



Informatica®  
10.2.2

# Developer 변환 가이드

## Informatica Developer 변환 가이드

10.2.2

2019년 2월

© 저작권 Informatica LLC 2009, 2019

이 소프트웨어와 설명서는 사용 및 공개에 대한 제한 사항이 포함되어 있는 별도의 사용권 계약에 따라서만 제공됩니다. 본 문서의 어떤 부분도 Informatica LLC의 사전 통지 없이 어떠한 형태나 수단(전자적, 사진 복사, 녹음 등)으로 복제되거나 전송될 수 없습니다.

미국 정부 권한. 미국 정부 고객에게 제공되는 프로그램, 소프트웨어, 데이터베이스, 관련 문서 및 기술 데이터는 해당하는 연방 입수 규정 및 기관별 보안 규정에 따라 "상용 컴퓨터 소프트웨어" 또는 "상용 기술 데이터"입니다. 따라서 사용, 복제, 공개, 수정 및 조정은 해당하는 정부 계약에 규정된 제한 사항 및 라이선스 조건을 따르며, 정부 계약 조건에 의해 적용 가능한 한도 내에서, FAR 52.227-19, 상용 소프트웨어 라이선스에 규정된 추가 권한이 적용됩니다.

Informatica, Informatica 로고 및 PowerCenter는 미국과 전 세계 여러 관할 국가에서 Informatica LLC의 상표 또는 등록 상표입니다. Informatica 상표의 현재 목록은 <https://www.informatica.com/trademarks.html> 웹에서 확인할 수 있습니다. 다른 회사 및 제품명은 해당 소유자의 상표 또는 등록 상표일 수 있습니다.

이 소프트웨어 및/또는 설명서의 일부에는 타사의 저작권이 적용될 수 있습니다. 필요한 타사 고지 사항은 제품에 포함되어 있습니다.

이 설명서의 정보는 예고 없이 변경될 수 있습니다. 이 문서에서 문제가 발견되는 경우 [infa\\_documentation@informatica.com](mailto:infa_documentation@informatica.com)으로 보고해 주십시오.

Informatica 제품은 제품이 제공될 당시의 계약 조건에 따라 보증됩니다. Informatica는 상품성과 특정 목적에의 적합성에 대한 보증 그리고 비침해에 대한 보증 또는 조건을 포함하여 어떠한 종류의 명시적이거나 묵시적인 보증 없이 이 문서의 정보를 "있는 그대로" 제공합니다.

발행 날짜: 2019-06-06

# 목차

<b>서문</b>	<b>31</b>
Informatica 리소스	31
Informatica 네트워크	31
Informatica 기술 자료	31
Informatica 설명서	32
Informatica Product Availability Matrix	32
Informatica Velocity	32
Informatica Marketplace	32
Informatica 글로벌 고객 지원 센터	32
<b>장 1: 변환 소개</b>	<b>33</b>
변환 소개 개요	33
활성 변환	33
수동 변환	34
연결되지 않은 변환	34
다중 전략 변환	34
변환 설명	35
변환 - 원시 및 비원시 환경	37
변환 데이터 유형 처리	39
10진수 데이터 유형	40
시간대가 포함된 타임스탬프	41
현지 시간대가 포함된 타임스탬프	42
변환 개발	42
다중 그룹 변환	42
다중 그룹 변환에 대한 규칙 및 지침	43
변환의 식	43
식 편집기	44
식의 포트 이름	44
포트에 식 추가	44
식의 설명	45
식 유효성 검사	45
식 테스트	45
데이터 유형 변환	46
로컬 변수	47
임시로 데이터 저장 및 복잡한 식 간소화	47
행의 값 저장	48
저장 프로시저의 값 캡처	48
변수 포트 구성 지침	48
변수 초기화	49

포트의 기본값.....	49
사용자 정의 기본값.....	50
사용자 정의 기본 입력값.....	51
기본값 유효성 검사.....	53
사용자 정의 기본 출력 값.....	53
기본값에 대한 일반 규칙.....	55
기본값 유효성 검사.....	56
추적 수준.....	56
재사용 가능한 변환.....	56
재사용 가능한 변환 인스턴스 및 상속된 변경 내용.....	57
재사용 가능한 변환 편집.....	57
재사용 가능 변환의 편집기 보기.....	57
재사용 불가능 변환.....	58
재사용 불가능 변환의 편집기 보기.....	58
변환 작성.....	58
<b>장 2: 변환 포트.....</b>	<b>59</b>
변환 포트 개요.....	59
포트 생성.....	59
포트 구성.....	60
포트 연결.....	60
일대일 링크.....	60
일대다 링크.....	60
수동으로 포트 연결.....	61
자동으로 포트 연결.....	61
포트 연결에 대한 규칙 및 지침.....	63
포트 특성 전달.....	63
종속성 유형.....	64
링크 경로 종속성.....	64
암시적 종속성.....	64
변환별 전달된 포트 특성.....	65
Excel에서 포트 복사.....	66
Excel에서 변환 편집.....	67
Developer tool에 메타데이터 복사.....	67
예: Excel에서 변환 편집.....	68
Excel에서 복사에 대한 규칙 및 지침.....	68
<b>장 3: 변환 캐시.....</b>	<b>69</b>
변환 캐시 개요.....	69
캐시 유형.....	70
캐시 파일.....	70
캐시 파일 디렉터리.....	71



캐시 크기.....	71
자동 캐시 크기.....	71
특정 캐시 크기.....	72
데이터 통합 서비스에 의한 캐시 크기 증가.....	72
분할된 캐시의 캐시 크기.....	73
캐시 크기 최적화.....	73
1단계. 추적 수준을 자세한 정보 표시 초기화로 설정합니다..	74
2단계. 자동 캐시 모드에서 매핑 실행.....	74
3단계. 캐싱 성능 분석.....	74
4단계. 특정 캐시 크기 구성.....	74

## **장 4: 주소 유효성 검사기 변환..... 76**

주소 유효성 검사기 변환 개요.....	77
주소 참조 데이터.....	77
주소 참조 데이터 유형.....	77
모드 및 템플릿.....	79
포트 그룹 및 포트 선택.....	79
주소 유효성 검사기 변환 입력 포트 그룹.....	79
주소 유효성 검사기 변환 출력 포트 그룹.....	80
여러 인스턴스 포트.....	83
주소 유효성 검사 프로젝트.....	84
형식이 지정된 주소 및 우편 운송업체 표준.....	85
부분 주소 완료.....	86
주소 유효성 검사기 상태 포트.....	86
요소 상태 코드 정의.....	87
주소 확인 코드 출력 포트 값.....	89
요소 입력 상태 출력 포트 값.....	89
요소 관련성 출력 포트 값.....	90
요소 결과 상태 출력 포트 값.....	91
확장 요소 결과 상태 출력 포트 값.....	92
편지 점수 출력 포트 값.....	93
일치 코드 출력 포트 값.....	94
좌표 상태 출력 포트 값.....	96
주소 유효성 검사기 변환의 일반 설정.....	97
기본 설정 창의 주소 유효성 검사 속성.....	98
주소 유효성 검사 데이터 속성.....	99
주소 유효성 검사 라이선스 속성.....	99
주소 유효성 검사 엔진 속성.....	100
주소 유효성 검사 고급 속성.....	101
별칭 로컬리티.....	101
별칭 거리.....	101
대/소문자 구분 스타일.....	102

출생지.....	103
국가 유형.....	103
기본 국가.....	104
이중 주소 우선 순위.....	104
요소 약어.....	104
실행 인스턴스.....	105
유동적 범위 확장.....	105
좌표 부여 데이터 유형.....	106
글로벌 최대 필드 길이.....	107
글로벌 기본 설정 설명자.....	107
입력 형식 유형.....	108
국가가 포함된 입력 형식.....	108
행 구분 기호.....	108
일치 대체.....	109
확장된 보관 일치.....	109
일치 범위.....	110
최대 결과 수.....	110
모드.....	110
최적화 수준.....	111
출력 형식 유형.....	111
국가가 포함된 출력 형식.....	112
기본 설정 언어.....	112
기본 설정 스크립트.....	118
확장할 범위.....	119
잘못된 주소 표준화.....	119
추적 수준.....	120
인증 보고서.....	120
AMAS 보고서 필드.....	121
CASS 보고서 필드.....	121
SendRight 보고서.....	122
SERP 보고서 필드.....	123
주소 유효성 검사기 변환 구성.....	123
주소 유효성 검사기 변환에 포트 추가.....	124
사용자 정의 템플릿 작성.....	124
주소 유효성 검사기 모델 정의.....	124
인증 보고서 정의.....	125
주소 유효성 검사기 변환 비원시 환경 내.....	125
주소 유효성 검사기 변환 Blaze 엔진 내.....	126
주소 유효성 검사기 변환 Spark 엔진 내.....	126

## 장 5: 집계 변환..... 127

집계 변환 개요.....	127
---------------	-----

동적 매핑의 집계 변환.....	128
집계 변환 개발.....	128
집계 변환 포트.....	128
집계 식.....	129
집계 함수.....	130
중첩 집계 함수.....	130
집계 식의 조건절.....	131
그룹 기준 포트.....	131
그룹 기준 포트 구성.....	132
그룹 기준 매개 변수.....	132
그룹 기준 포트의 기본값.....	133
비집계 식.....	133
집계 캐시.....	133
집계 변환을 위한 정렬된 입력.....	134
정렬된 입력 조건.....	134
집계 변환에서 데이터 정렬.....	134
집계 변환 고급 속성.....	135
재사용 가능한 집계 변환 작성.....	136
재사용 불가능 집계 변환 작성.....	136
집계 변환에 대한 팁.....	137
집계 변환 문제 해결.....	137
집계 변환 - 비원시 환경.....	138
집계 변환 - Blaze 엔진.....	138
집계 변환 - Spark 엔진.....	138
집계 변환 - Databricks Spark 엔진.....	139

## 장 6: 연관 변환..... 140

연관 변환 개요.....	140
메모리 할당.....	141
연관 변환 고급 속성.....	141

## 장 7: 잘못된 레코드 예외 변환..... 143

잘못된 레코드 예외 변환 개요.....	143
잘못된 레코드 예외 출력 레코드 유형.....	144
잘못된 레코드 예외 관리 프로세스 흐름.....	144
잘못된 레코드 예외 매핑.....	145
잘못된 레코드 예외 품질 문제.....	146
휴먼 태스크.....	146
잘못된 레코드 예외 포트.....	147
잘못된 레코드 예외 변환 입력 포트.....	147
잘못된 레코드 예외 변환 출력.....	148
잘못된 레코드 예외의 구성 보기.....	148

잘못된 레코드 테이블 및 문제 테이블 생성.....	150
잘못된 레코드 예외 문제 할당.....	150
품질 문제에 포트 할당.....	151
예외 변환의 고급 속성.....	151
잘못된 레코드 예외 변환 구성.....	151
잘못된 레코드 예외 매핑 예제.....	152
잘못된 레코드 예외 맵셋.....	152
잘못된 레코드 예외 예제 입력 그룹.....	153
잘못된 레코드 예외 예 구성.....	154
잘못된 레코드 예외 매핑 출력 예제.....	155

## 장 8: 대/소문자 변환기 변환..... 158

대/소문자 변환기 변환 개요.....	158
대/소문자 전략 속성.....	158
대/소문자 변환기 전략 구성.....	159
대/소문자 변환기 변환 고급 속성.....	159
대/소문자 변환기 변환 - 비원시 환경.....	160

## 장 9: 분류자 변환..... 161

분류자 변환 개요.....	161
분류자 모델.....	162
분류자 알고리즘.....	162
분류자 변환 옵션.....	162
분류자 전략.....	163
분류자 변환의 고급 속성.....	163
분류자 전략 구성.....	163
분류자 분석 예제.....	164
분류자 매핑 작성.....	165
입력 데이터 샘플.....	166
데이터 소스 구성.....	166
분류자 변환 구성.....	166
라우터 변환 구성.....	167
데이터 대상 구성.....	168
분류자 매핑 결과.....	168
분류자 변환 - 비원시 환경.....	169

## 장 10: 비교 변환..... 170

비교 변환 개요.....	170
필드 일치 전략.....	170
Bigram.....	170
Hamming 거리 측정법.....	171
편집 거리 측정법.....	171

Jaro 거리 측정법. . . . .	172
Hamming 거리 측정법 반전. . . . .	172
ID 일치 전략. . . . .	173
비교 전략 구성. . . . .	173
비교 변환 고급 속성. . . . .	174
비교 변환 - 비원시 환경. . . . .	174
<b>장 11: 통합 변환. . . . .</b>	<b>175</b>
통합 변환 개요. . . . .	175
통합 매핑. . . . .	176
통합 변환 포트. . . . .	176
통합 변환 보기. . . . .	176
통합 변환 전략 보기. . . . .	176
통합 변환의 고급 속성. . . . .	177
캐시 파일 크기. . . . .	178
단순 전략. . . . .	179
행 기반 전략. . . . .	179
고급 전략. . . . .	180
단순 통합 함수. . . . .	180
CONSOL_AVG . . . . .	181
CONSOL_LONGEST. . . . .	181
CONSOL_MAX. . . . .	182
CONSOL_MIN. . . . .	182
CONSOL_MOSTFREQ. . . . .	183
CONSOL_MOSTFREQ_NB. . . . .	183
CONSOL_SHORTEST. . . . .	184
행 기반 통합 함수. . . . .	184
CONSOL_GETROWFIELD. . . . .	185
CONSOL_MODELEXACT. . . . .	185
CONSOL_MOSTDATA. . . . .	186
CONSOL_MOSTFILLED. . . . .	187
통합 매핑 예제. . . . .	188
입력 데이터. . . . .	188
키 생성기 변환. . . . .	188
통합 변환. . . . .	189
통합 매핑 출력. . . . .	189
통합 변환 구성. . . . .	189
통합 변환 - 비원시 환경. . . . .	189
통합 변환 - Blaze 엔진. . . . .	190
통합 변환 - Spark 엔진. . . . .	190
통합 변환 - Databricks Spark 엔진. . . . .	190

<b>장 12: 데이터 마스킹 변환</b>	<b>191</b>
데이터 마스킹 변환 개요	191
마스킹 기술	192
무작위 마스킹	192
식 마스킹	194
키 마스킹	195
대체 마스킹	197
종속 마스킹	199
토큰화 마스킹	201
마스킹 규칙	201
마스크 형식	202
소스 문자열 문자	203
결과 문자열 대체 문자	203
범위	204
블러링	204
특수 마스크 형식	205
신용 카드 번호 마스킹	206
전자 메일 주소 마스킹	206
고급 전자 메일 마스킹	206
IP 주소 마스킹	207
전화 번호 마스킹	207
사회 보장 번호 마스킹	207
URL 주소 마스킹	208
사회 보험 번호 마스킹	208
기본값 파일	209
데이터 마스킹 변환 구성	210
데이터 통합 서비스 구성	210
데이터 마스킹 변환 작성	210
포트 정의	211
각 포트에 대한 데이터 마스킹 구성	211
마스킹된 데이터 미리보기	211
데이터 마스킹 변환의 런타임 속성	211
데이터 마스킹 예제	212
Read_Customer 데이터	213
고객 데이터 마스킹 변환	213
고객 테스트 데이터 결과	214
데이터 마스킹 변환의 고급 속성	214
데이터 마스킹 변환 - 비원시 환경	215
데이터 마스킹 변환 - Blaze 엔진	215
데이터 마스킹 변환 - Spark 엔진	215

