



Informatica® B2B Data Transformation
10.5.2

Benutzerhandbuch

Diese Software und die Dokumentation werden nur im Rahmen eines eigenen Lizenzvertrags zur Verfügung gestellt, der Beschränkungen für die Verwendung und Weitergabe enthält. Ohne ausdrückliche schriftliche Genehmigung der Informatica LLC darf kein Teil dieses Dokuments zu irgendeinem Zweck vervielfältigt oder übertragen werden, unabhängig davon, auf welche Art und Weise oder mit welchen Mitteln (elektronisch, mechanisch, durch Fotokopieren, Aufzeichnen usw.) dies geschieht.

Informatica, das Informatica-Logo, PowerCenter und PowerExchange sind Marken oder eingetragene Marken der Informatica LLC in den Vereinigten Staaten von Amerika und zahlreichen anderen Ländern der Welt. Eine aktuelle Liste der Informatica-Marken ist im Internet auf <https://www.informatica.com/trademarks.html> verfügbar. Alle weiteren Produkt- und Firmennamen sind möglicherweise Markennamen oder Warenzeichen der jeweiligen Eigentümer.

Gemäß Ihren Opt-out-Rechten überträgt die Software automatisch Informationen über die Computer- und Netzwerkumgebung, in der die Software bereitgestellt wird, sowie über die Datennutzung und Systemstatistiken der Bereitstellung an Informatica in den USA. Diese Übertragung gilt als Teil der Services/Dienste im Rahmen der Datenschutzrichtlinie von Informatica; die Verwendung und anderweitige Verarbeitung der Informationen durch Informatica erfolgen entsprechend der Datenschutzrichtlinie von Informatica, die hier zur Verfügung steht: <https://www.informatica.com/in/privacy-policy.html> Sie können die Sammlung von Nutzungsdaten im Administrator-Tool deaktivieren.

Den RECHTEN DER REGIERUNG DER VEREINIGTEN STAATEN unterliegende Programme, Software, Datenbanken und zugehörige Dokumentation und technische Daten, die an Kunden der Regierung der Vereinigten Staaten geliefert werden, sind "kommerzielle Computersoftware" oder "kommerzielle technische Daten" gemäß der anwendbaren Beschaffungsverordnung der Vereinigten Staaten (Federal Acquisition Regulation – FAR) und der ergänzenden Bestimmungen der spezifischen Behörde. Damit unterliegen die Nutzung, das Kopieren, die Offenlegung, das Modifizieren und die Anpassung den im anwendbaren Regierungsvertrag gemachten Einschränkungen und Lizenzbedingungen und, soweit im Rahmen der Bedingungen des Regierungsvertrags und der in FAR 52.227-19 aufgeführten Rechte anwendbar, der Lizenz für die kommerzielle Computersoftware.

Diese Software und die zugehörige Dokumentation enthalten proprietäre Informationen der Informatica LLC, werden unter einem Lizenzvertrag mit Einschränkungen hinsichtlich Verwendung und Veröffentlichung zur Verfügung gestellt und sind urheberrechtlich geschützt. Das Zurückentwickeln (Reverse Engineering) der Software ist untersagt. Ohne ausdrückliche schriftliche Genehmigung der Informatica LLC darf kein Teil dieses Dokuments zu irgendeinem Zweck vervielfältigt oder übertragen werden, unabhängig davon, auf welche Art und Weise oder mit welchen Mitteln (elektronisch, mechanisch, durch Fotokopieren, Aufzeichnen usw.) dies geschieht. Diese Software ist möglicherweise durch US-amerikanische und/oder internationale Patente und weitere angemeldete Patente geschützt.

Weitere Informationen über die Patente finden Sie unter <https://www.informatica.com/legal/patents.html>.

Die in dieser Dokumentation enthaltenen Informationen können jederzeit ohne vorherige Ankündigung geändert werden. Wenn Sie Probleme in dieser Dokumentation finden, melden Sie sie uns unter infa_documentation@Informatica.com.

Informatica-Produkte unterliegen einer Gewährleistung gemäß den Geschäftsbedingungen der Vereinbarungen, unter denen sie bereitgestellt werden. INFORMATICA STELLT DIE INFORMATIONEN IN DIESEM DOKUMENT OHNE MÄNGELGEWÄHR UND OHNE AUSDRÜCKLICHE ODER STILLSCHWEIGENDE GEWÄHRLEISTUNG JEDLICHER ART ZUR VERFÜGUNG. DIES GILT EINSCHLIESSLICH FÜR GEWÄHRLEISTUNGEN DER MARKTGÄNGIGKEIT, DER EIGNUNG FÜR EINEN BESTIMMTEN ZWECK UND GEWÄHRLEISTUNGEN ODER ZUSICHERUNGEN ÜBER DIE NICHTVERLETZUNG VON RECHTEN DRITTER.

Teile dieser Software und/oder Dokumentationen unterliegen dem Urheberrecht Dritter. Die erforderlichen Hinweise auf Drittanbieter sind im Lieferumfang des Produkts enthalten.

Inhalt

Einleitung	18
Informatica-Ressourcen.	18
Informatica Network.	18
Informatica-Wissensdatenbank.	18
Informatica-Dokumentation.	19
Informatica-Produktverfügbarkeitsmatrizen.	19
Informatica Velocity.	19
Informatica Marketplace.	19
Globaler Kundensupport von Informatica.	19
 Kapitel 1: Einführung in die Datenumwandlung.....	20
Data Transformation - Überblick.	20
Architektur des Data Transformation-Prozesses.	21
Data Transformation-Komponenten.	22
 Kapitel 2: Datenprozessor-Umwandlung.....	23
Datenprozessorumwandlung - Überblick.	23
Datenprozessorumwandlung - Ansichten.	24
Datenprozessorumwandlung - Ports.	25
Datenprozessorumwandlung - Eingabeports.	25
Datenprozessorumwandlung - Ausgabeports.	26
Pass-Through-Ports.	27
Startkomponente.	27
Verweise.	28
Datenprozessorumwandlung – Einstellungen.	28
Zeichencodierung.	29
Regeln und Richtlinien für die Zeichencodierung.	31
Ausgabeeinstellungen.	31
Verarbeitungseinstellungen.	32
XMap-Einstellungen.	33
Konfigurieren der XML-Ausgabe.	33
Ereignisse.	35
Ereignistypen.	35
Ansicht der Datenprozessor-Ereignisse.	36
Protokolle.	37
Entwicklungszeit-Ereignisprotokoll.	37
Laufzeit-Ereignisprotokoll.	38
Anzeigen eines Ereignisprotokolls in der Ansicht für Datenprozessor-Ereignisse.	38
Benutzerprotokoll.	38
Entwicklung der Datenprozessorumwandlung.	39

Erstellen der Datenprozessorumwandlung.	39
Auswählen der Schemaobjekte	40
Erstellen von Objekten in einer leeren Datenprozessorumwandlung.	40
Erstellen von Ports.	43
Testen der Umwandlung.	43
Export und Import von Datenprozessor-Umwandlungen.	44
Exportieren der Datenprozessorumwandlung als Dienst.	44
Importieren mehrerer Data Transformation-Dienste.	44
Importieren eines Data Transformation-Diensts	45
Exportieren eines Mappings mit einer Datenprozessorumwandlung nach PowerCenter.	45
Datenprozessorumwandlung Validierung.	46
Verwenden einer geschwindigkeitsverbesserten Datenumwandlungs-Engine für VRL-Validierungen.	47
Datenprozessorumwandlung in einer nicht nativen Umgebung.	47
Kapitel 3: Assistent für Eingabe- und Ausgabeformate.	48
Assistent für Eingabe- und Ausgabeformate – Übersicht.	48
Avro.	49
Avro-Eingabe und komplexer Datei-Reader.	49
Avro-Datenkomprimierung mit dem Snappy-Codec.	50
Konfigurieren einer Umwandlung mit Avro-Eingabe.	50
Konfigurieren einer Umwandlung mit Avro-Ausgabe.	53
COBOL-Verarbeitungsbibliothek.	54
Erstellen einer Umwandlung für COBOL.	54
COBOL-Datendefinitionen.	55
Testverfahren.	55
Bearbeiten einer Umwandlung für COBOL.	56
Optimieren der Verarbeitung einer umfangreichen COBOL-Datei in der Hadoop-Umgebung.	56
JSON.	57
JSON-Schemata.	57
JSON-Beispielschema.	58
Erstellen einer Umwandlung mit JSON.	59
Parquet.	60
Erstellen einer Umwandlung mit Parquet-Eingabe oder -Ausgabe.	60
Konfigurieren des komplexen Datei-Readers für Parquet-Eingabe.	61
Konfigurieren einer Umwandlung mit Parquet-Ausgabe.	61
XML.	62
Erstellen einer Umwandlung, die XML umwandelt.	63
Kapitel 4: Relationale Eingabe und Ausgabe.	64
Relationale Eingabe und Ausgabe – Übersicht.	64
Relationale Eingabe.	64
Konfiguration der relationalen Eingabeports.	65

Richtlinien für das Verknüpfen von Eingabeports.	66
Definieren Sie relationale Eingabeports in der Ansicht „Übersicht“.	66
Clustering_Key-Ports.	67
Normalisierte relationale Eingabe.	68
Pivotierte relationale Eingabe.	69
Denormalisierte relationale Eingabe.	69
Zuordnen von relationalen Ports zu hierarchischen Knoten.	70
Relationale Ausgabe.	71
Konfiguration von relationalen Eingabeports.	71
Definieren Sie relationale Ausgabeports in der Ansicht „Übersicht“.	72
Normalisierte relationale Ausgabe.	73
Pivotierte relationale Ausgabe.	73
Denormalisierte relationale Ausgabe.	74
Kapitel 5: Verwenden des IntelliScript-Editors.	75
IntelliScript-Editor – Übersicht.	75
Erstellen eines Skripts.	75
Öffnen eines IntelliScript-Editors.	76
IntelliScript und Daten-Viewer.	76
Finden von Ankern.	76
Komponenten und Eigenschaften.	77
Grundeigenschaften und erweiterte Eigenschaften.	77
Bearbeitungsverfahren.	77
Grundlegende Vorgehensweise bei der Bearbeitung.	77
Kopieren und Einfügen.	78
Ziehen und Ablegen.	78
Suchen und Ersetzen.	78
Einfügen von Komponenten in das IntelliScript.	78
Bearbeiten der Eigenschaften einer Komponente.	79
Einfügen von Tabulatoren, Zeilenumbrüchen und anderen Sonderzeichen.	79
Definieren einer globalen Komponente.	79
Anzeigen der Hilfe zu einer Komponente.	80
IntelliScript-Symbole.	80
Speichern des IntelliScripts.	81
Menüs des IntelliScript-Editors.	81
Kapitel 6: XMap.	84
Übersicht über XMap.	84
XMap-Schemata.	85
Mapping-Anweisungen.	86
Typen von Mapping-Anweisungen.	87
Map-Anweisungen.	88
Group-Anweisungen.	89

Anweisungen für Wiederholungsgruppen.	91
Router-Anweisungen.	93
Option-Anweisungen.	95
Default-Anweisungen.	96
Run XMap-Anweisungen.	97
RunMapplet-Anweisung.	98
MappletInput-Anweisung.	100
MappletOutput-Anweisung.	101
Erstellen einer XMap.	102
Verwenden des XMap-Editor-Gitters.	102
Mapping-Anweisungen erstellen.	103
Schnittstelle für das Gitter für Mapping-Anweisungen.	103
XPath-Ausdrücke.	104
Prädikate.	105
XPath-Ausdruckseditor.	108
Datenprozessor-Funktionen.	109
Beispiel für XPath-Ausdrücke.	110
Erstellen eines Ausdrucks.	111
XMap-Variablen.	111
Erstellen einer Variablen im XMap-Editor.	112
XMap-Beispiel.	112
XML-Eingabeschema - Beispiel.	112
XML-Ausgabeschema - Beispiel.	113
XML-Eingabedaten.	114
Eingabe- und Ausgabe-XML-Hierarchien.	115
Mapping-Anweisungen im Beispiel.	115
Gruppenanweisungsbeispiel.	117
Kapitel 7: Bibliotheken.	118
Bibliotheken - Übersicht.	118
Struktur von Bibliotheken.	119
Elementeigenschaften.	119
Bibliotheksverwaltung.	119
Bearbeiten von Bibliotheken mit dem Bibliothek-Editor.	120
Hinzufügen eines Elements mit dem Bibliothek-Editor.	121
Bearbeiten von Elementeigenschaften mit dem Bibliothek-Editor.	121
Testen einer Bibliothek.	121
Generieren der Bibliotheksobjekte.	122
Verwerfen der Bibliotheksobjekte.	122
Bearbeiten von Bibliotheken mit dem IntelliScript-Editor.	122
Kapitel 8: Schema-Objekt.	124
Schemaobjekt – Übersicht.	124

Schemadateien.	124
Ansicht Schemaobjekt - Übersicht.	125
Schemaobjekt-Ansicht „Schema“.	126
Namespace-Eigenschaften.	127
Elementeigenschaften.	127
Einfachtyp-Eigenschaften.	129
Komplextyp-Eigenschaften.	130
Attributeigenschaften.	131
Schemaobjekt-Ansicht „Erweitert“.	131
Erstellen eines Schemaobjekts.	132
Schema-Updates.	133
Schemasynchronisierung.	133
Bearbeitungen von Schemadateien.	134
Kapitel 9: Eingabeaufforderung.	137
Überblick über die Befehlszeilenschnittstelle.	137
CM_console	137
Kapitel 10: Skripte.	140
Übersicht über Skripts.	140
Skriptkomponenten.	141
Komponententypen.	141
Komponentennamen.	142
Hinzufügen einer globalen Komponente.	142
Hinzufügen einer lokalen Komponente.	143
Eigenschaften der Skriptkomponenten.	143
Einfache Eigenschaften.	143
Erweiterte Eigenschaften.	143
Eigenschaftswerte von Komponenten.	144
Skriptstartkomponenten.	145
Einrichten der Startkomponente mit dem IntelliScript-Editor.	145
Beispielquellen.	145
Hervorhebungen in der Beispielquelle.	146
Festlegen einer Beispielquelle im IntelliScript-Editor.	146
Anzeigen von Beispielquellen.	146
IntelliScript-Editor.	147
Validieren eines Skripts.	147
Muster-Skripte.	148
Beispielskript importieren.	149
Kapitel 11: Parser.	150
Parser - Überblick.	150
Plattformunabhängige Parser.	150

Zeilenvorschub-Marker.	150
Dateipfade.	150
Die Komponente Parser: Referenz.	151
Parser.	151
Kapitel 12: Skriptports.	154
Skriptports - Überblick.	154
Skript-Port-Komponente - Referenz.	154
AdditionalInputPort.	154
AdditionalOutputPort.	156
DocList.	159
FileSearch.	159
InputPort.	160
LocalFile.	160
OutputPort.	161
Text.	161
URL.	161
Kapitel 13: Dokumentprozessoren.	163
Überblick über Dokumentprozessoren.	163
Definieren eines Dokumentprozessors.	163
Anzeigen der Ausgabe von Dokumentprozessoren.	164
Dokumentprozessorkomponenten: Referenz.	164
AsnToXml.	164
ExcelToDataXml.	164
ExcelToXml.	165
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.	170
ProcessorPipeline.	170
RtfToTextML.	170
WordToXml.	170
XmlToDocument_372.	170
XmlToDocument_45.	171
XmlToExcel.	172
XmlToXlsx.	172
TextML-XML-Schema.	173

PdfToTxt_4-Editor zur Konfiguration von Tabellen.	174
Editor-Optionen.	175
Beispiel für eine PDF-Umwandlung.	176
Kapitel 14: Formate.	178
Übersicht über Formate.	178
Standardformateigenschaften.	179
Formatkomponenten: Referenz.	179
BinaryFormat.	180
CustomFormat.	181
HtmlFormat.	182
RtfFormat.	183
TextFormat.	183
XMLFormat.	184
Delimiter-Komponenten: Referenz.	185
CommaDelimited.	186
Delimiter.	187
DelimiterHierarchy.	187
EnclosingDelimiters.	188
HL7.	188
Positional.	188
PostScript.	189
RTF.	189
SGML.	189
SpaceDelimited.	189
TabDelimited.	190
Formatpräprozessor-Komponenten: Referenz.	190
HtmlProcessor.	190
RtfProcessor.	191
Kapitel 15: Datenbehälter.	192
Übersicht über Datenbehälter.	192
XML-Schemata.	192
Schemacodierung.	193
Einbezogene Schemadateien.	193
Namensräume.	193
Gemischter Inhalt.	193
Nicht unterstützte Schema-Funktionen.	193
Genauigkeit numerischer Daten.	195
Zuordnen von Ankern mit Hilfe eines Schemas.	195
IntelliScript-Darstellung der Datenbehälter.	195
Zuordnen gemischten Inhalts.	195
Zuordnen von XSI-Typen.	196

Erzeugen gültigen XML-Codes.	196
Aufgabe von Schemata beim Parsen.	197
Rolle von Schemata bei Serialisierung und Mapping.	197
Variablen.	198
Erstellen einer benutzerdefinierten Variable.	198
Systemvariablen.	198
Zuordnung von Ankern zu Variablen.	201
Variablen in Aktionen verwenden.	201
Initialisieren von Variablen während der Laufzeit.	201
Die Komponente Variable: Referenz.	202
Variable.	202
Mehrfachinstanz-Datenbehälter.	202
Attribute.	203
Indexierung.	203
Löschen von Instanzen.	204
Kapitel 16: Anker.	205
Überblick über Anker.	205
Die Anker Marker und Content.	205
Andere Ankertypen.	205
Gemeinsame Verwendung von Ankern und Delimitern.	206
Zuordnen von Content-Ankern zu Datenbehältern.	206
Zuordnen zu Variablen.	207
Zuordnen zu mehrfach vorkommenden Datenbehältern.	207
Zuordnen zu Elementen mit gemischtem Inhalt.	207
Definieren von Ankern.	208
Wo werden Anker definiert?.	208
Folge von Ankern.	208
Hinzufügen von Marker- oder Content-Ankern.	209
Definieren eines Ankers.	209
Standardeigenschaften von Ankern.	209
So sucht der Parser Anker.	211
Suchphasen.	211
Suchbereich und Suchkriterien.	212
Anpassen der Suchphase.	212
Anpassen des Suchbereichs.	213
Anpassen der Suchkriterien.	215
Eingrenzen der Suchkriterien mit Datentypen.	215
Anker mit geschachtelten Ankern.	217
Ankerkomponenten: Referenz.	217
Alternatives.	217
Content.	220
DelimitedSections.	224

EmbeddedParser.	227
EnclosedGroup.	228
ExtractRecord.	230
FindReplaceAnchor.	231
Group.	234
Marker.	237
RepeatingGroup.	239
StructureDefinition.	244
Suchkomponenten: Referenz.	248
AttributeSearch.	248
LearnByExample.	249
NewlineSearch.	250
OffsetSearch.	250
PatternSearch.	250
SegmentSearch.	251
TextSearch.	251
TypeSearch.	252
Die Unterkomponente Anker: Referenz.	253
AllStructure.	253
AllStructureLocal.	253
ChoiceStructure.	254
ChoiceStructureLocal.	255
Connect.	256
EmbeddedStructure.	256
RecordStructure.	257
RecordStructureLocal.	258
SequenceStructure.	259
SequenceStructureLocal.	259
Kapitel 17: Transformer.	261
Übersicht über Transformer.	261
Definieren von Transformern.	261
Verwenden von Transformern in Ankern.	261
Folgen von Transformern.	262
Standardtransformer.	262
Verwenden von Transformern als Dokumentprozessoren.	262
Verwenden von Transformern in Serialisierungsankern.	263
Verwenden von Transformern in Aktionen.	263
Standardeigenschaften von Transformern.	263
Transformer-Komponenten: Referenz.	264
AbsURL.	264
AddEmptyTagsTransformer.	264
AddString.	265

Base64Decode.	266
Base64Encode.	266
BidiConvert.	267
CDATADecode.	268
CDATAEncode.	268
ChangeCase.	269
CreateGuid.	270
CreateUUID.	270
DateFormatICU.	270
Dos96HebToAscii.	273
DynamicTable.	273
EbcdicToAscii.	273
EDIFACTValidation.	274
EncodeAsUrl.	274
Encoder.	275
FormatNumber.	276
FromFloat.	277
FromInteger.	278
FromPackDecimal.	278
FromSignedDecimal.	279
hebrewBidi.	280
HebrewDosToWindows.	280
HebrewEBCDICOldCodeToWindows.	280
hebUniToAscii.	280
hebUtf8ToAscii.	280
HtmlEntitiesToASCII.	280
HtmlProcessor.	281
InjectFP.	281
InjectString.	282
InlineTable.	283
JavaTransformer.	283
LookupTransformer.	284
NormalizeClosingTags.	286
RegularExpression.	286
RemoveMarginSpace.	288
RemoveRtfFormatting.	289
RemoveTags.	289
Replace.	290
Resize.	291
ReverseTransformer.	292
RtfProcessor.	292
RtfToASCII.	292

SubString.	293
ToFloat.	293
ToInteger.	294
ToPackDecimal.	295
TransformationStartTime.	295
TransformByParser.	296
TransformByProcessor.	297
TransformByService.	298
TransformerPipeline.	299
XMLLookupTable.	299
XSLTTransformer.	300
Kapitel 18: Aktionen.	301
Überblick über die Aktionen.	301
Funktionsweise von Aktionen.	301
Aktionen und Transformer im Vergleich.	302
Definieren von Aktionen.	302
Standardeigenschaften von Aktionen.	303
Die Komponente Aktion: Referenz.	303
AddEventAction.	303
AggregateValues.	304
AppendListItems.	306
AppendValues.	308
CalculateValue.	309
CombineValues.	311
CreateList.	313
CustomLog.	314
DateAddICU.	314
DateDiffICU.	315
DownloadFileToDataHolder.	316
DumpValues.	317
EnsureCondition.	318
ExcludeItems.	322
Map.	322
Benachrichtigen.	324
ResetVisitedPages.	325
RunMapper.	326
RunMapplet.	327
RunParser.	328
RunPCWebService.	329
RunSerializer.	330
RunXMap.	331
SetValue.	333

Sortieren.	334
ValidateValue.	335
WriteValue.	336
XSLTMap.	337
Aktions-Teilkomponenten: Referenz.	338
OutputDataHolder.	339
OutputFile.	339
ResultFile.	339
StandardErrorLog.	339
Kapitel 19: Serializer.	340
Übersicht über Serializer.	340
Steuern der Arbeitsweise des Befehls „Serializer erstellen“.	340
Fehlerbehebung bei automatisch erzeugten Serializern.	342
Erstellen eines Serializers durch Bearbeiten des Skripts.	343
Erstellen eines Serializers in der Aktion RunSerializer.	343
Serialisierungsanker.	343
Serialisierungsanker: Beispiel.	344
Reihenfolge von Serialisierungsankern.	344
Standardeigenschaften von Serializern.	345
Die Komponente Serializer: Referenz.	345
Serializer.	346
Serialisierungsanker-Komponenten: Referenz.	347
AlternativeSerializers.	347
ContentSerializer.	348
DelimitedSectionsSerializer.	349
EmbeddedSerializer.	352
GroupSerializer.	353
RepeatingGroupSerializer.	354
StringSerializer.	357
Kapitel 20: Mapper.	358
Erstellen eines Mappers.	358
In einen Mapper geschachtelte Komponenten.	358
Mapper: Beispiel.	359
XML-Quelle.	359
XML-Ausgabe.	359
Mapperkonfiguration.	360
Standardeigenschaften von Mappern.	360
Die Komponente Mapper: Referenz.	361
Mapper.	361
Mapper-Ankerkomponenten: Referenz.	363
AlternativeMappings.	363

EmbeddedMapper.	364
GroupMapping.	365
RepeatingGroupMapping.	366
Kapitel 21: Lokatoren, Schlüssel und Indexierung.	369
Überblick über Locator, Keys und Indexierung.	369
Lokatoren: Beispiel.	370
Eingabe und Ausgabe.	370
Falsche Lösung.	371
Richtige Lösung.	371
Indexierung nach Schlüssel: Beispiel.	371
Eingabe.	372
Ausgabe.	372
Konzept der Umwandlung.	372
Mapperkonfiguration.	373
Verwenden der Indexierung.	374
Die Eigenschaften „source“ und „target“.	374
Eigenschaft „source“.	375
Eigenschaft „target“.	378
Standardeigenschaften von Lokatoren und Schlüsseln.	381
Die Komponenten Locator und Key: Referenz.	381
Schlüssel.	381
Locator.	383
LocatorByKey.	384
LocatorByOccurrence.	385
Kapitel 22: Streamer.	387
Übersicht über Streamer.	387
Text-Streamer.	388
Segmente.	388
Einfache Segmente.	388
Komplexe Segmente.	388
Beispiel.	389
Verketten des Headers.	389
Ausgabe eines Streamers.	390
Verwendung von Markern und Variablen in Streamern.	390
Erstellen eines Streamers.	391
XML-Streamer.	392
Standardeigenschaften von Streamern.	394
Streamer-Komponente: Referenz.	395
ComplexSegment.	395
ComplexXmlSegment.	395
JsonStreamer.	396

MarkerStreamer.	397
SimpleSegment.	399
SimpleXmlSegment.	400
Streamer.	401
StreamerVariable.	402
XmlSegment.	402
XmlStreamer.	403
Streamer-Unterkomponente: Referenz.	404
AddHeaderModifier.	405
AddStringModifier.	406
DoNothingModifier.	406
WellFormedModifier.	406
WriteSegment.	407
Kapitel 23: Validatoren, Benachrichtigungen und Fehlerbehandlung.	409
Überblick über Validatoren, Benachrichtigungen und Fehlerbehandlung.	409
Fehlerbehandlung.	410
Verwendung der Eigenschaft „optional“ zur Fehlerbehandlung.	410
Schreiben einer Fehlermeldung in das Benutzerprotokoll.	411
Validatoren.	413
Standardeigenschaften von Validatoren.	414
Validator-Komponenten: Referenz.	414
AlternativeValidators.	415
EDIFACTValidation.	416
Aufzählungen.	417
LengthEquals.	418
MaxLength.	419
MaxNumber.	420
MinLength.	421
MinNumber.	422
NumberEquals.	423
ValidateByExpression.	424
ValidateByPattern.	425
ValidateByTransformer.	426
ValidateByType.	427
ValidateDate.	428
ValidatorPipeline.	429
Benachrichtigungen.	430
Referenz der Benachrichtigungskomponente.	431
Benachrichtigung.	431
NotificationGroup.	432
NotificationHandler.	432
NotifyFailure.	433

Kapitel 24: Validierungsregeln.....	435
Validierungsregeln - Übersicht.	435
Validierungsregelement - Verweis.	436
Attribute von Assert-Elementen.	436
Listenelementattribute.	437
Lookup-Elementattribute.	437
Regelementattribute.	438
Trace-Elementattribute.	438
Variablenelementattribute.	439
XPath-Editor.	439
XPath-Erweiterungen.	439
Bearbeiten der Validierungsregeln in einem externen Editor.	442
Erstellen eines Validierungsregelobjekts.	442
Importieren eines Data Transformation-Diensts mit Validierungsregeln.	443
 Kapitel 25: Benutzerdefinierte Skriptkomponenten.....	 444
Überblick über benutzerdefinierte Skriptkomponenten.	444
Beispiel für eine benutzerdefinierte Komponente.	444
Eigenschaften der benutzerdefinierten Komponente.	445
Entwickeln einer benutzerdefinierten Komponente.	445
Beispiel für eine Java-Benutzeroberfläche.	446
Beispiel für benutzerdefinierte Java-Komponenten.	446
Konfigurieren einer benutzerdefinierten Komponente.	447
Beispiel für Skript mit benutzerdefinierten Komponenten.	448
 Index.	 449

Einleitung

Verwenden Sie das *Data Transformation-Benutzerhandbuch*, um Informationen zum Entwerfen, Konfigurieren, Testen und Bereitstellen der Datenprozessorumwandlung zu erhalten. Dieses Handbuch enthält detaillierte Referenzabschnitte, in denen die Umwandlungskomponenten und -eigenschaften erläutert werden.

Das *Data Transformation-Benutzerhandbuch* richtet sich an Entwickler, Analysten und andere Benutzer von Data Transformation, die Umwandlungen entwerfen und implementieren. Es wird davon ausgegangen, dass Sie sich mit Informatica Developer auskennen. Außerdem geht es davon aus, dass Sie XML, Schemata und grundlegende Programmiertechniken beherrschen.

Informatica-Ressourcen

Informatica stellt Ihnen über das Informatica-Netzwerk und andere Online-Portale zahlreiche Produktressourcen zur Verfügung. Nutzen Sie die Ressourcen, um Ihre Informatica-Produkte und -Lösungen optimal zu nutzen und von anderen Informatica-Benutzern und Fachspezialisten zu lernen.

Informatica Network

Das Informatica Network bietet Zugriff auf zahlreiche Ressourcen, darunter die Informatica-Wissensdatenbank und der globale Kundensupport von Informatica. Um auf das Informatica Network zuzugreifen, besuchen Sie <https://network.informatica.com>.

Als Mitglied des Informatica Network haben Sie die folgenden Optionen:

- Durchsuchen Sie die Wissensdatenbank nach Produktressourcen.
- Zeigen Sie Informationen zur Produktverfügbarkeit an.
- Erstellen und überprüfen Sie Ihre Supportfälle.
- Ihr lokales Informatica Network für Benutzergruppen suchen und mit anderen Benutzern zusammenarbeiten.

Informatica-Wissensdatenbank

In der Informatica-Wissensdatenbank finden Sie Produktressourcen wie beispielsweise praktische Anleitungen, Best Practices, Videotutorials und Antworten auf häufig gestellte Fragen.

Für die Suche in der Wissensdatenbank besuchen Sie <https://search.informatica.com>. Wenn Sie Fragen, Kommentare oder Ideen zur Wissensdatenbank haben, wenden Sie sich per E-Mail an das Team der Informatica-Wissensdatenbank unter KB_Feedback@informatica.com.

Informatica-Dokumentation

Verwenden Sie das Informatica-Dokumentationsportal, um in einer umfangreichen Dokumentationsbibliothek nach aktuellen und neuen Produktversionen zu suchen. Um das Dokumentationsportal zu erkunden, besuchen Sie <https://docs.informatica.com>

Wenn Sie Fragen, Kommentare oder Ideen zur Produktdokumentation haben, wenden Sie sich an das Informatica-Dokumentationsteam unter infa_documentation@informatica.com

Informatica-Produktverfügbarkeitsmatrizen

Produktverfügbarkeitsmatrizen (PAMs) geben die Versionen der Betriebssysteme, Datenbanken und Typen von Datenquellen und Zielen an, die in einer Produktversion unterstützt werden. Sie können die Informatica-PAMs unter <https://network.informatica.com/community/informatica-network/product-availability-matrices> durchsuchen.

Informatica Velocity

Informatica Velocity ist eine Sammlung von Tipps und Best Practices, die von den Professionellen Informatica-Diensten entwickelt wurden und auf praktischen Erfahrungen aus Hunderten von Datenmanagementprojekten basieren. Informatica Velocity umfasst das gesammelte Wissen von Informatica-Beratern, die mit Unternehmen auf der ganzen Welt zusammenarbeiten, um erfolgreiche Datenmanagementlösungen zu planen, zu entwickeln, bereitzustellen und zu warten.

Die Informatica Velocity-Ressourcen finden Sie unter <http://velocity.informatica.com>. Wenn Sie Fragen, Anregungen oder Ideen zu Informatica Velocity haben, wenden Sie sich an die professionellen Informatica-Dienste unter ips@informatica.com.

Informatica Marketplace

Informatica Marketplace ist ein Forum, das Lösungen zur Erweiterung und Verbesserung Ihrer Informatica-Implementierungen bereitstellt. Nutzen Sie die zahlreichen Lösungen von Informatica-Entwicklern und -Partnern im Marketplace, um Ihre Produktivität zu steigern und die Implementierungsdauer Ihrer Projekte zu verkürzen. Den Informatica Marketplace finden Sie unter <https://marketplace.informatica.com>.

Globaler Kundensupport von Informatica

Sie können sich telefonisch oder über das Informatica-Netzwerk an ein Global Support-Center wenden.

Die Telefonnummer des globalen Kundensupports von Informatica vor Ort finden Sie auf der Informatica-Website unter folgender Verknüpfung:

<https://www.informatica.com/services-and-training/customer-success-services/contact-us.html>.

Um im Informatica-Netzwerk nach Online-Supportressourcen zu suchen, besuchen Sie <https://network.informatica.com> und wählen Sie die eSupport-Option aus.

KAPITEL 1

Einführung in die Datenumwandlung

Dieses Kapitel umfasst die folgenden Themen:

- [Data Transformation - Überblick, 20](#)
- [Architektur des Data Transformation-Prozesses, 21](#)
- [Data Transformation-Komponenten, 22](#)

Data Transformation - Überblick

Data Transformation ist eine Anwendung zur Verarbeitung komplexer Dateien, z. B. Messaging-Formaten, HTML-Seiten und PDF-Dokumenten. Data Transformation wandelt auch Formate wie ACORD, HIPAA, HL7, EDI-X12, EDIFACT, und SWIFT um.

Data Transformation wird standardmäßig bei der Installation von Informatica Developer (dem Developer Tool) installiert. Sie können eine Datenprozessorumwandlung definieren, um komplexe Dateien in ein Mapping zu transformieren. Wenn Sie ein Mapping mit der Datenprozessorumwandlung ausführen, wird durch den Datenintegrationsdienst die Data Transformation-Engine zur Verarbeitung der Daten aufgerufen.

Die Data Transformation-Anwendung hat die folgenden Elemente:

Datenprozessorumwandlung

Eine Umwandlung, die komplexe Dateien in einem Mapping verarbeitet. Definieren Sie zum Verarbeiten der Daten im Developer Tool ein Data Transformation-Skript, eine XMap, eine Bibliothek oder ein Validierungsregelobjekt. Sie können die Umwandlung in ein SQL-Datendienst-Mapping, einen Web-Dienst oder ein Mapping-Profil einfügen.

Data Transformation-Dienst:

Eine Gruppe von Data Transformation-Objekten, die Sie aus der Datenprozessorumwandlung exportieren und eigenständig ausführen können. Sie exportieren einen Dienst in ein Data Transformation-Repository und führen den Dienst von dort aus.

Data Transformation-Repository

Ein Verzeichnis zum Speichern ausführbarer Dienste, die aus einer Datenprozessor-Umwandlung exportiert werden. Der Name des Repository-Verzeichnisses ist ServiceDB.

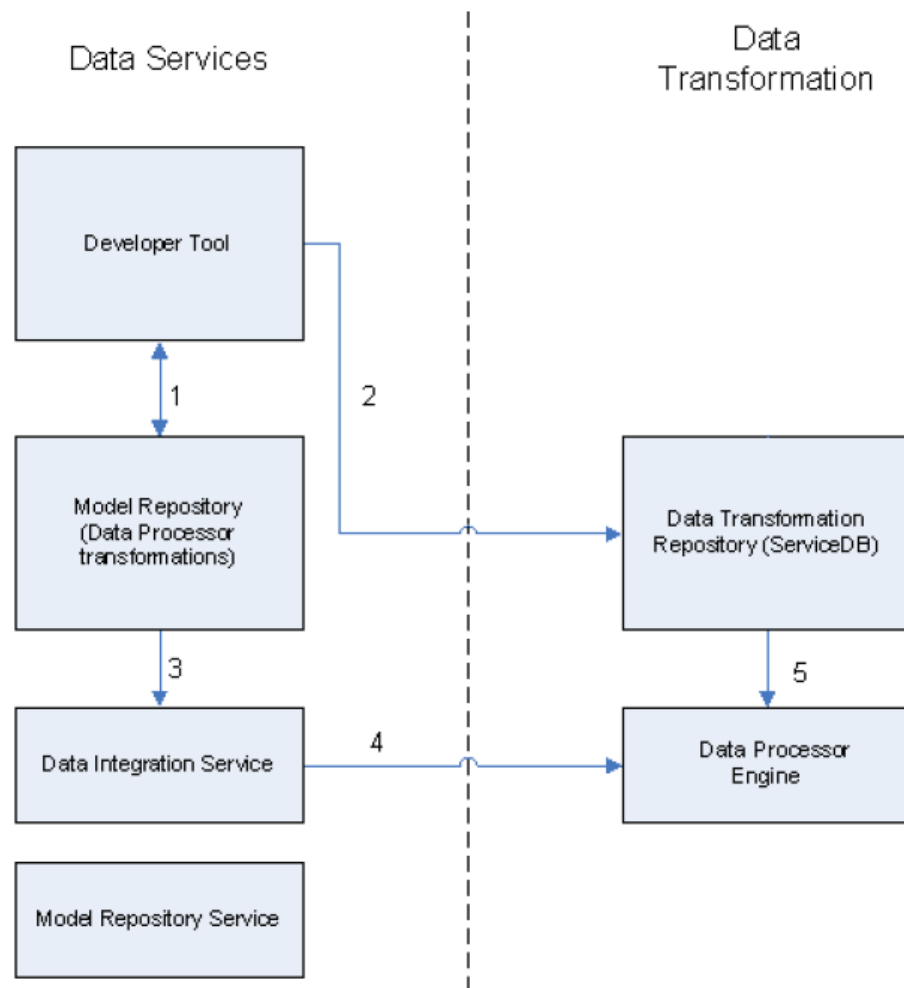
Datenprozessor-Engine

Ein Prozessor, der Objekte in der Datenprozessor-Umwandlung oder Dienste ausführt, die im Developer Tool erstellt werden.

Architektur des Data Transformation-Prozesses

Sie müssen Data Transformation installieren, um eine Datenprozessorumwandlung in Developer Tool konfigurieren und ausführen zu können. Die Datenprozessorumwandlung kann mehrere Skripte oder XMap-Objekte zur Umwandlung komplexer Dateien enthalten. Die Data Transformation-Engine führt die Skripte, Bibliotheken oder XMaps zur Umwandlung der Daten aus. Sie können eine Datenprozessorumwandlung in einem Datendienst, Web-Dienst oder Profil verwenden.

Die folgende Abbildung zeigt die Komponenten in der Data Transformation-Anwendung und die Komponenten, die Sie bei der Erstellung derselben Funktionalität im Developer Tool verwenden:



1. Erstellen Sie eine Datenprozessorumwandlung im Developer Tool. Speichern Sie die Umwandlung im Modellrepository.
2. Exportieren Sie die Datenprozessorumwandlung als Data Transformation-Dienst. Exportieren Sie den Dienst in das Data Transformation-Repository. Sie können den Dienst über das Repository ausführen.
3. Sie können eine Anwendung, die eine Datenprozessorumwandlung enthält, in einem Datenintegrationsdienst bereitstellen.

4. Der Datenintegrationsdienst führt die Anwendung aus und ruft die Datenprozessor-Engine zur Verarbeitung der Umwandlungslogik auf.
5. Die Datenprozessor-Engine führt auch Dienste aus dem Data Transformation-Repository aus.

Data Transformation-Komponenten

Wenn Sie einen Data Transformation-Dienst oder eine Datenprozessorumwandlung definieren, können Sie mehrere Komponenten kombinieren, um die Daten umzuwandeln.

Data Transformation enthält folgende Typen von Komponenten, die die Daten umwandeln:

Bibliothek

Wandeln Sie eine Eingabe eines branchenüblichen Meldungstyps in andere Formate um.

Mapper

Wandelt ein XML-Quelldokument in ein anderes XML-Dokument um.

Parser

Wandelt Quelldokumente in XML um. Die Eingabe kann jedes beliebige Format haben. Parser geben XML aus.

Serializer

Konvertiert eine XML-Datei in ein anderes Dokument. Die Ausgabe kann jedes beliebige Format haben.

Streamer

Spaltet umfangreiche Eingabedokumente, z. B. mehrere Gigabyte große Datenströme, in Segmente. Der Streamer teilt Dokumente, die mehrere Nachrichten oder mehrere Datensätze enthalten.

Transformer

Ändert Daten in jedes beliebige Format. Fügt Text hinzu, wandelt Text, entfernt und ändert Text. Verwenden Sie Transformer mit einem Parser, Mapper oder Serializer. Sie können einen Transformer auch als eigenständige Komponente ausführen.

XMap

Konvertiert eine hierarchische Quelle in eine andere hierarchische Struktur. XMap verfügt über dieselben Funktionen wie Mapper. Sie können jedoch ein Gitter im Developer Tool verwenden, um die Zuordnung zu definieren.

KAPITEL 2

Datenprozessor-Umwandlung

Dieses Kapitel umfasst die folgenden Themen:

- [Datenprozessorumwandlung - Überblick, 23](#)
- [Datenprozessorumwandlung - Ansichten, 24](#)
- [Datenprozessorumwandlung - Ports, 25](#)
- [Startkomponente, 27](#)
- [Verweise, 28](#)
- [Datenprozessorumwandlung – Einstellungen, 28](#)
- [Ereignisse, 35](#)
- [Protokolle, 37](#)
- [Entwicklung der Datenprozessorumwandlung, 39](#)
- [Export und Import von Datenprozessor-Umwandlungen, 44](#)
- [Datenprozessorumwandlung Validierung, 46](#)
- [Datenprozessorumwandlung in einer nicht nativen Umgebung, 47](#)

Datenprozessorumwandlung - Überblick

Die Datenprozessorumwandlungsprozesse verarbeiten unstrukturierte und halb strukturierte Dateiformate in einem Mapping. Konfigurieren Sie die Umwandlung für die Verarbeitung von Messaging-Formaten, HTML-Seiten, XML- JSON- und PDF-Dokumenten. Sie können auch strukturierte Formate wie ACORD, HIPAA, HL7, EDI-X12, EDIFACT und SWIFT konvertieren.

Ein Mapping verwendet eine Datenprozessorumwandlung, um Dokumente aus einem Format in ein anderes zu konvertieren. Die Datenprozessorumwandlung verarbeitet Dateien eines beliebigen Formats in einem Mapping. Wenn Sie eine Datenprozessorumwandlung erstellen, definieren Sie die Komponenten, die die Daten konvertieren.

Eine Datenprozessorumwandlung kann mehrere Komponenten zum Verarbeiten von Daten enthalten. Jede Komponente kann wiederum andere Komponenten enthalten.

Beispiel: Sie können Kundenrechnungen in Microsoft Word-Dateien erhalten. Sie konfigurieren eine Datenprozessorumwandlung, um die Daten von allen Word-Dateien zu parsen. Extrahieren Sie die Kundendaten in eine Kundentabelle. Extrahieren Sie Bestellinformationen in eine Tabelle der Bestellungen.

Wenn Sie eine Datenprozessorumwandlung erstellen, definieren Sie eine XMap, ein Skript oder eine Bibliothek. Eine XMap wandelt eine hierarchische Eingabedatei in eine hierarchische Ausgabedatei mit einer anderen Struktur um. Eine Bibliothek konvertiert einen branchenüblichen Nachrichtentyp in ein XML-

Dokument mit einer Hierarchiestruktur oder aus XML in ein branchenübliches Standardformat. Ein Skript kann Quelldokumente in ein hierarchisches Format parsen, das hierarchische Format in andere Dateiformate konvertieren oder ein hierarchisches Dokument einem anderen hierarchischen Format zuordnen.

Definieren Sie Skripts im IntelliScript-Editor der Datenprozessorumwandlung. Sie können die folgenden Skript-Typen definieren:

- **Parser.** Wandelt Quelldokumente in XML um. Parser geben immer XML aus. Die Eingabe kann ein beliebiges Format aufweisen, zum Beispiel Text, HTML, Word, PDF oder HL7.
- **Serializer.** Wandelt eine XML-Datei in ein Ausgabedokument beliebigen Formats um. Die Ausgabe eines Serializer kann ein beliebiges Format aufweisen, z. B. ein Text-, HTML- oder PDF-Dokument.
- **Mapper.** Wandelt ein XML-Quelldokument in eine andere XML-Struktur oder ein anderes XML-Schema um. Sie können dieselben XML-Dokumente wie in einer XMap konvertieren.
- **Transformer.** Ändert die Daten in jedes beliebige Format. Fügt Text hinzu, wandelt Text, entfernt und ändert Text. Verwenden Sie Transformer mit einem Parser, Mapper oder Serializer. Sie können einen Transformer auch als eigenständige Komponente ausführen.
- **Streamer.** Splittet umfangreiche Eingabedokumente, z. B. mehrere Gigabyte große Datenströme, in Segmente. Der Streamer verarbeitet Dokumente, die mehrere Meldungen oder Datensätze enthalten, wie z. B. HIPAA- oder EDI-Dateien.

Datenprozessorumwandlung - Ansichten

Die Datenprozessorumwandlung verfügt über zahlreiche Ansichten, auf die Sie beim Konfigurieren der Umwandlung zugreifen und die Sie im Developer Tool ausführen.

Einige der Ansichten der Datenprozessorumwandlung werden nicht standardmäßig im Developer Tool angezeigt. Um die Ansichten für die Umwandlung zu wechseln, klicken Sie auf **Fenster > Ansicht einblenden > Andere > Informatica**. Wählen Sie die Ansichten aus, die Sie verwenden möchten.

Die Datenprozessorumwandlung hat folgende feste Ansichten:

Ansicht „Übersicht“

Zum Konfigurieren von Ports und Definieren der Startkomponente.

Ansicht „Verweise“

Zum Hinzufügen oder Entfernen von Schemata aus der Umwandlung.

Einstellungen-Ansicht

Zum Konfigurieren von Umwandlungseinstellungen für die Codierung, Ausgabesteuerung und XML-Generierung.

Objekte-Ansicht

Hinzufügen, Ändern oder Löschen von Skript-, XMap- und Bibliotheks-Objekten aus der Umwandlung.

Sie haben auch Zugriff auf folgende Ansichten der Datenprozessorumwandlung:

Datenprozessor-Hex-Quelle-Ansicht

Zeigt ein Eingabedokument im Hexadezimalformat.

Ansicht der Datenprozessor-Ereignisse

Zeigt Informationen über Ereignisse, die bei der Umwandlung im Developer Tool auftreten. Zeigt Initialisierungs-, Ausführungs- und Übersichtereignisse.

Skript-Hilfe-Ansicht

Zeigt kontextbezogene Hilfe für den Skript-Editor an.

Daten-Viewer-Ansicht

Zum Anzeigen von Beispielseingabedaten, Ausführen der Umwandlung und Anzeigen der Ausgabeergebnisse.

Datenprozessorumwandlung - Ports

Definieren Sie die Datenprozessorumwandlungsports in der Ansicht **Übersicht**.

Eine Datenprozessor-Umwandlung kann Eingaben aus einer Datei, einem Puffer oder einem Streamline-Puffer aus einem Reader für komplexe Dateien lesen. Sie können einen Einfachdatei-Reader als Puffer zum Lesen einer ganzen Datei auf einmal verwenden. Sie können eine Eingabedatei auch aus einer Datenbank lesen.

Eine Datenprozessor-Umwandlung kann Eingaben aus einer Datei, einem Puffer oder einem Streamline-Puffer aus einem Reader für komplexe Dateien lesen. Sie können einen Einfachdatei-Reader als Puffer zum Lesen einer ganzen Datei auf einmal verwenden. Sie können eine Eingabedatei auch aus einer Datenbank lesen.

Die Ausgabeports, die Sie erstellen, hängen davon ab, ob Sie einen String, komplexe Dateien oder Zeilen relationaler Daten aus der Umwandlung zurückgeben möchten.

Datenprozessorumwandlung - Eingabeports

Wenn Sie eine Datenprozessorumwandlung erstellen, erstellt das Developer Tool einen Standard-Eingabeport. Wenn Sie einen weiteren Eingabeport in einer Skript-Startkomponente definieren, erstellt das Developer-Tool einen weiteren Eingabeport in der Umwandlung.

Der Eingabetyp bestimmt den Datentyp, der vom Datenintegrationsdienst an die Datenprozessorumwandlung übergeben wird. Der Eingabetyp bestimmt, ob es sich bei der Eingabe um Daten oder einen Quelldateipfad handelt.

Konfigurieren Sie einen der folgenden Eingabetypen:

Puffer

Die Datenprozessorumwandlung erhält Zeilen mit Quelldaten im Eingabeport. Wählen Sie den Puffereingabe-Typ, wenn Sie die Umwandlung für das Empfangen von Daten von einer flachen Datei oder von einer Informatica-Umwandlung konfigurieren.

Datei

Die Datenprozessorumwandlung empfängt den Quelldateipfad im Eingabeport. Die Startkomponente des Datenprozessors öffnet die Quelldatei. Verwenden Sie den Eingabetyp "Datei", um Binärdateien (z. B. Microsoft Excel- oder Microsoft Word-Dateien) zu analysieren. Sie können zur Verarbeitung mit einem Puffereingabeport auch den Datei-Eingabetyp für große Dateien verwenden, die möglicherweise viel Systemspeicher benötigen.

Dienstparameter

Die Datenprozessorumwandlung empfängt Werte zur Anwendung auf Variable in den Dienstparameter-Ports. Wenn Sie auswählen, dass die Variablen Eingabedaten empfangen, erstellt das Developer Tool einen Dienstparameterport für jede Variable.

Output_Filename

Wenn Sie den Standard-Ausgabeport so konfigurieren, dass er einen Dateinamen statt Rohdaten ausgibt, so erstellt das Developer Tool eine Output_Filename-Port. Sie können aus einem Mapping einen Dateinamen an den Output_Filename-Port übergeben.

Bei der Definition eines Eingabeports können Sie den Ort der Beispiels-Eingabedatei für den Port festlegen. Eine Beispiels-Eingabedatei ist ein kleines Beispiel der Eingabedatei. Referenzieren Sie eine Beispiels-Eingabedatei, wenn Sie Skripte erstellen. Sie verwenden die Beispiels-Eingabedatei auch, wenn Sie die Umwandlung in der **Daten-Viewer**-Ansicht testen. Definieren Sie die Beispiels-Eingabedatei im Feld **Eingabespeicherort**.

Ports für Dienstparameter

Sie können Eingabeports erstellen, die Werte für Variable empfangen. Die Variablen können einen beliebigen Datentyp enthalten, beispielsweise einen String, ein Datum oder eine Zahl. Darüber hinaus kann eine Variable eine Position für ein Quelldokument enthalten. Diese Variablen können Sie in einer Datenprozessor-Komponente referenzieren.

Wenn Sie für eine Variable einen Eingabeport erstellen, zeigt das Developer Tool eine Liste mit Variablen an, aus der Sie die Variable auswählen können.

Erstellen von Serviceparameterports

Sie können Eingabeports erstellen, die Werte für Variablen entgegennehmen. Sie können auch die Ports entfernen, die Sie aus Variablen erstellen.

1. Öffnen Sie die Ansicht **Übersicht** für die Datenprozessorumwandlung.
2. Klicken Sie auf **Auswählen**.
Das Developer Tool zeigt eine Liste der Variablen an und legt fest, welche Variablen bereits Ports haben.
3. Wählen Sie eine oder mehrere Variablen aus.
Das Developer Tool erstellt einen Puffereingabeport für jede von Ihnen ausgewählte Variable. Sie können den Port nicht ändern.
4. Um einen Port zu entfernen, den Sie aus einer Variablen erstellen, deaktivieren Sie die Auswahl aus der Variablenliste. Wenn Sie die Auswahl deaktivieren, entfernt das Developer Tool den Eingabeport.

Datenprozessorumwandlung - Ausgabeports

Die Datenprozessorumwandlung hat standardmäßig einen Ausgabeport. Wenn Sie zusätzliche Ausgabeports in einem Skript definieren, fügt das Developer Tool die Ports der Datenprozessorumwandlung hinzu. Sie können Gruppen und Ports erstellen, wenn Sie die Umwandlung zur Rückgabe relationaler Daten konfigurieren. Sie können auch Dienstparameterports und Pass-Through-Ports erstellen.

Standard-Ausgabeport

Die Datenprozessorumwandlung hat standardmäßig einen Ausgabeport. Wenn Sie relationale Ausgaben erstellen, können Sie Gruppen verbundener Ausgabeports anstatt des Standard-Ausgabeports definieren. Wenn Sie einen weiteren Ausgabeport in einer Skriptkomponente definieren, fügt das Developer-Tool einen weiteren Ausgabeport zur Umwandlung hinzu.

Konfigurieren Sie einen der folgenden Ausgabetypen für einen Standard-Ausgabeport:

Puffer

Die Datenprozessorumwandlung gibt XML über den Ausgabeport aus. Wählen Sie den Pufferdateityp, wenn Sie Dokumente parsen oder in der Datenprozessorumwandlung XML anderen XML-Dokumenten zuordnen.

Datei

Der Datenintegrationsdienst gibt einen Namen der Ausgabedatei im Ausgabeport für jede Quellinstanz oder -zeile aus. Die Datenprozessorumwandlungs-Komponente schreibt die Ausgabedatei, statt Daten über die Ausgabeports der Datenprozessorumwandlung auszugeben.

Wenn Sie einen Dateiausgabeport auswählen, erstellt das Developer Tool einen Output_Filename-Eingabeport. Sie können einen Dateinamen an den Output_Filename-Port übergeben. Die Datenprozessorumwandlung erstellt die Ausgabedatei mit einem Namen, den es in diesem Port empfängt.

Wenn der Name der Ausgabedatei leer ist, gibt der Datenintegrationsdienst einen Fehler aus. Wenn ein Fehler auftritt, schreibt der Datenintegrationsdienst einen Nullwert auf den Ausgabeport und gibt einen Zeilenfehler aus.

Wählen Sie den Ausgabetyt "Datei" aus, wenn Sie XML in eine Binärdatei, etwa eine PDF- oder Microsoft Excel-Datei, umwandeln möchten.

Pass-Through-Ports

Sie können Pass-Through-Ports für beliebige Datenprozessorumwandlungen konfigurieren. Pass-Through-Ports sind Eingabe- und Ausgabe-Ports, die Eingabedaten empfangen und dieselben Daten in ein zurückgeben, ohne sie zu ändern.

Sie können Pass-Through-Ports in einer Datenprozessorumwandlungsinstanz definieren, die sich in einem Mapping befindet.

Um einen Pass-Through-Port hinzuzufügen, ziehen Sie einen Port von einer anderen Umwandlung in das Mapping. Sie können Ports auch über die Registerkarte **Ports** in der Ansicht **Eigenschaften** hinzufügen. Klicken Sie auf **Neu**, um einen Pass-Through-Port hinzuzufügen.

Hinweis: Wenn Sie Pass-Through-Ports zu einer Datenprozessorumwandlung mit relationaler Eingabe und hierarchischer Ausgabe hinzufügen, fügen Sie die Ports zur Root-Gruppe der relationalen Struktur hinzu.

Datenprozessorumwandlungen können Pass-Through-Ports mit benutzerdefinierten Datentypen enthalten.

Startkomponente

Eine Startkomponente definiert die Komponente, die die Verarbeitung in der Datenprozessorumwandlung startet. Konfigurieren Sie die Startkomponente in der Ansicht **Übersicht**.

Eine Datenprozessorumwandlung kann mehrere Komponenten zum Verarbeiten von Daten enthalten. Jede Komponente kann wiederum andere Komponenten enthalten. Sie müssen die Komponente ermitteln, die den Eintrittspunkt für die Umwandlung darstellt.

Wenn Sie die Startkomponente in einer Datenprozessorumwandlung konfigurieren, können Sie als Startkomponente eine XMap, eine Bibliothek oder eine Skriptkomponente auswählen. In Bezug auf Skripte können Sie einen der folgenden Komponententypen auswählen:

- **Parser.** Wandelt Quelldokumente in XML um. Die Eingabe kann ein beliebiges Format aufweisen, beispielsweise Text, HTML, Word, PDF oder HL7.

- Mapper. Wandelt ein XML-Quelldokument in eine andere XML-Struktur oder ein anderes XML-Schema um.
- Serializer. Wandelt eine XML-Datei in ein Ausgabedokument beliebigen Formats um.
- Streamer. Splittet umfangreiche Eingabedokumente, z. B. mehrere Gigabyte große Datenströme, in Segmente.
- Transformer. Ändert die Daten in jedes beliebige Format. Fügt Text hinzu, wandelt Text, entfernt und ändert Text. Verwenden Sie Transformer mit einem Parser, Mapper oder Serializer. Sie können einen Transformer auch als eigenständige Komponente ausführen.

Hinweis: Wenn die Startkomponente keine XMap oder Bibliothek ist, können Sie die Startkomponente auch stattdessen in einem Skript in der Ansicht **Übersicht** konfigurieren.

Verweise

Sie können Umwandlungsverweise (z. B. Schema- oder Mapplet-Verweise) definieren, indem Sie ein Schema oder Mapplet auswählen, das als Verweis dienen soll. Bestimmte Datenprozessorumwandlungen erfordern ein hierarchisches Schema, um die Eingabe- oder Ausgabehierarchie für relevante Komponenten in der Umwandlung zu definieren. Zum Verwenden des Schemas in der Umwandlung definieren Sie einen Schemaverweis für die Umwandlung. Sie können auch eine spezielle Aktion (die RunMapplet-Aktion) verwenden, um ein Mapplet aus einer Datenprozessorumwandlung abzurufen. Zum Abrufen eines Mapplets müssen Sie zuerst einen Mapplet-Verweis für die Umwandlung definieren.

Sie können Umwandlungsverweise, wie z. B. Schema- oder Mapplet-Verweise, in der Ansicht **Verweise** der Umwandlung definieren.

Schemaverweise

Die Datenprozessorumwandlung verweist auf Schemaobjekte im Modellrepository. Die Schemaobjekte können im Repository vorhanden sein, bevor Sie die Umwandlung erstellen. Sie können Schemas auch aus der Ansicht **Verweise** der Umwandlung importieren.

Die Schemacodierung muss mit der Eingabecodierung für Serializer- oder Mapper-Objekte übereinstimmen. Die Schemacodierung muss mit der Ausgabecodierung für Parser- oder Mapper-Objekte übereinstimmen. Konfigurieren Sie die Arbeitscodierung in der Ansicht **Einstellungen** der Umwandlung.

Ein Schema kann auf zusätzliche Schemas verweisen. Das Developer-Tool zeigt den Namensraum und das Präfix für jedes Schema, auf das die Datenprozessorumwandlung verweist. Wenn Sie mit leeren Namensräumen auf mehrere Schemas verweisen, ist die Umwandlung ungültig.

Mapplet-Verweise

Sie können ein Mapplet mit der RunMapplet-Aktion aus einer Datenprozessorumwandlung abrufen. Bevor Sie die RunMapplet-Aktion zu einer Datenprozessorumwandlungs-Komponente hinzufügen, müssen Sie zuerst einen Verweis auf das aufzurufende Mapplet definieren.

Datenprozessorumwandlung – Einstellungen

Konfigurieren Sie Codepages, XML-Verarbeitungsoptionen und Protokollierungseinstellungen in der Ansicht **Einstellungen** der Datenprozessorumwandlung.

Zeichencodierung

Eine Zeichenkodierung ist eine Zuordnung der Zeichen von einer Sprache oder Sprachgruppe zu Hexadezimalcode.

Wenn Sie ein Skript zuweisen, definieren Sie die Codierung der Eingabe- und Ausgabedokumente. Definieren Sie die Arbeitscodierung, um festzulegen, wie der IntelliScript-Editor Zeichen anzeigt und wie die Datenprozessorumwandlung die Zeichen verarbeitet.

Arbeitscodierung

Die Arbeitscodierung ist die Codepage für die Daten im Arbeitsspeicher und die Codepage für die Daten, die in der Benutzerschnittstelle und in Arbeitsdateien erscheinen. Sie müssen eine Arbeitscodierung auswählen, die mit der Codierung der Schemata übereinstimmt, die Sie bei der Datenprozessorumwandlung aufrufen.

Aus der folgenden Tabelle sind die Einstellungen für die Arbeitscodierung ersichtlich:

Einstellung	Beschreibung
Standardcodepage der Datenverarbeitung verwenden	Verwendet die Standardcodierung aus der Datenprozessorumwandlung.
Andere	Wählen Sie die Codierung aus der Liste aus.
XML-Codierung von Sonderzeichen	<p>Legt die Darstellung von XML-Sonderzeichen fest. Sie können Keine oder XML auswählen.</p> <ul style="list-style-type: none">- Keine. Als & amp; & lt; & gt; & quot; & apos; lassen Entitätsreferenzen für XML-Sonderzeichen werden als Text interpretiert. Das Zeichen > erscheint zum Beispiel als & gt; Standardwert ist "keine".- XML. In & < > " ' umwandeln Entitätsreferenzen für XML-Sonderzeichen werden als normale Zeichen interpretiert. Beispielsweise erscheint & gt; als folgendes Zeichen: >

Eingabecodierung

Die Eingabecodierung bestimmt, wie Zeichendaten in Eingabedokumenten codiert werden. Sie können die Codierung für zusätzliche Eingabeports in einem Skript konfigurieren.

Die nachstehende Tabelle beschreibt die Codierungs-Einstellungen für den **Eingabe**-Bereich:

Einstellung	Beschreibung
Codierung aus Eingabedokument verwenden	<p>Verwendet die Codepage, die im Quelldokument festgelegt wurde, wie z. B. das Codierungsattribut eines XML-Dokuments.</p> <p>Wenn das Quelldokument keine Codierungsspezifikation enthält, verwendet die Datenprozessorumwandlung die Codierungseinstellungen aus der Ansicht Einstellungen.</p>
Arbeitscodierung verwenden	Es wird dieselbe Codierung verwendet wie für die Arbeitscodierung.
Andere	Wählen Sie die Eingabecodierung aus der Dropdown-Liste aus.

Einstellung	Beschreibung
XML-Codierung von Sonderzeichen	<p>Legt die Darstellung von XML-Sonderzeichen fest. Sie können Keine oder XML auswählen.</p> <ul style="list-style-type: none"> - Keine. Als & &lt; &gt; &quot; &apos; lassen Entitätsreferenzen für XML-Sonderzeichen werden als Text interpretiert, beispielsweise erscheint das Zeichen > als &gt; Standardeinstellung ist "Keine". - XML. In & < > " ' umwandeln Entitätsreferenzen für XML-Sonderzeichen werden als normale Zeichen interpretiert. Beispielsweise erscheint &gt; als folgendes Zeichen: >
Bytereihenfolge	<p>Beschreibt, wie Mehrbytezeichen im Eingabedokument angezeigt werden. Sie können die folgenden Optionen auswählen:</p> <ul style="list-style-type: none"> - Little-Endian. Das Byte mit den am wenigsten signifikanten Bit erscheint zuerst. Standard. - Big-Endian. Das Byte mit den signifikantesten Bits erscheint zuerst. - Keine binäre Umwandlung.

Ausgabecodierung

Die Ausgabecodierung bestimmt, wie Zeichendaten im Haupt-Ausgabedokument codiert werden.

Die nachstehende Tabelle beschreibt die Codierungs-Einstellungen für den **Ausgabe**-Bereich:

Einstellung	Beschreibung
Arbeitscodierung verwenden	Die Ausgabecodierung ist mit der Arbeitscodierung identisch.
Andere	Der Benutzer wählt die Ausgabecodierung aus der Liste aus.
XML-Codierung von Sonderzeichen	<p>Legt die Darstellung von XML-Sonderzeichen fest. Sie können Keine oder XML auswählen.</p> <ul style="list-style-type: none"> - Keine. Als & &lt; &gt; &quot; &apos; lassen Entitätsreferenzen für XML-Sonderzeichen werden als Text interpretiert, beispielsweise erscheint das Zeichen > als &gt; Standard. - XML. In & < > " ' umwandeln Entitätsreferenzen für XML-Sonderzeichen werden als normale Zeichen interpretiert. Beispielsweise erscheint &gt; als folgendes Zeichen: >
Identisch mit Eingabecodierung	Die Ausgabecodierung ist mit der Eingabecodierung identisch.
Bytereihenfolge	<p>Beschreibt, wie Mehrbytezeichen im Eingabedokument angezeigt werden. Sie können die folgenden Optionen auswählen:</p> <ul style="list-style-type: none"> - Little-Endian. Das Byte mit den am wenigsten signifikanten Bit erscheint zuerst. Standard. - Big-Endian. Das Byte mit den signifikantesten Bits erscheint zuerst. - Keine binäre Umwandlung.

Regeln und Richtlinien für die Zeichencodierung

Verwenden Sie die folgenden Regeln und Richtlinien, wenn Sie Codierungen konfigurieren:

- Um die Leistung zu steigern, legen Sie die Arbeitscodierung mit der Codierung fest, die auch das Ausgabedokument aufweist.
- Legen Sie die Eingabecodierung mit der Codierung des Eingabedokuments fest.
- Legen Sie die Ausgabecodierung mit der Codierung des Ausgabedokuments fest.
- Für Sprachen mit Mehrbyte-Zeichen legen Sie die Arbeitscodierung mit UTF-8 fest. Sie können für die Eingabe- und Ausgabecodierung eine Unicode-Codierung wie UTF-8 oder eine Doppelbyte-Codepage wie Big-5 oder Shift_JIS verwenden.

Ausgabeeinstellungen

Konfigurieren Sie Ausgabesteuerungseinstellungen, um zu steuern, ob die Datenprozessorumwandlung Ereignisprotokolle erstellt und Ausgabedokumente speichert.

Sie können den Meldungstyp steuern, den die Datenprozessorumwandlung in das Entwicklungszeit-Ereignisprotokoll schreibt: Wenn Sie die geparsten Eingabedokumente mit den Ereignisprotokollen speichern, können Sie den Kontext, in dem der Fehler auftrat, in der **Ereignis**-Ansicht anzeigen.

Die folgende Tabelle beschreibt die Einstellungen im **Entwurfszeit-Ereignisse** Bereich:

Einstellung	Beschreibung
Entwurfszeitereignisse protokollieren	Legt fest, ob ein Entwicklungszeit-Ereignisprotokoll erstellt wird. Standardmäßig protokolliert die Datenprozessorumwandlung Benachrichtigungen, Warnungen und Fehlschläge im Entwicklungszeit-Ereignisprotokoll. Sie können die folgenden Typen von Ereignissen ausschließen: <ul style="list-style-type: none">- Benachrichtigungen- Warnungen- Fehlschläge
Geparste Dokumente speichern	Legt fest, wann die Datenprozessorumwandlung ein geparstes Eingabedokument speichert. Sie können die folgenden Optionen auswählen: <ul style="list-style-type: none">- Immer.- Nie- Bei Ausfall Voreingestellt ist der Wert "Immer".

Die folgende Tabelle beschreibt die Einstellungen im **Laufzeitereignisse**-Bereich:

Einstellung	Beschreibung
Laufzeitereignisse protokollieren	Legt fest, ob bei der Ausführung der Umwandlung aus einer Zuordnung ein Ereignisprotokoll erstellt wird. <ul style="list-style-type: none">- Nie.- Bei Ausfall Standardwert ist „Nie“.

Die folgende Tabelle beschreibt die Einstellungen im **Ausgabe**-Bereich:

Einstellung	Beschreibung
Automatische Ausgabe deaktivieren	Legt fest, ob die Datenprozessorumwandlung die Ausgabe in die Standardausgabedatei schreibt. Deaktivieren Sie die Standardausgabe in folgenden Situationen: <ul style="list-style-type: none"> - Sie geben die Ausgabe eines Parsers an die Eingabe einer anderen Komponente weiter, bevor die Umwandlung eine Ausgabedatei erstellt. - Sie verwenden eine WriteValue-Aktion, um Daten direkt vom Skript in die Ausgabe zu schreiben, anstatt die Daten durch die Ausgabeports zu schicken.
Wertkomprimierung deaktivieren	Legt fest, ob die Datenprozessorumwandlung zur Optimierung der Speichernutzung Wertkomprimierung verwendet. Wichtig: Deaktivieren Sie Wertkomprimierung nur auf Anraten des globalen Kundensupports von Informatica.

Die folgende Tabelle beschreibt die Einstellungen im Bereich **Binärer Ausgabeport: Auflistungsmodus**. Sie können eine dieser Optionen für Binärausgabe für eine relationale in hierarchische Umwandlung mit XML-, Avro- oder Parquet-Ausgabe oder für einen Datenprozessor-Umwandlungs-Parser mit Avro- oder Parquet-Ausgabe auswählen.

Einstellung	Beschreibung
Eingabezeilen in einer einzigen Ausgabe auflisten	Legt fest, ob die Datenprozessor-Umwandlung die relationale Eingabe in einem einzelnen binären Ausgabeport akkumuliert.
Ausgabe teilen, wenn die Größe folgenden Wert überschreitet:	Legt fest, ob die Datenprozessor-Umwandlung die Ausgabe basierend auf der festgelegten Maximalgröße in Segmente unterteilt.
Ausgabezeile für jede Zeile (nicht auflisten)	Legt fest, ob die Datenprozessor-Umwandlung die Ausgabe in separate Zeilen übergibt.

Verarbeitungseinstellungen

Die Verarbeitungseinstellungen definieren, wie die Datenprozessorumwandlung ein Element ohne einen definierten Datentyp verarbeitet. Die Einstellungen betreffen Skripte. Die Einstellungen betreffen keine Elemente, die von einem XMap-Objekt verarbeitet werden.

Die folgende Tabelle beschreibt die Verarbeitungseinstellungen, die die XML-Verarbeitung in Skripten beeinflussen:

Einstellung	Beschreibung
Als xs:string behandeln	Die Datenprozessorumwandlung behandelt ein Element ohne Typ als Zeichenfolge. Im Dialogfeld XPath auswählen wird das Element oder Attribut als einzelner Knoten angezeigt.
Als xs:anyType behandeln	Die Datenprozessorumwandlung behandelt ein Element ohne Typ als anyType. Im Dialogfeld XPath auswählen erscheint das Element oder Attribut als Knotenbaumstruktur. Ein Knoten ist der Typ xs:string und alle benannten komplexen Datentypen erscheinen als Baumknoten.

In der folgenden Tabelle wird eine Verarbeitungseinstellung beschrieben, die sich auf die Streamer-Verarbeitung auswirkt:

Einstellung	Beschreibung
Streamer-Segmentgröße	Diese Einstellung definiert die Menge der Daten, die der Streamer jedes Mal aus einem Eingabedateistrom liest. Die Datenprozessorumwandlung wendet diese Einstellung auf einen Streamer mit einer Dateieingabe an.

In der folgenden Tabelle wird eine Verarbeitungseinstellung beschrieben, die sich auf die hierarchische und die relationale Umwandlungsverarbeitung auswirkt:

Einstellung	Beschreibung
Strikte Validierung erzwingen	Diese Einstellung legt fest, ob die Datenprozessor-Umwandlung strikte Validierung für hierarchische Eingabe durchführt. Bei strenger Validierung muss die hierarchische Eingabedatei strikt ihrem Schema entsprechen. Diese Option kann angewendet werden, wenn der Datenprozessormodus auf Ausgabebezuordnung eingestellt ist, womit Ausgabeports für die relationale Ausgabe erzeugt werden. Diese Option gilt nicht für Mappings mit JSON-Eingabe von Versionen vor Version 10.2.1.
XML-Eingabe normalisieren	Mit dieser Einstellung wird festgelegt, ob die Datenprozessor-Umwandlung die XML-Eingabe normalisiert. Standardmäßig führt die Umwandlung Normalisierung für XML-Eingabe durch. In bestimmten Fällen möchten Sie die automatische Normalisierung vielleicht überspringen, um die Leistung zu steigern.

XMap-Einstellungen

Die XMap-Einstellung definiert, wie die Datenprozessorumwandlung XMap-Eingabeelemente verarbeitet, die nicht in Ausgabeelemente umgewandelt werden. Die ungelesenen Elemente werden an einen dedizierten Port namens **XMap_Unread_Input_Values** übergeben. Die Einstellung hat nur Auswirkungen, wenn die XMap als Startkomponente ausgewählt ist. Die Einstellung hat keine Auswirkungen auf Elemente, die die XMap verarbeitet.

Um ungelesene XMap-Elemente an einen dedizierten Port zu übergeben, aktivieren Sie die Einstellung **Ungelesene Elemente in einen zusätzlichen Ausgabeport schreiben**.

Konfigurieren der XML-Ausgabe

Die Einstellungen der XML-Erzeugung definieren die Merkmale von XML-Ausgabedokumenten.

In der folgenden Tabelle werden die Einstellungen der XML-Erzeugung im Bereich **Schematitel** beschrieben:

Einstellung	Beschreibung
Speicherort des Schemas	Definiert die schemaLocation für das Stammelement des Hauptausgabedokuments.
Speicherort des Schemas ohne Namensraum	Definiert das Attribut „xsi:noNamespaceSchemaLocation“ für das Stammelement des Hauptausgabedokuments.

Konfigurieren Sie die Einstellungen des XML-Ausgabemodus, um festzulegen, wie sich die Datenprozessorumwandlung bei fehlenden Elementen oder Attributen im XML-Eingabedokument verhalten

soll. In der folgenden Tabelle werden die Einstellungen der XML-Erzeugung im Bereich **XML-Ausgabemodus** beschrieben:

Einstellung	Beschreibung
Unverändert	Fügen Sie keine leeren Elemente hinzu bzw. entfernen Sie sie nicht. Standardwert ist „Aktiviert“.
Komplett	Alle erforderlichen und optionalen Elemente, die im Ausgabeschema definiert sind, werden in die Ausgabe geschrieben. Elemente ohne Inhalt werden als leere Elemente geschrieben.
Kompakt	Entfernt leere Elemente aus der Ausgabe. Wenn Für Elemente hinzufügen aktiviert ist, entfernt die Datenprozessorumwandlung nur die optionalen Elemente. Wenn Für Elemente hinzufügen deaktiviert ist, entfernt die Datenprozessorumwandlung alle leeren Elemente. Möglicherweise ist die XML-Ausgabe nicht gültig.

In der folgenden Tabelle werden die Einstellungen der XML-Erzeugung im Bereich **Standardwerte für benötigte Knoten** beschrieben:

Einstellung	Beschreibung
Für Elemente hinzufügen	Wenn das Ausgabeschema einen Standardwert für ein erforderliches Element definiert, wird das Element mit einem Standardwert in die Ausgabe aufgenommen. Standardwert ist „Aktiviert“.
Für Attribute hinzufügen	Wenn das Ausgabeschema einen Standardwert für ein erforderliches Attribut definiert, wird das Attribut mit dem zugehörigen Standardwert in die Ausgabe aufgenommen. Standardwert ist „Aktiviert“.
Hinzugefügte Werte validieren	Legt fest, ob die Datenprozessorumwandlung leere Elemente validiert, die von der Ausgabe im kompletten Modus hinzugefügt werden. Standardwert ist „Deaktiviert“. Wenn Hinzugefügte Werte validieren aktiviert ist und das Schema keine leeren Elemente zulässt, ist die XML-Ausgabe möglicherweise ungültig.

In der folgenden Tabelle werden die Einstellungen der XML-Erzeugung im Bereich **Verarbeitungsanweisungen** beschrieben:

Einstellung	Beschreibung
XML-Verarbeitungsanweisungen hinzufügen	Definiert die Zeichencodierung und die XML-Version des Ausgabedokuments. Standardwert ist "selected".
XML-Version	Definiert die XML-Version. Die Einstellung der XML-Version weist die folgenden Optionen auf: - 1.0 - 1.1 Standard ist 1.0.

Einstellung	Beschreibung
Codierung	Definiert die Zeichencodierung, die in der Verarbeitungsanweisung angegeben ist. Die Einstellung „Codierung“ weist die folgenden Optionen auf: <ul style="list-style-type: none"> - Wie Ausgabecodierung. Die Ausgabecodierung in der Verarbeitungsanweisung ist mit der Ausgabecodierung identisch, die in den Einstellungen der Datenprozessorumwandlung definiert ist. - Benutzerdefiniert. Definiert die Ausgabecodierung in der Verarbeitungsanweisung. Der Benutzer gibt den Wert in dem Feld ein.
Benutzerdefinierte Verarbeitungsanweisungen hinzufügen	Fügt dem Ausgabedokument weitere Verarbeitungsanweisungen hinzu. Geben Sie die Verarbeitungsanweisung genau so ein, wie sie im Ausgabedokument angezeigt wird. Standardwert ist „Deaktiviert“.

In der folgenden Tabelle werden die Einstellungen der XML-Erzeugung im Bereich **XML-Root** beschrieben:

Einstellung	Beschreibung
XML-Stammelement hinzufügen	Fügt dem Ausgabedokument ein Stammelement hinzu. Verwenden Sie diese Option, wenn das Ausgabedokument mehrere Instanzen des Stammelements enthält, das im Ausgabeschema definiert ist. Standardwert ist „Deaktiviert“.
Name des Stammelements	Definiert einen Namen für das Stammelement, das dem Ausgabedokument hinzugefügt wird.

Ereignisse

Ein Ereignis ist ein Datensatz eines Verarbeitungsschritts einer Komponente in einer Datenprozessorumwandlung. In einem Skript oder einer Bibliothek generiert jeder Anker, jede Aktion oder jeder Transformer ein Ereignis. In einem XMap-Objekt generiert jede Mapping-Anweisung ein Ereignis.

Sie können Ereignisse in der Ansicht **Datenprozessor-Ereignisse** anzeigen.

Ereignistypen

Die Datenprozessorumwandlung schreibt Ereignisse in Protokolldateien. Jedes Ereignis hat einen Ereignistyp, der anzeigt, ob das Ereignis erfolgreich war, fehlgeschlagen ist oder mit Fehlern ausgeführt wurde.

Eine Komponente kann ein oder mehrere Ereignisse generieren. Die Komponente kann übergeben werden oder fehlschlagen, je nachdem, ob die Ereignisse erfolgreich sind oder fehlschlagen. Schlägt ein Ereignis fehl, so schlägt eine Komponente fehl.

Die folgende Tabelle beschreibt die Ereignistypen, die die Datenprozessorumwandlung generiert:

Ereignistyp	Beschreibung
Benachrichtigung	Normaler Vorgang.
Warnung	Die Datenprozessorumwandlung wurde ausgeführt, aber es trat eine unerwartete Bedingung auf. Beispiel: Die Datenprozessorumwandlung schrieb Daten mehrmals in dasselbe Element. Die Datenprozessorumwandlung generiert jedes Mal, wenn das Element überschrieben wird, eine Warnung.
Fehler	Die Datenprozessorumwandlung wurde ausgeführt, aber eine Komponente schlug fehl. Beispiel: Ein erforderliches Eingabeelement war leer.
Optionalen Fehler	Die Datenprozessorumwandlung wurde ausgeführt, aber eine optionale Komponente schlug fehl. Beispiel: Ein optionaler Anker fehlte im Quelldokument.
Schwerwiegender Fehler	Die Datenprozessorumwandlung schlug aufgrund eines schwerwiegenden Fehlers fehl. Beispiel: Das Eingabedokument war nicht vorhanden.

Ansicht der Datenprozessor-Ereignisse

Die **Datenprozessor-Ereignisse**-Ansicht zeigt Ereignisse an, wenn Sie eine Datenprozessorumwandlung über das Developer Tool ausführen.

Die **Datenprozessor-Ereignisse**-Ansicht hat einen **Navigationsbereich** und einen **Detailbereich**. Der Navigationsbereich enthält einen Navigationsbaum. Im Navigationsbaum sind die Komponenten aufgelistet, die die Umwandlung in chronologischer Reihenfolge ausführte. Jeder Knoten hat ein Symbol unten im Baum, das das schwerwiegendste Ereignis darstellt. Wenn Sie einen Knoten im **Navigationsbereich** auswählen, werden im **Detailbereich** Ereignisse angezeigt.

Der Navigationsbaum enthält die folgenden Knoten auf der höchsten Ebene:

- **Dienstinitialisierung.** Beschreibt die Dateien und Variablen, die die Datenprozessorumwandlung initialisiert.
- **Ausführung.** Listet die Komponenten auf, die durch das Skript, die Bibliothek oder die XMap ausgeführt wurden.
- **Zusammenfassung.** Zeigt Statistiken zu Verarbeitung an.

Wenn Sie XMap ausführen, hat jeder Knotenname im Navigationsbereich eine Zahl in eckigen Klammern, z. B. [5]. Um die Anweisung zu identifizieren, die die Ereignisse für den Knoten generierte, klicken Sie mit der rechten Maustaste in das Anweisungen-Gitter und wählen Sie "Zu Zeilennummer wechseln". Geben Sie die Knotennummer ein.

Wenn Sie ein Skript ausführen und auf ein Ereignis im **Navigations-** oder **Detailbereich** doppelklicken, hebt der Skript-Editor die Skriptkomponente hervor, die das Ereignis generiert hat. Der **Eingabe-Bereich** der **Daten-Viewer**-Ansicht hebt den Teil des Beispiel-Quelldokuments hervor, der das Ereignis generierte.

Protokolle

Ein Protokoll enthält einen Datensatz der Datenprozessorumwandlung. Die Datenprozessorumwandlung schreibt Ereignisse in Protokolle.

Die Datenprozessorumwandlung erstellt die folgenden Protokolltypen

Entwicklungszeit-Ereignisprotokoll

Das Entwicklungszeit-Ereignisprotokoll enthält Ereignisse, die eintreten, wenn Sie die Datenprozessorumwandlung in der Ansicht **Daten-Viewer** ausführen. Zeigen Sie das Entwicklungszeit-Protokoll in der Ansicht **Ereignisse** an.

Laufzeit-Ereignisprotokoll

Das Laufzeit-Ereignisprotokoll enthält Ereignisse, die eintreten, wenn Sie die Datenprozessorumwandlung in einem Mapping ausführen. Sie können das Laufzeit-Ereignisprotokoll in einem Texteditor anzeigen oder ein Laufzeit-Ereignisprotokoll in die Ansicht **Ereignisse** der Datenprozessorumwandlung ziehen.

Benutzerprotokoll

Das Benutzerprotokoll enthält Ereignisse, die Sie für Komponenten in einem Skript konfigurieren. Die Datenprozessorumwandlung schreibt in das Benutzerprotokoll, wenn Sie sie aus der Ansicht **Daten-Viewer** in einem Mapping ausführen. Sie können das Benutzerprotokoll in einem Texteditor ansehen.

Entwicklungszeit-Ereignisprotokoll

Das Entwicklungszeit-Ereignisprotokoll enthält die Ereignisse, die bei der Ausführung der Datenprozessorumwandlung über den **Daten-Viewer** im Developer Tool auftreten.

Bei der Ausführung einer Datenprozessorumwandlung über die **Daten-Viewer**-Ansicht wird das Entwicklungszeit-Ereignisprotokoll in der **Datenprozessor-Ereignisse**-Ansicht angezeigt. Standardmäßig enthält das Entwicklungszeit-Ereignisprotokoll Benachrichtigungen, Warnungen und Fehlschläge. In den Umwandlungseinstellungen können Sie die Datenprozessorumwandlung so konfigurieren, dass ein oder mehrere Ereignistypen aus dem Protokoll ausgeschlossen sind.

Wenn Sie die Eingabedokumente mit den Protokollen speichern, können Sie ein Ereignis in der **Datenprozessor**-Ansicht anklicken, um den Ort im Eingabedokument zu finden, der das Ereignis generierte. Wenn Sie die Einstellungen der Datenprozessorumwandlung konfigurieren, können Sie die Eingabedateien entweder bei jeder Ausführung oder nur bei einem Fehlschlag speichern lassen.

Das Entwicklungszeit-Ereignisprotokoll heißt `events.cme`. Sie finden das Entwicklungszeit-Ereignisprotokoll für die letzte Ausführung der Datenprozessorumwandlung im folgenden Verzeichnis:

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

Die Datenprozessorumwandlung überschreibt das Entwicklungszeit-Ereignisprotokoll jedes Mal, wenn Sie die Umwandlung im **Daten-Viewer** ausführen. Wenn Sie das Entwicklungszeit-Ereignisprotokoll nach einem späteren Ausführen der Umwandlung anzeigen oder die Protokolle verschiedener Ausführungen vergleichen möchten, müssen Sie es umbenennen. Beim Schließen des Developer Tools speichert der Developer keine Dateien im

Laufzeit-Ereignisprotokoll

Das Laufzeit-Ereignisprotokoll zeichnet Ereignisse auf, die eintreten, wenn Sie die Datenprozessorumwandlung in einem Mapping ausführen.

Wenn die Datenprozessorumwandlung den Durchgang ohne Fehler abschließt, wird kein Ereignisprotokoll geschrieben. Wenn Fehler vorkommen, wird die Datenprozessorumwandlung ein zweites Mal durchgeführt. Während des zweiten Durchgangs wird ein Ereignisprotokoll geschrieben. Das Laufzeit-Ereignisprotokoll hat die Bezeichnung `events.cme`.

Auf einem Windows-Rechner befindet sich das Laufzeit-Ereignisprotokoll im folgenden Verzeichnis:

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

Auf einem Linux- oder UNIX-Rechner befindet sich das Laufzeit-Ereignisprotokoll für einen Root-Benutzer im folgenden Verzeichnis:

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

Auf einem Linux- oder UNIX-Rechner befindet sich das Laufzeit-Ereignisprotokoll für einen Nicht-Root-Benutzer im folgenden Verzeichnis:

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

Sie können den Standort des Laufzeit-Ereignisprotokolls mit dem Konfigurationseditor ändern.

Anzeigen eines Ereignisprotokolls in der Ansicht für Datenprozessor-Ereignisse

Zeigen Sie mithilfe der Ansicht **Datenprozessor-Ereignisse** Ereignisprotokolle zur Entwicklungszeit oder zur Laufzeit an.

Öffnen Sie Windows Explorer und suchen Sie nach der Ereignisprotokolldatei, die Sie anzeigen möchten. Ziehen Sie das Protokoll aus dem Windows Explorer-Fenster in die Ansicht **Datenprozessor-Ereignisse**. Klicken Sie mit der rechten Maustaste auf die Ansicht **Datenprozessor-Ereignisse** und wählen Sie anschließend **Suchen** aus, um das Protokoll zu suchen.

Hinweis: Um das aktuellste Ereignisprotokoll zur Entwicklungszeit neu zu laden, klicken Sie mit der rechten Maustaste auf die Ansicht **Datenprozessor-Ereignisse** und wählen Sie anschließend **Projektereignisse neu laden** aus.

Benutzerprotokoll

Das Benutzerprotokoll enthält benutzerdefinierte Meldungen, die Sie zu Fehlern von Komponenten in einem Skript konfigurieren.

Die Datenprozessorumwandlung schreibt Meldungen in das Benutzerprotokoll, wenn Sie ein Skript von der Ansicht **Daten-Viewer** aus ausführen und wenn Sie es in einem Mapping ausführen.

Wenn eine Skriptkomponente die Eigenschaft **on_fail** aufweist, können Sie es so konfigurieren, dass es eine Meldung in das Benutzerprotokoll schreibt, wenn es fehlschlägt. Legen Sie im Skript die Eigenschaft **on_fail** auf einen der folgenden Werte fest:

- LogInfo
- LogWarning
- LogError

Bei jeder Ausführung des Skripts wird ein neues Benutzerprotokoll erstellt. Der Dateiname des Benutzerprotokolls enthält den Umwandlungsnamen mit einer eindeutigen GUID.

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

Beispiel: CalculateValue_Aa93a9d14-a01f-442a-b9cb-c9ba5541b538.log

Auf einem Windows-Computer befindet sich das Benutzerprotokoll im folgenden Verzeichnis:

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

Auf einem Linux- oder UNIX-Computer befindet sich das Benutzerprotokoll für den Root-Benutzer im folgenden Verzeichnis:

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

Auf einem Linux- oder UNIX-Computer befindet sich das Benutzerprotokoll für einen Benutzer, der kein Root-Benutzer ist, im folgenden Verzeichnis:

```
home/<Installation_Dirctory>/DataTransformation/UserLogs
```

Entwicklung der Datenprozessorumwandlung

Verwenden Sie den Assistenten für neue Umwandlungen, um eine Datenprozessorumwandlung automatisch zu generieren, oder erstellen Sie eine leere Datenprozessorumwandlung und konfigurieren Sie sie später. Wenn Sie eine leere Datenprozessorumwandlung erstellen, müssen Sie ein Skript-, XMap-, Bibliotheks- oder Validierungsregelobjekt in der Umwandlung erstellen. Ein Skript kann Quelldokumente in ein hierarchisches Format parsen, das hierarchische Format in andere Dateiformate konvertieren oder ein hierarchisches Dokument einem anderen hierarchischen Format zuordnen. Eine XMap wandelt eine hierarchische Eingabedatei in eine hierarchische Ausgabedatei mit einer anderen Struktur um. Eine Bibliothek konvertiert einen branchenüblichen Nachrichtentyp in ein XML-Dokument mit einer Hierarchiestruktur oder aus XML in ein branchenübliches Standardformat. Wählen Sie die Schemata aus, die die Eingabe- oder Ausgabe-hierarchien definieren.

1. Erstellen Sie die Umwandlung im Developer Tool.
2. Führen Sie für eine leere Datenprozessorumwandlung die folgenden zusätzlichen Schritte aus:
 - a. Fügen Sie Schemareferenzen hinzu, die die Eingabe- oder Ausgabe-XML-Hierarchien definieren.
 - b. Erstellen Sie ein Skript-, XMap-, Bibliotheks- oder Validierungsregelobjekt.
3. Konfigurieren Sie die Eingabe- und Ausgabeports.
4. Testen Sie die Umwandlung.

Erstellen der Datenprozessorumwandlung

Erstellen Sie eine Datenprozessorumwandlung im Developer Tool. Wenn Sie eine leere Datenprozessorumwandlung erstellen, müssen Sie anschließend ein Skript-, XMap-, Bibliotheks- oder Validierungsregelobjekt in der Umwandlung erstellen. Alternativ können Sie den Assistenten für neue Umwandlungen verwenden, um eine Datenprozessorumwandlung automatisch zu erzeugen.

1. Klicken Sie im Developer-Tool auf **Datei > Neu > Umwandlung**.
2. Wählen Sie die Datenprozessorumwandlung aus und klicken Sie auf **Weiter**.
3. Geben Sie einen Namen für die Umwandlung ein und suchen Sie nach einem Modellrepository-Speicherort, um die Umwandlung abzulegen.

4. Geben Sie an, ob die Datenprozessorumwandlung mit einem Assistenten oder ob eine leere Datenprozessorumwandlung erstellt werden soll.
5. Wenn Sie das Erstellen einer leeren Datenprozessorumwandlung ausgewählt haben, klicken Sie auf **Fertig stellen**.
Das Developer-Tool erstellt die leere Umwandlung im Repository. Im Developer-Tool wird die Ansicht **Übersicht** angezeigt.
6. Wenn Sie das Erstellen einer Datenprozessorumwandlung mit einem Assistenten ausgewählt haben, führen Sie die folgenden Schritte aus:
 - a. Klicken Sie auf **Weiter**.
 - b. Wählen Sie ein Eingabeformat aus.
 - c. Suchen und wählen Sie ein Schema, ein Copybook, eine Beispiel- oder Spezifikationsdatei aus, falls dies für bestimmte Eingabeformate wie COBOL, JSON oder ASN.1 erforderlich ist.
 - d. Wählen Sie ein Ausgabeformat aus.
 - e. Suchen und wählen Sie ggf. ein Schema, ein Copybook, eine Beispiel- oder Spezifikationsdatei für das Ausgabeformat aus.
 - f. Klicken Sie auf **Fertig stellen**. Der Assistent erstellt die Umwandlung im Repository.
Die Umwandlung enthält möglicherweise einen Parser, Serializer, Mapper oder ein Objekt mit gemeinsamen Komponenten. Wenn Sie ein Schema, ein Copybook, eine Beispiel- oder Spezifikationsdatei ausgewählt haben, erstellt der Assistent ebenfalls ein Schema im Repository, das der Hierarchie in der Datei entspricht.

Auswählen der Schemaobjekte

Wählen Sie die Schemaobjekte aus, die die Eingabe- oder Ausgabe-hierarchien für die jeweilige XMap oder Skriptkomponente definieren, deren Erstellung Sie beabsichtigen.

In der Ansicht „Referenzen“ können Sie Schemareferenzen hinzufügen. Alternativ können Sie die Schemareferenzen hinzufügen, wenn Sie Skript- oder XMap-Objekte erstellen. Bevor Sie ein Schemaobjekt in einem Skript oder einer XMap referenzieren können, muss es im Modellrepository vorhanden sein

1. Klicken Sie in der Ansicht **Referenzen** der Datenprozessorumwandlung auf **Hinzufügen**.
2. Wenn das Schemaobjekt im Modellrepository vorhanden ist, suchen Sie das Schema und wählen Sie es aus.
3. Wenn das Schema im Modellrepository nicht vorhanden ist, klicken Sie auf **Neues Schemaobjekt erstellen** und importieren Sie ein Schemaobjekt aus einer hierarchischen Schemadatei.
4. Klicken Sie auf "Fertigstellen", um der Datenprozessorumwandlung die Schemareferenz hinzuzufügen.

Erstellen von Objekten in einer leeren Datenprozessorumwandlung

Erstellen Sie ein Skript, Bibliotheks-, XMap- oder Validierungsregelobjekt in der Ansicht **Objekte** der Umwandlung. Nach der Erstellung des Objekts können Sie das Objekt von der Ansicht **Objekte** aus öffnen, um es zu konfigurieren.

Erstellen eines Skripts

Erstellen Sie ein Skriptobjekt und definieren Sie den zu erstellenden Skript-Komponententyp. Optional können Sie eine Schemareferenz und eine Beispielquelldatei definieren.

1. Klicken Sie in der Ansicht **Objekte** der Datenprozessorumwandlung auf **Neu**.
2. Geben Sie einen Namen für das Skript ein und klicken Sie auf **Weiter**.
3. Wählen Sie die Erstellung eines Parsers oder Serializers aus. Wählen Sie „Andere“, um eine Mapper-, Transformer- oder Streamer-Komponente zu erstellen.
4. Geben Sie einen Namen für die Komponente ein.
5. Handelt es sich bei der Komponente um die erste Komponente, die Daten in der Umwandlung verarbeitet, aktivieren Sie **Als Startkomponente festlegen**.
6. Klicken Sie auf **Weiter**, wenn Sie eine Schemareferenz für dieses Skript eingeben möchten. Klicken Sie auf **Beenden**, wenn Sie die Schemareferenz nicht eingeben möchten.
7. Wenn Sie eine Schemareferenz erstellen, wählen Sie **Referenz einem Schemaobjekt hinzufügen** aus und suchen Sie nach dem Schemaobjekt im Modellrepository. Klicken Sie auf **Neues Schemaobjekt erstellen**, um ein Schemaobjekt im Modellrepository zu erstellen.
8. Klicken Sie auf **Weiter**, um eine Beispielquellenreferenz oder einen Beispieltext einzugeben. Klicken Sie auf **Beenden**, wenn Sie keine Beispielquelle definieren möchten.
Verwenden Sie eine Beispielquelle, um Beispieldaten zu definieren und das Skript zu testen.
9. Wenn Sie eine Beispielquelle auswählen, wählen Sie **Datei** aus und suchen Sie nach der Beispieldatei.
Sie können auch Beispieltext im **Text**-Bereich eingeben. Das Developer-Tool verwendet den Text zum Testen eines Skripts.
10. Klicken Sie auf **Fertig stellen**.
Die **Skript**-Ansicht wird im Developer Tool-Editor angezeigt.

Erstellen einer XMap

Erstellen Sie eine XMap in der Data Transformation-Ansicht **Objekt**. Beim Erstellen einer XMap benötigen Sie ein Schema, das die Dokumente für das Eingabe- und Ausgabehierarchieschema beschreibt. Sie wählen das Element in dem Schema aus, bei dem es sich um das Root-Element für die Eingabehierarchie handelt.

1. Klicken Sie in der Ansicht **Objekte** der Datenprozessorumwandlung auf **Neu**.
2. Wählen Sie "XMap" und klicken Sie auf **Weiter**.
3. Geben Sie den Namen für die XMap ein.
4. Handelt es sich bei der XMap-Komponente um die erste Komponente, die Daten in der Umwandlung verarbeitet, aktivieren Sie **Als Startkomponente festlegen**.
Klicken Sie auf **Weiter**.
5. Wenn Sie eine Schemareferenz erstellen, wählen Sie **Referenz einem Schemaobjekt hinzufügen** aus und suchen Sie nach dem Schemaobjekt im Modellrepository.
Um ein neues Schemaobjekt zu importieren, klicken Sie auf **Neues Schemaobjekt erstellen**.
6. Wenn Sie über eine Beispielhierarchiedatei verfügen, mit der Sie die XMap testen können, navigieren Sie zu dieser Datei und wählen Sie sie im Dateisystem aus.
Sie können die Beispielhierarchiedatei ändern.
7. Wählen Sie die Root als Eingabe-Hierarchie.

Wählen Sie im Dialogfeld **Auswahl des Root-Elements** ein Element im Schema aus, bei dem es sich um das Root-Element für die Eingabehierarchiedatei handelt. Sie können im Schema nach einem Element suchen. Sie können Muster suchen. Für die Übereinstimmung einer beliebigen Zeichenanzahl in der Zeichenfolge geben Sie `*<string>` ein. Für die Übereinstimmung eines einzelnen Zeichens geben Sie `<character>` ein.

8. Klicken Sie auf **Fertig stellen**.

Das Developer-Tool erstellt eine Ansicht für jede XMap, die Sie erstellen. Klicken Sie auf die Ansicht, um das Mapping zu konfigurieren.

Erstellen einer Bibliothek

Erstellen Sie ein Bibliotheksobjekt in der Data Transformation-Ansicht **Objekte**. Wählen Sie den Meldungstyp, die Komponente und den Namen aus. Optional können Sie eine Beispielmeldungstyp-Quelldatei definieren, die Sie zum Testen des Bibliotheksobjekts verwenden können.

Bevor Sie eine Bibliothek in der Datenprozessorumwandlung erstellen, installieren Sie das Bibliothekssoftwarepaket auf Ihrem Computer.

1. Klicken Sie in der Ansicht **Objekte** der Datenprozessorumwandlung auf **Neu**.
2. Wählen Sie "Bibliothek" aus und klicken Sie auf **Weiter**.
3. Suchen Sie den Meldungstyp und wählen Sie ihn aus.
4. Wählen Sie die Erstellung eines Parsers oder Serializers aus.

Erstellen Sie einen Parser, wenn die Eingabe des Bibliotheksobjekts ein Meldungstyp und die Ausgabe XML ist. Erstellen Sie einen Serializer, wenn die Eingabe des Bibliotheksobjekts XML und die Ausgabe ein Meldungstyp ist.

5. Handelt es sich bei der Bibliothek um die erste Komponente, die Daten in der Datenprozessorumwandlung verarbeitet, aktivieren Sie **Als Startkomponente festlegen**.

Klicken Sie auf **Weiter**.

6. Wenn Sie über eine Beispielmeldungstyp-Quelldatei verfügen, die Sie zum Testen der Bibliothek verwenden können, suchen Sie die Datei im System und wählen Sie sie aus.

Sie können die Beispieldatei ändern.

7. Klicken Sie auf **Fertig stellen**.

Das Developer-Tool erstellt eine Ansicht für jeden von Ihnen erstellten Meldungstyp. Klicken Sie auf die Ansicht, um auf das Mapping zuzugreifen.

Erstellen von Validierungsregeln

Erstellen Sie ein Validierungsregelobjekt in der Ansicht **Objekte** der Datenprozessorumwandlung.

1. Klicken Sie in der Ansicht **Objekte** der Datenprozessorumwandlung auf **Neu**.
2. Wählen Sie „Validierungsregeln“ aus und klicken Sie auf **Weiter**.
3. Geben Sie einen Namen für die Validierungsregeln ein.
4. Wenn Sie über eine XML-Beispieldatei verfügen, mit der Sie die Validierungsregeln testen können, navigieren Sie zu dieser Datei und wählen Sie sie im Dateisystem aus.

Sie können die Beispiel-XML-Datei wechseln.

5. Klicken Sie auf **Fertig stellen**.

Das Developer-Tool erstellt ein Validierungsregelobjekt und öffnet es im Validierungsregeleditor.

Hinzufügen einer Beispielquelle

Wählen Sie die Beispielquelle aus, um das Skript, die XMap, die Bibliothek oder die Validierungsregeln zu testen, die Sie erstellen möchten.

Sie können eine Beispielquelle hinzufügen, wenn Sie ein Skript, eine XMap, eine Bibliothek oder Validierungsregeln erstellen. Nach der Auswahl wird die Beispielquelle zum Modellrepository hinzugefügt. Aufgrund von Modellrepository-Beschränkungen ist die Größe der Beispielquelldatei auf 5 MB begrenzt.

Sie können die Beispielquelle ändern.

Erstellen von Ports

Konfigurieren Sie in der Ansicht **Übersicht** die Eingabe- und die Ausgabeports.

Wenn Sie weitere Eingabe- oder Ausgabeports in einem Skript konfigurieren, fügt das Developer-Tool der Umwandlung standardmäßig weitere Eingabe- und Ausgabeports hinzu. Es werden keine Eingabeports in der Ansicht **Übersicht** hinzugefügt.

1. Wenn Sie die Ausgabedaten anstelle einer XML in Zeilen zurückgeben möchten, aktivieren Sie die Option **Relationale Ausgabe**.
Bei Aktivierung der relationalen Ausgabe entfernt das Developer-Tool den standardmäßigen Ausgabeport.
2. Wählen Sie den Datentyp des Eingabeports, den Porttyp, die Gesamtstellenanzahl und die Größenordnung aus.
3. Wenn Sie keine relationalen Ausgabeports definieren, definieren Sie den Datentyp des Ausgabeports, den Porttyp, die Gesamtstellenanzahl und die Größenordnung.
4. Wenn ein Skript weitere Eingabeports aufweist, können Sie den Speicherort der Beispieleingabedatei für die Ports definieren. Klicken Sie im Feld **Eingabespeicherort** auf die Schaltfläche **Öffnen**, um nach der Datei zu suchen.
5. Wenn die relationale Ausgabe aktiviert ist, klicken Sie auf **Mapping-Ausgabe**, um die Ausgabeports zu erstellen.
6. Ordnen Sie in der Ansicht „Ports“ den Feldern im Bereich **Relationale Ports** Knoten aus dem Bereich **Hierarchische Ausgabe** zu.

Testen der Umwandlung

Testen Sie in der Ansicht **Daten-Viewer** die Datenprozessorumwandlung.

Überprüfen Sie vor dem Testen der Umwandlung, ob die Startkomponente definiert wurde. Sie können die Startkomponente in einem Skript definieren oder Sie können die Startkomponente auf der Registerkarte **Übersicht** auswählen. Es muss außerdem eine Beispieleingabedatei zum Testen ausgewählt sein.

1. Öffnen Sie die Ansicht **Daten-Viewer**.
2. Klicken Sie auf **Ausführen**.
Das Developer Tool validiert die Umwandlung. Wenn kein Fehler vorliegt, zeigt das Developer Tool die Beispieldatei im Bereich **Eingabe** an. Die Ausgabeergebnisse werden in der Maske „Ausgabe“ angezeigt.
3. Klicken Sie auf **Ereignisse anzeigen**, um die Ansicht **Datenprozessor-Ereignisse** anzuzeigen.
4. Doppelklicken Sie in der Ansicht **Datenprozessor-Ereignisse** auf ein Ereignis, um das Ereignis im Skripteditor zu debuggen.

5. Klicken Sie auf **Mit Editor synchronisieren**, um die Eingabedatei zu ändern, falls Sie mehrere Komponenten mit jeweils einer anderen Beispieleingabedatei testen.

Wenn der Inhalt der Beispieldatei im Dateisystem geändert wird, werden die Änderungen im Bereich **Eingabe** angezeigt.

Export und Import von Datenprozessor-Umwandlungen

Sie können eine Datenprozessorumwandlung als Dienst exportieren und über ein Datenumwandlungs-Repository ausführen. Sie können auch einen Data Transformation-Dienst in das Developer-Tool importieren. Wenn Sie einen Data Transformation-Dienst importieren, erstellt das Developer-Tool eine Datenprozessorumwandlung aus dem Dienst.

Hinweis: Wenn Sie einen Data Transformation-Dienst in das Modellrepository importieren, importiert das Developer-Tool die zugehörigen Schemata in das Repository. Wenn Sie das Schema im Repository ändern, erscheinen die Änderungen nicht sofort in den Umwandlung-Schemareferenzen. Sie können die Verbindung zum Modellrepository schließen und öffnen oder das Developer Tool schließen und öffnen, damit die Schemaänderungen in der Umwandlung erscheinen.

Exportieren der Datenprozessorumwandlung als Dienst

Sie können die Datenprozessorumwandlung als Data Transformation-Dienst exportieren. Exportieren Sie den Dienst in das Dateisystem-Repository der Maschine, in der sie den Dienst ausführen möchten. Sie können den Dienst mit PowerCenter, benutzerdefinierten Anwendungen oder dem Befehl Data Transformation CM_console ausführen.

1. Klicken Sie in der Ansicht **Objekt-Explorer** mit der rechten Maustaste auf die zu exportierende Datenprozessorumwandlung, und wählen Sie **Exportieren** aus.
Das Dialogfeld **Export** wird eingeblendet.
2. Wählen Sie **Informatica > Datenprozessorumwandlung exportieren** aus und klicken Sie auf **Weiter**.
Die Seite **Auswählen** erscheint.
3. Klicken Sie auf **Weiter**.
Die Seite **Dienstnamen und Zielordner auswählen** erscheint.
4. Wählen Sie einen Zielordner aus:
 - Um den Dienst auf der Maschine zu exportieren, auf dem das Developer Tool gehostet wird, klicken Sie auf **Dienstordner**.
 - Um den Dienst auf einer anderen Maschine bereitzustellen, klicken Sie auf **Ordner**. Navigieren Sie zum Verzeichnis `\ServiceDB` auf der Maschine, auf der Sie den Dienst bereitstellen möchten.
5. Klicken Sie auf **Fertigstellen**.

Importieren mehrerer Data Transformation-Dienste

Sie können ein Verzeichnis mit Data Transformation-Diensten aus dem Computer importieren, auf dem das Verzeichnis gespeichert ist. Wenn Sie Data Transformation-Dienste in das Developer-Modellrepository importieren, importiert das Developer Tool die Umwandlungen, Schemas und Beispieldaten mit den CMW-

Dateien. Wenn Sie zahlreiche Dienste importieren müssen, importieren Sie ein Verzeichnis mit Diensten anstelle nur eines Diensts.

1. Klicken Sie auf **Datei > Importieren**.
Das Dialogfeld **Importieren** wird eingeblendet.
2. Wählen Sie **InformaticaData Transformation-Dienste importieren (Ordner)** aus und klicken Sie auf **Weiter**.
Die Seite **Data Transformation-Dienst importieren** wird angezeigt.
3. Navigieren Sie zu dem Verzeichnis, das Sie importieren möchten.
4. Navigieren Sie zu einem Speicherort in dem Repository, in dem Sie die Umwandlungen speichern möchten. Klicken Sie dann auf **Fertig stellen**.
Das Developer Tool importiert die Umwandlungen, Schemas und Beispieldaten mit der **CMW**-Datei.

Importieren eines Data Transformation-Diensts

Sie können die CMW-Datei eines Data Transformation-Diensts in das Modellrepository importieren, um eine Datenprozessor-Umwandlung zu erstellen. Das Developer-Tool importiert Umwandlungs-, Schema- und Beispieldaten mit der **.cmw**-Datei.

1. Klicken Sie auf **Datei > Importieren**.
Das Dialogfeld **Importieren** wird eingeblendet.
2. Wählen Sie **InformaticaData Transformation-Dienst importieren (Einzel)** aus und klicken Sie auf **Weiter**.
Die Seite **Data Transformation-Dienst importieren** wird angezeigt.
3. Navigieren Sie zur Dienstdatei mit der Erweiterung **.cmw**, die Sie importieren möchten.
Das Developer-Tool benennt die Umwandlung entsprechend des Dienstdateinamens. Sie können den Namen ändern.
4. Navigieren Sie zu einem Speicherort im Repository, wo Sie die Umwandlung speichern möchten. Dann klicken Sie auf **Fertig stellen**.
Das Developer-Tool importiert die Umwandlungs-, Schema- und Beispieldaten mit der **.cmw**-Datei.
5. Zum Bearbeiten der Umwandlung doppelklicken Sie in der Ansicht **Objekt-Explorer** auf die Umwandlung.

Exportieren eines Mappings mit einer Datenprozessorumwandlung nach PowerCenter

Wenn Sie ein Mapping mit einer Datenprozessorumwandlung in PowerCenter exportieren, können Sie die Objekte in eine lokale Datei exportieren und anschließend das Mapping in PowerCenter importieren. Alternativ können Sie das Mapping direkt in das PowerCenter-Repository exportieren.

1. Wählen Sie in der Ansicht **Objekt-Explorer** das zu exportierende Mapping aus. Klicken Sie mit der rechten Maustaste und wählen Sie **Exportieren** aus.
Das Dialogfeld **Export** wird eingeblendet.
2. Wählen Sie **Informatica > PowerCenter**.
3. Klicken Sie auf **Weiter**.
Das Dialogfeld **Export an PowerCenter** wird eingeblendet.
4. Wählen Sie das Projekt.
5. Wählen Sie die PowerCenter-Version.

6. Wählen Sie den Exportspeicherort, eine PowerCenter-Import-XML-Datei oder ein PowerCenter-Repository.
7. Geben Sie die Exportoptionen an.
8. Klicken Sie auf **Weiter**.
Sie werden im Developer-Tool aufgefordert, die Objekte für den Export auszuwählen.
9. Wählen Sie die zu exportierenden Objekte und klicken Sie auf **Fertig stellen**.
Das Developer-Tool exportiert die Objekte an den von Ihnen gewählten Speicherort. Wenn Sie das Mapping an einen Speicherort exportiert haben, exportiert das Developer-Tool auch die Datenprozessorumwandlungen im Mapping, wie z. B. Dienste, in einen Ordner an dem Speicherort, den Sie angegeben haben.
10. Wenn Sie das Mapping in ein PowerCenter-Repository exportiert haben, werden die Dienste in den folgenden Verzeichnispfad exportiert: %temp%\DTServiceExport2PC\
Die Exportfunktion erstellt einen separaten Ordner für jeden Dienst mit dem folgenden Namen:
`<date><serviceFullName>`
Wenn die Umwandlung relationale Mappings enthält, wird ein Ordner für relationale zu hierarchischen Mappings und ein separater Ordner für hierarchische zu relationalen Mappings erstellt.
11. Kopieren Sie den Ordner oder die Ordner mit Datenprozessorumwandlungsdiensten von dem lokalen Speicherort, wo Sie die Dateien in den PowerCenter ServiceDB-Ordner exportiert haben.
12. Wenn Sie das Mapping in eine PowerCenter-XML-Datei importiert haben, importieren Sie das Mapping in PowerCenter. Weitere Informationen über das Importieren eines Objekts in PowerCenter finden Sie im *PowerCenter 9.6.0 Repository-Handbuch*.

Datenprozessorumwandlung Validierung

Nachdem Sie eine Datenprozessorumwandlung als Dienst exportiert haben, können Sie VRL-Validierungen für den Dienst über das Datenumwandlungs-Repository ausführen.

Sie können eine geschwindigkeitsverbesserte Datenumwandlungs-Engine für VRL-Validierungen verwenden. Die geschwindigkeitsverbesserte Datenumwandlungs-Engine unterstützt die folgenden VRL-Funktionen:

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

Die geschwindigkeitsverbesserte Datenumwandlungs-Engine erzeugt die Ausgabe **ValidateValue**. **ValidateValue** enthält die Eigenschaft `max_error_count` mit einem Standardwert von 200 Fehlern. Wenn die Anzahl der Fehler `max_error_count` überschreitet, wird die Validierung gestoppt.

Hinweis: Die VRL-Syntax der geschwindigkeitsverbesserten Datenumwandlungs-Engine unterstützt das Tag `<list>` nicht.

Verwenden einer geschwindigkeitsverbesserten Datenumwandlungs-Engine für VRL-Validierungen

Nachdem Sie einen Datenumwandlungsdienst exportiert haben, können Sie eine geschwindigkeitsverbesserte Datenumwandlungs-Engine für VRL-Validierungen mit dem Dienst verwenden.

- Setzen Sie das folgende Flag in der .cmw-Datei des Dienstes: `optimize_vrl`.

Fügen Sie das Flag `optimize_vrl` zur Instanz `ServiceConfigProf` hinzu, wie im folgenden Beispiel gezeigt:

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

Datenprozessorumwandlung in einer nicht nativen Umgebung

Die Verarbeitung der Datenprozessorumwandlung in einer nicht nativen Umgebung hängt von der Engine ab, die die Umwandlung ausführt.

Ziehen Sie die Unterstützung für die folgenden nicht nativen Laufzeit-Engines in Betracht:

- Blaze-Engine Unterstützt ohne Einschränkungen.
- Spark-Engine Wird mit Einschränkungen in Batch-Mappings unterstützt. Wird in Streaming-Mappings nicht unterstützt.*
- Databricks-Spark-Engine Nicht unterstützt.

* Informationen zur Unterstützung der Datenprozessorumwandlung auf der Spark-Engine finden Sie im [KB article](#).

KAPITEL 3

Assistent für Eingabe- und Ausgabeformate

Dieses Kapitel umfasst die folgenden Themen:

- [Assistent für Eingabe- und Ausgabeformate – Übersicht, 48](#)
- [Avro, 49](#)
- [COBOL-Verarbeitungsbibliothek, 54](#)
- [JSON, 57](#)
- [Parquet, 60](#)
- [XML, 62](#)

Assistent für Eingabe- und Ausgabeformate – Übersicht

Verwenden Sie einen Assistenten zum Erstellen einer automatisch generierten Datenprozessor-Umwandlung mit Eingabe- und Ausgabeformaten wie COBOL, XML, relational oder JSON. Sie können den Assistenten auch zum Umwandeln von benutzerdefinierten Formaten verwenden.

Erstellen Sie eine Datenprozessor-Umwandlung und wählen Sie die Eingabe- und Ausgabeformate mit dem Assistenten für Datenprozessor-Umwandlung aus. Treffen Sie Ihre Auswahl aus den vorhandenen Formaten oder erstellen Sie benutzerdefinierte Formate. Fügen Sie für bestimmte Formate wie XML, JSON oder COBOL ein Schema, eine Spezifikationsdatei, eine Beispieldatei oder ein Copybook hinzu, die bzw. das die erwartete Struktur für die Eingabe oder Ausgabe definiert.

Der Assistent erstellt eine Umwandlung mit relevanten Skript-, XMap- oder Bibliotheksobjekten, die als Vorlagen zum Umwandeln des Eingabeformats in das Ausgabeformat dienen. Die Datenprozessor-Umwandlung erstellt eine Umwandlungslösung entsprechend der ausgewählten Formate und der Spezifikationsdatei, der Beispieldatei oder des Copybooks. Die Umwandlung ist möglicherweise nicht vollständig, enthält jedoch Bausteine, die Sie verbinden und anpassen, um die Umwandlungsdefinition abzuschließen.

Avro

Verwenden Sie den Assistenten zum Erstellen einer Umwandlung mit Avro-Eingabe oder -Ausgabe. Wenn Sie eine Datenprozessor-Umwandlung erstellen, um das Avro-Format umzuwandeln, wählen Sie ein Avro-Schema oder eine Beispieldatei aus, die die erwartete Struktur der Avro-Daten definiert. Der Assistent erstellt Komponenten, die das Avro-Format in andere Formate oder aus anderen Formaten in das Avro-Format umwandelt. Diese Komponenten können ein relationales-hierarchisches Mapping, ein hierarchisches-relationales Mapping und eine XMap enthalten. Nachdem der Assistent die Umwandlung erstellt hat, können Sie die Umwandlung weiter konfigurieren, um die Mapping-Logik zu bestimmen.

Apache Avro ist ein Datenserialisierungssystem im binären Format oder in anderen Datenformaten. Avro-Daten liegen in einem Format vor, das möglicherweise nicht direkt lesbar ist. Weitere Informationen zu Avro finden Sie unter <http://avro.apache.org/>

Hinweis: Verwenden Sie binär kodiertes Avro, um eine Umwandlung mit Avro-Eingabe oder -Ausgabe zu erstellen. Avro-Eingabe oder -Ausgabe in anderen Formaten kann nicht verarbeitet werden.

Eine Umwandlung, die Avro-Eingabe oder -Ausgabe liest, bezieht sich auf ein Schema. Wenn die Umwandlung Avro-Daten liest oder schreibt, verwendet sie das Schema zum Interpretieren der Hierarchie.

Wenn Sie eine Beispieldatei zum Definieren der Avro-Hierarchie auswählen, speichert der Assistent auch den ersten Datensatz in der Datei als separate Testdatei. Sie können diese Datei zum Testen der Umwandlung verwenden. Um die Datei zu suchen, überprüfen Sie im Bereich **Ports** der Ansicht **Übersicht** den im Feld **Eingabespeicherort** aufgelisteten Dateipfad.

Wenn Sie eine Datenprozessor-Umwandlung erstellen, die Avro in ein hierarchisches Format oder ein hierarchisches Format in Avro umwandelt, erstellt der Assistent eine XMap-Komponente in der Umwandlung. Der XMap-Editor zeigt die hierarchischen Schemaknoten und die Avro-Schemaknoten an. Verwenden Sie den XMap-Editor, um die Knoten zu verknüpfen und die Umwandlungslogik zu definieren. Weitere Informationen zum XMap-Objekt und -Editor finden Sie unter [“Übersicht über XMap” auf Seite 84](#).

Wenn Sie eine Umwandlung erstellen, die Avro in ein relationales Format oder ein relationales Format in Avro umwandelt, erstellt der Assistent ein relationales Mapping. Im Bereich **Ports** der Ansicht **Übersicht** werden die hierarchischen Avro-Schemaknoten und die relationalen Ports angezeigt. Verwenden Sie den Bereich **Ports**, um die hierarchischen Elemente mit den relationalen Ports und Gruppen zu verknüpfen. Weitere Informationen zum Umwandeln der relationalen Daten finden Sie unter [“Relationale Eingabe und Ausgabe – Übersicht” auf Seite 64](#).

Nachdem Sie eine Datenprozessor-Umwandlung für Avro-Eingabe erstellt haben, fügen Sie sie zu einem Mapping mit einem komplexen Datei-Reader hinzu. Der komplexe Datei-Reader übergibt Avro-Eingabe an die Umwandlung. Für eine Datenprozessor-Umwandlung mit Avro-Ausgabe fügen Sie einen komplexen Datei-Writer zum Mapping hinzu, um die Ausgabe aus der Umwandlung zu erhalten.

Avro-Eingabe und komplexer Datei-Reader

Damit die Datenprozessor-Umwandlung Avro-Eingabe umwandeln kann, empfängt die Umwandlung Dateneingabe aus einem komplexen Datei-Reader-Objekt. Nach dem Erstellen und Konfigurieren der Umwandlung fügen Sie die Umwandlung zu einem Mapping hinzu und verbinden den Eingabeport mit dem Ausgabeport des komplexen Datei-Readers.

Der komplexe Datei-Reader bietet Eingabe für eine Streamer-Komponente, die der Assistent für die Datenprozessor-Umwandlung als Teil der Umwandlung erstellt. Wenn die Umwandlung Avro-Eingabe in ein benutzerdefiniertes oder relationales Format konvertiert, müssen die Streamer-Einstellungen nicht geändert werden. Sie geben ein benutzerdefiniertes Ausgabeformat an, indem Sie **Andere** als Ausgabeformat im Assistenten auswählen.

Wenn die Umwandlung Avro-Eingabe in JSON-, XML- oder Avro-Format umwandelt, erstellt der Assistent eine XMap, um das Ausgabeformat zuzuordnen. Sie müssen die XMap im Streamer angeben, damit die Daten bei der Umwandlung ordnungsgemäß verarbeitet werden.

Außerdem müssen Sie den komplexen Datei-Reader zur Verarbeitung von Avro-Eingabe konfigurieren. Der Ausgabeport für den komplexen Datei-Reader sollte auf das Binärformat eingestellt sein. Auch sollte der Eingabeport für die Datenprozessor-Umwandlung auf binäre Eingabe gesetzt werden.

Avro-Datenkomprimierung mit dem Snappy-Codec

Sie können Avro-Daten mit dem komplexen Datei-Reader komprimieren. Wenn Sie den Snappy-Codec für Avro-Datenkomprimierung verwenden, müssen Sie die JAR-Datei für den Snappy-Codec aktualisieren, um die Umwandlung zu testen oder auszuführen.

Um den Snappy-Codec zu verwenden, ersetzen Sie die JAR-Standarddatei für Snappy bei der Informatica-Serverinstallation und in der Hadoop-Umgebung durch die aktualisierte Version. Die aktualisierte `snappy-java-1.0.4.1.jar`-Datei ist über folgenden Link verfügbar:

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

Aktualisieren des Snappy-Codec zur Aktivierung der Avro-Datenkomprimierung

Um die Avro-Datenkomprimierung mit dem Snappy-Codec zu aktivieren, ersetzen Sie die JAR-Standarddatei für Snappy durch die aktualisierte Version.

1. Ersetzen Sie auf dem Computer, auf dem Sie den Informatica-Server installiert haben, die Datei `snappy-java-1.0.4.1.jar` bei der Serverinstallation durch die Datei `snappy-java-1.1.0.1.jar`. Ersetzen Sie die JAR-Datei an folgendem Speicherort: `<Server_Installation>\services\shared\hadoop\<Hadoop_Distribution>\lib`
2. Ersetzen Sie auf dem Computer, auf dem Sie Hadoop installiert haben und ausführen, die Datei `snappy-java-1.0.4.1.jar` durch die Datei `snappy-java-1.1.0.1.jar`. Ersetzen Sie die JAR-Datei an folgendem Speicherort: `<Hadoop_rpm>\services\shared\hadoop\<Hadoop_Distribution>\lib`

Konfigurieren einer Umwandlung mit Avro-Eingabe

Um eine Datenprozessor-Umwandlung für Avro-Eingabe zu erstellen, verwenden Sie den Assistenten für Datenprozessor-Umwandlung. Der Assistent erstellt die Umwandlung im Modellrepository mit den Komponenten, die zur Umwandlung der Avro-Eingabe erforderlich sind. Verwenden Sie den IntelliScript-Editor, um den Streamer zu bearbeiten, und den XMap-Editor, um eine XMap zu bearbeiten (falls in der Umwandlung enthalten). Fügen Sie die Umwandlung einem Mapping mit einem komplexen Datei-Reader hinzu.

1. Erstellen Sie die Datenprozessor-Umwandlung mit dem Assistenten für neue Umwandlungen. Fügen Sie ein Avro-Schema oder eine Beispieldatei hinzu, die die erwartete Eingabestruktur definiert.
2. Wenn die Umwandlung ein XML-, JSON- oder ein anderes strukturiertes Ausgabeformat aufweist, verwenden Sie den XMap-Editor, um die XMap in der Umwandlung zu bearbeiten.
3. Verwenden Sie den IntelliScript-Editor, um den Streamer in der Umwandlung zu bearbeiten und anzupassen. Wenn die Umwandlung ein XML-, JSON- oder ein anderes strukturiertes Ausgabeformat aufweist, bearbeiten Sie den Streamer, um die XMap in der Umwandlung anzugeben.
4. Fügen Sie die Datenprozessor-Umwandlung einem Mapping mit einem komplexen Datei-Reader hinzu. Die Umwandlung sollte auf binäre Eingabe eingestellt bleiben, dies ist die Standardeinstellung für Avro-Eingabe. Konfigurieren Sie den komplexen Datei-Reader, um Avro zu verarbeiten. Die AusgabeEinstellung bleibt auf binär eingestellt, dies ist die Standardeinstellung. Verknüpfen Sie den Ausgabeport für den komplexen Datei-Reader mit dem Eingabeport für die Datenprozessor-Umwandlung, um Avro-Eingabe für die Umwandlung zu ermöglichen.

Schritt 1. Erstellen einer Umwandlung, die Avro umwandelt

Erstellen Sie eine Datenprozessor-Umwandlung mit Avro-Eingabe, Avro-Ausgabe oder beidem.

1. Klicken Sie im Developer-Tool auf **Datei > Neu > Umwandlung**.
2. Wählen Sie die Datenprozessor-Umwandlung aus und klicken Sie auf **Weiter**.
3. Geben Sie einen Namen für die Umwandlung ein und suchen Sie nach einem Speicherort für das Modellrepository, an dem die Umwandlung abzulegen ist.
4. Wählen Sie **Datenprozessor mit einem Assistenten erstellen** aus und klicken Sie auf **Weiter**.
5. Wählen Sie Avro- oder ein anderes Eingabeformat aus und klicken Sie auf **Weiter**.
6. Wenn Sie Avro als Eingabeformat ausgewählt haben, suchen und wählen Sie eine XSD-Schemadatei oder Avro-Beispieldatei aus. Klicken Sie auf **Weiter**.

Developer fügt dem Modellrepository eine XSD-Schemadatei hinzu, die die Avro-Hierarchie darstellt. Wenn Sie eine Beispieldatei auswählen, erstellt Developer eine Testdatei aus dem ersten Datensatz in der Beispieldatei. Sie können diese Datei zum Testen der Umwandlung verwenden. Um die Datei zu suchen, überprüfen Sie im Bereich **Ports** der Ansicht **Übersicht** den im Feld **Eingabespeicherort** aufgelisteten Dateipfad.

7. Wählen Sie Avro oder ein anderes Ausgabeformat aus und klicken Sie auf **Weiter**.
8. Wenn Sie Avro als Ausgabeformat auswählen, suchen und wählen Sie das zugehörige Schema oder eine Avro-Beispieldatei aus. Klicken Sie auf **Weiter**.
9. Klicken Sie auf **Fertig stellen**.

Das Developer-Tool erstellt die Umwandlung in dem Repository mit den zugehörigen Komponenten, wie zum Beispiel einer XMap, um die Avro-Hierarchie in ein anderes Hierarchieformat umzuwandeln. Im Developer-Tool wird die Ansicht **Übersicht** angezeigt.

10. Um die Komponenten in der Umwandlung zu bearbeiten, doppelklicken Sie in der Ansicht **Objekte** auf die Umwandlungskomponente, um sie im entsprechenden Editor zu öffnen.

Schritt 2. Bearbeiten der XMap

Um ein XMap-Objekt einer Datenprozessor-Umwandlung zu konfigurieren, fügen Sie Mapping-Anweisungen hinzu.

1. Um den XMap-Editor zu öffnen, klicken Sie in der Datenprozessor-Umwandlung unter **Objekte** auf das XMap-Objekt.
2. Um eine Map-, Group- oder Repeating Group-Mapping-Anweisung zu erstellen, führen Sie im XMap-Editor ein Drag & Drop von einem Knoten im Eingabehierarchieschema zu einem Knoten im Ausgabehierarchieschema durch.

Der XMap-Editor erstellt eine Mapping-Verknüpfung zwischen den Knoten. Die Mapping-Anweisung wird im Gitter angezeigt. Der XMap-Editor füllt die Felder für die Mapping-Anweisung automatisch aus.

3. Um eine bedingte Logik im Gitter zu erstellen, fügen Sie eine Router-Mapping-Anweisung wie folgt hinzu:
 - a. Erstellen Sie unter der Router-Mapping-Anweisung Option-Mapping-Anweisungen. Ziehen Sie Eingabe- und Ausgabeschemaknoten und fügen Sie sie in Option-Anweisungsfelder im Gitter ein.
 - b. Erstellen Sie unter der Router-Mapping-Anweisung eine Default-Mapping-Anweisung, um festzulegen, was passiert, wenn keine Option-Mapping-Anweisung angewendet wird.
 - c. Erstellen Sie unter der Option-Mapping-Anweisung Map-Mapping-Anweisungen, um Bedingungen für das Zuordnen der Eingabeknoten zu den Ausgabeknoten festzulegen.
4. Um gemeinsamen Kontext für eine Gruppe von Anweisungen bereitzustellen, fügen Sie eine Group-Mapping-Anweisung hinzu. Nest Map-Mapping-Anweisungen unter der Group-Mapping-Anweisung.

5. Um ein anderes XMap-Objekt aufzurufen, fügen Sie eine Run XMap-Anweisung hinzu.
6. Um den Kontext und die Logik für eine Mapping-Anweisung zu ändern, bearbeiten Sie die Eigenschaften der Mapping-Anweisung wie folgt:
 - a. Stufen Sie Anweisungen als untergeordnete Anweisungen tiefer oder Anweisungen als übergeordnete Anweisungen höher.
 - b. Erstellen Sie XPath-Ausdrücke, um den Kontext zu ändern, oder fügen Sie Vorhersagen mit dem XPath-Editor hinzu.

Schritt 3. Konfigurieren des Streamers

Wenn die Umwandlung XML-, JSON- oder ein anderes strukturiertes Ausgabeformat aufweist, erstellt der Assistent eine Streamer-Komponente und ein XMap-Objekt im Assistenten. Bearbeiten Sie den Streamer, um die XMap in der Umwandlung anzugeben.

1. Doppelklicken Sie in der Ansicht **Objekte** auf den Streamer bzw. ein Skriptobjekt, um es im IntelliScript-Editor zu öffnen.
2. Um den Streamer zum Angeben der XMap in der Umwandlung zu konfigurieren, führen Sie die folgenden Schritte aus:
 - a. Um das erforderliche Element zu konfigurieren, erweitern Sie das **contains**-Element im IntelliScript-Editor. Doppelklicken Sie auf den Doppelpfeile nach rechts >> neben dem Element.
 - b. Erweitern Sie das Element **repeating_segment**. Doppelklicken Sie auf den Doppelpfeile nach rechts >> neben dem Element.
 - c. Wählen Sie im Element **run_component** das jeweilige XMap-Objekt aus der Liste der Komponenten aus.
3. Um den Streamer zum Lesen der Daten aus Speicherorten im Quelldokument und zum Schreiben der Daten in XML zu konfigurieren, klicken Sie mit der rechten Maustaste auf
4. Um den Streamer zusätzlich zu konfigurieren, schachteln Sie innerhalb des **Streamer**-Elements die Komponenten **ComplexSegment** und **SimpleSegment** gemäß der Quellstruktur.
5. Definieren Sie für jede **SimpleSegment**-Komponente die öffnenden und schließenden Marker, sofern erforderlich. Definieren Sie die Umwandlung, die das Segment verarbeitet.

Schritt 4. Konfigurieren des komplexen Datei-Readers

Fügen Sie eine Datenprozessorumwandlung hinzu, die Avro in ein Mapping mit einem komplexen Datei-Reader umwandelt. Konfigurieren Sie den komplexen Datei-Reader, um Avro-Eingabe zu verarbeiten.

1. Erstellen Sie im Mapping-Editor ein komplexes Datei-Reader-Objekt.
2. Um den komplexen Datei-Reader zu konfigurieren, führen Sie die folgenden Schritte aus:
 - a. Wählen Sie auf der Registerkarte **Erweitert** in der Ansicht **Eigenschaften** die Eigenschaft **Dateiformat** und dann **Benutzerdefinierte Eingabe** aus.
 - b. Wählen Sie die Eigenschaft **Eingabeformat** aus und geben Sie dann `com.informatica.avro.AvroToXML` ein.
 - c. Verwenden Sie zur Leistungsoptimierung die Eigenschaft **Eingabeformatparameter** und passen Sie den Parameter `MaxOutputAccumulation` an. Standardmäßig ist der Parameter `MaxOutputAccumulation`, der die erwartete Anzahl an Ausgabedatensätzen definiert, auf 50.000 festgelegt. Um die Einstellung beispielsweise in 250.000 zu ändern, geben Sie `"MaxOutputAccumulation"="250000"` ein.

- d. Der komplexe Datei-Reader fügt das Schema standardmäßig zur Ausgabe des komplexen Datei-Readers innerhalb eines einzelnen Elements direkt nach dem Root-Element hinzu. Wenn Sie das Schema nicht zur Ausgabe hinzufügen möchten, wählen Sie die Eigenschaft **Eingabeformatparameter** aus und geben dann `"InjectSchema"="false"` ein.

Verwenden Sie ein Semikolon zum Trennen mehrerer Parameter, z. B.

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

3. Fügen Sie die Datenprozessorumwandlung zur Zuordnung hinzu. Der Umwandlungseingabeport sollte auf binäre Eingabe eingestellt bleiben, dies ist die Standardeinstellung für Avro-Eingabe.
4. Verknüpfen Sie den Ausgabeport für den komplexen Datei-Reader mit dem Eingabeport für die Datenprozessorumwandlung. Der Ausgabeport für den komplexen Datei-Reader sollte auf binäre Ausgabe eingestellt bleiben.

Konfigurieren einer Umwandlung mit Avro-Ausgabe

Um eine Datenprozessor-Umwandlung für Avro-Ausgabe zu erstellen, verwenden Sie den Assistenten für Datenprozessor-Umwandlung. Der Assistent erstellt die Umwandlung im Modellrepository mit den Komponenten, die für die Umwandlung der Avro-Ausgabe erforderlich sind. Verwenden Sie den XMap-Editor zum Bearbeiten der XMap (falls enthalten). Fügen Sie die Umwandlung zu einem Mapping mit einem komplexen Datei-Writer hinzu.

1. Erstellen Sie die Datenprozessor-Umwandlung mit dem Assistenten für neue Umwandlungen. Fügen Sie ein Avro-Schema oder eine Beispieldatei hinzu, die die erwartete Ausgabestruktur definiert.
2. Wenn die Umwandlung ein XML-, JSON- oder ein anderes strukturiertes Eingabeformat aufweist, verwenden Sie den XMap-Editor, um die XMap, die der Assistent in der Umwandlung erstellt hat, zu bearbeiten und anzupassen.
3. Um die Ausgabeeinstellungen für die Datenprozessorumwandlung zu konfigurieren, wählen Sie im Bereich **Ausgabesteuerung** in der Ansicht **Einstellungen** die entsprechende Ausgabeoption im Bereich **Binärer Ausgabeport: Auflistungsmodus** aus.

Hinweis: Um die Ausgabe so zu konfigurieren, dass ein Datensatz gleichzeitig mit dem Schema ausgewählt werden kann, wählen Sie die Option **Ausgabezeile für jede Zeile** aus. Um alle Datensätze mit dem Schema in einem Ausgabestream zu sammeln, wählen Sie die Option **Eingabezeilen in einer einzigen Ausgabe auflisten** aus.

4. Fügen Sie die Datenprozessor-Umwandlung zu einem Mapping hinzu. Die Ausgabeeinstellung der Umwandlung bleibt auf binär eingestellt, dies ist die Standardeinstellung für Avro-Ausgabe.
5. Erstellen und verknüpfen Sie einen komplexen Datei-Writer mit der Datenprozessor-Umwandlung im Mapping, um die Avro-Ausgabe aus der Umwandlung zu erhalten.

Hinweis: Die Eingabeeinstellung bleibt auf binär eingestellt, dies ist die Standardeinstellung. Der komplexe Datei-Writer unterstützt keine Kompression für Avro-Ausgabe.

6. Konfigurieren Sie den komplexen Datei-Writer. Wählen Sie im Datenobjektvorgang auf der Registerkarte **Erweitert** für das **Dateiformat** die Option **Benutzerdefinierte Ausgabe** aus. Geben Sie für **Ausgabeformat** die Zeichenfolge `com.informatica.avro.XMLToAvro` ein.

COBOL-Verarbeitungsbibliothek

Die COBOL-Bibliothek wandelt COBOL-Daten in und aus XML um. Wenn Sie mit dem Assistenten eine Umwandlung mit COBOL-Eingabe oder -Ausgabe erstellen, wählen Sie ein COBOL-Copybook aus, um die erwartete Struktur der Eingabe- oder Ausgabedaten zu definieren.

Wenn Sie eine Datenprozessor-Umwandlung mit COBOL-Eingabe oder -Ausgabe mit dem Assistenten erstellen, fügt Developer die folgenden Objekte zur Umwandlung hinzu:

- Ein Schema-Objekt, das eine XML-Darstellung der COBOL-Datenstruktur definiert.
- Für COBOL-Eingabe fügt Developer einen Parser hinzu, der Eingabedaten aus der COBOL-Datendefinition in XML umwandelt.
- Für COBOL-Ausgabe fügt Developer einen Serializer hinzu, der XML in COBOL umwandelt.

Hinweis: Sie können eine Datenprozessor-Umwandlung erstellen, die COBOL-Eingabe oder -Ausgabe, aber nicht beides verwendet. Zur Verarbeitung von EBCDIC-codiertem COBOL stellen Sie sicher, dass die Codierungseinstellungen für die Datenprozessor-Umwandlung in EBCDIC geändert werden.

Erstellen einer Umwandlung für COBOL

Verwenden Sie den Assistenten für Datenprozessor-Umwandlung, um eine Datenprozessor-Umwandlung mit COBOL-Eingabe oder -Ausgabe zu erstellen.

1. Klicken Sie im Developer Tool auf **Datei > Neu > Umwandlung**.
2. Wählen Sie die Datenprozessorumwandlung aus und klicken Sie auf **Weiter**.
3. Geben Sie einen Namen für die Umwandlung ein und suchen Sie nach einem Speicherort für das Modellrepository, an dem die Umwandlung abzulegen ist.
4. Wählen Sie **Datenprozessor mit einem Assistenten erstellen** aus und klicken Sie auf **Weiter**.
5. Wählen Sie ein Eingabeformat aus und klicken Sie auf **Weiter**.
6. Wenn Sie COBOL als Eingabeformat auswählen, suchen und wählen Sie ein COBOL-Copybook aus. Klicken Sie auf **Weiter**.
Die Copybook-Spezifikationsdatei weist normalerweise eine *.txt-Erweiterung auf. Developer fügt dem Modellrepository eine XSD-Schemadatei hinzu, die das Copybook darstellt.
7. Wählen Sie eine Ausgabe aus und klicken Sie auf **Weiter**.
8. Wenn Sie COBOL als Ausgabeformat auswählen, suchen und wählen Sie ein COBOL-Copybook aus. Klicken Sie auf **Weiter**.
Wenn Sie COBOL als Eingabeformat auswählen, steht Ihnen die Option zur Auswahl von COBOL als Ausgabeformat nicht zur Verfügung.
9. Klicken Sie auf **Fertig stellen**.
Das Developer Tool erstellt die Umwandlung im Repository. Im Developer Tool wird die Ansicht **Übersicht** angezeigt.
10. Doppelklicken Sie in der Ansicht **Objekte** auf den Parser, um ihn im IntelliScript-Editor zu öffnen.
11. Wenn die COBOL-Daten in EBCDIC codiert sind, ändern Sie in der Ansicht **Einstellungen** die Eingabe- oder Ausgabecodierung in die relevante EBCDIC-Codepage.

COBOL-Datendefinitionen

Das COBOL-Copybook, das Sie zum Erstellen einer Datenprozessor-Umwandlung verwenden, kann Datendefinitionen jeglicher Komplexität enthalten. Das COBOL-Copybook und die Eingabe müssen mit den in diesem Abschnitt erläuterten Datendefinitionsregeln übereinstimmen.

Unterstützte Datendefinitionen

Der COBOL-Import unterstützt Datendefinitionen beliebiger Komplexität. Mögliche Datentypen für die Datendefinition sind beispielsweise gepackte Dezimalzahlen (`COMP-3`), Binärdaten (`COMP-1`, `COMP-2` oder `COMP-4`) und logische Dezimalpunkte (`99V99`). Außerdem können sie Merkmale wie etwa die Bedingungen `REDEFINES`, `OCCURS` und `OCCURS DEPENDING ON` enthalten.

Datendefinitionsregeln

Eine COBOL-Datendefinition muss mit den folgenden Regeln übereinstimmen:

- Nicht mehr als 72 Zeichen für jede Zeile und keinen Text nach Spalte 72
- Die erste Zeile muss einen Kommentar mit einem * in Spalte 7 enthalten oder mit einer Ebenennummer beginnen.
- Die erste Ebenennummer muss sich in Spalte 1 oder 8 befinden.

Nicht unterstützte Datendefinitionen

Die Datenprozessor-Umwandlung unterstützt die folgenden COBOL-Datendefinitionen nicht:

- Die speziellen Ebenennummern 66, 77 und 88
- `USAGE`-Bedingungen auf Gruppenebene
- `INDEXED BY`-Bedingungen
- `POINTER` und `PROCEDURE-POINTER`

Testverfahren

Wenn Sie den COBOL-Parser testen, wandeln Sie die COBOL-Beispieldaten in XML um und überprüfen die Ausgabe. Nachdem Sie den Parser getestet haben, können Sie den COBOL-Serializer auf der Parserausgabe ausführen.

Testen eines COBOL-Parsers

Zum Testen des COBOL-Parsers benötigen Sie eine Eingabedatei, die COBOL-Beispieldaten enthält. Die Datenstruktur muss der importierten Datendefinition entsprechen. Der Parser wandelt COBOL-Beispieldaten in XML um und Sie überprüfen die Ausgabe.

1. Doppelklicken Sie in der Ansicht **Objekt** auf den COBOL-Parser.
Der Parser wird im Skriptbereich des IntelliScript-Editors angezeigt.
2. Klicken Sie mit der rechten Maustaste auf den Namen des Parsers und klicken Sie dann auf **Als Startkomponente festlegen**.
3. Erweitern Sie den IntelliScript-Baum und bearbeiten Sie die Eigenschaft `example_source` des Parsers. Ändern Sie ihren Wert von `Text in LocalFile`.

Der Assistent konfiguriert den COBOL-Parser so, dass kein Beispieldokument erforderlich ist. Nach Abschluss des Tests können Sie die Beispielquelle entfernen. Die Beispielquelle hat keine Auswirkungen auf die Umwandlung zur Laufzeit.

4. Um eine Beispieldatei zuzuweisen, erweitern Sie die `LocalFile`-Komponente, indem Sie auf den Doppelpfeil nach rechts **>>** klicken. Doppelklicken Sie auf die Eigenschaft `file_name` und navigieren Sie zu der Eingabedatei, die die COBOL-Beispieldaten enthält.
5. Im Bereich **Eingabe** der Ansicht **Daten-Viewer** können Sie die Beispieldatei untersuchen. Wenn das Dokument nicht angezeigt wird, klicken Sie mit der rechten Maustaste auf den Namen des Parsers im IntelliScript und klicken Sie dann auf **Beispielquelle öffnen**.
6. Klicken Sie auf **Ausführen > Daten-Viewer ausführen**, um den Parser zu testen.
7. Untersuchen Sie im Bereich **Ausgabe** der Ansicht **Daten-Viewer** die Parserausgabe.
8. Um zu bestätigen, dass der Parser fehlerfrei ausgeführt wurde, klicken Sie im Bereich **Ausgabe** der Ansicht **Daten-Viewer** auf die Schaltfläche **Ereignisse anzeigen**. Überprüfen Sie das Ausführungsprotokoll in der Ansicht **Datenprozessor-Ereignisse**.

Testen des COBOL-Serializers

Nachdem Sie den COBOL-Parser getestet haben, können Sie den COBOL-Serializer auf der Parserausgabe ausführen.

1. Doppelklicken Sie im Explorer für die Data Transformation auf die TGP-Skriptdatei des Serializers. Der Serializer wird im Skriptbereich des IntelliScript-Editors angezeigt.
2. Klicken Sie mit der rechten Maustaste auf den Namen des Serializers und klicken Sie dann auf **Als Startkomponente festlegen**.
3. Klicken Sie auf **Ausführen > Ausführen**, um den Serializer zu aktivieren. Wenn Sie dazu aufgefordert werden, navigieren Sie zur Datei `Results\output.xml`, die Sie beim Ausführen des Parsers erstellt haben.
4. Prüfen Sie das Ausführungsprotokoll in der Ansicht **Ereignisse**. Vergewissern Sie sich, dass der Serializer fehlerfrei ausgeführt wurde.
5. Doppelklicken Sie auf die Datei `Results\output.xml` im Explorer für die Data Transformation. Die Anzeige sollte mit der ursprünglichen Ausgabe übereinstimmen, auf der Sie den Parser ausgeführt haben.

Bearbeiten einer Umwandlung für COBOL

Sie können eine Umwandlung für COBOL bearbeiten, die Sie mit dem Assistenten für Datenprozessor-Umwandlung generieren.

Dabei sollten Sie die Bearbeitung sorgfältig dokumentieren. Diese Dokumentation kann später wichtig werden, wenn Sie die COBOL-Datendefinition ändern und erneut in ein neues Projekt importieren und dabei Ihre Bearbeitung rekonstruieren müssen.

Optimieren der Verarbeitung einer umfangreichen COBOL-Datei in der Hadoop-Umgebung

Sie können die Verarbeitung umfangreicher COBOL-Dateien in der Hadoop-Umgebung durch eine Zuordnung mit einem komplexen Datei-Reader und einer Datenprozessorumwandlung optimieren.

Zur Optimierung der Verarbeitung umfangreicher COBOL-Dateien müssen Sie einen regulären Ausdruck zum Aufteilen der Datensätze verwenden. Wenn die COBOL-Datei mit einem regulären Ausdruck aufgeteilt werden kann, können Sie einen Eingabeparameter für den komplexen Datei-Reader definieren, der einen regulären Ausdruck bereitstellt, der wiederum die Aufteilung der Datensatzverarbeitung in der Hadoop-Umgebung bestimmt.

Konfigurieren des komplexen Datei-Readers für COBOL

Fügen Sie eine Datenprozessorumwandlung hinzu, die COBOL-Eingabe in eine Zuordnung mit einem komplexen Datei-Reader umwandelt. Konfigurieren Sie den komplexen Datei-Reader, um die Verarbeitung von COBOL-Eingabe mithilfe der Zuordnung in einer Hive-Umgebung zu optimieren. Als Eingabecodierung für den komplexen Datei-Reader muss EBCDIC verwendet werden.

1. Erstellen Sie im Mapping-Editor ein komplexes Datei-Reader-Objekt.
2. Um den komplexen Datei-Reader zu konfigurieren, führen Sie die folgenden Schritte aus:
 - a. Wählen Sie auf der Registerkarte **Erweitert** in der Ansicht **Eigenschaften** die Eigenschaft **Dateiformat** und dann **Eingabeformat** aus.
 - b. Wählen Sie die Eigenschaft **Eingabeformat** aus und geben Sie dann `com.informatica.hadoop.reader.RegexInputFormat` ein.
 - c. Verwenden Sie zur Leistungsoptimierung die Eigenschaft **Eingabeformatparameter** und definieren Sie den regulären Ausdruck in der Form `Regex="<regular expression>"`.
3. Erstellen Sie eine Datenprozessorumwandlung für COBOL-Eingabe mithilfe des Assistenten.
4. Fügen Sie die Datenprozessorumwandlung zur Zuordnung hinzu. Der Umwandlungseingabeport sollte auf binäre Eingabe eingestellt bleiben. Dies ist die Standardeinstellung für COBOL-Eingabe.
5. Verknüpfen Sie den Ausgabeport für den komplexen Datei-Reader mit dem Eingabeport für die Datenprozessorumwandlung. Der Ausgabeport für den komplexen Datei-Reader sollte auf binäre Ausgabe eingestellt bleiben.

JSON

Wenn Sie mit JSON-Eingabe oder -Ausgabe eine Datenprozessor-Umwandlung erstellen, wählen Sie ein JSON-Schema oder eine Beispieldatei für die Umwandlung aus. Das Schema oder die Beispieldatei definiert die erwartete Struktur der Eingabe- oder Ausgabedatenhierarchie.

JavaScript Object Notation (JSON) ist ein hierarchisches auf dem Austausch von Daten basiertes Format, ähnlich dem XML-Format. Das JSON-Format wird häufig zum Übermitteln von strukturierten Daten über eine Netzwerkverbindung verwendet. Ein Mapper oder Serializer verwendet ein JSON-Eingabeschema oder -Eingabedokument in derselben Weise wie ein XML-Eingabeschema und -Eingabedokument, um die erwartete Eingabedatenhierarchie zu definieren. Ein Parser verwendet ein JSON-Ausgabeschema oder -Ausgabedokument zum Definieren der erwarteten Ausgabedatenhierarchie.

Wenn Sie den Assistenten für Datenprozessor-Umwandlung zum Erstellen einer Umwandlung mit JSON-Eingabe oder -Ausgabe verwenden, kann die Umwandlung einen Parser, Mapper, Transformer oder Serializer enthalten, der mit der JSON-Hierarchie verknüpft ist. Das JSON-Schema wird in eine .xsd-Datei umgewandelt, die die hierarchische Struktur der JSON-Datei definiert. Sie können diese .xsd-Schemadatei mit allen JSON-Dokumenten verwenden, die dasselbe hierarchische Format aufweisen.

JSON-Schemata

Wenn Sie eine Datenprozessor-Umwandlung mit dem Assistenten erstellen, verwenden Sie ein JSON-Schema oder eine Beispielquelle zum Definieren der JSON-Eingabe oder -Ausgabe.

Der Assistent für Datenprozessor-Umwandlung generiert ein XML-Schema im Modellrepository, das die JSON-Struktur angibt, die von den Umwandlungskomponenten verwendet wird. Die Umwandlung enthält einen dem Schema zugeordneten Transformer und kann je nach der von Ihnen im Assistenten ausgewählten Eingabe oder Ausgabe andere Komponenten enthalten.

Skripts verwenden Schemata zum Definieren der hierarchischen Strukturen der Eingabe und Ausgabe. Ein JSON-Eingabeschema muss dem von der Internet Engineering Task Force veröffentlichten Internetentwurf des JSON-Schemas entsprechen.

Weitere Informationen zur Syntax des JSON-Schemas finden Sie auf den folgenden Websites:

- Einführung in JSON: <http://www.json.org>
- Konvertiert ein JSON-Dokument in ein JSON-Schema: <http://jsonschema.net>

JSON-Beispielschema

Das folgende Beispiel ist ein JSON-Beispielschema:

```
{ "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
                }
              }
            }
          }
        }
      }
    }
  }
}
```

Dieses Schema definiert die Elemente und Attribute, die in einem JSON-Dokument vorkommen können. Das Schema verwendet die JSON-Syntax, um die Hierarchie und Sequenz der Elemente, den Elementdatentyp und die möglichen Werte anzugeben sowie um festzustellen, ob die Elemente erforderlich sind.

Das vorherige Beispielschema definiert das folgende JSON-Eingabedokument:

```
{ "OrgId": "ORG000000000001",
  "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"
        }
      ]
    }
  ]
}
```

Wenn Sie das Schema durchgehen, können Sie die Beziehung zwischen den Elementen des Schemas und dem Eingabedokument erkennen.

Die Schema-Hierarchie enthält das Objekt `Metriken`, das das Array `valueTrend` schachtelt. Das Array enthält die Felder `date` und `val` vom Datentyp `string`.

Erstellen einer Umwandlung mit JSON

Erstellen Sie eine Datenprozessor-Umwandlung mit JSON-Eingabe oder -Ausgabe.

1. Klicken Sie im Developer-Tool auf **Datei > Neu > Umwandlung**.
2. Wählen Sie die Datenprozessor-Umwandlung aus und klicken Sie auf **Weiter**.
3. Geben Sie einen Namen für die Umwandlung ein und suchen Sie nach einem Speicherort für das Modellrepository, an dem die Umwandlung abzulegen ist.
4. Wählen Sie **Datenprozessor mit einem Assistenten erstellen** aus und klicken Sie auf **Weiter**.
5. Wählen Sie ein Eingabeformat aus und klicken Sie auf **Weiter**.
6. Wenn Sie JSON als Eingabeformat auswählen, suchen und wählen Sie ein JSON-Schema oder eine JSON-Beispieldatei aus. Klicken Sie auf **Weiter**.
Normalerweise weist die JSON-Datei eine *.json-Erweiterung auf. Developer fügt dem Modellrepository eine XSD-Schemadatei hinzu, die die JSON-Hierarchie darstellt.
7. Wählen Sie eine Ausgabe aus und klicken Sie auf **Weiter**.
8. Wenn Sie JSON als Ausgabeformat auswählen, suchen und wählen Sie ein JSON-Schema oder eine JSON-Beispieldatei aus. Klicken Sie auf **Weiter**.
Wenn Sie JSON als Eingabeformat ausgewählt haben, steht Ihnen die Option zur Auswahl von JSON als Ausgabeformat nicht zur Verfügung.
9. Klicken Sie auf **Fertig stellen**.

Das Developer-Tool erstellt die Umwandlung im Repository. Im Developer-Tool wird die Ansicht **Übersicht** angezeigt.

10. Doppelklicken Sie in der Ansicht **Objekte** auf die Umwandlungskomponente, um sie im IntelliScript-Editor zu öffnen.

Parquet

Verwenden Sie den Assistenten zum Erstellen einer Umwandlung mit Parquet-Eingabe oder -Ausgabe. Wenn Sie eine Datenprozessorumwandlung erstellen, um das Parquet-Format umzuwandeln, wählen Sie ein Parquet-Schema oder eine Beispieldatei aus, die die erwartete Struktur der Parquet-Daten definiert. Der Assistent erstellt Komponenten, die das Parquet-Format in andere Formate umwandeln oder eine Umwandlung aus anderen Formaten in das Parquet-Format durchführen. Nachdem der Assistent die Umwandlung erstellt hat, können Sie die Umwandlung weiter konfigurieren, um die Mapping-Logik zu bestimmen.

Apache Parquet ist ein spaltenorientiertes Speicherformat, das in einer Hadoop-Umgebung verarbeitet werden kann. Parquet verwendet einen Algorithmus zum Entfernen und Zusammensetzen von Datensätzen und wird implementiert, um komplexe verschachtelte Datenstrukturen zu verarbeiten. Weitere Informationen über Parquet finden Sie unter <http://parquet.incubator.apache.org/documentation/latest/>.

Eine Umwandlung, die Parquet-Eingabe oder -Ausgabe liest, bezieht sich auf ein Schema. Wenn die Umwandlung Parquet-Daten liest oder schreibt, verwendet sie das Schema zum Interpretieren der Hierarchie.

Nachdem Sie eine Datenprozessorumwandlung für Parquet-Eingabe erstellt haben, fügen Sie sie mit einem komplexen Datei-Reader zu einer Umwandlung hinzu. Der komplexe Datei-Reader übergibt Parquet-Eingabe an die Umwandlung. Fügen Sie für eine Datenprozessorumwandlung mit Parquet-Ausgabe eine komplexe Reader-Datei zur Zuordnung hinzu, um die Ausgabe aus der Umwandlung zu erhalten.

Erstellen einer Umwandlung mit Parquet-Eingabe oder -Ausgabe

Erstellen Sie eine Datenprozessorumwandlung mit Parquet-Eingabe oder -Ausgabe.

1. Klicken Sie im Developer-Tool auf **Datei > Neu > Umwandlung**.
2. Wählen Sie die Datenprozessorumwandlung aus und klicken Sie auf **Weiter**.
3. Geben Sie einen Namen für die Umwandlung ein und suchen Sie nach einem Modellrepository-Speicherort, um die Umwandlung abzulegen.
4. Wählen Sie **Datenprozessor mit einem Assistenten erstellen** aus und klicken Sie auf **Weiter**.
5. Wählen Sie ein Eingabeformat aus und klicken Sie auf **Weiter**.
6. Wenn Sie „Parquet“ als Eingabeformat auswählen, führen Sie zum Auswählen eines Parquet-Schemas oder einer Parquet-Beispieldatei einen Suchlauf durch. Klicken Sie auf **Weiter**.

Das Developer-Tool fügt eine Schemaobjektdatei, die die Parquet-Hierarchie darstellt, zum Modellrepository hinzu.

7. Wählen Sie ein Ausgabeformat aus und klicken Sie auf **Weiter**.
8. Wenn Sie „Parquet“ als Ausgabeformat auswählen, führen Sie zum Auswählen eines Parquet-Schemas oder einer Parquet-Beispieldatei einen Suchlauf durch. Klicken Sie auf **Weiter**.

Wenn Sie „Parquet“ als Eingabeformat ausgewählt haben, können Sie „Parquet“ nicht mehr als Ausgabeformat auswählen.

9. Klicken Sie auf **Fertig stellen**.

Das Developer-Tool erstellt die Umwandlung im Repository. Im Developer-Tool wird die Ansicht **Übersicht** angezeigt.

10. Doppelklicken Sie in der Ansicht **Objekte** auf die Umwandlungskomponente, um sie im IntelliScript-Editor zu öffnen.

Fügen Sie zum Konfigurieren der Umwandlungskomponente Zuordnungsanweisungen hinzu.

Konfigurieren des komplexen Datei-Readers für Parquet-Eingabe

Nachdem Sie eine Datenprozessorumwandlung zum Umwandeln von Parquet-Eingabe erstellt haben, fügen Sie die Umwandlung mit einem komplexen Datei-Reader zu einer Zuordnung hinzu. Konfigurieren des komplexen Datei-Readers zur Verarbeitung der Parquet-Eingabe.

1. Erstellen Sie im Mapping-Editor ein komplexes Datei-Reader-Objekt.
2. Um den komplexen Datei-Reader zu konfigurieren, führen Sie die folgenden Schritte aus:
 - a. Wählen Sie auf der Registerkarte **Erweitert** in der Ansicht **Eigenschaften** die Eigenschaft **Dateiformat** und dann **Eingabeformat** aus.
 - b. Wählen Sie auf der Registerkarte **Erweitert** die Eigenschaft **Eingabeformat** aus und geben Sie dann `com.informatica.parquet.ParquetToXML` ein.
 - c. Verwenden Sie zur Leistungsoptimierung die Eigenschaft **Eingabeformatparameter** und passen Sie den Parameter `MaxOutputAccumulation` an. Standardmäßig ist der Parameter `MaxOutputAccumulation`, der die erwartete Anzahl an Ausgabedatensätzen definiert, auf 50.000 festgelegt. Um die Einstellung beispielsweise in 250.000 zu ändern, geben Sie `"MaxOutputAccumulation"="250000"` ein.
 - d. Der komplexe Datei-Reader fügt das Schema standardmäßig zur Ausgabe des komplexen Datei-Readers innerhalb eines einzelnen Elements direkt nach dem Root-Element hinzu. Wenn Sie das Schema nicht zur Ausgabe hinzufügen möchten, wählen Sie die Eigenschaft **Eingabeformatparameter** aus und geben dann `"InjectSchema"="false"` ein.

Verwenden Sie ein Semikolon zum Trennen mehrerer Parameter, z. B.
`"MaxOutputAccumulation"="250000";"InjectSchema"="false"`.
3. Fügen Sie die Datenprozessorumwandlung zur Zuordnung hinzu. Der Umwandlungseingabeport sollte auf binäre Eingabe eingestellt bleiben. Dies ist die Standardeinstellung für Parquet-Eingabe.
4. Verknüpfen Sie den Ausgabeport für den komplexen Datei-Reader mit dem Eingabeport für die Datenprozessorumwandlung. Der Ausgabeport für den komplexen Datei-Reader sollte auf binäre Ausgabe eingestellt bleiben.

Konfigurieren einer Umwandlung mit Parquet-Ausgabe

Um eine Datenprozessor-Umwandlung für Parquet-Ausgabe zu erstellen, verwenden Sie den Assistenten für Datenprozessor-Umwandlung. Der Assistent erstellt die Umwandlung im Modellrepository mit den Komponenten, die für die Umwandlung der Parquet-Ausgabe erforderlich sind. Verwenden Sie den XMap-Editor zum Bearbeiten der XMap (falls enthalten). Fügen Sie die Umwandlung zu einem Mapping mit einem komplexen Datei-Writer hinzu.

1. Erstellen Sie die Datenprozessor-Umwandlung mit dem Assistenten für neue Umwandlungen. Fügen Sie ein Parquet-Schema oder eine Beispieldatei hinzu, die die erwartete Ausgabestruktur definiert.
2. Wenn die Umwandlung ein XML-, JSON- oder ein anderes strukturiertes Eingabeformat aufweist, verwenden Sie den XMap-Editor, um die XMap, die der Assistent in der Umwandlung erstellt hat, zu bearbeiten und anzupassen.

