez l'espace de travail de dessin sur la taille appropriée et dessinez la version de l'image.
- ▶ Enregistrez chaque version (grande, petite, etc.) comme un fichier *.gif* séparé ayant les caractéristiques suivantes :
  - Arrière-plan transparent
  - Profondeur d'échantillonnage de 16 couleurs (4 bits) ou plus

La manière de rendre transparent l'arrière-plan de l'image dépend du logiciel graphique que vous utilisez. Par exemple, vous pourrez peut-être déterminer directement la transparence de la couleur de l'arrière-plan, ou vous devrez peut-être désigner une couleur de transparence puis "peindre" le fond de l'image avec cette couleur.

Pour les fichiers image, nous conseillons de respecter les conventions de dénomination de fichiers utilisées en interne par IBM® SPSS® Modeler :

Table 2-6  
Conventions de dénomination des fichiers image

Type d'image	Nom du fichier
Grand	lg_noeud.gif
Petit	sm_noeud.gif



Type d'image	Nom du fichier
Miniature	<i>noeud16.gif</i>
Modèle généré	<i>sm_gm_noeud.gif</i>

- Testez l'apparence de l'image en référençant les fichiers image dans le fichier de spécifications (consultez la rubrique [Ajouter les fichiers image à la Spécification du noeud](#)) et en ajoutant le nouveau noeud dans SPSS Modeler (consultez la rubrique [Test d'une extension CLEF sur p. 246](#)).

## **Ajouter les fichiers image à la Spécification du noeud**

Lorsque vous avez créé les fichiers image, copiez-les dans un dossier sur l'ordinateur avec lequel vous exécuterez IBM® SPSS® Modeler. Dans le fichier de spécifications, vous devrez indiquer un chemin d'image relatif à un dossier `\\ext\lib\provider.nodename` dans le répertoire d'installation de SPSS Modeler, c'est pourquoi il est conseillé de déployer les fichiers dans un dossier facile d'accès à partir de là. [Pour plus d'informations, reportez-vous à la section Icônes dans le chapitre 6 sur p. 134.](#)

Dans le fichier de spécifications, vous associez les fichiers graphiques de grandes et petites icônes à un noeud personnalisé à l'aide de l'élément `Icons` dans la section `UserInterface` de la spécification `Node`, par exemple :

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

Pour les noeuds de création de modèle ou de création de document, vous référez également la version miniature (16 x 16 pixel) dans la section `UserInterface` de la spécification `ModelOutput` (pour un noeud de création de modèle) ou la spécification `DocumentOutput` (pour un noeud de création de document), par exemple :

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

Pour les noeuds applicateurs de modèle, vous référez aussi la version du modèle généré dans la section `UserInterface` de la spécification `Node`, par exemple :

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

## **Conception des boîtes de dialogue**

Cette section décrit les caractéristiques des boîtes de dialogue des noeuds IBM® SPSS® Modeler standard vous aidant à concevoir des boîtes de dialogue cohérentes dans CLEF.

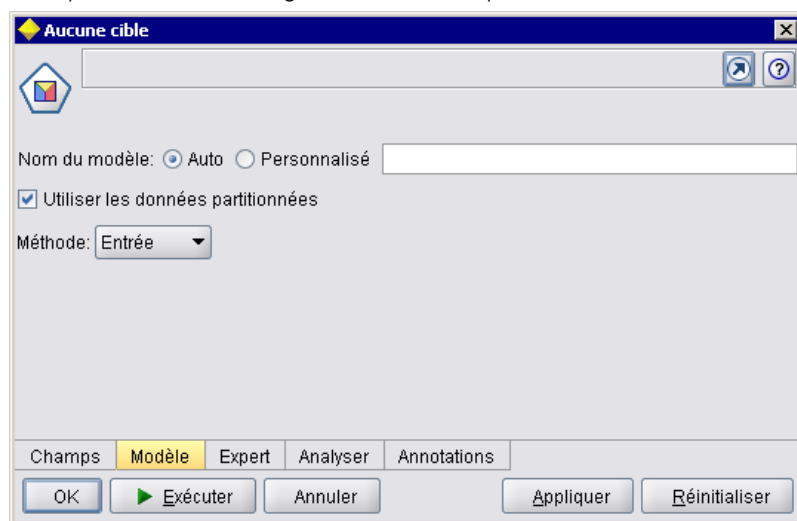
## A propos des boîtes de dialogue de noeud

Une boîte de dialogue de noeud fournit une interface qui permet à l'utilisateur final de modifier les paramètres d'exécution. L'aspect de la boîte de dialogue est très important. C'est l'endroit où le comportement du noeud est modifié. L'interface doit contenir toutes les informations nécessaires et doit être simple d'emploi.

Le comportement du noeud est modifié par l'utilisation de diverses **commandes** de la boîte de dialogue, qui sont des éléments de l'interface utilisateur avec lesquels un utilisateur peut interagir. Une boîte de dialogue peut inclure diverses commandes, telles que les cases d'option, les cases à cocher, les zones de texte et les menus. CLEF fournit de nombreuses commandes que vous pouvez concevoir dans vos boîtes de dialogue. [Pour plus d'informations, reportez-vous à la section Spécifications des contrôles de propriétés dans le chapitre 6 sur p. 151.](#)

Figure 2-13

Exemple de boîte de dialogue avec cases d'option, cases à cocher, zones combinées et onglets



Le type de paramètre modifié par une commande détermine le choix de la commande qui apparaît dans la boîte de dialogue. Certains types fournissent d'autres commandes. Vous pouvez regrouper des options sur de nouveaux onglets à l'aide des éléments Onglet dans le fichier de spécifications. [Pour plus d'informations, reportez-vous à la section Zone d'onglet sur p. 27.](#)

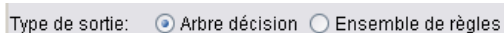
*Remarque* : Vous pouvez tester l'aspect de l'interface utilisateur pour une extension, même si vous n'avez pas spécifié le traitement à effectuer par l'extension. [Pour plus d'informations, reportez-vous à la section Test des extensions CLEF dans le chapitre 10 sur p. 246.](#)

## Instructions de conception des boîtes de dialogue

Lorsque vous définissez les commandes pour une boîte de dialogue, prenez en compte les conseils suivants :

- Tenez compte du texte à utiliser lorsque la commande dispose d'une étiquette. Ce texte doit être suffisamment concis tout en fournissant les informations correctes. S'il s'agit d'une conception pour un marché international, gardez à l'esprit que la longueur du texte traduit peut différer significativement de celle du texte d'origine.
- Utilisez la commande adaptée au paramètre. Par exemple, une case à cocher n'est pas toujours le meilleur choix pour un paramètre qui n'accepte que deux valeurs. La boîte de dialogue du noeud IBM® SPSS® Modeler C5.0 utilise des cases d'option pour permettre aux utilisateurs de sélectionner le type de sortie, qui a l'une des deux valeurs Arbre décision ou Ensemble de règles.

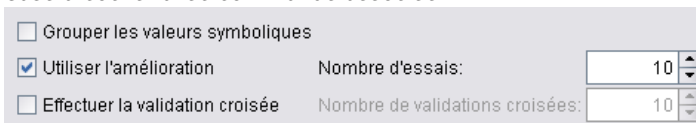
Figure 2-14  
Choisir entre deux valeurs



Ce paramètre peut être représenté par une case à cocher intitulée Arbre décision. Lorsque cette option est sélectionnée, le type de sortie est un arbre décision. Lorsqu'elle est désélectionnée, la sortie est un ensemble de règles. Bien que le résultat soit le même, l'utilisation de cases d'option facilite la compréhension par l'utilisateur des options.

- Les commandes de noms de fichiers sont généralement placées en haut de la boîte de dialogue.
- Les commandes qui forment les points centraux du noeud sont placées dans la partie supérieure de la boîte de dialogue. Par exemple, les noeuds de graphiques affichent les champs provenant des données. La sélection de ces champs est la fonction principale de la boîte de dialogue. Par conséquent, les paramètres de champ sont placés en haut.
- Les cases à cocher ou les cases d'option permettent souvent à l'utilisateur de sélectionner une option qui nécessite davantage d'informations. Par exemple, si vous sélectionnez Utiliser l'amélioration dans la boîte de dialogue C5.0, l'analyse doit comprendre un nombre indiquant le Nombre d'essais.

Figure 2-15  
Case à cocher avec commande associée



Les informations supplémentaires sont toujours placées après la sélection de l'option, à droite ou juste en dessous.

Les boîtes de dialogue CLEF utilisent l'édition commit de SPSS Modeler de la même manière que les boîtes de dialogue SPSS Modeler standard : les valeurs affichées dans les boîtes de dialogue ne sont pas copiées vers le noeud avant que l'utilisateur ne clique sur OK, Appliquer, ou dans le cas des noeuds terminaux, sur Exécuter. De la même façon, les informations affichées par la boîte de dialogue ne sont mises à jour (par exemple, lorsque les champs d'entrée du noeud ont été modifiés suite à une opération en amont sur le noeud actuel) que si l'utilisateur annule et affiche à nouveau la boîte de dialogue ou clique sur le bouton Rafraîchir.

## Composants de la boîte de dialogue

Les boîtes de dialogue ont les composants suivants :

- Barre de titre
- Zone d'icône
- Zone de barre d'outils et de menu comprenant :
  - Fichier, Générer, Affichage, Aperçu, Actualiser et d'autres boutons (selon le noeud)
  - Bouton Maximiser/taille normale
  - Bouton Aide
- Zone d'état
- Zone de panneau
- Zone d'onglet
- Zone de bouton

Chaque noeud personnalisé nécessite une boîte de dialogue qui s'affiche lorsque l'utilisateur ouvre le noeud. Si votre fichier de spécifications comprend un élément **Node** contenant une section **UserInterface** avec un élément **Tabs**, vous verrez tous les composants de la boîte de dialogue répertoriés au-dessus lorsque vous ouvrez le noeud. En fonction du type de noeud, le contenu minimum des zones d'onglet et de bouton est comme suit :

Table 2-7

Contenu minimum des zones d'onglet et de bouton pour les différents types de noeuds

Type de noeud	Onglets	Boutons
Lecteur de données	Annotations (avec bouton Actualiser dans la zone de la barre d'outils)	OK, Annuler, Appliquer, Réinitialiser
Transformateur de données	annotations	OK, Annuler, Appliquer, Réinitialiser
Rédacteur de données	Publier, Annotations	OK, Annuler, Exécuter, Appliquer, Réinitialiser
Créateur de modèle	annotations	OK, Annuler, Exécuter, Appliquer, Réinitialiser

Type de nœud	Onglets	Boutons
Créateur de document	annotations	OK, Annuler, Exécuter, Appliquer, Réinitialiser
Applicateur de modèle	Résumé, Annotations	OK, Annuler, Appliquer, Réinitialiser

Les boîtes de dialogue de noeuds sont initialement placées de manière à ce que lorsque l'utilisateur ouvre le noeud, l'icône du noeud est superposée sur le noeud qu'elle représente. L'utilisateur peut déplacer la boîte de dialogue, mais la nouvelle position n'est pas conservée lorsque le noeud s'ouvre la fois suivante. Si l'utilisateur a déplacé la boîte de dialogue et si celle-ci a ensuite été partiellement ou complètement cachée par une autre boîte de dialogue, double-cliquez sur le noeud d'origine dans l'espace de travail pour ramener la première boîte de dialogue au premier plan. La boîte de dialogue est non modale (la même entrée de l'utilisateur entraîne toujours la même action) et redimensionnable.

Tous les champs modifiables de la boîte de dialogue prennent en charge les raccourcis clavier suivants :

Raccourci	Effet
Ctrl+C	Copier
Ctrl+V	Coller
Ctrl+X	Couper

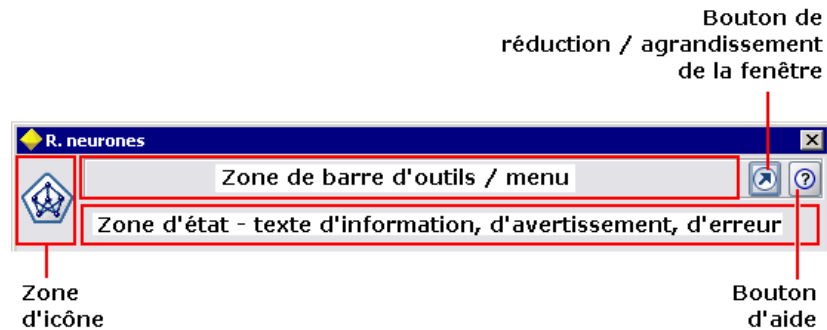
### **Barre de titre**

La barre de titre de la boîte de dialogue d'un noeud comprend une version miniature de l'icône nugget doré de IBM® SPSS® Modeler, suivie du nom du modèle. Le texte est tiré du paramètre des commandes de nom du modèle. Le bouton Fermer (X) est également fourni par défaut dans le coin supérieur droit.

### **Zone d'icône**

L'icône Noeud s'affiche dans la zone d'icône en haut à gauche de la boîte de dialogue.

Figure 2-16  
Haut de la boîte de dialogue du noeud



C'est la petite version (38 x 38 pixels) de l'icône également utilisée sur la palette de noeuds en bas de la fenêtre principale, et non la version plus grande qui apparaît dans l'espace de travail.

*Remarque* : L'icône miniature du nugget doré à l'extrémité gauche de la barre de titre est codée en dur dans toutes les boîtes de dialogue des noeuds.

### **Zone de barre d'outils et de menu**

La zone supérieure de la boîte de dialogue est réservée comme zone de barre d'outils et de menu.

Les boîtes de dialogue des noeuds de transformation de données et de lecture de données ont un bouton Aperçu dans cette zone qui présente un échantillon des données d'entrée.

Les boîtes de dialogue des noeuds de lecture de données ont également un bouton Actualiser qui met à jour les informations affichées par le noeud (par exemple, lorsque les champs d'entrée du noeud ont été modifiés).

Les noeuds applicateurs de modèle possèdent les boutons de menu Fichier, Générer et Affichage qui permettent aux utilisateurs d'effectuer diverses opérations telles que l'exportation du modèle ou la génération de nouveaux noeuds. Les noeuds applicateurs de modèle possèdent également le bouton Aperçu, qui présente dans ce cas un échantillon des données d'entrée avec les colonnes supplémentaires créées lorsque le noeud est appliqué.

Le côté droit de cette zone contient deux boutons dans chaque boîte de dialogue de noeud :

- Bouton Maximiser/taille normale
- Bouton Aide

### **Bouton Maximiser/taille normale**

Ce bouton redimensionne la boîte de dialogue en taille plein écran. Une utilisation ultérieure réduit la boîte de dialogue à la taille qu'elle avait avant la maximisation.

### ***Bouton Aide***

Ce bouton ouvre l'aide contextuelle du noeud. Pour une boîte de dialogue à onglets ou une fenêtre de sortie, l'aide pour cet onglet s'affiche. Vous pouvez aussi utiliser la touche F1 pour accéder à l'aide.

### ***Zone d'état***

Le reste de la zone au sommet de la boîte de dialogue est réservé à l'affichage d'informations, d'avertissements ou de texte d'erreur. Les noeuds source affichent ici le chemin et le nom complets du fichier de données source. Chaque noeud peut avoir d'autres informations spécifiques à afficher dans cette zone. Tout texte spécifié pour cette zone doit être limité à deux lignes.

### ***Zone de panneau***

C'est la zone principale de la boîte de dialogue, qui contient toutes les commandes et les zones d'affichage du noeud. Chaque onglet dispose d'une zone de panneau différente. Chaque panneau peut être de l'un des types suivants :

- Navigateur de texte
- Objet d'extension
- Properties

Vous pouvez également spécifier des sous-panneaux, qui sont des boîtes de dialogue séparées s'ouvrant dans une nouvelle fenêtre et appelées à partir de boutons d'action sur le panneau.

[Pour plus d'informations, reportez-vous à la section Spécifications de panneau dans le chapitre 6 sur p. 143.](#)

### ***Zone d'onglet***

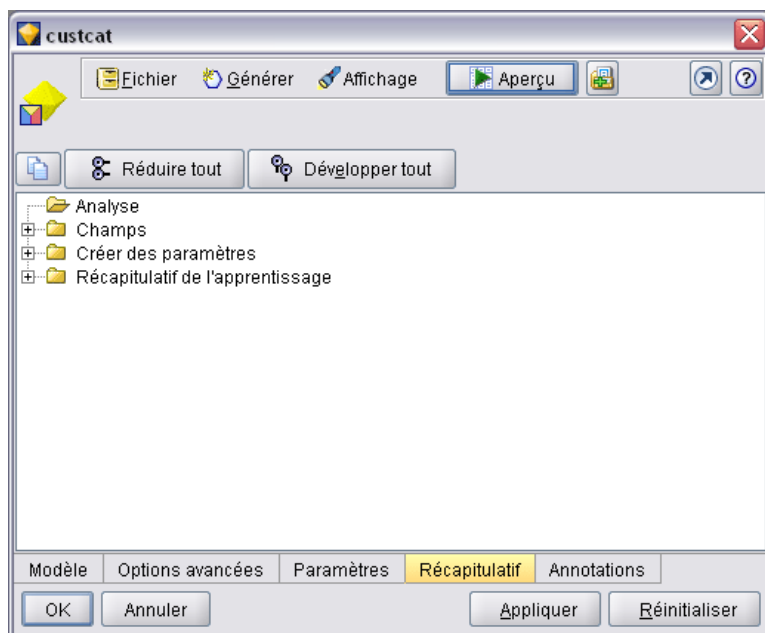
Les boîtes de dialogue de noeuds ont les onglets suivants :

- Un ou plusieurs onglets spécifiques au noeud fournis par l'utilisateur
- Un onglet Résumé (objets de sortie de modèle et noeuds d'application de modèle uniquement)
- Un onglet Annotations

Les onglets spécifiques au noeud sont définis dans la section **Onglets** du fichier de spécifications de CLEF. [Pour plus d'informations, reportez-vous à la section Onglets dans le chapitre 6 sur p. 140.](#)

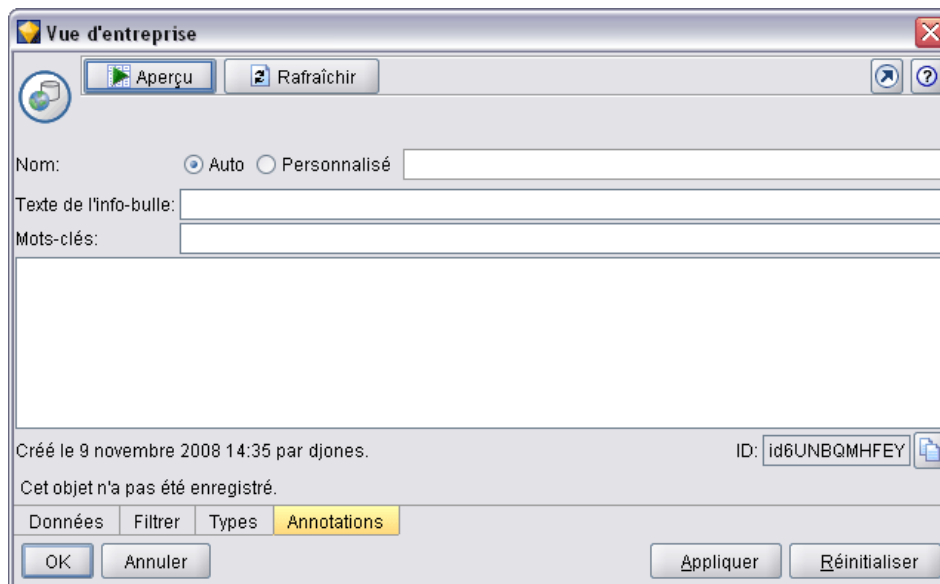
Les boîtes de dialogue des objets de sortie de modèle et les noeuds d'application de modèle ont un onglet Résumé fourni par le modèle. Il affiche des informations récapitulatives à propos d'un modèle généré, y compris les champs, les paramètres de création et le processus d'estimation du modèle utilisés. Les résultats sont présentés dans un arbre que vous pouvez développer ou réduire en cliquant sur des éléments précis.

Figure 2-17  
Onglet Récapitulatif



L'onglet Annotations est fourni par le système pour toutes les boîtes de dialogue de noeuds et permet à l'utilisateur de spécifier des informations à propos du noeud. Ceci comprend le nom du noeud, le texte d'info-bulle et un champ de commentaire plus long.

Figure 2-18  
Onglet Annotations





**Nom.** Le nom du noeud par défaut est indiqué dans l'attribut Label de l'élément Node dans le fichier de spécifications (consultez la rubrique [Noeud sur p. 61](#)). L'utilisateur peut renommer le noeud en sélectionnant Personnalisé, en entrant un nom dans le champ d'édition Personnalisé et en cliquant sur Appliquer ou sur OK. Le nouveau nom est préservé à travers les sessions, mais le nom par défaut peut être restauré en sélectionnant Auto. Un nom personnalisé spécifié sur l'onglet Annotations remplace un nom personnalisé spécifié sur tout autre onglet de la boîte de dialogue.

**Texte de l'info-bulle.** Le texte indiqué ici est affiché comme info-bulle pour le noeud dans l'espace de travail. Si aucun texte d'info-bulle n'est spécifié ici, aucune info-bulle n'est affichée lorsque l'utilisateur passe le curseur sur le noeud.

**Mots-clés.** L'utilisateur peut spécifier des mots-clés à utiliser dans les rapports de projet et lors de la recherche ou du suivi des objets stockés dans le IBM® SPSS® Collaboration and Deployment Services Repository.

**Panneau des commentaires.** Cette zone permet à l'utilisateur d'entrer un texte de commentaire.

**Création et enregistrement d'informations.** C'est une zone de texte non modifiable indiquant les informations de création ainsi que le nom et la date/l'heure auxquels un fichier a été enregistré (le format date/heure dépend des paramètres régionaux). Si aucun enregistrement n'a été effectué, ce champ indiquera "Cet élément n'a pas été enregistré."

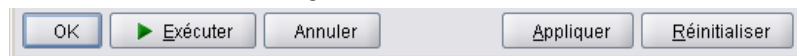
## Zone de bouton

En bas de chaque boîte de dialogue, les boutons Appliquer, Réinitialiser, OK et Annuler sont affichés. Si le noeud est un noeud terminal (un noeud exécutable qui traite les données de flux), un bouton Exécuter est également présent.

Figure 2-19  
Boutons de la boîte de dialogue



Figure 2-20  
Boutons de la boîte de dialogue du noeud terminal



**OK.** Applique tous les paramètres et ferme la boîte de dialogue. La première fois que la boîte de dialogue est ouverte à partir du noeud, ce bouton est activé (indiqué par un rectangle bleu autour du bouton) et le fait d'appuyer sur la touche Entrée exécute aussi l'opération OK.

**Annuler.** Ferme la boîte de dialogue et laisse les paramètres tels qu'ils étaient avant l'ouverture de la boîte de dialogue ou depuis la dernière opération Appliquer. Le fait d'appuyer sur la touche Echap lorsque l'ensemble de la boîte de dialogue est activée exécute aussi l'opération Annuler.

**Exécuter.** Applique tous les paramètres, ferme la boîte de dialogue et exécute le noeud terminal.

**Appliquer.** Enregistre les paramètres de la boîte de dialogue afin que les opérations en aval puissent les utiliser.

**Réinitialiser.** Réinitialise l'ensemble de la boîte de dialogue aux valeurs qu'elle contenait à l'ouverture de la boîte de dialogue ou depuis la dernière opération Appliquer.

## Conception des fenêtres de sortie

Cette section décrit les caractéristiques des fenêtres de sortie IBM® SPSS® Modeler standard vous aidant à concevoir des fenêtres de sortie cohérentes dans CLEF.

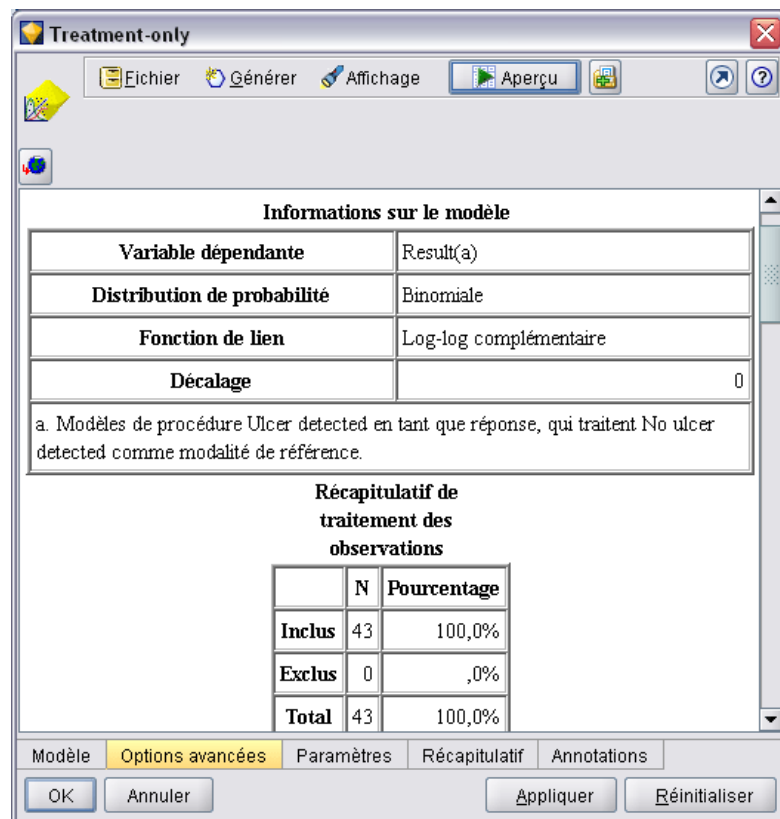
Les fenêtres de sortie vous permettent d'afficher la sortie de :

- Un modèle ; par exemple, évaluer (appliquer un modèle à) un ensemble de données
- Un document ; par exemple, un graphique ou un rapport

Pour plus d'informations, reportez-vous à la section A propos des interfaces utilisateur dans le chapitre 6 sur p. 128.

Une fenêtre de sortie typique a l'aspect suivant :

Figure 2-21  
Fenêtre de sortie de modèle



Les fenêtres de sortie sont semblables aux boîtes de dialogue de noeuds mais présentent les différences suivantes :

- La barre de titre dispose d'une icône miniature spécifique au noeud à la place de l'icône du nugget doré générique.
- L'icône principale du noeud est omise

- Dans la zone de la barre d'outils et du menu, le bouton Maximiser/taille normale est omis (il peut être remplacé par un bouton Fermer et supprimer dans une fenêtre de sortie de document), même si la fenêtre est toujours redimensionnable à l'aide de la souris.
- La zone d'état est omise
- Les onglets sont généralement :
  - un onglet Modèle (pour les fenêtres de sortie de modèle) pour afficher les données d'importance des valeurs prédites, si cette option est sélectionnée dans le noeud du modèle.
  - un onglet unique pour la sortie
  - un onglet Récapitulatif (pour les fenêtres de sortie de modèle) pour afficher des détails du récapitulatif concernant le modèle
  - un onglet Annotation (les valeurs d'annotations sont tirées du noeud qui a généré la sortie)
- La zone des boutons contient les boutons OK, Annuler, Appliquer et Réinitialiser

CLEF fournit des fenêtres de sortie de modèle et de sortie de document par défaut semblables à celle illustrée ci-dessus. Celles-ci sont normalement affichées lorsque vous utilisez un élément `ModelOutput` ou `DocumentOutput` dans le fichier de spécifications. [Pour plus d'informations, reportez-vous à la section Identificateur d'objet dans le chapitre 4 sur p. 60.](#)

Vous pouvez également spécifier un élément `ModelOutput` ou `DocumentOutput` de manière à remplacer complètement la fenêtre de sortie par défaut par une fenêtre personnalisée que vous avez conçue. [Pour plus d'informations, reportez-vous à la section Fenêtres de sortie personnalisées dans le chapitre 6 sur p. 201.](#)

# CLEF Exemples

## Présentation des exemples

Afin de vous familiariser avec CLEF, une installation IBM® SPSS® Modeler fournit des exemples de noeuds accompagnés de leur code source complet. Il s'agit de noeuds de base ayant une fonctionnalité limitée, qui visent à vous aider à mieux comprendre le fonctionnement de CLEF et à l'utiliser. Vous pouvez tester ces noeuds maintenant ou quand bon vous semble.

Les exemples sont les suivants :

- Noeud de lecture de données (Apache Log Reader)
- Noeud de transformation de données (URL Parser)
- Noeud de création de document (Web Status Report)
- Noeud de création de modèle (Interaction)

Vous devez activer les exemples pour pouvoir les utiliser.

## Activation des exemples

Les exemples se trouvent dans le répertoire *Demos* dans un format compressé ; ils sont installés en même temps que IBM® SPSS® Modeler. Pour activer les exemples, vous devez extraire les fichiers à leur emplacement approprié, comme suit.

Sur l'ordinateur sur lequel vous avez installé SPSS Modeler :

- ▶ Quittez le programme SPSS Modeler s'il est en cours d'exécution.
- ▶ Recherchez le fichier *clef\_examples\_ext\_lib.zip* dans le dossier *Demos* de l'installation SPSS Modeler.
- ▶ Extrayez le contenu du fichier *clef\_examples\_ext\_lib.zip* dans le dossier *\ext\lib* du répertoire d'installation SPSS Modeler.

Pour une installation limitée à SPSS Modeler :

- ▶ Extrayez le contenu du fichier *clef\_examples\_ext\_bin.zip* dans le dossier *\ext\bin* du répertoire d'installation SPSS Modeler.

Pour une configuration comportant des installations SPSS Modeler et IBM® SPSS® Modeler Server distinctes :

- ▶ Extrayez le contenu du fichier *clef\_examples\_ext\_bin.zip* dans le dossier *\ext\bin* des répertoires d'installation de SPSS Modeler et de SPSS Modeler Server.

- ▶ (Serveurs UNIX et Windows 64 bits uniquement) En plus de l'étape précédente, utilisez le fichier makefile fourni dans *clef\_examples\_ext\_bin.zip* pour compiler le code source des exemples. [Pour plus d'informations, reportez-vous à la section Analyse du code source sur p. 36.](#)

Dans tous les cas :

- ▶ Démarrez SPSS Modeler et vérifiez que les noeuds suivants sont visibles dans la palette de noeuds :

Onglet Palette	Noeud
Sources	Apache Log Reader
Ops sur champs	URL Parser
Modélisation	Interaction
Sortie	Web Status Report

## Noeud de lecture de données (Apache Log Reader)

Le noeud de lecture de données est un noeud source qui lit les données du fichier journal des accès du serveur Web HTTP Apache. Le journal des accès contient les informations de toutes les demandes que traite le serveur Web. Les enregistrements de journal ont le format Combined Log —par exemple :

```
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)"
```

Vous pouvez utiliser l'exemple de noeud pour convertir les enregistrements de journal en tableau pour en faciliter la lecture.

Pour utiliser le noeud Apache Log Reader :

- ▶ Si vous n'avez pas encore activé les exemples CLEF, faites-le maintenant. [Pour plus d'informations, reportez-vous à la section Activation des exemples sur p. 32.](#)
- ▶ Ouvrir IBM® SPSS® Modeler.
- ▶ Dans l'onglet Sources de la palette des noeuds, sélectionnez Apache Log Reader et ajoutez le noeud à l'espace de travail.
- ▶ Modifiez le noeud. Dans le champ Fichier journal Apache de l'onglet Option, saisissez :

```
demos_folder\combined_log_format.txt
```

où *demos\_folder* est l'emplacement du dossier *Demos* dans le répertoire d'installation IBM® SPSS® Modeler (n'utilisez pas le format *\$CLEO\_DEMOS*).

- ▶ Cliquez sur OK.
- ▶ Ajoutez un noeud Typer au flux.
- ▶ Modifiez le noeud Typer. Cliquez sur Lire les valeurs pour lire les données, puis cliquez sur OK.

- ▶ Liez un noeud Table au noeud Typer et exécutez le flux. Le contenu du fichier journal s'affiche dans un tableau.
- ▶ Enregistrez le flux pour pouvoir l'utiliser dans les deux exemples suivants.

## **Noeud de transformation de données (URL Parser)**

L'exemple de noeud de transformation de données traite également les données retournées par l'exemple précédent. Vous sélectionnez un champ d'ID (qui doit contenir une valeur unique pour chaque ligne) et un champ d'entrée contenant une URL. Le noeud génère une sortie constituée de ces deux champs et des données d'URL analysées, dans des champs générés distincts. Par exemple, si un enregistrement d'URL contient une chaîne de requête, telle que :

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

l'enregistrement est analysé comme suit :

Champ généré	Index
<code>URLfield_server</code>	<code>http://www.dummydomain.co.uk</code>
<code>URLfield_path</code>	<code>/resource.php</code>
<code>URLfield_field</code>	<code>res_id</code>
<code>URLfield_value</code>	<code>89</code>

Pour utiliser le noeud URL Parser :

- ▶ Si le flux de l'exemple précédent est fermé, ouvrez-le. Le flux contient les noeuds Apache Log Reader et Typer.
- ▶ Dans l'onglet Ops sur champs de la palette des noeuds, liez un noeud URL au noeud Typer.
- ▶ Modifiez le noeud URL Parser. Dans la liste déroulante Champ d'ID, sélectionnez ReturnedContentSize. Dans la liste déroulante Champ d'URL, sélectionnez ReferralURL. Cliquez sur OK.
- ▶ Liez un noeud Table au noeud URL Parser et exécutez le flux. Les champs ReturnedContentSize et ReferralURL s'affichent avec ReferralURL analysé dans quatre champs générés distincts : ReferralURL\_server, ReferralURL\_path, ReferralURL\_field et ReferralURL\_value.

## **Noeud de création de document (Web Status Report)**

L'exemple de noeud de création de document lit les données envoyées par le journal du serveur Web et génère un rapport dans un fichier de format HTML. Le rapport est constitué d'un tableau qui affiche les pourcentages d'enregistrements de journal retournant divers codes d'état HTTP (200, 302, 404, etc.).

Pour utiliser le noeud Web Status Report :

- ▶ Si le flux du premier exemple est fermé, ouvrez-le. Il s'agit du flux qui contient les noeuds Apache Log Reader et Typer. Si le flux contient un noeud URL Parser provenant du second exemple, ce noeud est ignoré dans cet exemple.

- ▶ Dans l'onglet Sortie de la palette des noeuds, liez un noeud Web Status Report au noeud Typer.
- ▶ Modifiez le noeud Web Status Report. Dans la liste déroulante Champ de code d'état, sélectionnez StatusCode. Cliquez sur Exécuter. Une fenêtre de sortie s'affiche avec le contenu du rapport.

## **Noeud de création de modèle (Interaction)**

L'exemple de noeud de création de modèle fonctionne indépendamment des autres exemples ; il permet de créer un modèle simple de manière normale (non interactive) ou d'interagir avec le modèle avant sa création. Le modèle tente de déterminer le score d'attrition de la clientèle d'un opérateur téléphonique.

Pour utiliser le noeud Interaction :

- ▶ Si vous n'avez pas encore activé les exemples CLEF, faites-le maintenant. [Pour plus d'informations, reportez-vous à la section Activation des exemples sur p. 32.](#)
- ▶ Créez un nouveau flux dans IBM® SPSS® Modeler.
- ▶ Ajoutez un noeud source Statistics qui importe le fichier *telco.sav* du répertoire *Demos*.
- ▶ Dans l'onglet Types, cliquez sur Lire les valeurs et cliquez sur OK dans la boîte de message pour confirmer.
- ▶ Définissez le rôle du champ d'attrition (dernier champ de la liste) sur Cible et cliquez sur OK.
- ▶ Dans l'onglet Modélisation de la palette des noeuds, liez un noeud Interaction au noeud source.

Pour tester la création de modèle (non interactive) standard :

- ▶ Exécutez le flux pour créer un nugget de modèle dans le flux et dans la palette Modèles en haut à droite de l'écran.
- ▶ Liez un noeud Table au nugget de modèle.
- ▶ Exécutez le noeud Table. Faites défiler vers la droite la fenêtre de sortie du tableau pour afficher les prévisions d'attrition. Le champ **\$I-churn** contient les prévisions, alors que **\$I-churn** indique les valeurs de confiance (0.0 à 0.1) des prévisions.

Pour tester la création de modèle interactive :

- ▶ Dans l'onglet Modèle de la boîte de dialogue de création de modèle Interaction, sélectionnez Démarrer une session interactive.
- ▶ Cliquez sur Exécuter pour afficher la boîte de dialogue Tester l'interaction.
- ▶ Dans cette boîte de dialogue, cliquez sur Démarrer la tâche de création pour afficher l'avancement de la création de modèle.
- ▶ Une fois la création de modèle terminée, sélectionnez la ligne qui a été ajoutée dans le tableau Tâches de création dans la boîte de dialogue.

- Dans la zone de barre d'outils dans la partie supérieure de la boîte de dialogue, cliquez sur le bouton comportant un losange jaune. Ceci génère l'objet de sortie du modèle (appelé `model_1`) dans la palette Modèles en haut à droite de l'écran.

Le modèle généré de manière interactive est identique au premier modèle, mais son nom est différent. Si vous répétez les opérations depuis l'étape Démarrer une tâche de création, vous générez un nouveau modèle, `model_2`, et ainsi de suite.

## Analyse des fichiers de spécifications

Une méthode efficace pour comprendre le fonctionnement de CLEF consiste à analyser les fichiers de spécifications des exemples fournis. Ces fichiers se trouvent dans :

```
install_dir\ext\lib\  
extension_folder\extension.xml
```

où `install_dir` est le répertoire d'installation de IBM® SPSS® Modeler et `extension_folder` correspond à l'un des éléments suivants :

- `spss.apachelogreader`
- `spss.interaction`
- `spss.urlparser`
- `spss.webstatusreport`

D'autres dossiers d'extension peuvent figurer sous `ext\lib` ; ces dossiers sont associés aux noeuds IBM® SPSS® Modeler fournis par le système produits par CLEF. Ces noeuds apparaissent ou non dans l'installation en fonction des modules SPSS Modeler pour lesquels vous disposez d'une licence. Il peut être également utile de consulter leurs fichiers de spécifications, mais **ne modifiez ces fichiers en aucun cas** afin de maintenir le bon fonctionnement de ces noeuds et d'éviter d'avoir à réinstaller le produit SPSS Modeler correspondant. IBM Corp. ne prend pas en charge les modifications des fichiers fournis par le système

## Analyse du code source

Le code source complet des exemples de noeuds est également fourni pour pouvoir vous y référer. Tous les exemples de noeuds utilisent des bibliothèques côté serveur C++, seul le noeud Interaction utilise en plus des classes Java client.

Les fichiers de code source sont extraits automatiquement lorsque vous activez les exemples ; ils sont installés comme suit :

Emplacement	Index
... \ext\lib\spss.interaction\src	Code source Java des fichiers .class dans le fichier ui.jar dans le dossier parent
... \ext\bin\spss.apachelogreader\src ... \ext\bin\spss.interaction\src ... \ext\bin\spss.urlparser\src ... \ext\bin\spss.webstatusreport\src	Fichiers source C++ et fichiers de projet pour les DLL dans le dossier parent



---

## ***Suppression des exemples***

Si vous voulez retirer les noeuds exemples de IBM® SPSS® Modeler, vous pouvez les supprimer comme suit :

- ▶ Quittez IBM® SPSS® Modeler.
- ▶ Supprimez les dossiers des exemples des répertoires *\ext\bin* et *\ext\lib* de l'installation SPSS Modeler. Ne supprimez pas les dossiers standard SPSS Modeler par inadvertance afin de ne pas avoir à réinstaller le produit SPSS Modeler correspondant. Vous devez supprimer les dossiers suivants :
  - *spss.apachelogreader*
  - *spss.urlparser*
  - *spss.webstatusreport*
  - *spss.interaction*

Les modifications sont appliquées lors du prochain redémarrage de SPSS Modeler.

# Fichier de spécifications

## Présentation des fichiers de spécifications

Chaque extension CLEF doit comprendre un fichier XML dans lequel vous définissez toutes les caractéristiques de l'extension. Ce fichier est le **fichier de spécifications** et est toujours nommé `extension.xml`. Un fichier de spécifications comprend les sections suivantes :

- une **déclaration XML**. Déclaration facultative de la version XML et d'informations supplémentaires.
- un **Elément Extension**. Partie principale du fichier ; il contient toutes les sections suivantes.
- **Section Extension Details**. Décrit les informations de base de l'extension.
- **Section Resources**. Décrit les ressources externes nécessaires au fonctionnement de l'extension, comme les groupes de ressources, les fichiers JAR et les bibliothèques partagées.
- **Section Common Objects**. (facultatif) Définit les éléments pouvant être utilisés ou référencés par d'autres objets dans l'extension, tels que les modèles, les documents et les types de propriétés.
- **Section User Interface (Palettes)**. (facultatif) Définit une palette ou sous-palette personnalisée dans laquelle doit apparaître un noeud.
- **Section Object Definition**. Identifie les objets définis par l'extension, tels que les noeuds, les sorties de modèle et les sorties de document.

Chaque section peut contenir des déclarations statiques (comme les composants dans un élément) ou des processus dynamiques simples (comme le calcul d'un modèle de données de sortie d'un noeud), ou les deux. Le format général d'un fichier de spécifications CLEF est le suivant :

```
<?xml version="1.0" encoding="UTF-8" ?>
<Extension ... >
  <ExtensionDetails ... />
  <Resources
    Section Resources
  </Resources>
  <CommonObjects>
    Section Common Objects
  </CommonObjects>
  <UserInterface>
    Section User Interface (Palettes)
  </UserInterface>
  Section Object Definition
  définition d'objet
  définition d'objet
  définition d'objet
  ...
</Extension>
```

**Lignes de commentaires**

A tout moment, vous pouvez inclure une ligne de commentaire dans le fichier de spécifications au format suivant :

```
<!-- texte de commentaire -->
```

**Obligatoire ou facultatif ?**

Dans les définitions d'éléments des sections suivantes (normalement identifiées par l'en-tête **Format**), les attributs d'éléments et les éléments enfant sont facultatifs sauf s'ils sont signalés comme "(Requis)". Pour obtenir la syntaxe complète des éléments, consultez la rubrique l'annexe A, *Schéma XML CLEF* sur p. 252.

**Exemple d'un fichier de spécifications**

Voici un exemple complet d'un fichier de spécifications CLEF, pour un noeud simple de transformation de données.

```
<?xml version="1.0" encoding="UTF-8"?>
<Extension version="1.0" debug="true">
  <ExtensionDetails id="urlparser" providerTag="spss" label="Module CLEF URL" version="1.0"
  provider="IBM Corp." copyright="(c) 2005-2011 IBM Corp." description="Extension CLEF pour transformer les URL"/>
  <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="Impossible d'initialiser un homologue" />
          <StatusCode code="1" status="error" message="erreur de lecture des données d'entrée" />
          <StatusCode code="2" status="erreur" message="Erreur interne" />
          <StatusCode code="3" status="erreur" message="Le champ d'entrée n'existe pas" />
        </StatusCodes>
      </Module>
    </Execution>
  </Node>
</Extension>
```

```

    </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>

```

L'élément `ExtensionDetails` fournit des informations de base sur l'extension utilisée en interne par IBM® SPSS® Modeler.

L'élément `Resources` définit l'emplacement d'une bibliothèque côté serveur qui sera référencée dans le fichier ultérieurement. La spécification du chemin indique que la bibliothèque se trouve dans le répertoire d'installation SPSS Modeler sous `\ext\bin\spss.urlparser\urlparser.dll`.

Ce fichier de spécifications particulier ne comprend pas d'élément `CommonObjects`.

L'élément `Node` décrit toutes les informations sur le noeud lui-même :

- Sous `Properties`, deux propriétés sont initialement déclarées pour une utilisation ultérieure sur un onglet de la boîte de dialogue du noeud.
- L'élément `UserInterface` définit la présentation et l'apparence de l'onglet de la boîte de dialogue du noeud spécifique à cette extension (SPSS Modeler fournit d'autres onglets).
- L'élément `Execution` définit les éléments utilisés lorsque le noeud est exécuté. Dans ce cas, ces éléments sont la bibliothèque coté serveur déclarée auparavant dans le fichier et un ensemble de messages à afficher si l'exécution renvoie un code d'état particulier.
- L'élément `OutputDataModel` définit la transformation de données effectuée par ce noeud. Il précise que le modèle de données d'entrée (l'ensemble d'entrées de champs de ce noeud) doit être remplacé par l'ensemble de champs définis ici, qui constituent le modèle de données de sortie (l'ensemble de champs transmis à tous les noeuds situés en aval, à moins que ce modèle soit modifié ultérieurement). Dans cet exemple précis, le noeud transmet les deux champs d'origine non modifiés (`id_fieldname` et `url_fieldname`) mais ajoute quatre champs supplémentaires dont les noms sont dérivés de `url_fieldname`.

Ce fichier de spécifications est extrait d'un des exemples de noeuds fournis avec l'installation IBM® SPSS® Modeler. [Pour plus d'informations, reportez-vous à la section Noeud de transformation de données \(URL Parser\) dans le chapitre 3 sur p. 34.](#)

## Déclaration XML

La déclaration XML est facultative et indique la version XML utilisée ainsi que les détails du format de codage des caractères.

**Exemple**

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

**Elément Extension**

L'élément Extension constitue la partie principale du fichier et contient toutes les autres sections. Le format est le suivant :

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

où :

version est le numéro de version de l'extension.

debug est facultatif ; s'il est défini sur true, il ajoute un onglet Déboguer aux boîtes de dialogue ou cadres associés au noeud ou à la sortie CLEF et permet d'accéder aux propriétés et aux conteneurs définis pour cet objet. La valeur par défaut est false. [Pour plus d'informations, reportez-vous à la section Utilisation de l'onglet Déboguer dans le chapitre 10 sur p. 248.](#)

**Section Extension Details**

La section Extension Details offre des informations de base sur l'extension.

**Format**

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

où :

providerTag (requis) est un nom qui identifie de manière unique le fournisseur de cette extension. Veuillez noter que la valeur ne doit pas comprendre la chaîne spss qui est réservée à un usage interne.

id (requis) est un nom qui identifie de manière unique cette extension et est utilisé dans les messages du système à son sujet. Par convention, le fichier d'extension est placé dans un dossier nommé `\ext\lib\providerTag.id` dans le répertoire d'installation de IBM® SPSS® Modeler.

label (requis) est l'étiquette d'affichage de l'extension. Ce texte est affiché dans le champ Nom du gestionnaire de palettes lorsque le noeud est ajouté. [Pour plus d'informations, reportez-vous à la section Test d'une extension CLEF dans le chapitre 10 sur p. 246.](#)

version est le numéro de version de cette extension.

provider est une chaîne qui identifie le fournisseur de l'extension. Ce texte est affiché dans le champ Fournisseur du gestionnaire de palettes lorsque le noeud est ajouté. Sa valeur par défaut est la chaîne (unknown).

copyright est le copyright de cette extension. Ce texte est affiché dans le champ Copyright du gestionnaire de palettes lorsque le noeud est ajouté.

description est une courte description de l'objet de l'extension. Ce texte est affiché dans le champ Description du gestionnaire de palettes lorsque le noeud est ajouté.

### **Exemple**

```
<ExtensionDetails providerTag="myco" id="sorter" name="Sort Data" version="1.2"
  provider="Ma société Inc." copyright="(c) 2005-2006 Ma société Inc."
  description="Un exemple d'extension qui trie les données à l'aide des commandes OS intégrées."/>
```

## **Section Resources**

Cette section définit les ressources externes nécessaires au fonctionnement de cette extension.

### **Format**

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

où :

Bundle identifie un ensemble de ressources localisées côté client. [Pour plus d'informations, reportez-vous à la section Groupes sur p. 43.](#)

JarFile identifie un fichier Java .jar côté client. [Pour plus d'informations, reportez-vous à la section Fichiers Jar sur p. 43.](#)

SharedLibrary identifie une bibliothèque ou une DLL côté serveur. [Pour plus d'informations, reportez-vous à la section Bibliothèques partagées sur p. 44.](#)

HelpInfo spécifie le type d'informations d'aide (le cas échéant) de l'extension. [Pour plus d'informations, reportez-vous à la section Mise en oeuvre d'un système d'aide dans le chapitre 7 sur p. 205.](#)

### **Exemple**

```
<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>
```

## **Groupes**

L'élément Bundle spécifie un groupe de ressources côté client (un ensemble de messages pour la localisation, par exemple) qui peut être implémenté en tant que fichier .properties ou un fichier Java .class. [Pour plus d'informations, reportez-vous à la section Localisation dans le chapitre 8 sur p. 208.](#)

### **Format**

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

où :

id (requis) est un identificateur unique de ce groupe.

path (requis) spécifie l'emplacement du fichier de groupe associé au dossier parent de ce fichier de spécifications. Où le groupe désigne un fichier .properties, le chemin ne doit pas comprendre d'extensions de langue ni de suffixe .properties.

### **Exemple**

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

Ceci indique qu'un groupe de ressources existe dans un fichier nommé messages.properties dans le même dossier que le fichier de spécifications.

## **Fichiers Jar**

L'élément JarFile désigne un fichier archive Java côté client (.jar) qui fournit des classes Java et d'autres ressources côté client pour cette extension.

### **Format**

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

où :

`id` (requis) est un identificateur unique de ce fichier `.jar`.

`path` (requis) spécifie l'emplacement du fichier `.jar` associé au dossier parent de ce fichier de spécifications.

### **Exemple**

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

Ceci indique qu'un fichier `.jar` pour cette extension se trouve dans le même dossier que le fichier de spécifications.

## **Bibliothèques partagées**

L'élément `SharedLibrary` désigne une bibliothèque partagée ou une DLL côté serveur. Ceci n'est généralement nécessaire que pour la prise en charge d'exécution de noeuds. Où une bibliothèque implémente plusieurs modules, un élément `Module` dans la section `Execution` de la spécification du noeud désigne un module spécifique de la bibliothèque.

### **Format**

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

où :

`id` (requis) est un identificateur unique de cette bibliothèque partagée.

`path` (requis) indique l'emplacement de la bibliothèque partagée associée au dossier `\ext\bin` dans le répertoire d'installation côté serveur. Veuillez noter que ce chemin ne doit pas comprendre l'extension de fichier de bibliothèque partagée (`.dll`).

### **Exemple**

La déclaration de bibliothèque partagée suivante :

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

spécifie que la bibliothèque partagée doit être chargée depuis :

```
install_dir\ext\bin\spss.binning\Binning.dll
```

où `install_dir` est le répertoire dans lequel les composants CLEF côté serveur sont installés.

Parce que la bibliothèque implémente plusieurs modules, le module spécifique requis (`supervisedBinning`) est identifié au moyen d'un élément `Module` dans la spécification du noeud de création et désigne l'identificateur de bibliothèque comme suit :

```
<Execution>
```

```
<Module libraryId="regroupement par casiers" name="supervisedBinning" .../>
```

```
...
```



</Execution>

## Informations d'aide

L'élément facultatif `HelpInfo` désigne quels types d'aide sont fournis pour cette extension. [Pour plus d'informations](#), reportez-vous à la section *Mise en oeuvre d'un système d'aide dans le chapitre 7 sur p. 205*.

## Section Common Objects

La section facultative `Common Objects` désigne des objets pouvant être partagés entre des éléments définis à un autre emplacement du fichier de spécifications. Certains types d'objet de cette section (les énumérations de propriétés, par exemple) peuvent également être définis localement à l'emplacement approprié alors que d'autres (les modèles et documents, par exemple) peuvent uniquement être définis ici.

### Format

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

où :

Les `PropertyTypes` permettent que les définitions de propriétés communes soient partagées entre les objets. [Pour plus d'informations](#), reportez-vous à la section *Types de propriétés sur p. 46*.

Les `PropertySets` sont généralement utilisés lorsque les noeuds de création de modèle, les objets de sortie de modèle et les noeuds applicateurs de modèle comprennent le même ensemble de propriétés. [Pour plus d'informations](#), reportez-vous à la section *Ensembles de propriétés sur p. 47*.

Les `ContainerTypes` définissent les types de conteneurs qui sont des objets pouvant gérer des structures de données complexes. [Pour plus d'informations](#), reportez-vous à la section *Types de conteneurs sur p. 48*.

Les `Actions` définissent les informations de base sur les interventions de l'utilisateur, au moyen des menus ou des barres d'outils par exemple. [Pour plus d'informations](#), reportez-vous à la section *Actions sur p. 50*.

`Catalogs` mettent en oeuvre une commande qui autorise le choix d'une ou de plusieurs options dans une liste de valeurs générées de manière dynamique par le serveur. [Pour plus d'informations](#), reportez-vous à la section *Catalogs sur p. 51*.

**Exemple**

```

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

```

**Types de propriétés**

La section facultative Property Types permet le partage des définitions de propriétés communes entre les objets. Il s'agit surtout d'une facilité de maintenance : par exemple, la définition d'une propriété peut apparaître à un seul endroit plutôt que d'être dupliquée à plusieurs endroits. Le partage des définitions permet également d'assurer la compatibilité entre les propriétés de différents objets dont les valeurs sont copiées lorsqu'une nouvelle instance d'un objet est créée.

Les types de propriétés ne peuvent être définis que dans la section Common Objects.

**Format**

```

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

```

Les attributs PropertyType sont les suivants.

id (requis) est un identificateur unique pour le type de propriété.

isKeyed, s'il est défini sur true, indique que le type de propriété est saisi. Une propriété saisie associe un ensemble d'opérations à un champ au moyen d'une commande définie par l'utilisateur (reportez-vous à la rubrique [Contrôle Propriété sur p. 173](#)). Si isKeyed est défini sur true, l'attribut valueType doit être défini sur structure. Pour plus d'informations sur les propriétés structurées, reportez-vous à la rubrique [Propriétés structurées sur p. 77](#).

isList indique si la propriété est une liste de valeurs du type de valeur spécifié (true) ou une valeur simple (false).

max et min sont les valeurs maximales et minimales d'un intervalle.

valueType peut être l'un des types suivants :

- string

- encryptedString
- fieldName
- integer
- double
- boolean
- date
- enum(reportez-vous à Propriétés énumérées sur p. 76)
- structure(reportez-vous à Propriétés structurées sur p. 77)
- databaseConnection

Les éléments enfants Enumeration et Structure s'excluent mutuellement. Les éléments enfants Enumeration, Structure et DefaultValue sont utilisés dans des cas spécifiques : reportez-vous à Propriétés énumérées sur p. 76, Propriétés structurées sur p. 77 et Valeurs par défaut sur p. 79.

## ***Ensembles de propriétés***

Les ensembles de propriétés sont généralement utilisés lorsque les noeuds de création de modèle, les objets de sortie de modèle et les noeuds applicateurs de modèle comprennent le même ensemble de propriétés. Par exemple, un noeud de création de modèle peut définir un ensemble de propriétés par défaut qui peut être configuré dans le créateur mais qui ne sera pas utilisé avant l'application du modèle. Afin d'être transférés automatiquement, ils doivent également être inclus dans la sortie du modèle.

### ***Format***

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

où id est un identificateur unique de cet ensemble de propriétés.

Pour une description de l'élément Property, consultez [Properties sur p. 65](#).

### ***Exemple***

Cette exemple présente la définition d'un ensemble de deux propriétés : le nombre de prédictions à produire et s'il faut ou non inclure les probabilités. Dans la section Common Objects, vous devrez définir :

```
<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>

```

Puis, dans chacune des définitions du noeud de création de modèle, de l'objet de sortie de modèle et du noeud applicateur de modèle, se trouve un attribut `includePropertySets` comme suit (cet exemple de définition ne vaut que pour le noeud de création de modèle) :

```

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

```

## Types de conteneurs

Les conteneurs sont des objets qui jouent le rôle de paramètre de substitution pour les structures de données complexes comme les modèles et les documents. Un conteneur est défini comme étant d'un type particulier et les types de conteneurs sont définis ici. Les types de conteneurs suivants peuvent être définis :

- types de modèle
- types de documents

Les types de conteneurs peuvent être transférés entre le client et le serveur, clonés et enregistrés dans un fichier ou dans un référentiel de contenu. Un modèle est cloné lorsqu'un noeud applicateur de modèle est généré à partir d'un objet de sortie de modèle.

Chaque type de conteneur possède un ensemble de propriétés prédéfinies bien qu'il soit possible d'ajouter des propriétés personnalisées. Les types de conteneurs ne peuvent être définis que dans la section Common Objects.

### Format

Le format de la section Container Types est le suivant :

```

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

```

où :

`ModelType` indique le format d'un type de modèle particulier. [Pour plus d'informations, reportez-vous à la section Types de modèle sur p. 49.](#)

DocumentType indique le format d'un type de document particulier. [Pour plus d'informations, reportez-vous à la section Types de documents sur p. 49.](#)

### Exemple

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

## Types de modèle

Un modèle doit fournir des informations comme le nom des algorithmes, le type de modèle et les modèles de données d'entrée et de sortie. Une définition de type de modèle indique le format d'un type de modèle particulier.

Les informations sur le type de modèle peuvent être définies ici de manière statique dans le fichier de spécifications ou de manière dynamique lorsque le modèle est construit par le noeud de création de modèle.

### Format

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

où :

- id (requis) est un identificateur unique pour le type de modèle.
- format (requis) est le format du type de modèle et peut être utf8 (texte) ou binary. Le format de modèle doit être spécifié dans les informations statiques.

### Exemple

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

## Types de documents

Un **document** est un objet de sortie comme un graphique ou un rapport. Une définition de type de document indique le format d'un type de document particulier.

### Format

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

où :

- id (requis) est un identificateur unique pour le type de document.
- format (requis) est le format du type de document et peut être utf8 (texte) ou binary.

### Exemples

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

## Actions

Les actions définissent les informations de base sur les interventions de l'utilisateur, au moyen des menus ou des barres d'outils par exemple. Chaque action définit sous quelle forme elle doit être représentée dans l'interface utilisateur : par une étiquette, une info-bulle ou une icône, par exemple. Un ensemble d'actions est géré par une classe Java côté client qui est définie pour chaque groupe d'actions. Les actions peuvent également être définies dans des objets spécifiques.

### Format

```
<Actions>
<Action id="identifie" 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>
```

où :

id (requis) est un identificateur unique de l'action.

label (requis) est le nom d'affichage de l'action comme il apparaît sur l'interface utilisateur.

labelKey identifie l'étiquette à des fins de localisation.

description est la description de l'action. Par exemple, pour un élément de menu personnalisé ou une icône d'action d'une barre d'outils, ce serait le texte de l'info-bulle de cet élément de menu ou de cette icône.

descriptionKey identifie la description à des fins de localisation.

imagePath est l'emplacement d'un fichier graphique, tel que l'image d'une icône par exemple. Cet emplacement est donné en fonction du répertoire dans lequel le fichier de spécifications est installé.

imagePathKey identifie le chemin d'accès à l'image à des fins de localisation.

le mnemonic est le caractère alphabétique utilisé en association avec la touche Alt pour activer cette commande (par exemple, si vous donnez la valeur S, l'utilisateur peut activer cette commande au moyen de Alt+S).

mnemonicKey identifie le mnemonic à des fins de localisation. Si vous n'utilisez ni mnemonic, ni mnemonicKey, aucun mnemonic n'est disponible pour cette commande. [Pour plus d'informations, reportez-vous à la section Touches d'accès et raccourcis clavier dans le chapitre 6 sur p. 142.](#)

shortcut représente une chaîne indiquant un raccourci clavier (par exemple, CTRL+SHIFT+A) qui peut être utilisé pour lancer cette action.

shortcutKey identifie le raccourci à des fins de localisation. Si ni shortcut ni shortcutKey ne sont utilisées, aucun raccourci n'est disponible pour cette action. [Pour plus d'informations, reportez-vous à la section Touches d'accès et raccourcis clavier dans le chapitre 6 sur p. 142.](#)

### Exemple

```
<Actions>
  <Action id="generateSelect" label="Select Node..." labelKey="generate.selectNode.LABEL"
    imagePath="images/generate.gif" description="Génère un noeud Sélectionner"
    descriptionKey="generate.selectNode.TOOLTIP"/>
  <Action id="generateDerive" label="Derive Node..." labelKey="generate.deriveNode.LABEL"
    imagePath="images/generate.gif" description="Génère un noeud Calculer"
    descriptionKey="generate.deriveNode.TOOLTIP"/>
</Actions>
```

## Catalogs

Catalogs vous permet d'associer une propriété à une commande qui permet à l'utilisateur de choisir une ou plusieurs options dans une liste de valeurs générée de manière dynamique par le serveur.

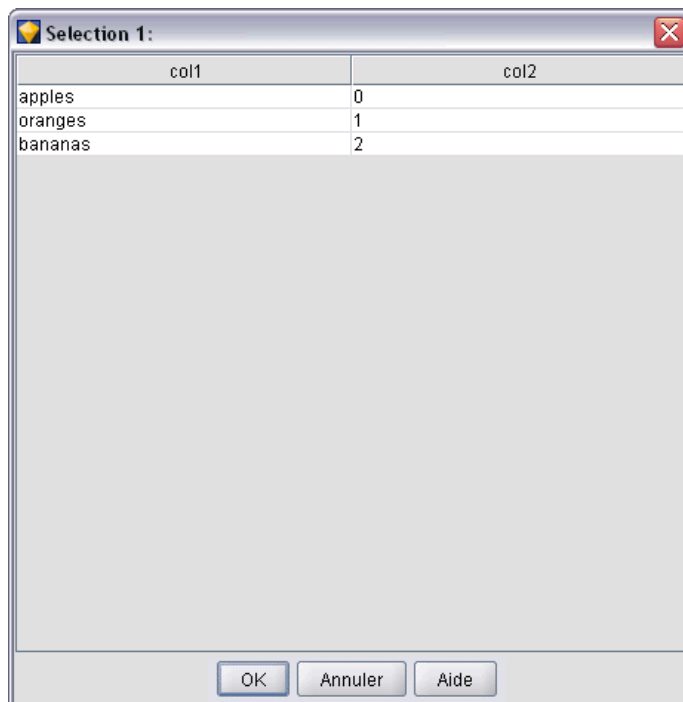
Figure 4-1

Commande permettant la sélection dans une liste de valeurs



Les valeurs sont affichées dans la commande sous forme de liste contextuelle lorsque l'utilisateur clique sur l'entrée <Sélectionner>.

Figure 4-2  
Liste des valeurs affichées



Lorsque l'utilisateur sélectionne une ligne dans la liste, la valeur de ligne d'une colonne spécifiée dans l'élément Catalog est placée dans la commande.

Figure 4-3  
Sélection effectuée à partir d'une liste de valeurs



### Format

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

où :

id (requis) est un identificateur unique du catalogue.

valueColumn (requis) est le numéro de la colonne dont la valeur est placée dans la commande lorsque l'utilisateur sélectionne une ligne. La numérotation des colonnes commence à 1.



Utilisez un élément `Attribute` par colonne, dans l'ordre des colonnes (voir l'exemple ci-dessous).

Lorsque l'utilisateur active une commande associée à un catalogue, le catalogue contenant la liste de valeurs est extrait du serveur grâce à un appel de la fonction `getCatalogInformation`. Cette fonction renvoie un document XML contenant la liste de valeurs. [Pour plus d'informations, reportez-vous à la section Fonctions d'homologue dans le chapitre 9 sur p. 220.](#)

### Exemple

Cet exemple illustre une partie du code utilisé pour définir les commandes de catalogue affichées auparavant. Trois catalogues sont définis et associés à trois commandes différentes dans un onglet de boîte de dialogue.

Pour commencer, les catalogues sont définis dans la section `Common Objects` :

```
<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>
```

Les propriétés à associer aux commandes sont ensuite définies dans la section `Propriétés de la définition du noeud` :

```
<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>
```

Dans la section `Interface utilisateur de la définition du noeud`, les commandes sont définies et associées aux définitions de catalogue par l'intermédiaire de références aux propriétés :

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

```

        <MultiItemChooserControl property="selection2" catalog="cat2" />
        <SingleItemChooserControl property="selection3" catalog="cat3" />
    </PropertiesPanel>
</Tab>

```

## Section User Interface (Palettes)

Cette section est facultative et n'est disponible que si vous souhaitez que cette extension définisse une palette ou une sous-palette personnalisée sur laquelle un noeud doit apparaître.

Là où une extension définit une palette ou une sous-palette personnalisée, les extensions chargées par la suite qui définissent les noeuds à inclure sur la même palette ou sous-palette peuvent ignorer cette section User Interface (Palettes). Il suffit que l'élément `Node` possède un attribut `customPalette` qui référence la palette. Les extensions sont chargées par ordre alphabétique de la valeur `providerTag.id` où ces valeurs sont les valeurs des attributs `providerTag` et `id` de l'élément `ExtensionDetails` de cette extension (consultez [Section Extension Details sur p. 41](#)). Par exemple, l'extension `myco.abc` est chargée avant l'extension `myco.def`.

*Remarque* : La section User Interface (Palettes) est différente de la section principale User Interface qui apparaît comme partie intégrante d'une définition d'objet individuel et qui est décrite dans le chapitre 6, *Création d'interfaces utilisateur* sur p. 128.

### Format

Le format de la section User Interface (Palettes) est le suivant :

```

<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>

```

Table 4-1  
Attributs de palette

Attribut	Description
id	(requis) Un identificateur unique de la palette ou de la sous-palette que vous définissez.
systemPalette	Utilisé uniquement lors de l'ajout d'une sous-palette à une palette du système. Identifie la palette du système dans laquelle cette sous-palette doit apparaître :  import - Sources recordOp - Ops sur lignes fieldOp - Ops sur champs graph - Graphiques modeling - Modélisation (voir ci-dessous) dbModeling - Modélisation de bases de données output - Sortie export - Exporter
customPalette	Utilisé uniquement lors de l'ajout d'une sous-palette à une palette personnalisée. Identifie la palette personnalisée dans laquelle cette sous-palette doit apparaître. C'est la valeur de l'attribut id de l'élément Palette qui définit la palette personnalisée.
relativePosition	Utilisé uniquement lors de la définition d'une palette personnalisée. Spécifie sa position sur la barre de palette au bas de l'écran.  Valeurs possibles :  first last before after  Si la valeur est <b>before</b> ou <b>after</b> , l'attribut <b>relativeTo</b> est également nécessaire (voir ci-dessous).  Si <b>relativePosition</b> est ignorée, la palette est placée à la fin de la barre.
relativeTo	Si la valeur de <b>relativePosition</b> est <b>before</b> ou <b>after</b> alors <b>relativeTo</b> est utilisé pour définir l'identificateur de la palette que cette palette personnalisée précède ou suit. Les identificateurs de palette sont répertoriés dans une liste comme valeurs de l'attribut de palette de l'élément Node (consultez <a href="#">Noeud sur p. 61</a> ).
label	(requis) Le nom d'affichage de la palette ou de la sous-palette comme il apparaît sur l'interface utilisateur.
labelKey	Identifie l'étiquette à des fins de localisation.
description	Le texte de l'info-bulle qui apparaît lorsque le curseur passe au-dessus de l'onglet de la palette (ne s'utilise pas pour les sous-palettes). Cette valeur sert aussi de description longue accessible du contrôle. <a href="#">Pour plus d'informations, reportez-vous à la section Accessibilité dans le chapitre 8 sur p. 215.</a>
descriptionKey	Identifie la description à des fins de localisation.
imagePath	Identifie l'emplacement de l'image utilisée sur l'onglet de la palette (n'est pas utilisé pour les sous-palettes). Cet emplacement est indiqué par rapport au répertoire d'installation du fichier de spécifications. Si vous ignorez cet attribut, aucune image n'est utilisée.

### Exemple : ajout d'un noeud à une palette du système

Supposons que votre société a développé un nouvel algorithme pour l'exploration de données audio et vidéo et que vous souhaitez intégrer cet algorithme à IBM® SPSS® Modeler. Commencez par définir un noeud de lecture de données personnalisé qui pourra lire les entrées de fichiers audio et vidéo.

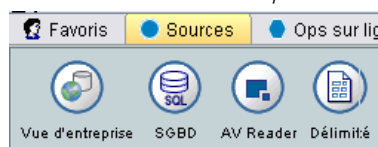
Vous décidez d'ajouter votre nouveau noeud de lecture de données à la palette du système Sources. Il vous suffit d'identifier la palette Sources au moyen de l'attribut `palette` de l'élément `Node`. [Pour plus d'informations, reportez-vous à la section Noeud sur p. 61.](#)

Ainsi, pour ajouter ce noeud après le noeud Base de données sur la palette Sources, utilisez :

```
<Node id="AVreader" type="dataReader" palette="import" relativePosition="after" relativeTo="database"
  label="Lecteur Audio-Vidéo">
```

Cela donne :

Figure 4-4  
Nouveau noeud sur une palette standard



### Exemple : ajout d'une palette personnalisée

Vous pouvez utiliser une palette IBM® SPSS® Modeler standard, mais optez pour une importance accrue de votre nouveau noeud. Pour ce faire, vous décidez de définir une palette personnalisée que vous placerez après la palette Favoris mais avant la palette Sources. Pour commencer, ajoutez la section User Interface (Palettes) pour définir la palette personnalisée comme suit :

```
<UserInterface>
  <Palettes>
    <Palette id="AV_mining" label="Exploration AV" relativePosition="before" relativeTo="import"
    description="Exploration Audio-Vidéo" />
  </Palettes>
</UserInterface>
```

L'attribut `relativeTo` doit utiliser l'identificateur interne de la palette Sources `import`.

Ensuite, modifiez la définition Node de la façon suivante :

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

Cette action place la palette Exploration AV entre les palettes Favoris et Sources du système.

Figure 4-5  
Nouveau noeud sur une nouvelle palette



### **Exemple : ajout d'une sous-palette personnalisée à une palette personnalisée**

Dans la continuité d'un des exemples précédents, vous décidez qu'il est préférable que le noeud de lecture des données soit placé sur sa propre sous-palette Sources AV de la palette Exploration AV. Pour ce faire, commencez par définir la sous-palette en ajoutant un deuxième élément Palette à la section User Interface (Palettes) :

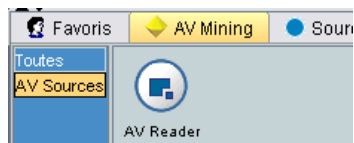
```
<UserInterface>
<Palettes>
<Palette id="AV_mining" label="Exploration AV" description="Exploration Audio-vidéo" />
  <Palette id="AV_mining.sources" customPalette="AV_mining" label="Sources AV" />
</Palettes>
</UserInterface>
```

Puis, modifiez l'élément Node pour se référer à l'identificateur de la sous-palette :

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

A présent, lorsque l'utilisateur clique sur l'onglet Exploration AV, il voit deux sous-palettes, l'une étiquetée Tous et l'autre Sources AV. Le noeud Lecteur AV apparaît sur les deux.

Figure 4-6  
Nouveau noeud sur une nouvelle sous-palette



Si vous ajoutez un autre nouveau noeud à une nouvelle sous-palette de exploration AV, ce nouveau noeud apparaît à la fois dans Tous et sur la nouvelle sous-palette mais pas sur la sous-palette Sources AV.

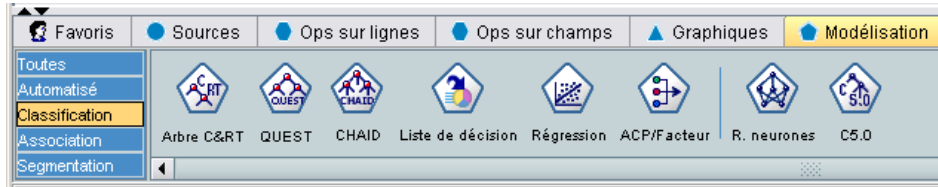
### **Exemple : ajout d'un noeud à une sous-palette du système**

Pour traiter les données des sources audio et vidéo, définissez un noeud de création de modèle. Vous décidez ensuite de l'ajouter à une palette Modélisation standard qui contient plusieurs sous-palettes standard. Vous choisissez de l'ajouter à la sous-palette Classification et de le placer juste avant le noeud R. neurones. Pour ce faire, vous définissez :

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

A présent, la sous-palette Classification ressemble à ceci :

Figure 4-7  
Nouveau noeud sur sous-palette standard



Veillez noter que le noeud est également ajouté dans la même position relative sur la sous-palette Tous de la palette Modélisation.

### Exemple : ajout d'une sous-palette personnalisée à une palette du système

Après avoir jeté un oeil au nombre de noeuds de création de modèle de la sous-palette Classification, vous réalisez que les utilisateurs risquent de ne pas remarquer facilement votre nouveau noeud. Une des façons de donner plus d'importance à votre noeud est d'ajouter votre propre sous-palette à la palette Modélisation et d'y placer votre noeud.

D'abord, définissez votre sous-palette personnalisée en ajoutant une section User Interface (Palettes) au fichier :

```
<UserInterface>
<Palettes>
<Palette id="modeling.av_modeling" systemPalette="modeling" label="Modélisation AV"
labelKey="av_modeling.LABEL" description="Contient des noeuds de modélisation pour l'exploration AV"
descriptionKey="av_modeling.TOOLTIP"/>
</Palettes>
</UserInterface>
```

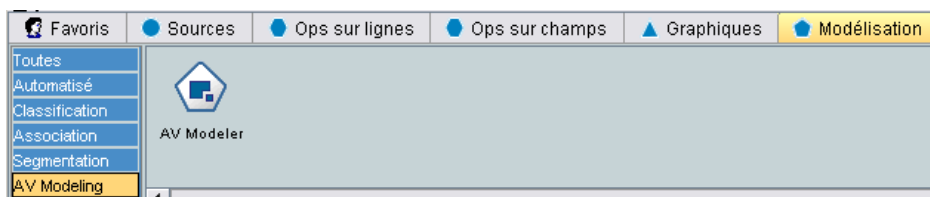
Veillez noter que vous devez définir systemPalette de manière explicite afin d'identifier la palette du système que vous étendez.

Puis, dans la section User Interface principale du noeud, précisez qu'il doit apparaître sur la sous-palette suivante :

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

Les sous-palettes personnalisées sont toujours placées après les sous-palettes du système, ainsi vous avez désormais :

Figure 4-8  
Nouveau noeud d'ajout personnalisé à une sous-palette standard



*Remarque* : Si vous souhaitez ajouter d'autres noeuds à une sous-palette Modélisation AV, leurs fichiers de spécifications ne nécessitent **pas** de section User Interface (Palettes), tant que l'extension Modeleur AV a été chargée en premier.

### ***Masquage ou suppression d'une palette ou d'une sous-palette personnalisée***

Si vous ne souhaitez plus que votre palette ou sous-palette personnalisée apparaisse, vous pouvez la masquer ou la supprimer avec le Gestionnaire de palettes de IBM® SPSS® Modeler.

Veillez noter qu'une opération de masquage affecte toutes les sessions SPSS Modeler mais qu'elle est contrôlée par une case à cocher et par conséquent réversible. Une opération de suppression est irréversible dans cette même session mais lorsque SPSS Modeler redémarre, l'élément réapparaît sauf si vous l'avez supprimé du fichier de spécifications ou avez supprimé l'extension entière. [Pour plus d'informations, reportez-vous à la section Désinstallation des extensions CLEF dans le chapitre 10 sur p. 251.](#)

Pour masquer ou supprimer une palette :

- ▶ Dans le menu principal de SPSS Modeler, sélectionnez :  
Outils > Favoris
- ▶ Sélectionnez une palette dans le champ Nom de palette, puis :
  - pour masquer la palette, désactivez la case Affiché ? correspondante
  - pour supprimer la palette, cliquez sur le bouton Supprimer.
- ▶ Cliquez sur OK.

Pour masquer ou supprimer une sous-palette :

- ▶ Dans le menu principal de SPSS Modeler, sélectionnez :  
Outils > Favoris
- ▶ Sélectionnez une palette dans le champ Nom de palette.
- ▶ Cliquez sur le bouton Sous-palettes.
- ▶ Sélectionnez une sous-palette dans le champ Nom de sous-palette, puis :
  - pour masquer la sous-palette, désactivez la case Affiché ? correspondante
  - pour supprimer la sous-palette, cliquez sur le bouton Supprimer.
- ▶ Cliquez sur OK.

### ***Section Object Definition***

Les éléments sont les parties les plus visibles d'une extension. La section Object Definition constitue le reste du fichier de spécifications CLEF et est utilisée pour définir les différents objets de l'extension. Les types d'objet suivants peuvent être définis :

- noeuds
- objets de sortie de modèle

- objets de sortie de document
- objets de sortie interactifs

Les **Noeuds** sont les objets qui apparaissent dans un flux. Les **objets de sortie de modèle** sont générés par les noeuds de création de modèle et apparaissent dans l'onglet Modèles du panneau de gestionnaire de la fenêtre principale. De la même façon, les **objets de sortie de document** sont générés par les noeuds de création de document et apparaissent dans l'onglet Sorties du même panneau. Les **objets de sortie interactifs** sont générés par les noeuds de création de modèle interactif et apparaissent dans l'onglet Sorties du panneau de gestionnaire.

La section Object Definition est composée d'une ou plusieurs de ces définitions d'objets.

Les éléments pouvant être définis pour les différents types d'objet sont décrits dans les sections suivantes. Certains de ces éléments sont communs à tous les types d'objet alors que d'autres sont spécifiques aux définitions de noeud ou de sortie de modèle. Les éléments propres aux objets sont signalés comme tels dans le texte.

- identificateur d'objet
- créateur de modèle
- créateur de document
- propriétés
- conteneurs
- interface utilisateur
- exécution
- modèle de données de sortie
- constructeurs

### ***Identificateur d'objet***

L'identificateur d'objet indique le type d'objet et est l'un des éléments suivants :

<Node .../>

<ModelOutput .../>

<DocumentOutput .../>

<InteractiveModelBuilder .../>

L'identificateur d'objet fournit également des informations sur la façon dont les objets doivent être présentés dans les scripts. L'attribut `scriptName` représente un nom unique de l'objet. Les scripts peuvent utiliser cet attribut pour définir un objet particulier (par exemple, un noeud dans un flux ou une sortie dans l'onglet Sorties).



## **Noeud**

Une définition de noeud décrit un objet pouvant apparaître dans un flux.

### **Format**

```
<Node id="identifieur" type="node_type" palette="palette" customPalette="custom_palette"
    relativePosition="position" relativeTo="noeud" label="display_label" labelKey="label_key"
    scriptName="script_name" helpLink="topic_id" description="description"
    descriptionKey="description_key">
  <ModelBuilder ... >
  ...
</ModelBuilder>
  <DocumentBuilder ... >
  ...
</DocumentBuilder>
  <ModelProvider ... />
  <Properties>
  ...
</Properties>
  <Containers>
  ...
</Containers>
  <UserInterface>
  ...
</UserInterface>
  <Execution>
  ...
</Execution>
  <OutputDataModel ...>
  ...
</OutputDataModel>
  <Constructors>
  ...
</Constructors>
</Node>
```

Les éléments autorisés dans la définition du noeud sont décrits dans les sections commençant à [Properties sur p. 65](#).

Table 4-2  
Attributs de noeud

Attribut	Description
id	(requis) Un identificateur de ce noeud, au format chaîne texte.
type	<p>(requis) Le type de noeud :</p> <p><b>dataReader</b> - noeud qui lit les données (par exemple, les noeuds de la palette Sources)  <b>dataWriter</b> - noeud qui écrit des données (par exemple, les noeuds de la palette Export)  <b>dataTransformer</b> - noeud qui transforme les données (par exemple, les noeuds Ops de lignes/de champs)  <b>modelBuilder</b> - noeud de création de modèle (par exemple, les noeuds de la palette Modélisation)  <b>documentBuilder</b> - noeud qui crée un graphique ou un rapport  <b>modelApplier</b> - noeud qui contient un modèle généré</p> <p>Le type de noeud détermine la forme de l'icône du noeud dans la palette ou l'espace de travail. <a href="#">Pour plus d'informations, reportez-vous à la section Présentation des noeuds dans le chapitre 2 sur p. 9.</a></p> <p>Si le noeud est de type <b>modelBuilder</b>, la définition du noeud doit comprendre l'élément <b>ModelBuilder</b> - consultez <a href="#">Concepteur de modèle sur p. 64.</a></p> <p>Si le noeud est de type <b>documentBuilder</b>, la définition du noeud doit comprendre l'élément <b>DocumentBuilder</b> - consultez <a href="#">Créateur de document sur p. 64.</a></p>
palette	<p>L'identificateur de l'une des palettes ou sous-palettes IBM® SPSS® Modeler standard dans laquelle le noeud doit apparaître, à savoir :</p> <p><b>import</b> - Sources  <b>recordOp</b> - Ops sur lignes  <b>fieldOp</b> - Ops sur champs  <b>graph</b> - Graphiques  <b>modeling</b> - Modélisation (voir ci-dessous)  <b>dbModeling</b> - Modélisation de bases de données  <b>output</b> - Sortie  <b>export</b> - Exporter</p> <p>La palette Modélisation contient plusieurs sous-palettes standard :</p> <p><b>modeling.classification</b> - Classification  <b>modeling.association</b> - Association  <b>modeling.segmentation</b> - Segmentation  <b>modeling.auto</b> - Automatisé</p> <p>Si vous omettez l'attribut <b>palette</b>, le noeud apparaît sur la palette Ops sur champs.</p> <p>Remarque : L'attribut <b>palette</b> est uniquement utilisé pour les noeuds de création de modèle.</p>
customPalette	<p>L'identificateur d'une palette ou sous-palette personnalisée dans laquelle doit apparaître un noeud. C'est la valeur de l'attribut <b>id</b> d'un élément <b>Palette</b> qui est spécifiée dans la section User Interface (Palettes) du fichier. <a href="#">Pour plus d'informations, reportez-vous à la section Section User Interface (Palettes) sur p. 54.</a></p>

Attribut	Description
relativePosition	Définit la position du noeud dans la palette. Valeurs possibles :  first last before after  Si la valeur est before ou after, l'attribut relativeTo est également nécessaire (voir ci-dessous).  Si relativePosition est ignorée, le noeud est placé en dernière position dans la palette.
relativeTo	Si la valeur de relativePosition est before ou after alors relativeTo est utilisé pour définir le noeud dans la palette que ce noeud précède ou suit. La valeur de relativeTo est le nom de script du noeud.  Le nom de script d'un noeud SPSS Modeler standard se trouve dans la section "Référence des propriétés" du <i>Guide de génération de scripts et d'automatisation SPSS Modeler</i> , mais sans le suffixe ...node (par exemple, pour le noeud Base de données, utilisez database, et non databasenode).  Pour un noeud CLEF, c'est la valeur de l'attribut scriptName de ce noeud.
label	(requis) Le nom d'affichage du noeud comme il apparaît dans la palette, l'espace de travail ou les boîtes de dialogue.
labelKey	Identifie l'étiquette à des fins de localisation.
scriptName	Utilisé pour identifier de façon unique le noeud lorsqu'il est référencé dans un script. <a href="#">Pour plus d'informations, reportez-vous à la section Utilisation de noeuds CLEF dans les scripts sur p. 93.</a>
helpLink	Un identificateur facultatif d'une rubrique d'aide qui apparaît lorsque l'utilisateur appelle le système d'aide, le cas échéant. Le format de l'identificateur dépend du type de système d'aide (consultez <a href="#">le chapitre 7, Ajout d'un système d'aide sur p. 205</a> ) :  Aide HTML - URL de la rubrique d'aide JavaHelp - ID de la rubrique
description	Une description texte du noeud.
descriptionKey	Identifie la description à des fins de localisation.

Les éléments autorisés dans la définition du noeud sont décrits dans les sections commençant à [Concepteur de modèle sur p. 64](#).

### **Exemple**

Pour un exemple de définition de noeud, consultez [Exemple d'un fichier de spécifications sur p. 39](#).

### **Sortie du modèle**

Une définition de sortie de modèle décrit un modèle généré : un objet qui apparaît dans l'onglet Modèles dans le panneau du gestionnaire après l'exécution d'un flux.

Pour des détails supplémentaires sur le codage de cette partie du fichier, consultez [Sortie du modèle sur p. 107](#).

### **Sortie de document**

Une définition de sortie de document décrit un objet (un tableau ou graphique généré, par exemple) qui apparaîtra dans l'onglet Sorties du panneau du gestionnaire après l'exécution d'un flux.

Pour des détails supplémentaires sur le codage de cette partie du fichier, consultez [Sortie de document sur p. 122](#).

### **Créateur de modèles interactifs**

Pour des détails supplémentaires sur le codage de cette partie du fichier, consultez [Création de modèles interactifs sur p. 108](#).

### **Concepteur de modèle**

*Cet élément est utilisé dans les définitions de l'élément Node uniquement.*

Pour des détails supplémentaires sur le codage de cette partie du fichier, consultez [le chapitre 5, Création de modèles et de documents sur p. 96](#).

### **Créateur de document**

*Cet élément est utilisé dans les définitions de l'élément Node uniquement.*

Pour des détails supplémentaires sur le codage de cette partie du fichier, consultez [le chapitre 5, Création de modèles et de documents sur p. 96](#).

### **Fournisseur de modèle**

*Cet élément est utilisé dans les définitions de l'élément Node uniquement.*

Pour définir un objet de sortie de modèle et un noeud applicateur de modèle, vous pouvez utiliser l'élément `ModelProvider` pour spécifier le conteneur de ce modèle. Vous pouvez également choisir si le modèle est stocké ou non au format PMML. Vous pouvez visualiser les modèles PMML dans un afficheur personnalisé ou dans un afficheur de résultats de modèle IBM® SPSS® Modeler standard fourni par l'élément `ModelViewerPanel`. [Pour plus d'informations, reportez-vous à la section Panneau Visualiseur de modèle dans le chapitre 6 sur p. 149](#).

#### **Format**

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

où :

container est le nom du conteneur du modèle.

isPMML indique si le modèle est stocké ou non au format PMML.

### **Exemple**

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

Pour un exemple de l'utilisation de ModelProvider dans le contexte d'un noeud applicateur de modèle, consultez l'exemple dans [Panneau Visualiseur de modèle sur p. 149](#).

## **Properties**

La définition d'une propriété est composée d'un ensemble de paires nom/valeur. Les définitions de propriétés individuelles, qui peuvent être nombreuses, se trouvent toutes dans une seule section Properties.

*Remarque* : Si une propriété est définie dans la section Properties, il est inutile de la définir pour une commande de propriété individuelle car les définitions de la section Properties sont prioritaires. Par conséquent, nous vous recommandons de définir les propriétés dans la section Properties.

La seule exception à cette règle concerne l'attribut label. Si l'attribut label est défini pour une commande de propriété, alors toute définition de propriété qui se trouve dans la déclaration de cette commande de propriété (et pas uniquement la définition de label) est prioritaire sur sa définition correspondante dans la section Properties. Veuillez noter que cette exception s'applique uniquement aux commandes de propriétés et pas à d'autres types de commandes comme les menus, les éléments de menus et les éléments de barres d'outils. Ces derniers doivent définir une étiquette de manière explicite, directement (menus) ou indirectement à travers un élément Action (éléments de menus et éléments de barres d'outils).

### **Format**

```
<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>
```

Les éléments Enumeration, Structure et DefaultValue sont utilisés dans des cas spécifiques. [Pour plus d'informations, reportez-vous à la section Types de valeur sur p. 75](#).

Les attributs d'élément Property sont les suivants :

Table 4-3  
Attributs des propriétés

Attribut	Description
name	(requis) Un nom unique pour la propriété.
scriptName	Le nom qui référence la propriété dans un script. <a href="#">Pour plus d'informations, reportez-vous à la section Utilisation de noeuds CLEF dans les scripts sur p. 93.</a>
valueType	Indique le type de valeur autorisée que cette propriété peut prendre qui est l'un des types suivants :  string encryptedString fieldName integer double boolean date enum structure databaseConnection  <a href="#">Pour plus d'informations, reportez-vous à la section Types de valeur sur p. 75.</a>
isList	Indique si la propriété est une liste de valeurs du type de valeur spécifié (true) ou une valeur simple (false).
defaultValue	La valeur par défaut de cette propriété. Celle-ci peut être exprimée sous la forme d'un attribut de valeur simple ou d'un élément composé et doit correspondre à des valeurs valides spécifiées.
label	Nom d'affichage de la valeur de propriété telle qu'elle apparaît sur l'interface utilisateur.
labelKey	Identifie l'étiquette à des fins de localisation.
description	Une description de la propriété.
descriptionKey	Identifie la description à des fins de localisation.

En option, les propriétés peuvent indiquer comment déterminer des valeurs valides :

- Pour les valeurs numériques, une valeur valide sera la valeur minimale et/ou maximale.
- Pour les chaînes, la valeur valide est généralement une sélection de champ (par exemple, tous les champs, tous les champs numériques, tous les champs discrets, etc.) mais peut également être une sélection de fichier.
- Pour les énumérations, cette valeur sera l'ensemble des valeurs valides.

Les propriétés saisies doivent également déclarer comment déterminer les clés valides. Veuillez noter que le type de clé d'une propriété saisie doit être une chaîne ou une énumération. [Pour plus d'informations, reportez-vous à la section Types de propriétés sur p. 46.](#)

La valeur facultative par défaut associée à cette propriété est évaluée lorsque l'objet associé est créé. Par exemple, les propriétés de noeud par défaut sont évaluées à chaque fois qu'une nouvelle instance de ce noeud est créée ; les propriétés d'exécution sont évaluées à chaque fois que le noeud est exécuté. Cette évaluation s'exécute selon l'ordre dans lequel les propriétés ont été déclarées.

Veillez noter qu'une définition de propriété peut se référer à un type de propriété déclaré dans la section Common Objects.

## Containers

Un conteneur est un paramètre de substitution pour un objet de sortie dont la génération est définie dans la section Constructors.

### Format

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

où :

name correspond à la valeur de l'attribut cible d'un élément CreateModel ou CreateDocument (voir dans [Utilisation des constructeurs sur p. 123](#)), et associe de manière indirecte le conteneur à l'un des types de conteneur déclarés dans la section Common Objects.

### Exemple

Pour commencer, les types de conteneur sont déclarés dans la section Common Objects. Il existe un type de conteneur pour les modèles, qui est au format texte, et deux types de conteneur pour les objets de sortie de document, l'un au format (texte) par défaut pour la sortie HTML et l'autre au format binaire pour une sortie compressée.

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

Dans la section Execution de la définition de noeud, les fichiers de sortie sont définis comme fichiers de conteneur avec des types de conteneur correspondant aux identificateurs spécifiés dans la section Common Objects :

```
<Node id="mynode" ... >
  ...
  <Execution>
    ...
    <OutputFiles>
      <ContainerFile id="pmmml" path="{tempfile}.pmmml" containerType="my_model" />
      <ContainerFile id="htmloutput" path="{tempfile}.html" containerType="html_output" />
      <ContainerFile id="zipoutput" path="{tempfile}.zip" containerType="zip_outputType" />
    </OutputFiles>
```

A la suite, la section Constructors définit les objets de sortie à générer lorsque le noeud est exécuté. Ici, les éléments CreateModel et CreateDocument ont un attribut sourceFile qui correspond à un fichier de conteneur, tel que spécifié dans la section Output Files :

```

    <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>

```

Finalement, la section Model Output associe un conteneur avec un objet de sortie de modèle ou un objet de sortie de document. Dans l'élément Container, l'attribut name correspond à l'attribut target dans les éléments CreateModel et CreateDocument qui viennent d'être spécifiés :

```

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

```

## Interface utilisateur

Le fichier de spécifications prend en charge une gamme de composants d'interface utilisateur qui permet aux objets de s'afficher et modifier les commandes et les propriétés. Des fonctions vous permettent de spécifier le comportement de la présentation et du redimensionnement du composant et de choisir si ce composant doit être activé ou non, ou rendu visible lorsque d'autres commandes sont modifiées.

La section User Interface spécifie l'apparence d'un objet. Cette spécification peut permettre de personnaliser un composant basique de l'interface utilisateur comme une boîte de dialogue de propriétés de noeud ou une fenêtre de sortie.

La section User Interface est une partie obligatoire de la spécification de l'élément Node.

Pour des détails supplémentaires sur le codage de cette partie du fichier, consultez [le chapitre 6, Création d'interfaces utilisateur](#) sur p. 128.

## Execution

*Cet élément est utilisé dans les définitions de l'élément Node uniquement.*

La section Execution définit les propriétés et les fichiers utilisés lors de l'exécution d'un noeud.



**Format**

```

<Execution>
  <Properties>
    ...
  </Properties>
  <InputFiles>
  <ContainerFile ... />
    ...
  </InputFiles>
  <OutputFiles>
    <ContainerFile ... />
    ...
  </OutputFiles>
  <Module ... >
    <StatusCodes ... />
  </Module>
  <Constructors ... />
</Execution>

```

La section Execution comprend la définition d'un ensemble de propriétés qui sont re-crées chaque fois que le noeud est exécuté et qui sont uniquement disponibles pendant l'exécution du noeud.

Les informations d'exécution peuvent également définir l'ensemble des fichiers d'entrée à générer avant l'exécution du noeud et tout fichier de sortie généré pendant cette exécution.

Un nombre illimité de fichiers d'entrée et de sortie peut être spécifié. Chaque fichier d'entrée est associé à un conteneur défini par le noeud. Chaque fichier de sortie est généralement utilisé pour créer des conteneurs d'objets générés. Le format d'un fichier d'entrée ou de sortie est déterminé par la déclaration du conteneur dans la section Common Objects.

**Exemple**

Pour un exemple de section Execution, consultez [Exemple d'un fichier de spécifications sur p. 39](#).

**Propriétés (Exécution)**

Cette section définit l'ensemble des propriétés runtime disponibles uniquement lors de l'exécution du noeud.

**Format**

Le format est semblable à celui de la section Properties dans la partie principale de la définition de l'élément. [Pour plus d'informations, reportez-vous à la section Properties sur p. 65](#).

Pendant l'exécution d'un noeud de création de modèle ou de document, un **fichier temporaire de serveur** est créé pour stocker l'objet de sortie de modèle ou de document. Le serveur accède à ce fichier et amène l'objet dans le client où il est encapsulé dans un conteneur. Vous devez spécifier ce fichier ici.

**Exemple**

Cet exemple présente la manière de spécifier le fichier temporaire de serveur.

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

**Fichiers d'entrée**

Cette section définit l'ensemble des fichiers d'entrée à générer avant l'exécution du noeud. Dans ce contexte, les fichiers d'entrée sont les fichiers qui constituent une entrée pour l'exécution du noeud sur le serveur. Par exemple, un noeud applicateur de modèle comprend un conteneur de modèle qui est transféré vers le fichier d'entrée spécifié lors de l'exécution du noeud.

**Format**

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

Dans l'élément ContainerFile d'un fichier d'entrée, les attributs sont ceux présentés dans le tableau suivant.

Table 4-4  
Attributs de fichier de conteneur - fichiers d'entrée

Attribut	Description
id	Un identificateur unique du fichier de conteneur.
path	L'emplacement sur le serveur où vous souhaitez générer le fichier d'entrée (par exemple, l'emplacement d'un fichier temporaire de serveur - consultez <a href="#">Propriétés (Exécution) sur p. 69</a> ).
container	L'identificateur du conteneur de l'objet envoyé au serveur comme entrée.

**Exemple**

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

**Fichiers de résultats**

Cette section spécifie les fichiers de sortie générés pendant l'exécution du noeud sur le serveur. Les fichiers de sortie (par exemple, les résultats de l'exécution d'un noeud de création de modèle ou de document) sont retournés au client après l'exécution.

**Format**

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

Dans l'élément ContainerFile, les attributs sont ceux présentés dans le tableau suivant.