<b>장 13: 데이터 프로세서 변환</b>	<b>217</b>
데이터 프로세서 변환 개요	217
데이터 프로세서 변환 보기	218
데이터 프로세서 변환 포트	219
데이터 프로세서 변환 입력 포트	219
데이터 프로세서 변환 출력 포트	220
통과 포트	220
시작 구성 요소	221
참조	221
데이터 프로세서 변환 설정	222
문자 인코딩	222
문자 인코딩에 대한 규칙 및 지침	224
출력 설정	224
처리 설정	225
XMap 설정	226
XML 출력 구성	226
이벤트	228
이벤트 유형	228
데이터 프로세서 이벤트 보기	228
로그	229
디자인 타임 이벤트 로그	229
런타임 이벤트 로그	230
데이터 프로세서 이벤트 보기에서 이벤트 로그 보기	230
사용자 로그	230
데이터 프로세서 변환 개발	231
데이터 프로세서 변환 작성	231
스키마 개체 선택	232
빈 데이터 프로세서 변환에서 개체 작성	232
포트 작성	234
변환 테스트	235
데이터 프로세서 변환 가져오기 및 내보내기	235
데이터 프로세서 변환을 서비스로 내보내기	235
여러 Data Transformation 서비스 가져오기	236
Data Transformation 서비스 가져오기	236
데이터 프로세서 변환이 포함된 매핑을 PowerCenter로 내보내기	236
데이터 프로세서 변환 비원시 환경 내	237
데이터 프로세서 변환 Blaze 엔진 내	237
<b>장 14: 결정 변환</b>	<b>238</b>
결정 변환 개요	238
결정 변환 함수	238

결정 변환의 조건문	241
결정 변환 연산자	241
결정 변환의 NULL 처리	242
결정 전략 구성	243
결정 변환의 고급 속성	243
결정 변환 - 비원시 환경	244

## 장 15: 중복 레코드 예외 변환 245

중복 레코드 예외 변환 개요	245
중복 레코드 예외 프로세스 흐름	246
중복 레코드 예외	246
중복 레코드 예외 구성 보기	246
중복 레코드 테이블 생성	248
포트	248
중복 레코드 예외 변환 입력 포트	249
중복 레코드 예외 변환의 출력 포트	249
포트 작성	250
중복 레코드 예외 변환의 고급 속성	250
중복 레코드 예외 매핑 예제	251
중복 레코드 예외 매핑	251
일치 변환	252
중복 레코드 예외 입력 그룹	252
중복 레코드 예외 예외 구성 보기	253
표준 출력 테이블 레코드	254
클러스터 출력	255
중복 레코드 예외 변환 작성	256

## 장 16: 식 변환 257

식 변환 개요	257
식 변환 포트	258
식 테스트	259
샘플 데이터에 대한 날짜 형식 문자열	259
식 테스트	260
포트 선택기	260
포트 선택기 구성	260
선택 규칙	261
포트 선택기 작성	262
창 작업	263
창 작업 구성	264
동적 식	267
출력 포트 설정	268
동적 식 작성	269



식 변환 고급 속성.....	271
식 변환 - 비원시 환경.....	272
식 변환 - Blaze 엔진.....	272
식 변환 - Spark 엔진.....	272
식 변환 - Databricks Spark 엔진.....	272
<b>장 17: 필터 변환.....</b>	<b>274</b>
필터 변환 개요.....	274
동적 매핑의 필터 변환.....	275
필터 조건.....	275
필터 조건 매개 변수화.....	276
Null 값을 가진 행 필터링.....	277
필터 변환 고급 속성.....	277
필터 변환 성능 팁.....	278
필터 변환 - 비원시 환경.....	278
필터 변환 - Blaze 엔진.....	278
<b>장 18: 계층-관계형 변환.....</b>	<b>279</b>
계층-관계형 변환 개요.....	279
예제 - 계층-관계형 변환.....	279
출력 관계형 포트 및 개요 보기.....	281
계층-관계형 변환 포트.....	281
스키마 참조.....	282
포트 구성.....	282
계층-관계형 변환 개발.....	282
계층-관계형 변환 작성.....	282
포트 및 매핑 구성.....	283
변환 테스트.....	283
<b>장 19: Java 변환.....</b>	<b>285</b>
Java 변환 개요.....	285
재사용 가능 및 재사용 불가능 Java 변환.....	286
활성 및 수동 Java 변환.....	286
데이터 유형 변환.....	286
복합 데이터 유형 변환 - Spark 엔진.....	288
Java 변환 디자인.....	289
Java 변환 포트.....	289
그룹 및 포트 작성.....	289
기본 포트 값 설정.....	290
Java 변환의 고급 속성.....	290
PowerCenterDeveloper tool 클라이언트에 대한 클래스 경로 구성.....	292
데이터 통합 서비스에 대한 클래스 경로 구성.....	293

Java 코드 개발. . . . .	293
Java 코드 조각 작성. . . . .	295
Java 패키지 가져오기. . . . .	295
도우미 코드 정의. . . . .	296
Java 변환의 Java 속성. . . . .	297
가져오기 탭. . . . .	297
도우미 탭. . . . .	297
입력 시 탭. . . . .	297
끝에서 탭. . . . .	298
함수 탭. . . . .	298
전체 코드 탭. . . . .	299
Java 변환을 통한 필터 최적화. . . . .	299
Java 변환의 초기 선택 최적화. . . . .	299
Java 변환의 푸시인 최적화. . . . .	300
Java 변환 작성. . . . .	301
재사용 가능 Java 변환 작성. . . . .	301
재사용 불가능 Java 변환 작성. . . . .	302
Java 변환 컴파일. . . . .	302
Java 변환 문제 해결. . . . .	303
컴파일 오류의 소스 찾기. . . . .	303
컴파일 오류의 소스 식별. . . . .	303
구조체 데이터로 변환의 예. . . . .	304
Java 변환 - 비원시 환경. . . . .	307
Java 변환 - Blaze 엔진. . . . .	307
Java 변환 - Spark 엔진. . . . .	308
<b>장 20: Java 변환 API 참조. . . . .</b>	<b>310</b>
Java 변환 API 메서드 개요. . . . .	310
커밋. . . . .	311
defineJExpression. . . . .	312
failSession. . . . .	312
generateRow. . . . .	313
getInRowType. . . . .	314
getMetadata. . . . .	314
incrementErrorCount. . . . .	315
invokeJExpression. . . . .	315
isNull. . . . .	316
logError. . . . .	317
logInfo. . . . .	318
resetNotification. . . . .	318
롤백. . . . .	319
setNull. . . . .	319

setOutRowType. . . . .	320
storeMetadata. . . . .	321
<b>장 21: Java 식. . . . .</b>	<b>322</b>
Java 식 개요. . . . .	322
식 함수 유형. . . . .	322
식 정의함수 정의 대화 상자를 사용하여 식 정의. . . . .	323
1단계. 함수 구성. . . . .	323
2단계. 식 작성 및 유효성 검사. . . . .	324
3단계. 식에 대한 Java 코드 생성. . . . .	324
식 정의함수 정의 대화 상자를 사용한 식 작성 및 Java 코드 생성. . . . .	324
Java 식 템플릿. . . . .	324
단순 인터페이스 작업. . . . .	325
invokeJExpression. . . . .	325
단순 인터페이스 예제. . . . .	326
고급 인터페이스 작업. . . . .	326
고급 인터페이스를 사용하여 식 호출. . . . .	327
고급 인터페이스 작업에 대한 규칙 및 지침. . . . .	327
EDatatype 클래스. . . . .	328
JExprParamMetadata 클래스. . . . .	328
defineJExpression. . . . .	329
JExpression 클래스. . . . .	329
고급 인터페이스 예제. . . . .	330
JExpression 클래스 API 참조. . . . .	331
getBytes. . . . .	331
getDouble. . . . .	331
getInt. . . . .	331
getLong. . . . .	332
getResultDataType. . . . .	332
getResultMetadata. . . . .	332
getStringBuffer. . . . .	332
invoke. . . . .	333
isResultNull. . . . .	333
<b>장 22: 조이너 변환. . . . .</b>	<b>334</b>
조이너 변환 개요. . . . .	334
조이너 변환의 고급 속성. . . . .	335
조이너 캐시. . . . .	336
조이너 변환의 포트. . . . .	337
동적 매핑의 조이너 변환. . . . .	337
조이너 변환의 포트 선택기. . . . .	338
선택 규칙. . . . .	338

포트 선택기 작성.....	339
조인 조건 정의.....	340
단순 조건 유형.....	341
고급 조건 유형.....	341
조인 조건의 포트 선택기.....	342
조인 조건의 동적 포트.....	343
식 매개 변수.....	343
조인 유형.....	343
일반 조인.....	344
마스터 외부 조인.....	345
세부 외부 조인.....	345
전체 외부 조인.....	345
조이너 변환의 정렬된 입력.....	346
정렬 순서 구성.....	346
매핑에 변환 추가.....	346
조인 조건에 대한 규칙 및 지침.....	347
조인 조건 및 정렬 순서의 예제.....	347
동일한 소스의 데이터 조인.....	349
동일한 파이프라인의 분기 2개 조인.....	349
동일한 소스의 인스턴스 2개 조인.....	350
동일한 소스의 데이터 조인 지침.....	351
소스 파이프라인 차단.....	351
정렬되지 않은 조이너 변환.....	351
정렬된 조이너 변환.....	351
조이너 변환 성능 팁.....	352
조이너 변환에 대한 규칙 및 지침.....	352
조이너 변환 - 비원시 환경.....	353
조이너 변환 - Blaze 엔진.....	353
조이너 변환 - Spark 엔진.....	353
조이너 변환 - Databricks Spark 엔진.....	353

## 장 23: 키 생성기 변환..... 354

키 생성기 변환 개요.....	354
Soundex 전략.....	355
Soundex 전략 속성.....	355
문자열 전략.....	355
문자열 전략 속성.....	356
NYSIIS 전략.....	356
키 생성기 출력 포트.....	356
그룹화 전략 구성.....	357
키 작성 속성.....	357
키 생성기 변환 고급 속성.....	358

키 생성기 변환 - 비원시 환경.....	358
------------------------	-----

## 장 24: 라벨러 변환..... 359

라벨러 변환 개요.....	359
라벨러 변환 사용 시기.....	360
라벨러 변환의 참조 데이터 사용.....	361
문자 집합.....	361
확률 모델.....	361
참조 테이블.....	362
정규식.....	362
토큰 집합.....	362
라벨러 변환 전략.....	362
문자 레이블 지정 작업.....	363
토큰 레이블 지정 작업.....	363
라벨러 변환 포트.....	363
문자 레이블 지정 속성.....	364
일반 속성.....	364
참조 테이블 속성.....	364
문자 집합 속성.....	365
필터 속성.....	365
토큰 레이블 지정 속성.....	366
일반 속성.....	366
토큰 집합 속성.....	366
사용자 지정 레이블 속성.....	367
확률 일치 속성.....	367
참조 테이블 속성.....	368
문자 레이블 지정 전략 구성.....	368
토큰 레이블 지정 전략 구성.....	369
라벨러 변환의 고급 속성.....	369
라벨러 변환 - 비원시 환경.....	370

## 장 25: 조회 변환..... 371

조회 변환 개요.....	371
연결된 조회 및 연결되지 않은 조회.....	372
연결된 조회.....	373
연결되지 않은 조회.....	374
조회 변환 개발.....	374
조회 쿼리.....	374
기본 조회 쿼리.....	375
조회 쿼리에 대한 SQL 재정의.....	375
SQL 재정의 쿼리의 매개 변수.....	375
예약어.....	376

조회 쿼리 재정의에 대한 지침.....	376
조회 쿼리 재정의.....	377
조회 소스 필터.....	377
조회에서 소스 행 필터링.....	378
조회 조건.....	378
조회 조건 구성.....	379
조회 변환 조건의 규칙 및 지침.....	380
조회 캐시.....	381
쿼리 속성.....	381
동적 매핑의 조회 변환.....	382
동적 포트 정의.....	382
조회 소스 변경.....	382
조회 소스 매개 변수화.....	383
매개 변수를 포함하는 조회 소스.....	385
중복 데이터 개체의 매개 변수 구성.....	385
포트 선택기.....	387
포트 선택기 구성.....	388
선택 규칙.....	389
조회 조건 매개 변수화.....	390
포트 선택기 작성.....	391
런타임 속성.....	392
고급 속성.....	393
재사용 가능한 조회 변환 작성.....	395
재사용 불가능한 조회 변환 작성.....	396
연결되지 않은 조회 변환 작성.....	397
연결되지 않은 조회 예제.....	397
조회 변환 - 비원시 환경.....	399
조회 변환 - Blaze 엔진.....	399
조회 변환 - Spark 엔진.....	399
조회 변환 - Databricks Spark 엔진.....	400

## **장 26: 조회 캐시..... 401**

조회 캐시 개요.....	401
조회 캐시 유형.....	402
캐싱되지 않은 조회.....	402
정적 조회 캐시.....	403
지속형 조회 캐시.....	403
지속형 조회 캐시 재작성.....	403
동적 조회 캐시.....	404
공유 조회 캐시.....	404
조회 캐시 공유에 대한 규칙 및 지침.....	405
캐시 비교.....	406

조회에 대한 캐시 분할. ....	406
--------------------	-----

## 장 27: 동적 조회 캐시..... 407

동적 조회 캐시 개요. ....	407
동적 조회 캐시 사용. ....	408
동적 조회 캐시 속성. ....	409
동적 조회 캐시 및 출력 값. ....	411
조회 변환 값. ....	411
조회 변환 값 예제. ....	411
SQL 재정의 및 동적 조회 캐시. ....	414
동적 조회 캐시의 매핑 구성. ....	414
삽입 기타 항목 업데이트. ....	415
업데이트 기타 항목 삽입. ....	415
동적 조회 캐시 및 대상 동기화. ....	416
조건부 동적 조회 캐시 업데이트. ....	417
조건부 동적 조회 캐시 처리. ....	417
조건부 동적 조회 캐시 구성. ....	417
식 결과로 동적 캐시 업데이트. ....	418
Null 식 값. ....	418
식 처리. ....	418
동적 캐시 업데이트를 위한 식 구성. ....	418
동적 조회 캐시 예제. ....	419
동적 조회 캐시에 대한 규칙 및 지침. ....	420

