



Informatica® B2B Data Transformation
10.5.2

Guide d'utilisateur

© Copyright Informatica LLC 2001, 2022

Ce logiciel et la documentation associée sont fournis uniquement sous un accord de licence séparé contenant des restrictions d'utilisation et de divulgation. Il est interdit de reproduire ou de transmettre sous quelle que forme et par quel que moyen que ce soit (électronique, photocopie, enregistrement ou autre) tout ou partie de ce document sans le consentement préalable d'Informatica LLC.

Informatica, le logo Informatica, PowerCenter et PowerExchange sont des marques ou des marques déposées d'Informatica LLC aux États-Unis et dans de nombreux autres pays. La liste actuelle des marques de commerce de Informatica est disponible sur le Web à l'adresse <https://www.informatica.com/trademarks.html>. Les autres noms de société ou de produit peuvent être des marques de commerce ou des marques déposées de leurs détenteurs respectifs.

Soumis à vos droits de retrait, le logiciel transmettra automatiquement certaines informations à Informatica (aux États-Unis) concernant l'environnement informatique et réseau dans lequel le Logiciel est déployé et les statistiques du système et d'utilisation des données du déploiement. Cette transmission est considérée comme faisant partie des Services selon la politique de confidentialité d'Informatica et Informatica utilisera et traitera par ailleurs ces informations conformément à la politique de confidentialité d'Informatica disponible sur <https://www.informatica.com/in/privacy-policy.html>. Il est possible de désactiver la collecte d'utilisation dans l'outil Administrator tool.

U.S. GOVERNMENT RIGHTS Les programmes, les logiciels, les bases de données et les documents connexes et les données techniques fournis aux clients du gouvernement américain sont des « logiciels commerciaux » ou des « données techniques commerciales », conformément au règlement fédéral sur les acquisitions et aux règlements supplémentaires propres à l'Agence. En tant que tel, l'utilisation, la duplication, la divulgation, la modification et l'adaptation sont assujetties aux restrictions et aux conditions de licence énoncées dans le contrat gouvernemental applicable et, dans la mesure applicable par les termes du contrat gouvernemental, les droits additionnels énoncés dans la réglementation FAR 52.227-19, licence de logiciel d'ordinateur commercial.

Ce logiciel et sa documentation contiennent des informations appartenant à Informatica LLC, protégées par la loi sur le droit d'auteur et fournies dans le cadre d'un accord de licence prévoyant des restrictions d'utilisation et de divulgation. Toute ingénierie inverse du logiciel est interdite. Il est interdit de reproduire ou transmettre sous quelque forme et par quelque moyen que ce soit (électronique, photocopie, enregistrement ou autre) tout ou partie de ce document sans le consentement préalable de Informatica LLC. Ce logiciel peut être protégé par des brevets américains et/ou internationaux, ainsi que par d'autres brevets en attente.

Consultez les brevets applicables à l'adresse <https://www.informatica.com/legal/patents.html>.

Les renseignements contenus dans cette documentation sont sujets à modification sans préavis. Si vous constatez des problèmes liés à la documentation, merci de les signaler par courriel à l'adresse infa_documentation@Informatica.com.

Les produits Informatica sont garantis conformément aux termes et conditions des accords en vertu desquels ils sont fournis. INFORMATICA FOURNIT LES INFORMATIONS DE CE DOCUMENT « EN L'ÉTAT » SANS GARANTIE D'AUCUNE SORTIE, EXPRESSE OU IMPLICITE, NOTAMMENT AUCUNE GARANTIE DE QUALITÉ MARCHANDE, D'ADAPTATION À UN USAGE PARTICULIER ET D'ABSENCE DE CONTREFAÇON

Certaines parties de ce logiciel et/ou de cette documentation sont soumises à des droits d'auteur détenus par des tiers. Les notifications de tiers requises sont incluses avec le produit.

Date de publication: 2022-05-12

Sommaire

Préface.....	18
Ressources Informatica.	18
Informatica Network.	18
Base de connaissances Informatica.	18
Documentation Informatica.	19
Matrices de disponibilité des produits Informatica.	19
Informatica Velocity.	19
Informatica Marketplace.	19
Support client international Informatica.	19
 Chapitre 1: Présentation de la transformation de données.....	 20
Présentation de Data Transformation.	20
Architecture du processus de Data Transformation.	21
Composants de Data Transformation.	22
 Chapitre 2: Transformation Processeur de données.....	 23
Présentation de la transformation Processeur de données.	23
Vues de la transformation Processeur de données.	24
Ports de la transformation Processeur de données.	25
Ports d'entrée de la transformation Processeur de données.	25
Ports de sortie de la transformation Processeur de données.	26
Ports d'intercommunication.	27
Composant de démarrage.	27
Références.	28
Paramètres de la transformation Processeur de données.	29
Codage de caractère.	29
Règles et directives pour le codage de caractères.	31
Paramètres de sortie.	31
Paramètres de traitement.	33
Paramètres XMap.	34
Configuration d'une sortie XML.	34
Événements.	36
Types d'événements.	36
Vue Événements du processeur de données.	37
Journaux.	37
Journal des événements au moment de la conception.	38
Journal des événements au moment de l'exécution.	38
Affichage d'un journal des événements dans la vue des événements du processeur de données.	39
Journal utilisateur.	39

Développement de la transformation Processeur de données.	40
Créer la transformation Processeur de données.	40
Sélectionner les objets de schéma	41
Créer des objets dans une transformation Processeur de données vide.	41
Créer les ports.	44
Test de la transformation.	44
Exportation et importation de la transformation du processeur de données.	44
Exportation de la transformation Processeur de données en tant que service.	45
Importation de plusieurs services Data Transformation.	45
Importation d'un service Data Transformation	46
Exportation d'un mappage avec une transformation Processeur de données vers PowerCenter.	46
Transformation Processeur de données Validation.	47
Utilisation d'un moteur Data Transformation à vitesse optimisée pour les validations VRL.	47
Transformation Processeur de données dans un environnement non natif.	48
Chapitre 3: Formats d'entrée et de sortie de l'assistant.	49
Présentation des formats d'entrée et de sortie et l'assistant.	49
Avro.	50
Entrée Avro et lecteur de fichier complexe.	50
Compression de données Avro avec le codec Snappy.	51
Configurer une transformation avec une entrée Avro.	51
Configurer une transformation avec une sortie Avro.	54
Bibliothèque de traitement COBOL.	54
Création d'une transformation pour COBOL.	55
Définitions de données COBOL.	55
Procédures de test.	56
Édition d'une transformation pour COBOL.	57
Optimisation du traitement des fichiers COBOL volumineux dans l'environnement Hadoop.	57
JSON.	58
Schémas JSON.	58
Exemple de schéma JSON.	59
Création d'une transformation avec JSON.	60
Parquet.	61
Création d'une transformation avec une entrée ou une sortie Parquet.	61
Configurez l'entrée Parquet du lecteur de fichier complexe.	62
Configurer une transformation avec une sortie Parquet.	62
XML.	63
Création d'une transformation qui transforme du XML.	63
Chapitre 4: Entrée et sortie relationnelle.	65
Présentation des entrées et sorties relationnelles.	65
Entrée relationnelle.	65

Configuration de port d'entrée relationnelle.	66
Directives pour lier les ports d'entrée.	67
Définir les ports d'entrée relationnelle avec la vue Présentation.	67
Ports Clustering_Key.	68
Entrée relationnelle normalisée.	69
Entrée relationnelle pivotée.	70
Entrée relationnelle dénormalisée.	70
Mappage de ports relationnels sur des nœuds hiérarchiques.	71
Sortie relationnelle.	72
Configuration du port de sortie relationnelle.	72
Définir les ports relationnels de sortie dans la vue Présentation.	73
Sortie relationnelle normalisée.	74
Sortie relationnelle pivotée.	74
Sortie relationnelle dénormalisée.	75
Chapitre 5: Utilisation de l'éditeur IntelliScript.	76
Présentation de l'éditeur IntelliScript.	76
Création d'un script.	76
Ouverture d'un éditeur IntelliScript.	77
IntelliScript et Visionneuse de données.	77
Recherche d'ancres.	77
Composants et propriétés.	78
Propriétés de base et avancées.	78
Procédures de modification.	78
Procédure de base de modification.	78
Copier et coller.	79
Glisser et déposer.	79
Rechercher et remplacer.	79
Insertion de composants dans IntelliScript.	79
Modification des propriétés d'un composant.	80
Insertion de tabulations, de nouvelles lignes et d'autres caractères spéciaux.	80
Définition d'un composant global.	80
Affichage de l'aide sur un composant.	81
Icônes IntelliScript.	81
Sauvegarde d'IntelliScript.	82
Menus de l'éditeur IntelliScript.	82
Chapitre 6: XMap.	84
Présentation de XMap.	84
Schémas XMap.	85
Instructions de mappage.	86
Types d'instructions Mappage.	86
Instructions Mappage.	88

Instructions de groupe.	89
Instructions du Groupe de répétition.	91
Instructions Routeur.	93
Instructions Option.	94
Instructions par défaut.	96
Instructions d'exécution XMap.	97
Instruction RunMapplet.	98
Instruction MappletInput.	99
Instruction MappletOutput.	100
Création d'une XMap.	101
Utilisation de la grille de l'éditeur XMap.	102
Création des instructions de mappage.	102
Interface de la grille des instructions de mappage.	103
Expressions XPath.	103
Prédicats.	104
Éditeur d'expression XPath.	108
Fonctions du processeur de données.	108
Exemple d'expressions XPath.	109
Création d'une expression.	110
Variables XMap.	110
Création d'une variable dans l'éditeur XMap.	111
Exemple XMap.	111
Exemple de schéma XML d'entrée.	111
Exemple de schéma XML de sortie.	112
Données d'entrée XML.	113
Hiérarchies XML d'entrée et de sortie.	114
Instructions de mappage dans l'exemple.	114
Exemple d'instructions Groupe.	116
Chapitre 7: Bibliothèques.....	117
Présentation des bibliothèques.	117
Structure de la bibliothèque.	118
Propriétés de l'élément.	118
Gestion de la bibliothèque.	118
Modifier les bibliothèques avec l'éditeur de bibliothèque.	120
Ajout d'un élément à l'aide de l'éditeur de bibliothèque.	120
Modification des propriétés d'un élément avec l'éditeur de bibliothèque.	120
Test d'une bibliothèque.	120
Génération des objets de la bibliothèque.	121
Suppression des objets de la bibliothèque.	121
Modifier les bibliothèques avec l'éditeur IntelliScript.	122

Chapitre 8: Objet de schéma.....	123
Présentation de l'objet de schéma.	123
Fichiers de schéma.	123
Vue de la présentation de l'objet de schéma.	124
Vue Schéma de l'objet de schéma.	125
Propriétés de l'espace de nom.	126
Propriétés de l'élément.	126
Propriétés du type simple.	128
Propriétés du type complexe	129
Propriétés de l'attribut.	130
Vue avancée de l'objet du schéma.	130
Création d'un objet de schéma.	131
Mises à jour des schémas.	132
Synchronisation du schéma.	132
Éditions d'un fichier de schéma.	133
 Chapitre 9: Interface de ligne de commande.....	 136
Présentation de l'interface de ligne de commande.	136
CM_console.	136
 Chapitre 10: Scripts.....	 139
Présentation des scripts.	139
Composants de script.	140
Types de composants.	140
Noms des composants.	141
Ajout d'un composant global.	141
Ajout d'un composant local.	142
Propriétés du composant de script.	142
Propriétés simples.	142
Propriétés avancées.	142
Valeurs de la propriété de composant.	143
Composants de démarrage de script.	144
Définition du composant de démarrage avec l'éditeur IntelliScript.	144
Sources d'exemple.	144
Surlignage de source d'exemple.	145
Définition d'une source d'exemple dans l'éditeur IntelliScript.	145
Affichage d'une source d'exemple.	146
Éditeur IntelliScript.	146
Valider un Script.	147
Échantillons de scripts.	147
Importation d'un échantillon de script.	149

Chapitre 11: Analyseurs.....	150
Présentation des analyseurs.	150
Analyseurs indépendants de la plateforme.	150
Marqueurs Retour à la ligne.	150
Chemins de fichiers.	150
Documentation de référence du composant analyseur.	151
Analyseur.	151
 Chapitre 12: Ports de script.....	 154
Présentation des ports de script.	154
Documentation de référence du composant Port de script.	154
AdditionalInputPort.	154
AdditionalOutputPort.	156
DocList.	159
FileSearch.	159
InputPort.	160
LocalFile.	160
OutputPort.	161
Text.	161
URL.	161
 Chapitre 13: Processeurs de document.....	 163
Présentation des processeurs de document.	163
Définition d'un processeur de document.	163
Affichage de la sortie du processeur de document.	164
Documentation de référence du composant Processeur de document.	164
AsnToXml.	164
ExcelToDataXml.	165
ExcelToXml.	166
ExcelToXml_03_07_10.	167
ExpandFrameSet.	167
ExternalJavaPreProcessor.	167
HIPAAValidator.	167
PdfFormToXml_1_00.	168
PdfToTxt_3_02.	168
PdfToTxt_4.	169
PowerpointToTextML.	169
ProcessByTransformers.	169
ProcessorPipeline.	170
RtfToTextML.	170
WordToXml.	170
XmlToDocument_372.	170

XmlToDocument_45.	171
XmlToExcel.	172
XmlToXlsx.	172
Schéma XML TextML.	173
Éditeur de configuration de tableau PdfToTxt_4.	174
Éditeur Options.	175
Exemple de conversion PDF.	176

Chapitre 14: Formats..... 178

Présentation des formats.	178
Propriétés standard du format.	179
Documentation de référence du composant Format.	179
BinaryFormat.	180
CustomFormat.	181
HtmlFormat.	182
RtfFormat.	183
TextFormat.	183
XmlFormat.	184
Documentation de référence du composant Délimiteurs.	185
CommaDelimited.	186
Délimiteur.	187
DelimiterHierarchy.	187
EnclosingDelimiters.	188
HL7.	188
Positionnel.	188
PostScript.	189
RTF.	189
SGML.	189
SpaceDelimited.	189
TabDelimited.	190
Documentation de référence du composant préprocesseur de format.	190
HtmlProcessor.	191
RtfProcessor.	191

Chapitre 15: Zones de stockage des données..... 192

Présentation des zones de stockage des données.	192
Schémas XML.	192
Codage du schéma.	193
Fichiers de schéma inclus.	193
Espaces de nommage.	193
Contenu mixte.	193
Fonctionnalités de schéma non prises en charge.	193
Précision des données numériques.	194

Utiliser un schéma pour mapper des ancrs.	195
Représentation des zones de stockage des données dans IntelliScript.	195
Mappage de contenu mixte.	195
Mappage de types XSL.	196
Génération de XML valide.	196
Rôle des schémas dans l'analyse.	197
Rôle des schémas dans la sérialisation et le mappage.	197
Variables.	198
Création d'une variable définie par l'utilisateur.	198
Variables système.	198
Mappage d'ancres aux variables.	201
Utilisation de variables dans les actions.	201
Initialisation de variables lors de l'exécution.	201
Documentation de référence du composant Variable.	201
Variable.	202
Zones de stockage des données à occurrences multiples.	202
Attributs.	203
Indexation.	203
Destruction des occurrences.	203
Chapitre 16: Ancres.	205
Présentation des ancrs.	205
Ancres Marqueur et Contenu.	205
Autres types d'ancres.	205
Comment les ancrs et les délimiteurs fonctionnent-ils ensemble ?.	206
Mappage d'ancres Contenu aux zones de stockage des données.	206
Mappage à des variables.	207
Mappage à des zones de stockage des données à occurrences multiples.	207
Mappage à des éléments de contenu mixte.	207
Définition des ancrs.	207
Où définir les ancrs.	208
Séquence d'ancres.	208
Ajout d'une ancre Marqueur ou Contenu.	208
Définition d'une ancre.	209
Propriétés des ancrs standard.	209
Comment un analyseur recherche-t-il les ancrs ?.	210
Phases de recherche.	211
Zone et critères de recherche.	211
Ajustement de la phase de recherche.	212
Ajustement de la zone de recherche.	212
Ajustement des critères de recherche.	214
Utiliser les types de données pour affiner les critères de recherche.	215
Ancres contenant des ancrs imbriquées.	216

Documentation de référence du composant Ancre.	217
Alternatives.	217
Contenu.	219
DelimitedSections.	223
EmbeddedParser.	225
EnclosedGroup.	227
ExtractRecord.	229
FindReplaceAnchor.	230
Groupe.	232
Marqueur.	235
RepeatingGroup.	237
StructureDefinition.	242
Référence du composant de recherche.	246
AttributeSearch.	246
LearnByExample.	247
NewlineSearch.	247
OffsetSearch.	248
PatternSearch.	248
SegmentSearch.	249
TextSearch.	249
TypeSearch.	250
Documentation de référence des sous-composants d'ancre.	250
AllStructure.	251
AllStructureLocal.	251
ChoiceStructure.	252
ChoiceStructureLocal.	253
Connecter.	253
EmbeddedStructure.	254
RecordStructure.	255
RecordStructureLocal.	256
SequenceStructure.	256
SequenceStructureLocal.	257
Chapitre 17: Transformateurs.	259
Présentation des transformateurs.	259
Définition des transformateurs.	259
Utiliser des transformateurs dans les ancrs.	259
Séquences de transformers.	260
Transformateurs par défaut.	260
Utiliser les transformers comme processeurs de document.	260
Utiliser des transformers dans les ancrs de sérialisation.	261
Utiliser des transformers dans les actions.	261
Propriétés d'un transformateur standard.	261

Documentation de référence du composant transformer.	262
AbsURL.	262
AddEmptyTagsTransformer.	262
AddString.	263
Base64Decode.	264
Base64Encode.	264
BidiConvert.	265
CDATADecode.	265
CDATAEncode.	266
ChangeCase.	267
CreateGuid.	267
CreateUUID.	267
DateFormatICU.	268
Dos96HebToAscii.	270
DynamicTable.	271
EbcdicToAscii.	271
EDIFACTValidation.	271
EncodeAsUrl.	272
Encodeur.	272
FormatNumber.	273
FromFloat.	274
FromInteger.	275
FromPackDecimal.	276
FromSignedDecimal.	276
hebrewBidi.	277
HebrewDosToWindows.	277
HebrewEBCDICOldCodeToWindows.	277
hebUniToAscii.	277
hebUtf8ToAscii.	277
HtmlEntitiesToASCII.	277
HtmlProcessor.	278
InjectFP.	278
InjectString.	279
InlineTable.	279
JavaTransformer.	280
LookupTransformer.	280
NormalizeClosingTags.	282
RegularExpression.	283
RemoveMarginSpace.	285
RemoveRtfFormatting.	285
RemoveTags.	286
Replacer.	286

Redimensionner.	287
ReverseTransformer.	288
RtfProcessor.	288
RtfToASCII.	289
SubString.	289
ToFloat.	289
ToInteger.	290
ToPackDecimal.	291
TransformationStartTime.	292
TransformByParser.	292
TransformByProcessor.	293
TransformByService.	294
TransformerPipeline.	295
XMLLookupTable.	295
XSLTTransformer.	296

Chapitre 18: Actions..... 297

Présentation des actions.	297
Comment les actions fonctionnent-elles ?	297
Comparaison entre les actions et les transformateurs.	298
Définition des actions.	298
Propriétés standard des actions.	298
Documentation de référence du composant Action.	299
AddEventAction.	299
AggregateValues.	300
AppendListItems.	302
AppendValues.	303
CalculateValue.	305
CombineValues.	307
CreateList.	308
CustomLog.	309
DateAddICU.	310
DateDiffICU.	311
DownloadFileToDataHolder.	312
DumpValues.	313
EnsureCondition.	314
ExcludeItems.	317
Mappage.	318
Notifier.	320
ResetVisitedPages.	320
RunMapper.	321
RunMapplet.	323
RunParser.	323

RunPCWebService.	325
RunSerializer.	325
RunXMap.	327
SetValue.	328
Trier.	329
ValidateValue.	330
WriteValue.	331
XSLTMap.	333
Documentation de référence du sous-composant Action.	334
OutputDataHolder.	334
OutputFile.	334
ResultFile.	335
StandardErrorLog.	335

Chapitre 19: Sérialseurs..... 336

Présentation des sérialiseurs.	336
Contrôle de la manière de fonctionner de la commande Créer le sérialiseur.	336
Dépanner un serializer généré automatiquement.	338
Création d'un sérialiseur en modifiant le script.	339
Création d'un sérialiseur dans une action RunSerializer.	339
Ancres de sérialisation.	339
Exemples d'ancres de sérialisation.	340
Séquence d'ancres de sérialisation.	340
Propriétés standard des serializers.	341
Documentation de référence du composant Serializer.	341
Sérialiseur.	342
Documentation de référence du composant ancre de sérialisation.	343
AlternativeSerializers.	343
ContentSerializer.	344
DelimitedSectionsSerializer.	345
EmbeddedSerializer.	348
GroupSerializer.	349
RepeatingGroupSerializer.	350
StringSerializer.	353

Chapitre 20: Mappeurs..... 354

Création d'un mappeur.	354
Composants imbriqués dans un mappeur.	354
Exemple de mappeur.	355
Source XML.	355
Sortie XML.	355
Configuration du mappeur.	356
Propriétés standard d'un mapper.	356

Documentation de référence du composant Mappeur.	357
Mappeur.	357
Documentation de référence du composant Ancre du mappeur.	359
AlternativeMappings.	359
EmbeddedMapper.	360
GroupMapping.	361
RepeatingGroupMapping.	362
Chapitre 21: Localisateurs, clés et indexation.	364
Présentation des Localisateurs, des Clés et de l'Indexation.	364
Exemple de localisateurs.	365
Entrée et Sortie.	365
Solution incorrecte.	366
Solution correcte.	366
Exemple d'indexation par clé.	367
Entrée.	367
Sortie.	367
Aperçu de l'approche de la transformation.	368
Configuration du mappeur.	368
Utilisation de l'indexation.	369
Propriétés Source et Target.	370
Propriété source.	370
Propriété target.	374
Propriétés standard des Localisateurs et Clés.	376
Documentation de référence des composants Localisateur et Clé.	376
Clé.	376
Localisateur.	378
LocatorByKey.	379
LocatorByOccurrence.	380
Chapitre 22: Répartiteurs.	382
Présentation des répartiteurs.	382
Répartiteurs de texte.	383
segments.	383
Segments simples.	383
Segments complexes.	383
Exemple.	384
Concaténation de l'en-tête.	384
Sortie d'un répartiteur.	385
Utilisation de marqueurs et de variables dans les répartiteurs.	385
Création d'un Répartiteur.	386
Répartiteurs XML.	387
Propriétés standard des répartiteurs.	389

Référence des composants du répartiteur.	390
ComplexSegment.	390
ComplexXmlSegment.	390
JsonStreamer.	391
MarkerStreamer.	392
SimpleSegment.	393
SimpleXmlSegment.	394
Répartiteur.	395
StreamerVariable.	396
XmlSegment.	397
XmlStreamer.	398
Référence des sous-composants de répartiteur.	399
AddHeaderModifier.	399
AddStringModifier.	400
DoNothingModifier.	400
WellFormedModifier.	400
WriteSegment.	401
Chapitre 23: Validateurs, Notifications et Traitement des échecs.	403
Présentation des validateurs, des notificateurs et du traitement des échecs.	403
Traitement des échecs.	404
Utilisation de la propriété Optional pour gérer les échecs.	404
Écriture d'un message d'échec dans le journal utilisateur.	405
Validateurs.	407
Propriétés standard des validateurs.	408
Référence du composant validateur.	408
AlternativeValidators.	409
EDIFACTValidation.	410
Énumération.	410
LengthEquals.	411
MaxLength.	413
MaxNumber.	413
MinLength.	414
MinNumber.	415
NumberEquals.	416
ValidateByExpression.	417
ValidateByPattern.	419
ValidateByTransformer.	419
ValidateByType.	420
ValidateDate.	421
ValidatorPipeline.	422
Notifications.	423
Documentation de référence du composant Notification.	424

Notification.	424
NotificationGroup.	425
NotificationHandler.	425
NotifyFailure.	426
Chapitre 24: Règles de validation.	427
Présentation de l'objet Règles de validation.	427
Référence de l'élément Règles de validation.	428
Attributs de l'élément Assertion.	428
Attributs de l'élément Liste.	429
Attributs de l'élément Recherche.	429
Attributs de l'élément Règle.	430
Attributs de l'élément Trace.	430
Attributs de l'élément Variable.	431
Éditeur XPath.	431
Extensions XPath.	431
Modifier l'objet Règles de validation dans un éditeur externe.	434
Créer un objet Règles de validation.	434
Importer un service Data Transformation avec l'objet Règles de validation.	435
Chapitre 25: Composants de script personnalisés.	436
Présentation des composants de script personnalisés.	436
Exemple de composant personnalisé.	436
Propriétés du composant personnalisé.	437
Développement d'un composant personnalisé.	437
Exemple d'interface Java.	438
Exemple de composants Java personnalisés.	438
Configuration d'un composant personnalisé.	439
Exemple de scripts contenant des composants personnalisés.	440
Index.	441

Préface

Le *Guide de l'utilisateur de Data Transformation* apprend à concevoir, configurer, tester et déployer la transformation Processeur de données. Ce guide contient des sections de référence détaillées qui document les composants et propriétés des transformations.

Le *Guide de l'utilisateur de Data Transformation* s'adresse aux développeurs, analystes et autres utilisateurs de Data Transformation qui conçoivent et implémentent des transformations. Il suppose que vous possédez les connaissances de base d'Informatica Developer. Il suppose également que vous comprenez le XML, les schémas et les techniques basiques de programmation.

Ressources Informatica

Informatica vous fournit toute une gamme de ressources de produits via Informatica Network et autres portails en ligne. Utilisez ces ressources pour tirer le meilleur parti de vos produits et solutions Informatica, et pour apprendre d'autres utilisateurs et experts en la matière d'Informatica.

Informatica Network

Informatica Network est la passerelle à de nombreuses ressources, y compris la base de connaissances Informatica et le support client international Informatica. Pour accéder à Informatica Network, visitez le site <https://network.informatica.com>.

En tant que membre d'Informatica Network, vous disposez des options suivantes :

- Rechercher les ressources de produits dans la base de connaissances.
- Afficher les informations de disponibilité des produits.
- Créer et vérifier vos dossiers de support.
- Rechercher votre réseau de groupe d'utilisateurs local Informatica et collaborer avec vos pairs.

Base de connaissances Informatica

Utilisez la base de connaissances Informatica pour rechercher des ressources de produits telles que des articles pratiques, des meilleures pratiques, des didacticiels vidéo et des questions fréquemment posées.

Pour effectuer des recherches dans la base de connaissances, visitez le site <https://search.informatica.com>. N'hésitez pas à contacter l'équipe de la base de connaissances Informatica à l'adresse KB_Feedback@informatica.com pour lui faire part de vos questions, commentaires et suggestions concernant la base de connaissances.

Documentation Informatica

Utilisez le portail de documentation Informatica pour explorer une vaste bibliothèque de documentation pour les versions de produits actuelles et récentes. Pour explorer le portail de documentation, visitez le site <https://docs.informatica.com>.

N'hésitez pas à contacter l'équipe Documentation Informatica à l'adresse infa_documentation@informatica.com pour lui faire part de vos questions, commentaires ou suggestions concernant la documentation des produits.

Matrices de disponibilité des produits Informatica

Les matrices de disponibilité des produits (PAM) indiquent les versions des systèmes d'exploitation, les bases de données et les types de source et cible de données pris en charge par une version d'un produit. Vous pouvez parcourir les PAM Informatica à l'adresse <https://network.informatica.com/community/informatica-network/product-availability-matrices>.

Informatica Velocity

Informatica Velocity est un ensemble de conseils et de meilleures pratiques développés par les services professionnels d'Informatica et basés sur les expériences réelles de centaines de projets de gestion des données. Informatica Velocity représente le savoir collectif de consultants d'Informatica qui collaborent avec des organisations du monde entier pour planifier, développer, déployer et gérer des solutions performantes de gestion des données.

Vous trouverez les ressources d'Informatica Velocity à l'adresse <http://velocity.informatica.com>. Si vous avez des questions, des commentaires ou des suggestions sur Informatica Velocity, contactez les services professionnels d'Informatica à l'adresse ips@informatica.com.

Informatica Marketplace

Informatica Marketplace est un forum dans lequel vous pouvez trouver des solutions qui permettent d'augmenter et d'améliorer vos implémentations Informatica. Exploitez les centaines de solutions de développeurs et de partenaires Informatica sur Marketplace pour améliorer votre productivité et accélérer le délai d'implémentation de vos projets. Vous trouverez Informatica Marketplace à l'adresse <https://marketplace.informatica.com>.

Support client international Informatica

Vous pouvez contacter un centre de support international par téléphone ou via le réseau Informatica.

Pour rechercher le numéro de téléphone du support client international Informatica local, visitez le site Web Informatica à l'adresse <https://www.informatica.com/services-and-training/customer-success-services/contact-us.html>.

Pour trouver des ressources de support en ligne sur le réseau Informatica, visitez le site <https://network.informatica.com> et sélectionnez l'option eSupport.

CHAPITRE 1

Présentation de la transformation de données

Ce chapitre comprend les rubriques suivantes :

- [Présentation de Data Transformation, 20](#)
- [Architecture du processus de Data Transformation, 21](#)
- [Composants de Data Transformation, 22](#)

Présentation de Data Transformation

Data Transformation est une application qui traite des fichiers complexes, comme des formats de messagerie, des pages HTML et des documents PDF. Data Transformation transforme également des formats tels que ACORD, HIPAA, HL7, EDI-X12, EDIFACT, et SWIFT.

Data Transformation s'installe par défaut lorsque vous installez Informatica Developer (l'outil Developer tool). Vous pouvez définir une transformation Processeur de données pour transformer des fichiers complexes dans un mappage. Lorsque vous exécutez un mappage avec la transformation Processeur de données, le service d'intégration de données demande au moteur de Data Transformation de traiter les données.

L'application Data Transformation possède les éléments suivants :

Transformation Processeur de données

Transformation qui traite des fichiers complexes dans un mappage. Définissez un objet Script, XMap, Bibliothèque ou Règles de validation de Data Transformation dans l'outil Developer tool pour traiter les données. Vous pouvez inclure la transformation dans un mappage de service de données SQL, un service Web ou un profil de mappage.

Service Data Transformation

Ensemble d'objets de Data Transformation que vous pouvez exporter depuis la transformation Processeur de données et exécuter de manière autonome. Vous exportez un service vers un référentiel de Data Transformation et y exécutez le service.

Référentiel de Data Transformation

Répertoire dans lequel stocker les services exécutables que vous exportez depuis une transformation Processeur de données. Le nom du répertoire du référentiel est ServiceDB.

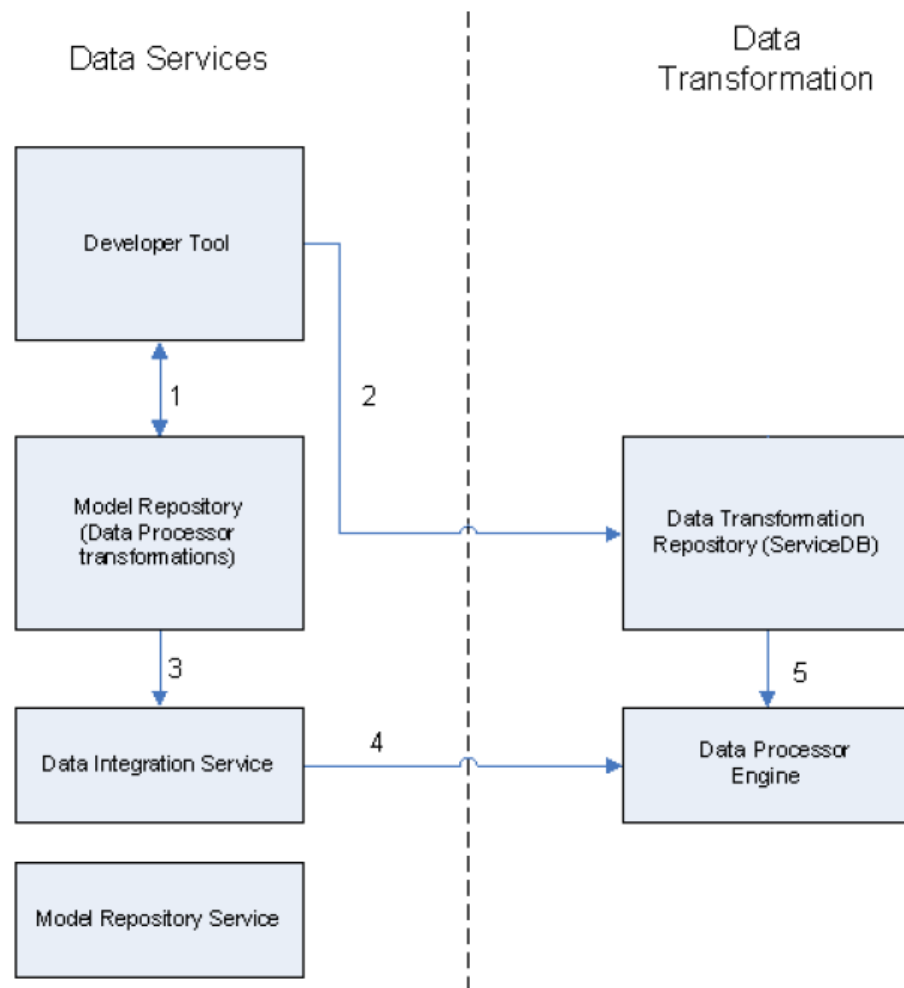
Moteur du processeur de données

Processeur qui exécute des objets dans la transformation Processeur de données ou des services que vous créez dans l'outil Developer tool.

Architecture du processus de Data Transformation

Vous devez installer la Data Transformation afin de configurer et exécuter la transformation Processeur de données dans l'outil Developer. La transformation Processeur de données peut contenir plusieurs scripts ou objets XMap pour transformer les fichiers complexes. Le moteur de Data Transformation exécute les objets Script, Bibliothèque ou XMaps pour transformer les données. Vous pouvez utiliser la transformation Processeur de données dans un service de données, un service Web ou un profil.

La figure suivante montre les composants dans l'application Data Transformation et les composants que vous utilisez pour créer la même fonctionnalité dans l'outil Developer :



1. Créez une transformation Processeur de données dans l'outil Developer. Enregistrez la transformation dans le référentiel Modèle.
2. Exportez la transformation Processeur de données en tant que service de Data Transformation. Exportez le service vers le référentiel de Data Transformation. Vous pouvez exécuter le service depuis le référentiel.
3. Vous pouvez déployer une application qui contient une transformation Processeur de données dans un service d'intégration de données.

4. Le service d'intégration de données exécute l'application et appelle le moteur du processeur de données pour traiter la logique de transformation.
5. Le moteur du processeur de données exécute aussi des services depuis le référentiel de Data Transformation.

Composants de Data Transformation

Lorsque vous définissez un service Data Transformation ou une transformation Processeur de données, vous pouvez combiner plusieurs composants pour transformer les données.

Data Transformation possède les types de composants suivants qui transforment les données :

Bibliothèque

Convertissez une entrée de type de message standard en d'autres formats.

Mappeur

Convertit un document source XML en un autre document XML.

Analyseur

Convertit des documents source en XML. L'entrée peut être de n'importe quel format. La sortie d'un analyseur est au format XML.

Sérialiseur

Convertit un fichier XML dans un autre document. La sortie peut avoir n'importe quel format.

Répartiteur

Divise des documents d'entrée volumineux, par exemple des flux de données de plusieurs gigaoctets, en segments. Le répartiteur fractionne les documents renfermant plusieurs messages ou plusieurs enregistrements.

Transformateur

Modifie des données dans n'importe quel format. Ajoute, supprime, convertit ou modifie du texte. Vous pouvez utiliser des transformateurs avec un analyseur, un mappeur ou un sérialiseur. Vous pouvez également exécuter un transformateur en tant que composant autonome.

XMap

Convertit une source hiérarchique en une autre structure hiérarchique. XMap a les mêmes fonctionnalités qu'un mappeur, mais vous pouvez utiliser une grille dans l'outil Developer tool pour définir le mappage.

CHAPITRE 2

Transformation Processeur de données

Ce chapitre comprend les rubriques suivantes :

- [Présentation de la transformation Processeur de données, 23](#)
- [Vues de la transformation Processeur de données, 24](#)
- [Ports de la transformation Processeur de données, 25](#)
- [Composant de démarrage, 27](#)
- [Références, 28](#)
- [Paramètres de la transformation Processeur de données, 29](#)
- [Événements, 36](#)
- [Journaux, 37](#)
- [Développement de la transformation Processeur de données, 40](#)
- [Exportation et importation de la transformation du processeur de données, 44](#)
- [Transformation Processeur de données Validation, 47](#)
- [Transformation Processeur de données dans un environnement non natif, 48](#)

Présentation de la transformation Processeur de données

La transformation Processeur de données traite les formats de fichier non structurés et semi-structurés dans un mappage. Configurez la transformation pour qu'elle traite les formats de messagerie, les pages HTML, les formats XML et JSON, ainsi que les documents PDF. Vous pouvez également convertir les formats structurés tels que ACORD, HIPAA, HL7, EDI-X12, EDIFACT et SWIFT.

Un mappage utilise une transformation Processeur de données pour faire passer des documents d'un format à un autre. La transformation Processeur de données traite des fichiers de n'importe quel format dans un mappage. Lorsque vous créez une transformation Processeur de données, vous définissez des composants qui convertissent les données.

Une transformation Processeur de données peut contenir plusieurs composants de traitement des données. Chaque composant peut contenir d'autres composants.

Par exemple, vous pourriez recevoir des factures clients dans des fichiers Microsoft Word. Vous configurez une transformation Processeur de données pour analyser les données de chaque fichier Word. Extrayez les

données clients dans une table Client. Extrayez les informations sur la commande dans une table de Commandes.

Lorsque vous créez une transformation Processeur de données, vous définissez une XMap, un script ou une bibliothèque. Une XMap convertit un fichier d'entrée hiérarchique en fichier de sortie hiérarchique à la structure différente. Une bibliothèque convertit un type de messagerie standard en document XML avec une structure de hiérarchie, ou un fichier XML en un format standard. Un script peut analyser des documents source en format hiérarchique, convertir un format hiérarchique en d'autres formats de fichiers ou mapper un document hiérarchique sur un autre format hiérarchique.

Définissez des scripts dans l'éditeur IntelliScript de la transformation Processeur de données. Vous pouvez définir les types de scripts suivants :

- **Analyseur.** Convertit des documents source en XML. La sortie d'un analyseur est toujours au format XML. L'entrée peut être de n'importe quel type de format : texte, HTML, Word, PDF ou HL7.
- **Sérialiseur.** Convertit un fichier XML en document de sortie à n'importe quel format. La sortie d'un sérialiseur peut être à n'importe quel type de format ; il peut s'agir par exemple d'un document texte, d'un document HTML ou d'un fichier PDF.
- **Mappeur.** Convertit un document source XML en une autre structure ou un autre schéma XML. Vous pouvez convertir les mêmes documents XML que dans une XMap.
- **Transformateur.** Modifie les données dans n'importe quel format. Ajoute, supprime, convertit ou modifie du texte. Vous pouvez utiliser des transformateurs avec un analyseur, un mappeur ou un sérialiseur. Vous pouvez également exécuter un transformateur en tant que composant autonome.
- **Répartiteur.** Fractionne en segments des documents d'entrée volumineux, par exemple des flux de données de plusieurs gigaoctets. Le répartiteur traite des documents contenant plusieurs messages ou plusieurs enregistrements, par exemple des fichiers HIPAA ou EDI.

Vues de la transformation Processeur de données

La transformation Processeur de données a plusieurs vues auxquelles vous avez accès lorsque vous configurez la transformation et l'exécutez dans l'outil Developer.

Certaines des vues de la transformation Processeur de données ne s'affichent pas dans l'outil Developer par défaut. Pour modifier les vues pour la transformation, cliquez sur **Fenêtres > Afficher la vue > Autres > Informatica**. Sélectionnez les vues que vous voulez voir.

La transformation Processeur de données a les vues fixes suivantes :

Vue Présentation

Configurez les ports et définissez le composant de démarrage.

Vue Références

Ajoutez ou supprimez des schémas depuis la transformation.

Vue Paramètres

Configurez les paramètres de transformation pour le codage, le contrôle de la sortie et la génération de XML.

Vue Objets

Ajoutez, modifiez ou supprimez les objets Script, XMap et Bibliothèque de la transformation.

Vous pouvez également accéder aux vues suivantes pour la transformation Processeur de données :

Vue Source Hex du processeur de données

Affiche un document d'entrée au format hexadécimal.

Vue Événements du processeur de données

Affiche les informations sur les événements qui se produisent lorsque vous exécutez la transformation dans l'outil Developer. Affiche les événements d'initialisation, d'exécution et de résumé.

Vue Aide du script

Affiche une aide contextuelle pour l'éditeur de script.

Vue Visionneuse de données

Visualisez des données d'entrée d'exemple, exécutez la transformation et affichez les résultats de sortie.

Ports de la transformation Processeur de données

Définissez les ports de la transformation Processeur de données dans la vue **Présentation** de la transformation.

Une transformation Processeur de données peut lire l'entrée d'un fichier, d'un tampon ou d'un tampon réparti depuis un lecteur de fichier complexe. Vous pouvez utiliser un lecteur de fichier plat comme un tampon pour lire un fichier entier en une fois. Vous pouvez également lire un fichier d'entrée depuis une base de données.

Une transformation Processeur de données peut lire l'entrée d'un fichier, d'un tampon ou d'un tampon réparti depuis un lecteur de fichier complexe. Vous pouvez utiliser un lecteur de fichier plat comme un tampon pour lire un fichier entier en une fois. Vous pouvez également lire un fichier d'entrée depuis une base de données.

Les ports de sortie que vous créez varient selon que vous voulez renvoyer une chaîne, des fichiers complexes ou des lignes de données relationnelles depuis la transformation.

Ports d'entrée de la transformation Processeur de données

Lorsque vous créez une transformation Processeur de données, l'outil Developer crée un port d'entrée par défaut. Lorsque vous définissez un port d'entrée supplémentaire dans un composant de démarrage de script, l'outil Developer crée un autre port d'entrée dans la transformation.

Le type d'entrée détermine le type de données que le Service d'intégration de données envoie à la transformation Processeur de données. Le type d'entrée détermine si l'entrée est constituée de données ou est un chemin de fichier source.

Configurez l'un des types d'entrée suivants :

Tampon

La transformation Processeur de données reçoit des lignes de données source dans le port d'entrée. Utilisez le type d'entrée de tampon lorsque vous configurez la transformation pour recevoir des données depuis un fichier simple ou depuis une transformation Informatica.

Fichier

La transformation Processeur de données reçoit le chemin de fichier source dans le port d'entrée. Le composant de démarrage du processeur de données ouvre le fichier source. Utilisez le type d'entrée fichier pour analyser des fichiers binaires tels que des fichiers Microsoft Excel ou Microsoft Word. Vous pouvez également utiliser le type d'entrée fichier pour des fichiers volumineux dont le traitement peut demander beaucoup de mémoire système avec un port d'entrée de tampon.

Paramètre de service

La transformation Processeur de données reçoit des valeurs à appliquer aux variables dans les ports de paramètres de service. Lorsque vous sélectionnez les variables pour recevoir les données d'entrée, l'outil Developer crée un port de paramètre de service pour chaque variable.

Output_Filename

Lorsque vous configurez le port de sortie par défaut pour renvoyer un nom de fichier au lieu de données de ligne, l'outil Developer crée un port Output_Filename. Vous pouvez envoyer un nom de fichier pour le port Output_Filename depuis un mappage.

Lorsque vous définissez un port d'entrée, vous pouvez définir l'emplacement du fichier d'entrée d'exemple pour le port. Un fichier d'entrée d'exemple est un petit exemple de fichier d'entrée. Référez un fichier d'entrée d'exemple lorsque vous créez des scripts. Vous utilisez aussi le fichier d'entrée d'exemple lorsque vous testez la transformation dans la vue **Visionneuse de données**. Définir le fichier d'entrée d'exemple dans le champ **Emplacement de l'entrée**.

Ports de paramètres de service

Vous pouvez créer des ports d'entrée qui reçoivent des valeurs pour les variables. Les variables peuvent contenir n'importe quel type de données telles qu'une chaîne, une date, ou un nombre. Une variable peut également contenir un emplacement pour un document source. Vous pouvez référencer les variables dans un composant Processeur de données.

Lorsque vous créez un port d'entrée pour une variable, l'outil Developer affiche une liste de variables depuis laquelle vous pouvez choisir.

Création de ports de paramètres de service

Vous pouvez créer des ports d'entrée qui reçoivent des valeurs pour les variables. Vous pouvez également supprimer les ports que vous créez depuis les variables.

1. Ouvrez la vue **Présentation** de la transformation Processeur de données.
2. Cliquez sur **Choisir**.
L'outil Developer affiche une liste de variables et indique lesquelles ont déjà des ports.
3. Sélectionnez une ou plusieurs variables.
L'outil Developer crée un port d'entrée de tampon pour chaque variable que vous sélectionnez. Vous ne pouvez pas modifier le port.
4. Pour supprimer un port que vous créez depuis une variable, désactivez la sélection depuis la liste des variables. Lorsque vous désactivez la sélection, l'outil Developer supprime le port d'entrée.

Ports de sortie de la transformation Processeur de données

La transformation Processeur de données a un port de sortie par défaut. Si vous définissez d'autres ports de sortie dans un script, l'outil Developer ajoute les ports dans la transformation Processeur de données. Vous pouvez créer des groupes de ports si vous configurez la transformation de façon à ce qu'elle renvoie des données relationnelles. Vous pouvez également créer des ports de paramètres de service et des ports d'intercommunication.

Port de sortie par défaut

La transformation Processeur de données a un port de sortie par défaut. Lorsque vous créez une sortie relationnelle, vous pouvez définir des groupes de ports de sortie associés à la place du port de sortie par

défaut. Lorsque vous définissez un port de sortie supplémentaire dans un composant du script, l'outil Developer ajoute un autre port de sortie à la transformation.

Configurez l'un des types de sortie suivants pour un port de sortie par défaut :

Tampon

La transformation Processeur de données renvoie XML via le port de sortie. Choisissez le type du fichier Tampon lorsque vous analysez des documents ou lorsque vous mappez XML à d'autres documents XML dans la transformation Processeur de données.

Fichier

Le service d'intégration de données renvoie le nom d'un fichier de sortie dans le port de sortie pour chaque instance ou ligne source. Le composant de la transformation Processeur de données enregistre le fichier de sortie au lieu des données renvoyées via les ports de sortie de la transformation Processeur de données.

Lorsque vous sélectionnez un port de sortie Fichier, l'outil Developer crée un port d'entrée Output_Filename. Vous pouvez transmettre un nom de fichier dans le port du nom de fichier Sortie. La transformation Processeur de données crée le fichier de sortie avec un nom qu'elle reçoit dans ce port.

Si le nom du fichier de sortie est vide, le service d'intégration de données renvoie une erreur de ligne. Lorsqu'une erreur se produit, le service d'intégration de données écrit une valeur Null dans le port de sortie et renvoie une erreur de ligne.

Choisissez le type de sortie Fichier lorsque vous transformez XML en un fichier de données binaires, tel qu'un fichier PDF ou un fichier Microsoft Excel.

Ports d'intercommunication

Vous pouvez configurer des ports d'intercommunication pour toutes les transformations Processeur de données. Les ports d'intercommunication sont des ports d'entrée et de sortie qui reçoivent des données d'entrée et renvoient les mêmes données à un mappage sans les modifier.

Vous pouvez configurer les ports d'intercommunication dans une instance de transformation Processeur de données située dans un mappage.

Pour ajouter un port d'intercommunication, faites glisser un port depuis une autre transformation dans le mappage. Vous pouvez également ajouter des ports dans l'onglet **Ports** de la vue **Propriétés**. Cliquez sur **Nouveau** pour ajouter un port d'intercommunication.

Remarque: Lorsque vous ajoutez des ports d'intercommunication à une transformation Processeur de données avec entrée relationnelle et sortie hiérarchique, ajoutez-les au groupe racine de la structure relationnelle.

Les transformations Processeur de données peuvent inclure des ports d'intercommunication avec des types de données personnalisés.

Composant de démarrage

Un composant de démarrage définit le composant qui démarre le traitement lors de la transformation du processeur de données. Vous pouvez configurer le composant de démarrage dans la vue **Présentation**.

Une transformation Processeur de données peut contenir plusieurs composants de traitement des données. Chaque composant peut contenir d'autres composants. Vous devez identifier quel composant est le point d'entrée de la transformation.

Lorsque vous configurez le composant de démarrage d'une transformation Processeur de données, vous pouvez choisir un composant XMap, Bibliothèque ou Script comme composant de démarrage. En termes de scripts, vous pouvez sélectionner l'un des types de composants suivants :

- **Analyseur.** Convertit des documents source en XML. L'entrée peut avoir tout type de format, par exemple : texte, HTML, Word, PDF ou HL7.
- **Mappeur.** Convertit un document source XML en une autre structure ou schéma XML.
- **Sérialiseur.** Convertit un fichier XML en document de sortie à n'importe quel format.
- **Répartiteur.** Fractionne en segments des documents d'entrée volumineux, par exemple des flux de données de plusieurs gigaoctets.
- **transformateur.** Modifie les données de tout format. Ajoute, supprime, convertit ou modifie du texte. Vous pouvez utiliser des transformateurs avec un analyseur, un mappeur ou un sérialiseur. Vous pouvez également exécuter un transformateur en tant que composant autonome.

Remarque: Si le composant de démarrage n'est pas de type XMap ou Bibliothèque, vous pouvez également configurer le composant de démarrage dans un script plutôt que dans la vue **Présentation**.

Références

Vous pouvez définir les références de transformation telles que les références de schéma ou de mapplet en sélectionnant un schéma ou un mapplet à utiliser comme référence. Certaines transformations Processeur de données requièrent un schéma hiérarchique pour définir la hiérarchie d'entrée ou de sortie des composants de la transformation. Pour utiliser le schéma au sein de la transformation, vous devez définir une référence de schéma relatif à la transformation. Vous pouvez également utiliser une action spécifique nommée Action RunMapplet pour appeler un mapplet dans une transformation Processeur de données. Pour appeler un mapplet, vous devez commencer par définir une référence de mapplet relative à la transformation.

Vous pouvez définir les références de transformation telles que les références de schéma ou de mapplet dans la vue **Références** de la transformation.

Références de schéma

La transformation Processeur de données fait référence aux objets de schéma du référentiel modèle. Les objets de schéma peuvent exister dans le référentiel avant de créer la transformation. Vous pouvez également importer les schémas depuis la vue **Références** de la transformation.

Le codage du schéma doit correspondre au codage d'entrée des objets Sérialiseur ou Mappeur. Le codage du schéma doit correspondre au codage de sortie des objets Analyseur ou Mappeur. Configurez le codage de travail dans la vue **Paramètres** de la transformation.

Un schéma peut faire référence à d'autres schémas. L'outil Developer affiche l'espace de noms et le préfixe de chaque schéma auquel la transformation Processeur de données fait référence. Lorsque vous faites référence à plusieurs schémas avec des espaces de nom vides, la transformation n'est pas valide.

Références de mapplet

Vous pouvez appeler un mapplet depuis une transformation Processeur de données à l'aide de l'action RunMapplet. Avant d'ajouter l'action RunMapplet à un composant de la transformation Processeur de données, vous devez d'abord définir une référence relative au mapplet à appeler.

Paramètres de la transformation Processeur de données

Configurez les pages de code, les options de traitement XML et les paramètres de journalisation dans la vue **Paramètres** de la transformation Processeur de données.

Codage de caractère

Un encodage de caractères est un mappage des caractères issus d'une langue ou d'un groupe de langues en code hexadécimal.

Lorsque vous concevez un script, vous devez définir le codage des documents d'entrée et le codage des documents de sortie. Définissez le codage de travail pour définir la manière dont l'éditeur IntelliScript affiche les caractères et la manière dont la transformation Processeur de données traite les caractères.

Codage de travail

Le codage de travail est la page de code pour les données en mémoire et la page de code pour les données qui s'affichent dans l'interface utilisateur et les fichiers de travail. Vous devez sélectionner un codage de travail qui est compatible avec le codage des schémas que vous référencez dans la transformation Processeur de données.

Le tableau suivant montre les paramètres de codage de travail :

Paramètre	Description
Utiliser la page de code par défaut du processeur de données	Utilisez le codage par défaut depuis la transformation Processeur de données.
Autre	Sélectionnez un codage dans la liste.
Codage de caractères XML spéciaux	Détermine la représentation des caractères XML spéciaux. Vous pouvez sélectionner Aucun ou XML . <ul style="list-style-type: none">- Aucun. Laissez comme & &lt; &gt; &quot; &apos; Les références d'entité pour les caractères spéciaux XML sont interprétées comme du texte. Par exemple, le caractère > s'affiche comme &gt; La valeur par défaut est Aucun.- XML. Convertir en & < > " ' Les références d'entité pour les caractères spéciaux XML sont interprétées comme des caractères réguliers. Par exemple, &gt; apparaît comme le caractère suivant : >

Encodage de l'entrée

Le codage d'entrée détermine la manière dont les données de caractères sont encodées dans les documents d'entrée. Vous pouvez configurer le codage de ports d'entrée supplémentaires dans un script.

Le tableau suivant décrit les paramètres de codage dans la zone **Entrée** :

Paramètre	Description
Utilisez le codage spécifié dans le document d'entrée	Utilisez la page de code définie dans le document source, telle que l'attribut de codage d'un document XML. Si aucune spécification de codage n'est définie pour le document source, la transformation Processeur de données utilise les paramètres de codage de la vue Paramètres .
Utiliser le codage de travail	Utilisez le même codage que le codage de travail.
Autre	Sélectionnez le codage d'entrée depuis une liste déroulante.
Codage de caractères XML spéciaux	Détermine la représentation des caractères XML spéciaux. Vous pouvez sélectionner Aucun ou XML . <ul style="list-style-type: none">- Aucun. Laissez comme &amp; &lt; &gt; &quot; &apos; Les références d'entité pour les caractères spéciaux XML sont interprétées comme des textes. Par exemple, le caractère > s'affiche comme &gt;; La valeur par défaut est Aucun.- XML. Convertir en & < > " ' Les références d'entité pour les caractères spéciaux XML sont interprétées comme des caractères réguliers. Par exemple, &gt; apparaît comme le caractère suivant : >
Ordre des octets	Décrit la manière dont les caractères multi-octets s'affichent dans le document d'entrée. Vous pouvez sélectionner les options suivantes : <ul style="list-style-type: none">- Little-endian. L'octet le moins significatif s'affiche en premier. Valeur par défaut.- Big-endian. L'octet le plus significatif s'affiche en premier.- Aucune conversion binaire.

Encodage de la sortie

Le codage de sortie détermine la manière dont les données de caractère sont encodées dans le document de sortie principal.

Le tableau suivant décrit les paramètres de codage dans la zone **Sortie** :

Paramètre	Description
Utiliser le codage de travail	Le codage de sortie est identique au codage de travail.
Autre	L'utilisateur sélectionne l'encodage de sortie dans la liste.

Paramètre	Description
Codage de caractères XML spéciaux	<p>Détermine la représentation des caractères XML spéciaux. Vous pouvez sélectionner Aucun ou XML.</p> <ul style="list-style-type: none"> - Aucun. Laissez comme &amp; &lt; &gt; &quot; &apos; Les références d'entité pour les caractères spéciaux XML sont interprétées comme des textes. Par exemple, le caractère > s'affiche comme &gt; Valeur par défaut. - XML. Convertir en & < > " ' Les références d'entité pour les caractères spéciaux XML sont interprétées comme des caractères réguliers. Par exemple, &gt; apparaît comme le caractère suivant : >
Identique au codage d'entrée	Le codage de sortie est identique au codage de sortie
Ordre des octets	<p>Décrit la manière dont les caractères multi-octets s'affichent dans le document d'entrée. Vous pouvez sélectionner les options suivantes :</p> <ul style="list-style-type: none"> - Little-endian. L'octet le moins significatif s'affiche en premier. Valeur par défaut. - Big-endian. L'octet le plus significatif s'affiche en premier. - Aucune conversion binaire.

Règles et directives pour le codage de caractères

Utilisez les règles et directives suivantes lorsque vous configurez des codages :

- Pour augmenter les performances, définissez le codage de travail pour qu'il soit le même codage que le document de sortie.
- Définissez le codage d'entrée pour le même codage comme le document d'entrée.
- Définissez le codage de sortie pour le même codage comme le document de sortie.
- Pour les langues qui ont des caractères multi-octets, définissez le codage de travail sur UTF-8. Pour le codage d'entrée et de sortie, vous pouvez utiliser un codage Unicode tel qu'UTF-8 ou une page de code double octet telle que Big5 ou Shift_JIS.

Paramètres de sortie

Configurez les paramètres de contrôle de sortie pour contrôler si la transformation Processeur de données crée des journaux d'événements et enregistre des documents de sortie.

Vous pouvez contrôler les types de messages que la transformation Processeur de données écrit dans le journal d'événement au moment de la conception. Si vous enregistrez les documents d'entrée analysés avec les journaux d'événements, vous pouvez afficher le contexte dans lequel l'erreur s'est produite dans la vue **Événement**.

Le tableau suivant décrit les paramètres dans la zone **Événements au moment de la conception** :

Paramètre	Description
Journaliser les événements au moment de la conception	Détermine si vous souhaitez créer un journal des événements au moment de la conception. Par défaut, la transformation Processeur de données journalise les notifications, les avertissements et les défaillances dans le journal d'événement au moment de la conception. Vous pouvez exclure les types d'événements suivants : <ul style="list-style-type: none"> - Notifications - Avertissements - Échecs
Enregistrer les documents analysés	Détermine le moment où la transformation Processeur de données enregistre un document d'entrée analysé. Vous pouvez sélectionner les options suivantes : <ul style="list-style-type: none"> - Toujours. - Jamais - En cas d'échec La valeur par défaut est Toujours.

Le tableau suivant décrit les paramètres dans la zone **Événements d'exécution** :

Paramètre	Description
Journaliser les événements d'exécution	Détermine la création ou non d'un journal des événements lors de l'exécution de la transformation depuis un mappage. <ul style="list-style-type: none"> - Jamais. - En cas d'échec La valeur par défaut est Jamais.

Le tableau suivant décrit les paramètres dans la zone **Sortie** :

Paramètre	Description
Désactiver la sortie automatique	Détermine si la transformation Processeur de données écrit la sortie dans le fichier de sortie standard. Désactivez la sortie standard dans le cas suivant : <ul style="list-style-type: none"> - Vous transmettez la sortie d'un analyseur à l'entrée d'un autre composant avant la création par la transformation d'un fichier de sortie. - Vous utilisez une action WriteValue pour écrire les données directement dans la sortie depuis un script au lieu de transmettre des données par les ports de sortie.
Désactiver la compression des valeurs	Détermine si la transformation Processeur de données utilise la compression des valeurs pour optimiser l'utilisation de la mémoire. Important : ne désactivez la compression des valeurs que lorsque le support client international Informatica vous le conseille.

Le tableau suivant décrit les paramètres dans la zone **Mode de collecte des ports de sortie binaires**. Vous pouvez sélectionner une de ces options pour la sortie binaire d'une transformation hiérarchique en

relationnelle avec sortie XML, Avro ou Parquet, ou pour un analyseur de transformation Processeur de données avec sortie Avro ou Parquet.

Paramètre	Description
Collecter les lignes d'entrée vers une seule sortie	Détermine si la transformation Processeur de données cumule l'entrée relationnelle dans un port de sortie binaire unique.
Fractionner la sortie lors du dépassement de la taille	Détermine si la transformation Processeur de données divise la sortie en fragments en fonction d'une taille de sortie maximale indiquée.
Sortir une ligne pour chaque ligne (ne pas collecter)	Détermine si la transformation Processeur de données transmet la sortie dans des lignes distinctes.

Paramètres de traitement

Les paramètres de traitement définissent la manière dont la transformation Processeur de données traite un élément sans un type de données défini. Les paramètres affectent les scripts. Les paramètres n'affectent pas les éléments traités par un XMap.

Le tableau suivant décrit les paramètres de traitement qui affectent le traitement XML dans les scripts :

Paramètre	Description
Traiter comme xs:string	La transformation Processeur de données traite un élément sans type comme une chaîne. Dans la boîte de dialogue Choisir XPath , l'élément ou l'attribut s'affiche comme un nœud unique.
Traiter comme xs:anyType	La transformation Processeur de données traite un élément sans type comme anyType. Dans la boîte de dialogue Choisir XPath , l'élément ou l'attribut apparaît comme une arborescence de nœuds. Un nœud est de type xs:string et tous les types de données complexes nommés apparaissent dans l'arborescence de nœuds.

Le tableau suivant décrit un paramètre de traitement qui affecte le traitement du répartiteur :

Paramètre	Description
Taille de fragment du répartiteur	Ce paramètre définit la quantité de données lue à chaque fois par le répartiteur depuis un flux de fichier d'entrée. La transformation Processeur de données applique ce paramètre à un répartiteur avec une entrée de fichier.

Le tableau suivant décrit un paramètre de traitement qui affecte le traitement d'une transformation relationnelle à hiérarchique :

Paramètre	Description
Appliquer une validation stricte	Ce paramètre détermine si la transformation Processeur de données effectue la validation stricte de l'entrée hiérarchique. Lorsque la validation stricte s'applique, le fichier d'entrée hiérarchique doit se conformer strictement à son schéma. Cette option peut s'appliquer lorsque le mode Processeur de données est défini sur Mappage de sortie , ce qui crée des ports de sortie pour la sortie relationnelle. Elle ne s'applique pas aux mappages générés avec une entrée JSON qui sont issus de versions antérieures à la version 10.2.1.
Normaliser l'entrée XML	Ce paramètre détermine si la transformation Processeur de données normalise l'entrée XML. Par défaut, la transformation effectue la normalisation de l'entrée XML. Dans certains cas, vous pouvez choisir d'ignorer la normalisation automatique pour augmenter les performances.

Paramètres XMap

Le paramètre XMap définit la façon dont la transformation Processeur de données traite les éléments d'entrée XMap qui ne sont pas transformés en éléments de sortie. Les éléments non lus sont transmis à un port dédié nommé **XMap_Unread_Input_Values**. Le paramètre prend effet uniquement lorsque la XMap est sélectionnée en tant que composant de démarrage. Le paramètre n'affecte pas les éléments traités par la XMap.

Pour envoyer des éléments XMap non lus vers un port dédié, activez le paramètre **Écrivez les éléments non lus vers un port de sortie supplémentaire**.

Configuration d'une sortie XML

Les paramètres de la génération XML définissent les caractéristiques des documents de sortie XML.

Le tableau suivant décrit les paramètres de la génération XML dans la zone **Titre de schéma** :

Paramètre	Description
Emplacement du schéma	Définit l'emplacement du schéma pour l'élément racine du document de sortie principal.
Pas d'emplacement de schéma d'espace de noms	Définit l'attribut xsi:noNamespaceSchemaLocation de l'élément racine du document de sortie principal.

Configurez les paramètres du mode de sortie XML pour déterminer comment la transformation Processeur de données traite les éléments ou les attributs manquants dans le document d'entrée XML. Le tableau suivant décrit les paramètres de la génération XML dans la zone **Mode de sortie XML** :

Paramètre	Description
En l'état	N'ajoute ni ne supprime les éléments vides. La valeur par défaut est activée.
Complet	Tous les éléments requis et optionnels définis dans le schéma de sortie sont écrits dans la sortie. Les éléments qui n'ont pas de contenu sont enregistrés comme des éléments vides.
Compact	Supprime les éléments vides de la sortie. Si l'option Ajouter pour des éléments est activée, alors la transformation Processeur de données supprime seulement les éléments optionnels. Si l'option Ajouter pour des éléments est désactivée, alors la transformation Processeur de données supprime tous les éléments vides. Il est possible que la sortie XML ne soit pas valide.

Le tableau suivant décrit les paramètres de la génération XML dans la zone **Valeurs par défaut pour les nœuds requis** :

Paramètre	Description
Ajouter pour des éléments	Lorsque le schéma de sortie définit une valeur par défaut pour un élément requis, la sortie inclut l'élément avec une valeur par défaut. La valeur par défaut est activée.
Ajouter pour des attributs	Lorsque le schéma de sortie définit une valeur par défaut pour un attribut requis, la sortie inclut l'attribut avec sa valeur par défaut. La valeur par défaut est activée.
Valider les valeurs ajoutées	Détermine si la transformation Processeur de données valide les éléments vides ajoutés par la sortie du mode Full. La valeur par défaut est désactivée. Si l'option Valider les valeurs ajoutées est activée et que le schéma n'autorise pas les éléments vides, il est possible que la sortie XML ne soit pas valide.

Le tableau suivant décrit les paramètres de la génération XML dans la zone **Traitement des instructions** :

Paramètre	Description
Ajouter des instructions de traitement XML	Définit le codage des caractères et la version XML du document de sortie. Option sélectionnée par défaut.
Version XML	Définit la version XML. Le paramètre de la version XML présente les options suivantes : - 1.0 - 1.1 Valeur par défaut : 1.0.

Paramètre	Description
Codage	Définit le codage de caractères spécifié dans l'instruction de traitement. Le paramètre Codage présente les options suivantes : <ul style="list-style-type: none"> - Identique au codage de sortie. Le codage de sortie dans l'instruction de traitement est identique au codage de sortie défini dans les paramètres de la transformation Processeur de données. - Personnalisé. Définit le codage de sortie dans l'instruction de traitement. L'utilisateur saisit la valeur dans le champ.
Ajouter des instructions de traitement personnalisées	Ajoute des instructions de traitement supplémentaires pour le document de sortie. Entrez l'instruction de traitement exactement telle qu'elle apparaît dans le document de sortie. La valeur par défaut est désactivée.

Le tableau suivant décrit les paramètres de la génération XML dans la zone **Racine XML** :

Paramètre	Description
Ajouter un élément racine XML	Ajoute un élément racine pour le document de sortie. Utilisez cette option lorsque le document de sortie contient plusieurs occurrences de l'élément racine défini dans le schéma de sortie. La valeur par défaut est désactivée.
Nom de l'élément racine	Définit un nom pour l'élément racine à ajouter au document de sortie.

Événements

Un événement est un enregistrement d'une étape de traitement d'un composant dans la transformation Processeur de données. Dans un objet Script ou Bibliothèque, chaque ancre, action ou transformateur génère un événement. Dans une instruction XMap, chaque mappage génère un événement.

Vous pouvez afficher les événements dans la vue **Événements du processeur de données**.

Types d'événements

La transformation Processeur de données écrit les événements dans des fichiers de journalisation. Chaque événement a un type d'événement qui indique si l'événement a réussi, échoué ou s'il a été exécuté avec des erreurs.

Un composant peut générer un ou plusieurs événements. Le composant peut réussir ou échouer selon que les événements réussissent ou échouent. Si un événement échoue, un composant échoue.

Le tableau suivant décrit les types d'événements que la transformation Processeur de données génère :

Type d'événement	Description
Notification	Opération normale.
Avertissement	La transformation Processeur de données s'est exécutée, mais une condition inattendue s'est produite. Par exemple, la transformation Processeur de données a écrit des données dans le même élément plusieurs fois. Chaque fois que l'élément est remplacé, la transformation Processeur de données génère un avertissement.

Type d'événement	Description
Échec	La transformation Processeur de données s'est exécutée, mais un composant a échoué. Par exemple, un élément d'entrée requis était vide.
Échec facultatif	La transformation Processeur de données s'est exécutée, mais un composant facultatif a échoué. Par exemple, une ancre optionnelle était manquante dans le document source.
Erreur irrécupérable	La transformation Processeur de données a échoué en raison d'une erreur grave. Par exemple, le document d'entrée n'existait pas.

Vue Événements du processeur de données

La vue **Événements du processeur de données** affiche les événements lorsque vous exécutez une transformation Processeur de données depuis l'outil Developer.

La vue **Événements du processeur de données** a un panneau **Navigation** et un panneau **Détails**. Le panneau de navigation contient un arbre de navigation. L'arbre de navigation répertorie les composants que la transformation a exécutés par ordre chronologique. Chaque nœud a une icône qui représente l'événement le plus sérieux, sous lui dans l'arborescence. Lorsque vous sélectionnez un nœud dans le panneau **Navigation**, les événements s'affichent dans le panneau **Détails**.

L'arbre de navigation contient les nœuds de niveau supérieur suivants :

- Initialisation du service. Décrit les fichiers et les variables que la transformation Processeur de données initialise.
- Exécution. Répertorie les composants exécutés par un objet Script, Bibliothèque ou XMap.
- Résumé. Affiche les statistiques du traitement.

Lorsque vous exécutez un XMap, chaque nom de nœud dans le panneau de navigation a un nombre entre crochets, par exemple [5]. Pour identifier l'instruction qui a généré les événements pour le nœud, faites un clic droit dans la grille d'instructions et sélectionnez **Accéder au numéro de ligne**. Saisissez le numéro du nœud.

Lorsque vous exécutez un script et double-cliquez sur un événement dans le panneau **Navigation** ou le panneau **Détails**, l'éditeur de script met en évidence le composant du script qui a généré l'événement. Le panneau **Entrée** de la vue **Visionneuse de données** met en évidence la partie du document de la source d'exemple qui a généré l'événement.

Journaux

Un journal contient un enregistrement de la transformation Processeur de données. La transformation Processeur de données écrit des événements dans les fichiers journaux.

La transformation Processeur de données crée les types suivants de journaux :

Journal des événements au moment de la conception

Le journal des événements de conception contient les événements qui se produisent lorsque vous exécutez la transformation Processeur de données dans la vue **Visionneuse de données**. Affichez le journal de conception dans la vue **Événements**.

Journal des événements au moment de l'exécution

Le journal des événements d'exécution contient les événements qui se produisent lorsque vous exécutez la transformation Processeur de données dans un mappage. Vous pouvez afficher le journal des événements d'exécution dans un éditeur de texte ou vous pouvez faire glisser un journal des événements d'exécution dans la vue **Événements** de la transformation Processeur de données.

Journal utilisateur

Le journal utilisateur contient les événements que vous configurez pour les composants dans un script. La transformation Processeur de données enregistre le journal utilisateur lorsque vous l'exécutez depuis la vue **Visionneuse de données** et lorsque vous l'exécutez dans un mappage. Vous pouvez afficher le journal utilisateur dans un éditeur de texte.

Journal des événements au moment de la conception

Le journal des événements au moment de la conception contient les événements qui se produisent lorsque vous exécutez la transformation Processeur de données depuis la **Visionneuse de données** dans l'outil Developer.

Lorsque vous exécutez une transformation Processeur de données depuis la vue **Visionneuse de données**, le journal des événements au moment de la conception s'affiche dans la vue **Événements du processeur de données**. Par défaut, le journal des événements au moment de la conception contient les notifications, les avertissements et les échecs. Dans les paramètres de la transformation, vous pouvez configurer la transformation Processeur de données pour qu'elle exclue un ou plusieurs types d'événements du journal.

Lorsque vous enregistrez les documents d'entrée dans le journal, vous pouvez cliquer sur un événement dans la vue **Événements du processeur de données** pour trouver l'emplacement dans le document d'entrée qui a généré l'événement. Lorsque vous configurez les paramètres de la transformation Processeur de données, vous pouvez choisir d'enregistrer les fichiers d'entrée à chaque exécution ou uniquement en cas d'échec.

Le journal des événements au moment de l'exécution est appelé `events.cme`. Vous pouvez trouver le journal des événements au moment de l'exécution, pour la dernière exécution de la transformation Processeur de données, dans le répertoire suivant :

```
C:\<Installation_directory>\clients\DT\CMReports\Init\events.cme
```

La transformation Processeur de données écrase le journal des événements au moment de l'exécution à chaque fois que vous exécutez la transformation dans la **Visionneuse de données**. Renommez le journal des événements au moment de l'exécution si vous voulez l'afficher après une exécution ultérieure de la transformation ou si vous voulez comparer les journaux de différentes exécutions. Lorsque vous fermez l'outil Developer, le développeur ne sauvegarde aucun fichier dans le

Journal des événements au moment de l'exécution

Le journal des événements au moment de l'exécution enregistre les événements qui se produisent pendant l'exécution de la transformation Processeur de données dans un mappage.

Si la transformation Processeur de données finalise l'exécution sans échecs, elle n'enregistre pas de journal des événements. S'il y a des échecs lors de l'exécution, la transformation Processeur de données s'exécute une deuxième fois et écrit un journal des événements pendant la deuxième exécution. Le journal des événements au moment de l'exécution est appelé `events.cme`.

Sur une machine Windows, le journal des événements au moment de l'exécution se trouve dans le répertoire suivant :

```
C:<Installation_Directory>\clients\DT\CMReports\Tmp\
```

Sur une machine Linux ou UNIX, le journal des événements au moment de l'exécution se trouve dans le répertoire suivant pour un utilisateur root :

```
/root/<Installation_Directory>/clients/DT/CMReports/Tmp
```

Sur une machine Linux ou UNIX, le journal des événements au moment de l'exécution se trouve dans le répertoire suivant pour un utilisateur non root :

```
/home/[UserName]/<Installation_Directory>/DT/CMReports/Tmp
```

Utilisez l'éditeur de configuration pour changer l'emplacement du journal des événements au moment de l'exécution.

Affichage d'un journal des événements dans la vue des événements du processeur de données

Utilisez la vue **Événements du processeur de données** pour afficher un journal des événements de l'environnement de développement ou un journal des événements d'exécution.

Ouvrez l'explorateur Windows, puis accédez au fichier journal des événements que vous souhaitez afficher. Faites glisser le journal depuis l'explorateur Windows vers la vue **Événements du processeur de données**. Cliquez avec le bouton droit de la souris dans la vue **Événements du processeur de données**, puis sélectionnez **Rechercher** pour rechercher le journal.

Remarque: Pour recharger le journal des événements de l'environnement de développement le plus récent, cliquez sur la vue **Événements du processeur de données** avec le bouton droit de la souris, puis sélectionnez **Recharger les événements du projet**.

Journal utilisateur

Le journal utilisateur contient les messages personnalisés que vous configurez sur les échecs de composants dans un script.

La transformation Processeur de données écrit des messages dans le journal utilisateur lorsque vous exécutez un script dans la vue **Visionneuse de données** et lorsque vous l'exécutez dans un mappage.

Lorsqu'un composant de script comprend la propriété **on_fail**, vous pouvez la configurer pour écrire un message dans le journal utilisateur lorsqu'il échoue. Dans le script, définissez la propriété **on_fail** sur l'une des valeurs suivantes :

- LogInfo
- LogWarning
- LogError

Chaque exécution de script produit un nouveau journal utilisateur. Le nom de fichier du journal utilisateur contient le nom de la transformation avec un GUID unique :

```
<Transformation_Name>_<GUID>.log
```

Par exemple, CalculateValue_Aa93a9d14-a01f-442a-b9cb-c9ba5541b538.log

Sur une machine Windows, vous trouverez le journal utilisateur dans le répertoire suivant :

```
c:\Users\[UserName]\AppData\Roaming\Informatica\DataTransformation\UserLogs
```

Sur une machine Linux ou UNIX, vous trouverez le journal utilisateur destiné à l'utilisateur racine dans le répertoire suivant :

```
/<Installation_Directory>/DataTransformation/UserLogs
```

Sur une machine Linux ou UNIX, vous trouverez le journal utilisateur destiné à un utilisateur non racine dans le répertoire suivant :

```
home/<Installation_Directory>/DataTransformation/UserLogs
```

Développement de la transformation Processeur de données

Utilisez l'assistant Nouvelle transformation pour générer automatiquement une transformation Processeur de données ou créez une transformation Processeur de données vide et configurez-la ultérieurement. Si vous créez une transformation Processeur de données vide, vous devez y créer un objet Script, XMap, Bibliothèque ou Règles de validation. Un script peut analyser des documents source en format hiérarchique, convertir un format hiérarchique en d'autres formats de fichiers ou mapper un document hiérarchique sur un autre format hiérarchique. Une XMap convertit un fichier d'entrée hiérarchique en fichier de sortie hiérarchique à la structure différente. Une bibliothèque convertit un type de messagerie standard en document XML avec une structure de hiérarchie ou un fichier XML en un format standard. Choisissez les schémas qui définissent les hiérarchies d'entrée ou de sortie.

1. Créez la transformation dans l'outil Developer.
2. S'il s'agit d'une transformation Processeur de données vide, effectuez les étapes supplémentaires suivantes :
 - a. Ajouter les références de schéma qui définissent les hiérarchies XML d'entrée ou de sortie.
 - b. Créez un objet Script, XMap, Bibliothèque ou Règles de validation.
3. Configurer les ports d'entrée et de sortie.
4. Tester la transformation.

Créer la transformation Processeur de données

Créez une transformation Processeur de données dans l'outil Developer. Si vous créez une transformation Processeur de données vide, vous devez y créer un objet Script, XMap, Bibliothèque ou Règles de validation. Vous pouvez également utiliser l'assistant Nouvelle transformation pour générer automatiquement une transformation Processeur de données.

1. Dans l'outil Developer, cliquez sur **Fichier > Nouveau > Transformation**.
2. Sélectionnez la transformation Processeur de données et cliquez sur **Suivant**.
3. Entrez un nom pour la transformation et sélectionnez un emplacement du référentiel modèle pour y stocker la transformation.
4. Créez la transformation Processeur de données à l'aide d'un assistant ou créez une transformation Processeur de données vide.
5. Si vous choisissez de créer une transformation Processeur de données vide, cliquez sur **Terminer**.
L'outil Developer crée la transformation vide dans le référentiel. La vue **Présentation** s'affiche dans l'outil Developer.
6. Si vous choisissez de créer une transformation Processeur de données à l'aide d'un assistant, effectuez les étapes suivantes :
 - a. Cliquez sur **Suivant**.

- b. Sélectionnez un format d'entrée.
- c. Si besoin, pour certains formats d'entrées tels que COBOL, JSON ou ASN.1, recherchez et sélectionnez un schéma, un copybook, un fichier exemple ou un fichier de spécification.
- d. Sélectionnez un format de sortie.
- e. Si besoin, recherchez et sélectionnez un schéma, un copybook, un fichier exemple ou un fichier de spécification pour le format de sortie.
- f. Cliquez sur **Terminer**. L'assistant crée la transformation dans le référentiel.

La transformation peut inclure un analyseur, un sérialiseur, un mappeur ou un objet avec des composants communs. Si vous avez sélectionné un schéma, un copybook, un fichier exemple ou un fichier de spécification, l'assistant crée également dans le référentiel un schéma équivalent à la hiérarchie du fichier.

Sélectionner les objets de schéma

Choisissez les objets de schéma qui définissent les hiérarchies d'entrée ou de sortie de chaque composant XMap ou Script que vous voulez créer.

Vous pouvez ajouter des références de schéma à la vue Références, ou vous pouvez les ajouter lorsque vous créez les objets de type XMap ou script. Un objet de schéma doit exister dans le référentiel modèle avant de pouvoir être référencé dans un script ou XMap.

1. Dans la vue **Références** de la transformation Processeur de données, cliquez sur **Ajouter**.
2. Si l'objet de schéma existe dans le référentiel modèle, repérez et sélectionnez le schéma.
3. Si le schéma n'existe pas dans le référentiel modèle, cliquez sur **Créer un nouvel objet de schéma** et importez un objet de schéma à partir d'un fichier de schéma hiérarchique.
4. Cliquez sur Terminer pour ajouter la référence de schéma à la transformation Processeur de données.

Créer des objets dans une transformation Processeur de données vide

Créez un objet Script, Bibliothèque, XMap ou Règles de validation dans la vue **Objets** de la transformation Processeur de données. Après avoir créé l'objet, vous pouvez l'ouvrir à partir de la vue **Objets** afin de le configurer.

Création d'un script

Créez un objet Script et définissez le type de composant Script à créer. Éventuellement, vous pouvez définir une référence de schéma et un fichier de source d'exemple.

1. Dans la vue **Objets** de la transformation Processeur de données, cliquez sur **Nouveau**.
2. Entrez le nom du script et cliquez sur **Suivant**.
3. Choisissez de créer un analyseur ou un sérialiseur. Sélectionnez Autre pour créer un composant Mappeur, Transformateur ou Répartiteur.
4. Entrez le nom du composant.
5. Si le composant est le premier à traiter les données dans la transformation, activez **Définir en tant que composant de démarrage**.
6. Cliquez sur **Suivant** si vous voulez entrer une référence de schéma pour ce script. Cliquez sur **Terminer** si vous ne voulez pas entrer la référence de schéma.

7. Si vous choisissez de créer une référence de schéma, sélectionnez **Ajouter une référence à un objet de schéma** et recherchez l'objet de schéma dans le référentiel modèle. Cliquez sur **Créer un objet de schéma** pour créer l'objet de schéma dans le référentiel modèle.
8. Cliquez sur **Suivant** pour entrer une référence de source d'exemple ou pour entrer un texte d'exemple. Cliquez sur **Terminer** si vous ne voulez pas définir une source d'exemple.
Utilisez une source d'exemple pour définir des données d'exemple et pour tester le script.
9. Si vous choisissez de sélectionner une source d'exemple, sélectionnez **Fichier** et recherchez le fichier d'exemple.
Vous pouvez également saisir un texte d'exemple dans la zone **Texte**. L'outil Developer utilise le texte pour tester un script.
10. Cliquez sur **Terminer**.
La vue **Script** s'affiche dans l'éditeur de l'outil Developer.

Création d'une XMap

Créez une XMap sur la vue **Objets** de Data Transformation. Lorsque vous créez une XMap, vous devez disposer d'un schéma décrivant les documents hiérarchiques d'entrée et de sortie. Vous sélectionnez l'élément du schéma qui est l'élément racine pour la hiérarchie d'entrée.

1. Dans la vue **Objets** de la transformation Processeur de données, cliquez sur **Nouveau**.
2. Sélectionnez XMap et cliquez sur **Suivant**.
3. Entrez un nom pour la XMap.
4. Si le composant XMap est le premier composant à traiter des données dans la transformation, activez **Définir comme composant de démarrage**.
Cliquez sur **Suivant**.
5. Si vous choisissez de créer une référence de schéma, sélectionnez **Ajouter la référence à un objet de schéma** et accédez à l'objet de schéma dans le référentiel modèle.
Pour importer un nouvel objet de schéma, cliquez sur **Créer un nouvel objet de schéma**.
6. Si vous disposez d'un exemple de fichier hiérarchique que vous pouvez utiliser pour tester l'objet XMap, recherchez-le dans le système de fichiers et sélectionnez-le.
L'exemple de fichier hiérarchique peut être modifié.
7. Sélectionnez la racine pour la hiérarchie d'entrée.
Dans la boîte de dialogue **Sélection d'élément racine**, sélectionnez l'élément du schéma qui est l'élément racine pour le fichier hiérarchique d'entrée. Vous pouvez rechercher un élément dans le schéma. Vous pouvez utiliser la recherche de filtre. Entrez `*<string>` pour faire correspondre tout nombre de caractères dans la chaîne. Entrez `?<character>` pour faire correspondre un seul caractère.
8. Cliquez sur **Terminer**.
L'outil Developer crée une vue pour chaque XMap que vous créez. Cliquez sur la vue pour configurer le mappage.

Création d'une bibliothèque

Créez un objet de bibliothèque dans la vue **Objets** de Data Transformation. Sélectionnez le type, le composant et le nom du message. Éventuellement, vous pouvez définir un fichier exemple source de type de message que vous pouvez utiliser pour tester l'objet de bibliothèque.

Avant de créer une bibliothèque dans la transformation Processeur de données, installez le package logiciel de la bibliothèque sur votre ordinateur.

1. Dans la vue **Objets** de la transformation Processeur de données, cliquez sur **Nouveau**.
2. Sélectionnez la bibliothèque et cliquez sur **Suivant**.
3. Naviguez pour sélectionner le type de message.
4. Choisissez de créer un analyseur ou un sérialiseur.
Créez un analyseur si l'entrée de l'objet de bibliothèque est de type message et la sortie de type XML.
Créez un sérialiseur si l'entrée de l'objet de bibliothèque est de type XML et la sortie de type message.
5. Si la bibliothèque est le premier composant à traiter les données dans la transformation Processeur de données, activez **Définir en tant que composant de démarrage**.
Cliquez sur **Suivant**.
6. Si vous avez un fichier exemple source de type de message que vous pouvez utiliser pour tester la bibliothèque, trouvez et sélectionnez le fichier dans le système de fichiers.
Vous pouvez changer le fichier d'exemple.
7. Cliquez sur **Terminer**.
L'outil Developer crée une vue pour chaque type de message que vous créez. Cliquez sur la vue pour accéder au mappage.

Création d'un objet Règles de validation

Créez un objet de Règles de validation dans la vue **Objets** de la transformation Processeur de données.

1. Dans la vue **Objets** de la transformation Processeur de données, cliquez sur **Nouveau**.
2. Sélectionnez l'objet Règles de validation et cliquez sur **Suivant**.
3. Entrez le nom de l'objet Règles de validation.
4. Si vous disposez d'un exemple de fichier XML que vous pouvez utiliser pour tester l'objet Règles de validation, recherchez-le dans le système de fichiers et sélectionnez-le.
Vous pouvez changer le fichier XML d'exemple.
5. Cliquez sur **Terminer**.
L'outil Developer crée un objet Règles de validation et s'ouvre dans l'éditeur Règles de validation.

Ajout d'une source d'exemple

Choisissez la source d'exemple pour tester les objets Script, XMap, Bibliothèque ou Règles de validation que vous voulez créer.

Vous pouvez ajouter une source d'exemple lorsque vous créez un objet Script, XMap, Bibliothèque ou Règles de validation. Une fois sélectionnée, la source d'exemple est ajoutée au référentiel modèle. En raison des limitations du référentiel modèle, la taille de fichier de la source d'exemple est limitée à 5 Mo.

La source d'exemple peut être modifiée.

Créer les ports

Configurez les ports d'entrée et de sortie dans la vue **Présentation**.

Lorsque vous configurez des ports d'entrée ou de sortie supplémentaires dans un script, l'outil Developer ajoute des ports d'entrée et de sortie supplémentaires à la transformation par défaut. Vous ne pouvez pas ajouter de ports d'entrée dans la vue **Présentation**.

1. Si vous voulez retourner des lignes de données de sortie plutôt que du XML, activez **Sortie relationnelle**.
Lorsque vous activez la sortie relationnelle, l'outil Developer retire le port de sortie par défaut.
2. Sélectionnez le type de données du port d'entrée, le type de port, la précision et l'échelle.
3. Si vous ne souhaitez pas définir de port de sortie relationnelle, définissez le type de données du port de sortie, le type de port, la précision et l'échelle.
4. Si un script comporte des ports d'entrée supplémentaires, vous pouvez définir l'emplacement du fichier d'entrée d'exemple pour les ports. Cliquez sur le bouton **Ouvrir** du champ **Emplacement de l'entrée** pour accéder au fichier.
5. Si vous avez activé la sortie relationnelle, cliquez sur **Sortie de mappage** pour créer les ports de sortie.
6. Dans la vue Ports, mappez les nœuds de la zone **Sortie hiérarchique** aux champs de la zone **Ports relationnels**.

Test de la transformation

Testez la transformation Processeur de données dans la vue **Visionneuse de données**.

Avant de tester la transformation, vérifiez que vous avez défini le composant de démarrage. Vous pouvez définir le composant de démarrage dans un script ou le sélectionner sous l'onglet **Présentation**. Vous devez également avoir choisi un exemple de fichier d'entrée pour effectuer le test.

1. Ouvrez la vue **Visionneuse de données**.
2. Cliquez sur **Exécuter**.
L'outil Developer valide la transformation. Si aucune erreur n'est détectée, l'outil Developer affiche l'exemple de fichier dans la zone **Entrée**. Les résultats de sortie s'affichent dans le panneau Sortie.
3. Cliquez sur **Afficher les événements** pour afficher la vue **Événements du processeur de données**.
4. Double-cliquez sur un événement dans la vue **Événements du processeur de données** pour effectuer le débogage de l'événement dans l'éditeur de script.
5. Cliquez sur **Synchroniser avec l'éditeur** pour modifier le fichier d'entrée lorsque vous testez plusieurs composants, chacun associé à un exemple de fichier d'entrée différent.
Si vous modifiez le contenu de l'exemple de fichier dans le système de fichiers, les modifications s'affichent dans la zone **Entrée**.

Exportation et importation de la transformation du processeur de données

Vous pouvez exporter la transformation Processeur de données en tant que service et l'exécuter depuis un référentiel Data Transformation. Vous pouvez également importer un service Data Transformation à l'outil

Developer. Lorsque vous importez un service Data Transformation, l'outil Developer crée une transformation Processeur de données depuis le service.

Remarque: Lorsque vous importez un service Data Transformation dans le référentiel modèle, l'outil Developer importe les schémas associés au référentiel. Si vous modifiez le schéma dans le référentiel, les modifications ne s'affichent parfois pas dans les références du schéma de la transformation. Vous pouvez fermer et ouvrir la connexion au référentiel modèle ou bien fermer et ouvrir l'outil Developer tool pour afficher les modifications du schéma dans la transformation.

Exportation de la transformation Processeur de données en tant que service

Vous pouvez exporter la transformation Processeur de données en tant que service Data Transformation. Exportez le service dans le référentiel du système de fichiers de l'ordinateur où vous souhaitez exécuter le service. Vous pouvez exécuter le service avec PowerCenter, des applications définies par l'utilisateur ou la commande CM_console de Data Transformation.

1. Dans la vue **Explorateur d'objets**, cliquez avec le bouton droit sur la transformation Processeur de données que vous voulez exporter et sélectionnez **Exporter**.
La boîte de dialogue **Exporter** s'affiche.
2. Sélectionnez **Informatica > Exporter la transformation Processeur de données** et cliquez sur **Suivant**.
La page **Sélectionner** s'affiche.
3. Cliquez sur **Suivant**.
La page **Sélectionner le nom de service et le dossier de destination** s'affiche.
4. Choisissez un dossier de destination :
 - Pour exporter le service sur la machine qui héberge l'outil Developer tool, cliquez sur **Dossier du service**.
 - Pour déployer le service sur une autre machine, cliquez sur **Dossier**. Parcourez le répertoire \ServiceDB sur la machine où vous voulez déployer le service.
5. Cliquez sur **Terminer**.

Importation de plusieurs services Data Transformation

Vous pouvez importer un répertoire de services Data Transformation à partir de l'emplacement d'enregistrement du répertoire. Lorsque vous importez des services Data Transformation dans le référentiel modèle Developer, l'outil Developer tool importe les transformations, schémas et exemples de données avec les fichiers .cmw. Si vous devez importer plusieurs services, importez un répertoire de services plutôt qu'un service à la fois.

1. Cliquez sur **Fichier > Importer**.
La boîte de dialogue **Importer** s'affiche.
2. Sélectionnez **InformaticaImporter les services Data Transformation (dossier)** et cliquez sur **Suivant**.
La page **Importer le service Data Transformation** s'affiche.
3. Accédez au répertoire à importer.
4. Accédez à un emplacement dans le référentiel dans lequel vous souhaitez enregistrer les transformations, puis cliquez sur **Terminer**.
L'outil Developer tool importe les transformations, les schémas et les exemples de données avec le fichier .cmw.

Importation d'un service Data Transformation

Vous pouvez importer un fichier .cmw de service Data Transformation dans le référentiel modèle pour créer une transformation Processeur de données. L'outil Developer importe la transformation, les schémas et les données d'exemple avec le fichier .cmw.

1. Cliquez sur **Fichier > Importer**.
La boîte de dialogue **Importer** s'affiche.
2. Sélectionnez **Informatica Importer le service Data Transformation (unique)** et cliquez sur **Suivant**.
La page **Importer le service Data Transformation** s'affiche.
3. Accédez au fichier .cmw du service que vous voulez importer.
L'outil Developer nomme la transformation d'après le nom du fichier de service. Vous pouvez modifier le nom.
4. Accédez à un emplacement dans le référentiel dans lequel vous souhaitez enregistrer la transformation, puis cliquez sur **Terminer**.
L'outil Developer importe la transformation, les schémas et les données d'exemple avec le fichier .cmw.
5. Pour modifier la transformation, double-cliquez dessus dans la vue **Explorateur d'objets**.

Exportation d'un mappage avec une transformation Processeur de données vers PowerCenter

Lorsque vous exportez un mappage avec une transformation Processeur de données vers PowerCenter, vous pouvez exporter les objets vers un fichier local, puis importer le mappage dans PowerCenter. Éventuellement, vous pouvez exporter le mappage directement dans le référentiel PowerCenter.

1. Dans la vue **Explorateur d'objets**, sélectionnez le mappage à exporter. Cliquez dessus avec le bouton droit de la souris et sélectionnez **Exporter**.
La boîte de dialogue **Exporter** s'affiche.
2. Sélectionnez **Informatica > PowerCenter**.
3. Cliquez sur **Suivant**.
La boîte de dialogue **Exporter vers PowerCenter** s'ouvre.
4. Sélectionnez le projet.
5. Sélectionnez la version de PowerCenter.
6. Choisissez l'emplacement de l'exportation, à savoir un fichier d'importation XML PowerCenter ou un référentiel PowerCenter.
7. Indiquez les options d'exportation.
8. Cliquez sur **Suivant**.
L'outil Developer vous invite à sélectionner les objets à exporter.
9. Sélectionnez les objets à exporter et cliquez sur **Terminer**.
L'outil Developer exporte les objets vers l'emplacement sélectionné. Si vous avez exporté le mappage vers un emplacement, l'outil Developer exporte également les transformations Processeur de données du mappage, en tant que services, vers un dossier à l'emplacement que vous avez spécifié.
10. Si vous avez exporté le mappage vers un référentiel PowerCenter, les services sont exportés vers le chemin de répertoire suivant : %temp%\DTServiceExport2PC\

La fonction d'exportation crée un dossier distinct pour chaque service avec le nom suivant :

`<date><serviceFullName>`

Si la transformation inclut le mappage relationnel, un dossier pour le mappage relationnel à hiérarchique et un autre pour le mappage hiérarchique à relationnel sont créés.

11. Copiez le ou les dossiers avec les services de la transformation Processeur de données depuis l'emplacement local où vous avez exporté les fichiers vers le dossier ServiceDB de PowerCenter.
12. Si vous avez exporté le mappage vers un fichier d'importation XML PowerCenter, importez le mappage dans PowerCenter. Pour plus d'informations sur l'importation d'un objet dans PowerCenter, consultez le *Guide du référentiel PowerCenter 9.6.0*.

Transformation Processeur de données Validation

Après avoir exporté une transformation Processeur de données en tant que service, vous pouvez exécuter des validations VRL sur le service à partir du référentiel Data Transformation.

Vous pouvez utiliser un moteur Data Transformation à vitesse optimisée pour les validations VRL. Le moteur Data Transformation à vitesse optimisée prend en charge les fonctions VRL suivantes :

- `dt:exist`
- `dt:empty`
- `dt:date-valid`
- `dt:next-sequence`
- `dt:all-equal`
- `dt:lookup`
- `dt:regex-match`

Le moteur Data Transformation à vitesse optimisée génère la sortie **ValidateValue**. **ValidateValue** contient la propriété `max_error_count`, avec une valeur par défaut de 200 erreurs. Lorsque le nombre d'erreurs dépasse le `max_error_count`, la validation s'arrête.

Remarque: La syntaxe VRL du moteur Data Transformation à vitesse optimisée ne prend pas en charge la balise `<list>`.

Utilisation d'un moteur Data Transformation à vitesse optimisée pour les validations VRL.

Après avoir exporté un service Data Transformation, vous pouvez utiliser un moteur Data Transformation à vitesse optimisée pour les validations VRL avec le service.

- Définissez l'indicateur suivant dans le fichier `.cmw` du service `optimiser_vrl`.

Ajoutez l'indicateur `optimize_vrl` à l'instance `ServiceConfigProf`, comme indiqué dans l'exemple suivant :

```
instance ServiceConfig = ServiceConfigProf<add_required_xml_elements,  
add_required_xml_attributes, optimize_vrl>
```

Transformation Processeur de données dans un environnement non natif

Le traitement de la transformation Processeur de données dans un environnement non natif dépend du moteur qui exécute la transformation.

Tenez compte de la prise en charge des moteurs d'exécution non natifs suivants :

- Moteur Blaze. Pris en charge sans restrictions.
- Moteur Spark. Pris en charge avec des restrictions dans les mappages de lots. Non pris en charge dans les mappages de streaming.*
- Moteur Databricks Spark. Non pris en charge.

** Pour plus d'informations sur la prise en charge de la transformation Processeur de données sur le moteur Spark, consultez l'[KB article](#).*

CHAPITRE 3

Formats d'entrée et de sortie de l'assistant

Ce chapitre comprend les rubriques suivantes :

- [Présentation des formats d'entrée et de sortie et l'assistant, 49](#)
- [Avro, 50](#)
- [Bibliothèque de traitement COBOL, 54](#)
- [JSON, 58](#)
- [Parquet, 61](#)
- [XML, 63](#)

Présentation des formats d'entrée et de sortie et l'assistant

Vous pouvez utiliser un assistant pour créer une transformation Processeur de données générée automatiquement avec des formats d'entrée et de sortie tels que COBOL, XML, JSON ou des formats relationnels. Vous pouvez également utiliser l'assistant pour transformer les formats définis par l'utilisateur.

Créez une transformation Processeur de données et sélectionnez les formats d'entrée et de sortie via l'assistant de la transformation Processeur de données. Faites votre choix parmi les formats existants ou créez des formats définis par l'utilisateur. Pour certains formats tels que XML, JSON ou COBOL, ajoutez un schéma, un fichier de spécification, un fichier d'exemple ou un copybook qui définit la structure attendue pour l'entrée ou la sortie.

L'assistant crée une transformation avec les objets Script, XMap ou Bibliothèque servant de modèles pour la transformation du format d'entrée en format de sortie. La transformation Processeur de données crée une solution de transformation en fonction des formats sélectionnés et du fichier de spécification, du fichier exemple ou du copybook. La transformation peut ne pas être complète, mais elle contient les composants que vous connectez et personnalisez pour effectuer la définition de la transformation.

Avro

Utilisez l'assistant pour créer une transformation avec une entrée ou une sortie Avro. Lorsque vous créez une transformation Processeur de données pour transformer le format Avro, vous sélectionnez un schéma ou un fichier exemple Avro qui définit la structure attendue des données Avro. L'assistant crée des composants qui transforment le format Avro en d'autres formats ou d'autres formats en format Avro. Ces composants peuvent inclure un mappage relationnel à hiérarchique, un mappage hiérarchique à relationnel et une XMap. Une fois que l'assistant a créé la transformation, vous pouvez configurer davantage la transformation afin de déterminer la logique de mappage.

Apache Avro est un système de sérialisation de données au format binaire ou dans d'autres formats de données. Le format des données Avro peut ne pas être contrôlable de visu. Pour plus d'informations sur Avro, voir <http://avro.apache.org/>

Remarque: Utilisez des données binaires codées Avro pour créer une transformation avec une entrée ou une sortie Avro. Toute entrée ou sortie Avro dans un autre format ne peut pas être traitée.

Une transformation qui lit une entrée ou une sortie Avro repose sur un schéma. Lorsque la transformation lit ou écrit des données Avro, la transformation utilise ce schéma pour interpréter la hiérarchie.

Lorsque vous sélectionnez un fichier exemple pour définir la hiérarchie Avro, l'assistant enregistre également le premier enregistrement du fichier en tant que fichier de test distinct. Vous pouvez utiliser ce fichier pour tester la transformation. Pour rechercher le fichier, dans le panneau **Ports** de la vue **Présentation**, consultez le chemin de fichier indiqué dans le champ **Emplacement de l'entrée**.

Lorsque vous créez une transformation Processeur de données qui transforme un format Avro en format hiérarchique, ou un format hiérarchique en format Avro, l'assistant crée un composant XMap dans la transformation. L'éditeur XMap affiche les nœuds du schéma hiérarchique et ceux du schéma Avro. Utilisez l'éditeur XMap pour lier les nœuds et définir la logique de transformation. Pour plus d'informations sur l'objet et l'éditeur XMap, voir ["Présentation de XMap" à la page 84](#).

Lorsque vous créez une transformation qui transforme un format Avro en format relationnel, ou un format relationnel ou en format Avro, l'assistant crée un mappage relationnel. Le panneau **Ports** de la vue **Présentation** affiche les nœuds du schéma hiérarchique Avro et les ports relationnels. Utilisez le panneau **Ports** pour lier les éléments hiérarchiques aux ports et aux groupes relationnels. Pour plus d'informations sur la transformation de données relationnelles, voir ["Présentation des entrées et sorties relationnelles" à la page 65](#).

Après avoir créé une transformation Processeur de données pour une entrée Avro, vous l'ajoutez à un mappage avec un lecteur de fichier complexe. Le lecteur de fichier complexe transmet l'entrée Avro à la transformation. Pour une transformation Processeur de données avec une sortie Avro, vous ajoutez un rédacteur de fichier complexe au mappage pour recevoir la sortie depuis la transformation.

Entrée Avro et lecteur de fichier complexe

Pour qu'une transformation Processeur de données puisse transformer une entrée Avro, la transformation doit recevoir des données d'entrée depuis un objet de lecteur de fichier complexe. Lorsque vous avez créé et configuré la transformation, vous l'ajoutez à un mappage et connectez le port d'entrée au port de sortie du lecteur de fichier complexe.

Le lecteur de fichier complexe fournit une entrée à un composant répartiteur que l'assistant de la transformation Processeur de données crée dans le cadre de la transformation. Si la transformation transforme l'entrée Avro en un format personnalisé ou relationnel, il n'est pas nécessaire de modifier les paramètres du répartiteur. Vous pouvez spécifier un format de sortie personnalisé en sélectionnant **Autre** en tant que format de sortie dans l'assistant.

Si la transformation transforme l'entrée Avro en format JSON, XML ou Avro, l'assistant crée une XMap pour mapper le format de sortie. Vous devez identifier la XMap auprès du répartiteur, de sorte que la transformation traite les données correctement.

Vous devez également configurer le lecteur de fichier complexe pour traiter une entrée Avro. Le port de sortie du lecteur de fichier complexe doit être défini au format binaire. De même, le port d'entrée de la transformation Processeur de données doit être défini sur le type d'entrée binaire.

Compression de données Avro avec le codec Snappy

Vous pouvez compresser des données Avro avec le lecteur de fichier complexe. Si vous utilisez le codec Snappy pour la compression de données Avro, vous devez mettre à jour le fichier .jar du codec Snappy avant de tester ou d'exécuter la transformation.

Pour utiliser le codec Snappy, remplacez le fichier .jar par défaut par sa version mise à jour dans le répertoire d'installation du serveur Informatica et dans l'environnement Hadoop. La version mise à jour du fichier `snappy-java-1.0.4.1.jar` est disponible en suivant ce lien :

<http://mvnrepository.com/artifact/org.xerial.snappy/snappy-java/1.1.0.1>

Mise à jour du codec Snappy pour activer la compression de données Avro

Pour activer la compression de données Avro à l'aide du codec Snappy, remplacez le fichier .jar par défaut du codec Snappy par sa version mise à jour.

1. Sur la machine sur laquelle vous avez installé le serveur Informatica, dans le répertoire d'installation du serveur, remplacez le fichier `snappy-java-1.0.4.1.jar` par le fichier `snappy-java-1.1.0.1.jar`. Remplacez le fichier .jar qui se trouve au chemin suivant : `<Répertoire_Installation_Serveur>\services\shared\hadoop\<Distribution_Hadoop>\lib`
2. Sur les machines sur lesquelles vous avez installé et exécutez Hadoop, remplacez le fichier `snappy-java-1.0.4.1.jar` par le fichier `snappy-java-1.1.0.1.jar`. Remplacez le fichier .jar qui se trouve au chemin suivant : `<Hadoop_rpm>\services\shared\hadoop\<Distribution_Hadoop>\lib`

Configurer une transformation avec une entrée Avro

Pour créer une transformation Processeur de données pour une entrée Avro, utilisez l'assistant de transformation Processeur de données. L'assistant crée la transformation dans le référentiel modèle avec les composants dont vous avez besoin pour transformer l'entrée Avro. Utilisez l'éditeur IntelliScript pour modifier le répartiteur, ainsi que l'éditeur XMap pour éditer une XMap, si celle-ci est incluse dans la transformation. Ajoutez la transformation à un mappage avec un lecteur de fichier complexe.

1. Créez la transformation Processeur de données à l'aide de l'assistant Nouvelle transformation. Ajoutez un schéma ou un fichier exemple Avro qui définit la structure d'entrée attendue.
2. Si la sortie de la transformation est au format XML, JSON ou un autre format structuré, utilisez l'éditeur XMap pour éditer la XMap dans la transformation.
3. Utilisez l'éditeur IntelliScript pour éditer et personnaliser le répartiteur dans la transformation. Si la sortie de la transformation est au format XML, JSON ou un autre format structuré, éditez le répartiteur pour identifier la XMap dans la transformation.
4. Ajoutez la transformation Processeur de données à un mappage avec un lecteur de fichier complexe. La transformation doit rester définie sur une entrée binaire, ce qui est le paramètre par défaut pour une entrée Avro. Configurez le lecteur de fichier complexe pour traiter des données Avro. Le paramètre de sortie reste défini sur binaire, ce qui est le paramètre par défaut. Liez le port de sortie du lecteur de fichier complexe au port d'entrée de la transformation Processeur de données pour fournir l'entrée Avro à la transformation.

Étape 1. Créer une transformation qui transforme des données Avro

Créez une transformation Processeur de données avec une entrée Avro, une sortie Avro ou les deux.

1. Dans l'outil Developer, cliquez sur **Fichier > Nouveau > Transformation**.
2. Sélectionnez la transformation Processeur de données et cliquez sur **Suivant**.
3. Entrez un nom pour la transformation et sélectionnez un emplacement du référentiel modèle pour y stocker la transformation.
4. Sélectionnez **Créer un processeur de données à l'aide d'un assistant**, puis cliquez sur **Suivant**.
5. Sélectionnez le format Avro ou un autre format d'entrée et cliquez sur **Suivant**.
6. Si vous avez sélectionné Avro en tant que format d'entrée, accédez à un fichier de schéma .xsd ou à un fichier exemple Avro et sélectionnez-le. Cliquez sur **Suivant**.

Developer ajoute un fichier de schéma .xsd représentant la hiérarchie Avro au référentiel modèle. Si vous sélectionnez un fichier exemple, Developer crée un fichier de test basé sur le premier enregistrement du fichier exemple. Vous pouvez utiliser ce fichier pour tester la transformation. Pour rechercher le fichier, dans le panneau **Ports** de la vue **Présentation**, consultez le chemin de fichier indiqué dans le champ **Emplacement de l'entrée**.

7. Sélectionnez le format Avro ou un autre format de sortie et cliquez sur **Suivant**.
8. Si vous sélectionnez Avro en tant que format de sortie, accédez à un schéma ou à un fichier exemple Avro associé et sélectionnez-le. Cliquez sur **Suivant**.
9. Cliquez sur **Terminer**.
L'outil Developer crée la transformation dans le référentiel avec les composants pertinents, tels qu'une XMap pour transformer la hiérarchie Avro en hiérarchie d'un autre format. La vue **Présentation** s'affiche dans l'outil Developer.
10. Pour éditer les composants de la transformation, dans la vue **Objets**, double-cliquez sur le composant de transformation afin de l'ouvrir dans l'éditeur adéquat.

Étape 2. Modifier la XMap

Pour configurer un objet XMap de transformation Processeur de données, ajoutez des instructions de mappage.

1. Pour ouvrir l'éditeur XMap, dans la transformation Processeur de données, sous **Objets**, cliquez sur l'objet XMap.
2. Pour créer une instruction de mappage Mappage, Groupe ou Groupe de répétition, dans l'éditeur XMap, effectuez un glisser-déplacer depuis un nœud du schéma hiérarchique d'entrée vers un nœud du schéma hiérarchique de sortie.
L'éditeur XMap crée un lien de mappage entre les nœuds. L'instruction de mappage s'affiche dans la grille. L'éditeur XMap remplit automatiquement les champs d'instruction de mappage.
3. Pour créer la logique conditionnelle dans la grille, ajoutez une instruction de mappage Routeur comme suit :
 - a. Dans l'instruction de mappage Routeur, créez des instructions de mappage Option. Déplacez et faites glisser les nœuds de schémas d'entrée et de sortie dans les champs d'instruction Option de la grille.
 - b. Dans l'instruction de mappage Routeur, créez une instruction de mappage par défaut pour spécifier ce qu'il se passe si aucune instruction de mappage Option ne s'applique.
 - c. Dans les instructions de mappage Option, créez des instructions de mappage Mappage pour spécifier les conditions pour mapper le nœud d'entrée au nœud de sortie.

4. Pour fournir un contexte standard pour un groupe d'instructions, ajoutez une instruction de mappage Groupe. Imbriquer les instructions de mappage Mappage dans l'instruction de mappage Groupe.
5. Pour appeler un autre objet XMap, ajoutez une instruction Exécuter XMap.
6. Pour modifier le contexte et la logique pour une instruction de mappage, éditez les propriétés de l'instruction de mappage comme suit :
 - a. Rétrograder les instructions à des instructions enfant ou avancer les instructions en instructions parent.
 - b. Créer des expressions XPath pour changer le contexte ou ajouter des prédicats à l'aide de l'éditeur XPath.

Étape 3. Configurer le répartiteur

Si le format sélectionné pour la sortie de la transformation est XML, JSON ou un autre format structuré, l'assistant crée un composant répartiteur et un objet XMap dans l'assistant. Modifiez le répartiteur pour identifier la XMap dans la transformation.

1. Dans la vue **Objets**, double-cliquez sur le répartiteur, un objet Script, pour l'ouvrir dans l'éditeur IntelliScript.
2. Pour configurer le répartiteur afin d'identifier la XMap dans la transformation, procédez comme suit :
 - a. Pour configurer l'élément nécessaire, développez l'élément **contains** dans l'éditeur IntelliScript. Double-cliquez sur la flèche double pointant vers la droite >> située en regard de l'élément.
 - b. Développez l'élément **repeating_segment**. Double-cliquez sur la flèche double pointant vers la droite >> située en regard de l'élément.
 - c. Dans l'élément **run_component**, sélectionnez l'objet XMap adéquat dans la liste des composants.
3. Pour configurer le répartiteur afin qu'il lise des données depuis des emplacements du document source et écrive des données en XML, cliquez avec le bouton droit de la souris sur
4. Pour configurer davantage le répartiteur, dans l'élément **Streamer**, imbriquez les composants **ComplexSegment** et **SimpleSegment** correspondant à la structure source.
5. Pour chaque **SimpleSegment**, définissez le marqueur d'ouverture et le marqueur de fermeture si nécessaire. Définissez la transformation qui traite le segment.

Étape 4. Configurer le lecteur de fichier complexe

Ajoutez une transformation Processeur de données qui transforme des données Avro en mappage avec un lecteur de fichier complexe. Configurez le lecteur de fichier complexe pour traiter une entrée Avro.

1. Dans l'éditeur de mappage, créez un objet de lecteur de fichier complexe.
2. Pour configurer le lecteur de fichier complexe, procédez comme suit :
 - a. Dans l'onglet **Avancé** de la vue **Propriétés**, sélectionnez la propriété **Format de fichier**, puis sélectionnez **Entrée personnalisée**.
 - b. Sélectionnez la propriété **Format d'entrée**, puis entrez `com.informatica.avro.AvroToXML`.
 - c. Pour optimiser les performances, utilisez la propriété **Paramètres du format d'entrée** pour régler le paramètre `MaxOutputAccumulation`. Par défaut, le paramètre `MaxOutputAccumulation` qui définit le nombre attendu d'enregistrements de sortie est défini sur 50 000. Pour modifier le paramètre sur 250 000, par exemple, entrez `"MaxOutputAccumulation"="250000"`.

- d. Par défaut, le lecteur de fichier complexe ajoute le schéma à la sortie du lecteur de fichier complexe dans un seul élément directement après l'élément racine. Si vous ne voulez pas ajouter le schéma à la sortie, sélectionnez la propriété **Paramètres du format d'entrée**, puis entrez
`"InjectSchema"="false"`.

Utilisez un point-virgule pour séparer plusieurs paramètres, par exemple

`"MaxOutputAccumulation"="250000"; "InjectSchema"="false"`.

3. Ajoutez la transformation Processeur de données au mappage. Le port d'entrée de la transformation doit rester défini sur une entrée binaire, ce qui est le paramètre par défaut pour une entrée Avro.
4. Liez le port de sortie du lecteur de fichier complexe au port d'entrée de la transformation Processeur de données. Le port de sortie du lecteur de fichier complexe doit rester défini sur une sortie binaire.

Configurer une transformation avec une sortie Avro

Pour créer une transformation Processeur de données pour une sortie Avro, utilisez l'assistant de transformation Processeur de données. L'assistant crée la transformation dans le référentiel modèle avec les composants dont vous avez besoin pour transformer la sortie Avro. Utilisez l'éditeur XMap pour éditer la XMap, si celle-ci est incluse. Ajoutez la transformation à un mappage avec un rédacteur de fichier complexe.

1. Créez la transformation Processeur de données à l'aide de l'assistant Nouvelle transformation. Ajoutez un schéma ou un fichier exemple Avro qui définit la structure de sortie attendue.
2. Si l'entrée de la transformation est au format XML, JSON ou un autre format structuré, utilisez l'éditeur XMap pour éditer et personnaliser la XMap créée par l'assistant dans la transformation.
3. Pour configurer les paramètres de sortie de la transformation Processeur de données, dans le panneau **Contrôle de sortie** de la vue **Paramètres**, sélectionnez l'option de sortie adéquate dans la zone **Mode de collecte des ports de sortie binaires**.

Remarque: Pour configurer la sortie afin d'envoyer un enregistrement à la fois avec le schéma, sélectionnez l'option **Sortir une ligne pour chaque ligne**. Pour collecter tous les enregistrements avec le schéma dans un seul flux de sortie, sélectionnez l'option **Collecter les lignes d'entrée vers une seule sortie**.

4. Ajoutez la transformation Processeur de données à un mappage. La transformation reste définie sur une sortie binaire, ce qui est le paramètre par défaut pour une sortie Avro.
5. Créez et liez un rédacteur de fichier complexe pour la transformation Processeur de données dans le mappage pour la réception de la sortie Avro depuis la transformation.

Remarque: Le paramètre d'entrée reste défini sur binaire, ce qui est le paramètre par défaut. Le rédacteur de fichier complexe ne prend pas en charge la compression pour la sortie Avro.

6. Configurez le rédacteur de fichier complexe. Dans l'opération d'objet de données, dans l'onglet **Avancé**, pour **Format de fichier**, sélectionnez **Sortie personnalisée**. Pour **Format de sortie**, entrez
`com.informatica.avro.XMLToAvro`.

Bibliothèque de traitement COBOL

La bibliothèque COBOL transforme des données COBOL en données XML et des données XML en données COBOL. Lorsque vous utilisez l'assistant pour créer une transformation Processeur de données avec entrée

ou sortie COBOL, vous devez sélectionner un copybook COBOL pour définir la structure attendue pour les données d'entrée ou de sortie.

Lorsque vous créez une transformation Processeur de données avec entrée ou sortie COBOL à l'aide de l'assistant, Developer ajoute les objets suivants à la transformation :

- Un objet de schéma qui définit une représentation XML de la structure de données COBOL.
- Pour l'entrée COBOL, un analyseur qui transforme les données d'entrée de la définition de données COBOL en données XML.
- Pour la sortie COBOL, un sérialiseur qui transforme les données XML en données COBOL.

Remarque: Vous pouvez créer une transformation Processeur de données qui utilise l'entrée ou la sortie COBOL, mais pas les deux. Pour traiter le COBOL codé en EBCDIC, veuillez à modifier les paramètres de codage de la transformation Processeur de données en EBCDIC.

Création d'une transformation pour COBOL

Utilisez l'assistant de la transformation Processeur de données pour créer une transformation Processeur de données avec entrée ou sortie COBOL.

1. Dans l'outil Developer tool, cliquez sur **Fichier > Nouveau > Transformation**.
2. Sélectionnez la transformation Processeur de données et cliquez sur **Suivant**.
3. Entrez un nom pour la transformation et sélectionnez un emplacement du référentiel modèle pour y stocker la transformation.
4. Sélectionnez **Créer un processeur de données à l'aide d'un assistant**, puis cliquez sur **Suivant**.
5. Sélectionnez un format d'entrée, puis cliquez sur **Suivant**.
6. Si vous sélectionnez COBOL comme format d'entrée, recherchez et sélectionnez un copybook COBOL. Cliquez sur **Suivant**.
L'extension du fichier de spécification du copybook est généralement *.txt. Developer ajoute un fichier de schéma XSD représentant le copybook au référentiel modèle.
7. Sélectionnez un format de sortie, puis cliquez sur **Suivant**.
8. Si vous sélectionnez COBOL comme format de sortie, recherchez et sélectionnez un copybook COBOL. Cliquez sur **Suivant**.
Si vous avez sélectionné COBOL comme format d'entrée, vous ne pouvez pas le sélectionner comme format de sortie.
9. Cliquez sur **Terminer**.
L'outil Developer tool crée la transformation dans le référentiel. La vue **Présentation** s'affiche dans l'outil Developer tool.
10. Dans la vue **Objets**, double-cliquez sur l'analyseur pour l'ouvrir dans l'éditeur IntelliScript.
11. Si les données COBOL sont codées en EBCDIC, dans la vue **Paramètres**, remplacez le codage d'entrée ou de sortie par la page de codes EBCDIC pertinente.

Définitions de données COBOL

Le copybook COBOL que vous utilisez pour créer une transformation Processeur de données peut contenir des définitions de données de n'importe quelle complexité. Le copybook et l'entrée COBOL doivent être conformes aux règles de définition de données décrites dans cette section.

Définitions de données prises en charge

L'importation COBOL prend en charge les définitions de données de n'importe quelle complexité. Par exemple, elles peuvent utiliser des types de données en décimal condensé (COMP-3), binaires (COMP-1, COMP-2 ou COMP-4) ou décimales logiques (99V99). Elles peuvent contenir des fonctionnalités telles que REDEFINES, OCCURS et OCCURS DEPENDING ON.

Règles de définition de données

Une définition de données COBOL doit respecter les règles suivantes :

- 72 caractères par ligne au maximum et pas de texte au-delà de la colonne 72.
- La première ligne doit être une remarque dont la colonne 7 contient un *, ou elle doit commencer par un numéro de niveau.
- Le numéro de premier niveau doit se trouver en colonne 1 ou 8.

Définitions de données non prises en charge

La transformation Processeur de données ne prend pas en charge les définitions de données COBOL suivantes :

- Les numéros de niveau particuliers 66, 77 et 88.
- Les clauses USAGE au niveau du groupe.
- Les clauses INDEXED BY.
- POINTER et PROCEDURE-POINTER.

Procédures de test

Lorsque vous testez l'analyseur COBOL, vous transformez l'exemple de données COBOL en données XML et vérifiez la sortie. Une fois l'analyseur testé, vous pouvez exécuter le sérialiseur COBOL sur la sortie de l'analyseur.

Test d'un analyseur COBOL

Pour tester l'analyseur COBOL, vous avez besoin d'un fichier d'entrée qui contient un exemple de données COBOL. La structure de données doit être conforme à la définition de données que vous avez importée. L'analyseur transforme l'exemple de données COBOL en données XML, puis vous vérifiez la sortie.

1. Dans la vue **Objet**, double-cliquez sur l'analyseur COBOL.
L'analyseur s'affiche dans le panneau de script de l'éditeur IntelliScript.
2. Cliquez avec le bouton droit de la souris sur le nom de l'analyseur, puis cliquez sur **Définir comme composant de démarrage**.
3. Développez l'arbre IntelliScript et modifiez la propriété `example_source` de l'analyseur. Remplacez sa valeur `Text` par `LocalFile`.
L'assistant configure l'analyseur COBOL de façon à ne pas requérir de document d'exemple de source. Une fois le test terminé, vous pouvez effacer ce dernier. Il n'a aucun effet sur la transformation au moment de l'exécution.
4. Pour attribuer un fichier exemple, développez le composant `LocalFile` en cliquant sur la double flèche droite **>>**. Double-cliquez sur la propriété `file_name` et recherchez le fichier d'entrée qui contient les données d'exemple COBOL.

5. Dans le panneau **Entrée** de la vue **Visionneuse de données**, vous pouvez examiner le fichier exemple. Si le document ne s'affiche pas, cliquez avec le bouton droit de la souris sur le nom de l'analyseur dans IntelliScript, puis cliquez sur **Ouvrir une source d'exemple**.
6. Cliquez sur **Exécuter > Exécuter la visionneuse de données** pour tester l'analyseur.
7. Dans le panneau **Sortie** de la vue **Visionneuse de données**, examinez la sortie de l'analyseur.
8. Pour vous assurer que l'analyseur a été exécuté sans erreur, dans le panneau **Sortie** de la vue **Visionneuse de données**, cliquez sur le bouton **Afficher les événements**. Examinez le journal d'exécution dans la vue **Événements du processeur de données**.

Test d'un sérialiseur COBOL

Une fois l'analyseur COBOL testé, vous pouvez exécuter le sérialiseur COBOL sur la sortie de l'analyseur.

1. Dans l'explorateur de Data Transformation, double-cliquez sur le fichier de script TGP du sérialiseur. Le sérialiseur s'affiche dans le panneau de script de l'éditeur IntelliScript.
2. Cliquez avec le bouton droit de la souris sur le nom du sérialiseur, puis cliquez sur **Définir comme composant de démarrage**.
3. Cliquez sur **Exécuter > Exécuter** pour activer le sérialiseur. À l'invite, accédez au fichier `Results\output.xml` que vous avez généré lors de l'exécution de l'analyseur.
4. Examinez le journal d'exécution dans la vue **Événements**. Assurez-vous que le sérialiseur a été exécuté sans erreur.
5. Pour afficher la sortie du sérialiseur, double-cliquez sur le fichier `Results\output.xml` dans l'explorateur de Data Transformation.
Les données affichées doivent être les mêmes que celles de l'entrée d'origine sur laquelle vous avez exécuté l'analyseur.

Édition d'une transformation pour COBOL

Vous pouvez modifier une transformation pour COBOL que vous générez avec l'assistant de la transformation Processeur de données.

Si vous faites une modification de ce genre, documentez-la. La documentation peut être essentielle si vous désirez plus tard mettre à jour la définition de données COBOL ou la réimporter dans une nouvelle transformation, et que vous avez besoin de reproduire votre modification.

Optimisation du traitement des fichiers COBOL volumineux dans l'environnement Hadoop

Vous pouvez optimiser la manière dont un mappage composé d'un lecteur de fichier complexe et d'une transformation Processeur de données traite des fichiers COBOL volumineux dans l'environnement Hadoop.

Pour optimiser le traitement de fichiers COBOL volumineux, vous devez utiliser une expression régulière pour séparer les enregistrements. Si le fichier COBOL peut être fractionné à l'aide d'une expression régulière, vous pouvez définir un paramètre d'entrée pour le lecteur de fichier complexe qui fournit une expression régulière déterminant la manière de séparer le traitement de l'enregistrement dans l'environnement Hadoop.

Configuration du lecteur de fichier complexe pour COBOL

Ajoutez une transformation Processeur de données qui transforme l'entrée COBOL en mappage avec un lecteur de fichier complexe. Configurez le lecteur de fichier complexe de façon à optimiser la manière dont le

mappage traite l'entrée COBOL dans un environnement Hive. Le codage d'entrée pour le lecteur de fichier complexe doit être EBCDIC.

1. Dans l'éditeur de mappage, créez un objet de lecteur de fichier complexe.
2. Pour configurer le lecteur de fichier complexe, procédez comme suit :
 - a. Dans l'onglet **Avancé** de la vue **Propriétés**, sélectionnez la propriété **Format de fichier**, puis sélectionnez **Format d'entrée**.
 - b. Sélectionnez la propriété **Format d'entrée**, puis entrez `com.informatica.hadoop.reader.RegexInputFormat`.
 - c. Pour optimiser les performances, utilisez la propriété **Paramètres du format d'entrée** pour définir l'expression régulière sous la forme `Regex="<expression régulière>"`.
3. Créez une transformation Processeur de données pour l'entrée COBOL à l'aide de l'assistant.
4. Ajoutez la transformation Processeur de données au mappage. Le port d'entrée de la transformation doit rester défini sur une entrée binaire, paramètre par défaut d'une entrée COBOL.
5. Liez le port de sortie du lecteur de fichier complexe au port d'entrée de la transformation Processeur de données. Le port de sortie du lecteur de fichier complexe doit rester défini sur une sortie binaire.

JSON

Lorsque vous créez une transformation Processeur de données avec entrée ou sortie JSON, vous devez sélectionner un schéma JSON ou un fichier exemple pour la transformation. Le schéma ou le fichier exemple définissent la structure de la hiérarchie attendue pour les données d'entrée ou de sortie.

JSON (JavaScript Object Notation) est un format d'échange de données hiérarchique semblable à XML. Le format JSON est souvent utilisé pour transmettre des données structurées sur une connexion réseau. Un mappeur ou un sérialiseur utilise un schéma d'entrée ou un document d'entrée JSON de la même manière qu'un schéma d'entrée ou un document d'entrée XML pour définir la hiérarchie attendue pour les données d'entrée. Un analyseur utilise un schéma de sortie ou un document de sortie JSON pour définir la hiérarchie attendue pour les données de sortie.

Lorsque vous utilisez l'assistant de la transformation Processeur de données pour créer une transformation avec une entrée ou une sortie JSON, la transformation peut contenir un analyseur, un mappeur, un transformateur ou un sérialiseur associé à la hiérarchie JSON. Le schéma JSON est converti en fichier .xsd, lequel définit la structure de la hiérarchie du fichier JSON. Vous pouvez utiliser ce schéma .xsd avec n'importe quel document JSON disposant du même format hiérarchique.

Schémas JSON

Lorsque vous créez une transformation Processeur de données avec l'assistant, vous utilisez un schéma JSON ou un exemple de source pour définir la hiérarchie de l'entrée ou de la sortie JSON.

L'assistant de la transformation Processeur de données génère un schéma XML dans le référentiel modèle qui spécifie la structure JSON utilisée par les composants de la transformation. La transformation contient un transformateur associé au schéma et peut contenir d'autres composants, en fonction de l'entrée ou de la sortie que vous avez sélectionnée dans l'assistant.

Les scripts utilisent les schémas pour définir la structure des hiérarchies d'entrée et de sortie. Un schéma d'entrée JSON doit être conforme à l'ébauche Internet Draft (I-D) publiée par le groupe Internet Engineering Task Force pour le schéma JSON.

Pour plus d'informations sur la syntaxe du schéma JSON, consultez les sites internet suivants :

- Introduction à JSON : <http://www.json.org>
- Convertir un document JSON en schéma JSON : <http://jsonschema.net>

Exemple de schéma JSON

Voici un exemple de schéma JSON :

```
{ "type": "object",
  "$schema": "http://json-schema.org/draft-03/schema",
  "id": "#",
  "required": false,
  "properties": {
    "OrgId": {
      "type": "string",
      "id": "OrgId",
      "required": false
    },
    "metrics": {
      "type": "array",
      "id": "metrics",
      "required": false,
      "items": {
        "type": "object",
        "id": "0",
        "required": false,
        "properties": {
          "name": {
            "type": "string",
            "id": "name",
            "required": false
          },
          "valueTrend": {
            "type": "array",
            "id": "valueTrend",
            "required": false,
            "items": {
              "type": "object",
              "id": "0",
              "required": false,
              "properties": {
                "date": {
                  "type": "string",
                  "id": "date",
                  "required": false
                },
                "val": {
                  "type": "string",
                  "id": "val",
                  "required": false
                }
              }
            }
          }
        }
      }
    }
  }
}
```

Le schéma définit les éléments et les attributs pouvant apparaître dans un document JSON. Il utilise la syntaxe JSON pour spécifier la hiérarchie et la séquence des éléments, indiquer si les éléments sont requis, définir le type de l'élément et indiquer les valeurs possibles.

L'exemple de schéma ci-dessus définit le document d'entrée JSON suivant :

```
{ "OrgId": "ORG0000000000001",
  "metrics": [
```

```

{
  "name": "COL1",
  "valueTrend": [
    {
      "date": "2011-11-01",
      "val": "122.456"
    },
    {
      "date": "2011-11-02",
      "val": "215.1"
    }
  ]
},
{
  "name": "COL2",
  "valueTrend": [
    {
      "date": "2011-11-01",
      "val": "122.456"
    },
    {
      "date": "2011-11-02",
      "val": "215.1"
    }
  ]
}
]
}

```

Si vous analysez le schéma, vous pouvez déterminer la relation entre les éléments du schéma et le document d'entrée.

La hiérarchie de schéma contient l'objet `metrics` dans lequel est imbriqué le tableau `valueTrend`. Le tableau contient les champs `date` et `val`, dont le type de données est `string`.

Création d'une transformation avec JSON

Créez une transformation Processeur de données avec entrée ou sortie JSON.

1. Dans l'outil Developer, cliquez sur **Fichier > Nouveau > Transformation**.
2. Sélectionnez la transformation Processeur de données et cliquez sur **Suivant**.
3. Entrez un nom pour la transformation et sélectionnez un emplacement du référentiel modèle pour y stocker la transformation.
4. Sélectionnez **Créer un processeur de données à l'aide d'un assistant**, puis cliquez sur **Suivant**.
5. Sélectionnez un format d'entrée, puis cliquez sur **Suivant**.
6. Si vous sélectionnez JSON comme format d'entrée, recherchez et sélectionnez un schéma JSON ou un fichier exemple JSON. Cliquez sur **Suivant**.

L'extension d'un fichier JSON est généralement `*.json`. Developer ajoute un fichier de schéma XSD représentant la hiérarchie JSON au référentiel modèle.

7. Sélectionnez un format de sortie, puis cliquez sur **Suivant**.
8. Si vous sélectionnez JSON comme format de sortie, recherchez et sélectionnez un schéma JSON ou un fichier exemple JSON. Cliquez sur **Suivant**.

Si vous avez sélectionné JSON comme format d'entrée, vous ne pouvez pas le sélectionner comme format de sortie.

9. Cliquez sur **Terminer**.

L'outil Developer crée la transformation dans le référentiel. La vue **Présentation** s'affiche dans l'outil Developer.

10. Dans la vue **Objets**, double-cliquez sur le composant de transformation pour l'ouvrir dans l'éditeur IntelliScript.

Parquet

Utilisez l'assistant pour créer une transformation avec une entrée ou une sortie Parquet. Lorsque vous créez une transformation Processeur de données pour transformer le format Parquet, vous sélectionnez un schéma ou un exemple de fichier Parquet qui définit la structure attendue des données Parquet. L'assistant crée des composants qui transforment le format Parquet en d'autres formats ou d'autres formats en format Parquet. Une fois que l'assistant a créé la transformation, vous pouvez configurer davantage la transformation afin de déterminer la logique de mappage.

Apache Parquet est un format de stockage qui peut être traité dans un environnement Hadoop. Le format Parquet est implémenté pour résoudre les structures de données complexes imbriquées et utilise un enregistrement shredding ainsi qu'un algorithme d'assemblage. Pour plus d'informations sur Parquet, consultez <http://parquet.incubator.apache.org/documentation/latest/>.

Une transformation qui lit une entrée ou une sortie Parquet repose sur un schéma. Lorsque la transformation lit ou écrit des données Parquet, la transformation utilise ce schéma pour interpréter la hiérarchie.

Après avoir créé une transformation Processeur de données pour une entrée Parquet, vous l'ajoutez à un mappage avec un lecteur de fichier complexe. Le lecteur de fichier complexe transmet l'entrée Parquet à la transformation. Pour une transformation Processeur de données avec une sortie Parquet, vous ajoutez un rédacteur de fichier complexe au mappage pour recevoir la sortie depuis la transformation.

Création d'une transformation avec une entrée ou une sortie Parquet

Créez une transformation Processeur de données avec une entrée ou une sortie Parquet.

1. Dans l'outil Developer, cliquez sur **Fichier > Nouveau > Transformation**.
2. Sélectionnez la transformation Processeur de données et cliquez sur **Suivant**.
3. Entrez un nom pour la transformation et sélectionnez un emplacement du référentiel modèle pour y stocker la transformation.
4. Sélectionnez **Créer un processeur de données à l'aide d'un assistant**, puis cliquez sur **Suivant**.
5. Sélectionnez un format d'entrée, puis cliquez sur **Suivant**.
6. Si vous sélectionnez Parquet comme format d'entrée, recherchez un schéma Parquet ou un exemple de fichier Parquet et sélectionnez-le. Cliquez sur **Suivant**.
L'outil Developer ajoute un fichier d'objet de schéma représentant la hiérarchie Parquet dans le référentiel modèle.
7. Sélectionnez un format de sortie, puis cliquez sur **Suivant**.
8. Si vous sélectionnez Parquet comme format de sortie, recherchez un schéma Parquet ou un exemple de fichier Parquet et sélectionnez-le. Cliquez sur **Suivant**.
Si vous avez sélectionné Parquet comme format d'entrée, vous ne pouvez pas le sélectionner comme format de sortie.
9. Cliquez sur **Terminer**.