Table 4-5  
Attributs de fichier de conteneur - fichiers de sortie

Attribut	Description
id	Un identificateur unique du fichier de conteneur.
path	L'emplacement sur le serveur de l'objet à transférer au client (par exemple, l'emplacement d'un fichier temporaire de serveur - consultez <a href="#">Propriétés (Exécution) sur p. 69</a> ).
containerType	L'identificateur du type de conteneur de l'objet (c'est-à-dire l'ID du type de modèle ou de document) permettant le transfert de l'objet au format approprié. <a href="#">Pour plus d'informations, reportez-vous à la section Types de conteneurs sur p. 48.</a>

**Exemple**

```
<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>
```

**Modules**

Cette section définit la bibliothèque partagée côté serveur à utiliser pendant l'exécution du noeud (par exemple, une DLL à charger dans la mémoire).

**Format**

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

où :

libraryId est l'identificateur d'une bibliothèque déclarée dans un élément Shared Library de la section Resources. [Pour plus d'informations, reportez-vous à la section Bibliothèques partagées sur p. 44.](#)

name est utilisé si la bibliothèque est partagée par plusieurs noeuds et identifie le noeud spécifique qui est exécuté. Si la bibliothèque est utilisée par un seul noeud, le nom peut être laissé vide.

**Exemple**

```

<Module libraryId="mynode1" name="mynode">
  <StatusCodes>
    <StatusCode code="0" status="error" message="Une exception a eu lieu" />
    <StatusCode code="1" status="error" message="Erreur lors de la lecture des données d'entrée" />
    ...
  </StatusCodes>
</Module>

```

**Codes d'état**

La majorité des programmes effectue une vérification des erreurs et affiche les messages nécessaires, généralement en renvoyant des entiers pour indiquer que l'opération s'est terminée avec succès ou d'autres états, le cas échéant. L'API côté serveur peut retourner un code d'état après l'exécution d'un flux contenant le noeud. [Pour plus d'informations, reportez-vous à la section Document Détails de l'état dans le chapitre 9 sur p. 238.](#)

La section Status Codes vous permet d'associer un message à un code d'état particulier et de l'afficher à l'utilisateur.

**Format**

```

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

```

Les attributs des codes d'état sont présentés dans le tableau suivant.

Table 4-6  
Attributs des codes d'état

Attribut	Description
code	Le code d'état (une valeur entière) auquel le message sera associé.
status	La classification de l'état : success - noeud exécuté avec succès warning - noeud exécuté avec des avertissements error - échec de l'exécution du noeud
message	Le message qui s'affiche lorsque ce code d'état est renvoyé.
messageKey	Identifie le message à des fins de localisation.

**Exemple**

Dans cet exemple, le texte du message d'erreur est compris dans l'élément StatusCode :

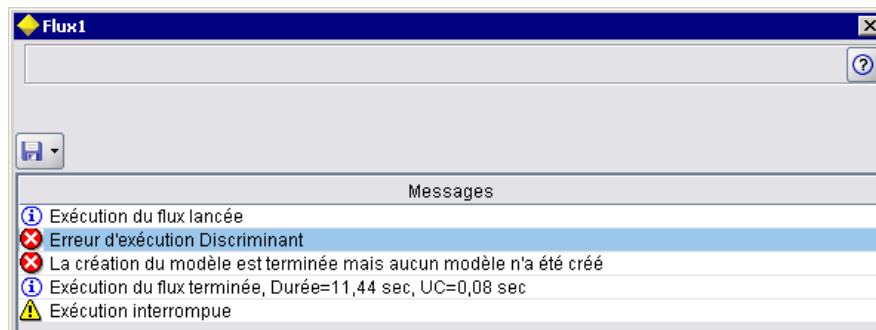
```

<StatusCodes>
  <StatusCode code="0" status="error" message="Impossible d'initialiser un homologue" />
  <StatusCode code="1" status="error" message="Erreur lors de la lecture des données d'entrée" />
  <StatusCode code="2" status="error" message="Erreur interne" />
  <StatusCode code="3" status="error" message="Le champ d'entrée n'existe pas" />
</StatusCodes>

```

Lors de l'exécution, si l'API côté serveur renvoie le code d'état 3, le message suivant apparaît :

Figure 4-9  
Affichage du message d'erreur



Dans l'exemple suivant, les textes des messages d'erreur sont référencés par un attribut `messageKey` :

```
<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>
```

Un fichier de propriétés (messages.properties par exemple), qui se trouve dans le même dossier que le fichier de spécifications, contient les textes des messages ainsi que tout autre texte à afficher :

```
...
initErrMsg.LABEL=Echec de l'initialisation.
inputErrMsg.LABEL=Erreur lors de la lecture des données d'entrée.
internalErrMsg.LABEL=Erreur interne.
invalidMetadataErrMsg.LABEL=Métadonnées (sur les champs d'entrée/sortie) non valides.
...
```

Cette méthode est utile lorsque le texte affiché doit être localisé pour les marchés étrangers, par exemple, ainsi tout le texte à localiser se trouve dans un seul fichier. [Pour plus d'informations, reportez-vous à la section Localisation dans le chapitre 8 sur p. 208.](#)

## Modèle de données de sortie

*Cet élément est utilisé dans les définitions de l'élément Node uniquement.*

La section Output Data Model indique la façon dont un modèle de données est affecté par certaines propriétés.

Le modèle de données de sortie peut être déterminé de trois façons :

- en utilisant les fonctions de définition de l'ensemble des champs du fichier de spécifications  
[Pour plus d'informations, reportez-vous à la section Ensembles de champs sur p. 86.](#)
- en utilisant une classe Java côté client, qui implémente une interface de fournisseur de modèle de données qui reçoit un ensemble de propriétés et le modèle de données d'entrée et renvoie une instance de modèle de données.
- en utilisant un composant de bibliothèque partagée coté serveur qui reçoit un ensemble de propriétés et le modèle de données d'entrée et renvoie un document de métadonnées.

La section Output Data Model définit la façon dont les propriétés d'un noeud affectent les champs qui traversent le noeud. Le modèle de données de sortie peut :

- ne pas modifier le modèle de données d'entrée
- modifier le modèle de données d'entrée
- remplacer le modèle de données d'entrée par un autre modèle de données

Par exemple, un noeud Trier n'affecte pas les propriétés mais les réorganise, un noeud Calculer modifie le modèle de données en ajoutant un nouveau champ et un noeud Agréger remplace complètement le modèle de données.

Si le modèle de données d'entrée est modifié, la définition peut ajouter de nouveaux champs ou modifier/supprimer des champs existants. Si le modèle de données est remplacé, seuls de nouveaux champs peuvent être ajoutés. Le fichier de spécifications prend en charge ces opérations de base (y compris la possibilité de créer de nouveaux champs dont le type est basé sur un champ d'entrée) ainsi que la possibilité d'itérer via l'ensemble de champs d'entrée ou une propriété de liste ou de clé représentant un groupe de champs dans l'ensemble de champs d'entrée.

### **Format**

Le format général de la section Output Data Model est le suivant, consultez les sections [Contrôle Sélecteur multi-champ sur p. 168](#) et [Contrôle Sélecteur d'un champ unique sur p. 177](#) pour des formats particuliers à ces cas.

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

Les attributs de modèle de données de sortie sont présentés dans le tableau suivant.

**Table 4-7**

*Attributs de modèle de données de sortie*

Attribut	Description
mode	Effet sur le modèle de données :  extend - ajoute un nouveau champ au modèle existant fixed - non modifié modify - modifie les champs existants (par exemple, les supprime ou les renomme) replace - remplace le modèle existant
libraryId	Le nom d'un conteneur côté serveur à partir duquel obtenir le modèle de données.

Les opérations de modèle de données sont celles permettant d'ajouter de nouveaux champs ou de modifier et de supprimer des champs existants. [Pour plus d'informations, reportez-vous à la section Opérations de modèle de données sur p. 80.](#)

### **Exemple**

Un élément `OutputDataModel` est compris dans l'exemple d'un fichier de spécifications. [Pour plus d'informations, reportez-vous à la section Exemple d'un fichier de spécifications sur p. 39.](#)

## **Constructors**

Les constructeurs définissent les objets provenant de l'exécution d'un noeud dans un flux ou du renvoi d'un objet vers le flux.

Pour des détails supplémentaires sur le codage de cette partie du fichier, consultez les sections commençant par [Utilisation des constructeurs sur p. 123.](#)

## **Fonctions communes**

Certaines fonctions peuvent être utilisées dans plusieurs sections du fichier de spécifications, à savoir :

- types de valeur
- chaînes évaluées
- opérations
- champs et métadonnées de champ
- ensembles de champs
- rôles
- Opérateurs logiques
- conditions

## **Types de valeur**

Les déclarations de types de valeur indiquent le type de valeur qu'une spécification de colonne, de propriété ou de type de propriété peut prendre.

### **Chaînes et chaînes codées**

Le format `valueType="string"` indique que la valeur est une chaîne texte. Une déclaration `valueType="encryptedString"` s'utilise pour une propriété associée à un champ dont le contenu doit être masqué lorsqu'il est saisi par un utilisateur, un champ de mot de passe, par exemple.

### **Noms de champ**

Là où une valeur prend la forme d'un nom de champ, utilisez le format `valueType="fieldName"`.

### **Expressions mathématiques, logiques et de date**

Si la valeur est mathématique (entière ou en double précision), logique (true/false) ou est une expression de date, définissez valueType sur integer, double, boolean ou date suivant le cas.

### **Propriétés énumérées**

Les propriétés énumérées sont contenues dans une section Enumeration qui suit immédiatement une déclaration valueType="enum". [Pour plus d'informations, reportez-vous à la section Propriétés énumérées sur p. 76.](#)

### **Déclarations de structure**

Une déclaration valueType="structure" indique une valeur composite contenant d'autres attributs nommés. Les attributs sont semblables aux propriétés mais ne peuvent être structurés ni saisis. [Pour plus d'informations, reportez-vous à la section Propriétés structurées sur p. 77.](#)

- **Indicateur saisi.** Indique si la propriété est une valeur simple ou un tableau de hachage où chaque valeur est du type de valeur spécifié.
- **Ensemble de valeurs.** Indique la façon dont l'ensemble des valeurs disponibles est déterminé.
- **Ensemble de clés.** Indique la façon dont l'ensemble de clés disponibles est déterminé pour les propriétés saisies. Ces informations permettent également de fournir des indications à l'interface utilisateur sur le type de contrôleur le plus approprié.

### **Connexions à la base de données**

Ce sont des chaînes de connexion permettant aux utilisateurs de se connecter à une base de données, par exemple user1@testdb. Les détails de la connexion devront d'abord avoir été définis pour la base de données. [Pour plus d'informations, reportez-vous à la section Contrôle Sélecteur de connexion à la base de données dans le chapitre 6 sur p. 165.](#)

## **Propriétés énumérées**

Une propriété énumérée peut prendre la valeur d'une liste de valeurs prédéfinie.

### **Format**

Le format des propriétés énumérées utilise une section Enumeration dans laquelle la liste des valeurs est définie comme suit :

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



où les attributs `PropertyType` sont :

- `id` : un identificateur unique de ce type de propriété.
- `valueType` indique que le type de propriété est énuméré.

et les attributs `Enum` sont :

- `value` (requis) est la valeur de propriété qui apparaît dans la liste des valeurs.
- `label` (requis) est le nom d'affichage de la valeur de propriété comme elle apparaît sur l'interface utilisateur.
- `labelKey` identifie l'étiquette à des fins de localisation.
- `description` est une description de la valeur énumérée.
- `descriptionKey` identifie la description à des fins de localisation.

### **Exemple**

```
<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>
```

### **Propriétés structurées**

Une propriété structurée est une propriété qui est utilisée dans une structure de grille comme une commande de tableau dans une boîte de dialogue.

#### **Format**

Le format des propriétés structurées utilise une section `Structure` dans laquelle cette structure est définie et qui est composée de plusieurs éléments `Attribute`, comme suit :

```
<PropertyTypes>
  <PropertyType id="identifieur" 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>
```

où les attributs d'élément `PropertyType` sont :

- `id` : un identificateur unique de ce type de propriété.

- `valueType` indique que le type de propriété est structuré.
- `isList` indique si la propriété est une liste de valeurs du type de valeur spécifié (`true`) ou une valeur simple (`false`).

et les attributs d'élément `Attribute` sont :

- `name` (requis) est l'identificateur de la colonne.
- `valueType` indique le type de valeur que peut prendre le contenu de cette colonne et est l'un des types suivants :
  - `string`
  - `encryptedString`
  - `integer`
  - `double`
  - `boolean`
  - `date`
  - `enum`
- `isList` indique si l'attribut est une liste de valeurs du type de valeur spécifié (`true`) ou une valeur simple (`false`). De cette manière, une propriété saisie peut être associée soit à un ensemble fixe d'attributs connus (par exemple, attributs booléens représentant les différentes opérations d'agrégation à exécuter sur un champ particulier), soit à une liste de valeurs (par exemple, en associant une liste de noms de champ à d'autres noms de champ).
- `label` (requis) est le nom d'affichage de la colonne comme il apparaît sur l'interface utilisateur.
- `labelKey` identifie l'étiquette à des fins de localisation.
- `defaultValue` est une valeur qui apparaît dans la colonne lorsque celle-ci est affichée.
- `description` est une description de la colonne.
- `descriptionKey` identifie la description à des fins de localisation.

### **Exemple - Contrôle Table**

Pour un exemple de l'utilisation des propriétés structurées dans un contrôle `Table`, consultez [Contrôle Table sur p. 181](#).

### **Exemples - Types de propriété saisie**

Le premier de ces exemples illustre l'utilisation d'un type de propriété saisie où chaque valeur associée est une structure représentant l'opération d'agrégation à appliquer à un champ à partir d'un ensemble fixe d'opérations :

```
<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>
```

```
</Structure>
</PropertyType>
```

Ainsi, une propriété déclarée pour utiliser le type de propriété `aggregateOps` pourrait être comme suit :

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

Dans ce cas, la propriété se compose de plusieurs valeurs, chacune ayant une clé différente. Par exemple, la clé `name` représente le nom d'un champ (MIN, MAX et ainsi de suite).

Dans l'exemple suivant de type de propriété saisie, chaque valeur associée est une structure contenant un attribut unique. Dans ce cas, l'attribut est une liste d'expressions double précision représentant des multiplicateurs à appliquer à un champ :

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

Une propriété déclarée pour utiliser le type de propriété `multiplierOps` pourrait être comme suit :

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

### **Valeurs par défaut**

L'élément `DefaultValue` permet de spécifier un répertoire ou un fichier temporaire de serveur, ou les deux. Ces derniers sont créés pour stocker un objet de sortie de modèle ou de document.

#### **Format**

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

où `basename` (requis) est le nom du fichier ou du répertoire temporaire.

#### **Exemple**

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

### **Chaînes évaluées**

Certaines chaînes déclarées dans le fichier de spécifications peuvent inclure des références aux noms de propriétés. Ces chaînes sont connues sous le nom de chaînes évaluées.

La syntaxe d'une référence de propriété est la suivante :

```
"${property_name}"
```

Lors de l'accès à une chaîne évaluée, toute référence de propriété est remplacée par la valeur de la propriété référencée. Si la propriété n'existe pas, une erreur se produit. Par exemple, lors de l'ajout d'un nouveau champ, il est possible qu'une propriété soit nommée `my_new_field` dans la définition du noeud et qu'une commande de la section User Interface permette à l'utilisateur de modifier la valeur de cette propriété.

### **Exemple**

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

## **Operations**

Certaines sections du fichier de spécifications prennent en charge des opérations comme l'ajout de champs, la création de composants et l'initialisation des propriétés. Les sections prenant en charge ces opérations sont :

- modèle de données de sortie (noeuds source et noeuds d'exécution)
- modèle de données d'entrée et de sortie (composants)
- création d'objet de sortie (noeuds de création de modèle et de document)
- création d'applicateur de modèle (sorties de modèle)

Les opérations sont réparties entre les types suivants :

- opérations de modèle de données : AddField, ChangeField, RemoveField
- itération : ForEach

### **Opérations de modèle de données**

Les opérations possibles sur un modèle de données sont :

- l'ajout d'un nouveau champ à un modèle de données existant
- la modification d'un champ existant dans un modèle de données
- la suppression d'un champ d'un modèle de données

### **Ajouter un champ**

L'élément AddField vous permet d'ajouter un nouveau champ à un modèle de données existant.

#### **Format**

```
<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="rôle" tag="propensity_type" value="value" >
  <Range min="min_value" max="max_value" />
```

</AddField>

Les attributs de AddField sont les suivants.

Table 4-8

Attributs AddField

Attribut	Description
prefix	Un préfixe à ajouter au nom du champ, par exemple pour signaler un champ de sortie de modèle.
name	(requis) Le nom du champ à ajouter. Il peut s'agir soit d'une chaîne codée en dur (par ex. field8) soit d'une chaîne évaluée (par ex. \${target} ) qui fait référence à un champ. <a href="#">Pour plus d'informations, reportez-vous à la section Chaînes évaluées sur p. 79.</a>
direction	La rôle du champ, par exemple si le champ est un champ d'entrée ou un champ cible. in, out, both, partition ou none. <a href="#">Pour plus d'informations, reportez-vous à la section Définition du rôle du champ dans le chapitre 4 dans IBM SPSS Modeler 14.2 Noeuds source, exécution et de sortie.</a>
directionRef	Indique que la direction du champ doit être calculée à partir de la direction du champ identifié par une chaîne évaluée (par ex. \${field1}) qui fait référence à un champ. <a href="#">Pour plus d'informations, reportez-vous à la section Chaînes évaluées sur p. 79.</a>
fieldRef	Indique que toutes les valeurs référencées (directionRef, missingValuesRef, storageRef et typeRef) doivent être calculées à partir des valeurs correspondantes du champ identifié par une chaîne évaluée (\${field1} par exemple) qui référence un champ. <a href="#">Pour plus d'informations, reportez-vous à la section Chaînes évaluées sur p. 79.</a>
group	Indique que le champ est membre d'un groupe de champs. <a href="#">Pour plus d'informations, reportez-vous à la section Champs de modèle dans le chapitre 5 sur p. 104.</a>
label	Une étiquette pour le champ à ajouter.
missingValuesRef	Indique que la façon dont les valeurs manquantes sont gérées doit être calculée à partir de la spécification des valeurs manquantes du champ identifié par une chaîne évaluée (par ex. \${field1}) qui fait référence à un champ. <a href="#">Pour plus d'informations, reportez-vous à la section Chaînes évaluées sur p. 79.</a>
storage	Le type de stockage de données de la valeur du champ - integer, real, string, date, time, timestamp ou unknown.
storageRef	Indique que le type de stockage doit être calculé à partir du type de stockage du champ identifié par une chaîne évaluée (par ex. \${field1}) qui fait référence à un champ. <a href="#">Pour plus d'informations, reportez-vous à la section Chaînes évaluées sur p. 79.</a>
targetField	Pour un champ de sortie de modèle, indique le champ cible à partir duquel sont calculées les données de ce nouveau champ. Il peut s'agir soit d'une chaîne codée en dur (par ex. field8) soit d'une chaîne évaluée (par ex. \${target} ) qui fait référence à un champ. <a href="#">Pour plus d'informations, reportez-vous à la section Chaînes évaluées sur p. 79.</a>
type	Le type de de données du champ - auto, range, discrete, set, orderedSet, flag ou typeless. <a href="#">Pour plus d'informations, reportez-vous à la section Niveaux de mesure dans le chapitre 4 dans IBM SPSS Modeler 14.2 Noeuds source, exécution et de sortie.</a>
typeRef	Indique que le type de données doit être calculé à partir du type de données du champ identifié par une chaîne évaluée (par ex. \${field1}) qui fait référence à un champ. <a href="#">Pour plus d'informations, reportez-vous à la section Chaînes évaluées sur p. 79.</a>
role	Le genre de données contenues dans un champ de sortie de modèle - unknown, predictedValue, predictedDisplayValue, probability, residual, standardError, entityId, entityAffinity, upperConfidenceLimit, lowerConfidenceLimit, propensity, value ou supplementary. <a href="#">Pour plus d'informations, reportez-vous à la section Rôles sur p. 87.</a>
tag	Utilisé uniquement si role a la valeur propensity ; indique le type de propension, et est soit RAW soit ADJUSTED.
value	Indique que les valeurs contenues par le nouveau champ doivent être calculées à partir du champ identifié par une chaîne évaluée (par ex. \${field1}) qui fait référence à un champ. <a href="#">Pour plus d'informations, reportez-vous à la section Chaînes évaluées sur p. 79.</a>

Les attributs pour **Range** sont les suivants.

Table 4-9

*Attributs de Range (intervalle)*

Attribut	Description
min	La valeur minimale que peut accepter le champ.
max	La valeur maximale que peut accepter le champ.

### **Exemples**

L'exemple suivant ajoute un champ de type chaîne nommé field8 :

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

L'exemple suivant explique comment utiliser une référence à un nom de propriété lors de l'ajout d'un champ. Le champ est ajouté ici avec un nom qui correspond à la valeur de la propriété pré-définie prop1 :

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

Dans l'exemple suivant, si le champ cible est nommé field1, le modèle crée un champ de sortie nommé \$\$-field1 qui contient la valeur prédite pour field1 :

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

Le prochain exemple ajoute un champ de sortie de modèle qui contient un score de probabilité entre 0,0 et 1,0 :

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

Dans ce dernier exemple, pour chaque champ de sortie de modèle, un champ de sortie est ajouté qui contient un score de probabilité compris entre 0,0 et 1,0 et dont la valeur est issue de celle de la variable fieldValue :

```
<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>
```

[Pour plus d'informations, reportez-vous à la section Chaînes évaluées sur p. 79.](#)

### **Modifier un champ**

L'élément **ChangeField** permet de modifier un champ existant dans un modèle de données.

**Format**

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

Les attributs de ChangeField sont les suivants.

Table 4-10  
Attributs de ChangeField

Attribut	Description
name	(requis) Le nom du champ à modifier.
fieldRef	Une valeur de référence pour le champ.
direction	La rôle du champ, par exemple si le champ est un champ d'entrée ou un champ cible. in, out, both, partition ou none. <a href="#">Pour plus d'informations, reportez-vous à la section Définition du rôle du champ dans le chapitre 4 dans IBM SPSS Modeler 14.2 Noeuds source, exécution et de sortie.</a>
storage	Le type de stockage de données de la valeur du champ - integer, real, string, date, time, timestamp ou unknown.
type	Le type de de données du champ - auto, range, discrete, set, orderedSet, flag ou typeless. <a href="#">Pour plus d'informations, reportez-vous à la section Niveaux de mesure dans le chapitre 4 dans IBM SPSS Modeler 14.2 Noeuds source, exécution et de sortie.</a>

Les attributs pour Range sont les suivants.

Table 4-11  
Attributs de Range (intervalle)

Attribut	Description
min	La valeur minimale que peut accepter le champ.
max	La valeur maximale que peut accepter le champ.

**Supprimer un champ**

L'élément RemoveField permet de supprimer un champ du modèle de données.

**Format**

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

où fieldRef est une valeur de référence pour le champ.

**Itération avec l'élément ForEach**

A certains endroits, il est utile de pouvoir effectuer la même opération plusieurs fois pour traiter chacun des ensembles de valeurs. Le fichier de spécifications prend en charge un itérateur ForEach simple qui associe une propriété temporaire à chaque valeur dans l'ensemble fourni. La boucle ForEach peut être définie pour itérer de l'une des façons suivantes :

- entre deux valeurs entières avec une taille de pas (facultatif)

- sur des valeurs d'une propriété de liste
- sur des clés dans une propriété saisie
- sur des champs dans un groupe de champs

### **Format**

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

où :

var (requis) indique le champ contenant les valeurs auxquelles l'itération s'applique.

from et to indiquent des entiers (ou des expressions qui représentent des entiers) qui correspondent aux limites supérieures et inférieures de l'itération avec l'attribut facultatif step qui indique une taille de pas d'entier.

inFields, inFieldValues et inProperty sont des alternatives au format from/to/step :

- inFields indique un ensemble de champs sur lequel effectuer une itération et est l'un des ensembles suivants :
  - inputs - les champs d'entrée du noeud
  - outputs - les champs de sortie du noeud
  - modelInput - les champs d'entrée spécifiés dans la signature du modèle
  - modelOutput - les champs de sortie spécifiés dans la signature du modèle
- inFieldValues indique un nom de champ (ou une propriété qui représente un nom de champ) et itère via les valeurs des métadonnées de ce champ
- inProperty indique le nom d'une propriété sur laquelle effectuer l'itération

L'opération de modèle de données qui peut être spécifiée dans un élément ForEach est l'un des éléments AddField, ChangeField ou RemoveField. [Pour plus d'informations, reportez-vous à la section Opérations de modèle de données sur p. 80.](#) Les éléments ForEach peuvent également être imbriqués.

### **Exemples**

L'exemple suivant effectue une opération dix fois de suite :

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

L'exemple suivant effectue une opération le nombre de fois indiqué par une propriété d'entier :

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



Dans l'exemple suivant, le traitement itère via les valeurs des champs de sortie du noeud :

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

L'exemple suivant itère via les valeurs des métadonnées du champ identifié par `#{field}` :

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

L'exemple suivant itère via les valeurs d'une propriété de liste :

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

L'exemple suivant itère via les valeurs clé d'une propriété saisie :

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

## **Champs et métadonnées de champ**

Les noeuds, les modèles et les sources de données agissent comme **fournisseurs de modèle de données** : ils peuvent définir les métadonnées de champ accessibles à partir d'autres objets.

Les fournisseurs de modèle de données ont un modèle de données d'entrée et un modèle de données de sortie. Un modèle de données de sortie peut être défini en termes du modèle de données d'entrée, par exemple lors de l'extension d'un modèle d'entrée en ajoutant un champ ou lors de la modification d'un modèle existant.

Chacun de ces objets a des exigences qui diffèrent légèrement.

**Noeuds.** Le modèle de données d'entrée peut être référencé mais n'est pas modifiable. Le modèle de données de sortie peut être basé sur le modèle de données d'entrée ou peut le remplacer. Le modèle de données de sortie est recalculé à chaque fois que les propriétés du noeud ou le modèle de données d'entrée sont modifiés. Le modèle de données de sortie d'un noeud applicateur de modèle peut également référencer le modèle de données de sortie du composant du modèle.

**Modèles :** Par défaut, les modèles de données d'entrée et de sortie (la signature des modèles) sont basés sur les paramètres des champs d'entrée et de sortie utilisés lors de la création du modèle. Idéalement, le processus de création du modèle renvoie un fichier de métadonnées qui définit les champs d'entrée requis et les champs de sortie générés. Une fois définie, la signature du modèle ne peut pas être modifiée. Cependant, les propriétés dans un noeud applicateur de modèle peuvent modifier la sortie du modèle de données du noeud applicateur. Par exemple, ces propriétés peuvent définir si un ID de classe est renvoyé sous la forme d'une chaîne ou d'un entier ou le nombre d'ID de séquence à générer. De plus, la signature de modèle indique généralement les

sorties avec une rôle de champ (direction) défini sur "out", alors que le noeud est susceptible de le générer avec la valeur "in".

**Sources de données.** Les sources de données utilisées dans les noeuds de lecture de données peuvent indiquer un modèle de données de sortie. Le modèle de données d'entrée est toujours vide.

## Ensembles de champs

Un ensemble de champs peut être utilisé à plusieurs endroits pour sélectionner un sous-ensemble de champs dans un fournisseur de modèle de données. Le fournisseur de modèle de données peut être l'objet contenant ou un conteneur de l'objet contenant. L'état initial d'un filtre de champ peut être d'inclure tous les champs disponibles puis d'exclure les types de champs spécifiques ou de commencer avec un ensemble de champs vide et d'inclure les champs requis ou d'ajouter de nouveaux champs.

L'exemple suivant indique la façon dont un noeud d'extension peut définir le modèle de données de sortie. Les champs-clés sont indiqués par une propriété de liste nommée `keys` et sont suivis d'un champ facultatif de comptage des enregistrements pouvant être généré et dont le nom est également indiqué par une propriété.

```
<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>
```

## Ensembles de champs et création de modèle

L'exemple suivant indique la façon dont un applicateur de modèle peut utiliser les informations du composant de modèle créé auparavant pour générer ses champs de sortie :

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

Les éléments `AddField` et `ForEach` indiquent tous deux un fournisseur de modèle de données et précisent lequel des modèles de données d'entrée ou de sortie doit être utilisé. Ils fournissent un mécanisme qui définit un ensemble (ou sous-ensemble) de champs du fournisseur de modèle de données. Le fournisseur par défaut est `this` qui représente l'élément contenant (non pas l'objet créé) avec l'ensemble de champs d'entrée utilisé par défaut. Si aucun ensemble de champs n'est spécifié, tous les champs disponibles sont utilisés.

Les ensembles de champs peuvent être basés sur le stockage, le type, le rôle du champ ou les noms. Lorsqu'ils sont basés sur des noms, il est nécessaire d'avoir une référence à une propriété de liste. L'ensemble de champs peut être plein (par défaut) ou vide ; le premier permet d'exclure des champs et le second permet d'inclure des champs. Plusieurs valeurs peuvent être spécifiées

pour chacun des filtres individuels et ces valeurs agissent comme opérateurs “intersection” ou “et”, par exemple :

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

Ceci commence par un ensemble de champs vides (spécifié par include="none") et inclut ensuite les champs dont le rôle de champ (direction) est défini sur "in" et le stockage de chaînes.

Un autre exemple est le suivant :

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

Ceci comprend tous les champs disponibles (spécifiés par l’attribut include="all" qui est le comportement par défaut) puis exclut tout champ de type typeless. Cela comprend les champs direction définis sur "in" ou "both".

Plusieurs filtres peuvent également être spécifiés et ceux-ci agissent comme opérateurs “union” ou “ou”, par exemple :

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

Ceci exclut les champs qui sont discrets avec stockage de réel ou discrets avec stockage d’entier.

Veillez noter que lorsque des champs sont inclus dans un ensemble initial de champs vides, l’ordre des instructions include ne modifie généralement pas l’ordre dans lequel les champs sont inclus. C’est-à-dire que les champs du fournisseur d’ensemble de champs sont évalués dans leur ordre naturel par rapport à chaque condition afin de déterminer s’ils doivent être inclus dans l’ensemble de champs.

## Rôles

Un rôle décrit le type de données contenu dans un champ de sortie de modèle de données. Un rôle peut être indiqué par un élément AddField et testé par un élément Condition.

Les rôles possibles sont les suivants.

Table 4-12  
Rôles de sortie de modèle

Rôle	Signification
unknown	Aucun rôle spécifié (ne devrait pas se produire).
predictedValue	Ce champ contient la valeur prédite du champ cible.
probability	La probabilité ou la confiance de la prédiction.
residual	La valeur résiduelle.

Rôle	Signification
standardError	L'erreur standard de la prédiction.
entityId	L'ID d'entité : représente généralement l'ID de classe d'un modèle de classe.
entityAffinity	L'affinité d'entité : représente généralement la distance depuis le centre de la classe dans un modèle.
upperConfidenceLimit	La limite supérieure de confiance de la prédiction.
lowerConfidenceLimit	La limite inférieure de confiance de la prédiction.
propensity	Le score de propension. Un attribut tag supplémentaire indique s'il s'agit d'une propension brute ou ajustée.
value	Utilisée pour représenter une valeur de modèles qui est associée à une autre sortie (voir ci-dessous).
supplementary	Informations générées par le modèle qui ne sont pas couvertes par les autres rôles.

Comme exemple de value, le modèle de détection d'anomalies génère des groupes de champs où chaque groupe est constitué de deux champs, l'un représentant un nom de champ et l'autre mesurant les anomalies de ce champ. Dans ce cas, value serait le nom du champ.

## Opérateurs logiques

Plusieurs éléments peuvent utiliser les opérateurs logiques And, Or et Not afin d'indiquer plusieurs types de traitements, par exemple lors du paramétrage des conditions composées (consultez [Conditions composées sur p. 92](#)).

### Format

Le format de l'élément And est le suivant : Le format des éléments Or et Not est presque identique, les seules différences sont les balises encadrantes <Or>...</Or> et <Not>...</Not>.

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

L'élément Condition indique une condition à tester. [Pour plus d'informations, reportez-vous à la section Conditions sur p. 89.](#)

Veuillez noter que les éléments enfant And, Or ou Not peuvent être imbriqués.

## Conditions

Le comportement de certains objets peut être modifié à l'aide des conditions qui sont spécifiées par les éléments Condition (l'équivalent des instructions IF). Par exemple, un noeud exécuté par une commande peut ajouter une condition aux informations d'exécution, comme n'inclure qu'une option spécifique si une propriété a une valeur particulière. De la même façon, une commande de propriété de l'interface utilisateur peut être activée ou visible uniquement si une autre commande a une valeur précise.

Les conditions peuvent être simples ou composées. Une **condition simple** est composée de :

- une source de valeurs (une propriété ou une commande)
- un test
- une valeur test facultative

Une **condition composée** permet aux autres conditions d'être combinées pour former des conditions logiques complexes. Les conditions composées incluent l'utilisation de :

- And
- Or
- Not

### Format

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

où :

container spécifie le nom d'un conteneur spécifique dont la valeur doit être testée par la condition.

control indique une commande de propriété dont la valeur doit être testée par la condition. *prop\_name* est la valeur de l'attribut property de l'élément dans lequel la commande est définie (par exemple, dans un panneau de propriétés sur un onglet de boîte de dialogue).

property indique une propriété dont la valeur doit être testée par la condition. *name* est la valeur de l'attribut name de l'élément Property dans lequel la propriété est définie.

op est l'opérateur de conditions. [Pour plus d'informations, reportez-vous à la section Opérateurs de conditions sur p. 90.](#)

value est la valeur spécifique que la condition doit tester.

### Exemples

Pour des exemples de paramétrage des conditions, consultez [Conditions simples sur p. 92](#) et [Conditions composées sur p. 92](#).

### Opérateurs de conditions

Un ensemble d'opérateurs est disponible et prend en charge la plupart des conditions.

Table 4-13

Tests pris en charge par toutes les valeurs

Opérateur	Valeur	Description
equals	<i>value</i>	True si la propriété est égale à la valeur fournie (respect de la casse pour les valeurs de chaîne).
notEquals	<i>value</i>	True si la propriété n'est pas égale à la valeur fournie (respect de la casse pour les valeurs de chaîne).
in	<i>liste des valeurs</i>	True si la propriété se trouve dans la liste de valeurs fournie.

Table 4-14

Tests pris en charge pour des valeurs numériques

Opérateur	Valeur	Description
lessThan	<i>nombre</i>	True si la propriété est inférieure au chiffre fourni.
lessOrEquals	<i>nombre</i>	True si la propriété est inférieure ou égale au chiffre fourni.
greaterThan	<i>nombre</i>	True si la propriété est supérieure au chiffre fourni.
greaterOrEquals	<i>nombre</i>	True si la propriété est supérieure ou égale au chiffre fourni.

Table 4-15

Tests pris en charge pour des valeurs de chaîne

Opérateur	Valeur	Description
isEmpty	-	True si la propriété a une chaîne de longueur zéro.
isNotEmpty	-	True si la propriété a une chaîne de longueur autre que zéro.
startsWith	<i>chaîne</i>	True si la propriété commence par la chaîne fournie (respect de la casse).
startsWithIgnoreCase	<i>chaîne</i>	True si la propriété commence par la chaîne fournie et ignore la casse.
endsWith	<i>chaîne</i>	True si la propriété se termine par la chaîne fournie (respect de la casse).
endsWithIgnoreCase	<i>chaîne</i>	True si la propriété se termine par la chaîne fournie et ignore la casse.
equalsIgnoreCase	<i>chaîne</i>	True si la propriété est égale à la chaîne fournie et ignore la casse.
hasSubstring	<i>chaîne</i>	True si la propriété contient la chaîne fournie (respect de la casse).
hasSubstringIgnoreCase	<i>chaîne</i>	True si la propriété contient la chaîne fournie et ignore la casse.
isSubstring	<i>chaîne</i>	True si la propriété est une sous-chaîne de la chaîne fournie (respect de la casse).
isSubstringIgnoreCase	<i>chaîne</i>	True si la propriété est une sous-chaîne de la chaîne fournie et ignore la casse.

Table 4-16  
Tests pris en charge pour les propriétés de liste

Opérateur	Valeur	Description
isEmpty	-	True si le nombre d'éléments de la liste est 0.
isNotEmpty	-	True si le nombre d'éléments de la liste n'est pas 0.
countEquals	<i>nombre</i>	True si le nombre d'éléments de la liste est égal à la valeur fournie.
countLessThan	<i>nombre</i>	True si le nombre d'éléments de la liste est inférieur à la valeur fournie.
countLessOrEquals	<i>nombre</i>	True si le nombre d'éléments de la liste est inférieur ou égal au chiffre fourni.
countGreaterThan	<i>nombre</i>	True si le nombre d'éléments de la liste est supérieur au chiffre fourni.
countGreaterOrEquals	<i>nombre</i>	True si le nombre d'éléments de la liste est supérieur ou égal au chiffre fourni.
contains	<i>value</i>	True si l'élément fourni se trouve dans la liste.

Table 4-17  
Tests pris en charge pour les propriétés de champ

Opérateur	Description
storageEquals	True si le stockage est égal à la valeur fournie.
typeEquals	True si le type est égal à la valeur fournie.
directionEquals	True si le rôle du champ (direction) est égal à la valeur fournie.
isMeasureDiscrete	True si le type de données de champ est discrete (c'est-à-dire qu'il peut uniquement être set, flag ou orderedSet).
isMeasureContinuous	True si le type de données de champ est range.
isMeasureTypeless	True si le type de données de champ est typeless.
isMeasureUnknown	True si le type de données de champ est unknown.
isStorageString	True si le type de stockage de champ est string.
isStorageNumeric	True si le type de stockage de champ est numeric.
isStorageDatetime	True si le type de stockage de champ est datetime.
isStorageUnknown	True si le type de stockage de champ est unknown.
isModelOutput	True si le champ est un champ de sortie de modèle.
modelOutputRoleEquals	True si le rôle du champ est l'un des rôles valides présentés dans le tableau suivant.
modelOutputTargetFieldEquals	True si le champ cible est égal à la valeur spécifiée (chaîne).
modelOutputHasValue	True si le champ est un champ de sortie de modèle et a une valeur qui lui est associée.
modelOutputTagEquals	True si la balise est égale à la valeur spécifiée (chaîne).

Les opérateurs de condition qui prennent en charge les propriétés saisies sont les suivants :

- isEmpty
- isEmpty
- countEquals
- countLessThan

- countLessOrEquals
- countGreaterThan
- countGreaterOrEquals
- contains

### **Conditions simples**

Une condition simple est composée d'une source de valeur initiale à tester (un nom de propriété ou de contrôleur ou une expression évaluée), du test à effectuer et facultativement d'une valeur à tester.

#### **Exemples**

L'exemple suivant est vrai si la propriété booléenne nommée `values_grouped` est vraie :

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

L'exemple suivant est vrai si une commande nommée `values_grouped` et qui affiche des valeurs booléennes a été vérifiée :

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

L'exemple suivant est vrai si une propriété de liste nommée `plot_fields` contient au moins une valeur :

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

L'exemple suivant est vrai si une propriété de liste nommée `input_fields` ne contient que des valeurs instanciées :

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

L'exemple suivant est vrai si une propriété de liste nommée `input_fields` contient au moins une valeur représentant un champ non instancié :

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

### **Conditions composées**

Les groupes de conditions simples peuvent être combinés à l'aide d'opérateurs logiques.

#### **Exemples**

L'exemple suivant est vrai si la propriété booléenne `values_grouped` est vraie et que `group_fields` contient au moins une valeur :

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



L'exemple suivant est vrai si la propriété booléenne `values_grouped` est vraie ou si `group_fields` contient au moins une valeur :

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

L'exemple suivant est vrai si `group_fields` contient au moins une valeur :

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

Les conditions composées peuvent être imbriquées pour produire toute combinaison de conditions.

## Utilisation de noeuds CLEF dans les scripts

Vous pouvez faire référence à un noeud CLEF dans un script, à l'aide de l'attribut `scriptName` de l'élément `Node`. De la même manière, vous pouvez faire référence à une propriété du noeud dans un script à l'aide de l'attribut `scriptName` de l'élément `Property`.

L'attribut `scriptName` est facultatif dans les deux cas, c'est pourquoi nous conseillons d'utiliser l'attribut pour éviter les conflits de noms entre les extensions ou les propriétés.

Si vous omettez le nom du script dans une définition de noeud, un script peut faire référence au noeud avec la valeur de l'attribut `id` précédée du nom de l'extension. Par exemple, pour une extension nommée `myext` qui définit un noeud de lecture de données avec l'ID `import`, un script référencera le noeud sous la forme `myextimport`.

Si vous omettez le nom du script dans une définition de propriété, un script peut référencer la propriété avec la valeur de l'attribut `name`.

Pour plus d'informations, reportez-vous au guide *IBM® SPSS® Modeler Référence de génération de scripts et d'automatisation*.

### Exemple - Editer et exécuter un noeud

L'exemple suivant décrit la manière d'utiliser un script pour automatiser les tâches d'édition et d'exécution de l'exemple de noeud de lecture de données présenté à la rubrique [Noeud de lecture de données \(Apache Log Reader\)](#) sur p. 33.

Dans le fichier de spécifications du noeud Apache Log Reader, les spécifications du noeud commencent comme suit :

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

Dans le script, vous pourriez référencer le noeud et la propriété comme suit :

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

où *installation\_directory* est le répertoire dans lequel IBM® SPSS® Modeler est installé.

Exécution de ce script :

- crée le noeud de lecture de données
- spécifie *combined\_log\_format.txt* comme le fichier journal Apache à lire
- crée un noeud Table
- connecte le noeud de lecture de données au noeud Table
- exécute le noeud Table

### **Exemple - Propriétés saisies**

Les propriétés saisies prennent en charge la syntaxe de génération de script standard. Par exemple, la structure présentée dans le premier exemple de types de propriété saisie à la rubrique [Propriétés structurées sur p. 77](#). pourrait être définie dans un script sous la forme :

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

Un attribut unique peut être modifié de la façon suivante :

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

## **Maintenance de la compatibilité descendante**

Lors de la planification de mises à jour d'une extension existante, vérifiez que la compatibilité avec la version précédente de cette extension est maintenue. Certaines modifications n'auront pas d'effets gênants, d'autres représentent un risque important et d'autres encore annulent la compatibilité et doivent être évités.

### **Modifications sans risque**

Les modifications suivantes n'auront pas d'incidence sur la compatibilité descendante :

- ajout de nouveaux éléments Node, ModelOutput, DocumentOutput ou InteractiveModelBuilder
- ajout de nouvelles définitions Property et de leurs nouvelles commandes associées à ces éléments
- ajout de nouveaux conteneurs à ces éléments\*
- ajout de nouvelles valeurs à une propriété d'énumération existante

\* Veuillez noter que tout code utilisant ces nouveaux conteneurs doit permettre que ces conteneurs soient vides d'objets créés à l'aide d'une version précédente de l'extension.

**Modifications avec risque important**

Les modifications apportées à des déclarations existantes engendrent un risque important d'annulation de la compatibilité. Ces modifications doivent être testées attentivement avant d'être utilisées.

**Modifications à éviter**

Les modifications suivantes annulent la compatibilité et doivent être évitées :

- modification de la valeur des attributs `id` ou `providerTag` dans l'élément `ExtensionDetail`
- modification de la valeur de l'attribut `id` dans un élément `Node`, `ModelOutput`, `DocumentOutput` ou `InteractiveModelBuilder`.
- suppression d'un élément `Node`, `ModelOutput`, `DocumentOutput` ou `InteractiveModelBuilder` de l'extension
- modification de la valeur de l'attribut `valueType` d'un élément `Property` ou `PropertyType`.

# ***Création de modèles et de documents***

## ***Introduction à la création de modèles et de documents***

Les modules standard IBM® SPSS® Modeler comprennent des noeuds qui permettent aux utilisateurs de générer (ou “créer”) un grand nombre de modèles et de graphiques. CLEF vous offre la possibilité de définir des noeuds supplémentaires afin de créer d’autres modèles et documents (graphiques et rapports) non fournis.

Lors de la définition de noeuds de création de modèles ou de création de documents, vous devez aussi définir les objets qui sont produits lors de l’exécution de ces noeuds. Pour ceci vous utilisez des éléments appelés “constructeurs”.

Les sections suivantes décrivent ce processus en détails.

## ***Modèles***

Un **modèle** est un ensemble de règles, une formule ou une équation qui peut être utilisé pour prévoir un résultat en fonction d’un ensemble de champs d’entrée. L’aptitude à prévoir un résultat est l’objectif central des analyses prédictives. Dans IBM® SPSS® Modeler, vous parviendrez en :

- générant un modèle à partir de données existantes
- appliquant le modèle généré aux données pour faire les prévisions

Le processus de génération d’un modèle est également appelé “création” de modèle, et dans SPSS Modeler, vous réalisez cette opération au moyen d’un noeud de modélisation. Dans CLEF, les noeuds de modélisation sont désignés **noeuds de création de modèle (model builder)**, un nom dérivé de la syntaxe de la déclaration SQL utilisée pour les définir. [Pour plus d’informations, reportez-vous à la section Noeuds de création de modèles dans le chapitre 2 sur p. 11.](#)

Le processus d’application d’un modèle aux données est appelé “scoring des données”. Il vous permet d’utiliser les informations que vous avez pu rassembler lors de la création de modèles pour générer des prévisions pour de nouveaux enregistrements. Dans SPSS Modeler, vous ajoutez l’icône d’un modèle généré à l’espace de travail de flux. L’icône se présente comme une pépite d’or (nugget), de sorte que le modèle généré est désigné dans SPSS Modeler par le terme “nugget de modèle”. Dans CLEF, un nugget de modèle sous l’onglet Modèles du panneau du gestionnaire est appelé **objet de sortie de modèle**, et lorsqu’il est ajouté à l’espace de travail, on l’appelle **noeud applicateur de modèle**. [Pour plus d’informations, reportez-vous à la section Noeuds applicateurs de modèle dans le chapitre 2 sur p. 13.](#)

## Documents

Dans certains cas, vous souhaitez générer un objet autre qu'un modèle, par exemple une sortie graphique ou un rapport. Dans IBM® SPSS® Modeler, ces objets sont appelés **documents** et sont générés au moyen d'un **noeud de création de document**. [Pour plus d'informations, reportez-vous à la section Noeuds de création de document dans le chapitre 2 sur p. 12.](#)

## Constructors

Les constructeurs définissent les objets produits suite à l'exécution d'un noeud dans un flux ou à la génération d'un objet dans le flux.

Les constructeurs peuvent être définis pour les éléments suivants :

- Noeud de création de modèles
- Noeud de création de documents
- Noeud applicateur de modèle
- Objet de sortie de modèle

Dans le cas d'un noeud de **création de modèle** ou de **création de document**, un constructeur permet à ces noeuds de définir la manière dont l'objet de sortie est généré lors de l'exécution du noeud. Une définition d'objet de sortie peut comprendre différentes propriétés et composants et la section Constructors définit la manière dont ces éléments sont initialisés ou créés à partir de l'objet généré par l'exécution.

Pour un noeud **applicateur de modèle**, un constructeur définit le type d'objet que le noeud peut générer dans le flux ou sous l'onglet Modèles.

Dans le cas où les constructeurs sont définis pour un **objet de sortie de modèle**, ils peuvent :

- Spécifier quel noeud applicateur de modèle créer lorsque l'objet de sortie du modèle est placé dans l'espace de travail du flux
- Générer un noeud de création de modèles avec les paramètres utilisés pour créer l'objet de sortie de modèle

[Pour plus d'informations, reportez-vous à la section Utilisation des constructeurs sur p. 123.](#)

## Création de modèles

Lors de la spécification d'un noeud à partir duquel un modèle peut être généré (c'est-à-dire un noeud de création de modèles), vous devez définir la manière dont le noeud communique avec le composant Créateur de modèles de IBM® SPSS® Modeler, c'est à dire le processus qui crée réellement le modèle. Vous effectuez cette opération dans la définition d'un élément **node** dans le fichier de spécifications.

Sauf indication contraire, la création de modèle débute dès que l'utilisateur final clique sur le bouton Exécuter dans la boîte de dialogue du noeud de création de modèles. Toutefois, il est également possible de définir un **modèle interactif** permettant à l'utilisateur final d'affiner ou de modifier les valeurs de données après avoir cliqué sur Exécuter, mais avant que le modèle ne

soit réellement créé. La création de modèles interactifs requiert également l'inclusion d'éléments spécifiques par lesquels l'interactivité est définie. [Pour plus d'informations, reportez-vous à la section Création de modèles interactifs sur p. 108.](#)

Lors de la définition d'un noeud de création de modèles, l'élément `Node` doit comprendre :

- Un attribut `type="modelBuilder"`
- Un élément enfant `ModelBuilder`
- Un élément enfant `Constructors` contenant un élément `CreateModelOutput` (consultez la rubrique [Utilisation des constructeurs sur p. 123](#))

Pour le format d'un élément de spécification `Node`, consultez la rubrique [Noeud sur p. 61](#).

*Remarque* : Dans les définitions des éléments des sections suivantes (normalement identifiées par l'en-tête **Format**), les attributs d'éléments et d'éléments enfants sont facultatifs sauf s'ils sont indiqués comme "(required)." Pour obtenir la syntaxe complète des éléments, consultez la rubrique l'annexe A, *Schéma XML CLEF* sur p. 252.

Pour la création de modèles, l'extension doit également disposer d'un élément `ModelOutput` pour décrire le modèle généré (consultez la rubrique [Sortie du modèle sur p. 107](#)). L'élément `ModelOutput` doit inclure un élément enfant `Constructors` contenant une définition `CreateModelApplier`. [Pour plus d'informations, reportez-vous à la section Création d'apporteur de modèle sur p. 126.](#)

## Concepteur de modèle

L'élément `ModelBuilder` définit le comportement d'un noeud de création de modèles. On utilise pour ce faire les attributs de l'élément et un ou plusieurs éléments enfants.

### Format

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

où :

- `allowNoInputs` et `allowNoOutputs` doivent être utilisés explicitement dans le cas où vous souhaitez créer un modèle ne comportant pas de champ de saisie ou de champ de sortie, respectivement.
- `nullifyBlanks`, s'il est défini sur `false`, désactive la fonction de remplacement des valeurs vides par des valeurs nulles (représentées par `$null$`) dans les données communiquées au composant Créateur de modèles (`ModelBuilder`) de IBM® SPSS® Modeler. Par défaut, les champs vides sont remplacés par des valeurs nulles, mais vous souhaitez peut-être désactiver

cette fonction, par exemple si votre algorithme doit traiter les champs vides différemment des champs nuls.

- `miningFunctions` (requis) identifie la ou les fonctions d'exploration des données exécutées par le modèle.

Table 5-1  
Fonctions de data mining

Fonction	Description
Classification	Prévoit une valeur discrète (c'est-à-dire un type de données <code>set</code> , <code>flag</code> ou <code>orderedSet</code> ensemble ordonné) d'un attribut cible inconnu provenant d'enregistrements dont les valeurs cibles sont connues.
approximation	Prévoit une valeur continue (c'est-à-dire un type de données <code>range</code> ) d'un attribut cible inconnu provenant d'enregistrements dont les valeurs cibles sont connues.
clustering	Identifie des groupes d'enregistrements similaires et les étiquette en conséquence
association	Identifie les événements ou attributs liés dans les données.
sequence	Recherche les modèles séquentiels dans les données à structure temporelle.
reduction	Réduit la complexité des données — par exemple, via des champs dérivés qui résument le contenu des champs de données d'origine.
conceptExtraction	Utilisé dans l'exploration de texte.
categorize	Utilisé dans l'exploration de texte.
timeSeries	Prévoit les valeurs futures à partir des motifs de données du passé.
anomalyDetection	Recherche les observations inhabituelles en se basant sur les écarts par rapport aux normes de leurs groupes.
attributImportance	Identifie les attributs présentant la plus forte influence sur un attribut cible.
supervisedMultiTarget	Estime la probabilité d'un résultat ( <i>oui</i> ou <i>non</i> ) pour l'une des différentes possibilités.

Si le modèle exécute plusieurs fonctions, les noms des fonctions sont séparés par des espaces dans les crochets, comme dans l'exemple suivant :

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

### Éléments enfants

Les éléments enfants de l'élément `ModelBuilder` sont les suivants :

Table 5-2  
Éléments enfants de la déclaration du créateur de modèles

Élément enfant	Description	Voir...
Algorithm	(requis) Spécifie l'algorithme utilisé pour générer le modèle.	Algorithme sur p. 100
ModelingFields	Spécifie l'identifiant à utiliser ensuite dans la section Interface utilisateur afin de définir l'emplacement des commandes de champs d'entrée et de sortie pour le modèle. Les commandes elles-mêmes sont définies	Champs de modélisation sur p. 100

Elément enfant	Description	Voir...
	dans les éléments enfants InputFields et OutputFields de ModelingFields.	
ModelGeneration	Spécifie l'identifiant à utiliser ensuite dans la section Interface utilisateur afin de définir l'emplacement des commandes de nom de modèle pour le modèle généré.	Génération de modèle sur p. 104
ModelFields	Spécifie l'ensemble des champs d'entrée et de sortie utilisés pour évaluer les données à l'aide de ce modèle.	Champs de modèle sur p. 104
AutoModeling	Permet à ce modèle d'être utilisé par un noeud de modélisation d'ensemble, tel que Classificateur automatique, Classification automatique ou Numérisation automatique.	Modélisation automatisée sur p. 113

### Algorithme

L'élément Algorithm définit les détails de l'algorithme utilisé pour générer le modèle.

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

où :

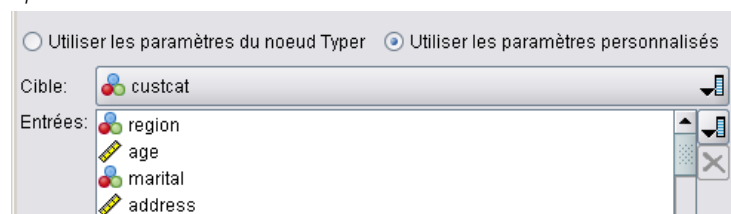
- value (requis) est le nom interne de l'algorithme. Il est référencé dans un certain nombre d'autres emplacements dans le fichier de spécifications. [Pour plus d'informations, reportez-vous à la section Exemple de création de modèles sur p. 105.](#)
- label (requis) est une description de l'algorithme.
- labelKey identifie l'étiquette à des fins de localisation.

### Champs de modélisation

Les champs d'entrée et de sortie de modèle sont généralement spécifiés au moyen du noeud Typer. Les utilisateurs définissent le rôle du champ sur In ou Out selon le cas. Pour un noeud de création de modèles, les utilisateurs peuvent également avoir le choix de remplacer les paramètres dans un noeud Typer en amont et d'utiliser des paramètres personnalisés — par exemple :

Figure 5-1

Spécification des entrées et sorties de modèles



Cette opération est réalisée au moyen de l'élément ModelingFields. Cet élément spécifie un identifiant utilisé par la suite dans la section Interface utilisateur de la déclaration du noeud de création de modèle pour définir l'emplacement des commandes pour les champs d'entrée et de sortie du modèle. Les commandes elles-mêmes sont définies dans les éléments enfants InputFields et OutputFields.



**Format**

```
<ModelingFields controlId="control_identifieur" ignoreBOTH="true_false" >
  <InputFields ... />
  <OutputFields ... />
</ModelingFields>
```

où :

- **controlId** (requis) est l'identifiant à utiliser par la suite dans un élément **SystemControls** dans la section Interface utilisateur de la déclaration du noeud de création de modèles. Cela permet d'identifier quel onglet de la boîte de dialogue du noeud doit contenir les contrôles de champ d'entrée et de sortie pour le modèle.
- **ignoreBOTH**, s'il est défini sur *true* (vrai) (valeur par défaut), spécifie que les champs présentant un rôle de champ défini sur *Both* (les deux) sont ignorés par le modèle.

Les éléments **InputFields** et **OutputFields** sont décrits dans les sections commençant à [Champs d'entrée sur p. 102](#).

**Exemple**

Cet exemple illustre l'utilisation d'un ensemble de commandes de champs de modélisation dans l'onglet Champs d'une boîte de dialogue de noeud de création de modèles. Tout d'abord, un identifiant est spécifié pour l'ensemble de commandes :

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

L'identifiant **modelingFields** est par la suite référencé dans la section Interface utilisateur de la boîte de dialogue du noeud, à l'endroit où l'onglet Champs est défini :

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

### Champs d'entrée

L'élément `InputFields` définit l'ensemble de champs parmi lesquels les utilisateurs pourront sélectionner un ou plusieurs champs d'entrée (c'est-à-dire prédicteurs) pour le modèle.

L'ensemble comprend tous les champs visibles sur ce noeud. Si les champs ont été filtrés plus en amont de ce noeud, seuls les champs qui sont passés dans le filtre sont visibles. Cette liste peut également être restreinte davantage en spécifiant que seuls les champs présentant des types de stockage et de données particuliers doivent être disponibles pour la sélection.

```
<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"/>
```

Vous pouvez limiter la liste de champs à utiliser comme champs d'entrée en spécifiant deux attributs, dont l'un doit provenir de la liste suivante :

- `storage` est une propriété de liste spécifiant le type de stockage de champs à attribuer dans la liste ; par exemple, `storage="[integer real]"` signifie que seuls les champs présentant ces types de stockage seront répertoriés. Pour connaître la liste des types de stockage possibles, reportez-vous au tableau sous Données et types de stockage sur p. 226.
- `onlyNumeric`, s'il est défini sur `true` (vrai), spécifie que seuls les champs présentant un type de stockage numérique sont répertoriés.
- `onlySymbolic`, s'il est défini sur `true` (vrai), spécifie que seuls les champs présentant un type de stockage symbolique (c'est-à-dire une chaîne) sont répertoriés.
- `onlyDatetime`, s'il est défini sur `true` (vrai), spécifie que seuls les champs présentant un type de stockage date et heure sont répertoriés

Le second attribut spécifié doit provenir de cette liste :

- `types` est une propriété de liste spécifiant le type de données des champs à attribuer dans la liste ; par exemple, `types="[range flag]"` signifie que seuls les champs présentant ces types de stockage seront répertoriés. Les types de données possibles sont :
  - intervalle
  - Booléen
  - ensemble
  - orderedSet (ensemble ordonné)
  - numérique
  - discrète
  - typeless (sans type)
- `onlyRanges`, s'il est défini sur `true` (vrai), spécifie que seuls les champs présentant un type de données intervalles sont répertoriés.
- `onlyDiscrete`, s'il est défini sur `true` (vrai), spécifie que seuls les champs présentant un type de données discret (c'est-à-dire booléen, ensemble ou sans type) sont répertoriés.

Par exemple, une commande spécifiant `storage="[integer]"` et `types="[flag]"` garantit que seuls les champs d'entiers qui sont des booléens apparaîtront dans la liste.

Les attributs restants sont les suivants :

- `property` est l'identifiant de la propriété à utiliser pour stocker les valeurs de champ.
- `multiple` spécifie si les valeurs de champ sont une liste énumérée (`true` (vrai)) ou non (`false` (faux)).
- `label` est le nom d'affichage de la commande.
- `labelKey` identifie l'étiquette à des fins de localisation.

### **Champs de sortie**

L'élément `OutputFields` définit l'ensemble de champs parmi lesquels les utilisateurs pourront sélectionner un ou plusieurs champs de sortie (c'est-à-dire cibles) pour le modèle.

L'ensemble comprend tous les champs visibles sur ce noeud. Si les champs ont été filtrés plus en amont de ce noeud, seuls les champs qui sont passés dans le filtre sont visibles. Cette liste peut également être restreinte davantage en spécifiant que seuls les champs présentant des types de stockage et de données particuliers doivent être disponibles pour la sélection.

```
<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"/>
```

Vous pouvez limiter la liste de champs à utiliser comme champs de sortie en spécifiant deux attributs, dont l'un doit provenir de la liste suivante :

- `storage` est une propriété de liste spécifiant le type de stockage de champs à attribuer dans la liste ; par exemple, `storage="[integer real]"` signifie que seuls les champs présentant ces types de stockage seront répertoriés. Pour connaître la liste des types de stockage possibles, reportez-vous au tableau sous Données et types de stockage sur p. 226.
- `onlyNumeric`, s'il est défini sur `true` (vrai), spécifie que seuls les champs présentant un type de stockage numérique sont répertoriés.
- `onlySymbolic`, s'il est défini sur `true` (vrai), spécifie que seuls les champs présentant un type de stockage symbolique (c'est-à-dire une chaîne) sont répertoriés.
- `onlyDatetime`, s'il est défini sur `true` (vrai), spécifie que seuls les champs présentant un type de stockage date et heure sont répertoriés.

Le second attribut spécifié doit provenir de cette liste :

- `types` est une propriété de liste spécifiant le type de données des champs à attribuer dans la liste ; par exemple, `types="[range flag]"` signifie que seuls les champs présentant ces types de stockage seront répertoriés. Les types de données possibles sont :

intervalle

Booléen

ensemble

orderedSet (ensemble ordonné)

numérique

discrète

typeless (sans type)

- `onlyRanges`, s'il est défini sur `true` (vrai), spécifie que seuls les champs présentant un type de données intervalles sont répertoriés.
- `onlyDiscrete`, s'il est défini sur `true` (vrai), spécifie que seuls les champs présentant un type de données discret (c'est-à-dire booléen, ensemble ou sans type) sont répertoriés.

Par exemple, une commande spécifiant `storage="[integer]"` et `types="[flag]"` garantit que seuls les champs d'entiers qui sont des booléens apparaîtront dans la liste.

Les attributs restants sont les suivants :

- `property` est l'identifiant de la propriété à utiliser pour stocker les valeurs de champ.
- `multiple` spécifie si les valeurs de champ sont une liste énumérée (`true` (vrai)) ou non (`false` (faux)).
- `label` est le nom d'affichage de la commande.
- `labelKey` identifie l'étiquette à des fins de localisation.

### Génération de modèle

L'élément `ModelGeneration` spécifie l'identifiant à utiliser ailleurs dans le fichier pour définir quel onglet de la boîte de dialogue du noeud de création de modèle doit contenir les commandes du nom de modèle pour le modèle généré :

Figure 5-2  
Commandes de nom de modèle



Le format est le suivant :

```
<ModelGeneration controlId="control_identifieur" />
```

L'attribut `controlId` est l'identifiant à utiliser par la suite dans un élément `SystemControls` dans la section Interface utilisateur de la spécification du noeud de création de modèles. L'onglet dont la spécification comprend cet élément `SystemControls` est celui qui contiendra les commandes de nom de modèle.

### Champs de modèle

L'élément `ModelFields` est utilisé pour construire la **signature du modèle** - l'ensemble de champs d'entrée et de sortie utilisés pour évaluer les données à l'aide de ce modèle.

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

```
...
</ModelFields>
```

où `inputDirections` et `outputDirections` spécifient la manière dont la signature du modèle doit être élaborée ; les valeurs peuvent être `in`, `out` ou `both`.

Les champs eux-mêmes sont spécifiés par un ou plusieurs éléments `AddField`. L'attribut `prefix` spécifie le préfixe à ajouter au nom d'un champ afin de désigner un champ généré par le modèle. Par exemple, si le nom du champ est `champ1` et que la valeur du préfixe est `$$`, le champ généré est nommé `$$-champ1`. [Pour plus d'informations, reportez-vous à la section Ajouter un champ dans le chapitre 4 sur p. 80.](#)

L'itération est possible au moyen des éléments `ForEach`. [Pour plus d'informations, reportez-vous à la section Itération avec l'élément ForEach dans le chapitre 4 sur p. 83.](#)

**Les groupes de champs** permettent de regrouper deux ou plusieurs champs de sortie du modèle à des fins d'itération. Les noms des champs de sortie sont suivis d'un suffixe indiquant l'itération, par exemple `$$-field1-1`, `$$-field1-2` etc. Une des utilisations des groupes de champs est de faire apparaître le même ensemble de champs plusieurs fois dans les sorties du modèle. [Pour plus d'informations, reportez-vous à la section Exemple de groupe de champs sur p. 106.](#)

## Automodeling

L'élément `AutoModeling` permet au modèle d'être utilisé par un noeud de modélisation d'ensemble, tel que `Classificateur automatique`, `Classification automatique` ou `Numérisation automatique`. [Pour plus d'informations, reportez-vous à la section Modélisation automatisée sur p. 113.](#)

## Exemple de création de modèles

La description suivante présente la section complète du créateur de modèles dans le fichier de spécifications pour l'exemple de noeud `Interaction` (voir [Noeud de création de modèle \(Interaction\) sur p. 35](#)):

```
<Node id="interaction.builder" type="modelBuilder" palette="modeling" label="Interaction">
  <ModelBuilder miningFunctions="[classification]">
    <Algorithm value="robd" label="Algorithme de Robert" />
    <ModelingFields controlsId="modellingFields">
      <InputFields property="inputs" multiple="true" label="Entrée" onlyDiscrete="true" />
      <OutputFields property="target" multiple="false" label="Cible" 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>
```

### Exemple de groupe de champs

Cet exemple est extrait du noeud SLRM, et ajoute un groupe de deux nouveaux champs à la signature du modèle pour qu'elle contienne les données générées lors de l'évaluation du modèle. Pour chaque enregistrement d'entrée, les données sont évaluées pour chacun des nouveaux champs selon le nombre de fois que l'utilisateur aura précisé et en fonction de la valeur de la propriété `max_predictions`.

Les deux nouveaux champs sont :

- `$$-cible` – qui contient la valeur prédite du champ cible
- `$$C-cible` – qui contient la valeur de probabilité pour cette prédiction

Afin de grouper ces deux champs, le même identificateur leur est attribué lorsqu'ils sont déclarés dans la section `ModelFields`. Les identificateurs de groupes sont attribués avec l'attribut `group` de l'élément `AddField`.

Par conséquent, la déclaration du noeud de création de modèle contient :

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

La déclaration du noeud d'application de modèle contient :

```
<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>
```

Le champ cible s'appelle `campagne` et l'utilisateur saisit `2` dans le champ correspondant à la propriété `max_predictions`. L'exécution du noeud de création de modèle ajoute les champs suivants au modèle :

- `$$-campagne-1`

- \$SC-campagne-1
- \$\$-campagne-2
- \$SC-campagne-2

## Sortie du modèle

L'élément `ModelOutput` décrit un objet de sortie de modèle - un objet qui apparaîtra sous l'onglet Modèles dans le panneau du gestionnaire suite à l'exécution d'un flux.

### Format

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

où :

- `id` (requis) est un identifiant unique pour le modèle généré.
- `label` (requis) est le nom d'affichage du modèle généré tel qu'il doit apparaître sous l'onglet Modèles.
- `labelKey` identifie l'étiquette à des fins de localisation.

Les éléments enfants pouvant figurer dans l'élément `ModelOutput` sont les suivants :

Table 5-3

Éléments enfants de la déclaration de sortie du modèle

Élément enfant	Définit	Voir...
ModelProvider	Le conteneur qui doit accueillir la sortie du modèle et si le modèle est au format PMML.	<a href="#">Fournisseur de modèle sur p. 64</a>
Properties	Propriétés à utiliser par le modèle généré.	<a href="#">Properties sur p. 65</a>
Containers	Conteneurs dans lesquels la sortie de modèle générée sera placée.	<a href="#">Containers sur p. 67</a>
UserInterface	L'interface utilisateur qui a généré la sortie du modèle peut être affichée — par exemple, les onglets Modèles et Paramètres dans un objet de sortie de modèle.	<a href="#">Interface utilisateur sur p. 68</a>
Constructors	Objets produits par le modèle généré.	<a href="#">Utilisation des constructeurs sur p. 123</a>

### Exemple

```
<ModelOutput id="interaction.model" label="Modèle d'Interaction">
  <Properties>
```

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

### ***Création de modèles interactifs***

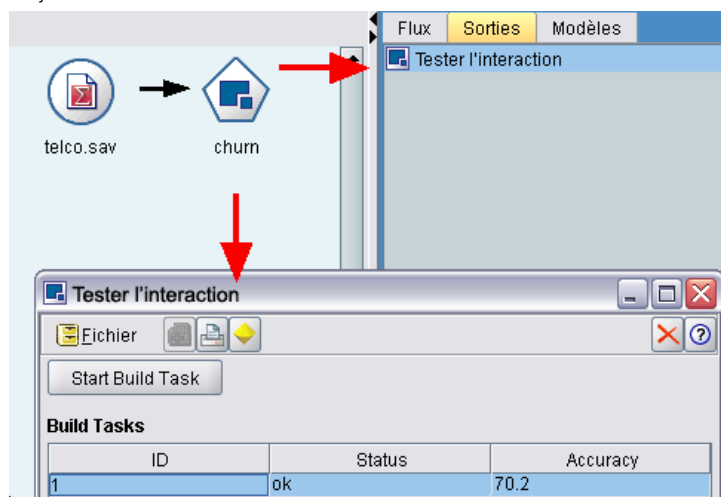
La modélisation interactive est une fonction qui vous permet de créer un objet de sortie avec lequel l'utilisateur final peut interagir avant de générer un modèle. Cet objet de sortie interactif est placé dans l'onglet Sorties du panneau du gestionnaire et contient un ensemble de données intermédiaires. L'ensemble de données intermédiaires peut être utilisé pour affiner ou simplifier le modèle avant sa génération. La modélisation interactive est obtenue en ajoutant quelques éléments supplémentaires à la spécification d'un noeud de création de modèle régulier :

- La section **Constructors** de la définition du **Node** comprend un élément **CreateInteractiveModelBuilder**.
- L'extension comprend un élément **InteractiveModelBuilder** dédié.

L'interaction de l'utilisateur avec l'ensemble de données intermédiaire s'effectue par le biais d'une fenêtre appelée **fenêtre d'interaction** qui s'affiche dès que l'objet de sortie est créé.



Figure 5-3  
Objet de sortie interactive et fenêtre d'interaction



L'interaction est spécifique à l'algorithme utilisé et mise en oeuvre par l'extension. La fenêtre d'interaction est définie dans la section Interface utilisateur de l'élément InteractiveModelBuilder. Vous pouvez définir une fenêtre d'interaction en spécifiant l'un des éléments suivants :

- Une classe de cadre (consultez la rubrique [Section Interface utilisateur sur p. 131](#)), qui définit complètement la fenêtre
- Une classe de panneau, spécifiée comme un attribut d'un panneau d'objet d'extension (consultez la rubrique [Panneau Objet d'extension sur p. 146](#)), pour chaque onglet de la fenêtre

Une fois fermée, la fenêtre d'interaction peut être affichée à nouveau en double-cliquant sur le nom de l'objet dans l'onglet Sorties.

La spécification de la fenêtre d'interaction doit comprendre le code permettant de générer le modèle une fois que l'utilisateur a terminé l'interaction. Dans l'exemple illustré, cela s'effectue au moyen du bouton de la barre d'outils présentant l'icône d'une pépite d'or, qui est associée à une action permettant de générer le modèle. Le code de cette opération est illustré dans la section InteractiveModelBuilder sous [Exemple de modélisation interactive sur p. 112](#).

### **Création de créateur de modèles interactif**

L'élément CreateInteractiveModelBuilder décrit l'objet de sortie avec lequel l'utilisateur pourra interagir. Il s'agit effectivement d'une version interactive de l'élément CreateModelOutput.

#### **Format**

Cet élément est utilisé dans la section Execution de la définition d'un noeud de création de modèles :

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

```

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

```

Le format de l'élément lui-même est :

```

<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>

```

où `type` (requis) est l'identifiant de l'objet de sortie qui est créé par l'élément `InteractiveModelBuilder`.

La section `Condition` vous permet de spécifier une ou plusieurs conditions. [Pour plus d'informations, reportez-vous à la section Conditions dans le chapitre 4 sur p. 89.](#)

Vous pouvez aussi spécifier des conditions complexes comprenant les opérateurs `And`, `Or` et `Not`. [Pour plus d'informations, reportez-vous à la section Opérateurs logiques dans le chapitre 4 sur p. 88.](#)

Dans les éléments `CreateModel` et `CreateDocument` :

- `type` est l'identifiant du modèle ou du document en cours de définition.
- `target` (obligatoire) est l'identifiant du conteneur du modèle ; ce conteneur est défini dans une section `Sortie de modèle`. [Pour plus d'informations, reportez-vous à la section Sortie du modèle sur p. 107.](#)
- `sourceFile` (obligatoire) est l'identifiant d'un fichier de sortie généré pendant l'exécution du noeud ; ce fichier est défini dans une section `Fichiers de sortie`. [Pour plus d'informations, reportez-vous à la section Fichiers de résultats dans le chapitre 4 sur p. 70.](#)

### Exemple

```

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

```

Cet exemple spécifie qu'un objet de sortie présentant l'identifiant `my.interaction` sera créé lors de l'exécution du noeud de création de modèles pour lequel la spécification contient cet élément. L'objet de sortie lui-même est défini ailleurs dans le fichier de spécification, par un élément `InteractiveModelBuilder` faisant référence à cet identifiant — par exemple :

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

### Créateur de modèles interactifs

Cet élément définit un objet de sortie interactif, qui permet à un utilisateur final d'affiner ou de simplifier un modèle avant de le générer.

L'élément `InteractiveModelBuilder` suit la définition d'un noeud de création de modèles contenant l'élément `CreateInteractiveModelBuilder` correspondant. [Pour plus d'informations, reportez-vous à la section Création de créateur de modèles interactif sur p. 109.](#)

#### Format

Le format de l'élément `InteractiveModelBuilder` est :

```
<Node ... type="modelBuilder" ... >
...
  -- Section Création de créateur de modèles interactif --
...
</Node>
...
<InteractiveModelBuilder id="identifiant" label="display_label" labelKey="label_key">
  <Properties>
    <Property name=... />
    ...
  </Properties>
  <Containers>
    <Container name="container_name"/>
  </Containers>
  <UserInterface ... />
  <Constructors ... />
</InteractiveModelBuilder>
```

où :

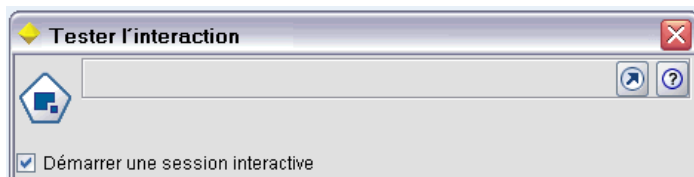
- `id` (requis) est un identifiant unique pour le modèle généré.
- `label` (requis) est le nom d'affichage du modèle généré tel qu'il doit apparaître sous l'onglet Modèles.
- `labelKey` identifie l'étiquette à des fins de localisation.

Pour plus d'informations à propos des éléments `Properties`, `Containers`, `UserInterface` et `Constructors`, consultez [Properties sur p. 65](#), [Containers sur p. 67](#), [Section Interface utilisateur sur p. 131](#) et [Utilisation des constructeurs sur p. 123](#).

### Exemple de modélisation interactive

Cet exemple illustre la manière dont vous pouvez définir un noeud de création de modèles de telle sorte que les utilisateurs peuvent choisir, via une simple case à cocher, s'ils doivent ou non interagir avant de générer le modèle — par exemple :

Figure 5-4  
Choix de l'interaction



Pour voir le fonctionnement de cette option, utilisez le noeud d'exemple Interaction fourni avec cette version. [Pour plus d'informations, reportez-vous à la section Noeud de création de modèle \(Interaction\) dans le chapitre 3 sur p. 35.](#)

Tout d'abord, le noeud de création de modèles spécifie une propriété booléenne :

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

Dans la section Interface utilisateur de la spécification du noeud, la section qui définit l'onglet Modèle comprend une référence à cette propriété :

```
<Tab label="Modèle">
  <PropertiesPanel>
    <CheckBoxControl property="Interactive" label="Démarrer une session interactive" />
  </PropertiesPanel>
</Tab>
```

Dans la section `CreateInteractiveModelBuilder` du même noeud, le paramètre de la propriété est testé et, s'il s'agit de `true`, un objet de sortie interactif est créé :

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

L'objet de sortie auquel il fait référence est défini dans la section `InteractiveModelBuilder` de l'extension :

```
<InteractiveModelBuilder id="my.interaction" label="Test d'interaction">
  <Properties>
  </Properties>
  <Containers>
  </Containers>
  <UserInterface actionHandler="ui.InteractionHandler">
```

```

<Controls>
  <ToolBarItem action="generateModel" showLabel="false" />
</Controls>
<Tabs>
  <Tab label="Modèle">
    <ExtensionObjectPanel id="model.panel" panelClass="ui.SampleInteractionPanel" />
  </Tab>
  <Tab label="Générique">
    <ExtensionObjectPanel id="generic.panel" panelClass="ui.GenericInteractionPanel" />
  </Tab>
</Tabs>
</UserInterface>
</InteractiveModelBuilder>

```

L'action permettant de générer le modèle est contrôlée par le bouton de la barre d'outils défini par l'élément `ToolBarItem`.

Notez l'utilisation de l'attribut `panelClass` de l'élément `ExtensionObjectPanel` pour spécifier une classe Java permettant de contrôler l'interface utilisateur pour chaque onglet de la fenêtre d'interaction.

## **Modélisation automatisée**

IBM® SPSS® Modeler fournit en standard un groupe de noeuds de modélisation d'ensembles, tels que les noeuds Classificateur automatique, Classification automatique et Numérisation automatique. Ces noeuds automatisent la création d'un certain nombre de modèles différents simultanément, ce qui permet à l'utilisateur de comparer les résultats et de choisir le meilleur modèle pour leurs données. CLEF fournit l'élément `AutoModeling` pour permettre à un modèle spécifié par l'élément `ModelBuilder` d'être utilisé par tout noeud d'ensemble.

Le format de l'élément `AutoModeling` est :

```

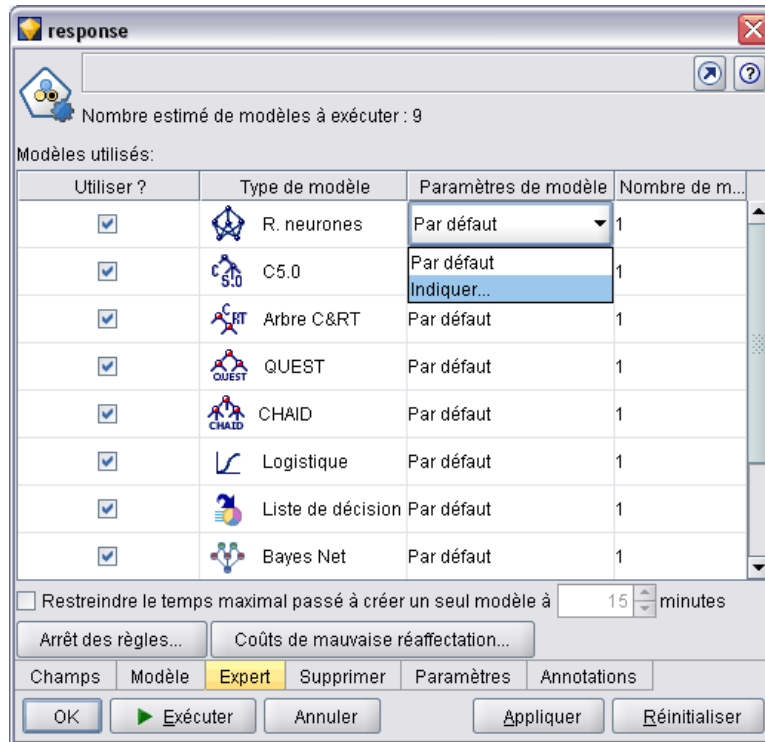
<AutoModeling enabledByDefault="true_false">
  <SimpleSettings ... />
  <ExpertSettings ... />
  <Constraint ... />
  <Constraint ... />
  ...
</AutoModeling>

```

où `enabledByDefault` spécifie si le modèle est activé pour être utilisé par défaut dans le noeud de modélisation d'ensemble (c'est-à-dire que la colonne `Use?` est activée par défaut pour ce modèle spécifique, comme indiqué dans l'illustration suivante). Si cet attribut est ignoré, la valeur `true` est supposée.

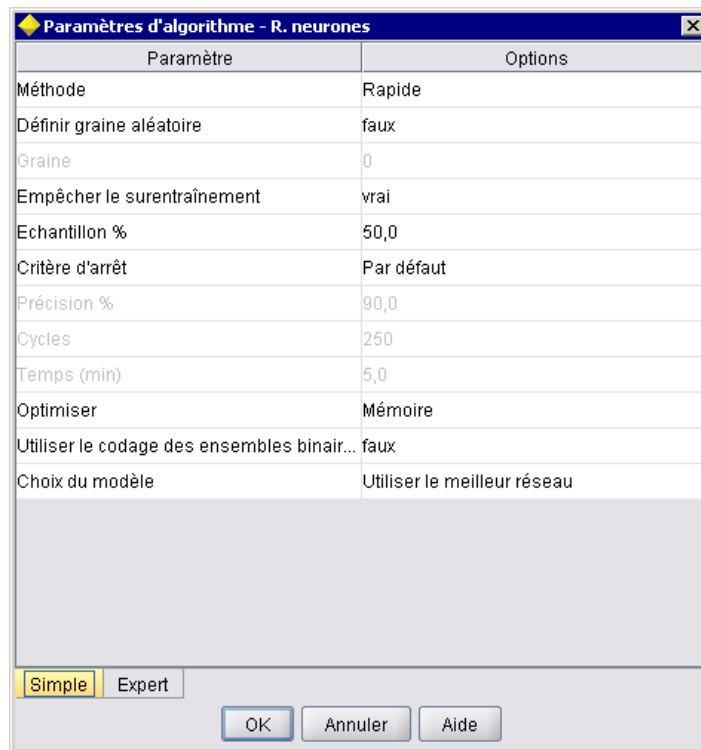
Dans la boîte de dialogue d'un noeud de modélisation d'ensemble, l'onglet Expert répertorie les modèles que les utilisateurs peuvent choisir de créer — par exemple :

Figure 5-5  
Onglet Expert d'un noeud de modélisation d'ensemble



Un clic sur Spécifier dans le champ Paramètres de modèle pour un modèle particulier affiche une boîte de dialogue Paramètres d'algorithme, où l'utilisateur peut choisir les options correspondant à ce type de modèle.

Figure 5-6  
Paramètres d'algorithme pour un noeud de modélisation d'ensemble



La boîte de dialogue Paramètres d'algorithme contient des onglets Simple et Expert, correspondant aux modes d'exécution Simple et Expert du noeud de modélisation. Le contenu des onglets Simple et Expert est contrôlé par les éléments SimpleSettings et ExpertSettings qui sont décrits dans les sections suivantes.

En outre, les éléments Constraint vous permettent de spécifier les conditions qui permettent à l'utilisateur final de modifier ou, dans certains cas, de restreindre les paramètres de la boîte de dialogue Paramètres d'algorithme. [Pour plus d'informations, reportez-vous à la section Contraintes sur p. 119.](#)

Certains paramètres de la boîte de dialogue Paramètres d'algorithme peuvent prendre plusieurs valeurs. Lorsque plusieurs valeurs sont spécifiées, le noeud d'ensemble tente de créer des modèles pour toutes les combinaisons possibles des valeurs de paramètres. Par exemple, avec un modèle linéaire généralisé, si l'utilisateur spécifie deux distributions (normale et gamma) et trois fonctions de lien (identité, log et puissance), le noeud Numérisation automatique tente de créer six modèles linéaires généralisés, un pour chaque combinaison possible de ces paramètres.

### **Paramètres simples**

L'élément SimpleSettings détermine quels paramètres doivent apparaître dans l'onglet Simple de la boîte de dialogue Paramètres d'algorithme pour ce modèle dans un noeud de modélisation d'ensemble. [Pour plus d'informations, reportez-vous à la section Modélisation automatisée sur p. 113.](#)

**Format**

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

Dans un élément PropertyGroup (au moins un est requis) :

label est une étiquette d'affichage du groupe de propriétés, insérée en tant que sous-titre dans la boîte de dialogue devant le premier paramètre du groupe.

labelKey identifie l'étiquette à des fins de localisation. Si vous n'utilisez ni label, ni labelKey, aucun sous-titre n'est inséré.

properties (obligatoire) est une liste d'une ou de plusieurs propriétés qui doivent apparaître sur l'onglet. La valeur de *prop\_name1*, *prop\_name2*, etc., est la valeur de l'attribut name de l'élément Property dans lequel cette propriété est définie. [Pour plus d'informations, reportez-vous à la section Propriétés dans le chapitre 4 sur p. 65.](#)

**Exemple**

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

Cet exemple du noeud Discriminant spécifie que seul le paramètre Méthode apparaîtra dans l'onglet Simple de la boîte de dialogue Paramètres d'algorithme pour ce modèle dans le noeud de modélisation d'ensemble pertinent (dans ce cas, le noeud Classificateur automatique). Dans la mesure où aucun attribut label ou labelKey n'est spécifié, aucun sous-titre du paramètre n'est affiché dans la boîte de dialogue.

**Paramètres Expert**

L'élément ExpertSettings détermine quels paramètres doivent apparaître dans l'onglet Expert de la boîte de dialogue Paramètres d'algorithme pour ce modèle dans un noeud de modélisation d'ensemble. [Pour plus d'informations, reportez-vous à la section Modélisation automatisée sur p. 113.](#)

**Format**

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



L'élément Condition spécifie une condition qui, si elle est définie sur true, active les paramètres identifiés par les éléments PropertyGroup suivants. [Pour plus d'informations, reportez-vous à la section Conditions dans le chapitre 4 sur p. 89.](#)

Dans un élément PropertyGroup (au moins un est requis) :

label est une étiquette d'affichage du groupe de propriétés, insérée en tant que sous-titre dans la boîte de dialogue devant le premier paramètre du groupe.

labelKey identifie l'étiquette à des fins de localisation. Si vous n'utilisez ni label, ni labelKey, aucun sous-titre n'est inséré.

properties (obligatoire) est une liste d'une ou de plusieurs propriétés qui doivent apparaître sur l'onglet. La valeur de *prop\_name1*, *prop\_name2*, etc., est la valeur de l'attribut name de l'élément Property dans lequel cette propriété est définie. [Pour plus d'informations, reportez-vous à la section Properties dans le chapitre 4 sur p. 65.](#)

### Exemples

Dans l'exemple suivant, l'onglet Expert de la boîte de dialogue Paramètres d'algorithme présente le paramètre Mode initialement défini sur Simple :

Figure 5-7  
Paramètres Expert désactivés



L'exemple suivant spécifie que les autres paramètres de l'onglet Expert ne sont activés que si l'utilisateur modifie le paramètre Mode sur Expert :

```
<ExpertSettings>
  <Condition property="mode" op="equals" value="Expert"/>
  <PropertyGroup properties="[mode prior_probabilities covariance_matrix]"/>
  ...
</ExpertSettings>
```

Le réglage du paramètre Mode sur Expert active les deux paramètres du groupe de propriétés :

Figure 5-8  
Paramètres Expert activés

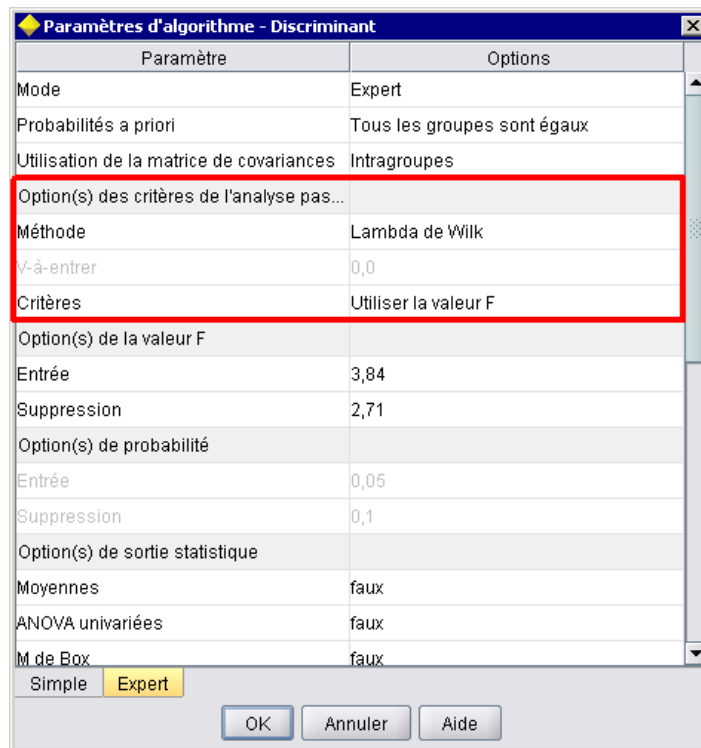


L'exemple suivant illustre l'utilisation des étiquettes pour le groupe de propriétés :

```
<ExpertSettings>
...
  <PropertyGroup labelKey="automodel.stepping_criteria_options"
    properties="[stepwise_method V_to_enter criteria]"/>
...
</ExpertSettings>
```

Ici, l'élément `PropertyGroup` contrôle les paramètres mis en surbrillance dans l'illustration :

Figure 5-9  
Paramètres Expert désactivés



L'attribut `labelKey` permet à CLEF de récupérer le texte de l'affichage pour le sous-titre du groupe de propriétés de l'entrée correspondante dans le fichier de propriétés pour l'extension :

**automodel.stepping\_criteria\_options=Stepping Criteria option(s)**

Pour plus d'informations, reportez-vous à la section [Fichiers de propriétés](#) dans le chapitre 8 sur p. 209.

## Contraintes

L'élément **Constraint** spécifie la condition qui permet l'édition ou présente une contrainte, quelle qu'elle soit, pour les paramètres répertoriés dans la boîte de dialogue Paramètres d'algorithme dans un noeud de modélisation d'ensemble. Par exemple, certains paramètres peuvent être désactivés si l'utilisateur final n'est pas autorisé actuellement à les modifier.

### Format

```
<Constraint property="prop_name" singleSelection="true_false">
  <Condition property="prop_name" op="operator" value="value"/>
  <And ... />
  <Or ... />
  <Not ... />
</Constraint>
```

où :

- **property** (requis) identifie un paramètre à modifier ou à contraindre. *prop\_name* est la valeur de l'attribut **name** de l'élément **Property** dans lequel la propriété correspondant à ce paramètre est définie. [Pour plus d'informations, reportez-vous à la section Propriétés dans le chapitre 4 sur p. 65.](#)
- **singleSelection** détermine si l'utilisateur final peut sélectionner plus d'une valeur parmi les valeurs disponibles pour un paramètre. S'il est défini sur **true**, une seule valeur peut être sélectionnée, même si plusieurs valeurs sont répertoriées dans le champ Options de ce paramètre dans la boîte de dialogue Paramètres d'algorithme. S'il est défini sur **false** (valeur par défaut), l'utilisateur peut choisir une ou plusieurs des valeurs disponibles, comme illustré dans l'exemple suivant :

L'élément **Condition** spécifie la contrainte réelle. [Pour plus d'informations, reportez-vous à la section Conditions dans le chapitre 4 sur p. 89.](#)

Les éléments **And**, **Or** et **Not** peuvent être utilisés pour spécifier des conditions composées. [Pour plus d'informations, reportez-vous à la section Opérateurs logiques dans le chapitre 4 sur p. 88.](#)

### Exemple

Cet exemple est extrait du fichier de spécifications pour le noeud linéaire généralisé. Même en mode Expert, certains paramètres ne sont pas activés par défaut.

Figure 5-10  
Effet d'une contrainte — paramètre désactivé

Paramètre	Options
Mode	Expert
Proportion	Normal
Fonction de lien	Identité
Puissance	0,0
Tolérance de singularité	1E-007
Ordre des valeurs pour les entrées cat...	Croissant
Option(s) d'estimation des paramètres	
Méthode	Hybride
Itérations maximales de scoring de Fis...	1
Méthode de paramètre d'échelle	Estimation du maximum de vraisembl...
Valeur	1,0

La contrainte spécifie les conditions dans lesquelles le paramètre Valeur est activé.

```
<Constraint property="scale_value">
  <And>
    <Condition property="scale_method" op="equals" value="FixedValue"/>
    <Condition property="distribution" op="in" value="[IGAUSS GAMMA NORMAL]"/>
  </And>
</Constraint>
```

Le paramètre Méthode de paramètre d'échelle (identifié par la propriété `scale_method`) doit être défini sur Valeur fixe, et le paramètre Distribution doit être Normal, Gaussien inverse ou Gamma pour que le paramètre Valeur soit activé.

Figure 5-11  
Effet d'un paramètre contrainte activé

Paramètre	Options
Mode	Expert
Proportion	Normal
Fonction de lien	Identité
Puissance	0,0
Tolérance de singularité	1E-007
Ordre des valeurs pour les entrées cat...	Croissant
Option(s) d'estimation des paramètres	
Méthode	Hybride
Itérations maximales de scoring de Fis...	1
Méthode de paramètre d'échelle	Valeur fixe
Valeur	1,0

## Application de modèles

L'application d'un modèle signifie l'utilisation d'un modèle généralisé pour évaluer les données, c'est-à-dire pour utiliser les informations recueillies lors de la création de modèle pour créer des prédictions pour les nouveaux enregistrements. Dans IBM® SPSS® Modeler, vous effectuez cette opération au moyen d'un noeud applicateur de modèle. [Pour plus d'informations, reportez-vous à la section Noeuds applicateurs de modèle dans le chapitre 2 sur p. 13.](#)

La définition d'un noeud applicateur de modèle dans le fichier de spécifications crée le cadre d'application d'un modèle généré. Dans SPSS Modeler, vous créez une instance d'un noeud applicateur de modèle en faisant glisser l'icône représentant l'objet de sortie de modèle à partir de l'onglet Modèle du panneau du gestionnaire sur le travail de flux. Sans définition de noeud applicateur de modèle, l'exécution du noeud de création de modèle générerait simplement un modèle non affiné, lequel ne peut être ajouté à l'espace de travail de flux.

Lors de la définition d'un noeud applicateur de modèles, l'élément Noeud doit comprendre :

- Un attribut type="modelApplier"
- Un élément enfant Constructors contenant un élément CreateModelOutput(consultez la rubrique [Utilisation des constructeurs sur p. 123](#))

Pour le format d'un élément de spécification Node, consultez la rubrique [Noeud sur p. 61](#).

## Création de documents

Lors de la définition d'un noeud de création de documents, l'élément Node doit comprendre :

- Un attribut type="documentBuilder"
- Un élément enfant DocumentBuilder

Pour la création de modèles, l'extension doit également disposer d'un élément DocumentOutput pour décrire le document généré. [Pour plus d'informations, reportez-vous à la section Sortie du modèle sur p. 107.](#)

Pour le format d'un élément de spécification Node, consultez la rubrique [Noeud sur p. 61](#).

## Créateur de document

L'élément DocumentBuilder définit le comportement d'un noeud de création de documents. La définition doit comprendre un élément enfant DocumentGeneration permettant de spécifier quel onglet de la boîte de dialogue du noeud de création de document contiendra les commandes de génération de document. Les commandes elles-mêmes sont définies dans la section Interface utilisateur (consultez la rubrique [le chapitre 6, Création d'interfaces utilisateur sur p. 128](#)).

### Format

```
<DocumentBuilder>
  <DocumentGeneration controlId="control_identifieur" />
</DocumentBuilder>
```

où `controlsId` (requis) est l'identifiant utilisé par les commandes du système pour spécifier l'endroit où les commandes de génération de document doivent apparaître.

### Exemple

```
<DocumentBuilder>
  <DocumentGeneration controlsId="1"/>
</DocumentBuilder>
```

## Sortie de document

L'élément `DocumentOutput` décrit un objet de sortie de document - un objet qui apparaîtra sous l'onglet Sorties dans le panneau du gestionnaire suite à l'exécution d'un flux.