## 장 28: 일치 변환..... 422

일치 변환 개요. ....	422
일치 분석. ....	423
열 분석. ....	423
단일 소스 분석 및 이중 소스 분석. ....	424
필드 일치 분석 및 ID 일치 분석. ....	424
일치 분석의 그룹. ....	425
일치 쌍 및 클러스터. ....	425
일치 점수 계산. ....	426
가중치 점수. ....	426
Null 일치 점수. ....	427
클러스터 출력 옵션. ....	427
클러스터 분석의 드라이버 점수 및 링크 점수. ....	428
마스터 데이터 분석. ....	429
매핑 재사용. ....	430
ID 일치 분석 및 지속형 인덱스 데이터. ....	430
지속형 인덱스 데이터에 대한 규칙 및 지침. ....	431
일치 매핑의 성능. ....	431

일치 클러스터 분석 데이터 보기.....	432
일치 성능 분석 데이터 보기.....	433
ID 분석의 일치 성능.....	433
ID 인덱스 데이터에 대한 데이터 저장소 작성.....	434
단일 소스 분석에서 인덱스 데이터 저장소 사용.....	435
일치 변환 보기.....	436
일치 변환 포트.....	437
일치 변환 입력 포트.....	437
일치 변환 출력 포트.....	438
지속성 상태 코드 및 지속성 상태 설명.....	439
출력 포트 및 일치 출력 선택.....	441
일치 맵셋.....	441
일치 맵셋 작성.....	442
일치 맵셋 사용.....	442
일치 분석 작업 구성.....	443
일치 변환 - 비원시 환경.....	443
일치 변환 - Blaze 엔진.....	443
일치 변환 - Spark 엔진.....	443
<b>장 29: 필드 분석의 일치 변환.....</b>	<b>444</b>
필드 일치 분석.....	444
필드 일치 분석을 위한 프로세스 흐름.....	444
필드 일치 유형 옵션.....	445
필드 일치 전략.....	445
필드 일치 알고리즘.....	445
필드 일치 전략 속성.....	447
필드 일치 출력 옵션.....	448
일치 출력 유형.....	448
일치 출력 속성.....	448
필드 일치의 고급 속성.....	449
필드 일치 분석 예제.....	450
매핑 작성.....	450
입력 데이터 샘플.....	451
키 생성기 변환 구성.....	451
일치 변환 구성.....	452
데이터 뷰어 실행.....	454
결론.....	454
<b>장 30: ID 분석의 일치 변환.....</b>	<b>455</b>
ID 일치 분석.....	455
ID 일치 분석의 프로세스 흐름.....	456
ID 일치 유형 속성.....	456



인덱스 디렉터리 및 캐시 디렉터리 속성.....	458
지속성 방법 매개 변수.....	459
ID 일치 전략.....	460
ID 일치 알고리즘.....	460
ID 일치 전략 속성.....	461
ID 일치 출력 옵션.....	462
일치 출력 유형.....	462
일치 출력 속성.....	463
ID 일치의 고급 속성.....	465
지속형 인덱스 사례 연구.....	466
ID 일치 분석 예제.....	468
매핑 작성.....	468
입력 데이터 샘플.....	469
식 변환 구성.....	469
일치 변환 구성.....	469
데이터 뷰어 실행.....	473
결론.....	474
<b>장 31: 병합 변환.....</b>	<b>475</b>
병합 변환 개요.....	475
병합 전략 구성.....	475
병합 변환 고급 속성.....	476
병합 변환 - 비원시 환경.....	476
<b>장 32: 노멀라이저 변환.....</b>	<b>477</b>
노멀라이저 변환 개요.....	477
여러 번 발생하는 필드.....	478
생성된 열 ID.....	478
여러 번 발생하는 레코드.....	478
입력 계층 정의.....	480
노멀라이저 변환 입력 포트.....	481
필드 병합.....	481
필드 플랫폼.....	482
노멀라이저 변환 출력 그룹 및 포트.....	487
출력 그룹 작성.....	488
출력 그룹 업데이트.....	490
출력 그룹에 대한 키 생성.....	491
노멀라이저 변환 고급 속성.....	491
첫 번째 수준의 출력 그룹 생성.....	491
노멀라이저 변환 작성.....	492
업스트림 소스에서 노멀라이저 변환 작성.....	492
노멀라이저 매핑 예제.....	493

노멀라이저 예제 매핑.....	493
노멀라이저 예제 정의.....	494
노멀라이저 예제 입력 및 출력 그룹.....	494
노멀라이저 예제 매핑 출력.....	495
노멀라이저 변환 - 비원시 환경.....	496

## 장 33: 파서 변환..... 497

파서 변환 개요.....	497
파서 변환 모드.....	498
파서 변환 사용 시기.....	498
파서 변환에서 참조 데이터 사용.....	499
패턴 집합.....	500
확률 모델.....	500
참조 테이블.....	500
정규식.....	501
토큰 집합.....	501
토큰 구문 분석 작업.....	501
토큰 구문 분석 포트.....	501
토큰 구문 분석 속성.....	502
일반 속성.....	502
확률 모델 속성.....	503
참조 테이블 속성.....	503
토큰 집합 속성.....	504
패턴 기반 구문 분석 모드.....	505
패턴 기반 구문 분석 포트.....	505
토큰 구문 분석 전략 구성.....	505
패턴 구문 분석 전략 구성.....	506
파서 변환 고급 속성.....	507
파서 변환 - 비원시 환경.....	507

## 장 34: Python 변환..... 508

Python 변환 개요.....	508
데이터 유형 변환.....	508
입력 및 출력 포트의 데이터 유형.....	509
Python 변환 포트.....	510
Python 변환 고급 속성.....	510
Python 변환 구성 요소.....	510
리소스 파일.....	511
Python 코드.....	511
규칙 및 지침.....	512
Python 변환 생성.....	512
재사용 가능한 Python 변환 생성.....	512

재사용 불가능한 Python 변환 생성.....	513
Python 변환 사용 사례.....	513
Python 변환 - 비원시 환경.....	514
Python 변환 - Spark 엔진.....	514

## 장 35: 순위 변환..... 515

순위 변환 개요.....	515
문자열 값 순위 지정.....	516
순위 변환 속성.....	516
동적 매핑의 순위 변환.....	516
순위 변환 포트.....	516
순위 인덱스.....	517
순위 포트.....	518
그룹 기준 포트 정의.....	518
그룹 기준 매개 변수.....	519
순위 캐시.....	519
순위 변환 고급 속성.....	520
순위 변환 - 비원시 환경.....	520
순위 변환 - Blaze 엔진.....	521
순위 변환 - Spark 엔진.....	521
순위 변환 - Databricks Spark 엔진.....	521

## 장 36: 읽기 변환..... 522

읽기 변환 개요.....	522
읽기 변환 속성.....	522
일반 속성.....	523
데이터 개체 속성.....	524
쿼리 속성.....	524
런타임 속성.....	524
소스 속성.....	525
고급 속성.....	525
관계형 데이터 개체 동기화.....	525
소스 데이터 개체 변경.....	526
읽기 변환 매개 변수화.....	527
읽기 변환 매개 변수.....	528
제약 조건.....	529
읽기 변환 작성.....	529
매핑 편집기에서 읽기 변환 작성.....	529

## 장 37: 관계형-계층 변환..... 531

관계형-계층 변환 개요.....	531
예제 - 관계형-계층 변환.....	531

입력 관계형 포트 및 개요 보기. ....	533
관계형-계층 변환 포트. ....	534
스키마 참조. ....	534
관계형-계층 변환 개발. ....	535
관계형-계층 변환 작성. ....	535
포트 작성. ....	535

## **장 38: REST 웹 서비스 소비자 변환..... 537**

REST 웹 서비스 소비자 변환 개요. ....	537
REST 웹 서비스 소비자 변환 프로세스. ....	538
REST 웹 서비스 소비자 변환 구성. ....	539
메시지 구성. ....	539
리소스 식별. ....	539
HTTP 메서드. ....	540
HTTP Get 메서드. ....	540
HTTP Post 메서드. ....	541
HTTP Put 메서드. ....	542
HTTP Delete 메서드. ....	542
REST 웹 서비스 소비자 변환 포트. ....	543
입력 포트. ....	543
출력 포트. ....	543
통과 포트. ....	544
인수 포트. ....	544
URL 포트. ....	544
HTTP 헤더 포트. ....	544
쿠키 포트. ....	545
출력 XML 포트. ....	545
응답 코드 포트. ....	545
REST 웹 서비스 소비자 변환 입력 매핑. ....	545
입력 포트를 요소에 매핑하기 위한 규칙 및 지침. ....	546
메서드 입력에 입력 포트 매핑. ....	546
REST 웹 서비스 소비자 변환 출력 매핑. ....	547
요소를 출력 포트에 매핑하기 위한 규칙 및 지침. ....	548
보기 사용자 지정의 옵션. ....	548
출력 포트에 메서드 출력 매핑. ....	549
REST 웹 서비스 소비자 변환의 고급 속성. ....	549
REST 웹 서비스 소비자 변환 작성. ....	550
REST 웹 서비스 소비자 변환 작성. ....	550
배열이 포함된 JSON 응답 메시지 구문 분석. ....	551
JSON 응답 메시지 예제. ....	551
응답 메시지의 명명되지 않은 배열. ....	552

<b>장 39: 라우터 변환.....</b>	<b>553</b>
라우터 변환 개요.....	553
동적 매핑의 라우터 변환.....	554
그룹 작업.....	554
입력 그룹.....	555
출력 그룹.....	555
그룹 필터 조건 사용.....	555
그룹 필터 조건의 동적 포트.....	557
그룹 필터 매개 변수화.....	557
그룹 추가.....	558
포트 작업.....	558
매핑에서 라우터 변환 연결.....	559
라우터 변환 고급 속성.....	559
라우터 변환 - 비원시 환경.....	559
 <b>장 40: 시퀀스 생성기 변환.....</b>	 <b>560</b>
시퀀스 생성기 변환 개요.....	560
시퀀스 생성기 포트.....	561
통과 포트.....	561
NEXTVAL 포트.....	561
CURRVAL.....	565
시퀀스 생성기 변환 속성.....	566
시작 값.....	567
증분 범위.....	568
끝 값.....	568
증분값.....	568
값 범위 반복.....	569
현재 값.....	569
총 캐시된 값.....	569
재사용 불가능 시퀀스 생성기.....	570
재사용 가능 시퀀스 생성기.....	570
재설정.....	571
행 순서 유지.....	571
시퀀스 데이터 개체.....	571
시퀀스 데이터 개체 생성.....	572
시퀀스 생성기 변환 작성.....	573
시퀀스 생성기 변환 작성.....	574
FAQ.....	575
시퀀스 생성기 변환 - 비원시 환경.....	576
시퀀스 생성기 변환 - Blaze 엔진.....	576
시퀀스 생성기 변환 - Spark 엔진.....	576

<b>장 41: 분류기 변환.....</b>	<b>577</b>
분류기 변환 개요.....	577
동적 매핑의 분류기 변환.....	578
분류기 변환 개발.....	578
분류기 변환 포트.....	578
정렬 탭.....	579
정렬 키 구성.....	579
정렬 키 매개 변수화.....	580
분류기 변환 고급 속성.....	581
분류기 캐시.....	582
분류기 캐시 최적화.....	582
분류기 변환 작성.....	582
재사용 가능한 분류기 변환 작성.....	583
재사용 불가능 분류기 변환 작성.....	583
분류기 변환 예제.....	583
분류기 변환 - 비원시 환경.....	584
분류기 변환 - Blaze 엔진.....	585
분류기 변환 - Spark 엔진.....	585
분류기 변환 - Databricks Spark 엔진.....	586
 <b>장 42: SQL 변환.....</b>	 <b>587</b>
SQL 변환 개요.....	587
SQL 변환 포트.....	588
입력 포트.....	588
출력 포트.....	589
통과 포트.....	590
SQLException 포트.....	591
영향을 받는 행 수.....	591
SQL 변환 고급 속성.....	592
SQL 변환 쿼리.....	593
SQL 쿼리 정의.....	594
입력 행 대 출력 행의 카디널리티.....	595
쿼리 문 처리.....	596
포트 구성.....	596
최대 출력 행 개수.....	596
오류 행.....	597
SQL 오류에 대해 계속.....	598
SQL 변환을 통한 필터 최적화.....	598
SQL 변환의 초기 선택 최적화.....	599
SQL 변환의 푸시인 최적화.....	599
SQL 쿼리가 포함된 SQL 변환 예제.....	600

논리적 데이터 개체 매핑.....	600
급여 테이블.....	600
직원 테이블.....	601
<b>SQL 변환.....</b>	<b>601</b>
출력.....	603
저장 프로시저.....	603
저장 프로시저용 SQL 변환 포트.....	604
저장 프로시저 결과 집합.....	605
저장 프로시저 예제.....	607
<b>SQL 변환 연결.....</b>	<b>608</b>
연결 이름 매개 변수 작성.....	608
수동으로 SQL 변환 작성.....	609
저장 프로시저에서 SQL 변환 작성.....	610
 <b>장 43: 표준화 변환.....</b>	 <b>611</b>
표준화 변환 개요.....	611
표준화 전략.....	611
표준화 속성.....	612
표준화 전략 구성.....	613
표준화 변환 고급 속성.....	613
 <b>장 44: 합집합 변환.....</b>	 <b>614</b>
합집합 변환 개요.....	614
그룹 및 포트.....	615
합집합 변환 고급 속성.....	615
합집합 변환 처리.....	616
합집합 변환 작성.....	616
재사용 가능한 합집합 변환 작성.....	616
재사용 불가능 합집합 변환 작성.....	616
합집합 변환 - 비원시 환경.....	617
합집합 변환 - Databricks Spark 엔진.....	617
 <b>장 45: 업데이트 전략 변환.....</b>	 <b>618</b>
업데이트 전략 변환 개요.....	618
업데이트 전략 설정.....	618
동적 매핑의 업데이트 전략 변환.....	619
매핑 내 행에 플래그 지정.....	619
업데이트 전략 식.....	619
업데이트 전략 변환 고급 속성.....	620
집계 및 업데이트 전략 변환.....	620
개별 대상의 업데이트 옵션 지정.....	620
업데이트 전략 변환 - 비원시 환경.....	621

업데이트 전략 변환 - Blaze 엔진.....	621
업데이트 전략 변환 - Spark 엔진.....	622

## 장 46: 웹 서비스 소비자 변환..... 624

웹 서비스 소비자 변환 개요.....	624
SOAP 메시지.....	624
WSDL 파일.....	625
작업.....	625
웹 서비스 보안.....	626
WSDL 선택.....	626
웹 서비스 소비자 변환 포트.....	627
HTTP 헤더 입력 포트.....	627
기타 입력 포트.....	628
웹 서비스 소비자 변환 입력 매핑.....	628
입력 포트를 노드에 매핑하기 위한 규칙 및 지침.....	629
보기 사용자 지정의 옵션.....	629
작업 입력에 입력 포트 매핑.....	630
웹 서비스 소비자 변환 출력 매핑.....	631
노드를 출력 포트에 매핑하기 위한 규칙 및 지침.....	632
SOAP 메시지를 XML로 매핑.....	633
보기 사용자 지정의 옵션.....	633
출력 포트로 작업 출력 매핑.....	633
웹 서비스 소비자 변환 고급 속성.....	634
웹 서비스 오류 처리.....	636
메시지 압축.....	637
동시 실행.....	637
필터 최적화.....	638
웹 서비스 소비자 변환의 초기 선택 최적화 활성화.....	638
웹 서비스 소비자 변환의 푸시인 최적화.....	638
웹 서비스 소비자 변환 작성.....	640
웹 서비스 소비자 변환 예제.....	642
입력 파일.....	642
논리적 데이터 개체 모델.....	642
논리적 데이터 개체 매핑.....	642
웹 서비스 소비자 변환.....	643