L'outil Developer crée la transformation dans le référentiel. La vue **Présentation** s'affiche dans l'outil Developer.

10. Dans la vue **Objets**, double-cliquez sur le composant de transformation pour l'ouvrir dans l'éditeur IntelliScript.

Pour configurer le composant de transformation, ajoutez des instructions de mappage.

Configurez l'entrée Parquet du lecteur de fichier complexe

Après avoir créé une transformation Processeur de données qui convertit une entrée Parquet, ajoutez la transformation à un mappage à l'aide d'un lecteur de fichier complexe. Configurez le lecteur de fichier complexe pour traiter une entrée Parquet.

1. Dans l'éditeur de mappage, créez un objet de lecteur de fichier complexe.
2. Pour configurer le lecteur de fichier complexe, procédez comme suit :
 - a. Dans l'onglet **Avancé** de la vue **Propriétés**, sélectionnez la propriété **Format de fichier**, puis sélectionnez **Format d'entrée**.
 - b. Dans l'onglet **Avancé**, sélectionnez la propriété **Format d'entrée**, puis entrez `com.informatica.parquet.ParquetToXML`.
 - c. Pour optimiser les performances, utilisez la propriété **Paramètres du format d'entrée** pour régler le paramètre `MaxOutputAccumulation`. Par défaut, le paramètre `MaxOutputAccumulation` qui définit le nombre attendu d'enregistrements de sortie est défini sur 50 000. Pour modifier le paramètre sur 250 000, par exemple, entrez `"MaxOutputAccumulation"="250000"`.
 - d. Par défaut, le lecteur de fichier complexe ajoute le schéma à la sortie du lecteur de fichier complexe dans un seul élément directement après l'élément racine. Si vous ne voulez pas ajouter le schéma à la sortie, sélectionnez la propriété **Paramètres du format d'entrée**, puis entrez `"InjectSchema"="false"`.

Utilisez un point-virgule pour séparer plusieurs paramètres, par exemple `"MaxOutputAccumulation"="250000"; "InjectSchema"="false"`.
3. Ajoutez la transformation Processeur de données au mappage. Le port d'entrée de la transformation doit rester défini sur une entrée binaire, qui correspond au paramètre par défaut pour une entrée Parquet.
4. Liez le port de sortie du lecteur de fichier complexe au port d'entrée de la transformation Processeur de données. Le port de sortie du lecteur de fichier complexe doit rester défini sur une sortie binaire.

Configurer une transformation avec une sortie Parquet

Pour créer une transformation Processeur de données pour une sortie Parquet, utilisez l'assistant de transformation Processeur de données. L'assistant crée la transformation dans le référentiel modèle avec les composants nécessaires à la transformation de la sortie Parquet. Utilisez l'éditeur XMap pour éditer la XMap, si celle-ci est incluse. Ajoutez la transformation à un mappage avec un rédacteur de fichier complexe.

1. Créez la transformation Processeur de données à l'aide de l'assistant Nouvelle transformation. Ajoutez un schéma ou un exemple de fichier Parquet qui définit la structure de sortie prévue.
2. Si l'entrée de la transformation est au format XML, JSON ou un autre format structuré, utilisez l'éditeur XMap pour éditer et personnaliser la XMap créée par l'assistant dans la transformation.
3. Pour configurer les paramètres de sortie de la transformation Processeur de données, dans le panneau **Contrôle de sortie** de la vue **Paramètres**, sélectionnez l'option de sortie adéquate dans la zone **Mode de collecte des ports de sortie binaires**.

Remarque: Pour configurer la sortie afin d'envoyer un enregistrement à la fois avec le schéma, sélectionnez l'option **Sortir une ligne pour chaque ligne**. Pour collecter tous les enregistrements avec le schéma dans un seul flux de sortie, sélectionnez l'option **Collecter les lignes d'entrée vers une seule sortie**.

4. Ajoutez la transformation Processeur de données à un mappage. La transformation reste définie sur une sortie binaire, ce qui est le paramètre par défaut pour une sortie Parquet.
5. Créez et liez un rédacteur de fichier complexe pour la transformation Processeur de données dans le mappage pour la réception de la sortie Parquet depuis la transformation.

Remarque: Le paramètre d'entrée reste défini sur binaire, ce qui est le paramètre par défaut. Le rédacteur de fichier complexe ne prend pas en charge la compression pour la sortie Parquet.

6. Configurez le rédacteur de fichier complexe. Dans l'opération d'objet de données, dans l'onglet **Avancé**, pour **Format de fichier**, sélectionnez **Sortie personnalisée**. Pour **Format de sortie**, entrez `com.informatica.parquet.XMLToParquet`.

XML

Utilisez l'assistant pour créer une transformation avec une entrée ou une sortie XML. Lorsque vous créez une transformation Processeur de données avec une entrée ou une sortie XML, vous sélectionnez un schéma .xsd ou un fichier exemple XML pour la transformation. Le schéma ou le fichier exemple définissent la structure de la hiérarchie attendue pour les données d'entrée ou de sortie. Si vous sélectionnez un fichier exemple, l'assistant crée un schéma à partir de la hiérarchie des données du fichier exemple.

L'assistant crée une transformation Processeur de données qui peut contenir un analyseur, un mappeur, une XMap ou un sérialiseur associé à la hiérarchie XML. Un mappeur, une XMap ou un sérialiseur utilise un schéma d'entrée pour définir la hiérarchie attendue des données d'entrée. Une XMap, un mappeur ou un analyseur utilise un schéma de sortie pour définir la hiérarchie attendue des données de sortie. Pour plus d'informations sur la syntaxe des schémas, voir <http://www.w3.org>.

L'assistant crée les composants de base ou le mappage relationnel dont la transformation a besoin selon les types d'entrée et de sortie. La transformation peut être complète ou servir de point de départ pour une configuration plus avancée.

Si la transformation a une entrée et une sortie structurées, l'assistant peut créer une XMap que vous configurez pour transformer des données d'une hiérarchie vers un autre. Utilisez l'éditeur XMap pour lier les nœuds des hiérarchies de schéma d'entrée et de sortie et définir la logique de transformation. Pour plus d'informations sur l'objet et l'éditeur XMap, voir ["Présentation de XMap" à la page 84](#).

Si la transformation a une entrée ou une sortie relationnelle que vous souhaitez transformer à partir de données structurées ou en données structurées, l'assistant crée un mappage relationnel. Le panneau **Ports** dans la vue **Présentation** affiche les nœuds du schéma hiérarchique et les ports relationnels. Utilisez le panneau **Ports** pour lier les éléments hiérarchiques aux ports et aux groupes relationnels. Pour plus d'informations sur la transformation de données relationnelles, voir ["Présentation des entrées et sorties relationnelles" à la page 65](#).

Création d'une transformation qui transforme du XML

Créez une transformation Processeur de données avec une entrée XML, une sortie XML ou les deux.

1. Dans l'outil Developer, cliquez sur **Fichier > Nouveau > Transformation**.
2. Sélectionnez la transformation Processeur de données et cliquez sur **Suivant**.

3. Entrez un nom pour la transformation et sélectionnez un emplacement du référentiel modèle pour y stocker la transformation.
4. Sélectionnez **Créer un processeur de données à l'aide d'un assistant**, puis cliquez sur **Suivant**.
5. Sélectionnez le format XML ou un autre format d'entrée et cliquez sur **Suivant**.
6. Si vous avez sélectionné XML en tant que format d'entrée, accédez à un schéma ou à un fichier exemple XML et sélectionnez-le. Cliquez sur **Suivant**.
Developer ajoute un fichier de schéma .xsd représentant la hiérarchie au référentiel modèle.
7. Sélectionnez le format XML ou un autre format de sortie et cliquez sur **Suivant**.
8. Si vous sélectionnez XML en tant que format de sortie, accédez à un schéma ou à un fichier exemple XML et sélectionnez-le. Cliquez sur **Suivant**.
9. Cliquez sur **Terminer**.
L'outil Developer crée la transformation dans le référentiel. La vue **Présentation** s'affiche dans l'outil Developer.
10. Pour éditer un composant spécifique de la transformation, dans la vue **Objets**, double-cliquez sur le composant de transformation afin de l'ouvrir dans l'éditeur IntelliScript.

CHAPITRE 4

Entrée et sortie relationnelle

Ce chapitre comprend les rubriques suivantes :

- [Présentation des entrées et sorties relationnelles, 65](#)
- [Entrée relationnelle, 65](#)
- [Sortie relationnelle, 72](#)

Présentation des entrées et sorties relationnelles

Une transformation Processeur de données peut lire l'entrée de la base de données relationnelles à partir de ports d'entrée et la transformer en d'autres formats. Une transformation peut envoyer des lignes de données relationnelles vers les ports de sortie. Vous pouvez définir le mappage avec des ports de la transformation Processeur de données dans la vue Présentation de la transformation. Vous pouvez également utiliser l'assistant de la transformation Processeur de données pour mapper automatiquement des données relationnelles.

Vous pouvez transformer les données relationnelles en données hiérarchiques. Pour transformer les groupes d'entrée en données hiérarchiques, mappez les nœuds du groupe de ports relationnels sur les ports hiérarchiques. Vous pouvez transférer les données des ports de sortie hiérarchiques vers une autre transformation du mappage.

Vous pouvez renvoyer une sortie relationnelle depuis la transformation Processeur de données. Si un composant renvoie des données relationnelles, créez des groupes de ports de sortie en mappant des nœuds de l'entrée hiérarchique sur des groupes de ports relationnels. Vous pouvez transférer les données depuis les ports de sortie relationnels vers une autre transformation dans un mappage.

Entrée relationnelle

Une transformation Processeur de données peut convertir une entrée relationnelle en sortie hiérarchique. Une base de données relationnelle est une base de données qui contient un ensemble de tables de données organisées selon un modèle relationnel. Les tables peuvent avoir des relations supplémentaires entre elles.

Dans le modèle relationnel, chaque schéma de table identifie une colonne appelée clé primaire, pour identifier de façon unique chaque ligne. Vous identifiez la relation entre chaque ligne de la table et une ligne d'une autre table avec une clé étrangère. Une clé étrangère est une colonne dans une table qui pointe vers la clé primaire d'une autre table.

Pour convertir des données de port relationnel en données hiérarchiques, vous devez définir la structure du mappage selon un schéma hiérarchique. Vous importez un schéma dans le référentiel modèle. Après avoir importé le schéma, vous pouvez visualiser les composants de schéma dans l'outil Developer. Si le schéma hiérarchique contient plusieurs éléments susceptibles de constituer un élément racine, choisissez un nœud comme élément racine.

Dans le panneau **Ports** de la vue **Présentation**, vous pouvez mapper les ports d'entrée relationnelle à des nœuds de schéma. Sur le côté gauche du panneau apparaît la zone **Entrée de transformation** et sur le côté droit de l'onglet apparaît la zone **Entrée de service**.

Lorsque vous faites glisser un nœud depuis un nœud **Entrée de service** vers un port **Entrée de transformation**, vous effectuez un mappage d'un nœud de schéma à un nœud d'entrée relationnelle. L'outil Developer crée les ports d'entrée pour mapper les données. Vous pouvez définir des groupes, des ports et mapper des nœuds de l'entrée vers les ports de sortie.

Une transformation Processeur de données avec entrée relationnelle peut contenir des ports d'intercommunication. Les ports d'intercommunication doivent être ajoutés au groupe racine de la structure relationnelle.

Configuration de port d'entrée relationnelle

Pour mapper des ports d'entrée relationnelle sur une sortie hiérarchique, sélectionnez un schéma pour définir la sortie. Vous devez également définir les ports d'entrée relationnelle. Dans le panneau Ports, créez et définissez des groupes de ports et mappez les nœuds des nœuds hiérarchiques sur les ports.

Pour définir un mappage d'entrée, sélectionnez le **mode du processeur de données** comme étant le **Mappage d'entrée** ou le **Mappage d'entrée et de service**. Le mappage utilise un schéma pour définir la sortie. Si le schéma comporte plusieurs éléments pouvant être un élément racine, vous pouvez choisir un nœud du schéma en tant qu'élément racine.

Pour définir un nœud en tant que racine, cliquez sur **Choisir Hiérarchie**. L'outil Developer affiche uniquement les nœuds au niveau de la racine et sous celle-ci dans la zone **Sortie de transformation**. Cliquez sur **Afficher sous forme hiérarchique** pour afficher les nœuds de sortie dans une hiérarchie. Chaque groupe enfant s'affiche sous le groupe parent.

Créez et définissez des ports d'entrée relationnelle avec l'une des méthodes suivantes :

Faites glisser les nœuds dans les ports

Faites glisser les nœuds depuis la zone **Sortie de transformation** vers la zone **Entrée de transformation**. Si vous faites glisser un nœud dans un groupe, l'outil Developer ajoute un port au groupe. Sinon, il crée un groupe avec le port à l'intérieur.

Créez les ports manuellement

Pour créer un port, sélectionnez un champ vide dans la zone **Entrée de transformation** et cliquez sur **Nouveau > Champ**. Si vous ne sélectionnez pas un champ dans un groupe, l'outil Developer crée un groupe et ajoute le port au groupe.

Créez les ports automatiquement

Cliquez sur **Mappage automatique**. L'outil Developer crée des groupes d'entrée et y ajoute des ports en fonction de la hiérarchie de sortie

Lorsque vous faites glisser des nœuds dans la zone **Entrée de transformation**, l'outil Developer met à jour le champ d'emplacement avec l'emplacement du nœud dans la hiérarchie. Si vous créez les ports manuellement, vous devez mapper un nœud au port. Cliquez sur la colonne **Emplacement** et sélectionnez un nœud dans la liste.

Lorsque vous faites glisser un nœud à occurrences multiples dans un groupe qui contient l'élément parent, vous pouvez configurer le nombre d'occurrences d'un élément enfant à inclure. Vous pouvez également remplacer le groupe parent par le groupe enfant à occurrences multiples dans la sortie de transformation.

Pour créer un groupe relationnel, faites glisser un nœud de sortie vers une colonne vide de la zone **Entrée de transformation**. Si vous faites glisser un nœud enfant à occurrences multiples vers une colonne d'entrée vide, l'outil Developer vous demande d'associer le groupe à d'autres groupes. Lorsque vous sélectionnez un groupe, l'outil Developer crée des clés pour associer les groupes.

Vous pouvez également créer un nouveau groupe relationnel en cliquant sur **Nouveau > Groupe** dans la zone **Entrée de transformation**. Donnez un nom au groupe. Configurez les groupes de ports d'entrée associés dans la zone **Entrée de transformation**. Quand l'outil Developer vous invite à associer des groupes de sortie, il ajoute les clés dans les groupes. Vous pouvez également ajouter des ports manuellement pour représenter les clés.

Pour afficher des lignes qui connectent les ports aux nœuds hiérarchiques, cliquez sur **Afficher les lignes**. Choisissez d'afficher toutes les lignes de connexion ou uniquement les lignes des ports sélectionnés.

Directives pour lier les ports d'entrée

Lorsque vous utilisez l'assistant Nouvelle transformation pour générer automatiquement une transformation Processeur de données, l'outil Developer crée des ports relationnels selon la hiérarchie du schéma, puis lie les ports relationnels aux nœuds hiérarchiques.

Tenez compte des règles et directives suivantes lorsque vous liez des ports d'entrée relationnelle à des nœuds de sortie hiérarchique.

- Vous pouvez lier un port d'entrée relationnelle à un nœud de la hiérarchie.
- Liez une clé primaire à un groupe relationnel en entrée depuis l'élément ou l'attribut pertinent de la hiérarchie. La clé primaire identifie chaque ligne dans les tables relationnelles.
- Liez une clé étrangère à un groupe relationnel en entrée depuis l'élément ou l'attribut pertinent de la hiérarchie. Dans l'entrée relationnelle, une clé étrangère est une colonne d'un tableau qui pointe vers la clé primaire d'un autre tableau.
- Les types de données du port d'entrée relationnelle et du nœud hiérarchique doivent être compatibles.
- Les clés peuvent être de type chaîne ou de type nombre entier.

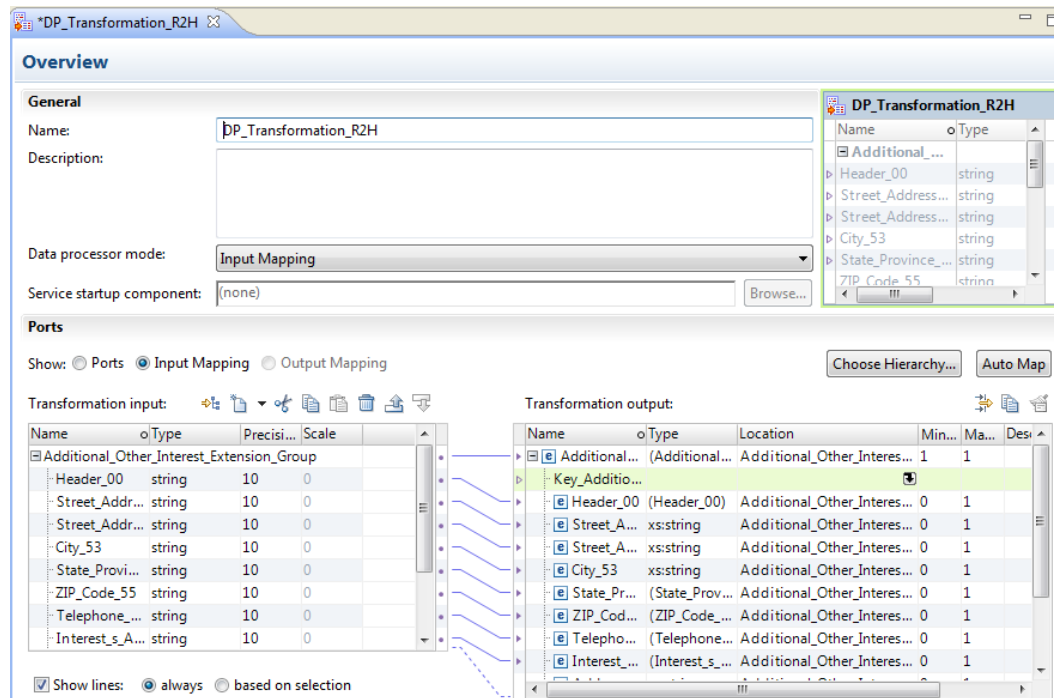
Définir les ports d'entrée relationnelle avec la vue Présentation

Pour transformer des données relationnelles en données hiérarchiques dans la transformation Processeur de données, mappez les nœuds d'un groupe de ports relationnels sur les nœuds hiérarchiques. Utilisez la vue Présentation pour lier des ports relationnels aux ports hiérarchiques.

Pour transformer une entrée relationnelle en sortie hiérarchique, activez l'entrée relationnelle depuis la vue **Présentation**. L'outil Developer supprime le port d'entrée par défaut dans la vue.

Sélectionnez **Mappage d'entrée**. Le panneau **Ports** s'affiche dans la vue **Présentation**.

L'image suivante montre le panneau **Ports** :



À gauche du panneau **Ports** se trouve la zone **Sortie de transformation**, qui contient les nœuds du schéma hiérarchique. Sur la droite apparaît la zone **Entrée de transformation** qui contient les éléments et les groupes relationnels.

Vous pouvez créer des ports dans la zone **Entrée de transformation** et lier les éléments relationnels aux nœuds du schéma. Vous pouvez également faire glisser le pointeur depuis un nœud du schéma vers un champ vide dans la zone **Entrée de transformation** pour créer un port. Lorsque vous connectez un port relationnel à un nœud de schéma, l'outil Developer affiche un lien entre eux.

Ports Clustering_Key

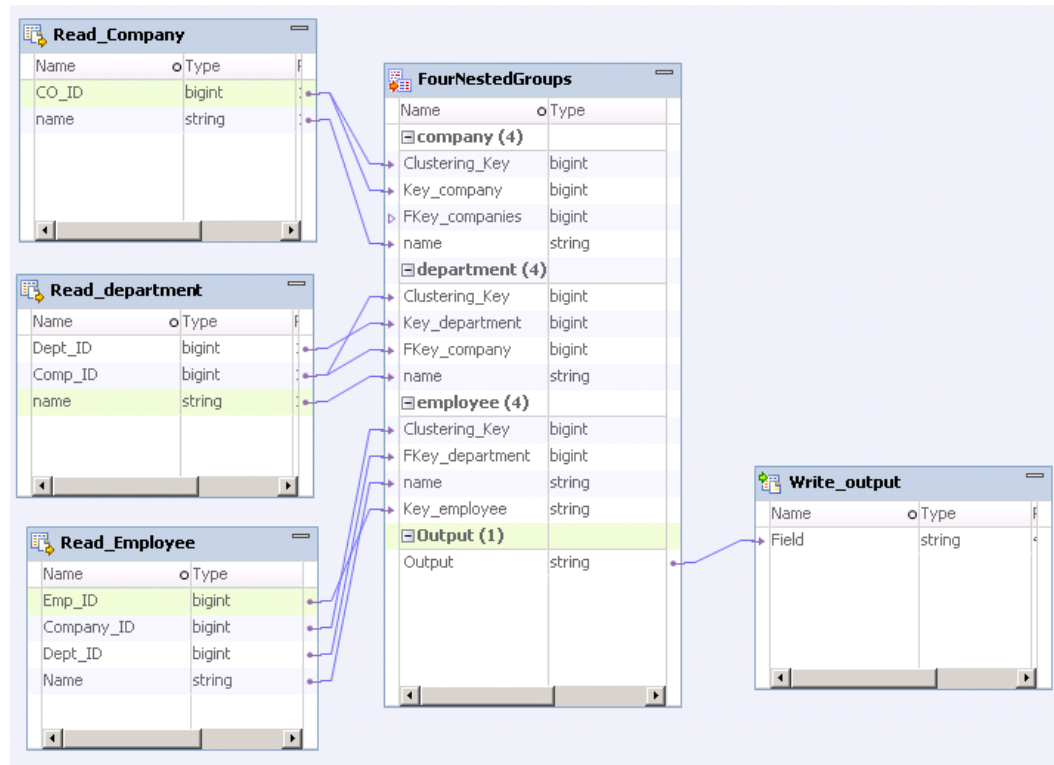
Lorsque vous créez une transformation Processeur de données d'un format relationnel en format hiérarchique avec plusieurs groupes dans l'environnement Hive, activez le partitionnement des données d'entrée pour vous assurer que les données de chaque ligne soient traitées correctement. Le système d'intégration de données partitionne les lignes d'entrée conformément à un port, qui fonctionne en tant que clé de partitionnement nommée **Clustering_Key**.

Pour partitionner les données d'entrée vers une transformation Processeur de données dans un mappage, sélectionnez la transformation dans le mappage, puis, dans l'onglet **Avancé** de la vue **Propriétés**, sélectionnez l'option d'activation du partitionnement. Lorsque vous activez le partitionnement, le développeur crée un port **Clustering_Key** dans la transformation Processeur de données pour chaque groupe d'entrées.

Chaque groupe d'entrées doit utiliser la même clé étrangère pour le groupe racine d'entrées afin de contribuer au partitionnement. Pour trier les données selon une clé, connectez le port d'entrée relationnel de la clé étrangère sélectionnée pour chaque objet de données au port **Clustering_Key** pertinent dans la transformation Processeur de données. Le service d'intégration de données utilise le port **Clustering_Key** pour partitionner les données et les traiter.

Vous devez utiliser la même clé dans tous les groupes d'entrées relationnelles. Si nécessaire, vous pouvez utiliser une transformation Jointure pour ajouter la clé à un groupe d'entrées relationnelles ne disposant pas de cette dernière.

L'image suivante montre un mappage dans lequel la clé étrangère Company_ID dans les groupes d'entrées relationnelles est liée aux ports Clustering_Key dans la transformation Processeur de données :



Entrée relationnelle normalisée

Lorsque vous normalisez les données d'entrée relationnelle dans la sortie hiérarchique, les valeurs de données ne se répètent pas dans le groupe hiérarchique. Vous créez une relation un à un entre les niveaux de hiérarchie dans les données de la sortie hiérarchique et les groupes de ports d'entrée.

Exemple d'entrée relationnelle normalisée

Vous voulez transformer l'entrée relationnelle avec un groupe qui contient les détails des gestionnaires de plusieurs entreprises pour séparer les hiérarchies XML. En entrée, chaque enregistrement de gestionnaire contient les détails de la société. En sortie, une hiérarchie XML contient les détails des entreprises et une autre hiérarchie XML contient les détails des gestionnaires.

Dans l'entrée relationnelle, les éléments Company_ID et Company_Name se répètent pour chaque gestionnaire de l'entreprise :

Company_ID	Company_Name	Manager_ID	Manager_Name
100	Percy Accounting	76500	Cindy Jacques
100	Percy Accounting	46501	Tom Jorry
100	Percy Accounting	86509	Delilah Smith

Si vous définissez la sortie XML pour qu'elle contienne un niveau de hiérarchie parent Company et un niveau de hiérarchie enfant Managers, vous pouvez utiliser les groupes de hiérarchie suivants :

```
Companies
  Company_Key
  Company_ID
```

```
Company_Name
```

Managers

```
Company_Key  
Manager_ID  
Manager_Name
```

L'élément `Company_Key` lie la hiérarchie `Managers` à la hiérarchie `Companies`.

Entrée relationnelle pivotée

Vous pouvez inclure un nombre spécifique d'éléments à occurrences multiples dans un groupe de sortie.

Pour pivoter des éléments à occurrences multiples, mappez l'élément de port à occurrences multiples à des nœuds de sortie spécifiques.

Exemple d'entrée relationnelle pivotée

Vous voulez transformer l'entrée relationnelle qui contient une table de numéros de téléphone à une hiérarchie XML avec des éléments séparés pour différents types de numéros de téléphone.

Dans l'entrée relationnelle, l'élément `Telephone_Type` définit le type de numéro de téléphone répertorié pour chaque personne :

Telephone_Number	Telephone_Type	Last_Name	First_Name
9173327437	Mobile	Sandrine	Jacques
9174562342	Mobile	Race	Tom
8484526471	Domicile	Race	Tom
7023847265	Bureau	Smith	Delilah
9174596725	Mobile	Smith	Delilah

Dans la hiérarchie XML de sortie `Telephones`, différents types de numéros de téléphone dans le groupe parent ont des éléments séparés :

```
Telephones  
  Telephone_Number  
  Last_Name  
    First_Name  
  Work_Telephone  
  Mobile_Telephone  
  Home_Telephone
```

Entrée relationnelle dénormalisée

Vous pouvez dénormaliser la sortie hiérarchique pour l'entrée relationnelle. Lorsque vous dénormalisez les données d'entrée, les valeurs d'élément provenant du groupe parent se répètent pour chaque élément enfant.

Pour dénormaliser les données d'entrée, mappez les nœuds d'un groupe de ports à un niveau de hiérarchie enfant. Tous les éléments se répètent au niveau de hiérarchie enfant.

Exemple d'entrée relationnelle dénormalisée

Vous voulez transformer l'entrée relationnelle avec des groupes séparés pour les détails des gestionnaires et des sociétés dans une hiérarchie JSON qui contient à la fois les détails des gestionnaires et ceux des sociétés.

L'élément Company_Name n'apparaît pas dans le groupe contenant les détails des gestionnaires. L'élément Company_ID est la clé étrangère dans le premier groupe relationnel.

Company_ID	Manager_ID	Manager_Name
100	56673	Kathy Jason
100	23501	Jackie Lyons
100	44509	Bob Terrence

Le deuxième groupe relationnel contient les détails de la société.

Company_ID	Company_Name
100	Percy Accounting
102	Sandy Auto Sales
410	Movers Inc.

L'élément Gestionnaires de la sortie JSON contient à la fois l'élément Company_ID et l'élément Company_Name :

```
Managers
  Company_ID
  Company_Name
  Manager_ID
  Manager_Name
```

Les éléments Company_ID et Company_Name se répètent pour chaque gestionnaire du service.

Mappage de ports relationnels sur des nœuds hiérarchiques

Dans le panneau Ports, définissez des groupes de ports et mappez les nœuds du schéma hiérarchique sur les ports.

1. Pour ajouter un schéma, dans la vue **Références**, cliquez sur **Ajouter**. Accédez à un schéma et sélectionnez-le.
2. Pour créer un mappage d'entrée, dans la zone **Général** de la vue **Présentation**, sélectionnez **Mappage d'entrée** comme mode du processeur de données.
3. Dans la zone **Ports**, sélectionnez **Mappage d'entrée**.
4. Sélectionnez un nœud en tant que racine.
5. Pour définir automatiquement un mappage, cliquez sur **Mappage automatique**. Pour définir manuellement un mappage, vous définissez une clé primaire pour le groupe d'entrée racine, puis définissez les groupes et les ports d'entrée.
6. Pour ajouter un groupe ou un port d'entrée dans la zone **Entrée de transformation**, utilisez une des méthodes suivantes :
 - Faites glisser un nœud de groupes hiérarchique ou un nœud enfant de la zone **Sortie de transformation** vers une colonne vide de la zone **Entrée de transformation**. S'il s'agit d'un nœud de groupe, l'outil Developer ajoute un groupe relationnel sans ports.
 - Pour ajouter un groupe relationnel, sélectionnez une ligne et cliquez avec le bouton droit de la souris pour sélectionner **Nouveau > Groupe**.
 - Pour ajouter un port relationnel, cliquez avec le bouton droit de la souris et sélectionnez **Nouveau > Champ**.
7. Pour mapper des nœuds en tant que clé primaire, utilisez l'une des méthodes suivantes :
 - Sélectionnez deux ou plusieurs nœuds de hiérarchie et faites-les glisser vers une clé dans la zone **Entrée de transformation**.

- Cliquez sur la colonne **Emplacement** d'une clé dans la zone **Sortie de transformation** et sélectionnez le port d'entrée relationnelle dans la zone **Entrée de transformation**.
8. Pour effacer les paramètres de nœuds hiérarchiques des emplacements de ports, utilisez l'une des méthodes suivantes :
- Sélectionnez un ou plusieurs nœuds dans la zone **Sortie de transformation**, cliquez avec le bouton droit de la souris et sélectionnez **Effacer**.
 - Sélectionnez une ou plusieurs lignes qui connectent les ports d'entrée relationnelle aux nœuds hiérarchiques de la zone **Sortie de transformation**, cliquez avec le bouton droit de la souris et sélectionnez **Supprimer**.

Sortie relationnelle

Lorsque vous configurez une sortie relationnelle, vous pouvez configurer un groupe de sortie séparé pour chaque nœud d'entrée à occurrences multiples. Vous pouvez également créer des groupes qui contiennent des données dénormalisées. Vous pouvez pivoter des éléments à occurrences multiples et limiter le nombre d'occurrences dans un groupe de sortie.

Configuration du port de sortie relationnelle

Pour transformer une entrée hiérarchique en sortie relationnelle, sélectionnez un schéma afin de définir les données hiérarchiques. Vous devez également définir les ports de sortie relationnelle. Dans le panneau Ports, définissez des groupes de ports et mappez les nœuds du schéma hiérarchique sur les ports.

Pour définir un mappage de sortie, sélectionnez le **Mode du processeur de données** comme étant le **Mappage de sortie** ou le **Mappage de sortie et service**.

Le mappage utilise un schéma pour définir l'entrée hiérarchique. Si le schéma comporte plusieurs éléments pouvant être un élément racine, choisissez un nœud en tant qu'élément racine. Pour définir un nœud en tant que racine, cliquez sur **Sélectionner une hiérarchie**. L'outil Developer affiche uniquement les nœuds à partir du niveau racine et sous celui-ci dans la zone **Entrée de transformation**.

Cliquez sur **Afficher sous forme hiérarchique** pour afficher les ports de sortie dans une hiérarchie. Chaque groupe enfant s'affiche sous le groupe parent.

Créez des ports à l'aide de l'une des méthodes suivantes :

Faites glisser les nœuds dans les ports

Faites glisser les nœuds depuis la zone **Entrée de transformation** vers la zone **Sortie de transformation**. Si vous faites glisser un nœud dans un groupe, l'outil Developer ajoute un port au groupe. Sinon, il crée un groupe avec le port à l'intérieur.

Créez les ports manuellement

Pour créer un port, sélectionnez un champ vide dans la zone **Sortie de transformation** et cliquez sur **Nouveau > Champ**. Si vous ne sélectionnez pas un champ dans un groupe, l'outil Developer crée un groupe et ajoute le port au groupe.

Lorsque vous faites glisser des nœuds dans la zone **Sortie de transformation**, l'outil Developer met à jour le champ d'emplacement avec l'emplacement du nœud dans la hiérarchie. Si vous créez les ports manuellement, vous devez mapper un nœud au port. Cliquez sur la colonne **Emplacement** et sélectionnez un nœud dans la liste.

Lorsque vous faites glisser un nœud à occurrences multiples dans un groupe qui contient l'élément parent, vous pouvez configurer le nombre d'occurrences d'un élément enfant à inclure. Vous pouvez également remplacer le groupe parent par le groupe enfant à occurrences multiples dans la sortie de la transformation.

Pour créer un groupe, faites glisser un nœud vers une colonne vide dans la zone **Sortie de transformation**. Si vous faites glisser un nœud enfant à occurrences multiples dans une colonne d'entrée ou de sortie vide, l'outil Developer vous demande d'associer le groupe à d'autres groupes de sortie. Lorsque vous sélectionnez un groupe, l'outil Developer crée des clés pour associer les groupes.

Vous pouvez également créer un nouveau groupe en cliquant sur **Nouveau > Groupe**. Donnez un nom au groupe.

Configurez les groupes de ports de sortie associés dans la zone **Sortie de transformation**. Quand l'outil Developer vous invite à associer des groupes de sortie, il ajoute les clés dans les groupes. Vous pouvez également ajouter des ports manuellement pour représenter les clés.

Pour afficher des lignes qui connectent les ports aux nœuds hiérarchiques, cliquez sur **Afficher les lignes**. Sélectionnez cette option pour afficher toutes les lignes de connexion ou uniquement les lignes des ports sélectionnés.

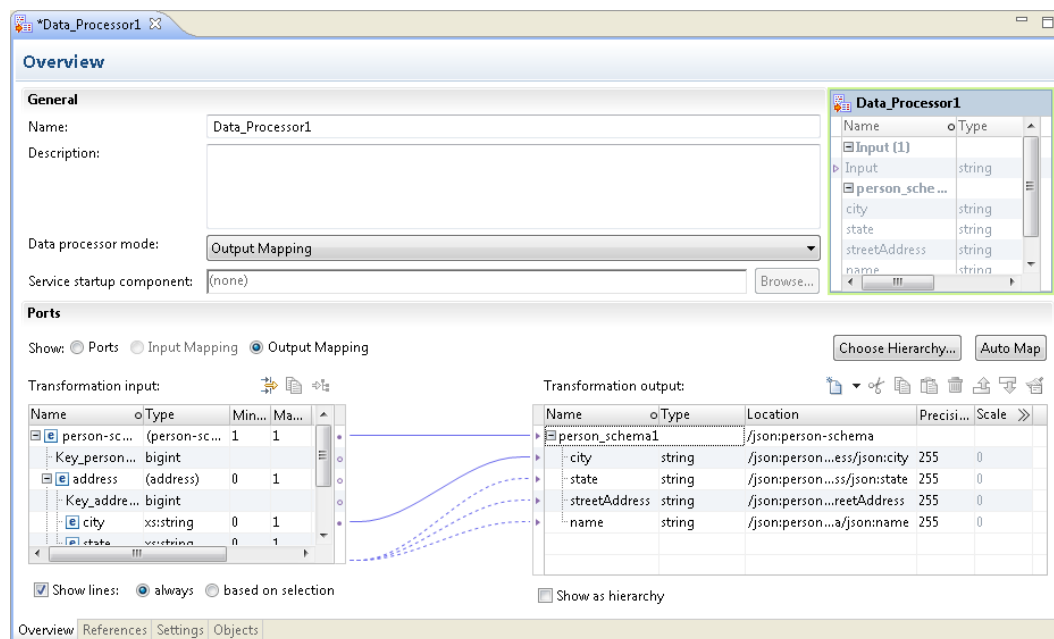
Définir les ports relationnels de sortie dans la vue Présentation

Pour convertir des données hiérarchiques en sortie relationnelle dans la transformation Processeur de données, liez des nœuds hiérarchiques à des ports relationnels. Utilisez la vue Présentation pour lier des ports relationnels aux ports hiérarchiques. Vous créez des groupes de ports de sortie en liant des nœuds de la sortie hiérarchique à des groupes de ports.

Pour renvoyer des groupes de ports relationnels, activez la sortie relationnelle depuis la vue **Présentation**. L'outil Developer supprime le port de sortie par défaut dans la vue.

Sélectionnez **Mappage de sortie**. Le panneau **Ports** s'affiche dans la vue **Présentation**.

L'image suivante montre le panneau **Ports** :



La zone **Entrée de transformation**, qui affiche le schéma de sortie, se trouve à gauche. La zone **Sortie de transformation**, qui affiche les ports de sortie relationnelle, se trouve à droite.

Définissez les ports de sortie relationnels dans la zone **Sortie de transformation** et liez les nœuds du schéma aux ports. Vous pouvez également faire glisser le pointeur depuis un nœud du schéma vers un champ vide dans la zone **Sortie de transformation** pour créer un port. Lorsque vous faites glisser un nœud depuis le schéma de sortie vers un port, l'outil Developer affiche un lien entre eux.

Sortie relationnelle normalisée

Lorsque vous créez des données de sortie normalisées, les valeurs des données ne se répètent pas dans un groupe de sortie. Vous créez une relation un à un entre les niveaux de hiérarchie dans les données de l'entrée hiérarchique et les groupes de ports de sortie.

Exemple de sortie relationnelle normalisée

Vous voulez transformer une hiérarchie JSON qui définit les détails des services et des employés en une sortie relationnelle avec des groupes séparés pour les détails des services et des employés.

L'entrée JSON contient une hiérarchie Ressources Humaines avec des éléments qui contiennent des détails relatifs aux employés et aux services.

```
Staff
  Department_ID
  Department_Name
  Employee_ID
  Employee_Name
```

Vous pouvez créer les groupes suivants de ports relationnels:

Department_Key	Employee_ID	Employee_Name
100	2673	Jason Stuart
100	1501	Lila Rose
100	4309	Sarah Jacobs

Department_Key	Department_ID	Department_Name
100	1982	Comptabilité
102	3297	Ventes
410	8276	Logistique

Department_Key est une clé générée qui associe le groupe Employés au groupe Service dans la sortie.

Sortie relationnelle pivotée

Vous pouvez inclure un nombre spécifique d'éléments à occurrences multiples dans un groupe de sortie.

Pour orienter des éléments à occurrences multiples, mappez l'élément enfant à occurrences multiples au groupe parent des ports de sortie. L'outil Developer vous invite à définir le nombre d'éléments enfant à inclure dans le parent.

Exemple de sortie relationnelle pivotée

Vous voulez transformer une hiérarchie XML qui contient les détails des employés avec plusieurs identifiants à un groupe de sortie relationnelle avec des éléments de sortie séparés pour les ID d'employé.

L'exemple suivant illustre deux instances d'Employee_ID dans le groupe de sortie relationnelle Departments :

```
Departments
  Department_ID
  Department_Name
  Employee_ID1
  Employee_ID2
```

Sortie relationnelle dénormalisée

Vous pouvez dénormaliser la sortie relationnelle. Lorsque vous dénormalisez les données de sortie, les valeurs de l'élément depuis le groupe parent se répètent pour chaque élément enfant.

Pour dénormaliser les données de sortie, liez les nœuds du niveau de la hiérarchie parent au groupe enfant de ports de sortie.

Exemple de sortie relationnelle dénormalisée

Vous voulez transformer une hiérarchie XML avec des groupes séparés pour les détails des employés et ceux des services en un groupe relationnel qui contient simultanément les détails des employés et des services.

Les hiérarchies XML séparent les détails de Department de ceux de Employee :

```
Department
  Department_ID
  Department_Name

Employees
  Department_ID
  Employee_ID
  Employee_Name
```

L'exemple suivant affiche le Department_ID et le Department_Name dans le groupe de sortie Employees :

Department_ID	Department_Name	Employee_ID	Employee_Name
100	Comptabilité	56500	Kathy Jones
100	Comptabilité	56501	Tom Lyons
100	Comptabilité	56509	Bob Smith

Les éléments Department_ID et Department_Name se répètent pour chaque employé du service.

CHAPITRE 5

Utilisation de l'éditeur IntelliScript

Ce chapitre comprend les rubriques suivantes :

- [Présentation de l'éditeur IntelliScript, 76](#)
- [Ouverture d'un éditeur IntelliScript, 77](#)
- [Procédures de modification, 78](#)
- [Menus de l'éditeur IntelliScript, 82](#)

Présentation de l'éditeur IntelliScript

L'éditeur IntelliScript est l'éditeur principal dans lequel vous pouvez configurer un composant Script, comme Analyseur, Mappeur ou Sérialiseur.

L'éditeur IntelliScript présente une structure arborescente. Le haut (niveau global de l'arborescence) affiche les composants de transformation exécutables tels que les Analyseurs. Ces composants sont appelés « exécutables » car vous pouvez les définir comme composants de démarrage de la transformation.

Vous pouvez également définir des composants non exécutables, tels que les variables ou les actions, au niveau global. Ceci permet d'utiliser les composants à d'autres emplacements du projet.

Aux niveaux imbriqués de l'éditeur IntelliScript, vous pouvez définir les composants tels que les ancres et les actions. Vous pouvez cliquer sur les symboles + ou – pour développer l'arborescence de l'éditeur IntelliScript et afficher les niveaux imbriqués.

Création d'un script

Créez un objet Script et définissez le type de composant Script à créer. Éventuellement, vous pouvez définir une référence de schéma et un fichier de source d'exemple.

1. Dans la vue **Objets** de la transformation Processeur de données, cliquez sur **Nouveau**.
2. Entrez le nom du script et cliquez sur **Suivant**.
3. Choisissez de créer un analyseur ou un sérialiseur. Sélectionnez Autre pour créer un composant Mappeur, Transformateur ou Répartiteur.
4. Entrez le nom du composant.
5. Si le composant est le premier à traiter les données dans la transformation, activez **Définir en tant que composant de démarrage**.
6. Cliquez sur **Suivant** si vous voulez entrer une référence de schéma pour ce script. Cliquez sur **Terminer** si vous ne voulez pas entrer la référence de schéma.

7. Si vous choisissez de créer une référence de schéma, sélectionnez **Ajouter une référence à un objet de schéma** et recherchez l'objet de schéma dans le référentiel modèle. Cliquez sur **Créer un objet de schéma** pour créer l'objet de schéma dans le référentiel modèle.
8. Cliquez sur **Suivant** pour entrer une référence de source d'exemple ou pour entrer un texte d'exemple. Cliquez sur **Terminer** si vous ne voulez pas définir une source d'exemple.
Utilisez une source d'exemple pour définir des données d'exemple et pour tester le script.
9. Si vous choisissez de sélectionner une source d'exemple, sélectionnez **Fichier** et recherchez le fichier d'exemple.
Vous pouvez également saisir un texte d'exemple dans la zone **Texte**. L'outil Developer utilise le texte pour tester un script.
10. Cliquez sur **Terminer**.
La vue **Script** s'affiche dans l'éditeur de l'outil Developer.

Ouverture d'un éditeur IntelliScript

L'éditeur IntelliScript affiche un composant Script dans la transformation Processeur de données. Pour ouvrir un éditeur IntelliScript, double-cliquez sur un composant Script dans la transformation Processeur de données. Vous pouvez afficher les composants de la transformation Processeur de données dans l'onglet **Structure**.

IntelliScript et Visionneuse de données

L'éditeur IntelliScript affiche un ou plusieurs composants de Script. Il s'agit de l'emplacement où vous définissez les composants de transformation.

Vous pouvez afficher l'exemple de document source d'une transformation dans l'onglet **Visionneuse de données**. Ce volet permet d'afficher, de configurer ou de tester une transformation.

Le volet d'exemple **Visionneuse de données** est en lecture seule. Vous ne pouvez pas modifier l'exemple de document source dans Data Transformation Studio.

Vous pouvez afficher le Script en mode script, avec une représentation du Script sous forme de code, ou dans IntelliMode, à l'aide de l'éditeur IntelliScript. Par défaut, vous affichez le Script dans IntelliMode.

Recherche d'ancres

Lorsque vous modifiez un analyseur, il est facile de rechercher les ancres correspondantes dans IntelliScript et dans les volets d'exemples.

- Cliquez avec le bouton droit de la souris sur une ancre dans IntelliScript, puis cliquez sur **Afficher le marquage**. Cela a pour effet de rechercher l'ancre dans le volet d'exemple.
- Cliquez sur une ancre dans le volet d'exemple. L'ancre est automatiquement sélectionnée dans IntelliScript.

Composants et propriétés

IntelliScript contient deux types d'éléments :

- Composants. Éléments que vous pouvez insérer et supprimer dans l'arborescence d'IntelliScript. Les analyseurs, les sérialiseurs, les ancrs, les actions et les transformateurs sont des exemples de composants.
- Propriétés. Éléments que vous pouvez modifier, mais que vous ne pouvez pas insérer ou supprimer, sauf en insérant ou en supprimant un composant qui les contient. La propriété `example_source` d'un analyseur ou la propriété `search` d'une ancre de marqueur sont des exemples de propriétés.

L'éditeur IntelliScript utilise des couleurs différentes pour afficher les composants et les propriétés.

Propriétés de base et avancées

Pour permettre de simplifier l'affichage, IntelliScript organise les propriétés en deux catégories :

- Propriétés de base. Propriétés importantes dans la plupart des utilisations du composant. Vous devez généralement attribuer ces propriétés pour utiliser le composant.
- Propriétés avancées. Propriétés qu'il n'est pas nécessaire d'attribuer souvent. Ces propriétés peuvent comporter des valeurs par défaut, ne nécessitant le plus souvent pas de modification, ou peuvent implémenter des options le plus souvent inutilisées.

IntelliScript affiche toujours les propriétés de base. Il masque les propriétés avancées sauf si vous choisissez de les afficher.

La distinction est utilisée à des fins d'affichage uniquement. Les propriétés avancées sont toutes aussi faciles à utiliser que les propriétés de base. N'hésitez pas à les utiliser si elles sont nécessaires dans vos projets.

Affichage des propriétés avancées d'un composant

- Cliquez sur l'icône **>>** à droite du nom du composant.
L'icône **>>** est remplacée par **<<**, et les propriétés avancées s'affichent.

Masquage des propriétés avancées

- Cliquez sur l'icône **<<** à droite du nom du composant.
L'icône **<<** redevient **>>**, et les propriétés avancées ne s'affichent plus. En revanche, si vous avez attribué une valeur autre que celle par défaut à une propriété avancée, elle reste visible.

Procédures de modification

Pour modifier IntelliScript, suivez les procédures décrites dans cette section.

Procédure de base de modification

Pour modifier IntelliScript :

1. Cliquez sur le composant ou sur la propriété à modifier.

2. Appuyez sur **ENTRÉE** pour passer en mode modification. Dans la plupart des emplacements, vous pouvez également double-cliquer au lieu d'appuyer sur **ENTRÉE**.
3. Attribuez le nom du composant ou la valeur de la propriété.
4. Appuyez à nouveau sur **ENTRÉE** pour terminer la modification.

Copier et coller

Vous pouvez copier et coller des composants dans IntelliScript.

Pour copier simultanément plusieurs composants, appuyez sur la touche **CTRL** ou **MAJ** tout en effectuant la sélection à l'aide de la souris. Les composants doivent tous se situer au même niveau d'imbrication dans IntelliScript.

Vous ne pouvez coller des composants qu'aux emplacements où ils ont un sens. Par exemple, vous pouvez copier les ancres de sérialisation sous un sérialiseur, mais pas sous un analyseur.

Glisser et déposer

Vous pouvez déplacer des composants d'un emplacement à un autre en les faisant glisser à l'aide de la souris. Par exemple, vous pouvez utiliser cette méthode pour modifier la séquence des ancres dans un analyseur.

Pour déplacer plusieurs composants, appuyez sur la touche **CTRL** ou **MAJ** tout en effectuant la sélection à l'aide de la souris. Relâchez la touche **CTRL** ou **MAJ**, puis faites-les glisser à l'aide de la souris.

Pour copier les composants sélectionnés (plutôt que de les déplacer), maintenez la touche **MAJ** enfoncée tout en faisant glisser les composants.

Rechercher et remplacer

Pour rechercher ou remplacer du texte dans IntelliScript, sélectionnez **Modifier > Rechercher** ou **Modifier > Remplacer**.

Insertion de composants dans IntelliScript

Sous de nombreux composants, IntelliScript affiche une ligne horizontale, généralement portant une étiquette telle que *contient*. La ligne est suivie d'une flèche et de trois points (. . .). Vous pouvez insérer des composants imbriqués au niveau des trois points.

Insertion d'un composant

1. Sélectionnez les trois points et appuyez sur **ENTRÉE**.
Une liste des composants s'affiche.
2. Sélectionnez un composant dans la liste.
Sinon, commencez à saisir le nom du composant. Une saisie semi-automatique du nom s'effectue après la saisie des premières lettres.
3. Appuyez à nouveau sur **ENTRÉE** pour terminer l'insertion.

Suppression d'un composant

- Sélectionnez le composant et appuyez sur **SUPPRIMER**.

Modification des propriétés d'un composant

1. Sélectionnez la valeur d'une propriété et appuyez sur **ENTRÉE**.
Selon le type de propriété, une zone de texte, une liste ou une boîte de dialogue s'affiche.
2. Saisissez ou sélectionnez la nouvelle valeur de propriété.
3. Appuyez sur **ENTRÉE** pour terminer l'attribution.

Insertion de tabulations, de nouvelles lignes et d'autres caractères spéciaux

Lorsque vous attribuez une propriété de texte, vous pouvez y insérer des caractères spéciaux en saisissant leurs codes ASCII numériques.

1. Sélectionnez une propriété et appuyez sur **ENTRÉE** pour commencer la modification.
2. Appuyez sur les touches **CTRL+A**.
Un petit point s'affiche, indiquant ainsi que le caractère est un code ASCII.
3. Saisissez le code ASCII à trois chiffres. Par exemple, vous pouvez saisir :

Code ASCII	Caractère
009	Tabulation
010	Nouvelle ligne
013	Retour chariot

4. Pour saisir une chaîne de codes ASCII, répétez les étapes 2 et 3.
Vous pouvez alterner des codes ASCII et du texte normal.
5. Appuyez sur **ENTRÉE** pour terminer la modification.
Une tabulation s'affiche sous la forme d'un symbole «. D'autres caractères s'affichent sous la forme de leurs codes de caractères ASCII.

Définition d'un composant global

Vous pouvez insérer des composants de portée globale ou locale.

Portée	Description
Portée globale	Le composant est défini au niveau supérieur d'IntelliScript. Vous pouvez y accéder ou l'utiliser à n'importe quel emplacement du projet.
Portée locale	Le composant est défini à un niveau imbriqué d'IntelliScript. Vous pouvez y accéder ou l'utiliser uniquement au niveau où il est imbriqué.

La plupart des composants de Data Transformation peuvent être globaux ou locaux.

Par exemple, les ancres sont généralement définies localement. Vous pouvez cependant définir une ancre comme composant global si vous souhaitez utiliser une même configuration d'ancre dans plusieurs analyseurs ou plusieurs fois dans un même analyseur. À chaque emplacement où vous souhaitez utiliser l'ancre définie globalement, vous pouvez la référencer par son identifiant.

Un analyseur peut alors utiliser `MyMarker`, plutôt que de répéter la configuration de l'ancre de marqueur à chaque fois qu'elle est nécessaire. Vous pouvez sélectionner `MyMarker` dans la liste de composants à l'emplacement approprié dans l'analyseur ou faire glisser `MyMarker` vers cet emplacement.

Restrictions de nommage

Les noms attribués aux composants IntelliScript ne doivent contenir que des caractères anglais (A-Z, a-z), des chiffres (0-9) et des traits de soulignement (_). Ils doivent commencer par une lettre. Ils peuvent contenir 127 caractères au maximum.

Affichage de l'aide sur un composant





Les rubriques d'aide en ligne décrivant les composants ou les propriétés peuvent être visualisées pendant la modification d'IntelliScript.

1. Affichez la vue **Aide**.
2. Sélectionnez le composant ou la propriété.
L'aide accède à l'emplacement qui décrit l'élément sélectionné.

Icônes IntelliScript

IntelliScript affiche chaque type de composant avec une icône caractéristique. Le tableau suivant décrit les icônes les plus courantes qui s'affichent dans IntelliScript.

Icône	Composant
	Parser
	Serializer
	Mapper
	Transformer
	Marker
	Content ContentSerializer
	Group
	RepeatingGroup
	StringSerializer
	Handle

Icône	Composant
	Key
	Autres ancrs
	Actions
	Icône par défaut, utilisée en l'absence d'icône spécifique pour un composant

Sauvegarde d'IntelliScript

Si un astérisque (*) s'affiche dans l'onglet Titre d'un éditeur IntelliScript, les modifications de l'éditeur ne sont pas sauvegardées. Si vous essayez de fermer l'éditeur ou de quitter alors que des modifications ne sont pas sauvegardées, vous êtes invité à sauvegarder le Script.

Menus de l'éditeur IntelliScript

Cliquez avec le bouton droit de la souris dans un éditeur IntelliScript pour afficher un menu. Les options de menu dépendent du contexte dans lequel vous cliquez.

Tableau 1. Menu du volet IntelliScript

Option	Description
Afficher le marquage	Met en surbrillance l'ancre sélectionnée dans l'exemple de source.
Définir comme composant d'installation	Définit le composant sélectionné comme composant de démarrage du projet.
Couper	Permet de couper les composants dans IntelliScript.
Copier	Permet de copier les composants dans IntelliScript.
Coller	Permet de coller les composants dans IntelliScript.
Insérer	Permet d'insérer les composants dans IntelliScript.
Supprimer	Permet de supprimer les composants dans IntelliScript.
Rendre facultatif	Sélectionne la propriété <code>optional</code> d'un composant. Si un composant est facultatif, un échec du composant n'entraîne pas celui de son composant parent. Pour plus d'informations, consultez le <i>Data Transformation Studio</i> .
Rendre obligatoire	Désélectionne la propriété <code>optional</code> d'un composant. Si un composant est facultatif, un échec du composant n'entraîne pas celui de son composant parent. Pour plus d'informations, consultez le <i>Data Transformation Studio</i> .

Option	Description
Activer	Sélectionne la propriété <code>disabled</code> d'un composant. Un composant désactivé est ignoré. Cette fonctionnalité est utile pour désactiver temporairement un composant à des fins de test et de débogage.
Désactiver	Désélectionne la propriété <code>disabled</code> d'un composant. Un composant désactivé est ignoré. Cette fonctionnalité est utile pour désactiver temporairement un composant à des fins de test et de débogage.
Mode Script	Le mode Script affiche le contenu brut du fichier *.tgp. Ce mode est uniquement destiné au dépannage avancé.
Mode Intelli	Le mode Intelli affiche IntelliScript sous forme d'une représentation graphique et lisible. Il s'agit du mode illustré tout au long de ce manuel et dans le reste de la documentation de Data Transformation.
Ouvrir l'exemple de source	Ouvre l'exemple de fichier source de l'analyseur, du sérialiseur ou du mappeur sélectionné.
Créer le sérialiseur	Crée un sérialiseur à partir de l'analyseur sélectionné. Le sérialiseur et l'analyseur effectuent des transformations inverses.

Tableau 2. Menu du volet d'exemple

Option	Description
Copier	Copie une chaîne dans le Presse-papiers.
Insérer un marqueur	Définit le texte sélectionné comme ancre de <code>marqueur</code> . L'ancre est ajoutée à IntelliScript.
Insérer un contenu	Définit le texte sélectionné comme ancre de <code>contenu</code> . L'ancre est ajoutée à IntelliScript.
Insérer un contenu de décalage	Définit l'emplacement sélectionné comme ancre de <code>contenu</code> . L'ancre est ajoutée à IntelliScript.
Insérer un RepeatingGroup	Définit le texte sélectionné comme séparateur d'une ancre <code>RepeatingGroup</code> . L'ancre est ajoutée à IntelliScript.
Afficher l'instance	Recherche l'ancre sélectionnée dans IntelliScript.
Afficher l'événement	Recherche l'événement correspondant dans la vue Événements .
Rechercher	Recherche une chaîne dans l'exemple de document source.
Codage logique	Si l'exemple de source contient du texte dans une langue se lisant de droite à gauche comme l'hébreu ou l'arabe, cette commande bascule l'affichage de gauche-à-droite vers de droite-à-gauche.
Retour automatique à la ligne	Renvoie à la ligne les longs segments de texte.
Enregistrer la source sous	Enregistre l'exemple de source à un emplacement spécifié, sous un nom spécifié.

CHAPITRE 6

XMap

Ce chapitre comprend les rubriques suivantes :

- [Présentation de XMap, 84](#)
- [Schémas XMap, 85](#)
- [Instructions de mappage, 86](#)
- [Expressions XPath, 103](#)
- [Variables XMap, 110](#)
- [Exemple XMap, 111](#)

Présentation de XMap

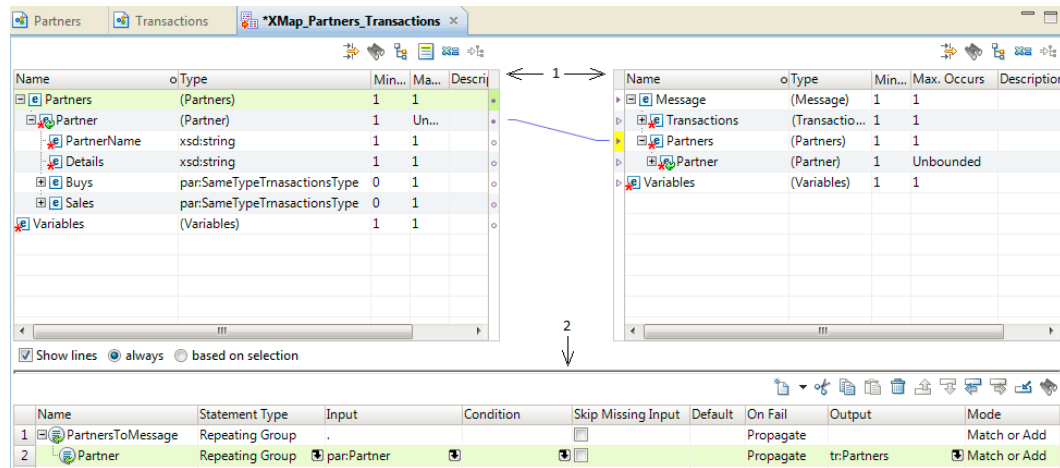
Une XMap est un objet de transformation Processeur de données qui modifie un document d'entrée hiérarchique en un autre document hiérarchique dont la structure hiérarchique est différente.

Une XMap utilise les schémas d'entrée et de sortie pour définir la hiérarchie attendue des documents d'entrée et de sortie. Utilisez l'éditeur XMap pour définir et gérer des instructions de mappage. L'éditeur XMap contient la hiérarchie du schéma d'entrée et celle du schéma de sortie. Les instructions de mappage lient les éléments du schéma d'entrée aux éléments du schéma de sortie.

Une XMap peut transformer tout document hiérarchique d'entrée dont les éléments correspondent à la hiérarchie du schéma d'entrée en document de sortie avec la hiérarchie du schéma de sortie.

Par exemple, une XMap peut transformer des factures client en une liste de commandes client triées par mois de l'année. L'entrée est un document JSON qui contient une hiérarchie d'éléments client. La sortie est un document XML qui contient une hiérarchie d'éléments mois. Le document XML de sortie groupe la commande client par mois et inclut les informations de contact et les totaux de la commande.

L'image suivante montre l'éditeur XMap.



1. L'éditeur XMap contient des schémas hiérarchiques d'entrée et de sortie. Faites glisser et déplacez les éléments de schéma pour créer des instructions de mappage.
2. La grille de l'éditeur XMap affiche les instructions de mappage. Utilisez la grille pour gérer et éditer les instructions de mappage.

Une XMap utilise les instructions de mappage pour définir comment transformer un élément de schéma d'entrée en élément de schéma de sortie. Vous pouvez glisser d'un nœud dans le schéma d'entrée vers un nœud dans le schéma de sortie pour créer un lien. Lorsque vous créez des liens, ils sont des instructions de mappage. L'éditeur XMap affiche l'instruction de mappage dans la grille.

Vous pouvez éditer les instructions de mappage dans la grille. Vous pouvez définir des conditions pour transformer et filtrer les données selon différents types d'instructions de mappage et d'expressions XPath. XPath est un langage de requête utilisé pour sélectionner des nœuds dans un document hiérarchique et exécuter des calculs.

Vous pouvez utiliser les expressions XPath pour définir le contexte pour une instruction de mappage. Vous pouvez également ajouter divers calculs arithmétiques à une instruction de mappage à l'aide d'expressions XPath.

Schémas XMap

Une XMap requiert des schémas hiérarchiques qui définissent les hiérarchies hiérarchiques d'entrée et de sortie. Une XMap utilise des schémas hiérarchiques d'entrée et de sortie pour déterminer quel type de données est attendu dans le document source d'entrée et le document de sortie. Vous liez des nœuds de schéma d'entrée et de sortie pour créer des instructions de mappage.

Un élément de schéma est l'élément de base d'une instruction de mappage. Lorsque vous définissez une instruction de mappage, vous pouvez vouloir utiliser une série d'éléments de schéma d'entrée dans l'instruction, ou ajouter une variable. Vous pouvez changer la racine d'entrée et de sortie, ajouter des variables et personnaliser la vue du schéma, mais vous ne pouvez pas modifier le schéma.

Vous pouvez utiliser les options suivantes pour gérer les schémas et la recherche d'éléments :

Personnaliser la vue

Change la manière dont s'affiche le schéma pour vous permettre de rechercher rapidement les éléments pertinents. Vous pouvez rechercher une séquence de nœuds, tous les nœuds ou un choix de nœuds. Affichez ces nœuds pour comprendre la logique du schéma. Cette vue n'affecte pas le mappage.

Search

Recherche des éléments dans le schéma. Vous pouvez effectuer une recherche séparément dans les schémas d'entrée et de sortie.

Variables

Définissez des variables pour stocker les données. Vous pouvez mapper des nœuds à des variables et vice-versa. Vous pouvez également mapper des variables à des variables. Lorsque vous créez une variable, la variable s'affiche en bas des deux schémas.

Sélection de la racine d'entrée ou de sortie

Modifiez l'élément racine dans le schéma d'entrée ou de sortie. Vous pouvez changer l'élément racine afin de référencer une autre partie du schéma.

Choisissez la source d'exemple

Définissez le fichier exemple source. Utilisez une source d'exemple pour tester la transformation et les expressions XPath.

Instructions de mappage

Une instruction de mappage détermine la méthode de mappage des données du document hiérarchique d'entrée sur le document hiérarchique de sortie. Lorsque vous faites glisser un nœud depuis le schéma d'entrée vers le schéma de sortie, l'éditeur XMap crée des instructions de mappage dans la grille.

Vous pouvez utiliser la grille pour créer des instructions de mappage simples ou détaillées. Vous pouvez faire glisser les éléments depuis les schémas d'entrée ou de sortie dans les champs de la grille pour les intégrer aux instructions de mappage.

Vous pouvez vérifier l'élément auquel une instruction de mappage fait référence. Lorsque vous cliquez sur une instruction de mappage dans la grille, l'éditeur XMap met en évidence les nœuds dans les schémas.

Ajouter des expressions XPath pour déterminer le contexte ou pour ajouter des calculs à une instruction de mappage. Quand une expression XPath identifie le contexte pour une instruction de mappage, la transformation Processeur de données exécute l'instruction de mappage pour chaque occurrence de l'élément d'entrée ou de l'expression dans le document d'entrée.

Imbriquer des instructions de mappage pour les rendre dépendantes d'autres instructions de mappage. Une instruction de mappage peut être un parent pour un groupe d'instructions enfant. Chaque fois que la transformation Processeur de données exécute l'instruction parent, il exécute les instructions enfant également. Les instructions enfant s'affichent en retrait du parent dans l'éditeur XMap.

Types d'instructions Mappage

Les types d'instructions de mappage définissent la logique de mappage XMap. Définir le type d'instruction de mappage selon que vous voulez mapper une valeur d'entrée simple à une valeur de sortie, faire une répétition sur un élément ou effectuer le mappage en fonction d'une condition.

Créer une instruction en faisant glisser un élément du schéma d'entrée vers un élément du schéma de sortie, ou en ajoutant une instruction de mappage à la grille. Lorsque vous créez une instruction, la transformation

Processeur de données identifie un type d'instruction de mappage selon que l'élément est un élément simple, un élément complexe ou un élément répétitif.

Le type d'instruction de mappage de base est un mappage qui mappe une valeur d'entrée simple à une valeur de sortie simple. D'autres instructions de mappage identifient les conditions ou les alternatives pour la logique de mappage, ou groupent un ensemble d'instructions logiques.

Vous pouvez définir les types d'instructions de mappage suivants dans la grille :

Mappage

Mappe un élément d'entrée simple à un élément de sortie simple. Une instruction Mappage est l'élément de base de la XMap.

Groupe

Un groupe logique d'instructions. D'autres types d'instructions de mappage sont imbriqués dans l'instruction Groupe.

Groupe de répétition

Une instruction de groupe que la transformation Processeur de données effectue chaque fois que l'élément d'entrée s'affiche dans le document d'entrée. Le Groupe de répétition contient des instructions Mappage qui sont répétées. Le Groupe de répétition identifie l'élément utilisé pour répéter le groupe.

Routeur

Contient un groupe d'instructions Option et sélectionne uniquement l'instruction Option dont les critères de condition correspondent à l'entrée. Si aucune des options ne s'applique, une action par défaut est engagée, s'il existe une instruction par défaut. Si aucune des options ne s'applique et qu'il n'existe aucune instruction par défaut, le routeur échoue.

Option

Une ou plusieurs instructions Option sont imbriquées dans l'instruction Routeur. L'instruction Option est identique à une instruction Groupe et contient un groupe logique d'instructions. L'instruction Option définit une condition pour mapper l'élément d'entrée à l'élément de sortie.

Par défaut

Une instruction par défaut peut être imbriquée dans l'instruction Routeur. L'instruction par défaut est exécutée quand aucune des instructions Option ne s'applique. Si toutes les instructions Option échouent et qu'il n'existe aucune instruction par défaut, l'instruction Routeur échoue.

Exécuter XMap

Appelle un autre objet XMap dans la transformation Processeur de données.

RunMapplet

Appelle un mapplet depuis la transformation Processeur de données.

MappletInput

Une ou plusieurs instructions MappletInput peuvent être imbriquées dans l'instruction RunMapplet. Les valeurs sont mappées aux ports d'entrée du mapplet dans l'ordre dans lequel elles apparaissent dans les instructions MappletInput.

MappletOutput

Une ou plusieurs instructions MappletOutput peuvent être imbriquées dans l'instruction RunMapplet. Les valeurs des ports de sortie du mapplet sont mappées à l'instruction MappletOutput dans l'ordre dans lequel elles apparaissent dans les ports du mapplet.

Les instructions de mappage contiennent des champs que vous pouvez configurer pour personnaliser l'instruction. Vous pouvez configurer l'entrée, la sortie et une condition pour le mappage d'un élément d'entrée à un élément de sortie.

Configurez si une instruction de mappage doit être ignorée lorsqu'elle échoue ou qu'il n'y a pas d'entrée.
Configurez si la transformation Processeur de données ajoute un élément de sortie ou fait correspondre un élément existant avec une valeur depuis une instruction de mappage.

Instructions Mappage

Une instruction Mappage est l'élément de base pour créer un objet XMap et mappe une valeur d'entrée simple à une valeur de sortie simple. L'entrée doit être une valeur unique ou une valeur constante. Vous devez définir l'entrée et la sortie dans une instruction Mappage.

Lorsque vous faites glisser et déplacez un nœud de schéma d'entrée non répétitif simple et un nœud de schéma de sortie non répétitif simple, une instruction Mappage est automatiquement créée.

Une instruction Groupe, Groupe de répétition, Option ou par défaut peut contenir une ou plusieurs instructions Mappage enfant.

Propriétés de l'instruction Mappage

Une instruction Mappage contient des propriétés que vous pouvez configurer pour personnaliser l'instruction. Vous pouvez configurer l'entrée, la sortie et une condition pour le mappage d'un élément d'entrée à un élément de sortie.

Une instruction de mappage comprend les propriétés suivantes :

Condition

Facultatif. Une expression XPath qui définit une condition de mappage de l'élément. Une condition est similaire à une expression de prédicat dans la colonne Entrée. Si vous définissez une expression XPath Entrée et une expression XPath Condition pour la même instruction de mappage, la transformation Processeur de données applique la Condition XPath au résultat de l'expression XPath Entrée.

Par défaut

Facultatif. La valeur par défaut à utiliser quand un élément est manquant dans l'entrée. Par exemple, vous pouvez définir une valeur par défaut pour initialiser un compteur.

Entrée

Obligatoire. Une expression XPath qui définit l'élément d'entrée. L'expression peut s'évaluer sur un nœud ou une valeur.

Mode

Obligatoire. Détermine si la transformation Processeur de données ajoute un élément de sortie ou fait correspondre un élément existant avec une valeur depuis une instruction de mappage. Choisissez l'une des options suivantes :

- Ajouter. Crée un élément dans le document hiérarchique de sortie. Si l'élément n'est pas à occurrences multiples et que les mêmes valeurs existent dans la sortie, l'instruction de mappage échoue.
- Correspondance. L'instruction attend pour trouver une correspondance pour l'élément dans les éléments de sortie. L'instruction échoue si l'élément n'existe pas dans le document hiérarchique de sortie.
- Correspondance ou ajout. Si un élément correspondant existe dans le document hiérarchique de sortie, la transformation Processeur de données n'ajoute pas d'élément de sortie. Si l'élément n'existe pas dans le document hiérarchique de sortie, la transformation crée un élément de sortie.

Nom

Facultatif. Un nom pour l'instruction. Vous pouvez changer le nom à tout moment. Le nom identifie les instructions de sorte que vous pouvez les trouver dans la grille de mappage ou dans un journal des événements. Les noms d'instruction ne doivent pas obligatoirement être uniques.

En cas d'échec

Obligatoire. Détermine l'action engagée en cas d'échec de l'instruction. Choisissez l'une des options suivantes :

- Ignorer. Si l'instruction échoue, ignorer l'instruction.
- Propager. Si l'instruction échoue, faire aussi échouer l'instruction parent.

Sortie

Obligatoire. Une expression XPath qui définit la valeur de l'élément dans le format hiérarchique de sortie en fonction des résultats de l'expression XPath Entrée.

Ignorer l'entrée manquante

Facultatif. Détermine si l'instruction doit être ignorée s'il n'y a pas de correspondance avec la valeur Entrée. Choisissez l'une des options suivantes :

- Activé. Si l'élément ne se trouve pas dans le document hiérarchique d'entrée, la transformation Processeur de données ignore l'instruction sans erreur.
- Désactivé. L'instruction échoue lorsque l'élément ne se trouve pas dans le document hiérarchique d'entrée.

Type d'instruction

Obligatoire. Identifie l'instruction comme une instruction Mappage.

Instructions de groupe

Une instruction de groupe contient un groupe logique d'instructions. Une instruction de groupe parent contient des instructions enfant. Les instructions enfant sont imbriquées sous l'instruction de groupe dans la grille de l'éditeur XMap.

Vous pouvez utiliser une instruction de groupe pour fournir un contexte commun ou des conditions communes pour définir le succès ou l'échec d'un groupe d'instructions. Vous pouvez utiliser une instruction de mappage de groupe si vous voulez qu'un ensemble d'instructions réussisse ou échoue pour toutes les instructions. Vous pouvez utiliser une instruction de mappage de groupe pour grouper un ensemble d'instructions en vue d'organiser et de simplifier une grille XMap.

Lorsque vous liez un élément complexe à occurrence simple à un élément complexe à occurrence simple ou à occurrences multiples, l'éditeur XMap crée une instruction de groupe. Un élément à occurrence simple a un **max. La valeur d'occurrence** de 1.

Exemple d'instruction de groupe




Vous voulez mapper un document hiérarchique d'entrée contenant des données employé du manager sur un document hiérarchique de sortie contenant les données travailleur. Il n'y a qu'un seul manager donc l'élément d'entrée employé est à occurrence simple.

Une instruction de mappage de groupe s'utilise lorsqu'un élément est complexe, à occurrence simple. Le schéma a un élément Employé à occurrence simple. L'employé a des éléments enfant FirstName et LastName :

```
Employee
  FirstName
  LastName
```

Vous créez une instruction de mappage de groupe et configurez Employé comme entrée. Chaque instruction de mappage que vous intégrez au groupe est dans le contexte Employé.

Dans la figure suivante, l'instruction 1 est l'instruction de groupe :

	Nom	Type d'inst...	Entrée	Condition	Ignore...	Valeur...	En échec	Sortie
1	 EmployéetoWorker	Groupe	.		<input type="checkbox"/>		Propager	
2	 FirstName to FirstName	Mapper	tns0:FirstName		<input checked="" type="checkbox"/>		Ignorer	tns0:Fi
3	 LastName to LastName	Mapper	tns0:LastName		<input checked="" type="checkbox"/>		Ignorer	tns0:La

La colonne d'entrée pour l'instruction de groupe indique que l'entrée est l'élément parent de FirstName et LastName. Les instructions 2 et 3 sont des instructions enfant de l'instruction 1. Les instructions enfant s'affichent avec indentation à partir de l'instruction parent. Pour chaque élément Employé d'entrée, mappez les éléments FirstName et LastName à la sortie.

Propriétés de l'instruction de groupe

L'instruction de groupe contient des propriétés que vous pouvez configurer pour personnaliser l'instruction. Vous pouvez configurer l'entrée, la sortie et une condition pour le mappage d'un élément d'entrée à un élément de sortie.

L'instruction de groupe comprend les propriétés suivantes :

Condition

Facultatif. Une expression XPath qui définit la condition d'entrée pour l'instruction de groupe et toutes ses instructions enfant. Une condition est similaire à une expression de prédicat dans la colonne Entrée. Si vous définissez une expression XPath Entrée et une expression XPath Condition pour la même instruction de mappage, la transformation Processeur de données applique la Condition XPath au résultat de l'expression XPath Entrée.

Entrée

Facultatif. Une expression XPath qui s'évalue de zéro à un élément ou valeur. Si laissée vide, l'instruction utilise le contexte actuel. Si l'expression s'évalue à plus d'une valeur, la première est utilisée.

Mode

Obligatoire. Détermine si la transformation Processeur de données ajoute un élément de sortie ou fait correspondre un élément existant avec une valeur depuis une instruction de mappage. Choisissez l'une des options suivantes :

- Ajouter. Crée un élément dans le document hiérarchique de sortie. Si l'élément n'est pas à occurrences multiples et que les mêmes valeurs existent dans la sortie, l'instruction de mappage échoue.
- Correspondance. L'instruction attend pour trouver une correspondance pour l'élément dans les éléments de sortie. L'instruction échoue si l'élément n'existe pas dans le document hiérarchique de sortie.
- Correspondance ou ajout. Si un élément correspondant existe dans le document hiérarchique de sortie, la transformation Processeur de données n'ajoute pas d'élément de sortie. Si l'élément n'existe pas dans le document hiérarchique de sortie, la transformation crée un élément de sortie.

Nom

Facultatif. Un nom pour l'instruction. Vous pouvez changer le nom à tout moment. Le nom identifie les instructions de sorte que vous pouvez les trouver dans la grille de mappage ou dans un journal des événements. Les noms d'instruction ne doivent pas obligatoirement être uniques.

En cas d'échec

Facultatif. Détermine l'action engagée en cas d'échec de l'instruction. Choisissez l'une des options suivantes :

- Ignorer. Si l'instruction échoue, ignorer l'instruction.
- Propager. Si l'instruction échoue, faire aussi échouer l'instruction parent.

Sortie

Facultatif. Une expression XPath qui définit la valeur de l'élément dans le format hiérarchique de sortie en fonction des résultats de l'expression XPath Entrée. Si laissée vide, l'instruction utilise le contexte actuel.

Ignorer l'entrée manquante

Facultatif. Détermine si l'instruction doit être ignorée s'il n'y a pas de correspondance avec la valeur Entrée. Choisissez l'une des options suivantes :

- Activé. Si l'élément ne se trouve pas dans le document hiérarchique d'entrée, la transformation Processeur de données ignore l'instruction sans erreur.
- Désactivé. L'instruction échoue lorsque l'élément ne se trouve pas dans le document hiérarchique d'entrée.

Type d'instruction

Obligatoire. Identifie l'instruction comme une instruction de groupe.

Instructions du Groupe de répétition

Une instruction de groupe de répétition est une instruction de groupe qui peut se produire plusieurs fois. L'entrée est une expression XPath qui peut s'évaluer à une séquence d'éléments ou de valeurs.

La transformation Processeur de données exécute une instruction Groupe de répétition pour chaque élément ou valeur qui est un résultat de l'expression XPath Entrée.

Lorsque vous liez un élément de schéma d'entrée répétitif à un élément de schéma de sortie répétitif, l'éditeur XMap crée une instruction Groupe de répétition dans la grille. Un élément se répète si le **max. La valeur d'occurrence** de l'élément est supérieure à 1.

Exemple d'instruction du Groupe de répétition

Un schéma d'entrée possède la hiérarchie suivante :

```
Employees
  Employee (Unbounded)
    LastName
    FirstName
```

Lorsque vous faites glisser Employé vers un élément de sortie dans l'éditeur XMap, l'outil Developer crée par défaut une instruction de mappage Groupe de répétition. Un groupe de répétition peut contenir des instructions de mappage pour renvoyer LastName et FirstName pour chaque Employé dans le document hiérarchique d'entrée.

Propriétés de l'instruction Groupe de répétition

L'instruction Groupe de répétition contient des propriétés que vous pouvez configurer pour personnaliser l'instruction. Vous pouvez configurer l'entrée, la sortie et une condition pour le mappage d'un élément d'entrée à un élément de sortie.

L'instruction Groupe de répétition comprend les propriétés suivantes :

Condition

Facultatif. Une expression XPath qui définit une condition de mappage de l'élément. Une condition est similaire à une expression de prédicat dans la colonne Entrée. Si vous définissez une expression XPath Entrée et une expression XPath Condition pour la même instruction de mappage, la transformation Processeur de données applique la Condition XPath au résultat de l'expression XPath Entrée.

Entrée

Obligatoire. Une expression XPath qui s'évalue à une séquence de nœuds ou de valeurs.

Mode

Obligatoire. Détermine si la transformation Processeur de données ajoute un élément de sortie ou fait correspondre un élément existant avec une valeur depuis une instruction de mappage. Choisissez l'une des options suivantes :

- Ajouter. Crée un élément dans le document hiérarchique de sortie. Si l'élément n'est pas à occurrences multiples et que les mêmes valeurs existent dans la sortie, l'instruction de mappage échoue.
- Correspondance. L'instruction attend pour trouver une correspondance pour l'élément dans les éléments de sortie. L'instruction échoue si l'élément n'existe pas dans le document hiérarchique de sortie.
- Correspondance ou ajout. Si un élément correspondant existe dans le document hiérarchique de sortie, la transformation Processeur de données n'ajoute pas d'élément de sortie. Si l'élément n'existe pas dans le document hiérarchique de sortie, la transformation crée un élément de sortie.

Nom

Facultatif. Un nom pour l'instruction. Vous pouvez changer le nom à tout moment. Le nom identifie les instructions de sorte que vous pouvez les trouver dans la grille de mappage ou dans un journal des événements. Les noms d'instruction ne doivent pas obligatoirement être uniques.

En cas d'échec

Facultatif. Détermine l'action engagée en cas d'échec de l'instruction. Choisissez l'une des options suivantes :

- Ignorer l'itération. Si une instruction imbriquée dans le Groupe de répétition échoue et si sa propriété En cas d'échec est définie sur **Propager**, l'iteration en cours du Groupe de répétition est ignorée.
- Propager. Si l'instruction échoue, faire aussi échouer l'instruction parent.

Sortie

Facultatif. Une expression XPath qui définit la valeur du nœud dans le format hiérarchique de sortie en fonction des résultats de l'expression XPath Entrée.

Ignorer l'entrée manquante

Facultatif. Détermine si l'instruction doit être ignorée s'il n'y a pas de correspondance avec la valeur Entrée. Choisissez l'une des options suivantes :

- Activé. Si l'élément ne se trouve pas dans le document hiérarchique d'entrée, la transformation Processeur de données ignore l'instruction sans erreur.
- Désactivé. L'instruction échoue lorsque l'élément ne se trouve pas dans le document hiérarchique d'entrée.

Type d'instruction

Obligatoire. Identifie l'instruction comme une instruction Groupe de répétition.

Instructions Routeur

Une instruction Routeur fournit des alternatives pour la logique de mappage selon les conditions du document d'entrée.

L'instruction Routeur contient une ou plusieurs instructions Option et peut contenir une instruction par défaut. Lorsque la transformation Processeur de données exécute une instruction Routeur, elle teste chaque Option imbriquée dans l'instruction Routeur.

La première instruction Option qui correspond est exécutée. L'instruction Option peut contenir une ou plusieurs instruction enfant de tout type. Si aucune instruction Option ne correspond, l'instruction par défaut est exécutée. S'il n'y a aucune instruction par défaut, l'instruction Routeur échoue.

Vous pouvez configurer plusieurs instructions Option dans le même groupe Routeur. La transformation Processeur de données exécute la première instruction Option qu'il accepte. La transformation Processeur de données n'exécute aucune instruction Option en dessous dans le groupe. Lorsque la transformation Processeur de données n'accepte pas une instruction Option, elle teste la prochaine instruction Option.

Lorsque la transformation n'accepte aucune instruction Option et qu'il n'y a aucune instruction par défaut, l'instruction Routeur échoue. Si l'instruction Option a une condition qui est vraie alors que les instructions de mappage qui sont à l'intérieur échouent et propagent l'échec, le Routeur échoue. Vous pouvez configurer le mappage pour qu'il ignore le Routeur en cas d'échec du Routeur.

Si un Routeur ne contient aucune instruction Option mais contient une instruction par défaut, l'instruction par défaut est toujours exécutée.

Exemple d'instruction Routeur

Une XMap contient un groupe de répétition dans le contexte d'Employé. La première instruction enfant dans le groupe est une instruction Routeur. L'instruction Routeur a une instruction Option. L'instruction Option a une condition qui vérifie si la valeur Rôle est égale à la valeur Manager. Si le rôle est égal au manager, l'instruction Option s'évalue à « True ». Le mappage évalue l'instruction Exécuter XMap imbriquée dans l'instruction Option. La transformation Processeur de données demande à la XMap EmployeeToWorker de mapper les éléments vers Manager.

Si le rôle n'est pas égal au manager, l'instruction par défaut est vraie. Le mappage évalue l'instruction suivante pour l'option par défaut. L'instruction de mappage par défaut demande à la XMap EmployeeToWorker de mapper les éléments à Travailleur.

La figure suivante montre l'instruction Routeur avec une instruction Option et une instruction par défaut :

Employee to Worker	Groupe répétition	tns0:Employee	<input type="checkbox"/>	Ignorer l'itération	
Employee	Routeur		<input type="checkbox"/>	Ignorer	
Employee to Manager	Option	tns0:Role="M...	<input checked="" type="checkbox"/>	Propager	
EmployeeToWorker	EmployeeToWorker		<input type="checkbox"/>	Propager	tns0:Manager
EmployeeToWorker	Valeur par défaut		<input type="checkbox"/>	Propager	
EmployeeToWorker	EmployeeToWorker		<input type="checkbox"/>	Propager	tns0:Worker

Propriétés de l'instruction Routeur

L'instruction Routeur contient des propriétés que vous pouvez configurer pour personnaliser l'instruction. Vous pouvez configurer l'entrée, la sortie et une condition pour le mappage d'un élément d'entrée à un élément de sortie.

L'instruction Routeur comprend les propriétés suivantes :

Condition

Facultatif. Une expression XPath qui définit une condition de mappage de l'élément. Une condition est similaire à une expression de prédicat dans la colonne Entrée. Si vous définissez une expression XPath

Entrée et une expression XPath Condition pour la même instruction de mappage, la transformation Processeur de données applique la Condition XPath au résultat de l'expression XPath Entrée.

Par défaut

Obligatoire. La valeur par défaut à utiliser quand un élément est manquant dans l'entrée. Par exemple, vous pouvez définir une valeur par défaut pour initialiser un compteur.

Entrée

Obligatoire. Une expression XPath qui s'évalue à une séquence de nœuds ou de valeurs.

Mode

Obligatoire. Détermine si la transformation Processeur de données ajoute un élément de sortie ou fait correspondre un élément existant avec une valeur depuis une instruction de mappage. Choisissez l'une des options suivantes :

- Ajouter. Crée un élément dans le document hiérarchique de sortie. Si l'élément n'est pas à occurrences multiples et que les mêmes valeurs existent dans la sortie, l'instruction de mappage échoue.
- Correspondance. L'instruction attend pour trouver une correspondance pour l'élément dans les éléments de sortie. L'instruction échoue si l'élément n'existe pas dans le document hiérarchique de sortie.
- Correspondance ou ajout. Si un élément correspondant existe dans le document hiérarchique de sortie, la transformation Processeur de données n'ajoute pas d'élément de sortie. Si l'élément n'existe pas dans le document hiérarchique de sortie, la transformation crée un élément de sortie.

Nom

Facultatif. Un nom pour l'instruction. Vous pouvez changer le nom à tout moment. Le nom identifie les instructions de sorte que vous pouvez les trouver dans la grille de mappage ou dans un journal des événements. Les noms d'instruction ne doivent pas obligatoirement être uniques.

En cas d'échec

Facultatif. Détermine l'action engagée en cas d'échec de l'instruction. Choisissez l'une des options suivantes :

- Ignorer. Si l'instruction échoue, ignorer l'instruction.
- Propager. Si l'instruction échoue, faire aussi échouer l'instruction parent.

Sortie

Obligatoire. Une expression XPath qui définit la valeur de l'élément dans le format hiérarchique de sortie en fonction des résultats de l'expression XPath Entrée. Le champ de sortie fournit le contexte pour les instructions enfant.

Type d'instruction

Obligatoire. Identifie l'instruction comme une instruction Routeur.

Instructions Option

L'instruction Option fournit la condition pour mapper le nœud d'entrée au nœud de sortie. Une instruction Option doit être imbriquée en dessous d'une instruction Routeur. L'instruction Routeur doit avoir une expression XPath Entrée ou une expression XPath Condition. Ou l'instruction Option peut inclure une expression XPath Entrée et une expression XPath Condition.

La transformation Processeur de données accepte une instruction Option lorsque les résultats de l'expression de champ d'entrée et une expression de champ de condition s'évaluent sur un nœud unique. Si

vous ne définissez pas d'expression de champ d'entrée, la transformation Processeur de données accepte l'instruction Option lorsque l'expression de champ de condition s'évalue à « True ».

L'instruction Option peut contenir une ou plusieurs instructions enfant de tout type, notamment Mappage, Groupe, Groupe de répétition, Exécuter XMap et d'autres instructions Routeur. Par exemple, une instruction Option peut contenir la condition suivante :

```
EmployeeID="100"
```

Quand EmployeeID est égal à 100, la condition est vraie. L'instruction enfant dans la grille définit l'instruction de mappage à évaluer quand la condition est vraie.

Propriétés de l'instruction Option

L'instruction Option contient des propriétés que vous pouvez configurer pour personnaliser l'instruction. Vous pouvez configurer l'entrée, la sortie et une condition pour le mappage d'un élément d'entrée à un élément de sortie.

L'instruction Option comprend les propriétés suivantes :

Condition

Obligatoire si l'entrée n'est pas définie. Une expression XPath qui définit une condition de mappage de l'élément. Une condition est similaire à une expression de prédicat dans la colonne Entrée. La transformation Processeur de données applique la condition XPath au résultat de l'entrée XPath.

Entrée

Obligatoire si la condition n'est pas définie. Une expression XPath qui définit l'élément d'entrée. L'expression peut s'évaluer sur un nœud ou une valeur.

Mode

Facultatif. Détermine si la transformation Processeur de données ajoute un élément de sortie ou fait correspondre un élément existant avec une valeur depuis une instruction de mappage. Choisissez l'une des options suivantes :

- Ajouter. Crée un élément dans le document hiérarchique de sortie. Si l'élément n'est pas à occurrences multiples et que les mêmes valeurs existent dans la sortie, l'instruction de mappage échoue.
- Correspondance. L'instruction attend pour trouver une correspondance pour l'élément dans les éléments de sortie. L'instruction échoue si l'élément n'existe pas dans le document hiérarchique de sortie.
- Correspondance ou ajout. Si un élément correspondant existe dans le document hiérarchique de sortie, la transformation Processeur de données n'ajoute pas d'élément de sortie. Si l'élément n'existe pas dans le document hiérarchique de sortie, la transformation crée un élément de sortie.

Nom

Facultatif. Un nom pour l'instruction. Vous pouvez changer le nom à tout moment. Le nom identifie les instructions de sorte que vous pouvez les trouver dans la grille de mappage ou dans un journal des événements. Les noms d'instruction ne doivent pas obligatoirement être uniques.

En cas d'échec

Facultatif. Détermine l'action engagée en cas d'échec de l'instruction. Choisissez l'une des options suivantes :

- Ignorer. Si l'instruction échoue, ignorer l'instruction.
- Propager. Si l'instruction échoue, faire aussi échouer l'instruction parent.

Sortie

Facultatif. Une expression XPath qui définit la valeur de l'élément dans le format hiérarchique de sortie en fonction des résultats de l'expression XPath Entrée.

Type d'instruction

Obligatoire. Identifie l'instruction comme une instruction Option.

Instructions par défaut

Une instruction par défaut est une instruction enfant d'une instruction Routeur. L'instruction Routeur contient une ou plusieurs instructions Option et peut contenir une instruction par défaut. La transformation Processeur de données effectue l'instruction par défaut lorsqu'aucune instruction Option ne s'applique.

Vous ne pouvez définir qu'une seule instruction par défaut dans un groupe d'instructions Routeur. L'instruction par défaut doit être la dernière instruction pour le groupe d'instructions Routeur. L'instruction par défaut ne peut pas avoir d'expressions XPath d'entrée ou de condition.

Propriétés de l'instruction par défaut

L'instruction par défaut contient des propriétés que vous pouvez configurer pour personnaliser l'instruction. Vous pouvez configurer la valeur par défaut si un élément d'entrée est manquant.

L'instruction par défaut comprend les propriétés suivantes :

Par défaut

Obligatoire. La valeur par défaut à utiliser quand un élément est manquant dans l'entrée. Par exemple, vous pouvez définir une valeur par défaut pour initialiser un compteur.

Mode

Facultatif. Détermine si la transformation Processeur de données ajoute un élément de sortie ou fait correspondre un élément existant avec une valeur depuis une instruction de mappage. Choisissez l'une des options suivantes :

- Ajouter. Crée un élément dans le document hiérarchique de sortie. Si l'élément n'est pas à occurrences multiples et que les mêmes valeurs existent dans la sortie, l'instruction de mappage échoue.
- Correspondance. L'instruction attend pour trouver une correspondance pour l'élément dans les éléments de sortie. L'instruction échoue si l'élément n'existe pas dans le document hiérarchique de sortie.
- Correspondance ou ajout. Si un élément correspondant existe dans le document hiérarchique de sortie, la transformation Processeur de données n'ajoute pas d'élément de sortie. Si l'élément n'existe pas dans le document hiérarchique de sortie, la transformation crée un élément de sortie.

Nom

Facultatif. Un nom pour l'instruction. Vous pouvez changer le nom à tout moment. Le nom identifie les instructions de sorte que vous pouvez les trouver dans la grille de mappage ou dans un journal des événements. Les noms d'instruction ne doivent pas obligatoirement être uniques.

Sortie

Facultatif. Une expression XPath qui définit la valeur du nœud dans le format hiérarchique de sortie en fonction des résultats de l'expression XPath Entrée.

Type d'instruction

Obligatoire. Identifie l'instruction comme une instruction par défaut.

Instructions d'exécution XMap

Une instruction Exécuter XMap appelle une autre XMap.

Lorsque vous créez une instruction de mappage Exécuter XMap, l'outil Developer répertorie les objets XMap de la transformation. Sélectionnez la XMap à appeler. L'outil Developer crée une instruction de mappage avec le nom XMap dans le champ **Type d'instruction**.

Les éléments racine d'entrée et de sortie dans la XMap appelée doivent être du même type que les valeurs d'entrée et de sortie qui lui sont transmises depuis la XMap d'appel. Vous pouvez appeler une XMap pour exécuter la logique de mappage qui est répétée.

Propriétés de l'instruction Exécuter XMap

L'instruction Exécuter XMap contient des propriétés que vous pouvez configurer pour personnaliser l'instruction. Vous pouvez configurer l'entrée, la sortie et une condition pour le mappage d'un élément d'entrée à un élément de sortie.

L'instruction Exécuter XMap comprend les propriétés suivantes :

Condition

Facultatif. Une expression XPath qui définit une condition de mappage de l'élément. Une condition est similaire à une expression de prédicat dans la colonne Entrée. Si vous définissez une expression XPath Entrée et une expression XPath Condition pour la même instruction de mappage, la transformation Processeur de données applique la Condition XPath au résultat de l'expression XPath Entrée.

Entrée

Facultatif. Une expression XPath qui s'évalue à une séquence de nœuds ou de valeurs. Le type d'instruction de mappage détermine la manière dont la transformation Processeur de données utilise les nœuds ou les valeurs dans le mappage.

Mode

Obligatoire. Détermine si la transformation Processeur de données ajoute un élément de sortie ou fait correspondre un élément existant avec une valeur depuis une instruction de mappage. Choisissez l'une des options suivantes :

- Ajouter. Crée un élément dans le document hiérarchique de sortie. Si l'élément n'est pas à occurrences multiples et que les mêmes valeurs existent dans la sortie, l'instruction de mappage échoue.
- Correspondance. L'instruction attend pour trouver une correspondance pour l'élément dans les éléments de sortie. L'instruction échoue si l'élément n'existe pas dans le document hiérarchique de sortie.
- Correspondance ou ajout. Si un élément correspondant existe dans le document hiérarchique de sortie, la transformation Processeur de données n'ajoute pas d'élément de sortie. Si l'élément n'existe pas dans le document hiérarchique de sortie, la transformation crée un élément de sortie.

Nom

Facultatif. Un nom pour l'instruction. Vous pouvez changer le nom à tout moment. Le nom identifie les instructions de sorte que vous pouvez les trouver dans la grille de mappage ou dans un journal des événements. Les noms d'instruction ne doivent pas obligatoirement être uniques.

En cas d'échec

Obligatoire. Détermine l'action engagée en cas d'échec de l'instruction. Choisissez l'une des options suivantes :

- Ignorer. Si l'instruction échoue, ignorer l'instruction.
- Propager. Si l'instruction échoue, faire aussi échouer l'instruction parent.

Sortie

Facultatif. Une expression XPath qui définit la valeur du nœud dans le format hiérarchique de sortie en fonction des résultats de l'expression XPath Entrée.

Ignorer l'entrée manquante

Obligatoire. Détermine si l'instruction doit être ignorée s'il n'y a pas de correspondance avec la valeur Entrée. Choisissez l'une des options suivantes :

- Activé. Si l'élément ne se trouve pas dans le document hiérarchique d'entrée, la transformation Processeur de données ignore l'instruction sans erreur.
- Désactivé. L'instruction échoue lorsque l'élément ne se trouve pas dans le document hiérarchique d'entrée.

Type d'instruction

Obligatoire. Identifie l'instruction comme une instruction Exécuter XMap.

Instruction RunMaplet

Une instruction RunMaplet appelle un maplet.

Lorsque vous créez une instruction de mappage RunMaplet, l'outil Developer tool répertorie les objets de référence de maplet associés à la transformation. Sélectionnez le maplet à appeler. L'outil Developer tool crée une instruction XMap avec le nom du maplet dans le champ **Type d'instruction**.

Les ports d'entrée et de sortie du maplet appelé doivent être du même type que les valeurs transmises par l'instruction XMap d'appel. Vous pouvez appeler un maplet pour effectuer des tâches telles que le masquage des données, la qualité des données, la recherche des données et d'autres activités généralement liées à une transformation, sans convertir les données dans un format relationnel, puis restaurer un format hiérarchique.

Remarque: L'action RunMaplet peut être utilisée pour appeler des maplets passifs uniquement.

Propriétés de l'instruction RunMaplet

L'instruction RunMaplet contient des propriétés que vous pouvez configurer pour la personnaliser. Vous pouvez configurer l'entrée, la sortie et une condition pour l'exécution de l'instruction RunMaplet. L'instruction RunMaplet peut contenir les instructions MapletInput et MapletOutput.

L'instruction RunMaplet comprend les propriétés suivantes :

Condition

Facultatif. Expression XPath qui définit une condition pour l'exécution de l'instruction RunMaplet. Une condition est similaire à une expression de prédicat dans la colonne Entrée. Si vous définissez une expression XPath d'entrée et une expression XPath de condition pour la même instruction de mappage, la transformation Processeur de données applique l'expression XPath de condition au résultat de l'expression XPath d'entrée.

Entrée

Facultatif. Expression XPath qui renvoie une séquence de nœuds ou de valeurs. Le type d'instruction de mappage détermine la manière dont la transformation Processeur de données utilise les nœuds ou les valeurs dans le mappage.

Mode

Obligatoire. Détermine si la transformation Processeur de données ajoute un élément de sortie ou fait correspondre un élément existant avec une valeur d'une instruction de mappage. Choisissez l'une des options suivantes :

- Ajouter. Crée un élément dans le document hiérarchique de sortie. Si l'élément n'est pas à occurrences multiples et que les mêmes valeurs existent dans la sortie, l'instruction de mappage échoue.
- Correspondance. L'instruction s'attend à trouver une correspondance pour l'élément dans les éléments de sortie. L'instruction échoue si l'élément n'existe pas dans le document hiérarchique de sortie.
- Correspondance ou ajout. Si un élément correspondant existe dans le document hiérarchique de sortie, la transformation Processeur de données n'ajoute pas d'élément de sortie. Si l'élément n'existe pas dans le document hiérarchique de sortie, la transformation crée un élément de sortie.

Nom

Facultatif. Un nom pour l'instruction. Vous pouvez changer le nom à tout moment. Le nom identifie les instructions de sorte que vous pouvez les trouver dans la grille de mappage ou dans un journal des événements. Les noms d'instruction ne doivent pas obligatoirement être uniques.

En échec

Facultatif. Détermine l'action engagée en cas d'échec de l'instruction. Choisissez l'une des options suivantes :

- Ignorer. Si l'instruction échoue, ignorer l'instruction.
- Propager. Si l'instruction échoue, faire aussi échouer l'instruction parent.

Sortie

Obligatoire. Expression XPath qui définit la valeur du nœud dans la hiérarchie de sortie en fonction des résultats de l'expression XPath d'entrée.

Ignorer l'entrée manquante

Facultatif. Détermine si l'instruction doit être ignorée en l'absence de correspondance avec la valeur d'entrée. Choisissez l'une des options suivantes :

- Activé. Si l'élément ne se trouve pas dans le document hiérarchique d'entrée, la transformation Processeur de données ignore l'instruction sans erreur.
- Désactivé. L'instruction échoue lorsque l'élément ne se trouve pas dans le document hiérarchique d'entrée.

Type d'instruction

Obligatoire. Identifie l'instruction comme une instruction RunMapplet. Le champ identifie le nom du mapplet référencé.

Instruction MappletInput

L'instruction MappletInput contient des propriétés que vous pouvez configurer pour la personnaliser. Vous pouvez configurer l'entrée pour mapper un élément d'entrée au port d'entrée du mapplet. Une ou plusieurs instructions MappletInput peuvent être imbriquées dans l'instruction RunMapplet.

L'instruction `MappletInput` est exécutée pour fournir une valeur à transmettre à l'entrée du mapplet. Les valeurs de l'instruction `RunMapplet` sont transmises aux ports du mapplet dans l'ordre dans lequel elles apparaissent dans l'instruction `RunMapplet`. Si une instruction est ignorée, une valeur `Null` est transmise au port d'entrée de mapplet.

Une instruction `MappletInput` transmet une valeur unique. Lorsque l'instruction `RunMapplet` est exécutée, les valeurs des instructions `MappletInput` imbriquées sont collectées et transmises au mapplet dans le même ordre que les instructions `MappletInput`.

Propriétés de l'instruction `MappletInput`

L'instruction `MappletInput` contient des propriétés que vous pouvez configurer pour la personnaliser.

L'instruction `RunMapplet` comprend les propriétés suivantes :

Valeur par défaut

Facultatif. Valeur par défaut à utiliser lorsqu'un élément est manquant dans l'entrée du port du mapplet.

Entrée

Obligatoire. Expression XPath qui renvoie une séquence de nœuds ou de valeurs. Les valeurs sont transmises aux ports d'entrée du mapplet.

Nom

Facultatif. Un nom pour l'instruction. Vous pouvez changer le nom à tout moment. Le nom identifie les instructions de sorte que vous pouvez les trouver dans la grille de mappage ou dans un journal des événements. Les noms d'instruction ne doivent pas obligatoirement être uniques.

Ignorer l'entrée manquante

Facultatif. Détermine si l'instruction doit être ignorée lorsqu'aucun port d'entrée de mapplet ne correspond à la valeur d'entrée. Choisissez l'une des options suivantes :

- **Activé.** Si l'élément ne se trouve pas dans le document hiérarchique d'entrée, la transformation `Processeur de données` ignore l'instruction sans erreur.
- **Désactivé.** L'instruction échoue lorsque l'élément ne se trouve pas dans le document hiérarchique d'entrée.

Instruction `MappletOutput`

L'instruction `MappletOutput` est exécutée pour obtenir une valeur transmise par la sortie du mapplet. Les valeurs sont transmises des ports du mapplet aux ports de sortie de l'instruction `RunMapplet` dans l'ordre dans lequel elles apparaissent dans l'instruction `RunMapplet`. Une ou plusieurs instructions `MappletOutput` peuvent être imbriquées dans l'instruction `RunMapplet`.

Une instruction `MappletOutput` obtient une valeur unique. Lorsque l'instruction `RunMapplet` s'exécute, les valeurs du mapplet sont collectées dans les instructions `MappletOutput` imbriquées dans le même ordre que les instructions `MappletOutput`.

Propriétés de l'instruction `MappletOutput`

L'instruction `MappletOutput` contient des propriétés que vous pouvez configurer pour la personnaliser.

L'instruction `RunMapplet` comprend les propriétés suivantes :

Valeur par défaut

Facultatif. Valeur par défaut à utiliser pour tester la transformation `Processeur de données` sans entrée de mappage. La valeur par défaut n'est pas utilisée lorsqu'il existe une entrée de mappage.

Mode

Facultatif. Détermine si la transformation ajoute un élément de sortie ou fait correspondre un élément existant avec une valeur d'une instruction de mappage. Choisissez l'une des options suivantes :

- Ajouter. Crée un élément dans le document hiérarchique de sortie. Si l'élément n'est pas à occurrences multiples et que les mêmes valeurs existent dans la sortie, l'instruction de mappage échoue.
- Correspondance. L'instruction s'attend à trouver une correspondance pour l'élément dans les éléments de sortie. L'instruction échoue si l'élément n'existe pas dans le document hiérarchique de sortie.
- Correspondance ou ajout. Si un élément correspondant existe dans le document hiérarchique de sortie, la transformation Processeur de données n'ajoute pas d'élément de sortie. Si l'élément n'existe pas dans le document hiérarchique de sortie, la transformation crée un élément de sortie.

Nom

Facultatif. Un nom pour l'instruction. Vous pouvez changer le nom à tout moment. Le nom identifie les instructions de sorte que vous pouvez les trouver dans la grille de mappage ou dans un journal des événements. Les noms d'instruction ne doivent pas obligatoirement être uniques.

Sortie

Obligatoire. Expression XPath qui renvoie une séquence de nœuds ou de valeurs. Les valeurs du port de sortie du mapplet sont transmises à la séquence de nœuds.

Création d'une XMap

Pour créer un objet XMap du Processeur de données, choisissez l'entrée et la sortie des schémas et ajoutez des instructions de mappage.

1. Sur la vue **Objets** de la transformation Processeur de données, créez une XMap. Sélectionnez une entrée de schéma, une source d'exemple et un schéma de sortie.
2. Pour ouvrir l'éditeur XMap, cliquez sur l'objet XMap.
3. Pour créer une instruction de mappage Mappage, Groupe ou Groupe de répétition, dans l'éditeur XMap, effectuez un glisser-déplacer depuis un nœud du schéma hiérarchique d'entrée vers un nœud du schéma hiérarchique de sortie.
L'éditeur XMap crée un lien de mappage entre les nœuds. L'instruction de mappage s'affiche dans la grille. L'éditeur XMap remplit automatiquement les champs d'instruction de mappage.
4. Pour créer la logique conditionnelle dans la grille, ajoutez une instruction de mappage Routeur comme suit :
 - a. Dans l'instruction de mappage Routeur, créez des instructions de mappage Option. Déplacez et faites glisser les nœuds de schémas d'entrée et de sortie dans les champs d'instruction Option de la grille.
 - b. Dans l'instruction de mappage Routeur, créez une instruction de mappage par défaut pour spécifier ce qu'il se passe si aucune instruction de mappage Option ne s'applique.
 - c. Dans les instructions de mappage Option, créez des instructions de mappage Mappage pour spécifier les conditions pour mapper le nœud d'entrée au nœud de sortie.
5. Pour fournir un contexte standard pour un groupe d'instructions, ajoutez une instruction de mappage Groupe. Imbriquer les instructions de mappage Mappage dans l'instruction de mappage Groupe.
6. Pour appeler un autre objet XMap, ajoutez une instruction Exécuter XMap.

7. Pour modifier le contexte et la logique pour une instruction de mappage, éditez les propriétés de l'instruction de mappage comme suit :
 - a. Rétrograder les instructions à des instructions enfant ou avancer les instructions en instructions parent.
 - b. Créer des expressions XPath pour changer le contexte ou ajouter des prédicats à l'aide de l'éditeur XPath.

Utilisation de la grille de l'éditeur XMap

Lorsque vous faites glisser un nœud depuis le schéma d'entrée vers le schéma de sortie, l'outil Developer crée une instruction de mappage Mappage, Groupe ou Groupe de répétition dans la grille. Vous pouvez mettre à jour l'instruction de mappage. Les champs d'entrée et de sortie contiennent les éléments issus du schéma. Si vous voulez créer des instructions de mappage pour fournir le contexte ou pour définir les options du routeur, vous pouvez saisir les instructions dans la grille.

Lorsque vous sélectionnez une instruction de mappage dans la grille, l'éditeur XMap met en évidence les nœuds des schémas d'entrée et de sortie qui sont dans l'instruction de la grille.

Vous pouvez copier une instruction d'une ligne à une autre dans la grille. Si la ligne n'est pas valide pour l'emplacement dans lequel vous la copiez, l'éditeur XMap affiche une boîte de dialogue avec une erreur de validation. Vous pouvez modifier les instructions XPath d'entrée et de sortie dans la boîte de dialogue afin d'ajuster le contexte de l'instruction de mappage ou vous pouvez modifier les champs XPath d'entrée et de sortie dans la grille.

Création des instructions de mappage

Créer des instructions de mappage dans l'éditeur XMap. Vous pouvez créer des instructions de mappage en déplaçant des nœuds depuis le schéma d'entrée vers le schéma de sortie et vous pouvez définir les instructions dans la grille d'instruction de mappage.

Utilisez les étapes suivantes pour définir les instructions de mappage dans la grille :

1. Pour créer une instruction de mappage, dans les options de la grille, cliquez sur **Nouveau**.
2. Sélectionnez le type d'instruction de mappage dans la liste.

Si vous choisissez **Exécuter XMap**, l'outil Developer affiche une liste des objets XMap dans la transformation.
3. Entrez un nom pour l'instruction de mappage.
4. Pour définir l'entrée de l'instruction de mappage, faites glisser un élément depuis le schéma d'entrée dans le champ Entrée. Vous pouvez également configurer une expression XPath ou une constante dans le champ Entrée.
5. Pour sélectionner le nœud de sortie de l'instruction de mappage, faites glisser un élément depuis le schéma de sortie dans le champ Sortie. Vous pouvez également configurer une expression XPath dans le champ Sortie.
6. Pour créer une expression XPath à l'aide de l'éditeur d'expression XPath pour un champ Entrée, Sortie ou Condition, cliquez sur le bouton **Ouvrir** de ce champ.
7. Pour changer le type d'instruction de mappage, cliquez sur le bouton **Ouvrir** dans le champ **Type d'instruction**. Choisissez le type d'instruction de mappage dans la liste.

Interface de la grille des instructions de mappage

Éditer les instructions de mappage dans la grille d'instruction de mappage de l'éditeur XMap. Vous devez créer une instruction avant de pouvoir modifier les champs de la grille d'instruction de mappage.

Vous pouvez effectuer les tâches suivantes dans la grille d'instruction de mappage :

- Faites glisser les nœuds depuis le schéma d'entrée ou de sortie vers les champs dans les instructions de mappage.
- Copiez les éléments dans les champs d'instruction de mappage ou saisissez les valeurs dans les champs.
- Vous pouvez déplacer une instruction de mappage vers le haut ou vers le bas dans la grille. Sélectionnez la ligne et cliquez sur les flèches d'option **Haut** ou **Bas**.
- Vous pouvez cliquer sur **Rétrograder** pour mettre en retrait une instruction de mappage, sous une autre instruction. Cliquez sur **Avancer** si vous voulez déplacer une instruction vers le haut dans la hiérarchie.
- Cliquez sur **Atteindre la ligne numéro** pour naviguer vers une ligne. Entrez la ligne vers laquelle vous voulez naviguer. L'outil Developer met en évidence la ligne pour vous.

Liaison des nœuds de schéma

Vous pouvez lier un nœud de schéma d'entrée à un nœud de schéma de sortie en le déplaçant avec la souris. Vous pouvez utiliser le glisser-déposer pour lier un nœud simple à un autre nœud simple, un nœud simple à un nœud complexe, un nœud complexe à un nœud simple, ou un nœud complexe à un autre nœud complexe.

L'éditeur XMap crée un lien de mappage entre le nœud de schéma d'entrée et le nœud de schéma de sortie. L'éditeur XMap crée également une instruction de mappage dans la grille.

Vous pouvez faire glisser et déplacer une instruction de mappage depuis une ligne de la grille vers une autre ligne pour changer la logique de la XMap. Par exemple, vous pouvez utiliser cette méthode pour modifier la séquence des instructions Option dans un Routeur.

Couper et coller les instructions de mappage

Vous pouvez couper et coller les instructions de mappage dans la grille d'instruction de mappage dans l'éditeur XMap.

Vous pouvez déplacer les instructions vers le haut ou vers le bas. Vous pouvez coller les instructions de mappage même dans les instructions imbriquées. Vous pouvez avancer une instruction pour être une instruction parent ou rétrograder une instruction à une instruction enfant.

Les instructions collées doivent généralement être corrigées car leur logique est souvent hors contexte. Après avoir collé une instruction, vous pouvez modifier les champs dans la grille d'instruction de mappage.

Expressions XPath

Les expressions XPath identifient des éléments ou des nœuds spécifiques dans des documents hiérarchiques ou vérifient des conditions dans les données. Utilisez des expressions XPath pour définir les champs Entrée, Condition ou Sortie d'une instruction de mappage.

XPath est une syntaxe qui définit les parties d'un document hiérarchique. Utilisez XPath pour sélectionner les séquences de nœuds ou les valeurs dans un document hiérarchique. XPath inclut une bibliothèque de fonctions standard que vous pouvez utiliser pour sélectionner des données.

Vous pouvez définir des expressions XPath 2.0 dans la transformation Processeur de données. Lorsque vous configurez les expressions XPath de sortie, vous pouvez utiliser un sous-ensemble de la syntaxe XPath 2.0 lorsque vous définissez des instructions de mappage pour le mode Ajouter ou le mode Correspondre ou Ajouter.

Pour plus d'informations sur XPath, consultez votre documentation XPath.

Le tableau suivant décrit certaines expressions XPath :

Expression XPath	Description
nom du nœud	Sélectionne tous les nœuds enfant portant le nom donné dans le contexte.
. (un point)	Sélectionne le nœud actuel.
..	Sélectionne le parent du nœud actuel.
@	Sélectionne l'attribut.
/	Sélectionne depuis le nœud racine ou un enfant du nœud actuel si précédé d'un nœud. Lorsque le chemin commence par une barre oblique (/), cela représente un chemin absolu vers un élément.
//	Sélectionne les nœuds à tout emplacement dans le document ou les descendants du nœud actuel si précédé d'un nœud.

Le tableau suivant présente certaines expressions XPath et le résultat de chaque expression :

Expression XPath	Résultat
/librairie	Sélectionne le nœud racine librairie.
librairie/livre	Sélectionne les nœuds livre qui sont des enfants de tous les nœuds librairie.
//livre	Sélectionne les nœuds livre dans le document dans tous les emplacements.
librairie//livre	Sélectionne tous les nœuds livre qui sont descendants de nœuds librairie.
/librairie/*	Sélectionne tous les nœuds enfant de l'élément racine librairie.
//*	Renvoie une séquence de tous les éléments dans le document.

Prédicats

Un prédicat est une expression que vous configurez pour rechercher un nœud dans un document hiérarchique. Vous pouvez configurer l'expression afin de trouver une valeur particulière. Créez un prédicat dans un champ Entrée, Condition ou Sortie d'une instruction de mappage.

Lorsque vous définissez un prédicat, placez l'expression entre crochets [] après le nœud.

```
<nœud >[expression]
```

Par exemple, l'expression suivante sélectionne les éléments du livre qui sont descendants de la librairie et ont un élément de prix avec une valeur supérieure à 55.00 :

```
/librairie/livre[prix>55.00]
```


L'expression suivante sélectionne les éléments de titre des éléments du livre qui sont descendants de la librairie avec une valeur d'élément de prix supérieure à 55.00 :

```
/librairie/livre[prix>55.00]/title
```

L'expression suivante sélectionne les éléments de titre qui ont un attribut lang avec une valeur "eng" :

```
//title[@lang="eng"]
```

Remarque: La transformation Processeur de données ne peut pas accepter toutes les instructions XPath dans le champ Sortie lorsque vous configurez une instruction de mappage avec le mode Ajouter ou Correspondre ou ajouter.

Référence des prédicats XPath

Les prédicats XPath trouvent un nœud correspondant ou une séquence de nœuds dans un document hiérarchique. Une expression de prédicat définit la valeur d'un champ Entrée, Condition ou Sortie d'une instruction de mappage. L'expression XPath détermine le contexte dans lequel une instruction de mappage est exécutée.

Utilisez les expressions XPath suivantes dans les prédicats pour sélectionner un nœud ou une séquence de nœuds :

ancêtre

Sélectionne tous les ancêtres, comme par exemple le parent ou le grand-parent, du nœud actuel. Par exemple, l'expression de prédicat "ancestor::book" sélectionne tous les ancêtres livre du nœud actuel.

ancêtre-ou-réflexif

Sélectionne tous les ancêtres, comme par exemple le parent ou le grand-parent, du nœud actuel, ainsi que le nœud actuel. Par exemple, l'expression de prédicat "ancestor-or-self::book" sélectionne tous les ancêtres livre du nœud actuel et le nœud actuel également.

attribut

Sélectionne tous les attributs du nœud actuel. Par exemple, l'expression de prédicat "attribute::lang" sélectionne l'attribut langue du nœud en cours.

enfant

Sélectionne tous les enfants du nœud actuel. Par exemple, l'expression de prédicat "child::book" sélectionne tous les nœuds livre qui sont des enfants du nœud actuel.

descendant

Sélectionne tous les descendants, tels que les enfants ou petits-enfants, du nœud actuel. Par exemple, l'expression de prédicat "descendant::book" sélectionne tous les descendants livre du nœud actuel.

descendant-ou-réflexif

Sélectionne tous les descendants, tels que les enfants ou petits-enfants, du nœud actuel, ainsi que le nœud actuel. Par exemple, l'expression de prédicat "descendant-or-self::book" sélectionne tous les descendants livre du nœud actuel, et le nœud actuel aussi s'il s'agit d'un nœud livre.

suivant

Sélectionne tout dans le document après la balise de fermeture du nœud actuel. Par exemple, l'expression de prédicat "following::book" sélectionne dans le document après la balise de fermeture du nœud livre.

frères et sœurs suivants

Sélectionne tous les frères et sœurs suivants, après le nœud actuel. Par exemple, l'expression de prédicat "following-sibling::book" sélectionne tous les frères et sœurs du document après le nœud livre.

espace de nom

Sélectionne tous les nœuds d'espace de nom du nœud actuel.

parent

Sélectionne le parent du nœud actuel. Par exemple, l'expression de prédicat "parent::book" sélectionne l'attribut langue du nœud actuel.

précédent

Sélectionne tous les nœuds qui s'affichent avant le nœud actuel dans le document, sauf les ancêtres, les nœuds d'attribut et les nœuds d'espace de nom. Par exemple, l'expression de prédicat "preceding::book" sélectionne les nœuds avant le nœud livre.

frères et sœurs précédents

Sélectionne tous les frères et sœurs qui s'affichent avant le nœud actuel dans le document. Par exemple, l'expression de prédicat "preceding-sibling::book" sélectionne les nœuds frères et sœurs avant le nœud livre.

self

Sélectionne le nœud actuel. Par exemple, l'expression de prédicat "self::book" sélectionne le nœud livre actuel.

Opérateurs arithmétiques XPath

Pour effectuer les calculs, ajoutez des opérateurs arithmétiques qui évaluent les nœuds des documents hiérarchiques. Vous pouvez ajouter des opérateurs arithmétiques à des expressions XPath dans les champs Entrée, Condition ou Sortie d'une instruction de mappage.

Le tableau suivant décrit les opérateurs arithmétiques XPath que vous pouvez utiliser dans les expressions XMap :

Expression XPath	Description
	Sélectionne deux ensembles de nœuds dans le contexte. Par exemple, l'expression de prédicat "//book //cd" renvoie un ensemble de nœuds avec tous les éléments de livre et de CD.
+	Ajoute les éléments. Par exemple, l'expression de prédicat "1+2" renvoie 3.
-	Soustrait les éléments. Par exemple, l'expression de prédicat "2-1" renvoie 1.
*	Multiplie des éléments. Par exemple, l'expression de prédicat "2*1" renvoie 2.
div	Divise les éléments. Par exemple, l'expression de prédicat "6 div 3" renvoie 2.
=	Sélectionne les éléments qui sont égaux à l'expression. Par exemple, l'expression de prédicat "cost=1.50" renvoie « True » si le coût est 1,50 et « False » si le coût est 1,60.
!=	Sélectionne les éléments qui ne sont pas égaux à l'expression. Par exemple, l'expression de prédicat "cost!=1.50" renvoie « True » si le coût est 1,60, et « False » si le coût est 1,50.
<	Sélectionne les éléments qui sont inférieurs à l'expression. Par exemple, l'expression de prédicat "taxe<1.50" renvoie « True » si la taxe est 1,00 et « False » si la taxe est 1,50.
<=	Sélectionne les éléments qui sont égaux ou inférieurs à l'expression. Par exemple, l'expression de prédicat "taxe<=1.50" renvoie « True » si la taxe est 1,50 et « False » si la taxe est 1,80.

Expression XPath	Description
>	Sélectionne les éléments supérieurs à l'expression. Par exemple, l'expression de prédicat "taxe>1.50" renvoie « True » si la taxe est 1,90 et « False » si la taxe est 1,50.
>=	Sélectionne les éléments qui sont supérieurs ou égaux à l'expression. Par exemple, l'expression de prédicat "taxe>=1.50" renvoie « True » si la taxe est 1,50 et « False » si la taxe est 1,00.
ou	Sélectionne les éléments qui peuvent satisfaire une ou plusieurs conditions. Par exemple, l'expression de prédicat "taxe=1.50 or taxe=1.70" renvoie « True » si la taxe est 1,50 et « False » si la taxe est 1,00.
et	Sélectionne les éléments qui peuvent satisfaire toutes les conditions données. Par exemple, l'expression de prédicat "prix>1.00 and price<1.90" renvoie « True » si le prix est 1,50 et « False » si le prix est 1,00.
mode	Effectue la division et donne le reste. Par exemple, l'expression de prédicat "3 mod 2" renvoie 1.

Expressions XPath de sortie

La transformation Processeur de données accepte un sous-ensemble d'instructions XPath dans le champ Sortie lorsque le champ Mode est défini sur **Ajouter** ou **Correspondre ou ajouter**. Lorsque vous sélectionnez ces paramètres de mode, la transformation Processeur de données crée les éléments nécessaires pour correspondre à l'expression XPath dans le champ Sortie.

Vous pouvez utiliser une simple expression XPath dans le champ Sortie. Une expression simple comporte des axes enfant, des axes parent ou des variables. Les expressions simples n'ont pas de prédicats, de fonctions ou d'axes complexes. Par exemple, vous pouvez utiliser les expressions du champ Sortie suivantes :

```

personne/données
/racine/ceo/nom
$var/name
personne/../../ceo

```

Vous pouvez utiliser un simple prédicat avec cardinalité pour un élément ayant plusieurs instances. Par exemple, vous pouvez utiliser l'expression du champ Sortie suivante :

```

person/phone[4]

```

Vous pouvez utiliser un simple prédicat avec une formule contenant un signe d'égalité, avec les XPaths simples situés à gauche du signe d'égalité. Par exemple, vous pouvez utiliser les expressions du champ Sortie suivantes :

```

Personne[id=10]
Personne[id=$id]
Personne[id=@dp:input()/ID]
Société[name=upper-case($compName)]
Personne[role="manager" and id=1]

```

Vous pouvez également utiliser la combinaison d'une expression simple avec cardinalité et d'une formule qui utilise un XPath simple sur le côté gauche du signe égal. Par exemple, vous pouvez utiliser les expressions du champ Sortie suivantes :

société[4]/détails[id=\$myid]/phone

Remarque: Lorsque le champ Mode est défini sur **Correspondre**, le champ Sortie peut également accepter des expressions XPath complexes.

Éditeur d'expression XPath

Créez des expressions dans l'éditeur d'expression XPath. XPath est un langage de requête utilisé pour sélectionner des nœuds dans un document hiérarchique et exécuter des calculs.

Vous pouvez utiliser les expressions XPath pour définir le contexte pour une instruction de mappage. Vous pouvez définir les conditions pour transformer et filtrer les données à l'aide de différentes expressions XPath dans les propriétés d'instruction de mappage. Vous pouvez ajouter divers calculs arithmétiques à une instruction de mappage à l'aide d'expressions XPath.

Lorsque vous cliquez sur le bouton Ouvrir dans le champ Entrée, Condition ou Sortie, l'Éditeur d'expressions s'affiche. La figure suivante présente l'Éditeur d'expression XPath :

Nom	Type	Min...	Nb...	Description
Employee	EmployeeT...	1	1	
id	xs:string	0	1	
FirstName	xs:string	0	1	
LastName	xs:string	0	1	
Role	xs:string	0	1	
StartDate	xs:date	0	1	
Variables	(Variables)	1	1	
\$deptName	xs:string	0	1	

☒ Afficher les lignes ☒ toujours ☐ selon la sélection

Nom	Type d'inst...	Entrée	Condition	Ignore...	Valeur...	En échec	Sortie	Mode
Employee to Worker	Groupe rép...			<input type="checkbox"/>		Propager		Correspon
FirstName to FirstName	Routeur	tns0:FirstName		<input checked="" type="checkbox"/>		Ignorer	tns0:FirstName	Correspon
LastName to LastName	Mapper	tns0:LastName		<input checked="" type="checkbox"/>		Ignorer	tns0:LastName	Correspon
Employee/@id to Id	Mapper	@id		<input type="checkbox"/>		Propager	tns0:Id	Correspon
StartDate to YearsOfService	Mapper	year-from-date(curr...		<input checked="" type="checkbox"/>		Ignorer	tns0:YearsOfService	Correspon

Créez des expressions dans le panneau **Expression**.

L'éditeur d'expression XPath présente un panneau **Navigation** et une bibliothèque de fonctions que vous pouvez utiliser pour créer des expressions XPath. Les fonctions sont des fonctions standard pour le langage W3C XML Path. La bibliothèque de fonctions inclut également des fonctions spécifiques à la transformation Processeur de données.

Fonctions du processeur de données

L'éditeur d'expression possède des fonctions Processeur de données que vous pouvez utiliser pour la transformation Processeur de données.

La transformation Processeur de données utilise les fonctions XPath suivantes :

dp:as_xml

Reçoit un nœud en entrée et renvoie la valeur du nœud et la valeur de tous les enfants en tant que chaîne XML de façon récursive. La fonction as_xml utilise la syntaxe suivante : `dp:as_xml(<node>)`

dp:get_id

Génère un identifiant unique associé à un nœud et le renvoie. Vous pouvez utiliser l'ID pour créer des relations de clé primaire-clé étrangère dans les données. Mappez l'identifiant sur un nœud du schéma et mappez-le sur des clés dans les données relationnelles. La fonction get_id utilise la syntaxe suivante :

```
dp:get_id(<node>)
```

dp:input

Renvoie le nœud qui fournit le contexte d'entrée actuel. Utilisez la fonction dans le champ Sortie pour faire référence à un nœud depuis le schéma d'entrée. La fonction d'entrée utilise la syntaxe suivante :

```
dp:input()
```

dp:transform

Appelez un transformateur de transformation Processeur de données que vous définissez dans un script. Vous pouvez effectuer une transformation intégrée ou externe. La fonction utilise la syntaxe suivante : dp:transform(<transform-name>,<transform-value>)

La fonction de transform utilise les paramètres suivants :

- Transform-name. Le nom du transformateur dans le script.
- Transform-value. La valeur de transformation avec laquelle effectuer la transformation.

Remarque: La fonction de recherche est disponible via l'utilisation de la fonction **dp:transform**.

dp:output

Renvoie le nœud qui fournit le contexte de sortie actuel. Utilisez la fonction dans le champ Entrée pour faire référence à un nœud dans le schéma de sortie. La fonction de sortie utilise la syntaxe suivante :

```
dp:output()
```

Exemple d'expressions XPath

Utilisez les expressions XPath dans les instructions de mappage. Les expressions XPath identifient les éléments dans le document d'entrée à mapper et à transformer dans le document de sortie. Une expression XPath s'utilise également pour effectuer une opération arithmétique.

La figure suivante présente les expressions XPath dans la grille :

Nom	Type	Min...	Nb...	Description
Children		1	1	
Child		1	1	Illu...
Name		1	1	
First	xs:string	1	1	
Last	xs:string	1	1	
Initial	xs:string	0	1	
Hobby	xs:string	1	1	Illu...
Variables	(Variables)	1	1	
\$var1	xs:string	0	1	

Nom	Type	Min...	Nb...	Description
Classes		1	1	
noOfClasses	xs:integer	0	1	
Class		1	1	Illu...
name	xs:string	0	1	
noOfChildren	xs:int	0	1	
Child	xs:string	1	1	Illu...
Variables	(Variables)	1	1	
\$var1	xs:string	0	1	

Nom	Type d'inst...	Entrée	Condition	Ignore...	Valeur...	En échec	Sortie	Mode	Ren
1 Children to Classes	Groupe rép...	Child				Propager		Ajouter	
2 Hobby to Class	Groupe rép...	Hobby				Propager	Class[@name = dp:i...	Correspondance ou ajout	
3 First to Child	Mapper	concat(../Name/First, ' ', ../Name/Initi...				Propager	Child	Ajouter	
4 Count Children in Class	Mapper	dp:output()/@noOfChildren + 1			1	Propager	@noOfChildren	Correspondance ou ajout	
5 Count # of Classes	Mapper	count(dp:output()/Class)				Propager	@noOfClasses	Correspondance ou ajout	

Le document d'entrée XMap est une liste d'enfants et de leurs hobbies. L'entrée racine est Children. Child est un élément à plusieurs occurrences au sein de Children. Chaque enfant (child) a un nom et plusieurs hobbies. Le nom est composé d'éléments First, Initial et Last, qui correspondent au prénom, aux initiales et au nom.

La sortie est une liste des classes avec le nombre d'enfants dans chaque classe. La racine de sortie est Classes. « Classes » a un attribut qui contient le nombre total de classes. Chaque élément d'entrée Hobby correspond à élément de sortie Class. Une instruction Mappage concatène les éléments First, Initial et Last dans l'élément de sortie Child. Une autre instruction Mappage compte le nombre d'enfants dans chaque classe. Une autre instruction compte le nombre de classes.

XMap contient les expressions suivantes :

Expression de la ligne 2 de la grille <Class[@name = dp:input()]>

Ajoute un élément Classe ou recherche une Classe qui correspond à Hobby. La fonction dp:input() est obligatoire car l'expression fait référence à un élément d'entrée.

Expression de la ligne 3 de la grille <concat(..Name/First,'',../Name/Initial,'',../Name/Last)>

Enchaîne le prénom, l'initiale du deuxième prénom et le nom de famille, et ajoute les espaces entre eux.

Expression de la ligne 4 de la grille <dp:output()/@noOfChildren + 1>

Pour chaque occurrence de Hobby, ajouter 1 au nombre d'enfants de cette classe. La fonction dp:output() est obligatoire car l'expression fait référence à un élément de sortie.

Expression de la ligne 5 de la grille <count(dp:output()/Class)>

Dénombrer les éléments Classe. La fonction dp:output() est obligatoire car l'expression fait référence à un élément de sortie.

Création d'une expression

Créer des expressions XPath dans l'**éditeur d'expressions**.

1. Dans l'instruction XMap, cliquez sur le bouton **Ouvrir** dans le champ **Entrée, Condition** ou **Sortie**. L'**éditeur d'expressions** s'affiche.
2. Pour ajouter un élément à une expression, double-cliquez sur les éléments dans le panneau **Navigation**.
3. Cliquez sur **Valider** pour valider l'expression.
4. Si l'expression est destinée au champ Entrée, cliquez sur **Tester l'expression** pour tester l'expression avec les données d'exemple.

Les résultats s'affichent chaque fois que l'outil Developer évalue l'expression à l'aide des données d'exemple. L'expression XPath peut renvoyer une séquence de zéros ou plusieurs nœuds ou valeurs. La **Longueur de séquence** indique le nombre de nœuds renvoyés par l'expression XPath.

Variables XMap

Vous pouvez ajouter des variables dans l'éditeur XMap. Vous pouvez mapper des valeurs à des variables et utiliser les variables dans des prédicats ou des zones de stockage temporaires pour les valeurs. Vous pouvez mapper des variables à des éléments de sortie.

Lorsque vous créez une variable, celle-ci s'affiche dans les schémas d'entrée et de sortie dans la vue XMap. L'outil Developer ajoute un dollar (\$) au nom de la variable pour indiquer qu'il s'agit d'une variable.

Vous pouvez créer une variable qui soit une liste de plusieurs valeurs. Vous pouvez utiliser une variable de liste pour atteindre le même objectif qu'un élément de schéma à plusieurs occurrences. Configurez une variable de liste en tant qu'entrée pour un groupe répétitif ou configurez un prédicat pour rechercher une valeur dans la variable de liste.

Par exemple, vous disposez d'un document XML qui contient des adresses. Vous devez créer une liste de tous les pays contenus dans les adresses. Mappez l'élément pays à une variable \$countries que vous définissez comme une liste.

Création d'une variable dans l'éditeur XMap

Vous pouvez créer des variables dans l'éditeur XMap.

1. Cliquez sur **Variables** au-dessus du schéma d'entrée ou de sortie dans l'éditeur XMap.
La boîte de dialogue **Variables** s'affiche.
2. Pour créer une variable, cliquez sur **Nouveau**.
3. Entrez un nom de variable et un type de données.
4. Activez **Liste** pour créer une variable à occurrences multiples.

Exemple XMap

Un document XML contient les données de l'employé comprenant le rôle de l'employé dans l'entreprise. Vous devez créer un document XML séparant le groupe des managers de celui des employés. Vous créez deux objets XMap dans la transformation Processeur de données pour restructurer le document XML.