### Format

```
<DocumentOutput id="identifiant" label="display_label" labelKey="label_key" >
  <Properties>
    <Property ... />
    ...
  </Properties>
  <Containers>
    <Container ... />
    ...
  </Containers>
  <UserInterface ... />
  <Constructors ... />
</DocumentOutput>
```

où :

- `id` (requis) est un identifiant unique pour le document généré.
- `label` (requis) est le nom d'affichage du document généré tel qu'il doit apparaître sous l'onglet Sorties.
- `labelKey` identifie l'étiquette à des fins de localisation.

Les éléments enfants pouvant figurer dans l'élément `DocumentOutput` sont les suivants :

Table 5-4  
Eléments enfants de la déclaration de sortie du document

Élément enfant	Définit	Voir...
Properties	Propriétés à utiliser par le document généré.	<a href="#">Properties sur p. 65</a>
Containers	Conteneurs dans lesquels la sortie de document générée sera placée.	<a href="#">Containers sur p. 67</a>
UserInterface	L'interface utilisateur par laquelle la sortie de document générée peut être affichée.	<a href="#">Interface utilisateur sur p. 68</a>
Constructors	Objets produits par le document généré.	<a href="#">Utilisation des constructeurs sur p. 123</a>

**Exemple**

```

<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>

```

**Utilisation des constructeurs**

Un élément `Constructors` peut apparaître à deux endroits dans le fichier de spécifications :

- Dans la section `Execution` de la définition d'un noeud (pour les objets de sortie de modèle ou de document)
- Dans la définition d'une sortie de modèle (pour un noeud applicateur de modèle)

Un seul objet de sortie peut être généré par un noeud. Cette limite est imposée pour des raisons de cohérence avec les noeuds existants et les interfaces de génération de script et d'API client pour ces types de noeuds.

**Format**

Le format de l'élément `Constructors`, lorsqu'il est utilisé dans une section `Execution`, est le suivant :

```

<Constructors>
  <CreateModelOutput ... />
  ...
  <CreateDocumentOutput ... />
  ...
  <CreateInteractiveModelBuilder ... />
  ...
</Constructors>

```

Dans une définition de sortie de modèle, le format est :

```

<Constructors>
  <CreateModelApplier ... />
</Constructors>

```

**Création de sortie de modèle**

Cette section définit la manière dont un objet de sortie de modèle est créée dans l'onglet `Modèles` ou la manière dont un objet de sortie de document est créé dans l'onglet `Sorties`.

**Format**

```

<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>

```

Dans l'élément `CreateModelOutput` :

- `type` (requis) est l'identifiant d'un objet de sortie de modèle qui est défini dans la section Sortie de modèle. [Pour plus d'informations, reportez-vous à la section Sortie du modèle dans le chapitre 4 sur p. 63.](#)

La section `Condition` vous permet de spécifier une ou plusieurs conditions. [Pour plus d'informations, reportez-vous à la section Conditions dans le chapitre 4 sur p. 89.](#)

Vous pouvez aussi spécifier des conditions complexes comprenant les opérateurs `And`, `Or` et `Not`. [Pour plus d'informations, reportez-vous à la section Opérateurs logiques dans le chapitre 4 sur p. 88.](#)

Dans les éléments `CreateModel` et `CreateDocument` :

- `type` est l'identifiant du modèle ou du document en cours de définition.
- `target` (obligatoire) est l'identifiant du conteneur du modèle ; ce conteneur est défini dans une section Sortie de modèle. [Pour plus d'informations, reportez-vous à la section Sortie du modèle sur p. 107.](#)
- `sourceFile` (obligatoire) est l'identifiant d'un fichier de sortie généré pendant l'exécution du noeud ; ce fichier est défini dans une section Fichiers de sortie. [Pour plus d'informations, reportez-vous à la section Fichiers de résultats dans le chapitre 4 sur p. 70.](#)

**Exemple**

```

<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>

```

**Création de sortie de document**

Cet élément est utilisé dans la section Execution de la définition du noeud de création de document et identifie l'objet de sortie de document qui est en cours de création.



**Format**

```

<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>

```

où **type** (requis) est l'identifiant d'un objet de sortie de document qui est défini dans la section Sortie de document. [Pour plus d'informations, reportez-vous à la section Sortie de document dans le chapitre 4 sur p. 64.](#)

La section **Condition** vous permet de spécifier une ou plusieurs conditions. [Pour plus d'informations, reportez-vous à la section Conditions dans le chapitre 4 sur p. 89.](#)

Vous pouvez aussi spécifier des conditions complexes comprenant les opérateurs **And**, **Or** et **Not**. [Pour plus d'informations, reportez-vous à la section Opérateurs logiques dans le chapitre 4 sur p. 88.](#)

Dans les éléments **CreateModel** et **CreateDocument** :

- **type** est l'identifiant du modèle ou du document en cours de définition.
- **target** (obligatoire) est l'identifiant du conteneur du modèle ; ce conteneur est défini dans une section Sortie de modèle. [Pour plus d'informations, reportez-vous à la section Sortie du modèle sur p. 107.](#)
- **sourceFile** (obligatoire) est l'identifiant d'un fichier de sortie généré pendant l'exécution du noeud ; ce fichier est défini dans une section Fichiers de sortie. [Pour plus d'informations, reportez-vous à la section Fichiers de résultats dans le chapitre 4 sur p. 70.](#)

**Exemple**

```

<Constructors>
  <CreateDocumentOutput type="webstatusreport">
    <CreateDocument type="webstatusreport" target="webstatusreportdata"
      sourceFile="webstatusreport_output_file" />
  </CreateDocumentOutput>
</Constructors>

```

**Création de créateur de modèles interactif**

Cet élément est utilisé dans la section Exécution de la définition d'un noeud de création de modèle et identifie l'objet de sortie avec lequel l'utilisateur interagira. [Pour plus d'informations, reportez-vous à la section Création de créateur de modèles interactif sur p. 109.](#)

## Création d'applicateur de modèle

Cet élément est utilisé dans un élément `Constructors` de la section `Sortie de modèle` de la définition d'un noeud de création de modèle (consultez la rubrique [Sortie du modèle sur p. 107](#)). L'élément `CreateModelApplier` définit la manière dont un noeud applicateur de modèle est créé lorsqu'un objet de sortie de modèle généré par le noeud de création de modèle est placé dans l'espace de travail.

### Format

```
<CreateModelApplier type="apply_node_identifieur">
  <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>
```

Dans l'élément `CreateModelApplier` :

- `type` (requis) est l'identifiant du noeud applicateur de modèle à créer ; ce noeud est en réalité défini dans un élément `Node ... type="modelApplier"` plus loin dans le fichier.

La section `Condition` vous permet de spécifier une ou plusieurs conditions. [Pour plus d'informations, reportez-vous à la section Conditions dans le chapitre 4 sur p. 89.](#)

Vous pouvez aussi spécifier des conditions complexes comprenant les opérateurs `And`, `Or` et `Not`. [Pour plus d'informations, reportez-vous à la section Opérateurs logiques dans le chapitre 4 sur p. 88.](#)

Dans les éléments `CreateModel` et `CreateDocument` :

- `type` est l'identifiant du modèle ou du document en cours de définition.
- `target` (obligatoire) est l'identifiant du conteneur du modèle ; ce conteneur est défini dans une section `Sortie de modèle`. [Pour plus d'informations, reportez-vous à la section Sortie du modèle sur p. 107.](#)
- `sourceFile` (obligatoire) est l'identifiant d'un fichier de sortie généré pendant l'exécution du noeud ; ce fichier est défini dans une section `Fichiers de sortie`. [Pour plus d'informations, reportez-vous à la section Fichiers de résultats dans le chapitre 4 sur p. 70.](#)

### Exemple

Dans l'exemple suivant, l'élément `CreateModelApplier` contient une référence en aval au noeud applicateur de modèle nommé `myapplier`. Ce noeud est défini dans l'élément `Node` qui suit.

```
<ModelOutput>
  <Constructors>
    <CreateModelApplier type="myapplier"></CreateModelApplier>
  </Constructors>
```

```
</ModelOutput>  
<Node id="myapplier" type="modelApplier">  
...  
</Node>
```

# Création d'interfaces utilisateur

## A propos des interfaces utilisateur

Lors de l'ajout d'un nouveau noeud CLEF, vous devez définir la manière dont l'utilisateur final interagit avec le noeud et toute sortie de modèle, sortie de document, ou noeud applicateur que l'extension indique. L'interface utilisateur pour chaque objet est définie dans une section `UserInterface` du fichier de spécifications pour l'extension. Ce chapitre offre une description détaillée de la section `UserInterface`.

*Remarque* : Plusieurs sections `UserInterface` peuvent figurer dans un seul fichier de spécifications, selon les types d'objets définis dans le fichier.

L'interface utilisateur d'un objet CLEF comprend un ou plusieurs des éléments suivants :

- Entrée de menu
- Entrée de palette
- Icônes
- Une ou plusieurs boîtes de dialogue
- Une ou plusieurs fenêtres de sortie

Les **entrées de menu** et les **entrées de palette** offrent l'accès à l'objet à partir du système de menus et des palettes de noeuds de IBM® SPSS® Modeler, respectivement. Les **icônes** identifient les objets sur les menus, sur la zone de dessin et dans les différentes palettes de noeuds. Les **boîtes de dialogue** contiennent des onglets et des contrôles pour permettre aux utilisateurs d'indiquer différentes options avant l'exécution d'un flux, et en option, de spécifier la sortie à générer à la fin de l'exécution. Les **fenêtres de sortie** permettent d'afficher la sortie de modèle (comme les résultats d'application d'un modèle à un ensemble de données) ou la sortie de document (comme un graphique).

CLEF permet d'ajouter des nouveaux types d'objets aux objets standard fournis par SPSS Modeler et adopte une approche cohérente pour la définition de l'interface utilisateur pour ces nouveaux objets. [Conception d'icônes de noeud sur p. 17](#) et [Conception des boîtes de dialogue sur p. 21](#) offre des directives pour la création d'icônes et de boîtes de dialogue dans CLEF.

Les **icônes** prennent la forme de graphiques qui identifient un noeud particulier. En outre, elles figurent à l'intérieur des formes géométriques qui identifient le type de noeud.

Les **boîtes de dialogue** et les **fenêtres de sortie** possèdent les caractéristiques suivantes :

- Barre de titre contenant l'icône miniature et le titre de l'objet
- Barre de menus qui peut contenir :
  - Menus
  - Boutons d'action propres à l'objet
  - Boutons d'action standard (par exemple, Agrandir ou Aide)

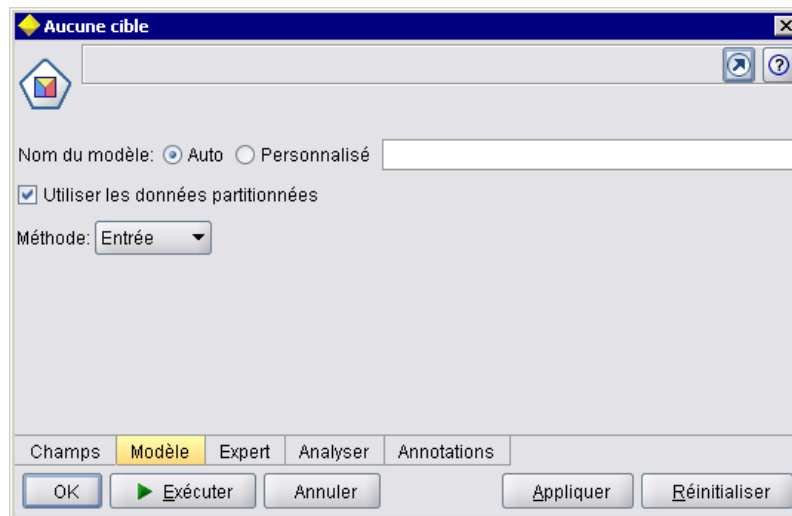
- Zone de contenu principale
- Plusieurs onglets pour organiser les composants IU en groupes logiques
- Capacité de redimensionnement

Un onglet modifie la zone de contenu principale de la fenêtre de différentes façons. Pour une boîte de dialogue, différents onglets affichent des ensembles distincts de contrôles pour les propriétés de l'objet. Les contrôles peuvent être modifiés, et les résultats appliqués à l'objet sous-jacent lorsque l'utilisateur clique sur le bouton Appliquer ou OK.

Pour une fenêtre de sortie, les onglets permettent à l'utilisateur d'effectuer différentes actions relatives à la sortie, telles que l'affichage d'un récapitulatif des résultats ou l'ajout d'annotations à l'intérieur.

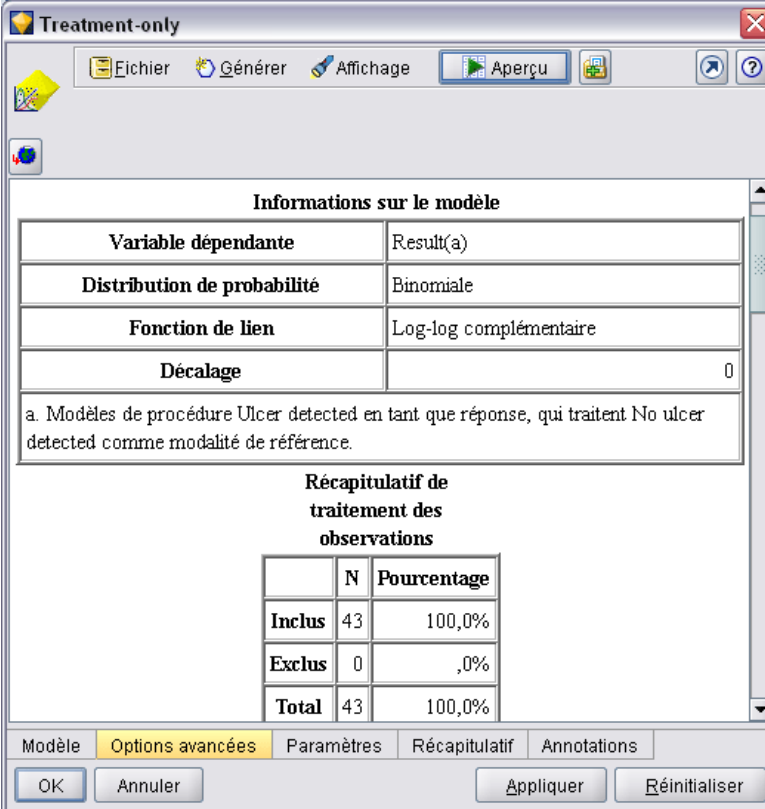
Vous trouverez ci-après une boîte de dialogue simple avec différents onglets et contrôles :

Figure 6-1  
Boîte de dialogue à onglets



Vous trouverez ci-après un exemple de fenêtre de sortie de modèle :

Figure 6-2  
Fenêtre de sortie de modèle



The screenshot shows a window titled "Treatment-only" with a menu bar containing "Eichier", "Générer", "Affichage", "Aperçu", and a help icon. The main content area is divided into two sections:

**Informations sur le modèle**

Variable dépendante	Result(a)
Distribution de probabilité	Binomiale
Fonction de lien	Log-log complémentaire
Décalage	0

a. Modèles de procédure Ulcer detected en tant que réponse, qui traitent No ulcer detected comme modalité de référence.

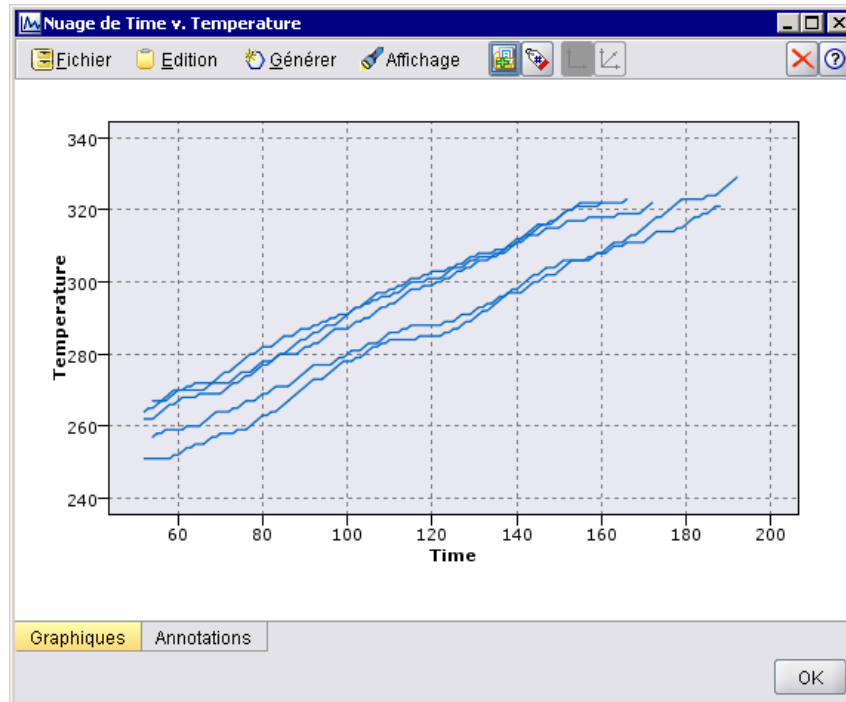
**Récapitulatif de traitement des observations**

	N	Pourcentage
Inclus	43	100,0%
Exclus	0	,0%
Total	43	100,0%

At the bottom, there are tabs for "Modèle", "Options avancées", "Paramètres", "Récapitulatif", and "Annotations". Below the tabs are buttons for "OK", "Annuler", "Appliquer", and "Réinitialiser".

Et finalement, un exemple de fenêtre de sortie de document (dans ce cas, un graphique) :

Figure 6-3  
Fenêtre de sortie de document



## Section Interface utilisateur

L'interface utilisateur d'un objet est déclarée dans un élément `UserInterface` dans la définition d'objet du fichier de spécifications. Plusieurs éléments `UserInterface` peuvent figurer dans le même fichier de spécifications, selon le nombre d'objets (par exemple, noeud concepteur de modèle, objet sortie de modèle, noeud programme d'application de modèle) définis dans le fichier.

Dans chaque section Interface utilisateur, vous pouvez définir :

- des icônes à afficher dans la zone de dessin ou les palettes ;
- des contrôles (éléments de barres d'outils et de menus personnalisés) à afficher dans les boîtes de dialogue ou les fenêtres de sortie ;
- des onglets qui définissent les ensembles de contrôles de propriétés (pour les boîtes de dialogue ou les fenêtres de sortie).

*Remarque* : Dans les définitions d'éléments suivantes (généralement identifiées par l'en-tête **Format**), les attributs d'éléments et les éléments enfants sont facultatifs sauf s'ils sont indiqués par «obligatoire». Pour obtenir la syntaxe complète de tous les éléments, reportez-vous à l'annexe A, *Schéma XML CLEF* sur p. 252.

Pour chaque interface utilisateur, vous indiquez le traitement à effectuer. Pour ce faire, utilisez un gestionnaire d'actions ou un attribut de classe de trame, qui sont tous deux facultatifs. Si aucun de ces deux éléments n'est précisé, le traitement à effectuer est indiqué ailleurs dans le fichier.

### **Format**

Le format de base de l'élément `UserInterface` est :

```
<UserInterface>
  <Icons>
    <Icon ... />
    ...
  </Icons>
  <Controls>
    <Menu ... />
    <MenuItem ... />
    ...
    <ToolBarItem ... />
    ...
  </Controls>
  <Tabs>
    <Tab ... />
    ...
  </Tabs>
</UserInterface>
```

Un **gestionnaire d'actions** est utilisé là où vous ajoutez des actions personnalisées à une fenêtre IBM® SPSS® Modeler standard. Le gestionnaire d'actions indique la classe Java appelée lorsqu'un utilisateur choisit une option de menu ou un bouton de barre d'outils personnalisé sur une boîte de dialogue de noeud, une fenêtre de sortie de modèle ou une fenêtre de sortie de document. Il s'agit d'une implémentation d'une classe `ExtensionObjectFrame` ou `ActionHandler`. Dans les deux cas, les composants standard de la fenêtre sont inclus automatiquement, par exemple : les menus standard, les onglets et les boutons de barre d'outils. [Pour plus d'informations, reportez-vous à la section Classes de l'API côté client dans le chapitre 9 sur p. 217.](#)

Le format d'un gestionnaire d'actions est identique au format de base sauf pour la première ligne :

```
<UserInterface actionHandler="Java_class"
">
  ...
</UserInterface>
```

où `actionHandler` est le nom de la classe Java du gestionnaire d'actions.

Une **classe de cadre** est utilisée pour les objets sortie de document ou sortie de modèle là où l'extension fournit sa propre fenêtre plutôt que de personnaliser une fenêtre SPSS Modeler standard. Une classe de cadre est une classe Java qui indique la totalité de la fenêtre et le traitement associé. Aucun composant standard de la fenêtre n'est inclus automatiquement, la classe doit les préciser individuellement. Les classes de cadre peuvent être utilisées pour les objets de sortie de modèle ou de sortie de document et pas pour les noeuds, qui utilisent toujours les



boîtes de dialogue SPSS Modeler. [Pour plus d'informations, reportez-vous à la section Fenêtres de sortie personnalisées sur p. 201.](#)

Le format d'une classe de cadre est simplement :

```
<UserInterface frameClass="Java_class" />
```

où `frameClass` est le nom de la classe de cadre pour un objet de sortie de modèle ou de sortie de document. Les icônes, les contrôles et les onglets sont spécifiés par la classe de cadre elle-même, ils ne sont pas utilisés dans ce format.

Les éléments enfants d'un élément `UserInterface` sont décrits dans les sections suivantes.

### Exemples

Le premier exemple indique l'interface utilisateur pour un noeud concepteur de modèle :

```
<UserInterface actionHandler="com.spss.myextension.MyActionHandler">
  <Icons>
    <Icon type="standardNode" imagePath="images/lq_discriminant.gif" />
    <Icon type="smallNode" imagePath="images/sm_discriminant.gif" />
  </Icons>
  <Tabs defaultTab="1">
    ...
  </Tabs>
</UserInterface>
```

La section correspondante pour un objet sortie de modèle est :

```
<UserInterface>
  <Icons>
    <Icon type="standardWindow" imagePath="images/browser_discriminant.gif" />
  </Icons>
  <Tabs>
    <Tab label="Avancé" labelKey="advancedTab.LABEL"
      helpLink="discriminant_output_advancedtab.htm">
      <ExtensionObjectPanel id="DiscriminantPanel"
        panelClass="com.spss.clef.discriminant.DiscriminantPanel"/>
    </Tab>
  </Tabs>
</UserInterface>
```

La section Interface utilisateur pour un noeud applicateur de modèle ressemble à ceci :

```
<UserInterface>
  <Icons>
    <Icon type="standardNode" imagePath="images/lq_gm_discriminant.gif" />
    <Icon type="smallNode" imagePath="images/sm_gm_discriminant.gif" />
  </Icons>
  <Tabs>
    <Tab label="Avancé" labelKey="advancedTab.LABEL"
      helpLink="discriminant_output_advancedtab.htm">
      <ExtensionObjectPanel id="DiscriminantPanel"

```

```

        panelClass="com.spss.clef.discriminant.DiscriminantPanel"/>
    </Tab>
</Tabs>
</UserInterface>

```

## Icônes

Cette section définit les icônes associées à l'objet.

Pour les noeuds, cette section doit définir deux éléments `Icon` :

- Une grande version pour l'affichage sur la zone de dessin
- Une petite version pour l'affichage sur la palette

Pour les sorties de modèles et de documents, ceci définit l'icône miniature qui figure en haut à gauche du cadre de la fenêtre.

[Conception d'icônes de noeud sur p. 17](#) offre des directives pour la création d'icônes dans CLEF.

### Format

```




<Icons>
  <Icon type="icon_type" imagePath="image_path" />
  ...
</Icons>

```

où :

`type` (obligatoire) est l'un de ces types d'icônes :

Table 6-1  
Types d'icônes

Type	Exemple	Description
standardNode		L'icône grande taille (49 x 49 pixels) affichée dans la forme du noeud sur l'espace de travail.
smallNode		L'icône petite taille (38 x 38 pixels) affichée dans la forme du noeud sur la palette de noeuds.
fenêtre		L'icône miniature (16 x 16 pixels) affichée dans un navigateur ou dans une fenêtre de sortie.

`imagePath` (obligatoire) identifie l'emplacement de l'image utilisée dans cette icône. Cet emplacement est indiqué par rapport au répertoire d'installation du fichier de spécifications.

### Exemples

```

<Icon type="standardNode" imagePath="images/lg_mynode.gif" />
<Icon type="smallNode" imagePath="images/sm_mynode.gif" />
<Icon type="window" imagePath="images/mynode16.gif" />

```

## Controls

Cette section définit les éléments de barre d'outils et de menu personnalisés utilisés pour invoquer des actions déclarées dans la section Common Objects du fichier de spécifications. [Pour plus d'informations, reportez-vous à la section Actions dans le chapitre 4 sur p. 50.](#)

### Format

```
<Controls>
  <Menu ... />
  ...
  <MenuItem ... />
  ...
  <ToolBarItem ... />
  ...
</Controls>
```

Les éléments `Menu`, `MenuItem` et `ToolBarItem` sont décrits dans les sections suivantes.

### Exemple

L'exemple suivant ajoute un élément au menu Générer et à la barre d'outils de la boîte de dialogue de la spécification dans laquelle il est inclus. Les deux éléments mettent en œuvre une action définie auparavant nommée `generateDerive` qui génère un noeud Calculer.

```
<Controls>
  <MenuItem action="generateDerive" systemMenu="generate"/>
  <ToolBarItem action="generateDerive" showLabel="false"/>
  ...
</Controls>
```

## Menus

Vous pouvez définir un menu personnalisé pour l'ajouter à l'un des menus standard.

### Format

```
<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="entier" mnemonic="mnemonic_char" mnemonicKey="mnemonic_key"/>
```

où :

`id` (obligatoire) est un identificateur unique pour le menu que vous ajoutez.

`label` (obligatoire) est le nom d'affichage pour le menu tel qu'il s'affiche sur l'interface utilisateur (à condition que l'option `showLabel` soit définie sur `true` (vrai)).

`labelKey` identifie l'étiquette à des fins de localisation.

`systemMenu` (obligatoire) identifie le menu standard auquel le menu personnalisé doit être ajouté. La valeur de `menu_name` est l'une des suivantes :

- SAS
- éditer
- insert\*
- affichage\*
- outils\*
- fenêtre\*
- générer
- aide\*
- file.project
- file.outputs
- file.models
- edit.stream
- edit.node
- edit.outputs
- edit.models
- edit.project
- tools.repository
- tools.options
- tools.streamProperties

\* valide uniquement si une fenêtre principale IBM® SPSS® Modeler est ajoutée

`showLabel` spécifie si l'étiquette d'affichage de l'élément doit apparaître sur l'interface utilisateur.

`showIcon` spécifie si l'icône associée à l'élément doit apparaître sur l'interface utilisateur.

`separatorBefore` spécifie si un séparateur (par exemple une barre horizontale pour les éléments du menu ou un espace pour les boutons de la barre d'outils) doit être ajouté au menu avant ce nouvel élément.

`separatorAfter` spécifie si un séparateur doit être ajouté au menu après ce nouvel élément.

`offset` est un entier non négatif qui définit la position du nouvel élément ; par exemple, un décalage de 0 l'ajoute en tant que premier élément (par défaut, il est ajouté à la fin).

le `mnemonic` est le caractère alphabétique utilisé en association avec la touche Alt pour activer cette commande (par exemple, si vous donnez la valeur S, l'utilisateur peut activer cette commande au moyen de Alt+S).

`mnemonicKey` identifie le mnemonic à des fins de localisation. Si vous n'utilisez ni `mnemonic`, ni `mnemonicKey`, aucun mnemonic n'est disponible pour cette commande. [Pour plus d'informations, reportez-vous à la section Touches d'accès et raccourcis clavier sur p. 142.](#)

## Éléments de menu

Vous pouvez définir un élément de menu personnalisé pour l'ajouter à l'un des menus standard ou personnalisés.

### Format

```
<Menuitem action="identifiant" systemMenu="menu_name" customMenu="menu_name"
  showLabel="true_false" showIcon="true_false" separatorBefore="true_false"
  separatorAfter="true_false" offset="entier" />
```

où :

`action` (obligatoire) est l'identificateur d'une action définie dans la section Common Objects et indique l'action devant être effectuée par cet élément de menu.

`systemMenu` indique que l'élément doit figurer dans le menu standard identifié par `menu_name`, qui est l'une des valeurs suivantes :

- SAS
- éditer
- insert\*
- affichage\*
- outils\*
- fenêtre\*
- générer
- aide\*
- file.project
- file.outputs
- file.models
- edit.stream
- edit.node
- edit.outputs
- edit.models
- edit.project
- tools.repository
- tools.options
- tools.streamProperties

\* valide uniquement si une fenêtre principale IBM® SPSS® Modeler est ajoutée

`customMenu` est un identificateur d'un élément Menu et spécifie que l'élément de menu doit s'afficher dans ce menu personnalisé.

`showLabel` spécifie si l'étiquette d'affichage de l'élément doit apparaître sur l'interface utilisateur.

`showIcon` spécifie si l'icône associée à l'élément doit apparaître sur l'interface utilisateur.

`separatorBefore` spécifie si un séparateur (par exemple une barre horizontale pour les éléments du menu ou un espace pour les boutons de la barre d'outils) doit être ajouté au menu avant ce nouvel élément.

`separatorAfter` spécifie si un séparateur doit être ajouté au menu après ce nouvel élément.

`offset` est un entier non négatif qui définit la position du nouvel élément ; par exemple, un décalage de 0 l'ajoute en tant que premier élément (par défaut, il est ajouté à la fin).

### **Exemple**

```
<MenuItem action="generateSelect" systemMenu="generate" showIcon="true"/>
```

## **Éléments de barre d'outils**

Vous pouvez définir un élément de barre d'outils personnalisé pour une boîte de dialogue ou une fenêtre de sortie.

### **Format**

```
<ToolBarItem action="action" showLabel="true_false" showIcon="true_false"
  separatorBefore="true_false" separatorAfter="true_false" offset="entier" />
```

où :

`action` (obligatoire) est l'identificateur d'une action définie dans la section Common Objects et indique l'action devant être effectuée par cet élément de barre d'outils.

`showLabel` spécifie si l'étiquette d'affichage de l'élément doit apparaître sur l'interface utilisateur.

`showIcon` spécifie si l'icône associée à l'élément doit apparaître sur l'interface utilisateur.

`separatorBefore` spécifie si un séparateur (par exemple une barre horizontale pour les éléments du menu ou un espace pour les boutons de la barre d'outils) doit être ajouté au menu avant ce nouvel élément.

`separatorAfter` spécifie si un séparateur doit être ajouté au menu après ce nouvel élément.

`offset` est un entier non négatif qui définit la position du nouvel élément ; par exemple, un décalage de 0 l'ajoute en tant que premier élément (par défaut, il est ajouté à la fin).

**Exemple**

```
<ToolBarItem action="generateDerive" showLabel="true"/>
```

**Exemple : Ajout à la fenêtre principale**

Ceci est un exemple d'un fichier de spécifications qui ajoute un nouvel élément au menu Outils dans la fenêtre principale. Il ne définit pas des objets standard (par exemple, noeud, fenêtre Sortie de modèle ou Sortie de document) mais comprend un élément `UserInterface` dans l'extension `Extension` de niveau supérieur qui signifie «modifier la fenêtre principale». Toutes les autres sections `UserInterface` doivent figurer dans l'une des définitions des objets standard et affectent les boîtes de dialogue associées à ces objets.

```
<?xml version="1.0" encoding="UTF-8"?>
<Extension version="1.0">
  <ExtensionDetails providerTag="example" id="main_window" label="Fenêtre principale" version="1.0"
    provider="IBM Corp." copyright="(c) 2005-2006 IBM Corp."
    description="Un exemple d'extension qui se connecte à la fenêtre principale"/>
  <Resources/>
  <CommonObjects>
    <Actions>
      <Action id="customTool1" label="Outil personnalisé..." labelKey="customTool.LABEL"
        imagePath="images/generate.gif" description="Exécute l'outil personnalisé"
        descriptionKey="customTool.TOOLTIP"/>
      <Action id="customTool2" label="Outil personnalisé..." labelKey="customTool.LABEL"
        imagePath="images/generate.gif" description="Exécute l'outil personnalisé"
        descriptionKey="customTool.TOOLTIP"/>
    </Actions>
  </CommonObjects>
  <UserInterface actionHandler="com.spss.cleftest.MainWindowActionHandler">
    <Controls>
      <Menu systemMenu="tools" id="toolsExtension" separatorBefore="true"
        label="Éléments d'extension" 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>
```

Ceci permet d'ajouter un sous-menu nommé *Eléments d'extension* au menu *Outils*. Ce nouveau sous-menu comprend une entrée unique nommée *Outils personnalisé* :

Figure 6-4  
*Extension des menus standard*



Vous pouvez tester cet exemple en enregistrant le code XML dans un fichier nommé *extension.xml* et en ajoutant cette extension à CLEF. [Pour plus d'informations, reportez-vous à la section Test d'une extension CLEF dans le chapitre 10 sur p. 246.](#)

## Onglets

La section `Tabs` définit les onglets qui peuvent s'afficher sur :

- la boîte de dialogue affichée lorsque l'utilisateur ouvre un noeud dans l'espace de travail ;
- une fenêtre de sortie de modèle ;
- une fenêtre de sortie de document.

Chaque section `Tabs` peut contenir plusieurs éléments `Tab`, dont chacun déclare un onglet personnalisé à afficher :

```
<Tabs defaultTab="entier">
  <Tab ... />
  <Tab ... />
  ...
</Tabs>
```

où `defaultTab` est un entier non négatif qui spécifie l'onglet à afficher lorsque la boîte de dialogue ou la fenêtre du noeud est ouverte pour la première fois dans un flux. Si l'utilisateur sélectionne un onglet différent, à la fermeture et la réouverture de la boîte de dialogue ou de la fenêtre alors que le flux est actif, le dernier onglet consulté est affiché au lieu de celui par défaut. La numérotation des onglets commence à 0.

Notez que les autres onglets sont susceptibles d'être inclus automatiquement dans la boîte de dialogue ou dans la fenêtre, par exemple, toutes les boîtes de dialogue et les fenêtres de sortie incluent un onglet `Annotations`, et toutes les boîtes de dialogue de noeud source de données incluent les onglets `Filtrer` et `Types`.

Un élément `Tab` doit déclarer l'étiquette de l'onglet (le texte qui s'affiche sur l'onglet lui-même). En outre, il doit comprendre une clé d'étiquette à utiliser comme référence si l'étiquette de l'onglet doit être traduit.



Dans chaque élément `Tab` figure une spécification de panneau, qui définit la présentation de la zone de contenu principale de l'onglet. La spécification de panneau peut être l'un des trois types suivants : navigateur de texte, objet d'extension ou propriétés. [Pour plus d'informations, reportez-vous à la section Spécifications de panneau sur p. 143.](#)

Le format d'un élément `Tab` individuel est :

```
<Tab id="identifiant" label="display_label" labelKey="label_key" helpLink="help_ID"
    mnemonic="mnemonic_char" mnemonicKey="mnemonic_key" >
  <TextBrowserPanel ... />
  <ExtensionObjectPanel ... />
  <PropertiesPanel ... />
  <ModelViewerPanel ... />
</Tab>
```

où :

`id` est un identificateur unique qui peut être utilisé pour référencer l'onglet dans un code Java.

`label` (obligatoire) est le nom d'affichage de l'onglet tel qu'il s'affiche sur l'interface utilisateur.

`labelKey` identifie l'étiquette à des fins de localisation.

`helpLink` est l'identifiant d'une rubrique d'aide à afficher lorsque l'utilisateur appelle le système d'aide, s'il est présent. Le format de l'identifiant dépend du type du système d'aide (voir [Définition de l'emplacement et du type du système d'aide sur p. 205](#)) :

Pour l'aide au format HTML : URL de la rubrique d'aide

Pour JavaHelp : ID de la rubrique

Pour l'aide native

Le `mnemonic` est le caractère alphabétique utilisé en association avec la touche `Alt` pour activer cette commande (par exemple, si vous donnez la valeur `S`, l'utilisateur peut activer cette commande au moyen de `Alt+S`).

`mnemonicKey` identifie le `mnemonic` à des fins de localisation. Si vous n'utilisez ni `mnemonic`, ni `mnemonicKey`, aucun `mnemonic` n'est disponible pour cette commande. [Pour plus d'informations, reportez-vous à la section Touches d'accès et raccourcis clavier sur p. 142.](#)

### **Exemples**

Pour obtenir des exemples d'éléments `Tab`, consultez les sections sur les différents types de spécification de panneau qu'ils peuvent contenir : [Panneau Navigateur de texte sur p. 144](#), [Panneau Objet d'extension sur p. 146](#), [Panneau Propriétés sur p. 147](#), et [Panneau Visualiseur de modèle sur p. 149](#).

## Touches d'accès et raccourcis clavier

Plutôt que d'accéder à des fonctions à l'aide de la souris dans l'interface utilisateur, vous pouvez spécifier plusieurs combinaisons de touches afin de permettre aux utilisateurs d'accéder à des fonctions au moyen du clavier.

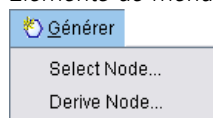
### Touches d'accès

Les touches d'accès sont des touches qui peuvent être utilisées en association avec la touche Alt. Pour des éléments de menus, des onglets de boîte de dialogue et plusieurs autres commandes de boîte de dialogue, vous pouvez spécifier des touches d'accès à l'aide de l'attribut mnemonic des éléments suivants.

Fonction	Élément	Voir...
Action à l'écran (par ex. pour un élément de menu)	action	<a href="#">Actions sur p. 50</a>
Menu	menu	<a href="#">Menus sur p. 135</a>
Onglet Boîte de dialogue	onglet	<a href="#">Onglets sur p. 140</a>
Contrôleurs	Divers	<a href="#">Contrôleurs sur p. 158</a>

Par exemple, pour spécifier des touches d'accès pour des éléments dans le menu suivant :

Figure 6-5  
Éléments de menu

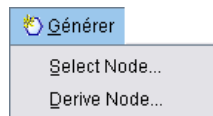


vous devez utiliser le code suivant :

```
<Actions>
  <Action id="generateSelect" label="Sélectionner noeud..." labelKey="generate.selectNode.LABEL"
    imagePath="images/generate.gif" description="Crée un nœud Sélectionner"
    descriptionKey="generate.selectNode.TOOLTIP" mnemonic="S" />
  <Action id="generateDerive" label="Noeud Calculer..." labelKey="generate.deriveNode.LABEL"
    imagePath="images/generate.gif" description="Génère un noeud Calculer"
    descriptionKey="generate.deriveNode.TOOLTIP" mnemonic="D" />
  ...
</Actions>
```

Cela vous donne les caractères de soulignement dans les éléments du menu :

Figure 6-6  
Utilisation de touches d'accès avec des éléments de menu



Les utilisateurs peuvent désormais accéder aux éléments de menu à l'aide de Alt+S ou de Alt+D, ainsi qu'en cliquant dessus avec la souris.

### Touches de raccourci

Les touches de raccourcis sont des combinaisons de touches qui permettent aux utilisateurs finaux de naviguer dans l'interface utilisateur. IBM® SPSS® Modeler offre plusieurs raccourcis clavier standard. Dans CLEF, vous pouvez ajouter des raccourcis uniquement à des éléments de menus personnalisés que vous avez ajoutés.

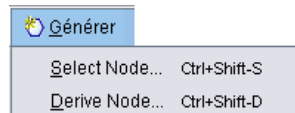
Par exemple, pour spécifier des raccourcis pour des éléments du menu affiché dans l'exemple des touches d'accès, vous devez utiliser le code suivant :

```
<Actions>
  <Action id="generateSelect" label="Sélectionner noeud..." labelKey="generate.selectNode.LABEL"
    imagePath="images/generate.gif" description="Crée un noeud Sélectionner"
    descriptionKey="generate.selectNode.TOOLTIP" mnemonic="S" shortcut="CTRL+SHIFT+S" />
  <Action id="generateDerive" label="Noeud Calculer..." labelKey="generate.deriveNode.LABEL"
    imagePath="images/generate.gif" description="Génère un noeud Calculer"
    descriptionKey="generate.deriveNode.TOOLTIP" mnemonic="D" shortcut="CTRL+SHIFT+D" />
  ...
</Actions>
```

Les combinaisons de touches de raccourcis sont désormais ajoutées aux éléments du menu :

Figure 6-7

Utilisation de touches de raccourci avec des éléments de menu



Les utilisateurs peuvent désormais accéder aux éléments de menu à l'aide des raccourcis clavier, ainsi qu'en cliquant dessus avec la souris. Vous pouvez spécifier des touches de raccourci en association avec des touches d'accès, comme dans cet exemple.

Veillez à ne pas utiliser de raccourcis qui ont déjà été spécifiés dans la même boîte de dialogue, ni aucun des raccourcis standard de SPSS Modeler. [Pour plus d'informations, reportez-vous à la section Accessibilité via le clavier dans l'annexe A dans Guide de l'utilisateur de IBM SPSS Modeler 14.2.](#)

## Spécifications de panneau

Chaque élément Tab peut contenir la spécification d'un seul panneau, qui peut être l'un des types suivants :

Table 6-2

Types de spécifications de panneau

Panneau	Affiche...	Voir...
Panneau Navigateur de texte	Contenu de texte d'un conteneur spécifié.	<a href="#">Panneau Navigateur de texte sur p. 144</a>
Panneau Objet d'extension	Contenu défini par la classe Java indiquée.	<a href="#">Panneau Objet d'extension sur p. 146</a>

Panneau	Affiche...	Voir...
Panneau Propriétés	Contrôles de propriété (par exemple, boutons, cases à cocher, champs d'entrée).	<a href="#">Panneau Propriétés sur p. 147</a>
Panneau Visualiseur de modèle	Sortie de modèle au format PMML d'un conteneur précisé.	<a href="#">Panneau Visualiseur de modèle sur p. 149</a>

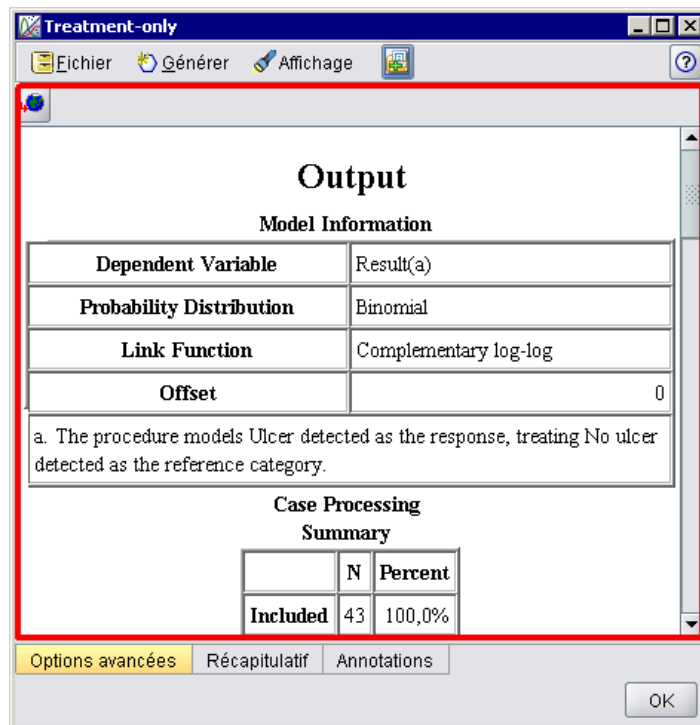
## Panneau Navigateur de texte

Un panneau navigateur de texte affiche le texte d'un conteneur spécifié dans l'extension. Les formats pris en charge (codage UTF-8) sont le texte brut, HTML et Rich Text Format (RTF).

Vous trouverez ci-après un exemple de fenêtre de sortie de modèle contenant un panneau navigateur de texte au format HTML :

Figure 6-8

Fenêtre de sortie de modèle avec panneau navigateur de texte sélectionné



### Format

```
<TextBrowserPanel container="name" textFormat="text_format" rows="entier"
  columns="entier" wrapLines="true_false" >
  -- options de présentation personnalisée avancées --
</TextBrowserPanel>
```

où :

`container` (obligatoire) est le nom du conteneur à partir duquel obtenir le contenu du panneau.

`textFormat` (obligatoire) indique le format du texte affiché dans le panneau et est l'un des suivants :

- `plainText`
- `html`
- `rtf`

Les `lines` et les `columns` indiquent le nombre de lignes et de colonnes de texte visibles lors de l'ouverture de la fenêtre contenant le panneau.

`wrapLines` indique s'il faut utiliser ou non le retour automatique à la ligne pour les longues lignes de texte (`true` (vrai)) ou requérir le défilement horizontal pour lire des lignes de texte longues (`false` (faux)). La valeur par défaut est `false` (faux).

Les options de présentation personnalisée avancées fournissent un fin degré de contrôle du positionnement et de l'affichage des composants de l'écran. [Pour plus d'informations, reportez-vous à la section Présentation personnalisée avancée sur p. 188.](#)

### **Exemple**

L'exemple suivant illustre la section `Tab` qui définit le panneau navigateur de texte indiqué précédemment :

```
<Tab label="Avancé" labelKey="advancedTab.LABEL" helpLink="genlin_output_advancedtab.htm">
  <TextBrowserPanel container="advanced_output" textFormat="html" />
</Tab>
```

La sortie de modèle est envoyée à un conteneur qui est défini dans la même section `ModelOutput` que la spécification de l'onglet :

```
<ModelOutput id="generalizedlinear" label="Linéaire généralisé">
  <Containers>
    ...
    <Container name="advanced_output"/>
  </Containers>
  <UserInterface>
    ...
    <Tabs>
      <Tab label="Avancé" ... >
        <TextBrowserPanel container="advanced_output" ... />
      </Tab>
    </Tabs>
  </UserInterface>
</ModelOutput>
```

Le conteneur est référencé dans un élément CreateDocument dans la section Execution pour le noeud de création pertinent :

```
<Execution>
...
<Constructors>
  <CreateModelOutput type="generalizedlinear">
    <CreateModel type="generalizedlinear_model" target="model" sourceFile="pmm1"/>
    <CreateDocument type="html_output" target="advanced_output" sourceFile="htmloutput"/>
  </CreateModelOutput>
</Constructors>
</Execution>
```

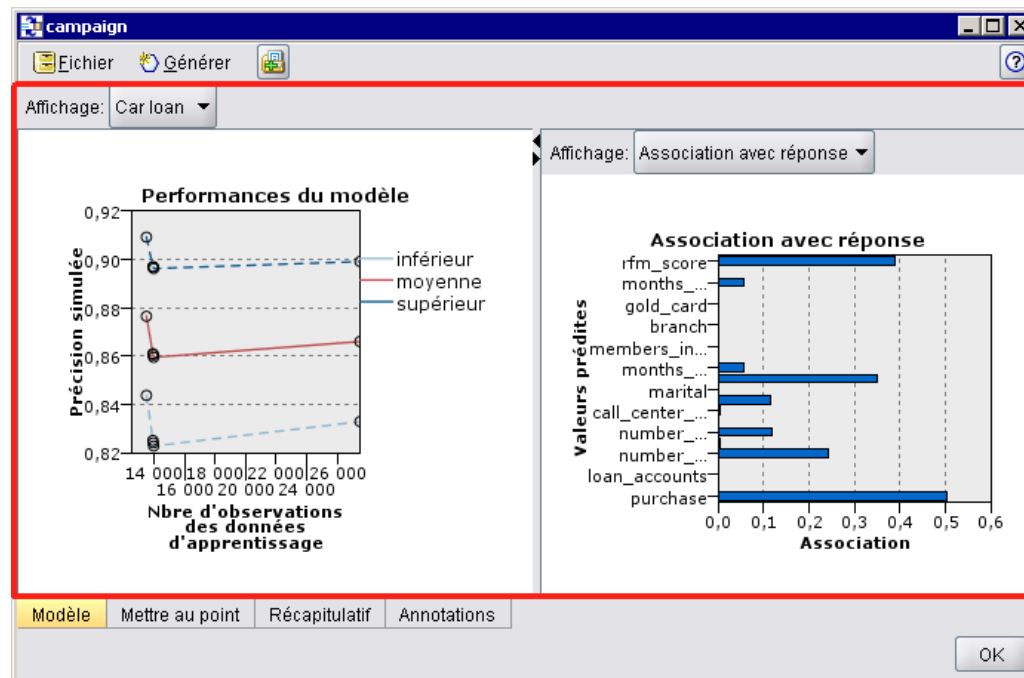
## Panneau Objet d'extension

Un panneau objet d'extension fonctionne de la même façon qu'un panneau navigateur de texte. Cependant, au lieu d'afficher le contenu au format texte d'un conteneur, il crée une instance d'une classe Java spécifiée qui met en œuvre l'interface ExtensionObjectPanel définie par l'API Java de CLEF.

Vous trouverez ci-après un exemple de boîte de dialogue noeud applicateur de modèle contenant un panneau d'objet d'extension :

Figure 6-9

Fenêtre de sortie de modèle avec panneau objet d'extension sélectionné



### **Format**

```
<ExtensionObjectPanel id="identifiant" panelClass="Java_class" >  
  -- options de présentation personnalisée avancées --  
</ExtensionObjectPanel>
```

où :

id est un identificateur unique qui peut être utilisé pour référencer le panneau dans un code Java.

panelClass (obligatoire) est le nom d'une classe Java que le panneau met en œuvre.

Les options de présentation personnalisée avancées fournissent un fin degré de contrôle du positionnement et de l'affichage des composants de l'écran. [Pour plus d'informations, reportez-vous à la section Présentation personnalisée avancée sur p. 188.](#)

### **Exemple**

L'exemple suivant illustre la section Onglet qui définit le panneau objet d'extension indiqué précédemment :

```
<Tab label="Modèle" labelKey="Model.LABEL" helpLink="selflearnnode_output.htm">  
  <ExtensionObjectPanel id="SelfLearningPanel"  
    panelClass="com.spss.clef.selflearning.SelfLearningPanel"/>  
</Tab>
```

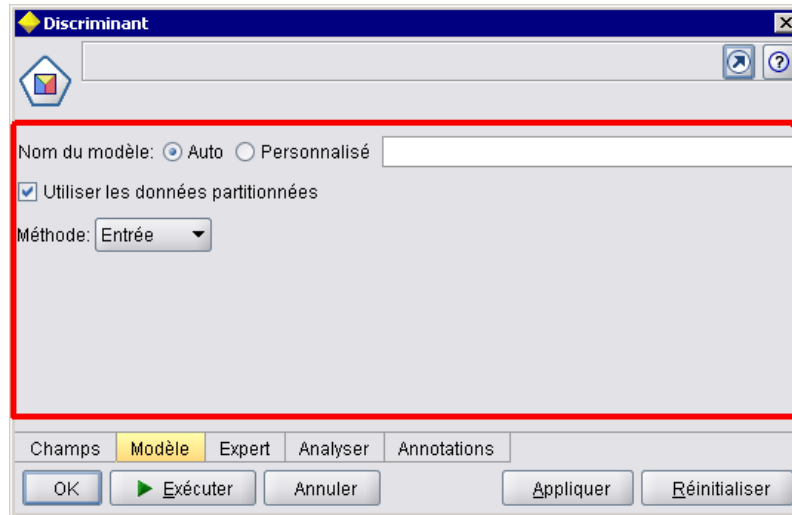
## **Panneau Propriétés**

Un panneau Propriétés permet à un onglet ou à un sous-panneau Propriétés (consultez la rubrique [Sous-panneau Propriétés sur p. 155](#)) d'afficher les **contrôles de propriétés**, qui sont des composants d'écran (boutons, cases à cocher, champs d'entrée, etc.) qui peuvent permettre de modifier les propriétés d'un objet affiché à l'écran. Le panneau Propriétés applique automatiquement les modifications effectuées avec ces contrôles lorsque l'utilisateur clique sur OK ou Appliquer. Lorsque l'utilisateur clique sur Annuler ou sur Réinitialiser, le panneau supprime toutes les modifications apportées depuis la dernière opération d'application.

Vous trouverez ci-après un exemple de noeud contenant un panneau Propriétés :

Figure 6-10

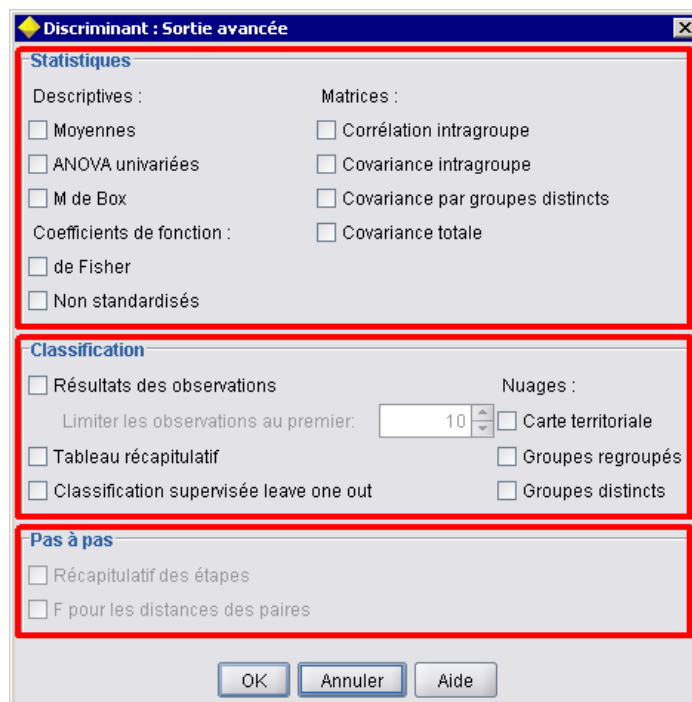
Boîte de dialogue de noeud avec le panneau Propriétés sélectionné



L'exemple suivant illustre un sous-panneau Propriétés contenant trois panneaux de propriétés :

Figure 6-11

Sous-panneau Propriétés avec panneaux de propriétés sélectionnés





### **Format**

```
<PropertiesPanel id="identifiant" label="display_label" labelKey="label_key">  
  -- options de présentation personnalisée avancées --  
  -- spécifications des contrôles de propriétés --  
</PropertiesPanel>
```

où :

id est un identificateur unique qui peut être utilisé pour référencer le panneau dans un code Java.

label est l'en-tête d'affichage pour un groupe de contrôles de propriétés (par exemple, Statistiques, Classification et Pas à pas dans le dernier exemple).

labelKey identifie l'étiquette à des fins de localisation.

Les options de présentation personnalisée avancées fournissent un fin degré de contrôle du positionnement et de l'affichage des composants de l'écran. [Pour plus d'informations, reportez-vous à la section Présentation personnalisée avancée sur p. 188.](#)

Les spécifications de contrôles de propriétés individuelles sont décrites dans [Spécifications des contrôles de propriétés sur p. 151.](#)

### **Exemple**

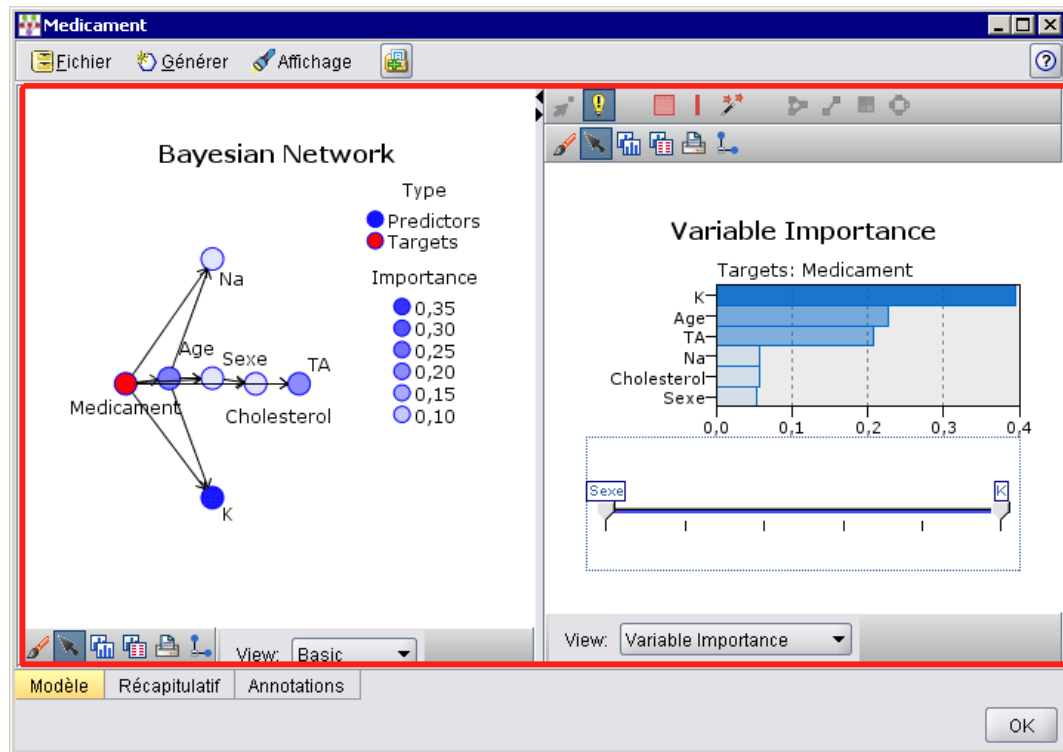
```
<Tab label="Modèle" labelKey="Model.LABEL" helpLink="discriminant_node_model.htm">  
  <PropertiesPanel>  
    <SystemControls controlsId="ModelGeneration" />  
    <ComboBoxControl property="method">  
      <Layout fill="none" />  
    </ComboBoxControl>  
  </PropertiesPanel>  
</Tab>
```

## **Panneau Visualiseur de modèle**

Un panneau afficheur de modèle affiche toute sortie de modèle au format PMML à partir d'un conteneur spécifié dans l'extension.

Vous trouverez ci-après un exemple de fenêtre noeud de programme d'application de modèle contenant un panneau afficheur de modèle.

Figure 6-12  
Fenêtre de sortie avec panneau afficheur de modèle sélectionné



### Format

```
<ModelViewerPanel container="container_name">
  -- options de présentation personnalisée avancées --
</ModelViewerPanel>
```

où container (requis) est le nom du conteneur auquel la sortie du modèle est attribuée.

Les options de présentation personnalisée avancées fournissent un fin degré de contrôle du positionnement et de l'affichage des composants de l'écran. [Pour plus d'informations, reportez-vous à la section Présentation personnalisée avancée sur p. 188.](#)

### Exemple

Cet exemple illustre l'utilisation d'un panneau afficheur de modèle dans une spécification d'application de modèle. La sortie de modèle a été auparavant affectée au conteneur nommé modèle. Ici, la spécification de l'application de modèle récupère ce conteneur et l'associe au panneau afficheur de modèle :

```
<Node id="applyBN" type="modelApplier">
  <ModelProvider container="modèle" isPMML="true" />
  ...
  <Containers>
    <Container name="modèle" />
  </Containers>
```

```

</Containers>
<UserInterface>
...
<Tabs>
  <Tab label="Modèle" labelKey="modelTab.LABEL" helpLink="BN_output_modeltab.htm">
    <ModelViewerPanel container="modèle"/>
  </Tab>
...
</Tabs>
</UserInterface>
...
</Node>

```

## Spécifications des contrôles de propriétés

Les contrôles de propriétés sont des composants d'écran (boutons, cases à cocher, champs d'entrée, etc.) qui peuvent permettre de modifier les propriétés d'un objet affiché à l'écran. Le format d'une spécification de contrôle propriété dépend du type de contrôle propriété, qui peut être l'un des suivants :

- Composant IU
- Panneau de propriétés
- Contrôleur

Les contrôles du **composant IU** sont des boutons d'action, du texte statique sur l'affichage et les contrôles système (un ensemble de contrôles qui gèrent les propriétés communes à toutes les boîtes de dialogue).

Les contrôles du **panneau de propriétés** sont des panneaux individuels dans la spécification du panneau de propriétés.

Les **contrôleurs** forment le plus grand groupe de contrôles de propriétés. Ils incluent des éléments tels que des cases à cocher, des zones de liste déroulantes ainsi que des contrôles Spinner.

### Contrôles du composant IU

Les contrôles du composant IU sont :

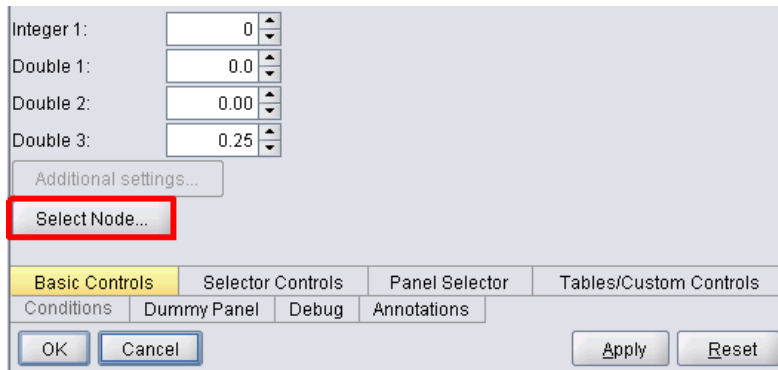
Table 6-3  
Contrôles du composant IU

Contrôle	Description
ActionButton	Un bouton d'écran qui exécute une action prédéfinie lorsque l'utilisateur clique dessus.
StaticText	Une chaîne de texte non variable affichée à l'écran.
SystemControls	Ensembles standard de contrôles qui gèrent les propriétés communes à tous les modèles.

## Bouton d'action

Définit un bouton de boîte de dialogue ou de barre d'outils qui exécute une action spécifiée dans la section Common Objects. L'action (par exemple, l'affichage d'un nouvel écran) est effectuée lorsque l'utilisateur clique sur ce bouton.

Figure 6-13  
Boîte de dialogue avec bouton d'action sélectionné



## Format

```
<ActionButton action="action" showLabel="true_false" showIcon="true_false" >
  -- options de présentation personnalisée avancées --
</ActionButton>
```

où :

action (obligatoire) est l'identificateur de l'action à effectuer.

showLabel spécifie s'il faut ou non afficher (true (vrai)) ou masquer (false (faux)) l'étiquette du bouton (par exemple, pour un bouton de barre d'outils, vous pouvez choisir d'afficher une icône mais pas une étiquette). La valeur par défaut est true (vrai).

showIcon spécifie s'il faut ou non afficher (true (vrai)) ou masquer (false (faux)) une icône associée au bouton. La valeur par défaut est false (faux).

Les options de présentation personnalisée avancées fournissent un fin degré de contrôle du positionnement et de l'affichage des composants de l'écran. [Pour plus d'informations, reportez-vous à la section Présentation personnalisée avancée sur p. 188.](#)

## Exemple

Le code pour créer le bouton d'action indiqué auparavant est :

```
<ActionButton action="generateSelect"/>
```

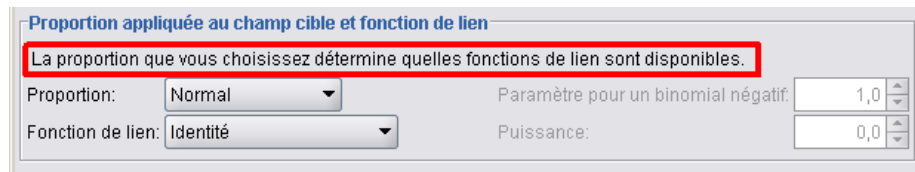
L'action est définie dans la section Common Objects comme suit (notez que l'étiquette du bouton est définie ici également) :

```
<CommonObjects extensionListenerClass="com.spss.cleftest.TestExtensionListener">
...
<Actions>
  <Action id="generateSelect" label="Sélectionner Noeud..." labelKey="generate.selectNode.LABEL"
    imagePath="images/generate.gif" description="Crée un nœud Sélectionner"
    descriptionKey="generate.selectNode.TOOLTIP"/>
...
</Actions>
</CommonObjects>
```

### Texte statique

Cet élément permet d'inclure une chaîne de texte non variable sur une boîte de dialogue ou une fenêtre de sortie. L'exemple suivant illustre un panneau Propriétés comportant un texte statique :

Figure 6-14  
Panneau Propriétés avec texte statique sélectionné



### Format

```
<StaticText text="static_text" textKey="text_key" >
  -- options de présentation personnalisée avancées --
</StaticText>
```

où :

text est la chaîne de texte à utiliser.

textKey identifie le texte statique à des fins de localisation.

Les options de présentation personnalisée avancées fournissent un fin degré de contrôle du positionnement et de l'affichage des composants de l'écran. [Pour plus d'informations, reportez-vous à la section Présentation personnalisée avancée sur p. 188.](#)

### Exemple

L'exemple suivant illustre la déclaration utilisée pour le texte statique indiqué auparavant :

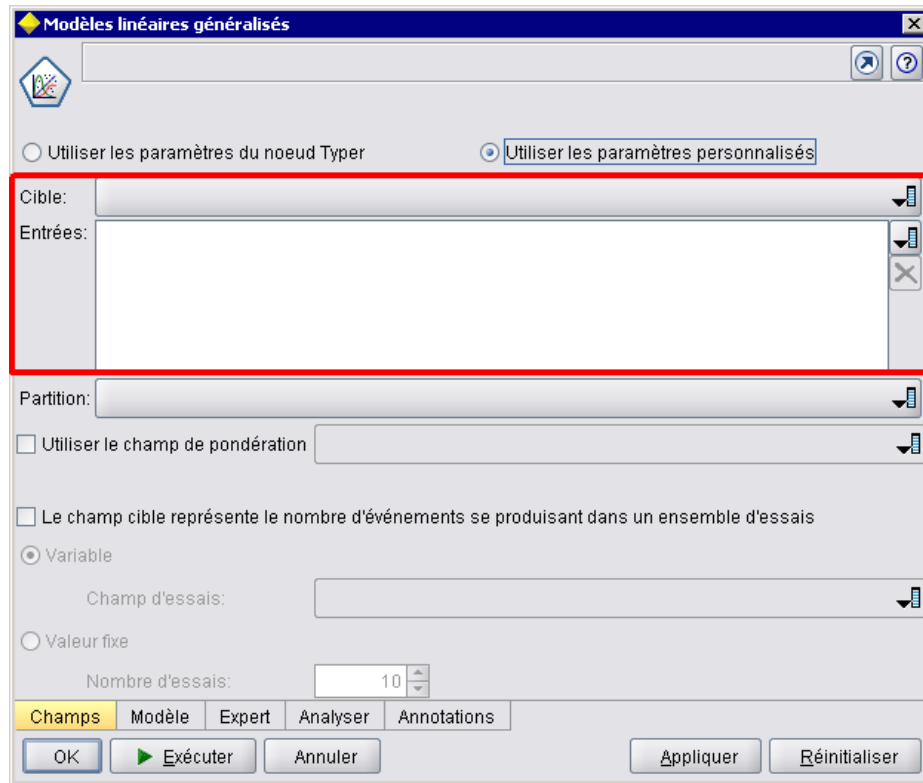
```
<StaticText text="la distribution que vous choisissez détermine les fonctions de liens disponibles."
  textKey="Genlin_staticText1"/>
```

### Contrôles système

Certaines propriétés sont communes à tous les modèles. Dans un noeud de création de modèle, les contrôles système sont des ensembles standard de contrôles qui gèrent ces propriétés.

Figure 6-15

Exemple de boîte de dialogue avec contrôles système sélectionnés



### Format

```
<SystemControls controlsID="identifiant" >
  -- options de présentation personnalisée avancées --
</SystemControls>
```

où controlsID est l'identificateur de l'ensemble des contrôles. L'identificateur doit être le même que celui indiqué dans l'attribut controlsID d'un élément ModelingFields dans une déclaration de création de modèle (reportez-vous à [Concepteur de modèle sur p. 64](#)).

Les options de présentation personnalisée avancées fournissent un fin degré de contrôle du positionnement et de l'affichage des composants de l'écran. [Pour plus d'informations, reportez-vous à la section Présentation personnalisée avancée sur p. 188.](#)

### Exemple

Cet exemple indique la déclaration utilisée pour les contrôles système dans la dernière illustration.

Dans la section Model Builder de la spécification du noeud, l'exemple suivant définit un ensemble de contrôles système, dans ce cas, comprenant les sélecteurs des champs d'entrée et de sortie du modèle :

```
<ModelBuilder ... >
  <ModelingFields controlId="modelingFields">
    <InputFields property="inputs" multiple="true" label="Entrées" types="[range set orderedSet flag]"
labelKey="inputFields.LABEL"/>
    <OutputFields property="target" multiple="false" types="[range flag]"
label="Cible" labelKey="targetField.LABEL"/>
  </ModelingFields>
  ...
</ModelBuilder>
```

Plus loin dans ce fichier, cet ensemble de contrôles est référencé dans la définition de l'onglet pour la boîte de dialogue noeud concepteur de modèle sur lequel ils figurent.

```
<Tab label="Champs" labelKey="Fields.LABEL" helpLink="genlin_node_fieldstab.htm">
  <PropertiesPanel>
    <SystemControls controlId="modelingFields">
    </SystemControls>
    ...
  </PropertiesPanel>
</Tab>
```

## Contrôles du panneau Propriétés

Les contrôles du panneaux Propriétés sont :

Table 6-4  
Contrôles du panneau Propriétés

Contrôle	Description
PropertiesSubPanel	Boîte de dialogue distincte qui s'affiche lorsque l'utilisateur clique sur un bouton d'un panneau Propriétés.
PropertiesPanel	Panneau Propriétés imbriqué dans une déclaration de sous-panneau de propriétés ou dans une déclaration de panneau de propriétés de niveau supérieur.

### Sous-panneau Propriétés

Définit une boîte de dialogue distincte qui s'affiche lorsque l'utilisateur clique sur un bouton d'un panneau Propriétés. La déclaration de sous-panneau de propriétés se compose de la spécification de panneau de propriétés principale pour un onglet.

#### Format

```
<PropertiesSubPanel buttonLabel="display_label" buttonLabelKey="label_key"
dialogTitle="display_title" dialogTitleKey="title_key" helpLink="help_ID"
mnemonic="mnemonic_char" mnemonicKey="mnemonic_key" >
-- options de présentation personnalisée avancées --
-- spécifications des contrôles de propriétés --
```

```
</PropertiesSubPanel>
```

où :

`buttonLabel` est l'étiquette du bouton qui offre l'accès au sous-panneau.

`buttonLabelKey` identifie l'étiquette du bouton à des fins de localisation.

`dialogTitle` est le texte qui doit s'afficher sur la barre de titre de la boîte de dialogue du sous-panneau.

`dialogTitleKey` identifie le titre de la boîte de dialogue du sous-panneau à des fins de localisation.

`helpLink` est l'identifiant d'une rubrique d'aide à afficher lorsque l'utilisateur appelle le système d'aide, s'il est présent. Le format de l'identifiant dépend du type du système d'aide (voir [Définition de l'emplacement et du type du système d'aide sur p. 205](#)) :

Pour l'aide au format HTML : URL de la rubrique d'aide

Pour JavaHelp : ID de la rubrique

Pour l'aide native

le `mnemonic` est le caractère alphabétique utilisé en association avec la touche Alt pour activer cette commande (par exemple, si vous donnez la valeur S, l'utilisateur peut activer cette commande au moyen de Alt+S).

`mnemonicKey` identifie le mnemonic à des fins de localisation. Si vous n'utilisez ni `mnemonic`, ni `mnemonicKey`, aucun mnemonic n'est disponible pour cette commande. [Pour plus d'informations, reportez-vous à la section Touches d'accès et raccourcis clavier sur p. 142.](#)

Les options de présentation personnalisée avancées fournissent un fin degré de contrôle du positionnement et de l'affichage des composants de l'écran. [Pour plus d'informations, reportez-vous à la section Présentation personnalisée avancée sur p. 188.](#)

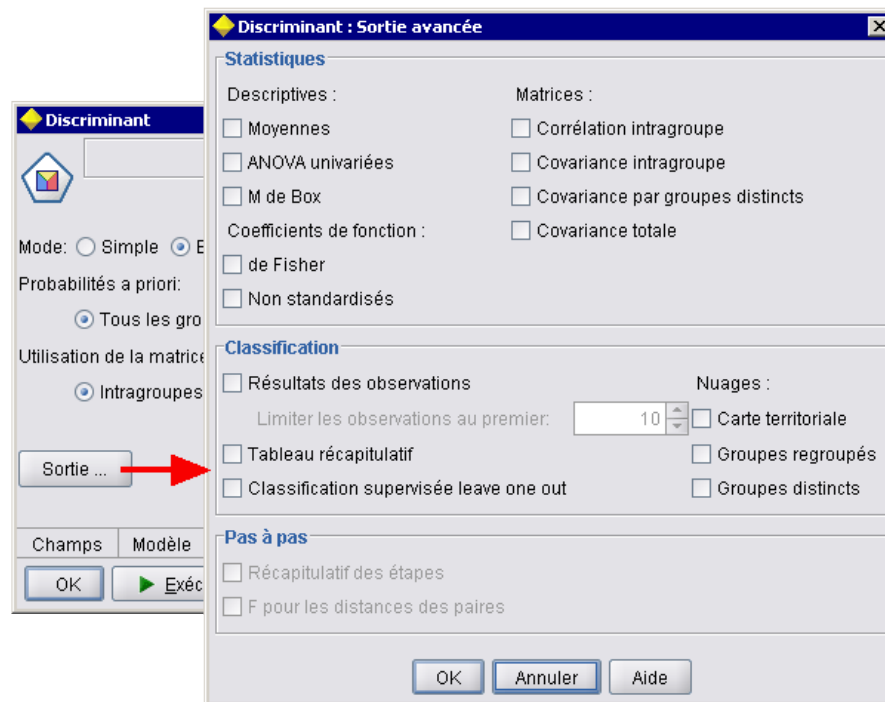
Les spécifications de contrôles de propriétés individuelles sont décrites dans [Spécifications des contrôles de propriétés sur p. 151.](#)

### **Exemple**

L'exemple suivant illustre un sous-panneau de propriétés qui est affiché lorsque l'utilisateur clique sur le bouton Sortie sur le panneau de propriétés principal d'un onglet.



Figure 6-16  
Sous-panneau de propriétés



Le code suivant indique les composants les plus importants de la déclaration utilisée pour obtenir le sous-panneau des propriétés illustré. Notez comment, dans la déclaration du sous-panneau, chacun des groupes de champs (Statistiques, Classification et Pas à pas) comprend sa propre spécification de panneau de propriétés :

```
<PropertiesSubPanel buttonLabel="Sortie..." buttonLabelKey="OutputSubPanel.LABEL"
  dialogTitle="Discriminant: Sortie avancée" dialogTitleKey="AdvancedOutputSubDialog.LABEL"
  helpLink="discriminant_node_outputdlg.htm">
  ...
  <PropertiesPanel>
    <PropertiesPanel label="Statistiques" ... >
    ...
  </PropertiesPanel>
  <PropertiesPanel label="Classification" ... >
  ...
  </PropertiesPanel>
  <PropertiesPanel label="Pas à pas" ... >
  ...
  </PropertiesPanel>
</PropertiesPanel>
</PropertiesSubPanel>
```

### **Panneau Propriétés (imbriqué)**

Vous pouvez imbriquer une spécification de panneau de propriétés dans une déclaration de sous-panneau de propriétés, pour définir le contenu de la boîte de dialogue affichée dans le sous-panneau. [Pour plus d'informations, reportez-vous à la section Sous-panneau Propriétés sur p. 155.](#)

Vous pouvez également imbriquer une spécification de panneau de propriétés dans une déclaration de panneau de propriétés de niveau supérieur. Un exemple illustrant ceci est lorsque le contenu d'un onglet intégral, comprenant plusieurs panneaux de propriétés, est activé ou désactivé en fonction de la sélection d'un bouton spécifique sur l'onglet. Dans ce cas, la spécification de l'onglet ressemble à ceci :

```
<Tab .... >
  <PropertiesPanel>
    --- spécification du bouton ---
    <PropertiesPanel>
      <Enabled>
        --- condition impliquant la valeur du bouton ---
      </Enabled>
    ...
  </PropertiesPanel>
  <PropertiesPanel>
    <Enabled>
      --- condition impliquant la valeur du bouton ---
    </Enabled>
  ...
</PropertiesPanel>
...
</PropertiesPanel>
</Tab>
```

Le format d'une spécification de panneau de propriétés imbriqué est identique à celui pour l'élément de niveau supérieur. [Pour plus d'informations, reportez-vous à la section Panneau Propriétés sur p. 147.](#)

## **Contrôleurs**

Les contrôleurs forment le plus grand groupe des contrôles de propriétés :

Table 6-5  
Contrôleurs

Contrôle	Description
CheckBoxControl	Case à cocher.
CheckBoxGroupControl	Ensemble de cases à cocher, une seule case par valeur enum.
ClientDirectoryChooserControl	Champ de texte à une seule ligne et bouton associé pour permettre à l'utilisateur de sélectionner un répertoire sur le client.
ClientFileChooserControl	Champ de texte à une seule ligne et bouton associé pour permettre à l'utilisateur de sélectionner un fichier sur le client.

Contrôle	Description
ComboBoxControl	Zone de liste déroulante contenant les valeurs enum.
DBConnectionChooserControl	Permet à l'utilisateur de sélectionner une source de données et de se connecter à une base de données.
DBTableChooserControl	Permet à l'utilisateur de sélectionner une table de base de données après une connexion réussie à la base de données.
MultiFieldChooserControl	(Noeuds uniquement) Liste de noms de champs permettant de choisir un ou plusieurs champs dans la liste.
MultiltemChooserControl	Permet à l'utilisateur de sélectionner un ou plusieurs éléments dans une liste de valeurs.
PasswordBoxControl	Champ de texte à une seule ligne où les caractères d'entrée sont masqués.
PropertyControl	Un contrôle personnalisable pour une propriété.
RadioButtonGroupControl	Ensemble de boutons radio où seul un bouton peut être sélectionné à la fois. Pour les propriétés enum, il existe un seul bouton radio par valeur enum ; pour les propriétés Booléennes, deux boutons radio sont affichés.
ServerDirectoryChooserControl	Champ de texte à une seule ligne et bouton associé pour permettre à l'utilisateur de sélectionner un répertoire sur le serveur.
ServerFileChooserControl	Champ de texte à une seule ligne et bouton associé pour permettre à l'utilisateur de sélectionner un fichier sur le serveur.
SingleFieldChooserControl	(Noeuds uniquement) Liste de noms de champs permettant de choisir un seul champ dans la liste.
SingleItemChooserControl	Permet à l'utilisateur de sélectionner un seul élément dans une liste de valeurs.
SpinnerControl	Contrôle Spinner (champs numériques avec flèches Haut et Bas pour modifier la valeur).
TableControl	Ajoute une table à une boîte de dialogue ou à une fenêtre.
TextAreaControl	Champ de texte de plusieurs lignes.
TextBoxControl	Champ de texte d'une seule ligne.

### Attributs du contrôleur

Les spécifications du contrôleur peuvent inclure les attributs suivants :

```
property="valeur" 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"
```

où :

property (obligatoire) est l'identificateur unique du contrôle propriété.

showLabel spécifie s'il faut ou non afficher (true (vrai)) ou masquer (false (faux)) l'étiquette d'affichage du contrôle propriété. La valeur par défaut est true (vrai).

label est le nom d’affichage du contrôle de propriété tel qu’il s’affiche sur l’interface utilisateur. Cette valeur sert aussi de description courte accessible du contrôle propriété. [Pour plus d’informations, reportez-vous à la section Accessibilité dans le chapitre 8 sur p. 215.](#)

labelKey identifie l’étiquette à des fins de localisation.

labelWidth est le nombre de colonnes de la grille d’affichage que l’étiquette couvre. La valeur par défaut est 1.

labelAbove indique si l’étiquette pour le contrôle doit figurer au-dessus de (true (vrai)) ou être adjacente (false (faux)) au contrôle. La valeur par défaut est false (faux).

description est le texte de l’info-bulle affichée lorsque le curseur survole le contrôle. Cette valeur sert aussi de description longue accessible du contrôle propriété. [Pour plus d’informations, reportez-vous à la section Accessibilité dans le chapitre 8 sur p. 215.](#)

descriptionKey identifie la description à des fins de localisation.

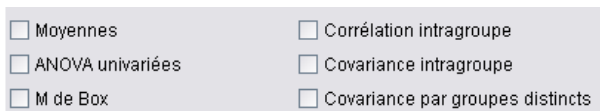
le mnemonic est le caractère alphabétique utilisé en association avec la touche Alt pour activer cette commande (par exemple, si vous donnez la valeur S, l’utilisateur peut activer cette commande au moyen de Alt+S).

mnemonicKey identifie le mnemonic à des fins de localisation. Si vous n’utilisez ni mnemonic, ni mnemonicKey, aucun mnemonic n’est disponible pour cette commande. [Pour plus d’informations, reportez-vous à la section Touches d’accès et raccourcis clavier sur p. 142.](#)

### **Contrôle Case à cocher**

Définit une case à cocher.

Figure 6-17  
Cases à cocher



#### **Format**

```
<CheckBoxControl controller_attributes invert="true_false" >
  -- options de présentation personnalisée avancées --
</CheckBoxControl>
```

où :

controller\_attributes sont décrits sous [Attributs du contrôleur sur p. 159.](#)

invert est rarement utilisée. Toutefois, si cette option est définie sur true (vrai), inverse l’effet de la sélection et de la désélection de la case à cocher. La valeur par défaut est false (faux).

Les options de présentation personnalisée avancées fournissent un fin degré de contrôle du positionnement et de l'affichage des composants de l'écran. [Pour plus d'informations, reportez-vous à la section Présentation personnalisée avancée sur p. 188.](#)

### Exemple

L'exemple suivant indique le code utilisé pour présenter les cases à cocher indiquées auparavant (les étiquettes de case à cocher sont définies ailleurs dans le fichier de spécifications). L'élément `Layout` est décrit dans [Présentation personnalisée avancée sur p. 188.](#)

```
<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>
```

### Contrôle Groupe de cases à cocher

Définit un ensemble de cases à cocher regroupées et traitées comme unité unique. Ceci peut être utilisée uniquement conjointement avec une propriété de liste énumérée définissant les membres du groupe.

Figure 6-18  
Groupe de cases à cocher



### Format

```
<CheckBoxGroupControl controller_attributes rows="entier" layoutByRow="true_false"
  useSubPanel="true_false" >
  -- options de présentation personnalisée avancées --
</CheckBoxGroupControl>
```

où :

`controller_attributes` sont décrits sous [Attributs du contrôleur sur p. 159.](#)

`rows` est un entier positif indiquant le nombre de lignes affichées à l'écran que le groupe de cases à cocher occupe. La valeur par défaut est 1.

`layoutByRow` indique si les cases à cocher doivent être présentées d'abord le long de la ligne (`true` (vrai)) ou en aval de la colonne (`false` (faux)). La valeur par défaut est `true` (vrai). Pour une utilisation similaire de `layoutByRow` avec un groupe de boutons radio, reportez-vous à [Modification de l'ordre des contrôles sur p. 188](#).

`useSubPanel` indique si oui (`true` (vrai)) ou non (`false` (faux)) les cases à cocher doivent être présentées comme un sous-panneau. La valeur par défaut est `true` (vrai).

Les groupes de cases à cocher sont généralement présentés comme un sous-panneau contenant toutes les zones dans le groupe. Toutefois, ceci peut entraîner des problèmes d'alignement si le groupe de cases à cocher est associé à un champ de texte adjacent. La définition de `useSubPanel` sur `false` (faux) permet de résoudre ce problème.

Les options de présentation personnalisée avancées fournissent un fin degré de contrôle du positionnement et de l'affichage des composants de l'écran. [Pour plus d'informations, reportez-vous à la section Présentation personnalisée avancée sur p. 188](#).

### Exemple

Le code pour créer le groupe de cases à cocher indiqué auparavant est :

```
<CheckBoxGroupControl property="enum3" label="Enum 3" labelKey="enum3.LABEL"/>
```

Les étiquettes et valeurs associées aux cases à cocher individuelles sont définies dans la section Propriétés du noeud pertinent :

```
<Property name="enum3" valueType="enum" isList="true" defaultValue="[value1 value3]">
  <Enumeration>
    <Enum value="value1" label="Valeur 3.1" labelKey="enum3.value1.LABEL"/>
    <Enum value="value2" label="Valeur 3.2" labelKey="enum3.value2.LABEL"/>
    <Enum value="value3" label="Valeur 3.3" labelKey="enum3.value3.LABEL"/>
    <Enum value="value4" label="Valeur 3.4" labelKey="enum3.value4.LABEL"/>
    <Enum value="value5" label="Valeur 3.5" labelKey="enum3.value5.LABEL"/>
  </Enumeration>
</Property>
```

### Contrôle Sélecteur de répertoire client

Définit un champ de texte d'une seule ligne et un bouton associé pour permettre à l'utilisateur de sélectionner un répertoire sur le client. Le répertoire doit déjà exister. L'utilisateur peut ouvrir un fichier depuis ce répertoire ou enregistrer un fichier à l'intérieur, selon le paramétrage du mode.

Figure 6-19  
Contrôle sélecteur de répertoire client



L'utilisateur peut entrer le chemin d'accès au répertoire et le nom directement dans le champ de texte. Il peut aussi cliquer sur le bouton adjacent pour afficher une boîte de dialogue d'où il peut sélectionner un répertoire.

### Format

```
<ClientDirectoryChooserControl controller_attributes mode="chooser_mode" >
  -- options de présentation personnalisée avancées --
</ClientDirectoryChooserControl>
```

où :

*controller\_attributes* sont décrits sous [Attributs du contrôleur sur p. 159](#).

mode détermine le bouton affiché sur la boîte de dialogue d'où l'utilisateur choisit un répertoire et il est l'un des suivants :

- open (par défaut) affiche un bouton Ouvrir.
- save affiche un bouton Enregistrer.

Les options de présentation personnalisée avancées fournissent un fin degré de contrôle du positionnement et de l'affichage des composants de l'écran. [Pour plus d'informations, reportez-vous à la section Présentation personnalisée avancée sur p. 188](#).

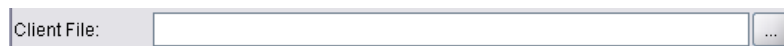
### Exemple

```
<ClientDirectoryChooserControl property="directory2" label="Répertoire Client"
  labelKey="directory2.LABEL"/>
```

## Contrôle Sélecteur de fichier client

Définit un champ de texte d'une seule ligne et un bouton associé pour permettre à l'utilisateur de sélectionner un fichier sur le client. Le fichier doit déjà exister. L'utilisateur peut ouvrir le fichier ou l'enregistrer, selon le paramétrage du mode.

Figure 6-20  
Contrôle sélecteur de fichier client



L'utilisateur peut entrer le chemin d'accès au fichier et le nom directement dans le champ de texte. Il peut aussi cliquer sur le bouton adjacent pour afficher une boîte de dialogue d'où il peut sélectionner un fichier.

### Format

```
<ClientFileChooserControl controller_attributes mode="chooser_mode" >
  -- options de présentation personnalisée avancées --
</ClientFileChooserControl>
```

où :

*controller\_attributes* sont décrits sous [Attributs du contrôleur sur p. 159](#).

mode détermine le bouton affiché sur la boîte de dialogue d'où l'utilisateur choisit un fichier et il est l'un des suivants :

- open (par défaut) affiche un bouton Ouvrir.
- save affiche un bouton Enregistrer.

Les options de présentation personnalisée avancées fournissent un fin degré de contrôle du positionnement et de l'affichage des composants de l'écran. [Pour plus d'informations, reportez-vous à la section Présentation personnalisée avancée sur p. 188.](#)

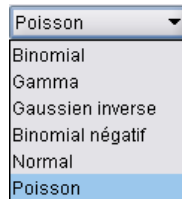
### Exemple

```
<ClientFileChooserControl property="file2" label="Fichier Client" labelKey="file2.LABEL"/>
```

## Contrôle Zone de liste déroulante

Définit une liste déroulante.

Figure 6-21  
zone de liste déroulante



### Format

```
<ComboBoxControl controller_attributes >
  -- options de présentation personnalisée avancées --
</ComboBoxControl>
```

où :

*controller\_attributes* sont décrits sous [Attributs du contrôleur sur p. 159](#).

Les options de présentation personnalisée avancées fournissent un fin degré de contrôle du positionnement et de l'affichage des composants de l'écran. [Pour plus d'informations, reportez-vous à la section Présentation personnalisée avancée sur p. 188.](#)



**Exemple**

L'exemple suivant illustre le code utilisé pour présenter la liste déroulante dans la dernière illustration :

```
<ComboBoxControl property="distribution" >
  <Layout rowIncrement="0" gridWidth="1" fill="none"/>
</ComboBoxControl>
```

L'élément Layout est décrit dans [Présentation personnalisée avancée sur p. 188](#).

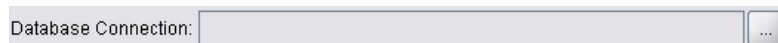
*Remarque* : Les entrées de liste réelles sont définies dans la section Propriétés du noeud pertinent ; dans ce cas, comme liste énumérée dans la déclaration pour la propriété distribution :

```
<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="Gaussien inverse" labelKey="distribution.IGAUSS.LABEL"/>
    <Enum value="NEGBIN" label="Binomial négatif" labelKey="distribution.NEGBIN.LABEL"/>
    <Enum value="NORMAL" label="Normal" labelKey="distribution.NORMAL.LABEL"/>
    <Enum value="POISSON" label="Poisson" labelKey="distribution.POISSON.LABEL"/>
  </Enumeration>
</Property>
```

**Contrôle Sélecteur de connexion à la base de données**

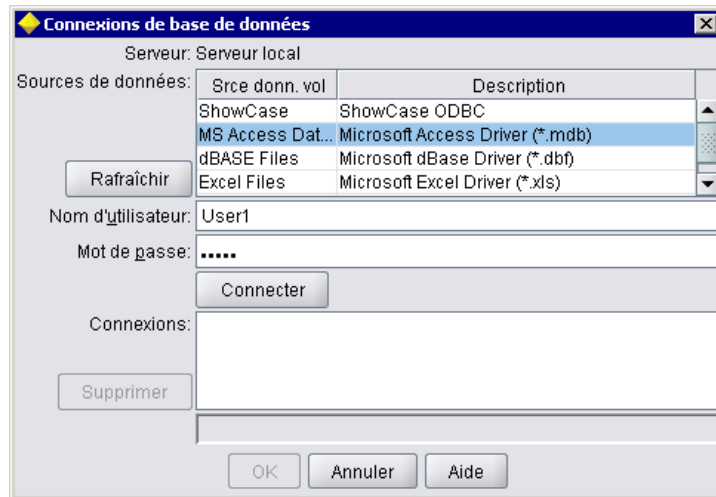
Définit une commande qui permet à l'utilisateur de sélectionner une source de données et de se connecter à une base de données.

Figure 6-22  
Contrôle Sélecteur de connexion à la base de données



L'utilisateur ne peut pas saisir de texte dans le champ de texte. Il doit plutôt cliquer sur le bouton pour afficher la boîte de dialogue Connexions de base de données IBM® SPSS® Modeler standard :

Figure 6-23  
boîte de dialogue Connexions de base de données



Si la connexion est établie, les détails de connexion sont affichés dans le champ de texte du contrôleur sélecteur de connexion à la base de données.

### Format

```
<DBCConnectionChooserControl controller_attributes >
  -- options de présentation personnalisée avancées --
</DBCConnectionChooserControl>
```

où :

*controller\_attributes* sont décrits sous [Attributs du contrôleur sur p. 159](#).

Les options de présentation personnalisée avancées fournissent un fin degré de contrôle du positionnement et de l'affichage des composants de l'écran. [Pour plus d'informations, reportez-vous à la section Présentation personnalisée avancée sur p. 188](#).

### Exemple

L'exemple suivant illustre la manière dont le contrôle requiert la définition d'une propriété de chaîne qu'il peut utiliser pour la chaîne de connexion.

```
<Node ... >
  <Properties>
    ...
    <Property name="dbconnect" valueType="databaseConnection" />
  </Properties>
  ...
```

```

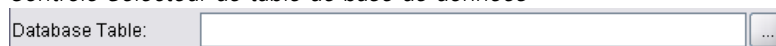
<UserInterface>
...
  <Tabs>
    <Tab label="Base de données">
      <PropertiesPanel>
        <DBConnectionChooserControl property="dbconnect" label="Connection base de données"/>
        ...
      </PropertiesPanel>
    </Tab>
  </Tabs>
</UserInterface>
</Node>

```

### Contrôle Sélecteur de table de base de données

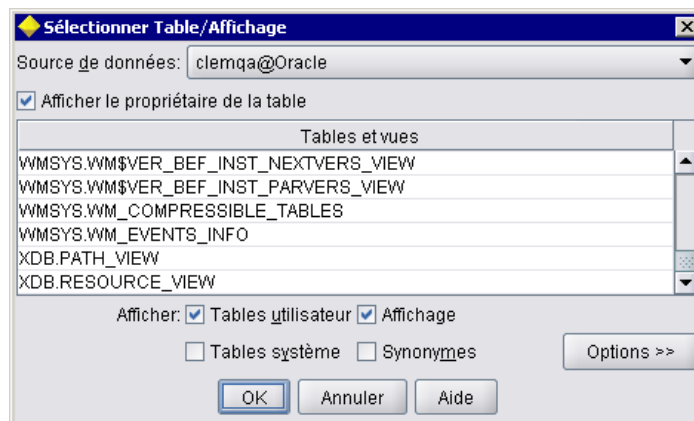
Définit un champ de texte et un bouton associé pour permettre à l'utilisateur de sélectionner une table de base de données après une connexion réussie à la base de données.

Figure 6-24  
Contrôle Sélecteur de table de base de données



L'utilisateur peut entrer le nom de la table directement dans le champ de texte ou cliquer sur le bouton et le sélectionner dans la liste :

Figure 6-25  
Liste de tables de base de données



### Format

```

<DBTableChooserControl connectionProperty="DB_connection_property" controller_attributes >
  -- options de présentation personnalisée avancées --
</DBTableChooserControl>

```

où :

connectionProperty est le nom de la propriété de connexion à la base de données qui a déjà été définie. Ceci est la valeur de l'attribut property d'un élément DBConnectionChooserControl qui a été auparavant défini pour le noeud.

*controller\_attributes* sont décrits sous [Attributs du contrôleur sur p. 159](#).

Les options de présentation personnalisée avancées fournissent un fin degré de contrôle du positionnement et de l'affichage des composants de l'écran. [Pour plus d'informations, reportez-vous à la section Présentation personnalisée avancée sur p. 188](#).

### Exemple

Cet exemple fait suite à celui pour `DBConnectionChooserControl` et indique la manière d'inclure un élément `DBTableChooserControl` pour sélectionner une table de base de données une fois la connexion à la base de données établie.

```
<Node ... >
  <Properties>
    ...
    <Property name="dbconnect" valueType="databaseConnection" />
    <Property name="dbtable" valueType="string" />
  </Properties>
  ...
  <UserInterface>
    ...
    <Tabs>
      <Tab label="Base de données">
        <PropertiesPanel>
          <DBConnectionChooserControl property="dbconnect" label="Connection base de données"/>
          <DBTableChooserControl property="dbtable" connectionProperty="dbconnect"
            label="Table base de données" />
          ...
        </PropertiesPanel>
      </Tab>
    </Tabs>
  </UserInterface>
</Node>
```

### Contrôle Sélecteur multi-champ

Définit une commande permettant à l'utilisateur de choisir un ou plusieurs noms de champs dans une liste.