## 장 47: 웹 서비스 SOAP 메시지 구문 분석..... 645

웹 서비스 SOAP 메시지 구문 분석 개요.....	645
변환 사용자 인터페이스.....	645
다중 발생 출력 구성.....	646
정규화된 관계형 출력.....	647
생성된 키.....	647



비정규화된 관계형 출력.....	648
피벗된 관계형 출력.....	648
anyType 요소 구문 분석.....	649
파생된 유형 구문 분석.....	650
QName 요소 구문 분석.....	651
대체 그룹 구문 분석.....	651
SOAP 메시지의 XML 구성 구문 분석.....	651
선택 요소.....	651
목록 요소.....	652
합집합 요소.....	652

## 장 48: 웹 서비스 SOAP 메시지 생성..... 653

웹 서비스 SOAP 메시지 생성 개요.....	653
변환 사용자 인터페이스.....	654
입력 포트 영역.....	654
작업 영역.....	655
포트 및 계층 수준 관계.....	655
키.....	656
포트 매핑.....	657
포트 매핑.....	658
그룹 매핑.....	658
여러 포트 매핑.....	659
여러 번 발생하는 포트 피벗.....	659
비정규화된 데이터 매핑.....	660
파생된 유형 및 요소 대체.....	661
파생된 유형 생성.....	662
anyType 요소 및 특성 생성.....	662
대체 그룹 생성.....	663
SOAP 메시지의 XML 구성 생성.....	663
선택 요소.....	663
목록 요소.....	664
합집합 요소.....	664

## 장 49: 가중치 평균 변환..... 666

가중치 평균 변환 개요.....	666
가중치 평균 변환 구성.....	666
가중치 일치 점수 예제.....	667
가중치 평균 변환 고급 속성.....	667
가중치 평균 변환 - 비원시 환경.....	667

## 장 50: 쓰기 변환..... 668

쓰기 변환 개요.....	668
---------------	-----

쓰기 변환 속성 .....	668
일반 속성 .....	669
데이터 개체 속성 .....	670
포트 속성 .....	670
런타임 속성 .....	670
런타임 링크 속성 .....	671
고급 속성 .....	672
쓰기 변환 작성 .....	674
데이터 개체에서 쓰기 변환 작성 .....	674
매핑 흐름에서 쓰기 변환 작성 .....	674
매개 변수에서 쓰기 변환 작성 .....	675
기존 변환에서 쓰기 변환 생성 .....	676
<b>부록 A: 변환 구분자</b> .....	<b>678</b>
변환 구분자 개요 .....	678
<b>인덱스</b> .....	<b>679</b>

# 서문

*Informatica Developer 변환 가이드*에는 **Developer tool**의 변환 기능에 대한 정보가 수록되어 있습니다. 이 가이드는 데이터 품질, 빅 데이터 및 데이터 서비스 개발자를 대상으로 작성되었습니다. 이 가이드에서는 사용자가 데이터 품질 개념, 플랫폼 파일, 관계형 데이터베이스 개념 및 사용자 환경의 데이터베이스 엔진을 알고 있다고 가정합니다.

비원시 환경에서는 런타임 엔진에 따라 매핑 논리가 다르게 처리될 수 있습니다. 비원시 환경에서 Informatica 변환은 완벽하게 지원되거나, 제한적으로 지원되거나, 지원되지 않을 수 있습니다. 마찬가지로, 원시 환경에서도 일부 Informatica 변환 및 변환 동작이 지원되지 않을 수 있습니다.

비원시 환경에서 매핑 유효성을 검사하고 매핑을 실행하기 전에 *Big Data Management 사용자 가이드*를 참조하여 비원시 환경에서 지원되는 변환과 처리 제한에 대해 확인하십시오.

## Informatica 리소스

Informatica는 Informatica Network 및 기타 온라인 포털을 통해 다양한 범위의 제품 리소스를 제공합니다. 리소스를 통해 Informatica 제품 및 솔루션을 최대한 활용하고 다른 Informatica 사용자 및 주제별 전문가로부터 배울 수 있습니다.

### Informatica 네트워크

Informatica Network는 Informatica 기술 자료, Informatica 글로벌 고객 지원 센터 등 여러 리소스로 연결되는 관문입니다. Informatica Network를 시작하려면 <https://network.informatica.com>을 방문하십시오.

Informatica Network 멤버인 경우 다음 옵션이 가능합니다.

- 기술 자료에서 제품 리소스를 검색할 수 있습니다.
- 제품 사용 가능 여부에 대한 정보를 봅니다.
- 지원 사례를 생성하고 검토할 수 있습니다.
- 거주 지역의 Informatica 사용자 그룹 네트워크를 검색하고 동료와 협업 관계 유지

### Informatica 기술 자료

Informatica 기술 자료를 사용하여 사용 방법 문서, 모범 사례, 비디오 자습서, 자주 묻는 질문에 대한 답변 등 제품 리소스를 확인할 수 있습니다.

기술 자료를 검색하려면 <https://search.informatica.com>을 방문하십시오. 기술 자료에 대한 질문, 의견 또는 아이디어가 있는 경우 [KB\\_Feedback@informatica.com](mailto:KB_Feedback@informatica.com)을 통해 Informatica 기술 자료 팀에 문의해 주시기 바랍니다.

## Informatica 설명서

Informatica 설명서 포털에서 확장된 설명서 라이브러리를 탐색하여 현재 및 최근 제품 릴리스를 확인할 수 있습니다. 설명서 포털을 탐색하려면 <https://docs.informatica.com>을 방문하십시오.

Informatica는 설명서 포털뿐 아니라 Informatica 기술 자료에서 여러 제품에 대한 설명서를 유지 관리합니다. 설명서 포털에서 제품 또는 제품 버전에 해당하는 설명서를 찾을 수 없는 경우 <https://search.informatica.com>에서 기술 자료를 검색하십시오.

제품 설명서에 대한 질문, 의견 또는 아이디어가 있는 경우 [infa\\_documentation@informatica.com](mailto:infa_documentation@informatica.com)에서 Informatica 설명서 팀에 문의해 주시기 바랍니다.

## Informatica Product Availability Matrix

PAM(Product Availability Matrix)은 제품 릴리스에서 지원하는 운영 체제 버전, 데이터베이스 및 데이터 소스 유형과 대상을 나타냅니다.

<https://network.informatica.com/community/informatica-network/product-availability-matrices>에서 Informatica PAM을 찾을 수 있습니다.

## Informatica Velocity

Informatica Velocity는 수백 가지 데이터 관리 프로젝트의 실제 경험을 토대로 Informatica 전문 서비스업에서 개발한 팁과 모범 사례 모음입니다. Informatica Velocity는 전 세계의 조직과 협력하여 성공적인 데이터 관리 솔루션을 계획, 개발, 배포 및 유지 관리하는 Informatica 컨설턴트의 포괄적인 지식을 보여줍니다.

Informatica Velocity 리소스는 <http://velocity.informatica.com>에서 확인할 수 있습니다. Informatica Velocity에 대한 질문, 주석 또는 아이디어가 있으시면 Informatica 전문 서비스업([ips@informatica.com](mailto:ips@informatica.com))에 문의하십시오.

## Informatica Marketplace

Informatica Marketplace는 Informatica 구현을 확대 및 개선하기 위한 솔루션을 찾을 수 있는 포럼입니다. Marketplace에서 Informatica 개발자와 파트너가 제공하는 수백 개의 솔루션을 활용하여 생산성을 향상시키고 프로젝트의 구현에 걸리는 시간을 줄일 수 있습니다. <https://marketplace.informatica.com>에서 Informatica Marketplace를 찾을 수 있습니다.

## Informatica 글로벌 고객 지원 센터

전화 또는 Informatica Network를 통해 글로벌 지원 센터에 문의할 수 있습니다.

해당 지역의 Informatica 글로벌 고객 지원 전화 번호는 Informatica 웹 사이트 (<https://www.informatica.com/services-and-training/customer-success-services/contact-us.html>)를 방문하여 찾을 수 있습니다.

Informatica Network에서 온라인 지원 리소스를 찾으려면 <https://network.informatica.com>을 방문하고 eSupport 옵션을 선택하십시오.

# 제 1 장

## 변환 소개

이 장에 포함된 항목:

- [변환 소개 개요, 33](#)
- [변환 - 원시 및 비원시 환경, 37](#)
- [변환 데이터 유형 처리, 39](#)
- [변환 개발, 42](#)
- [다중 그룹 변환, 42](#)
- [변환의 식, 43](#)
- [로컬 변수, 47](#)
- [포트의 기본값, 49](#)
- [추적 수준, 56](#)
- [재사용 가능한 변환, 56](#)
- [재사용 불가능 변환, 58](#)
- [변환 작성, 58](#)

## 변환 소개 개요

변환은 데이터를 생성하거나 수정하거나 전달하는 개체입니다.

**Informatica Developer**는 특정 기능을 수행하는 변환 집합을 제공합니다. 예를 들어 집계 변환은 데이터 그룹에 대한 계산을 수행합니다.

매핑의 변환은 데이터 통합 서비스가 데이터에 대해 수행하는 작업을 나타냅니다. 데이터는 사용자가 매핑 또는 맵렛에서 연결하는 변환 포트를 통해 전달됩니다.

변환은 활성 또는 수동일 수 있습니다. 변환은 데이터 흐름에 연결될 수도 있고 연결되지 않을 수도 있습니다.

### 활성 변환

활성 변환은 변환을 통해 전달되는 행의 수를 변경합니다. 또는 행 유형을 변경합니다.

예를 들어 필터 변환은 필터 조건과 일치하지 않는 행을 제거하기 때문에 활성입니다. 업데이트 전략 변환은 행에 삽입, 삭제, 업데이트 또는 거부 플래그를 지정하기 때문에 활성입니다.

데이터 통합 서비스는 활성 변환에서 전달되는 행을 연결하지 못할 수 있기 때문에 여러 개의 활성 변환 또는 활성 및 수동 변환을 동일한 다운스트림 변환 또는 변환 입력 그룹에 연결할 수 없습니다.

예를 들어 행에 삭제 플래그를 지정하는 업데이트 전략 변환이 매핑의 한 분기에 있다고 가정합니다. 행에 삽입 플래그를 지정하는 업데이트 전략 변환이 또 다른 분기에 있습니다. 이 두 변환을 단일 변환 입력 그룹에 연결하면 데이터 통합 서비스가 행에 대해 삭제 및 삽입 작업을 결합할 수 없습니다.

## 수동 변환

수동 변환은 변환을 통과하는 행 수를 변경하지 않고 트랜잭션 경계를 유지 관리하고 행 유형을 유지 관리합니다.

업스트림 분기의 모든 변환이 수동인 경우 여러 변환을 동일한 다운스트림 변환 또는 동일한 변환 입력 그룹에 연결할 수 있습니다. 분기를 발생시키는 변환은 활성이거나 수동일 수 있습니다.

## 연결되지 않은 변환

변환은 데이터 흐름에 연결될 수도 있고 연결되지 않을 수도 있습니다. 연결되지 않은 변환은 매핑의 다른 변환과 연결되어 있지 않습니다. 연결되지 않은 변환은 다른 변환 내에서 호출되며 값을 해당 변환에 반환합니다.

## 다중 전략 변환

전략은 변환이 데이터에 대해 수행하는 하나 이상의 작업 집합입니다. 변환의 각 전략에 서로 다른 입력 포트와 출력 포트 집합을 할당할 수 있습니다. 변환이 단일 변환 개체에서 정의하는 전략을 저장합니다.

**종속성** 보기를 사용하여 각 전략에서 사용하는 포트를 볼 수 있습니다.

다음과 같은 변환으로 여러 변환 전략을 정의할 수 있습니다.

- 대/소문자 변환기
- 분류자
- 결정
- 키 생성기
- 라벨러
- 일치
- 병합
- 파서
  - 파서 변환에서 여러 전략을 사용하려면 토큰을 구문 분석하도록 변환을 구성합니다.
- 표준화

## 변환 설명

Developer tool에는 일반 및 데이터 품질 변환이 있습니다. 일반 변환은 Informatica Data Quality와 Informatica Data Services에서 사용할 수 있습니다. 데이터 품질 변환은 Informatica Data Quality에서 사용할 수 있습니다.

다음 테이블에서는 각 변환에 대해 설명합니다.

변환	유형	설명
주소 유효성 검사기	활성 또는 수동/ 연결됨	우편 주소 레코드의 정확성을 확인 및 개선합니다. 또한 사용자가 우편 받는 사람을 선택하고 우편을 전달하는 데 도움이 되는 정보를 추가합니다.
연관	활성/ 연결됨	서로 다른 클러스터에 일치 변환이 할당한 중복 레코드 간의 링크를 작성합니다.
집계	활성/ 연결됨	집계 계산을 수행합니다.
잘못된 레코드 예외	활성/ 연결됨	데이터 오류가 포함되어 있을 수 있는 레코드를 식별하고, Analyst 도구 사용자가 검토 및 업데이트할 수 있는 테이블에 레코드를 로드합니다.
대/소문자 변환기	수동/ 연결됨	문자열의 대/소문자를 표준화합니다.
분류자	수동/ 연결됨	입력 포트 필드에 정보를 요약하는 레이블을 기록합니다. 필드에 상당한 양의 텍스트가 있을 때 사용합니다.
비교	수동/ 연결됨	입력 문자열 쌍 간의 유사도를 나타내는 숫자 점수를 생성합니다.
통합	활성/ 연결됨	일치 변환에서 중복으로 확인된 레코드에서 통합 레코드를 작성합니다.
데이터 마스킹	수동/ 연결됨 또는 연결되지 않음	비프로덕션 환경에 대해 중요한 프로덕션 데이터를 실제 테스트 데이터로 대체합니다.
데이터 프로세서	활성/ 연결됨	매핑에서 구조화되지 않은 파일 형식과 반구조화된 파일 형식을 처리합니다.
결정	수동/ 연결됨	입력 데이터의 조건을 평가하고 이러한 조건의 결과에 따라 출력을 작성합니다.
중복 레코드 예외	활성/ 연결됨	중복 정보가 포함되어 있을 수 있는 레코드를 식별하고, Analyst 도구 사용자가 검토 및 업데이트할 수 있는 테이블에 레코드를 로드합니다.
식	수동/ 연결됨	값을 계산합니다.
필터	활성/ 연결됨	데이터를 필터링합니다.