3. Um die Ausgabeeinstellungen für die Datenprozessorumwandlung zu konfigurieren, wählen Sie im Bereich **Ausgabesteuerung** in der Ansicht **Einstellungen** die entsprechende Ausgabeoption im Bereich **Binärer Ausgabereport: Auflistungsmodus** aus.

Hinweis: Um die Ausgabe so zu konfigurieren, dass ein Datensatz gleichzeitig mit dem Schema ausgewählt werden kann, wählen Sie die Option **Ausgabezeile für jede Zeile** aus. Um alle Datensätze mit dem Schema in einem Ausgabestream zu sammeln, wählen Sie die Option **Eingabezeilen in einer einzigen Ausgabe auflisten** aus.

4. Fügen Sie die Datenprozessor-Umwandlung zu einem Mapping hinzu. Die Ausgabeeinstellung der Umwandlung bleibt auf binär eingestellt, dies ist die Standardeinstellung für Parquet-Ausgabe.
5. Erstellen und verknüpfen Sie einen komplexen Datei-Writer mit der Datenprozessor-Umwandlung im Mapping, um die Parquet-Ausgabe aus der Umwandlung zu erhalten.

Hinweis: Die Eingabeeinstellung bleibt auf binär eingestellt, dies ist die Standardeinstellung. Der komplexe Datei-Writer unterstützt keine Komprimierung für Parquet-Ausgabe.

6. Konfigurieren Sie den komplexen Datei-Writer. Wählen Sie im Datenobjektvorgang auf der Registerkarte **Erweitert** für das **Dateiformat** die Option **Benutzerdefinierte Ausgabe** aus. Geben Sie für **Ausgabeformat** die Zeichenfolge `com.informatica.parquet.XMLToParquet` ein.

XML

Verwenden Sie den Assistenten zum Erstellen einer Umwandlung mit XML-Eingabe oder -Ausgabe. Wenn Sie eine Datenprozessor-Umwandlung mit XML-Eingabe oder -Ausgabe erstellen, wählen Sie eine XSD-Schemadatei oder eine XML-Beispieldatei für die Umwandlung aus. Das Schema oder die Beispieldatei definiert die erwartete Struktur der Eingabe- oder Ausgabedatenhierarchie. Wenn Sie eine Beispieldatei auswählen, erstellt der Assistent ein Schema aus der Beispieldatei-Datenhierarchie.

Der Assistent erstellt eine Datenprozessor-Umwandlung, die einen Parser, einen Mapper, eine XMap oder einen Serializer enthalten kann, der bzw. die mit einer XML-Hierarchie verknüpft ist. Ein Mapper, eine XMap oder ein Serializer verwendet ein Eingabeschema, um die erwartete Eingabedatenhierarchie zu definieren. Eine XMap, ein Mapper oder ein Parser verwendet ein Ausgabeschema, um die erwartete Ausgabedatenhierarchie zu definieren. Weitere Informationen zur Schemasyntax finden Sie unter <http://www.w3.org>.

Der Assistent erstellt die Basiskomponenten oder das relationale Mapping, die bzw. das für die Umwandlung basierend auf den Eingabe- und Ausgabetypen erforderlich sind. Die Umwandlung ist möglicherweise vollständig oder kann als Ausgangspunkt zur weiteren Konfiguration eingesetzt werden.

Wenn die Umwandlung über strukturierte Eingabe und Ausgabe verfügt, kann der Assistent eine XMap erstellen, konfigurieren, die Sie zum Umwandeln von Daten aus einer Hierarchie in eine andere konfigurieren. Verwenden Sie den XMap-Editor, um die Knoten für die Eingabe- und Ausgabeschema-Hierarchie zu verknüpfen und die Umwandlungslogik zu definieren. Weitere Informationen zum XMap-Objekt und -Editor finden Sie unter [“Übersicht über XMap” auf Seite 84](#).

Wenn die Umwandlung relationale Eingabe oder Ausgabe aufweist, die Sie aus strukturierten oder in strukturierte Daten umwandeln möchten, erstellt der Assistent ein relationales Mapping. Im Bereich **Ports** in der Ansicht **Übersicht** werden die hierarchischen Schemaknoten und die relationalen Ports angezeigt. Verwenden Sie den Bereich **Ports**, um die hierarchischen Elemente mit den relationalen Ports und Gruppen zu verknüpfen. Weitere Informationen zum Umwandeln der relationalen Daten finden Sie unter [“Relationale Eingabe und Ausgabe – Übersicht” auf Seite 64](#).

Erstellen einer Umwandlung, die XML umwandelt

Erstellen Sie eine Datenprozessor-Umwandlung mit XML-Eingabe, XML-Ausgabe oder beidem.

1. Klicken Sie im Developer-Tool auf **Datei > Neu > Umwandlung**.
2. Wählen Sie die Datenprozessor-Umwandlung aus und klicken Sie auf **Weiter**.
3. Geben Sie einen Namen für die Umwandlung ein und suchen Sie nach einem Speicherort für das Modellrepository, an dem die Umwandlung abzulegen ist.
4. Wählen Sie **Datenprozessor mit einem Assistenten erstellen** aus und klicken Sie auf **Weiter**.
5. Wählen Sie XML oder ein anderes Eingabeformat aus und klicken Sie auf **Weiter**.
6. Wenn Sie XML als Eingabeformat ausgewählt haben, suchen und wählen Sie ein Schema oder ein XML-Datei aus. Klicken Sie auf **Weiter**.

Developer fügt dem Modellrepository eine XSD-Schemadatei hinzu, die die Hierarchie darstellt.

7. Wählen Sie XML oder ein anderes Ausgabeformat aus und klicken Sie auf **Weiter**.
8. Wenn Sie XML als Ausgabeformat auswählen, suchen und wählen Sie ein Schema oder eine XML-Beispieldatei aus. Klicken Sie auf **Weiter**.
9. Klicken Sie auf **Fertig stellen**.

Das Developer-Tool erstellt die Umwandlung im Repository. Im Developer-Tool wird die Ansicht **Übersicht** angezeigt.

10. Um eine spezifische Komponente in der Umwandlung zu bearbeiten, doppelklicken Sie in der Ansicht **Objekte** auf die Umwandlungskomponente, um sie im IntelliScript-Editor zu öffnen.

KAPITEL 4

Relationale Eingabe und Ausgabe

Dieses Kapitel umfasst die folgenden Themen:

- [Relationale Eingabe und Ausgabe – Übersicht, 64](#)
- [Relationale Eingabe, 64](#)
- [Relationale Ausgabe, 71](#)

Relationale Eingabe und Ausgabe – Übersicht

Eine Datenprozessor-Umwandlung kann relationale Datenbankeingaben aus den Eingabeports lesen und in andere Formate umwandeln. Eine Umwandlung kann Zeilen mit relationalen Daten in Ausgabeports ausgeben. Sie können das Mapping mit Datenprozessor-Umwandlungsports in der Ansicht „Übersicht“ der Umwandlung definieren. Alternativ dazu können Sie den Assistenten für Datenprozessor-Umwandlung verwenden, um relationale Daten automatisch zuzuordnen.

Sie können relationale Daten in hierarchische Daten umwandeln. Um Eingabegruppen in hierarchische Daten umzuwandeln, ordnen Sie den hierarchischen Ports Knoten aus der Gruppe der relationalen Ports hinzu. Sie können die Daten aus den hierarchischen Ausgabeports an eine andere Umwandlung in der Zuordnung übergeben.

Sie können die relationale Ausgabe aus der Datenprozessor-Umwandlung zurückgeben. Wenn eine Komponente relationale Daten zurückgibt, erstellen Sie Gruppen von Ausgabeports, indem Sie Knoten aus der hierarchischen Eingabe zu Gruppen relationaler Ports zuordnen. Sie können die Daten aus den relationalen Ausgabeports an eine andere Umwandlung in einer Zuordnung übergeben.

Relationale Eingabe

Eine Datenprozessor-Umwandlung kann relationale Eingabe in hierarchische Ausgabe konvertieren. Eine relationale Datenbank ist eine Datenbank mit einer Sammlung von Tabellendaten, die in Anlehnung an ein relationales Modell organisiert ist. Tabellen weisen möglicherweise zusätzliche Beziehungen untereinander auf.

Im relationalen Modell gibt jedes Tabellenschema eine Spalte an, um jede Zeile eindeutig anzugeben. Diese Spalte wird als Primärschlüssel bezeichnet. Sie geben die Beziehung zwischen jeder Zeile in der Tabelle und einer Zeile in einer anderen Tabelle mit einem Fremdschlüssel an. Ein Fremdschlüssel ist eine Spalte in einer Tabelle, die auf den Primärschlüssel einer anderen Tabelle verweist.

Um relationale Portdaten in hierarchische Daten zu konvertieren, müssen Sie die Struktur des Mappings basierend auf einem hierarchischen Schema definieren. Sie importieren ein Schema in das Modellrepository. Nach dem Import können Sie die Schemakomponenten im Developer-Tool anzeigen. Wenn das hierarchische Schema mehrere Elemente enthält, bei denen es sich um ein Root-Element handeln kann, wählen Sie einen Knoten als Root-Element aus.

Im Bereich **Ports** der Ansicht **Übersicht** können Sie den Schemaknoten die relationalen Eingabeports zuordnen. Auf der linken Seite des Bereichs befindet sich der Bereich **Umwandlungseingabe** und auf der rechten Seite der Registerkarte befindet sich der Bereich **Diensteingabe**.

Wenn Sie einen Knoten aus dem Knoten **Diensteingabe** zum Port **Umwandlungseingabe** ziehen, findet die Zuordnung zwischen einem Schemaknoten und einem relationalen Eingabeknoten statt. Das Developer-Tool erstellt die Eingabeports, um die Daten zuzuordnen. Sie können Gruppen und Ports definieren und Ausgabeports Knoten aus der Eingabe zuordnen.

Eine Datenprozessorumwandlung mit einer relationalen Eingabe kann Pass-Through-Ports enthalten. Sie fügen Pass-Through-Ports zur Root-Gruppe der relationalen Struktur hinzu.

Konfiguration der relationalen Eingabeports

Um relationale Eingabeports zu hierarchischen Ports zuzuordnen, wählen Sie ein Schema zum Definieren der Ausgabe aus. Sie müssen außerdem die relationalen Eingabeports definieren. Erstellen und definieren Sie im Bereich „Ports“ Portgruppen und ordnen Sie den Ports die Knoten aus den hierarchischen Knoten zu.

Um ein Ausgabe-Mapping zu definieren, wählen Sie für den **Datenprozessor-Modus** die Option **Eingabe-Mapping** oder **Eingabe-Mapping und Dienst** aus. Das Mapping verwendet ein Schema zum Definieren der Ausgabe. Wenn das Schema mehr als ein Element enthält, das ein Root-Element sein kann, können Sie einen Knoten vom XML-Ausgabeschema als Root-Element auswählen.

Wenn Sie einen Knoten als Root-Element definieren möchten, klicken Sie auf **Hierarchie auswählen**. Im Developer-Tool werden nur die Knoten ab Stammebene angezeigt. Unterhalb der Stammebene werden sie im Bereich **Umwandlungsausgabe** angezeigt. Klicken Sie auf **Als Hierarchie zeigen**, damit die Ausgabeknoten in einer Hierarchie dargestellt werden. Jede untergeordnete Gruppe wird unterhalb der übergeordneten Gruppe angezeigt.

Erstellen und definieren Sie relationale Eingabeports mithilfe einer der folgenden Methoden:

Knoten auf Ports ziehen

Ziehen Sie Knoten aus dem Bereich **Umwandlungsausgabe** in den Bereich **Umwandlungseingabe**. Wenn Sie einen Knoten in eine Gruppe ziehen, fügt das Developer-Tool der Gruppe einen Port hinzu. Andernfalls wird eine Gruppe mit dem Port darin erstellt.

Manuelles Erstellen von Ports

Zum Erstellen eines Ports wählen Sie ein leeres Feld im Bereich **Umwandlungseingabe** und klicken auf **Neu > Feld**. Wenn Sie kein Feld innerhalb einer Gruppe auswählen, erstellt das Developer-Tool eine Gruppe und fügt der Gruppe den Port hinzu.

Automatisches Erstellen der Ports

Klicken Sie auf **Automatisch zuordnen**. Das Developer-Tool erstellt Eingabegruppen und fügt der Gruppe Ports basierend auf der Ausgabehierarchie hinzu.

Wenn Sie Knoten in den Bereich **Umwandlungseingabe** ziehen, aktualisiert das Developer-Tool das Speicherortfeld auf den Speicherort des Knotens in der Hierarchie. Wenn Sie Ports manuell erstellen, müssen Sie dem Knoten einen Code zuordnen. Klicken Sie auf die Spalte **Speicherort** und wählen Sie einen Knoten aus der Liste aus.

Wenn Sie einen mehrfach vorkommenden Knoten in eine Gruppe ziehen, die das übergeordnete Element enthält, können Sie die Anzahl der einzubeziehenden Vorkommen der untergeordneten Elemente

konfigurieren. Sie können die übergeordnete Gruppe mit der mehrfach vorkommenden untergeordneten Gruppe in der Umwandlungsausgabe ersetzen.

Um eine relationale Gruppe zu erstellen, ziehen Sie einen Ausgabeknoten in eine leere Spalte im Bereich **Umwandlungseingabe**. Wenn Sie einen mehrfach vorkommenden untergeordneten Knoten in eine leere Ausgabespalte ziehen, fordert Sie das Developer-Tool auf, die Gruppe mit anderen Gruppen zu verknüpfen. Wenn Sie eine Gruppe auswählen, erstellt das Developer-Tool Schlüssel, um die Gruppen zu verknüpfen.

Sie können auch eine neue relationale Gruppe erstellen, indem Sie auf **Neu > Gruppe** im Bereich **Umwandlungseingabe** klicken. Geben Sie einen Namen für die Gruppe ein. Konfigurieren Sie die zugehörigen Gruppen von Eingabeports im Bereich **Umwandlungseingabe**. Wenn das Developer-Tool Sie auffordert, Ausgabegruppen zu verknüpfen, fügt es den Gruppen die Schlüssel hinzu. Sie können auch manuell Ports hinzufügen, um Schlüssel zu repräsentieren.

Wenn Linien angezeigt werden sollen, die die Ports mit den hierarchischen Knoten verbinden, klicken Sie auf **Linien zeigen**. Legen Sie fest, ob alle Verbindungslinien oder nur die Linien für bestimmte Ports angezeigt werden sollen.

Richtlinien für das Verknüpfen von Eingabeports

Wenn Sie eine Datenprozessorumwandlung mit dem Assistenten für neue Umwandlungen automatisch erzeugen, erstellt das Developer Tool relationale Ports auf Basis der Schemahierarchie und verknüpft dann die relationalen Ports mit den hierarchischen Knoten.

Beachten Sie die folgenden Regeln und Richtlinien, wenn Sie relationale Eingabeports mit hierarchischen Ausgabeknoten verknüpfen.

- Sie können einen relationalen Eingabeport mit einem Knoten in der Hierarchie verknüpfen.
- Verknüpfen Sie einen Primärschlüssel aus dem jeweiligen Element oder Attribut in der Hierarchie mit einer relationalen Gruppe in der Eingabe. Der Primärschlüssel gibt jede Zeile in den relationalen Tabellen an.
- Verknüpfen Sie einen Fremdschlüssel aus dem jeweiligen Element oder Attribut in der Hierarchie mit einer relationalen Gruppe in der Eingabe. Ein Fremdschlüssel in der relationalen Eingabe ist eine Spalte in einer Tabelle, die auf den Primärschlüssel einer anderen Tabelle zeigt.
- Der relationale Eingabeport und der hierarchische Ausgabeknoten müssen kompatible Datentypen aufweisen.
- Die Schlüssel müssen vom Typ „string“ oder „integer“ sein.

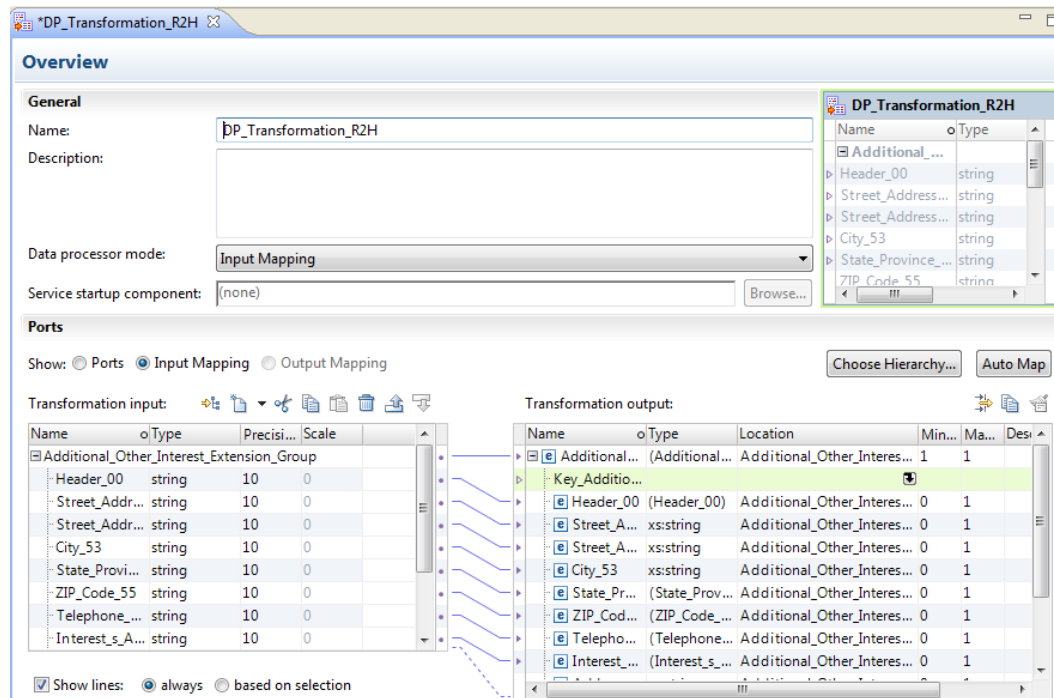
Definieren Sie relationale Eingabeports in der Ansicht „Übersicht“

Um relationale Daten in der Datenprozessor-Umwandlung in hierarchische Daten umzuwandeln, ordnen Sie den hierarchischen Knoten aus der Gruppe von relationalen Ports zu. Verwenden Sie die Ansicht „Übersicht“, um relationale Ports mit hierarchischen Ports zu verknüpfen.

Um relationale Eingabe in hierarchische Ausgabe umzuwandeln, aktivieren Sie die relationale Eingabe in der Ansicht **Übersicht**. Das Developer-Tool entfernt den Standard-Eingabeport aus der Ansicht.

Wählen Sie **Eingabe-Mapping** aus. Der Bereich **Ports** erscheint in der Ansicht **Übersicht**.

Die folgende Abbildung zeigt den Bereich **Ports**:



Links vom Bereich **Ports** befindet sich der Bereich **Umwandlungsausgabe**, der die hierarchischen Schema-Knoten enthält. Rechts befindet sich der Bereich **Umwandlungseingabe**, der die relationalen Elemente und Gruppen enthält.

Sie können Ports in der **Umwandlungseingabe** erstellen und relationale Elemente mit den Schema-Knoten verknüpfen. Sie können auch den Mauszeiger aus einem Knoten auf ein leeres Feld im Bereich **Umwandlungseingabe** ziehen, um einen Port zu erstellen. Wenn Sie einen relationalen Port mit einem Schema-Knoten verbinden, zeigt das Developer-Tool eine Verknüpfung zwischen ihnen an.

Clustering_Key-Ports

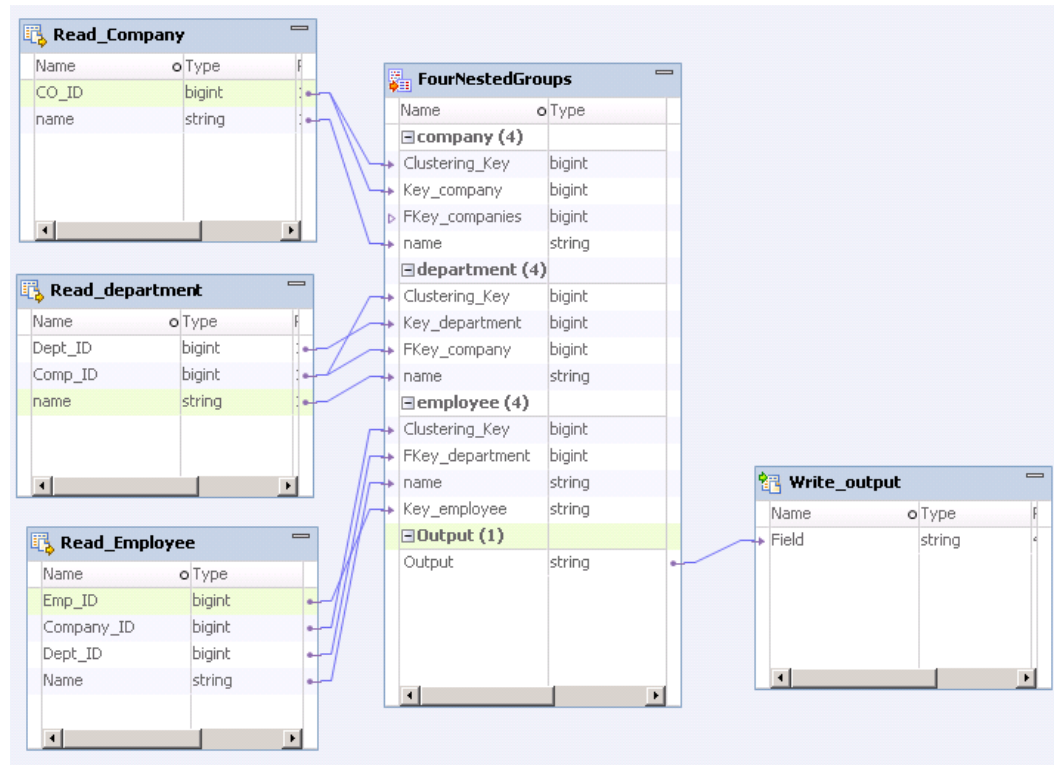
Wenn Sie eine Datenprozessor-Umwandlung (von relational nach hierarchisch) mit mehreren Gruppen in der Hive-Umgebung erstellen, aktivieren Sie Eingabedaten-Partitionierung, um sicherzustellen, dass Daten für jede Zeile ordnungsgemäß verarbeitet werden. Der Datenintegrationsdienst partitioniert die Eingabezeilen gemäß eines Ports, der als Partitionierungsschlüssel namens Clustering_Key fungiert.

Um Eingabedaten in eine Datenprozessor-Umwandlung in einem Mapping zu partitionieren, wählen Sie die Umwandlung im Mapping aus und aktivieren Sie die Partitionierung auf der Registerkarte **Erweitert** der Ansicht **Eigenschaften**. Wenn Sie Partitionierung aktivieren, erstellt Developer einen Clustering_Key-Port in der Datenprozessor-Umwandlung für jede Eingabegruppe.

Jede Eingabegruppe muss denselben Fremdschlüssel in der Eingabe-Root-Gruppe verwenden, um die Partitionierung zu unterstützen. Um die Daten entsprechend eines Schlüssels zu sortieren, verbinden Sie den relationalen Eingabeport des Fremdschlüssels jedes Datenobjekts mit dem entsprechenden Clustering_Key-Port in der Datenprozessor-Umwandlung. Der Datenintegrationsdienst verwendet den Clustering_Key-Port, um die Daten zu partitionieren und zu verarbeiten.

Sie müssen denselben Schlüssel in allen relationalen Eingabegruppen verwenden. Bei Bedarf können Sie eine Joiner-Umwandlung verwenden, um den Schlüssel zu einer relationalen Eingabegruppe hinzuzufügen, die nicht über diesen Schlüssel verfügt.

Das folgende Bild zeigt ein Mapping mit dem Fremdschlüssel Company_ID in den relationalen Eingabegruppen, die mit den Clustering_Key-Ports in der Datenprozessor-Umwandlung verknüpft sind:



Normalisierte relationale Eingabe

Wenn Sie die relationalen Eingabedaten in der hierarchischen Ausgabe normalisieren, werden die Datenwerte in der hierarchischen Gruppe nicht wiederholt. Erstellen Sie eine 1:1-Beziehung zwischen den Hierarchieebenen in den Hierarchieausgabedaten und den Eingabegruppen von Ports.

Normalisierte relationale Eingabe – Beispiel

Sie möchten eine relationale Eingabe mit einer Gruppe, die Details der Manager aus mehreren Unternehmen enthält, in separate XML-Hierarchien umwandeln. In der Eingabe enthält jeder Manager-Datensatz Unternehmensdetails. In der Ausgabe enthält eine XML-Hierarchie Details zu Unternehmen und eine separate XML-Hierarchie enthält Details zu Managern.

In der relationalen Eingabe werden die Company_ID- und Company_Name-Elemente für jeden Manager im Unternehmen wiederholt:

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

Wenn Sie die XML-Ausgabe so definieren, dass sie die übergeordnete Hierarchieebene „Unternehmen“ und die untergeordnete Ebene „Manager“ enthält, können Sie die folgenden Hierarchiegruppen verwenden:

```

Companies
  Company_Key
  Company_ID

```

```
Company_Name
```

Managers

```
Company_Key  
Manager_ID  
Manager_Name
```

Das Company_Key-Element verbindet die Manager-Hierarchie und die Unternehmen-Hierarchie.

Pivotierte relationale Eingabe

Sie können eine bestimmte Anzahl mehrfach vorkommender Elemente in eine Ausgabegruppe aufnehmen.

Um mehrfach vorkommende Elemente zu pivotieren, ordnen Sie die mehrfach vorkommenden Portelemente bestimmten Ausgabeknoten zu.

Pivotierte relationale Eingabe – Beispiel

Sie möchten eine relationale Eingabe, die eine Tabelle mit Telefonnummern enthält, in eine XML-Hierarchie mit separaten Elementen für verschiedene Arten von Telefonnummern umwandeln.

In der relationalen Eingabe definiert das Telephone_type-Element den Typ der für jede Person aufgelisteten Telefonnummer:

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

In der XML-Ausgabehierarchie „Telephones“ weisen verschiedene Typen von Telefonnummern in der übergeordneten Gruppe separate Elemente auf:

Telephones

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

Denormalisierte relationale Eingabe

Sie können hierarchische Ausgabe für relationale Eingabe denormalisieren. Bei der Denormalisierung von Eingabedaten werden die Elementwerte aus der übergeordneten Gruppe für jedes untergeordnete Element wiederholt.

Um Eingabedaten zu denormalisieren, ordnen Sie einer untergeordneten Hierarchie-Ebene Knoten aus einer Gruppe von Ports zu. Alle Elemente werden auf der untergeordneten Hierarchie-Ebene wiederholt.

Denormalisierte relationale Eingabe – Beispiel

Sie möchten relationale Eingabe mit separaten Gruppen für Managerdetails und Unternehmensdetails in eine JSON-Hierarchie umwandeln, die sowohl Manager- als auch Unternehmensdetails enthält.

Das Company_Name-Element wird in der Gruppe mit Managerdetails nicht angezeigt. Das Company_ID-Element ist der Fremdschlüssel in der ersten relationalen Gruppe.

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

Die zweite relationale Gruppe enthält Unternehmensdetails.

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

Das Manager-Element in der JSON-Ausgabe enthält die Elemente Company_ID und Company_Name:

```
Managers
  Company_ID
  Company_Name
  Manager_ID
  Manager_Name
```

Die Elemente Company_ID und Company_Name werden für jeden Manager in der Abteilung wiederholt.

Zuordnen von relationalen Ports zu hierarchischen Knoten

Definieren Sie im Bereich „Ports“ Gruppen von Ausgabeports und ordnen Sie die Knoten aus dem hierarchischen Schema den Ports zu.

1. Um ein Schema hinzuzufügen, klicken Sie in der Ansicht **Referenzen** auf **Hinzufügen**. Navigieren Sie zu einem Schema und wählen Sie es aus.
2. Um ein Eingabe-Mapping zu erstellen, wählen Sie in der Ansicht **Übersicht** im Bereich **Allgemein** die Option **Eingabe-Mapping** für den Datenprozessor-Modus aus.
3. Wählen Sie im Bereich **Ports** die Option **Eingabe-Mapping** aus.
4. Wählen Sie einen Knoten als Root aus.
5. Zum automatischen Definieren eines Mappings klicken Sie auf **Automatisch zuordnen**. Um ein Mapping manuell zu definieren, definieren Sie einen Primärschlüssel für die Root-Eingabegruppe und definieren Sie anschließend Eingabegruppen und Ports.
6. Verwenden Sie zum Hinzufügen einer Eingabegruppe oder eines Eingabeports zum Bereich **Umwandlungseingabe** eine der folgenden Methoden:
 - Ziehen Sie einen hierarchischen Gruppenknoten oder einen untergeordneten Knoten im Bereich **Umwandlungsausgabe** in eine leere Spalte im Bereich **Umwandlungseingabe**. Wenn es sich um einen Gruppenknoten handelt, fügt das Developer-Tool eine relationale Gruppe ohne Ports hinzu.
 - Um eine relationale Gruppe hinzuzufügen, wählen Sie eine Zeile aus und klicken Sie mit der rechten Maustaste und wählen Sie **Neu > Gruppe** aus.
 - Um einen relationalen Port hinzuzufügen, klicken Sie mit der rechten Maustaste und wählen Sie **Neu > Feld** aus.
7. Um Knoten als Primärschlüssel zuzuordnen, verwenden Sie eine der folgenden Methoden:
 - Wählen Sie zwei oder mehrere Hierarchieknoten aus und ziehen Sie sie zu einem Schlüssel im Bereich **Umwandlungseingabe**.
 - Klicken Sie auf die Spalte **Speicherort** eines Schlüssels im Bereich **Umwandlungsausgabe** und wählen Sie dann den relationalen Eingabeport im Bereich **Umwandlungseingabe** aus.

8. Verwenden Sie eine der folgenden Methoden, um die Einstellungen für den hierarchischen Knoten für Speicherorte von Ports zu löschen:
- Wählen Sie einen oder mehrere Knoten im Bereich **Umwandlungsausgabe** aus, klicken Sie mit der rechten Maustaste und wählen Sie **Löschen** aus.
 - Wählen Sie eine oder mehrere Verbindungslinien zwischen den relationalen Eingabeports und den hierarchischen Knoten im Bereich **Umwandlungsausgabe** aus, klicken Sie mit der rechten Maustaste und wählen Sie **Löschen** aus.

Relationale Ausgabe

Wenn Sie die relationale Ausgabe konfigurieren, können Sie eine getrennte Ausgabe für jeden mehrfach vorkommenden Eingabeknoten konfigurieren. Sie können auch Gruppen erstellen, die denormalisierte Daten enthalten. Sie können mehrfach vorkommende Elemente pivotieren und die Anzahl von Vorkommen in einer Ausgabegruppe begrenzen.

Konfiguration von relationalen Eingabeports

Um hierarchische Eingabe in relationale Ausgabe umzuwandeln, wählen Sie ein Schema zum Definieren von hierarchischen Daten aus. Sie müssen außerdem die relationalen Ausgabeports definieren. Definieren Sie im Bereich „Ports“ Gruppen von Ausgabeports und ordnen Sie die Knoten aus dem hierarchischen Schema den Ports zu.

Wählen Sie zum Definieren einer Eingabe- oder Ausgabezuordnung als **Datenprozessor-Modus** entweder **Ausgabezuordnung** oder **Ausgabezuordnung und Dienst** aus.

Das Mapping verwendet ein Schema zum Definieren der hierarchischen Eingabe. Wenn das Schema mehr als ein Element aufweist, das ein Root-Element sein kann, wählen Sie einen Knoten als Root-Element aus. Wenn Sie einen Knoten als Root-Element definieren möchten, klicken Sie auf **Hierarchie auswählen**. Im Developer-Tool werden nur die Knoten ab Stammebene angezeigt. Unterhalb der Stammebene werden sie im Bereich **Umwandlungseingabe** angezeigt.

Klicken Sie auf **Als Hierarchie zeigen**, damit die Ausgabeports in einer Hierarchie dargestellt werden. Jede untergeordnete Gruppe wird unterhalb der übergeordneten Gruppe angezeigt.

Erstellen Sie Ports mithilfe der folgenden Methoden:

Knoten auf Ports ziehen

Ziehen Sie Knoten aus der **Umwandlungseingabe** in den Bereich **Umwandlungsausgabe**. Wenn Sie einen Knoten in eine Gruppe ziehen, fügt das Developer-Tool der Gruppe einen Port hinzu. Andernfalls wird eine Gruppe mit dem Port darin erstellt.

Manuelles Erstellen von Ports

Zum Erstellen eines Ports wählen Sie ein leeres Feld im Bereich **Umwandlungsausgabe** und klicken auf **Neu > Feld**. Wenn Sie kein Feld innerhalb einer Gruppe auswählen, erstellt das Developer-Tool eine Gruppe und fügt der Gruppe den Port hinzu.

Wenn Sie Knoten in den Bereich **Umwandlungseingabe** ziehen, aktualisiert das Developer-Tool das Speicherortfeld auf den Speicherort des Knotens in der Hierarchie. Wenn Sie Ports manuell erstellen, müssen Sie dem Knoten einen Code zuordnen. Klicken Sie auf die Spalte **Ort** und wählen Sie einen Knoten aus der Liste.

Wenn Sie einen mehrfach vorkommenden Knoten in eine Gruppe ziehen, die das übergeordnete Element enthält, können Sie die Anzahl der einzubeziehenden Vorkommen der untergeordneten Elemente konfigurieren. Sie können die übergeordnete Gruppe mit der mehrfach vorkommenden untergeordneten Gruppe in der Umwandlungsausgabe ersetzen.

Um eine Gruppe zu erstellen, ziehen Sie einen Knoten in eine leere Spalte im Bereich **Umwandlungsausgabe**. Wenn Sie einen mehrfach vorkommenden untergeordneten Knoten in eine leere Ausgabespalte ziehen, fordert Sie das Developer-Tool auf, die Gruppe mit anderen Ausgabegruppen zu verknüpfen. Wenn Sie eine Gruppe auswählen, erstellt das Developer-Tool Schlüssel, um die Gruppen zu verknüpfen.

Sie können auch eine neue Gruppe erstellen, indem Sie auf **Neu > Gruppe** klicken. Geben Sie einen Namen für die Gruppe ein.

Konfigurieren Sie die zugehörigen Gruppen von Ausgabeports im Bereich **Umwandlungsausgabe**. Wenn das Developer-Tool Sie auffordert, Ausgabegruppen zu verknüpfen, fügt es den Gruppen die Schlüssel hinzu. Sie können auch manuell Ports hinzufügen, um Schlüssel zu repräsentieren.

Wenn Linien angezeigt werden sollen, die die Ports mit den hierarchischen Knoten verbinden, klicken Sie auf **Linien zeigen**. Legen Sie fest, ob alle Verbindungslinien angezeigt werden sollen oder nur die Linien für bestimmte Ports.

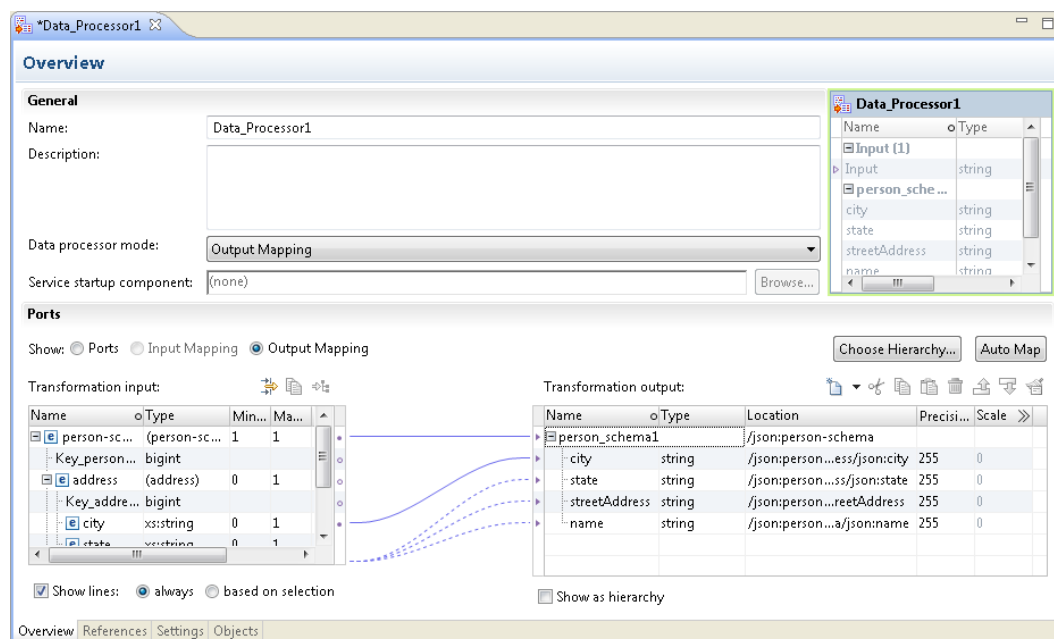
Definieren Sie relationale Ausgabeports in der Ansicht „Übersicht“

Um hierarchische Daten bei der Datenprozessor-Umwandlung in relationale Ausgabe zu konvertieren, verknüpfen Sie hierarchische Knoten mit relationalen Ports. Verwenden Sie die Ansicht „Übersicht“, um relationale Ports mit hierarchischen Ports zu verknüpfen. Sie erstellen Gruppen von Ausgabeports, indem Sie Knoten aus der hierarchischen Ausgabe mit Gruppen von Ports verknüpfen.

Um relationale Gruppen von Ports zurückzugeben, aktivieren Sie die relationale Ausgabe in der Ansicht **Übersicht**. Das Developer-Tool entfernt den Standard-Ausgabeport von der Ansicht.

Wählen Sie **Mapping-Ausgabe**. Der Bereich **Ports** erscheint in der Ansicht **Übersicht**.

Die folgende Abbildung zeigt den Bereich **Ports**:



Auf der linken Seite finden Sie den Bereich **Umwandlungseingabe** mit dem Ausgabeschema. Auf der rechten Seite finden Sie den Bereich **Umwandlungsausgabe** mit den relationalen Ausgabeports.

Definieren Sie die relationalen Ausgabeports im Bereich **Umwandlungsausgabe** und verknüpfen Sie die Knoten aus dem Schema mit den Ports. Sie können auch den Mauszeiger aus einem Knoten auf ein leeres Feld im Bereich **Umwandlungsausgabe** ziehen, um einen Port zu erstellen. Wenn Sie einen Knoten aus dem Ausgabeschema in einen Port ziehen, zeigt das Developer-Tool eine Verbindung zwischen den beiden an.

Normalisierte relationale Ausgabe

Beim Erstellen normalisierter Ausgabedaten werden die Datenwerte in einer Ausgabegruppe nicht wiederholt. Erstellen Sie eine 1:1-Beziehung zwischen den Hierarchieebenen in den Hierarchieeingabedaten und den Ausgabegruppen von Ports.

Normalisierte relationale Ausgabe – Beispiel

Sie möchten eine JSON-Hierarchie, die Abteilungs- und Mitarbeiterdetails definiert, in eine relationale Ausgabe mit separaten Gruppen für Abteilungs- und Mitarbeiterdetails umwandeln.

Die JSON-Eingabe beinhaltet eine Mitarbeiterhierarchie mit Elementen, die Mitarbeiter- und Abteilungsdetails enthalten.

```
Staff
  Department_ID
  Department_Name
  Employee_ID
  Employee_Name
```

Sie können Sie die folgenden Gruppen von relationalen Ports erstellen:

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	Accounting
102	3297	Sales
410	8276	Logistics

Der Department_Key ist ein generierter Schlüssel, der die Mitarbeitergruppe mit einer Abteilungsgruppe in der Ausgabe verbindet.

Pivotierte relationale Ausgabe

Sie können eine bestimmte Anzahl mehrfach vorkommender Elemente in eine Ausgabegruppe aufnehmen.

Um mehrfach vorkommende Elemente zu pivotieren, ordnen Sie die mehrfach vorkommenden untergeordneten Elemente der übergeordneten Gruppe von Ausgabeports zu. Das Developer-Tool fordert Sie auf, die Anzahl der untergeordneten Elemente festzulegen, die in die übergeordnete Gruppe aufgenommen werden sollen.

Pivotierte relationale Ausgabe – Beispiel

Sie möchten eine XML-Hierarchie, die Mitarbeiterdetails mit mehreren IDs enthält, in eine relationale Ausgabegruppe mit separaten Gruppenelementen für die Mitarbeiter-IDs umwandeln.

Das folgende Beispiel zeigt zwei Instanzen von „Employee_ID“ in der relationalen Ausgabegruppe „Departments“:

```
Departments
  Department_ID
  Department_Name
  Employee_ID1
  Employee_ID2
```

Denormalisierte relationale Ausgabe

Sie können relationale Ausgaben denormalisieren. Dabei werden die Elementwerte aus der übergeordneten Gruppe für jedes untergeordnete Element wiederholt.

Um Ausgabedaten zu denormalisieren, ordnen Sie der untergeordneten Gruppe von Ausgabepoints Knoten aus der übergeordneten Hierarchie-Ebene zu.

Denormalisierte relationale Ausgabe – Beispiel

Sie möchten eine XML-Eingabe mit separaten Gruppen für Mitarbeiterdetails und Abteilungsdetails in eine relationale Gruppe umwandeln, die sowohl Mitarbeiter- als auch Abteilungsdetails enthält.

Die XML-Hierarchien trennen die Abteilungsdetails von den Mitarbeiterdetails:

```
Department
  Department_ID
  Department_Name

Employees
  Department_ID
  Employee_ID
  Employee_Name
```

Das folgende Beispiel zeigt die Elemente „Department_ID“ und „Department_Name“ in der Ausgabegruppe „Employees“:

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

„Department_ID“ und „Department_Name“ werden für jeden Mitarbeiter in der Abteilung wiederholt.

KAPITEL 5

Verwenden des IntelliScript-Editors

Dieses Kapitel umfasst die folgenden Themen:

- [IntelliScript-Editor – Übersicht, 75](#)
- [Öffnen eines IntelliScript-Editors, 76](#)
- [Bearbeitungsverfahren, 77](#)
- [Menüs des IntelliScript-Editors, 81](#)

IntelliScript-Editor – Übersicht

Beim IntelliScript-Editor handelt es sich um den Haupteditor, in dem Sie eine Skriptkomponente konfigurieren können, wie z. B. Parser, Mapper oder Serializer.

Der IntelliScript-Editor verfügt über eine Baumstruktur. Die oberste, globale Ebene des Baums zeigt die ausführbaren Umwandlungskomponenten wie Parser an. Diese Komponenten werden als ausführbar bezeichnet, weil sie als Startkomponenten der Umwandlung festgelegt werden können.

Sie können auch nicht ausführbare Komponenten wie Variablen oder Aktionen auf der globalen Ebene definieren. Dadurch können Sie die Komponenten an anderen Speicherorten im Projekt verwenden.

Auf den geschachtelten Ebenen des IntelliScript-Editors können Sie Komponenten wie Anker und Aktionen definieren. Sie können auf die Symbole + oder – klicken, um die Struktur des IntelliScript-Editors zu erweitern und die geschachtelten Ebenen anzuzeigen.

Erstellen eines Skripts

Erstellen Sie ein Skriptobjekt und definieren Sie den zu erstellenden Skript-Komponententyp. Optional können Sie eine Schemareferenz und eine Beispielquelldatei definieren.

1. Klicken Sie in der Ansicht **Objekte** der Datenprozessorumwandlung auf **Neu**.
2. Geben Sie einen Namen für das Skript ein und klicken Sie auf **Weiter**.
3. Wählen Sie die Erstellung eines Parsers oder Serializers aus. Wählen Sie „Andere“, um eine Mapper-, Transformer- oder Streamer-Komponente zu erstellen.
4. Geben Sie einen Namen für die Komponente ein.

5. Handelt es sich bei der Komponente um die erste Komponente, die Daten in der Umwandlung verarbeitet, aktivieren Sie **Als Startkomponente festlegen**.
6. Klicken Sie auf **Weiter**, wenn Sie eine Schemareferenz für dieses Skript eingeben möchten. Klicken Sie auf **Beenden**, wenn Sie die Schemareferenz nicht eingeben möchten.
7. Wenn Sie eine Schemareferenz erstellen, wählen Sie **Referenz einem Schemaobjekt hinzufügen** aus und suchen Sie nach dem Schemaobjekt im Modellrepository. Klicken Sie auf **Neues Schemaobjekt erstellen**, um ein Schemaobjekt im Modellrepository zu erstellen.
8. Klicken Sie auf **Weiter**, um eine Beispielquellenreferenz oder einen Beispieltext einzugeben. Klicken Sie auf **Beenden**, wenn Sie keine Beispielquelle definieren möchten.
Verwenden Sie eine Beispielquelle, um Beispieldaten zu definieren und das Skript zu testen.
9. Wenn Sie eine Beispielquelle auswählen, wählen Sie **Datei** aus und suchen Sie nach der Beispieldatei.
Sie können auch Beispieltext im **Text**-Bereich eingeben. Das Developer-Tool verwendet den Text zum Testen eines Skripts.
10. Klicken Sie auf **Fertig stellen**.
Die **Skript**-Ansicht wird im Developer Tool-Editor angezeigt.

Öffnen eines IntelliScript-Editors

Der IntelliScript-Editor zeigt eine Skriptkomponente in der Datenprozessor-Umwandlung an. Zum Öffnen eines IntelliScript-Editors doppelklicken Sie auf eine Skriptkomponente in der Datenprozessor-Umwandlung. Sie können die Komponenten der Datenprozessor-Umwandlung auf der Registerkarte **Gliederung** anzeigen.

IntelliScript und Daten-Viewer

Im IntelliScript-Editor werden eine oder mehrere Skriptkomponenten angezeigt. Hier definieren Sie die Umwandlungskomponenten.

Sie können das Beispielquellendokument einer Umwandlung auf der Registerkarte **Daten-Viewer** anzeigen. In diesem Bereich können Sie eine Umwandlung anzeigen, konfigurieren oder testen.

Das Beispielfenster des **Daten-Viewer** ist schreibgeschützt. Das Beispielquellendokument kann nicht in Data Transformation Studio bearbeitet werden.

Sie können das Skript im Skriptmodus mit einer Codedarstellung des Skripts oder im Intelli-Modus mithilfe des IntelliScript-Editors anzeigen. Standardmäßig wird das Skript im Intelli-Modus angezeigt.

Finden von Anker

Wenn Sie einen Parser bearbeiten, lassen sich die entsprechenden Anker im IntelliScript- und Beispielbereich leicht finden.

- Klicken Sie mit der rechten Maustaste auf einen Anker im IntelliScript und wählen Sie **Markierung anzeigen** aus. Mit diesem Befehl finden Sie den Anker im Beispielbereich.
- Klicken Sie auf einen Anker im Beispielbereich. Dieser Anker wird automatisch im IntelliScript ausgewählt.

Komponenten und Eigenschaften

Das IntelliScript enthält zwei Arten von Elementen:

- **Komponenten.** Elemente, die Sie in den IntelliScript-Baum einfügen bzw. aus ihm löschen können. Beispiele für Komponenten sind Parser, Serializer, Anker, Aktionen und Transformer.
- **Eigenschaften.** Elemente, die Sie zwar bearbeiten, jedoch nicht einfügen oder löschen können (außer durch Einfügen bzw. Löschen einer Komponente, die diese Elemente enthält). Beispiele für Eigenschaften sind die Eigenschaft `example_source` eines Parsers oder die Eigenschaft `search` eines Marker-Ankers.

Die Namen der Komponenten und Eigenschaften werden im IntelliScript-Editor in unterschiedlichen Farben angezeigt.

Grundeigenschaften und erweiterte Eigenschaften

Zum Vereinfachen der Anzeige werden die Eigenschaften im IntelliScript in zwei Kategorien verwaltet:

- **Grundeigenschaften.** Eigenschaften, die für die meisten Verwendungsarten der Komponente wichtig sind. In der Regel müssen Sie diese Eigenschaften zuweisen, um die Komponente zu verwenden.
- **Erweiterte Eigenschaften.** Eigenschaften, die Sie häufig nicht zuweisen müssen. Diese Eigenschaften haben eventuell Standardwerte, die Sie in der Regel nicht ändern müssen, oder die Eigenschaften implementieren Optionen, die Sie meist nicht verwenden müssen.

Das IntelliScript zeigt die Grundeigenschaften immer an. Die erweiterten Eigenschaften sind solange ausgeblendet, bis Sie sie anzeigen.

Die Unterscheidung erfolgt nur aus Anzeigegründen. Erweiterte Eigenschaften lassen sich genauso einfach verwenden wie Grundeigenschaften. Zögern Sie nicht, erweiterte Eigenschaften zu verwenden, wenn Sie diese in Ihren Projekten benötigen.

Anzeigen der erweiterten Eigenschaften einer Komponente

- Klicken Sie auf das Symbol **>>** rechts vom Komponentennamen.
Das Symbol **>>** wird in **<<** geändert, und die erweiterten Eigenschaften werden angezeigt.

Ausblenden der erweiterten Eigenschaften

- Klicken Sie auf das Symbol **<<** rechts vom Komponentennamen.
Das Symbol **<<** wird wieder in **>>** geändert, und die erweiterten Eigenschaften werden ausgeblendet. Wenn Sie einer erweiterten Eigenschaft aber einen Nicht-Standard-Wert zugewiesen haben, wird diese weiterhin angezeigt.

Bearbeitungsverfahren

Um das IntelliScript zu bearbeiten, folgen Sie den in diesem Abschnitt beschriebenen Verfahren.

Grundlegende Vorgehensweise bei der Bearbeitung

So bearbeiten Sie das IntelliScript:

1. Klicken Sie auf die Komponente oder Eigenschaft, die Sie bearbeiten möchten.

2. Drücken Sie die **EINGABETASTE**, um in den Bearbeitungsmodus zu wechseln. An den meisten Stellen können Sie auch doppelklicken, statt die **EINGABETASTE** zu drücken.
3. Weisen Sie den Komponentennamen oder den Eigenschaftswert zu.
4. Drücken Sie nochmals die **EINGABETASTE**, um den Bearbeitungsvorgang abzuschließen.

Kopieren und Einfügen

Im IntelliScript können Sie Komponenten kopieren und einfügen.

Um mehrere Komponenten gleichzeitig zu kopieren, halten Sie **STRG** oder **UMSCHALT** gedrückt, während Sie die Komponenten mit der Maus auswählen. Die Komponenten müssen sich alle auf derselben Verschachtelungsebene im IntelliScript befinden.

Sie können Komponenten nur an Stellen einfügen, an denen dies sinnvoll ist. Beispielsweise können Sie Serialisierungsanker unter einem Serializer, jedoch nicht unter einem Parser einfügen.

Ziehen und Ablegen

Sie können per Drag&Drop Komponenten von einer Stelle an eine andere verschieben. Sie können diese Methode z. B. verwenden, um die Reihenfolge der Anker innerhalb eines Parsers zu ändern.

Um mehrere Komponenten zu verschieben, halten Sie **STRG** oder **UMSCHALT** gedrückt, während Sie die Komponenten mit der Maus auswählen. Lassen Sie **STRG** oder **UMSCHALT** los und ziehen Sie die Komponenten dann mit der Maus.

Um die ausgewählten Komponenten zu kopieren (statt sie zu verschieben), halten Sie beim Ziehen **STRG** gedrückt.

Suchen und Ersetzen

Wählen Sie zum Suchen oder Ersetzen von Text im IntelliScript die Optionen **Bearbeiten > Suchen** oder **Bearbeiten > Ersetzen** aus.

Einfügen von Komponenten in das IntelliScript

Unter zahlreichen Komponenten zeigt das IntelliScript eine horizontale Linie an. Diese enthält meist eine Beschriftung wie . Der Linie sind ein Pfeil und drei Punkte nachgestellt (. . .). Sie können geschachtelte Komponenten an den drei Punkten einfügen.

Einfügen einer Komponente

1. Markieren Sie die drei Punkte und drücken Sie die **EINGABETASTE**.
Eine Liste der Komponenten wird angezeigt
2. Wählen Sie in der Liste eine Komponente aus.
Alternativ können Sie die ersten Buchstaben des Komponentennamens eingeben. Der Name wird dann automatisch vervollständigt.
3. Drücken Sie nochmals die **EINGABETASTE**, um das Einfügen abzuschließen.

Löschen einer Komponente

- Wählen Sie die Komponente aus und drücken Sie **ENTF**.

Bearbeiten der Eigenschaften einer Komponente

1. Wählen Sie den Wert einer Eigenschaft aus und drücken Sie die **EINGABETASTE**.
Je nach Art der Eigenschaft wird ein Textfeld, eine Liste oder ein Dialogfeld angezeigt.
2. Geben Sie den neuen Eigenschaftswert ein bzw. wählen Sie diesen aus.
3. Drücken Sie die **EINGABETASTE**, um die Zuweisung abzuschließen.

Einfügen von Tabulatoren, Zeilenumbrüchen und anderen Sonderzeichen

Wenn Sie eine Text Eigenschaft zuweisen, können Sie Sonderzeichen einfügen, indem Sie die entsprechenden numerischen ASCII-Codes eingeben.

1. Wählen Sie eine Eigenschaft aus und drücken Sie die **EINGABETASTE**, um mit dem Bearbeiten zu beginnen.
2. Drücken Sie **STRG+A**.
Es wird ein kleiner Punkt angezeigt, der kennzeichnet, dass das Zeichen ein ASCII-Code ist.
3. Geben Sie den dreistelligen ASCII-Code ein. Sie können z. .B. Folgendes eingeben:

ASCII-Code	Zeichen
009	Tab
010	Neue Zeile
013	Wagenrücklauf

4. Um einen String aus ASCII-Codes einzugeben, wiederholen Sie die Schritte 2 und 3.
Sie können ASCII-Codes und normalen Text mischen.
5. Drücken Sie die **EINGABETASTE**, um die Bearbeitung abzuschließen.
Eine Registerkarte wird als «-Symbol angezeigt. Andere Zeichen werden mit den zugehörigen ASCII-Zeichencodes angezeigt.

Definieren einer globalen Komponente

Sie können Komponenten sowohl in einem globalen als auch in einem lokalen Bereich einfügen.

Bereich	Beschreibung
Globaler Bereich	Die Komponente ist auf der obersten Ebene des IntelliScripts definiert. Sie kann an jeder beliebigen Stelle des Projekts aufgerufen und verwendet werden.
Lokaler Bereich	Die Komponente ist auf einer geschachtelten Ebene des IntelliScripts definiert. Sie kann nur an dieser bestimmten Stelle des Projekts aufgerufen und verwendet werden.

Die meisten Data Transformation-Komponenten können sowohl global als auch lokal eingesetzt werden.

Anker z. B. sind in der Regel lokal definiert. Sie können einen Anker aber auch als globale Komponente definieren, wenn Sie dieselbe Ankerkonfiguration in mehreren Parsern oder mehrmals in demselben Parser

verwenden wollen. Sie können den als global definierten Anker an jeder gewünschten Stelle durch seinen Bezeichner referenzieren.

Ein Parser kann somit den Bezeichner `MyMarker` verwenden, anstatt jedes Mal, wenn der Marker-Anker benötigt wird, die Konfiguration des Marker-Ankers zu wiederholen. Sie können entweder `MyMarker` an der entsprechenden Stelle im Parser aus der Dropdown-Liste auswählen oder `MyMarker` an diese Stelle ziehen.

Einschränkungen bei Namen

Die Namen, die Sie den IntelliScript-Komponenten zuweisen, dürfen nur lateinische Buchstaben (A-Z, a-z), Ziffern (0-9) und Unterstriche (_) enthalten. Sie müssen mit einem Buchstaben beginnen und können bis zu 127 Zeichen lang sein.

Anzeigen der Hilfe zu einer Komponente










Beim Bearbeiten des IntelliScripts können Sie die Themen der Online-Hilfe anzeigen, die eine Komponente oder Eigenschaft beschreiben.

1. Zeigen Sie die Ansicht **Hilfe** an.
2. Wählen Sie die Komponente oder Eigenschaft aus.

Die Hilfe zeigt die Seite an, auf der das ausgewählte Element beschrieben wird.

IntelliScript-Symbole

Das IntelliScript zeigt jeden Komponententyp mit einem charakteristischen Symbol an. In der folgenden Tabelle werden die häufigsten Symbole erläutert, die im IntelliScript angezeigt werden.

Symbol	Komponente
	Parser
	Serializer
	Mapper
	Transformer
	Marker
	Content ContentSerializer
	Group
	RepeatingGroup
	StringSerializer

Symbol	Komponente
	Handle
	Key
	Andere Anker
	Aktionen
	Standardsymbol, wenn kein bestimmtes Symbol für eine Komponente vorhanden ist

Speichern des IntelliScripts

Wenn ein Sternchen (*) auf der Registerkarte des IntelliScript-Editors angezeigt wird, enthält der Editor ungespeicherte Änderungen. Wenn Sie den Editor schließen oder ohne Speichern der Änderungen beenden, werden Sie aufgefordert, das Skript zu speichern.

Menüs des IntelliScript-Editors

Klicken Sie im IntelliScript-Editor mit der rechten Maustaste, um ein Menü anzuzeigen. Die Menüoptionen hängen von dem Kontext ab, in dem Sie geklickt haben.

Tabelle 1. Menü des IntelliScript-Bereichs

Option	Beschreibung
Markierung anzeigen	Hebt den ausgewählten Anker in der Beispielquelle hervor.
Als Einrichtungskomponente festlegen	Legt die ausgewählte Komponente als Startkomponente des Projekts fest.
Ausschneiden	Ermöglicht das Ausschneiden von Komponenten im IntelliScript.
Kopieren	Ermöglicht das Kopieren von Komponenten im IntelliScript.
Einfügen	Ermöglicht das Einfügen von Komponenten in das IntelliScript.
Einfügen	Ermöglicht das Einfügen von Komponenten in das IntelliScript.
Löschen	Ermöglicht das Löschen von Komponenten im IntelliScript.
In „optional“ ändern	Wählt die Eigenschaft <code>optional</code> einer Komponente aus. Wenn eine Komponente optional ist, schlägt die übergeordnete Komponente bei einem Scheitern der Komponente nicht fehl. Weitere Informationen finden Sie im <i>Data Transformation Studio-Benutzerhandbuch</i> .

Option	Beschreibung
In „erforderlich“ ändern	Hebt die Auswahl der Eigenschaft <code>optional</code> einer Komponente auf. Wenn eine Komponente optional ist, schlägt die übergeordnete Komponente bei einem Scheitern der Komponente nicht fehl. Weitere Informationen finden Sie im <i>Data TransformationData Transformation Studio-Benutzerhandbuch</i> .
Aktivieren	Wählt die Eigenschaft <code>disabled</code> einer Komponente aus. Eine deaktivierte Komponente wird ignoriert. Diese Funktion ist hilfreich, um eine Komponente temporär zum Testen und zur Fehlerbehebung zu deaktivieren.
Deaktivieren	Hebt die Auswahl der Eigenschaft <code>disabled</code> einer Komponente auf. Eine deaktivierte Komponente wird ignoriert. Diese Funktion ist hilfreich, um eine Komponente temporär zum Testen und zur Fehlerbehebung zu deaktivieren.
Skriptmodus	Der Script-Modus zeigt den Rohinhalt der *.tgp-Datei an. Dieser Modus ist ausschließlich für eine erweiterte Fehlersuche vorgesehen.
Intelli-Modus	Der Intelli-Modus zeigt das IntelliScript in einer lesbaren grafischen Darstellung an. Das ist der Modus, der in diesem Handbuch sowie in der weiteren Data Transformation-Dokumentation dargestellt ist.
Beispielquelle öffnen	Öffnet die Beispielquelldatei des ausgewählten Parsers, Serializers oder Mappers.
Serializer erstellen	Erstellt einen Serializer aus dem ausgewählten Parser. Der Serializer und der Parser führen inverse Umwandlungen durch.

Tabelle 2. Menü des Beispielbereichs

Option	Beschreibung
Kopieren	Kopiert einen String in die Zwischenablage.
Marker einfügen	Definiert den ausgewählten Text als einen <code>Marker</code> -Anker. Der Anker wird im IntelliScript hinzugefügt.
Content einfügen	Definiert den ausgewählten Text als einen <code>Content</code> -Anker. Der Anker wird im IntelliScript hinzugefügt.
Abstandsgestützten Inhalt einfügen	Definiert die ausgewählte Position als einen <code>Content</code> -Anker. Der Anker wird im IntelliScript hinzugefügt.
RepeatingGroup einfügen	Definiert den ausgewählten Text als Trennzeichen eines <code>RepeatingGroup</code> -Ankers. Der Anker wird im IntelliScript hinzugefügt.
Instanz anzeigen	Sucht im IntelliScript nach dem ausgewählten Anker .
Ereignis anzeigen	Zeigt das zugehörige Ereignis in der Ansicht Ereignisse an.
Suchen	Sucht einen String im Beispielquellendokument.
Logisches Codieren	Wenn das Beispielquellendokument Text in einer Sprache mit Lesefolge von rechts nach links enthält, z. B. Hebräisch oder Arabisch, wechselt dieser Befehl die Anzeige von links-nach-rechts in rechts-nach-links.

Option	Beschreibung
Zeilenumbruch	Bricht lange Zeilen um.
Quelle speichern unter	Speichert das Beispielquellldokument an einem angegebenen Speicherort unter einem angegebenen Namen.

KAPITEL 6

XMap

Dieses Kapitel umfasst die folgenden Themen:

- [Übersicht über XMap, 84](#)
- [XMap-Schemata, 85](#)
- [Mapping-Anweisungen, 86](#)
- [XPath-Ausdrücke, 104](#)
- [XMap-Variablen, 111](#)
- [XMap-Beispiel, 112](#)

Übersicht über XMap

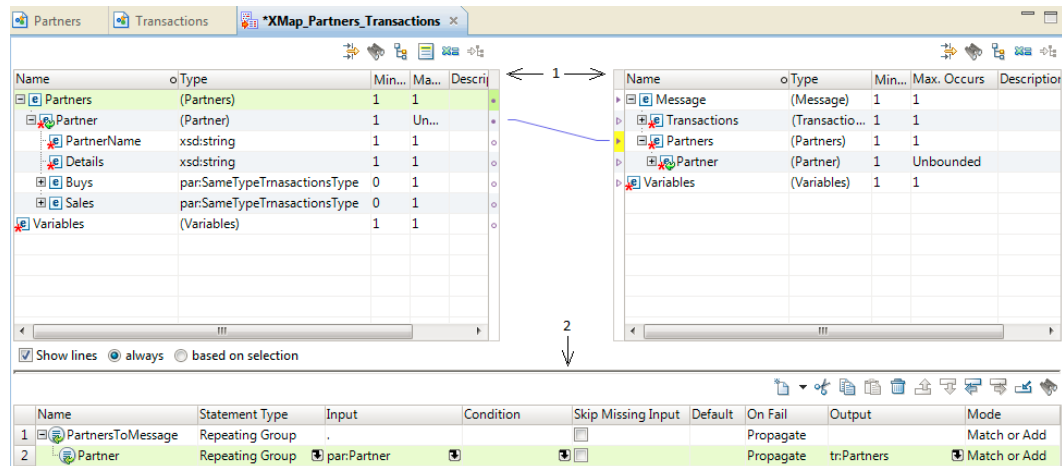
Eine XMap ist ein Datenprozessor-Umwandlungsobjekt, das ein hierarchisches Eingabedokument in ein anderes hierarchisches Dokument mit einer anderen Hierarchiestruktur konvertiert.

Eine XMap verwendet Eingabe- und Ausgabeschemata, um die erwartete Hierarchie von Eingabe- und Ausgabedokumenten zu definieren. Verwenden Sie den XMap-Editor, um Mapping-Anweisungen zu definieren und zu verwalten. Der XMap-Editor enthält die Eingabeschema-Hierarchie und die Ausgabeschema-Hierarchie. Mapping-Anweisungen verknüpfen Eingabeschema-Elemente mit Ausgabeschema-Elementen.

Eine XMap kann beliebige Eingabehierarchiedokumente, deren Elemente mit der Eingabeschema-Hierarchie übereinstimmen, in ein Ausgabeelement mit der Hierarchie des Ausgabeschemas umwandeln.

Beispiel: Eine XMap kann Kundenrechnungen in eine Liste von Kundenaufträgen sortiert nach Monat und Jahr umwandeln. Die Eingabe ist ein JSON-Dokument, das eine Hierarchie von Kundenelementen enthält. Die Ausgabe ist ein XML-Dokument, das eine Hierarchie von Monatelementen enthält. Das Ausgabe-XML-Dokument gruppiert Kundenaufträge nach Monat und enthält Kontaktinformationen und die Auftragssumme.

Die folgende Abbildung zeigt den XMap-Editor.



1. Der XMap-Editor enthält Eingabe- und Ausgabehierarchieschemata. Ziehen Sie die Schemaelemente und fügen Sie sie ein, um Mapping-Anweisungen zu erstellen.
2. Das XMap-Editor-Gitter zeigt Mapping-Anweisungen an. Verwenden Sie die Gitter zum Verwalten und Bearbeiten der Mapping-Anweisungen.

Eine XMap verwendet Mapping-Anweisungen, um festzulegen, wie ein Eingabeschemaelement in ein Ausgabeschemaelement umgewandelt werden soll. Sie können ein Element aus einem Knoten im Eingabeschema in einen Knoten im Ausgabeschema ziehen, um eine Verknüpfung zu erstellen. Wenn Sie Verknüpfungen erstellen, werden diese als Mapping-Anweisungen bezeichnet. Der XMap-Editor zeigt die Mapping-Anweisung im Gitter an.

Sie können die Mapping-Anweisungen im Gitter bearbeiten. Sie können Bedingungen definieren, um die Daten mithilfe der verschiedenen Mapping-Anweisungstypen und XPath-Ausdrücken umzuwandeln und zu filtern. XPath ist eine Abfragesprache, die verwendet wird, um Knoten in einem hierarchischen Dokument auszuwählen und Berechnungen durchzuführen.

Sie können XPath-Ausdrücke zum Definieren des Kontexts für eine Mapping-Anweisung verwenden. Mit XPath-Ausdrücken können Sie ebenfalls verschiedene arithmetische Berechnungen zu einer Mapping-Anweisung hinzufügen.

XMap-Schemata

Eine XMap erfordert hierarchische Schemata, die die hierarchischen Sie die Eingabe- und Ausgabe hierarchien definieren. Eine XMap verwendet Eingabe- und Ausgabe hierarchieschemata, um festzulegen, welcher Datentyp im Eingabequell dokument und Ausgabedokument erwartet wird. Sie verknüpfen Eingabe- und Ausgabeschemaknoten, um Mapping-Anweisungen zu erstellen.

Ein Schemaelement ist der grundlegende Baustein einer Mapping-Anweisung. Wenn Sie eine Mapping-Anweisung definieren, können Sie eine Reihe von Eingabeschemaelementen in der Anweisung verwenden oder eine Variable hinzufügen. Sie können die Eingabe- und Ausgabe-Root ändern, Variablen hinzufügen und die Schemaansicht anpassen, das Schema jedoch nicht bearbeiten.

Sie können die folgenden Optionen verwenden, um die Schemata zu verwalten und Elemente zu suchen.

Ansicht anpassen

Ändert die Anzeige des Schemas, sodass Sie schnell nach relevanten Elementen suchen können. Sie können nach einer Abfolge von Knoten, nach allen Knoten oder nach bestimmten Knoten suchen. Zeigen Sie diese Knoten an, damit Sie die Logik des Schemas verstehen. Diese Ansicht wirkt sich nicht auf das Zuordnen aus.

Search

Mit Auswahl dieser Option werden Elemente im Schema gesucht. Sie haben die Möglichkeit, in den Eingabe- und Ausgabeschemata separat zu suchen.

Variablen

Definiert Variablen zum Speichern von Daten. Sie können Variablen zu Knoten zuordnen und umgekehrt. Sie können außerdem Variablen zu Variablen zuordnen. Wenn Sie eine Variable erstellen, wird diese im unteren Bereich von beiden Schemata angezeigt.

Eingabe- oder Ausgabe-Root auswählen

Ändern Sie das Root-Element im Eingabe- oder Ausgabeschema. Sie können das Root-Element ändern, um einen anderen Abschnitt des Schemas zu referenzieren.

Beispielquelle auswählen

Definieren Sie die Beispielquelldatei. Testen Sie die Umwandlung und die XPath-Ausdrücke mithilfe einer Beispielquelle.

Mapping-Anweisungen

Eine Mapping-Anweisung legt das Mapping von Daten aus dem Eingabehierarchiedokument zum Ausgabehierarchiedokument fest. Wenn Sie einen Knoten aus dem Eingabeschema zum Ausgabeschema ziehen, erstellt der XMap-Editor Mapping-Anweisungen im Gitter.

Sie können das Gitter verwenden, um einfache oder detaillierte Mapping-Anweisungen zu erstellen. Sie können Elemente aus den Eingabe- oder Ausgabeschemata in Felder im Gitter ziehen, um sie in Mapping-Anweisungen einzuschließen.

Sie können das Element überprüfen, auf das eine Mapping-Anweisung verweist. Wenn Sie im Gitter auf eine Mapping-Anweisung klicken, markiert der XMap-Editor die Knoten in den Schemata.

Fügen Sie XPath-Ausdrücke hinzu, um den Kontext zu bestimmen, oder fügen Sie einer Mapping-Anweisung Berechtigungen hinzu. Wenn ein XPath-Ausdruck den Kontext für eine Mapping-Anweisung identifiziert, führt die Datenprozessor-Umwandlung die Mapping-Anweisung für jedes Vorkommen des Eingabeelements oder des Ausdrucks im Eingabedokument durch.

Schachteln Sie Mapping-Anweisungen, um eine Abhängigkeit zu anderen Mapping-Anweisungen herzustellen. Eine Mapping-Anweisung kann ein übergeordnetes Element einer Gruppe von untergeordneten Anweisungen sein. Jedes Mal, wenn die Datenprozessor-Umwandlung die übergeordnete Anweisung ausführt, werden ebenfalls die untergeordneten Anweisungen ausgeführt. Untergeordnete Anweisungen erscheinen mit einem Einzug unter der übergeordneten Anweisung im XMap-Editor.

Typen von Mapping-Anweisungen

Die Mapping-Anweisungstypen definieren XMap-Mapping-Logik. Definieren Sie den Mapping-Anweisungstyp basierend darauf, ob Sie einen einfachen Eingabewert einem Ausgabewert zuordnen, ein Element durchlaufen oder das Zuordnen basierend auf einer Bedingung durchführen möchten.

Erstellen Sie eine Anweisung, indem Sie ein Eingabeschema-Element in ein Ausgabeschema-Element ziehen oder eine Mapping-Anweisung zu einem Gitter hinzufügen. Beim Erstellen einer Anweisung identifiziert die Datenprozessor-Umwandlung einen Mapping-Anweisungstyp basierend darauf, ob es sich bei dem Element um ein einfaches, komplexes oder sich wiederholendes Segment handelt.

Der Basistyp einer Mapping-Anweisung ist eine Map, die einen einfachen Eingabewert zu einem einfachen Ausgabewert zuordnet. Andere Mapping-Anweisungen identifizieren Bedingungen oder Alternativen für die Mapping-Logik oder gruppieren eine Gruppe von logischen Anweisungen.

Sie können die folgenden Mapping-Anweisungs-Typen hinzufügen:

Map

Ordnet ein einfaches Eingabeelement zu einem einfachen Ausgabeelement zu. Eine Map-Anweisung ist der grundlegende Baustein der XMap.

Group

Eine logische Gruppe von Anweisungen. Andere Mapping-Anweisungstypen sind unter der Group-Anweisung geschachtelt.

Repeating Group

Eine Group-Anweisung, die die Datenprozessor-Umwandlung jedes Mal durchführt, wenn das Eingabeelement im Eingabedokument angezeigt wird. Die Repeating Group enthält Map-Anweisungen, die durchlaufen. Die Repeating Group identifiziert das zum Durchlaufen der Gruppe verwendete Element.

Router

Enthält eine Gruppe von Option-Anweisungen und wählt nur die Option-Anweisung aus, deren Bedungskriterien mit der Eingabe übereinstimmen. Wenn keine der Optionen angewendet wird, wird eine Standardaktion ausgeführt, wenn eine Default-Anweisung vorhanden ist. Wenn keine der Optionen angewendet wird und keine Default-Anweisung vorliegt, schlägt der Router fehl.

Option

Eine oder mehrere Option-Anweisungen werden unter der Router-Anweisung geschachtelt. Die Option-Anweisung ist wie eine Group-Anweisung und enthält eine logische Gruppe von Anweisungen. Die Option-Anweisung definiert eine Bedingung, um das Eingabeelement dem Ausgabeelement zuzuordnen.

Standard

Eine Default-Anweisung kann unter der Router-Anweisung geschachtelt werden. Die Default-Anweisung wird ausgeführt, wenn keine der Option-Anweisungen angewendet wird. Wenn alle Option-Anweisungen fehlschlagen und keine Default-Anweisung vorliegt, schlägt der Router fehl.

Run XMap

Ruft ein anderes XMap-Objekt in einer Datenprozessor-Umwandlung auf.

RunMapplet

Ruft ein Mapplet aus der Datenprozessorumwandlung auf.

MappletInput

Eine oder mehrere MappletInput-Anweisungen können unter der RunMapplet-Anweisung geschachtelt werden. Die Werte werden den Mapplet-Eingabeports in der gleichen Reihenfolge zugeordnet, wie sie in den MappletInput-Anweisungen aufgelistet sind.

MappletOutput

Ein oder mehrere MappletOutput-Anweisungen können unter der RunMapplet-Anweisung geschachtelt werden. Die Werte in den Mapplet-Ausgabeports werden den MappletOutput-Anweisungen in der gleichen Reihenfolge zugeordnet, wie sie in den Mapplet-Ports aufgelistet sind.

Mapping-Anweisungen enthalten Felder, die Sie konfigurieren können, um die Anweisung anzupassen. Sie können die Eingabe, Ausgabe und Bedingung für das Zuordnen eines Eingabeelements zu einem Ausgabeelement konfigurieren.

Konfigurieren Sie, ob eine Mapping-Anweisung übersprungen werden soll, wenn diese fehlschlägt bzw. keine Eingabe vorhanden ist. Konfigurieren Sie, ob die Datenprozessor-Umwandlung ein Ausgabeelement hinzufügen oder eine Übereinstimmung mit einem Wert aus einer Mapping-Anweisung für ein vorhandenes Element suchen soll.

Map-Anweisungen

Eine Map-Anweisung ist der grundlegende Baustein eines XMap-Objekts und ordnet einem einfachen Ausgabewert einen einfachen Eingabewert zu. Die Eingabe muss ein einfacher Wert oder ein konstanter Wert sein. In einer Map-Anweisung müssen Eingabe und Ausgabe definiert werden.

Wenn Sie Elemente zwischen einem einfachen, nicht wiederholenden Eingabeschemaknoten und einem einfachen, nicht wiederholenden Ausgabeschemaknoten durch Ziehen und Einfügen verschieben, wird automatisch eine Map-Anweisung erstellt.

Eine Group-, Repeating Group-, Option- oder Default-Anweisung kann eine oder mehrere untergeordnete Map-Anweisungen enthalten.

Eigenschaften der Map-Anweisung

Eine Map-Anweisung enthält Eigenschaften, die Sie zum Anpassen der Anweisung konfigurieren können. Sie können die Eingabe, Ausgabe und eine Bedingung für das Zuordnen eines Eingabeelements zu einem Ausgabeelement konfigurieren.

Eine Map-Anweisung hat die folgenden Eigenschaften:

Bedingung

Optional. Ein XPath-Ausdruck, der eine Bedingung für das Zuordnen des Elements definiert. Eine Bedingung ähnelt einem Prädikatsausdruck in der Eingabespalte. Wenn Sie einen Eingabe-XPath-Ausdruck und einen Bedingungs-XPath-Ausdruck für dieselbe Mapping-Anweisung definieren, wendet die Datenprozessor-Umwandlung den Bedingungs-XPath auf das Ergebnis des Eingabe-XPath an.

Default

Optional. Der zu verwendende Standardwert, wenn ein Element bei der Eingabe fehlt. Sie können zum Beispiel einen Standardwert festlegen, um einen Zähler zu initialisieren.

Eingabe

Erforderlich. Ein XPath-Ausdruck, der das Eingabeelement definiert. Der Ausdruck kann ein Knoten oder Wert sein.

Modus

Erforderlich. Legt fest, ob die Datenprozessor-Umwandlung ein Ausgabeelement hinzufügt oder ein vorhandenes Element mit einem Wert aus der Mapping-Anweisung abgleicht. Wählen Sie eine der folgenden Optionen aus:

- Hinzufügen. Erstellt ein Element im Ausgabehierarchiedokument. Wenn das Element nicht mehrmals vorkommt und derselbe Wert in der Ausgabe vorhanden ist, schlägt die Mapping-Anweisung fehl.

- **Match.** Die Anweisung erwartet, ein Match für das Element in den Ausgabeelementen zu finden. Die Anweisung schlägt fehl, wenn das Element im Ausgabehierarchiedokument nicht vorhanden ist.
- **Match oder Hinzufügen.** Wenn sich im Ausgabehierarchiedokument ein übereinstimmendes Dokument befindet, fügt die Datenprozessor-Umwandlung kein Ausgabeelement hinzu. Wenn sich das Dokument nicht im Ausgabehierarchiedokument befindet, erstellt die Umwandlung ein Ausgabeelement.

Name

Optional. Ein Name für die Anweisung. Sie können den Namen jederzeit ändern. Namen kennzeichnen die Anweisungen, damit Sie sie im Mapping-Gitter oder in einem Ereignisprotokoll wiederfinden. Anweisungsnamen brauchen nicht eindeutig zu sein.

Bei Versagen

Erforderlich. Bestimmt die Aktion, die beim Fehlschlagen der Anweisung durchgeführt wird. Wählen Sie eine der folgenden Optionen aus:

- **Überspringen.** Wenn die Anweisung fehlschlägt, überspringen Sie die Anweisung.
- **Verteilen.** Wenn die Anwendung fehlschlägt, erzwingen Sie ebenfalls das Fehlschlagen der übergeordneten Anweisung.

Ausgabe

Erforderlich. Ein XPath-Ausdruck, der den Wert des Elements in der Ausgabe-XML auf der Grundlage der Ergebnisse des Eingabe-XPath-Ausdrucks definiert.

Fehlende Eingabe überspringen

Optional. Legt fest, ob die Anweisung übersprungen werden soll, wenn für den Eingabewert kein Match vorhanden ist. Wählen Sie eine der folgenden Optionen aus:

- **Aktiviert.** Wenn sich das Element nicht im Eingabehierarchiedokument befindet, überspringt die Datenprozessor-Umwandlung die Anweisung ohne Fehler.
- **Deaktiviert.** Die Anweisung schlägt fehl, wenn sich das Element nicht im Eingabehierarchiedokument befindet.

Anweisungstyp

Erforderlich. Identifiziert die Anweisung als Map-Anweisung.

Group-Anweisungen

Eine Group-Anweisung enthält eine logische Gruppe von Anweisungen. Eine übergeordnete Group-Anweisung enthält untergeordnete Anweisungen. Die untergeordneten Anweisungen werden unterhalb der Group-Anweisung im XMap-Editor-Gitter geschachtelt.

Sie können eine Group-Anweisung verwenden, um einen allgemeinen Kontext oder eine allgemeine Bedingung für den Erfolg oder das Fehlschlagen einer Gruppe von Anweisungen bereitzustellen. Sie können eine Group-Mapping-Anweisung verwenden, wenn die Verarbeitung einer Gruppe von Anweisungen erfolgreich durchgeführt werden oder fehlschlagen soll. Sie können eine Group-Mapping-Anweisung verwenden, um eine Gruppe von Anweisungen zum Organisieren und Vereinfachen eines XMap-Gitters zu gruppieren.

Wenn Sie eine Verknüpfung zwischen einem komplexen Einzelement und einem Einzel- oder Mehrfachelement herstellen, erstellt der XMap-Editor eine Group-Anweisung. Der Wert **Max. Vorkommen** für ein Einzelement beträgt **1**.

Group-Anweisung - Beispiel

Sie möchten Daten aus einem Eingabe-XML-Dokument mit Manager-Employee-Daten zu einem Ausgabe-XML-Dokument mit Worker-Daten zuordnen. Es gibt nur einen Manager, das Eingabe-Employee-Element ist daher ein Einzelelement.

Eine Group-Mapping-Anweisung wird verwendet, wenn ein Element ein komplexes Einzelelement ist. Das Schema hat ein Employee-Einzelelement. Employee hat die untergeordneten Elemente FirstName und LastName:

```
Employee
  FirstName
  LastName
```

Sie erstellen eine Group-Mapping-Anweisung und konfigurieren Employee als Eingabe. Jede Mapping-Anweisung, die Sie in die Gruppe einbeziehen, befindet sich im Kontext von Employee.

In der folgenden Abbildung ist Anweisung 1 die Group-Anweisung:

	Name	Statement-...	Eingabe
1	Employee to Worker	Gruppe	.
2	FirstName to FirstName	Zuordnen	tns0:FirstName
3	LastName to LastName	Zuordnen	tns0:LastName

Die Eingabespalte für die Group-Anweisung zeigt, dass die Eingabe das übergeordnete Element von FirstName und LastName ist. Anweisung 2 und Anweisung 3 sind untergeordnete Anweisungen von Anweisung 1. Die untergeordneten Anweisungen werden gegenüber der übergeordneten Anweisung eingerückt angezeigt. Ordnen Sie für jedes Employee-Element das FirstName-Element und das LastName-Element der Ausgabe zu.

Eigenschaften der Group-Anweisung

Die Group-Anweisung beinhaltet Eigenschaften, die Sie zum Anpassen der Anweisung konfigurieren können. Sie können die Eingabe, Ausgabe und eine Bedingung für das Zuordnen eines Eingabeelements zu einem Ausgabeelement konfigurieren.

Die Group-Anweisung hat die folgenden Eigenschaften:

Bedingung

Optional. Ein XPath-Ausdruck, der die Eingabebedingung für die Group-Anweisung und für alle jeweils untergeordneten Anweisungen definiert. Eine Bedingung ähnelt einem Prädikatsausdruck in der Eingabespalte. Wenn Sie einen Eingabe-XPath-Ausdruck und einen Bedingungs-XPath-Ausdruck für dieselbe Mapping-Anweisung definieren, wendet die Datenprozessor-Umwandlung den Bedingungs-XPath auf das Ergebnis des Eingabe-XPath an.

Eingabe

Optional. Ein XPath-Ausdruck, bei dem es sich um 0 oder ein Element oder einen Wert handelt. Wenn der Ausdruck leer bleibt, verwendet er den aktuellen Kontext. Wenn es sich bei dem Ausdruck um mehr als einen Wert handelt, wird der erste verwendet.

Modus

Erforderlich. Legt fest, ob die Datenprozessor-Umwandlung ein Ausgabeelement hinzufügt oder ein vorhandenes Element mit einem Wert aus der Mapping-Anweisung abgleicht. Wählen Sie eine der folgenden Optionen aus:

- Hinzufügen. Erstellt ein Element im Ausgabehierarchiedokument. Wenn das Element nicht mehrmals vorkommt und derselbe Wert in der Ausgabe vorhanden ist, schlägt die Mapping-Anweisung fehl.
- Match. Die Anweisung erwartet, ein Match für das Element in den Ausgabeelementen zu finden. Die Anweisung schlägt fehl, wenn das Element im Ausgabehierarchiedokument nicht vorhanden ist.
- Match oder Hinzufügen. Wenn sich im Ausgabehierarchiedokument ein übereinstimmendes Dokument befindet, fügt die Datenprozessor-Umwandlung kein Ausgabeelement hinzu. Wenn sich das Dokument nicht im Ausgabehierarchiedokument befindet, erstellt die Umwandlung ein Ausgabeelement.

Name

Optional. Ein Name für die Anweisung. Sie können den Namen jederzeit ändern. Namen kennzeichnen die Anweisungen, damit Sie sie im Mapping-Gitter oder in einem Ereignisprotokoll wiederfinden. Anweisungsnamen brauchen nicht eindeutig zu sein.

Bei Versagen

Optional. Bestimmt die Aktion, die beim Fehlschlagen der Anweisung durchgeführt wird. Wählen Sie eine der folgenden Optionen aus:

- Überspringen. Wenn die Anweisung fehlschlägt, überspringen Sie die Anweisung.
- Verteilen. Wenn die Anwendung fehlschlägt, erzwingen Sie ebenfalls das Fehlschlagen der übergeordneten Anweisung.

Ausgabe

Optional. Ein XPath-Ausdruck, der den Wert des Elements in der Ausgabe-XML auf der Grundlage der Ergebnisse des Eingabe-XPath-Ausdrucks definiert. Wenn der Ausdruck leer bleibt, verwendet er den aktuellen Kontext.

Fehlende Eingabe überspringen

Optional. Legt fest, ob die Anweisung übersprungen werden soll, wenn für den Eingabewert kein Match vorhanden ist. Wählen Sie eine der folgenden Optionen aus:

- Aktiviert. Wenn sich das Element nicht im Eingabehierarchiedokument befindet, überspringt die Datenprozessor-Umwandlung die Anweisung ohne Fehler.
- Deaktiviert. Die Anweisung schlägt fehl, wenn sich das Element nicht im Eingabehierarchiedokument befindet.

Anweisungstyp

Erforderlich. Identifiziert die Anweisung als Group-Anweisung.

Anweisungen für Wiederholungsgruppen

Eine Repeating Group-Anweisung ist eine Group-Anweisung, die mehrere Male vorkommen kann. Die Eingabe ist ein XPath-Ausdruck, der in eine Sequenz von Werten ausgewertet werden kann.

Die Datenprozessor-Umwandlung führt die Repeating Group-Anweisung für jedes Element bzw. jeden Wert durch, der ein Ergebnis des Eingabe-XPath-Ausdrucks ist.

Wenn Sie eine Verknüpfung zwischen einem Schema-Element für wiederholende Eingabe mit einem Schema-Element für wiederholende Ausgabe herstellen, erstellt der XMap-Editor eine Repeating Group-Anweisung im Gitter. Ein Element wird wiederholt, wenn der Wert **Max. Vorkommen** für ein Element größer als 1 ist.

Beispiel für eine Repeating Group-Anweisung

Ein Eingabeschema hat die folgende Hierarchie:

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

Wenn Sie "Employee" in ein Ausgabeelement im XMap-Editor ziehen, erstellt das Developer-Tool standardmäßig eine Repeating Group-Mapping-Anweisung. Eine Wiederholungsgruppe kann Mapping-Anweisungen enthalten, die LastName und FirstName für jeden Mitarbeiter im Eingabehierarchiedokument zurückgeben.

Eigenschaften der Repeating Group-Anweisung

Die Repeating Group-Anweisung enthält Eigenschaften, die Sie zum Anpassen der Anweisung konfigurieren können. Sie können die Eingabe, Ausgabe und eine Bedingung für das Zuordnen eines Ausgabeelements zu einem Ausgabeelement konfigurieren.

Die Repeating Group-Anweisung hat folgende Eigenschaften:

Bedingung

Optional. Ein XPath-Ausdruck, der eine Bedingung für das Zuordnen des Elements definiert. Eine Bedingung ähnelt einem Prädikatsausdruck in der Eingabespalte. Wenn Sie einen Eingabe-XPath-Ausdruck und einen Bedingungs-XPath-Ausdruck für dieselbe Mapping-Anweisung definieren, wendet die Datenprozessor-Umwandlung den Bedingungs-XPath auf das Ergebnis des Eingabe-XPath an.

Eingabe

Erforderlich. Ein XPath-Ausdruck, bei dem es sich um eine Folge von Knoten oder Werten handelt.

Modus

Erforderlich. Legt fest, ob die Datenprozessor-Umwandlung ein Ausgabeelement hinzufügt oder ein vorhandenes Element mit einem Wert aus der Mapping-Anweisung abgleicht. Wählen Sie eine der folgenden Optionen aus:

- Hinzufügen. Erstellt ein Element im Ausgabehierarchiedokument. Wenn das Element nicht mehrmals vorkommt und derselbe Wert in der Ausgabe vorhanden ist, schlägt die Mapping-Anweisung fehl.
- Match. Die Anweisung erwartet, ein Match für das Element in den Ausgabeelementen zu finden. Die Anweisung schlägt fehl, wenn das Element im Ausgabehierarchiedokument nicht vorhanden ist.
- Match oder Hinzufügen. Wenn sich im Ausgabehierarchiedokument ein übereinstimmendes Dokument befindet, fügt die Datenprozessor-Umwandlung kein Ausgabeelement hinzu. Wenn sich das Dokument nicht im Ausgabehierarchiedokument befindet, erstellt die Umwandlung ein Ausgabeelement.

Name

Optional. Ein Name für die Anweisung. Sie können den Namen jederzeit ändern. Namen kennzeichnen die Anweisungen, damit Sie sie im Mapping-Gitter oder in einem Ereignisprotokoll wiederfinden. Anweisungsnamen brauchen nicht eindeutig zu sein.

Bei Versagen

Optional. Bestimmt die Aktion, die beim Fehlschlagen der Anweisung durchgeführt wird. Wählen Sie eine der folgenden Optionen aus:

- Überspringen Sie die Iteration. Wenn eine innerhalb der Repeating Group geschachtelte Anweisung auf **Verteilen** festgelegt ist, wird die aktuelle Iteration der Repeating Group übersprungen.

- Verteilen. Wenn die Anwendung fehlschlägt, erzwingen Sie ebenfalls das Fehlschlagen der übergeordneten Anweisung.

Ausgabe

Optional. Ein XPath-Ausdruck, der den Wert des Knotens in der Ausgabe-XML auf der Grundlage der Ergebnisse des Eingabe-XPath-Ausdrucks definiert.

Fehlende Eingabe überspringen

Optional. Legt fest, ob die Anweisung übersprungen werden soll, wenn für den Eingabewert kein Match vorhanden ist. Wählen Sie eine der folgenden Optionen aus:

- Aktiviert. Wenn sich das Element nicht im Eingabehierarchiedokument befindet, überspringt die Datenprozessor-Umwandlung die Anweisung ohne Fehler.
- Deaktiviert. Die Anweisung schlägt fehl, wenn sich das Element nicht im Eingabehierarchiedokument befindet.

Anweisungstyp

Erforderlich. Identifiziert die Anweisung als eine Repeating Group-Anweisung.

Router-Anweisungen

Eine Router-Anweisung liefert Alternativen für die Mapping-Logik basierend auf den Bedingungen in einem Eingabedokument.

Die Router-Anweisung enthält eine oder mehrere Option-Anweisungen und kann eine Default-Anweisung enthalten. Wenn die Datenprozessor-Umwandlung eine Router-Anweisung ausführt, testet sie jede darunter geschachtelte Router-Anweisung.

Die erste übereinstimmende Option-Anweisung wird ausgeführt. Die Option-Anweisung kann eine oder mehrere untergeordnete Anweisungen eines beliebigen Typs enthalten. Wenn keine Option-Anweisung übereinstimmt, wird die Default-Anweisung durchgeführt. Wenn keine Default-Anweisung vorhanden ist, schlägt die Router-Anweisung fehl.

In einer Router-Gruppe können mehrere Option-Anweisungen konfiguriert werden. Die Datenprozessor-Umwandlung führt die erste Option-Anweisung durch, die sie akzeptiert. Die Datenprozessor-Umwandlung führt keine Option-Anweisungen unterhalb dieser Anweisung in der Gruppe durch. Wenn die Datenprozessor-Umwandlung keine Option-Anweisung akzeptiert, testet sie die nächste Option-Anweisung.

Wenn die Umwandlung keine Option-Anweisung akzeptiert und keine Default-Anweisung vorliegt, schlägt die Router-Anweisung fehl. Wenn die Option-Anweisung eine Bedingung enthält, die "True" zurückgibt, jedoch die Mapping-Anweisungen darin fehlschlagen und den Fehlschlag verteilen, schlägt die Router-Anweisung fehl. Sie können die Mapping-Anweisung so konfigurieren, dass die Router-Anweisung übersprungen wird, falls sie fehlschlägt.

Wenn ein Router keine Option-Anweisung, aber eine Default-Anweisung enthält, wird immer die Default-Anweisung durchgeführt.

Beispiel für Router-Anweisung

Eine XMap enthält eine "Repeating Group" unter dem Kontext "Employee". Die erste untergeordnete Anweisung in der Gruppe ist eine Router-Anweisung. Die Router-Anweisung hat eine Option-Anweisung. Die Option-Anweisung enthält eine Bedingung, die prüft, ob der Wert der Rolle gleich dem Wert "Manager" ist. Wenn die Rolle gleich "Manager" ist, gibt die Option-Anweisung "True" aus. Das Mapping evaluiert die unter der Option-Anweisung geschachtelte Run XMap-Anweisung. Die Datenprozessor-Umwandlung ruft die EmployeeToWorker-XMap auf, um dem "Manager" Elemente zuzuordnen.

Wenn die Rolle nicht gleich "Manager" ist, ist die Default-Anweisung "True". Das Mapping evaluiert die nächste Anweisung für die Default-Option. Die Standard-Mapping-Anweisung ruft die EmployeeToWorker-XMap auf, um dem "Worker" Elemente zuzuordnen.

Die folgende Abbildung zeigt die Router-Anweisung mit einer Option-Anweisung und einer Default-Anweisung:

Employee to Worker	Wiederholende Gruppe	tns0:Employee	<input type="checkbox"/>	Iteration überspringen	
Employee	Router		<input type="checkbox"/>	Überspringen	
Employee to Manager	Option	tns0:Roles="Manager"	<input checked="" type="checkbox"/>	Verteilen	tns0:Manager
EmployeeToWorker	EmployeeToWorker		<input type="checkbox"/>	Verteilen	
EmployeeToWorker	Standard		<input type="checkbox"/>	Verteilen	
EmployeeToWorker	EmployeeToWorker		<input type="checkbox"/>	Verteilen	tns0:Worker

Eigenschaften der Router-Anweisung

Die Router-Anweisung enthält Eigenschaften, die Sie zum Anpassen der Anweisung konfigurieren können. Sie können die Eingabe, Ausgabe und eine Bedingung für das Zuordnen eines Eingabeelements zu einem Ausgabeelement konfigurieren.

Die Router-Anweisung hat die folgenden Eigenschaften:

Bedingung

Optional. Ein XPath-Ausdruck, der eine Bedingung für das Zuordnen des Elements definiert. Eine Bedingung ähnelt einem Prädikatsausdruck in der Eingabespalte. Wenn Sie einen Eingabe-XPath-Ausdruck und einen Bedingungs-XPath-Ausdruck für dieselbe Mapping-Anweisung definieren, wendet die Datenprozessor-Umwandlung den Bedingungs-XPath auf das Ergebnis des Eingabe-XPath an.

Default

Erforderlich. Der zu verwendende Standardwert, wenn ein Element bei der Eingabe fehlt. Sie können zum Beispiel einen Standardwert festlegen, um einen Zähler zu initialisieren.

Eingabe

Erforderlich. Ein XPath-Ausdruck, bei dem es sich um eine Folge von Knoten oder Werten handelt.

Modus

Erforderlich. Legt fest, ob die Datenprozessor-Umwandlung ein Ausgabeelement hinzufügt oder ein vorhandenes Element mit einem Wert aus der Mapping-Anweisung abgleicht. Wählen Sie eine der folgenden Optionen aus:

- **Hinzufügen.** Erstellt ein Element im Ausgabehierarchiedokument. Wenn das Element nicht mehrmals vorkommt und derselbe Wert in der Ausgabe vorhanden ist, schlägt die Mapping-Anweisung fehl.
- **Match.** Die Anweisung erwartet, ein Match für das Element in den Ausgabeelementen zu finden. Die Anweisung schlägt fehl, wenn das Element im Ausgabehierarchiedokument nicht vorhanden ist.
- **Match oder Hinzufügen.** Wenn sich im Ausgabehierarchiedokument ein übereinstimmendes Dokument befindet, fügt die Datenprozessor-Umwandlung kein Ausgabeelement hinzu. Wenn sich das Dokument nicht im Ausgabehierarchiedokument befindet, erstellt die Umwandlung ein Ausgabeelement.

Name

Optional. Ein Name für die Anweisung. Sie können den Namen jederzeit ändern. Namen kennzeichnen die Anweisungen, damit Sie sie im Mapping-Gitter oder in einem Ereignisprotokoll wiederfinden. Anweisungsamen brauchen nicht eindeutig zu sein.

Bei Versagen

Optional. Bestimmt die Aktion, die beim Fehlschlagen der Anweisung durchgeführt wird. Wählen Sie eine der folgenden Optionen aus:

- Überspringen. Wenn die Anweisung fehlschlägt, überspringen Sie die Anweisung.
- Verteilen. Wenn die Anwendung fehlschlägt, erzwingen Sie ebenfalls das Fehlschlagen der übergeordneten Anweisung.

Ausgabe

Erforderlich. Ein XPath-Ausdruck, der den Wert des Elements in der Ausgabe-XML auf der Grundlage der Ergebnisse des Eingabe-XPath-Ausdrucks definiert. Das Ausgabefeld liefert den Kontext für die untergeordneten Anweisungen.

Anweisungstyp

Erforderlich. Identifiziert die Anweisung als eine Router-Anweisung.

Option-Anweisungen

Die Option-Anweisung, die die Bedingung für das Zuordnen des Eingabeknotens zum Ausgabeknoten liefert. Eine Option-Anweisung muss unterhalb einer Router-Anweisung geschachtelt sein. Die Router-Anweisung muss einen Eingabe-XPath-Ausdruck oder einen Bedingungs-XPath-Ausdruck enthalten. Die Option-Anweisung kann einen Eingabe-XPath-Ausdruck und einen Bedingungs-XPath-Ausdruck enthalten.

Die Datenprozessor-Umwandlung akzeptiert eine Option-Anweisung, wenn die Ergebnisse des Eingabefeldausdrucks und eines Bedingungsfeldausdrucks einen einzelnen Knoten ergeben. Wenn Sie keinen Eingabefeldausdruck definieren, akzeptiert die Datenprozessor-Umwandlung die Option-Anweisung, wenn der Bedingungsfeldausdruck "True" ergibt.

Die Option-Anweisung kann eine oder mehrere untergeordnete Anweisungen beliebiger Art enthalten, etwa Map, Group, Repeating Group, Run XMap und andere Router-Anweisungen. Eine Option-Anweisung kann zum Beispiel die folgende Bedingung enthalten:

```
EmployeeID="100"
```

Wenn EmployeeID gleich 100 ist, ist die Bedingung "True". Die untergeordnete Anweisung im Gitter legt fest, welche Mapping-Anweisung evaluiert wird, wenn die Bedingung "True" ist.

Eigenschaften der Option-Anweisung

Die Option-Anweisung enthält Eigenschaften, die Sie zum Anpassen der Anweisung konfigurieren können. Sie können die Eingabe, Ausgabe und eine Bedingung für das Zuordnen eines Eingabeelements zu einem Ausgabeelement konfigurieren.

Die Option-Anweisung hat folgende Eigenschaften:

Bedingung

Erforderlich, wenn keine Eingabe definiert ist. Ein XPath-Ausdruck, der eine Bedingung für das Zuordnen des Elements definiert. Eine Bedingung ähnelt einem Prädikatsausdruck in der Eingabespalte. Die Datenprozessor-Umwandlung wendet den Bedingungs-XPath auf das Ergebnis des Eingabe-XPath an.

Eingabe

Erforderlich, wenn keine Bedingung definiert ist. Ein XPath-Ausdruck, der das Eingabeelement definiert. Der Ausdruck kann ein Knoten oder Wert sein.

Modus

Optional. Legt fest, ob die Datenprozessor-Umwandlung ein Ausgabeelement hinzufügt oder ein vorhandenes Element mit einem Wert aus der Mapping-Anweisung abgleicht. Wählen Sie eine der folgenden Optionen aus:

- Hinzufügen. Erstellt ein Element im Ausgabehierarchiedokument. Wenn das Element nicht mehrmals vorkommt und derselbe Wert in der Ausgabe vorhanden ist, schlägt die Mapping-Anweisung fehl.
- Match. Die Anweisung erwartet, ein Match für das Element in den Ausgabeelementen zu finden. Die Anweisung schlägt fehl, wenn das Element im Ausgabehierarchiedokument nicht vorhanden ist.
- Match oder Hinzufügen. Wenn sich im Ausgabehierarchiedokument ein übereinstimmendes Dokument befindet, fügt die Datenprozessor-Umwandlung kein Ausgabeelement hinzu. Wenn sich das Dokument nicht im Ausgabehierarchiedokument befindet, erstellt die Umwandlung ein Ausgabeelement.

Name

Optional. Ein Name für die Anweisung. Sie können den Namen jederzeit ändern. Namen kennzeichnen die Anweisungen, damit Sie sie im Mapping-Gitter oder in einem Ereignisprotokoll wiederfinden. Anweisungsnamen brauchen nicht eindeutig zu sein.

Bei Versagen

Optional. Bestimmt die Aktion, die beim Fehlschlagen der Anweisung durchgeführt wird. Wählen Sie eine der folgenden Optionen aus:

- Überspringen. Wenn die Anweisung fehlschlägt, überspringen Sie die Anweisung.
- Verteilen. Wenn die Anwendung fehlschlägt, erzwingen Sie ebenfalls das Fehlschlagen der übergeordneten Anweisung.

Ausgabe

Optional. Ein XPath-Ausdruck, der den Wert des Elements in der Ausgabe-XML auf der Grundlage der Ergebnisse des Eingabe-XPath-Ausdrucks definiert.

Anweisungstyp

Erforderlich. Identifiziert die Anweisung als eine Option-Anweisung.

Default-Anweisungen

Eine Default-Anweisung ist eine untergeordnete Anweisung einer Router-Anweisung. Die Router-Anweisung enthält eine oder mehrere Option-Anweisungen und kann eine Default-Anweisung enthalten. Die Datenprozessor-Umwandlung führt die Default-Anweisung durch, wenn keine der Option-Anweisungen angewendet wird.

Sie können nur eine Default-Anweisung in einer Router-Anweisungsgruppe definieren. Die Default-Anweisung muss die letzte Anweisung für die Router-Anweisungsgruppe sein. Die Default-Anweisung darf keine Eingabe- oder Bedingungs-XPath-Ausdrücke enthalten.

Eigenschaften der Default-Anweisung

Die Default-Anweisung enthält Eigenschaften, die Sie zum Anpassen der Anweisung konfigurieren können. Sie können den Standardwert konfigurieren, wenn ein Eingabeelement fehlt.

Die Default-Anweisung hat die folgenden Eigenschaften:

Default

Erforderlich. Der zu verwendende Standardwert, wenn ein Element bei der Eingabe fehlt. Sie können zum Beispiel einen Standardwert festlegen, um einen Zähler zu initialisieren.

Modus

Optional. Legt fest, ob die Datenprozessor-Umwandlung ein Ausgabeelement hinzufügt oder ein vorhandenes Element mit einem Wert aus der Mapping-Anweisung abgleicht. Wählen Sie eine der folgenden Optionen aus:

- Hinzufügen. Erstellt ein Element im Ausgabehierarchiedokument. Wenn das Element nicht mehrmals vorkommt und derselbe Wert in der Ausgabe vorhanden ist, schlägt die Mapping-Anweisung fehl.
- Match. Die Anweisung erwartet, ein Match für das Element in den Ausgabeelementen zu finden. Die Anweisung schlägt fehl, wenn das Element im Ausgabehierarchiedokument nicht vorhanden ist.
- Match oder Hinzufügen. Wenn sich im Ausgabehierarchiedokument ein übereinstimmendes Dokument befindet, fügt die Datenprozessor-Umwandlung kein Ausgabeelement hinzu. Wenn sich das Dokument nicht im Ausgabehierarchiedokument befindet, erstellt die Umwandlung ein Ausgabeelement.

Name

Optional. Ein Name für die Anweisung. Sie können den Namen jederzeit ändern. Namen kennzeichnen die Anweisungen, damit Sie sie im Mapping-Gitter oder in einem Ereignisprotokoll wiederfinden. Anweisungsnamen brauchen nicht eindeutig zu sein.

Ausgabe

Optional. Ein XPath-Ausdruck, der den Wert des Knotens in der Ausgabe-XML auf der Grundlage der Ergebnisse des Eingabe-XPath-Ausdrucks definiert.

Anweisungstyp

Erforderlich. Identifiziert die Anweisung als Default-Anweisung.

Run XMap-Anweisungen

Eine Run XMap-Anweisung ruft eine andere XMap auf.

Wenn Sie eine Run XMap-Mapping-Anweisung erstellen, listet das Developer-Tool die XMap-Objekte in der Umwandlung auf. Wählen Sie das XMap-Objekt aus, das aufgerufen werden soll. Das Developer-Tool erstellt eine Mapping-Anweisung mit dem XMap-Namen im Feld **Anweisungs-Typ**.

Die Eingabe- und Ausgabe-Root-Elemente im aufgerufenen XMap-Objekt müssen denselben Typ haben wie die Eingabe- und Ausgabewerte, die aus dem aufrufenden XMap-Objekt übergeben werden. Sie können eine XMap aufrufen, um Mapping-Logik auszuführen, die wiederholt wird.

Eigenschaften der Run XMap-Anweisung

Die Run XMap-Anweisung enthält Eigenschaften, die Sie zum Anpassen der Anweisung konfigurieren können. Sie können die Eingabe, Ausgabe und eine Bedingung für das Zuordnen eines Eingabeelements zu einem Ausgabeelement konfigurieren.

Die Run XMap-Anweisung hat die folgenden Eigenschaften:

Bedingung

Optional. Ein XPath-Ausdruck, der eine Bedingung für das Zuordnen des Elements definiert. Eine Bedingung ähnelt einem Prädikatsausdruck in der Eingabespalte. Wenn Sie einen Eingabe-XPath-Ausdruck und einen Bedingungs-XPath-Ausdruck für dieselbe Mapping-Anweisung definieren, wendet die Datenprozessor-Umwandlung den Bedingungs-XPath auf das Ergebnis des Eingabe-XPath an.

Eingabe

Optional. Ein XPath-Ausdruck, bei dem es sich um eine Folge von Knoten oder Werten handelt. Der Mapping-Anweisungstyp bestimmt, wie die Datenprozessor-Umwandlung die Knoten oder Werte beim Mapping verwendet.

Modus

Erforderlich. Legt fest, ob die Datenprozessor-Umwandlung ein Ausgabeelement hinzufügt oder ein vorhandenes Element mit einem Wert aus der Mapping-Anweisung abgleicht. Wählen Sie eine der folgenden Optionen aus:

- Hinzufügen. Erstellt ein Element im Ausgabehierarchiedokument. Wenn das Element nicht mehrmals vorkommt und derselbe Wert in der Ausgabe vorhanden ist, schlägt die Mapping-Anweisung fehl.
- Match. Die Anweisung erwartet, ein Match für das Element in den Ausgabeelementen zu finden. Die Anweisung schlägt fehl, wenn das Element im Ausgabehierarchiedokument nicht vorhanden ist.
- Match oder Hinzufügen. Wenn sich im Ausgabehierarchiedokument ein übereinstimmendes Dokument befindet, fügt die Datenprozessor-Umwandlung kein Ausgabeelement hinzu. Wenn sich das Dokument nicht im Ausgabehierarchiedokument befindet, erstellt die Umwandlung ein Ausgabeelement.

Name

Optional. Ein Name für die Anweisung. Sie können den Namen jederzeit ändern. Namen kennzeichnen die Anweisungen, damit Sie sie im Mapping-Gitter oder in einem Ereignisprotokoll wiederfinden. Anweisungsnamen brauchen nicht eindeutig zu sein.

Bei Versagen

Erforderlich. Bestimmt die Aktion, die beim Fehlschlagen der Anweisung durchgeführt wird. Wählen Sie eine der folgenden Optionen aus:

- Überspringen. Wenn die Anweisung fehlschlägt, überspringen Sie die Anweisung.
- Verteilen. Wenn die Anwendung fehlschlägt, erzwingen Sie ebenfalls das Fehlschlagen der übergeordneten Anweisung.

Ausgabe

Optional. Ein XPath-Ausdruck, der den Wert des Knotens in der Ausgabe-XML auf der Grundlage der Ergebnisse des Eingabe-XPath-Ausdrucks definiert.

Fehlende Eingabe überspringen

Erforderlich. Legt fest, ob die Anweisung übersprungen werden soll, wenn für den Eingabewert kein Match vorhanden ist. Wählen Sie eine der folgenden Optionen aus:

- Aktiviert. Wenn sich das Element nicht im Eingabehierarchiedokument befindet, überspringt die Datenprozessor-Umwandlung die Anweisung ohne Fehler.
- Deaktiviert. Die Anweisung schlägt fehl, wenn sich das Element nicht im Eingabehierarchiedokument befindet.

Anweisungstyp

Erforderlich. Identifiziert die Anweisung als Run XMap-Anweisung.

RunMapplet-Anweisung

Eine RunMapplet-Anweisung ruft ein Mapplet auf.

Beim Erstellen einer RunMapplet-Mapping-Anweisung erstellt das Developer tool eine Liste der Mapplet-Referenzobjekte, die mit der Umwandlung verbunden sind. Wählen Sie das Mapplet aus, das aufgerufen

werden soll. Das Developer tool erstellt eine XMap-Anweisung mit dem Mapplet-Namen im Feld **Anweisungstyp**.

Die Eingabe- und Ausgabeports im aufgerufenen Mapplet müssen denselben Typ aufweisen wie die Werte, die aus dem aufrufenden XMap übergeben werden. Sie können ein Mapplet aufrufen, um Aufgaben durchzuführen, wie z. B. Datenmaskierungs-, Datenqualitäts-, Daten-Lookup- oder andere Aktivitäten, die sich in der Regel auf relationale Umwandlungen beziehen. Hierbei ist es nicht erforderlich, Daten in das relationale Format und dann zurück in das hierarchische Format zu konvertieren.

Hinweis: Die RunMapplet-Aktion kann nur verwendet werden, um passive Mapplets aufzurufen.

Eigenschaften von RunMapplet-Anweisungen

Die RunMapplet-Anweisung enthält Eigenschaften, die Sie zum Anpassen der Anweisung konfigurieren können. Sie können die Eingabe und Ausgabe sowie eine Bedingung konfigurieren, um die RunMapplet-Anweisung auszuführen. Die RunMapplet-Anweisung kann die MappletInput-Anweisung und die MappletOutput-Anweisung enthalten.

Die RunMapplet-Anweisung hat die folgenden Eigenschaften:

Bedingung

Optional. Ein XPath-Ausdruck, der eine Bedingung für das Ausführen der RunMapplet-Anweisung definiert. Eine Bedingung ähnelt einem Prädikatsausdruck in der Eingabespalte. Wenn Sie einen Eingabe-XPath-Ausdruck und einen Bedingungs-XPath-Ausdruck für dieselbe Mapping-Anweisung definieren, wendet die Datenprozessorumwandlung den Bedingungs-XPath auf das Ergebnis des Eingabe-XPath an.

Eingabe

Optional. Ein XPath-Ausdruck, bei dem es sich um eine Folge von Knoten oder Werten handelt. Der Mapping-Anweisungstyp bestimmt, wie die Datenprozessorumwandlung die Knoten oder Werte im Mapping verwendet.

Modus

Erforderlich. Legt fest, ob die Datenprozessorumwandlung ein Ausgabeelement hinzufügt oder ein vorhandenes Element mit einem Wert aus der Mapping-Anweisung abgleicht. Wählen Sie eine der folgenden Optionen aus:

- Hinzufügen. Erstellt ein Element im Ausgabehierarchiedokument. Wenn das Element nicht mehrmals vorkommt und derselbe Wert in der Ausgabe vorhanden ist, schlägt die Mapping-Anweisung fehl.
- Match. Die Anweisung erwartet, ein Match für das Element in den Ausgabeelementen zu finden. Die Anweisung schlägt fehl, wenn das Element im Ausgabehierarchiedokument nicht vorhanden ist.
- Match oder Hinzufügen. Wenn sich im Ausgabehierarchiedokument ein übereinstimmendes Dokument befindet, fügt die Datenprozessorumwandlung kein Ausgabeelement hinzu. Wenn sich das Dokument nicht im Ausgabehierarchiedokument befindet, erstellt die Umwandlung ein Ausgabeelement.

Name

Optional. Ein Name für die Anweisung. Sie können den Namen jederzeit ändern. Namen kennzeichnen die Anweisungen, damit Sie sie im Mapping-Gitter oder in einem Ereignisprotokoll wiederfinden. Anweisungsnamen brauchen nicht eindeutig zu sein.

Bei Versagen

Optional. Bestimmt die Aktion, die beim Fehlschlagen der Anweisung durchgeführt wird. Wählen Sie eine der folgenden Optionen aus:

- Überspringen. Wenn die Anweisung fehlschlägt, überspringen Sie die Anweisung.

- Verteilen. Wenn die Anwendung fehlschlägt, erzwingen Sie ebenfalls das Fehlschlagen der übergeordneten Anweisung.

Ausgabe

Erforderlich. Ein XPath-Ausdruck, der den Wert des Knotens in der Ausgabehierarchie auf der Grundlage der Ergebnisse des Eingabe-XPath-Ausdrucks definiert.

Fehlende Eingabe überspringen

Optional. Legt fest, ob die Anweisung übersprungen werden soll, wenn für den Eingabewert kein Match vorhanden ist. Wählen Sie eine der folgenden Optionen aus:

- Aktiviert. Wenn sich das Element nicht im Eingabehierarchiedokument befindet, überspringt die Datenprozessorumwandlung die Anweisung ohne Fehler.
- Deaktiviert. Die Anweisung schlägt fehl, wenn sich das Element nicht im Eingabehierarchiedokument befindet.

Anweisungstyp

Erforderlich. Identifiziert die Anweisung als eine RunMapplet-Anweisung. Das Feld identifiziert den Namen des referenzierten Mapplets.

MappletInput-Anweisung

Die MappletInput-Anweisung enthält Eigenschaften, die Sie zum Anpassen der Anweisung konfigurieren können. Sie können die Eingabe konfigurieren, um dem Mapplet-Eingabeport ein Eingabeelement zuzuordnen. Eine oder mehrere MappletInput-Anweisungen können unter der RunMapplet-Anweisung geschachtelt werden.

Die MappletInput-Anweisung wird ausgeführt, um einen Wert bereitzustellen, der an die Mapplet-Eingabe übergeben werden kann. Die Werte in der RunMapplet-Anweisung werden an die Mapplet-Ports in der gleichen Reihenfolge übergeben, wie sie in der RunMapplet-Anweisung aufgelistet sind. Wenn eine Anweisung übersprungen wird, wird ein Nullwert an den Mapplet-Eingabeport übergeben.

Eine MappletInput-Anweisung übergibt einen einzelnen Wert. Wenn RunMapplet ausgeführt wird, werden die Werte aus den geschachtelten MappletInput-Anweisungen gesammelt und in der gleichen Reihenfolge wie die MappletInput-Anweisungen an das Mapplet übergeben.

Eigenschaften von MappletInput-Anweisungen

Die MappletInput-Anweisung enthält Eigenschaften, die Sie zum Anpassen der Anweisung konfigurieren können.

Die RunMapplet-Anweisung hat die folgenden Eigenschaften:

Standard

Optional. Der zu verwendende Standardwert, wenn ein Element aus der Mapplet-Port-Eingabe fehlt.

Eingabe

Erforderlich. Ein XPath-Ausdruck, bei dem es sich um eine Folge von Knoten oder Werten handelt. Die Werte werden an die Mapplet-Eingabeports übergeben.

Name

Optional. Ein Name für die Anweisung. Sie können den Namen jederzeit ändern. Namen kennzeichnen die Anweisungen, damit Sie sie im Mapping-Gitter oder in einem Ereignisprotokoll wiederfinden. Anweisungsamen brauchen nicht eindeutig zu sein.

Fehlende Eingabe überspringen

Optional. Legt fest, ob die Anweisung übersprungen wird, wenn für den Eingabewert kein Match des Mapplet-Eingabeports vorhanden ist. Wählen Sie eine der folgenden Optionen aus:

- Aktiviert. Wenn sich das Element nicht im Eingabehierarchiedokument befindet, überspringt die Datenprozessorumwandlung die Anweisung ohne Fehler.
- Deaktiviert. Die Anweisung schlägt fehl, wenn sich das Element nicht im Eingabehierarchiedokument befindet.

MappletOutput-Anweisung

Die MappletOutput-Anweisung wird ausgeführt, um einen von der Mapplet-Ausgabe übergebenen Wert zu erhalten. Die Werte werden von den Mapplet-Ports in der gleichen Reihenfolge an die RunMapplet-Ausgabeports übergeben, wie sie in der RunMapplet-Anweisung aufgelistet sind. Ein oder mehrere MappletOutput-Anweisungen können unter der RunMapplet-Anweisung geschachtelt werden.

Eine MappletOutput-Anweisung erhält einen einzelnen Wert. Wenn RunMapplet ausgeführt wird, werden die Werte aus dem Mapplet gesammelt und in der gleichen Reihenfolge wie die MappletOutput-Anweisungen an die geschachtelten MappletOutput-Anweisungen übergeben.

Eigenschaften von MappletOutput-Anweisungen

Die MappletOutput-Anweisung enthält Eigenschaften, die Sie zum Anpassen der Anweisung konfigurieren können.

Die RunMapplet-Anweisung hat die folgenden Eigenschaften:

Standard

Optional. Der zu verwendende Standardwert, wenn die Datenprozessorumwandlung ohne eine bestimmte Mapping-Eingabe getestet wird. Der Standardwert wird nicht verwendet, wenn eine Mapping-Eingabe vorhanden ist.

Modus

Optional. Legt fest, ob die Umwandlung ein Ausgabeelement hinzufügt oder ein vorhandenes Element mit einem Wert aus der Mapping-Anweisung abgleicht. Wählen Sie eine der folgenden Optionen aus:

- Hinzufügen. Erstellt ein Element im Ausgabehierarchiedokument. Wenn das Element nicht mehrmals vorkommt und derselbe Wert in der Ausgabe vorhanden ist, schlägt die Mapping-Anweisung fehl.
- Match. Die Anweisung erwartet, ein Match für das Element in den Ausgabeelementen zu finden. Die Anweisung schlägt fehl, wenn das Element im Ausgabehierarchiedokument nicht vorhanden ist.
- Match oder Hinzufügen. Wenn sich im Ausgabehierarchiedokument ein übereinstimmendes Dokument befindet, fügt die Datenprozessorumwandlung kein Ausgabeelement hinzu. Wenn sich das Dokument nicht im Ausgabehierarchiedokument befindet, erstellt die Umwandlung ein Ausgabeelement.

Name

Optional. Ein Name für die Anweisung. Sie können den Namen jederzeit ändern. Namen kennzeichnen die Anweisungen, damit Sie sie im Mapping-Gitter oder in einem Ereignisprotokoll wiederfinden. Anweisungsnamen brauchen nicht eindeutig zu sein.

Ausgabe

Erforderlich. Ein XPath-Ausdruck, bei dem es sich um eine Folge von Knoten oder Werten handelt. Die Werte des Mapplet-Ausgabeports werden an die Abfolge von Knoten übergeben.

Erstellen einer XMap

Wählen Sie zum Erstellen eines Datenprozessor-XMap-Objekts die Eingabe- und Ausgabeschemata aus und fügen Sie Mapping-Anweisungen hinzu.

1. Erstellen Sie in der Ansicht **Objekte** der Datenprozessor-Umwandlung eine XMap. Wählen Sie ein Eingabeschema, eine Beispielquelle und ein Ausgabeschema aus.
2. Öffnen Sie den XMap-Editor und klicken Sie auf das XMap-Objekt.
3. Um eine Map-, Group- oder Repeating Group-Mapping-Anweisung zu erstellen, führen Sie im XMap-Editor ein Drag & Drop von einem Knoten im Eingabehierarchieschema zu einem Knoten im Ausgabehierarchieschema durch.

Der XMap-Editor erstellt eine Mapping-Verknüpfung zwischen den Knoten. Die Mapping-Anweisung wird im Gitter angezeigt. Der XMap-Editor füllt die Felder für die Mapping-Anweisung automatisch aus.

4. Um eine bedingte Logik im Gitter zu erstellen, fügen Sie eine Router-Mapping-Anweisung wie folgt hinzu:
 - a. Erstellen Sie unter der Router-Mapping-Anweisung Option-Mapping-Anweisungen. Ziehen Sie Eingabe- und Ausgabeschemaknoten und fügen Sie sie in Option-Anweisungsfelder im Gitter ein.
 - b. Erstellen Sie unter der Router-Mapping-Anweisung eine Default-Mapping-Anweisung, um festzulegen, was passiert, wenn keine Option-Mapping-Anweisung angewendet wird.
 - c. Erstellen Sie unter der Option-Mapping-Anweisung Map-Mapping-Anweisungen, um Bedingungen für das Zuordnen der Eingabeknoten zu den Ausgabeknoten festzulegen.
5. Um gemeinsamen Kontext für eine Gruppe von Anweisungen bereitzustellen, fügen Sie eine Group-Mapping-Anweisung hinzu. Nest Map-Mapping-Anweisungen unter der Group-Mapping-Anweisung.
6. Um ein anderes XMap-Objekt aufzurufen, fügen Sie eine Run XMap-Anweisung hinzu.
7. Um den Kontext und die Logik für eine Mapping-Anweisung zu ändern, bearbeiten Sie die Eigenschaften der Mapping-Anweisung wie folgt:
 - a. Stufen Sie Anweisungen als untergeordnete Anweisungen tiefer oder Anweisungen als übergeordnete Anweisungen höher.
 - b. Erstellen Sie XPath-Ausdrücke, um den Kontext zu ändern, oder fügen Sie Vorhersagen mit dem XPath-Editor hinzu.

Verwenden des XMap-Editor-Gitters

Wenn Sie einen Knoten vom Eingabeschema in das Ausgabeschema ziehen, erstellt das Developer-Tool eine Map-, Group- oder Repeating Group-Mapping-Anweisung im Gitter. Sie können die Mapping-Anweisung aktualisieren. Die Eingabe- und Ausgabefelder enthalten die Elemente aus dem Schema. Um Mapping-Anweisungen für den Kontext zu erstellen oder Router-Optionen zu definieren, können Sie Anweisungen im Gitter eingeben.

Wenn Sie eine Mapping-Anweisung im Gitter auswählen, markiert der XMap-Editor die Knoten aus den Eingabe- und Ausgabeschemata, die in der Gitteranweisung enthalten sind.

Sie können eine Anweisung in einer Zeile des Gitters kopieren und in einer anderen Zeile einfügen. Wenn die Zeile für den Speicherort nicht gültig ist, in den Sie sie kopieren, zeigt der XMap-Editor ein Dialogfeld mit einem Validierungsfehler an. Sie können die Eingabe- und Ausgabe-XPath-Anweisungen im Dialogfeld ändern, um den Kontext der Mapping-Anweisung anzupassen. Wahlweise können Sie auch die Eingabe- und Ausgabe-XPath-Felder im Gitter ändern.

Mapping-Anweisungen erstellen

Erstellen Sie Mapping-Anweisungen im XMap-Editor. Sie können Mapping-Anweisungen erstellen, indem Sie Knoten vom Eingabeschema in das Ausgabeschema ziehen, und Sie können die Anweisungen im Mapping-Anweisung-Gitter definieren.

Gehen Sie wie folgt vor, um Mapping-Anweisungen im Gitter zu definieren:

1. Um eine Mapping-Anweisung zu erstellen, klicken Sie in den Gitteroptionen auf **Neu**.
2. Wählen Sie den Mapping-Anweisungstyp aus der Liste aus.
Wenn Sie **Run XMap** wählen, zeigt das Developer-Tool eine Liste der XMap-Objekte in der Umwandlung.
3. Geben Sie einen Namen für die Mapping-Anweisung ein.
4. Um die Eingabe für die Mapping-Anweisung zu definieren, ziehen Sie ein Element aus dem Eingabeschema in das Eingabefeld. Wahlweise können Sie einen XPath-Ausdruck oder eine Konstante im Eingabefeld konfigurieren.
5. Um die Ausgabeknoten für die Mapping-Anweisung auszuwählen, ziehen Sie ein Element aus dem Ausgabeschema in das Ausgabefeld. Wahlweise können Sie einen XPath-Ausdruck im Ausgabefeld konfigurieren.
6. Um einen XPath-Ausdruck mit den XPath-Ausdruckseditor für ein Eingabe-, Ausgabe- oder Bedingungsfeld zu erstellen, klicken Sie im Feld auf die Schaltfläche **Öffnen**.
7. Zum Ändern des Typs der Mapping-Anweisung klicken Sie auf die Schaltfläche **Öffnen** im Feld **Anweisungstyp**. Wählen Sie den Mapping-Anweisungstyp aus der Liste.

Schnittstelle für das Gitter für Mapping-Anweisungen

Bearbeiten Sie Mapping-Anweisungen im Gitter für Mapping-Anweisungen im XMap-Editor. Bevor Sie die Felder im Gitter für Mapping-Anweisungen ändern können, müssen Sie eine Anweisung erstellen.

Im Gitter für Mapping-Anweisungen können Sie die folgenden Aufgaben ausführen:

- Ziehen Sie Knoten aus dem Eingabe- oder Ausgabeschema in die Mapping-Anweisungen.
- Kopieren Sie Elemente in die Felder für Mapping-Anweisungen oder geben Sie die Werte in die Felder ein.
- Sie können eine Mapping-Anweisung im Gitter nach oben oder unten verschieben. Wählen Sie die Zeile aus und klicken Sie auf den Pfeil **nach oben** oder **nach unten**.
- Sie können auf **Tiefer stufen** klicken, um eine Mapping-Anweisung unter einer anderen Anweisung zu platzieren. Klicken Sie **Höher stufen**, wenn Sie eine Anweisung in der Hierarchie nach oben verschieben möchten.
- Klicken Sie auf **Zu Zeilennummer wechseln**, um zu einer Zeile zu navigieren. Geben Sie die Zeile ein, zu der Sie navigieren möchten. Das Developer-Tool markiert die Zeile für Sie.

Verknüpfen von Schemaknoten

Sie können einen Eingabeschemaknoten mit einem Ausgabeschemaknoten durch Ziehen mit der Maus verknüpfen. Durch Ziehen und Ablegen können Sie einen einfachen Knoten einem einfachen Knoten, einen einfachen Knoten einem komplexen Knoten, einen komplexen Knoten einem einfachen Knoten oder einen komplexen Knoten einem komplexen Knoten zuordnen.

Der XMap-Editor erstellt eine Mapping-Verknüpfung zwischen dem Eingabeschemaknoten und dem Ausgabeschemaknoten. Der XMap-Editor erstellt ebenfalls eine Mapping-Anweisung im Gitter.

Sie können eine Mapping-Anweisung durch Ziehen und Ablegen von einer Zeile im Gitter in eine andere Zeile verschieben, um die XMap-Logik zu ändern. Sie können diese Methode beispielsweise verwenden, um die Reihenfolge der Option-Anweisungen innerhalb eines Routers zu ändern.

Ausschneiden und Einfügen von Mapping-Anweisungen

Sie können Mapping-Anweisungen im Mapping-Anweisung-Gitter im XMap-Editor ausschneiden und einfügen.

Sie können Anweisungen nach oben oder nach unten verschieben. Sie können Mapping-Anweisungen auch in geschachtelte Anweisungen einfügen. Sie können eine Anweisung als übergeordnete Anweisung höher stufen oder als untergeordnete Anweisung tiefer stufen.

Eingefügte Anweisungen müssen normalerweise korrigiert werden, da deren Logik oft außerhalb des Kontexts ist. Nachdem Sie eine Anweisung eingefügt haben, können Sie Felder im Mapping-Anweisung-Gitter ändern.

XPath-Ausdrücke

XPath-Ausdrücke geben bestimmte Elemente oder Knoten in hierarchischen Dokumenten an oder suchen nach Bedingungen in den Daten. Verwenden Sie XPath-Ausdrücke zum Definieren der Eingabe-, Bedingungs- oder Ausgabefelder einer Mapping-Anweisung.

XPath ist eine Syntax, die Teile eines hierarchischen Dokuments definiert. Verwenden Sie XPath, um Knoten- oder Wertesequenzen in einem hierarchischen Dokument auszuwählen. XPath umfasst eine Bibliothek von Standardfunktionen, die Sie für die Datenauswahl verwenden können.

Sie können XPath 2.0.-Ausdrücke in der Datenprozessor-Umwandlung definieren. Wenn Sie Ausgabe-XPath-Ausdrücke konfigurieren, können Sie zur Definition von Mapping-Anweisungen für den Modus "Hinzufügen" oder "Match oder Hinzufügen" eine Teilmenge der XPath 2.0-Syntax verwenden.

Weitere Informationen über XPath finden Sie in Ihrer XPath-Dokumentation.

Die folgende Tabelle beschreibt einige XPath-Ausdrücke:

XPath-Ausdruck	Beschreibung
nodename	Wählt alle untergeordneten Knoten eines gegebenen Namens im Kontext aus.
. (Punkt)	Wählt den aktuellen Knoten aus.
..	Wählt den übergeordneten Knoten des aktuellen Knotens aus.
@	Wählt ein Attribut.
/	Wählt unter Root-Knoten oder untergeordneten Knoten des aktuellen Knotens aus, sofern ein Knoten vorhergeht. Wenn der Pfad mit einem Schrägstrich (/) beginnt, so handelt es sich um einen absoluten Pfad zu einem Element.
//	Wählt Knoten überall im Dokument oder Nachfolger des aktuellen Knotens aus, sofern ein Knoten vorhergeht.

Die folgende Tabelle listet einige XPath-Ausdrücke sowie das jeweils zugehörige Ergebnis auf:

XPath-Ausdruck	Ergebnis
/bookstore	Wählt den Root-Bookstore-Knoten aus.
bookstore/book	Wählt Buchknoten aus, die untergeordnete Knoten aller Bookstore-Knoten sind.
//book	Wählt die Buchknoten im Dokument an allen Positionen aus.
bookstore//book	Wählt alle Buchknoten aus, die Nachfolger der Bookstore-Knoten sind.
/bookstore/*	Wählt alle untergeordneten Knoten des Bookstore-Root-Elements aus.
//*	Gibt eine Sequenz aller Elemente in einem Dokument aus.

Prädikate

Ein Prädikat ist ein Ausdruck, den Sie konfigurieren können, um einen Knoten in einem hierarchischen Dokument zu finden. Sie können den Ausdruck konfigurieren, um einen bestimmten Wert zu finden. Erstellen Sie ein Prädikat in einem Eingabe-, Bedingungs- oder Ausgabefeld einer Mapping-Anweisung.

Wenn Sie ein Prädikat definieren, setzen Sie den Ausdruck in eckige Klammern [] nach dem Knoten.

```
<Knoten>[Ausdruck]
```

Beispiel: Der folgende Ausdruck wählt die Buchelemente aus, die der Buchhandlung untergeordnet sind und ein Preiselement mit einem Wert von mehr als 55,00 aufweisen:

```
/bookstore/book[price>55,00]
```

Der folgende Ausdruck wählt die Titelemente der Buchelemente aus, die der Buchhandlung untergeordnet sind und ein Preiselement von mehr als 55,00 aufweisen:

```
/bookstore/book[price>55,00]/title
```

Der folgende Ausdruck wählt die Titelemente aus, die das Attribut "lang" mit dem Wert "eng" aufweisen:

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

Hinweis: Die Datenprozessor-Umwandlung kann nicht alle XPath-Anweisungen im Ausgabefeld akzeptieren, wenn Sie eine Mapping-Anweisung mit dem Modus „Hinzufügen“ oder „Match oder Hinzufügen“ konfigurieren.

XPath-Vorhersagen - Referenz

XPath-Vorhersagen suchen Übereinstimmungsknoten oder eine Abfolge von Knoten in einem hierarchischen Dokument. Ein Vorhersage-Ausdruck definiert den Wert eines Eingabe-, Bedingungs- oder Ausgabefelds einer Mapping-Anweisung. Der XPath-Ausdruck legt den Kontext fest, in dem eine Mapping-Anweisung ausgeführt wird.

Verwenden Sie die folgenden XPath-Ausdrücke in Vorhersagen, um einen Knoten oder eine Abfolge von Knoten auszuwählen:

ancestor

Wählt alle Vorgänger aus, wie zum Beispiel die übergeordneten bzw. zwei Ebenen übergeordneten Knoten des aktuellen Knotens. Der Vorhersage-Ausdruck "ancestor::book" wählt beispielsweise alle Buch-Vorgänger des aktuellen Knotens aus.

ancestor-or-self

Wählt alle Vorgänger aus, wie zum Beispiel den aktuellen Knoten sowie die übergeordneten bzw. zwei Ebenen übergeordneten Knoten des aktuellen Knotens. Der Vorhersage-Ausdruck "ancestor-or-self::book" wählt beispielsweise alle Buch-Vorgänger des aktuellen Knotens und ebenfalls den aktuellen Knoten aus.

attribute

Wählt alle Attribute des aktuellen Knotens aus. Der Vorhersage-Ausdruck "attribute::lang" wählt beispielsweise das lang-Attribut des aktuellen Knotens aus.

child

Wählt alle untergeordneten Knoten des aktuellen Knotens aus. Der Vorhersage-Ausdruck "child::book" wählt beispielsweise alle Buch-Knoten aus, bei denen es sich um untergeordnete Knoten des aktuellen Knotens handelt.

descendant

Wählt alle Nachfolger aus, wie zum Beispiel die untergeordneten bzw. zwei Ebenen untergeordneten Knoten des aktuellen Knotens. Der Vorhersage-Ausdruck "descendant::book" wählt beispielsweise alle Buch-Nachfolger des aktuellen Knotens aus.

descendant-or-self

Wählt alle Nachfolger aus, wie zum Beispiel den aktuellen Knoten sowie die untergeordneten bzw. zwei Ebenen untergeordneten Knoten des aktuellen Knotens. Der Vorhersage-Ausdruck "descendant-or-self::book" wählt beispielsweise alle Buch-Nachfolger des aktuellen Knotens sowie den aktuellen Knoten aus, wenn es sich um einen Buch-Knoten handelt.

following

Wählt alles im Dokument nach dem schließenden Tag des aktuellen Knotens aus. Der Vorhersage-Ausdruck "following::book" wählt beispielsweise alles im Dokument nach dem schließenden Tag des Buch-Knotens aus.

following-sibling

Wählt alle Gleichgeordneten nach dem aktuellen Knoten aus. Der Vorhersage-Ausdruck "following-sibling::book" wählt beispielsweise alle Gleichgeordneten im Dokument nach dem Buch-Knoten aus.

namespace

Wählt alle Namespace-Knoten des aktuellen Knotens aus.

parent

Wählt den übergeordneten Knoten des aktuellen Knotens aus. Der Vorhersage-Ausdruck "parent::book" wählt beispielsweise das lang-Attribut des aktuellen Knotens aus.

preceding

Wählt alle Knoten aus, die vor dem aktuellen Knoten im Dokument erscheinen. Davon ausgenommen sind Vorgänger, Attributknoten und Namespace-Knoten. Der Vorhersage-Ausdruck "preceding::book" wählt beispielsweise die Knoten vor dem Buch-Knoten aus.

preceding-sibling

Wählt alle Gleichgeordneten aus, die vor dem aktuellen Knoten im Dokument erscheinen. Der Vorhersage-Ausdruck "preceding-sibling::book" wählt beispielsweise die gleichgeordneten Knoten vor dem Buch-Knoten aus.

self

Wählt den aktuellen Knoten aus. Der Vorhersage-Ausdruck "self::book" wählt beispielsweise den aktuellen Buch-Knoten aus.

Arithmetische XPath-Operatoren

Um Berechnungen durchzuführen, fügen Sie mathematische Operatoren hinzu, die hierarchische Dokumentknoten bewerten. Sie können arithmetische Operatoren zu XPath-Ausdrücken im Eingabe-, Bedingungs- oder Ausgabefeld einer Mapping-Anweisung hinzufügen.

In der folgenden Tabelle werden die arithmetischen XPath-Operatoren beschrieben, die Sie in XMap-Ausdrücken verwenden können:

XPath-Ausdruck	Beschreibung
	Wählt zwei Knotensätze im Kontext aus. Der Vorhersage-Ausdruck "//book //cd" gibt beispielsweise einen Knotensatz mit allen Buch- und CD-Elementen zurück.
+	Fügt die Elemente hinzu. Der Vorhersage-Ausdruck "1+2" gibt beispielsweise 3 zurück.
-	Subtrahiert die Elemente. Der Vorhersage-Ausdruck "2-1" gibt beispielsweise 1 zurück.
*	Multipliziert die Elemente. Der Vorhersage-Ausdruck "2*1" gibt beispielsweise 2 zurück.
div	Dividiert die Elemente. Der Vorhersage-Ausdruck "6 div 3" gibt beispielsweise 2 zurück.
=	Wählt die Elemente aus, die dem Ausdruck gleichen. Der Vorhersage-Ausdruck "cost=1.50" gibt beispielsweise "True" zurück, wenn die Kosten 1.50 betragen, und "False", wenn die Kosten 1.60 betragen.
!=	Wählt die Elemente aus, die dem Ausdruck nicht gleichen. Der Vorhersage-Ausdruck "cost!=1.50" gibt beispielsweise "True" zurück, wenn die Kosten 1.60 betragen, und "False", wenn die Kosten 1.50 betragen.
<	Wählt die Elemente, die kleiner als der Ausdruck sind. Der Vorhersage-Ausdruck "tax<1.50" gibt beispielsweise "True" zurück, wenn die Steuer 1.00 beträgt, und "False", wenn die Steuer 1.50 beträgt.
<=	Wählt die Elemente aus, die dem Ausdruck gleichen oder kleiner sind. Der Vorhersage-Ausdruck "tax<=1.50" gibt beispielsweise "True" zurück, wenn die Steuer 1.50 beträgt, und "False", wenn die Steuer 1.80 beträgt.
>	Wählt die Elemente aus, die größer als der Ausdruck sind. Der Vorhersage-Ausdruck "tax>1.50" gibt beispielsweise "True" zurück, wenn die Steuer 1.90 beträgt, und "False", wenn die Steuer 1.50 beträgt.
>=	Wählt die Elemente aus, die dem Ausdruck gleichen oder größer sind. Der Vorhersage-Ausdruck "tax>=1.50" gibt beispielsweise "True" zurück, wenn die Steuer 1.50 beträgt, und "False", wenn die Steuer 1.00 beträgt.
or	Wählt die Elemente aus, die eine oder mehrere Bedingungen erfüllen. Der Vorhersage-Ausdruck "tax=1.50 or tax=1.70" gibt beispielsweise "True" zurück, wenn die Steuer 1.50 beträgt, und "False", wenn die Steuer 1.00 beträgt.
and	Wählt die Elemente aus, die alle angegebenen Bedingungen erfüllen. Der Vorhersage-Ausdruck "price>1.00 and price<1.90" gibt beispielsweise "True" zurück, wenn der Preis 1.50 beträgt, und "False", wenn der Preis 1.00 beträgt.
mod	Führt eine Division durch und gibt den Rest zurück. Der Vorhersage-Ausdruck "3 mod 2" gibt beispielsweise 1 zurück.

Ausgabe von XPath-Ausdrücken

Die Datenprozessor-Umwandlung akzeptiert eine Teilmenge der XPath-Anweisungen im Ausgabefeld, wenn das Feld „Modus“ auf **Hinzufügen** oder **Match oder Hinzufügen** gesetzt ist. Wenn Sie diese Modus-Einstellungen auswählen, erstellt die Datenprozessor-Umwandlung Elemente nach Bedarf, um mit dem XPath-Ausdruck im Ausgabefeld übereinzustimmen.

Sie können einen einfachen XPath-Ausdruck im Ausgabefeld verwenden. Ein einfacher Ausdruck hat untergeordnete Achsen, übergeordnete Achsen oder Variablen. Einfache Ausdrücke haben keine Vorhersagen, Funktionen oder komplexen Achsen. Beispielsweise können Sie die folgenden Ausgabefeld-Ausdrücke verwenden:

```
person/data
/root/ceo/name
$var/name
person/../ceo
```

Sie können eine einfache Vorhersage mit Kardinalität für ein Element mit mehreren Instanzen verwenden. Beispielsweise können Sie den folgenden Ausgabefeld-Ausdruck verwenden:

```
person/phone[4]
```

Sie können ein einfaches Prädikat mit einer Formel mit einem Gleichheitszeichen mit einfachen XPath auf der linken Seite des Gleichheitszeichens verwenden. Beispielsweise können Sie die folgenden Ausgabefeld-Ausdrücke verwenden:

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

Außerdem können Sie eine Kombination aus einem einfachen Ausdruck mit Kardinalität und einer Formel verwenden, die einen einfachen XPath auf der linken Seite des Gleichheitszeichens verwendet. Beispielsweise können Sie die folgenden Ausgabefeld-Ausdrücke verwenden:

```
company[4]/details[id=$myid]/phone
```

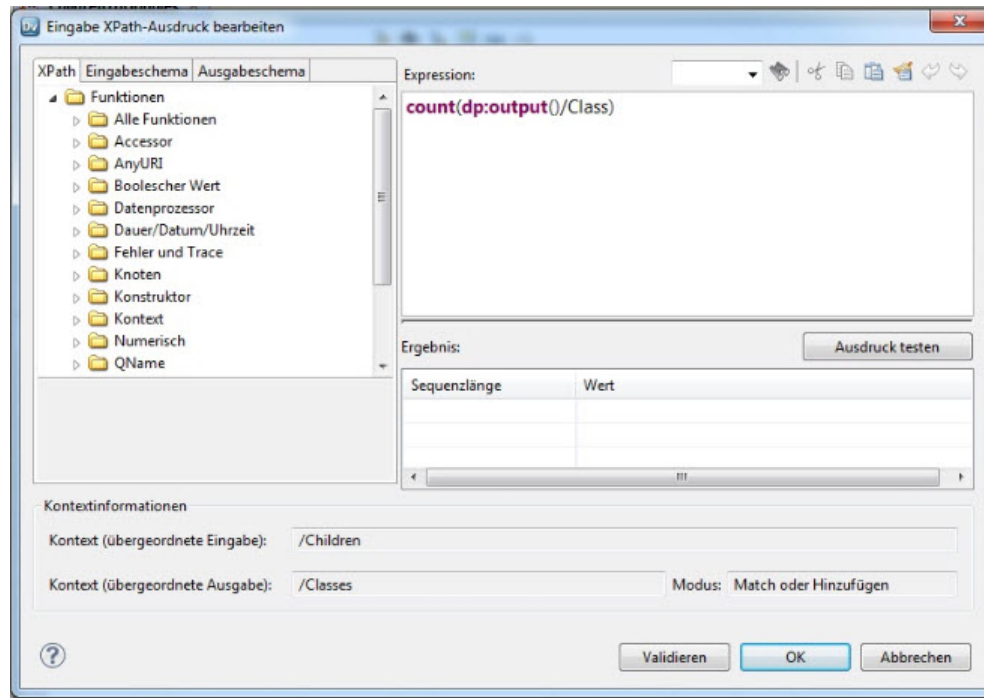
Hinweis: Wenn das Feld „Modus“ auf **Match** gesetzt wurde, kann das Ausgabefeld auch komplexe XPath-Ausdrücke akzeptieren.

XPath-Ausdruckseditor

Erstellen Sie Ausdrücke im XPath-Ausdruckseditor. XPath ist eine Abfragesprache, die verwendet wird, um Knoten in einem hierarchischen Dokument auszuwählen und Berechnungen durchzuführen.

Sie können XPath-Ausdrücke zum Definieren des Kontexts für eine Mapping-Anweisung verwenden. Sie können Bedingungen definieren, um die Daten mithilfe von verschiedenen XPath-Ausdrücken in den Eigenschaften einer Mapping-Anweisung umzuwandeln und zu filtern. Mit XPath-Ausdrücken können Sie verschiedene arithmetische Berechnungen zu einer Mapping-Anweisung hinzufügen.

Wenn Sie im Feld „Eingabe“, „Bedingung“ oder „Ausgabe“ auf die Schaltfläche „Öffnen“ klicken, wird der Ausdruckseditor angezeigt. Die folgende Abbildung zeigt den XPath-Ausdruckseditor:



Erstellen Sie Ausdrücke im Bereich **Ausdruck**.

Der XPath-Ausdruckseditor weist den Bereich **Navigation** mit einer Funktionsbibliothek auf, die Sie zum Erstellen von XPath-Ausdrücken verwenden können. Die Funktionen sind Standardfunktionen für die XML Path Language von W3C. Die Funktionsbibliothek enthält außerdem einige Funktionen, die speziell für die Datenprozessor-Umwandlung konzipiert sind.

Datenprozessor-Funktionen

Der Ausdruckseditor verfügt über Datenprozessor-Funktionen, die Sie für die Datenprozessor-Umwandlung verwenden können.

Die Datenprozessor-Umwandlung verwendet die folgenden XPath-Funktionen:

dp:as_xml

Erhält einen Knoten als eine Eingabe und gibt den Knotenwert sowie den Wert aller untergeordneten Elemente als XML-Zeichenfolge rekursiv zurück. Die as_xml-Funktion verwendet die folgende Syntax:

`dp:as_XML(<node>)`

dp:get_id

Generiert eine einem Knoten zugeordnete eindeutige ID und gibt sie aus. Sie können die ID verwenden, um Beziehungen zwischen Primärschlüssel und Fremdschlüssel in den Daten zu erstellen. Ordnen Sie die ID einem Knoten im Schema und Schlüsseln in relationalen Daten zu. Die get_id-Funktion verwendet die folgende Syntax: `dp:get_id(<node>)`

dp:input

Gibt den Knoten aus, der den aktuellen Eingabekontext bereitstellt. Verwenden Sie die Funktion im Ausgabefeld, um auf einen Knoten aus dem Eingabeschema zu verweisen. Die input-Funktion verwendet die folgende Syntax: `dp:input()`

dp:transform

Rufen Sie einen Datenprozessor-Umwandlungs-Transformer auf, den Sie im Skript definieren. Sie können eine interne oder externe Umwandlung durchführen. Die Funktion verwendet die folgende Syntax:

```
dp:transform(<transform-name>,<transform-value>)
```

Die transform-Funktion verwendet die folgenden Parameter:

- Transform-name. Der Name des Transformers im Skript.
- Transform-value. Der Umwandlungswert für die Durchführung der Umwandlung.

Hinweis: Die lookup-Funktion ist über die Verwendung der Funktion **dp:transform** verfügbar.

dp:output

Gibt den Knoten aus, der den aktuellen Ausgabekontext bereitstellt. Verwenden Sie die Funktion im Eingabefeld, um auf einen Knoten aus dem Ausgabeschema zu verweisen. Die output-Funktion verwendet die folgende Syntax: `dp:output()`

Beispiel für XPath-Ausdrücke

Verwenden Sie XPath-Ausdrücke in Mapping-Anweisungen. XPath-Ausdrücke geben Elemente in dem Eingabedokumenten an, das im Ausgabedokument zugeordnet und umgewandelt wird. Ein XPath-Ausdruck wird auch für die Durchführung einer arithmetischen Operation verwendet.

Die folgende Abbildung zeigt XPath-Ausdrücke im Gitter:

Das XMap-Eingabedokument entspricht einer Liste mit Kindern und ihren Hobbies. Der Eingabestamm ist „Children“. „Child ist ein mehrfach vorkommendes Element innerhalb von „Children“. Jedes Kind weist einen Namen (Name) und mehrfach vorkommende Hobbies auf. Der Name besteht aus den Elementen „First“, „Initial“ und „Last“.

Die Ausgabe ist eine Liste der Klassen mit der Anzahl der Kinder in jeder Klasse. Der Ausgabestamm lautet „Classes“. „Classes“ weist ein Attribut auf, das die Gesamtanzahl der Klassen enthält. Jedes Eingabeelement „Hobby“ wird einem Ausgabeelement „Class“ zugeordnet. Eine Map-Anweisung verkettet die Elemente „First“, „Initial“ und „Last“ in dem Ausgabeelement „Child“. Eine andere Map-Anweisung zählt die Anzahl der Kinder in jeder Klasse. Eine weitere Map-Anweisung zählt die Anzahl der Klassen.

Die XMap enthält die folgenden Ausdrücke:

Ausdruck in Gitterzeile 2 <Class[@name = dp:input()]>

Fügt ein Class-Element hinzu oder sucht nach einem Class-Element, das mit „Hobby“ übereinstimmt. „dp:input()“ ist erforderlich, da sich der Ausdruck auf ein Eingabeelement bezieht.

Ausdruck in Gitterzeile 3 <concat(..Name/First,'',../Name/Initial,'',../Name/Last)>

Verknüpft den Vornamen, mittleren Namen und Nachnamen und fügt Leerzeichen zwischen den Namen ein.

Ausdruck in Gitterzeile 4 <dp:output()/@noOfChildren + 1>

Fügt bei jedem auftretenden Hobby der Anzahl der Kinder für diese Klasse 1 hinzu. Die Funktion „dp:output()“ ist erforderlich, da sich der Ausdruck auf ein Ausgabeelement bezieht.

Ausdruck in Gitterzeile 5 <count(dp:output()/Class)>

Zählt die Class-Elemente. Die Funktion „dp:output()“ ist erforderlich, da sich der Ausdruck auf ein Ausgabeelement bezieht.

Erstellen eines Ausdrucks

Erstellen Sie XPath-Ausdrücke im **Ausdruckseditor**.

1. Klicken Sie in der XMap-Anweisung auf die Schaltfläche **Öffnen** im Feld **Eingabe, Bedingung** oder **Ausgabe**.
Der **Ausdruckseditor** wird angezeigt.
2. Um einem Ausdruck ein Element hinzuzufügen, doppelklicken Sie auf Elemente in der Maske **Navigation**.
3. Klicken Sie auf **Validieren**, um den Ausdruck zu validieren.
4. Wenn der Ausdruck für das Feld "Eingabe" vorgesehen ist, klicken Sie auf **Ausdruck testen**, um den Ausdruck anhand der Beispieldaten zu testen.
Die Ergebnisse erscheinen jedes Mal, wenn das Developer-Tool den Ausdruck anhand der Beispieldaten bewertet. Der XPath-Ausdruck kann eine Sequenz von null oder mehr Knoten oder Werten zurückgeben. Die **Sequenzlänge** gibt an, wie viele Knoten der XPath-Ausdruck zurückgibt.

XMap-Variablen

Sie können im XMap-Editor Variablen hinzufügen. Sie können Variablen Werte zuordnen und die Variablen in Prädikaten oder als temporäre Behälter für Werte verwenden. Sie können Ausgabeelementen Variablen zuordnen.

Wenn Sie eine Variable erstellen, wird diese in der XMap-Ansicht in den Eingabe- und Ausgabeschemata angezeigt. Das Developer Tool fügt dem Variablennamen ein Dollarzeichen (\$) hinzu, das anzeigt, dass es sich hierbei um eine Variable handelt.

Sie können eine Variable erstellen, die aus einer Liste mit mehreren Werten besteht. Sie können Listenvariablen für denselben Zweck wie ein mehrfach vorkommendes Schemaelement verwenden. Konfigurieren Sie eine Listenvariable als Eingabe für eine wiederholende Gruppe oder konfigurieren Sie ein Prädikat zur Suche nach einem Wert in der Listenvariable.

Beispielsweise haben Sie ein XML-Dokument, das Adressen enthält. Sie müssen aus den Adressen eine Liste mit allen Ländern erstellen. Ordnen Sie das Länderelement in einer Variablen „\$countries“ zu, die Sie als Liste definieren.

Erstellen einer Variablen im XMap-Editor

Sie können Variablen im XMap-Editor erstellen.

1. Klicken Sie auf **Variablen** über dem Eingabe- oder Ausgabeschema im XMap-Editor.
Das Dialogfeld **Variablen** wird eingeblendet.
2. Um eine Variable zu erstellen, klicken Sie auf **Neu**.
3. Geben Sie den Namen der Variablen und einen Datentyp ein.
4. Aktivieren Sie **Liste**, um eine mehrfach vorkommende Variable zu erstellen.

XMap-Beispiel

Ein XML-Dokument enthält Mitarbeiterdaten, einschließlich der Mitarbeiterrolle im Unternehmen. Sie wollen ein XML-Dokument erstellen, bei dem Manager und Mitarbeiter in getrennte Gruppen eingeteilt werden. Sie erstellen zwei XMap-Objekte in der Datenprozessor-Umwandlung, um das XML-Dokument zu restrukturieren.

Die Startkomponente ist ein XMap-Objekt, das eine Router-Anweisung enthält. Eine Option-Anweisung prüft, ob die Mitarbeiterrolle "Manager" ist. Wenn die Rolle "Manager" ist, weist das XMap-Objekt die Mitarbeiterelemente einer Manager-Ausgabegruppe zu. Sonst weist das XMap-Objekt die Mitarbeiterelemente einer "Worker"-Gruppe im Ausgabe-XML-Code zu.

Die XMap-Startkomponente ruft ein weiteres XMap-Objekt auf, um die Mitarbeiterelemente den Ausgabeelementen zuzuweisen.

XML-Eingabeschema - Beispiel

Das XML-Eingabeschema für das XMap-Beispiel hat die folgende Struktur:

```
<?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:sequence>
    </xs:complexType>
  </xs:element>

</xs:schema>
```



```

        <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>

```

Die Schema-Root ist "Input". Im "Input" tritt "Company" mehrfach auf. In jeder "Company" ist mehr als ein "Department". Jedes "Department" hat "Employees". "Employee" hat eine "Role", die festlegt, ob der Mitarbeiter Manager ist oder nicht.

XML-Ausgabeschema - Beispiel

Das XML-Ausgabeschema für das XMap-Beispiel hat folgende Struktur:

```

<?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: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>

```

Die Schema-Root ist "Output". Im "Output" tritt "Organization" mehrfach auf. Jede "Organization" hat "Worker" und "Manager". "Worker" und "Manager" sind vom Typ "WorkerTypes" "Worker" und "Manager" enthalten dieselben Elemente. Der "WorkerType" enthält ein YearsOfService-Element, das die XMap basierend auf dem StartDat berechnet.

XML-Eingabedaten

Im folgenden Text werden Beispieldaten vom XML-Eingabedokument aufgeführt:

```
<?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>

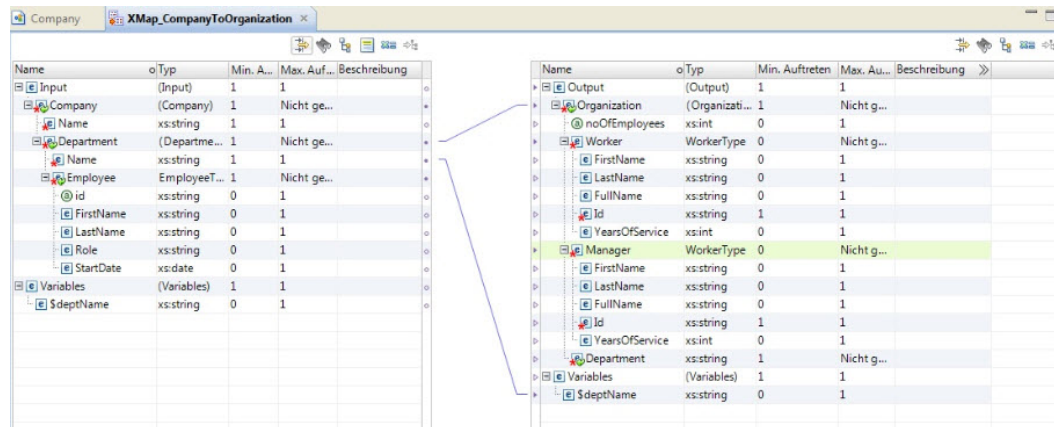
      <Employee id="761338290">
        <FirstName>Felix</FirstName>
        <LastName>Boyd</LastName>
        <Role>Worker</Role>
        <StartDate>2009-12-29</StartDate>
      </Employee>
    </Department>
  </Company>
```

Die Daten können mehrere Unternehmen enthalten. Jedes Unternehmen enthält mehrere Abteilungen. Jeder Mitarbeiter in einer Abteilung weist eine Rolle auf, die entweder einem Manager oder einem anderen Typ von Arbeitern entspricht.

Eingabe- und Ausgabe-XML-Hierarchien

Der XMap-Editor zeigt die Eingabehierarchie im linken Bereich der Ansicht und die Ausgabe-XML-Hierarchie im rechten Bereich der Ansicht.

Die folgende Abbildung zeigt die Eingabe- und Ausgabe-XML-Hierarchien:



Mapping-Anweisungen im Beispiel

Verwenden Sie den Gitterbereich des XMap-Editors, um Anweisungen für das Mapping der Eingabe-XML-Elemente zu den Ausgabe-XML-Dokumenten zu definieren. Erstellen und bearbeiten Sie die Mapping-Anweisungen im Gitter. Definieren Sie den Kontext, die Bedingung und die erwartete Eingabe und Ausgabe für eine Mapping-Anweisung. Sie können Variablen zu Mapping-Anweisungen hinzufügen.

In der folgenden Abbildung werden die Mapping-Anweisungen im Gitter dargestellt:

Name	Statement...	Eingabe	Bedingung	Fehlen...	Stand...	Bei Versagen	Ausgabe	Modus
1 Company	Wiederhol...	tns0:Company		<input type="checkbox"/>		Verteilen		Hinzufügen
2 Department	Wiederhol...	tns0:Department		<input type="checkbox"/>		Verteilen		Hinzufügen
3 NameToDepartment	Zuordnen	tns0:Name		<input type="checkbox"/>		Verteilen	\$deptName	Match oder Hinzufügen
4 MatchOrganization	Gruppe	.		<input type="checkbox"/>		Verteilen	tns0:Organization[tns0:Departments=\$d...	Match oder Hinzufügen
5 Employee to Worker	Wiederhol...	tns0:Employee		<input type="checkbox"/>		Iteration überspri...		Match oder Hinzufügen
6 Employee	Router			<input type="checkbox"/>		Überspringen		Match oder Hinzufügen
7 Employee to Manager	Option		tns0:Roles="Ma...	<input checked="" type="checkbox"/>		Verteilen		Match oder Hinzufügen
8 EmployeeToWorker	EmployeeT...			<input type="checkbox"/>		Verteilen	tns0:Manager	Hinzufügen
9 EmployeeToWorker	Standard			<input type="checkbox"/>		Verteilen		Match oder Hinzufügen
10 EmployeeToWorker	EmployeeT...			<input type="checkbox"/>		Verteilen	tns0:Worker	Hinzufügen
11 Increment employee cou...	Zuordnen	dp:output()/@noOfE...		<input type="checkbox"/>	1	Verteilen	@noOfEmployees	Match oder Hinzufügen

Das Gitter enthält die folgenden Mapping-Anweisungen:

Gitterzeile 1, Repeating Group-Anweisung namens Company

Die Company-Anweisung ist eine Repeating Group-Anweisung. Sie wiederholt sich für jedes Company-Element. Die Anweisung bietet den Kontext für den Rest der Anweisungen im Gitter. Die Datenprozessor-Umwandlung evaluiert für jede Firma die untergeordneten Anweisungen.

Gitterzeile 2, Repeating Group-Anweisung namens Department

Die Department-Anweisung ist eine Repeating Group-Anweisung. Sie wiederholt sich für jedes Department-Element. Die Anweisung bietet den Kontext für den Rest der Anweisungen im Gitter. Die Datenprozessor-Umwandlung evaluiert für jede Abteilung die untergeordneten Anweisungen.