Le composant de démarrage est un objet XMap qui contient une instruction Routeur. Une instruction Option vérifie si le rôle Employé est « Manager ». Si le rôle est Manager, XMap mappe les éléments de l'employé à un groupe de sortie manager. Sinon, XMap mappe les éléments de l'employé à un groupe Travailleur dans le XML de sortie.

Le composant de démarrage XMap demande à un autre XMap de mapper les éléments d'Employé aux éléments de sortie.

Exemple de schéma XML d'entrée

Le schéma XML d'entrée pour l'exemple XMap a la structure de suivante :

```
<?xml version="1.0" encoding="UTF-16LE"?>
<!-- edited with XMLSpy v2012 sp1 (x64) (http://www.altova.com) by Informatica
Corporation (Informatica Corporation) -->
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns="company"
targetNamespace="company" elementFormDefault="qualified"
attributeFormDefault="unqualified">

  <xs:element name="Employee" type="EmployeeType"/>

  <xs:element name="Input">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="Company" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
```

```

<xs:element name="Company">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="Name" type="xs:string"/>
      <xs:element ref="Department" maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>

<xs:element name="Department">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="Name" type="xs:string"/>
      <xs:element ref="Employee" maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>

<xs:complexType name="EmployeeType">
  <xs:sequence>
    <xs:element name="FirstName" type="xs:string" minOccurs="0"/>
    <xs:element name="LastName" type="xs:string" minOccurs="0"/>
    <xs:element name="Role" type="xs:string" minOccurs="0"/>
    <xs:element name="StartDate" type="xs:date" minOccurs="0"/>
  </xs:sequence>
  <xs:attribute name="id" type="xs:string"/>
</xs:complexType>

</xs:schema>

```

La racine du schéma est Entrée. L'entrée comprend des sociétés à occurrences multiples. Dans chaque société se trouvent plusieurs Départements. Chaque Département a des Employés. L'Employé a un rôle qui détermine si l'employé est manager ou non.

Exemple de schéma XML de sortie

Le schéma XML de sortie pour l'exemple XMap a la structure suivante :

```

<?xml version="1.0" encoding="UTF-16LE"?>
<!-- edited with XMLSpy v2012 sp1 (x64) (http://www.altova.com) by Informatica
Corporation (Informatica Corporation) -->
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns="organization"
targetNamespace="organization" elementFormDefault="qualified"
attributeFormDefault="unqualified">

  <xs:element name="Worker" type="WorkerType"/>

  <xs:element name="Output">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="Organization" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>

  <xs:element name="Organization">
    <xs:complexType>
      <xs:sequence>
        <xs:choice maxOccurs="unbounded">
          <xs:element name="Worker" type="WorkerType"/>
          <xs:element name="Manager" type="WorkerType"/>
        </xs:choice>
        <xs:element name="Department" type="xs:string" maxOccurs="unbounded"/>
      </xs:sequence>
      <xs:attribute name="noOfEmployees" type="xs:int"/>
    </xs:complexType>
  </xs:element>
</xs:schema>

```



```

</xs:element>

<xs:complexType name="WorkerType">
  <xs:sequence>
    <xs:element name="FirstName" type="xs:string" minOccurs="0"/>
    <xs:element name="LastName" type="xs:string" minOccurs="0"/>
    <xs:element name="FullName" type="xs:string" minOccurs="0"/>
    <xs:element name="Id" type="xs:string"/>
    <xs:element name="YearsOfService" type="xs:int" minOccurs="0"/>
  </xs:sequence>
</xs:complexType>

</xs:schema>

```

La racine du schéma est Sortie. La sortie comprend des organisations à occurrences multiples. Dans chaque organisation se trouvent des Travailleurs et des Managers. Travailleurs et Managers sont des WorkerTypes. Un Travailleur et un Manager contiennent les mêmes éléments. Le WorkerType inclut un élément YearsOfService que la XMap calcule en fonction de la StartDate.

Données d'entrée XML

Le texte suivant montre des exemples de données du document XML d'entrée :

```

<?xml version="1.0" encoding="windows-1252"?>
<Input xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="company
Company.xsd" xmlns="company">
  <Company>
    <Name>Hypostores</Name>

    <Department>
      <Name>Customer Service</Name>

      <Employee id="25721195">
        <FirstName>Blair</FirstName>
        <LastName>Conner</LastName>
        <Role>Manager</Role>
        <StartDate>1993-04-21</StartDate>
      </Employee>

      <Employee id="238036220">
        <FirstName>Karina</FirstName>
        <LastName>Rasmussen</LastName>
        <Role>Worker</Role>
        <StartDate>1993-08-15</StartDate>
      </Employee>
    </Department>

    <Department>
      <Name>Research and Development</Name>

      <Employee id="259089785">
        <FirstName>Thaddeus</FirstName>
        <LastName>Burt</LastName>
        <Role>Consultant</Role>
        <StartDate>1998-02-26</StartDate>
      </Employee>

      <Employee id="289021615">
        <FirstName>Christen</FirstName>
        <LastName>Fulton</LastName>
        <Role>Worker</Role>
        <StartDate>1997-11-16</StartDate>
      </Employee>

      <Employee id="761338290">
        <FirstName>Felix</FirstName>
        <LastName>Boyd</LastName>
        <Role>Worker</Role>

```

```

        <StartDate>2009-12-29</StartDate>
    </Employee>

</Department>
</Company>

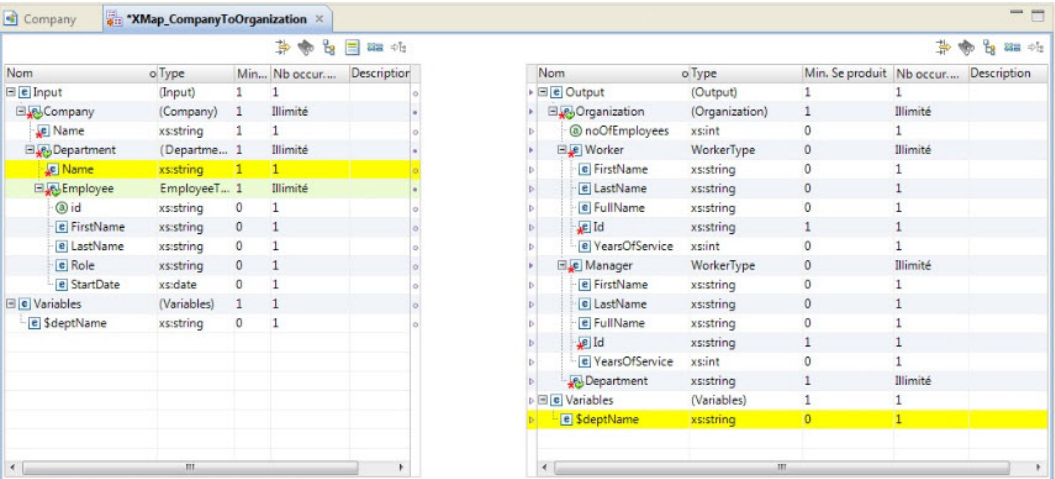
```

Les données peuvent inclure plusieurs entreprises. Chaque entreprise possède plusieurs départements. Chaque employé d'un département joue un rôle soit de responsable, soit d'une autre fonction.

Hiéarchies XML d'entrée et de sortie

L'éditeur XMap affiche la hiérarchie d'entrée dans la zone gauche de la vue et la hiérarchie XML de sortie dans la zone droite de la vue.

La figure suivante montre les hiérarchies XML d'entrée et de sortie :



Instructions de mappage dans l'exemple

Utilisez la zone de la grille de l'éditeur XMap pour définir les instructions qui mappent les éléments XML d'entrée aux éléments XML de sortie. Créer et modifier des instructions de mappage dans la grille. Définir le contexte, la condition ainsi que l'entrée et la sortie attendues pour une instruction de mappage. Vous pouvez ajouter des variables aux instructions de mappage.

La figure suivante montre les instructions de mappage dans la grille :

Nom	Type d'instruction	Entrée	Condition	Ignore...	Valeur...	En échec	Sortie	Mode	Remarque
1 Company	Groupe répétition	tns0:Company		<input type="checkbox"/>		Propager		Ajouter	
2 Department	Groupe répétition	tns0:Department		<input type="checkbox"/>		Propager		Ajouter	
3 NameToDepartment	Mapper	tns0:Name		<input type="checkbox"/>		Propager	\$deptName	Correspondance ou...	
4 MatchOrganization	Groupe	.		<input type="checkbox"/>		Propager	tns0:Organization[tn...	Correspondance ou...	
5 EmployeeToWorker	Groupe répétition	tns0:Employee		<input type="checkbox"/>		Ignorer l'it...		Correspondance ou...	
6 Employee	Routeur			<input type="checkbox"/>		Ignorer		Correspondance ou...	
7 EmployeeToManager	Option		tns0:Role="Ma...	<input checked="" type="checkbox"/>		Propager		Correspondance ou...	
8 EmployeeToWorker	EmployeeToWorker			<input type="checkbox"/>		Propager	tns0:Manager	Ajouter	
9 EmployeeToWorker	Valeur par défaut			<input type="checkbox"/>		Propager		Correspondance ou...	
10 EmployeeToWorker	EmployeeToWorker			<input type="checkbox"/>		Propager	tns0:Worker	Ajouter	
11 Increment employee count	Mapper	dp:output()/@noOfE...		<input type="checkbox"/>	1	Propager	@noOfEmployees	Correspondance ou...	

La grille contient les instructions de mappage suivants :

Grille ligne 1, instruction Groupe de répétition nommée Société

L'instruction Société est une instruction Groupe de répétition. Elle effectue une répétition pour chaque élément de la société. L'instruction fournit un contexte pour le reste des instructions de la grille. Pour chaque société, la transformation Processeur de données évalue les instructions enfant.

Grille ligne 2, instruction Groupe de répétition nommée Département

L'instruction Département est une instruction Groupe de répétition. Elle effectue une répétition pour chaque élément Département. L'instruction fournit un contexte pour le reste des instructions de la grille. Pour chaque département, la transformation Processeur de données évalue les instructions enfant.

Grille ligne 3, instruction Mappage nommée NametoDepartment

L'instruction NametoDepartment est une instruction Mappage. Elle mappe l'élément Nom à une variable \$deptName.

Grille ligne 4, instruction Groupe de répétition nommée MatchOrganization

L'instruction MatchOrganization est une instruction Groupe de répétition. Elle a une expression de sortie :

```
tns0:Organization[tns0:Department=$deptName]
```

L'instruction trouve l'élément Organisation à la sortie qui contient un élément Département enfant avec la valeur dans \$deptName. Ou, si l'élément Département n'existe pas, l'élément est créé.

Grille ligne 5, instruction Groupe de répétition nommée EmployeeToWorker

L'instruction EmployeeToWorker est une instruction Groupe de répétition. Elle effectue une répétition pour chaque élément Employé.

Grille ligne 6, instruction Routeur nommée Employé

L'instruction Employé est une instruction Routeur. L'instruction n'a pas d'entrée ni de sortie.

Grille ligne 7, instruction Option nommée EmployeeToMgr

L'instruction EmployeeToMgr est une instruction Option. L'instruction Option contient la condition suivante :

```
tns0:Role="Manager".
```

Lorsque le rôle est Manager, l'instruction est vraie et la transformation Processeur de données évalue les instructions imbriquées à l'intérieur de l'instruction Option.

Grille ligne 8, instruction Exécuter XMap nommée EmployeeToWorker

L'instruction EmployeeToWorker est une instruction Exécuter XMap. Elle appelle la XMap XMap_EmployeesToRoles pour transmettre les éléments Employé au type Manager.

Grille ligne 9, instruction par défaut nommée EmployeeToWorker

L'instruction EmployeeToWorker est une instruction par défaut. La transformation Processeur de données effectue les instructions enfant lorsque le rôle de l'employé n'est pas Manager.

Grille ligne 10, instruction Exécuter XMap nommée EmployeeToWorker

L'instruction EmployeeToWorker est une instruction Exécuter XMap. Elle appelle la XMap XMap_EmployeesToRoles pour transmettre les éléments Employé au type Travailleur.

Grille ligne 11, instruction de mappage nommée IncrementEmployeeCount

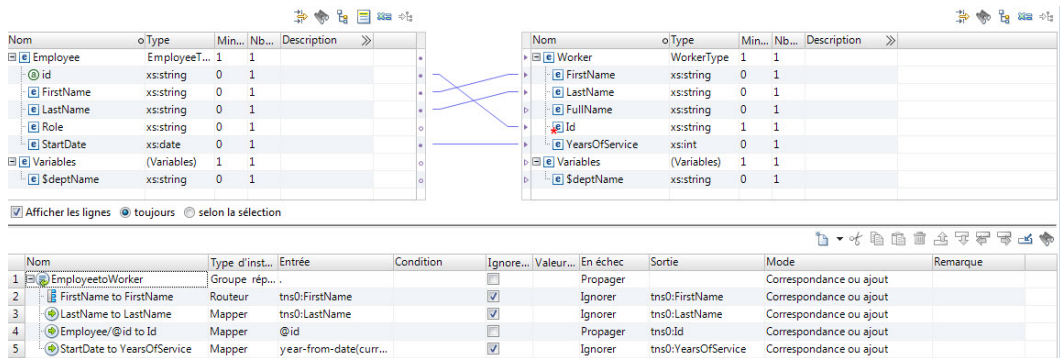
L'instruction IncrementEmployeeCount est une instruction Mappage. Elle appelle la transformation Processeur de données pour ajouter 1 à @noOfEmployees pour chaque employé qu'elle répète. L'instruction Mappage contient l'expression d'entrée suivante :

```
dp:output()/@ noOfEmployees + 1.
```

Exemple d'instructions Groupe

Le XMap EmployeeToWorker déplace des éléments d'un employé à un ouvrier. XMap traite un employé.

La figure suivante montre le XMap EmployeeToWorker dans l'éditeur XMap :



La grille contient les instructions de mappage suivants :

Grille ligne 1, instruction Groupe nommée EmployeeToWorker

L'instruction EmployeeToWorker est une instruction de groupe. Elle fournit le contexte pour le reste des instructions de mappage.

Grille ligne 2, instruction Mappage nommée FirstNametoFirstName

L'instruction FirstNametoFirstName est une instruction de mappage. Elle mappe le prénom au prénom.

Grille ligne 3, instruction Mappage nommée LastNametoLastName

L'instruction LastNametoLastName est une instruction de mappage. Elle mappe le nom de famille au nom de famille.

Grille ligne 4, instruction Mappage nommée Employee/@IDtoId

L'instruction Employee/@IDtoId est une instruction de mappage. Elle mappe l'ID de l'employé à l'ID de l'employé.

Grille ligne 5, instruction Mappage nommée StartDatetoYearsofService

L'instruction StartDatetoYearsofService est une instruction de mappage. Elle détermine le nombre d'années de service en soustrayant la date de début à la date en cours.

CHAPITRE 7

Bibliothèques

Ce chapitre comprend les rubriques suivantes :

- [Présentation des bibliothèques, 117](#)
- [Structure de la bibliothèque, 118](#)
- [Propriétés de l'élément, 118](#)
- [Gestion de la bibliothèque, 118](#)
- [Modifier les bibliothèques avec l'éditeur de bibliothèque, 120](#)
- [Modifier les bibliothèques avec l'éditeur IntelliScript, 122](#)

Présentation des bibliothèques

Une bibliothèque est un objet de transformation Processeur de données qui contient les composants prédéfinis utilisés pour transformer une série de normes de messagerie. La transformation Processeur de données utilise une bibliothèque pour convertir une entrée de type message standard dans d'autres formats. Vous pouvez créer des objets de bibliothèque pour toutes les bibliothèques.

Une bibliothèque contient un grand nombre d'objets et de composants tels que des analyseurs, des sérialiseurs et des schémas XML, qui transforment l'entrée standard et les messages spécifiques d'applications en sortie XML. Une bibliothèque peut contenir des objets pour la validation de messages, l'affichage d'accusés de réception et de diagnostics. Une bibliothèque utilise les objets pour transformer le type de messagerie depuis une entrée standard vers un XML et à partir du XML vers d'autres formats.

Vous pouvez créer des objets de bibliothèque pour les bibliothèques ACORD, BAI, CREST, DTCC-NSCC, EDIFACT, EDIT-UCS & WINS, EDI_VICS, EDI-X12, FIX, FpML, HIPAA, HIX, HL7, IATA, IDS, MDM Mapping, NACHA, NCPDP, SEPA, SWIFT et Telekurs.

Un éditeur de bibliothèque dédié vous permet de modifier les spécifications des bibliothèques DTCC-NSCC, EDIFACT, EDI-X12, HIPAA, HL7 et SWIFT. Un message de bibliothèque contient un élément racine, des éléments conteneurs et des éléments de données. Les types de conteneurs et les éléments de données varient en fonction du type de message. Vous pouvez ajouter et supprimer les éléments et configurer les propriétés des éléments pour changer les paramètres de validation.

Pour plus d'informations sur les types de message standard, consulter le *Guide des bibliothèques de Data Transformation*.

Structure de la bibliothèque

Une bibliothèque est un ensemble de transformations. Toutes les bibliothèques contiennent des analyseurs, des sérialiseurs et des schémas XML. Certaines bibliothèques contiennent des composants supplémentaires pour la validation de messages, les accusés de réception et les affichages de diagnostics. Un objet de bibliothèque est un ensemble de composants permettant de convertir un type de message standard.

Lorsque vous créez une transformation Processeur de données, vous pouvez inclure un objet de bibliothèque au lieu de créer vos propres scripts pour transformer un type de message standard. Vous pouvez utiliser un objet de bibliothèque sans y apporter de modification, ou vous pouvez le modifier selon vos besoins.

Chaque objet de bibliothèque transforme une norme industrielle particulière. Par exemple, la bibliothèque HL7 contient des composants pour chacun des types de message ou des structures de données disponibles dans la norme de messagerie HL7 pour les systèmes d'informations médicales.

La bibliothèque contient des types spécifiques de types de message. Par exemple, la bibliothèque HL7 contient des messages tels que :

```
ADT_A01_Admit_a_Patient  
ADT_A02_Transfer_a_Patient
```

Propriétés de l'élément

Un message de bibliothèque contient un élément racine, des éléments conteneurs et des éléments de données. Les types de conteneurs et les éléments de données varient en fonction du type de message. Vous pouvez configurer à la fois les propriétés de l'élément conteneur et de l'élément de données avec l'éditeur de bibliothèque.

Les propriétés comprennent les catégories suivantes :

Paramètres globaux

Propriétés qui ont la même valeur pour un élément avec plus d'une instance, chacune occupant une position hiérarchique différente. Toute modification effectuée sur une propriété des paramètres globaux s'affiche dans toutes les instances de l'élément.

Paramètres positionnels

Propriétés dont la valeur peut différer pour chaque instance de l'élément dans la hiérarchie.

Les propriétés de l'élément sont nommées conformément aux normes industrielles. L'éditeur de bibliothèque affiche différentes propriétés pour différentes bibliothèques.

Gestion de la bibliothèque

L'éditeur de bibliothèque vous permet de personnaliser les structures et les éléments de type de messages des bibliothèques DTCC-NSCC, EDIFACT, EDI-X12, HIPAA, HL7 et SWIFT. L'éditeur de bibliothèque vous permet de modifier la structure du type de message et d'ajouter, de modifier et de supprimer des éléments du message de la bibliothèque.

Vous pouvez avoir besoin de modifier le mode de transformation d'un type de message. Une transformation Bibliothèque contient un grand nombre d'objets et de composants qui la définissent, tels que des scripts

avec des analyseurs, des sérialiseurs et des schémas XML. Une transformation Bibliothèque peut contenir des objets pour la validation des messages, les accusés de réception et l'affichage des diagnostics.

Une transformation Bibliothèque utilise ses objets pour convertir le type de messagerie d'une entrée standard au format XML et du format XML vers d'autres formats. Pour accéder aux scripts, aux XMap et aux schémas que vous avez créés à l'aide de l'éditeur de bibliothèque, ainsi que pour les modifier, vous devez générer les objets de bibliothèque. Générez uniquement les objets de bibliothèque si vous devez y effectuer des modifications impossibles à réaliser à l'aide de l'éditeur de bibliothèque.

Après avoir généré les objets de bibliothèque, utilisez l'éditeur IntelliScript pour modifier les analyseurs, les sérialiseurs et les mappeurs. Par exemple, vous voulez modifier la structure de sortie pour satisfaire vos exigences. Vous générez les objets de bibliothèque et modifiez les scripts à l'aide de l'éditeur IntelliScript.

Si vous créez une bibliothèque pour des packages de bibliothèque qui ne prennent pas en charge l'éditeur de bibliothèque, les objets de bibliothèque sont pré-générés. Vous n'avez pas besoin de générer les objets de bibliothèque. Vous pouvez utiliser l'éditeur IntelliScript pour modifier directement les composants de script de ces bibliothèques.

Si vous avez généré des objets de bibliothèque ou si vous disposez d'objets pré-générés, vous ne pouvez pas modifier les éléments de la bibliothèque avec l'éditeur de bibliothèque. Pour utiliser à nouveau l'éditeur de bibliothèque, vous devez supprimer les objets générés des bibliothèques pouvant être modifiées avec l'éditeur de bibliothèque. Par exemple, vous pouvez décider que vous voulez ajouter des champs d'entrée mais pas modifier la structure de sortie. Supprimer les objets de la bibliothèque et ajoutez ensuite des éléments personnalisée avec l'éditeur de bibliothèque.

Remarque: Les modifications apportées aux objets de bibliothèque générés sont perdues lorsque vous supprimez les objets générés.

Exemple de bibliothèque pour EDI-X12

Vous et vos fournisseurs utilisez la norme de bon de commande X12 850 pour la documentation et le traitement des commandes. Vous voulez modifier les spécifications du type de message de la bibliothèque à partir des normes industrielles pour prendre en charge vos processus internes.

Les fabricants de meubles ont besoin d'éléments standards comme le modèle, la couleur, la taille, et d'un élément personnalisé pour le type de textile. Ils n'ont pas besoin de numéro de lot ni d'une date d'expiration. Utilisez l'éditeur de bibliothèque pour supprimer les éléments et ajouter des éléments personnalisés.

Vous pouvez également utiliser l'éditeur de bibliothèque pour changer les propriétés d'élément, ajouter des segments et copier des éléments. Par exemple, modifiez la propriété d'élément qui définit la liste des couleurs pour ajouter de nouvelles couleurs. Pour ajouter un élément de code fabricant au segment textile et au segment matériel, vous pouvez le copier.

Vous exécutez une transformation Processeur de données qui contient votre bibliothèque personnalisée. Le texte d'entrée de vos fournisseurs est formaté dans la version modifiée de la norme de bon de commande X12 850. Le format d'entrée modifié correspond aux modifications que vous avez effectuées sur la bibliothèque. Le texte est transformé en sortie hiérarchique qui peut être envoyée vers d'autres transformations pour un traitement ultérieur.

Modifier les bibliothèques avec l'éditeur de bibliothèque

Vous pouvez utiliser un éditeur de bibliothèque dédié pour modifier les spécifications des bibliothèques EDI-X12, SWIFT, HL7, HIPAA, DTCC-NSCC et EDIFACT.

L'éditeur de bibliothèque vous permet de personnaliser les structures et éléments de type message avec un éditeur personnalisé pour chaque type de message. Vous pouvez ajouter, modifier et supprimer des éléments de l'objet de bibliothèque avec l'éditeur de bibliothèque.

Ajout d'un élément à l'aide de l'éditeur de bibliothèque

Un message de bibliothèque contient un élément racine, des éléments conteneurs et des éléments de données. Utilisez l'éditeur de bibliothèque pour ajouter un élément à un message.

1. Dans la vue **Objets**, sélectionnez l'objet éditeur de bibliothèque et cliquez sur **Ouvrir**.
L'éditeur de bibliothèque s'affiche.
2. Cliquez sur l'icône **Ajouter l'élément** et choisissez l'emplacement où vous souhaitez ajouter l'élément.
 - **Nouveau**. Ajouter l'élément à la fin de la liste des éléments.
 - **Insérer au-dessus de**. Insérer l'élément au-dessus de l'élément sélectionné dans l'éditeur de bibliothèque.
 - **Insérer en dessous de**. Insérer l'élément en dessous de l'élément sélectionné dans l'éditeur de bibliothèque.
 - **Insérer dans**. Insérer l'élément dans l'élément conteneur sélectionné dans l'éditeur de bibliothèque.
3. Sélectionnez si vous souhaitez copier un élément existant ou créer un nouvel élément et cliquez sur **OK**.

Modification des propriétés d'un élément avec l'éditeur de bibliothèque

Utilisez l'éditeur de bibliothèque pour éditer un élément conteneur ou un élément de données dans une bibliothèque. Lorsque vous modifiez les propriétés d'un élément, vous modifiez les spécifications pour le type de message.

1. Dans la fenêtre **Définition du message** de l'éditeur de bibliothèque, sélectionnez l'élément à éditer.
La fenêtre **Propriétés** affiche les propriétés de l'élément.
2. Dans la fenêtre **Propriétés**, sélectionnez une propriété et entrez la nouvelle valeur de propriété.
Si vous entrez un type de valeur incorrect ou une valeur hors plage, vous serez invité à corriger la valeur.

Test d'une bibliothèque

Testez l'objet de bibliothèque dans la vue **Visionneuse de données**.

Avant de tester la bibliothèque, sélectionnez un exemple de fichier d'entrée à tester.

1. Pour cela, sélectionnez un fichier dans le panneau **Général** de l'éditeur de bibliothèque, près du champ **Exemple d'entrée**.
2. Ouvrez la vue **Visionneuse de données**.
3. Pour sélectionner le type de message de la bibliothèque que vous avez modifié en tant que composant d'exécution, cliquez sur **Synchroniser avec l'éditeur**.

4. Cliquez sur **Exécuter**.

L'outil Developer tool exécute l'objet de bibliothèque Analyseur. Les résultats de la sortie s'affichent dans la fenêtre **Sortie**.

5. Si des erreurs de validation se produisent, le panneau **Erreurs de sortie** de la fenêtre **Sortie** répertorie les erreurs et leur emplacement. Pour trouver la source de l'erreur, double-cliquez sur l'erreur.
6. Pour afficher les fichiers d'erreur de validation, cliquez sur les noms des fichiers dans le panneau **Sorties supplémentaires** de la fenêtre **Sortie**.

Lorsque vous cliquez sur le nom de fichier `ErrorsFound.txt` ou `Errors.xml`, le fichier s'ouvre dans un navigateur externe.

Génération des objets de la bibliothèque

Générez les objets de la bibliothèque pour pouvoir y accéder directement avec les éditeurs de transformation Processeur de données. Après avoir généré les objets de la bibliothèque, vous ne pouvez pas utiliser l'éditeur de bibliothèque pour modifier la bibliothèque.

Vous devez générer des objets de bibliothèque pour accéder et éditer les analyseurs préconfigurés, les mappers, les sérialiseurs et les schémas XML associés à la bibliothèque.

1. Dans la vue **Objets**, faites un clic droit sur la bibliothèque et sélectionnez **Générer des objets de bibliothèque**.

2. Pour accéder aux objets de bibliothèque, sélectionnez **Oui**.

L'outil Developer tool crée un dossier **Objets de bibliothèque générés** qui contient les objets de bibliothèque. Le composant de démarrage est généré.

Remarque: Si vous voulez modifier les objets de bibliothèque à générer, sélectionnez un autre composant.

3. Pour modifier un script ou un schéma, sélectionnez-le et cliquez sur **Ouvrir**.

Utilisez l'éditeur IntelliScript pour modifier le script.

Suppression des objets de la bibliothèque

Pour éditer les éléments de la bibliothèque avec l'éditeur de bibliothèque, vous devez retirer de la bibliothèque les objets que vous avez générés. Lorsque vous supprimez les objets de la bibliothèque, toute modification qui leur a été apportée est perdue.

1. Dans la vue **Objets**, faites un clic droit sur la bibliothèque et sélectionnez **Ignorer les objets de la bibliothèque créés**.
2. Pour supprimer les objets de la bibliothèque à partir des éditeurs de transformation Processeur de données, sélectionnez **Oui**.

Le dossier **Objets de la bibliothèque générés** qui est supprimé. Les composants et les objets de la bibliothèque existent toujours, mais ne sont pas accessibles via les éditeurs de transformation Processeur de données.

Modifier les bibliothèques avec l'éditeur IntelliScript

Lorsque vous créez une bibliothèque ACORD, BAI, FIX, HIX, IATA, IDS, NACHA, NCPDP, Telekurs, Bloomberg, SEPA, FpML ou Thompson Reuters, vous pouvez utiliser l'éditeur IntelliScript pour modifier directement ses composants de script. Vous n'avez pas besoin de générer les objets de bibliothèque.

Pour les bibliothèques EDI-X12, SWIFT, HL7, HIPAA, DTCC-NSCC, Edifact, SEPA et FpML, vous devez générer les objets de bibliothèque pour accéder aux scripts, aux XMap et aux schémas que vous avez créés avec l'éditeur de bibliothèque et les modifier avec l'éditeur IntelliScript.

L'éditeur IntelliScript est un outil graphique qui permet de modifier les scripts. Utilisez-le pour ajouter des composants au script et configurer les propriétés de ces composants. Pour plus d'informations sur les scripts et l'éditeur IntelliScript, consultez ["Présentation des scripts" à la page 139](#).

Si vous avez importé un projet de bibliothèque depuis une version antérieure en tant que service de Data Transformation, vous pouvez modifier les composants de script à l'aide de l'éditeur IntelliScript.

CHAPITRE 8

Objet de schéma

Ce chapitre comprend les rubriques suivantes :

- [Présentation de l'objet de schéma, 123](#)
- [Fichiers de schéma, 123](#)
- [Vue de la présentation de l'objet de schéma, 124](#)
- [Vue Schéma de l'objet de schéma, 125](#)
- [Vue avancée de l'objet du schéma, 130](#)
- [Création d'un objet de schéma, 131](#)
- [Mises à jour des schémas, 132](#)

Présentation de l'objet de schéma

Un objet de schéma est un schéma hiérarchique que vous importez vers le référentiel modèle. Après avoir importé le schéma, vous pouvez visualiser les composants de schéma dans l'outil Developer tool. Vous pouvez importer un schéma Avro, Parquet, XML ou JSON. L'outil Developer tool convertit le schéma en fichier .xsd dans le référentiel modèle.

Lorsque vous créez un service Web SOAP, vous pouvez définir la structure du service Web en fonction d'un schéma hiérarchique. Lorsque vous créez un service Web sans un WSDL, vous pouvez définir les opérations, d'entrée, de sortie et d'erreur des signatures en fonction des types et des éléments que le schéma définit.

Lorsque vous importez un schéma, vous pouvez éditer les propriétés dans le schéma général dans la vue **Présentation**. Éditez les propriétés avancées dans la vue **Avancée**. Affichez le contenu du fichier de schéma dans la vue **Schéma**.

Fichiers de schéma

Vous pouvez ajouter plusieurs fichiers .xsd racine à un objet de schéma. Vous pouvez également supprimer plusieurs fichiers .xsd racine d'un objet de schéma.

Lorsque vous ajoutez un fichier de schéma, l'outil Developer importe tous les fichiers .xsd importés ou inclus dans le fichier que vous ajoutez. L'outil Developer valide les fichiers que vous ajoutez par rapport aux fichiers faisant partie de l'objet de schéma. L'outil Developer ne vous permet pas d'ajouter un fichier si celui-ci est en conflit avec un fichier faisant partie de l'objet de schéma.

Par exemple, un objet de schéma contient un fichier de schéma racine appelé « BostonCust.xsd ». Vous voulez ajouter le fichier de schéma racine « LACust.xsd » à l'objet de schéma. Les deux fichiers de schéma ont le même espace de nom cible et définissent un élément appelé « Client ». Lorsque vous essayez d'ajouter le fichier de schéma « LACust.xsd » à l'objet de schéma, l'outil Developer vous invite à conserver le fichier « BostonCust.xsd » ou à le remplacer par le fichier « LACust.xsd ».

Vous pouvez utiliser l'attribut `xsd:nillable` pour marquer les éléments XSD comme nillable. Lorsque vous marquez un élément comme nillable, l'élément correspondant dans le fichier XML autorise les valeurs Null.

Vous pouvez supprimer tout fichier de schéma racine. Si vous le faites, l'outil Developer modifie le type des éléments qui étaient définis sur `xs:string` par le fichier de schéma.

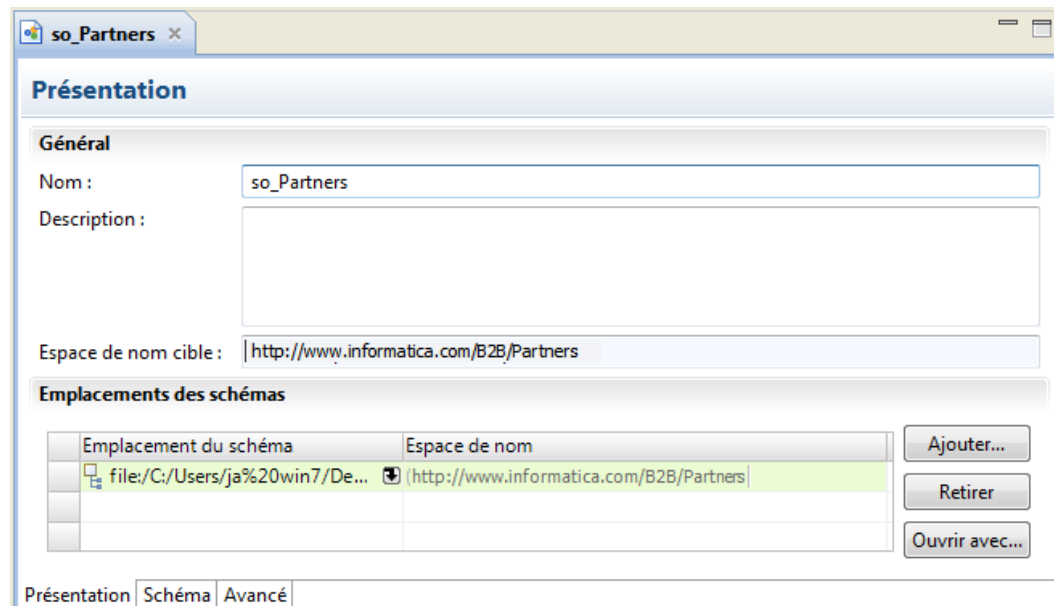
Pour ajouter un fichier de schéma, sélectionnez la vue **Présentation**, et cliquez sur le bouton **Ajouter** situé à côté de la liste **Emplacements des schémas**. Sélectionnez ensuite le fichier de schéma. Pour supprimer un fichier de schéma, sélectionnez le fichier et cliquez sur le bouton **Supprimer**.

Vue de la présentation de l'objet de schéma

Sélectionnez la vue **Présentation** pour mettre à jour le nom ou la description du schéma, afficher les espaces de noms et gérer les fichiers de schéma.

La vue **Présentation** affiche le nom, la description et l'espace de nom cible du schéma. Vous pouvez modifier le nom et la description du schéma. L'espace de nom cible affiche l'espace de nom auquel les composants du schéma appartiennent. Si aucun espace de nom cible ne s'affiche, les composants du schéma n'appartiennent pas à un espace de nom.

La figure suivante montre la vue **Présentation** d'un objet de schéma :



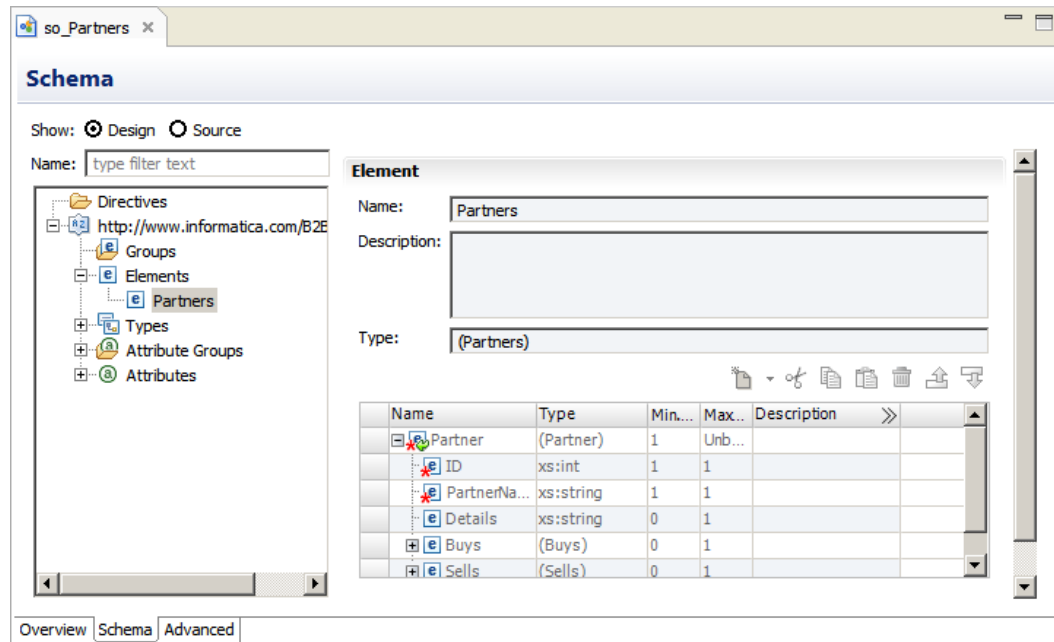
La zone **Emplacements des schémas** répertorie les fichiers de schéma et les espaces de noms. Vous pouvez ajouter plusieurs fichiers .xsd racine. Si un fichier de schéma comprend ou importe d'autres fichiers de schéma, l'outil Developer inclut tous les fichiers .xsd enfants dans le schéma.

Vue Schéma de l'objet de schéma

La vue **Schéma** affiche une liste alphabétique des groupes, éléments, types, groupes d'attributs et attributs dans le schéma. Lorsque vous sélectionnez un groupe, un élément, un type, un groupe d'attributs ou un attribut dans la vue **Schéma**, les propriétés s'affichent dans le panneau de droite. Vous pouvez également afficher chaque fichier .xsd dans la vue **Schéma**.

La vue **Schéma** contient une liste des espaces de nom et des fichiers .xsd dans l'objet de schéma.

La figure suivante montre la vue **Schéma** d'un objet de schéma :



Vous pouvez effectuer les tâches suivantes dans la vue **Schéma** :

- Pour afficher la liste des constructions de schéma, développez le dossier **Instructions**. Pour afficher l'espace de nom, le préfixe et l'emplacement, sélectionnez une construction de schéma dans la liste.
- Pour afficher le préfixe de l'espace de nom, le préfixe généré et l'emplacement, sélectionnez un espace de nom. Vous pouvez modifier le préfixe généré.
- Pour afficher l'objet de schéma comme fichier .xsd, sélectionnez **Source**. Si l'objet de schéma comprend d'autres schémas, vous pouvez sélectionner le fichier .xsd à afficher.
- Pour afficher une liste alphabétique des groupes, éléments, types, attributs de groupe et attributs dans chaque espace de nom du schéma, sélectionnez **Conception**. Vous pouvez entrer un ou plusieurs caractères dans le champ **Nom** pour filtrer les groupes, éléments, types, groupes d'attributs et attributs par nom.
- Pour afficher les propriétés de l'élément, sélectionnez un groupe, un élément, un type, un groupe d'attributs ou un attribut. L'outil Developer affiche différents champs dans le panneau de droite en fonction de l'objet que vous avez sélectionné.

Lorsque vous affichez les types, vous pouvez voir si un type est dérivé d'un autre. L'interface affiche le type parent. L'interface montre également si l'élément enfant a hérité des valeurs par restriction ou extension.

Propriétés de l'espace de nom

La vue **Espace de nom** affiche le préfixe et l'emplacement pour un espace de nom sélectionné.

L'espace de nom associé à chaque fichier de schéma différencie les éléments provenant de différentes sources mais ayant les mêmes noms. La référence Uniform Resource Identifier (URI) définit l'emplacement du fichier qui contient les éléments et les noms d'attributs.

Lorsque vous importez un schéma contenant plusieurs espaces de nom, l'outil Developer tool ajoute les espaces de nom à l'objet de schéma. Lorsque le fichier de schéma comprend les autres schémas, les espaces de nom pour ces schémas sont également inclus.

L'outil Developer crée un préfixe généré pour chaque espace de nom. Lorsque le schéma ne contient pas de préfixe, l'outil Developer tool génère le préfixe d'espace de nom tns0 et incrémente le numéro de préfixe pour chaque préfixe d'espace de nom supplémentaire. L'outil Developer réserve le préfixe d'espace de nom xs. Si vous importez un schéma contenant le préfixe d'espace de nom xs, l'outil Developer tool crée le préfixe généré xs1. L'outil Developer incrémente le numéro de préfixe lorsque le schéma contient la valeur de préfixe générée .

Par exemple, Customer_Orders.xsd possède un espace de nom. Le schéma comprend un autre schéma, Customers.xsd. Le schéma Costumers possède un espace de nom. L'outil Developer attribue un préfixe tns0 pour l'espace de nom Customer_Orders et un préfixe tns1 pour l'espace de nom Costomers.

Pour afficher l'emplacement et le préfixe de l'espace de nom, sélectionnez un espace de nom dans la vue **Schéma**.

Lorsque vous créez un service Web de plus d'un objet de schéma, chaque espace de nom doit posséder un préfixe unique. Vous pouvez modifier le préfixe généré pour chaque espace de nom.

Propriétés de l'élément

Un élément est un type simple ou complexe. Un type complexe contient d'autres types. Lorsque vous sélectionnez un élément dans la vue **Schéma**, l'outil Developer répertorie les éléments enfants et les propriétés dans le panneau à droite de l'écran.

Le tableau suivant décrit les propriétés de l'élément qui s'affichent lorsque vous sélectionnez un élément :

Propriété	Description
Nom	Le nom d'élément.
Description	Description du type.
Type	Type d'élément.

Le tableau suivant décrit les propriétés de l'élément enfant qui s'affichent lorsque vous sélectionnez un élément :

Propriété	Description
Nom	Le nom d'élément.
Type	Type d'élément.
Exécution minimale	Nombre minimal de fois que l'élément peut s'exécuter à un moment donné dans une instance.

Propriété	Description
Exécution maximale	Nombre maximal de fois que l'élément peut s'exécuter à un moment donné dans une instance.
Description	Description de l'élément.

Pour afficher les propriétés d'un élément enfant, double-cliquez sur la flèche dans la colonne de description pour développer la fenêtre.

Le tableau suivant décrit les propriétés supplémentaires d'un élément enfant qui s'affichent lorsque vous développez la colonne de description :

Propriété	Description
Valeur fixe	Valeur spécifique pour un élément qui ne change pas.
Nillable	L'élément peut avoir les valeurs nil. Un élément nil possède des balises d'élément mais n'a aucune valeur, ni aucun contenu.
Résumé	L'élément est un type abstrait. Une instance doit inclure le type dérivé de ce type. Un type abstrait n'est pas valide sans les types d'élément dérivés.
Valeur minimale	Valeur minimale d'un élément dans une instance.
Valeur maximale	Valeur maximale d'un élément dans une instance.
Longueur minimale	Longueur minimale d'un élément. La longueur est en octets, caractères ou éléments en fonction du type d'élément.
Longueur maximale	Longueur maximale d'un élément. La longueur est en octets, caractères ou éléments en fonction du type d'élément.
Énumération	Liste de toutes les valeurs légales d'un élément.
Modèle	Modèle d'expression qui définit les valeurs des éléments valables.

Propriétés de l'élément avancé

Pour afficher les propriétés avancées d'un élément, sélectionnez l'élément dans la vue **Schéma**. Cliquez sur **Avancé**.

Le tableau suivant décrit les propriétés de l'élément avancé :

Propriété	Description
Résumé	L'élément est un type abstrait. Un message SOAP doit inclure le type dérivé de ce type. Un type abstrait n'est pas valide sans les types d'élément dérivés.
Bloc	Empêche l'élément dérivé d'apparaître dans le fichier XML à la place de cet élément. La valeur de bloc peut contenir "# all" ou une liste qui comprend l'extension, la restriction ou la substitution.
Finale	Empêche le schéma d'étendre ou de restreindre le type simple comme un type dérivé.

Propriété	Description
Groupe de substitution	Nom d'un élément à substituer avec l'élément.
Nillible	L'élément peut avoir les valeurs nil. Un élément nil possède des balises d'élément mais n'a aucune valeur, ni aucun contenu.

Propriétés du type simple

Un élément de type simple est un élément qui contient un texte non structuré. Lorsque vous sélectionnez un élément de type simple dans la vue **Schéma**, les informations sur l'élément de type simple s'affichent dans le panneau de droite.

Le tableau suivant décrit les propriétés que vous pouvez afficher pour un type simple :

Propriété	Description
Type	Nom de l'élément.
Description	Description de l'élément.
Variété	Définit si le type simple est union, liste, anyType ou atomique. Un élément atomique ne contient pas d'autres éléments ou attributs.
Types de membre	Liste des types dans un UNION de construction.
Type d'élément	Type d'élément.
Base	Type de base d'un élément atomique, comme entier ou string.
Longueur minimale	Longueur minimale pour un élément. La longueur est en octets, caractères ou éléments en fonction du type d'élément.
Longueur maximale	Longueur maximale pour un élément. La longueur est en octets, caractères ou éléments en fonction du type d'élément.
Réduire l'espace vide	Supprime un espace vide de début ou de fin. Réduit plusieurs espaces à un espace unique.
Énumérations	Restreindre le type de la liste des valeurs juridiques.
Modèles	Restreindre le type de valeurs définies par une expression de modèle.

Propriétés avancées de type simple

Pour afficher les propriétés avancées d'un type simple, sélectionnez le type simple dans la vue **Schéma**. Cliquez sur **Avancé**.

Les propriétés avancées s'affichent sous les propriétés de type simple.

Le tableau suivant décrit la propriété avancée pour un type simple :

Propriété	Description
Finale	Empêche le schéma d'étendre ou de restreindre le type simple comme un type dérivé.

Propriétés du type complexe

Un type complexe est un élément qui contient d'autres éléments et attributs. Un type complexe contient les éléments qui sont de types simple ou complexes. Lorsque vous sélectionnez un type complexe dans la vue **Schéma**, l'outil Developer répertorie les éléments enfants et les propriétés de l'élément enfant dans le panneau de droite de l'écran.

Le tableau suivant décrit les propriétés de type complexes :

Propriété	Description
Nom	Nom du type.
Description	Description du type.
Hérité de	Nom du type parent.
Hérité par	Restriction ou extension. Un type complexe est dérivé depuis un type parent. Le type complexe peut réduire les éléments ou les attributs du parent. Ou bien, il peut ajouter des éléments et attributs.

Pour afficher les propriétés de chaque élément dans un type complexe, double-cliquez sur la flèche dans la colonne de description pour développer la fenêtre.

Propriétés avancées de type complexe

Pour afficher les propriétés avancées pour un type complexe, sélectionnez l'élément dans la vue **Schéma**. Cliquez sur **Avancé**.

Le tableau suivant décrit les propriétés avancées pour un élément ou type complexe :

Propriété	Description
Résumé	L'élément est un type abstrait. Un message SOAP doit inclure le type dérivé de ce type. Un type abstrait n'est pas valide sans les types d'élément dérivés.
Bloc	Empêche un élément dérivé d'apparaître dans la hiérarchie à la place de cet élément. La valeur de bloc peut contenir "# all" ou une liste qui comprend l'extension, la restriction ou la substitution.
Finale	Empêche le schéma d'étendre ou de restreindre le type simple comme un type dérivé.
Groupe de substitution	Nom d'un élément à substituer avec l'élément.
Nullible	L'élément peut avoir les valeurs nil. Un élément nil possède des balises d'élément mais n'a aucune valeur, ni aucun contenu.

Propriétés de l'attribut

Un attribut est un type simple. Les éléments et les types complexes contiennent des attributs. Les attributs globaux s'affichent comme une partie du schéma. Lorsque vous sélectionnez un attribut global dans la vue **Schéma**, l'outil Developer répertorie les propriétés de l'attribut et les propriétés du type dans le panneau de droite de l'écran.

Le tableau suivant décrit les propriétés de l'attribut :

Propriété	Description
Nom	Nom de l'attribut.
Description	Description de l'attribut.
Type	Type de l'attribut.
Valeur	Valeur du type de l'attribut. Indique si la valeur du type d'attribut est fixe ou possède une valeur par défaut. Si aucune valeur n'est définie, la propriété affiche la valeur par défaut = 0.

Le tableau suivant décrit les propriétés du type :

Propriété	Description
Longueur minimale	Longueur minimale du type. La longueur est en octets, caractères ou éléments en fonction du type.
Longueur maximale	Longueur maximale du type. La longueur est en octets, caractères ou éléments en fonction du type.
Réduire l'espace vide	Supprime un espace vide de début ou de fin. Réduit plusieurs espaces à un espace unique.
Énumérations	Restreindre le type de la liste des valeurs juridiques.
Modèles	Restreindre le type de valeurs définies par une expression de modèle.

Vue avancée de l'objet du schéma

Affichez les propriétés avancées de l'objet de schéma.

Le tableau suivant décrit les propriétés avancées pour un objet de schéma :

Nom	Valeur	Description
elementFormDefault	Qualifié ou non qualifié	Détermine si les éléments doivent posséder un espace de nom. Le schéma qualifie les éléments avec un préfixe ou par une déclaration de l'espace de nom cible. La valeur non qualifiée signifie que les éléments n'ont pas besoin d'un espace de nom.

Nom	Valeur	Description
attributeFormDefault	Qualifié ou non qualifié	Détermine si les attributs déclarés localement doivent posséder un espace de nom. Le schéma qualifie les attributs avec un préfixe ou par une déclaration de l'espace du nom cible. La valeur non qualifiée signifie que les attributs n'ont pas besoin d'un espace de nom.
Emplacement du fichier	Chemin d'accès complet du fichier .xsd	Emplacement du fichier .xsd lorsque vous l'importez.

Création d'un objet de schéma

Vous pouvez importer un fichier de schéma hiérarchique ou un fichier d'exemple pour créer un objet de schéma dans le référentiel.

1. Sélectionnez un projet ou un dossier dans la vue **Explorateur d'objets**.
2. Cliquez sur **Fichier > Nouveau > Schéma**.
La boîte de dialogue **Nouveau Schéma** s'affiche.
3. Pour importer un fichier de schéma, sélectionnez **Créer à partir d'un schéma**, puis accédez à un fichier de schéma hiérarchique et sélectionnez-le.
Vous pouvez entrer un URI ou un emplacement sur le système de fichiers pour y accéder. L'outil Developer tool valide le schéma que vous choisissez. Vérifiez les messages de validation. Vous pouvez sélectionner un fichier de schéma Avro, Parquet, JSON ou .xsd.
Remarque: L'importation peut échouer si l'identifiant URI contient des caractères non anglais. Copiez l'identifiant URI dans la barre d'adresse d'un navigateur. Copiez l'emplacement à partir du navigateur. L'outil Developer tool accepte les identifiants URI codés provenant du navigateur.
4. Pour créer un schéma à partir d'un fichier d'exemple, sélectionnez **Créer à partir d'un fichier d'exemple**, puis accédez à un fichier hiérarchique et sélectionnez-le.
Vous pouvez sélectionner un fichier Avro, Parquet, JSON ou XML.
Remarque: Si vous sélectionnez un fichier portant une autre extension et comprenant du contenu Avro, Parquet, JSON ou XML, l'assistant reconnaît le contenu du fichier.
5. Éventuellement, modifiez le nom de schéma.
6. Cliquez sur **Suivant** pour afficher une liste des éléments et de types dans le schéma.
7. Cliquez sur **Terminer** pour importer le schéma.
Le schéma s'affiche sous Objets de schéma dans la vue **Explorateur d'objets**. L'outil Developer tool stocke le schéma en tant que fichier .xsd.
8. Pour modifier le préfixe généré pour un espace de nom de schéma, sélectionnez l'espace de nom dans la vue **Explorateur d'objets**. Modifiez la propriété du **Préfixe généré** dans la vue **Espace de nom**.

Mises à jour des schémas

Vous pouvez mettre à jour un objet de schéma lorsque des éléments, des attributs, des types ou d'autres composants du schéma changent. Lorsque vous mettez à jour un objet de schéma, l'outil Developer met à jour les objets qui utilisent le schéma.

Vous pouvez mettre à jour un objet de schéma à l'aide des méthodes suivantes :

Synchronisation du schéma.

Synchronisez un objet de schéma lorsque vous mettez à jour les fichiers de schéma en dehors de l'outil Developer. Lorsque vous synchronisez un objet de schéma, l'outil Developer réimporte tous les fichiers de schéma au format .xsd contenant des modifications.

Édition d'un fichier de schéma.

Éditez un fichier de schéma lorsque vous voulez mettre à jour un fichier depuis l'outil Developer. Lorsque vous éditez un fichier de schéma, l'outil Developer ouvre le fichier dans l'éditeur que vous utilisez pour les fichiers .xsd. Vous pouvez ouvrir le fichier dans un autre éditeur ou définir un éditeur par défaut pour les fichiers .xsd dans l'outil Developer.

Vous pouvez utiliser un schéma pour définir les types d'élément d'un service Web. Lorsque vous mettez à jour un schéma inclus dans le WSDL d'un service Web, l'outil Developer met à jour le service Web et le marque comme modifié lorsque vous l'ouvrez. Lorsque l'outil Developer compare le nouveau schéma et l'ancien, il identifie les composants du schéma à l'aide des attributs de nom.

Si aucun attribut de nom n'a changé, l'outil Developer met à jour le service Web avec les modifications de schéma. Par exemple, vous éditez un fichier de schéma depuis l'outil Developer et modifiez l'attribut maxOccurs de l'élément « Item » de 10 à illimité. Lorsque vous enregistrez le fichier, l'outil Developer met à jour l'attribut maxOccurs dans les services Web référençant l'élément « Item ».

Si un attribut de nom a changé, l'outil Developer marque le service Web comme modifié lorsque vous l'ouvrez. Par exemple, vous éditez un fichier de schéma en dehors de l'outil Developer et modifiez le nom d'un type d'élément complexe de « Order » à « CustOrder ». Vous synchronisez ensuite le schéma. Lorsque vous ouvrez un service Web référençant l'élément, l'outil Developer marque d'un astérisque le nom du service Web dans l'éditeur afin d'indiquer que le service Web contient des modifications. L'outil Developer ajoute le type d'élément « CustOrder » au service Web, mais ne supprime pas le type d'élément « Order ». L'outil Developer ne pouvant plus déterminer le type de l'élément « Order », il remplace le type d'élément par xs:string.

Synchronisation du schéma

Vous pouvez synchroniser un objet de schéma lorsque les composants de schéma changent. Lorsque vous synchronisez un objet de schéma, l'outil Developer réimporte les métadonnées de l'objet depuis les fichiers de schéma.

Utilisez la synchronisation de schéma lorsque vous apportez des modifications complexes à l'objet de schéma en dehors de l'outil Developer. Par exemple, il se peut que vous synchronisiez un schéma après avoir entrepris les actions suivantes :

- Apport de modifications à plusieurs fichiers de schéma.
- Ajout ou suppression de fichiers de schéma pour un schéma.
- Modification des éléments importés ou inclus.

L'outil Developer valide les fichiers de schéma avant de mettre à jour l'objet de schéma. Si les fichiers de schéma contiennent des erreurs, l'outil Developer n'importe pas les fichiers.

Pour synchroniser un objet de schéma, cliquez avec le bouton droit de la souris sur l'objet de schéma dans la vue **Explorateur d'objets** et sélectionnez **Synchroniser**.

Éditions d'un fichier de schéma

Vous pouvez éditer un fichier de schéma dans l'outil Developer afin de mettre à jour les composants du schéma.

Éditez un fichier de schéma dans l'outil Developer pour apporter des mises à jour mineures à un petit nombre de fichiers. Par exemple, vous pouvez vouloir apporter l'une des mises à jour mineures suivantes à un fichier de schéma :

- Modifier les attributs minOccurs ou maxOccurs pour un élément.
- Ajouter un attribut à un type complexe.
- Modifier un type d'objet simple.

Lorsque vous éditez un fichier de schéma, l'outil Developer ouvre une copie temporaire du fichier de schéma dans un éditeur. Vous pouvez éditer les fichiers de schéma avec l'éditeur système que vous utilisez pour les fichiers .xsd, ou vous pouvez en sélectionner un autre. Vous pouvez également paramétrer l'éditeur par défaut de l'outil Developer pour les fichiers .xsd. Enregistrez le fichier de schéma temporaire après l'avoir édité.

L'outil Developer valide le fichier temporaire avant de mettre à jour l'objet de schéma. Si le fichier de schéma contient des erreurs ou des composants entrant en conflit avec d'autres fichiers de schéma de l'objet de schéma, l'outil Developer ne l'importe pas.

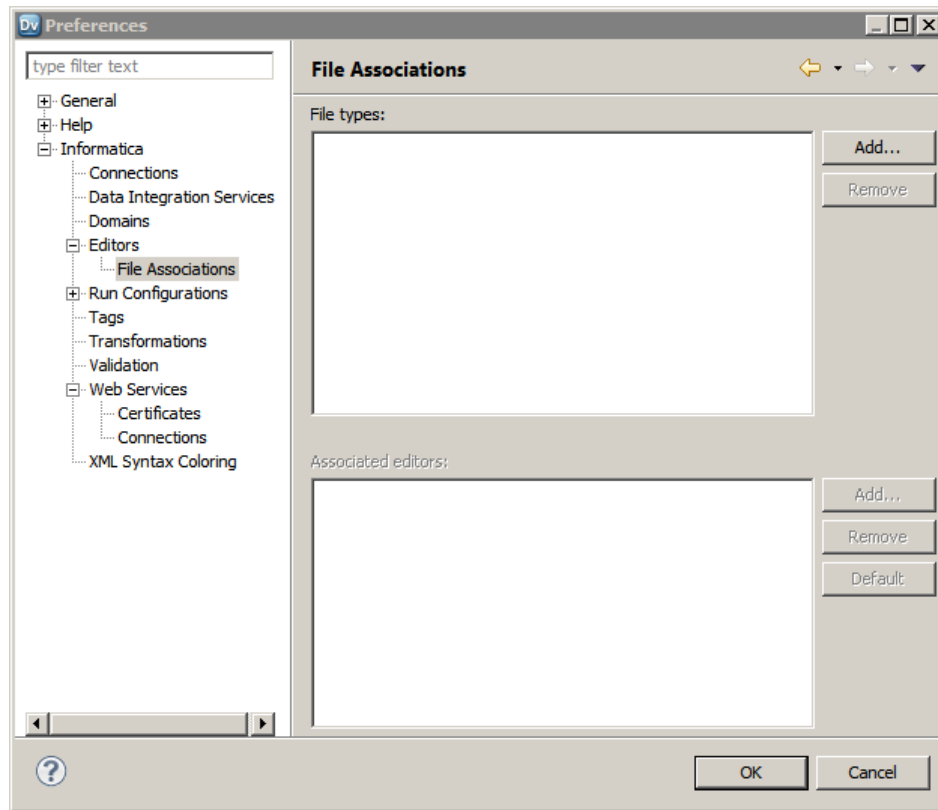
Remarque: Lorsque vous éditez et enregistrez le fichier de schéma temporaire, l'outil Developer ne met pas à jour le fichier de schéma qui apparaît dans la liste **Emplacements des schémas**. Si vous synchronisez un objet de schéma après avoir édité un fichier de schéma dans l'outil Developer, l'opération de synchronisation écrase vos modifications.

Configuration d'un éditeur de fichier de schéma par défaut

Vous pouvez configurer l'éditeur par défaut que l'outil Developer ouvre lorsque vous éditez un fichier de schéma.

1. Cliquez sur **Fenêtre > Préférences**.
La boîte de dialogue **Préférences** s'affiche.
2. Cliquez sur **Éditeurs > Associations de fichiers**.

La page **Associations de fichiers** de la boîte de dialogue **Préférences** s'affiche.



3. Cliquez sur **Ajouter** en regard de la zone **Types de fichiers**.
La boîte de dialogue **Ajouter un type de fichier** s'affiche.
4. Entrez `.xsd` comme type de fichier et cliquez sur **OK**.
5. Cliquez sur **Ajouter** en regard de la zone **Éditeurs associés**.
La boîte de dialogue **Sélection d'un éditeur** s'affiche.
6. Sélectionnez un éditeur dans la liste ou cliquez sur **Parcourir** pour en sélectionner un autre, puis cliquez sur **OK**.
L'éditeur sélectionné s'affiche dans la liste **Éditeurs associés**.
7. Ajoutez éventuellement d'autres éditeurs à la liste **Éditeurs associés**.
8. Si vous ajoutez plusieurs éditeurs, vous pouvez changer d'éditeur par défaut. Sélectionnez un éditeur et cliquez sur **Définir par défaut**.
9. Cliquez sur **OK**.

Édition d'un fichier de schéma

Vous pouvez éditer les fichiers de schéma d'un objet de schéma.

1. Ouvrez un objet de schéma.
2. Sélectionnez la vue **Présentation**.

La vue **Présentation** de l'objet de schéma s'affiche.

3. Sélectionnez un fichier de schéma dans la liste **Emplacements des schémas**.
4. Cliquez sur **Ouvrir avec** et sélectionnez l'une des options suivantes :

Option	Description
Éditeur système	Le fichier de schéma s'ouvre dans l'éditeur utilisé par votre système d'exploitation pour les fichiers .xsd.
Éditeur par défaut	Le fichier de schéma s'ouvre dans l'éditeur que vous définissez comme éditeur par défaut dans l'outil Developer. Cette option apparaît si vous définissez un éditeur par défaut.
Autre	Vous sélectionnez l'éditeur dans lequel ouvrir le fichier de schéma.

L'outil Developer ouvre une copie temporaire du fichier de schéma.

5. Mettez à jour le fichier de schéma temporaire, enregistrez les modifications et fermez l'éditeur.

L'outil Developer vous invite à mettre à jour l'objet de schéma.

6. Pour mettre à jour l'objet de schéma, cliquez sur **Mettre à jour l'objet de schéma**.

L'outil Developer met à jour le fichier de schéma avec les modifications que vous avez apportées.

CHAPITRE 9

Interface de ligne de commande

Ce chapitre comprend les rubriques suivantes :

- [Présentation de l'interface de ligne de commande, 136](#)
- [CM_console, 136](#)

Présentation de l'interface de ligne de commande

Vous pouvez exécuter un service de transformation de données depuis la ligne de commande de la machine qui héberge le service.

Exportez une transformation Processeur de données en tant que service dans le répertoire `/ServiceDB` de la machine sur laquelle vous souhaitez exécuter le service de transformation de données. Exécutez la commande `CM_console`.

CM_console

Exécute un service de transformation de données.

La commande `CM_console` utilise la syntaxe suivante :

```
CM_console <ServiceName>
[< -f | -u | -t >InputDocument]
[ -aServiceParameter=InitialValue]
[ -o<[Path]FileName | FileName>]
[ -r<curr | res | spec=OutputDirectory | guid>]
[ -lUserName -pPassword]
[ -v]
[ -S]
[ -x<f | u | t>InputPortName=InputDocument]
[ -xoOutputPortName=OutputDocument]
[ -e]
```


Remarque: Ne pas inclure un espace entre une option et son argument.

Le tableau suivant décrit les options et arguments de CM_console :

Option	Argument	Description
-	ServiceName	Obligatoire. Indique le nom du service.
-f	InputDocument	Facultatif. Indique un chemin et un nom de fichier sur le système de fichiers local. Par défaut, le service utilise le document défini dans la propriété example_source du composant de démarrage.
-t	InputDocument	Facultatif. Indique une chaîne entourée par des guillemets doubles.
-u	InputDocument	Facultatif. Spécifie une URL.
-a	ServiceParameter=InitialValue	Facultatif. Indique un paramètre d'entrée pour le service. ServiceParameter est le nom d'une variable telle que définie dans le service. InitialValue doit être d'un type de données qui est valide pour la variable définie. Vous pouvez entrer plusieurs paramètres d'entrée, séparés par des espaces.
-o	FileName [Path]FileName	Facultatif. Dirige la sortie vers Path/FileName. Si vous entrez uniquement FileName, vous devez définir le chemin avec l'option -r. Par défaut, la commande CM_console dirige la sortie vers l'écran.
-r	curr	Facultatif. Indique le répertoire dans lequel vous avez exécuté la commande CM_console.
-r	res	Facultatif. Indique le sous-répertoire <i>results</i> sous le répertoire qui contient le service dans le référentiel du système de fichiers.
-r	spec=OutputDirectory	Facultatif. Indique un répertoire sur le système de fichiers local.
-r	guid	Facultatif. Indique un répertoire avec un nom unique dans le répertoire CMReports/tmp. Vous pouvez utiliser l'éditeur de configuration pour changer l'emplacement de ce répertoire.
-l	UserName	Requis lorsque vous utilisez l'authentification HTTP. Indique le nom d'utilisateur pour l'authentification HTTP. Remarque: Cette option est un L minuscule.
-p	Mot de passe	Requis lorsque vous utilisez l'authentification HTTP. Indique le mot de passe pour l'authentification HTTP.
-v	-	Facultatif. Affiche en clair des informations sur la version de transformation de données, la version de la syntaxe de transformation de données, l'identifiant du package d'installation, la licence et d'autres informations.
-S	-	Requis si le composant de démarrage du service est un répartiteur. Vous devez également utiliser l'option -f pour définir le fichier d'entrée.
-xf	InputPortName=InputDocument	Facultatif. InputPortName indique le nom d'un AdditionalInputPort défini dans le service. InputDocument indique un chemin et un nom de fichier sur le système de fichiers local. Vous pouvez entrer plusieurs ports d'entrée, séparés par des espaces.

Option	Argument	Description
-xt	InputPortName=InputDocument	Facultatif. InputPortName indique le nom d'un AdditionalInputPort défini dans le service. InputDocument indique une chaîne entourée par des guillemets doubles. Vous pouvez entrer plusieurs ports d'entrée, séparés par des espaces.
-xu	InputPortName=InputDocument	Facultatif. InputPortName indique le nom d'un AdditionalInputPort défini dans le service. Inputdocument spécifie une URL. Vous pouvez entrer plusieurs ports d'entrée, séparés par des espaces.
-xo	OutputPortName=OutputDocument	Facultatif. OutputPortName indique le nom d'un AdditionalOutputPort défini dans le service. OutputDocument indique un chemin et un nom de fichier sur le système de fichiers local. Vous pouvez entrer plusieurs ports de sortie, séparés par des espaces.
-e	-	Facultatif. Par défaut, la commande CM_console se termine avec un code de sortie de 1 en cas de réussite et supérieure à 1 en cas d'erreur. Lorsque vous incluez l'option -e, la commande CM_console se termine avec un code de sortie de 0 en cas de réussite et supérieure à 1 en cas d'erreur.

Par exemple :

```
CM_console XYZparser -fInputFile.txt -aMaxLines=1000 -oResults.xml -rcurr
```

Cet exemple appelle le service XYZparser, en utilisant InputFile.txt en tant que document d'entrée principal. Il donne la valeur 1000 au paramètre *MaxLines* et écrit la sortie dans le fichier Results.xml dans le répertoire dans lequel vous avez exécuté la commande CM_console.

CHAPITRE 10

Scripts

Ce chapitre comprend les rubriques suivantes :

- [Présentation des scripts, 139](#)
- [Composants de script, 140](#)
- [Propriétés du composant de script, 142](#)
- [Composants de démarrage de script, 144](#)
- [Sources d'exemple, 144](#)
- [Éditeur IntelliScript, 146](#)
- [Valider un Script, 147](#)
- [Échantillons de scripts, 147](#)

Présentation des scripts

Un script exécute des transformations complexes sur les données d'entrée et écrit les données de sortie. Créez un script dans l'onglet **Objets** de la transformation Processeur de données. Utilisez l'éditeur IntelliScript pour afficher un script, ajouter et configurer des composants et définir le composant de démarrage d'un script.

Utilisez un script pour lire un ou plusieurs documents de n'importe quel format, par exemple HL7, PDF, XML ou Word. Vous pouvez enregistrer un ou plusieurs documents dans n'importe quel format. Vous pouvez écrire la sortie d'un script dans le système de fichiers local ou vous pouvez renvoyer la sortie via les ports de sortie de la transformation Processeur de données.

Un script est fait de composants qui définissent les documents d'entrée et de sortie, la logique d'entreprise, les variables contenant les données de manière temporaire et les paramètres de configuration. Les composants sont organisés dans une arborescence hiérarchique. Quand la transformation exécute un script, elle commence le traitement dans le composant que vous définissez en tant que composant de démarrage.

Lorsque vous configurez un script, vous définissez des sources d'exemples qui contiennent des échantillons de données pour chaque port d'entrée. Lorsque vous exécutez la transformation depuis la vue **Visionneuse de données**, la transformation lit les documents de la source d'exemple. Lorsque vous exécutez la transformation dans un mappage, la transformation lit les documents qu'elle reçoit via ses ports d'entrée.

La transformation Processeur de données qui contient le script doit faire référence à un schéma pour chaque document XML que le script lit ou écrit.

Composants de script

Un composant de script est une ligne ou un groupe de lignes d'un script qui définissent les documents d'entrée et de sortie, la logique d'entreprise, les variables qui stockent temporairement des données et les paramètres de configuration. Les composants d'un script s'affichent dans une arborescence hiérarchique. Certains composants s'affichent au niveau global du script. D'autres s'affichent en tant que composants enfants.

Le niveau global du script contient des composants de démarrage, des variables, ainsi que d'autres composants, tels que des ports d'entrée et des transformateurs supplémentaires. Un composant au niveau global doit avoir un nom.

Un composant peut avoir des propriétés qui régissent le comportement du composant. Les propriétés d'un composant s'affichent imbriquées à l'intérieur du composant. Une propriété peut s'afficher sur une ligne ou comme une hiérarchie de propriétés. Vous pouvez configurer les propriétés de certains composants pour remplacer les paramètres par défaut qui s'appliquent à la transformation Processeur de données.

Certains composants, tels que des analyseurs ou des mappeurs, peuvent contenir d'autres composants, comme des transformateurs ou des actions **RunParser**. Éventuellement, vous pouvez configurer la propriété **nom** d'un composant enfant.

Types de composants

Le contexte du script détermine les types de composants que vous pouvez ajouter.

Par exemple, les ancres doivent être imbriquées dans des analyseurs, des mappeurs ou des sérialiseurs. En outre, des ports d'entrée et des ports de sortie supplémentaires ne peuvent apparaître qu'au niveau global du script.

Le tableau suivant décrit les types de composants que vous pouvez ajouter à un script :

Type de composant	Description
Action	Prend des données dans une zone de stockage des données et exécute une opération sur elles. Par exemple, l'action RunParser exécute un analyseur.
Ancre	Identifie une section du document d'entrée.
Processeur de document	Exécute une transformation complexe sur un document d'entrée. Par exemple, le processeur de document PdfToTxt_4 convertit un document PDF en texte brut.
Format	Définit le format des documents devant être traités par l'analyseur.
Localisateur	Isole une seule occurrence d'une zone de stockage des données à occurrences multiples.
Mappeur	Lit et écrit des documents XML. Peut être défini en tant que composant de démarrage.
Notification	Écrit un message dans la sortie standard ou dans un journal. Par exemple, la notification XsdValidationError indique que le document d'entrée n'est pas valide en comparaison au schéma qui le définit.
Analyseur	Lit et écrit des documents dans n'importe quel format. Peut être défini en tant que composant de démarrage.
Port de script	Définit un document d'entrée ou de sortie.

Type de composant	Description
Sérialiseur	Lit des documents XML et écrit des documents dans n'importe quel format. Peut être défini en tant que composant de démarrage.
Répartiteur	Fractionne de grands fichiers d'entrée en fragments et transmet ces fragments à un analyseur, un mappeur ou un sérialiseur. Peut être défini en tant que composant de démarrage.
Transformateur	Transforme une chaîne d'entrée en une chaîne de sortie. Peut être défini en tant que composant de démarrage.
Valideur	Détermine si les données d'entrée sont conformes ou non à une définition de données spécifique.
Variable	Contient des données que le script reçoit via un paramètre de service ou issues du composant du script.

Noms des composants

Le nom d'un composant l'identifie dans le script, la vue **Événements du processeur de données** et le journal.

Lorsque la transformation Processeur de données effectue les instructions contenues dans un composant, le composant génère un événement qui s'affiche dans la vue **Événements du processeur de données** et le journal. Un composant qui apparaît au niveau global du script doit avoir un nom. Un composant qui s'affiche comme composant enfant d'un autre composant peut avoir un nom que vous configurez avec la propriété **nom**.

Le nom d'un composant doit commencer par une lettre, doit contenir uniquement des caractères anglais (A-Z, a-z), des chiffres (0-9) ou des traits bas (_) et ne doit pas contenir plus de 127 caractères.

Ajout d'un composant global

Définissez un composant de manière globale lorsque vous devez l'utiliser à deux ou plusieurs endroits dans le script ou lorsque le composant peut uniquement apparaître au niveau global.

1. En bas du niveau global du script, double-cliquez sur l'ellipse située à gauche (...).
Une zone de texte s'affiche.
2. Entrez le nom du composant, puis appuyez sur **ENTRÉE**.
3. Double-cliquez sur l'ellipse située à droite.
Une zone de liste s'affiche.
4. Cliquez sur la flèche vers le bas et sélectionnez le type de composant que vous voulez ajouter.
Le composant global apparaît dans le script.
5. Définissez les propriétés du composant, le cas échéant.

Ajout d'un composant local

Définissez un composant localement lorsque vous prévoyez de l'utiliser dans un seul emplacement du script ou lorsque le composant peut uniquement apparaître comme un composant enfant.

1. À l'endroit du script où vous voulez insérer un composant, double-cliquez sur l'ellipse.
Une zone de liste s'affiche.
2. Cliquez sur la flèche vers le bas à droite de la zone de liste.
La liste des composants disponibles s'affiche, comprenant notamment les composants de niveau global nommés.
3. Sélectionnez un composant.
Le composant apparaît dans le script.
4. Définissez les propriétés du composant, le cas échéant.

Propriétés du composant de script

Les propriétés d'un composant de script définissent la fonctionnalité du composant. Un composant peut avoir une ou plusieurs propriétés. Les propriétés apparaissent imbriquées dans le composant. Tous les composants de même type ont les mêmes propriétés.

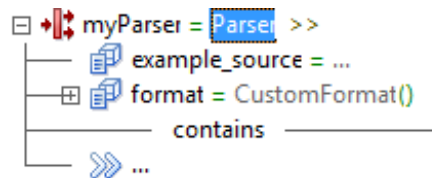
Par exemple, la propriété **example_source** d'un analyseur définit l'exemple de texte que l'analyseur utilise lorsque vous exécutez la transformation depuis la vue **Visionneuse de données**.

Propriétés simples

Les propriétés simples d'un composant sont celles que l'éditeur IntelliScript affiche en permanence.

La plupart des utilisateurs doivent modifier uniquement les propriétés simples.

La figure suivante présente les propriétés simples d'un composant **Analyseur** :



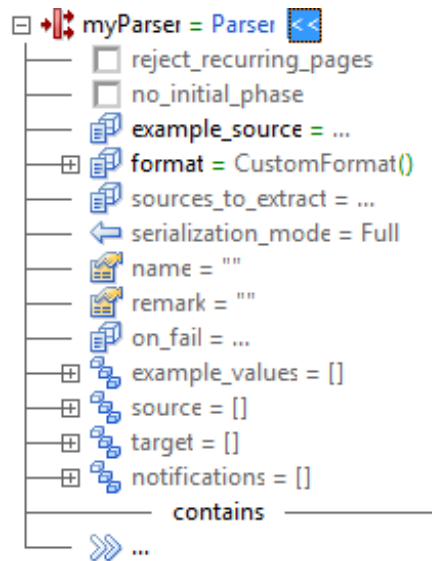
Propriétés avancées

Les propriétés avancées d'un composant sont généralement définies sur les valeurs par défaut que les utilisateurs ne changent habituellement pas.

L'éditeur IntelliScript affiche normalement uniquement les propriétés avancées que vous avez définies sur une valeur qui n'est pas la valeur par défaut.

Pour afficher les propriétés qui ne sont pas affichées, cliquez sur la double flèche vers la droite sur la première ligne.

La figure suivante affiche toutes les propriétés d'un composant **Analyseur** :



Valeurs de la propriété de composant

Vous définissez les valeurs des propriétés d'un composant.

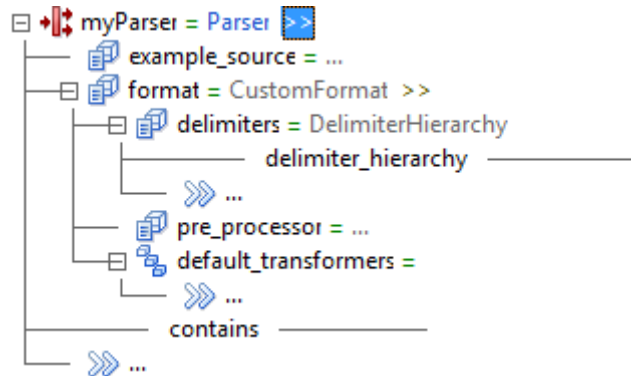
Lorsque la valeur d'une propriété a un type booléen, elle apparaît comme une case à cocher à gauche du nom de propriété. Par exemple, la propriété **facultatif** du composant **Contenu** est de type booléen.

Quand la valeur est une chaîne, elle apparaît à droite du nom de la propriété, entourée de guillemets doubles. Les valeurs valides sont des chaînes de caractères alphanumériques valides, de symboles ou de caractères de contrôle, mais n'incluant pas de caractères nuls. Pour entrer un caractère ne figurant pas sur le clavier dans un champ de texte, appuyez sur **CTRL+A**, puis saisissez le code décimal à trois chiffres du caractère. Par exemple, pressez **CTRL+A** 010 pour un saut de ligne ou **CTRL+A** 255 pour la lettre islandaise "thorn" (þ). Par exemple, la valeur de la propriété de l'**expression** d'un composant **CalculateValue** est une chaîne.

Quand la valeur est une sélection, elle apparaît à droite du nom de la propriété. Quand vous modifiez la valeur, une zone de liste s'affiche. Par exemple, la propriété **val_type** d'un composant **Variable** est une sélection.

Quand la valeur est une arborescence hiérarchique de propriétés, elle apparaît à droite et sous le nom de la propriété. Par exemple, lorsque vous définissez la propriété **format** d'un composant **Analyseur** sur CustomFormat, une arborescence de propriétés supplémentaires s'affiche.

La figure suivante montre la propriété **format** d'un composant **Analyseur** qui s'affiche comme une arborescence :



Composants de démarrage de script

Le composant de démarrage d'un script définit le point d'entrée à partir duquel la transformation Processeur de données commence à exécuter le script. Le composant de démarrage doit apparaître au niveau global du script.

Vous pouvez définir un analyseur, un mappeur, un sérialiseur, un répartiteur ou un transformateur en tant que composant de démarrage.

Vous pouvez définir le composant de démarrage depuis l'onglet **Présentation** de la transformation Processeur de données. Lorsque vous utilisez l'éditeur IntelliScript pour définir le composant de démarrage d'un script, le composant de démarrage du script devient le composant de démarrage de la transformation Processeur de données.

Définition du composant de démarrage avec l'éditeur IntelliScript

Vous pouvez utiliser l'éditeur IntelliScript pour définir un composant de script en tant que composant de démarrage de la transformation Processeur de données. Vous devez définir le composant de démarrage pour exécuter le script. Vous devez définir le composant de démarrage pour afficher la source d'exemple dans le panneau **Entrée** de la vue **Visionneuse de données**.

1. Ouvrez un script dans l'éditeur IntelliScript.
2. Effectuez un clic droit sur l'un des composants qui apparaissent au niveau global du script, puis sélectionnez **Définir comme composant de démarrage**.

Sources d'exemple

Une source d'exemple est un document qui contient des exemples de données d'entrée que le script doit traiter lors de la conception. Vous configurez un exemple de source pour chaque analyseur, mappeur,

sérialiseur ou port d'entrée supplémentaire. La source d'exemple contient le même type de données que les données que la transformation Processeur de données reçoit depuis un port d'entrée.

Par défaut, la vue **Visionneuse de données** affiche la source d'exemple définie pour le composant de démarrage. Vous pouvez également afficher la source d'exemple de tout autre composant qui définit une source d'exemple. Lorsque vous exécutez un script depuis la vue **Visionneuse de données**, la transformation Processeur de données lit les documents source d'exemple.

Vous pouvez configurer les types suivants de documents source d'exemple :

- LocalFile. Un fichier sur le système de fichiers local.
- Texte. Une chaîne codée en dur dans le script.
- URL. Un fichier sur le réseau local ou Internet.

Remarque: Lorsque vous exécutez un script dans un mappage et qu'un document d'entrée est manquant, la transformation utilise la source d'exemple. Si aucune source d'exemple n'est configurée et qu'aucun document d'entrée n'est présent, la transformation Processeur de données s'arrête et génère une erreur fatale.

Exemple de source d'exemple

Le texte de l'exemple suivant illustre une partie d'un fichier local que vous pouvez utiliser pour une source d'exemple lorsque vous analysez des documents HL7 :

```
MSH|^~\&|ADT1|MCM|FINGER|MCM|198808181126|SECURITY|ADT^A01|MSG00001|P|2.3.1
EVN|A01|198808181123
PID|1||PATID1234^5^M11^ADT1^MR^MCM~123456789^^^USSA^SS||
SMITH^WILLIAM^A^III||19610615|M||C|1200 N ELM STREET^^JERUSALEM^TN^999999?
1020|GL|(999)999?1212|(999)999?3333||S||PATID12345001^2^M10^ADT1^AN^A|
123456789|987654^NC
NK1|1|SMITH^OREGANO^K|WI^WIFE||||NK^NEXT OF KIN
PV1|1|I|2000^2012^01|||004777^CASTRO^FRANK^J.|||SUR|||ADM|AO
```

Surlignage de source d'exemple

Le panneau **Entrée** de la vue **Visionneuse de données** surligne des parties de l'échantillon de document source.

La vue **Visionneuse de données** utilise des couleurs différentes pour surligner les ancres contenu de la source d'exemple, les ancres marqueur qui définissent où la transformation trouve le contenu et les groupes de répétition d'ancres.

Définition d'une source d'exemple dans l'éditeur IntelliScript

Lorsque vous exécutez un script depuis la vue **Visionneuse de données**, vous devez avoir un exemple de source pour l'entrée principale et pour chaque port d'entrée supplémentaire. Définissez la source d'exemple dans l'éditeur IntelliScript. Vous pouvez également sélectionner la source d'exemple lorsque vous créez un script dans la transformation Processeur de données.

1. Sélectionnez le composant pour lequel vous souhaitez définir une source d'exemple, puis développez-le pour afficher ses propriétés.
2. En regard de la propriété **exemple_source**, double-cliquez sur l'ellipse.

- Sélectionnez un format d'entrée. Le tableau suivant décrit les options de format d'entrée :

Option	Description
LocalFile	La propriété file_name s'affiche sous la propriété example_source . Double-cliquez sur l'ellipse, puis accédez à un fichier du système de fichiers local.
Text	La propriété quote s'affiche sous la propriété example_source . Entrez une chaîne.
URL	La propriété stable_URL s'affiche sous la propriété example_source . Entrez une chaîne.

Affichage d'une source d'exemple

Vous pouvez afficher la source de l'exemple d'un analyseur, d'un mappeur, d'un sérialiseur ou d'un port d'entrée supplémentaire dans le panneau **Entrée** de la vue **Visionneuse de données**.

- Ouvrez un script dans l'éditeur IntelliScript.
- Définissez l'un des composants du script comme composant de démarrage.
- Dans l'éditeur IntelliScript, sélectionnez le composant qui possède l'exemple de source que vous voulez afficher.
- Dans la vue **Visionneuse de données**, cliquez sur **Synchroniser avec l'éditeur**.

Éditeur IntelliScript

L'éditeur IntelliScript est un outil graphique que vous utilisez pour modifier les scripts. Utilisez l'éditeur IntelliScript pour ajouter des composants au script, configurer les propriétés du composant et définir le composant de démarrage.

Quand vous ouvrez un script, l'éditeur IntelliScript s'affiche dans la zone de l'éditeur, au centre de l'interface de l'outil Developer. Par défaut, l'éditeur IntelliScript affiche les scripts en mode Intelli, faisant apparaître le script dans un format d'arborescence hiérarchique pouvant être développé, ou en mode Script, faisant apparaître le script sous forme de texte. Vous pouvez afficher ou modifier un script en mode Intelli. Certaines propriétés avancées sont masquées par défaut, mais vous pouvez les afficher en cliquant sur une flèche double graphique sur la première ligne du composant.

Vous pouvez insérer uniquement les composants valides pour le contexte. Pour le déplacer, vous pouvez faire glisser un composant ou effectuer un "copier/coller" avec **CTRL+C** et **CTRL+V**. Vous pouvez sélectionner plusieurs composants en faisant un clic de souris tout en pressant les touches **CTRL** et **SHIFT**.

Lorsque vous utilisez l'éditeur IntelliScript, les vues suivantes affichent les informations pertinentes :

- Visionneuse de données, panneau Entrée. Affiche la source d'exemple pour le composant de démarrage ou le composant sélectionné dans l'éditeur IntelliScript.
- Visionneuse de données, panneau Sortie. Affiche la sortie lorsque vous exécutez la transformation Processeur de données depuis la vue **Visionneuse de données**.
- Événements du processeur de données. Affiche les événements qui se produisent lorsque vous exécutez une transformation Processeur de données. Utilisez la vue **Événements Processeur de données** pour le dépannage.
- Aide de script du processeur de données. Affiche la documentation pertinente pour le composant ou la propriété actuellement sélectionnés dans l'éditeur IntelliScript.

- Source Hex du processeur de données. Affiche le document de la source d'exemple sous forme hexadécimale. Utilisez la vue **Source Hex du processeur de données** pour trouver des caractères non imprimables, tels que des tabulations.

Pour afficher la source d'un script, effectuez un clic droit dans l'éditeur IntelliScript, puis sélectionnez **Mode Script**. Pour revenir au mode Intelli, faites un clic droit dans l'éditeur IntelliScript, puis sélectionnez **Mode Intelli**.

Valider un Script

Lorsque vous créez un Script, vous pouvez le valider avant de l'exécuter. Lors de la validation d'un Script, l'outil Developer tool vérifie si des échecs peuvent empêcher un composant de traiter des données comme prévu.

Pour valider un Script, dans la vue **Structure** de l'outil Developer tool, sélectionnez le Script, puis cliquez avec le bouton droit de la souris et sélectionnez **Valider**. En cas d'erreurs, la vue **Journal de validation** apparaît et affiche des erreurs ou des avertissements sur les échecs que le processeur de validation a découverts.

Si vous double-cliquez sur une erreur dans la vue **Journal de validation**, la ligne pertinente dans le Script est mise en surbrillance dans l'éditeur IntelliScript.

Échantillons de scripts

Informatica fournit des exemples de scripts pour vous montrer des exemples de tâches qui peuvent être réalisées avec un script.

Vous pouvez trouver les exemples de scripts dans le sous-dossier suivant du répertoire d'installation :

\DataTransformation\samples\Projects

Pour afficher, modifier ou copier un exemple de script, vous devez d'abord l'importer.

Le tableau suivant décrit des exemples de scripts :

Nom du script	Description
Alternatives	Démontre la ramification et l'ancre Alternatives .
AppendListItems	Concatène les chaînes dans une zone de stockage des données à occurrences multiples et démontre l'action AppendListItems .
CalculateValue	Effectue un calcul numérique complexe et fait la démonstration de l'action CalculateValue .
CombineValues	Concatène des chaînes et fait la démonstration des actions CombineValues et DumpValue .
Contenu	Fait la démonstration de l'ancre Contenu et du contenu trouvé dans le document source en recherchant une chaîne spécifique, en calculant un décalage depuis la dernière ancre et en recherchant un attribut dans une paire nom=valeur.

Nom du script	Description
CopyValue	Copie un élément XML complexe entier avec l'action Mappage .
DelimitedSections	Fait la démonstration de l'ancre DelimitedSections dans un analyseur.
DocumentOrder	Fait la démonstration de la ramification et de l'ancre Alternatives , avec l'option selector définie sur DocumentOrder.
Dynamic_And_RepeatingGroup	Répète les lignes d'un document et fait la démonstration de l'ancre RepeatingGroup . Lit les données depuis un autre emplacement dans le document basé sur le contenu dans la zone de portée actuelle.
EmbeddedParser	Utilise un analyseur secondaire intégré pour analyser le contenu de l'analyseur principal et fait la démonstration de l'ancre EmbeddedParser .
EnsureCondition	Évalue une expression JavaScript booléenne pour sélectionner des alternatives et fait la démonstration de l'action EnsureCondition .
ManualSerializer	Fait la démonstration d'un sérialiseur personnalisé.
Marqueurs	Fait la démonstration des ancres Marqueur qui utilisent les options TextSearch, OffsetSearch, TypeSearch et PatternSearch.
Marking_Mode	Fait la démonstration de plusieurs méthodes de configuration des ancres Marqueur .
NonMarker	Fait la démonstration d'un analyseur qui utilise uniquement des ancres Contenu et la recherche inversée dans le document d'entrée.
Modèle	Fait la démonstration de l'extraction de données qui correspondent à une restriction définie dans le schéma.
persistent_search	Fait la démonstration de la propriété on_partial_match d'un Groupe et de la propriété adjacent d'un Marqueur .
ResetListVariable	Réinitialise une variable liste en utilisant un targetLocator.
RunSerializer	Fait la démonstration d'un analyseur qui appelle un sérialiseur secondaire.
HL7	Convertit un fichier HL7 en XML.
TabDelimited	Convertit un fichier HL7 délimité par des tabulations en XML.
Séparateur	Divise un fichier en deux fichiers et fait la démonstration de l'action WriteValue .
TransformByParser	Utilise un analyseur pour transformer un texte spécifique en retour à la ligne et fait la démonstration de l'action TransformByParser .
Transformers_Example	Fait la démonstration de l'ancre Contenu avec la propriété valeur définie sur LearnByExample .

Importation d'un échantillon de script

Importez un échantillon de script pour l'afficher ou en copier des parties dans un autre script.

1. Cliquez sur **Fichier > Importer**.
La boîte de dialogue **Importer** s'affiche.
2. Sélectionnez **Informatica > Importer le service DT**, puis cliquez sur **Suivant**.
La page **Importer le service DT** s'affiche.
3. En regard du champ **Fichier du service**, cliquez sur le bouton **Parcourir** et accédez au fichier CMW pour le service.
4. Cliquez sur **Terminer**.
L'échantillon de script s'affiche dans la transformation Processeur de données.

CHAPITRE 11

Analyseurs

Ce chapitre comprend les rubriques suivantes :

- [Présentation des analyseurs, 150](#)
- [Analyseurs indépendants de la plateforme, 150](#)
- [Documentation de référence du composant analyseur, 151](#)

Présentation des analyseurs

Les analyseurs sont des composants de script qui lisent les documents source dans n'importe quel format.

La sortie d'un analyseur est toujours au format XML. L'entrée peut avoir tout type de format, par exemple : texte, HTML, Word, PDF ou HL7. L'entrée peut être un document XML que l'analyseur traite comme des données de type chaîne.

Analyseurs indépendants de la plateforme

Les scripts de l'analyseur s'exécutent sous les systèmes Microsoft Windows et UNIX. La plupart des fonctions de l'analyseur s'exécutent aussi bien sur les deux plates-formes.

Il existe quelques exceptions à cette règle. Si vous prévoyez d'exécuter un analyseur sous Windows et UNIX, voici quelques conseils qui vous aideront à assurer l'indépendance vis-à-vis de la plate-forme.

Marqueurs Retour à la ligne

Évitez de définir des ancres **Marqueur** qui recherchent un caractère de saut de ligne suivi par un retour chariot (`\n\r`). Cette combinaison est habituellement utilisée sous Windows, mais rarement dans UNIX.

Au lieu de cela, configurez un **Marqueur** avec le composant intégré **NewlineSearch** qui recherche à la fois la séquence `\n\r` et les caractères `\n` ou `\r` seuls.

Chemins de fichiers

Utilisez des chemins de fichiers relatifs, en opposition aux chemins absolus. Rappelez-vous que les chemins des fichiers sous Unix sont sensibles à la casse.

Documentation de référence du composant analyseur

Un composant **Analyseur** convertit un document source en XML.

Analyseur

Un analyseur lit un document source, quel que soit son format. Vous pouvez ajouter des composants enfant pour effectuer des transformations sur les données.

Définissez des analyseurs au niveau global du script. Définissez un analyseur principal en tant que composant de démarrage. Appelez un analyseur secondaire avec l'action **RunParser**. Pour plus d'informations, voir ["RunParser" à la page 323](#)

Les propriétés de l'**Analyseur** apparaissent au-dessus de la ligne **contient**. Sous la ligne, vous pouvez insérer composants enfant comme des ancres et actions.

Le tableau suivant décrit les propriétés du composant **Analyseur** :

Propriété	Description
example_source	Définit un échantillon de document source à traiter dans l'environnement de développement. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none">- Vide. L'outil Developer vous invite à indiquer un document source lorsque vous exécutez l'analyseur.- InputPort. Définit un port d'entrée.- LocalFile. Définit un fichier sur le système de fichiers local.- Texte. Définit une chaîne.- URL. Définit une URL. La valeur par défaut est Vide. Remarque: Si la propriété sources_to_extract est définie, la propriété example_values est ignorée dans l'environnement de conception.
example_values	Définit des valeurs simulées qu'une autre transformation peut transmettre à l'analyseur. Utilisez cette propriété pour concevoir un analyseur appelé par un autre analyseur. Un analyseur utilise la propriété example_values uniquement lorsqu'il traite l'exemple de source. Il ignore la propriété quand il analyse un document source. Dans les composants ExampleValue imbriqués, spécifiez les zones de stockage des données que l'analyseur appelant transmet à cet analyseur, ainsi que leurs valeurs simulées.
ExampleValue	Définit un exemple de valeur dans la propriété example_values .
format	Indiquez le format du document source. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none">- BinaryFormat- CustomFormat- HtmlFormat- Format Rtf- TextFormat- XmlFormat La valeur par défaut est CustomFormat. Pour plus d'informations, voir "Documentation de référence du composant Format" à la page 179
nom	Libellé descriptif pour le composant. Ce libellé s'affiche dans le fichier journal et la vue Événements . Utilisez la propriété nom pour identifier le composant ayant provoqué l'événement.

Propriété	Description
no_initial_phase	Détermine si le script recherche des ancrs imbriquées dans la phase principale. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none"> - Effacé. Rechercher les ancrs imbriquées selon leurs propriétés individuelles. - Sélectionné. Rechercher les ancrs imbriquées dans la phase principale. Par défaut, la valeur est vide.
notifications	Définit une liste de composants NotificationHandler que l'analyseur exécute sur les notifications déclenchées par des composants imbriqués. Pour plus d'informations, voir "Notifications" à la page 423
on_fail	Action à effectuer si le composant échoue. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none"> - Vide. N'effectuer aucune action. - CustomLog. Écrire dans le journal utilisateur. - LogError. Écrire un message d'erreur dans le journal du moteur. - LogInfo. Écrire un message d'information dans le journal du moteur. - LogWarning. Écrire un message d'avertissement dans le journal du moteur. - NotifyFailure. Envoyer une notification. Par défaut, la valeur est vide. Pour plus d'informations sur la gestion des défaillances de composants, consultez la section "Traitement des échecs" à la page 404 .
reject_recurring_pages	Détermine le nombre de fois où l'analyseur analyse le même page. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none"> - Sélectionné. L'analyseur analyse une page une seule fois. - Effacé. L'analyseur analyse une page à chaque fois qu'il suit un lien vers cette page. Utilisez reject_recurring_pages quand le site Web contient plusieurs liens vers la même page. Remarque: L'action ResetVisitedPages réinitialise la liste de l'historique et permet à un analyseur de traiter à nouveau une page, même si reject_recurring_pages est sélectionné.
remarque	Commentaire défini par l'utilisateur, qui décrit l'objectif ou l'action du composant.
serialization_mode	Définit la façon dont le script traite des portions de l'exemple de source pour lesquelles l'analyseur ne produit pas de sortie XML, lorsque vous créez un sérialiseur depuis un analyseur. Pour plus d'informations, voir "Contrôle de la manière de fonctionner de la commande Créer le sérialiseur" à la page 336 Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none"> - Complet. Contraint la commande CreateSerializer à copier le texte non XML pour la configuration du sérialiseur. - Structure. Contraint la commande CreateSerializer à copier uniquement les délimiteurs de texte non XML dans la configuration du sérialiseur. Quand Structure est sélectionné, vous pouvez définir la propriété use_markers.
source	Définit une séquence de zones de stockage des données à entrer dans l'analyseur. Chaque zone de stockage des données est identifiée par l'une des propriétés suivantes : <ul style="list-style-type: none"> - Localisateur. Identifie une zone de stockage des données à occurrence unique ou à occurrences multiples. Pour les zones de stockage des données à occurrences multiples, chaque itération accède à une nouvelle occurrence. - LocatorByKey. Identifie une zone de stockage des données à occurrences multiples par clé. - LocatorByOccurence. Identifie une zone de stockage des données à occurrences multiples par numéro séquentiel. Dans un analyseur secondaire, définissez Analyseur > source > Localisateur > data_holder sur la zone de stockage des données définie dans les AdditionalInputPort > data_holder associés. Pour plus d'informations, voir "Propriété source" à la page 370

Propriété	Description
sources_to_extract	<p>Définit une liste codée en dur de documents source traités par l'analyseur. Vous pouvez choisir l'une des options suivantes :</p> <ul style="list-style-type: none"> - DocList. Définit une liste de composants LocalFile, Texte et URL. - Vide. L'analyseur traite exemple_source. - FileSearch. Définit un dossier dans le système de fichiers local et un filtre de nom de fichier. - InputPort. Définit un port d'entrée. N'utilisez pas cette option. - LocalFile. Définit un fichier sur le système de fichiers local. - Texte. Définit une chaîne. - URL. Définit une URL. <p>La valeur par défaut est Vide.</p> <p>Remarque: Utilisez la propriété sources_to_extract uniquement dans l'environnement de conception.</p>
cible	<p>Définit une séquence de zones de stockage des données pour la sortie depuis l'analyseur. Si la zone de stockage des données n'existe pas encore, l'analyseur la crée. Chaque zone de stockage des données est identifiée par l'une des propriétés suivantes :</p> <ul style="list-style-type: none"> - Localisateur. Identifie une zone de stockage des données à occurrence unique ou à occurrences multiples. Pour les zones de stockage des données à occurrences multiples, chaque itération crée une nouvelle occurrence. - LocatorByKey. Identifie une zone de stockage des données à occurrences multiples par clé. - LocatorByOccurrence. Identifie une zone de stockage des données à occurrences multiples par numéro séquentiel. <p>Utilisez la propriété cible lorsque la sortie de l'analyseur est utilisée par un autre composant. Pour plus d'informations, voir "Propriété target" à la page 374</p>
use_markers	<p>Détermine si la commande Créer le sérialiseur copie le contenu de l'ancre Marqueur, mais uniquement les délimiteurs des autres textes non XML. use_markers est une option sous la propriété serialization_mode quand Structure est aussi sélectionné. La valeur par défaut est Sélectionné.</p>

CHAPITRE 12

Ports de script

Ce chapitre comprend les rubriques suivantes :

- [Présentation des ports de script, 154](#)
- [Documentation de référence du composant Port de script, 154](#)

Présentation des ports de script

Un port de script spécifie l'entrée ou la sortie d'un script, par exemple un document source ou un document de sortie.

Par exemple, dans un composant **Analyseur**, les valeurs des propriétés **example_source** et **sources_to_extract** sont des ports d'entrée.

Dans certains composants, les ports de script sont définis de manière implicite. Par exemple, le fichier de sortie par défaut d'un analyseur est le fichier `output.xml`. Vous n'avez pas besoin de définir de port de sortie faisant référence au fichier `output.xml`.

Par défaut, chaque script a un port d'entrée et un port de sortie. Vous pouvez configurer des ports d'entrée et de sortie supplémentaires. Lorsque vous créez des ports d'entrée ou de sortie supplémentaires dans un script, l'outil Developer ajoute les ports supplémentaires dans la transformation Processeur de données.

Documentation de référence du composant Port de script

Un composant de port de script spécifie l'entrée ou la sortie d'une transformation, par exemple un document source ou un document de sortie.

AdditionalInputPort

Le port **AdditionalInputPort** définit un port d'entrée supplémentaire.

Le tableau suivant décrit les propriétés du port **AdditionalInputPort** :

Propriété	Description
code_page	Détermine le codage d'entrée pour le port. Quand aucune valeur n'est définie, AdditionalInputPort utilise l'encodage d'entrée défini dans les paramètres de la transformation Processeur de données. Par défaut, la valeur est vide.
data_holder	Définit une zone de stockage des données là où le port stocke le contenu du document d'entrée. Utilisez la même zone de stockage des données dans la propriété Analyseur > source > Localisateur > data_holder de l'analyseur, du mappeur ou du sérialiseur secondaire associé.
désactivé	Détermine si le script ignore le composant et tous les composants enfants. Utilisez cette propriété pour tester, déboguer et modifier un script. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none">- Sélectionné. Le script ignore le composant.- Effacé. Le script applique le composant. La valeur par défaut est vide.
encode_as_xml	Détermine si les caractères spéciaux sont convertis en entités XML. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none">- Sélectionné. Les caractères spéciaux sont convertis en entités XML.- Effacé. Les caractères spéciaux ne sont pas convertis. La propriété encode_as_xml est un enfant de la propriété input_encoding lorsqu'elle est définie sur PortEncoding . La valeur par défaut est Effacé.
example_source	Définit l'emplacement d'une source à traiter lors du test. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none">- LocalFile. Définit un fichier sur l'ordinateur local.- Texte. Définit une chaîne.- URL. Définit l'URL d'une page Web. Avertissement: Ne définissez pas un processeur de document dans la propriété AdditionalInputPort > example_source > pre_processor . Définissez-le dans AdditionalInputPort > pre_processor .
input_encoding	Définit l'encodage de l'entrée.
PortEncoding	Définit les paramètres personnalisés de l'entrée supplémentaire. La propriété PortEncoding possède les options suivantes : <ul style="list-style-type: none">- code_page- encode_as_xml
pre_processor	Définit le nom d'un processeur de document à appliquer à l'entrée avant le processeur de document défini dans RunParser > pre_processor . Pour plus d'informations, consultez la section "Documentation de référence du composant Processeur de document" à la page 164 Avertissement: Ne définissez pas un processeur de document dans la propriété AdditionalInputPort > example_source > pre_processor . Définissez-le dans AdditionalInputPort > pre_processor .

Définissez le port **AdditionalInputPort** au niveau global du script et attribuez-lui un nom.

Exemple de AdditionalInputPort

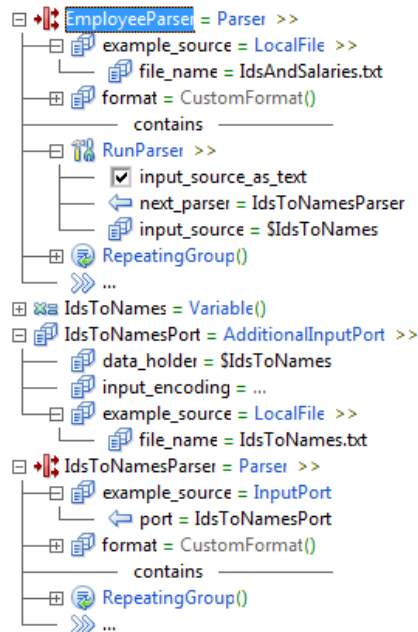
Imaginez que vous ayez deux fichiers textes :

- `IdsAndSalaries.txt` est une table contenant les ID et les salaires des employés.
- `IdsAndNames.txt` est une table de noms et d'identifiants d'employés.

Vous souhaitez analyser ces fichiers conjointement, en générant un fichier de sortie XML contenant les noms et salaires de l'employé. Vous pouvez configurer la transformation de la manière suivante :

- L'analyseur principal, nommé `EmployeeParser`, traite le fichier `IdsAndSalaries.txt`.
- L'analyseur principal active un analyseur secondaire, nommé `IdsToNamesParser`, qui traite le fichier `IdsAndNames.txt` et stocke le résultat dans une table XML.
- L'analyseur principal utilise un transformateur `LookupTransformer` pour convertir les identifiants en noms. La table de recherche est la sortie de l'analyseur secondaire.

La figure ci-dessous présente un exemple de script avec un analyseur secondaire qui référence un port `AdditionalInputPort` pour récupérer le fichier `IdsAndNames.txt` :



AdditionalOutputPort

Le port **AdditionalOutputPort** définit un port de sortie supplémentaire. Utilisez ce composant pour définir la sortie dans plusieurs emplacements ou plusieurs documents.

Le tableau suivant décrit les propriétés du port **AdditionalOutputPort** :

Propriété	Description
add_BOM_prefix	Ajoute un préfixe de marquage d'ordre des octets (bom) à la sortie. Le type de préfixe BOM est déterminé par le codage de sortie défini dans la propriété output_encoding . La valeur par défaut est Effacé.
code_page	Définit l'attribut codage de la sortie supplémentaire. Si cette propriété n'est pas définie, la sortie supplémentaire est générée avec le codage de sortie défini dans les paramètres de la transformation Processeur de données. La propriété code_page est un enfant de la propriété output_encoding lorsqu'elle est définie sur PortEncoding . La valeur par défaut est Effacé.

Propriété	Description
désactivé	Détermine si le script ignore le composant et tous les composants enfants. Utilisez cette propriété pour tester, déboguer et modifier un script. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none"> - Sélectionné. Le script ignore le composant. - Effacé. Le script applique le composant. La valeur par défaut est vide.
encode_as_xml	Détermine si les caractères spéciaux sont convertis en entités XML. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none"> - Sélectionné. Les caractères spéciaux sont convertis en entités XML. - Effacé. Les caractères spéciaux ne sont pas convertis. La propriété encode_as_xml est un enfant de la propriété output_encoding lorsqu'elle est définie sur PortEncoding . La valeur par défaut est Effacé.
file_extension	Définit l'extension du fichier pour le fichier de sortie supplémentaire dans l'environnement de conception. Le nom du fichier est le nom assigné au composant AdditionalOutputPort . Ce paramètre n'a aucun effet dans l'environnement de production. La valeur par défaut est .xml.
other_properties	Définit les propriétés de codage lorsqu'elle est définie sur XmlHeader . Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none"> - XmlHeader. Définit l'en-tête XML. La propriété XmlHeader offre les options suivantes : <ul style="list-style-type: none"> - add_BOM_prefix - process_instruction - process_instruction_string - root_element - xml_version - XSLT_stylesheet_name - Effacé. Les propriétés de sortie sont déterminées par les paramètres de la transformation Processeur de données. La valeur par défaut est Effacé.
output_encoding	Définit les propriétés de codage lorsqu'elle est définie sur PortEncoding . Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none"> - PortEncoding. La sortie supplémentaire a les paramètres personnalisés pour code_page et encode_as_xml. - Effacé. Les paramètres de la transformation Processeur de données contrôlent le codage de sortie et la conversion des entités XML. La valeur par défaut est Effacé.
PortEncoding	Définit les paramètres personnalisés pour la sortie supplémentaire. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none"> - code_page - encode_as_xml
process_instruction	Définit une instruction de traitement dans le fichier XML de sortie. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none"> - Aucun. N'enregistre pas d'instruction de traitement pour la sortie XML. - UseOutputCodePage. Les sorties que la page de code a définies dans la propriété output_encoding. - FreeEncodingString. Les sorties définies par la chaîne dans la propriété process_instruction_string. La valeur par défaut est UseOutputCodePage.
process_instruction_string	Définit une instruction de traitement définie par l'utilisateur. La propriété process_instruction_string ne fait effet que quand la propriété process_instruction est définie sur FreeEncodingString.

Propriété	Description
root_element	Définit le nom de l'élément racine qui est encapsulé autour de la sortie entière.
xml_version	Définit l'attribut version de l'instruction de traitement. Valeur par défaut : 1.0.
XSLT_stylesheet_name	Définit une feuille de style XSLT qui est écrite pour l'instruction de traitement.

Définition d'un port de sortie supplémentaire

1. Au niveau global du script, insérez un composant **AdditionalOutputPort** et attribuez-lui un nom.
2. Dans le composant de démarrage de la transformation Processeur de données, imbriquez une action **WriteValue**, définissez la propriété **sortie** sur **OutputPort** et donnez au **port** le nom du port de sortie supplémentaire.
3. Dans les paramètres de la transformation Processeur de données, sélectionnez **Contrôle de sortie**, puis cochez **Désactiver la sortie automatique**.

Nom de fichier de la sortie supplémentaire

Lorsque vous exécutez la transformation dans l'outil Developer, le système définit un nom de fichier pour la sortie supplémentaire et enregistre le fichier dans le dossier de résultats du projet. Par exemple, si le port est appelé **MyOutputPort**, le nom de fichier peut être `OUTPUT_MyOutputPort.xml`.

Pour déterminer le nom du fichier :

1. Cliquez sur **Exécuter > Exécuter**.
2. Cliquez sur **Détails** pour afficher le tableau **Ports E/S**.

Le tableau affiche le nom de chaque **AdditionalOutputPort** et son fichier de sortie.

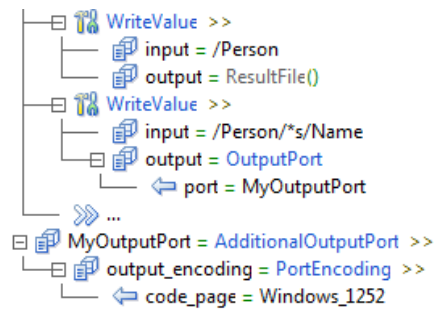
Lors du déploiement de la transformation en tant que service, une application qui exécute le service peut transmettre un emplacement de sortie supplémentaire comme paramètre. Par exemple, l'emplacement peut être un tampon.

Exemple de AdditionalOutputPort

Un analyseur génère la structure XML suivante :

```
<Person gender="M">
  <Name>
    <First>Ron</First>
    <Last>Lehrer</Last>
  </Name>
  <Id>547329876</Id>
  <Age>27</Age>
</Person>
```

La figure suivante présente un analyseur qui utilise deux actions `WriteValue` pour générer la sortie.



La première action `WriteValue` écrit la totalité de l'élément `<Personne>` dans le fichier de résultats par défaut.

```

<Person gender="M">
  <Name>
    <First>Ron</First>
    <Last>Lehrer</Last>
  </Name>
  <Id>547329876</Id>
  <Age>27</Age>
</Person>
  
```

La seconde action `WriteValue` référence un `AdditionalOutputPort` pour écrire l'élément imbriqué `<Name>` dans un autre fichier.

```

<Name>
  <First>Ron</First>
  <Last>Lehrer</Last>
</Name>
  
```

DocList

Le port **DocList** définit une liste des types de ports d'entrée suivants :

- LocalFile
- Texte
- URL

Le tableau suivant décrit les propriétés du port **DocList** :

Propriété	Description
désactivé	Détermine si le script ignore le composant et tous les composants enfants. Utilisez cette propriété pour tester, déboguer et modifier un script. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none"> - Sélectionné. Le script ignore le composant. - Effacé. Le script applique le composant. La valeur par défaut est vide.
pre_processor	Définit le nom du pré-processeur à appliquer aux fichiers d'entrée. Pour plus d'informations, consultez la section "Documentation de référence du composant Processeur de document" à la page 164.

FileSearch

Le port **FileSearch** définit les fichiers d'entrée sur un ordinateur du réseau local. Utilisez le port **FileSearch** dans la propriété `sources_to_extract` d'un **Analyseur**.

Le tableau suivant décrit les propriétés du port **FileSearch** :

Propriété	Description
répertoire	Définit un dossier qui contient les fichiers d'entrée.
désactivé	Détermine si le script ignore le composant et tous les composants enfants. Utilisez cette propriété pour tester, déboguer et modifier un script. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none">- Sélectionné. Le script ignore le composant.- Effacé. Le script applique le composant. La valeur par défaut est vide.
pre_processor	Définit le nom d'un processeur de document à appliquer aux fichiers d'entrée. Pour plus d'informations, consultez la section "Documentation de référence du composant Processeur de document" à la page 164 .
recursive	Détermine si les fichiers d'entrée peuvent être localisés dans des sous-dossiers du dossier spécifié. La valeur par défaut est Effacé.
caractère générique	Définit un critère de filtre des fichiers dans le dossier spécifié. Utilisez * comme caractère générique de remplacement. Par exemple, *.txt trouve tous les fichiers TXT. La valeur par défaut est *.*.

InputPort

Le port **InputPort** définit un port d'entrée nommé qui est défini avec le composant **AdditionalInputPort**.

Le tableau suivant décrit les propriétés du port **InputPort** :

Propriété	Description
entrée	Définit le nom du composant AdditionalInputPort qui définit l'entrée.

LocalFile

Le port **LocalFile** définit un fichier sur le réseau local.

Le tableau suivant décrit les propriétés du port **LocalFile** :

Propriété	Description
file_name	Définit le chemin et le nom d'un fichier sur le réseau local.
pre_processor	Définit le nom d'un processeur de document à appliquer au fichier. Pour plus d'informations, consultez la section "Documentation de référence du composant Processeur de document" à la page 164 .
simulated_url	Définit une URL à assigner au fichier. Cette propriété provoque le traitement du fichier par l'analyseur comme s'il était situé sur un serveur Web. Si le fichier contient des liens relatifs, l'analyseur résout les liens par rapport à l'URL. La partie de l'URL représentant le nom d'hôte n'est pas sensible à la casse.

OutputPort

Le port **OutputPort** définit un port de sortie nommé qui est défini avec le composant **AdditionalOutputPort**. Vous pouvez utiliser un port **OutputPort** dans une action **WriteValue**.

Le tableau suivant décrit les propriétés du port **OutputPort** :

Propriété	Description
port	Détermine le nom du AdditionalOutputPort .

Text

Le port **Text** définit une chaîne de texte qui est utilisée comme entrée de transformation.

Le tableau suivant décrit les propriétés du port **Texte** :

Propriété	Description
pre_processor	Définit le nom d'un processeur de document à appliquer à la chaîne. Pour plus d'informations, voir "Documentation de référence du composant Processeur de document" à la page 164
quote	Définit une chaîne de texte.
simulated_url	Définit une URL à assigner à la chaîne. Cette propriété provoque le traitement de la chaîne par l'analyseur comme s'il s'agissait d'un fichier situé sur un serveur Web. Si la chaîne contient des liens relatifs, l'analyseur résout les liens par rapport à l'URL.
size	Définit une taille fixe pour le tampon de texte. Utilisez la propriété size avec des sources binaires. La valeur par défaut est -1, ce qui signifie que le tampon est dimensionné dynamiquement.

URL

Le port **URL** définit l'URL d'un document disponible sur un serveur Web.

Le tableau suivant décrit les propriétés du port URL :

Propriété	Description
post_data	Définit les données que la transformation poste à l'URL.
pre_processor	Définit le nom d'un processeur de document à appliquer aux fichiers.
retries	Définit le nombre de tentatives effectuées par l'analyseur avant que ce dernier ne signale un échec. La valeur par défaut est 0.
seconds_to_wait	Définit le nombre de secondes d'attente entre les tentatives. La valeur par défaut est 60.
stable_url	Définit une adresse URL qui contient un document d'entrée.

Remarque: Ce composant est fourni à des fins de compatibilité avec des projets créés dans des versions antérieures de Data Transformation. Il va disparaître progressivement du système Data Transformation. Ne

l'utilisez pas lorsque vous développez des transformations. Notez que Data Transformation ne parvient pas à traiter une URL HTTPS dans un environnement Linux.

CHAPITRE 13

Processeurs de document

Ce chapitre comprend les rubriques suivantes :

- [Présentation des processeurs de document, 163](#)
- [Définition d'un processeur de document, 163](#)
- [Documentation de référence du composant Processeur de document, 164](#)
- [Schéma XML TextML, 173](#)
- [Éditeur de configuration de tableau PdfToTxt_4, 174](#)

Présentation des processeurs de document

Les processeurs de document sont des composants qui convertissent le format d'un document complet dans un autre format pour le traitement.

Vous pouvez utiliser un processeur de document en tant que préprocesseur, convertissant le format d'un document source avant une transformation. Par exemple, si le document source d'un analyseur est au format PDF, vous pouvez appliquer le processeur **PdfToTxt_4**. Il convertit le document source en texte, plus facile à analyser que le format binaire PDF.

Ne confondez pas les processeurs de document avec les préprocesseurs de format. Pour plus d'informations sur les préprocesseurs de format, consultez la section ["Présentation des formats" à la page 178](#).

Définition d'un processeur de document

Vous pouvez faire un prétraitement du document source avec n'importe quel processeur de document.

1. Assignez la propriété **example_source** de la transformation. La valeur de **example_source** est un port d'entrée, tel que **LocalFile** ou **Texte**.
2. Assignez la propriété **pre_processor** du port d'entrée.

Le script applique le processeur que vous définissez sous **example_source** à toutes les sources sur lesquelles vous exécutez la transformation.

Remarque: Vous pouvez également définir un préprocesseur dans la propriété **sources_to_extract** de l'analyseur. Le processeur que vous y définissez s'applique uniquement aux documents source que vous définissez dans **sources_to_extract** et à aucun autre document traité par l'analyseur.

Affichage de la sortie du processeur de document

Si vous assignez un processeur de document à l'exemple source, le panneau source de la vue **Visionneuse de données** affiche la sortie du processeur.

Documentation de référence du composant Processeur de document

Les processeurs de documents convertissent un document complet d'un format vers un autre avant son traitement par un analyseur, un mappeur ou un sérialiseur.

AsnToXml

Le processeur de document **AsnToXml** convertit un fichier binaire ASN.1 en XML.

Le tableau suivant décrit les propriétés du processeur de document **AsnToXml** :

Propriété	Description
asn_file	Définit un fichier de spécification ASN.1.
en-tête	Définit un en-tête à exclure du XML. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none">- NewlineSearch. L'en-tête est un symbole newline.- OffsetSearch. L'en-tête est défini par le nombre de caractères depuis le début du fichier.- PatternSearch. L'en-tête est défini par une expression régulière.- TextSearch. L'en-tête est défini par une chaîne explicite ou une chaîne que vous récupérez dynamiquement depuis le document source.
no_constraints	Détermine si le fichier ASN est traité avec des contraintes. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none">- true. Le fichier ASN est traité sans contraintes.- false. Le fichier ASN est traité avec des contraintes. La valeur par défaut est False.
pdu_type	Définit le type PDU. Utilisez cette propriété pour clarifier une ambiguïté.
process_first_message	Détermine si le fichier CDR est traité en entier. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none">- true. Seul le premier enregistrement est traité.- false. La totalité du fichier CDR est traitée. La valeur par défaut est False.
séparateur	Définit le texte à ignorer entre les enregistrements. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none">- NewlineSearch. Le séparateur est le symbole newline.- OffsetSearch. Le séparateur est défini par le nombre de caractères à partir de la fin de l'enregistrement précédent.- PatternSearch. Le séparateur est défini par une expression régulière.- TextSearch. Le séparateur est défini par une chaîne explicite ou une chaîne que vous récupérez dynamiquement depuis le document source.

ExcelToDataXml

Le processeur de document **ExcelToDataXml** convertit des documents Microsoft Excel en XML.

Le tableau suivant décrit les propriétés du processeur de document **ExcelToDataXml** :

Propriété	Description
activé	Détermine le contenu de la sortie. La propriété activé possède les options suivantes : <ul style="list-style-type: none">- Sélectionné. La sortie contient des données brutes et des données formatées.- Effacé. La sortie contient uniquement des données formatées. Cette option est sélectionnée par défaut.
param1	Détermine si les données brutes s'affichent dans la sortie du processeur de document lorsque les données brutes diffèrent des données formatées. param1 est nommé Display_raw_data_when_different et possède une seule propriété, activé .
param2	Détermine s'il faut ajouter ou non des éléments à la sortie lorsqu'une table source contient des cellules vides au milieu d'une ou de plusieurs lignes. Par exemple : Une table source comprend trois colonnes et deux lignes. Sur la première ligne, les trois colonnes sont remplies. Sur la deuxième ligne, les première et dernière colonnes sont remplies et la deuxième colonne est vide. Lorsque param2 est désactivé, le processeur crée deux éléments pour la deuxième ligne avec les valeurs des deux cellules de colonnes remplies. Lorsque param2 est activé, le processeur crée trois éléments pour la deuxième ligne : deux éléments avec les valeurs des cellules de colonnes remplies et un élément vide pour la cellule de colonne vide. Cette option est désactivée par défaut.
param3	Détermine s'il faut ajouter ou non des éléments à la sortie lorsqu'une table source contient des cellules vides à la fin d'une ou de plusieurs lignes. Par exemple : Une table source comprend trois colonnes et deux lignes. Sur la première ligne, les trois colonnes sont remplies. Sur la deuxième ligne, la première colonne est remplie et les deuxième et troisième colonnes sont vides. Lorsque param3 est désactivé, le processeur crée un élément pour la deuxième ligne, avec la valeur de la cellule de colonne remplie. Lorsque param3 est activé, le processeur crée trois éléments pour la deuxième ligne : un élément avec la valeur de la cellule de colonne remplie et deux éléments vides pour deux cellules de colonnes vides. Cette option est désactivée par défaut.

Le XML contient les données et les résultats de formules qui existaient dans le document Excel d'origine. Il ne conserve pas les formules elles-mêmes, les informations de formatage ou le code macro. Si vous devez utiliser un code macro, utilisez **ExcelToXml** plutôt que **ExcelToDataXml**.

La représentation XML est conforme à un sous-ensemble du schéma `ExcelToXml.xsd` que vous pouvez trouver dans le sous-dossier `doc` du répertoire d'installation.

La sortie du processeur est encodée en UTF-8. Si une transformation reçoit une entrée du processeur, vous devez définir le codage d'entrée à UTF-8.

Le processeur prend en charge Excel version 97 et supérieure. Il accède à son entrée directement et non via l'application Excel. Vous n'avez pas besoin d'installer Excel sur l'ordinateur. Le processeur prend en charge le format XLS et le format XLSX.

Ce composant est implémenté en Java et requiert une configuration correcte du Java Runtime Environment (JRE).

ExcelToXml

Le processeur de document **ExcelToXml** convertit des documents Microsoft Excel en XML.

Le tableau suivant décrit les propriétés du processeur de document **ExcelToXml** :

Propriété	Description
activé	Définit la valeur de param2 ou param3 .
param1	Définit les feuilles du classeur Excel à inclure dans le XML. Dans la sortie XML, chaque feuille est représentée par un élément <sheet>. param1 est appelé include_sheets et possède la propriété valeur .
param2	Détermine si le processeur de document inclut des cellules vides dans le fichier XML de sortie si les cellules sont formatées ou fusionnées. param2 est appelé include_empty_cells et possède la propriété activée qui a les options suivantes : <ul style="list-style-type: none">- Sélectionné. La sortie inclut des cellules vides.- Effacé. La sortie omet les cellules vides. La valeur par défaut est Sélectionné.
param3	Détermine si le processeur de document inclut du code macro Excel dans le XML de sortie. param3 est appelé include_macro_information et possède la propriété activée qui a les options suivantes : <ul style="list-style-type: none">- Sélectionné. Le processeur de document inclut un macro-code.- Effacé. Le processeur de document omet le macro-code. La valeur par défaut est Effacé.
param4	Détermine si le processeur de document inclut des cellules vides non formatées dans le XML de sortie pour les cellules. param4 est appelé include_empty_non_formatted_cells et possède la propriété activée qui a les options suivantes : <ul style="list-style-type: none">- Sélectionné. La sortie inclut des cellules vides.- Effacé. La sortie omet les cellules vides. La valeur par défaut est Effacé.
valeur	Définit une liste des options suivantes : <ul style="list-style-type: none">- La chaîne « Tout ». La sortie comprend toutes les feuilles.- zones de stockage des données contenant les noms des feuilles. La sortie inclut uniquement les feuilles nommées. Si vous listez une feuille qui n'existe pas dans le classeur, le processeur génère un élément <sheet> contenant un message d'avertissement. Les autres feuilles sont traitées normalement. La valeur par défaut est Tout.

Le XML conserve les données, les formules, le formatage et le code macro qui existaient dans le document Excel d'origine. Si seules les données sont requises, utilisez le processeur **ExcelToDataXml** qui offre une sortie plus petite et de meilleures performances.

La représentation XML est conforme au schéma `ExcelToXml.xsd` qui est dans le sous-dossier `doc` du répertoire d'installation.

La sortie du processeur est encodée en UTF-16LE. Si une transformation reçoit une entrée du processeur, vous devez définir le codage d'entrée à UTF-16LE.

Le processeur prend en charge Excel version 97-2003. Le processeur accède directement à l'entrée sans passer par Excel. Vous n'avez pas besoin d'installer Excel sur l'ordinateur.

Ce composant est implémenté en Java et requiert une configuration correcte du Java Runtime Environment (JRE).

ExcelToXml_03_07_10

Le processeur de document **ExcelToXml_03_07_10** convertit les fichiers suivants en XML :

- Fichiers XLSX créés avec Microsoft Excel 2007, 2010 ou 2013
- Fichiers XLS créés avec Microsoft Excel 2003, 2007, 2010 ou 2013

ExpandFrameSet

Le processeur de document **ExpandFrameSet** ouvre tous les cadres d'un document HTML. Utilisez ce processeur de document lorsque le document source d'un analyseur est un frameset HTML. L'analyseur s'exécute sur le contenu de tous les cadres.

ExternalJavaPreProcessor

Le processeur de document **ExternalJavaPreProcessor** exécute un processeur de document défini par l'utilisateur qui est implémenté en Java.

Le tableau suivant décrit les propriétés du processeur de document **ExternalJavaPreProcessor** :

Propriété	Description
jclass	Définit le chemin de la classe Java.
jmethod	Définit la méthode à exécuter.

Ce composant est implémenté en Java et requiert une configuration correcte du Java Runtime Environment (JRE).

Remarque: Ce composant est obsolète. L'éditeur IntelliScript l'affiche pour les scripts hérités. Ne l'utilisez pas dans de nouveaux scripts. Créez, à la place, un processeur de document Java personnalisé. Pour plus d'informations, consultez la section ["Développement d'un composant personnalisé" à la page 437](#).

HIPAAValidator

Le processeur de document **HIPAAValidator** valide les messages HIPAA et génère les accusés de réception HIPAA. Le projet **HIPAA_Validation** de la bibliothèque HIPAA utilise ce processeur.

Le tableau suivant décrit les propriétés du processeur de document **HIPAAValidator** :

Propriété	Description
param1	La propriété param1 est nommée validation_params et possède une seule propriété, valeur , qui a les options suivantes : <ul style="list-style-type: none">- LDNSB- Valideur
param2	Définit le type de validation. La propriété param2 est nommée types_to_validate et possède une seule propriété, valeur . Les valeurs valides sont comprises entre 1 et 7.
param3	Définit le format pour la sortie du rapport d'erreurs. La propriété param3 est nommée report_formats et possède une seule propriété, valeur , qui a les options suivantes : <ul style="list-style-type: none">- HTML. Utilisez pour afficher dans l'outil Developer.- XML. Utilisez pour un traitement ultérieur.

Propriété	Description
param4	Définit le type d'accusé de réception. La propriété param4 est nommée generate_acknowledgments et possède une seule propriété, valeur , qui a les options suivantes : <ul style="list-style-type: none"> - 277 - 824 - 997 - 999 - TA1
valeur	Définit la valeur de param1 , param2 , param3 ou param4 .

Remarque: Ce processeur de document fonctionne sur les plateformes Windows et Linux x64. Avant de pouvoir l'utiliser, vous devez installer et configurer le package complémentaire de validation HIPAA sur chaque ordinateur où vous exécutez **HIPAAValidator**.

PdfFormToXml_1_00

Le processeur de document **PdfFormToXml_1_00** convertit les formulaires PDF en XML. Le processeur prend en charge les formulaires conformes aux normes Adobe AcroForms.

PdfToTxt_3_02

Le processeur de document **PdfToTxt_3_02** convertit des fichiers PDF en texte.

Le tableau suivant décrit les propriétés du processeur de document **PdfToTxt_3_02** :

Propriété	Description
activé	Définit la valeur de param2 ou param4 .
param1	Définit une chaîne ou une variable qui contient le facteur d'espacement de mot. La propriété param1 est nommée WordSpacingFactor et n'a qu'une seule propriété, valeur , contenant la chaîne ou la variable. La valeur par défaut est 1.8.
param2	Détermine si le document de sortie est optimisé pour les tableaux. La propriété param2 est nommée OptimizeForTables et n'a qu'une seule propriété, activé , possédant les options suivantes : <ul style="list-style-type: none"> - Sélectionné. Le document de sortie est optimisé pour les tableaux. - Effacé. Le document de sortie n'est pas optimisé pour les tableaux. La valeur par défaut est Effacé.
param3	Définit une chaîne ou une variable qui contient le mot de passe. La propriété param3 est nommée Password et n'a qu'une seule propriété, valeur , contenant la chaîne ou la variable.
param4	La propriété param4 est nommée HideNewPageChar et n'a qu'une seule propriété, activé , possédant les options suivantes : <ul style="list-style-type: none"> - Sélectionné. Les caractères de nouvelle page sont masqués. - Effacé. Les caractères de nouvelle page ne sont pas masqués. La valeur par défaut est Effacé.
param5	Définit une chaîne ou une variable qui contient les optimisations avancées. La propriété param5 est nommée AdvancedOptimizations et n'a qu'une seule propriété, valeur , contenant la chaîne ou la variable.
valeur	Définit la valeur de param1 , param3 ou param5 .

Le préprocesseur PdfToTxt peut ne pas prendre en charge certains PDF avec des polices incorporées. Si le préprocesseur échoue, copiez le texte du PDF en entrée dans le Bloc-notes pour rechercher les polices incorporées. Si vous ne pouvez pas coller le texte ou si celui-ci est corrompu, le PDF contient probablement des polices incorporées.

Remarque: Ce composant est obsolète. L'éditeur IntelliScript l'affiche pour les projets hérités. Ne l'utilisez pas dans de nouveaux scripts.

PdfToTxt_4

Le processeur de document **PdfToTxt_4** convertit des fichiers PDF en texte ou XML.

Le tableau suivant décrit les propriétés du processeur de document **PdfToTxt_4** :

Propriété	Description
param1	Définit la mise en page de table PDF. La propriété param1 a une seule option : PdfLayout
valeur	Définit la mise en page de table PDF. Double-cliquez sur la propriété valeur pour ouvrir l'éditeur de configuration de table.

L'éditeur de configuration de table personnalise la manière dont les tables sont lues. Utilisez-le pour corriger les problèmes avec l'alignement des colonnes, l'encapsulation des mots, l'espace interligne et le débordement d'une cellule à une autre. Pour plus d'informations, voir ["Éditeur de configuration de tableau PdfToTxt_4" à la page 174](#)

Le processeur de document **PdfToTxt_4** génère une sortie texte par défaut. Utilisez l'éditeur de configuration de table pour sélectionner la sortie XML. Le XML correspond au schéma PDF4.xsd, que vous pouvez trouver dans le répertoire suivant :

```
<INSTALL_DIR>\DataTransformation\doc
```

Lorsque vous utilisez le processeur de document **PdfToTxt_4**, définissez le codage d'entrée sur UTF-8 pour permettre à l'analyseur, au mappeur ou au sérialiseur de lire le document correctement.

Remarque: Le préprocesseur PdfToTxt peut ne pas prendre en charge certains PDF avec des polices incorporées. Si le préprocesseur échoue, copiez le texte du PDF en entrée dans le Bloc-notes pour rechercher les polices incorporées. Si vous ne pouvez pas coller le texte ou si celui-ci est corrompu, le PDF contient probablement des polices incorporées.

PowerpointToTextML

Le processeur de document **PowerpointToTextML** convertit des présentations Microsoft PowerPoint (PPT) en schéma XML TextML. Pour plus d'informations, consultez la section ["Schéma XML TextML" à la page 173](#).

Ce composant prend en charge la version 97 de PowerPoint et les versions ultérieures. Il accède à son entrée directement et non via PowerPoint. Vous n'avez pas besoin d'installer Microsoft PowerPoint sur l'ordinateur.

Ce composant est implémenté en Java et requiert une configuration correcte du Java Runtime Environment (JRE).

ProcessByTransformers

Le processeur de document **ProcessByTransformers** exécute un transformateur ou une séquence de transformateurs sur la totalité du document. Une transformation peut ensuite être exécutée sur la sortie des transformateurs.

Définissez la liste des transformateurs dans la ligne **transformateurs**.

ProcessorPipeline

Le processeur de document **ProcessorPipeline** définit une séquence de processeurs de document à exécuter sur un document. Utilisez ce composant lorsque vous devez exécuter deux ou plusieurs processeurs de documents.