변환	유형	설명
계층-관계형	활성/ 연결됨	계층 입력을 처리하여 관계형 출력으로 변환합니다.
Java	활성 또는 수동/ 연결됨	Java로 코딩된 사용자 논리를 실행합니다. 리포지토리가 사용자 논리에 대한 바이트 코드를 저장합니다.
조이너	활성/ 연결됨	여러 데이터베이스 또는 플랫폼 파일 시스템의 데이터를 조인합니다.
키 생성기	활성/ 연결됨	사용자가 선택하는 열의 데이터 값을 기반으로 레코드를 그룹에 할당합니다.
라벨러	수동/ 연결됨	입력 포트 필드에 문자열 또는 특성을 설명하는 레이블을 기록합니다.
조회	활성 또는 수동/ 연결됨 또는 연 결되지 않음	플랫폼 파일, 논리적 데이터 개체, 참조 테이블, 관계형 테이블, 보기 또는 동의어에서 데이터를 조회하고 반환합니다.
일치	활성/ 연결됨	입력 레코드 간의 유사도를 나타내는 점수를 생성합니다.
병합	수동/ 연결됨	여러 입력 열에서 데이터 값을 읽고 단일 출력 열을 작성합니다.
노멀라이저	활성/ 연결됨	여러 번 발생하는 데이터가 포함된 소스 행을 처리하고 여러 번 발생하는 데이터의 각 인스턴스에 대해 대상 행을 반환합니다.
출력	수동/ 연결됨	맵렛 출력 행을 정의합니다.
파서	수동/ 연결됨	입력 포트 값에 포함된 정보의 유형에 따라 입력 포트 값을 별도의 출력 포트로 구문 분석합니다.
순위	활성/ 연결됨	레코드를 상위 또는 하위 범위로 제한합니다.
읽기	수동/ 연결됨	소스로부터 데이터를 읽습니다.
관계형-계층	활성/ 연결됨	관계형 입력을 처리하여 계층 출력으로 변환합니다.
REST 웹 서비스 소비자	활성/ 연결됨	데이터에 액세스하거나 데이터를 변환하기 위해 웹 서비스 클라이언트로 REST 웹 서비스에 연결합니다.
라우터	활성/ 연결됨	그룹 조건에 따라 데이터를 여러 변환으로 라우팅합니다.
시퀀스 생성기	수동/ 연결됨	값의 숫자 시퀀스를 생성합니다.



변환	유형	설명
분류기	활성/ 연결됨	정렬 키에 따라 데이터를 정렬합니다.
SQL	활성 또는 수동/ 연결됨	데이터베이스에 대해 SQL 쿼리를 실행합니다.
표준화	수동/ 연결됨	입력 문자열의 표준화된 버전을 생성합니다.
합집합	활성/ 연결됨	여러 데이터베이스 또는 플랫폼 파일 시스템의 데이터를 병합합니다.
업데이트 전략	활성/ 연결됨	행을 삽입, 삭제, 업데이트 또는 거부할지 여부를 지정합니다.
웹 서비스 소비자	활성/ 연결됨	데이터에 액세스하거나 데이터를 변환하기 위해 웹 서비스를 웹 서비스 클라이언트로 연결합니다.
가중치 평균	수동/ 연결됨	데이터 집합의 레코드에 대해 일치 변환에서 생성하는 점수 일치를 읽고 각 레코드 쌍에 대한 평균 점수를 계산합니다. 각 레코드 쌍에 대해 변환에서 생성하는 점수에 서로 다른 가중치를 적용할 수 있습니다.
쓰기	수동/ 연결됨	매핑이 데이터를 쓰는 대상을 나타냅니다.

## 변환 - 원시 및 비원시 환경

비원시 환경에서 실행되는 매핑은 원시 환경에서 실행되는 매핑과 다른 결과를 반환할 수 있습니다.

처리에 관한 다음 차이점을 고려하십시오.

- 비원시 환경은 분산 처리를 사용하며 서로 다른 노드에서 데이터를 처리합니다. 각 노드에서는 다른 노드에서 처리되는 데이터에 액세스할 수 없습니다. 따라서 런타임 엔진이 데이터가 시작된 순서를 확인하지 못할 수 있습니다. 그러므로 비원시 환경에서 매핑을 실행하고 원시 환경에서 동일한 매핑을 실행하면 두 매핑이 올바른 결과를 반환하지만 결과가 동일하지 않을 수 있습니다.
- 비원시 환경에서는 런타임 엔진에 따라 매핑 논리가 다르게 처리될 수 있습니다. 비원시 환경에서 **Informatica** 변환은 완벽하게 지원되거나, 제한적으로 지원되거나, 지원되지 않을 수 있습니다. 마찬가지로, 원시 환경에서도 일부 **Informatica** 변환 및 변환 동작이 지원되지 않을 수 있습니다.

다음 테이블에는 변환과 비원시 환경의 여러 엔진에 대한 지원이 나와 있습니다.

변환	지원되는 엔진
<i>이 테이블에 나열되지 않은 변환은 비원시 환경에서 지원되지 않습니다.</i>	
주소 유효성 검사기	<ul style="list-style-type: none"> <li>- Blaze</li> <li>- Spark*</li> </ul>

변환	지원되는 엔진
집계	- Blaze - Spark - Databricks Spark
대/소문자 변환기	- Blaze - Spark*
분류자	- Blaze - Spark*
비교	- Blaze - Spark*
통합	- Blaze - Spark*
데이터터 마스킹	- Blaze - Spark
데이터터 프로세서	- Blaze
결정	- Blaze - Spark*
식	- Blaze - Spark - Databricks Spark
필터	- Blaze - Spark - Databricks Spark
Java	- Blaze - Spark
조이너	- Blaze - Spark - Databricks Spark
키 생성기	- Blaze - Spark*
라벨러	- Blaze - Spark*
조회	- Blaze - Spark - Databricks Spark
일치	- Blaze - Spark*
병합	- Blaze - Spark*

변환	지원되는 엔진
노멀라이저	<ul style="list-style-type: none"> <li>- Blaze</li> <li>- Spark</li> <li>- Databricks Spark</li> </ul>
파서	<ul style="list-style-type: none"> <li>- Blaze</li> <li>- Spark*</li> </ul>
Python	<ul style="list-style-type: none"> <li>- Spark</li> </ul>
순위	<ul style="list-style-type: none"> <li>- Blaze</li> <li>- Spark</li> <li>- Databricks Spark</li> </ul>
라우터	<ul style="list-style-type: none"> <li>- Blaze</li> <li>- Spark</li> <li>- Databricks Spark</li> </ul>
시퀀스 생성기	<ul style="list-style-type: none"> <li>- Blaze</li> <li>- Spark*</li> </ul>
분류기	<ul style="list-style-type: none"> <li>- Blaze</li> <li>- Spark</li> <li>- Databricks Spark</li> </ul>
표준화	<ul style="list-style-type: none"> <li>- Blaze</li> <li>- Spark*</li> </ul>
합집합	<ul style="list-style-type: none"> <li>- Blaze</li> <li>- Spark</li> <li>- Databricks Spark</li> </ul>
업데이트 전략	<ul style="list-style-type: none"> <li>- Blaze</li> <li>- Spark*</li> </ul>
가중치 평균	<ul style="list-style-type: none"> <li>- Blaze</li> <li>- Spark*</li> </ul>
*Spark 엔진의 빅 데이터 스트리밍에 대해서는 지원되지 않습니다. 빅 데이터 스트리밍 변환에 대한 자세한 내용은 <i>Informatica Big Data Streaming 사용자 가이드</i> 를 참조하십시오.	

## 변환 데이터 유형 처리

변환은 데이터 유형 특정 함수를 처리하거나 데이터를 처리하지 않고 데이터 통과를 허용할 수 있습니다. 데이터 통합 서비스는 변환에 따라 10진수, 시간대가 포함된 타임스탬프 및 현지 시간대가 포함된 타임스탬프와 같은 일부 데이터 유형을 처리합니다.

## 10진수 데이터 유형

10진수 데이터 유형을 사용하여 플랫폼 파일 및 Oracle, Microsoft SQL Server, IBM DB2 및 ODBC와 같은 지원되는 데이터베이스에 데이터를 읽고 쓸 수 있습니다.

최대 38자리의 전체 자릿수를 지원하는 변환에서 전체 자릿수는 1 ~ 38자리이고 소수 자릿수는 0 ~ 38입니다.

### 10진수 데이터 유형을 지원하는 변환

다음 변환은 최대 38자리의 전체 자릿수가 있는 10진수 데이터 유형을 지원하며 데이터에 대한 계산을 수행할 수 있습니다.

- 집계
- 데이터 마스킹
- 식
- 필터
- Java
- 조이너
- 조회
- 노멀라이저
- 순위
- 라우터
- 시퀀스 생성기
- 분류기
- 합집합
- 업데이트 전략

### 10진수 데이터 유형에 대한 통과 지원이 있는 변환

일부 변환은 변환을 통해 최대 38자리의 전체 자릿수가 있는 10진수 데이터를 전달만 할 수 있습니다. 해당 변환은 데이터에 대한 계산을 수행할 수 없습니다. 변환에 대한 계산을 수행하는 데 최대 38자리의 전체 자릿수가 있는 10진수 데이터를 사용하는 경우 데이터 통합 서비스가 데이터 유형을 배경밀도로 처리합니다.

다음 변환에는 최대 38자리의 전체 자릿수가 있는 10진수 데이터 유형에 대한 통과 지원이 있습니다.

- 데이터 프로세서
- 계층-관계형
- REST 웹 서비스 소비자 변환
- SQL
- 웹 서비스 소비자

### 10진수 데이터 유형을 지원하지 않는 변환

Data Quality 변환과 같은 일부 변환은 10진수 데이터 유형을 지원하지 않습니다.

다음 Data Quality 변환 해당 목록은 최대 38자리의 전체 자릿수가 있는 10진수 데이터 유형을 지원하지 않습니다.

- 주소 유효성 검사기
- 연관
- 대/소문자 변환기

- 분류자
- 비교
- 통합
- 결정
- 키 생성기
- 라벨러
- 일치
- 병합
- 파서
- 표준화
- 가중치 평균

10진수 38 데이터 유형을 지원하지 않는 변환의 경우 10진수 데이터 유형에 28자리보다 큰 전체 자릿수가 있으면 데이터 통합 서비스가 10진수 값을 많은 전체 자릿수 모드에서 배정밀도로 변환합니다.

## 시간대가 포함된 타임스탬프

시간대가 포함된 타임스탬프는 시간대 오프셋 또는 시간대 지역이 포함된 타임스탬프 데이터 유형의 변형입니다.

다음 변환은 시간대 데이터 유형이 포함된 타임스탬프를 지원합니다.

- 집계
- 식
- 필터
- Java
- 조이너
- 조회
- 노멀라이저
- 순위
- 라우터
- 시퀀스 생성기
- 분류기
- 합집합
- 업데이트 전략

통과 지원은 변환을 통해 데이터를 전달할 수 있지만 시간대 데이터 유형이 포함된 타임스탬프의 기능은 수행할 수 없음을 의미합니다.

다음 변환에는 시간대 데이터 유형이 포함된 타임스탬프에 대한 통과 지원이 있습니다.

- 데이터 마스킹
- 데이터 프로세서
- 계층-관계형
- SQL

## 현지 시간대가 포함된 타임스탬프

현지 시간대가 포함된 타임스탬프는 변환에서 타임스탬프에 해당하는 기능으로 암시적으로 지원됩니다.

## 변환 개발

매핑을 작성할 때 변환을 추가하고 비즈니스 용도에 맞게 데이터를 처리하도록 변환을 구성합니다.

변환을 개발하여 매핑에 통합하려면 다음 작업을 완료합니다.

1. 재사용할 수 없는 변환을 매핑 또는 맵렛에 추가합니다. 또는 여러 매핑 또는 맵렛에 추가할 수 있는 재사용 가능한 변환을 작성합니다.
2. 변환을 구성합니다. 각 유형의 변환에는 구성 가능한 고유한 옵션 집합이 있습니다.
3. 변환이 재사용 가능하면 변환을 매핑 또는 맵렛에 추가합니다.
4. 변환을 매핑 또는 맵렛의 다른 개체에 연결합니다.

업스트림 개체의 포트를 변환 입력 포트에 끌어 놓습니다. 변환의 출력 포트를 다운스트림 개체의 포트에 끌어 놓습니다. 일부 변환에서는 사용자가 선택할 수 있는 미리 정의된 포트를 사용합니다.

**참고:** 재사용 가능한 변환을 작성하는 경우 변환을 다른 개체에 연결하기 전에 필요한 입력 및 출력 포트를 추가합니다. 맵렛 또는 매핑 캔버스에서 변환 인스턴스에 포트를 추가할 수 없습니다. 재사용 가능한 변환에서 포트를 업데이트하려면 리포지토리 프로젝트에서 변환 개체를 열고 포트를 추가합니다.

## 다중 그룹 변환

변환에 여러 입력 및 출력 그룹이 있을 수 있습니다. 그룹은 수신 또는 전송 데이터의 행을 정의하는 포트 집합입니다.

그룹은 관계형 소스 또는 대상 정의의 테이블과 비슷합니다. 대부분의 변환에는 하나의 입력 그룹과 하나의 출력 그룹이 있습니다. 그러나 일부 변환에는 여러 입력 그룹, 여러 출력 그룹 또는 둘 다 있습니다. 그룹은 변환으로 들어오거나 나가는 데이터 행의 표현입니다.

모든 다중 그룹 변환은 활성 변환입니다. 여러 활성 변환 또는 활성 및 수동 변환을 동일한 다운스트림 변환 또는 변환 입력 그룹에 연결할 수 없습니다.

일부 다중 입력 그룹 변환은 통합 서비스가 다른 입력 그룹의 행을 기다리는 동안 통합 서비스가 입력 그룹에서 데이터를 차단하도록 요구합니다. 차단 변환은 수신 데이터를 차단하는 다중 입력 그룹 변환입니다. 다음 변환은 차단 변환입니다.

- 입력이 차단할 수 있음 속성이 활성화된 사용자 지정 변환
- 정렬되지 않은 입력에 구성된 조이너 변환

매핑을 저장하거나 유효성을 검증하는 경우 활성 또는 차단 변환이 포함된 일부 매핑이 유효하지 않을 수 있습니다.

## 다중 그룹 변환에 대한 규칙 및 지침

매핑에서 변환을 연결하는 경우 다중 그룹 변환 연결에 대한 몇 가지 규칙 및 지침을 고려해야 합니다.

다중 그룹 변환에 대해 다음 규칙 및 지침을 고려하십시오.

- 하나의 그룹을 하나의 변환 또는 대상에 연결할 수 있습니다.
- 그룹에 있는 하나 이상의 출력 포트를 다중 변환 또는 대상에 연결할 수 있습니다.
- 한 변환의 다중 출력 그룹의 필드를 다른 변환의 동일한 입력 그룹에 연결할 수 없습니다.
- 소스와 변환 사이의 각 변환이 수동 변환이 아닌 한, 여러 변환에 있는 다중 출력 그룹의 필드를 다른 변환의 동일한 입력 그룹에 연결할 수 없습니다.
- 다른 변환이 차단 변환이 아닌 한, 한 변환의 다중 출력 그룹의 필드를 다른 변환의 동일한 입력 그룹에 연결할 수 없습니다.
- 그룹이 노멀라이저 변환에 있지 않는 한, 출력 필드를 동일한 입력 그룹의 다중 입력 필드에 연결할 수 없습니다.