Gitterzeile 3, Map-Anweisung namens NametoDepartment

Die NametoDepartment-Anweisung ist eine Map-Anweisung. Sie ordnet der Variable \$deptName das Name-Element zu.

Gitterzeile 4, Repeating Group-Anweisung namens MatchOrganization

Die MatchOrganization-Anweisung ist eine Repeating Group-Anweisung. Sie verfügt über einen Ausgabeausdruck:

```
tns0:Organization[tns0:Department=$deptName]
```

Die Anweisung findet das Organization-Element in der Ausgabe, die ein untergeordnetes Department-Element mit dem Wert in \$deptName enthält. Wenn das Department-Element nicht vorhanden ist, wird das Element erstellt.

Gitterzeile 5, Repeating Group-Anweisung namens EmployeeToWorker

Die EmployeeToWorker-Anweisung ist eine Repeating Group-Anweisung. Sie wiederholt sich für jedes Employee-Element.

Gitterzeile 6, Router-Anweisung namens Employee

Die Employee-Anweisung ist eine Router-Anweisung. Die Anweisung hat weder Eingabe noch Ausgabe.

Gitterzeile 7, Option-Anweisung namens EmployeeToMgr

Die EmployeeToMgr-Anweisung ist eine Option-Anweisung. Die Option-Anweisung enthält die folgende Bedingung:

```
tns0:Role="Manager".
```

Wenn es sich bei der Rolle um "Manager" handelt, ist die Anweisung "True" und die Datenprozessor-Umwandlung evaluiert die innerhalb der Option-Anweisung geschachtelten Anweisungen.

Gitterzeile 8, Run XMap-Anweisung namens EmployeeToWorker

Die EmployeeToWorker-Anweisung ist eine Run XMap-Anweisung. Sie ruft die XMap_EmployeesToRoles-XMap auf, die Employee-Elemente an den Manager-Typ weiterzugeben.

Gitterzeile 9, Default-Anweisung namens EmployeeToWorker

Die EmployeeToWorker-Anweisung ist eine Default-Anweisung. Die Datenprozessor-Umwandlung führt die untergeordneten Anweisungen aus, wenn es sich bei der Mitarbeiterrolle nicht um "Manager" handelt.

Gitterzeile 10, Run XMap-Anweisung namens EmployeeToWorker

Die EmployeeToWorker-Anweisung ist eine Run XMap-Anweisung. Ruft die XMap_EmployeesToRoles-XMap auf, die Employee-Elemente an den Worker-Typ weiterzugeben.

Gitterzeile 11, Map-Anweisung namens IncrementEmployeeCount

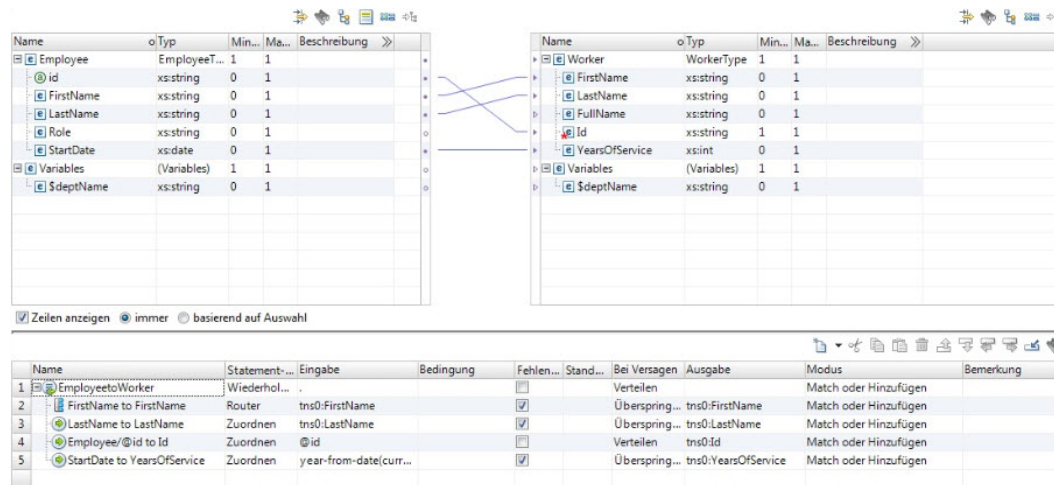
Die IncrementEmployeeCount-Anweisung ist eine Map-Anweisung. Sie ruft die Datenprozessor-Umwandlung auf, für jedes durchlaufene Employee-Element den Wert 1 zu @noOfEmployees hinzuzufügen. Die Map-Anweisung enthält den folgenden Eingabeausdruck:

```
dp:output()/@ noOfEmployees + 1.
```

Gruppenanweisungsbeispiel

Das XMap-Objekt "EmployeeToWorker" verschiebt Elemente von einem Angestellten zu einem Arbeiter. Das XMap-Objekt verarbeitet einen Angestellten.

Die folgende Abbildung zeigt das EmployeeToWorker-XMap -Objekt im XMap-Editor:



Das Gitter enthält die folgenden Mapping-Anweisungen:

Gitter Zeile 1, Gruppenanweisung namens EmployeeToWorker

Die EmployeeToWorker-Anweisung ist eine Group-Anweisung. Sie enthält Kontext für den Rest der Mapping-Anweisungen.

Gitter Zeile 2, Map-Anweisung namens FirstNametoFirstName

Die FirstNametoFirstName-Anweisung ist eine Map-Anweisung. Ordnet den Vornamen dem Vornamen zu.

Gitter Zeile 3, Map-Anweisung namens LastNametoLastName

Die LastNametoLastName-Anweisung ist eine Map-Anweisung. Ordnet den Nachnamen dem Nachnamen zu.

Gitter Zeile 4, Map-Anweisung namens Employee/@IDtoID

Die Employee/@IDtoID-Anweisung ist eine Map-Anweisung. Ordnet die Mitarbeiter-ID der Mitarbeiter-ID zu.

Gitter Zeile 5, Map-Anweisung namens StartDatetoYearsofService

Die StartDatetoYearsofService-Anweisung ist eine Map-Anweisung. Ermittelt die Anzahl der Dienstjahre durch Subtraktion eines Startdatums vom aktuellen Datum.

KAPITEL 7

Bibliotheken

Dieses Kapitel umfasst die folgenden Themen:

- [Bibliotheken - Übersicht, 118](#)
- [Struktur von Bibliotheken, 119](#)
- [Elementeigenschaften, 119](#)
- [Bibliotheksverwaltung, 119](#)
- [Bearbeiten von Bibliotheken mit dem Bibliothek-Editor, 120](#)
- [Bearbeiten von Bibliotheken mit dem IntelliScript-Editor, 122](#)

Bibliotheken - Übersicht

Eine Bibliothek ist ein Datenprozessor-Umwandlungsobjekt, das vordefinierte Komponenten enthält, die zur Umwandlung einer Reihe von Branchenstandards für Meldungen verwendet werden. Eine Datenprozessor-Umwandlung verwendet eine Bibliothek, um die Eingabe von branchenüblichen Meldungen in andere Formate umzuwandeln. Sie können Bibliotheksobjekte für alle Bibliotheken erstellen.

Eine Bibliothek enthält eine große Anzahl an Objekten und Komponenten, wie z. B. Parser, Serializer und XML-Schemas, die die branchenübliche Standardeingabe und spezifische Anwendungsmeldungen in XML-Ausgabe umwandeln. Eine Bibliothek kann Objekte zur Validierung und Bestätigung von Meldungen sowie Diagnoseanzeigen enthalten. Eine Bibliothek verwendet Objekte, um den branchenüblichen Eingabetyp der Meldungen in XML und von XML in andere Formate umzuwandeln.

Sie können Bibliotheksobjekte für die folgenden Bibliotheken erstellen: ACORD, BAI, CREST, DTCC-NSCC, EDIFACT, EDIT-UCS & WINS, EDI_VICS, EDI-X12, FIX, FpML, HIPAA, HIX, HL7, IATA, IDS, MDM-Zuordnung, NACHA, NCPDP, SEPA, SWIFT und Telekurs.

Sie können einen speziellen Bibliothek-Editor zum Bearbeiten der Bibliotheksspezifikationen für die Bibliotheken DTCC-NSCC, EDIFACT, EDI-X12, HIPAA, HL7 und SWIFT verwenden. Eine Bibliotheksmeldung enthält ein Root-Element sowie Container- und Datenelemente. Die Typen der Container- und Datenelemente unterscheiden sich je nach Meldungstyp. Sie können Elemente hinzufügen und löschen und die Eigenschaften der Elemente konfigurieren, um die Validierungseinstellungen zu ändern.

Weitere Informationen zu branchenüblichen Meldungstypen finden Sie im *Handbuch für Datenumwandlungsbibliotheken*.

Struktur von Bibliotheken

Eine Bibliothek ist ein Satz von Umwandlungen. Alle Bibliotheken enthalten Parser, Serializer und XML-Schemas. Bestimmte Bibliotheken enthalten zusätzliche Komponenten für die Meldungsvalidierung, Bestätigungen und Diagnoseanzeigen. Ein Bibliotheksobjekt ist ein Satz von Komponenten, die einen bestimmten branchenüblichen Meldungstyp umwandeln.

Beim Erstellen einer Datenprozessor-Umwandlung müssen Sie keine eigenen Skripts erstellen, sondern können ein Bibliotheksobjekt einbeziehen, um einen branchenüblichen Meldungstyp umzuwandeln. Sie können ein Bibliotheksobjekt unverändert verwenden oder es basierend auf Ihren Anforderungen bearbeiten.

Jedes Bibliotheksobjekt wandelt einen bestimmten Branchenstandard um. Die Bibliothek HL7 etwa enthält Komponenten für die einzelnen Meldungstypen und Datenstrukturen des Kommunikationsstandards HL7 für medizinische Informationssysteme.

Die Bibliothek enthält bestimmte Arten von Meldungstypen. Die Bibliothek HL7 enthält zum Beispiel die folgenden Meldungen:

```
ADT_A01_Admit_a_Patient
ADT_A02_Transfer_a_Patient
```

Elementeigenschaften

Eine Bibliotheksmeldung enthält ein Root-Element sowie Container- und Datenelemente. Die Typen der Container- und Datenelemente unterscheiden sich je nach Meldungstyp. Sie können sowohl Container- als auch Datenelementeigenschaften mit dem Bibliothek-Editor konfigurieren.

Eigenschaften weisen die folgenden Kategorien auf:

Globale Einstellungen

Eigenschaften, die denselben Wert für ein Element mit mehr als einer Instanz aufweisen, die jeweils eine andere Position in der Hierarchie besetzt. Alle Änderungen an der Einstellung "Globale Einstellungen" werden in allen Instanzen des Elements angezeigt.

Positionseinstellungen

Eigenschaften, deren Wert für jede Instanz des Elements in der Hierarchie abweichen kann.

Elementeigenschaften werden gemäß Branchenstandards benannt. Der Bibliothek-Editor zeigt je nach Bibliothek verschiedene Eigenschaften an.

Bibliotheksverwaltung

Sie können den Bibliothek-Editor verwenden, um die Strukturen und Elemente des Meldungstyps für die Bibliotheken DTCC-NSCC, EDIFACT, EDI-X12, HIPAA, HL7 und SWIFT anzupassen. Wenn Sie die Struktur des Meldungstyps ändern möchten, verwenden Sie den Bibliothek-Editor zum Hinzufügen, Bearbeiten und Löschen von Elementen aus der Bibliotheksmeldung.

Sie können die Art und Weise, wie eine Meldung umgewandelt wird, ebenfalls ändern. Eine Bibliotheksumwandlung enthält eine große Anzahl von Objekten und Komponenten, wie Skripts mit Parsern, Serializern und XML-Schemas, die die Umwandlung definieren. Eine Bibliotheksumwandlung kann Objekte zur Validierung und Bestätigung von Meldungen sowie Diagnoseanzeigen enthalten.

Eine Bibliotheksumwandlung verwendet ihre Objekte, um den Meldungstyp aus der branchenüblichen Standardeingabe in XML und aus XML in andere Formate umzuwandeln. Um auf die mit dem Bibliothek-Editor erstellten Skripts, XMaps und Schemas zuzugreifen und sie zu bearbeiten, müssen Sie Bibliotheksobjekte erstellen. Erzeugen Sie die Bibliotheksobjekte nur, wenn Sie eine Änderung an den Bibliotheksobjekten vornehmen müssen, die nicht mit dem Bibliothek-Editor durchgeführt werden kann.

Verwenden Sie nach dem Erstellen der Bibliotheksobjekte den IntelliScript-Editor, um Parser, Serializer und Mapper zu bearbeiten. Beispiel: Sie möchten die Ausgabestruktur ändern, um sie an Ihre Anforderungen anzupassen. Sie erzeugen die Bibliotheksobjekte und bearbeiten die Skripts mit dem IntelliScript-Editor.

Wenn Sie eine Bibliothek für Bibliothekspakete erstellen, die den Bibliothek-Editor nicht unterstützen, werden die Bibliotheksobjekte im Voraus erzeugt. Es ist nicht erforderlich, die Bibliotheksobjekte zu erstellen. Sie können die Skriptkomponenten für diese Bibliotheken direkt mit dem IntelliScript-Editor bearbeiten.

Nachdem Sie Bibliotheksobjekte generiert haben oder wenn die Bibliotheksobjekte vorgeneriert wurden, können Sie die Bibliothekselemente nicht mit dem Bibliothek-Editor bearbeiten. Um den Bibliothek-Editor erneut zu verwenden, müssen Sie die generierten Bibliotheksobjekte für die Bibliotheken, die Sie mit dem Bibliothek-Editor bearbeiten können, verwerfen. Sie möchten beispielsweise Eingabefelder hinzufügen, die Struktur der Ausgabe jedoch nicht verändern. Verwerfen Sie die Bibliotheksobjekte und fügen Sie anschließend benutzerdefinierte Elemente mit dem Bibliothek-Editor hinzu.

Hinweis: Alle an den generierten Bibliotheksobjekten vorgenommenen Änderungen gehen verloren, wenn Sie die generierten Objekte verwerfen.

Bibliothek – Beispiel für EDI-X12

Sie und Ihre Lieferanten verwenden den X12 850-Standard für Bestellungen, um Aufträge zu dokumentieren und zu verarbeiten. Sie möchten die Spezifikationen aus den Branchenstandards für den Bibliotheksmeldungstyp ändern, um Ihre internen Prozesse zu unterstützen.

Unternehmen aus der Möbelbranche benötigen Standardelemente wie Modell, Farbe und Größe sowie ein benutzerdefiniertes Element für die Materialart. Eine Batch-Nummer oder ein Ablaufdatum ist nicht erforderlich. Verwenden Sie den Bibliothek-Editor, um Elemente zu entfernen und benutzerdefinierte Elemente hinzuzufügen.

Außerdem können Sie mit dem Bibliothek-Editor Elementeigenschaften ändern, Segmente hinzufügen und Elemente kopieren. Beispiel: Sie fügen neue Farben hinzu, indem Sie die Elementeigenschaft ändern, die die Liste der Farben definiert. Um dem Materialsegment und dem Hardwaresegment ein Herstellercodesegment hinzuzufügen, können Sie es kopieren.

Sie führen eine Datenprozessor-Umwandlung durch, die Ihre angepasste Bibliothek enthält. Die Texteingabe von Ihren Lieferanten wird in einer modifizierten Version des X12 850-Standards für Bestellungen formatiert. Das geänderte Eingabeformat stimmt mit den an der Bibliothek vorgenommenen Änderungen überein. Der Text wird in eine hierarchische Ausgabe umgewandelt, die an andere Umwandlungen zur weiteren Verarbeitung gesendet werden kann.

Bearbeiten von Bibliotheken mit dem Bibliothek-Editor

Sie können einen speziellen Bibliothek-Editor zum Bearbeiten der Bibliotheksspezifikationen für die Bibliotheken EDI-X12, SWIFT, HL7, HIPAA, DTCC-NSCC und EDIFACT verwenden.

Über den Bibliothek-Editor können Sie die Meldungstypstrukturen und -elemente mit einem Editor anpassen, der auf die einzelnen Meldungstypen abgestimmt ist. Mit dem Bibliothek-Editor können Sie Elemente aus dem Bibliotheksobjekt hinzufügen, bearbeiten oder löschen.

Hinzufügen eines Elements mit dem Bibliothek-Editor

Eine Bibliotheksmeldung enthält ein Root-Element sowie Container- und Datenelemente. Verwenden Sie den Bibliothek-Editor, um ein Element zu einer Meldung hinzuzufügen.

1. Wählen Sie in der Ansicht **Objekte** des Bibliothek-Editor-Objekt aus und klicken Sie auf **Öffnen**.
Der Bibliothek-Editor wird angezeigt.
2. Klicken Sie auf das Symbol **Element hinzufügen** und wählen Sie aus, wo Sie das Element hinzufügen möchten.
 - **Neu**. Hängen Sie das Element an das Ende der Elementliste an.
 - **Einfügen über**. Fügen Sie das Element über dem im Bibliothek-Editor ausgewählten Element ein.
 - **Einfügen unter**. Fügen Sie das Element unter dem im Bibliothek-Editor ausgewählten Element ein.
 - **Einfügen zwischen**. Fügen Sie das Element innerhalb des im Bibliothek-Editor ausgewählten Container-Elements ein.
3. Wählen Sie aus, ob Sie ein vorhandenes Element kopieren oder ein neues Element erstellen möchten und klicken Sie auf **OK**.

Bearbeiten von Elementeigenschaften mit dem Bibliothek-Editor

Verwenden Sie den Bibliothek-Editor zum Bearbeiten eines Container- oder Datenelements in einer Bibliothek. Wenn Sie die Eigenschaften eines Elements ändern, ändern Sie die Spezifikationen für den Meldungstyp.

1. Wählen Sie im Fenster **Meldungsdefinition** des Bibliothek-Editors das zu bearbeitende Dokument aus.
Im Fenster **Eigenschaften** werden die Elementeigenschaften angezeigt.
2. Wählen Sie im Fenster **Eigenschaften** eine Eigenschaft aus und geben Sie den neuen Eigenschaftswert ein.

Wenn Sie einen falschen Wert oder einen außerhalb des zulässigen Bereichs liegenden Wert eingeben, werden Sie zum Korrigieren des Werts aufgefordert.

Testen einer Bibliothek

Testen Sie das Bibliotheksobjekt in der Ansicht **Daten-Viewer**.

Wählen Sie vor dem Testen der Bibliothek eine Beispieleingabedatei für den Test aus.

1. Um eine Beispieleingabedatei auszuwählen, durchsuchen Sie den Bereich **Allgemein** des Bibliothek-Editors in der Nähe des Felds **Mustereingabe** und wählen Sie eine Beispieleingabedatei aus.
2. Öffnen Sie die Ansicht **Daten-Viewer**.
3. Zum Auswählen des Bibliotheksmeldungstyps, den Sie als Komponente für die Ausführung bearbeitet haben, klicken Sie auf **Mit Editor synchronisieren**.
4. Klicken Sie auf **Ausführen**.
Das Developer Tool führt den Bibliotheksobjekt-Parser aus. Die Ausgabeergebnisse werden im Fenster **Ausgabe** angezeigt.
5. Sind Validierungsfehler vorhanden, werden im Bereich **Ausgabefehler** des Fensters **Ausgabe** die Fehler und der jeweilige Speicherort angezeigt. Doppelklicken Sie auf den Fehler, um nach der Fehlerquelle zu suchen.

6. Klicken Sie zum Anzeigen der Validierungsfehlerdateien im Bereich **Zusätzliche Ausgaben** des Fensters **Ausgabe** auf die Dateinamen.

Wenn Sie auf die Datei `ErrorsFound.txt` oder die Datei `Errors.xml` klicken, wird die entsprechende Datei in einem externen Browser geöffnet.

Generieren der Bibliotheksobjekte

Generieren Sie die Bibliotheksobjekte, damit Sie mit den Datenprozessor-Umwandlungseditoren direkt auf diese zugreifen können. Nach dem Generieren der Bibliotheksobjekte können Sie den Bibliothek-Editor nicht mehr zum Bearbeiten der Bibliothek verwenden.

Sie müssen Bibliotheksobjekte generieren, um auf die mit der Bibliothek verknüpften vorkonfigurierten Parser, Mapper, Serializer und XML-Schemata zugreifen und diese bearbeiten zu können.

1. Klicken Sie in der Ansicht **Objekte** mit der rechten Maustaste auf die Bibliothek und wählen Sie **Bibliotheksobjekte generieren** aus.

2. Wählen Sie **Ja** aus, um auf die Bibliotheksobjekte zuzugreifen.

Das Developer Tool erstellt einen Ordner namens **Generierte Bibliotheksobjekte**, in dem die Bibliotheksobjekte enthalten sind. Die Startkomponente wird generiert.

Hinweis: Wenn Sie ändern möchten, welche Bibliotheksobjekte generiert werden, wählen Sie eine andere Komponente aus.

3. Um ein Skript oder Schema zu bearbeiten, wählen Sie es aus und klicken Sie auf **Öffnen**.

Verwenden Sie den IntelliScript-Editor, um das Skript zu bearbeiten.

Verwerfen der Bibliotheksobjekte

Um Bibliothekselemente mit dem Bibliothek-Editor zu bearbeiten, müssen Sie die von Ihnen erstellten Bibliotheksobjekte entfernen. Wenn Sie Bibliotheksobjekte verwerfen, gehen alle daran vorgenommenen Änderungen verloren.

1. Klicken Sie in der Ansicht **Objekte** mit der rechten Maustaste auf die Bibliothek und wählen Sie **Generierte Bibliotheksobjekte verwerfen** aus.

2. Um Bibliotheksobjekte in den Datenprozessor-Umwandlungseditoren zu verwerfen, wählen Sie **Ja** aus.

Der Ordner **Generierte Bibliotheksobjekte**, der verworfen wurde. Die Bibliothekskomponenten und -objekte sind immer noch vorhanden, Datenprozessor-Umwandlungseditoren können jedoch nicht darauf zugreifen.

Bearbeiten von Bibliotheken mit dem IntelliScript-Editor

Wenn Sie eine Bibliothek für die Bibliotheken ACORD, BAI, FIX, HIX, IATA, IDS, NACHA, NCPDP, Telekurs, Bloomberg, SEPA, FpML und Thompson Reuters erstellen, können Sie den IntelliScript-Editor verwenden, um die Skriptkomponenten für diese Bibliotheken direkt zu bearbeiten. Es ist nicht erforderlich, die Bibliotheksobjekte zu erstellen.

Für die Bibliotheken EDI-X12, SWIFT, HL7, HIPAA, DTCC-NSCC, Edifact, SEPA und FpML müssen Sie die Bibliotheksobjekte erstellen, um auf die Skripts, XMaps und Schemas zur Bearbeitung zugreifen zu können, die Sie mit dem Bibliothek-Editor bzw. dem IntelliScript-Editor erstellt haben.

Der IntelliScript-Editor ist ein grafisches Tool, das zur Bearbeitung von Skripts dient. Verwenden Sie den IntelliScript-Editor, um Skriptkomponenten zum Skript hinzuzufügen und die Eigenschaften von Skriptkomponenten zu konfigurieren. Weitere Informationen über Skripts und den IntelliScript-Editor finden Sie unter ["Übersicht über Skripts" auf Seite 140](#).

Wenn Sie ein Bibliotheksprojekt aus einer früheren Version als Data Transformation-Dienst importiert haben, können Sie die Skriptkomponenten mit dem IntelliScript-Editor bearbeiten.

KAPITEL 8

Schema-Objekt

Dieses Kapitel umfasst die folgenden Themen:

- [Schemaobjekt – Übersicht, 124](#)
- [Schemadateien, 124](#)
- [Ansicht Schemaobjekt - Übersicht, 125](#)
- [Schemaobjekt-Ansicht „Schema“, 126](#)
- [Schemaobjekt-Ansicht „Erweitert“, 131](#)
- [Erstellen eines Schemaobjekts, 132](#)
- [Schema-Updates, 133](#)

Schemaobjekt – Übersicht

Ein Schemaobjekt ist ein hierarchisches Schema, das in das Modellrepository importiert wird. Nach dem Import des Schemas können Sie die Schemakomponenten im Developer Tool anzeigen. Sie können ein Avro-, Parquet-, XML- oder JSON-Schema importieren. Das Developer Tool wandelt das Schema in eine XSD-Datei im Modellrepository um.

Beim Erstellen eines SOAP-Webdiensts können Sie die Struktur des Webdiensts auf Grundlage eines hierarchischen Schemas definieren. Wenn Sie einen Web-Dienst ohne WSDL erstellen, können Sie Operationen, Eingabe, Ausgabe und Fehlersignaturen basierend auf den Typen und Elementen festlegen, die im Schema definiert sind.

Beim Importieren eines Schemas können die allgemeinen Schemaeigenschaften in der Ansicht **Übersicht** bearbeitet werden. Die erweiterten Eigenschaften bearbeiten Sie in der Ansicht **Erweitert**. Der Inhalt der Schemadatei kann in der Ansicht **Schema** eingesehen werden.

Schemadateien

Sie können einem Schemaobjekt mehrere XSD-Dateien der Stammebene hinzufügen. Sie können auch XSD-Dateien der Stammebene aus einem Schemaobjekt entfernen.

Wenn Sie eine Schemadatei hinzufügen, importiert das Developer Tool sämtliche XSD-Dateien, die durch die von Ihnen hinzugefügte Datei importiert werden oder in dieser enthalten sind. Das Developer Tool validiert die Dateien, die Sie hinzufügen, anhand der Dateien, die zum Schemaobjekt gehören. Das Developer Tool lässt

das Hinzufügen einer Datei nicht zu, die einen Konflikt mit einer Datei verursacht, die zum Schemaobjekt gehört.

Beispiel: Ein Schemaobjekt enthält die Stammschemadatei „BostonCust.xsd“. Sie möchten dem Schemaobjekt die Stammschemadatei „LACust.xsd“ hinzufügen. Beide Schemadateien weisen den gleichen Target-Namespace auf und definieren das Element „Customer“. Wenn Sie versuchen, dem Schemaobjekt die Schemadatei „LACust.xsd“ hinzuzufügen, werden Sie vom Developer Tool aufgefordert auszuwählen, ob die Datei „BostonCust.xsd“ beibehalten oder mit der Datei „LACust.xsd“ überschrieben werden soll.

Sie können das Attribut `xsd:nillable` verwenden, um XSD-Elemente als nullwertfähig zu markieren. Wenn Sie ein Element als nullwertfähig markieren, lässt das entsprechende Element in der XML-Datei Nullwerte zu.

Sie können jede Schemadatei der Stammebene entfernen. Wenn Sie eine Schemadatei entfernen, ändert das Developer Tool den Elementtyp der von der Schemadatei definierten Elemente in `xs:string`.

Um eine Schemadatei hinzuzufügen, wählen Sie die Ansicht **Übersicht** aus und klicken Sie auf die Schaltfläche **Hinzufügen** neben der Liste **Schemaspeicherorte**. Wählen Sie dann die Schemadatei aus. Um eine Schemadatei zu entfernen, wählen Sie die Datei aus und klicken Sie auf die Schaltfläche **Entfernen**.

Ansicht Schemaobjekt - Übersicht

Wählen Sie die Ansicht **Übersicht** aus, um den Schemanamen oder die Schemabeschreibung zu aktualisieren, Namespaces anzuzeigen und Schemadateien zu verwalten.

In der Ansicht **Übersicht** werden der Name, die Beschreibung und der Target-Namespace des Schemas angezeigt. Sie können den Namen und die Beschreibung des Schemas bearbeiten. Der Target-Namespace ist der Namespace, zu dem die Schemakomponenten gehören. Wenn kein Target-Namespace angezeigt wird, gehören die Schemakomponenten zu keinem Namespace.

Die folgende Abbildung zeigt die Ansicht **Übersicht** von einem Schemaobjekt:

The screenshot shows a software window titled "so_Partners" with a tab labeled "Übersicht". The window is divided into several sections:

- Allgemein**: Contains three input fields:
 - Name:** "so_Partners"
 - Beschreibung:** An empty text area.
 - Ziel-Namespace:** "(Kein Ziel-Namespace angegeben)"
- Speicherorte des Schemas**: A table with two columns: "Speicherort des Schemas" and "Namespace". It contains one row with the values "file://infa2008deu2/infa_sha..." and "(Kein Ziel-Namespace angegeben)". To the right of the table are three buttons: "Hinzufügen...", "Entfernen", and "Öffnen mit...".
- Navigation**: At the bottom, there are three tabs: "Übersicht" (which is active), "Schema", and "Erweitert".

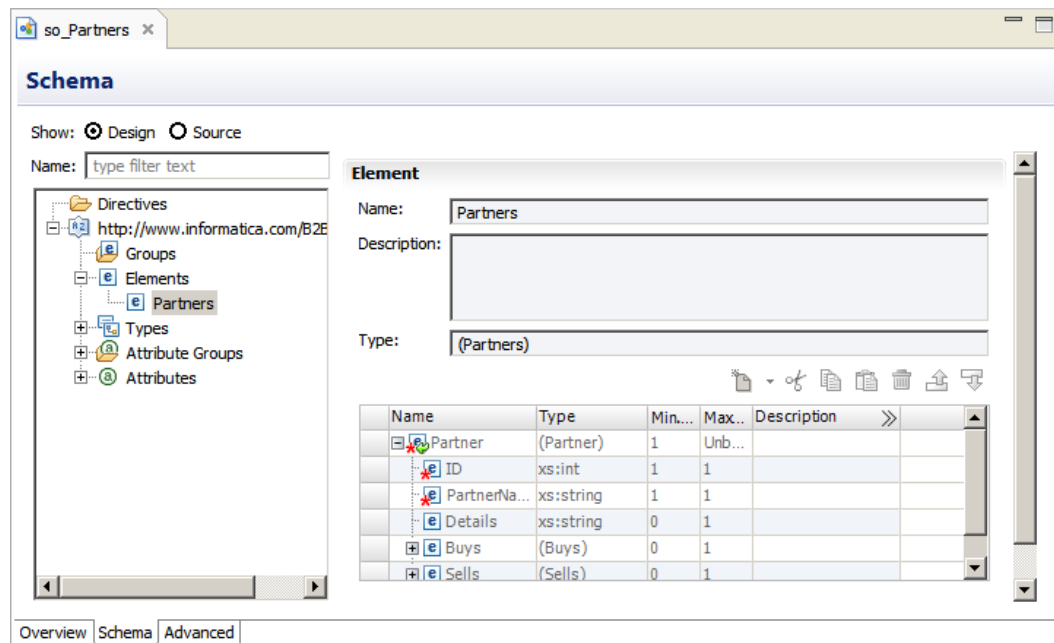
Im Bereich **Schemaspeicherorte** werden die Schemadateien und -namespaces aufgeführt. Sie können mehrere XSD-Stammdateien hinzufügen. Wenn eine Schemadatei andere Schemadateien enthält oder importiert, enthält das Developer Tool die XSD-Kinddateien im Schema.

Schemaobjekt-Ansicht „Schema“

Die Ansicht **Schema** zeigt eine alphanumerische Liste der Gruppen, Elemente, Typen, Attributgruppen und Attribute im Schema. Wenn Sie eine Gruppe, ein Element, einen Typ, eine Attributgruppe oder ein Attribut in der Ansicht **Schema** auswählen, werden die Eigenschaften im rechten Bereich angezeigt. Sie können auch die einzelnen XSD-Dateien in der Ansicht **Schema** anzeigen.

Die Ansicht **Schema** liefert eine Liste der Namespaces und XSD-Dateien im Schema.

Die folgende Abbildung zeigt die Ansicht **Schema** von einem Schemaobjekt:



In der Ansicht **Schema** können Sie die folgenden Aktionen ausführen:

- Um die Liste der Schemakonstrukte anzuzeigen, erweitern Sie den Ordner **Direktiven**. Um den Namespace, das Präfix und den Speicherort anzuzeigen, wählen Sie in der Liste ein Schemakonstrukt aus.
- Um das Namespacepräfix, das generierte Präfix und den Speicherort anzuzeigen, wählen Sie einen Namespace aus. Das generierte Präfix kann geändert werden.
- Wenn Sie das Schemaobjekt als XSD-Datei anzeigen möchten, wählen Sie **Quelle** aus. Wenn das Objekt weitere Schemas enthält, können Sie auswählen, welche XSD-Datei angezeigt werden soll.
- Wählen Sie **Entwurf** aus, um eine alphabetische Liste der Elemente, Typen und Attribute in jedem Namespace des Schemas anzuzeigen. Sie können im Feld **Name** ein oder mehrere Zeichen eingeben, um Gruppen, Elemente, Typen, Attributgruppen und Attribute nach Namen zu filtern.
- Um die Elementeigenschaften anzuzeigen, wählen Sie eine Gruppe, ein Element, einen Typ, eine Attributgruppe oder ein Attribut aus. Im rechten Bereich des Developer Tools werden je nach dem von Ihnen ausgewählten Objekt unterschiedliche Felder angezeigt.

Bei der Anzeige der Typen können Sie sehen, ob ein Typ von einem anderen Typ abgeleitet wurde. Die Benutzeroberfläche zeigt den übergeordneten Typ an. Die Benutzeroberfläche zeigt außerdem, ob Werte durch Einschränkung oder Erweiterung an untergeordnete Elemente vererbt wurden.

Namespace-Eigenschaften

Die Ansicht **Namespace** enthält das Präfix und den Speicherort des ausgewählten Namespace.

Der Namespace für die einzelnen Schemadateien unterscheidet zwischen Elementen, die aus unterschiedlichen Quellen stammen, jedoch die gleichen Namen aufweisen. Eine URI-Referenz (Uniform Resource Identifier) definiert den Speicherort der Datei, die die Elemente und Attributnamen enthält.

Wenn Sie ein Schema importieren, das mehr als einen Namespace enthält, fügt das Developer Tool die Namespaces zum Schemaobjekt hinzu. Wenn die Schemadatei weitere Schemata enthält, sind die Namespaces für diese Schemata ebenfalls enthalten.

Das Developer-Tool erstellt ein generiertes Präfix für jeden Namespace. Wenn das Schema kein Präfix enthält, erzeugt das Developer Tool das Namespace-Präfix tns0 und erhöht die Präfixzahl für jedes zusätzliche Namespace-Präfix. Das Developer-Tool reserviert das Namespace-Präfix xs. Wenn Sie ein Schema importieren, das das Namespace-Präfix xs enthält, erstellt das Developer Tool das erzeugte Präfix xs1. Das Developer-Tool erhöht die Präfixzahl, wenn das Schema den generierten Präfixwert enthält.

Beispiel: Customer_Orders.xsd enthält einen Namespace. Das Schema enthält ein weiteres Schema, Customers.xsd. Das Customers-Schema weist einen anderen Namespace auf. Das Developer-Tool weist nun Präfix tns0 dem Customer_Orders-Namespace und Präfix tns1 dem Customers-Namespace zu.

Zum Anzeigen des Namespace-Speicherorts und -Präfixes wählen Sie einen Namespace in der Ansicht **Schema** aus.

Bei Web-Diensten aus mehreren Schemaobjekten muss jeder Namespace über ein eindeutiges Präfix verfügen. Sie können das generierte Präfix für jeden Namespace ändern.

Elementeigenschaften

Ein Element ist entweder ein einfacher oder ein komplexer Typ. Komplextypen enthalten weitere Typen. Wenn Sie in der Ansicht **Schema** ein Element auswählen, zeigt das Developer-Tool die untergeordneten Elemente und Eigenschaften in der rechten Maske auf dem Bildschirm an.

Die folgende Tabelle beschreibt die Elementeigenschaften, die bei Auswahl eines Elements angezeigt werden:

Eigenschaft	Beschreibung
Name	Der Elementname
Beschreibung	Beschreibung des Typs
Typ	Der Elementtyp

Die folgende Tabelle beschreibt die Eigenschaften der untergeordneten Elemente, die bei Auswahl eines Elements angezeigt werden:

Eigenschaft	Beschreibung
Name	Der Elementname

Eigenschaft	Beschreibung
Typ	Der Elementtyp
Mindestvorkommen	Die zulässige Mindestanzahl der Vorkommen eines Elements an einem Punkt in einer Instanz.
Maximalvorkommen	Die zulässige Höchstanzahl der Vorkommen eines Elements an einem Punkt in einer Instanz.
Beschreibung	Eine Beschreibung des Elements

Um weitere untergeordnete Elemente anzuzeigen, klicken Sie auf den Doppelpfeil in der Beschreibungsspalte und erweitern das Fenster.

Die folgende Tabelle beschreibt die zusätzlichen untergeordneten Elementeneigenschaften, die beim Erweitern der Beschreibungsspalte angezeigt werden:

Eigenschaft	Beschreibung
Festwert	Eine bestimmter Wert für ein Element, der unverändert bleibt
Nullwertfähig	Das Element kann Nullwerte aufweisen. Ein Nullwert-Element besitzt Element-Tags, aber keinen Wert und keinen Inhalt.
Abstrakt	Das Element ist ein abstrakter Typ. Eine Instanz muss von diesem Typ abgeleitete Typen aufweisen. Ohne abgeleitete Elementtypen ist ein abstrakter Typ kein gültiger Typ.
Mindestwert	Der Mindestwert für ein Element in einer Instanz.
Maximalwert	Der Maximalwert für ein Element in einer Instanz.
Minimale Länge	Die Mindestlänge eines Elements; die Länge wird je nach Elementtyp in Byte, Zeichen oder Einträgen angegeben.
Maximale Länge	Die maximale Länge eines Elements; die Länge wird je nach Elementtyp in Byte, Zeichen oder Einträgen angegeben.
Aufzählungen	Eine Liste aller zulässigen Werte für ein Element
Muster	Ein Ausdrucksmuster, das gültige Elementwerte definiert

Erweiterte Elementeneigenschaften

Zum Anzeigen der erweiterten Eigenschaften eines Elements wählen Sie das Element in der Ansicht **Schema** aus. Klicken Sie auf **Erweitert**.

Die folgende Tabelle beschreibt die erweiterten Elementeneigenschaften:

Eigenschaft	Beschreibung
Abstrakt	Das Element ist ein abstrakter Typ. SOAP-Meldungen müssen von diesem Typ abgeleitete Typen aufweisen. Ohne abgeleitete Elementtypen ist ein abstrakter Typ kein gültiger Typ.
Block	Verhindert, dass anstelle dieses Elements ein abgeleitetes Element in der Hierarchie angezeigt wird. Der Block-Wert kann „#all“ oder eine Liste mit Erweiterung, Einschränkung oder Substitution enthalten.

Eigenschaft	Beschreibung
Endphase	Verhindert, dass das Schema den einfachen Typ als abgeleiteten Typ erweitert oder einschränkt.
Substitutionsgruppe	Der Name des Elements, durch das das betreffende Element ersetzt werden soll
Nullwertfähig	Das Element kann Nullwerte aufweisen. Ein Nullwert-Element besitzt Element-Tags, aber keinen Wert und keinen Inhalt.

Einfachtyp-Eigenschaften

Ein Einfachtypelement ist ein Element, das unstrukturierten Text enthält. Wenn Sie ein Einfachtyp-Element in der Ansicht **Schema** auswählen, erscheinen Informationen über das Einfachtyp-Element in der rechten Maske.

Die folgende Tabelle beschreibt die Eigenschaften, die für Einfachtypen angezeigt werden:

Eigenschaft	Beschreibung
Typ	Der Name des Elements
Beschreibung	Eine Beschreibung des Elements
Auswahl	Gibt an, ob es sich bei dem Einfachtyp um ein Union-, Listen-, anyType- oder atomares Element handelt. Ein atomares Element enthält keine anderen Elemente oder Attribute.
Mitgliedstypen	Eine Liste der Typen in einem UNION-Konstrukt
Eintragstyp	Der Elementtyp
Basis	Der Basistyp eines atomaren Elements, z. B. Integer oder String
Minimale Länge	Die Mindestlänge eines Elements; die Länge wird je nach Elementtyp in Byte, Zeichen oder Einträgen angegeben.
Maximale Länge	Die Höchstlänge eines Elements; die Länge wird je nach Elementtyp in Byte, Zeichen oder Einträgen angegeben.
Leerräume reduzieren	Entfernt die Leerzeichen am Anfang und am Ende. Reduziert mehrere Leerzeichen zu einem einzigen Leerzeichen.
Aufzählungen	Beschränkt die Typen auf die Liste zulässiger Werte.
Muster	Beschränkt die Typen auf die von einem Musterausdruck definierten Werte.

Erweiterte Einfachtyp-Eigenschaften

Zum Anzeigen der erweiterten Eigenschaften eines Einfachtyps wählen Sie ihn in der Ansicht **Schema** aus. Klicken Sie auf **Erweitert**.

Die erweiterten Eigenschaften erscheinen unterhalb der Einfachtyp-Eigenschaften.

Die folgende Tabelle beschreibt die erweiterte Eigenschaft für einen Einfachtyp:

Eigenschaft	Beschreibung
Endphase	Verhindert, dass das Schema den einfachen Typ als abgeleiteten Typ erweitert oder einschränkt.

Komplextyp-Eigenschaften

Ein komplexer Typ ist ein Element, das weitere Elemente und Attribute enthält. Komplextypen können Elemente einfachen oder komplexen Typs enthalten. Wenn Sie in der Ansicht **Schema** einen Komplextyp auswählen, zeigt das Developer-Tool die untergeordneten Elemente und deren Eigenschaften in der rechten Maske auf dem Bildschirm an.

Die folgende Tabelle beschreibt die Komplextyp-Eigenschaften:

Eigenschaft	Beschreibung
Name	Der Name des Typs
Beschreibung	Beschreibung des Typs
Übernommen aus	Der Name des übergeordneten Typs
Übernommen von	Einschränkung oder Erweiterung; Komplextypen werden aus übergeordneten Typen abgeleitet. Der Komplextyp kann jedoch weniger Elemente oder Attribute als der übergeordnete Typ aufweisen. Er kann allerdings auch zusätzliche Elemente und Attribute aufnehmen.

Um die Eigenschaften der einzelnen Elemente im Komplextyp anzuzeigen, klicken Sie auf den Doppelpfeil in der Beschreibungsspalte und erweitern das Fenster.

Erweiterte Komplextyp-Eigenschaften

Zum Anzeigen der erweiterten Eigenschaften eines Komplextyps wählen Sie das Element in der Ansicht **Schema** aus. Klicken Sie auf **Erweitert**.

Die folgende Tabelle beschreibt die erweiterten Eigenschaften eines komplexen Elements oder Typs:

Eigenschaft	Beschreibung
Abstrakt	Das Element ist ein abstrakter Typ. SOAP-Meldungen müssen von diesem Typ abgeleitete Typen aufweisen. Ohne abgeleitete Elementtypen ist ein abstrakter Typ kein gültiger Typ.
Block	Verhindert, dass anstelle dieses Elements ein abgeleitetes Element im Schema angezeigt wird. Der Block-Wert kann „#all“ oder eine Liste mit Erweiterung, Einschränkung oder Substitution enthalten.
Endphase	Verhindert, dass das Schema den einfachen Typ als abgeleiteten Typ erweitert oder einschränkt.
Substitutionsgruppe	Der Name des Elements, durch das das betreffende Element ersetzt werden soll
Nullwertfähig	Das Element kann Nullwerte aufweisen. Ein Nullwert-Element besitzt Element-Tags, aber keinen Wert und keinen Inhalt.

Attributeigenschaften

Ein Attribut ist ein Einfachtyp. Elemente und Komplextypen enthalten Attribute. Globale Attribute erscheinen als Teil des Schemas. Wenn Sie in der Ansicht **Schema** ein globales Attribut auswählen, zeigt das Developer-Tool die Attributeigenschaften und dazugehörige Typeigenschaften in der rechten Maske auf dem Bildschirm an.

Die folgende Tabelle beschreibt die Attributeigenschaften:

Eigenschaft	Beschreibung
Name	Der Name des Attributs
Beschreibung	Die Beschreibung des Attributs
Typ	Der Attributtyp
Wert	Der Wert des Attributtyps; gibt an, ob der Wert des Attributtyps fest ist oder einen Standardwert aufweist. Wenn kein Wert definiert ist, lautet der Eintrag „Standard=0“.

Die folgende Tabelle beschreibt die Typeigenschaften:

Eigenschaft	Beschreibung
Minimale Länge	Die Mindestlänge des Strings; die Länge wird je nach Typ in Byte, Zeichen oder Einträgen angegeben.
Maximale Länge	Die Höchstlänge des Strings; die Länge wird je nach Typ in Byte, Zeichen oder Einträgen angegeben.
Leerräume reduzieren	Entfernt die Leerzeichen am Anfang und am Ende. Reduziert mehrere Leerzeichen zu einem einzigen Leerzeichen.
Aufzählungen	Beschränkt die Typen auf die Liste zulässiger Werte.
Muster	Beschränkt die Typen auf die von einem Musterausdruck definierten Werte.

Schemaobjekt-Ansicht „Erweitert“

Sie können die erweiterten Eigenschaften des Schemaobjekts anzeigen.

Die folgende Tabelle beschreibt die erweiterten Eigenschaften eines Schemaobjekts:

Name	Wert	Beschreibung
elementFormDefault	Qualifiziert oder Unqualifiziert	Bestimmt, ob Elemente einen Namespace aufweisen müssen oder nicht. Das Schema qualifiziert Elemente mit einem Präfix oder durch Deklaration eines Ziel-Namespace. Unqualifiziert bedeutet, dass die Elemente keinen Namespace benötigen.

Name	Wert	Beschreibung
attributeFormDefault	Qualifiziert oder Unqualifiziert	Bestimmt, ob lokal deklarierte Attribute einen Namespace aufweisen müssen oder nicht. Das Schema qualifiziert Attribute mit einem Präfix oder durch Deklaration eines Ziel-Namespace. Unqualifiziert bedeutet, dass die Attribute keinen Namespace benötigen.
Speicherort	Der vollständige Pfad zur XSD-Datei	Der Speicherort der XSD-Datei zum Zeitpunkt des Imports

Erstellen eines Schemaobjekts

Sie können eine hierarchische Schemadatei oder Beispieldatei importieren, um ein Schemaobjekt im Repository zu erstellen.

1. Wählen Sie in der Ansicht **Object Explorer** ein Projekt oder einen Ordner aus.
2. Klicken Sie auf **Datei > Neu > Schema**.
Das Dialogfeld **Neues Schema** wird geöffnet.
3. Wählen Sie zum Importieren einer Schemadatei die Option **Aus Schema erstellen** aus. Navigieren Sie dann zu einer hierarchischen Schemadatei und wählen Sie sie aus.

Zum Durchsuchen können Sie einen URI oder einen Speicherort im Dateisystem eingeben. Das Developer Tool validiert das ausgewählte Schema. Überprüfen Sie die Validierungsnachrichten. Sie können eine Avro-, Parquet-, JSON- oder XSD-Schemadatei auswählen.
Hinweis: Wenn der URI Zeichen außerhalb des englischen Zeichensatzes enthält, kann der Import fehlschlagen. Kopieren Sie den URI in die Adressleiste eines Browsers. Kopieren Sie den Speicherort wieder aus dem Browser. Das Developer Tool akzeptiert den kodierten URI aus dem Browser.
4. Wählen Sie zum Erstellen eines Schemas aus einer Beispieldatei die Option **Aus einer Beispieldatei erstellen** aus. Navigieren Sie dann zu einer hierarchischen Datei und wählen Sie sie aus.

Sie können eine Avro-, Parquet-, JSON- oder XML-Datei auswählen.
Hinweis: Wenn Sie eine Datei mit einer anderen Erweiterung auswählen, die Avro-, Parquet-, JSON- oder XML-Inhalt enthält, erkennt der Assistent den Dateiinhalt.
5. Optional können Sie den Schemanamen auch ändern.
6. Klicken Sie auf **Weiter**, um eine Liste der Elemente und Typen im Schema anzuzeigen.
7. Klicken Sie auf **Fertig stellen**, um das Schema zu importieren.

Das Schema wird unter „Schemaobjekte“ im **Objekt-Explorer** aufgelistet. Das Developer Tool speichert das Schema als XSD-Datei.
8. Zum Ändern des generierten Präfixes eines Schema-Namespace wählen Sie den Namespace im **Objekt-Explorer** aus. Ändern Sie die Eigenschaft **Generiertes Präfix** in der Ansicht **Namespace**.

Schema-Updates

Sie können ein Schemaobjekt aktualisieren, wenn Elemente, Attribute, Typen oder andere Schemakomponenten geändert wurden. Wenn Sie ein Schemaobjekt aktualisieren, aktualisiert das Developer Tool Objekte, die das Schema verwenden.

Sie können Schemaobjekte mit den folgenden Methoden aktualisieren:

Synchronisieren Sie das Schema.

Synchronisieren Sie ein Schemaobjekt, wenn Sie die Schemadateien außerhalb des Developer Tools aktualisieren. Wenn Sie eine Schemaobjekt synchronisieren, importiert das Developer Tool alle XSD-Schemadateien erneut, die Änderungen enthalten.

Bearbeiten Sie eine Schemadatei.

Bearbeiten Sie eine Schemadatei, wenn Sie eine Datei im Developer Tool aktualisieren möchten. Wenn Sie eine Schemadatei bearbeiten, öffnet das Developer Tool die Datei in dem Editor, den Sie für XSD-Dateien verwenden. Sie können die Datei in einem anderen Editor öffnen oder im Developer Tool einen Standardeditor für XSD-Dateien festlegen.

Sie können mit einem Schema Elementtypen in einem Web-Dienst definieren. Wenn Sie ein Schema aktualisieren, das in der WSDL eines Web-Diensts enthalten ist, aktualisiert das Developer Tool den Web-Dienst und markiert diesen als geändert, wenn Sie ihn öffnen. Wenn das Developer Tool das neue Schema mit dem alten Schema vergleicht, identifiziert es Schemakomponenten anhand der name-Attribute.

Wenn kein name-Attribut geändert wurde, aktualisiert das Developer Tool den Web-Dienst mit den Schemaänderungen. Beispiel: Sie bearbeiten im Developer Tool eine Schemadatei und ändern das maxOccurs-Attribut für das Element „Item“ von „10“ in „unbounded“. Wenn Sie die Datei speichern, aktualisiert das Developer- Tool das maxOccurs-Attribut in jedem Web-Dienst, der auf das Item-Element verweist.

Wenn sich ein name-Attribut ändert, markiert das Developer Tool den Web-Dienst als geändert, sobald Sie ihn öffnen. Beispiel: Sie bearbeiten ein Schema außerhalb des Developer Tools und ändern den Namen eines komplexen Elementtyps von „Order“ in „CustOrder“. Anschließend synchronisieren Sie das Schema. Wenn Sie einen Web-Dienst öffnen, der auf das Element verweist, markiert das Developer Tool den Web-Dienst-Namen im Editor mit einem Sternchen, um anzugeben, dass der Web-Dienst Änderungen enthält. Das Developer Tool fügt dem Web-Dienst das CustOrder-Element hinzu, entfernt jedoch nicht den Order-Elementtyp. Da das Developer Tool den Typ für das Order-Element nicht mehr bestimmen kann, ändert es den Elementtyp in xs:string.

Schemasynchronisierung

Sie können ein Schemaobjekt synchronisieren, wenn sich die Schemakomponenten ändern. Wenn Sie ein Schemaobjekt synchronisieren, importiert das Developer Tool erneut die Objektmetadaten aus den Schemadateien.

Verwenden Sie Schemasynchronisierung, wenn Sie außerhalb des Developer Tools komplexe Änderungen am Schemaobjekt vornehmen. Beispielsweise sollten Sie ein Schema synchronisieren, nachdem Sie die folgenden Aktionen ausgeführt haben:

- Ausführen von Änderungen an mehreren Schemadateien.
- Hinzufügen oder Entfernen von Schemadateien zum Schema bzw. aus dem Schema.
- Ändern von import- oder include-Elementen.

Das Developer Tool validiert die Schemadateien, bevor es das Schemaobjekt aktualisiert. Wenn die Schemadateien Fehler enthalten, werden sie nicht vom Developer Tool importiert.

Um ein Schemaobjekt zu synchronisieren, klicken Sie in der Ansicht **Objekt-Explorer** mit der rechten Maustaste auf das Schemaobjekt und wählen Sie **Synchronisieren** aus.

Bearbeitungen von Schemadateien

Sie können eine Schemadatei im Developer Tool bearbeiten, um Schemakomponenten zu aktualisieren.

Bearbeiten Sie eine Schemadatei im Developer Tool, um geringfügige Änderungen an einer kleinen Anzahl von Dateien vorzunehmen. Angenommen, Sie nehmen an einer Schemadatei eine der folgenden geringfügigen Aktualisierungen vor:

- Ändern des minOccurs- oder maxOccurs-Attributs für ein Element.
- Hinzufügen eines Attributs zu einem komplexen Typ.
- Ändern eines einfachen Objekttyps.

Wenn Sie eine Schemadatei bearbeiten, öffnet das Developer Tool eine temporäre Kopie der Schemadatei in einem Editor. Sie können Schemadateien mit dem Systemeditor bearbeiten, den Sie für XSD-Dateien verwenden, oder Sie können einen anderen Editor auswählen. Sie können auch den Standardeditor des Developer Tools für XSD-Dateien festlegen. Speichern Sie die temporäre Schemadatei, nachdem Sie sie bearbeitet haben.

Das Developer Tool validiert die temporäre Datei, bevor es das Schemaobjekt aktualisiert. Wenn die Schemadatei Fehler enthält oder Komponenten enthält, die einen Konflikt mit anderen Schemadateien im Schemaobjekt verursachen, wird die Datei nicht vom Developer Tool importiert.

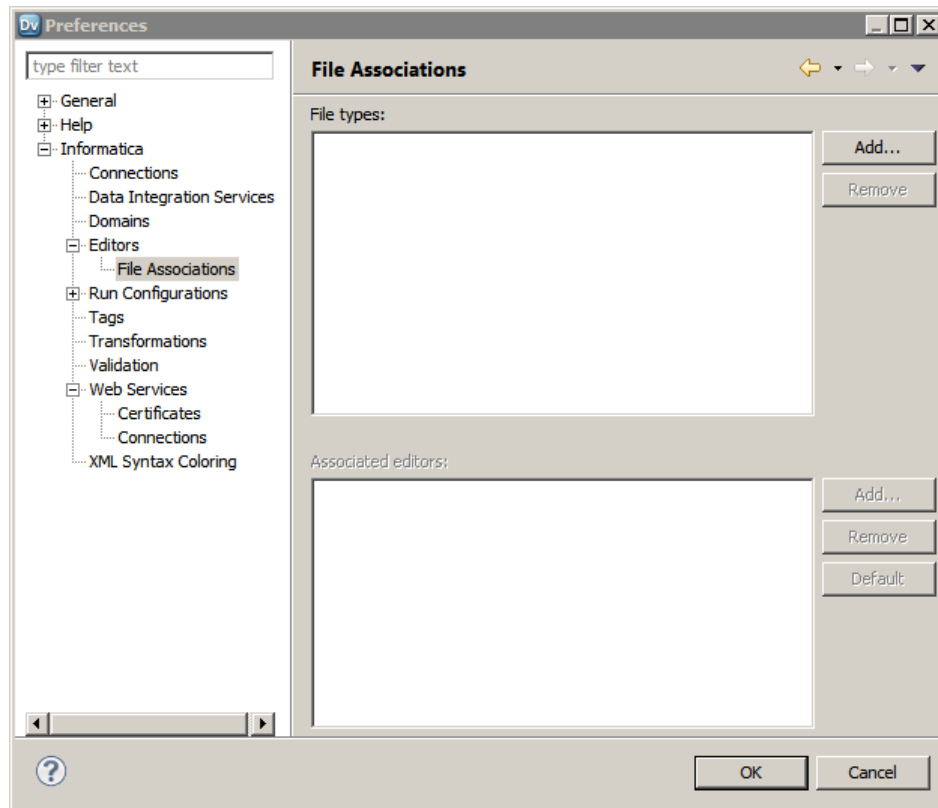
Hinweis: Wenn Sie die temporäre Schemadatei bearbeiten und speichern, aktualisiert das Developer-Tool nicht die Schemadatei, die in der Liste **Schemaspeicherorte** enthalten ist. Wenn Sie ein Schemaobjekt synchronisieren, nachdem Sie eine Schemadatei im Developer Tool bearbeitet haben, werden Ihre Bearbeitungen durch die Synchronisierung überschrieben.

Festlegen eines Standardeditors für Schemadateien

Sie können den Standardeditor festlegen, der vom Developer Tool geöffnet wird, wenn Sie eine Schemadatei bearbeiten.

1. Klicken Sie auf **Fenster > Einstellungen**.
Das Dialogfeld **Einstellungen** wird angezeigt.
2. Klicken Sie auf **Editoren > Dateizuordnungen**.

Die Seite **Dateizuordnungen** des Dialogfelds **Einstellungen** wird angezeigt.



3. Klicken Sie neben dem Bereich **Dateitypen** auf **Hinzufügen**.
Das Dialogfeld **Dateityp hinzufügen** wird angezeigt.
4. Geben Sie als Dateityp `.xsd` ein und klicken Sie auf **OK**.
5. Klicken Sie neben dem Bereich **Zugeordnete Editoren** auf **Hinzufügen**.
Das Dialogfeld **Editorauswahl** wird angezeigt.
6. Wählen Sie in der Liste der Editoren einen Editor aus, oder klicken Sie auf **Durchsuchen**, um einen anderen Editor auszuwählen. Klicken Sie dann auf **OK**.
Der von Ihnen ausgewählte Editor wird in der Liste **Zugeordnete Editoren** angezeigt.
7. Optional können Sie der Liste **Zugeordnete Editoren** weitere Editoren hinzufügen.
8. Wenn Sie mehrere Editoren hinzufügen, können Sie einen anderen Standardeditor festlegen. Wählen Sie einen Editor aus und klicken Sie auf **Standard**.
9. Klicken Sie auf **OK**.

Bearbeiten einer Schemadatei

Sie können jede Schemadatei in einem Schemaobjekt bearbeiten.

1. Öffnen Sie ein Schemaobjekt.
2. Wählen Sie die Ansicht **Übersicht** aus.

Die Ansicht **Übersicht** des Schemaobjekts wird eingeblendet.

so_Partners x

Übersicht

Allgemein

Name:

Beschreibung:

Ziel-Namespace:

Speicherorte des Schemas

Speicherort des Schemas	Namespace
file://infa2008deu2/infa_sha...	(Kein Ziel-Namespace angegeben)

Hinzufügen...
Entfernen
Öffnen mit...

Übersicht | Schema | Erweitert

3. Wählen Sie in der Liste **Schemaspeicherorte** eine Schemadatei aus.
4. Klicken Sie auf **Öffnen mit** und wählen Sie eine der folgenden Optionen aus:

Option	Beschreibung
Systemeditor	Die Schemadatei wird in dem Editor geöffnet, den das Betriebssystem für XSD-Dateien verwendet.
Standardeditor	Die Schemadatei wird in dem Editor geöffnet, den Sie im Developer Tool als Standardeditor festlegen. Diese Option wird angezeigt, wenn Sie einen Standardeditor festlegen.
Andere	Sie wählen den Editor aus, in dem die Schemadatei geöffnet werden soll.

Das Developer Tool öffnet eine temporäre Kopie der Schemadatei.

5. Aktualisieren Sie die temporäre Schemadatei, speichern Sie die Änderungen und schließen Sie den Editor.
Sie werden vom Developer Tool aufgefordert, das Schemaobjekt zu aktualisieren.
6. Klicken Sie zum Aktualisieren des Schemaobjekts auf **Schemaobjekt aktualisieren**.
Das Developer Tool aktualisiert die Schemadatei mit den Änderungen, die Sie vorgenommen haben.

KAPITEL 9

Eingabeaufforderung

Dieses Kapitel umfasst die folgenden Themen:

- [Überblick über die Befehlszeilenschnittstelle, 137](#)
- [CM_console, 137](#)

Überblick über die Befehlszeilenschnittstelle

Sie können einen Datenumwandlungsdienst von der Befehlszeile der Maschine aus ausführen, die als Host für den Dienst eingerichtet ist.

Exportieren Sie eine Datenprozessor-Umwandlung als Dienst in das Verzeichnis `/ServiceDB` auf dem System, auf dem Sie den Datenumwandlungsdienst ausführen möchten. Führen Sie den Befehl `CM_console` aus.

CM_console

Führt einen Datenumwandlungsdienst aus.

Der `CM_console`-Befehl verwendet die folgende Syntax:

```
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]
```

Hinweis: Setzen Sie keine Leerstelle zwischen eine Option und ihre Argumente.

In der folgenden Tabelle werden CM_console-Optionen und -Argumente beschrieben:

Option	Argument	Beschreibung
-	ServiceName	Erforderlich. Legt den Namen des Dienstes fest.
-f	InputDocument	Optional. Legt Pfad und Dateinamen im lokalen Dateisystem fest. Standardmäßig verwendet der Dienst das Dokument, das in der Eigenschaft example_source der Startkomponente definiert wurde.
-t	InputDocument	Optional. Gibt einen String mit doppelten Anführungszeichen an.
-u	InputDocument	Optional. Legt eine URL fest.
-a	ServiceParameter=InitialValue	Optional. Legt einen Eingabeparameter für den Dienst fest. ServiceParameter ist der Name einer im Dienst definierten Variablen. InitialValue muss einen Datentyp haben, der für die definierte Variable gültig ist. Sie können mehrere Eingabeparameter, getrennt durch Leerstellen, eingeben.
-o	FileName [Path]FileName	Optional. Lenkt die Ausgabe nach Path/FileName um. Wenn Sie nur FileName eingeben, müssen Sie den Pfad mit der Option -r angeben. Standardmäßig leitet der CM_console-Befehl Ausgaben zur Anzeige um.
-r	curr	Optional. Legt das Verzeichnis fest, von dem aus Sie den CM_console-Befehl ausgeführt haben.
-r	res	Optional. Gibt das Unterverzeichnis <i>results</i> unter dem Verzeichnis an, das den Dienst im Dateisystemrepository enthält.
-r	spec=OutputDirectory	Optional. Gibt ein Verzeichnis im lokalen Dateisystem an.
-r	guid	Optional. Gibt ein Verzeichnis mit einem eindeutigen Namen unter dem Verzeichnis <i>CMReports/tmp</i> an. Mithilfe des Konfigurations-Editors können Sie den Speicherort dieses Verzeichnisses ändern.
-l	UserName	Erforderlich, wenn Sie die HTTP-Authentifizierung verwenden. Legt den Benutzernamen für die HTTP-Authentifizierung fest. Hinweis: Diese Option ist der Kleinbuchstabe von "L".
-p	Password	Erforderlich, wenn Sie die HTTP-Authentifizierung verwenden. Legt das Kennwort für die HTTP-Authentifizierung fest.
-v	-	Optional. Zeigt ausführliche Informationen unter anderem zur Datenumwandlungsversion, zur Version der Datenumwandlungssyntax, zur Setup-Paket-ID und zur Lizenz an.
-S	-	Erforderlich, wenn die Startkomponente des Dienstes ein Streamer ist. Zur Definition der Eingabedatei muss auch die Option -f enthalten sein.
-xf	InputPortName=InputDocument	Optional. InputPortName legt den Namen eines AdditionalInputPort -Objekts fest, das im Dienst definiert ist. InputDocument definiert Pfad und Dateinamen im lokalen Dateisystem. Sie können mehrere Eingabeports, getrennt durch Leerstellen, eingeben.

Option	Argument	Beschreibung
-xt	InputPortName=InputDocument	Optional. InputPortName legt den Namen eines AdditionalInputPort -Objekts fest, das im Dienst definiert ist. InputDocument legt einen String mit doppelten Anführungszeichen fest. Sie können mehrere Eingabeports, getrennt durch Leerstellen, eingeben.
-xu	InputPortName=InputDocument	Optional. InputPortName legt den Namen eines AdditionalInputPort -Objekts fest, das im Dienst definiert ist. InputDocument legt eine URL fest. Sie können mehrere Eingabeports, getrennt durch Leerstellen, eingeben.
-xo	OutputPortName=OutputDocument	Optional. OutputPortName legt den Namen eines AdditionalOutputPort -Objekts fest, das im Dienst definiert ist. OutputDocument definiert Pfad und Dateinamen im lokalen Dateisystem. Sie können mehrere Ausgabeports, getrennt durch Leerstellen, eingeben.
-e	-	Optional. Standardmäßig endet der CM_console-Befehl mit einem Exit-Code 1 für "Erfolg" und größer als 1 für "Fehler". Wenn Sie den Switch -e einfügen, endet der CM_console-Befehl mit einem Exit-Code 0 für "Erfolg" und größer als 1 für "Fehler".

Beispiel:

```
CM_console XYZparser -fInputFile.txt -aMaxLines=1000 -oResults.xml -rcurr
```

Dieses Beispiel ruft den XYZparser-Dienst mithilfe von `InputFile.txt` als Haupteingabedokument auf. Es gibt dem Parameter *MaxLines* den Wert 1000 und schreibt die Ausgabe in die Datei `Results.xml` in dem Verzeichnis, von dem aus Sie den CM_console-Befehl ausgeführt haben.

KAPITEL 10

Skripte

Dieses Kapitel umfasst die folgenden Themen:

- [Übersicht über Skripts, 140](#)
- [Skriptkomponenten, 141](#)
- [Eigenschaften der Skriptkomponenten, 143](#)
- [Skriptstartkomponenten, 145](#)
- [Beispielquellen, 145](#)
- [IntelliScript-Editor, 147](#)
- [Validieren eines Skripts, 147](#)
- [Muster-Skripte, 148](#)

Übersicht über Skripts

Ein Skript führt komplexe Umwandlungen für Eingabedaten aus und schreibt Ausgabedaten. Erstellen Sie ein Skript auf der Registerkarte **Objekte** der Datenprozessor-Umwandlung. Verwenden Sie den IntelliScript-Editor, um ein Skript anzuzeigen, Komponenten hinzuzufügen und zu konfigurieren und die Startkomponente für ein Skript einzurichten.

Verwenden Sie ein Skript, um ein oder mehrere Dokumente in jedem Format, z. B. HL7, PDF, XML oder Word zu lesen. Sie können ein oder mehrere Dokumente in jedem Format schreiben. Sie können die Ausgabe eines Skripts in das lokale Dateisystem schreiben oder die Ausgabe über Ausgabeports der Datenprozessor-Umwandlung ausgeben.

Dieses Skript besteht aus Komponenten, die Eingabe- und Ausgabedokumente, Geschäftslogik, Variablen zur temporären Aufnahme von Daten und Konfigurationseinstellungen definieren. Die Komponenten sind als hierarchischer Baum angeordnet. Wenn die Umwandlung ein Skript ausführt, beginnt sie mit der Verarbeitung in der Komponente, die Sie als Startkomponente festgelegt haben.

Wenn Sie ein Skript konfigurieren, richten Sie Beispielquellen ein, die Beispieldaten für jeden Eingabeport enthalten. Wenn Sie die Umwandlung über die **Daten-Viewer**-Ansicht ausführen, liest die Umwandlung die Dokumente der Beispielquelle. Wenn Sie die Umwandlung in einem Mapping ausführen, liest die Umwandlung die Dokumente, die sie über ihre Eingabeports erhält.

Die Datenprozessor-Umwandlung, die das Skript enthält, muss ein Schema für jedes XML-Dokument referenzieren, das das Skript liest oder schreibt.

Skriptkomponenten

Eine Skriptkomponente ist eine Zeile oder eine Gruppe von Zeilen in einem Skript, das Eingabe- und Ausgabedokumente, Geschäftslogik, Variablen zur temporären Aufnahme von Daten und Konfigurationseinstellungen definiert. Die Komponenten eines Skripts werden in einer hierarchischen Struktur angezeigt. Einige Komponenten werden auf der globalen Ebene des Skripts angezeigt, andere als untergeordnete Komponenten.

Die globale Ebene des Skripts enthält Startkomponenten, Variablen und andere Komponenten wie zusätzliche Eingabeports und Transformer. Eine Komponente auf der globalen Ebene muss einen Namen haben.

Eine Komponente kann Eigenschaften haben, die das Verhalten der Komponente steuern. Die Eigenschaften einer Komponente erscheinen darin geschachtelt. Eine Eigenschaft kann auf einer Zeile oder als Hierarchie von Eigenschaften erscheinen. Sie können die Eigenschaften einiger Komponenten konfigurieren, um Standardeinstellungen zu überschreiben, die auf die Datenprozessor-Umwandlung angewendet werden.

Einige Komponenten, Parser oder Mapper können andere Komponenten enthalten, beispielsweise Transformer oder **RunParser**-Aktionen. Optional können Sie die Eigenschaft **name** einer untergeordneten Komponente konfigurieren.

Komponententypen

Der Kontext des Skripts legt die Komponententypen fest, die Sie hinzufügen können.

Beispiel: Anker müssen geschachtelt in Parser, Mapper oder Serializer angezeigt werden. Ebenso können weitere Eingabeports und Ausgabeports nur auf der globalen Ebene des Skripts angezeigt werden.

Die folgende Tabelle beschreibt die Komponententypen, die Sie zu einem Skript hinzufügen können:

Komponententyp	Beschreibung
Aktion	Nimmt Daten aus einem Datenbehälter und führt eine Operation damit aus. Beispiel: Die Aktion RunParser führt einen Parser aus.
Anker	Identifiziert einen Abschnitt des Eingabedokuments.
Dokumentprozessor	Führt eine komplexe Umwandlung an einem Eingabedokument aus. Beispiel: Der PdfToTxt_4 -Dokumentprozessor wandelt ein PDF-Dokument in Nur-Text um.
Format	Definiert das Format der Dokumente, die der Parser verarbeitet.
Locator	Isoliert eine Einzelinstanz eines mehrfach vorkommenden Datenbehälters.
Mapper	Liest und schreibt XML-Dokumente. Kann als Startkomponente eingerichtet werden.
Benachrichtigung	Schreibt eine Meldung an die Standardausgabe oder ein Protokoll. Beispiel: Die Benachrichtigung XsdValidationError zeigt an, dass das Eingabedokument gegenüber dem Schema, das sie definiert, nicht gültig ist.
Parser	Liest und schreibt Dokumente in jedem Format. Kann als Startkomponente eingerichtet werden.
Skript-Port	Definiert ein Eingabe- oder Ausgabedokument.
Serializer	Liest XML-Dokumente und schreibt Dokumente in jedem Format. Kann als Startkomponente eingerichtet werden.

Komponententyp	Beschreibung
Streamer	Teilt große Eingabedateien in Chunks und gibt die Chunks an einen Parser, Mapper oder Serializer weiter. Kann als Startkomponente eingerichtet werden.
Transformer	Wandelt eine Eingabe-String in einen Ausgabe-String um. Kann als Startkomponente eingerichtet werden.
Validierer	Legt fest, ob Eingabedaten einer bestimmten Datendefinition entsprechen.
Variable	Speichert Daten, die das Skript über einen Dienstparameter erhält, oder Daten aus einer Komponente des Skripts.

Komponentennamen

Eine Komponente wird durch ihren Namen im Skript, im Protokoll und in der Ansicht **Datenprozessor-Ereignisse** identifiziert.

Wenn die Datenprozessor-Umwandlung die Anweisungen in einer Komponente ausführt, generiert die Komponente ein Ereignis, das in der Ansicht **Datenprozessor-Ereignisse** und im Protokoll erscheint. Eine Komponente, die auf der globalen Ebene des Skripts angezeigt wird, muss einen Namen haben. Eine Komponente, die als untergeordnete Komponente einer anderen Komponente erscheint, kann einen Namen haben, den Sie mit der Eigenschaft **name** konfigurieren.

Der Name einer Komponente muss mit einem Buchstaben beginnen, darf nur einfache Buchstaben (A-Z, a-z), Ziffern (0-9) und Unterstriche (_) enthalten, und ist auf 127 Zeichen begrenzt.

Hinzufügen einer globalen Komponente

Definieren Sie eine Komponente global, wenn Sie sie an mehreren Stellen im Skript verwenden wollen oder die Komponente nur auf der globalen Ebene angezeigt werden kann.

1. Doppelklicken Sie am unteren Ende der globalen Ebene des Skripts auf das linke Auslassungszeichen (...).
Ein Textfeld wird angezeigt.
2. Geben Sie den Namen der Komponente ein und drücken Sie die **Eingabetaste**.
3. Doppelklicken Sie auf das rechte Auslassungszeichen.
Ein Listefeld wird angezeigt.
4. Klicken Sie auf den Pfeil nach unten und wählen Sie den Typ der Komponente aus, die Sie hinzufügen möchten.
Die globale Komponente wird im Skript angezeigt.
5. Legen Sie gegebenenfalls die Eigenschaften einer Komponente fest.

Hinzufügen einer lokalen Komponente

Definieren Sie eine Komponente lokal, wenn sie an nur einer Position des Skripts verwendet werden soll, oder die Komponente nur als untergeordnete Komponente angezeigt werden kann.

1. An der Stelle des Skripts, an der Sie eine Komponente einfügen möchten, doppelklicken Sie auf das Auslassungszeichen.
Ein Listefeld wird angezeigt.
2. Klicken Sie auf den Pfeil nach unten rechts vom Listefeld.
Eine Liste von verfügbaren Komponenten erscheint, einschließlich benannter globaler Komponenten.
3. Wählen Sie eine Komponente aus.
Die Komponente wird im Skript angezeigt.
4. Legen Sie gegebenenfalls die Eigenschaften einer Komponente fest.

Eigenschaften der Skriptkomponenten

Die Eigenschaften einer Skriptkomponente definieren die Funktionen der Komponente. Eine Komponente kann eine oder mehr Eigenschaften haben. Die Eigenschaften erscheinen geschachtelt innerhalb der Komponente. Alle Komponenten desselben Typs haben dieselben Eigenschaften.

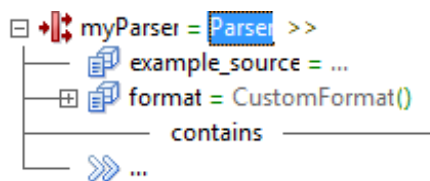
Beispiel: Die Eigenschaft **example_source** eines Parsers definiert Beispieltext, den der Parser verwendet, wenn Sie die Umwandlung aus der Ansicht **Daten-Viewer** ausführen.

Einfache Eigenschaften

Die einfachen Eigenschaften einer Komponente sind Eigenschaften, die vom IntelliScript-Editor immer angezeigt werden.

Die meisten Benutzer müssen lediglich die einfachen Eigenschaften ändern.

In der folgenden Abbildung werden die einfachen Eigenschaften einer **Parser**-Komponente dargestellt:



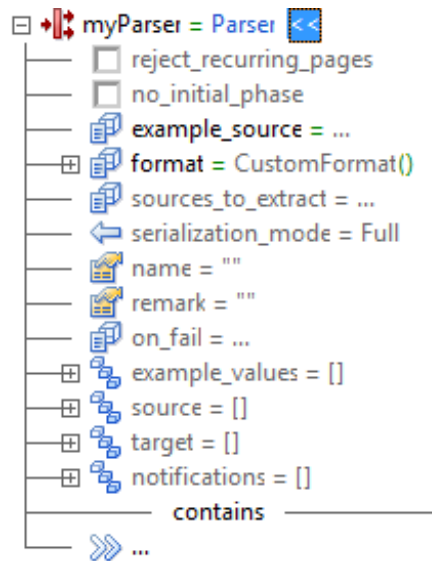
Erweiterte Eigenschaften

Die erweiterten Eigenschaften einer Komponente werden normalerweise auf die Standardwerte eingestellt, die die Benutzer in der Regel nicht ändern.

Der IntelliScript-Editor zeigt normalerweise nur die erweiterten Eigenschaften, die Sie auf einen anderen als den Standardwert eingestellt haben.

Um die nicht angezeigten Eigenschaften anzuzeigen, klicken Sie auf den Doppelpfeil nach rechts in der ersten Zeile.

Die folgende Abbildung zeigt alle Eigenschaften einer **Parser**-Komponente:



Eigenschaftswerte von Komponenten

Sie legen die Werte für die Eigenschaften einer Komponente fest.

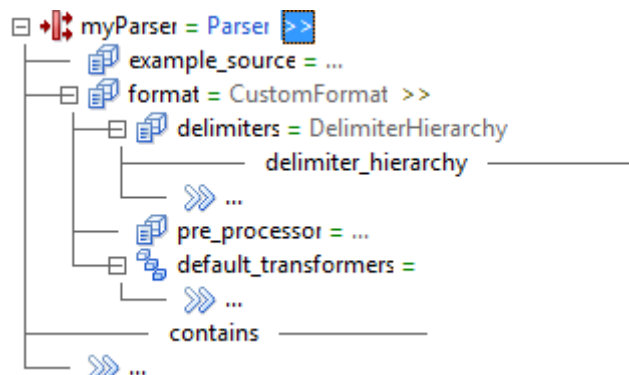
Hat die Eigenschaft einen Boolean-Wert, wird dieser als Kontrollkästchen links neben dem Eigenschaftsnamen angezeigt. Beispiel: Die **optional**-Eigenschaft einer **Content**-Komponente ist Boolean.

Ist der Wert ein String, wird er rechts vom Eigenschaftsnamen in doppelten Anführungszeichen angezeigt. Gültige Werte sind Strings mit gültigen alphanumerischen Zeichen, Symbolen oder Steuerungszeichen, jedoch ohne Null-Zeichen. Zur Eingabe eines Zeichens in ein Textfeld, das nicht auf der Tastatur vorhanden ist, drücken Sie **STRG+A** und geben Sie den dreistelligen Dezimalcode für dieses Zeichen ein. Beispiel: Geben Sie **STRG+A** 010 für einen Zeilenvorschub ein, oder **STRG+A** 255 für den isländischen Buchstaben "Thorn" (þ). Beispiel: Der Wert der **expression**-Eigenschaft der Komponente **CalculateValue** ist ein String.

Ist der Wert ein String, wird er rechts vom Eigenschaftsnamen angezeigt. Wenn Sie den Wert bearbeiten, wird ein Listefeld angezeigt. Beispiel: Die **val_type**-Eigenschaft einer **Variable**-Komponente ist eine Auswahl.

Ist der Wert ein hierarchischer Eigenschaften-Baum, wird er rechts und unterhalb des Eigenschaftsnamens angezeigt. Beispiel: Wenn Sie die **Format**-Eigenschaft einer **Parser**-Komponente auf CustomFormat setzen, wird ein Baum weiterer Eigenschaften angezeigt.

Die folgende Abbildung zeigt die **Format**-Eigenschaft einer **Parser**-Komponente, die als Baum angezeigt wird:



Skriptstartkomponenten

Die Startkomponente eines Skripts definiert den Eintrittspunkt, an dem die Datenprozessor-Umwandlung mit der Verarbeitung des Skripts beginnt. Die Startkomponente muss auf der globalen Ebene des Skripts angezeigt werden.

Sie können Parser, Mapper, Serializer, Streamer oder Transformer als Startkomponente verwenden.

Sie können die Startkomponente auf der Registerkarte **Übersicht** einer Datenprozessor-Umwandlung einrichten. Wenn Sie den IntelliScript-Editor zum Einrichten der Startkomponente eines Skripts verwenden, wird die Skriptstartkomponente zur Startkomponente der Datenprozessor-Umwandlung.

Einrichten der Startkomponente mit dem IntelliScript-Editor

Mithilfe des IntelliScript-Editors können Sie eine Skriptkomponente als Startkomponente für die Datenprozessor-Umwandlung festlegen. Sie müssen die Startkomponente zum Ausführen des Skripts festlegen. Sie müssen die Startkomponente so festlegen, dass die Beispielquelle in der Maske **Eingabe** der Ansicht **Daten-Viewer** angezeigt wird.

1. Öffnen Sie im IntelliScript-Editor ein Skript.
2. Klicken Sie mit der rechten Maustaste auf eine Komponente, die auf der globalen Ebene des Skripts angezeigt wird, und wählen Sie anschließend **Als Startkomponente festlegen** aus.

Beispielquellen

Eine Beispielquelle ist ein Dokument, das Beispieleingabedaten enthält, die das Skript während der Entwicklungszeit verarbeitet. Sie konfigurieren eine Beispielquelle für jeden Parser, Mapper, Serializer oder weiteren Eingabeport. Die Beispielquelle enthält denselben Datentyp, den die Datenprozessor-Umwandlung vom Eingabeport empfängt.

Standardmäßig zeigt die **Daten-Viewer**-Ansicht die für die Startkomponente definierte Beispielquelle an. Sie können auch die Beispielquelle jeder anderen Komponente anzeigen, die eine Beispielquelle definiert. Wenn Sie ein Skript über die Ansicht **Daten-Viewer** ausführen, liest die Datenprozessor-Umwandlung die Dokumente der Beispielquelle.

Sie können folgende Typen des Beispiel-Quelldokuments konfigurieren:

- LocalFile. Eine Datei im lokalen Dateisystem.
- Text. Ein im Skript hartcodierter String.
- URL. Eine Datei im lokalen Netzwerk oder im Internet.

Hinweis: Wenn Sie ein Skript in einem Mapping ausführen und ein Eingabedokument fehlt, verwendet die Umwandlung die Beispielquelle. Ist keine Beispielquelle konfiguriert und kein Eingabedokument vorhanden, stoppt die Datenprozessor-Umwandlung und generiert einen schwerwiegenden Fehler.

Beispiel für eine Beispielquelle

Das folgende Beispiel illustriert einen Teil einer lokalen Datei, die Sie als Beispielquelle verwenden können, wenn Sie HL7-Dokumente parsen:

```
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^99999?
```

```
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|A0
```

Hervorhebungen in der Beispielquelle

Der Bereich **Eingabe** der Ansicht **Daten-Viewer** hebt Teile des Beispiel-Quelldokuments hervor.

Die Ansicht **Daten-Viewer** verwendet verschiedene Farben, um die Inhaltsanker der Beispielquelle und die Marker-Anker, die definieren, wo die Umwandlung auf Inhalte trifft, und Wiederholungsgruppen von Ankern hervorzuheben.

Festlegen einer Beispielquelle im IntelliScript-Editor

Wenn Sie ein Skript von der Ansicht **Daten-Viewer** aus ausführen, müssen Sie über eine Beispielquelle für den Haupteingabebort sowie für jeden zusätzlichen Eingabebort verfügen. Legen Sie die Beispielquelle im IntelliScript-Editor fest. Sie können die Beispielquelle auch auswählen, wenn Sie ein Skript in der Datenprozessor-Umwandlung erstellen.

1. Wählen Sie die Komponente aus, für die Sie eine Beispielquelle erstellen möchten, und erweitern Sie sie, um ihre Eigenschaften anzuzeigen.
2. Doppelklicken Sie neben der Eigenschaft **example_source** auf die Auslassungspunkte.
3. Wählen Sie ein Eingabeformat aus. In der folgenden Tabelle werden die Optionen für das Eingabeformat beschrieben:

Option	Beschreibung
LocalFile	Die Eigenschaft _name wird unterhalb der Eigenschaft example_source angezeigt. Doppelklicken Sie auf die Auslassungspunkte, und wählen Sie eine Datei auf dem lokalen Dateisystem aus.
Text	Die Eigenschaft quote wird unterhalb der Eigenschaft example_source angezeigt. Geben Sie einen String ein.
URL	Die Eigenschaft stable_url wird unterhalb der Eigenschaft example_source angezeigt. Geben Sie einen String ein.

Anzeigen von Beispielquellen

Sie können die Beispielquelle von Parsern, Mappern, Serializern oder zusätzlichen Eingabeports im Bereich **Eingabe** der Ansicht **Daten-Viewer** anzeigen.

1. Öffnen Sie im IntelliScript-Editor ein Skript.
2. Legen Sie eine der Komponenten des Skripts als Startkomponente fest.
3. Wählen Sie im IntelliScript-Editor die Komponente aus, die die Beispielquelle enthält, die Sie anzeigen möchten.
4. Klicken Sie in der Ansicht **Daten-Viewer** auf **Mit Editor synchronisieren**.

IntelliScript-Editor

Der IntelliScript-Editor ist ein grafisches Tool, das zur Bearbeitung von Skripten dient. Verwenden Sie den IntelliScript-Editor, um Komponenten zum Skript hinzuzufügen, Komponenteneigenschaften zu konfigurieren und die Startkomponente einzurichten.

Wenn Sie ein Skriptobjekt öffnen, wird der IntelliScript-Editor im Editorbereich im Zentrum der Benutzeroberfläche des Developer-Tools angezeigt. Standardmäßig zeigt der IntelliScript-Editor Skripte im Intelli-Modus (der Skripte in einem erweiterbaren hierarchischen Baumformat anzeigt) oder im Skriptmodus (der Skript als Text anzeigt) an. Sie können ein Skript im Intelli-Modus anzeigen oder bearbeiten. Einige erweiterte Eigenschaften sind standardmäßig ausgeblendet, Sie können sie jedoch anzeigen, indem Sie auf einen grafischen Doppelpfeil in der ersten Zeile der Komponente klicken.

Sie können nur für den Kontext gültige Komponenten einfügen. Sie können eine Komponente ziehen, um sie zu bewegen, oder Sie können Sie mit **STRG+C** und **STRG+V** ausschneiden und einfügen. Sie können mit Mausklicks und den Tasten **STRG** und **UMSCHALT** mehrere Komponenten auswählen.

Wenn Sie den IntelliScript-Editor verwenden, werden in den folgenden Ansichten relevante Informationen angezeigt.

- Daten-Viewer, Eingabebereich. Zeigt die Beispielquelle für die Startkomponente oder die im IntelliScript-Editor ausgewählte Komponente an.
- Daten-Viewer, Ausgabebereich. Zeigt die Ausgabe an, wenn Sie die Datenprozessor-Umwandlung über die **Daten-Viewer**-Ansicht anzeigen.
- Datenprozessor-Ereignisse. Zeigt die Ereignisse an, die bei der Ausführung der Datenprozessor-Umwandlung eintreten. Verwenden Sie die Ansicht **Datenprozessor-Ereignisse** zur Fehlerbehebung.
- Datenprozessor-Skripthilfe. Zeigt für die aktuell im IntelliScript-Editor ausgewählte Komponente oder Eigenschaft relevante Dokumentation an.
- Datenprozessor-Hex-Quelle. Zeigt das Beispiel-Quelldokument in hexadezimaler Form an. Verwenden Sie die Ansicht **Datenprozessor-Hex-Quelle**, um nicht druckbare Zeichen, z. B. Tabstopps, anzuzeigen.

Zum Anzeigen der Quelle eines Skripts, klicken Sie mit der rechten Maustaste in den IntelliScript-Editor, und wählen Sie dann **Skriptmodus** aus. Um in den Intelli-Modus zurückzukehren, klicken Sie mit der rechten Maustaste in den IntelliScript-Editor, und wählen Sie dann **Intelli-Modus** aus.

Validieren eines Skripts

Wenn Sie ein Skript erstellen, können Sie es vor der Ausführung validieren. Beim Validieren eines Skripts überprüft das Developer Tool, ob Fehler vorhanden sind, die eine Komponente daran hindern könnten, Daten erwartungsgemäß zu verarbeiten.

Wählen Sie zum Validieren eines Skripts in der Ansicht **Gliederung** des Developer Tools das Skript aus, klicken Sie mit der rechten Maustaste und klicken Sie auf **Validieren**. Sind Fehler vorhanden, wird die Ansicht **Validierungsprotokoll** mit Fehlern oder Warnungen zu Fehlern angezeigt, die bei der Validierung erkannt wurden.

Wenn Sie in der Ansicht **Validierungsprotokoll** auf einen Fehler doppelklicken, wird die entsprechende Zeile im Skript im IntelliScript-Editor markiert.

Muster-Skripte

Informatica liefert Muster-Skripte als Beispiele für Aufgaben, die Sie mit einem Skript abwickeln können.

Sie finden diese Beispielskripte im folgenden Unterverzeichnis des Installationsordners:

```
\DataTransformation\samples\Projects
```

Zum Anzeigen, Ändern oder Kopieren eines Beispielskripts müssen Sie es zunächst importieren.

In der folgenden Tabelle werden die Beispielskripte beschrieben:

Skriptname	Beschreibung
Alternatives	Demonstriert Verzweigungen und den Alternatives -Anker.
AppendListItems	Verkettet Strings in einem Datenbehälter mit mehreren Instanzen und demonstriert die Aktion AppendListItems .
CalculateValue	Führt eine komplexe numerische Berechnung durch und demonstriert die Aktion CalculateValue .
CombineValues	Verkettet Strings und demonstriert die Aktionen CombineValues und DumpValue .
Content	Demonstriert den Content -Anker und das Suchen im Quelldokument mithilfe eines bestimmten Strings, mithilfe der Berechnung eines Offsets ab dem letzten Anker und durch Suchen nach einem Attribut in einem "Name=Value"-Paar.
CopyValue	Kopiert ein ganzes komplexes XML-Element mit der Mapping -Aktion.
DelimitedSections	Demonstriert den DelimitedSections -Anker in einem Parser.
DocumentOrder	Demonstriert Verzweigungen und den Alternatives -Anker, wobei die Option selector auf "DocumentOrder" gesetzt ist.
Dynamic_And_RepeatingGroup	Fährt wiederholt über die Zeilen eines Dokuments und demonstriert den RepeatingGroup -Anker. Liest Daten von einem anderen Ort auf der Grundlage von Inhalten im aktuellen Suchbereich in das Dokument ein.
EmbeddedParser	Verwendet einen eingebetteten sekundären Parser, um den Inhalt des Haupt-Parsers zu parsen und demonstriert den EmbeddedParser -Anker.
EnsureCondition	Evaluiert einen boolschen JavaScript-Ausdruck, um Alternativen auszuwählen und demonstriert die Aktion EnsureCondition .
ManualSerializer	Demonstriert einen benutzerdefinierten Serializer.
Marker	Demonstriert Marker -Anker, die Optionen "TextSearch", "OffsetSearch", "TypeSearch" und "PatternSearch" verwenden.
Marking_Modus	Demonstriert mehrere Methoden zum Konfigurieren von Marker -Ankern.
NonMarker	Demonstriert einen Parser, der nur Content -Anker verwendet und das Eingabedokument rückwärts durchsucht.
Muster	Demonstriert die Extraktion von Daten, die einer Einschränkung entsprechen, die im Schema definiert wurde.

Skriptname	Beschreibung
persistent_search	Demonstriert die Eigenschaft on_partial_match einer Group und die Eigenschaft adjacent eines Markers .
ResetListVariable	Setzt eine Listenvariable mithilfe eines targetLocator zurück.
RunSerializer	Demonstriert einen Parser, der einen sekundären Serializer aufruft.
HL7	Konvertiert eine HL7-Datei in XML.
TabDelimited	Konvertiert eine HL7-Datei mit Tabulator-Trennzeichen in XML.
Splitter	Teilt eine Datei in zwei Dateien und demonstriert die Aktion WriteValue .
TransformByParser	Verwendet einen Parser, um bestimmten Text in Zeilenschaltungen zu ändern und demonstriert die Aktion TransformByParser .
Transformers_Example	Demonstriert den Content -Anker, wobei die Eigenschaft value auf LearnByExample gesetzt ist.

Beispielskript importieren

Importieren Sie ein Beispielskript, um es anzuzeigen oder Teile in ein anderes Skript zu importieren.

1. Klicken Sie auf **Datei > Importieren**.
Das Dialogfeld **Importieren** wird eingeblendet.
2. Wählen Sie **Informatica > DT-Dienst importieren** aus und klicken Sie auf **Weiter**.
Die Seite **DT-Dienst importieren** erscheint.
3. Klicken Sie neben dem Feld **Dienstdatei** auf die Schaltfläche **Durchsuchen** und navigieren Sie zur CMW-Datei für den Dienst.
4. Klicken Sie auf **Fertig stellen**.
Das Beispielskript wird in einer Datenprozessor-Umwandlung angezeigt.

KAPITEL 11

Parser

Dieses Kapitel umfasst die folgenden Themen:

- [Parser - Überblick, 150](#)
- [Plattformunabhängige Parser, 150](#)
- [Die Komponente Parser: Referenz, 151](#)

Parser - Überblick

Parser sind Skriptkomponenten, die Quelldokumente in einem beliebigen Format einlesen.

Parser geben immer XML aus. Die Eingabe kann ein beliebiges Format aufweisen, zum Beispiel Text, HTML, Word, PDF oder HL7. Die Eingabe kann ein XML-Dokument sein, das der Parser als Zeichenfolgendaten verarbeitet.

Plattformunabhängige Parser

Parser-Skripte laufen unter Microsoft Windows und UNIX. Die meisten Parser-Merkmale funktionieren auf beiden Plattformen gleich gut.

Es gibt jedoch einige wenige Ausnahmen. Wenn Sie einen Parser unter Windows und UNIX ausführen möchten, sollten Sie die folgenden Tipps bezüglich der Plattformunabhängigkeit berücksichtigen.

Zeilenvorschub-Marker

Vermeiden Sie die Verwendung von **Marker**-Ankern, die nach einem Zeilenvorschubzeichen gefolgt von einem Wagenrücklaufzeichen (`\n\r`) suchen. Diese Zeichenfolge wird zwar allgemein in Windows verwendet, aber in der Regel nicht in UNIX.

Konfigurieren Sie stattdessen einen **Marker** mit integrierter **Newline-Search**-Komponente, die sowohl nach der Folge `\n\r` als auch nach den Einzelzeichen `\n` und `\r` sucht.

Dateipfade

Verwenden Sie relative Dateipfade, keine absoluten. Denken Sie daran, dass UNIX bei Dateipfaden zwischen Groß- und Kleinschreibung unterscheidet.

Die Komponente Parser: Referenz

Die **Parser**-Komponente wandelt ein Quelldokument in XML um.

Parser

Ein Parser liest ein Quelldokument in jedem Format. Sie können untergeordnete Komponenten hinzufügen, um Umwandlungen der Daten vorzunehmen.

Definieren Sie Parser auf der globalen Ebene des Skripts. Legen Sie einen Haupt-Parser als Startkomponente fest. Rufen Sie einen sekundären Parser mit der Aktion **RunParser** auf. Weitere Informationen hierzu finden Sie unter ["RunParser" auf Seite 328](#).

Die Eigenschaften des **Parser** erscheinen über der Zeile **contains**. Unter der Zeile können Sie untergeordnete Komponenten wie Anker und Aktionen einfügen.

Die nachstehende Tabelle beschreibt die Eigenschaften der Komponente **Parser**:

Eigenschaft	Beschreibung
example_source	Definiert ein Beispielquelltdokument zur Verarbeitung des Entwicklungsumfelds. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none">- Leer. Das Developer-Tool fordert Sie zur Eingabe eines Quelldokuments auf, wenn Sie den Parser ausführen.- InputPort. Definiert einen Eingabeport.- LocalFile. Definiert eine Datei im lokalen Dateisystem- Text. Definiert einen String.- URL. Definiert eine URL. Standardwert ist "Leer". Hinweis: Wenn die Eigenschaft sources_to_extract eingestellt ist, wird die Eigenschaft example_values in der Entwurfsumgebung ignoriert.
example_values	Definiert simulierte Werte, die von einer anderen Umwandlung an den Parser übergeben werden könnten. Verwenden Sie diese Eigenschaft, um einen Parser zu bezeichnen, der von einem anderen Parser aufgerufen wird. Ein Parser verwendet die Eigenschaft example_values nur für die Verarbeitung der Beispielquelle. Beim Parsen eines Quelldokuments wird die Eigenschaft ignoriert. In den geschachtelten ExampleValue -Komponenten geben Sie die Datenbehälter, die der aufrufende Parser an diesen Parser übergibt, sowie deren simulierte Werte an.
ExampleValue	Definiert einen Beispielwert unter der Eigenschaft example_values .
format	Definiert das Format des Quelldokuments. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none">- BinaryFormat- CustomFormat- HtmlFormat- Rtf Format- TextFormat- XmlFormat Standardwert ist CustomFormat. Weitere Informationen hierzu finden Sie unter "Formatkomponenten: Referenz" auf Seite 179 .
name	Eine beschreibende Kennzeichnung für die Komponente. Diese Kennzeichnung wird in der Protokolldatei sowie der Ereignisansicht angezeigt. Ermitteln Sie anhand der Eigenschaft name , welche Komponente das Ereignis verursacht hat.

Eigenschaft	Beschreibung
no_initial_phase	<p>Legt fest, ob das Skript nach geschachtelten Ankern in der Hauptphase sucht. Sie können eine der folgenden Optionen auswählen:</p> <ul style="list-style-type: none"> - Gelöscht. Es erfolgt eine Suche nach geschachtelten Ankern entsprechend ihren individuellen Eigenschaften. - Ausgewählt. Es erfolgt eine Suche nach geschachtelten Ankern in der Hauptphase. Die Standardoption lautet „Gelöscht“.
Benachrichtigungen	<p>Definiert eine Liste von NotificationHandler-Komponenten, die der Parser mit Benachrichtigungen durchführt, die von geschachtelten Komponenten ausgelöst werden. Weitere Informationen hierzu finden Sie unter "Benachrichtigungen" auf Seite 430.</p>
on_fail	<p>Die Aktion, die beim Ausfall der Komponente durchgeführt wird. Sie können eine der folgenden Optionen auswählen:</p> <ul style="list-style-type: none"> - Gelöscht. Es wird keine Aktion ausgeführt. - CustomLog. Es wird in das Benutzerprotokoll geschrieben. - LogError. Eine Fehlermeldung wird in das Engine-Protokoll geschrieben. - LogInfo. Eine Informationsmeldung wird in das Engine-Protokoll geschrieben. - LogWarning. Eine Warnmeldung wird in das Engine-Protokoll geschrieben. - NotifyFailure. Eine Mitteilung wird gesendet. <p>Die Standardoption lautet „Gelöscht“. Weitere Informationen über die Behandlung von Komponentenfehlern finden Sie in "Fehlerbehandlung" auf Seite 410.</p>
reject_recurring_pages	<p>Legt fest, wie oft der Parser dieselbe Seite parst. Sie können eine der folgenden Optionen auswählen:</p> <ul style="list-style-type: none"> - Ausgewählt. Der Parser parst eine Seite nur einmal. - Gelöscht. Der Parser parst die Seite jedes Mal, wenn er einem Link zu der Seite folgt. Verwenden Sie reject_recurring_pages, wenn eine Website viele Links zu derselben Seite enthält. <p>Hinweis: Die Aktion ResetVisitedPages setzt die Verlaufsliste zurück und ermöglicht es dem Parser, eine Seite erneut zu verarbeiten – auch dann, wenn reject_recurring_pages ausgewählt wurde.</p>
remark	<p>Ein benutzerdefinierter Kommentar, der den Zweck oder die Aktion der Komponente beschreibt.</p>
Serialisierungsmodus	<p>Diese Eigenschaft definiert, wie das Skript mit den Teilen der Beispielquelle umgehen soll, die vom Parser nicht in XML umgewandelt werden, wenn Sie einen Serializer aus einem Parser erstellen. Weitere Informationen hierzu finden Sie unter "Steuern der Arbeitsweise des Befehls „Serializer erstellen“" auf Seite 340.</p> <p>Sie können eine der folgenden Optionen auswählen:</p> <ul style="list-style-type: none"> - Vollständig. Bewirkt, dass der Befehl Serializer erstellen den nicht in XML umgewandelten Text in die Konfiguration des Serializers kopiert. - Outline. Bewirkt, dass der Befehl Serializer erstellen nur die Delimiter des nicht in XML umgewandelten Texts in die Konfiguration des Serializers kopiert. Wenn Outline ausgewählt ist, können Sie die Eigenschaft use_markers einstellen.
source	<p>Definiert eine Sequenz von Datenbehältern für die Eingabe in den Parser. Jeder Datenbehälter wird durch eine der folgenden Eigenschaften ausgewiesen:</p> <ul style="list-style-type: none"> - Locator. Identifiziert einen Einzel- oder Mehrfachinstanz-Datenbehälter. Bei Mehrfachinstanz-Datenbehältern greift jede Iteration auf eine neue Instanz zu. - LocatorByKey. Identifiziert einen Einzelinstanz-Datenbehälter nach Schlüssel. - LocatorByOccurrence. Identifiziert einen Mehrinstanz-Datenbehälter nach Sequenznummer. <p>Legen Sie in einem sekundären Parser Parser > source > Locator > data_holder mit dem Datenbehälter fest, der im zugehörigen AdditionalInputPort > data_holder definiert ist. Weitere Informationen hierzu finden Sie unter "Eigenschaft „source“" auf Seite 375.</p>

Eigenschaft	Beschreibung
sources_to_extract	<p>Definiert eine hartcodierte Liste von Quelldokumenten, die der Parser verarbeitet. Sie können eine der folgenden Optionen auswählen:</p> <ul style="list-style-type: none"> - DocList. Definiert eine Liste von LocalFile-, Text- und URL-Komponenten. - Leer. Der Parser verarbeitet die Komponente example_source. - FileSearch. Definiert einen Ordner im lokalen Dateisystem und einem Datennamenfilter. - InputPort. Definiert einen Eingabeport. Verwenden Sie diese Option nicht. - LocalFile. Definiert eine Datei im lokalen Dateisystem - Text. Definiert einen String. - URL. Definiert eine URL. <p>Standardwert ist "Leer".</p> <p>Hinweis: Verwenden Sie die Eigenschaft sources_to_extract nur in der Entwurfsumgebung.</p>
target	<p>Definiert eine Sequenz von Datenbehältern für die Ausgabe aus dem Parser. Wenn der Datenbehälter noch nicht vorhanden ist, wird er vom Parser erstellt. Jeder Datenbehälter wird durch eine der folgenden Eigenschaften ausgewiesen:</p> <ul style="list-style-type: none"> - Locator. Identifiziert einen Einzel- oder Mehrfachinstanz-Datenbehälter. Bei Mehrfachinstanz-Datenbehältern erstellt jede Iteration eine neue Instanz. - LocatorByKey. Identifiziert einen Einzelinstanz-Datenbehälter nach Schlüssel. - LocatorByOccurence. Identifiziert einen Mehrinstanz-Datenbehälter nach Sequenznummer. <p>Verwenden Sie die Eigenschaft target, wenn die Ausgabe des Parsers von einer anderen Komponente benutzt wird. Weitere Informationen hierzu finden Sie unter "Eigenschaft „target“" auf Seite 378.</p>
use_markers	<p>Legt fest, ob der Befehl Create Serializer den Inhalt der Marker-Anker kopiert, aber nur die Delimiter anderer Nicht-XML-Texte. use_markers ist eine Option unter der Eigenschaft serialization_mode, wenn outline ausgewählt ist. Standardwert ist "Ausgewählt".</p>

KAPITEL 12

Skriptports

Dieses Kapitel umfasst die folgenden Themen:

- [Skriptports - Überblick, 154](#)
- [Skript-Port-Komponente - Referenz, 154](#)

Skriptports - Überblick

Ein Skriptport gibt die Ein- oder Ausgabe eines Skripts an, beispielsweise ein Quell- oder ein Ausgabedokument.

In einer **Parser**-Komponente beispielsweise sind die Werte der Eigenschaften **example_source** und **sources_to_extract** die Eingabeports.

In einigen Komponenten sind die Skriptports implizit definiert. Beispielsweise ist die Standard-Ausgabedatei eines Parsers die Datei `output.xml`. Sie brauchen keinen Ausgabeport zu definieren, der auf die Datei `output.xml` verweist.

Standardmäßig hat jedes Skript einen Eingabe- und einen Ausgabeport. Sie können zusätzliche Eingabe- und Ausgabeports definieren. Wenn Sie zusätzliche Eingabe- oder Ausgabeports in einem Skript erstellen, fügt das Developer-Tool in der Datenprozessor-Umwandlung zusätzliche Ports hinzu.

Skript-Port-Komponente - Referenz

Ein Skriptport ist eine Komponente, die eine Ein- oder Ausgabe einer Umwandlung festlegt, wie beispielsweise ein Quell- oder ein Ausgabedokument.

AdditionalInputPort

Der **AdditionalInputPort**-Port definiert einen zusätzlichen Eingabeport.

Die nachstehende Tabelle beschreibt die Eigenschaften des **AdditionalInputPort**-Ports:

Eigenschaft	Beschreibung
code_page	Legt die Eingabecodierung für den Port fest. Ist kein Wert festgelegt, verwendet der AdditionalInputPort die in den Einstellungen der Datenprozessor-Umwandlung definierte Eingabecodierung. Standardwert ist "Leer".
data_holder	Definiert einen Datenbehälter, wo der Port den Inhalt des Eingabedokuments speichert. Verwenden Sie denselben Datenbehälter in der Parser > source > Locator > data_holder -Eigenschaft des zugeordneten zweiten Parsers, Mappers oder Serializers.
disabled	Legt fest, ob die Komponente und alle zugehörigen untergeordneten Komponenten vom Skript ignoriert werden. Verwenden Sie diese Eigenschaft zum Testen, Debuggen und Ändern eines Skripts. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none"> - Ausgewählt. Das Skript ignoriert die Komponente. - Gelöscht. Das Skript wendet die Komponente an. Standardmäßig ist die Eigenschaft deaktiviert.
encode_as_xml	Legt fest, ob Sonderzeichen in XML-Entitäten umgewandelt werden. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none"> - Ausgewählt. Sonderzeichen werden in XML-Entitäten umgewandelt. - Gelöscht. Sonderzeichen werden nicht umgewandelt. Die Eigenschaft encode_as_xml ist eine untergeordnete Eigenschaft der Eigenschaft input_encoding , wenn sie auf PortEncoding gesetzt ist. Standardwert ist "Gelöscht".
example_source	Definiert den Ort einer Quelle, die beim Testen zu verarbeiten ist. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none"> - LocalFile. Definiert eine Datei auf dem lokalen Computer. - Text. Definiert einen String. - URL. Definiert die URL einer Webseite. Vorsicht: Definieren Sie keinen Dokumentprozessor unter der Eigenschaft AdditionalInputPort > example_source > pre_processor . Definieren Sie ihn unter AdditionalInputPort > pre_processor .
input_encoding	Definiert die Codierung der Eingabe.
PortEncoding	Definiert benutzerdefinierte Einstellungen für die weitere Eingabe. Die Eigenschaft PortEncoding hat folgende Optionen: <ul style="list-style-type: none"> - code_page - encode_as_xml
pre_processor	Definiert den Namen eines Dokumentprozessors, der auf die Eingabe vor dem unter RunParser > pre_processor definierten Dokumentprozessor angewandt wird. Weitere Informationen hierzu finden Sie unter "Dokumentprozessorkomponenten: Referenz" auf Seite 164 . Vorsicht: Definieren Sie keinen Dokumentprozessor unter der Eigenschaft AdditionalInputPort > example_source > pre_processor . Definieren Sie ihn unter AdditionalInputPort > pre_processor .

Definieren Sie den Port **AdditionalInputPort** auf der globalen Ebene des Skripts und weisen Sie ihm einen Namen zu.

Beispiel für AdditionalInputPort

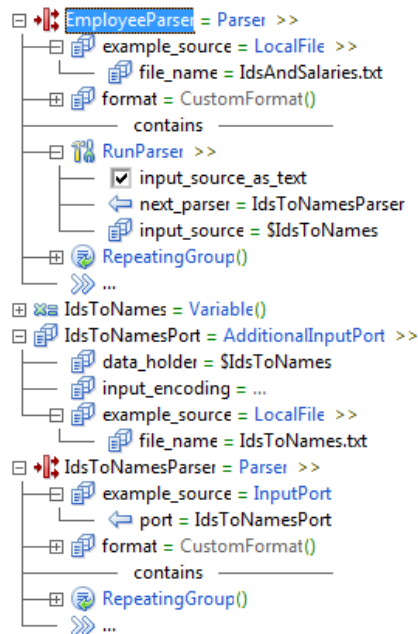
Angenommen, Sie haben zwei Textdateien:

- `IdsAndSalaries.txt` ist eine Tabelle der Mitarbeiter-IDs und -Gehälter.
- `IdsAndNames.txt` ist eine Tabelle der Mitarbeiter-IDs und -Namen.

Sie möchten diese Dateien gemeinsam parsen und eine XML-Ausgabedatei erstellen, die die Namen und Gehälter der Mitarbeiter enthält. Die Umwandlung kann auf folgende Weise konfiguriert werden:

- Der Hauptparser mit Namen `EmployeeParser` verarbeitet `IdsAndSalaries.txt`.
- Der Hauptparser aktiviert einen zweiten Parser namens `IdsToNamesParser`, der `IdsAndNames.txt` verarbeitet und die Ergebnisse in einer XML-Tabelle speichert.
- Der Hauptparser verwendet einen `LookupTransformer`, um die IDs in Namen zu konvertieren. Die Nachschlagetabelle ist die Ausgabe des zweiten Parsers.

Die folgende Abbildung zeigt ein Beispiel eines Skripts mit einem sekundären Parser an, der auf einen `AdditionalInputPort` verweist, um die Datei `IdsAndNames.txt` abzurufen:



AdditionalOutputPort

Der Port **AdditionalOutputPort** definiert einen zusätzlichen Ausgabeport. Verwenden Sie diese Komponente, um die Ausgabe an mehreren Standorten oder in mehreren Dokumenten zu definieren.

Die nachstehende Tabelle beschreibt die Eigenschaften des Ports **AdditionalOutputPort**:

Eigenschaft	Beschreibung
add_BOM_prefix	Fügt der Ausgabe ein Byte-Order-Markierungs-Präfix (BOM) hinzu. Die Art des BOM-Präfixes wird von der Ausgabecodierung bestimmt, die in der Eigenschaft output_encoding definiert wurde. Standardwert ist "Gelöscht".
code_page	Definiert das Codierungs -Attribut der zusätzlichen Ausgabe. Wenn diese Eigenschaft nicht festgelegt wird, wird die zusätzliche Ausgabe mit der Ausgabecodierung generiert, die in den Umwandlungseinstellungen des Datenprozessors definiert wurde. Die Eigenschaft code_page ist der Eigenschaft output_encoding untergeordnet, wenn sie auf PortEncoding gesetzt ist. Standardwert ist "Gelöscht".

Eigenschaft	Beschreibung
disabled	<p>Legt fest, ob die Komponente und alle zugehörigen untergeordneten Komponenten vom Skript ignoriert werden. Verwenden Sie diese Eigenschaft zum Testen, Debuggen und Ändern eines Skripts. Sie können eine der folgenden Optionen auswählen:</p> <ul style="list-style-type: none"> - Ausgewählt. Das Skript ignoriert die Komponente. - Gelöscht. Das Skript wendet die Komponente an. <p>Standardmäßig ist die Eigenschaft deaktiviert.</p>
encode_as_xml	<p>Legt fest, ob Sonderzeichen in XML-Entitäten umgewandelt werden. Sie können eine der folgenden Optionen auswählen:</p> <ul style="list-style-type: none"> - Ausgewählt. Sonderzeichen werden in XML-Entitäten umgewandelt. - Gelöscht. Sonderzeichen werden nicht umgewandelt. <p>Die Eigenschaft encode_as_xml ist der Eigenschaft output_encoding untergeordnet, wenn sie auf PortEncoding gesetzt ist. Standardwert ist "Gelöscht".</p>
file_extension	<p>Definiert die Dateinamenerweiterung für die zusätzliche Ausgabedatei in der Entwurfsumgebung. Der Name der Datei ist der Name, der der Komponente AdditionalOutputPort zugewiesen wurde. Diese Einstellung hat keinen Einfluss auf die Produktionsumgebung. Standardwert ist .xml.</p>
other_properties	<p>Definiert die Codierungseigenschaften, wenn XmlHeader festgelegt wurde. Sie können eine der folgenden Optionen auswählen:</p> <ul style="list-style-type: none"> - XmlHeader. Definiert den XML-Header. Die Eigenschaft XmlHeader hat folgende Optionen: <ul style="list-style-type: none"> - add_BOM_prefix - process_instruction - process_instruction_string - root_element - xml_version - XSLT_stylesheet_name - Gelöscht. Ausgabeigenschaften werden von den Datenprozessor-Umwandlungseinstellungen bestimmt. <p>Standardwert ist "Gelöscht".</p>
output_encoding	<p>Definiert die Codierungseigenschaften, wenn PortEncoding festgelegt wurde. Sie können eine der folgenden Optionen auswählen:</p> <ul style="list-style-type: none"> - PortEncoding. Die zusätzliche Ausgabe hat benutzerdefinierte Einstellungen für code_page und encode_as_xml. - Gelöscht. Die Datenprozessor-Umwandlungseinstellungen steuern Ausgabecodierung und Umwandlung der XML-Entitäten. <p>Standardwert ist "Gelöscht".</p>
PortEncoding	<p>Definiert die benutzerdefinierten Einstellungen für die zusätzliche Ausgabe. Sie können eine der folgenden Optionen auswählen:</p> <ul style="list-style-type: none"> - code_page - encode_as_xml
process_instruction	<p>Definiert eine Verarbeitungsanweisung in der XML-Ausgabedatei. Sie können eine der folgenden Optionen auswählen:</p> <ul style="list-style-type: none"> - Keine. Schreibt keine Verarbeitungsanweisung für die XML-Ausgabe. - UseOutputCodePage. Gibt die Codepage aus, die in der Eigenschaft output_encoding definiert wurde. - FreeEncodingString. Gibt den String aus, der in der Eigenschaft process_instruction_string definiert wurde. <p>Standardwert ist UseOutputCodePage.</p>
process_instruction_string	<p>Definiert eine benutzerdefinierte Verarbeitungsanweisung. Die Eigenschaft process_instruction_string hat nur dann eine Auswirkung, wenn die Eigenschaft process_instruction auf FreeEncodingString gesetzt ist.</p>

Eigenschaft	Beschreibung
root_element	Definiert den Namen und das Stammelement, das die gesamte Ausgabe umgibt.
xml_version	Definiert das Versions-Attribut der Verarbeitungsanweisung. Standard ist 1.0.
XSLT_stylesheet_name	Definiert eine XSLT-Formatvorlage für die Verarbeitungsanweisung.

Definieren eines zusätzlichen Ausgabeports

1. Geben Sie auf der globalen Ebene des Skripts eine **AdditionalOutputPort**-Komponente ein und weisen Sie ihr einen Namen zu.
2. Fügen Sie geschachtelt unter der Startkomponente der Datenprozessor-Umwandlung eine **WriteValue**-Aktion ein, legen Sie die Eigenschaft **output** als **OutputPort** fest und geben Sie **port** mit dem Namen des zusätzlichen Ausgabeports an.
3. Wählen Sie in den Einstellungen für die Datenprozessor-Umwandlung die Option **Ausgabesteuerung** aus und aktivieren Sie **Automatische Ausgabe deaktivieren**.

Dateiname der zusätzlichen Ausgabe

Wenn Sie die Umwandlung im Developer Tool durchführen, legt das System einen Dateinamen für den zusätzlichen Ausgabeport fest und speichert die Datei im Ergebnisordner des Projekts. Wenn der Port beispielsweise **MyOutputPort** heißt, könnte der Dateiname `output_MyOutputPort.xml` lauten.

So ermitteln Sie den Dateinamen:

1. Klicken Sie auf **Ausführen > Ausführen**.
2. Klicken Sie auf **Details**, um die Tabelle **I/O-Ports** anzuzeigen.
In der Tabelle wird der Name jedes einzelnen **AdditionalOutputPort** und die dazugehörige Ausgabedatei angezeigt.

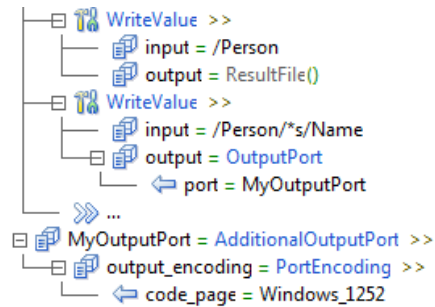
Wenn Sie die Datenumwandlung als -Dienst bereitstellen, kann eine Anwendung, die den Dienst ausführt, den zusätzlichen Ausgabe-Speicherort als einen Parameter weitergeben. Bei diesem Speicherort kann es sich zum Beispiel um einen Puffer handeln.

Beispiel für AdditionalOutputPort

Ein Parser generiert die folgende XML-Struktur:

```
<Person gender="M">
  <Name>
    <First>Ron</First>
    <Last>Lehrer</Last>
  </Name>
  <Id>547329876</Id>
  <Age>27</Age>
</Person>
```

Die folgende Abbildung zeigt einen Parser an, der zwei `WriteValue`-Aktionen zum Generieren der Ausgabe verwendet.



Die erste `WriteValue`-Aktion schreibt das gesamte Element `<Person>` in die Ergebnisdatei.

```
<Person gender="M">
  <Name>
    <First>Ron</First>
    <Last>Lehrer</Last>
  </Name>
  <Id>547329876</Id>
  <Age>27</Age>
</Person>
```

Die zweite `WriteValue`-Aktion verweist auf einen `AdditionalOutputPort`, um das geschachtelte Element `<Name>` in eine andere Datei zu schreiben.

```
<Name>
  <First>Ron</First>
  <Last>Lehrer</Last>
</Name>
```

DocList

Der Port **DocList** definiert eine Liste der folgenden Typen von Eingabeports:

- LocalFile
- Text
- URL

Die nachstehende Tabelle beschreibt die Eigenschaften des Ports **DocList**:

Eigenschaft	Beschreibung
disabled	Legt fest, ob die Komponente und alle zugehörigen untergeordneten Komponenten vom Skript ignoriert werden. Verwenden Sie diese Eigenschaft zum Testen, Debuggen und Ändern eines Skripts. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none"> - Ausgewählt. Das Skript ignoriert die Komponente. - Gelöscht. Das Skript wendet die Komponente an. Standardmäßig ist die Eigenschaft deaktiviert.
pre_processor	Definiert den Namen des Präprozessors, der auf die Eingabedateien anzuwenden ist. Weitere Informationen hierzu finden Sie unter "Dokumentprozessorkomponenten: Referenz" auf Seite 164 .

FileSearch

Der Port **FileSearch** definiert Eingabedateien auf einem Computer im lokalen Netzwerk. Verwenden Sie den Port **FileSearch** in der Eigenschaft `sources_to_extract` eines **Parser**.

Die nachstehende Tabelle beschreibt die Eigenschaften der Aktion **FileSearch**:

Eigenschaft	Beschreibung
directory	Definiert einen Ordner, der die Eingabedateien enthält.
disabled	Legt fest, ob die Komponente und alle zugehörigen untergeordneten Komponenten vom Skript ignoriert werden. Verwenden Sie diese Eigenschaft zum Testen, Debuggen und Ändern eines Skripts. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none">- Ausgewählt. Das Skript ignoriert die Komponente.- Gelöscht. Das Skript wendet die Komponente an. Standardmäßig ist die Eigenschaft deaktiviert.
pre_processor	Definiert den Namen eines Dokumentprozessors, um die Eingabedateien anzuwenden. Weitere Informationen hierzu finden Sie unter "Dokumentprozessorkomponenten: Referenz" auf Seite 164 .
recursive	Legt fest, ob die Eingabedateien in Unterordnern des angegebenen Ordners vorkommen können. Standardwert ist "Gelöscht".
wildcard	Definiert ein Kriterium für das Filtern der Dateien im angegebenen Ordner. Verwenden Sie * als Platzhalter. Beispielsweise werden mit *.txt alle TXT-Dateien gefunden. Standardwert ist *.*.

InputPort

Der Port **InputPort** definiert einen benannten Eingabeport, der mit der Komponente **AdditionalInputPort** definiert ist.

Die nachstehende Tabelle beschreibt die Eigenschaften des Ports **InputPort**:

Eigenschaft	Beschreibung
Eingabe	Definiert den Namen der Komponente AdditionalInputPort , die die Eingabe definiert.

LocalFile

Der **LocalFile**-Port definiert eine Datei im lokalen Netzwerk.

Die nachstehende Tabelle beschreibt die Eigenschaften des Ports **LocalFile**:

Eigenschaft	Beschreibung
file_name	Definiert den Pfad und den Dateinamen einer Datei im lokalen Netzwerk.
pre_processor	Definiert den Namen eines Dokumentprozessors, der auf die Datei anzuwenden ist. Weitere Informationen hierzu finden Sie unter "Dokumentprozessorkomponenten: Referenz" auf Seite 164 .
simulated_url	Definiert eine URL, die der Datei zuzuweisen ist. Diese Eigenschaft veranlasst den Parser, die Datei so zu behandeln, als befände sie sich auf einem Webserver. Falls die Datei relative Links enthält, löst der Parser die Links relativ zum angegebenen URL auf. Bei dem in der URL angegebenen Hostnamen wird nicht zwischen Groß- und Kleinschreibung unterschieden.

OutputPort

Der Port **OutputPort** definiert einen benannten Ausgabeport, der mit der Komponente **AdditionalOutputPort** definiert ist. Sie können einen **OutputPort**-Port in einer **WriteValue**-Aktion verwenden.

Die nachstehende Tabelle beschreibt die Eigenschaften des Ports **OutputPort**:

Eigenschaft	Beschreibung
Port	Legt den Namen des AdditionalOutputPort fest.

Text

Der Port **Text** definiert einen Textstring, der als Eingabe einer Umwandlung verwendet wird.

In der folgenden Tabelle werden die Eigenschaften des Ports **Text** beschrieben:

Eigenschaft	Beschreibung
pre_processor	Definiert den Namen eines Dokumentprozessors, der auf den String anzuwenden ist. Weitere Informationen hierzu finden Sie unter "Dokumentprozessorkomponenten: Referenz" auf Seite 164 .
quote	Definiert einen Textstring.
simulated_url	Definiert eine URL, die dem String zuzuweisen ist. Diese Eigenschaft weist den Parser an, die Zeichenfolge so zu behandeln, als befände sie sich auf einem Webserver. Falls die Zeichenfolge relative Links enthält, löst der Parser diese relativ zur angegebenen URL auf.
size	Definiert eine statische Größe für den Textpuffer. Verwenden Sie die Eigenschaft size mit binären Quellen. Der Standardwert „-1“ bedeutet, dass die Größe des Puffers dynamisch festgelegt wird.

URL

Der **URL**-Port definiert die URL eines Dokuments, das auf einem Webserver zur Verfügung steht.

In der folgenden Tabelle werden die Eigenschaften des URL-Ports beschrieben:

Eigenschaft	Beschreibung
post_data	Definiert die Daten, die von der Umwandlung an die URL gesendet werden.
pre_processor	Definiert den Namen eines Dokumentprozessors, der auf die Dateien anzuwenden ist.
retries	Definiert die Anzahl an erneuten Versuchen, die der Parser ausführt, bevor er einen Fehler meldet. Der Standardwert ist 0.
seconds_to_wait	Definiert den Zeitraum in Sekunden, der vor einem erneuten Versuch gewartet werden soll. Der Standardwert ist 60.
stable_url	Definiert eine URL-Adresse, die ein Eingabedokument enthält.

Hinweis: Diese Komponente wird aus Kompatibilitätsgründen mit Projekten bereitgestellt, die in früheren Data Transformation-Versionen erstellt wurden. Sie wird schrittweise aus dem Data Transformation-System entfernt. Verwenden Sie sie nicht, wenn Sie Umwandlungen entwickeln. Beachten Sie, dass eine Datenumwandlung eine HTTPS-URL in einer Linux-Umgebung nicht verarbeiten kann.

KAPITEL 13

Dokumentprozessoren

Dieses Kapitel umfasst die folgenden Themen:

- [Überblick über Dokumentprozessoren, 163](#)
- [Definieren eines Dokumentprozessors, 163](#)
- [Dokumentprozessorkomponenten: Referenz, 164](#)
- [TextML-XML-Schema, 173](#)
- [PdfToTxt_4-Editor zur Konfiguration von Tabellen, 174](#)

Überblick über Dokumentprozessoren

Dokumentprozessoren sind Komponenten, die das Format eines ganzen Dokuments für die Verarbeitung in ein anderes Format umwandeln.

Sie können einen Dokumentprozessor als Präprozessor einsetzen, um vor einer Umwandlung das Format eines Quelldokuments zu konvertieren. Liegt beispielsweise das Quelldokument eines Parsers im PDF-Format vor, können Sie den Prozessor **PdfToTxt_4** anwenden. Dieser wandelt das Quelldokument in Text um, der wesentlich einfach zu parsen ist als das binäre PDF-Format.

Verwechseln Sie Dokumentprozessoren nicht mit Formatpräprozessoren. Nähere Informationen zu Formatpräprozessoren finden Sie unter ["Übersicht über Formate" auf Seite 178](#).

Definieren eines Dokumentprozessors

Sie können das Quelldokument mit jedem Dokumentprozessor vorverarbeiten.

1. Weisen Sie die Eigenschaft **example_source** der Umwandlung zu. Der Wert von **example_source** ist ein Eingabeport, wie etwa **LocalFile** oder **Text**.
2. Weisen Sie die Eigenschaft **pre_processor** des Eingabeports zu.

Das Skript wendet den von Ihnen unter **example_source** definierten Prozessor auf alle Quellen an, für die die Umwandlung ausgeführt wird.

Hinweis: Sie können auch einen Präprozessor in der Eigenschaft **sources_to_extract** eines Parsers definieren. Der dort definierte Prozessor wird nur auf Quelldokumente angewendet, die in **sources_to_extract** definiert sind, nicht jedoch auf andere vom Parser verarbeitete Dokumente.

Anzeigen der Ausgabe von Dokumentprozessoren

Wenn Sie der Beispielquelle einen Dokumentprozessor zuweisen, wird im Beispielbereich der **Daten-Viewer**-Ansicht die Prozessorausgabe angezeigt.

Dokumentprozessorkomponenten: Referenz

Dokumentprozessoren wandeln ein komplettes Dokument von einem Format in ein anderes um, bevor es von einem Parser, Mapper oder Serializer verarbeitet wird.

AsnToXml

Der **AsnToXml**-Dokumentprozessor wandelt eine binäre ASN.1-Datei in XML um.

Die nachstehende Tabelle beschreibt die Eigenschaften des Dokumentprozessors **AsnToXml**:

Eigenschaft	Beschreibung
asn_file	Definiert eine ASN.1-Spezifikationsdatei.
Header	Definiert einen Header, der von der XML ausgeschlossen wird. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none">- NewlineSearch. Der Header ist ein Zeilenvorschub.- OffsetSearch. Der Header ist durch die Zeichenanzahl ab Dateibeginn definiert.- PatternSearch. Der Header wird durch einen regulären Ausdruck definiert.- TextSearch. Der Header ist durch einen expliziten String oder einen String definiert, der dynamisch aus dem Quelldokument abgerufen wird.
no_constraints	Legt fest, ob eine ASN-Datei mit Einschränkungen verarbeitet wird. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none">- Wahr. Die ASN-Datei wird ohne Einschränkungen verarbeitet.- Falsch. Die ASN-Datei wird mit Einschränkungen verarbeitet. Standardwert ist "Falsch".
pdu_type	Definiert den PDU-Typ. Verwenden Sie diese Eigenschaft, um eine Zweideutigkeit zu klären.
process_first_message	Legt fest, ob die gesamte CDR-Datei verarbeitet wird. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none">- Wahr. Es wird nur der erste Datensatz verarbeitet.- Falsch. Die gesamte CDR-Datei wird verarbeitet. Standardwert ist "Falsch".
separator	Definiert Text, der zwischen den Datensätzen zu ignorieren ist. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none">- NewlineSearch. Das Trennzeichen ist ein Zeilenvorschub.- OffsetSearch. Das Trennzeichen ist durch die Zeichenanzahl ab dem Ende des vorhergehenden Datensatzes definiert.- PatternSearch. Das Trennzeichen wird durch einen regulären Ausdruck definiert.- TextSearch. Das Trennzeichen ist durch einen expliziten String oder einen String definiert, der dynamisch aus dem Quelldokument abgerufen wird.

ExcelToDataXml

Der Dokumentprozessor **ExcelToDataXml** wandelt Microsoft-Excel-Dokumente in XML um.

Die nachstehende Tabelle beschreibt die Eigenschaften des Dokumentprozessors **ExcelToDataXml**:

Eigenschaft	Beschreibung
enabled	Legt den Inhalt der Ausgabe fest: Die Eigenschaft enabled hat folgende Optionen: - Ausgewählt. Die Ausgabe enthält Rohdaten und formatierte Daten. - Gelöscht. Die Ausgabe enthält nur die formatierte Daten: Standardwert ist "Ausgewählt".
param1	Legt fest, ob die Rohdaten in der Ausgabe des Dokumentprozessors angezeigt werden, wenn sie sich von den formatierten Daten unterscheiden. param1 wird als Display_raw_data_when_different bezeichnet und hat nur eine Eigenschaft, enabled .
param2	Legt fest, ob der Ausgabe Elemente hinzugefügt werden, wenn eine Quelltablelle leere Zellen in der Mitte einer oder mehrerer Zeilen enthält. Beispiel: Eine Quelltablelle enthält drei Spalten und zwei Zeilen. In der ersten Zeile sind die drei Spalten ausgefüllt. In der zweiten Zeile sind die erste und letzte Spalte ausgefüllt und die zweite Spalte ist leer. Wenn param2 deaktiviert ist, erstellt der Prozessor zwei Elemente für die zweite Zeile mit den Werten der beiden ausgefüllten Spaltenzellen. Wenn param2 aktiviert ist, erstellt der Prozessor drei Elemente für die zweite Zeile: zwei Elemente mit den Werten der ausgefüllten Spaltenzellen und ein leeres Element für die leere Spaltenzelle. Standardwert ist „Deaktiviert“.
param3	Legt fest, ob der Ausgabe Elemente hinzugefügt werden, wenn eine Quelltablelle leere Zellen am Ende einer oder mehrerer Zeilen enthält. Beispiel: Eine Quelltablelle enthält drei Spalten und zwei Zeilen. In der ersten Zeile sind die drei Spalten ausgefüllt. In der zweiten Zeile ist die erste Spalte ausgefüllt und die zweite und dritte Spalte sind leer. Wenn param3 deaktiviert ist, erstellt der Prozessor ein Element für die zweite Zeile mit den Werten der ausgefüllten Spaltenzelle. Wenn param3 aktiviert ist, erstellt der Prozessor drei Elemente für die zweite Zeile: ein Element mit den Werten der ausgefüllten Spaltenzelle und zwei leere Elemente für die zwei leeren Spaltenzellen. Standardwert ist „Deaktiviert“.

Die XML-Ausgabe enthält die Daten und die Ergebnisse der Formeln aus dem ursprünglichen Excel-Dokument. Die Formeln selbst, Formatierungsinformationen sowie Makrocode gehen jedoch verloren. Wenn Sie Makrocode einsetzen müssen, verwenden Sie **ExcelToXml**, nicht **ExcelToDataXml**.

Die XML-Darstellung richtet sich nach einem Teilabschnitt des `ExcelToXml.xsd`-Schemas, das sich im Unterverzeichnis `doc` des Installationsverzeichnisses von befindet.

Die Prozessorausgabe ist in UTF-8 codiert. Wenn eine Umwandlung Eingaben vom Prozessor erhält, muss die Eingabecodierung auf UTF-8 eingestellt werden.

Der Prozessor unterstützt Excel Version 97 und höher. Er greift auf Eingaben direkt zu, nicht über die Excel-Anwendung. Excel muss auf dem Computer nicht installiert sein. Der Prozessor unterstützt sowohl das XLS-Format als auch das XLSX-Format.

Die Komponente ist in Java implementiert und setzt korrekte Konfigurierung der Java-Laufzeit-Umgebung (JRE) voraus.

ExcelToXml

Der Dokumentprozessor **ExcelToXml** wandelt Microsoft-Excel-Dokumente in XML um.

Die nachstehende Tabelle beschreibt die Eigenschaften des Dokumentprozessors **ExcelToXml**:

Eigenschaft	Beschreibung
Aktiviert	Definiert den Wert für param2 oder param3 .
param1	Definiert die Arbeitsblätter der Excel-Arbeitsmappe, die in das XML aufgenommen werden sollen. In der XML-Ausgabe wird jedes Arbeitsblatt durch ein <code><sheet></code> -Element dargestellt. param1 wird als include_sheets bezeichnet und hat die Eigenschaft value .
param2	Legt fest, ob der Dokumentprozessor in der Ausgabe-XML leere Zellen enthält, wenn die Zellen formatiert oder zusammengeführt werden. param2 wird als include_empty_cells bezeichnet und hat eine Eigenschaft namens enabled mit den folgenden Optionen: <ul style="list-style-type: none"> - Ausgewählt. Die Ausgabe schließt leere Zellen ein. - Gelöscht. Die Ausgabe lässt leere Zellen weg. Standardwert ist "Ausgewählt".
param3	Legt fest, ob der Dokumentprozessor in der Ausgabe-XML Excel-Makrocode enthält. param3 wird als include_macro_information bezeichnet und hat die Eigenschaft namens enabled mit den folgenden Optionen: <ul style="list-style-type: none"> - Ausgewählt. Der Dokumentprozessor enthält Makrocode - Gelöscht. Der Dokumentprozessor lässt Makrocode weg. Standardwert ist "Gelöscht".
param4	Legt fest, ob der Dokumentprozessor in der Ausgabe-XML statt Zellen unformatierte leere Zellen enthält. param4 wird als include_empty_non_formatted_cells bezeichnet und hat die Eigenschaft enabled mit den folgenden Optionen: <ul style="list-style-type: none"> - Ausgewählt. Die Ausgabe schließt leere Zellen ein. - Gelöscht. Die Ausgabe lässt leere Zellen weg. Standardwert ist "Gelöscht".
value	Definiert eine Liste folgender Optionen: <ul style="list-style-type: none"> - Die String "Alle". Die Ausgabe schließt alle Blätter ein. - Datenbehälter, die die Namen der Arbeitsblätter enthalten. Die Ausgabe schließt alle benannten Blätter ein. Wenn Sie ein Arbeitsblatt angeben, das nicht in der Arbeitsmappe vorhanden ist, erzeugt der Prozessor ein <code><sheet></code> -Element mit einem Warnhinweis. Die anderen Arbeitsblätter werden normal verarbeitet. Standardwert ist "Alle".

Die Daten, Formeln, Formatierungen und der Makrocode aus dem ursprünglichen Excel-Dokument bleiben in der XML-Ausgabe erhalten. Falls nur die eigentlichen Daten benötigt werden, verwenden Sie den **ExcelToDataXml**-Prozessor. Dieser bietet eine weniger umfangreiche Ausgabe und bessere Leistung.

Die XML-Darstellung richtet sich nach dem `ExcelToXml.xsd`-Schema, das sich im Unterverzeichnis `doc` des Installationsverzeichnis befindet.

Die Prozessorausgabe ist in UTF-16LE codiert. Wenn eine Umwandlung Eingaben vom Prozessor erhält, muss die Eingabecodierung auf UTF-16LE eingestellt werden.

Der Prozessor unterstützt Excel Version 97-2003. Der Prozessor greift direkt auf Eingaben zu, nicht über Excel. Excel muss auf dem Computer nicht installiert sein.

Die Komponente ist in Java implementiert und setzt korrekte Konfigurierung der Java-Laufzeit-Umgebung (JRE) voraus.

ExcelToXml_03_07_10

Der Dokumentprozessor **ExcelToXml_03_07_10** wandelt folgende Dateien in XML um:

- XLSX-Dateien, die mit Microsoft Excel 2007, 2010 oder 2013 erstellt wurden
- XLS-Dateien, die mit Microsoft Excel 2003, 2007, 2010 oder 2013 erstellt wurden

ExpandFrameSet

Der Dokumentprozessor **ExpandFrameSet** öffnet alle Frames eines HTML-Dokuments. Verwenden Sie diesen Dokumentprozessor, wenn das Quelldokument eines Parsers ein HTML-Frameset ist. Der Parser wird auf dem Inhalt aller Frames ausgeführt.

ExternalJavaPreProcessor

Der Dokumentprozessor **ExternalJavaPreProcessor** führt einen benutzerdefinierten Dokumentprozessor aus, der in Java implementiert ist.

Die nachstehende Tabelle beschreibt die Eigenschaften des Dokumentprozessors **ExternalJavaPreProcessor**:

Eigenschaft	Beschreibung
jclass	Definiert den Pfad der Java-Klasse.
jmethod	Definiert die Methode, die auszuführen ist.

Die Komponente ist in Java implementiert und setzt korrekte Konfigurierung der Java-Laufzeit-Umgebung (JRE) voraus.

Hinweis: Diese Komponente ist veraltet. Der IntelliScript-Editor zeigt dies für veraltete Skripte an. Verwenden Sie sie nicht in neuen Skripten. Erstellen Sie anstelle dessen einen benutzerdefinierten Java-Dokumentprozessor. Weitere Informationen hierzu finden Sie unter ["Entwickeln einer benutzerdefinierten Komponente" auf Seite 445](#).

HIPAAValidator

Der **HIPAAValidator**-Dokumentprozessor validiert HIPAA-Meldungen und generiert HIPAA-Bestätigungen. Das **HIPAA_Validation**-Projekt der HIPAA-Bibliothek verwendet diesen Prozessor.

Die nachstehende Tabelle beschreibt die Eigenschaften des Dokumentprozessors **HIPAAValidator**:

Eigenschaft	Beschreibung
param1	Die Eigenschaft param1 wird als include_empty_cells bezeichnet und hat nur eine Eigenschaft, value , mit den folgenden Optionen: <ul style="list-style-type: none">- LDNSB- Validator
param2	Definiert den Validierungstyp. Die Eigenschaft param2 wird als types_to_validate bezeichnet und hat nur eine Eigenschaft, value . Gültige Werte sind 1 bis 7.

Eigenschaft	Beschreibung
param3	Definiert das Format einer Fehlerbericht-Ausgabe. Die Eigenschaft param3 wird als report_formats bezeichnet und hat nur eine Eigenschaft, value , mit den folgenden Optionen: <ul style="list-style-type: none"> - HTML. Verwenden Sie diese Anzeige im Developer Tool. - XML. Für weitere Verarbeitung.
param4	Definiert den Bestätigungstyp. Die Eigenschaft param4 wird als generate_acknowledgments bezeichnet und hat nur eine Eigenschaft, value , mit den folgenden Optionen: <ul style="list-style-type: none"> - 277 - 824 - 997 - 999 - TA1
value	Definiert den Wert für param1 , param2 , param3 oder param4 .

Hinweis: Dieser Dokumentprozessor arbeitet auf den Plattformen Windows und Linux x64. Sie müssen vor seiner Verwendung das Zusatzpaket zur HIPAA-Validierung auf jedem Computer installieren und konfigurieren, auf dem Sie **HIPAAValidator** ausführen möchten.

PdfFormToXml_1_00

Der **PdfFormToXml_1_00**-Dokumentprozessor konvertiert PDF-Formulare in XML. Der Prozessor unterstützt Formulare, die dem Adobe AcroForms-Standard entsprechen.

PdfToTxt_3_02

Der **PdfToTxt_3_02**-Dokumentprozessor wandelt PDF-Dateien in Text um.

Die nachstehende Tabelle beschreibt die Eigenschaften des Dokumentprozessors **PdfToTxt_3_02**:

Eigenschaft	Beschreibung
Aktiviert	Definiert den Wert für param2 oder param4 .
param1	Definiert einen String oder eine Variable, der bzw. die den Wort-Abstand-Faktor enthält. Die Eigenschaft param1 wird als WordSpacingFactor bezeichnet und hat nur eine Eigenschaft, value , mit folgender String oder Variablen: Standardwert ist 1.8.
param2	Legt fest, ob das Ausgabedokument für Tabellen optimiert wird. Die Eigenschaft param2 wird als OptimizeForTables bezeichnet und hat nur eine Eigenschaft, enabled , mit den folgenden Optionen: <ul style="list-style-type: none"> - Ausgewählt. Das Ausgabedokument wird für Tabellen optimiert. - Gelöscht. Das Ausgabedokument wird nicht für Tabellen optimiert. Standardwert ist "Gelöscht".
param3	Definiert einen String oder eine Variable, der bzw. die das Kennwort enthält. Die Eigenschaft param3 wird als Password bezeichnet und hat nur eine Eigenschaft, value , mit folgendem String oder folgender Variablen:
param4	Die Eigenschaft param4 wird als HideNewPageChar bezeichnet und hat nur eine Eigenschaft, enabled , mit den folgenden Optionen: <ul style="list-style-type: none"> - Ausgewählt. Neue Seite-Zeichen werden ausgeblendet. - Gelöscht. Neue Seite-Zeichen werden nicht ausgeblendet. Standardwert ist "Gelöscht".

Eigenschaft	Beschreibung
param5	Definiert einen String oder eine Variable, der bzw. die erweiterte Optimierungen enthält. Die Eigenschaft param5 wird als AdvancedOptimizations bezeichnet und hat nur eine Eigenschaft, value , mit folgendem String oder folgender Variablen:
value	Definiert den Wert für param1 , param3 oder param5 .

Der Präprozessor PdfToTxt unterstützt möglicherweise bestimmte PDFs mit eingebetteten Schriftarten nicht. Wenn der Präprozessor fehlschlägt, kopieren Sie den Text aus der Eingabe-PDF-Datei in Notepad, um nach eingebetteten Schriftarten zu suchen. Wenn Sie den Text nicht einfügen können oder wenn der Text beschädigt ist, enthält die PDF-Datei wahrscheinlich eingebettete Schriftarten.

Hinweis: Diese Komponente ist veraltet. Der IntelliScript-Editor zeigt dies für veraltete Projekte an. Verwenden Sie sie nicht in neuen Skripten.

PdfToTxt_4_

Der **PdfToTxt_4**-Dokumentprozessor wandelt PDF-Dateien in Text oder XML um.

Die nachstehende Tabelle beschreibt die Eigenschaften des Dokumentprozessors **PdfToTxt_4**:

Eigenschaft	Beschreibung
param1	Legt das PDF-Tabellen-Layout fest. Die Eigenschaft param1 hat nur eine Option: PdfLayout
value	Legt das PDF-Tabellen-Layout fest. Doppelklicken Sie auf die Eigenschaft value , um den Editor zur Konfiguration von Tabellen zu öffnen.

Der Tabellenkonfiguration-Editor passt die Art an, wie Tabellen gelesen werden. Verwenden Sie ihn, um Probleme mit Spaltenausrichtung, Wortumbruch, Zeilenabstand und Überlauf aus einer Zelle in eine andere zu beheben. Weitere Informationen hierzu finden Sie unter ["PdfToTxt_4-Editor zur Konfiguration von Tabellen" auf Seite 174](#).

Der Dokumentprozessor **PdfToTxt_4** generiert standardmäßig die Textausgabe. Verwenden Sie den Tabellenkonfiguration-Editor zur Auswahl der XML-Ausgabe. Der XML-Code entspricht dem Schema PDF4.xsd, das Sie im folgenden Verzeichnis finden:

```
<INSTALLATIONSVERZEICHNIS>\DataTransformation\doc
```

Wenn Sie den Dokumentprozessor **PdfToTxt_4** verwenden, legen Sie die Eingabecodierung auf UTF-8 fest, um dem Parser, Mapper oder Serializer ein korrektes Lesen des Dokuments zu ermöglichen.

Hinweis: Der Präprozessor PdfToTxt unterstützt möglicherweise bestimmte PDFs mit eingebetteten Schriftarten nicht. Wenn der Präprozessor fehlschlägt, kopieren Sie den Text aus der Eingabe-PDF-Datei in Notepad, um nach eingebetteten Schriftarten zu suchen. Wenn Sie den Text nicht einfügen können oder wenn der Text beschädigt ist, enthält die PDF-Datei wahrscheinlich eingebettete Schriftarten.

PowerpointToTextML

Der Dokumentprozessor **PowerpointToTextML** wandelt Microsoft PowerPoint-Dateien (*.ppt) in das XML-Schema TextML um. Weitere Informationen hierzu finden Sie unter ["TextML-XML-Schema" auf Seite 173](#).

Diese Komponente unterstützt PowerPoint Version 97 und höher. Sie greift auf Eingaben direkt zu, nicht über PowerPoint. Microsoft PowerPoint braucht auf dem Computer nicht installiert zu sein.

Die Komponente ist in Java implementiert und setzt korrekte Konfigurierung der Java-Laufzeit-Umgebung (JRE) voraus.

ProcessByTransformers

Der Dokumentprozessor **ProcessByTransformers** führt einen Transformer oder eine Sequenz von Transformatoren mit dem gesamten Dokument aus. An der Ausgabe der Transformer kann dann eine Umwandlung ausgeführt werden.

Definieren Sie die Liste von Transformatoren unter der Zeile **Transformer**.

ProcessorPipeline

Der Dokumentprozessor **ProcessorPipeline** definiert eine Sequenz von Dokumentprozessoren, um ein Dokument auszuführen. Verwenden Sie diese Komponente, wenn Sie einen oder mehr Dokumentprozessoren ausführen möchten.

Definieren Sie die Liste von Dokumentprozessoren unter der Zeile **pre_processor_list**.

RtfToTextML

Der Dokumentprozessor **RtfToTextML** konvertiert RTF-Dateien in das XML-Schema TextML. Weitere Informationen hierzu finden Sie unter ["TextML-XML-Schema" auf Seite 173](#).

Die Prozessorausgabe ist in UTF-16LE codiert. Wenn eine Umwandlung Eingaben vom Prozessor erhält, muss die Eingabecodierung auf UTF-16LE eingestellt werden.

WordToXml

Der Dokumentprozessor **WordToXml** wandelt Microsoft-Word-Dokumente in XML um.

Die Prozessorausgabe ist in UTF-16LE codiert. Wenn eine Umwandlung Eingaben vom Prozessor erhält, muss die Eingabecodierung auf UTF-16LE eingestellt werden.

Die Komponente unterstützt Word ab Version 97. Sie greift auf Eingaben direkt zu, nicht über Microsoft Word. Word muss auf dem Computer nicht installiert sein.

Die Komponente ist in Java implementiert und setzt korrekte Konfigurierung der Java-Laufzeit-Umgebung (JRE) voraus.

XmlToDocument_372

Der Dokumentprozessor **XmlToDocument_372** konvertiert XML-Daten in Dokumentformate wie PDF oder Excel. Sie können ihn als Postprozessor zum Konvertieren der Parser- oder Mapper-Ausgabe in verschiedene Dokumenttypen verwenden.

Diese Komponente verwendet zum Generieren der Ausgabedokumente das Eclipse-Add-On **Business Intelligence and Reporting Tool** (BIRT). In BIRT müssen Sie einen Bericht konfigurieren, der das XML in das gewünschte Dokumentformat umwandelt. Der Prozessor **XmlToDocument_372** führt den Bericht aus.

Sie können BIRT aus dem Speicherort herunterladen, der in der Datei `readme_BIRT.txt` unter `<Installationsverzeichnis der Data Transformation-Engine>/readme_Birt.txt` angegeben wird.

Weitere Informationen zu BIRT finden Sie unter <http://www.eclipse.org/birt>.

Hinweis: Verwenden Sie für BIRT Version 4.5 den Präprozessor **XmlToDocument_45** anstelle des Präprozessors **XmlToDocument_372**.

In der folgenden Tabelle werden die Eigenschaften des Dokumentprozessors **XmlToDocument_372** beschrieben:

Eigenschaft	Beschreibung
param1	Pfad und Dateiname der BIRT-Datei *.rptdesign. Der Name der Eigenschaft param1 lautet report_file . Sie enthält die Eigenschaft value mit dem Pfad und Dateinamen.
param2	Das Format des Ausgabedokuments. Der Name der Eigenschaft param2 lautet output_format . Sie enthält die Eigenschaft value , für die die folgenden Optionen verfügbar sind: <ul style="list-style-type: none">- pdf. PDF-Dokument- doc. Microsoft Word-Dokument- xls. Microsoft Excel-Arbeitsmappen- ppt. Microsoft PowerPoint-Präsentation- html. HTML-Webseite- ps. PostScript-Dokument Standardwert ist „pdf“.
param3	Eine Variable, die den Speicherort der Datei *.rptdesign enthält. Der Name der Eigenschaft param3 lautet report_location . Sie enthält die Eigenschaft value , die auf die Variable verweist. Standardwert ist \$VarServiceInfo/*s/ServiceLocation.
Wert	Enthält den Wert für param1 , param2 oder param3 .

Hinweis: Ab Version 9.5.1 ist der Prozessor **XmlToDocument** veraltet. Der IntelliScript-Editor zeigt den Präprozessor **XmlToDocument** in vorhandenen Skripts noch an, aber neuen Skripts können Sie diesen Präprozessor nicht mehr hinzufügen. Verwenden Sie stattdessen den Präprozessor **XmlToDocument_372**.

XmlToDocument_45

Der Dokumentprozessor **XmlToDocument_45** konvertiert XML-Daten in Dokumentformate, wie z. B. PDF oder Excel. Sie können ihn als Postprozessor zum Konvertieren der Parser- oder Mapper-Ausgabe in verschiedene Dokumenttypen verwenden.

Diese Komponente verwendet zum Erzeugen der Ausgabedokumente das Eclipse-Add-On **Business Intelligence and Reporting Tool** (BIRT) Version 4.5. In BIRT konfigurieren Sie einen Bericht, der XML in das gewünschte Dokumentformat umwandelt. Der Prozessor **XmlToDocument_45** führt den Bericht aus.

Sie können BIRT aus dem Speicherort herunterladen, der in der Datei `readme_BIRT.txt` unter `<Installationsverzeichnis der Data Transformation-Engine>/readme_Birt.txt` angegeben wird.

Weitere Informationen zu BIRT finden Sie unter <http://www.eclipse.org/birt>.

In der folgenden Tabelle werden die Eigenschaften des Dokumentprozessors **XmlToDocument_45** beschrieben:

Eigenschaft	Beschreibung
param1	Pfad und Dateiname der BIRT-Datei *.rptdesign. Der Name der Eigenschaft param1 lautet report_file . Sie enthält die Eigenschaft value mit dem Pfad und Dateinamen.
param2	Das Format des Ausgabedokuments. Der Name der Eigenschaft param2 lautet output_format . Sie enthält die Eigenschaft value , für die die folgenden Optionen verfügbar sind: <ul style="list-style-type: none"> - pdf. PDF-Dokument - doc. Microsoft Word-Dokument - xls. Microsoft Excel-Arbeitsmappen - ppt. Microsoft PowerPoint-Präsentation - html. HTML-Webseite - ps. PostScript-Dokument Standardwert ist „pdf“.
param3	Eine Variable, die den Speicherort der Datei *.rptdesign enthält. Der Name der Eigenschaft param3 lautet report_location . Sie enthält die Eigenschaft value , die auf die Variable verweist. Standardwert ist \$VarServiceInfo/*s/ServiceLocation.
Wert	Enthält den Wert für param1 , param2 oder param3 .

XmlToExcel

Der Dokumentprozessor **XmlToExcel** wandelt XML-Dokumente in das Microsoft Excel-Format um.

Der Prozessor wird an der XML-Darstellung von Excel-Arbeitsmappen ausgeführt. Die XML-Darstellung muss in UTF-16LE codiert sein und dem Schema `ExcelToXml.xsd` entsprechen. Dieses Schema befindet sich im Unterverzeichnis `doc` des Installationsverzeichnis. Die Schemadatei wird lediglich zu Informationszwecken mitgeliefert. Sie können den Prozessor verwenden, ohne das Schema zu Ihrem Projekt hinzuzufügen.

Der Prozessor kehrt eine **ExcelToXml**-Operation um. Beispielsweise können Sie mit **ExcelToXml** eine Excel-Arbeitsmappe in XML konvertieren. Sie können dann einige XML-Daten ändern und anschließend die Daten mit **XmlToExcel** wieder in eine Excel-Arbeitsmappe umwandeln.

Die Komponente unterstützt Excel ab Version 97. Die Ausgabe wird direkt geschrieben, nicht über Microsoft Excel. Excel muss auf dem Computer nicht installiert sein.

Die Komponente ist in Java implementiert und setzt korrekte Konfigurierung der Java-Laufzeit-Umgebung (JRE) voraus.

XmlToXlsx

Der Dokumentprozessor **XmlToXlsx** wandelt XML-Dokumente in das Microsoft Excel-Format „XLSX“ um. Der Dokumentprozessor **XmlToXlsx** kann optional eine XLSX-Vorlage zum Erzeugen des XLSX-Dokuments verwenden.

Der Prozessor wird an der XML-Darstellung von Excel-Arbeitsmappen ausgeführt. Die XML-Darstellung muss in UTF-8 codiert sein und dem Schema `ExcelToXml_03_07_10.xsd` entsprechen. Dieses Schema befindet sich im Unterverzeichnis `doc` des Installationsverzeichnis. Die Schemadatei wird zu Informationszwecken bereitgestellt.

Der Prozessor kehrt eine **ExcelToXml_03_07_10**-Operation um. Verwenden Sie den Prozessor **ExcelToXml_03_07_10** in einer Datenprozessor-Umwandlung, um eine Excel-Arbeitsmappe in XML

umzuwandeln. Nachdem Sie XML-Daten verarbeitet haben, verwenden Sie den Prozessor **XmlToXlsx**, um die Daten wieder in eine Excel-Arbeitsmappe umzuwandeln.

Die Komponente unterstützt Excel ab Version 2007. Die Ausgabe wird direkt geschrieben, nicht über Microsoft Excel. Excel muss auf dem Computer nicht installiert sein.

Die Komponente ist in Java implementiert und setzt korrekte Konfigurierung der Java-Laufzeit-Umgebung (JRE) voraus.

In der folgenden Tabelle werden die Eigenschaften des Dokumentprozessors **XmlToXlsx** beschrieben:

Eigenschaft	Beschreibung
param1	Eine Variable, die den Namen der Vorlagendatei *.xlsx enthält. Die Eigenschaft param1 enthält die Eigenschaft template_file , die den Namen der Vorlagendatei angibt.
param2	Eine Variable, die den Speicherort der Vorlagendatei *.xlsx enthält. Die Eigenschaft param2 enthält die Eigenschaft template_location , die den Pfad der Vorlagendatei angibt.

Hinweis: Excel-Arbeitsblätter, die nur in der Vorlage vorhanden sind, werden als Vorlage in der XLSX-Ausgabe dargestellt. Excel-Arbeitsblätter, die in der Vorlage und in der XML-Datei definiert sind, erhalten Zellstile aus der Vorlage und Zellwerte aus der XML-Datei.

Zum Anwenden eines Zellstils aus einer anderen Zelle in der Vorlage können Sie das Attribut `style_as` in der XML-Datei als Teil des Elements `cell` verwenden.

Beispiel

Wenn Sie das Attribut `style_as` zum Anwenden eines Zellstils aus einer Zelle im aktuellen Arbeitsblatt verwenden, legen Sie für das Attribut `style_as` die Zellnummer fest, die den Stil enthält. In folgendem Beispiel wird der Stil aus Feld A1 im aktuellen Arbeitsblatt auf die Zelle angewendet, die vom XML-Kontext definiert wird, nämlich Zeile 3 und Zelle 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>
```

Legen Sie zum Anwenden eines Zellstils aus einer Zelle in einem anderen Arbeitsblatt für das Attribut `style_as` das Arbeitsblatt und die Zellnummer fest, die den Stil enthalten. In folgendem Beispiel wird der Stil aus dem Feld A1 im Arbeitsblatt der Arbeitsmappe mit der Bezeichnung `Summary` angewendet.

```
<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>
```

TextML-XML-Schema

Einige der Dokumentprozessoren wandeln Dokumente in das XML-Vokabular TextML um. Dieses ist ein einfaches XML-Vokabular, mit dem der Dokumentinhalt ohne Layout gespeichert wird.

Das TextML-Schema (`textML.xsd`) befindet sich im Unterordner `doc` des Installationsordners.

Das folgende Beispiel stellt den Code eines TextML-Dokuments dar.