Définissez la liste des processeurs de document dans la ligne **pre_processor_list**.

RtfToTextML

Le processeur de document **RtfToTextML** convertit les fichiers RTF en schéma XML TextML. Pour plus d'informations, consultez la section ["Schéma XML TextML" à la page 173](#).

La sortie du processeur est encodée en UTF-16LE. Si une transformation reçoit une entrée du processeur, vous devez définir le codage d'entrée à UTF-16LE.

WordToXml

Le processeur de document **WordToXml** convertit les documents Microsoft Word en XML.

La sortie du processeur est encodée en UTF-16LE. Si une transformation reçoit une entrée du processeur, vous devez définir le codage d'entrée à UTF-16LE.

Ce composant prend en charge la version Word 97 et suivantes. Il accède à son entrée directement, et non pas via Microsoft Word. Vous n'avez pas besoin d'installer Word sur l'ordinateur.

Ce composant est implémenté en Java et requiert une configuration correcte du Java Runtime Environment (JRE).

XmlToDocument_372

Le processeur de document **XmlToDocument_372** convertit les données XML en formats de documents, tels que PDF ou Excel. Vous pouvez l'utiliser en tant que postprocesseur pour convertir la sortie de l'analyseur ou du mappeur en différents types de document.

Ce composant utilise le complément Eclipse **Business Intelligence and Reporting Tool** (BIRT) pour générer les documents de sortie. Dans BIRT, vous devez configurer un rapport qui convertit le XML dans le format de document désiré. Le processeur **XmlToDocument_372** exécute le rapport.

Vous pouvez télécharger BIRT à partir de l'emplacement mentionné dans le fichier `readme_BIRT.txt` à `<Data Transformation engine installation directory>/readme_Birt.txt`.

Pour plus d'informations sur BIRT, voir <http://www.eclipse.org/birt>.

Remarque: Pour utiliser BIRT version 4.5, utilisez le préprocesseur **XmlToDocument_45** au lieu du préprocesseur **XmlToDocument_372**.

Le tableau suivant décrit les propriétés du processeur de document **XmlToDocument_372** :

Propriété	Description
param1	Chemin et nom du fichier BIRT *.rptdesign. La propriété param1 est nommée report_file et contient la propriété value qui contient le chemin et le nom de fichier.
param2	Format du document de sortie. La propriété param2 est nommée output_format et contient la propriété value qui a les options suivantes : <ul style="list-style-type: none">- pdf. Document PDF.- doc. Document Microsoft Word.- xls. Classeur Microsoft Excel.- ppt. Présentation Microsoft PowerPoint.- html. Page Web HTML.- ps. Document PostScript. La valeur par défaut est pdf.
param3	Variable qui contient l'emplacement du fichier *.rptdesign. La propriété param3 est nommée report_location et contient la propriété value qui pointe vers la variable. La valeur par défaut est \$VarServiceInfo/*s/ServiceLocation.
valeur	Contient la valeur de param1 , param2 ou param3 .

Remarque: En vigueur dans la version 9.5.1, le processeur **XmlToDocument** est obsolète. L'éditeur IntelliScript affiche toujours le préprocesseur **XmlToDocument** dans les scripts existants, mais vous ne pouvez plus ajouter ce préprocesseur aux nouveaux scripts. Utilisez le préprocesseur **XmlToDocument_372** à la place.

XmlToDocument_45

Le processeur de document **XmlToDocument_45** convertit les données XML en formats de documents tels que PDF ou Excel. Vous pouvez l'utiliser en tant que postprocesseur pour convertir la sortie de l'analyseur ou du mappeur en différents types de document.

Ce composant utilise le complément Eclipse **Business Intelligence and Reporting Tool** (BIRT) version 4.5 pour générer les documents de sortie. Dans BIRT, vous devez configurer un rapport qui convertit le fichier XML en un format de document désiré. Le processeur **XmlToDocument_45** exécute le rapport.

Vous pouvez télécharger BIRT à partir de l'emplacement mentionné dans le fichier `readme_BIRT.txt` à `<Data Transformation engine installation directory>/readme_Birt.txt`.

Pour plus d'informations sur BIRT, voir <http://www.eclipse.org/birt>.

Le tableau suivant décrit les propriétés du processeur de document **XmlToDocument_45** :

Propriété	Description
param1	Chemin et nom du fichier BIRT *.rptdesign. La propriété param1 est nommée report_file et contient la propriété value qui contient le chemin et le nom de fichier.
param2	Format du document de sortie. La propriété param2 est nommée output_format et contient la propriété value qui a les options suivantes : <ul style="list-style-type: none">- pdf. Document PDF.- doc. Document Microsoft Word.- xls. Classeur Microsoft Excel.- ppt. Présentation Microsoft PowerPoint.- html. Page Web HTML.- ps. Document PostScript. La valeur par défaut est pdf.
param3	Variable qui contient l'emplacement du fichier *.rptdesign. La propriété param3 est nommée report_location et contient la propriété value qui pointe vers la variable. La valeur par défaut est \$VarServiceInfo/*s/ServiceLocation.
valeur	Contient la valeur de param1 , param2 ou param3 .

XmlToExcel

Le processeur de document **XmlToExcel** convertit des documents XML en format Microsoft Excel.

Le processeur opère sur une représentation XML d'un classeur Excel. La représentation XML doit être codée en UTF-16LE et doit être conforme au schéma `ExcelToXml.xsd`. Vous pouvez trouver le schéma dans le sous-répertoire `doc` du répertoire d'installation. Le fichier schéma est fourni à titre informatif. Vous pouvez utiliser le processeur sans ajouter de schéma à votre projet.

Le processeur inverse l'opération **ExcelToXml**. Par exemple, vous pouvez utiliser **ExcelToXml** pour convertir un classeur Excel en XML. Vous pouvez ensuite modifier certaines données XML et utiliser **XmlToExcel** pour reconvertir les données en classeur Excel.

Ce composant prend en charge la version 97 d'Excel et les suivantes. Il enregistre sa sortie directement, non via Microsoft Excel. Vous n'avez pas besoin d'installer Excel sur l'ordinateur.

Ce composant est implémenté en Java et requiert une configuration correcte du Java Runtime Environment (JRE).

XmlToXlsx

Le processeur de document **XmlToXlsx** convertit des documents XML au format .xlsx de Microsoft Excel. Le processeur de document **XmlToXlsx** peut éventuellement utiliser un modèle .xlsx pour générer le document .xlsx.

Le processeur opère sur une représentation XML d'un classeur Excel. La représentation XML doit être codée en UTF-8 et doit être conforme au schéma `ExcelToXml_03_07_10.xsd`. Vous pouvez trouver le schéma dans le sous-répertoire `doc` du répertoire d'installation. Le fichier de schéma est fourni à titre informatif.

Le processeur inverse l'opération **ExcelToXml_03_07_10**. Utilisez le processeur **ExcelToXml_03_07_10** dans une transformation Processeur de données afin de transformer un classeur Excel workbook au format XML. Après avoir traité les données XML, utilisez le processeur **XmlToXlsx** afin de retransformer les données au format Excel.

Ce composant prend en charge la version 2007 d'Excel et les suivantes. Il enregistre sa sortie directement, non via Microsoft Excel. Vous n'avez pas besoin d'installer Excel sur l'ordinateur.

Ce composant est implémenté en Java et requiert une configuration correcte du Java Runtime Environment (JRE).

Le tableau suivant décrit les propriétés du processeur de document **XmlToXlsx** :

Propriété	Description
param1	Variable contenant le nom du fichier de modèle * .xlsx. La propriété param1 a la propriété template_file qui spécifie le nom du fichier de modèle.
param2	Variable contenant l'emplacement du fichier de modèle * .xlsx. La propriété param2 a la propriété template_location qui spécifie le chemin du fichier de modèle.

Remarque: Les feuilles Excel qui existent uniquement dans le modèle sont représentées exactement comme dans le modèle dans la sortie .xlsx. Les feuilles Excel qui sont définies dans le modèle et dans le document XML reçoivent les styles de cellule du modèle et les valeurs de cellule du document XML.

Pour appliquer un style de cellule provenant d'une autre cellule du modèle, vous pouvez utiliser l'attribut `style_as` dans le document XML dans le cadre de l'élément `cell`.

Exemple

Pour utiliser l'attribut `style_as` afin d'appliquer un style de cellule provenant d'une cellule de la feuille active, définissez l'attribut `style_as` de sorte qu'il corresponde au numéro de cellule contenant ce style. Dans l'exemple suivant, le style provenant du champ A1 de la feuille active s'applique à la cellule définie par le contexte XML, à savoir ligne 3 et cellule 2.

```
<row rownumber="3" firstcol="1" lastcol="12" height="405" zeroheight="false" >
  <cell number="2" type="string" style_index="3" font_index="3" style_as="A1">
    <data>Informatica</data>
  </cell>
</row>
```

Pour appliquer un style de cellule provenant d'une cellule d'une autre feuille, définissez l'attribut `style_as` de sorte qu'il corresponde à la feuille et au numéro de cellule contenant ce style. Dans l'exemple suivant, le style appliqué provient du champ A1 de la feuille de classeur nommée Résumé.

```
<row rownumber="3" firstcol="1" lastcol="12" height="405" zeroheight="false" >
  <cell number="2" type="string" style_index="3" font_index="3"
  style_as="Summary:A1">
    <data>Informatica</data>
  </cell>
</row>
```

Schéma XML TextML

Certains processeurs de document convertissent les documents en vocabulaire XML appelé TextML. C'est un vocabulaire XML simple permettant d'enregistrer le contenu d'un document sans mise en page.

Le schéma TextML, `textML.xsd`, est disponible dans le sous-dossier `\doc` du dossier d'installation.

Voici un exemple de document TextML.

```
<?xml version="1.0" encoding="UTF-16LE"?>
<document>
```

```

<docinfo>
  <title>TextML Sample</title>
  <author>Tex Tomiller</author>
  <company>Acme Gizmos, Inc.</company>
  <modified>2004-03-14T14:39:00</modified>
  <created>2004-03-12T09:15:00</created>
  <last_author>Tex Tomiller</last_author>
  <word_count>16</word_count>
  <char_count>105</char_count>
  <version>2</version>
</docinfo>
<docbody>
  <p>This is a sample of the TextML XML vocabulary.</p>
  <p>TextML saves document content without layout information.<p>
</docbody>
</document>

```

Éditeur de configuration de tableau PdfToTxt_4

L'éditeur de configuration de tableau personnalise la manière dont le processeur de document **PdfToTxt_4** convertit des tableaux en documents PDF.

Utilisez l'éditeur de configuration de tableau quand les paramètres par défaut du processeur de document **PdfToTxt_4** ne rendent pas correctement l'alignement des colonnes, l'enveloppement des mots, l'espacement des lignes ou le débordement d'une cellule sur une autre.

Remarque: L'interface utilisateur de l'éditeur de configuration de tableau ne s'affiche qu'en anglais.

1. Ajoutez un analyseur, un mappeur, un sérialiseur ou un port **AdditionalInputPort** au script.
2. Dans la propriété **example_source**, définissez la propriété **pre_processor** sur PdfToTxt_4.
3. Dans la propriété **pre_processor**, double-cliquez sur la propriété **valeur**.

L'éditeur de configuration de tableau s'affiche. Le panneau supérieur affiche le document PDF d'entrée et le panneau inférieur affiche la sortie **PdfToTxt_4**.

Les commandes d'édition de tableau s'affichent dans la barre d'outils en haut de la fenêtre. Vous pouvez faire un clic droit pour afficher un menu d'édition.

4. Accédez à un tableau du document PDF et cliquez sur **Ajouter le tableau**.

Le nom du tableau s'affiche dans le champ **Tableaux** et dans le champ **Nom**.

5. Sélectionnez **Utiliser des expressions régulières**. Dans le champ **Début du tableau**, entrez une expression régulière qui définit l'angle supérieur gauche du tableau.

Astuce: Utilisez les titres des deux premières colonnes en tant qu'expression régulière. Ajouter plus de titres de colonnes que nécessaire pour rendre **Début du tableau** unique. Séparez les titres par un seul caractère d'espacement, même si les colonnes sont largement séparées.

6. Dans le champ **Fin du tableau**, entrez une expression régulière qui définit le texte immédiatement après le tableau.

Remarque: La valeur de **Fin du tableau** doit s'afficher dans le corps du document et non dans un pied de page.

7. Cliquez sur **Exécuter**.

L'éditeur affiche la configuration du tableau détectée par **PdfToTxt_4**. Le haut et le bas du tableau apparaissent sous la forme de lignes bleues horizontales. Les bordures de colonne par défaut apparaissent sous la forme de lignes rouges verticales.

8. Pour modifier les bordures de colonne, procédez comme suit :
 - Faites glisser une bordure de colonne vers la droite ou vers la gauche pour modifier sa position.
 - Cliquez sur **Ajouter la colonne** pour ajouter une colonne.
 - Cliquez sur **Supprimer la colonne** et sélectionnez une bordure de colonne pour supprimer une colonne.

Remarque: Si le tableau contient des cellules fusionnées horizontalement, **PdfToTxt_4** peut tronquer les entrées.

9. Examinez la fenêtre de sortie pour confirmer que le tableau est converti. Si ce n'est pas le cas, corrigez les définitions du tableau.
10. Répétez les étapes 1 à 9 pour chaque tableau dans le document PDF.
11. Cliquez sur **OK** pour revenir à l'outil Developer.

Une chaîne XML qui définit la configuration du tableau s'affiche dans la propriété **valeur** du processeur de document **PdfToTxt_4**.

Éditeur Options

Le tableau suivant décrit les contrôles et les champs dans l'éditeur de configuration du tableau **PdfToTxt_4**.

Contrôle ou Champ	Description
Zoom avant	Agrandir l'affichage du fichier PDF.
Zoom arrière	Diminuer l'affichage du fichier PDF.
Ajuster largeur	Affichez le document PDF en fonction de la largeur de la fenêtre.
Page préc.	Revenez à la page précédente.
Page suivante	Passez à la page suivante.
Rechercher	Rechercher une chaîne dans le fichier PDF.
Ajouter une table	Ajoutez une table à la configuration.
Sup. Table	Supprimez une table de la configuration.
Ajouter une colonne	Ajoutez une bordure de colonne pour la table en cours.
Sup. Colonne	Supprimez la bordure de la colonne actuellement sélectionnée.
Processus	Appliquez les définitions de la table en cours. Cliquez sur Exécuter après chaque action associé à une colonne et une table pour appliquer cette action.
Tables	Une liste de tables définie dans le PDF d'entrée. Vous pouvez sélectionner une table en cliquant dessus.
Nom	Nom de la table actuellement sélectionnée.
Début du tableau	Une expression définissant le coin supérieur gauche du tableau.
Fin de tableau	Une expression définissant le premier texte après le tableau.

Contrôle ou Champ	Description
En-tête de la page	Une expression définissant la fin de l'en-tête de la page. Utilisez cette option pour exclure l'en-tête du traitement du tableau.
Pied de page	Une expression définissant la fin du pied de page. Utilisez cette option pour exclure le pied de page du traitement du tableau.
Utiliser les expressions régulières	Si sélectionné, le processeur interprète les Début du tableau , Fin du tableau , En-tête de la page et Pied de page comme des expressions régulières et recherche le texte correspondant. Si désélectionné, le processeur interprète ces champs comme du texte littéral.
Recalculer pendant l'exécution.	Si vous sélectionnez cette option, PdfToTxt_4 ignore les configurations de la table que vous avez spécifiées à l'aide de l'éditeur de configuration de la table. Cette fonction est utile si les tableaux dans un fichier PDF sont suffisamment simples pour que PdfToTxt_4 puisse le traiter sans configuration spéciale. Par exemple, imaginons qu'un état financier dans un fichier PDF simple contienne un tableau dont les colonnes peuvent varier légèrement de mois en mois. Sélectionnez l'option Recalculer pendant l'exécution pour que PdfToTxt_4 ajuste les largeurs de colonnes lors de l'exécution.
Recalculer maintenant	Si vous avez modifié la définition du tableau, par exemple en modifiant les bordures de la colonne ou en ajoutant un En-tête de page ou un Pied de page , cliquez sur Recalculer maintenant pour mettre à jour la définition du tableau.
Page	Numéro de la page PDF actuellement visible.
Sortie en XML	Génère la sortie PdfToTxt_4 en XML au lieu de texte.
Délimiteur	Entrez un caractère à utiliser en tant que séparateur de colonne dans la sortie du texte. La valeur par défaut est une barre verticale ().
OK	Cliquez pour enregistrer la configuration de la table et revenir à l'outil Developer.
Annuler	Cliquez pour revenir à l'outil Developer sans enregistrer la configuration de la table.
Aide à la navigation du tableau	L'aide à la navigation du tableau affiche le nombre de fois qu'un tableau est trouvé dans le document PDF. Un exemple d'aide à la navigation est <code>Table « Table 1 » trouvé 2 fois</code> . Les flèches à côté de cette information vous permettent de sauter en avant ou en arrière entre les instances de la même structure de table.

Exemple de conversion PDF

Cet exemple illustre la procédure de configuration de la table **PdfToTxt_4** à l'aide d'un exemple de projet d'analyseur et d'un exemple de document PDF.

L'entrée du processeur est un petit rapport financier au format PDF. Le rapport contient du texte et deux tables. Utilisez l'éditeur de configuration de table pour vous assurer que le processeur convertisse correctement les tables en texte.

Configuration de la première table

1. Configurez un analyseur et assignez le document PDF en tant que **example_source**. Double-cliquez sur la propriété **valeur** pour ouvrir l'éditeur de configuration de tableau.
2. Dans l'affichage du fichier PDF, accédez au premier tableau.

3. Définissez Début du tableau = GID RMS ID, les titres des deux premières colonnes du tableau. Notez que l'expression est sensible à la casse.
4. Définissez Fin du tableau = Transfère les transactions des changes , le premier texte suivant le tableau. L'éditeur affiche la configuration de tableau.
5. Si nécessaire, ajustez la définition du tableau et les colonnes. Vous pouvez faire glisser, ajouter ou supprimer des bordures de colonnes.

Configuration de la seconde table

La seconde table s'étend sur plusieurs pages.

1. Cliquez sur **Ajouter la table**.
Le système affiche Table 2 dans les champs **Tables** et **Nom**.
2. Définissez Table Start = Ticker Shares Traded.
3. Définissez Table End = Conclusion, le premier texte de corps après la table.
4. Cliquez sur **Exécuter** pour configurer la table.
5. Ajustez les bordures de droite des colonnes **Actions** et **Devise** .
6. Suivez les étapes suivantes pour éliminer l'en-tête et le pied de page du document de sortie :
 - a. Définissez Page Header = Gain/Perte.
 - b. Définissez Page Footer = Page [1-9].
 - c. Cliquez sur **Exécuter**.

CHAPITRE 14

Formats

Ce chapitre comprend les rubriques suivantes :

- [Présentation des formats, 178](#)
- [Propriétés standard du format, 179](#)
- [Documentation de référence du composant Format, 179](#)
- [Documentation de référence du composant Délimiteurs, 185](#)
- [Documentation de référence du composant préprocesseur de format, 190](#)

Présentation des formats

La propriété `format` d'un analyseur définit le format des documents que la transformation doit traiter. La valeur de la propriété est l'un des composants de format suivants :

```
BinaryFormat  
CustomFormat  
HtmlFormat  
RtfFormat  
TextFormat  
XmlFormat
```

Le format possède ses propres propriétés, qui définissent de manière plus précise la manière dont l'analyseur interprète et traite l'entrée.

Le tableau suivant décrit les sous-composants que vous pouvez imbriquer dans un format :

Sous-composant	Description
Délimiteur	Définit une hiérarchie de caractères ou de chaînes qui organisent les informations dans le document, comme les retours à la ligne et les tabulations.
Préprocesseur de format	Nettoie la source avant que l'analyseur ne commence la recherche d'ancres.
Transformateur par défaut	Effectue les opérations prédéfinies dans la sortie de chaque ancre.

Propriétés standard du format

Le tableau suivant décrit les propriétés standard des composants de format :

Propriété	Description
default_Transformers	Définit une liste de transformateurs que l'analyseur applique à la sortie de chaque ancre de contenu.
délimiteurs	Définit la structure des informations dans le document. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none">- CommaDelimited. Les champs de données sont séparés par des virgules.- DelimiterHierarchy. Les champs de données sont séparés par ou entourés de caractères texte.- HL7. Les champs de données sont séparés comme défini dans la norme HL7.- Positionnel. Les champs de données sont définis par le nombre de caractères entre eux.- PostScript. Les champs de données sont définis selon le format PostScript.- RTF. Les champs de données sont définis selon le format RTF.- SGML. Les champs de données sont définis selon le format SGML.- SpaceDelimited. Les champs de données sont séparés par des espaces.- TabDelimited. Les champs de données sont séparés par des tabulations. Pour plus d'informations, consultez la section "Documentation de référence du composant Délimiteurs" à la page 185 .
nom	Libellé descriptif pour le composant. Ce libellé s'affiche dans le fichier journal et la vue Événements . Utilisez la propriété nom pour identifier le composant ayant provoqué l'événement.
pre_processor	Définit un préprocesseur de format qui traite l'entrée après tout processeur de document que vous avez défini pour la propriété pre_processor de exemple_source . Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none">- HtmlProcessor. Convertit toutes les combinaisons de tabulations, d'espaces ou de retours à la ligne en un seul caractère d'espacement. Il n'est pas restreint aux seuls documents HTML.- RtfProcessor. Normalise les fichiers RTF. Par défaut, la valeur est vide.
remarque	Commentaire défini par l'utilisateur, qui décrit l'objectif ou l'action du composant.

Documentation de référence du composant Format

Les composants Format définissent le format des documents d'entrée. Définissez les composants de format dans la propriété **format** d'un **Analyseur**.

BinaryFormat

Le format **BinaryFormat** traite les fichiers binaires et les fichiers textes que vous voulez considérer comme un tampon d'octets binaires.

Le tableau suivant décrit les propriétés du format **BinaryFormat** :

Propriété	Description
default_transformers	Définit une liste de transformateurs que l'analyseur applique à la sortie de chaque ancre de contenu. La valeur par défaut est Vide.
délimiteurs	Définit la structure des informations dans le document. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none">- CommaDelimited. Les champs de données sont séparés par des virgules.- DelimiterHierarchy. Les champs de données sont séparés par ou entourés de caractères texte.- HL7. Les champs de données sont séparés comme défini dans la norme HL7.- Positionnel. Les champs de données sont définis par le nombre de caractères entre eux.- PostScript. Les champs de données sont définis selon le format PostScript.- RTF. Les champs de données sont définis selon le format RTF.- SGML. Les champs de données sont définis selon le format SGML.- SpaceDelimited. Les champs de données sont séparés par des espaces.- TabDelimited. Les champs de données sont séparés par des tabulations. Pour plus d'informations, consultez la section "Documentation de référence du composant Délimiteurs" à la page 185 . La valeur par défaut est Positionnel.
nom	Libellé descriptif pour le composant. Ce libellé s'affiche dans le fichier journal et la vue Événements . Utilisez la propriété nom pour identifier le composant ayant provoqué l'événement.
pre_processor	Définit un préprocesseur de format qui traite l'entrée après tout processeur de document que vous avez défini pour la propriété pre_processor de example_source . Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none">- HtmlProcessor. Convertit toutes les combinaisons de tabulations, d'espaces ou de retours à la ligne en un seul caractère d'espacement. Il n'est pas restreint aux seuls documents HTML.- RtfProcessor. Normalise les fichiers RTF. La valeur par défaut est Vide.
remarque	Commentaire défini par l'utilisateur, qui décrit l'objectif ou l'action du composant.

CustomFormat

Le format **CustomFormat** est un format défini par l'utilisateur pour le traitement de tout type de document source.

Le tableau suivant décrit les propriétés du format **CustomFormat** :

Propriété	Description
default_transformers	Définit une liste de transformateurs que l'analyseur applique à la sortie de chaque ancre de contenu. La valeur par défaut est Vide.
délimiteurs	Définit la structure des informations dans le document. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none">- CommaDelimited. Les champs de données sont séparés par des virgules.- DelimiterHierarchy. Les champs de données sont séparés par ou entourés de caractères texte.- HL7. Les champs de données sont séparés comme défini dans la norme HL7.- Positionnel. Les champs de données sont définis par le nombre de caractères entre eux.- PostScript. Les champs de données sont définis selon le format PostScript.- RTF. Les champs de données sont définis selon le format RTF.- SGML. Les champs de données sont définis selon le format SGML.- SpaceDelimited. Les champs de données sont séparés par des espaces.- TabDelimited. Les champs de données sont séparés par des tabulations. Pour plus d'informations, consultez la section " Documentation de référence du composant Délimiteurs " à la page 185. La valeur par défaut est DelimiterHierarchy.
nom	Libellé descriptif pour le composant. Ce libellé s'affiche dans le fichier journal et la vue Événements . Utilisez la propriété nom pour identifier le composant ayant provoqué l'événement.
pre_processor	Définit un préprocesseur de format qui traite l'entrée après tout processeur de document que vous avez défini pour la propriété pre_processor de exemple_source . Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none">- HtmlProcessor. Convertit toutes les combinaisons de tabulations, d'espaces ou de retours à la ligne en un seul caractère d'espacement. Il n'est pas restreint aux seuls documents HTML.- RtfProcessor. Normalise les fichiers RTF. La valeur par défaut est Vide.
remarque	Commentaire défini par l'utilisateur, qui décrit l'objectif ou l'action du composant.

Exemple

Un document source a la structure suivante :

```
Ron      Lehrer  && 547329876:27
Evelyn   Kern    && 9875424: 53
```

Chaque ligne du document est un enregistrement contenant le nom d'une personne, son numéro d'identification et son âge. Les champs sont séparés par les symboles && et :. Les champs contiennent des espaces multiples à des emplacements aléatoires.

Une façon d'analyser ce document est d'utiliser **CustomFormat**. Dans la propriété **délimiteurs** du format, assignez **DelimiterHierarchy** contenant les symboles :

```
newline
&&
:
```

Dans la propriété **default_transformers**, assignez **HtmlProcessor** qui supprime les espaces en trop de la sortie.

HtmlFormat

Le format **HtmlFormat** définit le format des fichiers HTML.

Le tableau suivant décrit les propriétés du format **HtmlFormat** :

Propriété	Description
default_transformers	Définit une liste de transformateurs que l'analyseur applique à la sortie de chaque ancre de contenu. La valeur par défaut est la liste suivante de transformateurs : <ul style="list-style-type: none">- RemoveTags. Enlève les balises HTML.- HtmlEntitiesToASCII. Convertit des entités HTML en leurs équivalents ASCII.- HtmlProcessor. Convertit toutes les combinaisons de tabulations, d'espaces ou de retours à la ligne en un seul caractère d'espacement.- RemoveMarginSpace. Supprime les espaces de début et de fin.
délimiteurs	Définit la structure des informations dans le document. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none">- CommaDelimited. Les champs de données sont séparés par des virgules.- DelimiterHierarchy. Les champs de données sont séparés par ou entourés de caractères texte.- HL7. Les champs de données sont séparés comme défini dans la norme HL7.- Positionnel. Les champs de données sont définis par le nombre de caractères entre eux.- PostScript. Les champs de données sont définis selon le format PostScript.- RTF. Les champs de données sont définis selon le format RTF.- SGML. Les champs de données sont définis selon le format SGML.- SpaceDelimited. Les champs de données sont séparés par des espaces.- TabDelimited. Les champs de données sont séparés par des tabulations. Pour plus d'informations, consultez la section "Documentation de référence du composant Délimiteurs" à la page 185 . La valeur par défaut est SGML.
nom	Libellé descriptif pour le composant. Ce libellé s'affiche dans le fichier journal et la vue Événements . Utilisez la propriété nom pour identifier le composant ayant provoqué l'événement.
pre_processor	Définit un préprocesseur de format qui traite l'entrée après tout processeur de document que vous avez défini pour la propriété pre_processor de example_source . Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none">- HtmlProcessor. Convertit toutes les combinaisons de tabulations, d'espaces ou de retours à la ligne en un seul caractère d'espacement. Il n'est pas restreint aux seuls documents HTML.- RtfProcessor. Normalise les fichiers RTF. La valeur par défaut est HtmlProcessor.
remarque	Commentaire défini par l'utilisateur, qui décrit l'objectif ou l'action du composant.

RtfFormat

Le format **RtfFormat** définit le format des fichiers RTF.

Le tableau suivant décrit les propriétés du format **RtfFormat** :

Propriété	Description
default_Transformers	Définit une liste de transformateurs que l'analyseur applique à la sortie de chaque ancre de contenu. La valeur par défaut est la liste suivante de transformateurs : <ul style="list-style-type: none">- RtfToASCII. Supprime les mots de contrôle RTF de la sortie- RemoveRtfFormatting. Supprime les instructions de formatage RTF du texte.- HtmlProcessor. Convertit toutes les combinaisons de tabulations, d'espaces ou de retours à la ligne en un seul caractère d'espacement.- RemoveMarginSpace. Supprime les espaces de début et de fin.
délimiteurs	Définit la structure des informations dans le document. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none">- CommaDelimited. Les champs de données sont séparés par des virgules.- DelimiterHierarchy. Les champs de données sont séparés par ou entourés de caractères texte.- HL7. Les champs de données sont séparés comme défini dans la norme HL7.- Positionnel. Les champs de données sont définis par le nombre de caractères entre eux.- PostScript. Les champs de données sont définis selon le format PostScript.- RTF. Les champs de données sont définis selon le format RTF.- SGML. Les champs de données sont définis selon le format SGML.- SpaceDelimited. Les champs de données sont séparés par des espaces.- TabDelimited. Les champs de données sont séparés par des tabulations. Pour plus d'informations, consultez la section "Documentation de référence du composant Délimiteurs" à la page 185 . La valeur par défaut est RTF.
nom	Libellé descriptif pour le composant. Ce libellé s'affiche dans le fichier journal et la vue Événements . Utilisez la propriété nom pour identifier le composant ayant provoqué l'événement.
pre_processor	Définit un préprocesseur de format qui traite l'entrée après tout processeur de document que vous avez défini pour la propriété pre_processor de exemple_source . Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none">- HtmlProcessor. Convertit toutes les combinaisons de tabulations, d'espaces ou de retours à la ligne en un seul caractère d'espacement. Il n'est pas restreint aux seuls documents HTML.- RtfProcessor. Normalise les fichiers RTF. La valeur par défaut est RtfProcessor.
remarque	Commentaire défini par l'utilisateur, qui décrit l'objectif ou l'action du composant.

TextFormat

Le format **TextFormat** définit le format des fichiers texte.

Utilisez ce format en combinaison avec un processeur de document pour traiter d'autres types de documents. Par exemple, vous pouvez l'utiliser avec le processeur de document **PdfToTxt_4** pour traiter les documents PDF.

Le tableau suivant décrit les propriétés du format **TextFormat** :

Propriété	Description
default_transformers	Définit une liste de transformateurs que l'analyseur applique à la sortie de chaque ancre de contenu. La valeur par défaut est la liste suivante de transformateurs : <ul style="list-style-type: none">- HtmlProcessor. Convertit toutes les combinaisons de tabulations, d'espaces ou de retours à la ligne en un seul caractère d'espacement.- RemoveMarginSpace. Supprime les espaces de début et de fin.
délimiteurs	Définit la structure des informations dans le document. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none">- CommaDelimited. Les champs de données sont séparés par des virgules.- DelimiterHierarchy. Les champs de données sont séparés par ou entourés de caractères texte.- HL7. Les champs de données sont séparés comme défini dans la norme HL7.- Positionnel. Les champs de données sont définis par le nombre de caractères entre eux.- PostScript. Les champs de données sont définis selon le format PostScript.- RTF. Les champs de données sont définis selon le format RTF.- SGML. Les champs de données sont définis selon le format SGML.- SpaceDelimited. Les champs de données sont séparés par des espaces.- TabDelimited. Les champs de données sont séparés par des tabulations. Pour plus d'informations, consultez la section " Documentation de référence du composant Délimiteurs " à la page 185. La valeur par défaut est DelimiterHierarchy.
nom	Libellé descriptif pour le composant. Ce libellé s'affiche dans le fichier journal et la vue Événements . Utilisez la propriété nom pour identifier le composant ayant provoqué l'événement.
pre_processor	Définit un préprocesseur de format qui traite l'entrée après tout processeur de document que vous avez défini pour la propriété pre_processor de exemple_source . Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none">- HtmlProcessor. Convertit toutes les combinaisons de tabulations, d'espaces ou de retours à la ligne en un seul caractère d'espacement. Il n'est pas restreint aux seuls documents HTML.- RtfProcessor. Normalise les fichiers RTF. La valeur par défaut est Vide.
remarque	Commentaire défini par l'utilisateur, qui décrit l'objectif ou l'action du composant.

XmlFormat

Le format **XmlFormat** définit le format des fichiers XML.

L'analyseur traite le document d'entrée XML comme un texte ordinaire. Vous pouvez définir des séparateurs, des ancres et d'autres composants tout comme vous le faites pour un document texte basique.

Le tableau suivant décrit les propriétés du format **XmlFormat** :

Propriété	Description
default_transformers	Définit une liste de transformateurs que l'analyseur applique à la sortie de chaque ancre de contenu. La valeur par défaut est la liste suivante de transformateurs : <ul style="list-style-type: none">- RemoveTags. Retire les balises XML de la sortie.- HtmlEntitiesToASCII. Convertit les entités XML en leurs équivalents ASCII.- HtmlProcessor. Convertit toutes les combinaisons de tabulations, d'espaces ou de retours à la ligne en un seul caractère d'espacement.- RemoveMarginSpace. Supprime les espaces de début et de fin.
délimiteurs	Définit la structure des informations dans le document. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none">- CommaDelimited. Les champs de données sont séparés par des virgules.- DelimiterHierarchy. Les champs de données sont séparés par ou entourés de caractères texte.- HL7. Les champs de données sont séparés comme défini dans la norme HL7.- Positionnel. Les champs de données sont définis par le nombre de caractères entre eux.- PostScript. Les champs de données sont définis selon le format PostScript.- RTF. Les champs de données sont définis selon le format RTF.- SGML. Les champs de données sont définis selon le format SGML.- SpaceDelimited. Les champs de données sont séparés par des espaces.- TabDelimited. Les champs de données sont séparés par des tabulations. Pour plus d'informations, consultez la section "Documentation de référence du composant Délimiteurs" à la page 185 . La valeur par défaut est SGML.
nom	Libellé descriptif pour le composant. Ce libellé s'affiche dans le fichier journal et la vue Événements . Utilisez la propriété nom pour identifier le composant ayant provoqué l'événement.
pre_processor	Définit un préprocesseur de format qui traite l'entrée après tout processeur de document que vous avez défini pour la propriété pre_processor de exemple_source . Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none">- HtmlProcessor. Convertit toutes les combinaisons de tabulations, d'espaces ou de retours à la ligne en un seul caractère d'espacement. Il n'est pas restreint aux seuls documents HTML.- RtfProcessor. Normalise les fichiers RTF. La valeur par défaut est HtmlProcessor.
remarque	Commentaire défini par l'utilisateur, qui décrit l'objectif ou l'action du composant.

Documentation de référence du composant Délimiteurs

Un composant de délimiteurs définit une hiérarchie de caractères ou de chaînes qui organisent les informations dans un document, par exemple des retours à la ligne, des espaces, des onglets, des virgules ou barres verticales. Vous pouvez également utiliser un caractère générique pour définir les délimiteurs.

Le concept de délimiteur est applicable à la fois pour des documents de structure rigide, qui utilisent des caractères délimiteurs pour séparer les champs de données et pour des documents texte ou HTML de structure lâche délimités par des retours à la ligne et des marquages syntaxiques. Le concept de délimiteur englobe aussi les données structurées par position, où les champs sont placés avec des espacements fixes.

L'analyseur utilise les délimiteurs pour déterminer les critères de recherche des ancrs `Contenu` configurés avec l'option `LearnByExample`.

Supposez par exemple que vous configurez un format avec le composant de délimiteurs `TabDelimited`. Cela définit une hiérarchie utilisant les caractères suivants comme délimiteurs :

```
Newline
Tab
```

Vous pouvez définir une ancre `Contenu` située après deux tabulations suivant l'ancre `Marqueur` dans la source d'exemple, comme ceci :

```
MARKER<tab>abc<tab>CONTENT
```

Lorsqu'un analyseur traite un document source, il recherche `Contenu` deux tabulations après le `Marqueur`.

Dans un deuxième exemple, vous pouvez définir une ancre `Contenu` située trois lignes et une tabulation après une ancre `Marqueur`, dans la source d'exemple.

```
MARKER
abc<tab>de
fghi<tab>jkl<tab>mnop
pqrst<tab>CONTENT
```

Les tabulations ne sont pas comptées dans les lignes intermédiaires, car les retours à la ligne sont plus élevés dans la hiérarchie.

Beaucoup de composants délimiteurs, tels que `TabDelimited` ou `CommaDelimited`, affichent une hiérarchie prédéfinie de délimiteurs, que vous pouvez éditer selon vos besoins.

Le composant `DelimiterHierarchy` n'a pas de hiérarchie prédéfinie. Vous pouvez insérer tous les délimiteurs dont vous avez besoin.

CommaDelimited

Le composant de délimiteurs **`CommaDelimited`** définit la hiérarchie de délimiteurs :

```
Newline
Comma
```

Utilisez **`CommaDelimited`** lorsque chaque ligne d'un fichier texte contient un enregistrement et que chaque enregistrement contient des champs de données séparés par des virgules.

Vous pouvez ajouter des délimiteurs supplémentaires ou modifier la hiérarchie prédéfinie. Utilisez le même procédé que vous utilisez pour éditer le composant **`DelimiterHierarchy`**.

Exemple

Dans le document source, une ancre **`Contenu`** suit une ancre **`Marqueur`** deux lignes plus loin. Dans la troisième ligne, il y a trois virgules, ainsi que tout autre texte, avant l'ancre **`Contenu`** :

```
MARKER
abcdef, ghij
abc, def,ghi,CONTENT
```

Si vous assignez le composant **`CommaDelimited`**, l'analyseur apprend de l'exemple de source que l'ancre **`Contenu`** suit toujours le **`Marqueur`** de deux nouvelles lignes et de trois virgules. Dans un autre document source, l'analyseur trouvera toujours l'ancre **`Contenu`** suivante :

```
MARKER
    xyz, uvw, rst
,,,CONTENT
```

Délimiteur

Le sous-composant **Délimiteur** définit un caractère ou une chaîne délimiteur qui sépare les ancres. Vous pouvez ajouter des sous-composants **Délimiteur** dans la hiérarchie de délimiteurs.

Le tableau suivant décrit les propriétés du sous-composant **Délimiteur** :

Propriété	Description
rechercher	Définit le délimiteur. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none">- NewlineSearch. Le délimiteur est un retour à la ligne.- PatternSearch. Le délimiteur est défini par une expression régulière.- TextSearch. Le délimiteur est une chaîne explicite ou une chaîne que vous récupérez dynamiquement depuis le document source. Pour plus d'informations, voir "Référence du composant de recherche" à la page 246

Exemple

Le composant `TabDelimited` contient deux sous-composants `Délimiteur`. Le premier utilise `NewlineSearch` pour définir le caractère de retour à la ligne comme délimiteur. Le second utilise `TextSearch` pour définir le caractère de tabulation comme délimiteur. La tabulation est représentée graphiquement par le caractère «.

Le composant `SpaceDelimited` contient également deux sous-composants `Délimiteur`. Le premier est identique à celui de `TabDelimited`. Le second utilise un `PatternSearch` pour définir toute chaîne d'un ou plusieurs espaces comme délimiteur. L'expression régulière `[]+` signifie « un ou plusieurs caractères espace ». Notez l'espace entre les crochets.

DelimiterHierarchy

Le composant de délimiteurs **DelimiterHierarchy** vous permet de définir une hiérarchie de délimiteurs personnalisée.

Sous **DelimiterHierarchy**, vous pouvez imbriquer tout nombre de composants **Délimiteur** ou **EnclosingDelimiters**.

Exemple

Supposez que, dans l'échantillon de documents de document source, les ancres soient séparées par des virgules et entourées de crochets de la façon suivante :

```
MARKER,, [CONTENT]
```

Vous pourriez définir un composant **DelimiterHierarchy** contenant :

```
comma //défini comme un composant Délimiteur  
[] //Défini comme un composant EnclosingDelimiters
```

Dans cet exemple, l'analyseur apprend que l'ancre **Contenu** suit le **Marqueur** après deux virgules et est entourée de crochets. Dans un autre document source, l'analyseur trouvera l'ancre **Contenu** suivante :

```
MARKER,abc,def[CONTENT]
```

Exemple en ligne

Pour obtenir un échantillon en ligne, consultez `samples\Projects\EDI\EDI.cmw`. L'exemple utilise un **DelimiterHierarchy** pour définir le retour à la ligne et un astérisque (*) comme délimiteurs, dans un document source EDI.

EnclosingDelimiters

Le sous-composant **EnclosingDelimiters** définit une paire de caractères ou de chaînes délimiteurs, entourant les ancrs. Vous pouvez ajouter des sous-composants **EnclosingDelimiters** sous une hiérarchie de délimiteurs.

Vous pouvez utiliser ce composant pour définir les délimiteurs accolades ({}) qui entourent les blocs de code de programme C.

Le tableau suivant décrit les propriétés du sous-composant **EnclosingDelimiters** :

Propriété	Description
ouverture	Définit le délimiteur d'ouverture.
fermeture	Définit le délimiteur de fermeture.
escape_sequence	Définit un préfixe qui fait que l'analyseur ignore une instance du délimiteur d'ouverture ou de fermeture dans le document source.

HL7

Le composant **HL7** des délimiteurs définit la hiérarchie suivante des délimiteurs pour l'analyse des messages HL7 :

```
newline  
vertical bar (|)  
caret (^) or tab
```

Vous pouvez ajouter des délimiteurs supplémentaires ou modifier la hiérarchie prédéfinie. La procédure est identique à celle utilisée pour le composant **DelimiterHierarchy** .

La norme de messagerie HL7 autorise un message à définir ses propres délimiteurs. Vous pouvez analyser la déclaration des délimiteurs d'un message HL7 et créer une définition dynamique des délimiteurs de la manière suivante :

1. Utilisez les ancrs **Contenu** pour récupérer les caractères délimiteurs dans l'en-tête du message HL7. Stockez les caractères dans des variables.
2. Ajoutez des composants **Délimiteur** sous le composant **HL7**.
3. Assignez **TextSearch** à chaque composant **Délimiteur**.
4. Sous le composant **TextSearch**, assignez l'une des variables à la propriété **texte**.

Positionnel

Le composant de délimiteurs **Positional** contraint l'analyseur à trouver des ancrs de contenu en comptant les caractères depuis début de la zone de la recherche. Pour plus d'informations sur la portée de la recherche, voir ["Présentation des ancrs" à la page 205](#).

Exemple

Dans l'échantillon de échantillons de document source, imaginons qu'une ancre **Contenu** suive une ancre **Marqueur** à cinq caractères de distance, éventuels espaces, tabulations, etc. compris.

```
MARKERab cdCONTENTefg
```

Si vous assignez le composant **Positionnel**, l'analyseur apprend de l'exemple de source que l'ancre **Contenu** suit toujours l'ancre **Marqueur** de cinq caractères et qu'elle a une longueur de sept caractères. Dans un autre document source, l'analyseur trouvera toujours l'ancre **Contenu** suivante :

```
MARKERd<tab>cbaCONTENTzy,xwv
```

Utiliser l'analyse positionnelle avec des délimiteurs

Vous ne pouvez pas ajouter de délimiteurs au composant **Positionnel**.

Il peut parfois être nécessaire de définir un analyseur qui utilise des délimiteurs pour localiser certaines ancres et d'utiliser une définition positionnelle pour les autres ancres. Pour ce faire, sélectionnez l'un des autres composants de délimiteurs. N'utilisez pas **Positionnel**. Pour définir l'emplacement d'une ancre selon la méthode positionnelle, vous pouvez assigner l'option **OffsetSearch** dans les propriétés de l'ancre.

PostScript

Le composant **PostScript** des délimiteurs définit la hiérarchie des délimiteurs utilisée pour l'analyse des documents Adobe PostScript.

Vous ne pouvez pas éditer la hiérarchie de délimiteurs du composant **PostScript**.

RTF

Le composant de délimiteurs **RTF** définit une hiérarchie de délimiteurs pour l'analyse de documents RTF.

Vous ne pouvez pas éditer la hiérarchie de délimiteurs du composant **RTF**.

SGML

Les séparateurs **SGML** définissent une hiérarchie de séparateurs pour l'analyse des documents SGML, HTML et XML.

Vous ne pouvez pas éditer la hiérarchie des séparateurs du composant **SGML**.

SpaceDelimited

Le composant des séparateurs **SpaceDelimited** définit la hiérarchie de séparateurs suivante :

```
Newline
String of one or more space characters
```

SpaceDelimited est utilisé lorsque toutes les lignes d'un fichier texte contiennent un enregistrement et que chaque enregistrement contient des champs de données séparés par des espaces.

Vous pouvez ajouter des délimiteurs supplémentaires ou éditer la hiérarchie prédéfinie. La procédure est la même que celle utilisée pour le composant **DelimiterHierarchy**.

Exemple

Dans l'échantillon de document source, supposez qu'une ancre **Contenu** suive une ancre **Marqueur** deux lignes plus loin. Dans la troisième ligne, il y a deux caractères espace et une chaîne contenant plusieurs espaces avant l'ancre **Contenu**, comme suit :

```
MARKER
abcdef
abc def ghi          CONTENT
```

Si vous assignez le composant **SpaceDelimited**, l'analyseur apprend de l'exemple de source que l'ancre **Contenu** suit toujours le **Marqueur** de deux lignes et trois chaînes d'espaces. Dans un autre document source, l'analyseur trouvera toujours l'ancre **Contenu** suivante :

```
MARKER
      xyz
ghi    def abc CONTENT
```

TabDelimited

Le composant du délimiteur **TabDelimited** définit la hiérarchie de délimiteurs suivante :

```
Newline
Tab
```

TabDelimited est utilisé lorsque chaque ligne d'un fichier texte contient un enregistrement, et que chaque enregistrement contient des champs de données séparés par des tabulations.

Vous pouvez ajouter des délimiteurs supplémentaires ou éditer la hiérarchie prédéfinie. La procédure est la même que celle utilisée pour le composant **DelimiterHierarchy**.

Exemple

Dans l'échantillon de document source, supposez qu'une ancre **Contenu** suive une ancre **Marqueur** deux lignes plus loin. Dans la troisième ligne, il y a trois caractères de tabulation, ainsi que tout autre texte, avant l'ancre **Contenu**, comme ceci :

```
MARKER
abcdef
abc<tab> de,f<tab>ghi<tab>CONTENT
```

Si vous assignez le composant **TabDelimited**, l'analyseur apprend de l'exemple de source que l'ancre **Contenu** suit toujours le **Marqueur** de deux lignes et trois tabulations. Dans un autre document source, l'analyseur trouvera toujours l'ancre **Contenu** suivante :

```
MARKER
      xyz
<tab><tab><tab>CONTENT
```

Documentation de référence du composant préprocesseur de format

La liste suivante décrit les différences entre les préprocesseurs de format et les processeurs de document.

- Vous pouvez assigner un processeur de document à la propriété **pre_processor** d'un port d'entrée, situé dans la propriété **example_source** ou **sources_to_extract** d'un analyseur. Vous ne pouvez assigner un préprocesseur de format qu'à la propriété **pre_processor** d'un format.
- Un processeur de document s'exécute sur un document source avant d'effectuer toute autre opération.
- Un préprocesseur de format s'exécute sur le texte avant de rechercher les ancres. La sortie d'un préprocesseur de format n'est pas affichée.

Pour plus d'informations, consultez la section ["Présentation des processeurs de document" à la page 163](#)

HtmlProcessor

Le préprocesseur de format **HtmlProcessor**, qui fonctionne également comme un transformateur, normalise les espaces conformément aux conventions HTML. Il réduit toute combinaison de tabulations, de retours à la ligne et d'espaces en un seul caractère d'espacement.

Utilisez ce préprocesseur pour normaliser les espaces dans tout type de texte. Il n'est pas restreint aux seuls documents HTML.

RtfProcessor

Le préprocesseur de format **RtfProcessor** normalise le code des fichiers RTF.

CHAPITRE 15

Zones de stockage des données

Ce chapitre comprend les rubriques suivantes :

- [Présentation des zones de stockage des données, 192](#)
- [Schémas XML, 192](#)
- [Utiliser un schéma pour mapper des ancres, 195](#)
- [Génération de XML valide, 196](#)
- [Variables, 198](#)
- [Documentation de référence du composant Variable, 201](#)
- [Zones de stockage des données à occurrences multiples, 202](#)

Présentation des zones de stockage des données

Une zone de stockage des données est un objet dont le type est l'un des suivants :

- Un élément XML
- Un attribut XML
- Une variable

Les attributs et les éléments XML sont généralement utilisés dans le cadre de stockages permanents. Un analyseur, par exemple, stocke sa sortie dans l'un de ces types de zone de stockage des données.

Les variables sont utilisées pour un stockage temporaire. Par exemple, un analyseur peut stocker les données qu'il extrait d'un document source dans une variable. Il peut traiter davantage les données avant la création de la sortie.

Chaque zone de stockage des données a un type de données. Dans le cas des éléments et des attributs, les zones de stockage des données sont définies dans un schéma XML que vous devez fournir. Les variables sont définies dans un schéma interne que vous pouvez personnaliser en ajoutant des variables définies par l'utilisateur.

Schémas XML

Lorsque vous créez un analyseur, un sérialiseur, une XMap ou un mappeur, vous devez fournir un ou plusieurs schémas XML définissant la structure du XML. Le schéma définit les éléments et les attributs que la transformation peut utiliser.

Ajoutez le schéma au référentiel modèle. Vous pouvez alors mapper le contenu d'un document aux éléments et attributs définis dans le schéma.

Codage du schéma

Enregistrez le schéma dans un des codages d'entrée pris en charge.

Le codage de schéma doit être compatible avec le codage de travail que vous utilisez dans l'éditeur IntelliScript. Cela signifie que :

- Le codage du schéma est identique au codage de travail, ou
- Chaque caractère du schéma a son équivalent dans le codage de travail. Par exemple, si le schéma utilise l'encodage UTF-8 et que l'encodage de travail est Windows-1252, le schéma ne doit pas contenir de caractères Unicode qui n'ont pas d'équivalent dans Windows-1252.

Lorsque vous ajoutez à un projet un schéma provenant d'un emplacement externe, le script traduit la copie du schéma du projet dans l'encodage de travail.

Fichiers de schéma inclus

Un schéma peut référencer d'autres fichiers de schéma. Cette fonctionnalité permet de gérer un schéma de grande taille sous une forme modulaire.

Espaces de nommage

Si vous prévoyez de travailler avec des espaces de nommage XML, vous devrez assigner l'attribut **targetNamespace** du schéma. Vous pouvez modifier l'alias qui est affecté à l'espace de nom.

Vous ne pouvez pas ajouter deux schémas qui utilisent un alias vide pour des espaces de nom différents ou deux schémas qui utilisent le même alias pour des espaces de nom différents.

Contenu mixte

Les éléments peuvent contenir des données de caractères et des éléments imbriqués. Vous pouvez utiliser l'attribut **mixte** dans un schéma.

Le script distingue les données de caractères avant et après chaque élément. Pour plus d'informations, voir ["Mappage de contenu mixte" à la page 195](#)

Fonctionnalités de schéma non prises en charge

La version actuelle ne prend pas en charge certaines fonctionnalités de schéma. Le tableau suivant présente les limitations connues :

Fonction	Limitation
Contraintes d'unicité	Les éléments unique , key et keyref sont ignorés. Le journal des événements inclut un avertissement.
Valeurs par défaut des éléments de type mixte	Le script ignore la valeur par défaut. Le journal des événements inclut un avertissement.

Fonction	Limitation
Type de données par défaut	Si le type d'un élément n'est pas défini, le script traite ce dernier comme <code>xs:string</code> . Le journal des événements inclut un avertissement. Vous pouvez changer la valeur par défaut sur <code>xs:anyType</code> .
Expressions régulières	Il existe des différences mineures entre le processeur d'expression régulière et le schéma standard.
Séquence définissant plusieurs éléments portant le même nom	Si une <code>xs:sequence</code> contient plusieurs définitions <code>xs:element</code> portant le même nom, le script traite seulement la première <code>xs:element</code> . Le journal des événements inclut un avertissement. Pour résoudre le problème, intégrez chaque <code>xs:element</code> à une <code>xs:sequence</code> indépendante.
Dates minimum et maximum	Si une facette définit une valeur minimum ou maximum pour un élément <code>xs:date</code> , la transformation échoue.
Assouplir ou ignorer les options de validation	Dans un élément <code>xs:any</code> ou <code>xs:anyAttribute</code> , le script ignore une valeur processContents de lax ou skip . Il se comporte comme si la valeur était strict .
Groupe de substitution	Le script autorise un substitutionGroup , même si un attribut block ou blockDefault interdit les substitutions.
Type XSI	Le script autorise un attribut <code>xsi:type</code> même si un attribut block de l'interdit.
Types intégrés	Certains types intégrés ne comportent pas les modèles corrects, par exemple, lorsqu'ils incluent des caractères supérieurs à ASCII 127.
Groupe de substitution sans type	Le script échoue parfois quand un groupe de substitution n'a pas de type.
Espace de nom vide	Lorsque l'espace de nom est vide, le script ajoute un alias à tous les éléments du fichier source mais l'alias n'apparaît pas sur le localisateur et le localisateur échoue.
Liste	Le script lit une <code>xs:list</code> séparée par des espaces comme un seul élément, ce qui peut échouer si sa longueur dépasse la limite indiquée pour chaque élément de la liste.
Flottants et doublons	<code>xs:float</code> et <code>xs:double</code> n'acceptent pas les valeurs valides de INF , -INF ou NaN .
Élément avec attributs fixes et mixtes	Le script ne lit pas toutes les parties d'un élément qui comprend des attributs à la fois fixes et mixtes.
<code>maxOccurs=0</code>	Le script crée une sortie même quand <code>maxOccurs=0</code> .
Jeton	Le script n'analyse pas un <code>xs:token</code> qui contient des tabulations, des retours chariots ou des sauts de ligne.
Chaîne normalisée	Le script ne charge pas le XML lorsqu'une <code>xs:normalizedString</code> contient des tabulations, des retours chariots ou des sauts de ligne.

Précision des données numériques

Le script stocke les données `xs:decimal` et `xs:float` sous forme de chaînes, en conservant la précision des données.

Dans les calculs, le script convertit les données décimales et à virgule flottante en virgule flottante double précision et il arrondit le résultat à 15 chiffres décimaux. Cela signifie que les données décimales peuvent perdre en précision. Par exemple, le résultat de `xs:decimal 5,28 * 1` peut être affiché sous la forme 5,280000000000001.

Le script normalise les valeurs `xs:decimal`. Par exemple, il stocke 0004 comme 4, -0 comme 0 et 1.200 comme 1.2.

Utiliser un schéma pour mapper des ancres

Lorsque vous définissez un analyseur, vous devez mapper des ancres `Contenu` sur des zones de stockage des données de sortie. Lorsque vous définissez un sérialiseur, vous devez mapper des zones de stockage des données d'entrée à des ancres de sérialisation `ContentSerializer`.

Représentation des zones de stockage des données dans IntelliScript

Dans le script, les zones de stockage des données sont identifiées par une expression XPath modifiée, comme :

```
data_holder = /Person/*s/Name/*s/First
```

Pour changer cette valeur, sélectionnez la propriété **data_holder** et appuyez sur **Entrée**. Ceci s'ouvre une boîte de dialogue **Choisir XPath** depuis laquelle vous pouvez sélectionner la nouvelle valeur.

La syntaxe XPath est légèrement différente de la syntaxe XPath standard, qui est `Personne/Nom/Prénom`. Le script insère `*s`, `*c` et `*a`, faisant référence aux termes de schéma **séquence**, **choix** et **tout**. Les modifications résolvent les ambiguïtés lorsque le script utilise le schéma pour aider à construire la sortie XML.

Mappage de contenu mixte

Si le schéma prend en charge un contenu mixte, chaque élément possède les zones de stockage des données **avant** et **après**. Par exemple, prenez le contenu mixte suivant :

```
<Deal>
  We are pleased to offer you a price of
  <Price>34</Price>
  dollars. This is a special price for
  <Partner>
    <Name>Acme Gizmos, Inc.</Name>
    <ID>98765</ID>
  </Partner>
  valid only until December 31.
</Deal>
```

Cette structure contient des zones de stockage des données localisés dans les emplacements suivants :

- Immédiatement après la balise `<Deal>` et avant tout sous-élément.
- Avant l'élément `Prix`
- L'élément `Prix`
- Après l'élément `Prix`
- Avant l'élément `Partenaire`
- Les éléments `Partenaire/Nom` et `Partenaire/ID`

- Après l'élément `Partenaire`
- Immédiatement avant la balise `</Deal>` et après tous les sous-éléments.

Vous pouvez mapper le texte "Nous sommes heureux de vous faire bénéficier d'un prix de" au zone de stockage des données avant l'élément `Prix`. Vous pouvez mapper "dollars. " au zone de stockage des données après `Prix` et "Il s'agit d'un prix spécial pour " au zone de stockage des données avant `Partenaire`.

L'exemple suivant illustre le contenu mixte :

```
data_holder = /Deal/*s/Price/$text_before
```

Mappage de types XSI

Un schéma peut définir des types de données dérivés qui peuvent être utilisés à la place d'un type de base. Dans de tels cas, un document XML peut définir les types de données d'un élément en définissant un attribut `xsi:type`.

Par exemple, un schéma définit un élément `Personne` ayant un type `PersonT1` et contenant un contenu de chaîne. Il définit un type appelé `PersonT2` qui dérive `PersonT1` par l'ajout d'un attribut `Id`. Les éléments suivants sont des éléments `Personne` valides :

```
<!-- base type PersonT1 -->
<Person>Ron Lehrer</Person>

<!-- derived type PersonT2 -->
<Person Id="547329876" xsi:type="PersonT2">Ron Lehrer</Person>
```

Le script interprète les attributs `xsi:type` dans les documents XML d'entrée. Il ajoute les attributs `xsi:type` où cela est nécessaire pour produire les documents XML.

Sélectionnez le type approprié selon les données traitées par la transformation. Par exemple, si vous voulez qu'une ancre `Contenu` stocke les données dans un élément `Personne` avec le type `PersonT2`, sélectionnez `xsi:type=PersonT2`. La sélection s'affiche dans le script comme suit :

```
data_holder=/Person/*c/xsi:type=PersonT2
```

Dans les cas où le contenu peut nécessiter une zone de stockage des données `PersonT1` ou `PersonT2`, vous pouvez configurer une ancre **Alternatives** qui contient deux ancres **Contenu**. L'une des ancres **Contenu** est mappée à `PersonT1` et l'autre à `PersonT2`. Pour plus d'informations, voir ["Alternatives" à la page 217](#)

Si vous mappez une zone de stockage des données à l'élément non qualifié `Personne`, la zone de stockage des données prend le type de base `PersonT1` par défaut. Par conséquent les mappages suivants sont équivalents :

```
data_holder=/Person
data_holder=/Person/*c/xsi:type=PersonT1
```

Génération de XML valide

Le script génère un XML valide selon le schéma de sortie que vous avez défini.

Le schéma est utilisé en tant que guide pendant la génération du XML. Le schéma est appliqué pendant la génération et non après. Cette approche améliore les chances de réussite des transformations. Elle garantit la validité tout au long de l'exécution de la transformation.

Rôle des schémas dans l'analyse

Cette section explique quelques-unes des façons dont un analyseur utilise le schéma pour garantir la production d'une sortie XML.

La discussion présente des exemples de comportement.

Séquence des éléments

Lorsque le script exécute un analyseur, il organise la sortie dans l'ordre requis par le schéma.

Par exemple, un schéma peut requérir qu'un élément **LastName** précède un élément **FirstName**. Le script crée la sortie dans les emplacements définis par le schéma, même si les ancres qui produisent la sortie sont définies dans l'ordre inverse.

Nombre d'occurrences

Un analyseur peut tenter d'insérer plusieurs instances d'un élément dans la sortie XML. Le script utilise le schéma pour déterminer s'il faut joindre de nouvelles instances ou remplacer les éléments existants. L'analyseur supprime les éléments en excès au-delà de ceux permis par le schéma et consigne des avertissements dans le journal des événements.

Dans un autre exemple, supposons que le schéma définisse un élément sans spécifier un attribut **minOccurs** ou **maxOccurs**. La valeur par défaut de **minOccurs** et **maxOccurs** est 1, ce qui indique que l'élément doit se produire exactement une fois dans la sortie de l'analyseur. Si l'élément est manquant dans la sortie, l'analyseur peut l'ajouter.

Pour plus d'informations, voir ["Zones de stockage des données à occurrences multiples" à la page 202](#)

Éléments manquants ou vides

Dans les paramètres de la transformation Processeur de données, vous pouvez décider si un analyseur insère ou non des éléments vides pour respecter un schéma.

Types de données

Le script garantit que le texte stocké dans une zone de stockage des données a le type de données requis. Par exemple, si une ancre **Contenu** extrait la chaîne « 5 oranges pour un dollar » et que le type de la zone de stockage des données est `xs:integer`, alors l'ancre ne stocke que l'entier 5 dans la zone de stockage des données.

Pour plus d'informations, voir ["Utiliser les types de données pour affiner les critères de recherche" à la page 215](#)

Rôle des schémas dans la sérialisation et le mappage

Un sérialiseur ou un mappeur vérifie que son entrée est valide d'après le schéma XML. Il y a deux modes de validation :

- Validation partielle. Certaines déviations sont autorisées entre le document source XML et le schéma. Valeur par défaut.
- Validation stricte. Le document XML source doit être strictement conforme à son schéma.

Pour définir le niveau de validation, affectez la propriété **validate_source_document** du composant **Sérialiseur** ou **Mappeur**.

Si vous utilisez le mode strict, une erreur de validation entraîne l'échec du sérialiseur ou du mappeur. La vue **Événements** affiche les erreurs.

Si vous utilisez le mode partiel, la transformation peut continuer malgré certaines erreurs de validation. Par exemple, s'il y a plus d'occurrences d'un élément que ne l'autorise le schéma, un sérialiseur ignore généralement les éléments en trop, traite ceux qui sont valides et écrit un avertissement dans le journal des événements. De même, il peut ignorer un élément contenant un type de données non valide.

Le script utilise l'analyseur XML Xerces C, version 3.1, pour effectuer la validation.

Variables

Les variables sont des zones de stockage des données temporaires que vous pouvez utiliser à la place d'éléments ou d'attributs XML. Les variables sont utiles si vous devez stocker temporairement une valeur lors de l'opération d'une transformation, et que vous n'avez pas besoin de sortir la valeur dans le XML.

Par exemple, supposons que vous souhaitiez qu'un analyseur lise deux ancres **Contenu** et concatène leurs valeurs. Vous pouvez mapper chaque ancre **Contenu** à une variable définie par l'utilisateur. Vous pouvez alors utiliser une action pour concaténer les variables et sortir le résultat dans un élément XML.

Le script utilise également des variables système prédéfinies pour stocker les informations nécessaires dans certaines opérations.

Création d'une variable définie par l'utilisateur

1. Ajoutez un composant **Variable** au niveau global du script.
2. Entrez un nom pour la variable, puis appuyez sur **ENTRÉE**.
3. Sélectionnez le type de données que la variable peut stocker.

Vous pouvez sélectionner un type de standard comme `xs:string` ou `xs:integer`, ou un type global défini dans un schéma référencé dans le projet.

Variables système

Les paragraphes suivants décrivent les variables système et les façons de les utiliser.

Variables utilisées pour accéder aux documents sources

Plusieurs variables système stockent des données que les actions peuvent utiliser lorsqu'elles accèdent à des documents source. Par exemple, l'action **RunParser** peut utiliser `VarLinkURL`, qui contient un chemin de fichier.

La variable suivante est utilisée dans le processeur **XmlToDocument** :

Variable	Description
VarServiceInfo > <i>ServiceLocation</i>	Chemin d'accès au répertoire du script ou du service en cours d'exécution.

Variables d'accès en lecture seule

Les variables suivantes sont en lecture seule : Une transformation peut les utiliser visiter un document source plusieurs fois.

Variable	Description
<i>VarRequestedURL</i>	Le chemin du document source en cours de traitement par un analyseur.
<i>VarCurrentURL</i>	Le chemin du fichier actuellement traité par un analyseur. C'est généralement la même chose que VarRequestedURL . Si l'analyseur est configuré avec certains préprocesseurs, VarCurrentURL peut pointer vers un fichier temporaire plutôt que vers le document source original. VarRequestedURL pointe toujours sur le document source.
<i>VarCurrentPost</i>	Les données de formulaire soumises par un analyseur pour récupérer la page en cours.

Variables système de temps en lecture seule

VarSystem est une variable en lecture seule qui renvoie des informations système. Il s'agit d'une structure contenant plusieurs variables imbriquées :

Variable	Description
VarSystem > ExecStartTime > <i>Year</i>	Année au cours de laquelle la transformation a commencé l'exécution
VarSystem > ExecStartTime > <i>Month</i>	Mois numérique
VarSystem > ExecStartTime > <i>MonthName</i>	Nom du mois
VarSystem > ExecStartTime > <i>Day</i>	Jour du mois
VarSystem > ExecStartTime > <i>DayName</i>	Jour de la semaine
VarSystem > ExecStartTime > <i>Hour</i>	Heure
VarSystem > ExecStartTime > <i>Minute</i>	Minute
VarSystem > ExecStartTime > <i>Second</i>	Seconde
VarSystem > ExecStartTime > <i>Millisecond</i>	Milliseconde

Vous pouvez utiliser **VarSystem** pour insérer un horodatage dans la sortie d'une transformation.

Variables utilisées pour le traitement des échecs

VarLastFailure stocke l'échec de composant qui s'est produit le plus récemment dans une transformation. Par exemple, il peut enregistrer l'instance d'une ancre **Marqueur** qui n'est pas parvenue à trouver le texte du marqueur. Vous pouvez configurer un composant pour écrire **VarLastFailure** dans un journal utilisateur en cas d'échec. Pour plus d'informations, voir ["Traitement des échecs" à la page 404](#)

Remarque: Lorsque vous utilisez **VarLastFailure**, le service fonctionne en mode spécial, ce qui demande environ trois fois plus de temps d'UC.

VarServiceInfo stocke le nom du service, l'emplacement du répertoire journal utilisateur et le nom de fichier du journal utilisateur.

VarLastFailure et **VarServiceInfo** sont des structures qui contiennent les variables imbriquées suivantes :

Variable	Description
VarLastFailure > <i>InternalId</i>	Identificateur de l'échec
VarLastFailure > <i>Text</i>	Description de l'échec
VarLastFailure > <i>Location</i>	Emplacement de l'échec dans le script
VarLastFailure > <i>AnchorName</i>	Nom du composant qui a échoué
VarLastFailure > <i>Data</i>	Informations supplémentaires sur l'échec
VarServiceInfo > <i>ServiceName</i>	Nom du service
VarServiceInfo > StandardError > <i>StandardErrorDir</i>	Chemin du répertoire du journal utilisateur
VarServiceInfo > StandardError > <i>StandardErrorName</i>	Nom de fichier du journal utilisateur

Variables utilisées pour l'analyse structurée

VarStructureDetails conserve une trace de l'enregistrement actuel qu'une ancre **StructureDefinition** analyse. Il contient les variables imbriquées suivantes :

Variable	Description
VarStructureDetails > <i>Name</i>	Propriété name du sous-élément qui correspond à l'enregistrement.
VarStructureDetails > <i>Repetitions</i>	Nombre d'itérations de l'enregistrement.
VarStructureDetails > <i>RecordIndex</i>	Nombre d'index de l'enregistrement dans l'entrée globale StructureDefinition .
VarStructureDetails > <i>RecordId</i>	Identifiant de l'enregistrement. S'il existe plusieurs identifiants, la variable contient une liste séparée par des virgules.
VarStructureDetails > <i>InternalPath</i>	Informations internes, à ne pas utiliser.

Pour plus d'informations, voir ["StructureDefinition" à la page 242](#)

Variables utilisées dans les notifications

VarNotificationDetails stocke les informations concernant la notification qui a été déclenchée le plus récemment dans une transformation :

Variable	Description
VarNotificationDetails > <i>Name</i>	Nom de la notification.
VarNotificationDetails > <i>Path</i>	Chemin XPath de la zone de stockage des données à laquelle s'applique la notification. Par exemple, si un validateur déclenche une notification dans une ancre Contenu , le chemin correspond à la zone de stockage des données dans laquelle l'ancre Contenu stocke sa sortie.

Variable	Description
VarNotificationDetails > <i>Value</i>	Valeur d'entrée à l'origine de la notification. Si un validateur déclenche la notification, la valeur est la donnée d'entrée non valide. Si une action Notify déclenche la notification, vous pouvez indiquer la valeur dans la configuration Notify .
VarNotificationDetails > <i>Creator</i>	Emplacement dans le script où la notification a été déclenchée.

Pour plus d'informations, voir ["Notifications" à la page 423](#)

Mappage d'ancres aux variables

Vous pouvez mapper une ancre **Contenu** à une variable, de la même façon que vous mappez à toute autre zone de stockage des données.

Ne mappez pas une ancre à une variable système en lecture seule.

Utilisation de variables dans les actions

Les variables sont souvent utilisées en entrées d'actions. Vous pouvez utiliser une variable de la même façon que vous utilisez les autres zones de stockage des données. Pour plus d'informations, voir ["Présentation des actions" à la page 297](#)

Initialisation de variables lors de l'exécution

Vous pouvez initialiser les valeurs des variables d'une des manières suivantes :

- Dans le script, vous pouvez configurer la propriété **initialisation** de la **Variable**.
Les valeurs initiales que vous définissez avec cette approche sont utilisées lorsque vous exécutez la transformation dans l'outil Developer ou en tant que service.
- Une application peut transmettre les valeurs initiales comme paramètres de service pour un service lors de l'exécution.
Les paramètres de service remplacent la propriété **initialisation** des variables. Si le script spécifie une valeur initiale et que vous transmettez aussi une valeur depuis une application, cette dernière valeur est utilisée.

Documentation de référence du composant Variable

Un composant **Variable** est une variable définie par l'utilisateur.

Pour plus d'informations sur les variables système, voir ["Variables système" à la page 198](#).

Variable

Un composant **Variable** est une variable, définie par l'utilisateur, que vous utilisez dans un script.

Utilisez des variables pour le stockage temporaire de la même manière que vous utilisez un élément ou un attribut XML. Par exemple, vous pouvez mapper une ancre **Contenu** à une variable, et vous pouvez utiliser une variable en tant qu'entrée d'une action.

Les variables s'affichent au niveau global du script. Une variable peut être de n'importe quel type de données défini dans les schémas associés au projet, y compris les types standard et personnalisés. Un type personnalisé peut être soit simple, soit complexe. Une variable complexe est une structure qui contient des champs imbriqués. L'initialisation de variables complexes n'est pas prise en charge. Pour plus d'informations, voir ["Initialisation de variables lors de l'exécution" à la page 201](#)

Le tableau suivant décrit les propriétés du composant **Variable** :

Propriété	Description
initialisation	Définit une valeur initiale pour la variable. Vous pouvez initialiser des variables de types de données simples. Par défaut, la valeur est vide.
InitialValue	Définit une valeur initiale pour la variable. InitialValue a une propriété, value .
liste	Détermine si la variable a une ou plusieurs occurrences. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none">- Sélectionné. Détermine une variable à plusieurs occurrences.- Vide. Détermine une variable à occurrence unique. Par défaut, la valeur est vide. Pour plus d'informations, voir "Zones de stockage des données à occurrences multiples" à la page 202
val_type	Définit le type de données que la variable peut stocker. Les valeurs légales sont définies dans le schéma. La valeur par défaut est <code>xs:string</code> .
valeur	Définit la valeur initiale.

Zones de stockage des données à occurrences multiples

Dans un schéma, vous pouvez utiliser l'attribut **maxOccurs** pour définir le nombre maximum de fois que les éléments frères peuvent apparaître dans un document XML. De la même façon, vous pouvez définir une variable qui apparaît soit une seule fois, soit plusieurs fois. Un élément ou une variable qui ne peut apparaître qu'une fois est appelé zone de stockage des données à occurrence unique. Un élément ou une variable qui peut se produire plusieurs fois est appelé une zone de stockage des données à occurrences multiples.

Les zones de stockage des données à occurrence simple ou multiples ont un comportement différent lorsque le script y stocke des données, par exemple lorsque vous mappez des ancres **Contenu** à une zone de stockage des données.

- Dans une zone de stockage des données à occurrence unique, chaque affectation remplace l'affectation précédente.
- Dans une zone de stockage des données à occurrences multiples, chaque affectation génère une nouvelle occurrence du zone de stockage des données.

Pour le comprendre, imaginons qu'un schéma définisse un élément XML nommé `<Prénom>`. Si `maxOccurs = 1`, il s'agit d'une zone de stockage des données à occurrence unique. Si un analyseur mappe plusieurs ancres **Contenu** à l'élément `<Prénom>`, la sortie ne contient que le mappage final.

Imaginez ce qui devrait se produire si vous analysez un document source qui est une liste de prénoms :

```
Jack Jennie Larissa
```

Nous supposons que chaque nom est une ancre **Contenu** mappée à **Prénom**. Chaque nom remplace la valeur de **Prénom**. La sortie ne contient que le mappage :

```
<FirstName>Larissa</FirstName>
```

Imaginons maintenant que `maxOccurs = unbounded`. Cela signifie que **Prénom** est une zone de stockage des données à occurrences multiples. Si vous mappez plusieurs ancres **Contenu** sur l'élément, l'analyseur génère une liste de noms. La sortie est :

```
<FirstName>Jack</FirstName>
<FirstName>Jennie</FirstName>
<FirstName>Larissa</FirstName>
```

Le même principe s'applique aux variables. Si vous mappez plusieurs ancres à une variable à occurrences multiples, chaque ancre génère une nouvelle occurrence de la variable. Vous pouvez utiliser cette fonction, par exemple, pour préparer l'entrée des actions **AppendListItems** et **CombineValues** qui concatènent les occurrences.

Remarque: Le comportement décrit ici suppose que la zone de stockage des données à occurrences multiples a un type de données simple. Dans certains cas, si le type est complexe, chaque ancre peut ne pas générer une nouvelle occurrence. Pour contrôler ce comportement, vous pouvez utiliser un localisateur. Pour plus d'informations, consultez la section ["Présentation des Localisateurs, des Clés et de l'Indexation" à la page 364](#).

Attributs

Un attribut XML est toujours une zone de stockage des données à occurrence unique. Un attribut ne peut pas être à occurrences multiples, car XML ne permet pas au même attribut d'apparaître plus d'une fois dans le même élément.

Un attribut peut avoir un type de données qui est une liste séparée par des espaces. L'attribut `noms` de l'élément suivant est un exemple :

```
<Countries names="USA Canada Mexico"/>
```

Le script traite l'attribut comme une zone de stockage des données à occurrence unique avec un type de liste. Pour plus d'informations, voir ["Utiliser les types de données pour affiner les critères de recherche" à la page 215](#)

Indexation

Par défaut, le script accède à des instances d'une zone de stockage des données à occurrences multiples de façon séquentielle. Vous pouvez accéder aux instances de façon non séquentielle en utilisant la fonction d'indexation. Pour plus d'informations, voir ["Présentation des Localisateurs, des Clés et de l'Indexation" à la page 364](#)

Destruction des occurrences

Dans certains cas, vous pouvez décider de détruire toutes les occurrences existantes d'une zone de stockage des données à occurrences multiples et commencer à créer de nouvelles occurrences depuis le début de la liste. Ceci est utile, par exemple, si vous effectuez une analyse d'une structure itérative et que vous ne

souhaitez conserver que la dernière itération. Vous pouvez détruire les occurrences qui stockent les données depuis les itérations précédentes.

Vous pouvez obtenir cet effet en définissant une zone de stockage des données à occurrence unique contenant un élément imbriqué à occurrences multiples. Lorsque vous réutilisez la zone de stockage des données à occurrence unique, les occurrences imbriquées sont détruites.

Le scénario suivant est un exemple typique.

1. Ajoutez le schéma suivant à un projet :

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
  elementFormDefault="qualified" attributeFormDefault="unqualified">
  <xs:complexType name="MyListType">
    <xs:sequence>
      <xs:element name="item" type="xs:string" maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:complexType>
</xs:schema>
```

Le schéma définit un type de données personnalisé nommé **MyListType**. Le type contient un élément imbriqué à occurrences multiples, nommé **élément**.

2. Définissez une variable à occurrence unique appelée **MyList** qui a le type de données **MyListType**.
3. Utilisez la variable comme cible d'une structure itérative.

Pour plus d'informations, consultez la section ["Présentation des Localisateurs, des Clés et de l'Indexation" à la page 364](#).

Chaque itération réutilise l'occurrence unique de **MyList**. Au début de l'itération, les éléments imbriqués **élément** sont détruits. Les ancrs à l'intérieur de la structure itérative, telles qu'un **RepeatingGroup** imbriqué, commencent à assigner les éléments **élément** à partir du début de la liste.

Exemple en ligne

Pour obtenir un exemple de la façon de détruire plusieurs occurrences d'une zone de stockage des données, consultez l'échantillon en ligne suivant :

`samples\Projects\ResetListVariable\ResetListVariable.cmw`

CHAPITRE 16

Ancre

Ce chapitre comprend les rubriques suivantes :

- [Présentation des ancrs, 205](#)
- [Mappage d'ancres Contenu aux zones de stockage des données, 206](#)
- [Définition des ancrs, 207](#)
- [Propriétés des ancrs standard, 209](#)
- [Comment un analyseur recherche-t-il les ancrs ?, 210](#)
- [Documentation de référence du composant Ancre, 217](#)
- [Référence du composant de recherche, 246](#)
- [Documentation de référence des sous-composants d'ancre, 250](#)

Présentation des ancrs

Les ancrs sont les composants qui permettent à un analyseur de se placer à des emplacements spécifiques d'un document source dans le but de trouver des données et de les stocker dans des zones de stockage de données. Une ancre est une balise que vous placez dans un document, indiquant la position des données.

Ce chapitre explique les différents types d'ancrs et leur utilisation dans des analyseurs.

Ancrs Marqueur et Contenu

Les ancrs les plus utilisées sont appelées **Marqueur** et **Contenu**. Ces ancrs sont souvent utilisées en paires :

- Une ancre **Marqueur** libelle un emplacement dans un document.
- Une ancre **Contenu** récupère un texte depuis l'emplacement.

Pour comprendre ces ancrs, imaginez un questionnaire imprimé. La première ligne demande généralement le nom et le prénom de la personne, chaque libellé étant suivi d'un espace vierge pour y placer l'information. Les libellés imprimés **Nom** et **Prénom** sont des ancrs **Marqueur** et les espaces blancs sont des ancrs **Contenu**. Les ancrs fournissent un moyen de localiser les données pour pouvoir les extraire du document source.

Autres types d'ancrs

En plus des ancrs **Marqueur** et **Contenu**, il existe de nombreux autres types d'ancrs que vous pouvez utiliser pour analyser les documents. Par exemple, les ancrs **Groupe** et **RepeatingGroup** vous aident à

spécifier l'organisation des champs de données. Une ancre **Alternatives** vous permet de spécifier plusieurs types de données pouvant apparaître à un emplacement particulier dans un document source.

Comment les ancres et les délimiteurs fonctionnent-ils ensemble ?

Vous pouvez définir les ancres du document de la source d'exemple. L'analyseur apprend comment analyser le document en examinant les ancres et les délimiteurs qui les séparent. Pour plus d'informations sur les délimiteurs, consultez la section ["Présentation des formats" à la page 178](#).

Supposons, par exemple, que vous ayez spécifié que votre document utilise un format délimité par des tabulations. Une ligne de la source d'exemple affiche

```
First name:<tab>Ron
```

Où `<tab>` est un caractère de tabulation.

Vous pouvez définir `Prénom` : en tant qu'ancre **Marqueur**. Vous pouvez définir `Ron` en tant qu'ancre **Contenu**. L'analyseur apprend de ces définitions qu'il doit rechercher dans un document source la chaîne `Prénom` :. Il doit alors sauter un délimiteur de tabulation simple et récupérer le texte qui suit la tabulation.

Supposons que vous exécutiez l'analyseur sur un autre document source, qui contient le texte suivant :

```
First name:<tab>Jack
```

L'analyseur trouve les ancres comme ci-dessus et extrait le texte `Jack`.

Supposons maintenant que le document source contienne :

```
First name:<tab>Jack<tab>Age:<tab>34
```

L'analyseur extrait toujours le texte `Jack` au lieu de `Jack<tab>Age<tab>34`. Cela fonctionne, car vous avez défini le caractère tabulation comme délimiteur. Le script comprend que l'ancre `Contenu` commence après la première tabulation et se termine avant la deuxième tabulation. Bien entendu, vous pouvez définir des ancres supplémentaires qui récupèrent l'âge de Jack, soit `34`.

Remarque: Les exemples ci-dessus décrivent un comportement possible des ancres et des délimiteurs. Les ancres ont plusieurs propriétés qui vous permettent de modifier ce comportement. Par exemple, vous pouvez définir une ancre **Contenu** qui ignore les tabulations, même dans un format délimité par des tabulations. Pour plus d'informations, consultez la section ["Comment un analyseur recherche-t-il les ancres ?" à la page 210](#)

Mappage d'ancres Contenu aux zones de stockage des données

Une ancre **Contenu** stocke le texte qu'elle extrait depuis un document source dans une zone de stockage des données. Vous pourriez configurer, par exemple, une ancre **Contenu** qui stocke son résultat dans un élément XML appelé **Prénom**. Si l'ancre **Contenu** extrait le texte `Jack`, l'analyseur produit la sortie suivante :

```
<FirstName>Jack</FirstName>
```

Plus précisément, vous pouvez spécifier que l'ancre doit stocker le texte extrait au chemin `/Personne/*s/`

`Prénom` qui fait référence à un élément défini dans le schéma XML. La sortie réelle de l'analyseur devrait être :

```
<Person>
  <FirstName>Jack</FirstName>
</Person>
```

D'un autre côté, imaginons que le schéma définisse **Prénom** en tant qu'attribut de l'élément **Personne**. Vous pourriez mapper l'ancre **Contenu** à `/Personne/@Prénom`. La sortie serait :

```
<Person FirstName="Jack" />
```

Vous devez mapper à une zone de stockage des données ayant un type de données approprié. Par exemple, ne mappez pas `Jack` à un élément XML qui a un type de données `xs:integer` ou à un élément XML ayant un type de données complexe contenant des éléments imbriqués. Pour plus d'informations sur cette règle, consultez la section ["Utiliser les types de données pour affiner les critères de recherche" à la page 215](#).

Mappage à des variables

Vous pouvez mapper une ancre à une zone de stockage des données qui est un élément XML, un attribut XML ou une variable. L'option de variable est utile si vous souhaitez utiliser les données dans une étape de traitement ultérieure, mais que vous ne voulez pas inclure les données brutes dans la sortie de l'analyseur.

Par exemple, supposons que vous vouliez extraire plusieurs nombres d'un document source et sortir leur somme dans le XML. Vous ne voulez pas de nombres individuels dans la sortie. Vous pouvez mapper les ancres `Contenu` qui récupèrent les nombres aux variables et utiliser une action `CalculateValue` pour calculer et sortir la somme.

Vous pourriez aussi mapper à une variable que vous utilisez dans une ancre ultérieure pour définir, par exemple, un texte dynamique de recherche d'une ancre `Marqueur`.

Mappage à des zones de stockage des données à occurrences multiples

Si vous mappez des ancres `Contenu` à une zone de stockage des données à occurrence unique, chaque affectation du zone de stockage des données écrase l'affectation précédente.

Si vous les mappez à une zone de stockage des données à occurrences multiples, chaque affectation génère une nouvelle occurrence du zone de stockage des données. Par exemple, si chaque ancre `Contenu` récupère un nom de personne, la sortie est une liste de noms :

```
<FirstName>Jack</FirstName>
<FirstName>Jennie</FirstName>
<FirstName>Larissa</FirstName>
```

Pour plus d'informations, consultez la section ["Zones de stockage des données à occurrences multiples" à la page 202](#).

Mappage à des éléments de contenu mixte

Le terme contenu mixte fait référence à un élément XML contenant à la fois des données de caractères et des éléments imbriqués. Si le schéma autorise un élément à avoir un contenu mixte, la vue Schéma affiche les zones de stockage des données *avant* et *après* pour les éléments. Cela vous permet de mapper une ancre `Contenu` à des données de caractères situées avant ou après un élément imbriqué spécifique. Pour plus d'informations, consultez la section ["Mappage de contenu mixte" à la page 195](#).

Définition des ancres

Lorsque vous définissez un composant `Analyseur`, vous devez ajouter une séquence d'ancres. L'analyseur fonctionne en cherchant les ancres dans le document source et en exécutant les opérations dont vous avez configuré l'exécution dans les ancres.

Où définir les ancres

Dans le script, les ancres sont imbriquées dans un **analyseur**.

Si vous appuyez sur **Entrée** à l'emplacement indiqué, l'éditeur IntelliScript affiche une liste déroulante qui inclut les ancres et les autres composants que vous pouvez ajouter.

Une fois que vous avez ajouté les ancres, l'outil Developer les met en surbrillance dans la source d'exemple.

Certains types d'ancres peuvent contenir des ancres imbriquées. Par exemple, vous pouvez imbriquer des ancres dans les ancres **Alternatives**, **Groupe** ou **RepeatingGroup**.

Séquence d'ancres

La séquence d'ancres doit être la séquence de texte dans le document source.

Supposons par exemple que le document source soit :

```
First Name: Ron  
Last Name: Lehrer
```

En admettant que vous définissiez **Prénom** et **Nom** en tant qu'ancres **Marqueur**, et que vous définissiez **Ron** et **Lehrer** en tant qu'ancres **Contenu**, la séquence d'ancres requise dans la configuration de l'analyseur est la suivante :

Ancre	Texte dans le document source
Marqueur	First Name
Contenu	Ron
Marqueur	Last Name
Contenu	Lehrer

Exception : Séquence source de variable

Certains documents source peuvent présenter une séquence variable. Supposons, par exemple, que le document source contienne l'un des formats suivants :

```
First Name: Ron  
Last Name: Lehrer
```

ou

```
Last Name: Lehrer  
First Name: Ron
```

Dans de tels cas, vous pouvez utiliser la propriété `marquage` pour modifier la zone de recherche des ancres. Pour plus d'informations, consultez la section ["Comment un analyseur recherche-t-il les ancres ?" à la page 210](#)

Ajout d'une ancre Marqueur ou Contenu

Vous pouvez ajouter des ancres **Marqueur** et **Contenu** à l'aide de la méthode select-and-click.

1. Sélectionnez le texte d'ancre dans le fichier exemple de source.

2. Cliquez sur le texte sélectionné avec le bouton droit, puis cliquez sur **Insérer Marqueur** ou **Insérer Contenu**.
3. Définir les propriétés d'ancrage.

Définition d'une ancre

Vous pouvez créer n'importe quel type d'ancre en modifiant le script. La procédure est identique à celle de l'édition de tout autre composant.

1. À l'emplacement de l'ancre de votre choix, sélectionnez l'ellipse (...), puis appuyez sur **ENTRÉE**.
2. Sélectionnez ou saisissez le nom de l'ancre.
3. Appuyez de nouveau sur **ENTRÉE** pour confirmer votre sélection.
4. Modifiez les propriétés de l'ancre.

Propriétés des ancrs standard

Le tableau suivant décrit les propriétés des ancrs standard :

Propriété	Description
direction	<p>Sens de recherche de l'ancre dans le champ de recherche. Vous pouvez choisir l'une des options suivantes :</p> <ul style="list-style-type: none"> - en arrière. Rechercher à partir de la fin du champ de recherche et trouver la dernière instance de l'ancre. - en avant. Rechercher depuis le début du champ de recherche et trouver la première instance de l'ancre. <p>Pour une ancre Marqueur, vous pouvez modifier ce comportement en utilisant la propriété count. Par exemple, si direction = backward et count = 2, le script trouve l'avant-dernière instance. La valeur par défaut est en avant. Pour plus d'informations, voir "Comment un analyseur recherche-t-il les ancrs ?" à la page 210</p>
disabled	<p>Détermine si le script ignore le composant et tous les composants enfants. Utilisez cette propriété pour tester, déboguer et modifier un script. Vous pouvez choisir l'une des options suivantes :</p> <ul style="list-style-type: none"> - Sélectionné. Le script ignore le composant. - Effacé. Le script applique le composant. <p>La valeur par défaut est vide.</p>
marquage	<p>Détermine si une ancre est utilisée comme début du champ de recherche pour l'ancre qui suit. Vous pouvez choisir l'une des options suivantes :</p> <ul style="list-style-type: none"> - position de départ. Placer un point de référence avant l'ancre actuelle. - position de fin. Placer un point de référence après l'ancre actuelle. - complet. Placer un point de référence avant et après l'ancre actuelle. - aucun. Ne pas créer de point de référence. <p>Pour plus d'informations, voir "Comment un analyseur recherche-t-il les ancrs ?" à la page 210</p>
nom	<p>Libellé descriptif pour le composant. Ce libellé s'affiche dans le fichier journal et la vue Événements. Utilisez la propriété nom pour identifier le composant ayant provoqué l'événement.</p>

Propriété	Description
no_initial_phase	Détermine si le script recherche des ancrs imbriquées dans la phase principale. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none"> - Effacé. Rechercher les ancrs imbriquées selon leurs propriétés individuelles. - Sélectionné. Rechercher les ancrs imbriquées dans la phase principale. Par défaut, la valeur est vide.
notifications	Liste de composants Notificationhandler qui gèrent les notifications de composants imbriqués. Pour plus d'informations, voir "Notifications" à la page 423
on_fail	Action à effectuer si le composant échoue. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none"> - Vide. N'effectuer aucune action. - CustomLog. Écrire dans le journal utilisateur. - LogError. Écrire un message d'erreur dans le journal du moteur. - LogInfo. Écrire un message d'information dans le journal du moteur. - LogWarning. Écrire un message d'avertissement dans le journal du moteur. - NotifyFailure. Envoyer une notification. Par défaut, la valeur est vide. Pour plus d'informations sur la gestion des défaillances de composants, consultez la section "Traitement des échecs" à la page 404 .
facultatif	Détermine si la défaillance d'un composant entraîne l'échec du composant parent. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none"> - Sélectionné. La défaillance du composant n'entraîne pas l'échec du composant parent. - Effacé. La défaillance du composant entraîne l'échec du composant parent. Par défaut, la valeur est vide. Pour plus d'informations sur l'échec des composants, consultez la section "Traitement des échecs" à la page 404 .
phase	Détermine les situations dans lesquelles le script traite le composant. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none"> - initiale. Le script traite le composant pendant la phase initiale. - principale. Le script traite le composant pendant la phase principale. - finale. Le script traite le composant pendant la phase finale. Pour plus d'informations, voir "Comment un analyseur recherche-t-il les ancrs ?" à la page 210
remarque	Commentaire défini par l'utilisateur, qui décrit l'objectif ou l'action du composant.

Quand il n'est pas certain qu'une ancre existe dans un document source, sélectionnez la propriété **optional**. Si aucune ancre n'existe, l'**Analyseur** dans lequel l'ancre est imbriquée continue.

Si l'ancre est imbriquée dans une ancre **de groupe**, la propriété **optional** empêche le **groupe** d'échouer. Si l'ancre appartient à un **RepeatingGroup**, la propriété empêche l'échec d'une itération du **RepeatingGroup**.

Comment un analyseur recherche-t-il les ancrs ?

Pour concevoir un analyseur correctement, il est important de comprendre la manière dont le script recherche les ancrs dans l'analyseur. Il y a trois concepts principaux :

- La phase de recherche
- Zone de recherche
- Critères de recherche

Cette section explique les concepts et la manière dont vous pouvez contrôler chacun d'eux en définissant les propriétés de l'ancre.

Phases de recherche

Le script recherche une séquence d'ancres en trois phases :

- Initial
- Principal
- Final

Toutes les ancres **Marqueur** sont par défaut dans la phase initiale et toutes les ancres **Contenu** sont dans la phase principale. Cela signifie que le script trouve d'abord les ancres `Marqueur`, puis il trouve les ancres `Contenu` parmi elles.

Pour comprendre ce concept, prenez par exemple un analyseur qui traite le document source suivant :

```
First name: Ron    Last name: Lehrer
```

Admettons que vous ayez défini les ancres de la façon suivante, avec les propriétés d'ancres par défaut :

Ancre	Texte dans le document source	Phase
Marqueur	Prénom :	Initial
Contenu	Ron	Principal
Marqueur	Nom :	Initial
Contenu	Lehrer	Principal

Dans la phase initiale, le script recherche les ancres `Marqueur` :

- Il recherche `Prénom :`
- Il recherche `Nom :` à l'emplacement qui suit `Prénom :`

Dans la phase principale, le script recherche les ancres `Contenu` :

- Il recherche l'ancre `Ron` dans un emplacement entre `Prénom :` et `Nom :`
- Il recherche l'ancre `Lehrer` dans un emplacement après `Nom :`

Phases imbriquées

Les ancres possédant des ancres imbriquées, comme l'ancre `Groupe`, ont des phases imbriquées. Par exemple, si une ancre `Groupe` s'exécute dans la phase principale de l'analyseur, une ancre `Marqueur` imbriquée dans le `Groupe` s'exécute dans une phase initiale imbriquée. La phase initiale imbriquée fait partie de la phase principale de l'analyseur, mais est située avant les autres ancres dans le `Groupe`.

Un autre exemple est donné par l'ancre `RepeatingGroup` qui recherche à la fois les séparateurs et les ancres imbriquées. Pour identifier correctement les ancres imbriquées, elle recherche les séparateurs avant de rechercher les ancres imbriquées.

Zone et critères de recherche

L'exemple ci-dessus des phases de recherche illustre les concepts de zone de recherche et les critères de recherche. La zone de recherche est la partie d'un document dans laquelle le script recherche une ancre. Les critères de recherche sont les règles en fonction desquelles le script trouve l'ancre dans la zone de recherche.

Dans la phase initiale, le script démarre la recherche pour l'ancre `Marqueur` contenant `Prénom :` au début du document. La zone de recherche pour cette ancre est le document entier. Le critère de recherche est que l'ancre doit contenir le texte `Prénom :`.

La zone de recherche pour l'ancre `Nom :` commence à la fin de `Prénom :` et s'étend jusqu'à la fin du document. Le critère de recherche est que l'ancre doit contenir le texte `Nom :`.

Dans la phase principale, l'analyseur interpole les ancres `Contenu` entre les ancres `Marqueur`. La zone de recherche de l'ancre `Ron` s'étend de la fin de l'ancre `Prénom :` jusqu'au début de l'ancre `Nom :`. En supposant que l'analyseur utilise un format délimité par des espaces, les critères de recherche doivent récupérer tout le texte présent dans la zone de recherche, après le premier caractère espace et avant le deuxième caractère espace.

La zone de recherche de l'ancre `Lehrer` va de la fin de `Nom :` jusqu'à la fin du document. Les critères de recherche sont similaires à ceux utilisés pour l'ancre `Ron`.

Nous pouvons ajouter cette analyse à la table d'ancrage que nous avons présentée plus haut. Le tableau décrit à présent la méthode complète utilisée par l'analyseur pour trouver les ancres.

Ancre	Texte dans le document source	Phase	Zone de recherche	Critères de recherche
Marqueur	First name:	Initial	Document entier	Texte = Prénom :
Contenu	Ron	Main	Fin de Prénom : jusqu'au début de Nom :	Après l'espace de tête Avant l'espace suivant
Marqueur	Last name:	Initial	Fin de Prénom : jusqu'à la fin du document	Texte = Nom :
Contenu	Lehrer	Main	Fin de Nom : jusqu'à la fin du document	Après l'espace de tête Avant l'espace suivant

Ajustement de la phase de recherche

En assignant la propriété **phase** d'une ancre, vous pouvez changer la phase dans laquelle le script recherche l'ancre.

Considérons le document source suivant :

```
CONTENT<10 characters>MARKER
```

Dans cet exemple, l'ancre **Marqueur** est située 10 caractères après l'ancre **Contenu**.

Par défaut, le script recherche le **Marqueur** dans sa phase initiale et le **Contenu** dans sa phase principale. Cela ne fonctionne pas ici, car le script ne trouve pas le **Marqueur** à moins d'avoir trouvé le **Contenu** !

La solution est de modifier la propriété **phase** de l'une des ancres. Vous pouvez modifier le **Contenu** vers la phase initiale ou le **Marqueur** vers la phase principale. Dans les deux cas, le script trouve les ancres.

Ajustement de la zone de recherche

Il y a deux façons d'ajuster la zone de recherche pour une ancre :

- En définissant la propriété `phase` de l'ancre ou des ancres environnantes.
- En définissant la propriété `marquage` des ancres environnantes.

Propriété de phase

Si une ancre `Contenu` réside entre deux ancres `Marqueur`, la zone de recherche pour le `Contenu` correspond alors, par défaut, au segment compris entre les ancres `Marqueur`.

Si vous modifiez toutes les ancres dans la même phase, la zone de recherche du `Contenu` n'est plus limitée par le deuxième `Marqueur` : elle sera comprise entre la fin du premier `Marqueur` et la fin du document.

Prenez, par exemple, le document source suivant :

```
Tree Fig      Date<tab>October 27, 2003 (pruned)
Tree Date Palm Date April 27, 2003<tab>(planted)
```

Cet exemple suppose que le document source a une structure lâche, contenant un nombre variable d'espaces, de tabulations ou d'autres symboles intercalés dans le texte. Nous ne pouvons donc pas utiliser facilement les espaces et les tabulations comme délimiteurs. Un exemple comme celui-ci peut se produire lors de l'analyse de documents issus de logiciels de traitement de texte.

Nous pouvons analyser ce document en utilisant une ancre `RepeatingGroup` contenant les ancres imbriquées `Marqueur` et `Contenu`. Les ancres `Marqueur` sont les chaînes `Arbre` et `Date`. Les ancres `Contenu` correspondent à tout ce qui se trouve entre les ancres `Marqueur`, y compris les espaces et les tabulations.

Le problème dans l'analyse de ce document réside dans la seconde itération du `RepeatingGroup` qui analyse la deuxième ligne. Si nous laissons les ancres `Marqueur` dans la phase initiale, le script considère à tort la première instance du mot `Date` comme étant un `Marqueur`. Dans la phase principale, il ne réussit pas à trouver `Date Palm` parce que la zone de recherche est située entre les deux ancres `Marqueur` et qu'il n'y a pas de texte entre elles.

Une solution envisageable est de déplacer le `Marqueur` depuis `Date` vers la phase principale et de définir l'ancre `Contenu`, `Date Palm` utilisant une expression qui recherche un nom d'arborescence d'un ou deux mots. Dans la phase initiale du `RepeatingGroup`, le script trouve le `Marqueur` pour `Arbre`. Dans la phase principale, il trouve `Date Palm` suivi du `Marqueur` pour `Date`.

Avec le nouveau paramétrage de phase, nous avons modifié l'étendue de la recherche de nom d'arborescence. La portée s'étend maintenant depuis `Arbre` jusqu'à la fin de l'itération et le script trouve `Date Palm` avec succès.

Propriété de marquage

Prenez par exemple la structure de document source suivante :

```
MARKER
%%CONTENT A
^^^CONTENT B
```

Supposons que la séquence de `Contenu A` et `Contenu B` varie entre les documents source. Dans certains documents, `Contenu B` précède `Contenu A`.

Dans ce cas, les critères de recherche sont :

- `Contenu A` et `Contenu B` suivent tous deux l'ancre `Marqueur`
- `Contenu A` commence par `%%` et `Contenu B` commence par `^^^`.

Par défaut, la zone de recherche pour `Contenu A` s'étend de la fin du `Marqueur` à la fin du document. La zone de recherche pour `Contenu B` s'étend de la fin de `Contenu A` à la fin du document. Ceci ne fonctionne pas, car, dans certains documents source, `Contenu A` et `Contenu B` sont inversés.

La solution consiste à modifier la zone de recherche pour `Contenu B`. Vous pouvez effectuer cela en définissant la propriété `marquage` de `Contenu A`. La propriété `marquage` spécifie l'endroit où le script place les points de référence qui déterminent le début et la fin de la zone de recherche.

Le paramètre par défaut est `marquage = complet`, ce qui signifie que le script place des points de référence avant et après chaque ancre. La zone de recherche pour `Contenu B` débute au dernier point de référence, qui est celui suivant `Contenu A`. Ceci mène à une analyse incorrecte, comme nous l'avons vu.

Pour empêcher le script de placer des points de référence autour de `Contenu A`, définissez la propriété `marquage` de `Contenu A` sur `Aucun`. En résultat, la zone de recherche pour `Contenu B` débute à la fin du Marqueur. Cela permet au script de trouver `Contenu B`, même si celui-ci précède `Contenu A`.

Le tableau suivant décrit les quatre valeurs possibles de la propriété `Marquage`. La colonne `Résultat` part du principe que, dans l'exemple précédent, vous avez affecté la valeur `marquage` à `Contenu A`.

Propriété de marquage	Explication	Résultat
complet	Le script place des marques de référence au début et à la fin de l'ancre actuelle. C'est le comportement par défaut.	Le script recherche l'ancre suivante après la fin de l'ancre actuelle. <code>Contenu B</code> suit <code>Contenu A</code> .
position de départ	Le script place uniquement une marque de référence au début de l'ancre actuelle.	Le script recherche l'ancre suivante après la début de l'ancre actuelle. <code>Contenu B</code> chevauche ou suit <code>Contenu A</code> .
position de fin	Le script place uniquement une marque de référence à la fin de l'ancre actuelle.	Le script recherche l'ancre suivante après la fin de l'ancre actuelle. <code>Contenu B</code> suit <code>Contenu A</code> .
aucun	Le script ne place pas de marque de référence pour l'ancre actuelle.	Le script recherche l'ancre suivante après la fin de l'ancre précédente. <code>Contenu B</code> suit Marqueur, sans tenir compte de <code>ContenuA</code> .

Remarque: Il existe quelques situations dans lesquelles vous devez utiliser une ancre qui marque un point de référence. Un exemple est le séparateur d'un `RepeatingGroup`. Si le séparateur ne marque pas, il ne fait rien. Un avertissement apparaît si vous essayez d'utiliser une ancre ne marquant pas dans un emplacement où le marquage est requis.

Exemples en ligne

Pour obtenir un exemple en ligne de la propriété `marquage`, ouvrez le projet `samples\Projects\Marking_Mode\Marking_Mode.cmw`. L'exemple utilise la propriété pour altérer la zone de recherche d'une ancre `Contenu`.

Pour un autre exemple, consultez `samples\Projects\NonMarker\NonMarker.cmw`. Cet exemple utilise l'option `marquage = none` permettant à deux ancres `Contenu` de se chevaucher. L'exemple illustre aussi l'utilisation de `direction = en arrière` pour rechercher à partir de la fin de la zone de recherche.

Ajustement des critères de recherche

Le script peut rechercher des ancres selon un grand nombre de critères de recherche, par exemple :

- Selon l'emplacement du délimiteur dont le script prend connaissance depuis la source d'exemple
- Selon un décalage de position, autrement dit le nombre de caractères depuis un point de référence
- En recherchant un texte particulier
- En recherchant une forme ou une expression régulière
- En recherchant un type de données spécifié
- En recherchant une valeur d'attribut

Vous pouvez combiner ces critères de recherche d'à peu près toutes les façons. Par exemple, vous pouvez spécifier qu'une ancre `Contenu` commence deux tabulations après une ancre `Marqueur` et qu'elle possède 10 caractères. Si vous faites cela, vous utilisez un critère délimiteur pour définir le début de l'ancre `Contenu` et un critère de décalage pour définir la fin.

Les composants qui effectuent ces recherches sont appelés des composants chercheurs. Pour plus d'informations, consultez la section ["Référence du composant de recherche" à la page 246](#).

Utiliser les types de données pour affiner les critères de recherche

Par défaut, en plus des autres critères de recherche, le script recherche une ancre `Contenu` selon le type de données de sa zone de stockage des données.

Par exemple, supposons que le champ de recherche d'une ancre `Contenu` soit la chaîne suivante :

```
The students' grades were 81, 56, and 95, respectively.
```

Et admettons que vous ne définissiez aucun autre critère de recherche pour l'ancre. Si vous mappez l'ancre à une zone de stockage des données de type `xs:string`, l'ancre récupère la chaîne entière.

Si la zone de stockage des données est de type `xs:integer`, le script recherche la première sous-chaîne correspondant au type de données. En considérant que vous configurez l'ancre avec `direction = forward`, l'ancre récupère l'entier 81. Si `direction = backward`, l'ancre récupère 95.

Supposons maintenant que la zone de stockage des données soit de type `xs:integer`, et que le schéma limite les zones de stockage des données à des valeurs inférieure à 60. Le script recherche un nombre entier qui soit conforme à la restriction et récupère 56.

Types de données combinés avec d'autres critères de recherche

Vous pouvez combiner un critère de type de données avec d'autres critères de recherche. Dans l'exemple ci-dessus, admettons que vous configuriez l'ancre `Contenu` pour rechercher l'expression régulière :

```
[",. *,"]
```

L'expression cherche deux virgules séparées par n'importe quel caractère autre qu'un retour à la ligne. La recherche trouve la sous-chaîne

```
, 56,
```

Si le type de la zone de stockage des données est `xs:integer`, l'ancre récupère 56.

Types de données de liste

une zone de stockage des données peut être une liste séparée par des espaces. Le script filtre le texte récupéré par l'ancre `Contenu` afin qu'il corresponde aux types des éléments de la liste.

Supposons, par exemple, que le schéma définisse un attribut appelé `grades` qui est une liste d'éléments `xs:integer`. Dans l'exemple ci-dessus, si vous mappez l'ancre `Contenu` à `grades`, l'ancre renvoie une liste de nombres entiers dans la chaîne :

```
81 56 95
```

Si l'attribut `grades` appartient à un élément nommé `Étudiants`, la sortie XML est :

```
<Students grades="81 56 95" />
```

Si vous définissez l'ancre `Contenu` avec `direction = en arrière`, la liste est inversée :

```
<Students grades="95 56 81" />
```

Type décimal

Si une zone de stockage des données a le type `xs:decimal`, le script considère que le séparateur décimal est un point. Si votre paramètre régional utilise une virgule comme séparateur décimal, une recherche `xs:decimal` peut échouer.

Recherche de type avec marqueur de fermeture

Si une ancre `Contenu` comprend une propriété `closing_marker` mais pas `opening_marker`, le script renvoie la sous-chaîne la plus proche de `closing_marker` qui correspond au type de la zone de stockage des données.

Dans l'exemple ci-dessus, si vous définissez le mot `respectively` comme `closing_marker`, et que la zone de stockage des données est de type `xs:integer`, l'ancre récupère 95.

Exemple en ligne

Pour consulter un exemple en ligne de la recherche par type de données, ouvrez le projet `samples\Projects\Pattern\Pattern.cmw`. L'exemple est un analyseur contenant une seule ancre `Contenu` mappée à un élément XML. Le schéma utilise `xs:pattern` pour restreindre l'élément à certaines séquences de caractères. L'ancre extrait la portion du document source qui correspond au modèle.

Désactivation de la recherche Donnée-Type

Vous pouvez désactiver la recherche type de données en sélectionnant la propriété **`disable_XSD_type_search`** de l'ancre **Contenu**. Si vous effectuez cela, l'ancre recherche en fonction des autres critères, sans tenir compte du type de la zone de stockage des données.

Si le résultat n'a pas le bon type, il ne peut pas être stocké dans la zone de stockage des données et l'ancre échoue. Vous pouvez utiliser des transformateurs pour convertir le résultat dans le bon type et empêcher l'échec. Pour plus d'informations, consultez la section ["Présentation des transformateurs" à la page 259](#).

Par exemple, imaginons que le document source contienne une date au format `dd-mm-yyyy` et que vous souhaitiez stocker la date dans une zone de stockage des données `xs:date`. Vous pouvez gérer cette situation de la façon suivante :

1. Définissez une ancre **Contenu** qui récupère des données `dd-mm-yyyy`, en ignorant la discordance avec le type `xs:date`.
2. Configurez l'ancre avec le transformateur **DateFormatICU** qui convertit le résultat en `xs:date`.

Ancre contenant des ancrs imbriquées

Un point intéressant est la méthode utilisée par un analyseur pour rechercher une ancre possédant des ancrs imbriquées, telle qu'une ancre `Groupe`.

Le script ne recherche pas un `Groupe`, puis les ancrs imbriquées. Il recherche plutôt les ancrs imbriquées. La portée du `Groupe` est définie par les ancrs imbriquées que le script trouve.

Par exemple, supposons qu'un analyseur contienne la séquence d'ancres suivante. Nous partons du principe que les ancrs ont les propriétés `phase`, `marquage` et `facultatif` par défaut.

```
Marker A
Group
  Marker B
  Content C
  Marker D
Marker E
```


Le script recherche d'abord **Marqueur A** et **Marqueur E**. La portée de la recherche du Groupe est la région comprise entre **Marqueur A** et **Marqueur E**.

Ensuite, à l'intérieur de la zone de recherche du Groupe, le script recherche **Marqueur B** et **Marqueur D**. La zone située entre ces ancres **Marqueur** représente la zone de recherche de **Contenu C**.

Dans cette dernière zone de recherche, le script recherche **Contenu C**.

Documentation de référence du composant Ancre

Les composants **Ancre** indiquent les emplacements des données dans les documents source.

Alternatives

L'ancre **Alternatives** définit un jeu d'ancres alternatives imbriquées. Vous pouvez définir un critère pour la sélection d'une alternative dans le jeu.

Le tableau suivant décrit les propriétés de l'ancre **Alternatives** :

Propriété	Description
désactivé	Détermine si le script ignore le composant et tous les composants enfants. Utilisez cette propriété pour tester, déboguer et modifier un script. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none">- Sélectionné. Le script ignore le composant.- Effacé. Le script applique le composant. La valeur par défaut est vide.
marquage	Détermine si une ancre est utilisée comme début du champ de recherche pour l'ancre qui suit. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none">- position de départ. Placer un point de référence avant l'ancre actuelle.- position de fin. Placer un point de référence après l'ancre actuelle.- complet. Placer un point de référence avant et après l'ancre actuelle.- aucun. Ne pas créer de point de référence. Pour plus d'informations, voir "Comment un analyseur recherche-t-il les ancres ?" à la page 210
nom	Libellé descriptif pour le composant. Ce libellé s'affiche dans le fichier journal et la vue Événements . Utilisez la propriété nom pour identifier le composant ayant provoqué l'événement.
on_fail	Action à effectuer si le composant échoue. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none">- Vide. N'effectuer aucune action.- CustomLog. Écrire dans le journal utilisateur.- LogError. Écrire un message d'erreur dans le journal du moteur.- LogInfo. Écrire un message d'information dans le journal du moteur.- LogWarning. Écrire un message d'avertissement dans le journal du moteur.- NotifyFailure. Envoyer une notification. Par défaut, la valeur est vide. Pour plus d'informations sur la gestion des défaillances de composants, consultez la section "Traitement des échecs" à la page 404 .
facultatif	Détermine si la défaillance d'un composant entraîne l'échec du composant parent. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none">- Sélectionné. La défaillance du composant n'entraîne pas l'échec du composant parent.- Effacé. La défaillance du composant entraîne l'échec du composant parent. Par défaut, la valeur est vide. Pour plus d'informations sur l'échec des composants, consultez la section "Traitement des échecs" à la page 404 .

Propriété	Description
phase	<p>Détermine les situations dans lesquelles le script traite le composant. Vous pouvez choisir l'une des options suivantes :</p> <ul style="list-style-type: none"> - initiale. Le script traite le composant pendant la phase initiale. - principale. Le script traite le composant pendant la phase principale. - finale. Le script traite le composant pendant la phase finale. <p>Pour plus d'informations, voir "Comment un analyseur recherche-t-il les ancrés ?" à la page 210 La valeur par défaut est Principale.</p>
remarque	Commentaire défini par l'utilisateur, qui décrit l'objectif ou l'action du composant.
sélecteur	<p>Détermine le critère de sélection d'une ancre parmi les ancrés imbriqués sous l'ancre Alternatives. Vous pouvez choisir l'une des options suivantes :</p> <ul style="list-style-type: none"> - ScriptOrder. L'analyseur teste les ancrés imbriqués dans l'ordre défini dans le script. Il accepte la première ancre imbriquée qui réussit. Si toutes les ancrés imbriqués échouent, l'ancre Alternatives échoue. - DocumentOrder. L'analyseur teste toutes les ancrés imbriqués. Il accepte soit la première soit la dernière ancre réussie, en fonction des emplacements des ancrés dans le document source, comme déterminé par la propriété sélectionner. Si toutes les ancrés imbriqués échouent, l'ancre Alternatives échoue. - NameSwitch. L'analyseur recherche l'ancre dont la propriété nom est spécifiée dans la zone de stockage des données définie dans option_name. Il ignore les autres ancrés. Si l'ancre imbriquée nommée échoue, l'ancre Alternatives échoue.

Exemple

Imaginez que vous analysiez un document dans lequel la date peut apparaître dans l'un des modèles suivants :

```
21/10/03
October 21, 2003
```

Pour traiter ce contenu, vous devez définir une ancre `Alternatives` qui contient deux ancrés `Contenu`, stockant leur sortie dans des éléments XML différents. Chaque élément XML est contraint d'accepter l'un des modèles de date. L'ancre `Alternatives` est configurée avec `selector = ScriptOrder`.

Lorsque l'analyseur exécute l'ancre `Alternatives`, il teste la première ancre `Contenu`. Si la date correspond au modèle de la première ancre, l'ancre `Contenu` réussit. Si la date ne correspond pas au modèle, la première ancre `Contenu` échoue et l'ancre `Alternatives` teste la seconde ancre `Contenu`. De cette façon, l'analyseur peut traiter les deux modèles de date.

Comment définir une ancre Alternatives

Ajoutez une ancre **Alternatives** en modifiant le script. Imbriguez dans l'ancre **Alternatives** les ancrés alternatives.

Utiliser les alternatives pour sélectionner un analyseur secondaire

Vous pouvez utiliser l'ancre `Alternatives` pour contrôler quel analyseur secondaire, parmi plusieurs, doit traiter un document. L'analyseur principal peut utiliser cette fonctionnalité pour traiter des documents sources de plusieurs types.

Par exemple, supposons que la page d'accueil du site Web d'un journal possède des liens vers des articles. Après chaque lien, l'article est libellé **Actualités**, **Affaires** ou **Sports**. Vous voulez analyser les articles, en utilisant un analyseur différent pour chaque type, comme ceci :

```
<a href="PrincessWeds.html">Norwegian Princess Weds</a> - News  
<a href="BanksMerge.html">Local Banks to Merge</a> - Business  
<a href="HomeTeamWins.html">Bears Trounce Antelopes</a> - Sports
```

Vous pouvez prendre en charge cette situation de la manière suivante :

1. L'analyseur principal récupère le nom de fichier d'un article et le stocke dans une variable.
2. L'analyseur principal contient une ancre **Alternatives** configurée avec l'option **DocumentOrder**.
3. L'ancre **Alternatives** contient des ancres **Groupe** imbriquées.
4. Chaque ancre **Groupe** est configurée avec une ancre **Marqueur** et une action **RunParser**, comme suit :
 - Le premier groupe contient un marqueur qui recherche la chaîne **Actualités**. Le Groupe est configuré avec une action **RunParser** qui exécute un analyseur secondaire appelé **NewsParser**.
 - Le deuxième groupe contient un marqueur qui recherche **Affaires** et exécute **BusinessParser**.
 - Le troisième groupe contient un marqueur qui recherche **Sports** et exécute **SportsParser**.

L'ancre **Alternatives** teste les trois ancres **Groupe**. Elle accepte le groupe contenant le premier marqueur apparaissant après le nom de fichier. Le Groupe exécute l'analyseur approprié sur le fichier.

Exemple en ligne

Pour un échantillon en ligne de cette ancre, ouvrez le projet `samples\Projects\Alternatives\Alternatives.cmw`. L'exemple utilise des ancres **Alternatives** pour analyser différents formats de nom et de date pouvant exister dans un document source.

Contenu

Une ancre **Contenu** récupère le texte d'un document source. L'analyseur effectue la recherche dans une région définie conformément aux critères de recherche spécifiés et stocke les textes récupérés dans une zone de stockage des données.

Le tableau suivant décrit les propriétés de l'ancre **Contenu** :

Propriété	Description
<code>allow_empty_values</code>	Détermine si l'ancre Contenu peut être vide. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none">- Sélectionné. Une valeur vide est affectée au data_holder.- Effacé. Les valeurs vides ne sont pas autorisées. allow_empty_values doit être sélectionné dans les cas suivants : <ul style="list-style-type: none">- Lorsque l'ancre est configurée avec valeur = LearnByExample et qu'il n'y a rien entre les délimiteurs.- Quand il y a rien entre le opening_marker et closing_marker.
<code>closing_marker</code>	Définit la fin d'une région dans laquelle l'analyseur recherche l'ancre Contenu . Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none">- NewlineSearch. La fin de l'ancre Contenu est le caractère de retour à la ligne.- OffsetSearch. La fin de l'ancre Contenu est le nombre de caractères spécifié dans décalage.- PatternSearch. La fin de l'ancre Contenu ancre est le premier texte correspondant à une expression régulière spécifiée.- TextSearch. La fin de l'ancre Contenu est une chaîne de texte spécifiée.

Propriété	Description
data_holder	Définit une zone de stockage des données où l'ancre Contenu stocke les textes récupérés.
direction	<p>Sens de recherche de l'ancre dans le champ de recherche. Vous pouvez choisir l'une des options suivantes :</p> <ul style="list-style-type: none"> - en arrière. Rechercher à partir de la fin du champ de recherche et trouver la dernière instance de l'ancre. - en avant. Rechercher depuis le début du champ de recherche et trouver la première instance de l'ancre. <p>Pour une ancre Marqueur, vous pouvez modifier ce comportement en utilisant la propriété count. Par exemple, si direction = backward et count = 2, le script trouve l'avant-dernière instance.</p> <p>La valeur par défaut est en avant. Pour plus d'informations, voir "Comment un analyseur recherche-t-il les ancres ?" à la page 210</p>
disable_XSD_type_search	<p>Détermine si l'analyseur recherche du contenu correspondant au type de données de la zone de stockage des données. Vous pouvez choisir l'une des options suivantes :</p> <ul style="list-style-type: none"> - Sélectionné. L'analyseur effectue la recherche sans tenir compte du type de données. Après avoir appliqué les transformateurs au contenu, si le résultat ne correspond pas au type de données du détenteur de données, l'ancre échoue. - Effacé. L'analyseur recherche du contenu correspondant au type de données. <p>La valeur par défaut est Effacé. Pour plus d'informations, voir "Utiliser les types de données pour affiner les critères de recherche" à la page 215</p>
désactivé	<p>Détermine si le script ignore le composant et tous les composants enfants. Utilisez cette propriété pour tester, déboguer et modifier un script. Vous pouvez choisir l'une des options suivantes :</p> <ul style="list-style-type: none"> - Sélectionné. Le script ignore le composant. - Effacé. Le script applique le composant. <p>La valeur par défaut est vide.</p>
ignore_default_transformers	<p>Détermine si l'analyseur applique les transformateurs par défaut au contenu. La valeur par défaut est Effacé.</p> <p>Pour plus d'informations, voir "Présentation des transformateurs" à la page 259</p>
marquage	<p>Détermine si une ancre est utilisée comme début du champ de recherche pour l'ancre qui suit. Vous pouvez choisir l'une des options suivantes :</p> <ul style="list-style-type: none"> - position de départ. Placer un point de référence avant l'ancre actuelle. - position de fin. Placer un point de référence après l'ancre actuelle. - complet. Placer un point de référence avant et après l'ancre actuelle. - aucun. Ne pas créer de point de référence. <p>Pour plus d'informations, voir "Comment un analyseur recherche-t-il les ancres ?" à la page 210</p>
nom	Libellé descriptif pour le composant. Ce libellé s'affiche dans le fichier journal et la vue Événements . Utilisez la propriété nom pour identifier le composant ayant provoqué l'événement.

Propriété	Description
on_fail	<p>Action à effectuer si le composant échoue. Vous pouvez choisir l'une des options suivantes :</p> <ul style="list-style-type: none"> - Vide. N'effectuer aucune action. - CustomLog. Écrire dans le journal utilisateur. - LogError. Écrire un message d'erreur dans le journal du moteur. - LogInfo. Écrire un message d'information dans le journal du moteur. - LogWarning. Écrire un message d'avertissement dans le journal du moteur. - NotifyFailure. Envoyer une notification. <p>Par défaut, la valeur est vide. Pour plus d'informations sur la gestion des défaillances de composants, consultez la section "Traitement des échecs" à la page 404.</p>
opening_marker	<p>Définit le début d'une région dans laquelle l'analyseur recherche l'ancre Contenu. Les valeurs possibles sont les composants suivants :</p> <ul style="list-style-type: none"> - NewlineSearch. Le début de l'ancre Contenu est le caractère de retour à la ligne. - OffsetSearch. Le début de l'ancre Contenu est le nombre de caractères spécifié dans décalage. - PatternSearch. Le début de l'ancre Contenu est le premier texte indiqué correspondant à une expression régulière. - TextSearch. Le début de l'ancre Contenu est une chaîne de texte spécifiée.
facultatif	<p>Détermine si la défaillance d'un composant entraîne l'échec du composant parent. Vous pouvez choisir l'une des options suivantes :</p> <ul style="list-style-type: none"> - Sélectionné. La défaillance du composant n'entraîne pas l'échec du composant parent. - Effacé. La défaillance du composant entraîne l'échec du composant parent. <p>Par défaut, la valeur est vide. Pour plus d'informations sur l'échec des composants, consultez la section "Traitement des échecs" à la page 404.</p>
phase	<p>Détermine les situations dans lesquelles le script traite le composant. Vous pouvez choisir l'une des options suivantes :</p> <ul style="list-style-type: none"> - initiale. Le script traite le composant pendant la phase initiale. - principale. Le script traite le composant pendant la phase principale. - finale. Le script traite le composant pendant la phase finale. <p>Pour plus d'informations, voir "Comment un analyseur recherche-t-il les ancres ?" à la page 210 La valeur par défaut est Principale.</p>
remarque	<p>Commentaire défini par l'utilisateur, qui décrit l'objectif ou l'action du composant.</p>
transformateurs	<p>Définit une séquence de transformateurs que l'analyseur applique au texte récupéré. Pour plus d'informations : Chapitre 17, "Transformateurs" à la page 259</p>

Propriété	Description
validateurs	Définit une liste de validateurs appliqués aux données. Pour plus d'informations, voir "Validateurs" à la page 407
valeur	<p>Définit les critères pour une recherche dans la région définie par les attributs closing_marker et closing_marker. Si opening_marker n'est pas défini, la recherche se fait entre les points de référence alentour. Pour plus d'informations, voir "Comment un analyseur recherche-t-il les ancres ?" à la page 210 Vous pouvez choisir l'une des options suivantes :</p> <ul style="list-style-type: none"> - Vide. L'ancre Contenu récupère la totalité de l'étendue de la recherche. - AttributeSearch. L'ancre Contenu récupère la valeur d'une expression du type AttributeName=.... Utilisez cette option pour récupérer les valeurs d'attribut d'un document XML ou HTML source. - LearnByExample. L'analyseur apprend quel est le texte à récupérer selon le format de l'analyseur et l'exemple de source. Par exemple, si l'analyseur a un format délimité par des tabulations, il compte le nombre de tabulations depuis le début de la zone de recherche de l'exemple de texte. Il récupère le texte entre les tabulations correspondantes dans le document source. - PatternSearch. L'ancre Contenu extrait le premier texte qui correspond à une expression régulière spécifiée. - TypeSearch. L'ancre Contenu extrait le premier texte qui correspond à un type de données spécifié. <p>La valeur par défaut est Vide. Pour plus d'informations sur ces options, consultez le "Référence du composant de recherche" à la page 246. Outre les composants de recherche, l'analyseur utilise le type de données du data_holder comme critère de recherche. Pour plus d'informations, voir "Utiliser les types de données pour affiner les critères de recherche" à la page 215</p>

Les propriétés **opening_marker** et **closing_marker** sont équivalentes aux ancres **Marqueur** d'un composant **Groupe**.

- Une ancre **Contenu** avec **opening_marker** défini fonctionne comme un composant **Groupe** avec la séquence d'ancres suivante :
 1. Marqueur
 2. Contenu
- Une ancre **Contenu** avec **closing_marker** défini fonctionne comme un composant **Groupe** avec la séquence d'ancres suivante :
 1. Contenu
 2. Marqueur
- Une ancre **Contenu** avec **opening_marker** et **closing_marker** définis fonctionne comme un composant **Groupe** avec la séquence d'ancres suivante :
 1. Marqueur
 2. Contenu
 3. Marqueur

Pour plus d'informations, consultez le ["Référence du composant de recherche" à la page 246](#).

Direction de recherche

La propriété `direction` a plusieurs effets dans une ancre `Contenu`. Si `direction` = en arrière :

- Le script `recherche` `opening_marker` et `closing_marker`, à rebours, en commençant par la fin de la zone de recherche. `Opening_marker` continue de précéder `closing_marker`.

- Le composant chercheur recherche en sens inverse en partant de la fin de la zone de recherche.
- Si le composant chercheur est `LearnByExample`, il compte les délimiteurs à rebours en partant de la fin de la zone de recherche.

Exemple en ligne

Pour obtenir un échantillon en ligne des ancrs `Contenu`, ouvrez le projet `samples\Projects\Content\Content.cmw`. L'échantillon illustre plusieurs utilisations des propriétés `opening_marker`, `closing_marker` et `valeur` pour configurer les ancrs `Contenu`

DelimitedSections

L'ancr **DelimitedSections** analyse les données qui sont divisées en sections par un séparateur. Elle définit un groupe d'ancres imbriquées. Chaque ancre imbriquée analyse une seule section.

Le tableau suivant décrit les propriétés de l'ancr **DelimitedSections** :

Propriété	Description
désactivé	Détermine si le script ignore le composant et tous les composants enfants. Utilisez cette propriété pour tester, déboguer et modifier un script. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none"> - Sélectionné. Le script ignore le composant. - Effacé. Le script applique le composant. La valeur par défaut est vide.
marquage	Détermine si une ancre est utilisée comme début du champ de recherche pour l'ancr qui suit. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none"> - position de départ. Placer un point de référence avant l'ancr actuelle. - position de fin. Placer un point de référence après l'ancr actuelle. - complet. Placer un point de référence avant et après l'ancr actuelle. - aucun. Ne pas créer de point de référence. Pour plus d'informations, voir "Comment un analyseur recherche-t-il les ancrs ?" à la page 210
nom	Libellé descriptif pour le composant. Ce libellé s'affiche dans le fichier journal et la vue Événements . Utilisez la propriété nom pour identifier le composant ayant provoqué l'événement.
notifications	Liste de composants Notificationhandler qui gèrent les notifications de composants imbriqués. Pour plus d'informations, voir "Notifications" à la page 423
on_fail	Action à effectuer si le composant échoue. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none"> - Vide. N'effectuer aucune action. - CustomLog. Écrire dans le journal utilisateur. - LogError. Écrire un message d'erreur dans le journal du moteur. - LogInfo. Écrire un message d'information dans le journal du moteur. - LogWarning. Écrire un message d'avertissement dans le journal du moteur. - NotifyFailure. Envoyer une notification. Par défaut, la valeur est vide. Pour plus d'informations sur la gestion des défaillances de composants, consultez la section "Traitement des échecs" à la page 404 .
facultatif	Détermine si la défaillance d'un composant entraîne l'échec du composant parent. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none"> - Sélectionné. La défaillance du composant n'entraîne pas l'échec du composant parent. - Effacé. La défaillance du composant entraîne l'échec du composant parent. Par défaut, la valeur est vide. Pour plus d'informations sur l'échec des composants, consultez la section "Traitement des échecs" à la page 404 .

Propriété	Description
phase	Détermine les situations dans lesquelles le script traite le composant. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none"> - initiale. Le script traite le composant pendant la phase initiale. - principale. Le script traite le composant pendant la phase principale. - finale. Le script traite le composant pendant la phase finale. Pour plus d'informations, voir "Comment un analyseur recherche-t-il les ancrés ?" à la page 210 La valeur par défaut est Principale.
remarque	Commentaire défini par l'utilisateur, qui décrit l'objectif ou l'action du composant.
séparateur	Définit une ancre qui délimite les sections.
separator_position	Définit le positionnement du séparateur par rapport aux sections. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none"> - après. Il y a un séparateur après chaque section, y compris la dernière section. Par exemple, 1 2 3 4 - autour. Il y a des séparateurs avant et après chaque section, y compris la première et la dernière section. Par exemple, 1 2 3 4 - avant. Il y a un séparateur avant chaque section, y compris la première section. Par exemple, 1 2 3 4 - entre. Il y a un séparateur entre les sections consécutives, mais ni avant la première section, ni après la dernière. Par exemple, 1 2 3 4
using_placeholders	Détermine si l'ancre DelimitedSections recherche le séparateur d'une section facultative manquante dans le document source. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none"> - toujours. Le séparateur d'une section manquante existe toujours. Par exemple, 1 3 - jamais. Le séparateur d'une section manquante n'existe jamais. Par exemple, 1 3 - si nécessaire. Le séparateur d'une section interne manquante existe toujours. Le séparateur d'une section finale manquante n'existe jamais. Par exemple, 1 3 Dans ces exemples, separator_position est défini à avant et les sections 2 et 4 sont manquantes.

Exemple

Le curriculum vitae d'un employé contient plusieurs sections, chacune d'entre elles étant précédée d'une ligne de tirets :

```

-----
Jane Palmer
Employee ID 123456
-----
Professional Experience
...
-----
Education
...

```

Vous pouvez définir la région sectionnée comme une ancre `DelimitedSections`, avec la ligne de tirets en tant que `séparateur`. Parce que la ligne de tirets précède chaque section, vous devez définir `separator_position` sur `avant`.

Dans l'ancre `DelimitedSections`, imbriquez trois ancres `Groupe`. Le premier `Groupe` analyse la section Jane Palmer, le second `Groupe` analyse `Professional Experience` et ainsi de suite.

Sections facultatives

Dans l'exemple ci-dessus, imaginez que la seconde section, `Expérience professionnelle`, soit manquante dans certains des documents source. Son séparateur, la ligne de tirets, est toujours présent.

```
-----  
Jane Palmer  
Employee ID 123456  
-----  
-----  
Education  
...
```

Pour gérer cette situation, configurez `DelimitedSections` de la manière suivante :

- Dans la deuxième ancre `Groupe`, sélectionnez la propriété `facultatif`. Cela signifie que si `Groupe` échoue, il n'entraîne pas l'échec de `DelimitedSections`.
- Dans l'ancre `DelimitedSections`, définissez `using_placeholders = always`. Cela signifie que l'ancre cherche le séparateur de la section optionnelle, même si la section elle-même est manquante.

Supposez maintenant que, si la section `Expérience professionnelle` est absente, son séparateur l'est aussi.