## 변환의 식

일부 변환의 경우 **식 편집기**에서 식을 입력할 수 있습니다. 식은 데이터를 수정하거나 데이터가 조건과 일치하는지 여부를 테스트합니다.

변환 언어 함수를 사용하는 식을 작성합니다. 변환 언어 함수는 데이터를 변환하는 SQL과 비슷한 함수입니다.

입력 또는 입력/출력 포트의 데이터 값을 사용하는 식을 포트에 입력합니다. 예를 들어 모든 직원의 급여가 들어 있는 입력 포트 `IN_SALARY`를 포함하는 변환이 있을 수 있습니다. 나중에 매핑에서 `IN_SALARY` 열의 값을 사용할 수도 있습니다. 또한 변환을 사용하여 총 급여와 평균 급여도 계산할 수 있습니다. Developer 도구는 각 계산 값에 대해 별도의 출력 포트를 작성하도록 요구합니다.

다음 표에는 식을 입력할 수 있는 변환이 나열되어 있습니다.

변환	식	반환 값
집계	변환을 통해 전달되는 모든 데이터를 기반으로 집계 계산을 수행합니다. 또는 집계 계산에서 레코드에 대한 필터를 지정하여 특정 유형의 레코드를 제외할 수 있습니다. 예를 들어 이 변환을 사용하여 지점에 있는 모든 직원의 총 수와 평균 급여를 찾을 수 있습니다.	포트에 대한 집계 계산의 결과입니다.
식	단일 행 내의 값을 기반으로 계산을 수행합니다. 예를 들어 특정 품목의 가격 및 수량을 기반으로 주문에서 해당 품목의 총 구입 가격을 계산할 수 있습니다.	포트에 대한 행 수준 계산의 결과입니다.
필터	이 변환을 통해 전달되는 행을 필터링하는 데 사용되는 조건을 지정합니다. 예를 들어 미납금이 있는 고객에 대해 고객 데이터를 <code>BAD_DEBT</code> 테이블에 기록하려는 경우 필터 변환을 사용하여 고객 데이터를 필터링할 수 있습니다.	행이 지정된 조건을 만족하는지 여부에 따라 <code>TRUE</code> 또는 <code>FALSE</code> 입니다. 데이터 통합 서비스는 이 변환을 통해 <code>TRUE</code> 를 반환하는 행을 전달합니다. 변환은 변환을 통과하는 각 행에 이 값을 적용합니다.

변환	식	반환 값
조이너	정렬되지 않은 소스 데이터를 조인하는 데 사용되는 고급 조건을 지정합니다. 예를 들어 성 및 이름 마스터 포트를 연결한 후 전체 이름 정보 포트와 일치시킬 수 있습니다.	행이 지정된 조건을 만족하는지 여부에 따라 TRUE 또는 FALSE입니다. 선택된 조인 유형에 따라 데이터 통합 서비스는 행을 결과 집합에 추가하거나 행을 삭제합니다.
순위	순위에 포함되는 행에 대한 조건을 설정합니다. 예를 들어 회사에 고용된 상위 10명의 판매자에 대한 순위를 지정할 수 있습니다.	포트에 대한 조건 또는 계산의 결과입니다.
라우터	그룹 식에 따라 데이터를 여러 변환으로 라우팅합니다. 예를 들어 이 변환을 사용하여 세 가지 다른 급여 수준에서 직원의 급여를 비교할 수 있습니다. 라우터 변환에서 세 개의 그룹을 작성하여 이 작업을 수행할 수 있습니다. 예를 들어 각 급여 범위에 대한 그룹 식을 한 개 작성합니다.	행이 지정된 그룹 식을 만족하는지 여부에 따라 TRUE 또는 FALSE입니다. 데이터 통합 서비스는 TRUE를 반환하는 행을 이 변환의 각 사용자 정의 그룹을 통해 전달합니다. FALSE를 반환하는 행은 기본 그룹을 통해 전달됩니다.
업데이트 전략	업데이트, 삽입, 삭제 또는 거부할 행에 플래그를 지정합니다. 사용자가 적용하는 조건에 따라 대상에 대한 업데이트를 제어하려면 이 변환을 사용합니다. 예를 들어 메일 주소가 변경된 경우 업데이트 전략 변환을 사용하여 모든 고객 행에 업데이트 플래그를 지정할 수 있습니다. 또는 회사에서 더 이상 근무하지 않는 사람에 대해 모든 직원 행에 거부 플래그를 지정할 수 있습니다.	업데이트, 삽입, 삭제 또는 거부를 나타내는 숫자 코드입니다. 변환은 변환을 통과하는 각 행에 이 값을 적용합니다.

## 식 편집기

식 편집기를 사용하여 SQL과 유사한 문을 작성할 수 있습니다.

식을 수동으로 입력하거나 가리키고 클릭 방법을 사용할 수 있습니다. 식을 작성할 때 실수를 최소화할 수 있도록 가리키고 클릭 인터페이스를 통해 함수, 포트, 변수 및 연산자를 선택하십시오. 식에 포함할 수 있는 최대 문자 수는 32,767자입니다.

## 식의 포트 이름

식에 변환 포트 이름을 입력할 수 있습니다.

연결된 변환의 경우 식에 포트 이름을 사용하면 변환에서 포트 이름을 변경할 때 Developer 도구가 해당 식을 업데이트합니다. 예를 들어 Date\_Promised 및 Date\_Delivered의 두 날짜 간의 차이를 결정하는 식을 작성할 수 있습니다. Date\_Promised 포트 이름을 Due\_Date로 변경하면 Developer 도구가 식에서 Date\_Promised 포트 이름을 Due\_Date로 변경합니다.

**참고:** 매핑에서 이 포트를 사용하는 다른 재사용 불가능한 변환에 이름 Due\_Date를 전달할 수 있습니다.

## 포트에 식 추가

출력 포트에 식을 추가할 수 있습니다.

1. 변환에서 포트를 선택하고 **식 편집기**를 엽니다.
2. 식을 입력합니다. 함수 탭과 포트 탭 및 연산자 키를 사용합니다.



**참고:** 이스케이프 문자는 식에서 사용할 수 없습니다. 식에 이스케이프 문자를 포함하는 경우 **Developer tool**에서 구문 분석 오류가 표시될 수 있습니다.

3. 필요한 경우 식에 설명을 추가합니다.  
설명 표시기 `--` 또는 `//`를 사용합니다.
4. 유효성 검사 단추를 클릭하여 식의 유효성을 검사합니다.
5. **확인**을 클릭합니다.
6. 식이 유효하지 않으면 유효성 검사 오류를 수정하고 다시 식의 유효성을 검사합니다.
7. 식이 유효하면 **확인**을 클릭하여 **식 편집기**를 닫습니다.

## 식의 설명

식에 대해 알려 주는 설명을 식에 추가하거나 식에 대한 비즈니스 설명서에 액세스할 수 있는 유효한 URL을 지정할 수 있습니다.

식 내에서 설명을 추가하려면 `--` 또는 `//` 설명 표시기를 사용합니다.

## 식 유효성 검사

매핑을 실행하거나 맵렛 출력을 미리 보려면 식의 유효성을 검사해야 합니다.

**식 편집기**에서 유효성 검사 단추를 사용하여 식의 유효성을 검사합니다. 식의 유효성을 검사하지 않으면 **식 편집기**를 닫을 때 **Developer** 도구가 식의 유효성을 검사합니다. 식이 유효하지 않으면 **Developer** 도구가 경고를 표시합니다. 유효하지 않은 식을 저장할 수도 있고 수정할 수도 있습니다.

## 식 테스트

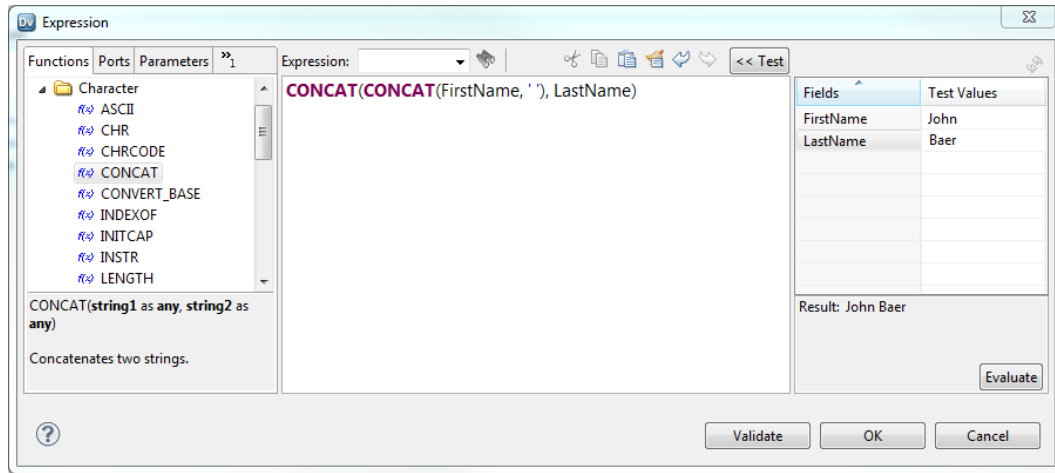
식 편집기에서 구성하는 일부 식을 테스트할 수 있습니다. 식을 테스트하는 경우 샘플 데이터를 입력한 다음 식을 평가합니다.

식을 구성할 때 다음 방법으로 식을 테스트할 수 있습니다.

- 식 변환의 출력 또는 변수 포트에서
- 변환을 매핑에 추가한 후 식 변환의 매핑 출력 보기에서

예를 들어 이름, 공백, 성을 연결하는 식을 구성한 후 포트에 대한 샘플 데이터를 입력한 다음 식을 평가하여 결과를 확인할 수 있습니다.

다음 이미지는 샘플 이름 및 성을 연결하는 식의 결과를 보여 줍니다.



## 샘플 데이터에 대한 날짜 형식 문자열

날짜/시간 또는 시간대가 포함된 타임스탬프 데이터 유형인 포트를 사용하는 식을 테스트하는 경우 필수 날짜 형식 문자열을 사용하여 포트에 대한 샘플 데이터를 입력해야 합니다.

날짜/시간 데이터 유형인 포트에 대해 샘플 데이터를 입력하려면 MM/DD/YYYY HH24:MI:SS 형식을 사용합니다. 식을 평가하면 식에서 지정한 형식을 사용하여 식 편집기에 결과가 표시됩니다. 식에서 형식 문자열을 생략한 경우 동일한 형식 MM/DD/YYYY HH24:MI:SS를 사용하여 식 편집기에 결과가 표시됩니다.

시간대가 포함된 타임스탬프 데이터 유형인 포트에 대한 샘플 데이터를 입력하려면 MM/DD/YYYY HH24:MI:SS TZR 형식을 사용합니다. 식을 평가하면 YYYY-MM-DD HH24:MI:SS.NS TZR 형식을 사용하여 식 편집기에 결과가 표시됩니다.

## 데이터 유형 변환

여러 식 및 집계 함수가 입력 데이터와 다른 데이터 유형의 데이터를 생성할 수 있습니다.

예를 들어 전체 자릿수가 18자리인 두 개의 10진수를 곱하면 결과 데이터 유형은 전체 자릿수가 28자리인 10진수일 수 있습니다.

전체 자릿수가 38자리인 10진수 입력 데이터 유형의 경우 특정 연산의 결과는 결과 데이터 유형에 맞지 않을 수 있는 데이터를 생성할 수 있습니다. 따라서 오버플로우 예외가 발생할 수 있습니다.

다음 함수에는 입력 데이터 유형과 비교 시 증가한 데이터 크기를 포함하기 위해 데이터 유형 변환이 필요할 수 있습니다.

- avg
- cume
- divide
- median
- movingavg
- movingsum
- multiply
- Percentile
- Sum

예를 들어 입력 데이터가 정수 데이터 유형이고 곱하기 연산을 사용하는 경우 결과 데이터 유형은 **Bigint** 데이터 유형일 수 있습니다. 마찬가지로 입력 데이터 유형이 전체 자릿수가 **18**자리인 **10**진수 데이터 유형인 경우 곱하기 연산의 결과는 대형 및 전체 자릿수가 **28**자리인 **10**진수 데이터 유형 내일 수 있습니다.

## 로컬 변수

집계, 식 및 순위 변환에서 로컬 변수를 사용하여 성능을 개선할 수 있습니다. 식에서 변수를 참조하거나 변수를 사용하여 임시로 데이터를 저장할 수 있습니다.

변수를 사용하여 다음 태스크를 완료할 수 있습니다.

- 데이터를 임시로 저장합니다.
- 복잡한 식을 간소화합니다.
- 이전 행의 값을 저장합니다.
- 저장 프로시저의 여러 반환 값을 캡처합니다.
- 값을 비교합니다.
- 연결되지 않은 조희 변환의 결과를 저장합니다.

## 임시로 데이터 저장 및 복잡한 식 간소화

동일한 변환에서 관련 식을 여러 번 입력하는 경우 변수를 사용하면 성능이 향상됩니다. 동일한 식 구성 요소를 변환에서 여러 번 구문 분석하고 유효성 검사를 수행하는 대신 구성 요소를 변수로 정의할 수 있습니다.

예를 들어 집계 변환이 합계 및 평균을 계산하기 전에 동일한 필터 조건을 사용하는 경우 이 조건을 변수로 정의한 다음 두 집계 계산에서 조건을 재사용할 수 있습니다.

복잡한 식을 간소화할 수 있습니다. 집계에서 여러 식에 동일한 계산이 포함된 경우 계산 결과를 저장하기 위한 변수를 작성하면 성능을 향상시킬 수 있습니다.

예를 들어 다음 식을 작성하여 동일한 데이터로 평균 급여와 총 급여를 모두 찾을 수 있습니다.

```
AVG( SALARY, ( ( JOB_STATUS = 'Full-time' ) AND ( OFFICE_ID = 1000 ) ) )
SUM( SALARY, ( ( JOB_STATUS = 'Full-time' ) AND ( OFFICE_ID = 1000 ) ) )
```

두 계산에 대해 동일한 인수를 입력하는 대신 이 계산에서 각 조건에 대해 변수 포트를 작성한 다음 식을 변경하여 변수를 사용할 수 있습니다.

다음 테이블은 변수를 사용하여 복잡한 식을 간소화하고 임시로 데이터를 저장하는 방법을 보여줍니다.

포트	값
V_CONDITION1	JOB_STATUS = 'Full-time'
V_CONDITION2	OFFICE_ID = 1000
AVG_SALARY	AVG(SALARY, (V_CONDITION1 AND V_CONDITION2) )
SUM_SALARY	SUM(SALARY, (V_CONDITION1 AND V_CONDITION2) )

## 행의 값 저장

소스 행의 데이터를 저장하도록 변환의 변수를 구성할 수 있습니다. 변환 식에서 변수를 사용할 수 있습니다.