```
<?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>
```

PdfToTxt_4-Editor zur Konfiguration von Tabellen

Der Editor zur Konfiguration von Tabellen passt die Art an, wie der **PdfToTxt_4**-Dokumentprozessor Tabellen in PDF-Dokumente konvertiert.

Verwenden Sie den Editor zur Konfiguration von Tabellen, wenn die Standardeinstellungen des **PdfToTxt_4**-Dokumentprozessors derzeit Spaltenausrichtung, Zeilenumbrüche, Zeilenabstände oder das Überlaufen von einer Zelle in die andere nicht wiedergeben.

Hinweis: Die Benutzerschnittstelle für den Editor zur Konfiguration von Tabellen erscheint nur auf Englisch.

1. Fügen Sie dem Skript einen Parser, Mapper, Serializer oder **AdditionalInputPort** hinzu.
2. Setzen Sie unter der Eigenschaft **example_source** die Eigenschaft **pre_processor** auf PdfToTxt_4.
3. Doppelklicken Sie unter der Eigenschaft **pre-processor** auf die Eigenschaft **value**.

Der Editor zur Konfiguration von Tabellen wird angezeigt. Im oberen Bereich wird das Eingabe-PDF-Dokument angezeigt und im unteren Bereich die Ausgabe von **PdfToTxt_4**.

Die Befehle zur Tabellenbearbeitung erscheinen in der Symbolleiste im oberen Fensterbereich. Durch Klicken auf die rechte Maustaste können Sie ein Bearbeitungsmenü anzeigen.

4. Navigieren Sie zu einer Tabelle im PDF-Dokument und klicken Sie auf **Tabelle hinzufügen**.

Das System zeigt den Namen der Tabelle in den Feldern **Tables** und **Name** an.

5. Wählen Sie **Use Regular Expressions** aus. Geben Sie in das Feld **Table Start** einen regulären Ausdruck ein, der die obere linke Ecke der Tabelle definiert.

Tipp: Verwenden Sie die Überschriften der ersten beiden Spalten als regulären Ausdruck. Fügen Sie bei Bedarf weitere Spaltenüberschriften hinzu, damit der **Table Start** eindeutig ist. Trennen Sie die Überschriften mit Hilfe eines Leerzeichens voneinander, auch wenn die Spalten voneinander getrennt sind.

6. Geben Sie in das Feld **Tabellenende** einen regulären Ausdruck ein, der den Text unmittelbar nach der Tabelle definiert.

Hinweis: Der Wert von **Table End** muss im Text des Dokuments vorkommen, nicht in der Fußzeile einer Seite.

7. Klicken Sie auf **Verarbeiten**.

Der Editor zeigt die Tabellenkonfiguration an, die von **PdfToTxt_4** erkannt wird. Anfang und Ende der Tabelle werden als horizontale blaue Linien dargestellt. Die standardmäßigen Spaltenbegrenzungen werden als vertikale rote Linien dargestellt.

8. Zum Bearbeiten der Spaltenränder, führen Sie einen oder mehrere der folgenden Schritte aus:

- Um die Position eines Spaltenrands zu verändern, ziehen Sie ihn nach links oder rechts.
- Klicken Sie auf **Add Column**, um eine Spalte hinzuzufügen.
- Klicken Sie auf **Remove Column** und wählen Sie einen Spaltenrand, um eine Spalte zu löschen.

Hinweis: Wenn die Tabelle horizontal verbundene Zellen umfasst, schneidet **PdfToTxt_4** die Einträge eventuell ab.

9. Betrachten Sie das Ausgabefenster, um zu prüfen, ob die Tabelle korrekt konvertiert worden ist. Ist dies nicht der Fall, korrigieren Sie die Tabellendefinitionen.

10. Wiederholen Sie die Schritte 1-9 für jede einzelne Tabelle im PDF-Dokument.

11. Klicken Sie auf **OK**, um zum Developer-Tool zurückzukehren.

In der Eigenschaft **value** des **PdfToTxt_4_-**Dokumentprozessors wird ein String angezeigt, der die Tabellenkonfiguration definiert.

Editor-Optionen

In der folgenden Tabelle werden die Steuerungen und Felder des **PdfToTxt_4**-Editors zur Konfiguration von Tabellen beschrieben.

Steuerung oder Feld	Beschreibung
Vergrößern	Vergrößert die PDF-Anzeige.
Verkleinern	Verkleinert die PDF-Anzeige.
Breite anpassen	Passt das PDF-Dokument der Breite des Fensters an.
Vorherige Seite	Geht zur vorherigen Seite.
Nächste Seite	Geht zur nächsten Seite.
Suchen	Sucht im PDF nach einem String.
Tabelle hinzufügen	Fügt eine Tabelle zur Konfiguration hinzu.
Tabelle entfernen	Entfernt eine Tabelle aus der Konfiguration.
Spalte hinzufügen	Fügt eine Spaltenbegrenzung zur aktuellen Tabelle hinzu.
Spalte entfernen	Löscht die markierte Spaltenbegrenzung.
Verarbeiten	Wendet die aktuellen Tabellendefinitionen an. Klicken Sie nach jeder Aktion, die mit der Tabelle oder mit Spalten zusammenhängt, auf Verarbeiten , um diese Aktion anzuwenden.
Tabellen	Eine Liste mit Tabellen, die im Eingabe-PDF definiert ist. Sie können eine Tabelle auswählen, indem Sie darauf klicken.
Name	Name der aktuell markierten Tabelle.

Steuerung oder Feld	Beschreibung
Tabellenbeginn	Ein Ausdruck, der die obere linke Ecke der Tabelle definiert
Tabellenende	Ein Ausdruck, der den direkt auf die Tabelle folgenden Text definiert
Kopfzeile	Ein Ausdruck, der das Ende der Kopfzeile definiert Mit Hilfe dieser Option können Sie die Kopfzeile von der Verarbeitung der Tabelle ausnehmen.
Fußzeile	Ein Ausdruck, der das Ende der Fußzeile definiert Mit Hilfe dieser Option können Sie die Fußzeile von der Verarbeitung der Tabelle ausnehmen.
Reguläre Ausdrücke verwenden	Ist diese Option ausgewählt, interpretiert der Prozessor Tabellenanfang , Tabellenende , Kopfzeile und Fußzeile als reguläre Ausdrücke und sucht nach übereinstimmendem Text. Ist diese Option nicht ausgewählt, interpretiert der Prozessor diese Felder als literalen Text.
Während der Laufzeit neu berechnen	Wenn Sie diese Option auswählen, ignoriert PdfToTxt_4 die Tabellenkonfigurationen, die Sie mit Hilfe des Editors zur Konfiguration von Tabellen angegeben haben. Diese Funktion ist nützlich, wenn die Tabellen in einem PDF-Dokument so einfach sind, dass sie mit Hilfe von PdfToTxt_4 ohne eine spezielle Konfiguration verarbeitet werden können. Nehmen wir einmal an, dass eine Finanzaufstellung im PDF-Format eine Tabelle enthält, deren Spalten von Monat zu Monat leicht voneinander abweichen. Aktivieren Sie die Option Während der Laufzeit neu berechnen , damit PdfToTxt_4 die Spaltenbreite während der Laufzeit anpasst.
Jetzt neu berechnen	Wenn Sie die Tabellendefinition geändert haben, beispielsweise indem Sie Spaltenbegrenzungen geändert oder eine Kopfzeile oder Fußzeile hinzugefügt haben, klicken Sie auf Jetzt neu berechnen , damit die Tabellendefinition aktualisiert wird.
Seite	Aktuell angezeigte Seitennummer der PDF-Datei.
Ausgabe als XML	Generiert die PdfToTxt_4 -Ausgabe als XML anstatt als Text.
Delimiter	Geben Sie ein Zeichen ein, das in der Textausgabe als Trennung der Spalten verwendet wird. Standardmäßig ist dies ein senkrechter Strich ().
OK	Klicken Sie darauf, um die Tabellenkonfiguration zu speichern und zum Developer Tool zurückzukehren.
Abbrechen	Klicken Sie darauf, um zum Developer Tool zurückzukehren, ohne die Tabellenkonfiguration zu speichern.
Tabellen-Navigationshilfe	Die Tabellen-Navigationshilfe zeigt an, wie oft eine Tabelle in einem PDF-Dokument vorkommt. Ein Beispiel dafür ist <code>Table 'Table 1' found 2 times</code> (Tabelle „Tabelle 1“ 2 Mal gefunden). Mit Hilfe der Pfeile neben dieser Information können Sie in den Instanzen derselben Tabellenstruktur vorwärts- und zurückgehen.

Beispiel für eine PDF-Umwandlung

In diesem Beispiel wird das Verfahren zur Konfiguration von Tabellen mit Hilfe von **PdfToTxt_4** anhand eines beispielhaften Parserprojekts und PDF-Dokuments erläutert.

Bei der Prozessoreingabe handelt es sich um einen kurzen Finanzbereich im PDF-Format. Der Bericht umfasst Text sowie zwei Tabellen. Der Editor wird zur Konfiguration von Tabellen verwendet, um sicherzustellen, dass der Prozessor die Tabellen korrekt in Text umwandelt.

Erste Tabelle konfigurieren

1. Konfigurieren Sie einen Parser und weisen Sie ihm das PDF-Dokument als **example_source** zu. Doppelklicken Sie auf die Eigenschaft **value**, um den Editor zur Konfiguration von Tabellen zu öffnen.
2. Gehen Sie in der PDF-Anzeige zur ersten Tabelle.
3. Legen Sie Table Start = GID RMS ID für die Überschriften der ersten beiden Spalten der Tabelle fest. Achten Sie darauf, dass bei den Ausdrücken die Groß- und Kleinschreibung beachtet werden muss.
4. Legen Sie Table End = Forward exchange transactions für den Text fest, der direkt auf die Tabelle folgt. Der Editor zeigt die Tabellenkonfiguration an.
5. Bei Bedarf passen Sie die Tabellendefinition und die Spalten an. Sie können Spaltenbegrenzungen verschieben, hinzufügen oder entfernen.

Zweite Tabelle konfigurieren

Die zweite Tabelle umfasst mehrere Seiten.

1. Klicken Sie auf **Tabelle hinzufügen**.
Das System zeigt in den Feldern **Tabellen** und **Name** die zweite Tabelle an.
2. Legen Sie fest: Tabellenbeginn = Ticker Shares Traded-
3. Legen Sie fest: Tabellenende = Conclusion. Dies ist der Text, der direkt auf die Tabelle folgt.
4. Klicken Sie auf **Verarbeiten**, um die Tabelle zu konfigurieren.
5. Passen Sie die rechten Begrenzungen der Spalten **Shares Traded** und **Currency** an.
6. Führen Sie die folgenden Schritte durch, um die Seitenkopf- und -fußzeile aus dem Ausgabedokument zu eliminieren:
 - a. Legen Sie fest: Kopfzeile = Gain/Loss.
 - b. Legen Sie fest: Fußzeile = Page [1-9].
 - c. Klicken Sie auf **Verarbeiten**.

KAPITEL 14

Formate

Dieses Kapitel umfasst die folgenden Themen:

- [Übersicht über Formate, 178](#)
- [Standardformateigenschaften, 179](#)
- [Formatkomponenten: Referenz, 179](#)
- [Delimiter-Komponenten: Referenz, 185](#)
- [Formatpräprozessor-Komponenten: Referenz, 190](#)

Übersicht über Formate

Die Eigenschaft `format` eines Parsers definiert das Format der Dokumente, die für diese Datenumwandlung verarbeitet werden sollen. Der Wert der Eigenschaft ist eine der folgenden Formatkomponenten:

```
BinaryFormat  
CustomFormat  
HtmlFormat  
RtfFormat  
TextFormat  
XmlFormat
```

Die `format`-Eigenschaft besitzt eigene Eigenschaften, die im Einzelnen festlegen, wie die Eingabe vom Parser interpretiert und verarbeitet wird.

In der folgenden Tabelle werden die Teilkomponenten beschrieben, die Sie in ein Format verschachteln können:

Teilkomponente	Beschreibung
Delimiter	Definiert eine Hierarchie von Zeichen oder Strings, die die im Dokument enthaltenen Informationen (zum Beispiel Zeilenvorschub und Tabulator) organisieren.
Formatpräprozessor	Bereinigt die Quelle, bevor der Parser nach Ankern zu suchen beginnt.
Standardtransformer	Führt vordefinierte Operationen mit der Ausgabe jedes Ankers durch.

Standardformateigenschaften

In der folgenden Tabelle werden die Standardeigenschaften von Formatkomponenten beschrieben:

Eigenschaft	Beschreibung
default_Transformers	Definiert eine Liste von Transformern, die der Parser auf die Ausgabe der einzelnen Content-Anker anwendet.
delimiters	<p>Definiert die Informationsstruktur im Dokument. Sie können eine der folgenden Optionen auswählen:</p> <ul style="list-style-type: none">- CommaDelimited. Datenfelder sind durch Kommas getrennt.- DelimiterHierarchy. Datenfelder sind getrennt oder umgeben von Textzeichen.- HL7. Datenfelder werden gemäß HL7-Standard voneinander getrennt.- Positionsbasiert. Datenfelder werden durch die Anzahl der Zeichen zwischen ihnen definiert.- PostScript. Datenfelder werden gemäß dem PostScript-Format definiert.- RTF. Datenfelder werden gemäß dem RTF-Format definiert.- SGML. Datenfelder werden gemäß dem SGML-Format definiert.- SpaceDelimited. Datenfelder sind durch Leerstellen getrennt.- TabDelimited. Datenfelder sind durch Tabulatorzeichen getrennt. <p>Weitere Informationen hierzu finden Sie unter "Delimiter-Komponenten: Referenz" auf Seite 185.</p>
name	Eine beschreibende Kennzeichnung für die Komponente. Diese Kennzeichnung wird in der Protokolldatei sowie der Ereignisansicht angezeigt. Ermitteln Sie anhand der Eigenschaft name , welche Komponente das Ereignis verursacht hat.
pre_processor	<p>Definiert einen Format-Präprozessor, der die Eingabe nach einem beliebigen von Ihnen für die pre_processor-Eigenschaft der example_source definierten Dokumentprozessor verarbeitet. Sie können eine der folgenden Optionen auswählen:</p> <ul style="list-style-type: none">- HtmlProcessor. Wandelt alle Kombinationen aus Tabulatorzeichen, Leerzeichen oder Zeilenwechselzeichen in ein einzelnes Leerzeichen um. Er ist nicht auf HTML-Dokumente beschränkt.- RtfProcessor. Normalisiert RTF-Dateien. <p>Standardmäßig ist keine Option ausgewählt.</p>
remark	Ein benutzerdefinierter Kommentar, der den Zweck oder die Aktion der Komponente beschreibt.

Formatkomponenten: Referenz

Formatkomponenten definieren das Format von Eingabedokumenten. Definieren Sie die Formatkomponenten unter der Eigenschaft **format** eines **Parser**.

BinaryFormat

Das **BinaryFormat**-Format verarbeitet Binärdateien und Textdateien, die als Puffer von Binärbytes behandelt werden sollen.

Die nachstehende Tabelle beschreibt die Eigenschaften des **BinaryFormat**-Formats:

Eigenschaft	Beschreibung
default_transformers	Definiert eine Liste von Transformern, die der Parser auf die Ausgabe der einzelnen Content-Anker anwendet. Standardwert ist "Leer".
delimiters	Definiert die Informationsstruktur im Dokument. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none">- CommaDelimited. Datenfelder sind durch Kommas getrennt.- DelimiterHierarchy. Datenfelder sind getrennt oder umgeben von Textzeichen.- HL7. Datenfelder werden gemäß HL7-Standard voneinander getrennt.- Positionsbasiert. Datenfelder werden durch die Anzahl der Zeichen zwischen ihnen definiert.- PostScript. Datenfelder werden gemäß dem PostScript-Format definiert.- RTF. Datenfelder werden gemäß dem RTF-Format definiert.- SGML. Datenfelder werden gemäß dem SGML-Format definiert.- SpaceDelimited. Datenfelder sind durch Leerstellen getrennt.- TabDelimited. Datenfelder sind durch Tabulatorzeichen getrennt. Weitere Informationen hierzu finden Sie unter "Delimiter-Komponenten: Referenz" auf Seite 185 . Standardwert ist "Positionsbasiert".
name	Eine beschreibende Kennzeichnung für die Komponente. Diese Kennzeichnung wird in der Protokolldatei sowie der Ereignisansicht angezeigt. Ermitteln Sie anhand der Eigenschaft name , welche Komponente das Ereignis verursacht hat.
pre_processor	Definiert einen Format-Präprozessor, der die Eingabe nach einem beliebigen von Ihnen für die pre_processor -Eigenschaft der example_source definierten Dokumentprozessor verarbeitet. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none">- HtmlProcessor. Wandelt alle Kombinationen aus Tabulatorzeichen, Leerzeichen oder Zeilenwechselzeichen in ein einzelnes Leerzeichen um. Er ist nicht auf HTML-Dokumente beschränkt.- RtfProcessor. Normalisiert RTF-Dateien. Standardwert ist "Leer".
remark	Ein benutzerdefinierter Kommentar, der den Zweck oder die Aktion der Komponente beschreibt.

CustomFormat

Das Format **CustomFormat** ist ein benutzerdefiniertes Format für die Verarbeitung eines beliebigen Quelldokumenttyps.

Die nachstehende Tabelle beschreibt die Eigenschaften des Formats **CustomFormat**:

Eigenschaft	Beschreibung
default_transformers	Definiert eine Liste von Transformern, die der Parser auf die Ausgabe der einzelnen Content-Anker anwendet. Standardwert ist "Leer".
delimiters	Definiert die Informationsstruktur im Dokument. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none">- CommaDelimited. Datenfelder sind durch Kommas getrennt.- DelimiterHierarchy. Datenfelder sind getrennt oder umgeben von Textzeichen.- HL7. Datenfelder werden gemäß HL7-Standard voneinander getrennt.- Positionsbasiert. Datenfelder werden durch die Anzahl der Zeichen zwischen ihnen definiert.- PostScript. Datenfelder werden gemäß dem PostScript-Format definiert.- RTF. Datenfelder werden gemäß dem RTF-Format definiert.- SGML. Datenfelder werden gemäß dem SGML-Format definiert.- SpaceDelimited. Datenfelder sind durch Leerstellen getrennt.- TabDelimited. Datenfelder sind durch Tabulatorzeichen getrennt. Weitere Informationen hierzu finden Sie unter "Delimiter-Komponenten: Referenz" auf Seite 185 . Standardwert ist DelimiterHierarchy.
name	Eine beschreibende Kennzeichnung für die Komponente. Diese Kennzeichnung wird in der Protokolldatei sowie der Ereignisansicht angezeigt. Ermitteln Sie anhand der Eigenschaft name , welche Komponente das Ereignis verursacht hat.
pre_processor	Definiert einen Format-Präprozessor, der die Eingabe nach einem beliebigen von Ihnen für die pre_processor -Eigenschaft der example_source definierten Dokumentprozessor verarbeitet. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none">- HtmlProcessor. Wandelt alle Kombinationen aus Tabulatorzeichen, Leerzeichen oder Zeilenwechselzeichen in ein einzelnes Leerzeichen um. Er ist nicht auf HTML-Dokumente beschränkt.- RtfProcessor. Normalisiert RTF-Dateien. Standardwert ist "Leer".
remark	Ein benutzerdefinierter Kommentar, der den Zweck oder die Aktion der Komponente beschreibt.

Beispiel

Ein Quelldokument weist die folgende Struktur auf:

```
Ron      Lehrer  && 547329876:27
Evelyn   Kern    && 9875424: 53
```

Jede Zeile im Dokument ist ein Datensatz mit dem Namen, der Kennung und dem Alter einer Person. Die einzelnen Felder sind durch die Symbole && und : voneinander getrennt. Sie können mehrere Leerzeichen an beliebigen Stellen enthalten.

Mit Hilfe von **CustomFormat** lässt sich dieses Dokument parsen. Weisen Sie in der Eigenschaft **delimiters** des Formats eine **DelimiterHierarchy** zu, die folgende Symbole enthält:

```
newline
&&
:
```

Außerdem weisen Sie in der Eigenschaft **default_transformers** den **HtmlProcessor** zu, der die überflüssigen Leerzeichen aus der Ausgabe entfernt.

HtmlFormat

Das **HtmlFormat**-Format definiert das Format von HTML-Dateien.

Die nachstehende Tabelle beschreibt die Eigenschaften des **HtmlFormat**-Formats:

Eigenschaft	Beschreibung
default_transformers	<p>Definiert eine Liste von Transformern, die der Parser auf die Ausgabe der einzelnen Content-Anker anwendet.</p> <p>Standardwert ist die folgende Liste von Transformern:</p> <ul style="list-style-type: none">- RemoveTags. Entfernt HTML-Tags.- HtmlEntitiesToASCII. Konvertiert HTML-Entitäten in deren ASCII-Äquivalente.- HtmlProcessor. Wandelt alle Kombinationen aus Tabulatorzeichen, Leerzeichen oder Zeilenwechselzeichen in ein einzelnes Leerzeichen um.- RemoveMarginSpace. Entfernt vorangestellte und angehängte Leerzeichen.
delimiters	<p>Definiert die Informationsstruktur im Dokument. Sie können eine der folgenden Optionen auswählen:</p> <ul style="list-style-type: none">- CommaDelimited. Datenfelder sind durch Kommas getrennt.- DelimiterHierarchy. Datenfelder sind getrennt oder umgeben von Textzeichen.- HL7. Datenfelder werden gemäß HL7-Standard voneinander getrennt.- Positionsbasiert. Datenfelder werden durch die Anzahl der Zeichen zwischen ihnen definiert.- PostScript. Datenfelder werden gemäß dem PostScript-Format definiert.- RTF. Datenfelder werden gemäß dem RTF-Format definiert.- SGML. Datenfelder werden gemäß dem SGML-Format definiert.- SpaceDelimited. Datenfelder sind durch Leerstellen getrennt.- TabDelimited. Datenfelder sind durch Tabulatorzeichen getrennt. <p>Weitere Informationen hierzu finden Sie unter "Delimiter-Komponenten: Referenz" auf Seite 185. Standardwert ist SGML.</p>
name	<p>Eine beschreibende Kennzeichnung für die Komponente. Diese Kennzeichnung wird in der Protokolldatei sowie der Ereignisansicht angezeigt. Ermitteln Sie anhand der Eigenschaft name, welche Komponente das Ereignis verursacht hat.</p>
pre_processor	<p>Definiert einen Format-Präprozessor, der die Eingabe nach einem beliebigen von Ihnen für die pre_processor-Eigenschaft der example_source definierten Dokumentprozessor verarbeitet. Sie können eine der folgenden Optionen auswählen:</p> <ul style="list-style-type: none">- HtmlProcessor. Wandelt alle Kombinationen aus Tabulatorzeichen, Leerzeichen oder Zeilenwechselzeichen in ein einzelnes Leerzeichen um. Er ist nicht auf HTML-Dokumente beschränkt.- RtfProcessor. Normalisiert RTF-Dateien. <p>Standardwert ist HtmlProcessor.</p>
remark	<p>Ein benutzerdefinierter Kommentar, der den Zweck oder die Aktion der Komponente beschreibt.</p>

RtfFormat

Das **RtfFormat**-Format definiert das Format von RTF-Dateien.

Die nachstehende Tabelle beschreibt die Eigenschaften des **RtfFormat**-Formats:

Eigenschaft	Beschreibung
default_Transformers	Definiert eine Liste von Transformern, die der Parser auf die Ausgabe der einzelnen Content-Anker anwendet. Standardwert ist die folgende Liste von Transformern: <ul style="list-style-type: none">- RtfToASCII. Entfernt RTF-Steuerwörter aus der Ausgabe.- RemoveRtfFormatting. Entfernt RTF-Formatierungsanweisungen aus dem Text.- HtmlProcessor. Wandelt alle Kombinationen aus Tabulatorzeichen, Leerzeichen oder Zeilenwechselzeichen in ein einzelnes Leerzeichen um.- RemoveMarginSpace. Entfernt vorangestellte und angehängte Leerzeichen.
delimiters	Definiert die Informationsstruktur im Dokument. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none">- CommaDelimited. Datenfelder sind durch Kommas getrennt.- DelimiterHierarchy. Datenfelder sind getrennt oder umgeben von Textzeichen.- HL7. Datenfelder werden gemäß HL7-Standard voneinander getrennt.- Positionsbasiert. Datenfelder werden durch die Anzahl der Zeichen zwischen ihnen definiert.- PostScript. Datenfelder werden gemäß dem PostScript-Format definiert.- RTF. Datenfelder werden gemäß dem RTF-Format definiert.- SGML. Datenfelder werden gemäß dem SGML-Format definiert.- SpaceDelimited. Datenfelder sind durch Leerstellen getrennt.- TabDelimited. Datenfelder sind durch Tabulatorzeichen getrennt. Weitere Informationen hierzu finden Sie unter "Delimiter-Komponenten: Referenz" auf Seite 185 . Standardwert ist "RTF".
name	Eine beschreibende Kennzeichnung für die Komponente. Diese Kennzeichnung wird in der Protokolldatei sowie der Ereignisansicht angezeigt. Ermitteln Sie anhand der Eigenschaft name , welche Komponente das Ereignis verursacht hat.
pre_processor	Definiert einen Format-Präprozessor, der die Eingabe nach einem beliebigen von Ihnen für die pre_processor -Eigenschaft der example_source definierten Dokumentprozessor verarbeitet. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none">- HtmlProcessor. Wandelt alle Kombinationen aus Tabulatorzeichen, Leerzeichen oder Zeilenwechselzeichen in ein einzelnes Leerzeichen um. Er ist nicht auf HTML-Dokumente beschränkt.- RtfProcessor. Normalisiert RTF-Dateien. Standardwert ist "RtfProcessor".
remark	Ein benutzerdefinierter Kommentar, der den Zweck oder die Aktion der Komponente beschreibt.

TextFormat

Das Format **TextFormat** definiert das Format der Textdateien.

Verwenden Sie dieses Format in Kombination mit einem Dokumentprozessor, um andere Dokumenttypen zu verarbeiten. Beispielsweise können Sie mit dem Dokumentprozessor **PdfToTxt_4** PDF-Dokumente verarbeiten.

In der folgenden Tabelle werden die Eigenschaften des Formats **TextFormat** beschrieben:

Eigenschaft	Beschreibung
default_transformers	Definiert eine Liste von Transformatern, die der Parser auf die Ausgabe der einzelnen Content-Anker anwendet. Standardwert ist die folgende Liste von Transformatern: <ul style="list-style-type: none">- HtmlProcessor. Wandelt alle Kombinationen aus Tabulatorzeichen, Leerzeichen oder Zeilenwechselzeichen in ein einzelnes Leerzeichen um.- RemoveMarginSpace. Entfernt vorangestellte und angehängte Leerzeichen.
delimiters	Definiert die Informationsstruktur im Dokument. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none">- CommaDelimited. Datenfelder sind durch Kommas getrennt.- DelimiterHierarchy. Datenfelder sind getrennt oder umgeben von Textzeichen.- HL7. Datenfelder werden gemäß HL7-Standard voneinander getrennt.- Positionsbasiert. Datenfelder werden durch die Anzahl der Zeichen zwischen ihnen definiert.- PostScript. Datenfelder werden gemäß dem PostScript-Format definiert.- RTF. Datenfelder werden gemäß dem RTF-Format definiert.- SGML. Datenfelder werden gemäß dem SGML-Format definiert.- SpaceDelimited. Datenfelder sind durch Leerstellen getrennt.- TabDelimited. Datenfelder sind durch Tabulatorzeichen getrennt. Weitere Informationen hierzu finden Sie unter "Delimiter-Komponenten: Referenz" auf Seite 185 . Standardwert ist DelimiterHierarchy.
name	Eine beschreibende Kennzeichnung für die Komponente. Diese Kennzeichnung wird in der Protokolldatei sowie der Ereignisansicht angezeigt. Ermitteln Sie anhand der Eigenschaft name , welche Komponente das Ereignis verursacht hat.
pre_processor	Definiert einen Format-Präprozessor, der die Eingabe nach einem beliebigen von Ihnen für die pre_processor -Eigenschaft der example_source definierten Dokumentprozessor verarbeitet. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none">- HtmlProcessor. Wandelt alle Kombinationen aus Tabulatorzeichen, Leerzeichen oder Zeilenwechselzeichen in ein einzelnes Leerzeichen um. Er ist nicht auf HTML-Dokumente beschränkt.- RtfProcessor. Normalisiert RTF-Dateien. Standardwert ist "Leer".
remark	Ein benutzerdefinierter Kommentar, der den Zweck oder die Aktion der Komponente beschreibt.

XMLFormat

Das Format **XmlFormat** definiert das Format der XML-Dateien.

Der Parser behandelt die XML-Eingabedokumente als gewöhnlichen Text. Sie können Delimiter, Anker und sonstige Komponenten genau wie bei gewöhnlichen Textdokumenten definieren.

In der folgenden Tabelle werden die Eigenschaften des Formats **XmlFormat** beschrieben:

Eigenschaft	Beschreibung
default_transformers	Definiert eine Liste von Transformatoren, die der Parser auf die Ausgabe der einzelnen Content-Anker anwendet. Standardwert ist die folgende Liste von Transformatoren: <ul style="list-style-type: none">- RemoveTags. Entfernt XML-Tags aus der Ausgabe.- HtmlEntitiesToASCII. Wandelt XML-Entitäten in ihre ASCII-Äquivalente um.- HtmlProcessor. Wandelt alle Kombinationen aus Tabulatorzeichen, Leerzeichen oder Zeilenwechselzeichen in ein einzelnes Leerzeichen um.- RemoveMarginSpace. Entfernt vorangestellte und angehängte Leerzeichen.
delimiters	Definiert die Informationsstruktur im Dokument. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none">- CommaDelimited. Datenfelder sind durch Kommas getrennt.- DelimiterHierarchy. Datenfelder sind getrennt oder umgeben von Textzeichen.- HL7. Datenfelder werden gemäß HL7-Standard voneinander getrennt.- Positionsbasiert. Datenfelder werden durch die Anzahl der Zeichen zwischen ihnen definiert.- PostScript. Datenfelder werden gemäß dem PostScript-Format definiert.- RTF. Datenfelder werden gemäß dem RTF-Format definiert.- SGML. Datenfelder werden gemäß dem SGML-Format definiert.- SpaceDelimited. Datenfelder sind durch Leerstellen getrennt.- TabDelimited. Datenfelder sind durch Tabulatorzeichen getrennt. Weitere Informationen hierzu finden Sie unter "Delimiter-Komponenten: Referenz" auf Seite 185 . Standardwert ist SGML.
name	Eine beschreibende Kennzeichnung für die Komponente. Diese Kennzeichnung wird in der Protokolldatei sowie der Ereignisansicht angezeigt. Ermitteln Sie anhand der Eigenschaft name , welche Komponente das Ereignis verursacht hat.
pre_processor	Definiert einen Format-Präprozessor, der die Eingabe nach einem beliebigen von Ihnen für die pre_processor -Eigenschaft der example_source definierten Dokumentprozessor verarbeitet. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none">- HtmlProcessor. Wandelt alle Kombinationen aus Tabulatorzeichen, Leerzeichen oder Zeilenwechselzeichen in ein einzelnes Leerzeichen um. Er ist nicht auf HTML-Dokumente beschränkt.- RtfProcessor. Normalisiert RTF-Dateien. Standardwert ist HtmlProcessor.
remark	Ein benutzerdefinierter Kommentar, der den Zweck oder die Aktion der Komponente beschreibt.

Delimiter-Komponenten: Referenz

Eine Delimiterkomponente definiert eine Hierarchie von Zeichen oder Strings, die die in einem Dokument enthaltenen Informationen (zum Beispiel Zeilenvorschub, Leerstellen, Kommata, horizontale Linien) organisieren. Delimiter können auch mit Hilfe von Platzhalterzeichen definiert werden.

Delimiter gibt es sowohl bei strikt strukturierten Dokumenten, in denen die einzelnen Datenfelder durch Delimiter voneinander getrennt werden, als auch bei HTML-Dokumenten und Dokumenten mit weniger präziser Struktur, die durch Delimiter wie etwa Zeilenvorschub oder syntaktische Auszeichnungen separiert werden. Der Begriff „Delimiter“ trifft auch bei Dokumenten mit absoluter Positionierung zu, in denen sich die Datenfelder in festen Abständen voneinander befinden.

Der Parser verwendet die Delimiter, um die Suchkriterien für `Inhalt-Anker` zu bestimmen, die mit der Option `LearnByExample` konfiguriert wurden.

Beispiel: Angenommen, Sie konfigurieren ein Format mit der Delimiterkomponente `TabDelimited`. Dadurch wird eine Hierarchie festgelegt, in der die folgenden Zeichen als Delimiter verwendet werden:

```
Newline
Tab
```

Sie können nun einen `Content-Anker` definieren, der sich in der Beispielquelle zwei Tabulatorzeichen hinter dem vorhergehenden `Marker-Anker` befindet:

```
MARKER<tab>abc<tab>CONTENT
```

Beim Verarbeiten eines Quelldokuments sucht ein Parser diesen `Content` zwei Tabulatorzeichen nach dem `Marker`.

Sie können zum Beispiel aber auch einen `Content-Anker` definieren, der sich in der Beispielquelle drei Zeilenvorschübe und einen Tabulator hinter einem `Marker-Anker` befindet:

```
MARKER
abc<tab>de
fghi<tab>jkl<tab>mnop
pqrst<tab>CONTENT
```

Die Tabulatoren in den dazwischen liegenden Zeilen werden ignoriert, da die Zeilenvorschübe in der Hierarchie höher stehen.

Viele Delimiterkomponenten, zum Beispiel `TabDelimited` oder `CommaDelimited`, enthalten eine vordefinierte Hierarchie von Delimitern, die Sie nach Bedarf bearbeiten können.

Die Komponente `DelimiterHierarchy` hat jedoch keine vordefinierte Hierarchie. Sie können beliebige Delimiter hinzufügen.

CommaDelimited

Die Delimiterkomponente **`CommaDelimited`** definiert die folgende Delimiter-Hierarchie:

```
Newline
Comma
```

Verwenden Sie **`CommaDelimited`**, wenn jede Zeile einer Textdatei einen Datensatz und jeder Datensatz Datenfelder enthält, die durch Kommata voneinander getrennt sind.

Sie können weitere Delimiter hinzufügen oder die vordefinierte Hierarchie bearbeiten. Gehen Sie genau so vor wie beim Bearbeiten der Komponente **`DelimiterHierarchy`**.

Beispiel

In einem Beispiel-Quelldokument folgt ein **`Content-Anker`** zwei Zeilen nach einem **`Marker-Anker`**. In der dritten Zeile stehen vor dem **`Content-Anker`** drei Kommata (und weiterer Text):

```
MARKER
abcdef, ghij
abc, def,ghi,CONTENT
```

Wenn Sie die Komponente **`CommaDelimited`** zuweisen, lernt der Parser aus der Beispielquelle, dass der **`Content-Anker`** immer zwei Zeilenvorschübe und drei Kommata nach dem **`Marker-Anker`** folgt. Dadurch findet der Parser in einem anderen Quelldokument auch den folgenden **`Content-Anker`**:

```
MARKER
    xyz, uvw, rst
,, ,CONTENT
```

Delimiter

Die Teilkomponente **Delimiter** definiert ein Delimiterzeichen oder einen Delimiterstring, durch die Anker voneinander getrennt werden. **Delimiter**-Teilkomponenten können in eine Delimiterhierarchie eingefügt werden.

Die nachstehende Tabelle beschreibt die Eigenschaften der Teilkomponente **Delimiter**:

Eigenschaft	Beschreibung
search	Definiert den Delimiter. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none">- <code>NewlineSearch</code>. Der Delimiter ist ein Zeilenvorschub.- <code>PatternSearch</code>. Der Delimiter wird durch einen regulären Ausdruck definiert.- <code>TextSearch</code>. Der Delimiter ist ein expliziter String oder ein String, der dynamisch aus dem Quelldokument abgerufen wird. Weitere Informationen hierzu finden Sie unter "Suchkomponenten: Referenz" auf Seite 248 .

Beispiel

Die Komponente `TabDelimited` enthält zwei **Delimiter**-Teilkomponenten. Die erste definiert mit `NewlineSearch` das Zeilenvorschubzeichen als Delimiter. Die zweite definiert mit `TextSearch` das Tabulatorzeichen als Delimiter. Der Tabulator wird grafisch durch das Zeichen « dargestellt.

Die Komponente `SpaceDelimited` enthält ebenfalls zwei **Delimiter**-Teilkomponenten. Die erste ist identisch mit der von `TabDelimited`. Die zweite definiert mit `PatternSearch` einen beliebigen, aus mindestens einem Leerzeichen bestehenden String als Delimiter. Der reguläre Ausdruck `[]+` steht für „ein oder mehr Leerzeichen“. Beachten Sie die Leerstelle zwischen den eckigen Klammern.

DelimiterHierarchy

Mit der Delimiterkomponente **DelimiterHierarchy** können Sie eine eigene Delimiterhierarchie definieren.

In **DelimiterHierarchy** können Sie beliebig viele **Delimiter**- oder **EnclosingDelimiters**-Komponenten verschachteln.

Beispiel

Die Anker in einem Beispiel-Quelldokument sind durch Kommata voneinander getrennt und in eckige Klammern eingeschlossen:

```
MARKER,, [CONTENT]
```

Dafür können Sie eine **DelimiterHierarchy** mit folgendem Inhalt definieren:

```
comma //definiert als Delimiterkomponente  
[]    //definiert als EnclosingDelimiters-Komponente
```

Anhand dieses Beispiels lernt der Parser, dass der **Content**-Anker zwei Kommata nach dem **Marker** folgt und in eckige Klammern eingeschlossen ist. Dadurch findet der Parser in einem anderen Quelldokument auch den folgenden **Content**-Anker:

```
MARKER,abc,def[CONTENT]
```

Online-Beispiel

Ein Online-Beispiel finden Sie unter `samples\Projects\EDI\EDI.cmw`. In diesem Beispiel wird anhand der **DelimiterHierarchy** das Zeilenvorschubzeichen und das Sternchen (*) als Delimiter für ein EDI-Quelldokument definiert.

EnclosingDelimiters

Die Teilkomponente **EnclosingDelimiters** definiert zwei Delimiter oder zwei Delimiter-Strings, die einen Anker umschließen. **EnclosingDelimiters**-Teilkomponenten können unter eine Delimiterhierarchie eingefügt werden.

Sie können diese Komponente verwenden, um die Delimiter in Form geschweifter Klammern ({}) zu definieren, die Blöcke von C-Programmcode umschließen.

Die nachstehende Tabelle beschreibt die Eigenschaften der Teilkomponente **EnclosingDelimiters**:

Eigenschaft	Beschreibung
opening	Definiert den öffnenden Delimiter.
closing	Definiert den schließenden Delimiter
escape_sequence	Definiert ein Präfix, das bewirkt, dass der Parser eine Instanz des öffnenden oder schließenden Delimiters im Quelldokument ignoriert.

HL7

Die Komponente **HL7**-Delimiter definiert die folgende Hierarchie von Delimitern für das Parsing von HL7-Nachrichten:

```
newline  
vertical bar (|)  
caret (^) or tab
```

Sie können weitere Delimiter hinzufügen oder die vordefinierte Hierarchie bearbeiten. Dabei gehen Sie genauso vor wie bei der Komponente **DelimiterHierarchy**.

Gemäß dem Nachrichtenstandard HL7 können von Nachrichten eigene Delimiter definiert werden. Sie können die Deklaration der Delimiter einer HL7-Nachricht parsen und eine dynamische Delimiterdefinition erzeugen. Gehen sie dazu wie folgt vor:

1. Rufen Sie mit Hilfe von **Content**-Ankern die Delimiterzeichen aus dem Header der HL7-Nachricht ab. Speichern Sie die Zeichen in Variablen.
2. Fügen Sie unter der **HL7**-Komponente die gewünschten **Delimiter**-Komponenten hinzu.
3. Weisen Sie jeder **Delimiter**-Komponente den Wert **TextSearch** zu.
4. Weisen Sie unter der Komponente **TextSearch** der Eigenschaft **text** eine der Variablen zu.

Positional

Die Delimiterkomponente **Positional** bewirkt, dass der Parser Content-Anker findet, indem die Zeichen vom Beginn jedes Suchbereichs gezählt werden. Weitere Informationen zum Suchbereich finden Sie unter ["Überblick über Anker" auf Seite 205](#).

Beispiel

Angenommen, in einem Beispiel-Quelldokument folgt ein **Content**-Anker fünf Zeichen (einschließlich Leerzeichen, Tabulatoren usw.) hinter einem **Marker**-Anker:

```
MARKERab cdCONTENTefg
```

Wenn Sie die Komponente **Positional** zuweisen, lernt der Parser aus der Beispielquelle, dass der Anker **Content** immer fünf Zeichen nach dem **Marker** steht und sieben Zeichen lang ist. Dadurch findet er in einem anderen Quelldokument dann auch den folgenden **Content**-Anker:

```
MARKERd<tab>cbaCONTENTzy,xwv
```

Positionsparsen in Kombination mit Delimitern

Der Komponente **Positional** können keine Delimiter hinzugefügt werden.

Manchmal ist es jedoch wünschenswert, einen Parser zu definieren, der einige Anker anhand von Delimitern findet und für andere Anker eine Positionsdefinition verwendet. Hierfür müssen Sie eine der anderen Delimiter-Komponenten auswählen. Verwenden Sie nicht **Positional**. Um den Ort eines Ankers anhand seiner Position zu definieren, können Sie in den Ankereigenschaften die Option **OffsetSearch** zuweisen.

PostScript

Die Delimiterkomponente **PostScript** definiert eine Delimiterhierarchie für das Parsen von Adobe PostScript-Dokumenten.

Die Delimiterhierarchie der **PostScript**-Komponente kann nicht geändert werden.

RTF

Die Delimiterkomponente **RTF** definiert eine Delimiter-Hierarchie für das Parsing von RTF-Dokumenten.

Die Delimiterhierarchie der **RTF**-Komponente kann nicht geändert werden.

SGML

Die Delimiterkomponente **SGML** definiert, welche Delimiterhierarchie für das Parsen von SGML-, HTML- und XML-Dokumenten angewendet werden soll.

Die Delimiterhierarchie der Komponente **SGML** kann nicht bearbeitet werden.

SpaceDelimited

Die Delimiterkomponente **SpaceDelimited** definiert die folgende Delimiterhierarchie:

```
Newline
String of one or more space characters
```

SpaceDelimited ist zum Beispiel dann sinnvoll, wenn jede Zeile einer Textdatei einen Datensatz und jeder Datensatz Datenfelder enthält, die durch Leerzeichen voneinander getrennt sind.

Sie können weitere Delimiter hinzufügen oder die vordefinierte Hierarchie bearbeiten. Dabei gehen Sie genauso vor wie bei der Komponente **DelimiterHierarchy**.

Beispiel

In einem Beispiel-Quelldokument folgt ein **Content**-Anker zwei Zeilen nach einem **Marker**-Anker. In der dritten Zeile stehen vor dem **Content**-Anker zwei Leerzeichen sowie ein String mit mehreren Leerzeichen:

```
MARKER
abcdef
abc def ghi          CONTENT
```

Wenn Sie die Komponente **SpaceDelimited** zuweisen, lernt der Parser anhand der Beispielquelle, dass der **Content**-Anker immer zwei Zeilen und drei Leerzeichen-Zeichenfolgen nach dem **Marker** steht. Dadurch findet der Parser in einem anderen Quelldokument auch den folgenden **Content**-Anker:

```
MARKER
    xyz
ghi      def abc CONTENT
```

TabDelimited

Die Delimiterkomponente **TabDelimited** definiert die folgende Delimiterhierarchie:

```
Newline
Tab
```

TabDelimited ist dann sinnvoll, wenn jede Zeile einer Textdatei einen Datensatz und jeder Datensatz Datenfelder enthält, die durch Tabulatoren voneinander getrennt sind.

Sie können weitere Delimiter hinzufügen oder die vordefinierte Hierarchie bearbeiten. Dabei gehen Sie genauso vor wie bei der Komponente **DelimiterHierarchy**.

Beispiel

In einem Beispiel-Quelldokument folgt ein **Content**-Anker zwei Zeilen nach einem **Marker**-Anker. In der dritten Zeile stehen vor dem **Content**-Anker drei Tabulatoren sowie weiterer Text:

```
MARKER
abcdef
abc<tab> de,f<tab>ghi<tab>CONTENT
```

Wenn Sie die Komponente **TabDelimited** zuweisen, lernt der Parser anhand der Beispielquelle, dass der **Content**-Anker immer zwei Zeilen und drei Tabulatoren nach dem **Marker** steht. Dadurch findet der Parser in einem anderen Quelldokument auch den folgenden **Content**-Anker:

```
MARKER
    xyz
<tab><tab><tab>CONTENT
```

Formatpräprozessor-Komponenten: Referenz

Die folgende Liste beschreibt die Unterschiede zwischen Formatpräprozessoren und Dokumentprozessoren:

- Einen Dokumentprozessor können Sie der Eigenschaft **pre_processor** eines Eingabeports zuweisen, die sich unter der Eigenschaft **example_source** oder **sources_to_extract** eines Parsers befindet. Einen Formatpräprozessor können Sie nur der Eigenschaft **pre_processor** eines Formats zuweisen.
- Ein Dokumentprozessor wird mit dem Quelldokument vor allen anderen Operationen ausgeführt.
- Ein Formatpräprozessor wird mit dem Text ausgeführt, bevor er nach Anker sucht. Die Ausgabe des Formatpräprozessors wird nicht angezeigt.

Weitere Informationen hierzu finden Sie unter ["Überblick über Dokumentprozessoren" auf Seite 163](#).

HtmlProcessor

Der Formatpräprozessor **HtmlProcessor** normalisiert Leerräume entsprechend den HTML-Konventionen. Er ist auch als Transformer verfügbar. Er wandelt Kombinationen von Tabulatoren, Zeilenumbrüchen und Leerzeichen in ein einziges Leerzeichen um.

Mit diesem Präprozessor können Sie in jedem beliebigen Texttyp Leerräume normalisieren. Er ist nicht auf HTML-Dokumente beschränkt.

RtfProcessor

Der Formatpräprozessor **RtfProcessor** normalisiert den Code von RTF-Dateien.

KAPITEL 15

Datenbehälter

Dieses Kapitel umfasst die folgenden Themen:

- [Übersicht über Datenbehälter, 192](#)
- [XML-Schemata, 192](#)
- [Zuordnen von Anker mit Hilfe eines Schemas, 195](#)
- [Erzeugen gültigen XML-Codes, 196](#)
- [Variablen, 198](#)
- [Die Komponente Variable: Referenz, 202](#)
- [Mehrfachinstanz-Datenbehälter, 202](#)

Übersicht über Datenbehälter

Ein Datenbehälter ist ein Objekt, das einem der folgenden Typen entspricht:

- Ein XML-Element
- Ein XML-Attribut
- Eine Variable

XML-Elemente und XML-Attribute werden üblicherweise für permanente Speicherung verwendet. Ein Parser speichert zum Beispiel seine Ausgabe in diesen Datenbehältertypen.

Variablen werden für temporäre Speicherung verwendet. Ein Parser kann zum Beispiel Daten, die er aus dem Quelldokument extrahiert, in einer Variablen speichern. Vor Erzeugung der Ausgabe kann er die Daten weiter verarbeiten.

Jeder Datenbehälter hat einen Datentyp. Bei Elementen und Attributen werden die Datenbehälter gemäß einem von Ihnen bereit zu stellenden XML-Schema definiert. Variablen werden dagegen nach einem internen Schema definiert, das Sie durch Hinzufügen benutzerdefinierter Variablen anpassen können.

XML-Schemata

Wenn Sie einen Parser, einen Serializer, eine XMap oder einen Mapper erstellen, müssen Sie mindestens ein XML-Schema bereitstellen, das die Struktur der XML definiert. Dieses Schema legt die Elemente und Attribute fest, die von der Umwandlung verwendet werden können.

Fügen Sie das Schema zum Modellrepository hinzu. Danach lässt sich der Inhalt eines Dokuments den im Schema definierten Elementen und Attributen zuordnen.

Schemacodierung

Speichern Sie das Schema in einer der unterstützten Eingabecodierungen.

Die Schemacodierung muss mit der Arbeitscodierung kompatibel sein, die für den IntelliScript-Editor verwendet wird. Das bedeutet:

- Die Schemacodierung ist mit der Arbeitscodierung identisch, oder
- für jedes Zeichen im Schema gibt es eine Entsprechung in der Arbeitscodierung. Beispiel: Wird für das Schema die Codierung UTF-8 und als Arbeitscodierung Windows-1252 verwendet, darf das Schema keine Unicode-Zeichen enthalten, für die es in Windows-1252 keine Entsprechung gibt.

Wenn Sie ein extern gespeichertes Schema zu einem Projekt hinzufügen, übersetzt das Skript die Projektkopie des Schemas in die Arbeitscodierung.

Einbezogene Schemadateien

Ein Schema kann zusätzliche Schemadateien referenzieren. Dadurch können Sie ein umfangreiches Schema in mehrere Dateien zerlegen.

Namensräume

Sofern Sie mit XML-Namensräumen arbeiten möchten, müssen Sie dem Attribut **targetNamespace** des Schemas einen Wert zuweisen. Sie können das dem Namensraum zugewiesene Alias bearbeiten.

Sie können nicht zwei Schemata hinzufügen, die ein leeres Alias für zwei Schemata verwenden, oder zwei Schemata, die dasselbe Alias für verschiedene Namensbereiche verwenden.

Gemischter Inhalt

Elemente können Zeichendaten und geschachtelte Elemente enthalten. Sie können das Attribut **mixed** in einem Schema verwenden.

Das Skript unterscheidet zwischen Zeichendaten vor und nach den einzelnen Elementen. Weitere Informationen hierzu finden Sie unter ["Zuordnen gemischten Inhalts" auf Seite 195](#).

Nicht unterstützte Schema-Funktionen

Die aktuelle Version unterstützt einige Schema-Funktionen nur eingeschränkt. In der folgenden Tabelle werden die bekannten Einschränkungen aufgeführt:

Funktion	Einschränkung
Vorgaben für die Eindeutigkeit	Die Elemente unique , key und keyref werden ignoriert. Das Ereignisprotokoll enthält eine Warnung.
Standardwerte für Elemente gemischten Typs	Die Standardwerte werden vom Skript ignoriert. Das Ereignisprotokoll enthält eine Warnung.

Funktion	Einschränkung
Standarddatentyp	Wenn der Typ eines Elements nicht definiert ist, wird es vom Skript als <code>xs:string</code> verarbeitet. Das Ereignisprotokoll enthält eine Warnung. Sie können den Standardtyp in <code>xs:anyType</code> ändern.
Reguläre Ausdrücke	Es sind geringe Unterschiede zwischen dem Prozessor für reguläre Ausdrücke und dem Schemastandard vorhanden.
Folge, die mehrere Elemente mit demselben Namen definiert	Wenn eine <code>xs:sequence</code> mehrere <code>xs:element</code> -Definitionen enthält, die denselben Namen aufweisen, verarbeitet das Skript nur das erste <code>xs:element</code> . Das Ereignisprotokoll enthält eine Warnung. Um das Problem zu beheben, betten Sie jedes <code>xs:element</code> in eine unabhängige <code>xs:sequence</code> ein.
Minimale und maximale Datumsangaben	Wenn eine Facette einen minimalen oder maximalen Wert für ein <code>xs:date</code> -Element definiert, schlägt die Umwandlung fehl.
Validierungsoptionen „lax“ oder „skip“	In einem <code>xs:any</code> - oder einem <code>xs:anyAttribute</code> -Element ignoriert das Skript den processContents -Wert lax oder skip . Es verhält sich so, als ob der Wert strict lauten würde.
Substitutionsgruppe	Das Skript lässt eine substitutionGroup zu, selbst wenn Substitutionen durch ein block - oder blockDefault -Attribut untersagt sind.
XSI-Typ	Das Skript lässt ein <code>xsi:type</code> -Attribut zu, selbst wenn ein block -Attribut dies untersagt.
Integrierte Typen	Einige integrierten Typen weisen kein ordnungsgemäßes Muster auf, dies ist beispielsweise der Fall, wenn sie Zeichen enthalten, die größer als ASCII 127 sind.
Substitutionsgruppe ohne Typ	Das Skript kann fehlschlagen, wenn eine Substitutionsgruppe keinen Typ aufweist.
Leerer Namespace	Wenn der Namespace leer ist, fügt das Skript allen Elementen in der Quelldatei einen Alias hinzu, jedoch wird der Alias nicht im Locator angezeigt und der Locator schlägt fehl.
Liste	Das Skript liest eine durch Leerzeichen getrennte <code>xs:list</code> als ein einzelnes Element. Möglicherweise schlägt der Lesevorgang fehl, wenn die Länge den festgelegten Grenzwert für einzelne Elemente in der Liste überschreitet.
Typen „float“ und „double“	Die Typen <code>xs:float</code> und <code>xs:double</code> akzeptieren keine gültigen Werte von INF , -INF oder NaN .
Element mit festen und gemischten Attributen	Das Skript liest nicht alle Teile eines Elements, das sowohl feste als auch gemischte Attribute aufweist.
max_occurs=0	Das Skript erstellt eine Ausgabe, selbst wenn <code>max_occurs=0</code> festgelegt ist.
Token	Das Skript führt kein Parsing für <code>xs:token</code> aus, das Tabulatoren, Wagenrückläufe oder Zeilenvorschübe enthält.
Normalisierter String	Das Skript lädt keine XML, wenn der <code>xs:normalizedString</code> Tabulatoren, Wagenrückläufe oder Zeilenvorschübe enthält.

Genauigkeit numerischer Daten

Das Skript speichert Daten der Typen `xs:decimal` und `xs:float` als Strings, um die Gesamtstellenanzahl der Daten beizubehalten.

In Berechnungen konvertiert das Skript Dezimal- und Gleitkommadaten in Double-Precision-Gleitkommadaten und rundet das Ergebnis auf 15 Dezimalstellen. Dies bedeutet, dass bei Dezimaldaten die Genauigkeit geringfügig abnimmt. Beispielsweise wird das Resultat von `xs:decimal 5,28 * 1` eventuell als `5,280000000000001` angezeigt.

Das Skript normalisiert `xs:decimal`-Werte. Beispielsweise wird `0004` als `4`, `-0` als `0` und `1,200` als `1,2` gespeichert.

Zuordnen von Ankern mit Hilfe eines Schemas

Beim Definieren eines Parsers ordnen Sie `Content`-Ankern Ausgabedatenbehälter zu. Beim Definieren eines Serializers ordnen Sie `ContentSerializer`-Serialisierungsankern Eingabedatenbehälter zu.

IntelliScript-Darstellung der Datenbehälter

Im Skript werden Datenbehälter durch einen modifizierten XPath-Ausdruck dargestellt:

```
data_holder = /Person/*s/Name/*s/First
```

Um diesen Wert zu ändern, wählen Sie die Eigenschaft **data_holder** aus und drücken Sie die **Eingabetaste**. Damit wird ein Dialogfeld zum Auswählen des **XPath** geöffnet, in dem Sie den neuen Wert auswählen können.

Die XPath-Syntax weicht ein wenig von der XPath-Standardsyntax ab. Diese lautet in diesem Fall: `Person/Name/First`. Das Skript fügt die Angaben `*s`, `*c` und `*a` ein, die für die Schemabegriffe **sequence** (Folge), **choice** (Auswahl) und **all** (alle) stehen. Die Änderungen lösen Zweideutigkeiten, wenn das Skript das Schema verwendet, um die XML-Ausgabe zu erstellen.

Zuordnen gemischten Inhalts

Wenn das Schema gemischte Inhalte unterstützt, hat jedes Element einen vorangehenden und einen nachfolgenden Datenbehälter (**before** und **after**). Betrachten Sie zum Beispiel folgenden gemischten Inhalt:

```
<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>
```

Diese Struktur enthält Datenbehälter an den folgenden Orten:

- unmittelbar hinter dem Tag `<Deal>`, vor allen Teilelementen
- vor dem Element `Price`
- im Element `Price`
- hinter dem Element `Price`
- vor dem Element `Partner`

- in den Elementen `Partner/Name` und `Partner/ID`
- hinter dem Element `Partner`
- unmittelbar vor dem Tag `</Deal>`, nach allen Teilelementen

Sie können den Text `We are pleased to offer you a price of` dem Datenbehälter vor dem Element `Price` zuordnen. `„dollars. “` lässt sich dem Datenbehälter hinter `Price` und `„This is a special price for “` dem Datenbehälter vor `Partner` zuordnen.

Das folgende Beispiel zeigt gemischten Inhalt:

```
data_holder = /Deal/*s/Price/$text_before
```

Zuordnen von XSI-Typen

Ein Schema kann abgeleitete Datentypen definieren, die anstelle eines Basistyps verwendet werden können. In solchen Fällen kann ein XML-Dokument den tatsächlichen Datentypen eines Elements durch Angabe des Attributs `xsi:type` definieren.

Beispielsweise definiert ein Schema das Element `Person` mit dem Typ `PersonT1` und mit String-Inhalt. Es definiert einen Typen namens `PersonT2`, der `PersonT1` erweitert, indem das Attribut `Id` hinzugefügt wird. Im Folgenden werden gültige `Person`-Elemente aufgeführt:

```
<!-- base type PersonT1 -->
<Person>Ron Lehrer</Person>

<!-- derived type PersonT2 -->
<Person Id="547329876" xsi:type="PersonT2">Ron Lehrer</Person>
```

Das Skript interpretiert `xsi:type`-Attribute in Eingabe-XML-Dokumenten. Es fügt, sofern erforderlich, `xsi:type`-Attribute zu Ausgabe-XML-Dokumenten hinzu.

Wählen Sie je nach den umzuwandelnden Daten den entsprechenden Typ aus. Wenn beispielsweise der Anker `Content` Daten im Element `Person` mit dem Typen `PersonT2` speichern soll, wählen Sie `xsi:type=PersonT2` aus. Die Auswahl wird wie folgt im Skript angezeigt:

```
data_holder=/Person/*c/xsi:type=PersonT2
```

In Fällen, in denen aufgrund des Inhalts entweder der Datenbehälter `PersonT1` oder `PersonT2` erforderlich ist, können Sie den Anker **Alternatives** konfigurieren, der zwei Anker **Content** enthält. Einer der Anker **Content** wird `PersonT1` zugeordnet und der andere `PersonT2`. Weitere Informationen hierzu finden Sie unter [“Alternatives” auf Seite 217](#).

Wenn Sie einen Datenbehälter dem nicht qualifizierten Element `Person` zuordnen, verwendet der Datenbehälter automatisch den Basistypen `PersonT1`. Aus diesem Grund sind die folgenden Zuordnungen äquivalent:

```
data_holder=/Person
data_holder=/Person/*c/xsi:type=PersonT1
```

Erzeugen gültigen XML-Codes

Das Skript erzeugt XML-Code, der gemäß dem von Ihnen definierten Ausgabeschema gültig ist.

Das Schema wird bei der Erzeugung des XML-Dokuments als Richtlinie verwendet. Das Schema wird also bereits während der Erzeugung angewendet, nicht erst danach. Dieses Verfahren gewährleistet einen erfolgreichen Verlauf der Umwandlungen. Es sichert die Gültigkeit kontinuierlich im Verlauf der Umwandlung.

Aufgabe von Schemata beim Parsen

Dieser Abschnitt erläutert einige Möglichkeiten, mit denen ein Parser mithilfe des Schemas sicherstellen kann, dass die Ausgabe gültiges XML ist.

Das Verhalten wird anhand von Beispielen veranschaulicht.

Elementfolge

Wenn das Skript einen Parser ausführt, organisiert es die Ausgabe in der Reihenfolge, die das Schema vorgibt.

Beispielsweise kann ein Schema vorgeben, dass ein Element **LastName** vor einem Element **FirstName** steht. Das Skript erzeugt die Ausgabe an den vom Schema definierten Positionen, und das selbst dann, wenn die Anker, die die Ausgabe hervorbringen, in der umgekehrten Reihenfolge definiert sind.

Anzahl der Instanzen

Ein Parser kann versuchen, mehrere Instanzen eines Elementes in die XML-Ausgabe einzufügen. Das Skript verwendet das Schema, um zu ermitteln, ob neue Instanzen anzuhängen oder bestehende Elemente zu überschreiben sind. Der Parser löscht alle Elemente, die über die vom Schema zugelassene Anzahl hinausgehen, und schreibt Warnhinweise in das Ereignisprotokoll.

Angenommen, ein Schema definiert ein Element ohne Angabe der Attribute **minOccurs** oder **maxOccurs**. Der Standardwert für **minOccurs** und **maxOccurs** ist jeweils 1. Das Element muss also in der Parserausgabe genau einmal vorkommen. Wenn das Element in der Ausgabe fehlt, kann der Parser es hinzufügen.

Weitere Informationen hierzu finden Sie unter ["Mehrfachinstanz-Datenbehälter" auf Seite 202](#).

Fehlende oder leere Elemente

In den Einstellungen für die Datenprozessor-Umwandlung können Sie konfigurieren, ob ein Parser leere Elemente einfügt, um die Kompatibilität mit einem Schema zu gewährleisten.

Datentypen

Das Skript stellt sicher, dass in einem Datenbehälter gespeicherte Texte den erforderlichen Datentyp aufweisen. Wenn zum Beispiel ein **Content**-Anker den String „oranges 5 for a dollar“ abrufen und der Datentyp des Datenbehälters `xs:integer` ist, speichert der Anker nur die Ganzzahl 5 im Datenbehälter.

Weitere Informationen hierzu finden Sie unter ["Eingrenzen der Suchkriterien mit Datentypen" auf Seite 215](#).

Rolle von Schemata bei Serialisierung und Mapping

Ein Serializer oder Mapper prüft, ob seine Eingabe gemäß dem XML-Schema gültig ist. Es gibt zwei Validierungsmodi:

- Teilvalidierung Zwischen dem XML-Quelldokument und dem Schema sind einige Abweichungen zulässig. Standard.
- Strikte Validierung Das XML-Quelldokument muss strikt dem Schema entsprechen.

Um den Grad der Validierung festzulegen, müssen Sie die Eigenschaft **validate_source_document** der Komponente **Serializer** oder **Mapper** zuweisen.

Wenn Sie den strikten Modus verwenden, schlägt der Serializer oder Mapper aufgrund eines Validierungsfehlers fehl. In der Ansicht **Ereignisse** werden die Fehler angezeigt.

Wenn Sie die Teilvalidierung verwenden, wird die Umwandlung trotz bestimmter Validierungsfehler weitergeführt. Kommt ein Element beispielsweise öfter vor als laut Schema zulässig, ignoriert der Serializer normalerweise die zusätzlichen Elemente, bearbeitet die gültigen Elemente und schreibt eine Warnung in das Ereignisprotokoll. Es kann auch ein Element mit einem ungültigen Datentyp ignoriert werden.

Das Skript verwendet den Xerces C XML-Parser, Version 3.1, für die Validierung.

Variablen

Variablen sind temporäre Datenbehälter, die Sie anstelle von XML-Elementen oder -Attributen verwenden können. Variablen sind insbesondere dann nützlich, wenn Sie einen Wert während einer Umwandlung vorübergehend speichern müssen, ihn aber nicht im XML ausgeben möchten.

Angenommen, ein Parser soll zwei **Content**-Anker lesen und deren Werte verketteten. In diesem Fall können Sie jedem **Content**-Anker eine benutzerdefinierte Variable zuordnen, die Variablen anschließend mit einer Aktion verketteten und das Ergebnis in ein XML-Element ausgeben.

Das Skript verwendet außerdem vordefinierte Systemvariablen, um Informationen zu speichern, die bei bestimmten Vorgängen erforderlich sind.

Erstellen einer benutzerdefinierten Variable

1. Fügen Sie auf der globalen Ebene des Skripts eine Komponente **Variable** hinzu.
2. Geben Sie einen Namen für die Variable ein und drücken Sie die **Eingabetaste**.
3. Wählen Sie den Datentyp aus, den die Variable speichern kann.

Sie können einen Standardtyp wie `xs:string` oder `xs:integer` oder einen globalen Typ auswählen, der in einem im Projekt referenzierten Schema definiert ist.

Systemvariablen

In den folgenden Abschnitten werden die Systemvariablen und ihre Verwendungsweise beschrieben.

Variablen für den Zugriff auf Quelldokumente

Einige Systemvariablen speichern Daten, die Aktionen für den Zugriff auf Quelldokumente verwenden. Beispielsweise kann die Aktion **RunParser** die Variable `VarLinkURL` verwenden, die einen Dateipfad enthält.

Die folgende Variable wird im **XmlToDocument**-Prozessor verwendet.

Variable	Beschreibung
VarServiceInfo > <i>ServiceLocation</i>	Der Verzeichnispfad des aktuell ausgeführten Skripts oder Dienstes.

Variablen für den schreibgeschützten Zugriff

Die folgenden Variablen sind schreibgeschützt. Eine Umwandlung kann sie verwenden, um ein Quelldokument mehr als einmal aufzurufen.

Variable	Beschreibung
<i>VarRequestedURL</i>	Der Pfad des Quelldokuments, das ein Parser verarbeitet.
<i>VarCurrentURL</i>	Der Pfad des aktuellen Dokuments, das ein Parser verarbeitet. Normalerweise ist diese Variable identisch mit VarRequestedURL . Wenn der Parser mit bestimmten Präprozessoren konfiguriert wird, kann VarCurrentURL auf eine temporäre Datei statt auf das ursprüngliche Quelldokument verweisen. VarRequestedURL verweist immer auf das Quelldokument.
<i>VarCurrentPost</i>	Die Formulardaten, die ein Parser übermittelt hat, um die aktuelle Seite abzurufen.

Schreibgeschützte Variablen für die Systemzeit

VarSystem ist eine schreibgeschützte Variable, die Systeminformationen zurückgibt. Sie stellt eine Struktur mit mehreren geschachtelten Variablen dar.

Variable	Beschreibung
VarSystem > ExecStartTime > <i>Year</i>	Jahr des Beginns der Umwandlung
VarSystem > ExecStartTime > <i>Month</i>	Monat (numerische Angabe)
VarSystem > ExecStartTime > <i>MonthName</i>	Monatsname
VarSystem > ExecStartTime > <i>Day</i>	Tag des Monats
VarSystem > ExecStartTime > <i>DayName</i>	Tag der Woche
VarSystem > ExecStartTime > <i>Hour</i>	Stunde
VarSystem > ExecStartTime > <i>Minute</i>	Minute
VarSystem > ExecStartTime > <i>Second</i>	Sekunde
VarSystem > ExecStartTime > <i>Millisecond</i>	Millisekunde

Mithilfe der Variablen **VarSystem** können Sie einen Zeitstempel in eine Umwandlung einfügen.

Bei der Fehlerbehandlung verwendete Variablen

VarLastFailure speichert neueste, fehlgeschlagene Komponente der Umwandlung. Beispielsweise kann die Instanz eines Ankers **Marker** gespeichert werden, die den Marker-Text nicht gefunden hat. Sie können eine Komponente so konfigurieren, dass sie bei Auftreten eines Fehlers **VarLastFailure** in ein Benutzerprotokoll schreibt. Weitere Informationen hierzu finden Sie unter ["Fehlerbehandlung" auf Seite 410](#).

Hinweis: Bei der Verwendung von **VarLastFailure** wird der Dienst in einem speziellen Modus ausgeführt, für den ca. die dreifache CPU-Zeit erforderlich ist.

VarServiceInfo speichert den Dienstnamen, die Position des Verzeichnisses für das Benutzerprotokoll und den Dateinamen des Benutzerprotokolls.

VarLastFailure und **VarServiceInfo** sind Strukturen, die die folgenden geschachtelten Variablen enthalten:

Variable	Beschreibung
VarLastFailure > <i>InternalId</i>	Fehlerkennung
VarLastFailure > <i>Text</i>	Beschreibung des Fehlers
VarLastFailure > <i>Location</i>	Position des Fehlers im Skript
VarLastFailure > <i>AnchorName</i>	Name der fehlgeschlagenen Komponente
VarLastFailure > <i>Data</i>	Weitere Informationen zum Fehler
VarServiceInfo > <i>ServiceName</i>	Name des Dienstes
VarServiceInfo > StandardError > <i>StandardErrorDir</i>	Verzeichnispfad des Benutzerprotokolls
VarServiceInfo > StandardError > <i>StandardErrorName</i>	Dateiname des Benutzerprotokolls

Beim strukturierten Parsing verwendete Variablen

VarStructureDetails dient der Nachverfolgung des aktuellen Datensatzes, für den ein **StructureDefinition**-Anker ein Parsing ausführt. Sie enthält die folgenden geschachtelten Variablen:

Variable	Beschreibung
VarStructureDetails / <i>Name</i>	Die Eigenschaft name des Unterelements, des mit dem Datensatz übereinstimmt.
VarStructureDetails / <i>Repetitions</i>	Die Anzahl an Iterationen des Datensatzes.
VarStructureDetails / <i>RecordIndex</i>	Die Indexnummer des Datensatzes in der StructureDefinition -Gesamteingabe.
VarStructureDetails / <i>RecordId</i>	Die Datensatzkennung. Wenn mehrere Kennungen vorhanden sind, enthält die Variable eine durch Kommas getrennte Liste.
VarStructureDetails / <i>InternalPath</i>	Interne Informationen, nicht zur Verwendung.

Weitere Informationen hierzu finden Sie unter ["StructureDefinition" auf Seite 244](#).

In Benachrichtigungen verwendete Variablen

VarNotificationDetails speichert Informationen zur letzten Benachrichtigung, die in einer Umwandlung ausgelöst wurde.

Variable	Beschreibung
VarNotificationDetails > <i>Name</i>	Der Name der Benachrichtigung.
VarNotificationDetails > <i>Path</i>	Der XPath des Datenbehälters, zu dem die Benachrichtigung gehört. Wenn beispielsweise ein Validator eine Benachrichtigung in einem Content -Anker auslöst, entspricht der Path dem Datenbehälter, in dem der Content -Anker seine Ausgabe speichert.
VarNotificationDetails > <i>Value</i>	Der Eingabewert, der die Benachrichtigung ausgelöst hat. Wenn ein Validator die Benachrichtigung auslöst, stellt Value die ungültigen Eingabedaten dar. Wenn von einer Notify -Aktion die Benachrichtigung ausgelöst wird, können Sie die Variable Value in der Konfiguration für Notify festlegen.
VarNotificationDetails > <i>Creator</i>	Die Position im Skript, an der die Benachrichtigung ausgelöst wurde.

Weitere Informationen hierzu finden Sie unter ["Benachrichtigungen" auf Seite 430](#).

Zuordnung von Ankern zu Variablen

Eine Variable wird einem **Content**-Anker auf dieselbe Weise zugeordnet wie jeder andere Datenbehälter.

Schreibgeschützte Systemvariablen dürfen Ankern nicht zugeordnet werden.

Variablen in Aktionen verwenden

Variablen werden häufig als Eingabe für Aktionen verwendet. Sie können Variablen in derselben Weise einsetzen wie andere Datenbehälter. Weitere Informationen hierzu finden Sie unter ["Überblick über die Aktionen" auf Seite 301](#).

Initialisieren von Variablen während der Laufzeit

Sie können die Werte von Variablen auf folgende Weise initialisieren:

- Sie können im Skript die Eigenschaft **initialization** der **Variablen** konfigurieren.

Die Anfangswerte, die Sie auf diese Weise festlegen, werden verwendet, wenn Sie die Umwandlung im Developer Tool oder als Dienst ausführen.

- Eine Anwendung kann die Anfangswerte während der Laufzeit als Dienstparameter an einen Dienst übergeben.

Die Dienstparameter setzen die Eigenschaft **initialization** der Variablen außer Kraft. Wenn im Skript ein Anfangswert angegeben ist und Sie außerdem einen Wert aus einer Anwendung übergeben, wird der letztere Wert verwendet.

Die Komponente Variable: Referenz

Eine **Variable**-Komponente ist eine benutzerdefinierte Variable.

Weitere Informationen zu Systemvariablen finden Sie unter ["Systemvariablen" auf Seite 198](#).

Variable

Eine **Variable**-Komponente ist eine benutzerdefinierte Variable, die Sie in einem Skript verwenden.

Verwenden Sie Variablen für eine temporäre Speicherung in derselben Weise wie XML-Elemente oder -Attribute. Sie können beispielsweise einer Variablen einen **Content**-Anker zuordnen oder eine Variable als Eingabe für eine Aktion verwenden.

Variablen werden auf der globalen Ebene des Skripts angezeigt. Eine Variable kann jeden beliebigen Datentyp aufweisen, der in den mit dem Projekt zugeordneten Schemata definiert ist, einschließlich Standardtypen und benutzerdefinierte Typen. Benutzerdefinierte Typen können einfach oder komplex sein. Eine komplexe Variable ist eine Struktur mit geschachtelten Feldern. Die Initialisierung komplexer Variablen wird nicht unterstützt. Weitere Informationen hierzu finden Sie unter ["Initialisieren von Variablen während der Laufzeit" auf Seite 201](#).

In der folgenden Tabelle werden die Eigenschaften der Komponente **Variable** beschrieben:

Eigenschaft	Beschreibung
initialization	Definiert einen Anfangswert für die Variable. Sie können Variablen initialisieren, die einfache Datentypen aufweisen. Standardwert ist "Leer".
InitialValue	Definiert einen Anfangswert für die Variable. InitialValue weist eine Eigenschaft auf, und zwar value .
list	Legt fest, ob es sich bei der Variablen um eine einfach oder mehrfach vorkommende Variable handelt. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none">- Ausgewählt. Legt eine mehrfachen vorkommende Variable fest.- Gelöscht. Legt eine einzeln vorkommende Variable fest. Die Standardoption lautet „Gelöscht“. Weitere Informationen hierzu finden Sie unter "Mehrfachinstanz-Datenbehälter" auf Seite 202 .
val_type	Definiert den Datentyp, den die Variable speichern kann. Gültige Werte werden im Schema definiert. Der Standardtyp lautet <code>xs:string</code> .
value	Definiert den Anfangswert.

Mehrfachinstanz-Datenbehälter

In einem Schema können Sie mit dem Attribut **maxOccurs** festlegen, wie oft Schwesterelemente in einem XML-Dokument vorkommen dürfen. Auf dieselbe Art können Sie auch eine Variable definieren, die entweder einmal oder mehrmals vorkommen kann. Elemente oder Variablen, die nur einmal auftreten dürfen, werden als Einzelinstanz-Datenbehälter bezeichnet. Dementsprechend werden Elemente oder Variablen, die mehrfach auftreten können, Mehrfachinstanz-Datenbehälter genannt.

Einzel- und Mehrfachvorkommen-Datenbehälter verhalten sich unterschiedlich, wenn das Skript Daten in ihnen speichert. Dies ist beispielsweise beim Zuweisen von Datenbehältern zu **Content**-Ankern der Fall.

- In einem Einzelinstanz-Datenbehälter überschreibt jede Zuordnung die vorherige Zuordnung.
- In einem Mehrfachinstanz-Datenbehälter erzeugt jede Zuordnung eine neue Instanz des Datenbehälters.

Beispiel: Angenommen, in einem Schema ist ein XML-Element namens `FirstName` definiert. Wenn `maxOccurs = 1` ist, handelt es sich um einen Einzelinstanz-Datenbehälter. Ordnet ein Parser dem Element `<FirstName>` mehr als einen **Content**-Anker zu, enthält die Ausgabe nur die letzte Zuordnung.

Stellen Sie sich vor, Sie parsen ein Quelldokument, das eine Liste mit Vornamen darstellt.

```
Jack Jennie Larissa
```

Jeder Name ist ein **Content**-Anker, der dem Element **FirstName** zugeordnet ist. Der Wert von **FirstName** wird durch die Namen überschrieben. Die Ausgabe enthält nur die Zuordnung:

```
<FirstName>Larissa</FirstName>
```

Nehmen wir nun an, dass `maxOccurs = unbounded` ist. In diesem Fall ist **FirstName** ein Mehrfachinstanz-Datenbehälter. Wenn Sie dem Element mehrere **Content**-Anker zuordnen, erzeugt der Parser eine Namensliste. Die Ausgabe sieht folgendermaßen aus:

```
<FirstName>Jack</FirstName>
<FirstName>Jennie</FirstName>
<FirstName>Larissa</FirstName>
```

Dasselbe Prinzip gilt für Variablen. Wenn Sie einer Mehrfachinstanz-Variable mehrere Anker zuordnen, erzeugt jeder Anker eine neue Instanz der Variablen. Dies ist zum Beispiel dann nützlich, wenn Sie die Eingabe für die Aktionen **AppendListItems** und **CombineValues** vorbereiten, die die Instanzen verketteten.

Hinweis: Das hier beschriebene Verhalten geht davon aus, dass der Mehrfachinstanz-Datenbehälter über einen einfachen Datentyp verfügt. Wenn unter bestimmten Umständen ein Typ komplex ist, kann evtl. nicht jeder Anker eine neue Instanz generieren. Zur Kontrolle dieses Verhaltens können Sie einen Locator verwenden. Weitere Informationen hierzu finden Sie unter ["Überblick über Locator, Keys und Indexierung" auf Seite 369](#).

Attribute

Ein XML-Attribut ist immer ein Einzelinstanz-Datenbehälter. Es kann kein Mehrfachinstanz-Datenbehälter sein, da in XML kein Attribut mehr als einmal in demselben Element vorkommen darf.

Ein Attribut kann einen Datentyp in Form einer durch Leerzeichen getrennten Liste haben. Das Attribut `names` im folgenden Element ist ein Beispiel dafür:

```
<Countries names="USA Canada Mexico"/>
```

Das Skript behandelt das Attribut als Einzelvorkommen-Datenbehälter mit einem Listentyp. Weitere Informationen hierzu finden Sie unter ["Eingrenzen der Suchkriterien mit Datentypen" auf Seite 215](#).

Indexierung

Standardmäßig greift das Skript sequenziell auf die einzelnen Vorkommen eines Mehrfachvorkommen-Datenbehälters zu. Mit Hilfe der Indexierungsfunktion können Sie nichtsequenziell auf die Instanzen zugreifen. Weitere Informationen hierzu finden Sie unter ["Überblick über Locator, Keys und Indexierung" auf Seite 369](#).

Löschen von Instanzen

Unter bestimmten Umständen kann es sinnvoll sein, alle vorhandenen Instanzen eines Mehrfachinstanz-Datenbehälters zu löschen und ab Anfang der Liste neue Instanzen zu erstellen. Dies ist beispielsweise dann sinnvoll, wenn Sie eine iterative Struktur parsen und nur die letzte Iteration erhalten bleiben soll. In solchen Fällen können Sie die Instanzen, in denen Daten aus früheren Iterationen gespeichert sind, löschen.

Zu diesem Zweck definieren Sie einen Einzelinstanz-Datenbehälter, der ein geschachteltes Mehrfachinstanz-Element enthält. Bei erneuter Verwendung des Einzelinstanz-Datenbehälters werden die geschachtelten Instanzen gelöscht.

Das folgende Szenario ist ein typisches Beispiel.

1. Fügen Sie folgendes Schema zu einem Projekt hinzu:

```
<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>
```

Das Schema beschreibt einen benutzerdefinierten Datentyp namens **MyListType**. Dieser Datentyp enthält ein geschachteltes Mehrfachinstanz-Element namens **item**.

2. Definieren Sie eine Einzelinstanz-Variable mit dem Namen **MyList** und dem Datentyp **MyListType**.
3. Verwenden Sie die Variable als Ziel einer iterativen Struktur.

Weitere Informationen hierzu finden Sie unter ["Überblick über Locator, Keys und Indexierung" auf Seite 369](#).

Bei der nächsten Iteration wird die einzelne Instanz von **MyList** erneut verwendet. Beim Start der Iteration werden die geschachtelten **item**-Elemente gelöscht. Von den Ankern innerhalb der iterativen Struktur, z. B. **RepeatingGroup**, werden die **item**-Elemente vom Beginn der Liste neu zugeordnet.

Online-Beispiel

Die Vorgehensweise zum Löschen von Mehrfachinstanzen eines Datenbehälters zeigt das folgende Online-Beispiel:

```
samples\Projects\ResetListVariable\ResetListVariable.cmw
```

KAPITEL 16

Anker

Dieses Kapitel umfasst die folgenden Themen:

- [Überblick über Anker, 205](#)
- [Zuordnen von Content-Ankern zu Datenbehältern, 206](#)
- [Definieren von Ankern, 208](#)
- [Standardeigenschaften von Ankern, 209](#)
- [So sucht der Parser Anker, 211](#)
- [Ankerkomponenten: Referenz, 217](#)
- [Suchkomponenten: Referenz, 248](#)
- [Die Unterkomponente Anker: Referenz, 253](#)

Überblick über Anker

Anker sind die Komponenten, mit denen ein Parser an bestimmten Stellen eines Quelldokuments einhaken kann, um Daten zu suchen und in Datenbehältern zu speichern. Sie dienen als Wegweiser, die Sie in das Dokument einfügen und die die Position von Daten anzeigen.

In diesem Kapitel werden die verschiedenen Ankertypen und ihre Verwendung in Parseern erläutert.

Die Anker Marker und Content

Am häufigsten werden die beiden Anker **Marker** und **Content** verwendet. Diese Anker werden oft als Paar verwendet:

- Ein **Marker**-Anker bezeichnet eine Position in einem Dokument.
- Ein **Content**-Anker ruft Text von dieser Position ab.

Stellen Sie sich zum Beispiel einen gedruckten Fragebogen vor. Die erste Zeile eines Fragebogens fragt in der Regel nach dem Vor- und Nachnamen, wobei jeder Beschriftung ein Leerraum folgt, in den die Informationen eingegeben werden. Die Rubriken **Last Name** und **First Name** werden als **Marker**-Anker und die Leerräume als **Content**-Anker bezeichnet. Mit Hilfe der Anker können Sie gezielt auf die Daten zugreifen und sie aus dem Quelldokument extrahieren.

Andere Ankertypen

Neben **Marker**- und **Content**-Ankern sind noch viele andere Typen von Ankern vorhanden, die Sie zum Parsen von Dokumenten einsetzen können. Beispielsweise können Sie mit **Group**- und **RepeatingGroup**-Ankern die

Anordnung der Datenfelder bestimmen. Und mit einem **Alternatives**-Anker geben Sie verschiedene Datentypen an, die an einer bestimmten Stelle im Quelldokument vorkommen können.

Gemeinsame Verwendung von Ankern und Delimitern

Sie können Anker im Beispiel-Quelldokument definieren. Damit der Parser lernt, wie er ein Dokument parsen soll, analysiert er die Anker und die dazwischen liegenden Delimiter. Weitere Informationen zu Delimitern enthält der Abschnitt ["Übersicht über Formate" auf Seite 178](#).

Angenommen, Sie haben für Ihr Dokument ein tabulatorbegrenztes Format angegeben. Die Beispielquelle enthält die folgende Zeile:

```
First name:<tab>Ron
```

wobei `<tab>` ein Tabulatorzeichen darstellt.

Sie können nun `First name:` als einen **Marker**-Anker und `Ron` als einen **Content**-Anker definieren. Der Parser lernt aus diesen Definitionen, dass er im Quelldokument nach der Zeichenfolge `First name:` suchen muss. Anschließend muss er einen einzelnen Tabulator, den Delimiter, überspringen und den danach folgenden Text abrufen.

Nun führen Sie den Parser an einem weiteren Quelldokument aus, das den folgenden Text enthält:

```
First name:<tab>Jack
```

Der Parser findet die Anker wie oben beschrieben und ruft den Text `Jack` ab.

Nehmen wir nun an, dass das Quelldokument die folgende Zeile enthält:

```
First name:<tab>Jack<tab>Age:<tab>34
```

Der Parser ruft weiterhin den Text `Jack` ab und nicht etwa `Jack<tab>Age<tab>34`. Dies liegt daran, dass Sie das Tabulatorzeichen als Delimiter definiert haben. Das Skript hat gelernt, dass der **Content**-Anker nach dem ersten Tabulator beginnt und vor dem zweiten Tabulator endet. Sie müssen also weitere Anker definieren, wenn Sie Jacks Alter (34) abrufen möchten.

Hinweis: Die obigen Beispiele beschreiben ein mögliches Verhalten von Ankern und Delimitern. Anker enthalten bestimmte Eigenschaften, mit denen Sie dieses Verhalten ändern können. Beispielsweise können Sie einen **Content**-Anker definieren, der Tabulatoren auch in einem tabulatorbegrenzten Format ignoriert. Weitere Informationen hierzu finden Sie unter ["So sucht der Parser Anker" auf Seite 211](#).

Zuordnen von Content-Ankern zu Datenbehältern

Der Anker **Content** speichert den Text, den er aus einem Quelldokument abruft, in einem Datenbehälter. Sie können beispielsweise einen **Content**-Anker konfigurieren, der sein Ergebnis in einem XML-Element namens **FirstName** speichert. Wenn dieser **Content**-Anker den Text `Jack` abruft, erzeugt der Parser die folgende Ausgabe:

```
<FirstName>Jack</FirstName>
```

Sie können präziser angeben, dass der Anker den abgerufenen Text im Pfad `/Person/*s/FirstName` speichern soll, der sich auf ein im XML-Schema gespeichertes Element bezieht. Der Parser gibt das folgende Ergebnis aus:

```
<Person>
  <FirstName>Jack</FirstName>
</Person>
```

Nehmen wir nun an, dass im Schema **FirstName** als Attribut des Elements **Person** definiert ist. Hierbei können Sie den **Content**-Anker zu `/Person/@FirstName` zuordnen. In diesem Fall erhalten Sie die folgende Ausgabe:

```
<Person FirstName="Jack" />
```

Das Zuordnungsziel muss immer ein Datenbehälter mit dem passenden Datentyp sein. Es ist zum Beispiel nicht möglich, `Jack` einem XML-Element zuzuordnen, das den Datentyp `xs:integer` hat, oder einem XML-Element, das einen komplexen Datentyp mit geschachtelten Elementen aufweist. Weitere Informationen zu dieser Regel enthält der Abschnitt [“Eingrenzen der Suchkriterien mit Datentypen” auf Seite 215](#).

Zuordnen zu Variablen

Ein Anker kann einem Datenbehälter zugeordnet werden, der ein XML-Element, ein XML-Attribut oder eine Variable ist. Die Zuordnung zu einer Variablen ist dann sinnvoll, wenn Sie die Daten in einem späteren Verarbeitungsschritt verwenden möchten, aber die Rohdaten nicht in die Parserausgabe aufgenommen werden sollen.

Angenommen, Sie möchten mehrere Zahlen aus einem Quelldokument extrahieren und ihre Summe in die XML-Ausgabe einfügen. Die einzelnen Zahlen sollen jedoch nicht ausgegeben werden. Zu diesem Zweck können Sie die **Content**-Anker, welche die Zahlen abrufen, entsprechenden Variablen zuweisen und die Summe mit Hilfe der Aktion `CalculateValue` berechnen und ausgeben lassen.

Ebenso ist auch die Zuordnung zu einer Variablen möglich, die in einem nachfolgenden Anker verwendet wird, zum Beispiel zur Generierung eines dynamischen Suchtextes für einen **Marker**-Anker.

Zuordnen zu mehrfach vorkommenden Datenbehältern

Wenn Sie Einzelinstanz-Datenbehältern **Content**-Anker zuordnen, überschreibt jede Zuweisung des Datenbehälters die vorherige Zuweisung.

Bei der Zuordnung zu einem Mehrfachinstanz-Datenbehälter erzeugt jede Zuordnung eine neue Instanz des Datenbehälters. Wenn zum Beispiel jeder **Content**-Anker den Namen einer Person abrufen, bildet die Ausgabe eine Namensliste:

```
<FirstName>Jack</FirstName>
<FirstName>Jennie</FirstName>
<FirstName>Larissa</FirstName>
```

Weitere Informationen hierzu finden Sie unter [“Mehrfachinstanz-Datenbehälter” auf Seite 202](#).

Zuordnen zu Elementen mit gemischtem Inhalt

Der Begriff „Gemischter Inhalt“ bezieht sich auf ein XML-Element, das Daten und geschachtelte Elemente enthält. Sofern laut Schema ein XML-Element gemischten Inhalt enthalten darf, wird in der Schema-Ansicht ein **vorangestellter** und ein **nachfolgender** Datenbehälter für diese Elemente angezeigt. Auf diese Weise lässt sich ein **Content**-Anker Zeichendaten zuordnen, die sich vor oder hinter einem bestimmten geschachtelten Element befinden. Weitere Informationen hierzu finden Sie unter [“Zuordnen gemischten Inhalts” auf Seite 195](#).

Definieren von Anker

Wenn Sie eine `Parser`-Komponente definieren, müssen Sie eine Folge von Anker hinzufügen. Die Funktion des Parsers besteht darin, die Anker im Quelldokument zu suchen und die Operationen auszuführen, die Sie für die Anker vorgegeben haben.

Wo werden Anker definiert?

Im Skript werden Anker innerhalb eines **Parsers** geschachtelt.

Wenn Sie an der angegebenen Position die **Eingabetaste** drücken, zeigt der IntelliScript-Editor eine Dropdown-Liste mit den Anker (und anderen Komponenten) an, die Sie hinzufügen können.

Nach dem Hinzufügen werden die Anker vom Developer Tool in der Beispielquelle hervorgehoben angezeigt.

Einige Ankertypen können geschachtelte Anker enthalten. Beispielsweise können in **Alternatives**-, **Group**- oder **RepeatingGroup**-Anker weitere Anker geschachtelt werden.

Folge von Anker

Die Folge von Anker muss der Reihenfolge des Textes im Quelldokument entsprechen.

Angenommen, das Quelldokument lautet folgendermaßen:

```
First Name: Ron  
Last Name: Lehrer
```

Wenn Sie `First Name` und `Last Name` als Marker-Anker sowie `Ron` und `Lehrer` als Content-Anker definieren, müssen Sie die Anker in der folgenden Reihenfolge in die Parserkonfiguration einfügen:

Anker	Text im Quelldokument
Marker	First Name
Content	Ron
Marker	Last Name
Content	Lehrer

Ausnahme: Variable Reihenfolge im Quelldokument

In einigen Quelldokumenten ist die Reihenfolge variabel. Nehmen wir an, dass das Quelldokument alternativ eines der folgenden Formate haben kann:

```
First Name: Ron  
Last Name: Lehrer
```

oder

```
Last Name: Lehrer  
First Name: Ron
```

In solchen Fällen können Sie mit der Eigenschaft `marking` den Suchbereich der Anker ändern. Weitere Informationen hierzu finden Sie unter ["So sucht der Parser Anker" auf Seite 211](#).

Hinzufügen von Marker- oder Content-Ankern

Die Anker **Marker** und **Content** können durch Auswählen und Anklicken hinzugefügt werden.

1. Wählen Sie den Ankertext in der Beispielquelldatei aus.
2. Klicken Sie mit der rechten Maustaste auf den ausgewählten Text, und klicken Sie anschließend auf **Marker einfügen** oder **Content einfügen**.
3. Legen Sie die Eigenschaften des Ankers fest.

Definieren eines Ankers

Sie können jede beliebige Art von Anker erstellen, indem Sie das Skript bearbeiten. Dabei gehen Sie genauso vor wie bei der Bearbeitung einer beliebigen anderen Komponente.

1. Wählen Sie an der gewünschten Position für den Anker die Auslassungspunkte (...) aus und drücken Sie die **Eingabetaste**.
2. Wählen Sie den Namen des Ankers aus oder geben Sie ihn ein.
3. Drücken Sie erneut die **Eingabetaste**, um den Namen zu bestätigen.
4. Bearbeiten Sie die Eigenschaften des Ankers.

Standardeigenschaften von Ankern

In der folgenden Tabelle werden die Standardeigenschaften von Ankern beschrieben:

Eigenschaft	Beschreibung
direction	<p>Eine Suchrichtung für den Anker innerhalb des Suchbereichs. Sie können eine der folgenden Optionen auswählen:</p> <ul style="list-style-type: none">- Zurück. Die Suche beginnt am Ende des Suchbereichs und findet die letzte Instanz des Ankers.- Vorwärtssuche. Die Suche beginnt am Anfang des Suchbereichs und findet die erste Instanz des Ankers. <p>Bei einem Marker-Anker können Sie dieses Verhalten mithilfe der Eigenschaft count ändern. Wenn beispielsweise direction = backward und count = 2 ist, findet das Skript die vorletzte Instanz. Standardwert ist "Vorwärtssuche". Weitere Informationen hierzu finden Sie unter "So sucht der Parser Anker" auf Seite 211.</p>
disabled	<p>Legt fest, ob die Komponente und alle zugehörigen untergeordneten Komponenten vom Skript ignoriert werden. Verwenden Sie diese Eigenschaft zum Testen, Debuggen und Ändern eines Skripts. Sie können eine der folgenden Optionen auswählen:</p> <ul style="list-style-type: none">- Ausgewählt. Das Skript ignoriert die Komponente.- Gelöscht. Das Skript wendet die Komponente an. <p>Standardmäßig ist die Eigenschaft deaktiviert.</p>
marking	<p>Legt fest, ob ein Anker als Anfang des Suchbereichs für den danach folgenden Anker verwendet wird. Sie können eine der folgenden Optionen auswählen:</p> <ul style="list-style-type: none">- Anfangsposition. Setzt einen Referenzpunkt vor dem aktuellen Anker.- Endposition. Setzt einen Referenzpunkt hinter dem aktuellen Anker.- Vollständig. Setzt einen Referenzpunkt vor und hinter dem aktuellen Anker.- Keine. Es wird kein Referenzpunkt erstellt. <p>Weitere Informationen hierzu finden Sie unter "So sucht der Parser Anker" auf Seite 211.</p>

Eigenschaft	Beschreibung
name	Eine beschreibende Kennzeichnung für die Komponente. Diese Kennzeichnung wird in der Protokolldatei sowie der Ereignisansicht angezeigt. Ermitteln Sie anhand der Eigenschaft name , welche Komponente das Ereignis verursacht hat.
no_initial_phase	Legt fest, ob das Skript nach geschachtelten Anker in der Hauptphase sucht. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none"> - Gelöscht. Es erfolgt eine Suche nach geschachtelten Anker entsprechend ihren individuellen Eigenschaften. - Ausgewählt. Es erfolgt eine Suche nach geschachtelten Anker in der Hauptphase. Die Standardoption lautet „Gelöscht“.
notifications	Eine Liste mit NotificationHandler -Komponenten, die Benachrichtigungen aus geschachtelten Komponenten verarbeiten. Weitere Informationen hierzu finden Sie unter "Benachrichtigungen" auf Seite 430 .
on_fail	Die Aktion, die beim Ausfall der Komponente durchgeführt wird. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none"> - Gelöscht. Es wird keine Aktion ausgeführt. - CustomLog. Es wird in das Benutzerprotokoll geschrieben. - LogError. Eine Fehlermeldung wird in das Engine-Protokoll geschrieben. - LogInfo. Eine Informationsmeldung wird in das Engine-Protokoll geschrieben. - LogWarning. Eine Warnmeldung wird in das Engine-Protokoll geschrieben. - NotifyFailure. Eine Mitteilung wird gesendet. Die Standardoption lautet „Gelöscht“. Weitere Informationen über die Behandlung von Komponentenfehlern finden Sie in "Fehlerbehandlung" auf Seite 410 .
optional	Legt fest, ob ein Komponentenfehler den Ausfall der übergeordneten Komponente verursacht. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none"> - Ausgewählt. Ein Komponentenfehler führt nicht zum Ausfall der übergeordneten Komponente. - Gelöscht. Ein Komponentenfehler verursacht den Ausfall der übergeordneten Komponente. Standardwert ist "Gelöscht". Weitere Informationen über Komponentenfehler finden Sie in "Fehlerbehandlung" auf Seite 410 .
phase	Legt den Zeitpunkt fest, an dem das Skript die Komponente verarbeitet. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none"> - Anfangsphase. Das Skript verarbeitet die Komponente während der Anfangsphase. - Hauptphase. Das Skript verarbeitet die Komponente während der Hauptphase. - Endphase. Das Skript verarbeitet die Komponente während der Endphase. Weitere Informationen hierzu finden Sie unter "So sucht der Parser Anker" auf Seite 211 .
remark	Ein benutzerdefinierter Kommentar, der den Zweck oder die Aktion der Komponente beschreibt.

Wenn nicht sicher ist, ob ein Anker in einem Quelldokument vorhanden ist, wählen Sie die Eigenschaft **optional** aus. Wenn der Anker nicht vorhanden ist, fährt der **Parser**, in den der Anker geschachtelt ist, mit der Verarbeitung fort.

Ist der Anker in einen **Group**-Anker geschachtelt, wird mithilfe der Eigenschaft „optional“ vermieden, dass die **Group** fehlschlägt. Befindet sich der Anker in einer **RepeatingGroup**, wird mit dieser Eigenschaft vermieden, dass eine Iteration der **RepeatingGroup** fehlschlägt.

So sucht der Parser Anker

Für eine ordnungsgemäße Erstellung eines Parsers sollten Sie wissen, wie das Skript im Parser nach Ankern sucht. In diesem Zusammenhang sind drei Begriffe wichtig:

- Suchphase
- Suchbereich
- Suchkriterien

Dieser Abschnitt erklärt diese Funktionen und beschreibt, wie Sie sie mit Hilfe der Ankereigenschaften steuern.

Suchphasen

Das Skript sucht in drei Phasen nach einer Folge von Ankern:

- Anfangsphase
- Hauptphase
- Endphase

Standardmäßig wird nach den **Marker**-Ankern in der Anfangsphase und den **Content**-Ankern in der Hauptphase gesucht. Somit findet das Skript zunächst die **Marker**-Anker und dann die dazwischen liegenden **Content**-Anker.

Ein Beispiel zur Veranschaulichung: Ein Parser verarbeitet das folgende Quelldokument:

```
First name: Ron      Last name: Lehrer
```

Dabei sind die Anker folgendermaßen mit Standardeigenschaften definiert:

Anker	Text im Quelldokument	Phase
Marker	First name:	Anfangsphase
Content	Ron	Hauptphase
Marker	Last name:	Anfangsphase
Content	Lehrer	Hauptphase

In der Anfangsphase sucht das Skript nach den **Marker**-Ankern:

- Zuerst nach `First name:.`
- Dann nach `Last name:` an einer Position hinter `First name:.`

In der Hauptphase sucht das Skript nach den **Content**-Ankern:

- Zuerst nach dem Anker `Ron` an einer Position zwischen `First name:` und `Last name:.`
- Dann nach dem Anker `Lehrer` an einer Position hinter `Last name:.`

Geschachtelte Phasen

Anker mit geschachtelten Ankern, zum Beispiel `Group`, haben auch geschachtelte Phasen. Wird beispielsweise in der Hauptphase eines Parsers ein `Group`-Anker ausgeführt, wird ein innerhalb der `Group`

geschachtelter **Marker**-Anker in einer geschachtelten Anfangsphase ausgeführt. Die geschachtelte Anfangsphase ist Bestandteil der Hauptphase des Parsers, liegt jedoch vor den übrigen Ankern der **Group**.

Ein weiteres Beispiel ist ein **RepeatingGroup**-Anker, der sowohl nach Trennzeichen als auch nach geschachtelten Ankern sucht. Damit er die geschachtelten Anker fehlerfrei finden kann, sucht er zunächst nach den Trennzeichen.

Suchbereich und Suchkriterien

Das obige Beispiel mit Suchphasen verdeutlicht das Suchbereich- und Suchkriterien-Konzept. Der Suchbereich ist der Teil eines Dokuments, in dem das Skript nach einem Anker sucht. Die Suchkriterien sind die Regeln, anhand derer das Skript den Anker im Suchbereich findet.

In der Anfangsphase beginnt das Skript mit der Suche nach dem **Marker**-Anker, der **First name:** am Beginn des Dokuments enthält. Der Suchbereich für diesen Anker ist das gesamte Dokument. Das Suchkriterium lautet, dass der Anker den Text **First name:** enthalten muss.

Der Suchbereich für den Anker **Last name:** beginnt am Ende von **First name:** und reicht bis zum Ende des Dokuments. Das Suchkriterium lautet, dass der Anker den Text **Last name:** enthalten muss.

In der Hauptphase ermittelt der Parser die **Content**-Anker zwischen den **Marker**-Ankern. Der Suchbereich für den Anker **Ron** reicht vom Ende des Ankers **First name:** bis zum Anfang des Ankers **Last name:**. Wenn dieser Anker ein leerzeichengetrenntes Format verwendet, lauten die Suchkriterien, dass der gesamte Text im Suchbereich abgerufen werden muss – nach dem vorangestellten und vor dem zweiten Leerzeichen.

Der Suchbereich für den Anker **Lehrer** reicht vom Ende von **Last Name:** bis zum Dokumentende. Die Suchkriterien gleichen denen des Ankers **Ron**.

Wir können nun diese Analyse in die oben aufgeführte Ankertabelle einfügen. Damit beschreibt die Tabelle nun die ganze Methode, mit der der Parser die Anker sucht.

Anker	Text im Quelldokument	Phase	Suchbereich	Suchkriterien
Marker	First name:	Initial	Gesamtes Dokument	Text = First name:
Inhalt	Ron	Main	Ende von First name: bis Anfang von Nachname:	Nach dem vorangestellten Leerzeichen Vor dem nächsten Leerzeichen
Marker	Last name:	Initial	Ende von First name: bis Dokumentende	Text = Last name:
Inhalt	Lehrer	Main	Ende von Last name: bis Dokumentende	Nach dem vorangestellten Leerzeichen Vor dem nächsten Leerzeichen

Anpassen der Suchphase

Durch Zuweisen der Eigenschaft **Phase** des Ankers können Sie die Phase ändern, in der das Skript nach einem Anker sucht.

Ein Quelldokument enthält die folgende Zeile:

```
CONTENT<10 characters>MARKER
```

In diesem Beispiel befindet sich der **Marker**-Anker zehn Zeichen hinter dem **Content**-Anker.

Standardmäßig sucht das Skript in der Anfangsphase nach dem **Marker**-Anker und in der Hauptphase nach dem **Content**-Anker. In diesem Fall funktioniert dieses Verfahren nicht, da das Skript den **Marker** erst finden kann, nachdem es den **Content** gefunden hat!

Die Lösung besteht darin, die Eigenschaft **phase** eines der Anker zu ändern. Sie können entweder den **Content** auf die Anfangsphase oder den **Marker** auf die Hauptphase umstellen. In beiden Fällen findet das Skript die Anker.

Anpassen des Suchbereichs

Es gibt zwei Möglichkeiten, den Suchbereich eines Ankers anzupassen.

- Durch Konfigurieren der Eigenschaft `phase` des Ankers oder der umgebenden Anker.
- Durch Konfigurieren der Eigenschaft `marking` der umgebenden Anker.

Eigenschaft „phase“

Wenn ein `Content`-Anker zwischen zwei `Marker`-Ankern liegt, ist der Suchbereich für den `Content`-Anker standardmäßig das Segment zwischen den `Marker`-Ankern.

Wenn Sie alle Anker auf dieselbe Phase einstellen, wird der Suchbereich für den `Content` nicht mehr durch den zweiten `Marker` begrenzt. Er reicht dann vom ersten `Marker` bis zum Ende des Dokuments.

Das folgende Quelldokument veranschaulicht das:

```
Tree Fig      Date<tab>October 27, 2003 (pruned)
Tree Date Palm Date April 27, 2003<tab>(planted)
```

Dieses Beispiel geht davon aus, dass das Quelldokument eine lockere Struktur aufweist, in der verschieden viele Leerzeichen, Tabulatoren und andere Zeichen in den Text eingestreut sind. Die Leerzeichen und Tabulatoren eignen sich daher nicht ohne weiteres als Delimiter. Dies ist häufig beim Parsen von Textverarbeitungsdokumenten der Fall.

Dieses Dokument lässt sich mit einem `RepeatingGroup`-Anker parsen, in den `Marker`- und `Content`-Anker geschachtelt sind. Die `Marker`-Anker sind die Strings `Baum` und `Datum`. Die `Content`-Anker umfassen den gesamten Text zwischen den `Marker`-Ankern, einschließlich der Leerzeichen und Tabulatoren.

Das Problem beim Parsen dieses Dokuments entsteht bei der zweiten Iteration der `RepeatingGroup`, welche die zweite Zeile parst. Wenn wir die `Marker`-Anker in der Anfangsphase belassen, hält das Skript die erste Instanz des Worts `Datum` irrtümlich für einen `Marker`. In der Hauptphase wird `Date Palm` nicht gefunden, da der Suchbereich zwischen den beiden `Marker`-Ankern liegt und sich dazwischen kein Text befindet.

Eine mögliche Lösung für dieses Problem besteht darin, den `Marker` für `Date` in die Hauptphase zu verschieben und den `Content`-Anker `Date Palm` unter Verwendung eines Ausdrucks zu definieren, der nach einem aus einem oder zwei Wörtern bestehenden Baumnamen sucht. In der Anfangsphase der `RepeatingGroup` findet das Skript den `Marker` für `Baum`. In der Hauptphase findet es `Date Palm`, gefolgt vom `Marker` für `Date`.

Durch die Änderung der Phaseneinstellungen haben wir den Suchbereich für den Baumnamen geändert. Er reicht nun von `Baum` bis zum Ende der Iteration, und `Date Palm` wird vom Skript ordnungsgemäß gefunden.

Eigenschaft „marking“

Ein Quelldokument weist die folgende Struktur auf:

```
MARKER
%%CONTENT A
^^^CONTENT B
```

Angenommen, Content A und Content B treten in den Quelldokumenten in unterschiedlicher Reihenfolge auf. In einigen Dokumenten steht Content B vor Content A.

In diesem Fall lauten die Suchkriterien:

- Content A und Content B folgen beide nach dem Marker-Anker.
- Content A beginnt mit %%, und Content B beginnt mit ^^.

Standardmäßig reicht der Suchbereich für Content A vom Ende des Marker-Ankers bis zum Ende des Dokuments. Der Suchbereich für Content B reicht vom Ende von Content A bis zum Ende des Dokuments. Dieses Verfahren funktioniert jedoch nicht in allen Dokumenten, da Content A und Content B auch in umgekehrter Reihenfolge auftreten können.

Die Lösung besteht darin, den Suchbereich für Content B zu ändern. Zu diesem Zweck konfigurieren Sie die Eigenschaft `marking` von Content A. Die Eigenschaft `marking` legt fest, wo das Skript die Referenzpunkte einfügen soll, die den Anfang und das Ende des Suchbereichs bestimmen.

Die Standardeinstellung ist `marking = full`. Dies bedeutet, dass das Skript Referenzpunkte vor und hinter jeden Anker setzt. Der Suchbereich für Content B beginnt beim letzten Referenzpunkt, also bei dem hinter Content A. Wie wir bereits gesehen haben, führt dies beim Parsen zu Fehlern.

Um zu verhindern, dass das Skript Referenzpunkte um Content A herum platziert, setzen Sie die Eigenschaft `marking` von Content A auf `none`. Dadurch beginnt der Suchbereich für Content B am Ende des Marker-Ankers. Damit kann das Skript nun Content B auch dann finden, wenn er vor Content A steht.

In der folgenden Tabelle werden alle vier möglichen Werte der Eigenschaft `marking` beschrieben. Bei der Spalte „Ergebnis“ wurde unterstellt, dass im obigen Beispiel der Wert `marking` dem Content A zugewiesen wurde.

Eigenschaft „marking“	Erläuterung	Ergebnis
full	Das Skript setzt am Anfang und am Ende des aktuellen Ankers jeweils einen Referenzpunkt. Dies ist das Standardverhalten.	Das Skript sucht den nächsten Anker nach dem Ende des aktuellen Ankers. Content B folgt nach Content A.
Anfangsposition	Das Skript setzt nur am Anfang des aktuellen Ankers einen Referenzpunkt.	Das Skript sucht den nächsten Anker nach dem Beginn des aktuellen Ankers. Content B überlappt oder folgt nach Content A.
Endposition	Das Skript setzt nur am Ende des aktuellen Ankers einen Referenzpunkt.	Das Skript sucht den nächsten Anker nach dem Ende des aktuellen Ankers. Content B folgt nach Content A.
none	Das Skript setzt beim aktuellen Anker keine Referenzpunkte.	Das Skript sucht den nächsten Anker nach dem Ende des vorherigen Ankers. Content B folgt nach Marker, unabhängig von der Position von Content A.

Hinweis: In einigen Fällen müssen Sie einen Anker verwenden, der einen Referenzpunkt markiert. Dies ist beispielsweise beim Trennzeichen einer `RepeatingGroup` der Fall. Falls das Trennzeichen keinen Punkt markieren kann, tut es nichts. Es erscheint ein Warnhinweis, wenn Sie an einer Stelle, an der eine Markierung erforderlich ist, einen nicht-markierenden Anker verwenden.

Online-Beispiele

Ein Online-Beispiel zur Eigenschaft „marking“ enthält das Projekt `samples\Projects\Marking_Mode\Marking_Mode.cmw`. In diesem Beispiel wird durch Konfigurieren dieser Eigenschaft der Suchbereich für einen Content-Anker verändert.

Ein weiteres Beispiel finden Sie unter `samples\Projects\NonMarker\NonMarker.cmw`. Dieses Beispiel zeigt, wie sich mit Hilfe der Option `marking = none` zwei Content-Anker überschneiden können. Außerdem verdeutlicht das Beispiel, wie mit `direction = backward` die Suche vom Ende des Suchbereichs aus gestartet werden kann.

Anpassen der Suchkriterien

Das Skript kann anhand zahlreicher Suchkriterien nach Ankern suchen. Hier nur einige Beispiele:

- Nach der Position der Delimiter, die das Skript der Beispielquelle entnimmt
- Nach dem positionalen Abstand, d. h. der Anzahl der Zeichen zu einem Referenzpunkt
- Durch Suchen nach einem bestimmten Text
- Durch Suchen nach einem Muster oder einem regulären Ausdruck
- Durch Suchen nach einem angegebenen Datentyp
- Durch Suchen nach einem Attributwert

Diese Suchkriterien können Sie fast uneingeschränkt miteinander kombinieren. So können Sie zum Beispiel angeben, dass ein Content-Anker zwei Tabulatoren hinter einem Marker-Anker beginnt und zehn Zeichen lang ist. In diesem Fall definieren Sie den Anfang des Content-Ankers mit einem Delimiter-Kriterium und das Ende mit einem Abstandskriterium.

Diejenigen Komponenten, die diese Suche ausführen, werden als Suchkomponenten bezeichnet. Weitere Informationen hierzu finden Sie unter [“Suchkomponenten: Referenz” auf Seite 248](#).

Eingrenzen der Suchkriterien mit Datentypen

Standardmäßig sucht das Skript neben anderen Suchkriterien nach einem Content-Anker anhand des Datentyps seines Datenbehälters.

Im folgenden Beispiel entspricht der Suchbereich eines Content-Ankers dem folgenden String:

```
The students' grades were 81, 56, and 95, respectively.
```

Für den Anker sind keine weiteren Suchkriterien definiert. Wenn Sie den Anker nun einem Datenbehälter des Typs `xs:string` zuordnen, ruft der Anker den gesamten String ab.

Weist der Datenbehälter den Typ `xs:integer` auf, dann sucht das Skript nach dem ersten Teilstring, der diesem Datentyp entspricht. Wenn Sie den Anker mit `direction = forward` konfigurieren, ruft er die Zahl 81 ab. Bei `direction = backward` ruft der Anker die Zahl 95 ab.

Nehmen wir nun an, dass der Datenbehälter den Typ `xs:integer` hat und durch Vorgabe seitens des Schemas auf Werte unter 60 beschränkt ist. In diesem Fall sucht das Skript nach einer Ganzzahl, die diese Einschränkung erfüllt, und gibt daher 56 zurück.

Datentypen in Kombination mit anderen Suchkriterien

Sie können die Suche anhand eines Datentyps mit anderen Suchkriterien kombinieren. Angenommen, Sie konfigurieren den Content-Anker im obigen Beispiel so, dass er den folgenden regulären Ausdruck sucht:

```
[", .*, "]
```

Dieser reguläre Ausdruck sucht nach zwei Kommata, die durch beliebige Zeichen außer einem Zeilenvorschub voneinander getrennt sind. Die Suche findet den folgenden Teilstring:

```
, 56,
```

Wenn der Typ des Datenbehälters `xs:integer` ist, ruft der Anker den Wert 56 ab.

Listen-Datentypen

Ein Datenbehälter kann eine durch Leerstellen getrennte Liste sein. Das Skript filtert den vom `Content`-Anker abgerufenen Text gemäß den XSD-Typen der Listenelemente.

Angenommen, im Schema ist ein Attribut `grades` definiert, das eine Liste von Elementen des Typs `xs:integer` darstellt. Wenn Sie dem oben beschriebenen `Content`-Anker das Attribut `grades` zuordnen, gibt der Anker eine Liste der im String enthaltenen ganzen Zahlen zurück:

```
81 56 95
```

Falls das Attribut `grades` zu einem Element namens `Students` gehört, sieht die XML-Ausgabe folgendermaßen aus:

```
<Students grades="81 56 95" />
```

Wenn Sie den `Content`-Anker mit `direction = backward` definieren, wird die Liste umgekehrt:

```
<Students grades="95 56 81" />
```

Typ des Dezimaltrennzeichens

Bei Datenbehältern des Typs `xs:decimal` geht das Skript davon aus, dass das Dezimaltrennzeichen ein Punkt ist. Wenn in Ihren landesspezifischen Einstellungen ein Komma als Dezimaltrennzeichen vorgesehen ist, schlägt die Suche mit `xs:decimal` eventuell fehl.

Typsuche mit `closing_marker`

Wenn ein `Content`-Anker die Eigenschaft `closing_marker`, jedoch nicht die Eigenschaft `opening_marker` aufweist, gibt das Skript den Teilstring zurück, der `closing_marker` am nächsten steht und mit dem Typ des Datenbehälters übereinstimmt.

Wenn Sie im obigen Beispiel das Wort `respectively` als `closing_marker` definieren und der Datenbehälter den Typ `xs:integer` aufweist, ruft der Anker den Wert 95 ab.

Online-Beispiel

Ein Online-Beispiel zur Suche nach einem Datentyp enthält das Projekt `samples\Projects\Pattern\Pattern.cmw`. Das Beispiel bezieht sich auf einen Parser mit einem einzelnen `Content`-Anker, der einem XML-Element zugeordnet ist. Das Schema beschränkt das Element anhand des Typs `xs:pattern` auf bestimmte akzeptable Zeichenfolgen. Der Anker gibt den Teil des Quelldokumentes aus, der diesem Muster entspricht.

Deaktivieren der Datentypsuche

Sie können die Suche nach Datentyp deaktivieren, indem Sie die Eigenschaft **`disable_XSD_type_search`** des **`Content`**-Ankers auswählen. Der Anker sucht dann nur anhand der anderen Kriterien, ohne Berücksichtigung des Typs des Datenbehälters.

Falls das Ergebnis nicht den richtigen Datentyp hat, kann es nicht im Datenbehälter gespeichert werden, und der Anker schlägt fehl. Durch Anwendung von Transformern können Sie das Ergebnis in den richtigen Typ

konvertieren und so ein Fehlschlagen verhindern. Weitere Informationen hierzu finden Sie unter ["Übersicht über Transformer" auf Seite 261](#).

Angenommen, das Quelldokument enthält ein Datum im Format `dd-mm-yyyy`, und Sie möchten das Datum in einem Datenbehälter des Typs `xs:date` speichern. Diese Anforderung können Sie auf folgende Weise realisieren:

1. Definieren Sie einen **Content**-Anker, der das Datum `dd-mm-yyyy` abrufen, wobei Sie die Nichtübereinstimmung mit dem Datentyp `xs:date` zunächst ignorieren.
2. Konfigurieren Sie den Anker mit einem **DateFormatICU**-Transformer, der das Ergebnis in den Typ `xs:date` konvertiert.

Anker mit geschachtelten Ankern

Eine interessante Frage ist, wie ein Parser nach einem Anker sucht, der geschachtelte Anker enthält, zum Beispiel nach einem `Group`-Anker.

Das Skript sucht nicht zuerst nach `Group` und dann die geschachtelten Anker. Es sucht vielmehr direkt nach den geschachtelten Ankern. Der Geltungsbereich der `Group` wird durch die geschachtelten Anker definiert, die das Skript findet.

Angenommen, ein Parser enthält beispielsweise die folgende Ankerfolge. Dabei wird unterstellt, dass die Anker die Standardeigenschaften `phase`, `marking` und `optional` aufweisen.

```
Marker A
Group
  Marker B
  Content C
  Marker D
Marker E
```

Das Skript sucht zunächst nach `Marker A` und `Marker E`. Der Suchbereich der `Group` ist der Bereich zwischen `Marker A` und `Marker E`.

Innerhalb des Suchbereichs der `Group` sucht das Skript dann nach `Marker B` und `Marker D`. Der Bereich zwischen diesen `Marker`-Ankern ist der Suchbereich für `Content C`.

Im zuletzt genannten Suchbereich sucht das Skript nach `Content C`.

Ankerkomponenten: Referenz

Die Komponenten **Anker** zeigen die Stellen an, an denen sich in den Quelldokumenten Daten befinden.

Alternatives

Der **Alternatives**-Anker definiert eine Gruppe alternativer, geschachtelter Anker. Sie können ein Kriterium für die Auswahl einer Alternative über die Befehlszeile definieren.

Die nachstehende Tabelle beschreibt die Eigenschaften des **Alternatives**-Ankers:

Eigenschaft	Beschreibung
disabled	<p>Legt fest, ob die Komponente und alle zugehörigen untergeordneten Komponenten vom Skript ignoriert werden. Verwenden Sie diese Eigenschaft zum Testen, Debuggen und Ändern eines Skripts. Sie können eine der folgenden Optionen auswählen:</p> <ul style="list-style-type: none"> - Ausgewählt. Das Skript ignoriert die Komponente. - Gelöscht. Das Skript wendet die Komponente an. <p>Standardmäßig ist die Eigenschaft deaktiviert.</p>
marking	<p>Legt fest, ob ein Anker als Anfang des Suchbereichs für den danach folgenden Anker verwendet wird. Sie können eine der folgenden Optionen auswählen:</p> <ul style="list-style-type: none"> - Anfangsposition. Setzt einen Referenzpunkt vor dem aktuellen Anker. - Endposition. Setzt einen Referenzpunkt hinter dem aktuellen Anker. - Vollständig. Setzt einen Referenzpunkt vor und hinter dem aktuellen Anker. - Keine. Es wird kein Referenzpunkt erstellt. <p>Weitere Informationen hierzu finden Sie unter "So sucht der Parser Anker" auf Seite 211.</p>
name	<p>Eine beschreibende Kennzeichnung für die Komponente. Diese Kennzeichnung wird in der Protokolldatei sowie der Ereignisansicht angezeigt. Ermitteln Sie anhand der Eigenschaft name, welche Komponente das Ereignis verursacht hat.</p>
on_fail	<p>Die Aktion, die beim Ausfall der Komponente durchgeführt wird. Sie können eine der folgenden Optionen auswählen:</p> <ul style="list-style-type: none"> - Gelöscht. Es wird keine Aktion ausgeführt. - CustomLog. Es wird in das Benutzerprotokoll geschrieben. - LogError. Eine Fehlermeldung wird in das Engine-Protokoll geschrieben. - LogInfo. Eine Informationsmeldung wird in das Engine-Protokoll geschrieben. - LogWarning. Eine Warnmeldung wird in das Engine-Protokoll geschrieben. - NotifyFailure. Eine Mitteilung wird gesendet. <p>Die Standardoption lautet „Gelöscht“. Weitere Informationen über die Behandlung von Komponentengefehlern finden Sie in "Fehlerbehandlung" auf Seite 410.</p>
optional	<p>Legt fest, ob ein Komponentengefehle den Ausfall der übergeordneten Komponente verursacht. Sie können eine der folgenden Optionen auswählen:</p> <ul style="list-style-type: none"> - Ausgewählt. Ein Komponentengefehle führt nicht zum Ausfall der übergeordneten Komponente. - Gelöscht. Ein Komponentengefehle verursacht den Ausfall der übergeordneten Komponente. <p>Standardwert ist "Gelöscht". Weitere Informationen über Komponentengefehle finden Sie in "Fehlerbehandlung" auf Seite 410.</p>
phase	<p>Legt den Zeitpunkt fest, an dem das Skript die Komponente verarbeitet. Sie können eine der folgenden Optionen auswählen:</p> <ul style="list-style-type: none"> - Anfangsphase. Das Skript verarbeitet die Komponente während der Anfangsphase. - Hauptphase. Das Skript verarbeitet die Komponente während der Hauptphase. - Endphase. Das Skript verarbeitet die Komponente während der Endphase. <p>Weitere Informationen hierzu finden Sie unter "So sucht der Parser Anker" auf Seite 211. Standardwert ist "Hauptphase".</p>

Eigenschaft	Beschreibung
remark	Ein benutzerdefinierter Kommentar, der den Zweck oder die Aktion der Komponente beschreibt.
selector	<p>Legt das Kriterium für die Auswahl eines Ankers der unter dem Alternatives-Anker geschachtelten Anker fest. Sie können eine der folgenden Optionen auswählen:</p> <ul style="list-style-type: none"> - ScriptOrder. Der Parser testet die geschachtelten Anker in der im Skript definierten Reihenfolge. Er akzeptiert den ersten erfolgreich getesteten Anker. Wenn alle Anker fehlschlagen, schlägt auch der Alternatives-Anker fehl. - DocumentOrder. Der Parser testet alle geschachtelten Anker. Er akzeptiert entweder den ersten oder den letzten Anker, je nach den Positionen der Anker im Quelldokument, wie in der Eigenschaft select festgelegt. Wenn alle Anker fehlschlagen, schlägt auch der Alternatives-Anker fehl. - NameSwitch. Der Parser sucht den Anker, dessen name-Eigenschaft in dem in option_name definierten Datenbehälter angegeben ist. Die übrigen Anker werden ignoriert. Wenn der benannte geschachtelte Anker fehlschlägt, schlägt auch der Alternatives-Anker fehl.

Beispiel

Sie parsen ein Dokument, in dem eine Datumsangabe in einem der folgenden Formate vorkommen kann:

```
21/10/03
October 21, 2003
```

Um diesen Inhalt zu verarbeiten, können Sie einen `Alternatives`-Anker mit zwei enthaltenen `Content`-Ankern definieren, die ihre Ausgabe in verschiedenen XML-Elementen speichern. Jedes XML-Element kann nur eines dieser Datumsmuster annehmen. Der `Alternatives`-Anker wird mit der Eigenschaft `selector = ScriptOrder` konfiguriert.

Beim Ausführen des `Alternatives`-Ankers prüft der Parser den ersten `Content`-Anker. Falls das Datum mit dem Muster des ersten Ankers übereinstimmt, wird der erste `Content`-Anker abgerufen. Stimmt das Datum nicht mit dem Muster überein, schlägt der erste `Content`-Anker fehl, und der `Alternatives`-Anker prüft den zweiten `Content`-Anker. Auf diese Weise kann der Parser beide Datumsmuster verarbeiten.

So definieren Sie einen Alternatives-Anker

Fügen Sie einen **Alternatives**-Anker hinzu, indem Sie das Skript bearbeiten. In den **Alternatives**-Anker schachteln Sie die alternativen Anker.

Auswählen eines sekundären Parsers mit Alternatives-Anker

Mit dem Anker `Alternatives` können Sie steuern, welcher der verfügbaren sekundären Parser ein Dokument verarbeitet. Auf diese Weise kann der Hauptparser Quelldokumente verschiedenen Typs verarbeiten.

Angenommen, die Startseite einer Zeitungs-Website enthält Links zu Artikeln. Die einzelnen Artikel sind den Kategorien `News`, `Business` oder `Sports` zugeordnet. Sie möchten die Artikel parsen und dabei für jeden Typ einen anderen Parser verwenden:

```
<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
```

Diese Anforderung können Sie auf folgende Weise realisieren:

1. Der Hauptparser ruft den Dateinamen eines Artikels ab und speichert ihn in einer Variablen.
2. Der Hauptparser enthält einen `Alternatives`-Anker, der mit der Option `DocumentOrder` konfiguriert ist.
3. Der `Alternatives`-Anker enthält geschachtelte `Group`-Anker.

4. Jeder **Group-Anker** ist mit einem **Marker-Anker** und einer **RunParser-Aktion** konfiguriert:
 - Der erste **Group-Anker** enthält einen **Marker-Anker**, der den String **News** sucht. Der **Group-Anker** ist mit einer **RunParser-Aktion** konfiguriert, die einen sekundären Parser namens **NewsParser** ausführt.
 - Der zweite **Group-Anker** enthält einen **Marker-Anker**, der nach **Business** sucht und **BusinessParser** ausführt.
 - Der dritte **Group-Anker** enthält einen **Marker-Anker**, der nach **Sports** sucht und **SportsParser** ausführt.

Vom **Alternatives-Anker** werden alle drei **Group-Anker** geprüft. Der **Group-Anker** mit dem ersten **Marker-Anker**, der nach dem Dateinamen vorkommt, wird akzeptiert. Vom **Group-Anker** wird der entsprechende Parser an der Datei ausgeführt.

Online-Beispiel

Ein Online-Beispiel zu diesem Anker befindet sich im Projekt `samples\Projects\Alternatives\Alternatives.cmw`. In diesem Beispiel werden verschiedene Namens- und Datumsformate, die im Quelldokument vorkommen können, mit Hilfe eines **Alternatives-Ankers** geparkt.

Content

Ein **Content-Anker** ruft Text aus dem Quelldokument ab. Der Parser sucht in einem definierten Bereich gemäß den festgelegten Suchkriterien und speichert den abgerufenen Text in einem Datenbehälter.

Die nachstehende Tabelle beschreibt die Eigenschaften des Ankers **Content**:

Eigenschaft	Beschreibung
<code>allow_empty_values</code>	<p>Legt fest, ob der Content-Anker leer sein kann. Sie können eine der folgenden Optionen auswählen:</p> <ul style="list-style-type: none"> - Ausgewählt. Der Eigenschaft data_holder wird ein leerer Wert zugewiesen. - Gelöscht. Leere Werte sind nicht zulässig. <p>allow_empty_values muss unter folgenden Umständen ausgewählt werden:</p> <ul style="list-style-type: none"> - Wenn der Anker mit value = LearnByExample konfiguriert ist und zwischen den Delimitern nichts ist - Wenn zwischen opening_marker und closing_marker nichts ist.
<code>closing_marker</code>	<p>Definiert das Ende eines Bereichs, in dem der Parser nach dem Content-Anker sucht. Sie können eine der folgenden Optionen auswählen:</p> <ul style="list-style-type: none"> - NewlineSearch. Das Ende des Content-Ankers ist das nächste Newline-Zeichen. - OffsetSearch. Das Ende des Content-Ankers ist die Anzahl der Zeichen, die in offset angegeben sind. - PatternSearch. Das Ende des Content-Ankers ruft den ersten Text ab, der mit einem angegebenen regulären Ausdruck übereinstimmt. - TextSearch. Das Ende des Content-Ankers ist ein festgelegter Textstring.
<code>data_holder</code>	<p>Definiert einen Datenbehälter, in dem der Content-Anker den abgerufenen Text speichert.</p>

Eigenschaft	Beschreibung
direction	<p>Eine Suchrichtung für den Anker innerhalb des Suchbereichs. Sie können eine der folgenden Optionen auswählen:</p> <ul style="list-style-type: none"> - Zurück. Die Suche beginnt am Ende des Suchbereichs und findet die letzte Instanz des Ankers. - Vorwärtssuche. Die Suche beginnt am Anfang des Suchbereichs und findet die erste Instanz des Ankers. <p>Bei einem Marker-Anker können Sie dieses Verhalten mithilfe der Eigenschaft count ändern. Wenn beispielsweise direction = backward und count = 2 ist, findet das Skript die vorletzte Instanz.</p> <p>Standardwert ist "Vorwärtssuche". Weitere Informationen hierzu finden Sie unter "So sucht der Parser Anker" auf Seite 211.</p>
disable_XSD_type_search	<p>Legt fest, ob der Parser nach Inhalten sucht, die mit dem Datentyp des Datenbehälters übereinstimmen. Sie können eine der folgenden Optionen auswählen:</p> <ul style="list-style-type: none"> - Ausgewählt. Der Parser sucht ohne Berücksichtigung des Datentyps. Nachdem Transformer auf den Inhalt angewendet wurden und das Ergebnis nicht zum Datentyp passt, schlägt der Anker fehl. - Gelöscht. Der Parser sucht nach Inhalt, der mit dem Datentyp übereinstimmt. <p>Standardwert ist "Gelöscht". Weitere Informationen hierzu finden Sie unter "Eingrenzen der Suchkriterien mit Datentypen" auf Seite 215.</p>
disabled	<p>Legt fest, ob die Komponente und alle zugehörigen untergeordneten Komponenten vom Skript ignoriert werden. Verwenden Sie diese Eigenschaft zum Testen, Debuggen und Ändern eines Skripts. Sie können eine der folgenden Optionen auswählen:</p> <ul style="list-style-type: none"> - Ausgewählt. Das Skript ignoriert die Komponente. - Gelöscht. Das Skript wendet die Komponente an. <p>Standardmäßig ist die Eigenschaft deaktiviert.</p>
ignore_default_transformers	<p>Legt fest, ob der Parser die Standard-Transformer auf den Inhalt anwendet.</p> <p>Standardwert ist "Gelöscht".</p> <p>Weitere Informationen hierzu finden Sie unter "Übersicht über Transformer" auf Seite 261.</p>
marking	<p>Legt fest, ob ein Anker als Anfang des Suchbereichs für den danach folgenden Anker verwendet wird. Sie können eine der folgenden Optionen auswählen:</p> <ul style="list-style-type: none"> - Anfangsposition. Setzt einen Referenzpunkt vor dem aktuellen Anker. - Endposition. Setzt einen Referenzpunkt hinter dem aktuellen Anker. - Vollständig. Setzt einen Referenzpunkt vor und hinter dem aktuellen Anker. - Keine. Es wird kein Referenzpunkt erstellt. <p>Weitere Informationen hierzu finden Sie unter "So sucht der Parser Anker" auf Seite 211.</p>
name	<p>Eine beschreibende Kennzeichnung für die Komponente. Diese Kennzeichnung wird in der Protokolldatei sowie der Ereignisansicht angezeigt. Ermitteln Sie anhand der Eigenschaft name, welche Komponente das Ereignis verursacht hat.</p>
on_fail	<p>Die Aktion, die beim Ausfall der Komponente durchgeführt wird. Sie können eine der folgenden Optionen auswählen:</p> <ul style="list-style-type: none"> - Gelöscht. Es wird keine Aktion ausgeführt. - CustomLog. Es wird in das Benutzerprotokoll geschrieben. - LogError. Eine Fehlermeldung wird in das Engine-Protokoll geschrieben. - LogInfo. Eine Informationsmeldung wird in das Engine-Protokoll geschrieben. - LogWarning. Eine Warnmeldung wird in das Engine-Protokoll geschrieben. - NotifyFailure. Eine Mitteilung wird gesendet. <p>Die Standardoption lautet „Gelöscht“. Weitere Informationen über die Behandlung von Komponentenfehlern finden Sie in "Fehlerbehandlung" auf Seite 410.</p>

Eigenschaft	Beschreibung
opening_marker	<p>Definiert den Beginn eines Bereichs, in dem der Parser nach dem Content-Anker sucht. Die möglichen Werte sind folgende Komponenten:</p> <ul style="list-style-type: none"> - NewlineSearch. Der Beginn des Content-Ankers ist das nächste Newline-Zeichen. - OffsetSearch. Der Beginn des Content-Ankers ist die Anzahl der Zeichen, die in offset angegeben sind. - PatternSearch. Der Beginn des Content-Ankers ruft den ersten Text ab, der mit einem angegebenen regulären Ausdruck übereinstimmt. - TextSearch. Der Beginn des Content-Ankers ist ein festgelegter Textstring.
optional	<p>Legt fest, ob ein Komponentenfehler den Ausfall der übergeordneten Komponente verursacht. Sie können eine der folgenden Optionen auswählen:</p> <ul style="list-style-type: none"> - Ausgewählt. Ein Komponentenfehler führt nicht zum Ausfall der übergeordneten Komponente. - Gelöscht. Ein Komponentenfehler verursacht den Ausfall der übergeordneten Komponente. <p>Standardwert ist "Gelöscht". Weitere Informationen über Komponentenfehler finden Sie in "Fehlerbehandlung" auf Seite 410.</p>
phase	<p>Legt den Zeitpunkt fest, an dem das Skript die Komponente verarbeitet. Sie können eine der folgenden Optionen auswählen:</p> <ul style="list-style-type: none"> - Anfangsphase. Das Skript verarbeitet die Komponente während der Anfangsphase. - Hauptphase. Das Skript verarbeitet die Komponente während der Hauptphase. - Endphase. Das Skript verarbeitet die Komponente während der Endphase. <p>Weitere Informationen hierzu finden Sie unter "So sucht der Parser Anker" auf Seite 211. Standardwert ist "Hauptphase".</p>
remark	Ein benutzerdefinierter Kommentar, der den Zweck oder die Aktion der Komponente beschreibt.
Transformer	Definiert eine Sequenz von Transformern, die der Parser auf den abgerufenen Text anwendet. Weitere Informationen unter Kapitel 17, "Transformer" auf Seite 261 .

Eigenschaft	Beschreibung
validators	Definiert eine Liste von Validatoren, die auf die Daten angewendet werden. Weitere Informationen hierzu finden Sie unter "Validatoren" auf Seite 413 .
value	<p>Definiert Kriterien für eine Suche in dem Bereich, die von den Attributen opening_marker und closing_marker definiert werden. Wenn kein opening_marker definiert ist, wird zwischen den umgebenden Referenzpunkten gesucht. Weitere Informationen hierzu finden Sie unter "So sucht der Parser Anker" auf Seite 211. Sie können eine der folgenden Optionen auswählen:</p> <ul style="list-style-type: none"> - Leer. Der Content-Anker ruft den gesamten Suchbereich ab. - AttributeSearch. Der Content-Anker ruft den Wert aus einem Ausdruck des Typs AttributeName = ... ab. Verwenden Sie diese Option, um Attributwerte aus einem XML- oder HTML-Quelldokument abzurufen. - LearnByExample. Der Parser lernt anhand des Parserformats und der Beispielquelle, welchen Text er abrufen soll. Wenn der Parser zum Beispiel ein tabulatorbegrenztes Format hat, zählt er die Anzahl der Tabulatoren vom Anfang des Suchbereichs ab bis zum Beispieltext. Er ruft den Text zwischen den entsprechenden Tabulatoren im Quelldokument ab. - PatternSearch. Der Content-Anker ruft den ersten Text ab, der mit einem angegebenen regulären Ausdruck übereinstimmt. - TypeSearch. Der Content-Anker ruft den ersten Text ab, der mit einem angegebenen Datentyp übereinstimmt. <p>Standardwert ist "Leer". Weitere Informationen zu diesen Optionen finden Sie im Abschnitt "Suchkomponenten: Referenz" auf Seite 248. Neben den Suchkomponenten verwendet der Parser den Datentyp von data_holder als Suchkriterium. Weitere Informationen hierzu finden Sie unter "Eingrenzen der Suchkriterien mit Datentypen" auf Seite 215.</p>

Die Eigenschaften **opening_marker** und **closing_marker** sind den **Marker**-Ankern in einer **Group**-Komponente gleichgestellt.

- Ein **Content**-Anker mit eingestellttem **opening_marker** ist wie eine **Group**-Komponente mit der folgenden Ankersequenz:
 1. Marker
 2. Content
- Ein **Content**-Anker mit eingestellttem **closing_marker** ist wie eine **Group**-Komponente mit der folgenden Ankersequenz:
 1. Content
 2. Marker
- Ein **Content**-Anker mit eingestellttem **opening_marker** und **closing_marker** ist wie eine **Group**-Komponente mit der folgenden Ankersequenz:
 1. Marker
 2. Content
 3. Marker

Weitere Informationen hierzu finden Sie im Abschnitt ["Suchkomponenten: Referenz" auf Seite 248](#).

Suchrichtung

Die Eigenschaft `direction` kann verschiedene Auswirkungen auf den Content-Anker haben. Bei `direction = backward`:

- Das Skript sucht rückwärts vom Ende des Suchbereichs nach dem `opening_marker` und dem `closing_marker`. `opening_marker` kommt dennoch vor `closing_marker`.
- sucht die Suchkomponente rückwärts vom Ende des Suchbereichs an.
- Bei Verwendung der Suchkomponente `LearnByExample` werden die Delimiter rückwärts vom Ende des Suchbereichs an gezählt.

Online-Beispiel

Ein Online-Beispiel zu Content-Ankern befindet sich im Projekt `samples\Projects\Content\Content.cmw`. Das Beispiel zeigt verschiedene Anwendungsmöglichkeiten der Eigenschaften `opening_marker`, `closing_marker` und `value` zur Konfiguration von Content-Ankern.

DelimitedSections

Der **DelimitedSections**-Anker parst Daten, die durch Trennzeichen unterteilt sind. Er definiert eine Gruppe geschachtelter Anker. Jeder geschachtelte Anker parst nur einen Abschnitt.

Die nachstehende Tabelle beschreibt die Eigenschaften des Ankers **DelimitedSections**:

Eigenschaft	Beschreibung
<code>disabled</code>	Legt fest, ob die Komponente und alle zugehörigen untergeordneten Komponenten vom Skript ignoriert werden. Verwenden Sie diese Eigenschaft zum Testen, Debuggen und Ändern eines Skripts. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none">- Ausgewählt. Das Skript ignoriert die Komponente.- Gelöscht. Das Skript wendet die Komponente an. Standardmäßig ist die Eigenschaft deaktiviert.
<code>marking</code>	Legt fest, ob ein Anker als Anfang des Suchbereichs für den danach folgenden Anker verwendet wird. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none">- Anfangsposition. Setzt einen Referenzpunkt vor dem aktuellen Anker.- Endposition. Setzt einen Referenzpunkt hinter dem aktuellen Anker.- Vollständig. Setzt einen Referenzpunkt vor und hinter dem aktuellen Anker.- Keine. Es wird kein Referenzpunkt erstellt. Weitere Informationen hierzu finden Sie unter "So sucht der Parser Anker" auf Seite 211 .
<code>name</code>	Eine beschreibende Kennzeichnung für die Komponente. Diese Kennzeichnung wird in der Protokolldatei sowie der Ereignisansicht angezeigt. Ermitteln Sie anhand der Eigenschaft name , welche Komponente das Ereignis verursacht hat.
Benachrichtigungen	Eine Liste mit NotificationHandler -Komponenten, die Benachrichtigungen aus geschachtelten Komponenten verarbeiten. Weitere Informationen hierzu finden Sie unter "Benachrichtigungen" auf Seite 430 .

Eigenschaft	Beschreibung
on_fail	<p>Die Aktion, die beim Ausfall der Komponente durchgeführt wird. Sie können eine der folgenden Optionen auswählen:</p> <ul style="list-style-type: none"> - Gelöscht. Es wird keine Aktion ausgeführt. - CustomLog. Es wird in das Benutzerprotokoll geschrieben. - LogError. Eine Fehlermeldung wird in das Engine-Protokoll geschrieben. - LogInfo. Eine Informationsmeldung wird in das Engine-Protokoll geschrieben. - LogWarning. Eine Warnmeldung wird in das Engine-Protokoll geschrieben. - NotifyFailure. Eine Mitteilung wird gesendet. <p>Die Standardoption lautet „Gelöscht“. Weitere Informationen über die Behandlung von Komponentenfehlern finden Sie in "Fehlerbehandlung" auf Seite 410.</p>
optional	<p>Legt fest, ob ein Komponentenfehler den Ausfall der übergeordneten Komponente verursacht. Sie können eine der folgenden Optionen auswählen:</p> <ul style="list-style-type: none"> - Ausgewählt. Ein Komponentenfehler führt nicht zum Ausfall der übergeordneten Komponente. - Gelöscht. Ein Komponentenfehler verursacht den Ausfall der übergeordneten Komponente. <p>Standardwert ist "Gelöscht". Weitere Informationen über Komponentenfehler finden Sie in "Fehlerbehandlung" auf Seite 410.</p>
phase	<p>Legt den Zeitpunkt fest, an dem das Skript die Komponente verarbeitet. Sie können eine der folgenden Optionen auswählen:</p> <ul style="list-style-type: none"> - Anfangsphase. Das Skript verarbeitet die Komponente während der Anfangsphase. - Hauptphase. Das Skript verarbeitet die Komponente während der Hauptphase. - Endphase. Das Skript verarbeitet die Komponente während der Endphase. <p>Weitere Informationen hierzu finden Sie unter "So sucht der Parser Anker" auf Seite 211. Standardwert ist "Hauptphase".</p>
remark	Ein benutzerdefinierter Kommentar, der den Zweck oder die Aktion der Komponente beschreibt.
separator	Definiert einen Anker, der die Abschnitte voneinander trennt.
separator_position	<p>Definiert die Positionierung des Trennzeichens relativ zu den Abschnitten. Sie können eine der folgenden Optionen auswählen:</p> <ul style="list-style-type: none"> - Nachher. Ein Trennzeichen steht nach jedem Abschnitt, auch nach dem letzten. Beispiel: 1 2 3 4 - Um. Ein Trennzeichen steht vor und nach jedem Abschnitt, auch vor dem ersten und nach dem letzten Abschnitt. Beispiel: 1 2 3 4 - Vorher. Ein Trennzeichen steht vor jedem Abschnitt, auch vor dem ersten. Beispiel: 1 2 3 4 - Zwischen. Ein Trennzeichen steht jeweils zwischen aufeinander folgenden Abschnitten, jedoch nicht vor dem ersten und nach dem letzten Abschnitt. Beispiel: 1 2 3 4
using_placeholders	<p>Diese Eigenschaft gibt an, ob der Anker DelimitedSections nach dem Trennzeichen eines optionalen Abschnitts sucht, der im Quelldokument fehlt. Sie können eine der folgenden Optionen auswählen:</p> <ul style="list-style-type: none"> - Immer. Das Trennzeichen eines fehlenden Abschnittes ist immer vorhanden. Beispiel: 1 3 - Nie. Das Trennzeichen eines fehlenden Abschnittes ist nie vorhanden. Beispiel: 1 3 - Wenn nötig. Das Trennzeichen eines fehlenden internen Abschnittes ist immer vorhanden. Das Trennzeichen eines fehlenden Endabschnittes ist nie vorhanden. Beispiel: 1 3 <p>In diesen Beispielen ist separator_position auf before gesetzt und Abschnitte 2 und 4 fehlen.</p>

Beispiel

Ein Lebenslaufformular enthält mehrere Abschnitte, denen jeweils eine Zeile mit Bindestrichen vorangestellt ist:

```
-----
Jane Palmer
Employee ID 123456
-----
Professional Experience
...
-----
Education
...
```

Sie können den in Abschnitte unterteilten Bereich als einen `DelimitedSections`-Anker und die Gedankenstrichzeile als `separator` (Trennzeichen) definieren. Da die Zeile mit den Gedankenstrichen immer vor dem zugehörigen Abschnitt steht, definieren Sie `separator_position` als `before`.

In den `DelimitedSections`-Anker schachteln Sie drei `Group`-Anker. Der erste `Group`-Anker parst den Abschnitt `Jane Palmer`, der zweite `Group`-Anker den Abschnitt `Professional Experience` und so weiter.

Optionale Abschnitte

Nehmen wir nun an, dass im obigen Beispiel der zweite Abschnitt, `Professional Experience`, in einigen Quelldokumenten fehlt. Das zugehörige Trennzeichen, nämlich die Zeile mit Bindestrichen, ist jedoch immer vorhanden.

```
-----
Jane Palmer
Employee ID 123456
-----
Education
...
```

Um diese Situation zu bewältigen, konfigurieren Sie `DelimitedSections` folgendermaßen:

- Wählen Sie im zweiten `Group`-Anker die Eigenschaft `optional` aus. Dies bedeutet, dass ein Fehler des `Group`-Ankers nicht zu einem Fehler des `DelimitedSections`-Ankers führt.
- Setzen Sie im `DelimitedSections`-Anker den Wert `using_placeholders = always`. Dadurch sucht der Anker das Trennzeichen des optionalen Abschnitts auch dann, wenn dieser Abschnitt selbst fehlt.

Unterstellen wir nun, dass nicht nur der Abschnitt `Professional Experience` fehlt, sondern auch das zugehörige Trennzeichen.