```
-----  
Jane Palmer  
Employee ID 123456  
-----  
Education  
...
```

Dans ce cas, configurez `DelimitedSections` comme suit :

- Dans la deuxième ancre `Groupe`, sélectionnez la propriété `facultatif`.
- Dans l'ancre `DelimitedSections`, définissez `using_placeholders = never`. Cela signifie que l'ancre ne doit pas chercher le séparateur d'une section manquante.

Comment définir une ancre `DelimitedSections`

Ajoutez une ancre **`DelimitedSections`** en modifiant le script dans l'éditeur IntelliScript. Sous l'ancre **`DelimitedSections`**, ajoutez une séquence d'ancres qui analyse les sections.

Exemple en ligne

Pour obtenir un échantillon en ligne de cette ancre, ouvrez le projet `samples\Projects\DelimitedSections\DelimitedSections.cmw`. L'exemple illustre une ancre `DelimitedSections` qui analyse les sections séparées par un symbole `|`. Chaque section est analysée par une seule ancre `Contenu`.

EmbeddedParser

L'ancre **`EmbeddedParser`** utilise un analyseur secondaire pour analyser sa zone de recherche. Il peut s'appeler lui-même de façon récursive.

Le tableau suivant décrit les propriétés de l'ancre **EmbeddedParser** :

Propriété	Description
désactivé	Détermine si le script ignore le composant et tous les composants enfants. Utilisez cette propriété pour tester, déboguer et modifier un script. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none"> - Sélectionné. Le script ignore le composant. - Effacé. Le script applique le composant. La valeur par défaut est vide.
marquage	Détermine si une ancre est utilisée comme début du champ de recherche pour l'ancre qui suit. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none"> - position de départ. Placer un point de référence avant l'ancre actuelle. - position de fin. Placer un point de référence après l'ancre actuelle. - complet. Placer un point de référence avant et après l'ancre actuelle. - aucun. Ne pas créer de point de référence. Pour plus d'informations, voir "Comment un analyseur recherche-t-il les ancres ?" à la page 210
nom	Libellé descriptif pour le composant. Ce libellé s'affiche dans le fichier journal et la vue Événements . Utilisez la propriété nom pour identifier le composant ayant provoqué l'événement.
on_fail	Action à effectuer si le composant échoue. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none"> - Vide. N'effectuer aucune action. - CustomLog. Écrire dans le journal utilisateur. - LogError. Écrire un message d'erreur dans le journal du moteur. - LogInfo. Écrire un message d'information dans le journal du moteur. - LogWarning. Écrire un message d'avertissement dans le journal du moteur. - NotifyFailure. Envoyer une notification. Par défaut, la valeur est vide. Pour plus d'informations sur la gestion des défaillances de composants, consultez la section "Traitement des échecs" à la page 404 .
facultatif	Détermine si la défaillance d'un composant entraîne l'échec du composant parent. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none"> - Sélectionné. La défaillance du composant n'entraîne pas l'échec du composant parent. - Effacé. La défaillance du composant entraîne l'échec du composant parent. Par défaut, la valeur est vide. Pour plus d'informations sur l'échec des composants, consultez la section "Traitement des échecs" à la page 404 .
Analyseur	Détermine le nom d'un analyseur secondaire défini dans le même projet.
phase	Détermine les situations dans lesquelles le script traite le composant. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none"> - initiale. Le script traite le composant pendant la phase initiale. - principale. Le script traite le composant pendant la phase principale. - finale. Le script traite le composant pendant la phase finale. Pour plus d'informations, voir "Comment un analyseur recherche-t-il les ancres ?" à la page 210 La valeur par défaut est Principale.
remarque	Commentaire défini par l'utilisateur, qui décrit l'objectif ou l'action du composant.
schema_connections	Définit une liste de sous-composants Connecter qui définissent la relation entre des zones de stockage des données dans la sortie de l'analyseur principal et l'analyseur secondaire. Pour plus d'informations, consultez la section "Connecter" à la page 253
source_transformers	Définit une séquence de transformateurs que l'analyseur applique à la zone de recherche avant son traitement par l'analyseur secondaire.

Exemple

Un document est délimité par des tabulations, sauf pour une section délimitée par des virgules.

Pour analyser le document, vous pouvez définir un analyseur principal qui utilise le format `TabDelimited`. Définissez un autre analyseur qui utilise le format `CommaDelimited`. Utilisez une ancre `EmbeddedParser` pour exécuter le second analyseur au sein de l'exécution du premier analyseur.

Exemple en ligne

Pour un échantillon en ligne de cette ancre, ouvrez le projet `samples\Projects\EmbeddedParser\EmbeddedParser.cmw`. L'exemple utilise un analyseur principal pour déterminer l'emplacement d'une adresse. Ensuite, il exécute `EmbeddedParser` pour analyser l'adresse.

EnclosedGroup

L'ancre **EnclosedGroup** définit une région limitée qui contient des ancres imbriquées. Les limites sont spécifiées par des ancres **ouverture** et **fermeture**. Dans le cas de limites imbriquées (comme des parenthèses ou des balises HTML), **EnclosedGroup** trouve les balises appariées.

Le tableau suivant décrit les propriétés de l'ancre **EnclosedGroup** :

Propriété	Description
fermeture	Définit l'ancre de fermeture de EnclosedGroup .
désactivé	Détermine si le script ignore le composant et tous les composants enfants. Utilisez cette propriété pour tester, déboguer et modifier un script. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none">- Sélectionné. Le script ignore le composant.- Effacé. Le script applique le composant. La valeur par défaut est vide.
marquage	Détermine si une ancre est utilisée comme début du champ de recherche pour l'ancre qui suit. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none">- position de départ. Placer un point de référence avant l'ancre actuelle.- position de fin. Placer un point de référence après l'ancre actuelle.- complet. Placer un point de référence avant et après l'ancre actuelle.- aucun. Ne pas créer de point de référence. Pour plus d'informations, voir "Comment un analyseur recherche-t-il les ancres ?" à la page 210
nom	Libellé descriptif pour le composant. Ce libellé s'affiche dans le fichier journal et la vue Événements . Utilisez la propriété nom pour identifier le composant ayant provoqué l'événement.
no_initial_phase	Détermine si le script recherche des ancres imbriquées dans la phase principale. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none">- Effacé. Rechercher les ancres imbriquées selon leurs propriétés individuelles.- Sélectionné. Rechercher les ancres imbriquées dans la phase principale. Par défaut, la valeur est vide.
notifications	Liste de composants Notificationhandler qui gèrent les notifications de composants imbriqués. Pour plus d'informations, voir "Notifications" à la page 423

Propriété	Description
on_fail	<p>Action à effectuer si le composant échoue. Vous pouvez choisir l'une des options suivantes :</p> <ul style="list-style-type: none"> - Vide. N'effectuer aucune action. - CustomLog. Écrire dans le journal utilisateur. - LogError. Écrire un message d'erreur dans le journal du moteur. - LogInfo. Écrire un message d'information dans le journal du moteur. - LogWarning. Écrire un message d'avertissement dans le journal du moteur. - NotifyFailure. Envoyer une notification. <p>Par défaut, la valeur est vide. Pour plus d'informations sur la gestion des défaillances de composants, consultez la section "Traitement des échecs" à la page 404.</p>
ouverture	Définit l'ancre d'ouverture de EnclosedGroup .
facultatif	<p>Détermine si la défaillance d'un composant entraîne l'échec du composant parent. Vous pouvez choisir l'une des options suivantes :</p> <ul style="list-style-type: none"> - Sélectionné. La défaillance du composant n'entraîne pas l'échec du composant parent. - Effacé. La défaillance du composant entraîne l'échec du composant parent. <p>Par défaut, la valeur est vide. Pour plus d'informations sur l'échec des composants, consultez la section "Traitement des échecs" à la page 404.</p>
phase	<p>Détermine les situations dans lesquelles le script traite le composant. Vous pouvez choisir l'une des options suivantes :</p> <ul style="list-style-type: none"> - initiale. Le script traite le composant pendant la phase initiale. - principale. Le script traite le composant pendant la phase principale. - finale. Le script traite le composant pendant la phase finale. <p>Pour plus d'informations, voir "Comment un analyseur recherche-t-il les ancres ?" à la page 210 La valeur par défaut est Principale.</p>
remarque	Commentaire défini par l'utilisateur, qui décrit l'objectif ou l'action du composant.
source	<p>Définit une séquence de zones de stockage des données à entrer dans le EnclosedGroup. Chaque zone de stockage des données est identifiée par l'une des propriétés suivantes :</p> <ul style="list-style-type: none"> - Localisateur. Identifie une zone de stockage des données à occurrence unique ou à occurrences multiples. Pour les zones de stockage des données à occurrences multiples, chaque itération accède à une nouvelle occurrence. - LocatorByKey. Identifie une zone de stockage des données à occurrences multiples par clé. - LocatorByOccurrence. Identifie une zone de stockage des données à occurrences multiples par numéro séquentiel. <p>Utilisez la propriété source lorsque EnclosedGroup est appelé par un autre composant. Pour plus d'informations, voir "Propriété source" à la page 370</p>
cible	<p>Définit une séquence de zones de stockage des données pour la sortie depuis le EnclosedGroup. Si une zone de stockage des données n'existe pas encore, elle est créée. Chaque zone de stockage des données est identifiée par l'une des propriétés suivantes :</p> <ul style="list-style-type: none"> - Localisateur. Identifie une zone de stockage des données à occurrence unique ou à occurrences multiples. Pour les zones de stockage des données à occurrences multiples, chaque itération crée une nouvelle occurrence. - LocatorByKey. Identifie une zone de stockage des données à occurrences multiples par clé. - LocatorByOccurrence. Identifie une zone de stockage des données à occurrences multiples par numéro séquentiel. <p>Utilisez la propriété cible lorsque de la sortie de EnclosedGroup est utilisée par un autre composant. Pour plus d'informations, voir "Propriété target" à la page 374</p>

Un **EnclosedGroup** est similaire à une ancre **Contenu** avec un **opening_marker** et un **closing_marker**. Toutefois :

- L'ancre **Contenu** récupère l'ensemble du contenu situé entre les marqueurs d'ouverture et de fermeture, sans autre analyse.
- Le **EnclosedGroup** vous permet d'analyser davantage le contenu entre les ancres **ouverture** et **fermeture**.

Exemple

Vous pouvez définir une table HTML en tant que **EnclosedGroup**, avec les balises `<table>` et `</table>` faisant office de marqueurs d'ouverture et de fermeture. Les ancres imbriquées analysent le contenu de la table.

Supposons que l'élément `<table>` contienne un élément imbriqué `<table>`. Autrement dit, une table est imbriquée dans une cellule de table. L'ancre **EnclosedGroup** fait correspondre la balise `<table>` parente avec la balise `</table>` parente. Elle ne fait pas correspondre la balise `<table>` parente avec la balise `</table>` imbriquée, ce qui serait une erreur d'identification de la table.

Comment définir une ancre EnclosedGroup

Vous pouvez définir une ancre **EnclosedGroup** en modifiant le script dans l'éditeur IntelliScript. Ajoutez les ancres imbriquées qui effectueront l'analyse du contenu.

ExtractRecord

L'ancre **ExtractRecord** extrait un enregistrement, assigne les identifiants à l'enregistrement et transmet l'enregistrement aux sous-éléments d'une **StructureDefinition**. **ExtractRecord** est utilisé dans la propriété **format_definition** d'une **StructureDefinition**.

ExtractRecord extrait sa portée de recherche entière. Par exemple, si vous insérez un **ExtractRecord** entre deux ancres **Marqueur**, il extrait la portée entre les marqueurs.

Le tableau suivant décrit les propriétés de l'ancre **ExtractRecord** :

Propriété	Description
désactivé	Détermine si le script ignore le composant et tous les composants enfants. Utilisez cette propriété pour tester, déboguer et modifier un script. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none">- Sélectionné. Le script ignore le composant.- Effacé. Le script applique le composant. La valeur par défaut est vide.
ids	Définit une liste des identifiants attachés à l'enregistrement. StructureDefinition utilise les identifiants pour faire correspondre l'enregistrement à un sous-élément. Dans l'entrée de chaque liste, entrez une valeur d'identifiant ou accédez à une zone de stockage des données contenant la valeur.
nom	Libellé descriptif pour le composant. Ce libellé s'affiche dans le fichier journal et la vue Événements . Utilisez la propriété nom pour identifier le composant ayant provoqué l'événement.
facultatif	Détermine si la défaillance d'un composant entraîne l'échec du composant parent. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none">- Sélectionné. La défaillance du composant n'entraîne pas l'échec du composant parent.- Effacé. La défaillance du composant entraîne l'échec du composant parent. Par défaut, la valeur est vide. Pour plus d'informations sur l'échec des composants, consultez la section "Traitement des échecs" à la page 404 .

Propriété	Description
phase	<p>Détermine les situations dans lesquelles le script traite le composant. Vous pouvez choisir l'une des options suivantes :</p> <ul style="list-style-type: none"> - initiale. Le script traite le composant pendant la phase initiale. - principale. Le script traite le composant pendant la phase principale. - finale. Le script traite le composant pendant la phase finale. <p>Pour plus d'informations, voir "Comment un analyseur recherche-t-il les ancrés ?" à la page 210 La valeur par défaut est Principale.</p>
remarque	Commentaire défini par l'utilisateur, qui décrit l'objectif ou l'action du composant.

FindReplaceAnchor

L'ancre **FindReplaceAnchor** marque le texte source et spécifie le texte de remplacement pour la transformation par le transformateur **TransformByParser**.

Le tableau suivant décrit les propriétés de l'ancre **FindReplaceAnchor** :

Propriété	Description
désactivé	<p>Détermine si le script ignore le composant et tous les composants enfants. Utilisez cette propriété pour tester, déboguer et modifier un script. Vous pouvez choisir l'une des options suivantes :</p> <ul style="list-style-type: none"> - Sélectionné. Le script ignore le composant. - Effacé. Le script applique le composant. <p>La valeur par défaut est vide.</p>
marquage	<p>Détermine si une ancre est utilisée comme début du champ de recherche pour l'ancre qui suit. Vous pouvez choisir l'une des options suivantes :</p> <ul style="list-style-type: none"> - position de départ. Placer un point de référence avant l'ancre actuelle. - position de fin. Placer un point de référence après l'ancre actuelle. - complet. Placer un point de référence avant et après l'ancre actuelle. - aucun. Ne pas créer de point de référence. <p>Pour plus d'informations, voir "Comment un analyseur recherche-t-il les ancrés ?" à la page 210</p>
nom	Libellé descriptif pour le composant. Ce libellé s'affiche dans le fichier journal et la vue Événements . Utilisez la propriété nom pour identifier le composant ayant provoqué l'événement.
no_initial_phase	<p>Détermine si le script recherche des ancrés imbriqués dans la phase principale. Vous pouvez choisir l'une des options suivantes :</p> <ul style="list-style-type: none"> - Effacé. Rechercher les ancrés imbriqués selon leurs propriétés individuelles. - Sélectionné. Rechercher les ancrés imbriqués dans la phase principale. <p>Par défaut, la valeur est vide.</p>
on_fail	<p>Action à effectuer si le composant échoue. Vous pouvez choisir l'une des options suivantes :</p> <ul style="list-style-type: none"> - Vide. N'effectuer aucune action. - CustomLog. Écrire dans le journal utilisateur. - LogError. Écrire un message d'erreur dans le journal du moteur. - LogInfo. Écrire un message d'information dans le journal du moteur. - LogWarning. Écrire un message d'avertissement dans le journal du moteur. - NotifyFailure. Envoyer une notification. <p>Par défaut, la valeur est vide. Pour plus d'informations sur la gestion des défaillances de composants, consultez la section "Traitement des échecs" à la page 404.</p>

Propriété	Description
on_partial_match	Détermine le comportement lorsque FindReplaceAnchor ne trouve pas toutes ses ancrs imbriquées non facultatives. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none"> - échec. FindReplaceAnchor échoue. Valeur par défaut. - ignorer. FindReplaceAnchor supprime la zone couverte par les ancrs imbriquées avec succès à partir de sa zone de recherche et tente de trouver à nouveau toutes les ancrs imbriquées. Il répète ce processus jusqu'à ce qu'il trouve les ancrs ou jusqu'à l'échec.
facultatif	Détermine si la défaillance d'un composant entraîne l'échec du composant parent. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none"> - Sélectionné. La défaillance du composant n'entraîne pas l'échec du composant parent. - Effacé. La défaillance du composant entraîne l'échec du composant parent. Par défaut, la valeur est vide. Pour plus d'informations sur l'échec des composants, consultez la section "Traitement des échecs" à la page 404 .
phase	Détermine les situations dans lesquelles le script traite le composant. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none"> - initiale. Le script traite le composant pendant la phase initiale. - principale. Le script traite le composant pendant la phase principale. - finale. Le script traite le composant pendant la phase finale. Pour plus d'informations, voir "Comment un analyseur recherche-t-il les ancrs ?" à la page 210 La valeur par défaut est Principale.
remarque	Commentaire défini par l'utilisateur, qui décrit l'objectif ou l'action du composant.
replace_with	Définit une chaîne de remplacement littérale ou une zone de stockage des données qui contient la chaîne de remplacement.
source	Définit une séquence de zone de stockage des données pour l'entrée dans le FindReplaceAnchor . Chaque zone de stockage des données est identifiée par l'une des propriétés suivantes : <ul style="list-style-type: none"> - Localisateur. Identifie une zone de stockage des données à occurrence unique ou à occurrences multiples. Pour les zones de stockage des données à occurrences multiples, chaque itération accède à une nouvelle occurrence. - LocatorByKey. Identifie une zone de stockage des données à occurrences multiples par clé. - LocatorByOccurrence. Identifie une zone de stockage des données à occurrences multiples par numéro séquentiel. Utilisez la propriété source quand FindReplaceAnchor est appelé par un autre composant. Pour plus d'informations, voir "Propriété source" à la page 370
cible	Définit une séquence de zones de stockage des données pour la sortie depuis le FindReplaceAnchor . Si une zone de stockage des données n'existe pas encore, elle est créée. Chaque zone de stockage des données est identifiée par l'une des propriétés suivantes : <ul style="list-style-type: none"> - Localisateur. Identifie une zone de stockage des données à occurrence unique ou à occurrences multiples. Pour les zones de stockage des données à occurrences multiples, chaque itération crée une nouvelle occurrence. - LocatorByKey. Identifie une zone de stockage des données à occurrences multiples par clé. - LocatorByOccurrence. Identifie une zone de stockage des données à occurrences multiples par numéro séquentiel. Utilisez la propriété cible lorsque la sortie de FindReplaceAnchor est utilisée par un autre composant. Pour plus d'informations, voir "Propriété target" à la page 374

Si **FindReplaceAnchor** ne contient pas d'ancres imbriquées, il marque tout le texte présent dans sa zone de recherche. Par exemple, si **FindReplaceAnchor** se trouve entre deux ancrs **Marqueur**, il marque le texte entre celles-ci.

Si **FindReplaceAnchor** contient une ancre **Marqueur**, il marque le **Marqueur** pour le remplacement.

Si **FindReplaceAnchor** contient deux ancrs **Marqueur**, il marque les ancrs **Marqueur** et le segment entre eux pour le remplacement.

Le texte de remplacement peut être une chaîne de remplacement statique ou une chaîne récupérée dynamiquement depuis le document source.

Pour plus d'informations, consultez la section ["TransformByParser" à la page 292](#)

Exemple

Vous voulez ajouter des numéros de lignes à un document texte. Vous pouvez ajouter les numéros de lignes de la façon suivante :

1. Créez un analyseur et ajoutez-y un `RepeatingGroup`.
2. Dans `RepeatingGroup`, ajoutez `FindReplaceAnchor`.
3. Dans `FindReplaceAnchor`, ajoutez une ancre `Marqueur` et définissez sa propriété `recherche` à `NewlineSearch`.
Cela entraîne le marquage par `FindReplaceAnchor` de chaque retour à la ligne dans le document.
4. Configurez `RepeatingGroup` pour qu'il stocke `current_iteration` dans une variable. Affectez la propriété `replace_with` de `FindReplaceAnchor` la variable.
5. Au niveau global du script, définissez un transformateur `TransformByParser`. Définissez sa propriété `analyseur` sur l'analyseur.
6. Définissez `TransformByParser` comme composant de démarrage de la transformation.
Le transformateur renvoie une version modifiée du fichier original, contenant les numéros de lignes.

Comment définir une ancre `FindReplaceAnchor` ?

Vous pouvez définir une ancre **FindReplaceAnchor** en modifiant le script dans l'éditeur IntelliScript. Si nécessaire, ajoutez les ancres imbriquées marquant une sous-chaîne à remplacer.

Groupe

L'ancre **Groupe** lie ensemble une séquence d'ancres et d'actions.

Les propriétés du **Groupe** s'appliquent à tous les composants enfant. Utilisez un **Groupe** pour définir des opérations que le script doit effectuer dans un ensemble d'ancres ou pour contrôler la phase des ancres imbriquées.

Le tableau suivant décrit les propriétés de l'ancre **Groupe** :

Propriété	Description
absent	Définit le comportement de l'ancre Groupe lorsqu'une de ses ancres ou actions imbriquées et non facultatives échoue. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none">- Sélectionné. Échec du Groupe.- Effacé. Comportement normal. Utilisez cette fonctionnalité pour tester l'absence d'ancres imbriquées.
désactivé	Détermine si le script ignore le composant et tous les composants enfants. Utilisez cette propriété pour tester, déboguer et modifier un script. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none">- Sélectionné. Le script ignore le composant.- Effacé. Le script applique le composant. La valeur par défaut est vide.

Propriété	Description
marquage	<p>Détermine si une ancre est utilisée comme début du champ de recherche pour l'ancre qui suit. Vous pouvez choisir l'une des options suivantes :</p> <ul style="list-style-type: none"> - position de départ. Placer un point de référence avant l'ancre actuelle. - position de fin. Placer un point de référence après l'ancre actuelle. - complet. Placer un point de référence avant et après l'ancre actuelle. - aucun. Ne pas créer de point de référence. <p>Pour plus d'informations, voir "Comment un analyseur recherche-t-il les ancres ?" à la page 210</p>
nom	<p>Libellé descriptif pour le composant. Ce libellé s'affiche dans le fichier journal et la vue Événements. Utilisez la propriété nom pour identifier le composant ayant provoqué l'événement.</p>
no_initial_phase	<p>Détermine si le script recherche des ancres imbriquées dans la phase principale. Vous pouvez choisir l'une des options suivantes :</p> <ul style="list-style-type: none"> - Effacé. Rechercher les ancres imbriquées selon leurs propriétés individuelles. - Sélectionné. Rechercher les ancres imbriquées dans la phase principale. <p>Par défaut, la valeur est vide.</p>
notifications	<p>Liste de composants Notificationhandler qui gèrent les notifications de composants imbriqués. Pour plus d'informations, voir "Notifications" à la page 423</p>
on_fail	<p>Action à effectuer si le composant échoue. Vous pouvez choisir l'une des options suivantes :</p> <ul style="list-style-type: none"> - Vide. N'effectuer aucune action. - CustomLog. Écrire dans le journal utilisateur. - LogError. Écrire un message d'erreur dans le journal du moteur. - LogInfo. Écrire un message d'information dans le journal du moteur. - LogWarning. Écrire un message d'avertissement dans le journal du moteur. - NotifyFailure. Envoyer une notification. <p>Par défaut, la valeur est vide. Pour plus d'informations sur la gestion des défaillances de composants, consultez la section "Traitement des échecs" à la page 404.</p>
on_partial_match	<p>Détermine le comportement lorsque le Groupe ne trouve pas toutes ses ancres imbriquées, non facultatives. Vous pouvez choisir l'une des options suivantes :</p> <ul style="list-style-type: none"> - échec. Échec du Groupe. Valeur par défaut. - ignorer. Le Groupe retire la zone étendue par les ancres imbriquées avec succès depuis sa portée de recherche et tente de trouver toutes les ancres imbriquées à nouveau. Il répète ce processus jusqu'à ce qu'il trouve les ancres ou jusqu'à l'échec.
facultatif	<p>Détermine si la défaillance d'un composant entraîne l'échec du composant parent. Vous pouvez choisir l'une des options suivantes :</p> <ul style="list-style-type: none"> - Sélectionné. La défaillance du composant n'entraîne pas l'échec du composant parent. - Effacé. La défaillance du composant entraîne l'échec du composant parent. <p>Par défaut, la valeur est vide. Pour plus d'informations sur l'échec des composants, consultez la section "Traitement des échecs" à la page 404.</p>
phase	<p>Détermine les situations dans lesquelles le script traite le composant. Vous pouvez choisir l'une des options suivantes :</p> <ul style="list-style-type: none"> - initiale. Le script traite le composant pendant la phase initiale. - principale. Le script traite le composant pendant la phase principale. - finale. Le script traite le composant pendant la phase finale. <p>Pour plus d'informations, voir "Comment un analyseur recherche-t-il les ancres ?" à la page 210 La valeur par défaut est Principale.</p>
remarque	<p>Commentaire défini par l'utilisateur, qui décrit l'objectif ou l'action du composant.</p>

Propriété	Description
search_order	Définit la direction de traitement des ancres imbriquées. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none"> - top-down. Le traitement des ancres imbriquées est réalisé dans l'ordre défini dans le script. - bottom-up. Les ancres imbriquées sont traitées dans l'ordre inverse. Utilisez cette option lorsque les données d'une version ultérieure de l'ancre affectent le traitement d'une première ancre.
source	Définit une séquence de zones de stockage des données à entrer dans le Groupe . Chaque zone de stockage des données est identifiée par l'une des propriétés suivantes : <ul style="list-style-type: none"> - Localisateur. Identifie une zone de stockage des données à occurrence unique ou à occurrences multiples. Pour les zones de stockage des données à occurrences multiples, chaque itération accède à une nouvelle occurrence. - LocatorByKey. Identifie une zone de stockage des données à occurrences multiples par clé. - LocatorByOccurrence. Identifie une zone de stockage des données à occurrences multiples par numéro séquentiel. Utilisez la propriété source lorsque la propriété Groupe est appelée par un autre composant. Pour plus d'informations, voir "Propriété source" à la page 370
cible	Définit une séquence de zones de stockage des données pour la sortie depuis le Groupe . Si une zone de stockage des données n'existe pas encore, elle est créée. Chaque zone de stockage des données est identifiée par l'une des propriétés suivantes : <ul style="list-style-type: none"> - Localisateur. Identifie une zone de stockage des données à occurrence unique ou à occurrences multiples. Pour les zones de stockage des données à occurrences multiples, chaque itération crée une nouvelle occurrence. - LocatorByKey. Identifie une zone de stockage des données à occurrences multiples par clé. - LocatorByOccurrence. Identifie une zone de stockage des données à occurrences multiples par numéro séquentiel. Utilisez la propriété cible lorsque la sortie du Groupe est utilisée par un autre composant. Pour plus d'informations, voir "Propriété target" à la page 374

Comment définir une ancre Groupe ?

Vous pouvez définir une ancre **Groupe** en modifiant le script dans l'éditeur IntelliScript. Ajoutez des ancres et des actions imbriquées qui analysent le contenu d'un **Groupe**.

Groupe optionnel

Vous pouvez utiliser la propriété **facultatif** d'un **Groupe** pour empêcher le script de tenter de récupérer le texte depuis une section manquante d'un document.

Par exemple, pour analyser la source

```
First name: Ron
```

vous pouvez définir **Prénom** : en tant que **Marqueur** et **Ron** en tant que **Contenu**. Si certains documents source ne contiennent pas de données prénom, vous pouvez placer le **Marqueur** et le **Contenu** dans un **Groupe** et le rendre optionnel. Si **Prénom** : est introuvable, le **Groupe** échoue immédiatement et l'analyseur ne recherche pas l'ancre **Contenu**.

Il y a une différence entre rendre le **Groupe** optionnel et rendre optionnelles ses ancres imbriquées. Si vous rendez optionnels à la fois **Marqueur** et **Contenu** au lieu du **Groupe**, le script ignore l'échec du **Marqueur** et recherche le **Contenu**. Le résultat peut être l'extraction d'un texte non pertinent.

Exemple en ligne

Pour un échantillon en ligne de cette ancre, ouvrez le projet `samples\Projects\persistent_search\persistent_search.cmw`.

L'exemple illustre un Groupe configuré avec la propriété `on_partial_match = skip`. Le Groupe contient deux ancres **Marqueur** :

- Le premier **Marqueur** recherche le texte A.
- Le deuxième **Marqueur** recherche une chaîne contenant tout nombre de caractères *. Il a la propriété `adjacent`, ce qui signifie qu'il doit être adjacent au premier **Marqueur**.

Au premier passage, le Groupe trouve un caractère A au début du document source. Toutefois il ne trouve pas le deuxième **Marqueur** adjacent au caractère A.

Le Groupe réduit donc la portée de sa recherche en éliminant le premier caractère A et recherche à nouveau les deux ancres **Marqueur** adjacentes. Il poursuit cette procédure jusqu'à ce qu'il trouve une chaîne A*, qui contient les ancres **Marqueur** adjacentes.

Vous pouvez observer le comportement dans le journal des événements. Le journal enregistre que le Groupe échoue aux deux premières tentatives et réussit à la troisième.

Faites des expériences avec les réglages `on_partial_match` et `adjacent`. Vous pouvez voir les effets dans le codage couleur de l'exemple de source.

Vous pouvez aussi essayer d'exécuter l'exemple, bien que le fichier de résultat soit vide, car l'analyseur ne contient pas d'ancre **Contenu**. Si vous définissez `on_partial_match = fail`, vous verrez dans le journal des événements que l'analyseur échoue, car le Groupe ne peut pas trouver les ancres adjacentes.

Marqueur

Une ancre **Marqueur** définit un emplacement dans un document source. Elle est utilisée comme un point de référence, à partir duquel le script cherche les ancres suivantes.

Par défaut, la propriété **phase** d'un **Marqueur** est `initiale`, ce qui signifie que le script parcourt un document pour trouver les ancres **Marqueur** avant d'y chercher les ancres **Contenu**. Pour plus d'informations, consultez la section [“Comment un analyseur recherche-t-il les ancres ?” à la page 210](#).

Le tableau suivant décrit les propriétés de l'ancre **Marqueur** :

Propriété	Description
absent	Détermine si le texte ou le modèle spécifiés sont absents dans le document. La propriété absent possède les options suivantes : <ul style="list-style-type: none">- Sélectionné. Si le texte spécifié apparaît dans le document, Marqueur échoue.- Effacé. Si le texte spécifié apparaît dans le document, Marqueur réussit. La valeur par défaut est Effacé.
adjacent	Si sélectionné, le Marqueur doit être adjacent à l'ancre au début de sa zone de recherche. Si direction est défini sur <code>en arrière</code> , il doit être adjacent à l'ancre située à la fin de la zone de recherche. S'il est désélectionné, le script peut ignorer le texte jusqu'à ce qu'il trouve le Marqueur . La propriété adjacent possède les options suivantes : <ul style="list-style-type: none">- Sélectionné. Le Marqueur doit se produire immédiatement après le début de la zone de recherche si direction est défini sur <code>en avant</code> ou immédiatement avant la fin de la zone de recherche si direction est défini sur <code>en arrière</code>.- Effacé. Le Marqueur peut se produire n'importe où dans la zone de recherche. La valeur par défaut est Effacé.

Propriété	Description
compteur	Définit le numéro de l'occurrence à trouver. Par exemple, pour définir le Marqueur au deuxième retour à la ligne suivant l'ancre précédente, définissez search sur <code>NewlineSearch</code> et compteur sur 2.
direction	<p>Sens de recherche de l'ancre dans le champ de recherche. Vous pouvez choisir l'une des options suivantes :</p> <ul style="list-style-type: none"> - en arrière. Rechercher à partir de la fin du champ de recherche et trouver la dernière instance de l'ancre. - en avant. Rechercher depuis le début du champ de recherche et trouver la première instance de l'ancre. <p>Pour une ancre Marqueur, vous pouvez modifier ce comportement en utilisant la propriété count. Par exemple, si direction = backward et count = 2, le script trouve l'avant-dernière instance.</p> <p>La valeur par défaut est en avant. Pour plus d'informations, voir "Comment un analyseur recherche-t-il les ancres ?" à la page 210</p>
désactivé	<p>Détermine si le script ignore le composant et tous les composants enfants. Utilisez cette propriété pour tester, déboguer et modifier un script. Vous pouvez choisir l'une des options suivantes :</p> <ul style="list-style-type: none"> - Sélectionné. Le script ignore le composant. - Effacé. Le script applique le composant. <p>La valeur par défaut est vide.</p>
marquage	<p>Détermine si une ancre est utilisée comme début du champ de recherche pour l'ancre qui suit. Vous pouvez choisir l'une des options suivantes :</p> <ul style="list-style-type: none"> - position de départ. Placer un point de référence avant l'ancre actuelle. - position de fin. Placer un point de référence après l'ancre actuelle. - complet. Placer un point de référence avant et après l'ancre actuelle. - aucun. Ne pas créer de point de référence. <p>Pour plus d'informations, voir "Comment un analyseur recherche-t-il les ancres ?" à la page 210</p>
nom	Libellé descriptif pour le composant. Ce libellé s'affiche dans le fichier journal et la vue Événements . Utilisez la propriété nom pour identifier le composant ayant provoqué l'événement.
on_fail	<p>Action à effectuer si le composant échoue. Vous pouvez choisir l'une des options suivantes :</p> <ul style="list-style-type: none"> - Vide. N'effectuer aucune action. - CustomLog. Écrire dans le journal utilisateur. - LogError. Écrire un message d'erreur dans le journal du moteur. - LogInfo. Écrire un message d'information dans le journal du moteur. - LogWarning. Écrire un message d'avertissement dans le journal du moteur. - NotifyFailure. Envoyer une notification. <p>Par défaut, la valeur est vide. Pour plus d'informations sur la gestion des défaillances de composants, consultez la section "Traitement des échecs" à la page 404.</p>
facultatif	<p>Détermine si la défaillance d'un composant entraîne l'échec du composant parent. Vous pouvez choisir l'une des options suivantes :</p> <ul style="list-style-type: none"> - Sélectionné. La défaillance du composant n'entraîne pas l'échec du composant parent. - Effacé. La défaillance du composant entraîne l'échec du composant parent. <p>Par défaut, la valeur est vide. Pour plus d'informations sur l'échec des composants, consultez la section "Traitement des échecs" à la page 404.</p>
phase	<p>Détermine les situations dans lesquelles le script traite le composant. Vous pouvez choisir l'une des options suivantes :</p> <ul style="list-style-type: none"> - initiale. Le script traite le composant pendant la phase initiale. - principale. Le script traite le composant pendant la phase principale. - finale. Le script traite le composant pendant la phase finale. <p>Pour plus d'informations, voir "Comment un analyseur recherche-t-il les ancres ?" à la page 210</p> <p>La valeur par défaut est initiale.</p>

Propriété	Description
remarque	Commentaire défini par l'utilisateur, qui décrit l'objectif ou l'action du composant.
rechercher	<p>Définit les critères de recherche du Marqueur. Les critères de recherche déterminent l'endroit où le Marqueur est localisé dans la zone de recherche. Par exemple, NewlineSearch localise le Marqueur sur un caractère de retour à la ligne. TextSearch localise le Marqueur à une chaîne spécifiée. Pour plus d'informations, voir "Comment un analyseur recherche-t-il les ancrés ?" à la page 210</p> <p>La valeur de cette propriété est l'un des composants chercheurs suivants :</p> <ul style="list-style-type: none"> - NewlineSearch. Recherche un caractère de retour à la ligne. - TextSearch. Recherche une chaîne de texte prédéfinie ou une chaîne de texte qui est stockée dans une zone de stockage des données. - PatternSearch. Recherche une chaîne correspondant à une expression régulière. - OffsetSearch. Ignore un nombre de caractères prédéfini suivant le point de référence précédent ou un nombre de caractères stocké dans une zone de stockage des données. Le Marqueur est le point suivant les caractères ignorés. - TypeSearch. Recherche une chaîne conforme au type de données spécifié. <p>Pour plus d'informations, consultez le "Référence du composant de recherche" à la page 246.</p>

Comment définir une ancre Marqueur

Vous pouvez définir une ancre **Marqueur** en modifiant le script dans l'éditeur IntelliScript. Pour plus d'informations, consultez la section ["Définition des ancrés" à la page 207](#).

Exemple en ligne

Dans le dossier Online Samples, ouvrez `Projects\Markers\Markers.cmw`. L'exemple montre des ancrés **Marqueur** recherchant :

- Une chaîne de texte prédéfinie
- Un caractère de retour à la ligne
- Un décalage
- Un type de données
- Une expression régulière

Si vous exécutez l'analyseur, notez que le fichier de résultat est vide, car la configuration n'a aucune ancre Contenu.

RepeatingGroup

L'ancre **RepeatingGroup** analyse une région qui contient des segments répétitifs. Chaque segment est appelé une itération et peut être délimité par un **séparateur**. **RepeatingGroup** contient une séquence d'ancres et d'actions imbriquées qui analysent chaque itération de la même manière.

Le tableau suivant décrit les propriétés de l'ancre **RepeatingGroup** :

Propriété	Description
compteur	Définit un nombre ou une zone de stockage des données qui contient le nombre d'itérations à exécuter. S'il est laissé vide, les itérations se poursuivent jusqu'à la fin de la zone de recherche. Si compteur est égal à 0, RepeatingGroup ne cherche pas d'itérations. Dans ce cas, RepeatingGroup réussit, mais ne produit aucune sortie.
current_iteration	Définit une zone de stockage des données dans laquelle RepeatingGroup sort le numéro de l'itération en cours.
désactivé	Détermine si le script ignore le composant et tous les composants enfants. Utilisez cette propriété pour tester, déboguer et modifier un script. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none"> - Sélectionné. Le script ignore le composant. - Effacé. Le script applique le composant. La valeur par défaut est vide.
iteration_order	Définit l'ordre dans lequel les itérations sont traitées. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none"> - top-down. Le traitement des itérations est réalisé dans l'ordre défini par le script. - bottom-up. Les itérations sont traitées dans l'ordre inverse. Utilisez cette option si les données issues d'une itération ultérieure affectent la manière dont vous traitez une itération antérieure.
marquage	Détermine si une ancre est utilisée comme début du champ de recherche pour l'ancre qui suit. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none"> - position de départ. Placer un point de référence avant l'ancre actuelle. - position de fin. Placer un point de référence après l'ancre actuelle. - complet. Placer un point de référence avant et après l'ancre actuelle. - aucun. Ne pas créer de point de référence. Pour plus d'informations, voir "Comment un analyseur recherche-t-il les ancres ?" à la page 210
nom	Libellé descriptif pour le composant. Ce libellé s'affiche dans le fichier journal et la vue Événements . Utilisez la propriété nom pour identifier le composant ayant provoqué l'événement.
no_initial_phase	Détermine si le script recherche des ancres imbriquées dans la phase principale. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none"> - Effacé. Rechercher les ancres imbriquées selon leurs propriétés individuelles. - Sélectionné. Rechercher les ancres imbriquées dans la phase principale. Par défaut, la valeur est vide.
notifications	Liste de composants Notificationhandler qui gèrent les notifications de composants imbriqués. Pour plus d'informations, voir "Notifications" à la page 423
on_fail	Action à effectuer si le composant échoue. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none"> - Vide. N'effectuer aucune action. - CustomLog. Écrire dans le journal utilisateur. - LogError. Écrire un message d'erreur dans le journal du moteur. - LogInfo. Écrire un message d'information dans le journal du moteur. - LogWarning. Écrire un message d'avertissement dans le journal du moteur. - NotifyFailure. Envoyer une notification. Par défaut, la valeur est vide. Pour plus d'informations sur la gestion des défaillances de composants, consultez la section "Traitement des échecs" à la page 404 .

Propriété	Description
on_iteration_fail	<p>Définit l'action lorsqu'une itération unique échoue. Vous pouvez choisir l'une des options suivantes :</p> <ul style="list-style-type: none"> - Effacé. Aucune action. - CustomLog. Écrit dans le journal utilisateur. - LogError. Écrit un message d'erreur dans le journal du moteur. - LogInfo. Écrit un message d'information dans le journal du moteur. - LogWarning. Écrit un message d'avertissement dans le journal du moteur. - NotifyFailure. Déclenche une notification. <p>Utilisez la propriété on_fail pour écrire une entrée si RepeatingGroup échoue complètement. Pour plus d'informations, consultez la section "Traitement des échecs" à la page 404</p>
on_partial_match	<p>Définit le comportement quand une partie seulement des ancrs requises imbriquées dans RepeatingGroup s'affichent dans l'entrée. La propriété on_partial_match possède les options suivantes :</p> <ul style="list-style-type: none"> - échec. L'itération échoue. - ignorer. RepeatingGroup supprime la zone couverte par les ancrs imbriquées avec succès à partir de sa zone de recherche et tente de trouver à nouveau toutes les ancrs imbriquées. La procédure suppression-nouvel-essai est répétée jusqu'à ce que l'itération réussisse ou bien jusqu'à ce qu'il n'y ait plus de correspondance partielle. S'il n'y a aucune correspondance partielle, l'itération échoue.
facultatif	<p>Détermine si la défaillance d'un composant entraîne l'échec du composant parent. Vous pouvez choisir l'une des options suivantes :</p> <ul style="list-style-type: none"> - Sélectionné. La défaillance du composant n'entraîne pas l'échec du composant parent. - Effacé. La défaillance du composant entraîne l'échec du composant parent. <p>Par défaut, la valeur est vide. Pour plus d'informations sur l'échec des composants, consultez la section "Traitement des échecs" à la page 404.</p>
phase	<p>Détermine les situations dans lesquelles le script traite le composant. Vous pouvez choisir l'une des options suivantes :</p> <ul style="list-style-type: none"> - initiale. Le script traite le composant pendant la phase initiale. - principale. Le script traite le composant pendant la phase principale. - finale. Le script traite le composant pendant la phase finale. <p>Pour plus d'informations, voir "Comment un analyseur recherche-t-il les ancrs ?" à la page 210 La valeur par défaut est Principale.</p>
remarque	Commentaire défini par l'utilisateur, qui décrit l'objectif ou l'action du composant.
search_order	<p>Définit l'ordre de traitement des ancrs imbriquées dans chaque itération. Vous pouvez choisir l'une des options suivantes :</p> <ul style="list-style-type: none"> - top-down. Le traitement des ancrs imbriquées est réalisé dans l'ordre défini dans le script. - bottom-up. Les ancrs imbriquées sont traitées dans l'ordre inverse. Sélectionnez cette option si les données issues d'une ancre ultérieure affectent la manière dont vous traitez une ancre antérieure.
séparateur	<p>Définit une ancre qui délimite les sections.</p> <p>Si vous laissez la propriété séparateur vide, RepeatingGroup ne recherche pas de délimiteur entre les itérations. Au contraire, il part du principe qu'une itération est terminée lorsqu'il a trouvé toutes les ancrs imbriquées. Il commence alors l'analyse de l'itération suivante en partant du haut de la séquence d'ancres imbriquées.</p> <p>Vous pouvez construire un séparateur complexe en insérant un Group dans la propriété séparateur au lieu d'un Marqueur.</p>

Propriété	Description
separator_position	<p>Définit le positionnement du séparateur par rapport aux sections. Vous pouvez choisir l'une des options suivantes :</p> <ul style="list-style-type: none"> - après. Il y a un séparateur après chaque section, y compris la dernière section. Par exemple, 1 2 3 4 - autour. Il y a des séparateurs avant et après chaque section, y compris la première et la dernière section. Par exemple, 1 2 3 4 - avant. Il y a un séparateur avant chaque section, y compris la première section. Par exemple, 1 2 3 4 - entre. Il y a un séparateur entre les sections consécutives, mais ni avant la première section, ni après la dernière. Par exemple, 1 2 3 4
skip_failed_iterations	<p>Détermine si les itérations échouées sont ignorées. Vous pouvez choisir l'une des options suivantes :</p> <ul style="list-style-type: none"> - Sélectionné. RepeatingGroup ignore une itération échouée et passe à l'itération suivante. Si une itération réussit, RepeatingGroup réussit. - Effacé. RepeatingGroup échoue si n'importe quelle itération échoue. <p>La propriété skip_failed_iterations a uniquement un effet si séparateur est défini. La valeur par défaut est Sélectionné.</p>
source	<p>Définit une séquence de zones de stockage des données pour l'entrée dans le RepeatingGroup. Chaque zone de stockage des données est identifiée par l'une des propriétés suivantes :</p> <ul style="list-style-type: none"> - Localisateur. Identifie une zone de stockage des données à occurrence unique ou à occurrences multiples. Pour les zones de stockage des données à occurrences multiples, chaque itération accède à une nouvelle occurrence. - LocatorByKey. Identifie une zone de stockage des données à occurrences multiples par clé. - LocatorByOccurrence. Identifie une zone de stockage des données à occurrences multiples par numéro séquentiel. <p>Utilisez la propriété source quand RepeatingGroup est appelé par un autre composant. Pour plus d'informations, consultez la section "Propriété source" à la page 370</p>
cible	<p>Définit une séquence de zones de stockage des données pour la sortie depuis le RepeatingGroup. Si une zone de stockage des données n'existe pas encore, elle est créée. Chaque zone de stockage des données est identifiée par l'une des propriétés suivantes :</p> <ul style="list-style-type: none"> - Localisateur. Identifie une zone de stockage des données à occurrence unique ou à occurrences multiples. Pour les zones de stockage des données à occurrences multiples, chaque itération crée une nouvelle occurrence. - LocatorByKey. Identifie une zone de stockage des données à occurrences multiples par clé. - LocatorByOccurrence. Identifie une zone de stockage des données à occurrences multiples par numéro séquentiel. <p>Utilisez la propriété cible lorsque la sortie de RepeatingGroup est utilisée par un autre composant. Pour plus d'informations, consultez la section "Propriété target" à la page 374</p>

Remarque: Pour analyser une région de sections qui nécessitent différents traitements, utilisez une ancre **DelimitedSections**.

Comment définir une ancre de groupe de répétition

Vous pouvez définir une ancre **RepeatingGroup** en modifiant le script dans l'éditeur IntelliScript. Ajoutez les ancres et les actions imbriquées qui analysent chaque itération de **RepeatingGroup**.

Recherche des itérations

Par défaut, `RepeatingGroup` recherche les itérations depuis le début jusqu'à la fin de sa zone de recherche. Pour plus d'informations, consultez la section ["Comment un analyseur recherche-t-il les ancrés ?" à la page 210](#).

Vous avez la possibilité, éventuellement, de définir la propriété `iteration_order` pour inverser le sens de la recherche.

Dans chaque itération :

- Si `RepeatingGroup` est configuré avec un `séparateur`, il recherche le séparateur suivant. Puis, il recherche les ancrés situés entre une paire de séparateurs.
- Si `RepeatingGroup` n'est pas configuré avec un `séparateur`, il recherche uniquement les ancrés.

Fin d'un RepeatingGroup

Vous pouvez signaler la fin d'un `RepeatingGroup` d'une des façons suivantes :

- Le `RepeatingGroup` peut continuer jusqu'à la fin du document.
- Vous pouvez insérer un `Marqueur` après le `RepeatingGroup`. Par défaut, le `Marqueur` est dans une phase de recherche antérieure à celle du `RepeatingGroup`. Cela force l'analyseur à rechercher tout d'abord le `Marqueur` et à l'utiliser pour limiter la zone de recherche du `RepeatingGroup`. Pour plus d'informations, veuillez consulter ["Ajustement de la phase de recherche" à la page 212](#).
- Vous pouvez définir la propriété `compteur` qui limite la recherche à un nombre maximum d'itérations.
- Si le `RepeatingGroup` n'a pas de `séparateur`, il se termine lorsque l'analyseur ne trouve plus aucune itération.

Réussite ou échec d'un RepeatingGroup

Si un élément **`RepeatingGroup`** ne trouve pas les ancrés non optionnelles dans une itération, celle-ci échoue.

Lorsqu'une itération échoue, **`RepeatingGroup`** peut se terminer, échouer ou ignorer l'itération en échec. Le comportement est le suivant :

- Si **`RepeatingGroup`** ne comporte pas de **`séparateur`**, **`RepeatingGroup`** se termine. A condition qu'il y ait eu au moins une itération réussie avant l'itération ayant échoué, **`RepeatingGroup`** réussit.
- Si **`RepeatingGroup`** comporte un **`séparateur`** et que la propriété **`skip_failed_iterations`** n'est pas sélectionnée, **`RepeatingGroup`** échoue.
- Si **`RepeatingGroup`** comporte un **`séparateur`** et que la propriété **`skip_failed_iterations`** est sélectionnée, le script ignore l'itération ayant échoué et passe à l'itération suivante. A condition qu'au moins une des itérations réussisse, **`RepeatingGroup`** réussit.

Journal des événements d'un groupe de répétition

Le journal des événements enregistre les événements pour chaque itération d'un **`RepeatingGroup`**.

Si la propriété **`skip_failed_iterations`** est sélectionnée, le **`RepeatingGroup`** peut générer un événement d'échec optionnel suite aux itérations exécutées avec succès. Un événement d'échec peut être imbriqué dans l'échec facultatif. Ces événements se produisent lorsque le **`RepeatingGroup`** ne peut pas trouver d'itérations supplémentaires à analyser. Les événements sont normaux et ne doivent pas être une cause de soucis.

Exemples en ligne

Pour obtenir un exemple en ligne de cette ancre, ouvrez le projet `samples\Projects\Dynamic_And_RepeatingGroup\Dynamic_And_RepeatingGroup.cmw`. L'exemple utilise un `RepeatingGroup` pour répéter les lignes d'un document.

Certaines lignes du document source contiennent une référence de note de page entre parenthèses, telle que "(1)". `RepeatingGroup` contient un `Groupe`, dont le rôle est d'analyser la note de bas de page et d'insérer son contenu dans la sortie.

Le `Groupe` contient une ancre `Contenu` qui extrait la référence de la note de bas de page et la stocke dans une variable. Le `Groupe` provoque ensuite une action `RunParser` qui active un analyseur secondaire. L'analyseur secondaire trouve la note de bas de page référencée par la variable, l'analyse et insère le résultat dans la sortie.

StructureDefinition

L'ancre **StructureDefinition** traite les entrées bien structurées, par exemple les messages texte conformes aux protocoles de messagerie standard du secteur. La sortie de **StructureDefinition** est une représentation XML des données.

Les données d'entrée comportent des enregistrements délimités. Vous organisez les enregistrements d'entrée selon des méthodes prédéfinies. Par exemple, un enregistrement de type A précède un enregistrement de type B, suivi d'un à trois enregistrements de type C. Chaque enregistrement contient un ensemble organisé de champs.

Le tableau suivant décrit les propriétés de l'ancre **StructureDefinition** :

Propriété	Description
désactivé	Détermine si le script ignore le composant et tous les composants enfants. Utilisez cette propriété pour tester, déboguer et modifier un script. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none">- Sélectionné. Le script ignore le composant.- Effacé. Le script applique le composant. La valeur par défaut est vide.
format_definition	Définit une liste d'ancres et d'actions qui identifient et extraient les enregistrements. La liste doit contenir une ancre ExtractRecord .
nom	Libellé descriptif pour le composant. Ce libellé s'affiche dans le fichier journal et la vue Événements . Utilisez la propriété nom pour identifier le composant ayant provoqué l'événement.
notifications	Liste de composants Notificationhandler qui gèrent les notifications de composants imbriqués. Pour plus d'informations, voir "Notifications" à la page 423
phase	Détermine les situations dans lesquelles le script traite le composant. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none">- initiale. Le script traite le composant pendant la phase initiale.- principale. Le script traite le composant pendant la phase principale.- finale. Le script traite le composant pendant la phase finale. Pour plus d'informations, voir "Comment un analyseur recherche-t-il les ancres ?" à la page 210 La valeur par défaut est Principale.
remarque	Commentaire défini par l'utilisateur, qui décrit l'objectif ou l'action du composant.
cible	Définit une zone de stockage des données dans laquelle le composant stocke son résultat.

Un élément **StructureDefinition** comprend les parties suivantes :

- propriété **format_definition**. Extrait les enregistrements et identifie leurs types.
- Une hiérarchie de composants enfant. Chaque composant enfant analyse les enregistrements de type particulier.

La propriété **format_definition** peut contenir un élément **RepeatingGroup** qui trouve les enregistrements. Dans l'élément **RepeatingGroup**, une ou plusieurs ancres **Contenu** récupèrent les identificateurs de type d'enregistrement. L'élément **RepeatingGroup** contient une ancre **ExtractRecord** qui transmet l'enregistrement à la liste de sous-éléments.

Remarque: Lorsqu'une transformation Bibliothèque possède une **StructureDefinition** avec une **format_definition** qui contient un **RepeatingGroup**, la transformation remplace l'élément répétitif au lieu d'effectuer une itération sur l'élément.

La hiérarchie des sous-éléments reflète l'organisation requise des enregistrements. Vous pouvez configurer des séquences, des choix, des boucles d'enregistrements et des enregistrements requis ou facultatifs.

Les sous-éléments reçoivent les enregistrements de **ExtractRecord**. Le système associe chaque enregistrement à un sous-élément selon les critères suivants :

- Les identificateurs \$id et \$qualifier de l'enregistrement doivent correspondre aux valeurs spécifiées dans le sous-élément.
- L'emplacement de l'enregistrement de l'entrée doit correspondre à celui du sous-élément dans la hiérarchie.

Le sous-élément correspondant analyse l'enregistrement.

Si aucun sous-élément ne correspond, **StructureDefinition** déclenche une notification. Vous pouvez insérer des composants **NotificationHandler** pour traiter la notification. La transformation utilise l'ancre **StructureDefinition** pour rechercher et diagnostiquer les erreurs d'entrée. Généralement, l'ancre **StructureDefinition** peut continuer à analyser le reste de l'entrée.

Exemple

Un analyseur traite l'entrée avec la structure suivante :

```
ST*12*23~
NM1*12*23*4~
N1*12*23~
NM1*13*23*4~
N2*1*2*3~
SE*12*2:3:4~
```

Les enregistrements sont délimités par des caractères retour à la ligne. Dans chaque enregistrement, les champs sont délimités par les caractères ~, * et :.

Le premier champ de chaque enregistrement identifie le type d'enregistrement. Il y a cinq types d'enregistrement principaux :

```
ST
NM1
N1
N2
SE
```

Dans les enregistrements NM1, le deuxième champ est un sous-type. Il y a deux sous-types :

```
NM1*12
NM1*13
```

Les enregistrements doivent se produire dans l'ordre suivant :

1. Un enregistrement ST.

2. Un enregistrement `NM1*12` suivi de `N1`.
3. Un enregistrement `NM1*13` suivi de `N2`.
4. Un enregistrement `SE`.

Vous pouvez analyser cette entrée en configurant une ancre `StructureDefinition`.

La propriété `format_definition` contient un `RepeatingGroup` qui trouve les enregistrements. `RepeatingGroup` effectue les opérations suivantes :

1. Il trouve le contenu de l'enregistrement, jusqu'au délimiteur retour à la ligne.
2. Il extrait l'identifiant de type enregistrement, tel que `ST` ou `NM1` et le stocke dans la variable `$id`.
3. Si le type d'enregistrement est `NM1`, il extrait le sous-type (`12` ou `13`) et le stocke dans la variable `$qualifier`.
4. Il exécute une ancre `ExtractRecord` qui transmet l'enregistrement aux sous-éléments. `ExtractRecord` attache les identifiants `$id` et `$qualifier` à l'enregistrement.

L'élément est configuré pour correspondre à n'importe quel enregistrement ayant l'identifiant `ST`.

`format_definition` rencontre le premier enregistrement d'entrée et le transmet aux sous-éléments. Le premier enregistrement correspond au premier sous-élément, car il possède l'identifiant `ST`. Le premier sous-élément contient des ancres `Contenu` `Marqueur` `Contenu` qui analysent l'enregistrement.

Le deuxième sous-élément définit une séquence de sous-éléments imbriqués. Le premier sous-élément imbriqué correspond à un enregistrement ayant les identifiants `NM1` et `12`. Le second sous-élément imbriqué correspond à un enregistrement ayant un identifiant `N1`.

Les deuxième et troisième enregistrements d'entrée sont `NM1*12` et `N1`. Ils correspondent à la séquence des sous-éléments. Chaque sous-élément imbriqué analyse l'enregistrement correspondant.

Imaginons que les deuxième et troisième enregistrements sont `NM1*12` et `N2`. Ils ne correspondent pas à la hiérarchie des sous-éléments et ne seront donc pas analysés.

Les enregistrements suivants sont `NM1*13` et `N2`. Ils correspondent au troisième sous-élément, nommé `Loop2000`.

Le dernier enregistrement est `SE`, correspondant au dernier sous-élément.

Tous les enregistrements d'entrée correspondent à la hiérarchie des sous-éléments. `StructureDefinition` analyse donc l'entrée complète avec succès.

Composants des sous-éléments

Dans la hiérarchie des sous-éléments, vous pouvez insérer les composants suivants :

Sous-élément	Description
<code>RecordStructureLocal</code>	Associe et analyse un seul enregistrement.
<code>SequenceStructureLocal</code>	Définit une séquence de sous-éléments imbriqués. Les enregistrements doivent se produire dans la même séquence que les sous-éléments imbriqués.
<code>ChoiceStructureLocal</code>	Définit un choix de sous-éléments imbriqués. Un enregistrement doit correspondre à l'un des sous-éléments imbriqués.
<code>AllStructureLocal</code>	Définit un ensemble de sous-éléments imbriqués, sans séquence spécifiée. Les enregistrements peuvent correspondre aux sous-éléments imbriqués dans n'importe quel ordre.

Les noms se terminent par `Local` car vous pouvez les configurer dans des emplacements imbriqués, non-globaux du script. Il existe un ensemble correspondant de composants appelés `RecordStructure`, `SequenceStructure`, etc, sans le suffixe `Local`. Vous pouvez configurer ces derniers au niveau global du script et les référencer selon vos besoins. Pour référencer les composants globaux, insérez un sous-élément `EmbeddedStructure`.

La liste des sous-éléments de niveau supérieur d'une `StructureDefinition` est équivalente à celle de `SequenceStructureLocal`. Les enregistrements doivent se produire lors de la même séquence que les sous-éléments de niveau supérieur.

Par défaut, chaque sous-élément doit se produire exactement une fois. Pour modifier la valeur par défaut, définissez les propriétés `minOccurs` et `maxOccurs` du sous-élément. Par exemple, s'il arrive qu'un sous-élément soit manquant ou se produise jusqu'à 3 fois, définissez `minOccurs = 0` et `maxOccurs = 3`. Pour autoriser des occurrences illimitées, définissez `maxOccurs = -1`.

Pour plus d'informations sur les composants des sous-éléments, consultez le ["Documentation de référence des sous-composants d'ancre" à la page 250](#).

Notifications

Si un enregistrement ou un ensemble d'enregistrements ne correspond pas à la hiérarchie des sous-éléments, **StructureDefinition** déclenche une notification.

Le tableau suivant décrit les types de notification :

Notification	Description
MandatoryStructureMissing	Un enregistrement obligatoire n'apparaît pas dans l'entrée.
MismatchIDs	L'enregistrement et les identifiants du sous-élément correspondent partiellement. Par exemple, il y a deux identifiants d'enregistrement et seulement un des deux correspond.
StructureBelowMinOccurs	Il y a moins d'enregistrements du sous-élément qui correspondent que le nombre défini dans minOccurs .
StructureExceedsMaxOccurs	Il y a plus d'enregistrements du sous-élément qui correspondent que le nombre défini dans maxOccurs .
StructureOutOfSequence	Les enregistrements correspondent aux sous-éléments, mais pas dans l'ordre requis. Par exemple, les sous-éléments définissent une séquence ABC, mais l'entrée contient ACB.
UnexpectedRecord	Les enregistrements correspondent aux sous-éléments, mais pas dans la hiérarchie requise. Par exemple, le sous-élément définit une séquence ABC et D définie dans un autre emplacement. L'entrée contient ABD.
UnrecognizedRecord	Aucun sous-élément ne correspond à un identifiant de l'enregistrement.
XsdValidationError	L'entrée ne correspond pas aux exigences du schéma.

Configurez les composants **NotificationHandler** dans la propriété **notifications** de **StructureDefinition** ou un sous-élément. Vous pouvez également configurer des gestionnaires dans la propriété **notifications** d'un composant de plus haut niveau tel qu'un **Analyseur** ou un **Groupe** qui contient la **StructureDefinition**. Si un gestionnaire existe dans le sous-élément dans lequel une discordance se produit, il traite la notification. Si aucun gestionnaire n'existe, la notification gravite la hiérarchie IntelliScript jusqu'à ce qu'un gestionnaire le

traite. S'il n'existe aucun gestionnaire pour une notification, la notification est ignorée et la **StructureDefinition** poursuit le traitement de l'entrée.

Suivi de la progression

Comme `format_definition` extrait des enregistrements, il met à jour la variable système `VarStructureDetails`. Vous pouvez utiliser la variable dans les notifications. Par exemple, pour faire état de l'identifiant de l'enregistrement, un gestionnaire de notification peut insérer `VarStructureDetails/RecordId` dans sa sortie.

Pour plus d'informations, consultez la section ["Variables système" à la page 198](#)

Référence du composant de recherche

Les composants de recherche sont utilisés pour les objectifs suivants :

- Pour définir l'emplacement des ancres. Pour plus d'informations, voir ["Documentation de référence du composant Ancre" à la page 217](#)
- Pour définir des caractères ou des chaînes séparateurs. Pour plus d'informations, voir ["Documentation de référence du composant Format" à la page 179](#)
- Pour définir la chaîne `find_what` d'un transformer **Remplacer**. Pour plus d'informations, voir ["Documentation de référence du composant transformer" à la page 262](#)

AttributeSearch

Le composant chercheur **AttributeSearch** recherche la valeur d'un attribut spécifié dans un document source. Le composant récupère la valeur depuis une expression dans l'un des formats suivants :

- `AttributeName = value`
- `AttributeName = "value"`

où `AttributeName` est le nom de l'attribut, les guillemets peuvent être uniques ou doubles et les espaces sont facultatifs.

AttributeSearch est un des paramètres de la propriété **valeur** de l'ancre **Contenu**. Pour plus d'informations, consultez la section ["Contenu" à la page 219](#).

Le tableau suivant décrit les propriétés du composant chercheur **AttributeSearch** :

Propriété	Description
<code>att</code>	Définit le nom de l'attribut.
<code>match_case</code>	Détermine si le nom de l'attribut est sensible à la casse. La propriété match_case possède les options suivantes : <ul style="list-style-type: none">- Sélectionné. Le nom de l'attribut est sensible à la casse.- Effacé. Le nom de l'attribut n'est pas sensible à la casse.

Exemple

Un document HTML contient l'élément :

```
<img src='MyPicture.gif'>
```

Vous pouvez utiliser `AttributeSearch` pour extraire la valeur de l'attribut `src`. Il renvoie le texte `MyPicture.gif`.

Syntaxe d'attribut valide

AttributeSearch lit les paires nom-valeur qui contiennent un signe d'égalité. Le signe d'égalité peut être entouré d'espaces. La valeur peut être entourée de guillemets doubles ou simples, ou sans guillemets.

Par exemple, supposons que **AttributeSearch** soit configuré pour rechercher un attribut appelé `time`. Tous les exemples suivants comportent une syntaxe valide et renvoient la même valeur, `12:55:33`.

```
time = "12:55:33"
time="12:55:33"
time = '12:55:33'
time='12:55:33'
time = 12:55:33
time=12:55:33
```

Exemple en ligne

Pour obtenir un échantillon en ligne de ce composant, ouvrez le projet `samples\Projects\Content\Content.cmw`. L'exemple illustre l'utilisation de `AttributeSearch` pour analyser un document texte qui a une structure `variable = valeur`.

LearnByExample

Le composant de recherche **LearnByExample** apprend comment rechercher un texte en examinant les emplacements du texte dans l'échantillon de document source. Il utilise le format de l'analyseur pour interpréter le document source.

Par exemple, si l'analyseur a un format délimité par tabulations, **LearnByExample** compte le nombre de tabulations entre le début de la recherche et l'exemple de texte. Il recherche dans le document source le texte se trouvant au même nombre de tabulations comptées depuis le début de l'étendue de la recherche.

LearnByExample est un des paramètres de la propriété **valeur** de l'ancre **Contenu**. Pour plus d'informations, voir ["Contenu" à la page 219](#)

Si l'attribut **direction** de l'ancre **Contenu** est défini sur `en arrière`, le composant compte les délimiteurs depuis la fin de la portée de recherche.

Le tableau suivant décrit les propriétés du composant de recherche **LearnByExample** :

Propriété	Description
exemple	Définit le texte dans l'échantillon de document source à l'emplacement de l'ancre.

Remarque: Le composant du programme de recherche **LearnByExample** applique des heuristiques sensibles. Pour utiliser le composant du programme de recherche **LearnByExample**, l'exemple doit avoir le même format que l'entrée attendue. Le format doit être uniforme dans tous les fichiers d'entrée. Une différence de format peut entraîner l'échec du composant du programme de recherche **LearnByExample**.

NewlineSearch

Le composant chercheur **NewlineSearch** recherche un caractère retour à la ligne ou saut de ligne (0x0A), un caractère retour chariot (0x0D) ou les deux.

L'ancre **Marqueur** peut utiliser **NewlineSearch** pour trouver des marqueurs de retour à la ligne. Un composant **Délimiteur** peut utiliser **NewlineSearch** pour trouver des délimiteurs retour à la ligne.

OffsetSearch

Le composant de recherche **OffsetSearch** définit le nombre de caractères entre un point de référence et une ancre. Par exemple, il peut définir le nombre de caractères entre la fin d'un **Marqueur** et le début d'une ancre **Contenu**.

Le tableau suivant décrit les propriétés du composant de recherche **OffsetSearch** :

Propriété	Description
allow_smaller_offset	Détermine si un décalage qui s'étend au-delà de la portée de la recherche est valide. Sélectionnez cette propriété pour autoriser une taille de champ tronquée à la fin d'un document. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none">- Sélectionné. OffsetSearch réussit quand un décalage s'étend au-delà de la portée de la recherche.- Effacé. OffsetSearch échoue quand un décalage s'étend au-delà de la portée de la recherche.
décalage	Définit le nombre de caractères entre le point de référence et l'ancre. Dans certains emplacements où OffsetSearch est utilisé, comme dans une ancre Marqueur , l'éditeur IntelliScript affiche un bouton parcourir à côté de la propriété décalage . Vous pouvez entrer une valeur ou accéder à une zone de stockage des données contenant la valeur.

PatternSearch

Le composant de recherche **PatternSearch** recherche une chaîne qui correspond à une expression régulière.

Les ancres peuvent utiliser **PatternSearch** pour trouver des marqueurs ou des contenus. Un composant **Délimiteur** peut utiliser **PatternSearch** pour trouver des délimiteurs. Le transformateur **Replace** peut utiliser **PatternSearch** pour trouver le texte à remplacer.

Le tableau suivant décrit les propriétés du composant de recherche **PatternSearch**.

Propriété	Description
escape_sequence	Définit un préfixe qui contraint le composant de recherche à ignorer une instance du modèle dans le document source.
modèle	Définit l'expression régulière.

Pour plus d'informations sur la syntaxe des expressions régulières, voir ["Syntaxe d'expression régulière" à la page 284](#).

Exemple

Imaginons que vous vouliez définir comme délimiteur la chaîne `%%%`, contenant un ou plusieurs symboles `%`. Dans le composant **Délimiteur**, vous pouvez utiliser **PatternSearch** avec l'expression régulière suivante :

`%+`

Dans un autre exemple, supposons que vous vouliez définir la virgule ou le point virgule comme délimiteurs, au même niveau dans la hiérarchie des délimiteurs. Vous pouvez utiliser l'expression régulière suivante :

[,;]

SegmentSearch

Le composant de recherche **SegmentSearch** recherche les marqueurs d'ouverture et de fermeture dans une chaîne de texte. Il renvoie le segment du marqueur d'ouverture au marqueur de fermeture, y compris les marqueurs eux-mêmes. **SegmentSearch** est l'une des options pour l'attribut **find_what** du transformer **Remplacer**.

Le tableau suivant décrit les propriétés du composant de recherche **SegmentSearch** :

Propriété	Description
opening	Définit le critère de recherche pour le marqueur d'ouverture. Les options sont les composants de recherche suivants : <ul style="list-style-type: none">- NewlineSearch- OffsetSearch- PatternSearch- TextSearch
closing	Définit le critère de recherche pour le marqueur de fermeture. Les options sont les composants de recherche suivants : <ul style="list-style-type: none">- NewlineSearch- OffsetSearch- PatternSearch- TextSearch

TextSearch

Le composant de recherche **TextSearch** recherche une chaîne explicite.

Les ancres peuvent utiliser **TextSearch** pour trouver des marqueurs. Le composant **Delimiter** peut utiliser **TextSearch** pour trouver des délimiteurs. Le transformer **Replace** peut utiliser **TextSearch** pour trouver le texte à remplacer.

Le tableau suivant décrit les propriétés du composant de recherche **TextSearch** :

Propriétés	Description
escape_sequence	Définit un préfixe pour que la recherche ignore une instance de la chaîne dans le document source. Dans les emplacements prenant en charge la recherche dynamique, vous pouvez accéder à une zone de stockage des données qui contient la séquence d'échappement.
match_case	Détermine si le texte défini doit correspondre exactement, avec les mêmes lettres majuscules et minuscules. Par défaut, la valeur est vide.
text	Définit la chaîne à rechercher. Dans les emplacements prenant en charge la recherche dynamique, vous pouvez accéder à une zone de stockage des données qui contient la chaîne.

Exemple

Pour définir la chaîne pourcent-pourcent-tabulation en tant que délimiteur, créez un composant `Délimiteur` et définissez sa propriété `search` avec `TextSearch`. Dans la propriété `texte`, saisissez :

%%

Pressez ensuite `CTRL+A` et saisissez `009` (le code ASCII d'un caractère de tabulation).

Spécifier dynamiquement une chaîne à rechercher

Dans certains emplacements où `TextSearch` est utilisé, comme dans un composant `Délimiteur` ou une ancre `Marqueur`, un bouton `Parcourir` apparaît à droite de la zone de texte. Accédez à une zone de stockage des données qui contient le texte de recherche.

Pour rechercher des instances répétées du premier mot dans un document, vous pouvez définir une ancre `Contenu` qui récupère le premier mot et le stocke dans une variable. Vous pouvez ensuite définir des ancres `Marqueur` qui utilisent `TextSearch` pour trouver d'autres instances du mot que vous avez stocké dans la variable.

Exemple en ligne

Pour obtenir un échantillon en ligne de ce composant, ouvrez le projet `samples\Projects\Dynamic_And_RepeatingGroup\Dynamic_And_RepeatingGroup.cmw`.

Dans le composant `GetRemarkParser` de cet exemple, une ancre `Marqueur` utilise un `TextSearch` défini dynamiquement pour rechercher une note de pied de page à la fin du document source. Pour plus d'informations sur cet exemple, consultez la section [“RepeatingGroup” à la page 237](#).

TypeSearch

Le composant de recherche **TypeSearch** recherche une ancre appartenant à un type de données spécifié.

TypeSearch est l'un des paramètres de la propriété **value** de l'ancre de **contenu**. Pour plus d'informations, voir [“Contenu” à la page 219](#)

Le tableau suivant décrit les propriétés du composant de recherche **TypeSearch** :

Propriété	Description
<code>val_type</code>	Détermine le type de données de l'ancre à rechercher.

Documentation de référence des sous-composants d'ancre

Les sous-composants d'ancre sont affectés en tant que valeurs de certaines propriétés d'ancre.

AllStructure

Le composant **AllStructure** définit un jeu de sous-éléments imbriqués sans tenir compte de leur ordre. Un jeu d'enregistrements correspond à **AllStructure** s'il correspond à tous les sous-éléments dans n'importe quel ordre. Pour plus d'informations, consultez la section ["StructureDefinition" à la page 242](#).

Le composant **AllStructure** apparaît au niveau global du script et possède la même fonction que **AllStructureLocal**. Vous pouvez le référencer dans l'attribut **ref** de **EmbeddedStructure**.

Le tableau suivant décrit les propriétés du composant **AllStructure** :

Propriété	Description
action	Définit une action qui s'exécute sur la liste des sous-composants.
désactivé	Détermine si le script ignore le composant et tous les composants enfants. Utilisez cette propriété pour tester, déboguer et modifier un script. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none">- Sélectionné. Le script ignore le composant.- Effacé. Le script applique le composant. La valeur par défaut est vide.
nom	Libellé descriptif pour le composant. Ce libellé s'affiche dans le fichier journal et la vue Événements . Utilisez la propriété nom pour identifier le composant ayant provoqué l'événement.
notifications	Liste de composants Notificationhandler qui gèrent les notifications de composants imbriqués. Pour plus d'informations, voir "Notifications" à la page 423
remarque	Commentaire défini par l'utilisateur, qui décrit l'objectif ou l'action du composant.
cible	Définit une zone de stockage des données dans laquelle le composant stocke son résultat.

AllStructureLocal

Le composant **AllStructureLocal** définit un ensemble de sous-éléments imbriqués sans tenir compte de l'ordre. Un jeu d'enregistrements correspond à **AllStructureLocal** s'il correspond à tous les sous-éléments dans n'importe quel ordre. Pour plus d'informations, consultez la section ["StructureDefinition" à la page 242](#).

AllStructureLocal est un sous-élément de l'ancre **StructureDefinition** et a la même fonction que **AllStructure**. Au niveau global du script, utilisez **AllStructure**.

Le tableau suivant décrit les propriétés du composant **AllStructure** :

Propriété	Description
action	Définit une action qui s'exécute sur la liste des sous-composants.
désactivé	Détermine si le script ignore le composant et tous les composants enfants. Utilisez cette propriété pour tester, déboguer et modifier un script. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none">- Sélectionné. Le script ignore le composant.- Effacé. Le script applique le composant. La valeur par défaut est vide.
minOccurs	Définit le nombre minimum d'enregistrements concordants. La valeur par défaut est 1.

Propriété	Description
maxOccurs	Définit le nombre maximum d'enregistrements concordants. La valeur par défaut est 1. Utilisez -1 pour un nombre illimité.
nom	Libellé descriptif pour le composant. Ce libellé s'affiche dans le fichier journal et la vue Événements . Utilisez la propriété nom pour identifier le composant ayant provoqué l'événement.
notifications	Liste de composants Notificationhandler qui gèrent les notifications de composants imbriqués. Pour plus d'informations, voir "Notifications" à la page 423
remarque	Commentaire défini par l'utilisateur, qui décrit l'objectif ou l'action du composant.
sub_elements	Définit une liste de sous-éléments imbriqués.
cible	Définit une zone de stockage des données dans laquelle le composant stocke son résultat.

ChoiceStructure

Le composant **ChoiceStructure** définit un ensemble de sous-éléments imbriqués. Un enregistrement correspond à **ChoiceStructure** s'il correspond à tout sous-élément imbriqué. Pour plus d'informations, consultez la section ["StructureDefinition" à la page 242](#).

Le composant **ChoiceStructure** apparaît au niveau global du script et a la même fonction que **ChoiceStructureLocal**. Vous pouvez le référencer dans l'attribut **ref** de **EmbeddedStructure**.

Le tableau suivant décrit les propriétés du composant **ChoiceStructure** :

Propriété	Description
action	Définit une action qui s'exécute sur la liste des sous-composants.
désactivé	Détermine si le script ignore le composant et tous les composants enfants. Utilisez cette propriété pour tester, déboguer et modifier un script. Vous pouvez choisir l'une des options suivantes : - Sélectionné. Le script ignore le composant. - Effacé. Le script applique le composant. La valeur par défaut est vide.
nom	Libellé descriptif pour le composant. Ce libellé s'affiche dans le fichier journal et la vue Événements . Utilisez la propriété nom pour identifier le composant ayant provoqué l'événement.
notifications	Liste de composants Notificationhandler qui gèrent les notifications de composants imbriqués. Pour plus d'informations, voir "Notifications" à la page 423
remarque	Commentaire défini par l'utilisateur, qui décrit l'objectif ou l'action du composant.
cible	Définit une zone de stockage des données dans laquelle le composant stocke son résultat.

ChoiceStructureLocal

Le composant **ChoiceStructureLocal** définit un jeu de sous-éléments imbriqués. Un enregistrement correspond à **ChoiceStructureLocal** s'il correspond à n'importe quel sous-élément imbriqué. Pour plus d'informations, consultez la section ["StructureDefinition" à la page 242](#).

ChoiceStructureLocal est un sous-élément de l'ancre **StructureDefinition** et a la même fonction que **ChoiceStructure**. Au niveau global du script, utilisez **ChoiceStructure**.

Le tableau suivant décrit les propriétés du composant **ChoiceStructureLocal** :

Propriété	Description
action	Définit une action qui s'exécute sur la liste des sous-composants.
désactivé	Détermine si le script ignore le composant et tous les composants enfants. Utilisez cette propriété pour tester, déboguer et modifier un script. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none">- Sélectionné. Le script ignore le composant.- Effacé. Le script applique le composant. La valeur par défaut est vide.
minOccurs	Définit le nombre minimum d'enregistrements concordants. La valeur par défaut est 1.
maxOccurs	Définit le nombre maximum d'enregistrements concordants. La valeur par défaut est 1. Utilisez -1 pour un nombre illimité.
nom	Libellé descriptif pour le composant. Ce libellé s'affiche dans le fichier journal et la vue Événements . Utilisez la propriété nom pour identifier le composant ayant provoqué l'événement.
notifications	Liste de composants Notificationhandler qui gèrent les notifications de composants imbriqués. Pour plus d'informations, voir "Notifications" à la page 423
remarque	Commentaire défini par l'utilisateur, qui décrit l'objectif ou l'action du composant.
sub_elements	Définit une liste de sous-éléments imbriqués.
cible	Définit une zone de stockage des données dans laquelle le composant stocke son résultat.

Connecter

Le composant **Connecter** spécifie un lien entre les zones de stockage des données de deux composants. Les deux zones de stockage des données doivent être du même type de données

Le tableau suivant décrit les propriétés du composant **Connecter** :

Propriété	Description
data_holder	Définit une zone de stockage des données référencée dans l'analyseur, le sérialiseur ou le mappeur principal. Remarque: Si la zone de stockage des données est une variable, la transformation lui affecte une valeur par défaut vide. Si le type de données de la variable n'accepte pas de valeur vide, par exemple <code>xs:boolean</code> , assurez-vous que la variable a une valeur avant d'exécuter la transformation intégrée.
embedded_data_holder	Définit une zone de stockage des données référencée dans l'analyseur, le sérialiseur ou le mappeur secondaire.
nom	Libellé descriptif pour le composant. Ce libellé s'affiche dans le fichier journal et la vue Événements . Utilisez la propriété nom pour identifier le composant ayant provoqué l'événement.
remarque	Commentaire défini par l'utilisateur, qui décrit l'objectif ou l'action du composant.

La propriété **schema_connections** des composants suivants peut avoir une ou plusieurs instances du composant **Connecter** :

- EmbeddedParser. Spécifie l'emplacement où un analyseur secondaire stocke ses résultats dans la sortie de l'analyseur principal.
- EmbeddedSerializer. Spécifie un lien entre les zones de stockage des données d'entrée d'un sérialiseur secondaire et les zones de stockage des données d'entrée du sérialiseur principal.
- EmbeddedMapper. Spécifie un lien entre les zones de stockage des données d'entrée et de sortie.
- EmbeddedStructure. Spécifie un lien entre les cibles des sous-éléments **StructureDefinition** globaux et locaux.

Exemple

Un analyseur secondaire produit un élément XML appelé `ID`. Vous voulez que l'analyseur principal stocke ce résultat dans une variable appelée `VarID`. Vous pouvez connecter `ID` à `VarID`.

Pour un autre exemple, voir ["EmbeddedSerializer" à la page 348](#).

EmbeddedStructure

Le composant **EmbeddedStructure** active les composants définis au niveau global du script. Pour plus d'informations, consultez la section ["StructureDefinition" à la page 242](#)

EmbeddedStructure est un sous-élément de l'ancre **StructureDefinition**.

Le tableau suivant décrit les propriétés du composant **EmbeddedStructure** :

Propriété	Description
désactivé	Détermine si le script ignore le composant et tous les composants enfants. Utilisez cette propriété pour tester, déboguer et modifier un script. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none"> - Sélectionné. Le script ignore le composant. - Effacé. Le script applique le composant. La valeur par défaut est vide.
minOccurs	Définit le nombre minimum d'enregistrements concordants.
maxOccurs	Définit le nombre maximum d'enregistrements concordants. Utilisez -1 pour un nombre illimité.
nom	Libellé descriptif pour le composant. Ce libellé s'affiche dans le fichier journal et la vue Événements . Utilisez la propriété nom pour identifier le composant ayant provoqué l'événement.
notifications	Liste de composants Notificationhandler qui gèrent les notifications de composants imbriqués. Pour plus d'informations, voir "Notifications" à la page 423
ref	Définit le nom du composant configuré de manière globale.
remarque	Commentaire défini par l'utilisateur, qui décrit l'objectif ou l'action du composant.
schema_connection s	Connecte la cible du sous-élément références à la cible de EmbeddedStructure . Pour plus d'informations, consultez la section "Connecter" à la page 253 .
cible	Définit une zone de stockage des données dans laquelle le composant stocke son résultat.

RecordStructure

Le composant **RecordStructure** définit un ensemble de composants. Un ensemble d'enregistrements correspond à **RecordStructure** s'il a les mêmes identifiants. Pour plus d'informations, consultez la section ["StructureDefinition" à la page 242](#).

Le composant **RecordStructure** apparaît au niveau global du script et a la même fonction que **RecordStructureLocal**. Vous pouvez le référencer dans l'attribut **ref** de **EmbeddedStructure**.

Le tableau suivant décrit les propriétés du composant **RecordStructure** :

Propriété	Description
désactivé	Détermine si le script ignore le composant et tous les composants enfants. Utilisez cette propriété pour tester, déboguer et modifier un script. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none"> - Sélectionné. Le script ignore le composant. - Effacé. Le script applique le composant. La valeur par défaut est vide.
ids	Définit une ou plusieurs chaînes.
nom	Libellé descriptif pour le composant. Ce libellé s'affiche dans le fichier journal et la vue Événements . Utilisez la propriété nom pour identifier le composant ayant provoqué l'événement.

Propriété	Description
notifications	Liste de composants Notificationhandler qui gèrent les notifications de composants imbriqués. Pour plus d'informations, voir "Notifications" à la page 423
remarque	Commentaire défini par l'utilisateur, qui décrit l'objectif ou l'action du composant.
cible	Définit une zone de stockage des données dans laquelle le composant stocke son résultat.

RecordStructureLocal

Le composant **RecordStructureLocal** définit un ensemble de composants. Un ensemble d'enregistrements correspond à **RecordStructureLocal** s'il a les mêmes identifiants. Pour plus d'informations, consultez la section ["StructureDefinition" à la page 242](#).

RecordStructureLocal est un sous-élément de l'ancre **StructureDefinition** et a le même fonction que **RecordStructure**. Au niveau global du script, utilisez **RecordStructure**.

Le tableau suivant décrit les propriétés du composant **RecordStructureLocal** :

Propriété	Description
désactivé	Détermine si le script ignore le composant et tous les composants enfants. Utilisez cette propriété pour tester, déboguer et modifier un script. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none"> - Sélectionné. Le script ignore le composant. - Effacé. Le script applique le composant. La valeur par défaut est vide.
ids	Définit une ou plusieurs chaînes.
minOccurs	Définit le nombre minimum d'enregistrements concordants. La valeur par défaut est 1.
maxOccurs	Définit le nombre maximum d'enregistrements concordants. La valeur par défaut est 1. Utilisez -1 pour un nombre illimité.
nom	Libellé descriptif pour le composant. Ce libellé s'affiche dans le fichier journal et la vue Événements . Utilisez la propriété nom pour identifier le composant ayant provoqué l'événement.
notifications	Liste de composants Notificationhandler qui gèrent les notifications de composants imbriqués. Pour plus d'informations, voir "Notifications" à la page 423
remarque	Commentaire défini par l'utilisateur, qui décrit l'objectif ou l'action du composant.
cible	Définit une zone de stockage des données dans laquelle le composant stocke son résultat.

SequenceStructure

Le composant **SequenceStructure** définit une séquence de sous-éléments imbriqués. Un ensemble d'enregistrements correspond à **SequenceStructure** s'il correspond à tous les sous-éléments imbriqués dans la séquence. Pour plus d'informations, consultez la section ["StructureDefinition" à la page 242](#).

Le composant **SequenceStructure** apparaît au niveau global du script et a la même fonction que **SequenceStructureLocal**. Vous pouvez le référencer dans l'attribut **ref** de **EmbeddedStructure**.

Le tableau suivant décrit les propriétés du composant **SequenceStructure** :

Propriété	Description
action	Définit une action qui s'exécute sur la liste des sous-composants.
désactivé	Détermine si le script ignore le composant et tous les composants enfants. Utilisez cette propriété pour tester, déboguer et modifier un script. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none"> - Sélectionné. Le script ignore le composant. - Effacé. Le script applique le composant. La valeur par défaut est vide.
nom	Libellé descriptif pour le composant. Ce libellé s'affiche dans le fichier journal et la vue Événements . Utilisez la propriété nom pour identifier le composant ayant provoqué l'événement.
notifications	Liste de composants Notificationhandler qui gèrent les notifications de composants imbriqués. Pour plus d'informations, voir "Notifications" à la page 423
remarque	Commentaire défini par l'utilisateur, qui décrit l'objectif ou l'action du composant.
cible	Définit une zone de stockage des données dans laquelle le composant stocke son résultat.

SequenceStructureLocal

Le composant **SequenceStructureLocal** définit une séquence de sous-éléments imbriqués. Un ensemble d'enregistrements correspond à **SequenceStructureLocal** s'il correspond à tous les sous-éléments imbriqués dans la séquence. Pour plus d'informations, consultez la section ["StructureDefinition" à la page 242](#).

SequenceStructureLocal est un sous-élément de l'ancre **StructureDefinition** et a la même fonction que **SequenceStructure**. Au niveau global du script, utilisez **SequenceStructure**.

Le tableau suivant décrit les propriétés du composant **SequenceStructureLocal** :

Propriété	Description
action	Définit une action qui s'exécute sur la liste des sous-composants.
désactivé	Détermine si le script ignore le composant et tous les composants enfants. Utilisez cette propriété pour tester, déboguer et modifier un script. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none"> - Sélectionné. Le script ignore le composant. - Effacé. Le script applique le composant. La valeur par défaut est vide.
minOccurs	Définit le nombre minimum d'enregistrements concordants. La valeur par défaut est 1.
maxOccurs	Définit le nombre maximum d'enregistrements concordants. La valeur par défaut est 1. Utilisez -1 pour un nombre illimité.

Propriété	Description
nom	Libellé descriptif pour le composant. Ce libellé s'affiche dans le fichier journal et la vue Événements . Utilisez la propriété nom pour identifier le composant ayant provoqué l'événement.
notifications	Liste de composants Notificationhandler qui gèrent les notifications de composants imbriqués. Pour plus d'informations, voir " Notifications " à la page 423
remarque	Commentaire défini par l'utilisateur, qui décrit l'objectif ou l'action du composant.
sub_elements	Définit une liste de sous-éléments imbriqués.
cible	Définit une zone de stockage des données dans laquelle le composant stocke son résultat.

CHAPITRE 17

Transformateurs

Ce chapitre comprend les rubriques suivantes :

- [Présentation des transformateurs, 259](#)
- [Définition des transformateurs, 259](#)
- [Propriétés d'un transformateur standard, 261](#)
- [Documentation de référence du composant transformer, 262](#)

Présentation des transformateurs

Les transformateurs modifient la sortie d'autres composants.

Vous pouvez utiliser les transformateurs dans des composants tels que des ancres, des ancres de sérialisation, et des actions. Par exemple, si vous utilisez un transformateur dans une ancre `Contenu`, il modifie les données que l'ancre extrait du document source.

Vous pouvez utiliser les transformateurs comme processeurs de documents. Vous pouvez également définir un transformateur au niveau global d'un script et le configurer comme composant de démarrage.

Définition des transformateurs

Vous pouvez définir des transformateurs dans les emplacements suivants du script :

- Dans la propriété **transformateurs** d'une ancre ou d'une ancre de sérialisation
- Dans la propriété **default_transformers** d'un format ou d'un sérialiseur
- Dans le processeur de document **ProcessByTransformers**
- Dans la propriété **transformateurs** de certaines actions
- Au niveau global, en tant que composant autonome, exécutable, qui modifie un document source.

Utiliser des transformateurs dans les ancres

Vous pouvez utiliser des transformateurs dans des ancres qui créent des sorties XML, comme `Contenu`. Dans le script, imbriquez les composants du transformateur dans la propriété **transformateurs** de l'ancre.

L'entrée du transformateur est dans la sortie brute de l'ancre, avant que l'ancre n'insère la sortie dans une zone de stockage des données.

Supposez par exemple que vous analysiez le document source suivant :

```
First name: Ron
Last name: Lehrer
```

Vous désirez créer une sortie XML en ALL CAPS, comme ceci :

```
<Person>
  <FirstName>RON</FirstName>
  <LastName>LEHRER</LastName>
</Person>
```

Pour cela, vous pouvez configurer les ancrs Contenu, qui récupèrent les chaînes Ron et Lehrer, avec le transformateur `ChangeCase`.

Séquences de transformers

Vous pouvez configurer une ancre avec une séquence de transformers. Chaque transformer modifie la sortie du transformer précédent.

Dans l'exemple Ron Lehrer, supposez que vous souhaitiez obtenir la sortie suivante :

```
<Person>
  <FirstName>- RON -</FirstName>
  <LastName>- LEHRER -</LastName>
</Person>
```

Pour ce faire, vous pouvez configurer les ancrs Contenu à l'aide des transformers `ChangeCase` et `Addstring`. Les transformers modifient la casse et ajoutent les traits d'union dans la séquence.

Transformateurs par défaut

Bien souvent, vous souhaitez que le même transformateur s'exécute sur toutes les ancrs **Contenu** d'un analyseur. Vous pouvez configurer le composant de format de l'analyseur avec les transformateurs par défaut. Cela vous évite d'avoir à ajouter les mêmes transformateurs à chaque ancre de l'analyseur.

Pour cela, imbriquez les transformateurs dans la propriété **default_transformers** du format. Pour plus d'informations, consultez la section [“Documentation de référence du composant Format” à la page 179](#).

De nombreux composants du format prédéfini incluent des transformateurs par défaut. Par exemple, le composant **HtmlFormat** a des transformateurs par défaut qui suppriment les balises HTML de la sortie et convertissent les entités HTML en texte brut. Vous pouvez changer les transformateurs par défaut en éditant la propriété **default_transformers**.

Si une ancre possède ses propres transformateurs, ils s'exécutent après les transformateurs par défaut.

Vous pouvez annuler les transformateurs par défaut pour des ancrs particulières. Pour cela, paramétrez la propriété **ignore_default_transformers** de l'ancre.

Utiliser les transformers comme processeurs de document

Vous pouvez exécuter un transformer ou une séquence de transformers comme processeur de document.

Par exemple, vous pouvez exécuter le transformer `RemoveTags` comme processeur d'un document HTML. Le transformateur retire les balises HTML avant qu'un analyseur commence à rechercher des ancrs dans le document.

Pour cela, configurez le composant de format de l'analyseur avec le processeur de document `ProcessByTransformers` et imbriquez les transformateurs dans le composant.

Utiliser des transformers dans les ancrs de sérialisation

Vous pouvez utiliser des transformers dans les ancrs de sérialisation qui écrivent dans le document de sortie, comme par exemple `ContentSerializer`. Le transformer modifie les données avant que le sérialiseur ne les écrive dans le document.

Par exemple, `ContentSerializer` peut écrire le contenu d'une zone de stockage des données appelée `DoctorName` dans un document de sortie. Vous pouvez configurer le `ContentSerializer` avec un transformer `AddString` qui ajoute le préfixe "Dr. " au contenu. Supposons que le XML d'entrée présente la forme suivante :

```
<DoctorName>Albert Schweitzer</DoctorName>
```

Le transformer modifie le contenu, aboutissant à la sortie suivante :

```
Dr. Albert Schweitzer
```

Vous pouvez ajouter des transformers à la propriété `default_transformers` d'un sérialiseur. Le transformer que vous ajoutez ici s'exécute dans toutes les ancrs de sérialisation `ContentSerializer` avant qu'elles n'écrivent dans le document de sortie.

Utiliser des transformers dans les actions

Certaines actions, comme par exemple **SetValue** et **Map**, appliquent les transformers à leurs sortie. Pour plus d'informations, consultez la section ["Présentation des actions" à la page 297](#).

Propriétés d'un transformateur standard

Le tableau suivant décrit les propriétés standards des transformateurs :

Propriété	Définition
nom	Libellé descriptif pour le composant. Ce libellé s'affiche dans le fichier journal et la vue Événements . Utilisez la propriété nom pour identifier le composant ayant provoqué l'événement.
remarque	Commentaire défini par l'utilisateur, qui décrit l'objectif ou l'action du composant.
désactivé	Détermine si le script ignore le composant et tous les composants enfants. Utilisez cette propriété pour tester, déboguer et modifier un script. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none">- Sélectionné. Le script ignore le composant.- Effacé. Le script applique le composant. La valeur par défaut est vide.
facultatif	Détermine si la défaillance d'un composant entraîne l'échec du composant parent. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none">- Sélectionné. La défaillance du composant n'entraîne pas l'échec du composant parent.- Effacé. La défaillance du composant entraîne l'échec du composant parent. Par défaut, la valeur est vide. Pour plus d'informations sur l'échec des composants, consultez la section "Traitement des échecs" à la page 404 .

Documentation de référence du composant transformer

Les transformers modifient des données.

AbsURL

Le transformateur **AbsURL** convertit l'URL ou le chemin relatif d'un fichier en chemin absolu.

Par exemple, si l'entrée est `test.html` et l'URL de base est `http://www.example.com`, la sortie est `http://www.example.com/test.html`.

Si l'entrée est un chemin absolu, le transformateur ne l'altère pas.

Le tableau suivant décrit les propriétés du transformateur **AbsURL** :

Propriété	Description
base_URL	Définit le chemin ou l'URL de base.
désactivé	Détermine si le script ignore le composant et tous les composants enfants. Utilisez cette propriété pour tester, déboguer et modifier un script. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none">- Sélectionné. Le script ignore le composant.- Effacé. Le script applique le composant. La valeur par défaut est vide.
nom	Libellé descriptif pour le composant. Ce libellé s'affiche dans le fichier journal et la vue Événements . Utilisez la propriété nom pour identifier le composant ayant provoqué l'événement.
facultatif	Détermine si la défaillance d'un composant entraîne l'échec du composant parent. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none">- Sélectionné. La défaillance du composant n'entraîne pas l'échec du composant parent.- Effacé. La défaillance du composant entraîne l'échec du composant parent. Par défaut, la valeur est vide. Pour plus d'informations sur l'échec des composants, consultez la section "Traitement des échecs" à la page 404 .
remarque	Commentaire défini par l'utilisateur, qui décrit l'objectif ou l'action du composant.

AddEmptyTagsTransformer

Le transformateur **AddEmptyTagsTransformer** vérifie si tous les éléments définis dans le schéma existent dans l'entrée XML. Dans le cas contraire, il ajoute des éléments vides au XML. Il s'agit d'un transformateur XML-to-XML.

Le tableau suivant décrit les propriétés du transformateur **AddEmptyTagsTransformer** :

Propriété	Description
désactivé	Détermine si le script ignore le composant et tous les composants enfants. Utilisez cette propriété pour tester, déboguer et modifier un script. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none">- Sélectionné. Le script ignore le composant.- Effacé. Le script applique le composant. La valeur par défaut est vide.
nom	Libellé descriptif pour le composant. Ce libellé s'affiche dans le fichier journal et la vue Événements . Utilisez la propriété nom pour identifier le composant ayant provoqué l'événement.
remarque	Commentaire défini par l'utilisateur, qui décrit l'objectif ou l'action du composant.
root_element	Définit l'élément racine du XML.

AddString

Le transformateur **AddString** ajoute des chaînes avant et après le texte d'entrée.

Le tableau suivant décrit les propriétés du transformateur **AddString** :

Propriété	Description
désactivé	Détermine si le script ignore le composant et tous les composants enfants. Utilisez cette propriété pour tester, déboguer et modifier un script. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none">- Sélectionné. Le script ignore le composant.- Effacé. Le script applique le composant. La valeur par défaut est vide.
nom	Libellé descriptif pour le composant. Ce libellé s'affiche dans le fichier journal et la vue Événements . Utilisez la propriété nom pour identifier le composant ayant provoqué l'événement.
facultatif	Détermine si la défaillance d'un composant entraîne l'échec du composant parent. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none">- Sélectionné. La défaillance du composant n'entraîne pas l'échec du composant parent.- Effacé. La défaillance du composant entraîne l'échec du composant parent. Par défaut, la valeur est vide. Pour plus d'informations sur l'échec des composants, consultez la section "Traitement des échecs" à la page 404 .
pre	Définit la chaîne à ajouter avant le texte.
post	Définit la chaîne à ajouter après le texte.
remarque	Commentaire défini par l'utilisateur, qui décrit l'objectif ou l'action du composant.

Exemple en ligne

Pour obtenir un échantillon en ligne, ouvrez `samples\Projects\Transformers_Example\Transformers_Example.cmw`. La première ancre Contenu de l'analyseur est configurée avec un transformateur **AddString**.

Base64Decode

Le transformateur **Base64Decode** convertit le codage MIME base64 en chaîne binaire.

Le tableau suivant décrit les propriétés du transformateur **Base64Decode** :

Propriété	Description
désactivé	Détermine si le script ignore le composant et tous les composants enfants. Utilisez cette propriété pour tester, déboguer et modifier un script. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none">- Sélectionné. Le script ignore le composant.- Effacé. Le script applique le composant. La valeur par défaut est vide.
nom	Libellé descriptif pour le composant. Ce libellé s'affiche dans le fichier journal et la vue Événements . Utilisez la propriété nom pour identifier le composant ayant provoqué l'événement.
facultatif	Détermine si la défaillance d'un composant entraîne l'échec du composant parent. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none">- Sélectionné. La défaillance du composant n'entraîne pas l'échec du composant parent.- Effacé. La défaillance du composant entraîne l'échec du composant parent. Par défaut, la valeur est vide. Pour plus d'informations sur l'échec des composants, consultez la section "Traitement des échecs" à la page 404 .
remarque	Commentaire défini par l'utilisateur, qui décrit l'objectif ou l'action du composant.
tolerance	Cette propriété contrôle la manière dont le transformateur traite les caractères d'espacement ou les sections non codées en Base64 de son entrée. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none">- ignore_white_spaces. Traite tous les caractères à l'exception des espaces. Valeur par défaut.- ignore_none. Traite tous les caractères.- ignore_non_base64. Traite uniquement les caractères base64.

Base64Encode

Le transformateur **Base64Encode** convertit une chaîne binaire dans l'encodage MIME base64. Ceci est utile pour l'enregistrement de données binaires en XML.

Le tableau suivant décrit les propriétés du transformateur **Base64Encode** :

Propriété	Description
désactivé	Détermine si le script ignore le composant et tous les composants enfants. Utilisez cette propriété pour tester, déboguer et modifier un script. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none">- Sélectionné. Le script ignore le composant.- Effacé. Le script applique le composant. La valeur par défaut est vide.
nom	Libellé descriptif pour le composant. Ce libellé s'affiche dans le fichier journal et la vue Événements . Utilisez la propriété nom pour identifier le composant ayant provoqué l'événement.

Propriété	Description
facultatif	Détermine si la défaillance d'un composant entraîne l'échec du composant parent. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none"> - Sélectionné. La défaillance du composant n'entraîne pas l'échec du composant parent. - Effacé. La défaillance du composant entraîne l'échec du composant parent. Par défaut, la valeur est vide. Pour plus d'informations sur l'échec des composants, consultez la section "Traitement des échecs" à la page 404 .
remarque	Commentaire défini par l'utilisateur, qui décrit l'objectif ou l'action du composant.

BidiConvert

Le transformateur **BidiConvert** inverse les chaînes qui sont écrites dans les langues allant de droite à gauche (RTL), comme l'hébreu ou l'arabe. L'entrée doit être au format RTL. La sortie est LTR.

Le transformateur **BidiConvert** opère sous Windows où le langage par défaut est RTL. Pour un transformateur similaire fonctionnant sur toutes les plates-formes, utilisez **hebrewBidi**. Les deux transformateurs utilisent des algorithmes légèrement différents qui donnent parfois des résultats différents.

Le tableau suivant décrit les propriétés du transformateur **BidiConvert** :

Propriété	Description
désactivé	Détermine si le script ignore le composant et tous les composants enfants. Utilisez cette propriété pour tester, déboguer et modifier un script. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none"> - Sélectionné. Le script ignore le composant. - Effacé. Le script applique le composant. La valeur par défaut est vide.
nom	Libellé descriptif pour le composant. Ce libellé s'affiche dans le fichier journal et la vue Événements . Utilisez la propriété nom pour identifier le composant ayant provoqué l'événement.
remarque	Commentaire défini par l'utilisateur, qui décrit l'objectif ou l'action du composant.

Remarque: Ce composant ne prend pas en charge l'encodage d'entrée UTF-16LE. Solution palliative : Utilisez **hebrewBidi**.

CDATADecode

Le transformateur **CDATADecode** décode une section **CDATA** d'un document XML. Par exemple, il convertit

```
<![CDATA[100 < 200]]>
```

en

```
100 < 200
```

Remarque: Si vous écrivez le résultat en XML, le script le réencode en utilisant le codage XML standard :

```
100 &lt; 200
```

Le tableau suivant décrit les propriétés du transformateur **CDATADecode** :

Propriété	Description
désactivé	Détermine si le script ignore le composant et tous les composants enfants. Utilisez cette propriété pour tester, déboguer et modifier un script. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none">- Sélectionné. Le script ignore le composant.- Effacé. Le script applique le composant. La valeur par défaut est vide.
nom	Libellé descriptif pour le composant. Ce libellé s'affiche dans le fichier journal et la vue Événements . Utilisez la propriété nom pour identifier le composant ayant provoqué l'événement.
facultatif	Détermine si la défaillance d'un composant entraîne l'échec du composant parent. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none">- Sélectionné. La défaillance du composant n'entraîne pas l'échec du composant parent.- Effacé. La défaillance du composant entraîne l'échec du composant parent. Par défaut, la valeur est vide. Pour plus d'informations sur l'échec des composants, consultez la section "Traitement des échecs" à la page 404 .
remarque	Commentaire défini par l'utilisateur, qui décrit l'objectif ou l'action du composant.

CDATAEncode

Le transformateur **CDATAEncode** convertit une chaîne en une section **CDATA** d'un document XML. Par exemple, il convertit

```
100 < 200
```

en

```
<![CDATA[100 < 200]]>
```

Le tableau suivant décrit les propriétés du transformateur **CDATAEncode** :

Propriété	Description
désactivé	Détermine si le script ignore le composant et tous les composants enfants. Utilisez cette propriété pour tester, déboguer et modifier un script. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none">- Sélectionné. Le script ignore le composant.- Effacé. Le script applique le composant. La valeur par défaut est vide.
nom	Libellé descriptif pour le composant. Ce libellé s'affiche dans le fichier journal et la vue Événements . Utilisez la propriété nom pour identifier le composant ayant provoqué l'événement.
facultatif	Détermine si la défaillance d'un composant entraîne l'échec du composant parent. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none">- Sélectionné. La défaillance du composant n'entraîne pas l'échec du composant parent.- Effacé. La défaillance du composant entraîne l'échec du composant parent. Par défaut, la valeur est vide. Pour plus d'informations sur l'échec des composants, consultez la section "Traitement des échecs" à la page 404 .
remarque	Commentaire défini par l'utilisateur, qui décrit l'objectif ou l'action du composant.

ChangeCase

Le transformateur **ChangeCase** change une chaîne de texte tout en majuscules ou tout en minuscules ou seulement la première lettre en majuscule. Ce transformateur fonctionne sur des caractères anglais. Il peut échouer sur des caractères non anglais. Par exemple, il ne convertit pas la minuscule allemande ß en majuscules SS.

Le tableau suivant décrit les propriétés du transformateur **ChangeCase** :

Propriété	Description
case_type	Définit la casse de la sortie. La propriété case_type possède les options suivantes : <ul style="list-style-type: none">- all_caps. La sortie est entièrement en majuscules.- all_lower. La sortie est entièrement en minuscules.- first_cap. La première lettre de la sortie est en majuscule et le reste est en minuscule. La valeur par défaut est all_caps.
désactivé	Détermine si le script ignore le composant et tous les composants enfants. Utilisez cette propriété pour tester, déboguer et modifier un script. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none">- Sélectionné. Le script ignore le composant.- Effacé. Le script applique le composant. La valeur par défaut est vide.
nom	Libellé descriptif pour le composant. Ce libellé s'affiche dans le fichier journal et la vue Événements . Utilisez la propriété nom pour identifier le composant ayant provoqué l'événement.
remarque	Commentaire défini par l'utilisateur, qui décrit l'objectif ou l'action du composant.

Exemple en ligne

Pour obtenir un échantillon en ligne, ouvrez `samples\Projects\Transformers_Example\Transformers_Example.cmw`. La troisième ancre Contenu de l'analyseur est configurée avec un transformateur `ChangeCase`.

CreateGuid

Le transformateur **CreateGuid** génère un identifiant GUID. Le GUID en résultant est unique à chaque fois que ce transformateur s'exécute.

Les GUID peuvent avoir un format non standard sur Linux et les plates-formes UNIX. Pour un transformateur entièrement compatible UNIX, utilisez **CreateUUID**. Pour plus d'informations, consultez la section ["CreateUUID" à la page 267](#)

CreateUUID

Le transformateur **CreateUUID** génère un identifiant UUID compatible avec Windows, Linux et les plates-formes UNIX. Le UUID en résultat est unique à chaque exécution du transformateur.

Le tableau suivant décrit les propriétés du transformateur **CreateUUID** :

Propriété	Description
désactivé	Détermine si le script ignore le composant et tous les composants enfants. Utilisez cette propriété pour tester, déboguer et modifier un script. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none">- Sélectionné. Le script ignore le composant.- Effacé. Le script applique le composant. La valeur par défaut est vide.
nom	Libellé descriptif pour le composant. Ce libellé s'affiche dans le fichier journal et la vue Événements . Utilisez la propriété nom pour identifier le composant ayant provoqué l'événement.
facultatif	Détermine si la défaillance d'un composant entraîne l'échec du composant parent. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none">- Sélectionné. La défaillance du composant n'entraîne pas l'échec du composant parent.- Effacé. La défaillance du composant entraîne l'échec du composant parent. Par défaut, la valeur est vide. Pour plus d'informations sur l'échec des composants, consultez la section "Traitement des échecs" à la page 404 .
remarque	Commentaire défini par l'utilisateur, qui décrit l'objectif ou l'action du composant.

DateFormatICU

Le transformateur **DateFormatICU** convertit une date ou heure dans un format spécifié par l'utilisateur.

Le tableau suivant décrit les propriétés du transformateur **DateFormatICU** :

Propriété	Description
désactivé	Détermine si le script ignore le composant et tous les composants enfants. Utilisez cette propriété pour tester, déboguer et modifier un script. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none">- Sélectionné. Le script ignore le composant.- Effacé. Le script applique le composant. La valeur par défaut est vide.
input_format	Définit le format de la date en entrée, par exemple, d/M/yy. Saisissez le format ou sélectionnez une zone de stockage des données qui contient le format.
nom	Libellé descriptif pour le composant. Ce libellé s'affiche dans le fichier journal et la vue Événements . Utilisez la propriété nom pour identifier le composant ayant provoqué l'événement.
facultatif	Détermine si la défaillance d'un composant entraîne l'échec du composant parent. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none">- Sélectionné. La défaillance du composant n'entraîne pas l'échec du composant parent.- Effacé. La défaillance du composant entraîne l'échec du composant parent. Par défaut, la valeur est vide. Pour plus d'informations sur l'échec des composants, consultez la section "Traitement des échecs" à la page 404 .
output_format	Définit le format de la date en sortie, par exemple, MM/dd/yyyy. Saisissez le format ou sélectionnez une zone de stockage des données qui contient le format.
remarque	Commentaire défini par l'utilisateur, qui décrit l'objectif ou l'action du composant.

Exemple

Supposons que vous configuriez un transformateur `DateFormatICU` avec :

```
input_format = "d/M/yy"
output_format = "MM/dd/yyyy"
```

Si l'entrée est

13/3/05

la sortie est

03/13/2005

Formats pris en charge

Le transformateur **DateFormatICU** utilise les conventions ICU pour représenter le format de date et d'heure. Le tableau suivant liste les symboles utilisables comme modèles de format. Pour plus d'informations, voir :

<http://icu.sourceforge.net/apiref/icu4c/classSimpleDateFormat.html>

Symbole du modèle	Signification	Type	Exemples
G	Indicateur de période	Text	AD
o	Année	Nombre	1996
u	Année prolongée	Nombre	-200, c'est-à-dire 201 BC
M	Mois de l'année	Texte ou nombre	Juillet 07
d	Jour du mois	Nombre	10
h	Heure au format AM/PM (de 1 à 12)	Nombre	12
H	Heure de la journée (de 0 à 23)	Nombre	0
m	Minutes de l'heure	Nombre	30
s	Secondes de la minute	Nombre	55
S	Fraction de seconde	Nombre	978
E	Jour de la semaine	Text	Mardi
e	Jour de la semaine (format local de 1 à 7)	Nombre	2
D	Jour de l'année	Nombre	189
F	Jour de la semaines du mois	Nombre	2 , c'est-à-dire le deuxième mercredi de juillet
w	Semaine de l'année	Nombre	27
W	Semaine du mois	Nombre	2
a	Marqueur AM/PM	Text	PM

Symbole du modèle	Signification	Type	Exemples
k	Heure de la journée (de 1 à 24)	Nombre	24
K	Heure au format AM/PM (de 0 à 11)	Nombre	0
z	Fuseau horaire	Heure	Heure normale du Pacifique
Z	Fuseau horaire (RFC 822)	Nombre	-0800
v	Fuseau horaire (générique)	Text	Heure du Pacifique
g	Date ordinale	Nombre	2451334
A	Millisecondes de la journée	Nombre	69540000
' '	Le texte entre guillemets simples est interprété comme une chaîne littérale	Text	'Today is' jj/MM/aaaa Génère une sortie comme Today is 15/03/2005
' '	Guillemet simple littéral	Text	'o' 'clock' génère la sortie o'clock

Le nombre de symboles du modèle détermine également le format :

- Pour le texte : quatre symboles ou plus correspondent à l'utilisation de la forme complète. Un nombre inférieur à quatre correspond à l'utilisation d'une forme brève ou abrégée, le cas échéant. Par exemple, si `EEEE` produit `Lundi`, `EEE` produit `Lun`.
- Pour les nombres : le nombre de symboles du modèle est le nombre minimum de chiffres. Les nombres les plus courts sont accolés à un zéro. Par exemple, si `m` produit `6`, `mm` produit `06`.
- Pour les années : l'année à deux chiffres est `aa`, et l'année à quatre chiffres est `aaaa`. Par exemple, si `aa` produit `05`, `aaaa` produit `2005`.
- Pour les mois : si `M` produit `1`, `MM` produit `01`, `MMM` produit `Jan` et `MMMM` produit `Janvier`.

Tous les caractères non alphabétiques sont interprétés comme des littéraux, même s'ils ne sont pas entourés de guillemets simples. Par exemple, `jj/MM/aaaaHH:mm` produit `15/03/2005 13:15`.

Dos96HebToAscii

Le transformateur **Dos96HebToAscii** convertit le codage Hebrew 7-bit en page de code Windows-1255.

DynamicTable

Le composant **DynamicTable** définit une zone de stockage des données qui contient une table de recherche. La table est utilisée par le transformateur **LookupTransformer**.

Le tableau suivant décrit les propriétés du composant **DynamicTable** :

Propriété	Description
table	Définit la zone de stockage des données qui contient la table.

EbcdicToAscii

Le transformateur **EbcdicToAscii** convertit des EBCDIC en texte ASCII.

EDIFACTValidation

Le validateur **EDIFACTValidation** teste si une chaîne source est un message EDIFACT valide.

Le tableau suivant décrit les propriétés du validateur **EDIFACTValidation** :

Propriété	Description
désactivé	Détermine si le script ignore le composant et tous les composants enfants. Utilisez cette propriété pour tester, déboguer et modifier un script. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none">- Sélectionné. Le script ignore le composant.- Effacé. Le script applique le composant. La valeur par défaut est vide.
activé	Détermine le paramètre pour param1 .
nom	Libellé descriptif pour le composant. Ce libellé s'affiche dans le fichier journal et la vue Événements . Utilisez la propriété nom pour identifier le composant ayant provoqué l'événement.
facultatif	Détermine si la défaillance d'un composant entraîne l'échec du composant parent. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none">- Sélectionné. La défaillance du composant n'entraîne pas l'échec du composant parent.- Effacé. La défaillance du composant entraîne l'échec du composant parent. Par défaut, la valeur est vide. Pour plus d'informations sur l'échec des composants, consultez la section "Traitement des échecs" à la page 404 .
param1	Détermine si l'entrée est facultative. param1 est nommé is_optional et n'a qu'une seule propriété, activé . activé a les options suivantes : <ul style="list-style-type: none">- Sélectionné. Les données d'entrée sont facultatives.- Effacé. Les données d'entrée sont obligatoires.
param2	Définit un type de données EDI. param2 est nommé input_type et n'a qu'une seule propriété, valeur . valeur est une chaîne entrée en dur ou une zone de stockage des données.
param3	Définit une plage de nombres entiers. param3 est nommé minmax_limits et n'a qu'une seule propriété, valeur . valeur est une chaîne entrée en dur ou une zone de stockage des données qui spécifie deux entiers séparés par un trait d'union.
param4	Définit une liste de valeurs. param4 est nommé énumérations et n'a qu'une seule propriété, valeur . valeur est une chaîne entrée en dur ou une zone de stockage des données qui spécifie une liste de chaînes ou d'entiers séparés par des virgules.

Propriété	Description
remarque	Commentaire défini par l'utilisateur, qui décrit l'objectif ou l'action du composant.
valeur	Définit une valeur pour param1 , param2 ou param3

Remarque: Ce composant est obsolète. L'éditeur IntelliScript l'affiche pour les projets hérités. Ne l'utilisez pas dans de nouveaux scripts. **Solution de contournement :** Utilisez d'autres composants du validateur.

EncodeAsUrl

Le transformateur **EncodeAsUrl** code les espaces et les caractères spéciaux, comme requis dans une URL. Les caractères sont codés sous la forme d'un signe de pourcentage (%) suivi d'un nombre hexadécimal.

Par exemple, le transformateur **EncodeAsUrl** convertit

```
http://www.example.com?name=John Doe
```

en

```
http://www.example.com?name=John%20Doe
```

Remarque: Les caractères parenthèses ne sont pas codés.

Le tableau suivant décrit les propriétés du transformateur **EncodeAsUrl** :

Propriété	Description
désactivé	Détermine si le script ignore le composant et tous les composants enfants. Utilisez cette propriété pour tester, déboguer et modifier un script. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none"> - Sélectionné. Le script ignore le composant. - Effacé. Le script applique le composant. La valeur par défaut est vide.
nom	Libellé descriptif pour le composant. Ce libellé s'affiche dans le fichier journal et la vue Événements . Utilisez la propriété nom pour identifier le composant ayant provoqué l'événement.
remarque	Commentaire défini par l'utilisateur, qui décrit l'objectif ou l'action du composant.

Exemple en ligne

Pour obtenir un échantillon en ligne, ouvrez `samples\Projects\Transformers_Example\Transformers_Example.cmw`. La quatrième ancre Contenu de l'analyseur est configurée avec un transformateur `EncodeAsUrl`.

Encodeur

Le transformateur **Encodeur** convertit le texte d'une page de code vers une autre.

Le tableau suivant décrit les propriétés du transformateur **Encodeur** :

Propriété	Description
add_prefix	Ajoute un indicateur d'ordre des octets (BOM) quand le codage de la sortie est UTF-16LE ou UTF-16.
désactivé	Détermine si le script ignore le composant et tous les composants enfants. Utilisez cette propriété pour tester, déboguer et modifier un script. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none"> - Sélectionné. Le script ignore le composant. - Effacé. Le script applique le composant. La valeur par défaut est vide.
input_code_page	Définit la page de code du texte d'entrée.
nom	Libellé descriptif pour le composant. Ce libellé s'affiche dans le fichier journal et la vue Événements . Utilisez la propriété nom pour identifier le composant ayant provoqué l'événement.
facultatif	Détermine si la défaillance d'un composant entraîne l'échec du composant parent. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none"> - Sélectionné. La défaillance du composant n'entraîne pas l'échec du composant parent. - Effacé. La défaillance du composant entraîne l'échec du composant parent. Par défaut, la valeur est vide. Pour plus d'informations sur l'échec des composants, consultez la section "Traitement des échecs" à la page 404 .
output_code_page	Définit la page de code du texte de sortie.
remarque	Commentaire défini par l'utilisateur, qui décrit l'objectif ou l'action du composant.

FormatNumber

Le transformateur **FormatNumber** formate un nombre en ajoutant un signe, un point décimal, des zéros au début ou à la fin et une unité.

Le tableau suivant décrit les propriétés du transformateur **FormatNumber** :

Propriété	Description
désactivé	Détermine si le script ignore le composant et tous les composants enfants. Utilisez cette propriété pour tester, déboguer et modifier un script. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none"> - Sélectionné. Le script ignore le composant. - Effacé. Le script applique le composant. La valeur par défaut est vide.
insert_decimal_point	Définit le symbole de point décimal. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none"> - virgule. Le séparateur de décimale est une virgule. - aucun. La sortie n'a pas de décimale. - point. Le séparateur de décimale est un point. La valeur par défaut est Aucun.
nom	Libellé descriptif pour le composant. Ce libellé s'affiche dans le fichier journal et la vue Événements . Utilisez la propriété nom pour identifier le composant ayant provoqué l'événement.

Propriété	Description
nombre_de_décimales	Rajoute des zéros en fin de partie décimale pour l'amener à la taille indiquée. La valeur par défaut est 0.
facultatif	Détermine si la défaillance d'un composant entraîne l'échec du composant parent. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none"> - Sélectionné. La défaillance du composant n'entraîne pas l'échec du composant parent. - Effacé. La défaillance du composant entraîne l'échec du composant parent. Par défaut, la valeur est vide. Pour plus d'informations sur l'échec des composants, consultez la section "Traitement des échecs" à la page 404 .
remarque	Commentaire défini par l'utilisateur, qui décrit l'objectif ou l'action du composant.
signe	Détermine le signe du nombre en sortie. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none"> - un_signed. Supprime un signe, si présent. - leading_sign. Un plus ou un moins est ajouté devant le nombre de sortie. - trailing_sign. Un plus ou un moins est ajouté après le nombre de sortie. - signe négatif uniquement. Un signe moins est ajouté au nombre si le nombre est négatif. - comme dans la source. Ne modifie pas le signe d'entrée. La valeur par défaut est un_signed.
size_of_integer_part	Rajoute des zéros en tête de la partie entière pour l'amener à la taille indiquée. La valeur par défaut est 0.
unit_type	Définit l'unité de mesure après le nombre. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none"> - cm - pouce - mètre - mm - indéfini. Aucune unité n'est ajoutée. La valeur par défaut est Indéfini.

FromFloat

Le transformateur **FromFloat** convertit un nombre à virgule flottante de binaire à une représentation de chaîne ASCII. La conversion est effectuée dans le codage d'entrée dans l'ordre d'octet de l'entrée.

Le tableau suivant décrit les propriétés du transformateur **FromFloat** :

Propriété	Description
désactivé	Détermine si le script ignore le composant et tous les composants enfants. Utilisez cette propriété pour tester, déboguer et modifier un script. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none"> - Sélectionné. Le script ignore le composant. - Effacé. Le script applique le composant. La valeur par défaut est vide.
nom	Libellé descriptif pour le composant. Ce libellé s'affiche dans le fichier journal et la vue Événements . Utilisez la propriété nom pour identifier le composant ayant provoqué l'événement.
facultatif	Détermine si la défaillance d'un composant entraîne l'échec du composant parent. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none"> - Sélectionné. La défaillance du composant n'entraîne pas l'échec du composant parent. - Effacé. La défaillance du composant entraîne l'échec du composant parent. Par défaut, la valeur est vide. Pour plus d'informations sur l'échec des composants, consultez la section "Traitement des échecs" à la page 404 .

Propriété	Description
remarque	Commentaire défini par l'utilisateur, qui décrit l'objectif ou l'action du composant.
taille	Détermine la taille du nombre en entrée. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none"> - single_precision_32_bit - double_precision_64_bit La valeur par défaut est single_precision_32_bit.

Remarque: Ce composant ne prend pas en charge l'encodage d'entrée UTF-16LE.

FromInteger

Le transformateur **FromInteger** convertit un nombre entier de binaire à une représentation de chaîne ASCII, en décimal, octal ou hexadécimal. La conversion est effectuée dans le codage d'entrée dans l'ordre d'octet de l'entrée.

Le tableau suivant décrit les propriétés du transformateur **FromInteger** :

Propriété	Description
désactivé	Détermine si le script ignore le composant et tous les composants enfants. Utilisez cette propriété pour tester, déboguer et modifier un script. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none"> - Sélectionné. Le script ignore le composant. - Effacé. Le script applique le composant. La valeur par défaut est vide.
nom	Libellé descriptif pour le composant. Ce libellé s'affiche dans le fichier journal et la vue Événements . Utilisez la propriété nom pour identifier le composant ayant provoqué l'événement.
facultatif	Détermine si la défaillance d'un composant entraîne l'échec du composant parent. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none"> - Sélectionné. La défaillance du composant n'entraîne pas l'échec du composant parent. - Effacé. La défaillance du composant entraîne l'échec du composant parent. Par défaut, la valeur est vide. Pour plus d'informations sur l'échec des composants, consultez la section "Traitement des échecs" à la page 404 .
remarque	Commentaire défini par l'utilisateur, qui décrit l'objectif ou l'action du composant.
signé	Détermine si le nombre en sortie a un signe. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none"> - Sélectionné. Le nombre en sortie a un signe. - Effacé. Le nombre en sortie n'a pas de signe. La valeur par défaut est Effacé.
taille	Définit la taille en octets de l'entrée binaire. Les valeurs prises en charge sont comprises entre 1 et 8. La valeur par défaut est 1.
to_base	Définit la base de la sortie. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none"> - décimal. Base 10. - hexadécimal. Base 16 en utilisant les lettres majuscules A-F - hexadécimal minuscule. Base 16 en utilisant les lettres minuscules a-f - octal. Base 8. La valeur par défaut est Décimal.

Remarque: Ce composant ne prend pas en charge l'encodage d'entrée UTF-16LE.

FromPackDecimal

Le transformateur **FromPackDecimal** convertit un nombre depuis un décimal condensé vers une représentation de chaîne ASCII. La conversion est effectuée dans le codage d'entrée dans l'ordre d'octet de l'entrée.

Le tableau suivant décrit les propriétés du transformateur **FromPackDecimal** :

Propriété	Description
désactivé	Détermine si le script ignore le composant et tous les composants enfants. Utilisez cette propriété pour tester, déboguer et modifier un script. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none">- Sélectionné. Le script ignore le composant.- Effacé. Le script applique le composant. La valeur par défaut est vide.
nom	Libellé descriptif pour le composant. Ce libellé s'affiche dans le fichier journal et la vue Événements . Utilisez la propriété nom pour identifier le composant ayant provoqué l'événement.
facultatif	Détermine si la défaillance d'un composant entraîne l'échec du composant parent. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none">- Sélectionné. La défaillance du composant n'entraîne pas l'échec du composant parent.- Effacé. La défaillance du composant entraîne l'échec du composant parent. Par défaut, la valeur est vide. Pour plus d'informations sur l'échec des composants, consultez la section "Traitement des échecs" à la page 404 .
remarque	Commentaire défini par l'utilisateur, qui décrit l'objectif ou l'action du composant.

Remarque: Ce composant ne prend pas en charge l'encodage d'entrée UTF-16LE.

FromSignedDecimal

Le transformateur **FromSignedDecimal** convertit un nombre depuis un décimal signé vers une représentation de chaîne ASCII. La conversion est effectuée dans le codage d'entrée dans l'ordre d'octet de l'entrée.

Le tableau suivant décrit les propriétés du transformateur **FromSignedDecimal** :

Propriété	Description
désactivé	Détermine si le script ignore le composant et tous les composants enfants. Utilisez cette propriété pour tester, déboguer et modifier un script. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none">- Sélectionné. Le script ignore le composant.- Effacé. Le script applique le composant. La valeur par défaut est vide.
insert_sign_symbol	Définit le signe du nombre. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none">- après. Un signe plus ou moins est ajouté après le nombre de sortie.- avant. Un signe plus ou moins est ajouté devant le nombre de sortie.- non. La sortie est non signée. La valeur par défaut est Non.
nom	Libellé descriptif pour le composant. Ce libellé s'affiche dans le fichier journal et la vue Événements . Utilisez la propriété nom pour identifier le composant ayant provoqué l'événement.

Propriété	Description
facultatif	Détermine si la défaillance d'un composant entraîne l'échec du composant parent. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none"> - Sélectionné. La défaillance du composant n'entraîne pas l'échec du composant parent. - Effacé. La défaillance du composant entraîne l'échec du composant parent. Par défaut, la valeur est vide. Pour plus d'informations sur l'échec des composants, consultez la section "Traitement des échecs" à la page 404 .
remarque	Commentaire défini par l'utilisateur, qui décrit l'objectif ou l'action du composant.

Remarque: Ce composant ne prend pas en charge l'encodage d'entrée UTF-16LE.

hebrewBidi

Le transformateur **hebrewBidi** inverse une chaîne qui est écrite dans une langue écrite de droite à gauche (RTL), par exemple l'hébreu et l'arabe.

L'entrée doit être au format RTL. La sortie est LTR.

HebrewDosToWindows

Le transformateur **HebrewDosToWindows** convertit des documents en hébreu depuis la page de code Hébreu MS-DOS vers la page de code Hébreu de Windows.

HebrewEBCDICOldCodeToWindows

Le transformateur **HebrewEBCDICOldCodeToWindows** convertit un texte hébreu depuis EBCDIC vers la page de code Windows-1255.

hebUniToAscii

Le transformateur **hebUniToAscii** convertit un texte hébreu depuis l'Unicode UTF-16 en page de code Windows-1255.

hebUtf8ToAscii

Le transformateur **hebUtf8ToAscii** convertit du texte en hébreu depuis Unicode UTF-16LE en page de code Windows-1255.

HtmlEntitiesToASCII

Le transformateur **HtmlEntitiesToASCII** convertit les entités HTML en texte brut. Par exemple, il convertit `©` ou `©` en symbole de copyright (©).

Le tableau suivant décrit les propriétés du transformateur **HtmlEntitiesToASCII** :

Propriété	Description
Désactivé	Détermine si le script ignore le composant et tous les composants enfants. Utilisez cette propriété pour tester, déboguer et modifier un script. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none">- Sélectionné. Le script ignore le composant.- Effacé. Le script applique le composant. La valeur par défaut est vide.
nom	Libellé descriptif pour le composant. Ce libellé s'affiche dans le fichier journal et la vue Événements . Utilisez la propriété nom pour identifier le composant ayant provoqué l'événement.
remarque	Commentaire défini par l'utilisateur, qui décrit l'objectif ou l'action du composant.

Entités prises en charge

Le transformateur prend en charge les entités ISO 8859-1 (Latin-1) qui sont définies dans la référence HTML 4.0, <http://www.w3.org/TR/1998/REC-html40-19980424/sgml/entities.html>. Les entités prises en charge comprennent :

- & amp; amp;, & amp; lt;, & amp; g; et & amp; quot; (& < > ", respectivement)
- Les codes en caractères numériques de & # 0; à & # 255;
- Les entités des caractères Latin-1 : & amp; nbsp; = espace insécable, & amp; copy; = copyright, etc.

Le transformateur ne prend pas en charge les caractères étendus, c'est-à-dire les codes d'une longueur supérieure à 255 caractères ou en caractères autres que Latin-1.

Encodage de sortie pour les caractères ASCII majuscules

Si la sortie du transformateur contient des caractères ASCII majuscules, sélectionnez un encodage de sortie qui prend en charge ces caractères, comme Windows-1252 ou UTF-16LE.

Remarque: Incluez un attribut d'encodage dans l'instruction de traitement XML. Sinon, l'outil Developer peut ne pas être capable d'afficher les caractères.

HtmlProcessor

Le transformateur **HtmlProcessor** normalise les espaces conformément aux conventions HTML. Il convertit toute séquence de tabulations, de retours à la ligne et d'espaces en un caractère espace unique. Ce transformateur opère sur du texte HTML et tout autre type de texte. Vous pouvez également l'utiliser en tant que préprocesseur de format. Pour plus d'informations, consultez la section "[Documentation de référence du composant préprocesseur de format](#)" à la page 190.

InjectFP

Le transformateur **InjectFP** insère un point décimal à l'emplacement spécifié dans un nombre. Par exemple, le transformateur peut convertir 12345 en 123.45.

Le tableau suivant décrit les propriétés du transformateur **InjectFP** :

Propriété	Description
digits_after_decimal_point	Détermine le nombre de chiffres après le point décimal.
désactivé	Détermine si le script ignore le composant et tous les composants enfants. Utilisez cette propriété pour tester, déboguer et modifier un script. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none">- Sélectionné. Le script ignore le composant.- Effacé. Le script applique le composant. La valeur par défaut est vide.
nom	Libellé descriptif pour le composant. Ce libellé s'affiche dans le fichier journal et la vue Événements . Utilisez la propriété nom pour identifier le composant ayant provoqué l'événement.
facultatif	Détermine si la défaillance d'un composant entraîne l'échec du composant parent. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none">- Sélectionné. La défaillance du composant n'entraîne pas l'échec du composant parent.- Effacé. La défaillance du composant entraîne l'échec du composant parent. Par défaut, la valeur est vide. Pour plus d'informations sur l'échec des composants, consultez la section "Traitement des échecs" à la page 404 .
remarque	Commentaire défini par l'utilisateur, qui décrit l'objectif ou l'action du composant.

InjectString

Le transformateur **InjectString** insère une chaîne dans un texte.

Le tableau suivant décrit les propriétés du transformateur **InjectString** :

Propriété	Description
désactivé	Détermine si le script ignore le composant et tous les composants enfants. Utilisez cette propriété pour tester, déboguer et modifier un script. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none">- Sélectionné. Le script ignore le composant.- Effacé. Le script applique le composant. La valeur par défaut est vide.
injection_place	Définit le nombre de caractères depuis le début du texte jusqu'à l'endroit où la chaîne est insérée.
nom	Libellé descriptif pour le composant. Ce libellé s'affiche dans le fichier journal et la vue Événements . Utilisez la propriété nom pour identifier le composant ayant provoqué l'événement.
remarque	Commentaire défini par l'utilisateur, qui décrit l'objectif ou l'action du composant.
string_to_inject	Définit la chaîne à insérer.

InlineTable

Le composant **InlineTable** définit une table de recherche dans le script. La table est utilisée par le transformateur **LookupTransformer**.

Le tableau suivant décrit les propriétés du composant **InlineTable** :

Propriété	Description
Entrée	Définit une paire clé et valeur .
clé	Définit une chaîne d'entrée unique.
match_case	Détermine si la chaîne clé est sensible à la casse. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none">- Sélectionné. clé est sensible à la casse.- Effacé. clé n'est pas sensible à la casse. La valeur par défaut est Effacé.
table	Définit une liste de composants Entry .
valeur	Définit une chaîne de sortie.

JavaTransformer

Le transformateur **JavaTransformer** exécute un transformateur personnalisé qui est implémenté dans Java.

Le tableau suivant décrit les propriétés du transformateur **JavaTransformer** :

Propriété	Description
désactivé	Détermine si le script ignore le composant et tous les composants enfants. Utilisez cette propriété pour tester, déboguer et modifier un script. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none">- Sélectionné. Le script ignore le composant.- Effacé. Le script applique le composant. La valeur par défaut est vide.
java_class	Définit le chemin de la classe Java.
méthode	Définit la méthode à exécuter.
nom	Libellé descriptif pour le composant. Ce libellé s'affiche dans le fichier journal et la vue Événements . Utilisez la propriété nom pour identifier le composant ayant provoqué l'événement.
facultatif	Détermine si la défaillance d'un composant entraîne l'échec du composant parent. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none">- Sélectionné. La défaillance du composant n'entraîne pas l'échec du composant parent.- Effacé. La défaillance du composant entraîne l'échec du composant parent. Par défaut, la valeur est vide. Pour plus d'informations sur l'échec des composants, consultez la section "Traitement des échecs" à la page 404 .
remarque	Commentaire défini par l'utilisateur, qui décrit l'objectif ou l'action du composant.

Remarque: Ce composant est obsolète. L'éditeur IntelliScript l'affiche pour les scripts hérités. Ne l'utilisez pas dans de nouveaux scripts. Créez, à la place, un transformateur Java personnalisé. Pour plus d'informations, consultez la section ["Développement d'un composant personnalisé" à la page 437](#).

LookupTransformer

Le transformateur **LookupTransformer** recherche une valeur dans un tableau.

Le tableau suivant décrit les propriétés du transformateur **LookupTransformer** :

Propriété	Description
désactivé	Détermine si le script ignore le composant et tous les composants enfants. Utilisez cette propriété pour tester, déboguer et modifier un script. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none"> - Sélectionné. Le script ignore le composant. - Effacé. Le script applique le composant. La valeur par défaut est vide.
look_at	Définit le type de table de recherche utilisée par le transformateur. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none"> - DynamicTable. La propriété tableau de la propriété look_at définit une zone de stockage des données qui contient le tableau. Pour plus d'informations, consultez la section "DynamicTable" à la page 271. - InlineTable. La propriété tableau de la propriété look_at définit une liste de composants Entrée, chacun d'eux contenant une clé et une valeur. Pour plus d'informations, consultez la section "InlineTable" à la page 279. - XMLLookupTable. La propriété xml_file_name de la propriété look_at définit le chemin et le nom d'un fichier XML qui définit la table. Pour plus d'informations, consultez la section "XMLLookupTable" à la page 295. - [TableName]. Une table DynamicTable, InlineTable ou XMLLookupTable définie au niveau global du script. Par défaut, la valeur est vide.
nom	Libellé descriptif pour le composant. Ce libellé s'affiche dans le fichier journal et la vue Événements . Utilisez la propriété nom pour identifier le composant ayant provoqué l'événement.
facultatif	Détermine si la défaillance d'un composant entraîne l'échec du composant parent. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none"> - Sélectionné. La défaillance du composant n'entraîne pas l'échec du composant parent. - Effacé. La défaillance du composant entraîne l'échec du composant parent. Par défaut, la valeur est vide. Pour plus d'informations sur l'échec des composants, consultez la section "Traitement des échecs" à la page 404 .
remarque	Commentaire défini par l'utilisateur, qui décrit l'objectif ou l'action du composant.

Si vous utilisez la même table de recherche de manière répétée, pensez à définir une table **InlineTable** ou **XMLLookupTable** au niveau global du script. Vous pouvez alors référencer le tableau par son nom dans la propriété **look_at**.

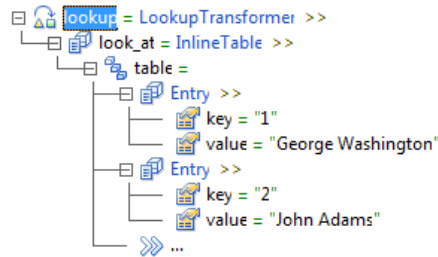
Par exemple, vous pouvez configurer un **LookupTransformer** pour rechercher des valeurs dans la table suivante :

Clé	Valeur
1	George Washington
2	John Adams
3	Thomas Jefferson
4	James Madison

Si l'entrée du transformateur est 3, la sortie est `Thomas Jefferson`.

Définition d'une table intégrée

Pour définir une table intégrée, configurez les paires clé-valeur dans le script, comme dans l'exemple suivant :



Stockage d'une table de recherche dans un fichier XML

Préparez un fichier XML conforme au schéma `lookuptabledefinition.xsd`. Vous pouvez trouver le schéma dans le sous-répertoire `\doc` du répertoire d'installation. Voici un exemple de document XML :

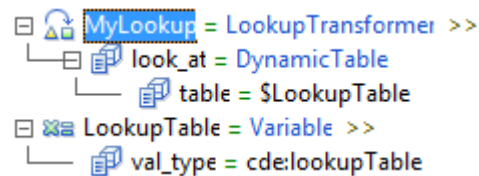
```
<?xml version="1.0" encoding="windows-1252" ?>
<lt:LookupTable xmlns:lt="http://www.itemfield.com/Engine/V4/lookupTable"
matchCase="false">
  <lt:Entry key="1" value="George Washington" />
  <lt:Entry key="2" value="John Adams" />
</lt:LookupTable>
```

Création d'une table de recherche XML de façon dynamique

Une transformation peut créer une table de recherche XML lors de l'exécution. Par exemple, la transformation peut exécuter un analyseur secondaire qui génère la structure XML.

La transformation doit stocker la chaîne XML dans une zone de stockage des données à occurrences multiples de type `cde:lookupTable`. Stockez chaque paire clé-valeur dans une occurrence de la zone de stockage des données. Par exemple, vous pouvez configurer un **RepeatingGroup** contenant une action **WriteValue**. Chaque itération du **RepeatingGroup** crée une occurrence de la zone de stockage des données et écrit une paire clé-valeur dans l'occurrence.

Configurez ensuite un **LookupTransformer** avec l'option **DynamicTable** et spécifiez la zone de stockage des données.



NormalizeClosingTags

Pour une entrée XML, le transformateur **NormalizeClosingTags** supprime les balises de fermeture shorthand des éléments vides. Il change `<balise/>` en `<balise></balise>`.

Le transformateur ne corrige pas les XML incorrects. Il convertit les XML bien structurés d'un style de balises de fermeture à un autre.

Le tableau suivant décrit les propriétés du transformateur **NormalizeClosingTags** :

Propriété	Description
désactivé	Détermine si le script ignore le composant et tous les composants enfants. Utilisez cette propriété pour tester, déboguer et modifier un script. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none">- Sélectionné. Le script ignore le composant.- Effacé. Le script applique le composant. La valeur par défaut est vide.
nom	Libellé descriptif pour le composant. Ce libellé s'affiche dans le fichier journal et la vue Événements . Utilisez la propriété nom pour identifier le composant ayant provoqué l'événement.
remarque	Commentaire défini par l'utilisateur, qui décrit l'objectif ou l'action du composant.

RegularExpression

Le transformateur **RegularExpression** effectue une recherche selon un modèle sur le texte d'entrée. Il remplace les instances du modèle avec une chaîne indiquée.

Le tableau suivant décrit les propriétés du transformateur **RegularExpression** :

Propriété	Description
désactivé	Détermine si le script ignore le composant et tous les composants enfants. Utilisez cette propriété pour tester, déboguer et modifier un script. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none">- Sélectionné. Le script ignore le composant.- Effacé. Le script applique le composant. La valeur par défaut est vide.
exp	Définit une expression régulière pour le critère de recherche.
nom	Libellé descriptif pour le composant. Ce libellé s'affiche dans le fichier journal et la vue Événements . Utilisez la propriété nom pour identifier le composant ayant provoqué l'événement.
facultatif	Détermine si la défaillance d'un composant entraîne l'échec du composant parent. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none">- Sélectionné. La défaillance du composant n'entraîne pas l'échec du composant parent.- Effacé. La défaillance du composant entraîne l'échec du composant parent. Par défaut, la valeur est vide. Pour plus d'informations sur l'échec des composants, consultez la section "Traitement des échecs" à la page 404 .
remarque	Commentaire défini par l'utilisateur, qui décrit l'objectif ou l'action du composant.
remplacement	Définit le texte de remplacement.

Par exemple, supposons qu'un **Contenu** ancre extrait le texte suivant :