예를 들어 소스 파일에 다음과 같은 행이 있습니다.

```
California
California
California
Hawaii
Hawaii
New Mexico
New Mexico
New Mexico
```

각 행은 상태를 포함합니다. 행 수를 카운트하고 각 상태에 대해 행 카운트를 반환해야 합니다.

```
California,3
Hawaii,2
New Mexico,3
```

상태별로 소스 행을 그룹화하고 각 그룹의 행 수를 카운트하도록 집계 변환을 구성할 수 있습니다. 행 카운트를 저장하도록 집계 변환의 변수를 구성합니다. 다른 변수를 정의하여 이전 행의 상태 이름을 저장합니다.

집계 변환에는 다음 포트가 있습니다.

포트	포트 유형	식	설명
상태	통과	n/a	상태의 이름입니다. 소스 행은 상태 이름에 따라 그룹화됩니다. 집계 변환이 각 상태에 대해 하나의 행을 반환합니다.
State_Count	변수	IIF (PREVIOUS_STATE = STATE, STATE_COUNT +1, 1)	현재 State의 행 카운트입니다. 현재 State 열의 값이 Previous_State 열과 동일한 경우 통합 서비스가 State_Count를 증분합니다. 그렇지 않은 경우 State_Count를 1로 재설정합니다.
Previous_State	변수	주	이전 행의 State 열 값입니다. 통합 서비스가 행을 처리할 때 State 값이 Previous_State로 이동합니다.
State_Counter	출력	State_Count	상태에 대해 처리된 집계 변환의 행 수입니다. 통합 서비스는 각 상태에 대해 State_Counter를 한 번 반환합니다.

## 저장 프로시저의 값 캡처

변수를 사용하면 저장 프로시저에서 여러 열의 반환 값을 캡처할 수 있습니다.

## 변수 포트 구성 지침

변환에서 변수 포트를 구성할 경우 다음 요소를 고려하십시오.

- **포트 순서.** 통합 서비스는 포트를 종속성에 따라 평가합니다. 변환의 포트 순서는 평가 순서와 일치해야 합니다. 즉 입력 포트, 변수 포트, 출력 포트 순서입니다.
- **데이터 유형.** 선택한 데이터 유형에 따라 입력한 식의 반환 값이 반영됩니다.
- **변수 초기화.** 통합 서비스는 변수 포트에 초기 값을 설정하므로 변수 포트에서 카운터를 작성할 수 있습니다.

## 포트 순서

통합 서비스는 먼저 입력 포트를 평가합니다. 통합 서비스는 변수를 그 다음에 평가하고 출력 포트를 마지막으로 평가합니다.

통합 서비스는 다음 순서로 포트를 평가합니다.

1. **입력 포트.** 통합 서비스는 입력 포트가 다른 포트에 종속되지 않기 때문에 첫 번째로 모든 입력 포트를 평가합니다. 따라서 원하는 순서로 입력 포트를 작성할 수 있습니다. 통합 서비스는 입력 포트가 다른 포트를 참조하지 않기 때문에 입력 포트의 순서를 지정하지 않습니다.
2. **변수 포트.** 변수 포트는 입력 포트 및 변수 포트를 참조할 수 있지만 출력 포트는 참조할 수 없습니다. 변수 포트가 입력 포트를 참조할 수 있기 때문에 통합 서비스는 입력 포트 다음에 변수 포트를 평가합니다. 변수가 다른 변수를 참조할 수 있으므로 변수 포트의 표시 순서는 통합 서비스가 각 변수를 평가하는 순서와 동일합니다.  
예를 들어 빌딩의 원래 값을 계산한 다음 감가상각에 대해 적용하는 경우 원래 값 계산을 변수 포트에 작성할 수 있습니다. 이 변수 포트는 감가상각에 적용하는 포트 전에 나타나야 합니다.
3. **출력 포트.** 출력 포트가 입력 포트 및 변수 포트를 참조할 수 있기 때문에 통합 서비스는 출력 포트를 마지막으로 평가합니다. 출력 포트는 다른 출력 포트를 참조할 수 없기 때문에 출력 포트의 표시 순서는 중요하지 않습니다. 출력 포트가 포트의 목록 아래에 표시되는지 확인합니다.

## 데이터 유형

포트를 변수로 구성하는 경우 모든 식 또는 조건을 포트에 입력할 수 있습니다. 이 포트에 대한 데이터 유형은 입력한 식의 반환 값에 따라 선택합니다. 변수 포트를 통해 조건을 지정하는 경우 데이터 유형이 숫자인 모든 포트는 TRUE(0이 아닌 경우) 및 FALSE(0인 경우)의 값을 반환합니다.

## 변수 초기화

통합 서비스는 변수의 초기 값을 NULL로 설정하지 않습니다.

통합 서비스는 변수의 초기 값을 설정하기 위해 다음 지침을 사용합니다.

- 숫자 포트의 경우 0
- 문자열 포트의 경우 빈 문자열
- 날짜/시간 포트의 경우 01/01/0001

그러므로 초기 값이 필요한 카운터로 변수를 사용합니다. 예를 들어 다음 식을 사용하여 숫자 변수를 작성할 수 있습니다.

```
VAR1 + 1
```

이 식은 VAR1 포트의 행 수를 카운트합니다. 변수의 초기 값이 NULL로 설정되면 식이 항상 NULL로 평가됩니다. 그래서 초기 값이 0으로 설정되는 것입니다.

## 포트의 기본값

모든 변환은 기본값을 사용하여 입력 Null 값 및 출력 변환 오류에 대한 통합 서비스의 처리 방식을 결정합니다.

입력, 출력 및 입력/출력 포트에는 시스템 기본값이 있으며 이러한 기본값은 사용자 정의 기본값으로 재정의할 수 있습니다. 기본값의 함수는 포트 유형에 따라 다릅니다.

- **입력 포트.** Null 입력 포트에 대한 시스템 기본값은 **NULL**입니다. 기본값은 변환에서 공백으로 표시됩니다. 입력 값이 **NULL**인 경우 통합 서비스가 **NULL**로 유지합니다.
- **출력 포트.** 출력 변환 오류에 대한 시스템 기본값은 **ERROR**입니다. 기본값은 변환에서 **ERROR**(“변환 오류”)로 표시됩니다. 변환 오류가 발생하면 통합 서비스가행을 건너뛵니다. 통합 서비스는 **ERROR** 함수가 건너뛰는 모든 입력 행을 로그 파일에 기록합니다.  
다음 오류는 변환 오류입니다.
  - 숫자를 날짜 함수에 전달하는 것과 같은 데이터 변환 오류.
  - 0으로 나누는 것과 같은 식 평가 오류.
  - **ERROR** 함수에 대한 호출.
- **통과 포트.** Null 입력에 대한 시스템 기본값은 입력 포트와 동일한 **NULL**입니다. 시스템 기본값은 변환에서 공백으로 표시됩니다. 출력 변환 오류에 대한 기본값은 출력 포트와 동일합니다. 출력 변환 오류에 대한 기본값은 변환에 표시되지 않습니다.

**참고:** Java 변환은 Java 변환 포트 유형에 따라 **PowerCenter®** 데이터 유형을 **Java** 데이터 유형으로 변환합니다. Null 입력에 대한 기본값은 Java 데이터 유형에 따라 다릅니다.

다음 표에는 연결된 변환의 포트에 대한 시스템 기본값이 표시됩니다.

포트 유형	기본값	통합 서비스 동작	지원되는 사용자 정의 기본값
입력, 통과	NULL	통합 서비스가 모든 입력 Null 값을 NULL로 전달합니다.	입력, 입력/출력
출력, 통과	ERROR	통합 서비스가 출력 포트 변환 오류에 대해 <b>ERROR</b> 함수를 호출합니다. 통합 서비스는 오류가 있는 행을 건너뛰고 입력 데이터 및 오류 메시지를 로그 파일에 기록합니다.	출력

변수 포트는 기본값을 지원하지 않습니다. 통합 서비스는 데이터 유형에 따라 변수 포트를 초기화합니다.

일부 기본값을 재정의하여 Null 입력 값 및 출력 변환 오류 발생 시의 통합 서비스 동작을 변경할 수 있습니다.

## 사용자 정의 기본값

연결된 변환 내의 지원되는 입력, 통과 및 출력 포트에 대해 사용자 정의 기본값을 사용하여 시스템 기본값을 재정의할 수 있습니다.

포트의 시스템 기본값을 재정의하려면 다음 규칙 및 지침을 사용합니다.

- **입력 포트.** 통합 서비스가 **NULL** 값을 **NULL**로 처리하지 않게 하려는 경우 입력 포트에 대해 사용자 정의 기본값을 입력할 수 있습니다. 입력 포트에 **NULL**이 전달되면 통합 서비스가 **NULL**을 기본값으로 바꿉니다.
- **출력 포트.** 통합 서비스가행을 건너뛰지 않게 하거나 건너뛴 행과 함께 특정 메시지를 로그에 쓰게 하려는 경우 출력 포트에 사용자 정의 기본값을 입력할 수 있습니다. 출력 포트에 기본값이 정의된 경우 출력 포트에서 변환 오류가 발생하면 통합 서비스가행을 기본값으로 바꿉니다.
- **통과 포트.** 통합 서비스가 **NULL** 값을 **NULL**로 처리하지 않게 하려는 경우 통과 포트에 대해 사용자 정의 기본값을 입력할 수 있습니다. 통과 포트의 출력 변환 오류에 대해서는 사용자 정의 기본값을 입력할 수 없습니다.

**참고:** 통합 서비스는 연결되지 않은 변환의 사용자 정의 기본값을 무시합니다. 예를 들어 조회 또는 저장 프로시저 변환을 식을 통해 호출하는 경우 통합 서비스는 사용자 정의 기본값을 무시하고 시스템 기본값을 적용합니다.

다음 옵션을 사용하여 사용자 정의 기본값을 입력합니다.

- **상수 값.** NULL을 포함하여 상수(숫자 또는 텍스트)를 사용합니다.
- **상수 식.** 상수 매개 변수를 사용하는 변환 함수를 포함할 수 있습니다.
- **ERROR.** 변환 오류를 생성합니다. 세션 로그 또는 행 오류 로그에 행 및 메시지를 씁니다.
- **ERROR.** 변환 오류를 생성합니다. 매핑 로그 또는 행 오류 로그에 행 및 메시지를 씁니다.
- **ABORT.** 세션을 중단합니다.
- **ABORT.** 매핑을 중단합니다.

## 상수 값

모든 상수 값을 기본값으로 입력할 수 있습니다. 상수 값은 포트의 데이터 유형과 일치해야 합니다.

예를 들어 숫자 포트의 기본값은 숫자 상수여야 합니다. 일부 상수 값은 다음과 같습니다.

```
0
9999
NULL
'Unknown Value'
'Null input data'
```

## 상수 식

상수 식은 집계 함수를 제외한 변환 함수를 사용하여 상수 식을 작성하는 식입니다. 상수 식에서 입력, 입력/출력 또는 변수 포트의 값을 사용할 수 없습니다.

일부 유효한 상수 식에는 다음이 포함됩니다.

```
500 * 1.75
TO_DATE('January 1, 1998, 12:05 AM', 'MONTH DD, YYYY, HH:MI AM')
ERROR('Null not allowed')
ABORT('Null not allowed')
SESSSTARTIME
```

통합 서비스가 세션을 초기화할 경우 전체 매핑에 대한 기본값을 할당하기 때문에 식 내에서 포트의 값을 사용할 수 없습니다.

통합 서비스가 매핑을 초기화할 경우 전체 매핑에 대한 기본값을 할당하기 때문에 식 내에서 포트의 값을 사용할 수 없습니다.

다음 예제에서는 포트의 값을 사용하기 때문에 올바르지 않습니다.

```
AVG(IN_SALARY)
IN_PRICE * IN_QUANTITY
:LKP(LKP_DATES, DATE_SHIPPED)
```

**참고:** 기본값 식에서 저장 프로시저 또는 조회 테이블을 호출할 수 없습니다.

## ERROR 및 ABORT 함수

ERROR 및 ABORT 함수를 입력 및 출력 포트의 기본값으로 사용하고 입력/출력 포트의 입력 값으로 사용할 수 있습니다. 통합 서비스가 ERROR 함수를 만나면 행을 건너뜁니다. ABORT 함수를 만나면 세션을 중단합니다.

ERROR 및 ABORT 함수를 입력 및 출력 포트의 기본값으로 사용하고 입력/출력 포트의 입력 값으로 사용할 수 있습니다. 통합 서비스가 ERROR 함수를 만나면 행을 건너뜁니다. ABORT 함수를 만나면 매핑을 중단합니다.

## 사용자 정의 기본 입력값

통합 서비스가 NULL 값을 NULL로 처리하지 않게 하려는 경우 사용자 정의 기본 입력 값을 입력할 수 있습니다.

NULL 값을 재정의하려면 다음 태스크 중 하나를 완료하십시오.

- NULL 값을 상수 값 또는 상수 식으로 바꿉니다.
- ERROR 함수를 사용하여 NULL 값을 건너뜁니다.
- ABORT 함수를 사용하여 매핑을 중단합니다.
- ABORT 함수를 사용하여 세션을 중단합니다.

다음 테이블에는 통합 서비스가 입력 및 입력/출력 포트에 대해 NULL 입력을 처리하는 방식이 요약되어 있습니다.

기본값	기본값 유형	설명
NULL(빈 상태로 표시)	시스템	통합 서비스가 NULL을 전달합니다.
상수 또는 상수 식	사용자 정의	통합 서비스가 NULL 값을 상수 또는 상수 식 값으로 바꿉니다.
오류	사용자 정의	통합 서비스가 변환 오류로 처리합니다. <ul style="list-style-type: none"> <li>- 변환 오류 카운트를 1씩 늘립니다.</li> <li>- 행을 건너 뛰고 오류 메시지를 로그 파일 또는 행 오류 로그에 씁니다.</li> </ul> 통합 서비스가 거부 파일에 행을 쓰지 않습니다.
ABORT	사용자 정의	통합 서비스에서 Null 입력 값이 발생하면 세션이 중단됩니다. 통합 서비스가 오류 카운트를 늘리거나 행을 거부 파일에 쓰지 않습니다. 통합 서비스에서 Null 입력 값이 발생하면 매핑이 중단됩니다. 통합 서비스가 오류 카운트를 늘리거나 행을 거부 파일에 쓰지 않습니다.

## Null 값 바꾸기

상수 값 또는 식을 사용하여 포트의 Null 값을 지정된 값으로 대체할 수 있습니다.

예를 들어 입력 문자열 포트의 이름이 DEPT\_NAME이고 Null 값을 문자열 'UNKNOWN DEPT'로 바꾸려는 경우 기본값을 'UNKNOWN DEPT'로 설정할 수 있습니다. 변환에 따라 통합 서비스가 'UNKNOWN DEPT'를 변환 내의 변수 또는 식으로 전달하거나 데이터 흐름의 다음 변환으로 전달합니다.

예를 들어 통합 서비스가 포트의 모든 Null 값을 문자열 'UNKNOWN DEPT'로 바꿉니다.