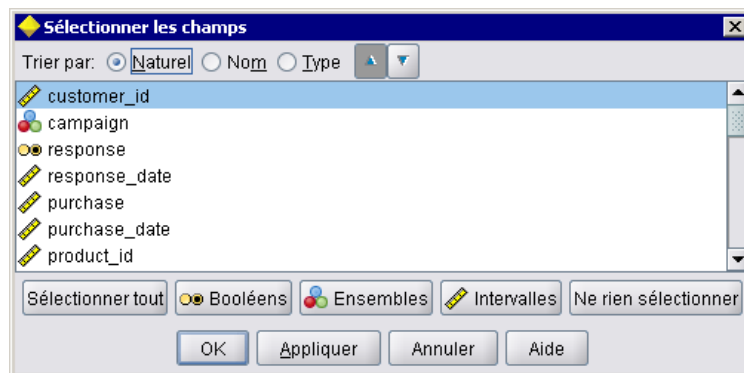
Figure 6-26  
*Contrôle Sélecteur multi-champ*



Lorsque l'utilisateur clique sur cette commande, une liste de champs est affichée d'où l'utilisateur peut en choisir un ou plus.

L'ensemble comprend tous les champs visibles sur ce noeud. Si les champs ont été filtrés plus en amont de ce noeud, seuls les champs qui sont passés dans le filtre sont visibles. Cette liste peut également être restreinte davantage en spécifiant que seuls les champs présentant des types de stockage et de données particuliers doivent être disponibles pour la sélection.

Figure 6-27  
Liste multi-champ



Chaque commande sélecteur multi-champ indique un attribut de propriété, qui est déclaré ailleurs dans le fichier. En outre, il définit le mode d'affichage de la liste sur la boîte de dialogue d'un noeud.

### Format

```
<MultiFieldChooserControl controller_attributes storage="storage_types" onlyNumeric="true_false"
  onlySymbolic="true_false" onlyDatetime="true_false" types="data_types"
  onlyRanges="true_false" onlyDiscrete="true_false" >
  -- options de présentation personnalisée avancées --
</MultiFieldChooserControl>
```

où :

*controller\_attributes* sont décrits sous [Attributs du contrôleur sur p. 159](#).

En outre, vous pouvez davantage limiter la liste des champs en précisant deux autres attributs, dont l'un doit figurer dans la liste suivante :

- **storage** est une propriété de liste spécifiant le type de stockage de champs à attribuer dans la liste ; par exemple, `storage="[integer real]"` signifie que seuls les champs présentant ces types de stockage seront répertoriés. Pour connaître la liste des types de stockage possibles, reportez-vous au tableau sous Données et types de stockage sur p. 226.
- **onlyNumeric**, s'il est défini sur `true` (vrai), spécifie que seuls les champs présentant un type de stockage numérique sont répertoriés.
- **onlySymbolic**, s'il est défini sur `true` (vrai), spécifie que seuls les champs présentant un type de stockage symbolique (c'est-à-dire une chaîne) sont répertoriés.
- **onlyDatetime**, s'il est défini sur `true` (vrai), spécifie que seuls les champs présentant un type de stockage date et heure sont répertoriés

Le second attribut spécifié doit provenir de cette liste :

- `types` est une propriété de liste spécifiant le type de données des champs à attribuer dans la liste ; par exemple, `types="range flag"` signifie que seuls les champs présentant ces types de stockage seront répertoriés. Les types de données possibles sont :
  - intervalle
  - Booléen
  - ensemble
  - orderedSet (ensemble ordonné)
  - numérique
  - discrète
  - typeless (sans type)
- `onlyRanges`, s'il est défini sur `true` (vrai), spécifie que seuls les champs présentant un type de données intervalles sont répertoriés.
- `onlyDiscrete`, s'il est défini sur `true` (vrai), spécifie que seuls les champs présentant un type de données discret (c'est-à-dire booléen, ensemble ou sans type) sont répertoriés.

Par exemple, une commande spécifiant `storage="integer"` et `types="flag"` garantit que seuls les champs d'entiers qui sont des booléens apparaîtront dans la liste.

Les options de présentation personnalisée avancées fournissent un fin degré de contrôle du positionnement et de l'affichage des composants de l'écran. [Pour plus d'informations, reportez-vous à la section Présentation personnalisée avancée sur p. 188.](#)

*Remarque* : Cette commande n'est utilisée que pour les définitions de l'élément Noeud. Pour spécifier un sélecteur multi-champ dans une définition Modèle de données de sortie, utilisez le format suivant :

```
<OutputDataModel mode="mode">
...
  <ForEach var="field" inProperty="prop_name">
    <AddField name="{field_name}_NEW" fieldRef="{field_name}" />
  </ForEach>
...
</OutputDataModel>
```

[Pour plus d'informations, reportez-vous à la section Modèle de données de sortie dans le chapitre 4 sur p. 73.](#) L'élément `ForEach` est décrit dans [Itération avec l'élément ForEach sur p. 83](#). `AddField` est décrit dans [Ajouter un champ sur p. 80](#).

### Exemple

L'exemple suivant illustre le code utilisé pour spécifier la commande sélecteur multi-champ dans la dernière illustration.

```
<MultiFieldChooserControl property="inputs" >
  <Enabled>
```

```

    <Condition control="custom_fields" op="equals" value="true"/>
  </Enabled>
</MultiFieldChooserControl>

```

La section `Enabled` permet au contrôle d'être activé uniquement si le contrôle `custom_fields` est sélectionné.

*Remarque* : Le contenu de cette liste est régi par la déclaration pour la propriété `inputs` dans la section Propriétés pour le noeud pertinent :

```
<Property name="inputs" valueType="string" isList="true" label="Entrées" labelKey="inputs.LABEL"/>
```

### Contrôle Sélecteur multi-élément

Définit une commande permettant à l'utilisateur de choisir un ou plusieurs éléments dans une liste de valeurs. Il associe une propriété à un catalogue qui contient une liste de valeurs. [Pour plus d'informations, reportez-vous à la section Catalogs dans le chapitre 4 sur p. 51.](#)

Figure 6-28  
Contrôle Sélecteur multi-élément



#### Format

```

<MultiItemChooserControl controller_attributes catalog="catalog_name" >
  -- options de présentation personnalisée avancées --
</MultiItemChooserControl>

```

où :

`controller_attributes` sont décrits sous [Attributs du contrôleur sur p. 159.](#)

`catalog` (obligatoire) est le nom du catalogue à associer. La bibliothèque à partir de laquelle le catalogue est obtenu est celle spécifiée dans l'élément `Module` de la section `Execution`. [Pour plus d'informations, reportez-vous à la section Modules dans le chapitre 4 sur p. 71.](#)

Les options de présentation personnalisée avancées fournissent un fin degré de contrôle du positionnement et de l'affichage des composants de l'écran. [Pour plus d'informations, reportez-vous à la section Présentation personnalisée avancée sur p. 188.](#)

#### Exemple

```
<MultiItemChooserControl property="selection2" catalog="cat2" />
```

La propriété référencée par l'attribut `property` (`selection2` dans ce cas) doit être une propriété ayant un attribut `isList="true"`. Pour une explication et des exemples de l'utilisation de `MultiItemChooserControl`, reportez-vous à [Catalogs sur p. 51.](#)

### **Contrôle Zone de mot de passe**

Définit un champ de texte d'une seule ligne où les caractères d'entrée sont masqués lors de leur saisie.

Figure 6-29

*Contrôle Zone de mot de passe*



#### **Format**

```
<PasswordBoxControl controller_attributes columns="entier" >  
  -- options de présentation personnalisée avancées --  
</PasswordBoxControl>
```

où :

*controller\_attributes* sont décrits sous [Attributs du contrôleur sur p. 159](#).

*columns* est un entier positif qui définit le nombre de colonnes de caractères que la zone de mot de passe doit occuper. La valeur par défaut est 20.

Les options de présentation personnalisée avancées fournissent un fin degré de contrôle du positionnement et de l'affichage des composants de l'écran. [Pour plus d'informations, reportez-vous à la section Présentation personnalisée avancée sur p. 188](#).

#### **Exemple**

```
<PasswordBoxControl property="encrypted_string1" label="Chaîne codée 1" labelKey="encryptedString1.LABEL"/>
```

Le champ de texte est codé par l'association à une propriété définie comme une chaîne cryptée dans la section Properties pour le noeud pertinent :

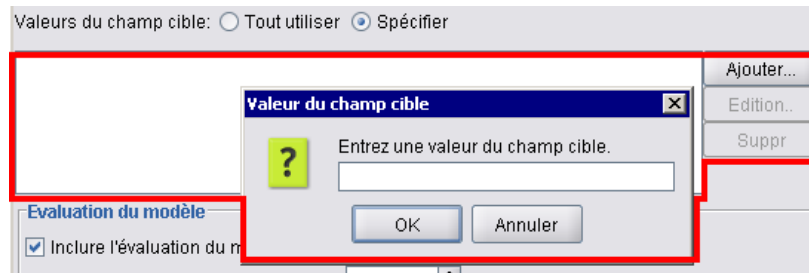
```
<Property name="encrypted_string1" valueType="encryptedString"/>
```



## Contrôle Propriété

Un contrôle de propriété est entièrement personnalisable et permet à l'utilisateur de saisir des propriétés pour le noeud. Le traitement est géré par une classe Java écrite par un utilisateur. Vous trouverez ci-après un exemple de contrôle de propriété :

Figure 6-30  
Sélection de boîte de dialogue avec exemple de contrôle de propriété sélectionné



### Format

```
<PropertyControl controller_attributes controlClass="Java_class" >
  -- options de présentation personnalisée avancées --
</PropertyControl>
```

où :

*controller\_attributes* sont décrits sous [Attributs du contrôleur sur p. 159](#).

*controlClass* (obligatoire) est le chemin d'accès dans un fichier *jar* à la classe Java qui met en œuvre le contrôle propriété. (*Remarque* : Le fichier *jar* est déclaré dans un élément *JarFile* dans la section Ressources. [Pour plus d'informations, reportez-vous à la section Fichiers Jar dans le chapitre 4 sur p. 43.](#))

Les options de présentation personnalisée avancées fournissent un fin degré de contrôle du positionnement et de l'affichage des composants de l'écran. [Pour plus d'informations, reportez-vous à la section Présentation personnalisée avancée sur p. 188.](#)

### Exemple

```
<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>
```

Le contrôle de propriété est associé à une propriété définie dans la section Propriétés pour le noeud pertinent :

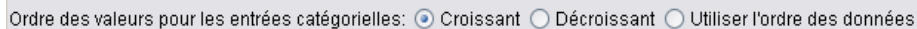
```
<Property name="target_field_values_specify" valueType="string" isList="true" label=""
  labelKey="target_field_values_specify.LABEL"/>
```

### Contrôle Groupe de boutons radio

Définit un ensemble de boutons radio où un seul bouton peut être sélectionné à la fois.

Figure 6-31

Contrôle Groupe de boutons radio



Ordre des valeurs pour les entrées catégorielles:  Croissant  Décroissant  Utiliser l'ordre des données

Chaque commande groupe de boutons radio possède un attribut de propriété qui associe le groupe à une propriété spécifique. Cette propriété est définie ailleurs dans le fichier et spécifie les boutons qui composent le groupe.

La propriété associée peut être une liste énumérée ou une propriété Booléenne. Pour les listes énumérées (où l'attribut propriété `valueType="enum"`), un bouton radio est affiché pour chaque valeur `enum`. Pour les propriétés Booléennes (où `valueType="boolean"`), deux boutons radio sont affichés en permanence.

#### Format

```
<RadioButtonGroupControl controller_attributes
  rows="entier" layoutByRow="true_false" useSubPanel="true_false"
  falseLabel="button_label" falseLabelKey="label_key" trueLabel="?button_label"
  trueLabelKey="label_key" trueFirst="true_false" >
  -- options de présentation personnalisée avancées --
</RadioButtonGroupControl>
```

où :

`controller_attributes` sont décrits sous [Attributs du contrôleur sur p. 159](#).

`rows` est un entier positif qui indique le nombre de lignes d'écran sur lesquelles le groupe est affiché. La valeur par défaut est 1.

`layoutByRow` indique si les boutons radio doivent être présentés d'abord le long de la ligne (`true` (vrai)) ou en aval de la colonne (`false` (faux)). La valeur par défaut est `true` (vrai). Pour obtenir un exemple d'utilisation de `layoutByRow` avec un groupe de boutons radio, reportez-vous à [Modification de l'ordre des contrôles sur p. 188](#).

`useSubPanel` indique si oui (`true` (vrai)) ou non (`false` (faux)) les boutons radio doivent être présentés comme un sous-panneau. La valeur par défaut est `true` (vrai).

Les groupes de boutons radio sont généralement présentés comme un sous-panneau contenant tous les boutons dans le groupe. Toutefois, ceci peut entraîner des problèmes d'alignement si le groupe de boutons radio est associé à un champ de texte adjacent. La définition de `useSubPanel` sur `false` (faux) permet de résoudre ce problème.

`falseLabel` est l'étiquette pour la valeur «false (faux)» d'une propriété Booléenne (voir le deuxième exemple ci-dessous). Uniquement utilisé avec les propriétés Booléennes, dans quel cas il est obligatoire.

`falseLabelKey` identifie l'étiquette «false (faux)» à des fins de localisation.

`trueLabel` est l'étiquette pour la valeur «true (vrai)» d'une propriété Booléenne (voir le deuxième exemple ci-dessous). Uniquement utilisé avec les propriétés Booléennes, dans quel cas il est obligatoire.

`trueLabelKey` identifie l'étiquette «true (vrai)» à des fins de localisation.

`trueFirst`, if set to `true (vrai)`, permet l'inversion de l'ordre d'affichage des boutons pour une propriété Booléenne, de sorte que le bouton représentant la valeur «true (vrai)» est d'abord affiché. La valeur par défaut est `false (faux)`, indiquant que le bouton représentant la valeur «false (faux)» est d'abord affiché.

Les options de présentation personnalisée avancées fournissent un fin degré de contrôle du positionnement et de l'affichage des composants de l'écran. [Pour plus d'informations, reportez-vous à la section Présentation personnalisée avancée sur p. 188.](#)

### Exemples

Le premier exemple illustre le code utilisé pour le groupe de boutons radio indiqué auparavant.

```
<RadioButtonGroupControl property="value_order" labelWidth="2">
  <Layout gridWidth="4"/>
</RadioButtonGroupControl>
```

L'élément `Layout` est décrit dans [Présentation personnalisée avancée sur p. 188.](#)

*Remarque* : Le nombre de boutons et leurs étiquettes sont définies dans la section Propriétés du noeud pertinent ; dans ce cas, comme liste énumérée dans la déclaration pour la propriété `value_order` : Cette déclaration inclut également l'étiquette pour le groupe lui-même :

```
<Property name="value_order" valueType="enum" label="Ordre des valeurs pour entrées
catégorielles" labelKey="value_order.LABEL">
  <Enumeration>
    <Enum value="Ascendant" label="Ascendant" labelKey="value_order.Ascending.LABEL"/>
    <Enum value="Descendant" label="Descendant" labelKey="value_order.Descending.LABEL"/>
    <Enum value="OrdreDonnées" label="Utiliser Ordre Données" labelKey="value_order.UseDataOrder.LABEL"/>
  </Enumeration>
</Property>
```

Le deuxième exemple illustre l'utilisation de `falseLabel` et `trueLabel` pour un bouton radio qui contrôle une propriété Booléenne telle qu'une propriété permettant de contrôler l'activation des paramètres standard ou personnalisés :

Figure 6-32

Groupe de boutons radio contrôlant une propriété Booléenne

Boolean 5:  Standard  Custom

Le code à obtenir est :

```
<RadioButtonGroupControl property="boolean5" label="Booléen 5" labelKey="boolean5.LABEL"
  falseLabel="Standard" falseLabelKey="boolean5.false.LABEL" trueLabel="Personnalisé"
  trueLabelKey="boolean5.true.LABEL" />
```

Dans ce cas, les étiquettes de boutons et les étiquettes de groupes sont définies dans l'élément `RadioButtonGroupControl` lui-même. La propriété avec laquelle le groupe est associé est définie dans la section Propriétés du noeud :

```
<Property name="boolean5" valueType="boolean" defaultValue="false"/>
```

### Contrôle Sélecteur de répertoire serveur

Définit un champ de texte d'une seule ligne et un bouton associé pour permettre à l'utilisateur de sélectionner un répertoire sur le serveur. Le répertoire doit déjà exister. L'utilisateur peut ouvrir un fichier depuis ce répertoire ou enregistrer un fichier à l'intérieur, selon le paramétrage du mode.

Figure 6-33  
Contrôle sélecteur de répertoire serveur



L'utilisateur peut entrer le chemin d'accès au répertoire et le nom directement dans le champ de texte. Il peut aussi cliquer sur le bouton adjacent pour afficher une boîte de dialogue d'où il peut sélectionner un répertoire.

#### Format

```
<ServerDirectoryChooserControl controller_attributes mode="chooser_mode" >
  -- options de présentation personnalisée avancées --
</ServerDirectoryChooserControl>
```

où :

`controller_attributes` sont décrits sous [Attributs du contrôleur sur p. 159](#).

`mode` détermine le bouton affiché sur la boîte de dialogue d'où l'utilisateur choisit un répertoire et il est l'un des suivants :

- `open` (par défaut) affiche un bouton Ouvrir.
- `save` affiche un bouton Enregistrer.

Les options de présentation personnalisée avancées fournissent un fin degré de contrôle du positionnement et de l'affichage des composants de l'écran. [Pour plus d'informations, reportez-vous à la section Présentation personnalisée avancée sur p. 188.](#)

#### Exemple

```
<ServerDirectoryChooserControl property="directory1" label="Répertoire Serveur"
  labelKey="directory1.LABEL"/>
```

### Contrôle Sélecteur de fichier serveur

Définit un champ de texte d'une seule ligne et un bouton associé pour permettre à l'utilisateur de sélectionner un fichier sur le serveur. Le fichier doit déjà exister. L'utilisateur peut ouvrir le fichier ou l'enregistrer, selon le paramétrage du mode.

Figure 6-34

Contrôle sélecteur de fichier serveur



L'utilisateur peut entrer le chemin d'accès au fichier et le nom directement dans le champ de texte. Il peut aussi cliquer sur le bouton adjacent pour afficher une boîte de dialogue d'où il peut sélectionner un fichier.

#### Format

```
<ServerFileChooserControl controller_attributes mode="chooser_mode" >
  -- options de présentation personnalisée avancées --
</ServerFileChooserControl>
```

où :

*controller\_attributes* sont décrits sous [Attributs du contrôleur sur p. 159](#).

mode détermine le bouton affiché sur la boîte de dialogue d'où l'utilisateur choisit un fichier et il est l'un des suivants :

- open (par défaut) affiche un bouton Ouvrir.
- save affiche un bouton Enregistrer.

Les options de présentation personnalisée avancées fournissent un fin degré de contrôle du positionnement et de l'affichage des composants de l'écran. [Pour plus d'informations, reportez-vous à la section Présentation personnalisée avancée sur p. 188.](#)

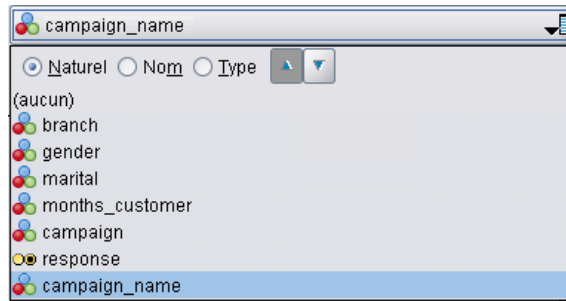
#### Exemple

```
<ServerFileChooserControl property="file1" label="Fichier serveur" labelKey="file1.LABEL"/>
```

### Contrôle Sélecteur d'un champ unique

Définit une commande permettant à l'utilisateur de choisir un seul champ dans une liste.

Figure 6-35  
Contrôle Sélecteur d'un champ unique



Lorsque l'utilisateur clique sur cette commande, une liste de champs est affichée d'où un seul champ peut être choisi.

L'ensemble comprend tous les champs visibles sur ce noeud. Si les champs ont été filtrés plus en amont de ce noeud, seuls les champs qui sont passés dans le filtre sont visibles. Cette liste peut également être restreinte davantage en spécifiant que seuls les champs présentant des types de stockage et de données particuliers doivent être disponibles pour la sélection.

### Format

```
<SingleFieldChooserControl controller_attributes storage="storage_types" onlyNumeric="true_false"
  onlySymbolic="true_false" onlyDatetime="true_false" types="data_types"
  onlyRanges="true_false" onlyDiscrete="true_false" >
  -- options de présentation personnalisée avancées --
</SingleFieldChooserControl>
```

où :

*controller\_attributes* sont décrits sous [Attributs du contrôleur sur p. 159](#).

En outre, vous pouvez davantage limiter la liste des champs en précisant deux autres attributs, dont l'un doit figurer dans la liste suivante :

- **storage** est une propriété de liste spécifiant le type de stockage de champs à attribuer dans la liste ; par exemple, `storage="[integer real]"` signifie que seuls les champs présentant ces types de stockage seront répertoriés. Pour connaître la liste des types de stockage possibles, reportez-vous au tableau sous Données et types de stockage sur p. 226.
- **onlyNumeric**, s'il est défini sur `true` (vrai), spécifie que seuls les champs présentant un type de stockage numérique sont répertoriés.
- **onlySymbolic**, s'il est défini sur `true` (vrai), spécifie que seuls les champs présentant un type de stockage symbolique (c'est-à-dire une chaîne) sont répertoriés.
- **onlyDatetime**, s'il est défini sur `true` (vrai), spécifie que seuls les champs présentant un type de stockage date et heure sont répertoriés.

Le second attribut spécifié doit provenir de cette liste :

- `types` est une propriété de liste spécifiant le type de données des champs à attribuer dans la liste ; par exemple, `types="[range flag]"` signifie que seuls les champs présentant ces types de stockage seront répertoriés. Les types de données possibles sont :
  - intervalle
  - Booléen
  - ensemble
  - orderedSet (ensemble ordonné)
  - numérique
  - discrète
  - typeless (sans type)
- `onlyRanges`, s'il est défini sur `true` (vrai), spécifie que seuls les champs présentant un type de données intervalles sont répertoriés.
- `onlyDiscrete`, s'il est défini sur `true` (vrai), spécifie que seuls les champs présentant un type de données discret (c'est-à-dire booléen, ensemble ou sans type) sont répertoriés.

Par exemple, une commande spécifiant `storage="[integer]"` et `types="[flag]"` garantit que seuls les champs d'entiers qui sont des booléens apparaîtront dans la liste.

Les options de présentation personnalisée avancées fournissent un fin degré de contrôle du positionnement et de l'affichage des composants de l'écran. [Pour plus d'informations, reportez-vous à la section Présentation personnalisée avancée sur p. 188.](#)

*Remarque* : Cette commande n'est utilisé que pour les définitions de noeud. Pour spécifier un sélecteur multi-champ dans une définition Modèle de données de sortie, utilisez le format suivant :

```
<OutputDataModel mode="mode">
...
  <ForEach var="field" from="1" to="{entier}">
    <AddField name="{chaîne}_{field}" fieldRef="{field_ref}" />
  </ForEach>
...
</OutputDataModel>
```

[Pour plus d'informations, reportez-vous à la section Modèle de données de sortie dans le chapitre 4 sur p. 73.](#) L'élément `ForEach` est décrit dans [Itération avec l'élément ForEach sur p. 83.](#) `AddField` est décrit dans [Ajouter un champ sur p. 80.](#)

### Exemple

L'exemple suivant illustre le code utilisé pour spécifier la commande sélecteur d'un champ unique dans la dernière illustration.

```
<SingleFieldChooserControl property="target" storage="string" onlyDiscrete="true"/>
```

*Remarque* : Le contenu réel de la liste est défini dans la section Properties du noeud pertinent ; dans ce cas, dans la déclaration pour la propriété `target` :

```
<Property name="target" valueType="string" label="Champ cible" labelKey="target.LABEL"/>
```

### Contrôle Sélecteur élément unique

Définit une commande permettant à l'utilisateur de choisir un seul élément dans une liste de valeurs. Il associe une propriété à un catalogue qui contient une liste de valeurs. [Pour plus d'informations, reportez-vous à la section Catalogs dans le chapitre 4 sur p. 51.](#)

Figure 6-36  
Contrôle Sélecteur élément unique



#### Format

```
<SingleItemChooserControl controller_attributes catalog="catalog_name" >
  -- options de présentation personnalisée avancées --
</SingleItemChooserControl
```

où :

`controller_attributes` sont décrits sous [Attributs du contrôleur sur p. 159.](#)

`catalog` (obligatoire) est le nom du catalogue à associer. La bibliothèque à partir de laquelle le catalogue est obtenu est celle spécifiée dans l'élément `Module` de la section Execution. [Pour plus d'informations, reportez-vous à la section Modules dans le chapitre 4 sur p. 71.](#)

Les options de présentation personnalisée avancées fournissent un fin degré de contrôle du positionnement et de l'affichage des composants de l'écran. [Pour plus d'informations, reportez-vous à la section Présentation personnalisée avancée sur p. 188.](#)

#### Exemple

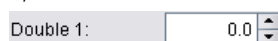
```
<SingleItemChooserControl property="selection1" catalog="cat1" />
```

Pour une explication et des exemples de l'utilisation de cette commande, reportez-vous à [Catalogs sur p. 51.](#)

### Contrôle Spinner

Définit un spinner (champ numérique avec flèches Haut et Bas pour modifier la valeur de champ).

Figure 6-37  
Spinner





### **Format**

```
<SpinnerControl controller_attributes columns="entier" stepSize="increment"  
  minDecimalDigits="number" maxDecimalDigits="number" >  
  -- options de présentation personnalisée avancées --  
</SpinnerControl>
```

où :

*controller\_attributes* sont décrits sous [Attributs du contrôleur sur p. 159](#).

*columns* est un entier positif indiquant le nombre de colonnes de caractères que la commande couvre. La valeur par défaut est 5.

*stepSize* est un nombre décimal qui spécifie le nombre par lequel la valeur de champ change lorsque l'utilisateur clique sur l'une des flèches. La valeur par défaut est 1,0.

*minDecimalDigits* est le nombre minimum de décimales à afficher pour la valeur de champ. La valeur par défaut est 1.

*maxDecimalDigits* est le nombre maximum de décimales à afficher pour la valeur de champ.

Les options de présentation personnalisée avancées fournissent un fin degré de contrôle du positionnement et de l'affichage des composants de l'écran. [Pour plus d'informations, reportez-vous à la section Présentation personnalisée avancée sur p. 188](#).

### **Exemple**

L'exemple suivant illustre le code utilisé pour spécifier le contrôle spinner dans la dernière illustration :

```
<SpinnerControl property="double1" label="Double 1" labelKey="double1.LABEL"/>
```

La précision et l'intervalle valide du contenu de champ numérique sont définis dans la section Propriétés pour le noeud pertinent ; dans ce cas, dans la déclaration pour la propriété `double1` :

```
<Property name="double1" valueType="double" min="0" max="100"/>
```

### **Contrôle Table**

Définit un élément de présentation de table à afficher dans la boîte de dialogue de noeud ou dans la fenêtre de sortie.

Figure 6-38  
Contrôle Table

Structured:	Name	Yes/No	Count	Limit
		<input checked="" type="checkbox"/>	0	0.0

### Format

```
<TableControl controller_attributes rows="entier" columns="entier" columnWidths="Liste" >
  -- options de présentation personnalisée avancées --
</TableControl>
```

où :

*controller\_attributes* sont décrits sous [Attributs du contrôleur sur p. 159](#).

*rows* est un entier positif indiquant le nombre de lignes de table visibles à l'écran. La valeur par défaut est 8.

*columns* est un entier positif indiquant le nombre de colonnes de caractères que la table couvre. La valeur par défaut est 20.

*columnwidths* est une liste de valeurs qui indique les largeurs relatives de colonnes. Par exemple, une valeur de [30 5 10] indique que la colonne 1 est trois fois plus large que la colonne 3.

Les options de présentation personnalisée avancées fournissent un fin degré de contrôle du positionnement et de l'affichage des composants de l'écran. [Pour plus d'informations, reportez-vous à la section Présentation personnalisée avancée sur p. 188](#).

### Exemple

Le code pour spécifier la commande table dans la dernière illustration est :

```
<TableControl property="structure1" allowReorder="true" label="Structuré"
  labelKey="structure1.LABEL" columnWidths="[20 6 10 10]">
  <ColumnControl column="0" editor="fieldValue" fieldControl="field1"/>
</TableControl>
```

La structure de la commande table est définie comme type de propriété dans la section Objets communs du fichier de spécifications :

```
<PropertyType id="shared_structure1" valueType="structure" isList="true">
  <Structure >
    <Attribute name="id" valueType="string" label="Nom" labelKey="structure1.id.LABEL"/>
    <Attribute name="yesno" valueType="boolean" label="Oui/Non" labelKey="structure1.yesno.LABEL"
      defaultValue="true"/>
```

```

    <Attribute name="count" valueType="integer" label="Nombre" labelKey="structure1.count.LABEL"
      defaultValue="0"/>
    <Attribute name="limit" valueType="double" label="Limite" labelKey="structure1.limit.LABEL"
      defaultValue="0.0"/>
  </Structure>
</PropertyType>

```

Dans la spécification de noeud, l'identificateur de ce type de propriété est ensuite associé à celui du contrôle table au moyen d'une déclaration de propriété :

```
<Property name="structure1" type="shared_structure1"/>
```

En cas de référence au noeud dans un script, vous pouvez définir les valeurs dans la propriété à l'aide de crochets [] pour la liste et des accolades {} pour la structure. Par exemple, vous pouvez définir une grille de deux structures pour la propriété structure1 comme suit :

```
set:node_ID.structure1 = [{"hello" true 4 0.21} {"bye" false 5 0.95}]
```

Notez que l'ordre des valeurs doit être cohérent avec celui dans lequel les définitions d'Attribut sont effectuées.

### Contrôle Zone de texte

Définit un champ d'entrée de texte multi-ligne

Figure 6-39  
Contrôle Zone de texte



#### Format

```

<TextAreaControl controller_attributes rows="entier" columns="entier" wrapLines="true_false" >
  -- options de présentation personnalisée avancées --
</TextAreaControl>

```

où :

*controller\_attributes* sont décrits sous [Attributs du contrôleur sur p. 159](#).

*rows* est un entier positif indiquant le nombre de lignes d'écran que la zone de texte couvre. La valeur par défaut est 8.

*columns* est un entier positif indiquant le nombre de colonnes de caractères que la zone de texte couvre. La valeur par défaut est 20.

*wrapLines* indique s'il faut utiliser ou non le retour automatique à la ligne pour les longues lignes de texte (*true* (vrai)) ou requérir le défilement horizontal pour lire des lignes de texte longues (*false* (faux)). La valeur par défaut est *true* (vrai).

Les options de présentation personnalisée avancées fournissent un fin degré de contrôle du positionnement et de l'affichage des composants de l'écran. [Pour plus d'informations, reportez-vous à la section Présentation personnalisée avancée sur p. 188.](#)

### Exemple

Le code pour créer l'exemple indiqué auparavant est :

```
<TextAreaControl property="string2" label="Chaîne 2" labelKey="string2.LABEL"/>
```

Dans ce cas, l'étiquette pour la zone de texte est définie dans la déclaration de commande de la zone de texte, alors que le type de donnée d'entrée est défini dans la section Propriétés du noeud concerné, dans la déclaration de la propriété `string2` :

```
<Property name="string2" valueType="string" />
```

### Contrôle Zone de texte

Définit un champ d'entrée de texte à une seule ligne.

Figure 6-40  
Contrôle Zone de texte



### Format

```
<TextBoxControl controller_attributes columns="entier" >
  -- options de présentation personnalisée avancées --
</TextBoxControl>
```

où :

`controller_attributes` sont décrits sous [Attributs du contrôleur sur p. 159.](#)

`columns` est un entier positif indiquant le nombre de colonnes de caractères que la zone de texte couvre. La valeur par défaut est 20.

Les options de présentation personnalisée avancées fournissent un fin degré de contrôle du positionnement et de l'affichage des composants de l'écran. [Pour plus d'informations, reportez-vous à la section Présentation personnalisée avancée sur p. 188.](#)

### Exemple

Le code pour créer la zone de texte indiquée auparavant est :

```
<TextBoxControl property="string1" label="Chaîne 1" labelKey="string1.LABEL"/>
```

Le type de données d'entrée pour la zone de texte est défini dans la section Propriétés pour le noeud pertinent ; dans ce cas, dans la déclaration pour la propriété `string1` :

```
<Property name="string1" valueType="string" />
```

## Présentations du Contrôle Propriété

Cette section décrit les méthodes de présentation standard utilisées pour les boîtes de dialogue et les fenêtres, ainsi que les moyens de les modifier pour obtenir vos propres présentations personnalisées.

### Présentation du contrôle standard

Un panneau de propriétés peut être considéré comme une grille de cellules en 2D. Chaque ligne peut comporter une hauteur différente, et chaque colonne une largeur différente. Les composants de l'interface utilisateur peuvent être affectés à plusieurs cellules contiguës. Toutefois, un composant IU est généralement alloué à une seule cellule.

Par défaut, un contrôle propriété est affecté à une seule ligne, et chaque contrôle occupe deux colonnes : une pour l'étiquette et l'autre pour le composant(s) du contrôle. La colonne contenant les étiquettes est développée jusqu'à la largeur de l'étiquette la plus grande. Par exemple, selon les éléments suivants dans le fichier de spécifications :

```
<TextBoxControl property="string1" label="Chaîne 1"/>  
<PasswordBoxControl property="encryptedString1" label="Chaîne codée 1"/>  
<TextAreaControl property="string2" label="Chaîne 2"/>
```

le panneau obtenu ressemble à ceci :

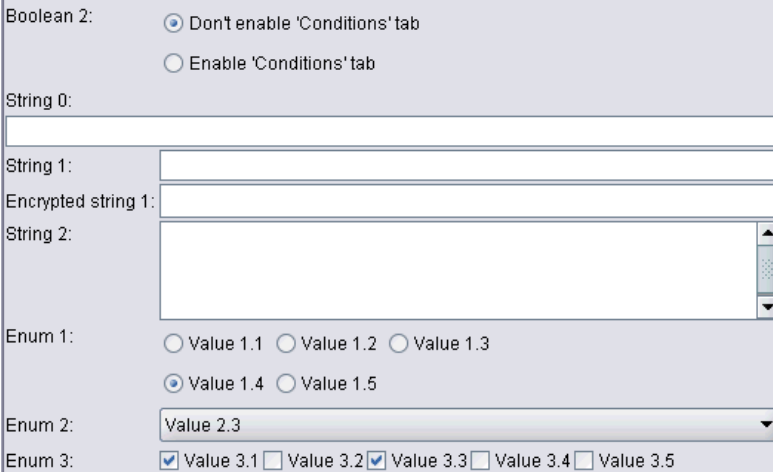
Figure 6-41  
Panneau de propriétés simple

String 1:	<input type="text"/>
Encrypted string 1:	<input type="password"/>
String 2:	<input type="text"/>

Notez que le caractère “:” à la fin de l'étiquette est ajouté automatiquement.

Une contrôle de propriété qui contient plusieurs composants d'interface utilisateur crée sa propre zone rectangulaire invisible dans laquelle disposer ces composants. Les éléments `RadioButtonGroupControl` et `CheckBoxGroupControl` sont des exemples de ces commandes. Dans l'écran suivant, les commandes étiquetées `Boolean2`, `Enum1` et `Enum3` illustrent ceci :

**Figure 6-42**  
*Panneau des propriétés avec différents contrôles à plusieurs composants*



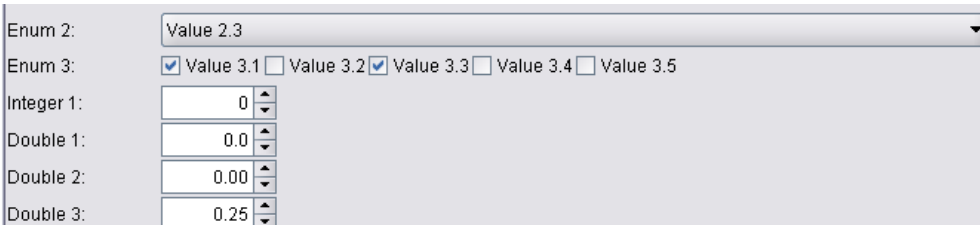
The screenshot shows a property panel with the following controls:

- Boolean 2:** Two radio buttons: "Don't enable 'Conditions' tab" (selected) and "Enable 'Conditions' tab".
- String 0:** A single-line text box.
- String 1:** A single-line text box.
- Encrypted string 1:** A single-line text box.
- String 2:** A multi-line text area with a vertical scrollbar.
- Enum 1:** Five radio buttons: "Value 1.1", "Value 1.2", "Value 1.3", "Value 1.4" (selected), and "Value 1.5".
- Enum 2:** A dropdown menu showing "Value 2.3".
- Enum 3:** Five checkboxes: "Value 3.1" (checked), "Value 3.2", "Value 3.3" (checked), "Value 3.4", and "Value 3.5".

Veillez noter que la forme de la zone rectangulaire dans laquelle les composants sont disposés diffère en fonction du contrôle de propriété. Par conséquent, la présentation des différentes commandes n'est pas toujours parfaitement alignée. Comparez `Enum 1` et `Enum 3` dans l'illustration précédente.

Certains contrôles de propriété incluent des composants qui remplissent complètement la colonne des composants et qui sont redimensionnés lorsque la largeur de la fenêtre est agrandie ou réduite. Des exemples de ces commandes sont celles spécifiées par les éléments `TextBoxControl`, `PasswordBoxControl` et `TextAreaControl`, comme le montrent les commandes étiquetées `String 1`, `Encrypted string 1` et `String 2` dans les illustrations précédentes. Cependant, seuls quelques composants effectuent ceci. Par exemple, les contrôles cases à cocher et spinner n'occupent qu'une quantité fixe d'espace horizontal, même lorsque la largeur de la fenêtre est agrandie :

**Figure 6-43**  
*Panneau des propriétés avec commandes cases à cocher et spinner*



The screenshot shows a property panel with the following controls:

- Enum 2:** A dropdown menu showing "Value 2.3".
- Enum 3:** Five checkboxes: "Value 3.1" (checked), "Value 3.2", "Value 3.3" (checked), "Value 3.4", and "Value 3.5".
- Integer 1:** A spinner control showing the value "0".
- Double 1:** A spinner control showing the value "0.0".
- Double 2:** A spinner control showing the value "0.00".
- Double 3:** A spinner control showing the value "0.25".

## Présentation de contrôle personnalisée

La présentation standard des contrôles peut être modifiée de plusieurs façons, certaines sont simples et d'autres complexes.

### Présentation personnalisée simple

Les trois méthodes de personnalisation de la présentation des contrôles sont :

- positionner une étiquette sur son composant ;
- modifier le nombre de lignes sur lequel les contrôles sont disposés ;
- Modifier l'ordre de présentation des contrôles

### Positionnement d'une étiquette sur son composant

Vous pouvez positionner une étiquette dans une ligne distincte au-dessus de son composant en définissant l'attribut `labelAbove` du contrôle sur `true` (vrai). Par exemple :

```
<TextBoxControl property="string0" label="Chaîne 0" labelAbove="true"/>
<TextBoxControl property="string1" label="Chaîne 1"/>
<PasswordBoxControl property="encryptedString1" label="Chaîne codée 1"/>
```

Outre le positionnement de l'étiquette au-dessus du composant, les composant(s) IU sont affectés à la colonne d'étiquette de l'affichage. Ceci entraîne le panneau suivant, avec l'étiquette Chaîne 0 affichée au-dessus de son champ correspondant :

Figure 6-44

Panneau avec étiquette de champ dans une ligne distincte

### Modification du nombre de lignes

Par défaut, les groupes de cases à cocher ou de cases d'option sont disposés sur une seule ligne et la largeur de la boîte de dialogue est ajustée pour les accommoder. Si un groupe de cases d'option ou de cases à cocher a un grand nombre d'options, la boîte de dialogue peut alors être très large. Ceci peut être évité en modifiant le nombre de lignes utilisées pour afficher la commande. Pour ce faire, définissez l'attribut `rows` de la définition de la commande sur la valeur désirée. Par exemple :

```
<RadioButtonGroupControl property="enum1" label="Enum 1" rows="2"/>
```

Ceci entraîne un panneau où le groupe de boutons radio est disposé sur deux lignes :

Figure 6-45

Panneau avec groupe de boutons radio disposés sur deux lignes

### Modification de l'ordre des contrôles

Pour les groupes de boutons radio et de cases à cocher, vous pouvez aussi modifier l'ordre dans lequel les commandes de chaque valeur enum sont ajoutés au panneau.

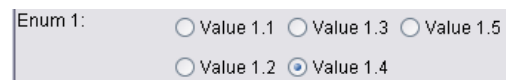
Par défaut, les commandes sont ajoutées dans l'ordre des lignes, comme dans l'exemple précédent, où la première, deuxième et troisième valeur sont ajoutées à la première ligne, avec la quatrième et cinquième valeur ajoutées à la deuxième ligne. A la place, vous pouvez ajouter les contrôles dans l'ordre des colonnes dans le nombre indiqué de lignes en définissant `layoutByRow` sur `false` (faux). Par exemple :

```
<RadioButtonGroupControl property="enum1" label="Enum 1" rows="2" layoutByRow="false"/>
```

Les valeurs restent affichées sur deux lignes. Toutefois, la première et la deuxième valeur sont ajoutées à la première colonne. Tandis que la troisième et la quatrième valeur à la deuxième colonne, et la cinquième valeur à la troisième colonne :

Figure 6-46

Panneau avec groupe de boutons radio disposés dans l'ordre des colonnes



Pour les propriétés Booléennes affichées comme deux boutons radio, le comportement de l'ordre par défaut doit afficher le bouton «False (Faux)» avant le bouton «True (Vrai)». Vous pouvez inverser cet ordre en définissant l'attribut `trueFirst` sur `true` (vrai).

Vous pouvez également empêcher les groupes de boutons radio et de cases à cocher d'utiliser un sous-panneau en définissant l'attribut `useSubPanel` sur `false` (faux). Toutefois, ceci peut conduire à un comportement de présentation non souhaitable sauf en cas d'utilisation conjointe avec l'élément `Layout` (reportez-vous à [Définition de positions de contrôle précises avec l'élément Layout sur p. 188](#)).

### Présentation personnalisée avancée

Dans chaque déclaration de commande, vous pouvez préciser des présentations de commandes complexes à l'aide d'éléments différents. Vous pouvez :

- spécifier des positions de commandes précises sur l'écran avec l'élément `Layout` ;
- contrôler des caractéristiques d'affichage avec l'élément `Enabled` ;
- contrôler la visibilité des composants d'écran avec l'élément `Visible`.

### Définition de positions de contrôle précises avec l'élément Layout

Un positionnement précis de présentation peut être obtenu en indiquant un élément `Layout` explicite et en l'associant au contrôle :

#### Format

```
<property_control ... >
  <Layout attributes
```



```

    --- spécification de cellule ---
    ...
</property_control>

```

où :

*property\_control* est l'un des contrôles propriété (reportez-vous à [Spécifications des contrôles de propriétés sur p. 151](#)).

*attributes* sont l'un de ces attributs :

Table 6-6  
Attributs de présentation

Attribut	Valeurs	Description
anchor	north northeast east southeast south southwest west northwest center	Définit le point d'ancrage du contrôle
columnWeight	0.0–1.0	Définit la manière dont une modification de la taille horizontale de la fenêtre affecte la largeur du contrôle. La somme de tous les attributs columnWeight dans un panneau ne doit pas dépasser 1,0.
fill	aucun horizontal vertical both	Définit si et comment le contrôle doit remplir les cellules auxquelles il a été affecté.
gridColumn	integer >= 0	Définit la première colonne où le contrôle doit commencer à être disposé.
gridHeight	entier	Définit le nombre de lignes sur lesquelles le contrôle doit être disposé. Une valeur de 0 (paramètre par défaut) affecte le contrôle sur toutes les lignes restantes.
gridRow	integer >= 0	Définit la première ligne où le contrôle doit être disposé. Par défaut, l'index de ligne de la grille est incrémenté automatiquement.
gridWidth	entier	Définit le nombre de colonnes sur lesquelles le contrôle doit être disposé. Une valeur de 0 (paramètre par défaut) affecte le contrôle sur toutes les colonnes restantes.
leftIndent	entier	Définit le nombre de pixels par lequel mettre en retrait le contrôle de sa position par défaut.
rowWeight	0.0–1.0	Définit la manière dont une modification de la taille verticale de la fenêtre affecte la hauteur du contrôle. La somme de tous les attributs rowWeight dans un panneau ne doit pas dépasser 1,0.

Une **spécification de cellule** permet d'indiquer la position précise d'un contrôle à l'écran. Le format est le suivant :

```
<Cell row="entier" column="entier" width="entier" />
```

où :

`row` (obligatoire) est un entier non négatif indiquant la position des lignes pour le début du contrôle.

`column` (obligatoire) est un entier non négatif indiquant la position des colonnes pour le début du contrôle.

`width` (obligatoire) est un entier non négatif indiquant le nombre de colonnes de grille d'écran que le contrôle occupe.

Par exemple, supposons une grille d'écran de trois colonnes et de trois lignes avec une présentation de contrôle personnalisée au format suivant :

Figure 6-47  
Exemples de présentation de contrôle utilisant des cellules

	colonne 0	colonne 1	colonne 2
ligne	commande 1		
ligne	commande 2		
ligne		commande 3	

requiert un élément `Layout` avec les spécifications de cellules suivantes :

```
<Layout ... >
  <Cell row="0" column="0" width="2">
  <Cell row="1" column="0" width="1">
  <Cell row="2" column="0" width="3">
</Layout>
```

Vous trouverez ci-après des exemples plus détaillés offrant d'autres illustrations de l'utilisation de l'élément `Layout`.

### Exemple : Case à cocher activant un champ de texte

Cet exemple illustre l'utilisation d'une case à cocher pour activer un champ de texte sur la même ligne de l'affichage.

Lors de l'utilisation d'une case à cocher pour activer un autre contrôle sur la même ligne, un élément `Layout` simple est requis pour permettre l'affichage correct des contrôles. (*Remarque* : Le mécanisme d'activation et de désactivation des contrôles est décrit dans [Contrôle des caractéristiques d'affichage avec l'élément Activé sur p. 196](#)).

Supposez que vous voulez obtenir le suivant dans un affichage :

Figure 6-48  
Case à cocher activant un champ de texte



Il existe deux contrôles ici :

- Une case à cocher avec une étiquette qui fait aussi office d'étiquette pour un champ de texte
- Le champ de texte lui-même

Le point de départ est une déclaration normale de deux contrôles :

```
<CheckBoxControl property="boolean3" label="Case à cocher 3"/>
<TextBoxControl property="string3" label="Chaîne 3"/>
```

Ceci entraîne le panneau semblable :

Figure 6-49

Case à cocher et champ de texte dans des lignes distinctes



D'abord, nous voulons empêcher l'affichage de l'étiquette de champ de texte Chaîne 3. Ceci est obtenu en définissant l'attribut `showLabel` du contrôle de champ de texte sur `false` (faux):

```
<CheckBoxControl property="boolean3" label="Case à cocher 3"/>
<TextBoxControl property="string3" label="Chaîne 3" showLabel="false"/>
```

Le champ de texte est développé pour remplir la zone précédemment occupée par l'étiquette :

Figure 6-50

Case à cocher, et champ de texte sans étiquette



Nous souhaitons désormais permettre l'affichage de texte sur la même ligne que la case à cocher. Pour ce faire, nous ajoutons un élément `Layout` dans l'élément `CheckBoxControl` pour définir l'incrément de ligne sur 0 (par défaut, la ligne est incrémentée de 1 pour chaque contrôle) :

```
<CheckBoxControl property="boolean3" label="Case à cocher 3">
  <Layout rowIncrement="0"/>
</CheckBoxControl>
<TextBoxControl property="string3" label="Chaîne 3" showLabel="false"/>
```

Toutefois, l'affichage obtenu ressemble à ceci :

Figure 6-51

Champ de texte écrasant la case à cocher



Le champ de texte a été monté d'une ligne. Cependant, il occupe toujours la ligne entière. Il a donc écrasé la case à cocher.

*Remarque* : Si l'affichage ressemble à ceci :

Figure 6-52

Case à cocher écrasant le champ de texte



la case à cocher est dessinée après le champ de texte. Par conséquent les quelques premiers caractères du champ de texte sont masqués.

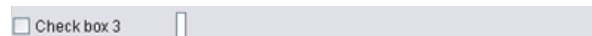
Quel que soit l'objet dessiné en premier, l'affectation de plusieurs composants IU à des résultats de cellules identiques entraîne un comportement non souhaitable ou indéfini et doit donc être évitée. Pour résoudre ce problème, nous devons ajouter un deuxième élément `Layout`, cette fois dans l'élément `TextBoxControl`, pour forcer le champ de texte à débiter dans la deuxième colonne de l'affichage :

```
<CheckBoxControl property="boolean3" label="Case à cocher 3">
  <Layout rowIncrement="0"/>
</CheckBoxControl>
<TextBoxControl property="string3" label="Chaîne 3" showLabel="false">
  <Layout gridColumn="1"/>
</TextBoxControl>
```

Toutefois, ceci n'est qu'une solution partielle, car le résultat est :

Figure 6-53

*Placement correct mais champ de texte très court*



Les deux contrôles sont placés correctement, mais le champ de texte est très court. Le problème est qu'une fois une présentation personnalisée associée à un contrôle, ceci annule les valeurs par défaut «intelligentes» associées à chaque type de contrôle. Dans ce cas, le comportement de remplissage par défaut de l'élément `Layout` (c'est à dire, la manière dont le composant remplit les cellules disponibles) est de ne pas remplir les cellules disponibles et donc d'occuper aussi peu d'espace à l'écran que possible. Pour modifier ceci, nous indiquons simplement au champ de texte de remplir l'espace horizontal :

```
<CheckBoxControl property="boolean3" label="Case à cocher 3">
  <Layout rowIncrement="0"/>
</CheckBoxControl>
<TextBoxControl property="string3" label="Chaîne 3" showLabel="false">
  <Layout gridColumn="1" fill="horizontal" columnWeight="0.001"/>
</TextBoxControl>
```

L'ajout d'une petite valeur `columnWeight` est requise pour permettre à Java d'allouer correctement l'espace rempli.

Ceci nous donne notre présentation prévue :

Figure 6-54

*Case à cocher activant un champ de texte*



Ceci semble correct, mais il reste un problème à résoudre. Actuellement, la case à cocher tente d'occuper la totalité de la ligne, même si nous plaçons un autre contrôle sur la même ligne. Le problème n'est pas actuellement visible car l'étiquette de la case à cocher est relativement courte. En outre, d'autres étiquettes sur le panneau (non indiquées dans l'illustration) ont déplacé la

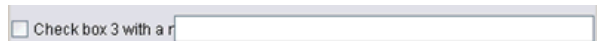
deuxième colonne d'affichage. De ce fait, il n'existe pas de chevauchement. Le problème devient évident si nous agrandissons l'étiquette de la case à cocher :

```
<CheckBoxControl property="boolean3" label="Case à cocher 3 avec une étiquette
beaucoup plus longue qu'avant">
  <Layout rowIncrement="0"/>
</CheckBoxControl>
<TextBoxControl property="string3" label="Chaîne 3" showLabel="false">
  <Layout gridColumn="1" fill="horizontal" columnWeight="0.001"/>
</TextBoxControl>
```

Ce qui nous donne les éléments suivants :

Figure 6-55

Longue étiquette de case à cocher écrasée par champ de texte



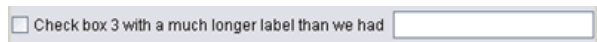
Tous ce que nous avons à faire est d'indiquer à la case à cocher de limiter sa largeur disponible à une seule colonne :

```
<CheckBoxControl property="boolean3" label="Case à cocher 3 avec une étiquette beaucoup plus longue">
  <Layout rowIncrement="0" gridWidth="1"/>
</CheckBoxControl>
<TextBoxControl property="string3" label="Chaîne 3" showLabel="false">
  <Layout gridColumn="1" fill="horizontal" columnWeight="0.001"/>
</TextBoxControl>
```

Ceci donne finalement ce dont nous avons besoin :

Figure 6-56

Longue étiquette de case à cocher entièrement affichée



### Exemple : Groupes de boutons radio et champs de texte

Cet exemple illustre un moyen d'associer chaque bouton dans un groupe de boutons radio avec son propre champ de texte.

Nous voulons définir un panneau qui ressemble à ceci :

Figure 6-57

Groupe de boutons radio avec champs de texte



Cette fois-ci, nous disposons de quatre contrôle :

- Un groupe de boutons radio pour une liste énumérée de trois valeurs
- Trois champs de texte, un pour chaque valeur

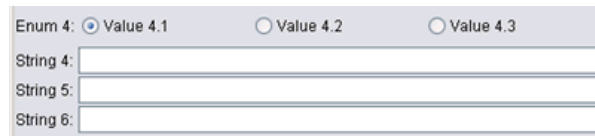
Comme dans l'exemple précédent, nous commençons par une déclaration simple des contrôles :

```
<RadioButtonGroupControl property="enum4" label="Enum 4"/>
<TextBoxControl property="string4" label="Chaîne 4"/>
<TextBoxControl property="string5" label="Chaîne 5"/>
<TextBoxControl property="string6" label="Chaîne 6"/>
```

Ceci permet d'avoir un affichage ressemblant à ceci :

Figure 6-58

Groupe de boutons radio avec champs de texte et étiquettes



Nous voulons utiliser les étiquettes des boutons radio pour identifier les champs de texte. Par conséquent, notre première tâche consiste à aligner les boutons radio dans une seule colonne de trois lignes et à masquer les étiquettes des champs de texte :

```
<RadioButtonGroupControl property="enum4" label="Enum 4" rows="3"/>
<TextBoxControl property="string4" label="Chaîne 4" showLabel="false"/>
<TextBoxControl property="string5" label="Chaîne 5" showLabel="false"/>
<TextBoxControl property="string6" label="Chaîne 6" showLabel="false"/>
```

Ceci entraîne l'affichage suivant :

Figure 6-59

Les boutons radio dans une seule colonne, et les champs de texte



Un léger problème est déjà identifié ici ; l'étiquette du groupe de boutons radios n'est pas alignée avec le premier bouton radio. Nous résoudrons ceci ultérieurement. Nous devons déjà aligner grossièrement nos champs de texte avec chaque bouton radio correctement.

La procédure est semblable à celle dans l'exemple 1. Nous devons :

- changer l'incrément de ligne du groupe de boutons radio en 1 ;
- limiter la largeur de grille pour que les champs de texte et les boutons radio ne se chevauchent pas ;
- présenter chaque champ de texte dans la même ligne que son bouton radio.

Par conséquent, nous ajoutons des éléments `Layout`, comme dans l'exemple précédent. Dans ce cas, nous modifions le fichier de spécifications comme suit :

```
<RadioButtonGroupControl property="enum4" label="Enum 4" rows="3">
  <Layout rowIncrement="0" gridWidth="1"/>
```

```

</RadioButtonGroupControl>
<TextBoxControl property="string4" label="Chaîne 4" showLabel="false">
  <Layout gridColumn="1" fill="horizontal" columnWeight="0.001"/>
</TextBoxControl>
<TextBoxControl property="string5" label="Chaîne 5" showLabel="false">
  <Layout gridColumn="1" fill="horizontal" columnWeight="0.001"/>
</TextBoxControl>
<TextBoxControl property="string6" label="Chaîne 6" showLabel="false">
  <Layout gridColumn="1" fill="horizontal" columnWeight="0.001"/>
</TextBoxControl>

```

Malheureusement, l'affichage ressemble désormais à ceci :

Figure 6-60

*Champs de texte écrasant les boutons radio*



Nous avons utilisé exactement les mêmes éléments Layout qui ont fonctionné dans l'exemple 1. Que s'est-il donc passé ?

La réponse est que, contrairement au contrôle case à cocher dans l'exemple précédent, le groupe de boutons radio (comme la plupart des contrôles) possède une étiquette distincte ainsi que le contrôle réel. Ceci signifie que le groupe de boutons radio requiert une colonne supplémentaire. Par conséquent, nous devons indiquer aux champs de texte de commencer une colonne ultérieurement, à la colonne 2 au lieu de la colonne 1. Par conséquent, dans les éléments Layout pour les champs de texte, nous définissons les valeurs gridColumn sur 2 :

```

<RadioButtonGroupControl property="enum4" label="Enum 4" rows="3">
  <Layout rowIncrement="0" gridWidth="1"/>
</RadioButtonGroupControl>
<TextBoxControl property="string4" label="Chaîne 4" showLabel="false">
  <Layout gridColumn="2" fill="horizontal" columnWeight="0.001"/>
</TextBoxControl>
<TextBoxControl property="string5" label="Chaîne 5" showLabel="false">
  <Layout gridColumn="2" fill="horizontal" columnWeight="0.001"/>
</TextBoxControl>
<TextBoxControl property="string6" label="Chaîne 6" showLabel="false">
  <Layout gridColumn="2" fill="horizontal" columnWeight="0.001"/>
</TextBoxControl>

```

Remarque : bien que nous incrémentons la colonne de grille du champ de texte à 2, nous n'augmentons pas la largeur de grille du groupe de boutons radio de 1. Et ceci car pour les contrôles de propriété, la plupart des attributs Layout n'affectent que la part éditable du contrôle, plutôt que l'étiquette du contrôle.

Nous avons désormais :

Figure 6-61

*Champs de texte n'écrasant plus les boutons radio*



Ceci est beaucoup plus proche de ce que nous voulons. Toutefois, des problèmes d'alignement subsistent entre les boutons radio et les champs de texte.

Le problème principal est que les boutons radio sont disposés dans un sous-panneau distinct. Par conséquent, il n'existe pas de réelle relation de présentation entre un bouton radio et son champ de texte. Tout ce que nous devons faire est d'empêcher le groupe de boutons radio d'utiliser un sous-panneau :

```
<RadioButtonGroupControl property="enum4" label="Enum 4" rows="3" useSubPanel="false">
  <Layout rowIncrement="0" gridWidth="1"/>
</RadioButtonGroupControl>
<TextBoxControl property="string4" label="Chaîne 4" showLabel="false">
  <Layout gridColumn="2" fill="horizontal" columnWeight="0.001"/>
</TextBoxControl>
<TextBoxControl property="string5" label="Chaîne 5" showLabel="false">
  <Layout gridColumn="2" fill="horizontal" columnWeight="0.001"/>
</TextBoxControl>
<TextBoxControl property="string6" label="Chaîne 6" showLabel="false">
  <Layout gridColumn="2" fill="horizontal" columnWeight="0.001"/>
</TextBoxControl>
```

Finalement, nous avons la présentation souhaitée :

Figure 6-62

*Groupe de boutons radio avec champs de texte*



### **Contrôle des caractéristiques d'affichage avec l'élément Activé**

Vous pouvez utiliser l'élément `Enabled` pour permettre l'activation ou la désactivation d'un contrôle, généralement selon une condition particulière qui peut être vérifiée ou non.

Des conditions peuvent être associées à des panneaux et contrôles de propriétés pour déterminer différentes caractéristiques d'affichage. Par exemple, une case à cocher peut être utilisée pour activer un champ de texte associé, ou un bouton radio peut permettre à un groupe de champs masqués d'être visibles.

Les conditions dans l'interface utilisateur sont généralement basées sur la valeur d'un autre contrôle plutôt que d'une propriété. Les conditions basées sur des propriétés ne prennent effet que lorsque les modifications ont été restaurées dans l'objet sous-jacent (par exemple, noeud, sortie de modèle, ou sortie de document). Dans l'interface utilisateur, les contrôles doivent être activés dès qu'un contrôle associé a été modifié.



**Format**

```

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

```

L'élément `Condition` indique une condition à tester pour déterminer si le contrôle est activé.

Les éléments `And`, `Or` et `Not` permettent de spécifier des conditions composées.

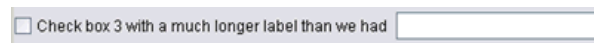
Pour plus d'informations, reportez-vous à la section [Conditions](#) dans le chapitre 4 sur p. 89.

**Exemple : Activation des contrôles avec une condition simple**

Dans [Exemple : Case à cocher activant un champ de texte sur p. 190](#), nous avons conçu une case à cocher pour activer un champ de texte lorsque la case est sélectionnée :

Figure 6-63

Longue étiquette de case à cocher entièrement affichée



Nous voulons que le champ de texte soit activé dès que la case est cochée et pas au moment de la modification de la propriété de l'objet sous-jacent. Pour ce faire, nous avons besoin d'ajouter une condition `Activé` :

```

<CheckBoxControl property="boolean3" label="Case à cocher 3 avec une étiquette beaucoup plus longue">
  <Layout rowIncrement="0" gridWidth="1"/>
</CheckBoxControl>
<TextBoxControl property="string3" label="Chaîne 3" showLabel="false">
  <Layout gridColumn="1" fill="horizontal" columnWeight="0.001"/>
  <Enabled>
    <Condition control="boolean3" op="equals" value="true"/>
  </Enabled>
</TextBoxControl>

```

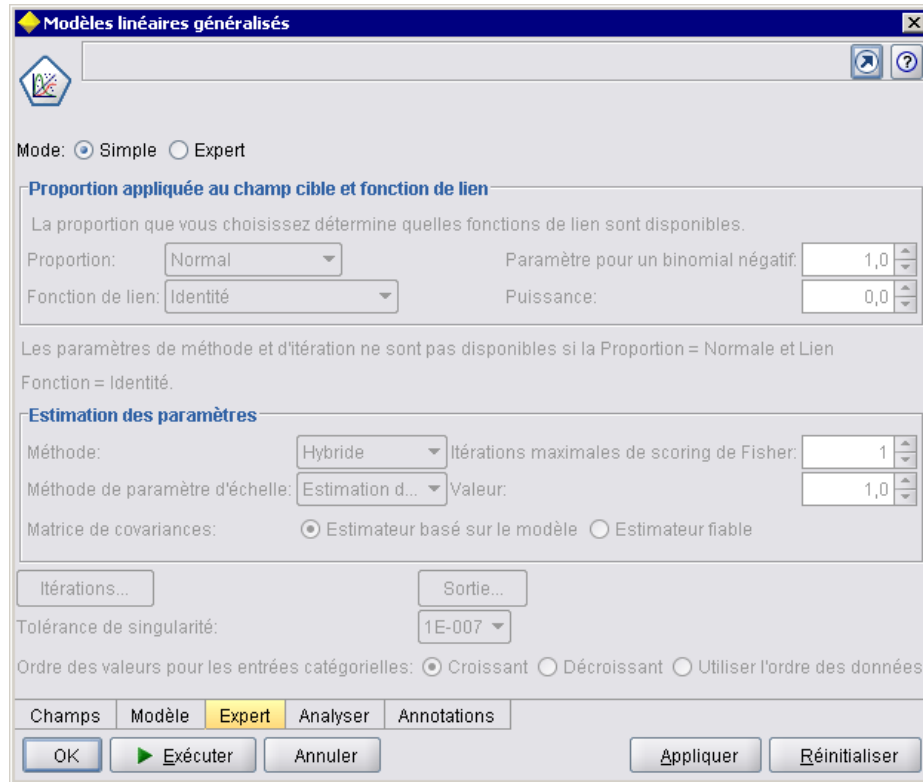
Ceci garantit que le champ de texte n'est activé que si la valeur Booléenne associée à la case à cocher est `true` (vrai).

**Exemple : Activation des contrôles avec une condition complexe**

Pour illustrer le codage de conditions complexes, nous examinons l'un des onglets de boîte de dialogue pour le noeud `Modèles linéaires`, qui a été développé à l'aide de CLEF.

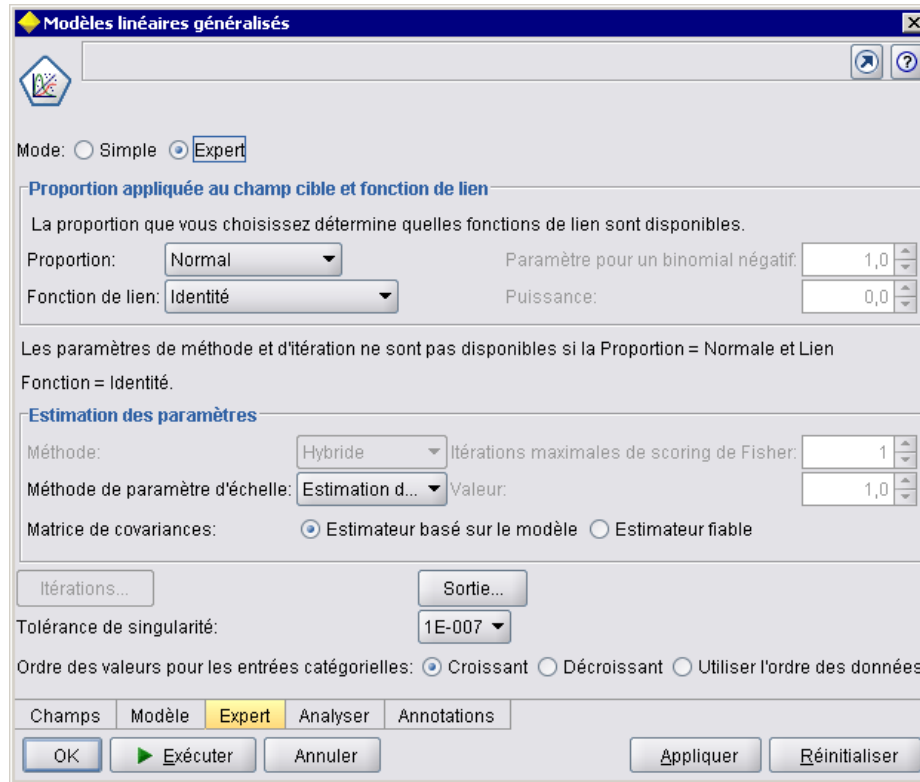
La boîte de dialogue du noeud possède un onglet Expert, qui contient des options destinées aux utilisateurs avec une connaissance détaillée de tels modèles. Toutes les options sur l'onglet sont au départ désactivées :

Figure 6-64  
Onglet Expert avec toutes les options désactivées



La définition de la case à cocher Mode sur Expert active un certain nombre de ces options :

Figure 6-65  
Onglet Expert avec Mode défini sur Expert



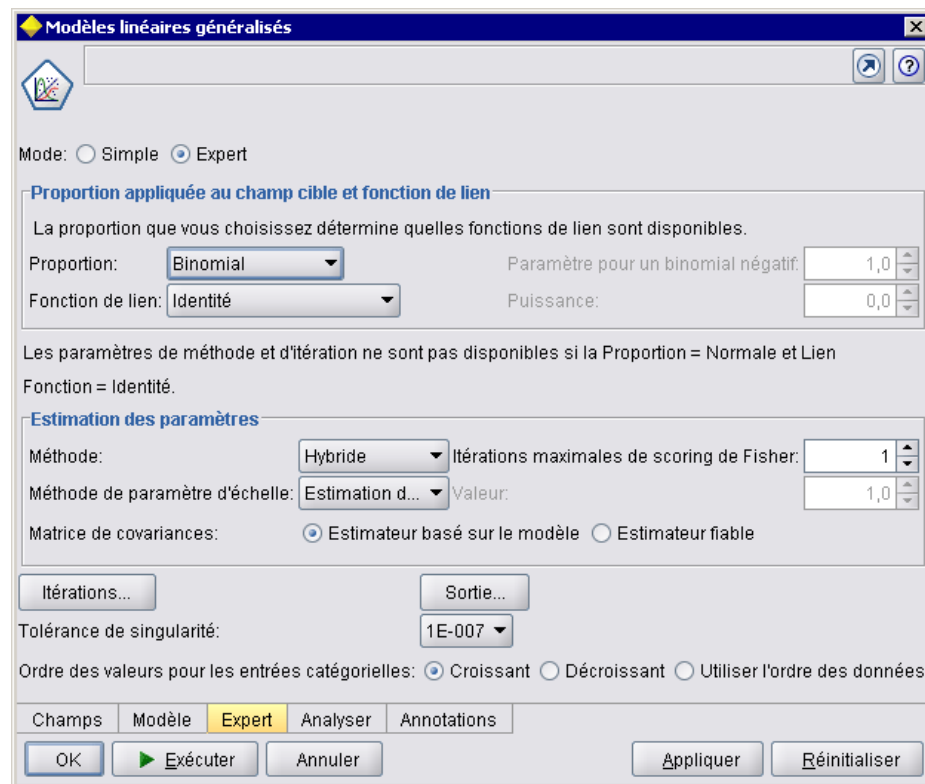
Toutefois, certains restent désactivés, comme le contrôle Itérations vers le bas de la boîte de dialogue. Ce contrôle est désactivé uniquement lorsque les **deux** conditions suivantes sont true (vraies) :

- Distribution est défini sur Normal
- Fonction de lien est défini sur Identité

Cette combinaison est en fait le paramétrage par défaut pour cet onglet en mode Expert. En outre, la modification du paramètre de l'une de ces zones de liste déroulantes permet l'activation de ces Itérations :

Figure 6-66

Le changement en paramètre Distribution active le bouton Itérations



Le code pour obtenir ceci est contenu dans une déclaration PropertiesSubPanel pour le bouton Itérations, comme suit :

```
<PropertiesSubPanel buttonLabel="Itérations..." 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>
```

L'élément `Condition` dans la section `And` indique que `Mode` doit être défini sur `Expert` avant toute modification préalable. A condition que cette condition est vraie (vraie), la section `Not` indique que le bouton n'est pas activé (c'est à dire, est désactivé lorsque les *deux* conditions de la section `And` sont réunies. Ainsi en mode `Expert`, le bouton `Itérations` est activé si `Distribution` ou `Fonction de lien` est défini sur une option autre que la valeur par défaut.

### **Contrôle des caractéristiques d'affichage avec l'élément visible**

Vous pouvez aussi utiliser des conditions pour permettre l'affichage ou le masquage des contrôles selon les circonstances spécifiées. Pour ce faire, utilisez l'élément `Visible`.

#### **Format**

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

L'élément `Condition` indique une condition à tester pour déterminer si le contrôle est visible.

Les éléments `And`, `Or` et `Not` permettent de spécifier des conditions composées.

Pour plus d'informations, reportez-vous à la section `Conditions` dans le chapitre 4 sur p. 89.

#### **Exemple**

L'exemple suivant permet l'affichage du panneau de propriétés spécifié uniquement si la condition `source_language` est réunie :

```
<PropertiesPanel>
  <Visible>
    <Condition control="source_language" op="equals" value="eng" />
  </Visible>
  ...
</PropertiesPanel>
```

## **Fenêtres de sortie personnalisées**

Pour des objets sortie de modèle, sortie de document et sortie interactive (mais pas des noeuds), il est possible qu'une extension remplace entièrement la fenêtre de sortie par défaut par une fenêtre personnalisée. Ceci est mis en œuvre par une classe `java.awt.Frame` standard.

Pour offrir une fenêtre personnalisée, indiquez une classe Java comme attribut `frameClass` de l'élément `UserInterface`, comme suit :

```
<DocumentOutput id="my_modelling_node" type="modelBuilder" ...>
  <Properties>
```

```
<Property name="use_custom_type" valueType="boolean" .../>
...
</Properties>
<UserInterface frameClass="com.myextension.MyOutputFrame"/>
...
</DocumentOutput>
```

La classe spécifiée doit mettre en œuvre l'interface `ExtensionObjectFrame` définie par l'API côté client de CLEF. Ceci couvre le cycle de vie de la fenêtre :

- Accès à l'objet `java.awt.Frame` sous-jacent
- Initialisation de fenêtre, dont l'accès à l'objet de sortie et à la session
- Synchronisation de la fenêtre et de l'objet sous-jacent lorsque l'objet est sur le point d'être enregistré ou supprimé
- Suppression de fenêtre

Pour plus d'informations, reportez-vous à la section [Classes de l'API côté client](#) dans le chapitre 9 sur p. 217.

# ***Ajout d'un système d'aide***

## ***Types de système d'aide***

Lors du développement d'une extension CLEF, vous souhaitez généralement inclure un système d'aide en ligne pour expliquer comment utiliser cette extension. CLEF prend en charge les types de système d'aide suivants :

- Aide au format HTML
- JavaHelp

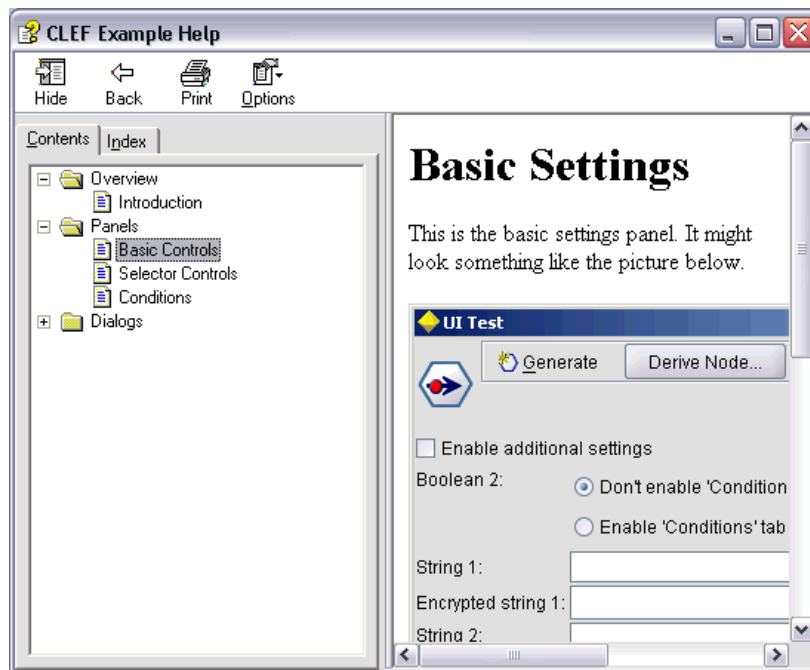
## ***Aide au format HTML***

L'aide au format HTML est un format propriétaire développé par Microsoft, qui ne fonctionne que sur les plates-formes Windows. Un système d'aide HTML est constitué d'une série de fichiers .htm ou .html, compilés en un seul fichier au format compressé portant l'extension .chm. L'aide de IBM® SPSS® Modeler est fournie au format HTML.

L'aide HTML prend en charge les fonctions de table des matières, d'index et de recherche sur l'intégralité du texte (des termes de glossaire peuvent même être intégrés sous forme de fenêtres contextuelles). Vous pouvez créer les fichiers de rubrique .htm ou .html source dans un éditeur HTML ou à l'aide d'un outil auteur commercial de création de système d'aide. Pour créer le fichier .chm, une des options est d'utiliser HTML Help Workshop, à télécharger gratuitement sur le site Web du Centre de téléchargement Microsoft (pour plus d'informations sur la création de fichiers .chm, reportez-vous au système d'aide de HTML Help Workshop). Sinon, vous pouvez utiliser un outil auteur de création de système d'aide prenant en charge le format d'aide HTML pour compiler vos fichiers de rubrique et les éventuels graphiques en un fichier .chm.

Le navigateur d'aide HTML ressemble à ceci :

Figure 7-1  
Navigateur d'aide HTML



## JavaHelp

JavaHelp est un format d'aide Open Source développé par Sun Microsystems, qui fonctionne sur toutes les plates-formes compatibles Java. Un système JavaHelp inclut les fichiers suivants :

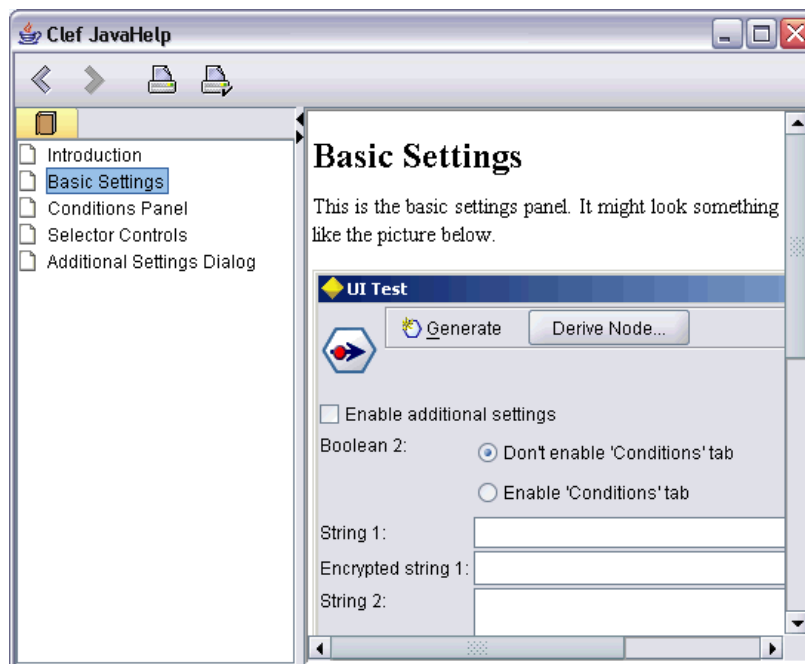
- Les fichiers de rubrique .htm ou .html source
- Les fichiers graphiques utilisés dans les rubriques
- Un fichier de système d'aide (portant l'extension .hs) qui contrôle ce système
- Un fichier map.xml servant à associer les ID de rubrique avec les fichiers correspondants et à définir la fenêtre qui affiche les rubriques d'aide
- Un fichier index.xml contenant les entrées d'index
- Un fichier toc.xml contenant les entrées de la table des matières

JavaHelp prend en charge les fonctions de table des matières, d'index, de recherche sur l'intégralité du texte et de glossaire. Vous pouvez créer les fichiers de rubrique .htm ou .html source dans un éditeur HTML ou à l'aide d'un outil auteur commercial de création de système d'aide. Vous devrez aussi utiliser le logiciel JavaHelp, disponible gratuitement sur le site Web Sun Developer Network (pour plus d'informations, reportez-vous au *Manuel d'utilisation du système JavaHelp*, également disponible sur le site Web).



Le navigateur JavaHelp ressemble à ceci :

Figure 7-2  
Navigateur JavaHelp



## Mise en oeuvre d'un système d'aide

Cette section explique comment définir les composants pertinents du système d'aide dans le fichier de spécifications.

### Définition de l'emplacement et du type du système d'aide

Le type du système d'aide éventuel utilisé pour une extension est défini dans un élément `HelpInfo` figurant dans la section `Resources` du fichier de spécifications de cette extension.

#### Format

```
<Resources>
...
  <HelpInfo id="name" type="help_type" path="help_path" helpset="helpset_loc"
    default="default_topicID" />
...
</Resources>
```

où :

`id` (obligatoire) est l'identificateur des informations d'aide de l'extension.

`type` (obligatoire) indique le type du système d'aide et a l'une des valeurs suivantes :

- `htmlhelp` : aide HTML, contenue dans le fichier `.chm` identifié par l'attribut `path`.
- `javahelp` : aide JavaHelp utilisant le fichier de système d'aide (`.hs`) identifié par l'attribut `helpset`, avec les fichiers source et fichiers associés.

Si le type de l'aide est `htmlhelp`, vous devez préciser l'attribut supplémentaire suivant :

- `path` : emplacement (relatif par rapport au fichier de spécifications) et nom du fichier `.chm` contenant le système d'aide.

Si le type de l'aide est `javahelp`, vous devez préciser les attributs supplémentaires suivants :

- `helpset` : emplacement (relatif par rapport au fichier de spécifications) et nom du fichier de système d'aide `.hs` à utiliser.
- `default` : identificateur de la rubrique par défaut à afficher si l'utilisateur ne précise aucune rubrique pour un onglet donné.

Si aucun élément `HelpInfo` n'est indiqué, cette extension ne possède pas de système d'aide.

### **Exemples**

Le premier exemple illustre un élément `HelpInfo` pour une aide HTML :

```
<HelpInfo id="help" type="htmlhelp" path="help/mynode.chm" />
```

L'équivalent pour un système JavaHelp est le suivant :

```
<HelpInfo id="help" type="javahelp" helpset="help/mynode.hs"/>
```

Attention, dans le cas de JavaHelp, les fichiers associés (images, fichier de carte, index et fichiers de contenu) doivent se trouver dans le même dossier que le fichier de système d'aide `.hs`.

## **Désignation d'une rubrique particulière à afficher**

Vous pouvez désigner la rubrique d'aide particulière à afficher si l'utilisateur appelle l'aide à partir d'une boîte de dialogue de noeud, d'un onglet particulier ou d'un sous-panneau de propriétés. Pour ce faire, vous utilisez l'attribut `helpLink` de spécification du noeud, de l'onglet ou du sous-panneau de propriétés.

Si vous ne précisez pas l'attribut `helpLink`, la rubrique par défaut du système d'aide apparaît lorsque l'utilisateur appelle l'aide.

Pour plus d'informations, reportez-vous aux descriptions de l'attribut `helpLink` dans [Noeud sur p. 61](#), [Onglets sur p. 140](#) et [Sous-panneau Propriétés sur p. 155](#).

### **Exemple**

Cet exemple part de l'hypothèse que vous utilisez l'aide HTML ; il vous montre comment provoquer l'affichage de différentes rubriques contextuelles, en fonction de la fenêtre qui est active lorsque l'utilisateur sélectionne la commande d'aide.

```

<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>

```

Ce script indique que, si l'onglet Commandes de base est actif et que l'utilisateur appelle l'aide, la rubrique issue du fichier `basic_controls.htm` dans `mynode.chm` est affichée. Si l'utilisateur clique ensuite sur le bouton Paramètres supplémentaires pour ouvrir la boîte de dialogue du même nom et sélectionne Aide à cet endroit, la rubrique du fichier `addsettingsdlg.htm` est affichée. Si l'utilisateur ferme la boîte de dialogue Paramètres supplémentaires, clique sur l'onglet Commandes de sélection et choisit Aide une nouvelle fois, le système affiche la rubrique correspondant au fichier `selector_controls.htm`.

Pour JavaHelp, la valeur de l'attribut `helpLink` doit correspondre à celle de l'attribut `target` figurant dans le fichier `map.xml`. Par exemple, si le fichier `map.xml` inclut les éléments suivants :

```

<map version="1.0">
...
  <mapID target="basic_controls" url="basic_controls.htm"/>
...
</map>

```

Vous devez donner à l'attribut `helpLink` correspondant la valeur suivante :

```
helpLink="basic_controls"
```

En effet, lorsque l'aide Java est appelée, JavaHelp lit la valeur de l'attribut `target` et le mappe à la valeur url associée pour trouver le fichier à afficher.

# Localisation et accessibilité

## Introduction

La **localisation** désigne le processus d'adaptation d'un logiciel, de l'aide et de la documentation pour une région particulière. Cela comprend la traduction de l'interface utilisateur, de l'aide et de la documentation ainsi que le test du système dans la région appropriée. Si vous êtes amenés à distribuer votre extension à des utilisateurs se trouvant dans des régions différentes de la vôtre, vous pouvez distribuer des versions localisées de l'extension.

L'**accessibilité** désigne, dans ce contexte, le fait d'inclure des fonctions dans l'interface utilisateur qui rendent l'accès au système plus simple pour les utilisateurs handicapés, tel les malvoyants ou les personnes à dextérité manuelle réduite.

## Localisation

IBM® SPSS® Modeler a lui-même été localisé pour plusieurs régions du monde. Pour toutes les langues prises en charge, lorsque l'utilisateur définit l'option régionale Windows sur son propre paramètre régional, les composants IU de IBM® SPSS® Modeler apparaissent dans cette langue, par exemple :

- les menus système et les entrées des menus
- les boutons système (Générer, OK, Exécuter, Annuler, Appliquer, Réinitialiser)
- les onglets de boîtes de dialogue standard (Annotations et Déboguer, s'ils sont utilisés)
- Les messages d'erreur et les messages système (par exemple, « Cet objet n'a pas été enregistré. »)

Aux endroits où votre extension utilise ces composants SPSS Modeler standard, ceux-ci seront automatiquement affichés dans la langue sélectionnée, si elle est prise en charge.

En ce qui concerne les autres composants de votre extension, CLEF propose une fonction d'aide à la localisation. Vous pouvez localiser :

- les noms des noeuds (sur les palettes et les espaces de travail)
- les noms des modèles (dans l'onglet Modèles du panneau du gestionnaire)
- les noms des documents (dans l'onglet Sorties du panneau du gestionnaire)
- l'emplacement d'une image icône associée à une action
- le texte des info-bulles
- les systèmes d'aide
- les boîtes de dialogue des noeuds :
  - le texte de la barre de titre
  - les menus personnalisés et les entrées des menus
  - les étiquettes des champs, des propriétés, des boutons et des onglets

- les textes statiques
- les messages système et les messages d'erreur

Les chaînes de texte doivent être relativement courtes pour permettre un texte plus long lorsque l'élément est traduit.

Les messages système et les messages d'erreur peuvent être localisés grâce à une combinaison du fichier de spécifications, des fichiers de propriétés et de l'API côté serveur. [Pour plus d'informations, reportez-vous à la section Document Détails de l'état dans le chapitre 9 sur p. 238.](#)

## ***Fichiers de propriétés***

Les chaînes de texte des éléments que vous pouvez localiser sont stockées dans des fichiers appelés **fichiers de propriétés**, qui utilisent un format Java standard pour le stockage des groupes de ressources de localisation. Chaque fichier de propriétés se compose d'une série d'enregistrements, une pour chaque élément localisé de l'extension. Un champ dans chaque enregistrement correspond à un attribut `labelKey` dans le fichier de spécifications, ce qui permet à CLEF de lire la chaîne de texte correspondante dans le fichier de propriétés et de l'afficher à l'emplacement correct.

Un fichier de propriétés doit posséder l'extension `.properties`, et doit se trouver dans le même répertoire que le fichier de spécifications du noeud auquel il est lié. IBM® SPSS® Modeler recherche initialement le fichier de propriétés par défaut qui porte le nom :

`path.properties`

où `path` est la valeur de l'attribut `path` dans l'élément `Bundle` (dans la section `Resources`) qui définit le groupe de propriétés. Par exemple :

```
<Bundle id="bundle" path="my_resources"/>
```

S'il n'y a pas de fichier de propriétés par défaut, SPSS Modeler lit les chaînes de texte à partir des définitions du fichier de spécifications.

Il doit y avoir un fichier de propriétés pour chaque langue prise en charge par la localisation. Les fichiers pour les langues autres que la langue par défaut se distinguent par un suffixe dans le nom du fichier. Par exemple :

```
my_resources.properties
my_resources_de.properties
my_resources_fr.properties
```

Les suffixes sont conformes à la norme ISO 639-1 à deux caractères pour les codes de langue.

Chaque enregistrement dans un fichier de propriétés a le format suivant :

```
id=text_string
```

où :

`id` est l'identificateur provenant d'un attribut `buttonLabelKey`, `descriptionKey`, `dialogTitleKey`, `falseLabelKey`, `imagePathKey`, `labelKey`, `messageKey`, `textKey`, ou `trueLabelKey` dans le fichier de spécifications. Cet identificateur est généralement suivi du suffixe `.LABEL` qui permet de le

reconnaître facilement, bien qu'il puisse être suivi de n'importe quel suffixe ou d'aucun suffixe, suivant comme il apparaît dans le fichier de spécifications.

*text\_string* correspond au texte de l'élément.

### **Exemple : Localisation d'un onglet de boîte de dialogue**

Cet exemple d'un onglet localisé dans une boîte de dialogue de noeud utilise deux fichiers de propriétés, la version par défaut (en anglais) et la version en français aux emplacements suivants :

```
extension_folder\my_resources.properties
extension_folder\my_resources_fr.properties
```

où *extension\_folder* représente le dossier contenant le fichier de spécifications.

Dans le fichier de spécifications, les fichiers de propriétés sont référencés au moyen d'un élément **Bundle** dans la section **Resources** :

```
<Resources>
  <Bundle id="bundle" type="properties" path="my_resources"/>
</Resources>
```

Notez que l'attribut *path* ne doit pas comprendre d'extension de langue ni de suffixe *properties*.

Les autres parties pertinentes du fichier de spécifications sont :

```
<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>
```

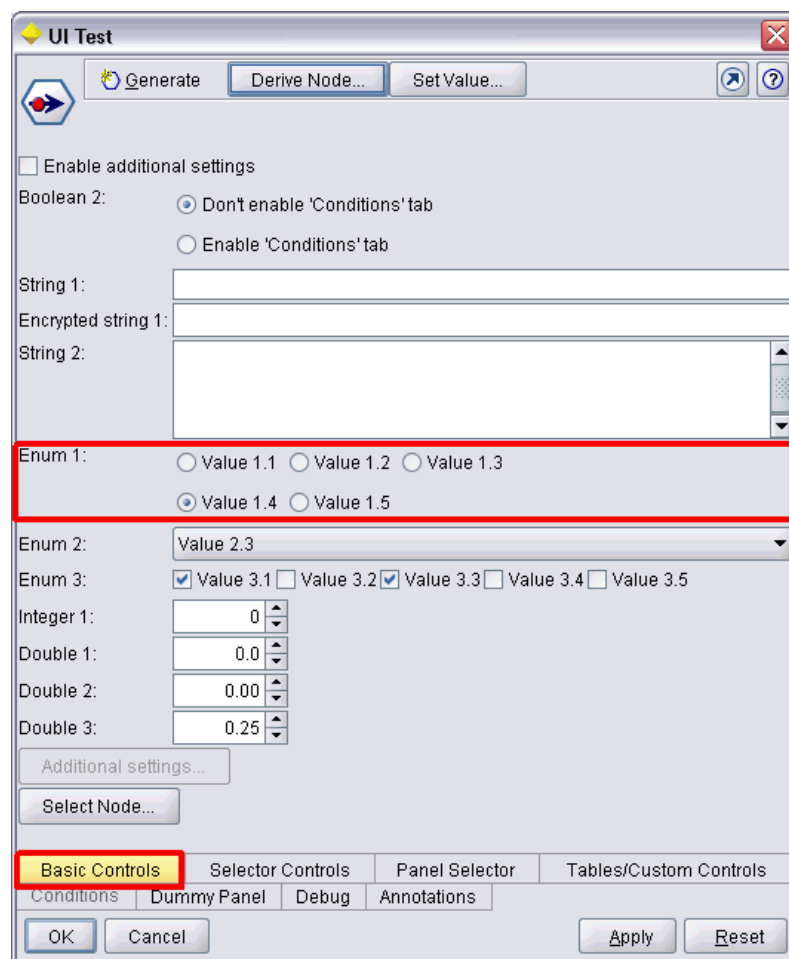
Dans le fichier de propriétés, la version en anglais du fichier de propriétés contient les enregistrements suivants :

```
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

Les parties de la boîte de dialogue concernées par ces enregistrements sont mises en évidence ici :

**Figure 8-1**  
 Onglet non localisé

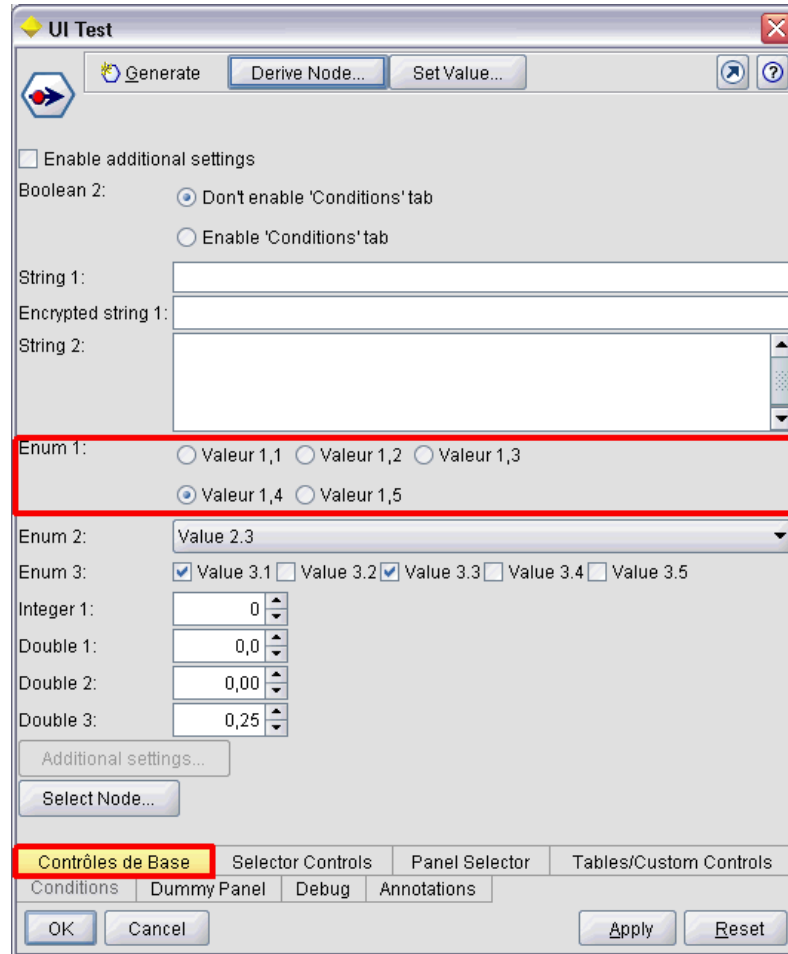


La section correspondante dans la version française du fichier de propriétés(my\_reources\_fr.properties) est :

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

Ces enregistrements provoquent l'affichage du texte traduit dans les parties concernées de l'écran :

Figure 8-2  
Onglet localisé



Notez que la localisation des quatre boutons au bas de l'écran est gérée par une fonction IBM® SPSS® Modeler standard et non par CLEF.

### **Exemple : utilisation de caractères spéciaux**

Dans le fichier de propriétés, vous devrez utiliser des séquences d'échappement Unicode pour tout caractère spécial dans des chaînes texte ASCII standard. Voici une partie d'un fichier de propriétés qui a été localisé en français :

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
```

```
...
```

Pour les langues qui utilisent des caractères non latin, comme le japonais ou le chinois, vous devrez utiliser des séquences d'échappement Unicode pour la totalité des chaînes de texte. Voici le même ensemble d'enregistrements traduits en japonais :

```
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
```

```
...
```

## ***Fichiers d'aide***

Si vous traduisez une extension qui possède un système d'aide, vous devrez aussi fournir une version localisée du système d'aide. Vous devez fournir un système d'aide localisé pour chaque extension localisée.

### ***Localisation de l'aide au format HTML***

Si l'extension que vous localisez utilise un fichier d'aide au format HTML (avec le suffixe .chm), vous pouvez remplacer le fichier .chm par défaut par une version traduite. Pour plus d'informations sur les systèmes d'aide au format HTML reportez-vous à [Aide au format HTML sur p. 203](#).