```
transformer
```

Vous configurez l'ancre avec un transformateur **RegularExpression** qui recherche le modèle `t.+s`. Le modèle signifie la lettre `t`, suivie d'un ou plusieurs caractères, suivis par la lettre `s`. Vous configurez le transformateur pour remplacer le modèle avec le caractère `x`.

Le modèle correspond à la sous-chaîne `trans` de l'entrée. Le transformateur remplace la sous-chaîne et produit :

`Xformer`

Syntaxe d'expression régulière

Une expression régulière définit un modèle de recherche conformément à une syntaxe standard.

La transformation Processeur de données utilise l'implémentation d'expressions régulières `Regex++`, © 1998-2003 par Dr. John Maddock, Version 1.33, 18 avril 2000.

Remarque: `Regex++` ne prend pas en charge les paramètres régionaux.

Le tableau suivant répertorie plusieurs caractères spéciaux que vous pouvez utiliser dans les expressions régulières :

Caractère	Signification	Exemple
*	Correspond à zéro ou plusieurs instances du caractère précédent.	<code>ab*c</code> correspond à <code>ac</code> , <code>abc</code> ou <code>abbbc</code> .
?	Correspond à zéro ou une instance du caractère précédent.	<code>ab?c</code> correspond à <code>ac</code> ou <code>abc</code> .
+	Correspond à une ou plusieurs instances du caractère précédent.	<code>a+</code> correspond à <code>a</code> ou <code>aaaa</code> .
{ }	Correspond au nombre spécifié d'instances du caractère précédent.	<code>ab{2}c</code> correspond à <code>abbc</code> .
[]	Correspond à n'importe quel caractère du jeu de caractères spécifié.	<code>a[bst]c</code> correspond à <code>abc</code> , <code>asc</code> ou <code>atc</code> .
-	Définit une plage de caractères à l'intérieur des crochets.	<code>[A-Za-z]</code> correspond à n'importe quel caractère de l'alphabet anglais. <code>[A-Za-zü]</code> correspond à n'importe quel caractère de l'alphabet anglais ou au caractère allemand <code>ü</code> .
.	Correspond à n'importe quel caractère unique.	<code>a.c</code> correspond à <code>abc</code> , <code>a c</code> ou <code>a1c</code> .
^	Correspond au début du texte d'entrée.	<code>^P.</code> correspond à <code>Pe</code> , mais pas à <code>Pi</code> dans "Peter Piper."
\$	Correspond à la fin du texte d'entrée.	<code>r.\$</code> correspond à <code>rs</code> dans "Peter Piper's peppers."
	Correspond à l'une ou l'autre des deux expressions.	<code>abc ded</code> correspond à <code>abc</code> ou <code>def</code> .
()	Regroupement	<code>A(abc def)</code> correspon à <code>Aabc</code> ou <code>Adef</code> .
\	Ne prend pas en compte un des autres caractères spéciaux, en le considérant comme un caractère littéral.	<code>\.</code> correspond à un point littéral, au lieu de n'importe quel autre caractère.

Conservation de portions du texte d'origine

Dans la propriété `exp`, vous pouvez inclure des portions de l'expression régulière entre parenthèses. Dans la propriété `replacement`, vous pouvez utiliser :

- `$0` pour identifier la totalité du texte correspondant à l'expression régulière.
- `$1` pour identifier la sous-chaîne qui correspond à la première portion entre parenthèses de l'expression régulière.
- `$2`, `$3`, etc. pour identifier les sous-chaînes qui correspondent à la seconde, troisième, etc. portion entre parenthèses.

Supposez par exemple, que vous paramétrez :

```
exp = abc([0-9]+)(def)
replacement = $1
```

Cela remplace `abc5624def` par `5624`.

Ou bien supposez que vous paramétrez :

```
exp = abc([0-9]+)(def)
replacement = $2ZYX$1
```

Cela remplace `abc5624def` par `defZYX5624`.

RemoveMarginSpace

Le transformateur **RemoveMarginSpace** supprime les caractères d'espacement au début et à la fin du texte.

Le tableau suivant décrit les propriétés du transformateur **RemoveMarginSpace** :

Propriété	Description
désactivé	Détermine si le script ignore le composant et tous les composants enfants. Utilisez cette propriété pour tester, déboguer et modifier un script. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none">- Sélectionné. Le script ignore le composant.- Effacé. Le script applique le composant. La valeur par défaut est vide.
nom	Libellé descriptif pour le composant. Ce libellé s'affiche dans le fichier journal et la vue Événements . Utilisez la propriété nom pour identifier le composant ayant provoqué l'événement.
remarque	Commentaire défini par l'utilisateur, qui décrit l'objectif ou l'action du composant.

RemoveRtfFormatting

Le transformateur **RemoveRtfFormatting** supprime les instructions de formatage RTF depuis le texte.

Le tableau suivant décrit les propriétés du transformateur **RemoveRtfFormatting** :

Propriété	Description
désactivé	Détermine si le script ignore le composant et tous les composants enfants. Utilisez cette propriété pour tester, déboguer et modifier un script. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none">- Sélectionné. Le script ignore le composant.- Effacé. Le script applique le composant. La valeur par défaut est vide.
nom	Libellé descriptif pour le composant. Ce libellé s'affiche dans le fichier journal et la vue Événements . Utilisez la propriété nom pour identifier le composant ayant provoqué l'événement.
remarque	Commentaire défini par l'utilisateur, qui décrit l'objectif ou l'action du composant.

RemoveTags

Le transformateur **RemoveTags** supprime les balises HTML du texte d'entrée. Il remplace les balises dans des emplacements internes dans le texte par une chaîne séparatrice, comme le caractère d'espace. Il n'insère pas la chaîne séparatrice au début ou à la fin du texte. Plusieurs balises contiguës sont transformées en un seul séparateur.

Le tableau suivant décrit les propriétés du transformateur **RemoveTags** :

Propriété	Description
désactivé	Détermine si le script ignore le composant et tous les composants enfants. Utilisez cette propriété pour tester, déboguer et modifier un script. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none">- Sélectionné. Le script ignore le composant.- Effacé. Le script applique le composant. La valeur par défaut est vide.
nom	Libellé descriptif pour le composant. Ce libellé s'affiche dans le fichier journal et la vue Événements . Utilisez la propriété nom pour identifier le composant ayant provoqué l'événement.
facultatif	Détermine si la défaillance d'un composant entraîne l'échec du composant parent. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none">- Sélectionné. La défaillance du composant n'entraîne pas l'échec du composant parent.- Effacé. La défaillance du composant entraîne l'échec du composant parent. Par défaut, la valeur est vide. Pour plus d'informations sur l'échec des composants, consultez la section "Traitement des échecs" à la page 404 .
remarque	Commentaire défini par l'utilisateur, qui décrit l'objectif ou l'action du composant.
replace_with	Définit la chaîne séparatrice. La valeur par défaut est " " (espace).

Remplacer

Le transformateur **Replace** cherche et remplace des chaînes du texte d'entrée. Laisser la propriété **replace_with** vide supprime le texte trouvé.

Le tableau suivant décrit les propriétés du transformateur **Replace** :

Propriété	Description
désactivé	Détermine si le script ignore le composant et tous les composants enfants. Utilisez cette propriété pour tester, déboguer et modifier un script. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none">- Sélectionné. Le script ignore le composant.- Effacé. Le script applique le composant. La valeur par défaut est vide.
find_what	Définit le texte à trouver. La valeur est l'un des composants de recherche suivants : <ul style="list-style-type: none">- NewlineSearch. Recherche un caractère de retour à la ligne.- PatternSearch. Recherche un texte correspondant à une expression régulière.- SegmentSearch. Recherche un segment d'un marqueur d'ouverture spécifié à un marqueur de fermeture.- TextSearch. Recherche une chaîne spécifiée. La valeur par défaut est TextSearch. Pour plus d'informations, consultez le "Référence du composant de recherche" à la page 246 .
nom	Libellé descriptif pour le composant. Ce libellé s'affiche dans le fichier journal et la vue Événements . Utilisez la propriété nom pour identifier le composant ayant provoqué l'événement.
occurrence	Spécifie les occurrences à remplacer : tout, premier ou dernier.
facultatif	Détermine si la défaillance d'un composant entraîne l'échec du composant parent. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none">- Sélectionné. La défaillance du composant n'entraîne pas l'échec du composant parent.- Effacé. La défaillance du composant entraîne l'échec du composant parent. Par défaut, la valeur est vide. Pour plus d'informations sur l'échec des composants, consultez la section "Traitement des échecs" à la page 404 .
remarque	Commentaire défini par l'utilisateur, qui décrit l'objectif ou l'action du composant.
replace_with	Définit la chaîne de remplacement.

Exemple en ligne

Pour un échantillon en ligne, ouvrez `samples\Projects\Transformers_Example\Transformers_Example.cmw`. La deuxième et la cinquième ancre Contenu de l'analyseur sont configurées avec des transformateurs Replace.

Redimensionner

Le transformateur **Resize** fait rentrer le texte d'entrée dans la taille spécifiée. Il rajoute des espaces ou tronque le texte comme requis.

Le tableau suivant décrit les propriétés du transformateur **Resize** :

Propriété	Description
aligner	Définit l'alignement du texte à l'intérieur de la chaîne redimensionnée. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none">- gauche. Le remplissage ou la coupe se fait sur la droite.- droite. Le remplissage ou la coupe se fait sur la gauche.
allow_smaller_size	Si cette propriété est sélectionnée, le transformateur Redimensionnement adapte le texte d'entrée à la taille spécifiée (par remplissage si la chaîne est inférieure à la taille définie dans le paramètre size). Si elle n'est pas sélectionnée et que la chaîne d'entrée est inférieure à la taille spécifiée, le transformateur échoue.
name	Libellé descriptif pour le composant. Ce libellé s'affiche dans le fichier journal et la vue Événements . Utilisez la propriété nom pour identifier le composant ayant provoqué l'événement.
padding_character	Définit le caractère de remplissage. Saisissez le caractère ou sélectionnez une zone de stockage des données contenant un caractère.
remark	Commentaire défini par l'utilisateur, qui décrit l'objectif ou l'action du composant.
taille	Définit la taille du texte de sortie. Saisissez un nombre entier ou sélectionnez une zone de stockage des données contenant un nombre entier.

ReverseTransformer

Le transformateur **ReverseTransformer** inverse une chaîne. Par exemple, il transforme 1234 en 4321.

Le tableau suivant décrit les propriétés du transformateur **ReverseTransformer** :

Propriété	Description
désactivé	Détermine si le script ignore le composant et tous les composants enfants. Utilisez cette propriété pour tester, déboguer et modifier un script. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none">- Sélectionné. Le script ignore le composant.- Effacé. Le script applique le composant. La valeur par défaut est vide.
nom	Libellé descriptif pour le composant. Ce libellé s'affiche dans le fichier journal et la vue Événements . Utilisez la propriété nom pour identifier le composant ayant provoqué l'événement.
facultatif	Détermine si la défaillance d'un composant entraîne l'échec du composant parent. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none">- Sélectionné. La défaillance du composant n'entraîne pas l'échec du composant parent.- Effacé. La défaillance du composant entraîne l'échec du composant parent. Par défaut, la valeur est vide. Pour plus d'informations sur l'échec des composants, consultez la section "Traitement des échecs" à la page 404 .
remarque	Commentaire défini par l'utilisateur, qui décrit l'objectif ou l'action du composant.

RtfProcessor

Le transformateur **RtfProcessor** normalise du code RTF. Il est également disponible en tant que préprocesseur de format. Pour plus d'informations, voir ["Documentation de référence du composant préprocesseur de format" à la page 190](#)

RtfToASCII

Le transformateur **RtfToASCII** convertit l'entrée RTF en texte brut. Il supprime les mots de contrôle RTF du texte.

Le tableau suivant décrit les propriétés du transformateur **RtfToASCII** :

Propriété	Description
désactivé	Détermine si le script ignore le composant et tous les composants enfants. Utilisez cette propriété pour tester, déboguer et modifier un script. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none">- Sélectionné. Le script ignore le composant.- Effacé. Le script applique le composant. La valeur par défaut est vide.
nom	Libellé descriptif pour le composant. Ce libellé s'affiche dans le fichier journal et la vue Événements . Utilisez la propriété nom pour identifier le composant ayant provoqué l'événement.
remarque	Commentaire défini par l'utilisateur, qui décrit l'objectif ou l'action du composant.

SubString

Le transformer **SubString** renvoie une sous-chaîne de l'entrée, commençant et se terminant à des emplacements spécifiés.

Le tableau suivant décrit les propriétés du transformer **SubString** :

Propriété	Description
begin	Définit l'emplacement initial. 0 correspond à un démarrage au début de l'entrée.
disabled	Détermine si le script ignore le composant et tous les composants enfants. Utilisez cette propriété pour tester, déboguer et modifier un script. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none">- Sélectionné. Le script ignore le composant.- Effacé. Le script applique le composant. La valeur par défaut est vide.
end	Définit l'emplacement de fin.
name	Libellé descriptif pour le composant. Ce libellé s'affiche dans le fichier journal et la vue Événements . Utilisez la propriété nom pour identifier le composant ayant provoqué l'événement.
remark	Commentaire défini par l'utilisateur, qui décrit l'objectif ou l'action du composant.

ToFloat

Le transformer **ToFloat** convertit un nombre à virgule flottante d'une représentation de chaîne ASCII en nombre binaire. La conversion est effectuée dans le codage de sortie dans l'ordre des octets de sortie.

Le tableau suivant décrit les propriétés du transformer **ToFloat** :

Propriété	Description
disabled	Détermine si le script ignore le composant et tous les composants enfants. Utilisez cette propriété pour tester, déboguer et modifier un script. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none">- Sélectionné. Le script ignore le composant.- Effacé. Le script applique le composant. La valeur par défaut est vide.
name	Libellé descriptif pour le composant. Ce libellé s'affiche dans le fichier journal et la vue Événements . Utilisez la propriété nom pour identifier le composant ayant provoqué l'événement.
optional	Détermine si la défaillance d'un composant entraîne l'échec du composant parent. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none">- Sélectionné. La défaillance du composant n'entraîne pas l'échec du composant parent.- Effacé. La défaillance du composant entraîne l'échec du composant parent. Par défaut, la valeur est vide. Pour plus d'informations sur l'échec des composants, consultez la section "Traitement des échecs" à la page 404 .
remark	Commentaire défini par l'utilisateur, qui décrit l'objectif ou l'action du composant.
size	Détermine la taille du nombre de sortie. La propriété size comprend les options suivantes : <ul style="list-style-type: none">- single_precision_32_bit- double_precision_64_bit La valeur par défaut est single_precision_32_bit.

Remarque: Ce composant ne prend pas en charge l'encodage d'entrée UTF-16LE.

ToInteger

Le transformer **ToInteger** convertit un nombre d'une représentation de chaîne ASCII à un nombre entier binaire. La chaîne d'entrée peut être décimale, octale ou hexadécimale. La conversion est effectuée dans le codage de sortie dans l'ordre des octets de sortie.

Le tableau suivant décrit les propriétés du transformer **ToInteger** :

Propriété	Description
disabled	Détermine si le script ignore le composant et tous les composants enfants. Utilisez cette propriété pour tester, déboguer et modifier un script. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none">- Sélectionné. Le script ignore le composant.- Effacé. Le script applique le composant. La valeur par défaut est vide.
from_base	Définit la base de l'entrée. La propriété to_base comprend les options suivantes : <ul style="list-style-type: none">- décimal. Base 10.- hexadécimal. Base 16 à l'aide des majuscules A à F.- minuscule hexadécimal. Base 16 à l'aide des minuscules a à f.- octal. Base 8. La valeur par défaut est « décimal ».
name	Libellé descriptif pour le composant. Ce libellé s'affiche dans le fichier journal et la vue Événements . Utilisez la propriété nom pour identifier le composant ayant provoqué l'événement.

Propriété	Description
optional	Détermine si la défaillance d'un composant entraîne l'échec du composant parent. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none"> - Sélectionné. La défaillance du composant n'entraîne pas l'échec du composant parent. - Effacé. La défaillance du composant entraîne l'échec du composant parent. Par défaut, la valeur est vide. Pour plus d'informations sur l'échec des composants, consultez la section "Traitement des échecs" à la page 404 .
remark	Commentaire défini par l'utilisateur, qui décrit l'objectif ou l'action du composant.
signed	Détermine si le numéro d'entrée a un signe. La propriété to_base comprend les options suivantes : <ul style="list-style-type: none"> - Sélectionné. Le nombre de sortie a un signe. - Vide. Le nombre de sortie n'a pas de signe. Par défaut, la valeur est vide.
size	Définit la taille en octets de la représentation binaire. Les valeurs prises en compte vont de 1 à 8.

Remarque: Ce composant ne prend pas en charge l'encodage d'entrée UTF-16LE.

ToPackDecimal

Le transformateur **ToPackDecimal** convertit un nombre d'une représentation de chaîne ASCII en décimaux condensés. La conversion est effectuée dans le codage de sortie dans l'ordre des octets de sortie.

Le tableau suivant décrit les propriétés du transformateur **ToPackDecimal** :

Propriété	Description
disabled	Détermine si le script ignore le composant et tous les composants enfants. Utilisez cette propriété pour tester, déboguer et modifier un script. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none"> - Sélectionné. Le script ignore le composant. - Effacé. Le script applique le composant. La valeur par défaut est vide.
name	Libellé descriptif pour le composant. Ce libellé s'affiche dans le fichier journal et la vue Événements . Utilisez la propriété nom pour identifier le composant ayant provoqué l'événement.
optional	Détermine si la défaillance d'un composant entraîne l'échec du composant parent. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none"> - Sélectionné. La défaillance du composant n'entraîne pas l'échec du composant parent. - Effacé. La défaillance du composant entraîne l'échec du composant parent. Par défaut, la valeur est vide. Pour plus d'informations sur l'échec des composants, consultez la section "Traitement des échecs" à la page 404 .
remark	Commentaire défini par l'utilisateur, qui décrit l'objectif ou l'action du composant.
unsigned	Détermine si le décimal condensé est signé. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none"> - Sélectionné. Le décimal condensé n'est pas signé. - Vide. Le décimal condensé est signé. Par défaut, la valeur est vide.

Remarque: Ce composant ne prend pas en charge l'encodage d'entrée UTF-16LE.

TransformationStartTime

Le transformeur **TransformationStartTime** produit en sortie la date et l'heure lors du démarrage de l'exécution de la transformation.

Le transformeur copie la date et l'heure à partir de la variable *VarSystem* et formate la sortie conformément à votre spécification.

Le tableau suivant décrit les propriétés du transformeur **TransformationStartTime** :

Propriété	Description
disabled	Détermine si le script ignore le composant et tous les composants enfants. Utilisez cette propriété pour tester, déboguer et modifier un script. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none">- Sélectionné. Le script ignore le composant.- Effacé. Le script applique le composant. La valeur par défaut est vide.
format	Définit le format de date et d'heure. Tapez le format ou sélectionnez une zone de stockage des données qui contient le format. Pour plus d'informations sur les formats pris en charge, voir "DateFormatICU" à la page 268 .
name	Libellé descriptif pour le composant. Ce libellé s'affiche dans le fichier journal et la vue Événements . Utilisez la propriété nom pour identifier le composant ayant provoqué l'événement.
optional	Détermine si la défaillance d'un composant entraîne l'échec du composant parent. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none">- Sélectionné. La défaillance du composant n'entraîne pas l'échec du composant parent.- Effacé. La défaillance du composant entraîne l'échec du composant parent. Par défaut, la valeur est vide. Pour plus d'informations sur l'échec des composants, consultez la section "Traitement des échecs" à la page 404 .
remark	Commentaire défini par l'utilisateur, qui décrit l'objectif ou l'action du composant.

TransformByParser

Le transformateur **TransformByParser** exécute un analyseur sur son texte d'entrée. L'analyseur doit contenir des composants **FindReplaceAnchor** qui marquent les segments du texte à remplacer. Lorsque l'exécution de l'analyseur est terminée, le transformateur effectue les remplacements.

La sortie du transformateur est le texte modifié. Le script ignore toute sortie XML générée par l'analyseur.

Le tableau suivant décrit les propriétés du transformateur **TransformByParser**.

Propriété	Description
désactivé	Détermine si le script ignore le composant et tous les composants enfants. Utilisez cette propriété pour tester, déboguer et modifier un script. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none">- Sélectionné. Le script ignore le composant.- Effacé. Le script applique le composant. La valeur par défaut est vide.
nom	Libellé descriptif pour le composant. Ce libellé s'affiche dans le fichier journal et la vue Événements . Utilisez la propriété nom pour identifier le composant ayant provoqué l'événement.

Propriété	Description
facultatif	Détermine si la défaillance d'un composant entraîne l'échec du composant parent. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none"> - Sélectionné. La défaillance du composant n'entraîne pas l'échec du composant parent. - Effacé. La défaillance du composant entraîne l'échec du composant parent. Par défaut, la valeur est vide. Pour plus d'informations sur l'échec des composants, consultez la section "Traitement des échecs" à la page 404 .
Analyseur	Définit le nom de l'analyseur.
remarque	Commentaire défini par l'utilisateur, qui décrit l'objectif ou l'action du composant.

Exemple en ligne

Pour obtenir un exemple en ligne, ouvrez `<installation>\client\DT\samples\Projects\TransformByParser\TransformByParser.cmw`. L'exemple utilise `TransformByParser` pour remplacer chaque instance de la chaîne `~NL~` par un retour chariot suivi par un saut de ligne.

Remarque: Les exemples sont uniquement disponibles lorsque vous effectuez une installation du client.

Pour exécuter l'exemple `TransformByParser` :

1. Définissez `MyTransformByParser` comme composant de démarrage.
2. Exécutez le transformateur.
3. À l'invite, sélectionnez le fichier source `Report.edi`.

Le transformateur stocke son entrée dans `Results\Transformation of Report.edi`. Vous pouvez comparer la sortie avec la source dans le Bloc-notes.

TransformByProcessor

Le transformer **TransformByProcessor** exécute un processeur de document sur son entrée. La sortie du transformer est la sortie du processeur de document. Pour plus d'informations, voir ["Présentation des processeurs de document" à la page 163](#)

Le tableau suivant décrit les propriétés du transformer **TransformByProcessor** :

Propriété	Description
désactivé	Détermine si le script ignore le composant et tous les composants enfants. Utilisez cette propriété pour tester, déboguer et modifier un script. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none"> - Sélectionné. Le script ignore le composant. - Effacé. Le script applique le composant. La valeur par défaut est vide.
nom	Libellé descriptif pour le composant. Ce libellé s'affiche dans le fichier journal et la vue Événements . Utilisez la propriété nom pour identifier le composant ayant provoqué l'événement.

Propriété	Description
facultatif	Détermine si la défaillance d'un composant entraîne l'échec du composant parent. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none"> - Sélectionné. La défaillance du composant n'entraîne pas l'échec du composant parent. - Effacé. La défaillance du composant entraîne l'échec du composant parent. Par défaut, la valeur est vide. Pour plus d'informations sur l'échec des composants, consultez la section "Traitement des échecs" à la page 404 .
processor	Définit le nom du processeur de document.
remarque	Commentaire défini par l'utilisateur, qui décrit l'objectif ou l'action du composant.

TransformByService

Le transformateur **TransformByService** exécute un service de transformation du processeur de données sur son entrée. La sortie du transformateur est la sortie du service.

Si vous utilisez le transformateur pour appeler un service d'analyseur, la sortie du transformateur est une chaîne XML.

Remarque: Le transformateur **TransformByService** prend en charge des services à entrée unique. Ne l'utilisez pas avec un service qui comporte plusieurs ports d'entrée.

Le tableau suivant décrit les propriétés du transformateur **TransformByService** :

Propriété	Description
désactivé	Détermine si le script ignore le composant et tous les composants enfants. Utilisez cette propriété pour tester, déboguer et modifier un script. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none"> - Sélectionné. Le script ignore le composant. - Effacé. Le script applique le composant. La valeur par défaut est vide.
disable_automatic_encoding	Détermine si le script applique les codages d'entrée et de sortie définis dans le service. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none"> - Sélectionné. Le script ignore les codages d'entrée et de sortie définis dans le service. - Vide. Le script applique les codages d'entrée et de sortie définis dans le service.
nom	Libellé descriptif pour le composant. Ce libellé s'affiche dans le fichier journal et la vue Événements . Utilisez la propriété nom pour identifier le composant ayant provoqué l'événement.
facultatif	Détermine si la défaillance d'un composant entraîne l'échec du composant parent. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none"> - Sélectionné. La défaillance du composant n'entraîne pas l'échec du composant parent. - Effacé. La défaillance du composant entraîne l'échec du composant parent. Par défaut, la valeur est vide. Pour plus d'informations sur l'échec des composants, consultez la section "Traitement des échecs" à la page 404 .
paramètres	Définit une liste de valeurs initiales que le script assigne à des variables définies dans le service. Dans chaque élément de la liste, indiquez le nom d'une variable et sa valeur.

Propriété	Description
remarque	Commentaire défini par l'utilisateur, qui décrit l'objectif ou l'action du composant.
service_name	Définit le nom du service qui est exécuté sur l'entrée.

TransformerPipeline

Le transformeur **TransformerPipeline** applique une séquence de transformeurs imbriqués à son entrée.

Le tableau suivant décrit les propriétés du transformeur **TransformerPipeline** :

Propriété	Description
disabled	Détermine si le script ignore le composant et tous les composants enfants. Utilisez cette propriété pour tester, déboguer et modifier un script. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none"> - Sélectionné. Le script ignore le composant. - Effacé. Le script applique le composant. La valeur par défaut est vide.
name	Libellé descriptif pour le composant. Ce libellé s'affiche dans le fichier journal et la vue Événements . Utilisez la propriété nom pour identifier le composant ayant provoqué l'événement.
optional	Détermine si la défaillance d'un composant entraîne l'échec du composant parent. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none"> - Sélectionné. La défaillance du composant n'entraîne pas l'échec du composant parent. - Effacé. La défaillance du composant entraîne l'échec du composant parent. Par défaut, la valeur est vide. Pour plus d'informations sur l'échec des composants, consultez la section "Traitement des échecs" à la page 404 .
remark	Commentaire défini par l'utilisateur, qui décrit l'objectif ou l'action du composant.

XMLLookupTable

Le composant **XMLLookupTable** indique un fichier XML qui contient une table de recherche. La table est conforme au schéma `lookupTableDefinition.xsd` dans le sous-répertoire `\doc` du répertoire d'installation. La table est utilisée par le transformeur **LookupTransformer**.

Le tableau suivant décrit les propriétés du composant **XMLLookupTable** :

Propriété	Description
xml_file_name	Définit le chemin et le nom du fichier XML.

Le document XML suivant est valide par rapport au schéma :

```
<?xml version="1.0" encoding="windows-1252" ?>
<lt:LookupTable xmlns:lt="http://www.Itemfield.com/Engine/V4/lookupTable"
  matchCase="false">
  <lt:Entry key="1" value="George Washington" />
  <lt:Entry key="2" value="John Adams" />
</lt:LookupTable>
```

Si l'attribut optionnel **matchCase** est **true**, l'attribut **clé** est considéré comme sensible à la casse.

XSLTTransformer

Le transformeur **XSLTTransformer** applique une transformation XSLT au texte d'entrée XML.

Par exemple, vous pouvez utiliser un analyseur pour extraire des données d'un document XML. Une ancre **Contenu** récupère une branche complète et bien structurée de l'arborescence XML. Vous pouvez configurer l'ancre **Contenu** avec **XSLTTransformer**, qui exécute une transformation XSLT sur la branche.

Le tableau suivant décrit les propriétés du transformeur **XSLTTransformer** :

Propriété	Description
désactivé	Détermine si le script ignore le composant et tous les composants enfants. Utilisez cette propriété pour tester, déboguer et modifier un script. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none">- Sélectionné. Le script ignore le composant.- Effacé. Le script applique le composant. La valeur par défaut est vide.
nom	Libellé descriptif pour le composant. Ce libellé s'affiche dans le fichier journal et la vue Événements . Utilisez la propriété nom pour identifier le composant ayant provoqué l'événement.
facultatif	Détermine si la défaillance d'un composant entraîne l'échec du composant parent. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none">- Sélectionné. La défaillance du composant n'entraîne pas l'échec du composant parent.- Effacé. La défaillance du composant entraîne l'échec du composant parent. Par défaut, la valeur est vide. Pour plus d'informations sur l'échec des composants, consultez la section "Traitement des échecs" à la page 404 .
remarque	Commentaire défini par l'utilisateur, qui décrit l'objectif ou l'action du composant.
xslt_file	Définit le chemin et le nom du fichier XSLT.

CHAPITRE 18

Actions

Ce chapitre comprend les rubriques suivantes :

- [Présentation des actions, 297](#)
- [Propriétés standard des actions, 298](#)
- [Documentation de référence du composant Action, 299](#)
- [Documentation de référence du sous-composant Action, 334](#)

Présentation des actions

Les actions sont des composants qui effectuent des opérations sur les données extraites par le script d'un document source. Voici quelques exemples d'actions prises en charge :

- Calculs arithmétiques
- Concaténation de chaînes
- Soumission de formulaires à un serveur Web
- Activation d'un analyseur, d'un sérialiseur ou d'un mappeur secondaire
- Interrogation d'une base de données

La transformation Processeur de données fournit plusieurs actions et vous pouvez définir des actions personnalisées.

Comment les actions fonctionnent-elles ?

Une action prend ses entrées dans les zones de stockage des données actuellement disponibles. Une action seule peut avoir plusieurs entrées.

Si l'action est intégrée dans un analyseur, les zones de stockage des données disponibles sont celles générées par l'analyseur. Dans un sérialiseur, les zones de stockage des données sont celles existant dans l'entrée XML, ainsi que celles générées par le sérialiseur. Pour un mappeur, les zones de stockage des données peuvent être dans l'entrée ou dans la sortie.

L'action effectue des opérations sur l'entrée et génère une sortie. Vous pouvez configurer beaucoup d'actions pour qu'elles stockent leur sortie dans des zones de stockage des données.

Dans la plupart des actions, les zones de stockage des données d'entrée et de sortie doivent avoir des types de données simples. Elles ne doivent pas contenir d'éléments imbriqués. Peu d'actions fonctionnent avec des zones de stockage des données contenant des éléments imbriqués, avec des zones de stockage des données à occurrences multiples ou avec d'autres types spéciaux.

Une action peut avoir des effets supplémentaires, tels qu'écrire dans un fichier, mettre à jour une base de données ou soumettre des données à une application externe.

Comparaison entre les actions et les transformateurs

Certaines actions effectuent des opérations similaires à celles des transformateurs, par exemple, modifier une chaîne ou interroger une base de données. Toutefois, les actions diffèrent des transformateurs dans certains aspects fondamentaux.

Le tableau suivant récapitule les différences :

Opération	Transformateurs	Actions
Entrée	L'entrée d'un transformateur est une chaîne simple.	L'entrée est implémentée par l'action. Une action peut avoir plusieurs entrées. Les entrées peuvent être des zones de stockage des données.
Sortie	La sortie d'un transformateur est une chaîne.	La sortie est implémentée par l'action. Par exemple, une action peut créer une zone de stockage des données de sortie.
Effets secondaires	Un transformateur n'a pas d'effet secondaire autre que modifier la chaîne d'entrée.	Une action peut avoir des effets secondaires, comme mettre à jour une base de données.

Définition des actions

Modifiez le script pour définir une action. Vous pouvez insérer les actions dans la ligne de composants **contient** telle que **Analyseur**, **Sérialiseur**, **Mappeur**, **Groupe** ou **RepeatingGroup**. En bref, vous pouvez insérer des actions dans tout emplacement où vous pouvez insérer des ancres, des ancres de sérialisation ou des ancres du mappeur.

Les actions sont exécutées en séquence avec les ancres que vous avez spécifiées dans le même emplacement. Dans un analyseur, vous pouvez définir la propriété **phase** d'une action qui détermine si elle s'exécute lors de l'étape initiale, principale ou finale du processus d'analyse. Pour plus d'informations, consultez la section ["Phases de recherche" à la page 211](#).

Propriétés standard des actions

Le tableau suivant décrit les propriétés standard des actions :

Propriété	Description
disabled	Détermine si le script ignore le composant et tous les composants enfants. Utilisez cette propriété pour tester, déboguer et modifier un script. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none">- Sélectionné. Le script ignore le composant.- Effacé. Le script applique le composant. La valeur par défaut est vide.
name	Libellé descriptif pour le composant. Ce libellé s'affiche dans le fichier journal et la vue Événements . Utilisez la propriété nom pour identifier le composant ayant provoqué l'événement.

Propriété	Description
on_fail	<p>Action à effectuer si le composant échoue. Vous pouvez choisir l'une des options suivantes :</p> <ul style="list-style-type: none"> - Vide. N'effectuer aucune action. - CustomLog. Écrire dans le journal utilisateur. - LogError. Écrire un message d'erreur dans le journal du moteur. - LogInfo. Écrire un message d'information dans le journal du moteur. - LogWarning. Écrire un message d'avertissement dans le journal du moteur. - NotifyFailure. Envoyer une notification. <p>Par défaut, la valeur est vide. Pour plus d'informations sur la gestion des défaillances de composants, consultez la section "Traitement des échecs" à la page 404.</p>
optional	<p>Détermine si la défaillance d'un composant entraîne l'échec du composant parent. Vous pouvez choisir l'une des options suivantes :</p> <ul style="list-style-type: none"> - Sélectionné. La défaillance du composant n'entraîne pas l'échec du composant parent. - Effacé. La défaillance du composant entraîne l'échec du composant parent. <p>Par défaut, la valeur est vide. Pour plus d'informations sur l'échec des composants, consultez la section "Traitement des échecs" à la page 404.</p>
phase	<p>Détermine les situations dans lesquelles le script traite le composant. Vous pouvez choisir l'une des options suivantes :</p> <ul style="list-style-type: none"> - initiale. Le script traite le composant pendant la phase initiale. - principale. Le script traite le composant pendant la phase principale. - finale. Le script traite le composant pendant la phase finale. <p>Pour plus d'informations, voir "Comment un analyseur recherche-t-il les ancrés ?" à la page 210</p>
remark	Commentaire défini par l'utilisateur, qui décrit l'objectif ou l'action du composant.

Documentation de référence du composant Action

Les actions effectuent des opérations dans le système telles que le téléchargement d'un fichier depuis un emplacement distant ou la validation d'une valeur.

AddEventAction

L'action **AddEventAction** ajoute un message au journal des événements.

Le tableau suivant décrit les propriétés de l'action **AddEventAction** :

Propriété	Description
désactivé	<p>Détermine si le script ignore le composant et tous les composants enfants. Utilisez cette propriété pour tester, déboguer et modifier un script. Vous pouvez choisir l'une des options suivantes :</p> <ul style="list-style-type: none"> - Sélectionné. Le script ignore le composant. - Effacé. Le script applique le composant. <p>La valeur par défaut est vide.</p>
message	Définit la chaîne de message.
nom	<p>Libellé descriptif pour le composant. Ce libellé s'affiche dans le fichier journal et la vue Événements. Utilisez la propriété nom pour identifier le composant ayant provoqué l'événement.</p>

Propriété	Description
on_fail	<p>Action à effectuer si le composant échoue. Vous pouvez choisir l'une des options suivantes :</p> <ul style="list-style-type: none"> - Vide. N'effectuer aucune action. - CustomLog. Écrire dans le journal utilisateur. - LogError. Écrire un message d'erreur dans le journal du moteur. - LogInfo. Écrire un message d'information dans le journal du moteur. - LogWarning. Écrire un message d'avertissement dans le journal du moteur. - NotifyFailure. Envoyer une notification. <p>Par défaut, la valeur est vide. Pour plus d'informations sur la gestion des défaillances de composants, consultez la section "Traitement des échecs" à la page 404.</p>
phase	<p>Détermine les situations dans lesquelles le script traite le composant. Vous pouvez choisir l'une des options suivantes :</p> <ul style="list-style-type: none"> - initiale. Le script traite le composant pendant la phase initiale. - principale. Le script traite le composant pendant la phase principale. - finale. Le script traite le composant pendant la phase finale. <p>Pour plus d'informations, voir "Comment un analyseur recherche-t-il les ancrés ?" à la page 210 La valeur par défaut est Principale.</p>
remarque	Commentaire défini par l'utilisateur, qui décrit l'objectif ou l'action du composant.
gravité	<p>Détermine le niveau de sévérité d'un message. Vous pouvez choisir l'une des options suivantes :</p> <ul style="list-style-type: none"> - notification - avertissement - échec - erreur fatale <p>La valeur par défaut est Notification.</p>

AggregateValues

L'action **AggregateValues** effectue un calcul sur une agrégation d'une zone de stockage des données à occurrences multiples.

Le tableau suivant décrit les propriétés de l'action **AggregateValues** :

Propriété	Description
aggregation_function	<p>Détermine la fonction à exécuter sur l'agrégation. Vous pouvez choisir l'une des options suivantes :</p> <ul style="list-style-type: none"> - AllEqual. Renvoie <code>True</code> si les valeurs sont toutes les mêmes ou <code>False</code> si elles ne sont pas identiques. - Compteur. Renvoie le nombre d'occurrences de la zone de stockage des données. - Join. Renvoie une liste de toutes les valeurs, séparées par le séparateur spécifié dans la propriété séparateur. - Sum. Renvoie la somme des valeurs.
AllEqual	Définit une option dans la propriété aggregation_function .
Compteur	Définit une option dans la propriété aggregation_function .
data_holder	Définit la zone de stockage des données qui stocke la sortie.

Propriété	Description
désactivé	Détermine si le script ignore le composant et tous les composants enfants. Utilisez cette propriété pour tester, déboguer et modifier un script. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none"> - Sélectionné. Le script ignore le composant. - Effacé. Le script applique le composant. La valeur par défaut est vide.
include_empty_values	Détermine si l'agrégat inclut des occurrences qui ne contiennent aucune donnée. <ul style="list-style-type: none"> - Sélectionné. L'action inclut les occurrences vides. - Effacé. L'action ignore les occurrences vides. La valeur par défaut est Sélectionné.
Jointure	Définit une option dans la propriété aggregation_function .
nom	Libellé descriptif pour le composant. Ce libellé s'affiche dans le fichier journal et la vue Événements . Utilisez la propriété nom pour identifier le composant ayant provoqué l'événement.
on_fail	Action à effectuer si le composant échoue. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none"> - Vide. N'effectuer aucune action. - CustomLog. Écrire dans le journal utilisateur. - LogError. Écrire un message d'erreur dans le journal du moteur. - LogInfo. Écrire un message d'information dans le journal du moteur. - LogWarning. Écrire un message d'avertissement dans le journal du moteur. - NotifyFailure. Envoyer une notification. Par défaut, la valeur est vide. Pour plus d'informations sur la gestion des défaillances de composants, consultez la section "Traitement des échecs" à la page 404 .
facultatif	Détermine si la défaillance d'un composant entraîne l'échec du composant parent. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none"> - Sélectionné. La défaillance du composant n'entraîne pas l'échec du composant parent. - Effacé. La défaillance du composant entraîne l'échec du composant parent. Par défaut, la valeur est vide. Pour plus d'informations sur l'échec des composants, consultez la section "Traitement des échecs" à la page 404 .
phase	Détermine les situations dans lesquelles le script traite le composant. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none"> - initiale. Le script traite le composant pendant la phase initiale. - principale. Le script traite le composant pendant la phase principale. - finale. Le script traite le composant pendant la phase finale. Pour plus d'informations, voir "Comment un analyseur recherche-t-il les ancrés ?" à la page 210 La valeur par défaut est Principale.
remarque	Commentaire défini par l'utilisateur, qui décrit l'objectif ou l'action du composant.
root_element	Détermine la zone de stockage des données à la racine de la branche XML contenant la zone de stockage des données à occurrences multiples. root_element peut être à occurrences uniques ou multiples.
sub_element	Détermine la zone de stockage des données à occurrences multiples pour laquelle l'action calcule une agrégation. Si sub_element n'est pas assigné, l'action calcule l'agrégation de root_element .
Sum	Définit une option dans la propriété aggregation_function .

En fonction du **root_element** que vous configurez, l'action peut agréger des occurrences à différents niveaux de ramification. Par exemple, un document XML possède une structure de type :

```
<Company>
  <Division name="America">
    <Employee>...<Employee>
    <Employee>...<Employee>
    <Employee>...<Employee>
  </Division>
  <Division name="Europe">
    <Employee>...<Employee>
    <Employee>...<Employee>
  </Division>
</Company>
```

Si le **root_element** est *Société* et que vous configurez l'action pour qu'elle compte les occurrences d'Employé, l'action compte tous les éléments *Employé* qui sont des descendants de *Société*. L'action renvoie 5.

Si **root_element** est *Division*, l'action compte le nombre d'occurrences d'Employé dans la *Division* que la transformation est en train de traiter. Quand l'action traite *Amérique*, elle renvoie 3. Lorsqu'elle traite *Europe*, elle renvoie 2.

AppendListItems

L'action **AppendListItems** concatène les chaînes dans une zone de stockage des données à occurrences multiples.

Le tableau suivant décrit les propriétés de l'action **AppendListItems** :

Propriété	Description
désactivé	Détermine si le script ignore le composant et tous les composants enfants. Utilisez cette propriété pour tester, déboguer et modifier un script. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none"> - Sélectionné. Le script ignore le composant. - Effacé. Le script applique le composant. La valeur par défaut est vide.
entrée	Détermine la zone de stockage des données à occurrences multiples pour l'entrée. La zone de stockage des données doit avoir un type de données simple.
nom	Libellé descriptif pour le composant. Ce libellé s'affiche dans le fichier journal et la vue Événements . Utilisez la propriété nom pour identifier le composant ayant provoqué l'événement.
on_fail	Action à effectuer si le composant échoue. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none"> - Vide. N'effectuer aucune action. - CustomLog. Écrire dans le journal utilisateur. - LogError. Écrire un message d'erreur dans le journal du moteur. - LogInfo. Écrire un message d'information dans le journal du moteur. - LogWarning. Écrire un message d'avertissement dans le journal du moteur. - NotifyFailure. Envoyer une notification. Par défaut, la valeur est vide. Pour plus d'informations sur la gestion des défaillances de composants, consultez la section "Traitement des échecs" à la page 404 .
facultatif	Détermine si la défaillance d'un composant entraîne l'échec du composant parent. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none"> - Sélectionné. La défaillance du composant n'entraîne pas l'échec du composant parent. - Effacé. La défaillance du composant entraîne l'échec du composant parent. Par défaut, la valeur est vide. Pour plus d'informations sur l'échec des composants, consultez la section "Traitement des échecs" à la page 404 .

Propriété	Description
sortie	Détermine la zone de stockage des données qui stocke la sortie. La zone de stockage des données doit avoir un type de données simple.
phase	Détermine les situations dans lesquelles le script traite le composant. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none"> - initiale. Le script traite le composant pendant la phase initiale. - principale. Le script traite le composant pendant la phase principale. - finale. Le script traite le composant pendant la phase finale. Pour plus d'informations, voir "Comment un analyseur recherche-t-il les ancres ?" à la page 210 La valeur par défaut est Principale.
remarque	Commentaire défini par l'utilisateur, qui décrit l'objectif ou l'action du composant.

Pour plus d'informations sur la préparation de l'entrée pour cette action, voir ["Mappage à des zones de stockage des données à occurrences multiples" à la page 207](#).

Exemple

Un document source contient le texte suivant séparé par des espaces :

```
H E L L O
```

Lorsque vous analysez le document, vous souhaitez supprimer les espaces et enregistrer le résultat dans un élément XML appelé `Greeting`.

Créez une variable à occurrences multiples appelée `VarLetter`. Créez plusieurs ancres `Contenu` qui extraient les lettres individuelles et stockez-les dans des occurrences de `VarLetter`.

Ensuite, utilisez l'action **AppendListItems** pour concaténer les occurrences de `VarLetter` et stockez le résultat dans l'élément `Greeting`. Le résultat est :

```
<Greeting>HELLO</Greeting>
```

Exemple en ligne

Pour un échantillon en ligne de cette action, ouvrez le projet `samples\Projects\AppendListItems\AppendListItems.cmw`. L'exemple utilise un `RepeatingGroup` pour stocker les valeurs dans une variable à occurrences multiples. Il utilise ensuite une action `AppendListItems` pour concaténer les valeurs.

AppendValues

L'action **AppendValues** concatène les chaînes.

Le tableau suivant décrit les propriétés l'action **AppendValues** :

Propriété	Description
désactivé	Détermine si le script ignore le composant et tous les composants enfants. Utilisez cette propriété pour tester, déboguer et modifier un script. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none"> - Sélectionné. Le script ignore le composant. - Effacé. Le script applique le composant. La valeur par défaut est vide.
entrée	Détermine la liste des zones de stockage des données contenant les valeurs à joindre. Les zones de stockage des données doivent avoir des types de données simples.
nom	Libellé descriptif pour le composant. Ce libellé s'affiche dans le fichier journal et la vue Événements . Utilisez la propriété nom pour identifier le composant ayant provoqué l'événement.
on_fail	Action à effectuer si le composant échoue. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none"> - Vide. N'effectuer aucune action. - CustomLog. Écrire dans le journal utilisateur. - LogError. Écrire un message d'erreur dans le journal du moteur. - LogInfo. Écrire un message d'information dans le journal du moteur. - LogWarning. Écrire un message d'avertissement dans le journal du moteur. - NotifyFailure. Envoyer une notification. Par défaut, la valeur est vide. Pour plus d'informations sur la gestion des défaillances de composants, consultez la section "Traitement des échecs" à la page 404 .
facultatif	Détermine si la défaillance d'un composant entraîne l'échec du composant parent. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none"> - Sélectionné. La défaillance du composant n'entraîne pas l'échec du composant parent. - Effacé. La défaillance du composant entraîne l'échec du composant parent. Par défaut, la valeur est vide. Pour plus d'informations sur l'échec des composants, consultez la section "Traitement des échecs" à la page 404 .
sortie	Détermine la zone de stockage des données qui stocke la sortie. La zone de stockage des données doit avoir un type de données simple.
phase	Détermine les situations dans lesquelles le script traite le composant. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none"> - initiale. Le script traite le composant pendant la phase initiale. - principale. Le script traite le composant pendant la phase principale. - finale. Le script traite le composant pendant la phase finale. Pour plus d'informations, voir "Comment un analyseur recherche-t-il les ancrés ?" à la page 210 La valeur par défaut est Principale.
remarque	Commentaire défini par l'utilisateur, qui décrit l'objectif ou l'action du composant.
skip_unfound_values	Détermine si l'action continue lorsqu'une des zones de stockage des données d'entrée est manquante. <ul style="list-style-type: none"> - Sélectionné. L'action continue. - Effacé. L'action échoue. La valeur par défaut est Sélectionné.

Exemple

Un analyseur a généré le XML suivant :

```
<Name>
  <First>Ron</First>
```



```
<Last>Lehrer</Last>  
<Name>
```

Vous pouvez configurer une action **AppendValues** qui donne en sortie :

```
<FullName>Ron Lehrer</FullName>
```

CalculateValue

Calcule les valeurs numériques ou concatène les valeurs de chaîne.

Pour calculer des valeurs numériques, utilisez les opérateurs suivants entre les paramètres :

- +
- -
- *
- /

Vous pouvez utiliser des parenthèses pour clarifier l'expression numérique. Vous pouvez utiliser des variables ayant les types de données suivants :

- xs:anyType
- xs:anySimpleType
- types de données numériques
- types de données chaîne

Si les paramètres ont tous des types de données numériques ou chaînes numériques, CalculateValue effectue un calcul arithmétique. Les résultats qui ne sont pas des entiers sont arrondis à 14 décimales.

Pour concaténer les chaînes, utilisez l'opérateur signe plus (+) entre les paramètres et les chaînes.

Le tableau suivant décrit les propriétés de l'action **CalculateValue** :

Propriété	Description
désactivé	Détermine si le script ignore le composant et tous les composants enfants. Utilisez cette propriété pour tester, déboguer et modifier un script. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none">- Sélectionné. Le script ignore le composant.- Effacé. Le script applique le composant. La valeur par défaut est vide.
expression	Définit une expression JavaScript. Pour représenter un paramètre d'entrée, utilisez un signe dollar (\$) suivi d'un nombre entier. Pour représenter une chaîne, placez-la entre guillemets simples.
failure_action	Détermine le comportement en cas d'échec. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none">- Ignorer. La transformation continue.- HaltExecution. La transformation s'interrompt. La valeur par défaut est Ignorer.
nom	Libellé descriptif pour le composant. Ce libellé s'affiche dans le fichier journal et la vue Événements . Utilisez la propriété nom pour identifier le composant ayant provoqué l'événement.

Propriété	Description
on_fail	<p>Action à effectuer si le composant échoue. Vous pouvez choisir l'une des options suivantes :</p> <ul style="list-style-type: none"> - Vide. N'effectuer aucune action. - CustomLog. Écrire dans le journal utilisateur. - LogError. Écrire un message d'erreur dans le journal du moteur. - LogInfo. Écrire un message d'information dans le journal du moteur. - LogWarning. Écrire un message d'avertissement dans le journal du moteur. - NotifyFailure. Envoyer une notification. <p>Par défaut, la valeur est vide. Pour plus d'informations sur la gestion des défaillances de composants, consultez la section "Traitement des échecs" à la page 404.</p>
facultatif	<p>Détermine si la défaillance d'un composant entraîne l'échec du composant parent. Vous pouvez choisir l'une des options suivantes :</p> <ul style="list-style-type: none"> - Sélectionné. La défaillance du composant n'entraîne pas l'échec du composant parent. - Effacé. La défaillance du composant entraîne l'échec du composant parent. <p>Par défaut, la valeur est vide. Pour plus d'informations sur l'échec des composants, consultez la section "Traitement des échecs" à la page 404.</p>
params	Définit une liste des zones de stockage des données qui contiennent les paramètres d'entrée.
phase	<p>Détermine les situations dans lesquelles le script traite le composant. Vous pouvez choisir l'une des options suivantes :</p> <ul style="list-style-type: none"> - initiale. Le script traite le composant pendant la phase initiale. - principale. Le script traite le composant pendant la phase principale. - finale. Le script traite le composant pendant la phase finale. <p>Pour plus d'informations, voir "Comment un analyseur recherche-t-il les ancrés ?" à la page 210 La valeur par défaut est Principale.</p>
remarque	Commentaire défini par l'utilisateur, qui décrit l'objectif ou l'action du composant.
résultat	Détermine une zone de stockage des données qui stocke la sortie.

Remarque: Pour plus d'informations sur la syntaxe JavaScript prise en charge par la transformation Processeur de données, consultez la section ["EnsureCondition" à la page 314](#). Pour plus d'informations sur la précision des valeurs `xs:decimal` et `xs:float`, consultez la section ["Précision des données numériques" à la page 194](#).

Exemple

Un analyseur a généré le XML suivant :

```
<ItemOrdered>
  <Name>Gizmo</Name>
  <Quantity>100</Quantity>
  <Price>25<Price>
</ItemOrdered>
```

Vous pouvez utiliser une action **CalculateValue** pour générer la sortie :

```
<ItemOrdered>
  <Name>Gizmo</Name>
  <Quantity>100</Quantity>
  <Price>25<Price>
  <Total>2500</Total>
</ItemOrdered>
```

Définissez les éléments `Nom` et `Quantité` en tant que paramètres d'entrée. Spécifiez l'expression JavaScript `$1 * $2` et stockez le résultat dans l'élément `Total`.

Exemple en ligne

Pour obtenir un échantillon en ligne de cette action, ouvrez le projet `samples\Projects\CalculateValue\CalculateValue.cmw`. L'exemple récupère trois numéros d'un document source et les stocke dans des variables. Il utilise une action `CalculateValue` pour calculer une fonction mathématique des nombres.

CombineValues

L'action **CombineValues** concatène des chaînes.

L'entrée est une liste de zones de stockage des données et de variables. La sortie est une zone de stockage des données à occurrences multiples.

Si l'entrée est une zone de stockage des données à occurrences multiples, l'action **CombineValues** génère une itération pour chaque instance de la zone de stockage des données. À chaque itération, l'action **CombineValues** combine toutes les zones de stockage des données d'entrée et écrit la sortie dans une instance de la zone de stockage des données de sortie.

Le tableau suivant décrit les propriétés de l'action **CombineValues** :

Propriété	Description
désactivé	Détermine si le script ignore le composant et tous les composants enfants. Utilisez cette propriété pour tester, déboguer et modifier un script. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none">- Sélectionné. Le script ignore le composant.- Effacé. Le script applique le composant. La valeur par défaut est vide.
entrée	Définit une liste de zones de stockage des données pour l'entrée. Les zones de stockage des données doivent avoir un type de données simple.
nom	Libellé descriptif pour le composant. Ce libellé s'affiche dans le fichier journal et la vue Événements . Utilisez la propriété nom pour identifier le composant ayant provoqué l'événement.
on_fail	Action à effectuer si le composant échoue. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none">- Vide. N'effectuer aucune action.- CustomLog. Écrire dans le journal utilisateur.- LogError. Écrire un message d'erreur dans le journal du moteur.- LogInfo. Écrire un message d'information dans le journal du moteur.- LogWarning. Écrire un message d'avertissement dans le journal du moteur.- NotifyFailure. Envoyer une notification. Par défaut, la valeur est vide. Pour plus d'informations sur la gestion des défaillances de composants, consultez la section "Traitement des échecs" à la page 404 .
facultatif	Détermine si la défaillance d'un composant entraîne l'échec du composant parent. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none">- Sélectionné. La défaillance du composant n'entraîne pas l'échec du composant parent.- Effacé. La défaillance du composant entraîne l'échec du composant parent. Par défaut, la valeur est vide. Pour plus d'informations sur l'échec des composants, consultez la section "Traitement des échecs" à la page 404 .
sortie	Détermine la zone de stockage des données à occurrences multiples dans laquelle l'action stocke la sortie. La zone de stockage des données doit avoir un type de données simple.

Propriété	Description
phase	Détermine les situations dans lesquelles le script traite le composant. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none"> - initiale. Le script traite le composant pendant la phase initiale. - principale. Le script traite le composant pendant la phase principale. - finale. Le script traite le composant pendant la phase finale. Pour plus d'informations, voir "Comment un analyseur recherche-t-il les ancres ?" à la page 210 La valeur par défaut est Principale.
remarque	Commentaire défini par l'utilisateur, qui décrit l'objectif ou l'action du composant.

Exemple

Dans une variable à occurrences multiples appelée *VarDay*, vous avez stocké la liste lundi, mardi. Dans une variable à occurrences multiples appelée *VarTime*, vous avez stocké matin, après-midi. Dans une variable à occurrence unique appelée *VarSpace*, vous avez stocké un caractère espace.

Supposons que vous exécutiez **CombineValues** sur *VarDay*, *VarSpace* et *VarTime* avec une zone de stockage des données de sortie appelé **DayTime**. La sortie est :

```
<DayTime>Monday morning</DayTime>
<DayTime>Monday afternoon</DayTime>
<DayTime>Tuesday morning</DayTime>
<DayTime>Tuesday afternoon</DayTime>
```

Exemple en ligne

Pour un exemple en ligne de cette action, ouvrez le projet `samples\Projects\CombineValues\CombineValues.cmw`. L'exemple récupère des listes de jours, de mois et années d'un document source. Il utilise une action `CombineValues` pour générer toutes les dates possibles à partir des listes.

CreateList

L'action **createlist** insère des données dans une liste. La sortie est une zone de stockage des données à occurrences multiples contenant la liste. Pour plus d'informations, voir ["Zones de stockage des données à occurrences multiples" à la page 202](#)

Imbriquées dans ce composant, entrez les valeurs des données.

Le tableau suivant décrit les propriétés de l'action **createlist** :

Propriété	Description
data_holder	Définit les zones de stockage des données à occurrences multiples où l'action stocke la liste. La zone de stockage des données doit avoir un type de données simple.
désactivé	Détermine si le script ignore le composant et tous les composants enfants. Utilisez cette propriété pour tester, déboguer et modifier un script. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none"> - Sélectionné. Le script ignore le composant. - Effacé. Le script applique le composant. La valeur par défaut est vide.

Propriété	Description
nom	Libellé descriptif pour le composant. Ce libellé s'affiche dans le fichier journal et la vue Événements . Utilisez la propriété nom pour identifier le composant ayant provoqué l'événement.
on_fail	<p>Action à effectuer si le composant échoue. Vous pouvez choisir l'une des options suivantes :</p> <ul style="list-style-type: none"> - Vide. N'effectuer aucune action. - CustomLog. Écrire dans le journal utilisateur. - LogError. Écrire un message d'erreur dans le journal du moteur. - LogInfo. Écrire un message d'information dans le journal du moteur. - LogWarning. Écrire un message d'avertissement dans le journal du moteur. - NotifyFailure. Envoyer une notification. <p>Par défaut, la valeur est vide. Pour plus d'informations sur la gestion des défaillances de composants, consultez la section "Traitement des échecs" à la page 404.</p>
phase	<p>Détermine les situations dans lesquelles le script traite le composant. Vous pouvez choisir l'une des options suivantes :</p> <ul style="list-style-type: none"> - initiale. Le script traite le composant pendant la phase initiale. - principale. Le script traite le composant pendant la phase principale. - finale. Le script traite le composant pendant la phase finale. <p>Pour plus d'informations, voir "Comment un analyseur recherche-t-il les ancrés ?" à la page 210 La valeur par défaut est Principale.</p>
remarque	Commentaire défini par l'utilisateur, qui décrit l'objectif ou l'action du composant.

Exemple

Si les valeurs des données d'entrée sont

```
Jack
Jennie
Larissa
```

l'action peut créer la sortie suivante :

```
<Name>
  <First>Jack</First>
  <First>Jennie</First>
  <First>Larissa</First>
</Name>
```

CustomLog

L'action **CustomLog** peut être utilisée comme valeur de la propriété **on_fail**. Quand un échec se produit, l'action **CustomLog** exécute un sérialiseur qui prépare un message de journalisation. Le système écrit le message dans un emplacement de sortie spécifié.

Le tableau suivant décrit les propriétés de l'action **CustomLog** :

Propriété	Description
run_serializer	Détermine le sérialiseur qui prépare le message de journalisation. Définissez un sérialiseur dans cet emplacement ou entrez le nom d'un sérialiseur défini globalement.
sortie	Détermine l'emplacement de la sortie. La propriété sortie possède les options suivantes : <ul style="list-style-type: none"> - OutputDataHolder. Écrit dans une zone de stockage des données. - OutputFile. Écrit dans un fichier. - OutputPort. Définit le nom d'un AdditionalOutputPort dans lequel les données sont écrites. - ResultFile. Écrit dans le fichier de résultats par défaut de la transformation. - StandardErrorLog. Écrit dans le journal utilisateur. La valeur par défaut est StandardErrorLog. Pour plus d'informations sur ces options, consultez les sections " Documentation de référence du sous-composant Action " à la page 334 et " Traitement des échecs " à la page 404.

Pour plus d'informations sur la propriété **on_fail**, consultez la section "[Traitement des échecs](#)" à la page 404.

DateAddICU

L'action **DateAddICU** incrémente une date.

Le tableau suivant décrit les propriétés de l'action **DateAddICU** :

Propriété	Description
désactivé	Détermine si le script ignore le composant et tous les composants enfants. Utilisez cette propriété pour tester, déboguer et modifier un script. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none"> - Sélectionné. Le script ignore le composant. - Effacé. Le script applique le composant. La valeur par défaut est vide.
input_date	Définit la date à incrémenter.
input_format	Définit une chaîne ou une zone de stockage des données qui définit le format de date, par exemple, dd/MM/yy. Pour plus d'informations, consultez la section " DateFormatICU " à la page 268.
nom	Libellé descriptif pour le composant. Ce libellé s'affiche dans le fichier journal et la vue Événements . Utilisez la propriété nom pour identifier le composant ayant provoqué l'événement.
num_of_days	Définit un entier positif ou négatif ou une zone de stockage des données qui contient le nombre de jours à ajouter.
on_fail	Action à effectuer si le composant échoue. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none"> - Vide. N'effectuer aucune action. - CustomLog. Écrire dans le journal utilisateur. - LogError. Écrire un message d'erreur dans le journal du moteur. - LogInfo. Écrire un message d'information dans le journal du moteur. - LogWarning. Écrire un message d'avertissement dans le journal du moteur. - NotifyFailure. Envoyer une notification. Par défaut, la valeur est vide. Pour plus d'informations sur la gestion des défaillances de composants, consultez la section " Traitement des échecs " à la page 404.

Propriété	Description
facultatif	Détermine si la défaillance d'un composant entraîne l'échec du composant parent. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none"> - Sélectionné. La défaillance du composant n'entraîne pas l'échec du composant parent. - Effacé. La défaillance du composant entraîne l'échec du composant parent. Par défaut, la valeur est vide. Pour plus d'informations sur l'échec des composants, consultez la section "Traitement des échecs" à la page 404 .
sortie	Détermine la zone de stockage des données qui stocke la date de sortie.
phase	Détermine les situations dans lesquelles le script traite le composant. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none"> - initiale. Le script traite le composant pendant la phase initiale. - principale. Le script traite le composant pendant la phase principale. - finale. Le script traite le composant pendant la phase finale. Pour plus d'informations, voir "Comment un analyseur recherche-t-il les ancrés ?" à la page 210 La valeur par défaut est Principale.
remarque	Commentaire défini par l'utilisateur, qui décrit l'objectif ou l'action du composant.

DateDiffICU

L'action **DateDiffICU** calcule la différence entre deux dates.

Le tableau suivant décrit les propriétés de l'action **DateDiffICU** :

Propriété	Description
date1	Définit une chaîne ou une zone de stockage des données qui définit la première date.
date2	Définit une chaîne ou une zone de stockage des données qui définit la seconde date.
date_format1	Définit une chaîne ou une zone de stockage des données qui définit le format de la première date, par exemple, dd/MM/yy. Si vous omettez le format, le format système par défaut est utilisé. Pour plus d'informations, consultez la section "DateFormatICU" à la page 268 .
date_format2	Définit une chaîne ou une zone de stockage des données qui définit le format de la seconde date.
désactivé	Détermine si le script ignore le composant et tous les composants enfants. Utilisez cette propriété pour tester, déboguer et modifier un script. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none"> - Sélectionné. Le script ignore le composant. - Effacé. Le script applique le composant. La valeur par défaut est vide.
nom	Libellé descriptif pour le composant. Ce libellé s'affiche dans le fichier journal et la vue Événements . Utilisez la propriété nom pour identifier le composant ayant provoqué l'événement.

Propriété	Description
on_fail	<p>Action à effectuer si le composant échoue. Vous pouvez choisir l'une des options suivantes :</p> <ul style="list-style-type: none"> - Vide. N'effectuer aucune action. - CustomLog. Écrire dans le journal utilisateur. - LogError. Écrire un message d'erreur dans le journal du moteur. - LogInfo. Écrire un message d'information dans le journal du moteur. - LogWarning. Écrire un message d'avertissement dans le journal du moteur. - NotifyFailure. Envoyer une notification. <p>Par défaut, la valeur est vide. Pour plus d'informations sur la gestion des défaillances de composants, consultez la section "Traitement des échecs" à la page 404.</p>
facultatif	<p>Détermine si la défaillance d'un composant entraîne l'échec du composant parent. Vous pouvez choisir l'une des options suivantes :</p> <ul style="list-style-type: none"> - Sélectionné. La défaillance du composant n'entraîne pas l'échec du composant parent. - Effacé. La défaillance du composant entraîne l'échec du composant parent. <p>Par défaut, la valeur est vide. Pour plus d'informations sur l'échec des composants, consultez la section "Traitement des échecs" à la page 404.</p>
sortie	Définit la zone de stockage des données qui stocke les résultats.
phase	<p>Détermine les situations dans lesquelles le script traite le composant. Vous pouvez choisir l'une des options suivantes :</p> <ul style="list-style-type: none"> - initiale. Le script traite le composant pendant la phase initiale. - principale. Le script traite le composant pendant la phase principale. - finale. Le script traite le composant pendant la phase finale. <p>Pour plus d'informations, voir "Comment un analyseur recherche-t-il les ancrés ?" à la page 210 La valeur par défaut est Principale.</p>
remarque	Commentaire défini par l'utilisateur, qui décrit l'objectif ou l'action du composant.

DownloadFileToDataHolder

L'action **DownloadFileToDataHolder** télécharge un fichier d'un serveur Web et stocke son contenu dans une zone de stockage des données. L'action convertit les symboles en entités XML.

Le tableau suivant décrit les propriétés de l'action **DownloadFileToDataHolder** :

Propriété	Description
désactivé	<p>Détermine si le script ignore le composant et tous les composants enfants. Utilisez cette propriété pour tester, déboguer et modifier un script. Vous pouvez choisir l'une des options suivantes :</p> <ul style="list-style-type: none"> - Sélectionné. Le script ignore le composant. - Effacé. Le script applique le composant. <p>La valeur par défaut est vide.</p>
file_url	Détermine une zone de stockage des données qui stocke l'URL du fichier.
nom	Libellé descriptif pour le composant. Ce libellé s'affiche dans le fichier journal et la vue Événements . Utilisez la propriété nom pour identifier le composant ayant provoqué l'événement.

Propriété	Description
on_fail	<p>Action à effectuer si le composant échoue. Vous pouvez choisir l'une des options suivantes :</p> <ul style="list-style-type: none"> - Vide. N'effectuer aucune action. - CustomLog. Écrire dans le journal utilisateur. - LogError. Écrire un message d'erreur dans le journal du moteur. - LogInfo. Écrire un message d'information dans le journal du moteur. - LogWarning. Écrire un message d'avertissement dans le journal du moteur. - NotifyFailure. Envoyer une notification. <p>Par défaut, la valeur est vide. Pour plus d'informations sur la gestion des défaillances de composants, consultez la section "Traitement des échecs" à la page 404.</p>
facultatif	<p>Détermine si la défaillance d'un composant entraîne l'échec du composant parent. Vous pouvez choisir l'une des options suivantes :</p> <ul style="list-style-type: none"> - Sélectionné. La défaillance du composant n'entraîne pas l'échec du composant parent. - Effacé. La défaillance du composant entraîne l'échec du composant parent. <p>Par défaut, la valeur est vide. Pour plus d'informations sur l'échec des composants, consultez la section "Traitement des échecs" à la page 404.</p>
sortie	Détermine la zone de stockage des données qui stocke les contenus téléchargés.
phase	<p>Détermine les situations dans lesquelles le script traite le composant. Vous pouvez choisir l'une des options suivantes :</p> <ul style="list-style-type: none"> - initiale. Le script traite le composant pendant la phase initiale. - principale. Le script traite le composant pendant la phase principale. - finale. Le script traite le composant pendant la phase finale. <p>Pour plus d'informations, voir "Comment un analyseur recherche-t-il les ancres ?" à la page 210 La valeur par défaut est Principale.</p>
remarque	Commentaire défini par l'utilisateur, qui décrit l'objectif ou l'action du composant.

DumpValues

L'action **DumpValues** est un outil de débogage. Elle écrit des données dans un élément `<DumpValues>...</DumpValues>`. Définissez les zones de stockage des données que vous voulez vider en les imbriquant comme des composants enfants.

Le tableau suivant décrit les propriétés de l'action **DumpValues** :

Propriété	Description
désactivé	<p>Détermine si le script ignore le composant et tous les composants enfants. Utilisez cette propriété pour tester, déboguer et modifier un script. Vous pouvez choisir l'une des options suivantes :</p> <ul style="list-style-type: none"> - Sélectionné. Le script ignore le composant. - Effacé. Le script applique le composant. <p>La valeur par défaut est vide.</p>
nom	<p>Libellé descriptif pour le composant. Ce libellé s'affiche dans le fichier journal et la vue Événements. Utilisez la propriété nom pour identifier le composant ayant provoqué l'événement.</p>

Propriété	Description
on_fail	<p>Action à effectuer si le composant échoue. Vous pouvez choisir l'une des options suivantes :</p> <ul style="list-style-type: none"> - Vide. N'effectuer aucune action. - CustomLog. Écrire dans le journal utilisateur. - LogError. Écrire un message d'erreur dans le journal du moteur. - LogInfo. Écrire un message d'information dans le journal du moteur. - LogWarning. Écrire un message d'avertissement dans le journal du moteur. - NotifyFailure. Envoyer une notification. <p>Par défaut, la valeur est vide. Pour plus d'informations sur la gestion des défaillances de composants, consultez la section "Traitement des échecs" à la page 404.</p>
sortie	<p>Détermine la zone de stockage des données qui stocke le résultat. Vous pouvez choisir l'une des options suivantes :</p> <ul style="list-style-type: none"> - OutputFile - ResultFile - StandardErrorLog <p>La valeur par défaut est ResultFile.</p>
phase	<p>Détermine les situations dans lesquelles le script traite le composant. Vous pouvez choisir l'une des options suivantes :</p> <ul style="list-style-type: none"> - initiale. Le script traite le composant pendant la phase initiale. - principale. Le script traite le composant pendant la phase principale. - finale. Le script traite le composant pendant la phase finale. <p>Pour plus d'informations, voir "Comment un analyseur recherche-t-il les ancrés ?" à la page 210 La valeur par défaut est Principale.</p>
remarque	Commentaire défini par l'utilisateur, qui décrit l'objectif ou l'action du composant.

EnsureCondition

L'action **EnsureCondition** évalue une expression Javascript booléenne. Si l'expression est `false`, l'action échoue.

Le tableau suivant décrit les propriétés de l'action **EnsureCondition** :

Propriété	Description
condition	<p>Définit une expression JavaScript pour l'évaluation. Dans l'expression, faites référence aux paramètres définis dans params avec un signe dollar (\$) suivi d'un nombre entier. Par exemple, l'expression suivante vérifie si le premier paramètre a pour valeur <code>Ron Lehrer</code> :</p> <pre>\$1 == "Ron Lehrer"</pre>
désactivé	<p>Détermine si le script ignore le composant et tous les composants enfants. Utilisez cette propriété pour tester, déboguer et modifier un script. Vous pouvez choisir l'une des options suivantes :</p> <ul style="list-style-type: none"> - Sélectionné. Le script ignore le composant. - Effacé. Le script applique le composant. <p>La valeur par défaut est vide.</p>
nom	<p>Libellé descriptif pour le composant. Ce libellé s'affiche dans le fichier journal et la vue Événements. Utilisez la propriété nom pour identifier le composant ayant provoqué l'événement.</p>

Propriété	Description
on_fail	<p>Action à effectuer si le composant échoue. Vous pouvez choisir l'une des options suivantes :</p> <ul style="list-style-type: none"> - Vide. N'effectuer aucune action. - CustomLog. Écrire dans le journal utilisateur. - LogError. Écrire un message d'erreur dans le journal du moteur. - LogInfo. Écrire un message d'information dans le journal du moteur. - LogWarning. Écrire un message d'avertissement dans le journal du moteur. - NotifyFailure. Envoyer une notification. <p>Par défaut, la valeur est vide. Pour plus d'informations sur la gestion des défaillances de composants, consultez la section "Traitement des échecs" à la page 404.</p>
params	Définit une liste de zones de stockage des données.
phase	<p>Détermine les situations dans lesquelles le script traite le composant. Vous pouvez choisir l'une des options suivantes :</p> <ul style="list-style-type: none"> - initiale. Le script traite le composant pendant la phase initiale. - principale. Le script traite le composant pendant la phase principale. - finale. Le script traite le composant pendant la phase finale. <p>Pour plus d'informations, voir "Comment un analyseur recherche-t-il les ancres ?" à la page 210 La valeur par défaut est Principale.</p>
remarque	Commentaire défini par l'utilisateur, qui décrit l'objectif ou l'action du composant.

Syntaxe JavaScript standard

Le processeur JavaScript prend en charge les expressions JavaScript standard contenant les fonctions suivantes.

- Les opérateurs unaires et binaires :

```
() + - * / % == != < <= > >= && ||
```

- L'opérateur ternaire ? : .

- Les méthodes suivantes :

```
charAt
indexOf
lastIndexOf
length
join
substring
toString
```

L'application de ces méthodes à un littéral possédant un type de donnée élémentaire exige la mise entre parenthèses de ce littéral, par exemple :

```
123.toString();           //Wrong
(123).toString();        //Right

"Hello, World".substring(3,7); //Wrong
("Hello, World").substring(3,7); //Right
```

- Les fonctions suivantes :

```
Math.ceil
Math.floor
Math.max
Math.min
Math.pow
Math.sqrt
parseFloat
parseInt
```

Le processeur JavaScript ne prend pas en charge les fonctionnalités suivantes :

- Les opérateurs unaires et binaires :

`++ -- typeof void >> >>> << === !== ~ & | ^`

- Les opérateurs d'assignation :

`= += -= *= /= >>= >>>= <<= &= |= ^=`

- L'opérateur virgule (,)
- Les valeurs NaN, null, infinity ou 0 (0 négatif).
- Les types de données autres que string, number et boolean.
- L'objet Date.
- La fonction equalsIgnoreCase.

Extensions JavaScript

Le processeur JavaScript applique les méthodes suivantes qui ne sont pas définies dans le JavaScript standard. Vous pouvez utiliser ces extensions dans n'importe quel emplacement où le script accepte une expression JavaScript.

La plupart des fonctions sont des implémentations JavaScript de transformateurs ou actions.

```
extra.sum(number1, number 2, ...)
```

Renvoie la somme des paramètres.

```
extra.allSame(param1, param2, ...)
```

Renvoie true si tous les paramètres ont la même valeur.

```
rechercher.<lookup_name>(clé)
```

Cette fonction accède à une table de recherche globale par nom.

Dans le script, définissez une table `InlineTable` ou `XmlLookupTable` globale. Ensuite, dans une expression JavaScript, vous pouvez accéder à la table. Par exemple, si vous définissez une `InlineTable` globale appelée `USPresidents`, `lookup.USPresidents (1)` renvoie George Washington.

Pour plus d'informations, voir ["LookupTransformer" à la page 280](#)

```
extra.formatDate(date, input_format, output_format)
```

Formate une date ou un temps. Pour plus d'informations, voir ["DateFormatICU" à la page 268](#)

```
extra.substitute(text, oldSubstring, newSubstring, useRegex)
```

Remplace toutes les instances d'une sous-chaîne avec une autre sous-chaîne. Si `useRegex` est true, la méthode interprète `oldSubstring` comme une expression régulière.

```
extra.formatNumber(number, size_of_integer_part, number_of_decimals, sign,
insert_decimal_point, unit_type)
```

Formate un nombre. Pour plus d'informations, voir ["FormatNumber" à la page 273](#)

```
extra.insertString(text, injections_place, string_to_inject)
```

Insère une chaîne dans le texte. Pour plus d'informations, voir ["InjectString" à la page 279](#)

```
extra.formatTransformationStartTime(format)
```

Sort la date et/ou l'heure à laquelle la transformation a démarré son exécution. Pour plus d'informations, voir ["TransformationStartTime" à la page 292](#)

```
extra.resize(text, size, padding_character, align)
```

Adapte le texte d'entrée à une taille spécifiée, par remplissage ou troncation comme requis. Pour plus d'informations, voir ["Redimensionner" à la page 287](#)

```
extra.dateAdd(date_format, date, days_to_add)
```

Incrémente une date par un nombre de jours donné. Pour plus d'informations sur `date_format`, voir ["DateFormatICU" à la page 268](#).

```
extra.dateAddMonths(date_format, date, months_to_add)
```

Incrémente une date par un nombre de mois donné. Pour plus d'informations sur `date_format`, voir ["DateFormatICU" à la page 268](#).

```
extra.dateAddYears(date_format, date, years_to_add)
```

Incrémente une date par un nombre d'années donné. Pour plus d'informations sur `date_format`, voir ["DateFormatICU" à la page 268](#).

```
extra.dateDiff(date_format1, date1, date_format2, date2)
```

Calcule la différence entre deux dates. Pour plus d'informations, voir ["DateDiffICU" à la page 311](#)

```
extra.createGuid(0)
```

Génère un identifiant GUID. Pour plus d'informations, voir ["CreateGuid" à la page 267](#) Vous devez fournir une valeur de paramètre comme 0.

```
extra.upper(text), extra.lower(text), extra.capitalize(text)
```

Change le texte en majuscules, en minuscules ou seulement la première lettre en majuscule. Pour plus d'informations, veuillez consulter ["ChangeCase" à la page 267](#).

```
extra.rtl2ltr(text)
```

Inverse une chaîne écrite dans une langue écrite de droite à gauche en écriture gauche à droite. Pour plus d'informations, voir ["hebrewBidi" à la page 277](#)

```
extra.trim(text)
```

Supprime les espaces de début et de fin dans le texte. Pour plus d'informations, voir ["RemoveMarginSpace" à la page 285](#)

ExcludItems

L'action **ExcludItems** supprime les valeurs spécifiées du zone de stockage des données à occurrences multiples. Le type de zone de stockage des données doit être simple. Pour exclure des chaînes spécifiques du zone de stockage des données, définissez-les en tant que composants enfants. Pour plus d'informations, consultez la section ["Zones de stockage des données à occurrences multiples" à la page 202](#)

Le tableau suivant décrit les propriétés de l'action **ExcludItems** :

Propriété	Description
data_holder	Définit une zone de stockage des données à occurrences multiples.
désactivé	Détermine si le script ignore le composant et tous les composants enfants. Utilisez cette propriété pour tester, déboguer et modifier un script. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none">- Sélectionné. Le script ignore le composant.- Effacé. Le script applique le composant. La valeur par défaut est vide.

Propriété	Description
nom	Libellé descriptif pour le composant. Ce libellé s'affiche dans le fichier journal et la vue Événements . Utilisez la propriété nom pour identifier le composant ayant provoqué l'événement.
on_fail	Action à effectuer si le composant échoue. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none"> - Vide. N'effectuer aucune action. - CustomLog. Écrire dans le journal utilisateur. - LogError. Écrire un message d'erreur dans le journal du moteur. - LogInfo. Écrire un message d'information dans le journal du moteur. - LogWarning. Écrire un message d'avertissement dans le journal du moteur. - NotifyFailure. Envoyer une notification. Par défaut, la valeur est vide. Pour plus d'informations sur la gestion des défaillances de composants, consultez la section "Traitement des échecs" à la page 404 .
facultatif	Détermine si la défaillance d'un composant entraîne l'échec du composant parent. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none"> - Sélectionné. La défaillance du composant n'entraîne pas l'échec du composant parent. - Effacé. La défaillance du composant entraîne l'échec du composant parent. Par défaut, la valeur est vide. Pour plus d'informations sur l'échec des composants, consultez la section "Traitement des échecs" à la page 404 .
phase	Détermine les situations dans lesquelles le script traite le composant. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none"> - initiale. Le script traite le composant pendant la phase initiale. - principale. Le script traite le composant pendant la phase principale. - finale. Le script traite le composant pendant la phase finale. Pour plus d'informations, voir "Comment un analyseur recherche-t-il les ancrés ?" à la page 210 La valeur par défaut est Principale.
remarque	Commentaire défini par l'utilisateur, qui décrit l'objectif ou l'action du composant.

Mappage

L'action **Mappage** copie une valeur depuis une zone de stockage des données vers un autre.

Le tableau suivant décrit les propriétés de l'action **Mappage** :

Propriété	Description
désactivé	Détermine si le script ignore le composant et tous les composants enfants. Utilisez cette propriété pour tester, déboguer et modifier un script. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none"> - Sélectionné. Le script ignore le composant. - Effacé. Le script applique le composant. La valeur par défaut est vide.
nom	Libellé descriptif pour le composant. Ce libellé s'affiche dans le fichier journal et la vue Événements . Utilisez la propriété nom pour identifier le composant ayant provoqué l'événement.

Propriété	Description
on_fail	<p>Action à effectuer si le composant échoue. Vous pouvez choisir l'une des options suivantes :</p> <ul style="list-style-type: none"> - Vide. N'effectuer aucune action. - CustomLog. Écrire dans le journal utilisateur. - LogError. Écrire un message d'erreur dans le journal du moteur. - LogInfo. Écrire un message d'information dans le journal du moteur. - LogWarning. Écrire un message d'avertissement dans le journal du moteur. - NotifyFailure. Envoyer une notification. <p>Par défaut, la valeur est vide. Pour plus d'informations sur la gestion des défaillances de composants, consultez la section "Traitement des échecs" à la page 404.</p>
facultatif	<p>Détermine si la défaillance d'un composant entraîne l'échec du composant parent. Vous pouvez choisir l'une des options suivantes :</p> <ul style="list-style-type: none"> - Sélectionné. La défaillance du composant n'entraîne pas l'échec du composant parent. - Effacé. La défaillance du composant entraîne l'échec du composant parent. <p>Par défaut, la valeur est vide. Pour plus d'informations sur l'échec des composants, consultez la section "Traitement des échecs" à la page 404.</p>
phase	<p>Détermine les situations dans lesquelles le script traite le composant. Vous pouvez choisir l'une des options suivantes :</p> <ul style="list-style-type: none"> - initiale. Le script traite le composant pendant la phase initiale. - principale. Le script traite le composant pendant la phase principale. - finale. Le script traite le composant pendant la phase finale. <p>Pour plus d'informations, voir "Comment un analyseur recherche-t-il les ancrés ?" à la page 210 La valeur par défaut est Principale.</p>
remarque	Commentaire défini par l'utilisateur, qui décrit l'objectif ou l'action du composant.
source	Détermine la zone de stockage des données source.
source_default	Détermine la zone de stockage des données source par défaut.
cible	Détermine la zone de stockage des données de destination.
transformateurs	Définit une séquence de transformateurs qui modifie la valeur. N'affectez pas cette propriété si la source et la destination sont des éléments XML complexes.
validateurs	Définit une liste de validateurs appliqués aux données source. Pour plus d'informations, consultez la section "Validateurs" à la page 407 .

Lorsque vous copiez une zone de stockage des données ayant un type de données simple, la source et la destination doivent avoir des types de données compatibles. L'action peut appliquer des transformateurs à la valeur copiée.

Lorsque vous copiez une zone de stockage des données à occurrences multiples qui a un type simple et que l'action n'est pas située dans un composant itératif comme **RepeatingGroup**, l'action copie toutes les occurrences du zone de stockage des données.

Lorsque vous copiez une zone de stockage des données qui a un type complexe, la source et la destination doivent avoir des structures internes identiques et des types de données identiques. L'action copie les attributs et les éléments imbriqués.

Exemple en ligne

Pour un échantillon en ligne de cette action, ouvrez le projet `samples\Projects\CopyValue\CopyValue.cmw`. L'exemple utilise une action `Mappage` pour copier un élément complexe qui contient un attribut et des éléments imbriqués.

Notifier

L'action **Notifier** déclenche une notification. Utilisez-la pour insérer un message d'avertissement dans la sortie de la transformation.

Le tableau suivant décrit les propriétés de l'action **Notifier** :

Propriété	Description
désactivé	Détermine si le script ignore le composant et tous les composants enfants. Utilisez cette propriété pour tester, déboguer et modifier un script. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none">- Sélectionné. Le script ignore le composant.- Effacé. Le script applique le composant. La valeur par défaut est vide.
nom	Libellé descriptif pour le composant. Ce libellé s'affiche dans le fichier journal et la vue Événements . Utilisez la propriété nom pour identifier le composant ayant provoqué l'événement.
notifier	Définit une notification. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none">- <code>MandatoryStructureMissing</code>. Un enregistrement obligatoire n'apparaît pas dans l'entrée.- <code>MismatchIDs</code>. L'enregistrement et les identifiants du sous-élément correspondent partiellement.- <code>StructureBelowMinOccurs</code>. Il y a moins d'enregistrements du sous-élément qui correspondent que le nombre défini dans minOccurs.- <code>StructureExceedsMaxOccurs</code>. Il y a plus d'enregistrements du sous-élément qui correspondent que le nombre défini dans maxOccurs.- <code>StructureOutOfSequence</code>. Les enregistrements correspondent aux sous-éléments, mais pas dans l'ordre requis.- <code>UnexpectedRecord</code>. Les enregistrements correspondent aux sous-éléments, mais pas dans la hiérarchie requise.- <code>UnrecognizedRecord</code>. Aucun sous-élément ne correspond à un identifiant de l'enregistrement.- <code>XsdValidationError</code>. L'entrée ne correspond pas aux exigences du schéma. Notification définie par l'utilisateur ou composant NotificationGroup . Message défini par l'utilisateur. Pour plus d'informations, consultez la section "Notifications" à la page 245 .
phase	Détermine les situations dans lesquelles le script traite le composant. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none">- initiale. Le script traite le composant pendant la phase initiale.- principale. Le script traite le composant pendant la phase principale.- finale. Le script traite le composant pendant la phase finale. Pour plus d'informations, voir "Comment un analyseur recherche-t-il les ancres ?" à la page 210 . La valeur par défaut est Finale.
valeur	Définit une valeur à affecter à la variable <code>VarNotificationDetails/Value</code> . NotificationHandler peut inclure la valeur dans sa sortie.

ResetVisitedPages

L'action **ResetVisitedPages** efface la liste des pages visitées des analyseurs secondaires spécifiés. Il autorise les visites multiples sur la même page, même si **reject_recurring_pages** est sélectionné. Utilisez cette action pour publier diverses données d'entrée dans la même page Web. Cette action est utilisée avec la propriété **reject_recurring_pages** d'un composant **Analyseur**.

Le tableau suivant décrit les propriétés de l'action **ResetVisitedPages** :

Propriété	Description
désactivé	Détermine si le script ignore le composant et tous les composants enfants. Utilisez cette propriété pour tester, déboguer et modifier un script. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none">- Sélectionné. Le script ignore le composant.- Effacé. Le script applique le composant. La valeur par défaut est vide.
nom	Libellé descriptif pour le composant. Ce libellé s'affiche dans le fichier journal et la vue Événements . Utilisez la propriété nom pour identifier le composant ayant provoqué l'événement.
on_fail	Action à effectuer si le composant échoue. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none">- Vide. N'effectuer aucune action.- CustomLog. Écrire dans le journal utilisateur.- LogError. Écrire un message d'erreur dans le journal du moteur.- LogInfo. Écrire un message d'information dans le journal du moteur.- LogWarning. Écrire un message d'avertissement dans le journal du moteur.- NotifyFailure. Envoyer une notification. Par défaut, la valeur est vide. Pour plus d'informations sur la gestion des défaillances de composants, consultez la section "Traitement des échecs" à la page 404 .
analyseurs	Définit une liste d'analyseurs. La liste des pages visitées de chaque analyseur est réinitialisée.
phase	Détermine les situations dans lesquelles le script traite le composant. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none">- initiale. Le script traite le composant pendant la phase initiale.- principale. Le script traite le composant pendant la phase principale.- finale. Le script traite le composant pendant la phase finale. Pour plus d'informations, voir "Comment un analyseur recherche-t-il les ancrés ?" à la page 210 La valeur par défaut est Principale.
remarque	Commentaire défini par l'utilisateur, qui décrit l'objectif ou l'action du composant.

RunMapper

L'action **RunMapper** exécute un mappeur en tant que sous-composant d'un analyseur, d'un mappeur ou d'un sérialiseur.

Le tableau suivant décrit les propriétés de l'action **RunMapper** :

Propriété	Description
désactivé	Détermine si le script ignore le composant et tous les composants enfants. Utilisez cette propriété pour tester, déboguer et modifier un script. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none"> - Sélectionné. Le script ignore le composant. - Effacé. Le script applique le composant. La valeur par défaut est vide.
entrée	Définit une zone de stockage des données qui contient le texte XML sur lequel exécuter le mappeur. La zone de stockage des données doit avoir un type de données simple, tel que <code>xs:string</code> . La valeur de la chaîne peut être un texte XML de n'importe quelle complexité. Pour plus d'informations sur la méthode d'exécution d'un mappeur sur une zone de stockage des données ayant un type complexe, consultez la section "EmbeddedMapper" à la page 360 . Si vous omettez cette propriété, le mappeur utilise les zones de stockage des données disponibles dans la portée de l'action. Par exemple, si l'action est imbriquée dans un analyseur, le mappeur est exécuté sur la sortie de l'analyseur. Si l'action est dans un Groupe , elle est exécutée sur la sortie du Groupe .
mappeur	Définit le mappeur. Sélectionnez le nom d'un composant Mappeur existant ou définissez un composant Mappeur à cet emplacement du script. Pour plus d'informations, consultez la section "Documentation de référence du composant Mappeur" à la page 357 .
nom	Libellé descriptif pour le composant. Ce libellé s'affiche dans le fichier journal et la vue Événements . Utilisez la propriété nom pour identifier le composant ayant provoqué l'événement.
on_fail	Action à effectuer si le composant échoue. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none"> - Vide. N'effectuer aucune action. - CustomLog. Écrire dans le journal utilisateur. - LogError. Écrire un message d'erreur dans le journal du moteur. - LogInfo. Écrire un message d'information dans le journal du moteur. - LogWarning. Écrire un message d'avertissement dans le journal du moteur. - NotifyFailure. Envoyer une notification. Par défaut, la valeur est vide. Pour plus d'informations sur la gestion des défaillances de composants, consultez la section "Traitement des échecs" à la page 404 .
facultatif	Détermine si la défaillance d'un composant entraîne l'échec du composant parent. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none"> - Sélectionné. La défaillance du composant n'entraîne pas l'échec du composant parent. - Effacé. La défaillance du composant entraîne l'échec du composant parent. Par défaut, la valeur est vide. Pour plus d'informations sur l'échec des composants, consultez la section "Traitement des échecs" à la page 404 .
phase	Détermine les situations dans lesquelles le script traite le composant. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none"> - initiale. Le script traite le composant pendant la phase initiale. - principale. Le script traite le composant pendant la phase principale. - finale. Le script traite le composant pendant la phase finale. Pour plus d'informations, voir "Comment un analyseur recherche-t-il les ancres ?" à la page 210 La valeur par défaut est Principale.
remarque	Commentaire défini par l'utilisateur, qui décrit l'objectif ou l'action du composant.

RunMapplet

L'action **RunMapplet** exécute un mapplet. La sortie de **RunMapplet** est lue dans la zone de stockage des données spécifiée dans l'action RunMapplet.

Utilisez l'action **RunMapplet** pour effectuer des tâches telles que le masquage des données, la qualité des données, la recherche des données et d'autres activités généralement liées à une transformation, sans besoin de convertir les données en un format relationnel, puis à un format hiérarchique.

Remarque: L'action RunMapplet ne peut être utilisée que pour appeler des mapplets passifs.

Les paramètres de sortie doivent être spécifiés dans le même ordre que celui dans lequel ils s'affichent dans les ports de mapplet cibles.

Le tableau suivant décrit les propriétés de l'action **RunMapplet** :

Propriété	Description
inputs	Spécifie les valeurs d'entrée à envoyer vers le mapplet. Les paramètres d'entrée doivent être spécifiés dans le même ordre que celui dans lequel ils s'affichent dans les ports de mapplet sources.
mapplet_name	Indique le mapplet à exécuter. Remarque: Si vous voulez appeler un mapplet à l'aide de l'action RunMapplet, vous devez commencer par ajouter une référence dans l'onglet Références . Pour plus d'informations, voir "Références" à la page 28 .
outputs	Spécifie l'emplacement de stockage des valeurs de sortie renvoyées depuis le mapplet. Les paramètres de sortie doivent être spécifiés dans le même ordre que celui dans lequel ils s'affichent dans les ports de mapplet cibles. Dans le paramètre OutputLocation , spécifiez les valeurs suivantes : <ul style="list-style-type: none">- data_holder : spécifie l'emplacement de stockage des valeurs renvoyées par le mapplet.- initialization : lorsque vous testez la transformation, la transformation Processeur de données n'exécute pas le mapplet appelé par l'action RunMapplet. Vous pouvez spécifier une valeur à utiliser en tant que chaîne de sortie temporaire lors du test de la transformation au moment de la conception.

RunParser

L'action **RunParser** exécute un analyseur secondaire. La sortie de **RunParser** est ajoutée à la sortie du composant principal qui l'a activée, à savoir un analyseur ou un sérialiseur.

Utilisez l'action **RunParser** pour suivre les liens dans un fichier HTML et exécuter un analyseur secondaire sur les destinations du lien. Dans un sérialiseur, vous pouvez l'utiliser pour analyser des bouts de données non structurées dans l'entrée.

Notez la différence suivante entre l'action **RunParser** et l'ancre **EmbeddedParser** :

- **RunParser** analyse une nouvelle source.
- **EmbeddedParser** analyse une section d'une source existante.

Le tableau suivant décrit les propriétés de l'action **RunParser** :

Propriété	Description
désactivé	Détermine si le script ignore le composant et tous les composants enfants. Utilisez cette propriété pour tester, déboguer et modifier un script. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none"> - Sélectionné. Le script ignore le composant. - Effacé. Le script applique le composant. La valeur par défaut est vide.
exclude_strings	Définit les chaînes qui doivent être absentes de input_source . Si une chaîne spécifiée est présente, l'action RunParser n'accède pas à la source ni n'active l'analyseur secondaire.
include_strings	Définit les chaînes qui doivent être présentes dans la valeur input_source . Si une chaîne spécifiée est manquante, l'action RunParser n'accède pas à la source ni n'active l'analyseur secondaire.
input_source	Définit une zone de stockage des données qui contient un des objets suivants : <ul style="list-style-type: none"> - Si input_source_as_text est sélectionné, input_source contient une chaîne. - Si input_source_as_text est effacée, input_source contient le chemin et le nom du document d'entrée. La valeur par défaut est la variable système <i>VarLinkURL</i> .
input_source_as_text	Détermine le type de données dans la zone de stockage des données input_source . <ul style="list-style-type: none"> - Sélectionné. input_source contient une chaîne de texte. - Effacé. input_source contient un chemin de fichier. La valeur par défaut est Effacé.
nom	Libellé descriptif pour le composant. Ce libellé s'affiche dans le fichier journal et la vue Événements . Utilisez la propriété nom pour identifier le composant ayant provoqué l'événement.
next_Parser	Définit le nom de l'analyseur à exécuter. Un appel récursif au même analyseur est autorisé.
on_fail	Action à effectuer si le composant échoue. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none"> - Vide. N'effectuer aucune action. - CustomLog. Écrire dans le journal utilisateur. - LogError. Écrire un message d'erreur dans le journal du moteur. - LogInfo. Écrire un message d'information dans le journal du moteur. - LogWarning. Écrire un message d'avertissement dans le journal du moteur. - NotifyFailure. Envoyer une notification. Par défaut, la valeur est vide. Pour plus d'informations sur la gestion des défaillances de composants, consultez la section " Traitement des échecs " à la page 404.
facultatif	Détermine si la défaillance d'un composant entraîne l'échec du composant parent. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none"> - Sélectionné. La défaillance du composant n'entraîne pas l'échec du composant parent. - Effacé. La défaillance du composant entraîne l'échec du composant parent. Par défaut, la valeur est vide. Pour plus d'informations sur l'échec des composants, consultez la section " Traitement des échecs " à la page 404.
phase	Détermine les situations dans lesquelles le script traite le composant. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none"> - initiale. Le script traite le composant pendant la phase initiale. - principale. Le script traite le composant pendant la phase principale. - finale. Le script traite le composant pendant la phase finale. Pour plus d'informations, voir " Comment un analyseur recherche-t-il les ancres ? " à la page 210 La valeur par défaut est Principale.

Propriété	Description
pre_processor	Définit un processeur de document à appliquer à la source après le processeur de document défini dans le pre_processor AdditionalInputPort associé.
remarque	Commentaire défini par l'utilisateur, qui décrit l'objectif ou l'action du composant.
tentatives	Définit le nombre de tentatives en cas d'échec de la requête. La valeur par défaut est 0.
seconds_to_wait	Définit l'intervalle en secondes entre chaque tentative. La valeur par défaut est 60.

Exemple

Un fichier HTML a un lien vers un second fichier. Une ancre **Contenu** stocke le chemin de fichier de la destination du lien dans la variable système *VarLinkURL*. L'action **RunParser** accède au fichier de destination et exécute un analyseur secondaire sur celui-ci.

Dans un autre exemple, l'analyseur principal contient une ancre **Alternatives** qui sélectionne un analyseur secondaire en fonction du texte qui se trouve dans le document source. Pour plus d'informations, consultez le ["Alternatives" à la page 217](#).

RunPCWebService

L'action **RunPCWebService** exécute un mapplet *_PowerCenter* à partir d'une transformation Processeur de données. La sortie de **RunPCWebService** est lue dans la zone de stockage des données spécifiée dans l'action **RunPCWebService**.

Remarque: L'action **RunPCWebService** est utilisée pour appeler les mapplets passifs.

Les paramètres de sortie doivent être spécifiés dans le même ordre que celui dans lequel ils s'affichent dans les ports de mapplet cibles.

Le tableau suivant décrit les propriétés de l'action **RunPCWebService** :

Propriété	Description
wsdl	Spécifiez le langage WSDL du service Web à exécuter.
operationName	Indique l'opération à exécuter. Sélectionnez une opération.
inputs	Spécifie les valeurs d'entrée à envoyer vers le mapplet. Les paramètres d'entrée doivent être spécifiés dans le même ordre que celui dans lequel ils s'affichent dans les ports de mapplet sources.
outputs	Spécifie l'emplacement de stockage des valeurs de sortie renvoyées depuis le mapplet. Les paramètres de sortie doivent être spécifiés dans le même ordre que celui dans lequel ils s'affichent dans les ports de mapplet cibles.

RunSerializer

L'action **RunSerializer** exécute un sérialiseur en tant que sous-composant d'un analyseur, d'un mappeur ou d'un sérialiseur. La sortie du sérialiseur est stockée dans une zone de stockage des données.

Le tableau suivant décrit les propriétés de l'action **RunSerializer** :

Propriété	Description
désactivé	Détermine si le script ignore le composant et tous les composants enfants. Utilisez cette propriété pour tester, déboguer et modifier un script. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none"> - Sélectionné. Le script ignore le composant. - Effacé. Le script applique le composant. La valeur par défaut est vide.
entrée	Définit une zone de stockage des données qui contient le texte XML sur lequel le sérialiseur s'exécute. La zone de stockage des données doit avoir un type de données simple, tel que <code>xs:string</code> . La valeur de la chaîne peut être un texte XML de n'importe quelle complexité. Pour plus d'informations sur la méthode d'exécution d'un sérialiseur sur une zone de stockage des données de type complexe, voir "EmbeddedSerializer" à la page 348 . Si vous omettez cette propriété, le sérialiseur utilise les zones de stockage des données disponibles dans la portée de l'action. Par exemple, si l'action est imbriquée dans un analyseur, le sérialiseur est exécuté sur la sortie de l'analyseur. Si l'action est dans un Groupe , elle est exécutée sur la sortie du Groupe .
nom	Libellé descriptif pour le composant. Ce libellé s'affiche dans le fichier journal et la vue Événements . Utilisez la propriété nom pour identifier le composant ayant provoqué l'événement.
on_fail	Action à effectuer si le composant échoue. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none"> - Vide. N'effectuer aucune action. - CustomLog. Écrire dans le journal utilisateur. - LogError. Écrire un message d'erreur dans le journal du moteur. - LogInfo. Écrire un message d'information dans le journal du moteur. - LogWarning. Écrire un message d'avertissement dans le journal du moteur. - NotifyFailure. Envoyer une notification. Par défaut, la valeur est vide. Pour plus d'informations sur la gestion des défaillances de composants, consultez la section "Traitement des échecs" à la page 404 .
facultatif	Détermine si la défaillance d'un composant entraîne l'échec du composant parent. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none"> - Sélectionné. La défaillance du composant n'entraîne pas l'échec du composant parent. - Effacé. La défaillance du composant entraîne l'échec du composant parent. Par défaut, la valeur est vide. Pour plus d'informations sur l'échec des composants, consultez la section "Traitement des échecs" à la page 404 .
sortie	Définit une zone de stockage des données pour la sortie du sérialiseur.
phase	Détermine les situations dans lesquelles le script traite le composant. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none"> - initiale. Le script traite le composant pendant la phase initiale. - principale. Le script traite le composant pendant la phase principale. - finale. Le script traite le composant pendant la phase finale. Pour plus d'informations, voir "Comment un analyseur recherche-t-il les ancres ?" à la page 210 La valeur par défaut est Principale.
remarque	Commentaire défini par l'utilisateur, qui décrit l'objectif ou l'action du composant.
sérialiseur	Définit un sérialiseur. Sélectionnez le nom d'un composant Sérialiseur existant ou définissez un composant Sérialiseur à cet emplacement du script. Pour plus d'informations, consultez la section "Documentation de référence du composant Serializer" à la page 341 .

Exemple en ligne

Pour un échantillon en ligne de cette action, ouvrez le projet `samples\Projects\RunSerializer` \RunSerializer.cmw.

Pour observer comment l'exemple fonctionne, définissez `MainParser` en tant que composant de démarrage et exécutez-le. `MainParser` contient un `RepeatingGroup` qui analyse les paires de noms et les stocke dans des variables. Après chaque itération, le `RepeatingGroup` exécute une action `RunSerializer` qui concatène les variables avec du texte prédéfini. L'action stocke sa sortie dans un élément XML, qui est ajouté à la sortie de l'analyseur.

RunXMap

L'action **RunXMap** exécute un objet XMap comme sous-composant d'un analyseur, d'un mappeur ou d'un sérialiseur.

Le tableau suivant décrit les propriétés de l'action **RunXMap** :

Propriété	Description
désactivé	Détermine si le script ignore le composant et tous les composants enfants. Utilisez cette propriété pour tester, déboguer et modifier un script. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none">- Sélectionné. Le script ignore le composant.- Effacé. Le script applique le composant. La valeur par défaut est vide.
entrée	Entrée de l'objet XMap. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none">- Zone de stockage des données. Utilisez un type complexe qui doit correspondre au type d'entrée XMap.- Chaîne XML. La chaîne XML est l'entrée, ou une zone de stockage des données de type simple comme <code>xs:string</code>. Le type racine XML doit correspondre au type d'entrée XMap. Si vous n'utilisez pas cette propriété, l'objet XMap utilise son entrée par défaut.
nom	Libellé descriptif pour le composant. Ce libellé s'affiche dans le fichier journal et la vue Événements . Utilisez la propriété nom pour identifier le composant ayant provoqué l'événement.
n_fail	Action à effectuer si le composant échoue. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none">- Vide. N'effectuer aucune action.- CustomLog. Écrire dans le journal utilisateur.- LogError. Écrire un message d'erreur dans le journal du moteur.- LogInfo. Écrire un message d'information dans le journal du moteur.- LogWarning. Écrire un message d'avertissement dans le journal du moteur.- NotifyFailure. Envoyer une notification. Par défaut, la valeur est vide. Pour plus d'informations sur la gestion des défaillances de composants, consultez la section "Traitement des échecs" à la page 404 .
facultatif	Détermine si la défaillance d'un composant entraîne l'échec du composant parent. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none">- Sélectionné. La défaillance du composant n'entraîne pas l'échec du composant parent.- Effacé. La défaillance du composant entraîne l'échec du composant parent. Par défaut, la valeur est vide. Pour plus d'informations sur l'échec des composants, consultez la section "Traitement des échecs" à la page 404 .
sortie	Zone de stockage des données de type complexe qui doit correspondre au type de sortie XMap. Si vous omettez cette propriété, l'objet XMap écrit dans sa définition de sortie par défaut.

Propriété	Description
phase	Détermine les situations dans lesquelles le script traite le composant. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none"> - initiale. Le script traite le composant pendant la phase initiale. - principale. Le script traite le composant pendant la phase principale. - finale. Le script traite le composant pendant la phase finale. Pour plus d'informations, voir "Comment un analyseur recherche-t-il les ancrés ?" à la page 210 La valeur par défaut est Principale.
remarque	Commentaire défini par l'utilisateur, qui décrit l'objectif ou l'action du composant.
xmap	Nom du XMap. Vous pouvez choisir le nom d'un objet XMap existant.

SetValue

L'action **SetValue** place des contenus prédéfinis dans une zone de stockage des données. Si la zone de stockage des données est à occurrence unique, le contenu existant est remplacé. Si la zone de stockage des données est à plusieurs occurrences, le contenu défini est ajouté à la fin.

Le tableau suivant décrit les propriétés de l'action **SetValue** :

Propriété	Description
data_holder	Définit une zone de stockage des données qui reçoit la sortie.
disabled	Détermine si le script ignore le composant et tous les composants enfants. Utilisez cette propriété pour tester, déboguer et modifier un script. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none"> - Sélectionné. Le script ignore le composant. - Effacé. Le script applique le composant. La valeur par défaut est vide.
name	Libellé descriptif pour le composant. Ce libellé s'affiche dans le fichier journal et la vue Événements . Utilisez la propriété nom pour identifier le composant ayant provoqué l'événement.
on_fail	Action à effectuer si le composant échoue. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none"> - Vide. N'effectuer aucune action. - CustomLog. Écrire dans le journal utilisateur. - LogError. Écrire un message d'erreur dans le journal du moteur. - LogInfo. Écrire un message d'information dans le journal du moteur. - LogWarning. Écrire un message d'avertissement dans le journal du moteur. - NotifyFailure. Envoyer une notification. Par défaut, la valeur est vide. Pour plus d'informations sur la gestion des défaillances de composants, consultez la section "Traitement des échecs" à la page 404 .
optional	Détermine si la défaillance d'un composant entraîne l'échec du composant parent. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none"> - Sélectionné. La défaillance du composant n'entraîne pas l'échec du composant parent. - Effacé. La défaillance du composant entraîne l'échec du composant parent. Par défaut, la valeur est vide. Pour plus d'informations sur l'échec des composants, consultez la section "Traitement des échecs" à la page 404 .

Propriété	Description
phase	Détermine les situations dans lesquelles le script traite le composant. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none"> - initiale. Le script traite le composant pendant la phase initiale. - principale. Le script traite le composant pendant la phase principale. - finale. Le script traite le composant pendant la phase finale. Pour plus d'informations, voir "Comment un analyseur recherche-t-il les ancrés ?" à la page 210 La valeur par défaut est « principale ».
quote	Définit une chaîne pour le contenu de la zone de stockage des données.
remark	Commentaire défini par l'utilisateur, qui décrit l'objectif ou l'action du composant.
transformers	Définit une liste de transformers qui sont appliqués au contenu avant qu'il soit enregistré dans la zone de stockage des données.

Trier

L'action **Trier** trie les occurrences d'une zone de stockage des données à plusieurs occurrences. La sortie du contenu original du détenteur de la zone de stockage des données. Le tri est sensible à la casse.

Si vous exécutez l'action sur un élément XML qui contient des attributs ou des éléments imbriqués, vous pouvez les utiliser comme clés de tri.

Le tableau suivant décrit les propriétés de l'action **Trier** :

Propriété	Description
by_fields	Définit une liste de clés de tri par ordre de priorité décroissante. Pour chaque champ, sélectionnez la zone de stockage des données et un ordre croissant ou décroissant . Vous pouvez sélectionner une zone de stockage des données à plusieurs occurrences en tant que telle, ou l'un de ses éléments ou attributs imbriqués. Pour effectuer un tri numérique, une clé de tri doit avoir un type de données numérique tel <code>xs:integer</code> .
disabled	Détermine si le script ignore le composant et tous les composants enfants. Utilisez cette propriété pour tester, déboguer et modifier un script. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none"> - Sélectionné. Le script ignore le composant. - Effacé. Le script applique le composant. La valeur par défaut est vide.
name	Libellé descriptif pour le composant. Ce libellé s'affiche dans le fichier journal et la vue Événements . Utilisez la propriété nom pour identifier le composant ayant provoqué l'événement.
on_fail	Action à effectuer si le composant échoue. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none"> - Vide. N'effectuer aucune action. - CustomLog. Écrire dans le journal utilisateur. - LogError. Écrire un message d'erreur dans le journal du moteur. - LogInfo. Écrire un message d'information dans le journal du moteur. - LogWarning. Écrire un message d'avertissement dans le journal du moteur. - NotifyFailure. Envoyer une notification. Par défaut, la valeur est vide. Pour plus d'informations sur la gestion des défaillances de composants, consultez la section "Traitement des échecs" à la page 404 .

Propriété	Description
phase	Détermine les situations dans lesquelles le script traite le composant. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none"> - initiale. Le script traite le composant pendant la phase initiale. - principale. Le script traite le composant pendant la phase principale. - finale. Le script traite le composant pendant la phase finale. Pour plus d'informations, voir "Comment un analyseur recherche-t-il les ancres ?" à la page 210 La valeur par défaut est « principale ».
recurring_element	Définit une zone de stockage des données à trier.
remark	Commentaire défini par l'utilisateur, qui décrit l'objectif ou l'action du composant.

Limitation

Vous ne pouvez pas utiliser l'action **Trier** si une **Clé** est définie dans la zone de stockage des données à occurrences multiples. Pour plus d'informations, consultez la section ["Présentation des Localisateurs, des Clés et de l'Indexation" à la page 364](#).

ValidateValue

L'action **ValidateValue** valide les données XML en fonction d'un ensemble de règles défini dans un objet Règles de validation. Si les données enfreignent les règles, l'action enregistre un rapport de validation dans une zone de stockage des données.

L'entrée de l'action est une zone de stockage des données. Si la zone de stockage des données est la racine d'une branche XML, l'action analyse toute la branche.

Le tableau suivant décrit les propriétés de l'action **ValidateValue** :

Propriété	Description
disabled	Détermine si le script ignore le composant et tous les composants enfants. Utilisez cette propriété pour tester, déboguer et modifier un script. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none"> - Sélectionné. Le script ignore le composant. - Effacé. Le script applique le composant. La valeur par défaut est vide.
errors_found	Définit une zone de stockage des données qui compte le nombre de violations de règle de validation dans l'entrée.
errors_output	Définit une zone de stockage des données dans laquelle l'action stocke le rapport d'erreur de validation XML. Si la zone de stockage des données est de type <code>xs:string</code> , la sortie est une chaîne contenant des balises XML. Si la zone de stockage des données est de type <code>cde:validationErrors</code> , la sortie est une structure contenant les zones de stockage des données imbriquées.
input	Définit la zone de stockage des données d'entrée dont l'action analyse la conformité aux règles de validation.
name	Libellé descriptif pour le composant. Ce libellé s'affiche dans le fichier journal et la vue Événements . Utilisez la propriété nom pour identifier le composant ayant provoqué l'événement.

Propriété	Description
optional	Détermine si la défaillance d'un composant entraîne l'échec du composant parent. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none"> - Sélectionné. La défaillance du composant n'entraîne pas l'échec du composant parent. - Effacé. La défaillance du composant entraîne l'échec du composant parent. Par défaut, la valeur est vide. Pour plus d'informations sur l'échec des composants, consultez la section "Traitement des échecs" à la page 404 .
phase	Détermine les situations dans lesquelles le script traite le composant. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none"> - initiale. Le script traite le composant pendant la phase initiale. - principale. Le script traite le composant pendant la phase principale. - finale. Le script traite le composant pendant la phase finale. Pour plus d'informations, voir "Comment un analyseur recherche-t-il les ancres ?" à la page 210
remark	Commentaire défini par l'utilisateur, qui décrit l'objectif ou l'action du composant.

WriteValue

L'action **WriteValue** écrit la valeur d'une zone de stockage des données dans un emplacement comme un fichier ou une zone de stockage des données de type string.

Si la zone de stockage des données est un élément XML, l'action écrit à la fois l'élément et tout élément ou attribut imbriqué.

Le tableau suivant décrit les propriétés de l'action **WriteValue** :

Propriété	Description
désactivé	Détermine si le script ignore le composant et tous les composants enfants. Utilisez cette propriété pour tester, déboguer et modifier un script. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none"> - Sélectionné. Le script ignore le composant. - Effacé. Le script applique le composant. La valeur par défaut est vide.
entrée	Définit la zone de stockage des données depuis laquelle écrire.
nom	Libellé descriptif pour le composant. Ce libellé s'affiche dans le fichier journal et la vue Événements . Utilisez la propriété nom pour identifier le composant ayant provoqué l'événement.
no_tags	Détermine si le résultat est entouré par des balises XML. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none"> - Sélectionné. Les balises XML sont exclues. Ce n'est valable que si input est une zone de stockage des données simple, qui ne contient aucun élément ou attribut imbriqué. - Vide. Le résultat est entouré par des balises XML. Il s'agit de la valeur par défaut. Par défaut, la valeur est vide.

Propriété	Description
on_fail	<p>Action à effectuer si le composant échoue. Vous pouvez choisir l'une des options suivantes :</p> <ul style="list-style-type: none"> - Vide. N'effectuer aucune action. - CustomLog. Écrire dans le journal utilisateur. - LogError. Écrire un message d'erreur dans le journal du moteur. - LogInfo. Écrire un message d'information dans le journal du moteur. - LogWarning. Écrire un message d'avertissement dans le journal du moteur. - NotifyFailure. Envoyer une notification. <p>Par défaut, la valeur est vide. Pour plus d'informations sur la gestion des défaillances de composants, consultez la section "Traitement des échecs" à la page 404.</p>
facultatif	<p>Détermine si la défaillance d'un composant entraîne l'échec du composant parent. Vous pouvez choisir l'une des options suivantes :</p> <ul style="list-style-type: none"> - Sélectionné. La défaillance du composant n'entraîne pas l'échec du composant parent. - Effacé. La défaillance du composant entraîne l'échec du composant parent. <p>Par défaut, la valeur est vide. Pour plus d'informations sur l'échec des composants, consultez la section "Traitement des échecs" à la page 404.</p>
sortie	<p>Définit l'emplacement de la sortie. Vous pouvez choisir l'une des options suivantes :</p> <ul style="list-style-type: none"> - OutputDataHolder. Écrit dans une zone de stockage des données. - OutputFile. Écrit dans un fichier. Sélectionnez Synchroniser pour verrouiller le fichier de sorte de joindre les données au même fichier. - OutputPort. Définit le nom d'un élément AdditionalOutputPort dans lequel les données sont écrites. - ResultFile. Écrit dans le fichier de résultats par défaut de la transformation. - StandardErrorLog. Écrit dans le journal utilisateur. Pour plus d'informations, voir "Traitement des échecs" à la page 404 <p>La valeur par défaut est ResultFile. Pour plus d'informations sur ces options, voir "Documentation de référence du sous-composant Action" à la page 334.</p>
phase	<p>Détermine les situations dans lesquelles le script traite le composant. Vous pouvez choisir l'une des options suivantes :</p> <ul style="list-style-type: none"> - initiale. Le script traite le composant pendant la phase initiale. - principale. Le script traite le composant pendant la phase principale. - finale. Le script traite le composant pendant la phase finale. <p>Pour plus d'informations, voir "Comment un analyseur recherche-t-il les ancrés ?" à la page 210 La valeur par défaut est Principale.</p>
remarque	Commentaire défini par l'utilisateur, qui décrit l'objectif ou l'action du composant.
transformateurs	Définit une liste de transformateurs qui modifient la valeur avant écriture. L'entrée des transformateurs est la zone de stockage des données entrée complète, y compris les balises XML.

Exemple en ligne

Pour un échantillon en ligne de cette action, ouvrez le projet `samples\Projects\Splitter\Splitter.cmw`.

L'exemple montre comment diviser un fichier en deux fichiers. Un analyseur utilise un **RepeatingGroup** pour extraire les enregistrements d'un fichier HL7. Il utilise une action **Mappage** pour créer un nom de fichier unique pour chaque enregistrement et une action **WriteValue** pour écrire les enregistrements dans les fichiers. Les fichiers de sortie `MyOutput1.txt` et `MyOutput2.txt` sont stockés dans le dossier `Results` du projet.

Remarque: Vous pouvez utiliser un répartiteur pour fractionner des entrées volumineuses. Pour plus d'informations, consultez la section ["Présentation des répartiteurs" à la page 382](#)

XSLTMap

L'action **XSLTMap** exécute une transformation XSLT. L'entrée et la sortie sont des branches d'un document XML. Elles peuvent correspondre à la sortie d'un analyseur ou à l'entrée d'un sérialiseur.

Le tableau suivant décrit les propriétés de l'action **XSLTMap** :

Propriété	Description
désactivé	Détermine si le script ignore le composant et tous les composants enfants. Utilisez cette propriété pour tester, déboguer et modifier un script. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none">- Sélectionné. Le script ignore le composant.- Effacé. Le script applique le composant. La valeur par défaut est vide.
entrée	Définit une zone de stockage des données qui contient l'élément XML à transformer.
nom	Libellé descriptif pour le composant. Ce libellé s'affiche dans le fichier journal et la vue Événements . Utilisez la propriété nom pour identifier le composant ayant provoqué l'événement.
on_fail	Action à effectuer si le composant échoue. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none">- Vide. N'effectuer aucune action.- CustomLog. Écrire dans le journal utilisateur.- LogError. Écrire un message d'erreur dans le journal du moteur.- LogInfo. Écrire un message d'information dans le journal du moteur.- LogWarning. Écrire un message d'avertissement dans le journal du moteur.- NotifyFailure. Envoyer une notification. Par défaut, la valeur est vide. Pour plus d'informations sur la gestion des défaillances de composants, consultez la section "Traitement des échecs" à la page 404 .
facultatif	Détermine si la défaillance d'un composant entraîne l'échec du composant parent. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none">- Sélectionné. La défaillance du composant n'entraîne pas l'échec du composant parent.- Effacé. La défaillance du composant entraîne l'échec du composant parent. Par défaut, la valeur est vide. Pour plus d'informations sur l'échec des composants, consultez la section "Traitement des échecs" à la page 404 .
sortie	Définit une zone de stockage des données pour stocker la sortie.
phase	Détermine les situations dans lesquelles le script traite le composant. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none">- initiale. Le script traite le composant pendant la phase initiale.- principale. Le script traite le composant pendant la phase principale.- finale. Le script traite le composant pendant la phase finale. Pour plus d'informations, voir "Comment un analyseur recherche-t-il les ancres ?" à la page 210 . La valeur par défaut est Principale.
remarque	Commentaire défini par l'utilisateur, qui décrit l'objectif ou l'action du composant.
xslt_file	Définit le fichier XSLT.

Exemple

Supposons que le code XML suivant soit le résultat d'un analyseur :

```
<Person>
  <First>Ron</First>
  <Last>Lehrer</Last>
</Person>
```

Avec un fichier XSLT approprié, vous pouvez utiliser l'action **XSLTMap** pour le convertir en :

```
<Person Name="Lehrer, Ron" />
```

Documentation de référence du sous-composant Action

Les sous-composants d'action servent en tant que valeurs de certaines propriétés d'actions.

OutputDataHolder

Le sous-composant **OutputDataHolder** dirige la sortie vers une zone de stockage des données. Il est utilisé dans la propriété **sortie** de l'action **WriteValue**.

Le tableau suivant décrit les propriétés du sous-composant **OutputDataHolder** :

Propriété	Description
data_holder	Définit la zone de stockage des données de sortie.
nom	Libellé descriptif pour le composant. Ce libellé s'affiche dans le fichier journal et la vue Événements . Utilisez la propriété nom pour identifier le composant ayant provoqué l'événement.
remarque	Commentaire défini par l'utilisateur, qui décrit l'objectif ou l'action du composant.
transformateurs	Définit une séquence de transformateurs modifiant le flux avant l'écriture.

OutputFile

Le sous-composant **OutputFile** dirige la sortie vers un fichier. Il est utilisé dans la propriété **sortie** des actions **DumpValues** et **WriteValue**.

Le tableau suivant décrit les propriétés du sous-composant **OutputFile** :

Propriété	Description
adjonction	Détermine si les données sont ajoutées au contenu existant du fichier. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none">- Sélectionné. Les données sont ajoutées à la fin du contenu existant.- Effacé. Les données écrasent le contenu existant. Par défaut, la valeur est vide.
fichier	Définit une chaîne ou zone de stockage des données qui définit le chemin et le nom de fichier. Le chemin peut être absolu ou relatif. Si le chemin d'accès est relatif, le script résout le chemin d'accès relatif au dossier de sortie de la transformation.
nom	Libellé descriptif pour le composant. Ce libellé s'affiche dans le fichier journal et la vue Événements . Utilisez la propriété nom pour identifier le composant ayant provoqué l'événement.
remarque	Commentaire défini par l'utilisateur, qui décrit l'objectif ou l'action du composant.

ResultFile

Le sous-composant **ResultFile** indique que la sortie est le fichier de sortie normal d'une transformation Processeur de données. Il est utilisé dans les actions **DumpValues** et **WriteValue**.

StandardErrorLog

Le sous-composant **StandardErrorLog** indique que la sortie est le journal utilisateur.

CHAPITRE 19

Sérialiseurs

Ce chapitre comprend les rubriques suivantes :

- [Présentation des sérialiseurs, 336](#)
- [Ancres de sérialisation, 339](#)
- [Propriétés standard des serializers, 341](#)
- [Documentation de référence du composant Serializer, 341](#)
- [Documentation de référence du composant ancre de sérialisation, 343](#)

Présentation des sérialiseurs

Un sérialiseur convertit un fichier XML ou JSON en document de sortie à n'importe quel format. La sérialisation est l'opposé de l'analyse. Par exemple, la sortie d'un sérialiseur peut être un document texte, un document HTML ou même un autre document XML.

Vous pouvez créer un sérialiseur par le biais des méthodes suivantes :

- Inversion de la configuration d'un analyseur existant
- Modification du script et insertion d'un composant **Sérialiseur**

Vous pouvez combiner et inverser un analyseur et modifier le script du sérialiseur qui en résulte.

Il est généralement plus facile de créer un sérialiseur qu'un analyseur car l'entrée XML ou JSON est entièrement structurée. La structure facilite l'identification des données requises et leur écriture, par une procédure séquentielle, dans la sortie. À l'inverse, il est possible qu'un analyseur doive traiter une entrée non structurée ou semi structurée, tâche qui s'avère souvent complexe.

Les principaux composants imbriqués dans un sérialiseur sont des ancres de sérialisation. La fonction des ancres de sérialisation est d'identifier les données XML ou JSON et de les écrire dans la sortie. Les ancres de sérialisation sont similaires aux ancres qui se trouvent dans un analyseur, hormis le fait qu'elles fonctionnent dans la direction opposée.

Contrôle de la manière de fonctionner de la commande Créer le sérialiseur

Lorsque vous exécutez la commande **Créer un sérialiseur**, l'outil Developer convertit les ancres **Contenu** de l'analyseur en ancres de sérialisation **ContentSerializer**.

Par défaut, la commande convertit tout autre texte de l'exemple de source en ancres de sérialisation **StringSerializer**. Si l'autre texte est un contenu squelette, la sortie du sérialiseur contient tout le squelette qui était dans l'exemple de source d'origine.

Par exemple, imaginez que l'analyseur s'exécute dans des documents source délimités par des tabulations ayant la structure suivante :

```
Name (first and last):<tab>Ron Lehrer
```

Partez du principe que les ancres sont définies de la façon suivante :

Texte source	Ancre
Nom	Marqueur
(premier et dernier):<tab>	Non marqué en tant qu'ancre
Ron Lehrer	Contenu

La sortie XML de l'analyseur est la suivante :

```
<FullName>Ron Lehrer<FullName>
```

Générez maintenant un sérialiseur depuis cet analyseur, puis exécutez le sérialiseur sur l'entrée suivante :

```
<FullName>Larissa Chan<FullName>
```

La sortie du sérialiseur est la suivante :

```
Name (first and last):<tab>Larissa Chan
```

Mode de sérialisation

L'exemple de source peut contenir un texte dont vous ne voulez pas dans la sortie du sérialiseur. Dans ce cas, vous pouvez modifier le comportement de la commande **Créer un sérialiseur**, de manière à ne pas générer les ancres de sérialisation **StringSerializer**.

Pour ce faire, définissez la propriété **serialization_mode** du composant **Analyseur**. Les valeurs possibles de **serialization_mode** sont expliquées dans le tableau suivant :

Valeur	Description
Full	La commande Créer un sérialiseur copie le texte non XML dans la configuration du sérialiseur. C'est le comportement par défaut.
Structure	La commande Créer un sérialiseur copie uniquement les délimiteurs du texte non XML dans la configuration du sérialiseur. Sous l'option Structure , vous pouvez sélectionner l'option use_markers . Dans ce cas, la commande Créer un sérialiseur copie le contenu des ancres de Marqueur , mais seulement les délimiteurs des autres textes non XML.

Le tableau suivant illustre les résultats des paramètres **serialization_mode** :

serialization_mode	Comportement	Exemple de sortie du sérialiseur
structure Lorsque use_markers est vide	La commande Créer un sérialiseur convertit : <ul style="list-style-type: none">- Les ancres Content en ancres de sérialisation ContentSerializer- Les délimiteurs d'autres textes dans l'exemple de source en ancres de sérialisation StringSerializer	<tab>Larissa Chan
structure Lorsque use_markers est sélectionné	La commande Créer un sérialiseur convertit : <ul style="list-style-type: none">- Les ancres Content en ancres de sérialisation ContentSerializer- Le texte complet des ancres Marqueur en ancres de sérialisation StringSerializer- Les délimiteurs d'autres textes dans l'exemple de source en ancres de sérialisation StringSerializer	Nom<tab>Larissa Chan
full	La commande Créer un sérialiseur convertit : <ul style="list-style-type: none">- Les ancres Content en ancres de sérialisation ContentSerializer- Tous autres textes de l'exemple de source en ancres de sérialisation StringSerializer	Nom (prénom et nom) :<tab>Larissa Chan

Dépanner un serializer généré automatiquement

Vous pouvez souvent utiliser directement un sérialiseur généré automatiquement. Si nécessaire, vous pouvez éditer le sérialiseur généré automatiquement pour corriger les limitations ou les problèmes que vous y rencontrez.

Les paragraphes suivants répertorient certaines des circonstances caractéristiques dans lesquelles vous devez éditer le sérialiseur, ainsi les étapes d'édition suggérées.

Balise racine

Dans l'onglet Génération XML des paramètres de la transformation Processeur de données, vous pouvez configurer le script pour qu'il enroule un élément racine autour de l'élément racine spécifié dans le schéma de sortie.

Si vous utilisez ensuite la sortie XML en tant qu'entrée d'un sérialiseur généré automatiquement, vous devez définir la propriété **root_tag** du sérialiseur en lui donnant le nom de l'élément racine défini dans les paramètres de la génération XML.

Variables

Si l'analyseur utilise une variable pour stocker les résultats intermédiaires, un sérialiseur généré automatiquement peut échouer. Pour résoudre ce problème, révisez la logique du sérialiseur et retirez la variable si nécessaire.

Composants supplémentaires

La commande **Créer un sérialiseur** inverse les ancres d'un analyseur. Elle n'inverse pas les composants tels que les processeurs de document, les transformateurs ou les actions.

Imaginez, par exemple, qu'un analyseur utilise un processeur de document `PdfToTxt_4` pour convertir des documents source PDF en texte. L'analyseur contient des ancres transformant le texte en XML.

Le sérialiseur généré automatiquement retransforme le XML en texte. Il ne convertit pas le texte en PDF. Pour obtenir une sortie PDF, éditez le sérialiseur et insérez un processeur `XmlToDocument`.

Dans un autre exemple, imaginez qu'un analyseur utilise un transformateur `AddString` pour ajouter un préfixe à la sortie d'une ancre `Contenu`. Le sérialiseur généré automatiquement ne supprime pas le préfixe. Si vous devez le supprimer, vous pouvez insérer un composant, tel qu'un transformateur `Replace`.

Création d'un sérialiseur en modifiant le script

1. Au niveau global du script, double-cliquez sur le symbole de l'ellipse (...), entrez un nom pour le sérialiseur et appuyez sur **ENTRÉE**.
2. À droite du signe d'égalité, double-cliquez sur l'ellipse, sélectionnez un **Sérialiseur** et appuyez sur **ENTRÉE**.
3. Développez l'arborescence sous le composant **Sérialiseur**. Attribuez ses propriétés, comme requis.
4. Ajoutez un schéma qui définit la syntaxe XML de l'entrée du sérialiseur.
5. Dans la ligne **contient**, ajoutez une séquence d'ancres et d'actions de sérialisation imbriquées.
6. Exécutez et testez le sérialiseur et modifiez le script, comme requis.

Exemple en ligne

Pour obtenir un exemple d'un sérialiseur que nous avons créé en modifiant le script, ouvrez le projet `samples\Projects\ManualSerializer\ManualSerializer.cmw`. Vous pouvez exécuter le sérialiseur sur le fichier d'entrée `Example XML of Person.xml`.

Création d'un sérialiseur dans une action `RunSerializer`

En plus de définir un sérialiseur au niveau global, il est possible de définir un sérialiseur au sein d'une action `RunSerializer`.

Ancres de sérialisation

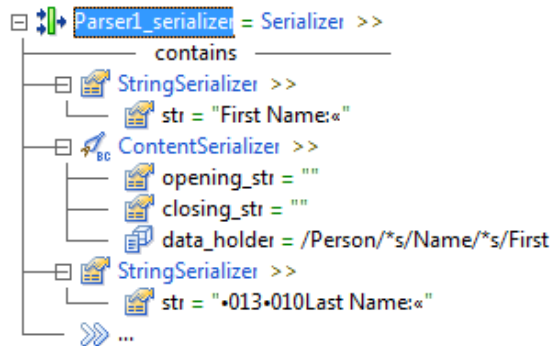
Les ancres de sérialisation sont les principaux composants utilisés dans un sérialiseur. Elles sont similaires aux ancres utilisées dans un analyseur, hormis le fait qu'elles fonctionnent dans la direction opposée. Les ancres lisent les données depuis des emplacements dans le document source et écrivent les données dans un XML. Les ancres de sérialisation lisent des données XML et écrivent les données à des emplacements dans le document de sortie.

Les ancres de sérialisation les plus importantes sont **ContentSerializer** et **StringSerializer**.

- **ContentSerializer** écrit le contenu d'une zone de stockage des données spécifiée dans le document de sortie. C'est l'inverse d'une ancre **Contenu**, qui lit le contenu d'un document source.
- **StringSerializer** écrit une chaîne prédéfinie dans la sortie. C'est l'inverse d'une ancre **Marqueur**, qui repère une chaîne prédéfinie dans un document source.

Exemples d'ancres de sérialisation

L'exemple suivant illustre trois ancres de sérialisation :



La première ancre `StringSerializer` demande au sérialiseur d'écrire le texte suivant dans le document de sortie :