```
-----
Jane Palmer
Employee ID 123456
-----
Education
...
```

In diesem Fall konfigurieren Sie `DelimitedSections` folgendermaßen:

- Wählen Sie im zweiten `Group`-Anker die Eigenschaft `optional` aus.
- Setzen Sie im `DelimitedSections`-Anker den Wert `using_placeholders = never`. Dadurch sucht der Anker nicht nach dem Trennzeichen des fehlenden Abschnitts.

So definieren Sie einen `DelimitedSections`-Anker

Fügen Sie einen **`DelimitedSections`**-Anker hinzu, indem Sie das Skript im IntelliScript-Editor bearbeiten. Im **`DelimitedSections`**-Anker verschachteln Sie eine Folge von Ankern, die die einzelnen Abschnitte parsen.

Online-Beispiel

Ein Online-Beispiel zu diesem Anker befindet sich im Projekt `samples\Projects\DelimitedSections\DelimitedSections.cmw`. Im Beispiel wird gezeigt, wie durch das Symbol `|` getrennte Abschnitte mit Hilfe eines `DelimitedSections`-Ankers geparkt werden. Jeder Abschnitt wird von einem einzelnen `Content`-Anker geparkt.

EmbeddedParser

Der Anker **EmbeddedParser** parst seinen Suchbereich mit einem zweiten Parser. Er kann sich selbst rekursiv aufrufen.

Die nachstehende Tabelle beschreibt die Eigenschaften des **EmbeddedParser**-Ankers:

Eigenschaft	Beschreibung
disabled	Legt fest, ob die Komponente und alle zugehörigen untergeordneten Komponenten vom Skript ignoriert werden. Verwenden Sie diese Eigenschaft zum Testen, Debuggen und Ändern eines Skripts. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none">- Ausgewählt. Das Skript ignoriert die Komponente.- Gelöscht. Das Skript wendet die Komponente an. Standardmäßig ist die Eigenschaft deaktiviert.
marking	Legt fest, ob ein Anker als Anfang des Suchbereichs für den danach folgenden Anker verwendet wird. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none">- Anfangsposition. Setzt einen Referenzpunkt vor dem aktuellen Anker.- Endposition. Setzt einen Referenzpunkt hinter dem aktuellen Anker.- Vollständig. Setzt einen Referenzpunkt vor und hinter dem aktuellen Anker.- Keine. Es wird kein Referenzpunkt erstellt. Weitere Informationen hierzu finden Sie unter "So sucht der Parser Anker" auf Seite 211 .
name	Eine beschreibende Kennzeichnung für die Komponente. Diese Kennzeichnung wird in der Protokolldatei sowie der Ereignisansicht angezeigt. Ermitteln Sie anhand der Eigenschaft name , welche Komponente das Ereignis verursacht hat.
on_fail	Die Aktion, die beim Ausfall der Komponente durchgeführt wird. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none">- Gelöscht. Es wird keine Aktion ausgeführt.- CustomLog. Es wird in das Benutzerprotokoll geschrieben.- LogError. Eine Fehlermeldung wird in das Engine-Protokoll geschrieben.- LogInfo. Eine Informationsmeldung wird in das Engine-Protokoll geschrieben.- LogWarning. Eine Warnmeldung wird in das Engine-Protokoll geschrieben.- NotifyFailure. Eine Mitteilung wird gesendet. Die Standardoption lautet „Gelöscht“. Weitere Informationen über die Behandlung von Komponentenfehlern finden Sie in "Fehlerbehandlung" auf Seite 410 .
optional	Legt fest, ob ein Komponentenfehler den Ausfall der übergeordneten Komponente verursacht. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none">- Ausgewählt. Ein Komponentenfehler führt nicht zum Ausfall der übergeordneten Komponente.- Gelöscht. Ein Komponentenfehler verursacht den Ausfall der übergeordneten Komponente. Standardwert ist "Gelöscht". Weitere Informationen über Komponentenfehler finden Sie in "Fehlerbehandlung" auf Seite 410 .
Parser	Legt den Namen für den zweiten Parser fest, der im selben Projekt definiert ist.

Eigenschaft	Beschreibung
Phase	Legt den Zeitpunkt fest, an dem das Skript die Komponente verarbeitet. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none"> - Anfangsphase. Das Skript verarbeitet die Komponente während der Anfangsphase. - Hauptphase. Das Skript verarbeitet die Komponente während der Hauptphase. - Endphase. Das Skript verarbeitet die Komponente während der Endphase. Weitere Informationen hierzu finden Sie unter "So sucht der Parser Anker" auf Seite 211 . Standardwert ist "Hauptphase".
remark	Ein benutzerdefinierter Kommentar, der den Zweck oder die Aktion der Komponente beschreibt.
schema_connections	Definiert eine Liste von Connect -Unterkomponenten, die die Relation zwischen Datenbehältern in der Ausgabe des Hauptparsers und des zweiten Parsers definieren. Weitere Informationen hierzu finden Sie unter "Connect" auf Seite 256 .
source_transformers	Definiert eine Folge von Transformatoren, die der Parser auf den Suchbereich anwendet, bevor dieser vom sekundären Parser verarbeitet wird.

Beispiel

Ein tabulatorbegrenztes Dokument enthält einen einzigen Abschnitt, der das Komma als Delimiter verwendet.

Um dieses Dokument zu parsen, definieren Sie einen Hauptparser mit dem Format `TabDelimited`. Definieren Sie einen weiteren Parser mit dem Format `CommaDelimited`. Führen Sie den zweiten Parser mit Hilfe eines `EmbeddedParser`-Ankers innerhalb der Ausführung des ersten Parsers aus.

Online-Beispiel

Ein Online-Beispiel zu diesem Anker befindet sich im Projekt `samples\Projects\EmbeddedParser\EmbeddedParser.cmw`. In diesem Beispiel wird mit einem Hauptparser die Position einer Adresse ermittelt. Anschließend wird ein `EmbeddedParser` ausgeführt, der die Adresse parst.

EnclosedGroup

Der Anker **EnclosedGroup** definiert einen eingegrenzten Bereich, der geschachtelte Anker enthält. Die Grenzen werden durch **opening**- und **closing**-Anker gekennzeichnet. Bei geschachtelten Begrenzungen, zum Beispiel Klammern oder HTML-Tags, findet **EnclosedGroup** die zueinander gehörenden Begrenzungen.

Die nachstehende Tabelle beschreibt die Eigenschaften des Ankers **EnclosedGroup**:

Eigenschaft	Beschreibung
closing	Definiert den schließenden Anker der EnclosedGroup .
disabled	Legt fest, ob die Komponente und alle zugehörigen untergeordneten Komponenten vom Skript ignoriert werden. Verwenden Sie diese Eigenschaft zum Testen, Debuggen und Ändern eines Skripts. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none"> - Ausgewählt. Das Skript ignoriert die Komponente. - Gelöscht. Das Skript wendet die Komponente an. Standardmäßig ist die Eigenschaft deaktiviert.

Eigenschaft	Beschreibung
marking	<p>Legt fest, ob ein Anker als Anfang des Suchbereichs für den danach folgenden Anker verwendet wird. Sie können eine der folgenden Optionen auswählen:</p> <ul style="list-style-type: none"> - Anfangsposition. Setzt einen Referenzpunkt vor dem aktuellen Anker. - Endposition. Setzt einen Referenzpunkt hinter dem aktuellen Anker. - Vollständig. Setzt einen Referenzpunkt vor und hinter dem aktuellen Anker. - Keine. Es wird kein Referenzpunkt erstellt. <p>Weitere Informationen hierzu finden Sie unter "So sucht der Parser Anker" auf Seite 211.</p>
name	<p>Eine beschreibende Kennzeichnung für die Komponente. Diese Kennzeichnung wird in der Protokolldatei sowie der Ereignisansicht angezeigt. Ermitteln Sie anhand der Eigenschaft name, welche Komponente das Ereignis verursacht hat.</p>
no_initial_phase	<p>Legt fest, ob das Skript nach geschachtelten Ankern in der Hauptphase sucht. Sie können eine der folgenden Optionen auswählen:</p> <ul style="list-style-type: none"> - Gelöscht. Es erfolgt eine Suche nach geschachtelten Ankern entsprechend ihren individuellen Eigenschaften. - Ausgewählt. Es erfolgt eine Suche nach geschachtelten Ankern in der Hauptphase. <p>Die Standardoption lautet „Gelöscht“.</p>
Benachrichtigungen	<p>Eine Liste mit NotificationHandler-Komponenten, die Benachrichtigungen aus geschachtelten Komponenten verarbeiten. Weitere Informationen hierzu finden Sie unter "Benachrichtigungen" auf Seite 430.</p>
on_fail	<p>Die Aktion, die beim Ausfall der Komponente durchgeführt wird. Sie können eine der folgenden Optionen auswählen:</p> <ul style="list-style-type: none"> - Gelöscht. Es wird keine Aktion ausgeführt. - CustomLog. Es wird in das Benutzerprotokoll geschrieben. - LogError. Eine Fehlermeldung wird in das Engine-Protokoll geschrieben. - LogInfo. Eine Informationsmeldung wird in das Engine-Protokoll geschrieben. - LogWarning. Eine Warnmeldung wird in das Engine-Protokoll geschrieben. - NotifyFailure. Eine Mitteilung wird gesendet. <p>Die Standardoption lautet „Gelöscht“. Weitere Informationen über die Behandlung von Komponentengefehlern finden Sie in "Fehlerbehandlung" auf Seite 410.</p>
opening	<p>Definiert den öffnenden Anker der EnclosedGroup.</p>
optional	<p>Legt fest, ob ein Komponentengefehle den Ausfall der übergeordneten Komponente verursacht. Sie können eine der folgenden Optionen auswählen:</p> <ul style="list-style-type: none"> - Ausgewählt. Ein Komponentengefehle führt nicht zum Ausfall der übergeordneten Komponente. - Gelöscht. Ein Komponentengefehle verursacht den Ausfall der übergeordneten Komponente. <p>Standardwert ist "Gelöscht". Weitere Informationen über Komponentengefehle finden Sie in "Fehlerbehandlung" auf Seite 410.</p>
phase	<p>Legt den Zeitpunkt fest, an dem das Skript die Komponente verarbeitet. Sie können eine der folgenden Optionen auswählen:</p> <ul style="list-style-type: none"> - Anfangsphase. Das Skript verarbeitet die Komponente während der Anfangsphase. - Hauptphase. Das Skript verarbeitet die Komponente während der Hauptphase. - Endphase. Das Skript verarbeitet die Komponente während der Endphase. <p>Weitere Informationen hierzu finden Sie unter "So sucht der Parser Anker" auf Seite 211. Standardwert ist "Hauptphase".</p>
remark	<p>Ein benutzerdefinierter Kommentar, der den Zweck oder die Aktion der Komponente beschreibt.</p>

Eigenschaft	Beschreibung
source	<p>Definiert eine Sequenz von Datenbehältern zur Eingabe in die EnclosedGroup. Jeder Datenbehälter wird durch eine der folgenden Eigenschaften ausgewiesen:</p> <ul style="list-style-type: none"> - Locator. Identifiziert einen Einzel- oder Mehrfachinstanz-Datenbehälter. Bei Mehrfachinstanz-Datenbehältern greift jede Iteration auf eine neue Instanz zu. - LocatorByKey. Identifiziert einen Einzelinstanz-Datenbehälter nach Schlüssel. - LocatorByOccurence. Identifiziert einen Mehrinstanz-Datenbehälter nach Sequenznummer. <p>Verwenden Sie die Eigenschaft source, wenn die EnclosedGroup von einer anderen Komponente aufgerufen wird. Weitere Informationen hierzu finden Sie unter "Eigenschaft „source“" auf Seite 375.</p>
target	<p>Definiert eine Sequenz von Datenbehältern zur Ausgabe von der EnclosedGroup. Wenn der Datenbehälter noch nicht existiert, wird er erzeugt. Jeder Datenbehälter wird durch eine der folgenden Eigenschaften ausgewiesen:</p> <ul style="list-style-type: none"> - Locator. Identifiziert einen Einzel- oder Mehrfachinstanz-Datenbehälter. Bei Mehrfachinstanz-Datenbehältern erstellt jede Iteration eine neue Instanz. - LocatorByKey. Identifiziert einen Einzelinstanz-Datenbehälter nach Schlüssel. - LocatorByOccurence. Identifiziert einen Mehrinstanz-Datenbehälter nach Sequenznummer. <p>Verwenden Sie die Eigenschaft target, wenn die Ausgabe der EnclosedGroup von einer anderen Komponente aufgerufen wird. Weitere Informationen hierzu finden Sie unter "Eigenschaft „target“" auf Seite 378.</p>

Eine **EnclosedGroup** ähnelt einem **Content**-Anker mit einem **opening_marker** und einem **closing_marker**, allerdings mit folgenden Unterschieden:

- Der **Content**-Anker ruft den gesamten Inhalt aus dem Bereich zwischen den öffnenden und schließenden Markern ab, ohne ihn weiter zu parsen.
- Mit **EnclosedGroup** kann der Inhalt im Bereich zwischen dem **opening**- und dem **closing**-Anker weiter geparkt werden.

Beispiel

Sie können eine HTML-Tabelle als **EnclosedGroup** definieren, wobei die Tags `<table>` und `</table>` als öffnende und schließende Marker dienen. Die geschachtelten Anker parsen den Inhalt der Tabelle.

Angenommen, das Element `<table>` enthält ein geschachteltes `<table>`-Element. Mit anderen Worten, eine Tabelle ist in eine andere Tabelle geschachtelt. Der **EnclosedGroup**-Anker findet die zueinander gehörenden Eltern-Tags `<table>` und `</table>`. Er ordnet das Eltern-Tag `<table>` nicht dem geschachtelten Tag `</table>` zu, was eine fehlerhafte Interpretation der Tabelle zur Folge hätte.

So definieren Sie einen EnclosedGroup-Anker

Sie definieren einen **EnclosedGroup**-Anker, indem Sie das Skript im IntelliScript-Editor bearbeiten. Fügen Sie die geschachtelten Anker ein, die den Inhalt parsen.

ExtractRecord

Der **ExtractRecord**-Anker extrahiert einen Datensatz, weist dem Datensatz Bezeichner zu und reicht ihn an die Unterelemente einer **StructureDefinition** weiter. **ExtractRecord** wird in der **Format__Definition**-Eigenschaft einer **StructureDefinition** verwendet.

ExtractRecord extrahiert den gesamten Suchbereich. Beispiel: Wenn Sie einen **ExtractRecord** zwischen zwei **Marker**-Anker einfügen, wird der Bereich zwischen den Markern extrahiert.

Die nachstehende Tabelle beschreibt die Eigenschaften des **ExtractRecord**-Ankers:

Eigenschaft	Beschreibung
disabled	Legt fest, ob die Komponente und alle zugehörigen untergeordneten Komponenten vom Skript ignoriert werden. Verwenden Sie diese Eigenschaft zum Testen, Debuggen und Ändern eines Skripts. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none"> - Ausgewählt. Das Skript ignoriert die Komponente. - Gelöscht. Das Skript wendet die Komponente an. Standardmäßig ist die Eigenschaft deaktiviert.
ids	Definiert eine Liste von Bezeichnern, die an den Datensatz angehängt werden. StructureDefinition verwendet Bezeichner, um den Datensatz an ein Unterelement anzupassen. Geben Sie in jedem Listeneintrag einen Bezeichner-Wert ein, oder wählen Sie den Datenbehälter aus, der den gesuchten Wert enthält.
name	Eine beschreibende Kennzeichnung für die Komponente. Diese Kennzeichnung wird in der Protokolldatei sowie der Ereignisansicht angezeigt. Ermitteln Sie anhand der Eigenschaft name , welche Komponente das Ereignis verursacht hat.
optional	Legt fest, ob ein Komponentenfehler den Ausfall der übergeordneten Komponente verursacht. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none"> - Ausgewählt. Ein Komponentenfehler führt nicht zum Ausfall der übergeordneten Komponente. - Gelöscht. Ein Komponentenfehler verursacht den Ausfall der übergeordneten Komponente. Standardwert ist "Gelöscht". Weitere Informationen über Komponentenfehler finden Sie in "Fehlerbehandlung" auf Seite 410 .
phase	Legt den Zeitpunkt fest, an dem das Skript die Komponente verarbeitet. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none"> - Anfangsphase. Das Skript verarbeitet die Komponente während der Anfangsphase. - Hauptphase. Das Skript verarbeitet die Komponente während der Hauptphase. - Endphase. Das Skript verarbeitet die Komponente während der Endphase. Weitere Informationen hierzu finden Sie unter "So sucht der Parser Anker" auf Seite 211 . Standardwert ist "Hauptphase".
remark	Ein benutzerdefinierter Kommentar, der den Zweck oder die Aktion der Komponente beschreibt.

FindReplaceAnchor

Der **FindReplaceAnchor**-Anker markiert den Quelltext und gibt Ersatztext für die Umwandlung durch den **TransformByParser**-Transformer an.

Die nachstehende Tabelle beschreibt die Eigenschaften des Ankers **FindReplaceAnchor**:

Eigenschaft	Beschreibung
disabled	<p>Legt fest, ob die Komponente und alle zugehörigen untergeordneten Komponenten vom Skript ignoriert werden. Verwenden Sie diese Eigenschaft zum Testen, Debuggen und Ändern eines Skripts. Sie können eine der folgenden Optionen auswählen:</p> <ul style="list-style-type: none"> - Ausgewählt. Das Skript ignoriert die Komponente. - Gelöscht. Das Skript wendet die Komponente an. <p>Standardmäßig ist die Eigenschaft deaktiviert.</p>
marking	<p>Legt fest, ob ein Anker als Anfang des Suchbereichs für den danach folgenden Anker verwendet wird. Sie können eine der folgenden Optionen auswählen:</p> <ul style="list-style-type: none"> - Anfangsposition. Setzt einen Referenzpunkt vor dem aktuellen Anker. - Endposition. Setzt einen Referenzpunkt hinter dem aktuellen Anker. - Vollständig. Setzt einen Referenzpunkt vor und hinter dem aktuellen Anker. - Keine. Es wird kein Referenzpunkt erstellt. <p>Weitere Informationen hierzu finden Sie unter "So sucht der Parser Anker" auf Seite 211.</p>
name	<p>Eine beschreibende Kennzeichnung für die Komponente. Diese Kennzeichnung wird in der Protokolldatei sowie der Ereignisansicht angezeigt. Ermitteln Sie anhand der Eigenschaft name, welche Komponente das Ereignis verursacht hat.</p>
no_initial_phase	<p>Legt fest, ob das Skript nach geschachtelten Ankern in der Hauptphase sucht. Sie können eine der folgenden Optionen auswählen:</p> <ul style="list-style-type: none"> - Gelöscht. Es erfolgt eine Suche nach geschachtelten Ankern entsprechend ihren individuellen Eigenschaften. - Ausgewählt. Es erfolgt eine Suche nach geschachtelten Ankern in der Hauptphase. <p>Die Standardoption lautet „Gelöscht“.</p>
on_fail	<p>Die Aktion, die beim Ausfall der Komponente durchgeführt wird. Sie können eine der folgenden Optionen auswählen:</p> <ul style="list-style-type: none"> - Gelöscht. Es wird keine Aktion ausgeführt. - CustomLog. Es wird in das Benutzerprotokoll geschrieben. - LogError. Eine Fehlermeldung wird in das Engine-Protokoll geschrieben. - LogInfo. Eine Informationsmeldung wird in das Engine-Protokoll geschrieben. - LogWarning. Eine Warnmeldung wird in das Engine-Protokoll geschrieben. - NotifyFailure. Eine Mitteilung wird gesendet. <p>Die Standardoption lautet „Gelöscht“. Weitere Informationen über die Behandlung von Komponentenfehlern finden Sie in "Fehlerbehandlung" auf Seite 410.</p>
on_partial_match	<p>Bestimmt das Verhalten, wenn FindReplaceAnchor nicht alle geschachtelten nicht-optionalen Anker findet. Sie können eine der folgenden Optionen auswählen:</p> <ul style="list-style-type: none"> - Fehlschlag. FindReplaceAnchor schlägt fehl. Standard. - Überspringen. FindReplaceAnchor entfernt den von den erfolgreich geschachtelten Ankern abgedeckten Bereich aus dem Suchbereich und versucht erneut, alle geschachtelten Anker zu finden. Der Prozess wird wiederholt, bis der Anker gefunden wird oder ein Fehlschlag eintritt.
optional	<p>Legt fest, ob ein Komponentenfehler den Ausfall der übergeordneten Komponente verursacht. Sie können eine der folgenden Optionen auswählen:</p> <ul style="list-style-type: none"> - Ausgewählt. Ein Komponentenfehler führt nicht zum Ausfall der übergeordneten Komponente. - Gelöscht. Ein Komponentenfehler verursacht den Ausfall der übergeordneten Komponente. <p>Standardwert ist "Gelöscht". Weitere Informationen über Komponentenfehler finden Sie in "Fehlerbehandlung" auf Seite 410.</p>

Eigenschaft	Beschreibung
phase	<p>Legt den Zeitpunkt fest, an dem das Skript die Komponente verarbeitet. Sie können eine der folgenden Optionen auswählen:</p> <ul style="list-style-type: none"> - Anfangsphase. Das Skript verarbeitet die Komponente während der Anfangsphase. - Hauptphase. Das Skript verarbeitet die Komponente während der Hauptphase. - Endphase. Das Skript verarbeitet die Komponente während der Endphase. <p>Weitere Informationen hierzu finden Sie unter "So sucht der Parser Anker" auf Seite 211. Standardwert ist "Hauptphase".</p>
remark	Ein benutzerdefinierter Kommentar, der den Zweck oder die Aktion der Komponente beschreibt.
replace_with	Definiert einen literalen Ersetzungs-String oder einen Datenbehälter, der den Ersetzungs-String enthält.
source	<p>Definiert eine Sequenz von Datenbehältern zur Eingabe in den FindReplaceAnchor. Jeder Datenbehälter wird durch eine der folgenden Eigenschaften ausgewiesen:</p> <ul style="list-style-type: none"> - Locator. Identifiziert einen Einzel- oder Mehrfachinstanz-Datenbehälter. Bei Mehrfachinstanz-Datenbehältern greift jede Iteration auf eine neue Instanz zu. - LocatorByKey. Identifiziert einen Einzelinstanz-Datenbehälter nach Schlüssel. - LocatorByOccurence. Identifiziert einen Mehrinstanz-Datenbehälter nach Sequenznummer. <p>Verwenden Sie die Eigenschaft source, wenn der FindReplaceAnchor von einer anderen Komponente aufgerufen wird. Weitere Informationen hierzu finden Sie unter "Eigenschaft „source“" auf Seite 375.</p>
target	<p>Definiert eine Sequenz von Datenbehältern zur Ausgabe vom FindReplaceAnchor. Wenn der Datenbehälter noch nicht existiert, wird er erzeugt. Jeder Datenbehälter wird durch eine der folgenden Eigenschaften ausgewiesen:</p> <ul style="list-style-type: none"> - Locator. Identifiziert einen Einzel- oder Mehrfachinstanz-Datenbehälter. Bei Mehrfachinstanz-Datenbehältern erstellt jede Iteration eine neue Instanz. - LocatorByKey. Identifiziert einen Einzelinstanz-Datenbehälter nach Schlüssel. - LocatorByOccurence. Identifiziert einen Mehrinstanz-Datenbehälter nach Sequenznummer. <p>Verwenden Sie die Eigenschaft target, wenn die Ausgabe vom FindReplaceAnchor von einer anderen Komponente verwendet wird. Weitere Informationen hierzu finden Sie unter "Eigenschaft „target“" auf Seite 378.</p>

Wenn **FindReplaceAnchor** keine geschachtelten Anker enthält, ersetzt er den gesamten Text innerhalb des Suchbereichs. Befindet sich **FindReplaceAnchor** zum Beispiel zwischen zwei **Marker**-Ankern, wird der gesamte Text zwischen diesen Ankern markiert.

Wenn **FindReplaceAnchor** einen **Marker**-Anker enthält, markiert er den **Marker** zum Ersetzen.

Wenn **FindReplaceAnchor** zwei **Marker**-Anker enthält, markiert er die **Marker**-Anker und das Segment zwischen ihnen zum Ersetzen.

Der Ersetzungstext kann einen statischen Ersetzungs-String oder ein String sein, der dynamisch aus dem Quelldokument abgerufen wird.

Weitere Informationen hierzu finden Sie unter ["TransformByParser" auf Seite 296](#).

Beispiel

Sie möchten Zeilennummern einem Textdokument hinzufügen. Führen Sie zum Hinzufügen der Zeilennummern die folgenden Schritte aus:

1. Erstellen Sie einen Parser, und fügen Sie eine `RepeatingGroup` hinzu.
2. Fügen Sie innerhalb der `RepeatingGroup` einen `FindReplaceAnchor`-Anker hinzu.

3. Fügen Sie im `FindReplaceAnchor` einen Marker-Anker ein, und setzen Sie dessen Eigenschaft `search` auf `NewlineSearch`.
Vom `FindReplaceAnchor` wird nun jedes Zeilenvorschubzeichen im Dokument markiert.
4. Konfigurieren Sie den `RepeatingGroup`-Anker so, dass die `current_iteration` in einer Variablen gespeichert wird. Legen Sie die Variable als Wert für die Eigenschaft `replace_with` von `FindReplaceAnchor` fest.
5. Definieren Sie auf der globalen Ebene des Skripts einen `TransformByParser`-Transformer. Legen Sie den Parser als Wert für die Eigenschaft `parser` fest.
6. Legen Sie `TransformByParser` als Startkomponente der Umwandlung fest.
Der Transformer gibt eine geänderte Version der Originaldatei aus, die Zeilennummern enthält.

So definieren Sie einen FindReplaceAnchor-Anker

Sie definieren einen **FindReplaceAnchor**-Anker, indem Sie das Skript im IntelliScript-Editor bearbeiten. Falls erforderlich, können geschachtelte Anker hinzugefügt werden, die einen zu ersetzenden Teilstring markieren.

Group

Der Anker **Group** dient dazu, eine Folge von Ankern und Aktionen aneinander zu binden.

Eigenschaften des **Group**-Objekts betreffen alle untergeordneten Komponenten. Verwenden Sie ein **Group**-Objekt, um die Vorgänge zu definieren, die das Skript mit einer Reihe von Ankern ausführen soll, oder um die Phase der geschachtelten Anker zu steuern.

Die nachstehende Tabelle beschreibt die Eigenschaften des **Group**-Ankers:

Eigenschaft	Beschreibung
absent	Definiert das Verhalten des Group -Ankers, wenn einer der geschachtelten, nicht-optionalen Anker oder eine der Aktionen fehlschlägt. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none"> - Ausgewählt. Group schlägt fehl. - Gelöscht. Normales Verhalten. Mit dieser Eigenschaft können Sie prüfen, ob geschachtelte Anker fehlen.
disabled	Legt fest, ob die Komponente und alle zugehörigen untergeordneten Komponenten vom Skript ignoriert werden. Verwenden Sie diese Eigenschaft zum Testen, Debuggen und Ändern eines Skripts. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none"> - Ausgewählt. Das Skript ignoriert die Komponente. - Gelöscht. Das Skript wendet die Komponente an. Standardmäßig ist die Eigenschaft deaktiviert.
marking	Legt fest, ob ein Anker als Anfang des Suchbereichs für den danach folgenden Anker verwendet wird. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none"> - Anfangsposition. Setzt einen Referenzpunkt vor dem aktuellen Anker. - Endposition. Setzt einen Referenzpunkt hinter dem aktuellen Anker. - Vollständig. Setzt einen Referenzpunkt vor und hinter dem aktuellen Anker. - Keine. Es wird kein Referenzpunkt erstellt. Weitere Informationen hierzu finden Sie unter "So sucht der Parser Anker" auf Seite 211 .
name	Eine beschreibende Kennzeichnung für die Komponente. Diese Kennzeichnung wird in der Protokolldatei sowie der Ereignisansicht angezeigt. Ermitteln Sie anhand der Eigenschaft name , welche Komponente das Ereignis verursacht hat.

Eigenschaft	Beschreibung
no_initial_phase	<p>Legt fest, ob das Skript nach geschachtelten Anker in der Hauptphase sucht. Sie können eine der folgenden Optionen auswählen:</p> <ul style="list-style-type: none"> - Gelöscht. Es erfolgt eine Suche nach geschachtelten Anker entsprechend ihren individuellen Eigenschaften. - Ausgewählt. Es erfolgt eine Suche nach geschachtelten Anker in der Hauptphase. Die Standardoption lautet „Gelöscht“.
Benachrichtigungen	<p>Eine Liste mit NotificationHandler-Komponenten, die Benachrichtigungen aus geschachtelten Komponenten verarbeiten. Weitere Informationen hierzu finden Sie unter "Benachrichtigungen" auf Seite 430.</p>
on_fail	<p>Die Aktion, die beim Ausfall der Komponente durchgeführt wird. Sie können eine der folgenden Optionen auswählen:</p> <ul style="list-style-type: none"> - Gelöscht. Es wird keine Aktion ausgeführt. - CustomLog. Es wird in das Benutzerprotokoll geschrieben. - LogError. Eine Fehlermeldung wird in das Engine-Protokoll geschrieben. - LogInfo. Eine Informationsmeldung wird in das Engine-Protokoll geschrieben. - LogWarning. Eine Warnmeldung wird in das Engine-Protokoll geschrieben. - NotifyFailure. Eine Mitteilung wird gesendet. <p>Die Standardoption lautet „Gelöscht“. Weitere Informationen über die Behandlung von Komponentenfehlern finden Sie in "Fehlerbehandlung" auf Seite 410.</p>
on_partial_match	<p>Bestimmt das Verhalten, wenn das Group-Objekt nicht alle geschachtelten nicht-optionalen Anker findet. Sie können eine der folgenden Optionen auswählen:</p> <ul style="list-style-type: none"> - Fehlschlag. Group schlägt fehl. Standard. - Überspringen. Das Group-Objekt entfernt den von den erfolgreich geschachtelten Anker abgedeckten Bereich aus dem Suchbereich und versucht erneut, alle geschachtelten Anker zu finden. Der Prozess wird wiederholt, bis der Anker gefunden wird oder ein Fehlschlag eintritt.
optional	<p>Legt fest, ob ein Komponentenfehler den Ausfall der übergeordneten Komponente verursacht. Sie können eine der folgenden Optionen auswählen:</p> <ul style="list-style-type: none"> - Ausgewählt. Ein Komponentenfehler führt nicht zum Ausfall der übergeordneten Komponente. - Gelöscht. Ein Komponentenfehler verursacht den Ausfall der übergeordneten Komponente. Standardwert ist "Gelöscht". Weitere Informationen über Komponentenfehler finden Sie in "Fehlerbehandlung" auf Seite 410.
phase	<p>Legt den Zeitpunkt fest, an dem das Skript die Komponente verarbeitet. Sie können eine der folgenden Optionen auswählen:</p> <ul style="list-style-type: none"> - Anfangsphase. Das Skript verarbeitet die Komponente während der Anfangsphase. - Hauptphase. Das Skript verarbeitet die Komponente während der Hauptphase. - Endphase. Das Skript verarbeitet die Komponente während der Endphase. <p>Weitere Informationen hierzu finden Sie unter "So sucht der Parser Anker" auf Seite 211. Standardwert ist "Hauptphase".</p>
remark	<p>Ein benutzerdefinierter Kommentar, der den Zweck oder die Aktion der Komponente beschreibt.</p>
search_order	<p>Legt die Verarbeitungsrichtung für die geschachtelten Anker fest. Sie können eine der folgenden Optionen auswählen:</p> <ul style="list-style-type: none"> - Von oben nach unten. Die geschachtelten Anker werden in der Reihenfolge verarbeitet, die im Skript definiert ist. - Von unten nach oben. Die geschachtelten Anker werden in umgekehrter Reihenfolge verarbeitet. Dies ist dann nützlich, wenn Daten aus einem späteren Anker Auswirkungen auf die Verarbeitung eines früheren Ankers haben.

Eigenschaft	Beschreibung
source	<p>Definiert eine Sequenz von Datenbehältern zur Eingabe in das Group-Objekt. Jeder Datenbehälter wird durch eine der folgenden Eigenschaften ausgewiesen:</p> <ul style="list-style-type: none"> - Locator. Identifiziert einen Einzel- oder Mehrfachinstanz-Datenbehälter. Bei Mehrfachinstanz-Datenbehältern greift jede Iteration auf eine neue Instanz zu. - LocatorByKey. Identifiziert einen Einzelinstanz-Datenbehälter nach Schlüssel. - LocatorByOccurence. Identifiziert einen Mehrinstanz-Datenbehälter nach Sequenznummer. <p>Verwenden Sie die Eigenschaft source, wenn das Group-Objekt von einer anderen Komponente aufgerufen wird. Weitere Informationen hierzu finden Sie unter "Eigenschaft „source“" auf Seite 375.</p>
target	<p>Definiert eine Sequenz von Datenbehältern zur Ausgabe vom Group-Objekt. Wenn der Datenbehälter noch nicht existiert, wird er erzeugt. Jeder Datenbehälter wird durch eine der folgenden Eigenschaften ausgewiesen:</p> <ul style="list-style-type: none"> - Locator. Identifiziert einen Einzel- oder Mehrfachinstanz-Datenbehälter. Bei Mehrfachinstanz-Datenbehältern erstellt jede Iteration eine neue Instanz. - LocatorByKey. Identifiziert einen Einzelinstanz-Datenbehälter nach Schlüssel. - LocatorByOccurence. Identifiziert einen Mehrinstanz-Datenbehälter nach Sequenznummer. <p>Verwenden Sie die Eigenschaft target, wenn die Ausgabe vom Group-Objekt von einer anderen Komponente verwendet wird. Weitere Informationen hierzu finden Sie unter "Eigenschaft „target“" auf Seite 378.</p>

So definieren Sie einen Group-Anker

Sie definieren einen **Group**-Anker, indem Sie das Skript im IntelliScript-Editor bearbeiten. Fügen Sie geschachtelte Anker und Aktionen hinzu, die den Inhalt der **Group** parsen.

Optionale Gruppe

Mit der Eigenschaft **optional** von **Group** können Sie verhindern, dass das Skript Text aus einem fehlenden Dokumentabschnitt abzurufen versucht.

Angenommen, Sie möchten die folgende Quelle parsen:

```
First name: Ron
```

Dafür definieren Sie `First name:` als **Marker** und `Ron` als **Content**. Einige Quelldokumente enthalten keine Vornamen. Daher fassen Sie **Marker** und **Content** in einer **Group** zusammen und definieren diese als optional. Wenn `First name:` nicht gefunden wird, schlägt **Group** sofort fehl und der Parser sucht nicht nach dem **Content**-Anker.

Es macht einen Unterschied, ob Sie eine **Group** oder ihre geschachtelten Anker als optional definieren. Wenn anstelle der **Group** sowohl der **Marker** als auch der **Content** optional sind, ignoriert das Skript das Fehlschlagen von **Marker** und sucht den **Content**. Das kann dazu führen, dass irrelevanter Text abgerufen wird.

Online-Beispiel

Ein Online-Beispiel zu diesem Anker befindet sich im Projekt `samples\Projects\persistent_search\persistent_search.cmw`.

Das Beispiel beschreibt einen **Group**-Anker, der mit der Eigenschaft `on_partial_match = skip` konfiguriert ist. Die **Group** enthält zwei **Marker**-Anker:

- Der erste **Marker** sucht nach dem Text `A`.
- Der zweite **Marker** sucht nach einem String mit dem Zeichen `*` in beliebiger Anzahl. Die Eigenschaft `adjacent` dieses **Marker**-Ankers gibt vor, dass er an den ersten **Marker** angrenzen muss.

Im ersten Durchgang findet der `Group`-Anker am Anfang des Quelldokuments ein Zeichen `A`. Allerdings findet er keinen zweiten `Marker` neben dem Zeichen `A`.

Daher verkleinert `Group` den Suchbereich durch Ausschließen des ersten Zeichens `A` und sucht erneut nach zwei aneinander angrenzenden `Marker`-Ankern. Dieses Verfahren wird so lange fortgesetzt, bis ein String `A*` gefunden wird, der die beiden `Marker`-Anker enthält.

Dieses Verhalten können Sie im Ereignisprotokoll verfolgen. Im Ereignisprotokoll wird aufgezeichnet, dass `Group` bei den ersten beiden Versuchen fehlgeschlagen ist, beim dritten jedoch erfolgreich ausgeführt wurde.

Experimentieren Sie mit den Einstellungen von `on_partial_match` und `adjacent`. Die Auswirkungen können Sie an der Farbcodierung in der Beispielquelle erkennen.

Sie können auch das Beispiel ausführen. Dessen Ergebnisdatei bleibt jedoch leer, da der Parser keine `Content`-Anker enthält. Wenn Sie `on_partial_match = fail` setzen, können Sie im Ereignisprotokoll nachvollziehen, dass der Parser fehlschlägt, da `Group` die aneinander angrenzenden Anker nicht finden kann.

Marker

Ein **Marker**-Anker definiert eine Position in einem Quelldokument. Er dient als Referenzpunkt, von dem aus das Skript die danach folgenden Anker sucht.

Standardmäßig ist die Eigenschaft **phase** eines **Marker** auf `initial` gesetzt. Dies bedeutet, dass das Skript ein Dokument nach **Marker**-Ankern durchsucht, bevor es nach **Content**-Ankern sucht. Weitere Informationen hierzu finden Sie unter ["So sucht der Parser Anker" auf Seite 211](#).

Die nachstehende Tabelle beschreibt die Eigenschaften des **Marker**-Ankers:

Eigenschaft	Beschreibung
<code>absent</code>	Bestimmt, ob der angegebene Text oder das angegebene Muster im Dokument fehlt. Die Eigenschaft absent hat folgende Optionen: <ul style="list-style-type: none">- Ausgewählt. Wenn der angegebene Text im Dokument auftaucht, schlägt die Marker-Komponente fehl.- Gelöscht. Wenn der angegebene Text im Dokument auftaucht, ist die Marker-Komponente erfolgreich. Standardwert ist "Gelöscht".
<code>adjacent</code>	Wenn diese Eigenschaft ausgewählt ist, muss der Marker an den Anker am Anfang seines Suchbereichs angrenzen. Wenn Eigenschaft direction auf <code>backward</code> gesetzt ist, muss er an den Anker am Ende seines Suchbereichs angrenzen. Ist die Eigenschaft nicht ausgewählt, kann das Skript Text überspringen, bis es den Marker findet. Die Eigenschaft adjacent hat folgende Optionen: <ul style="list-style-type: none">- Ausgewählt. Der Marker muss unmittelbar nach dem Anfang des Suchbereichs auftreten, wenn direction auf <code>forward</code> gesetzt ist, oder unmittelbar vor dem Ende des Suchbereichs, wenn direction auf <code>backward</code> gesetzt ist.- Gelöscht. Der Marker kann sich an beliebiger Stelle innerhalb des Suchbereichs befinden. Standardwert ist "Gelöscht".
<code>count</code>	Gibt an, die wievielte Instanz gesucht werden soll. Um den Marker zum Beispiel auf das zweite Zeilenvorschubzeichen nach dem vorherigen Anker zu setzen, definieren Sie search als <code>NewlineSearch</code> und count als 2.

Eigenschaft	Beschreibung
direction	<p>Eine Suchrichtung für den Anker innerhalb des Suchbereichs. Sie können eine der folgenden Optionen auswählen:</p> <ul style="list-style-type: none"> - Zurück. Die Suche beginnt am Ende des Suchbereichs und findet die letzte Instanz des Ankers. - Vorwärtssuche. Die Suche beginnt am Anfang des Suchbereichs und findet die erste Instanz des Ankers. <p>Bei einem Marker-Anker können Sie dieses Verhalten mithilfe der Eigenschaft count ändern. Wenn beispielsweise direction = backward und count = 2 ist, findet das Skript die vorletzte Instanz.</p> <p>Standardwert ist "Vorwärtssuche". Weitere Informationen hierzu finden Sie unter "So sucht der Parser Anker" auf Seite 211.</p>
disabled	<p>Legt fest, ob die Komponente und alle zugehörigen untergeordneten Komponenten vom Skript ignoriert werden. Verwenden Sie diese Eigenschaft zum Testen, Debuggen und Ändern eines Skripts. Sie können eine der folgenden Optionen auswählen:</p> <ul style="list-style-type: none"> - Ausgewählt. Das Skript ignoriert die Komponente. - Gelöscht. Das Skript wendet die Komponente an. <p>Standardmäßig ist die Eigenschaft deaktiviert.</p>
marking	<p>Legt fest, ob ein Anker als Anfang des Suchbereichs für den danach folgenden Anker verwendet wird. Sie können eine der folgenden Optionen auswählen:</p> <ul style="list-style-type: none"> - Anfangsposition. Setzt einen Referenzpunkt vor dem aktuellen Anker. - Endposition. Setzt einen Referenzpunkt hinter dem aktuellen Anker. - Vollständig. Setzt einen Referenzpunkt vor und hinter dem aktuellen Anker. - Keine. Es wird kein Referenzpunkt erstellt. <p>Weitere Informationen hierzu finden Sie unter "So sucht der Parser Anker" auf Seite 211.</p>
name	<p>Eine beschreibende Kennzeichnung für die Komponente. Diese Kennzeichnung wird in der Protokolldatei sowie der Ereignisansicht angezeigt. Ermitteln Sie anhand der Eigenschaft name, welche Komponente das Ereignis verursacht hat.</p>
on_fail	<p>Die Aktion, die beim Ausfall der Komponente durchgeführt wird. Sie können eine der folgenden Optionen auswählen:</p> <ul style="list-style-type: none"> - Gelöscht. Es wird keine Aktion ausgeführt. - CustomLog. Es wird in das Benutzerprotokoll geschrieben. - LogError. Eine Fehlermeldung wird in das Engine-Protokoll geschrieben. - LogInfo. Eine Informationsmeldung wird in das Engine-Protokoll geschrieben. - LogWarning. Eine Warnmeldung wird in das Engine-Protokoll geschrieben. - NotifyFailure. Eine Mitteilung wird gesendet. <p>Die Standardoption lautet „Gelöscht“. Weitere Informationen über die Behandlung von Komponentenfehlern finden Sie in "Fehlerbehandlung" auf Seite 410.</p>
optional	<p>Legt fest, ob ein Komponentenfehler den Ausfall der übergeordneten Komponente verursacht. Sie können eine der folgenden Optionen auswählen:</p> <ul style="list-style-type: none"> - Ausgewählt. Ein Komponentenfehler führt nicht zum Ausfall der übergeordneten Komponente. - Gelöscht. Ein Komponentenfehler verursacht den Ausfall der übergeordneten Komponente. <p>Standardwert ist "Gelöscht". Weitere Informationen über Komponentenfehler finden Sie in "Fehlerbehandlung" auf Seite 410.</p>
phase	<p>Legt den Zeitpunkt fest, an dem das Skript die Komponente verarbeitet. Sie können eine der folgenden Optionen auswählen:</p> <ul style="list-style-type: none"> - Anfangsphase. Das Skript verarbeitet die Komponente während der Anfangsphase. - Hauptphase. Das Skript verarbeitet die Komponente während der Hauptphase. - Endphase. Das Skript verarbeitet die Komponente während der Endphase. <p>Weitere Informationen hierzu finden Sie unter "So sucht der Parser Anker" auf Seite 211. Standardwert ist "Anfangsphase".</p>

Eigenschaft	Beschreibung
remark	Ein benutzerdefinierter Kommentar, der den Zweck oder die Aktion der Komponente beschreibt.
search	<p>Definiert die Suchkriterien für den Marker. Die Suchkriterien bestimmen, an welcher Position innerhalb des Suchbereichs sich der Marker befindet. Beispielsweise platziert NewlineSearch den Marker bei einem Zeilenvorschubzeichen. TextSearch platziert den Marker bei einem angegebenen String. Weitere Informationen hierzu finden Sie unter "So sucht der Parser Anker" auf Seite 211.</p> <p>Den Wert dieser Eigenschaft bildet eine der folgenden Suchkomponenten.</p> <ul style="list-style-type: none"> - NewlineSearch. Sucht ein Zeilenvorschubzeichen. - TextSearch. Sucht nach einem vordefinierten Textstring oder nach einem Textstring, der in einem Datenbehälter gespeichert ist. - PatternSearch. Sucht nach einem Textstring, der mit einem angegebenen regulären Ausdruck übereinstimmt. - OffsetSearch. Überspringt eine vordefinierte Anzahl von Zeichen nach einem Referenzpunkt oder eine in einem Datenbehälter gespeicherte Anzahl von Zeichen. Der Marker ist die Stelle, die nach den übersprungenen Zeichen folgt. - TypeSearch. Sucht nach einem String, der mit einem angegebenen XSD-Datentyp übereinstimmt. <p>Weitere Informationen hierzu finden Sie im Abschnitt "Suchkomponenten: Referenz" auf Seite 248.</p>

So definieren Sie einen Marker-Anker

Sie definieren einen **Marker**-Anker, indem Sie das Skript im IntelliScript-Editor bearbeiten. Weitere Informationen hierzu finden Sie unter ["Definieren von Ankern" auf Seite 208](#).

Online-Beispiel

Öffnen Sie im Ordner „Online Samples“ die Datei `Projects\Markers\Markers.cmw`. Das Beispiel zeigt Marker-Anker, die nach folgenden Elementen suchen:

- einem vordefinierten Textstring
- einem Zeilenvorschubzeichen
- einem Abstand
- einem Datentyp
- einem regulären Ausdruck

Wenn Sie den Parser ausführen, bleibt die Ergebnisdatei leer, da in der Konfiguration keine **Content**-Anker definiert sind.

RepeatingGroup

Der **RepeatingGroup**-Anker parst einen Bereich, der sich wiederholende Segmente enthält. Jedes Segment wird als Iteration bezeichnet und kann durch einen **Separator** begrenzt werden. Der **RepeatingGroup**-Anker enthält eine Folge geschachtelter Anker und bzw. Aktionen, die die einzelnen Iterationen auf dieselbe Weise parsen.

Die nachstehende Tabelle beschreibt die Eigenschaften des Ankers **RepeatingGroup**:

Eigenschaft	Beschreibung
count	<p>Definiert eine Anzahl oder einen Datenbehälter, der die auszuführende Anzahl der Iterationen enthält. Wenn diese Eigenschaft leer bleibt, werden die Iterationen so lange fortgesetzt, bis der Suchbereich ausgeschöpft ist.</p> <p>Bei count = 0 sucht die RepeatingGroup nicht nach Iterationen. In diesem Fall wird die RepeatingGroup zwar erfolgreich ausgeführt, erzeugt jedoch keine Ausgabe.</p>
current_iteration	Definiert einen Datenbehälter, in den die RepeatingGroup die Nummer der aktuellen Iteration ausgibt.
disabled	<p>Legt fest, ob die Komponente und alle zugehörigen untergeordneten Komponenten vom Skript ignoriert werden. Verwenden Sie diese Eigenschaft zum Testen, Debuggen und Ändern eines Skripts. Sie können eine der folgenden Optionen auswählen:</p> <ul style="list-style-type: none"> - Ausgewählt. Das Skript ignoriert die Komponente. - Gelöscht. Das Skript wendet die Komponente an. <p>Standardmäßig ist die Eigenschaft deaktiviert.</p>
iteration_order	<p>Definiert die Reihenfolge, in der die Iterationen verarbeitet werden. Sie können eine der folgenden Optionen auswählen:</p> <ul style="list-style-type: none"> - Von oben nach unten. Die Iterationen werden in der Reihenfolge verarbeitet, die im Skript definiert ist. - Von unten nach oben. Die Iterationen werden in umgekehrter Reihenfolge verarbeitet. <p>Verwenden Sie diese Option, wenn Daten aus einer späteren Iteration Auswirkungen auf die Verarbeitung einer früheren Iteration haben.</p>
marking	<p>Legt fest, ob ein Anker als Anfang des Suchbereichs für den danach folgenden Anker verwendet wird. Sie können eine der folgenden Optionen auswählen:</p> <ul style="list-style-type: none"> - Anfangsposition. Setzt einen Referenzpunkt vor dem aktuellen Anker. - Endposition. Setzt einen Referenzpunkt hinter dem aktuellen Anker. - Vollständig. Setzt einen Referenzpunkt vor und hinter dem aktuellen Anker. - Keine. Es wird kein Referenzpunkt erstellt. <p>Weitere Informationen hierzu finden Sie unter "So sucht der Parser Anker" auf Seite 211.</p>
name	Eine beschreibende Kennzeichnung für die Komponente. Diese Kennzeichnung wird in der Protokolldatei sowie der Ereignisansicht angezeigt. Ermitteln Sie anhand der Eigenschaft name , welche Komponente das Ereignis verursacht hat.
no_initial_phase	<p>Legt fest, ob das Skript nach geschachtelten Ankern in der Hauptphase sucht. Sie können eine der folgenden Optionen auswählen:</p> <ul style="list-style-type: none"> - Gelöscht. Es erfolgt eine Suche nach geschachtelten Ankern entsprechend ihren individuellen Eigenschaften. - Ausgewählt. Es erfolgt eine Suche nach geschachtelten Ankern in der Hauptphase. <p>Die Standardoption lautet „Gelöscht“.</p>
Benachrichtigungen	Eine Liste mit NotificationHandler -Komponenten, die Benachrichtigungen aus geschachtelten Komponenten verarbeiten. Weitere Informationen hierzu finden Sie unter "Benachrichtigungen" auf Seite 430 .

Eigenschaft	Beschreibung
on_fail	<p>Die Aktion, die beim Ausfall der Komponente durchgeführt wird. Sie können eine der folgenden Optionen auswählen:</p> <ul style="list-style-type: none"> - Gelöscht. Es wird keine Aktion ausgeführt. - CustomLog. Es wird in das Benutzerprotokoll geschrieben. - LogError. Eine Fehlermeldung wird in das Engine-Protokoll geschrieben. - LogInfo. Eine Informationsmeldung wird in das Engine-Protokoll geschrieben. - LogWarning. Eine Warnmeldung wird in das Engine-Protokoll geschrieben. - NotifyFailure. Eine Mitteilung wird gesendet. <p>Die Standardoption lautet „Gelöscht“. Weitere Informationen über die Behandlung von Komponentengefehlern finden Sie in "Fehlerbehandlung" auf Seite 410.</p>
on_iteration_fail	<p>Definiert die Aktion, wenn eine einzelne Iteration fehlschlägt. Sie können eine der folgenden Optionen auswählen:</p> <ul style="list-style-type: none"> - Gelöscht. Keine Aktion. - CustomLog. Schreibt in das Benutzerprotokoll. - LogError. Schreibt eine Fehlermeldung in das Engine-Protokoll. - LogInfo. Schreibt eine Informationsmeldung in das Engine-Protokoll. - LogWarning. Schreibt eine Warnmeldung in das Engine-Protokoll. - NotifyFailure. Löst eine Benachrichtigung aus. <p>Verwenden Sie die Eigenschaft on_fail zum Schreiben eines Eintrags, wenn die gesamte RepeatingGroup fehlschlägt. Weitere Informationen hierzu finden Sie unter "Fehlerbehandlung" auf Seite 410.</p>
on_partial_match	<p>Definiert das Verhalten, wenn einige, aber nicht alle der unter der RepeatingGroup geschachtelten Anker in der Eingabe erscheinen. Die Eigenschaft on_partial_match hat folgende Optionen:</p> <ul style="list-style-type: none"> - Fehlschlag. Die Iteration schlägt fehl. - Überspringen. RepeatingGroup entfernt den Bereich, der von erfolgreichen geschachtelten Ankern abgedeckt wird, aus dem Suchbereich und versucht, alle geschachtelten Anker erneut zu finden. Die Prozedur des Ausschließens und erneuten Suchens wird so lange wiederholt, bis die Iteration erfolgreich verläuft oder keine weitere teilweise Übereinstimmung mehr vorhanden ist. Wenn keine Teilübereinstimmung vorliegt, schlägt die Iteration fehl.
optional	<p>Legt fest, ob ein Komponentengefehler den Ausfall der übergeordneten Komponente verursacht. Sie können eine der folgenden Optionen auswählen:</p> <ul style="list-style-type: none"> - Ausgewählt. Ein Komponentengefehler führt nicht zum Ausfall der übergeordneten Komponente. - Gelöscht. Ein Komponentengefehler verursacht den Ausfall der übergeordneten Komponente. <p>Standardwert ist "Gelöscht". Weitere Informationen über Komponentengefehler finden Sie in "Fehlerbehandlung" auf Seite 410.</p>
phase	<p>Legt den Zeitpunkt fest, an dem das Skript die Komponente verarbeitet. Sie können eine der folgenden Optionen auswählen:</p> <ul style="list-style-type: none"> - Anfangsphase. Das Skript verarbeitet die Komponente während der Anfangsphase. - Hauptphase. Das Skript verarbeitet die Komponente während der Hauptphase. - Endphase. Das Skript verarbeitet die Komponente während der Endphase. <p>Weitere Informationen hierzu finden Sie unter "So sucht der Parser Anker" auf Seite 211. Standardwert ist "Hauptphase".</p>
remark	<p>Ein benutzerdefinierter Kommentar, der den Zweck oder die Aktion der Komponente beschreibt.</p>

Eigenschaft	Beschreibung
search_order	<p>Definiert die Reihenfolge der Verarbeitung der geschachtelten Anker in jeder Iteration. Sie können eine der folgenden Optionen auswählen:</p> <ul style="list-style-type: none"> - Von oben nach unten. Die geschachtelten Anker werden in der Reihenfolge verarbeitet, die im Skript definiert ist. - Von unten nach oben. Die geschachtelten Anker werden in umgekehrter Reihenfolge verarbeitet. Wählen Sie diese Option, wenn Daten aus einem späteren Anker Auswirkungen auf die Verarbeitung eines früheren Ankers haben.
separator	<p>Definiert einen Anker, der die Abschnitte voneinander trennt.</p> <p>Wenn Sie die Eigenschaft separator leer lassen, sucht der RepeatingGroup-Anker zwischen den Iterationen nicht nach einem Delimiter. Stattdessen geht er davon aus, dass eine Iteration abgeschlossen ist, wenn er alle geschachtelten Anker gefunden hat. Anschließend parst er die nächste Iteration vom oberen Ende der Folge geschachtelter Anker aus.</p> <p>Sie können ein komplexes Trennzeichen erstellen, indem Sie in die Eigenschaft separator eine Group anstelle eines Marker einfügen.</p>
separator_position	<p>Definiert die Positionierung des Trennzeichens relativ zu den Abschnitten. Sie können eine der folgenden Optionen auswählen:</p> <ul style="list-style-type: none"> - Nachher. Ein Trennzeichen steht nach jedem Abschnitt, auch nach dem letzten. Beispiel: 1 2 3 4 - Um. Ein Trennzeichen steht vor und nach jedem Abschnitt, auch vor dem ersten und nach dem letzten Abschnitt. Beispiel: 1 2 3 4 - Vorher. Ein Trennzeichen steht vor jedem Abschnitt, auch vor dem ersten. Beispiel: 1 2 3 4 - Zwischen. Ein Trennzeichen steht jeweils zwischen aufeinander folgenden Abschnitten, jedoch nicht vor dem ersten und nach dem letzten Abschnitt. Beispiel: 1 2 3 4
skip_failed_iterations	<p>Legt fest, ob fehlgeschlagene Iterationen übersprungen werden. Sie können eine der folgenden Optionen auswählen:</p> <ul style="list-style-type: none"> - Ausgewählt. RepeatingGroup überspringt eine fehlgeschlagene Iteration und fährt mit der nächsten Iteration fort. Wenn eine Iteration erfolgreich ist, ist auch RepeatingGroup. - Gelöscht. RepeatingGroup schlägt fehl, wenn eine Iteration fehlschlägt. <p>Die Eigenschaft skip_failed_iterations hat nur dann einen Effekt, wenn separator definiert ist. Standardwert ist "Ausgewählt".</p>
source	<p>Definiert eine Sequenz von Datenbehältern für die Eingabe in RepeatingGroup. Jeder Datenbehälter wird durch eine der folgenden Eigenschaften ausgewiesen:</p> <ul style="list-style-type: none"> - Locator. Identifiziert einen Einzel- oder Mehrfachinstanz-Datenbehälter. Bei Mehrfachinstanz-Datenbehältern greift jede Iteration auf eine neue Instanz zu. - LocatorByKey. Identifiziert einen Einzelinstanz-Datenbehälter nach Schlüssel. - LocatorByOccurrence. Identifiziert einen Mehrinstanz-Datenbehälter nach Sequenznummer. <p>Verwenden Sie die Eigenschaft source, wenn die RepeatingGroup von einer anderen Komponente aufgerufen wird. Weitere Informationen hierzu finden Sie unter "Eigenschaft „source“ auf Seite 375.</p>
target	<p>Definiert eine Sequenz von Datenbehältern für die Ausgabe aus dem RepeatingGroup. Wenn der Datenbehälter noch nicht existiert, wird er erzeugt. Jeder Datenbehälter wird durch eine der folgenden Eigenschaften ausgewiesen:</p> <ul style="list-style-type: none"> - Locator. Identifiziert einen Einzel- oder Mehrfachinstanz-Datenbehälter. Bei Mehrfachinstanz-Datenbehältern erstellt jede Iteration eine neue Instanz. - LocatorByKey. Identifiziert einen Einzelinstanz-Datenbehälter nach Schlüssel. - LocatorByOccurrence. Identifiziert einen Mehrinstanz-Datenbehälter nach Sequenznummer. <p>Verwenden Sie die Eigenschaft target, wenn die Ausgabe von RepeatingGroup von einer anderen Komponente benutzt wird. Weitere Informationen hierzu finden Sie unter "Eigenschaft „target“ auf Seite 378.</p>

Hinweis: Um einen Bereich von Abschnitten zu parsen, die eine andere Verarbeitung benötigen, verwenden Sie einen **DelimitedSections**-Anker.

So definieren Sie einen RepeatingGroup-Anker

Sie definieren einen **RepeatingGroup**-Anker, indem Sie das Skript im IntelliScript-Editor bearbeiten. Fügen Sie geschachtelte Anker und Aktionen hinzu, die die einzelnen Iterationen der **RepeatingGroup** parsen.

Iterationen suchen

Standardmäßig sucht `RepeatingGroup` vom Anfang bis zum Ende seines Suchbereichs nach Iterationen. Weitere Informationen hierzu finden Sie unter ["So sucht der Parser Anker" auf Seite 211](#).

Auf Wunsch können Sie mit der Eigenschaft `iteration_order` eine umgekehrte Suchreihenfolge definieren.

In jeder Iteration erfolgt die Suche auf dieselbe Weise:

- Wenn die `RepeatingGroup` mit einem `separator` konfiguriert ist, wird das nächste Trennzeichen gesucht. Anschließend werden die Anker gesucht, die zwischen einem Trennzeichenpaar liegen.
- Ist die `RepeatingGroup` nicht mit einem `separator` konfiguriert, werden nur die Anker gesucht.

Ende einer RepeatingGroup

Das Ende einer `RepeatingGroup` können Sie unter anderem auf folgende Arten kennzeichnen:

- Die `RepeatingGroup` setzt sich bis zum Ende des Dokuments fort.
- Sie können einen `Marker` hinter der `RepeatingGroup` einfügen. Standardmäßig ist der `Marker` für eine frühere Suchphase konfiguriert als die `RepeatingGroup`. Daher sucht der Parser zuerst nach dem `Marker` und verwendet diesen anschließend zum Begrenzen des Suchbereichs der `RepeatingGroup`. Weitere Informationen finden Sie unter ["Anpassen der Suchphase" auf Seite 212](#).
- Sie können die Eigenschaft `count` definieren, um die Suche auf eine bestimmte Anzahl von Iterationen zu beschränken.
- Falls die `RepeatingGroup` keinen `separator` hat, endet sie dann, wenn der Parser keine weiteren Iterationen mehr findet.

Erfolgreiche Ausführung oder Fehlschlagen einer RepeatingGroup

Findet eine **RepeatingGroup** die nicht-optionalen Anker in einer Iteration nicht, schlägt die Iteration fehl.

Bei einem Fehlschlagen einer Iteration kann die **RepeatingGroup** entweder enden, fehlschlagen oder die fehlgeschlagene Iteration überspringen. Dabei gelten für das Verhalten folgende Regeln:

- Wenn die **RepeatingGroup** keinen **separator** hat, endet sie. Sofern vor der fehlgeschlagenen Iteration mindestens eine Iteration erfolgreich ausgeführt wurde, wird auch die **RepeatingGroup** insgesamt erfolgreich ausgeführt.
- Wenn die **RepeatingGroup** einen **separator** hat und die Eigenschaft **skip_failed_iterations** Nicht ausgewählt ist, schlägt die **RepeatingGroup** fehl.
- Wenn die **RepeatingGroup** ein Trennzeichen enthält und die Eigenschaft **skip_failed_iterations** ausgewählt ist, überspringt das Skript die fehlgeschlagene Iteration und fährt mit der nächsten fort. Sofern mindestens eine Iteration erfolgreich ausgeführt wird, wird auch die **RepeatingGroup** insgesamt erfolgreich ausgeführt.

Ereignisprotokoll einer Wiederholungsgruppe

Im Ereignisprotokoll werden für jede Iteration einer **RepeatingGroup** Ereignisse aufgezeichnet.

Wenn die Eigenschaft **skip_failed_iterations** ausgewählt ist, generiert die **RepeatingGroup** nach den erfolgreichen Iterationen unter Umständen ein optionales Fehlschlagereignis. Ein Fehlschlagereignis kann in das optionale Fehlschlagereignis geschachtelt sein. Diese Ereignisse treten auf, wenn die **RepeatingGroup** keine weiteren Iterationen zum Parsen findet. Diese Ereignisse sind normal und kein Anlass zur Sorge.

Online-Beispiele

Ein Online-Beispiel zu diesem Anker befindet sich im Projekt `samples\Projects\Dynamic_And_RepeatingGroup\Dynamic_And_RepeatingGroup.cmw`. Im Beispiel wird gezeigt, wie eine **RepeatingGroup** über die Zeilen eines Dokuments iteriert.

Einige Zeilen des Quelldokuments enthalten in Klammern einen Verweis auf eine Fußnote, zum Beispiel "(1)". Die **RepeatingGroup** enthält eine **Group**, die die Fußnote parst und ihren Inhalt in die Ausgabe einfügt.

Die **Group** enthält einen **Content**-Anker, der den Verweis auf die Fußnote abrufen und in einer Variablen speichern. Anschließend führt die **Group** die Aktion **RunParser** aus, die einen sekundären Parser aktiviert. Der sekundäre Parser findet die durch die Variable referenzierte Fußnote, parst sie und fügt das Ergebnis in die Ausgabe ein.

StructureDefinition

Der Anker **StructureDefinition** verarbeitet gut strukturierte Eingaben, beispielsweise Textnachrichten, die Nachrichtenprotokollen gemäß Branchenstandards entsprechen. Die Ausgabe der **StructureDefinition** erfolgt als XML-Darstellung der Daten.

Die Eingabedaten beinhalten durch Trennzeichen getrennten Datensätze. Sie organisieren die Eingabedatensätze in vordefinierter Weise. Beispiel: Einem Datensatz des Typs A geht ein Datensatz des Typs B voraus und er wird gefolgt von einem bis drei Datensätzen des Typs C. Jeder Datensatz enthält einen organisierten Satz an Feldern.

In der folgenden Tabelle werden die Eigenschaften des Ankers **StructureDefinition** beschrieben:

Eigenschaft	Beschreibung
disabled	Legt fest, ob die Komponente und alle zugehörigen untergeordneten Komponenten vom Skript ignoriert werden. Verwenden Sie diese Eigenschaft zum Testen, Debuggen und Ändern eines Skripts. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none">- Ausgewählt. Das Skript ignoriert die Komponente.- Gelöscht. Das Skript wendet die Komponente an. Standardmäßig ist die Eigenschaft deaktiviert.
format_definition	Definiert eine Liste mit Ankern und Aktionen, die die Datensätze ermitteln und extrahieren. Die Liste muss einen Anker ExtractRecord enthalten.
name	Eine beschreibende Kennzeichnung für die Komponente. Diese Kennzeichnung wird in der Protokolldatei sowie der Ereignisansicht angezeigt. Ermitteln Sie anhand der Eigenschaft name , welche Komponente das Ereignis verursacht hat.
Benachrichtigungen	Eine Liste mit NotificationHandler -Komponenten, die Benachrichtigungen aus geschachtelten Komponenten verarbeiten. Weitere Informationen hierzu finden Sie unter "Benachrichtigungen" auf Seite 430 .

Eigenschaft	Beschreibung
phase	Legt den Zeitpunkt fest, an dem das Skript die Komponente verarbeitet. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none"> - Anfangsphase. Das Skript verarbeitet die Komponente während der Anfangsphase. - Hauptphase. Das Skript verarbeitet die Komponente während der Hauptphase. - Endphase. Das Skript verarbeitet die Komponente während der Endphase. Weitere Informationen hierzu finden Sie unter "So sucht der Parser Anker" auf Seite 211 . Standardwert ist "Hauptphase".
remark	Ein benutzerdefinierter Kommentar, der den Zweck oder die Aktion der Komponente beschreibt.
target	Definiert einen Datenbehälter, in dem die Komponente ihre Ausgabe speichert.

Eine **StructureDefinition** weist die folgenden Teile auf:

- Eigenschaft **format_definition**. Extrahiert die Datensätze und ermittelt den jeweiligen Typ.
- Eine Hierarchie der untergeordneten Komponenten. Jede untergeordnete Komponente führt ein Parsing der Datensätze eines bestimmten Typs aus.

Die **format_definition** kann eine **RepeatingGroup** enthalten, die die Datensätze sucht. Innerhalb der **RepeatingGroup** ruft mindestens ein **Content**-Anker die IDs der Datensatztypen ab. Die **RepeatingGroup** enthält einen **ExtractRecord**-Anker, der den Datensatz an die Liste der Unterelemente übergibt.

Hinweis: Wenn eine Bibliotheksumwandlung eine **StructureDefinition** mit einer **format_definition** aufweist, die eine **RepeatingGroup** enthält, überschreibt die Umwandlung das sich wiederholende Element, anstatt es zu iterieren.

Die Hierarchie der Unterelemente zeigt die erforderliche Organisation der Datensätze an. Sie können Sequenzen, Auswahlen, Schleifen von Datensätzen und erforderliche oder optionale Datensätze konfigurieren.

Die Unterelemente empfangen Datensätze von **ExtractRecord**. Das System gleicht jeden Datensatz mit einem Unterelement nach den folgenden Kriterien ab:

- Die IDs `$id` und `$qualifier` des Datensatzes müssen mit den Werten übereinstimmen, die im Unterelement festgelegt sind.
- Die Position des Datensatzes in der Ausgabe muss mit der Position des Unterelements in der Hierarchie übereinstimmen.

Das übereinstimmende Unterelement führt für den Datensatz ein Parsing aus.

Wenn kein übereinstimmendes Unterelement vorhanden ist, löst die **StructureDefinition** eine Benachrichtigung aus. Sie können **NotificationHandler**-Komponenten einfügen, die die Benachrichtigung verarbeiten. Die Umwandlung verwendet den **StructureDefinition**-Anker, um Eingabefehler zu suchen und zu diagnostizieren. Normalerweise kann der **StructureDefinition**-Anker mit dem Parsen des Rests der Eingabe fortfahren.

Beispiel

Ein Parser verarbeitet Eingaben mit der folgenden Struktur:

```
ST*12*23~
NM1*12*23*4~
N1*12*23~
NM1*13*23*4~
N2*1*2*3~
SE*12*2:3:4~
```

Die Datensätze sind durch Zeilenvorschub-Zeichen begrenzt. In jedem Datensatz sind die Felder durch die Zeichen ~, * und : begrenzt.

Das erste Feld eines Datensatzes identifiziert den Datensatztyp. Es gibt fünf Haupt-Datensatztypen:

```
ST
NM1
N1
N2
SE
```

In den NM1-Datensätzen ist das zweite Feld ein Untertyp. Es gibt zwei Untertypen:

```
NM1*12
NM1*13
```

Die Datensätze müssen in der folgenden Reihenfolge vorkommen.

1. Ein ST-Datensatz.
2. Ein NM1*12-Datensatz, gefolgt durch N1.
3. Ein NM1*13-Datensatz, gefolgt durch N2.
4. Ein SE-Datensatz.

Sie können diese Eingabe durch Konfigurieren eines `StructureDefinition`-Ankers parsen.

Die Eigenschaft `format_definition` enthält eine `RepeatingGroup`, die die Datensätze findet. Die `RepeatingGroup` führt folgende Operationen aus:

1. Sie sucht den Datensatz-Content, bis zum Zeilenvorschub-Delimiter.
2. Sie extrahiert den Datensatztyp-Bezeichner, etwa `ST` oder `NM1` und speichert ihn in der `$id`-Variablen.
3. Ist der Datensatztyp `NM1`, so wird der Untertyp (12 oder 13) extrahiert und in der `$qualifier`-Variable gespeichert.
4. Sie führt einen `ExtractRecord`-Anker aus, der den Datensatz an die Unterelemente weitergibt.
`ExtractRecord` hängt die Bezeichner `$id` und `$qualifier` an den Datensatz.

Das Element wird so konfiguriert, dass es mit jedem Datensatz mit dem Bezeichner `ST` übereinstimmt.

Die `format_definition` findet den ersten Eingabe-Datensatz und übergibt ihn an die Unterelemente. Der erste Datensatz stimmt mit dem ersten Unterelement überein, da es den `ST`-Bezeichner hat. Das erste Unterelement enthält `Content Marker Content`-Anker, die den Datensatz parsen.

Das zweite Unterelement definiert eine Sequenz geschachtelter Unterelemente. Das erste geschachtelte Unterelement stimmt mit einem Datensatz mit den Bezeichnern `NM1` und `12` überein. Das zweite geschachtelte Unterelement stimmt mit einem Datensatz mit einem `N1`-Bezeichner überein.

Der zweite und dritte Eingabe-Datensatz sind `NM1*12` und `N1`. Diese stimmen mit der Sequenz der Unterelemente überein. Jedes geschachtelte Unterelement parst den entsprechenden Datensatz.

Nehmen wir an, der zweite und dritte Datensatz sind `NM1*12` und `N2`. Diese stimmen nicht mit der Hierarchie der Unterelemente überein, werden also nicht geparst.

Die nächsten Datensätze sind `NM1*13` und `N2`. Diese stimmen mit dem dritten Unterelement namens `Loop2000` überein.

Der letzte Datensatz ist `SE` und stimmt mit dem letzten Unterelement überein.

Alle Eingabe-Datensätze stimmen mit der Hierarchie der Unterelemente überein, sodass die `StructureDefinition` die gesamte Eingabe mit Erfolg parst.

Komponenten des Unterelements

Sie können in die Hierarchie der Unterelemente die folgenden Komponenten einfügen:

Unterelement	Beschreibung
RecordStructureLocal	Gleicht einen einzelnen Datensatz ab und führt ein Parsing durch.
SequenceStructureLocal	Definiert eine Folge von geschachtelten Unterelementen. Die Datensätze müssen in derselben Reihenfolge wie die geschachtelten Unterelemente vorkommen.
ChoiceStructureLocal	Definiert eine Auswahl an geschachtelten Unterelementen. Ein Datensatz muss mit einem der geschachtelten Unterelemente übereinstimmen.
AllStructureLocal	Definiert einen Satz an geschachtelten Unterelementen ohne eine festgelegte Folge. Die Datensätze können mit den geschachtelten Unterelementen in jeder beliebigen Reihenfolge übereinstimmen.

Die Namen enden auf `Local`, da sie in geschachtelten Positionen definiert werden können, die sich nicht auf der globalen Ebene des Skripts befinden. Es sind entsprechende Sätze an Komponenten mit der Bezeichnung `RecordStructure`, `SequenceStructure` usw. ohne das Suffix `Local` vorhanden. Diese können Sie auf der globalen Ebene des Skripts konfigurieren und an jeder beliebigen Stelle referenzieren, an der dies erforderlich ist. Fügen Sie ein `EmbeddedStructure`-Unterelement ein, um die globalen Komponenten zu referenzieren.

Die Liste der Unterelemente der obersten Ebene einer `StructureDefinition` entspricht `SequenceStructureLocal`. Die Datensätze müssen in derselben Reihenfolge wie die Unterelemente der obersten Ebene vorkommen.

Standardmäßig muss jedes Unterelement genau ein Mal vorkommen. Um diese Standardeinstellung zu ändern, legen Sie die Eigenschaften `minOccurs` und `maxOccurs` des Unterelements fest. Wenn beispielsweise ein Unterelement fehlen oder bis zu drei Mal vorkommen kann, legen Sie `minOccurs = 0` und `maxOccurs = 3` fest. Wenn ein Unterelement unbegrenzt vorkommen darf, legen Sie `maxOccurs = -1` fest.

Weitere Informationen zu Unterelementkomponenten finden Sie unter ["Die Unterkomponente Anker: Referenz" auf Seite 253](#).

Benachrichtigungen

Wenn ein Datensatz oder eine Gruppe von Datensätzen nicht der Hierarchie der Unterelemente entspricht, löst **StructureDefinition** eine Benachrichtigung aus.

In der folgenden Tabelle werden die Benachrichtigungstypen beschrieben:

Benachrichtigung	Beschreibung
MandatoryStructureMissing	Ein verpflichtender Datensatz erscheint nicht in der Eingabe.
MismatchIDs	Die ID des Datensatzes und des Unterelements stimmen teilweise überein. Beispiel: Es gibt zwei Datensatz-Bezeichner, und nur einer davon entspricht.
StructureBelowMinOccurs	Es gibt weniger passende übereinstimmende Datensätze des Unterelements als in minOccurs definiert sind.
StructureExceedsMaxOccurs	Es gibt mehr passende übereinstimmende Datensätze des Unterelements als in maxOccurs definiert sind.

Benachrichtigung	Beschreibung
StructureOutOfSequence	Die Datensätze stimmen mit den Unterelementen überein, aber nicht in der erforderlichen Sequenz. Beispiel: Die Unterelemente definieren eine Sequenz ABC, die Eingabe enthält jedoch ACB.
UnexpectedRecord	Die Datensätze stimmen mit den Unterelementen überein, aber nicht in der erforderlichen Hierarchie. Beispiel: Das Unterelement definierte eine Sequenz ABC, und D ist an einem anderen Ort definiert. Die Eingabe enthält ABD.
UnrecognizedRecord	Kein Unterelement stimmt mit den Datensatzbezeichnern überein.
XsdValidationError	Die Eingabe stimmt nicht mit den Anforderungen des Schemas überein.

Konfigurieren Sie die **NotificationHandler**-Komponenten in der **Benachrichtigungen**-Eigenschaft der **StructureDefinition** oder einem Unterelement. Sie können auch Handler in der **Benachrichtigungen**-Eigenschaft einer höherstufigen Komponente wie eines **Parsers** oder einer **Gruppe** konfigurieren, die die **StructureDefinition** enthält. Wenn ein Handler im Unterelement existiert, in dem eine Nichtübereinstimmung auftritt, wird die Benachrichtigung verarbeitet. Existiert kein Handler, zeigt die Benachrichtigung die IntelliScript-Hierarchie in einer Blase an, bis der Handler sie verarbeitet. Gibt es keinen Handler für eine Benachrichtigung, so wird die Benachrichtigung ignoriert und fährt die **StructureDefinition** mit der Verarbeitung der Eingabe fort.

Fortschrittsverfolgung

Da mit `format_definition` Datensätze extrahiert werden, aktualisiert die Eigenschaft die Systemvariable `VarStructureDetails`. Sie können die Variable in Benachrichtigungen verwenden. Beispiel: Um den Bezeichner des Datensatzes auszugeben, kann ein Benachrichtigungshandler `VarStructureDetails/RecordId` in seine Ausgabe einfügen.

Weitere Informationen hierzu finden Sie unter ["Systemvariablen" auf Seite 198](#).

Suchkomponenten: Referenz

Suchkomponenten werden für die folgenden Zwecke verwendet:

- Zur Definition der Position von Anker. Weitere Informationen hierzu finden Sie unter ["Ankerkomponenten: Referenz" auf Seite 217](#).
- Zur Definition von Delimitern oder Strings. Weitere Informationen hierzu finden Sie unter ["Formatkomponenten: Referenz" auf Seite 179](#).
- Zur Definition des Strings `find_what` eines **Replace**-Transformers. Weitere Informationen hierzu finden Sie unter ["Transformer-Komponenten: Referenz" auf Seite 264](#).

AttributeSearch

Die Suchkomponente **AttributeSearch** sucht ein Quelldokument, um den Wert eines bestimmten Attributs zu finden. Die Komponente ruft den Wert von einem Ausdruck in einem der folgenden Formate ab:

- `AttributeName = value`
- `AttributeName = "value"`

Dabei gilt: `AttributeName` ist der Name des Attributs. Die Anführungszeichen können einfach oder doppelt sein und die Leerzeichen sind optional.

AttributeSearch ist eine der Einstellungen für die Eigenschaft **value** des **Content**-Ankers. Weitere Informationen hierzu finden Sie unter ["Content" auf Seite 220](#).

Die nachstehende Tabelle beschreibt die Eigenschaften der Suchkomponente **AttributeSearch**:

Eigenschaft	Beschreibung
att	Definiert den Namen des Attributs.
match_case	Legt fest, ob der Name des Attributs die Groß-/Kleinschreibung berücksichtigt. Die Eigenschaft match_case hat folgende Optionen: <ul style="list-style-type: none">- Ausgewählt. Der Name des Attributs berücksichtigt die Groß-/Kleinschreibung.- Gelöscht. Der Name des Attributs berücksichtigt die Groß-/Kleinschreibung nicht.

Beispiel

Ein HTML-Dokument enthält das folgende Element:

```
<img src='MyPicture.gif'>
```

Mit der Komponente `AttributeSearch` können Sie den Wert des Attributs `src` abrufen. Zurückgegeben wird der Text `MyPicture.gif`.

Gültige Attributsyntax

AttributeSearch liest Name-Wert-Paare, die ein Gleichheitszeichen aufweisen. Optional kann das Gleichheitszeichen in Leerzeichen eingeschlossen werden. Der Wert kann von doppelten oder einfachen Anführungszeichen umschlossen oder ohne Anführungszeichen angegeben werden.

Nehmen wir beispielsweise an, dass die Komponente **AttributeSearch** dahingehend konfiguriert ist, ein Attribut mit der Bezeichnung `time` zu suchen. Alle folgenden Beispiele weisen eine gültige Syntax auf und geben denselben Wert (`12:55:33`) zurück.

```
time = "12:55:33"  
time="12:55:33"  
time = '12:55:33'  
time='12:55:33'  
time = 12:55:33  
time=12:55:33
```

Online-Beispiel

Ein Online-Beispiel zu dieser Komponente befindet sich im Projekt `samples\Projects\Content\Content.cmw`. Das Beispiel veranschaulicht, wie mit Hilfe der `AttributeSearch`-Komponente ein Textdokument mit der Struktur `variable = value` geparkt werden kann.

LearnByExample

Die Suchkomponente **LearnByExample** lernt, wie ein Text durchsucht werden soll, indem sie die Textposition im Beispiel-Quelldokument analysiert. Sie parst das Quelldokument mit dem Parserformat.

Wenn der Parser zum Beispiel ein tabulatorbegrenztes Format hat, zählt die **LearnByExample**-Komponente die Anzahl der Tabulatoren ab dem Anfang des Suchbereichs bis zum Beispieltext. Anschließend sucht die Komponente denjenigen Text im Quelldokument, der dieselbe Anzahl von Tabulatoren vom Anfang des Suchbereichs entfernt ist.

LearnByExample ist eine der Einstellungen für die Eigenschaft **value** des **Content**-Ankers. Weitere Informationen hierzu finden Sie unter ["Content" auf Seite 220](#).

Wenn das Attribut **direction** des **Content**-Ankers auf `backward` gesetzt ist, zählt die Komponente die Delimiter ab dem Ende des Suchbereichs.

Die nachstehende Tabelle beschreibt die Eigenschaften der Suchkomponente **LearnByExample**:

Eigenschaft	Beschreibung
example	Definiert den Text im Beispielquellldokument an der Ankerposition.

Hinweis: Die Suchkomponente **LearnByExample** wendet sensible Heuristik an. Damit die Suchkomponente **LearnByExample** verwendet werden kann, muss das Beispiel dieselbe Form wie die erwartete Ausgabe aufweisen. Die Form muss für die gesamte Eingabedatei bzw. alle Eingabedateien einheitlich sein. Stimmt das Beispiel nicht mit der Eingabe überein, schlägt die Suchkomponente **LearnByExample** unter Umständen fehl.

NewlineSearch

Die Suchkomponente **NewlineSearch** sucht nach einem Zeichen für Newline (neue Zeile) oder Linefeed (Zeilenvorschub) (0x0A), ein Carriage Return-Zeichen (Wagenrücklauf) (0x0D) oder beide.

Der **Marker**-Anker kann **NewlineSearch** verwenden, um Newline-Marker zu finden. **Delimiter**-Komponenten können mit **NewlineSearch** Zeilenvorschub-Delimiter finden.

OffsetSearch

Die Suchkomponente **OffsetSearch** definiert die Anzahl der Zeichen zwischen einem Referenzpunkt und einem Anker. Beispielsweise kann sie die Anzahl der Zeichen zwischen dem Ende eines **Marker**-Ankers und dem Anfang eines **Content**-Ankers bestimmen.

Die nachstehende Tabelle beschreibt die Eigenschaften der Suchkomponente **OffsetSearch**:

Eigenschaft	Beschreibung
allow_smaller_offset	Legt fest, ob ein Offset, der über den Suchbereich hinausreicht, gültig ist. Wählen Sie diese Eigenschaft, damit am Ende eines Dokuments eine gekürzte Feldgröße zulässig ist. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none">- Ausgewählt. OffsetSearch ist erfolgreich, wenn ein Offset über den Suchbereich hinausreicht.- Gelöscht. OffsetSearch schlägt fehl, wenn ein Offset über den Suchbereich hinausreicht.
Offset	Definiert die Anzahl der Zeichen zwischen dem Referenzpunkt und dem Anker. An einigen Stellen, an denen OffsetSearch verwendet wird, zum Beispiel in einem Marker -Anker, zeigt der IntelliScript-Editor neben der Eigenschaft offset die Schaltfläche „Durchsuchen“ an. Sie können einen Wert eingeben oder einen Datenbehälter auswählen, der den gesuchten Wert enthält.

PatternSearch

Die Suchkomponente **PatternSearch** sucht nach einem String, der mit einem regulären Ausdruck übereinstimmt.

Anker können mit Hilfe von **PatternSearch** Marker oder Content finden. Die **Delimiter**-Komponente kann **PatternSearch** nutzen, um Delimiter finden. Der **Replace**-Transformer findet mit **PatternSearch** den zu ersetzenden Text.

Die nachstehende Tabelle beschreibt die Eigenschaften der Suchkomponente **PatternSearch**:

Eigenschaft	Beschreibung
escape_sequence	Definiert ein Präfix, das bewirkt, dass die Suchkomponente eine Instanz des Musters im Quelldokument ignoriert.
pattern	Definiert den regulären Ausdruck.

Weitere Informationen über die Syntax von regulären Ausdrücken finden Sie unter [“Regulärer Ausdruck - Syntax” auf Seite 287](#).

Beispiel

Angenommen, Sie möchten den String `%%`, der ein oder mehrere `%`-Zeichen enthält, als Delimiter definieren. In der **Delimiter**-Komponente können Sie eine **PatternSearch** mit folgendem regulären Ausdruck festlegen:

```
%+
```

Ein weiteres Beispiel: Sie möchten ein Komma und ein Semikolon als alternative Delimiter auf derselben Ebene der Delimiter-Hierarchie definieren. Dafür können Sie folgenden regulären Ausdruck verwenden:

```
[, ;]
```

SegmentSearch

Die Suchkomponente **SegmentSearch** sucht öffnende und schließende Marker in einem Textstring. Sie gibt das Segment vom öffnenden bis zum schließenden Marker zurück, einschließlich der Marker selbst. Die Komponente **SegmentSearch** stellt eine der Optionen für das Attribut **find_what** des Transformers **Replace** dar.

In der folgenden Tabelle werden die Eigenschaften der Suchkomponente **SegmentSearch** beschrieben:

Eigenschaft	Beschreibung
opening	Definiert das Suchkriterium für den öffnenden Marker. Die Optionen bestehen aus den folgenden Suchkomponenten: <ul style="list-style-type: none">- NewlineSearch- OffsetSearch- PatternSearch- TextSearch
closing	Definiert das Suchkriterium für den schließenden Marker. Die Optionen bestehen aus den folgenden Suchkomponenten: <ul style="list-style-type: none">- NewlineSearch- OffsetSearch- PatternSearch- TextSearch

TextSearch

Die Suchkomponente **TextSearch** sucht nach einem expliziten String.

Anker können mit Hilfe von **TextSearch** Marker finden. Die **Delimiter**-Komponente kann mit **TextSearch** Delimiter finden. Der **Replace**-Transformer findet mit **TextSearch** den zu ersetzenden Text.

In der folgenden Tabelle werden die Eigenschaften der Suchkomponente **TextSearch** beschrieben:

Eigenschaften	Beschreibung
escape_sequence	Definiert ein Präfix, das bei der Suche anweist, eine Instanz des Strings im Quelldokument zu ignorieren. An Stellen, an denen die dynamische Suche unterstützt wird, können Sie einen Datenbehälter angeben, der die Escape-Sequenz enthält.
match_case	Legt fest, ob der definierte Text unter Berücksichtigung der Groß- und Kleinschreibung exakt übereinstimmen muss. Die Standardoption lautet „Gelöscht“.
text	Definiert den zu suchenden String. An Stellen, an denen die dynamische Suche unterstützt wird, können Sie einen Datenbehälter angeben, der den String enthält.

Beispiel

Um den String Prozentzeichen-Prozentzeichen-Tabulatorzeichen als Delimiter zu definieren, erstellen Sie eine **Delimiter**-Komponente und setzen deren Eigenschaft `search` auf **TextSearch**. Geben Sie in der Eigenschaft `text` folgenden String ein:

%%

Drücken Sie dann **STRG+A** und geben Sie den ASCII-Code für das Tabulatorzeichen ein: 009

Dynamische Definition eines Suchstrings

An einigen Stellen, an denen **TextSearch** verwendet wird (zum Beispiel in einer **Delimiter**-Komponente oder einem **Marker**-Anker), wird neben dem Textfeld die Schaltfläche „Durchsuchen“ angezeigt. Suchen Sie nach einem Datenbehälter, der den Suchtext enthält.

Wenn Sie nach wiederholten Instanzen des ersten Wortes in einem Dokument suchen möchten, können Sie einen **Content**-Anker definieren, der das erste Wort abrufen und in einer Variablen speichern. Anschließend definieren Sie **Marker**-Anker, die mithilfe von **TextSearch** weitere Instanzen des in dieser Variablen gespeicherten Wortes finden.

Online-Beispiel

Ein Online-Beispiel zu dieser Komponente befindet sich im Projekt `samples\Projects\Dynamic_And_RepeatingGroup\Dynamic_And_RepeatingGroup.cmw`.

In diesem Beispiel wird gezeigt, wie in der Komponente `GetRemarkParser` ein **Marker**-Anker mit Hilfe einer dynamisch definierten **TextSearch**-Komponente eine Fußnote am Ende des Quelldokuments findet. Weitere Informationen zu diesem Beispiel finden Sie im Abschnitt [“RepeatingGroup” auf Seite 239](#).

TypeSearch

Die Suchkomponente **TypeSearch** sucht nach einem Anker eines angegebenen Datentyps.

Die Komponente **TypeSearch** stellt eine der Einstellungen für die Eigenschaft **value** des **Content**-Ankers dar. Weitere Informationen hierzu finden Sie unter [“Content” auf Seite 220](#).

In der folgenden Tabelle werden die Eigenschaften der Suchkomponente **TypeSearch** beschrieben:

Eigenschaft	Beschreibung
val_type	Legt den zu suchenden Datentyp des Ankers fest.

Die Unterkomponente Anker: Referenz

Die Unterkomponenten "Anker" werden als Werte bestimmter Ankereigenschaften zugewiesen.

AllStructure

Die Komponente **AllStructure** definiert eine Gruppe geschachtelter Unterelemente ohne Berücksichtigung der Sequenz. Eine Gruppe von Datensätzen stimmt mit **AllStructure** überein, wenn sie mit allen Unterelementen in einer Sequenz übereinstimmt. Weitere Informationen hierzu finden Sie unter ["StructureDefinition" auf Seite 244](#).

Die Komponente **AllStructure** wird auf der globalen Ebene des Skripts angezeigt und hat dieselbe Funktion wie **AllStructureLocal**. Sie können darauf im Attribut **ref** einer **EmbeddedStructure** verweisen.

Die nachstehende Tabelle beschreibt die Eigenschaften der Komponente **AllStructure**:

Eigenschaft	Beschreibung
action	Definiert eine Aktion, die mit der Liste der Teilkomponenten durchgeführt wird.
disabled	Legt fest, ob die Komponente und alle zugehörigen untergeordneten Komponenten vom Skript ignoriert werden. Verwenden Sie diese Eigenschaft zum Testen, Debuggen und Ändern eines Skripts. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none">- Ausgewählt. Das Skript ignoriert die Komponente.- Gelöscht. Das Skript wendet die Komponente an. Standardmäßig ist die Eigenschaft deaktiviert.
name	Eine beschreibende Kennzeichnung für die Komponente. Diese Kennzeichnung wird in der Protokolldatei sowie der Ereignisansicht angezeigt. Ermitteln Sie anhand der Eigenschaft name , welche Komponente das Ereignis verursacht hat.
notifications	Eine Liste mit NotificationHandler -Komponenten, die Benachrichtigungen aus geschachtelten Komponenten verarbeiten. Weitere Informationen hierzu finden Sie unter "Benachrichtigungen" auf Seite 430 .
remark	Ein benutzerdefinierter Kommentar, der den Zweck oder die Aktion der Komponente beschreibt.
target	Definiert einen Datenbehälter, in dem die Komponente ihre Ausgabe speichert.

AllStructureLocal

Die Komponente **AllStructureLocal** definiert eine Gruppe von geschachtelten Unterelementen ohne Berücksichtigung der Sequenz. Eine Gruppe von Datensätzen stimmt mit **AllStructureLocal** überein, wenn sie mit allen Unterelementen in einer Sequenz übereinstimmt. Weitere Informationen hierzu finden Sie unter ["StructureDefinition" auf Seite 244](#).

AllStructureLocal ist ein Unterelement des Ankers **StructureDefinition** und hat dieselbe Funktion wie **AllStructure**. Verwenden Sie auf der globalen Ebene des Skripts **AllStructure**.

Die nachstehende Tabelle beschreibt die Eigenschaften der Komponente **AllStructure**:

Eigenschaft	Beschreibung
action	Definiert eine Aktion, die mit der Liste der Teilkomponenten durchgeführt wird.
disabled	Legt fest, ob die Komponente und alle zugehörigen untergeordneten Komponenten vom Skript ignoriert werden. Verwenden Sie diese Eigenschaft zum Testen, Debuggen und Ändern eines Skripts. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none"> - Ausgewählt. Das Skript ignoriert die Komponente. - Gelöscht. Das Skript wendet die Komponente an. Standardmäßig ist die Eigenschaft deaktiviert.
minOccurs	Definiert die Mindestanzahl der übereinstimmenden Datensätze. Standardwert ist 1.
maxOccurs	Definiert die Höchstanzahl der übereinstimmenden Datensätze. Standardwert ist 1. Verwenden Sie -1 für eine unbegrenzte Anzahl.
name	Eine beschreibende Kennzeichnung für die Komponente. Diese Kennzeichnung wird in der Protokolldatei sowie der Ereignisansicht angezeigt. Ermitteln Sie anhand der Eigenschaft name , welche Komponente das Ereignis verursacht hat.
notifications	Eine Liste mit NotificationHandler -Komponenten, die Benachrichtigungen aus geschachtelten Komponenten verarbeiten. Weitere Informationen hierzu finden Sie unter "Benachrichtigungen" auf Seite 430 .
remark	Ein benutzerdefinierter Kommentar, der den Zweck oder die Aktion der Komponente beschreibt.
sub_elements	Definiert eine Liste geschachtelter Unterelemente.
target	Definiert einen Datenbehälter, in dem die Komponente ihre Ausgabe speichert.

ChoiceStructure

Die Komponente **ChoiceStructure** definiert eine Gruppe von geschachtelten Unterelementen. Ein Datensatz stimmt mit **ChoiceStructure** überein, wenn er mit einem geschachtelten Unterelement übereinstimmt. Weitere Informationen hierzu finden Sie unter ["StructureDefinition" auf Seite 244](#).

Die Komponente **ChoiceStructure** wird auf der globalen Ebene des Skripts angezeigt und hat dieselbe Funktion wie **ChoiceStructureLocal**. Sie können darauf im Attribut **ref** einer **EmbeddedStructure** verweisen.

Die nachstehende Tabelle beschreibt die Eigenschaften der Komponente **ChoiceStructure**:

Eigenschaft	Beschreibung
action	Definiert eine Aktion, die mit der Liste der Teilkomponenten durchgeführt wird.
disabled	Legt fest, ob die Komponente und alle zugehörigen untergeordneten Komponenten vom Skript ignoriert werden. Verwenden Sie diese Eigenschaft zum Testen, Debuggen und Ändern eines Skripts. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none"> - Ausgewählt. Das Skript ignoriert die Komponente. - Gelöscht. Das Skript wendet die Komponente an. Standardmäßig ist die Eigenschaft deaktiviert.

Eigenschaft	Beschreibung
name	Eine beschreibende Kennzeichnung für die Komponente. Diese Kennzeichnung wird in der Protokolldatei sowie der Ereignisansicht angezeigt. Ermitteln Sie anhand der Eigenschaft name , welche Komponente das Ereignis verursacht hat.
notifications	Eine Liste mit NotificationHandler -Komponenten, die Benachrichtigungen aus geschachtelten Komponenten verarbeiten. Weitere Informationen hierzu finden Sie unter "Benachrichtigungen" auf Seite 430 .
remark	Ein benutzerdefinierter Kommentar, der den Zweck oder die Aktion der Komponente beschreibt.
target	Definiert einen Datenbehälter, in dem die Komponente ihre Ausgabe speichert.

ChoiceStructureLocal

Die Komponente **ChoiceStructureLocal** definiert eine Gruppe von geschachtelten Unterelementen. Ein Datensatz stimmt mit **ChoiceStructureLocal** überein, wenn er mit einem geschachtelten Unterelement übereinstimmt. Weitere Informationen hierzu finden Sie unter ["StructureDefinition" auf Seite 244](#).

ChoiceStructureLocal ist ein Unterelement des Ankers **StructureDefinition** und hat dieselbe Funktion wie **ChoiceStructure**. Verwenden Sie auf globaler Skriptebene **ChoiceStructure**.

Die nachstehende Tabelle beschreibt die Eigenschaften der Komponente **ChoiceStructureLocal**:

Eigenschaft	Beschreibung
action	Definiert eine Aktion, die mit der Liste der Teilkomponenten durchgeführt wird.
disabled	Legt fest, ob die Komponente und alle zugehörigen untergeordneten Komponenten vom Skript ignoriert werden. Verwenden Sie diese Eigenschaft zum Testen, Debuggen und Ändern eines Skripts. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none"> - Ausgewählt. Das Skript ignoriert die Komponente. - Gelöscht. Das Skript wendet die Komponente an. Standardmäßig ist die Eigenschaft deaktiviert.
minOccurs	Definiert die Mindestanzahl der übereinstimmenden Datensätze. Standardwert ist 1.
maxOccurs	Definiert die Höchstanzahl der übereinstimmenden Datensätze. Standardwert ist 1. Verwenden Sie -1 für eine unbegrenzte Anzahl.
name	Eine beschreibende Kennzeichnung für die Komponente. Diese Kennzeichnung wird in der Protokolldatei sowie der Ereignisansicht angezeigt. Ermitteln Sie anhand der Eigenschaft name , welche Komponente das Ereignis verursacht hat.
notifications	Eine Liste mit NotificationHandler -Komponenten, die Benachrichtigungen aus geschachtelten Komponenten verarbeiten. Weitere Informationen hierzu finden Sie unter "Benachrichtigungen" auf Seite 430 .
remark	Ein benutzerdefinierter Kommentar, der den Zweck oder die Aktion der Komponente beschreibt.
sub_elements	Definiert eine Liste geschachtelter Unterelemente.
target	Definiert einen Datenbehälter, in dem die Komponente ihre Ausgabe speichert.

Connect

Die **Connect**-Komponente bestimmt einen Link zwischen Datenbehältern in zwei Komponenten. Die beiden Datenbehälter müssen denselben Datentyp haben.

Die nachstehende Tabelle beschreibt die Eigenschaften der Komponente **Connect**:

Eigenschaft	Beschreibung
data_holder	Definiert einen Datenbehälter, der im ersten Parser, Serializer oder Mapper referenziert wird. Hinweis: Ist der Datenbehälter eine Variable, so weist die Umwandlung ihm einen leeren Standardwert zu. Wenn die Variable einen Datentyp hat, der keinen leeren Wert akzeptiert, etwa <code>:Boolean</code> , müssen Sie sicherstellen, dass die Variable einen Wert hat, bevor die eingebettete Umwandlung ausgeführt wird.
embedded_data_holder	Definiert einen Datenbehälter, der im zweiten Parser, Serializer oder Mapper referenziert wird.
name	Eine beschreibende Kennzeichnung für die Komponente. Diese Kennzeichnung wird in der Protokolldatei sowie der Ereignisansicht angezeigt. Ermitteln Sie anhand der Eigenschaft name , welche Komponente das Ereignis verursacht hat.
remark	Ein benutzerdefinierter Kommentar, der den Zweck oder die Aktion der Komponente beschreibt.

Die **schema_connections**-Eigenschaft der folgenden Komponenten kann eine oder mehrere Instanzen der **Connect**-Komponente haben:

- EmbeddedParser. Gibt an, wo der zweite Parser seine Ergebnisse in der Ausgabe des Hauptparsers speichert.
- EmbeddedSerializer. Gibt einen Link zwischen den Eingabe-Datenbehältern eines zweiten Serializers und den Eingabe-Datenbehältern des Hauptserializers an.
- EmbeddedMapper. Gibt einen Link zwischen den Eingabe- und den Ausgabe-Datenbehältern an.
- EmbeddedStructure. Gibt einen Link zwischen den Zielen globaler und lokaler **StructureDefinition**-Unterelemente an.

Beispiel

Ein sekundärer Parser gibt das XML-Element ID aus. Der Hauptparser soll dieses Ergebnis in der Variable `VarID` speichern. **Dazu verbinden Sie ID mit VarID.**

Ein weiteres Beispiel hierzu finden Sie im Abschnitt [“EmbeddedSerializer” auf Seite 352](#).

EmbeddedStructure

Die Komponente **EmbeddedStructure** aktiviert Komponenten, die auf der globalen Ebene des Skripts definiert sind. Weitere Informationen hierzu finden Sie unter [“StructureDefinition” auf Seite 244](#).

EmbeddedStructure ist ein Unterelement des Ankers **StructureDefinition**.

Die nachstehende Tabelle beschreibt die Eigenschaften der Komponente **EmbeddedStructure**:

Eigenschaft	Beschreibung
disabled	Legt fest, ob die Komponente und alle zugehörigen untergeordneten Komponenten vom Skript ignoriert werden. Verwenden Sie diese Eigenschaft zum Testen, Debuggen und Ändern eines Skripts. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none"> - Ausgewählt. Das Skript ignoriert die Komponente. - Gelöscht. Das Skript wendet die Komponente an. Standardmäßig ist die Eigenschaft deaktiviert.
minOccurs	Definiert die Mindestanzahl der übereinstimmenden Datensätze.
maxOccurs	Definiert die Höchstanzahl der übereinstimmenden Datensätze. Verwenden Sie -1 für eine unbegrenzte Anzahl.
name	Eine beschreibende Kennzeichnung für die Komponente. Diese Kennzeichnung wird in der Protokolldatei sowie der Ereignisansicht angezeigt. Ermitteln Sie anhand der Eigenschaft name , welche Komponente das Ereignis verursacht hat.
notifications	Eine Liste mit NotificationHandler -Komponenten, die Benachrichtigungen aus geschachtelten Komponenten verarbeiten. Weitere Informationen hierzu finden Sie unter "Benachrichtigungen" auf Seite 430 .
ref	Definiert den Namen der global konfigurierten Komponente.
remark	Ein benutzerdefinierter Kommentar, der den Zweck oder die Aktion der Komponente beschreibt.
schema_connections	Verbindet das Ziel des Referenz-Unterelements mit dem Ziel der EmbeddedStructure . Weitere Informationen hierzu finden Sie unter "Connect" auf Seite 256 .
target	Definiert einen Datenbehälter, in dem die Komponente ihre Ausgabe speichert.

RecordStructure

Die Komponente **RecordStructure** definiert eine Gruppe von Komponenten. Eine Gruppe von Datensätzen stimmt mit **RecordStructure** überein, wenn sie dieselben Bezeichner hat. Weitere Informationen hierzu finden Sie unter ["StructureDefinition" auf Seite 244](#).

Die Komponente **RecordStructure** wird auf der globalen Ebene des Skripts angezeigt und hat dieselbe Funktion wie **RecordStructureLocal**. Sie können darauf im Attribut **ref** einer **EmbeddedStructure** verweisen.

Die nachstehende Tabelle beschreibt die Eigenschaften der Komponente **RecordStructure**:

Eigenschaft	Beschreibung
disabled	Legt fest, ob die Komponente und alle zugehörigen untergeordneten Komponenten vom Skript ignoriert werden. Verwenden Sie diese Eigenschaft zum Testen, Debuggen und Ändern eines Skripts. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none"> - Ausgewählt. Das Skript ignoriert die Komponente. - Gelöscht. Das Skript wendet die Komponente an. Standardmäßig ist die Eigenschaft deaktiviert.
ids	Definiert einen oder mehrere Strings

Eigenschaft	Beschreibung
name	Eine beschreibende Kennzeichnung für die Komponente. Diese Kennzeichnung wird in der Protokolldatei sowie der Ereignisansicht angezeigt. Ermitteln Sie anhand der Eigenschaft name , welche Komponente das Ereignis verursacht hat.
notifications	Eine Liste mit NotificationHandler -Komponenten, die Benachrichtigungen aus geschachtelten Komponenten verarbeiten. Weitere Informationen hierzu finden Sie unter "Benachrichtigungen" auf Seite 430 .
remark	Ein benutzerdefinierter Kommentar, der den Zweck oder die Aktion der Komponente beschreibt.
target	Definiert einen Datenbehälter, in dem die Komponente ihre Ausgabe speichert.

RecordStructureLocal

Die Komponente **RecordStructureLocal** definiert eine Gruppe von Komponenten. Eine Gruppe von Datensätzen stimmt mit **RecordStructureLocal** überein, wenn sie dieselben Bezeichner hat. Weitere Informationen hierzu finden Sie unter ["StructureDefinition" auf Seite 244](#).

RecordStructureLocal ist ein Unterelement des Ankers **StructureDefinition** und hat dieselbe Funktion wie **RecordStructure**. Verwenden Sie auf der globalen Ebene des Skripts **RecordStructure**.

Die nachstehende Tabelle beschreibt die Eigenschaften der Komponente **RecordStructureLocal**:

Eigenschaft	Beschreibung
disabled	Legt fest, ob die Komponente und alle zugehörigen untergeordneten Komponenten vom Skript ignoriert werden. Verwenden Sie diese Eigenschaft zum Testen, Debuggen und Ändern eines Skripts. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none"> - Ausgewählt. Das Skript ignoriert die Komponente. - Gelöscht. Das Skript wendet die Komponente an. Standardmäßig ist die Eigenschaft deaktiviert.
ids	Definiert einen oder mehrere Strings
minOccurs	Definiert die Mindestanzahl der übereinstimmenden Datensätze. Standardwert ist 1.
maxOccurs	Definiert die Höchstanzahl der übereinstimmenden Datensätze. Standardwert ist 1. Verwenden Sie -1 für eine unbegrenzte Anzahl.
name	Eine beschreibende Kennzeichnung für die Komponente. Diese Kennzeichnung wird in der Protokolldatei sowie der Ereignisansicht angezeigt. Ermitteln Sie anhand der Eigenschaft name , welche Komponente das Ereignis verursacht hat.
notifications	Eine Liste mit NotificationHandler -Komponenten, die Benachrichtigungen aus geschachtelten Komponenten verarbeiten. Weitere Informationen hierzu finden Sie unter "Benachrichtigungen" auf Seite 430 .
remark	Ein benutzerdefinierter Kommentar, der den Zweck oder die Aktion der Komponente beschreibt.
target	Definiert einen Datenbehälter, in dem die Komponente ihre Ausgabe speichert.

SequenceStructure

Die Komponente **SequenceStructure** definiert eine Folge von geschachtelten Unterelementen. Eine Gruppe von Datensätzen stimmt mit der Komponente **SequenceStructure** überein, wenn sie mit allen geschachtelten Unterelementen in Folge übereinstimmt. Weitere Informationen hierzu finden Sie unter ["StructureDefinition" auf Seite 244](#).

Die Komponente **SequenceStructure** wird auf der globalen Ebene des Skripts angezeigt und weist dieselbe Funktion wie die Komponente **SequenceStructureLocal** auf. Sie können darauf im Attribut **ref** einer **EmbeddedStructure** verweisen.

In der folgenden Tabelle werden die Eigenschaften der Komponente **SequenceStructure** beschrieben:

Eigenschaft	Beschreibung
action	Definiert eine Aktion, die mit der Liste der Teilkomponenten durchgeführt wird.
disabled	Legt fest, ob die Komponente und alle zugehörigen untergeordneten Komponenten vom Skript ignoriert werden. Verwenden Sie diese Eigenschaft zum Testen, Debuggen und Ändern eines Skripts. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none">- Ausgewählt. Das Skript ignoriert die Komponente.- Gelöscht. Das Skript wendet die Komponente an. Standardmäßig ist die Eigenschaft deaktiviert.
name	Eine beschreibende Kennzeichnung für die Komponente. Diese Kennzeichnung wird in der Protokolldatei sowie der Ereignisansicht angezeigt. Ermitteln Sie anhand der Eigenschaft name , welche Komponente das Ereignis verursacht hat.
notifications	Eine Liste mit NotificationHandler -Komponenten, die Benachrichtigungen aus geschachtelten Komponenten verarbeiten. Weitere Informationen hierzu finden Sie unter "Benachrichtigungen" auf Seite 430 .
remark	Ein benutzerdefinierter Kommentar, der den Zweck oder die Aktion der Komponente beschreibt.
target	Definiert einen Datenbehälter, in dem die Komponente ihre Ausgabe speichert.

SequenceStructureLocal

Die Komponente **SequenceStructureLocal** definiert eine Folge von geschachtelten Unterelementen. Eine Gruppe von Datensätzen stimmt mit der Komponente **SequenceStructureLocal** überein, wenn sie mit allen geschachtelten Unterelementen in Folge übereinstimmt. Weitere Informationen hierzu finden Sie unter ["StructureDefinition" auf Seite 244](#).

Die Komponente **SequenceStructureLocal** ist ein Unterelement des Ankers **StructureDefinition** und weist dieselbe Funktion wie die Komponente **SequenceStructure** auf. Verwenden Sie auf der globalen Ebene des Skripts die Komponente **SequenceStructure**.

In der folgenden Tabelle werden die Eigenschaften der Komponente **SequenceStructureLocal** beschrieben:

Eigenschaft	Beschreibung
action	Definiert eine Aktion, die mit der Liste der Teilkomponenten durchgeführt wird.
disabled	Legt fest, ob die Komponente und alle zugehörigen untergeordneten Komponenten vom Skript ignoriert werden. Verwenden Sie diese Eigenschaft zum Testen, Debuggen und Ändern eines Skripts. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none"> - Ausgewählt. Das Skript ignoriert die Komponente. - Gelöscht. Das Skript wendet die Komponente an. Standardmäßig ist die Eigenschaft deaktiviert.
minOccurs	Definiert die Mindestanzahl der übereinstimmenden Datensätze. Standardwert ist 1.
maxOccurs	Definiert die Höchstanzahl der übereinstimmenden Datensätze. Standardwert ist 1. Verwenden Sie -1 für eine unbegrenzte Anzahl.
name	Eine beschreibende Kennzeichnung für die Komponente. Diese Kennzeichnung wird in der Protokolldatei sowie der Ereignisansicht angezeigt. Ermitteln Sie anhand der Eigenschaft name , welche Komponente das Ereignis verursacht hat.
notifications	Eine Liste mit NotificationHandler -Komponenten, die Benachrichtigungen aus geschachtelten Komponenten verarbeiten. Weitere Informationen hierzu finden Sie unter "Benachrichtigungen" auf Seite 430 .
remark	Ein benutzerdefinierter Kommentar, der den Zweck oder die Aktion der Komponente beschreibt.
sub_elements	Definiert eine Liste geschachtelter Unterelemente.
target	Definiert einen Datenbehälter, in dem die Komponente ihre Ausgabe speichert.

KAPITEL 17

Transformer

Dieses Kapitel umfasst die folgenden Themen:

- [Übersicht über Transformer, 261](#)
- [Definieren von Transformern, 261](#)
- [Standardeigenschaften von Transformern, 263](#)
- [Transformer-Komponenten: Referenz, 264](#)

Übersicht über Transformer

Transformer ändern die Ausgabe anderer Komponenten.

Transformer können in anderen Komponenten wie Anker, Serialisierungsankern und Aktionen verwendet werden. Wenn Sie beispielsweise einen Transformer in einem `Content`-Anker verwenden, ändert er die Daten, die der Anker aus dem Quelldokument extrahiert.

Sie können Transformer als Dokumentprozessoren verwenden. Sie können außerdem einen Transformer auf der globalen Ebene eines Skripts definieren und als Startkomponente festlegen.

Definieren von Transformern

Transformer können an den folgenden Positionen im Skript definiert werden:

- in der Eigenschaft **transformers** eines Ankers oder Serialisierungsankers
- in der Eigenschaft **default_transformers** eines Formats oder Serializers
- im Dokumentprozessor **ProcessByTransformers**
- in der Eigenschaft **transformers** bestimmter Aktionen
- auf globaler Ebene als selbstständige ausführbare Komponente zum Bearbeiten eines Quelldokuments

Verwenden von Transformern in Anker

Sie können Transformer in einem Anker verwenden, der eine XML-Ausgabe erstellt, beispielsweise im `Content`-Anker. Schachteln Sie dazu die Transformer-Komponenten im Skript in die Eigenschaft **transformers** des Ankers.

Die Eingabe eines Transformers ist die Rohausgabe des Ankers, bevor der Anker die Ausgabe in einen Datenbehälter einfügt.

Nehmen wir zum Beispiel an, dass Sie das folgende Quelldokument parsen:

```
First name: Ron
Last name: Lehrer
```

Sie möchten eine XML-Ausgabe in Großbuchstaben (`ALL CAPS`) erstellen:

```
<Person>
  <FirstName>RON</FirstName>
  <LastName>LEHRER</LastName>
</Person>
```

Konfigurieren Sie dazu die `Content`-Anker, die die Strings `Ron` und `Lehrer` abrufen, mit dem Transformer `ChangeCase`.

Folgen von Transformern

Sie können einen Anker mit einer Folge von Transformern konfigurieren. Jeder Transformer bearbeitet die Ausgabe des vorangehenden Transformers.

Angenommen, Sie möchten im Beispiel `Ron Lehrer` folgende Ausgabe erzeugen:

```
<Person>
  <FirstName>- RON -</FirstName>
  <LastName>- LEHRER -</LastName>
</Person>
```

Zu diesem Zweck konfigurieren Sie die `Content`-Anker mit den Transformern `ChangeCase` und `AddString`. Von diesen Transformern wird zunächst die Klein- in Großschreibung umgewandelt. Anschließend werden Bindestriche hinzugefügt.

Standardtransformer

Häufig möchte man dieselben Transformer auf allen **Content**-Ankern in einem Parser ausführen. Sie können die Formatkomponente des Parsers mit Standardtransformatern konfigurieren. Sie brauchen dann diese Transformer nicht mehr für jeden Anker im Parser einzeln anzugeben.

Zu diesem Zweck schachteln Sie die Transformer in die Eigenschaft **default_transformers** des Formats. Weitere Informationen hierzu finden Sie unter ["Formatkomponenten: Referenz" auf Seite 179](#).

Viele vordefinierte Formatkomponenten enthalten Standardtransformer. Beispielsweise verfügt die Komponente **HTMLFormat** über Standardtransformer, die HTML-Tags aus der Ausgabe entfernen und HTML-Entitäten in reinen Text umwandeln. Sie können die Standardtransformer ändern, indem Sie die Eigenschaft **default_transformers** des Formats bearbeiten.

Wenn ein Anker eigene Transformer hat, werden diese nach den Standardtransformatern ausgeführt.

Sie können die Standardtransformer für einzelne Anker deaktivieren. Dazu muss die Eigenschaft **ignore_default_transformers** des betreffenden Ankers festgelegt werden.

Verwenden von Transformern als Dokumentprozessoren

Ein Transformer oder eine Folge von Transformern kann als Dokumentprozessor ausgeführt werden.

Beispielsweise können Sie den Transformer `RemoveTags` als Prozessor an einem HTML-Dokument ausführen. Der Transformer entfernt die HTML-Tags aus dem Dokument, bevor das Dokument von einem Parser nach Ankern durchsucht wird.

Konfigurieren Sie zu diesem Zweck die Parserformatkomponente mit dem Dokumentprozessor `ProcessByTransformers` und schachteln die Transformer in die Komponente.

Verwenden von Transformern in Serialisierungsankern

Transformer können in Serialisierungsankern eingesetzt werden, die in das Ausgabedokument schreiben, zum Beispiel `ContentSerializer`. Die Daten werden vom Transformer modifiziert, bevor sie der Serializer in das Dokument schreibt.

Beispiel: Ein `ContentSerializer` schreibt den Inhalt des Datenbehälters `DoctorName` in ein Ausgabedokument. Sie können den `ContentSerializer` mit einem `AddString`-Transformer konfigurieren, der dem Inhalt das Präfix „Dr. “ hinzufügt. Angenommen, die XML-Eingabe hat folgende Form:

```
<DoctorName>Albert Schweitzer</DoctorName>
```

Der Transformer bearbeitet den Inhalt, sodass die folgende Ausgabe erzeugt wird:

```
Dr. Albert Schweitzer
```

Sie können Transformer zur Eigenschaft `default_transformers` eines Serializers hinzufügen. Diese Transformer werden vor dem Schreiben des Ausgabedokuments in allen `ContentSerializer`-Serialisierungsankern ausgeführt.

Verwenden von Transformern in Aktionen

Bei einigen Aktionen, beispielsweise **SetValue** und **Map**, können Sie Transformer auf die Ausgabe der Aktion anwenden. Weitere Informationen hierzu finden Sie unter [“Überblick über die Aktionen” auf Seite 301](#).

Standardeigenschaften von Transformern

In der folgenden Tabelle werden die Standardeigenschaften von Transformern beschrieben:

Eigenschaft	Definition
name	Eine beschreibende Kennzeichnung für die Komponente. Diese Kennzeichnung wird in der Protokolldatei sowie der Ereignisansicht angezeigt. Ermitteln Sie anhand der Eigenschaft name , welche Komponente das Ereignis verursacht hat.
remark	Ein benutzerdefinierter Kommentar, der den Zweck oder die Aktion der Komponente beschreibt.
disabled	Legt fest, ob die Komponente und alle zugehörigen untergeordneten Komponenten vom Skript ignoriert werden. Verwenden Sie diese Eigenschaft zum Testen, Debuggen und Ändern eines Skripts. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none">- Ausgewählt. Das Skript ignoriert die Komponente.- Gelöscht. Das Skript wendet die Komponente an. Standardmäßig ist die Eigenschaft deaktiviert.
optional	Legt fest, ob ein Komponentenfehler den Ausfall der übergeordneten Komponente verursacht. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none">- Ausgewählt. Ein Komponentenfehler führt nicht zum Ausfall der übergeordneten Komponente.- Gelöscht. Ein Komponentenfehler verursacht den Ausfall der übergeordneten Komponente. Standardwert ist "Gelöscht". Weitere Informationen über Komponentenfehler finden Sie in “Fehlerbehandlung” auf Seite 410 .

Transformer-Komponenten: Referenz

Transformer dienen zum Ändern von Daten.

AbsURL

Der Transformer **AbsURL** wandelt einen relativen Dateipfad bzw. eine URL in einen absoluten Pfad um.

Lautet die Eingabe zum Beispiel `test.html` und die Basis-URL `http://www.example.com`, lautet die Ausgabe `http://www.example.com/test.html`.

Wenn die Eingabe ein absoluter Pfad ist, ändert sie der Transformer nicht.

Die nachstehende Tabelle beschreibt die Eigenschaften des Transformers **AbsURL**:

Eigenschaft	Beschreibung
base_URL	Definiert den Basispfad bzw. die URL.
disabled	Legt fest, ob die Komponente und alle zugehörigen untergeordneten Komponenten vom Skript ignoriert werden. Verwenden Sie diese Eigenschaft zum Testen, Debuggen und Ändern eines Skripts. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none">- Ausgewählt. Das Skript ignoriert die Komponente.- Gelöscht. Das Skript wendet die Komponente an. Standardmäßig ist die Eigenschaft deaktiviert.
name	Eine beschreibende Kennzeichnung für die Komponente. Diese Kennzeichnung wird in der Protokolldatei sowie der Ereignisansicht angezeigt. Ermitteln Sie anhand der Eigenschaft name , welche Komponente das Ereignis verursacht hat.
optional	Legt fest, ob ein Komponentenfehler den Ausfall der übergeordneten Komponente verursacht. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none">- Ausgewählt. Ein Komponentenfehler führt nicht zum Ausfall der übergeordneten Komponente.- Gelöscht. Ein Komponentenfehler verursacht den Ausfall der übergeordneten Komponente. Standardwert ist "Gelöscht". Weitere Informationen über Komponentenfehler finden Sie in "Fehlerbehandlung" auf Seite 410 .
remark	Ein benutzerdefinierter Kommentar, der den Zweck oder die Aktion der Komponente beschreibt.

AddEmptyTagsTransformer

Der Transformer **AddEmptyTagsTransformer** prüft, ob alle im Schema definierten Elemente in der XML-Eingabe vorhanden sind. Ist dies nicht der Fall, fügt er leere Elemente zur XML-Eingabe hinzu. Dies ist ein XML-zu-XML-Transformer.

Die nachstehende Tabelle beschreibt die Eigenschaften des Transformers **AddEmptyTagsTransformer**:

Eigenschaft	Beschreibung
disabled	Legt fest, ob die Komponente und alle zugehörigen untergeordneten Komponenten vom Skript ignoriert werden. Verwenden Sie diese Eigenschaft zum Testen, Debuggen und Ändern eines Skripts. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none">- Ausgewählt. Das Skript ignoriert die Komponente.- Gelöscht. Das Skript wendet die Komponente an. Standardmäßig ist die Eigenschaft deaktiviert.
name	Eine beschreibende Kennzeichnung für die Komponente. Diese Kennzeichnung wird in der Protokolldatei sowie der Ereignisansicht angezeigt. Ermitteln Sie anhand der Eigenschaft name , welche Komponente das Ereignis verursacht hat.
remark	Ein benutzerdefinierter Kommentar, der den Zweck oder die Aktion der Komponente beschreibt.
root_element	Definiert das Stammelement des XML-Dokuments.

AddString

Der Transformer **AddString** fügt Strings vor und nach dem Eingabetext hinzu.

Die nachstehende Tabelle beschreibt die Eigenschaften des Transformers **AddString**:

Eigenschaft	Beschreibung
disabled	Legt fest, ob die Komponente und alle zugehörigen untergeordneten Komponenten vom Skript ignoriert werden. Verwenden Sie diese Eigenschaft zum Testen, Debuggen und Ändern eines Skripts. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none">- Ausgewählt. Das Skript ignoriert die Komponente.- Gelöscht. Das Skript wendet die Komponente an. Standardmäßig ist die Eigenschaft deaktiviert.
name	Eine beschreibende Kennzeichnung für die Komponente. Diese Kennzeichnung wird in der Protokolldatei sowie der Ereignisansicht angezeigt. Ermitteln Sie anhand der Eigenschaft name , welche Komponente das Ereignis verursacht hat.
optional	Legt fest, ob ein Komponentenfehler den Ausfall der übergeordneten Komponente verursacht. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none">- Ausgewählt. Ein Komponentenfehler führt nicht zum Ausfall der übergeordneten Komponente.- Gelöscht. Ein Komponentenfehler verursacht den Ausfall der übergeordneten Komponente. Standardwert ist "Gelöscht". Weitere Informationen über Komponentenfehler finden Sie in "Fehlerbehandlung" auf Seite 410 .
pre	Definiert den vor dem Text einzufügenden String.
post	Definiert den nach dem Text einzufügenden String.
remark	Ein benutzerdefinierter Kommentar, der den Zweck oder die Aktion der Komponente beschreibt.

Online-Beispiel

Ein Online-Beispiel für diesen Anker befindet sich in der Datei `samples\Projects\Transformers_Example\Transformers_Example.cmw`. Der erste Content-Anker im Parser ist mit einem `AddString-Transformer` konfiguriert.

Base64Decode

Der Transformer **Base64Decode** wandelt die MIME-Codierung base64 in einen binären String um.

Die nachstehende Tabelle beschreibt die Eigenschaften des Transformers **Base64Decode**:

Eigenschaft	Beschreibung
disabled	Legt fest, ob die Komponente und alle zugehörigen untergeordneten Komponenten vom Skript ignoriert werden. Verwenden Sie diese Eigenschaft zum Testen, Debuggen und Ändern eines Skripts. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none">- Ausgewählt. Das Skript ignoriert die Komponente.- Gelöscht. Das Skript wendet die Komponente an. Standardmäßig ist die Eigenschaft deaktiviert.
name	Eine beschreibende Kennzeichnung für die Komponente. Diese Kennzeichnung wird in der Protokolldatei sowie der Ereignisansicht angezeigt. Ermitteln Sie anhand der Eigenschaft name , welche Komponente das Ereignis verursacht hat.
optional	Legt fest, ob ein Komponentenfehler den Ausfall der übergeordneten Komponente verursacht. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none">- Ausgewählt. Ein Komponentenfehler führt nicht zum Ausfall der übergeordneten Komponente.- Gelöscht. Ein Komponentenfehler verursacht den Ausfall der übergeordneten Komponente. Standardwert ist "Gelöscht". Weitere Informationen über Komponentenfehler finden Sie in "Fehlerbehandlung" auf Seite 410 .
remark	Ein benutzerdefinierter Kommentar, der den Zweck oder die Aktion der Komponente beschreibt.
tolerance	Steuert, wie der Transformer Leerzeichen oder nicht in Base64 codierte Abschnitte in der Eingabe verarbeitet. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none">- ignore_white_spaces. Verarbeitet alle Zeichen außer Leerzeichen. Standard.- ignore_none. Verarbeitet alle Zeichen.- ignore_non_base64. Verarbeitet nur base-64-Zeichen.

Base64Encode

Der Transformer **Base64Encode** wandelt einen binären String in die Codierung base64 MIME um. Dies ist für binäre Daten in XML nützlich.

Die nachstehende Tabelle beschreibt die Eigenschaften des Transformers **Base64Encode**:

Eigenschaft	Beschreibung
disabled	Legt fest, ob die Komponente und alle zugehörigen untergeordneten Komponenten vom Skript ignoriert werden. Verwenden Sie diese Eigenschaft zum Testen, Debuggen und Ändern eines Skripts. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none">- Ausgewählt. Das Skript ignoriert die Komponente.- Gelöscht. Das Skript wendet die Komponente an. Standardmäßig ist die Eigenschaft deaktiviert.
name	Eine beschreibende Kennzeichnung für die Komponente. Diese Kennzeichnung wird in der Protokolldatei sowie der Ereignisansicht angezeigt. Ermitteln Sie anhand der Eigenschaft name , welche Komponente das Ereignis verursacht hat.
optional	Legt fest, ob ein Komponentenfehler den Ausfall der übergeordneten Komponente verursacht. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none">- Ausgewählt. Ein Komponentenfehler führt nicht zum Ausfall der übergeordneten Komponente.- Gelöscht. Ein Komponentenfehler verursacht den Ausfall der übergeordneten Komponente. Standardwert ist "Gelöscht". Weitere Informationen über Komponentenfehler finden Sie in "Fehlerbehandlung" auf Seite 410 .
remark	Ein benutzerdefinierter Kommentar, der den Zweck oder die Aktion der Komponente beschreibt.

BidiConvert

Der Transformer **BidiConvert** kehrt Strings um, die in von rechts nach links (RTL, "right-to-left") geschriebenen Sprachen, wie etwa Hebräisch und Arabisch, verfasst sind. Die Eingabe muss im RTL-Format vorliegen. Die Ausgabe ist von links nach rechts geschrieben.

Der Transformer **BidiConvert** arbeitet unter Windows, wo die Standardsprache RTL ist. Verwenden Sie für einen ähnlichen Transformer, der auf allen Plattformen ausgeführt werden kann, **hebrewBidi**. Die beiden Transformer verwenden ein wenig unterschiedliche Algorithmen, die gelegentlich unterschiedliche Ergebnisse ausgeben.

Die nachstehende Tabelle beschreibt die Eigenschaften des Transformers **BidiConvert**:

Eigenschaft	Beschreibung
disabled	Legt fest, ob die Komponente und alle zugehörigen untergeordneten Komponenten vom Skript ignoriert werden. Verwenden Sie diese Eigenschaft zum Testen, Debuggen und Ändern eines Skripts. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none">- Ausgewählt. Das Skript ignoriert die Komponente.- Gelöscht. Das Skript wendet die Komponente an. Standardmäßig ist die Eigenschaft deaktiviert.
name	Eine beschreibende Kennzeichnung für die Komponente. Diese Kennzeichnung wird in der Protokolldatei sowie der Ereignisansicht angezeigt. Ermitteln Sie anhand der Eigenschaft name , welche Komponente das Ereignis verursacht hat.
remark	Ein benutzerdefinierter Kommentar, der den Zweck oder die Aktion der Komponente beschreibt.

Hinweis: Diese Komponente unterstützt keine UTF-16LE-Eingabecodierung. Behelfslösung: Verwenden Sie **hebrewBidi**.

CDATADecode

Der Transformer **CDATADecode** decodiert einen `CDATA`-Abschnitt eines XML-Dokuments. Er wandelt zum Beispiel

```
<![CDATA[100 < 200]]>
```

um in

```
100 < 200
```

Hinweis: Wird das Ergebnis in XML geschrieben, codiert das Skript es wieder in XML-Standardcodierung:

```
100 &lt; 200
```

Die nachstehende Tabelle beschreibt die Eigenschaften des Transformers **CDATADecode**:

Eigenschaft	Beschreibung
disabled	Legt fest, ob die Komponente und alle zugehörigen untergeordneten Komponenten vom Skript ignoriert werden. Verwenden Sie diese Eigenschaft zum Testen, Debuggen und Ändern eines Skripts. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none">- Ausgewählt. Das Skript ignoriert die Komponente.- Gelöscht. Das Skript wendet die Komponente an. Standardmäßig ist die Eigenschaft deaktiviert.
name	Eine beschreibende Kennzeichnung für die Komponente. Diese Kennzeichnung wird in der Protokolldatei sowie der Ereignisansicht angezeigt. Ermitteln Sie anhand der Eigenschaft name , welche Komponente das Ereignis verursacht hat.
optional	Legt fest, ob ein Komponentenfehler den Ausfall der übergeordneten Komponente verursacht. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none">- Ausgewählt. Ein Komponentenfehler führt nicht zum Ausfall der übergeordneten Komponente.- Gelöscht. Ein Komponentenfehler verursacht den Ausfall der übergeordneten Komponente. Standardwert ist "Gelöscht". Weitere Informationen über Komponentenfehler finden Sie in "Fehlerbehandlung" auf Seite 410 .
remark	Ein benutzerdefinierter Kommentar, der den Zweck oder die Aktion der Komponente beschreibt.

CDATAEncode

Der Transformer **CDATAEncode** konvertiert eine Zeichenfolge in den `CDATA`-Abschnitt eines XML-Dokuments. Er wandelt zum Beispiel

```
100 < 200
```

um in

```
<![CDATA[100 < 200]]>
```

Die nachstehende Tabelle beschreibt die Eigenschaften des Transformers **CDataEncode**:

Eigenschaft	Beschreibung
disabled	Legt fest, ob die Komponente und alle zugehörigen untergeordneten Komponenten vom Skript ignoriert werden. Verwenden Sie diese Eigenschaft zum Testen, Debuggen und Ändern eines Skripts. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none"> - Ausgewählt. Das Skript ignoriert die Komponente. - Gelöscht. Das Skript wendet die Komponente an. Standardmäßig ist die Eigenschaft deaktiviert.
name	Eine beschreibende Kennzeichnung für die Komponente. Diese Kennzeichnung wird in der Protokolldatei sowie der Ereignisansicht angezeigt. Ermitteln Sie anhand der Eigenschaft name , welche Komponente das Ereignis verursacht hat.
optional	Legt fest, ob ein Komponentenfehler den Ausfall der übergeordneten Komponente verursacht. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none"> - Ausgewählt. Ein Komponentenfehler führt nicht zum Ausfall der übergeordneten Komponente. - Gelöscht. Ein Komponentenfehler verursacht den Ausfall der übergeordneten Komponente. Standardwert ist "Gelöscht". Weitere Informationen über Komponentenfehler finden Sie in "Fehlerbehandlung" auf Seite 410 .
remark	Ein benutzerdefinierter Kommentar, der den Zweck oder die Aktion der Komponente beschreibt.

ChangeCase

Der **ChangeCase**-Transformer ändert einen Textstring in Großbuchstaben, Kleinbuchstaben oder so, dass nur der erste Buchstabe ein Großbuchstabe ist. Dieser Transformer unterstützt die im Englischen gebräuchlichen Zeichen. Bei einigen im Englischen nicht vorkommenden Zeichen sind Fehler nicht auszuschließen. Beispielsweise wird das deutsche ß nicht in SS umgewandelt.

Die nachstehende Tabelle beschreibt die Eigenschaften des Transformers **ChangeCase**:

Eigenschaft	Beschreibung
case_type	Definiert die Groß-/Kleinschreibung der Ausgabe. Die Eigenschaft case_type hat folgende Optionen: <ul style="list-style-type: none"> - first_cap. Die Ausgabe besteht nur aus Großbuchstaben. - all_lower. Die Ausgabe besteht nur aus Kleinbuchstaben. - first_cap. Der erste Buchstabe der Ausgabe ist ein Großbuchstabe, der Rest sind Kleinbuchstaben. Standardwert ist "all_caps".
disabled	Legt fest, ob die Komponente und alle zugehörigen untergeordneten Komponenten vom Skript ignoriert werden. Verwenden Sie diese Eigenschaft zum Testen, Debuggen und Ändern eines Skripts. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none"> - Ausgewählt. Das Skript ignoriert die Komponente. - Gelöscht. Das Skript wendet die Komponente an. Standardmäßig ist die Eigenschaft deaktiviert.
name	Eine beschreibende Kennzeichnung für die Komponente. Diese Kennzeichnung wird in der Protokolldatei sowie der Ereignisansicht angezeigt. Ermitteln Sie anhand der Eigenschaft name , welche Komponente das Ereignis verursacht hat.
remark	Ein benutzerdefinierter Kommentar, der den Zweck oder die Aktion der Komponente beschreibt.

Online-Beispiel

Ein Online-Beispiel für diesen Anker befindet sich in der Datei `samples\Projects\Transformers_Example\Transformers_Example.cmw`. Der dritte Content-Anker im Parser ist mit einem `ChangeCase-Transformer` konfiguriert.

CreateGuid

Der Transformer **CreateGuid** generiert einen GUID-Bezeichner. Die daraus entstehende GUID ist jedes Mal eindeutig, wenn dieser Transformer läuft.

GUIDs können ein Format haben, das nicht zum Standard von Linux- und UNIX-Plattformen gehört. Falls Sie einen uneingeschränkt mit UNIX kompatiblen Transformer benötigen, verwenden Sie stattdessen **CreateUUID**. Weitere Informationen hierzu finden Sie unter ["CreateUUID" auf Seite 270](#).

CreateUUID

Der Transformer **CreateUUID** generiert einen UUID-Bezeichner, der mit Windows-, Linux- und UNIX-Plattformen kompatibel ist. Die daraus entstehende UUID ist jedes Mal eindeutig, wenn der Transformer läuft.

Die nachstehende Tabelle beschreibt die Eigenschaften des Transformers **CreateUUID**:

Eigenschaft	Beschreibung
disabled	Legt fest, ob die Komponente und alle zugehörigen untergeordneten Komponenten vom Skript ignoriert werden. Verwenden Sie diese Eigenschaft zum Testen, Debuggen und Ändern eines Skripts. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none">- Ausgewählt. Das Skript ignoriert die Komponente.- Gelöscht. Das Skript wendet die Komponente an. Standardmäßig ist die Eigenschaft deaktiviert.
name	Eine beschreibende Kennzeichnung für die Komponente. Diese Kennzeichnung wird in der Protokolldatei sowie der Ereignisansicht angezeigt. Ermitteln Sie anhand der Eigenschaft name , welche Komponente das Ereignis verursacht hat.
optional	Legt fest, ob ein Komponentenfehler den Ausfall der übergeordneten Komponente verursacht. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none">- Ausgewählt. Ein Komponentenfehler führt nicht zum Ausfall der übergeordneten Komponente.- Gelöscht. Ein Komponentenfehler verursacht den Ausfall der übergeordneten Komponente. Standardwert ist "Gelöscht". Weitere Informationen über Komponentenfehler finden Sie in "Fehlerbehandlung" auf Seite 410 .
remark	Ein benutzerdefinierter Kommentar, der den Zweck oder die Aktion der Komponente beschreibt.

DateFormatICU

Der Transformer **DateFormatICU** wandelt ein Datum oder eine Zeit in ein vom Benutzer definiertes Format um.

Die nachstehende Tabelle beschreibt die Eigenschaften des Transformers **DateFormatICU**:

Eigenschaft	Beschreibung
disabled	Legt fest, ob die Komponente und alle zugehörigen untergeordneten Komponenten vom Skript ignoriert werden. Verwenden Sie diese Eigenschaft zum Testen, Debuggen und Ändern eines Skripts. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none">- Ausgewählt. Das Skript ignoriert die Komponente.- Gelöscht. Das Skript wendet die Komponente an. Standardmäßig ist die Eigenschaft deaktiviert.
input_format	Definiert das Format des eingegebenen Datums, zum Beispiel <code>d/M/yy</code> . Geben Sie das Format ein oder wählen Sie einen Datenbehälter, der das Format enthält.
name	Eine beschreibende Kennzeichnung für die Komponente. Diese Kennzeichnung wird in der Protokolldatei sowie der Ereignisansicht angezeigt. Ermitteln Sie anhand der Eigenschaft name , welche Komponente das Ereignis verursacht hat.
optional	Legt fest, ob ein Komponentenfehler den Ausfall der übergeordneten Komponente verursacht. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none">- Ausgewählt. Ein Komponentenfehler führt nicht zum Ausfall der übergeordneten Komponente.- Gelöscht. Ein Komponentenfehler verursacht den Ausfall der übergeordneten Komponente. Standardwert ist "Gelöscht". Weitere Informationen über Komponentenfehler finden Sie in "Fehlerbehandlung" auf Seite 410 .
output_format	Definiert das Format des ausgegebenen Datums, zum Beispiel <code>MM/dd/yyyy</code> . Geben Sie das Format ein oder wählen Sie einen Datenbehälter, der das Format enthält.
remark	Ein benutzerdefinierter Kommentar, der den Zweck oder die Aktion der Komponente beschreibt.

Beispiel

Angenommen, Sie konfigurieren einen `DateFormatICU`-Transformer mit:

```
input_format = "d/M/yy"
output_format = "MM/dd/yyyy"
```

Wenn die Eingabe

`13/3/05`

ist, lautet die Ausgabe

`03/13/2005`

Unterstützte Formate

Der Transformer **DateFormatICU** verwendet zur Darstellung des Datums- und Uhrzeitformats die ICU-Konventionen. In der folgenden Tabelle sind die Symbole aufgeführt, die Sie in den Formatmustern verwenden können: Weitere Informationen finden Sie unter

<http://icu.sourceforge.net/apiref/icu4c/classSimpleDateFormat.html>

Formatmuster	Bedeutung	Typ	Beispiele
G	Epochenbezeichner	Text	AD
y	Jahr	Zahl	1996
u	Erweitertes Jahr	Zahl	-200, Bedeutung: 201 vor Christus
M	Monat des Jahres	Text oder Zahl	Juli 07
d	Tag des Monats	Zahl	10
h	Stunde im AM/PM-Format (1-12)	Zahl	12
H	Stunde des Tages (0-23)	Zahl	0
m	Minute der Stunde	Zahl	30
s	Sekunde der Minute	Zahl	55
S	Sekundenbruchzahl	Zahl	978
E	Tag der Woche	Text	Dienstag
e	Tag der Woche (lokal 1-7)	Zahl	2
D	Tag des Jahres	Zahl	189
F	Wochentag des Monats	Zahl	2, Bedeutung: 2. Mittwoch im Juli
w	Woche des Jahres	Zahl	27
W	Woche des Monats	Zahl	2
a	AM/PM-Bezeichner	Text	PM
k	Stunde des Tages (1-24)	Zahl	24
K	Stunde im AM/PM-Format (0-11)	Zahl	0
z	Zeitzone	Uhrzeit	Pacific Standard Time
Z	Zeitzone (RFC 822)	Zahl	-0800
v	Zeitzone (generisch)	Text	Pacific Time
g	Tag des julianischen Kalenders	Zahl	2451334
A	Millisekunde des Tages	Zahl	69540000

Formatmuster	Bedeutung	Typ	Beispiele
' '	Der Text zwischen den einzelnen Anführungsstrichen wird als Zeichenfolgenliteral interpretiert.	Text	'Heute ist der 'dd/MM/yyyy' erzeugt eine Ausgabe wie die folgende: Heute ist der 15/03/2005
' '	Einzelnes Anführungsstrichliteral	Text	'o'clock' erzeugt die folgende Ausgabe: o'clock

Das Format wird durch die Anzahl der Formatmuster näher definiert.

- Text: Bei vier oder mehr Formatmustern wird das ausführliche Format verwendet. Bei weniger als vier Formatmustern wird eine Kurzform oder Abkürzung verwendet, sofern diese existiert. Beispiel: Wenn `EEEE` die Ausgabe `Monday` erzeugt, dann erzeugt `EEE` die Ausgabe `Mon`.
- Zahlen: Die Anzahl der Formatmuster gibt die Mindestanzahl der Stellen an. Kürzere Zahlen erhalten eine vorangestellte Null. Beispiel: Wenn `m` die Ausgabe `6` erzeugt, dann erzeugt `mm` die Ausgabe `06`.
- Jahreszahlen: `yy` ist eine zweistellige Jahreszahl, `yyyy` eine vierstellige. Beispiel: Wenn `yy` die Ausgabe `05` erzeugt, dann erzeugt `yyyy` die Ausgabe `2005`.
- Monatsangaben: Wenn `M` die Ausgabe `1` erzeugt, dann erzeugt `MM` die Ausgabe `01`, `MMM` die Ausgabe `Jan` und `MMMM` die Ausgabe `January`.

Alle nicht-alphabetischen Zeichen werden als Literale interpretiert, auch wenn sie nicht in einzelne Anführungsstriche gesetzt sind. Beispielsweise erzeugt `dd/MM/yyyy HH:mm` die Ausgabe `15/03/2005 13:15`.

Dos96HebToAscii

Der Transformer **Dos96HebToAscii** konvertiert die hebräische 7-Bit-Codierung zur Windows-1255-Codepage.

DynamicTable

Die Komponente **DynamicTable** definiert einen Datenbehälter, der eine Lookup-Tabelle enthält. Die Tabelle wird vom Transformer "**LookupTransformer**" verwendet.

Die nachstehende Tabelle beschreibt die Eigenschaften der Komponente **DynamicTable**:

Eigenschaft	Beschreibung
table	Definiert den Datenbehälter, der die Tabelle enthält.

EbcdicToAscii

Der Transformer **EbcdicToAscii** wandelt EBCDIC- in ASCII-Text um.

EDIFACTValidation

Der Validator **EDIFACTValidation** testet, ob ein Quell-String eine gültige EDIFACT-Nachricht ist.

Die nachstehende Tabelle beschreibt die Eigenschaften des Validators **EDIFACTValidation**:

Eigenschaft	Beschreibung
disabled	Legt fest, ob die Komponente und alle zugehörigen untergeordneten Komponenten vom Skript ignoriert werden. Verwenden Sie diese Eigenschaft zum Testen, Debuggen und Ändern eines Skripts. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none">- Ausgewählt. Das Skript ignoriert die Komponente.- Gelöscht. Das Skript wendet die Komponente an. Standardmäßig ist die Eigenschaft deaktiviert.
Aktiviert	Legt die Einstellung für param1 fest.
name	Eine beschreibende Kennzeichnung für die Komponente. Diese Kennzeichnung wird in der Protokolldatei sowie der Ereignisansicht angezeigt. Ermitteln Sie anhand der Eigenschaft name , welche Komponente das Ereignis verursacht hat.
optional	Legt fest, ob ein Komponentenfehler den Ausfall der übergeordneten Komponente verursacht. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none">- Ausgewählt. Ein Komponentenfehler führt nicht zum Ausfall der übergeordneten Komponente.- Gelöscht. Ein Komponentenfehler verursacht den Ausfall der übergeordneten Komponente. Standardwert ist "Gelöscht". Weitere Informationen über Komponentenfehler finden Sie in "Fehlerbehandlung" auf Seite 410 .
param1	Legt fest, ob die Eingabe optional ist. param1 wird als is_optional bezeichnet und hat nur eine Eigenschaft, enabled . enabled hat folgende Optionen: <ul style="list-style-type: none">- Ausgewählt. Die Eingabedaten sind optional.- Gelöscht. Die Eingabedaten sind verpflichtend.
param2	Definiert einen EDI-Datentyp. param2 wird als input_type bezeichnet und hat nur eine Eigenschaft, value . value ist ein hartcodierter String oder ein Datenbehälter.
param3	Definiert einen Bereich von ganzen Zahlen. param3 wird als minmax_limits bezeichnet und hat nur eine Eigenschaft, value . value ist ein hartcodierter String oder ein Datenbehälter, der zwei, durch einen Bindestrich getrennte, ganze Zahlen festlegt.
param4	Definiert eine Liste von Werten. param4 wird als enumerations bezeichnet und hat nur eine Eigenschaft, value . value ist ein hartcodierter String oder Datenbehälter, der eine kommagetrennte Liste von Strings oder ganzen Zahlen angibt.
remark	Ein benutzerdefinierter Kommentar, der den Zweck oder die Aktion der Komponente beschreibt.
value	Definiert einen Wert für param1 , param2 oder param3

Hinweis: Diese Komponente ist veraltet. Der IntelliScript-Editor zeigt dies für veraltete Projekte an. Verwenden Sie sie nicht in neuen Skripten. **Umgehung:** Verwenden Sie andere Validator-Komponenten.

EncodeAsUrl

Der Transformer **EncodeAsUrl** codiert Leerzeichen und Sonderzeichen wie für eine URL erforderlich. Die Zeichen werden als Prozentzeichen codiert (%), gefolgt von einer hexadezimalen Zahl.

Beispiel: Der Transformer **EncodeAsUrl** wandelt zum Beispiel

```
http://www.example.com?name=John Doe
```

um in

`http://www.example.com?name=John%20Doe`

Hinweis: Klammerzeichen werden nicht codiert.

Die nachstehende Tabelle beschreibt die Eigenschaften des Transformers **EncodeAsUrl**:

Eigenschaft	Beschreibung
disabled	Legt fest, ob die Komponente und alle zugehörigen untergeordneten Komponenten vom Skript ignoriert werden. Verwenden Sie diese Eigenschaft zum Testen, Debuggen und Ändern eines Skripts. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none">- Ausgewählt. Das Skript ignoriert die Komponente.- Gelöscht. Das Skript wendet die Komponente an. Standardmäßig ist die Eigenschaft deaktiviert.
name	Eine beschreibende Kennzeichnung für die Komponente. Diese Kennzeichnung wird in der Protokolldatei sowie der Ereignisansicht angezeigt. Ermitteln Sie anhand der Eigenschaft name , welche Komponente das Ereignis verursacht hat.
remark	Ein benutzerdefinierter Kommentar, der den Zweck oder die Aktion der Komponente beschreibt.

Online-Beispiel

Ein Online-Beispiel für diesen Anker befindet sich in der Datei `samples\Projects\Transformers_Example\Transformers_Example.cmw`. Der vierte Content-Anker im Parser ist mit einem **EncodeAsUrl**-Transformer konfiguriert.

Encoder

Der Transformer **Encoder** wandelt Text aus einer Codepage in eine andere um.

Die nachstehende Tabelle beschreibt die Eigenschaften des Transformers **Encoder**:

Eigenschaft	Beschreibung
add_prefix	Fügt eine Byte Order Mark (BOM) hinzu, wenn die Ausgabecodierung UTF-16LE oder UTF-16 ist.
disabled	Legt fest, ob die Komponente und alle zugehörigen untergeordneten Komponenten vom Skript ignoriert werden. Verwenden Sie diese Eigenschaft zum Testen, Debuggen und Ändern eines Skripts. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none">- Ausgewählt. Das Skript ignoriert die Komponente.- Gelöscht. Das Skript wendet die Komponente an. Standardmäßig ist die Eigenschaft deaktiviert.
input_code_page	Definiert die Codepage des Eingabetexts.
name	Eine beschreibende Kennzeichnung für die Komponente. Diese Kennzeichnung wird in der Protokolldatei sowie der Ereignisansicht angezeigt. Ermitteln Sie anhand der Eigenschaft name , welche Komponente das Ereignis verursacht hat.

Eigenschaft	Beschreibung
optional	Legt fest, ob ein Komponentenfehler den Ausfall der übergeordneten Komponente verursacht. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none"> - Ausgewählt. Ein Komponentenfehler führt nicht zum Ausfall der übergeordneten Komponente. - Gelöscht. Ein Komponentenfehler verursacht den Ausfall der übergeordneten Komponente. Standardwert ist "Gelöscht". Weitere Informationen über Komponentenfehler finden Sie in "Fehlerbehandlung" auf Seite 410.
output_code_page	Definiert die Codepage des Ausgabetexts.
remark	Ein benutzerdefinierter Kommentar, der den Zweck oder die Aktion der Komponente beschreibt.

FormatNumber

Der **FormatNumber**-Transformer formatiert eine Zahl, indem er ihr ein Vorzeichen, ein Dezimalzeichen, vorangestellte oder angehängte Nullen und eine Einheit hinzufügt.

Die nachstehende Tabelle beschreibt die Eigenschaften des **FormatNumber**-Transformers:

Eigenschaft	Beschreibung
disabled	Legt fest, ob die Komponente und alle zugehörigen untergeordneten Komponenten vom Skript ignoriert werden. Verwenden Sie diese Eigenschaft zum Testen, Debuggen und Ändern eines Skripts. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none"> - Ausgewählt. Das Skript ignoriert die Komponente. - Gelöscht. Das Skript wendet die Komponente an. Standardmäßig ist die Eigenschaft deaktiviert.
insert_decimal_point	Definiert das Dezimalzeichen. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none"> - Komma. Das Dezimalzeichen ist ein Komma. - Keine. Die Ausgabe hat kein Dezimalzeichen. - Punkt. Das Dezimalzeichen ist ein Punkt. Standardwert ist "Keine".
name	Eine beschreibende Kennzeichnung für die Komponente. Diese Kennzeichnung wird in der Protokolldatei sowie der Ereignisansicht angezeigt. Ermitteln Sie anhand der Eigenschaft name , welche Komponente das Ereignis verursacht hat.
number_of_decimals	Füllt die Kommastellen bis zur angegebenen Größe mit angehängten Nullen auf. Standardwert ist 0.
optional	Legt fest, ob ein Komponentenfehler den Ausfall der übergeordneten Komponente verursacht. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none"> - Ausgewählt. Ein Komponentenfehler führt nicht zum Ausfall der übergeordneten Komponente. - Gelöscht. Ein Komponentenfehler verursacht den Ausfall der übergeordneten Komponente. Standardwert ist "Gelöscht". Weitere Informationen über Komponentenfehler finden Sie in "Fehlerbehandlung" auf Seite 410.
remark	Ein benutzerdefinierter Kommentar, der den Zweck oder die Aktion der Komponente beschreibt.

Eigenschaft	Beschreibung
sign	Legt das Zeichen der Ausgabezahl fest. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none"> - un_signed. Löscht ein eventuell vorhandenes Vorzeichen. - leading_sign. Vor der Ausgabezahl wird ein Plus oder Minus hinzugefügt. - trailing_sign. Nach der Ausgabezahl wird ein Plus oder Minus hinzugefügt. - Nur negatives Vorzeichen. Bei negativen Zahlen wird ein Minuszeichen hinzugefügt. - Wie in der Quelle. Verändert das Vorzeichen der Eingabe nicht. Standardwert ist "trailing_sign".
size_of_integer_part	Füllt den ganzzahligen Teil bis zur angegebenen Größe mit vorangestellten Nullen auf. Standardwert ist 0.
unit_type	Definiert die Maßeinheit nach der Zahl. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none"> - cm - in - m - mm - undefiniert. Es wird keine Maßeinheit hinzugefügt. Standardwert ist "Undefiniert".

FromFloat

Der Transformer **FromFloat** wandelt die binäre Darstellung einer Gleitkommazahl in einen ASCII-String um. Die Umwandlung wird in der Eingabecodierung mit der Eingabe-Bytereihenfolge durchgeführt.

Die nachstehende Tabelle beschreibt die Eigenschaften des Transformers **FromFloat**:

Eigenschaft	Beschreibung
disabled	Legt fest, ob die Komponente und alle zugehörigen untergeordneten Komponenten vom Skript ignoriert werden. Verwenden Sie diese Eigenschaft zum Testen, Debuggen und Ändern eines Skripts. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none"> - Ausgewählt. Das Skript ignoriert die Komponente. - Gelöscht. Das Skript wendet die Komponente an. Standardmäßig ist die Eigenschaft deaktiviert.
name	Eine beschreibende Kennzeichnung für die Komponente. Diese Kennzeichnung wird in der Protokolldatei sowie der Ereignisansicht angezeigt. Ermitteln Sie anhand der Eigenschaft name , welche Komponente das Ereignis verursacht hat.
optional	Legt fest, ob ein Komponentenfehler den Ausfall der übergeordneten Komponente verursacht. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none"> - Ausgewählt. Ein Komponentenfehler führt nicht zum Ausfall der übergeordneten Komponente. - Gelöscht. Ein Komponentenfehler verursacht den Ausfall der übergeordneten Komponente. Standardwert ist "Gelöscht". Weitere Informationen über Komponentenfehler finden Sie in "Fehlerbehandlung" auf Seite 410 .
remark	Ein benutzerdefinierter Kommentar, der den Zweck oder die Aktion der Komponente beschreibt.
size	Legt die Größe der Eingabezahl fest. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none"> - single_precision_32_bit - double_precision_64_bit Standardwert ist single_precision_32_bit.

Hinweis: Diese Komponente unterstützt keine UTF-16LE-Eingabecodierung.

FromInteger

Der Transformer **FromInteger** wandelt eine ganze Zahl aus der binären Darstellung in einen ASCII-String im Dezimal-, Oktal- oder Hexadezimalformat um. Die Umwandlung wird in der Eingabecodierung mit der Eingabe-Bytereihenfolge durchgeführt.

Die nachstehende Tabelle beschreibt die Eigenschaften des Transformers **FromInteger**:

Eigenschaft	Beschreibung
disabled	Legt fest, ob die Komponente und alle zugehörigen untergeordneten Komponenten vom Skript ignoriert werden. Verwenden Sie diese Eigenschaft zum Testen, Debuggen und Ändern eines Skripts. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none">- Ausgewählt. Das Skript ignoriert die Komponente.- Gelöscht. Das Skript wendet die Komponente an. Standardmäßig ist die Eigenschaft deaktiviert.
name	Eine beschreibende Kennzeichnung für die Komponente. Diese Kennzeichnung wird in der Protokolldatei sowie der Ereignisansicht angezeigt. Ermitteln Sie anhand der Eigenschaft name , welche Komponente das Ereignis verursacht hat.
optional	Legt fest, ob ein Komponentenfehler den Ausfall der übergeordneten Komponente verursacht. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none">- Ausgewählt. Ein Komponentenfehler führt nicht zum Ausfall der übergeordneten Komponente.- Gelöscht. Ein Komponentenfehler verursacht den Ausfall der übergeordneten Komponente. Standardwert ist "Gelöscht". Weitere Informationen über Komponentenfehler finden Sie in "Fehlerbehandlung" auf Seite 410 .
remark	Ein benutzerdefinierter Kommentar, der den Zweck oder die Aktion der Komponente beschreibt.
signed	Legt fest, ob die Ausgabezahl ein Vorzeichen hat. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none">- Ausgewählt. Die Ausgabezahl hat ein Vorzeichen.- Gelöscht. Die Ausgabezahl hat kein Vorzeichen. Standardwert ist "Gelöscht".
size	Definiert die Größe der binären Eingabe in Byte. Unterstützt werden die Werte 1 bis 8. Standardwert ist 1.
to_base	Definiert die Basis der Ausgabe. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none">- Dezimal. Basis 10.- Hexadezimal. Basis 16 mit Großbuchstaben A-F.- Hexadezimal mit Kleinbuchstaben. Basis 16 mit Kleinbuchstaben a-f.- Oktal. Basis 8. Standardwert ist "Dezimal".

Hinweis: Diese Komponente unterstützt keine UTF-16LE-Eingabecodierung.

FromPackDecimal

Der Transformer **FromPackDecimal** wandelt eine Zahl von gepackten Dezimalzahlen in eine ASCII-Zeichenkette um. Die Umwandlung wird in der Eingabecodierung mit der Eingabe-Bytereihenfolge durchgeführt.

Die nachstehende Tabelle beschreibt die Eigenschaften des Transformers **FromPackDecimal**:

Eigenschaft	Beschreibung
disabled	Legt fest, ob die Komponente und alle zugehörigen untergeordneten Komponenten vom Skript ignoriert werden. Verwenden Sie diese Eigenschaft zum Testen, Debuggen und Ändern eines Skripts. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none">- Ausgewählt. Das Skript ignoriert die Komponente.- Gelöscht. Das Skript wendet die Komponente an. Standardmäßig ist die Eigenschaft deaktiviert.
name	Eine beschreibende Kennzeichnung für die Komponente. Diese Kennzeichnung wird in der Protokolldatei sowie der Ereignisansicht angezeigt. Ermitteln Sie anhand der Eigenschaft name , welche Komponente das Ereignis verursacht hat.
optional	Legt fest, ob ein Komponentenfehler den Ausfall der übergeordneten Komponente verursacht. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none">- Ausgewählt. Ein Komponentenfehler führt nicht zum Ausfall der übergeordneten Komponente.- Gelöscht. Ein Komponentenfehler verursacht den Ausfall der übergeordneten Komponente. Standardwert ist "Gelöscht". Weitere Informationen über Komponentenfehler finden Sie in "Fehlerbehandlung" auf Seite 410 .
remark	Ein benutzerdefinierter Kommentar, der den Zweck oder die Aktion der Komponente beschreibt.

Hinweis: Diese Komponente unterstützt keine UTF-16LE-Eingabecodierung.

FromSignedDecimal

Der Transformer **FromSignedDecimal** wandelt eine Zahl aus einer Dezimalzahl mit Vorzeichen in eine ASCII-String-Darstellung um. Die Umwandlung wird in der Eingabecodierung mit der Eingabe-Bytereihenfolge durchgeführt.

Die nachstehende Tabelle beschreibt die Eigenschaften des Transformers **FromSignedDecimal**:

Eigenschaft	Beschreibung
disabled	Legt fest, ob die Komponente und alle zugehörigen untergeordneten Komponenten vom Skript ignoriert werden. Verwenden Sie diese Eigenschaft zum Testen, Debuggen und Ändern eines Skripts. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none">- Ausgewählt. Das Skript ignoriert die Komponente.- Gelöscht. Das Skript wendet die Komponente an. Standardmäßig ist die Eigenschaft deaktiviert.
insert_sign_symbol	Definiert ein Vorzeichen für die Zahl. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none">- Nachher. Ein Plus- oder Minuszeichen wird der Ausgabezahl nachgestellt.- Vorher. Ein Plus- oder Minuszeichen wird der Ausgabezahl vorangestellt.- Nein. Die Ausgabe ist ohne Vorzeichen. Standard ist "Nein".
name	Eine beschreibende Kennzeichnung für die Komponente. Diese Kennzeichnung wird in der Protokolldatei sowie der Ereignisansicht angezeigt. Ermitteln Sie anhand der Eigenschaft name , welche Komponente das Ereignis verursacht hat.

Eigenschaft	Beschreibung
optional	Legt fest, ob ein Komponentenfehler den Ausfall der übergeordneten Komponente verursacht. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none"> - Ausgewählt. Ein Komponentenfehler führt nicht zum Ausfall der übergeordneten Komponente. - Gelöscht. Ein Komponentenfehler verursacht den Ausfall der übergeordneten Komponente. Standardwert ist "Gelöscht". Weitere Informationen über Komponentenfehler finden Sie in "Fehlerbehandlung" auf Seite 410 .
remark	Ein benutzerdefinierter Kommentar, der den Zweck oder die Aktion der Komponente beschreibt.

Hinweis: Diese Komponente unterstützt keine UTF-16LE-Eingabecodierung.

hebrewBidi

Der Transformer **hebrewBidi** kehrt einen String um, der in von rechts nach links geschriebenen Sprachen wie etwa Hebräisch und Arabisch verfasst ist.

Die Eingabe muss im RTL-Format vorliegen. Die Ausgabe ist von links nach rechts geschrieben.

HebrewDosToWindows

Der Transformer **HebrewDosToWindows** wandelt hebräische Dokumente aus der Codepage MS-DOS Hebräisch in die Windows-Codepage für Hebräisch um.

HebrewEBCDICOldCodeToWindows

Der Transformer **HebrewEBCDICOldCodeToWindows** wandelt hebräischen Text aus EBCDIC in die Codepage Windows-1255 um.

hebUniToAscii

Der Transformer **hebUniToAscii** wandelt hebräischen Text aus Unicode UTF-16 in die Codepage Windows-1255 um.

hebUtf8ToAscii

Der Transformer **hebUtf8ToAscii** wandelt hebräischen Text aus Unicode UTF-16LE in die Codepage Windows-1255 um.

HtmlEntitiesToASCII

Der Transformer **HtmlEntitiesToASCII** wandelt HTML-Entitäten in einfachen Text um. Beispielsweise konvertiert er `©` oder `©` in das Copyright-Symbol (©).

Die nachstehende Tabelle beschreibt die Eigenschaften des Transformers **HtmlEntitiesToASCII**:

Eigenschaft	Beschreibung
disabled	Legt fest, ob die Komponente und alle zugehörigen untergeordneten Komponenten vom Skript ignoriert werden. Verwenden Sie diese Eigenschaft zum Testen, Debuggen und Ändern eines Skripts. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none">- Ausgewählt. Das Skript ignoriert die Komponente.- Gelöscht. Das Skript wendet die Komponente an. Standardmäßig ist die Eigenschaft deaktiviert.
name	Eine beschreibende Kennzeichnung für die Komponente. Diese Kennzeichnung wird in der Protokolldatei sowie der Ereignisansicht angezeigt. Ermitteln Sie anhand der Eigenschaft name , welche Komponente das Ereignis verursacht hat.
remark	Ein benutzerdefinierter Kommentar, der den Zweck oder die Aktion der Komponente beschreibt.

Unterstützte Entitäten

Der Transformer unterstützt die Entitäten des Zeichensatzes ISO 8859-1 (Latin-1), die in der HTML 4.0-Referenz definiert sind (siehe <http://www.w3.org/TR/1998/REC-html40-19980424/sgml/entities.html>). Zu den unterstützten Entitäten zählen u. a. die folgenden:

- `&`, `<`, `>` und `"`; (bzw. `&` `<` `>` `"`)
- die numerischen Zeichencodes `�` bis `ÿ`
- Entitäten für Latin-1-Zeichen: ` ` = geschütztes Leerzeichen, `©` = Copyright-Zeichen usw.

Der Transformer unterstützt keine erweiterten Zeichen, d. h. Codes größer als 255 und Zeichen, die nicht im Zeichensatz Latin-1 enthalten sind.

Codierung der Ausgabe für große ASCII-Zeichen

Wenn die Transformer-Ausgabe höhere ASCII-Zeichen enthält, müssen Sie für die Ausgabe eine Codierung wählen, die diese Zeichen unterstützt. Das ist zum Beispiel bei Windows-1252 und UTF-16LE der Fall.

Hinweis: Fügen Sie dazu ein Codierungsattribut in die XML-Verarbeitungsanweisung ein. Anderenfalls zeigt das Developer Tool die Zeichen eventuell nicht korrekt an.

HtmlProcessor

Der Transformer **HtmlProcessor** normalisiert Leerräume entsprechend den HTML-Konventionen. Er wandelt jede Folge von Tabulatoren, Zeilenumbrüchen und Leerzeichen in ein einziges Leerzeichen um. Dieser Transformer arbeitet mit HTML-T und anderen Texttypen. Sie können ihn auch als Formatpräprozessor verwenden. Weitere Informationen hierzu finden Sie unter ["Formatpräprozessor-Komponenten: Referenz" auf Seite 190](#).

InjectFP

Der Transformer **InjectFP** fügt an einer angegebenen Stelle ein Dezimalzeichen in eine Zahl ein. Er wandelt zum Beispiel 12345 in 123.45 um.

Die nachstehende Tabelle beschreibt die Eigenschaften des Transformers **InjectFP**:

Eigenschaft	Beschreibung
digits_after_decimal_point	Legt die Anzahl der Stellen nach dem Dezimalzeichen fest.
disabled	Legt fest, ob die Komponente und alle zugehörigen untergeordneten Komponenten vom Skript ignoriert werden. Verwenden Sie diese Eigenschaft zum Testen, Debuggen und Ändern eines Skripts. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none"> - Ausgewählt. Das Skript ignoriert die Komponente. - Gelöscht. Das Skript wendet die Komponente an. Standardmäßig ist die Eigenschaft deaktiviert.
name	Eine beschreibende Kennzeichnung für die Komponente. Diese Kennzeichnung wird in der Protokolldatei sowie der Ereignisansicht angezeigt. Ermitteln Sie anhand der Eigenschaft name , welche Komponente das Ereignis verursacht hat.
optional	Legt fest, ob ein Komponentenfehler den Ausfall der übergeordneten Komponente verursacht. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none"> - Ausgewählt. Ein Komponentenfehler führt nicht zum Ausfall der übergeordneten Komponente. - Gelöscht. Ein Komponentenfehler verursacht den Ausfall der übergeordneten Komponente. Standardwert ist "Gelöscht". Weitere Informationen über Komponentenfehler finden Sie in "Fehlerbehandlung" auf Seite 410 .
remark	Ein benutzerdefinierter Kommentar, der den Zweck oder die Aktion der Komponente beschreibt.

InjectString

Der Transformer **InjectString** fügt einen String in Text ein.

Die nachstehende Tabelle beschreibt die Eigenschaften des Transformers **InjectString**:

Eigenschaft	Beschreibung
disabled	Legt fest, ob die Komponente und alle zugehörigen untergeordneten Komponenten vom Skript ignoriert werden. Verwenden Sie diese Eigenschaft zum Testen, Debuggen und Ändern eines Skripts. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none"> - Ausgewählt. Das Skript ignoriert die Komponente. - Gelöscht. Das Skript wendet die Komponente an. Standardmäßig ist die Eigenschaft deaktiviert.
injection_place	Definiert die Anzahl der Zeichen vom Beginn des Texts bis zu der Stelle, an der der String eingefügt wird.
name	Eine beschreibende Kennzeichnung für die Komponente. Diese Kennzeichnung wird in der Protokolldatei sowie der Ereignisansicht angezeigt. Ermitteln Sie anhand der Eigenschaft name , welche Komponente das Ereignis verursacht hat.
remark	Ein benutzerdefinierter Kommentar, der den Zweck oder die Aktion der Komponente beschreibt.
string_to_inject	Definiert den einzufügenden String.

InlineTable

Die Komponente **InlineTable** definiert eine Lookup-Tabelle im Skript. Die Tabelle wird vom Transformer "**LookupTransformer**" verwendet.

Die nachstehende Tabelle beschreibt die Eigenschaften der Komponente **InlineTable**:

Eigenschaft	Beschreibung
Entry	Definiert ein Paar aus key (Schlüssel) und value (Wert).
key	Definiert einen eindeutigen Eingabestring.
match_case	Legt fest, ob der key -String die Groß- und Kleinschreibung berücksichtigt. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none">- Ausgewählt. key berücksichtigt die Groß-/Kleinschreibung.- Gelöscht. key berücksichtigt die Groß-/Kleinschreibung nicht. Standardwert ist "Gelöscht".
table	Definiert eine Liste von Entry -Komponenten.
value	Definiert einen Ausgabestring.

JavaTransformer

Der Transformer **JavaTransformer** ist ein in Java implementierter benutzerdefinierter Transformer.

Die nachstehende Tabelle beschreibt die Eigenschaften des Transformers **JavaTransformer**:

Eigenschaft	Beschreibung
disabled	Legt fest, ob die Komponente und alle zugehörigen untergeordneten Komponenten vom Skript ignoriert werden. Verwenden Sie diese Eigenschaft zum Testen, Debuggen und Ändern eines Skripts. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none">- Ausgewählt. Das Skript ignoriert die Komponente.- Gelöscht. Das Skript wendet die Komponente an. Standardmäßig ist die Eigenschaft deaktiviert.
java_class	Definiert den Pfad der Java-Klasse.
method	Definiert die Methode, die auszuführen ist.
name	Eine beschreibende Kennzeichnung für die Komponente. Diese Kennzeichnung wird in der Protokolldatei sowie der Ereignisansicht angezeigt. Ermitteln Sie anhand der Eigenschaft name , welche Komponente das Ereignis verursacht hat.
optional	Legt fest, ob ein Komponentenfehler den Ausfall der übergeordneten Komponente verursacht. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none">- Ausgewählt. Ein Komponentenfehler führt nicht zum Ausfall der übergeordneten Komponente.- Gelöscht. Ein Komponentenfehler verursacht den Ausfall der übergeordneten Komponente. Standardwert ist "Gelöscht". Weitere Informationen über Komponentenfehler finden Sie in "Fehlerbehandlung" auf Seite 410 .
remark	Ein benutzerdefinierter Kommentar, der den Zweck oder die Aktion der Komponente beschreibt.

Hinweis: Diese Komponente ist veraltet. Der IntelliScript-Editor zeigt dies für veraltete Skripte an. Verwenden Sie sie nicht in neuen Skripten. Erstellen Sie anstelle dessen einen benutzerdefinierten Java-Transformer.

Weitere Informationen hierzu finden Sie unter ["Entwickeln einer benutzerdefinierten Komponente" auf Seite 445](#).

LookupTransformer

Der **LookupTransformer**-Transformer schlägt einen Wert in einer Tabelle nach.

Die nachstehende Tabelle beschreibt die Eigenschaften des **LookupTransformer**-Transformers:

Eigenschaft	Beschreibung
disabled	Legt fest, ob die Komponente und alle zugehörigen untergeordneten Komponenten vom Skript ignoriert werden. Verwenden Sie diese Eigenschaft zum Testen, Debuggen und Ändern eines Skripts. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none">- Ausgewählt. Das Skript ignoriert die Komponente.- Gelöscht. Das Skript wendet die Komponente an. Standardmäßig ist die Eigenschaft deaktiviert.
look_at	Definiert den Typ der durch den Transformer verwendeten Lookup-Tabelle. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none">- DynamicTable. Die Tabelle-Eigenschaft der look_at-Eigenschaft definiert einen Datenbehälter, der die Tabelle enthält. Weitere Informationen hierzu finden Sie unter "DynamicTable" auf Seite 273.- InlineTable. Die Tabelle-Eigenschaft der look_at-Eigenschaft definiert eine Liste von Eintrag-Komponenten mit einem Schlüssel und einem Wert. Weitere Informationen hierzu finden Sie unter "InlineTable" auf Seite 283.- XMLLookupTable. Die xml_file_name-Eigenschaft der look_at-Eigenschaft definiert den Pfad und den Dateinamen einer XML-Datei, die die Tabelle definiert. Weitere Informationen hierzu finden Sie unter "XMLLookupTable" auf Seite 299.- [TableName]. Eine auf der globalen Ebene des Skripts definierte DynamicTable, InlineTable oder XMLLookupTable. Standardwert ist "Leer".
name	Eine beschreibende Kennzeichnung für die Komponente. Diese Kennzeichnung wird in der Protokolldatei sowie der Ereignisansicht angezeigt. Ermitteln Sie anhand der Eigenschaft name , welche Komponente das Ereignis verursacht hat.
optional	Legt fest, ob ein Komponentenfehler den Ausfall der übergeordneten Komponente verursacht. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none">- Ausgewählt. Ein Komponentenfehler führt nicht zum Ausfall der übergeordneten Komponente.- Gelöscht. Ein Komponentenfehler verursacht den Ausfall der übergeordneten Komponente. Standardwert ist "Gelöscht". Weitere Informationen über Komponentenfehler finden Sie in "Fehlerbehandlung" auf Seite 410 .
remark	Ein benutzerdefinierter Kommentar, der den Zweck oder die Aktion der Komponente beschreibt.

Wenn Sie dieselbe Lookup-Tabelle mehrmals verwenden, empfiehlt es sich, eine **InlineTable** bzw. **XMLLookupTable** auf der globalen Ebene des Skripts zu definieren. In diesem Fall können Sie die Tabelle in der Eigenschaft **look_at** namentlich referenzieren.

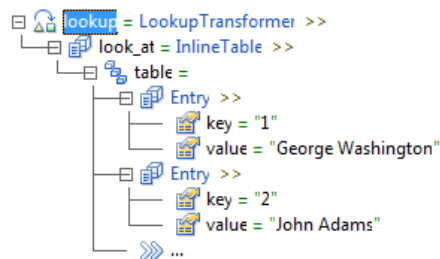
Sie können beispielsweise einen **LookupTransformer** dazu konfigurieren, Werte in der folgenden Tabelle nachzuschlagen:

Schlüssel	Wert
1	George Washington
2	John Adams
3	Thomas Jefferson
4	James Madison

Bei der Eingabe 3 gibt der Transformer den Wert `Thomas Jefferson` aus.

Definition einer Inline-Tabelle

Zur Definition einer Inline-Tabelle werden die Schlüsselwertpaare im Skript konfiguriert (siehe folgendes Beispiel):



Speichern einer XML-Nachschlagetabelle in einer Datei

Bereiten Sie eine XML-Datei vor, die dem Schema `lookupTableDefinition.xsd` entspricht. Dieses Schema befindet sich im Unterverzeichnis `doc` des Installationsverzeichnis. Das folgende XML-Dokument dient als Beispiel:

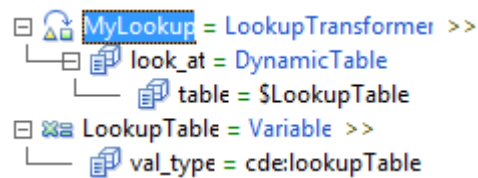
```
<?xml version="1.0" encoding="windows-1252" ?>
<lt:LookupTable xmlns:lt="http://www.itemfield.com/Engine/V4/lookupTable"
matchCase="false">
  <lt:Entry key="1" value="George Washington" />
  <lt:Entry key="2" value="John Adams" />
</lt:LookupTable>
```

Dynamische Erstellung einer XML-Nachschlagetabelle

Eine Umwandlung kann während der Laufzeit eine XML-Nachschlagetabelle erstellen. Die Umwandlung könnte beispielsweise einen zweiten Parser ausführen, der die XML-Struktur erzeugt.

Die Umwandlung muss den XML-String in einem Datenbehälter für mehrere Instanzen des Typs `cde:lookupTable` speichern. Speichern Sie jedes Schlüssel-Wert-Paar in einer Instanz des Datenbehälters. Beispiel: Sie können eine **RepeatingGroup** konfigurieren, die eine **WriteValue**-Aktion enthält. Jede Iteration der **RepeatingGroup** erstellt eine Instanz des Datenbehälters und schreibt ein Schlüssel-Wert-Paar in die Instanz.

Danach konfigurieren Sie einen **LookupTransformer** mit der Option **DynamicTable** und legen einen Datenbehälter fest.



NormalizeClosingTags

Der Transformer **NormalizeClosingTags** entfernt in der XML-Eingabe verkürzte schließende Tags aus leeren Elementen. Er wandelt `<tag/>` in `<tag></tag>` um.

Der Transformer korrigiert fehlerhaften XML-Code nicht. Er wandelt lediglich wohlgeformtes XML aus einem Stil für schließende Tags in einen anderen um.

Die nachstehende Tabelle beschreibt die Eigenschaften des Transformers **NormalizeClosingTags**:

Eigenschaft	Beschreibung
disabled	Legt fest, ob die Komponente und alle zugehörigen untergeordneten Komponenten vom Skript ignoriert werden. Verwenden Sie diese Eigenschaft zum Testen, Debuggen und Ändern eines Skripts. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none"> - Ausgewählt. Das Skript ignoriert die Komponente. - Gelöscht. Das Skript wendet die Komponente an. Standardmäßig ist die Eigenschaft deaktiviert.
name	Eine beschreibende Kennzeichnung für die Komponente. Diese Kennzeichnung wird in der Protokolldatei sowie der Ereignisansicht angezeigt. Ermitteln Sie anhand der Eigenschaft name , welche Komponente das Ereignis verursacht hat.
remark	Ein benutzerdefinierter Kommentar, der den Zweck oder die Aktion der Komponente beschreibt.

RegularExpression

Der Transformer **RegularExpression** führt am Eingabetext eine Suche mit Mustervergleich aus. Er ersetzt die Instanzen des Musters durch einen angegebenen String.

Die nachstehende Tabelle beschreibt die Eigenschaften des Transformers **RegularExpression**:

Eigenschaft	Beschreibung
disabled	Legt fest, ob die Komponente und alle zugehörigen untergeordneten Komponenten vom Skript ignoriert werden. Verwenden Sie diese Eigenschaft zum Testen, Debuggen und Ändern eines Skripts. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none"> - Ausgewählt. Das Skript ignoriert die Komponente. - Gelöscht. Das Skript wendet die Komponente an. Standardmäßig ist die Eigenschaft deaktiviert.
exp	Definiert einen regulären Ausdruck für das Suchkriterium.
name	Eine beschreibende Kennzeichnung für die Komponente. Diese Kennzeichnung wird in der Protokolldatei sowie der Ereignisansicht angezeigt. Ermitteln Sie anhand der Eigenschaft name , welche Komponente das Ereignis verursacht hat.

Eigenschaft	Beschreibung
optional	Legt fest, ob ein Komponentenfehler den Ausfall der übergeordneten Komponente verursacht. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none"> - Ausgewählt. Ein Komponentenfehler führt nicht zum Ausfall der übergeordneten Komponente. - Gelöscht. Ein Komponentenfehler verursacht den Ausfall der übergeordneten Komponente. Standardwert ist "Gelöscht". Weitere Informationen über Komponentenfehler finden Sie in "Fehlerbehandlung" auf Seite 410 .
remark	Ein benutzerdefinierter Kommentar, der den Zweck oder die Aktion der Komponente beschreibt.
replacement	Definiert den Ersatztext.

Angenommen, ein **Content**-Anker ruft folgenden Text ab:

```
transformer
```

Sie können den Anker mit einem **RegularExpression**-Transformer konfigurieren, der nach dem Muster `t.+s` sucht. Das Muster steht für den Buchstaben `t`, gefolgt von einem oder mehreren Zeichen, wiederum gefolgt von dem Buchstaben `s`. Sie konfigurieren den Transformer dahin gehend, das Muster durch den Buchstaben `x` zu ersetzen.

Das Muster stimmt mit dem Teilstring `trans` der Eingabe überein. Der Transformer ersetzt den Teilstring und gibt folgende Ausgabe zurück:

```
Xformer
```

Regulärer Ausdruck - Syntax

Ein regulärer Ausdruck definiert ein Suchmuster entsprechend einer Standardsyntax.

Die Datenprozessor-Umwandlung verwendet die Regex++-Implementierung regulärer Ausdrücke, © 1998-2003 Dr. John Maddock, Version 1.33, 18. April 2000.

Hinweis: Regex++ unterstützt Ländereinstellungen nicht.

In der folgenden Tabelle sind einige Sonderzeichen aufgeführt, die in regulären Ausdrücken verwendet werden können:

Zeichen	Bedeutung	Beispiel
*	Entspricht null oder mehr Instanzen des vorherigen Zeichens	<code>ab*c</code> entspricht <code>ac</code> , <code>abc</code> oder <code>abbbc</code> .
?	Entspricht keiner oder einer Instanz des vorherigen Zeichens	<code>ab?c</code> entspricht <code>ac</code> oder <code>abc</code> .
+	Entspricht einer oder mehr Instanzen des vorherigen Zeichens	<code>a+</code> entspricht <code>a</code> oder <code>aaaa</code> .
{}	Entspricht der angegebenen Anzahl der Instanzen des vorherigen Zeichens.	<code>ab{2}c</code> entspricht <code>abbc</code> .
[]	Entspricht einer Auswahl von Zeichen.	<code>a[bst]c</code> entspricht <code>abc</code> , <code>asc</code> oder <code>atc</code> .

Zeichen	Bedeutung	Beispiel
-	Definiert einen Zeichenbereich zwischen eckigen Klammern.	[A-Za-z] entspricht jedem Zeichen im englischen Alphabet. [A-Za-zü] entspricht allen Zeichen des lateinischen Alphabets und dem deutschen ü.
.	Findet eine Instanz eines beliebigen einzelnen Zeichens.	a.c entspricht abc, a c, oder a1c.
^	Entspricht dem Anfang des Eingabetextes.	^P. in „Peter Piper“ entspricht Pe, aber nicht Pi.
\$	Entspricht dem Ende des Eingabetextes.	r.\$ entspricht rs in "Peter Piper's peppers."
	Entspricht einem von zwei Ausdrücken.	abc def entspricht abc oder def.
()	Gruppierung	A(abc def) entspricht Aabc oder Adef.
\	Dient als Escape-Zeichen für andere Sonderzeichen und behandelt diese als literales Zeichen.	entspricht einem literalen Punkt statt einem Zeichen.

Beibehalten von Teilen des Ursprungstextes

In die Eigenschaft `exp` können Sie Teile des regulären Ausdrucks in Klammern einschließen. Außerdem können Sie in der Eigenschaft `replacement` folgende Anweisungen angeben:

- `$0` - bezeichnet den gesamten gefundenen Text, der dem regulären Ausdruck entspricht.
- `$1` - bezeichnet den Teilstring, der dem ersten in Klammern gesetzten Teil des regulären Ausdrucks entspricht.
- `$2, $3` usw. bezeichnen den Teilstring, der dem zweiten, dritten usw. Teil in Klammern entspricht

Angenommen, Sie setzen:

```
exp = abc([0-9]+) (def)
replacement = $1
```

In diesem Fall wird `abc5624def` durch `5624` ersetzt.

Oder Sie setzen:

```
exp = abc([0-9]+) (def)
replacement = $2ZYX$1
```

In diesem Fall wird `abc5624def` durch `defZYX5624` ersetzt.

RemoveMarginSpace

Der Transformer **RemoveMarginSpace** löscht vorangestellte und angehängte Leerzeichen aus dem Text.

Die nachstehende Tabelle beschreibt die Eigenschaften des Transformers **RemoveMarginSpace**:

Eigenschaft	Beschreibung
disabled	Legt fest, ob die Komponente und alle zugehörigen untergeordneten Komponenten vom Skript ignoriert werden. Verwenden Sie diese Eigenschaft zum Testen, Debuggen und Ändern eines Skripts. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none">- Ausgewählt. Das Skript ignoriert die Komponente.- Gelöscht. Das Skript wendet die Komponente an. Standardmäßig ist die Eigenschaft deaktiviert.
name	Eine beschreibende Kennzeichnung für die Komponente. Diese Kennzeichnung wird in der Protokolldatei sowie der Ereignisansicht angezeigt. Ermitteln Sie anhand der Eigenschaft name , welche Komponente das Ereignis verursacht hat.
remark	Ein benutzerdefinierter Kommentar, der den Zweck oder die Aktion der Komponente beschreibt.

RemoveRtfFormatting

Der Transformer **RemoveRtfFormatting** entfernt RTF-Formatierungsanweisungen aus dem Text.

Die nachstehende Tabelle beschreibt die Eigenschaften des Transformers **RemoveRtfFormatting**:

Eigenschaft	Beschreibung
disabled	Legt fest, ob die Komponente und alle zugehörigen untergeordneten Komponenten vom Skript ignoriert werden. Verwenden Sie diese Eigenschaft zum Testen, Debuggen und Ändern eines Skripts. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none">- Ausgewählt. Das Skript ignoriert die Komponente.- Gelöscht. Das Skript wendet die Komponente an. Standardmäßig ist die Eigenschaft deaktiviert.
name	Eine beschreibende Kennzeichnung für die Komponente. Diese Kennzeichnung wird in der Protokolldatei sowie der Ereignisansicht angezeigt. Ermitteln Sie anhand der Eigenschaft name , welche Komponente das Ereignis verursacht hat.
remark	Ein benutzerdefinierter Kommentar, der den Zweck oder die Aktion der Komponente beschreibt.

RemoveTags

Der Transformer **RemoveTags** entfernt HTML-Tags aus dem Eingabetext. Er ersetzt die Tags an Positionen innerhalb des Textes durch einen Trennzeichenstring, zum Beispiel ein Leerzeichen. Am Anfang und am Ende des Textes fügt er keinen Trennzeichenstring ein. Aneinander grenzende Tags werden in ein einziges Trennzeichen umgewandelt.

Die nachstehende Tabelle beschreibt die Eigenschaften des Transformers **RemoveTags**:

Eigenschaft	Beschreibung
disabled	Legt fest, ob die Komponente und alle zugehörigen untergeordneten Komponenten vom Skript ignoriert werden. Verwenden Sie diese Eigenschaft zum Testen, Debuggen und Ändern eines Skripts. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none"> - Ausgewählt. Das Skript ignoriert die Komponente. - Gelöscht. Das Skript wendet die Komponente an. Standardmäßig ist die Eigenschaft deaktiviert.
name	Eine beschreibende Kennzeichnung für die Komponente. Diese Kennzeichnung wird in der Protokolldatei sowie der Ereignisansicht angezeigt. Ermitteln Sie anhand der Eigenschaft name , welche Komponente das Ereignis verursacht hat.
optional	Legt fest, ob ein Komponentenfehler den Ausfall der übergeordneten Komponente verursacht. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none"> - Ausgewählt. Ein Komponentenfehler führt nicht zum Ausfall der übergeordneten Komponente. - Gelöscht. Ein Komponentenfehler verursacht den Ausfall der übergeordneten Komponente. Standardwert ist "Gelöscht". Weitere Informationen über Komponentenfehler finden Sie in "Fehlerbehandlung" auf Seite 410 .
remark	Ein benutzerdefinierter Kommentar, der den Zweck oder die Aktion der Komponente beschreibt.
replace_with	Definiert den Separator-String. Standardwert ist " " (Leerzeichen).

Replace

Der Transformer **Replace** sucht und ersetzt Zeichenketten im Eingabetext. Wenn Sie die Eigenschaft **replace_with** leer lassen, wird der gefundene Text gelöscht.

Die nachstehende Tabelle beschreibt die Eigenschaften des Transformers **Replace**:

Eigenschaft	Beschreibung
disabled	Legt fest, ob die Komponente und alle zugehörigen untergeordneten Komponenten vom Skript ignoriert werden. Verwenden Sie diese Eigenschaft zum Testen, Debuggen und Ändern eines Skripts. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none"> - Ausgewählt. Das Skript ignoriert die Komponente. - Gelöscht. Das Skript wendet die Komponente an. Standardmäßig ist die Eigenschaft deaktiviert.
find_what	Definiert den gesuchten Text. Der Wert dieser Eigenschaft ist eine der folgenden Suchkomponenten: <ul style="list-style-type: none"> - NewlineSearch. Findet ein Zeilenvorschubzeichen. - PatternSearch. Findet Text, der mit einem regulären Ausdruck übereinstimmt. - SegmentSearch. Findet ein Segment von einem angegebenen öffnenden bis zu einem schließenden Referenzpunkt. - TextSearch. Findet eine angegebene Zeichenkette. Standard ist TextSearch. Weitere Informationen hierzu finden Sie im Abschnitt "Suchkomponenten: Referenz" auf Seite 248 .
name	Eine beschreibende Kennzeichnung für die Komponente. Diese Kennzeichnung wird in der Protokolldatei sowie der Ereignisansicht angezeigt. Ermitteln Sie anhand der Eigenschaft name , welche Komponente das Ereignis verursacht hat.

Eigenschaft	Beschreibung
occurrence	Gibt an, welche Instanzen ersetzt werden sollen: <code>all</code> , <code>first</code> oder <code>last</code> .
optional	Legt fest, ob ein Komponentenfehler den Ausfall der übergeordneten Komponente verursacht. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none"> - Ausgewählt. Ein Komponentenfehler führt nicht zum Ausfall der übergeordneten Komponente. - Gelöscht. Ein Komponentenfehler verursacht den Ausfall der übergeordneten Komponente. Standardwert ist "Gelöscht". Weitere Informationen über Komponentenfehler finden Sie in "Fehlerbehandlung" auf Seite 410 .
remark	Ein benutzerdefinierter Kommentar, der den Zweck oder die Aktion der Komponente beschreibt.
replace_with	Definiert den Ersatzstring.

Online-Beispiel

Ein Online-Beispiel für diesen Anker befindet sich in der Datei `samples\Projects\Transformers_Example\Transformers_Example.cmw`. Der zweite und der fünfte Content-Anker im Parser sind mit `Replace`-Transformern konfiguriert.

Resize

Der Transformer **Resize** passt den Eingabetext einer angegebenen Größe an. Je nach Bedarf füllt er ihn auf oder beschneidet ihn.

Die nachstehende Tabelle beschreibt die Eigenschaften des Transformers **Resize**:

Eigenschaft	Beschreibung
align	Definiert die Textausrichtung im angepassten String. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none"> - Links. Rechts auffüllen oder beschneiden. - Rechts. Links auffüllen oder beschneiden.
allow_smaller_size	Der ausgewählte Resize-Transformer passt den Eingabetext durch Auffüllen an eine angegebene Größe an, wenn die Zeichenfolge kleiner als die vom Parameter size definierte Größe ist. Wenn der Resize-Transformer nicht ausgewählt und die Eingabezeichenfolge kleiner als die angegebene Größe ist, schlägt der Transformer fehl.
name	Eine beschreibende Kennzeichnung für die Komponente. Diese Kennzeichnung wird in der Protokolldatei sowie der Ereignisansicht angezeigt. Ermitteln Sie anhand der Eigenschaft name , welche Komponente das Ereignis verursacht hat.
padding_character	Definiert das Auffüllungszeichen. Geben Sie das Zeichen ein oder wählen Sie einen Datenbehälter, der ein Zeichen enthält.
remark	Ein benutzerdefinierter Kommentar, der den Zweck oder die Aktion der Komponente beschreibt.
size	Definiert die Größe des Ausgabetexts. Geben Sie eine ganze Zahl ein oder wählen Sie einen Datenbehälter aus, der eine ganze Zahl enthält.

ReverseTransformer

Der Transformer **ReverseTransformer** kehrt den String um. Er wandelt beispielsweise 1234 in 4321 um.

Die nachstehende Tabelle beschreibt die Eigenschaften des Transformers **ReverseTransformer**:

Eigenschaft	Beschreibung
disabled	Legt fest, ob die Komponente und alle zugehörigen untergeordneten Komponenten vom Skript ignoriert werden. Verwenden Sie diese Eigenschaft zum Testen, Debuggen und Ändern eines Skripts. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none">- Ausgewählt. Das Skript ignoriert die Komponente.- Gelöscht. Das Skript wendet die Komponente an. Standardmäßig ist die Eigenschaft deaktiviert.
name	Eine beschreibende Kennzeichnung für die Komponente. Diese Kennzeichnung wird in der Protokolldatei sowie der Ereignisansicht angezeigt. Ermitteln Sie anhand der Eigenschaft name , welche Komponente das Ereignis verursacht hat.
optional	Legt fest, ob ein Komponentenfehler den Ausfall der übergeordneten Komponente verursacht. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none">- Ausgewählt. Ein Komponentenfehler führt nicht zum Ausfall der übergeordneten Komponente.- Gelöscht. Ein Komponentenfehler verursacht den Ausfall der übergeordneten Komponente. Standardwert ist "Gelöscht". Weitere Informationen über Komponentenfehler finden Sie in "Fehlerbehandlung" auf Seite 410 .
remark	Ein benutzerdefinierter Kommentar, der den Zweck oder die Aktion der Komponente beschreibt.

RtfProcessor

Der Transformer **RtfProcessor** normalisiert den RTF-Code. Er kann auch als Formatpräprozessor genutzt werden. Weitere Informationen hierzu finden Sie unter ["Formatpräprozessor-Komponenten: Referenz" auf Seite 190](#).

RtfToASCII

Der Transformer **RtfToASCII** wandelt eine RTF-Eingabe in einfachen Text um. Er entfernt RTF-Steuerungswörter aus dem Text.

Die nachstehende Tabelle beschreibt die Eigenschaften des Transformers **RtfToASCII**:

Eigenschaft	Beschreibung
disabled	Legt fest, ob die Komponente und alle zugehörigen untergeordneten Komponenten vom Skript ignoriert werden. Verwenden Sie diese Eigenschaft zum Testen, Debuggen und Ändern eines Skripts. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none">- Ausgewählt. Das Skript ignoriert die Komponente.- Gelöscht. Das Skript wendet die Komponente an. Standardmäßig ist die Eigenschaft deaktiviert.
name	Eine beschreibende Kennzeichnung für die Komponente. Diese Kennzeichnung wird in der Protokolldatei sowie der Ereignisansicht angezeigt. Ermitteln Sie anhand der Eigenschaft name , welche Komponente das Ereignis verursacht hat.
remark	Ein benutzerdefinierter Kommentar, der den Zweck oder die Aktion der Komponente beschreibt.

SubString

Der Transformer **SubString** gibt einen Teilstring der Eingabe zurück, die an angegebenen Positionen beginnt und endet.

In der folgenden Tabelle werden die Eigenschaften des Transformers **SubString** beschrieben:

Eigenschaft	Beschreibung
begin	Definiert die Startposition. Beim Wert „0“ entspricht der Start dem Anfang der Eingabe.
disabled	Legt fest, ob die Komponente und alle zugehörigen untergeordneten Komponenten vom Skript ignoriert werden. Verwenden Sie diese Eigenschaft zum Testen, Debuggen und Ändern eines Skripts. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none">- Ausgewählt. Das Skript ignoriert die Komponente.- Gelöscht. Das Skript wendet die Komponente an. Standardmäßig ist die Eigenschaft deaktiviert.
end	Definiert die Endposition.
name	Eine beschreibende Kennzeichnung für die Komponente. Diese Kennzeichnung wird in der Protokolldatei sowie der Ereignisansicht angezeigt. Ermitteln Sie anhand der Eigenschaft name , welche Komponente das Ereignis verursacht hat.
remark	Ein benutzerdefinierter Kommentar, der den Zweck oder die Aktion der Komponente beschreibt.

ToFloat

Der Transformer **ToFloat** wandelt eine Gleitkommazahl aus einem ASCII-String in die Binärdarstellung um. Die Umwandlung wird in der Ausgabecodierung mit der Ausgabe-Bytereihenfolge durchgeführt.

In der folgenden Tabelle werden die Eigenschaften des Transformers **ToFloat** beschrieben:

Eigenschaft	Beschreibung
disabled	Legt fest, ob die Komponente und alle zugehörigen untergeordneten Komponenten vom Skript ignoriert werden. Verwenden Sie diese Eigenschaft zum Testen, Debuggen und Ändern eines Skripts. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none">- Ausgewählt. Das Skript ignoriert die Komponente.- Gelöscht. Das Skript wendet die Komponente an. Standardmäßig ist die Eigenschaft deaktiviert.
name	Eine beschreibende Kennzeichnung für die Komponente. Diese Kennzeichnung wird in der Protokolldatei sowie der Ereignisansicht angezeigt. Ermitteln Sie anhand der Eigenschaft name , welche Komponente das Ereignis verursacht hat.
optional	Legt fest, ob ein Komponentenfehler den Ausfall der übergeordneten Komponente verursacht. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none">- Ausgewählt. Ein Komponentenfehler führt nicht zum Ausfall der übergeordneten Komponente.- Gelöscht. Ein Komponentenfehler verursacht den Ausfall der übergeordneten Komponente. Standardwert ist "Gelöscht". Weitere Informationen über Komponentenfehler finden Sie in "Fehlerbehandlung" auf Seite 410 .

Eigenschaft	Beschreibung
remark	Ein benutzerdefinierter Kommentar, der den Zweck oder die Aktion der Komponente beschreibt.
size	Legt die Größe der Ausgabenzahl fest. Die Eigenschaft size weist die folgenden Optionen auf: <ul style="list-style-type: none"> - single_precision_32_bit - double_precision_64_bit Die Standardoption lautet „single_precision_32_bit“.

Hinweis: Diese Komponente unterstützt keine UTF-16LE-Eingabecodierung.

TolInteger

Der Transformer **TolInteger** wandelt eine Zahl aus einer ASCII-Stringdarstellung in eine binäre Ganzzahl um. Beim Eingabestring kann es sich um einen dezimalen, oktalen oder hexadezimalen String handeln. Die Umwandlung wird in der Ausgabecodierung mit der Ausgabe-Bytereihenfolge durchgeführt.

In der folgenden Tabelle werden die Eigenschaften des Transformers **TolInteger** beschrieben:

Eigenschaft	Beschreibung
disabled	Legt fest, ob die Komponente und alle zugehörigen untergeordneten Komponenten vom Skript ignoriert werden. Verwenden Sie diese Eigenschaft zum Testen, Debuggen und Ändern eines Skripts. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none"> - Ausgewählt. Das Skript ignoriert die Komponente. - Gelöscht. Das Skript wendet die Komponente an. Standardmäßig ist die Eigenschaft deaktiviert.
from_base	Definiert die Basis der Eingabe. Die Eigenschaft to_base weist die folgenden Optionen auf: <ul style="list-style-type: none"> - decimal. Basis 10. - hexadecimal. Basis 16 mit Großbuchstaben A-F. - lowercase hexadecimal. Basis 16 mit Kleinbuchstaben a-f. - octal. Basis 8. Standardmäßig wird „decimal“ verwendet.
name	Eine beschreibende Kennzeichnung für die Komponente. Diese Kennzeichnung wird in der Protokolldatei sowie der Ereignisansicht angezeigt. Ermitteln Sie anhand der Eigenschaft name , welche Komponente das Ereignis verursacht hat.
optional	Legt fest, ob ein Komponentenfehler den Ausfall der übergeordneten Komponente verursacht. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none"> - Ausgewählt. Ein Komponentenfehler führt nicht zum Ausfall der übergeordneten Komponente. - Gelöscht. Ein Komponentenfehler verursacht den Ausfall der übergeordneten Komponente. Standardwert ist "Gelöscht". Weitere Informationen über Komponentenfehler finden Sie in "Fehlerbehandlung" auf Seite 410 .
remark	Ein benutzerdefinierter Kommentar, der den Zweck oder die Aktion der Komponente beschreibt.
signed	Legt fest, ob die Eingabezahl ein Vorzeichen enthält. Die Eigenschaft to_base weist die folgenden Optionen auf: <ul style="list-style-type: none"> - Ausgewählt. Die Ausgabeszahl enthält ein Vorzeichen. - Gelöscht. Die Ausgabeszahl enthält kein Vorzeichen. Die Standardoption lautet „Gelöscht“.
size	Definiert die Größe der Binärdarstellung in Byte. Unterstützt werden die Werte 1 bis 8.

Hinweis: Diese Komponente unterstützt keine UTF-16LE-Eingabecodierung.

ToPackDecimal

Der Transformer **ToPackDecimal** wandelt eine Zahl aus einer ASCII-Stringdarstellung in gepackte Dezimalzahlen um. Die Umwandlung wird in der Ausgabecodierung mit der Ausgabe-Bytereihenfolge durchgeführt.

In der folgenden Tabelle werden die Eigenschaften des Transformers **ToPackDecimal** beschrieben:

Eigenschaft	Beschreibung
disabled	Legt fest, ob die Komponente und alle zugehörigen untergeordneten Komponenten vom Skript ignoriert werden. Verwenden Sie diese Eigenschaft zum Testen, Debuggen und Ändern eines Skripts. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none">- Ausgewählt. Das Skript ignoriert die Komponente.- Gelöscht. Das Skript wendet die Komponente an. Standardmäßig ist die Eigenschaft deaktiviert.
name	Eine beschreibende Kennzeichnung für die Komponente. Diese Kennzeichnung wird in der Protokolldatei sowie der Ereignisansicht angezeigt. Ermitteln Sie anhand der Eigenschaft name , welche Komponente das Ereignis verursacht hat.
optional	Legt fest, ob ein Komponentenfehler den Ausfall der übergeordneten Komponente verursacht. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none">- Ausgewählt. Ein Komponentenfehler führt nicht zum Ausfall der übergeordneten Komponente.- Gelöscht. Ein Komponentenfehler verursacht den Ausfall der übergeordneten Komponente. Standardwert ist "Gelöscht". Weitere Informationen über Komponentenfehler finden Sie in "Fehlerbehandlung" auf Seite 410 .
remark	Ein benutzerdefinierter Kommentar, der den Zweck oder die Aktion der Komponente beschreibt.
unsigned	Legt fest, ob die gepackte Dezimalzahl ein Vorzeichen enthält. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none">- Ausgewählt. Die gepackte Dezimalzahl enthält kein Vorzeichen.- Gelöscht. Die gepackte Dezimalzahl enthält ein Vorzeichen. Die Standardoption lautet „Gelöscht“.

Hinweis: Diese Komponente unterstützt keine UTF-16LE-Eingabecodierung.

TransformationStartTime

Der Transformer **TransformationStartTime** gibt das Datum und die Uhrzeit des Starts der Umwandlung aus.

Er kopiert das Datum und die Uhrzeit aus der Variablen *VarSystem* und formatiert die Ausgabe gemäß den vorliegenden Angaben.

In der folgenden Tabelle werden die Eigenschaften des Transformers **TransformationStartTime** beschrieben:

Eigenschaft	Beschreibung
disabled	Legt fest, ob die Komponente und alle zugehörigen untergeordneten Komponenten vom Skript ignoriert werden. Verwenden Sie diese Eigenschaft zum Testen, Debuggen und Ändern eines Skripts. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none"> - Ausgewählt. Das Skript ignoriert die Komponente. - Gelöscht. Das Skript wendet die Komponente an. Standardmäßig ist die Eigenschaft deaktiviert.
format	Das Format von Datum und Uhrzeit. Geben Sie das Format ein oder wählen Sie einen Datenbehälter aus, der das Format enthält. Weitere Informationen zu den unterstützten Formaten finden Sie unter "DateFormatICU" auf Seite 270 .
name	Eine beschreibende Kennzeichnung für die Komponente. Diese Kennzeichnung wird in der Protokolldatei sowie der Ereignisansicht angezeigt. Ermitteln Sie anhand der Eigenschaft name , welche Komponente das Ereignis verursacht hat.
optional	Legt fest, ob ein Komponentenfehler den Ausfall der übergeordneten Komponente verursacht. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none"> - Ausgewählt. Ein Komponentenfehler führt nicht zum Ausfall der übergeordneten Komponente. - Gelöscht. Ein Komponentenfehler verursacht den Ausfall der übergeordneten Komponente. Standardwert ist "Gelöscht". Weitere Informationen über Komponentenfehler finden Sie in "Fehlerbehandlung" auf Seite 410 .
remark	Ein benutzerdefinierter Kommentar, der den Zweck oder die Aktion der Komponente beschreibt.

TransformByParser

Der Transformer **TransformByParser** führt für seinen Eingabetext einen Parser aus. Der Parser muss **FindReplaceAnchor**-Komponenten beinhalten, die zu ersetzende Segmente des Textes markieren. Im Anschluss an die Ausführung des Parsers führt der Transformer die Ersetzungen aus.

Die Ausgabe des Transformers ist der bearbeitete Text. Das Skript ignoriert sämtliche XML-Ausgaben, die vom Parser erzeugt werden.

In der folgenden Tabelle werden die Eigenschaften des Transformers **TransformByParser** beschrieben:

Eigenschaft	Beschreibung
disabled	Legt fest, ob die Komponente und alle zugehörigen untergeordneten Komponenten vom Skript ignoriert werden. Verwenden Sie diese Eigenschaft zum Testen, Debuggen und Ändern eines Skripts. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none"> - Ausgewählt. Das Skript ignoriert die Komponente. - Gelöscht. Das Skript wendet die Komponente an. Standardmäßig ist die Eigenschaft deaktiviert.
name	Eine beschreibende Kennzeichnung für die Komponente. Diese Kennzeichnung wird in der Protokolldatei sowie der Ereignisansicht angezeigt. Ermitteln Sie anhand der Eigenschaft name , welche Komponente das Ereignis verursacht hat.

Eigenschaft	Beschreibung
optional	Legt fest, ob ein Komponentenfehler den Ausfall der übergeordneten Komponente verursacht. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none"> - Ausgewählt. Ein Komponentenfehler führt nicht zum Ausfall der übergeordneten Komponente. - Gelöscht. Ein Komponentenfehler verursacht den Ausfall der übergeordneten Komponente. Standardwert ist "Gelöscht". Weitere Informationen über Komponentenfehler finden Sie in "Fehlerbehandlung" auf Seite 410 .
Parser	Definiert den Namen des Parsers.
remark	Ein benutzerdefinierter Kommentar, der den Zweck oder die Aktion der Komponente beschreibt.

Online-Beispiel

Ein Online-Beispiel finden Sie unter `<installation>\client\DT\samples\Projects\TransformByParser\TransformByParser.cmw`. In dem Beispiel wird mit Hilfe von `TransformByParser` jede Instanz der Zeichenfolge `~NL~` durch ein Wagenrücklauf- und ein Zeilenvorschubzeichen ersetzt.

Hinweis: Die Beispiele sind nur verfügbar, wenn Sie eine Client-Installation durchführen.

So führen Sie das `TransformByParser`-Beispiel aus:

1. Definieren Sie `MyTransformByParser` als Startkomponente.
2. Führen Sie den Transformer aus.
3. Wählen Sie an der Eingabeaufforderung die Quelldatei `Report.edi` aus.

Der Transformer speichert seine Ausgabe in der Datei `Results\Transformation of Report.edi`. Im Microsoft Editor können Sie die Ausgabe mit der Quelldatei vergleichen.

TransformByProcessor

Der Transformer **TransformByProcessor** führt für seine Eingabe einen Dokumentprozessor aus. Die Ausgabe des Transformers ist die Ausgabe des Dokumentprozessors. Weitere Informationen hierzu finden Sie unter ["Überblick über Dokumentprozessoren" auf Seite 163](#).

In der folgenden Tabelle werden die Eigenschaften des Transformers **TransformByProcessor** beschrieben:

Eigenschaft	Beschreibung
disabled	Legt fest, ob die Komponente und alle zugehörigen untergeordneten Komponenten vom Skript ignoriert werden. Verwenden Sie diese Eigenschaft zum Testen, Debuggen und Ändern eines Skripts. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none"> - Ausgewählt. Das Skript ignoriert die Komponente. - Gelöscht. Das Skript wendet die Komponente an. Standardmäßig ist die Eigenschaft deaktiviert.
name	Eine beschreibende Kennzeichnung für die Komponente. Diese Kennzeichnung wird in der Protokolldatei sowie der Ereignisansicht angezeigt. Ermitteln Sie anhand der Eigenschaft name , welche Komponente das Ereignis verursacht hat.

Eigenschaft	Beschreibung
optional	Legt fest, ob ein Komponentenfehler den Ausfall der übergeordneten Komponente verursacht. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none"> - Ausgewählt. Ein Komponentenfehler führt nicht zum Ausfall der übergeordneten Komponente. - Gelöscht. Ein Komponentenfehler verursacht den Ausfall der übergeordneten Komponente. Standardwert ist "Gelöscht". Weitere Informationen über Komponentenfehler finden Sie in "Fehlerbehandlung" auf Seite 410 .
processor	Definiert den Namen des Dokumentprozessors.
remark	Ein benutzerdefinierter Kommentar, der den Zweck oder die Aktion der Komponente beschreibt.

TransformByService

Der Transformer **TransformByService** führt für seine Eingabe einen Datenprozessor-Umwandlungsdienst aus. Die Ausgabe des Transformers ist die Ausgabe des Dienstes.

Wenn Sie mit dem Transformer zum Beispiel einen Parserdienst aufrufen, ist die Ausgabe des Transformers eine XML-Zeichenfolge.

Hinweis: Der Transformer **TransformByService** unterstützt Einzeleingabe-Dienste. Verwenden Sie ihn nicht mit Diensten, die über mehrfache Eingabeports verfügen.

In der folgenden Tabelle werden die Eigenschaften des Transformers **TransformByService** beschrieben:

Eigenschaft	Beschreibung
disabled	Legt fest, ob die Komponente und alle zugehörigen untergeordneten Komponenten vom Skript ignoriert werden. Verwenden Sie diese Eigenschaft zum Testen, Debuggen und Ändern eines Skripts. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none"> - Ausgewählt. Das Skript ignoriert die Komponente. - Gelöscht. Das Skript wendet die Komponente an. Standardmäßig ist die Eigenschaft deaktiviert.
disable_automatic_encoding	Legt fest, ob das Skript die im Dienst definierten Eingabe- und Ausgabecodierungen anwendet. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none"> - Ausgewählt. Das Skript ignoriert die im Dienst definierten Eingabe- und Ausgabecodierungen. - Gelöscht. Das Skript wendet die im Dienst definierten Eingabe- und Ausgabecodierungen an.
name	Eine beschreibende Kennzeichnung für die Komponente. Diese Kennzeichnung wird in der Protokolldatei sowie der Ereignisansicht angezeigt. Ermitteln Sie anhand der Eigenschaft name , welche Komponente das Ereignis verursacht hat.
optional	Legt fest, ob ein Komponentenfehler den Ausfall der übergeordneten Komponente verursacht. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none"> - Ausgewählt. Ein Komponentenfehler führt nicht zum Ausfall der übergeordneten Komponente. - Gelöscht. Ein Komponentenfehler verursacht den Ausfall der übergeordneten Komponente. Standardwert ist "Gelöscht". Weitere Informationen über Komponentenfehler finden Sie in "Fehlerbehandlung" auf Seite 410 .

Eigenschaft	Beschreibung
parameters	Definiert eine Liste mit Anfangswerten, die das Skript den im Dienst definierten Variablen zuweist. Geben Sie in jedem Element der Liste den Namen und den Wert einer Variable an.
remark	Ein benutzerdefinierter Kommentar, der den Zweck oder die Aktion der Komponente beschreibt.
service_name	Definiert den Namen des Dienstes, der für die Eingabe ausgeführt wird.

TransformerPipeline

Der Transformer **TransformerPipeline** wendet für seine Eingabe eine Folge geschachtelter Transformer an.

In der folgenden Tabelle werden die Eigenschaften des Transformers **TransformerPipeline** beschrieben:

Eigenschaft	Beschreibung
disabled	Legt fest, ob die Komponente und alle zugehörigen untergeordneten Komponenten vom Skript ignoriert werden. Verwenden Sie diese Eigenschaft zum Testen, Debuggen und Ändern eines Skripts. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none"> - Ausgewählt. Das Skript ignoriert die Komponente. - Gelöscht. Das Skript wendet die Komponente an. Standardmäßig ist die Eigenschaft deaktiviert.
name	Eine beschreibende Kennzeichnung für die Komponente. Diese Kennzeichnung wird in der Protokolldatei sowie der Ereignisansicht angezeigt. Ermitteln Sie anhand der Eigenschaft name , welche Komponente das Ereignis verursacht hat.
optional	Legt fest, ob ein Komponentenfehler den Ausfall der übergeordneten Komponente verursacht. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none"> - Ausgewählt. Ein Komponentenfehler führt nicht zum Ausfall der übergeordneten Komponente. - Gelöscht. Ein Komponentenfehler verursacht den Ausfall der übergeordneten Komponente. Standardwert ist "Gelöscht". Weitere Informationen über Komponentenfehler finden Sie in " Fehlerbehandlung " auf Seite 410.
remark	Ein benutzerdefinierter Kommentar, der den Zweck oder die Aktion der Komponente beschreibt.

XMLLookupTable

Die Komponente **XMLLookupTable** gibt eine XML-Datei an, die eine Lookup-Tabelle enthält. Die Tabelle entspricht dem Schema `lookupTableDefinition.xsddoc` im Unterverzeichnis `\doc` des Installationsverzeichnis. Die Tabelle wird vom Transformer **LookupTransformer** verwendet.

In der folgenden Tabelle werden die Eigenschaften der Komponente **XMLLookupTable** beschrieben:

Eigenschaft	Beschreibung
xml_file_name	Definiert den Pfad- und den Dateinamen der XML-Datei.

Das folgende XML-Dokument entspricht dem Schema:

```
<?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>
```

Wenn das optionale **matchCase**-Attribut auf **true** festgelegt ist, unterscheidet das **key**-Attribut zwischen Groß- und Kleinschreibung.

XSLTTransformer

Der Transformer **XSLTTransformer** wendet auf den XML-Eingabetext eine XSLT-Umwandlung an.

Beispielsweise können Sie mit einem Parser Daten aus einem XML-Dokument extrahieren. Von einem **Content**-Anker wird ein vollständiger, wohlgeformter Zweig des XML-Baums abgerufen. Der **Content**-Anker kann mit einem **XSLTTransformer** konfiguriert werden, der an dem Zweig eine XSLT-Umwandlung ausführt.

In der folgenden Tabelle werden die Eigenschaften des Transformers **XSLTTransformer** beschrieben:

Eigenschaft	Beschreibung
disabled	Legt fest, ob die Komponente und alle zugehörigen untergeordneten Komponenten vom Skript ignoriert werden. Verwenden Sie diese Eigenschaft zum Testen, Debuggen und Ändern eines Skripts. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none">- Ausgewählt. Das Skript ignoriert die Komponente.- Gelöscht. Das Skript wendet die Komponente an. Standardmäßig ist die Eigenschaft deaktiviert.
name	Eine beschreibende Kennzeichnung für die Komponente. Diese Kennzeichnung wird in der Protokolldatei sowie der Ereignisansicht angezeigt. Ermitteln Sie anhand der Eigenschaft name , welche Komponente das Ereignis verursacht hat.
optional	Legt fest, ob ein Komponentenfehler den Ausfall der übergeordneten Komponente verursacht. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none">- Ausgewählt. Ein Komponentenfehler führt nicht zum Ausfall der übergeordneten Komponente.- Gelöscht. Ein Komponentenfehler verursacht den Ausfall der übergeordneten Komponente. Standardwert ist "Gelöscht". Weitere Informationen über Komponentenfehler finden Sie in "Fehlerbehandlung" auf Seite 410 .
remark	Ein benutzerdefinierter Kommentar, der den Zweck oder die Aktion der Komponente beschreibt.
xslt_file	Definiert den Pfad- und den Dateinamen der XSLT-Datei.

KAPITEL 18

Aktionen

Dieses Kapitel umfasst die folgenden Themen:

- [Überblick über die Aktionen, 301](#)
- [Standardeigenschaften von Aktionen, 303](#)
- [Die Komponente Aktion: Referenz, 303](#)
- [Aktions-Teilkomponenten: Referenz, 338](#)

Überblick über die Aktionen

Aktionen sind Komponenten, die Vorgänge für Daten durchführen, die das Skript aus einem Quelldokument extrahiert hat. Unterstützt werden unter anderem die folgenden Aktionen:

- Mathematische Berechnungen
- Verketteten von Strings
- Senden von Formularen an einen Webserver
- Aktivieren eines sekundären Parsers, Serializers oder Mappers
- Abfragen einer Datenbank

Die Datenprozessor-Umwandlung bietet viele Aktionen und Sie können benutzerdefinierte Aktionen definieren.

Funktionsweise von Aktionen

Eine Aktion entnimmt ihre Eingabe den derzeit verfügbaren Datenbehältern. Eine Aktion kann mehrere Eingaben haben.

Falls die Aktion in einen Parser geschachtelt ist, sind die Datenbehälter verfügbar, die vom Parser erzeugt wurden. In einem Serializer sind die in der XML-Eingabe enthaltenen Datenbehälter sowie alle vom Serializer erzeugten Datenbehälter verfügbar. Bei einem Mapper können sich die Datenbehälter in der Eingabe oder der Ausgabe befinden.

Die Aktion führt an der Eingabe Operationen durch und erzeugt eine Ausgabe. Zahlreiche Aktionen können dahin gehend konfiguriert werden, dass sie ihre Ausgabe in Datenbehältern speichern.

Bei den meisten Aktionen müssen die in den Eingabe- und Ausgabedatenbehältern enthaltenen Daten einfache Datentypen sein. Sie dürfen also keine geschachtelten Elemente enthalten. Einige Aktionen arbeiten mit Datenbehältern, die geschachtelte Elemente enthalten, mit Mehrfachinstanz-Datenbehältern oder mit anderen Sondertypen.

Eine Aktion kann weitere Auswirkungen haben. Sie kann zum Beispiel in eine Datei schreiben, eine Datenbank aktualisieren oder Daten an eine externe Anwendung übertragen.

Aktionen und Transformer im Vergleich

Einige Aktionen führen Operationen aus, die denen der Transformer gleichen, zum Beispiel die Bearbeitung einer Zeichenkette oder das Abfragen einer Datenbank. Allerdings unterscheiden sich Aktionen und Transformer in wichtigen Aspekten.

In der folgenden Tabelle sind die Unterschiede zusammengefasst.

Betrieb	Transformer	Aktionen
Eingabe	Die Eingabe eines Transformers ist ein einzelner String.	Die Eingabe wird durch die Aktion implementiert. Eine Aktion kann mehrere Eingaben haben. Die Eingaben können Datenbehälter sein.
Ausgabe	Die Ausgabe eines Transformers ist ein String.	Die Ausgabe wird durch die Aktion implementiert. Eine Aktion kann zum Beispiel Ausgabedatenbehälter erzeugen.
Nebenwirkungen	Ein Transformer hat keine Nebenwirkungen. Er bearbeitet lediglich den Eingabestring.	Eine Aktion kann Nebenwirkungen haben, zum Beispiel die Aktualisierung einer Datenbank.

Definieren von Aktionen

Bearbeiten Sie das Skript, um eine Aktion zu definieren. Sie können die Aktionen in die Zeile **contains** von Komponenten wie **Parser**, **Serializer**, **Mapper**, **Group** und **RepeatingGroup** einfügen. Im Prinzip können Sie die Aktionen an denselben Stellen einfügen wie Anker, Serialisierungsanker oder Mapper-Anker.

Die Aktionen werden zugleich mit den Ankern ausgeführt, die an derselben Stelle angegeben sind. In einem Parser können Sie die Eigenschaft **phase** einer Aktion festlegen. Diese legt fest, ob die Aktion in der Anfangs-, Haupt- oder Endphase des Parser-Prozesses ausgeführt wird. Weitere Informationen hierzu finden Sie unter ["Suchphasen" auf Seite 211](#).

Standardeigenschaften von Aktionen

In der folgenden Tabelle werden die Standardeigenschaften von Aktionen beschrieben:

Eigenschaft	Beschreibung
disabled	Legt fest, ob die Komponente und alle zugehörigen untergeordneten Komponenten vom Skript ignoriert werden. Verwenden Sie diese Eigenschaft zum Testen, Debuggen und Ändern eines Skripts. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none">- Ausgewählt. Das Skript ignoriert die Komponente.- Gelöscht. Das Skript wendet die Komponente an. Standardmäßig ist die Eigenschaft deaktiviert.
name	Eine beschreibende Kennzeichnung für die Komponente. Diese Kennzeichnung wird in der Protokolldatei sowie der Ereignisansicht angezeigt. Ermitteln Sie anhand der Eigenschaft name , welche Komponente das Ereignis verursacht hat.
on_fail	Die Aktion, die beim Ausfall der Komponente durchgeführt wird. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none">- Gelöscht. Es wird keine Aktion ausgeführt.- CustomLog. Es wird in das Benutzerprotokoll geschrieben.- LogError. Eine Fehlermeldung wird in das Engine-Protokoll geschrieben.- LogInfo. Eine Informationsmeldung wird in das Engine-Protokoll geschrieben.- LogWarning. Eine Warnmeldung wird in das Engine-Protokoll geschrieben.- NotifyFailure. Eine Mitteilung wird gesendet. Die Standardoption lautet „Gelöscht“. Weitere Informationen über die Behandlung von Komponentenfehlern finden Sie in "Fehlerbehandlung" auf Seite 410 .
optional	Legt fest, ob ein Komponentenfehler den Ausfall der übergeordneten Komponente verursacht. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none">- Ausgewählt. Ein Komponentenfehler führt nicht zum Ausfall der übergeordneten Komponente.- Gelöscht. Ein Komponentenfehler verursacht den Ausfall der übergeordneten Komponente. Standardwert ist "Gelöscht". Weitere Informationen über Komponentenfehler finden Sie in "Fehlerbehandlung" auf Seite 410 .
phase	Legt den Zeitpunkt fest, an dem das Skript die Komponente verarbeitet. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none">- Anfangsphase. Das Skript verarbeitet die Komponente während der Anfangsphase.- Hauptphase. Das Skript verarbeitet die Komponente während der Hauptphase.- Endphase. Das Skript verarbeitet die Komponente während der Endphase. Weitere Informationen hierzu finden Sie unter "So sucht der Parser Anker" auf Seite 211 .
remark	Ein benutzerdefinierter Kommentar, der den Zweck oder die Aktion der Komponente beschreibt.

Die Komponente Aktion: Referenz

Aktionen führen Vorgänge im System durch, beispielsweise Herunterladen einer Datei von einem entfernten Standort oder Validieren eines Werts.

AddEventAction

Die Aktion **AddEventAction** fügt dem Ereignisprotokoll eine Meldung hinzu.

Die nachstehende Tabelle beschreibt die Eigenschaften der Aktion **AddEventAction**:

Eigenschaft	Beschreibung
disabled	Legt fest, ob die Komponente und alle zugehörigen untergeordneten Komponenten vom Skript ignoriert werden. Verwenden Sie diese Eigenschaft zum Testen, Debuggen und Ändern eines Skripts. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none"> - Ausgewählt. Das Skript ignoriert die Komponente. - Gelöscht. Das Skript wendet die Komponente an. Standardmäßig ist die Eigenschaft deaktiviert.
message	Definiert den Nachrichten-String.
name	Eine beschreibende Kennzeichnung für die Komponente. Diese Kennzeichnung wird in der Protokolldatei sowie der Ereignisansicht angezeigt. Ermitteln Sie anhand der Eigenschaft name , welche Komponente das Ereignis verursacht hat.
on_fail	Die Aktion, die beim Ausfall der Komponente durchgeführt wird. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none"> - Gelöscht. Es wird keine Aktion ausgeführt. - CustomLog. Es wird in das Benutzerprotokoll geschrieben. - LogError. Eine Fehlermeldung wird in das Engine-Protokoll geschrieben. - LogInfo. Eine Informationsmeldung wird in das Engine-Protokoll geschrieben. - LogWarning. Eine Warnmeldung wird in das Engine-Protokoll geschrieben. - NotifyFailure. Eine Mitteilung wird gesendet. Die Standardoption lautet „Gelöscht“. Weitere Informationen über die Behandlung von Komponentenfehlern finden Sie in "Fehlerbehandlung" auf Seite 410 .
phase	Legt den Zeitpunkt fest, an dem das Skript die Komponente verarbeitet. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none"> - Anfangsphase. Das Skript verarbeitet die Komponente während der Anfangsphase. - Hauptphase. Das Skript verarbeitet die Komponente während der Hauptphase. - Endphase. Das Skript verarbeitet die Komponente während der Endphase. Weitere Informationen hierzu finden Sie unter "So sucht der Parser Anker" auf Seite 211 . Standardwert ist "Hauptphase".
remark	Ein benutzerdefinierter Kommentar, der den Zweck oder die Aktion der Komponente beschreibt.
severity	Legt den Schweregrad der Nachricht fest. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none"> - Benachrichtigung - Warnung - Fehler - schwerwiegender Fehler Standardwert ist "Benachrichtigung".

AggregateValues

Die Aktion **AggregateValues** führt eine Berechnung mit einem Aggregat eines mehrfach auftretenden Datenbehälters durch.

Die nachstehende Tabelle beschreibt die Eigenschaften der Aktion **AggregateValues**:

Eigenschaft	Beschreibung
aggregation_function	Bestimmt die Funktion, die mit dem Aggregat ausgeführt werden soll. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none"> - AllEqual. Gibt <code>True</code> zurück, wenn die Werte alle gleich sind, bzw. <code>False</code>, wenn sie nicht gleich sind. - Zählwert. Gibt die Anzahl der Vorkommen eines Datenbehälters zurück. - Join. Gibt eine Liste aller Werte zurück, getrennt durch das in der Eigenschaft separator angegebene Trennzeichen. - Summe. Gibt die Summe der Werte zurück.
AllEqual	Definiert eine Option unter der Eigenschaft aggregation_function .
Count	Definiert eine Option unter der Eigenschaft aggregation_function .
data_holder	Definiert den Datenbehälter, der die Ausgabe speichert.
disabled	Legt fest, ob die Komponente und alle zugehörigen untergeordneten Komponenten vom Skript ignoriert werden. Verwenden Sie diese Eigenschaft zum Testen, Debuggen und Ändern eines Skripts. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none"> - Ausgewählt. Das Skript ignoriert die Komponente. - Gelöscht. Das Skript wendet die Komponente an. Standardmäßig ist die Eigenschaft deaktiviert.
include_empty_values	Bestimmt, ob das Aggregat die Vorkommen ohne Daten einbezieht. <ul style="list-style-type: none"> - Ausgewählt. Die Aktion bezieht leere Vorkommen ein. - Gelöscht. Die Aktion ignoriert leere Vorkommen. Standardwert ist "Ausgewählt".
Join	Definiert eine Option unter der Eigenschaft aggregation_function .
name	Eine beschreibende Kennzeichnung für die Komponente. Diese Kennzeichnung wird in der Protokolldatei sowie der Ereignisansicht angezeigt. Ermitteln Sie anhand der Eigenschaft name , welche Komponente das Ereignis verursacht hat.
on_fail	Die Aktion, die beim Ausfall der Komponente durchgeführt wird. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none"> - Gelöscht. Es wird keine Aktion ausgeführt. - CustomLog. Es wird in das Benutzerprotokoll geschrieben. - LogError. Eine Fehlermeldung wird in das Engine-Protokoll geschrieben. - LogInfo. Eine Informationsmeldung wird in das Engine-Protokoll geschrieben. - LogWarning. Eine Warnmeldung wird in das Engine-Protokoll geschrieben. - NotifyFailure. Eine Mitteilung wird gesendet. Die Standardoption lautet „Gelöscht“. Weitere Informationen über die Behandlung von Komponentenfehlern finden Sie in "Fehlerbehandlung" auf Seite 410 .
optional	Legt fest, ob ein Komponentenfehler den Ausfall der übergeordneten Komponente verursacht. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none"> - Ausgewählt. Ein Komponentenfehler führt nicht zum Ausfall der übergeordneten Komponente. - Gelöscht. Ein Komponentenfehler verursacht den Ausfall der übergeordneten Komponente. Standardwert ist "Gelöscht". Weitere Informationen über Komponentenfehler finden Sie in "Fehlerbehandlung" auf Seite 410 .

Eigenschaft	Beschreibung
phase	Legt den Zeitpunkt fest, an dem das Skript die Komponente verarbeitet. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none"> - Anfangsphase. Das Skript verarbeitet die Komponente während der Anfangsphase. - Hauptphase. Das Skript verarbeitet die Komponente während der Hauptphase. - Endphase. Das Skript verarbeitet die Komponente während der Endphase. Weitere Informationen hierzu finden Sie unter "So sucht der Parser Anker" auf Seite 211 . Standardwert ist "Hauptphase".
remark	Ein benutzerdefinierter Kommentar, der den Zweck oder die Aktion der Komponente beschreibt.
root_element	Bestimmt den Datenbehälter auf der Stammebene des XML-Zweigs, der den mehrfach vorkommenden Datenbehälter enthält. Das root_element kann ein- oder mehrmals vorkommen.
sub_element	Bestimmt den mehrmals vorkommenden Datenbehälter, für den die Aktion ein Aggregat berechnet. Wenn sub_element nicht zugewiesen ist, berechnet die Aktion das Aggregat von root_element .
Sum	Definiert eine Option unter der Eigenschaft aggregation_function .

Je nach dem **root_element**, das Sie konfiguriert haben, kann die Aktion Vorkommen auf verschiedenen Verzweigungsebenen aggregieren. Ein XML-Dokument hat zum Beispiel die Struktur:

```
<Company>
  <Division name="America">
    <Employee>...<Employee>
    <Employee>...<Employee>
    <Employee>...<Employee>
  </Division>
  <Division name="Europe">
    <Employee>...<Employee>
    <Employee>...<Employee>
  </Division>
</Company>
```

Wenn das **root_element** *Firma* ist und Sie konfigurieren die Aktion so, dass sie die Vorkommen von *Mitarbeiter* zählt, zählt die Aktion alle *Mitarbeiter*-Elemente, die *Firma* untergeordnet sind. Die Aktion gibt 5 zurück.

Wenn das **root_element** *Abteilung* ist, zählt die Aktion die Anzahl der Vorkommen von *Mitarbeiter* in der *Abteilung*, die die Umwandlung derzeit verarbeitet. Wenn die Aktion *Amerika* verarbeitet, gibt sie 3 zurück. Wenn die Aktion *Europa* verarbeitet, gibt sie 2 zurück.

AppendListItems

Die Aktion **AppendListItems** verkettet die Zeichenketten in einem Mehrfachinstanz-Datenbehälter.

Die nachstehende Tabelle beschreibt die Eigenschaften des Validators **AppendListItems**:

Eigenschaft	Beschreibung
disabled	Legt fest, ob die Komponente und alle zugehörigen untergeordneten Komponenten vom Skript ignoriert werden. Verwenden Sie diese Eigenschaft zum Testen, Debuggen und Ändern eines Skripts. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none"> - Ausgewählt. Das Skript ignoriert die Komponente. - Gelöscht. Das Skript wendet die Komponente an. Standardmäßig ist die Eigenschaft deaktiviert.
input	Legt den Datenbehälter mit mehrfachen Vorkommen für die Eingabe fest. Der Datenbehälter muss über einfache Datentypen verfügen.
name	Eine beschreibende Kennzeichnung für die Komponente. Diese Kennzeichnung wird in der Protokolldatei sowie der Ereignisansicht angezeigt. Ermitteln Sie anhand der Eigenschaft name , welche Komponente das Ereignis verursacht hat.
on_fail	Die Aktion, die beim Ausfall der Komponente durchgeführt wird. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none"> - Gelöscht. Es wird keine Aktion ausgeführt. - CustomLog. Es wird in das Benutzerprotokoll geschrieben. - LogError. Eine Fehlermeldung wird in das Engine-Protokoll geschrieben. - LogInfo. Eine Informationsmeldung wird in das Engine-Protokoll geschrieben. - LogWarning. Eine Warnmeldung wird in das Engine-Protokoll geschrieben. - NotifyFailure. Eine Mitteilung wird gesendet. Die Standardoption lautet „Gelöscht“. Weitere Informationen über die Behandlung von Komponentenfehlern finden Sie in "Fehlerbehandlung" auf Seite 410 .
optional	Legt fest, ob ein Komponentenfehler den Ausfall der übergeordneten Komponente verursacht. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none"> - Ausgewählt. Ein Komponentenfehler führt nicht zum Ausfall der übergeordneten Komponente. - Gelöscht. Ein Komponentenfehler verursacht den Ausfall der übergeordneten Komponente. Standardwert ist "Gelöscht". Weitere Informationen über Komponentenfehler finden Sie in "Fehlerbehandlung" auf Seite 410 .
output	Legt den Datenbehälter fest, der die Ausgabe speichert. Der Datenbehälter muss über einfache Datentypen verfügen.
Phase	Legt den Zeitpunkt fest, an dem das Skript die Komponente verarbeitet. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none"> - Anfangsphase. Das Skript verarbeitet die Komponente während der Anfangsphase. - Hauptphase. Das Skript verarbeitet die Komponente während der Hauptphase. - Endphase. Das Skript verarbeitet die Komponente während der Endphase. Weitere Informationen hierzu finden Sie unter "So sucht der Parser Anker" auf Seite 211 . Standardwert ist "Hauptphase".
remark	Ein benutzerdefinierter Kommentar, der den Zweck oder die Aktion der Komponente beschreibt.

Informationen über die Vorbereitung der Eingabe für diese Aktion finden Sie im Abschnitt ["Zuordnen zu mehrfach vorkommenden Datenbehältern" auf Seite 207](#).

Beispiel

Ein Quelldokument enthält den folgenden durch Leerzeichen separierten Text.

H E L L O

Beim Parsen des Dokuments sollen die Leerzeichen entfernt werden. Das Ergebnis soll in einem XML-Element mit dem Namen `Greeting` gespeichert werden.

Erstellen Sie eine mehrfach vorkommende Variable mit der Bezeichnung `VarLetter`. Erstellen Sie mehrere `Content`-Anker, die die einzelnen Buchstaben abrufen und sie in Instanzen von `VarLetter` speichern.

Anschließend verketteten Sie die Instanzen von `VarLetter` mit Hilfe der Aktion **AppendListItems** und speichern das Ergebnis im Element `Greeting`. Das Ergebnis sieht folgendermaßen aus:

```
<Greeting>HELLO</Greeting>
```

Online-Beispiel

Ein Online-Beispiel zu dieser Aktion befindet sich im Projekt `samples\Projects\AppendListItems\AppendListItems.cmw`. In dem Beispiel werden mit Hilfe einer `RepeatingGroup` Werte in einer Mehrfachinstanz-Variablen gespeichert. Anschließend werden die Werte mit der Aktion `AppendListItems` verkettet.

AppendValues

Die Aktion **AppendValues** verkettet Zeichenketten.

Die nachstehende Tabelle beschreibt die Eigenschaften des Validators **AppendValues**:

Eigenschaft	Beschreibung
disabled	Legt fest, ob die Komponente und alle zugehörigen untergeordneten Komponenten vom Skript ignoriert werden. Verwenden Sie diese Eigenschaft zum Testen, Debuggen und Ändern eines Skripts. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none">- Ausgewählt. Das Skript ignoriert die Komponente.- Gelöscht. Das Skript wendet die Komponente an. Standardmäßig ist die Eigenschaft deaktiviert.
input	Legt eine Liste von Datenbehältern mit den Werten fest, die angehängt werden sollen. Die Datenbehälter müssen über einfache Datentypen verfügen.
name	Eine beschreibende Kennzeichnung für die Komponente. Diese Kennzeichnung wird in der Protokolldatei sowie der Ereignisansicht angezeigt. Ermitteln Sie anhand der Eigenschaft name , welche Komponente das Ereignis verursacht hat.
on_fail	Die Aktion, die beim Ausfall der Komponente durchgeführt wird. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none">- Gelöscht. Es wird keine Aktion ausgeführt.- CustomLog. Es wird in das Benutzerprotokoll geschrieben.- LogError. Eine Fehlermeldung wird in das Engine-Protokoll geschrieben.- LogInfo. Eine Informationsmeldung wird in das Engine-Protokoll geschrieben.- LogWarning. Eine Warnmeldung wird in das Engine-Protokoll geschrieben.- NotifyFailure. Eine Mitteilung wird gesendet. Die Standardoption lautet „Gelöscht“. Weitere Informationen über die Behandlung von Komponentenfehlern finden Sie in "Fehlerbehandlung" auf Seite 410 .
optional	Legt fest, ob ein Komponentenfehler den Ausfall der übergeordneten Komponente verursacht. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none">- Ausgewählt. Ein Komponentenfehler führt nicht zum Ausfall der übergeordneten Komponente.- Gelöscht. Ein Komponentenfehler verursacht den Ausfall der übergeordneten Komponente. Standardwert ist "Gelöscht". Weitere Informationen über Komponentenfehler finden Sie in "Fehlerbehandlung" auf Seite 410 .

Eigenschaft	Beschreibung
output	Legt den Datenbehälter fest, der die Ausgabe speichert. Der Datenbehälter muss über einfache Datentypen verfügen.
Phase	Legt den Zeitpunkt fest, an dem das Skript die Komponente verarbeitet. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none"> - Anfangsphase. Das Skript verarbeitet die Komponente während der Anfangsphase. - Hauptphase. Das Skript verarbeitet die Komponente während der Hauptphase. - Endphase. Das Skript verarbeitet die Komponente während der Endphase. Weitere Informationen hierzu finden Sie unter "So sucht der Parser Anker" auf Seite 211 . Standardwert ist "Hauptphase".
remark	Ein benutzerdefinierter Kommentar, der den Zweck oder die Aktion der Komponente beschreibt.
skip_unfound_values	Legt fest, ob die Aktion fortgesetzt wird, wenn einer der Eingabe-Datenbehälter fehlt. <ul style="list-style-type: none"> - Ausgewählt. Die Aktion wird fortgesetzt. - Gelöscht. Die Aktion schlägt fehl. Standardwert ist "Ausgewählt".

Beispiel

Ein Parser hat den folgenden XML-Code erzeugt.

```
<Name>
  <First>Ron</First>
  <Last>Lehrer</Last>
</Name>
```

Sie können eine **AppendValues**-Aktion konfigurieren, die die folgende Ausgabe erzeugt:

```
<FullName>Ron Lehrer</FullName>
```

CalculateValue

Berechnet numerische Werte oder verkettet Stringwerte.

Verwenden Sie die folgenden Operatoren zwischen den Parametern, um numerische Werte zu berechnen:

- +
- -
- *
- /

Zur Verdeutlichung numerischer Ausdrücke können Sie Klammern verwenden. Sie können Variablen der folgenden Datentypen verwenden:

- xs:anyType
- xs:anySimpleType
- numerische Datentypen
- String-Datentypen

Sind alle Parameter numerische Datentypen oder numerische Strings, führt CalculateValue eine mathematische Berechnung durch. Ergebnisse mit Nicht-Ganzzahlwerten werden auf 14 Dezimalstellen gerundet.

Verwenden Sie zur Verkettung von Strings den Pluszeichen (+)-Operator zwischen Parametern und Strings.

Die nachstehende Tabelle beschreibt die Eigenschaften der Aktion **CalculateValue**:

Eigenschaft	Beschreibung
disabled	Legt fest, ob die Komponente und alle zugehörigen untergeordneten Komponenten vom Skript ignoriert werden. Verwenden Sie diese Eigenschaft zum Testen, Debuggen und Ändern eines Skripts. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none"> - Ausgewählt. Das Skript ignoriert die Komponente. - Gelöscht. Das Skript wendet die Komponente an. Standardmäßig ist die Eigenschaft deaktiviert.
expression	Definiert einen JavaScript-Ausdruck. Verwenden Sie zur Darstellung eines Eingabeparameters ein Dollarzeichen (\$) mit nachfolgender Ganzzahl. Zur Darstellung eines Strings setzen Sie diesen in einfache Anführungszeichen.
failure_action	Legt das Verhalten im Fall eines Fehlers fest. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none"> - Ignorieren. Die Umwandlung wird fortgesetzt. - HaltExecution. Die Umwandlung wird angehalten. Standardwert ist "Ignorieren".
name	Eine beschreibende Kennzeichnung für die Komponente. Diese Kennzeichnung wird in der Protokolldatei sowie der Ereignisansicht angezeigt. Ermitteln Sie anhand der Eigenschaft name , welche Komponente das Ereignis verursacht hat.
on_fail	Die Aktion, die beim Ausfall der Komponente durchgeführt wird. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none"> - Gelöscht. Es wird keine Aktion ausgeführt. - CustomLog. Es wird in das Benutzerprotokoll geschrieben. - LogError. Eine Fehlermeldung wird in das Engine-Protokoll geschrieben. - LogInfo. Eine Informationsmeldung wird in das Engine-Protokoll geschrieben. - LogWarning. Eine Warnmeldung wird in das Engine-Protokoll geschrieben. - NotifyFailure. Eine Mitteilung wird gesendet. Die Standardoption lautet „Gelöscht“. Weitere Informationen über die Behandlung von Komponentenfehlern finden Sie in "Fehlerbehandlung" auf Seite 410 .
optional	Legt fest, ob ein Komponentenfehler den Ausfall der übergeordneten Komponente verursacht. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none"> - Ausgewählt. Ein Komponentenfehler führt nicht zum Ausfall der übergeordneten Komponente. - Gelöscht. Ein Komponentenfehler verursacht den Ausfall der übergeordneten Komponente. Standardwert ist "Gelöscht". Weitere Informationen über Komponentenfehler finden Sie in "Fehlerbehandlung" auf Seite 410 .
params	Definiert eine Liste mit Datenbehältern, die die Eingabeparameter enthalten.
phase	Legt den Zeitpunkt fest, an dem das Skript die Komponente verarbeitet. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none"> - Anfangsphase. Das Skript verarbeitet die Komponente während der Anfangsphase. - Hauptphase. Das Skript verarbeitet die Komponente während der Hauptphase. - Endphase. Das Skript verarbeitet die Komponente während der Endphase. Weitere Informationen hierzu finden Sie unter "So sucht der Parser Anker" auf Seite 211 . Standardwert ist "Hauptphase".
remark	Ein benutzerdefinierter Kommentar, der den Zweck oder die Aktion der Komponente beschreibt.
result	Legt einen Datenbehälter fest, der die Ausgabe speichert.

Hinweis: Weitere Informationen zur JavaScript-Syntax, die die Datenprozessor-Umwandlung unterstützt, finden Sie in ["EnsureCondition" auf Seite 318](#). Weiteres zur Genauigkeit der Werte `xs:decimal` und `xs:float` erfahren Sie im Abschnitt ["Genauigkeit numerischer Daten" auf Seite 195](#).

Beispiel

Ein Parser hat den folgenden XML-Code erzeugt.

```
<ItemOrdered>
  <Name>Gizmo</Name>
  <Quantity>100</Quantity>
  <Price>25</Price>
</ItemOrdered>
```

Mit Hilfe der Aktion **CalculateValue** können Sie die folgende Ausgabe erzeugen:

```
<ItemOrdered>
  <Name>Gizmo</Name>
  <Quantity>100</Quantity>
  <Price>25</Price>
  <Total>2500</Total>
</ItemOrdered>
```

Definieren Sie die Elemente `Name` und `Quantity` als Eingabeparameter. Geben Sie den JavaScript-Ausdruck `$1 * $2` an, und lassen Sie das Ergebnis im Element `Total` speichern.

Online-Beispiel

Ein Online-Beispiel zu dieser Aktion befindet sich im Projekt `samples\Projects\CalculateValue\CalculateValue.cmw`. In dem Beispiel werden aus dem Quelldokument drei Zahlen abgerufen und als Variablen gespeichert. Mit Hilfe der Aktion `CalculateValue` wird anschließend eine mathematische Funktion der Zahlen berechnet.

CombineValues

Die Aktion **CombineValues** verkettet Strings.

Die Eingabe ist eine Liste von Datenbehältern und Variablen. Die Ausgabe ist ein Mehrfachinstanz-Datenbehälter.

Ist die Eingabe ein mehrfach vorkommender Datenbehälter, so generiert die **CombineValues**-Aktion Iteration für jede Instanz des Datenbehälters. Bei jeder Iteration kombiniert die **CombineValues**-Aktion alle Eingabe-Datenbehälter und schreibt die Ausgabe an eine Instanz des Ausgabe-Datenbehälters.

Die nachstehende Tabelle beschreibt die Eigenschaften der Aktion **CombineValues**:

Eigenschaft	Beschreibung
disabled	Legt fest, ob die Komponente und alle zugehörigen untergeordneten Komponenten vom Skript ignoriert werden. Verwenden Sie diese Eigenschaft zum Testen, Debuggen und Ändern eines Skripts. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none">- Ausgewählt. Das Skript ignoriert die Komponente.- Gelöscht. Das Skript wendet die Komponente an. Standardmäßig ist die Eigenschaft deaktiviert.
input	Definiert eine Liste mit Datenbehältern für die Eingabe. Die Datenbehälter müssen über einfache Datentypen verfügen.
name	Eine beschreibende Kennzeichnung für die Komponente. Diese Kennzeichnung wird in der Protokolldatei sowie der Ereignisansicht angezeigt. Ermitteln Sie anhand der Eigenschaft name , welche Komponente das Ereignis verursacht hat.

Eigenschaft	Beschreibung
on_fail	<p>Die Aktion, die beim Ausfall der Komponente durchgeführt wird. Sie können eine der folgenden Optionen auswählen:</p> <ul style="list-style-type: none"> - Gelöscht. Es wird keine Aktion ausgeführt. - CustomLog. Es wird in das Benutzerprotokoll geschrieben. - LogError. Eine Fehlermeldung wird in das Engine-Protokoll geschrieben. - LogInfo. Eine Informationsmeldung wird in das Engine-Protokoll geschrieben. - LogWarning. Eine Warnmeldung wird in das Engine-Protokoll geschrieben. - NotifyFailure. Eine Mitteilung wird gesendet. <p>Die Standardoption lautet „Gelöscht“. Weitere Informationen über die Behandlung von Komponentenfehlern finden Sie in "Fehlerbehandlung" auf Seite 410.</p>
optional	<p>Legt fest, ob ein Komponentenfehler den Ausfall der übergeordneten Komponente verursacht. Sie können eine der folgenden Optionen auswählen:</p> <ul style="list-style-type: none"> - Ausgewählt. Ein Komponentenfehler führt nicht zum Ausfall der übergeordneten Komponente. - Gelöscht. Ein Komponentenfehler verursacht den Ausfall der übergeordneten Komponente. <p>Standardwert ist "Gelöscht". Weitere Informationen über Komponentenfehler finden Sie in "Fehlerbehandlung" auf Seite 410.</p>
output	<p>Legt den mehrfach vorkommenden Datenbehälter fest, wo die Aktion die Ausgabe speichert. Der Datenbehälter muss über einfache Datentypen verfügen.</p>
phase	<p>Legt den Zeitpunkt fest, an dem das Skript die Komponente verarbeitet. Sie können eine der folgenden Optionen auswählen:</p> <ul style="list-style-type: none"> - Anfangsphase. Das Skript verarbeitet die Komponente während der Anfangsphase. - Hauptphase. Das Skript verarbeitet die Komponente während der Hauptphase. - Endphase. Das Skript verarbeitet die Komponente während der Endphase. <p>Weitere Informationen hierzu finden Sie unter "So sucht der Parser Anker" auf Seite 211. Standardwert ist "Hauptphase".</p>
remark	<p>Ein benutzerdefinierter Kommentar, der den Zweck oder die Aktion der Komponente beschreibt.</p>

Beispiel

In einer Mehrfachinstanz-Variablen namens *VarDay* ist die Liste `Monday, Tuesday` gespeichert. In einer Mehrfachinstanz-Variablen namens *VarTime* sind die Werte `morning, afternoon` gespeichert. In einer Einzelinstanz-Variablen namens *VarSpace* ist ein Leerzeichen gespeichert.

Nun führen Sie **CombineValues** an *VarDay*, *VarSpace* und *VarTime* aus, mit einem Ausgabedatenbehälter namens **DayTime**. Die Ausgabe sieht folgendermaßen aus:

```
<DayTime>Monday morning</DayTime>
<DayTime>Monday afternoon</DayTime>
<DayTime>Tuesday morning</DayTime>
<DayTime>Tuesday afternoon</DayTime>
```

Online-Beispiel

Ein Online-Beispiel zu dieser Aktion befindet sich im Projekt `samples\Projects\CombineValues\CombineValues.cmw`. In dem Beispiel werden aus dem Quelldokument Listen mit Tagen, Monaten und Jahren abgerufen. Mit Hilfe der Aktion `CombineValues` werden aus den Listen sämtliche möglichen Daten erzeugt.

CreateList

Die Aktion **CreateList** fügt Daten in eine Liste ein. Die Ausgabe ist ein Mehrfachinstanz-Datenbehälter, der die Liste enthält. Weitere Informationen hierzu finden Sie unter ["Mehrfachinstanz-Datenbehälter" auf Seite 202](#).

Geben Sie die Datenwerte in diese Komponente geschachtelt ein.

Die nachstehende Tabelle beschreibt die Eigenschaften des Validators **CreateList**:

Eigenschaft	Beschreibung
data_holder	Definiert den Datenbehälter mit Mehrfachvorkommen, in dem die Aktion die Liste speichert. Der Datenbehälter muss über einfache Datentypen verfügen.
disabled	Legt fest, ob die Komponente und alle zugehörigen untergeordneten Komponenten vom Skript ignoriert werden. Verwenden Sie diese Eigenschaft zum Testen, Debuggen und Ändern eines Skripts. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none">- Ausgewählt. Das Skript ignoriert die Komponente.- Gelöscht. Das Skript wendet die Komponente an. Standardmäßig ist die Eigenschaft deaktiviert.
name	Eine beschreibende Kennzeichnung für die Komponente. Diese Kennzeichnung wird in der Protokolldatei sowie der Ereignisansicht angezeigt. Ermitteln Sie anhand der Eigenschaft name , welche Komponente das Ereignis verursacht hat.
on_fail	Die Aktion, die beim Ausfall der Komponente durchgeführt wird. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none">- Gelöscht. Es wird keine Aktion ausgeführt.- CustomLog. Es wird in das Benutzerprotokoll geschrieben.- LogError. Eine Fehlermeldung wird in das Engine-Protokoll geschrieben.- LogInfo. Eine Informationsmeldung wird in das Engine-Protokoll geschrieben.- LogWarning. Eine Warnmeldung wird in das Engine-Protokoll geschrieben.- NotifyFailure. Eine Mitteilung wird gesendet. Die Standardoption lautet „Gelöscht“. Weitere Informationen über die Behandlung von Komponentenfehlern finden Sie in "Fehlerbehandlung" auf Seite 410 .
phase	Legt den Zeitpunkt fest, an dem das Skript die Komponente verarbeitet. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none">- Anfangsphase. Das Skript verarbeitet die Komponente während der Anfangsphase.- Hauptphase. Das Skript verarbeitet die Komponente während der Hauptphase.- Endphase. Das Skript verarbeitet die Komponente während der Endphase. Weitere Informationen hierzu finden Sie unter "So sucht der Parser Anker" auf Seite 211 . Standardwert ist "Hauptphase".
remark	Ein benutzerdefinierter Kommentar, der den Zweck oder die Aktion der Komponente beschreibt.

Beispiel

Die Eingabedatenwerte sind:

```
Jack
Jennie
Larissa
```

Die Aktion kann daraus die folgende Ausgabe erzeugen:

```
<Name>
  <First>Jack</First>
  <First>Jennie</First>
```

```
<First>Larissa</First>
</Name>
```

CustomLog

Die Aktion **CustomLog** kann als Wert für die Eigenschaft **on_fail** verwendet werden. Wenn ein Fehlschlag eintritt, führt die Aktion **CustomLog** einen Serializer aus, der eine Protokollnachricht vorbereitet. Das System schreibt die Nachricht in einen angegebenen Ausgabeort.

Die nachstehende Tabelle beschreibt die Eigenschaften der Aktion **CustomLog**:

Eigenschaft	Beschreibung
run_serializer	Legt den Serializer fest, der die Protokollnachricht vorbereitet. Definieren Sie einen Serializer an diesem Speicherort, oder geben Sie den Namen eines global definierten Serializers ein.
output	<p>Legt den Ausgabeort fest. Die Eigenschaft output hat folgende Optionen:</p> <ul style="list-style-type: none"> - OutputDataHolder. Schreibt in einen Datenbehälter. - OutputFile. Schreibt in eine Datei. - OutputPort. Definiert den Namen eines AdditionalOutputPort, in den die Daten geschrieben werden. - ResultFile. Schreibt in die Standardergebnisdatei der Umwandlung. - StandardErrorLog. Schreibt in das Benutzerprotokoll. <p>Standardwert ist StandardErrorLog. Weitere Informationen zu diesen Optionen finden Sie in den Abschnitten "Aktions-Teilkomponenten: Referenz" auf Seite 338 und "Fehlerbehandlung" auf Seite 410.</p>

Weitere Informationen zur Eigenschaft **on_fail** finden Sie in unter ["Fehlerbehandlung" auf Seite 410](#).

DateAddICU

Die **DateAddICU**-Aktion erhöht ein Datum.

Die nachstehende Tabelle beschreibt die Eigenschaften der Aktion **DateAddICU**:

Eigenschaft	Beschreibung
disabled	<p>Legt fest, ob die Komponente und alle zugehörigen untergeordneten Komponenten vom Skript ignoriert werden. Verwenden Sie diese Eigenschaft zum Testen, Debuggen und Ändern eines Skripts. Sie können eine der folgenden Optionen auswählen:</p> <ul style="list-style-type: none"> - Ausgewählt. Das Skript ignoriert die Komponente. - Gelöscht. Das Skript wendet die Komponente an. <p>Standardmäßig ist die Eigenschaft deaktiviert.</p>
input_date	Definiert das Datum, das erhöht werden soll.
input_format	Definiert einen String oder einen Datenbehälter, der das Datenformat definiert, beispielsweise <code>dd/MM/yy</code> . Weitere Informationen hierzu finden Sie unter "DateFormatICU" auf Seite 270 .
name	Eine beschreibende Kennzeichnung für die Komponente. Diese Kennzeichnung wird in der Protokolldatei sowie der Ereignisansicht angezeigt. Ermitteln Sie anhand der Eigenschaft name , welche Komponente das Ereignis verursacht hat.
num_of_days	Definiert eine positive oder negative Ganzzahl oder einen Datenbehälter, der die Anzahl der hinzuzufügenden Tagen enthält.

Eigenschaft	Beschreibung
on_fail	<p>Die Aktion, die beim Ausfall der Komponente durchgeführt wird. Sie können eine der folgenden Optionen auswählen:</p> <ul style="list-style-type: none"> - Gelöscht. Es wird keine Aktion ausgeführt. - CustomLog. Es wird in das Benutzerprotokoll geschrieben. - LogError. Eine Fehlermeldung wird in das Engine-Protokoll geschrieben. - LogInfo. Eine Informationsmeldung wird in das Engine-Protokoll geschrieben. - LogWarning. Eine Warnmeldung wird in das Engine-Protokoll geschrieben. - NotifyFailure. Eine Mitteilung wird gesendet. <p>Die Standardoption lautet „Gelöscht“. Weitere Informationen über die Behandlung von Komponentenfehlern finden Sie in "Fehlerbehandlung" auf Seite 410.</p>
optional	<p>Legt fest, ob ein Komponentenfehler den Ausfall der übergeordneten Komponente verursacht. Sie können eine der folgenden Optionen auswählen:</p> <ul style="list-style-type: none"> - Ausgewählt. Ein Komponentenfehler führt nicht zum Ausfall der übergeordneten Komponente. - Gelöscht. Ein Komponentenfehler verursacht den Ausfall der übergeordneten Komponente. <p>Standardwert ist "Gelöscht". Weitere Informationen über Komponentenfehler finden Sie in "Fehlerbehandlung" auf Seite 410.</p>
output	Legt den Datenbehälter fest, der das Ausgabedatum speichert.
phase	<p>Legt den Zeitpunkt fest, an dem das Skript die Komponente verarbeitet. Sie können eine der folgenden Optionen auswählen:</p> <ul style="list-style-type: none"> - Anfangsphase. Das Skript verarbeitet die Komponente während der Anfangsphase. - Hauptphase. Das Skript verarbeitet die Komponente während der Hauptphase. - Endphase. Das Skript verarbeitet die Komponente während der Endphase. <p>Weitere Informationen hierzu finden Sie unter "So sucht der Parser Anker" auf Seite 211. Standardwert ist "Hauptphase".</p>
remark	Ein benutzerdefinierter Kommentar, der den Zweck oder die Aktion der Komponente beschreibt.

DateDiffICU

Die Aktion **DateDiffICU** berechnet die Differenz zwischen zwei Datumsangaben.

Die nachstehende Tabelle beschreibt die Eigenschaften der Aktion **DateDiffICU**:

Eigenschaft	Beschreibung
date1	Definiert einen String oder einen Datenbehälter, der das erste Datum definiert.
date2	Definiert einen String oder einen Datenbehälter, der das zweite Datum definiert.
date_format1	<p>Definiert einen String oder einen Datenbehälter, der das Format des ersten Datums definiert, z. B. dd/MM/yy. Wenn Sie das Format weglassen, wird das Standardformat des Systems verwendet.</p> <p>Weitere Informationen hierzu finden Sie unter "DateFormatICU" auf Seite 270.</p>
date_format2	Definiert einen String oder Datenbehälter, der das Format des zweiten Datums definiert.

Eigenschaft	Beschreibung
disabled	<p>Legt fest, ob die Komponente und alle zugehörigen untergeordneten Komponenten vom Skript ignoriert werden. Verwenden Sie diese Eigenschaft zum Testen, Debuggen und Ändern eines Skripts. Sie können eine der folgenden Optionen auswählen:</p> <ul style="list-style-type: none"> - Ausgewählt. Das Skript ignoriert die Komponente. - Gelöscht. Das Skript wendet die Komponente an. <p>Standardmäßig ist die Eigenschaft deaktiviert.</p>
name	<p>Eine beschreibende Kennzeichnung für die Komponente. Diese Kennzeichnung wird in der Protokolldatei sowie der Ereignisansicht angezeigt. Ermitteln Sie anhand der Eigenschaft name, welche Komponente das Ereignis verursacht hat.</p>
on_fail	<p>Die Aktion, die beim Ausfall der Komponente durchgeführt wird. Sie können eine der folgenden Optionen auswählen:</p> <ul style="list-style-type: none"> - Gelöscht. Es wird keine Aktion ausgeführt. - CustomLog. Es wird in das Benutzerprotokoll geschrieben. - LogError. Eine Fehlermeldung wird in das Engine-Protokoll geschrieben. - LogInfo. Eine Informationsmeldung wird in das Engine-Protokoll geschrieben. - LogWarning. Eine Warnmeldung wird in das Engine-Protokoll geschrieben. - NotifyFailure. Eine Mitteilung wird gesendet. <p>Die Standardoption lautet „Gelöscht“. Weitere Informationen über die Behandlung von Komponentenfehlern finden Sie in "Fehlerbehandlung" auf Seite 410.</p>
optional	<p>Legt fest, ob ein Komponentenfehler den Ausfall der übergeordneten Komponente verursacht. Sie können eine der folgenden Optionen auswählen:</p> <ul style="list-style-type: none"> - Ausgewählt. Ein Komponentenfehler führt nicht zum Ausfall der übergeordneten Komponente. - Gelöscht. Ein Komponentenfehler verursacht den Ausfall der übergeordneten Komponente. <p>Standardwert ist "Gelöscht". Weitere Informationen über Komponentenfehler finden Sie in "Fehlerbehandlung" auf Seite 410.</p>
output	<p>Definiert den Datenbehälter, der das Ergebnis speichert.</p>
phase	<p>Legt den Zeitpunkt fest, an dem das Skript die Komponente verarbeitet. Sie können eine der folgenden Optionen auswählen:</p> <ul style="list-style-type: none"> - Anfangsphase. Das Skript verarbeitet die Komponente während der Anfangsphase. - Hauptphase. Das Skript verarbeitet die Komponente während der Hauptphase. - Endphase. Das Skript verarbeitet die Komponente während der Endphase. <p>Weitere Informationen hierzu finden Sie unter "So sucht der Parser Anker" auf Seite 211. Standardwert ist "Hauptphase".</p>
remark	<p>Ein benutzerdefinierter Kommentar, der den Zweck oder die Aktion der Komponente beschreibt.</p>

DownloadFileToDataHolder

Die Aktion **DownloadFileToDataHolder** lädt eine Datei von einem Webserver herunter und speichert ihren Inhalt in einem Datenbehälter. Die Aktion wandelt Symbole in XML-Entitäten um.

Die nachstehende Tabelle beschreibt die Eigenschaften der Aktion **DownloadFileToDataHolder**:

Eigenschaft	Beschreibung
disabled	Legt fest, ob die Komponente und alle zugehörigen untergeordneten Komponenten vom Skript ignoriert werden. Verwenden Sie diese Eigenschaft zum Testen, Debuggen und Ändern eines Skripts. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none"> - Ausgewählt. Das Skript ignoriert die Komponente. - Gelöscht. Das Skript wendet die Komponente an. Standardmäßig ist die Eigenschaft deaktiviert.
file_url	Legt einen Datenbehälter fest, in dem die URL der Datei gespeichert ist.
name	Eine beschreibende Kennzeichnung für die Komponente. Diese Kennzeichnung wird in der Protokolldatei sowie der Ereignisansicht angezeigt. Ermitteln Sie anhand der Eigenschaft name , welche Komponente das Ereignis verursacht hat.
on_fail	Die Aktion, die beim Ausfall der Komponente durchgeführt wird. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none"> - Gelöscht. Es wird keine Aktion ausgeführt. - CustomLog. Es wird in das Benutzerprotokoll geschrieben. - LogError. Eine Fehlermeldung wird in das Engine-Protokoll geschrieben. - LogInfo. Eine Informationsmeldung wird in das Engine-Protokoll geschrieben. - LogWarning. Eine Warnmeldung wird in das Engine-Protokoll geschrieben. - NotifyFailure. Eine Mitteilung wird gesendet. Die Standardoption lautet „Gelöscht“. Weitere Informationen über die Behandlung von Komponentenfehlern finden Sie in "Fehlerbehandlung" auf Seite 410 .
optional	Legt fest, ob ein Komponentenfehler den Ausfall der übergeordneten Komponente verursacht. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none"> - Ausgewählt. Ein Komponentenfehler führt nicht zum Ausfall der übergeordneten Komponente. - Gelöscht. Ein Komponentenfehler verursacht den Ausfall der übergeordneten Komponente. Standardwert ist "Gelöscht". Weitere Informationen über Komponentenfehler finden Sie in "Fehlerbehandlung" auf Seite 410 .
output	Legt den Datenbehälter fest, der den heruntergeladenen Inhalt speichert.
Phase	Legt den Zeitpunkt fest, an dem das Skript die Komponente verarbeitet. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none"> - Anfangsphase. Das Skript verarbeitet die Komponente während der Anfangsphase. - Hauptphase. Das Skript verarbeitet die Komponente während der Hauptphase. - Endphase. Das Skript verarbeitet die Komponente während der Endphase. Weitere Informationen hierzu finden Sie unter "So sucht der Parser Anker" auf Seite 211 . Standardwert ist "Hauptphase".
remark	Ein benutzerdefinierter Kommentar, der den Zweck oder die Aktion der Komponente beschreibt.

DumpValues

Die Aktion **DumpValues** ist ein Debugging-Tool. Es schreibt Daten in das Element `<DumpValues>...</DumpValues>`. Definieren Sie die Datenbehälter, die Sie sichern wollen, indem Sie sie als untergeordnete Komponenten schachteln.

Die nachstehende Tabelle beschreibt die Eigenschaften der Aktion **DumpValues**:

Eigenschaft	Beschreibung
disabled	Legt fest, ob die Komponente und alle zugehörigen untergeordneten Komponenten vom Skript ignoriert werden. Verwenden Sie diese Eigenschaft zum Testen, Debuggen und Ändern eines Skripts. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none"> - Ausgewählt. Das Skript ignoriert die Komponente. - Gelöscht. Das Skript wendet die Komponente an. Standardmäßig ist die Eigenschaft deaktiviert.
name	Eine beschreibende Kennzeichnung für die Komponente. Diese Kennzeichnung wird in der Protokolldatei sowie der Ereignisansicht angezeigt. Ermitteln Sie anhand der Eigenschaft name , welche Komponente das Ereignis verursacht hat.
on_fail	Die Aktion, die beim Ausfall der Komponente durchgeführt wird. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none"> - Gelöscht. Es wird keine Aktion ausgeführt. - CustomLog. Es wird in das Benutzerprotokoll geschrieben. - LogError. Eine Fehlermeldung wird in das Engine-Protokoll geschrieben. - LogInfo. Eine Informationsmeldung wird in das Engine-Protokoll geschrieben. - LogWarning. Eine Warnmeldung wird in das Engine-Protokoll geschrieben. - NotifyFailure. Eine Mitteilung wird gesendet. Die Standardoption lautet „Gelöscht“. Weitere Informationen über die Behandlung von Komponentenfehlern finden Sie in "Fehlerbehandlung" auf Seite 410 .
output	Legt den Datenbehälter fest, der die Ausgabe speichert. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none"> - OutputFile - ResultFile - StandardErrorLog Standardwert ist ResultFile.
Phase	Legt den Zeitpunkt fest, an dem das Skript die Komponente verarbeitet. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none"> - Anfangsphase. Das Skript verarbeitet die Komponente während der Anfangsphase. - Hauptphase. Das Skript verarbeitet die Komponente während der Hauptphase. - Endphase. Das Skript verarbeitet die Komponente während der Endphase. Weitere Informationen hierzu finden Sie unter "So sucht der Parser Anker" auf Seite 211 . Standardwert ist "Hauptphase".
remark	Ein benutzerdefinierter Kommentar, der den Zweck oder die Aktion der Komponente beschreibt.

EnsureCondition

Die Aktion **EnsureCondition** wertet einen booleschen JavaScript-Ausdruck aus. Wenn der Ausdruck `false` ist, schlägt die Aktion fehl.

Die nachstehende Tabelle beschreibt die Eigenschaften der Aktion **EnsureCondition**:

Eigenschaft	Beschreibung
Bedingung	Definiert einen JavaScript-Ausdruck für die Auswertung. Im Ausdruck referenzieren Sie die Parameter, die in params definiert sind, mit einem Dollarzeichen (\$), auf das eine ganze Zahl folgt. Beispiel: Der folgende Ausdruck überprüft, ob der erste Parameter den Wert <code>Ron Lehrer</code> hat: <code>\$1 == "Ron Lehrer"</code>
disabled	Legt fest, ob die Komponente und alle zugehörigen untergeordneten Komponenten vom Skript ignoriert werden. Verwenden Sie diese Eigenschaft zum Testen, Debuggen und Ändern eines Skripts. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none"> - Ausgewählt. Das Skript ignoriert die Komponente. - Gelöscht. Das Skript wendet die Komponente an. Standardmäßig ist die Eigenschaft deaktiviert.
name	Eine beschreibende Kennzeichnung für die Komponente. Diese Kennzeichnung wird in der Protokolldatei sowie der Ereignisansicht angezeigt. Ermitteln Sie anhand der Eigenschaft name , welche Komponente das Ereignis verursacht hat.
on_fail	Die Aktion, die beim Ausfall der Komponente durchgeführt wird. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none"> - Gelöscht. Es wird keine Aktion ausgeführt. - CustomLog. Es wird in das Benutzerprotokoll geschrieben. - LogError. Eine Fehlermeldung wird in das Engine-Protokoll geschrieben. - LogInfo. Eine Informationsmeldung wird in das Engine-Protokoll geschrieben. - LogWarning. Eine Warnmeldung wird in das Engine-Protokoll geschrieben. - NotifyFailure. Eine Mitteilung wird gesendet. Die Standardoption lautet „Gelöscht“. Weitere Informationen über die Behandlung von Komponentengefehlern finden Sie in "Fehlerbehandlung" auf Seite 410 .
params	Definiert eine Liste der Datenbehälter.
Phase	Legt den Zeitpunkt fest, an dem das Skript die Komponente verarbeitet. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none"> - Anfangsphase. Das Skript verarbeitet die Komponente während der Anfangsphase. - Hauptphase. Das Skript verarbeitet die Komponente während der Hauptphase. - Endphase. Das Skript verarbeitet die Komponente während der Endphase. Weitere Informationen hierzu finden Sie unter "So sucht der Parser Anker" auf Seite 211 . Standardwert ist "Hauptphase".
remark	Ein benutzerdefinierter Kommentar, der den Zweck oder die Aktion der Komponente beschreibt.

JavaScript-Standardsyntax

Der JavaScript-Prozessor unterstützt JavaScript-Standardausdrücke, die folgende Merkmale aufweisen.

- Unäre und binäre Operatoren:

`() + - * / % == != < <= > >= && ||`

- Den ternären Operator `?:`.
- Die folgenden Methoden:

```
charAt
indexOf
lastIndexOf
length
join
```

```
substring  
toString
```

Wenn Sie diese Methoden auf ein Literal mit einem einfachen Datentyp anwenden, müssen Sie das Literal in Klammern einschließen. Beispiel:

```
123.toString();           //Wrong  
(123).toString();        //Right  
  
"Hello, World".substring(3,7); //Wrong  
("Hello, World").substring(3,7); //Right
```

- Die folgenden Funktionen:

```
Math.ceil  
Math.floor  
Math.max  
Math.min  
Math.pow  
Math.sqrt  
parseFloat  
parseInt
```

Merkmale wie die folgenden unterstützt der interne JavaScript-Prozessor nicht:

- Unäre und binäre Operatoren:

```
++ -- typeof void >> >>> << === !== ~ & | ^
```

- Zuordnungsoperatoren:

```
= += -= *= /= >>= >>>= <<= &= |= ^=
```

- Der Komma-Operator `(,)`.
- Die Werte `NaN`, `null`, `infinity` oder `-0` (negative 0).
- Andere Datentypen als „string“, „number“ und „boolean“.
- Das Objekt `Date`.
- Die Funktion `equalsIgnoreCase`.

JavaScript-Erweiterungen

Der JavaScript-Prozessor implementiert die folgenden Methoden, die standardmäßig nicht in JavaScript definiert sind. Sie können diese Erweiterungen an jeder beliebigen Position verwenden, wo das Skript einen JavaScript-Ausdruck akzeptiert.

Die meisten Funktionen sind JavaScript-Implementierungen von Transformern oder Aktionen.

```
extra.sum(number1, number 2, ...)
```

Gibt die Summe der Parameter zurück.

```
extra.allSame(param1, param2, ...)
```

Gibt "True" zurück, wenn alle Parameter denselben Wert haben.

```
lookup.<lookup_name>(Schlüssel)
```

Diese Funktion greift nach Namen auf eine globale Lookup-Tabelle zu.

Definieren Sie im Skript eine globale `InlineTable` oder `XMLLookupTable`. Im JavaScript-Ausdruck können Sie auf die Tabelle zugreifen. Wenn Sie zum Beispiel eine globale `InlineTable` mit dem Namen `USPresidents` erstellen, gibt `lookup.USPresidents(1)` das Ergebnis `George Washington` zurück.

Weitere Informationen hierzu finden Sie unter ["LookupTransformer" auf Seite 284](#).


```
extra.formatDate(date, input_format, output_format)
```

Formatiert eine Datums- oder Uhrzeitangabe. Weitere Informationen hierzu finden Sie unter ["DateFormatICU" auf Seite 270](#).

```
extra.substitute(text, oldSubstring, newSubstring, useRegex)
```

Ersetzt alle Instanzen einer Unterzeichenfolge durch eine andere Unterzeichenfolge. Wenn `useRegex` "wahr" ist, interpretiert die Methode `oldSubstring` als regulären Ausdruck.

```
extra.formatNumber(number, size_of_integer_part, number_of_decimals, sign, insert_decimal_point, unit_type)
```

Formatiert eine Zahl. Weitere Informationen hierzu finden Sie unter ["FormatNumber" auf Seite 276](#).

```
extra.insertString(text, injections_place, string_to_inject)
```

Fügt einen String in Text ein. Weitere Informationen hierzu finden Sie unter ["InjectString" auf Seite 282](#).

```
extra.formatTransformationStartTime(format)
```

Gibt das Datum und die Uhrzeit aus, zu denen die Umwandlung begonnen wurde. Weitere Informationen hierzu finden Sie unter ["TransformationStartTime" auf Seite 295](#).

```
extra.resize(text, size, padding_character, align)
```

Passt den Eingabetext einer angegebenen Größe an und ergänzt oder kürzt gegebenenfalls. Weitere Informationen hierzu finden Sie unter ["Resize" auf Seite 291](#).

```
extra.dateAdd(date_format, date, days_to_add)
```

Erhöht ein Datum um eine vorgegebene Anzahl von Tagen. Weitere Informationen über `date_format` finden Sie unter ["DateFormatICU" auf Seite 270](#).

```
extra.dateAddMonths(date_format, date, months_to_add)
```

Erhöht ein Datum um eine vorgegebene Anzahl von Monaten. Weitere Informationen über `date_format` finden Sie unter ["DateFormatICU" auf Seite 270](#).

```
extra.dateAddYears(date_format, date, years_to_add)
```

Erhöht ein Datum um eine vorgegebene Anzahl von Jahren. Weitere Informationen über `date_format` finden Sie unter ["DateFormatICU" auf Seite 270](#).

```
extra.dateDiff(date_format1, date1, date_format2, date2)
```

Berechnet die Differenz zwischen zwei Daten. Weitere Informationen hierzu finden Sie unter ["DateDiffICU" auf Seite 315](#).

```
extra.createGuid(0)
```

Erzeugt den Bezeichner GUID. Weitere Informationen hierzu finden Sie unter ["CreateGuid" auf Seite 270](#). Sie müssen einen Parameterwert angeben, etwa 0.

```
extra.upper(text), extra.lower(text), extra.capitalize(text)
```

Ändert den Text in Großbuchstaben, in Kleinbuchstaben oder so, dass nur der erste Buchstabe ein Großbuchstabe ist. Weitere Informationen finden Sie unter ["ChangeCase" auf Seite 269](#).

```
extra.rtl2ltr(text)
```

Ändert einen Text, der in einer Sprache von rechts nach links geschrieben ist, in Text, der von links nach rechts fließt. Weitere Informationen hierzu finden Sie unter ["hebrewBidi" auf Seite 280](#).

```
extra.trim(text)
```

Löscht vorangestellte und angehängte Leerzeichen aus dem Text. Weitere Informationen hierzu finden Sie unter ["RemoveMarginSpace" auf Seite 288](#).

ExcludelItems

Die Aktion **ExcludelItems** entfernt festgelegte Werte aus einem Datenbehälter mit mehreren Vorkommen. Der Datenbehältertyp muss einfach sein. Um spezifische Strings aus einem Datenbehälter auszuschließen, definieren Sie sie als untergeordnete Komponenten. Weitere Informationen hierzu finden Sie unter ["Mehrfachinstanz-Datenbehälter" auf Seite 202](#).

Die nachstehende Tabelle beschreibt die Eigenschaften der Aktion **ExcludelItems**:

Eigenschaft	Beschreibung
data_holder	Definiert einen Datenbehälter mit mehreren Vorkommen.
disabled	Legt fest, ob die Komponente und alle zugehörigen untergeordneten Komponenten vom Skript ignoriert werden. Verwenden Sie diese Eigenschaft zum Testen, Debuggen und Ändern eines Skripts. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none">- Ausgewählt. Das Skript ignoriert die Komponente.- Gelöscht. Das Skript wendet die Komponente an. Standardmäßig ist die Eigenschaft deaktiviert.
name	Eine beschreibende Kennzeichnung für die Komponente. Diese Kennzeichnung wird in der Protokolldatei sowie der Ereignisansicht angezeigt. Ermitteln Sie anhand der Eigenschaft name , welche Komponente das Ereignis verursacht hat.
on_fail	Die Aktion, die beim Ausfall der Komponente durchgeführt wird. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none">- Gelöscht. Es wird keine Aktion ausgeführt.- CustomLog. Es wird in das Benutzerprotokoll geschrieben.- LogError. Eine Fehlermeldung wird in das Engine-Protokoll geschrieben.- LogInfo. Eine Informationsmeldung wird in das Engine-Protokoll geschrieben.- LogWarning. Eine Warnmeldung wird in das Engine-Protokoll geschrieben.- NotifyFailure. Eine Mitteilung wird gesendet. Die Standardoption lautet „Gelöscht“. Weitere Informationen über die Behandlung von Komponentenfehlern finden Sie in "Fehlerbehandlung" auf Seite 410 .
optional	Legt fest, ob ein Komponentenfehler den Ausfall der übergeordneten Komponente verursacht. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none">- Ausgewählt. Ein Komponentenfehler führt nicht zum Ausfall der übergeordneten Komponente.- Gelöscht. Ein Komponentenfehler verursacht den Ausfall der übergeordneten Komponente. Standardwert ist "Gelöscht". Weitere Informationen über Komponentenfehler finden Sie in "Fehlerbehandlung" auf Seite 410 .
Phase	Legt den Zeitpunkt fest, an dem das Skript die Komponente verarbeitet. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none">- Anfangsphase. Das Skript verarbeitet die Komponente während der Anfangsphase.- Hauptphase. Das Skript verarbeitet die Komponente während der Hauptphase.- Endphase. Das Skript verarbeitet die Komponente während der Endphase. Weitere Informationen hierzu finden Sie unter "So sucht der Parser Anker" auf Seite 211 . Standardwert ist "Hauptphase".
remark	Ein benutzerdefinierter Kommentar, der den Zweck oder die Aktion der Komponente beschreibt.

Map

Die Aktion **Map** kopiert einen Wert aus einem Datenbehälter in einen anderen.

Die nachstehende Tabelle beschreibt die Eigenschaften der Aktion **Map**:

Eigenschaft	Beschreibung
disabled	Legt fest, ob die Komponente und alle zugehörigen untergeordneten Komponenten vom Skript ignoriert werden. Verwenden Sie diese Eigenschaft zum Testen, Debuggen und Ändern eines Skripts. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none"> - Ausgewählt. Das Skript ignoriert die Komponente. - Gelöscht. Das Skript wendet die Komponente an. Standardmäßig ist die Eigenschaft deaktiviert.
name	Eine beschreibende Kennzeichnung für die Komponente. Diese Kennzeichnung wird in der Protokolldatei sowie der Ereignisansicht angezeigt. Ermitteln Sie anhand der Eigenschaft name , welche Komponente das Ereignis verursacht hat.
on_fail	Die Aktion, die beim Ausfall der Komponente durchgeführt wird. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none"> - Gelöscht. Es wird keine Aktion ausgeführt. - CustomLog. Es wird in das Benutzerprotokoll geschrieben. - LogError. Eine Fehlermeldung wird in das Engine-Protokoll geschrieben. - LogInfo. Eine Informationsmeldung wird in das Engine-Protokoll geschrieben. - LogWarning. Eine Warnmeldung wird in das Engine-Protokoll geschrieben. - NotifyFailure. Eine Mitteilung wird gesendet. Die Standardoption lautet „Gelöscht“. Weitere Informationen über die Behandlung von Komponentenfehlern finden Sie in "Fehlerbehandlung" auf Seite 410 .
optional	Legt fest, ob ein Komponentenfehler den Ausfall der übergeordneten Komponente verursacht. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none"> - Ausgewählt. Ein Komponentenfehler führt nicht zum Ausfall der übergeordneten Komponente. - Gelöscht. Ein Komponentenfehler verursacht den Ausfall der übergeordneten Komponente. Standardwert ist "Gelöscht". Weitere Informationen über Komponentenfehler finden Sie in "Fehlerbehandlung" auf Seite 410 .
phase	Legt den Zeitpunkt fest, an dem das Skript die Komponente verarbeitet. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none"> - Anfangsphase. Das Skript verarbeitet die Komponente während der Anfangsphase. - Hauptphase. Das Skript verarbeitet die Komponente während der Hauptphase. - Endphase. Das Skript verarbeitet die Komponente während der Endphase. Weitere Informationen hierzu finden Sie unter "So sucht der Parser Anker" auf Seite 211 . Standardwert ist "Hauptphase".
remark	Ein benutzerdefinierter Kommentar, der den Zweck oder die Aktion der Komponente beschreibt.
source	Legt den Quelldatenbehälter fest.
source_default	Legt den Standard-Quelldatenbehälter fest.
target	Legt den Zieldatenbehälter fest.
transformers	Definiert eine Folge von Transformern, die den Wert bearbeiten. Diese Eigenschaft darf nicht zugewiesen werden, wenn Quelle und Ziel komplexe XML-Elemente sind.
validators	Definiert eine Liste von Validatoren, die auf die Quelldaten angewandt werden. Weitere Informationen hierzu finden Sie unter "Validatoren" auf Seite 413 .

Wenn ein Datenbehälter eines einfachen Datentyps kopiert wird, müssen Quelle und Ziel kompatible Datentypen haben. Die Aktion kann auf den kopierten Wert Transformer anwenden.

Wenn Sie einen Mehrfachinstanz-Datenbehälter mit Einfachtyp kopieren und sich die Aktion nicht in einer Iterationskomponente, etwa einer **RepeatingGroup**, befindet, kopiert die Aktion alle Instanzen des Datenbehälters.

Wird ein Datenbehälter eines Komplextyps kopiert, müssen die interne Struktur und der Datentyp von Quelle und Ziel genau übereinstimmen. Die Aktion kopiert auch die geschachtelten Elemente und Attribute.

Online-Beispiel

Ein Online-Beispiel zu dieser Aktion befindet sich im Projekt `samples\Projects\CopyValue\CopyValue.cmw`. In diesem Beispiel wird mit Hilfe der Aktion `Map` ein komplexes Element kopiert, das ein Attribut und geschachtelte Elemente enthält.

Benachrichtigen

Die **Benachrichtigen**-Aktion löst eine Benachrichtigung aus. Verwenden Sie sie für das Einfügen einer Warnmeldung in die Umwandlungsausgabe.

Die nachstehende Tabelle beschreibt die Eigenschaften der Aktion **Notify**:

Eigenschaft	Beschreibung
disabled	Legt fest, ob die Komponente und alle zugehörigen untergeordneten Komponenten vom Skript ignoriert werden. Verwenden Sie diese Eigenschaft zum Testen, Debuggen und Ändern eines Skripts. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none"> - Ausgewählt. Das Skript ignoriert die Komponente. - Gelöscht. Das Skript wendet die Komponente an. Standardmäßig ist die Eigenschaft deaktiviert.
name	Eine beschreibende Kennzeichnung für die Komponente. Diese Kennzeichnung wird in der Protokolldatei sowie der Ereignisansicht angezeigt. Ermitteln Sie anhand der Eigenschaft name , welche Komponente das Ereignis verursacht hat.
notify	Definiert eine Benachrichtigung. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none"> - MandatoryStructureMissing. Ein verpflichtender Datensatz erscheint nicht in der Eingabe. - MismatchIDs. Die ID des Datensatzes und des Unterelements stimmen teilweise überein. - StructureBelowMinOccurs. Es gibt weniger passende übereinstimmende Datensätze des Unterelements als in minOccurs definiert sind. - StructureExceedsMaxOccurs. Es gibt mehr passende übereinstimmende Datensätze des Unterelements als in maxOccurs definiert sind. - StructureOutOfSequence. Die Datensätze stimmen mit den Unterelementen überein, aber nicht in der erforderlichen Sequenz. - UnexpectedRecord. Die Datensätze stimmen mit den Unterelementen überein, aber nicht in der erforderlichen Hierarchie. - UnrecognizedRecord. Kein Unterelement stimmt mit den Datensatzbezeichnern überein. - XsdValidationError. Die Eingabe stimmt nicht mit den Anforderungen des Schemas überein. - Benutzerdefinierte Benachrichtigung- oder NotificationGroup-Komponente. Benutzerdefinierte Meldung. Weitere Informationen hierzu finden Sie unter "Benachrichtigungen" auf Seite 247 .

Eigenschaft	Beschreibung
phase	<p>Legt den Zeitpunkt fest, an dem das Skript die Komponente verarbeitet. Sie können eine der folgenden Optionen auswählen:</p> <ul style="list-style-type: none"> - Anfangsphase. Das Skript verarbeitet die Komponente während der Anfangsphase. - Hauptphase. Das Skript verarbeitet die Komponente während der Hauptphase. - Endphase. Das Skript verarbeitet die Komponente während der Endphase. <p>Weitere Informationen hierzu finden Sie unter "So sucht der Parser Anker" auf Seite 211. Standardwert ist "Endgültig".</p>
value	<p>Definiert einen Wert für die Zuweisung an die <i>VarNotificationDetails/Value</i>-Variable. Ein NotificationHandler kann den Wert in seiner Ausgabe enthalten.</p>

ResetVisitedPages

Die Aktion **ResetVisitedPages** löscht die Liste der besuchten Seiten der angegebenen sekundären Parser. Sie lässt mehrere Besuche derselben Seite zu, auch wenn **reject_recurring_pages** ausgewählt ist. Verwenden Sie diese Aktion, um verschiedene Eingabedaten auf derselben Webseite zu veröffentlichen. Sie wird gemeinsam mit der Eigenschaft **reject_recurring_pages** einer **Parser**-Komponente verwendet.

Die nachstehende Tabelle beschreibt die Eigenschaften der Aktion **ResetVisitedPages**:

Eigenschaft	Beschreibung
disabled	<p>Legt fest, ob die Komponente und alle zugehörigen untergeordneten Komponenten vom Skript ignoriert werden. Verwenden Sie diese Eigenschaft zum Testen, Debuggen und Ändern eines Skripts. Sie können eine der folgenden Optionen auswählen:</p> <ul style="list-style-type: none"> - Ausgewählt. Das Skript ignoriert die Komponente. - Gelöscht. Das Skript wendet die Komponente an. <p>Standardmäßig ist die Eigenschaft deaktiviert.</p>
name	<p>Eine beschreibende Kennzeichnung für die Komponente. Diese Kennzeichnung wird in der Protokolldatei sowie der Ereignisansicht angezeigt. Ermitteln Sie anhand der Eigenschaft name, welche Komponente das Ereignis verursacht hat.</p>
on_fail	<p>Die Aktion, die beim Ausfall der Komponente durchgeführt wird. Sie können eine der folgenden Optionen auswählen:</p> <ul style="list-style-type: none"> - Gelöscht. Es wird keine Aktion ausgeführt. - CustomLog. Es wird in das Benutzerprotokoll geschrieben. - LogError. Eine Fehlermeldung wird in das Engine-Protokoll geschrieben. - LogInfo. Eine Informationsmeldung wird in das Engine-Protokoll geschrieben. - LogWarning. Eine Warnmeldung wird in das Engine-Protokoll geschrieben. - NotifyFailure. Eine Mitteilung wird gesendet. <p>Die Standardoption lautet „Gelöscht“. Weitere Informationen über die Behandlung von Komponentenfehlern finden Sie in "Fehlerbehandlung" auf Seite 410.</p>
Parser	<p>Definiert eine Liste von Parsern. Die Liste der besuchten Seiten jedes Parsers wird zurückgesetzt.</p>

Eigenschaft	Beschreibung
phase	<p>Legt den Zeitpunkt fest, an dem das Skript die Komponente verarbeitet. Sie können eine der folgenden Optionen auswählen:</p> <ul style="list-style-type: none"> - Anfangsphase. Das Skript verarbeitet die Komponente während der Anfangsphase. - Hauptphase. Das Skript verarbeitet die Komponente während der Hauptphase. - Endphase. Das Skript verarbeitet die Komponente während der Endphase. <p>Weitere Informationen hierzu finden Sie unter "So sucht der Parser Anker" auf Seite 211. Standardwert ist "Hauptphase".</p>
remark	Ein benutzerdefinierter Kommentar, der den Zweck oder die Aktion der Komponente beschreibt.

RunMapper

Die Aktion **RunMapper** führt einen Mapper als Teilkomponente eines Parsers, Mappers oder Serializers aus.

Die nachstehende Tabelle beschreibt die Eigenschaften der Aktion **RunMapper**:

Eigenschaft	Beschreibung
disabled	<p>Legt fest, ob die Komponente und alle zugehörigen untergeordneten Komponenten vom Skript ignoriert werden. Verwenden Sie diese Eigenschaft zum Testen, Debuggen und Ändern eines Skripts. Sie können eine der folgenden Optionen auswählen:</p> <ul style="list-style-type: none"> - Ausgewählt. Das Skript ignoriert die Komponente. - Gelöscht. Das Skript wendet die Komponente an. <p>Standardmäßig ist die Eigenschaft deaktiviert.</p>
input	<p>Definiert einen Datenbehälter, der XML-Text enthält, an dem der Mapper ausgeführt werden soll. Der Datenbehälter muss einen einfachen Datentyp haben, zum Beispiel <code>xs:string</code>. Der Wert des Strings kann XML-Text beliebiger Komplexität sein. Informationen zum Ausführen eines Mappers an einem Datenbehälter komplexen Typs finden Sie im Abschnitt "EmbeddedMapper" auf Seite 364. Wenn Sie diese Eigenschaft weglassen, verwendet der Mapper die im Gültigkeitsbereich der Aktion vorhandenen Datenbehälter. Ist die Aktion beispielsweise in einen Parser geschachtelt, wird der Mapper an der Ausgabe des Parsers ausgeführt. Befindet sich die Aktion in einer Group, wird er an der Ausgabe der Group ausgeführt.</p>
mapper	<p>Definiert den Mapper. Wählen Sie den Namen einer vorhandenen Mapper-Komponente oder definieren Sie eine Mapper-Komponente an dieser Position im Skript. Weitere Informationen hierzu finden Sie unter "Die Komponente Mapper: Referenz" auf Seite 361.</p>
name	<p>Eine beschreibende Kennzeichnung für die Komponente. Diese Kennzeichnung wird in der Protokolldatei sowie der Ereignisansicht angezeigt. Ermitteln Sie anhand der Eigenschaft name, welche Komponente das Ereignis verursacht hat.</p>
on_fail	<p>Die Aktion, die beim Ausfall der Komponente durchgeführt wird. Sie können eine der folgenden Optionen auswählen:</p> <ul style="list-style-type: none"> - Gelöscht. Es wird keine Aktion ausgeführt. - CustomLog. Es wird in das Benutzerprotokoll geschrieben. - LogError. Eine Fehlermeldung wird in das Engine-Protokoll geschrieben. - LogInfo. Eine Informationsmeldung wird in das Engine-Protokoll geschrieben. - LogWarning. Eine Warnmeldung wird in das Engine-Protokoll geschrieben. - NotifyFailure. Eine Mitteilung wird gesendet. <p>Die Standardoption lautet „Gelöscht“. Weitere Informationen über die Behandlung von Komponentenfehlern finden Sie in "Fehlerbehandlung" auf Seite 410.</p>

Eigenschaft	Beschreibung
optional	Legt fest, ob ein Komponentenfehler den Ausfall der übergeordneten Komponente verursacht. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none"> - Ausgewählt. Ein Komponentenfehler führt nicht zum Ausfall der übergeordneten Komponente. - Gelöscht. Ein Komponentenfehler verursacht den Ausfall der übergeordneten Komponente. Standardwert ist "Gelöscht". Weitere Informationen über Komponentenfehler finden Sie in "Fehlerbehandlung" auf Seite 410 .
phase	Legt den Zeitpunkt fest, an dem das Skript die Komponente verarbeitet. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none"> - Anfangsphase. Das Skript verarbeitet die Komponente während der Anfangsphase. - Hauptphase. Das Skript verarbeitet die Komponente während der Hauptphase. - Endphase. Das Skript verarbeitet die Komponente während der Endphase. Weitere Informationen hierzu finden Sie unter "So sucht der Parser Anker" auf Seite 211 . Standardwert ist "Hauptphase".
remark	Ein benutzerdefinierter Kommentar, der den Zweck oder die Aktion der Komponente beschreibt.

RunMapplet

Mit der Aktion **RunMapplet** wird ein Mapplet ausgeführt. Die Ausgabe von **RunMapplet** wird in den Datenbehälter gelesen, der in der RunMapplet-Aktion angegeben wurde.

Verwenden Sie die Aktion **RunMapplet**, um Aufgaben durchzuführen, wie z. B. Datenmaskierungs-, Datenqualitäts-, Daten-Lookup- oder andere Aktivitäten, die sich in der Regel auf relationale Umwandlungen beziehen. Hierbei entfällt die Notwendigkeit, Daten in das relationale Format und dann zurück in das hierarchische Format zu konvertieren.

Hinweis: Die RunMapplet-Aktion kann nur zum Aufrufen von passiven Mapplets verwendet werden.

Die Ausgabeparameter müssen in derselben Reihenfolge angegeben werden, in der sie in den Zielpoints des Mapplets angezeigt werden.

In der folgenden Tabelle werden die Eigenschaften der Aktion **RunMapplet** beschrieben:

Eigenschaft	Beschreibung
Eingaben	Gibt die an das Mapplet zu übergebenden Eingabewerte an. Die Eingabeparameter müssen in derselben Reihenfolge angegeben werden, in der sie in den Quellports des Mapplets angezeigt werden.
mapplet_name	Gibt das auszuführende Mapplet an. Hinweis: Zuerst müssen Sie für jedes Mapplet, das mit der RunMapplet-Aktion aufgerufen werden soll, einen Verweis auf der Registerkarte Verweise hinzufügen. Weitere Informationen hierzu finden Sie unter "Verweise" auf Seite 28 .
Ausgaben	Gibt den Speicherort für die vom Mapplet zurückgegebenen Ausgabewerte an. Die Ausgabeparameter müssen in derselben Reihenfolge angegeben werden, in der sie in den Zielpoints des Mapplets angezeigt werden. Geben Sie im Parameter OutputLocation folgende Werte an: <ul style="list-style-type: none"> - data_holder: Gibt den Speicherort für die Werte an, die vom Mapplet zurückgegeben werden. - initialization: Beim Testen der Umwandlung führt die Datenprozessorumwandlung nicht das Mapplet aus, das von der RunMapplet-Aktion aufgerufen wurde. Sie können einen Wert angeben, den Sie als temporäre Ausgabezeichenfolge verwenden, wenn Sie die Umwandlung zur Entwurfszeit testen.

RunParser

Die Aktion **RunParser** führt einen sekundären Parser aus. Die Ausgabe der Aktion **RunParser** wird an die Ausgabe der Hauptkomponente angehängt, die die Aktion aktiviert hat, zum Beispiel ein Parser oder ein Serializer.

Verwenden Sie die Aktion **RunParser**, um den Links in einer HTML-Datei zu folgen und führen Sie mit den Link-Zielen einen sekundären Parser aus. In einem Serializer können Sie die Aktion verwenden, um in der Eingabe vorhandene unstrukturierte Daten zu parsen.

Beachten Sie den folgenden Unterschied zwischen der Aktion **RunParser** und dem Anker **EmbeddedParser**:

- **RunParser** parst eine neue Quelle.
- **EmbeddedParser** parst einen Abschnitt einer vorhandenen Quelle.

Die nachstehende Tabelle beschreibt die Eigenschaften der Aktion **RunParser**:

Eigenschaft	Beschreibung
disabled	Legt fest, ob die Komponente und alle zugehörigen untergeordneten Komponenten vom Skript ignoriert werden. Verwenden Sie diese Eigenschaft zum Testen, Debuggen und Ändern eines Skripts. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none">- Ausgewählt. Das Skript ignoriert die Komponente.- Gelöscht. Das Skript wendet die Komponente an. Standardmäßig ist die Eigenschaft deaktiviert.
exclude_strings	Definiert die Strings, die im Wert von input_source nicht vorhanden sein dürfen. Wenn eine angegebene Zeichenfolge vorhanden ist, greift die Aktion RunParser weder auf die Quelle zu noch aktiviert sie den zweiten Parser.
include_strings	Definiert Strings, die im Wert von input_source vorhanden sein müssen. Wenn eine angegebene Zeichenfolge nicht vorhanden ist, greift die Aktion RunParser weder auf die Quelle zu noch aktiviert sie den zweiten Parser.
input_source	Definiert einen Datenbehälter, der eines der folgenden Objekte enthält: <ul style="list-style-type: none">- Wenn input_source_as_text ausgewählt ist, enthält input_source einen String.- Wenn input_source_as_text nicht ausgewählt ist, enthält input_source Pfad und Dateiname des Eingabedokuments. Standardwert ist die Systemvariable <i>VarLinkURL</i> .
input_source_as_text	Bestimmt den Datentyp im Datenbehälter input_source . <ul style="list-style-type: none">- Ausgewählt. input_source enthält einen Text-String.- Gelöscht. input_source enthält einen Dateipfad. Standardwert ist "Gelöscht".
name	Eine beschreibende Kennzeichnung für die Komponente. Diese Kennzeichnung wird in der Protokolldatei sowie der Ereignisansicht angezeigt. Ermitteln Sie anhand der Eigenschaft name , welche Komponente das Ereignis verursacht hat.
next_Parser	Der Name des auszuführenden Parsers. Ein wiederholter Aufruf ein und desselben Parsers ist zulässig.

Eigenschaft	Beschreibung
on_fail	<p>Die Aktion, die beim Ausfall der Komponente durchgeführt wird. Sie können eine der folgenden Optionen auswählen:</p> <ul style="list-style-type: none"> - Gelöscht. Es wird keine Aktion ausgeführt. - CustomLog. Es wird in das Benutzerprotokoll geschrieben. - LogError. Eine Fehlermeldung wird in das Engine-Protokoll geschrieben. - LogInfo. Eine Informationsmeldung wird in das Engine-Protokoll geschrieben. - LogWarning. Eine Warnmeldung wird in das Engine-Protokoll geschrieben. - NotifyFailure. Eine Mitteilung wird gesendet. <p>Die Standardoption lautet „Gelöscht“. Weitere Informationen über die Behandlung von Komponentengefehlern finden Sie in "Fehlerbehandlung" auf Seite 410.</p>
optional	<p>Legt fest, ob ein Komponentengefehler den Ausfall der übergeordneten Komponente verursacht. Sie können eine der folgenden Optionen auswählen:</p> <ul style="list-style-type: none"> - Ausgewählt. Ein Komponentengefehler führt nicht zum Ausfall der übergeordneten Komponente. - Gelöscht. Ein Komponentengefehler verursacht den Ausfall der übergeordneten Komponente. <p>Standardwert ist "Gelöscht". Weitere Informationen über Komponentengefehler finden Sie in "Fehlerbehandlung" auf Seite 410.</p>
phase	<p>Legt den Zeitpunkt fest, an dem das Skript die Komponente verarbeitet. Sie können eine der folgenden Optionen auswählen:</p> <ul style="list-style-type: none"> - Anfangsphase. Das Skript verarbeitet die Komponente während der Anfangsphase. - Hauptphase. Das Skript verarbeitet die Komponente während der Hauptphase. - Endphase. Das Skript verarbeitet die Komponente während der Endphase. <p>Weitere Informationen hierzu finden Sie unter "So sucht der Parser Anker" auf Seite 211. Standardwert ist "Hauptphase".</p>
pre_processor	<p>Definiert einen Dokumentprozessor, der nach dem im zugeordneten AdditionalInputPort > pre_processor definierten Dokumentprozessor auf die Quelle angewendet wird.</p>
remark	<p>Ein benutzerdefinierter Kommentar, der den Zweck oder die Aktion der Komponente beschreibt.</p>
retries	<p>Die Anzahl der erneuten Versuche, wenn die Anfrage fehlschlägt. Standardwert ist 0.</p>
seconds_to_wait	<p>Definiert das Intervall zwischen den Versuchen in Sekunden. Standardwert ist 60.</p>

Beispiel

Eine HTML-Datei enthält einen Link zu einer zweiten Datei. Ein **Content**-Anker speichert den Dateipfad des Linkzieles in der Systemvariablen *VarLinkURL*. Die Aktion **RunParser** greift auf die Zielfeile zu und führt daran einen sekundären Parser aus.

Ein weiteres Beispiel: Der Hauptparser enthält einen **Alternatives**-Anker, der einen sekundären Parser entsprechend dem Text im Quelldokument auswählt. Weitere Informationen hierzu finden Sie im Abschnitt ["Alternatives" auf Seite 217](#).

RunPCWebService

Die Aktion **RunPCWebService** führt ein PowerCenter-Mapplet über eine Datenprozessorumwandlung aus. Die Ausgabe von **RunPCWebService** wird in den Datenbehälter gelesen, der in der Aktion **RunPCWebService** angegeben wurde.

Hinweis: Die Aktion **RunPCWebService** wird verwendet, um passive Mapplets aufzurufen.

Die Ausgabeparameter müssen in derselben Reihenfolge angegeben werden, in der sie in den Zielports des Mapplets angezeigt werden.

In der folgenden Tabelle werden die Eigenschaften der Aktion **RunPCWebService** beschrieben:

Eigenschaft	Beschreibung
wsdl	Geben Sie die WSDL für den Webservice an, der ausgeführt werden soll.
operationName	Gibt den auszuführenden Vorgang an. Wählen Sie einen Vorgang aus.
Eingaben	Gibt die an das Mapplet zu übergebenden Eingabewerte an. Die Eingabeparameter müssen in derselben Reihenfolge angegeben werden, in der sie in den Quellports des Mapplets angezeigt werden.
Ausgaben	Gibt den Speicherort für die vom Mapplet zurückgegebenen Ausgabewerte an. Die Ausgabeparameter müssen in derselben Reihenfolge angegeben werden, in der sie in den Zielports des Mapplets angezeigt werden.

RunSerializer

Die Aktion **RunSerializer** führt einen Serializer als Teilkomponente eines Parsers, Mappers oder Serializers aus. Die Ausgabe des Serializers wird in einem Datenbehälter gespeichert.

Die nachstehende Tabelle beschreibt die Eigenschaften der Aktion **RunSerializer**:

Eigenschaft	Beschreibung
disabled	Legt fest, ob die Komponente und alle zugehörigen untergeordneten Komponenten vom Skript ignoriert werden. Verwenden Sie diese Eigenschaft zum Testen, Debuggen und Ändern eines Skripts. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none"> - Ausgewählt. Das Skript ignoriert die Komponente. - Gelöscht. Das Skript wendet die Komponente an. Standardmäßig ist die Eigenschaft deaktiviert.
input	Definiert einen Datenbehälter, der XML-Text enthält, an dem der Serializer ausgeführt wird. Der Datenbehälter muss einen einfachen Datentyp haben, zum Beispiel <code>xs:string</code> . Der Wert des Strings kann XML-Text beliebiger Komplexität sein. Informationen zum Ausführen eines Serializers an einem Datenbehälter komplexen Typs finden Sie im Abschnitt "EmbeddedSerializer" auf Seite 352 . Wenn Sie diese Eigenschaft weglassen, verwendet der Serializer die im Gültigkeitsbereich der Aktion vorhandenen Datenbehälter. Ist die Aktion beispielsweise in einen Parser geschachtelt, wird der Serializer an der Ausgabe des Parsers ausgeführt. Befindet sich die Aktion in einer Group , wird er an der Ausgabe der Group ausgeführt.
name	Eine beschreibende Kennzeichnung für die Komponente. Diese Kennzeichnung wird in der Protokolldatei sowie der Ereignisansicht angezeigt. Ermitteln Sie anhand der Eigenschaft name , welche Komponente das Ereignis verursacht hat.

Eigenschaft	Beschreibung
on_fail	<p>Die Aktion, die beim Ausfall der Komponente durchgeführt wird. Sie können eine der folgenden Optionen auswählen:</p> <ul style="list-style-type: none"> - Gelöscht. Es wird keine Aktion ausgeführt. - CustomLog. Es wird in das Benutzerprotokoll geschrieben. - LogError. Eine Fehlermeldung wird in das Engine-Protokoll geschrieben. - LogInfo. Eine Informationsmeldung wird in das Engine-Protokoll geschrieben. - LogWarning. Eine Warnmeldung wird in das Engine-Protokoll geschrieben. - NotifyFailure. Eine Mitteilung wird gesendet. <p>Die Standardoption lautet „Gelöscht“. Weitere Informationen über die Behandlung von Komponentengefehlern finden Sie in "Fehlerbehandlung" auf Seite 410.</p>
optional	<p>Legt fest, ob ein Komponentengefehle den Ausfall der übergeordneten Komponente verursacht. Sie können eine der folgenden Optionen auswählen:</p> <ul style="list-style-type: none"> - Ausgewählt. Ein Komponentengefehle führt nicht zum Ausfall der übergeordneten Komponente. - Gelöscht. Ein Komponentengefehle verursacht den Ausfall der übergeordneten Komponente. <p>Standardwert ist "Gelöscht". Weitere Informationen über Komponentengefehle finden Sie in "Fehlerbehandlung" auf Seite 410.</p>
output	Definiert einen Datenbehälter für die Serializer-Ausgabe.
phase	<p>Legt den Zeitpunkt fest, an dem das Skript die Komponente verarbeitet. Sie können eine der folgenden Optionen auswählen:</p> <ul style="list-style-type: none"> - Anfangsphase. Das Skript verarbeitet die Komponente während der Anfangsphase. - Hauptphase. Das Skript verarbeitet die Komponente während der Hauptphase. - Endphase. Das Skript verarbeitet die Komponente während der Endphase. <p>Weitere Informationen hierzu finden Sie unter "So sucht der Parser Anker" auf Seite 211. Standardwert ist "Hauptphase".</p>
remark	Ein benutzerdefinierter Kommentar, der den Zweck oder die Aktion der Komponente beschreibt.
serializer	Definiert einen Serializer. Wählen Sie den Namen einer vorhandenen Serializer -Komponente oder definieren Sie eine Serializer -Komponente an dieser Position des Skripts. Weitere Informationen hierzu finden Sie unter "Die Komponente Serializer: Referenz" auf Seite 345 .

Online-Beispiel

Ein Online-Beispiel zu dieser Aktion befindet sich im Projekt `samples\Projects\RunSerializer\RunSerializer.cmw`.

Um zu sehen, wie dieses Beispiel funktioniert, legen Sie `MainParser` als Startkomponente fest, und führen Sie diesen aus. `MainParser` enthält eine `RepeatingGroup`, die Namenspaare parst und in Variablen speichert. Nach jeder Iteration führt die `RepeatingGroup` die Aktion `RunSerializer` aus, die die Variablen mit vordefiniertem Text verkettet. Die Ausgabe der Aktion wird in einem XML-Element gespeichert, das zur Ausgabe des Parsers hinzugefügt wird.

RunXMap

Der Aktion **RunXMap** führt ein XMap-Objekt als Teilkomponente eines Parsers, Mappers oder Serializers aus.

In der folgenden Tabelle werden die Eigenschaften der Aktion **RunXMap** beschrieben:

Eigenschaft	Beschreibung
disabled	<p>Legt fest, ob die Komponente und alle zugehörigen untergeordneten Komponenten vom Skript ignoriert werden. Verwenden Sie diese Eigenschaft zum Testen, Debuggen und Ändern eines Skripts. Sie können eine der folgenden Optionen auswählen:</p> <ul style="list-style-type: none"> - Ausgewählt. Das Skript ignoriert die Komponente. - Gelöscht. Das Skript wendet die Komponente an. <p>Standardmäßig ist die Eigenschaft deaktiviert.</p>
input	<p>Die Eingabe für das XMap-Objekt. Sie können eine der folgenden Optionen auswählen:</p> <ul style="list-style-type: none"> - Datenbehälter. Verwenden Sie einen Komplextyp, der dem Typ der XMap-Eingabe entspricht. - XML-String. Der XML-String ist die Eingabe oder ein Einfachtyp-Datenbehälter wie xs:string. Der XML-Root-Typ muss dem XMap-Eingabetyp entsprechen. <p>Wenn Sie diese Eigenschaft nicht verwenden, verwendet das XMap-Objekt die Standardeingabe.</p>
name	<p>Eine beschreibende Kennzeichnung für die Komponente. Diese Kennzeichnung wird in der Protokolldatei sowie der Ereignisansicht angezeigt. Ermitteln Sie anhand der Eigenschaft name, welche Komponente das Ereignis verursacht hat.</p>
on_fail	<p>Die Aktion, die beim Ausfall der Komponente durchgeführt wird. Sie können eine der folgenden Optionen auswählen:</p> <ul style="list-style-type: none"> - Gelöscht. Es wird keine Aktion ausgeführt. - CustomLog. Es wird in das Benutzerprotokoll geschrieben. - LogError. Eine Fehlermeldung wird in das Engine-Protokoll geschrieben. - LogInfo. Eine Informationsmeldung wird in das Engine-Protokoll geschrieben. - LogWarning. Eine Warnmeldung wird in das Engine-Protokoll geschrieben. - NotifyFailure. Eine Mitteilung wird gesendet. <p>Die Standardoption lautet „Gelöscht“. Weitere Informationen über die Behandlung von Komponentenfehlern finden Sie in "Fehlerbehandlung" auf Seite 410.</p>
optional	<p>Legt fest, ob ein Komponentenfehler den Ausfall der übergeordneten Komponente verursacht. Sie können eine der folgenden Optionen auswählen:</p> <ul style="list-style-type: none"> - Ausgewählt. Ein Komponentenfehler führt nicht zum Ausfall der übergeordneten Komponente. - Gelöscht. Ein Komponentenfehler verursacht den Ausfall der übergeordneten Komponente. <p>Standardwert ist "Gelöscht". Weitere Informationen über Komponentenfehler finden Sie in "Fehlerbehandlung" auf Seite 410.</p>
output	<p>Ein Datenbehälter eines komplexen Typs, der dem XMap-Ausgabebetyp entsprechen muss. Wenn Sie diese Eigenschaft auslassen, schreibt das XMap-Objekt in die festgelegte Standardausgabe.</p>
phase	<p>Legt den Zeitpunkt fest, an dem das Skript die Komponente verarbeitet. Sie können eine der folgenden Optionen auswählen:</p> <ul style="list-style-type: none"> - Anfangsphase. Das Skript verarbeitet die Komponente während der Anfangsphase. - Hauptphase. Das Skript verarbeitet die Komponente während der Hauptphase. - Endphase. Das Skript verarbeitet die Komponente während der Endphase. <p>Weitere Informationen hierzu finden Sie unter "So sucht der Parser Anker" auf Seite 211. Standardwert ist „Hauptphase“.</p>
remark	<p>Ein benutzerdefinierter Kommentar, der den Zweck oder die Aktion der Komponente beschreibt.</p>
xmap	<p>Name der XMap. Sie können den Namen eines vorhandenen XMap-Objekts auswählen.</p>

SetValue

Die Aktion **SetValue** legt vordefinierten Inhalt in einem Datenbehälter ab. Wenn es sich bei dem Datenbehälter um einen Einzelinstanz-Datenbehälter handelt, wird vorhandener Inhalt ersetzt. Wenn es sich bei dem Datenbehälter um einen Mehrfachinstanz-Datenbehälter handelt, wird der definierte Inhalt am Ende angehängt.

In der folgenden Tabelle werden die Eigenschaften der Aktion **SetValue** beschrieben:

Eigenschaft	Beschreibung
data_holder	Definiert einen Datenbehälter, der die Ausgabe empfängt.
disabled	Legt fest, ob die Komponente und alle zugehörigen untergeordneten Komponenten vom Skript ignoriert werden. Verwenden Sie diese Eigenschaft zum Testen, Debuggen und Ändern eines Skripts. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none">- Ausgewählt. Das Skript ignoriert die Komponente.- Gelöscht. Das Skript wendet die Komponente an. Standardmäßig ist die Eigenschaft deaktiviert.
name	Eine beschreibende Kennzeichnung für die Komponente. Diese Kennzeichnung wird in der Protokolldatei sowie der Ereignisansicht angezeigt. Ermitteln Sie anhand der Eigenschaft name , welche Komponente das Ereignis verursacht hat.
on_fail	Die Aktion, die beim Ausfall der Komponente durchgeführt wird. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none">- Gelöscht. Es wird keine Aktion ausgeführt.- CustomLog. Es wird in das Benutzerprotokoll geschrieben.- LogError. Eine Fehlermeldung wird in das Engine-Protokoll geschrieben.- LogInfo. Eine Informationsmeldung wird in das Engine-Protokoll geschrieben.- LogWarning. Eine Warnmeldung wird in das Engine-Protokoll geschrieben.- NotifyFailure. Eine Mitteilung wird gesendet. Die Standardoption lautet „Gelöscht“. Weitere Informationen über die Behandlung von Komponentenfehlern finden Sie in "Fehlerbehandlung" auf Seite 410 .
optional	Legt fest, ob ein Komponentenfehler den Ausfall der übergeordneten Komponente verursacht. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none">- Ausgewählt. Ein Komponentenfehler führt nicht zum Ausfall der übergeordneten Komponente.- Gelöscht. Ein Komponentenfehler verursacht den Ausfall der übergeordneten Komponente. Standardwert ist "Gelöscht". Weitere Informationen über Komponentenfehler finden Sie in "Fehlerbehandlung" auf Seite 410 .
phase	Legt den Zeitpunkt fest, an dem das Skript die Komponente verarbeitet. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none">- Anfangsphase. Das Skript verarbeitet die Komponente während der Anfangsphase.- Hauptphase. Das Skript verarbeitet die Komponente während der Hauptphase.- Endphase. Das Skript verarbeitet die Komponente während der Endphase. Weitere Informationen hierzu finden Sie unter "So sucht der Parser Anker" auf Seite 211 . Standardwert ist "Hauptphase".
quote	Definiert einen String für den Inhalt des Datenbehälters.
remark	Ein benutzerdefinierter Kommentar, der den Zweck oder die Aktion der Komponente beschreibt.
transformers	Definiert eine Liste mit Transformern, die auf den Inhalt angewendet werden, bevor er im Datenbehälter gespeichert wird.

Sortieren

Die Aktion **Sort** sortiert die Instanzen eines Mehrfachinstanz-Datenbehälters. Die Ausgabe wird im ursprünglichen Inhalt des Datenbehälters gespeichert. Bei der Sortierung wird die Groß- und Kleinschreibung beachtet.

Sofern Sie die Aktion an einem XML-Element ausführen, das Attribute oder geschachtelte Elemente enthält, können Sie sie als Sortierschlüssel verwenden.

In der folgenden Tabelle werden die Eigenschaften der Aktion **Sort** beschrieben:

Eigenschaft	Beschreibung
by_fields	Definiert eine Liste mit Sortierschlüsseln in absteigender Reihenfolge hinsichtlich ihres Vorrangs. Wählen Sie für jedes Feld den Datenbehälter und eine aufsteigende (ascending) oder absteigende (descending) Sortierung aus. Sie können den Mehrfachinstanz-Datenbehälter selbst oder eines seiner geschachtelten Elemente oder Attribute auswählen. Für eine numerische Sortierung muss der Sortierschlüssel einen numerischen Datentyp wie <code>xs:integer</code> aufweisen.
disabled	Legt fest, ob die Komponente und alle zugehörigen untergeordneten Komponenten vom Skript ignoriert werden. Verwenden Sie diese Eigenschaft zum Testen, Debuggen und Ändern eines Skripts. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none">- Ausgewählt. Das Skript ignoriert die Komponente.- Gelöscht. Das Skript wendet die Komponente an. Standardmäßig ist die Eigenschaft deaktiviert.
name	Eine beschreibende Kennzeichnung für die Komponente. Diese Kennzeichnung wird in der Protokolldatei sowie der Ereignisansicht angezeigt. Ermitteln Sie anhand der Eigenschaft name , welche Komponente das Ereignis verursacht hat.
on_fail	Die Aktion, die beim Ausfall der Komponente durchgeführt wird. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none">- Gelöscht. Es wird keine Aktion ausgeführt.- CustomLog. Es wird in das Benutzerprotokoll geschrieben.- LogError. Eine Fehlermeldung wird in das Engine-Protokoll geschrieben.- LogInfo. Eine Informationsmeldung wird in das Engine-Protokoll geschrieben.- LogWarning. Eine Warnmeldung wird in das Engine-Protokoll geschrieben.- NotifyFailure. Eine Mitteilung wird gesendet. Die Standardoption lautet „Gelöscht“. Weitere Informationen über die Behandlung von Komponentenfehlern finden Sie in "Fehlerbehandlung" auf Seite 410 .
phase	Legt den Zeitpunkt fest, an dem das Skript die Komponente verarbeitet. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none">- Anfangsphase. Das Skript verarbeitet die Komponente während der Anfangsphase.- Hauptphase. Das Skript verarbeitet die Komponente während der Hauptphase.- Endphase. Das Skript verarbeitet die Komponente während der Endphase. Weitere Informationen hierzu finden Sie unter "So sucht der Parser Anker" auf Seite 211 . Standardwert ist "Hauptphase".
recurring_element	Definiert einen zu sortierenden Mehrfachinstanzen-Datenbehälter.
remark	Ein benutzerdefinierter Kommentar, der den Zweck oder die Aktion der Komponente beschreibt.

Einschränkung

Die Aktion **Sort** kann nicht verwendet werden, wenn für den Mehrfachinstanz-Datenbehälter ein **Key** definiert ist. Weitere Informationen hierzu finden Sie unter ["Überblick über Locator, Keys und Indexierung" auf Seite 369](#).

ValidateValue

Die Aktion **ValidateValue** validiert XML-Daten gemäß einer Gruppe von Regeln, die in einem Validierungsregelobjekt definiert wurden. Wenn die Daten die Regeln verletzen, speichert die Aktion einen Validierungsbericht in einem Datenbehälter.

Die Eingabe der Aktion ist ein Datenbehälter. Wenn der Datenbehälter das Stammelement eines XML-Zweiges ist, analysiert die Aktion den gesamten Zweig.

In der folgenden Tabelle werden die Eigenschaften der Aktion **ValidateValue** beschrieben:

Eigenschaft	Beschreibung
disabled	Legt fest, ob die Komponente und alle zugehörigen untergeordneten Komponenten vom Skript ignoriert werden. Verwenden Sie diese Eigenschaft zum Testen, Debuggen und Ändern eines Skripts. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none">- Ausgewählt. Das Skript ignoriert die Komponente.- Gelöscht. Das Skript wendet die Komponente an. Standardmäßig ist die Eigenschaft deaktiviert.
errors_found	Definiert einen Datenbehälter, der die Anzahl an Validierungsregelverletzungen in der Eingabe erfasst.
errors_output	Definiert einen Datenbehälter, in dem die Aktion den Fehlerbericht zur XML-Validierung speichert. Wenn der Datenbehälter den Typ <code>xs:string</code> aufweist, besteht die Ausgabe aus einer Zeichenfolge mit XML-Tags. Wenn der Datenbehälter den Typ <code>code:validationErrors</code> aufweist, besteht die Ausgabe aus einer Struktur mit verschachtelten Datenbehältern.
input	Definiert den Eingabedatenbehälter, der von der Aktion hinsichtlich Konformität mit den Validierungsregeln analysiert wird.
Name	Eine beschreibende Kennzeichnung für die Komponente. Diese Kennzeichnung wird in der Protokolldatei sowie der Ereignisansicht angezeigt. Ermitteln Sie anhand der Eigenschaft name , welche Komponente das Ereignis verursacht hat.
optional	Legt fest, ob ein Komponentenfehler den Ausfall der übergeordneten Komponente verursacht. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none">- Ausgewählt. Ein Komponentenfehler führt nicht zum Ausfall der übergeordneten Komponente.- Gelöscht. Ein Komponentenfehler verursacht den Ausfall der übergeordneten Komponente. Standardwert ist "Gelöscht". Weitere Informationen über Komponentenfehler finden Sie in "Fehlerbehandlung" auf Seite 410 .
Phase	Legt den Zeitpunkt fest, an dem das Skript die Komponente verarbeitet. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none">- Anfangsphase. Das Skript verarbeitet die Komponente während der Anfangsphase.- Hauptphase. Das Skript verarbeitet die Komponente während der Hauptphase.- Endphase. Das Skript verarbeitet die Komponente während der Endphase. Weitere Informationen hierzu finden Sie unter "So sucht der Parser Anker" auf Seite 211 .
Bemerkung	Ein benutzerdefinierter Kommentar, der den Zweck oder die Aktion der Komponente beschreibt.

WriteValue

Die Aktion **WriteValue** schreibt den Wert eines Datenbehälters in Speicherorte wie Dateien oder Zeichenfolgen-Datenbehälter.

Wenn der Eingabe-Datenbehälter ein XML-Element ist, schreibt die Aktion sowohl das Element als auch alle geschachtelten Elemente und Attribute.

In der folgenden Tabelle werden die Eigenschaften der Aktion **WriteValue** beschrieben:

Eigenschaft	Beschreibung
disabled	Legt fest, ob die Komponente und alle zugehörigen untergeordneten Komponenten vom Skript ignoriert werden. Verwenden Sie diese Eigenschaft zum Testen, Debuggen und Ändern eines Skripts. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none">- Ausgewählt. Das Skript ignoriert die Komponente.- Gelöscht. Das Skript wendet die Komponente an. Standardmäßig ist die Eigenschaft deaktiviert.
input	Definiert einen Datenbehälter, aus dem geschrieben wird.
name	Eine beschreibende Kennzeichnung für die Komponente. Diese Kennzeichnung wird in der Protokolldatei sowie der Ereignisansicht angezeigt. Ermitteln Sie anhand der Eigenschaft name , welche Komponente das Ereignis verursacht hat.
no_tags	Legt fest, ob das Ergebnis von XML-Tags eingeschlossen wird. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none">- Ausgewählt. Es werden keine XML-Tags verwendet. Dies ist nur dann sinnvoll, wenn input ein einfacher Datenbehälter ohne geschachtelte Elemente oder Attribute ist.- Gelöscht. Das Ergebnis wird von XML-Tags eingeschlossen. Dies ist die Standardeinstellung. Standardwert ist "Gelöscht".
on_fail	Die Aktion, die beim Ausfall der Komponente durchgeführt wird. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none">- Gelöscht. Es wird keine Aktion ausgeführt.- CustomLog. Es wird in das Benutzerprotokoll geschrieben.- LogError. Eine Fehlermeldung wird in das Engine-Protokoll geschrieben.- LogInfo. Eine Informationsmeldung wird in das Engine-Protokoll geschrieben.- LogWarning. Eine Warnmeldung wird in das Engine-Protokoll geschrieben.- NotifyFailure. Eine Mitteilung wird gesendet. Die Standardoption lautet „Gelöscht“. Weitere Informationen über die Behandlung von Komponentenfehlern finden Sie in "Fehlerbehandlung" auf Seite 410 .
optional	Legt fest, ob ein Komponentenfehler den Ausfall der übergeordneten Komponente verursacht. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none">- Ausgewählt. Ein Komponentenfehler führt nicht zum Ausfall der übergeordneten Komponente.- Gelöscht. Ein Komponentenfehler verursacht den Ausfall der übergeordneten Komponente. Standardwert ist "Gelöscht". Weitere Informationen über Komponentenfehler finden Sie in "Fehlerbehandlung" auf Seite 410 .
output	Definiert die Ausgabeposition. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none">- OutputDataHolder. Schreibt in einen Datenbehälter.- OutputFile. Schreibt in eine Datei. Wählen Sie Synchronisieren zum Sperren der Datei aus, um Daten an dieselbe Datei anzuhängen.- OutputPort. Definiert den Namen eines AdditionalOutputPort, in dem die Daten geschrieben werden.- ResultFile. Schreibt in die Standardergebnisdatei der Umwandlung.- StandardErrorLog. Schreibt in das Benutzerprotokoll. Weitere Informationen hierzu finden Sie unter "Fehlerbehandlung" auf Seite 410. Die Standardoption lautet „ResultFile“. Weitere Informationen zu diesen Optionen finden Sie unter "Aktions-Teilkomponenten: Referenz" auf Seite 338 .

Eigenschaft	Beschreibung
phase	<p>Legt den Zeitpunkt fest, an dem das Skript die Komponente verarbeitet. Sie können eine der folgenden Optionen auswählen:</p> <ul style="list-style-type: none"> - Anfangsphase. Das Skript verarbeitet die Komponente während der Anfangsphase. - Hauptphase. Das Skript verarbeitet die Komponente während der Hauptphase. - Endphase. Das Skript verarbeitet die Komponente während der Endphase. <p>Weitere Informationen hierzu finden Sie unter "So sucht der Parser Anker" auf Seite 211. Standardwert ist "Hauptphase".</p>
remark	Ein benutzerdefinierter Kommentar, der den Zweck oder die Aktion der Komponente beschreibt.
transformers	Definiert eine Liste mit Transformern, die den Wert vor dem Schreiben bearbeiten. Die Eingabe für die Transformer ist der vollständige Inhalt des Datenbehälters input , einschließlich der XML-Tags.

Online-Beispiel

Ein Online-Beispiel zu dieser Aktion befindet sich im Projekt `samples\Projects\Splitter\Splitter.cmw`.

Das Beispiel veranschaulicht, wie Sie eine Datei in zwei Dateien zerlegen. Ein Parser ruft mit Hilfe einer **RepeatingGroup** die Datensätze einer HL7-Datei ab. Mit der Aktion **Map** erzeugt er für jeden Datensatz einen eindeutigen Dateinamen, und mit der Aktion **WriteValue** schreibt er die Datensätze in die Dateien. Die Ausgabedateien `MyOutput1.txt` und `MyOutput2.txt` sind im Ordner `Results` des Projekts gespeichert.

Hinweis: Sie können zum Zerlegen umfangreicher Eingaben auch einen Streamer verwenden. Weitere Informationen hierzu finden Sie unter ["Übersicht über Streamer" auf Seite 387](#).

XSLTMap

Die Aktion **XSLTMap** führt eine XSLT-Umwandlung aus. Die Eingabe und die Ausgabe sind Zweige eines XML-Dokuments. Sie können beispielsweise die Ausgabe eines Parsers oder die Eingabe eines Serializers sein.

In der folgenden Tabelle werden die Eigenschaften der Aktion **XSLTMap** beschrieben:

Eigenschaft	Beschreibung
disabled	<p>Legt fest, ob die Komponente und alle zugehörigen untergeordneten Komponenten vom Skript ignoriert werden. Verwenden Sie diese Eigenschaft zum Testen, Debuggen und Ändern eines Skripts. Sie können eine der folgenden Optionen auswählen:</p> <ul style="list-style-type: none"> - Ausgewählt. Das Skript ignoriert die Komponente. - Gelöscht. Das Skript wendet die Komponente an. <p>Standardmäßig ist die Eigenschaft deaktiviert.</p>
input	Definiert einen Datenbehälter, der das umzuwandelnde XML-Element enthält.
name	Eine beschreibende Kennzeichnung für die Komponente. Diese Kennzeichnung wird in der Protokolldatei sowie der Ereignisansicht angezeigt. Ermitteln Sie anhand der Eigenschaft name , welche Komponente das Ereignis verursacht hat.

Eigenschaft	Beschreibung
on_fail	<p>Die Aktion, die beim Ausfall der Komponente durchgeführt wird. Sie können eine der folgenden Optionen auswählen:</p> <ul style="list-style-type: none"> - Gelöscht. Es wird keine Aktion ausgeführt. - CustomLog. Es wird in das Benutzerprotokoll geschrieben. - LogError. Eine Fehlermeldung wird in das Engine-Protokoll geschrieben. - LogInfo. Eine Informationsmeldung wird in das Engine-Protokoll geschrieben. - LogWarning. Eine Warnmeldung wird in das Engine-Protokoll geschrieben. - NotifyFailure. Eine Mitteilung wird gesendet. <p>Die Standardoption lautet „Gelöscht“. Weitere Informationen über die Behandlung von Komponentenfehlern finden Sie in "Fehlerbehandlung" auf Seite 410.</p>
optional	<p>Legt fest, ob ein Komponentenfehler den Ausfall der übergeordneten Komponente verursacht. Sie können eine der folgenden Optionen auswählen:</p> <ul style="list-style-type: none"> - Ausgewählt. Ein Komponentenfehler führt nicht zum Ausfall der übergeordneten Komponente. - Gelöscht. Ein Komponentenfehler verursacht den Ausfall der übergeordneten Komponente. <p>Standardwert ist "Gelöscht". Weitere Informationen über Komponentenfehler finden Sie in "Fehlerbehandlung" auf Seite 410.</p>
output	Definiert einen Datenbehälter, in dem die Ausgabe gespeichert wird.
phase	<p>Legt den Zeitpunkt fest, an dem das Skript die Komponente verarbeitet. Sie können eine der folgenden Optionen auswählen:</p> <ul style="list-style-type: none"> - Anfangsphase. Das Skript verarbeitet die Komponente während der Anfangsphase. - Hauptphase. Das Skript verarbeitet die Komponente während der Hauptphase. - Endphase. Das Skript verarbeitet die Komponente während der Endphase. <p>Weitere Informationen hierzu finden Sie unter "So sucht der Parser Anker" auf Seite 211. Standardwert ist "Hauptphase".</p>
remark	Ein benutzerdefinierter Kommentar, der den Zweck oder die Aktion der Komponente beschreibt.
xslt_file	Definiert die XSLT-Datei.

Beispiel

Der folgende XML-Code ist das Ergebnis eines Parsers:

```
<Person>
  <First>Ron</First>
  <Last>Lehrer</Last>
</Person>
```

Mit einer geeigneten XSLT-Datei können Sie die **XSLTMap**-Aktion verwenden, um dies umzuwandeln in:

```
<Person Name="Lehrer, Ron" />
```

Aktions-Teilkomponenten: Referenz

Aktion-Teilkomponenten dienen als Werte bestimmter Eigenschaften von Aktionen.

OutputDataHolder

Die Teilkomponente **OutputDataHolder** leitet die Ausgabe in einen Datenbehälter um. Sie wird in der Eigenschaft **output** der Aktion **WriteValue** verwendet.

Die nachstehende Tabelle beschreibt die Eigenschaften der Teilkomponente **OutputDataHolder**:

Eigenschaft	Beschreibung
data_holder	Definiert den Ausgabe-Datenbehälter.
name	Eine beschreibende Kennzeichnung für die Komponente. Diese Kennzeichnung wird in der Protokolldatei sowie der Ereignisansicht angezeigt. Ermitteln Sie anhand der Eigenschaft name , welche Komponente das Ereignis verursacht hat.
remark	Ein benutzerdefinierter Kommentar, der den Zweck oder die Aktion der Komponente beschreibt.
transformers	Definiert eine Folge von Transformern, die den Strom vor dem Schreiben bearbeiten.

OutputFile

Die Teilkomponente **OutputFile** lenkt die Ausgabe in eine Datei. Sie wird in der Eigenschaft **output** der Aktionen **DumpValues** und **WriteValue** verwendet.

Die nachstehende Tabelle beschreibt die Eigenschaften der Teilkomponente **OutputFile**:

Eigenschaft	Beschreibung
append	Legt fest, ob die Daten an den bestehenden Inhalt der Datei angehängt werden. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none">- Ausgewählt. Die Daten werden an den bestehenden Inhalt angehängt.- Gelöscht. Die Daten überschreiben den bestehenden Inhalt. Standardwert ist "Gelöscht".
file	Definiert einen String oder Datenbehälter, der den Pfad und den Dateinamen definiert. Der Pfad kann absolut oder relativ sein. Wenn der Pfad relativ ist, löst das Skript den Pfad relativ zum Ausgabeordner der Umwandlung auf.
name	Eine beschreibende Kennzeichnung für die Komponente. Diese Kennzeichnung wird in der Protokolldatei sowie der Ereignisansicht angezeigt. Ermitteln Sie anhand der Eigenschaft name , welche Komponente das Ereignis verursacht hat.
remark	Ein benutzerdefinierter Kommentar, der den Zweck oder die Aktion der Komponente beschreibt.

ResultFile

Die Teilkomponente **ResultFile** gibt an, dass die Ausgabe die normale Ausgabedatei einer Datenprozessor-Umwandlung ist. Sie wird in den Aktionen **DumpValues** und **WriteValue** verwendet.

StandardErrorLog

Die Unterkomponente **StandardErrorLog** gibt an, dass die Ausgabe in das Benutzerprotokoll geschrieben wird.

KAPITEL 19

Serializer

Dieses Kapitel umfasst die folgenden Themen:

- [Übersicht über Serializer, 340](#)
- [Serialisierungsanker, 343](#)
- [Standardeigenschaften von Serializern, 345](#)
- [Die Komponente Serializer: Referenz, 345](#)
- [Serialisierungsanker-Komponenten: Referenz, 347](#)

Übersicht über Serializer

Ein Serializer wandelt eine XML- oder JSON-Datei in ein Ausgabedokument um, das ein beliebiges Format aufweisen kann. Die Serialisierung ist die Umkehrung des Parsens. Die Ausgabe eines Serializers kann zum Beispiel ein Textdokument, ein HTML-Dokument oder auch ein anderes XML-Dokument sein.

Es gibt mehrere Möglichkeiten, einen Serializer zu erstellen:

- Umkehren der Konfiguration eines vorhandenen Parsers
- Bearbeiten des Skripts und Einfügen einer **Serializer**-Komponente

Sie können einen Parser umkehren und das Skript des so entstandenen Serializers bearbeiten.

Im Allgemeinen ist die Erstellung eines Serializers einfacher als die eines Parsers, da die XML- oder JSON-Eingabe vollständig strukturiert ist. Aufgrund dieser Struktur ist es einfacher, die erforderlichen Daten zu ermitteln und der Reihe nach in die Ausgabe zu schreiben. Ein Parser muss hingegen möglicherweise eine nicht oder nur teilweise strukturierte Eingabe verarbeiten. Diese Aufgabe ist häufig sehr komplex.

Die Hauptkomponenten, die in Serializer geschachtelt sind, werden als Serialisierungsanker bezeichnet. Die Aufgabe der Serialisierungsanker besteht darin, die XML- oder JSON-Daten zu ermitteln und in die Ausgabe zu schreiben. Serialisierungsanker entsprechen im Wesentlichen den in Parsern verwendeten Ankern, arbeiten jedoch in umgekehrter Richtung.

Steuern der Arbeitsweise des Befehls „Serializer erstellen“

Wenn Sie den Befehl **Serializer erstellen** ausführen, wandelt das Developer-Tool die **Content**-Anker des Parsers in **ContentSerializer**-Serialisierungsanker um.

Standardmäßig wandelt der Befehl sämtlichen übrigen Text in der Beispielquelle in **StringSerializer**-Serialisierungsanker um. Falls der übrige Text aus Textbausteinen besteht, enthält die Ausgabe des Serializers denselben Textbaustein-Text wie die ursprüngliche Beispielquelle.

Angenommen, der Parser wird auf tabulatorbegrenzten Quelldokumenten mit der folgenden Struktur ausgeführt:

```
Name (first and last):<tab>Ron Lehrer
```

Dabei sind die Anker folgendermaßen definiert:

Quelltext	Anker
Name	Marker
(first and last):<tab>	Nicht als Anker markiert
Ron Lehrer	Content

Die XML-Ausgabe des Parsers lautet:

```
<FullName>Ron Lehrer<FullName>
```

Erzeugen Sie nun aus diesem Parser einen Serializer und führen diesen an der folgenden Eingabe aus:

```
<FullName>Larissa Chan<FullName>
```

Die Ausgabe des Serializers lautet:

```
Name (first and last):<tab>Larissa Chan
```

Serialisierungsmodus

Die Beispielquelle kann Text enthalten, der nicht in die Ausgabe des Serializers eingefügt werden soll. In diesem Fall können Sie das Verhalten des Befehls **Serializer erstellen** dahin gehend ändern, dass die **StringSerializer**-Serialisierungsanker nicht generiert werden.

Legen Sie zu diesem Zweck die Eigenschaft **serialization_mode** der **Parser**-Komponente fest. Die möglichen Werte der Eigenschaft **serialization_mode** sind in der folgenden Tabelle aufgeführt:

Wert	Beschreibung
Komplett	Der Befehl Serializer erstellen kopiert den nicht in XML umgewandelten Text in die Konfiguration des Serializers. Dies ist das Standardverhalten.
Outline	Der Befehl Serializer erstellen kopiert nur die Delimiter des nicht in XML umgewandelten Textes in die Konfiguration des Serializers. Unter der Option Outline können Sie die Option use_markers auswählen. Der Befehl „Serializer erstellen“ kopiert dann den gesamten Inhalt der Marker -Anker, aber nur die Delimiter des übrigen nicht in XML umgewandelten Textes.

In der folgenden Tabelle wird die Wirkung der **serialization_mode**-Einstellungen beschrieben.

Serialisierungsmodus	Verhalten	Serializer-Ausgabebeispiel
outline ohne Auswahl von use_markers	Durch den Befehl Serializer erstellen vorgenommene Umwandlungen: <ul style="list-style-type: none"> - Content-Anker in ContentSerializer-Serialisierungsanker - Delimiter des übrigen Textes in der Beispielquelle in StringSerializer-Serialisierungsanker 	<tab>Larissa Chan
outline mit Auswahl von use_markers	Durch den Befehl Serializer erstellen vorgenommene Umwandlungen: <ul style="list-style-type: none"> - Content-Anker in ContentSerializer-Serialisierungsanker - Gesamter Text von Marker-Ankern in StringSerializer-Serialisierungsanker - Delimiter des übrigen Textes in der Beispielquelle in StringSerializer-Serialisierungsanker. 	Name<tab>Larissa Chan
Vollständig.	Durch den Befehl Serializer erstellen vorgenommene Umwandlungen: <ul style="list-style-type: none"> - Content-Anker in ContentSerializer-Serialisierungsanker - Gesamten übrigen Text in der Beispielquelle in StringSerializer-Serialisierungsanker 	Name (first and last):<tab>Larissa Chan

Fehlerbehebung bei automatisch erzeugten Serializern

Häufig können Sie einen automatisch erzeugten Serializer direkt verwenden. Falls erforderlich, können Sie jedoch die automatisch erzeugte Serializer-Konfiguration auch bearbeiten, um eventuelle Einschränkungen oder Probleme zu korrigieren.

Die folgenden Abschnitte erläutern einige typische Situationen, in denen ein Serializer bearbeitet werden muss, sowie die empfohlenen Bearbeitungsschritte.

Wurzeltag

Auf der Registerkarte „XML-Generierung“ der Einstellungen für die Datenprozessor-Umwandlung können Sie das Skript so konfigurieren, dass es ein Stammelement um das im Ausgabeschema definierte Stammelement legt.

Wenn Sie dann den ausgegebenen XML-Code als Eingabe in einen automatisch generierten Serializer verwenden, müssen Sie die Eigenschaft **root_tag** des Serializers auf den Namen des Stammelements setzen, das in den Einstellungen für die XML-Generierung festgelegt ist.

Variablen

Wenn der Parser Zwischenergebnisse in einer Variablen speichert, führt dies bei einem automatisch erzeugten Serializer möglicherweise zu einem Fehler. Um dieses Problem zu lösen, untersuchen Sie die Logik des Serializers, und entfernen Sie gegebenenfalls die Variable.

Weitere Komponenten

Der Befehl „Serializer erstellen“ kehrt die Anker eines Parsers um. Komponenten wie Dokumentprozessoren, Transformer und Aktionen kehrt er nicht um.

Angenommen, ein Parser wandelt mit Hilfe eines `PdfToTxt_4`-Dokumentprozessors PDF-Quelldokumente in Text um. Der Parser enthält Anker, die den Text in XML umwandeln.

Der automatisch erzeugte Serializer wandelt XML wieder in Text um. Er wandelt jedoch nicht den Text in PDF um. Bearbeiten Sie den Serializer und geben Sie einen `XmlToDocument`-Prozessor ein, um eine PDF-Ausgabe zu erhalten.

Ein weiteres Beispiel: Ein Parser fügt mit Hilfe eines `AddString`-Transformers ein Präfix zur Ausgabe eines `Content`-Ankers hinzu. Der automatisch erzeugte Serializer entfernt dieses Präfix nicht. Falls es entfernt werden soll, können Sie eine Komponente, z. B. einen `Replace`-Transformer, einfügen.

Erstellen eines Serializers durch Bearbeiten des Skripts

1. Doppelklicken Sie auf der globalen Ebene des Skripts auf das Symbol `...`, geben Sie einen Namen für den Serializer ein und drücken Sie die **Eingabetaste**.
2. Doppelklicken Sie rechts vom Gleichheitszeichen auf das Auslassungszeichen, wählen Sie einen **Serializer** aus und drücken Sie die **Eingabetaste**.
3. Erweitern Sie den Baum unter der **Serializer**-Komponente. Konfigurieren Sie ihre Eigenschaften nach Bedarf.
4. Fügen Sie ein Schema hinzu, das die XML-Syntax der Serializereingabe definiert.
5. Fügen Sie unter der Zeile **contains** eine Folge von geschachtelten Serialisierungsankern und Aktionen hinzu.
6. Führen Sie den Serializer testweise aus und bearbeiten Sie das Skript nach Bedarf.

Online-Beispiel

Ein Beispiel für einen durch Bearbeiten des Skripts erstellten Serializer ist im Projekt `samples\Projects\ManualSerializer\ManualSerializer.cmw` enthalten. Sie können den Serializer an der Eingabedatei `Example XML of Person.xml` ausführen.

Erstellen eines Serializers in der Aktion RunSerializer

Serializer können nicht nur auf der globalen Ebene definiert werden, sondern auch innerhalb einer `RunSerializer`-Aktion.

Serialisierungsanker

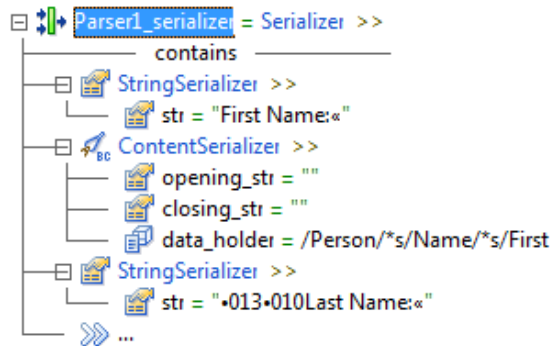
Serialisierungsanker stellen die Hauptkomponenten dar, die Sie in einem Serializer verwenden. Serialisierungsanker entsprechen im Wesentlichen den in Parsern verwendeten Ankern, arbeiten jedoch in umgekehrter Richtung. Anker lesen Daten aus bestimmten Stellen des Quelldokuments aus und schreiben sie in XML. Serialisierungsanker lesen dagegen XML-Daten aus und schreiben sie an bestimmte Stellen in das Ausgabedokument.

Die wichtigsten Serialisierungsanker sind **ContentSerializer** und **StringSerializer**.

- Ein **ContentSerializer** schreibt den Inhalt eines angegebenen Datenbehälters in das Ausgabedokument. Er ist das Gegenstück zu einem **Content**-Anker, der Inhalt aus einem Quelldokument liest.
- Ein **StringSerializer** schreibt einen vordefinierten String in das Ausgabedokument. Er ist das Gegenstück zu einem **Marker**-Anker, der einen vordefinierten String in einem Quelldokument findet.

Serialisierungsanker: Beispiel

Das folgende Beispiel zeigt drei Serialisierungsanker.



Der erste StringSerializer weist den Serializer an, folgenden Text in das Ausgabedokument zu schreiben:

```
First Name:<tab>
```

Der ContentSerializer schreibt den Wert des Elements `Person/Name/First` in die Ausgabe.

Der zweite StringSerializer schreibt folgenden String:

```
<newline>Last Name:<tab>
```

Hinweis: Im IntelliScript-Editor werden der Zeilenvorschub und der Tabulator mit ihrem ASCII-Code und dem Symbol « dargestellt.

Nehmen wir nun an, dass Sie den Serializer an folgendem XML-Code ausführen:

```
<Person gender="M">
  <Name>
    <First>Ron</First>
    <Last>Lehrer</Last>
  </Name>
  <Id>547329876</Id>
  <Age>27</Age>
</Person>
```

Die Ausgabe der gezeigten Serialisierungsanker lautet:

```
First Name<tab>Ron<newline>Last Name<tab>
```

Dieser Text wird folgendermaßen angezeigt:

```
First Name:      Ron
Last Name:
```

Der Serializer enthält weitere Serialisierungsanker, die im obigen Beispiel nicht dargestellt wurden. Die vollständige Ausgabe des Serializers sieht folgendermaßen aus:

```
First Name:      Ron
Last Name:      Lehrer
Id:             547329876
Age:            27
Gender:         M
```

Reihenfolge von Serialisierungsankern

Ein Serializer führt die Serialisierungsanker in der Reihenfolge ihrer Definitionen aus.

Serialisierungsanker schreiben Daten sequenziell und hängen sie immer an das Ende des Ausgabedokuments an. Sie können diese Reihenfolge ändern, indem Sie die Folge in der Konfiguration des Serializers ändern.

Zwischen die Serialisierungsanker können Sie auch Aktionen einfügen. Die Aktionen werden als Teil der Folge ausgeführt.

Standardeigenschaften von Serializern

In der folgenden Tabelle werden die Standardeigenschaften in der Komponente **Serializer** sowie in zahlreichen Serialisierungsankern beschrieben:

Eigenschaft	Beschreibung
disabled	Legt fest, ob die Komponente und alle zugehörigen untergeordneten Komponenten vom Skript ignoriert werden. Verwenden Sie diese Eigenschaft zum Testen, Debuggen und Ändern eines Skripts. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none">- Ausgewählt. Das Skript ignoriert die Komponente.- Gelöscht. Das Skript wendet die Komponente an. Standardmäßig ist die Eigenschaft deaktiviert.
name	Eine beschreibende Kennzeichnung für die Komponente. Diese Kennzeichnung wird in der Protokolldatei sowie der Ereignisansicht angezeigt. Ermitteln Sie anhand der Eigenschaft name , welche Komponente das Ereignis verursacht hat.
notifications	Eine Liste mit NotificationHandler -Komponenten, die Benachrichtigungen aus geschachtelten Komponenten verarbeiten. Weitere Informationen hierzu finden Sie unter "Benachrichtigungen" auf Seite 430 .
on_fail	Die Aktion, die beim Ausfall der Komponente durchgeführt wird. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none">- Gelöscht. Es wird keine Aktion ausgeführt.- CustomLog. Es wird in das Benutzerprotokoll geschrieben.- LogError. Eine Fehlermeldung wird in das Engine-Protokoll geschrieben.- LogInfo. Eine Informationsmeldung wird in das Engine-Protokoll geschrieben.- LogWarning. Eine Warnmeldung wird in das Engine-Protokoll geschrieben.- NotifyFailure. Eine Mitteilung wird gesendet. Die Standardoption lautet „Gelöscht“. Weitere Informationen über die Behandlung von Komponentenfehlern finden Sie in "Fehlerbehandlung" auf Seite 410 .
optional	Legt fest, ob ein Komponentenfehler den Ausfall der übergeordneten Komponente verursacht. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none">- Ausgewählt. Ein Komponentenfehler führt nicht zum Ausfall der übergeordneten Komponente.- Gelöscht. Ein Komponentenfehler verursacht den Ausfall der übergeordneten Komponente. Standardwert ist "Gelöscht". Weitere Informationen über Komponentenfehler finden Sie in "Fehlerbehandlung" auf Seite 410 .
remark	Ein benutzerdefinierter Kommentar, der den Zweck oder die Aktion der Komponente beschreibt.

Die Komponente Serializer: Referenz

Ein **Serializer** wandelt XML-Dokumente in Ausgabedokumente beliebigen Formats um. Mithilfe von Serialisierungsankern ermittelt und bearbeitet er Daten in der Quelle.

Serializer

Ein **Serializer** wandelt XML-Dokumente in Ausgabedokumente um.

In der folgenden Tabelle werden die Eigenschaften der Komponente **Serializer** beschrieben:

Eigenschaft	Beschreibung
default_transformers	Eine Liste mit Transformern, die der Serializer auf alle serialisierten Daten anwendet.
example_source	Definiert ein XML-Beispielquelltdokument. Wenn Sie den Serializer in Developer-Tool ausführen, bearbeitet er das Beispieldokument. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none">- Leer. Beim Ausführen des Serializers werden Sie nach einem Quelldokument gefragt. Standard.- InputPort. Definiert einen Eingabeport.- LocalFile. Definiert eine Datei im lokalen Dateisystem- Text. Definiert einen String.- URL. Definiert eine URL.
name	Eine beschreibende Kennzeichnung für die Komponente. Diese Kennzeichnung wird in der Protokolldatei sowie der Ereignisansicht angezeigt. Ermitteln Sie anhand der Eigenschaft name , welche Komponente das Ereignis verursacht hat.
Benachrichtigungen	Eine Liste mit NotificationHandler -Komponenten, die Benachrichtigungen aus geschachtelten Komponenten verarbeiten. Weitere Informationen hierzu finden Sie unter "Benachrichtigungen" auf Seite 430 .
on_fail	Die Aktion, die beim Ausfall der Komponente durchgeführt wird. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none">- Gelöscht. Es wird keine Aktion ausgeführt.- CustomLog. Es wird in das Benutzerprotokoll geschrieben.- LogError. Eine Fehlermeldung wird in das Engine-Protokoll geschrieben.- LogInfo. Eine Informationsmeldung wird in das Engine-Protokoll geschrieben.- LogWarning. Eine Warnmeldung wird in das Engine-Protokoll geschrieben.- NotifyFailure. Eine Mitteilung wird gesendet. Die Standardoption lautet „Gelöscht“. Weitere Informationen über die Behandlung von Komponentengefehlern finden Sie in "Fehlerbehandlung" auf Seite 410 .
output_file_extension	Definiert die Dateierweiterung der generierten Ausgabedatei. Die Standarderweiterung lautet „.txt“.
remark	Ein benutzerdefinierter Kommentar, der den Zweck oder die Aktion der Komponente beschreibt.
root_tag	Definiert den Namen eines XML-Stammelements, das nicht im Eingabeschema für den Serializer enthalten ist. Hinweis: Wenn die Eingabe des Serializers in XML von einer anderen Komponente im Skript erfolgt und durch die Eigenschaften der Datenprozessor-Umwandlung um die Ausgabe ein einbettendes Stammelement hinzugefügt wird, müssen Sie diese Eigenschaft auf den Namen des einbettenden Stammelements festlegen.
source	Definiert einen Datenbehälter, der die Quelle für die Serialisierung enthält. Weitere Informationen hierzu finden Sie unter "Überblick über Locator, Keys und Indexierung" auf Seite 369 .

Eigenschaft	Beschreibung
target	Definiert einen Datenbehälter, der das Ergebnis der Serialisierung enthält. Weitere Informationen hierzu finden Sie unter "Überblick über Locator, Keys und Indexierung" auf Seite 369 .
validate_source_document	Bestimmt die Ebene der Quell-XML-Validierung, die der Serializer ausführt. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none"> - Teilweise. Gestattet eine gewisse Abweichung vom Schema. Standard. - Streng. Erfordert die strenge Einhaltung des Schemas.

Serialisierungsanker-Komponenten: Referenz

Serialisierungsanker in einem **Serializer** ermitteln und ändern Daten im Quelldokument.

AlternativeSerializers

Der Serialisierungsanker **AlternativeSerializers** definiert eine Gruppe alternativer Serialisierungsanker, die unter dem übergeordneten Serializer geschachtelt sind. Definieren Sie ein Kriterium, das bestimmt, welche Alternative der Serializer akzeptiert. Nur die akzeptierte Alternative hat Auswirkungen auf die Ausgabe des Serializers. Die anderen Serialisierungsanker haben keine Auswirkung auf die Ausgabe.

Die nachstehende Tabelle beschreibt die Eigenschaften des Serialisierungsankers **AlternativeSerializers**:

Eigenschaft	Beschreibung
disabled	Legt fest, ob die Komponente und alle zugehörigen untergeordneten Komponenten vom Skript ignoriert werden. Verwenden Sie diese Eigenschaft zum Testen, Debuggen und Ändern eines Skripts. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none"> - Ausgewählt. Das Skript ignoriert die Komponente. - Gelöscht. Das Skript wendet die Komponente an. Standardmäßig ist die Eigenschaft deaktiviert.
name	Eine beschreibende Kennzeichnung für die Komponente. Diese Kennzeichnung wird in der Protokolldatei sowie der Ereignisansicht angezeigt. Ermitteln Sie anhand der Eigenschaft name , welche Komponente das Ereignis verursacht hat.
on_fail	Die Aktion, die beim Ausfall der Komponente durchgeführt wird. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none"> - Gelöscht. Es wird keine Aktion ausgeführt. - CustomLog. Es wird in das Benutzerprotokoll geschrieben. - LogError. Eine Fehlermeldung wird in das Engine-Protokoll geschrieben. - LogInfo. Eine Informationsmeldung wird in das Engine-Protokoll geschrieben. - LogWarning. Eine Warnmeldung wird in das Engine-Protokoll geschrieben. - NotifyFailure. Eine Mitteilung wird gesendet. Die Standardoption lautet „Gelöscht“. Weitere Informationen über die Behandlung von Komponentenfehlern finden Sie in "Fehlerbehandlung" auf Seite 410 .

Eigenschaft	Beschreibung
optional	Legt fest, ob ein Komponentenfehler den Ausfall der übergeordneten Komponente verursacht. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none"> - Ausgewählt. Ein Komponentenfehler führt nicht zum Ausfall der übergeordneten Komponente. - Gelöscht. Ein Komponentenfehler verursacht den Ausfall der übergeordneten Komponente. Standardwert ist "Gelöscht". Weitere Informationen über Komponentenfehler finden Sie in "Fehlerbehandlung" auf Seite 410.
remark	Ein benutzerdefinierter Kommentar, der den Zweck oder die Aktion der Komponente beschreibt.
selector	Definiert das Kriterium für die Auswahl eines der alternativen Serialisierungsanker. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none"> - ScriptOrder. Der Serializer testet die geschachtelten Serialisierungsanker in der Reihenfolge, in der sie im Skript definiert sind. Der erste erfolgreich geprüfte Serialisierungsanker wird akzeptiert. Falls alle geschachtelten Serialisierungsanker fehlschlagen, schlägt auch die AlternativeSerializers-Komponente fehl. - NameSwitch. Der Serializer sucht nach dem geschachtelten Serialisierungsanker, dessen Eigenschaft name in einem Datenbehälter angegeben ist. Die übrigen Serialisierungsanker werden ignoriert. Wenn der benannte Serialisierungsanker fehlschlägt, schlägt auch die AlternativeSerializers-Komponente fehl. Standardwert ist ScriptOrder.

Beispiel

Die XML-Eingabe enthält entweder das Element `Product` oder das Element `Service`, aber nicht beide. Es soll jeweils das in der Eingabe vorhandene Element serialisiert werden.

Definieren Sie einen **AlternativeSerializers**-Serialisierungsanker und setzen Sie seine Eigenschaft **selector** auf `ScriptOrder`.

Schachteln Sie unter der Komponente **AlternativeSerializers** zwei **ContentSerializer**-Serialisierungsanker. Konfigurieren Sie einen dieser Serialisierunganker zum Verarbeiten des Elements `Product` und den anderen zum Verarbeiten des Elements `Service`.

ContentSerializer

Dieser **ContentSerializer**-Serialisierungsanker schreibt die serialisierten Daten in das Ausgabedokument.

Die nachstehende Tabelle beschreibt die Eigenschaften des Serialisierungsankers **ContentSerializer**:

Eigenschaft	Beschreibung
allow_empty_values	Legt fest, ob data_holder leer sein kann. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none"> - Ausgewählt. Der data_holder kann leer sein. - Gelöscht. Der Datenbehälter darf nicht leer sein, sonst schlägt der ContentSerializer fehl. Standardwert ist "Gelöscht".
closing_str	Definiert eine String, die der Anker nach dem data_holder schreibt.
data_holder	Definiert den Datenbehälter, der die serialisierten Daten enthält.

Eigenschaft	Beschreibung
disabled	<p>Legt fest, ob die Komponente und alle zugehörigen untergeordneten Komponenten vom Skript ignoriert werden. Verwenden Sie diese Eigenschaft zum Testen, Debuggen und Ändern eines Skripts. Sie können eine der folgenden Optionen auswählen:</p> <ul style="list-style-type: none"> - Ausgewählt. Das Skript ignoriert die Komponente. - Gelöscht. Das Skript wendet die Komponente an. <p>Standardmäßig ist die Eigenschaft deaktiviert.</p>
ignore_default_transformers	<p>Legt fest, ob die Standardtransformer des Serializers auf die serialisierten Daten angewandt werden. Sie können eine der folgenden Optionen auswählen:</p> <ul style="list-style-type: none"> - Ausgewählt. Die Standardtransformer des Serializers werden nicht angewandt. - Gelöscht. Die Standardtransformer des Serializers werden angewandt. <p>Standardwert ist "Gelöscht".</p>
name	<p>Eine beschreibende Kennzeichnung für die Komponente. Diese Kennzeichnung wird in der Protokolldatei sowie der Ereignisansicht angezeigt. Ermitteln Sie anhand der Eigenschaft name, welche Komponente das Ereignis verursacht hat.</p>
on_fail	<p>Die Aktion, die beim Ausfall der Komponente durchgeführt wird. Sie können eine der folgenden Optionen auswählen:</p> <ul style="list-style-type: none"> - Gelöscht. Es wird keine Aktion ausgeführt. - CustomLog. Es wird in das Benutzerprotokoll geschrieben. - LogError. Eine Fehlermeldung wird in das Engine-Protokoll geschrieben. - LogInfo. Eine Informationsmeldung wird in das Engine-Protokoll geschrieben. - LogWarning. Eine Warnmeldung wird in das Engine-Protokoll geschrieben. - NotifyFailure. Eine Mitteilung wird gesendet. <p>Die Standardoption lautet „Gelöscht“. Weitere Informationen über die Behandlung von Komponentenfehlern finden Sie in "Fehlerbehandlung" auf Seite 410.</p>
opening_str	<p>Definiert den String, den der Anker vor den Inhalten des data_holder schreibt.</p>
optional	<p>Legt fest, ob ein Komponentenfehler den Ausfall der übergeordneten Komponente verursacht. Sie können eine der folgenden Optionen auswählen:</p> <ul style="list-style-type: none"> - Ausgewählt. Ein Komponentenfehler führt nicht zum Ausfall der übergeordneten Komponente. - Gelöscht. Ein Komponentenfehler verursacht den Ausfall der übergeordneten Komponente. <p>Standardwert ist "Gelöscht". Weitere Informationen über Komponentenfehler finden Sie in "Fehlerbehandlung" auf Seite 410.</p>
remark	<p>Ein benutzerdefinierter Kommentar, der den Zweck oder die Aktion der Komponente beschreibt.</p>
transformers	<p>Definiert eine Liste mit Transformern, die auf die serialisierten Daten angewendet werden.</p>

DelimitedSectionsSerializer

Der Serialisierungsanker **DelimitedSectionsSerializer** verarbeitet Datenabschnitte. Die Abschnitte der Ausgabe werden durch einen definierten Trennzeichen-String getrennt.

Unter dem **DelimitedSectionsSerializer** können weitere Serialisierungsanker geschachtelt werden. Jeder geschachtelte Serialisierungsanker gibt einen einzigen Abschnitt aus.

Die nachstehende Tabelle beschreibt die Eigenschaften des Serialisierungsankers **DelimitedSectionsSerializer**:

Eigenschaft	Beschreibung
disabled	<p>Legt fest, ob die Komponente und alle zugehörigen untergeordneten Komponenten vom Skript ignoriert werden. Verwenden Sie diese Eigenschaft zum Testen, Debuggen und Ändern eines Skripts. Sie können eine der folgenden Optionen auswählen:</p> <ul style="list-style-type: none"> - Ausgewählt. Das Skript ignoriert die Komponente. - Gelöscht. Das Skript wendet die Komponente an. <p>Standardmäßig ist die Eigenschaft deaktiviert.</p>
name	<p>Eine beschreibende Kennzeichnung für die Komponente. Diese Kennzeichnung wird in der Protokolldatei sowie der Ereignisansicht angezeigt. Ermitteln Sie anhand der Eigenschaft name, welche Komponente das Ereignis verursacht hat.</p>
Benachrichtigungen	<p>Eine Liste mit NotificationHandler-Komponenten, die Benachrichtigungen aus geschachtelten Komponenten verarbeiten. Weitere Informationen hierzu finden Sie unter "Benachrichtigungen" auf Seite 430.</p>
on_fail	<p>Die Aktion, die beim Ausfall der Komponente durchgeführt wird. Sie können eine der folgenden Optionen auswählen:</p> <ul style="list-style-type: none"> - Gelöscht. Es wird keine Aktion ausgeführt. - CustomLog. Es wird in das Benutzerprotokoll geschrieben. - LogError. Eine Fehlermeldung wird in das Engine-Protokoll geschrieben. - LogInfo. Eine Informationsmeldung wird in das Engine-Protokoll geschrieben. - LogWarning. Eine Warnmeldung wird in das Engine-Protokoll geschrieben. - NotifyFailure. Eine Mitteilung wird gesendet. <p>Die Standardoption lautet „Gelöscht“. Weitere Informationen über die Behandlung von Komponentenfehlern finden Sie in "Fehlerbehandlung" auf Seite 410.</p>
optional	<p>Legt fest, ob ein Komponentenfehler den Ausfall der übergeordneten Komponente verursacht. Sie können eine der folgenden Optionen auswählen:</p> <ul style="list-style-type: none"> - Ausgewählt. Ein Komponentenfehler führt nicht zum Ausfall der übergeordneten Komponente. - Gelöscht. Ein Komponentenfehler verursacht den Ausfall der übergeordneten Komponente. <p>Standardwert ist "Gelöscht". Weitere Informationen über Komponentenfehler finden Sie in "Fehlerbehandlung" auf Seite 410.</p>
remark	<p>Ein benutzerdefinierter Kommentar, der den Zweck oder die Aktion der Komponente beschreibt.</p>
separator	<p>Definiert einen Serializer, der die Trennzeichen-Zeichenfolge festlegt. Sie können eine der folgenden Optionen auswählen:</p> <ul style="list-style-type: none"> - AlternativeSerializers - ContentSerializer - EmbeddedSerializer - GroupSerializer - RepeatingGroupSerializer - StringSerializer - Benutzerdefinierter Serializer <p>Standardwert ist StringSerializer.</p>

Eigenschaft	Beschreibung
separator_position	<p>Definiert die Position des Trennzeichens relativ zu den Abschnitten. Sie können eine der folgenden Optionen auswählen:</p> <ul style="list-style-type: none"> - Nachher. Schreibt hinter jeden Abschnitt ein Trennzeichen, auch hinter den letzten Abschnitt. Beispiel: 1 2 3 4 - Um. Schreibt vor und hinter jeden Abschnitt Trennzeichen, auch vor den ersten und hinter den letzten Abschnitt. Beispiel: 1 2 3 4 - Vorher. Schreibt vor jeden Abschnitt ein Trennzeichen, auch vor den ersten Abschnitt. Beispiel: 1 2 3 4 - Zwischen. Schreibt zwischen aufeinander folgende Abschnitte jeweils ein Trennzeichen, jedoch nicht vor den ersten und hinter den letzten Abschnitt. Beispiel: 1 2 3 4 <p>Standardwert ist "Vorher".</p>
using_placeholders	<p>Legt fest, ob der DelimitedSectionsSerializer das Trennzeichen eines optionalen Abschnitts schreibt, der in der XML-Eingabe fehlt. Sie können eine der folgenden Optionen auswählen:</p> <ul style="list-style-type: none"> - Immer. Schreibt das Trennzeichen eines fehlenden Abschnitts immer. Beispiel: 1 3 - Nie. Schreibt das Trennzeichen eines fehlenden Abschnitts nie. Beispiel: 1 3 - Wenn nötig. Schreibt das Trennzeichen eines fehlenden internen Abschnitts immer. Schreibt das Trennzeichen eines fehlenden Endabschnitts nie. Beispiel: 1 3 <p>Standardwert ist "Wenn nötig".</p>

Beispiel

Die XML-Eingabe enthält den Lebenslauf eines Mitarbeiters. Sie möchten die Daten im folgenden Format in ein Ausgabedokument schreiben:

```

-----
Jane Palmer
Employee ID 123456
-----
Professional Experience
...
-----
Education
...

```

Zu diesem Zweck definieren Sie einen **DelimitedSectionsSerializer** mit einer aus Bindestrichen bestehenden Zeile als Trennzeichen (`separator`). Da die aus Bindestrichen bestehende Zeile vor jeden Abschnitt eingefügt werden soll, setzen Sie **separator_position** auf `before`.

In den **DelimitedSectionsSerializer** schachteln Sie drei **GroupSerializer**-Komponenten. Der erste **GroupSerializer** schreibt den Abschnitt `Jane Palmer`, der zweite den Abschnitt `Professional Experience` und so weiter.

Optionale Abschnitte

In diesem Beispiel wird angenommen, dass der zweite Abschnitt, `Professional Experience`, in einigen Eingabe-XML-Dokumenten fehlt, Sie aber seinen Separator wie folgt in die Ausgabe schreiben wollen:

```

-----
Jane Palmer
Employee ID 123456
-----

```

```

-----
Education
...

```

Um diese Situation zu bewältigen, konfigurieren Sie **DelimitedSectionsSerializer** folgendermaßen:

- Wählen Sie im zweiten **GroupSerializer** die Eigenschaft **optional** aus. Dies bewirkt, dass bei einem Fehler des **GroupSerializer** nicht auch der **DelimitedSectionsSerializer** fehlschlägt.
- Setzen Sie in **DelimitedSectionsSerializer** den Wert **using_placeholders** mit `always` fest. Dies bewirkt, dass das Trennzeichen eines optionalen Abschnitts auch dann geschrieben wird, wenn der Abschnitt selbst fehlt.

Nehmen wir weiter an, dass das Trennzeichen des Abschnitts `Professional Experience` nicht geschrieben werden soll, wenn dieser Abschnitt fehlt:

```

-----
Jane Palmer
Employee ID 123456
-----
Education
...

```

In diesem Fall konfigurieren Sie den **DelimitedSectionsSerializer** folgendermaßen:

- Wählen Sie im zweiten **GroupSerializer** die Eigenschaft **optional** aus.
- Setzen Sie in **DelimitedSectionsSerializer** den Wert **using_placeholders** mit `never` fest. Dies bedeutet, dass das Trennzeichen eines fehlenden Abschnittes nie geschrieben wird.

EmbeddedSerializer

Der **EmbeddedSerializer**-Serialisierungsanker aktiviert einen zweiten **Serializer**, der seine Ausgabe in dasselbe Ausgabedokument schreibt.

Die nachstehende Tabelle beschreibt die Eigenschaften des Serialisierungsankers **EmbeddedSerializer**:

Eigenschaft	Beschreibung
disabled	Legt fest, ob die Komponente und alle zugehörigen untergeordneten Komponenten vom Skript ignoriert werden. Verwenden Sie diese Eigenschaft zum Testen, Debuggen und Ändern eines Skripts. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none"> - Ausgewählt. Das Skript ignoriert die Komponente. - Gelöscht. Das Skript wendet die Komponente an. Standardmäßig ist die Eigenschaft deaktiviert.
name	Eine beschreibende Kennzeichnung für die Komponente. Diese Kennzeichnung wird in der Protokolldatei sowie der Ereignisansicht angezeigt. Ermitteln Sie anhand der Eigenschaft name , welche Komponente das Ereignis verursacht hat.
on_fail	Die Aktion, die beim Ausfall der Komponente durchgeführt wird. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none"> - Gelöscht. Es wird keine Aktion ausgeführt. - CustomLog. Es wird in das Benutzerprotokoll geschrieben. - LogError. Eine Fehlermeldung wird in das Engine-Protokoll geschrieben. - LogInfo. Eine Informationsmeldung wird in das Engine-Protokoll geschrieben. - LogWarning. Eine Warnmeldung wird in das Engine-Protokoll geschrieben. - NotifyFailure. Eine Mitteilung wird gesendet. Die Standardoption lautet „Gelöscht“. Weitere Informationen über die Behandlung von Komponentenfehlern finden Sie in "Fehlerbehandlung" auf Seite 410 .

Eigenschaft	Beschreibung
optional	Legt fest, ob ein Komponentenfehler den Ausfall der übergeordneten Komponente verursacht. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none"> - Ausgewählt. Ein Komponentenfehler führt nicht zum Ausfall der übergeordneten Komponente. - Gelöscht. Ein Komponentenfehler verursacht den Ausfall der übergeordneten Komponente. Standardwert ist "Gelöscht". Weitere Informationen über Komponentenfehler finden Sie in "Fehlerbehandlung" auf Seite 410.
remark	Ein benutzerdefinierter Kommentar, der den Zweck oder die Aktion der Komponente beschreibt.
schema_connections	Verbindet die im zweiten Serializer referenzierten Datenbehälter mit den im übergeordneten Serializer referenzierten Datenbehältern. Die Eigenschaft enthält eine Liste von Connect -Teilkomponenten, die die Zugehörigkeiten definieren. Weitere Informationen hierzu finden Sie unter "Connect" auf Seite 256 . Wenn alle Datenbehälter im Hauptserializer und dem sekundären Serializer identisch sind, können Sie diese Eigenschaft weglassen. Bestehen jedoch Unterschiede zwischen einigen Datenbehältern, müssen die Datenbehälter explizit miteinander verbunden werden. Dies gilt auch für diejenigen, die identisch sind.
Serializer	Definiert den Namen eines zweiten Serializers, der auf der globalen Ebene des Skripts definiert ist.

Beispiel

Die XML-Eingabe ist ein Familienstammbaum. Sie enthält `Person`-Elemente, die folgendermaßen rekursiv geschachtelt sind:

```

<Person>          <!-- Parent -->
  ...
  <Person>        <!-- Child -->
    ...
    <Person>      <!-- Grandchild -->
      ...
    </Person>
  </Person>
</Person>

```

In einem **Serializer** kann sich eine **EmbeddedSerializer**-Komponente rekursiv selbst aufrufen, bis alle Schachtelungsebenen ausgeschöpft sind.

In diesem Beispiel verbindet die Eigenschaft **schema_connections** das Element `Person` mit `Person/Person`. Dadurch wird die zweite Instanz des Serializers angewiesen, eine geschachtelte Ebene der Eingabe zu verarbeiten. Wenn zwei `Person`-Elemente denselben Datentyp haben, reicht es, nur das übergeordnete Element (`Person`) zu verbinden und nicht die geschachtelten Elemente (`Person/*s/Name`, `Person/*s/BirthDate` usw.)

GroupSerializer

Beim **GroupSerializer**-Serialisierungsanker sind die geschachtelten Serialisierungsanker aneinander gebunden. Für den **GroupSerializer** festgelegte Eigenschaften wirken sich auch auf die Elemente der Gruppe aus.

Die nachstehende Tabelle beschreibt die Eigenschaften des Serialisierungsankers **GroupSerializer**:

Eigenschaft	Beschreibung
disabled	Legt fest, ob die Komponente und alle zugehörigen untergeordneten Komponenten vom Skript ignoriert werden. Verwenden Sie diese Eigenschaft zum Testen, Debuggen und Ändern eines Skripts. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none"> - Ausgewählt. Das Skript ignoriert die Komponente. - Gelöscht. Das Skript wendet die Komponente an. Standardmäßig ist die Eigenschaft deaktiviert.
name	Eine beschreibende Kennzeichnung für die Komponente. Diese Kennzeichnung wird in der Protokolldatei sowie der Ereignisansicht angezeigt. Ermitteln Sie anhand der Eigenschaft name , welche Komponente das Ereignis verursacht hat.
Benachrichtigungen	Eine Liste mit NotificationHandler -Komponenten, die Benachrichtigungen aus geschachtelten Komponenten verarbeiten. Weitere Informationen hierzu finden Sie unter "Benachrichtigungen" auf Seite 430 .
on_fail	Die Aktion, die beim Ausfall der Komponente durchgeführt wird. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none"> - Gelöscht. Es wird keine Aktion ausgeführt. - CustomLog. Es wird in das Benutzerprotokoll geschrieben. - LogError. Eine Fehlermeldung wird in das Engine-Protokoll geschrieben. - LogInfo. Eine Informationsmeldung wird in das Engine-Protokoll geschrieben. - LogWarning. Eine Warnmeldung wird in das Engine-Protokoll geschrieben. - NotifyFailure. Eine Mitteilung wird gesendet. Die Standardoption lautet „Gelöscht“. Weitere Informationen über die Behandlung von Komponentenfehlern finden Sie in "Fehlerbehandlung" auf Seite 410 .
optional	Legt fest, ob ein Komponentenfehler den Ausfall der übergeordneten Komponente verursacht. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none"> - Ausgewählt. Ein Komponentenfehler führt nicht zum Ausfall der übergeordneten Komponente. - Gelöscht. Ein Komponentenfehler verursacht den Ausfall der übergeordneten Komponente. Standardwert ist "Gelöscht". Weitere Informationen über Komponentenfehler finden Sie in "Fehlerbehandlung" auf Seite 410 .
remark	Ein benutzerdefinierter Kommentar, der den Zweck oder die Aktion der Komponente beschreibt.
source	Definiert einen Datenbehälter, der die Quelle für die Serialisierung enthält. Weitere Informationen hierzu finden Sie unter "Überblick über Validatoren, Benachrichtigungen und Fehlerbehandlung" auf Seite 409 .
target	Definiert einen Datenbehälter, der das Ergebnis der Serialisierung enthält. Weitere Informationen hierzu finden Sie unter "Überblick über Validatoren, Benachrichtigungen und Fehlerbehandlung" auf Seite 409 .

RepeatingGroupSerializer

Dieser Serialisierungsanker **RepeatingGroupSerializer** schreibt eine wiederholte Struktur in das Ausgabedokument.

Verwenden Sie einen **RepeatingGroupSerializer**, wenn die XML-Daten einen Datenbehälter mit mehreren Instanzen enthält. Er iteriert über die einzelnen Instanzen des Datenbehälters und gibt die Daten aus. Weitere Informationen hierzu finden Sie unter ["Mehrfachinstanz-Datenbehälter" auf Seite 202](#).

Unter dem **RepeatingGroupSerializer** schachteln Sie Serialisierungsanker, die die einzelnen Instanzen des Datenbehälters verarbeiten und ausgeben. Sie können auch ein Trennzeichen definieren, das vom **RepeatingGroupSerializer** in der Ausgabe zwischen die Iterationen geschrieben wird.

Die nachstehende Tabelle beschreibt die Eigenschaften des Serialisierungsankers **RepeatingGroupSerializer**:

Eigenschaft	Beschreibung
count	Definiert die Anzahl der auszuführenden Iterationen. Wenn diese Eigenschaft leer bleibt, werden die Iterationen solange fortgesetzt, bis die Eingabe erschöpft ist.
current_iteration	Ein Datenbehälter, in den der RepeatingGroupSerializer die Nummer der aktuellen Iteration ausgibt. Mit einem ContentSerializer können Sie die Nummer in die Ausgabe schreiben.
disabled	Legt fest, ob die Komponente und alle zugehörigen untergeordneten Komponenten vom Skript ignoriert werden. Verwenden Sie diese Eigenschaft zum Testen, Debuggen und Ändern eines Skripts. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none"> - Ausgewählt. Das Skript ignoriert die Komponente. - Gelöscht. Das Skript wendet die Komponente an. Standardmäßig ist die Eigenschaft deaktiviert.
name	Eine beschreibende Kennzeichnung für die Komponente. Diese Kennzeichnung wird in der Protokolldatei sowie der Ereignisansicht angezeigt. Ermitteln Sie anhand der Eigenschaft name , welche Komponente das Ereignis verursacht hat.
Benachrichtigungen	Eine Liste mit NotificationHandler -Komponenten, die Benachrichtigungen aus geschachtelten Komponenten verarbeiten. Weitere Informationen hierzu finden Sie unter "Benachrichtigungen" auf Seite 430 .
on_fail	Die Aktion, die beim Ausfall der Komponente durchgeführt wird. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none"> - Gelöscht. Es wird keine Aktion ausgeführt. - CustomLog. Es wird in das Benutzerprotokoll geschrieben. - LogError. Eine Fehlermeldung wird in das Engine-Protokoll geschrieben. - LogInfo. Eine Informationsmeldung wird in das Engine-Protokoll geschrieben. - LogWarning. Eine Warnmeldung wird in das Engine-Protokoll geschrieben. - NotifyFailure. Eine Mitteilung wird gesendet. Die Standardoption lautet „Gelöscht“. Weitere Informationen über die Behandlung von Komponentenfehlern finden Sie in "Fehlerbehandlung" auf Seite 410 .
on_iteration_fail	Legt die Aktion fest, wenn eine Iteration fehlschlägt. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none"> - Gelöscht. Keine Aktion. - CustomLog. Schreibt in das Benutzerprotokoll. - LogError. Schreibt eine Fehlermeldung in das Engine-Protokoll. - LogInfo. Schreibt eine Informationsmeldung in das Engine-Protokoll. - LogWarning. Schreibt eine Warnmeldung in das Engine-Protokoll. - NotifyFailure. Löst eine Benachrichtigung aus. Die Eigenschaft on_iteration_fail wird zum Schreiben eines Eintrags verwendet, wenn eine einzelne Iteration fehlschlägt. Die Eigenschaft on_fail wird zum Schreiben eines Eintrags verwendet, wenn der gesamte RepeatingGroupSerializer fehlschlägt. Weitere Informationen hierzu finden Sie unter "Fehlerbehandlung" auf Seite 410 .
optional	Legt fest, ob ein Komponentenfehler den Ausfall der übergeordneten Komponente verursacht. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none"> - Ausgewählt. Ein Komponentenfehler führt nicht zum Ausfall der übergeordneten Komponente. - Gelöscht. Ein Komponentenfehler verursacht den Ausfall der übergeordneten Komponente. Standardwert ist "Gelöscht". Weitere Informationen über Komponentenfehler finden Sie in "Fehlerbehandlung" auf Seite 410 .

Eigenschaft	Beschreibung
remark	Ein benutzerdefinierter Kommentar, der den Zweck oder die Aktion der Komponente beschreibt.
separator	Definiert einen Serialisierungsanker, der den Trennzeichen-String definiert. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none"> - AlternativeSerializers - ContentSerializer - EmbeddedSerializer - GroupSerializer - RepeatingGroupSerializer - StringSerializer - Benutzerdefinierter Serializer Standardwert ist "Leer".
separator_position	Definiert die Position des Trennzeichens relativ zu den Abschnitten. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none"> - Nachher. Schreibt hinter jeden Abschnitt ein Trennzeichen, auch hinter den letzten Abschnitt. Beispiel: 1 2 3 4 - Um. Schreibt vor und hinter jeden Abschnitt Trennzeichen, auch vor den ersten und hinter den letzten Abschnitt. Beispiel: 1 2 3 4 - Vorher. Schreibt vor jeden Abschnitt ein Trennzeichen, auch vor den ersten Abschnitt. Beispiel: 1 2 3 4 - Zwischen. Schreibt zwischen aufeinander folgende Abschnitte jeweils ein Trennzeichen, jedoch nicht vor den ersten und hinter den letzten Abschnitt. Beispiel: 1 2 3 4 Standardwert ist "Vorher".
skip_failed_iterations	Legt fest, ob fehlgeschlagene Iterationen übersprungen werden. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none"> - Ausgewählt. RepeatingGroup überspringt eine fehlgeschlagene Iteration und fährt mit der nächsten Iteration fort. Wenn eine Iteration erfolgreich ist, ist auch RepeatingGroup. - Gelöscht. RepeatingGroup schlägt fehl, wenn eine Iteration fehlschlägt. Die Eigenschaft skip_failed_iterations hat nur dann einen Effekt, wenn separator definiert ist. Standardwert ist "Ausgewählt".
source	Definiert einen Datenbehälter, der die Quelle für die Serialisierung enthält. Weitere Informationen hierzu finden Sie unter "Überblick über Validatoren, Benachrichtigungen und Fehlerbehandlung" auf Seite 409 .
target	Definiert einen Datenbehälter, der das Ergebnis der Serialisierung enthält. Weitere Informationen hierzu finden Sie unter "Überblick über Validatoren, Benachrichtigungen und Fehlerbehandlung" auf Seite 409 .

Beispiel

Die XML-Eingabe enthält die folgende Struktur:

```
<Persons>
  <Person>
    <Name>John</Name>
    <Age>35</Age>
  </Person>
  <Person>
    <Name>Larissa</Name>
    <Age>42</Age>
  </Person>
```

```
<...  
</Persons>
```

Von einem **RepeatingGroupSerializer** mit einem Zeilenvorschubzeichen als Trennzeichen werden die Daten folgendermaßen ausgegeben:

```
John      35  
Larissa   42
```

Die Iteration kann über mehrere Mehrfachinstanz-Datenbehälter gleichzeitig erfolgen. Beispielsweise kann die Komponente über eine Liste mit Männern und eine Liste mit Frauen iterieren und eine Liste von Ehepaaren ausgeben. Zu diesem Zweck fügen Sie für jeden Datenbehälter einen **ContentSerializer** in die Wiederholungsgruppe ein.

StringSerializer

Der Serialisierungsanker **StringSerializer** schreibt einen vordefinierten String in das Ausgabedokument.

In der folgenden Tabelle werden die Eigenschaften des Serialisierungsankers **StringSerializer** beschrieben:

Eigenschaft	Beschreibung
disabled	Legt fest, ob die Komponente und alle zugehörigen untergeordneten Komponenten vom Skript ignoriert werden. Verwenden Sie diese Eigenschaft zum Testen, Debuggen und Ändern eines Skripts. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none">- Ausgewählt. Das Skript ignoriert die Komponente.- Gelöscht. Das Skript wendet die Komponente an. Standardmäßig ist die Eigenschaft deaktiviert.
name	Eine beschreibende Kennzeichnung für die Komponente. Diese Kennzeichnung wird in der Protokolldatei sowie der Ereignisansicht angezeigt. Ermitteln Sie anhand der Eigenschaft name , welche Komponente das Ereignis verursacht hat.
on_fail	Die Aktion, die beim Ausfall der Komponente durchgeführt wird. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none">- Gelöscht. Es wird keine Aktion ausgeführt.- CustomLog. Es wird in das Benutzerprotokoll geschrieben.- LogError. Eine Fehlermeldung wird in das Engine-Protokoll geschrieben.- LogInfo. Eine Informationsmeldung wird in das Engine-Protokoll geschrieben.- LogWarning. Eine Warnmeldung wird in das Engine-Protokoll geschrieben.- NotifyFailure. Eine Mitteilung wird gesendet. Die Standardoption lautet „Gelöscht“. Weitere Informationen über die Behandlung von Komponentenfehlern finden Sie in "Fehlerbehandlung" auf Seite 410 .
remark	Ein benutzerdefinierter Kommentar, der den Zweck oder die Aktion der Komponente beschreibt.
str	Definiert die Zeichenfolge, die der Serializer in die Ausgabe schreibt.

KAPITEL 20

Mapper

Dieses Kapitel umfasst die folgenden Themen:

- [Erstellen eines Mappers, 358](#)
- [In einen Mapper geschachtelte Komponenten, 358](#)
- [Mapper: Beispiel, 359](#)
- [Standardeigenschaften von Mappern, 360](#)
- [Die Komponente Mapper: Referenz, 361](#)
- [Mapper-Ankerkomponenten: Referenz, 363](#)

Erstellen eines Mappers

1. Fügen Sie Eingabe- und Ausgabeschemata zum Schemaobjekt hinzu und referenzieren Sie die Schemata in der Datenprozessor-Umwandlung.
2. Fügen Sie auf der globalen Ebene des Skripts eine **Mapper**-Komponente hinzu.
3. Weisen Sie die Eigenschaften **source** und **Ziel** des **Mapper** zu den Eingabe- und Ausgabeelementen des **Mapper** zu.
4. Ordnen Sie die Eigenschaft **example_source** einem Beispiel-XML-Eingabedokument zu.
Durch Hinzufügen von Komponenten zum Mapper kennzeichnet das Developer-Tool die entsprechenden Stellen in der Beispielquelle mit Farbcodes. Mithilfe der Farben können Sie überprüfen, ob die Komponenten richtig definiert sind.
5. Bearbeiten Sie die sonstigen Eigenschaften des **Mappers** entsprechend Ihren Erfordernissen.
6. Schachteln Sie innerhalb des **Mappers** eine Folge von **Map**-Aktionen, Mapper-Ankern und sonstigen erforderlichen Komponenten.
7. Testen Sie den Mapper und ändern Sie das Skript.

In einen Mapper geschachtelte Komponenten

In einen `Mapper` können Sie die folgenden Komponenten schachteln:

- Eine beliebige Anzahl von `Map`-Aktionen. Diese Aktionen rufen aus der Eingabe einen Datenbehälter ab und schreiben seinen Inhalt in die Ausgabe.

- Optional eine beliebige Anzahl von Mapper-Ankern. Weitere Informationen hierzu finden Sie im Abschnitt [“Mapper-Ankerkomponenten: Referenz” auf Seite 363](#).
- Optional jede Anzahl an zusätzlichen Aktionen.

Die `Map`-Aktionen und Mapper-Anker können in beliebiger Reihenfolge angeordnet sein. Sie können auch weitere Aktionen in diese Folge einfügen.

Zum Schreiben der Daten in die XML-Ausgabe verwendet der Mapper keine Mapper-Anker, sondern `Map`-Aktionen. In diesem Punkt scheinen sich Mapper von Parsern und Serializern zu unterscheiden, bei denen die Ausgabe von Ankern bzw. Serialisierungsankern geschrieben wird. Tatsächlich ist dies eher eine Frage der Terminologie. Die `Map`-Aktion hätte ebenso als ein Mapper-Anker bezeichnet werden können. Sie ist als Aktion definiert, da sie auch in anderen Fällen einsetzbar ist, die nichts mit Mappern zu tun haben.

Mapper: Beispiel

Das folgende Beispiel veranschaulicht die Konfiguration eines Mappers.

XML-Quelle

Die Eingabe des Mappers ist ein XML-Dokument, das eine Liste von Eigennamen und die ihnen zugeordneten Kennnummern enthält.

```
<Persons>
  <Person ID="10">Bob</Person>
  <Person ID="17">Larissa</Person>
  <Person ID="13">Marie</Person>
</Persons>
```

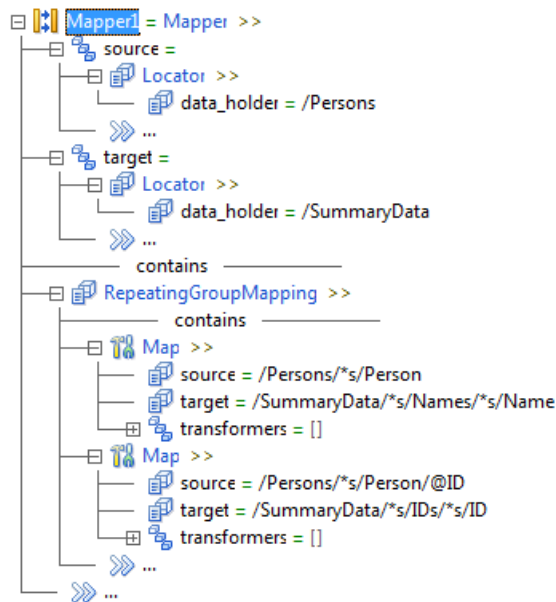
XML-Ausgabe

Der Mapper soll eine in XML formatierte Liste von Namen und Kennnummern ausgeben, zwischen denen keine Zuordnung besteht.

```
<SummaryData>
  <Names>
    <Name>Bob</Name>
    <Name>Larissa</Name>
    <Name>Marie</Name>
  </Names>
  <IDs>
    <ID>10</ID>
    <ID>17</ID>
    <ID>13</ID>
  </IDs>
</SummaryData>
```

Mapperkonfiguration

Die folgende Mapper-Konfiguration nimmt die gewünschte Umwandlung vor:



Der Mapper-Anker `RepeatingGroupMapping` iteriert über die `Person`-Elemente der Eingabe. Er schreibt die Daten mit Hilfe von `Map`-Aktionen in die Elemente `Name` und `ID` der Ausgabe.

Standardeigenschaften von Mappern

In der folgenden Tabelle werden die Standardeigenschaften in der Komponente **Mapper** sowie in zahlreichen Mapper-Ankern beschrieben:

Eigenschaft	Beschreibung
disabled	Legt fest, ob die Komponente und alle zugehörigen untergeordneten Komponenten vom Skript ignoriert werden. Verwenden Sie diese Eigenschaft zum Testen, Debuggen und Ändern eines Skripts. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none"> - Ausgewählt. Das Skript ignoriert die Komponente. - Gelöscht. Das Skript wendet die Komponente an. Standardmäßig ist die Eigenschaft deaktiviert.
name	Eine beschreibende Kennzeichnung für die Komponente. Diese Kennzeichnung wird in der Protokolldatei sowie der Ereignisansicht angezeigt. Ermitteln Sie anhand der Eigenschaft name , welche Komponente das Ereignis verursacht hat.
notifications	Eine Liste mit NotificationHandler -Komponenten, die Benachrichtigungen aus geschachtelten Komponenten verarbeiten. Weitere Informationen hierzu finden Sie unter "Benachrichtigungen" auf Seite 430 .

Eigenschaft	Beschreibung
on_fail	<p>Die Aktion, die beim Ausfall der Komponente durchgeführt wird. Sie können eine der folgenden Optionen auswählen:</p> <ul style="list-style-type: none"> - Gelöscht. Es wird keine Aktion ausgeführt. - CustomLog. Es wird in das Benutzerprotokoll geschrieben. - LogError. Eine Fehlermeldung wird in das Engine-Protokoll geschrieben. - LogInfo. Eine Informationsmeldung wird in das Engine-Protokoll geschrieben. - LogWarning. Eine Warnmeldung wird in das Engine-Protokoll geschrieben. - NotifyFailure. Eine Mitteilung wird gesendet. <p>Die Standardoption lautet „Gelöscht“. Weitere Informationen über die Behandlung von Komponentenfehlern finden Sie in "Fehlerbehandlung" auf Seite 410.</p>
optional	<p>Legt fest, ob ein Komponentenfehler den Ausfall der übergeordneten Komponente verursacht. Sie können eine der folgenden Optionen auswählen:</p> <ul style="list-style-type: none"> - Ausgewählt. Ein Komponentenfehler führt nicht zum Ausfall der übergeordneten Komponente. - Gelöscht. Ein Komponentenfehler verursacht den Ausfall der übergeordneten Komponente. <p>Standardwert ist "Gelöscht". Weitere Informationen über Komponentenfehler finden Sie in "Fehlerbehandlung" auf Seite 410.</p>
remark	Ein benutzerdefinierter Kommentar, der den Zweck oder die Aktion der Komponente beschreibt.

Die Komponente Mapper: Referenz

Ein Mapper liest ein XML-Quelldokument und wandelt es in ein anderes XML-Dokument um.

Mapper

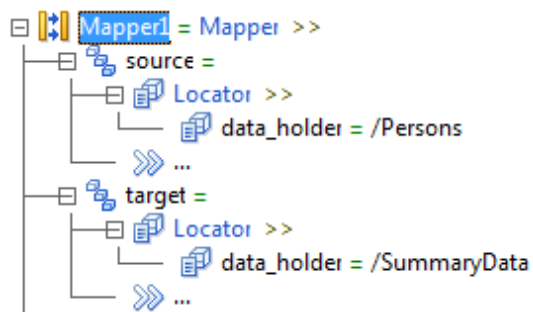
Ein **Mapper** führt Umwandlungen zwischen XML und XML durch. Er wandelt ein XML-Quelldokument in ein Ausgabedokument mit einer anderen XML-Struktur um.

Die nachstehende Tabelle beschreibt die Eigenschaften der Komponente **Mapper**:

Eigenschaft	Beschreibung
example_source	<p>Definiert ein XML-Beispielquelltdokument. Wenn Sie den Mapper im Developer-Tool ausführen, bearbeitet er das Beispieldokument. Sie können eine der folgenden Optionen auswählen:</p> <ul style="list-style-type: none"> - Leer. Der Benutzer wird beim Ausführen des Skripts zur Eingabe eines Quelldokuments aufgefordert. - InputPort. Definiert einen Eingabeport. - LocalFile. Definiert eine Datei im lokalen Dateisystem. - Text. Definiert einen String. - URL. Definiert eine URL. <p>Standardwert ist "Leer".</p>
name	<p>Eine beschreibende Kennzeichnung für die Komponente. Diese Kennzeichnung wird in der Protokolldatei sowie der Ereignisansicht angezeigt. Ermitteln Sie anhand der Eigenschaft name, welche Komponente das Ereignis verursacht hat.</p>

Eigenschaft	Beschreibung
Benachrichtigungen	Eine Liste mit NotificationHandler -Komponenten, die Benachrichtigungen aus geschachtelten Komponenten verarbeiten. Weitere Informationen hierzu finden Sie unter "Benachrichtigungen" auf Seite 430 .
on_fail	Die Aktion, die beim Ausfall der Komponente durchgeführt wird. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none"> - Gelöscht. Es wird keine Aktion ausgeführt. - CustomLog. Es wird in das Benutzerprotokoll geschrieben. - LogError. Eine Fehlermeldung wird in das Engine-Protokoll geschrieben. - LogInfo. Eine Informationsmeldung wird in das Engine-Protokoll geschrieben. - LogWarning. Eine Warnmeldung wird in das Engine-Protokoll geschrieben. - NotifyFailure. Eine Mitteilung wird gesendet. Die Standardoption lautet „Gelöscht“. Weitere Informationen über die Behandlung von Komponentenfehlern finden Sie in "Fehlerbehandlung" auf Seite 410 .
remark	Ein benutzerdefinierter Kommentar, der den Zweck oder die Aktion der Komponente beschreibt.
root_tag	Definiert den Namen eines XML-Root-Elements, der nicht im Schema definiert ist. Beispiel: Angenommen, das Element oberster Ebene des Schemas ist <code>Person</code> , in der XML-Eingabe ist <code>Person</code> jedoch in ein Element namens <code>InputWrapper</code> geschachtelt. In diesem Fall müssen Sie root_tag=InputWrapper eingeben.
source	Definiert eine Locator-Komponente, die einen XML-Datenbehälter definiert. Der Datenbehälter enthält die Stammebene der XML-Quelle für das Mapping. Weitere Informationen hierzu finden Sie unter "Überblick über Locator, Keys und Indexierung" auf Seite 369 .
target	Definiert eine Locator-Komponente, die einen XML-Datenbehälter definiert. Der Datenbehälter enthält die Stammebene der Ausgabe-XML-Quelle für das Mapping. Weitere Informationen hierzu finden Sie unter "Überblick über Locator, Keys und Indexierung" auf Seite 369 .
validate_source_document	Bestimmt die Ebene der Quell-XML-Validierung, die der Serializer ausführt. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none"> - Teilweise. Gestattet eine gewisse Abweichung vom Schema. - Streng. Erfordert die strenge Einhaltung des Schemas. Standardwert ist "Teilweise".

In den Eigenschaften **source** und **target** werden die Stammelemente der XML-Dokumente angegeben. Wenn beispielsweise das Dokumentelement der XML-Quelle `Persons` ist und das Dokumentelement der Ausgabe `SummaryData`, müssen **source** und **target** folgendermaßen zugewiesen werden:



Mapper-Ankerkomponenten: Referenz

Mapper-Anker in einem Mapper identifizieren und bearbeiten Daten in einem XML-Dokument.

AlternativeMappings

Der **AlternativeMappings**-Mapper-Anker definiert eine Gruppe alternativer Mapper-Anker. Definieren Sie ein Kriterium für die Auswahl einer Alternative. Nur die akzeptierte Alternative hat Auswirkungen auf die Ausgabe.

In der nachstehenden Tabelle werden die Eigenschaften des **AlternativeMappings**-Mapper-Ankers beschrieben:

Eigenschaft	Beschreibung
disabled	Legt fest, ob die Komponente und alle zugehörigen untergeordneten Komponenten vom Skript ignoriert werden. Verwenden Sie diese Eigenschaft zum Testen, Debuggen und Ändern eines Skripts. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none">- Ausgewählt. Das Skript ignoriert die Komponente.- Gelöscht. Das Skript wendet die Komponente an. Standardmäßig ist die Eigenschaft deaktiviert.
name	Eine beschreibende Kennzeichnung für die Komponente. Diese Kennzeichnung wird in der Protokolldatei sowie der Ereignisansicht angezeigt. Ermitteln Sie anhand der Eigenschaft name , welche Komponente das Ereignis verursacht hat.
on_fail	Die Aktion, die beim Ausfall der Komponente durchgeführt wird. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none">- Gelöscht. Es wird keine Aktion ausgeführt.- CustomLog. Es wird in das Benutzerprotokoll geschrieben.- LogError. Eine Fehlermeldung wird in das Engine-Protokoll geschrieben.- LogInfo. Eine Informationsmeldung wird in das Engine-Protokoll geschrieben.- LogWarning. Eine Warnmeldung wird in das Engine-Protokoll geschrieben.- NotifyFailure. Eine Mitteilung wird gesendet. Die Standardoption lautet „Gelöscht“. Weitere Informationen über die Behandlung von Komponentenfehlern finden Sie in "Fehlerbehandlung" auf Seite 410 .
optional	Legt fest, ob ein Komponentenfehler den Ausfall der übergeordneten Komponente verursacht. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none">- Ausgewählt. Ein Komponentenfehler führt nicht zum Ausfall der übergeordneten Komponente.- Gelöscht. Ein Komponentenfehler verursacht den Ausfall der übergeordneten Komponente. Standardwert ist "Gelöscht". Weitere Informationen über Komponentenfehler finden Sie in "Fehlerbehandlung" auf Seite 410 .
remark	Ein benutzerdefinierter Kommentar, der den Zweck oder die Aktion der Komponente beschreibt.
selector	Legt das Kriterium für die Auswahl eines alternativen Mappers fest. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none">- ScriptOrder. Das Skript testet die geschachtelten Mapper-Anker in der Reihenfolge, in der sie im Skript definiert sind. Der erste erfolgreich geprüfte Anker wird akzeptiert. Wenn alle geschachtelten Mapper-Anker fehlschlagen, schlägt auch die AlternativeMappings-Komponente fehl.- NameSwitch. Das Skript sucht nach dem geschachtelten Mapper-Anker, dessen Eigenschaft name in einem Datenbehälter angegeben ist. Die übrigen geschachtelten Mapper-Anker werden ignoriert. Wenn der benannte Mapper-Anker fehlschlägt, schlägt auch die AlternativeMappings-Komponente fehl. Standardwert ist ScriptOrder.

Beispiel

Die XML-Eingabe enthält entweder das Element `Product` oder das Element `Service`, aber nicht beide. Sie möchten immer das in der Eingabe vorhandene Element verarbeiten.

Definieren Sie einen **AlternativeMappings**-Mapper-Anker, und setzen Sie seine Eigenschaft **selector** auf `ScriptOrder`.

Schachteln Sie in **AlternativeMappings** zwei `Map`-Aktionen. Konfigurieren Sie einen dieser Serialisierunganker zum Verarbeiten des Elements `Product` und den anderen zum Verarbeiten des Elements `Service`.

EmbeddedMapper

Der **EmbeddedMapper**-Mapper-Anker aktiviert einen zweiten **Mapper**, der seine Ausgabe in dasselbe Ausgabedokument schreibt.

In der nachstehenden Tabelle werden die Eigenschaften des **EmbeddedMapper**-Mapper-Ankers beschrieben:

Eigenschaft	Beschreibung
disabled	Legt fest, ob die Komponente und alle zugehörigen untergeordneten Komponenten vom Skript ignoriert werden. Verwenden Sie diese Eigenschaft zum Testen, Debuggen und Ändern eines Skripts. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none">- Ausgewählt. Das Skript ignoriert die Komponente.- Gelöscht. Das Skript wendet die Komponente an. Standardmäßig ist die Eigenschaft deaktiviert.
Mapper	Definiert den zweiten Mapper.
name	Eine beschreibende Kennzeichnung für die Komponente. Diese Kennzeichnung wird in der Protokolldatei sowie der Ereignisansicht angezeigt. Ermitteln Sie anhand der Eigenschaft name , welche Komponente das Ereignis verursacht hat.
on_fail	Die Aktion, die beim Ausfall der Komponente durchgeführt wird. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none">- Gelöscht. Es wird keine Aktion ausgeführt.- CustomLog. Es wird in das Benutzerprotokoll geschrieben.- LogError. Eine Fehlermeldung wird in das Engine-Protokoll geschrieben.- LogInfo. Eine Informationsmeldung wird in das Engine-Protokoll geschrieben.- LogWarning. Eine Warnmeldung wird in das Engine-Protokoll geschrieben.- NotifyFailure. Eine Mitteilung wird gesendet. Die Standardoption lautet „Gelöscht“. Weitere Informationen über die Behandlung von Komponentenfehlern finden Sie in "Fehlerbehandlung" auf Seite 410 .
optional	Legt fest, ob ein Komponentenfehler den Ausfall der übergeordneten Komponente verursacht. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none">- Ausgewählt. Ein Komponentenfehler führt nicht zum Ausfall der übergeordneten Komponente.- Gelöscht. Ein Komponentenfehler verursacht den Ausfall der übergeordneten Komponente. Standardwert ist "Gelöscht". Weitere Informationen über Komponentenfehler finden Sie in "Fehlerbehandlung" auf Seite 410 .

Eigenschaft	Beschreibung
remark	Ein benutzerdefinierter Kommentar, der den Zweck oder die Aktion der Komponente beschreibt.
schema_connections	<p>Verbindet die im zweiten Mapper referenzierten Datenbehälter mit den im übergeordneten Mapper referenzierten Datenbehältern. Die Eigenschaft enthält eine Liste von Connect-Teilkomponenten, die die Zugehörigkeiten definieren. Weitere Informationen hierzu finden Sie unter "Connect" auf Seite 256.</p> <p>Wenn alle Datenbehälter im Hauptmapper und dem sekundären Mapper identisch sind, können Sie diese Eigenschaft weglassen. Bestehen jedoch Unterschiede zwischen einigen Datenbehältern, müssen die Datenbehälter explizit miteinander verbunden werden.</p>

Beispiel

Die XML-Eingabe ist ein Familienstammbaum. Sie enthält `Person`-Elemente, die folgendermaßen rekursiv geschachtelt sind:

```

<Person>          <!-- Parent -->
...
  <Person>        <!-- Child -->
...
    <Person>      <!-- Grandchild -->
...
    </Person>
  </Person>
</Person>

```

Ein **Mapper** kann sich mit Hilfe der **EmbeddedMapper**-Komponente rekursiv selbst aufrufen, bis alle Schachtelungsebenen ausgeschöpft sind.

In diesem Beispiel ist `Person` mit `Person/Person` verbunden. Dadurch wird die zweite Instanz des Mappers angewiesen, eine geschachtelte Ebene der Eingabe zu verarbeiten. Wenn zwei `Person`-Elemente denselben Datentyp haben, reicht es, nur das übergeordnete Element (`Person`) zu verbinden und nicht die geschachtelten Elemente (`Person/*s/Name`, `Person/*s/BirthDate` usw.)

GroupMapping

Beim **GroupMapping**-Mapper-Anker sind die geschachtelten Mapper-Anker und Aktionen aneinander gebunden. Für **GroupMapping** festgelegte Eigenschaften wirken sich auch auf die Elemente der Gruppe aus.

In der nachstehenden Tabelle werden die Eigenschaften des **GroupMapping**-Mapper-Ankers beschrieben:

Eigenschaft	Beschreibung
absent	Legt das Verhalten von GroupMapping fest, wenn einer seiner obligatorischen geschachtelten Mapper-Anker oder Aktionen fehlschlägt. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none"> - Ausgewählt. GroupMapping erfolgreich. - Gelöscht. GroupMapping scheitert. Verwenden Sie diese Eigenschaft, um zu prüfen, ob geschachtelte Mapper-Anker fehlen. Standardwert ist "Gelöscht".
disabled	Legt fest, ob die Komponente und alle zugehörigen untergeordneten Komponenten vom Skript ignoriert werden. Verwenden Sie diese Eigenschaft zum Testen, Debuggen und Ändern eines Skripts. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none"> - Ausgewählt. Das Skript ignoriert die Komponente. - Gelöscht. Das Skript wendet die Komponente an. Standardmäßig ist die Eigenschaft deaktiviert.
name	Eine beschreibende Kennzeichnung für die Komponente. Diese Kennzeichnung wird in der Protokolldatei sowie der Ereignisansicht angezeigt. Ermitteln Sie anhand der Eigenschaft name , welche Komponente das Ereignis verursacht hat.
Benachrichtigungen	Eine Liste mit NotificationHandler -Komponenten, die Benachrichtigungen aus geschachtelten Komponenten verarbeiten. Weitere Informationen hierzu finden Sie unter "Benachrichtigungen" auf Seite 430 .
on_fail	Die Aktion, die beim Ausfall der Komponente durchgeführt wird. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none"> - Gelöscht. Es wird keine Aktion ausgeführt. - CustomLog. Es wird in das Benutzerprotokoll geschrieben. - LogError. Eine Fehlermeldung wird in das Engine-Protokoll geschrieben. - LogInfo. Eine Informationsmeldung wird in das Engine-Protokoll geschrieben. - LogWarning. Eine Warnmeldung wird in das Engine-Protokoll geschrieben. - NotifyFailure. Eine Mitteilung wird gesendet. Die Standardoption lautet „Gelöscht“. Weitere Informationen über die Behandlung von Komponentenfehlern finden Sie in "Fehlerbehandlung" auf Seite 410 .
optional	Legt fest, ob ein Komponentenfehler den Ausfall der übergeordneten Komponente verursacht. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none"> - Ausgewählt. Ein Komponentenfehler führt nicht zum Ausfall der übergeordneten Komponente. - Gelöscht. Ein Komponentenfehler verursacht den Ausfall der übergeordneten Komponente. Standardwert ist "Gelöscht". Weitere Informationen über Komponentenfehler finden Sie in "Fehlerbehandlung" auf Seite 410 .
remark	Ein benutzerdefinierter Kommentar, der den Zweck oder die Aktion der Komponente beschreibt.
source	Definiert eine Locator-Komponente, die einen XML-Datenbehälter definiert. Der Datenbehälter enthält die Stammebene der XML-Quelle für das Mapping. Weitere Informationen hierzu finden Sie unter "Überblick über Locator, Keys und Indexierung" auf Seite 369 .
target	Definiert eine Locator-Komponente, die einen XML-Datenbehälter definiert. Der Datenbehälter enthält die Stammebene der Ausgabe-XML-Quelle für das Mapping. Weitere Informationen hierzu finden Sie unter "Überblick über Locator, Keys und Indexierung" auf Seite 369 .

RepeatingGroupMapping

Der **RepeatingGroupMapping**-Mapper-Anker verarbeitet eine wiederholte Struktur in der Eingabe oder Ausgabe.

Verwenden Sie ein **RepeatingGroupMapping**, wenn die XML-Eingabe oder -Ausgabe einen mehrfach vorkommenden Datenbehälter enthält. Er iteriert über die Instanzen der Datenbehälter. Weitere Informationen hierzu finden Sie unter ["Mehrfachinstanz-Datenbehälter" auf Seite 202](#).

Unter dem **RepeatingGroupMapping** werden Mapper-Anker und Aktionen geschachtelt, die die einzelnen Instanzen des Datenbehälters verarbeiten.

In der nachstehenden Tabelle werden die Eigenschaften des **RepeatingGroupMapping**-Mapper-Ankers beschrieben:

Eigenschaft	Beschreibung
count	Definiert die Anzahl der auszuführenden Iterationen. Wenn diese Eigenschaft leer bleibt, werden die Iterationen solange fortgesetzt, bis die Eingabe erschöpft ist.
current_iteration	Definiert einen Datenbehälter, in den die RepeatingGroupMapping die Nummer der aktuellen Iteration ausgibt.
disabled	Legt fest, ob die Komponente und alle zugehörigen untergeordneten Komponenten vom Skript ignoriert werden. Verwenden Sie diese Eigenschaft zum Testen, Debuggen und Ändern eines Skripts. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none"> - Ausgewählt. Das Skript ignoriert die Komponente. - Gelöscht. Das Skript wendet die Komponente an. Standardmäßig ist die Eigenschaft deaktiviert.
name	Eine beschreibende Kennzeichnung für die Komponente. Diese Kennzeichnung wird in der Protokolldatei sowie der Ereignisansicht angezeigt. Ermitteln Sie anhand der Eigenschaft name , welche Komponente das Ereignis verursacht hat.
Benachrichtigungen	Eine Liste mit NotificationHandler -Komponenten, die Benachrichtigungen aus geschachtelten Komponenten verarbeiten. Weitere Informationen hierzu finden Sie unter "Benachrichtigungen" auf Seite 430 .
on_fail	Die Aktion, die beim Ausfall der Komponente durchgeführt wird. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none"> - Gelöscht. Es wird keine Aktion ausgeführt. - CustomLog. Es wird in das Benutzerprotokoll geschrieben. - LogError. Eine Fehlermeldung wird in das Engine-Protokoll geschrieben. - LogInfo. Eine Informationsmeldung wird in das Engine-Protokoll geschrieben. - LogWarning. Eine Warnmeldung wird in das Engine-Protokoll geschrieben. - NotifyFailure. Eine Mitteilung wird gesendet. Die Standardoption lautet „Gelöscht“. Weitere Informationen über die Behandlung von Komponentenfehlern finden Sie in "Fehlerbehandlung" auf Seite 410 .
on_iteration_fail	Legt die Aktion fest, wenn eine Iteration fehlschlägt. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none"> - Gelöscht. Keine Aktion. - CustomLog. Schreibt in das Benutzerprotokoll. - LogError. Schreibt eine Fehlermeldung in das Engine-Protokoll. - LogInfo. Schreibt eine Informationsmeldung in das Engine-Protokoll. - LogWarning. Schreibt eine Warnmeldung in das Engine-Protokoll. - NotifyFailure. Löst eine Benachrichtigung aus. Die Eigenschaft on_iteration_fail wird zum Schreiben eines Eintrags verwendet, wenn eine einzelne Iteration fehlschlägt. Verwenden Sie die Eigenschaft on_fail , um einen Eintrag zu schreiben, falls die gesamte RepeatingGroupMapping fehlschlägt. Weitere Informationen hierzu finden Sie unter "Fehlerbehandlung" auf Seite 410 .

Eigenschaft	Beschreibung
optional	<p>Legt fest, ob ein Komponentenfehler den Ausfall der übergeordneten Komponente verursacht. Sie können eine der folgenden Optionen auswählen:</p> <ul style="list-style-type: none"> - Ausgewählt. Ein Komponentenfehler führt nicht zum Ausfall der übergeordneten Komponente. - Gelöscht. Ein Komponentenfehler verursacht den Ausfall der übergeordneten Komponente. Standardwert ist "Gelöscht". Weitere Informationen über Komponentenfehler finden Sie in "Fehlerbehandlung" auf Seite 410.
remark	Ein benutzerdefinierter Kommentar, der den Zweck oder die Aktion der Komponente beschreibt.
skip_failed_iterations	<p>Legt fest, ob fehlgeschlagene Iterationen übersprungen werden. Sie können eine der folgenden Optionen auswählen:</p> <ul style="list-style-type: none"> - Ausgewählt. RepeatingGroup überspringt eine fehlgeschlagene Iteration und fährt mit der nächsten Iteration fort. Wenn eine Iteration erfolgreich ist, ist auch RepeatingGroup. - Gelöscht. RepeatingGroup schlägt fehl, wenn eine Iteration fehlschlägt. <p>Die Eigenschaft skip_failed_iterations hat nur dann einen Effekt, wenn separator definiert ist. Standardwert ist "Ausgewählt".</p>
source	Definiert einen Datenbehälter, der die Quelle für die Mapping enthält. Weitere Informationen hierzu finden Sie unter "Überblick über Locator, Keys und Indexierung" auf Seite 369 .
target	Definiert einen Datenbehälter, der das Ergebnis für das Mapping enthält. Weitere Informationen hierzu finden Sie unter "Überblick über Locator, Keys und Indexierung" auf Seite 369 .

Beispiel

Weitere Informationen sowie ein Beispiel zu **RepeatingGroupMapping** enthält der Abschnitt ["Mapper: Beispiel" auf Seite 359](#).

KAPITEL 21

Lokatoren, Schlüssel und Indexierung

Dieses Kapitel umfasst die folgenden Themen:

- [Überblick über Locator, Keys und Indexierung, 369](#)
- [Lokatoren: Beispiel, 370](#)
- [Indexierung nach Schlüssel: Beispiel, 371](#)
- [Die Eigenschaften „source“ und „target“, 374](#)
- [Standardeigenschaften von Lokatoren und Schlüsseln, 381](#)
- [Die Komponenten Locator und Key: Referenz, 381](#)

Überblick über Locator, Keys und Indexierung

Beim Entwerfen einer Umwandlung tritt immer wieder das Problem auf, wie die zu verarbeitenden Datenbehälter ausfindig zu machen sind. Wenn dieselben Datenbehälter mehrere Male in einer XML-Struktur vorkommen können, sind die einzelnen Instanzen nicht immer eindeutig. In diesem Kapitel wird erläutert, wie Mehrdeutigkeiten mit Hilfe der Komponenten `Locator` und `Key` aufgelöst werden können.

Mit den in diesem Kapitel beschriebenen Komponenten können Sie die Instanzen von Mehrfachinstanz-Datenbehältern auf dreierlei Weise kennzeichnen:

- Sequenziell: Jede Iteration einer Komponente verarbeitet die jeweils nächste Instanz des Datenbehälters.
- Nach Instanznummer: Von einer Komponente kann zum Beispiel die dritte Instanz eines Datenbehälters ausgewählt werden.
- Nach Schlüssel. Ein Schlüssel kann ein Attribut oder ein geschachteltes Element sein. Durch den Schlüssel wird eine Instanz des Datenbehälters eindeutig bezeichnet.

Das sequenzielle Verfahren ist der Standard. Allerdings hat es einige komplexe Aspekte, die Sie mit der `Locator`-Komponente steuern können.

Die Kennzeichnung nach Instanznummer und Schlüssel wird als Indexierung bezeichnet. Die Indexierung funktioniert ähnlich wie das Stichwortverzeichnis eines Buchs, in dem Sie Informationen anhand einer Seitennummer oder eines Stichworts finden. Sie wird mit Hilfe der Komponenten `LocatorByOccurrence`, `LocatorByKey` und `Key` implementiert.

Die `Locator`- und `Key`-Komponenten können in Parsern, Serializern und Mappern verwendet werden. Mit diesen Komponenten können Sie Datenbehälter in der Eingabe, der Ausgabe oder beidem identifizieren.

Die Lokator-Komponenten werden in die Eigenschaften `source` und `target` verschiedener anderer Umwandlungskomponenten geschachtelt. Im Folgenden wird die Bedeutung und Verwendung der Eigenschaften `source` und `target` erläutert.

Lokatoren: Beispiel

Das folgende Beispiel veranschaulicht, welche Aspekte bei der Identifizierung von Datenbehältern wichtig sind. Das Beispiel zeigt die Verwendung

- der Eigenschaft `target`
- der `Locator`-Komponente

Das Beispiel wird hier zunächst grob skizziert. In den nachfolgenden Abschnitten wird dann genauer auf die Funktionsweise von `target` und `Locator` eingegangen.

Eingabe und Ausgabe

Angenommen, das Ausgabeschema eines Parsers unterstützt die folgende Struktur:

```
<Report>
  <Company>
    <Employee>John</Employee>
    <Employee>Leslie</Employee>
    <Employee>Pedro</Employee>
  </Company>
  <Company>
    <Employee>Marie</Employee>
    <Employee>Larry</Employee>
    <Employee>Frances</Employee>
  </Company>
</Report>
```

Das Quelldokument, das der Parser verarbeiten soll, ist eine Liste mit jeweils einem Angestellten pro Unternehmen:

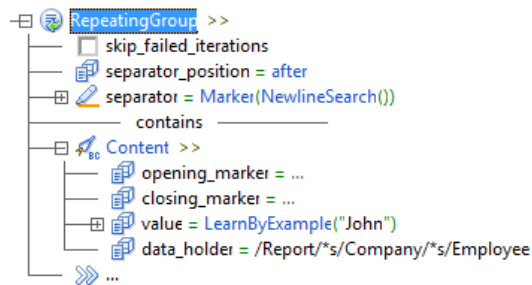
```
John
Marie
```

Der Parser soll folgende Ausgabe erzeugen:

```
<Report>
  <Company>
    <Employee>John</Employee>
  </Company>
  <Company>
    <Employee>Marie</Employee>
  </Company>
</Report>
```

Falsche Lösung

Angenommen, Sie parsen das Quelldokument mit der folgenden **RepeatingGroup**:



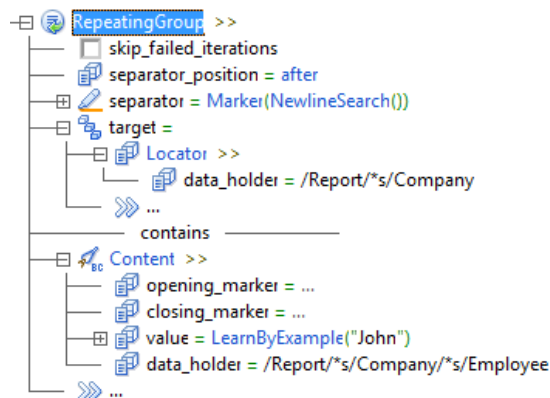
Dadurch erhalten Sie eine fehlerhafte Ausgabe:

```
<Report>
  <Company>
    <Employee>John</Employee>
    <Employee>Marie</Employee>
  </Company>
</Report>
```

Das Problem besteht darin, dass sowohl `Company` als auch `Employee` Mehrfachinstanz-Elemente sind. Die `RepeatingGroup` erzeugt zwar richtigerweise mehrere `Employee`-Elemente. Sie erkennt jedoch nicht, dass jedes `Employee`-Element in ein separates `Company`-Element geschachtelt werden soll.

Richtige Lösung

Sie können die Mehrdeutigkeit auflösen, indem Sie die Eigenschaft `target` der `RepeatingGroup` konfigurieren.



Die Eigenschaft `target` bezeichnet den Datenbehälter, den die `RepeatingGroup` erstellen soll. In der Eigenschaft `target` ist eine `Locator`-Komponente enthalten, die auf das Element `Company` verweist. Dies bewirkt, dass bei jeder Iteration der `RepeatingGroup` eine neue Instanz des `Company`-Elements erzeugt wird.

Wenn Sie die `RepeatingGroup` auf diese Weise konfigurieren, erhalten Sie die gewünschte Ausgabe:

Indexierung nach Schlüssel: Beispiel

Dieser Abschnitt veranschaulicht die Indexierung nach Schlüssel an einem Beispiel.

Das Beispiel zeigt einen Mapper, der mit Hilfe der Indexierung die Instanzen eines Datenbehälters in der Eingabe und der Ausgabe identifiziert. In der Eingabe führt die Indexierung zueinander gehörende Daten aus verschiedenen Teilen einer XML-Struktur zusammen. In der Ausgabe dient die Indexierung dem Auffinden der richtigen Position eines Elements in einer XML-Struktur.

Das Beispiel zeigt die Verwendung

- der Eigenschaften `source` und `target`
- der Komponenten `Locator`, `Key` und `LocatorByKey`

In den folgenden Abschnitten dieses Kapitels wird die Verwendung dieser Eigenschaften und Komponenten ausführlich erklärt.

Eingabe

Die XML-Eingabe ist ein Bericht mit Namen von Eltern und Kindern.

- Für die Eltern nennt der XML-Code einen Vornamen, einen Nachnamen und eine Kennung.
- Für die Kinder nennt der XML-Code einen Vornamen, ein Hobby und die Kennung der Eltern.

```
<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>
```

Ausgabe

Ausgegeben werden soll eine Liste der Hobbys und der Kinder, die dieses Hobby betreiben.

```
<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>
```

Konzept der Umwandlung

Die Umwandlung ist folgendermaßen konzipiert:

1. In der XML-Eingabe werden die zueinander gehörenden `Child`- und `Parent`-Elemente auf folgende Weise identifiziert:

`id`-Attribut von `Parent` = `parentID`-Attribut von `Child`

2. Die Datenumwandlung erzeugt die Elemente `Hobby` und `Person`. Sie identifiziert das `Hobby`-Element, in das das jeweilige `Person`-Element geschachtelt werden soll, folgendermaßen:

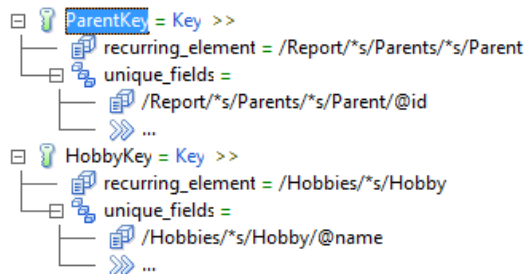
`name`-Attribut von `Hobby` = `hobby`-Attribut von `Child`.

3. Die Umwandlung schreibt den Vornamen des Kinds in das Element `Person`.
4. Die Umwandlung schreibt den Nachname der Eltern in das Element `Person`.

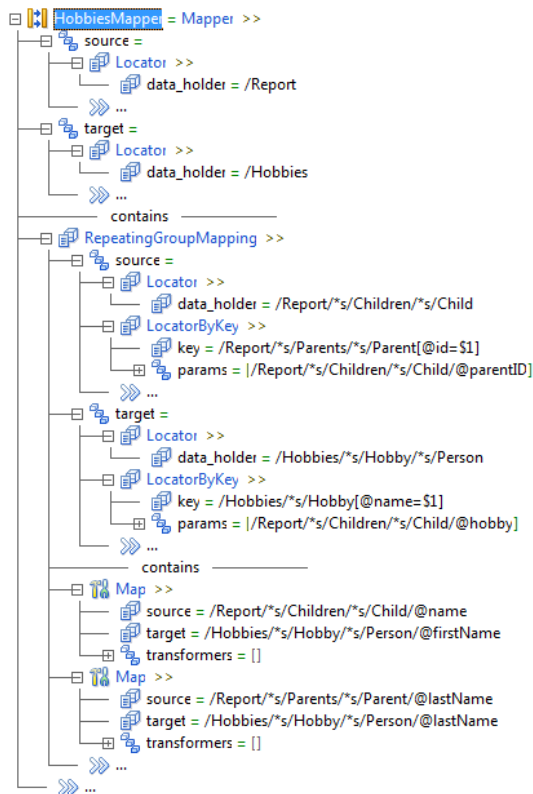
Mapperkonfiguration

Schlüssel-Komponenten definieren Bezeichner für die Datenbehälter.

- Die erste `Key`-Komponente gibt an, dass das Attribut `id` ein eindeutiger Bezeichner für das Element `Parent` ist.
- Die zweite `Key`-Komponente gibt an, dass das Attribut `name` ein eindeutiger Bezeichner für das Element `Hobby` ist.



Das Skript definiert dann einen Mapper mit der folgenden Konfiguration:



Die Komponenten des `Mappers` führen die folgenden Funktionen aus:

- Die Eigenschaft `source` von `RepeatingGroupMapping` gibt an, dass jede Iteration ihre Eingabe aus den folgenden Datenbehältern entnimmt:
 - aus einer Instanz des Elements `Child`
 - aus der entsprechenden Instanz des Elements `Parent`
- Die Eigenschaft `target` von `RepeatingGroupMapping` gibt an, dass jede Iteration ihre Ausgabe in den folgenden Datenbehältern speichert:
 - in einer Instanz des Elements `Person`
 - in der entsprechenden Instanz des Elements `Hobby`
- Die erste `Map`-Aktion kopiert das Attribut `name` von `Child` in das Attribut `firstName` von `Person`.
- Die zweite `Map`-Aktion kopiert das Attribut `lastName` von `Parent` in das Attribut `lastName` von `Person`.

Verwenden der Indexierung

In dem Beispiel werden die Instanzen der Datenbehälter `Parent` und `Hobby` durch Indexierung identifiziert.

- In der Eigenschaft `source` von `RepeatingGroupMapping` identifiziert die Indexierung diejenige Instanz von `Parent`, die zu einem `Child` gehört.
- In der Eigenschaft `target` identifiziert die Indexierung diejenige Instanz von `Hobby`, in die eine `Person` geschachtelt werden soll.

Die Eigenschaften „source“ und „target“

Die Eigenschaften `source` und `target` sind in den folgenden Komponenten vorhanden:

- In `Parsern`:
 - `Parser`
 - `Group`
 - `RepeatingGroup`
 - `EnclosedGroup`
 - `FindReplaceAnchor`
- In `Serializern`:
 - `Serializer`
 - `GroupSerializer`
 - `RepeatingGroupSerializer`
- In `Mappern`:
 - `Mapper`
 - `GroupMapping`
 - `RepeatingGroupMapping`

In allen diesen Kategorien haben die Eigenschaften dieselbe Bedeutung und werden gleich verwendet:

- Die Eigenschaft `source` bezeichnet vorhandene Datenbehälter, die bei einer Umwandlung verwendet werden sollen.
- Die Eigenschaft `target` kann Datenbehälter bezeichnen, die nicht notwendigerweise bereits vorhanden sein müssen. Wenn sie vorhanden sind, werden sie in der Umwandlung verwendet. Wenn sie nicht vorhanden sind, werden sie während der Umwandlung erstellt.

Nachdem `source` bzw. `target` definiert sind, werden die durch sie angegebenen Datenbehälter von den nachgeordneten Komponenten verwendet. Wenn Sie beispielsweise die Eigenschaft `target` einer `Group` definieren, verwenden die in die `Group` geschachtelten Anker die durch `target` identifizierten Datenbehälter.

Hinweis: Einige andere Komponenten, zum Beispiel `Map`, enthalten ebenfalls Eigenschaften namens `source` und `target`. Diese Eigenschaften haben eine andere Bedeutung als die oben genannten Eigenschaften und werden auf andere Art verwendet. Eine Erklärung zu diesen Eigenschaften finden Sie unter den betreffenden Komponenten.

Eigenschaft „source“

Die Eigenschaft **source** bezeichnet eine bestimmte Instanz eines Datenbehälters. Der Wert der Eigenschaft **source** ist eine Liste mit mindestens einer der folgenden Komponenten:

Quelle	Beschreibung
Locator	Identifiziert einen Einzelinstanz- oder Mehrfachinstanz-Datenbehälter. Bei Mehrfachinstanz-Datenbehältern greift jede Iteration auf die nächste Instanz in der Folge zu.
LocatorByKey	Identifiziert eine Instanz eines Mehrfachinstanz-Datenbehälters anhand eines Schlüssels.
LocatorByOccurence	Identifiziert eine Instanz eines Mehrfachinstanz-Datenbehälters anhand einer Nummer.

Standardverhalten

Falls die Eigenschaft `source` einer Komponente nicht konfiguriert ist, werden Datenbehälter auf folgende Weise identifiziert:

- Ist nur eine Instanz des Datenbehälters vorhanden, verwendet die Komponente diese Instanz.
- Sind mehrere Instanzen vorhanden, gelten folgende Verhaltensregeln:
 - In einem iterativen Kontext, zum Beispiel in einem `RepeatingGroupSerializer`, greift jede Iteration auf die nächstfolgende Instanz des Datenbehälters zu.
 - Wenn nicht iteriert wird, zum Beispiel in einem `GroupSerializer`, der nicht in eine iterative Komponente geschachtelt ist, greift die Komponente auf die erste Instanz des Datenbehälters zu.

Mehrdeutigkeiten im Standardverhalten

Im Standardverhalten kann es unter Umständen zu Mehrdeutigkeiten kommen. Mehrdeutigkeiten können beispielsweise in folgenden Situationen auftreten:

- Ein Mehrfachinstanz-Element ist in ein anderes Mehrfachinstanz-Element geschachtelt. Weitere Informationen hierzu finden Sie unter [“Beispiel 1: Geschachtelte Mehrfachinstanz-Datenbehälter” auf Seite 376](#).
- Das Schema lässt alternative Datenbehälter zu. Definition: `xs:choice`.
- Das Schema lässt zu, dass ein Datenbehälter fehlt. Definition: `minOccurs = 0`.

In solchen Fällen sollte die Eigenschaft `source` explizit zugewiesen werden.

Datenbehälter muss vorhanden sein

Die Eigenschaft `source` kann nur Datenbehälter identifizieren, die im Gültigkeitsbereich der Umwandlung bereits vorhanden sind. Existiert der Datenbehälter nicht, schlägt die Komponente, in der die Eigenschaft `source` enthalten ist, fehl.

Angenommen, die Eigenschaft „source“ einer `Group` enthält eine nicht-optionale `LocatorByOccurrence`-Komponente, die auf die dritte Instanz eines Datenbehälters verweist. Falls nur zwei Instanzen vorhanden sind, schlägt die `Group` fehl.

Verwenden der Eigenschaft „source“ für Eingabe und Ausgabe

Im Allgemeinen gibt die Eigenschaft `source` an, woher eine Komponente ihre Eingabe bezieht. Beispielsweise kann ein `GroupSerializer` mit dieser Eigenschaft die Instanz identifizieren, die serialisiert werden soll.

Außerdem kann mit dieser Eigenschaft angegeben werden, wo die Komponente ihre Ausgabe speichern soll. Angenommen, ein Parser hat bereits zehn Instanzen eines XML-Elements erzeugt. Nach der Erzeugung der Instanzen wird in einer der Instanzen des Elements von einem `Group`-Anker ein Attribut zugewiesen. Die `Group` kann nun anhand der Eigenschaft `source` diese Instanz identifizieren.

Beispiel 1: Geschachtelte Mehrfachinstanz-Datenbehälter

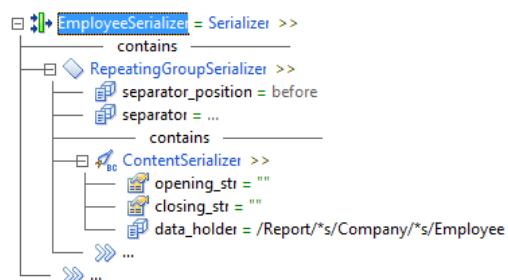
Angenommen, das Eingabeschema eines Serializers erfordert die folgende Struktur:

```
<Report>
  <Company>
    <Employee>John</Employee>
    <Employee>Leslie</Employee>
    <Employee>Pedro</Employee>
  </Company>
  <Company>
    <Employee>Marie</Employee>
    <Employee>Larry</Employee>
    <Employee>Frances</Employee>
  </Company>
</Report>
```

Die Komponente soll über alle `Employee`-Elemente iterieren und die folgende Ausgabe erzeugen:

```
John
Leslie
Pedro
Marie
Larry
Frances
```

Sie könnten einen `RepeatingGroupSerializer` erstellen und dahingehend konfigurieren, dass er den Datenbehälter `Mitarbeiter` ausgibt.

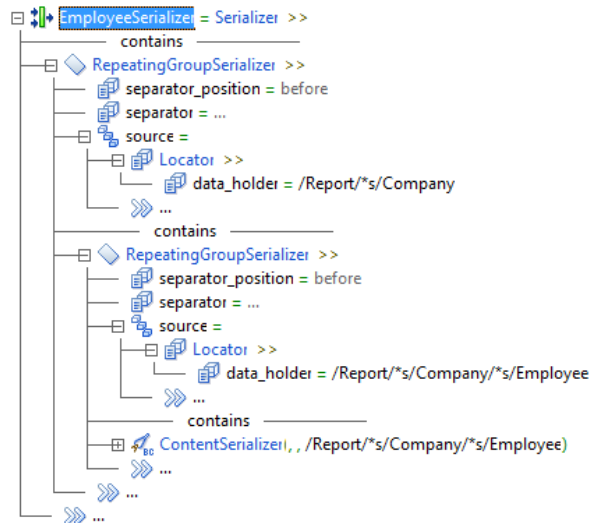


Das funktioniert jedoch nicht. Standardmäßig wird bei jeder Iteration eine neue Instanz von `Employee` innerhalb derselben `Company` ausgewählt. Ergebnis ist die folgende Ausgabe:

```
John
Leslie
Pedro
```

Mit anderen Worten: Der `RepeatingGroupSerializer` greift nur auf die erste `Company` zu.

Dieses Problem können Sie lösen, indem Sie den `RepeatingGroupSerializer` in einen weiteren `RepeatingGroupSerializer` schachteln. Zur Vermeidung potenzieller Mehrdeutigkeiten können Sie die `source`-Eigenschaften jeweils explizit konfigurieren.



Jede Iteration des äußeren `RepeatingGroupSerializer` verarbeitet eine andere Instanz von `Company`. Jede Iteration des geschachtelten `RepeatingGroupSerializer` verarbeitet eine andere Instanz von `Employee`. Nun erhalten Sie die gewünschte Ausgabe.

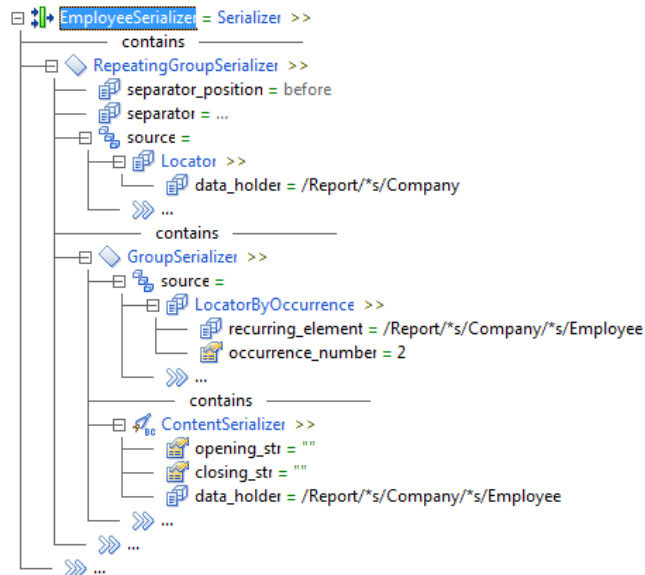
Ein weiteres Beispiel: Es soll nur über das zweite `Employee`-Element jeder `Company` iteriert werden. Die gewünschte Ausgabe lautet:

```

Leslie
Larry
  
```

Dieses Ergebnis erzielen Sie, indem Sie einen einzelnen `RepeatingGroupSerializer` konfigurieren, dessen Quelle `Company` ist. Dies bewirkt, dass jede Iteration auf die nächstfolgende Instanz von `Company` zugreift. Innerhalb der Iteration können Sie einen `GroupSerializer` konfigurieren, dessen Eigenschaft `source` mit Hilfe

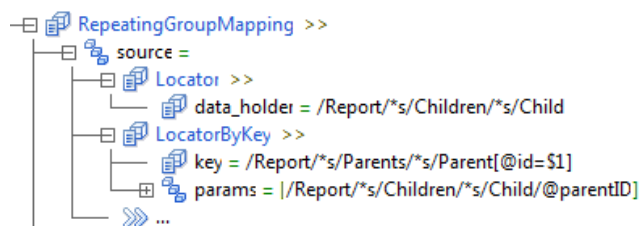
eines `LocatorByOccurrence` jeweils den gewünschten zweiten `Employee` auswählt. Dadurch erhalten Sie die gewünschte Ausgabe.



Beispiel 2: Indexierung

Im Beispiel für die Indexierung mittels Schlüssel am Anfang dieses Kapitels wurde ein `RepeatingGroupMapping` mit der unten gezeigten Konfiguration verwendet. In diesem Beispiel identifiziert die Eigenschaft `source` zwei Datenbehälter:

- Mit Hilfe einer `Locator`-Komponente identifiziert sie eine Instanz von `Child`. Bei jeder Iteration wird die nächstfolgende Instanz von `Child` verarbeitet.
- Mit Hilfe einer `LocatorByKey`-Komponente identifiziert sie eine Instanz von `Parent`. Dadurch wird bei jeder Iteration die Instanz von `Parent` verarbeitet, die zu der Instanz von `Child` gehört.



Weitere Informationen hierzu finden Sie unter ["Indexierung nach Schlüssel: Beispiel" auf Seite 371](#).

Eigenschaft „target“

Die Eigenschaft **target** kann eine Instanz eines Datenbehälters identifizieren, die nicht notwendigerweise bereits vorhanden sein muss. Wenn die Instanz existiert, wird sie von der Komponente verwendet. Wenn die Instanz nicht existiert, wird sie von der Komponente erstellt.

Der Wert der Eigenschaft **target** ist eine Liste mit mindestens einer der folgenden Komponenten:

Ziel	Beschreibung
Locator	Identifiziert einen einfach oder mehrfach vorkommenden Datenbehälter. Bei einem Mehrfachinstanz-Datenbehälter erzeugt die Iteration eine neue Instanz.
LocatorByKey	Identifiziert ein Vorkommen eines mehrfach vorkommenden Datenbehälters anhand eines Indexierungsschlüssels. Wenn die Instanz nicht existiert, wird sie erzeugt.
LocatorByOccurrence	Identifiziert ein Vorkommen eines mehrfach vorkommenden Datenbehälters anhand einer Nummer. Wenn die Instanz nicht vorhanden ist, wird sie mitsamt allen erforderlichen dazwischen kommenden Instanzen erzeugt. Wenn beispielsweise vier Instanzen vorhanden sind und LocatorByOccurrence die zehnte Instanz angibt, werden die Instanzen 5-9 ebenfalls erzeugt, aber leer gelassen.

Standardverhalten

Falls die Eigenschaft **target** einer Komponente nicht konfiguriert ist, werden Datenbehälter auf folgende Weise identifiziert:

- Wenn das Schema nur ein Vorkommen des Datenbehälters zulässt, greift das Skript auf dieses Vorkommen zu oder erzeugt es.
- Falls mehrere Instanzen des Datenbehälters vorhanden sein können, gelten folgende Verhaltensregeln:
 - In einem iterativen Kontext, zum Beispiel in einer **RepeatingGroup**, wird bei jeder Iteration eine neue Instanz des Datenbehälters erzeugt.
 - In einem nicht-iterativen Kontext, zum Beispiel in einer **Group**, die nicht in eine iterative Komponente geschachtelt ist, erzeugt die Komponente eine neue Instanz des Datenbehälters.

Mehrdeutigkeiten im Standardverhalten

Im Standardverhalten kann es unter Umständen zu Mehrdeutigkeiten kommen. Mehrdeutigkeiten können beispielsweise in folgenden Situationen auftreten:

- Ein Mehrfachinstanz-Element ist in ein anderes Mehrfachinstanz-Element geschachtelt. Weitere Informationen hierzu finden Sie unter ["Beispiel 1: Geschachtelte Mehrfachinstanz-Datenbehälter" auf Seite 380](#).
- Das Schema lässt alternative Datenbehälter zu. Definition: `xs:choice`.
- Das Schema lässt zu, dass ein Datenbehälter fehlt. Definition: `minOccurs = 0`.

In solchen Fällen sollte die Eigenschaft **target** explizit zugewiesen werden.

Datenbehälter können erzeugt werden

Die Eigenschaft **target** kann einen Datenbehälter identifizieren, der nicht notwendigerweise bereits im Gültigkeitsbereich der Umwandlung vorhanden sein muss. Wenn der Datenbehälter noch nicht existiert, wird er erzeugt.

Angenommen, die Eigenschaft **target** einer **Group** enthält einen **LocatorByKey**, der auf eine bestimmte Instanz eines Datenbehälters verweist. Falls die Instanz existiert, wird sie von der **Group** verwendet. Falls die Instanz nicht existiert, wird sie von der **Group** erzeugt.

Verwenden der Eigenschaft „target“ für Eingabe und Ausgabe

Im Allgemeinen gibt die Eigenschaft `target` an, wo die Ausgabe einer Komponente gespeichert werden soll. Beispielsweise kann eine `Group` mit dieser Eigenschaft die Instanz identifizieren, in der Daten gespeichert werden sollen.

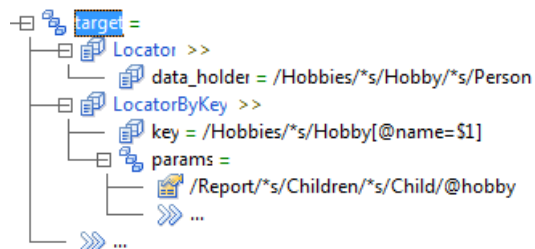
Außerdem kann mit dieser Eigenschaft angegeben werden, woher die Komponente ihre Eingabe beziehen soll. Angenommen, ein `GroupSerializer` enthält eine Aktion, die Daten berechnet und in einer Variablen speichert. Anschließend aktiviert der `GroupSerializer` einen `ContentSerializer`, der die Variable in die Ausgabe schreibt. Mit Hilfe der Eigenschaft `target` können Sie die Instanz der Variablen erzeugen, die der `GroupSerializer` verwenden soll. Die Variable dient dann als Eingabe für den `ContentSerializer`.

Beispiel 1: Geschachtelte Mehrfachinstanz-Datenbehälter

Im Suchbeispiel am Anfang dieses Kapitels wird gezeigt, wie Sie mit der Eigenschaft `target` zwischen Mehrfachinstanz-Datenbehältern der über- und untergeordneten Ebene unterscheiden können. Weitere Informationen hierzu finden Sie unter [“Lokatoren: Beispiel” auf Seite 370](#).

Beispiel 2: Indexierung

Im Beispiel für die Indexierung mittels Schlüssel am Anfang dieses Kapitels wird gezeigt, wie Sie die Eigenschaft `target` mit Indexierung verwenden. Die folgende Abbildung zeigt, wie die Eigenschaft `target` von `RepeatingGroupMapping` konfiguriert werden soll:



Die Eigenschaft `target` bezeichnet die folgenden Datenbehälter:

- Eine `Locator`-Komponente identifiziert eine Instanz von `Person`. Bei jeder Iteration wird eine neue Instanz von `Person` erzeugt.
- Mit einer `LocatorByKey`-Komponente wird die Instanz des Elements `Hobby` identifiziert, in die die Instanz von `Person` geschachtelt werden soll. Wenn das Element `Hobby` bereits vorhanden ist, wird es von der Umwandlung verwendet. Ist das Element `Hobby` noch nicht vorhanden, wird es von der Umwandlung erzeugt.

Weitere Informationen hierzu finden Sie unter [“Indexierung nach Schlüssel: Beispiel” auf Seite 371](#).

Standardeigenschaften von Lokatoren und Schlüsseln

In der folgenden Tabelle werden die Standardeigenschaften beschrieben, die in den Komponenten Lokator und Schlüssel verwendet werden:

Eigenschaft	Beschreibung
disabled	Legt fest, ob die Komponente und alle zugehörigen untergeordneten Komponenten vom Skript ignoriert werden. Verwenden Sie diese Eigenschaft zum Testen, Debuggen und Ändern eines Skripts. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none">- Ausgewählt. Das Skript ignoriert die Komponente.- Gelöscht. Das Skript wendet die Komponente an. Standardmäßig ist die Eigenschaft deaktiviert.
optional	Legt fest, ob ein Komponentenfehler den Ausfall der übergeordneten Komponente verursacht. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none">- Ausgewählt. Ein Komponentenfehler führt nicht zum Ausfall der übergeordneten Komponente.- Gelöscht. Ein Komponentenfehler verursacht den Ausfall der übergeordneten Komponente. Standardwert ist "Gelöscht". Weitere Informationen über Komponentenfehler finden Sie in "Fehlerbehandlung" auf Seite 410 .
remark	Ein benutzerdefinierter Kommentar, der den Zweck oder die Aktion der Komponente beschreibt.

Die Komponenten Locator und Key: Referenz

Locator- und Key-Komponenten identifizieren Elemente im Skript oder in Datenbehältern.

Schlüssel

Die Komponente **Key** (Schlüssel) definiert Attribute oder Elemente, die als eindeutige Bezeichner ihres jeweiligen Elternelements dienen.

Sie können eine **Key**-Komponente nur auf der globalen Ebene des Skripts definieren und **Key** überall im Skript referenzieren. Bei einem **Key**-Namen wird zwischen Groß- und Kleinschreibung unterschieden.

Die nachstehende Tabelle beschreibt die Eigenschaften der Komponente **Key**:

Eigenschaft	Beschreibung
disabled	Legt fest, ob die Komponente und alle zugehörigen untergeordneten Komponenten vom Skript ignoriert werden. Verwenden Sie diese Eigenschaft zum Testen, Debuggen und Ändern eines Skripts. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none">- Ausgewählt. Das Skript ignoriert die Komponente.- Gelöscht. Das Skript wendet die Komponente an. Standardmäßig ist die Eigenschaft deaktiviert.
recurring_element	Definiert ein Mehrfachinstanz-Element, dessen Instanzen durch den Schlüssel (Key-Komponente) bezeichnet werden.

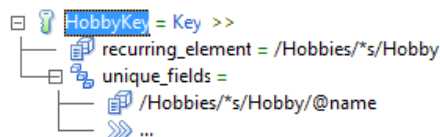
Eigenschaft	Beschreibung
remark	Ein benutzerdefinierter Kommentar, der den Zweck oder die Aktion der Komponente beschreibt.
unique_fields	Definiert den Schlüssel.

Beispiel

Im Beispiel für die Indexierung nach Schlüssel wird ein Schlüssel für das Element `Hobby` in der folgenden Struktur definiert:

```
<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>
```

Der Schlüssel ist das Attribut `name`, durch das jedes `Hobby` eindeutig bezeichnet wird.



Zusammengesetzte Schlüssel

Optional können Sie eine Liste mit Datenbehältern als einen zusammengesetzten Schlüssel definieren. Zu diesem Zweck schachteln Sie mehrere Datenbehälter unter der Eigenschaft `unique_fields`.

Betrachten Sie das folgende Beispiel:

```
<Persons>
  <Person ID="17" SubID="A">Bob</Person>
  <Person ID="17" SubID="B">Jane</Person>
  <Person ID="35" SubID="A">Larry</Person>
</Persons>
```

Weder das Attribut `ID` noch das Attribut `SubID` bezeichnen eindeutig ein bestimmtes `Person`-Element. Erst die Kombination aus `ID` und `SubID` bildet einen eindeutigen Bezeichner. Sie können `ID` und `SubID` als zusammengesetzten Schlüssel definieren.

Beschränkungen bei Schlüsseln

Die `unique_fields` müssen in `recurring_element` geschachtelt sein. Sie können entweder Attribute des Elements, geschachtelte Elemente auf einer beliebigen Schachtelungsebene oder Attribute der geschachtelten Elemente sein.

Beispielsweise wäre `Persons/Person/SocialSecurity/@Number` ein gültiger Schlüssel für `Persons/Person`, da `@Number` in `Persons/Person` geschachtelt ist. `Persons/Child` ist dagegen kein gültiger Schlüssel für `Persons/Person`, da die Schachtelung nicht korrekt ist.

Die `unique_fields` müssen den nächstliegenden Vorgänger bezeichnen, der mehrere Instanzen haben kann. Wenn zum Beispiel sowohl `Parent` als auch `Child` Mehrfachinstanz-Datenbehälter sind, wäre `Parent/Child/@name` zwar ein gültiger Schlüssel für `Parent/Child`, nicht aber für `Parent`.

Die `unique_fields` müssen einfache Datentypen haben. Sie können keine Strukturen sein.

Geschwisterinstanzen und nicht verschwisterte Instanzen

Ein Schlüssel bezeichnet Geschwisterinstanzen eines Elements eindeutig. Instanzen, die keine Geschwister sind, dürfen denselben Schlüssel haben.

Betrachten Sie die folgende XML-Struktur:

```
<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>
```

Das Attribut `ID` wäre ein gültiger Schlüssel für `Employee`, weil es einen `Employee` in einem einzelnen `Company`-Element eindeutig bezeichnet. Auch wenn die `ID`-Werte in anderen `Company`-Elementen ebenfalls vorkommen, ist dieser Schlüssel dennoch gültig.

Schlüssel wiederverwendbarer Elemente

Sie können einen Schlüssel für ein wiederverwendbares Element festlegen, das im XSD-Schema definiert ist.

Angenommen, `Persons/Person` kann im XML-Dokument in mehreren verschiedenen Zusammenhängen vorkommen. Wenn Sie „ID“ als Schlüssel für `Persons/Person` festlegen, ist der Schlüssel in jedem Kontext gültig, in dem `Persons/Person` verwendet wird.

Erzwungene Eindeutigkeit eines Schlüssels

Das Skript erzwingt die Eindeutigkeit eines **Schlüssels**. Dies hat die folgenden Konsequenzen:

- Wenn zwei oder mehr Geschwistervorkommen eines Eingabeelements dieselben Schlüsselwerte haben, geht das Skript davon aus, dass jedes Vorkommen die vorherigen Vorkommen überschreibt. Es verwendet nur die letzte Instanz, die es findet.
- Wenn in einer Instanz eines Eingabeelements ein Schlüsselwert fehlt, wird diese Instanz ignoriert.
- Wenn das Skript ein mit einem Schlüssel versehenes Element ausgibt und bereits ein Geschwisterelement mit demselben Schlüsselwert existiert, wird das vorhandene Vorkommen überschrieben.

In solchen Fällen schreibt das Skript einen Warnhinweis in das Ereignisprotokoll.

Locator

Die Komponente **Locator** identifiziert eine Instanz eines Einfach- oder Mehrinstanz-Datenbehälters in den Eigenschaften **source** und **target**. Bei einem Mehrfachinstanz-Datenbehälter verarbeitet jede Komponente, die **Locator** verwendet, jeweils die nächstfolgende Instanz des Datenbehälters.

Die nachstehende Tabelle beschreibt die Eigenschaften der Komponente **Locator**:

Eigenschaft	Beschreibung
data_holder	Der Datenbehälter, den die Locator -Komponente bezeichnet.
disabled	Legt fest, ob die Komponente und alle zugehörigen untergeordneten Komponenten vom Skript ignoriert werden. Verwenden Sie diese Eigenschaft zum Testen, Debuggen und Ändern eines Skripts. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none"> - Ausgewählt. Das Skript ignoriert die Komponente. - Gelöscht. Das Skript wendet die Komponente an. Standardmäßig ist die Eigenschaft deaktiviert.
optional	Legt fest, ob ein Komponentenfehler den Ausfall der übergeordneten Komponente verursacht. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none"> - Ausgewählt. Ein Komponentenfehler führt nicht zum Ausfall der übergeordneten Komponente. - Gelöscht. Ein Komponentenfehler verursacht den Ausfall der übergeordneten Komponente. Standardwert ist "Gelöscht". Weitere Informationen über Komponentenfehler finden Sie in "Fehlerbehandlung" auf Seite 410 .
remark	Ein benutzerdefinierter Kommentar, der den Zweck oder die Aktion der Komponente beschreibt.

LocatorByKey

Die Komponente **LocatorByKey** identifiziert eine Instanz eines Mehrinstanz-Datenbehälters in den Eigenschaften **source** und **target**.

Bevor Sie **LocatorByKey** verwenden können, müssen Sie einen **Key** auf der globalen Ebene des Skripts definieren. Der **Key** gibt die Datenbehälter an, welche die Instanz eindeutig bezeichnen.

Die nachstehende Tabelle beschreibt die Eigenschaften der Komponente **LocatorByKey**:

Eigenschaft	Beschreibung
disabled	Legt fest, ob die Komponente und alle zugehörigen untergeordneten Komponenten vom Skript ignoriert werden. Verwenden Sie diese Eigenschaft zum Testen, Debuggen und Ändern eines Skripts. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none"> - Ausgewählt. Das Skript ignoriert die Komponente. - Gelöscht. Das Skript wendet die Komponente an. Standardmäßig ist die Eigenschaft deaktiviert.
key	Definiert die XPath-Prädikatdarstellung eines Key-Objekts. Wenn Sie zum Beispiel <code>Hobbies/Hobby/@name</code> als Key definiert haben, können Sie <code>Hobbies/Hobby[@name=\$1]</code> auswählen. Diese Eigenschaft ist erforderlich.
optional	Legt fest, ob ein Komponentenfehler den Ausfall der übergeordneten Komponente verursacht. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none"> - Ausgewählt. Ein Komponentenfehler führt nicht zum Ausfall der übergeordneten Komponente. - Gelöscht. Ein Komponentenfehler verursacht den Ausfall der übergeordneten Komponente. Standardwert ist "Gelöscht". Weitere Informationen über Komponentenfehler finden Sie in "Fehlerbehandlung" auf Seite 410 .

Eigenschaft	Beschreibung
params	Definiert die Wert der Parameter im XPath-Prädikat. Jeder Wert hat ein Dollarzeichen (\$) und eine ganze Zahl, die die Position des Parameters in der Liste der Parameter ausweist. Diese Eigenschaft ist erforderlich.
remark	Ein benutzerdefinierter Kommentar, der den Zweck oder die Aktion der Komponente beschreibt.

Konflikte zwischen Lokatoren

Bei Konflikten werden Elternlokatoren durch geschachtelte **LocatorByKey**-Komponenten überschrieben.

Angenommen, die Eigenschaft **target** einer **Group** enthält einen **LocatorByKey**, der auf die dritte Instanz eines Elements verweist. Außerdem enthält eine geschachtelte **Group** einen **LocatorByKey**, der auf die fünfte Instanz verweist. In diesem Fall wird von der geschachtelten **Group** die fünfte Instanz verwendet.

LocatorByOccurrence

Die **LocatorByOccurrence** Komponente wird in der Eigenschaft **source** zur Identifizierung einer Instanz eines Mehrfachinstanz-Datenbehälters verwendet.

Beispiel:

- Ein Element, das mehrmals in einem XML-Dokument vorkommen kann
- Eine Variable, die mehrmals vorkommen kann

Die Komponente bezeichnet die Instanz anhand einer Nummer. Beispielsweise können Sie mit **LocatorByOccurrence** bei einem Datenbehälter mit zehn Instanzen festlegen, dass die dritte Instanz verarbeitet wird. **LocatorByOccurrence** kann für die Iteration über die Instanzen in einer sich wiederholenden Struktur, wie beispielsweise einem **RepeatingGroup**-Anker, verwendet werden.

Die Komponente **LocatorByKey** identifiziert eine Instanz eines Mehrinstanz-Datenbehälters in den Eigenschaften **source** und **target**.

Bevor Sie **LocatorByKey** verwenden können, müssen Sie einen **Key** auf der globalen Ebene des Skripts definieren. Der **Key** gibt die Datenbehälter an, welche die Instanz eindeutig bezeichnen.

Die nachstehende Tabelle beschreibt die Eigenschaften der Komponente **LocatorByOccurrence**:

Eigenschaft	Beschreibung
disabled	Legt fest, ob die Komponente und alle zugehörigen untergeordneten Komponenten vom Skript ignoriert werden. Verwenden Sie diese Eigenschaft zum Testen, Debuggen und Ändern eines Skripts. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none"> - Ausgewählt. Das Skript ignoriert die Komponente. - Gelöscht. Das Skript wendet die Komponente an. Standardmäßig ist die Eigenschaft deaktiviert.
occurrence_number	Definiert die Nummer der Instanz.

Eigenschaft	Beschreibung
optional	Legt fest, ob ein Komponentenfehler den Ausfall der übergeordneten Komponente verursacht. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none"> - Ausgewählt. Ein Komponentenfehler führt nicht zum Ausfall der übergeordneten Komponente. - Gelöscht. Ein Komponentenfehler verursacht den Ausfall der übergeordneten Komponente. Standardwert ist "Gelöscht". Weitere Informationen über Komponentenfehler finden Sie in "Fehlerbehandlung" auf Seite 410.
recurring_element	Definiert den Datenbehälter, den die Komponente bezeichnet.
remark	Ein benutzerdefinierter Kommentar, der den Zweck oder die Aktion der Komponente beschreibt.

Konflikte zwischen Lokatoren

Bei Konflikten werden Elternlokatoren durch geschachtelte **LocatorByOccurrence**-Komponenten überschrieben.

Angenommen, die Eigenschaft **target** einer **Group** enthält einen **LocatorByOccurrence**, der auf die dritte Instanz eines Elements verweist. Außerdem enthält eine geschachtelte **Group** einen **LocatorByOccurrence**, der auf die fünfte Instanz verweist. In diesem Fall wird von der geschachtelten **Group** die fünfte Instanz verwendet.

KAPITEL 22

Streamer

Dieses Kapitel umfasst die folgenden Themen:

- [Übersicht über Streamer, 387](#)
- [Text-Streamer, 388](#)
- [XML-Streamer, 392](#)
- [Standardeigenschaften von Streamern, 394](#)
- [Streamer-Komponente: Referenz, 395](#)
- [Streamer-Unterkomponente: Referenz, 404](#)

Übersicht über Streamer

Ein Streamer teilt große Quelldokumente in kleinere Teile auf, die separat in einer Umwandlung verarbeitet werden können. Streamer lassen sich insbesondere in Umwandlungen einsetzen, die sehr umfangreiche Eingaben verarbeiten, beispielsweise Datenströme von mehreren Gigabyte. Ein Streamer kann über eine Puffereingabe oder eine Dateieingabe verfügen.

Streamer bieten folgende Vorteile:

- Die Umwandlung parst jedes Quellesegment, wenn es verfügbar ist, anstatt zu warten, bis die gesamte Quelle verfügbar ist.
- Die Umwandlung hat geringere Speicheranforderungen.

Ein Streamer-Eingabestream enthält möglicherweise Transaktionsdaten für den Aktienmarkt. Der Datenstrom übermittelt kontinuierlich Daten an den Server im Verlauf eines kompletten Börsenhandelstages. Ein Skript mit einem Streamer verarbeitet jede Transaktion, wenn diese ankommt, anstatt auf das Ende des Tages zu warten.

Ein weiteres Beispiel: Sie empfangen eine umfangreiche Quelldatei über eine FTP-Verbindung. Mithilfe eines Streamers kann das Skript sofort die Verarbeitung der Datei starten, bevor diese vollständig übertragen wurde.

Die Datenprozessor-Umwandlung stellt die folgenden Arten von Streamern bereit:

- Streamer. Verarbeitet umfangreiche Texteingaben.
- XmlStreamer. Verarbeitet umfangreiche XML-Eingaben.

Streamer sind ausführbare Komponenten. Definieren Sie die **Streamer**- oder **XmlStreamer**-Komponente auf der globalen Skriptebene und legen Sie sie als Startkomponente der Umwandlung fest. Die Streamer-Funktion teilt die Eingabe in Segmente auf und übergibt diese an andere ausführbare Komponenten wie Parser, Mapper oder Serializer.

Text-Streamer

Eine `Streamer`-Komponente teilt große Quelldokumente in kleinere Teile auf. Der Streamer teilt die Textquelle in Header-, Footer- und wiederholende Segmente auf. Wenn es von der Quellstruktur vorgegeben ist, kann der `Streamer` die wiederholenden Segmente wiederum in geschachtelte Header-, Footer- und wiederholende Segmente unterteilen. Der `Streamer` kann jedes Segment einer entsprechenden Umwandlung übergeben.

Segmente

Ein Streamer kennzeichnet Segmente seiner Eingabe. Er übergibt diese Segmente einzeln an Umwandlungen wie Parser, Mapper oder Serializer, die die Segmentdaten verarbeiten.

Ein Streamer geht davon aus, dass die Quelle aus den folgenden Teilen besteht:

- Einem Header-Segment
- Einer beliebigen Anzahl von sich wiederholenden Segmenten
- Einem Footer-Segment

Für jeden Segmenttyp definiert der Streamer eine Umwandlung, die das Segment verarbeitet.

Die wiederholt vorkommenden Segmente können einfach oder komplex sein. Ein einfaches Segment ist eine einzelne Dateneinheit. Komplexe Segmente haben eigene geschachtelte Header, sich wiederholende Segmente und Footer.

Header und Footer sind immer einfache Segmente.

Einfache Segmente

Ein einfaches Segment hat einen öffnenden Marker, der seinen Beginn festlegt, und einen schließenden Marker, der sein Ende festlegt. Somit weist ein einfaches Segment die folgende Struktur auf:

```
Opening marker
Data
Closing marker
```

Der Streamer übergibt das Segment an die angegebene Umwandlungskomponente, beispielsweise einen Parser.

Optional können einzelne Marker aus der Streamerdefinition übersprungen werden. Beispiel:

- Wenn Sie den öffnenden Marker überspringen, wird der Anfang der Quelldatei als Beginn des Headers betrachtet.
- Wenn Sie den schließenden Marker überspringen, endet das Segment beim öffnenden Marker des nächsten Segments.

Komplexe Segmente

Ein komplexes Segment hat einen Header und Footer. Zwischen dem Header und dem Footer kann es eine beliebige Anzahl von geschachtelten einfachen Segmenten (Simple Segment) enthalten.

```
Header
Simple segment
Simple segment
Simple segment
Footer
```

Ein komplexes Segment (Complex Segment) kann auch geschachtelte komplexe Elemente enthalten, zum Beispiel:

```
Header
Complex segment
Complex segment
Complex segment
Footer
```

Sie können auch ein komplexes Segment definieren, das keinen Header oder Footer enthält, zum Beispiel:

```
Simple segment
Simple segment
Simple segment
```

Die geschachtelten einfachen Segmente müssen alle denselben Typ haben. Sie müssen alle durch dieselben öffnenden und schließenden Marker bezeichnet werden.

Beispiel

Ein Datenstrom enthält Börsentransaktionsdaten. Der Datenstrom weist folgende Struktur auf:

- Der Header beginnt mit dem String `yy-MM-dd/`, also einer Datumsangabe, gefolgt von einem Schrägstrich.
- Danach enthält der Header verschiedene Daten, gefolgt von dem String `ENDHEAD/`.
- Die sich wiederholenden Segmente beginnen mit dem String `TRANS HH:mm nnn/`, wobei `HH:mm` die Uhrzeit im 24-Stunden-Format und `nnn` eine Seriennummer beliebiger Länge ist.
- Der Datenstrom endet mit dem String `END/`.

Der folgende Beispieldatenstrom entspricht dieser Spezifikation. Die drei Punkte `...` stellen jeweils die zu parsenden Daten dar.

```
06-12-13/...ENDHEAD/TRANS 09:30 1...TRANS 09:30 2...TRANS 09:31 03...TRANS 09:32
14...END/
```

Diesen Datenstrom können Sie mit einem Streamer parsen, der die folgende Schemastruktur aufweist. Die Position der öffnenden und schließenden Marker wird ermittelt, indem nach einem bestimmten Muster oder String gesucht wird.

Segment	Typ	Öffnender Marker	Schließender Marker
Kopfzeile	Simple	<code>[0-9][0-9]-[0-9][0-9]-[0-9][0-9]/</code>	<code>ENDHEAD/</code>
Repeating	Simple	<code>TRANS [0-9][0-9]:[0-9][0-9] [0-9]+/</code>	<code>none</code>
Fußzeile	Simple	<code>END/</code>	<code>none</code>

Verketteten des Headers

Optional können Sie einen Streamer so konfigurieren, dass das Header-Segment mit jedem sich wiederholenden Segment verkettet wird. Der Streamer übergibt das verkettete Ergebnis an eine Umwandlung.

Angenommen, ein Streamer übergibt das sich wiederholende Segment an einen Parser. Die Quelle hat die folgende Struktur, wobei `Segment1` etc. Instanzen des wiederholenden Segments sind:

```
Header
Segment1
Segment2
Segment3
```

Bei Auswahl der Verkettungsoption sendet der Streamer folgende Daten an den Parser:

```
HeaderSegment1
HeaderSegment2
HeaderSegment3
```

Ausgabe eines Streamers

Vom Streamer wird für jedes Quellsegment ein unabhängiges Ausgabedokument erzeugt.

Ausgabe in der Design-Umgebung

Falls Sie den Streamer in der Design-Umgebung ausführen, werden die einzelnen Ausgabesegmente in eine einheitliche Ausgabedatei zusammengeführt.

Nehmen wir beispielsweise an, dass der Streamer jedes Segment an einen Parser übergibt. Die Ausgabe jedes Parsers ist ein XML-Dokument. Die zusammengeführte Ausgabe ist eine Folge von XML-Dokumenten, zum Beispiel:

```
<?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>
```

Diese Ausgabe stellt kein wohlgeformtes XML dar, da sie mehrere Root-Elemente enthält.

Einbetten der Ausgabe in ein Root-Tag

Sie können die kombinierte Ausgabe eines Streamers in einem Root-Tag einbetten, um die Ausgabe in eine wohlgeformte XML umzuwandeln.

Wählen Sie in den Einstellungen zur Datenprozessor-Umwandlung auf der Registerkarte „XML-Generierung“ die Option **XML-Wurzelement hinzufügen** aus und geben Sie den Namen des einbettenden Wurzelements ein.

Wenn Sie beispielsweise ein einbettendes Wurzelement namens `MyRoot` definieren, lautet die umgewandelte Ausgabe wie folgt:

```
<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>
```

Verwendung von Markern und Variablen in Streamern

Innerhalb einer `Streamer`-Komponente ist die Verwendung der Komponenten `Marker` und `Variable`, wie dies in anderen Umwandlungsarten geschieht, nicht möglich. Stattdessen müssen die öffnenden und

schließenden Marker von einfachen Segmenten mit Hilfe der Komponente `MarkerStreamer` definiert werden. Mit der Komponente `StreamerVariable` können Sie temporäre Daten speichern, die von allen Segmenten gemeinsam genutzt werden.

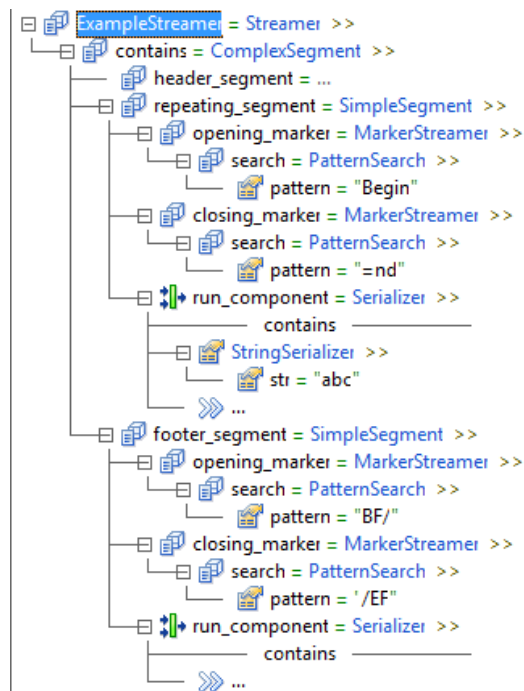
Erstellen eines Streamers

1. Analysieren Sie die Struktur der Quelle, und bestimmen Sie die Segmenttypen.
2. Erstellen oder öffnen Sie ein Skript.
3. Konfigurieren Sie im Skript eine Umwandlung, beispielsweise einen Parser, Mapper oder Serializer, der einfache Segmente jedes Typs verarbeiten kann.
4. Konfigurieren Sie in demselben Skript eine **Streamer**-Komponente.
5. Schachteln Sie in die **Streamer**-Komponente **ComplexSegment**- und **SimpleSegment**-Komponenten entsprechend der Quellstruktur.
6. Definieren Sie für jede **SimpleSegment**-Komponente die öffnenden und schließenden Marker, sofern erforderlich. Definieren Sie die Umwandlung, die das Segment verarbeitet.
7. Definieren Sie die **Streamer**-Komponente als Startkomponente.

Beispiel 1 für eine Konfiguration eines Streamers

Der folgende Streamer enthält einfache Segmente. Für jedes Segment ist ein öffnender und ein schließender Marker vordefiniert.

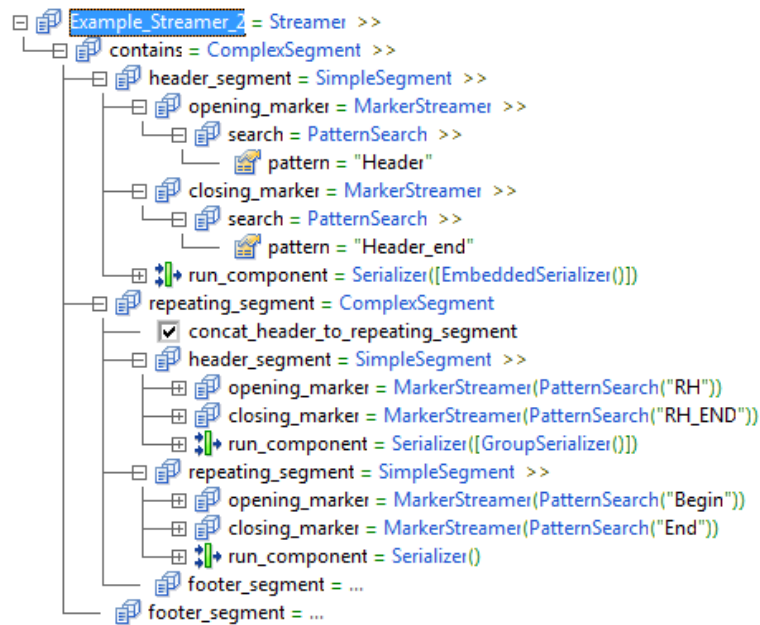
Der Streamer übergibt den Header und die sich wiederholenden Segmente an einen Parser mit dem Namen `body_p`. Den Footer übergibt er an einen Parser namens `foot_p`.



Beispiel 2 für eine Konfiguration eines Streamers

Der folgende Streamer enthält ein geschachteltes, sich wiederholendes `ComplexSegment`. Das geschachtelte `ComplexSegment`-Segment weist einen eigenen Header und ein geschachteltes, sich wiederholendes `SimpleSegment` auf. Das geschachtelte `ComplexSegment` besitzt keinen Footer.

Die Option `concat_header_to_repeating_segment` ist aktiviert. Diese Eigenschaft bewirkt, dass der Header mit jeder Instanz des sich wiederholenden Segments verkettet wird. Die verketteten Segmente werden vom Streamer an den Parser `body_p` übergeben.



XML-Streamer

Eine `XmlStreamer`-Komponente teilt große XML-Dokumente in kleinere Teile auf. Der `XmlStreamer` teilt die XML-Quelle in Header-, Footer- und Hauptteilsegmente auf. Die Hauptteilsegmente können wiederholende oder nicht wiederholende Segmente enthalten. Der `XmlStreamer` kann jedes XML-Segment einer entsprechenden Umwandlung übergeben, üblicherweise einem Mapper oder einem Serializer.

Ein `XmlStreamer` funktioniert ungefähr auf dieselbe Weise wie ein `Streamer`, abgesehen von einigen kleinen Unterschieden aufgrund der strukturierten XML-Eingabe. Um Folgenden werden die Hauptfunktionen aufgeführt:

- Die Hauptteilsegmente werden als XML-Elemente definiert. Sie können den Hauptteil mit mehreren Elementen desselben oder eines anderen Typs in jeder beliebigen Reihenfolge konfigurieren.
- Der Header wird als gesamter Teil der XML definiert, der dem ersten Hauptteilsegment vorausgeht. Bei der Konfiguration des `XmlStreamer` ist es nicht erforderlich, die Elemente zu definieren, die der Header umfasst.
- Der Footer wird als gesamter Teil der XML definiert, die auf den letzten Hauptteilsegment folgt. Bei der Konfiguration des `XmlStreamer` ist es nicht erforderlich, die Elemente zu definieren, die der Footer umfasst.

- Häufig stellen die Header- und die Footer-Segmente keine wohlgeformte XML dar. Damit die Segmente an einen Mapper oder einen Serializer übergeben werden können, können Sie Modifizierer-Komponenten konfigurieren, die die Segmente in wohlgeformte XML umwandeln.

Betrachten Sie für ein besseres Verständnis dieser Funktionen die folgende XML-Quellstruktur:

```
<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>
```

In diesem Beispiel können Sie die Hauptteilsegmente als `substream`-Elemente definieren. Der Header besteht aus allem, das dem ersten `substream` vorausgeht:

```
<stream>
  <headerline1>MainHeader</headerline1>
  <substreams>
```

Der Footer besteht aus allem, das auf den letzten `substream` folgt:

```
  </substreams>
  <footerline1>MainFooter</footerline1>
</stream>
```

Die Header- und die Footer-Segmente stellen keine wohlgeformte XML dar. Sie können Modifizierer anwenden, die schließende oder öffnende Tags hinzufügen, damit die XML wohlgeformt wird. Beispielsweise kann ein Modifizierer den Header in Folgendes umwandeln:

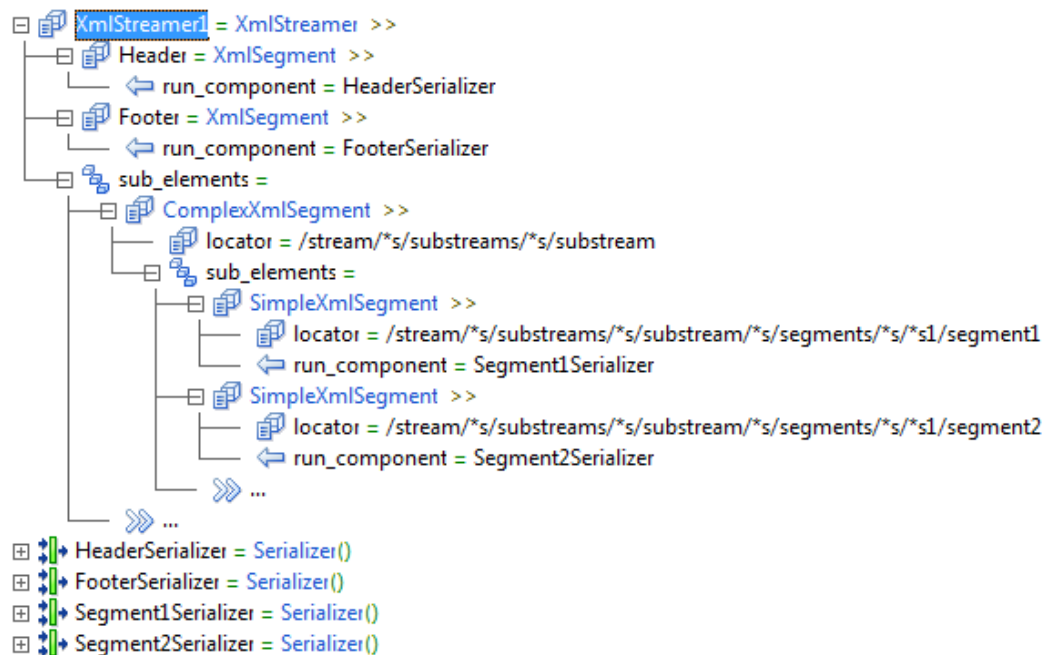
```
<stream>
  <headerline1>MainHeader</headerline1>
  <substreams>
  </substreams>
</stream>
```

Sie können den `XmlStreamer` so konfigurieren, dass das Header-Segment, das Footer-Segment und jede Instanz des `substream`-Segments einer entsprechenden Umwandlung übergeben wird, beispielsweise einem Mapper oder einem Serializer.

Hinweis: Die Header-Segmentelemente sind bei der Verarbeitung der Hauptteilelemente verfügbar. Die Footer-Elemente sind bei der Verarbeitung der Hauptteilelemente jedoch noch nicht verfügbar. Footer-Elemente werden nur verarbeitet, nachdem die Hauptteilelemente von der Umwandlung gelesen wurden.

Alternativ können Sie die `substream`-Elemente in die Segmente `segment1` und `segment2` unterteilen und jedes dieser Segmente an einen eigenen Mapper oder Serializer senden. Beachten Sie, dass `segment1` und `segment2` in einer zufälligen Reihenfolge aufeinander folgen. Der `XmlStreamer` ignoriert die Reihenfolge und verarbeitet `segment1` und `segment2` in der beliebigen Reihenfolge ihres Auftretens.

In der folgenden Abbildung wird die für diesen Zweck entsprechende Konfiguration dargestellt. Das Skript definiert unabhängige Serializer für die Header-, Footer-, `segment1`- und `segment2`-Segmente.



Hinweis: Auch wenn die Footer-Ausführungskomponente in diesem Beispiel vor den Hauptteilelementen angezeigt wird, werden Footer-Elemente erst dann verarbeitet, wenn die Hauptteilelemente von der Umwandlung gelesen wurden.

Für eine weitere Präzisierung können Sie Umwandlungen für die geschachtelten Header und Footer innerhalb der jeweiligen substream-Elemente definieren.

Standardeigenschaften von Streamern

In der folgenden Tabelle werden die allgemeinen Eigenschaften von Streamer-Komponenten beschrieben:

Eigenschaft	Beschreibung
disabled	Legt fest, ob die Komponente und alle zugehörigen untergeordneten Komponenten vom Skript ignoriert werden. Verwenden Sie diese Eigenschaft zum Testen, Debuggen und Ändern eines Skripts. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none"> - Ausgewählt. Das Skript ignoriert die Komponente. - Gelöscht. Das Skript wendet die Komponente an. Standardmäßig ist die Eigenschaft deaktiviert.
name	Eine beschreibende Kennzeichnung für die Komponente. Diese Kennzeichnung wird in der Protokolldatei sowie der Ereignisansicht angezeigt. Ermitteln Sie anhand der Eigenschaft name , welche Komponente das Ereignis verursacht hat.
remark	Ein benutzerdefinierter Kommentar, der den Zweck oder die Aktion der Komponente beschreibt.

Streamer-Komponente: Referenz

Der Streamer teilt große Quelldokumente in kleinere Teile auf, die separat in einer Umwandlung verarbeitet werden können.

ComplexSegment

Eine **ComplexSegment**-Komponente definiert eine Quellstruktur mit einem Header, einem sich wiederholenden Teil sowie einem Footer.

Die nachstehende Tabelle beschreibt die Eigenschaften der Komponente **ComplexSegment**:

Eigenschaft	Beschreibung
concat_header_to_repeating_segment	Legt fest, ob das System das header_segment mit jeder Instanz des repeating_segment enthält. Das Ergebnis wird der Ausführungskomponente (run_component) des repeating_segment zugewiesen. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none">- Ausgewählt. Jedes wiederholende Segment hat eine Kopie des Headers.- Gelöscht. Jedes wiederholende Segment wird ohne den Header angezeigt. Weitere Informationen hierzu finden Sie unter "Verketteten des Headers" auf Seite 389 . Standardwert ist "Gelöscht".
footer_segment	Definiert den Footer-Abschnitt der Quelle. Unter dieser Eigenschaft kann ein SimpleSegment geschachtelt werden, das den Footer definiert. Ist die Eigenschaft nicht definiert, verarbeitet das Skript die Quelle, als hätte sie keinen Footer.
header_segment	Definiert den Header-Abschnitt der Quelle. Unter dieser Eigenschaft kann ein SimpleSegment geschachtelt werden, das den Header definiert. Ist die Eigenschaft nicht definiert, verarbeitet das Skript die Quelle, als hätte sie keinen Header.
repeating_segment	Definiert den sich wiederholenden Abschnitt der Quelle. Unter dieser Eigenschaft kann ein SimpleSegment geschachtelt werden, das die sich wiederholenden Daten definiert. Alternativ kann darin auch ein ComplexSegment mit einer eigenen Header-Wiederholung-Footer-Struktur geschachtelt werden.

ComplexXmlSegment

Eine **ComplexXmlSegment**-Komponente definiert eine geschachtelte Struktur im Hauptteil einer **XmlStreamer**-Eingabe. Die geschachtelte Struktur kann einen eigenen Header, Body und Footer.

Unter einem **ComplexXmlSegment** können Sie **XmlSegment**-, **ComplexXmlSegment**- und **SimpleXmlSegment**-Komponenten schachteln.

Die nachstehende Tabelle beschreibt die Eigenschaften der Komponente **ComplexXmlSegment**:

Eigenschaft	Beschreibung
allow_unmarked_text	Legt fest, ob die Segmente in der sub_elements -Liste durch zwischengeschalteten Text oder andere Elemente getrennt werden können. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none"> - Ausgewählt. Die Segmente können durch zwischengeschalteten Text oder andere Elemente getrennt werden. Der zwischengeschaltete Inhalt wird ignoriert. - Gelöscht. Die Segmente dürfen nur durch Leerzeichen getrennt werden. Standardwert ist "Gelöscht".
disabled	Legt fest, ob die Komponente und alle zugehörigen untergeordneten Komponenten vom Skript ignoriert werden. Verwenden Sie diese Eigenschaft zum Testen, Debuggen und Ändern eines Skripts. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none"> - Ausgewählt. Das Skript ignoriert die Komponente. - Gelöscht. Das Skript wendet die Komponente an. Standardmäßig ist die Eigenschaft deaktiviert.
Footer	Definiert, wie der Footer des ComplexXmlSegment zu verarbeiten ist. Weitere Informationen hierzu finden Sie unter "XmlSegment" auf Seite 402 . Standardmäßig wird „XmlSegment“ verwendet.
Header	Definiert, wie der Header des ComplexXmlSegment zu verarbeiten ist. Weitere Informationen hierzu finden Sie unter "XmlSegment" auf Seite 402 . Standardmäßig wird „XmlSegment“ verwendet.
Locator	Definiert einen Datenbehälter.
name	Eine beschreibende Kennzeichnung für die Komponente. Diese Kennzeichnung wird in der Protokolldatei sowie der Ereignisansicht angezeigt. Ermitteln Sie anhand der Eigenschaft name , welche Komponente das Ereignis verursacht hat.
remark	Ein benutzerdefinierter Kommentar, der den Zweck oder die Aktion der Komponente beschreibt.
sub_elements	Definiert eine Liste von ComplexXmlSegment - oder SimpleXmlSegment -Komponenten, die definieren, wie der Body des ComplexXmlSegment zu verarbeiten ist.

JsonStreamer

Der **JsonStreamer** akzeptiert eine sehr umfangreiche JSON-Dateieingabe und unterteilt die Eingabe in Segmente. Er übergibt den jeweiligen Segmenttyp an eine vordefinierte Umwandlung, beispielsweise an einen Parser, Mapper oder Serializer.

Der **JsonStreamer** muss auf der globalen Ebene des Skripts definiert werden und als Startkomponente der Umwandlung fungieren.

Hinweis: Die Anzahl der Segmente wird automatisch anhand der Gesamtstellenanzahl des Eingabeports bestimmt.

In der folgenden Tabelle werden die Eigenschaften der Komponente **JsonStreamer** beschrieben:

Eigenschaft	Beschreibung
run_component	Definiert eine Umwandlung, die das Segment verarbeitet. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none">- Den Namen eines Parsers, eines Serializers oder eines Mappers, der auf der globalen Ebene des Skripts konfiguriert ist.- Eine Mapper- oder eine Serializer-Komponente. Konfigurieren Sie den Mapper oder Serializer innerhalb des Segments.- Eine WriteSegment-Komponente, die das Segment in die Ausgabe kopiert. Weitere Informationen hierzu finden Sie unter "WriteSegment" auf Seite 407.

MarkerStreamer

Eine **MarkerStreamer**-Komponente definiert die öffnenden und schließenden Marker von einfachen Segmenten. Er ähnelt einem gewöhnlichen **Marker**-Anker, wird aber nur in Streamern verwendet.

Die nachstehende Tabelle beschreibt die Eigenschaften der Komponente **MarkerStreamer**:

Eigenschaft	Beschreibung
adjacent	Legt fest, ob der MarkerStreamer an das Ende des vorherigen Segments angrenzen muss. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none">- Ausgewählt. Erfordert, dass MarkerStreamer an das Ende des vorherigen Segments angrenzt.- Gelöscht. MarkerStreamer kann vom Ende des vorherigen Segments getrennt sein. Standardwert ist "Gelöscht". Verwenden Sie diese Eigenschaft um sicherzustellen, dass die Segmente nicht durch sonstigen Text oder Leerräume getrennt werden.
count	Legt fest, mit welcher Instanz des Markers die Verarbeitung beginnt. Beispiel: Setzen Sie count auf 3, um die ersten beiden Instanzen des Markers zu überspringen. Diese Eigenschaft läuft aus. Sie ist zur Wahrung der Kompatibilität mit bestehenden Projekten verfügbar. Verwenden Sie sie nicht in neuen Projekten.
disabled	Legt fest, ob die Komponente und alle zugehörigen untergeordneten Komponenten vom Skript ignoriert werden. Verwenden Sie diese Eigenschaft zum Testen, Debuggen und Ändern eines Skripts. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none">- Ausgewählt. Das Skript ignoriert die Komponente.- Gelöscht. Das Skript wendet die Komponente an. Standardmäßig ist die Eigenschaft deaktiviert.
marking	Legt fest, ob der Marker als Referenzpunkt zur Kennzeichnung des darauf folgenden Segments oder Markers verwendet wird. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none">- Anfangsposition. Setzt nur vor dem aktuellen Anker einen Referenzpunkt.- Endposition. Setzt nur hinter dem aktuellen Anker einen Referenzpunkt.- Vollständig. Setzt vor und hinter dem aktuellen Marker einen Referenzpunkt. Standardwert ist "Vollständig".
name	Eine beschreibende Kennzeichnung für die Komponente. Diese Kennzeichnung wird in der Protokolldatei sowie der Ereignisansicht angezeigt. Ermitteln Sie anhand der Eigenschaft name , welche Komponente das Ereignis verursacht hat.

Eigenschaft	Beschreibung
remark	Ein benutzerdefinierter Kommentar, der den Zweck oder die Aktion der Komponente beschreibt.
search	Definiert, wie der MarkerStreamer Text findet. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none"> - NewlineSearch. Sucht ein Zeilenvorschubzeichen. - OffsetSearch. Überspringt eine vordefinierte Anzahl von Zeichen ab dem vorherigen Referenzpunkt. - PatternSearch. Sucht nach einem regulären Ausdruck. - TextSearch. Sucht nach einem angegebenen String.

Verwenden der Eigenschaft „marking“ zum Definieren der Segmentbegrenzungen

Über die Eigenschaft „marking“ können Sie steuern, ob die Daten im öffnenden und schließenden Marker in das Segment einbezogen und an eine Umwandlung übergeben werden.

Dabei gilt die Regel, dass der Streamer immer die Daten zwischen den innersten Referenzpunkten um das Segment übergibt. Beispiel:

- Wenn der öffnende Marker auf `marker = begin position` gesetzt ist, liegt der innerste Referenzpunkt am Beginn. In diesem Fall wird der gesamte Marker in das Segment einbezogen.
- Ist der öffnende Marker auf `marker = end position` oder auf `full` gesetzt, liegt der innerste Referenzpunkt am Ende. In diesem Fall wird der Marker aus dem Segment ausgeschlossen.

Für den schließenden Marker gelten die umgekehrten Angaben.

Betrachten Sie zur Veranschaulichung ein einfaches Segment mit folgender Struktur:

```
BEGIN...data...END
```

Ein **MarkerStreamer** identifiziert den öffnenden Marker, indem nach dem Text `BEGIN` sucht. Ein weiterer **MarkerStreamer** identifiziert den schließenden Marker, indem er nach dem Text `END` sucht.

In der folgenden Tabelle wird beschrieben, wie sich die Einstellungen der Eigenschaft „marking“ auf die Segmentbegrenzungen auswirken.

Einstellung des öffnenden Markers	Einstellung des schließenden Markers	An Umwandlung übergebenes Segment
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...

Einstellung des öffnenden Markers	Einstellung des schließenden Markers	An Umwandlung übergebenes Segment
end	begin	...data...
end	end	...data...END

SimpleSegment

Eine **SimpleSegment**-Komponente bezeichnet eine Dateneinheit mit einem öffnenden und einem schließenden Marker. Sie definiert außerdem die Umwandlung zur Verarbeitung der Dateneinheit.

Die öffnenden und schließenden Marker werden mit regulären Expressionen definiert. Weitere Informationen zur Syntax von regulären Expressionen finden Sie unter ["RegularExpression" auf Seite 286](#).

In der folgenden Tabelle werden die Eigenschaften der Komponente **SimpleSegment** beschrieben:

Eigenschaft	Beschreibung
closing_marker	Definiert eine reguläre Expression, die das Segmentende angibt. Wenn diese Eigenschaft nicht definiert ist, entspricht das Segmentende dem Ende der Quelle oder dem Beginn des nächsten Segments. Standardmäßig wird „MarkerStreamer“ verwendet.
count	Definiert die maximale Anzahl an Segmenten, die der Umwandlung übergeben werden. Wenn beispielsweise count „3“ lautet, sucht der Streamer nach drei aufeinanderfolgenden Instanzen des Segments. Er übergibt der Umwandlung die drei Segmente zusammen. Werden nur ein oder zwei Segmente gefunden, übergibt er diese Segmente. Wenn die Segmente klein sind, kann die Leistung durch die Übergabe von mehreren Segmenten an eine Umwandlung verbessert werden, da der Streamer-Overhead verringert wird. Verwenden Sie bei der Umwandlung eine Komponente wie RepeatingGroup , um die einzelnen Segmente zu verarbeiten. Standardwert ist 1.
disabled	Legt fest, ob die Komponente und alle zugehörigen untergeordneten Komponenten vom Skript ignoriert werden. Verwenden Sie diese Eigenschaft zum Testen, Debuggen und Ändern eines Skripts. Sie können eine der folgenden Optionen auswählen: - Ausgewählt. Das Skript ignoriert die Komponente. - Gelöscht. Das Skript wendet die Komponente an. Standardmäßig ist die Eigenschaft deaktiviert.
name	Eine beschreibende Kennzeichnung für die Komponente. Diese Kennzeichnung wird in der Protokolldatei sowie der Ereignisansicht angezeigt. Ermitteln Sie anhand der Eigenschaft name , welche Komponente das Ereignis verursacht hat.
opening_marker	Definiert eine reguläre Expression, die den Segmentbeginn angibt. Wenn diese Eigenschaft nicht definiert ist, entspricht der Segmentbeginn dem Anfang der Quelle oder dem Ende des vorherigen Segments. Standardmäßig wird „MarkerStreamer“ verwendet.

Eigenschaft	Beschreibung
remark	Ein benutzerdefinierter Kommentar, der den Zweck oder die Aktion der Komponente beschreibt.
run_component	Definiert eine Umwandlung, die das Segment verarbeitet. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none"> - Den Namen eines Parsers, eines Serializers oder eines Mappers, der auf der globalen Ebene des Skripts konfiguriert ist. - Eine Mapper- oder eine Serializer-Komponente. Konfigurieren Sie den Mapper oder Serializer innerhalb des Segments. - Eine WriteSegment-Komponente, die das Segment in die Ausgabe kopiert. Weitere Informationen hierzu finden Sie unter "WriteSegment" auf Seite 407.

SimpleXmlSegment

Eine **SimpleXmlSegment**-Komponente definiert ein Hauptteilsegment einer **XmlStreamer**-Eingabe. Sie definiert das Element, das das Segment enthält, und die Umwandlung, die das Segment verarbeiten soll.

Da ein **SimpleXmlSegment** ein XML-Element darstellt, ist das Segment immer wohlgeformt. Sie können einen Modifizierer anwenden, der das Segment ändert, bevor es der Umwandlung übergeben wird.

In der folgenden Tabelle werden die Eigenschaften der Komponente **SimpleXmlSegment** beschrieben:

Eigenschaft	Beschreibung
count	Definiert die maximale Anzahl an Segmenten, die der Umwandlung übergeben werden. Wenn beispielsweise count „3“ lautet, sucht der Streamer nach drei aufeinanderfolgenden Instanzen des Segments. Er übergibt der Umwandlung die drei Segmente zusammen. Werden nur ein oder zwei Segmente gefunden, übergibt er diese Segmente. Wenn die Segmente klein sind, kann die Leistung durch die Übergabe von mehreren Segmenten an eine Umwandlung verbessert werden, da der Streamer-Overhead verringert wird. Verwenden Sie bei der Umwandlung eine Komponente wie RepeatingGroup , um die einzelnen Segmente zu verarbeiten. Standardwert ist 1.
disabled	Legt fest, ob die Komponente und alle zugehörigen untergeordneten Komponenten vom Skript ignoriert werden. Verwenden Sie diese Eigenschaft zum Testen, Debuggen und Ändern eines Skripts. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none"> - Ausgewählt. Das Skript ignoriert die Komponente. - Gelöscht. Das Skript wendet die Komponente an. Standardmäßig ist die Eigenschaft deaktiviert.
Locator	Definiert einen Datenbehälter.
Modifizierer	Definiert, wie das Segment geändert wird, bevor es einer Umwandlung übergeben wird. Sie können die folgenden Modifiziererkomponenten auswählen: <ul style="list-style-type: none"> - AddHeaderModifier. Übergibt das Segment zusammen mit dem Header des XML-Abschnitts, in dem sich das Segment befindet. - AddStringModifier. Verkettet das Segment mit Präfix- oder Suffixstrings. - DoNothingModifier. Das Segment wird nicht geändert. - WellFormedModifier. Fügt schließende Tags und/oder ein Stammelement hinzu, um sicherzustellen, dass das Segment einer wohlgeformten XML entspricht. Weitere Informationen zu den Modifizierern finden Sie unter "Streamer-Unterkomponente: Referenz" auf Seite 404 . Standardmäßig wird „DoNothingModifier“ verwendet.
name	Eine beschreibende Kennzeichnung für die Komponente. Diese Kennzeichnung wird in der Protokolldatei sowie der Ereignisansicht angezeigt. Ermitteln Sie anhand der Eigenschaft name , welche Komponente das Ereignis verursacht hat.

Eigenschaft	Beschreibung
remark	Ein benutzerdefinierter Kommentar, der den Zweck oder die Aktion der Komponente beschreibt.
run_component	Definiert eine Umwandlung, die das Segment verarbeitet. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none"> - Den Namen eines Parsers, eines Serializers oder eines Mappers, der auf der globalen Ebene des Skripts konfiguriert ist. - Eine Mapper- oder eine Serializer-Komponente. Konfigurieren Sie den Mapper oder Serializer innerhalb des Segments. - Eine WriteSegment-Komponente, die das Segment in die Ausgabe kopiert. Weitere Informationen hierzu finden Sie unter "WriteSegment" auf Seite 407.

Streamer

Die Komponente **Streamer** teilt Texteingabe in Segmente auf. Sie übergibt den jeweiligen Segmenttyp an eine vordefinierte Umwandlung, beispielsweise an einen Parser, Mapper oder Serializer.

Der **Streamer** muss auf der globalen Ebene des Skripts definiert werden und die Startkomponente der Umwandlung sein.

In den **Streamer** muss eine **ComplexSegment**-Komponente geschachtelt werden. Das **ComplexSegment** kann geschachtelte **SimpleSegment**- oder **ComplexSegment**-Komponenten enthalten.

In der folgenden Tabelle werden die Eigenschaften der Komponente **Streamer** beschrieben:

Eigenschaft	Beschreibung
contains	Definiert die Gesamtstruktur der Quelle. Standardmäßig ist dies „ComplexSegment“.
disabled	Legt fest, ob die Komponente und alle zugehörigen untergeordneten Komponenten vom Skript ignoriert werden. Verwenden Sie diese Eigenschaft zum Testen, Debuggen und Ändern eines Skripts. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none"> - Ausgewählt. Das Skript ignoriert die Komponente. - Gelöscht. Das Skript wendet die Komponente an. Standardmäßig ist die Eigenschaft deaktiviert.
max_lookup_size	Die maximale Datenmenge in Kilobyte, die der Streamer für jedes neue Segment durchsucht. Legen Sie diese Eigenschaft auf ungefähr das Doppelte der maximal möglichen Segmentgröße fest, um die optimale Leistung zu erreichen. Wenn eine Anwendung einen bereitgestellten Streamer -Dienst über eine API aktiviert, muss der Chunk-Größenparameter auf einen Wert festgelegt werden, der kleiner als der Wert für max_lookup_size ist. Standard ist 10000.
name	Eine beschreibende Kennzeichnung für die Komponente. Diese Kennzeichnung wird in der Protokolldatei sowie der Ereignisansicht angezeigt. Ermitteln Sie anhand der Eigenschaft name , welche Komponente das Ereignis verursacht hat.
on_end_of_input	Definiert eine Umwandlung, die am Ende des Eingabestreams ausgeführt wird. Beispielsweise kann die Umwandlung eine zusammenfassende Meldung ausgeben. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none"> - Den Namen eines Parsers, eines Serializers oder eines Mappers, der auf der globalen Ebene des Skripts konfiguriert ist. - Eine Mapper- oder eine Serializer-Komponente. Konfigurieren Sie den Mapper oder Serializer innerhalb des Streamers.

Eigenschaft	Beschreibung
remark	Ein benutzerdefinierter Kommentar, der den Zweck oder die Aktion der Komponente beschreibt.
root_tag	Definiert ein XML-Tag, in das der Streamer die zusammengeführten Ausgaben aller Segmente einbettet. Weitere Informationen hierzu finden Sie unter "Ausgabe eines Streamers" auf Seite 390 .

StreamerVariable

Eine **StreamerVariable**-Komponente ist eine benutzerdefinierte Variable, die für alle Segmente einer **Streamer**- oder einer **XmlStreamer**-Komponente gültig ist.

Wenn der Streamer beispielsweise drei Parser enthält, ist der Wert der **StreamerVariable** für alle drei Parser verfügbar. Beispielsweise ruft ein Parser, der ein Header-Segment verarbeitet, möglicherweise Daten aus dem Header ab und speichert diese in einer **StreamerVariable**. Die anderen Parser, die das wiederholende Segment und das Footer-Segment verarbeiten, können auf den Wert der **StreamerVariable** zugreifen. Die Verwendung einer gewöhnlichen **Variable** ist für diesen Zweck nicht möglich, da der Wert der Variablen nicht von mehreren Segmenten genutzt werden kann.

Unter vielen anderen Aspekten ist die **StreamerVariable**-Komponente einer gewöhnlichen **Variable** sehr ähnlich. Eine **StreamerVariable** muss allerdings einen einfachen Einzelinstanz-Datentyp aufweisen. Weitere Informationen hierzu finden Sie unter ["Variablen" auf Seite 198](#).

Eine **StreamerVariable** kann nur auf der globalen Ebene des Skripts definiert werden.

In der folgenden Tabelle werden die Eigenschaften der Komponente **StreamerVariable** beschrieben:

Eigenschaft	Beschreibung
initialization	Zuordnung eines Anfangswerts zu einer StreamerVariable beim Start der Umwandlung. Wählen Sie InitialValue aus und geben Sie den gewünschten Wert ein.
val_type	Definiert den Datentyp, den die Variable speichern kann. Weisen Sie ihr einen einfachen Datentyp zu, beispielsweise <code>xs:string</code> oder <code>xs:integer</code> . Streamer-Variable dürfen keine komplexen oder mehrfach vorkommenden Typen aufweisen. Der Standardtyp lautet <code>xs:string</code> .

XmlSegment

Eine **XmlSegment**-Komponente definiert ein Header- oder Footer-Segment einer **XmlStreamer**-Eingabe. Sie definiert außerdem die Umwandlung zur Verarbeitung des Headers oder des Footers.

Ein nicht geänderter Header oder Footer ist nicht unbedingt eine wohlgeformte XML. Wenn Sie eine Modifizierer-Komponente zuweisen, können Sie das **XmlSegment** so konfigurieren, dass immer wohlgeformte XML zurückgegeben wird.

In der folgenden Tabelle werden die Eigenschaften der Komponente **XmlSegment** beschrieben:

Eigenschaft	Beschreibung
disabled	Legt fest, ob die Komponente und alle zugehörigen untergeordneten Komponenten vom Skript ignoriert werden. Verwenden Sie diese Eigenschaft zum Testen, Debuggen und Ändern eines Skripts. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none"> - Ausgewählt. Das Skript ignoriert die Komponente. - Gelöscht. Das Skript wendet die Komponente an. Standardmäßig ist die Eigenschaft deaktiviert.
Modifizierer	Definiert, wie das Segment geändert wird, bevor es einer Umwandlung übergeben wird. Sie können die folgenden Modifiziererkomponenten auswählen: <ul style="list-style-type: none"> - AddHeaderModifier. Übergibt das Segment zusammen mit dem Header des XML-Abschnitts, in dem sich das Segment befindet. - AddStringModifier. Verkettet das Segment mit Präfix- oder Suffixstrings. - DoNothingModifier. Das Segment wird nicht geändert. Dies ist die Standardeinstellung. - WellFormedModifier. Fügt schließende Tags und/oder ein Stammelement hinzu, um sicherzustellen, dass das Segment einer wohlgeformten XML entspricht. Weitere Informationen zu den Modifizierern finden Sie unter "Streamer-Unterkomponente: Referenz" auf Seite 404 . Die Standardeigenschaft lautet „WellFormedModifier“.
name	Eine beschreibende Kennzeichnung für die Komponente. Diese Kennzeichnung wird in der Protokolldatei sowie der Ereignisansicht angezeigt. Ermitteln Sie anhand der Eigenschaft name , welche Komponente das Ereignis verursacht hat.
remark	Ein benutzerdefinierter Kommentar, der den Zweck oder die Aktion der Komponente beschreibt.
run_component	Definiert eine Umwandlung, die das Segment verarbeitet. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none"> - Den Namen eines Parsers, eines Serializers oder eines Mappers, der auf der globalen Ebene des Skripts konfiguriert ist. - Eine Mapper- oder eine Serializer-Komponente. Konfigurieren Sie den Mapper oder Serializer innerhalb des Segments. - Eine WriteSegment-Komponente, die das Segment in die Ausgabe kopiert. Weitere Informationen hierzu finden Sie unter "WriteSegment" auf Seite 407.

XmlStreamer

Die Komponente **XmlStreamer** teilt eine XML-Eingabe in Header-, Footer- und Hauptteilsegmente auf. Sie übergibt den jeweiligen Segmenttyp an eine vordefinierte Umwandlung, beispielsweise an einen Mapper oder Serializer.

Der **XmlStreamer** muss auf der globalen Ebene des Skripts definiert werden und die Startkomponente der Umwandlung sein.

Unter einem **XmlStreamer** können die Komponenten **XmlSegment**, **ComplexXmlSegment** und **SimpleXmlSegment** geschachtelt werden.

In der folgenden Tabelle werden die Eigenschaften der Komponente **XmlStreamer** beschrieben:

Eigenschaft	Beschreibung
allow_unmarked_text	Legt fest, ob die Segmente in der sub_elements -Liste durch zwischengeschalteten Text oder andere Elemente getrennt werden können. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none"> - Ausgewählt. Die Segmente können durch zwischengeschalteten Text oder andere Elemente getrennt werden. Der zwischengeschaltete Inhalt wird ignoriert. - Gelöscht. Die Segmente dürfen nur durch Leerzeichen getrennt werden. Standardwert ist "Gelöscht".
disabled	Legt fest, ob die Komponente und alle zugehörigen untergeordneten Komponenten vom Skript ignoriert werden. Verwenden Sie diese Eigenschaft zum Testen, Debuggen und Ändern eines Skripts. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none"> - Ausgewählt. Das Skript ignoriert die Komponente. - Gelöscht. Das Skript wendet die Komponente an. Standardmäßig ist die Eigenschaft deaktiviert.
Footer	Definiert, wie der Footer des ComplexXmlSegment zu verarbeiten ist. Weitere Informationen hierzu finden Sie unter "XmlSegment" auf Seite 402 . Standardmäßig wird „XmlSegment“ verwendet.
Header	Definiert, wie der Header des ComplexXmlSegment zu verarbeiten ist. Weitere Informationen hierzu finden Sie unter "XmlSegment" auf Seite 402 . Standardmäßig wird „XmlSegment“ verwendet.
max_lookup_size	Die maximale Datenmenge in Kilobyte, die der XmlStreamer für jedes neue Segment durchsucht. Legen Sie diese Eigenschaft auf ungefähr das Doppelte der maximal möglichen Segmentgröße fest, um die optimale Leistung zu erreichen. Wenn eine Anwendung einen bereitgestellten XmlStreamer -Dienst über eine API aktiviert, muss der Chunk-Größenparameter auf einen Wert festgelegt werden, der kleiner als der Wert für max_lookup_size ist. Standard ist 10000.
name	Eine beschreibende Kennzeichnung für die Komponente. Diese Kennzeichnung wird in der Protokolldatei sowie der Ereignisansicht angezeigt. Ermitteln Sie anhand der Eigenschaft name , welche Komponente das Ereignis verursacht hat.
remark	Ein benutzerdefinierter Kommentar, der den Zweck oder die Aktion der Komponente beschreibt.
sub_elements	Definiert eine Liste mit ComplexXmlSegment - oder SimpleXmlSegment -Komponenten, die festlegen, wie der Hauptteil der XML-Eingabe zu verarbeiten ist.

Streamer-Unterkomponente: Referenz

Die Streamer-Unterkomponenten ändern Segmente einer Komponente **Streamer** oder einer Komponente **XmlStreamer**.

AddHeaderModifier

In einem **XmlStreamer** fügt die Komponente **AddHeaderModifier** den Header des aktuellen Segments dem Segment hinzu. Die Komponente fügt die schließenden Tags hinzu, die erforderlich sind um sicherzustellen, dass das Ergebnis wohlgeformtes XML ist.

Im Kontext dieses Headers können Sie mithilfe von **AddHeaderModifier** ein Segment an eine Umwandlung weitergeben.

Die nachstehende Tabelle beschreibt die Eigenschaften der Komponente **AddHeaderModifier**:

Eigenschaft	Beschreibung
disabled	Legt fest, ob die Komponente und alle zugehörigen untergeordneten Komponenten vom Skript ignoriert werden. Verwenden Sie diese Eigenschaft zum Testen, Debuggen und Ändern eines Skripts. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none">- Ausgewählt. Das Skript ignoriert die Komponente.- Gelöscht. Das Skript wendet die Komponente an. Standardmäßig ist die Eigenschaft deaktiviert.
name	Eine beschreibende Kennzeichnung für die Komponente. Diese Kennzeichnung wird in der Protokolldatei sowie der Ereignisansicht angezeigt. Ermitteln Sie anhand der Eigenschaft name , welche Komponente das Ereignis verursacht hat.
remark	Ein benutzerdefinierter Kommentar, der den Zweck oder die Aktion der Komponente beschreibt.

Im folgenden Beispiel ist `<segment1>` ein wiederholendes Segment mit vorhergehendem Header und nachfolgendem Footer.