Pour créer un fichier localisé .chm :

- ▶ Créez des versions traduites des fichiers individuels des rubriques d'aide (.htm ou .html) qui constituent le système d'aide en conservant les mêmes noms de fichier.
- ▶ Si vous le souhaitez, utilisez des versions localisées des graphiques compris dans le système d'aide (par exemple, des captures d'écran).
- ▶ Utilisez Microsoft HTML Help Workshop ou tout autre outil auteur d'aide pour compiler les fichiers en un fichier .chm localisé.
- ▶ Testez le système d'aide avec le noeud localisé. [Pour plus d'informations, reportez-vous à la section Test d'un noeud CLEF localisé sur p. 214.](#)

### ***Localisation de JavaHelp***

Si l'extension que vous localisez utilise un système JavaHelp, vous devrez fournir des versions localisées des fichiers d'aide source pour chaque langue prise en charge. JavaHelp prend en charge l'affichage de la bonne version localisée si elle existe. [Pour plus d'informations, reportez-vous à la section JavaHelp dans le chapitre 7 sur p. 204.](#)

Pour créer un système JavaHelp localisé :

- ▶ Créez des versions traduites des fichiers individuels des rubriques d'aide (.htm ou .html) qui constituent le système d'aide en conservant les mêmes noms de fichier.
- ▶ Si vous le souhaitez, utilisez des versions localisées des graphiques compris dans le système d'aide (par exemple, des captures d'écran).
- ▶ Générez l'ensemble de l'aide et les autres fichiers requis (fichiers de mappage, fichiers de contenu et d'index).
- ▶ Testez le système d'aide avec le noeud localisé. [Pour plus d'informations, reportez-vous à la section Test d'un noeud CLEF localisé sur p. 214.](#)

### **Test d'un noeud CLEF localisé**

Pour tester un noeud localisé et son système d'aide :

- ▶ Dans la section Resources du fichier de spécifications du noeud localisé, changez l'attribut path de l'élément HelpInfo pour qu'il fasse référence au fichier .chm ou .hs localisé. Par exemple, pour l'aide au format HTML, vous pouvez utiliser :

```
<Resources>
...
  <HelpInfo id="help" type="HTMLHelp" path="help/mynode_fr.chm "/>
</Resources>
```

Pour JavaHelp, vous pouvez utiliser :

```
<Resources>
...
  <HelpInfo id="help" type="javahelp" helpset="help/mynode_fr.hs "/>
</Resources>
```

- ▶ Copiez le fichier localisé .chm ou .jar à l'emplacement indiqué dans l'attribut path.
- ▶ Configurez la région Windows sur celle souhaitée :  
Panneau de configuration > Options régionales et linguistiques > Options régionales > Standards et formats > <Langage>
- ▶ Démarrez IBM® SPSS® Modeler en vérifiant que l'affichage est dans la langue souhaitée.
- ▶ Ajoutez le noeud localisé à SPSS Modeler. [Pour plus d'informations, reportez-vous à la section Test d'une extension CLEF dans le chapitre 10 sur p. 246.](#)
- ▶ Placez une copie du noeud dans l'espace de travail.  
Ouvrez la boîte de dialogue du noeud et vérifiez qu'elle s'affiche correctement dans la langue souhaitée.
- ▶ Cliquez sur le bouton Aide de la boîte de dialogue et vérifiez que la bonne rubrique d'aide s'affiche dans la langue souhaitée.

## Accessibilité

Les noeuds CLEF bénéficient de toutes les fonctions d'accessibilité IBM® SPSS® Modeler standard, tels que les équivalents clavier des actions de la souris et la prise en charge d'un lecteur d'écran. [Pour plus d'informations, reportez-vous à la section Présentation de l'accessibilité dans IBM SPSS Modeler dans l'annexe A dans \*Guide de l'utilisateur de IBM SPSS Modeler 14.2\*.](#)

De plus, vous pouvez fournir un noeud CLEF avec du texte d'info-bulle personnalisé à des fins d'accessibilité.

Vous pouvez aussi spécifier des combinaisons de touches afin d'offrir aux utilisateurs finaux un autre accès à plusieurs fonctions de l'interface utilisateur que vous avez ajoutées à CLEF. [Pour plus d'informations, reportez-vous à la section Touches d'accès et raccourcis clavier dans le chapitre 6 sur p. 142.](#)

Pour les boutons d'action et pour chacun des composants d'écran qui sont classés comme contrôleurs (telles les cases à cocher ou les cases d'option), vous pouvez définir :

- modifiée :
- Description

L'**étiquette** est le texte qui s'affiche à l'écran comme nom du composant et qui peut être lu par un logiciel de lecture d'écran. Pour les utilisateurs malvoyants, la taille de la police d'affichage de l'étiquette peut être modifiée par des commandes sur l'onglet Afficher de la boîte de dialogue Options utilisateur, à laquelle vous pouvez accéder depuis :

Outils > Options utilisateur

La **description** est le texte de l'info-bulle qui s'affiche lorsque le pointeur de la souris passe sur le composant. L'info-bulle offre davantage d'informations sur le composant que celles transmises par le nom seul. Les infos-bulles peuvent aussi être lues par un lecteur d'écran configuré pour les lire.

Les étiquettes et les descriptions sont définies au moyen des attributs `label` et `description` dans l'élément qui définit le composant dans le fichier de spécifications. Tous deux peuvent être localisés au moyen des attributs `labelKey` et `descriptionKey` respectivement.

### Exemple

Cet exemple de bouton d'action illustre l'utilisation des fonctions d'étiquette et de description.

```
<Action id="setValue" label="Définir Valeur..." labelKey="setValue.LABEL"
description="Définit une valeur" descriptionKey="setValue.TOOLTIP"/>
```

# Programmation

## A propos de la programmation pour les noeuds CLEF

Pour permettre à un noeud d'effectuer le traitement qui ne peut pas être défini dans le fichier de spécifications, CLEF propose les interfaces API suivantes vers lesquelles vos programmes peuvent effectuer des appels :

- **API côté client.** Une API Java peut être utilisée par les extensions qui requièrent des contrôles supplémentaires, des composants de l'interface utilisateur ou une interactivité non fournie directement par le fichier de spécifications.
- **API Predictive Server (API PS).** Une API Java qui présente les fonctionnalités de IBM® SPSS® Modeler à utiliser par les applications nécessitant des capacités de Data mining et d'analyse prédictive. L'API PSA est l'utilitaire de Data mining SPSS Modeler partageant la même technologie sous-jacente.
- **API côté serveur.** API à base C qui couvre des aspects tels que la définition et l'obtention des paramètres d'exécution, la persistance de ces paramètres, les commentaires relatifs à l'exécution, le contrôle du travail (par exemple, l'interruption d'exécution), la génération SQL et les objets renvoyés.

## CLEF Documentation de l'API

Les sections suivantes offrent une présentation des API côté client et côté serveur. Une documentation de l'API plus complète est fournie sous la forme d'un fichier zip dans une installation de IBM® SPSS® Modeler et doit être extraite avant de pouvoir être utilisée.

Pour extraire la documentation de l'API :

- ▶ Recherchez le fichier *clef\_apidoc.zip* dans le produit DVD, dans le dossier *\Documentation\en*.
- ▶ A l'aide de WinZip ou d'un outil semblable, extrayez le contenu du fichier zip dans le répertoire souhaité. Cette opération permet de créer un sous-dossier *clef\_apidoc* dans ce répertoire, contenant toute la documentation API.

Pour afficher la documentation de l'API :

- ▶ Accédez au sous-dossier *clef\_apidoc* et ouvrez le fichier *clef\_apidoc.htm*.
- ▶ Choisissez l'option API PS/côté client ou côté serveur, comme vous le souhaitez.

## API côté client

CLEF offre un certain nombre de classes Java contenant des méthodes que vous pouvez utiliser pour le traitement côté client. Par exemple, la classe `DataModelProvider` permet de calculer le modèle de données de sortie où les modifications apportées au modèle de données d'entrée sont trop complexes pour utiliser les fonctionnalités fournies par le fichier de spécifications.

### Classes de l'API côté client

Les classes côté client sont les suivantes :

Table 9-1  
Classes de l'API côté client

Class	Description
ActionHandler	Permet à l'extension de gérer les actions demandées par l'utilisateur via les options de menu et les boutons de barre d'outils
DataModelProvider	Permet aux noeuds qui apportent des modifications complexes au modèle de données d'utiliser Java pour calculer le modèle de données de sortie.
ExtensionObjectFrame	Définit les fonctionnalités associées aux fenêtres utilisées pour afficher les sorties de modèle ou de document
ExtensionObjectPanel	Définit les fonctionnalités associées aux panneaux d'objets d'extension
PropertyControl	Définit les fonctionnalités associées à un contrôle de propriété personnalisé sur un panneau de propriétés

De plus amples informations sur ces classes sont disponibles dans la documentation de l'API côté client [Pour plus d'informations, reportez-vous à la section CLEF Documentation de l'API sur p. 216.](#)

### Utilisation de l'API côté client

Pour inclure les appels de fonctions côté client dans un noeud CLEF :

- ▶ Créez les fichiers source `.java` contenant les appels de fonctions.
- ▶ Compilez les fichiers source dans les fichiers `.class`.
- ▶ Vous pouvez combiner plusieurs fichiers `.class` dans un fichier `.jar` et inclure une référence vers le fichier `.jar` dans le fichier de spécifications—par exemple :

```
<Resources>
...
  <JarFile id="selfjar" path="selflearning.jar"/>
...
</Resources>
```

Certains éléments CLEF permettent de faire explicitement référence à une classe. Par exemple, vous pouvez inclure une référence de classe dans l'attribut `controlClass` d'un élément `PropertyControl` dans le fichier de spécification, comme indiqué ci-dessous :

```
<PropertyControl property="target_field_values_specify" ...  
  controlClass="com.spss.clef.selflearning.propertycontrols.list.CustomListControl" ... />
```

où `CustomListControl` est le nom de la classe qui implémente le contrôle de propriété et `com.spss.clef.selflearning.propertycontrols.list` est le chemin d'accès à cette classe dans le fichier `jar` déclaré dans l'élément `JarFile`.

Pour plus d'informations, reportez-vous à la section [Section Resources dans le chapitre 4 sur p. 42](#).

Il pourrait également vous être utile de consulter le code source pour les exemples de noeuds fournis avec cette version. [Pour plus d'informations, reportez-vous à la section Analyse du code source dans le chapitre 3 sur p. 36](#).

## **API Predictive Server (API PS)**

L'API PS offre une interface de programmation à la technologie Predictive Server sous-jacente. Les éléments principaux de l'API sont spécifiés sous la forme d'interfaces Java. La plupart de ces interfaces sont mises en œuvre via des classes internes fournies par l'API PS. Toutefois, elles ne font pas partie des spécifications de API PS. Cette approche vise à protéger l'utilisateur de l'API PS des modifications apportées à la technologie Predictive Server (comme les modifications architecturales, les modifications des protocoles client/serveur privés, etc.).

De plus amples informations sur ces classes sont disponibles dans la documentation de l'API PS. [Pour plus d'informations, reportez-vous à la section CLEF Documentation de l'API sur p. 216](#).

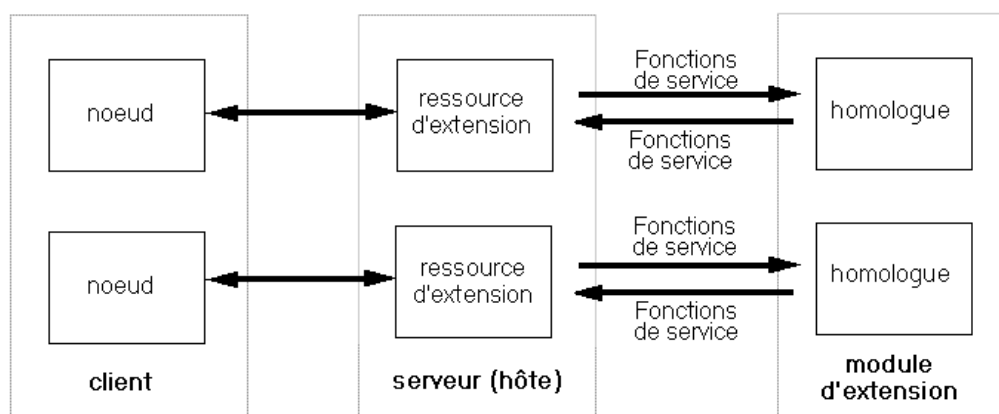
## **API côté serveur**

L'API côté serveur est définie comme une API de langage C. Toutefois, elle prend en charge les implémentations dans C++. Le développeur d'un module d'extension peut choisir de programmer directement avec l'API de langage C en programmant dans C ou C++. D'autres langages peuvent être utilisés si le développeur dispose d'une méthode de liaison à l'API C. CLEF offre également plusieurs fichiers d'aide source C++ qui font office de wrappers pour certaines des API C.

## **Architecture**

Un **noeud** d'extension sur le client est complété par un **homologue** d'extension sur le serveur. Un homologue est défini par un **module d'extension**, qui est mis en œuvre comme bibliothèque partagée hébergée par le serveur. La communication entre un noeud et son homologue est arbitrée par une **ressource** d'extension gérée par le serveur. Une ressource appelle les **fonctions de service** définies par le module d'extension pour créer et manipuler son homologue. Quant à ce dernier, il utilise les **fonctions de rappel** pour demander des informations et des services depuis son hôte.

Figure 9-1  
CLEF Architecture de l'API



## Fonctions de service

Les fonctions de service sont mises en œuvre par le module d'extension. Ce dernier doit mettre en œuvre toutes les fonctions désignées comme étant obligatoires. En outre, il est susceptible d'implémenter celles qui ne le sont pas.

Deux types de fonctions de service sont disponibles :

- Fonctions de module
- Fonctions d'homologue

Les sections suivantes offrent une présentation des fonctions de service. D'autres descriptions détaillées de ces fonctions sont disponibles dans la documentation de l'API côté serveur, comme suit :

- ▶ Dans l'écran Documentation de l'API CLEF, choisissez Présentation de l'API côté serveur.
- ▶ Cliquez sur l'onglet Modules.
- ▶ Sélectionnez Fonctions de service de l'API à mettre en œuvre par le module d'extension.

Pour plus d'informations sur l'accès à la documentation de l'API CLEF, reportez-vous à [CLEF Documentation de l'API sur p. 216](#).

## Fonctions de module

Les fonctions de module sont appelées depuis un seul thread.

Fonction	Obligatoire ?	Description
clemext_initialise	Oui	Initialise un module d'extension
clemext_cleanup	Oui	Supprime les ressources attribuées par un module d'extension

Fonction	Obligatoire ?	Description
clemtxt_getModuleInformation	Non	Obtient des informations sur un module d'extension
clemtxt_create_peer	Oui	Crée une instance homologue pour un noeud d'extension
clemtxt_destroy_peer	Oui	Supprime une instance homologue

### **Fonctions d'homologue**

Les fonctions d'homologue sont appliquées à un descripteur d'instance homologue renvoyé par un appel précédent à `clemtxt_create_peer`. Les fonctions d'homologue sont susceptibles d'être appelées simultanément depuis des threads différents lorsque les descripteurs d'homologue sont distincts. La seule exception à ceci est que la fonction `clemtxt_peer_cancelExecution` (si définie) peut être appelée à partir de tout thread à tout moment pour interrompre une exécution durable sur un thread différent.

Fonction	Obligatoire ?	Description
clemtxt_peer_configure	Oui	Demande à une instance homologue de traiter ses paramètres et son modèle de données
clemtxt_peer_getDataModel	Oui	Obtient le modèle de données de sortie à partir d'une instance homologue
clemtxt_peer_getCatalogueInformation	Non	Obtient les informations de catalogue à partir d'un module
clemtxt_peer_getExecutionRequirements	Non	Obtient les conditions requises d'exécution d'une instance homologue
clemtxt_peer_beginExecution	Oui	Commence l'exécution d'une instance homologue
clemtxt_peer_nextRecord	Oui	Passe à l'enregistrement suivant d'un ensemble de résultats d'homologue
clemtxt_peer_getRecordValue	Oui	Renvoie la valeur d'un champ spécifié dans le dernier enregistrement de résultat récupéré
clemtxt_peer_endExecution	Oui	Indique à une instance homologue que l'exécution est terminée
clemtxt_peer_cancelExecution	Non	Appelée depuis un thread distinct pour interrompre une fonction <code>beginExecution</code> ou <code>nextRecord</code> durable
clemtxt_peer_getSQLGeneration	Non	Génère SQL depuis une instance homologue
clemtxt_peer_getErrorDetail	Oui	Récupère les informations complémentaires sur la dernière erreur propre au module sur un homologue



Les fonctions d'homologue suivantes sont conçues pour être utilisées avec Interactive Model Builder :

Fonction	Obligatoire ?	Description
clemext_peer_beginInteraction	Non	Commence l'interaction avec une instance homologue
clemext_peer_request	Non	Effectue une demande interactive sur un homologue
clemext_peer_getRequestReply	Non	Récupère la réponse de la dernière demande correctement exécutée
clemext_peer_endInteraction	Non	Indique à une instance homologue que l'interaction est terminée

## Fonctions de rappel

Lorsqu'un module d'extension requiert des informations ou un service à partir du processus hôte, il doit le faire via un **rappel**. Ce dernier est appliqué à un **descripteur**, qui est un pointeur qui identifie la cible de la demande.

Les rappels sont invoqués en transmettant l'identificateur de l'objet IBM® SPSS® Modeler vers lequel l'objet est dirigé. Les descripteurs sont intégrés dans un module d'extension comme paramètres dans les fonctions de service.

En cas d'échec de la fonction de rappel, elle doit renvoyer d'autres informations dans le code d'erreur propre au module associé (désigné par **CLEMEXTErrorCode**). Le module peut à son tour gérer ceci en renvoyant une erreur de rappel et en transmettant cette information pour que l'hôte puisse l'inspecter.

Les types suivants de fonctions de rappel sont disponibles :

- Fonctions d'hôte
- Fonctions de noeud
- Fonctions d'itérateur
- Fonctions de progression
- Fonctions de canal (pour les modèles itératifs uniquement)

Les sections suivantes offrent une présentation des fonctions de rappel. D'autres descriptions détaillées de ces fonctions sont disponibles dans la documentation de l'API côté serveur, comme suit :

- ▶ Dans l'écran Documentation de l'API CLEF, choisissez Présentation de l'API côté serveur.
- ▶ Cliquez sur l'onglet Modules.
- ▶ Sélectionnez Rappels généraux.

Pour plus d'informations sur l'accès à la documentation de l'API CLEF, reportez-vous à [CLEF Documentation de l'API sur p. 216](#).

**Fonctions d'hôte**

Les fonctions d'hôte sont définies sur un descripteur d'hôte transmis via `clemext_initialise`.

Fonction	Description
<code>clemext_host_getHostInformation</code>	Obtient des informations statiques sur l'environnement de l'hôte

**Fonctions de noeud**

Les fonctions de noeud sont définies sur un descripteur de noeud transmis via `clemext_create_peer`.

Fonction	Description
<code>clemext_node_getNodeInformation</code>	Obtient des informations statiques sur un noeud
<code>clemext_node_getParameters</code>	Obtient des paramètres depuis un noeud
<code>clemext_node_getDataModel</code>	Obtient le modèle de données d'entrée pour un noeud
<code>clemext_node_getOutputDataModel</code>	Obtient le modèle de données de sortie pour un noeud
<code>clemext_node_getSQLGeneration</code>	Obtient les informations de génération SQL en amont pour un noeud
<code>clemext_node_getPassword</code>	Récupère le texte brut pour un identifiant de mot de passe
<code>clemext_node_getFilePath</code>	Récupère le chemin d'accès d'un fichier échangé entre le client et le serveur lors de l'exécution

**Fonctions d'itérateur**

Les fonctions d'itérateur sont définies sur un descripteur d'itérateur transmis via `clemext_peer_beginExecution`. Un itérateur présente un ensemble de données d'entrée à un module d'extension.

Fonction	Description
<code>clemext_iterator_nextRecord</code>	Passé à l'enregistrement suivant dans l'ensemble de données d'entrée
<code>clemext_iterator_getRecordValue</code>	Renvoie la valeur d'un champ spécifié dans le dernier enregistrement d'entrée récupéré
<code>clemext_iterator_rewind</code>	Restaure l'état de l'ensemble de données d'entrée de sorte que l'appel suivant vers <code>nextRecord</code> passe au premier enregistrement dans l'ensemble

**Fonctions de progression**

Les fonctions de progression sont définies sur un descripteur de progression transmis via `clemext_peer_beginExecution`.

Fonction	Description
<code>clemext_progress_report</code>	Indique la progression à l'hôte

### Fonctions de canal

Les fonctions de canal sont utilisées avec des modèles interactifs uniquement et sont définies sur un descripteur de canal transmis via `clemext_peer_beginInteraction`.

Fonction	Description
<code>clemext_channel_send</code>	Envoie un message asynchrone au client, et permet ainsi à un thread d'homologue d'indiquer les informations d'état et de progression.

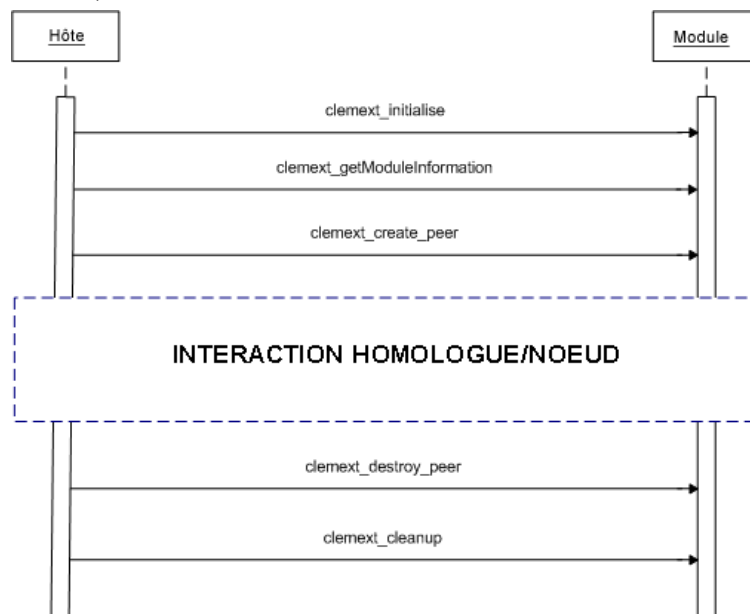
### Flux de processus

Un module d'extension appelle un certain nombre de fonctions de service et de rappel pour effectuer son traitement. Les fonctions réelles appelées dépendent du traitement que le module doit effectuer.

#### Exemple

Le schéma de séquence pour une exécution du module standard est le suivant :

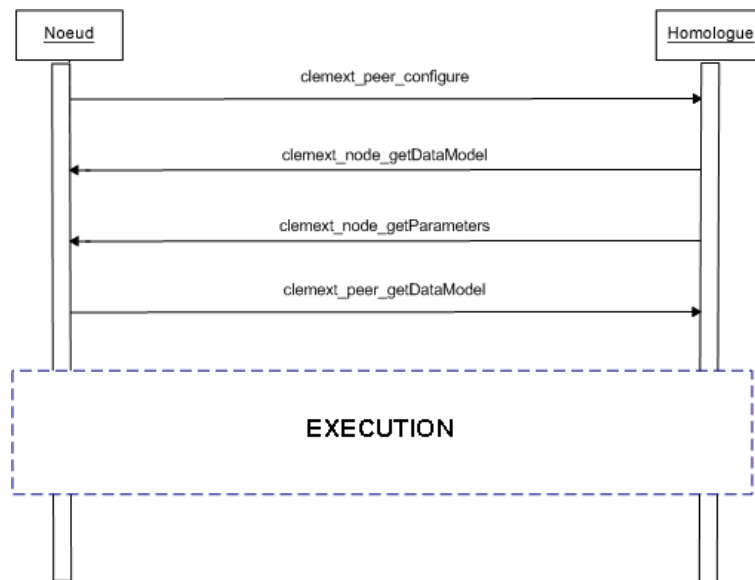
Figure 9-2  
Flux de processus standard



Le bloc d'interaction homologue/noeud ressemble à ceci :

Figure 9-3

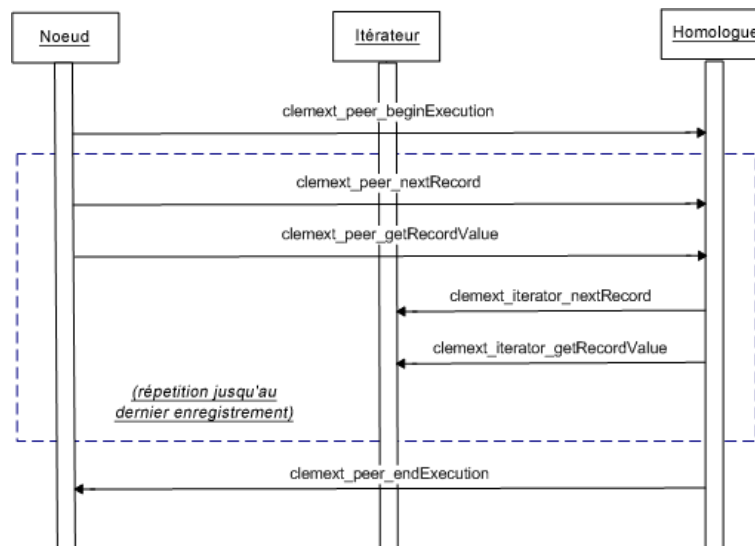
Bloc d'interaction homologue/noeud standard



Un bloc d'exécution standard ressemble à ceci :

Figure 9-4

Bloc d'exécution standard



Remarques :

- Un module est susceptible d'être chargé dans le processus du serveur IBM® SPSS® Modeler au démarrage du serveur. Le chargement peut également avoir lieu ultérieurement sur demande lorsque ses services sont préalablement requis.
- Une fois le module chargé, l'hôte invoque la fonction de service `clemext_initialise` une fois.

- Une fois le module chargé et initialisé, l'hôte peut interroger le module à l'aide de la fonction de service `clmext_getModuleInformation`.
- Une fois le module chargé, ses services sont invoqués via des objets homologues que le module propose. Dans le module, l'objet homologue est créé par la fonction de service `clmext_create_peer` comme équivalent de l'objet noeud de l'hôte pour gérer l'exécution d'une tâche selon les instructions de l'application hôte. Plusieurs objets homologues du même type peuvent exister et être exécutés simultanément dans un processus à un moment donné.
- Une fois l'objet homologue créé, il peut être configuré par la fonction de service `clmext_peer_configure`.
- A ce stade, l'homologue est susceptible d'exécuter des fonctions de rappel pour obtenir des informations du client, telles que `clmext_node_getDataModel` et `clmext_node_getParameters`.
- SPSS Modeler obtient le modèle de données de sortie de l'instance homologue par le biais d'une fonction de service `clmext_peer_getDataModel`.
- L'exécution de l'instance homologue commence par la fonction de service `clmext_peer_beginExecution`.
- La fonction de service `clmext_peer_nextRecord` permet d'activer l'enregistrement suivant dans l'ensemble de résultats de l'homologue (ou le premier enregistrement s'il s'agit du premier appel de la fonction). Ceci est suivi par la fonction de service `clmext_peer_getRecordValue`, qui renvoie la valeur d'un champ spécifié dans l'enregistrement en cours.
- Les fonctions de rappel de l'itérateur `clmext_iterator_nextRecord` et `clmext_iterator_getRecordValue` peuvent être appelées par le module CLEF pour effectuer une séquence dans les enregistrements d'entrée et renvoyer les valeurs de champs indiquées.
- L'exécution de l'instance homologue termine par un appel à la fonction de service `clmext_peer_endExecution`.
- L'instance homologue est supprimée en appelant `clmext_destroy_peer`.
- Avant le déchargement du module, l'hôte invoque la fonction de service `clmext_cleanup`.
- Il est possible de décharger un module à l'arrêt du processus de serveur ou auparavant lorsque ses services ne sont plus requis.

### **Fonctions de l'API côté serveur**

Cette section décrit certaines des fonctions de l'API côté serveur :

- Informations de type noeud
- Types de données représentant différents types de stockage de données
- Bibliothèque partagée côté serveur
- Espaces fichiers et fichiers temporaires
- Répercussions SQL pour exécuter les instructions SQL dans la base de données
- Echange des informations de modèles de données entre IBM® SPSS® Modeler et l'extension
- Documents de sortie
- Fichiers d'aide C++

### Types de noeuds

Dans le fichier de spécifications, une définition de noeud adopte le format suivant :

```
<Node id="identifiant" type="node_type" .../>
```

L'attribut `id` est une chaîne qui identifie de façon unique le noeud.

L'attribut `type` identifie le noeud comme étant l'un des types suivants :

- Lecteur de données
- Rédacteur de données
- Transformateur de données
- Créateur de modèle
- Applicateur de modèle
- Créateur de document

Pour plus d'informations, reportez-vous à la section [Présentation des noeuds dans le chapitre 2 sur p. 9](#).

La fonction `clmext_create_peer` inclut les valeurs des attributs `id` et `type` de l'élément `Node` comme arguments.

Un seul module d'extension est susceptible de mettre en œuvre des noeuds de types différents, effectuant ainsi diverses fonctions dans chaque type. Par exemple, un module peut mettre en œuvre :

- un lecteur de données et un rédacteur de données pour une source de données ;
- des créateurs et des applicateurs de modèles pour divers algorithmes de modélisation ;
- des créateurs de documents pour divers types de graphiques.

### Données et types de stockage

Une instance homologue obtient des données d'entrée en appelant `clmext_iterator_getRecordValue` sur l'itérateur qui lui est fourni au début de l'exécution. En outre, elle propose des données de sortie en réponse à une demande `clmext_peer_getRecordValue` à partir de l'hôte. Les données sont transférées au format binaire, et l'homologue ainsi que l'hôte doivent s'accorder sur le type de données.

Le type de données binaires est déterminé par le modèle de données et est associé à l'attribut de stockage d'un champ.

Le tableau suivant répertorie les types de stockage possibles avec les types de données utilisés pour les représenter :

Table 9-2

Types de stockage

Type de stockage	Représenté par	Remarques
chaîne	char *	Les chaînes de caractère sont toujours codées en UTF-8
réel	CLEMEXTReal	Nombre à virgule flottante en précision double
entier	CLEMEXTInteger	entier signé sur 64 bits

Type de stockage	Représenté par	Remarques
Date	CLEMEXTDate	entier signé sur 64 bits représentant le nombre de jours depuis le 1er janvier 1970
heure	CLEMEXTTime	entier signé sur 64 bits représentant le nombre de secondes depuis minuit
horodatage	CLEMEXTTimestamp	entier signé sur 64 bits représentant le nombre de jours depuis minuit au 1er janvier 1970
inconnu	-	Désigne un type de données inconnu—les valeurs ne peuvent pas être représentées

## Bibliothèques

Une bibliothèque partagée côté serveur peut être déclarée dans le fichier de spécifications pour prendre en charge l'exécution des noeuds. Le chemin d'accès à la bibliothèque partagée permet de rechercher une bibliothèque partagée chargée de façon dynamique dans le processus hôte. La bibliothèque partagée doit définir toutes les fonctions API requises. [Pour plus d'informations, reportez-vous à la section Bibliothèques partagées dans le chapitre 4 sur p. 44.](#)

Si un nom de module est fourni dans le fichier de spécifications (dans la section **Execution** de la définition du noeud), le nom est transmis au paramètre `nodeId` de la fonction de service `clmext_create_peer` pour créer un objet homologue. De cette façon, l'extension peut créer un module homologue de type approprié. La valeur du paramètre `nodeType` est également susceptible d'influencer le type d'homologue créé. Le nom de module peut être vierge car une bibliothèque partagée peut ne pas mettre en œuvre plusieurs modules de chaque type.

Les bibliothèques dépendantes sont susceptibles d'être requises par une bibliothèque partagée qui met en œuvre un module d'extension. Elles doivent figurer dans le même répertoire que la bibliothèque partagée d'extensions.

## Fichiers temporaires

Le fichier de spécifications du client et le module d'extension du serveur peuvent spécifier des noms de chemins d'accès relatifs à un **espace fichier**, qui est un espace privé temporaire où un homologue peut créer des fichiers à utiliser pendant son exécution. Un espace fichier est un sous-répertoire de répertoire temporaire du serveur créé pour un homologue. Il est créé selon les besoins et supprimé lors de la destruction de l'homologue.

L'homologue possède un contrôle total sur l'espace fichier tant qu'il existe. Le nom de chemin d'accès complet de l'espace fichier est inclus dans un **document d'informations sur le noeud**. Il s'agit d'informations au format XML renvoyées à la suite de l'exécution d'une fonction de rappel `clmext_node_getNodeInformation`. [Pour plus d'informations, reportez-vous à la section Document Informations sur le noeud sur p. 236.](#)

## Répercussion SQL

Là où un flux IBM® SPSS® Modeler lit les données depuis une base de données SQL et effectue un traitement sur les données, les utilisateurs avancés peuvent améliorer l'efficacité de cette opération en répercutant les instructions SQL à exécuter dans la base de données elle-même.

Plusieurs noeuds SPSS Modeler standard prennent en charge la répercussion SQL. En outre, l'API côté serveur inclut des appels de fonctions pour rendre également ceci possible pour les noeuds CLEF.

La fonction de service `clemext_peer_getSQLGeneration` génère SQL depuis une instance homologue. En outre, elle permet de répercuter l'exécution de SQL dans la base de données. Pour un noeud de lecteur de données, le SQL généré doit être suffisant pour créer l'ensemble de résultats de l'homologue. Pour tout autre type de noeud, le SQL généré dépendra fort vraisemblablement du SQL généré pour les noeuds en amont qui fournissent les entrées à l'homologue. Un homologue peut obtenir le SQL en amont en appelant la fonction de rappel `clemext_node_getSQLGeneration` sur son descripteur de noeud associé.

### **Gestion des modèles de données**

Certains des appels API côté serveur se rapportent à l'échange des informations sur les modèles de données entre IBM® SPSS® Modeler et le module d'extension :

- `clemext_node_getDataModel` obtient le modèle de données d'entrée pour un noeud
- `clemext_peer_getDataModel` obtient le modèle de données de sortie à partir d'une instance homologue
- `clemext_node_getOutputDataModel` obtient le modèle de données de sortie pour un noeud

Les autres appels se rapportent aux méthodes pour l'échange des données dans le module. Le modèle de données détermine l'index utilisé pour rechercher les valeurs de champs dans les fonctions suivantes, qui renvoient la valeur d'un champ spécifié dans le dernier enregistrement d'entrée récupéré :

- `clemext_peer_getRecordValue`
- `clemext_iterator_getRecordValue`

SPSS Modeler appelle `clemext_node_getDataModel` pour obtenir les informations sur les champs du modèle de données d'entrée. Les informations sont renvoyées au format XML—par exemple :

```
<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>
```

Un module peut utiliser ces informations pour fournir l'index de champ lors de l'extraction des valeurs à partir d'un enregistrement d'entrée à l'aide de la fonction `clemext_iterator_getRecordValue`—par exemple :

**Figure 9-5**  
*Exemple de modèle de données d'entrée*

Index	0	1	2
Champ	abc	uvw	xyz



La manière dont le module affecte le modèle de données d'entrée est contrôlée par la valeur de l'attribut `mode` de l'élément `OutputDataModel` dans le fichier de spécifications. Le module peut :

- étendre le modèle en y ajoutant des champs ;
- modifier le modèle en supprimant ou renommant des champs existants ;
- remplacer le modèle existant par de nouveaux champs ;
- laisser le modèle tel quel.

Les exemples suivants illustrent l'extension et le remplacement du modèle.

### **Exemple—Extension du modèle de données d'entrée**

Il s'agit du cas le plus simple : permettre à un module d'ajouter des champs et de définir leurs valeurs, mais pas de supprimer ou de modifier les valeurs des champs existants.

Supposez que le fichier de spécifications contienne les instructions suivantes dans la définition du noeud :

```
<OutputDataModel mode="extend">
  <AddField name="field1" storage="string" ... />
  <AddField name="field2" storage="real" ... />
  ...
</OutputDataModel>
```

Ici, le modèle de données de sortie est défini comme comprenant tous les champs dans le modèle de données d'entrée ainsi que les deux champs supplémentaires indiqués dans l'élément `OutputDataModel`. Ainsi, le modèle de données de sortie comprend cinq champs.

**Figure 9-6**

*Exemple d'extension du modèle de données d'entrée par le modèle de données de sortie*

Index	0	1	2	3	4
Champ	abc	uvw	xyz	champ1	champ2

La fonction `clmext_peer_getDataModel` renvoie des informations uniquement sur les champs ajoutés, par exemple :

```
<DataModel>
  <Fields>
  <Field name="field1" storage="string" ... />
  <Field name="field2" storage="real" ... />
  <Fields>
</DataModel>
```

Les informations renvoyées doivent correspondre aux valeurs de type et de nombre (mais pas au nom) des éléments `<AddField>` dans le fichier de spécifications.

Un module peut utiliser la fonction de rappel `clmext_node_getOutputDataModel` pour obtenir les détails des champs qui seront ajoutés selon les attentes de IBM® SPSS® Modeler. Ces informations peuvent être réacheminées directement vers SPSS Modeler en réponse à un appel à

`clemt_peer_getDataModel`. Cette opération est utile dans les cas où la logique du fichier de spécifications pour créer et nommer les champs de sortie est complexe.

Le module propose les nouvelles valeurs pour chaque enregistrement de sortie lorsque SPSS Modeler appelle `clemt_peer_getRecordValue`. Les index de champ pour les nouveaux champs commencent après le dernier index des champs d'entrée. Dans cet exemple, le modèle de données d'entrée contient trois champs (aux positions d'index 0, 1 et 2). Par conséquent, les index de champ 3 et 4 sont affectés aux deux champs de sortie. SPSS Modeler n'appelle pas `clemt_peer_getRecordValue` avec des index de champ correspondant aux champs d'entrée car le module ne peut pas modifier ces champs.

### **Exemple—Remplacement du modèle de données d'entrée (1)**

Dans cet exemple, le module d'extension exclut tous les champs du modèle de données d'entrée de sa sortie, les remplaçant par de nouveaux champs.

Le fichier de spécifications contient les éléments suivants :

```
<OutputDataModel mode="modify">
  <AddField name="key" storage="integer" ... />
  <AddField name="field1" storage="real" ... />
  <AddField name="field2" storage="real" ... />
  ...
</OutputDataModel>
```

Cette fois-ci, les données XML renvoyées par un appel vers `clemt_peer_getDataModel` décrivent tous les champs dans le modèle de données de sortie :

```
<DataModel>
  <Fields>
  <Field name="key" storage="integer" ... />
    <Field name="field1" storage="real" ... />
    <Field name="field2" storage="real" ... />
  </Fields>
</DataModel>
```

Le modèle de données de sortie présente la structure suivante :

**Figure 9-7**

*Exemple de remplacement du modèle de données d'entrée par le modèle de données de sortie*

Index	0	1	2
Champ	key	champ1	champ2

Les index de champ utilisés dans les appels à `clemt_peer_getRecordValue` commencent à 0 pour le premier champ de sortie (`key`), 1 pour le champ suivant (`field1`), etc.

### Exemple—Remplacement du modèle de données d'entrée (2)

Dans cet exemple, le modèle de données de sortie fourni par l'extension remplace également le modèle de données d'entrée, comme dans l'exemple précédent. Dans ce cas, cependant, le modèle de données de sortie n'est pas défini dans le fichier de spécifications mais est plutôt calculé au moment de l'exécution par le module d'extension sur le serveur. Le fichier de spécifications contient les éléments suivants :

```
<OutputDataModel mode="modify" method="sharedLibrary" libraryId="myLibraryId" />
```

Pour calculer le modèle de données de sortie, IBM® SPSS® Modeler appelle d'abord `clemext_peer_configure`, puis `clemext_peer_getDataModel`. Comme dans l'exemple précédent, aucun des champs dans le modèle de données d'entrée n'est automatiquement inclus dans le modèle de données de sortie, qui est entièrement défini par la réponse de `clemext_peer_getDataModel`.

*Remarque* : dans de tels cas, où le module d'extension définit le modèle de données de sortie sur le serveur, le module ne peut pas utiliser `clemext_node_getOutputDataModel` pour obtenir le modèle de données de sortie car ceci entraînerait une erreur « opération non valide ».

### Documents de sortie XML

Certaines fonctions de service et de rappel transfèrent les informations entre l'hôte et le module d'extension sous la forme de documents de sortie XML. Un certain nombre de documents différents sont disponibles :

document de sortie XML	Remarques	Renvoyé par appel vers...
Document Catalogue	Contient la liste des valeurs d'une commande associée à un catalogue.	<code>clemext_peer_getCatalogInformation</code>
Document Modèle de données	Décrit l'ensemble des champs entrant et sortant d'un noeud	<code>clemext_peer_getDataModel</code> <code>clemext_node_getDataModel</code> <code>clemext_node_getOutputDataModel</code>
Document des détails de l'erreur	Offre des informations sur une erreur ou une autre condition	<code>clemext_peer_getErrorDetail</code>
Document des conditions requises pour l'exécution	Décrit le support d'exécution, tel que le cache de données ou les champs d'entrée obligatoires, requis par une instance homologue	<code>clemext_peer_getExecutionRequirements</code>
Document d'informations sur l'hôte	Offre des informations sur l'environnement de l'hôte, dont : identifiant du produit, description, version, fournisseur, copyright et détails de la plate-forme	<code>clemext_host_getHostInformation</code>

document de sortie XML	Remarques	Renvoyé par appel vers...
Document d'informations sur le module	Offre des informations sur le module d'extension, dont : identifiant du module, description, version, fournisseur, copyright et détails de la licence	clemext_getModuleInformation
Document d'informations sur le noeud	Offre des informations sur le noeud associé à une instance homologue, dont : identifiant de noeud, type et détails de l'espace fichier	clemext_node_getNodeInformation
Document des paramètres	Contient les paramètres de configuration du noeud client ; le contenu est déterminé par l'extension	clemext_node_getParameters
Document Génération de SQL	Décrit le mode de conversion de l'exécution d'un homologue en SQL	clemext_peer_getSQLGeneration clemext_node_getSQLGeneration
Document Détails de l'état	Offre des informations complémentaires sur la progression et les avertissements ou les autres conditions survenant lors de l'exécution	clemext_progress_report

### **Document Catalogue**

Un document Catalogue décrit le contenu d'un catalogue qui contient une liste des valeurs pouvant être affichées à partir d'un contrôle IU.

Le module CLEF exécute un appel à `getCatalogInformation` comme suit :

```

CLEMEXTStatus
getCatalogInformation(
    const char *catalogId,
    char* buffer,
    size_t buffer_size,
    size_t* data_size,
    CLEMEXTErrorCode* errorCode) {
    ...
}

```

où `catalogId` est l'identificateur d'un catalogue spécifique, comme défini dans l'élément `Catalog` du fichier de spécifications.

Cette fonction renvoie le document Catalogue.

#### **Exemple**

```

<?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>

```

### **Document Modèle de données**

Un document Modèle de données décrit le **modèle de données**—l'ensemble des champs avec leurs noms, types et informations connexes entrant et sortant d'un noeud. Il encapsule les informations disponibles dans un noeud Typer.

Un homologue qui n'utilise pas d'entrée (un noeud source) comporte un modèle d'entrée vide. En outre, celui qui ne génère pas de sortie (noeud terminal) comprend un modèle de sortie vide. Un homologue qui utilise une entrée et génère une sortie (noeud de processus) doit savoir calculer son modèle de sortie à partir de son entrée.

Un homologue peut obtenir son modèle de données d'entrée en appelant `clmext_node_getDataModel` sur l'indicateur de noeud associé. Un homologue fournit son modèle de données de sortie en réponse à une demande `clmext_peer_getDataModel` de son hôte.

Tout modèle de données peut être exprimé directement comme dictionnaire de données qui énumère tous les champs dans le modèle de données avec leurs propriétés. Un modèle de données d'entrée fourni par un noeud à un homologue est toujours de ce format. Un modèle de données de sortie généré par un homologue est susceptible de posséder le même format ou d'être exprimé comme une séquence d'opérations (ajouter un champ, supprimer un champ, modifier un champ) appliquée au modèle d'entrée. Ceci simplifie considérablement le modèle de sortie pour certains noeuds.

L'ordre des champs dans le document Modèle de données est important. En outre, il détermine la séquence dans laquelle les données sont présentées dans l'ensemble de données d'entrée ou de sortie correspondant.

Un modèle de données est susceptible d'être incomplet et n'offre qu'une spécification partielle des données. Un modèle suffisamment précisé pour permettre à un homologue de calculer un plan d'exécution est appelé **exécutable** pour cet homologue. Un modèle de données exécutable doit toujours contenir le type binaire pour chaque champ de sorte que les données d'entrée et de sortie puissent être correctement triées.

### **Exemple**

```

<?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>

```

### ***Document Détails de l'erreur***

Un document Détails de l'erreur permet de renvoyer des messages (erreur, avertissement, informations) à IBM® SPSS® Modeler et fournit des informations sur une erreur ou une autre condition. Un module d'extension peut fournir un document Détails de l'erreur pour expliquer une erreur propre au module en réponse à une demande `clemext_peer_getErrorDetail` de l'hôte.

Les détails de l'erreur sont composés d'un ou de plusieurs éléments Diagnostic où chaque diagnostic comprend au moins un code d'erreur, un message, et un ensemble d'un ou de plusieurs paramètres contenant d'autres informations à insérer dans le message. Les codes d'erreur correspondent aux valeurs dans les éléments `StatusCode` dans le fichier de spécifications.

Le message est susceptible de posséder différentes variantes linguistiques. Le client peut également utiliser le code d'erreur pour sélectionner un message localisé à partir d'un groupe de ressources. Une séquence d'éléments de diagnostic décrit une chaîne causale d'erreurs.

### **Exemple**

```
<?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>
```

### **Document Conditions requises d'exécution**

Un document Conditions requises d'exécution décrit le support d'exécution requis par une instance homologue. Une instance homologue peut offrir un document Conditions requises d'exécution en réponse à une requête `clemtxt_peer_getExecutionRequirements` de l'hôte. L'hôte consulte le document des conditions requises avant d'appeler `clemtxt_peer_beginExecution` sur l'homologue afin de fournir l'environnement d'exécution approprié.

L'hôte peut fournir un service cache de données pour permettre au module d'effectuer plusieurs passes sur les données d'entrée à l'aide de la fonction `clemtxt_iterator_rewind`.

### **Exemple**

```
<?xml version="1.0" encoding="utf-8"?>
<ExecutionRequirements>
  <Cache/><!-- Ceci garantit que le module CLEFpuisse effectuer plusieurs passes sur les données d'entrée -->
</ExecutionRequirements>
```

### **Document Informations sur l'hôte**

Un document Informations sur l'hôte décrit l'environnement de l'hôte. Un module d'extension peut obtenir des informations sur l'hôte en appelant `clemtxt_host_getHostInformation` sur l'indicateur de l'hôte.

Les informations renvoyées comportent les détails de l'identificateur de produit, la description, la version, le fournisseur, les copyright et la plate-forme.

### **Exemple**

```
<?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="14.2" 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\14.2\ext\bin\my.module\myModule.dll"/>
</Host>
</HostInformation>

```

### **Document Informations sur le module**

Un document Informations sur le module décrit un module d'extension. Un module d'extension doit fournir un document Informations sur le module en réponse à une requête `clmext_getModuleInformation` de l'hôte.

Les informations renvoyées comportent les détails de l'identificateur de module, la description, la version, le fournisseur, les copyright et la licence.

#### **Exemple**

```

<?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>Offre un test complet de la nouvelle structure d'extensions.</Description>
  </Module>
</ModuleInformation>

```

### **Document Informations sur le noeud**

Un document Informations sur le noeud décrit le noeud associé à une instance homologue. Cette dernière peut obtenir des informations sur le noeud en appelant `clmext_node_getNodeInformation` sur un indicateur de noeud. Les informations sur le noeud comportent des détails de l'identificateur de noeud, de type et de l'espace fichier.

#### **Exemple**

```

<?xml version="1.0" encoding="utf-8"?>
<NodeInformation>
  <Node name="databaseImport" type="dataReader">
    <FileSpace path="C:\Program Files\IBM SPSS Modeler Server14.2\tmp\ext-8005-6711-01"/>
  </Node>
</NodeInformation>

```

### **Document Paramètres**

Un document Paramètres contient les détails de chaque élément Property défini dans le fichier de spécifications. Les détails sont renvoyés sous la forme de paramètres de configuration, qu'un homologue peut obtenir en appelant `clmext_node_getParameters` sur l'indicateur d'un noeud.



Un paramètre comporte un nom et une valeur, où la valeur peut être :

- Une valeur simple (chaîne)
- Une valeur de clé (clé et valeur).
- Une valeur structurée (ensemble de valeurs nommées)
- Liste des valeurs

Le contenu du document Paramètres est entièrement déterminé par le module d'extension. Les paramètres sont définis dans le fichier de spécifications du client et sont interprétés par le module d'extension du serveur.

### **Exemple**

```
<?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>
```

### **Document Génération de SQL**

Un document Génération de SQL décrit le mode de conversion de l'exécution d'un homologue en SQL.

Un homologue peut offrir un document Génération de SQL en réponse à une requête `clemext_peer_getSQLGeneration` de l'hôte. L'hôte tente d'exécuter le SQL plutôt que l'homologue au niveau interne.

Un homologue, qui utilise une entrée, peut obtenir son SQL d'entrée en appelant `clemext_node_getSQLGeneration` sur l'indicateur de noeud associé.

Le composant principal d'un document Génération de SQL est une instruction SQL qui réplique le comportement d'exécution d'un fragment de noeud ou de flux. Pour un noeud qui génère des données (c'est à dire, un noeud lecteur de données ou transformateur de données), l'instruction doit être `SELECT`. En outre, elle doit être accompagnée d'un dictionnaire qui mappe les noms de champs dans le modèle de données aux noms de la colonne dans l'instruction `SELECT`.

Un document Génération de SQL est également susceptible d'inclure les propriétés de connexion de base de données avec laquelle l'instruction est exécutée, comme le nom de la source de données et du produit. Un homologue utilise ces propriétés pour aider à déterminer le SQL qu'il génère.

**Exemple**

```

<?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>

```

**Document Détails de l'état**

Un document Détails de l'état offre des informations sur la progression et les avertissements non fatals ou les autres conditions survenant lors de l'exécution. Un module d'extension peut envoyer les documents Détails de l'état de façon asynchrone à l'aide de la fonction de rappel `clmext_progress_report`.

Le document Détails de l'état se compose d'un ensemble d'un ou de plusieurs éléments **Diagnostic** où chaque diagnostic comprend au moins un code de condition, un message (sauf s'il se trouve dans un fichier de propriétés), et un ensemble d'un ou de plusieurs paramètres contenant d'autres informations à insérer dans le message. L'élément **StatusDetail** peut également avoir un attribut facultatif `destination` qui permet de transmettre le message à un des éléments suivants :

- Un fichier de trace local géré par IBM® SPSS® Modeler
- Le client (pour les messages d'intérêt pour l'utilisateur)
- Toutes (envoi à toutes les destinations possibles)

Le format de l'élément **Diagnostic** est :

```

<Diagnostic code="entier" severity="severity_level">
  <Message>message_text</Message>
  <Parameter>value</Parameter>
</Diagnostic>

```

où :

code (obligatoire) est un entier indiquant le code de condition.

severity indique la sévérité d'une condition et est soit : unknown, information, warning, error ou fatal.

### **Exemple**

```
<?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>
```

### **Utilisation des messages localisés**

Pour utiliser les messages localisés d'un fichier de propriétés, vous devez ignorer l'élément Message du document Détails de l'état et utiliser les clés de message du fichier de spécifications, comme dans l'exemple suivant :

```
...
<Execution ...>
...
  <StatusCodes>
    ...
    <StatusCode code="21" status="warning" messageKey="fieldIgnoredMsg.LABEL"/>
    ...
  </StatusCodes>
</Execution>
...
```

Le fichier messages.properties contiendrait alors :

**fieldIgnoredMsg.LABEL**=Field "{0}" cannot be used for model building and was ignored

Dans le document Détails de l'état, vous pourrez ensuite envoyer un paramètre (le nom d'un champ par exemple) pour qu'il soit utilisé dans le message localisé, par exemple :

```
<?xml version="1.0" encoding="utf-8"?>
<StatusDetail>
  <Diagnostic code="21">
    <Parameter>BP</Parameter>
  </Diagnostic>
</StatusDetail>
```

### Fichiers d'aide C++

Certains des exemples de noeuds CLEF comprennent un certain nombre de fichiers source C++ prédéfinis, appelés **fichiers d'aide**. Ils font office de wrappers pour certaines des API côté serveur à base C. En outre, ils peuvent facilement être compilés en CLEF C++.

Table 9-3  
Fichiers d'aide C++

Fichiers d'aide	Description
BufferHelper	Gère les tampons mémoire redimensionnables utilisés dans l'API C
DataHelper	Aide à encapsuler les opérations de lecture et d'écriture de données
HostHelper	Encapsule l'objet HostHandle de CLEMEXT
XMLHelper	Encapsule le traitement de XML

Les fichiers d'aide prennent la forme de deux fichiers *.cpp* et *.h* —par exemple, *BufferHelper.cpp* et *BufferHelper.h*.

Pour plus d'informations sur l'affichage de ces fichiers d'aide, reportez-vous à [Analyse du code source sur p. 36](#).

D'autres descriptions détaillées de ces fichiers sont disponibles dans la documentation de l'API côté serveur, comme suit :

- ▶ Dans l'écran Documentation de l'API CLEF, choisissez Présentation de l'API côté serveur.
- ▶ Cliquez sur l'onglet Fichiers.
- ▶ Cliquez sur le nom du fichier *.h* correspondant au fichier d'aide pour lequel vous souhaitez obtenir des informations.
- ▶ Dans Structures de données, cliquez sur le nom de classe correspondant pour afficher la documentation.

Pour plus d'informations sur l'accès à la documentation de l'API CLEF, reportez-vous à [CLEF Documentation de l'API sur p. 216](#).

### Gestion des erreurs

Chaque appel de fonction API renvoie un code d'état (CLEMEXTStatus) et un code d'erreur propre au module en option (CLEMEXTErrorCode). Le code d'état peut être *success* (pas d'erreur) ou l'un des codes d'erreur énumérés pour la fonction API ; ceux-ci comportent presque toujours « erreur propre au module ». Le code d'erreur propre au module peut être 0 pour indiquer « pas d'erreur propre au module ».

Les messages de code d'état sont fournis par IBM® SPSS® Modeler. Plus d'informations sur les codes d'état standard sont disponibles dans la documentation de l'API côté serveur, comme suit :

- ▶ Dans l'écran Documentation de l'API CLEF, choisissez Présentation de l'API côté serveur.
- ▶ Cliquez sur l'onglet Modules.

- ▶ Sélectionnez Codes d'état standard.

Pour plus d'informations sur l'accès à la documentation de l'API CLEF, reportez-vous à [CLEF Documentation de l'API sur p. 216](#).

Les messages d'erreur propres au module peuvent être fournis :

- Dans le fichier de spécifications (dans l'élément `StatusCodes` de la section Module)
- Dans un groupe de ressources référencé dans le fichier de spécifications
- Par le module d'extension

Pour un code d'erreur propre au module, ce dernier peut fournir des détails supplémentaires sur l'erreur comportant un message d'erreur par défaut (pour les erreurs non expliquées dans le fichier de spécifications ou dans le groupe de ressources) ainsi que les paramètres à insérer dans le message. Plusieurs messages d'erreur peuvent décrire des chaînes d'erreurs causales.

Un rapport d'erreur sur le client possède le format suivant :

*node\_label:message*

où :

- *node\_label* est la valeur de l'attribut `label` de l'élément `Node` dans lequel le module est spécifié.
- *message* est le texte du message, qui peut être fourni par le serveur ou défini dans le fichier de spécifications (ou dans un fichier *.properties* pour la localisation).

## **API d'analyse XML**

IBM® SPSS® Modeler Inclut l'analyseur Xerces-C XML d'Apache, et offre ainsi un certain nombre de rappels permettant à un module de lire et d'écrire des données XML. Vous pouvez remplacer votre propre analyseur XML, si vous le souhaitez.

### **Utilisation de l'API côté serveur**

Pour inclure les appels de fonctions côté serveur dans un noeud :

- ▶ Créez les fichiers source C++ *.cpp* et *.h* comportant les appels de fonction.
- ▶ Compilez les fichiers source dans un fichier de bibliothèque de liens dynamiques (*.dll*).
- ▶ Ajoutez une référence au fichier *.dll* à partir du fichier de spécifications—par exemple :

```
<Resources>
.
  <SharedLibrary id="mylib1" path="mycorp.mynode/mylib" />
.
</Resources>
```

Pour plus d'informations, reportez-vous à la section [Bibliothèques partagées](#) dans le chapitre 4 sur p. 44.

Il pourrait également vous être utile de consulter le code source pour les exemples de noeuds fournis avec cette version. [Pour plus d'informations, reportez-vous à la section Analyse du code source dans le chapitre 3 sur p. 36.](#)

### ***Conseils de programmation côté serveur***

La DLL (bibliothèque de liens dynamiques) côté serveur qui fait partie d'un module CLEF doit suivre un certain nombre de conseils pour assurer un bon fonctionnement du module et éviter de compromettre le fonctionnement de IBM® SPSS® Modeler. Un module CLEF doit :

- permettre une exécution des homologues indépendante
- prendre en charge diverses instances d'homologue en un seul processus
- garantir la sécurité des threads
- éviter la modification des environnements de thread ou de processus
- restreindre l'utilisation des threads dans le module
- gérer correctement les requêtes d'annulation de l'exécution
- redémarrer les appels système interrompus (UNIX)
- s'occuper des appels à CoInitialize ou CoUninitialize (Windows)
- éviter les suppositions quant à l'heure de déchargement du module
- s'occuper du démarrage des sous-processus
- éviter d'écrire sur la sortie standard ou sur l'erreur standard

Les sections suivantes détaillent chacun de ces domaines.

#### ***Permettre une exécution des homologues indépendante***

Les instances d'homologue ne devraient pas émettre d'hypothèses quant à l'existence d'instances d'homologue dans le processus du serveur IBM® SPSS® Modeler. SPSS Modeler peut programmer l'exécution pour que les instances d'homologue correspondant à des noeuds qui sont directement adjacents dans un flux soient bien exécutées au cours de phases différentes afin que l'existence et l'exécution de ces instances ne se chevauchent pas.

Par conséquent, les instances d'homologue doivent être indépendantes et ne doivent pas essayer de communiquer directement avec d'autres instances d'homologue, par exemple par des conduits ou des sockets. Toutes les communications entre différentes instances d'homologue doivent être effectuées soit directement, en lisant ou en écrivant des données dans le flux, soit indirectement, en passant par un agent externe (par exemple, un serveur de base de données qui gère les données que se partagent les homologues).

#### ***Prendre en charge diverses instances d'homologue en un seul processus***

Les utilisateurs finaux peuvent créer diverses instances d'homologue d'un module CLEF spécifique (c'est-à-dire plusieurs noeuds du même type) dans un processus de serveur lors de l'exécution d'un flux. Ainsi, toutes les données statiques dans le module CLEF sont partagées entre les diverses instances d'homologue et ne devront pas être utilisées pour stocker des données

spécifiques à un objet homologue. Des exemples de données statiques sont les membres statiques des classes C++ et les variables globales ou statiques dans les unités de compilation C.

Les fonctions API du module CLEF doivent être réentrantes et doivent éviter tout appel système non réentrant. Par exemple, lorsqu'une instance d'homologue extrait des données d'entrée de son itérateur d'entrée à l'aide de `clemext_iterator_nextRecord`, il est ensuite possible d'appeler `clemext_peer_nextRecord` lors d'une seconde instance d'homologue située en amont de la première instance d'homologue et qui produit les données qui seront finalement utilisées par le premier homologue.

Les appels système comme `strtok` ne sont pas réentrants et ne doivent pas être utilisés. Pour des détails sur d'autres appels réentrants, consultez la documentation sur le système d'exploitation de la plateforme utilisée.

### ***Garantir la sécurité des threads***

IBM® SPSS® Modeler peut entrelacer l'exécution de diverses instances d'homologue à partir de différents threads d'exécution. Par conséquent, l'accès aux ressources partagées entre les objets homologues doit être protégé, par exemple par une synchronisation avec les mutex (objets d'exclusion mutuelle) ou avec des services de bibliothèques de threads similaires.

Les modules CLEF doivent éviter tout appel système sans thread sécurisé. Pour des informations supplémentaires, consultez la documentation de votre système d'exploitation ou les pages man d'UNIX.

### ***Eviter la modification des environnements de thread ou de processus***

Evitez d'utiliser des appels système qui pourraient modifier l'environnement du thread ou du processus appelant.

Des exemples de ces appels apparaissent dans cette liste non exhaustive :

- `setlocale` lorsqu'il est utilisé pour modifier un paramètre régional plutôt que de lire les informations relatives aux paramètres régionaux.
- `SetCurrentDirectory` (Windows) ou `chdir` (UNIX)
- `LogonUser` (Windows) ou `seteuid` (UNIX)
- `putenv`
- `exit`
- `signal`

*Remarque* : sous Windows, un appel qui modifie l'environnement d'un thread mais qui peut être nécessaire est `CoInitialize`. [Pour plus d'informations, reportez-vous à la section S'occuper des appels à `CoInitialize` ou `CoUninitialize` \(Windows\) sur p. 244.](#)

### ***Restreindre l'utilisation des threads dans le module***

Généralement, les modules peuvent utiliser des threads à l'interne. Mais les fonctions de rappel doivent être invoquées uniquement sur le thread que IBM® SPSS® Modeler utilise pour appeler les fonctions du module CLEF (à l'exception de `clemext_peer_cancelExecution`).

Les fonctions de rappel suivantes peuvent être appelées de façon asynchrone à partir de tout thread en exécution dans le module :

- `clemext_progress_report`
- `clemext_channel_send`

Une instance d'homologue doit garantir que plusieurs threads n'invoquent pas chacun de ces appels en même temps.

### ***Gérer correctement les requêtes d'annulation de l'exécution***

Lorsqu'un utilisateur final demande l'annulation de l'exécution d'une instance d'homologue, IBM® SPSS® Modeler effectue un appel asynchrone à la fonction `clemext_peer_cancelExecution` du module. Les développeurs devraient essayer d'implémenter cet appel. Veuillez noter que cette fonction doit être appelée de façon asynchrone et pendant qu'une autre fonction API CLEF est exécutée par le module.

### ***Redémarrer les appels système interrompus (UNIX)***

Sous UNIX, l'application IBM® SPSS® Modeler utilise des signaux et des gestionnaires de signaux. Certains appels système UNIX peuvent renvoyer un code EINTR si le processus reçoit un signal pendant l'exécution de l'appel. Vérifiez les pages man pour l'appel système de votre plateforme UNIX.

Si cet évènement se produit, le code d'appel doit vérifier le code EINTR renvoyé et recommencer l'appel. Pour cela, il est possible de créer une simple fonction de wrapping (`open_safe`) pour que votre application appelle ce wrapping :

```
int
open_safe(const char* path, int oflag, mode_t mode) {
    int res;
    while ((res = ::open(path, oflag, mode)) == -1
           && errno == EINTR) {
    }
    return res;
}
```

### ***S'occuper des appels à Colnitalize ou CoUnitalize (Windows)***

Sous Windows, les threads qui ont besoin d'utiliser les services de bibliothèque du modèle d'objet du composant Windows (COM) doivent appeler la fonction `Colnitalize` de l'API du système avant d'utiliser les services COM et doivent ensuite appeler la fonction `CoUnitalize`. Les threads à partir desquels IBM® SPSS® Modeler invoque l'API de CLEF pour un module peuvent avoir ou non déjà appelé `Colnitalize`.

Un module CLEF qui souhaite utiliser les services COM de ces threads doit appeler `Colnitalize`, généralement dans une fonction `clemext_create_peer` ou une fonction `clemext_peer_beginExecution`. Si cet appel réussit, le module doit également appeler



CoUninitialize ultérieurement lorsque le thread aura été exécuté, généralement dans `clmext_destroy_peer` ou `clmext_peer_endExecution` respectivement.

Pour des informations supplémentaires sur l'appel à `Colnitiaize`, consultez la documentation sur le site Microsoft Developer Network (MSDN) à l'adresse : <http://msdn.microsoft.com>.

### ***Éviter les suppositions quant à l'heure de déchargement du module***

Actuellement, un module CLEF reste chargé jusqu'à la fin de la session (c'est-à-dire que les modules ne peuvent pas être déchargés et rechargés à la demande). La fonction `clmext_cleanup` n'est jamais appelée même après avoir quitté le processus du serveur IBM® SPSS® Modeler dans lequel le module est chargé. Par conséquent, les développeurs ne doivent jamais supposer qu'un module sera déchargé et ses ressources libérées, à quelque moment que ce soit.

### ***S'occuper du démarrage des sous-processus***

Démarrer un sous-processus, avec `CreateProcess` (Windows) ou `fork` (UNIX), peut amener un certain nombre de complications au niveau de l'interaction des processus parent et enfant, notamment lorsque le processus enfant hérite de ressources ouvertes dans le processus parent.

Si un module CLEF doit invoquer une exécution hors-processus, pensez à utiliser une autre architecture appropriée. Par exemple, le module CLEF peut utiliser les services proposés par un serveur d'application pour exécuter la tâche requise.

Plus précisément, les processus Windows devraient éviter de démarrer des sous-processus avec la fonction `CreateProcess` et le paramètre `blnheritHandles` défini sur `TRUE`. Le processus enfant hériterait alors de tous les descripteurs de fichiers ouverts dans le processus parent (serveur IBM® SPSS® Modeler).

### ***Éviter d'écrire sur la sortie standard ou sur l'erreur standard***

Si un module CLEF écrit sur le flux de sortie standard ou d'erreur standard d'un processus (peut-être dans le but d'un débogage), cela ne sera généralement pas visible pour l'utilisateur final. Mais lorsque qu'un flux contenant des noeuds CLEF est déployé avec IBM® SPSS® Modeler Solution Publisher et est exécuté à partir d'un shell de ligne de commande (que ce soit sous Windows ou UNIX), cette sortie sera visible et peut embrouiller les utilisateurs.

Au contraire, les modules CLEF peuvent invoquer un service de traçage en appelant la fonction de rappel hôte `clmext_host_trace` et en transmettant le message pour qu'il s'affiche au format chaîne. Le traçage doit être activé dans l'installation IBM® SPSS® Modeler avec le paramètre suivant dans le fichier des options de configuration du IBM® SPSS® Modeler Server (`/config/options.cfg` dans le répertoire d'installation SPSS Modeler) :

```
trace_extension, 1
```

Les messages tracés sont écrits dans le fichier `/log/trace-<process_ID>-<process_ID>.log` dans le répertoire d'installation de SPSS Modeler, où `process_ID` représente l'identifiant du processus SPSS Modeler Server. Évitez le traçage de plusieurs sessions en même temps car elles partagent toutes le même fichier journal.

# Test et distribution

## Test des extensions CLEF

Il est vivement recommandé de tester une nouvelle extension localement avant de la distribuer à d'autres utilisateurs.

Après la création d'un fichier de spécifications et de tout autre groupe de ressources associées, fichiers .jar, bibliothèques partagées et fichiers d'aide utilisateur, vous pouvez tester l'extension en organisant les fichiers dans la structure de fichiers requise et en les copiant vers votre installation locale IBM® SPSS® Modeler. La prochaine fois que vous démarrerez SPSS Modeler, la nouvelle extension sera disponible dans l'interface utilisateur de SPSS Modeler.

## Test d'une extension CLEF

- ▶ Fermez IBM® SPSS® Modeler si l'application est ouverte.
- ▶ Si l'extension définit un noeud ou une sortie CLEF, nous vous recommandons d'activer l'onglet Débuguer de la boîte de dialogue de l'extension jusqu'à ce que l'extension fonctionne correctement. [Pour plus d'informations, reportez-vous à la section Utilisation de l'onglet Débuguer sur p. 248.](#)
- ▶ Organisez les fichiers côté-client et côté-serveur dans la structure requise. Assurez-vous que le fichier de spécification et que toutes les ressources associées dont le noeud a besoin (telles que les fichiers .jar ou .dll) sont copiés aux bons emplacements. [Pour plus d'informations, reportez-vous à la section Structure de fichier dans le chapitre 1 sur p. 6.](#)
- ▶ Copiez le répertoire côté-client dans le dossier `\ext\lib` sous le répertoire d'installation de SPSS Modeler.
- ▶ Copiez le répertoire côté-serveur dans le dossier `\ext\bin` sous le répertoire d'installation de SPSS Modeler.
- ▶ Démarrer SPSS Modeler.
- ▶ Si l'extension définit un menu ou un élément de menu, assurez-vous que celui-ci apparaît correctement dans le menu principal. Si l'extension définit un nouveau noeud, assurez-vous que le noeud apparaît à la position souhaitée dans la palette des noeuds appropriée tel que défini dans le fichier de spécifications.
- ▶ Testez l'extension intégralement.

Assurez-vous, par exemple, que :

- les performances du noeud ne se dégradent pas avec l'augmentation du nombre de champs et d'enregistrements

- les valeurs nulles sont traitées de façon cohérente
- les différents paramètres régionaux sont pris en charge si nécessaire (par exemple Europe, Extrême-Orient)
- ▶ Une fois qu'une extension a été ajoutée, vous pouvez encore apporter des modifications à son fichier de spécifications. Toutefois, les modifications effectuées ne prennent effet qu'une fois SPSS Modeler redémarré.

## Débogage d'une extension CLEF

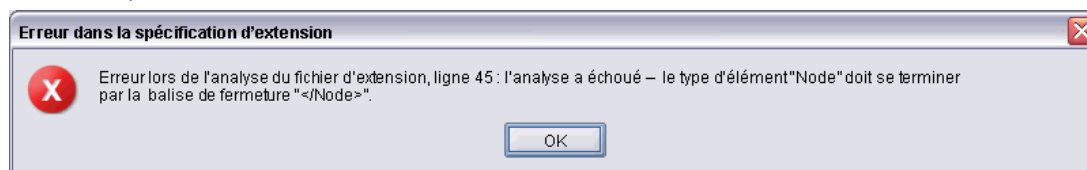
CLEF offre les fonctionnalités suivantes pour aider au débogage d'une extension :

- messages d'erreur de syntaxe XML
- exécution externe
- Onglet Déboguer

### Erreurs de syntaxe XML

Une syntaxe XML incorrecte dans un fichier de spécifications est signalée par un message d'erreur de l'analyseur XML, par exemple :

Figure 10-1  
Erreur de syntaxe XML



Le message affiche le numéro de ligne approximatif de l'erreur, avec une indication de ce qui est incorrect.

Pour résoudre le problème :

- ▶ Corrigez l'erreur dans le fichier.
- ▶ Testez de nouveau le fichier en suivant la procédure suivante [Test d'une extension CLEF sur p. 246](#).
- ▶ Répétez cette procédure jusqu'à ce qu'il n'y ait plus aucune erreur de syntaxe dans le fichier de spécifications.

### Exécution externe

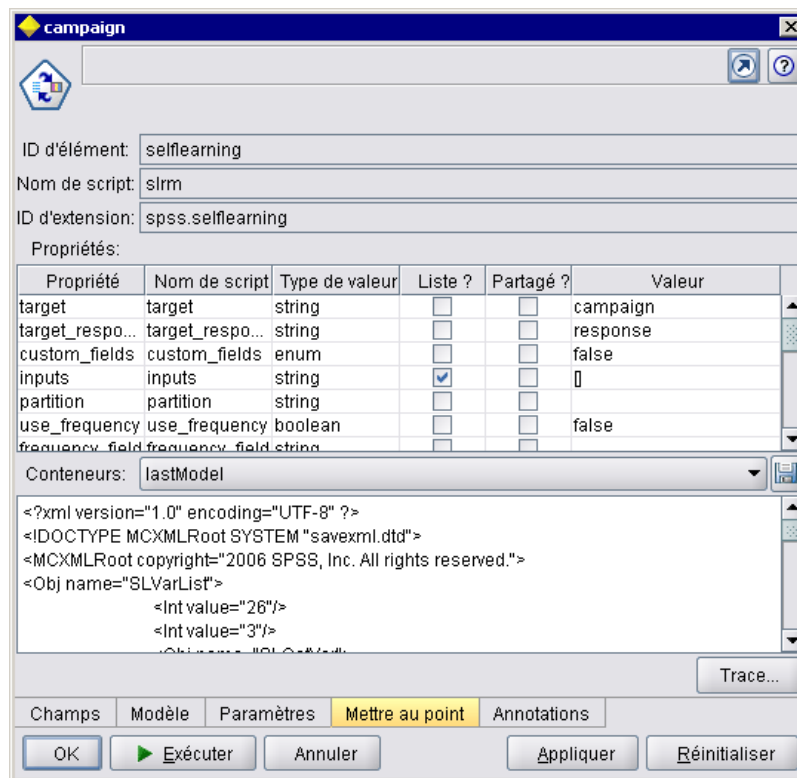
Normalement, une extension CLEF écrite par l'utilisateur s'exécute dans son propre processus, distinct du processus IBM® SPSS® Modeler. Cela peut aider lors du débogage, en cas d'échec d'un processus d'extension, cela ne fait pas tomber le processus IBM® SPSS® Modeler Server dans son entier.

*Remarque* : Il est possible d'ignorer cette option par défaut. [Pour plus d'informations, reportez-vous à la section Modification des options d'exécution sur p. 249.](#)

### Utilisation de l'onglet Déboguer

Pour toute boîte de dialogue ou cadre associé à un noeud ou à une sortie CLEF, vous pouvez activer un onglet Déboguer pour vous permettre d'examiner les paramètres d'une propriété pour l'objet. Vous pouvez également visualiser le contenu de tout conteneur défini dans l'extension, et sauvegarder ce contenu dans un fichier pour un examen ultérieur. [Pour plus d'informations, reportez-vous à la section Containers dans le chapitre 4 sur p. 67.](#)

Figure 10-2  
Onglet Déboguer sur les boîtes de dialogue



Vous activez l'onglet Déboguer en définissant la valeur de l'attribut debug de l'élément Extension sur true dans le fichier de spécifications. [Pour plus d'informations, reportez-vous à la section Élément Extension dans le chapitre 4 sur p. 41.](#)

Les champs sur l'onglet sont les suivants :

**ID d'élément.** L'identifiant unique de l'extension — la valeur de l'attribut id de l'élément ExtensionDetails dans le fichier de spécifications.

**Nom de script.** L'identifiant unique du noeud quand il est référencé dans un script - la valeur de l'attribut scriptName de l'élément Node.

**ID d'extension.** Le nom du dossier d'extension où se situent les ressources de fichier et de répertoire pour l'extension. La valeur est formée par concaténation des attributs `providerTag` et `id` (séparé par un caractère « . ») de l'élément `ExtensionDetails`. Remarque : pour la portion `providerTag` de l'identifiant, la valeur ne doit pas inclure la chaîne `spss` qui est réservée à un usage interne.

**Propriétés.** Cette table affiche les détails sélectionnés pour les déclarations `Property` du noeud :

- **Propriété.** L'identifiant unique de la propriété - la valeur du champ `name` de l'élément `Property`.
- **Nom de script.** L'identifiant unique de la propriété référencée dans le script - la valeur de l'attribut `scriptName` de l'élément `Property`.
- **Type de valeur.** Le type de valeur possible pour cette propriété, tel que défini par l'attribut `valueType` de l'élément `Property`.
- **Liste ?** Indique si la propriété est une liste de valeurs du type de valeur spécifié - la valeur de l'attribut `isList` de l'élément `Property`.
- **Partagé ?** Si coché, indique que cette propriété est utilisée à plusieurs endroits dans l'extension (par exemple : noeud de création de modèle, sortie de modèle, applicateur de modèle).
- **Valeur.** La valeur par défaut de la propriété, le cas échéant.

**Conteneurs.** Affiche le contenu du conteneur sélectionné (par exemple, données de modèle). Cliquez sur le champ pour afficher une liste des autres conteneurs définis pour l'extension, et sélectionnez-les pour en afficher le contenu. Cliquez sur le bouton `Enregistrer le conteneur adjacent` pour enregistrer au format XML le contenu du conteneur sélectionné, pour un examen ultérieur.

**Tracer.** Affiche une boîte de dialogue permettant de visualiser la trace en sortie lorsque le noeud est exécuté.

### ***Modification des options d'exécution***

Par défaut, une extension CLEF écrite par l'utilisateur s'exécute dans son propre processus, distinct du processus IBM® SPSS® Modeler principal. De cette façon, une défaillance du processus d'extension n'impliquera pas l'échec du processus SPSS Modeler. Par contre, les modules IBM Corp. fournis s'exécutent par défaut dans le processus principal.

Deux options de configuration sont fournies pour le serveur, afin de permettre à l'administrateur système de choisir l'une ou l'autre des options pour un ou plusieurs des modules nommés. Les deux options comportent une liste séparée par des virgules des identifiants des modules, pour indiquer lesquels sont concernés par la modification.

*Remarque :* La modification de l'une de ces options ne peut être effectuée, en principe, qu'à la demande d'un représentant du service clientèle.

Les options sont les suivantes :

### **Option Exécution en cours de processus**

Cette option permet aux modules d'extension qui sont normalement chargés dans un processus externe (typiquement, les modules écrits par un utilisateur), d'être chargés directement dans SPSS Modeler. Le format est le suivant :

```
clef_inprocess_execution, "  
moduleID1[,  
moduleID2[,...  
moduleIDn]]"
```

où *moduleID* est la valeur de l'attribut id de l'élément `ExtensionDetails` dans le fichier de spécification concerné. Exemple :

```
clef_inprocess_execution, "test.example_filereader"
```

### **Option d'exécution externe**

Cette option permet aux modules d'extension qui sont normalement chargés dans SPSS Modeler (particulièrement, les modules IBM Corp. fournis), d'être chargés dans un processus externe. Le format est le suivant :

```
clef_external_execution, "  
moduleID1[,  
moduleID2[,...  
moduleIDn]]"
```

où *moduleID* est le même que pour `clef_inprocess_execution`. Exemple (fictif) :

```
clef_external_execution, "spss.naivebayes,spss.terminator"
```

### **Ajout ou modification d'une option d'exécution**

Pour ajouter ou modifier une option d'exécution, suivez la procédure indiquée dans « Utilisation du fichier options.cfg » du *guide des performances et de l'administration du serveur SPSS Modeler 14.2*.

## **Distribution des extensions CLEF**

Lorsqu'une nouvelle extension est entièrement testée et prête à être distribuée :

- ▶ Désactivez l'onglet Déboguer si celui-ci était activé. [Pour plus d'informations, reportez-vous à la section Utilisation de l'onglet Déboguer sur p. 248.](#)
- ▶ Créez une structure de fichier reflétant exactement la manière dont vous souhaitez que les fichiers d'extension soient installés. [Pour plus d'informations, reportez-vous à la section Structure de fichier dans le chapitre 1 sur p. 6.](#)

- ▶ Comprimez la structure de fichier en un fichier .zip. Il peut être plus facile de créer des fichiers .zip distincts pour les installations côte-client et côté-serveur.
- ▶ Distribuez les fichiers .zip aux utilisateurs finaux.

## ***Installation des extensions CLEF***

Pour installer une extension CLEF :

- ▶ A la réception d'un fichier .zip contenant une structure de fichier d'extension, procédez à l'extraction des fichiers côté client dans le dossier \ext\lib sous le répertoire d'installation IBM® SPSS® Modeler.
- ▶ Procédez à l'extraction des fichiers côté serveur dans le dossier \ext\bin sous le répertoire d'installation SPSS Modeler (ou son équivalent si vous utilisez IBM® SPSS® Modeler Server).
- ▶ Démarrez IBM® SPSS® Modeler, et vérifiez que les nouveaux noeuds apparaissent dans la palette de noeuds aux emplacements prévus.

## ***Désinstallation des extensions CLEF***

Pour désinstaller une extension CLEF :

- ▶ Localisez le dossier d'extension dans le répertoire \ext\lib sous le répertoire d'installation IBM® SPSS® Modeler.

Si l'extension a également installé un dossier d'extension côté serveur, localisez ce dossier dans le répertoire \ext\bin sous le répertoire d'installation SPSS Modeler ou IBM® SPSS® Modeler Server.

- ▶ Supprimer le ou les dossiers d'extension.

La modification prendra effet au prochain démarrage de IBM® SPSS® Modeler.

# Schéma XML CLEF

## Références des éléments CLEF

Cette section présente les références de tous les éléments dans CLEF.

Chaque rubrique contient une liste des attributs valides d'un élément et de ses éléments parent et enfant. Le diagramme présente tous les enfants des éléments. Veuillez noter que les flèches du diagramme indiquent les éléments qui peuvent être partagés parmi d'autres éléments. Ces éléments apparaissent sous forme de liste dans la table des matières comme enfant de cette rubrique (« Références des éléments CLEF ») plutôt que comme enfant de la rubrique parent.

### Action Élément

Table A-1  
Attributs pour Action

Attribut	Utiliser	Description	Valeurs valides
description	optional		string
descriptionKey	optional		string
id	<b>obligatoire</b>		string
imagePath	optional		string
imagePathKey	optional		string
label	<b>obligatoire</b>		string
labelKey	optional		string
mnemonic	optional		string
mnemonicKey	optional		string
shortcut	optional		string
shortcutKey	optional		string

### Représentation XML

```
<xs:element name="Action">
  <xs:attribute name="id" type="xs:string" use="required"/></xs:attribute>
  <xs:attribute name="label" type="xs:string" use="required"/></xs:attribute>
  <xs:attribute name="labelKey" type="xs:string" use="optional"/></xs:attribute>
  <xs:attribute name="description" type="xs:string" use="optional"/></xs:attribute>
  <xs:attribute name="descriptionKey" type="xs:string" use="optional"/></xs:attribute>
  <xs:attribute name="imagePath" type="xs:string" use="optional"/></xs:attribute>
  <xs:attribute name="imagePathKey" type="xs:string" use="optional"/></xs:attribute>
  <xs:attribute name="mnemonic" type="xs:string" use="optional"/></xs:attribute>
  <xs:attribute name="mnemonicKey" type="xs:string" use="optional"/></xs:attribute>
  <xs:attribute name="shortcut" type="xs:string" use="optional"/></xs:attribute>
  <xs:attribute name="shortcutKey" type="xs:string" use="optional"/></xs:attribute>
</xs:element>
```



**Parent Eléments**[Actions Elément](#)**ActionButton Elément**

Table A-2  
Attributs pour ActionButton

Attribut	Utiliser	Description	Valeurs valides
action	<b>obligatoire</b>		<i>string</i>
showIcon	optional		<i>boolean</i>
showLabel	optional		<i>boolean</i>

**Représentation XML**

```
<xs:element name="ActionButton">
  <xs:sequence>
    <xs:choice>
      <xs:element ref="Layout" minOccurs="0" maxOccurs="1"></xs:element>
      <xs:element ref="Enabled" minOccurs="0" maxOccurs="1"></xs:element>
      <xs:element ref="Visible" minOccurs="0" maxOccurs="1"></xs:element>
    </xs:choice>
  </xs:sequence>
  <xs:attribute name="action" type="xs:string" use="required"></xs:attribute>
  <xs:attribute name="showLabel" type="xs:boolean" use="optional" default="true"></xs:attribute>
  <xs:attribute name="showIcon" type="xs:boolean" use="optional" default="true"></xs:attribute>
</xs:element>
```

**Parent Eléments**[PropertiesPanel Elément](#), [PropertiesSubPanel Elément](#)**Enfant Eléments**[Enabled Elément](#), [Layout Elément](#), [Visible Elément](#)**Eléments connexes Eléments**[ComboBoxControl Elément](#), [ExtensionObjectPanel Elément](#), [ModelViewerPanel Elément](#), [SelectorPanel Elément](#), [StaticText Elément](#), [SystemControls Elément](#), [TabbedPanel Elément](#), [TextBrowserPanel Elément](#)**Actions Elément****Représentation XML**

```
<xs:element name="Actions">
  <xs:sequence minOccurs="0" maxOccurs="unbounded">
    <xs:choice>
      <xs:element ref="Action"></xs:element>
    </xs:choice>
  </xs:sequence>
```

&lt;/xs:element&gt;

**Parent Eléments**[CommonObjects Elément](#)**Enfant Eléments**[Action Elément](#)**AddField Elément**Table A-3  
Attributs pour AddField

Attribut	Utiliser	Description	Valeurs valides
direction	facultatif		<b>in</b> <b>out</b> <b>both</b> <b>none</b> <b>partition</b>
directionRef	optional		
fieldRef	optional		
group	optional		<i>string</i>
label	facultatif		<i>string</i>
missingValuesRef	optional		
name	<b>obligatoire</b>		
prefix	optional		<i>string</i>
role	optional		<b>unknown</b> <b>predictedValue</b> <b>predictedDisplayValue</b> <b>probability</b> <b>residual</b> <b>standardError</b> <b>entityId</b> <b>entityAffinity</b> <b>upperConfidenceLimit</b> <b>lowerConfidenceLimit</b> <b>propensity</b> <b>value</b> <b>supplementary</b>
storage	facultatif		<b>unknown</b> <b>integer</b> <b>real</b> <b>string</b> <b>date</b> <b>time</b> <b>timestamp</b>
storageRef	optional		
tag	optional		<i>string</i>
targetField	optional		<i>string</i>

Attribut	Utiliser	Description	Valeurs valides
type	facultatif		<b>auto</b> <b>range</b> <b>discrete</b> <b>set</b> <b>orderedSet</b> <b>flag</b> <b>typeless</b>
typeRef	optional		
value	optional		<i>string</i>

### Représentation XML

```

<xs:element name="AddField">
  <xs:sequence>
    <xs:choice>
      <xs:element ref="Range" minOccurs="0"/></xs:element>
      <xs:element ref="Values" minOccurs="0"/></xs:element>
      <xs:element ref="NumericInfo" minOccurs="0"/></xs:element>
      <xs:element name="MissingValues" minOccurs="0">
        <xs:sequence>
          <xs:element ref="Values" minOccurs="0" maxOccurs="unbounded"/></xs:element>
          <xs:element ref="Range" minOccurs="0"/></xs:element>
        </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>
  <xs:attribute name="storage" type="FIELD-STORAGE">
    <xs:enumeration value="unknown"/></xs:enumeration>
    <xs:enumeration value="integer"/></xs:enumeration>
    <xs:enumeration value="real"/></xs:enumeration>
    <xs:enumeration value="string"/></xs:enumeration>
    <xs:enumeration value="date"/></xs:enumeration>
    <xs:enumeration value="time"/></xs:enumeration>
    <xs:enumeration value="timestamp"/></xs:enumeration>
  </xs:attribute>
  <xs:attribute name="type" type="FIELD-TYPE">
    <xs:enumeration value="auto"/></xs:enumeration>
    <xs:enumeration value="range"/></xs:enumeration>
    <xs:enumeration value="discrete"/></xs:enumeration>
    <xs:enumeration value="set"/></xs:enumeration>
    <xs:enumeration value="orderedSet"/></xs:enumeration>
    <xs:enumeration value="flag"/></xs:enumeration>
    <xs:enumeration value="typeless"/></xs:enumeration>
  </xs:attribute>
  <xs:attribute name="direction" type="FIELD-DIRECTION">
    <xs:enumeration value="in"/></xs:enumeration>
    <xs:enumeration value="out"/></xs:enumeration>
    <xs:enumeration value="both"/></xs:enumeration>
    <xs:enumeration value="none"/></xs:enumeration>
    <xs:enumeration value="partition"/></xs:enumeration>
  </xs:attribute>
  <xs:attribute name="label" type="xs:string"/></xs:attribute>
  <xs:attribute name="fieldRef" type="EVALUATED-STRING" use="optional"/></xs:attribute>

```

```

<xs:attribute name="storageRef" type="EVALUATED-STRING" use="optional"/></xs:attribute>
<xs:attribute name="typeRef" type="EVALUATED-STRING" use="optional"/></xs:attribute>
<xs:attribute name="directionRef" type="EVALUATED-STRING" use="optional"/></xs:attribute>
<xs:attribute name="missingValuesRef" type="EVALUATED-STRING" use="optional"/></xs:attribute>
<xs:attribute name="role" type="MODEL-FIELD-ROLE" use="optional">
  <xs:enumeration value="unknown"/></xs:enumeration>
  <xs:enumeration value="predictedValue"/></xs:enumeration>
  <xs:enumeration value="predictedDisplayValue"/></xs:enumeration>
  <xs:enumeration value="probability"/></xs:enumeration>
  <xs:enumeration value="residual"/></xs:enumeration>
  <xs:enumeration value="standardError"/></xs:enumeration>
  <xs:enumeration value="entityId"/></xs:enumeration>
  <xs:enumeration value="entityAffinity"/></xs:enumeration>
  <xs:enumeration value="upperConfidenceLimit"/></xs:enumeration>
  <xs:enumeration value="lowerConfidenceLimit"/></xs:enumeration>
  <xs:enumeration value="propensity"/></xs:enumeration>
  <xs:enumeration value="value"/></xs:enumeration>
  <xs:enumeration value="supplementary"/></xs:enumeration>
</xs:attribute>
<xs:attribute name="targetField" type="xs:string" use="optional"/></xs:attribute>
<xs:attribute name="value" type="xs:string" use="optional"/></xs:attribute>
<xs:attribute name="group" type="xs:string" use="optional"/></xs:attribute>
<xs:attribute name="tag" type="xs:string" use="optional"/></xs:attribute>
<xs:attribute name="prefix" type="xs:string" use="optional"/></xs:attribute>
</xs:element>

```

### **Parent Eléments**

[ForEach Elément](#), [ModelFields Elément](#)

### **Enfant Eléments**

[MissingValues Elément](#), [ModelField Elément](#), [NumericInfo Elément](#), [Range Elément](#), [Values Elément](#)

### **Eléments connexes Eléments**

[ChangeField Elément](#)

## **MissingValues Elément**

Table A-4  
Attributs pour MissingValues

Attribut	Utiliser	Description	Valeurs valides
treatNullAsMissing	optional		boolean
treatWhitespaceAsMissing	optional		boolean

### **Représentation XML**

```

<xs:element name="MissingValues" minOccurs="0">
  <xs:sequence>
    <xs:element ref="Values" minOccurs="0" maxOccurs="unbounded"/></xs:element>
    <xs:element ref="Range" minOccurs="0"/></xs:element>
  </xs:sequence>

```

```

<xs:attribute name="treatWhitespaceAsMissing" type="xs:boolean" use="optional"
  default="true"></xs:attribute>
<xs:attribute name="treatNullAsMissing" type="xs:boolean" use="optional" default="true"></xs:attribute>
</xs:element>

```

### Parent Eléments

[AddField Elément](#)

### Enfant Eléments

[Range Elément](#), [Values Elément](#)

## ModelField Elément

Table A-5  
Attributs pour ModelField

Attribut	Utiliser	Description	Valeurs valides
group	facultatif		
role	<b>obligatoire</b>		<b>unknown</b> <b>predictedValue</b> <b>predictedDisplayValue</b> <b>probability</b> <b>residual</b> <b>standardError</b> <b>entityId</b> <b>entityAffinity</b> <b>upperConfidenceLimit</b> <b>lowerConfidenceLimit</b> <b>propensity</b> <b>value</b> <b>supplementary</b>
tag	facultatif		<i>string</i>
targetField	facultatif		<i>string</i>
value	facultatif		<i>string</i>

### Représentation XML

```

<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>
    <xs:enumeration value="predictedValue"></xs:enumeration>
    <xs:enumeration value="predictedDisplayValue"></xs:enumeration>
    <xs:enumeration value="probability"></xs:enumeration>
    <xs:enumeration value="residual"></xs:enumeration>
    <xs:enumeration value="standardError"></xs:enumeration>
    <xs:enumeration value="entityId"></xs:enumeration>
    <xs:enumeration value="entityAffinity"></xs:enumeration>
    <xs:enumeration value="upperConfidenceLimit"></xs:enumeration>
    <xs:enumeration value="lowerConfidenceLimit"></xs:enumeration>
    <xs:enumeration value="propensity"></xs:enumeration>
    <xs:enumeration value="value"></xs:enumeration>
    <xs:enumeration value="supplementary"></xs:enumeration>
  </xs:attribute>
  <xs:attribute name="targetField" type="xs:string"></xs:attribute>

```

```

    <xs:attribute name="value" type="xs:string"></xs:attribute>
    <xs:attribute name="group" type="MODEL-FIELD-GROUP"></xs:attribute>
    <xs:attribute name="tag" type="xs:string"></xs:attribute>
  </xs:element>

```

### Parent Eléments

[AddField Elément](#)

## And Elément

### Représentation XML

```

<xs:element name="And">
  <xs:sequence minOccurs="2" maxOccurs="unbounded">
    <xs:group ref="CONDITION-EXPRESSION">
      <xs:choice>
        <xs:element ref="Condition"></xs:element>
        <xs:element ref="And"></xs:element>
        <xs:element ref="Or"></xs:element>
        <xs:element ref="Not"></xs:element>
      </xs:choice>
    </xs:group>
  </xs:sequence>
</xs:element>

```

### Parent Eléments

[Command Elément](#), [Constraint Elément](#), [CreateDocument Elément](#), [CreateDocumentOutput Elément](#), [CreateInteractiveDocumentBuilder Elément](#), [CreateInteractiveModelBuilder Elément](#), [CreateModel Elément](#), [CreateModelApplier Elément](#), [CreateModelOutput Elément](#), [Enabled Elément](#), [Not Elément](#), [Option Elément](#), [Or Elément](#), [Run Elément](#), [Visible Elément](#)

### Enfant Eléments

[And Elément](#), [Condition Elément](#), [Not Elément](#), [Or Elément](#)

## Attribute Elément

Table A-6  
Attributs pour Attribute

Attribut	Utiliser	Description	Valeurs valides
defaultValue	optional		
description	optional		<i>string</i>
descriptionKey	optional		<i>string</i>
isList	optional		<i>boolean</i>
label	<b>obligatoire</b>		<i>string</i>
labelKey	optional		<i>string</i>

Attribut	Utiliser	Description	Valeurs valides
name	<b>obligatoire</b>		<i>string</i>
valueType	facultatif		<b>string</b> <b>encryptedString</b> <b>integer</b> <b>double</b> <b>boolean</b> <b>date</b> <b>enum</b>

### Représentation XML

```

<xs:element name="Attribute">
  <xs:attribute name="name" type="xs:string" use="required"/></xs:attribute>
  <xs:attribute name="label" type="xs:string" use="required"/></xs:attribute>
  <xs:attribute name="labelKey" type="xs:string" use="optional"/></xs:attribute>
  <xs:attribute name="description" type="xs:string" use="optional"/></xs:attribute>
  <xs:attribute name="descriptionKey" type="xs:string" use="optional"/></xs:attribute>
  <xs:attribute name="valueType" type="ATTRIBUTE-VALUE-TYPE">
    <xs:enumeration value="string"/></xs:enumeration>
    <xs:enumeration value="encryptedString"/></xs:enumeration>
    <xs:enumeration value="integer"/></xs:enumeration>
    <xs:enumeration value="double"/></xs:enumeration>
    <xs:enumeration value="boolean"/></xs:enumeration>
    <xs:enumeration value="date"/></xs:enumeration>
    <xs:enumeration value="enum"/></xs:enumeration>
  </xs:attribute>
  <xs:attribute name="defaultValue" type="EVALUATED-STRING" use="optional"/></xs:attribute>
  <xs:attribute name="isList" type="xs:boolean" use="optional" default="false"/></xs:attribute>
</xs:element>

```

### Parent Eléments

[Catalog Elément](#), [Structure Elément](#)

## BinaryFormat Elément

### Représentation XML

```
<xs:element name="BinaryFormat"/></xs:element>
```

### Parent Eléments

[FileFormatType Elément](#)

## Catalog Elément

Table A-7  
Attributs pour Catalog

Attribut	Utiliser	Description	Valeurs valides
id	<b>obligatoire</b>		<i>string</i>
valueColumn	<b>obligatoire</b>		<i>integer</i>