```
First Name:<tab>
```

`ContentSerializer` écrit la valeur de l'élément `Personne/Nom/Prénom` dans la sortie.

Le deuxième `StringSerializer` écrit la chaîne :

```
<newline>Last Name:<tab>
```

Remarque: L'éditeur `IntelliScript` représente les caractères de retour à la ligne et de tabulation respectivement avec des codes ASCII et «.

Supposez maintenant que vous exécutez le sérialiseur sur le code XML suivant :

```
<Person gender="M">
  <Name>
    <First>Ron</First>
    <Last>Lehrer</Last>
  </Name>
  <Id>547329876</Id>
  <Age>27</Age>
</Person>
```

La sortie des ancres de sérialisation illustrées est :

```
First Name<tab>Ron<newline>Last Name<tab>
```

L'affichage de ce texte est :

```
First Name:      Ron
Last Name:
```

Le sérialiseur contient des ancres de sérialisation supplémentaires qui n'apparaissent pas dans l'illustration ci-dessus. La sortie complète du sérialiseur est la suivante :

```
First Name:      Ron
Last Name:      Lehrer
Id:             547329876
Age:            27
Gender:         M
```

Séquence d'ancres de sérialisation

Un sérialiseur exécute les ancres de sérialisation dans l'ordre de leurs définitions.

Les ancres de sérialisation écrivent les données de façon séquentielle, en les ajoutant toujours en fin du document de sortie . Vous pouvez modifier cet ordre en changeant la séquence dans la configuration du sérialiseur.

Vous pouvez intercaler des actions avec les ancres de sérialisation. Les actions sont exécutées comme partie de la séquence.

Propriétés standard des serializers

Le tableau suivant décrit les propriétés standard du composant **Sérialiseur** et de la plupart des ancres de sérialisation :

Propriété	Description
disabled	Détermine si le script ignore le composant et tous les composants enfants. Utilisez cette propriété pour tester, déboguer et modifier un script. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none">- Sélectionné. Le script ignore le composant.- Effacé. Le script applique le composant. La valeur par défaut est vide.
nom	Libellé descriptif pour le composant. Ce libellé s'affiche dans le fichier journal et la vue Événements . Utilisez la propriété nom pour identifier le composant ayant provoqué l'événement.
notifications	Liste de composants Notificationhandler qui gèrent les notifications de composants imbriqués. Pour plus d'informations, voir "Notifications" à la page 423
on_fail	Action à effectuer si le composant échoue. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none">- Vide. N'effectuer aucune action.- CustomLog. Écrire dans le journal utilisateur.- LogError. Écrire un message d'erreur dans le journal du moteur.- LogInfo. Écrire un message d'information dans le journal du moteur.- LogWarning. Écrire un message d'avertissement dans le journal du moteur.- NotifyFailure. Envoyer une notification. Par défaut, la valeur est vide. Pour plus d'informations sur la gestion des défaillances de composants, consultez la section "Traitement des échecs" à la page 404 .
optional	Détermine si la défaillance d'un composant entraîne l'échec du composant parent. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none">- Sélectionné. La défaillance du composant n'entraîne pas l'échec du composant parent.- Effacé. La défaillance du composant entraîne l'échec du composant parent. Par défaut, la valeur est vide. Pour plus d'informations sur l'échec des composants, consultez la section "Traitement des échecs" à la page 404 .
remark	Commentaire défini par l'utilisateur, qui décrit l'objectif ou l'action du composant.

Documentation de référence du composant Serializer

Un **sérialiseur** convertit les documents XML en documents de sortie de n'importe quel format. Il utilise les ancres de sérialisation pour identifier et manipuler des données dans la source.

Sérialiseur

Un **sérialiseur** convertit les documents XML en documents de sortie.

Le tableau suivant décrit les propriétés du composant **Sérialiseur** :

Propriété	Description
default_transformers	Définit une liste de transformateurs que le sérialiseur applique à toutes les données sérialisées.
example_source	Définit un exemple de document XML source. Lorsque vous exécutez le sérialiseur dans l'outil Developer, il fonctionne sur l'exemple de document. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none">- Vide. Vous êtes invité à fournir un document source lorsque vous exécutez le sérialiseur. Valeur par défaut.- InputPort. Définit un port d'entrée.- LocalFile. Définit un fichier sur le système de fichiers local.- Texte. Définit une chaîne.- URL. Définit une URL.
nom	Libellé descriptif pour le composant. Ce libellé s'affiche dans le fichier journal et la vue Événements . Utilisez la propriété nom pour identifier le composant ayant provoqué l'événement.
notifications	Liste de composants Notificationhandler qui gèrent les notifications de composants imbriqués. Pour plus d'informations, voir "Notifications" à la page 423
on_fail	Action à effectuer si le composant échoue. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none">- Vide. N'effectuer aucune action.- CustomLog. Écrire dans le journal utilisateur.- LogError. Écrire un message d'erreur dans le journal du moteur.- LogInfo. Écrire un message d'information dans le journal du moteur.- LogWarning. Écrire un message d'avertissement dans le journal du moteur.- NotifyFailure. Envoyer une notification. Par défaut, la valeur est vide. Pour plus d'informations sur la gestion des défaillances de composants, consultez la section "Traitement des échecs" à la page 404 .
output_file_extension	Définit l'extension du fichier de sortie généré. La valeur par défaut est « .txt ».
remarque	Commentaire défini par l'utilisateur, qui décrit l'objectif ou l'action du composant.
root_tag	Définit le nom d'un élément XML racine qui ne se trouve pas dans le schéma d'entrée pour le sérialiseur. Remarque: Si l'entrée du sérialiseur en XML d'un autre composant du script et les paramètres de transformation Processeur de données ajoutent un élément racine de wrapper autour de la sortie, vous devez définir cette propriété sur le nom de l'élément racine du wrapper.
source	Définit une zone de stockage des données qui contient la source pour la sérialisation. Pour plus d'informations, consultez la section "Présentation des Localisateurs, des Clés et de l'Indexation" à la page 364 .

Propriété	Description
cible	Définit une zone de stockage des données qui contient le résultat de la sérialisation. Pour plus d'informations, consultez la section "Présentation des Localisateurs, des Clés et de l'Indexation" à la page 364.
validate_source_document	Détermine le niveau de validation de source XML que le sérialiseur effectue. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none"> - Partielle. Autorise des déviations par rapport au schéma. Valeur par défaut. - Stricte. Applique le schéma strictement.

Documentation de référence du composant ancre de sérialisation

Les ancrs de sérialisation dans un **sérialiseur** identifie et manipule les données dans le document source.

AlternativeSerializers

L'ancre de sérialisation **AlternativeSerializers** définit un ensemble d'ancres de sérialisation alternatives qui sont imbriquées dans le sérialiseur parent. Définissez un critère pour l'alternative que le sérialiseur doit accepter. Seule l'alternative acceptée affecte la sortie du sérialiseur. Les autres ancrs de sérialisation n'ont aucun effet sur la sortie.

Le tableau suivant décrit les propriétés de l'ancre de sérialisation **AlternativeSerializers** :

Propriété	Description
désactivé	Détermine si le script ignore le composant et tous les composants enfants. Utilisez cette propriété pour tester, déboguer et modifier un script. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none"> - Sélectionné. Le script ignore le composant. - Effacé. Le script applique le composant. La valeur par défaut est vide.
nom	Libellé descriptif pour le composant. Ce libellé s'affiche dans le fichier journal et la vue Événements . Utilisez la propriété nom pour identifier le composant ayant provoqué l'événement.
on_fail	Action à effectuer si le composant échoue. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none"> - Vide. N'effectuer aucune action. - CustomLog. Écrire dans le journal utilisateur. - LogError. Écrire un message d'erreur dans le journal du moteur. - LogInfo. Écrire un message d'information dans le journal du moteur. - LogWarning. Écrire un message d'avertissement dans le journal du moteur. - NotifyFailure. Envoyer une notification. Par défaut, la valeur est vide. Pour plus d'informations sur la gestion des défaillances de composants, consultez la section "Traitement des échecs" à la page 404.

Propriété	Description
facultatif	Détermine si la défaillance d'un composant entraîne l'échec du composant parent. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none"> - Sélectionné. La défaillance du composant n'entraîne pas l'échec du composant parent. - Effacé. La défaillance du composant entraîne l'échec du composant parent. Par défaut, la valeur est vide. Pour plus d'informations sur l'échec des composants, consultez la section "Traitement des échecs" à la page 404 .
remarque	Commentaire défini par l'utilisateur, qui décrit l'objectif ou l'action du composant.
sélecteur	Définit le critère de sélection d'une des ancrs de sérialisation alternatives. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none"> - ScriptOrder. Le sérialiseur teste les ancrs de sérialisation imbriquées dans l'ordre dans lequel elles sont définies dans le script. Il accepte la première qui réussit. Si toutes les ancrs de sérialisation imbriquées échouent, le composant AlternativeSerializers échoue aussi. - NameSwitch. Le sérialiseur cherche l'ancre de sérialisation imbriquée dont la propriété nom est spécifiée dans une zone de stockage des données. Il ignore les autres ancrs de sérialisation imbriquées. Si l'ancre de sérialisation nommée échoue, le composant AlternativeSerializers échoue aussi. La valeur par défaut est ScriptOrder.

Exemple

L'entrée XML peut contenir un élément `Produit` ou un élément `Service`, mais pas les deux. Vous souhaitez sérialiser tout élément se trouvant dans l'entrée.

Définissez une ancre de sérialisation **AlternativeSerializers** et définissez sa propriété **selector** à `ScriptOrder`.

Dans le composant **AlternativeSerializers**, imbriquez deux ancrs de sérialisation **ContentSerializer**. Configurez l'une d'elles pour traiter l'élément `Produit` et l'autre pour traiter `Service`.

ContentSerializer

L'ancre de sérialisation **ContentSerializer** enregistre les données sérialisées dans le document de sortie.

Le tableau suivant décrit les propriétés de l'ancre de sérialisation **ContentSerializer** :

Propriété	Description
allow_empty_values	Détermine si data_holder peut être vide. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none"> - Sélectionné. data_holder peut être vide. - Effacé. data_holder ne peut pas être vide, sinon ContentSerializer échoue. La valeur par défaut est Effacé.
closing_str	Définit la chaîne que l'ancre écrit après le data_holder .
data_holder	Définit la zone de stockage des données qui contient les données sérialisées.

Propriété	Description
désactivé	Détermine si le script ignore le composant et tous les composants enfants. Utilisez cette propriété pour tester, déboguer et modifier un script. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none"> - Sélectionné. Le script ignore le composant. - Effacé. Le script applique le composant. La valeur par défaut est vide.
ignore_default_transformers	Détermine si les transformateurs par défaut du Sérialiseur sont appliqués aux données sérialisées. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none"> - Sélectionné. Les transformateurs par défaut du Sérialiseur ne sont pas appliqués. - Effacé. Les transformateurs par défaut du Sérialiseur sont appliqués. La valeur par défaut est Effacé.
nom	Libellé descriptif pour le composant. Ce libellé s'affiche dans le fichier journal et la vue Événements . Utilisez la propriété nom pour identifier le composant ayant provoqué l'événement.
on_fail	Action à effectuer si le composant échoue. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none"> - Vide. N'effectuer aucune action. - CustomLog. Écrire dans le journal utilisateur. - LogError. Écrire un message d'erreur dans le journal du moteur. - LogInfo. Écrire un message d'information dans le journal du moteur. - LogWarning. Écrire un message d'avertissement dans le journal du moteur. - NotifyFailure. Envoyer une notification. Par défaut, la valeur est vide. Pour plus d'informations sur la gestion des défaillances de composants, consultez la section "Traitement des échecs" à la page 404 .
opening_str	Définit la chaîne que l'ancre écrit avant le contenu du data_holder .
facultatif	Détermine si la défaillance d'un composant entraîne l'échec du composant parent. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none"> - Sélectionné. La défaillance du composant n'entraîne pas l'échec du composant parent. - Effacé. La défaillance du composant entraîne l'échec du composant parent. Par défaut, la valeur est vide. Pour plus d'informations sur l'échec des composants, consultez la section "Traitement des échecs" à la page 404 .
remarque	Commentaire défini par l'utilisateur, qui décrit l'objectif ou l'action du composant.
transformateurs	Définit une liste de transformateurs qui sont appliqués aux données sérialisées.

DelimitedSectionsSerializer

L'ancre de sérialisation **DelimitedSectionsSerializer** traite les sections de données. Les sections de la sortie sont séparées par une chaîne séparatrice définie.

Dans **DelimitedSectionsSerializer**, imbriquez d'autres ancres de sérialisation. Chaque ancre de sérialisation imbriquée sort une seule section.

Le tableau suivant décrit les propriétés de l'ancre de sérialisation **DelimitedSectionsSerializer** :

Propriété	Description
désactivé	Détermine si le script ignore le composant et tous les composants enfants. Utilisez cette propriété pour tester, déboguer et modifier un script. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none"> - Sélectionné. Le script ignore le composant. - Effacé. Le script applique le composant. La valeur par défaut est vide.
nom	Libellé descriptif pour le composant. Ce libellé s'affiche dans le fichier journal et la vue Événements . Utilisez la propriété nom pour identifier le composant ayant provoqué l'événement.
notifications	Liste de composants Notificationhandler qui gèrent les notifications de composants imbriqués. Pour plus d'informations, voir "Notifications" à la page 423
on_fail	Action à effectuer si le composant échoue. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none"> - Vide. N'effectuer aucune action. - CustomLog. Écrire dans le journal utilisateur. - LogError. Écrire un message d'erreur dans le journal du moteur. - LogInfo. Écrire un message d'information dans le journal du moteur. - LogWarning. Écrire un message d'avertissement dans le journal du moteur. - NotifyFailure. Envoyer une notification. Par défaut, la valeur est vide. Pour plus d'informations sur la gestion des défaillances de composants, consultez la section "Traitement des échecs" à la page 404 .
facultatif	Détermine si la défaillance d'un composant entraîne l'échec du composant parent. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none"> - Sélectionné. La défaillance du composant n'entraîne pas l'échec du composant parent. - Effacé. La défaillance du composant entraîne l'échec du composant parent. Par défaut, la valeur est vide. Pour plus d'informations sur l'échec des composants, consultez la section "Traitement des échecs" à la page 404 .
remarque	Commentaire défini par l'utilisateur, qui décrit l'objectif ou l'action du composant.
séparateur	Définit un sérialiseur qui définit la chaîne séparatrice. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none"> - AlternativeSerializers - ContentSerializer - EmbeddedSerializer - GroupSerializer - RepeatingGroupSerializer - StringSerializer - Sérialiseur défini par l'utilisateur La valeur par défaut est StringSerializer.

Propriété	Description
separator_position	<p>Définit la position du séparateur par rapport aux sections. Vous pouvez choisir l'une des options suivantes :</p> <ul style="list-style-type: none"> - après. Écrit un séparateur après chaque section, y compris pour la première section. Par exemple, 1 2 3 4 - autour. Écrit des séparateurs avant et après chaque section, y compris pour la première section. Par exemple, 1 2 3 4 - avant. Écrit un séparateur avant chaque section, y compris pour la première section. Par exemple, 1 2 3 4 - entre. Écrit un séparateur entre les sections consécutives, mais ni avant la première section, ni après la dernière. Par exemple, 1 2 3 4 <p>La valeur par défaut est Avant.</p>
using_placeholders	<p>Détermine si DelimitedSectionsSerializer écrit le séparateur dans une section optionnelle qui manque dans l'entrée XML. Vous pouvez choisir l'une des options suivantes :</p> <ul style="list-style-type: none"> - toujours. Écrit toujours le séparateur d'une section manquante. Par exemple, 1 3 - jamais. N'écrit jamais le séparateur d'une section manquante. Par exemple, 1 3 - si nécessaire. Écrit toujours le séparateur d'une section interne manquante. N'écrit jamais le séparateur d'une section finale manquante. Par exemple, 1 3 <p>La valeur par défaut est Si nécessaire.</p>

Exemple

L'entrée XML contient le curriculum vitae d'un employé. Vous souhaitez écrire les données dans un document texte de sortie au format suivant :

```

-----
Jane Palmer
Employee ID 123456
-----
Professional Experience
...
-----
Education
...

```

Définissez un **DelimitedSectionsSerializer** avec la ligne de tirets comme son **séparateur**. Pour obtenir une ligne de tirets avant chaque section, paramétrez **separator_position** sur **avant**.

Dans **DelimitedSectionsSerializer**, imbriquez trois composants **GroupSerializer**. Le premier **GroupSerializer** écrit la section `Jane Palmer`, le deuxième écrit la section `Experience professionnelle` et ainsi de suite.

Sections facultatives

Dans cet exemple, imaginez que la deuxième section `Expérience professionnelle` manque dans certains documents d'entrée XML, mais que vous vouliez écrire son séparateur dans la sortie, comme suit :

```

-----
Jane Palmer
Employee ID 123456
-----
-----
Education
...

```

Pour gérer cette situation, configurez **DelimitedSectionsSerializer** de la manière suivante :

- Dans la deuxième ancre **GroupSerializer**, sélectionnez la propriété **facultatif**. Cela signifie que si **GroupSerializer** échoue, il ne doit pas entraîner l'échec de **DelimitedSectionsSerializer**.
- Dans **DelimitedSectionsSerializer**, définissez **using_placeholders** sur `toujours`. Cela signifie qu'il faut écrire le séparateur d'une section optionnelle, même si la section elle-même est absente.

Supposez maintenant que, si la section `Experience professionnelle` manque, vous ne voulez pas écrire son séparateur :

```
-----  
Jane Palmer  
Employee ID 123456  
-----  
Education  
...
```

Dans ce cas, configurez le **DelimitedSectionsSerializer** comme suit :

- Dans la deuxième ancre **GroupSerializer**, sélectionnez la propriété **facultatif**.
- Dans **DelimitedSectionsSerializer**, définissez **using_placeholders** sur `jamais`. Cela signifie qu'il ne faut pas écrire le séparateur d'une section manquante.

EmbeddedSerializer

L'ancre de sérialisation **EmbeddedSerializer** active un **Sérialiseur** secondaire qui enregistre ses résultats dans le même document de sortie.

Le tableau suivant décrit les propriétés de l'ancre de sérialisation **EmbeddedSerializer** :

Propriété	Description
désactivé	Détermine si le script ignore le composant et tous les composants enfants. Utilisez cette propriété pour tester, déboguer et modifier un script. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none">- Sélectionné. Le script ignore le composant.- Effacé. Le script applique le composant. La valeur par défaut est vide.
nom	Libellé descriptif pour le composant. Ce libellé s'affiche dans le fichier journal et la vue Événements . Utilisez la propriété nom pour identifier le composant ayant provoqué l'événement.
on_fail	Action à effectuer si le composant échoue. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none">- Vide. N'effectuer aucune action.- CustomLog. Écrire dans le journal utilisateur.- LogError. Écrire un message d'erreur dans le journal du moteur.- LogInfo. Écrire un message d'information dans le journal du moteur.- LogWarning. Écrire un message d'avertissement dans le journal du moteur.- NotifyFailure. Envoyer une notification. Par défaut, la valeur est vide. Pour plus d'informations sur la gestion des défaillances de composants, consultez la section "Traitement des échecs" à la page 404 .
facultatif	Détermine si la défaillance d'un composant entraîne l'échec du composant parent. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none">- Sélectionné. La défaillance du composant n'entraîne pas l'échec du composant parent.- Effacé. La défaillance du composant entraîne l'échec du composant parent. Par défaut, la valeur est vide. Pour plus d'informations sur l'échec des composants, consultez la section "Traitement des échecs" à la page 404 .

Propriété	Description
remarque	Commentaire défini par l'utilisateur, qui décrit l'objectif ou l'action du composant.
schema_connections	Connecte les zones de stockage des données référencées dans le sérialiseur secondaire aux zones de stockage des données référencées dans le sérialiseur parent. La propriété contient une liste de sous-composants Connecter qui définissent les correspondances. Pour plus d'informations, consultez la section " Connecter " à la page 253. Si toutes les zones de stockage des données des sérialiseurs principal et secondaire sont identiques, vous pouvez omettre cette propriété. S'il y a une différence quelconque entre les zones de stockage des données, vous devez connecter les zones de stockage des données de façon explicite, y compris ceux qui sont identiques.
Sérialiseur	Définit le nom d'un sérialiseur secondaire défini au niveau global du script.

Exemple

L'entrée XML est un arbre généalogique. L'entrée contient des éléments `Personne` récursivement imbriqués comme suit :

```
<Person>          <!-- Parent -->
...
  <Person>        <!-- Child -->
    ...
      <Person>    <!-- Grandchild -->
        ...
      </Person>
    </Person>
  </Person>
```

Dans un **Sérialiseur**, un composant **EmbeddedSerializer** peut s'appeler lui-même de façon récursive jusqu'à ce que tous les niveaux d'imbrication soient épuisés.

Dans cet exemple, la propriété **schema_connections** connecte `Personne` à `Personne/Personne`. Ceci donne l'ordre à l'instance secondaire du sérialiseur de traiter un niveau imbriqué de l'entrée. Lorsque les deux éléments `Personne` ont le même type de données, il suffit de connecter uniquement l'élément parent (`Personne`) et non les éléments imbriqués (`Personne/*s/Nom`, `Personne/*s/DateNaiss`, etc.)

GroupSerializer

L'ancre de sérialisation **GroupSerializer** lie ensemble ses ancres de sérialisation imbriquées. Vous pouvez définir les propriétés de **GroupSerializer** qui affectent les membres du groupe.

Le tableau suivant décrit les propriétés de l'ancre de sérialisation **GroupSerializer** :

Propriété	Description
désactivé	Détermine si le script ignore le composant et tous les composants enfants. Utilisez cette propriété pour tester, déboguer et modifier un script. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none"> - Sélectionné. Le script ignore le composant. - Effacé. Le script applique le composant. La valeur par défaut est vide.
nom	Libellé descriptif pour le composant. Ce libellé s'affiche dans le fichier journal et la vue Événements . Utilisez la propriété nom pour identifier le composant ayant provoqué l'événement.
notifications	Liste de composants Notificationhandler qui gèrent les notifications de composants imbriqués. Pour plus d'informations, voir "Notifications" à la page 423
on_fail	Action à effectuer si le composant échoue. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none"> - Vide. N'effectuer aucune action. - CustomLog. Écrire dans le journal utilisateur. - LogError. Écrire un message d'erreur dans le journal du moteur. - LogInfo. Écrire un message d'information dans le journal du moteur. - LogWarning. Écrire un message d'avertissement dans le journal du moteur. - NotifyFailure. Envoyer une notification. Par défaut, la valeur est vide. Pour plus d'informations sur la gestion des défaillances de composants, consultez la section "Traitement des échecs" à la page 404 .
facultatif	Détermine si la défaillance d'un composant entraîne l'échec du composant parent. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none"> - Sélectionné. La défaillance du composant n'entraîne pas l'échec du composant parent. - Effacé. La défaillance du composant entraîne l'échec du composant parent. Par défaut, la valeur est vide. Pour plus d'informations sur l'échec des composants, consultez la section "Traitement des échecs" à la page 404 .
remarque	Commentaire défini par l'utilisateur, qui décrit l'objectif ou l'action du composant.
source	Définit une zone de stockage des données qui contient la source pour la sérialisation. Pour plus d'informations, consultez la section "Présentation des validateurs, des notificateurs et du traitement des échecs" à la page 403 .
cible	Définit une zone de stockage des données qui contient le résultat de la sérialisation. Pour plus d'informations, consultez la section "Présentation des validateurs, des notificateurs et du traitement des échecs" à la page 403 .

RepeatingGroupSerializer

L'ancre de sérialisation **RepeatingGroupSerializer** écrit une structure répétitive dans le document de sortie.

Utilisez un **RepeatingGroupSerializer** lorsque les données XML contiennent une zone de stockage des données à occurrences multiples. Il se répète sur les occurrences du zone de stockage des données et sort les données. Pour plus d'informations, consultez la section ["Zones de stockage des données à occurrences multiples" à la page 202](#).

Dans **RepeatingGroupSerializer**, imbriquez des ancres de sérialisation qui traitent et sortent chaque occurrence du détenteur de données. Vous pouvez définir un séparateur que **RepeatingGroupSerializer** écrit dans la sortie entre les itérations.

Le tableau suivant décrit les propriétés de l'ancre de sérialisation **RepeatingGroupSerializer** :

Propriété	Description
compteur	Définit le nombre d'itérations à exécuter. Si cette propriété est vide, les itérations continuent jusqu'à ce que l'entrée soit terminée.
current_iteration	Définit une zone de stockage des données dans laquelle RepeatingGroupSerializer sort le numéro de l'itération actuelle. Vous pouvez utiliser un ContentSerializer pour écrire le numéro dans la sortie.
désactivé	Détermine si le script ignore le composant et tous les composants enfants. Utilisez cette propriété pour tester, déboguer et modifier un script. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none"> - Sélectionné. Le script ignore le composant. - Effacé. Le script applique le composant. La valeur par défaut est vide.
nom	Libellé descriptif pour le composant. Ce libellé s'affiche dans le fichier journal et la vue Événements . Utilisez la propriété nom pour identifier le composant ayant provoqué l'événement.
notifications	Liste de composants Notificationhandler qui gèrent les notifications de composants imbriqués. Pour plus d'informations, voir "Notifications" à la page 423
on_fail	Action à effectuer si le composant échoue. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none"> - Vide. N'effectuer aucune action. - CustomLog. Écrire dans le journal utilisateur. - LogError. Écrire un message d'erreur dans le journal du moteur. - LogInfo. Écrire un message d'information dans le journal du moteur. - LogWarning. Écrire un message d'avertissement dans le journal du moteur. - NotifyFailure. Envoyer une notification. Par défaut, la valeur est vide. Pour plus d'informations sur la gestion des défaillances de composants, consultez la section "Traitement des échecs" à la page 404 .
on_iteration_fail	Détermine l'action lorsqu'une itération échoue. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none"> - Effacé. Aucune action. - CustomLog. Écrit dans le journal utilisateur. - LogError. Écrit un message d'erreur dans le journal du moteur. - LogInfo. Écrit un message d'information dans le journal du moteur. - LogWarning. Écrit un message d'avertissement dans le journal du moteur. - NotifyFailure. Déclenche une notification. Utilisez on_iteration_fail pour écrire une entrée quand une seule itération échoue. Utilisez la propriété on_fail pour écrire une entrée quand RepeatingGroupSerializer échoue. Pour plus d'informations, consultez la section "Traitement des échecs" à la page 404 .
facultatif	Détermine si la défaillance d'un composant entraîne l'échec du composant parent. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none"> - Sélectionné. La défaillance du composant n'entraîne pas l'échec du composant parent. - Effacé. La défaillance du composant entraîne l'échec du composant parent. Par défaut, la valeur est vide. Pour plus d'informations sur l'échec des composants, consultez la section "Traitement des échecs" à la page 404 .
remarque	Commentaire défini par l'utilisateur, qui décrit l'objectif ou l'action du composant.

Propriété	Description
séparateur	Définit une ancre de sérialisation qui définit la chaîne séparatrice. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none"> - AlternativeSerializers - ContentSerializer - EmbeddedSerializer - GroupSerializer - RepeatingGroupSerializer - StringSerializer - Sérialiseur défini par l'utilisateur Par défaut, la valeur est vide.
separator_position	Définit la position du séparateur par rapport aux sections. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none"> - après. Écrit un séparateur après chaque section, y compris pour la première section. Par exemple, 1 2 3 4 - autour. Écrit des séparateurs avant et après chaque section, y compris pour la première section. Par exemple, 1 2 3 4 - avant. Écrit un séparateur avant chaque section, y compris pour la première section. Par exemple, 1 2 3 4 - entre. Écrit un séparateur entre les sections consécutives, mais ni avant la première section, ni après la dernière. Par exemple, 1 2 3 4 La valeur par défaut est Avant.
skip_failed_iterations	Détermine si les itérations échouées sont ignorées. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none"> - Sélectionné. RepeatingGroup ignore une itération échouée et passe à l'itération suivante. Si une itération réussit, RepeatingGroup réussit. - Effacé. RepeatingGroup échoue si n'importe quelle itération échoue. La propriété skip_failed_iterations a un effet uniquement si séparateur est défini. La valeur par défaut est Sélectionné.
source	Définit une zone de stockage des données qui contient la source pour la sérialisation. Pour plus d'informations, consultez la section "Présentation des validateurs, des notificateurs et du traitement des échecs" à la page 403 .
cible	Définit une zone de stockage des données qui contient le résultat de la sérialisation. Pour plus d'informations, consultez la section "Présentation des validateurs, des notificateurs et du traitement des échecs" à la page 403 .

Exemple

L'entrée XML contient la structure suivante :

```
<Persons>
  <Person>
    <Name>John</Name>
    <Age>35</Age>
  </Person>
  <Person>
    <Name>Larissa</Name>
    <Age>42</Age>
  </Person>
  ...
</Persons>
```


Un **RepeatingGroupSerializer** utilisant le caractère de retour à la ligne comme séparateur, peut sortir ses données dans :

```
John      35
Larissa   42
```

Vous pouvez itérer en parallèle sur plusieurs zones de stockage des données à occurrences multiples. Par exemple, vous pouvez itérer sur une liste d'hommes et une liste de femmes et sortir une liste de couples mariés. Pour effectuer ceci, insérez un **ContentSerializer** dans le groupe de répétition pour chaque zone de stockage des données.

StringSerializer

L'ancre de sérialisation **StringSerializer** écrit une chaîne prédéfinie dans le document de sortie.

Le tableau suivant décrit les propriétés de l'ancre de sérialisation **StringSerializer** :

Propriété	Description
désactivé	Détermine si le script ignore le composant et tous les composants enfants. Utilisez cette propriété pour tester, déboguer et modifier un script. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none">- Sélectionné. Le script ignore le composant.- Effacé. Le script applique le composant. La valeur par défaut est vide.
nom	Libellé descriptif pour le composant. Ce libellé s'affiche dans le fichier journal et la vue Événements . Utilisez la propriété nom pour identifier le composant ayant provoqué l'événement.
on_fail	Action à effectuer si le composant échoue. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none">- Vide. N'effectuer aucune action.- CustomLog. Écrire dans le journal utilisateur.- LogError. Écrire un message d'erreur dans le journal du moteur.- LogInfo. Écrire un message d'information dans le journal du moteur.- LogWarning. Écrire un message d'avertissement dans le journal du moteur.- NotifyFailure. Envoyer une notification. Par défaut, la valeur est vide. Pour plus d'informations sur la gestion des défaillances de composants, consultez la section "Traitement des échecs" à la page 404 .
remarque	Commentaire défini par l'utilisateur, qui décrit l'objectif ou l'action du composant.
str	Définit la chaîne écrite par le sérialiseur dans la sortie.

CHAPITRE 20

Mappeurs

Ce chapitre comprend les rubriques suivantes :

- [Création d'un mappeur, 354](#)
- [Composants imbriqués dans un mappeur, 354](#)
- [Exemple de mappeur, 355](#)
- [Propriétés standard d'un mapper, 356](#)
- [Documentation de référence du composant Mappeur, 357](#)
- [Documentation de référence du composant Ancre du mappeur, 359](#)

Création d'un mappeur

1. Ajoutez les schémas d'entrée et de sortie à l'objet de schéma, puis référencez les schémas dans la transformation Processeur de données.
2. Au niveau global du script, ajoutez un composant **Mappeur**.
3. Assignez respectivement les propriétés **source** et **cible** du **mappeur** aux éléments d'entrée et de sortie du **mappeur**.
4. Assignez la propriété **exemple_source** à un document d'entrée d'échantillon XML.
Lorsque vous ajoutez des composants au mappeur, l'outil Developer applique des codes couleur aux emplacements correspondants dans l'exemple de source. Les couleurs peuvent vous aider à confirmer que les composants sont définis correctement.
5. Modifiez les autres propriétés du **mappeur** comme requis.
6. Dans le **mappeur**, imbriquez une séquence d'actions **Mappage**, d'ancres du mappeur et de tout autre composant requis.
7. Testez le mappeur et modifiez le script.

Composants imbriqués dans un mappeur

Dans un **Mappeur**, vous pouvez imbriquer les composants suivants :

- N'importe quel nombre d'actions **Mappage**. Les actions récupèrent une zone de stockage des données depuis la sortie et écrivent le contenu dans la sortie.

- Éventuellement, n'importe quelle quantité d'ancres de mappeur. Pour plus d'informations, consultez la section ["Documentation de référence du composant Ancre du mappeur" à la page 359](#).
- Éventuellement, n'importe quelle quantité d'actions supplémentaires.

Les actions `Mappage` et les ancres de mappeur peuvent être dans n'importe quelle séquence. Vous pouvez également insérer d'autres actions dans la séquence.

Notez qu'un mappeur utilise des actions `Mappage` plutôt que des ancres de mappeur pour écrire dans la sortie XML. Cela peut sembler un peu différent des analyseurs et des sérialiseurs, où la sortie est créée respectivement par des ancres et des ancres de sérialisation. Il ne s'agit en fait que d'un problème de terminologie. L'action `Mappage` aurait pu être définie en tant qu'ancre du mappeur. Elle est définie comme une action, car elle est utile dans d'autres circonstances, sans rapport avec les mappeurs.

Exemple de mappeur

Pour illustrer la configuration du mappeur, nous présentons un exemple simple.

Source XML

L'entrée du mappeur est un document XML contenant une liste de noms de personnes ainsi que des numéros d'identification qui leur correspondent.

```
<Persons>
  <Person ID="10">Bob</Person>
  <Person ID="17">Larissa</Person>
  <Person ID="13">Marie</Person>
</Persons>
```

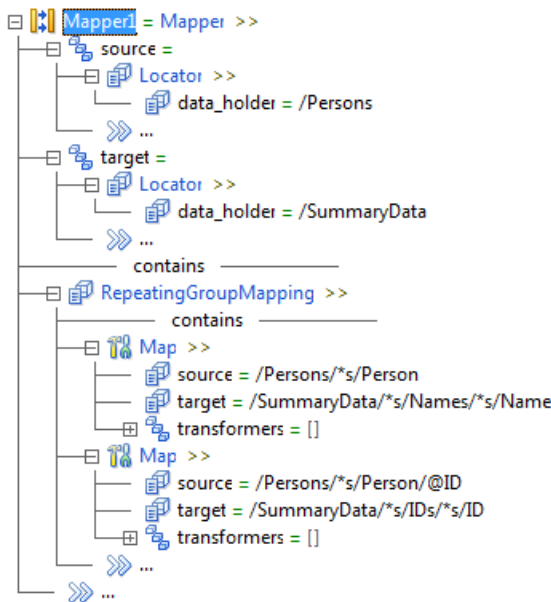
Sortie XML

La sortie souhaitée du mappeur est une liste XML de noms et de numéros d'identification, sans association entre eux.

```
<SummaryData>
  <Names>
    <Name>Bob</Name>
    <Name>Larissa</Name>
    <Name>Marie</Name>
  </Names>
  <IDs>
    <ID>10</ID>
    <ID>17</ID>
    <ID>13</ID>
  </IDs>
</SummaryData>
```

Configuration du mappeur

La configuration de mappeur suivante exécute la transformation souhaitée :



RepeatingGroupMapping se répète sur les éléments `Personne` de l'entrée. Il utilise les actions `Mappage` pour écrire les données dans les éléments `Nom` et `ID` de la sortie.

Propriétés standard d'un mapper

Le tableau suivant décrit les propriétés standard du composant **Mappeur** et de nombreuses ancres du mappeur :

Propriété	Description
disabled	Détermine si le script ignore le composant et tous les composants enfants. Utilisez cette propriété pour tester, déboguer et modifier un script. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none">- Sélectionné. Le script ignore le composant.- Effacé. Le script applique le composant. La valeur par défaut est vide.
nom	Libellé descriptif pour le composant. Ce libellé s'affiche dans le fichier journal et la vue Événements . Utilisez la propriété nom pour identifier le composant ayant provoqué l'événement.
notifications	Liste de composants Notificationhandler qui gèrent les notifications de composants imbriqués. Pour plus d'informations, voir "Notifications" à la page 423

Propriété	Description
on_fail	<p>Action à effectuer si le composant échoue. Vous pouvez choisir l'une des options suivantes :</p> <ul style="list-style-type: none"> - Vide. N'effectuer aucune action. - CustomLog. Écrire dans le journal utilisateur. - LogError. Écrire un message d'erreur dans le journal du moteur. - LogInfo. Écrire un message d'information dans le journal du moteur. - LogWarning. Écrire un message d'avertissement dans le journal du moteur. - NotifyFailure. Envoyer une notification. <p>Par défaut, la valeur est vide. Pour plus d'informations sur la gestion des défaillances de composants, consultez la section "Traitement des échecs" à la page 404.</p>
optional	<p>Détermine si la défaillance d'un composant entraîne l'échec du composant parent. Vous pouvez choisir l'une des options suivantes :</p> <ul style="list-style-type: none"> - Sélectionné. La défaillance du composant n'entraîne pas l'échec du composant parent. - Effacé. La défaillance du composant entraîne l'échec du composant parent. <p>Par défaut, la valeur est vide. Pour plus d'informations sur l'échec des composants, consultez la section "Traitement des échecs" à la page 404.</p>
remark	Commentaire défini par l'utilisateur, qui décrit l'objectif ou l'action du composant.

Documentation de référence du composant Mappeur

Un mappeur lit un document source XML et le convertit en un autre document XML.

Mappeur

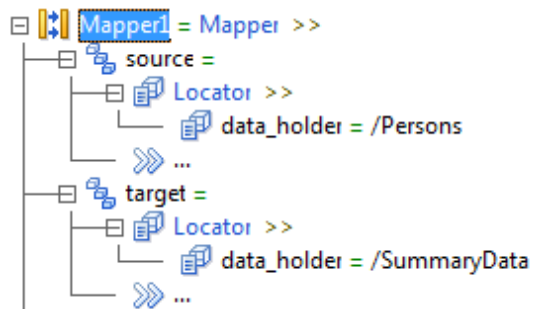
Un **Mappeur** effectue des transformations xml-vers-xml. Il convertit un document XML source en document de sortie ayant une structure XML différente.

Le tableau suivant décrit les propriétés du composant **Mappeur** :

Propriété	Description
example_source	<p>Définit un exemple de document XML source. Lorsque vous exécutez le mappeur dans l'outil Developer, il fonctionne sur l'exemple de document. Vous pouvez choisir l'une des options suivantes :</p> <ul style="list-style-type: none"> - Vide. Vous êtes invité à fournir un document source lorsque vous exécutez le script. - InputPort. Définit un port d'entrée. - LocalFile. Définit un fichier sur le système de fichiers local. - Texte. Définit une chaîne. - URL. Définit une URL. <p>La valeur par défaut est Vide.</p>
nom	Libellé descriptif pour le composant. Ce libellé s'affiche dans le fichier journal et la vue Événements . Utilisez la propriété nom pour identifier le composant ayant provoqué l'événement.
notifications	Liste de composants Notificationhandler qui gèrent les notifications de composants imbriqués. Pour plus d'informations, voir "Notifications" à la page 423

Propriété	Description
on_fail	<p>Action à effectuer si le composant échoue. Vous pouvez choisir l'une des options suivantes :</p> <ul style="list-style-type: none"> - Vide. N'effectuer aucune action. - CustomLog. Écrire dans le journal utilisateur. - LogError. Écrire un message d'erreur dans le journal du moteur. - LogInfo. Écrire un message d'information dans le journal du moteur. - LogWarning. Écrire un message d'avertissement dans le journal du moteur. - NotifyFailure. Envoyer une notification. <p>Par défaut, la valeur est vide. Pour plus d'informations sur la gestion des défaillances de composants, consultez la section "Traitement des échecs" à la page 404.</p>
remarque	Commentaire défini par l'utilisateur, qui décrit l'objectif ou l'action du composant.
root_tag	<p>Définit le nom d'un élément racine XML qui n'est pas défini dans le schéma.</p> <p>Par exemple, si l'élément de niveau supérieur du schéma est <code>Personne</code>, mais que l'entrée XML imbriquée <code>Personne</code> dans un élément appelé <code>InputWrapper</code>, définissez root_tag sur <code>InputWrapper</code>.</p>
source	<p>Définit un composant de localisateur qui définit une zone de stockage des données XML. La zone de stockage des données contient la racine de la source XML pour le mappage. Pour plus d'informations, consultez la section "Présentation des Localisateurs, des Clés et de l'Indexation" à la page 364.</p>
cible	<p>Définit un composant de localisateur qui définit une zone de stockage des données XML. La zone de stockage des données contient la racine de la sortie XML pour le mappage. Pour plus d'informations, consultez la section "Présentation des Localisateurs, des Clés et de l'Indexation" à la page 364.</p>
validate_source_document	<p>Détermine le niveau de validation de source XML que le sérialiseur effectue. Vous pouvez choisir l'une des options suivantes :</p> <ul style="list-style-type: none"> - Partielle. Autorise certaines déviations depuis le schéma. - Stricte. Applique le schéma strictement. <p>La valeur par défaut est Partiel.</p>

Vous devez utiliser les propriétés **source** et **cible** pour identifier les éléments racine des documents XML. Par exemple, si l'élément document de la source est `Persons` et que l'élément document de la sortie est `SummaryData`, définissez **source** et **cible** comme suit :



Documentation de référence du composant Ancre du mappeur

Les ancrs de mappeur, dans un mappeur, identifient et manipulent des données dans un document XML.

AlternativeMappings

L'ancre du mappeur **AlternativeMappings** définit un ensemble d'ancres du mappeur alternatives. Définissez un critère de sélection d'une alternative. Seule l'alternative acceptée affecte la sortie.

Le tableau suivant décrit les propriétés de l'ancre du mappeur **AlternativeMappings** :

Propriété	Description
désactivé	Détermine si le script ignore le composant et tous les composants enfants. Utilisez cette propriété pour tester, déboguer et modifier un script. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none">- Sélectionné. Le script ignore le composant.- Effacé. Le script applique le composant. La valeur par défaut est vide.
nom	Libellé descriptif pour le composant. Ce libellé s'affiche dans le fichier journal et la vue Événements . Utilisez la propriété nom pour identifier le composant ayant provoqué l'événement.
on_fail	Action à effectuer si le composant échoue. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none">- Vide. N'effectuer aucune action.- CustomLog. Écrire dans le journal utilisateur.- LogError. Écrire un message d'erreur dans le journal du moteur.- LogInfo. Écrire un message d'information dans le journal du moteur.- LogWarning. Écrire un message d'avertissement dans le journal du moteur.- NotifyFailure. Envoyer une notification. Par défaut, la valeur est vide. Pour plus d'informations sur la gestion des défaillances de composants, consultez la section "Traitement des échecs" à la page 404 .
facultatif	Détermine si la défaillance d'un composant entraîne l'échec du composant parent. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none">- Sélectionné. La défaillance du composant n'entraîne pas l'échec du composant parent.- Effacé. La défaillance du composant entraîne l'échec du composant parent. Par défaut, la valeur est vide. Pour plus d'informations sur l'échec des composants, consultez la section "Traitement des échecs" à la page 404 .
remarque	Commentaire défini par l'utilisateur, qui décrit l'objectif ou l'action du composant.
sélecteur	Détermine le critère de sélection d'un mappeur alternatif. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none">- ScriptOrder. Le script teste les ancrs du mappeur imbriquées dans l'ordre dans lequel elles sont définies dans le script. Il accepte la première qui réussit. Si toutes les ancrs du mappeur imbriquées échouent, le composant AlternativeMappings échoue aussi.- NameSwitch. Le script recherche l'ancre du mappeur imbriquée dont la propriété nom est spécifiée dans une zone de stockage des données. Il ignore les autres ancrs du mappeur imbriquées. Si l'ancre du mappeur nommée échoue, le composant AlternativeMappings échoue aussi. La valeur par défaut est ScriptOrder.

Exemple

L'entrée XML peut contenir un élément `Product` ou un élément `Service`, mais pas les deux. Vous souhaitez traiter tout élément se trouvant dans l'entrée.

Définissez une ancre de mappeur **AlternativeMappings** et définissez sa propriété **sélecteur** sur `ScriptOrder`.

Dans **AlternativeMappings**, imbriquez deux actions `Mappage`. Configurez l'une d'elles pour traiter l'élément `Product` et l'autre pour traiter `Service`.

EmbeddedMapper

L'ancre de mappeur **EmbeddedMapper** active un **mappeur** secondaire, qui stocke ses résultats dans le même document de sortie.

Le tableau suivant décrit les propriétés de l'ancre de mappeur **EmbeddedMapper** :

Propriété	Description
désactivé	Détermine si le script ignore le composant et tous les composants enfants. Utilisez cette propriété pour tester, déboguer et modifier un script. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none">- Sélectionné. Le script ignore le composant.- Effacé. Le script applique le composant. La valeur par défaut est vide.
Mappeur	Définit le nom du mappeur secondaire.
nom	Libellé descriptif pour le composant. Ce libellé s'affiche dans le fichier journal et la vue Événements . Utilisez la propriété nom pour identifier le composant ayant provoqué l'événement.
on_fail	Action à effectuer si le composant échoue. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none">- Vide. N'effectuer aucune action.- CustomLog. Écrire dans le journal utilisateur.- LogError. Écrire un message d'erreur dans le journal du moteur.- LogInfo. Écrire un message d'information dans le journal du moteur.- LogWarning. Écrire un message d'avertissement dans le journal du moteur.- NotifyFailure. Envoyer une notification. Par défaut, la valeur est vide. Pour plus d'informations sur la gestion des défaillances de composants, consultez la section "Traitement des échecs" à la page 404 .
facultatif	Détermine si la défaillance d'un composant entraîne l'échec du composant parent. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none">- Sélectionné. La défaillance du composant n'entraîne pas l'échec du composant parent.- Effacé. La défaillance du composant entraîne l'échec du composant parent. Par défaut, la valeur est vide. Pour plus d'informations sur l'échec des composants, consultez la section "Traitement des échecs" à la page 404 .
remarque	Commentaire défini par l'utilisateur, qui décrit l'objectif ou l'action du composant.
schema_connection s	Connecte les zones de stockage des données référencées dans le mappeur secondaire aux zones de stockage des données référencées dans le mappeur parent. La propriété contient une liste de sous-composants Connecter qui définissent les correspondances. Pour plus d'informations, consultez la section "Connecter" à la page 253 . Si toutes les zones de stockage des données des mappeurs principal et secondaire sont identiques, vous pouvez omettre cette propriété. S'il existe des différences entre les zones de stockage des données, vous devez connecter les zones de stockage des données explicitement.

Exemple

L'entrée XML est un arbre généalogique. L'entrée contient des éléments `Personne` récursivement imbriqués comme suit :

```
<Person>          <!-- Parent -->
...
  <Person>         <!-- Child -->
    ...
      <Person>     <!-- Grandchild -->
        ...
      </Person>
    </Person>
  </Person>
```

Un **Mappeur** peut se servir d'un composant **EmbeddedMapper** pour s'appeler lui-même récursivement, jusqu'à ce que tous les niveaux d'imbrication soient épuisés.

Dans cet exemple, `Personne` est connecté à `Personne/Personne`. Cela donne l'ordre à l'instance secondaire du mappeur de traiter un niveau imbriqué de l'entrée. Lorsque les deux éléments `Personne` ont le même type de données, il suffit de connecter uniquement l'élément parent (`Personne`) et non les éléments imbriqués (`Personne/*s/Nom`, `Personne/*s/DateNaiss`, etc.)

GroupMapping

L'ancre du mappeur **GroupMapping** lie ensemble ses ancres de mappeur et ses actions imbriquées. Vous pouvez définir les propriétés de **GroupMapping** qui affectent les membres du groupe.

Le tableau suivant décrit les propriétés de l'ancre de mappeur **GroupMapping** :

Propriété	Description
absent	Détermine le comportement de GroupMapping lorsqu'une de ces ancres du mappeur ou actions imbriquées obligatoires échoue. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none">- Sélectionné. Réussite de GroupMapping.- Effacé. Échec de GroupMapping. Utilisez cette fonctionnalité pour tester l'absence d'ancres de mappeur imbriquées. La valeur par défaut est Effacé.
désactivé	Détermine si le script ignore le composant et tous les composants enfants. Utilisez cette propriété pour tester, déboguer et modifier un script. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none">- Sélectionné. Le script ignore le composant.- Effacé. Le script applique le composant. La valeur par défaut est vide.
nom	Libellé descriptif pour le composant. Ce libellé s'affiche dans le fichier journal et la vue Événements . Utilisez la propriété nom pour identifier le composant ayant provoqué l'événement.
notifications	Liste de composants Notificationhandler qui gèrent les notifications de composants imbriqués. Pour plus d'informations, voir "Notifications" à la page 423
on_fail	Action à effectuer si le composant échoue. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none">- Vide. N'effectuer aucune action.- CustomLog. Écrire dans le journal utilisateur.- LogError. Écrire un message d'erreur dans le journal du moteur.- LogInfo. Écrire un message d'information dans le journal du moteur.- LogWarning. Écrire un message d'avertissement dans le journal du moteur.- NotifyFailure. Envoyer une notification. Par défaut, la valeur est vide. Pour plus d'informations sur la gestion des défaillances de composants, consultez la section "Traitement des échecs" à la page 404 .

Propriété	Description
facultatif	Détermine si la défaillance d'un composant entraîne l'échec du composant parent. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none"> - Sélectionné. La défaillance du composant n'entraîne pas l'échec du composant parent. - Effacé. La défaillance du composant entraîne l'échec du composant parent. Par défaut, la valeur est vide. Pour plus d'informations sur l'échec des composants, consultez la section "Traitement des échecs" à la page 404 .
remarque	Commentaire défini par l'utilisateur, qui décrit l'objectif ou l'action du composant.
source	Définit un composant de localisateur qui définit une zone de stockage des données XML. La zone de stockage des données contient la racine de la source XML pour le mappage. Pour plus d'informations, consultez la section "Présentation des Localisateurs, des Clés et de l'Indexation" à la page 364 .
cible	Définit un composant de localisateur qui définit une zone de stockage des données XML. La zone de stockage des données contient la racine de la sortie XML pour le mappage. Pour plus d'informations, consultez la section "Présentation des Localisateurs, des Clés et de l'Indexation" à la page 364 .

RepeatingGroupMapping

L'ancre de mappeur **RepeatingGroupMapping** traite une structure répétitive dans l'entrée ou la sortie.

Utilisez **RepeatingGroupMapping** lorsque le l'entrée ou la sortie XML contient une zone de stockage des données à occurrences multiples. Il se répète sur des occurrences des zones de stockage des données. Pour plus d'informations, consultez la section ["Zones de stockage des données à occurrences multiples" à la page 202](#)

Dans **RepeatingGroupMapping**, imbriquez des ancres de mappeur et des actions qui traitent chaque occurrence de la zone de stockage des données.

Le tableau suivant décrit les propriétés de l'ancre de mappeur **RepeatingGroupMapping** :

Propriété	Description
compteur	Définit le nombre d'itérations à exécuter. Si cette propriété est vide, les itérations continuent jusqu'à ce que l'entrée soit terminée.
current_iteration	Définit une zone de stockage des données dans laquelle RepeatingGroupMapping sort le numéro de l'itération en cours.
désactivé	Détermine si le script ignore le composant et tous les composants enfants. Utilisez cette propriété pour tester, déboguer et modifier un script. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none"> - Sélectionné. Le script ignore le composant. - Effacé. Le script applique le composant. La valeur par défaut est vide.
nom	Libellé descriptif pour le composant. Ce libellé s'affiche dans le fichier journal et la vue Événements . Utilisez la propriété nom pour identifier le composant ayant provoqué l'événement.
notifications	Liste de composants Notificationhandler qui gèrent les notifications de composants imbriqués. Pour plus d'informations, voir "Notifications" à la page 423

Propriété	Description
on_fail	<p>Action à effectuer si le composant échoue. Vous pouvez choisir l'une des options suivantes :</p> <ul style="list-style-type: none"> - Vide. N'effectuer aucune action. - CustomLog. Écrire dans le journal utilisateur. - LogError. Écrire un message d'erreur dans le journal du moteur. - LogInfo. Écrire un message d'information dans le journal du moteur. - LogWarning. Écrire un message d'avertissement dans le journal du moteur. - NotifyFailure. Envoyer une notification. <p>Par défaut, la valeur est vide. Pour plus d'informations sur la gestion des défaillances de composants, consultez la section "Traitement des échecs" à la page 404.</p>
on_iteration_fail	<p>Détermine l'action lorsqu'une itération échoue. Vous pouvez choisir l'une des options suivantes :</p> <ul style="list-style-type: none"> - Effacé. Aucune action. - CustomLog. Écrit dans le journal utilisateur. - LogError. Écrit un message d'erreur dans le journal du moteur. - LogInfo. Écrit un message d'information dans le journal du moteur. - LogWarning. Écrit un message d'avertissement dans le journal du moteur. - NotifyFailure. Déclenche une notification. <p>Utilisez on_iteration_fail pour écrire une entrée quand une seule itération échoue. Utilisez la propriété on_fail pour écrire une entrée quand RepeatingGroupMapping échoue complètement. Pour plus d'informations, consultez la section "Traitement des échecs" à la page 404.</p>
facultatif	<p>Détermine si la défaillance d'un composant entraîne l'échec du composant parent. Vous pouvez choisir l'une des options suivantes :</p> <ul style="list-style-type: none"> - Sélectionné. La défaillance du composant n'entraîne pas l'échec du composant parent. - Effacé. La défaillance du composant entraîne l'échec du composant parent. <p>Par défaut, la valeur est vide. Pour plus d'informations sur l'échec des composants, consultez la section "Traitement des échecs" à la page 404.</p>
remarque	Commentaire défini par l'utilisateur, qui décrit l'objectif ou l'action du composant.
skip_failed_iterations	<p>Détermine si les itérations échouées sont ignorées. Vous pouvez choisir l'une des options suivantes :</p> <ul style="list-style-type: none"> - Sélectionné. RepeatingGroup ignore une itération échouée et passe à l'itération suivante. Si une itération réussit, RepeatingGroup réussit. - Effacé. RepeatingGroup échoue si n'importe quelle itération échoue. <p>La propriété skip_failed_iterations a un effet uniquement si séparateur est défini. La valeur par défaut est Sélectionné.</p>
source	Définit une zone de stockage des données qui contient la source pour le mappage. Pour plus d'informations, consultez la section "Présentation des Localisateurs, des Clés et de l'Indexation" à la page 364 .
cible	Définit une zone de stockage des données qui contient le résultat du mappage. Pour plus d'informations, consultez la section "Présentation des Localisateurs, des Clés et de l'Indexation" à la page 364 .

Exemple

Pour plus d'informations et notamment pour obtenir un exemple de **RepeatingGroupMapping**, consultez la section ["Exemple de mappeur" à la page 355](#).

CHAPITRE 21

Localisateurs, clés et indexation

Ce chapitre comprend les rubriques suivantes :

- [Présentation des Localisateurs, des Clés et de l'Indexation, 364](#)
- [Exemple de localisateurs, 365](#)
- [Exemple d'indexation par clé, 367](#)
- [Propriétés Source et Target, 370](#)
- [Propriétés standard des Localisateurs et Clés, 376](#)
- [Documentation de référence des composants Localisateur et Clé, 376](#)

Présentation des Localisateurs, des Clés et de l'Indexation

Lors de la conception d'une transformation, une question se pose souvent sur la manière de localiser les zones de stockage des données que vous voulez traiter. Si les mêmes zones de stockage des données peuvent apparaître plusieurs fois dans une structure XML, il peut y avoir des ambiguïtés lors de l'identification des occurrences. Ce chapitre explique comment utiliser les composants `Localisateur` et `Clé` pour résoudre les ambiguïtés.

Les composants décrits dans ce chapitre vous permettent d'identifier les occurrences des zones de stockage des données à occurrences multiples de trois façons :

- Séquentiellement. Chaque itération d'un composant traite l'occurrence suivante du zone de stockage des données.
- Par numéro d'occurrence. Par exemple, un composant peut sélectionner la troisième occurrence d'une zone de stockage des données.
- Par une clé, comme un attribut ou un élément imbriqué. La clé identifie de façon unique l'occurrence du zone de stockage des données.

L'approche par défaut est séquentielle. Cette approche est assez complexe, mais vous pouvez la contrôler en utilisant le composant `Localisateur`.

Les approches par numéro d'occurrence et par clé sont regroupées sous le terme d'indexation. Le terme est analogue à celui de l'index d'un livre, où vous utilisez un numéro de page ou une clé de sujet pour identifier l'emplacement d'une information. Vous pouvez implémenter l'indexation en utilisant des composants appelés `LocatorByOccurrence`, `LocatorByKey` et `Clé`.

Vous pouvez utiliser les composants localisateur et clé dans les analyseurs, les sérialiseurs ou les mappeurs. Vous pouvez utiliser les composants pour identifier les occurrences de zones de stockage des données dans l'entrée, la sortie ou les deux.

Les composants du localisateur sont imbriqués dans les propriétés `source` et `cible` de différents composants de transformation. La signification et l'utilisation des propriétés `source` et `cible` sont expliquées ici.

Exemple de localisateurs

Pour comprendre les questions impliquées dans l'identification des zones de stockage des données, considérez l'exemple suivant. L'exemple illustre l'utilisation de :

- La propriété `cible`
- Le composant `Localisateur`

Nous expliquerons ici les grandes lignes de l'exemple. Dans les sections suivantes du chapitre, nous reviendrons et expliqueront en détail la manière dont fonctionnent `cible` et le `Localisateur`.

Entrée et Sortie

Imaginez que le schéma de sortie d'un analyseur prenne en charge la structure suivante :

```
<Report>
  <Company>
    <Employee>John</Employee>
    <Employee>Leslie</Employee>
    <Employee>Pedro</Employee>
  </Company>
  <Company>
    <Employee>Marie</Employee>
    <Employee>Larry</Employee>
    <Employee>Frances</Employee>
  </Company>
</Report>
```

Le document source traité par l'analyseur est une liste contenant un seul employé par entreprise :

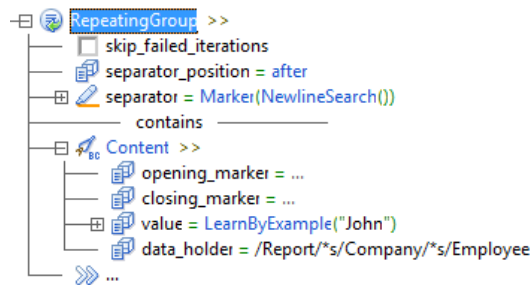
```
John
Marie
```

La sortie de l'analyseur doit être :

```
<Report>
  <Company>
    <Employee>John</Employee>
  </Company>
  <Company>
    <Employee>Marie</Employee>
  </Company>
</Report>
```

Solution incorrecte

Admettons que vous utilisiez le **RepeatingGroup** suivant pour analyser le document source :



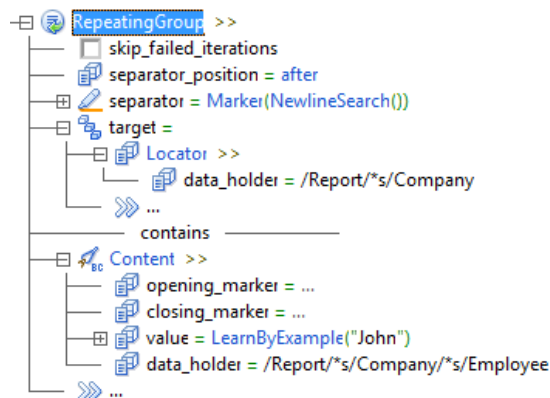
La sortie est incorrecte :

```
<Report>
  <Company>
    <Employee>John</Employee>
    <Employee>Marie</Employee>
  </Company>
</Report>
```

Le problème vient du fait que `Entreprise` et `Employé` sont tous deux des éléments à occurrences multiples. `RepeatingGroup` crée correctement plusieurs éléments `Employé`, mais ne sait pas que chaque élément `Employé` doit être imbriqué dans un élément `Entreprise` séparé.

Solution correcte

Pour résoudre l'ambiguïté, vous pouvez affecter la propriété `cible` au `RepeatingGroup`.



La cible identifie la zone de stockage des données devant être créée par `RepeatingGroup`. La cible contient un composant Localisateur pointant vers l'élément `Société`. Cela signifie que chaque itération de `RepeatingGroup` devrait créer une nouvelle occurrence de l'élément `Société`.

Si vous configurez `RepeatingGroup` de cette façon, la sortie est correcte.

Exemple d'indexation par clé

Pour aller plus avant dans la présentation de l'identification des conteneurs de données, nous donnons un exemple d'indexation par clé.

L'exemple est un mappeur utilisant l'indexation pour identifier les occurrences des zones de stockage des données à la fois dans son entrée et sa sortie. Du côté de l'entrée, l'indexation compare des données de différentes parties d'une structure XML. Du côté de la sortie, l'indexation trouve l'emplacement correct d'un élément dans une structure XML.

L'exemple illustre l'utilisation de :

- Les propriétés `source` et `cible`
- Les composants `Locator`, `Clé` et `LocatorByKey`

Dans les sections suivantes de ce chapitre, nous expliquerons les opérations détaillées de ces propriétés et de ces composants.

Entrée

L'entrée XML est un rapport énumérant les noms de parents et de leurs enfants.

- Pour chaque parent, le XML liste un prénom, un nom et un identifiant.
- Pour chaque enfant, le XML liste un prénom, un hobby et l'identifiant du parent.

```
<Report>
  <Parents>
    <Parent id="1" firstName="John" lastName="Smith"/>
    <Parent id="2" firstName="Jane" lastName="Doe"/>
  </Parents>
  <Children>
    <Child name="Eric" hobby="Swimming" parentID="1"/>
    <Child name="Elizabeth" hobby="Biking" parentID="2"/>
    <Child name="Mary" hobby="Painting" parentID="1"/>
    <Child name="Edward" hobby="Swimming" parentID="2"/>
  </Children>
</Report>
```

Sortie

La sortie souhaitée est une liste de passe-temps et les enfants qui s'y adonnent.

```
<Hobbies>
  <Hobby name="Swimming">
    <Person firstName="Eric" lastName="Smith"/>
    <Person firstName="Edward" lastName="Doe"/>
  </Hobby>
  <Hobby name="Biking">
    <Person firstName="Elizabeth" lastName="Doe"/>
  </Hobby>
  <Hobby name="Painting">
    <Person firstName="Mary" lastName="Smith"/>
  </Hobby>
</Hobbies>
```

Aperçu de l'approche de la transformation

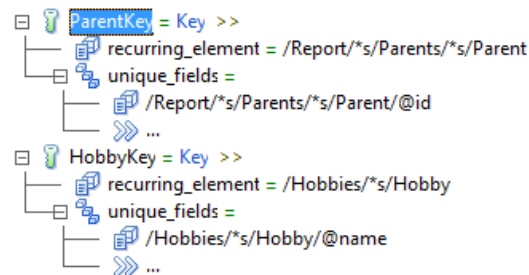
La transformation utilise l'approche suivante :

1. Dans le XML d'entrée, la transformation identifie les éléments `Enfant` et `Parent` correspondants, comme suit :
attribut `id` de `Parent` = attribut `parentID` d'`Enfant`
2. La transformation crée les éléments `Hobby` et `Personne`. Elle identifie l'élément `Hobby` où elle doit imbriquer chaque élément `Personne`, comme suit :
Attribut `nom` de `Hobby` = attribut `hobby` d'`Enfant`
3. La transformation écrit le prénom de l'enfant dans l'élément `Personne`.
4. La transformation écrit le nom de famille du parent dans l'élément `Personne`.

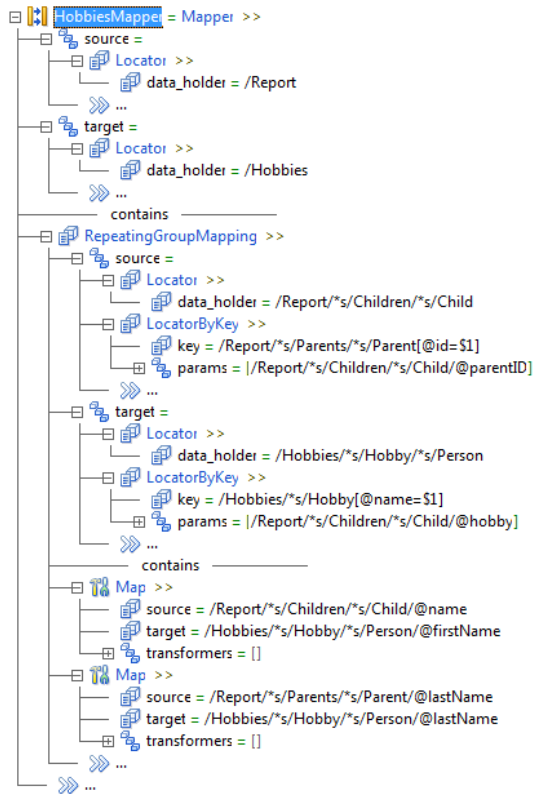
Configuration du mappeur

Les composants `Clé` définissent des identifiants pour les zones de stockage des données.

- La première `Clé` spécifie que l'attribut `id` est un identificateur unique d'un élément `Parent`.
- La seconde `Clé` spécifie que l'attribut `nom` est un identificateur unique d'un élément `Hobby`.



Le script définit alors un `Mappeur` ayant la configuration suivante :



Les composants du `Mappeur` exécutent les fonctions suivantes :

- La propriété `source` du `RepeatingGroupMapping` spécifie que chaque itération tient son entrée des zones de stockage des données suivants :
 - D'une occurrence de l'élément `Enfant`
 - De l'occurrence correspondante de l'élément `Parent`
- La propriété `cible` de `RepeatingGroupMapping` spécifie que chaque itération stocke sa sortie dans les zones de stockage des données suivants :
 - Dans une occurrence de l'élément `Personne`
 - Dans l'occurrence correspondante de l'élément `Hobby`
- La première action `Mappage` copie l'attribut `nom` d'`Enfant` dans l'attribut `Prénom` de `Personne`.
- La seconde action `Mappage` copie l'attribut `Nom` de `Parent` dans l'attribut `Nom` de `Personne`.

Utilisation de l'indexation

L'exemple utilise l'indexation par clés pour identifier les occurrences des zones de stockage des données `Parent` et `Hobby`.

- Dans la propriété `source` de `RepeatingGroupMapping`, l'indexation identifie l'occurrence de l'élément `Parent` qui correspond à un élément `Enfant`.
- Dans la propriété `target`, l'indexation identifie l'occurrence de l'élément `Hobby` à l'endroit où une personne doit être imbriquée.

Propriétés Source et Target

Les propriétés `source` et `target` existent notamment dans les composants suivants :

- Dans les analyseurs :

```
Parser
Group
RepeatingGroup
EnclosedGroup
FindReplaceAnchor
```

- Dans les serializers :

```
Serializer
GroupSerializer
RepeatingGroupSerializer
```

- Dans les mappers :

```
Mapper
GroupMapping
RepeatingGroupMapping
```

Dans ces catégories, la signification et l'usage des propriétés est identique:

- La propriété `source` identifie les zones de stockage des données existantes qu'une transformation de données doit utiliser.
- La propriété `target` identifie les zones de stockage des données existantes ou non. Si elles existent, la transformation de données les utilise. Si elles n'existent pas, la transformation de données les crée.

Une fois que vous avez défini la propriété `source` et/ou `target`, les composants ultérieurs utilisent les zones de stockage des données identifiées. Par exemple, si vous définissez la propriété `target` d'un groupe, les ancres imbriquées dans le groupe utilisent les zones de stockage des données identifiées par celle-ci.

Remarque: Il existe des propriétés nommées `source` et `target` dans certains autres composants comme `Carte`. Ces propriétés ont une signification et une utilisation différentes de ci-dessus. Pour une explication, veuillez voir les composants où les propriétés sont utilisées.

Propriété source

La propriété **source** identifie les occurrences existantes des zones de stockage des données. La valeur de la propriété **source** est une liste contenant un ou plusieurs des composants suivants :

Source	Description
Localisateur	Identifie une zone de stockage des données simple ou multi-occurrence. Dans le dernier cas, chaque itération accède à l'occurrence suivante, en ordre séquentiel.
LocatorByKey	Identifie par une clé une occurrence d'une zone de stockage des données multi-occurrence.
LocatorByOccurrence	Identifie par un numéro une occurrence d'une zone de stockage des données multi-occurrence.

Comportement par défaut

Si vous n'affectez pas la propriété `source` d'un composant, le composant identifie les zones de stockage des données de la façon suivante :

- S'il n'y a qu'une occurrence de la zone de stockage des données, le composant utilise l'occurrence existante.

- S'il y a plusieurs occurrences de la zone de stockage des données, le comportement est le suivant :
 - Dans un contexte itératif, par exemple dans un `RepeatingGroupSerializer`, chaque itération accède à l'occurrence suivante de la zone de stockage des données, séquentiellement.
 - Dans un contexte non itératif, par exemple, un `GroupSerializer` qui n'est pas imbriqué dans un composant itératif, le composant accède à la première occurrence de la zone de stockage des données.

Ambiguïtés dans le comportement par défaut

Il peut y avoir quelques ambiguïtés dans le comportement par défaut. Les ambiguïtés peuvent par exemple survenir dans les circonstances suivantes.

- Dans les cas où un élément à occurrences multiples est imbriqué dans un autre élément à occurrences multiples. Pour plus d'informations, consultez la section ["Exemple 1 : zones de stockage des données à occurrences multiples imbriquées" à la page 371](#).
- Dans les cas où le schéma autorise des zones de stockage des données alternatives, définies avec `xs:choice`.
- Dans les cas où le schéma autorise l'absence d'une zone de stockage des données, définie avec `minOccurs = 0`.

Dans de tels cas, il est prudent d'affecter la propriété `source` explicitement.

La zone de stockage des données doit exister

La propriété `source` identifie une zone de stockage des données qui existe déjà dans la portée de la transformation. Si la zone de stockage des données n'existe pas, le composant contenant la propriété `source` échoue.

Supposez, par exemple, que la propriété `source` d'un `Groupe` contienne un `LocatorByOccurrence` non optionnel, pointant sur la troisième occurrence d'une zone de stockage des données. Si seulement deux occurrences existent, le `Groupe` échoue.

Utiliser la propriété `source` pour l'entrée et la sortie

Généralement, un composant utilise la propriété `source` pour identifier où il doit obtenir l'entrée. Par exemple, `GroupSerializer` peut utiliser la propriété pour identifier l'occurrence qu'il doit sérialiser.

Il est aussi possible d'utiliser la propriété pour identifier l'endroit où le composant doit enregistrer la sortie. Supposez, par exemple, qu'un analyseur a déjà créé 10 occurrences d'un élément XML. Après la création des occurrences, une ancre `Groupe` affecte un attribut à une occurrence de l'élément. Le `groupe` peut utiliser la propriété `source` pour identifier l'occurrence.

Exemple 1 : zones de stockage des données à occurrences multiples imbriquées

Supposez que le schéma de sortie d'un sérialiseur prenne en charge la structure suivante :

```
<Report>
  <Company>
    <Employee>John</Employee>
    <Employee>Leslie</Employee>
    <Employee>Pedro</Employee>
  </Company>
  <Company>
    <Employee>Marie</Employee>
    <Employee>Larry</Employee>
    <Employee>Frances</Employee>
```

```

    </Company>
</Report>

```

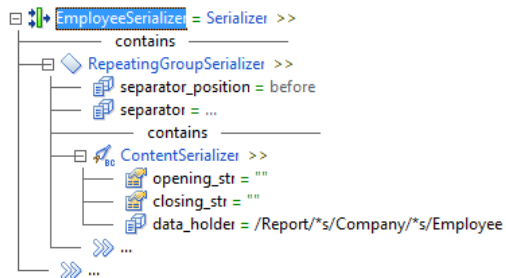
Vous voulez repasser sur tous les éléments `Employé` et produire la sortie suivante :

```

John
Leslie
Pedro
Marie
Larry
Frances

```

Vous pourriez créer un `RepeatingGroupSerializer` et le configurer pour sortir la zone de stockage des données `Employé`.



Ceci ne fonctionne pas correctement ! Par défaut, chaque itération sélectionne une nouvelle instance d'`Employé` dans la même `Entreprise`. Le résultat est la sortie suivante :

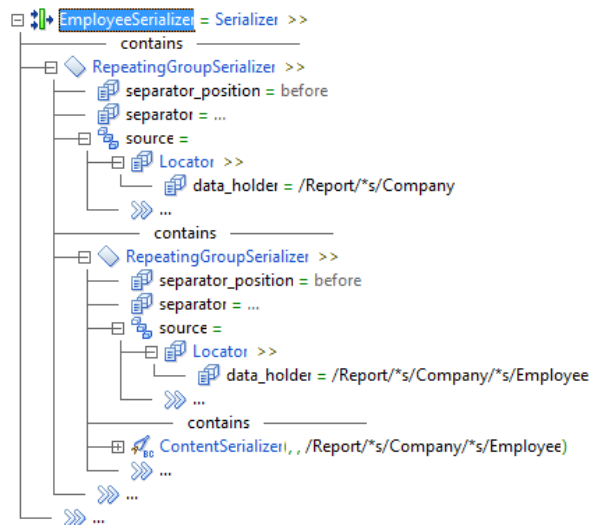
```

John
Leslie
Pedro

```

En d'autres mots, `RepeatingGroupSerializer` accède seulement à la première `Entreprise`.

Vous pouvez résoudre ce problème en imbriquant le `RepeatingGroupSerializer` dans un autre `RepeatingGroupSerializer`. Pour résoudre d'éventuelles ambiguïtés, vous pouvez configurer les propriétés `source` explicitement.

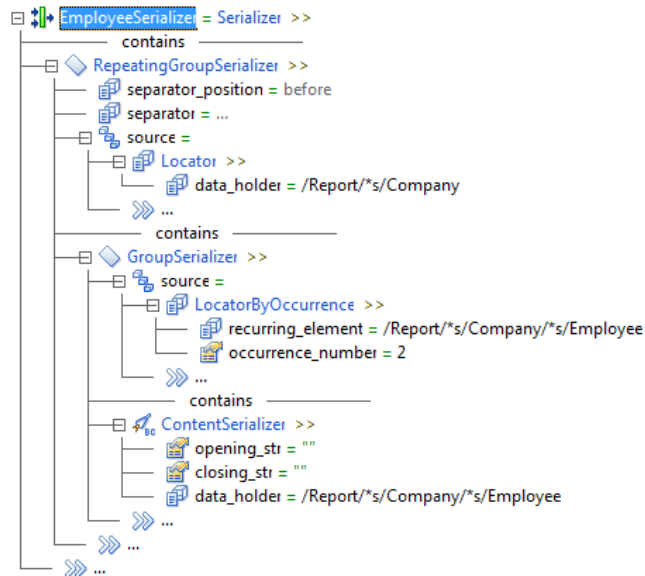


Chaque itération du `RepeatingGroupSerializer` extérieur traite une occurrence différente d'`Entreprise`. Chaque itération du `RepeatingGroupSerializer` imbriqué traite une occurrence différente d'`Employé`. Le résultat est la sortie attendue.

Supposez que vous vouliez passer seulement sur le deuxième élément `Employé` de chaque `Entreprise`. La sortie souhaitée est :

```
Leslie
Larry
```

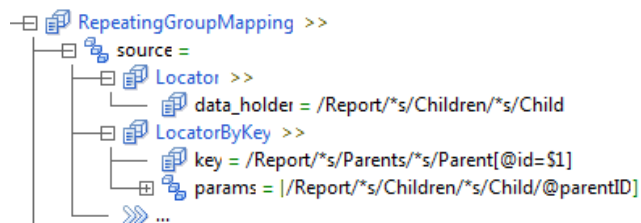
Vous pouvez réaliser cela en configurant un seul `RepeatingGroupSerializer` dont la source est `Entreprise`. Chaque itération accède alors à l'instance suivante d'`Entreprise`. Vous pouvez configurer dans l'itération un `GroupSerializer`, dont la propriété `source` utilise un `LocatorByOccurrence` pour sélectionner le second `Employé`. Cela génère la sortie attendue.



Exemple 2 : Indexation

Dans l'exemple de l'indexation par clé au début de ce chapitre, nous utilisons un `RepeatingGroupMapping` configuré comme suit. Dans cet exemple, la propriété `source` identifie deux zones de stockage des données :

- Elle utilise un composant `Localisateur` pour identifier une occurrence d'`Enfant`. Chaque itération traite l'occurrence d'`Enfant` suivante, séquentiellement.
- Elle utilise un composant `LocatorByKey` pour identifier une occurrence de `Parent`. Cela provoque le traitement par chaque itération de l'occurrence de `Parent` qui correspond à l'occurrence d'`Enfant`.



Pour plus d'informations, consultez la section ["Exemple d'indexation par clé" à la page 367](#).

Propriété target

La propriété **target** identifie une occurrence d'une zone de stockage des données qui peut exister ou ne pas encore exister. Si l'occurrence existe, le composant l'utilise. Si l'occurrence n'existe pas, le composant la crée.

La valeur de la propriété **target** est une liste contenant un ou plusieurs des composants suivants :

Target	Description
Localisateur	Identifie une zone de stockage des données simple ou multi-occurrence. Dans le dernier cas, chaque itération crée une nouvelle occurrence.
LocatorByKey	Identifie par une clé d'index une occurrence d'une zone de stockage des données multi-occurrence. Si l'occurrence n'existe pas, le composant la crée.
LocatorByOccurrence	Identifie par un numéro une occurrence d'une zone de stockage des données multi-occurrence. Si l'occurrence n'existe pas, elle est créée ainsi que les occurrences intervenant le cas échéant. Par exemple, si quatre occurrences existent et que LocatorByOccurrence spécifie la dixième occurrence, les occurrences 5 à 9 sont également créées, mais restent vides.

Comportement par défaut

Si vous n'affectez pas la propriété **cible** d'un composant, le composant identifie les zones de stockage des données de la façon suivante :

- Si le schéma n'autorise qu'une seule occurrence de la zone de stockage des données, le script accède à l'occurrence ou la crée.
- Si la zone de stockage des données peut avoir plusieurs occurrences, le comportement est le suivant :
 - Dans un contexte itératif, par exemple dans un **RepeatingGroup**, chaque itération crée une nouvelle occurrence de la zone de stockage des données.
 - Dans un contexte non itératif, par exemple un **Groupe** qui n'est pas imbriqué dans un composant itératif, le composant crée une nouvelle occurrence de la zone de stockage des données.

Ambiguïtés dans le comportement par défaut

Il peut y avoir quelques ambiguïtés dans le comportement par défaut. Les ambiguïtés peuvent par exemple survenir dans les circonstances suivantes.

- Dans les cas où un élément à occurrences multiples est imbriqué dans un autre élément à occurrences multiples. Pour plus d'informations, consultez la section ["Exemple 1 : zones de stockage des données à occurrences multiples imbriquées" à la page 375](#).
- Dans les cas où le schéma autorise des zones de stockage des données alternatives, définies avec `xs:choice`.
- Dans les cas où le schéma autorise l'absence d'une zone de stockage des données, définie avec `minOccurs = 0`.

Dans de tels cas, il est prudent d'affecter la propriété `cible` explicitement.

Les zones de stockage des données peuvent être créées

La propriété `cible` identifie une zone de stockage des données qui peut déjà exister ou non dans l'étendue de la transformation. Si la zone de stockage des données n'existe pas, elle est créée.

Imaginez, par exemple, que la propriété `cible` d'un `Groupe` contienne un `LocatorByKey` pointant sur une occurrence particulière d'une zone de stockage des données. Si l'occurrence existe déjà, le `Groupe` l'utilise. Si l'occurrence n'existe pas, le `Groupe` la crée.

Utiliser la propriété `target` pour l'entrée et la sortie

Généralement, un composant utilise la propriété `target` pour identifier où il doit stocker la sortie. Par exemple, un `groupe` peut utiliser la propriété pour identifier l'occurrence où il doit stocker les données.

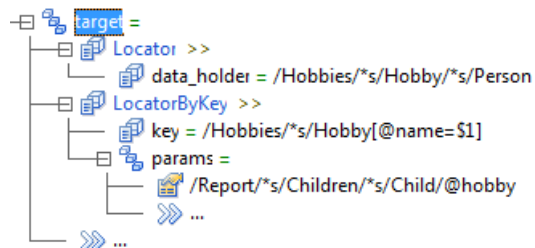
Il est aussi possible d'utiliser la propriété pour identifier l'endroit où le composant doit obtenir son entrée. Supposez par exemple que `GroupSerializer` contienne une action qui calcule des données et les stocke dans une variable. `GroupSerializer` active ensuite `ContentSerializer` qui enregistre la variable dans la sortie. Vous pouvez utiliser la propriété `target` pour créer l'occurrence de la variable que `GroupSerializer` utilise. La variable sert alors d'entrée à `ContentSerializer`.

Exemple 1 : zones de stockage des données à occurrences multiples imbriquées

L'exemple des localisateurs au début de ce chapitre illustre comment utiliser la propriété `cible` pour différencier les zones de stockage des données à occurrences multiples parent et fils. Pour plus d'informations, consultez la section ["Exemple de localisateurs" à la page 365](#).

Exemple 2 : Indexation

L'exemple d'indexation par clé au début de ce chapitre illustre comment utiliser la propriété `cible` avec l'indexation. La figure suivante illustre comment configurer la propriété `cible` du `RepeatingGroupMapping` :



La propriété `cible` identifie les zones de stockage des données suivantes :

- Un composant `Locator` identifie une occurrence de `Personne`. Chaque itération crée une nouvelle occurrence de `Personne`.
- Un composant `LocatorByKey` identifie l'occurrence de l'élément `Hobby`, où l'occurrence de `Personne` doit être imbriquée. Si l'élément `Hobby` existe déjà, la transformation de données l'utilise. Si l'élément `Hobby` n'existe pas, la transformation de données le crée.

Pour plus d'informations, voir ["Exemple d'indexation par clé" à la page 367](#)

Propriétés standard des Localisateurs et Clés

Le tableau suivant décrit les propriétés standard qui sont utilisées dans les composants du localisateur et des clés :

Propriété	Description
disabled	Détermine si le script ignore le composant et tous les composants enfants. Utilisez cette propriété pour tester, déboguer et modifier un script. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none">- Sélectionné. Le script ignore le composant.- Effacé. Le script applique le composant. La valeur par défaut est vide.
optional	Détermine si la défaillance d'un composant entraîne l'échec du composant parent. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none">- Sélectionné. La défaillance du composant n'entraîne pas l'échec du composant parent.- Effacé. La défaillance du composant entraîne l'échec du composant parent. Par défaut, la valeur est vide. Pour plus d'informations sur l'échec des composants, consultez la section "Traitement des échecs" à la page 404 .
remark	Commentaire défini par l'utilisateur, qui décrit l'objectif ou l'action du composant.

Documentation de référence des composants Localisateur et Clé

Les composants Localisateur et Clé identifient les éléments dans le script ou les zones de stockage des données.

Clé

Le composant **Clé** définit les attributs ou les éléments servant d'identifiant unique de l'élément de leur parent.

Vous pouvez définir une **Clé** uniquement au niveau global du script et vous pouvez référencer la **Clé** n'importe où dans le script. Le nom d'une **Clé** est sensible à la casse.

Le tableau suivant décrit les propriétés du composant **Clé** :

Propriété	Description
désactivé	Détermine si le script ignore le composant et tous les composants enfants. Utilisez cette propriété pour tester, déboguer et modifier un script. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none">- Sélectionné. Le script ignore le composant.- Effacé. Le script applique le composant. La valeur par défaut est vide.
recurring_element	Définit un élément à occurrences multiples dont les occurrences sont identifiées par la clé.

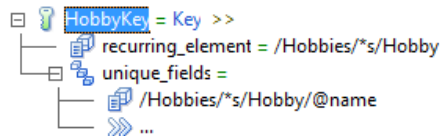
Propriété	Description
remarque	Commentaire défini par l'utilisateur, qui décrit l'objectif ou l'action du composant.
unique_fields	Définit la clé.

Exemple

Exemple d'indexation par clé définit une clé pour l'élément `Hobby` dans la structure suivante :

```
<Hobbies>
  <Hobby name="Swimming">
    <Person firstName="Eric" lastName="Smith"/>
    <Person firstName="Edward" lastName="Doe"/>
  </Hobby>
  <Hobby name="Biking">
    <Person firstName="Elizabeth" lastName="Doe"/>
  </Hobby>
  <Hobby name="Painting">
    <Person firstName="Mary" lastName="Smith"/>
  </Hobby>
</Hobbies>
```

La clé est l'attribut `nom` qui identifie de façon unique chaque `Hobby`.



Clés composites

Vous pouvez éventuellement définir une liste de zones de stockage des données comme clé composite. Imbriquez pour cela plusieurs zones de stockage des données dans la propriété `unique_fields`.

Considérons l'exemple suivant :

```
<Persons>
  <Person ID="17" SubID="A">Bob</Person>
  <Person ID="17" SubID="B">Jane</Person>
  <Person ID="35" SubID="A">Larry</Person>
</Persons>
```

Ni l'attribut `ID`, ni l'attribut `SubID` identifient un élément `Personne` de manière unique. Par contre, la combinaison des attributs `ID` et `SubID` est un identifiant unique. Vous pouvez définir `ID` et `SubID` comme clé composite.

Restrictions sur la clé

`unique_fields` doivent être imbriqués dans `recurring_element`. Ils peuvent représenter des attributs de l'élément, ils peuvent représenter des éléments imbriqués à n'importe quel niveau d'imbrication ou ils peuvent représenter des attributs d'éléments imbriqués.

Par exemple, cela signifie que `Personnes/Personne/SécuritéSociale/@Numero` peut être une clé valide pour `Personnes/Personne`, car `@Numero` est imbriqué dans `Personnes/Personne`. D'un autre côté, `Personnes/Enfant` n'est pas une clé valide pour `Personnes/Personne`, car elle n'est pas imbriquée correctement.

Les `unique_fields` doivent identifier l'ancêtre le plus proche pouvant avoir plusieurs occurrences. Par exemple, si `Parent` et `Enfant` sont tous deux des éléments à occurrences multiples, alors `Parent/Enfant/@nom` peut être une clé valide pour `Parent/Enfant`, mais pas pour `Parent`.

Les `unique_fields` doivent avoir des types de données simples. Ils ne peuvent pas être des structures.

Occurrences sœurs et non-sœurs

Une clé identifie de façon unique les occurrences sœurs d'un élément. Il est permis pour des occurrences non sœurs d'avoir la même clé.

Observez la structure XML suivante :

```
<Report>
  <Company>
    <Employee ID="1">John</Employee>
    <Employee ID="2">Leslie</Employee>
  </Company>
  <Company>
    <Employee ID="1">Marie</Employee>
    <Employee ID="2">Larry</Employee>
  </Company>
</Report>
```

L'attribut `ID` peut être une clé valide pour `Employé` car il identifie un employé dans une seule entreprise. La duplication des valeurs `ID` dans différents éléments `Entreprise` n'invalide pas la clé.

Clés d'éléments réutilisables

Vous pouvez définir une clé sur un élément réutilisable défini dans un schéma

Supposez par exemple que `Personnes/Personne` peut apparaître dans différents contextes dans le XML. Si vous définissez `ID` comme clé de `Personnes/Personne`, la clé est valide dans tous les contextes où `Personnes/Personne` est utilisé.

Unicité forcée d'une clé

Le script applique l'unicité d'une **Clé**. Cela a pour conséquences :

- Si deux occurrences sœurs ou plus d'un élément d'entrée ont la même valeur de clé, le script considère que chaque occurrence remplace les occurrences précédentes. Il utilise seulement la dernière occurrence rencontrée.
- S'il manque une valeur de clé à une occurrence d'un élément d'entrée, l'occurrence est ignorée.
- Si le script sort un élément avec clé et qu'un élément frère ayant la même valeur de clé existe déjà, l'occurrence existante est écrasée.

Dans ce cas, le script écrit un avertissement dans le journal des événements.

Localisateur

Le composant **Localisateur** identifie une zone de stockage des données à occurrence unique ou à occurrences multiples dans les propriétés **source** et **cible**. Pour les zones de stockage des données à occurrences multiples, chaque itération d'un composant qui utilise le **Localisateur** traite l'occurrence suivante du zone de stockage des données.

Le tableau suivant décrit les propriétés du composant **Locator** :

Propriété	Description
data_holder	Définit la zone de stockage des données que le composant Localisateur identifie.
désactivé	Détermine si le script ignore le composant et tous les composants enfants. Utilisez cette propriété pour tester, déboguer et modifier un script. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none">- Sélectionné. Le script ignore le composant.- Effacé. Le script applique le composant. La valeur par défaut est vide.
facultatif	Détermine si la défaillance d'un composant entraîne l'échec du composant parent. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none">- Sélectionné. La défaillance du composant n'entraîne pas l'échec du composant parent.- Effacé. La défaillance du composant entraîne l'échec du composant parent. Par défaut, la valeur est vide. Pour plus d'informations sur l'échec des composants, consultez la section "Traitement des échecs" à la page 404 .
remarque	Commentaire défini par l'utilisateur, qui décrit l'objectif ou l'action du composant.

LocatorByKey

Le composant **LocatorByKey** identifie une occurrence d'une zone de stockage des données à occurrences multiples dans les propriétés **source** et **cible**.

Avant d'utiliser **LocatorByKey**, vous devez définir une **Clé** au niveau global du script. La **Clé** spécifie les zones de stockage des données qui identifient l'occurrence de façon unique.

Le tableau suivant décrit les propriétés du composant **LocatorByKey** :

Propriété	Description
désactivé	Détermine si le script ignore le composant et tous les composants enfants. Utilisez cette propriété pour tester, déboguer et modifier un script. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none">- Sélectionné. Le script ignore le composant.- Effacé. Le script applique le composant. La valeur par défaut est vide.
clé	Définit la représentation du prédicat XPath de la clé. Par exemple, si vous avez défini <code>Hobbies/Hobby/@name</code> en tant que Clé , sélectionnez <code>Hobbies/Hobby[@name=\$1]</code> . Cette propriété est requise.
facultatif	Détermine si la défaillance d'un composant entraîne l'échec du composant parent. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none">- Sélectionné. La défaillance du composant n'entraîne pas l'échec du composant parent.- Effacé. La défaillance du composant entraîne l'échec du composant parent. Par défaut, la valeur est vide. Pour plus d'informations sur l'échec des composants, consultez la section "Traitement des échecs" à la page 404 .

Propriété	Description
params	Définit les valeurs des paramètres dans le prédicat XPath. Chaque valeur a un signe dollar (\$) et un nombre entier qui représente la position du paramètre dans la liste des paramètres. Cette propriété est requise.
remarque	Commentaire défini par l'utilisateur, qui décrit l'objectif ou l'action du composant.

Conflits entre les localisateurs

En cas de conflits, un **LocatorByKey** imbriqué supplante un localisateur parent.

Supposez par exemple que la propriété **cible** d'un **Groupe** contienne un **LocatorByKey** pointant sur la troisième occurrence d'un élément. Un **Groupe** imbriqué contient un **LocatorByKey** qui pointe sur la cinquième occurrence. Le **Groupe** imbriqué utilise la cinquième occurrence.

LocatorByOccurrence

Le composant **LocatorByOccurrence** est utilisé dans la propriété **source** pour identifier une occurrence d'une zone de stockage des données à occurrences multiples.

Par exemple :

- Un élément qui peut apparaître plusieurs fois dans un document XML
- Une variable qui peut apparaître plusieurs fois

Le composant identifie l'occurrence par son numéro. Par exemple, s'il y a dix occurrences d'une zone de stockage des données, vous pouvez utiliser **LocatorByOccurrence** pour traiter la troisième occurrence.

LocatorByOccurrence peut être utilisé pour répéter le processus sur les occurrences dans une structure récursive telle qu'une ancre **RepeatingGroup**.

Le composant **LocatorByKey** identifie une occurrence d'une zone de stockage des données à occurrences multiples dans les propriétés **source** et **cible**.

Avant d'utiliser **LocatorByKey**, vous devez définir une **Clé** au niveau global du script. La **Clé** spécifie les zones de stockage des données qui identifient l'occurrence de façon unique.

Le tableau suivant décrit les propriétés du composant **LocatorByOccurrence** :

Propriété	Description
désactivé	Détermine si le script ignore le composant et tous les composants enfants. Utilisez cette propriété pour tester, déboguer et modifier un script. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none"> - Sélectionné. Le script ignore le composant. - Effacé. Le script applique le composant. La valeur par défaut est vide.
occurrence_number	Définit le numéro de l'occurrence.

Propriété	Description
facultatif	Détermine si la défaillance d'un composant entraîne l'échec du composant parent. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none"> - Sélectionné. La défaillance du composant n'entraîne pas l'échec du composant parent. - Effacé. La défaillance du composant entraîne l'échec du composant parent. Par défaut, la valeur est vide. Pour plus d'informations sur l'échec des composants, consultez la section "Traitement des échecs" à la page 404 .
recurring_element	Définit la zone de stockage des données identifié par le composant.
remarque	Commentaire défini par l'utilisateur, qui décrit l'objectif ou l'action du composant.

Conflits entre les localisateurs

En cas de conflits, un **LocatorByOccurrence** imbriqué remplace un localisateur parent.

Imaginez, par exemple, que la propriété **cible** d'un **Groupe** contienne un **LocatorByOccurrence** pointant sur la troisième occurrence d'un élément. Un **Groupe** imbriqué contient un **LocatorByOccurrence** pointant vers la cinquième occurrence. Le **Groupe** imbriqué utilise la cinquième occurrence.

CHAPITRE 22

Répartiteurs

Ce chapitre comprend les rubriques suivantes :

- [Présentation des répartiteurs, 382](#)
- [Répartiteurs de texte, 383](#)
- [Répartiteurs XML, 387](#)
- [Propriétés standard des répartiteurs, 389](#)
- [Référence des composants du répartiteur, 390](#)
- [Référence des sous-composants de répartiteur, 399](#)

Présentation des répartiteurs

Un répartiteur partage un document source volumineux en portions plus réduites, qu'une transformation peut traiter séparément. Les répartiteurs sont utiles dans les transformations qui traitent des entrées très volumineuses, par exemple des flux de données de plusieurs gigaoctets. Un répartiteur peut avoir une entrée de tampon ou une entrée de fichier.

Un répartiteur présente les avantages suivants :

- La transformation analyse chaque segment source lorsqu'il est disponible, au lieu d'attendre de recevoir l'intégralité de la source.
- La transformation a réduit la mémoire requise.

Par exemple, le flux d'entrée d'un répartiteur peut contenir des données de transactions boursières. Le flux est transmis à un serveur sans interruption tout au long de la session boursière. Un script doté d'un répartiteur traite chaque transaction lors de son arrivée, au lieu d'attendre la fin de la journée.

Prenons un autre exemple. Supposons que vous receviez un fichier source volumineux via une connexion FTP. Grâce à un répartiteur, le script peut traiter du fichier avant qu'il ne soit entièrement reçu.

La transformation Processeur de données fournit les types de répartiteurs suivants :

- Répartiteur. Traite des entrées de texte volumineuses.
- XmlStreamer. Traite des entrées XML volumineuses.

Les répartiteurs sont des composants exécutables. Définissez le composant **Répartiteur** ou **XmlStreamer** au niveau global du script et définissez-le en tant que composant de démarrage de la transformation. Le répartiteur fractionne l'entrée en segments et les transmet aux autres composants exécutables, qui peuvent être des analyseurs, des mappeurs ou des sérialiseurs.

Répartiteurs de texte

Un composant `Répartiteur` partage un document texte volumineux en portions plus réduites. Le répartiteur divise la source du texte en un en-tête, un pied de page et des segments récurrents. Comme requis par la structure de la source, le composant `Répartiteur` peut subdiviser les segments récurrents en en-têtes imbriqués, pieds de page et segments récurrents. Le composant `Répartiteur` peut transmettre chaque segment à une transformation appropriée.

segments

Un répartiteur identifie les segments de son entrée. Il transmet les segments individuellement à des transformations telles que des analyseurs, des mappeurs ou des sérialiseurs, qui traitent les données du segment.

Un répartiteur suppose que la source est composée des éléments suivants :

- Un segment d'en-tête
- Un certain nombre de segments répétitifs
- Un segment de pied de page

Pour chaque type de segment, le répartiteur définit une transformation qui traite le segment.

Les segments répétitifs peuvent être soit simples, soit complexes. Un segment simple est une unité de données unique. Un segment complexe possède ses propres en-tête, segments répétitifs et pied de page imbriqués.

Les en-têtes et pieds de page sont toujours des segments simples.

Segments simples

Un segment simple possède un marqueur d'ouverture qui identifie l'endroit où il commence, et un marqueur de fermeture qui identifie l'endroit où il se termine. Par conséquent, la structure d'un segment simple est la suivante :

```
Opening marker
Data
Closing marker
```

Le répartiteur transmet le segment au composant de transformation spécifié, par exemple un analyseur.

Il est possible d'omettre certains marqueurs de la définition du répartiteur. Par exemple :

- Si vous excluez le marqueur d'ouverture de l'en-tête source, l'en-tête est supposée commencer au début de la source.
- Si vous excluez le marqueur de fermeture, alors le segment se termine au marqueur d'ouverture du segment suivant.

Segments complexes

Un segment complexe a un en-tête et un pied de page. Entre l'en-tête et le pied de page, il peut contenir n'importe quel nombre de segments simples imbriqués, par exemple :

```
Header
Simple segment
Simple segment
Simple segment
Footer
```

Un segment complexe peut contenir des segments complexes imbriqués, par exemple :

```
Header
Complex segment
Complex segment
Complex segment
Footer
```

Vous pouvez également définir un segment complexe pour qui il manque l'en-tête ou le pied de page, par exemple :

```
Simple segment
Simple segment
Simple segment
```

Les segments simples imbriqués doivent tous être du même type. Ils doivent par conséquent tous être identifiés par les mêmes marqueurs d'ouverture et de fermeture.

Exemple

Un flux de données contient des données de transaction boursière. Le flux a la structure suivante :

- L'en-tête commence par la chaîne `yy-MM-dd` qui est une date suivie par une barre oblique.
- L'en-tête contient différentes données, suivies par la chaîne `ENDHEAD/`.
- Les segments répétitifs commencent par la chaîne `TRANS HH:mm nnn/`, où `HH:mm` est l'heure sur une horloge de 24 heures et `nnn` est un numéro de série de n'importe quelle longueur.
- Le flux de données se termine par la chaîne `END/`.

Le flux de données d'exemple suivant est conforme à cette spécification, où `...` représente des données arbitraires devant être analysées :

```
06-12-13/...ENDHEAD/TRANS 09:30 1...TRANS 09:30 2...TRANS 09:31 03...TRANS 09:32
14...END/
```

Vous pouvez analyser ce flux en utilisant un répartiteur dont la structure schématique est la suivante : Notez que les marqueurs d'ouverture et de fermeture sont localisés en recherchant un modèle ou une chaîne particuliers.

Segment	Type	Marqueur d'ouverture	Marqueur de fermeture
En-tête	Simple	[0-9][0-9]-[0-9][0-9]-[0-9][0-9]/	ENDHEAD/
Répétitif	Simple	TRANS [0-9][0-9]:[0-9][0-9] [0-9]+/	none
Pied de page	Simple	END/	none

Concaténation de l'en-tête

Éventuellement, vous pouvez configurer un répartiteur pour concaténer l'en-tête du segment avec chaque segment récurrent. Le répartiteur transmet le résultat concaténé à une transformation.

Par exemple, supposons qu'un répartiteur transmette le segment récurrent à un analyseur. La source a la structure suivante, où `Segment1` et suivants sont des instances du segment répété :

```
Header
Segment1
Segment2
Segment3
```


Si vous sélectionnez l'option de concaténation, le répartiteur envoie les données suivantes à l'analyseur :

```
HeaderSegment1
HeaderSegment2
HeaderSegment3
```

Sortie d'un répartiteur

Un répartiteur génère un document de sortie indépendant pour chacun des segments source.

Sortie dans l'environnement de conception

Si vous exécutez un répartiteur dans l'environnement de conception, il combine les segments de sortie individuels en une seule sortie.

Par exemple, supposons que le répartiteur transmette chaque segment à un analyseur. La sortie de chaque analyseur est un document XML. La sortie combinée est une séquence de documents XML, par exemple :

```
<?xml version="1.0" encoding="windows-1252"?>
<header>...</header>

<?xml version="1.0" encoding="windows-1252"?>
<repeating_segment>...</repeating_segment>

<?xml version="1.0" encoding="windows-1252"?>
<repeating_segment>...</repeating_segment>

<?xml version="1.0" encoding="windows-1252"?>
<footer>...</footer>
```

Cette sortie n'est pas un XML bien structuré, car elle contient plusieurs éléments racines.

Enveloppement d'une sortie dans une balise racine

Vous pouvez envelopper la sortie combinée d'un répartiteur dans une balise racine pour convertir la sortie en langage XML bien formé.

Dans l'onglet Génération XML des paramètres de la transformation Processeur de données, sélectionnez **Ajouter un élément racine XML** et entrez le nom de l'élément racine wrapper.

Par exemple, si vous spécifiez un élément racine wrapper nommé `MyRoot`, la sortie devient :

```
<MyRoot>
  <?xml version="1.0" encoding="windows-1252"?>
  <header>...</header>

  <?xml version="1.0" encoding="windows-1252"?>
  <repeating_segment>...</repeating_segment>

  <?xml version="1.0" encoding="windows-1252"?>
  <repeating_segment>...</repeating_segment>

  <?xml version="1.0" encoding="windows-1252"?>
  <footer>...</footer>
</MyRoot>
```

Utilisation de marqueurs et de variables dans les répartiteurs

Dans un composant Répartiteur, vous ne pouvez pas utiliser les composants ordinaires `Marqueur` et `Variable` qui sont utilisés dans d'autres types de transformations. Au lieu de cela, vous devez utiliser le composant `MarkerStreamer` pour définir les marqueurs d'ouverture et de fermeture des segments simples. Vous pouvez utiliser les composants `StreamerVariable` pour stocker les données temporaires partagées par tous les segments.

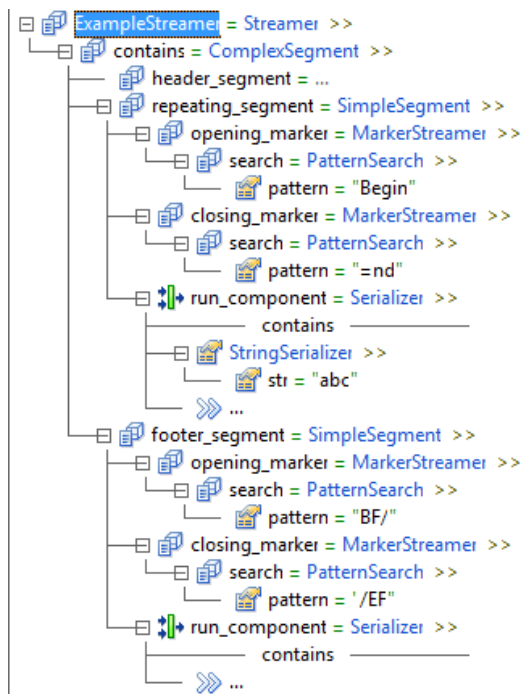
Création d'un Répartiteur

1. Analysez la structure source et identifiez les types de segments.
2. Créez ou ouvrez un script.
3. Dans le script, configurez une transformation, telle qu'un analyseur, un mappeur ou un sérialiseur, qui peut traiter chaque type de segment simple.
4. Dans le même script, configurez un composant **Répartiteur**.
5. Dans le **Répartiteur**, imbriquez les composants **ComplexSegment** et **SimpleSegment** correspondant à la structure source.
6. Pour chaque **SimpleSegment**, définissez le marqueur d'ouverture et le marqueur de fermeture si nécessaire. Définissez la transformation qui traite le segment.
7. Définissez le **Répartiteur** comme composant de démarrage.

Exemple 1 de configuration de répartiteur

Le répartiteur suivant contient des segments simples. Chaque segment comporte un marqueur d'ouverture et de fermeture prédéfini.

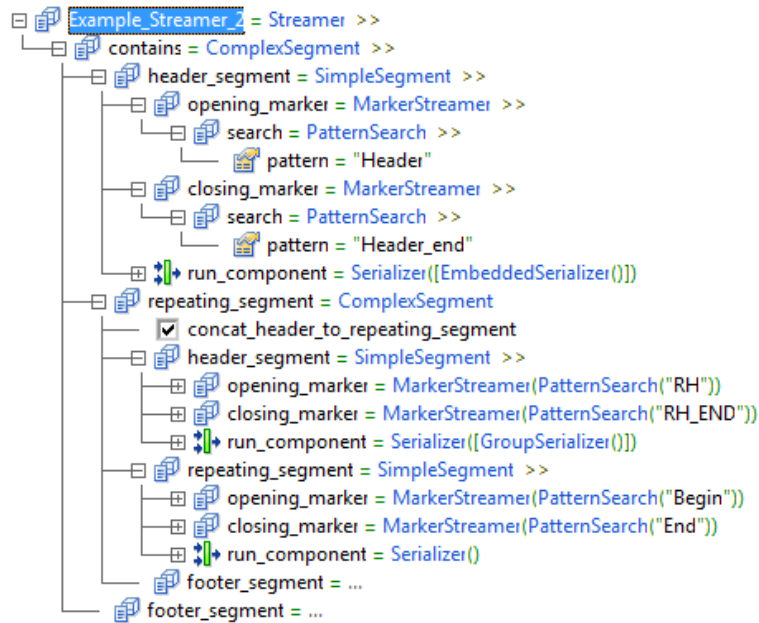
Le répartiteur transmet l'en-tête et les segments récurrents à un analyseur appelé `body_p`. Il transmet le bas de page à un analyseur appelé `foot_p`.



Exemple 2 de configuration du répartiteur

Le répartiteur suivant contient un `ComplexSegment` imbriqué, récurrent. Le segment imbriqué `ComplexSegment` possède ses propres en-tête et `SimpleSegment` imbriqué récurrent. Le `ComplexSegment` imbriqué ne comporte pas de bas de page.

Notez que la propriété `concat_header_to_repeating_segment` a été sélectionnée. L'effet de cette propriété est de concaténer l'en-tête pour chaque instance du segment récurrent. Le répartiteur transmet les segments concaténés à l'analyseur `body_p`.



Répartiteurs XML

Un composant `XmlStreamer` divise un grand document XML en portions plus petites. Le composant `XmlStreamer` divise la source XML en trois types de segment : en-tête, corps et pied de page. Les segments de corps peuvent contenir des éléments répétitifs ou non. `XmlStreamer` peut transmettre chaque segment XML à une transformation adaptée, généralement à un mappeur ou à un sérialiseur.

`XmlStreamer` a un fonctionnement similaire à celui d'un répartiteur, avec quelques différences dues à l'entrée XML structurée. Les fonctions principales sont les suivantes :

- Les segments de corps sont définis comme des éléments XML. Vous pouvez configurer le corps avec plusieurs éléments de même type ou de types différents, dans n'importe quelle séquence.
- L'en-tête est défini comme la portion complète du XML qui précède le premier segment de corps. Dans la configuration `XmlStreamer`, il n'est pas nécessaire de définir les éléments qui comprennent l'en-tête.
- Le pied de page est défini comme la portion complète du XML qui suit le dernier segment de corps. Dans la configuration `XmlStreamer`, il n'est pas nécessaire de définir les éléments qui comprennent le pied de page.
- Dans de nombreux cas, les segments d'en-tête et de pied de page ne sont pas des documents XML bien formés. Pour activer la transmission des segments à un mappeur ou à un sérialiseur, vous pouvez configurer des composants de modificateur qui convertissent les segments en documents XML bien formés.

Pour aider à comprendre ces fonctions, observez la structure de source XML suivante :

```
<stream>
  <headerline1>MainHeader</headerline1>
  <substreams>
```

```

<substream>
  <subheaderline1>SubHeader</subheaderline1>
  <segments>
    <segment1>Segment1A</segment1>
    <segment1>Segment1B</segment1>
    <segment2>Segment2A</segment2>
    <segment1>Segment1C</segment1>
    <segment2>Segment2B</segment2>
  </segments>
  <subfooterline1>SubFooter</subfooterline1>
</substream>
<substream>...</substream>
<substream>...</substream>
</substreams>
<footerline1>MainFooter</footerline1>
</stream>

```

Dans cet exemple, vous pouvez définir les segments de corps comme des éléments `substream`. L'en-tête correspond à tout qui précède le premier élément `substream` :

```

<stream>
  <headerline1>MainHeader</headerline1>
  <substreams>

```

Le pied de page correspond à tout ce qui suit le dernier élément `substream` :

```

  </substreams>
  <footerline1>MainFooter</footerline1>
</stream>

```

Les segments d'en-tête et de pied de page ne sont pas des documents XML bien formés. Vous pouvez appliquer des modificateurs qui ajoutent des balises de fermeture ou d'ouverture pour les former correctement. Par exemple, un modificateur peut convertir l'en-tête en :

```

<stream>
  <headerline1>MainHeader</headerline1>
  <substreams>
  </substreams>
</stream>

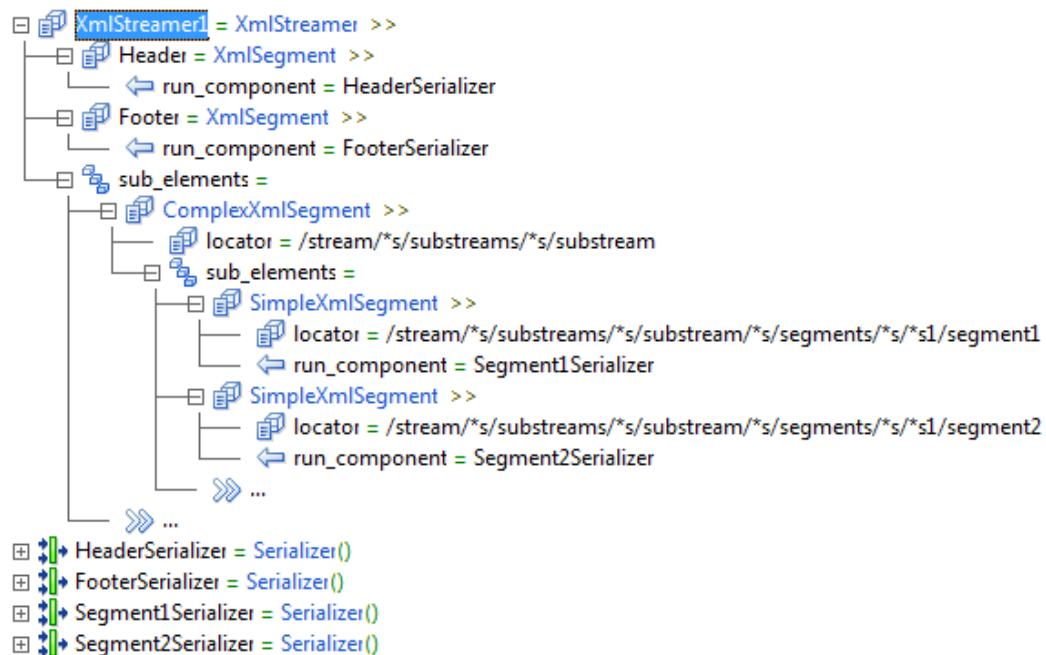
```

Vous pouvez configurer `XmlStreamer` pour qu'il transmette le segment d'en-tête, le segment de pied de page et chaque instance du segment `substream` à une transformation adaptée, telle qu'un mappeur ou un sérialiseur.

Remarque: les éléments du segment d'en-tête sont disponibles lors du traitement des éléments de corps. Les éléments de pied de page, par contre, ne sont pas encore disponibles à ce moment-là. Les éléments de pied de page sont traités uniquement après que la transformation a terminé de lire les éléments de corps.

Vous pouvez également subdiviser les éléments `substream` en deux segments (`segment1` et `segment2`), puis envoyer chacun de ces segments à son propre mappeur ou sérialiseur. Notez que `segment1` et `segment2` se suivent dans une séquence aléatoire. `XmlStreamer` ignore la séquence et traite `segment1` et `segment2`, peu importe l'ordre dans lequel ils apparaissent.

La figure suivante illustre la configuration à suivre pour atteindre cet objectif. Le script définit des sérialiseurs indépendants pour les segments d'en-tête, de pied de page, `segment1` et `segment2`.



Remarque: Même si le composant d'exécution du pied de page apparaît avant les éléments de corps dans cet exemple, les éléments de pied de page sont traités uniquement après que la transformation a terminé de lire les éléments de corps.

Pour affiner encore davantage, vous pouvez définir des transformations pour les en-têtes et les pieds de page imbriqués dans chaque élément substream.

Propriétés standard des répartiteurs

Le tableau suivant décrit les propriétés communes des composants des répartiteurs :

Propriété	Description
désactivé	Détermine si le script ignore le composant et tous les composants enfants. Utilisez cette propriété pour tester, déboguer et modifier un script. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none"> - Sélectionné. Le script ignore le composant. - Effacé. Le script applique le composant. La valeur par défaut est vide.
nom	Libellé descriptif pour le composant. Ce libellé s'affiche dans le fichier journal et la vue Événements . Utilisez la propriété nom pour identifier le composant ayant provoqué l'événement.
remarque	Commentaire défini par l'utilisateur, qui décrit l'objectif ou l'action du composant.

Référence des composants du répartiteur

Les répartiteurs divisent les documents source volumineux en portions plus réduites qu'une transformation peut traiter séparément.

ComplexSegment

Un composant **ComplexSegment** définit une structure source qui contient un en-tête, une partie répétitive et un bas de page.

Le tableau suivant décrit les propriétés du composant **ComplexSegment** :

Propriété	Description
concat_header_to_repeating_segment	Détermine si le système inclut ou non header_segment à chaque instance du repeating_segment . Il transmet le résultat au run_component du repeating_segment . Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none">- Sélectionné. Chaque segment répétitif a une copie de l'en-tête.- Effacé. Chaque segment répétitif apparaît sans en-tête. Pour plus d'informations, consultez la section " Concaténation de l'en-tête " à la page 384 . Par défaut, la valeur est vide.
footer_segment	Définit la partie du pied de page de la source. Dans cette propriété, vous pouvez imbriquer un SimpleSegment qui définit le bas de page. Si cette propriété n'est pas définie, le script traite la source comme si elle n'avait pas de bas de page.
header_segment	Définit la partie en-tête de la source. Dans cette propriété, vous pouvez imbriquer un SimpleSegment qui définit l'en-tête. Si cette propriété n'est pas définie, le script traite la source comme si elle n'avait pas d'en-tête.
repeating_segment	Définit la partie répétitive de la source. Dans cette propriété, vous pouvez imbriquer un SimpleSegment qui définit les données répétitives. Vous pouvez également imbriquer un ComplexSegment qui a sa propre structure en-tête-répétition-bas de page.

ComplexXmlSegment

Un composant **ComplexXmlSegment** définit une structure imbriquée dans le corps d'une entrée **XmlStreamer**. La structure imbriquée peut avoir ses propres en-tête, corps et bas de page.

Dans un composant **ComplexXmlSegment**, vous pouvez imbriquer les composants **XmlSegment**, **ComplexXmlSegment** et **SimpleXmlSegment**.

Le tableau suivant décrit les propriétés du composant **ComplexXmlSegment** :

Propriété	Description
allow_unmarked_text	Détermine si les segments dans la liste sub_elements peuvent être séparés par du texte intermédiaire ou par d'autres éléments. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none"> - Sélectionné. Les segments peuvent être séparés par du texte intermédiaire ou par d'autres éléments. Le contenu intermédiaire est ignoré. - Effacé. Les segments ne peuvent être séparés que par des espaces. La valeur par défaut est Effacé.
désactivé	Détermine si le script ignore le composant et tous les composants enfants. Utilisez cette propriété pour tester, déboguer et modifier un script. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none"> - Sélectionné. Le script ignore le composant. - Effacé. Le script applique le composant. La valeur par défaut est vide.
Pied de page	Définit la méthode de traitement du bas de page de ComplexXmlSegment . Pour plus d'informations, consultez la section " XmlSegment " à la page 397. La valeur par défaut est XmlSegment.
En-tête	Définit la méthode de traitement de l'en-tête de ComplexXmlSegment . Pour plus d'informations, consultez la section " XmlSegment " à la page 397. La valeur par défaut est XmlSegment.
localisateur	Définit une zone de stockage des données.
nom	Libellé descriptif pour le composant. Ce libellé s'affiche dans le fichier journal et la vue Événements . Utilisez la propriété nom pour identifier le composant ayant provoqué l'événement.
remarque	Commentaire défini par l'utilisateur, qui décrit l'objectif ou l'action du composant.
sub_elements	Définit une liste de composants ComplexXmlSegment ou SimpleXmlSegment qui définissent la méthode de traitement du corps de ComplexXmlSegment .

JsonStreamer

Le composant **JsonStreamer** accepte une très grande entrée de fichier JSON qu'il fractionne en segments. Il transmet chaque type de segment à une transformation prédéfinie, par exemple un analyseur, un mappeur ou un sérialiseur.

JsonStreamer doit être défini au niveau global du script et doit être le composant de démarrage de la transformation.

Remarque: Le nombre de segments est déterminé automatiquement en fonction de la précision du port d'entrée.

Le tableau suivant décrit les propriétés du composant **JsonStreamer** :

Propriété	Description
run_component	Définit une transformation qui traite le segment. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none">- Le nom d'un analyseur, d'un sérialiseur ou d'un mappeur configuré au niveau global du script.- Un composant Mappeur ou Sérialiseur. Configurez le mappeur ou le sérialiseur à l'intérieur du segment.- Un composant WriteSegment qui copie le segment pour la sortie. Pour plus d'informations, voir "WriteSegment" à la page 401

MarkerStreamer

Un composant **MarkerStreamer** définit les marqueurs d'ouverture et de fermeture de segments simples. Il est similaire à une ancre **Marqueur** normale, mais il est uniquement utilisé dans des répartiteurs.

Le tableau suivant décrit les propriétés du composant **MarkerStreamer** :

Propriété	Description
adjacent	Détermine si le MarkerStreamer doit être adjacent à la fin du segment précédent. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none">- Sélectionné. Requiert que MarkerStreamer soit adjacent à la fin du précédent segment.- Effacé. MarkerStreamer peut être séparé de la fin du précédent segment. La valeur par défaut est Effacé. Utilisez cette propriété pour vous assurer que les segments ne sont pas séparés par un autre texte ou des espaces.
compteur	Détermine l'occurrence du marqueur par laquelle commencer le traitement. Par exemple, définissez compteur sur 3 pour ignorer la première et la deuxième occurrence du marqueur. Cette propriété est en cours de suppression. Elle est disponible pour une question de compatibilité avec les projets existants. Ne l'utilisez pas dans les nouveaux projets.
désactivé	Détermine si le script ignore le composant et tous les composants enfants. Utilisez cette propriété pour tester, déboguer et modifier un script. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none">- Sélectionné. Le script ignore le composant.- Effacé. Le script applique le composant. La valeur par défaut est vide.
marquage	Détermine si le marqueur est utilisé comme un point de référence pour identifier le segment ou le marqueur suivant. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none">- position de départ. Avant uniquement.- position de fin. Après uniquement.- complet. Place un point de référence avant et après le marqueur actuel. La valeur par défaut est Complet.
nom	Libellé descriptif pour le composant. Ce libellé s'affiche dans le fichier journal et la vue Événements . Utilisez la propriété nom pour identifier le composant ayant provoqué l'événement.
remarque	Commentaire défini par l'utilisateur, qui décrit l'objectif ou l'action du composant.
rechercher	Définit la façon dont MarkerStreamer trouve un texte. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none">- NewlineSearch. Recherche un caractère de retour à la ligne.- OffsetSearch. Ignore un nombre de caractères prédéfini après le point de référence précédent.- PatternSearch. Recherche une expression régulière.- TextSearch. Recherche une chaîne explicite.

Utilisation de la propriété de marquage pour définir des limites de segments

Vous pouvez utiliser la propriété de marquage pour contrôler si les données des marqueurs d'ouverture et de fermeture sont incluses dans le segment et transmises à une transformation.

Généralement, le répartiteur transmet les données entre les points de référence les plus internes qui entourent le segment. Par exemple :

- Si le marqueur d'ouverture comprend `marker = begin position`, le point de référence le plus interne se trouve au début. L'ensemble du marqueur est inclus dans le segment.
- Si le marqueur d'ouverture comprend `marker = end position` ou `full`, le point de référence le plus interne se trouve à la fin. Le marqueur est exclu du segment.

La relation inverse s'applique au marqueur de fermeture.

En guise d'exemple, considérons un segment simple présentant la structure suivante :

```
BEGIN...data...END
```

Un `MarkerStreamer` identifie le marqueur d'ouverture en recherchant le texte `BEGIN`. Un autre `MarkerStreamer` identifie le marqueur de fermeture en recherchant `END`.

Le tableau suivant illustre comment la propriété de marquage affecte les limites de segments.

Marquage du marqueur d'ouverture	Marquage du marqueur de fermeture	Segment transmis à la transformation
full	full	...data...
full	begin	...data...
full	end	...data...END
begin	full	BEGIN...data...
begin	begin	BEGIN...data...
begin	end	BEGIN...data...END
end	full	...data...
end	begin	...data...
end	end	...data...END

SimpleSegment

Un composant **SimpleSegment** définit une unité de données qui contient un marqueur d'ouverture et un marqueur de fermeture. Elle définit également la transformation qui traite l'unité de données.

Les marqueurs d'ouverture et de fermeture sont définis par des expressions régulières. Pour plus d'informations sur la syntaxe d'expression régulière, voir ["RegularExpression" à la page 283](#).

Le tableau suivant décrit les propriétés du composant **SimpleSegment** :

Propriété	Description
closing_marker	Définit une expression régulière qui identifie la fin du segment. Si cette propriété n'est pas définie, la fin du segment correspond à la fin de la source ou au début du segment suivant. La valeur par défaut est <code>MarkerStreamer</code> .
compteur	Définit le nombre maximum de segments à transmettre à la transformation. Par exemple, si count est égal à 3, le répartiteur recherche trois instances consécutives du segment. Il transmet les trois segments conjointement à la transformation. S'il ne trouve qu'un ou deux segments, il transmet ces segments. Si les segments sont petits, la transmission de plusieurs segments à une transformation peut améliorer les performances, car cela réduit la charge du répartiteur. Dans la transformation, utilisez un composant comme RepeatingGroup pour traiter les segments individuels. La valeur par défaut est 1.
désactivé	Détermine si le script ignore le composant et tous les composants enfants. Utilisez cette propriété pour tester, déboguer et modifier un script. Vous pouvez choisir l'une des options suivantes : - Sélectionné. Le script ignore le composant. - Effacé. Le script applique le composant. La valeur par défaut est vide.
nom	Libellé descriptif pour le composant. Ce libellé s'affiche dans le fichier journal et la vue Événements . Utilisez la propriété nom pour identifier le composant ayant provoqué l'événement.
opening_marker	Définit une expression régulière qui identifie le début du segment. Si cette propriété n'est pas définie, le début du segment correspond au début de la source ou à la fin du segment précédent. La valeur par défaut est <code>MarkerStreamer</code> .
remarque	Commentaire défini par l'utilisateur, qui décrit l'objectif ou l'action du composant.
run_component	Définit une transformation qui traite le segment. Vous pouvez choisir l'une des options suivantes : - Le nom d'un analyseur, d'un sérialiseur ou d'un mappeur configuré au niveau global du script. - Un composant Mappeur ou Sérialiseur . Configurez le mappeur ou le sérialiseur à l'intérieur du segment. - Un composant WriteSegment qui copie le segment pour la sortie. Pour plus d'informations, voir "WriteSegment" à la page 401

SimpleXmlSegment

Un composant **SimpleXmlSegment** définit un composant corps d'une entrée **XmlStreamer**. Il définit l'élément qui contient le segment et la transformation qui doivent traiter le segment.

Puisque **SimpleXmlSegment** est un élément XML, le segment est toujours formé correctement. Vous pouvez appliquer un modificateur qui altère le segment avant qu'il ne soit transmis à une transformation.

Le tableau suivant décrit les propriétés du composant **SimpleXmlSegment** :

Propriété	Description
compteur	Définit le nombre maximum de segments à transmettre à la transformation. Par exemple, si count est égal à 3, le répartiteur recherche trois instances consécutives du segment. Il transmet les trois segments conjointement à la transformation. S'il ne trouve qu'un ou deux segments, il transmet ces segments. Si les segments sont petits, la transmission de plusieurs segments à une transformation peut améliorer les performances, car cela réduit la charge du répartiteur. Dans la transformation, utilisez un composant comme RepeatingGroup pour traiter les segments individuels. La valeur par défaut est 1.
désactivé	Détermine si le script ignore le composant et tous les composants enfants. Utilisez cette propriété pour tester, déboguer et modifier un script. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none"> - Sélectionné. Le script ignore le composant. - Effacé. Le script applique le composant. La valeur par défaut est vide.
localisateur	Définit une zone de stockage des données.
modifier	Définit la manière dont le segment est modifié avant qu'il ne soit transmis à une transformation. Vous pouvez sélectionner les composants de modificateur suivants : <ul style="list-style-type: none"> - AddHeaderModifier. Transmet le segment conjointement à l'en-tête de la section XML dans laquelle le segment est situé. - AddStringModifier. Concatène le segment avec des chaînes de préfixe ou de suffixe. - DoNothingModifier. Ne modifie pas le segment. - WellFormedModifier. Ajoute des balises de fermeture et/ou un élément racine pour garantir que le segment adopte un langage XML bien formé. Pour plus d'informations sur les modificateurs, voir " Référence des sous-composants de répartiteur " à la page 399. La valeur par défaut est DoNothingModifier.
nom	Libellé descriptif pour le composant. Ce libellé s'affiche dans le fichier journal et la vue Événements . Utilisez la propriété nom pour identifier le composant ayant provoqué l'événement.
remarque	Commentaire défini par l'utilisateur, qui décrit l'objectif ou l'action du composant.
run_component	Définit une transformation qui traite le segment. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none"> - Le nom d'un analyseur, d'un sérialiseur ou d'un mappeur configuré au niveau global du script. - Un composant Mappeur ou Sérialiseur. Configurez le mappeur ou le sérialiseur à l'intérieur du segment. - Un composant WriteSegment qui copie le segment pour la sortie. Pour plus d'informations, voir "WriteSegment" à la page 401

Répartiteur

Le composant **Répartiteur** divise l'entrée de texte en segments. Il transmet chaque type de segment à une transformation prédéfinie, par exemple un analyseur, un mappeur ou un sérialiseur.

Le **Répartiteur** doit être défini au niveau global du script et doit être le composant de démarrage de la transformation.

Sous un **Répartiteur**, vous devez imbriquer un **ComplexSegment**. Le **ComplexSegment** peut contenir des **SimpleSegment** imbriqués ou des composants **ComplexSegment**.

Le tableau suivant décrit les propriétés du composant **Répartiteur** :

Propriété	Description
contient	Définit la structure globale de la source. La valeur par défaut est <code>ComplexSegment</code> .
désactivé	Détermine si le script ignore le composant et tous les composants enfants. Utilisez cette propriété pour tester, déboguer et modifier un script. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none"> - Sélectionné. Le script ignore le composant. - Effacé. Le script applique le composant. La valeur par défaut est vide.
max_lookup_size	Définit la quantité maximum de nouvelles données, en kilooctets, que le répartiteur recherche pour chaque nouveau segment. Pour des performances optimales, définissez cette propriété à deux fois la taille de segment maximum. Quand une application active un service déployé de Répartiteur via un API, il doit définir le paramètre de la taille de fragment à une valeur inférieure à max_lookup_size . Valeur par défaut : 10000.
nom	Libellé descriptif pour le composant. Ce libellé s'affiche dans le fichier journal et la vue Événements . Utilisez la propriété nom pour identifier le composant ayant provoqué l'événement.
on_end_of_input	Définit une transformation qui s'exécute à la fin du flux d'entrée. Par exemple, la transformation peut envoyer un message de synthèse. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none"> - Le nom d'un analyseur, d'un sérialiseur ou d'un mappeur configuré au niveau global du script. - Un composant Mappeur ou Sérialiseur. Configurez le mappeur ou le sérialiseur dans le répartiteur.
remarque	Commentaire défini par l'utilisateur, qui décrit l'objectif ou l'action du composant.
root_tag	Définit une balise XML dans laquelle le répartiteur intègre la sortie combinée de tous les segments. Pour plus d'informations, consultez la section "Sortie d'un répartiteur" à la page 385 .

StreamerVariable

Un composant **StreamerVariable** est une variable définie par l'utilisateur dont le champ comprend tous les segments d'un **Répartiteur** ou **XmlStreamer**.

Par exemple, si un répartiteur contient trois analyseurs, la valeur de **StreamerVariable** est disponible pour ces trois analyseurs. Un analyseur qui traite un segment d'en-tête peut récupérer les données de l'en-tête et les stocker dans **StreamerVariable**. Les autres analyseurs, qui traitent le segment récurrent et le segment de pied de page, peuvent accéder à la valeur de **StreamerVariable**. Vous ne pouvez pas utiliser de **variable** ordinaire pour cette fonction, car la valeur de la variable n'est pas partagée entre segments.

Par ailleurs, le composant **StreamerVariable** est similaire à une **variable** ordinaire. Cependant, **StreamerVariable** doit présenter un type de données simples, à occurrence unique. Pour plus d'informations, voir ["Variables" à la page 198](#)

Vous pouvez définir **StreamerVariable** uniquement au niveau global du script.

Le tableau suivant décrit les propriétés du composant **StreamerVariable** :

Propriété	Description
initialisation	Valeur initiale de StreamerVariable , attribuée lors du démarrage de la transformation. Sélectionnez InitialValue et entrez la valeur.
val_type	Définit le type de données que la variable peut stocker. Attribue un type simple comme <code>xs:string</code> ou <code>xs:integer</code> . Les variables de répartiteur ne peuvent pas comporter de types de données complexes ou à occurrences multiples. La valeur par défaut est <code>xs:string</code> .

XmlSegment

Un composant **XmlSegment** définit un segment d'en-tête ou de pied de page pour une entrée **XmlStreamer**. Il définit également la transformation qui traite l'en-tête ou le pied de page.

Un en-tête ou un pied de page non modifié ne correspond pas forcément à un langage XML approprié. En assignant un composant modificateur, vous pouvez configurer **XmlSegment** pour qu'il renvoie toujours des textes XML bien formés.

Le tableau suivant décrit les propriétés du composant **XmlSegment** :

Propriété	Description
désactivé	Détermine si le script ignore le composant et tous les composants enfants. Utilisez cette propriété pour tester, déboguer et modifier un script. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none"> - Sélectionné. Le script ignore le composant. - Effacé. Le script applique le composant. La valeur par défaut est vide.
modifier	Définit la manière dont le segment est modifié avant qu'il ne soit transmis à une transformation. Vous pouvez sélectionner les composants de modificateur suivants : <ul style="list-style-type: none"> - AddHeaderModifier. Transmet le segment conjointement à l'en-tête de la section XML dans laquelle le segment est situé. - AddStringModifier. Concatène le segment avec des chaînes de préfixe ou de suffixe. - DoNothingModifier. Ne modifie pas le segment. Il s'agit de la valeur par défaut. - WellFormedModifier. Ajoute des balises de fermeture et/ou un élément racine pour garantir que le segment adopte un langage XML bien formé. Pour plus d'informations sur les modificateurs, voir "Référence des sous-composants de répartiteur" à la page 399 . La valeur par défaut est WellFormedModifier.
nom	Libellé descriptif pour le composant. Ce libellé s'affiche dans le fichier journal et la vue Événements . Utilisez la propriété nom pour identifier le composant ayant provoqué l'événement.
remarque	Commentaire défini par l'utilisateur, qui décrit l'objectif ou l'action du composant.
run_component	Définit une transformation qui traite le segment. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none"> - Le nom d'un analyseur, d'un sérialiseur ou d'un mappeur configuré au niveau global du script. - Un composant Mappeur ou Sérialiseur. Configurez le mappeur ou le sérialiseur à l'intérieur du segment. - Un composant WriteSegment qui copie le segment pour la sortie. Pour plus d'informations, voir "WriteSegment" à la page 401

XmlStreamer

Le composant **XmlStreamer** divise une entrée XML en des segments d'en-tête, de corps et de pied de page. Il transmet chaque type de segment à une transformation prédéfinie, par exemple un mappeur ou un sérialiseur.

Le **XmlStreamer** doit être défini au niveau global du script et doit être le composant de démarrage de la transformation.

Sous **XmlStreamer**, vous pouvez imbriquer des composants **XmlSegment**, **ComplexXmlSegment** et **SimpleXmlSegment**.

Le tableau suivant décrit les propriétés du composant **XmlStreamer** :

Propriété	Description
allow_unmarked_text	Détermine si les segments dans la liste sub_elements peuvent être séparés par du texte intermédiaire ou par d'autres éléments. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none">- Sélectionné. Les segments peuvent être séparés par du texte intermédiaire ou par d'autres éléments. Le contenu intermédiaire est ignoré.- Effacé. Les segments ne peuvent être séparés que par des espaces. La valeur par défaut est Effacé.
désactivé	Détermine si le script ignore le composant et tous les composants enfants. Utilisez cette propriété pour tester, déboguer et modifier un script. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none">- Sélectionné. Le script ignore le composant.- Effacé. Le script applique le composant. La valeur par défaut est vide.
Pied de page	Définit la méthode de traitement du bas de page de ComplexXmlSegment . Pour plus d'informations, consultez la section " XmlSegment " à la page 397. La valeur par défaut est XmlSegment.
En-tête	Définit la méthode de traitement de l'en-tête de ComplexXmlSegment . Pour plus d'informations, consultez la section " XmlSegment " à la page 397. La valeur par défaut est XmlSegment.
max_lookup_size	Définit la quantité maximum de nouvelles données, en kilo-octets, que XmlStreamer recherche pour chaque nouveau segment. Pour des performances optimales, définissez cette propriété à deux fois la taille de segment maximum. Lorsqu'une application active un service XmlStreamer déployé via une API, elle doit définir le paramètre de taille de fragment à une valeur inférieure à la taille max_lookup_size . Valeur par défaut : 10000.
nom	Libellé descriptif pour le composant. Ce libellé s'affiche dans le fichier journal et la vue Événements . Utilisez la propriété nom pour identifier le composant ayant provoqué l'événement.
remarque	Commentaire défini par l'utilisateur, qui décrit l'objectif ou l'action du composant.
sub_elements	Définit une liste de composants ComplexXmlSegment ou SimpleXmlSegment qui définissent comment traiter le corps de l'entrée XML.

Référence des sous-composants de répartiteur

Les sous-composants de répartiteur modifient les segments d'un **Répartiteur** ou d'un **XmlStreamer**.

AddHeaderModifier

Dans **XmlStreamer**, le composant **AddHeaderModifier** ajoute l'en-tête du segment actuel au segment. Le composant ajoute des balises de fermeture comme requis pour garantir que le résultat est du XML bien structuré.

Vous pouvez utiliser **AddHeaderModifier** pour transmettre le segment à une transformation, dans le contexte de son en-tête.

Le tableau suivant décrit les propriétés du composant **AddHeaderModifier** :

Propriété	Description
désactivé	Détermine si le script ignore le composant et tous les composants enfants. Utilisez cette propriété pour tester, déboguer et modifier un script. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none">- Sélectionné. Le script ignore le composant.- Effacé. Le script applique le composant. La valeur par défaut est vide.
nom	Libellé descriptif pour le composant. Ce libellé s'affiche dans le fichier journal et la vue Événements . Utilisez la propriété nom pour identifier le composant ayant provoqué l'événement.
remarque	Commentaire défini par l'utilisateur, qui décrit l'objectif ou l'action du composant.

Dans l'exemple suivant, `<segment1>` est un segment répétitif, précédé par un en-tête et suivi d'un pied de page.