```
<stream>
  <headerline1>...</headerline1>
  <segments>
    <segment1>...</segment1>
    <segment1>...</segment1>
    <segment1>...</segment1>
  </segments>
  <footerline1>...</footerline1>
</stream>
```

Sie können einen **XmlStreamer** konfigurieren, der den folgenden Header zurückgibt, bei dem es sich nicht um wohlgeformtes XML handelt:

```
<stream>
  <headerline1>...</headerline1>
  <segments>
```

Wenn Sie **AddHeaderModifier** auf `<segment1>` anwenden, versteht der Modifikator jede Instanz von `<segment1>` mit dem Header als Präfix. Er fügt schließende Tags hinzu, um sicherzustellen, dass das XML wohlgeformt ist. Ergebnis ist das folgende Segment:

```
<stream>
  <headerline1>...</headerline1>
  <segments>
    <segment1>...</segment1>
  </segments>
</stream>
```

Wenn Sie **AddHeaderModifier** auf ein Header-Segment anwenden, fügt der Modifikator den Header des übergeordneten Elements dem Segment hinzu. Wenden Sie **AddHeaderModifier** nicht auf den ersten Header der **XmlStreamer**-Eingabe an, da der erste Header selbst kein übergeordnetes Element hat.

AddStringModifier

In einem **XmlStreamer** fügt die Komponente **AddStringModifier** Strings vor und nach einem Segment hinzu.

Die nachstehende Tabelle beschreibt die Eigenschaften der Komponente **AddStringModifier**:

Eigenschaft	Beschreibung
disabled	Legt fest, ob die Komponente und alle zugehörigen untergeordneten Komponenten vom Skript ignoriert werden. Verwenden Sie diese Eigenschaft zum Testen, Debuggen und Ändern eines Skripts. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none">- Ausgewählt. Das Skript ignoriert die Komponente.- Gelöscht. Das Skript wendet die Komponente an. Standardmäßig ist die Eigenschaft deaktiviert.
name	Eine beschreibende Kennzeichnung für die Komponente. Diese Kennzeichnung wird in der Protokolldatei sowie der Ereignisansicht angezeigt. Ermitteln Sie anhand der Eigenschaft name , welche Komponente das Ereignis verursacht hat.
pre	Definiert den String vor dem Segment.
post	Definiert den String nach dem Segment.
remark	Ein benutzerdefinierter Kommentar, der den Zweck oder die Aktion der Komponente beschreibt.

DoNothingModifier

In einem **XmlStreamer** ist diese Komponente ein Platzhalter. Sie ändert das Segment nicht, auf das sie angewendet wird.

WellFormedModifier

In einem **XmlStreamer** stellt die Komponente **WellFormedModifier** sicher, dass ein Segment einer wohlgeformten XML entspricht. Zu diesem Zweck kann sie ggf. öffnende oder schließende Tags sowie Root-Tags hinzufügen.

In der folgenden Tabelle werden die Eigenschaften der Komponente **WellFormedModifier** beschrieben:

Eigenschaft	Beschreibung
disabled	Legt fest, ob die Komponente und alle zugehörigen untergeordneten Komponenten vom Skript ignoriert werden. Verwenden Sie diese Eigenschaft zum Testen, Debuggen und Ändern eines Skripts. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none">- Ausgewählt. Das Skript ignoriert die Komponente.- Gelöscht. Das Skript wendet die Komponente an. Standardmäßig ist die Eigenschaft deaktiviert.
name	Eine beschreibende Kennzeichnung für die Komponente. Diese Kennzeichnung wird in der Protokolldatei sowie der Ereignisansicht angezeigt. Ermitteln Sie anhand der Eigenschaft name , welche Komponente das Ereignis verursacht hat.

Eigenschaft	Beschreibung
new_root_element	Definiert das Stammelement. Das Stammelement muss ein Vorgänger des Segments sein. Wenn Sie diese Eigenschaft nicht zuweisen, fügt der Modifizierer kein Stammelement hinzu.
remark	Ein benutzerdefinierter Kommentar, der den Zweck oder die Aktion der Komponente beschreibt.

Betrachten Sie die folgende XML-Eingabe:

```
<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>
```

Angenommen, Sie konfigurieren einen **XmlStreamer**, der den folgenden Header zurückgibt, der keine wohlgeformte XML darstellt:

```
<substream>
  <subheaderline1>...</subheaderline1>
  <segments>
```

Wenn Sie **WellFormedModifier** auf diesen Header anwenden, fügt der Modifizierer die schließenden Tags hinzu. Nun erhalten Sie das folgende wohlgeformte Segment:

```
<substream>
  <subheaderline1>...</subheaderline1>
  <segments>
  </segments>
</substream>
```

Nehmen wir nun an, dass Sie **WellFormedModifier** so konfigurieren, dass als Stammelement `<stream>` hinzugefügt wird. Das Ergebnis sieht folgendermaßen aus:

```
<stream>
  <substreams>
    <substream>
      <subheaderline1>...</subheaderline1>
      <segments>
      </segments>
    </substream>
  </substreams>
</stream>
```

Beachten Sie, dass der Modifizierer die Elemente `<stream>` und `<substreams>` hinzugefügt hat, wobei die Grundstruktur der ursprünglichen XML beibehalten wurde.

WriteSegment

Die Komponente **WriteSegment** kopiert ein Segment in eine angegebene Ausgabeposition. Das kopierte Segment wird von der Komponente nicht geändert. Die Komponente **WriteSegment** stellt eine der Optionen für die Eigenschaft **run_component** der Komponente **XmlSegment** dar.

In der folgenden Tabelle werden die Eigenschaften der Komponente **WriteSegment** beschrieben:

Eigenschaft	Beschreibung
output	<p>Definiert die Ausgabeposition. Die Eigenschaft output weist die folgenden Optionen auf:</p> <ul style="list-style-type: none">- OutputDataHolder. Schreibt in einen Datenbehälter.- OutputFile. Schreibt in eine Datei.- OutputPort. Definiert den Namen eines AdditionalOutputPort, in dem die Daten geschrieben werden.- ResultFile. Schreibt in die Standardergebnisdatei der Umwandlung.- StandardErrorLog. Schreibt in das Benutzerprotokoll. Weitere Informationen hierzu finden Sie unter "Fehlerbehandlung" auf Seite 410. <p>Weitere Informationen zu diesen Optionen finden Sie unter "Aktions-Teilkomponenten: Referenz" auf Seite 338. Die Standardoption lautet „ResultFile“.</p>

KAPITEL 23

Validatoren, Benachrichtigungen und Fehlerbehandlung

Dieses Kapitel umfasst die folgenden Themen:

- [Überblick über Validatoren, Benachrichtigungen und Fehlerbehandlung, 409](#)
- [Fehlerbehandlung, 410](#)
- [Validatoren, 413](#)
- [Standardeigenschaften von Validatoren, 414](#)
- [Validator-Komponenten: Referenz, 414](#)
- [Benachrichtigungen, 430](#)
- [Referenz der Benachrichtigungskomponente, 431](#)

Überblick über Validatoren, Benachrichtigungen und Fehlerbehandlung

Beim Entwerfen einer Umwandlung müssen Sie die folgenden Fragen beachten:

- Was geschieht bei ungültigen Eingabedaten? Beispiel: Ein Datum hat das falsche Format, eine String ist zu lang oder die Datensätze sind nicht in der richtigen Reihenfolge.
- Was geschieht, wenn Daten in der Eingabe fehlen? Beispiel: In einer Adresse wird die Hausnummer ausgelassen.
- Was geschieht, wenn die Eingabe eine ungewöhnliche Struktur hat? Beispiel: Die Datensätze sind nicht in der richtigen Reihenfolge.

Alle diese Bedingungen können aufgrund eines Eingabefehlers auftreten. Und sie können Umwandlungsfehler und Fehlschläge verursachen.

Die Bedingungen können auch unter normalen Umständen auftreten. Beispiel: Ein Eingabeprotokoll erlaubt, dass bestimmte Felder fehlen.

Sie können Umwandlungsfunktionen integrieren, die solche Bedingungen ausmachen und entsprechende Aktionen ausführen. Mögliche Ansätze sind die folgenden Aktionen:

- Brechen Sie die Umwandlung ab und erstellen Sie keine Ausgabe.
- Brechen Sie die Umwandlung teilweise ab, machen Sie ihre Ausgabe rückgängig, aber erlauben Sie die Umwandlung zur Erstellung der Ausgabe für andere Teile der Daten.

- Lassen Sie die gesamte Umwandlung weiterlaufen, aber schreiben Sie eine Meldung in ein Benutzerprotokoll.
- Lassen Sie die gesamte Umwandlung weiterlaufen, aber schreiben Sie eine Meldung in die Ergebnisdatei der Umwandlung.

In diesem Kapitel wird erläutert, was bei einem Fehlschlag einer Umwandlung passiert, und wie man mit Fehlerbedingungen umgeht. Des weiteren wird erklärt, wie potenziell Fehlschläge verursachend Daten-Validierungsfehler ausgemacht werden können und wie Sie Benachrichtigungen über solche Bedingungen in die Ausgabe schreiben können.

Fehlerbehandlung

Ein Fehlschlag ist ein Ereignis, das eine Komponente daran hindert, Daten in der erwarteten Weise zu verarbeiten. Ein Anker schlägt beispielsweise fehl, wenn er nach Text sucht, der im Quelldokument nicht vorhanden ist. Ein Transformer oder eine Aktion schlagen möglicherweise fehl, wenn die Eingabe leer ist oder über einen ungeeigneten Datentyp verfügt.

Ein Fehlschlag kann ein völlig normales Ereignis sein. Ein Quelldokument kann beispielsweise ein optionales Datum enthalten. Ein Parser enthält einen **Content**-Anker, der das Datum (falls vorhanden) verarbeitet. Wenn das Datum in keinem bestimmten Quelldokument vorhanden ist, schlägt der **Content**-Anker fehl.

Durch entsprechendes Konfigurieren der Umwandlung können Sie das Ergebnis eines Fehlschlags steuern. Im obigen Beispiel könnten Sie den Parser so konfigurieren, dass die fehlenden Daten ignoriert werden und mit der Bearbeitung fortgefahren wird.

Das Ereignisprotokoll zeigt Warnungen über Fehlschläge an. Außerdem können Sie eine Umwandlung so konfigurieren, dass eine Fehlermeldung in ein Benutzerprotokoll geschrieben wird.

Verwendung der Eigenschaft „optional“ zur Fehlerbehandlung

Mithilfe der Eigenschaft `optional` können Sie das Verhalten einer Umwandlung bei einem Fehler steuern.

Fehler führt zu einem Fehler beim übergeordneten Element

Sofern die Eigenschaft **optional** einer Komponente nicht ausgewählt ist, hat ein Fehlschlagen der Komponente auch das Fehlschlagen der Elternkomponente zur Folge. Wenn die Elternkomponente ebenfalls nicht-optional ist, schlägt auch deren Elternkomponente fehl und so weiter.

Angenommen, ein **Parser** enthält eine **Group**, und die **Group** enthält einen **Marker**. Alle diese Komponenten sind nicht-optional. Wenn der **Marker** im Quelldokument nicht vorhanden ist, schlägt die **Marker**-Komponente fehl. Daraufhin schlägt auch die **Group** fehl, was wiederum zum Fehlschlagen des **Parser** führt.

Schematisch kann diese Beziehung folgendermaßen dargestellt werden:

```
Parser      //Fehlgeschlagen
Group       //Fehlgeschlagen
Marker      //Fehlgeschlagen
```

Optionaler Fehlschlag führt nicht zu einem Fehler beim übergeordneten Element

Sofern die Eigenschaft **optional** einer Komponente ausgewählt ist, zieht das Fehlschlagen einer Komponente nicht das Fehlschlagen der Elternkomponente nach sich.

Nehmen wir für das obige Beispiel an, dass die **Group** optional ist. Der fehlgeschlagene **Marker** bewirkt, dass die **Group** fehlschlägt. Der **Parser** wird jedoch erfolgreich ausgeführt.

```
Parser      //Erfolgreich
Group       //Fehlgeschlagen
Marker      //Fehlgeschlagen
```

Rollback

Wenn eine Komponente scheitert, werden ihre Auswirkungen rückgängig gemacht.

Angenommen, eine **Group** enthält drei nicht-optionale **Content**-Anker, die Werte in Datenbehältern speichern. Falls der dritte **Content**-Anker fehlschlägt, schlägt auch die **Group** fehl. Das Skript macht die Wirkungen der ersten beiden **Content**-Anker rückgängig. Die Daten, die von den ersten beiden **Content**-Ankern bereits in Datenbehältern gespeichert wurden, werden wieder entfernt.

Rückgängig gemacht werden nur die primären Wirkungen einer Umwandlung, etwa bei einem Parser das Speichern von Werten in Datenbehältern oder bei einem Serializer das Schreiben von Daten in die Ausgabedatei. Nebenwirkungen werden nicht rückgängig gemacht. Im obenstehenden Beispiel gilt: Wenn **Group** eine **WriteValue**-Aktion enthält, die eine Zeile in eine Textausgabedatei schreibt, wird die Zeile nicht gelöscht.

Festlegen der Eigenschaft „optional“

Sie können die Eigenschaft **optional** einer Komponente auf die folgenden Weisen festlegen:

- Bearbeiten Sie die erweiterten Eigenschaften einer Komponente im Skript.
- Klicken Sie mit der rechten Maustaste auf die Komponente und anschließend auf **Make Optional** (Optional machen) bzw. auf **Make Mandatory** (Obligatorisch machen).

Komponenten ohne Eigenschaft „optional“

Bei bestimmten Komponenten ist die Eigenschaft **optional** nicht vorhanden, da unabhängig von ihrer Eingabe kein Fehlschlagen möglich ist.

Ein Beispiel hierfür ist die Aktion **Sort**. Wenn die Aktion **Sort** keine Daten zum Sortieren findet, tut sie einfach überhaupt nichts. Daher wird auch kein Fehlschlag gemeldet.

Schreiben einer Fehlermeldung in das Benutzerprotokoll

Sie können eine Komponente so konfigurieren, dass Fehlerereignisse in ein benutzerdefiniertes Protokoll ausgegeben werden. Wenn ein Anker beispielsweise Text im Quelldokument nicht findet, kann eine Meldung im Benutzerprotokoll erstellt werden. Dies kann auch vorkommen, wenn der Anker als optional definiert ist, sodass die Umwandlung aufgrund des Fehlers nicht abgebrochen wird.

Das Benutzerprotokoll kann beispielsweise folgende Informationen enthalten:

- Fehlschlagebene: Information, Warnung oder Fehler
- Name der fehlgeschlagenen Komponente
- Beschreibung des Fehlers
- Position der fehlgeschlagenen Komponente im Skript
- Zusätzliche Informationen zum Status der Umwandlung, beispielsweise Werte von Datenbehältern.

Konfigurieren der Benutzerprotokollausgabe

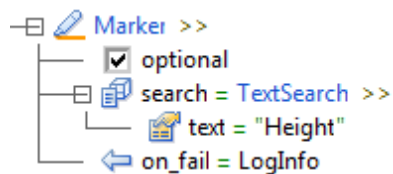
Um die Ausgabe des Benutzerprotokolls zu definieren, müssen Sie die Eigenschaft `on_fail` der entsprechenden Umwandlungskomponenten zuweisen. Die folgenden Komponenten haben eine `on_fail`-Eigenschaft:

- Parser und Anker
- Serializer und Serialisierungsanker
- Mapper und Mapper-Anker

Die `on_fail` Eigenschaft kann folgende Werte übernehmen:

- `LogError`. Schreibt eine Fehlermeldung mit der Systemvariablen `VarLastFailure` in das Benutzerprotokoll.
- `LogWarning`. Dasselbe wie `LogError`, die Meldung wird jedoch als Warnung und nicht als Fehler angezeigt.
- `LogInfo`. Dasselbe wie `LogError`, die Meldung wird jedoch als Information und nicht als Fehler angezeigt.
- `CustomLog`. Führt einen Serializer aus, der eine benutzerdefinierte Meldung in das Benutzerprotokoll oder an einen anderen Speicherort schreibt. Weitere Informationen hierzu finden Sie unter ["CustomLog" auf Seite 314](#).
- `NotifyFailure`. Löst eine Benachrichtigung aus.

Das nachstehende Beispiel veranschaulicht einen Marker-Anker mit einer `LogInfo`-Konfiguration:



Wenn der Marker im Quelldokument nicht vorhanden ist, schreibt das System folgenden Eintrag in das Benutzerprotokoll:

```
*** INFO *** : Marker, [MyParser[11].Marker], Can't find Marker<optional>('Height').
```

Anzeigen des Benutzerprotokolls

Das Benutzerprotokoll entspricht einer ASCII-Textdatei. Auf Windows-Plattformen lautet der standardmäßige Speicherort für Benutzerprotokolle wie folgt:

```
c:\Informatica\DataTransformation\UserLogs
```

Auf Unix-Plattformen lautet der standardmäßige Speicherort wie folgt:

```
<INSTALL_DIR>/UserLogs
```

Standardmäßig wird bei jeder Umwandlung ein Benutzerprotokoll mit einem eindeutigen Namen erstellt:

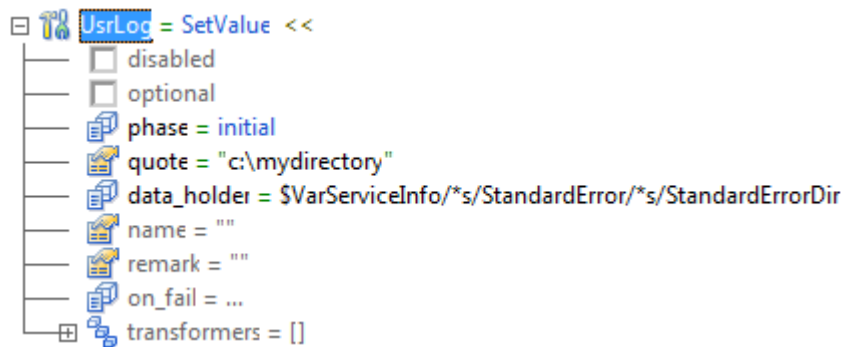
```
<service_name>+<unique_string>.log
```

Eine Umwandlung kann den Speicherort des Benutzerprotokolls während der Laufzeit festlegen, indem mithilfe der **SetValue**-Aktionen die folgenden Systemvariablen zugewiesen werden. Legen Sie die Eigenschaft

„phase“ von **SetValue** auf **initial** fest. Stellen Sie sicher, dass **SetValue** vor allen Komponenten ausgeführt wird, die in das Benutzerprotokoll schreiben.

Variable	Beschreibung
VarServiceInfo / StandardError > <i>StandardErrorDir</i>	Verzeichnispfad des Benutzerprotokolls
VarServiceInfo / StandardError > <i>StandardErrorName</i>	Dateiname des Benutzerprotokolls

Im folgenden Beispiel stellt eine **SetValue**-Aktion das Benutzerprotokollverzeichnis auf `c:\mydirectory` ein.



Validatoren

Eine Validator-Komponente stellt sicher, dass ihre Eingabe mit einer Bedingung übereinstimmt. Mithilfe von Validatoren können Sie die Eingabe hinsichtlich der maximalen oder minimalen Längen von Strings oder numerischen Werten, Konformität mit Ausdrücken und vielen weiteren Bedingungen überprüfen. Sie können auf dieselbe Eingabe mehrere Validatoren anwenden.

Wenn die Eingabe nicht mit der Bedingung übereinstimmt, löst der Validator eine Benachrichtigung aus. Die Benachrichtigung kann von einer **NotificationHandler**-Komponente verarbeitet werden. Wenn Sie beispielsweise Validatoren in einem Parser verwenden, kann ein **NotificationHandler** eine Warnmeldung in die Ausgabe des Parsers einfügen. Weitere Informationen hierzu finden Sie unter ["Benachrichtigungen" auf Seite 430](#).

Sie können Validatoren in Positionen einfügen, beispielsweise in die Eigenschaft **validators** eines **Content**-Ankers oder einer **Map**-Aktion. Die Validatoren ermöglichen es, Sie zu warnen, wenn die Eingabe ungültig ist, ohne dass der **Content**-Anker oder die **Map**-Aktion zwangsläufig fehlschlagen muss.

Zusätzlich zu den in diesem Kapitel beschriebenen Validatoren können Sie Daten anhand einer Reihe von benutzerdefinierten Regeln auswerten und einen XML-Validierungsbericht erzeugen. Weitere Informationen hierzu finden Sie unter ["ValidateValue" auf Seite 335](#).

Standardeigenschaften von Validatoren

In der folgenden Tabelle werden die Standardeigenschaften von Validatoren beschrieben:

Eigenschaft	Beschreibung
allow_empty_value	Legt fest, ob eine leere Eingabe als gültig akzeptiert wird. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none">- Ausgewählt. Leere Eingabe ist gültig.- Gelöscht. Leere Eingabe ist ungültig. Die Standardoption lautet „Gelöscht“.
disabled	Legt fest, ob die Komponente und alle zugehörigen untergeordneten Komponenten vom Skript ignoriert werden. Verwenden Sie diese Eigenschaft zum Testen, Debuggen und Ändern eines Skripts. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none">- Ausgewählt. Das Skript ignoriert die Komponente.- Gelöscht. Das Skript wendet die Komponente an. Standardmäßig ist die Eigenschaft deaktiviert.
name	Eine beschreibende Kennzeichnung für die Komponente. Diese Kennzeichnung wird in der Protokolldatei sowie der Ereignisansicht angezeigt. Ermitteln Sie anhand der Eigenschaft name , welche Komponente das Ereignis verursacht hat.
negation	Legt fest, ob die Validierungsbedingung negiert wird. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none">- Ausgewählt. Wenn die Bedingung "true" ist, ist die Eingabe nicht gültig. Wenn die Bedingung "false" ist, ist die Eingabe gültig.- Gelöscht. Wenn die Bedingung "true" ist, ist die Eingabe gültig. Wenn die Bedingung "false" ist, ist die Eingabe nicht gültig. Die Standardoption lautet „Gelöscht“.
notify	Definiert den Namen einer Benachrichtigung. Wenn die Eingabe nicht mit der Validierungsbedingung übereinstimmt, löst der Validator eine Benachrichtigung aus. Weitere Informationen hierzu finden Sie unter "Benachrichtigungen" auf Seite 430 . Die Standardoption lautet „Gelöscht“.
optional	Legt fest, ob ein Komponentenfehler den Ausfall der übergeordneten Komponente verursacht. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none">- Ausgewählt. Ein Komponentenfehler führt nicht zum Ausfall der übergeordneten Komponente.- Gelöscht. Ein Komponentenfehler verursacht den Ausfall der übergeordneten Komponente. Standardwert ist "Gelöscht". Weitere Informationen über Komponentenfehler finden Sie in "Fehlerbehandlung" auf Seite 410 .
remark	Ein benutzerdefinierter Kommentar, der den Zweck oder die Aktion der Komponente beschreibt.
Transformer	Definiert eine Liste mit Transformern, die für die Eingabe gelten. Die Validierungsbedingung wird auf das Ergebnis der Transformer angewendet. Die Transformer haben lediglich zum Zweck der Validierung eine temporäre Auswirkung auf die Daten. Die Eingabe wird nicht permanent geändert.

Validator-Komponenten: Referenz

Validator-Komponenten testen Eingabedaten hinsichtlich Konformität zu definierten Regeln.

AlternativeValidators

Der **AlternativeValidators**-Validator enthält eine Gruppe von geschachtelten Validatoren, die auf die Eingabe angewendet werden. Verwenden Sie einen **AlternativeValidators**, um die ODER-Logik auf eine Gruppe von Validierungsbedingungen anzuwenden. Die Daten sind gültig, wenn eine der Bedingungen erfüllt ist.

Die nachstehende Tabelle beschreibt die Eigenschaften des Validators **AlternativeValidators**:

Eigenschaft	Beschreibung
allow_empty_value	Legt fest, ob eine leere Eingabe als gültig akzeptiert wird. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none">- Ausgewählt. Leere Eingabe ist gültig.- Gelöscht. Leere Eingabe ist ungültig. Die Standardoption lautet „Gelöscht“.
disabled	Legt fest, ob die Komponente und alle zugehörigen untergeordneten Komponenten vom Skript ignoriert werden. Verwenden Sie diese Eigenschaft zum Testen, Debuggen und Ändern eines Skripts. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none">- Ausgewählt. Das Skript ignoriert die Komponente.- Gelöscht. Das Skript wendet die Komponente an. Standardmäßig ist die Eigenschaft deaktiviert.
name	Eine beschreibende Kennzeichnung für die Komponente. Diese Kennzeichnung wird in der Protokolldatei sowie der Ereignisansicht angezeigt. Ermitteln Sie anhand der Eigenschaft name , welche Komponente das Ereignis verursacht hat.
negation	Legt fest, ob die Validierungsbedingung negiert wird. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none">- Ausgewählt. Wenn die Bedingung "true" ist, ist die Eingabe nicht gültig. Wenn die Bedingung "false" ist, ist die Eingabe gültig.- Gelöscht. Wenn die Bedingung "true" ist, ist die Eingabe gültig. Wenn die Bedingung "false" ist, ist die Eingabe nicht gültig. Die Standardoption lautet „Gelöscht“.
notify	Definiert den Namen einer Benachrichtigung. Wenn die Eingabe nicht mit der Validierungsbedingung übereinstimmt, löst der Validator eine Benachrichtigung aus. Weitere Informationen hierzu finden Sie unter "Benachrichtigungen" auf Seite 430 . Die Standardoption lautet „Gelöscht“.
optional	Legt fest, ob ein Komponentenfehler den Ausfall der übergeordneten Komponente verursacht. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none">- Ausgewählt. Ein Komponentenfehler führt nicht zum Ausfall der übergeordneten Komponente.- Gelöscht. Ein Komponentenfehler verursacht den Ausfall der übergeordneten Komponente. Standardwert ist "Gelöscht". Weitere Informationen über Komponentenfehler finden Sie in "Fehlerbehandlung" auf Seite 410 .
remark	Ein benutzerdefinierter Kommentar, der den Zweck oder die Aktion der Komponente beschreibt.

Eigenschaft	Beschreibung
selector	<p>Legt das Kriterium für die Auswahl eines Validators aus den unter der Komponente AlternativeValidators geschachtelten Validatoren fest. Sie können eine der folgenden Optionen auswählen:</p> <ul style="list-style-type: none"> - ScriptOrder. Der Parser testet die geschachtelten Validatoren in der im Skript definierten Reihenfolge. Der erste erfolgreich geprüfte Validator wird akzeptiert. Gibt es keinen erfolgreich geprüften Validator, ist die Eingabe ungültig. - NameSwitch. Der Parser sucht den geschachtelten Validator, dessen name-Eigenschaft in dem in option_name definierten Datenbehälter angegeben ist. Die übrigen Validatoren werden ignoriert. Schlägt die Prüfung des Validators fehl, ist die Eingabe ungültig. <p>Standardwert ist ScriptOrder.</p>
Transformer	<p>Definiert eine Liste mit Transformern, die für die Eingabe gelten. Die Validierungsbedingung wird auf das Ergebnis der Transformer angewendet. Die Transformer haben lediglich zum Zweck der Validierung eine temporäre Auswirkung auf die Daten. Die Eingabe wird nicht permanent geändert.</p>

EDIFACTValidation

Der Validator **EDIFACTValidation** testet, ob ein Quell-String eine gültige EDIFACT-Nachricht ist.

Die nachstehende Tabelle beschreibt die Eigenschaften des Validators **EDIFACTValidation**:

Eigenschaft	Beschreibung
disabled	<p>Legt fest, ob die Komponente und alle zugehörigen untergeordneten Komponenten vom Skript ignoriert werden. Verwenden Sie diese Eigenschaft zum Testen, Debuggen und Ändern eines Skripts. Sie können eine der folgenden Optionen auswählen:</p> <ul style="list-style-type: none"> - Ausgewählt. Das Skript ignoriert die Komponente. - Gelöscht. Das Skript wendet die Komponente an. <p>Standardmäßig ist die Eigenschaft deaktiviert.</p>
Aktiviert	Legt die Einstellung für param1 fest.
name	<p>Eine beschreibende Kennzeichnung für die Komponente. Diese Kennzeichnung wird in der Protokolldatei sowie der Ereignisansicht angezeigt. Ermitteln Sie anhand der Eigenschaft name, welche Komponente das Ereignis verursacht hat.</p>
optional	<p>Legt fest, ob ein Komponentenfehler den Ausfall der übergeordneten Komponente verursacht. Sie können eine der folgenden Optionen auswählen:</p> <ul style="list-style-type: none"> - Ausgewählt. Ein Komponentenfehler führt nicht zum Ausfall der übergeordneten Komponente. - Gelöscht. Ein Komponentenfehler verursacht den Ausfall der übergeordneten Komponente. <p>Standardwert ist "Gelöscht". Weitere Informationen über Komponentenfehler finden Sie in "Fehlerbehandlung" auf Seite 410.</p>
param1	<p>Legt fest, ob die Eingabe optional ist. param1 wird als is_optional bezeichnet und hat nur eine Eigenschaft, enabled. enabled hat folgende Optionen:</p> <ul style="list-style-type: none"> - Ausgewählt. Die Eingabedaten sind optional. - Gelöscht. Die Eingabedaten sind verpflichtend.
param2	<p>Definiert einen EDI-Datentyp. param2 wird als input_type bezeichnet und hat nur eine Eigenschaft, value. value ist ein hartcodierter String oder ein Datenbehälter.</p>
param3	<p>Definiert einen Bereich von ganzen Zahlen. param3 wird als minmax_limits bezeichnet und hat nur eine Eigenschaft, value. value ist ein hartcodierter String oder ein Datenbehälter, der zwei, durch einen Bindestrich getrennte, ganze Zahlen festlegt.</p>

Eigenschaft	Beschreibung
param4	Definiert eine Liste von Werten. param4 wird als enumerations bezeichnet und hat nur eine Eigenschaft, value . value ist ein hartcodierter String oder Datenbehälter, der eine kommagetrennte Liste von Strings oder ganzen Zahlen angibt.
remark	Ein benutzerdefinierter Kommentar, der den Zweck oder die Aktion der Komponente beschreibt.
value	Definiert einen Wert für param1 , param2 oder param3

Hinweis: Diese Komponente ist veraltet. Der IntelliScript-Editor zeigt dies für veraltete Skripte an. Verwenden Sie sie nicht in neuen Skripten. Verwenden Sie stattdessen andere Validator-Komponenten.

Aufzählungen

Der Validator **Enumeration** testet, ob ein Wert ein Mitglied einer Wertgruppe ist.

Die nachstehende Tabelle beschreibt die Eigenschaften des Validators **Enumeration**:

Eigenschaft	Beschreibung
allow_empty_value	Legt fest, ob eine leere Eingabe als gültig akzeptiert wird. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none"> - Ausgewählt. Leere Eingabe ist gültig. - Gelöscht. Leere Eingabe ist ungültig. Die Standardoption lautet „Gelöscht“.
disabled	Legt fest, ob die Komponente und alle zugehörigen untergeordneten Komponenten vom Skript ignoriert werden. Verwenden Sie diese Eigenschaft zum Testen, Debuggen und Ändern eines Skripts. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none"> - Ausgewählt. Das Skript ignoriert die Komponente. - Gelöscht. Das Skript wendet die Komponente an. Standardmäßig ist die Eigenschaft deaktiviert.
enumerations	Definiert eine Liste von Werten.
ignore_case	Legt fest, ob der Vergleich die Groß-/Kleinschreibung beachtet. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none"> - Ausgewählt. Die Groß-/Kleinschreibung wird beim Vergleich nicht berücksichtigt. - Gelöscht. Die Groß-/Kleinschreibung wird beim Vergleich berücksichtigt. Standardwert ist "Gelöscht".
name	Eine beschreibende Kennzeichnung für die Komponente. Diese Kennzeichnung wird in der Protokolldatei sowie der Ereignisansicht angezeigt. Ermitteln Sie anhand der Eigenschaft name , welche Komponente das Ereignis verursacht hat.
negation	Legt fest, ob die Validierungsbedingung negiert wird. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none"> - Ausgewählt. Wenn die Bedingung "true" ist, ist die Eingabe nicht gültig. Wenn die Bedingung "false" ist, ist die Eingabe gültig. - Gelöscht. Wenn die Bedingung "true" ist, ist die Eingabe gültig. Wenn die Bedingung "false" ist, ist die Eingabe nicht gültig. Die Standardoption lautet „Gelöscht“.

Eigenschaft	Beschreibung
notify	Definiert den Namen einer Benachrichtigung. Wenn die Eingabe nicht mit der Validierungsbedingung übereinstimmt, löst der Validator eine Benachrichtigung aus. Weitere Informationen hierzu finden Sie unter "Benachrichtigungen" auf Seite 430 . Die Standardoption lautet „Gelöscht“.
optional	Legt fest, ob ein Komponentenfehler den Ausfall der übergeordneten Komponente verursacht. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none"> - Ausgewählt. Ein Komponentenfehler führt nicht zum Ausfall der übergeordneten Komponente. - Gelöscht. Ein Komponentenfehler verursacht den Ausfall der übergeordneten Komponente. Standardwert ist "Gelöscht". Weitere Informationen über Komponentenfehler finden Sie in "Fehlerbehandlung" auf Seite 410 .
remark	Ein benutzerdefinierter Kommentar, der den Zweck oder die Aktion der Komponente beschreibt.
transformers	Definiert eine Liste mit Transformatoren, die für die Eingabe gelten. Die Validierungsbedingung wird auf das Ergebnis der Transformer angewendet. Die Transformer haben lediglich zum Zweck der Validierung eine temporäre Auswirkung auf die Daten. Die Eingabe wird nicht permanent geändert.

LengthEquals

Der Validator **LengthEquals** testet, ob die Länge eines Strings gleich dem angegebenen Wert ist.

Die nachstehende Tabelle beschreibt die Eigenschaften des Validators **LengthEquals**:

Eigenschaft	Beschreibung
allow_empty_value	Legt fest, ob eine leere Eingabe als gültig akzeptiert wird. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none"> - Ausgewählt. Leere Eingabe ist gültig. - Gelöscht. Leere Eingabe ist ungültig. Die Standardoption lautet „Gelöscht“.
disabled	Legt fest, ob die Komponente und alle zugehörigen untergeordneten Komponenten vom Skript ignoriert werden. Verwenden Sie diese Eigenschaft zum Testen, Debuggen und Ändern eines Skripts. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none"> - Ausgewählt. Das Skript ignoriert die Komponente. - Gelöscht. Das Skript wendet die Komponente an. Standardmäßig ist die Eigenschaft deaktiviert.
length	Definiert die Länge des Strings.
name	Eine beschreibende Kennzeichnung für die Komponente. Diese Kennzeichnung wird in der Protokolldatei sowie der Ereignisansicht angezeigt. Ermitteln Sie anhand der Eigenschaft name , welche Komponente das Ereignis verursacht hat.
negation	Legt fest, ob die Validierungsbedingung negiert wird. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none"> - Ausgewählt. Wenn die Bedingung "true" ist, ist die Eingabe nicht gültig. Wenn die Bedingung "false" ist, ist die Eingabe gültig. - Gelöscht. Wenn die Bedingung "true" ist, ist die Eingabe gültig. Wenn die Bedingung "false" ist, ist die Eingabe nicht gültig. Die Standardoption lautet „Gelöscht“.

Eigenschaft	Beschreibung
notify	Definiert den Namen einer Benachrichtigung. Wenn die Eingabe nicht mit der Validierungsbedingung übereinstimmt, löst der Validator eine Benachrichtigung aus. Weitere Informationen hierzu finden Sie unter "Benachrichtigungen" auf Seite 430 . Die Standardoption lautet „Gelöscht“.
optional	Legt fest, ob ein Komponentenfehler den Ausfall der übergeordneten Komponente verursacht. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none"> - Ausgewählt. Ein Komponentenfehler führt nicht zum Ausfall der übergeordneten Komponente. - Gelöscht. Ein Komponentenfehler verursacht den Ausfall der übergeordneten Komponente. Standardwert ist "Gelöscht". Weitere Informationen über Komponentenfehler finden Sie in "Fehlerbehandlung" auf Seite 410 .
remark	Ein benutzerdefinierter Kommentar, der den Zweck oder die Aktion der Komponente beschreibt.
transformers	Definiert eine Liste mit Transformatoren, die für die Eingabe gelten. Die Validierungsbedingung wird auf das Ergebnis der Transformer angewendet. Die Transformer haben lediglich zum Zweck der Validierung eine temporäre Auswirkung auf die Daten. Die Eingabe wird nicht permanent geändert.

MaxLength

Der Validator **MaxLength** testet, ob die Länge eines Strings maximal gleich dem angegebenen Wert ist.

Die nachstehende Tabelle beschreibt die Eigenschaften des Validators **MaxLength**:

Eigenschaft	Beschreibung
allow_empty_value	Legt fest, ob eine leere Eingabe als gültig akzeptiert wird. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none"> - Ausgewählt. Leere Eingabe ist gültig. - Gelöscht. Leere Eingabe ist ungültig. Die Standardoption lautet „Gelöscht“.
disabled	Legt fest, ob die Komponente und alle zugehörigen untergeordneten Komponenten vom Skript ignoriert werden. Verwenden Sie diese Eigenschaft zum Testen, Debuggen und Ändern eines Skripts. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none"> - Ausgewählt. Das Skript ignoriert die Komponente. - Gelöscht. Das Skript wendet die Komponente an. Standardmäßig ist die Eigenschaft deaktiviert.
length	Definiert die maximale Länge des Strings.
name	Eine beschreibende Kennzeichnung für die Komponente. Diese Kennzeichnung wird in der Protokolldatei sowie der Ereignisansicht angezeigt. Ermitteln Sie anhand der Eigenschaft name , welche Komponente das Ereignis verursacht hat.
negation	Legt fest, ob die Validierungsbedingung negiert wird. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none"> - Ausgewählt. Wenn die Bedingung "true" ist, ist die Eingabe nicht gültig. Wenn die Bedingung "false" ist, ist die Eingabe gültig. - Gelöscht. Wenn die Bedingung "true" ist, ist die Eingabe gültig. Wenn die Bedingung "false" ist, ist die Eingabe nicht gültig. Die Standardoption lautet „Gelöscht“.

Eigenschaft	Beschreibung
notify	Definiert den Namen einer Benachrichtigung. Wenn die Eingabe nicht mit der Validierungsbedingung übereinstimmt, löst der Validator eine Benachrichtigung aus. Weitere Informationen hierzu finden Sie unter "Benachrichtigungen" auf Seite 430 . Die Standardoption lautet „Gelöscht“.
optional	Legt fest, ob ein Komponentenfehler den Ausfall der übergeordneten Komponente verursacht. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none"> - Ausgewählt. Ein Komponentenfehler führt nicht zum Ausfall der übergeordneten Komponente. - Gelöscht. Ein Komponentenfehler verursacht den Ausfall der übergeordneten Komponente. Standardwert ist "Gelöscht". Weitere Informationen über Komponentenfehler finden Sie in "Fehlerbehandlung" auf Seite 410 .
remark	Ein benutzerdefinierter Kommentar, der den Zweck oder die Aktion der Komponente beschreibt.
transformers	Definiert eine Liste mit Transformatoren, die für die Eingabe gelten. Die Validierungsbedingung wird auf das Ergebnis der Transformer angewendet. Die Transformer haben lediglich zum Zweck der Validierung eine temporäre Auswirkung auf die Daten. Die Eingabe wird nicht permanent geändert.

MaxNumber

Der Validator **MaxNumber** testet, ob eine Zahl maximal gleich dem angegebenen Wert ist.

Die nachstehende Tabelle beschreibt die Eigenschaften des Validators **MaxNumber**:

Eigenschaft	Beschreibung
allow_empty_value	Legt fest, ob eine leere Eingabe als gültig akzeptiert wird. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none"> - Ausgewählt. Leere Eingabe ist gültig. - Gelöscht. Leere Eingabe ist ungültig. Die Standardoption lautet „Gelöscht“.
disabled	Legt fest, ob die Komponente und alle zugehörigen untergeordneten Komponenten vom Skript ignoriert werden. Verwenden Sie diese Eigenschaft zum Testen, Debuggen und Ändern eines Skripts. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none"> - Ausgewählt. Das Skript ignoriert die Komponente. - Gelöscht. Das Skript wendet die Komponente an. Standardmäßig ist die Eigenschaft deaktiviert.
name	Eine beschreibende Kennzeichnung für die Komponente. Diese Kennzeichnung wird in der Protokolldatei sowie der Ereignisansicht angezeigt. Ermitteln Sie anhand der Eigenschaft name , welche Komponente das Ereignis verursacht hat.
negation	Legt fest, ob die Validierungsbedingung negiert wird. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none"> - Ausgewählt. Wenn die Bedingung "true" ist, ist die Eingabe nicht gültig. Wenn die Bedingung "false" ist, ist die Eingabe gültig. - Gelöscht. Wenn die Bedingung "true" ist, ist die Eingabe gültig. Wenn die Bedingung "false" ist, ist die Eingabe nicht gültig. Die Standardoption lautet „Gelöscht“.

Eigenschaft	Beschreibung
notify	Definiert den Namen einer Benachrichtigung. Wenn die Eingabe nicht mit der Validierungsbedingung übereinstimmt, löst der Validator eine Benachrichtigung aus. Weitere Informationen hierzu finden Sie unter "Benachrichtigungen" auf Seite 430 . Die Standardoption lautet „Gelöscht“.
optional	Legt fest, ob ein Komponentenfehler den Ausfall der übergeordneten Komponente verursacht. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none"> - Ausgewählt. Ein Komponentenfehler führt nicht zum Ausfall der übergeordneten Komponente. - Gelöscht. Ein Komponentenfehler verursacht den Ausfall der übergeordneten Komponente. Standardwert ist "Gelöscht". Weitere Informationen über Komponentenfehler finden Sie in "Fehlerbehandlung" auf Seite 410 .
remark	Ein benutzerdefinierter Kommentar, der den Zweck oder die Aktion der Komponente beschreibt.
transformers	Definiert eine Liste mit Transformatoren, die für die Eingabe gelten. Die Validierungsbedingung wird auf das Ergebnis der Transformer angewendet. Die Transformer haben lediglich zum Zweck der Validierung eine temporäre Auswirkung auf die Daten. Die Eingabe wird nicht permanent geändert.
value	Definiert den maximalen Wert der Zahl.

MinLength

Der Validator **MinLength** testet, ob die Länge eines Strings mindestens gleich dem angegebenen Wert ist.

Die nachstehende Tabelle beschreibt die Eigenschaften des Validators **MinLength**:

Eigenschaft	Beschreibung
allow_empty_value	Legt fest, ob eine leere Eingabe als gültig akzeptiert wird. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none"> - Ausgewählt. Leere Eingabe ist gültig. - Gelöscht. Leere Eingabe ist ungültig. Die Standardoption lautet „Gelöscht“.
disabled	Legt fest, ob die Komponente und alle zugehörigen untergeordneten Komponenten vom Skript ignoriert werden. Verwenden Sie diese Eigenschaft zum Testen, Debuggen und Ändern eines Skripts. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none"> - Ausgewählt. Das Skript ignoriert die Komponente. - Gelöscht. Das Skript wendet die Komponente an. Standardmäßig ist die Eigenschaft deaktiviert.
length	Definiert die minimale Länge des Strings.
name	Eine beschreibende Kennzeichnung für die Komponente. Diese Kennzeichnung wird in der Protokolldatei sowie der Ereignisansicht angezeigt. Ermitteln Sie anhand der Eigenschaft name , welche Komponente das Ereignis verursacht hat.

Eigenschaft	Beschreibung
negation	<p>Legt fest, ob die Validierungsbedingung negiert wird. Sie können eine der folgenden Optionen auswählen:</p> <ul style="list-style-type: none"> - Ausgewählt. Wenn die Bedingung "true" ist, ist die Eingabe nicht gültig. Wenn die Bedingung "false" ist, ist die Eingabe gültig. - Gelöscht. Wenn die Bedingung "true" ist, ist die Eingabe gültig. Wenn die Bedingung "false" ist, ist die Eingabe nicht gültig. <p>Die Standardoption lautet „Gelöscht“.</p>
notify	<p>Definiert den Namen einer Benachrichtigung. Wenn die Eingabe nicht mit der Validierungsbedingung übereinstimmt, löst der Validator eine Benachrichtigung aus. Weitere Informationen hierzu finden Sie unter "Benachrichtigungen" auf Seite 430. Die Standardoption lautet „Gelöscht“.</p>
optional	<p>Legt fest, ob ein Komponentenfehler den Ausfall der übergeordneten Komponente verursacht. Sie können eine der folgenden Optionen auswählen:</p> <ul style="list-style-type: none"> - Ausgewählt. Ein Komponentenfehler führt nicht zum Ausfall der übergeordneten Komponente. - Gelöscht. Ein Komponentenfehler verursacht den Ausfall der übergeordneten Komponente. <p>Standardwert ist "Gelöscht". Weitere Informationen über Komponentenfehler finden Sie in "Fehlerbehandlung" auf Seite 410.</p>
remark	<p>Ein benutzerdefinierter Kommentar, der den Zweck oder die Aktion der Komponente beschreibt.</p>
transformers	<p>Definiert eine Liste mit Transformern, die für die Eingabe gelten. Die Validierungsbedingung wird auf das Ergebnis der Transformer angewendet. Die Transformer haben lediglich zum Zweck der Validierung eine temporäre Auswirkung auf die Daten. Die Eingabe wird nicht permanent geändert.</p>

MinNumber

Der Validator **MinNumber** testet, ob eine Zahl mindestens gleich dem angegebenen Wert ist.

Die nachstehende Tabelle beschreibt die Eigenschaften des Validators **MinNumber**:

Eigenschaft	Beschreibung
allow_empty_value	<p>Legt fest, ob eine leere Eingabe als gültig akzeptiert wird. Sie können eine der folgenden Optionen auswählen:</p> <ul style="list-style-type: none"> - Ausgewählt. Leere Eingabe ist gültig. - Gelöscht. Leere Eingabe ist ungültig. <p>Die Standardoption lautet „Gelöscht“.</p>
disabled	<p>Legt fest, ob die Komponente und alle zugehörigen untergeordneten Komponenten vom Skript ignoriert werden. Verwenden Sie diese Eigenschaft zum Testen, Debuggen und Ändern eines Skripts. Sie können eine der folgenden Optionen auswählen:</p> <ul style="list-style-type: none"> - Ausgewählt. Das Skript ignoriert die Komponente. - Gelöscht. Das Skript wendet die Komponente an. <p>Standardmäßig ist die Eigenschaft deaktiviert.</p>
name	<p>Eine beschreibende Kennzeichnung für die Komponente. Diese Kennzeichnung wird in der Protokolldatei sowie der Ereignisansicht angezeigt. Ermitteln Sie anhand der Eigenschaft name, welche Komponente das Ereignis verursacht hat.</p>

Eigenschaft	Beschreibung
negation	<p>Legt fest, ob die Validierungsbedingung negiert wird. Sie können eine der folgenden Optionen auswählen:</p> <ul style="list-style-type: none"> - Ausgewählt. Wenn die Bedingung "true" ist, ist die Eingabe nicht gültig. Wenn die Bedingung "false" ist, ist die Eingabe gültig. - Gelöscht. Wenn die Bedingung "true" ist, ist die Eingabe gültig. Wenn die Bedingung "false" ist, ist die Eingabe nicht gültig. <p>Die Standardoption lautet „Gelöscht“.</p>
notify	<p>Definiert den Namen einer Benachrichtigung. Wenn die Eingabe nicht mit der Validierungsbedingung übereinstimmt, löst der Validator eine Benachrichtigung aus. Weitere Informationen hierzu finden Sie unter "Benachrichtigungen" auf Seite 430. Die Standardoption lautet „Gelöscht“.</p>
optional	<p>Legt fest, ob ein Komponentenfehler den Ausfall der übergeordneten Komponente verursacht. Sie können eine der folgenden Optionen auswählen:</p> <ul style="list-style-type: none"> - Ausgewählt. Ein Komponentenfehler führt nicht zum Ausfall der übergeordneten Komponente. - Gelöscht. Ein Komponentenfehler verursacht den Ausfall der übergeordneten Komponente. <p>Standardwert ist "Gelöscht". Weitere Informationen über Komponentenfehler finden Sie in "Fehlerbehandlung" auf Seite 410.</p>
remark	<p>Ein benutzerdefinierter Kommentar, der den Zweck oder die Aktion der Komponente beschreibt.</p>
transformers	<p>Definiert eine Liste mit Transformern, die für die Eingabe gelten. Die Validierungsbedingung wird auf das Ergebnis der Transformer angewendet. Die Transformer haben lediglich zum Zweck der Validierung eine temporäre Auswirkung auf die Daten. Die Eingabe wird nicht permanent geändert.</p>
value	<p>Definiert den minimalen Wert der Zahl.</p>

NumberEquals

Der Validator **NumberEquals** testet, ob eine Zahl gleich einem angegebenen Wert ist.

Die nachstehende Tabelle beschreibt die Eigenschaften des Validators **NumberEquals**:

Eigenschaft	Beschreibung
allow_empty_value	<p>Legt fest, ob eine leere Eingabe als gültig akzeptiert wird. Sie können eine der folgenden Optionen auswählen:</p> <ul style="list-style-type: none"> - Ausgewählt. Leere Eingabe ist gültig. - Gelöscht. Leere Eingabe ist ungültig. <p>Die Standardoption lautet „Gelöscht“.</p>
disabled	<p>Legt fest, ob die Komponente und alle zugehörigen untergeordneten Komponenten vom Skript ignoriert werden. Verwenden Sie diese Eigenschaft zum Testen, Debuggen und Ändern eines Skripts. Sie können eine der folgenden Optionen auswählen:</p> <ul style="list-style-type: none"> - Ausgewählt. Das Skript ignoriert die Komponente. - Gelöscht. Das Skript wendet die Komponente an. <p>Standardmäßig ist die Eigenschaft deaktiviert.</p>
name	<p>Eine beschreibende Kennzeichnung für die Komponente. Diese Kennzeichnung wird in der Protokolldatei sowie der Ereignisansicht angezeigt. Ermitteln Sie anhand der Eigenschaft name, welche Komponente das Ereignis verursacht hat.</p>

Eigenschaft	Beschreibung
negation	<p>Legt fest, ob die Validierungsbedingung negiert wird. Sie können eine der folgenden Optionen auswählen:</p> <ul style="list-style-type: none"> - Ausgewählt. Wenn die Bedingung "true" ist, ist die Eingabe nicht gültig. Wenn die Bedingung "false" ist, ist die Eingabe gültig. - Gelöscht. Wenn die Bedingung "true" ist, ist die Eingabe gültig. Wenn die Bedingung "false" ist, ist die Eingabe nicht gültig. <p>Die Standardoption lautet „Gelöscht“.</p>
notify	<p>Definiert den Namen einer Benachrichtigung. Wenn die Eingabe nicht mit der Validierungsbedingung übereinstimmt, löst der Validator eine Benachrichtigung aus. Weitere Informationen hierzu finden Sie unter "Benachrichtigungen" auf Seite 430. Die Standardoption lautet „Gelöscht“.</p>
optional	<p>Legt fest, ob ein Komponentenfehler den Ausfall der übergeordneten Komponente verursacht. Sie können eine der folgenden Optionen auswählen:</p> <ul style="list-style-type: none"> - Ausgewählt. Ein Komponentenfehler führt nicht zum Ausfall der übergeordneten Komponente. - Gelöscht. Ein Komponentenfehler verursacht den Ausfall der übergeordneten Komponente. <p>Standardwert ist "Gelöscht". Weitere Informationen über Komponentenfehler finden Sie in "Fehlerbehandlung" auf Seite 410.</p>
remark	<p>Ein benutzerdefinierter Kommentar, der den Zweck oder die Aktion der Komponente beschreibt.</p>
transformers	<p>Definiert eine Liste mit Transformern, die für die Eingabe gelten. Die Validierungsbedingung wird auf das Ergebnis der Transformer angewendet. Die Transformer haben lediglich zum Zweck der Validierung eine temporäre Auswirkung auf die Daten. Die Eingabe wird nicht permanent geändert.</p>
value	<p>Definiert den Wert der Zahl.</p>

ValidateByExpression

Der Validator **ValidateByExpression** wertet einen JavaScript-Ausdruck aus. Wenn der Ausdruck falsch ist, betrachtet der Validator die Eingabe als ungültig.

In der folgenden Tabelle werden die Eigenschaften des Validators **ValidateByExpression** beschrieben:

Eigenschaft	Beschreibung
allow_empty_value	<p>Legt fest, ob eine leere Eingabe als gültig akzeptiert wird. Sie können eine der folgenden Optionen auswählen:</p> <ul style="list-style-type: none"> - Ausgewählt. Leere Eingabe ist gültig. - Gelöscht. Leere Eingabe ist ungültig. <p>Die Standardoption lautet „Gelöscht“.</p>
disabled	<p>Legt fest, ob die Komponente und alle zugehörigen untergeordneten Komponenten vom Skript ignoriert werden. Verwenden Sie diese Eigenschaft zum Testen, Debuggen und Ändern eines Skripts. Sie können eine der folgenden Optionen auswählen:</p> <ul style="list-style-type: none"> - Ausgewählt. Das Skript ignoriert die Komponente. - Gelöscht. Das Skript wendet die Komponente an. <p>Standardmäßig ist die Eigenschaft deaktiviert.</p>

Eigenschaft	Beschreibung
expression	<p>Definiert einen JavaScript-Ausdruck. Verwenden Sie <code>\$0</code> für die Validatoreingabe. Verwenden Sie ein Dollarzeichen (\$) zusammen mit einer Ganzzahl für weitere Datenbehälter, die unter params definiert sind. Beginnen Sie dabei mit <code>\$1</code>. Beispiel: Der folgende Ausdruck überprüft, ob die Eingabe den Wert <code>Ron Lehrer</code> aufweist:</p> <pre>\$0 == "Ron Lehrer"</pre>
name	Eine beschreibende Kennzeichnung für die Komponente. Diese Kennzeichnung wird in der Protokolldatei sowie der Ereignisansicht angezeigt. Ermitteln Sie anhand der Eigenschaft name , welche Komponente das Ereignis verursacht hat.
negation	<p>Legt fest, ob die Validierungsbedingung negiert wird. Sie können eine der folgenden Optionen auswählen:</p> <ul style="list-style-type: none"> - Ausgewählt. Wenn die Bedingung "true" ist, ist die Eingabe nicht gültig. Wenn die Bedingung "false" ist, ist die Eingabe gültig. - Gelöscht. Wenn die Bedingung "true" ist, ist die Eingabe gültig. Wenn die Bedingung "false" ist, ist die Eingabe nicht gültig. <p>Die Standardoption lautet „Gelöscht“.</p>
notify	Definiert den Namen einer Benachrichtigung. Wenn die Eingabe nicht mit der Validierungsbedingung übereinstimmt, löst der Validator eine Benachrichtigung aus. Weitere Informationen hierzu finden Sie unter "Benachrichtigungen" auf Seite 430 . Die Standardoption lautet „Gelöscht“.
optional	<p>Legt fest, ob ein Komponentenfehler den Ausfall der übergeordneten Komponente verursacht. Sie können eine der folgenden Optionen auswählen:</p> <ul style="list-style-type: none"> - Ausgewählt. Ein Komponentenfehler führt nicht zum Ausfall der übergeordneten Komponente. - Gelöscht. Ein Komponentenfehler verursacht den Ausfall der übergeordneten Komponente. <p>Standardwert ist "Gelöscht". Weitere Informationen über Komponentenfehler finden Sie in "Fehlerbehandlung" auf Seite 410.</p>
params	Definiert eine Liste mit Datenbehältern, die die im Ausdruck zu verwendenden Parameter enthalten.
remark	Ein benutzerdefinierter Kommentar, der den Zweck oder die Aktion der Komponente beschreibt.
Transformer	Definiert eine Liste mit Transformern, die für die Eingabe gelten. Die Validierungsbedingung wird auf das Ergebnis der Transformer angewendet. Die Transformer haben lediglich zum Zweck der Validierung eine temporäre Auswirkung auf die Daten. Die Eingabe wird nicht permanent geändert.

ValidateByPattern

Der Validator **ValidateByPattern** testet, ob ein String mit einem regulären Ausdruck übereinstimmt. Weitere Informationen hierzu finden Sie unter ["RegularExpression" auf Seite 286](#).

In der folgenden Tabelle werden die Eigenschaften des Validators **ValidateByPattern** beschrieben:

Eigenschaft	Beschreibung
allow_empty_value	Legt fest, ob eine leere Eingabe als gültig akzeptiert wird. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none"> - Ausgewählt. Leere Eingabe ist gültig. - Gelöscht. Leere Eingabe ist ungültig. Die Standardoption lautet „Gelöscht“.
disabled	Legt fest, ob die Komponente und alle zugehörigen untergeordneten Komponenten vom Skript ignoriert werden. Verwenden Sie diese Eigenschaft zum Testen, Debuggen und Ändern eines Skripts. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none"> - Ausgewählt. Das Skript ignoriert die Komponente. - Gelöscht. Das Skript wendet die Komponente an. Standardmäßig ist die Eigenschaft deaktiviert.
expression	Definiert einen regulären Ausdruck.
name	Eine beschreibende Kennzeichnung für die Komponente. Diese Kennzeichnung wird in der Protokolldatei sowie der Ereignisansicht angezeigt. Ermitteln Sie anhand der Eigenschaft name , welche Komponente das Ereignis verursacht hat.
negation	Legt fest, ob die Validierungsbedingung negiert wird. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none"> - Ausgewählt. Wenn die Bedingung "true" ist, ist die Eingabe nicht gültig. Wenn die Bedingung "false" ist, ist die Eingabe gültig. - Gelöscht. Wenn die Bedingung "true" ist, ist die Eingabe gültig. Wenn die Bedingung "false" ist, ist die Eingabe nicht gültig. Die Standardoption lautet „Gelöscht“.
notify	Definiert den Namen einer Benachrichtigung. Wenn die Eingabe nicht mit der Validierungsbedingung übereinstimmt, löst der Validator eine Benachrichtigung aus. Weitere Informationen hierzu finden Sie unter "Benachrichtigungen" auf Seite 430 . Die Standardoption lautet „Gelöscht“.
optional	Legt fest, ob ein Komponentenfehler den Ausfall der übergeordneten Komponente verursacht. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none"> - Ausgewählt. Ein Komponentenfehler führt nicht zum Ausfall der übergeordneten Komponente. - Gelöscht. Ein Komponentenfehler verursacht den Ausfall der übergeordneten Komponente. Standardwert ist "Gelöscht". Weitere Informationen über Komponentenfehler finden Sie in "Fehlerbehandlung" auf Seite 410 .
remark	Ein benutzerdefinierter Kommentar, der den Zweck oder die Aktion der Komponente beschreibt.
Transformer	Definiert eine Liste mit Transformatoren, die für die Eingabe gelten. Die Validierungsbedingung wird auf das Ergebnis der Transformer angewendet. Die Transformer haben lediglich zum Zweck der Validierung eine temporäre Auswirkung auf die Daten. Die Eingabe wird nicht permanent geändert.

ValidateByTransformer

Der Validator **ValidateByTransformer** wendet eine Liste mit einem oder mehreren Transformatoren auf die Eingabe an. Wenn die Liste mit Transformatoren fehlschlägt, betrachtet der Validator die Eingabe als ungültig.

In der folgenden Tabelle werden die Eigenschaften des Validators **ValidateByTransformer** beschrieben:

Eigenschaft	Beschreibung
allow_empty_value	Legt fest, ob eine leere Eingabe als gültig akzeptiert wird. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none"> - Ausgewählt. Leere Eingabe ist gültig. - Gelöscht. Leere Eingabe ist ungültig. Die Standardoption lautet „Gelöscht“.
disabled	Legt fest, ob die Komponente und alle zugehörigen untergeordneten Komponenten vom Skript ignoriert werden. Verwenden Sie diese Eigenschaft zum Testen, Debuggen und Ändern eines Skripts. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none"> - Ausgewählt. Das Skript ignoriert die Komponente. - Gelöscht. Das Skript wendet die Komponente an. Standardmäßig ist die Eigenschaft deaktiviert.
name	Eine beschreibende Kennzeichnung für die Komponente. Diese Kennzeichnung wird in der Protokolldatei sowie der Ereignisansicht angezeigt. Ermitteln Sie anhand der Eigenschaft name , welche Komponente das Ereignis verursacht hat.
negation	Legt fest, ob die Validierungsbedingung negiert wird. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none"> - Ausgewählt. Wenn die Bedingung "true" ist, ist die Eingabe nicht gültig. Wenn die Bedingung "false" ist, ist die Eingabe gültig. - Gelöscht. Wenn die Bedingung "true" ist, ist die Eingabe gültig. Wenn die Bedingung "false" ist, ist die Eingabe nicht gültig. Die Standardoption lautet „Gelöscht“.
notify	Definiert den Namen einer Benachrichtigung. Wenn die Eingabe nicht mit der Validierungsbedingung übereinstimmt, löst der Validator eine Benachrichtigung aus. Weitere Informationen hierzu finden Sie unter "Benachrichtigungen" auf Seite 430 . Die Standardoption lautet „Gelöscht“.
optional	Legt fest, ob ein Komponentenfehler den Ausfall der übergeordneten Komponente verursacht. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none"> - Ausgewählt. Ein Komponentenfehler führt nicht zum Ausfall der übergeordneten Komponente. - Gelöscht. Ein Komponentenfehler verursacht den Ausfall der übergeordneten Komponente. Standardwert ist "Gelöscht". Weitere Informationen über Komponentenfehler finden Sie in "Fehlerbehandlung" auf Seite 410 .
remark	Ein benutzerdefinierter Kommentar, der den Zweck oder die Aktion der Komponente beschreibt.
run_transformers	Definiert eine Liste mit Transformatoren.
Transformer	Definiert eine Liste mit Transformatoren, die für die Eingabe gelten. Die Validierungsbedingung wird auf das Ergebnis der Transformer angewendet. Die Transformer haben lediglich zum Zweck der Validierung eine temporäre Auswirkung auf die Daten. Die Eingabe wird nicht permanent geändert.

ValidateByType

Der Validator **ValidateByType** testet, ob seine Eingabe mit einem angegebenen Datentyp übereinstimmt.

In der folgenden Tabelle werden die Eigenschaften des Validators **ValidateByType** beschrieben:

Eigenschaft	Beschreibung
allow_empty_value	Legt fest, ob eine leere Eingabe als gültig akzeptiert wird. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none"> - Ausgewählt. Leere Eingabe ist gültig. - Gelöscht. Leere Eingabe ist ungültig. Die Standardoption lautet „Gelöscht“.
disabled	Legt fest, ob die Komponente und alle zugehörigen untergeordneten Komponenten vom Skript ignoriert werden. Verwenden Sie diese Eigenschaft zum Testen, Debuggen und Ändern eines Skripts. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none"> - Ausgewählt. Das Skript ignoriert die Komponente. - Gelöscht. Das Skript wendet die Komponente an. Standardmäßig ist die Eigenschaft deaktiviert.
name	Eine beschreibende Kennzeichnung für die Komponente. Diese Kennzeichnung wird in der Protokolldatei sowie der Ereignisansicht angezeigt. Ermitteln Sie anhand der Eigenschaft name , welche Komponente das Ereignis verursacht hat.
negation	Legt fest, ob die Validierungsbedingung negiert wird. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none"> - Ausgewählt. Wenn die Bedingung "true" ist, ist die Eingabe nicht gültig. Wenn die Bedingung "false" ist, ist die Eingabe gültig. - Gelöscht. Wenn die Bedingung "true" ist, ist die Eingabe gültig. Wenn die Bedingung "false" ist, ist die Eingabe nicht gültig. Die Standardoption lautet „Gelöscht“.
notify	Definiert den Namen einer Benachrichtigung. Wenn die Eingabe nicht mit der Validierungsbedingung übereinstimmt, löst der Validator eine Benachrichtigung aus. Weitere Informationen hierzu finden Sie unter "Benachrichtigungen" auf Seite 430 . Die Standardoption lautet „Gelöscht“.
optional	Legt fest, ob ein Komponentenfehler den Ausfall der übergeordneten Komponente verursacht. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none"> - Ausgewählt. Ein Komponentenfehler führt nicht zum Ausfall der übergeordneten Komponente. - Gelöscht. Ein Komponentenfehler verursacht den Ausfall der übergeordneten Komponente. Standardwert ist "Gelöscht". Weitere Informationen über Komponentenfehler finden Sie in "Fehlerbehandlung" auf Seite 410 .
remark	Ein benutzerdefinierter Kommentar, der den Zweck oder die Aktion der Komponente beschreibt.
Transformer	Definiert eine Liste mit Transformatoren, die für die Eingabe gelten. Die Validierungsbedingung wird auf das Ergebnis der Transformer angewendet. Die Transformer haben lediglich zum Zweck der Validierung eine temporäre Auswirkung auf die Daten. Die Eingabe wird nicht permanent geändert.
val_type	Definiert einen Datentyp. Wählen Sie einen Standardtyp oder einen Typ aus, der in den Projektschemata definiert ist.

ValidateDate

Der Validator **ValidateDate** testet, ob ein Datum mit einem angegebenen ICU-Datumsformat übereinstimmt, beispielsweise `yyyy-MM-dd`. Weitere Informationen hierzu finden Sie unter ["DateFormatICU" auf Seite 270](#).

In der folgenden Tabelle werden die Eigenschaften des Validators **ValidateDate** beschrieben:

Eigenschaft	Beschreibung
allow_empty_value	Legt fest, ob eine leere Eingabe als gültig akzeptiert wird. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none"> - Ausgewählt. Leere Eingabe ist gültig. - Gelöscht. Leere Eingabe ist ungültig. Die Standardoption lautet „Gelöscht“.
disabled	Legt fest, ob die Komponente und alle zugehörigen untergeordneten Komponenten vom Skript ignoriert werden. Verwenden Sie diese Eigenschaft zum Testen, Debuggen und Ändern eines Skripts. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none"> - Ausgewählt. Das Skript ignoriert die Komponente. - Gelöscht. Das Skript wendet die Komponente an. Standardmäßig ist die Eigenschaft deaktiviert.
format_string	Definiert ein ICU-Datumsformat.
name	Eine beschreibende Kennzeichnung für die Komponente. Diese Kennzeichnung wird in der Protokolldatei sowie der Ereignisansicht angezeigt. Ermitteln Sie anhand der Eigenschaft name , welche Komponente das Ereignis verursacht hat.
negation	Legt fest, ob die Validierungsbedingung negiert wird. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none"> - Ausgewählt. Wenn die Bedingung "true" ist, ist die Eingabe nicht gültig. Wenn die Bedingung "false" ist, ist die Eingabe gültig. - Gelöscht. Wenn die Bedingung "true" ist, ist die Eingabe gültig. Wenn die Bedingung "false" ist, ist die Eingabe nicht gültig. Die Standardoption lautet „Gelöscht“.
notify	Definiert den Namen einer Benachrichtigung. Wenn die Eingabe nicht mit der Validierungsbedingung übereinstimmt, löst der Validator eine Benachrichtigung aus. Weitere Informationen hierzu finden Sie unter "Benachrichtigungen" auf Seite 430 . Die Standardoption lautet „Gelöscht“.
optional	Legt fest, ob ein Komponentenfehler den Ausfall der übergeordneten Komponente verursacht. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none"> - Ausgewählt. Ein Komponentenfehler führt nicht zum Ausfall der übergeordneten Komponente. - Gelöscht. Ein Komponentenfehler verursacht den Ausfall der übergeordneten Komponente. Standardwert ist "Gelöscht". Weitere Informationen über Komponentenfehler finden Sie in "Fehlerbehandlung" auf Seite 410 .
remark	Ein benutzerdefinierter Kommentar, der den Zweck oder die Aktion der Komponente beschreibt.
Transformer	Definiert eine Liste mit Transformern, die für die Eingabe gelten. Die Validierungsbedingung wird auf das Ergebnis der Transformer angewendet. Die Transformer haben lediglich zum Zweck der Validierung eine temporäre Auswirkung auf die Daten. Die Eingabe wird nicht permanent geändert.

ValidatorPipeline

Der Validator **ValidatorPipeline** wendet eine Liste mit Validatoren auf die Daten an. Wenn einer der Validatoren eine Ungültigkeit dokumentiert oder wenn ein Validator als **optional** gekennzeichnet ist und fehlschlägt, löst die **ValidatorPipeline** eine Benachrichtigung aus.

Wenden Sie mithilfe einer **ValidatorPipeline** die AND-Logik auf eine Reihe von Validierungsbedingungen an. Die Daten sind gültig, wenn sie mit allen Bedingungen übereinstimmen.

In der folgenden Tabelle werden die Eigenschaften des Validators **ValidatorPipeline** beschrieben:

Eigenschaft	Beschreibung
allow_empty_value	Legt fest, ob eine leere Eingabe als gültig akzeptiert wird. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none"> - Ausgewählt. Leere Eingabe ist gültig. - Gelöscht. Leere Eingabe ist ungültig. Die Standardoption lautet „Gelöscht“.
disabled	Legt fest, ob die Komponente und alle zugehörigen untergeordneten Komponenten vom Skript ignoriert werden. Verwenden Sie diese Eigenschaft zum Testen, Debuggen und Ändern eines Skripts. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none"> - Ausgewählt. Das Skript ignoriert die Komponente. - Gelöscht. Das Skript wendet die Komponente an. Standardmäßig ist die Eigenschaft deaktiviert.
name	Eine beschreibende Kennzeichnung für die Komponente. Diese Kennzeichnung wird in der Protokolldatei sowie der Ereignisansicht angezeigt. Ermitteln Sie anhand der Eigenschaft name , welche Komponente das Ereignis verursacht hat.
negation	Legt fest, ob die Validierungsbedingung negiert wird. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none"> - Ausgewählt. Wenn die Bedingung "true" ist, ist die Eingabe nicht gültig. Wenn die Bedingung "false" ist, ist die Eingabe gültig. - Gelöscht. Wenn die Bedingung "true" ist, ist die Eingabe gültig. Wenn die Bedingung "false" ist, ist die Eingabe nicht gültig. Die Standardoption lautet „Gelöscht“.
notify	Definiert den Namen einer Benachrichtigung. Wenn die Eingabe nicht mit der Validierungsbedingung übereinstimmt, löst der Validator eine Benachrichtigung aus. Weitere Informationen hierzu finden Sie unter "Benachrichtigungen" auf Seite 430 . Die Standardoption lautet „Gelöscht“.
optional	Legt fest, ob ein Komponentenfehler den Ausfall der übergeordneten Komponente verursacht. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none"> - Ausgewählt. Ein Komponentenfehler führt nicht zum Ausfall der übergeordneten Komponente. - Gelöscht. Ein Komponentenfehler verursacht den Ausfall der übergeordneten Komponente. Standardwert ist "Gelöscht". Weitere Informationen über Komponentenfehler finden Sie in "Fehlerbehandlung" auf Seite 410 .
remark	Ein benutzerdefinierter Kommentar, der den Zweck oder die Aktion der Komponente beschreibt.
Transformer	Definiert eine Liste mit Transformern, die für die Eingabe gelten. Die Validierungsbedingung wird auf das Ergebnis der Transformer angewendet. Die Transformer haben lediglich zum Zweck der Validierung eine temporäre Auswirkung auf die Daten. Die Eingabe wird nicht permanent geändert.

Benachrichtigungen

Eine Benachrichtigung ist ein Signal, dass eine Bedingung in der Umwandlung eingetreten ist. Wenn die Bedingung eintritt, löst eine Umwandlung eine Benachrichtigung aus. Sie können Handler konfigurieren, die die Benachrichtigungen verarbeiten.

Die folgenden Beispiele zeigen einige Methoden für den Einsatz von Benachrichtigungen:

- Ein Validator kann eine Benachrichtigung auslösen. Eine **NotificationHandler**-Komponente kann eine Validierungswarnungsnachricht in die Ergebnisdatei der Umwandlung oder in ein Protokoll schreiben.
- Ein **StructureDefinition**-Anker kann eine Gruppe von **NotificationHandler**-Komponenten definieren, um Nichtübereinstimmungen zwischen den Eingabedatensätzen und der erforderlichen Eingabestruktur zu verarbeiten. Wenn eine Nichtübereinstimmung auftritt, schreibt der entsprechende **NotificationHandler** eine Nachricht in die Ergebnisdatei oder ein Protokoll.
- Eine **Notify**-Aktion zum Auslösen einer Benachrichtigung an jeder Stelle einer Umwandlung. Ein **NotificationHandler** kann eine Nachricht in die Ergebnisdatei oder in ein Protokoll schreiben.

In der folgenden Tabelle werden die Benachrichtigungstypen beschrieben:

Benachrichtigung	Beschreibung
MandatoryStructureMissing	Ein verpflichtender Datensatz erscheint nicht in der Eingabe.
MismatchIDs	Die ID des Datensatzes und des Unterelements stimmen teilweise überein. Beispiel: Es gibt zwei Datensatz-Bezeichner, und nur einer davon entspricht.
StructureBelowMinOccurs	Es gibt weniger passende übereinstimmende Datensätze des Unterelements als in minOccurs definiert sind.
StructureExceedsMaxOccurs	Es gibt mehr passende übereinstimmende Datensätze des Unterelements als in maxOccurs definiert sind.
StructureOutOfSequence	Die Datensätze stimmen mit den Unterelementen überein, aber nicht in der erforderlichen Sequenz. Beispiel: Die Unterelemente definieren eine Sequenz ABC, die Eingabe enthält jedoch ACB.
UnexpectedRecord	Die Datensätze stimmen mit den Unterelementen überein, aber nicht in der erforderlichen Hierarchie. Beispiel: Das Unterelement definierte eine Sequenz ABC, und D ist an einem anderen Ort definiert. Die Eingabe enthält ABD.
UnrecognizedRecord	Kein Unterelement stimmt mit den Datensatzbezeichnern überein.
XsdValidationError	Die Eingabe stimmt nicht mit den Anforderungen des Schemas überein.


Referenz der Benachrichtigungskomponente

Benachrichtigungskomponenten ("Notification") führen eine Aktion aus, wenn eine Komponente fehlschlägt.

Benachrichtigung

Die Komponente definiert den Namen einer Notification-Komponente. Konfigurieren Sie die Komponente auf der globalen Ebene des Skripts.

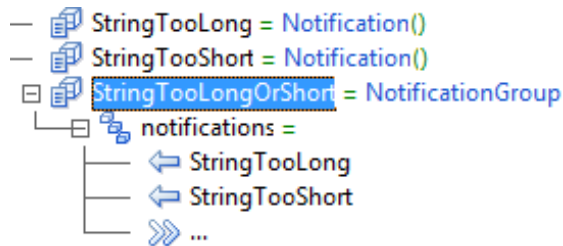
Die folgende Abbildung zeigt eine Benachrichtigung mit der Bezeichnung `StringTooLong`:

 `StringTooLong = Notification()`

NotificationGroup

Die Komponente definiert einen einzelnen Namen, der sich auf eine Gruppe von Notification-Namen bezieht. Konfigurieren Sie die Komponente auf der globalen Ebene des Skripts.

Das nachstehende Beispiel zeigt eine Gruppe mit der Bezeichnung `StringTooLongOrShort`:



Sie können einen **NotificationHandler** konfigurieren, um `StringTooLongOrShort` zu verarbeiten. Wenn eine Umwandlung eine `StringTooLong`- oder `StringTooShort`-Benachrichtigung auslöst, verarbeitet der Handler die Notification-Komponente.

Die nachstehende Tabelle beschreibt die Eigenschaften der Komponente **NotificationGroup**.

Eigenschaft	Beschreibung
notifications	Definiert eine Liste von Benachrichtigungen.

NotificationHandler

Die **NotificationHandler**-Komponente definiert eine Liste von Aktionen für eine angegebene Benachrichtigung.

Fügen Sie die **NotificationHandler**-Komponente an Positionen wie der **Benachrichtigungen**-Eigenschaft einer **Gruppe** oder **RepeatingGroup** ein. Innerhalb der Komponente können Sie eine **WriteValue**-Aktion einfügen, die eine Meldung in einem Datenbehälter speichert.

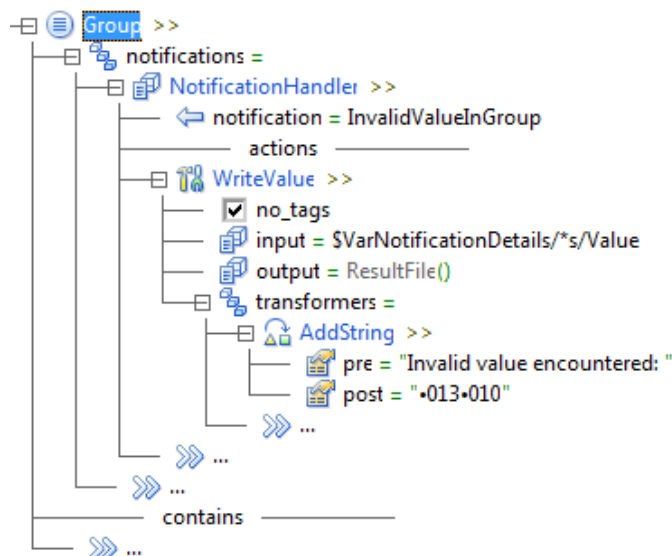
Die Variable `VarNotificationDetails` speichert Informationen über die zuletzt ausgelöste Benachrichtigung. Ein **NotificationHandler** schreibt die gespeicherten Informationen in `VarNotificationDetails` in die Ausgabe. Weitere Informationen zu `VarNotificationDetails` enthält der Abschnitt ["Systemvariablen" auf Seite 198](#).

Die nachstehende Tabelle beschreibt die Eigenschaften der Komponente **NotificationHandler**:

Eigenschaft	Beschreibung
disabled	Legt fest, ob die Komponente und alle zugehörigen untergeordneten Komponenten vom Skript ignoriert werden. Verwenden Sie diese Eigenschaft zum Testen, Debuggen und Ändern eines Skripts. Sie können eine der folgenden Optionen auswählen: <ul style="list-style-type: none">- Ausgewählt. Das Skript ignoriert die Komponente.- Gelöscht. Das Skript wendet die Komponente an. Standardmäßig ist die Eigenschaft deaktiviert.
name	Eine beschreibende Kennzeichnung für die Komponente. Diese Kennzeichnung wird in der Protokolldatei sowie der Ereignisansicht angezeigt. Ermitteln Sie anhand der Eigenschaft name , welche Komponente das Ereignis verursacht hat.

Eigenschaft	Beschreibung
Benachrichtigung	Definiert den Namen einer Benachrichtigung, die durch NotificationHandler zu verarbeiten ist. Wählen Sie eine vordefinierte Benachrichtigung oder einen Benachrichtigungsnamen, der in einer Komponente Notification oder NotificationGroup definiert ist. Zum Konfigurieren eines Handlers für die Verarbeitung einer Benachrichtigung wählen Sie anyNotification aus.
remark	Ein benutzerdefinierter Kommentar, der den Zweck oder die Aktion der Komponente beschreibt.
source	Definiert einen Datenbehälter, den Sie für die Eingabe an den NotificationHandler verwenden können.
target	Definiert einen Datenbehälter für die Ausgabe des NotificationHandler .

In der folgenden Abbildung wird ein **Gruppe**-Anker dargestellt, der mit **NotificationHandler** konfiguriert wird:



Löst eine Komponente in der **Gruppe** eine **InvalidValueInGroup**-Benachrichtigung aus, wird diese durch den Handler verarbeitet. Der Handler schreibt die *VarNotificationDetails/Value*-Variable zusammen mit einem Textstring in die Ergebnisdatei der Umwandlung.

NotifyFailure

NotifyFailure ist ein möglicher Wert der Eigenschaft **on_fail** für Anker und andere Komponenten. Wenn die Komponente fehlschlägt, löst **NotifyFailure** eine Benachrichtigung aus.

Die nachstehende Tabelle beschreibt die Eigenschaften der Komponente **NotifyFailure**.

Eigenschaft	Beschreibung
notify	Definiert eine auszulösende Notification-Komponente. Wählen Sie eine vordefinierte Benachrichtigung oder einen Benachrichtigungsnamen, der in einer Komponente Notification oder NotificationGroup definiert ist.
value	Definiert den Wert der Variablen <i>VarNotificationDetails/Value</i> . Ein NotificationHandler kann den Wert in seiner Ausgabe enthalten.

KAPITEL 24

Validierungsregeln

Dieses Kapitel umfasst die folgenden Themen:

- [Validierungsregeln - Übersicht, 435](#)
- [Validierungsregelement - Verweis, 436](#)
- [Bearbeiten der Validierungsregeln in einem externen Editor, 442](#)
- [Erstellen eines Validierungsregelobjekts, 442](#)
- [Importieren eines Data Transformation-Diensts mit Validierungsregeln, 443](#)

Validierungsregeln - Übersicht

Eine Datenprozessorumwandlung verwendet Validierungsregeln, um die Eingabe- oder Ausgabedaten für die Umwandlung zu überprüfen. Sie können ein Validierungsregelobjekt verwenden, um die XML-Daten gemäß einer Gruppe benutzerdefinierter Regeln zu validieren. Wenn die Daten die Regeln verletzen, erzeugt die Aktion einen XML-Validierungsbericht. Beim Erstellen eines Validierungsregelobjekts können Sie eine Beispieldatei zum Testen des Validierungsregelobjekts bereitstellen.

Mit Validierungsregeln werden die Eingabe- und Ausgabeelemente für eine Datenprozessorumwandlung überprüft. Verwenden Sie den Validierungsregeleditor zum Erstellen, Definieren, Bearbeiten und Verwalten von Regeln.

Nach dem Erstellen eines Validierungsregelobjekts fügen Sie Validierungsregelemente hinzu. Sie definieren jedes Element im Editor. Der Editor fügt die Elemente zur Validierungsregelhierarchie hinzu.

Die Stammebene der Hierarchie enthält eine Beschreibung des Validierungsregelobjekts sowie einen Verweis auf die XML-Beispieldatei, die zum Debuggen des Validierungsregelobjekts verwendet werden kann. Die Validierungsregelhierarchie enthält die Ordner „Lookups“ und „Regeln“. Ein Lookup-Element definiert eine Lookup-Tabelle, die Codes und Übersetzungen enthält. Sie können dem Ordner „Lookups“ mindestens ein Lookup-Element hinzufügen.

Das Regelement definiert eine Validierungsregel. Wenn die Regel auf FALSE evaluiert wird, zeichnet das Validierungsregelobjekt einen Fehler auf. Sie können dem Ordner „Regeln“ mindestens ein Regelement hinzufügen.

Sie können die folgenden Elemente in einem Regelement verschachteln:

Variablenelement

Ein Variablenelement definiert eine Variable mit einem einfachen Datentyp. Das Element enthält einen XPath-Ausdruck, der auf den Wert der Variable evaluiert wird.

Assert-Element

Das Assert-Element definiert die Logik einer Regel. Das Element enthält einen XPath-Ausdruck. Wenn der Ausdruck falsch ist, zeichnet die Regel einen Validierungsfehler auf.

Listenelement

Das Listenelement definiert eine komplexe Variable, die eine Liste mit Werten enthält.

Trace-Element

Das Trace-Element fügt der Ereignisdatei den Wert eines XPath-Ausdrucks hinzu.

Nach der Erstellung eines Elements definieren Sie die Attribute für das Element. Die Attribute definieren die Logik für das jeweilige Element.

Sie können Elemente kopieren und in den Validierungsregeleditor einfügen. Sie können das Validierungsregelobjekt auch in einem externen Editor bearbeiten und Elemente in XML hinzufügen, bearbeiten oder löschen.

Sie können einen Data Transformation-Dienst mit einer beliebigen Anzahl an VRL-Dateien importieren. Sie können die gesamte oder einen Teil einer VRL-Datei in den Validierungsregeleditor kopieren. Sie können die VRL-Datei in einem externen Editor öffnen, Elemente kopieren und diese dann in die Hierarchie des Validierungsregeleditors einfügen.

Sie können das Validierungsregelobjekt mit der Aktion **ValidateValue** aus einer Skriptkomponente der Datenprozessorumwandlung abrufen.

Validierungsregelelement - Verweis

Die oberste Ebene einer Validierungsregelhierarchie enthält ein Regel- oder Lookup-Element. Innerhalb eines Regelements können Sie die Assert-, Listen-, Trace und Variablenelemente verschachteln.

Attribute von Assert-Elementen

Das Assert-Element definiert die Logik einer Regel. Das Element enthält einen XPath-Ausdruck. Wenn der Ausdruck falsch ist, zeichnet die Regel einen Validierungsfehler auf.

Eine Regel muss mindestens ein Assert-Element enthalten.

In der folgenden Tabelle werden die Eigenschaften des Assert-Elements beschrieben:

Eigenschaft	Beschreibung
Zusätzliche Daten	Optional. Ein XPath-Ausdruck, der bewertet und in den Fehlerbericht eingefügt werden kann.
Code	Optional. Ein Zeichenfolgenattribut zur Angabe des Assert-Elements. Wenn kein Code angegeben wird, wird der Code aus der übergeordneten Regel verwendet.
Beschreibung	Optional. Eine Beschreibung des Assert-Elements. Wenn keine Beschreibung angegeben wird, wird die Beschreibung aus der übergeordneten Regel verwendet.
Speicherort	Optional. Ein Zeichenfolgenwert, der an den XPath-Ausdruck angehängt werden kann, der dem von der übergeordneten Regel ausgewählten Knoten entspricht. Der Ausdruck wird dann in den Fehlerbericht eingefügt.

Eigenschaft	Beschreibung
Regelbeschreibung	Eine schreibgeschützte Beschreibung der Regel, unter der das Element verschachtelt ist.
XPath	Obligatorisch. Ein boolescher XPath-Ausdruck, der die Logik der Regel ausdrückt.

Listenelementattribute

Das Listenelement definiert eine komplexe Variable, die eine Liste mit Werten enthält.

In der folgenden Tabelle werden die Eigenschaften des Listenelements beschrieben:

Eigenschaft	Beschreibung
Anhängen	Ein boolesches Attribut, das bestimmt, was geschehen soll, wenn die Liste denselben Namen trägt wie eine vorhandene Liste. Wenn diese Eigenschaft aktiviert ist, werden die neuen Werte zur vorhandenen Liste hinzugefügt. Wenn diese Eigenschaft deaktiviert ist, wird die vorhandene Liste gelöscht und eine neue Liste wird erstellt. Ist standardmäßig aktiviert.
Funktion	Optional. Ein XPath-Ausdruck, der relativ zu jedem der ausgewählten Knoten evaluiert wird. Der Wert des Ausdrucks wird zur Liste hinzugefügt.
Name	Obligatorisch. Der Name der Liste. Verwenden Sie diesen Namen, um in XPath-Ausdrücken von Elementen, die in der übergeordneten Regel verschachtelt sind, auf die Liste zu verweisen.
Regelbeschreibung	Eine schreibgeschützte Beschreibung der Regel, unter der das Element verschachtelt ist.
Auswählen	Obligatorisch. Ein XPath-Ausdruck, der Knoten zum Berechnen der Elemente in der Liste auswählt.

Lookup-Elementattribute

Das Lookup-Element definiert eine Lookup-Tabelle, die Codes und Übersetzungen enthält.

In der folgenden Tabelle werden die Eigenschaften des Lookup-Elements beschrieben:

Eigenschaft	Beschreibung
Datei	Obligatorisch. Die Datei, die die Lookup-Tabelle enthält.
Name	Obligatorisch. Der Name der Lookup-Tabelle.

Lookup-Datei

Das folgende Beispiel zeigt eine Lookup-Beispieldatei:

```
<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>
```

Das Lookup-Tabellenformat ist identisch mit einer `XMLLookupTable`, die in der Komponente `LookupTransformer` verwendet wird. Weitere Informationen hierzu finden Sie unter ["LookupTransformer" auf Seite 284](#).

Sie können die Funktion `dt:lookup()` in einem XPath-Ausdruck verwenden, um nach einem Wert in der Lookup-Tabelle zu suchen.

Regelementattribute

Das Regelement definiert eine Validierungsregel. Wenn die Regel auf FALSE evaluiert wird, zeichnet die Validierungsregel einen Fehler auf.

In der folgenden Tabelle werden die Eigenschaften des Regelements beschrieben:

Eigenschaft	Beschreibung
Code	Obligatorisch. Eine ID für die Regel. Muss ein einzelnes Wort sein.
Beschreibung	Obligatorisch. Eine Beschreibung der Regel mit frei formuliertem Text in beliebiger Länge.
Aktiviert	Legt fest, ob die Regel aktiviert oder deaktiviert ist. Ist standardmäßig aktiviert.
Alle Asserts ausführen	Gibt an, ob die unter der Regel verschachtelten Asserts ausgeführt werden. Da die Eigenschaft standardmäßig aktiviert ist, verarbeitet die Regel alle Asserts. Wenn die Eigenschaft deaktiviert ist, stoppt die Regel die Verarbeitung von Asserts, sobald ein Assert fehlschlägt.
Auswählen	Obligatorisch. Ein XPath-Ausdruck, der den Knoten oder Knotensatz auswählt, für den die Regel gilt.

Trace-Elementattribute

Das Trace-Element fügt dem Fehlerbericht den Wert eines XPath-Ausdrucks hinzu. Das Element enthält den XPath-Ausdruck.

In der folgenden Tabelle werden die Eigenschaften des Trace-Elements beschrieben:

Eigenschaft	Beschreibung
Beschreibung	Obligatorisch. Eine Beschreibung des Trace-Elements.
Aktiviert	Gibt an, ob das Trace-Element aktiviert oder deaktiviert ist. Ist standardmäßig aktiviert.
Regelbeschreibung	Eine schreibgeschützte Beschreibung der Regel, unter der das Element verschachtelt ist.
XPath	Obligatorisch. Ein XPath-Ausdruck, der bewertet und in den Fehlerbericht eingefügt werden kann.

Variablenelementattribute

Ein Variablenelement definiert eine Variable mit einem einfachen Datentyp. Das Element enthält einen XPath-Ausdruck. Der Wert der Variable entspricht dem Wert des Ausdrucks.

In der folgenden Tabelle werden die Eigenschaften des Variablenelements beschrieben:

Eigenschaft	Beschreibung
Name	Obligatorisch. Der Name der Variable. Verwenden Sie diesen Namen, um in XPath-Ausdrücken von Elementen, die in der übergeordneten Regel verschachtelt sind, auf die Variable zu verweisen.
Regelbeschreibung	Eine schreibgeschützte Beschreibung der Regel, unter der das Element verschachtelt ist.
XPath	Obligatorisch. Ein XPath-Ausdruck zum Berechnen von oder Verweisen auf die Knoten, für die die Variable gilt.

XPath-Editor

Bestimmte Elemente enthalten XPath-Ausdrücke. Zum Erstellen der Ausdrücke geben Sie einen XPath-Ausdruck im XPath-Editor ein.

XPath-Erweiterungen

Die Regel-, Assert-, Listen-, Variablen- und Trace-Elemente können XPath-Ausdrücke enthalten. Die folgenden Funktionen können Teil des XPath-Ausdrucks sein.

Das Validierungsregelobjekt definiert die folgenden Funktionen als Erweiterungen von XPath 1.0. Die Erweiterungen gehören zum Namensraum **dt**, der als <http://validations.informatica.com> definiert ist.

Funktion	Beschreibung
<code>dt:all-equal(param1[, param2, ...])</code>	Gibt TRUE zurück, wenn alle Elemente in der Liste gleich sind. Bei den Parametern kann es sich um einfache Werte oder Listen handeln.
<code>dt:check-uniqueness(value1[, value2, ...])</code>	Gibt TRUE zurück, wenn alle Werte in der Parameterliste eindeutig sind. Jeder Wert kann Folgendes sein: <ul style="list-style-type: none">- Ein einfacher Wert- Eine Variable, die eine Liste mit Zeichenfolgen enthält- Eine Variable, die eine Liste mit Knoten enthält, wobei jeder Knoten ein einfacher Wert ist.
<code>dt:date-add(startDate, format, numberOfDays)</code>	Gibt das Datum zurück. Wenn numberOfDays den Typ float aufweist, rundet die Funktion den Wert auf die nächstniedrigere Ganzzahl ab.
<code>dt:date-diff(startDate, format, endDate, format)</code>	Gibt die Anzahl an Tagen zwischen zwei Datumsangaben zurück.

Funktion	Beschreibung
<code>dt:date-format (date, inputFormat, outputFormat)</code>	Wandelt das Format von date aus inputFormat in outputFormat um.
<code>dt:date-valid (string-to-match, format)</code>	Gibt TRUE zurück, wenn die Zeichenfolge mit dem angegebenen Datumsformat übereinstimmt.
<code>dt:deep-equal (element1, element2)</code>	Gibt TRUE zurück, wenn beide Bedingungen wahr sind: <ul style="list-style-type: none"> - element1 und element2 weisen dieselben Attribute mit denselben Werten. - element1 und element2 weisen dieselben untergeordneten Elemente in derselben Reihenfolge auf und alle untergeordneten Elemente sind rekursiv gleich.
<code>dt:empty (xpath)</code>	Gibt TRUE zurück, wenn ein XPath keine untergeordneten Elemente enthält.
<code>dt:exist (xpath)</code>	Gibt TRUE zurück, wenn ein XPath vorhanden ist. Äquivalent zu count(xpath) > 0 .
<code>dt:is-sorted-lex (ascending, param1[, param2, ...])</code>	Gibt TRUE zurück, wenn die Listenelemente lexikografisch sortiert sind. Der erste Parameter ist ein boolescher Wert, der die Sortierrichtung festlegt: true für aufsteigend oder false für absteigend.
<code>dt:last-day-of-month (date, format)</code>	Gibt den letzten Tag des ausgewählten Monats zurück.
<code>dt:list-items (list, separator)</code>	Gibt eine Zeichenfolge aller Elemente in der Liste zurück (getrennt durch das Trennzeichen). Als Standardtrennzeichen wird ein Komma verwendet.
<code>dt:lookup (string, lookupName, default)</code>	Sucht nach einer Zeichenfolge in einer Lookup-Tabelle. Geben Sie die zu suchende Zeichenfolge, den Namen der Lookup-Tabelle und einen Standardwert an, der zurückgegeben werden soll, wenn die Zeichenfolge in der Tabelle nicht vorhanden ist.
<code>dt:min-lex (value1[, value2, ...])</code>	Sortiert die Liste der Eingabeparameter lexikografisch und gibt das erste Element in der sortierten Liste zurück. Jeder Wert kann Folgendes sein: <ul style="list-style-type: none"> - Ein einfacher Wert - Eine Variable, die eine Liste mit Zeichenfolgen enthält - Eine Variable, die eine Liste mit Knoten enthält, wobei jeder Knoten ein einfacher Wert ist.
<code>dt:next-sequence ()</code>	Gibt eine Knotengruppe mit dem aktuellen Element und allen folgenden gleichgeordneten Elementen bis zu einem anderen Element desselben Namens zurück.
<code>dt:regex-match (string-to-match, regex)</code>	Gibt TRUE zurück, wenn eine Zeichenfolge einem regulären Ausdruck entspricht.

Funktion	Beschreibung
<code>dt:regex-replace(inputString, patternRegex, replacementString)</code>	Gibt inputString mit allen Instanzen von patternRegex ersetzt durch replacementString zurück. Sind keine Entsprechungen für patternRegex vorhanden, wird der Wert inputString ohne Änderungen zurückgegeben.
<code>dt:string-replace(string1, string2, string3)</code>	Ersetzt alle Instanzen von string2 in string1 durch string3 .
<code>dt:unadjusted-calculation- period-dates(startDate, endDate, timeUnit, timeUnitMultiplier, lookupDate, dateFormat)</code>	Gibt TRUE zurück, wenn sich lookupDate innerhalb des ersten angegebenen Zeitraums im festgelegten Bereich befindet, wobei der Zeitraum timeUnitMultiplier mal timeUnit ist und der Bereich zwischen startDate und endDate liegt. <ul style="list-style-type: none"> - timeUnitMultiplier ist eine Ganzzahl. - timeUnit ist eine der folgenden Zeichenfolgen: <ul style="list-style-type: none"> - Tag oder T - Woche oder W - Monat oder M - Jahr oder J - lookupDate ist ein Datum im Bereich zwischen startDate und endDate. - dateFormat definiert die Formate von lookupDate, startDate und endDate

Weitere Informationen über das Datumsformat finden Sie unter [“DateFormatICU” auf Seite 270](#).

Beispiel: Verwendung von dt:next-sequence()

Sie können die Funktion `dt:next-sequence()` verwenden, um logisch auf hierarchische Daten zuzugreifen, die nicht im aktuellen Element verschachtelt sind.

Beachten Sie die folgende Eingabe:

```
<Root>
  <A/>
  <B/>
  <C/>
</Root>
```

Der XPath `/Root/A/dt:next-sequence()` gibt folgenden Knotensatz zurück:

```
<A/>
<B/>
<C/>
```

Im folgenden Beispiel ist jedes `id`-Element mit einer Gruppe von gleichgeordneten Elementen mit der Bezeichnung `name`, `quantity` und `price` verknüpft:

```
<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>
```

Die Elemente `name`, `quantity` und `price` sind alle logisch in `id` verschachtelt, selbst wenn sie physisch nicht verschachtelt sind.

Die folgende Regel erfordert, dass `price*quantity` für jedes `id`-Element kleiner oder gleich 50 ist:

```
<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>
```

Für das erste `id`-Element ist die Variable `$current` ein Knotensatz mit der folgenden Regel:

```
<id>100</id>
<name>Plate</name>
<quantity>4</quantity>
<price>10</price>
```

Der Ausdruck `$current[3]*$current[4]` liefert das Ergebnis $4 \cdot 10 = 40$. Die Regel bestätigt, dass $40 < 50$ ist.

Für die nachfolgenden `id`-Elemente evaluiert die Regel den Ausdruck in ähnlicher Weise.

Bearbeiten der Validierungsregeln in einem externen Editor

Sie können Validierungsregeln in einem externen Editor bearbeiten. Der Editor zeigt die Validierungsregelhierarchie in XML mit allen verschachtelten Elementen an, die Sie erstellt haben. Sie können Elemente in XML kopieren, bearbeiten oder löschen.

Hinweis: Wenn Sie beim Bearbeiten eines Validierungsregelobjekts in einem externen Editor die Änderungen speichern, werden die gesamte Umwandlung und nicht nur die Änderungen an den Validierungsregeln gespeichert.

Erstellen eines Validierungsregelobjekts

Sie können den Validierungsregeleditor verwenden, um ein Validierungsregelobjekt zu erstellen, anzuzeigen und zu bearbeiten.

1. Erstellen Sie in der Ansicht **Objekte** der Datenprozessorumwandlung Validierungsregeln.
2. Klicken Sie zum Öffnen der Validierungsregeln auf das Validierungsregelobjekt.
3. Klicken Sie zum Hinzufügen eines Elements im linken Bereich des Editors mit der rechten Maustaste auf die Regelhierarchie und wählen Sie **Neu > <Element>** aus, wobei `<Element>` für den Elementtyp steht, der hinzugefügt werden soll.
Ein leeres Element wird angezeigt.
4. Definieren Sie im rechten Bereich die Attribute für das Element.
5. Fügen Sie jedem Element nach Bedarf Elemente und Attribute hinzu.

Sie können zu jedem Zeitpunkt das Kontextmenü öffnen und die folgenden Vorgänge durchführen:

- Anhängen eines gleichgeordneten Elements

- Hinzufügen eines untergeordneten Elements
 - Löschen eines Elements
 - Aktivieren oder Deaktivieren einer Regel
 - Rückgängigmachen oder Wiederherstellen von Vorgängen
6. Speichern Sie das Validierungsregelobjekt.

Importieren eines Data Transformation-Diensts mit Validierungsregeln

Sie können einen Data Transformation-Dienst, der Validierungsregeln enthält, aus dem Dateisystem-Repository des Computers importieren, auf dem Sie den Dienst gespeichert haben. Importieren Sie eine CMW-Datei des Data Transformation-Diensts in das Modellrepository, um eine Datenprozessorumwandlung zu erstellen. Das Developer-Tool importiert die Umwandlung und die Validierungsregeln mit der CMW-Datei.

1. Klicken Sie auf **Datei > Importieren**,
Das Dialogfeld **Importieren** wird eingeblendet.
2. Wählen Sie **InformaticaData Transformation-Dienst importieren** aus und klicken Sie auf **Weiter**.
Die Seite **Data Transformation-Dienst importieren** wird angezeigt.
3. Navigieren Sie zur Dienstdatei mit der Erweiterung .cmw, die Sie importieren möchten.
Das Developer-Tool benennt die Umwandlung entsprechend des Dienstdateinamens. Sie können den Namen ändern.
4. Navigieren Sie zu einem Speicherort im Repository, wo Sie die Umwandlung speichern möchten. Dann klicken Sie auf **Fertig stellen**.
Das Developer-Tool importiert die Umwandlung und die Validierungsregeln mit der Datei .cmw.
5. Doppelklicken Sie zum Bearbeiten des Validierungsregelobjekts in der Ansicht **Objekt-Explorer** auf das Validierungsregelobjekt.
Der Validierungsregeleditor wird geöffnet und zeigt die Hierarchie des Validierungsregelobjekts an.

KAPITEL 25

Benutzerdefinierte Skriptkomponenten

Dieses Kapitel umfasst die folgenden Themen:

- [Überblick über benutzerdefinierte Skriptkomponenten, 444](#)
- [Beispiel für eine benutzerdefinierte Komponente, 444](#)
- [Eigenschaften der benutzerdefinierten Komponente, 445](#)
- [Entwickeln einer benutzerdefinierten Komponente, 445](#)
- [Konfigurieren einer benutzerdefinierten Komponente, 447](#)

Überblick über benutzerdefinierte Skriptkomponenten

Wenn Sie ein Datenprozessor-Umwandlungsskript entwerfen und konfigurieren, können Sie eine große Anzahl von integrierten Komponenten verwenden. Sie können auch benutzerdefinierte Komponenten programmieren, beispielsweise Dokumentprozessoren oder Transformer, und in ein Skript einfügen. Wenn Sie die Datenprozessor-Umwandlung als Datenumwandlungsdienst exportieren, läuft der Dienst mit den benutzerdefinierten Komponenten.

Sie implementieren die benutzerdefinierten Komponenten in Java. Weitere Informationen zu den zu implementierenden Schnittstellen finden Sie in der *Java-Schnittstellenreferenz für externe Komponenten*.

Beispiel für eine benutzerdefinierte Komponente

Angenommen, Sie müssen ein proprietäres binäres Datenformat parsen. Statt die binären Daten direkt zu parsen, empfiehlt es sich, diese in eine Textdarstellung umzuwandeln, die problemloser geparkt werden kann.

Zu diesem Zweck können Sie einen benutzerdefinierten Dokumentprozessor programmieren, den Sie beispielsweise `MyBinaryToText` nennen. Der Prozessor weist möglicherweise folgende Eigenschaften auf:

Eigenschaft	Beschreibung
KeepLineBreaks	Eine boolesche Eigenschaft. True (wahr) bedeutet, dass der Prozessor die Zeilenumbrüche in den binären Daten beibehält.
MaxLineLength	Eine integer-Eigenschaft. Gibt die Maximallänge der Textlänge für die Ausgabe an.
Ignore	Eine String-Eigenschaft. Sie weist den Prozessor an, Datenfelder, die mit einem bestimmten String beginnen, zu ignorieren.

Nachdem Sie den Prozessor entwickelt haben, können Sie ihn installieren und in Skripten einsetzen.

Eigenschaften der benutzerdefinierten Komponente

Die Eigenschaften einer benutzerdefinierten Komponente kann die Datentypen "integer", "Boolean", "string" oder "list-of-string" haben. Sie können entweder einen konstanten Eigenschaftswert oder den Namen eines Datenbehälters zuordnen, der den Wert enthält.

Sie können einige der Eigenschaften im IntelliScript-Editor verbergen. Beispielsweise kann eine benutzerdefinierte Komponente vier Eigenschaften unterstützen. In der TGP-Konfigurationsdatei können Sie die Komponente so konfigurieren, dass nur die ersten beiden Eigenschaften angezeigt werden. Das Skript übergibt nur die angezeigten Eigenschaften an die Komponente. Die Komponente kann den versteckten Eigenschaften die eigenen Standardwerte zuweisen.

Die maximale Anzahl der Eigenschaften, die eine benutzerdefinierte Komponente haben kann, hängt vom Komponententyp ab. Für eine Dokumentprozessorkomponente beträgt die maximale Anzahl der Eigenschaften 4. Für eine Transformer-Komponente beträgt die maximale Anzahl der Eigenschaften zehn.

Entwickeln einer benutzerdefinierten Komponente

1. Erstellen Sie eine Klasse, die eine oder mehrere der folgenden Schnittstellen implementiert.

Komponententyp	Eingabetyp	Schnittstelle
Dokumentprozessor	Datei	CMXFileProcessor
Dokumentprozessor	Puffer	CMXByteArrayProcessor
Transformer	String	CMXStringTransformer
Transformer	Puffer	CMXByteArrayTransform

Weitere Informationen zu diesen Schnittstellen finden Sie unter *Externe Komponenten – Java-Schnittstellen-Referenz*.

2. Kompilieren Sie das Projekt in einer JAR-Datei.
3. Speichern Sie die JAR-Datei im Unterverzeichnis `externLibs\user` des Installationsverzeichnis auf jedem Computer, auf dem Sie die Komponente verwenden möchten.
4. Erstellen Sie eine Skriptdatei, die den Anzeigenamen der Komponente und ihrer Eigenschaften definiert. Speichern Sie die Datei im Unterverzeichnis `autoInclude\user` des Installationsverzeichnis.

Weitere Informationen zu diesem Schritt finden Sie unter ["Konfigurieren einer benutzerdefinierten Komponente" auf Seite 447](#).

Anschließend können Sie die benutzerdefinierten Komponenten in Umwandlungen verwenden.

Beispiel für eine Java-Benutzeroberfläche

Denken Sie beispielsweise an einen Dokumentprozessor, der als Eingabe eine Datei akzeptiert. Der Prozessor muss die `CMXFileProcessor`-Klasse mit folgender Methode implementieren:

```
public String process(
    CMXContext context,
    String in,
    String additionalFilesDir,
    CMXEventReporter reporter)
    throws Exception
```

Die Parameter haben die folgende Bedeutung:

Parameter	Beschreibung
Kontext	Eingabeparameter. Ein Objekt, das die Eigenschaften enthält, die das Skript an die Komponente übergibt. Die Methode parameters des Objekts gibt einen Vektor zurück, der die Eigenschaftswerte enthält.
Ein	Eingabeparameter. Der vollständige Pfad der Datei, mit der die Komponente arbeitet.
additionalFilesDir	Optionaler Ausgabeparameter. Der Pfad eines temporären Verzeichnisses, in das die Komponente die Dateien schreibt. Nach Abschluss der Verarbeitung löscht das Skript den gesamten Inhalt des Verzeichnisses.
reporter	Eingabeparameter. Ein Objekt, das die Methode "report" bereitstellt, mit der die Komponente Ereignisse in die Protokolldatei schreiben kann.

Beispiel für benutzerdefinierte Java-Komponenten

Beispiele für die Implementierung der benutzerdefinierten Komponenten finden Sie im folgenden Unterverzeichnis des Installationsverzeichnis:

```
samples\SDK\ExternalParameters\Java_SDK\Java
```

Das Verzeichnis enthält die folgenden Beispiele:

Beispiel	Beschreibung
FilePP.java	Ein Dokumentprozessor, der als Eingabe eine Datei akzeptiert.
ByteArrayPP.java	Ein Dokumentprozessor, der als Eingabe einen Puffer akzeptiert.

Beispiel	Beschreibung
StringTT.java	Ein Transformer, der als Eingabe einen String akzeptiert.
ByteArrayTT.java	Ein Transformer, der als Eingabe einen Puffer akzeptiert.

Konfigurieren einer benutzerdefinierten Komponente

Nachdem Sie eine benutzerdefinierte Komponente entwickelt haben, müssen Sie eine Skriptdatei vorbereiten, die die Komponente definiert. Die TGP-Datei können Sie nicht im IntelliScript-Editor vorbereiten. Stattdessen müssen Sie dafür einen Texteditor verwenden.

Nach der Installation der Komponente und der TGP-Datei kann die externe Komponente im IntelliScript-Editor konfiguriert werden.

1. Erstellen Sie eine Textdatei und speichern Sie sie mit der Erweiterung *.tgp.

Hinweis: Sie können in einer einzigen TGP-Datei mehr als eine externe Komponente definieren.

2. Fügen Sie der TGP-Datei für jede Eigenschaft, die Ihre externe Komponente unterstützt, Zeilen wie die folgenden hinzu:

```
profile <CustomPropertyName> ofPT <DataType>
{
    paramName = "<CustomPropertyName>" ;
}
```

<CustomPropertyName> ist der Name einer Eigenschaft, die im IntelliScript-Editor angezeigt werden soll. <DataType> ist der Datentyp der Eigenschaft. Die unterstützten Datentypen sind NamedParamIntT für eine numerische Eigenschaft, NamedParamBoolT für eine Boolesche Eigenschaft, NamedParamStringT für eine Stringeigenschaft oder NamedParamListT für eine Eigenschaft in Form einer Stringliste.

3. Fügen Sie der TGP-Datei für jede externe Komponente, die Sie definieren möchten, Zeilen wie die folgenden hinzu:

```
profile <ExternalComponentName> ofPT <ComponentType>
{
    jclass = "<ClassName>" ;
    param1 = <CustomPropertyName>() ;
    param2 = <CustomPropertyName2>() ;
}
```

<ExternalComponentName> ist der Name der externen Komponente, die im IntelliScript-Editor angezeigt werden soll. <ComponentType> ist einer der folgenden Werte:

Für	Komponententyp
Ein Java-Dokumentprozessor mit 0 bis 4 Eigenschaften	ExternalJavaProcessorNoParamsT ExternalJavaProcessor1ParamsT ExternalJavaProcessor2ParamsT ...
Ein Java-Transformer mit 0 bis 10 Eigenschaften	ExternalJavaTransformerNoParamsT ExternalJavaTransformer1ParamsT ExternalJavaTransformer2ParamsT ...

<ClassName> ist der vollständige Name der Java-Klasse. Bei Windows ist <DllName> der Name der DLL, ohne die Erweiterung *.dll. Unter Linux oder UNIX ist es der Name des gemeinsam verwendeten Objekts, ohne das Präfix lib oder die Erweiterung *.so.

<CustomPropertyName1>, <CustomPropertyName2> usw. sind die Namen der Eigenschaften, die Sie im 2. Schritt konfiguriert haben.

4. Speichern Sie die *.tgp-Datei.
5. Speichern Sie die Datei im Unterverzeichnis `DataTransformation\autoInclude\user` des Installationsverzeichnisses eines jeden Computers, auf dem Sie die Komponente verwenden möchten.
6. Wenn das Developer-Tool geöffnet ist, schließen Sie es und öffnen es dann erneut.
7. Falls ein `autoInclude`-Fehler angezeigt wird, prüfen Sie die TGP-Datei auf Syntaxfehler oder inkonsistente Namen. Öffnen Sie das Developer-Tool dann erneut.
8. Öffnen Sie ein Projekt und fügen Sie die benutzerdefinierte Komponente in das Skript ein. Der Name der benutzerdefinierten Komponente, den Sie in Schritt 3 zugewiesen haben, sollte jetzt in der Dropdown-Liste des IntelliScript angezeigt werden. Der IntelliScript-Editor zeigt die betreffenden Eigenschaften an.

Beispiel für Skript mit benutzerdefinierten Komponenten

Beispiele von Skriptdateien, die benutzerdefinierte Komponenten enthalten, finden Sie in den folgenden Unterverzeichnissen des Installationsverzeichnisses:

```
samples\SDK\ExternalParameters\Java_SDK\autoInclude
samples\SDK\ExternalParameters\Cpp_SDK\autoInclude
```


INDEX

A

abgeleitete Datentypen

XSI-Typ [196](#)

Abstrakt, Eigenschaft

Schemaobjekt [127](#)

AbsURL

Komponente [264](#)

AcroForms

Verarbeiten von PDF-Formularen [168](#)

AddEmptyTagsTransformer

Komponente [264](#)

AddEventAction

Komponente [303](#)

AddHeaderModifier

Komponente [405](#)

AdditionalInputPort

Komponente [154](#)

AdditionalOutputPort

Komponente [156](#)

AddString

Komponente [265](#)

AddStringModifier

Komponente [406](#)

AggregateValues

Komponente [304](#)

Aktionen

definieren [302](#)

Eigenschaften von [303](#), [414](#)

Eingabe und Ausgabe [301](#)

Nebenwirkungen [301](#)

verglichen mit Transformatoren [302](#)

AllStructure

Komponente [253](#)

AllStructureLocal

Komponente [253](#)

alternative Parser

auswählen [219](#)

AlternativeMappings

Komponente [363](#)

Alternatives

Komponente [217](#)

AlternativeSerializers

Komponente [347](#)

AlternativeValidators

Komponente [415](#)

Anker

Beziehung zu Delimitern [206](#)

Beziehung zu XML [206](#)

definieren [208](#)

Eigenschaften von [209](#)

Marker und Content [205](#)

Phase [211](#)

Position in IntelliScript [208](#)

Referenz [217](#)

Schachtelungsbereich [217](#)

Anker (*Fortsetzung*)

Serialisierung [340](#), [343](#)

Transformer verwenden [261](#)

Anker, Wiederholungsgruppe

alle Iterationen hervorheben [146](#)

Ansicht „Ports“

Datenprozessor-Umwandlung [65](#)

Datenprozessorumwandlung [71](#)

Ansicht anpassen

XMap-Schema, Bereich [85](#)

Ansicht der Datenprozessor-Ereignisse

Datenprozessorumwandlung [36](#)

Ansicht der Einstellungen

Datenprozessorumwandlung [28](#)

Ansicht für Ereignisse, Datenprozessor

Ereignisprotokoll anzeigen [38](#)

Ansicht, Hexadezimale Quelle

Beschreibung [147](#)

Ansicht, IntelliScript-Hilfe

Beschreibung [147](#)

Anweisungen für Wiederholungsgruppen

Beschreibung [91](#)

AppendListItems

Komponente [306](#)

AppendValues

Komponente [308](#)

Assistent für Datenprozessor-Umwandlung

Beschreibung [48](#)

Attribute

Datenbehälter [192](#)

attributeFormDefault

Schemaobjekt [131](#)

Attributeigenschaften

Schemaobjekt [131](#)

AttributeSearch

Komponente [248](#)

aufteilen

Dateien [337](#)

Aufzählungen

Komponente [417](#)

Aufzählungen, Eigenschaft

Schemaobjekt [127](#)

Ausdruck testen

XPath-Ausdruckseditor für Eingabe [108](#)

Ausdruckseditor für Eingabe

Datenprozessor-Umwandlung [108](#)

Ausgabesteuerungseinstellungen

Datenprozessorumwandlung [31](#)

Ausgeblendete Eigenschaften

anzeigen [143](#)

Ausschneiden und Einfügen

Skriptkomponenten [147](#)

Auswahl, Eigenschaft

Schemaobjekt [129](#)

Auswählen und Anklicken

Anker definieren [209](#)

autoInclude
Benutzerdefinierte Komponenten [447](#)

B

Base64Decoder
Komponente [266](#)
Base64Encoder
Komponente [266](#)
Basis, Eigenschaft
Schemaobjekt [129](#)
Befehlszeilenschnittstelle
Befehl CM_console [137](#)
Behandlung
Fehlsläche [410](#)
Validierungsfehler [147](#)
Beispiel-Quelldokument, Hervorhebung
Beschreibung [146](#)
Beispieldatei
Avro [49](#)
JSON [57](#)
Parquet [60](#)
XML [62](#)
Beispielquelldatei
Definieren in der Datenprozessorumwandlung [41](#), [75](#)
Beispielquelle
anzeigen [146](#)
Beschreibung [145](#)
Einstellung [146](#)
In der Datenprozessorumwandlung erstellen [43](#)
XMap-Schema, Bereich [85](#)
Beispiels-Eingabedatei
Datenprozessorumwandlung [25](#)
Beispielskript
importieren [149](#)
Benachrichtigen
Komponenten [324](#)
Benachrichtigung - Ereignis
Datenprozessorumwandlung [35](#)
Benachrichtigungen
Auslösen [324](#)
Generieren [409](#)
Schreiben von Nachrichten [430](#)
Benutzerdefinierte Komponente
Beispiel [444](#)
Java, unter Java entwickeln [445](#)
Benutzerdefinierte Skriptkomponenten
Beschreibung [444](#)
Eigenschaften [445](#)
Benutzerprotokoll
Datenprozessorumwandlung [38](#)
Speicherort definierende Variable [199](#)
Berichtsgenerator
BIRT [170](#), [171](#)
Bestätigen
Validierungsregelement [436](#)
Bezeichner
IntelliScript [80](#)
Bibliothek
Elementeigenschaften [119](#)
In der Datenprozessorumwandlung erstellen [42](#)
Übersicht [118](#)
BidiConvert
Komponente [267](#)
BinaryFormat
Komponente [180](#)

BIRT
Berichtsgenerator XmlToDocument_372 [170](#)
XmlToDocument_45-Berichtsgenerator [171](#)
XmlToDocument-Berichtsgenerator [170](#), [171](#)
Block, Eigenschaft
Schemaobjekt [128](#)

C

CalculateValue
Komponente [309](#)
CDATADecode
Komponente [268](#)
CDATAEncode
Komponente [268](#)
ChangeCase
Komponente [269](#)
ChoiceStructure
Komponente [254](#)
ChoiceStructureLocal
Komponente [255](#)
COBOL
Importieren einer Datendefinition [54](#)
Parser testen [55](#)
Serializer testen [56](#)
unterstützte Funktionen [55](#)
Codepages
umwandeln [275](#)
XSD-Schema [193](#)
Codieren
Codepage-Transformer [275](#)
Codierung
XSD-Schema [193](#)
Codierungsrichtlinien
Datenprozessorumwandlung [31](#)
CombineValues
Komponente [311](#)
CommaDelimited
Komponente [186](#)
ComplexSegment
Komponente [395](#)
ComplexXmlSegment
Komponente [395](#)
condition
im Quelldokument einhalten [318](#)
Connect
Komponente [256](#)
Content
Komponente [220](#)
Content abrufen
Content-Anker [220](#)
Content extrahieren
Content-Anker [220](#)
ContentSerializer
Komponente [343](#), [348](#)
CreateGuid
Komponente [270](#)
CreateList
Komponente [313](#)
CreateUUID
Komponente [270](#)
CustomFormat
Komponente [181](#)
CustomLog
Komponente [314](#)

D

- Data Transformation-Dienst
 - Importieren in das Modellrepository [45](#)
 - Mehrere Dienste importieren [45](#)
- Data Transformation-Dienst:
 - Importieren in das Modellrepository [45](#)
- DateAddICU
 - Komponente [314](#)
- DateDiff
 - Komponente [315](#)
- DateDiffICU
 - Komponente [315](#)
- DateFormat
 - Komponente [271](#)
- DateFormatICU
 - Komponente [270](#)
- Dateiausgabeport
 - Datenprozessorumwandlung [26](#)
- Dateeingabeport
 - Datenprozessorumwandlung [25](#)
- Daten
 - validieren [413](#)
- Datenbehälter
 - Einzel- oder Mehrfachinstanz [202](#)
 - gemischter Inhalt [195](#)
 - Instanzen löschen [204](#)
 - Mehrfachinstanz indexieren [369](#)
 - source und target identifizieren [374](#)
- Datenprozessor-Ereignisse-Ansicht
 - Ereignisprotokoll anzeigen [38](#)
- Datenprozessor-Funktionen
 - Beschreibung [109](#)
- Datenprozessor-Hex-Quelle-Ansicht
 - Beschreibung [24](#)
- Datenprozessor-Umwandlung
 - Denormalisierte relationale Ausgabe [74](#)
 - Denormalisierte relationale Eingabe [69](#)
 - Normalisierte relationale Ausgabe [73](#)
 - Normalisierte relationale Eingabe [68](#)
 - Pivotierte Ausgabe [73](#)
 - Pivotierte Eingabe [69](#)
 - Relationale Ausgabe
 - Datenprozessorumwandlung [72](#)
 - Relationale Eingabe
 - Datenprozessorumwandlung [66](#)
 - Zuordnen von XML zu relationalen Ports [65](#)
- Datenprozessorumwandlung
 - Einstellungen für die Codierung [29](#)
 - Ansichten [24](#)
 - Ausgabeports [26](#)
 - Ausgabesteuerungseinstellungen [31](#)
 - Benutzerprotokolle [38](#)
 - Beschreibung [23](#), [24](#)
 - Eingabeports [25](#)
 - Einstellungen [28](#)
 - erstellen [39](#)
 - exportieren als Dienst [44](#)
 - nicht native Umgebung [47](#)
 - Ports [25](#)
 - Ports für Dienstparameter [26](#)
 - Startkomponente [27](#)
 - Testen einer Bibliothek [121](#)
 - testen im Daten-Viewer [43](#)
 - Verarbeitungseinstellungen [32](#)
 - XMap-Einstellungen [33](#)
 - XML-Einstellungen [33](#)
 - Zuordnen von XML zu Ports [71](#)

- Datensätze
 - Extrahieren in StructureDefinition [230](#)
 - extrahieren und validieren [244](#)
- Datentypen
 - suchen nach [215](#)
- Datumwandlungsdienst:
 - Ausführung von der Befehlszeile aus [137](#)
- Datumsangaben
 - Format von [270](#)
- Default
 - Komponente [96](#)
- Default-Anweisung
 - XMap-Editor [87](#)
- DelimitedSections
 - Komponente [224](#)
- DelimitedSectionsSerializer
 - Komponente [349](#)
- Delimiter
 - Komponente [187](#)
- DelimiterHierarchy
 - Komponente [187](#)
- delimiters
 - benutzerdefinierte Hierarchie [181](#)
 - Beziehung zu Ankern [206](#)
- Denormalisierte relationale Ausgabe
 - Datenprozessor-Umwandlung [74](#)
- Denormalisierte relationale Eingabe
 - Datenprozessor-Umwandlung [69](#)
- Dienst, Datumwandlung
 - Ausführung von der Befehlszeile aus [137](#)
- Dienstparameter
 - an Umwandlung übergeben [201](#)
- Dienstparameterport
 - Datenprozessorumwandlung [25](#)
- DocList
 - Komponente [159](#)
- Dokument, Beispielquelle
 - Beschreibung [145](#)
- Dokumentprozessoren
 - benutzerdefiniert, Java [167](#)
 - definieren [163](#)
 - mehrere ausführen [170](#)
 - Referenz [164](#)
- DoNothingModifier
 - Komponente [406](#)
- Dos96HebToAscii
 - Komponente [273](#)
- DownloadFileToDataHolder
 - Komponente [316](#)
- dp:as_XML
 - Datenprozessor-Funktion [109](#)
- dp:get_id
 - Datenprozessor-Funktion [109](#)
- dp:lookup
 - Datenprozessor-Funktion [109](#)
- dp:output
 - Datenprozessor-Funktion [109](#)
- DumpValues
 - Komponente [317](#)
- DynamicTable
 - Komponente [273](#)
- dynamisch
 - search [252](#)
- Dynamisch
 - Offset [250](#)

E

- EbcdicToAscii
 - Komponente [273](#)
- EDI
 - Delimiter für das Parsen [187](#)
- Editor zur Konfiguration von Tabellen
 - PdfToTxt_4_ [174](#)
- Editor, IntelliScript
 - Beschreibung [147](#)
- Editoren
 - IntelliScript [75](#)
- Eigenschaft „direction“
 - von Ankern [209](#)
- Eigenschaft „disabled“
 - Auswählen im Menü [81](#)
- Eigenschaft „marking“
 - von Ankern [209](#), [397](#)
- Eigenschaft „optional“
 - Auswählen im Menü [81](#)
 - Einstellung [411](#)
 - Fehlerbehandlung [410](#)
- Eigenschaft „phase“
 - von Ankern [209](#)
- Eigenschaften
 - Benutzerdefinierte Skriptkomponenten [445](#)
 - in IntelliScript [77](#)
 - Skriptkomponenten, Beschreibung [143](#)
 - Transformer [263](#)
 - von Aktionen [303](#), [414](#)
 - von Ankern [209](#)
 - von Mappern [360](#)
 - von Serializern [345](#)
 - von Streamern [394](#)
- Eigenschaften, ausgeblendete
 - anzeigen [143](#)
- Eigenschaften, einfach
 - Beschreibung [143](#)
- Eigenschaften, erweiterte
 - Beschreibung [143](#)
- Eigenschaften, von Komponenten
 - Werte, Beschreibung [144](#)
- einfache Eigenschaften
 - Beschreibung [143](#)
- Einfachtyp
 - Schemaobjekt [129](#)
- Einfachtyp, Ansicht
 - Schemaobjekt [129](#)
- eingeschlossen
 - Gruppe [228](#)
- Einstellungen für die Codierung
 - Datenprozessorumwandlung [29](#)
- Eintrittspunkt, Skript
 - Beschreibung [145](#)
- Einzelinstanz
 - Datenbehälter [202](#)
- Elemente
 - Datenbehälter [192](#)
- elementFormDefault
 - Schemaobjekt [131](#)
- EmbeddedMapper
 - Komponente [364](#)
- EmbeddedParser
 - Komponente [227](#)
- EmbeddedSerializer
 - Komponente [352](#)
- EmbeddedStructure
 - Komponente [256](#)

- EnclosedGroup
 - Komponente [228](#)
- EnclosingDelimiters
 - Komponente [188](#)
- EncodeAsUrl
 - Komponente [274](#)
- Encoder
 - Komponente [275](#)
- EnsureCondition
 - Komponente [318](#)
- Entwicklungszeit, Ereignisprotokoll
 - Beschreibung und Ort [37](#)
- Ereignisansicht
 - Datenprozessorumwandlung [36](#)
- Ereignisbenachrichtigungen
 - StructureDefinition [247](#)
- Ereignisprotokoll
 - anzeigen [38](#)
 - benutzerdefinierte Ereignisse [303](#)
- Ereignisprotokoll, Entwicklungszeit
 - Beschreibung und Ort [37](#)
- Ereignisse
 - Behandlung [409](#)
 - Datenprozessorumwandlung [35](#)
- Ereignistypen
 - Datenprozessorumwandlung [35](#)
- erweiterte Eigenschaften
 - Beschreibung [143](#)
- example_source, Eigenschaft
 - Mapper [361](#)
 - Serializer [346](#)
- Excel
 - als XML parsen [164](#), [165](#)
 - aus XML erzeugen [172](#)
- ExcelToDataXml
 - Komponente [164](#)
- ExcelToXml
 - Komponente [165](#)
- Excludeltems
 - Komponente [322](#)
- ExpandFrameSet
 - Komponente [167](#)
- ExternalJavaPreProcessor
 - Komponente [167](#)
- ExtractRecord
 - Komponente [230](#)

F

- Farben
 - Hervorhebung im Beispiel-Quelldokument [146](#)
- fehlende Daten
 - Fehlerbehandlung [410](#)
- fehlender Text
 - mit optionaler Gruppe suchen [236](#)
- Fehler
 - Auswirkungen auf Elternkomponente [410](#)
 - Fehlerbehandlung [410](#)
 - Verarbeitung von Validierungsfehlern [147](#)
- Fehlerbehandlung
 - Variablen für [199](#)
- Fehlschlag - Ereignis
 - Datenprozessorumwandlung [35](#)
- Fehlschläge
 - Behandlung [409](#), [410](#)
- Festwert, Eigenschaft
 - Schemaobjekt [127](#)

- FileSearch
 - Komponente [159](#)
- FindReplaceAnchor
 - Komponente [231](#)
- Footer-Segment
 - Streamer [388](#)
- Format
 - Präprozessoren [190](#)
- FormatNumber
 - Komponente [276](#)
- Formulare
 - PDF verarbeiten [168](#)
- Frameset
 - HTML parsen [167](#)
- FromFloat
 - Komponente [277](#)
- FromInteger
 - Komponente [278](#)
- FromPackDecimal
 - Komponente [278](#)
- FromSignedDecimal
 - Komponente [279](#)

G

- Gemischter Inhalt
 - Datenbehälter in [195](#)
 - im Schema [193](#)
 - zuordnen [207](#)
- Generiertes Präfix
 - Ändern für den Namespace [126](#)
- gepackte Dezimalzahlen
 - Zahlen [278](#)
- Gitter
 - XMap [86](#)
- globale Komponente
 - definieren [142](#)
- globale Komponenten
 - Definieren [79](#)
- große Eingaben aufteilen
 - Streamer [387](#)
- Group
 - Komponente [234](#)
- Group-Anweisungen
 - XMap [89](#)
- GroupMapping
 - Komponente [365](#)
- GroupSerializer
 - Komponente [353](#)
- Gruppe
 - Aktionen ausführen an [234](#)
 - Wiederholungsgruppe [239](#)

H

- Header-Segment
 - Streamer [388](#)
- Hebräisch
 - Codepage-Umwandlung [280](#)
- hebrewBidi
 - Komponente [280](#)
- HebrewDosToWindows
 - Komponente [280](#)
- HebrewEBCDICOldCodeToWindows
 - Komponente [280](#)

- hebUniToAscii
 - Komponente [280](#)
- hebUtf8ToAscii
 - Komponente [280](#)
- Hervorhebung im Beispiel-Quelldokument
 - Beschreibung [146](#)
- Hex-Quelle-Ansicht, Datenprozessor
 - Beschreibung [24](#)
- Hexadezimale Quellansicht
 - Beschreibung [147](#)
- Hierarchie auswählen
 - Datenprozessor-Umwandlung [65](#)
 - Datenprozessorumwandlung [71](#)
- HIPAA
 - Validierung [167](#)
- HIPAAValidator
 - Komponente [167](#)
- HL7
 - Komponente [188](#)
- HTML
 - Entitäten umwandeln [280](#)
 - Tags entfernen [289](#)
- HtmlEntitiesToASCII
 - Komponente [280](#)
- HtmlFormat
 - Komponente [182](#)
- HtmlProcessor
 - Komponente [190](#), [281](#)

I

- Indexierung
 - Beispiel [371](#)
 - Mehrfachinstanz-Datenbehälter [203](#)
- Inhaltsanker
 - Hervorhebung im Beispiel-Quelldokument [146](#)
- initialization
 - Variablen [201](#)
- InjectFP
 - Komponente [281](#)
- InjectString
 - Komponente [282](#)
- InlineTable
 - Komponente [283](#)
- InputPort
 - Komponente [160](#)
- Intelli-Modus
 - Beschreibung [147](#)
- IntelliScript
 - Anker definieren in [209](#)
 - Bearbeiten [77](#)
 - Einschränkungen bei Namen [80](#)
 - verwendete Symbole [80](#)
- IntelliScript-Editor
 - Beschreibung [147](#)
 - Komponenten und Eigenschaften [77](#)
- IntelliScript-Hilfe - Ansicht
 - Beschreibung [147](#)
- Iterationen
 - RepeatingGroup-Anker [239](#)

J

- Java
 - Benutzerdefinierte Komponente entwickeln [445](#)

- JavaScript
 - Erweiterungen [320](#)
 - Syntaxreferenz [319](#)
- JavaTransformer
 - Komponente [283](#)

K

- key
 - Eigenschaften von [381](#)
- Kombinationen
 - von Listen [311](#)
- komplexe Segmente
 - Streamer [388](#)
- Komplextyp
 - Erweiterte Eigenschaften [130](#)
- Komponente, global
 - definieren [142](#)
- Komponente, lokal
 - definieren [143](#)
- Komponente, Skript
 - Benutzerdefinierten Java-Code entwickeln [445](#)
 - Beschreibung [141](#)
- Komponente, Start im Skript
 - Beschreibung [145](#)
- Komponenten
 - in IntelliJScript [77](#)
- Komponenten, benutzerdefiniertes Skript
 - Beschreibung [444](#)
 - Eigenschaften [445](#)
- Komponenten, Skript
 - Beschreibung [141](#)
 - Eigenschaften, Beschreibung [143](#)
 - Namen [142](#)
- Komponentenansicht
 - Datenprozessorumwandlung [24](#)
- Komponenteneigenschaften
 - Werte, Beschreibung [144](#)

L

- Laufzeit-Ereignisprotokoll
 - Datenprozessorumwandlung [38](#)
- LearnByExample
 - Komponente [249](#)
- Leerräume reduzieren, Eigenschaft
 - Schemaobjekt [129](#)
- Leistungsprobleme
 - VarLastFailure-Variable [199](#)
- LengthEquals
 - Komponente [418](#)
- Liste
 - Validierungsregelement [437](#)
- Listen
 - combining [311](#)
 - erstellen [313](#)
 - Mehrfachinstanz-Datenbehälter [202](#)
 - sortieren [334](#)
 - von Variablen [202](#)
- Listen-Datentypen
 - Attribute [203](#)
- Listentypen
 - zuordnen [216](#)
- LocalFile
 - Komponente [160](#)

- LocalFile-Beispielquelle
 - Beschreibung [145](#)
- Locator
 - Komponente [383](#)
- LocatorByKey
 - Komponente [384](#)
- LocatorByOccurrence
 - Komponente [385](#)
- lokale Komponente
 - definieren [143](#)
- Lokatoren
 - Eigenschaften von [381](#)
- Lookup
 - Validierungsregelement [437](#)
- LookupTransformer
 - Komponente [284](#)

M

- Map
 - Komponente [88](#), [322](#)
- mapper
 - Validierung [197](#)
- Mapper
 - Beschreibung [23](#)
 - Eigenschaften von [360](#)
 - erstellen [358](#)
 - In Parser ausführen [326](#)
 - Komponente [361](#)
 - mit Indexierung [371](#)
 - zweiten abrufen [364](#)
- Mapper-Anker
 - Eigenschaften von [360](#)
 - Referenz [363](#)
- Mapping-Anweisungen
 - Ausschneiden und Einfügen [104](#)
 - XMap-Editor [86](#)
- Mapplet
 - Als Zweites ausführen [327](#), [329](#)
 - Referenz [28](#)
- Marker
 - in Streamern [390](#)
 - Komponente [237](#)
- Marker-Anker
 - Hervorhebung im Beispiel-Quelldokument [146](#)
- MarkerStreamer
 - Komponente [397](#)
- mathematische
 - Berechnungen [309](#)
- Maximale Länge
 - Schemaobjekt [127](#)
- Maximalvorkommen
 - Schemaobjekt [127](#)
- MaxLength
 - Komponente [419](#)
- MaxNumber
 - Komponente [420](#)
- Mehrfachinstanz-Datenbehälter
 - Anker zuordnen [207](#)
 - combining [311](#)
 - Indexierung [369](#)
 - Listen erstellen in [313](#)
 - sortieren [334](#)
- Mehrfachinstanzen
 - Datenbehälter [202](#)
 - Instanzen löschen [204](#)
 - Variablen [202](#)

- Mindestvorkommen
 - Schemaobjekt [127](#)
- Minimale Länge
 - Schemaobjekt [127](#)
- MinLength
 - Komponente [421](#)
- MinNumber
 - Komponente [422](#)
- mit Editor synchronisieren
 - Datenprozessorumwandlung [43](#)
- Mitgliedstypen
 - Schemaobjekt [129](#)
- Muster
 - Segment öffnen und schließen [388](#)
- Muster, Eigenschaft
 - Schemaobjekt [127](#)

N

- Nachrichten
 - Warnungsbenachrichtigungen [430](#)
- Name des Dienstes
 - Variablen zum Speichern [199](#)
- Namen
 - IntelliScript [80](#)
 - Skriptkomponenten [142](#)
- Namespace
 - Ändern des generierten Präfixes [126](#)
- Namespaces
 - Schemaobjekt [127](#)
- NewlineSearch
 - Komponente [250](#)
- Normalisierte relationale Ausgabe
 - Datenprozessor-Umwandlung [73](#)
- Normalisierte relationale Eingabe
 - Datenprozessor-Umwandlung [68](#)
- NormalizeClosingTags
 - Komponente [286](#)
- Notification-Komponente [431](#)
- NotificationGroup
 - Komponente [432](#)
- NotificationHandler-Komponente [432](#)
- NotifyFailure-Komponente [433](#)
- Nullwertfähig, Eigenschaft
 - Schemaobjekt [127](#)
- NumberEquals
 - Komponente [423](#)

O

- Offset
 - dynamisch definiert [250](#)
- OffsetSearch
 - Komponente [250](#)
- Option
 - Komponente [95](#)
- Option-Anweisung
 - XMap-Editor [87](#)
- Option-Anweisungen
 - XMap-Editor [93](#)
- Optionaler Fehlschlag
 - Auswirkungen auf Elternkomponente [410](#)
- Optionaler Fehlschlag - Ereignis
 - Datenprozessorumwandlung [35](#)
- OutputDataHolder
 - Komponente [339](#)

- OutputFile
 - Komponente [339](#)
- OutputPort
 - Komponente [161](#)

P

- parameters
 - an Umwandlung übergeben [201](#)
- Parser
 - Beschreibung [23](#)
 - für COBOL-Daten [54](#)
 - Komponente [151](#)
 - Skriptkomponenten [150](#)
 - zweiten abrufen [227](#), [352](#)
 - zweiten ausführen [328](#)
- PatternSearch
 - Komponente [250](#)
- PDF
 - Verarbeiten von PDF-Formularen [168](#)
- PDF-Ausgabe
 - XmlToDocument postprocessor_372 [170](#)
 - XmlToDocument postprocessor_45 [171](#)
 - XmlToDocument-Postprozessor [170](#), [171](#)
- PDF-Dateien
 - PdfToTxt_4-Prozessor verwenden [174](#)
- PDF-Umwandlung
 - konfigurieren [175](#)
- PDF-Unterstützung
 - PDF-Dateien umwandeln [168](#), [169](#)
- PdfFormToXml_1_00
 - Komponente [168](#)
- PdfToTxt_3_02
 - Komponente [168](#)
- PdfToTxt_4_
 - Komponente [169](#)
 - verwenden [174](#)
- Pfad
 - relativ auflösen [264](#)
- Phase
 - der Ankersuche [211](#)
- Phasen
 - geschachtelt [211](#)
- Pivotierte relationale Ausgabe
 - Datenprozessor-Umwandlung [73](#)
- Pivotierte relationale Eingabe
 - Datenprozessor-Umwandlung [69](#)
- Plattformunabhängigkeit
 - Parser [150](#)
- Ports für Dienstparameter
 - Datenprozessorumwandlung [26](#)
- Positional
 - Komponente [188](#)
- Positionen
 - im Quelldokument markieren [237](#)
- Postprozessor
 - XmlToDocument [170](#), [171](#)
 - XmlToDocument_372 [170](#)
 - XmlToDocument_45 [171](#)
- PostScript
 - Komponente [189](#)
- PowerpointToTextML
 - Komponente [169](#)
- Prädikatanweisung
 - XPath-Ausdrücke [105](#)
- Präprozessoren
 - definieren [163](#)

- Präprozessoren (*Fortsetzung*)
 - Dokument [163](#)
 - Format [190](#)
- ProcessByTransformers
 - Komponente [170](#)
- ProcessorPipeline
 - Komponente [170](#)
- Protokoll
 - Definition [37](#)
- Protokoll, Entwicklungszeit-Ereignis
 - Beschreibung und Ort [37](#)
- Protokolle
 - schreiben in [409](#)
 - Schreiben in [430](#)
- Prozessoren
 - benutzerdefiniert, Java [167](#)
 - Dokument [163](#)
 - Referenz [164](#)
 - Transformer verwenden als [262](#)
- Pufferausgabeport
 - Datenprozessorumwandlung [26](#)
- Puffereingabeport
 - Datenprozessorumwandlung [25](#)

Q

- Quelldokument, Hervorhebung
 - Beschreibung [146](#)

R

- RecordStructure
 - Komponente [257](#)
- RecordStructureLocal
 - Komponente [258](#)
- Referenz
 - Anker [217](#)
 - delimiters [185](#)
 - Dokumentprozessoren [164](#)
 - Formate [179](#)
 - Formatpräprozessor [190](#)
 - Indexierung [381](#)
 - Mapper [361](#)
 - Mapper-Anker [363](#)
 - parsers [151](#)
 - Serialisierungsanker [347](#)
- Referenzen-Ansicht
 - Datenprozessorumwandlung [28](#)
- Referenzpunkte
 - Marker-Anker [237](#)
 - Suchbereich [213](#)
 - um Anker [209](#), [397](#)
- Regel
 - Validierungsregelelement [438](#)
- regex
 - reguläre Ausdrücke [287](#)
- reguläre Ausdrücke
 - Syntax [287](#)
- RegularExpression
 - Komponente [286](#)
- RemoveMarginSpace
 - Komponente [288](#)
- RemoveRtfFormatting
 - Komponente [289](#)
- RemoveTags
 - Komponente [289](#)

- Repeating Group-Anweisung
 - Run XMap-Anweisung
 - XMap-Editor [87](#)
 - RunMapplet-Anweisung
 - XMap-Editor [87](#)
- RepeatingGroup
 - Komponente [239](#)
- RepeatingGroupMapping
 - Komponente [366](#)
- RepeatingGroupSerializer
 - Komponente [354](#)
- Replace
 - Komponente [290](#)
- Repository-Ansicht
 - Datenprozessorumwandlung [24](#)
- ResetVisitedPages
 - Komponente [325](#)
- Resize
 - Komponente [291](#)
- ResultFile
 - Komponente [339](#)
- ReverseTransformer
 - Komponente [292](#)
- Router-Anweisung
 - XMap-Editor [87](#)
- Router-Anweisungen
 - XMap-Editor [93](#)
- RTF
 - Komponente [189](#)
- RtfFormat
 - Komponente [183](#)
- RtfProcessor
 - Komponente [191](#), [292](#)
- RtfToASCII
 - Komponente [292](#)
- RtfToTextML
 - Komponente [170](#)
- rückgängig machen
 - nach Fehlschlag [411](#)
- Run XMap-Anweisung
 - Mapping der Anweisung [97](#)
- RunMapper
 - Komponente [326](#)
- RunMapplet
 - Komponente [327](#)
- RunMapplet-Anweisung
 - Mapping-Anweisung [98](#)
- RunParser
 - Komponente [328](#)
- RunPCWebService
 - Komponente [329](#)
- RunSerializer
 - Komponente [330](#)
- RunXMap
 - Komponente [331](#)

S

- Schema
 - Avro
 - Schema [49](#)
 - Codierung [193](#)
 - für COBOL-Daten [54](#)
 - JSON
 - Schema [57](#)

Schema (Fortsetzung)

- Parquet
 - Definition [60](#)
 - Schema [60](#)
- XML
 - Schema [62](#)
- Schema, Ansicht
 - Einfachtyp, erweiterte Eigenschaften [129](#)
 - Schemaobjekt [126](#)
- Schemadateien
 - bearbeiten [135](#)
 - Entfernen aus Schemaobjekten [124](#)
 - Festlegen eines Standardeditors [134](#)
 - Hinzufügen zu Schemaobjekten [124](#)
- Schemaobjekt
 - Elemente, erweiterte Eigenschaften [128](#)
 - Abstrakt, Eigenschaft [127](#)
 - attributeFormDefault [131](#)
 - Attributeigenschaften [131](#)
 - Bearbeiten einer Schemadatei [133](#)
 - Block, Eigenschaft [128](#)
 - Einfachtyp [129](#)
 - Elementeigenschaften [127](#)
 - elementFormDefault [131](#)
 - Festlegen eines Standardeditors [134](#)
 - Importieren [132](#)
 - Komplexe Elemente [130](#)
 - Komplexe Elemente, erweiterte Eigenschaften [130](#)
 - Namespaces [127](#)
 - Schema, Ansicht [126](#)
 - Schemadateien [124](#)
 - Speicherort [131](#)
 - Substitutionsgruppe [128](#)
 - Synchronisation [133](#)
 - Übernommen aus, Eigenschaft [130](#)
 - Übernommen von, Eigenschaft [130](#)
 - Übersicht [124](#)
 - Übersicht (Ansicht) [125](#)
- Schemata
 - Datenprozessorumwandlung [28](#)
 - eingefügte Schemata [193](#)
 - IntelliScript-Darstellung [195](#)
 - nicht unterstützte Funktionen [193](#)
 - Validierung [196](#)
 - XSD-Dateien [192](#)
- Schleife
 - RepeatingGroup-Anker [239](#)
- Schlüssel
 - Komponente [381](#)
- Schwerwiegender Fehler - Ereignis
 - Datenprozessorumwandlung [35](#)
- search
 - dynamisch definierter Suchstring [252](#)
- Search
 - Ankerrichtung [209](#)
- Segmente
 - in Streamer verarbeiten [388](#)
- SegmentSearch
 - Komponente [251](#)
- SequenceStructure
 - Komponente [259](#)
- SequenceStructureLocal
 - Komponente [259](#)
- Serialisierung
 - Modus [341](#)
 - Transformer verwenden in [263](#)
- Serialisierungsanker
 - Eigenschaften von [345](#)

Serialisierungsanker (Fortsetzung)

- Operationsreihenfolge [344](#)
- Referenz [347](#)
- serializer
 - Validierung [197](#)
- Serializer
 - automatische Erzeugung steuern [340](#)
 - Eigenschaften von [345](#)
 - Fehlerbehebung, automatisch erzeugt [342](#)
 - für COBOL-Daten [54](#)
 - In Parser ausführen [330](#)
 - Komponente [346](#)
- ServiceLocation
 - Variable [198](#)
- SetValue
 - Komponente [333](#)
- SGML
 - Komponente [189](#)
- SimpleSegment
 - Komponente [399](#)
- SimpleXmlSegment
 - Komponente [400](#)
- Skript
 - Beispiele [148](#)
 - Beschreibung [140](#)
 - Editor, IntelliScript [147](#)
 - Importieren eines Beispielskripts [149](#)
 - In der Datenprozessorumwandlung erstellen [41, 75](#)
 - Struktur [141](#)
- Skript-Hilfe-Ansicht
 - Datenprozessorumwandlung [24](#)
- Skriptkomponente
 - Benutzerdefinierten Java-Code entwickeln [445](#)
 - Beschreibung [141](#)
- Skriptkomponenten
 - Beschreibung [141](#)
 - Eigenschaften, Beschreibung [143](#)
 - Namen [142](#)
- Skriptkomponenten, benutzerdefinierte
 - Beschreibung [444](#)
 - Eigenschaften [445](#)
- Skriptmodus
 - Beschreibung [147](#)
- Sonderzeichen
 - In IntelliScript eingeben [79](#)
- sortieren
 - Mehrfachinstanz-Datenbehälter [334](#)
- Sortieren
 - Komponente [334](#)
- source
 - Eigenschaft [374, 375](#)
- SpaceDelimited
 - Komponente [189](#)
- Sprache der Validierungsregeln
 - VRL (Validation Rule Language) [335](#)
- Standardtransformer
 - in Format [262](#)
- Startkomponente
 - Datenprozessorumwandlung [27](#)
- Startkomponente, Skript
 - Beschreibung [145](#)
- Streamer
 - Beschreibung [23](#)
 - Eigenschaften von [394](#)
 - erstellen [391](#)
 - Footer-Segment [388](#)
 - große Eingaben aufteilen [387](#)
 - Header-Segment [388](#)

- Streamer (*Fortsetzung*)
 - JsonStreamer [396](#)
 - komplexe Segmente [388](#)
 - Komponente [401](#)
 - Muster zum Öffnen und Schließen von Segmenten [388](#)
 - output [390](#)
 - Verketten des Headers [389](#)
 - wiederholendes Segment [388](#)
- StreamerVariable
 - Komponente [402](#)
- Strings
 - Verketten [306](#), [308](#)
- StringSerializer
 - Komponente [343](#)
- StructureDefinition
 - Extrahieren von Datensätzen [230](#)
 - Komponente [244](#)
- strukturiertes Parsing
 - StructureDefinition [244](#)
- Substitutionsgruppe
 - Schemaobjekt [128](#)
- SubString
 - Komponente [293](#)
- Suchbereich
 - anpassen [213](#)
 - nach Ankern [212](#)
- Suche
 - Komponenten [215](#), [248](#)
- Suchkriterien
 - nach Ankern [212](#)
- Symbole
 - IntelliScript [80](#)
- System
 - Variablen [198](#)
- Systemzeit
 - Variable [199](#)

T

- TabDelimited
 - Komponente [190](#)
- Tabellen
 - PDF verarbeiten [174](#)
- target
 - Eigenschaft [374](#), [378](#)
- Testen einer Bibliothek
 - Datenprozessorumwandlung [121](#)
- Text
 - Komponente [161](#)
- Text ersetzen
 - in Quelldokument [231](#)
- Text rechts nach links
 - reversing [267](#), [280](#)
- Text-Beispielquelle
 - Beschreibung [145](#)
- Text-Streamer
 - Funktionsweise [388](#)
- TextFormat
 - Komponente [183](#)
- TextML
 - XML-Schema [173](#)
- TextSearch
 - Komponente [251](#)
- TGP-Datei
 - für benutzerdefinierte Komponenten [447](#)
- ToFloat
 - Komponente [293](#)

- ToInteger
 - Komponente [294](#)
- ToPackDecimal
 - Komponente [295](#)
- Trace
 - Validierungsregelement [438](#)
- TransformationStartTime
 - Komponente [295](#)
- TransformByParser
 - Komponente [296](#)
- TransformByProcessor
 - Komponente [297](#)
- TransformByService
 - Komponente [298](#)
- Transformer
 - als Dokumentpräprozessoren [262](#)
 - Beschreibung [23](#)
 - Eigenschaften von [263](#)
 - Folgen von [262](#)
 - in Ankern verwenden [261](#)
 - in Serialisierung [263](#)
 - Standard [262](#)
- TransformerPipeline
 - Komponente [299](#)
- transformers
 - als Dokumentprozessor verwenden [170](#)
 - benutzerdefiniert, Java [283](#)
 - definieren [261](#)
 - verglichen mit Aktionen [302](#)
- Typen
 - XSI [196](#)
- TypeSearch
 - Komponente [252](#)

U

- Übereinstimmung von Muster
 - reguläre Ausdrücke [287](#)
- Übernommen aus, Eigenschaft
 - Schemaobjekt [130](#)
- Übernommen von, Eigenschaft
 - Schemaobjekt [130](#)
- Uhrzeit
 - System [199](#)
- Ungültige Daten
 - Erkennen [409](#)
- UNIX
 - Parser entwickeln für [150](#)
- URL
 - relativ zu absolut [264](#)
- URL-Beispielquelle
 - Beschreibung [145](#)

V

- ValidateByExpression
 - Komponente [424](#)
- ValidateByPattern
 - Komponente [425](#)
- ValidateByTransformer
 - Komponente [426](#)
- ValidateByType
 - Komponente [427](#)
- ValidateDate-Komponente [428](#)
- ValidateValue
 - Komponente [335](#)

- Validatoren
 - Eingabedaten [413](#)
- ValidatorPipeline
 - Komponente [429](#)
- Validierung
 - XML [196](#)
 - XML-Eingabe [197](#)
 - XML-Parserausgabe [197](#)
- Validierungsfehler
 - Behandlung [147](#)
- Validierungsregel
 - Assert-Element [436](#)
 - Element [436–439](#)
 - Listenelement [437](#)
 - Lookup-Element [437](#)
 - Regelement [438](#)
 - Trace-Element [438](#)
 - Variablenelement [439](#)
 - XPath-Editor [439](#)
- Validierungsregelemente
 - XPath-Editor [439](#)
- Validierungsregeln
 - Data Transformation-Dienst importieren [443](#)
 - Definition [435](#)
 - Editor [435, 442](#)
 - Im externen Editor bearbeiten [442](#)
- Validierungsregelobjekt
 - In der Datenprozessorumwandlung erstellen [42](#)
- VarCurrentPost
 - Variable [199](#)
- VarCurrentURL
 - Variable [199](#)
- Variable
 - Komponente [202](#)
 - Validierungsregelement [439](#)
- Variablen
 - Anker zuordnen [201, 207](#)
 - benutzerdefiniert [198](#)
 - Datenbehälter [192](#)
 - in Aktionen verwenden [201](#)
 - in Streamern [390](#)
 - initialization [201](#)
 - Listen [202](#)
 - System [198](#)
 - XMap-Editor [111](#)
- VarLastFailure
 - Variable [199](#)
- VarLinkURL
 - Variable [198](#)
- VarRequestedURL
 - Variable [199](#)
- VarServiceInfo
 - Variable [199](#)
- VarSystem
 - Systemzeit [199](#)
- Verkettung
 - Strings [306](#)
- vorzeichenbehaftete Zahl
 - Zahlen [279](#)
- VRL (Validation Rule Language)
 - Sprache der Validierungsregeln [335](#)

W

- Warnung - Ereignis
 - Datenprozessorumwandlung [35](#)

- WellFormedModifier
 - Komponente [406](#)
- Werte, Komponenteneigenschaften
 - Beschreibung [144](#)
- wiederholendes Segment
 - Streamer [388](#)
- Wiederholungsgruppe
 - Hervorhebung im Beispiel-Quelldokument [146](#)
- Wiederholungsgruppe-Anker
 - alle Iterationen hervorheben [146](#)
- Word
 - als XML parsen [170](#)
- WordToXml
 - Komponente [170](#)
- WriteSegment
 - Komponente [407](#)
- WriteValue
 - Komponente [336](#)

X

- Xerces
 - XML-Validierung [197](#)
- XMap
 - Beschreibung [84](#)
 - Group-Anweisungen [89](#)
 - In Mapper ausführen [331](#)
 - In Parser ausführen [331](#)
 - In Serializer ausführen [331](#)
 - Router-Anweisungen [93](#)
 - Schema, Bereich [85](#)
 - Typen von Anweisungen [87](#)
 - Variablen [111](#)
- XMap-Editor
 - Gitter [86](#)
- XMap-Objekt
 - In der Datenprozessorumwandlung erstellen [41](#)
- XML
 - Anker zuordnen [206](#)
 - leere Tags hinzufügen [264](#)
 - Schemata [192](#)
 - Validierung [197](#)
 - XSLT-Umwandlung [300](#)
- XML-Attribute
 - Datenbehälter [192](#)
- XML-Elemente
 - Datenbehälter [192](#)
- XML-Streamer
 - Funktionsweise [392](#)
- XMLFormat
 - Komponente [184](#)
- XMLLookupTable
 - Komponente [299](#)
- XmlSegment
 - Komponente [402](#)
- XmlStreamer
 - Komponente [403](#)
- XmlToDocument
 - Komponente [170, 171](#)
- XmlToDocument_372
 - Komponente [170](#)
- XmlToDocument_45
 - Komponente [171](#)
- XmlToExcel
 - Komponente [172](#)
- XmlToXlsx
 - Komponente [172](#)

- XPath
 - veränderte Schreibweise [195](#)
- XPath-Ausdrücke
 - XMap [104](#)
- XPath-Ausdruckseditor
 - Datenprozessor-Umwandlung [108](#)
- XPath-Editor
 - Validierungsregelemente [439](#)
- XSD
 - Editoren [192](#)
 - nicht unterstützte Schema-Funktionen [193](#)
 - Schemacodierung [193](#)
- XSD-Dateien
 - Schemata [192](#)
- XSI-Typen
 - Datenbehälter zuordnen [196](#)
- XSLT
 - Umwandlungen ausführen [337](#)
- XSLTMap
 - Komponente [337](#)
- XSLTTransformer
 - Komponente [300](#)

Z

- Zahlen
 - Formatierung [276](#)
- Zeichen
 - Sonderzeichen in IntelliScript [79](#)
- Zeichen eingeben
 - nicht auf der Tastatur vorhandene [144](#)
- Zeichen, die nicht auf der Tastatur vorhanden sind
 - eingeben [144](#)
- Zeiten
 - Format von [270](#)
- zuweisen
 - Wert zu Ausgabe [333](#)
- zweiter Mapper
 - EmbeddedMapper-Anker [364](#)
- zweiter Parser
 - EmbeddedParser-Anker [227, 352](#)