**Représentation XML**

```

<xs:element name="Catalog">
  <xs:sequence minOccurs="1" maxOccurs="unbounded">
    <xs:element ref="Attribute"/></xs:element>
  </xs:sequence>
  <xs:attribute name="id" type="xs:string" use="required"/></xs:attribute>
  <xs:attribute name="valueColumn" type="xs:integer" use="required"/></xs:attribute>
</xs:element>

```

**Parent Eléments**[Catalogs Elément](#)**Enfant Eléments**[Attribute Elément](#)**Catalogs Elément****Représentation XML**

```

<xs:element name="Catalogs">
  <xs:sequence minOccurs="0" maxOccurs="unbounded">
    <xs:choice>
      <xs:element ref="Catalog"/></xs:element>
    </xs:choice>
  </xs:sequence>
</xs:element>

```

**Parent Eléments**[CommonObjects Elément](#)**Enfant Eléments**[Catalog Elément](#)**ChangeField Elément**

Table A-8  
Attributs pour ChangeField

Attribut	Utiliser	Description	Valeurs valides
direction	facultatif		<b>in</b> <b>out</b> <b>both</b> <b>none</b> <b>partition</b>
directionRef	optional		
fieldRef	<b>obligatoire</b>		
label	facultatif		<i>string</i>
missingValuesRef	optional		



Attribut	Utiliser	Description	Valeurs valides
name	<b>obligatoire</b>		
storage	facultatif		<b>unknown</b> <b>integer</b> <b>real</b> <b>string</b> <b>date</b> <b>time</b> <b>timestamp</b>
storageRef	optional		
type	facultatif		<b>auto</b> <b>range</b> <b>discrete</b> <b>set</b> <b>orderedSet</b> <b>flag</b> <b>typeless</b>
typeRef	optional		

### Représentation XML

```

<xs:element name="ChangeField">
  <xs:sequence>
    <xs:choice>
      <xs:element ref="Range" minOccurs="0"></xs:element>
      <xs:element ref="Values" minOccurs="0"></xs:element>
      <xs:element ref="NumericInfo" minOccurs="0"></xs:element>
      <xs:element name="MissingValues" minOccurs="0">
        <xs:sequence>
          <xs:element ref="Values" minOccurs="0" maxOccurs="unbounded"></xs:element>
          <xs:element ref="Range" minOccurs="0"></xs:element>
        </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>
  <xs:attribute name="storage" type="FIELD-STORAGE">
    <xs:enumeration value="unknown"></xs:enumeration>
    <xs:enumeration value="integer"></xs:enumeration>
    <xs:enumeration value="real"></xs:enumeration>
    <xs:enumeration value="string"></xs:enumeration>
    <xs:enumeration value="date"></xs:enumeration>
    <xs:enumeration value="time"></xs:enumeration>
    <xs:enumeration value="timestamp"></xs:enumeration>
  </xs:attribute>
  <xs:attribute name="type" type="FIELD-TYPE">
    <xs:enumeration value="auto"></xs:enumeration>
    <xs:enumeration value="range"></xs:enumeration>
    <xs:enumeration value="discrete"></xs:enumeration>
    <xs:enumeration value="set"></xs:enumeration>
    <xs:enumeration value="orderedSet"></xs:enumeration>
    <xs:enumeration value="flag"></xs:enumeration>
    <xs:enumeration value="typeless"></xs:enumeration>
  </xs:attribute>
  <xs:attribute name="direction" type="FIELD-DIRECTION">

```

```

    <xs:enumeration value="in"></xs:enumeration>
    <xs:enumeration value="out"></xs:enumeration>
    <xs:enumeration value="both"></xs:enumeration>
    <xs:enumeration value="none"></xs:enumeration>
    <xs:enumeration value="partition"></xs:enumeration>
  </xs:attribute>
  <xs:attribute name="label" type="xs:string"></xs:attribute>
  <xs:attribute name="fieldRef" type="EVALUATED-STRING" use="required"></xs:attribute>
  <xs:attribute name="storageRef" type="EVALUATED-STRING" use="optional"></xs:attribute>
  <xs:attribute name="typeRef" type="EVALUATED-STRING" use="optional"></xs:attribute>
  <xs:attribute name="directionRef" type="EVALUATED-STRING" use="optional"></xs:attribute>
  <xs:attribute name="missingValuesRef" type="EVALUATED-STRING" use="optional"></xs:attribute>
</xs:element>

```

**Parent Eléments**

[ForEach Elément](#), [ModelFields Elément](#)

**Enfant Eléments**

[MissingValues Elément](#), [ModelField Elément](#), [NumericInfo Elément](#), [Range Elément](#), [Values Elément](#)

**Eléments connexes Eléments**

[AddField Elément](#)

**MissingValues Elément**

Table A-9  
Attributs pour MissingValues

Attribut	Utiliser	Description	Valeurs valides
treatNullAsMissing	optional		boolean
treatWhitespaceAsMissing	optional		boolean

**Représentation XML**

```

<xs:element name="MissingValues" minOccurs="0">
  <xs:sequence>
    <xs:element ref="Values" minOccurs="0" maxOccurs="unbounded"></xs:element>
    <xs:element ref="Range" minOccurs="0"></xs:element>
  </xs:sequence>
  <xs:attribute name="treatWhitespaceAsMissing" type="xs:boolean" use="optional"
  default="true"></xs:attribute>
  <xs:attribute name="treatNullAsMissing" type="xs:boolean" use="optional" default="true"></xs:attribute>
</xs:element>

```

**Parent Eléments**

[ChangeField Elément](#)

**Enfant Eléments**

[Range Elément](#), [Values Elément](#)

**ModelField Élément**

Table A-10  
Attributs pour ModelField

Attribut	Utiliser	Description	Valeurs valides
group	facultatif		
role	<b>obligatoire</b>		<b>unknown</b> <b>predictedValue</b> <b>predictedDisplayValue</b> <b>probability</b> <b>residual</b> <b>standardError</b> <b>entityId</b> <b>entityAffinity</b> <b>upperConfidenceLimit</b> <b>lowerConfidenceLimit</b> <b>propensity</b> <b>value</b> <b>supplementary</b>
tag	facultatif		<i>string</i>
targetField	facultatif		<i>string</i>
value	facultatif		<i>string</i>

**Représentation XML**

```
<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>
    <xs:enumeration value="predictedValue"></xs:enumeration>
    <xs:enumeration value="predictedDisplayValue"></xs:enumeration>
    <xs:enumeration value="probability"></xs:enumeration>
    <xs:enumeration value="residual"></xs:enumeration>
    <xs:enumeration value="standardError"></xs:enumeration>
    <xs:enumeration value="entityId"></xs:enumeration>
    <xs:enumeration value="entityAffinity"></xs:enumeration>
    <xs:enumeration value="upperConfidenceLimit"></xs:enumeration>
    <xs:enumeration value="lowerConfidenceLimit"></xs:enumeration>
    <xs:enumeration value="propensity"></xs:enumeration>
    <xs:enumeration value="value"></xs:enumeration>
    <xs:enumeration value="supplementary"></xs:enumeration>
  </xs:attribute>
  <xs:attribute name="targetField" type="xs:string"></xs:attribute>
  <xs:attribute name="value" type="xs:string"></xs:attribute>
  <xs:attribute name="group" type="MODEL-FIELD-GROUP"></xs:attribute>
  <xs:attribute name="tag" type="xs:string"></xs:attribute>
</xs:element>
```

**Parent Éléments**

[ChangeField Élément](#)

## CheckBoxControl Élément

Table A-11  
Attributs pour CheckBoxControl

Attribut	Utiliser	Description	Valeurs valides
description	optional		string
descriptionKey	optional		string
invert	optional		boolean
label	optional		string
labelAbove	optional		boolean
labelKey	optional		string
labelWidth	optional		positiveInteger
mnemonic	optional		string
mnemonicKey	optional		string
property	<b>obligatoire</b>		string
showLabel	optional		boolean

### Représentation XML

```
<xs:element name="CheckBoxControl">
  <xs:sequence>
    <xs:choice>
      <xs:element ref="Layout" minOccurs="0" maxOccurs="1"></xs:element>
      <xs:element ref="Enabled" minOccurs="0" maxOccurs="1"></xs:element>
      <xs:element ref="Visible" minOccurs="0" maxOccurs="1"></xs:element>
    </xs:choice>
  </xs:sequence>
  <xs:attribute name="property" type="xs:string" use="required"></xs:attribute>
  <xs:attribute name="showLabel" type="xs:boolean" use="optional" default="true"></xs:attribute>
  <xs:attribute name="label" type="xs:string" use="optional"></xs:attribute>
  <xs:attribute name="labelKey" type="xs:string" use="optional"></xs:attribute>
  <xs:attribute name="mnemonic" type="xs:string" use="optional"></xs:attribute>
  <xs:attribute name="mnemonicKey" type="xs:string" use="optional"></xs:attribute>
  <xs:attribute name="labelWidth" type="xs:positiveInteger" use="optional" default="1"></xs:attribute>
  <xs:attribute name="labelAbove" type="xs:boolean" use="optional" default="false"></xs:attribute>
  <xs:attribute name="description" type="xs:string" use="optional"></xs:attribute>
  <xs:attribute name="descriptionKey" type="xs:string" use="optional"></xs:attribute>
  <xs:attribute name="invert" type="xs:boolean" use="optional" default="false"></xs:attribute>
</xs:element>
```

### Parent Éléments

[PropertiesPanel Élément](#), [PropertiesSubPanel Élément](#)

### Enfant Éléments

[Enabled Élément](#), [Layout Élément](#), [Visible Élément](#)

### Éléments connexes Éléments

[CheckBoxGroupControl Élément](#), [ClientDirectoryChooserControl Élément](#),  
[ClientFileChooserControl Élément](#), [DBConnectionChooserControl Élément](#),  
[DBTableChooserControl Élément](#), [MultiFieldChooserControl Élément](#), [PasswordBoxControl](#)

Élément, PropertyControl Élément, RadioButtonGroupControl Élément, ServerDirectoryChooserControl Élément, ServerFileChooserControl Élément, SingleFieldChooserControl Élément, SingleFieldValueChooserControl Élément, SpinnerControl Élément, TableControl Élément, TextAreaControl Élément, TextBoxControl Élément

## CheckBoxGroupControl Élément

Table A-12  
Attributs pour CheckBoxGroupControl

Attribut	Utiliser	Description	Valeurs valides
description	optional		string
descriptionKey	optional		string
label	optional		string
labelAbove	optional		boolean
labelKey	optional		string
labelWidth	optional		positiveInteger
layoutByRow	optional		boolean
mnemonic	optional		string
mnemonicKey	optional		string
property	<b>obligatoire</b>		string
rows	optional		positiveInteger
showLabel	optional		boolean
useSubPanel	optional		boolean

### Représentation XML

```
<xs:element name="CheckBoxGroupControl">
  <xs:sequence>
    <xs:choice>
      <xs:element ref="Layout" minOccurs="0" maxOccurs="1"/></xs:element>
      <xs:element ref="Enabled" minOccurs="0" maxOccurs="1"/></xs:element>
      <xs:element ref="Visible" minOccurs="0" maxOccurs="1"/></xs:element>
    </xs:choice>
  </xs:sequence>
  <xs:attribute name="property" type="xs:string" use="required"/></xs:attribute>
  <xs:attribute name="showLabel" type="xs:boolean" use="optional" default="true"/></xs:attribute>
  <xs:attribute name="label" type="xs:string" use="optional"/></xs:attribute>
  <xs:attribute name="labelKey" type="xs:string" use="optional"/></xs:attribute>
  <xs:attribute name="mnemonic" type="xs:string" use="optional"/></xs:attribute>
  <xs:attribute name="mnemonicKey" type="xs:string" use="optional"/></xs:attribute>
  <xs:attribute name="labelWidth" type="xs:positiveInteger" use="optional" default="1"/></xs:attribute>
  <xs:attribute name="labelAbove" type="xs:boolean" use="optional" default="false"/></xs:attribute>
  <xs:attribute name="description" type="xs:string" use="optional"/></xs:attribute>
  <xs:attribute name="descriptionKey" type="xs:string" use="optional"/></xs:attribute>
  <xs:attribute name="rows" type="xs:positiveInteger" use="optional" default="1"/></xs:attribute>
  <xs:attribute name="layoutByRow" type="xs:boolean" use="optional" default="true"/></xs:attribute>
  <xs:attribute name="useSubPanel" type="xs:boolean" use="optional" default="true"/></xs:attribute>
</xs:element>
```

**Parent Eléments**

[PropertiesPanel Elément](#), [PropertiesSubPanel Elément](#)

**Enfant Eléments**

[Enabled Elément](#), [Layout Elément](#), [Visible Elément](#)

**Eléments connexes Eléments**

[CheckBoxControl Elément](#), [ClientDirectoryChooserControl Elément](#), [ClientFileChooserControl Elément](#), [DBConnectionChooserControl Elément](#), [DBTableChooserControl Elément](#), [MultiFieldChooserControl Elément](#), [PasswordBoxControl Elément](#), [PropertyControl Elément](#), [RadioButtonGroupControl Elément](#), [ServerDirectoryChooserControl Elément](#), [ServerFileChooserControl Elément](#), [SingleFieldChooserControl Elément](#), [SingleFieldValueChooserControl Elément](#), [SpinnerControl Elément](#), [TableControl Elément](#), [TextAreaControl Elément](#), [TextBoxControl Elément](#)

**ClientDirectoryChooserControl Elément**

Table A-13

Attributs pour *ClientDirectoryChooserControl*

Attribut	Utiliser	Description	Valeurs valides
description	optional		<i>string</i>
descriptionKey	optional		<i>string</i>
label	optional		<i>string</i>
labelAbove	optional		<i>boolean</i>
labelKey	optional		<i>string</i>
labelWidth	optional		<i>positiveInteger</i>
mnemonic	optional		<i>string</i>
mnemonicKey	optional		<i>string</i>
mode	<b>obligatoire</b>		<b>open save import export</b>
property	<b>obligatoire</b>		<i>string</i>
showLabel	optional		<i>boolean</i>

**Représentation XML**

```
<xs:element name="ClientDirectoryChooserControl">
  <xs:sequence>
    <xs:choice>
      <xs:element ref="Layout" minOccurs="0" maxOccurs="1"/></xs:element>
      <xs:element ref="Enabled" minOccurs="0" maxOccurs="1"/></xs:element>
      <xs:element ref="Visible" minOccurs="0" maxOccurs="1"/></xs:element>
    </xs:choice>
  </xs:sequence>
  <xs:attribute name="property" type="xs:string" use="required"/></xs:attribute>
  <xs:attribute name="showLabel" type="xs:boolean" use="optional" default="true"/></xs:attribute>
  <xs:attribute name="label" type="xs:string" use="optional"/></xs:attribute>
</xs:element>
```

```

<xs:attribute name="labelKey" type="xs:string" use="optional"/></xs:attribute>
<xs:attribute name="mnemonic" type="xs:string" use="optional"/></xs:attribute>
<xs:attribute name="mnemonicKey" type="xs:string" use="optional"/></xs:attribute>
<xs:attribute name="labelWidth" type="xs:positiveInteger" use="optional" default="1"/></xs:attribute>
<xs:attribute name="labelAbove" type="xs:boolean" use="optional" default="false"/></xs:attribute>
<xs:attribute name="description" type="xs:string" use="optional"/></xs:attribute>
<xs:attribute name="descriptionKey" type="xs:string" use="optional"/></xs:attribute>
<xs:attribute name="mode" type="FILE-CHOOSER-MODE" use="required">
  <xs:enumeration value="open"/></xs:enumeration>
  <xs:enumeration value="save"/></xs:enumeration>
  <xs:enumeration value="import"/></xs:enumeration>
  <xs:enumeration value="export"/></xs:enumeration>
</xs:attribute>
</xs:element>

```

### Parent Eléments

[PropertiesPanel Elément](#), [PropertiesSubPanel Elément](#)

### Enfant Eléments

[Enabled Elément](#), [Layout Elément](#), [Visible Elément](#)

### Eléments connexes Eléments

[CheckBoxControl Elément](#), [CheckBoxGroupControl Elément](#), [ClientFileChooserControl Elément](#), [DBConnectionChooserControl Elément](#), [DBTableChooserControl Elément](#), [MultiFieldChooserControl Elément](#), [PasswordBoxControl Elément](#), [PropertyControl Elément](#), [RadioButtonGroupControl Elément](#), [ServerDirectoryChooserControl Elément](#), [ServerFileChooserControl Elément](#), [SingleFieldChooserControl Elément](#), [SingleFieldValueChooserControl Elément](#), [SpinnerControl Elément](#), [TableControl Elément](#), [TextAreaControl Elément](#), [TextBoxControl Elément](#)

## ClientFileChooserControl Elément

Table A-14  
Attributs pour ClientFileChooserControl

Attribut	Utiliser	Description	Valeurs valides
description	optional		string
descriptionKey	optional		string
label	optional		string
labelAbove	optional		boolean
labelKey	optional		string
labelWidth	optional		positiveInteger
mnemonic	optional		string
mnemonicKey	optional		string
mode	<b>obligatoire</b>		<b>open</b> <b>save</b> <b>import</b> <b>export</b>
property	<b>obligatoire</b>		string
showLabel	optional		boolean

**Représentation XML**

```

<xs:element name="ClientFileChooserControl">
  <xs:sequence>
    <xs:choice>
      <xs:element ref="Layout" minOccurs="0" maxOccurs="1"></xs:element>
      <xs:element ref="Enabled" minOccurs="0" maxOccurs="1"></xs:element>
      <xs:element ref="Visible" minOccurs="0" maxOccurs="1"></xs:element>
    </xs:choice>
  </xs:sequence>
  <xs:attribute name="property" type="xs:string" use="required"></xs:attribute>
  <xs:attribute name="showLabel" type="xs:boolean" use="optional" default="true"></xs:attribute>
  <xs:attribute name="label" type="xs:string" use="optional"></xs:attribute>
  <xs:attribute name="labelKey" type="xs:string" use="optional"></xs:attribute>
  <xs:attribute name="mnemonic" type="xs:string" use="optional"></xs:attribute>
  <xs:attribute name="mnemonicKey" type="xs:string" use="optional"></xs:attribute>
  <xs:attribute name="labelWidth" type="xs:positiveInteger" use="optional" default="1"></xs:attribute>
  <xs:attribute name="labelAbove" type="xs:boolean" use="optional" default="false"></xs:attribute>
  <xs:attribute name="description" type="xs:string" use="optional"></xs:attribute>
  <xs:attribute name="descriptionKey" type="xs:string" use="optional"></xs:attribute>
  <xs:attribute name="mode" type="FILE-CHOOSER-MODE" use="required">
    <xs:enumeration value="open"></xs:enumeration>
    <xs:enumeration value="save"></xs:enumeration>
    <xs:enumeration value="import"></xs:enumeration>
    <xs:enumeration value="export"></xs:enumeration>
  </xs:attribute>
</xs:element>

```

**Parent Éléments**

[PropertiesPanel Élément](#), [PropertiesSubPanel Élément](#)

**Enfant Éléments**

[Enabled Élément](#), [Layout Élément](#), [Visible Élément](#)

**Éléments connexes Éléments**

[CheckBoxControl Élément](#), [CheckBoxGroupControl Élément](#), [ClientDirectoryChooserControl Élément](#), [DBConnectionChooserControl Élément](#), [DBTableChooserControl Élément](#), [MultiFieldChooserControl Élément](#), [PasswordBoxControl Élément](#), [PropertyControl Élément](#), [RadioButtonGroupControl Élément](#), [ServerDirectoryChooserControl Élément](#), [ServerFileChooserControl Élément](#), [SingleFieldChooserControl Élément](#), [SingleFieldValueChooserControl Élément](#), [SpinnerControl Élément](#), [TableControl Élément](#), [TextAreaControl Élément](#), [TextBoxControl Élément](#)

**ComboBoxControl Élément**

Table A-15  
Attributs pour *ComboBoxControl*

Attribut	Utiliser	Description	Valeurs valides
description	optional		<i>string</i>
descriptionKey	optional		<i>string</i>



Attribut	Utiliser	Description	Valeurs valides
label	optional		string
labelAbove	optional		boolean
labelKey	optional		string
labelWidth	optional		positiveInteger
mnemonic	optional		string
mnemonicKey	optional		string
property	<b>obligatoire</b>		string
showLabel	optional		boolean

### Représentation XML

```
<xs:element name="ComboBoxControl" type="CONTROLLER">
  <xs:sequence>
    <xs:choice>
      <xs:element ref="Layout" minOccurs="0" maxOccurs="1"/></xs:element>
      <xs:element ref="Enabled" minOccurs="0" maxOccurs="1"/></xs:element>
      <xs:element ref="Visible" minOccurs="0" maxOccurs="1"/></xs:element>
    </xs:choice>
  </xs:sequence>
  <xs:attribute name="property" type="xs:string" use="required"/></xs:attribute>
  <xs:attribute name="showLabel" type="xs:boolean" use="optional" default="true"/></xs:attribute>
  <xs:attribute name="label" type="xs:string" use="optional"/></xs:attribute>
  <xs:attribute name="labelKey" type="xs:string" use="optional"/></xs:attribute>
  <xs:attribute name="mnemonic" type="xs:string" use="optional"/></xs:attribute>
  <xs:attribute name="mnemonicKey" type="xs:string" use="optional"/></xs:attribute>
  <xs:attribute name="labelWidth" type="xs:positiveInteger" use="optional" default="1"/></xs:attribute>
  <xs:attribute name="labelAbove" type="xs:boolean" use="optional" default="false"/></xs:attribute>
  <xs:attribute name="description" type="xs:string" use="optional"/></xs:attribute>
  <xs:attribute name="descriptionKey" type="xs:string" use="optional"/></xs:attribute>
</xs:element>
```

Table A-16  
Étendus Types

Type	Description
<a href="#">ItemChooserControl</a>	

### Parent Éléments

[PropertiesPanel Élément](#), [PropertiesSubPanel Élément](#)

### Enfant Éléments

[Enabled Élément](#), [Layout Élément](#), [Visible Élément](#)

### Éléments connexes Éléments

[ActionButton Élément](#), [ExtensionObjectPanel Élément](#), [ModelViewerPanel Élément](#),  
[SelectorPanel Élément](#), [StaticText Élément](#), [SystemControls Élément](#), [TabbedPanel Élément](#),  
[TextBrowserPanel Élément](#)

## Command Elément

Table A-17  
Attributs pour Command

Attribut	Utiliser	Description	Valeurs valides
path	obligatoire		

### Représentation XML

```
<xs:element name="Command">
  <xs:sequence>
    <xs:group ref="CONDITION-EXPRESSION" minOccurs="0">
      <xs:choice>
        <xs:element ref="Condition"></xs:element>
        <xs:element ref="And"></xs:element>
        <xs:element ref="Or"></xs:element>
        <xs:element ref="Not"></xs:element>
      </xs:choice>
    </xs:group>
  </xs:sequence>
  <xs:attribute name="path" type="EVALUATED-STRING" use="required"></xs:attribute>
</xs:element>
```

### Parent Eléments

[Run Elément](#)

### Enfant Eléments

[And Elément](#), [Condition Elément](#), [Not Elément](#), [Or Elément](#)

## CommonObjects Elément

Provides a location for definitions that are global to the extension

Table A-18  
Attributs pour CommonObjects

Attribut	Utiliser	Description	Valeurs valides
extensionListenerClass	optional		<i>string</i>

### Représentation XML

```
<xs:element name="CommonObjects">
  <xs:all>
    <xs:element ref="PropertyTypes" minOccurs="0"></xs:element>
    <xs:element ref="PropertySets" minOccurs="0"></xs:element>
    <xs:element ref="FileFormatTypes" minOccurs="0"></xs:element>
    <xs:element ref="ContainerTypes" minOccurs="0"></xs:element>
    <xs:element ref="Actions" minOccurs="0"></xs:element>
    <xs:element ref="Catalogs" minOccurs="0"></xs:element>
  </xs:all>
  <xs:attribute name="extensionListenerClass" type="xs:string" use="optional"></xs:attribute>
</xs:element>
```

**Parent Eléments**

[Extension Elément](#)

**Enfant Eléments**

[Actions Elément](#), [Catalogs Elément](#), [ContainerTypes Elément](#), [FileFormatTypes Elément](#), [PropertySets Elément](#), [PropertyTypes Elément](#)

**Condition Elément**

Table A-19  
Attributs pour Condition

Attribut	Utiliser	Description	Valeurs valides
container	optional		<i>string</i>
control	optional		<i>string</i>
expression	optional		
listMode	optional		<b>all</b> <b>any</b>
op	<b>obligatoire</b>		<b>equals</b> <b>notEquals</b> <b>isEmpty</b> <b>isNotEmpty</b> <b>lessThan</b> <b>lessOrEquals</b> <b>greaterThan</b> <b>greaterOrEquals</b> <b>equalsIgnoreCase</b> <b>isSubstring</b> <b>startsWith</b> <b>endsWith</b> <b>startsWithIgnoreCase</b> <b>endsWithIgnoreCase</b> <b>isSubstring</b> <b>hasSubstring</b> <b>isSubstringIgnoreCase</b> <b>hasSubstringIgnoreCase</b> <b>in</b> <b>countEquals</b> <b>countLessThan</b> <b>countLessOrEquals</b> <b>countGreaterThan</b> <b>countGreaterOrEquals</b> <b>contains</b> <b>storageEquals</b> <b>typeEquals</b> <b>directionEquals</b> <b>isMeasureDiscrete</b> <b>isMeasureContinuous</b> <b>isMeasureTypeless</b> <b>isMeasureUnknown</b> <b>isStorageString</b> <b>isStorageNumeric</b> <b>isStorageDatetime</b> <b>isStorageUnknown</b> <b>isModelOutput</b> <b>modelOutputRoleEquals</b>

Attribut	Utiliser	Description	Valeurs valides
			<b>modelOutputTargetField-Equals</b> <b>modelOutputHasValue</b> <b>modelOutputTagEquals</b> <b>enumRestriction</b>
property	optional		<i>string</i>
value	optional		

### Représentation XML

```

<xs:element name="Condition">
  <xs:attribute name="expression" type="EVALUATED-STRING" use="optional"/></xs:attribute>
  <xs:attribute name="control" type="xs:string" use="optional"/></xs:attribute>
  <xs:attribute name="property" type="xs:string" use="optional"/></xs:attribute>
  <xs:attribute name="container" type="xs:string" use="optional"/></xs:attribute>
  <xs:attribute name="op" type="CONDITION-TEST" use="required">
    <xs:enumeration value="equals"/></xs:enumeration>
    <xs:enumeration value="notEquals"/></xs:enumeration>
    <xs:enumeration value="isEmpty"/></xs:enumeration>
    <xs:enumeration value="isNotEmpty"/></xs:enumeration>
    <xs:enumeration value="lessThan"/></xs:enumeration>
    <xs:enumeration value="lessOrEquals"/></xs:enumeration>
    <xs:enumeration value="greaterThan"/></xs:enumeration>
    <xs:enumeration value="greaterOrEquals"/></xs:enumeration>
    <xs:enumeration value="equalsIgnoreCase"/></xs:enumeration>
    <xs:enumeration value="isSubstring"/></xs:enumeration>
    <xs:enumeration value="startsWith"/></xs:enumeration>
    <xs:enumeration value="endsWith"/></xs:enumeration>
    <xs:enumeration value="startsWithIgnoreCase"/></xs:enumeration>
    <xs:enumeration value="endsWithIgnoreCase"/></xs:enumeration>
    <xs:enumeration value="isSubstring"/></xs:enumeration>
    <xs:enumeration value="hasSubstring"/></xs:enumeration>
    <xs:enumeration value="isSubstringIgnoreCase"/></xs:enumeration>
    <xs:enumeration value="hasSubstringIgnoreCase"/></xs:enumeration>
    <xs:enumeration value="in"/></xs:enumeration>
    <xs:enumeration value="countEquals"/></xs:enumeration>
    <xs:enumeration value="countLessThan"/></xs:enumeration>
    <xs:enumeration value="countLessOrEquals"/></xs:enumeration>
    <xs:enumeration value="countGreaterThan"/></xs:enumeration>
    <xs:enumeration value="countGreaterOrEquals"/></xs:enumeration>
    <xs:enumeration value="contains"/></xs:enumeration>
    <xs:enumeration value="storageEquals"/></xs:enumeration>
    <xs:enumeration value="typeEquals"/></xs:enumeration>
    <xs:enumeration value="directionEquals"/></xs:enumeration>
    <xs:enumeration value="isMeasureDiscrete"/></xs:enumeration>
    <xs:enumeration value="isMeasureContinuous"/></xs:enumeration>
    <xs:enumeration value="isMeasureTypeless"/></xs:enumeration>
    <xs:enumeration value="isMeasureUnknown"/></xs:enumeration>
    <xs:enumeration value="isStorageString"/></xs:enumeration>
    <xs:enumeration value="isStorageNumeric"/></xs:enumeration>
    <xs:enumeration value="isStorageDatetime"/></xs:enumeration>
    <xs:enumeration value="isStorageUnknown"/></xs:enumeration>
    <xs:enumeration value="isModelOutput"/></xs:enumeration>
    <xs:enumeration value="modelOutputRoleEquals"/></xs:enumeration>
    <xs:enumeration value="modelOutputTargetFieldEquals"/></xs:enumeration>
  </xs:attribute>
</xs:element>

```

```

    <xs:enumeration value="modelOutputHasValue"></xs:enumeration>
    <xs:enumeration value="modelOutputTagEquals"></xs:enumeration>
    <xs:enumeration value="enumRestriction"></xs:enumeration>
  </xs:attribute>
  <xs:attribute name="value" type="EVALUATED-STRING" use="optional"></xs:attribute>
  <xs:attribute name="listMode" use="optional" default="all">
    <xs:simpleType>
      <xs:restriction base="xs:string">
        <xs:enumeration value="all"></xs:enumeration>
        <xs:enumeration value="any"></xs:enumeration>
      </xs:restriction>
    </xs:simpleType>
  </xs:attribute>
</xs:element>

```

### Parent Eléments

[And Elément](#), [Command Elément](#), [Constraint Elément](#), [CreateDocument Elément](#),  
[CreateDocumentOutput Elément](#), [CreateInteractiveDocumentBuilder Elément](#),  
[CreateInteractiveModelBuilder Elément](#), [CreateModel Elément](#), [CreateModelApplier Elément](#),  
[CreateModelOutput Elément](#), [Enabled Elément](#), [ExpertSettings Elément](#), [Not Elément](#), [Option  
Elément](#), [Or Elément](#), [Run Elément](#), [Visible Elément](#)

## Constraint Elément

Table A-20  
Attributs pour Constraint

Attribut	Utiliser	Description	Valeurs valides
property	<b>obligatoire</b>		<i>string</i>
singleSelection	optional		<i>boolean</i>

### Représentation XML

```

<xs:element name="Constraint">
  <xs:sequence>
    <xs:group ref="CONDITION-EXPRESSION">
      <xs:choice>
        <xs:element ref="Condition"></xs:element>
        <xs:element ref="And"></xs:element>
        <xs:element ref="Or"></xs:element>
        <xs:element ref="Not"></xs:element>
      </xs:choice>
    </xs:group>
  </xs:sequence>
  <xs:attribute name="property" type="xs:string" use="required"></xs:attribute>
  <xs:attribute name="singleSelection" type="xs:boolean" use="optional" default="false"></xs:attribute>
</xs:element>

```

### Parent Eléments

[AutoModeling Elément](#)

**Enfant Eléments**

[And Elément](#), [Condition Elément](#), [Not Elément](#), [Or Elément](#)

**Constructors Elément****Représentation XML**

```
<xs:element name="Constructors">
  <xs:sequence minOccurs="0" maxOccurs="unbounded">
    <xs:choice>
      <xs:element ref="CreateModelOutput"></xs:element>
      <xs:element ref="CreateDocumentOutput"></xs:element>
      <xs:element ref="CreateInteractiveModelBuilder"></xs:element>
      <xs:element ref="CreateInteractiveDocumentBuilder"></xs:element>
      <xs:element ref="CreateModelApplier"></xs:element>
    </xs:choice>
  </xs:sequence>
</xs:element>
```

**Parent Eléments**

[DocumentOutput Elément](#), [Execution Elément](#), [InteractiveDocumentBuilder Elément](#),  
[InteractiveModelBuilder Elément](#), [ModelOutput Elément](#), [Node Elément](#)

**Enfant Eléments**

[CreateDocumentOutput Elément](#), [CreateInteractiveDocumentBuilder Elément](#),  
[CreateInteractiveModelBuilder Elément](#), [CreateModelApplier Elément](#), [CreateModelOutput Elément](#)

**Container Elément**

Table A-21  
Attributs pour Container

Attribut	Utiliser	Description	Valeurs valides
name	<b>obligatoire</b>		<i>string</i>
type	optional		<i>string</i>

**Représentation XML**

```
<xs:element name="Container">
  <xs:attribute name="name" type="xs:string" use="required"></xs:attribute>
  <xs:attribute name="type" type="xs:string" use="optional"></xs:attribute>
</xs:element>
```

**Parent Eléments**

[Containers Elément](#), [Containers Elément](#), [Containers Elément](#), [Containers Elément](#), [Containers Elément](#)

## ContainerFile Elément

Table A-22  
Attributs pour ContainerFile

Attribut	Utiliser	Description	Valeurs valides
container	optional		
containerType	optional		
path	<b>obligatoire</b>		

### Représentation XML

```
<xs:element name="ContainerFile" type="SERVER-CONTAINER-FILE">
  <xs:attribute name="path" type="EVALUATED-STRING" use="required"/></xs:attribute>
  <xs:attribute name="container" type="EVALUATED-STRING" use="optional"/></xs:attribute>
  <xs:attribute name="containerType" type="EVALUATED-STRING" use="optional"/></xs:attribute>
</xs:element>
```

### Parent Eléments

[InputFiles Elément](#), [OutputFiles Elément](#)

## ContainerTypes Elément

### Représentation XML

```
<xs:element name="ContainerTypes">
  <xs:sequence minOccurs="0" maxOccurs="unbounded">
    <xs:choice>
      <xs:element ref="DocumentType"/></xs:element>
      <xs:element ref="ModelType"/></xs:element>
    </xs:choice>
  </xs:sequence>
</xs:element>
```

### Parent Eléments

[CommonObjects Elément](#)

### Enfant Eléments

[DocumentType Elément](#), [ModelType Elément](#)

## Controls Elément

### Représentation XML

```
<xs:element name="Controls">
  <xs:sequence minOccurs="0" maxOccurs="unbounded">
    <xs:choice>
      <xs:element ref="Menu"/></xs:element>
      <xs:element ref="MenuItem"/></xs:element>
      <xs:element ref="ToolBarItem"/></xs:element>
    </xs:choice>
  </xs:sequence>
</xs:element>
```

```

    </xs:choice>
  </xs:sequence>
</xs:element>

```

### Parent Eléments

[UserInterface Elément](#)

### Enfant Eléments

[Menu Elément](#), [MenuItem Elément](#), [ToolBarItem Elément](#)

## CreateDocument Elément

Table A-23  
Attributs pour CreateDocument

Attribut	Utiliser	Description	Valeurs valides
sourceFile	<b>obligatoire</b>		<i>string</i>
target	<b>obligatoire</b>		<i>string</i>
type	optional		<i>string</i>

### Représentation XML

```

<xs:element name="CreateDocument">
  <xs:group ref="CONDITION-EXPRESSION" minOccurs="0">
    <xs:choice>
      <xs:element ref="Condition"></xs:element>
      <xs:element ref="And"></xs:element>
      <xs:element ref="Or"></xs:element>
      <xs:element ref="Not"></xs:element>
    </xs:choice>
  </xs:group>
  <xs:attribute name="sourceFile" type="xs:string" use="required"></xs:attribute>
  <xs:attribute name="target" type="xs:string" use="required"></xs:attribute>
  <xs:attribute name="type" type="xs:string" use="optional"></xs:attribute>
</xs:element>

```

### Parent Eléments

[CreateDocumentOutput Elément](#), [CreateInteractiveDocumentBuilder Elément](#),  
[CreateInteractiveModelBuilder Elément](#), [CreateModelApplier Elément](#), [CreateModelOutput Elément](#)

### Enfant Eléments

[And Elément](#), [Condition Elément](#), [Not Elément](#), [Or Elément](#)

### Eléments connexes Eléments

[CreateModel Elément](#)



## CreateDocumentOutput Élément

Table A-24  
Attributs pour CreateDocumentOutput

Attribut	Utiliser	Description	Valeurs valides
type	obligatoire		string

### Représentation XML

```
<xs:element name="CreateDocumentOutput">
  <xs:sequence>
    <xs:group ref="CONDITION-EXPRESSION" minOccurs="0">
      <xs:choice>
        <xs:element ref="Condition"></xs:element>
        <xs:element ref="And"></xs:element>
        <xs:element ref="Or"></xs:element>
        <xs:element ref="Not"></xs:element>
      </xs:choice>
    </xs:group>
    <xs:sequence minOccurs="0" maxOccurs="unbounded">
      <xs:choice>
        <xs:element ref="SetProperty"></xs:element>
        <xs:element ref="SetContainer"></xs:element>
        <xs:element ref="CreateModel"></xs:element>
        <xs:element ref="CreateDocument"></xs:element>
      </xs:choice>
    </xs:sequence>
  </xs:sequence>
  <xs:attribute name="type" type="xs:string" use="required"></xs:attribute>
</xs:element>
```

### Parent Éléments

[Constructors Élément](#)

### Enfant Éléments

[And Élément](#), [Condition Élément](#), [CreateDocument Élément](#), [CreateModel Élément](#), [Not Élément](#), [Or Élément](#), [SetContainer Élément](#), [SetProperty Élément](#)

### Éléments connexes Éléments

[CreateInteractiveDocumentBuilder Élément](#), [CreateInteractiveModelBuilder Élément](#), [CreateModelApplier Élément](#), [CreateModelOutput Élément](#)

## CreateInteractiveDocumentBuilder Élément

Table A-25  
Attributs pour CreateInteractiveDocumentBuilder

Attribut	Utiliser	Description	Valeurs valides
type	obligatoire		string

**Représentation XML**

```

<xs:element name="CreateInteractiveDocumentBuilder">
  <xs:sequence>
    <xs:group ref="CONDITION-EXPRESSION" minOccurs="0">
      <xs:choice>
        <xs:element ref="Condition"></xs:element>
        <xs:element ref="And"></xs:element>
        <xs:element ref="Or"></xs:element>
        <xs:element ref="Not"></xs:element>
      </xs:choice>
    </xs:group>
    <xs:sequence minOccurs="0" maxOccurs="unbounded">
      <xs:choice>
        <xs:element ref="SetProperty"></xs:element>
        <xs:element ref="SetContainer"></xs:element>
        <xs:element ref="CreateModel"></xs:element>
        <xs:element ref="CreateDocument"></xs:element>
      </xs:choice>
    </xs:sequence>
  </xs:sequence>
  <xs:attribute name="type" type="xs:string" use="required"></xs:attribute>
</xs:element>

```

**Parent Éléments**

[Constructors Élément](#)

**Enfant Éléments**

[And Élément](#), [Condition Élément](#), [CreateDocument Élément](#), [CreateModel Élément](#), [Not Élément](#), [Or Élément](#), [SetContainer Élément](#), [SetProperty Élément](#)

**Éléments connexes Éléments**

[CreateDocumentOutput Élément](#), [CreateInteractiveModelBuilder Élément](#), [CreateModelApplier Élément](#), [CreateModelOutput Élément](#)

**CreateInteractiveModelBuilder Élément**

Table A-26

Attributs pour CreateInteractiveModelBuilder

Attribut	Utiliser	Description	Valeurs valides
type	obligatoire		string

**Représentation XML**

```

<xs:element name="CreateInteractiveModelBuilder">
  <xs:sequence>
    <xs:group ref="CONDITION-EXPRESSION" minOccurs="0">
      <xs:choice>
        <xs:element ref="Condition"></xs:element>
        <xs:element ref="And"></xs:element>
        <xs:element ref="Or"></xs:element>
      </xs:choice>
    </xs:group>
  </xs:sequence>
</xs:element>

```

```

        <xs:element ref="Not"></xs:element>
      </xs:choice>
    </xs:group>
  </xs:sequence minOccurs="0" maxOccurs="unbounded">
    <xs:choice>
      <xs:element ref="SetProperty"></xs:element>
      <xs:element ref="SetContainer"></xs:element>
      <xs:element ref="CreateModel"></xs:element>
      <xs:element ref="CreateDocument"></xs:element>
    </xs:choice>
  </xs:sequence>
</xs:sequence>
<xs:attribute name="type" type="xs:string" use="required"></xs:attribute>
</xs:element>

```

### Parent Eléments

[Constructors Elément](#)

### Enfant Eléments

[And Elément](#), [Condition Elément](#), [CreateDocument Elément](#), [CreateModel Elément](#), [Not Elément](#), [Or Elément](#), [SetContainer Elément](#), [SetProperty Elément](#)

### Eléments connexes Eléments

[CreateDocumentOutput Elément](#), [CreateInteractiveDocumentBuilder Elément](#), [CreateModelApplier Elément](#), [CreateModelOutput Elément](#)

## CreateModel Elément

Table A-27  
Attributs pour CreateModel

Attribut	Utiliser	Description	Valeurs valides
signatureFile	optional		string
sourceFile	<b>obligatoire</b>		string
target	<b>obligatoire</b>		string
type	optional		string

### Représentation XML

```

<xs:element name="CreateModel">
  <xs:group ref="CONDITION-EXPRESSION" minOccurs="0">
    <xs:choice>
      <xs:element ref="Condition"></xs:element>
      <xs:element ref="And"></xs:element>
      <xs:element ref="Or"></xs:element>
      <xs:element ref="Not"></xs:element>
    </xs:choice>
  </xs:group>
  <xs:attribute name="sourceFile" type="xs:string" use="required"></xs:attribute>
  <xs:attribute name="target" type="xs:string" use="required"></xs:attribute>
  <xs:attribute name="type" type="xs:string" use="optional"></xs:attribute>

```

```

<xs:sequence>
  <xs:element name="ModelDetail" maxOccurs="unbounded"/></xs:element>
</xs:sequence>
<xs:attribute name="signatureFile" type="xs:string" use="optional"/></xs:attribute>
</xs:element>

```

**Parent Eléments**

[CreateDocumentOutput Elément](#), [CreateInteractiveDocumentBuilder Elément](#),  
[CreateInteractiveModelBuilder Elément](#), [CreateModelApplier Elément](#), [CreateModelOutput Elément](#)

**Enfant Eléments**

[And Elément](#), [Condition Elément](#), [ModelDetail Elément](#), [Not Elément](#), [Or Elément](#)

**Eléments connexes Eléments**

[CreateDocument Elément](#)

**ModelDetail Elément**

Table A-28  
*Attributs pour ModelDetail*

Attribut	Utiliser	Description	Valeurs valides
algorithm	<b>obligatoire</b>		<i>string</i>

**Représentation XML**

```

<xs:element name="ModelDetail" maxOccurs="unbounded">
  <xs:attribute name="algorithm" type="xs:string" use="required"/></xs:attribute>
</xs:element>

```

**Parent Eléments**

[CreateModel Elément](#)

**CreateModelApplier Elément**

Table A-29  
*Attributs pour CreateModelApplier*

Attribut	Utiliser	Description	Valeurs valides
type	<b>obligatoire</b>		<i>string</i>

**Représentation XML**

```

<xs:element name="CreateModelApplier">
  <xs:sequence>
    <xs:group ref="CONDITION-EXPRESSION" minOccurs="0">
      <xs:choice>
        <xs:element ref="Condition"/></xs:element>
        <xs:element ref="And"/></xs:element>
      </xs:choice>
    </xs:group>
  </xs:sequence>
</xs:element>

```

```

        <xs:element ref="Or"></xs:element>
        <xs:element ref="Not"></xs:element>
    </xs:choice>
</xs:group>
<xs:sequence minOccurs="0" maxOccurs="unbounded">
    <xs:choice>
        <xs:element ref="SetProperty"></xs:element>
        <xs:element ref="SetContainer"></xs:element>
        <xs:element ref="CreateModel"></xs:element>
        <xs:element ref="CreateDocument"></xs:element>
    </xs:choice>
</xs:sequence>
</xs:sequence>
<xs:attribute name="type" type="xs:string" use="required"></xs:attribute>
</xs:element>

```

### Parent Eléments

[Constructors Elément](#)

### Enfant Eléments

[And Elément](#), [Condition Elément](#), [CreateDocument Elément](#), [CreateModel Elément](#), [Not Elément](#), [Or Elément](#), [SetContainer Elément](#), [SetProperty Elément](#)

### Eléments connexes Eléments

[CreateDocumentOutput Elément](#), [CreateInteractiveDocumentBuilder Elément](#), [CreateInteractiveModelBuilder Elément](#), [CreateModelOutput Elément](#)

## CreateModelOutput Elément

Table A-30  
Attributs pour CreateModelOutput

Attribut	Utiliser	Description	Valeurs valides
type	obligatoire		string

### Représentation XML

```

<xs:element name="CreateModelOutput">
    <xs:sequence>
        <xs:group ref="CONDITION-EXPRESSION" minOccurs="0">
            <xs:choice>
                <xs:element ref="Condition"></xs:element>
                <xs:element ref="And"></xs:element>
                <xs:element ref="Or"></xs:element>
                <xs:element ref="Not"></xs:element>
            </xs:choice>
        </xs:group>
        <xs:sequence minOccurs="0" maxOccurs="unbounded">
            <xs:choice>
                <xs:element ref="SetProperty"></xs:element>
                <xs:element ref="SetContainer"></xs:element>
                <xs:element ref="CreateModel"></xs:element>
            </xs:choice>
        </xs:sequence>
    </xs:sequence>
</xs:element>

```

```

        <xs:element ref="CreateDocument"></xs:element>
      </xs:choice>
    </xs:sequence>
  </xs:sequence>
  <xs:attribute name="type" type="xs:string" use="required"></xs:attribute>
</xs:element>

```

### Parent Eléments

[Constructors Elément](#)

### Enfant Eléments

[And Elément](#), [Condition Elément](#), [CreateDocument Elément](#), [CreateModel Elément](#), [Not Elément](#), [Or Elément](#), [SetContainer Elément](#),  [SetProperty Elément](#)

### Eléments connexes Eléments

[CreateDocumentOutput Elément](#), [CreateInteractiveDocumentBuilder Elément](#), [CreateInteractiveModelBuilder Elément](#), [CreateModelApplier Elément](#)

## DatabaseConnectionValue Elément

A value specifying the details of a database connection.

Table A-31  
Attributs pour DatabaseConnectionValue

Attribut	Utiliser	Description	Valeurs valides
connectionType	<b>obligatoire</b>		<i>string</i>
datasourceName	<b>obligatoire</b>		<i>string</i>
password	<b>obligatoire</b>		<i>string</i>
userName	<b>obligatoire</b>		<i>string</i>

### Représentation XML

```

<xs:element name="DatabaseConnectionValue" type="DATABASE-CONNECTION-VALUE">
  <xs:attribute name="connectionType" type="xs:string" use="required"></xs:attribute>
  <xs:attribute name="datasourceName" type="xs:string" use="required"></xs:attribute>
  <xs:attribute name="userName" type="xs:string" use="required"></xs:attribute>
  <xs:attribute name="password" type="xs:string" use="required"></xs:attribute>
</xs:element>

```

### Parent Eléments

[ListValue Elément](#), [Attribute Elément](#), [ListValue Elément](#), [Parameter Elément](#), [Attribute Elément](#), [ListValue Elément](#)

## DataFile Élément

Table A-32  
Attributs pour DataFile

Attribut	Utiliser	Description	Valeurs valides
path	obligatoire		

### Représentation XML

```
<xs:element name="DataFile" type="SERVER-DATA-FILE">
  <xs:attribute name="path" type="EVALUATED-STRING" use="required"/></xs:attribute>
  <xs:choice>
    <xs:element ref="DelimitedDataFormat"/></xs:element>
  </xs:choice>
</xs:element>
```

### Parent Elements

[InputFiles Élément](#), [OutputFiles Élément](#)

### Enfant Elements

[DelimitedDataFormat Élément](#)

## DataFormat Élément

### Représentation XML

```
<xs:element name="DataFormat">
  <xs:group ref="DATA-FORMAT-TYPE">
    <xs:choice>
      <xs:element ref="DelimitedDataFormat"/></xs:element>
      <xs:element ref="SPSSDataFormat"/></xs:element>
    </xs:choice>
  </xs:group>
</xs:element>
```

### Parent Elements

[FileFormatType Élément](#)

### Enfant Elements

[DelimitedDataFormat Élément](#), [SPSSDataFormat Élément](#)

## DataModel Élément

The data model coming into or out of a node. An input/Output data model is a set of Fieldsl.

### Représentation XML

```
<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:element>
            <xs:choice minOccurs="0">
              <xs:element ref="Range"/></xs:element>
              <xs:element ref="Values"/></xs:element>
            </xs:choice>
            <xs:element ref="MissingValues"/></xs:element>
          </xs:sequence>
        </xs:group>
      </xs:element>
    </xs:sequence>
  </xs:element>
</xs:sequence>
</xs:element>

```

### Enfant Eléments

[FieldFormats Elément](#), [FieldGroups Elément](#), [Fields Elément](#)

### FieldFormats Elément

Defines the default field formats. Field formats are used when displaying values in output such as the general format (standard number, scientific or currency formats), number of decimal places to display, decimal separator etc. Currently field formats are only used for numeric fields although this may change in future versions.

Table A-33  
Attributs pour FieldFormats

Attribut	Utiliser	Description	Valeurs valides
count	facultatif		<i>nonNegativeInteger</i>
defaultNumberFormat	<b>obligatoire</b>		<i>string</i>



**Représentation XML**

```

<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>
  <xs:attribute name="count" type="xs:nonNegativeInteger"></xs:attribute>
</xs:element>

```

**Parent Eléments**[DataModel Elément](#)**Enfant Eléments**[NumberFormat Elément](#)**NumberFormat Elément**

Defines format information for a numeric field.

Table A-34

Attributs pour NumberFormat

Attribut	Utiliser	Description	Valeurs valides
decimalPlaces	<b>obligatoire</b>		<i>nonNegativeInteger</i>
decimalSymbol	<b>obligatoire</b>		<b>period</b> <b>comma</b>
formatType	<b>obligatoire</b>		<b>standard</b> <b>scientific</b> <b>currency</b>
groupingSymbol	<b>obligatoire</b>		<b>none</b> <b>period</b> <b>comma</b> <b>space</b>
name	<b>obligatoire</b>		<i>string</i>

**Représentation XML**

```

<xs:element name="NumberFormat" type="NUMBER-FORMAT-DECLARATION" minOccurs="0"
  maxOccurs="unbounded">
  <xs:attribute name="name" type="xs:string" use="required"></xs:attribute>
  <xs:attribute name="formatType" type="NUMBER-FORMAT-TYPE" use="required">
    <xs:enumeration value="standard"></xs:enumeration>
    <xs:enumeration value="scientific"></xs:enumeration>
    <xs:enumeration value="currency"></xs:enumeration>
  </xs:attribute>
  <xs:attribute name="decimalPlaces" type="xs:nonNegativeInteger" use="required"></xs:attribute>
  <xs:attribute name="decimalSymbol" type="DECIMAL-SYMBOL" use="required">
    <xs:enumeration value="period"></xs:enumeration>
    <xs:enumeration value="comma"></xs:enumeration>
  </xs:attribute>
  <xs:attribute name="groupingSymbol" type="NUMBER-GROUPING-SYMBOL" use="required">

```

```

    <xs:enumeration value="none"></xs:enumeration>
    <xs:enumeration value="period"></xs:enumeration>
    <xs:enumeration value="comma"></xs:enumeration>
    <xs:enumeration value="space"></xs:enumeration>
  </xs:attribute>
</xs:element>

```

### Parent Elements

[FieldFormats Élément](#)

### FieldGroups Élément

Defines the field groups. Field groups are used to associate related fields. For example, a survey question that asks a respondent to select which locations they have visited from a set of options will be represented as a set of flag fields. A field group may be used to identify which fields are associated with that survey question.

Table A-35  
Attributs pour FieldGroups

Attribut	Utiliser	Description	Valeurs valides
count	facultatif		<i>nonNegativeInteger</i>

### Représentation XML

```

<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:attribute>
</xs:element>

```

### Parent Elements

[DataModel Élément](#)

### Enfant Elements

[FieldGroup Élément](#)

### FieldGroup Élément

Defines a field group. A field group consists of a list of field names and information about the field group such as the group name and optional label, type of group and for multi-dichotomy groups, the counted value i.e. the value which represents “true”.

Table A-36  
Attributs pour FieldGroup

Attribut	Utiliser	Description	Valeurs valides
count	facultatif		<i>nonNegativeInteger</i>
countedValue	facultatif		<i>string</i>
displayLabel	facultatif		<i>string</i>
groupType	<b>obligatoire</b>		<b>fieldGroup</b> <b>multiCategorySet</b> <b>multiDichotomySet</b>
name	<b>obligatoire</b>		

### Représentation XML

```
<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>
  <xs:attribute name="displayLabel" type="xs:string"></xs:attribute>
  <xs:attribute name="groupType" type="FIELD-GROUP-TYPE" use="required">
    <xs:enumeration value="fieldGroup"></xs:enumeration>
    <xs:enumeration value="multiCategorySet"></xs:enumeration>
    <xs:enumeration value="multiDichotomySet"></xs:enumeration>
  </xs:attribute>
  <xs:attribute name="countedValue" type="xs:string"></xs:attribute>
  <xs:attribute name="count" type="xs:nonNegativeInteger"></xs:attribute>
</xs:element>
```

### Parent Eléments

[FieldGroups Elément](#)

### Enfant Eléments

[FieldName Elément](#)

### FieldName Elément

Table A-37  
Attributs pour FieldName

Attribut	Utiliser	Description	Valeurs valides
name	<b>obligatoire</b>		

### Représentation XML

```
<xs:element name="FieldName">
  <xs:attribute name="name" type="FIELD-NAME" use="required"></xs:attribute>
</xs:element>
```

**Parent Eléments**[FieldGroup Elément](#)**Fields Elément**

Table A-38

Attributs pour Fields

Attribut	Utiliser	Description	Valeurs valides
count	facultatif		<i>nonNegativeInteger</i>

**Représentation XML**

```

<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:element>
          <xs:choice minOccurs="0">
            <xs:element ref="Range"></xs:element>
            <xs:element ref="Values"></xs:element>
          </xs:choice>
          <xs:element ref="MissingValues"></xs:element>
        </xs:sequence>
      </xs:group>
    </xs:element>
  </xs:sequence>
  <xs:attribute name="count" type="xs:nonNegativeInteger"/></xs:attribute>
</xs:element>

```

**Parent Eléments**[DataModel Elément](#)**Enfant Eléments**[Field Elément](#)**Field Elément**

Table A-39

Attributs pour Field

Attribut	Utiliser	Description	Valeurs valides
direction	facultatif		<b>in</b> <b>out</b> <b>both</b> <b>none</b> <b>partition</b>
displayLabel	facultatif		<i>string</i>
name	<b>obligatoire</b>		<i>string</i>

Attribut	Utiliser	Description	Valeurs valides
storage	facultatif		unknown integer real string date time timestamp
type	facultatif		auto range discrete set orderedSet flag typeless

### Représentation XML

```

<xs:element name="Field" type="FIELD" minOccurs="0" maxOccurs="unbounded">
  <xs:group ref="FIELD-CONTENT">
    <xs:sequence>
      <xs:element ref="DisplayLabel"></xs:element>
      <xs:choice minOccurs="0">
        <xs:element ref="Range"></xs:element>
        <xs:element ref="Values"></xs:element>
      </xs:choice>
      <xs:element ref="MissingValues"></xs:element>
    </xs:sequence>
  </xs:group>
  <xs:attribute name="name" type="xs:string" use="required"></xs:attribute>
  <xs:attribute name="type" type="FIELD-TYPE" default="auto">
    <xs:enumeration value="auto"></xs:enumeration>
    <xs:enumeration value="range"></xs:enumeration>
    <xs:enumeration value="discrete"></xs:enumeration>
    <xs:enumeration value="set"></xs:enumeration>
    <xs:enumeration value="orderedSet"></xs:enumeration>
    <xs:enumeration value="flag"></xs:enumeration>
    <xs:enumeration value="typeless"></xs:enumeration>
  </xs:attribute>
  <xs:attribute name="storage" type="FIELD-STORAGE" default="unknown">
    <xs:enumeration value="unknown"></xs:enumeration>
    <xs:enumeration value="integer"></xs:enumeration>
    <xs:enumeration value="real"></xs:enumeration>
    <xs:enumeration value="string"></xs:enumeration>
    <xs:enumeration value="date"></xs:enumeration>
    <xs:enumeration value="time"></xs:enumeration>
    <xs:enumeration value="timestamp"></xs:enumeration>
  </xs:attribute>
  <xs:attribute name="direction" type="FIELD-DIRECTION" default="in">
    <xs:enumeration value="in"></xs:enumeration>
    <xs:enumeration value="out"></xs:enumeration>
    <xs:enumeration value="both"></xs:enumeration>
    <xs:enumeration value="none"></xs:enumeration>
    <xs:enumeration value="partition"></xs:enumeration>
  </xs:attribute>
  <xs:attribute name="displayLabel" type="xs:string"></xs:attribute>
</xs:element>

```

**Parent Eléments**[Fields Elément](#)**Enfant Eléments**[DisplayLabel Elément](#), [MissingValues Elément](#), [Range Elément](#), [Values Elément](#)**DBConnectionChooserControl Elément**

Table A-40  
Attributs pour DBConnectionChooserControl

Attribut	Utiliser	Description	Valeurs valides
description	optional		string
descriptionKey	optional		string
label	optional		string
labelAbove	optional		boolean
labelKey	optional		string
labelWidth	optional		positiveInteger
mnemonic	optional		string
mnemonicKey	optional		string
property	<b>obligatoire</b>		string
showLabel	optional		boolean

**Représentation XML**

```

<xs:element name="DBConnectionChooserControl">
  <xs:sequence>
    <xs:choice>
      <xs:element ref="Layout" minOccurs="0" maxOccurs="1"></xs:element>
      <xs:element ref="Enabled" minOccurs="0" maxOccurs="1"></xs:element>
      <xs:element ref="Visible" minOccurs="0" maxOccurs="1"></xs:element>
    </xs:choice>
  </xs:sequence>
  <xs:attribute name="property" type="xs:string" use="required"></xs:attribute>
  <xs:attribute name="showLabel" type="xs:boolean" use="optional" default="true"></xs:attribute>
  <xs:attribute name="label" type="xs:string" use="optional"></xs:attribute>
  <xs:attribute name="labelKey" type="xs:string" use="optional"></xs:attribute>
  <xs:attribute name="mnemonic" type="xs:string" use="optional"></xs:attribute>
  <xs:attribute name="mnemonicKey" type="xs:string" use="optional"></xs:attribute>
  <xs:attribute name="labelWidth" type="xs:positiveInteger" use="optional" default="1"></xs:attribute>
  <xs:attribute name="labelAbove" type="xs:boolean" use="optional" default="false"></xs:attribute>
  <xs:attribute name="description" type="xs:string" use="optional"></xs:attribute>
  <xs:attribute name="descriptionKey" type="xs:string" use="optional"></xs:attribute>
</xs:element>

```

**Parent Eléments**[PropertiesPanel Elément](#), [PropertiesSubPanel Elément](#)

**Enfant Eléments**

[Enabled Elément](#), [Layout Elément](#), [Visible Elément](#)

**Eléments connexes Eléments**

[CheckBoxControl Elément](#), [CheckBoxGroupControl Elément](#), [ClientDirectoryChooserControl Elément](#), [ClientFileChooserControl Elément](#), [DBTableChooserControl Elément](#), [MultiFieldChooserControl Elément](#), [PasswordBoxControl Elément](#), [PropertyControl Elément](#), [RadioButtonGroupControl Elément](#), [ServerDirectoryChooserControl Elément](#), [ServerFileChooserControl Elément](#), [SingleFieldChooserControl Elément](#), [SingleFieldValueChooserControl Elément](#), [SpinnerControl Elément](#), [TableControl Elément](#), [TextAreaControl Elément](#), [TextBoxControl Elément](#)

**DBTableChooserControl Elément**

Table A-41

Attributs pour DBTableChooserControl

Attribut	Utiliser	Description	Valeurs valides
connectionProperty	<b>obligatoire</b>		<i>string</i>
description	optional		<i>string</i>
descriptionKey	optional		<i>string</i>
label	optional		<i>string</i>
labelAbove	optional		<i>boolean</i>
labelKey	optional		<i>string</i>
labelWidth	optional		<i>positiveInteger</i>
mnemonic	optional		<i>string</i>
mnemonicKey	optional		<i>string</i>
property	<b>obligatoire</b>		<i>string</i>
showLabel	optional		<i>boolean</i>

**Représentation XML**

```
<xs:element name="DBTableChooserControl">
  <xs:sequence>
    <xs:choice>
      <xs:element ref="Layout" minOccurs="0" maxOccurs="1"/></xs:element>
      <xs:element ref="Enabled" minOccurs="0" maxOccurs="1"/></xs:element>
      <xs:element ref="Visible" minOccurs="0" maxOccurs="1"/></xs:element>
    </xs:choice>
  </xs:sequence>
  <xs:attribute name="property" type="xs:string" use="required"/></xs:attribute>
  <xs:attribute name="showLabel" type="xs:boolean" use="optional" default="true"/></xs:attribute>
  <xs:attribute name="label" type="xs:string" use="optional"/></xs:attribute>
  <xs:attribute name="labelKey" type="xs:string" use="optional"/></xs:attribute>
  <xs:attribute name="mnemonic" type="xs:string" use="optional"/></xs:attribute>
  <xs:attribute name="mnemonicKey" type="xs:string" use="optional"/></xs:attribute>
  <xs:attribute name="labelWidth" type="xs:positiveInteger" use="optional" default="1"/></xs:attribute>
  <xs:attribute name="labelAbove" type="xs:boolean" use="optional" default="false"/></xs:attribute>
  <xs:attribute name="description" type="xs:string" use="optional"/></xs:attribute>
  <xs:attribute name="descriptionKey" type="xs:string" use="optional"/></xs:attribute>
</xs:element>
```

```
<xs:attribute name="connectionProperty" type="xs:string" use="required"/></xs:attribute>
</xs:element>
```

### **Parent Eléments**

PropertiesPanel Elément, PropertiesSubPanel Elément

### **Enfant Eléments**

Enabled Elément, Layout Elément, Visible Elément

### **Eléments connexes Eléments**

CheckBoxControl Elément, CheckBoxGroupControl Elément, ClientDirectoryChooserControl Elément, ClientFileChooserControl Elément, DBConnectionChooserControl Elément, MultiFieldChooserControl Elément, PasswordBoxControl Elément, PropertyControl Elément, RadioButtonGroupControl Elément, ServerDirectoryChooserControl Elément, ServerFileChooserControl Elément, SingleFieldChooserControl Elément, SingleFieldValueChooserControl Elément, SpinnerControl Elément, TableControl Elément, TextAreaControl Elément, TextBoxControl Elément

## **DefaultValue Elément**

### **Représentation XML**

```
<xs:element name="DefaultValue">
  <xs:choice>
    <xs:element name="ServerTempFile"/></xs:element>
    <xs:element name="ServerTempDir"/></xs:element>
    <xs:element name="Identifiant"/></xs:element>
  </xs:choice>
</xs:element>
```

### **Parent Eléments**

Property Elément, PropertyType Elément

### **Enfant Eléments**

Identifiant Elément, ServerTempDir Elément, ServerTempFile Elément

## **ServerTempFile Elément**

Table A-42  
Attributs pour ServerTempFile

Attribut	Utiliser	Description	Valeurs valides
basename	obligatoire		

### **Représentation XML**

```
<xs:element name="ServerTempFile">
```



```
<xs:attribute name="basename" type="EVALUATED-STRING" use="required"></xs:attribute>
</xs:element>
```

### **Parent Eléments**

[DefaultValue Elément](#)

### **ServerTempDir Elément**

Table A-43

Attributs pour ServerTempDir

Attribut	Utiliser	Description	Valeurs valides
basename	<b>obligatoire</b>		

### **Représentation XML**

```
<xs:element name="ServerTempDir">
  <xs:attribute name="basename" type="EVALUATED-STRING" use="required"></xs:attribute>
</xs:element>
```

### **Parent Eléments**

[DefaultValue Elément](#)

### **Identifieur Elément**

Table A-44

Attributs pour Identifieur

Attribut	Utiliser	Description	Valeurs valides
basename	<b>obligatoire</b>		

### **Représentation XML**

```
<xs:element name="Identifieur">
  <xs:attribute name="basename" type="EVALUATED-STRING" use="required"></xs:attribute>
</xs:element>
```

### **Parent Eléments**

[DefaultValue Elément](#)

## DelimitedDataFormat Élément

Table A-45  
Attributs pour DelimitedDataFormat

Attribut	Utiliser	Description	Valeurs valides
delimiter	optional		<b>tab</b> <b>comma</b> <b>semicolon</b> <b>colon</b> <b>verticalBar</b> <b>other</b>
eol	optional		<b>cr</b> <b>crlf</b> <b>lf</b> <b>other</b>
includeFieldNames	optional		<i>boolean</i>
otherDelimiter	optional		<i>string</i>
otherEol	optional		<i>string</i>
quoteStrings	optional		<i>boolean</i>
stringQuote	optional		<i>string</i>

### Représentation XML

```

<xs:element name="DelimitedDataFormat">
  <xs:attribute name="delimiter" use="optional" default="tab">
    <xs:simpleType>
      <xs:restriction base="xs:string">
        <xs:enumeration value="tab"></xs:enumeration>
        <xs:enumeration value="comma"></xs:enumeration>
        <xs:enumeration value="semicolon"></xs:enumeration>
        <xs:enumeration value="colon"></xs:enumeration>
        <xs:enumeration value="verticalBar"></xs:enumeration>
        <xs:enumeration value="other"></xs:enumeration>
      </xs:restriction>
    </xs:simpleType>
  </xs:attribute>
  <xs:attribute name="otherDelimiter" type="xs:string" use="optional"></xs:attribute>
  <xs:attribute name="eol" use="optional" default="cr">
    <xs:simpleType>
      <xs:restriction base="xs:string">
        <xs:enumeration value="cr"></xs:enumeration>
        <xs:enumeration value="crlf"></xs:enumeration>
        <xs:enumeration value="lf"></xs:enumeration>
        <xs:enumeration value="other"></xs:enumeration>
      </xs:restriction>
    </xs:simpleType>
  </xs:attribute>
  <xs:attribute name="otherEol" type="xs:string" use="optional"></xs:attribute>
  <xs:attribute name="includeFieldNames" type="xs:boolean" use="optional" default="true"></xs:attribute>
  <xs:attribute name="quoteStrings" type="xs:boolean" use="optional" default="false"></xs:attribute>
  <xs:attribute name="stringQuote" type="xs:string" use="optional" default="""></xs:attribute>
</xs:element>

```

**Parent Eléments**

[DataFile Elément](#), [DataFormat Elément](#)

**DisplayLabel Elément**

A display label for a field or value in a specified language. The displayLabel attribute can be used where there is no label for a particular language.

Table A-46

Attributs pour DisplayLabel

Attribut	Utiliser	Description	Valeurs valides
lang	facultatif		NMTOKEN
value	obligatoire		string

**Représentation XML**

```
<xs:element name="DisplayLabel" type="DISPLAY-LABEL" minOccurs="0" maxOccurs="unbounded">
  <xs:attribute name="value" type="xs:string" use="required"></xs:attribute>
  <xs:attribute name="lang" type="xs:NMTOKEN" default="en"></xs:attribute>
</xs:element>
```

**Parent Eléments**

[Field Elément](#)

**DocumentBuilder Elément****Représentation XML**

```
<xs:element name="DocumentBuilder">
  <xs:sequence>
    <xs:element name="DocumentGeneration"></xs:element>
  </xs:sequence>
</xs:element>
```

**Parent Eléments**

[Node Elément](#)

**Enfant Eléments**

[DocumentGeneration Elément](#)

**DocumentGeneration Elément**

Table A-47

Attributs pour DocumentGeneration

Attribut	Utiliser	Description	Valeurs valides
controlsId	obligatoire		string

**Représentation XML**

```
<xs:element name="DocumentGeneration">
  <xs:attribute name="controlslid" type="xs:string" use="required"/></xs:attribute>
</xs:element>
```

**Parent Eléments**

[DocumentBuilder Elément](#)

**Eléments connexes Eléments**

[ModelingFields Elément](#), [ModelGeneration Elément](#), [ModelEvaluation Elément](#)

**DocumentOutput Elément**

Table A-48

Attributs pour DocumentOutput

Attribut	Utiliser	Description	Valeurs valides
deprecatedScriptNames	optional		<i>string</i>
id	<b>obligatoire</b>		<i>string</i>
scriptName	optional		<i>string</i>

**Représentation XML**

```
<xs:element name="DocumentOutput">
  <xs:sequence maxOccurs="unbounded">
    <xs:choice maxOccurs="unbounded">
      <xs:element ref="Properties"/></xs:element>
      <xs:element name="Containers" minOccurs="0">
        <xs:sequence maxOccurs="unbounded">
          <xs:element ref="Container"/></xs:element>
        </xs:sequence>
      </xs:element>
      <xs:element ref="UserInterface"/></xs:element>
      <xs:element ref="Constructors" minOccurs="0"/></xs:element>
      <xs:element ref="ModelProvider" minOccurs="0"/></xs:element>
    </xs:choice>
  </xs:sequence>
  <xs:attribute name="id" type="xs:string" use="required"/></xs:attribute>
  <xs:attribute name="scriptName" type="xs:string" use="optional"/></xs:attribute>
  <xs:attribute name="deprecatedScriptNames" type="xs:string" use="optional"/></xs:attribute>
</xs:element>
```

**Parent Eléments**

[Extension Elément](#)

**Enfant Eléments**

[Constructors Elément](#), [Containers Elément](#), [ModelProvider Elément](#), [Properties Elément](#), [UserInterface Elément](#)

**Eléments connexes Eléments**

[InteractiveDocumentBuilder Elément](#), [InteractiveModelBuilder Elément](#), [ModelOutput Elément](#), [Node Elément](#)

**Containers Elément****Représentation XML**

```
<xs:element name="Containers" minOccurs="0">
  <xs:sequence maxOccurs="unbounded">
    <xs:element ref="Container"></xs:element>
  </xs:sequence>
</xs:element>
```

**Parent Eléments**

[DocumentOutput Elément](#)

**Enfant Eléments**

[Container Elément](#)

**DocumentType Elément**

Défines a new document type

Table A-49

Attributs pour *DocumentType*

Attribut	Utiliser	Description	Valeurs valides
format	<b>obligatoire</b>		<b>utf8</b> <b>binary</b>
id	<b>obligatoire</b>		<i>string</i>
type	optional		<b>unknown</b> <b>rowSet</b> <b>report</b> <b>graph</b>

**Représentation XML**

```
<xs:element name="DocumentType">
  <xs:attribute name="id" type="xs:string" use="required"></xs:attribute>
  <xs:attribute name="format" use="required">
    <xs:simpleType>
      <xs:restriction base="xs:string">
        <xs:enumeration value="utf8"></xs:enumeration>
        <xs:enumeration value="binary"></xs:enumeration>
      </xs:restriction>
    </xs:simpleType>
  </xs:attribute>
  <xs:attribute name="type" type="DOCUMENT-TYPE" use="optional">
    <xs:enumeration value="unknown"></xs:enumeration>
    <xs:enumeration value="rowSet"></xs:enumeration>
```

```

    <xs:enumeration value="report"></xs:enumeration>
    <xs:enumeration value="graph"></xs:enumeration>
  </xs:attribute>
</xs:element>

```

### **Parent Eléments**

[ContainerTypes Elément](#)

### **Eléments connexes Eléments**

[ModelType Elément](#)

## **Enabled Elément**

### **Représentation XML**

```

<xs:element name="Enabled">
  <xs:sequence>
    <xs:group ref="CONDITION-EXPRESSION" minOccurs="0">
      <xs:choice>
        <xs:element ref="Condition"></xs:element>
        <xs:element ref="And"></xs:element>
        <xs:element ref="Or"></xs:element>
        <xs:element ref="Not"></xs:element>
      </xs:choice>
    </xs:group>
  </xs:sequence>
</xs:element>

```

### **Parent Eléments**

[ActionButton Elément](#), [CheckBoxControl Elément](#), [CheckBoxGroupControl Elément](#), [ClientDirectoryChooserControl Elément](#), [ClientFileChooserControl Elément](#), [ComboBoxControl Elément](#), [DBConnectionChooserControl Elément](#), [DBTableChooserControl Elément](#), [ExtensionObjectPanel Elément](#), [ModelViewerPanel Elément](#), [MultiFieldChooserControl Elément](#), [MultiItemChooserControl Elément](#), [PasswordBoxControl Elément](#), [PropertiesPanel Elément](#), [PropertiesSubPanel Elément](#), [PropertyControl Elément](#), [RadioButtonGroupControl Elément](#), [SelectorPanel Elément](#), [ServerDirectoryChooserControl Elément](#), [ServerFileChooserControl Elément](#), [SingleFieldChooserControl Elément](#), [SingleFieldValueChooserControl Elément](#), [SingleItemChooserControl Elément](#), [SpinnerControl Elément](#), [StaticText Elément](#), [SystemControls Elément](#), [TabbedPanel Elément](#), [TableControl Elément](#), [TextAreaControl Elément](#), [TextBoxControl Elément](#), [TextBrowserPanel Elément](#), [ItemChooserControl Type](#)

### **Enfant Eléments**

[And Elément](#), [Condition Elément](#), [Not Elément](#), [Or Elément](#)

## Enumeration Élément

### Représentation XML

```
<xs:element name="Enumeration">
  <xs:sequence>
    <xs:element name="Enum" maxOccurs="unbounded"></xs:element>
  </xs:sequence>
</xs:element>
```

### Parent Elements

[PropertyType Élément](#)

### Enfant Elements

[Enum Élément](#)

## Enum Élément

Table A-50  
Attributs pour Enum

Attribut	Utiliser	Description	Valeurs valides
description	optional		<i>string</i>
descriptionKey	optional		<i>string</i>
imagePath	optional		<i>string</i>
imagePathKey	optional		<i>string</i>
label	<b>obligatoire</b>		<i>string</i>
labelKey	optional		<i>string</i>
value	<b>obligatoire</b>		<i>string</i>

### Représentation XML

```
<xs:element name="Enum" maxOccurs="unbounded">
  <xs:attribute name="value" type="xs:string" use="required"></xs:attribute>
  <xs:attribute name="label" type="xs:string" use="required"></xs:attribute>
  <xs:attribute name="labelKey" type="xs:string" use="optional"></xs:attribute>
  <xs:attribute name="description" type="xs:string" use="optional"></xs:attribute>
  <xs:attribute name="descriptionKey" type="xs:string" use="optional"></xs:attribute>
  <xs:attribute name="imagePath" type="xs:string" use="optional"></xs:attribute>
  <xs:attribute name="imagePathKey" type="xs:string" use="optional"></xs:attribute>
</xs:element>
```

### Parent Elements

[Enumeration Élément](#)

## ErrorDetail Élément

Supplementary information about an error or other condition.

**Représentation XML**

```

<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:element>
      </xs:sequence>
    </xs:element>
  </xs:sequence>
</xs:element>

```

**Enfant Eléments**[Diagnostic Elément](#)**Diagnostic Elément**

Table A-51  
Attributs pour Diagnostic

Attribut	Utiliser	Description	Valeurs valides
code	<b>obligatoire</b>		<i>integer</i>
severity	facultatif		<b>unknown information warning error fatal</b>
source	facultatif		<i>string</i>
subCode	facultatif		<i>integer</i>

**Représentation XML**

```

<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:element>
  </xs:sequence>
  <xs:attribute name="code" type="xs:integer" use="required"/></xs:attribute>
  <xs:attribute name="subCode" type="xs:integer" default="0"/></xs:attribute>
  <xs:attribute name="severity" type="DIAGNOSTIC-SEVERITY" default="error">
    <xs:enumeration value="unknown"/></xs:enumeration>
    <xs:enumeration value="information"/></xs:enumeration>
    <xs:enumeration value="warning"/></xs:enumeration>
    <xs:enumeration value="error"/></xs:enumeration>
    <xs:enumeration value="fatal"/></xs:enumeration>
  </xs:attribute>
  <xs:attribute name="source" type="xs:string"/></xs:attribute>
</xs:element>

```

**Parent Eléments**[ErrorDetail Elément](#)



**Enfant Eléments**

[Message Elément](#), [Parameter Elément](#)

**Message Elément**

Table A-52  
Attributs pour Message

Attribut	Utiliser	Description	Valeurs valides
lang	facultatif		NMTOKEN

**Représentation XML**

```
<xs:element name="Message" type="DIAGNOSTIC-MESSAGE" minOccurs="0">
  <xs:attribute name="lang" type="xs:NMTOKEN"/></xs:attribute>
</xs:element>
```

**Parent Eléments**

[Diagnostic Elément](#)

**Parameter Elément****Représentation XML**

```
<xs:element name="Parameter" type="xs:string" minOccurs="0" maxOccurs="unbounded"/></xs:element>
```

**Parent Eléments**

[Diagnostic Elément](#)

**Executable Elément****Représentation XML**

```
<xs:element name="Executable">
  <xs:sequence>
    <xs:element ref="Run" maxOccurs="unbounded"/></xs:element>
  </xs:sequence>
</xs:element>
```

**Parent Eléments**

[Execution Elément](#)

**Enfant Eléments**

[Run Elément](#)

## Execution Élément

### Représentation XML

```
<xs:element name="Execution">
  <xs:sequence>
    <xs:element ref="Properties" minOccurs="0"></xs:element>
    <xs:element ref="InputFiles"></xs:element>
    <xs:element ref="OutputFiles"></xs:element>
    <xs:choice>
      <xs:element ref="Executable"></xs:element>
      <xs:element ref="Module"></xs:element>
    </xs:choice>
    <xs:element ref="Constructors" minOccurs="0"></xs:element>
  </xs:sequence>
</xs:element>
```

### Parent Elements

[Node Élément](#)

### Enfant Elements

[Constructors Élément](#), [Executable Élément](#), [InputFiles Élément](#), [Module Élément](#), [OutputFiles Élément](#), [Properties Élément](#)

## Extension Élément

Defines the top-level extension container.

Table A-53

*Attributs pour Extension*

Attribut	Utiliser	Description	Valeurs valides
debug	optional		<i>boolean</i>
version	<b>obligatoire</b>		<i>string</i>

### Représentation XML

```
<xs:element name="Extension">
  <xs:sequence>
    <xs:element ref="ExtensionDetails"></xs:element>
    <xs:element ref="Resources"></xs:element>
    <xs:element ref="License" minOccurs="0"></xs:element>
    <xs:element ref="CommonObjects"></xs:element>
    <xs:element ref="UserInterface" minOccurs="0"></xs:element>
    <xs:sequence minOccurs="0" maxOccurs="unbounded">
      <xs:choice>
        <xs:element ref="Node"></xs:element>
        <xs:element ref="ModelOutput"></xs:element>
        <xs:element ref="DocumentOutput"></xs:element>
        <xs:element ref="InteractiveModelBuilder"></xs:element>
        <xs:element ref="InteractiveDocumentBuilder"></xs:element>
      </xs:choice>
    </xs:sequence>
  </xs:sequence>
```

```

</xs:sequence>
<xs:attribute name="version" type="xs:string" use="required"></xs:attribute>
<xs:attribute name="debug" type="xs:boolean" use="optional" default="false"></xs:attribute>
</xs:element>

```

### Enfant Eléments

[CommonObjects Elément](#), [DocumentOutput Elément](#), [ExtensionDetails Elément](#), [InteractiveDocumentBuilder Elément](#), [InteractiveModelBuilder Elément](#), [License Elément](#), [ModelOutput Elément](#), [Node Elément](#), [Resources Elément](#), [UserInterface Elément](#)

## ExtensionDetails Elément

Defines information about the extension such as the extension id, the extension provider and version information.

Table A-54  
Attributs pour ExtensionDetails

Attribut	Utiliser	Description	Valeurs valides
copyright	optional		string
description	optional		string
id	<b>obligatoire</b>		string
label	<b>obligatoire</b>		string
provider	optional		string
providerTag	<b>obligatoire</b>		string
version	facultatif		string

### Représentation XML

```

<xs:element name="ExtensionDetails">
  <xs:attribute name="providerTag" type="xs:string" use="required"></xs:attribute>
  <xs:attribute name="id" type="xs:string" use="required"></xs:attribute>
  <xs:attribute name="label" type="xs:string" use="required"></xs:attribute>
  <xs:attribute name="version" type="xs:string"></xs:attribute>
  <xs:attribute name="provider" type="xs:string" use="optional" default="(unknown)"></xs:attribute>
  <xs:attribute name="copyright" type="xs:string" use="optional"></xs:attribute>
  <xs:attribute name="description" type="xs:string" use="optional"></xs:attribute>
</xs:element>

```

### Parent Eléments

[Extension Elément](#)

## ExtensionObjectPanel Elément

Table A-55  
Attributs pour ExtensionObjectPanel

Attribut	Utiliser	Description	Valeurs valides
id	optional		string
panelClass	<b>obligatoire</b>		string

**Représentation XML**

```

<xs:element name="ExtensionObjectPanel">
  <xs:sequence>
    <xs:choice>
      <xs:element ref="Layout" minOccurs="0" maxOccurs="1"></xs:element>
      <xs:element ref="Enabled" minOccurs="0" maxOccurs="1"></xs:element>
      <xs:element ref="Visible" minOccurs="0" maxOccurs="1"></xs:element>
    </xs:choice>
  </xs:sequence>
  <xs:attribute name="panelClass" type="xs:string" use="required"></xs:attribute>
  <xs:attribute name="id" type="xs:string" use="optional"></xs:attribute>
</xs:element>

```

**Parent Eléments**

[PropertiesPanel Élément](#), [PropertiesSubPanel Élément](#), [Tab Élément](#)

**Enfant Eléments**

[Enabled Élément](#), [Layout Élément](#), [Visible Élément](#)

**Eléments connexes Eléments**

[ActionButton Élément](#), [ComboBoxControl Élément](#), [ModelViewerPanel Élément](#), [SelectorPanel Élément](#), [StaticText Élément](#), [SystemControls Élément](#), [TabbedPanel Élément](#), [TextBrowserPanel Élément](#)

**Field Élément**

Table A-56  
Attributs pour Field

Attribut	Utiliser	Description	Valeurs valides
direction	facultatif		<b>in</b> <b>out</b> <b>both</b> <b>none</b> <b>partition</b>
label	facultatif		<i>string</i>
name	<b>obligatoire</b>		
storage	facultatif		<b>unknown</b> <b>integer</b> <b>real</b> <b>string</b> <b>date</b> <b>time</b> <b>timestamp</b>
type	facultatif		<b>auto</b> <b>range</b> <b>discrete</b> <b>set</b> <b>orderedSet</b> <b>flag</b> <b>typeless</b>

**Représentation XML**

```

<xs:element name="Field" type="FIELD-DECLARATION">
  <xs:sequence>
    <xs:choice>
      <xs:element ref="Range" minOccurs="0"></xs:element>
      <xs:element ref="Values" minOccurs="0"></xs:element>
      <xs:element ref="NumericInfo" minOccurs="0"></xs:element>
      <xs:element name="MissingValues" minOccurs="0">
        <xs:sequence>
          <xs:element ref="Values" minOccurs="0" maxOccurs="unbounded"></xs:element>
          <xs:element ref="Range" minOccurs="0"></xs:element>
        </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>
  <xs:attribute name="storage" type="FIELD-STORAGE">
    <xs:enumeration value="unknown"></xs:enumeration>
    <xs:enumeration value="integer"></xs:enumeration>
    <xs:enumeration value="real"></xs:enumeration>
    <xs:enumeration value="string"></xs:enumeration>
    <xs:enumeration value="date"></xs:enumeration>
    <xs:enumeration value="time"></xs:enumeration>
    <xs:enumeration value="timestamp"></xs:enumeration>
  </xs:attribute>
  <xs:attribute name="type" type="FIELD-TYPE">
    <xs:enumeration value="auto"></xs:enumeration>
    <xs:enumeration value="range"></xs:enumeration>
    <xs:enumeration value="discrete"></xs:enumeration>
    <xs:enumeration value="set"></xs:enumeration>
    <xs:enumeration value="orderedSet"></xs:enumeration>
    <xs:enumeration value="flag"></xs:enumeration>
    <xs:enumeration value="typeless"></xs:enumeration>
  </xs:attribute>
  <xs:attribute name="direction" type="FIELD-DIRECTION">
    <xs:enumeration value="in"></xs:enumeration>
    <xs:enumeration value="out"></xs:enumeration>
    <xs:enumeration value="both"></xs:enumeration>
    <xs:enumeration value="none"></xs:enumeration>
    <xs:enumeration value="partition"></xs:enumeration>
  </xs:attribute>
  <xs:attribute name="label" type="xs:string"></xs:attribute>
</xs:element>

```

**Enfant Eléments**

[MissingValues Élément](#), [ModelField Élément](#), [NumericInfo Élément](#), [Range Élément](#), [Values Élément](#)

## MissingValues Élément

Table A-57  
Attributs pour MissingValues

Attribut	Utiliser	Description	Valeurs valides
treatNullAsMissing	optional		boolean
treatWhitespaceAsMissing	optional		boolean

## Représentation XML

```
<xs:element name="MissingValues" minOccurs="0">
  <xs:sequence>
    <xs:element ref="Values" minOccurs="0" maxOccurs="unbounded"></xs:element>
    <xs:element ref="Range" minOccurs="0"></xs:element>
  </xs:sequence>
  <xs:attribute name="treatWhitespaceAsMissing" type="xs:boolean" use="optional"
    default="true"></xs:attribute>
  <xs:attribute name="treatNullAsMissing" type="xs:boolean" use="optional" default="true"></xs:attribute>
</xs:element>
```

## Parent Éléments

[Field Élément](#)

## Enfant Éléments

[Range Élément](#), [Values Élément](#)

## ModelField Élément

Table A-58  
Attributs pour ModelField

Attribut	Utiliser	Description	Valeurs valides
group	facultatif		
role	<b>obligatoire</b>		<b>unknown</b> <b>predictedValue</b> <b>predictedDisplayValue</b> <b>probability</b> <b>residual</b> <b>standardError</b> <b>entityId</b> <b>entityAffinity</b> <b>upperConfidenceLimit</b> <b>lowerConfidenceLimit</b> <b>propensity</b> <b>value</b> <b>supplementary</b>
tag	facultatif		string
targetField	facultatif		string
value	facultatif		string

**Représentation XML**

```

<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>
    <xs:enumeration value="predictedValue"></xs:enumeration>
    <xs:enumeration value="predictedDisplayValue"></xs:enumeration>
    <xs:enumeration value="probability"></xs:enumeration>
    <xs:enumeration value="residual"></xs:enumeration>
    <xs:enumeration value="standardError"></xs:enumeration>
    <xs:enumeration value="entityId"></xs:enumeration>
    <xs:enumeration value="entityAffinity"></xs:enumeration>
    <xs:enumeration value="upperConfidenceLimit"></xs:enumeration>
    <xs:enumeration value="lowerConfidenceLimit"></xs:enumeration>
    <xs:enumeration value="propensity"></xs:enumeration>
    <xs:enumeration value="value"></xs:enumeration>
    <xs:enumeration value="supplementary"></xs:enumeration>
  </xs:attribute>
  <xs:attribute name="targetField" type="xs:string"></xs:attribute>
  <xs:attribute name="value" type="xs:string"></xs:attribute>
  <xs:attribute name="group" type="MODEL-FIELD-GROUP"></xs:attribute>
  <xs:attribute name="tag" type="xs:string"></xs:attribute>
</xs:element>

```

**Parent Éléments**

[Field Élément](#)

**FieldFormats Élément**

Defines the default field formats. Field formats are used when displaying values in output such as the general format (standard number, scientific or currency formats), number of decimal places to display, decimal separator etc. Currently field formats are only used for numeric fields although this may change in future versions.

Table A-59

Attributs pour FieldFormats

Attribut	Utiliser	Description	Valeurs valides
count	facultatif		<i>nonNegativeInteger</i>
defaultNumberFormat	<b>obligatoire</b>		<i>string</i>

**Représentation XML**

```

<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>
  <xs:attribute name="count" type="xs:nonNegativeInteger"></xs:attribute>
</xs:element>

```

**Enfant Eléments**[NumberFormat Elément](#)**NumberFormat Elément**

Defines format information for a numeric field.

Table A-60

Attributs pour NumberFormat

Attribut	Utiliser	Description	Valeurs valides
decimalPlaces	<b>obligatoire</b>		<i>nonNegativeInteger</i>
decimalSymbol	<b>obligatoire</b>		<b>period</b> <b>comma</b>
formatType	<b>obligatoire</b>		<b>standard</b> <b>scientific</b> <b>currency</b>
groupingSymbol	<b>obligatoire</b>		<b>none</b> <b>period</b> <b>comma</b> <b>space</b>
name	<b>obligatoire</b>		<i>string</i>

**Représentation XML**

```
<xs:element name="NumberFormat" type="NUMBER-FORMAT-DECLARATION" minOccurs="0"
maxOccurs="unbounded">
  <xs:attribute name="name" type="xs:string" use="required"></xs:attribute>
  <xs:attribute name="formatType" type="NUMBER-FORMAT-TYPE" use="required">
    <xs:enumeration value="standard"></xs:enumeration>
    <xs:enumeration value="scientific"></xs:enumeration>
    <xs:enumeration value="currency"></xs:enumeration>
  </xs:attribute>
  <xs:attribute name="decimalPlaces" type="xs:nonNegativeInteger" use="required"></xs:attribute>
  <xs:attribute name="decimalSymbol" type="DECIMAL-SYMBOL" use="required">
    <xs:enumeration value="period"></xs:enumeration>
    <xs:enumeration value="comma"></xs:enumeration>
  </xs:attribute>
  <xs:attribute name="groupingSymbol" type="NUMBER-GROUPING-SYMBOL" use="required">
    <xs:enumeration value="none"></xs:enumeration>
    <xs:enumeration value="period"></xs:enumeration>
    <xs:enumeration value="comma"></xs:enumeration>
    <xs:enumeration value="space"></xs:enumeration>
  </xs:attribute>
</xs:element>
```

**Parent Eléments**[FieldFormats Elément](#)



## FieldGroup Élément

Defines a field group. A field group consists of a list of field names and information about the field group such as the group name and optional label, type of group and for multi-dichotomy groups, the counted value i.e. the value which represents “true”.

Table A-61  
Attributs pour FieldGroup

Attribut	Utiliser	Description	Valeurs valides
count	facultatif		<i>nonNegativeInteger</i>
countedValue	facultatif		<i>string</i>
displayLabel	facultatif		<i>string</i>
groupType	<b>obligatoire</b>		<b>fieldGroup</b> <b>multiCategorySet</b> <b>multiDichotomySet</b>
name	<b>obligatoire</b>		

### Représentation XML

```
<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>
  <xs:attribute name="displayLabel" type="xs:string"></xs:attribute>
  <xs:attribute name="groupType" type="FIELD-GROUP-TYPE" use="required">
    <xs:enumeration value="fieldGroup"></xs:enumeration>
    <xs:enumeration value="multiCategorySet"></xs:enumeration>
    <xs:enumeration value="multiDichotomySet"></xs:enumeration>
  </xs:attribute>
  <xs:attribute name="countedValue" type="xs:string"></xs:attribute>
  <xs:attribute name="count" type="xs:nonNegativeInteger"></xs:attribute>
</xs:element>
```

### Enfant Éléments

[FieldName Élément](#)

## FieldName Élément

Table A-62  
Attributs pour FieldName

Attribut	Utiliser	Description	Valeurs valides
name	<b>obligatoire</b>		

### Représentation XML

```
<xs:element name="FieldName">
  <xs:attribute name="name" type="FIELD-NAME" use="required"></xs:attribute>
</xs:element>
```

**Parent Eléments**[FieldGroup Elément](#)**FieldGroups Elément**

Defines the field groups. Field groups are used to associate related fields. For example, a survey question that asks a respondent to select which locations they have visited from a set of options will be represented as a set of flag fields. A field group may be used to identify which fields are associated with that survey question.

Table A-63  
Attributs pour FieldGroups

Attribut	Utiliser	Description	Valeurs valides
count	facultatif		<i>nonNegativeInteger</i>

**Représentation XML**

```
<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:attribute>
</xs:element>
```

**Enfant Eléments**[FieldGroup Elément](#)**FieldGroup Elément**

Defines a field group. A field group consists of a list of field names and information about the field group such as the group name and optional label, type of group and for multi-dichotomy groups, the counted value i.e. the value which represents “true”.