```
<stream>
  <headerline1>...</headerline1>
  <segments>
    <segment1>...</segment1>
    <segment1>...</segment1>
    <segment1>...</segment1>
  </segments>
  <footerline1>...</footerline1>
</stream>
```

Vous pouvez configurer un **XmlStreamer** qui renvoie l'en-tête suivant, qui n'est pas du XML bien structuré :

```
<stream>
  <headerline1>...</headerline1>
  <segments>
```

Si vous appliquez **AddHeaderModifier** à `<segment1>`, le modificateur préfixe chaque instance de `<segment1>` avec l'en-tête. Il ajoute des balises de fermeture pour garantir que le XML est bien structuré. Le résultat est le segment suivant :

```
<stream>
  <headerline1>...</headerline1>
  <segments>
    <segment1>...</segment1>
  </segments>
</stream>
```

Si vous appliquez **AddHeaderModifier** à un segment d'en-tête, le modificateur ajoute l'en-tête de l'élément parent au segment. N'appliquez pas **AddHeaderModifier** à l'en-tête initial de l'entrée **XmlStreamer**, car l'en-tête initial ne possède pas lui-même un élément parent.

AddStringModifier

Dans un **XmlStreamer**, le composant **AddStringModifier** ajoute des chaînes avant et après un segment.

Le tableau suivant décrit les propriétés du composant **AddStringModifier** :

Propriété	Description
désactivé	Détermine si le script ignore le composant et tous les composants enfants. Utilisez cette propriété pour tester, déboguer et modifier un script. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none">- Sélectionné. Le script ignore le composant.- Effacé. Le script applique le composant. La valeur par défaut est vide.
nom	Libellé descriptif pour le composant. Ce libellé s'affiche dans le fichier journal et la vue Événements . Utilisez la propriété nom pour identifier le composant ayant provoqué l'événement.
pre	Définit la chaîne avant le segment.
post	Définit la chaîne après le segment.
remarque	Commentaire défini par l'utilisateur, qui décrit l'objectif ou l'action du composant.

DoNothingModifier

Dans un **XmlStreamer**, ce composant est un paramètre fictif. Il ne modifie pas le segment sur lequel il est appliqué.

WellFormedModifier

Dans **XmlStreamer**, le composant **WellFormedModifier** garantit qu'un segment est rédigé correctement en langage XML. Il peut ajouter des balises d'ouverture, de fermeture ou de racine nécessaires à la réalisation de cet objectif.

Le tableau suivant décrit les propriétés du composant **WellFormedModifier** :

Propriété	Description
désactivé	Détermine si le script ignore le composant et tous les composants enfants. Utilisez cette propriété pour tester, déboguer et modifier un script. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none">- Sélectionné. Le script ignore le composant.- Effacé. Le script applique le composant. La valeur par défaut est vide.
nom	Libellé descriptif pour le composant. Ce libellé s'affiche dans le fichier journal et la vue Événements . Utilisez la propriété nom pour identifier le composant ayant provoqué l'événement.

Propriété	Description
new_root_element	Définit l'élément racine. L'élément racine doit être un ancêtre du segment. Si vous n'affectez pas cette propriété, le modificateur n'ajoute pas d'élément racine.
remarque	Commentaire défini par l'utilisateur, qui décrit l'objectif ou l'action du composant.

Observez l'entrée XML suivante :

```
<stream>
  <headerline1>...</headerline1>
  <substreams>
    <substream>
      <subheaderline1>...</subheaderline1>
      <segments>
        <segment1>...</segment1>
        <segment1>...</segment1>
        <segment1>...</segment1>
      </segments>
      <subfooterline1>...</subfooterline1>
    </substream>
    <substream>...</substream>
    <substream>...</substream>
  </substreams>
  <footerline1>...</footerline1>
</stream>
```

Supposons que vous configuriez un élément **XmlStreamer** qui renvoie l'en-tête suivante, qui n'est pas une en-tête XML bien formée :

```
<substream>
  <subheaderline1>...</subheaderline1>
  <segments>
```

Si vous appliquez **WellFormedModifier** à cette en-tête, le modificateur ajoute les balises de fermeture. Le résultat est le segment bien formé suivant :

```
<substream>
  <subheaderline1>...</subheaderline1>
  <segments>
    </segments>
  </substream>
```

Supposons à présent que vous configuriez **WellFormedModifier** pour ajouter `<stream>` comme élément racine. Le résultat est :

```
<stream>
  <substreams>
    <substream>
      <subheaderline1>...</subheaderline1>
      <segments>
        </segments>
      <substream>
        <substreams>
          </substreams>
        </substream>
      </substreams>
    </substream>
  </substreams>
  <footerline1>...</footerline1>
</stream>
```

Comme vous pouvez le constater, le modificateur a ajouté les éléments `<stream>` et `<substreams>`, en conservant ainsi la structure du XML d'origine.

WriteSegment

Le composant **WriteSegment** copie un segment à un emplacement de sortie spécifié. Le composant n'altère pas le segment copié. Le composant **WriteSegment** est une option de la propriété **run_component** du composant **XmlSegment**.

Le tableau suivant décrit les propriétés du composant **WriteSegment** :

Propriété	Description
output	<p>Définit l'emplacement de la sortie. La propriété output a les options suivantes :</p> <ul style="list-style-type: none">- OutputDataHolder. Écrit dans une zone de stockage des données.- OutputFile. Écrit dans un fichier- OutputPort. Définit le nom d'un élément AdditionalOutputPort dans lequel les données sont écrites.- ResultFile. Écrit dans le fichier de résultats par défaut de la transformation.- StandardErrorLog. Ecrit dans le journal utilisateur. Pour plus d'informations, voir "Traitement des échecs" à la page 404 <p>Pour plus d'informations sur ces options, voir "Documentation de référence du sous-composant Action" à la page 334. La valeur par défaut est ResultFile.</p>

CHAPITRE 23

Validateurs, Notifications et Traitement des échecs

Ce chapitre comprend les rubriques suivantes :

- [Présentation des validateurs, des notificateurs et du traitement des échecs, 403](#)
- [Traitement des échecs, 404](#)
- [Validateurs, 407](#)
- [Propriétés standard des validateurs, 408](#)
- [Référence du composant validateur, 408](#)
- [Notifications, 423](#)
- [Documentation de référence du composant Notification, 424](#)

Présentation des validateurs, des notificateurs et du traitement des échecs

Lorsque vous concevez une transformation, vous devez prendre en compte les questions suivantes :

- Que se passe-t-il si les données d'entrée ne sont pas valides ? Par exemple, une date peut avoir un format incorrect, une chaîne peut être trop longue ou les enregistrements peuvent être hors de la séquence.
- Que se passe-t-il si des données manquent dans l'entrée ? Par exemple, une adresse peut ne pas comporter le numéro de la maison.
- Que se passe-t-il si l'entrée a une structure inhabituelle ? Par exemple, les enregistrements peuvent être hors de la séquence.

N'importe laquelle de ces conditions peut se produire en raison d'une erreur d'entrée. Dans ce cas, elles peuvent entraîner des erreurs de transformations et des échecs.

Les conditions peuvent également se produire dans des circonstances normales. Par exemple, un protocole d'entrée peut autoriser l'absence de certains champs.

Vous pouvez incorporer des fonctionnalités de transformation qui détectent de telles conditions et prennent les actions appropriées. Les approches suivantes font partie des actions possibles :

- Échec de la transformation et génération d'aucune sortie.
- Échouez une partie de la transformation, annulez sa sortie, mais autorisez la transformation à générer la sortie pour d'autres parties des données.

- Continuez la transformation en entier, mais écrivez un message dans le journal utilisateur.
- Continuez la transformation en entier, mais écrivez un message dans le fichier de résultat de la transformation.

Ce chapitre explique ce qui se passe en cas d'échec d'une transformation et la manière dont vous pouvez gérer les conditions d'échec. Il explique ensuite la manière dont vous pouvez détecter les erreurs de validation des données qui peuvent entraîner des échecs et la manière dont vous pouvez écrire des notifications sur de telles conditions dans la sortie.

Traitement des échecs

Un échec est un événement qui empêche un composant de traiter des données suivant la méthode attendue. Une ancre peut échouer si elle recherche un texte qui n'existe pas dans le document source. Un transformateur ou une action peut échouer si son entrée est vide ou a un type de données inapproprié.

Un échec peut être une occurrence absolument normale. Par exemple, un document source peut contenir une date optionnelle. Un analyseur contient une ancre **Contenu** qui traite la date, si elle existe. Si la date n'existe pas dans un document source spécifique, l'ancre **Contenu** échoue.

En configurant la transformation de façon appropriée, vous pouvez contrôler le résultat d'un échec. Dans l'exemple ci-dessus, vous pouvez configurer l'analyseur pour qu'il ignore les données manquantes et continue le traitement.

Le journal des événements affiche des avertissements à propos des échecs. En outre, vous pouvez configurer une transformation pour écrire un message d'échec dans un journal utilisateur.

Utilisation de la propriété `Optional` pour gérer les échecs

Vous pouvez utiliser la propriété `optional` pour contrôler le comportement d'une transformation quand un échec se produit.

L'échec a provoqué l'échec du parent

Si la propriété **facultatif** d'un composant n'est pas sélectionnée, un échec du composant provoque l'échec de son parent. Si le parent n'est également pas facultatif, son propre parent échoue, etc.

Imaginez par exemple qu'un **Analyseur** contienne un **Groupe** et que le **Groupe** contienne un **Marqueur**. Tous les composants sont non optionnels. Si le **Marqueur** n'existe pas dans le document source, le **Marqueur** échoue. Cela provoque l'échec du **Groupe** qui, à son tour, provoque l'échec de l'**Analyseur**.

Pour l'illustrer, nous pouvons représenter ces relations de la façon suivante :

```
Parser      //Échoué
  Group     //Échoué
    Marker  //Échoué
```

L'échec optionnel ne provoque pas l'échec du parent

Si la propriété **facultatif** d'un composant est sélectionnée, un échec du composant ne remonte pas au parent.

Dans l'exemple ci-dessus, supposez que le **Groupe** soit optionnel. Le **Marqueur** en échec provoque l'échec du **Groupe**, mais l'**Analyseur** n'échoue pas.

```
Parser      //Échoué
Group       //Échoué
Marker      //Échoué
```

Restauration

Si un composant échoue, ses effets sont annulés.

Imaginez par exemple qu'un **Groupe** contienne trois ancres **Contenu** non optionnelles, qui stockent des valeurs dans des zones de stockage des données. Si la troisième ancre **Contenu** échoue, le **Groupe** échoue. Le script annule les effets des deux premières ancres **Contenu**. Les données des deux premières ancres **Contenu** déjà stockées dans les zones de stockage des données sont supprimées.

Le retour en arrière ne s'applique qu'aux effets principaux d'une transformation, par exemple dans le cas d'un analyseur stockant des valeurs dans des zones de stockage des données ou d'un sérialiseur écrivant dans sa sortie. L'annulation ne s'applique pas aux effets secondaires. Dans l'exemple ci-dessus, si le **Groupe** contient une action **WriteValue** qui écrit une ligne dans un fichier texte de sortie, la ligne n'est pas supprimée.

Définition de la propriété Optional

Vous pouvez définir la propriété **optional** d'un composant des manières suivantes :

- Modifiez les propriétés avancées d'un composant du script.
- Cliquez sur le composant avec le bouton droit de la souris, puis cliquez sur **Rendre facultatif** ou sur **Rendre obligatoire**.

Composants dont une propriété Facultatif manque

Certains composants ne contiennent pas la propriété **facultatif**, car les composants n'échouent jamais, indépendamment de leur entrée.

L'action **Trier** en est un exemple. Si l'action **Trier** ne trouve aucune donnée à trier, elle se contente de ne rien faire. Il ne signale pas un échec.

Écriture d'un message d'échec dans le journal utilisateur

Vous pouvez configurer un composant de sorte qu'il consigne les échecs de sortie dans un journal défini par l'utilisateur. Par exemple, si une ancre ne réussit pas à trouver du texte dans le document source, elle peut écrire un message dans le journal utilisateur. Cela peut se produire même si l'ancre est définie comme optionnelle, de sorte que l'échec ne mette pas fin au traitement de la transformation.

Le journal utilisateur peut contenir des informations comme :

- Niveau de l'échec : information, avertissement ou erreur
- Nom du composant qui a échoué
- Description de l'échec
- Emplacement du composant ayant échoué dans le script
- Informations supplémentaires sur l'état de la transformation, comme les valeurs des zones de stockage des données.

Configuration de la sortie du journal utilisateur

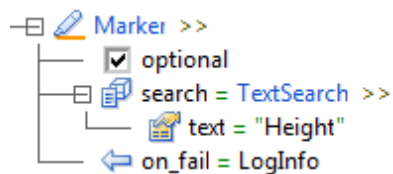
Pour définir la sortie du journal utilisateur, assignez la propriété `on_fail` des composants de transformation appropriés. Les composants suivants ont une propriété `on_fail` :

- Analyseurs et ancrs
- Sérialiseurs et ancrs de sérialisation
- Mappeurs et ancrs du mappeur

La propriété `on_fail` peut avoir les valeurs suivantes :

- `LogError`. Écrit un message d'erreur contenant la variable système `VarLastFailure` dans le journal utilisateur.
- `LogWarning`. Identique à `LogError`, mais affiche le message sous la forme d'un avertissement au lieu d'une erreur.
- `LogInfo`. Identique à `LogError`, mais affiche le message sous la forme d'une information plutôt que d'une erreur.
- `CustomLog`. Exécute un sérialiseur qui écrit un message personnalisé dans le journal utilisateur ou dans un autre emplacement. Pour plus d'informations, consultez la section [“CustomLog” à la page 309](#).
- `NotifyFailure`. Déclenche une notification.

L'exemple suivant illustre une ancre `Marqueur` avec une configuration `LogInfo` :



Si le `Marqueur` n'existe pas dans le document source, le système écrit l'entrée suivante dans le journal utilisateur :

```
*** INFO *** : Marker, [MyParser[11].Marker], Can't find Marker<optional>('Height').
```

Affichage du journal utilisateur

Le journal utilisateur est un fichier texte ASCII. Sur les plates-formes Windows, l'emplacement par défaut du journal utilisateur est le suivant :

```
c:\Informatica\DataTransformation\UserLogs
```

Sur les plates-formes Unix, l'emplacement par défaut est le suivant :

```
<RÉP_INSTALL>/UserLogs
```

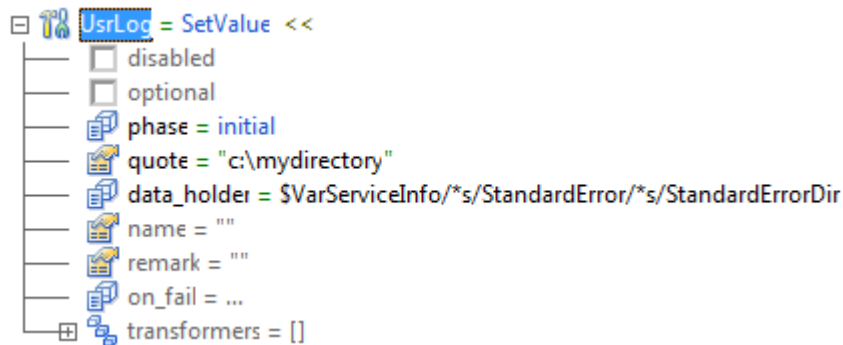
Par défaut, chaque exécution d'une transformation génère un journal utilisateur qui possède un nom unique :

```
<nom_service>+<chaîne_unique>.log
```

Une transformation peut définir l'emplacement user-log lors de l'exécution en utilisant des actions **SetValue** pour affecter les variables système suivantes. Définissez la propriété de phase de **SetValue** sur **initial**. Vérifiez que **SetValue** est exécuté avant tous les composants qui écrivent dans le journal utilisateur.

Variable	Description
VarServiceInfo > StandardError > <i>StandardErrorDir</i>	Chemin du répertoire du journal utilisateur.
VarServiceInfo > StandardError > <i>StandardErrorName</i>	Nom du fichier journal utilisateur.

Dans l'exemple suivant, une action **SetValue** définit le répertoire user-log sur `c:\mydirectory`.



Validateurs

Un composant de validateur confirme que son entrée est conforme à une condition. Vous pouvez utiliser des validateurs pour vérifier les longueurs de chaîne maximum ou minimum ou les valeurs numériques dans l'entrée, la conformité à des expressions, ou bien d'autres conditions. Vous pouvez appliquer plusieurs validateurs à la même entrée.

Si l'entrée n'est pas conforme à la condition, le validateur déclenche une notification. Un composant **NotificationHandler** peut traiter la notification. Par exemple, si vous utilisez des validateurs dans un analyseur, **NotificationHandler** peut insérer un message d'avertissement dans la sortie de l'analyseur. Pour plus d'informations, voir ["Notifications" à la page 423](#)

Vous pouvez insérer des validateurs dans emplacements comme la propriété **validators** d'une ancre **Contenu** ou d'une action **Carte**. Les validateurs vous permettent de signaler toute entrée non valide, sans forcément d'échec au niveau du **contenu** ou de la **carte**.

En plus des validateurs décrits dans ce chapitre, vous pouvez valider les données conformément à un ensemble de règles définies par l'utilisateur et générer un rapport de validation XML. Pour plus d'informations, voir ["ValidateValue" à la page 330](#)

Propriétés standard des validateurs

Le tableau suivant décrit les propriétés standard des validateurs :

Propriété	Description
allow_empty_value	Détermine si une entrée vide est acceptée comme valide. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none">- Sélectionné. Une entrée vide est valide.- Vide. Une entrée vide n'est pas valide. Par défaut, la valeur est vide.
disabled	Détermine si le script ignore le composant et tous les composants enfants. Utilisez cette propriété pour tester, déboguer et modifier un script. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none">- Sélectionné. Le script ignore le composant.- Effacé. Le script applique le composant. La valeur par défaut est vide.
name	Libellé descriptif pour le composant. Ce libellé s'affiche dans le fichier journal et la vue Événements . Utilisez la propriété nom pour identifier le composant ayant provoqué l'événement.
negation	Détermine si la condition de validation est infirmée. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none">- Sélectionné. Si la condition est vraie, l'entrée n'est pas valide. Si la condition est fausse, l'entrée est valide.- Vide. Si la condition est vraie, l'entrée est valide. Si la condition est fausse, l'entrée n'est pas valide. Par défaut, la valeur est vide.
notify	Définit le nom d'une notification. Si l'entrée n'est pas conforme à la condition de validation, le validateur déclenche la notification. Pour plus d'informations, voir "Notifications" à la page 423 . Par défaut, la valeur est vide.
optional	Détermine si la défaillance d'un composant entraîne l'échec du composant parent. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none">- Sélectionné. La défaillance du composant n'entraîne pas l'échec du composant parent.- Effacé. La défaillance du composant entraîne l'échec du composant parent. Par défaut, la valeur est vide. Pour plus d'informations sur l'échec des composants, consultez la section "Traitement des échecs" à la page 404 .
remark	Commentaire défini par l'utilisateur, qui décrit l'objectif ou l'action du composant.
transformers	Définit une liste de transformers qui s'appliquent à l'entrée. La condition de validation est appliquée au résultat des transformers. Les transformers n'ont qu'un effet temporaire sur les données à des fins de validation. L'entrée n'est pas définitivement altérée.

Référence du composant validateur

Les composants validateurs testent la conformité des données d'entrée aux règles définies.

AlternativeValidators

Le validateur **AlternativeValidators** contient un ensemble de validateurs imbriqués qui s'appliquent à l'entrée. Utilisez **AlternativeValidators** pour appliquer la logique OR à un ensemble de conditions de validation. Les données sont valides si elles satisfont une des conditions.

Le tableau suivant décrit les propriétés du validateur **AlternativeValidators** :

Propriété	Description
allow_empty_value	Détermine si une entrée vide est acceptée comme valide. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none">- Sélectionné. Une entrée vide est valide.- Vide. Une entrée vide n'est pas valide. Par défaut, la valeur est vide.
désactivé	Détermine si le script ignore le composant et tous les composants enfants. Utilisez cette propriété pour tester, déboguer et modifier un script. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none">- Sélectionné. Le script ignore le composant.- Effacé. Le script applique le composant. La valeur par défaut est vide.
nom	Libellé descriptif pour le composant. Ce libellé s'affiche dans le fichier journal et la vue Événements . Utilisez la propriété nom pour identifier le composant ayant provoqué l'événement.
négation	Détermine si la condition de validation est infirmée. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none">- Sélectionné. Si la condition est vraie, l'entrée n'est pas valide. Si la condition est fausse, l'entrée est valide.- Vide. Si la condition est vraie, l'entrée est valide. Si la condition est fausse, l'entrée n'est pas valide. Par défaut, la valeur est vide.
notifier	Définit le nom d'une notification. Si l'entrée n'est pas conforme à la condition de validation, le validateur déclenche la notification. Pour plus d'informations, voir "Notifications" à la page 423 . Par défaut, la valeur est vide.
facultatif	Détermine si la défaillance d'un composant entraîne l'échec du composant parent. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none">- Sélectionné. La défaillance du composant n'entraîne pas l'échec du composant parent.- Effacé. La défaillance du composant entraîne l'échec du composant parent. Par défaut, la valeur est vide. Pour plus d'informations sur l'échec des composants, consultez la section "Traitement des échecs" à la page 404 .
remarque	Commentaire défini par l'utilisateur, qui décrit l'objectif ou l'action du composant.
sélecteur	Détermine le critère de sélection d'un validateur parmi les validateurs imbriqués dans le composant AlternativeValidators . Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none">- ScriptOrder. L'analyseur teste les validateurs imbriqués dans l'ordre défini dans le script. Il accepte le premier validateur qui réussit. Si les validateurs échouent, l'entrée est non valide.- NameSwitch. L'analyseur recherche le validateur imbriqué dont la propriété nom est spécifiée dans la zone de stockage des données définie dans option_name. Il ignore les autres validateurs. Si le validateur nommé échoue, l'entrée est non valide. La valeur par défaut est ScriptOrder.
transformateurs	Définit une liste de transformers qui s'appliquent à l'entrée. La condition de validation est appliquée au résultat des transformers. Les transformers n'ont qu'un effet temporaire sur les données à des fins de validation. L'entrée n'est pas définitivement altérée.

EDIFACTValidation

Le validateur **EDIFACTValidation** teste si une chaîne source est un message EDIFACT valide.

Le tableau suivant décrit les propriétés du validateur **EDIFACTValidation** :

Propriété	Description
désactivé	Détermine si le script ignore le composant et tous les composants enfants. Utilisez cette propriété pour tester, déboguer et modifier un script. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none">- Sélectionné. Le script ignore le composant.- Effacé. Le script applique le composant. La valeur par défaut est vide.
activé	Détermine le paramètre pour param1 .
nom	Libellé descriptif pour le composant. Ce libellé s'affiche dans le fichier journal et la vue Événements . Utilisez la propriété nom pour identifier le composant ayant provoqué l'événement.
facultatif	Détermine si la défaillance d'un composant entraîne l'échec du composant parent. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none">- Sélectionné. La défaillance du composant n'entraîne pas l'échec du composant parent.- Effacé. La défaillance du composant entraîne l'échec du composant parent. Par défaut, la valeur est vide. Pour plus d'informations sur l'échec des composants, consultez la section "Traitement des échecs" à la page 404 .
param1	Détermine si l'entrée est facultative. param1 est nommé is_optional et n'a qu'une seule propriété, activé . activé a les options suivantes : <ul style="list-style-type: none">- Sélectionné. Les données d'entrée sont facultatives.- Effacé. Les données d'entrée sont obligatoires.
param2	Définit un type de données EDI. param2 est nommé input_type et n'a qu'une seule propriété, valeur . valeur est une chaîne entrée en dur ou une zone de stockage des données.
param3	Définit une plage de nombres entiers. param3 est nommé minmax_limits et n'a qu'une seule propriété, valeur . valeur est une chaîne entrée en dur ou une zone de stockage des données qui spécifie deux entiers séparés par un trait d'union.
param4	Définit une liste de valeurs. param4 est nommé énumérations et n'a qu'une seule propriété, valeur . valeur est une chaîne entrée en dur ou une zone de stockage des données qui spécifie une liste de chaînes ou d'entiers séparés par des virgules.
remarque	Commentaire défini par l'utilisateur, qui décrit l'objectif ou l'action du composant.
valeur	Définit une valeur pour param1 , param2 ou param3

Remarque: Ce composant est obsolète. L'éditeur IntelliScript l'affiche pour les scripts hérités. Ne l'utilisez pas dans de nouveaux scripts. Utilisez plutôt d'autres composants du validateur.

Énumération

Le validateur **Énumération** teste si une valeur est un membre d'un ensemble de valeurs.

Le tableau suivant décrit les propriétés du validateur **Énumération** :

Propriété	Description
allow_empty_value	Détermine si une entrée vide est acceptée comme valide. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none"> - Sélectionné. Une entrée vide est valide. - Vide. Une entrée vide n'est pas valide. Par défaut, la valeur est vide.
désactivé	Détermine si le script ignore le composant et tous les composants enfants. Utilisez cette propriété pour tester, déboguer et modifier un script. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none"> - Sélectionné. Le script ignore le composant. - Effacé. Le script applique le composant. La valeur par défaut est vide.
énumérations	Définit une liste de valeurs.
ignore_case	Détermine si la comparaison est sensible à la casse. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none"> - Sélectionné. La comparaison n'est pas sensible à la casse. - Effacé. La comparaison est sensible à la casse. La valeur par défaut est Effacé.
nom	Libellé descriptif pour le composant. Ce libellé s'affiche dans le fichier journal et la vue Événements . Utilisez la propriété nom pour identifier le composant ayant provoqué l'événement.
négation	Détermine si la condition de validation est infirmée. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none"> - Sélectionné. Si la condition est vraie, l'entrée n'est pas valide. Si la condition est fausse, l'entrée est valide. - Vide. Si la condition est vraie, l'entrée est valide. Si la condition est fausse, l'entrée n'est pas valide. Par défaut, la valeur est vide.
notifier	Définit le nom d'une notification. Si l'entrée n'est pas conforme à la condition de validation, le validateur déclenche la notification. Pour plus d'informations, voir "Notifications" à la page 423 Par défaut, la valeur est vide.
facultatif	Détermine si la défaillance d'un composant entraîne l'échec du composant parent. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none"> - Sélectionné. La défaillance du composant n'entraîne pas l'échec du composant parent. - Effacé. La défaillance du composant entraîne l'échec du composant parent. Par défaut, la valeur est vide. Pour plus d'informations sur l'échec des composants, consultez la section "Traitement des échecs" à la page 404 .
remarque	Commentaire défini par l'utilisateur, qui décrit l'objectif ou l'action du composant.
transformateurs	Définit une liste de transformers qui s'appliquent à l'entrée. La condition de validation est appliquée au résultat des transformers. Les transformers n'ont qu'un effet temporaire sur les données à des fins de validation. L'entrée n'est pas définitivement altérée.

LengthEquals

Le validateur **LengthEquals** teste si la longueur d'une chaîne est égale à une valeur spécifiée.

Le tableau suivant décrit les propriétés du validateur **LengthEquals** :

Propriété	Description
allow_empty_value	Détermine si une entrée vide est acceptée comme valide. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none"> - Sélectionné. Une entrée vide est valide. - Vide. Une entrée vide n'est pas valide. Par défaut, la valeur est vide.
désactivé	Détermine si le script ignore le composant et tous les composants enfants. Utilisez cette propriété pour tester, déboguer et modifier un script. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none"> - Sélectionné. Le script ignore le composant. - Effacé. Le script applique le composant. La valeur par défaut est vide.
longueur	Définit la longueur de la chaîne.
nom	Libellé descriptif pour le composant. Ce libellé s'affiche dans le fichier journal et la vue Événements . Utilisez la propriété nom pour identifier le composant ayant provoqué l'événement.
négation	Détermine si la condition de validation est infirmée. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none"> - Sélectionné. Si la condition est vraie, l'entrée n'est pas valide. Si la condition est fausse, l'entrée est valide. - Vide. Si la condition est vraie, l'entrée est valide. Si la condition est fausse, l'entrée n'est pas valide. Par défaut, la valeur est vide.
notifier	Définit le nom d'une notification. Si l'entrée n'est pas conforme à la condition de validation, le validateur déclenche la notification. Pour plus d'informations, voir "Notifications" à la page 423 . Par défaut, la valeur est vide.
facultatif	Détermine si la défaillance d'un composant entraîne l'échec du composant parent. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none"> - Sélectionné. La défaillance du composant n'entraîne pas l'échec du composant parent. - Effacé. La défaillance du composant entraîne l'échec du composant parent. Par défaut, la valeur est vide. Pour plus d'informations sur l'échec des composants, consultez la section "Traitement des échecs" à la page 404 .
remarque	Commentaire défini par l'utilisateur, qui décrit l'objectif ou l'action du composant.
transformateurs	Définit une liste de transformers qui s'appliquent à l'entrée. La condition de validation est appliquée au résultat des transformers. Les transformers n'ont qu'un effet temporaire sur les données à des fins de validation. L'entrée n'est pas définitivement altérée.

MaxLength

Le validateur **MaxLength** teste si la longueur d'une chaîne est inférieure ou égale à une valeur spécifiée.

Le tableau suivant décrit les propriétés du validateur **MaxLength** :

Propriété	Description
allow_empty_value	Détermine si une entrée vide est acceptée comme valide. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none">- Sélectionné. Une entrée vide est valide.- Vide. Une entrée vide n'est pas valide. Par défaut, la valeur est vide.
désactivé	Détermine si le script ignore le composant et tous les composants enfants. Utilisez cette propriété pour tester, déboguer et modifier un script. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none">- Sélectionné. Le script ignore le composant.- Effacé. Le script applique le composant. La valeur par défaut est vide.
longueur	Définit la longueur maximum de la chaîne.
nom	Libellé descriptif pour le composant. Ce libellé s'affiche dans le fichier journal et la vue Événements . Utilisez la propriété nom pour identifier le composant ayant provoqué l'événement.
négation	Détermine si la condition de validation est infirmée. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none">- Sélectionné. Si la condition est vraie, l'entrée n'est pas valide. Si la condition est fausse, l'entrée est valide.- Vide. Si la condition est vraie, l'entrée est valide. Si la condition est fausse, l'entrée n'est pas valide. Par défaut, la valeur est vide.
notifier	Définit le nom d'une notification. Si l'entrée n'est pas conforme à la condition de validation, le validateur déclenche la notification. Pour plus d'informations, voir "Notifications" à la page 423 . Par défaut, la valeur est vide.
facultatif	Détermine si la défaillance d'un composant entraîne l'échec du composant parent. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none">- Sélectionné. La défaillance du composant n'entraîne pas l'échec du composant parent.- Effacé. La défaillance du composant entraîne l'échec du composant parent. Par défaut, la valeur est vide. Pour plus d'informations sur l'échec des composants, consultez la section "Traitement des échecs" à la page 404 .
remarque	Commentaire défini par l'utilisateur, qui décrit l'objectif ou l'action du composant.
transformateurs	Définit une liste de transformateurs qui s'appliquent à l'entrée. La condition de validation est appliquée au résultat des transformateurs. Les transformateurs n'ont qu'un effet temporaire sur les données à des fins de validation. L'entrée n'est pas définitivement altérée.

MaxNumber

Le validateur **MaxNumber** teste si un nombre est inférieur ou égal à une valeur spécifiée.

Le tableau suivant décrit les propriétés du validateur **MaxNumber** :

Propriété	Description
allow_empty_value	Détermine si une entrée vide est acceptée comme valide. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none">- Sélectionné. Une entrée vide est valide.- Vide. Une entrée vide n'est pas valide. Par défaut, la valeur est vide.
désactivé	Détermine si le script ignore le composant et tous les composants enfants. Utilisez cette propriété pour tester, déboguer et modifier un script. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none">- Sélectionné. Le script ignore le composant.- Effacé. Le script applique le composant. La valeur par défaut est vide.
nom	Libellé descriptif pour le composant. Ce libellé s'affiche dans le fichier journal et la vue Événements . Utilisez la propriété nom pour identifier le composant ayant provoqué l'événement.
négation	Détermine si la condition de validation est infirmée. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none">- Sélectionné. Si la condition est vraie, l'entrée n'est pas valide. Si la condition est fausse, l'entrée est valide.- Vide. Si la condition est vraie, l'entrée est valide. Si la condition est fausse, l'entrée n'est pas valide. Par défaut, la valeur est vide.
notifier	Définit le nom d'une notification. Si l'entrée n'est pas conforme à la condition de validation, le validateur déclenche la notification. Pour plus d'informations, voir "Notifications" à la page 423 . Par défaut, la valeur est vide.
facultatif	Détermine si la défaillance d'un composant entraîne l'échec du composant parent. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none">- Sélectionné. La défaillance du composant n'entraîne pas l'échec du composant parent.- Effacé. La défaillance du composant entraîne l'échec du composant parent. Par défaut, la valeur est vide. Pour plus d'informations sur l'échec des composants, consultez la section "Traitement des échecs" à la page 404 .
remarque	Commentaire défini par l'utilisateur, qui décrit l'objectif ou l'action du composant.
transformateurs	Définit une liste de transformers qui s'appliquent à l'entrée. La condition de validation est appliquée au résultat des transformers. Les transformers n'ont qu'un effet temporaire sur les données à des fins de validation. L'entrée n'est pas définitivement altérée.
valeur	Définit la valeur maximum du nombre.

MinLength

Le validateur **MinLength** teste si la longueur d'une chaîne est supérieure ou égale à une valeur spécifiée.

Le tableau suivant décrit les propriétés du validateur **MinLength** :

Propriété	Description
allow_empty_value	Détermine si une entrée vide est acceptée comme valide. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none">- Sélectionné. Une entrée vide est valide.- Vide. Une entrée vide n'est pas valide. Par défaut, la valeur est vide.
désactivé	Détermine si le script ignore le composant et tous les composants enfants. Utilisez cette propriété pour tester, déboguer et modifier un script. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none">- Sélectionné. Le script ignore le composant.- Effacé. Le script applique le composant. La valeur par défaut est vide.
longueur	Définit la longueur minimum de la chaîne.
nom	Libellé descriptif pour le composant. Ce libellé s'affiche dans le fichier journal et la vue Événements . Utilisez la propriété nom pour identifier le composant ayant provoqué l'événement.
négation	Détermine si la condition de validation est infirmée. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none">- Sélectionné. Si la condition est vraie, l'entrée n'est pas valide. Si la condition est fausse, l'entrée est valide.- Vide. Si la condition est vraie, l'entrée est valide. Si la condition est fausse, l'entrée n'est pas valide. Par défaut, la valeur est vide.
notifier	Définit le nom d'une notification. Si l'entrée n'est pas conforme à la condition de validation, le validateur déclenche la notification. Pour plus d'informations, voir "Notifications" à la page 423 . Par défaut, la valeur est vide.
facultatif	Détermine si la défaillance d'un composant entraîne l'échec du composant parent. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none">- Sélectionné. La défaillance du composant n'entraîne pas l'échec du composant parent.- Effacé. La défaillance du composant entraîne l'échec du composant parent. Par défaut, la valeur est vide. Pour plus d'informations sur l'échec des composants, consultez la section "Traitement des échecs" à la page 404 .
remarque	Commentaire défini par l'utilisateur, qui décrit l'objectif ou l'action du composant.
transformateurs	Définit une liste de transformers qui s'appliquent à l'entrée. La condition de validation est appliquée au résultat des transformers. Les transformers n'ont qu'un effet temporaire sur les données à des fins de validation. L'entrée n'est pas définitivement altérée.

MinNumber

Le validateur **MinNumber** teste si un nombre est supérieur ou égal à une valeur spécifiée.

Le tableau suivant décrit les propriétés du validateur **MinNumber** :

Propriété	Description
allow_empty_value	Détermine si une entrée vide est acceptée comme valide. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none">- Sélectionné. Une entrée vide est valide.- Vide. Une entrée vide n'est pas valide. Par défaut, la valeur est vide.
désactivé	Détermine si le script ignore le composant et tous les composants enfants. Utilisez cette propriété pour tester, déboguer et modifier un script. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none">- Sélectionné. Le script ignore le composant.- Effacé. Le script applique le composant. La valeur par défaut est vide.
nom	Libellé descriptif pour le composant. Ce libellé s'affiche dans le fichier journal et la vue Événements . Utilisez la propriété nom pour identifier le composant ayant provoqué l'événement.
négation	Détermine si la condition de validation est infirmée. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none">- Sélectionné. Si la condition est vraie, l'entrée n'est pas valide. Si la condition est fausse, l'entrée est valide.- Vide. Si la condition est vraie, l'entrée est valide. Si la condition est fausse, l'entrée n'est pas valide. Par défaut, la valeur est vide.
notifier	Définit le nom d'une notification. Si l'entrée n'est pas conforme à la condition de validation, le validateur déclenche la notification. Pour plus d'informations, voir "Notifications" à la page 423 . Par défaut, la valeur est vide.
facultatif	Détermine si la défaillance d'un composant entraîne l'échec du composant parent. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none">- Sélectionné. La défaillance du composant n'entraîne pas l'échec du composant parent.- Effacé. La défaillance du composant entraîne l'échec du composant parent. Par défaut, la valeur est vide. Pour plus d'informations sur l'échec des composants, consultez la section "Traitement des échecs" à la page 404 .
remarque	Commentaire défini par l'utilisateur, qui décrit l'objectif ou l'action du composant.
transformateurs	Définit une liste de transformers qui s'appliquent à l'entrée. La condition de validation est appliquée au résultat des transformers. Les transformers n'ont qu'un effet temporaire sur les données à des fins de validation. L'entrée n'est pas définitivement altérée.
valeur	Définit la valeur minimum du nombre.

NumberEquals

Le validateur **NumberEquals** teste si un nombre est égal à une valeur spécifiée.

Le tableau suivant décrit les propriétés du validateur **NumberEquals** :

Propriété	Description
allow_empty_value	Détermine si une entrée vide est acceptée comme valide. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none">- Sélectionné. Une entrée vide est valide.- Vide. Une entrée vide n'est pas valide. Par défaut, la valeur est vide.
désactivé	Détermine si le script ignore le composant et tous les composants enfants. Utilisez cette propriété pour tester, déboguer et modifier un script. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none">- Sélectionné. Le script ignore le composant.- Effacé. Le script applique le composant. La valeur par défaut est vide.
nom	Libellé descriptif pour le composant. Ce libellé s'affiche dans le fichier journal et la vue Événements . Utilisez la propriété nom pour identifier le composant ayant provoqué l'événement.
négation	Détermine si la condition de validation est infirmée. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none">- Sélectionné. Si la condition est vraie, l'entrée n'est pas valide. Si la condition est fausse, l'entrée est valide.- Vide. Si la condition est vraie, l'entrée est valide. Si la condition est fausse, l'entrée n'est pas valide. Par défaut, la valeur est vide.
notifier	Définit le nom d'une notification. Si l'entrée n'est pas conforme à la condition de validation, le validateur déclenche la notification. Pour plus d'informations, voir "Notifications" à la page 423 . Par défaut, la valeur est vide.
facultatif	Détermine si la défaillance d'un composant entraîne l'échec du composant parent. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none">- Sélectionné. La défaillance du composant n'entraîne pas l'échec du composant parent.- Effacé. La défaillance du composant entraîne l'échec du composant parent. Par défaut, la valeur est vide. Pour plus d'informations sur l'échec des composants, consultez la section "Traitement des échecs" à la page 404 .
remarque	Commentaire défini par l'utilisateur, qui décrit l'objectif ou l'action du composant.
transformateurs	Définit une liste de transformers qui s'appliquent à l'entrée. La condition de validation est appliquée au résultat des transformers. Les transformers n'ont qu'un effet temporaire sur les données à des fins de validation. L'entrée n'est pas définitivement altérée.
valeur	Définit la valeur du nombre.

ValidateByExpression

Le validateur **ValidateByExpression** évalue une expression JavaScript. Si l'expression est fausse, le validateur considère l'entrée comme valide.

Le tableau suivant décrit les propriétés du validateur **ValidateByExpression** :

Propriété	Description
allow_empty_value	Détermine si une entrée vide est acceptée comme valide. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none"> - Sélectionné. Une entrée vide est valide. - Vide. Une entrée vide n'est pas valide. Par défaut, la valeur est vide.
disabled	Détermine si le script ignore le composant et tous les composants enfants. Utilisez cette propriété pour tester, déboguer et modifier un script. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none"> - Sélectionné. Le script ignore le composant. - Effacé. Le script applique le composant. La valeur par défaut est vide.
expression	Définit une expression JavaScript. Utilisez \$0 pour l'entrée du validateur. Utilisez un symbole dollar (\$) et un nombre entier pour les autres zones de stockage des données définies dans params , en commençant par \$1. Par exemple, l'expression suivante vérifie si l'entrée contient la valeur Ron Lehrer : <pre>\$0 == "Ron Lehrer"</pre>
name	Libellé descriptif pour le composant. Ce libellé s'affiche dans le fichier journal et la vue Événements . Utilisez la propriété nom pour identifier le composant ayant provoqué l'événement.
negation	Détermine si la condition de validation est infirmée. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none"> - Sélectionné. Si la condition est vraie, l'entrée n'est pas valide. Si la condition est fausse, l'entrée est valide. - Vide. Si la condition est vraie, l'entrée est valide. Si la condition est fausse, l'entrée n'est pas valide. Par défaut, la valeur est vide.
notify	Définit le nom d'une notification. Si l'entrée n'est pas conforme à la condition de validation, le validateur déclenche la notification. Pour plus d'informations, voir "Notifications" à la page 423 . Par défaut, la valeur est vide.
optional	Détermine si la défaillance d'un composant entraîne l'échec du composant parent. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none"> - Sélectionné. La défaillance du composant n'entraîne pas l'échec du composant parent. - Effacé. La défaillance du composant entraîne l'échec du composant parent. Par défaut, la valeur est vide. Pour plus d'informations sur l'échec des composants, consultez la section "Traitement des échecs" à la page 404 .
params	Définit une liste de zones de stockage des données qui contiennent des paramètres à utiliser dans l'expression.
remark	Commentaire défini par l'utilisateur, qui décrit l'objectif ou l'action du composant.
transformers	Définit une liste de transformers qui s'appliquent à l'entrée. La condition de validation est appliquée au résultat des transformers. Les transformers n'ont qu'un effet temporaire sur les données à des fins de validation. L'entrée n'est pas définitivement altérée.

ValidateByPattern

Le validateur **ValidateByPattern** teste si une chaîne correspond à une expression régulière. Pour plus d'informations, voir ["RegularExpression" à la page 283](#)

Le tableau suivant décrit les propriétés du validateur **ValidateByPattern** :

Propriété	Description
allow_empty_value	Détermine si une entrée vide est acceptée comme valide. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none">- Sélectionné. Une entrée vide est valide.- Vide. Une entrée vide n'est pas valide. Par défaut, la valeur est vide.
disabled	Détermine si le script ignore le composant et tous les composants enfants. Utilisez cette propriété pour tester, déboguer et modifier un script. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none">- Sélectionné. Le script ignore le composant.- Effacé. Le script applique le composant. La valeur par défaut est vide.
expression	Définit une expression régulière.
name	Libellé descriptif pour le composant. Ce libellé s'affiche dans le fichier journal et la vue Événements . Utilisez la propriété nom pour identifier le composant ayant provoqué l'événement.
negation	Détermine si la condition de validation est infirmée. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none">- Sélectionné. Si la condition est vraie, l'entrée n'est pas valide. Si la condition est fausse, l'entrée est valide.- Vide. Si la condition est vraie, l'entrée est valide. Si la condition est fausse, l'entrée n'est pas valide. Par défaut, la valeur est vide.
notify	Définit le nom d'une notification. Si l'entrée n'est pas conforme à la condition de validation, le validateur déclenche la notification. Pour plus d'informations, voir "Notifications" à la page 423 Par défaut, la valeur est vide.
optional	Détermine si la défaillance d'un composant entraîne l'échec du composant parent. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none">- Sélectionné. La défaillance du composant n'entraîne pas l'échec du composant parent.- Effacé. La défaillance du composant entraîne l'échec du composant parent. Par défaut, la valeur est vide. Pour plus d'informations sur l'échec des composants, consultez la section "Traitement des échecs" à la page 404 .
remark	Commentaire défini par l'utilisateur, qui décrit l'objectif ou l'action du composant.
transformers	Définit une liste de transformers qui s'appliquent à l'entrée. La condition de validation est appliquée au résultat des transformers. Les transformers n'ont qu'un effet temporaire sur les données à des fins de validation. L'entrée n'est pas définitivement altérée.

ValidateByTransformer

Le validateur **ValidateByTransformer** applique une liste d'un ou plusieurs transformers à l'entrée. Si la liste de transformers échoue, le validateur considère l'entrée comme non valide.

Le tableau suivant décrit les propriétés du validateur **ValidateByTransformer** :

Propriété	Description
allow_empty_value	Détermine si une entrée vide est acceptée comme valide. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none">- Sélectionné. Une entrée vide est valide.- Vide. Une entrée vide n'est pas valide. Par défaut, la valeur est vide.
disabled	Détermine si le script ignore le composant et tous les composants enfants. Utilisez cette propriété pour tester, déboguer et modifier un script. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none">- Sélectionné. Le script ignore le composant.- Effacé. Le script applique le composant. La valeur par défaut est vide.
name	Libellé descriptif pour le composant. Ce libellé s'affiche dans le fichier journal et la vue Événements . Utilisez la propriété nom pour identifier le composant ayant provoqué l'événement.
negation	Détermine si la condition de validation est infirmée. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none">- Sélectionné. Si la condition est vraie, l'entrée n'est pas valide. Si la condition est fausse, l'entrée est valide.- Vide. Si la condition est vraie, l'entrée est valide. Si la condition est fausse, l'entrée n'est pas valide. Par défaut, la valeur est vide.
notify	Définit le nom d'une notification. Si l'entrée n'est pas conforme à la condition de validation, le validateur déclenche la notification. Pour plus d'informations, voir "Notifications" à la page 423 . Par défaut, la valeur est vide.
optional	Détermine si la défaillance d'un composant entraîne l'échec du composant parent. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none">- Sélectionné. La défaillance du composant n'entraîne pas l'échec du composant parent.- Effacé. La défaillance du composant entraîne l'échec du composant parent. Par défaut, la valeur est vide. Pour plus d'informations sur l'échec des composants, consultez la section "Traitement des échecs" à la page 404 .
remark	Commentaire défini par l'utilisateur, qui décrit l'objectif ou l'action du composant.
run_transformers	Définit une liste de transformateurs.
transformers	Définit une liste de transformateurs qui s'appliquent à l'entrée. La condition de validation est appliquée au résultat des transformateurs. Les transformateurs n'ont qu'un effet temporaire sur les données à des fins de validation. L'entrée n'est pas définitivement altérée.

ValidateByType

Le validateur **ValidateByType** teste si son entrée est conforme à un type de données spécifié.

Le tableau suivant décrit les propriétés du validateur **ValidateByType** :

Propriété	Description
allow_empty_value	Détermine si une entrée vide est acceptée comme valide. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none">- Sélectionné. Une entrée vide est valide.- Vide. Une entrée vide n'est pas valide. Par défaut, la valeur est vide.
disabled	Détermine si le script ignore le composant et tous les composants enfants. Utilisez cette propriété pour tester, déboguer et modifier un script. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none">- Sélectionné. Le script ignore le composant.- Effacé. Le script applique le composant. La valeur par défaut est vide.
name	Libellé descriptif pour le composant. Ce libellé s'affiche dans le fichier journal et la vue Événements . Utilisez la propriété nom pour identifier le composant ayant provoqué l'événement.
negation	Détermine si la condition de validation est infirmée. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none">- Sélectionné. Si la condition est vraie, l'entrée n'est pas valide. Si la condition est fausse, l'entrée est valide.- Vide. Si la condition est vraie, l'entrée est valide. Si la condition est fausse, l'entrée n'est pas valide. Par défaut, la valeur est vide.
notify	Définit le nom d'une notification. Si l'entrée n'est pas conforme à la condition de validation, le validateur déclenche la notification. Pour plus d'informations, voir "Notifications" à la page 423 . Par défaut, la valeur est vide.
optional	Détermine si la défaillance d'un composant entraîne l'échec du composant parent. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none">- Sélectionné. La défaillance du composant n'entraîne pas l'échec du composant parent.- Effacé. La défaillance du composant entraîne l'échec du composant parent. Par défaut, la valeur est vide. Pour plus d'informations sur l'échec des composants, consultez la section "Traitement des échecs" à la page 404 .
remark	Commentaire défini par l'utilisateur, qui décrit l'objectif ou l'action du composant.
transformers	Définit une liste de transformers qui s'appliquent à l'entrée. La condition de validation est appliquée au résultat des transformers. Les transformers n'ont qu'un effet temporaire sur les données à des fins de validation. L'entrée n'est pas définitivement altérée.
val_type	Définit un type de données. Sélectionnez un type de standard ou un type qui est défini dans les schémas du projet.

ValidateDate

Le validateur **ValidateDate** teste si une date est conforme à un format de date ICU spécifié, par exemple, `aaaa-MM-jj`. Pour plus d'informations, voir ["DateFormatICU" à la page 268](#)

Le tableau suivant décrit les propriétés du validateur **ValidateDate** :

Propriété	Description
allow_empty_value	Détermine si une entrée vide est acceptée comme valide. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none">- Sélectionné. Une entrée vide est valide.- Vide. Une entrée vide n'est pas valide. Par défaut, la valeur est vide.
disabled	Détermine si le script ignore le composant et tous les composants enfants. Utilisez cette propriété pour tester, déboguer et modifier un script. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none">- Sélectionné. Le script ignore le composant.- Effacé. Le script applique le composant. La valeur par défaut est vide.
format_string	Définit un format de date ICU.
name	Libellé descriptif pour le composant. Ce libellé s'affiche dans le fichier journal et la vue Événements . Utilisez la propriété nom pour identifier le composant ayant provoqué l'événement.
negation	Détermine si la condition de validation est infirmée. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none">- Sélectionné. Si la condition est vraie, l'entrée n'est pas valide. Si la condition est fausse, l'entrée est valide.- Vide. Si la condition est vraie, l'entrée est valide. Si la condition est fausse, l'entrée n'est pas valide. Par défaut, la valeur est vide.
notify	Définit le nom d'une notification. Si l'entrée n'est pas conforme à la condition de validation, le validateur déclenche la notification. Pour plus d'informations, voir "Notifications" à la page 423 . Par défaut, la valeur est vide.
optional	Détermine si la défaillance d'un composant entraîne l'échec du composant parent. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none">- Sélectionné. La défaillance du composant n'entraîne pas l'échec du composant parent.- Effacé. La défaillance du composant entraîne l'échec du composant parent. Par défaut, la valeur est vide. Pour plus d'informations sur l'échec des composants, consultez la section "Traitement des échecs" à la page 404 .
remark	Commentaire défini par l'utilisateur, qui décrit l'objectif ou l'action du composant.
transformers	Définit une liste de transformers qui s'appliquent à l'entrée. La condition de validation est appliquée au résultat des transformers. Les transformers n'ont qu'un effet temporaire sur les données à des fins de validation. L'entrée n'est pas définitivement altérée.

ValidatorPipeline

Le validateur **ValidatorPipeline** applique une liste de validateurs aux données. Si l'un des validateurs signale une invalidité, ou si un validateur marqué comme **optionnel** échoue, **ValidatorPipeline** déclenche une notification.

Utilisez **ValidatorPipeline** pour appliquer une logique AND à un ensemble de conditions de validation. Les données sont valides si elles remplissent toutes les conditions.

Le tableau suivant décrit les propriétés du validateur **ValidatorPipeline** :

Propriété	Description
allow_empty_value	Détermine si une entrée vide est acceptée comme valide. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none">- Sélectionné. Une entrée vide est valide.- Vide. Une entrée vide n'est pas valide. Par défaut, la valeur est vide.
disabled	Détermine si le script ignore le composant et tous les composants enfants. Utilisez cette propriété pour tester, déboguer et modifier un script. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none">- Sélectionné. Le script ignore le composant.- Effacé. Le script applique le composant. La valeur par défaut est vide.
name	Libellé descriptif pour le composant. Ce libellé s'affiche dans le fichier journal et la vue Événements . Utilisez la propriété nom pour identifier le composant ayant provoqué l'événement.
negation	Détermine si la condition de validation est infirmée. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none">- Sélectionné. Si la condition est vraie, l'entrée n'est pas valide. Si la condition est fausse, l'entrée est valide.- Vide. Si la condition est vraie, l'entrée est valide. Si la condition est fausse, l'entrée n'est pas valide. Par défaut, la valeur est vide.
notify	Définit le nom d'une notification. Si l'entrée n'est pas conforme à la condition de validation, le validateur déclenche la notification. Pour plus d'informations, voir "Notifications" à la page 423 . Par défaut, la valeur est vide.
optional	Détermine si la défaillance d'un composant entraîne l'échec du composant parent. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none">- Sélectionné. La défaillance du composant n'entraîne pas l'échec du composant parent.- Effacé. La défaillance du composant entraîne l'échec du composant parent. Par défaut, la valeur est vide. Pour plus d'informations sur l'échec des composants, consultez la section "Traitement des échecs" à la page 404 .
remark	Commentaire défini par l'utilisateur, qui décrit l'objectif ou l'action du composant.
transformers	Définit une liste de transformers qui s'appliquent à l'entrée. La condition de validation est appliquée au résultat des transformers. Les transformers n'ont qu'un effet temporaire sur les données à des fins de validation. L'entrée n'est pas définitivement altérée.

Notifications

Une notification est un signal qu'une condition s'est produite dans une transformation. Quand la condition se produit, une transformation déclenche la notification. Vous pouvez configurer des gestionnaires qui traitent les notifications.

Les exemples suivants illustrent des manières d'utiliser les notifications :

- Un validateur peut déclencher une notification. Un composant **NotificationHandler** peut écrire un message d'avertissement de validation dans le fichier de résultat de la transformation ou dans un journal.

- Une ancre **StructureDefinition** peut définir un ensemble de composants **NotificationHandler** pour qu'il traite les discordances entre les enregistrements d'entrée et la structure d'entrée requise. Si une discordance se produit, le **NotificationHandler** approprié écrit un message dans le fichier de résultat ou dans un journal.
- Une action **Notification** déclenche une notification dans n'importe quel emplacement d'une transformation. **NotificationHandler** peut écrire un message dans le fichier de résultat ou dans un journal.

Le tableau suivant décrit les types de notification :

Notification	Description
MandatoryStructureMissing	Un enregistrement obligatoire n'apparaît pas dans l'entrée.
MismatchIDs	L'enregistrement et les identifiants du sous-élément correspondent partiellement. Par exemple, il y a deux identifiants d'enregistrement et seulement un des deux correspond.
StructureBelowMinOccurs	Il y a moins d'enregistrements du sous-élément qui correspondent que le nombre défini dans minOccurs .
StructureExceedsMaxOccurs	Il y a plus d'enregistrements du sous-élément qui correspondent que le nombre défini dans maxOccurs .
StructureOutOfSequence	Les enregistrements correspondent aux sous-éléments, mais pas dans l'ordre requis. Par exemple, les sous-éléments définissent une séquence ABC, mais l'entrée contient ACB.
UnexpectedRecord	Les enregistrements correspondent aux sous-éléments, mais pas dans la hiérarchie requise. Par exemple, le sous-élément définit une séquence ABC et D définie dans un autre emplacement. L'entrée contient ABD.
UnrecognizedRecord	Aucun sous-élément ne correspond à un identifiant de l'enregistrement.
XsdValidationError	L'entrée ne correspond pas aux exigences du schéma.


Documentation de référence du composant Notification

Les composants de notification exécutent des actions quand un composant échoue.

Notification

Le composant définit le nom d'une notification. Configurez le composant au niveau global du script.

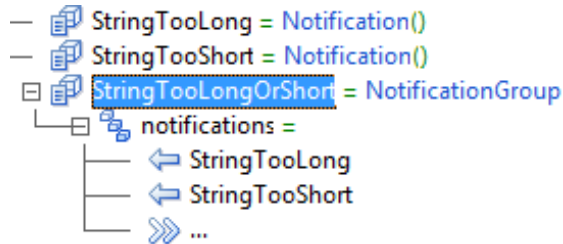
La figure suivante montre une notification appelée `StringTooLong` :

 `StringTooLong = Notification()`

NotificationGroup

Le composant définit un nom unique qui fait référence à un ensemble de noms de notification. Configurez le composant au niveau global du script.

L'exemple suivant montre un groupe appelé `StringTooLongOrShort` :



Vous pouvez configurer **NotificationHandler** pour qu'il traite `StringTooLongOrShort`. Si une transformation déclenche une notification `StringTooLong` ou `StringTooShort`, le gestionnaire traite la notification.

Le tableau suivant décrit les propriétés du composant **NotificationGroup**.

Propriété	Description
notifications	Définit une liste de notifications.

NotificationHandler

Le composant **NotificationHandler** définit une liste d'actions à prendre pour une notification spécifiée.

Insérez le composant **NotificationHandler** dans des emplacements comme la propriété **notifications** d'un **Groupe** ou **RepeatingGroup**. Dans le composant, vous pouvez insérer une action **WriteValue** qui stocke un message dans une zone de stockage des données.

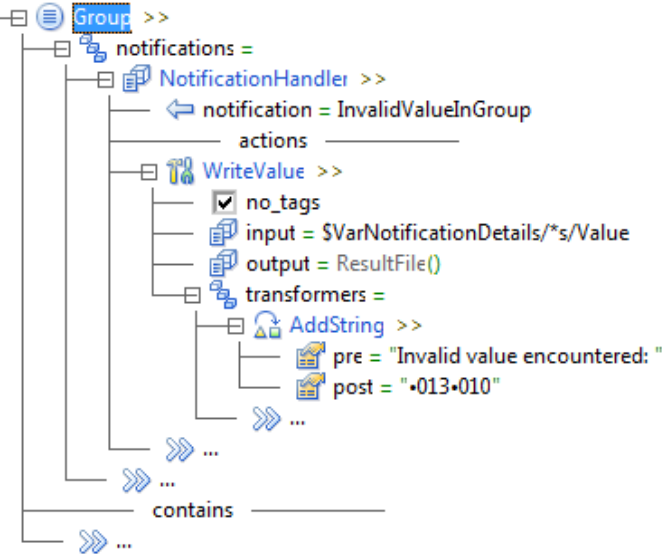
La variable `VarNotificationDetails` stocke des informations sur la notification qui a été le plus récemment déclenchée. Un **NotificationHandler** écrit les informations stockées dans `VarNotificationDetails` dans la sortie. Pour plus d'informations sur `VarNotificationDetails`, voir "[Variables système](#)" à la page 198.

Le tableau suivant décrit les propriétés du composant **NotificationHandler** :

Propriété	Description
désactivé	Détermine si le script ignore le composant et tous les composants enfants. Utilisez cette propriété pour tester, déboguer et modifier un script. Vous pouvez choisir l'une des options suivantes : <ul style="list-style-type: none">- Sélectionné. Le script ignore le composant.- Effacé. Le script applique le composant. La valeur par défaut est vide.
nom	Libellé descriptif pour le composant. Ce libellé s'affiche dans le fichier journal et la vue Événements . Utilisez la propriété nom pour identifier le composant ayant provoqué l'événement.
notification	Définit le nom d'une notification que NotificationHandler doit traiter. Sélectionnez une notification prédéfinie ou un nom de notification qui est défini dans un composant Notification ou NotificationGroup . Pour configurer un gestionnaire qui traite toute notification, sélectionnez anyNotification .
remarque	Commentaire défini par l'utilisateur, qui décrit l'objectif ou l'action du composant.

Propriété	Description
source	Définit une zone de stockage des données que vous pouvez utiliser pour l'entrée pour le NotificationHandler .
cible	Définit une zone de stockage des données pour la sortie du NotificationHandler .

La figure suivante montre une ancre **Groupe** qui est configurée avec un **NotificationHandler** :



Si un composant dans le **Groupe** déclenche une notification **InvalidValueInGroup**, le gestionnaire la traite. Le gestionnaire écrit la variable *VarNotificationDetails/Value*, avec une chaîne de texte, dans le fichier de résultat de la transformation.

NotifyFailure

NotifyFailure est une valeur possible pour la propriété **on_fail** d'ancres et d'autres composants. Si le composant échoue, **NotifyFailure** déclenche une notification.

Le tableau suivant décrit les propriétés du composant **NotifyFailure**.

Propriété	Description
notifier	Définit la notification à déclencher. Sélectionnez une notification prédéfinie ou un nom de notification qui est défini dans un composant Notification ou NotificationGroup
valeur	Définit la valeur de la variable <i>VarNotificationDetails/Value</i> . NotificationHandler peut inclure la valeur dans sa sortie.

CHAPITRE 24

Règles de validation

Ce chapitre comprend les rubriques suivantes :

- [Présentation de l'objet Règles de validation, 427](#)
- [Référence de l'élément Règles de validation, 428](#)
- [Modifier l'objet Règles de validation dans un éditeur externe, 434](#)
- [Créer un objet Règles de validation, 434](#)
- [Importer un service Data Transformation avec l'objet Règles de validation, 435](#)

Présentation de l'objet Règles de validation

Une transformation Processeur de données utilise l'objet Règles de validation pour vérifier les données d'entrée ou de sortie de la transformation. Vous pouvez utiliser un objet Règles de validation pour valider des données XML en fonction d'un ensemble de règles défini par l'utilisateur. Si les données violent les règles, l'action génère un rapport de validation XML. Lorsque vous créez un objet Règles de validation, vous pouvez fournir un exemple de fichier pour le tester.

L'objet Règles de validation vérifie les éléments d'entrée ou de sortie d'une transformation Processeur de données. Utilisez l'éditeur Règles de validation pour créer, définir, modifier et gérer les règles.

Après avoir créé un objet Règles de validation, vous devez ajouter les éléments Règles de validation. Chaque élément doit être défini dans l'éditeur. Ce dernier ajoute les éléments à la hiérarchie Règles de validation.

Le niveau racine de la hiérarchie contient une description de l'objet Règles de validation ainsi qu'une référence à l'exemple de fichier XML que vous pouvez utiliser pour déboguer l'objet Règles de validation. La hiérarchie Règles de validation contient les dossiers Recherche et Règles. L'élément Recherche définit une table de recherche contenant des codes et des traductions. Vous pouvez ajouter une ou plusieurs éléments Recherche au dossier Recherche.

L'élément Règle définit une règle de validation. Si la règle renvoie False, l'objet Règles de validation rapporte une erreur. Vous pouvez ajouter une ou plusieurs éléments Règle au dossier Règles.

Vous pouvez imbriquer les éléments suivants dans un élément Règle :

Élément Variable

Un élément Variable définit une variable dont le type de données est simple. Il contient une expression XPath qui évalue la valeur de la variable.

Élément Assertion

L'élément Assertion définit la logique d'une règle. Il contient une expression XPath. Si la valeur de l'expression est False, la règle rapporte une erreur de validation.

Élément Liste

L'élément Liste définit une variable complexe contenant une liste de valeurs.

Élément Trace

L'élément Trace ajoute la valeur d'une expression XPath dans le fichier des événements.

Après avoir créé un élément, vous devez définir ses attributs. Les attributs définissent la logique de cet élément.

Vous pouvez copier et coller les éléments dans l'éditeur Règles de validation. Vous pouvez également modifier l'objet Règles de validation dans un éditeur externe et ajouter, modifier ou supprimer des éléments au format XML.

Vous pouvez importer un service Data Transformation avec n'importe quel nombre de fichiers VRL. Vous pouvez copier l'intégralité ou une partie d'un fichier VRL dans l'éditeur Règles de validation. Vous pouvez ouvrir le fichier VRL dans un éditeur externe, puis copier les éléments qu'il contient et les coller dans la hiérarchie de l'éditeur Règles de validation.

Vous pouvez appeler l'objet Règles de validation à partir d'un composant Script de la transformation Processeur de données à l'aide de l'action **ValidateValue**.

Référence de l'élément Règles de validation

Le niveau supérieur d'une hiérarchie de règle de validation contient un élément Règle ou Recherche. Vous pouvez imbriquer les éléments Assertion, Liste, Trace et Variable dans un élément Règle.

Attributs de l'élément Assertion

L'élément Assertion définit la logique d'une règle. Il contient une expression XPath. Si la valeur de l'expression est False, la règle rapporte une erreur de validation.

Une règle doit contenir au moins un élément Assertion.

Le tableau suivant décrit les propriétés de l'élément Assertion :

Propriété	Description
Données supplémentaires	Facultatif. Expression XPath qui peut être évaluée et insérée dans le rapport d'erreur.
Code	Facultatif. Attribut de chaîne qui identifie l'élément Assertion. Si aucun code n'est spécifié, le code utilisé est celui de la règle parent.
Description	Facultatif. Description de l'élément Assertion. Si aucune description n'est spécifiée, la description utilisée est celle de la règle parent.
Emplacement	Facultatif. Valeur de chaîne pouvant être ajoutée à l'expression XPath équivalente au nœud sélectionné par la règle parent. L'expression est ensuite insérée dans le rapport d'erreur.
Description de la règle	Description en lecture seule de la règle dans laquelle l'élément est imbriqué.
XPath	Obligatoire. Expression booléenne XPath qui exprime la logique de la règle.

Attributs de l'élément Liste

L'élément Liste définit une variable complexe contenant une liste de valeurs.

Le tableau suivant décrit les propriétés de l'élément Liste :

Propriété	Description
Ajouter	Attribut booléen qui détermine ce qu'il se passe si le nom de la liste est le même que celui d'une liste existante. Si cette propriété est activée, les nouvelles valeurs sont ajoutées à la liste existante. Si elle est désactivée, la liste existante est supprimée et une nouvelle liste est créée. La valeur par défaut est activée.
Fonction	Facultatif. Expression XPath évaluée par rapport à chacun des nœuds sélectionnés. La valeur de l'expression est ajoutée à la liste.
Name	Obligatoire. Nom de la liste. Utilisez ce nom pour faire référence à la liste dans les expressions XPath d'éléments imbriqués dans la règle parent.
Description de la règle	Description en lecture seule de la règle dans laquelle l'élément est imbriqué.
Sélectionner	Obligatoire. Expression XPath qui sélectionne les nœuds permettant de calculer les éléments de la liste.

Attributs de l'élément Recherche

L'élément Recherche définit une table de recherche contenant des codes et des traductions.

Le tableau suivant décrit les propriétés de l'élément Recherche :

Propriété	Description
Fichier	Obligatoire. Fichier contenant la table de recherche.
Name	Obligatoire. Nom de la table de recherche.

Fichier de recherche

L'exemple suivant montre un exemple de fichier de recherche :

```
<LookupTable xmlns="http://www.Itemfield.com/Engine/V4/lookupTable" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <Entry key="a" value="1"/>
  <Entry key="b" value="2"/>
  <Entry key="c" value="3"/>
</LookupTable>
```

Le format de la table de recherche est identique à celui d'une table `XMLLookupTable` utilisée dans le composant `LookupTransformer`. Pour plus d'informations, voir ["LookupTransformer" à la page 280](#).

Vous pouvez utiliser la fonction `dt:lookup()` dans une expression XPath pour rechercher une valeur dans la table de recherche.

Attributs de l'élément Règle

L'élément Règle définit une règle de validation. Si la règle renvoie False, la règle de validation rapporte une erreur.

Le tableau suivant décrit les propriétés de l'élément Règle :

Propriété	Description
Code	Obligatoire. Identificateur de la règle. Doit être un seul mot.
Description	Obligatoire. Description de la règle de n'importe quelle longueur en texte libre.
Activé	Détermine si la règle est activée ou désactivée. La valeur par défaut est activée.
Exécuter toutes les assertions	Détermine si les assertions imbriquées dans la règle sont exécutées. La propriété est activée par défaut, de sorte que la règle traite toutes les assertions. Si cette propriété est désactivée, la règle arrête le traitement des assertions après l'échec de l'une d'entre elles.
Sélectionner	Obligatoire. Expression XPath qui sélectionne le nœud ou l'ensemble de nœuds auquel la règle s'applique.

Attributs de l'élément Trace

L'élément Trace ajoute la valeur d'une expression XPath dans le rapport d'erreur. Il contient une expression XPath.

Le tableau suivant décrit les propriétés de l'élément Trace :

Propriété	Description
Description	Obligatoire. Description de l'élément Trace.
Activé	Détermine si la trace est activée ou désactivée. La valeur par défaut est activée.
Description de la règle	Description en lecture seule de la règle dans laquelle l'élément est imbriqué.
XPath	Obligatoire. Expression XPath qui peut être évaluée et insérée dans le rapport d'erreur.

Attributs de l'élément Variable

Un élément Variable définit une variable dont le type de données est simple. Il contient une expression XPath. La valeur de l'élément Variable est la valeur de l'expression.

Le tableau suivant décrit les propriétés de l'élément Variable :

Propriété	Description
Nom	Obligatoire. Nom de la variable. Utilisez ce nom pour faire référence à la variable dans les expressions XPath d'éléments imbriqués dans la règle parent.
Description de la règle	Description en lecture seule de la règle dans laquelle l'élément est imbriqué.
XPath	Obligatoire. Expression XPath qui calcule ou fait référence à des nœuds auxquels la variable s'applique.

Éditeur XPath

Certains éléments contiennent des expressions XPath. Pour créer les expressions, vous devez saisir une expression XPath dans l'éditeur XPath.

Extensions XPath

Les éléments Règle, Assertion, Liste, Variable et Trace peuvent contenir des expressions XPath. Les fonctions suivantes peuvent faire partie de l'expression XPath.

L'objet Règles de validation définit les fonctions suivantes en tant qu'extensions de XPath 1.0. Les extensions appartiennent à l'espace de noms **dt**, dont vous trouverez la définition sur <http://validations.informatica.com>.

Fonction	Description
<code>dt:all-equal(param1[, param2, ...])</code>	Renvoie True si tous les éléments de la liste sont identiques. Les paramètres peuvent être des valeurs ou des listes simples.
<code>dt:check-uniqueness(value1[, value2, ...])</code>	Renvoie True si toutes les valeurs de la liste des paramètres sont uniques. Chaque valeur peut être : <ul style="list-style-type: none">- Une valeur simple- Une variable contenant une liste de chaînes- Une variable contenant une liste de nœuds dans laquelle chaque nœud est une valeur simple.
<code>dt:date-add(startDate, format, numberOfDays)</code>	Renvoie la date. Si la valeur numberOfDays est flottante , la fonction arrondit la valeur au nombre entier inférieur le plus proche.

Fonction	Description
<code>dt:date-diff(startDate, format, endDate, format)</code>	Calcule le nombre de jours compris entre deux dates.
<code>dt:date-format(date, inputFormat, outputFormat)</code>	Convertit le format de date de inputFormat à outputFormat .
<code>dt:date-valid(string-to-match, format)</code>	Renvoie True si la chaîne correspond au format de date spécifié.
<code>dt:deep-equal(element1, element2)</code>	Renvoie True si la valeur True est attribuée aux deux éléments suivants : <ul style="list-style-type: none"> - element1 et element2 disposent des mêmes attributs et des mêmes valeurs. - element1 et element2 disposent des mêmes éléments enfants (dans le même ordre) et tous les éléments enfants ont la même profondeur.
<code>dt:empty(xpath)</code>	Renvoie True si une expression XPath ne contient pas d'éléments enfants.
<code>dt:exist(xpath)</code>	Renvoie True s'il existe une expression XPath. Équivalent à count(xpath) > 0 .
<code>dt:is-sorted-lex(ascending, param1[, param2, ...])</code>	Renvoie True si les éléments de la liste sont triés de manière lexicographique. Le premier paramètre est une valeur booléenne qui définit le sens de tri : True pour le tri par ordre croissant ou False pour le tri par ordre décroissant.
<code>dt:last-day-of-month(date, format)</code>	Renvoie le dernier jour du mois sélectionné.
<code>dt:list-items(list, separator)</code>	Renvoie une chaîne de tous les éléments de la liste, séparés par le séparateur . Le séparateur par défaut est la virgule.
<code>dt:lookup(string, lookupName, default)</code>	Recherche une chaîne dans une table de recherche. Spécifiez la chaîne à rechercher, le nom de la table de recherche et une valeur par défaut à renvoyer si la chaîne n'existe pas dans la table.
<code>dt:min-lex(value1[, value2, ...])</code>	Trie la liste des paramètres d'entrée de manière lexicographique et renvoie le premier élément dans la liste triée. Chaque valeur peut être : <ul style="list-style-type: none"> - Une valeur simple - Une variable contenant une liste de chaînes - Une variable contenant une liste de nœuds dans laquelle chaque nœud est une valeur simple.
<code>dt:next-sequence()</code>	Renvoie un ensemble de nœuds contenant l'élément actif ainsi que tous ses frères suivants, jusqu'à un autre élément du même nom.

Fonction	Description
<code>dt:regex-match(string-to-match, regex)</code>	Renvoie True si une chaîne correspond à une expression régulière.
<code>dt:regex-replace(inputString, patternRegex, replacementString)</code>	Renvoie inputString avec toutes les instances de patternRegex remplacées par replacementString . Si aucun élément ne correspond à patternRegex , la valeur inputString renvoyée est renvoyée sans modification.
<code>dt:string-replace(string1, string2, string3)</code>	Remplace toutes les instances de string2 dans string1 par string3 .
<code>dt:unadjusted-calculation-period-dates(startDate, endDate, timeUnit, timeUnitMultiplier, lookupDate, dateFormat)</code>	Renvoie True si la valeur lookupDate est incluse dans la première période indiquée dans la plage spécifiée, dans laquelle la période est timeUnitMultiplier fois timeUnit , et la plage est comprise entre startDate et endDate . <ul style="list-style-type: none"> - La valeur timeUnitMultiplier est un nombre entier. - La valeur timeUnit est l'une des chaînes suivantes : <ul style="list-style-type: none"> - Day ou D - Week ou W - Month ou M - Year ou Y - La valeur lookupDate est une date comprise entre les valeurs startDate et endDate. - La valeur dateFormat définit les formats des valeurs lookupDate, startDate et endDate

Pour plus d'informations sur le format de date, consultez ["DateFormatICU" à la page 268](#).

Exemple : Utilisation de la fonction dt:next-sequence()

Vous pouvez utiliser la fonction `dt:next-sequence()` pour accéder aux données logiquement hiérarchiques non imbriquées dans l'élément actif.

Observez l'entrée suivante :

```
<Root>
  <A/>
  <B/>
  <C/>
  <A/>
</Root>
```

Le chemin XPath `/Root/A/dt:next-sequence()` renvoie l'ensemble de nœuds suivants :

```
<A/>
<B/>
<C/>
```

Dans l'exemple suivant, chaque élément `id` est associé à un ensemble d'éléments frères appelés `name`, `quantity` et `price` :

```
<items>
  <id>100</id>
  <name>Plate</name>
  <quantity>4</quantity>
  <price>10</price>
  <id>101</id>
  <name>Toaster</name>
```

```

    <quantity>6</quantity>
    <price>10</price>
    <id>102</id>
    <name>Knife</name>
    <quantity>10</quantity>
    <price>5</price>
  </items>

```

Les éléments `name`, `quantity` et `price` sont tous logiquement imbriqués dans l'élément `id`, même s'ils ne sont pas imbriqués physiquement.

Dans la règle suivante, l'élément `price*quantity` de chaque élément `id` doit être inférieur ou égal à 50 :

```

<rule code="sum1" select="/items/id" description="For each id, check that the total
price (price*quantity) does not exceed 50">
  <variable name="current">dt:next-sequence()</variable>
  <variable name="total">${current[3]}*${current[4]}</variable>
  <assert additionalData="$total"><![CDATA[ $total <= 50 ]]></assert>
</rule>

```

Pour l'élément `id`, la variable `$current` est un ensemble de nœuds comprenant la valeur suivante :

```

<id>100</id>
<name>Plate</name>
<quantity>4</quantity>
<price>10</price>

```

L'expression `$current[3]*$current[4]` évalue à $4*10 = 40$. La règle confirme que $40 < 50$.

Pour les autres éléments `id`, la règle évalue l'expression de manière similaire.

Modifier l'objet Règles de validation dans un éditeur externe

Vous pouvez sélectionner un objet Règle de validation à modifier dans un éditeur externe. L'éditeur affiche la hiérarchie Règles de validation au format XML avec des éléments imbriqués que vous avez créés. Vous pouvez copier, modifier ou supprimer les éléments au format XML.

Remarque: Si vous choisissez d'enregistrer les modifications lors de la modification d'un objet Règles de validation dans un éditeur externe, vous enregistrez toute la transformation, pas seulement les modifications apportées à l'objet Règles de validation.

Créer un objet Règles de validation

Vous pouvez utiliser l'éditeur Règles de validation pour créer, afficher et modifier un objet Règles de validation.

1. Dans la transformation Processeur de données, la vue **Objets** permet de créer un objet Règles de validation.
2. Cliquez dessus pour l'ouvrir.
3. Pour ajouter un élément, cliquez avec le bouton droit de la souris sur la règle de hiérarchie dans le volet de gauche de l'éditeur et sélectionnez Nouveau > <élément>, où <élément> remplace le type d'élément que vous voulez ajouter.

Un élément vide s'affiche.

4. Dans le volet de droite, définissez les attributs de l'élément.
5. Ajoutez des éléments et assignez des attributs à chaque élément selon les besoins.

Vous pouvez à chaque étape cliquer avec le bouton droit de la souris et effectuer les opérations suivantes :

- Ajouter un élément frère
- Ajouter un élément enfant
- Supprimer un élément
- Activer ou désactiver une règle
- Annuler ou rétablir les opérations

6. Enregistrer l'objet Règles de validation.

Importer un service Data Transformation avec l'objet Règles de validation

Vous pouvez importer un service Data Transformation contenant un objet Règles de validation depuis le référentiel du système de fichiers de la machine sur laquelle vous avez enregistré le service. Importez un fichier .cmw de service Data Transformation dans le référentiel modèle pour créer une transformation Processeur de données. L'outil Developer importe les règles de transformation et de validation avec le fichier .cmw.

1. Cliquez sur **Fichier > Importer**,

La boîte de dialogue **Importer** s'affiche.

2. Sélectionnez **InformaticaImporter le service Data Transformation** et cliquez sur **Suivant**.

La page **Importer le service Data Transformation** s'affiche.

3. Accédez au fichier .cmw du service que vous voulez importer.

L'outil Developer nomme la transformation d'après le nom du fichier de service. Vous pouvez modifier le nom.

4. Accédez à un emplacement dans le référentiel dans lequel vous souhaitez enregistrer la transformation, puis cliquez sur **Terminer**.

L'outil Developer importe les règles de transformation et de validation avec le fichier .cmw.

5. Pour modifier l'objet Règles de validation, double-cliquez dessus dans la vue **Explorateur d'objets**.

L'éditeur Règles de validation apparaît et affiche la hiérarchie de l'objet Règles de validation.

CHAPITRE 25

Composants de script personnalisés

Ce chapitre comprend les rubriques suivantes :

- [Présentation des composants de script personnalisés, 436](#)
- [Exemple de composant personnalisé, 436](#)
- [Propriétés du composant personnalisé, 437](#)
- [Développement d'un composant personnalisé, 437](#)
- [Configuration d'un composant personnalisé, 439](#)

Présentation des composants de script personnalisés

Lorsque vous concevez et configurez un script Transformation Processeur de données, vous pouvez utiliser un grand nombre de composants intégrés. Vous pouvez également programmer des composants personnalisés, tels que des processeurs de document ou des transformateurs et les insérer dans un script. Lorsque vous exportez la transformation Processeur de données en tant que service de transformation de données, le service exécute les composants personnalisés.

Vous pouvez implémenter les composants personnalisés dans Java. Pour plus d'informations sur les interfaces que vous devez implémenter, consultez la *Référence d'interface Java d'un composant externe*.

Exemple de composant personnalisé

Supposons que vous deviez analyser un format de données binaires propriétaire. Plutôt que d'analyser les données binaires directement, vous préférez convertir les données en une représentation texte plus facile à analyser.

Pour effectuer cela, vous pouvez programmer un processeur de document personnalisé que vous pouvez appeler `MyBinaryToText`. Le processeur peut notamment avoir les propriétés suivantes :

Propriété	Description
KeepLineBreaks	Une propriété booléenne. Quand la propriété a la valeur <code>True</code> , le processeur conserve les caractères de retour à la ligne dans les données binaires.
MaxLineLength	Une propriété de type entier. Spécifie la longueur maximum des lignes de texte de la sortie.
Ignorer	Une propriété de type chaîne. Demande au processeur d'ignorer les champs de données commençant par la chaîne indiquée.

Après avoir développé le processeur, vous pouvez l'installer et l'utiliser dans les scripts.

Propriétés du composant personnalisé

Les propriétés d'un composant personnalisé peuvent avoir les types de données booléen, entier, chaîne ou liste de chaînes. Vous pouvez attribuer une valeur de propriété constante ou le nom de détenteur de données qui contient la valeur.

Vous pouvez masquer certaines propriétés dans l'éditeur IntelliScript. Par exemple, un composant personnalisé peut prendre en charge quatre propriétés. Dans son fichier de configuration TGP, vous pouvez le configurer pour afficher seulement les deux premières propriétés. Le script envoie uniquement au composant les propriétés affichées. Le composant peut affecter ses propres des valeurs par défaut pour les propriétés masquées.

Le nombre maximal de propriétés dont un composant personnalisé peut disposer dépend du type de composant. Pour un composant processeur de documents, le nombre maximal de propriétés est de 4. Pour un composant transformateur, le nombre maximal de propriétés est de dix.

Développement d'un composant personnalisé

1. Créez une classe qui implémente une ou plusieurs des interfaces suivantes :

Type de composant	Type d'entrée	Interface
Processeur de documents	Fichier	<code>CMXFileProcessor</code>
Processeur de documents	Tampon	<code>CMXByteArrayProcessor</code>
Transformateur	Chaîne	<code>CMXStringTransformer</code>
Transformateur	Tampon	<code>CMXByteArrayTransform</code>

Pour plus d'informations sur ces interfaces, consultez la *documentation de référence sur l'interface Java des composants externes*.

2. Compilez le projet dans un fichier JAR.
3. Stockez le fichier JAR dans le sous-dossier `externLibs\user` du répertoire d'installation de chaque ordinateur sur lequel vous prévoyez d'utiliser le composant.
4. Créez un fichier de script qui définit le nom d'affichage du composant et ses propriétés. Stockez le fichier dans le sous-dossier `autoInclude\user` du répertoire d'installation.

Pour plus d'informations sur cette étape, consultez la section [“Configuration d'un composant personnalisé” à la page 439](#).

Vous pouvez ensuite utiliser le composant personnalisé dans les transformations.

Exemple d'interface Java

Prenez, par exemple, un processeur de document qui accepte l'entrée de fichier. Le processeur doit implémenter la classe `CMXFileProcessor` qui possède la méthode suivante :

```
public String process(
    CMXContext context,
    String in,
    String additionalFilesDir,
    CMXEventReporter reporter)
    throws Exception
```

Les paramètres ont la signification suivante :

Paramètre	Description
contexte	Paramètre d'entrée. Un objet contenant les propriétés que le script transmet au composant. La méthode parameters de l'objet renvoie un vecteur contenant les valeurs de la propriété.
in	Paramètre d'entrée. Le chemin d'accès complet du fichier sur lequel le composant fonctionne.
additionalFilesDir	Paramètre de sortie facultatif. Le chemin d'un répertoire temporaire où le composant écrit des fichiers. À la fin du traitement, le script supprime la totalité du contenu du répertoire.
reporteur	Paramètre d'entrée. Un objet fournissant la méthode de compte-rendu que le composant peut utiliser pour écrire des événements dans le journal des événements.

Exemple de composants Java personnalisés

Pour obtenir des exemples d'implémentation des composants personnalisés, consultez le sous-dossier suivant dans le répertoire d'installation :

```
samples\SDK\ExternalParameters\Java_SDK\Java
```

Le répertoire contient les échantillons suivants :

Échantillon	Description
FilePP.java	Un processeur de document acceptant l'entrée du fichier.
ByteArrayPP.java	Un processeur de document acceptant l'entrée de tampon.
StringTT.java	Un transformateur acceptant une entrée chaîne.
ByteArrayTT.java	Un transformateur acceptant l'entrée de tampon.

Configuration d'un composant personnalisé

Après avoir développé un composant personnalisé, vous devez préparer un fichier de script qui définit le composant. Vous ne pouvez pas préparer le fichier TGP dans l'éditeur IntelliScript. Au lieu de cela, vous devez le préparer dans un éditeur de texte.

Après avoir installé le composant et le fichier TGP, vous pouvez configurer le composant personnalisé dans l'éditeur IntelliScript.

1. Créez un fichier texte et enregistrez-le avec une extension *.tgp.

Remarque: Vous pouvez définir plus d'un composant externe dans un seul fichier TGP.

2. Pour chaque propriété prise en charge par votre composant externe, ajoutez des lignes comme celles qui suivent dans le fichier TGP :

```
profile <CustomPropertyName> ofPT <DataType>
{
    paramName = "<CustomPropertyName>" ;
}
```

<CustomPropertyName> est le nom d'une propriété que vous voulez afficher dans l'éditeur IntelliScript. <DataType> est le type de données de la propriété. Les types de données pris en charge sont NamedParamIntT pour une propriété de type entier, NamedParamBoolT pour une propriété booléenne, NamedParamStringT pour une propriété de type chaîne ou NamedParamListT pour une propriété qui est une liste de chaînes.

3. Pour chaque composant externe que vous désirez définir, ajoutez des lignes comme celles qui suivent dans le fichier TGP :

```
profile <ExternalComponentName> ofPT <ComponentType>
{
    jclass = "<ClassName>" ;
    param1 = <CustomPropertyName>() ;
    param2 = <CustomPropertyName2>() ;
}
```

<ExternalComponentName> est le nom du composant externe que vous voulez afficher dans l'éditeur IntelliScript. <ComponentType> a l'une des valeurs suivantes :

Pour	ComponentType
Un processeur de documents Java avec de 0 à 4 propriétés	ExternalJavaProcessorNoParamsT ExternalJavaProcessor1ParamsT ExternalJavaProcessor2ParamsT ...
Un transformateur Java avec de 0 à 10 propriétés	ExternalJavaTransformerNoParamsT ExternalJavaTransformer1ParamsT ExternalJavaTransformer2ParamsT ...

<ClassName> est le nom absolu de la classe Java. Sous Windows, <DllName> est le nom de la DLL, sans l'extension *.dll. Sous Linux ou UNIX, il s'agit du nom de l'objet partagé, sans le préfixe lib ou l'extension *.so.

<CustomPropertyName> et <CustomPropertyName2> sont les noms des propriétés que vous avez configurées à l'étape 2.

4. Enregistrez le fichier *.tgp.
5. Stockez le fichier dans le sous-répertoire DataTransformation\autoInclude\user du répertoire d'installation de chaque ordinateur sur lequel vous souhaitez utiliser le composant.

6. Si l'outil Developer est ouvert, fermez-le et rouvrez-le.
7. Si une erreur `autoInclude` apparaît, consultez le fichier TGP pour vérifier qu'il n'y a pas d'erreur de syntaxe ou d'incohérence dans le nommage et rouvrez l'outil Developer.
8. Ouvrez un projet et insérez le composant personnalisé dans le script. Le nom du composant personnalisé, que vous avez assigné à l'étape 3 ci-dessus, s'affiche dans la liste déroulante IntelliScript. L'éditeur IntelliScript affiche ses propriétés.

Exemple de scripts contenant des composants personnalisés

Vous pouvez trouver des exemples de fichiers de script contenant des composants personnalisés dans les sous-dossiers suivants du répertoire d'installation :

```
samples\SDK\ExternalParameters\Java_SDK\autoInclude  
samples\SDK\ExternalParameters\Cpp_SDK\autoInclude
```


INDEX

A

- AbsURL
 - composant [262](#)
- AcroForms
 - Traitement de formulaires PDF [168](#)
- actions
 - comparées aux transformateurs [298](#)
 - définition [298](#)
 - effets secondaires [297](#)
 - entrée et sortie [297](#)
 - propriétés d' [298](#), [408](#)
- AddEmptyTagsTransformer
 - composant [262](#)
- AddEventAction
 - composant [299](#)
- AddHeaderModifier
 - composant [399](#)
- AdditionalInputPort
 - composant [154](#)
- AdditionalOutputPort
 - composant [156](#)
- AddString
 - composant [263](#)
- AddStringModifier
 - composant [400](#)
- AggregateValues
 - composant [300](#)
- AllStructure
 - composant [251](#)
- AllStructureLocal
 - composant [251](#)
- AlternativeMappings
 - composant [359](#)
- Alternatives
 - composant [217](#)
- AlternativeSerializers
 - composant [343](#)
- AlternativeValidators
 - composant [409](#)
- analyse structurée
 - StructureDefinition [242](#)
- analyseur
 - Composants de script [150](#)
- Analyseur
 - composant [151](#)
 - description [23](#)
 - pour données COBOL [55](#)
- analyseur secondaire
 - ancre EmbeddedParser [225](#), [348](#)
- analyseurs
 - appeler un secondaire [348](#)
- Analyseurs
 - appeler un secondaire [225](#)
 - exécution du secondaire [323](#)
- analyseurs secondaires
 - sélection [218](#)
- ancre de groupe répétitif
 - mise en surbrillance de toutes les itérations [146](#)
- ancre, groupe répétitif
 - mise en surbrillance de toutes les itérations [146](#)
- ancres
 - définition [207](#)
 - emplacement dans IntelliScript [208](#)
 - étendue des complexes [216](#)
 - Marqueur et Contenu [205](#)
 - phase [211](#)
 - propriétés d' [209](#)
 - référence [217](#)
 - relation au XML [206](#)
 - relations aux délimiteurs [206](#)
 - sérialisation [336](#), [339](#)
 - utilisation des transformateurs [259](#)
- ancres contenu
 - surlignage dans un échantillon de document source [145](#)
- ancres de sérialisation
 - propriétés d' [341](#)
 - référence [343](#)
 - séquence d'opération [340](#)
- Ancres du mappeur
 - propriétés d' [356](#)
 - référence [359](#)
- ancres marqueur
 - surlignage dans un échantillon de document source [145](#)
- AppendListItems
 - composant [302](#)
- AppendValues
 - composant [303](#)
- arithmétique
 - calculs [305](#)
- Assertion
 - Élément Règle de validation [428](#)
- assistant de la transformation Processeur de données
 - description [49](#)
- attributeFormDefault
 - objet de schéma [130](#)
- AttributeSearch
 - composant [246](#)
- attribution
 - valeur de sortie [328](#)
- attributs
 - zones de stockage des données [192](#)
- attributs XML
 - zones de stockage des données [192](#)
- autoInclude
 - composants personnalisés [439](#)

B

- Base64Decoder
 - composant [264](#)
- Base64Encoder
 - composant [264](#)
- Bibliothèque
 - création dans une transformation Processeur de données [43](#)
 - Présentation [117](#)
 - propriétés de l'élément [118](#)
- BidiConvert
 - composant [265](#)
- BinaryFormat
 - composant [180](#)
- BIRT
 - générateur de rapport XmlToDocument [170](#), [171](#)
 - générateur de rapport XmlToDocument_372 [170](#)
 - générateur de rapport XmlToDocument_45 [171](#)
- boucle
 - Ancre RepeatingGroup [237](#)

C

- CalculateValue
 - composant [305](#)
- caractères
 - spéciaux dans IntelliScript [80](#)
- caractères ne figurant pas sur le clavier
 - entrer [143](#)
- caractères spéciaux
 - saisie dans IntelliScript [80](#)
- CDATADecode
 - composant [265](#)
- CDATAEncode
 - composant [266](#)
- chaînes
 - concaténation [302](#), [303](#)
- ChangeCase
 - composant [267](#)
- chemin
 - résolution relative [262](#)
- chercheur
 - composants [214](#)
- ChoiceStructure
 - composant [252](#)
- ChoiceStructureLocal
 - composant [253](#)
- clé
 - propriétés d' [376](#)
- Clé
 - composant [376](#)
- COBOL
 - fonctionnalités prises en charge [55](#)
 - importation d'une définition de données [55](#)
 - test de l'analyseur [56](#)
 - test du sérialiseur [57](#)
- codage
 - schéma XSD [193](#)
 - transformateur de page de code [272](#)
- combinaisons
 - de listes [307](#)
- CombineValues
 - composant [307](#)
- CommaDelimited
 - composant [186](#)
- ComplexSegment
 - composant [390](#)

- ComplexXmlSegment
 - composant [390](#)
- composant de démarrage
 - transformation Processeur de données [27](#)
- composant de démarrage, script
 - description [144](#)
- Composant de script
 - description [140](#)
 - Java personnalisé, développement [437](#)
- composant global
 - définition [141](#)
- composant local
 - définition [142](#)
- composant Notification [424](#)
- Composant NotificationHandler [425](#)
- Composant NotifyFailure [426](#)
- composant personnalisé
 - exemple [436](#)
 - Java, développement [437](#)
- Composant ValidateDate [421](#)
- composant, démarrage dans le script
 - description [144](#)
- composant, global
 - définition [141](#)
- composant, local
 - définition [142](#)
- composant, script
 - description [140](#)
 - Java personnalisé, développement [437](#)
- composants
 - dans IntelliScript [78](#)
- Composants de script
 - description [140](#)
 - noms [141](#)
 - propriétés, description [142](#)
- composants de script personnalisés
 - description [436](#)
 - propriétés [437](#)
- composants de script, personnalisés
 - description [436](#)
 - propriétés [437](#)
- composants globaux
 - définition [80](#)
- composants, script
 - description [140](#)
 - noms [141](#)
 - propriétés, description [142](#)
- composants, script personnalisé
 - description [436](#)
 - propriétés [437](#)
- concaténation
 - chaînes [302](#)
- condition
 - assurer dans le document source [314](#)
- Connecter
 - composant [253](#)
- ContentSerializer
 - composant [339](#), [344](#)
- Contenu
 - composant [219](#)
- contenu mixte
 - dans le schéma [193](#)
 - mappage aux [207](#)
 - zones de stockage des données dans [195](#)
- Conversion de PDF
 - configuration [175](#)
- correspondance de modèle
 - expressions régulières [284](#)

- couleurs
 - Surlignage dans un échantillon de document source [145](#)
- couper et coller
 - Composants de script [146](#)
- CreateGuid
 - composant [267](#)
- CreateList
 - composant [308](#)
- CreateUUID
 - composant [267](#)
- critères de recherche
 - pour les ancrs [211](#)
- CustomFormat
 - composant [181](#)
- CustomLog
 - composant [309](#)

D

- DateAddICU
 - composant [310](#)
- DateDiff
 - composant [311](#)
- DateDiffICU
 - composant [311](#)
- DateFormat
 - composant [269](#)
- DateFormatICU
 - composant [268](#)
- dates
 - format des [268](#)
- décalage
 - défini dynamiquement [248](#)
- décimal condensé [291](#)
- décimales condensées
 - nombres [276](#)
- décimales signées
 - nombres [276](#)
- DelimitedSections
 - composant [223](#)
- DelimitedSectionsSerializer
 - composant [345](#)
- DelimiterHierarchy
 - composant [187](#)
- Délimiteur
 - composant [187](#)
- délimiteurs
 - hiérarchie personnalisée [181](#)
 - relations aux ancrs [206](#)
- DocList
 - composant [159](#)
- document source, surlignage
 - description [145](#)
- document, source d'exemple
 - description [145](#)
- données
 - validation [407](#)
- données manquantes
 - traitement des échecs [404](#)
- données non valides
 - détection [403](#)
- DoNothingModifier
 - composant [400](#)
- Dos96HebToAscii
 - composant [270](#)
- DownloadFileToDataHolder
 - composant [312](#)

- dp:as_XML
 - Fonction du processeur de données [108](#)
- dp:get_id
 - Fonction du processeur de données [108](#)
- dp:lookup
 - Fonction du processeur de données [108](#)
- dp:output
 - Fonction du processeur de données [108](#)
- DumpValues
 - composant [313](#)
- DynamicTable
 - composant [271](#)
- dynamique
 - décalage [248](#)
 - search [250](#)

E

- EbcdicToAscii
 - composant [271](#)
- échantillon de document source, surlignage
 - description [145](#)
- échantillon de script
 - importation [149](#)
- échec
 - effet sur le parent [404](#)
- échec optionnel
 - effet sur le parent [404](#)
- échecs
 - gestion [403](#)
 - manipulation [404](#)
- échecs de validation
 - traitement [147](#)
- EDI
 - délimiteurs pour l'analyse [187](#)
- éditeur d'expression d'entrée
 - transformation Processeur de données [108](#)
- éditeur de configuration de tableau
 - PdfToTxt_4 [174](#)
- éditeur IntelliScript
 - composants et propriétés [78](#)
 - description [146](#)
- éditeur, IntelliScript
 - description [146](#)
- éditeurs
 - IntelliScript [76](#)
- effondrement de la propriété d'espace blanc
 - objet de schéma [128](#)
- elementFormDefault
 - objet de schéma [130](#)
- éléments
 - zones de stockage des données [192](#)
- éléments XML
 - zones de stockage des données [192](#)
- EmbeddedMapper
 - composant [360](#)
- EmbeddedParser
 - composant [225](#)
- EmbeddedSerializer
 - composant [348](#)
- EmbeddedStructure
 - composant [254](#)
- emplacements
 - marquage dans le document source [235](#)
- EnclosedGroup
 - composant [227](#)

- EnclosingDelimiters
 - composant [188](#)
- EncodeAsUrl
 - composant [272](#)
- Encodeur
 - composant [272](#)
- enregistrements
 - extraction dans StructureDefinition [229](#)
 - extraction et validation [242](#)
- EnsureCondition
 - composant [314](#)
- entre guillemets
 - groupe [227](#)
- entrée de caractères
 - non présent sur le clavier [143](#)
- entrée relationnelle dénormalisée
 - transformation Processeur de données [70](#)
- entrée relationnelle normalisée
 - transformation Processeur de données [69](#)
- Entrée relationnelle pivotée
 - Transformation Processeur de données [70](#)
- erreurs
 - traitement des échecs [404](#)
 - traitement des échecs de validation [147](#)
- espace de nom
 - modification de préfixe généré [125](#)
- Espaces de nom
 - objet de schéma [126](#)
- événement d'avertissement
 - Transformation Processeur de données [36](#)
- événement d'échec
 - Transformation Processeur de données [36](#)
- événement d'échec optionnel
 - Transformation Processeur de données [36](#)
- événement d'erreur fatale
 - Transformation Processeur de données [36](#)
- événement de notification
 - Transformation Processeur de données [36](#)
- événements
 - gestion [403](#)
 - Transformation Processeur de données [36](#)
- événements de notification
 - StructureDefinition [245](#)
- Excel
 - analyse en tant que XML [165](#), [166](#)
 - génération à partir d'un XML [172](#)
- ExcelToDataXml
 - composant [165](#)
- ExcelToXml
 - composant [166](#)
- ExcludelItems
 - composant [317](#)
- exécution maximale
 - objet de schéma [126](#)
- exécution minimale
 - objet de schéma [126](#)
- exemple de fichier
 - Parquet [61](#)
- exemple de fichier source
 - définition dans une transformation Processeur de données [41](#), [76](#)
- ExpandFrameSet
 - composant [167](#)
- expression de test
 - éditeur d'expression d'entrée XPath [108](#)
- expressions régulières
 - syntaxe [284](#)
- Expressions XPath
 - XMap [103](#)

- ExternalJavaPreProcessor
 - composant [167](#)
- extraction du contenu
 - Ancre Contenu [219](#)
- ExtractRecord
 - composant [229](#)

F

- fichier d'entrée d'exemple
 - Transformation Processeur de données [25](#)
- fichier exemple
 - Avro [50](#)
 - JSON [58](#)
 - XML [63](#)
- Fichier TGP
 - pour les composants personnalisés [439](#)
- fichiers de schéma
 - ajout à des objets de schéma [123](#)
 - configuration d'un éditeur par défaut [133](#)
 - édition [134](#)
 - suppression depuis des objets de schéma [123](#)
- Fichiers PDF
 - utilisation du processeur PdfToTxt_4 [174](#)
- fichiers XSD
 - schémas [192](#)
- FileSearch
 - composant [159](#)
- FindReplaceAnchor
 - composant [230](#)
- fois
 - format des [268](#)
- Fonctions du processeur de données
 - description [108](#)
- format
 - préprocesseurs [190](#)
- FormatNumber
 - composant [273](#)
- formulaires
 - traitement des PDF [168](#)
- fractionnement
 - fichiers [332](#)
- frameset
 - analyse HTML [167](#)
- FromFloat
 - composant [274](#)
- FromInteger
 - composant [275](#)
- FromPackDecimal
 - composant [276](#)
- FromSignedDecimal
 - composant [276](#)

G

- générateur de rapport
 - BIRT [170](#), [171](#)
- gestion
 - échecs [404](#)
- grille
 - XMap [86](#)
- groupe
 - exécution d'actions sur [232](#)
 - répétition [237](#)
- Groupe
 - composant [232](#)

- groupe de répétition
 - surlignage dans un échantillon de document source [145](#)
- groupe de substitution
 - objet de schéma [127](#)
- GroupMapping
 - composant [361](#)
- GroupSerializer
 - composant [349](#)

H

- Hébreu
 - conversion de page de code [277](#)
- hebrewBidi
 - composant [277](#)
- HebrewDosToWindows
 - composant [277](#)
- HebrewEBCDICOldCodeToWindows
 - composant [277](#)
- hebUniToAscii
 - composant [277](#)
- hebUtf8ToAscii
 - composant [277](#)
- hérité de la propriété
 - objet de schéma [129](#)
- hérité par la propriété
 - objet de schéma [129](#)
- heure
 - système [199](#)
- heure système
 - variable [199](#)
- HIPAA
 - validation [167](#)
- HIPAAValidator
 - composant [167](#)
- HL7
 - composant [188](#)
- HTML
 - supprimer des balises [286](#)
 - transformation des entités [277](#)
- HtmlEntitiesToASCII
 - composant [277](#)
- HtmlFormat
 - composant [182](#)
- HtmlProcessor
 - composant [191](#), [278](#)

I

- icônes
 - IntelliScript [81](#)
- identificateurs
 - IntelliScript [81](#)
- indépendance vis-à-vis de la plateforme:
 - Analyseurs [150](#)
- indexation
 - exemple [367](#)
 - zones de stockage des données à occurrences multiples [203](#)
- initialisation
 - variables [201](#)
- InjectFP
 - composant [278](#)
- InjectString
 - composant [279](#)
- InlineTable
 - composant [279](#)

- InputPort
 - composant [160](#)
- instruction de prédicat
 - Expressions XPath [104](#)
- Instruction Exécuter XMap
 - instruction de mappage [97](#)
- Instruction Groupe de répétition
 - Éditeur XMap [86](#)
 - Instruction Exécuter XMap
 - Éditeur XMap [86](#)
 - Instruction RunMapplet
 - Éditeur XMap [86](#)
- Instruction Option
 - Éditeur XMap [86](#)
- Instruction par défaut
 - Éditeur XMap [86](#)
- Instruction Routeur
 - Éditeur XMap [86](#)
- Instruction RunMapplet
 - instruction de mappage [98](#)
- instructions de codage
 - Transformation Processeur de données [31](#)
- Instructions de groupe
 - XMap [89](#)
- instructions de mappage
 - Éditeur XMap [86](#)
- Instructions de mappage
 - Couper et coller [103](#)
- instructions du Groupe de répétition
 - description [91](#)
- Instructions Option
 - Éditeur XMap [93](#)
- Instructions Routeur
 - Éditeur XMap [93](#)
- IntelliScript
 - définition d'ancres dans [209](#)
 - icônes utilisées dans [81](#)
 - modification [78](#)
 - restrictions de nommage [81](#)
- interface de ligne de commande
 - CM_console command [136](#)
- itérations
 - Ancre RepeatingGroup [237](#)

J

- Java
 - composant personnalisé, développement [437](#)
- JavaScript
 - extensions [316](#)
 - référence de syntaxe [315](#)
- JavaTransformer
 - composant [280](#)
- journal
 - définition [37](#)
- journal des événements
 - affichage [39](#)
 - événements personnalisés [299](#)
- journal des événements au moment de l'exécution
 - Transformation Processeur de données [38](#)
- journal des événements au moment de la conception
 - description et emplacement [38](#)
- journal des événements, au moment de la conception
 - description et emplacement [38](#)
- journal utilisateur
 - emplacement de définition de la variable [199](#)
 - transformation Processeur de données [39](#)

journal, événement au moment de la conception
description et emplacement [38](#)
journaux
écriture dans [403](#), [423](#)

L

Langage de règle de validation
VRL [330](#)
LearnByExample
composant [247](#)
LengthEquals
composant [411](#)
Liste
Élément Règle de validation [429](#)
listes
combinaison [307](#)
création [308](#)
de variables [202](#)
tri [329](#)
zones de stockage des données à occurrences multiples [202](#)
LocalFile
composant [160](#)
Localisateur
composant [378](#)
localisateurs
propriétés d' [376](#)
LocatorByKey
composant [379](#)
LocatorByOccurrence
composant [380](#)
longueur maximale
objet de schéma [126](#)
longueur minimale
objet de schéma [126](#)
LookupTransformer
composant [280](#)

M

Mappage
composant [88](#), [318](#)
mappeur
validation d'entrée [197](#)
Mappeur
appeler un secondaire [360](#)
composant [357](#)
création [354](#)
description [23](#)
mappeur secondaire
Ancre EmbeddedMapper [360](#)
mappeurs
utilisation de l'indexation [367](#)
Mappeurs
exécution dans un analyseur [321](#)
propriétés d' [356](#)
Mapplet
exécution du secondaire [323](#), [325](#)
référence [28](#)
MarkerStreamer
composant [392](#)
Marqueur
composant [235](#)
marqueurs
dans des répartiteurs [385](#)

masquer les propriétés
affichage [142](#)
MaxLength
composant [413](#)
MaxNumber
composant [413](#)
messages
notifications d'avertissements [423](#)
MinLength
composant [414](#)
MinNumber
composant [415](#)
mode Intelli
description [146](#)
mode script
description [146](#)
modèles
ouverture et fermeture de segment [383](#)
Mot
analyse comme XML [170](#)

N

NewlineSearch
composant [247](#)
nom du service
stockage de la variable [199](#)
nombres
formatage [273](#)
noms
Composants de script [141](#)
IntelliScript [81](#)
NormalizeClosingTags
composant [282](#)
NotificationGroup
composant [425](#)
notifications
déclenchement [320](#)
écriture de messages [423](#)
génération [403](#)
Notifier
composants [320](#)
NumberEquals
composant [416](#)

O

objet de schéma
propriétés des éléments avancés [127](#)
attributeFormDefault [130](#)
configuration d'un éditeur par défaut [133](#)
édition d'un fichier de schéma [132](#)
elementFormDefault [130](#)
éléments complexes [129](#)
emplacement du fichier [130](#)
Espaces de nom [126](#)
fichiers de schéma [123](#)
groupe de substitution [127](#)
hérité de la propriété [129](#)
hérité par la propriété [129](#)
importation [131](#)
présentation [123](#)
propriété abstraite [126](#)
propriété de bloc [127](#)
propriétés avancées des éléments complexes [129](#)
propriétés de l'attribut [130](#)

- objet de schéma (*a continué*)
 - propriétés de l'élément [126](#)
 - synchronisation [132](#)
 - type simple [128](#)
 - vue Présentation [124](#)
 - vue Schéma [125](#)
- Objet Règle de validation
 - création dans une transformation Processeur de données [43](#)
- Objet XMap
 - création dans une transformation Processeur de données [42](#)
- occurrence unique
 - zones de stockage des données [202](#)
- occurrences multiples
 - destruction des occurrences [203](#)
 - variables [202](#)
 - zones de stockage des données [202](#)
- OffsetSearch
 - composant [248](#)
- Option
 - composant [94](#)
- OutputDataHolder
 - composant [334](#)
- OutputFile
 - composant [334](#)
- OutputPort
 - composant [161](#)

P

- pages de code
 - schéma XSD [193](#)
- Pages de code
 - transformation [272](#)
- Par défaut
 - composant [96](#)
- paramètres
 - envoi à la transformation [201](#)
- Paramètres de codage
 - Transformation Processeur de données [29](#)
- paramètres de service
 - envoi à la transformation [201](#)
- paramètres du contrôle de sortie
 - Transformation Processeur de données [31](#)
- partage des entrées volumineuses
 - Répartiteur [382](#)
- PatternSearch
 - composant [248](#)
- PDF
 - Traitement de formulaires PDF [168](#)
- PdfFormToXml_1_00
 - composant [168](#)
- PdfToTxt_3_02
 - composant [168](#)
- PdfToTxt_4
 - composant [169](#)
 - utilisation [174](#)
- personnaliser la vue
 - Panneau schéma XMap [85](#)
- phase
 - de la recherche d'ancres [211](#)
- phases
 - imbriquées [211](#)
- point d'entrée, script
 - description [144](#)
- points de référence
 - ancre Marqueur [235](#)
 - autour des ancres [209](#), [392](#)
- points de référence (*a continué*)
 - de la zone de recherche [213](#)
- port d'entrée de tampon
 - Transformation Processeur de données [25](#)
- port d'entrée du fichier
 - Transformation Processeur de données [25](#)
- port de paramètre de service
 - Transformation Processeur de données [25](#)
- port de sortie de tampon
 - Transformation Processeur de données [27](#)
- port de sortie du fichier
 - Transformation Processeur de données [27](#)
- ports de paramètres de service
 - transformation Processeur de données [26](#)
- Positionnel
 - composant [188](#)
- post processeur
 - XmlToDocument [170](#), [171](#)
 - XmlToDocument_372 [170](#)
 - XmlToDocument_45 [171](#)
- PostScript
 - composant [189](#)
- PowerpointToTextML
 - composant [169](#)
- préfixe généré
 - modification pour l'espace de nom [125](#)
- préprocesseurs
 - définition [163](#)
 - document [163](#)
 - format [190](#)
- Prise en charge du format PDF
 - conversion de fichiers PDF [168](#), [169](#)
- problèmes liés aux performances
 - Variable VarLastFailure [199](#)
- ProcessByTransformers
 - composant [169](#)
- processeurs
 - document [163](#)
 - Java personnalisé [167](#)
 - référence [164](#)
 - utilisation des transformers comme [260](#)
- processeurs de document
 - définition [163](#)
 - exécution de plusieurs [170](#)
 - Java personnalisé [167](#)
 - référence [164](#)
- ProcessorPipeline
 - composant [170](#)
- propriété abstraite
 - objet de schéma [126](#)
- propriété d'énumération
 - objet de schéma [126](#)
- propriété de base
 - objet de schéma [128](#)
- propriété de bloc
 - objet de schéma [127](#)
- propriété de direction
 - des ancres [209](#)
- propriété de forme
 - objet de schéma [126](#)
- propriété de la phase
 - des ancres [209](#)
- propriété de la valeur fixe
 - objet de schéma [126](#)
- propriété de marquage
 - d'ancres [392](#)
 - des ancres [209](#)

- propriété disabled
 - sélection dans le menu [82](#)
- propriété facultatif
 - traitement des échecs [404](#)
- propriété nillible
 - objet de schéma [126](#)
- propriété optional
 - paramètre [405](#)
 - sélection dans le menu [82](#)
- propriété variété
 - objet de schéma [128](#)
- propriétés
 - composants de script personnalisés [437](#)
 - composants de scripts, description [142](#)
 - dans IntelliScript [78](#)
 - des actions [298, 408](#)
 - des ancrs [209](#)
 - des mappeurs [356](#)
 - des répartiteurs [389](#)
 - des sérialiseurs [341](#)
 - Transformateurs [261](#)
- propriétés avancées
 - description [142](#)
- propriétés de composant
 - valeurs, description [143](#)
- propriétés de exemple_source
 - Mappeur [357](#)
 - Sérialiseur [342](#)
- propriétés de l'attribut
 - objet de schéma [130](#)
- propriétés simples
 - description [142](#)
- propriétés, avancées
 - description [142](#)
- propriétés, composant
 - valeurs, description [143](#)
- propriétés, masquer
 - affichage [142](#)
- propriétés, simples
 - description [142](#)

R

- recherche
 - composants [246](#)
- Recherche
 - Élément Règle de validation [429](#)
- rechercher
 - direction de l'ancre [209](#)
- RecordStructure
 - composant [255](#)
- RecordStructureLocal
 - composant [256](#)
- récupération du contenu
 - Ancre Contenu [219](#)
- Redimensionner
 - composant [287](#)
- référence
 - analyseurs [151](#)
 - ancres [217](#)
 - ancres de sérialisation [343](#)
 - Ancres du mappeur [359](#)
 - délimiteurs [185](#)
 - formats [179](#)
 - indexation [376](#)
 - Mappeurs [357](#)
 - préprocesseurs de format [190](#)

- référence (*a continué*)
 - processeurs de document [164](#)
- regex
 - expressions régulières [284](#)
- Règle
 - Élément Règle de validation [430](#)
- Règle de validation
 - Éditeur XPath [431](#)
 - élément [428–431](#)
 - Élément Assertion [428](#)
 - Élément Liste [429](#)
 - Élément Recherche [429](#)
 - Élément Règle [430](#)
 - Élément Trace [430](#)
 - Élément Variable [431](#)
- Règles de validation
 - définition [427](#)
 - éditeur [427, 434](#)
 - importation du service Data Transformation [435](#)
 - modifier dans l'éditeur externe [434](#)
- RegularExpression
 - composant [283](#)
- RemoveMarginSpace
 - composant [285](#)
- RemoveRtfFormatting
 - composant [285](#)
- RemoveTags
 - composant [286](#)
- remplacement du texte
 - dans le document source [230](#)
- Remplacer
 - composant [286](#)
- répartiteur
 - segments complexes [383](#)
- Répartiteur
 - composant [395](#)
 - concaténation de l'en-tête [384](#)
 - création [386](#)
 - description [23](#)
 - JsonStreamer [391](#)
 - modèles d'ouverture et fermeture de segment [383](#)
 - partage des entrées volumineuses [382](#)
 - segment d'en-tête [383](#)
 - segment de pied de page [383](#)
 - segment répétitif [383](#)
 - sortie [385](#)
- répartiteur de texte
 - principe de l'opération [383](#)
- répartiteur XML
 - principe de l'opération [387](#)
- Répartiteurs
 - propriétés d' [389](#)
- RepeatingGroup
 - composant [237](#)
- RepeatingGroupMapping
 - composant [362](#)
- RepeatingGroupSerializer
 - composant [350](#)
- ResetVisitedPages
 - composant [320](#)
- restauration
 - après un échec [405](#)
- ResultFile
 - composant [335](#)
- ReverseTransformer
 - composant [288](#)
- RTF
 - composant [189](#)

- RtfFormat
 - composant [183](#)
- RtfProcessor
 - composant [191](#), [288](#)
- RtfToASCII
 - composant [289](#)
- RtfToTextML
 - composant [170](#)
- RunMapper
 - composant [321](#)
- RunMapplet
 - composant [323](#)
- RunParser
 - composant [323](#)
- RunPCWebService
 - composant [325](#)
- RunSerializer
 - composant [325](#)
- RunXMap
 - composant [327](#)

S

- schéma
 - Avro
 - schéma [50](#)
 - codage [193](#)
 - JSON
 - schéma [58](#)
 - Parquet
 - définition [61](#)
 - schéma [61](#)
 - pour données COBOL [55](#)
 - XML
 - schéma [63](#)
- schémas
 - fichiers XSD [192](#)
 - fonctionnalités non prises en charge [193](#)
 - représentation IntelliScript [195](#)
 - schémas inclus [193](#)
 - Transformation Processeur de données [28](#)
 - validation [196](#)
- Script
 - création dans une transformation Processeur de données [41](#), [76](#)
 - description [139](#)
 - éditeur, IntelliScript [146](#)
 - exemples [147](#)
 - importation d'échantillons [149](#)
 - structure [140](#)
- search
 - chaîne de recherche définie dynamiquement [250](#)
- segment d'en-tête
 - Répartiteur [383](#)
- segment de pied de page
 - Répartiteur [383](#)
- segment répétitif
 - Répartiteur [383](#)
- segments
 - traitement dans un répartiteur [383](#)
- segments complexes
 - répartiteur [383](#)
- SegmentSearch
 - composant [249](#)
- select-and-click
 - définition d'ancres [208](#)
- sélectionner une hiérarchie
 - transformation Processeur de données [66](#)

- sélectionner une hiérarchie (*a continué*)
 - Transformation Processeur de données [72](#)
- SequenceStructure
 - composant [256](#)
- SequenceStructureLocal
 - composant [257](#)
- sérialisation
 - mode [337](#)
 - utilisation des transformers dans [261](#)
- sérialiseur
 - validation d'entrée [197](#)
- Sérialiseur
 - composant [342](#)
 - contrôle de la génération automatique [336](#)
 - pour données COBOL [55](#)
- Sérialiseurs
 - dépannage généré automatiquement [338](#)
 - exécution dans un analyseur [325](#)
 - propriétés d' [341](#)
- Service Data Transformation
 - importation dans le référentiel modèle [45](#), [46](#)
 - importer plusieurs services [45](#)
- Service de transformation de données
 - chargement depuis la ligne de commande [136](#)
- service, transformation de données
 - chargement depuis la ligne de commande [136](#)
- ServiceLocation
 - variable [198](#)
- SetValue
 - composant [328](#)
- SGML
 - composant [189](#)
- SimpleSegment
 - composant [393](#)
- SimpleXmlSegment
 - composant [394](#)
- sortie PDF
 - post processeur XmlToDocument [170](#), [171](#)
 - XmlToDocument postprocessor_372 [170](#)
 - XmlToDocument postprocessor_45 [171](#)
- sortie relationnelle dénormalisée
 - Transformation Processeur de données [75](#)
- sortie relationnelle normalisée
 - Transformation Processeur de données [74](#)
- sortie relationnelle pivotée
 - Transformation Processeur de données [74](#)
- source
 - propriété [370](#)
- source d'exemple
 - affichage [146](#)
 - description [145](#)
 - Panneau schéma XMap [85](#)
 - paramètre [145](#)
- Source d'exemple
 - création dans une transformation Processeur de données [43](#)
- Source d'exemple LocalFile
 - description [145](#)
- Source d'exemple texte
 - description [145](#)
- SpaceDelimited
 - composant [189](#)
- StreamerVariable
 - composant [396](#)
- StringSerializer
 - composant [339](#)
- StructureDefinition
 - composant [242](#)
 - extraction des enregistrements [229](#)

- SubString
 - composant [289](#)
- surlignage dans un échantillon de document source
 - description [145](#)
- synchroniser avec l'éditeur
 - transformation Processeur de données [44](#)
- système
 - variables [198](#)

T

- TabDelimited
 - composant [190](#)
- tableaux
 - traitement des PDF [174](#)
- target
 - propriété [370](#), [374](#)
- tester une bibliothèque
 - transformation Processeur de données [120](#)
- Text
 - composant [161](#)
- texte de droite à gauche
 - inversion [265](#)
 - Inversion [277](#)
- texte manquant
 - recherche par Groupe optionnel [234](#)
- TextFormat
 - composant [183](#)
- TextML
 - Schéma XML [173](#)
- TextSearch
 - composant [249](#)
- ToFloat
 - composant [289](#)
- ToInteger
 - composant [290](#)
- ToPackDecimal
 - composant [291](#)
- Trace
 - Élément Règle de validation [430](#)
- traitement
 - échecs de validation [147](#)
- traitement des échecs
 - variables pour [199](#)
- Transformateur
 - description [23](#)
- transformateurs
 - comme préprocesseurs de document [260](#)
 - comparés aux actions [298](#)
 - définition [259](#)
 - Java personnalisé [280](#)
 - utilisation dans les ancres [259](#)
 - utilisation en tant que processeurs de document [169](#)
 - valeur par défaut [260](#)
- Transformateurs
 - propriétés d' [261](#)
- transformateurs par défaut
 - dans le format [260](#)
- Transformation de traitement de données
 - composant de démarrage [27](#)
- transformation Processeur de données
 - entrée relationnelle
 - transformation Processeur de données [67](#)
 - entrée relationnelle dénormalisée [70](#)
 - entrée relationnelle normalisée [69](#)
 - environnement non natif [48](#)
 - journaux utilisateurs [39](#)

- transformation Processeur de données (*a continué*)
 - mappage de XML sur des ports relationnels [66](#)
 - paramètres XMap [34](#)
 - Paramètres XML [34](#)
 - ports de paramètres de service [26](#)
 - test d'une bibliothèque [120](#)
 - Test dans la visionneuse de données [44](#)
- Transformation Processeur de données
 - Paramètres de codage [29](#)
 - création [40](#)
 - description [23](#), [24](#)
 - Entrée pivotée [70](#)
 - exportation en tant que service [45](#)
 - mappage de XML sur les ports [72](#)
 - paramètres [29](#)
 - Paramètres de traitement [33](#)
 - paramètres du contrôle de sortie [31](#)
 - ports [25](#)
 - ports d'entrée [25](#)
 - ports de sortie [27](#)
 - sortie orientée [74](#)
 - Sortie relationnelle
 - Transformation Processeur de données [73](#)
 - sortie relationnelle dénormalisée [75](#)
 - sortie relationnelle normalisée [74](#)
 - vues [24](#)
- TransformationStartTime
 - composant [292](#)
- TransformByParser
 - composant [292](#)
- TransformByProcessor
 - composant [293](#)
- TransformByService
 - composant [294](#)
- TransformerPipeline
 - composant [295](#)
- transformers
 - dans la sérialisation [261](#)
 - séquences de [260](#)
- tri
 - zones de stockage des données à plusieurs occurrences [329](#)
- Trier
 - composant [329](#)
- type complexe
 - propriétés avancées [129](#)
- type simple
 - objet de schéma [128](#)
- types
 - XSI [196](#)
- types d'événements
 - Transformation Processeur de données [36](#)
- types de données
 - recherche de [215](#)
- types de données de liste
 - attributs [203](#)
- types de données dérivés
 - Type XSI [196](#)
- types de liste
 - Mappage aux [215](#)
- types de membre
 - objet de schéma [128](#)
- Types XSI
 - mappage de zones de stockage des données [196](#)
- TypeSearch
 - composant [250](#)

U

- UNIX
 - conception d'analyseurs pour [150](#)
- URL
 - relatif en absolu [262](#)
- URL de la source d'exemple
 - description [145](#)

V

- valeurs, propriétés du composant
 - description [143](#)
- ValidateByExpression
 - composant [417](#)
- ValidateByPattern
 - composant [419](#)
- ValidateByTransformer
 - composant [419](#)
- ValidateByType
 - composant [420](#)
- validateurs
 - données d'entrée [407](#)
- ValidateValue
 - composant [330](#)
- validation
 - entrée XML [197](#)
 - sortie de l'analyseur XML [197](#)
 - XML [196](#)
- ValidatorPipeline
 - composant [422](#)
- VarCurrentPost
 - variable [199](#)
- VarCurrentURL
 - variable [199](#)
- Variable
 - composant [202](#)
 - Élément Règle de validation [431](#)
- variables
 - dans des répartiteurs [385](#)
 - défini par l'utilisateur [198](#)
 - Éditeur XMap [110](#)
 - initialisation [201](#)
 - listes [202](#)
 - mappage des ancras au [201](#), [207](#)
 - système [198](#)
 - utilisation dans les actions [201](#)
 - zones de stockage des données [192](#)
- VarLastFailure
 - variable [199](#)
- VarLinkURL
 - variable [198](#)
- VarRequestedURL
 - variable [199](#)
- VarServiceInfo
 - variable [199](#)
- VarSystem
 - heure système [199](#)
- VRL
 - Langage de règle de validation [330](#)
- Vue Aide du script
 - Transformation Processeur de données [24](#)
- Vue Aide IntelliScript
 - description [146](#)
- Vue Composant
 - Transformation Processeur de données [24](#)

- vue de type simple
 - objet de schéma [128](#)
- vue Événements
 - Transformation Processeur de données [37](#)
- vue Événements du processeur de données
 - affichage du journal des événements [39](#)
- Vue Événements du processeur de données
 - Transformation Processeur de données [37](#)
- Vue Événements, Processeur de données
 - affichage du journal des événements [39](#)
- vue Paramètres
 - Transformation Processeur de données [29](#)
- vue Ports
 - transformation Processeur de données [66](#), [72](#)
- Vue Références
 - Transformation Processeur de données [28](#)
- Vue Référentiel
 - Transformation Processeur de données [24](#)
- vue schéma
 - propriétés avancées de type simple [128](#)
- vue Schéma
 - objet de schéma [125](#)
- Vue Source Hex du processeur de données
 - description [24](#)
- Vue Source Hex, processeur de données
 - description [24](#)
- Vue Source hexadécimale
 - description [146](#)
- vue, Aide IntelliScript
 - description [146](#)
- vue, source hexadécimale
 - description [146](#)

W

- WellFormedModifier
 - composant [400](#)
- WordToXml
 - composant [170](#)
- WriteSegment
 - composant [401](#)
- WriteValue
 - composant [331](#)

X

- Xerces
 - validation XML [197](#)
- XMap
 - description [84](#)
 - exécution dans un analyseur [327](#)
 - exécution dans un mappeur [327](#)
 - exécution dans un sérialiseur [327](#)
 - Instructions de groupe [89](#)
 - Instructions Routeur [93](#)
 - Panneau schéma [85](#)
 - types d'instructions [86](#)
 - variables [110](#)
- XML
 - ajout de balises vides [262](#)
 - mappage des ancras au [206](#)
 - schémas [192](#)
 - transformation XSLT [296](#)
 - validation [197](#)
- XmlFormat
 - composant [184](#)

- XMLLookupTable
 - composant [295](#)
- XmlSegment
 - composant [397](#)
- XmlStreamer
 - composant [398](#)
- XmlToDocument
 - composant [170](#), [171](#)
- XmlToDocument_372
 - composant [170](#)
- XmlToDocument_45
 - composant [171](#)
- XmlToExcel
 - composant [172](#)
- XmlToXlsx
 - composant [172](#)
- XPath
 - notation modifiée [195](#)
- XSD
 - codage du schéma [193](#)
 - éditeurs [192](#)
 - fonctionnalités de schéma non prises en charge [193](#)
- XSLT
 - exécution de transformations [333](#)

- XSLTMap
 - composant [333](#)
- XSLTTransformer
 - composant [296](#)

Z

- zone de recherche
 - pour les ancrs [211](#)
 - réglage [213](#)
- zones de stockage des données
 - contenu mixte [195](#)
 - destruction des occurrences [203](#)
 - identification de la source et de la cible [370](#)
 - indexation à occurrences multiples [364](#)
 - occurrence unique ou occurrences multiples [202](#)
- zones de stockage des données à occurrences multiples
 - combinaison [307](#)
 - création de listes dans [308](#)
 - indexation [364](#)
 - mappage des ancrs au [207](#)
- zones de stockage des données à plusieurs occurrences
 - tri [329](#)