Table A-64  
Attributs pour FieldGroup

Attribut	Utiliser	Description	Valeurs valides
count	facultatif		<i>nonNegativeInteger</i>
countedValue	facultatif		<i>string</i>
displayLabel	facultatif		<i>string</i>
groupType	<b>obligatoire</b>		<b>fieldGroup</b> <b>multiCategorySet</b> <b>multiDichotomySet</b>
name	<b>obligatoire</b>		

**Représentation XML**

```

<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>
  <xs:attribute name="displayLabel" type="xs:string"></xs:attribute>
  <xs:attribute name="groupType" type="FIELD-GROUP-TYPE" use="required">
    <xs:enumeration value="fieldGroup"></xs:enumeration>
    <xs:enumeration value="multiCategorySet"></xs:enumeration>
    <xs:enumeration value="multiDichotomySet"></xs:enumeration>
  </xs:attribute>
  <xs:attribute name="countedValue" type="xs:string"></xs:attribute>
  <xs:attribute name="count" type="xs:nonNegativeInteger"></xs:attribute>
</xs:element>

```

**Parent Eléments**[FieldGroups Elément](#)**Enfant Eléments**[FieldName Elément](#)**FieldName Elément**

Table A-65  
Attributs pour FieldName

Attribut	Utiliser	Description	Valeurs valides
name	obligatoire		

**Représentation XML**

```

<xs:element name="FieldName">
  <xs:attribute name="name" type="FIELD-NAME" use="required"></xs:attribute>
</xs:element>

```

**Parent Eléments**[FieldGroup Elément](#)**FileFormatType Elément**

Table A-66  
Attributs pour FileFormatType

Attribut	Utiliser	Description	Valeurs valides
name	optional		

**Représentation XML**

```

<xs:element name="FileFormatType">

```

```

<xs:sequence>
  <xs:group ref="FILE-FORMAT">
    <xs:choice>
      <xs:element ref="UTF8Format"></xs:element>
      <xs:element ref="BinaryFormat"></xs:element>
      <xs:element ref="DataFormat"></xs:element>
    </xs:choice>
  </xs:group>
</xs:sequence>
<xs:attribute name="name" type="EVALUATED-STRING" use="optional"></xs:attribute>
</xs:element>

```

**Parent Eléments**[FileFormatTypes Elément](#)**Enfant Eléments**[BinaryFormat Elément](#), [DataFormat Elément](#), [UTF8Format Elément](#)**FileFormatTypes Elément****Représentation XML**

```

<xs:element name="FileFormatTypes">
  <xs:sequence>
    <xs:element ref="FileFormatType" minOccurs="0" maxOccurs="unbounded"></xs:element>
  </xs:sequence>
</xs:element>

```

**Parent Eléments**[CommonObjects Elément](#)**Enfant Eléments**[FileFormatType Elément](#)**ForEach Elément**

Table A-67  
Attributs pour ForEach

Attribut	Utiliser	Description	Valeurs valides
container	optional		<i>string</i>
from	optional		<i>string</i>
inFields	optional		<i>string</i>
inFieldValues	optional		<i>string</i>
inProperty	optional		<i>string</i>
step	optional		<i>string</i>
to	optional		<i>string</i>
var	<b>obligatoire</b>		<i>string</i>

**Représentation XML**

```

<xs:element name="ForEach">
  <xs:sequence maxOccurs="unbounded">
    <xs:group ref="DATA-MODEL-EXPRESSION">
      <xs:choice>
        <xs:element ref="ForEach"></xs:element>
        <xs:element ref="AddField"></xs:element>
        <xs:element ref="ChangeField"></xs:element>
        <xs:element ref="RemoveField"></xs:element>
      </xs:choice>
    </xs:group>
  </xs:sequence>
  <xs:attribute name="var" type="xs:string" use="required"></xs:attribute>
  <xs:attribute name="inProperty" type="xs:string" use="optional"></xs:attribute>
  <xs:attribute name="inFields" type="xs:string" use="optional"></xs:attribute>
  <xs:attribute name="inFieldValues" type="xs:string" use="optional"></xs:attribute>
  <xs:attribute name="from" type="xs:string" use="optional"></xs:attribute>
  <xs:attribute name="to" type="xs:string" use="optional"></xs:attribute>
  <xs:attribute name="step" type="xs:string" use="optional"></xs:attribute>
  <xs:attribute name="container" type="xs:string" use="optional"></xs:attribute>
</xs:element>

```

**Parent Eléments**

[ModelFields Elément](#)

**Enfant Eléments**

[AddField Elément](#), [ChangeField Elément](#), [ForEach Elément](#), [RemoveField Elément](#)

**Icon Elément**

Table A-68  
Attributs pour Icon

Attribut	Utiliser	Description	Valeurs valides
imagePath	<b>obligatoire</b>		<i>string</i>
resourceID	optional		<i>string</i>
type	<b>obligatoire</b>		<b>standardNode</b> <b>smallNode</b> <b>standardWindow</b>

**Représentation XML**

```

<xs:element name="Icon">
  <xs:attribute name="type" use="required">
    <xs:simpleType>
      <xs:restriction base="xs:string">
        <xs:enumeration value="standardNode"></xs:enumeration>
        <xs:enumeration value="smallNode"></xs:enumeration>
        <xs:enumeration value="standardWindow"></xs:enumeration>
      </xs:restriction>
    </xs:simpleType>
  </xs:attribute>

```

```
<xs:attribute name="imagePath" type="xs:string" use="required"/></xs:attribute>
<xs:attribute name="resourceID" type="xs:string" use="optional"/></xs:attribute>
</xs:element>
```

### **Parent Eléments**

[Icons Elément](#), [Palette Elément](#)

## **Icons Elément**

### **Représentation XML**

```
<xs:element name="Icons">
  <xs:sequence>
    <xs:element ref="Icon" maxOccurs="unbounded"/></xs:element>
  </xs:sequence>
</xs:element>
```

### **Parent Eléments**

[UserInterface Elément](#)

### **Enfant Eléments**

[Icon Elément](#)

## **InputFiles Elément**

### **Représentation XML**

```
<xs:element name="InputFiles">
  <xs:group ref="RUNTIME-FILES">
    <xs:sequence>
      <xs:element ref="DataFile"/></xs:element>
      <xs:element ref="ContainerFile" minOccurs="0" maxOccurs="unbounded"/></xs:element>
    </xs:sequence>
  </xs:group>
</xs:element>
```

### **Parent Eléments**

[Execution Elément](#), [Module Elément](#)

### **Enfant Eléments**

[ContainerFile Elément](#), [DataFile Elément](#)

## InteractiveDocumentBuilder Élément

Table A-69  
Attributs pour InteractiveDocumentBuilder

Attribut	Utiliser	Description	Valeurs valides
deprecatedScriptNames	optional		string
id	<b>obligatoire</b>		string
scriptName	optional		string

### Représentation XML

```
<xs:element name="InteractiveDocumentBuilder">
  <xs:sequence maxOccurs="unbounded">
    <xs:choice maxOccurs="unbounded">
      <xs:element ref="Properties"></xs:element>
      <xs:element name="Containers" minOccurs="0">
        <xs:sequence maxOccurs="unbounded">
          <xs:element ref="Container"></xs:element>
        </xs:sequence>
      </xs:element>
      <xs:element ref="UserInterface"></xs:element>
      <xs:element ref="Constructors" minOccurs="0"></xs:element>
      <xs:element ref="ModelProvider" minOccurs="0"></xs:element>
    </xs:choice>
  </xs:sequence>
  <xs:attribute name="id" type="xs:string" use="required"></xs:attribute>
  <xs:attribute name="scriptName" type="xs:string" use="optional"></xs:attribute>
  <xs:attribute name="deprecatedScriptNames" type="xs:string" use="optional"></xs:attribute>
</xs:element>
```

### Parent Éléments

[Extension Élément](#)

### Enfant Éléments

[Constructors Élément](#), [Containers Élément](#), [ModelProvider Élément](#), [Properties Élément](#), [UserInterface Élément](#)

### Éléments connexes Éléments

[DocumentOutput Élément](#), [InteractiveModelBuilder Élément](#), [ModelOutput Élément](#), [Node Élément](#)

## Containers Élément

### Représentation XML

```
<xs:element name="Containers" minOccurs="0">
  <xs:sequence maxOccurs="unbounded">
    <xs:element ref="Container"></xs:element>
  </xs:sequence>
</xs:element>
```

**Parent Eléments**[InteractiveDocumentBuilder Elément](#)**Enfant Eléments**[Container Elément](#)**InteractiveModelBuilder Elément**

Table A-70  
Attributs pour InteractiveModelBuilder

Attribut	Utiliser	Description	Valeurs valides
deprecatedScriptNames	optional		<i>string</i>
id	<b>obligatoire</b>		<i>string</i>
scriptName	optional		<i>string</i>

**Représentation XML**

```

<xs:element name="InteractiveModelBuilder">
  <xs:sequence maxOccurs="unbounded">
    <xs:choice maxOccurs="unbounded">
      <xs:element ref="Properties"></xs:element>
      <xs:element name="Containers" minOccurs="0">
        <xs:sequence maxOccurs="unbounded">
          <xs:element ref="Container"></xs:element>
        </xs:sequence>
      </xs:element>
      <xs:element ref="UserInterface"></xs:element>
      <xs:element ref="Constructors" minOccurs="0"></xs:element>
      <xs:element ref="ModelProvider" minOccurs="0"></xs:element>
    </xs:choice>
  </xs:sequence>
  <xs:attribute name="id" type="xs:string" use="required"></xs:attribute>
  <xs:attribute name="scriptName" type="xs:string" use="optional"></xs:attribute>
  <xs:attribute name="deprecatedScriptNames" type="xs:string" use="optional"></xs:attribute>
</xs:element>

```

**Parent Eléments**[Extension Elément](#)**Enfant Eléments**

[Constructors Elément](#), [Containers Elément](#), [ModelProvider Elément](#), [Properties Elément](#), [UserInterface Elément](#)

**Eléments connexes Eléments**

[DocumentOutput Elément](#), [InteractiveDocumentBuilder Elément](#), [ModelOutput Elément](#), [Node Elément](#)



## Containers Élément

### Représentation XML

```
<xs:element name="Containers" minOccurs="0">
  <xs:sequence maxOccurs="unbounded">
    <xs:element ref="Container"></xs:element>
  </xs:sequence>
</xs:element>
```

### Parent Elements

[InteractiveModelBuilder Élément](#)

### Enfant Elements

[Container Élément](#)

## Layout Élément

Table A-71  
Attributs pour Layout

Attribut	Utiliser	Description	Valeurs valides
anchor	optional		north northeast east southeast south southwest west northwest center
columnWeight	optional		double
fill	optional		horizontal vertical both none
gridColumn	optional		nonNegativeInteger
gridHeight	optional		nonNegativeInteger
gridRow	optional		nonNegativeInteger
gridWidth	optional		nonNegativeInteger
leftIndent	optional		nonNegativeInteger
rowIncrement	optional		nonNegativeInteger
rowWeight	optional		double

### Représentation XML

```
<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>
```

```

<xs:attribute name="gridColumn" type="xs:nonNegativeInteger" use="optional"></xs:attribute>
<xs:attribute name="rowIncrement" type="xs:nonNegativeInteger" use="optional"></xs:attribute>
<xs:attribute name="gridWidth" type="xs:nonNegativeInteger" use="optional" default="1"></xs:attribute>
<xs:attribute name="gridHeight" type="xs:nonNegativeInteger" use="optional" default="1"></xs:attribute>
<xs:attribute name="rowWeight" type="xs:double" use="optional"></xs:attribute>
<xs:attribute name="columnWeight" type="xs:double" use="optional"></xs:attribute>
<xs:attribute name="fill" type="UI-COMPONENT-FILL" use="optional" default="none">
  <xs:enumeration value="horizontal"></xs:enumeration>
  <xs:enumeration value="vertical"></xs:enumeration>
  <xs:enumeration value="both"></xs:enumeration>
  <xs:enumeration value="none"></xs:enumeration>
</xs:attribute>
<xs:attribute name="anchor" type="UI-COMPONENT-ANCHOR" use="optional" default="west">
  <xs:enumeration value="north"></xs:enumeration>
  <xs:enumeration value="northeast"></xs:enumeration>
  <xs:enumeration value="east"></xs:enumeration>
  <xs:enumeration value="southeast"></xs:enumeration>
  <xs:enumeration value="south"></xs:enumeration>
  <xs:enumeration value="southwest"></xs:enumeration>
  <xs:enumeration value="west"></xs:enumeration>
  <xs:enumeration value="northwest"></xs:enumeration>
  <xs:enumeration value="center"></xs:enumeration>
</xs:attribute>
<xs:attribute name="leftIndent" type="xs:nonNegativeInteger" use="optional"></xs:attribute>
</xs:element>

```

### Parent Éléments

[ActionButton Élément](#), [CheckBoxControl Élément](#), [CheckBoxGroupControl Élément](#),  
[ClientDirectoryChooserControl Élément](#), [ClientFileChooserControl Élément](#), [ComboBoxControl  
 Élément](#), [DBConnectionChooserControl Élément](#), [DBTableChooserControl Élément](#),  
[ExtensionObjectPanel Élément](#), [ModelViewerPanel Élément](#), [MultiFieldChooserControl Élément](#),  
[MultiItemChooserControl Élément](#), [PasswordBoxControl Élément](#), [PropertiesPanel Élément](#),  
[PropertiesSubPanel Élément](#), [PropertyControl Élément](#), [RadioButtonGroupControl Élément](#),  
[SelectorPanel Élément](#), [ServerDirectoryChooserControl Élément](#), [ServerFileChooserControl  
 Élément](#), [SingleFieldChooserControl Élément](#), [SingleFieldValueChooserControl Élément](#),  
[SingleItemChooserControl Élément](#), [SpinnerControl Élément](#), [StaticText Élément](#),  
[SystemControls Élément](#), [TabbedPanel Élément](#), [TableControl Élément](#), [TextAreaControl  
 Élément](#), [TextBoxControl Élément](#), [TextBrowserPanel Élément](#), [ItemChooserControl Type](#)

### Enfant Éléments

[Cell Élément](#)

### Cell Élément

Table A-72  
Attributs pour Cell

Attribut	Utiliser	Description	Valeurs valides
column	<b>obligatoire</b>		<i>nonNegativeInteger</i>
row	<b>obligatoire</b>		<i>nonNegativeInteger</i>
width	<b>obligatoire</b>		<i>nonNegativeInteger</i>

**Représentation XML**

```
<xs:element name="Cell">
  <xs:attribute name="row" type="xs:nonNegativeInteger" use="required"/></xs:attribute>
  <xs:attribute name="column" type="xs:nonNegativeInteger" use="required"/></xs:attribute>
  <xs:attribute name="width" type="xs:nonNegativeInteger" use="required"/></xs:attribute>
</xs:element>
```

**Parent Elements**

[Layout Élément](#)

**License Élément**

Reserved for system use.

**Représentation XML**

```
<xs:element name="License">
  <xs:sequence minOccurs="0" maxOccurs="unbounded">
    <xs:element ref="OptionCode"/></xs:element>
  </xs:sequence>
</xs:element>
```

**Parent Elements**

[Extension Élément](#)

**Enfant Elements**

[OptionCode Élément](#)

**ListValue Élément**

A sequence of values. All values must have the same content type but this is not checked.

**Représentation XML**

```
<xs:element name="ListValue" type="LIST-VALUE">
  <xs:group ref="PARAMETER-CONTENT" minOccurs="0" maxOccurs="unbounded">
    <xs:choice>
      <xs:element ref="MapValue"/></xs:element>
      <xs:element ref="StructuredValue"/></xs:element>
      <xs:element ref="ListValue"/></xs:element>
      <xs:element ref="Value"/></xs:element>
      <xs:element ref="DatabaseConnectionValue"/></xs:element>
    </xs:choice>
  </xs:group>
</xs:element>
```

**Parent Eléments**

[Attribute Élément](#), [ListValue Élément](#), [Parameter Élément](#), [Attribute Élément](#), [ListValue Élément](#)

**Enfant Eléments**

[DatabaseConnectionValue Élément](#), [ListValue Élément](#), [MapValue Élément](#), [StructuredValue Élément](#), [Value Élément](#)

**MapValue Élément**

A set of map entries, each consisting if a key and a value.

**Représentation XML**

```
<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>
                  <xs:element ref="StructuredValue"></xs:element>
                  <xs:element ref="ListValue"></xs:element>
                  <xs:element ref="Value"></xs:element>
                  <xs:element ref="DatabaseConnectionValue"></xs:element>
                </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>
                  <xs:element ref="StructuredValue"></xs:element>
                  <xs:element ref="ListValue"></xs:element>
                  <xs:element ref="Value"></xs:element>
                  <xs:element ref="DatabaseConnectionValue"></xs:element>
                </xs:choice>
              </xs:group>
            </xs:element>
          </xs:sequence>
        </xs:element>
      </xs:sequence>
    </xs:element>
  </xs:sequence>
</xs:element>
```

**Parent Eléments**

[ListValue Elément](#), [Attribute Elément](#), [ListValue Elément](#), [Parameter Elément](#), [Attribute Elément](#), [ListValue Elément](#)

**Enfant Eléments**

[MapEntry Elément](#)

**MapEntry Elément**

An entry in a keyed property map. Each entry consists of a key and an associated value.

**Représentation XML**

```
<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>
              <xs:element ref="StructuredValue"></xs:element>
              <xs:element ref="ListValue"></xs:element>
              <xs:element ref="Value"></xs:element>
              <xs:element ref="DatabaseConnectionValue"></xs:element>
            </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>
              <xs:element ref="StructuredValue"></xs:element>
              <xs:element ref="ListValue"></xs:element>
              <xs:element ref="Value"></xs:element>
              <xs:element ref="DatabaseConnectionValue"></xs:element>
            </xs:choice>
          </xs:group>
        </xs:element>
      </xs:sequence>
    </xs:element>
  </xs:sequence>
</xs:element>
```

**Parent Eléments**

[MapValue Elément](#)

**Enfant Eléments**

[KeyValue Elément](#), [StructuredValue Elément](#)

**KeyValue Elément**

The key value in a map entry.

Table A-73

Attributs pour *KeyValue*

Attribut	Utiliser	Description	Valeurs valides
value	obligatoire		string

**Représentation XML**

```
<xs:element name="KeyValue" type="KEY-VALUE">
  <xs:attribute name="value" type="xs:string" use="required"/></xs:attribute>
</xs:element>
```

**Parent Eléments**

[MapEntry Elément](#)

**StructuredValue Elément**

A sequence of named values (“attributes”).

**Représentation XML**

```
<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>
          <xs:element ref="StructuredValue"/></xs:element>
          <xs:element ref="ListValue"/></xs:element>
          <xs:element ref="Value"/></xs:element>
          <xs:element ref="DatabaseConnectionValue"/></xs:element>
        </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>
          <xs:element ref="StructuredValue"/></xs:element>
          <xs:element ref="ListValue"/></xs:element>
          <xs:element ref="Value"/></xs:element>
          <xs:element ref="DatabaseConnectionValue"/></xs:element>
        </xs:choice>
      </xs:group>
    </xs:element>
  </xs:sequence>
```

```

    </xs:element>
  </xs:sequence>
</xs:element>

```

### Parent Eléments

[MapEntry Elément](#)

### Enfant Eléments

[Attribute Elément](#)

## Attribute Elément

Table A-74  
Attributs pour Attribute

Attribut	Utiliser	Description	Valeurs valides
name	<b>obligatoire</b>		<i>string</i>
value	facultatif		<i>string</i>

### Représentation XML

```

<xs:element name="Attribute" type="ATTRIBUTE" maxOccurs="unbounded">
  <xs:group ref="PARAMETER-CONTENT" minOccurs="0">
    <xs:choice>
      <xs:element ref="MapValue"></xs:element>
      <xs:element ref="StructuredValue"></xs:element>
      <xs:element ref="ListValue"></xs:element>
      <xs:element ref="Value"></xs:element>
      <xs:element ref="DatabaseConnectionValue"></xs:element>
    </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>
          <xs:element ref="StructuredValue"></xs:element>
          <xs:element ref="ListValue"></xs:element>
          <xs:element ref="Value"></xs:element>
          <xs:element ref="DatabaseConnectionValue"></xs:element>
        </xs:choice>
      </xs:group>
    </xs:element>
  </xs:sequence>
  <xs:attribute name="name" type="xs:string" use="required"></xs:attribute>
  <xs:attribute name="value" type="xs:string"></xs:attribute>
</xs:element>

```

### Parent Eléments

[StructuredValue Elément](#)

**Enfant Eléments**

[DatabaseConnectionValue Elément](#), [ListValue Elément](#), [ListValue Elément](#), [MapValue Elément](#), [StructuredValue Elément](#), [Value Elément](#)

**ListValue Elément**

A sequence of values. All values must have the same content type but this is not checked.

**Représentation XML**

```
<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>
      <xs:element ref="StructuredValue"></xs:element>
      <xs:element ref="ListValue"></xs:element>
      <xs:element ref="Value"></xs:element>
      <xs:element ref="DatabaseConnectionValue"></xs:element>
    </xs:choice>
  </xs:group>
</xs:element>
```

**Parent Eléments**

[Attribute Elément](#)

**Enfant Eléments**

[DatabaseConnectionValue Elément](#), [ListValue Elément](#), [MapValue Elément](#), [StructuredValue Elément](#), [Value Elément](#)

**Menu Elément**

Table A-75  
Attributs pour Menu

Attribut	Utiliser	Description	Valeurs valides
id	<b>obligatoire</b>		<i>string</i>
label	<b>obligatoire</b>		<i>string</i>
labelKey	optional		<i>string</i>
mnemonic	optional		<i>string</i>
mnemonicKey	optional		<i>string</i>
offset	optional		<i>nonNegativeInteger</i>
separatorAfter	optional		<i>boolean</i>
separatorBefore	optional		<i>boolean</i>
showIcon	optional		<i>boolean</i>



Attribut	Utiliser	Description	Valeurs valides
showLabel	optional		boolean
systemMenu	<b>obligatoire</b>		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

### Représentation XML

```

<xs:element name="Menu">
  <xs:attribute name="id" type="xs:string" use="required"/></xs:attribute>
  <xs:attribute name="label" type="xs:string" use="required"/></xs:attribute>
  <xs:attribute name="labelKey" type="xs:string" use="optional"/></xs:attribute>
  <xs:attribute name="mnemonic" type="xs:string" use="optional"/></xs:attribute>
  <xs:attribute name="mnemonicKey" type="xs:string" use="optional"/></xs:attribute>
  <xs:attribute name="systemMenu" type="STANDARD-MENU" use="required">
    <xs:enumeration value="file"/></xs:enumeration>
    <xs:enumeration value="edit"/></xs:enumeration>
    <xs:enumeration value="insert"/></xs:enumeration>
    <xs:enumeration value="view"/></xs:enumeration>
    <xs:enumeration value="tools"/></xs:enumeration>
    <xs:enumeration value="window"/></xs:enumeration>
    <xs:enumeration value="help"/></xs:enumeration>
    <xs:enumeration value="generate"/></xs:enumeration>
    <xs:enumeration value="file.project"/></xs:enumeration>
    <xs:enumeration value="file.outputs"/></xs:enumeration>
    <xs:enumeration value="file.models"/></xs:enumeration>
    <xs:enumeration value="edit.stream"/></xs:enumeration>
    <xs:enumeration value="edit.node"/></xs:enumeration>
    <xs:enumeration value="edit.outputs"/></xs:enumeration>
    <xs:enumeration value="edit.models"/></xs:enumeration>
    <xs:enumeration value="edit.project"/></xs:enumeration>
    <xs:enumeration value="tools.repository"/></xs:enumeration>
    <xs:enumeration value="tools.options"/></xs:enumeration>
    <xs:enumeration value="tools.streamProperties"/></xs:enumeration>
  </xs:attribute>
  <xs:attribute name="showLabel" type="xs:boolean" use="optional" default="true"/></xs:attribute>
  <xs:attribute name="showIcon" type="xs:boolean" use="optional" default="false"/></xs:attribute>
  <xs:attribute name="separatorBefore" type="xs:boolean" use="optional" default="false"/></xs:attribute>
  <xs:attribute name="separatorAfter" type="xs:boolean" use="optional" default="false"/></xs:attribute>
  <xs:attribute name="offset" type="xs:nonNegativeInteger" use="optional" default="0"/></xs:attribute>
</xs:element>

```

**Parent Eléments**

## Controls Elément

**MenuItem Elément**

Table A-76

Attributs pour MenuItem

Attribut	Utiliser	Description	Valeurs valides
action	<b>obligatoire</b>		<i>string</i>
customMenu	optional		<i>string</i>
offset	optional		<i>nonNegativeInteger</i>
separatorAfter	optional		<i>boolean</i>
separatorBefore	optional		<i>boolean</i>
showIcon	optional		<i>boolean</i>
showLabel	optional		<i>boolean</i>
systemMenu	optional		<b>file</b> <b>edit</b> <b>insert</b> <b>view</b> <b>tools</b> <b>window</b> <b>help</b> <b>generate</b> <b>file.project</b> <b>file.outputs</b> <b>file.models</b> <b>edit.stream</b> <b>edit.node</b> <b>edit.outputs</b> <b>edit.models</b> <b>edit.project</b> <b>tools.repository</b> <b>tools.options</b> <b>tools.streamProperties</b>

**Représentation XML**

```

<xs:element name="MenuItem">
  <xs:attribute name="action" type="xs:string" use="required"/></xs:attribute>
  <xs:attribute name="systemMenu" type="STANDARD-MENU" use="optional">
    <xs:enumeration value="file"/></xs:enumeration>
    <xs:enumeration value="edit"/></xs:enumeration>
    <xs:enumeration value="insert"/></xs:enumeration>
    <xs:enumeration value="view"/></xs:enumeration>
    <xs:enumeration value="tools"/></xs:enumeration>
    <xs:enumeration value="window"/></xs:enumeration>
    <xs:enumeration value="help"/></xs:enumeration>
    <xs:enumeration value="generate"/></xs:enumeration>
    <xs:enumeration value="file.project"/></xs:enumeration>
    <xs:enumeration value="file.outputs"/></xs:enumeration>
    <xs:enumeration value="file.models"/></xs:enumeration>
    <xs:enumeration value="edit.stream"/></xs:enumeration>
    <xs:enumeration value="edit.node"/></xs:enumeration>
    <xs:enumeration value="edit.outputs"/></xs:enumeration>
  </xs:attribute>
</xs:element>

```

```

    <xs:enumeration value="edit.models"></xs:enumeration>
    <xs:enumeration value="edit.project"></xs:enumeration>
    <xs:enumeration value="tools.repository"></xs:enumeration>
    <xs:enumeration value="tools.options"></xs:enumeration>
    <xs:enumeration value="tools.streamProperties"></xs:enumeration>
  </xs:attribute>
  <xs:attribute name="customMenu" type="xs:string" use="optional"></xs:attribute>
  <xs:attribute name="showLabel" type="xs:boolean" use="optional" default="true"></xs:attribute>
  <xs:attribute name="showIcon" type="xs:boolean" use="optional" default="false"></xs:attribute>
  <xs:attribute name="separatorBefore" type="xs:boolean" use="optional" default="false"></xs:attribute>
  <xs:attribute name="separatorAfter" type="xs:boolean" use="optional" default="false"></xs:attribute>
  <xs:attribute name="offset" type="xs:nonNegativeInteger" use="optional" default="0"></xs:attribute>
</xs:element>

```

### Parent Eléments

[Controls Elément](#)

## MissingValues Elément

Table A-77  
Attributs pour MissingValues

Attribut	Utiliser	Description	Valeurs valides
treatNullAsMissing	facultatif		<i>boolean</i>
treatWhitespaceAsMissing	facultatif		<i>boolean</i>

### Représentation XML

```

<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>
  <xs:attribute name="treatWhitespaceAsMissing" type="xs:boolean" default="false"></xs:attribute>
</xs:element>

```

### Parent Eléments

[Field Elément](#)

### Enfant Eléments

[Range Elément](#), [Values Elément](#)

**Range Élément**

Table A-78  
Attributs pour Range

Attribut	Utiliser	Description	Valeurs valides
maxValue	<b>obligatoire</b>		<i>string</i>
minValue	<b>obligatoire</b>		<i>string</i>

**Représentation XML**

```
<xs:element name="Range" type="RANGE">
  <xs:attribute name="minValue" type="xs:string" use="required"></xs:attribute>
  <xs:attribute name="maxValue" type="xs:string" use="required"></xs:attribute>
</xs:element>
```

**Parent Elements**

[MissingValues Élément](#)

**Values Élément**

Table A-79  
Attributs pour Values

Attribut	Utiliser	Description	Valeurs valides
count	facultatif		<i>nonNegativeInteger</i>

**Représentation XML**

```
<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:attribute>
</xs:element>
```

**Parent Elements**

[MissingValues Élément](#)

**Enfant Elements**

[Value Élément](#)

**Value Élément**

Table A-80  
Attributs pour Value

Attribut	Utiliser	Description	Valeurs valides
code	<b>obligatoire</b>		<i>integer</i>
displayLabel	facultatif		<i>string</i>
flagValue	facultatif		<i>boolean</i>
value	<b>obligatoire</b>		<i>string</i>

**Représentation XML**

```
<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>
  <xs:attribute name="code" type="xs:integer" use="required"></xs:attribute>
  <xs:attribute name="flagValue" type="xs:boolean"></xs:attribute>
  <xs:attribute name="displayLabel" type="xs:string"></xs:attribute>
</xs:element>
```

**Parent Éléments**

[Values Élément](#)

**Enfant Éléments**

[DisplayLabel Élément](#)

**DisplayLabel Élément**

A display label for a field or value in a specified language. The displayLabel attribute can be used where there is no label for a particular language.

Table A-81  
Attributs pour DisplayLabel

Attribut	Utiliser	Description	Valeurs valides
lang	facultatif		<i>NMTOKEN</i>
value	<b>obligatoire</b>		<i>string</i>

**Représentation XML**

```
<xs:element name="DisplayLabel" type="DISPLAY-LABEL" minOccurs="0" maxOccurs="unbounded">
  <xs:attribute name="value" type="xs:string" use="required"></xs:attribute>
  <xs:attribute name="lang" type="xs:NMTOKEN" default="en"></xs:attribute>
</xs:element>
```

**Parent Éléments**

[Value Élément](#)

## ModelBuilder Elément

Table A-82  
Attributs pour ModelBuilder

Attribut	Utiliser	Description	Valeurs valides
allowNoInputs	optional		boolean
allowNoOutputs	optional		boolean
miningFunctions	<b>obligatoire</b>		tout
nullifyBlanks	optional		boolean

### Représentation XML

```

<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>
            <xs:element ref="AddField"></xs:element>
            <xs:element ref="ChangeField"></xs:element>
            <xs:element ref="RemoveField"></xs:element>
          </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:element>
          </xs:sequence>
        </xs:element>
        <xs:element name="ExpertSettings" minOccurs="0">
          <xs:sequence>
            <xs:element ref="Condition" minOccurs="0"></xs:element>
            <xs:element ref="PropertyGroup" maxOccurs="unbounded"></xs:element>
          </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:element>
  </xs:sequence>
</xs:element>
</xs:sequence>
<xs:attribute name="miningFunctions" use="required"></xs:attribute>
<xs:attribute name="allowNoInputs" type="xs:boolean" use="optional" default="false"></xs:attribute>
<xs:attribute name="allowNoOutputs" type="xs:boolean" use="optional" default="false"></xs:attribute>
<xs:attribute name="nullifyBlanks" type="xs:boolean" use="optional" default="true"></xs:attribute>
</xs:element>

```

### Parent Eléments

[Node Elément](#)

### Enfant Eléments

[Algorithm Elément](#), [AutoModeling Elément](#), [ModelEvaluation Elément](#), [ModelFields Elément](#), [ModelGeneration Elément](#), [ModelingFields Elément](#)

## Algorithm Elément

Table A-83  
Attributs pour Algorithm

Attribut	Utiliser	Description	Valeurs valides
label	<b>obligatoire</b>		<i>string</i>
labelKey	optional		<i>string</i>
value	<b>obligatoire</b>		<i>string</i>

### Représentation XML

```

<xs:element name="Algorithm">
  <xs:attribute name="value" type="xs:string" use="required"></xs:attribute>
  <xs:attribute name="label" type="xs:string" use="required"></xs:attribute>
  <xs:attribute name="labelKey" type="xs:string" use="optional"></xs:attribute>
</xs:element>

```

### Parent Eléments

[ModelBuilder Elément](#)

## ModelingFields Elément

Table A-84  
Attributs pour ModelingFields

Attribut	Utiliser	Description	Valeurs valides
controlsId	<b>obligatoire</b>		<i>string</i>
ignoreBOTH	optional		<i>boolean</i>

**Représentation XML**

```
<xs:element name="ModelingFields" minOccurs="0">
  <xs:attribute name="controlslid" type="xs:string" use="required"/></xs:attribute>
  <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:attribute>
</xs:element>
```

**Parent Eléments**

[ModelBuilder Elément](#)

**Enfant Eléments**

[InputFields Elément](#), [OutputFields Elément](#)

**Eléments connexes Eléments**

[DocumentGeneration Elément](#), [ModelGeneration Elément](#), [ModelEvaluation Elément](#)

**InputFields Elément**

Table A-85  
Attributs pour InputFields

Attribut	Utiliser	Description	Valeurs valides
label	<b>obligatoire</b>		<i>string</i>
labelKey	optional		<i>string</i>
multiple	<b>obligatoire</b>		<i>boolean</i>
onlyDatetime	optional		<i>boolean</i>
onlyDiscrete	optional		<i>boolean</i>
onlyNumeric	optional		<i>boolean</i>
onlyRanges	optional		<i>boolean</i>
onlySymbolic	optional		<i>boolean</i>
property	<b>obligatoire</b>		<i>string</i>
storage	optional		<i>string</i>
types	optional		<i>string</i>

**Représentation XML**

```
<xs:element name="InputFields" minOccurs="0">
  <xs:attribute name="storage" type="xs:string" use="optional"/></xs:attribute>
  <xs:attribute name="onlyNumeric" type="xs:boolean" use="optional"/></xs:attribute>
  <xs:attribute name="onlySymbolic" type="xs:boolean" use="optional"/></xs:attribute>
  <xs:attribute name="onlyDatetime" type="xs:boolean" use="optional"/></xs:attribute>
  <xs:attribute name="types" type="xs:string" use="optional"/></xs:attribute>
  <xs:attribute name="onlyRanges" type="xs:boolean" use="optional"/></xs:attribute>
  <xs:attribute name="onlyDiscrete" type="xs:boolean" use="optional"/></xs:attribute>
  <xs:attribute name="property" type="xs:string" use="required"/></xs:attribute>
  <xs:attribute name="multiple" type="xs:boolean" use="required"/></xs:attribute>
```



```

<xs:attribute name="label" type="xs:string" use="required"/></xs:attribute>
<xs:attribute name="labelKey" type="xs:string" use="optional"/></xs:attribute>
</xs:element>

```

### Parent Eléments

[ModelingFields Elément](#)

### Eléments connexes Eléments

[OutputFields Elément](#)

## OutputFields Elément

Table A-86  
Attributs pour OutputFields

Attribut	Utiliser	Description	Valeurs valides
label	<b>obligatoire</b>		<i>string</i>
labelKey	optional		<i>string</i>
multiple	<b>obligatoire</b>		<i>boolean</i>
onlyDatetime	optional		<i>boolean</i>
onlyDiscrete	optional		<i>boolean</i>
onlyNumeric	optional		<i>boolean</i>
onlyRanges	optional		<i>boolean</i>
onlySymbolic	optional		<i>boolean</i>
property	<b>obligatoire</b>		<i>string</i>
storage	optional		<i>string</i>
types	optional		<i>string</i>

### Représentation XML

```

<xs:element name="OutputFields" minOccurs="0">
  <xs:attribute name="storage" type="xs:string" use="optional"/></xs:attribute>
  <xs:attribute name="onlyNumeric" type="xs:boolean" use="optional"/></xs:attribute>
  <xs:attribute name="onlySymbolic" type="xs:boolean" use="optional"/></xs:attribute>
  <xs:attribute name="onlyDatetime" type="xs:boolean" use="optional"/></xs:attribute>
  <xs:attribute name="types" type="xs:string" use="optional"/></xs:attribute>
  <xs:attribute name="onlyRanges" type="xs:boolean" use="optional"/></xs:attribute>
  <xs:attribute name="onlyDiscrete" type="xs:boolean" use="optional"/></xs:attribute>
  <xs:attribute name="property" type="xs:string" use="required"/></xs:attribute>
  <xs:attribute name="multiple" type="xs:boolean" use="required"/></xs:attribute>
  <xs:attribute name="label" type="xs:string" use="required"/></xs:attribute>
  <xs:attribute name="labelKey" type="xs:string" use="optional"/></xs:attribute>
</xs:element>

```

### Parent Eléments

[ModelingFields Elément](#)

### Eléments connexes Eléments

[InputFields Elément](#)

## ModelGeneration Élément

Table A-87  
Attributs pour ModelGeneration

Attribut	Utiliser	Description	Valeurs valides
controlsId	<b>obligatoire</b>		<i>string</i>

### Représentation XML

```
<xs:element name="ModelGeneration">
  <xs:attribute name="controlsId" type="xs:string" use="required"/></xs:attribute>
</xs:element>
```

### Parent Elements

[ModelBuilder Élément](#)

### Eléments connexes Eléments

[DocumentGeneration Élément](#), [ModelingFields Élément](#), [ModelEvaluation Élément](#)

## ModelFields Élément

### Représentation XML

```
<xs:element name="ModelFields">
  <xs:sequence minOccurs="0" maxOccurs="unbounded">
    <xs:group ref="DATA-MODEL-EXPRESSION">
      <xs:choice>
        <xs:element ref="ForEach"/></xs:element>
        <xs:element ref="AddField"/></xs:element>
        <xs:element ref="ChangeField"/></xs:element>
        <xs:element ref="RemoveField"/></xs:element>
      </xs:choice>
    </xs:group>
  </xs:sequence>
</xs:element>
```

### Parent Elements

[ModelBuilder Élément](#)

### Enfant Elements

[AddField Élément](#), [ChangeField Élément](#), [ForEach Élément](#), [RemoveField Élément](#)

**ModelEvaluation Élément**

Table A-88  
Attributs pour ModelEvaluation

Attribut	Utiliser	Description	Valeurs valides
container	<b>obligatoire</b>		<i>tout</i>
controlsId	<b>obligatoire</b>		<i>string</i>
outputContainer	optional		<i>tout</i>

**Représentation XML**

```
<xs:element name="ModelEvaluation" minOccurs="0">
  <xs:attribute name="controlsId" type="xs:string" use="required"/></xs:attribute>
  <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>
  <xs:attribute name="outputContainer" use="optional"/></xs:attribute>
</xs:element>
```

**Parent Eléments**

[ModelBuilder Élément](#)

**Enfant Eléments**

[AdjustedPropensity Élément](#), [RawPropensity Élément](#), [VariableImportance Élément](#)

**Eléments connexes Eléments**

[DocumentGeneration Élément](#), [ModelingFields Élément](#), [ModelGeneration Élément](#)

**RawPropensity Élément**

Table A-89  
Attributs pour RawPropensity

Attribut	Utiliser	Description	Valeurs valides
availabilityProperty	optional		<i>string</i>
defaultValue	optional		<i>boolean</i>
property	optional		<i>string</i>

**Représentation XML**

```
<xs:element name="RawPropensity" minOccurs="0">
  <xs:attribute name="property" type="xs:string" use="optional"/></xs:attribute>
  <xs:attribute name="availabilityProperty" type="xs:string" use="optional"/></xs:attribute>
  <xs:attribute name="defaultValue" type="xs:boolean" use="optional"/></xs:attribute>
</xs:element>
```

**Parent Eléments**[ModelEvaluation Elément](#)**AdjustedPropensity Elément**

Table A-90  
Attributs pour AdjustedPropensity

Attribut	Utiliser	Description	Valeurs valides
availabilityProperty	optional		string
defaultValue	optional		boolean
property	optional		string

**Représentation XML**

```
<xs:element name="AdjustedPropensity" minOccurs="0">
  <xs:attribute name="property" type="xs:string" use="optional"/></xs:attribute>
  <xs:attribute name="availabilityProperty" type="xs:string" use="optional"/></xs:attribute>
  <xs:attribute name="defaultValue" type="xs:boolean" use="optional"/></xs:attribute>
</xs:element>
```

**Parent Eléments**[ModelEvaluation Elément](#)**VariableImportance Elément**

Table A-91  
Attributs pour VariableImportance

Attribut	Utiliser	Description	Valeurs valides
availabilityProperty	optional		string
defaultValue	optional		boolean
property	optional		string

**Représentation XML**

```
<xs:element name="VariableImportance" minOccurs="0">
  <xs:attribute name="property" type="xs:string" use="optional"/></xs:attribute>
  <xs:attribute name="availabilityProperty" type="xs:string" use="optional"/></xs:attribute>
  <xs:attribute name="defaultValue" type="xs:boolean" use="optional"/></xs:attribute>
</xs:element>
```

**Parent Eléments**[ModelEvaluation Elément](#)**AutoModeling Elément**

Table A-92  
Attributs pour AutoModeling

Attribut	Utiliser	Description	Valeurs valides
enabledByDefault	optional		tout

**Représentation XML**

```

<xs:element name="AutoModeling" minOccurs="0">
  <xs:sequence>
    <xs:element name="SimpleSettings">
      <xs:sequence>
        <xs:element ref="PropertyGroup" maxOccurs="unbounded"/></xs:element>
      </xs:sequence>
    </xs:element>
    <xs:element name="ExpertSettings" minOccurs="0">
      <xs:sequence>
        <xs:element ref="Condition" minOccurs="0"/></xs:element>
        <xs:element ref="PropertyGroup" maxOccurs="unbounded"/></xs:element>
      </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:element>
  </xs:sequence>
  <xs:attribute name="enabledByDefault" use="optional"/></xs:attribute>
</xs:element>

```

**Parent Eléments**

[ModelBuilder Elément](#)

**Enfant Eléments**

[Constraint Elément](#), [ExpertSettings Elément](#), [PropertyMap Elément](#), [SimpleSettings Elément](#)

**SimpleSettings Elément****Représentation XML**

```

<xs:element name="SimpleSettings">
  <xs:sequence>
    <xs:element ref="PropertyGroup" maxOccurs="unbounded"/></xs:element>
  </xs:sequence>
</xs:element>

```

**Parent Eléments**

[AutoModeling Elément](#)

**Enfant Eléments**

[PropertyGroup Elément](#)

## **ExpertSettings Élément**

### **Représentation XML**

```
<xs:element name="ExpertSettings" minOccurs="0">
  <xs:sequence>
    <xs:element ref="Condition" minOccurs="0"></xs:element>
    <xs:element ref="PropertyGroup" maxOccurs="unbounded"></xs:element>
  </xs:sequence>
</xs:element>
```

### **Parent Elements**

[AutoModeling Élément](#)

### **Enfant Elements**

[Condition Élément](#), [PropertyGroup Élément](#)

## **PropertyMap Élément**

### **Représentation XML**

```
<xs:element name="PropertyMap" minOccurs="0">
  <xs:sequence>
    <xs:element name="PropertyMapping" maxOccurs="unbounded"></xs:element>
  </xs:sequence>
</xs:element>
```

### **Parent Elements**

[AutoModeling Élément](#)

### **Enfant Elements**

[PropertyMapping Élément](#)

## **PropertyMapping Élément**

Table A-93  
Attributs pour PropertyMapping

Attribut	Utiliser	Description	Valeurs valides
property	<b>obligatoire</b>		<i>string</i>
systemProperty	<b>obligatoire</b>		<i>string</i>

### **Représentation XML**

```
<xs:element name="PropertyMapping" maxOccurs="unbounded">
  <xs:attribute name="property" type="xs:string" use="required"></xs:attribute>
  <xs:attribute name="systemProperty" type="xs:string" use="required"></xs:attribute>
</xs:element>
```

**Parent Eléments**[PropertyMap Elément](#)**ModelOutput Elément**

Table A-94  
Attributs pour ModelOutput

Attribut	Utiliser	Description	Valeurs valides
deprecatedScriptNames	optional		string
helpLink	optional		string
id	<b>obligatoire</b>		string
label	optional		string
labelKey	optional		string
scriptName	optional		string

**Représentation XML**

```

<xs:element name="ModelOutput">
  <xs:sequence maxOccurs="unbounded">
    <xs:choice maxOccurs="unbounded">
      <xs:element ref="Properties"></xs:element>
      <xs:element name="Containers" minOccurs="0">
        <xs:sequence maxOccurs="unbounded">
          <xs:element ref="Container"></xs:element>
        </xs:sequence>
      </xs:element>
      <xs:element ref="UserInterface"></xs:element>
      <xs:element ref="Constructors" minOccurs="0"></xs:element>
      <xs:element ref="ModelProvider" minOccurs="0"></xs:element>
    </xs:choice>
  </xs:sequence>
  <xs:attribute name="id" type="xs:string" use="required"></xs:attribute>
  <xs:attribute name="scriptName" type="xs:string" use="optional"></xs:attribute>
  <xs:attribute name="deprecatedScriptNames" type="xs:string" use="optional"></xs:attribute>
  <xs:attribute name="label" type="xs:string" use="optional"></xs:attribute>
  <xs:attribute name="labelKey" type="xs:string" use="optional"></xs:attribute>
  <xs:attribute name="helpLink" type="xs:string" use="optional"></xs:attribute>
</xs:element>

```

**Parent Eléments**[Extension Elément](#)**Enfant Eléments**

[Constructors Elément](#), [Containers Elément](#), [ModelProvider Elément](#), [Properties Elément](#), [UserInterface Elément](#)

**Eléments connexes Eléments**

[DocumentOutput Elément](#), [InteractiveDocumentBuilder Elément](#), [InteractiveModelBuilder Elément](#), [Node Elément](#)

## Containers Élément

### Représentation XML

```
<xs:element name="Containers" minOccurs="0">
  <xs:sequence maxOccurs="unbounded">
    <xs:element ref="Container"></xs:element>
  </xs:sequence>
</xs:element>
```

### Parent Elements

[ModelOutput Élément](#)

### Enfant Elements

[Container Élément](#)

## ModelProvider Élément

Table A-95  
Attributs pour ModelProvider

Attribut	Utiliser	Description	Valeurs valides
container	<b>obligatoire</b>		<i>string</i>
isPMML	optional		<i>boolean</i>

### Représentation XML

```
<xs:element name="ModelProvider">
  <xs:attribute name="container" type="xs:string" use="required"></xs:attribute>
  <xs:attribute name="isPMML" type="xs:boolean" use="optional" default="true"></xs:attribute>
</xs:element>
```

### Parent Elements

[DocumentOutput Élément](#), [InteractiveDocumentBuilder Élément](#), [InteractiveModelBuilder Élément](#), [ModelOutput Élément](#), [Node Élément](#)

## ModelType Élément

Defines a new model type

Table A-96  
Attributs pour ModelType

Attribut	Utiliser	Description	Valeurs valides
format	<b>obligatoire</b>		<b>utf8</b> <b>binary</b>
id	<b>obligatoire</b>		<i>string</i>
inputDirection	optional		<i>string</i>
outputDirection	optional		<i>string</i>



**Représentation XML**

```

<xs:element name="ModelType">
  <xs:attribute name="id" type="xs:string" use="required"/></xs:attribute>
  <xs:attribute name="format" use="required">
    <xs:simpleType>
      <xs:restriction base="xs:string">
        <xs:enumeration value="utf8"/></xs:enumeration>
        <xs:enumeration value="binary"/></xs:enumeration>
      </xs:restriction>
    </xs:simpleType>
  </xs:attribute>
  <xs:attribute name="inputDirection" type="xs:string" use="optional" default="[in]"/></xs:attribute>
  <xs:attribute name="outputDirection" type="xs:string" use="optional" default="[out]"/></xs:attribute>
</xs:element>

```

**Parent Eléments**[ContainerTypes Elément](#)**Eléments connexes Eléments**[DocumentType Elément](#)**ModelViewerPanel Elément**

Table A-97  
Attributs pour ModelViewerPanel

Attribut	Utiliser	Description	Valeurs valides
container	<b>obligatoire</b>		<i>string</i>

**Représentation XML**

```

<xs:element name="ModelViewerPanel">
  <xs:sequence>
    <xs:choice>
      <xs:element ref="Layout" minOccurs="0" maxOccurs="1"/></xs:element>
      <xs:element ref="Enabled" minOccurs="0" maxOccurs="1"/></xs:element>
      <xs:element ref="Visible" minOccurs="0" maxOccurs="1"/></xs:element>
    </xs:choice>
  </xs:sequence>
  <xs:attribute name="container" type="xs:string" use="required"/></xs:attribute>
</xs:element>

```

**Parent Eléments**[Tab Elément](#)**Enfant Eléments**[Enabled Elément](#), [Layout Elément](#), [Visible Elément](#)

**Eléments connexes Eléments**

[ActionButton Elément](#), [ComboBoxControl Elément](#), [ExtensionObjectPanel Elément](#),  
[SelectorPanel Elément](#), [StaticText Elément](#), [SystemControls Elément](#), [TabbedPanel Elément](#),  
[TextBrowserPanel Elément](#)

**Module Elément**

Table A-98

Attributs pour Module

Attribut	Utiliser	Description	Valeurs valides
libraryId	optional		string
name	optional		string
systemModule	optional		<b>SmartScore</b>

**Représentation XML**

```
<xs:element name="Module">
  <xs:sequence>
    <xs:element ref="InputFiles"></xs:element>
    <xs:element ref="OutputFiles"></xs:element>
    <xs:element ref="StatusCodes" minOccurs="0" maxOccurs="1"></xs:element>
  </xs:sequence>
  <xs:attribute name="systemModule" use="optional" default="SmartScore">
    <xs:simpleType>
      <xs:restriction base="xs:string">
        <xs:enumeration value="SmartScore"></xs:enumeration>
      </xs:restriction>
    </xs:simpleType>
  </xs:attribute>
  <xs:attribute name="libraryId" type="xs:string" use="optional"></xs:attribute>
  <xs:attribute name="name" type="xs:string" use="optional"></xs:attribute>
</xs:element>
```

**Parent Eléments**

[Execution Elément](#)

**Enfant Eléments**

[InputFiles Elément](#), [OutputFiles Elément](#), [StatusCodes Elément](#)

**MultiFieldChooserControl Elément**

Table A-99

Attributs pour MultiFieldChooserControl

Attribut	Utiliser	Description	Valeurs valides
description	optional		string
descriptionKey	optional		string
label	optional		string
labelAbove	optional		boolean

Attribut	Utiliser	Description	Valeurs valides
labelKey	optional		string
labelWidth	optional		positiveInteger
mnemonic	optional		string
mnemonicKey	optional		string
onlyDatetime	optional		boolean
onlyDiscrete	optional		boolean
onlyNumeric	optional		boolean
onlyRanges	optional		boolean
onlySymbolic	optional		boolean
property	<b>obligatoire</b>		string
showLabel	optional		boolean
storage	optional		string
types	optional		string

### Représentation XML

```

<xs:element name="MultiFieldChooserControl">
  <xs:sequence>
    <xs:choice>
      <xs:element ref="Layout" minOccurs="0" maxOccurs="1"></xs:element>
      <xs:element ref="Enabled" minOccurs="0" maxOccurs="1"></xs:element>
      <xs:element ref="Visible" minOccurs="0" maxOccurs="1"></xs:element>
    </xs:choice>
  </xs:sequence>
  <xs:attribute name="property" type="xs:string" use="required"></xs:attribute>
  <xs:attribute name="showLabel" type="xs:boolean" use="optional" default="true"></xs:attribute>
  <xs:attribute name="label" type="xs:string" use="optional"></xs:attribute>
  <xs:attribute name="labelKey" type="xs:string" use="optional"></xs:attribute>
  <xs:attribute name="mnemonic" type="xs:string" use="optional"></xs:attribute>
  <xs:attribute name="mnemonicKey" type="xs:string" use="optional"></xs:attribute>
  <xs:attribute name="labelWidth" type="xs:positiveInteger" use="optional" default="1"></xs:attribute>
  <xs:attribute name="labelAbove" type="xs:boolean" use="optional" default="false"></xs:attribute>
  <xs:attribute name="description" type="xs:string" use="optional"></xs:attribute>
  <xs:attribute name="descriptionKey" type="xs:string" use="optional"></xs:attribute>
  <xs:attribute name="storage" type="xs:string" use="optional"></xs:attribute>
  <xs:attribute name="onlyNumeric" type="xs:boolean" use="optional"></xs:attribute>
  <xs:attribute name="onlySymbolic" type="xs:boolean" use="optional"></xs:attribute>
  <xs:attribute name="onlyDatetime" type="xs:boolean" use="optional"></xs:attribute>
  <xs:attribute name="types" type="xs:string" use="optional"></xs:attribute>
  <xs:attribute name="onlyRanges" type="xs:boolean" use="optional"></xs:attribute>
  <xs:attribute name="onlyDiscrete" type="xs:boolean" use="optional"></xs:attribute>
</xs:element>

```

### Parent Éléments

[PropertiesPanel Élément](#), [PropertiesSubPanel Élément](#)

### Enfant Éléments

[Enabled Élément](#), [Layout Élément](#), [Visible Élément](#)

**Éléments connexes Eléments**

CheckBoxControl Elément, CheckBoxGroupControl Elément, ClientDirectoryChooserControl Elément, ClientFileChooserControl Elément, DBConnectionChooserControl Elément, DBTableChooserControl Elément, PasswordBoxControl Elément, PropertyControl Elément, RadioButtonGroupControl Elément, ServerDirectoryChooserControl Elément, ServerFileChooserControl Elément, SingleFieldChooserControl Elément, SingleFieldValueChooserControl Elément, SpinnerControl Elément, TableControl Elément, TextAreaControl Elément, TextBoxControl Elément

**MultitemChooserControl Elément**

Table A-100  
Attributs pour MultitemChooserControl

Attribut	Utiliser	Description	Valeurs valides
catalog	<b>obligatoire</b>		<i>string</i>
description	optional		<i>string</i>
descriptionKey	optional		<i>string</i>
label	optional		<i>string</i>
labelAbove	optional		<i>boolean</i>
labelKey	optional		<i>string</i>
labelWidth	optional		<i>positiveInteger</i>
mnemonic	optional		<i>string</i>
mnemonicKey	optional		<i>string</i>
property	<b>obligatoire</b>		<i>string</i>
showLabel	optional		<i>boolean</i>

**Représentation XML**

```
<xs:element name="MultitemChooserControl">
  <xs:sequence>
    <xs:choice>
      <xs:element ref="Layout" minOccurs="0" maxOccurs="1"></xs:element>
      <xs:element ref="Enabled" minOccurs="0" maxOccurs="1"></xs:element>
      <xs:element ref="Visible" minOccurs="0" maxOccurs="1"></xs:element>
    </xs:choice>
  </xs:sequence>
  <xs:attribute name="property" type="xs:string" use="required"></xs:attribute>
  <xs:attribute name="showLabel" type="xs:boolean" use="optional" default="true"></xs:attribute>
  <xs:attribute name="label" type="xs:string" use="optional"></xs:attribute>
  <xs:attribute name="labelKey" type="xs:string" use="optional"></xs:attribute>
  <xs:attribute name="mnemonic" type="xs:string" use="optional"></xs:attribute>
  <xs:attribute name="mnemonicKey" type="xs:string" use="optional"></xs:attribute>
  <xs:attribute name="labelWidth" type="xs:positiveInteger" use="optional" default="1"></xs:attribute>
  <xs:attribute name="labelAbove" type="xs:boolean" use="optional" default="false"></xs:attribute>
  <xs:attribute name="description" type="xs:string" use="optional"></xs:attribute>
  <xs:attribute name="descriptionKey" type="xs:string" use="optional"></xs:attribute>
  <xs:attribute name="catalog" type="xs:string" use="required"></xs:attribute>
</xs:element>
```

**Parent Eléments**

[PropertiesPanel Elément](#), [PropertiesSubPanel Elément](#)

**Enfant Eléments**

[Enabled Elément](#), [Layout Elément](#), [Visible Elément](#)

**Eléments connexes Eléments**

[SingleItemChooserControl Elément](#)

**Node Elément**

Table A-101  
Attributs pour Node

Attribut	Utiliser	Description	Valeurs valides
deprecatedScriptNames	optional		<i>string</i>
description	optional		<i>string</i>
descriptionKey	optional		<i>string</i>
helpLink	optional		<i>string</i>
id	<b>obligatoire</b>		<i>string</i>
label	<b>obligatoire</b>		<i>string</i>
labelKey	optional		<i>string</i>
palette	optional		<b>import</b> <b>fieldOp</b> <b>recordOp</b> <b>modeling</b> <b>dbModeling</b> <b>graph</b> <b>output</b> <b>export</b> <b>modeling.classification</b> <b>modeling.association</b> <b>modeling.segmentation</b> <b>modeling.auto</b>
relativePosition	optional		<i>string</i>
relativeTo	optional		<i>string</i>
scriptName	optional		<i>string</i>
type	<b>obligatoire</b>		<b>dataReader</b> <b>dataWriter</b> <b>dataTransformer</b> <b>modelApplier</b> <b>modelBuilder</b> <b>documentBuilder</b>

**Représentation XML**

```
<xs:element name="Node">
  <xs:sequence maxOccurs="unbounded">
    <xs:choice maxOccurs="unbounded">
      <xs:element ref="Properties"></xs:element>
      <xs:element name="Containers" minOccurs="0">
```

```

        <xs:sequence maxOccurs="unbounded">
            <xs:element ref="Container"></xs:element>
        </xs:sequence>
    </xs:element>
    <xs:element ref="UserInterface"></xs:element>
    <xs:element ref="Constructors" minOccurs="0"></xs:element>
    <xs:element ref="ModelProvider" minOccurs="0"></xs:element>
</xs:choice>
</xs:sequence>
<xs:attribute name="id" type="xs:string" use="required"></xs:attribute>
<xs:attribute name="scriptName" type="xs:string" use="optional"></xs:attribute>
<xs:attribute name="deprecatedScriptNames" type="xs:string" use="optional"></xs:attribute>
<xs:sequence>
    <xs:element ref="ModelBuilder" minOccurs="0"></xs:element>
    <xs:element ref="DocumentBuilder" minOccurs="0"></xs:element>
    <xs:element ref="Execution"></xs:element>
    <xs:element ref="OutputDataModel" minOccurs="0"></xs:element>
</xs:sequence>
<xs:attribute name="type" type="NODE-TYPE" use="required">
    <xs:enumeration value="dataReader"></xs:enumeration>
    <xs:enumeration value="dataWriter"></xs:enumeration>
    <xs:enumeration value="dataTransformer"></xs:enumeration>
    <xs:enumeration value="modelApplier"></xs:enumeration>
    <xs:enumeration value="modelBuilder"></xs:enumeration>
    <xs:enumeration value="documentBuilder"></xs:enumeration>
</xs:attribute>
<xs:attribute name="label" type="xs:string" use="required"></xs:attribute>
<xs:attribute name="labelKey" type="xs:string" use="optional"></xs:attribute>
<xs:attribute name="description" type="xs:string" use="optional"></xs:attribute>
<xs:attribute name="descriptionKey" type="xs:string" use="optional"></xs:attribute>
<xs:attribute name="palette" type="SYSTEM-PALETTE" use="optional">
    <xs:enumeration value="import"></xs:enumeration>
    <xs:enumeration value="fieldOp"></xs:enumeration>
    <xs:enumeration value="recordOp"></xs:enumeration>
    <xs:enumeration value="modeling"></xs:enumeration>
    <xs:enumeration value="dbModeling"></xs:enumeration>
    <xs:enumeration value="graph"></xs:enumeration>
    <xs:enumeration value="output"></xs:enumeration>
    <xs:enumeration value="export"></xs:enumeration>
    <xs:enumeration value="modeling.classification"></xs:enumeration>
    <xs:enumeration value="modeling.association"></xs:enumeration>
    <xs:enumeration value="modeling.segmentation"></xs:enumeration>
    <xs:enumeration value="modeling.auto"></xs:enumeration>
</xs:attribute>
<xs:attribute name="helpLink" type="xs:string" use="optional"></xs:attribute>
<xs:attribute name="relativeTo" type="xs:string" use="optional"></xs:attribute>
<xs:attribute name="relativePosition" type="xs:string" use="optional"></xs:attribute>
</xs:element>

```

### **Parent Elements**

#### **Extension Element**

**Enfant Eléments**

[Constructors Elément](#), [Containers Elément](#), [DocumentBuilder Elément](#), [Execution Elément](#), [ModelBuilder Elément](#), [ModelProvider Elément](#), [OutputDataModel Elément](#), [Properties Elément](#), [UserInterface Elément](#)

**Eléments connexes Eléments**

[DocumentOutput Elément](#), [InteractiveDocumentBuilder Elément](#), [InteractiveModelBuilder Elément](#), [ModelOutput Elément](#)

**Containers Elément****Représentation XML**

```
<xs:element name="Containers" minOccurs="0">
  <xs:sequence maxOccurs="unbounded">
    <xs:element ref="Container"></xs:element>
  </xs:sequence>
</xs:element>
```

**Parent Eléments**

[Node Elément](#)

**Enfant Eléments**

[Container Elément](#)

**Not Elément****Représentation XML**

```
<xs:element name="Not">
  <xs:sequence>
    <xs:group ref="CONDITION-EXPRESSION">
      <xs:choice>
        <xs:element ref="Condition"></xs:element>
        <xs:element ref="And"></xs:element>
        <xs:element ref="Or"></xs:element>
        <xs:element ref="Not"></xs:element>
      </xs:choice>
    </xs:group>
  </xs:sequence>
</xs:element>
```

**Parent Eléments**

[And Elément](#), [Command Elément](#), [Constraint Elément](#), [CreateDocument Elément](#), [CreateDocumentOutput Elément](#), [CreateInteractiveDocumentBuilder Elément](#), [CreateInteractiveModelBuilder Elément](#), [CreateModel Elément](#), [CreateModelApplier Elément](#),

[CreateModelOutput Élément](#), [Enabled Élément](#), [Option Élément](#), [Or Élément](#), [Run Élément](#), [Visible Élément](#)

### **Enfant Éléments**

[And Élément](#), [Condition Élément](#), [Not Élément](#), [Or Élément](#)

## **NumberFormat Élément**

Defines format information for a numeric field.

Table A-102

Attributs pour NumberFormat

Attribut	Utiliser	Description	Valeurs valides
decimalPlaces	<b>obligatoire</b>		<i>nonNegativeInteger</i>
decimalSymbol	<b>obligatoire</b>		<b>period</b> <b>comma</b>
formatType	<b>obligatoire</b>		<b>standard</b> <b>scientific</b> <b>currency</b>
groupingSymbol	<b>obligatoire</b>		<b>none</b> <b>period</b> <b>comma</b> <b>space</b>
name	<b>obligatoire</b>		<i>string</i>

### **Représentation XML**

```
<xs:element name="NumberFormat" type="NUMBER-FORMAT-DECLARATION">
  <xs:attribute name="name" type="xs:string" use="required"/></xs:attribute>
  <xs:attribute name="formatType" type="NUMBER-FORMAT-TYPE" use="required">
    <xs:enumeration value="standard"/></xs:enumeration>
    <xs:enumeration value="scientific"/></xs:enumeration>
    <xs:enumeration value="currency"/></xs:enumeration>
  </xs:attribute>
  <xs:attribute name="decimalPlaces" type="xs:nonNegativeInteger" use="required"/></xs:attribute>
  <xs:attribute name="decimalSymbol" type="DECIMAL-SYMBOL" use="required">
    <xs:enumeration value="period"/></xs:enumeration>
    <xs:enumeration value="comma"/></xs:enumeration>
  </xs:attribute>
  <xs:attribute name="groupingSymbol" type="NUMBER-GROUPING-SYMBOL" use="required">
    <xs:enumeration value="none"/></xs:enumeration>
    <xs:enumeration value="period"/></xs:enumeration>
    <xs:enumeration value="comma"/></xs:enumeration>
    <xs:enumeration value="space"/></xs:enumeration>
  </xs:attribute>
</xs:element>
```



## NumericInfo Elément

Table A-103  
Attributs pour NumericInfo

Attribut	Utiliser	Description	Valeurs valides
mean	facultatif		double
standardDeviation	facultatif		double

### Représentation XML

```
<xs:element name="NumericInfo">
  <xs:attribute name="mean" type="xs:double"></xs:attribute>
  <xs:attribute name="standardDeviation" type="xs:double"></xs:attribute>
</xs:element>
```

### Parent Eléments

[AddField Elément](#), [ChangeField Elément](#), [Field Elément](#)

## Option Elément

Table A-104  
Attributs pour Option

Attribut	Utiliser	Description	Valeurs valides
ifProperty	optional		string
unlessProperty	optional		string
value	<b>obligatoire</b>		

### Représentation XML

```
<xs:element name="Option">
  <xs:sequence>
    <xs:group ref="CONDITION-EXPRESSION" minOccurs="0">
      <xs:choice>
        <xs:element ref="Condition"></xs:element>
        <xs:element ref="And"></xs:element>
        <xs:element ref="Or"></xs:element>
        <xs:element ref="Not"></xs:element>
      </xs:choice>
    </xs:group>
  </xs:sequence>
  <xs:attribute name="value" type="EVALUATED-STRING" use="required"></xs:attribute>
  <xs:attribute name="ifProperty" type="xs:string" use="optional"></xs:attribute>
  <xs:attribute name="unlessProperty" type="xs:string" use="optional"></xs:attribute>
</xs:element>
```

### Parent Eléments

[Run Elément](#)

### Enfant Eléments

[And Elément](#), [Condition Elément](#), [Not Elément](#), [Or Elément](#)

## OptionCode Élément

Table A-105  
Attributs pour OptionCode

Attribut	Utiliser	Description	Valeurs valides
code	facultatif		<i>long</i>
description	facultatif		<i>string</i>
type	facultatif		<b>mandatory optional</b>

### Représentation XML

```
<xs:element name="OptionCode">
  <xs:attribute name="code" type="xs:long"></xs:attribute>
  <xs:attribute name="type" type="LicenseType">
    <xs:enumeration value="mandatory"></xs:enumeration>
    <xs:enumeration value="optional"></xs:enumeration>
  </xs:attribute>
  <xs:attribute name="description" type="xs:string"></xs:attribute>
</xs:element>
```

### Parent Elements

[License Élément](#)

## Or Élément

### Représentation XML

```
<xs:element name="Or">
  <xs:sequence minOccurs="2" maxOccurs="unbounded">
    <xs:group ref="CONDITION-EXPRESSION">
      <xs:choice>
        <xs:element ref="Condition"></xs:element>
        <xs:element ref="And"></xs:element>
        <xs:element ref="Or"></xs:element>
        <xs:element ref="Not"></xs:element>
      </xs:choice>
    </xs:group>
  </xs:sequence>
</xs:element>
```

### Parent Elements

[And Élément](#), [Command Élément](#), [Constraint Élément](#), [CreateDocument Élément](#), [CreateDocumentOutput Élément](#), [CreateInteractiveDocumentBuilder Élément](#), [CreateInteractiveModelBuilder Élément](#), [CreateModel Élément](#), [CreateModelApplier Élément](#), [CreateModelOutput Élément](#), [Enabled Élément](#), [Not Élément](#), [Option Élément](#), [Run Élément](#), [Visible Élément](#)

**Enfant Eléments**

[And Elément](#), [Condition Elément](#), [Not Elément](#), [Or Elément](#)

**OutputDataModel Elément**

Table A-106

Attributs pour OutputDataModel

Attribut	Utiliser	Description	Valeurs valides
libraryId	optional		<i>string</i>
method	optional		<b>xml</b> <b>dataModelProvider</b> <b>sharedLibrary</b>
mode	optional		<b>fixed</b> <b>modify</b> <b>extend</b> <b>replace</b>
providerClass	optional		<i>string</i>

**Représentation XML**

```

<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>
        <xs:enumeration value="modify"></xs:enumeration>
        <xs:enumeration value="extend"></xs:enumeration>
        <xs:enumeration value="replace"></xs:enumeration>
      </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>
        <xs:enumeration value="dataModelProvider"></xs:enumeration>
        <xs:enumeration value="sharedLibrary"></xs:enumeration>
      </xs:restriction>
    </xs:simpleType>
  </xs:attribute>
  <xs:attribute name="providerClass" type="xs:string" use="optional"></xs:attribute>
  <xs:attribute name="libraryId" type="xs:string" use="optional"></xs:attribute>
</xs:element>

```

**Parent Eléments**

[Node Elément](#)

## OutputFiles Élément

### Représentation XML

```
<xs:element name="OutputFiles">
  <xs:group ref="RUNTIME-FILES">
    <xs:sequence>
      <xs:element ref="DataFile"></xs:element>
      <xs:element ref="ContainerFile" minOccurs="0" maxOccurs="unbounded"></xs:element>
    </xs:sequence>
  </xs:group>
</xs:element>
```

### Parent Elements

[Execution Élément](#), [Module Élément](#)

### Enfant Elements

[ContainerFile Élément](#), [DataFile Élément](#)

## Palette Élément

Table A-107  
Attributs pour Palette

Attribut	Utiliser	Description	Valeurs valides
description	optional		<i>string</i>
descriptionKey	optional		<i>string</i>
id	<b>obligatoire</b>		<i>string</i>
label	<b>obligatoire</b>		<i>string</i>
labelKey	optional		<i>string</i>
position	optional		<b>atStart</b> <b>atEnd</b> <b>before</b> <b>after</b>
systemPalette	optional		<b>import</b> <b>fieldOp</b> <b>recordOp</b> <b>modeling</b> <b>dbModeling</b> <b>graph</b> <b>output</b> <b>export</b> <b>modeling.classification</b> <b>modeling.association</b> <b>modeling.segmentation</b> <b>modeling.auto</b>

### Représentation XML

```
<xs:element name="Palette">
  <xs:sequence>
    <xs:element ref="Icon"></xs:element>
  </xs:sequence>
</xs:element>
```

```

</xs:sequence>
<xs:attribute name="id" type="xs:string" use="required"></xs:attribute>
<xs:attribute name="label" type="xs:string" use="required"></xs:attribute>
<xs:attribute name="labelKey" type="xs:string" use="optional"></xs:attribute>
<xs:attribute name="description" type="xs:string" use="optional"></xs:attribute>
<xs:attribute name="descriptionKey" type="xs:string" use="optional"></xs:attribute>
<xs:attribute name="position" use="optional">
  <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:enumeration value="atStart"></xs:enumeration>
      <xs:enumeration value="atEnd"></xs:enumeration>
      <xs:enumeration value="before"></xs:enumeration>
      <xs:enumeration value="after"></xs:enumeration>
    </xs:restriction>
  </xs:simpleType>
</xs:attribute>
<xs:attribute name="systemPalette" type="SYSTEM-PALETTE" use="optional">
  <xs:enumeration value="import"></xs:enumeration>
  <xs:enumeration value="fieldOp"></xs:enumeration>
  <xs:enumeration value="recordOp"></xs:enumeration>
  <xs:enumeration value="modeling"></xs:enumeration>
  <xs:enumeration value="dbModeling"></xs:enumeration>
  <xs:enumeration value="graph"></xs:enumeration>
  <xs:enumeration value="output"></xs:enumeration>
  <xs:enumeration value="export"></xs:enumeration>
  <xs:enumeration value="modeling.classification"></xs:enumeration>
  <xs:enumeration value="modeling.association"></xs:enumeration>
  <xs:enumeration value="modeling.segmentation"></xs:enumeration>
  <xs:enumeration value="modeling.auto"></xs:enumeration>
</xs:attribute>
</xs:element>

```

### Enfant Eléments

[Icon Elément](#)

## Parameters Elément

Configuration parameters from the extension node.

Table A-108

Attributs pour Parameters

Attribut	Utiliser	Description	Valeurs valides
count	facultatif		<i>nonNegativeInteger</i>

### Représentation XML

```

<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>
          <xs:element ref="StructuredValue"></xs:element>
          <xs:element ref="ListValue"></xs:element>
        </xs:choice>
      </xs:group>
    </xs:element>
  </xs:sequence>
</xs:element>

```

```

        <xs:element ref="Value"></xs:element>
        <xs:element ref="DatabaseConnectionValue"></xs:element>
    </xs:choice>
</xs:group>
</xs:element>
</xs:sequence>
<xs:attribute name="count" type="xs:nonNegativeInteger"></xs:attribute>
</xs:element>

```

### Enfant Eléments

#### Parameter Elément

### Parameter Elément

A parameter has a name and a value. A simple value can be expressed with the value attribute; a compound value uses the content model described by ParameterContent. This combination of attribute and content is repeated for nested values.

Table A-109  
Attributs pour Parameter

Attribut	Utiliser	Description	Valeurs valides
name	<b>obligatoire</b>		<i>string</i>
value	facultatif		<i>string</i>

### Représentation XML

```

<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>
      <xs:element ref="StructuredValue"></xs:element>
      <xs:element ref="ListValue"></xs:element>
      <xs:element ref="Value"></xs:element>
      <xs:element ref="DatabaseConnectionValue"></xs:element>
    </xs:choice>
  </xs:group>
  <xs:attribute name="name" type="xs:string" use="required"></xs:attribute>
  <xs:attribute name="value" type="xs:string"></xs:attribute>
</xs:element>

```

### Parent Eléments

#### Parameters Elément

### Enfant Eléments

[DatabaseConnectionValue Elément](#), [ListValue Elément](#), [MapValue Elément](#), [StructuredValue Elément](#), [Value Elément](#)

## PasswordBoxControl Élément

Table A-110  
Attributs pour PasswordBoxControl

Attribut	Utiliser	Description	Valeurs valides
columns	optional		<i>positiveInteger</i>
description	optional		<i>string</i>
descriptionKey	optional		<i>string</i>
label	optional		<i>string</i>
labelAbove	optional		<i>boolean</i>
labelKey	optional		<i>string</i>
labelWidth	optional		<i>positiveInteger</i>
mnemonic	optional		<i>string</i>
mnemonicKey	optional		<i>string</i>
property	<b>obligatoire</b>		<i>string</i>
showLabel	optional		<i>boolean</i>

### Représentation XML

```
<xs:element name="PasswordBoxControl">
  <xs:sequence>
    <xs:choice>
      <xs:element ref="Layout" minOccurs="0" maxOccurs="1"/></xs:element>
      <xs:element ref="Enabled" minOccurs="0" maxOccurs="1"/></xs:element>
      <xs:element ref="Visible" minOccurs="0" maxOccurs="1"/></xs:element>
    </xs:choice>
  </xs:sequence>
  <xs:attribute name="property" type="xs:string" use="required"/></xs:attribute>
  <xs:attribute name="showLabel" type="xs:boolean" use="optional" default="true"/></xs:attribute>
  <xs:attribute name="label" type="xs:string" use="optional"/></xs:attribute>
  <xs:attribute name="labelKey" type="xs:string" use="optional"/></xs:attribute>
  <xs:attribute name="mnemonic" type="xs:string" use="optional"/></xs:attribute>
  <xs:attribute name="labelWidth" type="xs:positiveInteger" use="optional" default="1"/></xs:attribute>
  <xs:attribute name="labelAbove" type="xs:boolean" use="optional" default="false"/></xs:attribute>
  <xs:attribute name="description" type="xs:string" use="optional"/></xs:attribute>
  <xs:attribute name="descriptionKey" type="xs:string" use="optional"/></xs:attribute>
  <xs:attribute name="columns" type="xs:positiveInteger" use="optional" default="20"/></xs:attribute>
</xs:element>
```

### Parent Éléments

[PropertiesPanel Élément](#), [PropertiesSubPanel Élément](#)

### Enfant Éléments

[Enabled Élément](#), [Layout Élément](#), [Visible Élément](#)

### Éléments connexes Éléments

[CheckBoxControl Élément](#), [CheckBoxGroupControl Élément](#), [ClientDirectoryChooserControl Élément](#), [ClientFileChooserControl Élément](#), [DBConnectionChooserControl Élément](#), [DBTableChooserControl Élément](#), [MultiFieldChooserControl Élément](#), [PropertyControl](#)

Élément, RadioButtonGroupControl Élément, ServerDirectoryChooserControl Élément, ServerFileChooserControl Élément, SingleFieldChooserControl Élément, SingleFieldValueChooserControl Élément, SpinnerControl Élément, TableControl Élément, TextAreaControl Élément, TextBoxControl Élément

## Properties Élément

### Représentation XML

```
<xs:element name="Properties">
  <xs:sequence>
    <xs:element ref="Property" minOccurs="0" maxOccurs="unbounded"/></xs:element>
  </xs:sequence>
</xs:element>
```

### Parent Elements

DocumentOutput Élément, Execution Élément, InteractiveDocumentBuilder Élément, InteractiveModelBuilder Élément, ModelOutput Élément, Node Élément

### Enfant Elements

Property Élément

## PropertiesPanel Élément

Table A-111  
Attributs pour PropertiesPanel

Attribut	Utiliser	Description	Valeurs valides
id	optional		string
label	optional		string
labelKey	optional		string

### Représentation XML

```
<xs:element name="PropertiesPanel">
  <xs:sequence>
    <xs:choice>
      <xs:element ref="Layout" minOccurs="0" maxOccurs="1"/></xs:element>
      <xs:element ref="Enabled" minOccurs="0" maxOccurs="1"/></xs:element>
      <xs:element ref="Visible" minOccurs="0" maxOccurs="1"/></xs:element>
    </xs:choice>
  </xs:sequence>
  <xs:sequence maxOccurs="unbounded">
    <xs:choice>
      <xs:element ref="CheckBoxControl"/></xs:element>
      <xs:element ref="TextBoxControl"/></xs:element>
      <xs:element ref="PasswordBoxControl"/></xs:element>
      <xs:element ref="TextAreaControl"/></xs:element>
      <xs:element ref="RadioButtonGroupControl"/></xs:element>
      <xs:element ref="CheckBoxGroupControl"/></xs:element>
      <xs:element ref="ComboBoxControl"/></xs:element>
    </xs:choice>
  </xs:sequence>
</xs:element>
```



```

<xs:element ref="SpinnerControl"/></xs:element>
<xs:element ref="ServerFileChooserControl"/></xs:element>
<xs:element ref="ServerDirectoryChooserControl"/></xs:element>
<xs:element ref="ClientFileChooserControl"/></xs:element>
<xs:element ref="ClientDirectoryChooserControl"/></xs:element>
<xs:element ref="TableControl"/></xs:element>
<xs:element ref="SingleFieldChooserControl"/></xs:element>
<xs:element ref="MultiFieldChooserControl"/></xs:element>
<xs:element ref="SingleFieldValueChooserControl"/></xs:element>
<xs:element ref="SingleItemChooserControl"/></xs:element>
<xs:element ref="MultiItemChooserControl"/></xs:element>
<xs:element ref="DBConnectionChooserControl"/></xs:element>
<xs:element ref="DBTableChooserControl"/></xs:element>
<xs:element ref="PropertyControl"/></xs:element>
<xs:element ref="StaticText"/></xs:element>
<xs:element ref="SystemControls"/></xs:element>
<xs:element ref="ActionButton"/></xs:element>
<xs:element ref="PropertiesPanel"/></xs:element>
<xs:element ref="PropertiesSubPanel"/></xs:element>
<xs:element ref="SelectorPanel"/></xs:element>
<xs:element ref="ExtensionObjectPanel"/></xs:element>
</xs:choice>
</xs:sequence>
<xs:attribute name="id" type="xs:string" use="optional"/></xs:attribute>
<xs:attribute name="label" type="xs:string" use="optional"/></xs:attribute>
<xs:attribute name="labelKey" type="xs:string" use="optional"/></xs:attribute>
</xs:element>

```

### **Parent Éléments**

[PropertiesSubPanel Élément](#), [Tab Élément](#)

### **Enfant Éléments**

[ActionButton Élément](#), [CheckBoxControl Élément](#), [CheckBoxGroupControl Élément](#), [ClientDirectoryChooserControl Élément](#), [ClientFileChooserControl Élément](#), [ComboBoxControl Élément](#), [DBConnectionChooserControl Élément](#), [DBTableChooserControl Élément](#), [Enabled Élément](#), [ExtensionObjectPanel Élément](#), [Layout Élément](#), [MultiFieldChooserControl Élément](#), [MultiItemChooserControl Élément](#), [PasswordBoxControl Élément](#), [PropertiesPanel Élément](#), [PropertiesSubPanel Élément](#), [PropertyControl Élément](#), [RadioButtonGroupControl Élément](#), [SelectorPanel Élément](#), [ServerDirectoryChooserControl Élément](#), [ServerFileChooserControl Élément](#), [SingleFieldChooserControl Élément](#), [SingleFieldValueChooserControl Élément](#), [SingleItemChooserControl Élément](#), [SpinnerControl Élément](#), [StaticText Élément](#), [SystemControls Élément](#), [TableControl Élément](#), [TextAreaControl Élément](#), [TextBoxControl Élément](#), [Visible Élément](#)

### **Éléments connexes Éléments**

[PropertiesSubPanel Élément](#)

## PropertiesSubPanel Elément

Table A-112  
Attributs pour PropertiesSubPanel

Attribut	Utiliser	Description	Valeurs valides
buttonLabel	optional		string
buttonLabelKey	optional		string
dialogTitle	optional		string
dialogTitleKey	optional		string
helpLink	optional		string
mnemonic	optional		string
mnemonicKey	optional		string

### Représentation XML

```
<xs:element name="PropertiesSubPanel">
  <xs:sequence>
    <xs:choice>
      <xs:element ref="Layout" minOccurs="0" maxOccurs="1"></xs:element>
      <xs:element ref="Enabled" minOccurs="0" maxOccurs="1"></xs:element>
      <xs:element ref="Visible" minOccurs="0" maxOccurs="1"></xs:element>
    </xs:choice>
  </xs:sequence>
  <xs:sequence maxOccurs="unbounded">
    <xs:choice>
      <xs:element ref="CheckBoxControl"></xs:element>
      <xs:element ref="TextBoxControl"></xs:element>
      <xs:element ref="PasswordBoxControl"></xs:element>
      <xs:element ref="TextAreaControl"></xs:element>
      <xs:element ref="RadioButtonGroupControl"></xs:element>
      <xs:element ref="CheckBoxGroupControl"></xs:element>
      <xs:element ref="ComboBoxControl"></xs:element>
      <xs:element ref="SpinnerControl"></xs:element>
      <xs:element ref="ServerFileChooserControl"></xs:element>
      <xs:element ref="ServerDirectoryChooserControl"></xs:element>
      <xs:element ref="ClientFileChooserControl"></xs:element>
      <xs:element ref="ClientDirectoryChooserControl"></xs:element>
      <xs:element ref="TableControl"></xs:element>
      <xs:element ref="SingleFieldChooserControl"></xs:element>
      <xs:element ref="MultiFieldChooserControl"></xs:element>
      <xs:element ref="SingleFieldValueChooserControl"></xs:element>
      <xs:element ref="SingleItemChooserControl"></xs:element>
      <xs:element ref="MultiItemChooserControl"></xs:element>
      <xs:element ref="DBConnectionChooserControl"></xs:element>
      <xs:element ref="DBTableChooserControl"></xs:element>
      <xs:element ref="PropertyControl"></xs:element>
      <xs:element ref="StaticText"></xs:element>
      <xs:element ref="SystemControls"></xs:element>
      <xs:element ref="ActionButton"></xs:element>
      <xs:element ref="PropertiesPanel"></xs:element>
      <xs:element ref="PropertiesSubPanel"></xs:element>
      <xs:element ref="SelectorPanel"></xs:element>
      <xs:element ref="ExtensionObjectPanel"></xs:element>
    </xs:choice>
  </xs:sequence>
</xs:element>
```

```

<xs:attribute name="buttonLabel" type="xs:string" use="optional"/></xs:attribute>
<xs:attribute name="buttonLabelKey" type="xs:string" use="optional"/></xs:attribute>
<xs:attribute name="mnemonic" type="xs:string" use="optional"/></xs:attribute>
<xs:attribute name="mnemonicKey" type="xs:string" use="optional"/></xs:attribute>
<xs:attribute name="dialogTitle" type="xs:string" use="optional"/></xs:attribute>
<xs:attribute name="dialogTitleKey" type="xs:string" use="optional"/></xs:attribute>
<xs:attribute name="helpLink" type="xs:string" use="optional"/></xs:attribute>
</xs:element>

```

### Parent Eléments

[PropertiesPanel Elément](#)

### Enfant Eléments

[ActionButton Elément](#), [CheckBoxControl Elément](#), [CheckBoxGroupControl Elément](#), [ClientDirectoryChooserControl Elément](#), [ClientFileChooserControl Elément](#), [ComboBoxControl Elément](#), [DBConnectionChooserControl Elément](#), [DBTableChooserControl Elément](#), [Enabled Elément](#), [ExtensionObjectPanel Elément](#), [Layout Elément](#), [MultiFieldChooserControl Elément](#), [MultiItemChooserControl Elément](#), [PasswordBoxControl Elément](#), [PropertiesPanel Elément](#), [PropertiesSubPanel Elément](#), [PropertyControl Elément](#), [RadioButtonGroupControl Elément](#), [SelectorPanel Elément](#), [ServerDirectoryChooserControl Elément](#), [ServerFileChooserControl Elément](#), [SingleFieldChooserControl Elément](#), [SingleFieldValueChooserControl Elément](#), [SingleItemChooserControl Elément](#), [SpinnerControl Elément](#), [StaticText Elément](#), [SystemControls Elément](#), [TableControl Elément](#), [TextAreaControl Elément](#), [TextBoxControl Elément](#), [Visible Elément](#)

### Eléments connexes Eléments

[PropertiesPanel Elément](#)

## Property Elément

Table A-113  
Attributs pour Property

Attribut	Utiliser	Description	Valeurs valides
defaultValue	optional		
defaultValueKey	optional		string
deprecatedScriptNames	optional		string
description	optional		string
descriptionKey	optional		string
isList	optional		boolean
label	optional		string
labelKey	optional		string
max	optional		string
min	optional		string
name	<b>obligatoire</b>		string
scriptName	optional		string

Attribut	Utiliser	Description	Valeurs valides
type	optional		<i>string</i>
valueType	facultatif		<b>string</b> <b>encryptedString</b> <b>fieldName</b> <b>integer</b> <b>double</b> <b>boolean</b> <b>date</b> <b>enum</b> <b>structure</b> <b>databaseConnection</b>

### Représentation XML

```

<xs:element name="Property">
  <xs:choice>
    <xs:element ref="DefaultValue" minOccurs="0"/></xs:element>
  </xs:choice>
  <xs:attribute name="valueType" type="PROPERTY-VALUE-TYPE">
    <xs:enumeration value="string"/></xs:enumeration>
    <xs:enumeration value="encryptedString"/></xs:enumeration>
    <xs:enumeration value="fieldName"/></xs:enumeration>
    <xs:enumeration value="integer"/></xs:enumeration>
    <xs:enumeration value="double"/></xs:enumeration>
    <xs:enumeration value="boolean"/></xs:enumeration>
    <xs:enumeration value="date"/></xs:enumeration>
    <xs:enumeration value="enum"/></xs:enumeration>
    <xs:enumeration value="structure"/></xs:enumeration>
    <xs:enumeration value="databaseConnection"/></xs:enumeration>
  </xs:attribute>
  <xs:attribute name="isList" type="xs:boolean" use="optional" default="false"/></xs:attribute>
  <xs:attribute name="min" type="xs:string" use="optional"/></xs:attribute>
  <xs:attribute name="max" type="xs:string" use="optional"/></xs:attribute>
  <xs:attribute name="name" type="xs:string" use="required"/></xs:attribute>
  <xs:attribute name="scriptName" type="xs:string" use="optional"/></xs:attribute>
  <xs:attribute name="deprecatedScriptNames" type="xs:string" use="optional"/></xs:attribute>
  <xs:attribute name="type" type="xs:string" use="optional"/></xs:attribute>
  <xs:attribute name="defaultValue" type="EVALUATED-STRING" use="optional"/></xs:attribute>
  <xs:attribute name="defaultValueKey" type="xs:string" use="optional"/></xs:attribute>
  <xs:attribute name="label" type="xs:string" use="optional"/></xs:attribute>
  <xs:attribute name="labelKey" type="xs:string" use="optional"/></xs:attribute>
  <xs:attribute name="description" type="xs:string" use="optional"/></xs:attribute>
  <xs:attribute name="descriptionKey" type="xs:string" use="optional"/></xs:attribute>
</xs:element>

```

### Parent Eléments

[Properties Elément](#), [PropertySets Elément](#)

### Enfant Eléments

[DefaultValue Elément](#)

**Eléments connexes Eléments**[PropertyType Elément](#)**PropertyControl Elément**Table A-114  
Attributs pour PropertyControl

Attribut	Utiliser	Description	Valeurs valides
controlClass	<b>obligatoire</b>		<i>string</i>
description	optional		<i>string</i>
descriptionKey	optional		<i>string</i>
label	optional		<i>string</i>
labelAbove	optional		<i>boolean</i>
labelKey	optional		<i>string</i>
labelWidth	optional		<i>positiveInteger</i>
mnemonic	optional		<i>string</i>
mnemonicKey	optional		<i>string</i>
property	<b>obligatoire</b>		<i>string</i>
showLabel	optional		<i>boolean</i>

**Représentation XML**

```

<xs:element name="PropertyControl">
  <xs:sequence>
    <xs:choice>
      <xs:element ref="Layout" minOccurs="0" maxOccurs="1"></xs:element>
      <xs:element ref="Enabled" minOccurs="0" maxOccurs="1"></xs:element>
      <xs:element ref="Visible" minOccurs="0" maxOccurs="1"></xs:element>
    </xs:choice>
  </xs:sequence>
  <xs:attribute name="property" type="xs:string" use="required"></xs:attribute>
  <xs:attribute name="showLabel" type="xs:boolean" use="optional" default="true"></xs:attribute>
  <xs:attribute name="label" type="xs:string" use="optional"></xs:attribute>
  <xs:attribute name="labelKey" type="xs:string" use="optional"></xs:attribute>
  <xs:attribute name="mnemonic" type="xs:string" use="optional"></xs:attribute>
  <xs:attribute name="mnemonicKey" type="xs:string" use="optional"></xs:attribute>
  <xs:attribute name="labelWidth" type="xs:positiveInteger" use="optional" default="1"></xs:attribute>
  <xs:attribute name="labelAbove" type="xs:boolean" use="optional" default="false"></xs:attribute>
  <xs:attribute name="description" type="xs:string" use="optional"></xs:attribute>
  <xs:attribute name="descriptionKey" type="xs:string" use="optional"></xs:attribute>
  <xs:attribute name="controlClass" type="xs:string" use="required"></xs:attribute>
</xs:element>

```

**Parent Eléments**[PropertiesPanel Elément](#), [PropertiesSubPanel Elément](#)**Enfant Eléments**[Enabled Elément](#), [Layout Elément](#), [Visible Elément](#)

**Eléments connexes Eléments**

[CheckBoxControl Elément](#), [CheckBoxGroupControl Elément](#), [ClientDirectoryChooserControl Elément](#), [ClientFileChooserControl Elément](#), [DBConnectionChooserControl Elément](#), [DBTableChooserControl Elément](#), [MultiFieldChooserControl Elément](#), [PasswordBoxControl Elément](#), [RadioButtonGroupControl Elément](#), [ServerDirectoryChooserControl Elément](#), [ServerFileChooserControl Elément](#), [SingleFieldChooserControl Elément](#), [SingleFieldValueChooserControl Elément](#), [SpinnerControl Elément](#), [TableControl Elément](#), [TextAreaControl Elément](#), [TextBoxControl Elément](#)

**PropertyGroup Elément**

Table A-115  
Attributs pour PropertyGroup

Attribut	Utiliser	Description	Valeurs valides
label	optional		string
labelKey	optional		string
properties	<b>obligatoire</b>		string

**Représentation XML**

```
<xs:element name="PropertyGroup">
  <xs:attribute name="label" type="xs:string" use="optional"/></xs:attribute>
  <xs:attribute name="labelKey" type="xs:string" use="optional"/></xs:attribute>
  <xs:attribute name="properties" type="xs:string" use="required"/></xs:attribute>
</xs:element>
```

**Parent Eléments**

[SimpleSettings Elément](#), [ExpertSettings Elément](#)

**PropertySets Elément****Représentation XML**

```
<xs:element name="PropertySets">
  <xs:sequence>
    <xs:element ref="Property" minOccurs="0" maxOccurs="unbounded"/></xs:element>
  </xs:sequence>
</xs:element>
```

**Parent Eléments**

[CommonObjects Elément](#)

**Enfant Eléments**

[Property Elément](#)

## PropertyType Elément

Table A-116  
Attributs pour PropertyType

Attribut	Utiliser	Description	Valeurs valides
id	<b>obligatoire</b>		<i>string</i>
isKeyed	optional		<i>boolean</i>
isList	optional		<i>boolean</i>
max	optional		<i>string</i>
min	optional		<i>string</i>
valueType	facultatif		<b>string</b> <b>encryptedString</b> <b>fieldName</b> <b>integer</b> <b>double</b> <b>boolean</b> <b>date</b> <b>enum</b> <b>structure</b> <b>databaseConnection</b>

### Représentation XML

```
<xs:element name="PropertyType">
  <xs:choice>
    <xs:element ref="DefaultValue" minOccurs="0"/></xs:element>
  </xs:choice>
  <xs:attribute name="valueType" type="PROPERTY-VALUE-TYPE">
    <xs:enumeration value="string"/></xs:enumeration>
    <xs:enumeration value="encryptedString"/></xs:enumeration>
    <xs:enumeration value="fieldName"/></xs:enumeration>
    <xs:enumeration value="integer"/></xs:enumeration>
    <xs:enumeration value="double"/></xs:enumeration>
    <xs:enumeration value="boolean"/></xs:enumeration>
    <xs:enumeration value="date"/></xs:enumeration>
    <xs:enumeration value="enum"/></xs:enumeration>
    <xs:enumeration value="structure"/></xs:enumeration>
    <xs:enumeration value="databaseConnection"/></xs:enumeration>
  </xs:attribute>
  <xs:attribute name="isList" type="xs:boolean" use="optional" default="false"/></xs:attribute>
  <xs:attribute name="min" type="xs:string" use="optional"/></xs:attribute>
  <xs:attribute name="max" type="xs:string" use="optional"/></xs:attribute>
  <xs:choice>
    <xs:element ref="Enumeration" minOccurs="0"/></xs:element>
    <xs:element ref="Structure" minOccurs="0"/></xs:element>
  </xs:choice>
  <xs:attribute name="id" type="xs:string" use="required"/></xs:attribute>
  <xs:attribute name="isKeyed" type="xs:boolean" use="optional" default="false"/></xs:attribute>
</xs:element>
```

### Parent Eléments

[PropertyTypes Elément](#)

**Enfant Eléments**

[DefaultValue Elément](#), [Enumeration Elément](#), [Structure Elément](#)

**Eléments connexes Eléments**

[Property Elément](#)

**PropertyTypes Elément****Représentation XML**

```
<xs:element name="PropertyTypes">
  <xs:sequence>
    <xs:element ref="PropertyType" minOccurs="0" maxOccurs="unbounded"/></xs:element>
  </xs:sequence>
</xs:element>
```

**Parent Eléments**

[CommonObjects Elément](#)

**Enfant Eléments**

[PropertyType Elément](#)

**RadioButtonGroupControl Elément**

Table A-117

Attributs pour *RadioButtonGroupControl*

Attribut	Utiliser	Description	Valeurs valides
description	optional		<i>string</i>
descriptionKey	optional		<i>string</i>
falseLabel	optional		<i>string</i>
falseLabelKey	optional		<i>string</i>
label	optional		<i>string</i>
labelAbove	optional		<i>boolean</i>
labelKey	optional		<i>string</i>
labelWidth	optional		<i>positiveInteger</i>
layoutByRow	optional		<i>boolean</i>
mnemonic	optional		<i>string</i>
mnemonicKey	optional		<i>string</i>
property	<b>obligatoire</b>		<i>string</i>
rows	optional		<i>positiveInteger</i>
showLabel	optional		<i>boolean</i>
trueFirst	optional		<i>boolean</i>
trueLabel	optional		<i>string</i>
trueLabelKey	optional		<i>string</i>
useSubPanel	optional		<i>boolean</i>



**Représentation XML**

```

<xs:element name="RadioButtonGroupControl">
  <xs:sequence>
    <xs:choice>
      <xs:element ref="Layout" minOccurs="0" maxOccurs="1"></xs:element>
      <xs:element ref="Enabled" minOccurs="0" maxOccurs="1"></xs:element>
      <xs:element ref="Visible" minOccurs="0" maxOccurs="1"></xs:element>
    </xs:choice>
  </xs:sequence>
  <xs:attribute name="property" type="xs:string" use="required"></xs:attribute>
  <xs:attribute name="showLabel" type="xs:boolean" use="optional" default="true"></xs:attribute>
  <xs:attribute name="label" type="xs:string" use="optional"></xs:attribute>
  <xs:attribute name="labelKey" type="xs:string" use="optional"></xs:attribute>
  <xs:attribute name="mnemonic" type="xs:string" use="optional"></xs:attribute>
  <xs:attribute name="mnemonicKey" type="xs:string" use="optional"></xs:attribute>
  <xs:attribute name="labelWidth" type="xs:positiveInteger" use="optional" default="1"></xs:attribute>
  <xs:attribute name="labelAbove" type="xs:boolean" use="optional" default="false"></xs:attribute>
  <xs:attribute name="description" type="xs:string" use="optional"></xs:attribute>
  <xs:attribute name="descriptionKey" type="xs:string" use="optional"></xs:attribute>
  <xs:attribute name="rows" type="xs:positiveInteger" use="optional" default="1"></xs:attribute>
  <xs:attribute name="layoutByRow" type="xs:boolean" use="optional" default="true"></xs:attribute>
  <xs:attribute name="useSubPanel" type="xs:boolean" use="optional" default="true"></xs:attribute>
  <xs:attribute name="falseLabel" type="xs:string" use="optional"></xs:attribute>
  <xs:attribute name="falseLabelKey" type="xs:string" use="optional"></xs:attribute>
  <xs:attribute name="trueLabel" type="xs:string" use="optional"></xs:attribute>
  <xs:attribute name="trueLabelKey" type="xs:string" use="optional"></xs:attribute>
  <xs:attribute name="trueFirst" type="xs:boolean" use="optional" default="false"></xs:attribute>
</xs:element>

```

**Parent Eléments**

[PropertiesPanel Elément](#), [PropertiesSubPanel Elément](#)

**Enfant Eléments**

[Enabled Elément](#), [Layout Elément](#), [Visible Elément](#)

**Eléments connexes Eléments**

[CheckBoxControl Elément](#), [CheckBoxGroupControl Elément](#), [ClientDirectoryChooserControl Elément](#), [ClientFileChooserControl Elément](#), [DBConnectionChooserControl Elément](#), [DBTableChooserControl Elément](#), [MultiFieldChooserControl Elément](#), [PasswordBoxControl Elément](#), [PropertyControl Elément](#), [ServerDirectoryChooserControl Elément](#), [ServerFileChooserControl Elément](#), [SingleFieldChooserControl Elément](#), [SingleFieldValueChooserControl Elément](#), [SpinnerControl Elément](#), [TableControl Elément](#), [TextAreaControl Elément](#), [TextBoxControl Elément](#)

## Range Elément

Table A-118  
Attributs pour Range

Attribut	Utiliser	Description	Valeurs valides
max	facultatif		<i>string</i>
min	facultatif		<i>string</i>

### Représentation XML

```
<xs:element name="Range">
  <xs:attribute name="min" type="xs:string"/></xs:attribute>
  <xs:attribute name="max" type="xs:string"/></xs:attribute>
</xs:element>
```

### Parent Eléments

[AddField Elément](#), [MissingValues Elément](#), [ChangeField Elément](#), [MissingValues Elément](#), [Field Elément](#), [MissingValues Elément](#)

## Range Elément

Table A-119  
Attributs pour Range

Attribut	Utiliser	Description	Valeurs valides
maxValue	<b>obligatoire</b>		<i>string</i>
minValue	<b>obligatoire</b>		<i>string</i>

### Représentation XML

```
<xs:element name="Range" type="RANGE">
  <xs:attribute name="minValue" type="xs:string" use="required"/></xs:attribute>
  <xs:attribute name="maxValue" type="xs:string" use="required"/></xs:attribute>
</xs:element>
```

### Parent Eléments

[Field Elément](#)

## RemoveField Elément

Table A-120  
Attributs pour RemoveField

Attribut	Utiliser	Description	Valeurs valides
fieldRef	<b>obligatoire</b>		

### Représentation XML

```
<xs:element name="RemoveField">
  <xs:attribute name="fieldRef" type="EVALUATED-STRING" use="required"/></xs:attribute>
</xs:element>
```

**Parent Eléments**

[ForEach Elément](#), [ModelFields Elément](#)

**Resources Elément**

Defines common resources such as client-side libraries and resource bundles, and server-side libraries.

**Représentation XML**

```
<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>
```

**Parent Eléments**

[Extension Elément](#)

**Enfant Eléments**

[Bundle Elément](#), [HelpInfo Elément](#), [JarFile Elément](#), [SharedLibrary Elément](#)

**Bundle Elément**

Table A-121  
Attributs pour Bundle

Attribut	Utiliser	Description	Valeurs valides
id	<b>obligatoire</b>		<i>string</i>
path	<b>obligatoire</b>		
type	<b>obligatoire</b>		<b>list</b> <b>properties</b>

**Représentation XML**

```
<xs:element name="Bundle" minOccurs="0">
  <xs:attribute name="id" type="xs:string" use="required"></xs:attribute>
  <xs:attribute name="type" use="required">
    <xs:simpleType>
      <xs:restriction base="xs:string">
        <xs:enumeration value="list"></xs:enumeration>
        <xs:enumeration value="properties"></xs:enumeration>
      </xs:restriction>
    </xs:simpleType>
  </xs:attribute>
  <xs:attribute name="path" type="EVALUATED-STRING" use="required"></xs:attribute>
```

</xs:element>

### Parent Eléments

[Resources Elément](#)

### JarFile Elément

Table A-122  
Attributs pour JarFile

Attribut	Utiliser	Description	Valeurs valides
id	obligatoire		string
path	obligatoire		

### Représentation XML

```
<xs:element name="JarFile" minOccurs="0">
  <xs:attribute name="id" type="xs:string" use="required"/></xs:attribute>
  <xs:attribute name="path" type="EVALUATED-STRING" use="required"/></xs:attribute>
</xs:element>
```

### Parent Eléments

[Resources Elément](#)

### SharedLibrary Elément

Table A-123  
Attributs pour SharedLibrary

Attribut	Utiliser	Description	Valeurs valides
id	obligatoire		string
path	obligatoire		

### Représentation XML

```
<xs:element name="SharedLibrary" minOccurs="0">
  <xs:attribute name="id" type="xs:string" use="required"/></xs:attribute>
  <xs:attribute name="path" type="EVALUATED-STRING" use="required"/></xs:attribute>
</xs:element>
```

### Parent Eléments

[Resources Elément](#)

### HelpInfo Elément

Table A-124  
Attributs pour HelpInfo

Attribut	Utiliser	Description	Valeurs valides
default	optional		string
helpset	optional		

Attribut	Utiliser	Description	Valeurs valides
id	optional		string
missing	optional		string
path	optional		
type	<b>obligatoire</b>		<b>native javahelp html</b>

### Représentation XML

```

<xs:element name="HelpInfo" minOccurs="0">
  <xs:attribute name="id" type="xs:string" use="optional"/></xs:attribute>
  <xs:attribute name="type" use="required">
    <xs:simpleType>
      <xs:restriction base="xs:string">
        <xs:enumeration value="native"/></xs:enumeration>
        <xs:enumeration value="javahelp"/></xs:enumeration>
        <xs:enumeration value="html"/></xs:enumeration>
      </xs:restriction>
    </xs:simpleType>
  </xs:attribute>
  <xs:attribute name="path" type="EVALUATED-STRING" use="optional"/></xs:attribute>
  <xs:attribute name="helpset" type="EVALUATED-STRING" use="optional"/></xs:attribute>
  <xs:attribute name="default" type="xs:string" use="optional"/></xs:attribute>
  <xs:attribute name="missing" type="xs:string" use="optional"/></xs:attribute>
</xs:element>

```

### Parent Éléments

[Resources Élément](#)

## Run Élément

### Représentation XML

```

<xs:element name="Run">
  <xs:sequence>
    <xs:group ref="CONDITION-EXPRESSION" minOccurs="0">
      <xs:choice>
        <xs:element ref="Condition"/></xs:element>
        <xs:element ref="And"/></xs:element>
        <xs:element ref="Or"/></xs:element>
        <xs:element ref="Not"/></xs:element>
      </xs:choice>
    </xs:group>
    <xs:element ref="Command" minOccurs="0" maxOccurs="unbounded"/></xs:element>
    <xs:element ref="Option" minOccurs="0" maxOccurs="unbounded"/></xs:element>
    <xs:element ref="StatusCodes" minOccurs="0"/></xs:element>
  </xs:sequence>
</xs:element>

```

**Parent Eléments**[Executable Elément](#)**Enfant Eléments**[And Elément](#), [Command Elément](#), [Condition Elément](#), [Not Elément](#), [Option Elément](#), [Or Elément](#), [StatusCodes Elément](#)**SelectorPanel Elément**

Table A-125

Attributs pour SelectorPanel

Attribut	Utiliser	Description	Valeurs valides
control	<b>obligatoire</b>		<i>string</i>

**Représentation XML**

```

<xs:element name="SelectorPanel">
  <xs:sequence>
    <xs:choice>
      <xs:element ref="Layout" minOccurs="0" maxOccurs="1"></xs:element>
      <xs:element ref="Enabled" minOccurs="0" maxOccurs="1"></xs:element>
      <xs:element ref="Visible" minOccurs="0" maxOccurs="1"></xs:element>
    </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:attribute>
</xs:element>

```

**Parent Eléments**[PropertiesPanel Elément](#), [PropertiesSubPanel Elément](#)**Enfant Eléments**[Enabled Elément](#), [Layout Elément](#), [Selector Elément](#), [Visible Elément](#)**Eléments connexes Eléments**[ActionButton Elément](#), [ComboBoxControl Elément](#), [ExtensionObjectPanel Elément](#), [ModelViewerPanel Elément](#), [StaticText Elément](#), [SystemControls Elément](#), [TabbedPanel Elément](#), [TextBrowserPanel Elément](#)**Selector Elément**

Table A-126

Attributs pour Selector

Attribut	Utiliser	Description	Valeurs valides
controlValue	<b>obligatoire</b>		<i>string</i>
panelId	<b>obligatoire</b>		<i>string</i>

**Représentation XML**

```
<xs:element name="Selector">
  <xs:attribute name="panelId" type="xs:string" use="required"/></xs:attribute>
  <xs:attribute name="controlValue" type="xs:string" use="required"/></xs:attribute>
</xs:element>
```

**Parent Eléments**

[SelectorPanel](#) Élément

**ServerDirectoryChooserControl Élément**

Table A-127

Attributs pour *ServerDirectoryChooserControl*

Attribut	Utiliser	Description	Valeurs valides
description	optional		<i>string</i>
descriptionKey	optional		<i>string</i>
label	optional		<i>string</i>
labelAbove	optional		<i>boolean</i>
labelKey	optional		<i>string</i>
labelWidth	optional		<i>positiveInteger</i>
mnemonic	optional		<i>string</i>
mnemonicKey	optional		<i>string</i>
mode	<b>obligatoire</b>		<b>open</b> <b>save</b> <b>import</b> <b>export</b>
property	<b>obligatoire</b>		<i>string</i>
showLabel	optional		<i>boolean</i>

**Représentation XML**

```
<xs:element name="ServerDirectoryChooserControl">
  <xs:sequence>
    <xs:choice>
      <xs:element ref="Layout" minOccurs="0" maxOccurs="1"/></xs:element>
      <xs:element ref="Enabled" minOccurs="0" maxOccurs="1"/></xs:element>
      <xs:element ref="Visible" minOccurs="0" maxOccurs="1"/></xs:element>
    </xs:choice>
  </xs:sequence>
  <xs:attribute name="property" type="xs:string" use="required"/></xs:attribute>
  <xs:attribute name="showLabel" type="xs:boolean" use="optional" default="true"/></xs:attribute>
  <xs:attribute name="label" type="xs:string" use="optional"/></xs:attribute>
  <xs:attribute name="labelKey" type="xs:string" use="optional"/></xs:attribute>
  <xs:attribute name="mnemonic" type="xs:string" use="optional"/></xs:attribute>
  <xs:attribute name="mnemonicKey" type="xs:string" use="optional"/></xs:attribute>
  <xs:attribute name="labelWidth" type="xs:positiveInteger" use="optional" default="1"/></xs:attribute>
  <xs:attribute name="labelAbove" type="xs:boolean" use="optional" default="false"/></xs:attribute>
  <xs:attribute name="description" type="xs:string" use="optional"/></xs:attribute>
  <xs:attribute name="descriptionKey" type="xs:string" use="optional"/></xs:attribute>
  <xs:attribute name="mode" type="FILE-CHOOSER-MODE" use="required">
    <xs:enumeration value="open"/></xs:enumeration>
```

```

    <xs:enumeration value="save"></xs:enumeration>
    <xs:enumeration value="import"></xs:enumeration>
    <xs:enumeration value="export"></xs:enumeration>
  </xs:attribute>
</xs:element>

```

### Parent Eléments

PropertiesPanel Elément, PropertiesSubPanel Elément

### Enfant Eléments

Enabled Elément, Layout Elément, Visible Elément

### Eléments connexes Eléments

CheckBoxControl Elément, CheckBoxGroupControl Elément, ClientDirectoryChooserControl Elément, ClientFileChooserControl Elément, DBConnectionChooserControl Elément, DBTableChooserControl Elément, MultiFieldChooserControl Elément, PasswordBoxControl Elément, PropertyControl Elément, RadioButtonGroupControl Elément, ServerFileChooserControl Elément, SingleFieldChooserControl Elément, SingleFieldValueChooserControl Elément, SpinnerControl Elément, TableControl Elément, TextAreaControl Elément, TextBoxControl Elément

## ServerFileChooserControl Elément

Table A-128

Attributs pour ServerFileChooserControl

Attribut	Utiliser	Description	Valeurs valides
description	optional		<i>string</i>
descriptionKey	optional		<i>string</i>
label	optional		<i>string</i>
labelAbove	optional		<i>boolean</i>
labelKey	optional		<i>string</i>
labelWidth	optional		<i>positiveInteger</i>
mnemonic	optional		<i>string</i>
mnemonicKey	optional		<i>string</i>
mode	<b>obligatoire</b>		<b>open</b> <b>save</b> <b>import</b> <b>export</b>
property	<b>obligatoire</b>		<i>string</i>
showLabel	optional		<i>boolean</i>

### Représentation XML

```

<xs:element name="ServerFileChooserControl">
  <xs:sequence>
    <xs:choice>
      <xs:element ref="Layout" minOccurs="0" maxOccurs="1"></xs:element>
      <xs:element ref="Enabled" minOccurs="0" maxOccurs="1"></xs:element>
    </xs:choice>
  </xs:sequence>
</xs:element>

```



```

    <xs:element ref="Visible" minOccurs="0" maxOccurs="1"></xs:element>
  </xs:choice>
</xs:sequence>
<xs:attribute name="property" type="xs:string" use="required"></xs:attribute>
<xs:attribute name="showLabel" type="xs:boolean" use="optional" default="true"></xs:attribute>
<xs:attribute name="label" type="xs:string" use="optional"></xs:attribute>
<xs:attribute name="labelKey" type="xs:string" use="optional"></xs:attribute>
<xs:attribute name="mnemonic" type="xs:string" use="optional"></xs:attribute>
<xs:attribute name="mnemonicKey" type="xs:string" use="optional"></xs:attribute>
<xs:attribute name="labelWidth" type="xs:positiveInteger" use="optional" default="1"></xs:attribute>
<xs:attribute name="labelAbove" type="xs:boolean" use="optional" default="false"></xs:attribute>
<xs:attribute name="description" type="xs:string" use="optional"></xs:attribute>
<xs:attribute name="descriptionKey" type="xs:string" use="optional"></xs:attribute>
<xs:attribute name="mode" type="FILE-CHOOSER-MODE" use="required">
  <xs:enumeration value="open"></xs:enumeration>
  <xs:enumeration value="save"></xs:enumeration>
  <xs:enumeration value="import"></xs:enumeration>
  <xs:enumeration value="export"></xs:enumeration>
</xs:attribute>
</xs:element>

```

### Parent Eléments

[PropertiesPanel Elément](#), [PropertiesSubPanel Elément](#)

### Enfant Eléments

[Enabled Elément](#), [Layout Elément](#), [Visible Elément](#)

### Eléments connexes Eléments

[CheckBoxControl Elément](#), [CheckBoxGroupControl Elément](#), [ClientDirectoryChooserControl Elément](#), [ClientFileChooserControl Elément](#), [DBConnectionChooserControl Elément](#), [DBTableChooserControl Elément](#), [MultiFieldChooserControl Elément](#), [PasswordBoxControl Elément](#), [PropertyControl Elément](#), [RadioButtonGroupControl Elément](#), [ServerDirectoryChooserControl Elément](#), [SingleFieldChooserControl Elément](#), [SingleFieldValueChooserControl Elément](#), [SpinnerControl Elément](#), [TableControl Elément](#), [TextAreaControl Elément](#), [TextBoxControl Elément](#)

## SetContainer Elément

Table A-129

Attributs pour SetContainer

Attribut	Utiliser	Description	Valeurs valides
source	<b>obligatoire</b>		<i>string</i>
target	<b>obligatoire</b>		<i>string</i>

### Représentation XML

```

<xs:element name="SetContainer">
  <xs:attribute name="source" type="xs:string" use="required"></xs:attribute>
  <xs:attribute name="target" type="xs:string" use="required"></xs:attribute>
</xs:element>

```

**Parent Eléments**

[CreateDocumentOutput Elément](#), [CreateInteractiveDocumentBuilder Elément](#),  
[CreateInteractiveModelBuilder Elément](#), [CreateModelApplier Elément](#), [CreateModelOutput Elément](#)

**Eléments connexes Eléments**

[SetProperty Elément](#)

**SetProperty Elément**

Table A-130  
 Attributs pour SetProperty

Attribut	Utiliser	Description	Valeurs valides
source	<b>obligatoire</b>		<i>string</i>
target	<b>obligatoire</b>		<i>string</i>

**Représentation XML**

```
<xs:element name="SetProperty">
  <xs:attribute name="source" type="xs:string" use="required"></xs:attribute>
  <xs:attribute name="target" type="xs:string" use="required"></xs:attribute>
</xs:element>
```

**Parent Eléments**

[CreateDocumentOutput Elément](#), [CreateInteractiveDocumentBuilder Elément](#),  
[CreateInteractiveModelBuilder Elément](#), [CreateModelApplier Elément](#), [CreateModelOutput Elément](#)

**Eléments connexes Eléments**

[SetContainer Elément](#)

**SingleFieldChooserControl Elément**

Table A-131  
 Attributs pour SingleFieldChooserControl

Attribut	Utiliser	Description	Valeurs valides
description	optional		<i>string</i>
descriptionKey	optional		<i>string</i>
label	optional		<i>string</i>
labelAbove	optional		<i>boolean</i>
labelKey	optional		<i>string</i>
labelWidth	optional		<i>positiveInteger</i>
mnemonic	optional		<i>string</i>
mnemonicKey	optional		<i>string</i>
onlyDatetime	optional		<i>boolean</i>
onlyDiscrete	optional		<i>boolean</i>

Attribut	Utiliser	Description	Valeurs valides
onlyNumeric	optional		boolean
onlyRanges	optional		boolean
onlySymbolic	optional		boolean
property	<b>obligatoire</b>		string
showLabel	optional		boolean
storage	optional		string
types	optional		string

### Représentation XML

```

<xs:element name="SingleFieldChooserControl">
  <xs:sequence>
    <xs:choice>
      <xs:element ref="Layout" minOccurs="0" maxOccurs="1"></xs:element>
      <xs:element ref="Enabled" minOccurs="0" maxOccurs="1"></xs:element>
      <xs:element ref="Visible" minOccurs="0" maxOccurs="1"></xs:element>
    </xs:choice>
  </xs:sequence>
  <xs:attribute name="property" type="xs:string" use="required"></xs:attribute>
  <xs:attribute name="showLabel" type="xs:boolean" use="optional" default="true"></xs:attribute>
  <xs:attribute name="label" type="xs:string" use="optional"></xs:attribute>
  <xs:attribute name="labelKey" type="xs:string" use="optional"></xs:attribute>
  <xs:attribute name="mnemonic" type="xs:string" use="optional"></xs:attribute>
  <xs:attribute name="mnemonicKey" type="xs:string" use="optional"></xs:attribute>
  <xs:attribute name="labelWidth" type="xs:positiveInteger" use="optional" default="1"></xs:attribute>
  <xs:attribute name="labelAbove" type="xs:boolean" use="optional" default="false"></xs:attribute>
  <xs:attribute name="description" type="xs:string" use="optional"></xs:attribute>
  <xs:attribute name="descriptionKey" type="xs:string" use="optional"></xs:attribute>
  <xs:attribute name="storage" type="xs:string" use="optional"></xs:attribute>
  <xs:attribute name="onlyNumeric" type="xs:boolean" use="optional"></xs:attribute>
  <xs:attribute name="onlySymbolic" type="xs:boolean" use="optional"></xs:attribute>
  <xs:attribute name="onlyDatetime" type="xs:boolean" use="optional"></xs:attribute>
  <xs:attribute name="types" type="xs:string" use="optional"></xs:attribute>
  <xs:attribute name="onlyRanges" type="xs:boolean" use="optional"></xs:attribute>
  <xs:attribute name="onlyDiscrete" type="xs:boolean" use="optional"></xs:attribute>
</xs:element>

```

### Parent Éléments

[PropertiesPanel Élément](#), [PropertiesSubPanel Élément](#)

### Enfant Éléments

[Enabled Élément](#), [Layout Élément](#), [Visible Élément](#)

### Éléments connexes Éléments

[CheckBoxControl Élément](#), [CheckBoxGroupControl Élément](#), [ClientDirectoryChooserControl Élément](#), [ClientFileChooserControl Élément](#), [DBConnectionChooserControl Élément](#), [DBTableChooserControl Élément](#), [MultiFieldChooserControl Élément](#), [PasswordBoxControl Élément](#), [PropertyControl Élément](#), [RadioButtonGroupControl Élément](#), [ServerDirectoryChooserControl Élément](#), [ServerFileChooserControl Élément](#),

[SingleFieldValueChooserControl Élément](#), [SpinnerControl Élément](#), [TableControl Élément](#), [TextAreaControl Élément](#), [TextBoxControl Élément](#)

## SingleFieldValueChooserControl Élément

Table A-132  
Attributs pour SingleFieldValueChooserControl

Attribut	Utiliser	Description	Valeurs valides
description	optional		string
descriptionKey	optional		string
fieldControl	optional		string
fieldDirection	optional		in out both none partition
label	optional		string
labelAbove	optional		boolean
labelKey	optional		string
labelWidth	optional		positiveInteger
mnemonic	optional		string
mnemonicKey	optional		string
property	<b>obligatoire</b>		string
showLabel	optional		boolean

### Représentation XML

```
<xs:element name="SingleFieldValueChooserControl">
  <xs:sequence>
    <xs:choice>
      <xs:element ref="Layout" minOccurs="0" maxOccurs="1"/></xs:element>
      <xs:element ref="Enabled" minOccurs="0" maxOccurs="1"/></xs:element>
      <xs:element ref="Visible" minOccurs="0" maxOccurs="1"/></xs:element>
    </xs:choice>
  </xs:sequence>
  <xs:attribute name="property" type="xs:string" use="required"/></xs:attribute>
  <xs:attribute name="showLabel" type="xs:boolean" use="optional" default="true"/></xs:attribute>
  <xs:attribute name="label" type="xs:string" use="optional"/></xs:attribute>
  <xs:attribute name="labelKey" type="xs:string" use="optional"/></xs:attribute>
  <xs:attribute name="mnemonic" type="xs:string" use="optional"/></xs:attribute>
  <xs:attribute name="mnemonicKey" type="xs:string" use="optional"/></xs:attribute>
  <xs:attribute name="labelWidth" type="xs:positiveInteger" use="optional" default="1"/></xs:attribute>
  <xs:attribute name="labelAbove" type="xs:boolean" use="optional" default="false"/></xs:attribute>
  <xs:attribute name="description" type="xs:string" use="optional"/></xs:attribute>
  <xs:attribute name="descriptionKey" type="xs:string" use="optional"/></xs:attribute>
  <xs:attribute name="fieldControl" type="xs:string" use="optional"/></xs:attribute>
  <xs:attribute name="fieldDirection" type="FIELD-DIRECTION" use="optional">
    <xs:enumeration value="in"/></xs:enumeration>
    <xs:enumeration value="out"/></xs:enumeration>
    <xs:enumeration value="both"/></xs:enumeration>
    <xs:enumeration value="none"/></xs:enumeration>
    <xs:enumeration value="partition"/></xs:enumeration>
  </xs:attribute>
</xs:element>
```

```
</xs:attribute>
</xs:element>
```

### Parent Eléments

PropertiesPanel Elément, PropertiesSubPanel Elément

### Enfant Eléments

Enabled Elément, Layout Elément, Visible Elément

### Eléments connexes Eléments

CheckBoxControl Elément, CheckBoxGroupControl Elément, ClientDirectoryChooserControl Elément, ClientFileChooserControl Elément, DBConnectionChooserControl Elément, DBTableChooserControl Elément, MultiFieldChooserControl Elément, PasswordBoxControl Elément, PropertyControl Elément, RadioButtonGroupControl Elément, ServerDirectoryChooserControl Elément, ServerFileChooserControl Elément, SingleFieldChooserControl Elément, SpinnerControl Elément, TableControl Elément, TextAreaControl Elément, TextBoxControl Elément

## SingleItemChooserControl Elément

Table A-133  
Attributs pour SingleItemChooserControl

Attribut	Utiliser	Description	Valeurs valides
catalog	<b>obligatoire</b>		<i>string</i>
description	optional		<i>string</i>
descriptionKey	optional		<i>string</i>
label	optional		<i>string</i>
labelAbove	optional		<i>boolean</i>
labelKey	optional		<i>string</i>
labelWidth	optional		<i>positiveInteger</i>
mnemonic	optional		<i>string</i>
mnemonicKey	optional		<i>string</i>
property	<b>obligatoire</b>		<i>string</i>
showLabel	optional		<i>boolean</i>

### Représentation XML

```
<xs:element name="SingleItemChooserControl">
  <xs:sequence>
    <xs:choice>
      <xs:element ref="Layout" minOccurs="0" maxOccurs="1"></xs:element>
      <xs:element ref="Enabled" minOccurs="0" maxOccurs="1"></xs:element>
      <xs:element ref="Visible" minOccurs="0" maxOccurs="1"></xs:element>
    </xs:choice>
  </xs:sequence>
  <xs:attribute name="property" type="xs:string" use="required"></xs:attribute>
  <xs:attribute name="showLabel" type="xs:boolean" use="optional" default="true"></xs:attribute>
```

```

<xs:attribute name="label" type="xs:string" use="optional"/></xs:attribute>
<xs:attribute name="labelKey" type="xs:string" use="optional"/></xs:attribute>
<xs:attribute name="mnemonic" type="xs:string" use="optional"/></xs:attribute>
<xs:attribute name="mnemonicKey" type="xs:string" use="optional"/></xs:attribute>
<xs:attribute name="labelWidth" type="xs:positiveInteger" use="optional" default="1"/></xs:attribute>
<xs:attribute name="labelAbove" type="xs:boolean" use="optional" default="false"/></xs:attribute>
<xs:attribute name="description" type="xs:string" use="optional"/></xs:attribute>
<xs:attribute name="descriptionKey" type="xs:string" use="optional"/></xs:attribute>
<xs:attribute name="catalog" type="xs:string" use="required"/></xs:attribute>
</xs:element>

```

### Parent Éléments

[PropertiesPanel Élément](#), [PropertiesSubPanel Élément](#)

### Enfant Éléments

[Enabled Élément](#), [Layout Élément](#), [Visible Élément](#)

### Éléments connexes Éléments

[MultiItemChooserControl Élément](#)

## SpinnerControl Élément

Table A-134

Attributs pour SpinnerControl

Attribut	Utiliser	Description	Valeurs valides
columns	optional		<i>positiveInteger</i>
description	optional		<i>string</i>
descriptionKey	optional		<i>string</i>
label	optional		<i>string</i>
labelAbove	optional		<i>boolean</i>
labelKey	optional		<i>string</i>
labelWidth	optional		<i>positiveInteger</i>
maxDecimalDigits	optional		<i>positiveInteger</i>
minDecimalDigits	optional		<i>positiveInteger</i>
mnemonic	optional		<i>string</i>
mnemonicKey	optional		<i>string</i>
property	<b>obligatoire</b>		<i>string</i>
showLabel	optional		<i>boolean</i>
stepSize	optional		<i>decimal</i>

### Représentation XML

```

<xs:element name="SpinnerControl">
  <xs:sequence>
    <xs:choice>
      <xs:element ref="Layout" minOccurs="0" maxOccurs="1"/></xs:element>
      <xs:element ref="Enabled" minOccurs="0" maxOccurs="1"/></xs:element>
      <xs:element ref="Visible" minOccurs="0" maxOccurs="1"/></xs:element>
    </xs:choice>
  </xs:sequence>
</xs:element>

```

```

    </xs:choice>
  </xs:sequence>
  <xs:attribute name="property" type="xs:string" use="required"></xs:attribute>
  <xs:attribute name="showLabel" type="xs:boolean" use="optional" default="true"></xs:attribute>
  <xs:attribute name="label" type="xs:string" use="optional"></xs:attribute>
  <xs:attribute name="labelKey" type="xs:string" use="optional"></xs:attribute>
  <xs:attribute name="mnemonic" type="xs:string" use="optional"></xs:attribute>
  <xs:attribute name="mnemonicKey" type="xs:string" use="optional"></xs:attribute>
  <xs:attribute name="labelWidth" type="xs:positiveInteger" use="optional" default="1"></xs:attribute>
  <xs:attribute name="labelAbove" type="xs:boolean" use="optional" default="false"></xs:attribute>
  <xs:attribute name="description" type="xs:string" use="optional"></xs:attribute>
  <xs:attribute name="descriptionKey" type="xs:string" use="optional"></xs:attribute>
  <xs:attribute name="columns" type="xs:positiveInteger" use="optional" default="5"></xs:attribute>
  <xs:attribute name="stepSize" type="xs:decimal" use="optional" default="1.0"></xs:attribute>
  <xs:attribute name="minDecimalDigits" type="xs:positiveInteger" use="optional" default="1"></xs:attribute>
  <xs:attribute name="maxDecimalDigits" type="xs:positiveInteger" use="optional"></xs:attribute>
</xs:element>

```

### **Parent Eléments**

[PropertiesPanel Elément](#), [PropertiesSubPanel Elément](#)

### **Enfant Eléments**

[Enabled Elément](#), [Layout Elément](#), [Visible Elément](#)

### **Eléments connexes Eléments**

[CheckBoxControl Elément](#), [CheckBoxGroupControl Elément](#), [ClientDirectoryChooserControl Elément](#), [ClientFileChooserControl Elément](#), [DBConnectionChooserControl Elément](#), [DBTableChooserControl Elément](#), [MultiFieldChooserControl Elément](#), [PasswordBoxControl Elément](#), [PropertyControl Elément](#), [RadioButtonGroupControl Elément](#), [ServerDirectoryChooserControl Elément](#), [ServerFileChooserControl Elément](#), [SingleFieldChooserControl Elément](#), [SingleFieldValueChooserControl Elément](#), [TableControl Elément](#), [TextAreaControl Elément](#), [TextBoxControl Elément](#)

## **SPSSDataFormat Elément**

### **Représentation XML**

```
<xs:element name="SPSSDataFormat"></xs:element>
```

### **Parent Eléments**

[DataFormat Elément](#)

## StaticText Élément

Table A-135  
Attributs pour StaticText

Attribut	Utiliser	Description	Valeurs valides
text	optional		string
textKey	optional		string

### Représentation XML

```
<xs:element name="StaticText">
  <xs:sequence>
    <xs:choice>
      <xs:element ref="Layout" minOccurs="0" maxOccurs="1"/></xs:element>
      <xs:element ref="Enabled" minOccurs="0" maxOccurs="1"/></xs:element>
      <xs:element ref="Visible" minOccurs="0" maxOccurs="1"/></xs:element>
    </xs:choice>
  </xs:sequence>
  <xs:attribute name="text" type="xs:string" use="optional"/></xs:attribute>
  <xs:attribute name="textKey" type="xs:string" use="optional"/></xs:attribute>
</xs:element>
```

### Parent Elements

[PropertiesPanel Élément](#), [PropertiesSubPanel Élément](#)

### Enfant Elements

[Enabled Élément](#), [Layout Élément](#), [Visible Élément](#)

### Éléments connexes Elements

[ActionButton Élément](#), [ComboBoxControl Élément](#), [ExtensionObjectPanel Élément](#),  
[ModelViewerPanel Élément](#), [SelectorPanel Élément](#), [SystemControls Élément](#), [TabbedPanel  
Élément](#), [TextBrowserPanel Élément](#)

## StatusCode Élément

Table A-136  
Attributs pour StatusCode

Attribut	Utiliser	Description	Valeurs valides
code	<b>obligatoire</b>		integer
message	optional		string
messageKey	optional		string
status	optional		<b>success</b> <b>warning</b> <b>error</b>

### Représentation XML

```
<xs:element name="StatusCode">
  <xs:attribute name="code" type="xs:integer" use="required"/></xs:attribute>
```



```

<xs:attribute name="status" use="optional" default="success">
  <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:enumeration value="success"></xs:enumeration>
      <xs:enumeration value="warning"></xs:enumeration>
      <xs:enumeration value="error"></xs:enumeration>
    </xs:restriction>
  </xs:simpleType>
</xs:attribute>
<xs:attribute name="message" type="xs:string" use="optional"></xs:attribute>
<xs:attribute name="messageKey" type="xs:string" use="optional"></xs:attribute>
</xs:element>

```

**Parent Eléments**[StatusCodes Elément](#)**StatusCodes Elément**

Table A-137  
Attributs pour StatusCodes

Attribut	Utiliser	Description	Valeurs valides
defaultMessage	optional		<i>string</i>
defaultMessageKey	optional		<i>string</i>

**Représentation XML**

```

<xs:element name="StatusCodes">
  <xs:sequence>
    <xs:element ref="StatusCode" maxOccurs="unbounded"></xs:element>
  </xs:sequence>
  <xs:attribute name="defaultMessage" type="xs:string" use="optional"></xs:attribute>
  <xs:attribute name="defaultMessageKey" type="xs:string" use="optional"></xs:attribute>
</xs:element>

```

**Parent Eléments**[Module Elément](#), [Run Elément](#)**Enfant Eléments**[StatusCode Elément](#)**StatusDetail Elément**

Supplementary information about a progress or other conditions.

Table A-138  
Attributs pour StatusDetail

Attribut	Utiliser	Description	Valeurs valides
destination	facultatif		<b>client</b> <b>tracefile</b> <b>console</b>

**Représentation XML**

```

<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:element>
      </xs:sequence>
    </xs:element>
  </xs:sequence>
  <xs:attribute name="destination" type="STATUS-DESTINATION" default="client">
    <xs:enumeration value="client"/></xs:enumeration>
    <xs:enumeration value="tracefile"/></xs:enumeration>
    <xs:enumeration value="console"/></xs:enumeration>
  </xs:attribute>
</xs:element>

```

**Enfant Eléments**[Diagnostic Elément](#)**Diagnostic Elément**

Table A-139  
Attributs pour Diagnostic

Attribut	Utiliser	Description	Valeurs valides
code	<b>obligatoire</b>		<i>integer</i>
severity	facultatif		<b>unknown information warning error fatal</b>
source	facultatif		<i>string</i>
subCode	facultatif		<i>integer</i>

**Représentation XML**

```

<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:element>
  </xs:sequence>
  <xs:attribute name="code" type="xs:integer" use="required"/></xs:attribute>
  <xs:attribute name="subCode" type="xs:integer" default="0"/></xs:attribute>
  <xs:attribute name="severity" type="DIAGNOSTIC-SEVERITY" default="error">
    <xs:enumeration value="unknown"/></xs:enumeration>
    <xs:enumeration value="information"/></xs:enumeration>
    <xs:enumeration value="warning"/></xs:enumeration>
    <xs:enumeration value="error"/></xs:enumeration>
    <xs:enumeration value="fatal"/></xs:enumeration>
  </xs:attribute>
  <xs:attribute name="source" type="xs:string"/></xs:attribute>

```

</xs:element>

### **Parent Eléments**

[StatusDetail Élément](#)

### **Enfant Eléments**

[Message Élément](#), [Parameter Élément](#)

## **Message Élément**

Table A-140

*Attributs pour Message*

Attribut	Utiliser	Description	Valeurs valides
lang	facultatif		<i>NMTOKEN</i>

### **Représentation XML**

```
<xs:element name="Message" type="DIAGNOSTIC-MESSAGE" minOccurs="0">
  <xs:attribute name="lang" type="xs:NMTOKEN"></xs:attribute>
</xs:element>
```

### **Parent Eléments**

[Diagnostic Élément](#)

## **Parameter Élément**

### **Représentation XML**

```
<xs:element name="Parameter" type="xs:string" minOccurs="0" maxOccurs="unbounded"></xs:element>
```

### **Parent Eléments**

[Diagnostic Élément](#)

## **Structure Élément**

### **Représentation XML**

```
<xs:element name="Structure">
  <xs:sequence>
    <xs:element ref="Attribute" minOccurs="0" maxOccurs="unbounded"></xs:element>
  </xs:sequence>
</xs:element>
```

### **Parent Eléments**

[PropertyType Élément](#)

**Enfant Eléments**[Attribute Elément](#)**StructuredValue Elément**

A sequence of named values (“attributes”).

**Représentation XML**

```

<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>
          <xs:element ref="StructuredValue"></xs:element>
          <xs:element ref="ListValue"></xs:element>
          <xs:element ref="Value"></xs:element>
          <xs:element ref="DatabaseConnectionValue"></xs:element>
        </xs:choice>
      </xs:group>
    </xs:sequence>
    <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>
            <xs:element ref="StructuredValue"></xs:element>
            <xs:element ref="ListValue"></xs:element>
            <xs:element ref="Value"></xs:element>
            <xs:element ref="DatabaseConnectionValue"></xs:element>
          </xs:choice>
        </xs:group>
      </xs:element>
    </xs:sequence>
  </xs:element>

```

**Parent Eléments**

[ListValue Elément](#), [Attribute Elément](#), [ListValue Elément](#), [Parameter Elément](#), [Attribute Elément](#), [ListValue Elément](#)

**Enfant Eléments**[Attribute Elément](#)

**Attribute Élément**

Table A-141  
Attributs pour Attribute

Attribut	Utiliser	Description	Valeurs valides
name	<b>obligatoire</b>		<i>string</i>
value	facultatif		<i>string</i>

**Représentation XML**

```

<xs:element name="Attribute" type="ATTRIBUTE" maxOccurs="unbounded">
  <xs:group ref="PARAMETER-CONTENT" minOccurs="0">
    <xs:choice>
      <xs:element ref="MapValue"></xs:element>
      <xs:element ref="StructuredValue"></xs:element>
      <xs:element ref="ListValue"></xs:element>
      <xs:element ref="Value"></xs:element>
      <xs:element ref="DatabaseConnectionValue"></xs:element>
    </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>
          <xs:element ref="StructuredValue"></xs:element>
          <xs:element ref="ListValue"></xs:element>
          <xs:element ref="Value"></xs:element>
          <xs:element ref="DatabaseConnectionValue"></xs:element>
        </xs:choice>
      </xs:group>
    </xs:element>
  </xs:sequence>
  <xs:attribute name="name" type="xs:string" use="required"></xs:attribute>
  <xs:attribute name="value" type="xs:string"></xs:attribute>
</xs:element>

```

**Parent Elements**

[StructuredValue Élément](#)

**Enfant Elements**

[DatabaseConnectionValue Élément](#), [ListValue Élément](#), [ListValue Élément](#), [MapValue Élément](#), [StructuredValue Élément](#), [Value Élément](#)

**ListValue Élément**

A sequence of values. All values must have the same content type but this is not checked.

**Représentation XML**

```

<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>
      <xs:element ref="StructuredValue"></xs:element>
      <xs:element ref="ListValue"></xs:element>
      <xs:element ref="Value"></xs:element>
      <xs:element ref="DatabaseConnectionValue"></xs:element>
    </xs:choice>
  </xs:group>
</xs:element>

```

### Parent Eléments

[Attribute Elément](#)

### Enfant Eléments

[DatabaseConnectionValue Elément](#), [ListValue Elément](#), [MapValue Elément](#), [StructuredValue Elément](#), [Value Elément](#)

## SystemControls Elément

Table A-142  
Attributs pour SystemControls

Attribut	Utiliser	Description	Valeurs valides
controlsId	<b>obligatoire</b>		<i>string</i>

### Représentation XML

```

<xs:element name="SystemControls">
  <xs:sequence>
    <xs:choice>
      <xs:element ref="Layout" minOccurs="0" maxOccurs="1"></xs:element>
      <xs:element ref="Enabled" minOccurs="0" maxOccurs="1"></xs:element>
      <xs:element ref="Visible" minOccurs="0" maxOccurs="1"></xs:element>
    </xs:choice>
  </xs:sequence>
  <xs:attribute name="controlsId" type="xs:string" use="required"></xs:attribute>
</xs:element>

```

### Parent Eléments

[PropertiesPanel Elément](#), [PropertiesSubPanel Elément](#)

### Enfant Eléments

[Enabled Elément](#), [Layout Elément](#), [Visible Elément](#)

### Eléments connexes Eléments

[ActionButton Elément](#), [ComboBoxControl Elément](#), [ExtensionObjectPanel Elément](#), [ModelViewerPanel Elément](#), [SelectorPanel Elément](#), [StaticText Elément](#), [TabbedPanel Elément](#), [TextBrowserPanel Elément](#)

## Tab Élément

Table A-143  
Attributs pour Tab

Attribut	Utiliser	Description	Valeurs valides
helpLink	optional		string
id	optional		string
label	<b>obligatoire</b>		string
labelKey	optional		string
mnemonic	optional		string
mnemonicKey	optional		string

### Représentation XML

```
<xs:element name="Tab">
  <xs:sequence minOccurs="0" maxOccurs="unbounded">
    <xs:choice>
      <xs:element ref="PropertiesPanel"></xs:element>
      <xs:element ref="ExtensionObjectPanel"></xs:element>
      <xs:element ref="TextBrowserPanel"></xs:element>
      <xs:element ref="ModelViewerPanel"></xs:element>
      <xs:element ref="TabbedPanel"></xs:element>
    </xs:choice>
  </xs:sequence>
  <xs:attribute name="id" type="xs:string" use="optional"></xs:attribute>
  <xs:attribute name="label" type="xs:string" use="required"></xs:attribute>
  <xs:attribute name="labelKey" type="xs:string" use="optional"></xs:attribute>
  <xs:attribute name="mnemonic" type="xs:string" use="optional"></xs:attribute>
  <xs:attribute name="mnemonicKey" type="xs:string" use="optional"></xs:attribute>
  <xs:attribute name="helpLink" type="xs:string" use="optional"></xs:attribute>
</xs:element>
```

### Parent Elements

[Tabs Élément](#)

### Enfant Elements

[ExtensionObjectPanel Élément](#), [ModelViewerPanel Élément](#), [PropertiesPanel Élément](#), [TabbedPanel Élément](#), [TextBrowserPanel Élément](#)

## TabbedPanel Élément

Table A-144  
Attributs pour TabbedPanel

Attribut	Utiliser	Description	Valeurs valides
style	optional		<b>standard sidebar</b>

### Représentation XML

```
<xs:element name="TabbedPanel">
```

```

<xs:sequence>
  <xs:choice>
    <xs:element ref="Layout" minOccurs="0" maxOccurs="1"></xs:element>
    <xs:element ref="Enabled" minOccurs="0" maxOccurs="1"></xs:element>
    <xs:element ref="Visible" minOccurs="0" maxOccurs="1"></xs:element>
  </xs:choice>
</xs:sequence>
<xs:sequence maxOccurs="unbounded">
  <xs:element ref="Tabs"></xs:element>
</xs:sequence>
<xs:attribute name="style" use="optional">
  <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:enumeration value="standard"></xs:enumeration>
      <xs:enumeration value="sidebar"></xs:enumeration>
    </xs:restriction>
  </xs:simpleType>
</xs:attribute>
</xs:element>

```

**Parent Eléments**[Tab Elément](#)**Enfant Eléments**[Enabled Elément](#), [Layout Elément](#), [Tabs Elément](#), [Visible Elément](#)**Eléments connexes Eléments**[ActionButton Elément](#), [ComboBoxControl Elément](#), [ExtensionObjectPanel Elément](#), [ModelViewerPanel Elément](#), [SelectorPanel Elément](#), [StaticText Elément](#), [SystemControls Elément](#), [TextBrowserPanel Elément](#)**TableControl Elément**Table A-145  
Attributs pour TableControl

Attribut	Utiliser	Description	Valeurs valides
columns	optional		<i>positiveInteger</i>
columnWidths	optional		<i>string</i>
description	optional		<i>string</i>
descriptionKey	optional		<i>string</i>
label	optional		<i>string</i>
labelAbove	optional		<i>boolean</i>
labelKey	optional		<i>string</i>
labelWidth	optional		<i>positiveInteger</i>
mnemonic	optional		<i>string</i>
mnemonicKey	optional		<i>string</i>
property	<b>obligatoire</b>		<i>string</i>
rows	optional		<i>positiveInteger</i>
showLabel	optional		<i>boolean</i>



**Représentation XML**

```

<xs:element name="TableControl">
  <xs:sequence>
    <xs:choice>
      <xs:element ref="Layout" minOccurs="0" maxOccurs="1"></xs:element>
      <xs:element ref="Enabled" minOccurs="0" maxOccurs="1"></xs:element>
      <xs:element ref="Visible" minOccurs="0" maxOccurs="1"></xs:element>
    </xs:choice>
  </xs:sequence>
  <xs:attribute name="property" type="xs:string" use="required"></xs:attribute>
  <xs:attribute name="showLabel" type="xs:boolean" use="optional" default="true"></xs:attribute>
  <xs:attribute name="label" type="xs:string" use="optional"></xs:attribute>
  <xs:attribute name="labelKey" type="xs:string" use="optional"></xs:attribute>
  <xs:attribute name="mnemonic" type="xs:string" use="optional"></xs:attribute>
  <xs:attribute name="mnemonicKey" type="xs:string" use="optional"></xs:attribute>
  <xs:attribute name="labelWidth" type="xs:positiveInteger" use="optional" default="1"></xs:attribute>
  <xs:attribute name="labelAbove" type="xs:boolean" use="optional" default="false"></xs:attribute>
  <xs:attribute name="description" type="xs:string" use="optional"></xs:attribute>
  <xs:attribute name="descriptionKey" type="xs:string" use="optional"></xs:attribute>
  <xs:attribute name="rows" type="xs:positiveInteger" use="optional" default="8"></xs:attribute>
  <xs:attribute name="columns" type="xs:positiveInteger" use="optional" default="20"></xs:attribute>
  <xs:attribute name="columnWidths" type="xs:string" use="optional"></xs:attribute>
</xs:element>

```

**Parent Eléments**

[PropertiesPanel Elément](#), [PropertiesSubPanel Elément](#)

**Enfant Eléments**

[Enabled Elément](#), [Layout Elément](#), [Visible Elément](#)

**Eléments connexes Eléments**

[CheckBoxControl Elément](#), [CheckBoxGroupControl Elément](#), [ClientDirectoryChooserControl Elément](#), [ClientFileChooserControl Elément](#), [DBConnectionChooserControl Elément](#), [DBTableChooserControl Elément](#), [MultiFieldChooserControl Elément](#), [PasswordBoxControl Elément](#), [PropertyControl Elément](#), [RadioButtonGroupControl Elément](#), [ServerDirectoryChooserControl Elément](#), [ServerFileChooserControl Elément](#), [SingleFieldChooserControl Elément](#), [SingleFieldValueChooserControl Elément](#), [SpinnerControl Elément](#), [TextAreaControl Elément](#), [TextBoxControl Elément](#)

**Tabs Elément**

Table A-146  
Attributs pour Tabs

Attribut	Utiliser	Description	Valeurs valides
defaultTab	optional		<i>nonNegativeInteger</i>

**Représentation XML**

```

<xs:element name="Tabs">

```

```

<xs:sequence>
  <xs:element ref="Tab" minOccurs="0" maxOccurs="unbounded"/></xs:element>
</xs:sequence>
<xs:attribute name="defaultTab" type="xs:nonNegativeInteger" use="optional" default="0"/></xs:attribute>
</xs:element>

```

**Parent Eléments**

[TabbedPanel Elément](#), [UserInterface Elément](#)

**Enfant Eléments**

[Tab Elément](#)

**TextAreaControl Elément**

Table A-147  
Attributs pour *TextAreaControl*

Attribut	Utiliser	Description	Valeurs valides
columns	optional		<i>positiveInteger</i>
description	optional		<i>string</i>
descriptionKey	optional		<i>string</i>
label	optional		<i>string</i>
labelAbove	optional		<i>boolean</i>
labelKey	optional		<i>string</i>
labelWidth	optional		<i>positiveInteger</i>
mnemonic	optional		<i>string</i>
mnemonicKey	optional		<i>string</i>
property	<b>obligatoire</b>		<i>string</i>
rows	optional		<i>positiveInteger</i>
showLabel	optional		<i>boolean</i>
wrapLines	optional		<i>boolean</i>

**Représentation XML**

```

<xs:element name="TextAreaControl">
  <xs:sequence>
    <xs:choice>
      <xs:element ref="Layout" minOccurs="0" maxOccurs="1"/></xs:element>
      <xs:element ref="Enabled" minOccurs="0" maxOccurs="1"/></xs:element>
      <xs:element ref="Visible" minOccurs="0" maxOccurs="1"/></xs:element>
    </xs:choice>
  </xs:sequence>
  <xs:attribute name="property" type="xs:string" use="required"/></xs:attribute>
  <xs:attribute name="showLabel" type="xs:boolean" use="optional" default="true"/></xs:attribute>
  <xs:attribute name="label" type="xs:string" use="optional"/></xs:attribute>
  <xs:attribute name="labelKey" type="xs:string" use="optional"/></xs:attribute>
  <xs:attribute name="mnemonic" type="xs:string" use="optional"/></xs:attribute>
  <xs:attribute name="mnemonicKey" type="xs:string" use="optional"/></xs:attribute>
  <xs:attribute name="labelWidth" type="xs:positiveInteger" use="optional" default="1"/></xs:attribute>
  <xs:attribute name="labelAbove" type="xs:boolean" use="optional" default="false"/></xs:attribute>
  <xs:attribute name="description" type="xs:string" use="optional"/></xs:attribute>

```

```

<xs:attribute name="descriptionKey" type="xs:string" use="optional"/></xs:attribute>
<xs:attribute name="rows" type="xs:positiveInteger" use="optional" default="8"/></xs:attribute>
<xs:attribute name="columns" type="xs:positiveInteger" use="optional" default="20"/></xs:attribute>
<xs:attribute name="wrapLines" type="xs:boolean" use="optional" default="true"/></xs:attribute>
</xs:element>

```

### Parent Eléments

[PropertiesPanel Elément](#), [PropertiesSubPanel Elément](#)

### Enfant Eléments

[Enabled Elément](#), [Layout Elément](#), [Visible Elément](#)

### Eléments connexes Eléments

[CheckBoxControl Elément](#), [CheckBoxGroupControl Elément](#), [ClientDirectoryChooserControl Elément](#), [ClientFileChooserControl Elément](#), [DBConnectionChooserControl Elément](#), [DBTableChooserControl Elément](#), [MultiFieldChooserControl Elément](#), [PasswordBoxControl Elément](#), [PropertyControl Elément](#), [RadioButtonGroupControl Elément](#), [ServerDirectoryChooserControl Elément](#), [ServerFileChooserControl Elément](#), [SingleFieldChooserControl Elément](#), [SingleFieldValueChooserControl Elément](#), [SpinnerControl Elément](#), [TableControl Elément](#), [TextBoxControl Elément](#)

## TextBoxControl Elément

Table A-148

Attributs pour TextBoxControl

Attribut	Utiliser	Description	Valeurs valides
columns	optional		<i>positiveInteger</i>
description	optional		<i>string</i>
descriptionKey	optional		<i>string</i>
label	optional		<i>string</i>
labelAbove	optional		<i>boolean</i>
labelKey	optional		<i>string</i>
labelWidth	optional		<i>positiveInteger</i>
mnemonic	optional		<i>string</i>
mnemonicKey	optional		<i>string</i>
property	<b>obligatoire</b>		<i>string</i>
showLabel	optional		<i>boolean</i>

### Représentation XML

```

<xs:element name="TextBoxControl">
  <xs:sequence>
    <xs:choice>
      <xs:element ref="Layout" minOccurs="0" maxOccurs="1"/></xs:element>
      <xs:element ref="Enabled" minOccurs="0" maxOccurs="1"/></xs:element>
      <xs:element ref="Visible" minOccurs="0" maxOccurs="1"/></xs:element>
    </xs:choice>
  </xs:sequence>
</xs:element>

```

```

</xs:sequence>
<xs:attribute name="property" type="xs:string" use="required"></xs:attribute>
<xs:attribute name="showLabel" type="xs:boolean" use="optional" default="true"></xs:attribute>
<xs:attribute name="label" type="xs:string" use="optional"></xs:attribute>
<xs:attribute name="labelKey" type="xs:string" use="optional"></xs:attribute>
<xs:attribute name="mnemonic" type="xs:string" use="optional"></xs:attribute>
<xs:attribute name="mnemonicKey" type="xs:string" use="optional"></xs:attribute>
<xs:attribute name="labelWidth" type="xs:positiveInteger" use="optional" default="1"></xs:attribute>
<xs:attribute name="labelAbove" type="xs:boolean" use="optional" default="false"></xs:attribute>
<xs:attribute name="description" type="xs:string" use="optional"></xs:attribute>
<xs:attribute name="descriptionKey" type="xs:string" use="optional"></xs:attribute>
<xs:attribute name="columns" type="xs:positiveInteger" use="optional" default="20"></xs:attribute>
</xs:element>

```

### Parent Eléments

[PropertiesPanel Elément](#), [PropertiesSubPanel Elément](#)

### Enfant Eléments

[Enabled Elément](#), [Layout Elément](#), [Visible Elément](#)

### Eléments connexes Eléments

[CheckBoxControl Elément](#), [CheckBoxGroupControl Elément](#), [ClientDirectoryChooserControl Elément](#), [ClientFileChooserControl Elément](#), [DBConnectionChooserControl Elément](#), [DBTableChooserControl Elément](#), [MultiFieldChooserControl Elément](#), [PasswordBoxControl Elément](#), [PropertyControl Elément](#), [RadioButtonGroupControl Elément](#), [ServerDirectoryChooserControl Elément](#), [ServerFileChooserControl Elément](#), [SingleFieldChooserControl Elément](#), [SingleFieldValueChooserControl Elément](#), [SpinnerControl Elément](#), [TableControl Elément](#), [TextAreaControl Elément](#)

## TextBrowserPanel Elément

Table A-149

Attributs pour TextBrowserPanel

Attribut	Utiliser	Description	Valeurs valides
columns	optional		<i>string</i>
container	<b>obligatoire</b>		<i>string</i>
rows	optional		<i>string</i>
textFormat	<b>obligatoire</b>		<b>plainText</b> <b>html</b> <b>rtf</b>
wrapLines	optional		<i>boolean</i>

### Représentation XML

```

<xs:element name="TextBrowserPanel">
  <xs:sequence>
    <xs:choice>
      <xs:element ref="Layout" minOccurs="0" maxOccurs="1"></xs:element>
      <xs:element ref="Enabled" minOccurs="0" maxOccurs="1"></xs:element>
    </xs:choice>
  </xs:sequence>
</xs:element>

```

```

        <xs:element ref="Visible" minOccurs="0" maxOccurs="1"/></xs:element>
    </xs:choice>
</xs:sequence>
<xs:attribute name="container" type="xs:string" use="required"/></xs:attribute>
<xs:attribute name="textFormat" use="required">
    <xs:simpleType>
        <xs:restriction base="xs:string">
            <xs:enumeration value="plainText"/></xs:enumeration>
            <xs:enumeration value="html"/></xs:enumeration>
            <xs:enumeration value="rtf"/></xs:enumeration>
        </xs:restriction>
    </xs:simpleType>
</xs:attribute>
<xs:attribute name="rows" type="xs:string" use="optional"/></xs:attribute>
<xs:attribute name="columns" type="xs:string" use="optional"/></xs:attribute>
<xs:attribute name="wrapLines" type="xs:boolean" use="optional" default="false"/></xs:attribute>
</xs:element>

```

### Parent Eléments

[Tab Elément](#)

### Enfant Eléments

[Enabled Elément](#), [Layout Elément](#), [Visible Elément](#)

### Eléments connexes Eléments

[ActionButton Elément](#), [ComboBoxControl Elément](#), [ExtensionObjectPanel Elément](#),  
[ModelViewerPanel Elément](#), [SelectorPanel Elément](#), [StaticText Elément](#), [SystemControls Elément](#), [TabbedPanel Elément](#)

## ToolBarItem Elément

Table A-150  
 Attributs pour ToolBarItem

Attribut	Utiliser	Description	Valeurs valides
action	<b>obligatoire</b>		<i>string</i>
offset	optional		<i>nonNegativeInteger</i>
separatorAfter	optional		<i>boolean</i>
separatorBefore	optional		<i>boolean</i>
showIcon	optional		<i>boolean</i>
showLabel	optional		<i>boolean</i>

### Représentation XML

```

<xs:element name="ToolBarItem">
    <xs:attribute name="action" type="xs:string" use="required"/></xs:attribute>
    <xs:attribute name="showLabel" type="xs:boolean" use="optional" default="false"/></xs:attribute>
    <xs:attribute name="showIcon" type="xs:boolean" use="optional" default="true"/></xs:attribute>
    <xs:attribute name="separatorBefore" type="xs:boolean" use="optional" default="false"/></xs:attribute>
    <xs:attribute name="separatorAfter" type="xs:boolean" use="optional" default="false"/></xs:attribute>
    <xs:attribute name="offset" type="xs:nonNegativeInteger" use="optional" default="0"/></xs:attribute>

```

</xs:element>

### **Parent Eléments**

[Controls Élément](#)

## **UserInterface Élément**

Table A-151  
Attributs pour UserInterface

Attribut	Utiliser	Description	Valeurs valides
actionHandler	optional		<i>tout</i>
frameClass	optional		<i>tout</i>

### **Représentation XML**

```
<xs:element name="UserInterface">
  <xs:sequence>
    <xs:element ref="Icons" minOccurs="0"/></xs:element>
    <xs:element ref="Controls" minOccurs="0"/></xs:element>
    <xs:element ref="Tabs" minOccurs="0"/></xs:element>
  </xs:sequence>
  <xs:attribute name="frameClass" use="optional"/></xs:attribute>
  <xs:attribute name="actionHandler" use="optional"/></xs:attribute>
</xs:element>
```

### **Parent Eléments**

[DocumentOutput Élément](#), [Extension Élément](#), [InteractiveDocumentBuilder Élément](#),  
[InteractiveModelBuilder Élément](#), [ModelOutput Élément](#), [Node Élément](#)

### **Enfant Eléments**

[Controls Élément](#), [Icons Élément](#), [Tabs Élément](#)

## **UTF8Format Élément**

### **Représentation XML**

```
<xs:element name="UTF8Format"/></xs:element>
```

### **Parent Eléments**

[FileFormatType Élément](#)

## **Value Élément**

A simple value.

Table A-152  
Attributs pour Value

Attribut	Utiliser	Description	Valeurs valides
value	<b>obligatoire</b>		<i>string</i>

### Représentation XML

```
<xs:element name="Value" type="SIMPLE-VALUE">
  <xs:attribute name="value" type="xs:string" use="required"/></xs:attribute>
</xs:element>
```

### Parent Eléments

[ListValue Elément](#), [Attribute Elément](#), [ListValue Elément](#), [Parameter Elément](#), [Attribute Elément](#), [ListValue Elément](#)

## Values Elément

### Représentation XML

```
<xs:element name="Values">
  <xs:sequence>
    <xs:element name="Value" minOccurs="0" maxOccurs="unbounded"/></xs:element>
  </xs:sequence>
</xs:element>
```

### Parent Eléments

[AddField Elément](#), [MissingValues Elément](#), [ChangeField Elément](#), [MissingValues Elément](#), [Field Elément](#), [MissingValues Elément](#)

### Enfant Eléments

[Value Elément](#)

## Value Elément

Table A-153  
Attributs pour Value

Attribut	Utiliser	Description	Valeurs valides
flagProperty	facultatif		<b>trueValue</b> <b>falseValue</b>
value	<b>obligatoire</b>		<i>string</i>
valueLabel	optional		<i>string</i>

### Représentation XML

```
<xs:element name="Value" minOccurs="0" maxOccurs="unbounded">
  <xs:attribute name="value" type="xs:string" use="required"/></xs:attribute>
  <xs:attribute name="valueLabel" type="xs:string" use="optional"/></xs:attribute>
  <xs:attribute name="flagProperty"/>
```

```

    <xs:simpleType>
      <xs:restriction base="xs:string">
        <xs:enumeration value="trueValue"></xs:enumeration>
        <xs:enumeration value="falseValue"></xs:enumeration>
      </xs:restriction>
    </xs:simpleType>
  </xs:attribute>
</xs:element>

```

### Parent Eléments

### Values Elément

## Values Elément

Table A-154  
Attributs pour Values

Attribut	Utiliser	Description	Valeurs valides
code	<b>obligatoire</b>		<i>integer</i>
displayLabel	facultatif		<i>string</i>
flagValue	facultatif		<i>boolean</i>
value	<b>obligatoire</b>		<i>string</i>

### Représentation XML

```

<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>
  <xs:attribute name="code" type="xs:integer" use="required"></xs:attribute>
  <xs:attribute name="flagValue" type="xs:boolean"></xs:attribute>
  <xs:attribute name="displayLabel" type="xs:string"></xs:attribute>
</xs:element>

```

### Parent Eléments

### Field Elément

### Enfant Eléments

### DisplayLabel Elément

## DisplayLabel Elément

A display label for a field or value in a specified language. The displayLabel attribute can be used where there is no label for a particular language.



Table A-155  
Attributs pour DisplayLabel

Attribut	Utiliser	Description	Valeurs valides
lang	facultatif		NMTOKEN
value	obligatoire		string

### Représentation XML

```
<xs:element name="DisplayLabel" type="DISPLAY-LABEL" minOccurs="0" maxOccurs="unbounded">
  <xs:attribute name="value" type="xs:string" use="required"></xs:attribute>
  <xs:attribute name="lang" type="xs:NMTOKEN" default="en"></xs:attribute>
</xs:element>
```

### Parent Eléments

[Values Elément](#)

## Visible Elément

### Représentation XML

```
<xs:element name="Visible">
  <xs:sequence>
    <xs:group ref="CONDITION-EXPRESSION" minOccurs="0">
      <xs:choice>
        <xs:element ref="Condition"></xs:element>
        <xs:element ref="And"></xs:element>
        <xs:element ref="Or"></xs:element>
        <xs:element ref="Not"></xs:element>
      </xs:choice>
    </xs:group>
  </xs:sequence>
</xs:element>
```

### Parent Eléments

[ActionButton Elément](#), [CheckBoxControl Elément](#), [CheckBoxGroupControl Elément](#), [ClientDirectoryChooserControl Elément](#), [ClientFileChooserControl Elément](#), [ComboBoxControl Elément](#), [DBConnectionChooserControl Elément](#), [DBTableChooserControl Elément](#), [ExtensionObjectPanel Elément](#), [ModelViewerPanel Elément](#), [MultiFieldChooserControl Elément](#), [MultiItemChooserControl Elément](#), [PasswordBoxControl Elément](#), [PropertiesPanel Elément](#), [PropertiesSubPanel Elément](#), [PropertyControl Elément](#), [RadioButtonGroupControl Elément](#), [SelectorPanel Elément](#), [ServerDirectoryChooserControl Elément](#), [ServerFileChooserControl Elément](#), [SingleFieldChooserControl Elément](#), [SingleFieldValueChooserControl Elément](#), [SingleItemChooserControl Elément](#), [SpinnerControl Elément](#), [StaticText Elément](#), [SystemControls Elément](#), [TabbedPanel Elément](#), [TableControl Elément](#), [TextAreaControl Elément](#), [TextBoxControl Elément](#), [TextBrowserPanel Elément](#), [ItemChooserControl Type](#)

### Enfant Eléments

[And Elément](#), [Condition Elément](#), [Not Elément](#), [Or Elément](#)

## Étendus Types

Les types étendus étendent les éléments d'un document XML en ajoutant des attributs et des éléments enfant. Afin d'utiliser un type étendu dans un document XML, précisez le type étendu avec l'attribut `xsi:type` pour cet élément. Vous pouvez ensuite utiliser les attributs et les éléments définis par le type étendu.

### ItemChooserControl Type

Table A-156  
Attributs pour ItemChooserControl

Attribut	Utiliser	Description	Valeurs valides
catalog	<b>obligatoire</b>		<i>string</i>
description	optional		<i>string</i>
descriptionKey	optional		<i>string</i>
label	optional		<i>string</i>
labelAbove	optional		<i>boolean</i>
labelKey	optional		<i>string</i>
labelWidth	optional		<i>positiveInteger</i>
mnemonic	optional		<i>string</i>
mnemonicKey	optional		<i>string</i>
property	<b>obligatoire</b>		<i>string</i>
showLabel	optional		<i>boolean</i>

### Représentation XML

```
<xs:complexType name="ItemChooserControl" mixed="false">
  <xs:sequence>
    <xs:choice>
      <xs:element ref="Layout" minOccurs="0" maxOccurs="1"></xs:element>
      <xs:element ref="Enabled" minOccurs="0" maxOccurs="1"></xs:element>
      <xs:element ref="Visible" minOccurs="0" maxOccurs="1"></xs:element>
    </xs:choice>
  </xs:sequence>
</xs:complexType>
```

### Étend

[ComboBoxControl Élément](#)

### Enfant Elements

[Enabled Élément](#), [Layout Élément](#), [Visible Élément](#)

# Avis

This information was developed for products and services offered worldwide.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:

*IBM Director of Licensing, IBM Corporation, North Castle Drive, Armonk, NY 10504-1785, U.S.A.*

For license inquiries regarding double-byte character set (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

*Intellectual Property Licensing, Legal and Intellectual Property Law, IBM Japan Ltd., 1623-14, Shimotsuruma, Yamato-shi, Kanagawa 242-8502 Japan.*

**Le paragraphe suivant ne s'applique pas au Royaume Uni ni à aucun pays dont la réglementation locale est incompatible avec les dispositions suivantes :** SPSS INC., AN IBM COMPANY, FOURNIT CETTE PUBLICATION « EN L'ÉTAT » SANS GARANTIE D'AUCUNE SORTE, EXPLICITE OU IMPLICITE, Y COMPRIS, MAIS SANS SE LIMITER À, LES GARANTIES TACITES DE NON CONTREFAÇON, DE CONFORMITÉ OU D'APTITUDE À UN USAGE PARTICULIER. Certains états n'autorisent pas les clauses de non garantie explicites ou implicites lors de certaines transactions, par conséquent, il est possible que cette disposition ne vous concerne pas.

Ces informations peuvent contenir des erreurs techniques ou typographiques. Ces informations sont modifiées périodiquement ; ces modifications seront incluses dans les nouvelles versions de cette publication. SPSS Inc. a le droit d'apporter des améliorations et/ou des modifications au(x) produit(s) et/ou au(x) programme(s) décrits dans cette publication à tout moment et sans avertissement préalable.

Toute référence, contenue dans ces informations, à des sites Internet autres que ceux de SPSS et IBM est fournie pour des raisons de commodité uniquement et ne peut, en aucune sorte, être considérée comme de la publicité pour ces sites Internet. Le matériel de ces sites Internet ne fait pas partie du matériel de ce produit SPSS Inc. et l'utilisation de ces sites Internet se fait à vos risques et périls.

Lorsque vous envoyez des informations à IBM ou SPSS, vous octroyez à IBM et SPSS le droit non exclusif d'utiliser ou de distribuer ces informations de la façon dont IBM ou SPSS le juge approprié sans aucune obligation envers vous.

Les informations sur des produits autres que SPSS ont été obtenues auprès des fournisseurs de ces produits, à partir de leurs annonces publiées ou d'autres sources publiques disponibles. SPSS n'a pas effectué de test sur ces produits et ne peut confirmer la validité de leurs performances, de leur compatibilité ou de toute autre prétention associée aux produits autres que SPSS. Les questions concernant les performances des produits autres que SPSS devront être adressées aux fournisseurs de ces produits.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

*IBM Software Group, Attention: Licensing, 233 S. Wacker Dr., Chicago, IL 60606, USA.*

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this document and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement or any equivalent agreement between us.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurements may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

All statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

Ces informations contiennent des exemples de données et de rapports utilisées dans les opérations quotidiennes de l'entreprise. Pour les illustrer le plus précisément possible, ces exemples contiennent les noms de personnes, d'entreprises, de marques et de produits. Tous ces noms sont fictifs et toute ressemblance avec des noms et adresses utilisés par une entreprise réelle ne serait que pure coïncidence.

If you are viewing this information softcopy, the photographs and color illustrations may not appear.

### **Marques commerciales**

IBM, le logo IBM et [ibm.com](http://www.ibm.com) sont des marques commerciales de IBM Corporation, déposées dans de nombreuses juridictions à travers le monde. Une liste mise à jour des marques commerciales IBM est disponible sur Internet à l'adresse <http://www.ibm.com/legal/copytrade.shtml>.

SPSS est une marque commerciale de SPSS Inc., an IBM Company, déposées dans de nombreuses juridictions à travers le monde.

---

Adobe, le logo Adobe, PostScript et le logo PostScript sont des marques déposées ou des marques commerciales de Adobe Systems Incorporated aux États-Unis et/ou dans d'autres pays.

IT Infrastructure Library est une marque déposée du Central Computer and Telecommunications Agency qui fait désormais partie du Office of Government Commerce.

Intel, le logo Intel, Intel Inside, le logo Intel Inside, Intel Centrino, le logo Intel Centrino, Celeron, Intel Xeon, Intel SpeedStep, Itanium et Pentium sont des marques commerciales ou déposées de Intel Corporation ou de ses filiales aux États-Unis ou dans d'autres pays.

Linux est une marque déposée de Linus Torvalds aux États-Unis, dans d'autres pays, ou les deux.

Microsoft, Windows, Windows NT et le logo Windows sont des marques de Microsoft Corporation aux États-Unis, dans d'autres pays, ou les deux.

ITIL est une marque déposée et une marque communautaire déposée du Office of Government Commerce et a été enregistrée auprès du Bureau américain des brevets et des marques de commerce.

UNIX est une marque déposée de The Open Group aux États-Unis et dans d'autres pays.

Cell Broadband Engine est une marque de Sony Computer Entertainment, Inc. aux États-Unis, dans d'autres pays ou les deux et est par conséquent sous contrat de licence.

Java et les marques et logo Java sont des marques de Sun Microsystems, Inc. aux États-Unis, dans d'autres pays, ou les deux.

Linear Tape-Open, LTO, the LTO Logo, Ultrium, and the Ultrium logo are trademarks of HP, IBM Corp. and Quantum in the U.S. and other countries.

D'autres noms de produits ou de services peuvent être des marques de IBM, SPSS ou d'autres entreprises.

- action
  - boutons, 152
  - gestionnaires, 132
- Action élément, 252
- ActionButton élément, 253
- Actions élément, 253
- AddField élément, 254
- AdjustedPropensity élément, 336
- Aide au format HTML
  - liaison à, 203
  - localisation, 213
- Algorithm élément, 331
- algorithme, spécification pour le noeud de création de modèles, 100
- analyse, XML, 241
- And élément, 258
- API à base C, 5
- API côté client, 5, 217
  - classes, 217
  - Utilisation, 217
- API côté serveur, 5, 218
  - architecture, 218
  - fonctions, 225
  - Utilisation, 241
- API Predictive Server (API PS), 218
- API PS, 5
- application de modèles, 121
- architecture
  - API côté serveur, 218
  - système, 1
- arrière-plans, icône, 19
- Attribute élément, 258, 323, 385
- attributs, contrôleur, 159
- AutoModeling élément, 336
- avis juridiques, 399
  
- barre de titre, boîte de dialogue, 25
- barre d'outils
  - éléments, personnalisés, 15, 138
  - zone, boîte de dialogue, 26
- base de données
  - sélecteur de connexion, 165
  - sélecteur de table, 167
- bibliothèques partagées, 44, 71, 227
- bibliothèques, partagées (côté serveur), 44, 71, 227
- BinaryFormat élément, 259
- Boîte de dialogue Paramètres d'algorithme, 115–116, 119
- boîtes de dialogue, conception, 21
- bordures, icône, 18
- Bundle élément, 367
  
- C++
  - fichiers d'aide, 240
  - Langage, 218
- cache, données, 235
  
- cases à cocher, 160
- Catalog élément, 259
- catalogs, 51
- Catalogs élément, 260
- Cell élément, 318
- chaînes
  - codées, 75
  - évaluées, 79
- chaînes codées, 75
- chaînes évaluées, 79
- champ
  - ensembles, 86
  - groups, 105–106
  - métadonnées, 85
- champ, groupes, 105–106
- ChangeField élément, 260
- CheckBoxControl élément, 264
- CheckBoxGroupControl élément, 265
- classe de cadre, 132
- classes, 7
  - API côté client, 217
- client
  - sélecteur de fichier, 163
  - sélecteur de répertoire, 162
- ClientDirectoryChooserControl élément, 266
- ClientFileChooserControl élément, 267
- clonage, modèle, 48
- ComboBoxControl élément, 268
- Command élément, 270
- commandes, boîte de dialogue de noeud, 21
- CommonObjects élément, 270
- compatibilité avec les versions précédentes, maintenance d'une extension, 94
- compatibilité descendante, maintenance, 94
- composants côté client, 1
- composants IU
  - boutons d'action, 152
  - contrôles système, 154
  - texte statique, 153
- Condition élément, 271
- conditions composées, 92
- conditions, dans le fichier de spécifications, 89
  - composées, 92
  - Simple, 92
  - utilisation pour contrôler la visibilité des composants d'écran, 201
  - utilisation pour contrôler les caractéristiques d'affichage, 196
- Constraint élément, 273
- constructeurs, 97
- constructeurs, utilisation, 123
- Constructors élément, 274
- Container élément, 274
- ContainerFile élément, 275
- Containers élément, 297, 315, 317, 340, 347
- ContainerTypes élément, 275

- conteneur
  - contenu, examen, 248
  - fichiers, 71
  - types, 48
- conteneurs, 48, 67
  - examen du contenu de, 248
- contrôle Sélecteur d'un champ unique, 177
- contrôle Sélecteur élément unique, 180
- contrôle Sélecteur multi-élément, 171
- Contrôle Sélecteur multi-élément, fichier de spécifications, 171
- contrôles de propriétés, 151
  - composants IU, 151
  - contrôleurs, 158
  - Elément PropertyControl, 173
  - panneaux de propriétés, 155
- contrôles du panneau de propriétés
  - panneau de propriétés (imbriqué), 158
  - sous-panneau propriétés, 155
- contrôles spinner, 180
- contrôles table, 181
- contrôles, propriété d'écran, 151
  - composants IU, 151
  - contrôleurs, 158
  - panneaux de propriétés, 155
- contrôleurs, 158
  - attributs de, 159
  - case à cocher, 160
  - contrôle de propriété, 173
  - contrôle sélecteur d'un champ unique, 177
  - groupe de boutons radio, 174
  - groupe de cases à cocher, 161
  - sélecteur de connexion à la base de données, 165
  - sélecteur de fichier client, 163
  - sélecteur de fichier serveur, 177
  - sélecteur de répertoire client, 162
  - sélecteur de répertoire serveur, 176
  - sélecteur de table de base de données, 167
  - sélecteur d'un élément unique, 180
  - sélecteur multi-champ, 168
  - sélecteur multi-élément, 171
  - spinner, 180
  - table, 181
  - zone de liste déroulante, 164
  - zone de mot de passe, 172
  - zone de texte, 183–184
- Controls élément, 275
- côté serveur
  - bibliothèques, 44, 71, 227
  - composants, 2
- CreateDocument élément, 276
- CreateDocumentOutput élément, 277
- CreateInteractiveDocumentBuilder élément, 277
- CreateInteractiveModelBuilder élément, 278
- CreateModel élément, 279
- CreateModelApplier élément, 280
- CreateModelOutput élément, 281
- création
  - Modèles, 97
  - modèles interactifs, 97, 108
- DatabaseConnectionValue élément, 282
- DataFile élément, 283
- DataFormat élément, 283
- DataModel élément, 283
- DBConnectionChooserControl élément, 290
- DBTableChooserControl élément, 291
- débogage
  - extensions, 247
  - modification des options de configuration du serveur, 249
  - Onglet Débuguer, boîte de dialogue du noeud, 41, 248
- déclarations de structure, 75
- DefaultValue élément, 292
- DelimitedDataFormat élément, 294
- descripteurs, dans les fonctions de rappel, 221
- désinstallation des extensions, 251
- d'exploitation
  - contrôles, 154
  - menus, 135
- Diagnostic élément, 300, 382
- DisplayLabel élément, 295, 329, 396
- distribution des extensions, 250
- document
  - noeuds de création, 12, 34, 62, 97, 121
  - objets de sortie, 12
  - sortie, définition pour les noeuds, 122
  - types, 49
- Document Conditions requises d'exécution, sortie XML, 235
- Document Détails de l'erreur, sortie XML, 234
- Document Détails de l'état, sortie XML, 238
- Document Génération de SQL, sortie XML, 237
- Document Informations sur le module, sortie XML, 236
- Document Informations sur le noeud, sortie XML, 236
- Document Informations sur l'hôte, sortie XML, 235
- Document Modèle de données, sortie XML, 233
- Document Paramètres, sortie XML, 236
- DocumentBuilder élément, 295
- DocumentGeneration élément, 295
- DocumentOutput élément, 296
- documents, 49, 97
  - création, 121
- DocumentType élément, 297
- Données
  - fonctions d'exploration, création de modèles, 98
  - noeuds de lecture, 10, 33, 62
  - noeuds rédacteurs, 14, 62
  - noeuds Transformation, 11, 34, 62
  - types, 226
- dossier, extension, 6
- Elément Action, fichier de spécifications, 50
- Elément ActionButton, fichier de spécifications, 152

- Elément Actions, fichier de spécifications, 50  
Elément Activé, fichier de spécifications, 196  
Elément AddField, fichier de spécifications, 80, 86  
Elément Algorithm, fichier de spécifications, 100  
Elément And, fichier de spécifications, 88  
Elément Attribute (Catalogs), fichier de spécifications, 51  
Elément Attribute, fichier de spécifications, 77  
Elément Automodeling, fichier de spécifications, 113  
Elément Bundle, fichier de spécifications, 43  
Elément Catalog, fichier de spécifications, 51  
Elément Catalogs, fichier de spécifications, 51  
Elément Cell, fichier de spécifications, 189  
Elément ChangeField, fichier de spécifications, 82  
Elément CheckBoxControl, fichier de spécifications, 160  
Elément CheckBoxGroupControl, fichier de spécifications, 161  
Elément ClientDirectoryChooserControl, fichier de spécifications, 162  
Elément ClientFileChooserControl, fichier de spécifications, 163  
Elément ColumnControl, fichier de spécifications, 182  
Elément ComboBoxControl, fichier de spécifications, 164  
Elément CommonObjects, fichier de spécifications, 45  
Elément Condition, fichier de spécifications, 89  
Elément Constraint, fichier de spécifications, 119  
Elément Constructors, fichier de spécifications, 123  
Elément Container, fichier de spécifications, 67  
Elément ContainerFile des fichiers de sortie, fichier de spécifications, 70  
Elément ContainerFile des fichiers d'entrée, fichier de spécifications, 70  
Elément Containers, fichier de spécifications, 67  
Elément ContainerTypes, fichier de spécifications, 48  
Elément Contrôle sélecteur élément unique, fichier de spécifications, 180  
Elément Controls, fichier de spécifications, 135  
Elément CreateDocument, fichier de spécifications, 123  
Elément CreateDocumentOutput, fichier de spécifications, 124  
Elément CreateInteractiveModelBuilder, fichier de spécifications, 109  
Elément CreateModel, fichier de spécifications, 123  
Elément CreateModelApplier, fichier de spécifications, 126  
Elément CreateModelOutput, fichier de spécifications, 123  
Elément DBConnectionChooserControl, fichier de spécifications, 165  
Elément DBTableChooserControl, fichier de spécifications, 167  
Elément DefaultValue, fichier de spécifications, 70  
Elément Diagnostique, document Détails de l'état, 238  
Elément DocumentBuilder, fichier de spécifications, 121  
Elément DocumentGeneration, fichier de spécifications, 121  
Elément DocumentOutput, fichier de spécifications, 122  
Elément DocumentType, fichier de spécifications, 49  
Elément Enum, fichier de spécifications, 76  
Elément Enumeration, fichier de spécifications, 76  
Elément Exclude, fichier de spécifications, 86  
Elément Execution, fichier de spécifications, 68  
Elément ExpertSettings, fichier de spécifications, 116  
Elément Extension, fichier de spécifications, 41  
Elément ExtensionDetails, fichier de spécifications, 41  
Elément ExtensionObjectPanel, fichier de spécifications, 147  
Elément FieldSet, fichier de spécifications, 86  
Elément ForEach, fichier de spécifications, 83, 86  
élément HelpInfo, fichier de spécifications, 205  
Elément Icon, fichier de spécifications, 134  
Elément Icons, fichier de spécifications, 134  
Elément Include, fichier de spécifications, 86  
Elément InputFields, fichier de spécifications, 102  
Elément InputFiles, fichier de spécifications, 70  
Elément InteractiveModelBuilder, fichier de spécifications, 111  
Elément JarFile, fichier de spécifications, 43  
Elément Layout, fichier de spécifications, 188  
Elément Menu, fichier de spécifications, 135  
Elément MenuItem, fichier de spécifications, 137  
Elément Message, document Détails de l'état, 238  
Elément ModelBuilder, fichier de spécifications, 98  
Elément ModelFields, fichier de spécifications, 104  
Elément ModelGeneration, fichier de spécifications, 104  
Elément ModelingFields, fichier de spécifications, 100  
Elément ModelOutput, fichier de spécifications, 107  
Elément ModelProvider, fichier de spécifications, 64  
Elément ModelType, fichier de spécifications, 49  
Elément ModelViewerPanel, fichier de spécifications, 150  
Elément Module, fichier de spécifications, 71  
Elément MultiFieldChooserControl, fichier de spécifications, 168  
Elément Node, fichier de spécifications, 61  
Elément Not, fichier de spécifications, 88  
Elément Or, fichier de spécifications, 88  
Elément OutputDataModel, fichier de spécifications, 73  
Elément OutputFields, fichier de spécifications, 103  
Elément OutputFiles, fichier de spécifications, 70  
Elément Palette, fichier de spécifications, 54  
Elément Palettes, fichier de spécifications, 54  
Elément Paramètre, document Détails de l'état, 238  
Elément PasswordBoxControl, fichier de spécifications, 172  
Elément Properties, fichier de spécifications, 65  
Runtime, 69  
Elément PropertiesPanel, fichier de spécifications imbriqué, 158  
utilisé à partir de l'onglet ou du sous-panneau propriétés, 149  
Elément PropertiesSubPanel, fichier de spécifications, 155  
Elément Property, fichier de spécifications, 65  
Runtime, 69  
Elément PropertyControl, fichier de spécifications, 173  
Elément PropertyGroup, fichier de spécifications, 115–116  
Elément PropertySet, fichier de spécifications, 47



- Elément PropertySets, fichier de spécifications, 47
- Elément PropertyType, fichier de spécifications, 46
- Elément PropertyTypes, fichier de spécifications, 46
- Elément RadioButtonGroupControl, fichier de spécifications, 174
- Elément RemoveField, fichier de spécifications, 83
- Elément Resources, fichier de spécifications, 42
- Elément ServerDirectoryChooserControl, fichier de spécifications, 176
- Elément ServerFileChooserControl, fichier de spécifications, 177
- Elément SharedLibrary, fichier de spécifications, 44
- Elément SimpleSettings, fichier de spécifications, 115
- Elément SingleFieldChooserControl, fichier de spécifications, 177
- Elément SpinnerControl, fichier de spécifications, 180
- Elément StaticText, fichier de spécifications, 153
- Elément StatusCode, fichier de spécifications, 72, 234
- Elément StatusCodes, fichier de spécifications, 72
- Elément Structure, fichier de spécifications, 77
- Elément SystemControls, fichier de spécifications, 154
- Elément Tab, fichier de spécifications, 141
- Elément TableControl, fichier de spécifications, 181
- Elément Tabs, fichier de spécifications, 140
- Elément TextAreaControl, fichier de spécifications, 183
- Elément TextBoxControl, fichier de spécifications, 184
- Elément TextBrowserPanel, fichier de spécifications, 144
- Elément ToolStripItem, fichier de spécifications, 138
- Elément UserInterface, fichier de spécifications, 68
  - palettes personnalisées, 54
- Elément Visible, fichier de spécifications, 201
- Enabled élément, 298
- Enum élément, 299
- Enumeration élément, 299
- ErrorDetail élément, 299
- espace fichiers, 227
- état de mise en cache, noeud, 18
- étiquettes, positionnement au-dessus du composant, 187
- évaluation des données, 30
- Executable élément, 301
- Execution élément, 302
- exécution externe du processus d'extension, 247
- exécution, externe (du processus d'extension), 247
- exemples de noeuds, CLEF, 32
- exigences graphiques, icônes, 20
- ExpertSettings élément, 338
- extension
  - dossier, 6
  - modules, 218
  - panneaux d'objets, 146
- Extension élément, 302
- ExtensionDetails élément, 303
- ExtensionObjectPanel élément, 303
- extensions, 1
  - désinstallation, 251
  - distribution, 250
  - installation, 251
- localisation, 208
- maintenance de compatibilité descendante, 94
- fenêtre interaction, 108
- fenêtre principale, personnalisation, 139
- fenêtres de sortie, 128
  - conception, 30
  - personnalisé, 201
- fenêtres de sortie personnalisée, 201
- fichier de spécifications, 1, 4, 38
- fichier extension.xml, 6, 38
- fichiers de propriétés (.properties), 209
- fichiers de système d'aide, JavaHelp, 204
- fichiers d'entrée, 5, 69
- fichiers temporaires, 227
  - serveur, 69
- Field élément, 288, 304
- FieldFormats élément, 284, 307
- FieldGroup élément, 286, 309–310
- FieldGroups élément, 286, 310
- FieldName élément, 287, 309, 311
- Fields élément, 288
- FileFormatType élément, 311
- FileFormatTypes élément, 312
- flux de processus, API côté serveur, 223
- fonctions d'accessibilité, 208, 215
- Fonctions de canal, API, 223
- fonctions de module, API, 219
- fonctions de progression, API, 222
- fonctions de rappel, API, 218, 221
- fonctions de service, API, 218–219
- fonctions d'exploration, création de modèles, 98
- fonctions d'hôte, les API, 222
- Fonctions d'itérateur, API, 222
- ForEach élément, 312
- format PMML, sortie de modèle, 64, 149
- fournisseurs, modèle de données, 85
- gestion des erreurs, 240
- glyphes, 17
- graphiques, 49
- groupes de boutons radio, 174
  - modification de l'ordre d'affichage dans, 188
  - modification du nombre de lignes, 187
- groupes de cases à cocher, 161
  - modification de l'ordre d'affichage dans, 188
  - modification du nombre de lignes, 187
- groupes de ressources, 43
- HelpInfo élément, 368
- homologue, 218
  - fonctions, API, 220
- Icon élément, 313

- icône
  - types, 134
  - zone, boîte de dialogue, 25
- Icônes
  - conception, 17
  - création d'images pour, 20
  - exigences graphiques, 20
  - modèle généré, 18
  - noeud, 18
- Icons élément, 314
- identificateurs d'objet, 60
- Identifier élément, 293
- images, création pour les icônes, 20
- InputFields élément, 332
- InputFiles élément, 314
- Installation des extensions, 251
- Interactive
  - modèles, création, 97, 108
- InteractiveDocumentBuilder élément, 315
- InteractiveModelBuilder élément, 316
- interface de programmation d'application (API)
  - à base C, 5
  - API PS, 5, 218
  - côté client, 5, 217
  - côté serveur, 5, 218
  - documentation, 216
  - Java, 5
- interface utilisateur
  - conception, 22
  - définition, 128
- ItemChooserControl type, 398
- itération, dans le fichier de spécifications, 83
  
- JarFile élément, 368
- Java, 7
  - API, 5
  - classes, 43, 50, 74, 132, 146, 173, 201
- JavaHelp
  - liaison à, 204
  - localisation, 213
  
- KeyValue élément, 322
  
- Langage
  - codes, norme ISO, 209
  - définition, 208
- Layout élément, 317
- License élément, 319
- liens d'aide, définition des noeuds, 63
- ligne de commentaires, dans le fichier de spécifications, 39
- lignes, modification du nombre pour les groupes de cases à cocher et de boutons radio, 187
- liste de valeurs, utilisée par les propriétés énumérées, 76
- liste des valeurs, 51
- ListValue élément, 319, 324, 385
  
- localisation
  - extensions, 208
  - messages d'erreur, 239
  - systèmes d'aide, 213
  
- MapEntry élément, 321
- MapValue élément, 320
- marques commerciales, 400
- masquage de palettes et de sous-palettes, 59
- menu
  - éléments, personnalisés, 14, 137
  - zone, boîte de dialogue, 26
- Menu élément, 324
- MenuItem élément, 326
- menus, standard et personnalisés, 14, 135
- Message élément, 301, 383
- messages d'erreur, localisation, 239
- métadonnées, champ, 85
- MissingValues élément, 256, 262, 306, 327
- ModelBuilder élément, 330
- ModelDetail élément, 280
- modèle
  - panneau de visualiseur, 149
- Modèle
  - noeuds applicateurs, 13, 62, 96–97, 126
  - noeuds de création, 11, 35, 62, 96–97
  - nugget, 11
  - objets de sortie, 96
  - signature, 104
  - types, 49
- modèle de données, 4, 233
  - fournisseurs, 85
  - traitement, 228
- Modèles, 96
  - application, 121
  - automatisé, 113
  - création, 97
  - Données, 4
  - Interactive, 108
- ModelEvaluation élément, 335
- ModelField élément, 257, 263, 306
- ModelFields élément, 334
- ModelGeneration élément, 334
- ModelingFields élément, 331
- modélisation automatisée, 113
- ModelOutput élément, 339
- ModelProvider élément, 340
- ModelType élément, 340
- ModelViewerPanel élément, 341
- Module élément, 342
- modules, extension, 218
- MultiFieldChooserControl élément, 342
- MultiItemChooserControl élément, 344
  
- Node élément, 345
- noeud
  - attributs, 62

- document d'informations (XML), 227
- état de mise en cache, 18
- fonctions, API, 222
- icônes, conception, 17
- nom, personnalisé, 29
- types, 4, 226
- noeuds, 4, 9
  - applicateur de modèle, 13
  - créateur de document, 12
  - créateur de modèle, 11
  - Définition, 61
  - ensemble, 113
  - lecteur de données, 10
  - rédacteur de données, 14
  - test des extensions CLEF, 246
  - transformateur de données, 11
- noeuds de modélisation d'ensembles, 113
- noms de scripts, 60
  - définition des noeuds, 63, 93
  - spécification des propriétés, 66
- norme ISO, codes de langue, 209
- Not élément, 347
- nugget, modèle, 11
- NumberFormat élément, 285, 308, 348
- NumericInfo élément, 349
  
- objets de sortie
  - document, 12
  - Modèle, 11
- objets générés
  - graphique ou rapport, 121
  - Modèle, 97
- Onglet Annotations, boîte de dialogue du noeud, 28
- Onglet Modèles, panneau gestionnaire, 107
- Onglet Sorties, panneau gestionnaire, 122
- onglets, définition sur une boîte de dialogue ou une fenêtre, 140
- opérations, dans le fichier de spécifications, 80
- Option élément, 349
- OptionCode élément, 350
- Or élément, 350
- ordre des contrôles, modification, 188
- output
  - fichiers, 5
- OutputDataModel élément, 351
- OutputFields élément, 333
- OutputFiles élément, 352
  
- Palette élément, 352
- palettes
  - masquage, 59
  - spécification d'un noeud, 16, 54, 62
  - Suppression, 59
- panneaux
  - afficheur de modèle, 149
  - navigateur de texte, 144
  - objet d'extension, 146
  - panneau de propriétés, 147
  - panneaux de propriétés, 155
  - sous-panneau propriétés, 155
  - spécification, 143
- Parameter élément, 301, 354, 383
- Parameters élément, 353
- paramètres d'une propriété, examen, 248
- PasswordBoxControl élément, 355
- positions de contrôle précises, définition, 188
- présentation de contrôle propriété personnalisée, 187
  - options avancées, 188
  - simple, 187
- présentation du contrôle propriété
  - personnalisé, 187
  - standard, 185
- présentations, contrôle propriété
  - personnalisé, 187
  - standard, 185
- propensions ajustées, 81
- propensions brutes, 81
- propensions, à spécifier dans le modèle de données, 81, 87
- Propertes élément, 356
- PropertiesPanel élément, 356
- PropertiesSubPanel élément, 358
- Property élément, 359
- PropertyControl élément, 361
- PropertyGroup élément, 362
- PropertyMap élément, 338
- PropertyMapping élément, 338
- PropertySets élément, 362
- PropertyType élément, 363
- PropertyTypes élément, 364
- propriétés
  - Définition, 65
  - énumérées, 75
  - examen des paramètres de, 248
  - panneau, 147
  - panneau (imbriqué), 158
  - Runtime, 69
  - saisi, 46, 78
  - sous-panneau, 155
- propriétés énumérées, 75–76
- propriétés runtime, 69
- propriétés saisies, 46, 78
- propriétés structurées, 77
- propriétés, types de
  - énumérées, 76
  - structurées, 77
  
- raccourcis
  - dans le CLEF, 50, 142
- raccourcis clavier, 142
- RadioButtonGroupControl élément, 364
- Range élément, 328, 366
- RawPropensity élément, 335
- région, configuration dans Windows, 208
- RemoveField élément, 366

- Répercussions SQL, 227
- Resources élément, 367
- ressources, extension, 218
- rôles, dans la sortie de modèle, 87
- rubriques d'aide, spécification pour l'affichage, 206
- Run élément, 369
  
- Section Interface utilisateur, fichier de spécifications, 131
  - palettes personnalisées, 54
- Section Object Definition, fichier de spécifications, 59
- sélecteur multi-champ, 168
- Selector élément, 370
- SelectorPanel élément, 370
- ServerDirectoryChooserControl élément, 371
- ServerFileChooserControl élément, 372
- ServerTempDir élément, 293
- ServerTempFile élément, 292
- serveur
  - contrôle sélecteur de fichier, 177
  - contrôle sélecteur de répertoire, 176
  - fichier temporaire, 69
  - options de configuration, modifications pour le débogage, 249
- SetContainer élément, 373
- SetProperty élément, 374
- SharedLibrary élément, 368
- signature, modèle, 104
- SimpleSettings élément, 337
- SingleFieldChooserControl élément, 374
- SingleFieldValueChooserControl élément, 376
- SingleItemChooserControl élément, 377
- sortie
  - documents (XML), 231
  - fichiers, 69
- sortie de modèle
  - définition pour les noeuds, 107
  - Objets, 11, 97
- sous-palettes
  - masquage, 59
  - spécification d'un noeud, 16, 54, 62
  - Suppression, 59
- SpinnerControl élément, 378
- SPSSDataFormat élément, 379
- StaticText élément, 380
- StatusCode élément, 380
- StatusCodes élément, 381
- StatusDetail élément, 381
- structure de fichier, 6
- Structure élément, 383
- StructuredValue élément, 322, 384
- Suppression
  - palettes et sous-palettes, 59
- SystemControls élément, 386
- systèmes d'aide
  - emplacement, 205
  - liaison à, 203
  - localisation, 213
  
- Tab élément, 387
- TabbedPanel élément, 387
- Tableaux de bord, 49
- TableControl élément, 388
- Tabs élément, 389
- test
  - extensions CLEF, 246
  - noeuds localisés et aide, 214
- text
  - contrôles de zone, 183–184
  - panneaux de navigateur, 144
- TextAreaControl élément, 390
- TextBoxControl élément, 391
- TextBrowserPanel élément, 392
- texte d'info-bulle, spécification, 29, 50
- texte statique, 153
- ToolBarItem élément, 393
- touches d'accès, 142
- types de stockage, 226
- types de valeur, propriété, 75
  
- UserInterface élément, 394
- UTF8Format élément, 394
  
- Value élément, 329, 394–395
- Values élément, 328, 395–396
- VariableImportance élément, 336
- visibilité des composants d'écran, contrôle, 201
- Visible élément, 397
  
- XML
  - API d'analyse, 241
  - déclaration, fichier de spécifications, 40
  - documents de sortie, 231
  
- zone de bouton, boîte de dialogue, 29
- zone de mot de passe, 172
- zone de panneau, boîte de dialogue, 27
- zone d'état, boîte de dialogue, 27
- zone d'onglet, boîte de dialogue, 27
- zones de liste déroulantes, 164