DEPT_NAME	REPLACED VALUE
Housewares	Housewares
NULL	UNKNOWN DEPT
Produce	Produce

## Null 레코드 건너뛰기

null 값이 변환으로 전달되지 않도록 하려면 ERROR 함수를 기본값으로 사용합니다. DEPT\_NAME의 입력 값이 NULL일 때 행을 건너뛰려는 경우를 예로 들어 보겠습니다. 다음 식을 기본값으로 사용할 수 있습니다.

```
ERROR('Error. DEPT is NULL')
```

ERROR 함수를 기본값으로 사용할 때 통합 서비스는 null 값이 포함된 행을 건너뜁니다. 통합 서비스는 ERROR 함수가 건너뀀 모든 행을 로그 파일에 씁니다. 그러나 이러한 행을 거부 파일에 쓰지는 않습니다.

DEPT_NAME	RETURN VALUE
Housewares	Housewares
NULL	'Error. DEPT is NULL' (Row is skipped)
Produce	Produce



다음 로그는 통합 서비스가 null 값이 포함된 행을 건너뛰는 경우를 보여 줍니다.

```
TE_11019 Port [DEPT_NAME]: Default value is: ERROR(<<Transformation Error>> [error]: Error. DEPT is NULL
... error('Error. DEPT is NULL')
).
CMN_1053 EXPTRANS: : ERROR: NULL input column DEPT_NAME: Current Input data:
CMN_1053 Input row from SRCTRANS: Rowdata: ( RowType=4 Src Rowid=2 Targ Rowid=2
DEPT_ID (DEPT_ID:Int:): "2"
DEPT_NAME (DEPT_NAME:Char.25:): "NULL"
MANAGER_ID (MANAGER_ID:Int:): "1"
)
```

## 매핑 중단

통합 서비스가 Null 입력 값을 발견할 때 매핑을 중단하려면 **ABORT** 함수를 사용합니다.

## 기본값 유효성 검사

기본값을 입력할 때 기본값의 유효성을 검사할 수 있습니다. 디자이너에는 유효한 기본값을 보장할 수 있는 유효성 검사 단추가 있습니다. 기본값이 유효한지 표시하는 메시지가 나타납니다.

매핑을 저장할 때에도 디자이너에서 기본값 유효성을 검사합니다. 잘못된 기본값을 입력할 경우 디자이너에서 매핑을 올바르게 않은 매핑으로 표시합니다.

사용자가 기본값을 입력할 때 **Developer tool**에서 기본값의 유효성을 검사합니다.

사용자가 매핑을 저장할 때 **Developer tool**에서 기본값 유효성을 검사합니다. 올바르게 않은 기본값을 입력할 경우 **Developer tool**에서 매핑을 올바르게 않은 매핑으로 표시합니다.

## 사용자 정의 기본 출력 값

사용자 정의 기본값을 생성하여 출력 포트에 대한 시스템 기본값을 재정의할 수 있습니다.

통합 서비스에서 오류가 발생한 행을 건너뛰지 않도록 하거나 건너뀀 행이 포함된 특정 메시지를 로그에 쓰도록 하려는 경우 출력 포트에 대해 사용자 정의 기본값을 입력할 수 있습니다. 통합 서비스에서 출력 변환 오류가 발생하면 기본값을 입력하여 다음 함수를 완료할 수 있습니다.

- 오류는 상수 값이나 상수 식으로 바꾸십시오. 통합 서비스에서 행을 건너뛰지 않습니다.
- **ABORT** 함수를 사용하여 세션을 중단합니다.
- **ABORT** 함수를 사용하여 매핑을 중단합니다.
- 변환 오류에 대한 구체적인 메시지를 로그에 씁니다.

입력/출력 포트에 대해서는 사용자 정의 기본 출력 값을 입력할 수 없습니다.

다음 테이블에는 통합 서비스가 변환의 출력 포트 변환 오류와 기본값을 처리하는 방법이 요약되어 있습니다.

기본값	기본값 유형	설명
변환 오류	시스템	기본값을 재정의하지 않은 상태에서 변환 오류가 발생하면 통합 서비스는 다음 태스크를 수행합니다. - 변환 오류 카운트를 1씩 늘립니다. - 행을 건너뛰고 세션 구성에 따라 오류 및 입력 행을 세션 로그 파일이나 행 오류 로그에 씁니다. 통합 서비스가 행을 거부 파일에 쓰지 않습니다.
상수 또는 상수 식	사용자 정의	통합 서비스가 오류를 기본값으로 바꿉니다. 통합 서비스가 오류 수를 늘리거나 메시지를 로그에 쓰지 않습니다.
ABORT	사용자 정의	세션이 중단되며 통합 서비스가 메시지를 세션 로그에 씁니다. 매핑이 중단되며 통합 서비스가 메시지를 로그에 씁니다. 통합 서비스가 오류 카운트를 늘리거나 행을 거부 파일에 쓰지 않습니다.

## 대체 오류

변환 오류 발생 시 통합 서비스가 행을 건너뛰지 않도록 하려면 출력 포트의 기본값으로 상수 또는 상수 식을 사용합니다.

예를 들어 `NET_SALARY`라는 숫자 출력 포트가 있는 경우 변환 오류 발생 시 상수 값 '9999'를 사용하려면 `NET_SALARY` 포트에 기본값 9999를 할당합니다. `NET_SALARY`의 값을 계산하는 동안 0으로 나누는 등의 변환 오류가 발생하면 통합 서비스는 기본값인 9999를 사용합니다.

## 매핑 중단

통합 서비스에 Null 입력 값이 발생할 경우 세션을 중단하려면 `ABORT` 함수를 사용하십시오.

## 매핑 로그에 메시지 쓰기

통합 서비스가 매핑 로그에 건너뛴 행에 대해 특정 메시지를 쓰게 하려는 경우 출력 포트에서 사용자 정의 기본값을 구성하면 됩니다. 시스템 기본값은 `ERROR`('변환 오류')이며 통합 서비스가 로그에 건너뛴 행과 함께 '변환 오류' 메시지를 씁니다. 다른 메시지를 쓰려는 경우 '변환 오류'를 바꿀 수 있습니다.

## 출력 포트 식의 ERROR 함수

`ERROR` 함수를 사용하는 식을 입력할 경우 출력 포트의 사용자 정의 기본값이 식의 `ERROR` 함수를 재정의할 수 있습니다.

예를 들어, 통합 서비스에 오류가 발생할 경우 'Negative Sale' 값을 사용하도록 지시하는 다음 식을 입력합니다.

```
IIF( TOTAL_SALES>0, TOTAL_SALES, ERROR ('Negative Sale'))
```

다음 예제에서는 사용자 정의 기본값이 식의 `ERROR` 함수를 재정의하는 방법을 보여 줍니다.

- **상수 값 또는 식.** 상수 값 또는 식이 출력 포트 식의 `ERROR` 함수를 재정의합니다.

예를 들어, '0'을 기본값으로 입력할 경우 통합 서비스에서 출력 포트 식의 `ERROR` 함수를 재정의합니다. 통합 서비스에서 오류가 발생할 경우 값 0을 전달합니다. 행을 건너뛰거나 로그에 'Negative Sale'을 기록하지 않습니다.

- **ABORT.** `ABORT` 함수가 출력 포트 식의 `ERROR` 함수를 재정의합니다.

ABORT 함수를 기본값으로 사용할 경우 변환 오류가 발생하면 통합 서비스가 중단됩니다. ABORT 함수가 출력 포트 식의 ERROR 함수를 재정의합니다.

- **ERROR.** ERROR 함수를 기본값으로 사용할 경우 통합 서비스의 로그에 다음 정보가 포함됩니다.

- 기본값의 오류 메시지
- 출력 포트 식의 ERROR 함수에 표시된 오류 메시지
- 건너뀀 행

예를 들어, 다음 ERROR 함수로 기본값을 재정의할 수 있습니다.

```
ERROR('No default value')
```

통합 서비스에서 행을 건너뛰고, 두 오류 메시지를 로그에 포함합니다.

```
TE_7007 Transformation Evaluation Error; current row skipped...
TE_7007 [<<Transformation Error>> [error]: Negative Sale
... error('Negative Sale')
]
Sun Sep 20 13:57:28 1998
TE_11019 Port [OUT_SALES]: Default value is: ERROR(<<Transformation Error>> [error]: No default value
... error('No default value')
```

## 기본값에 대한 일반 규칙

기본값을 작성할 경우 다음 규칙 및 지침을 사용하십시오.

- 기본값은 NULL, 상수 값, 상수 식, ERROR 함수 또는 ABORT 함수여야 합니다.
- 입력/출력 포트의 경우 통합 서비스에서 기본값을 사용하여 Null 입력 값을 처리합니다. 입력/출력 포트의 출력 기본값은 항상 ERROR('변환 오류')입니다.
- 변수 포트에서는 기본값을 사용하지 않습니다.
- 집계 및 순위 변환에서 그룹 기준 포트에 기본값을 할당할 수 있습니다.
- 모든 변환의 일부 포트 유형에서만 사용자 정의 기본값을 허용합니다. 포트에서 사용자 정의 기본값을 허용하지 않을 경우 기본값 필드가 비활성화됩니다.
- 일부 변환에서만 사용자 정의 기본값을 허용합니다.
- 변환이 매핑 데이터 흐름에 연결되지 않은 경우 통합 서비스에서 사용자 정의 기본값을 무시합니다.
- 입력 포트가 연결되지 않은 경우 해당 값은 NULL로 추정되고 통합 서비스가 해당 입력 포트에 대한 기본값을 사용합니다.
- 입력 포트 기본값에 ABORT 함수가 포함되고 입력 값이 NULL일 경우 통합 서비스가 즉시 세션을 중지합니다. ABORT 함수를 기본값으로 사용하여 Null 입력 값을 제한합니다. 입력 포트의 첫 번째 Null 값에서 세션을 중지합니다.
- 출력 포트 기본값에 ABORT 함수가 포함되고 해당 포트에 대한 변환 오류가 발생할 경우, 세션이 즉시 중지됩니다. 변환 오류에 대해 엄격한 규칙을 적용하려면 ABORT 함수를 기본값으로 사용하십시오. 이 포트에 대한 첫 번째 변환 오류에서 세션을 중지합니다.
- 입력 포트 기본값에 ABORT 함수가 포함되고 입력 값이 NULL일 경우 통합 서비스가 즉시 매핑을 중지합니다. ABORT 함수를 기본값으로 사용하여 Null 입력 값을 제한합니다. 입력 포트의 첫 번째 Null 값에서 매핑을 중지합니다.
- 출력 포트 기본값에 ABORT 함수가 포함되고 해당 포트에 대한 변환 오류가 발생할 경우, 매핑이 즉시 중지됩니다. 변환 오류에 대해 엄격한 규칙을 적용하려면 ABORT 함수를 기본값으로 사용하십시오. 이 포트에 대한 첫 번째 변환 오류에서 매핑을 중지합니다.
- ABORT 함수, 상수 값 및 상수 식이 출력 포트 식에 구성된 ERROR 함수를 재정의합니다.

## 기본값 유효성 검사

기본값을 입력할 때 기본값의 유효성을 검사할 수 있습니다. 디자이너에는 유효한 기본값을 보장할 수 있는 유효성 검사 단추가 있습니다. 기본값이 유효한지 표시하는 메시지가 나타납니다.

매핑을 저장할 때에도 디자이너에서 기본값 유효성을 검사합니다. 잘못된 기본값을 입력할 경우 디자이너에서 매핑을 올바르게 않은 매핑으로 표시합니다.

사용자가 기본값을 입력할 때 **Developer tool**에서 기본값의 유효성을 검사합니다.

사용자가 매핑을 저장할 때 **Developer tool**에서 기본값 유효성을 검사합니다. 올바르게 않은 기본값을 입력할 경우 **Developer tool**에서 매핑을 올바르게 않은 매핑으로 표시합니다.

## 추적 수준

변환을 구성하는 경우 데이터 통합 서비스가 로그에 쓰는 세부 정보 양을 설정할 수 있습니다.

기본적으로 모든 변환의 추적 수준은 보통입니다. 예측한 대로 작동하지 않는 변환 문제를 해결해야 하는 경우 추적 수준을 자세한 정보 표시 설정으로 변경합니다. 로그에 최소 양의 세부 정보를 기록하려는 경우 추적 수준을 간단으로 설정합니다.

고급 탭에서 다음 속성을 구성합니다.

### 추적 수준

변환에 대해 로그에 표시하는 세부 정보 양입니다.

다음 테이블에는 추적 수준에 대해 설명되어 있습니다.

추적 수준	설명
간단	거부된 데이터의 초기화 정보, 오류 메시지, 알림을 기록합니다.
일반	변환 행 오류로 인해 건너뛴 행, 초기화/상태 정보, 발생한 오류를 기록합니다. 매핑 결과를 요약하지만 개별 행 수준이 아닙니다. 기본값은 보통입니다.
자세한 정보 표시 초기화	보통 추적 이외에 추가 초기화 세부 정보, 사용되는 데이터 파일과 인덱스의 이름 및 세부 변환 통계를 기록합니다.
자세한 정보 표시 데이터	자세한 정보 표시 초기화 추적 외에 매핑으로 전달되는 각 행을 기록합니다. 또한 열의 전체 자릿수에 맞게 문자열 데이터를 자르는 위치를 기록하며 세부 변환 통계를 제공합니다. 이 추적 수준을 구성하는 경우 변환이 처리될 때 블록의 모든 행에 대한 행 데이터를 로그에 기록합니다.

## 재사용 가능한 변환

재사용 가능한 변환은 여러 매핑 또는 맵셋에서 사용할 수 있는 변환입니다.

예를 들어 캐나다에서 비즈니스 활동 비용을 분석하기 위해 판매 부가 가치세를 계산하는 식 변환을 작성할 수 있습니다. 동일한 작업을 매번 수행하기 보다 재사용 가능한 변환을 작성할 수 있습니다. 이러한 변환을 매핑에

통합해야 하는 경우 변환의 인스턴스를 매핑에 추가합니다. 변환의 정의를 변경하면 변환의 모든 인스턴스가 변경 내용을 상속합니다.

**Developer** 도구는 재사용 가능한 변환을 사용하는 매핑 또는 맵렛과 별개의 메타데이터로 각 변환을 저장합니다. 재사용 가능한 변환을 프로젝트 또는 폴더로 저장합니다.

재사용 가능한 변환의 인스턴스를 매핑에 추가하는 경우 변환을 변경하면 매핑이 무효화되거나 예상치 않은 데이터가 생성될 수 있습니다.

## 재사용 가능한 변환 인스턴스 및 상속된 변경 내용

재사용 가능한 변환을 매핑 또는 맵렛에 추가하는 경우 변환의 인스턴스를 추가합니다. 변환의 정의는 계속 매핑 또는 맵렛 외부에 존재하지만 변환의 인스턴스는 매핑 또는 맵렛 내부에 나타납니다.

변환을 변경하면 변환의 인스턴스에서 이러한 변경 내용이 반영됩니다. 동일한 변환을 사용하는 모든 매핑에서 변환을 업데이트하는 대신 재사용 가능한 변환을 한 번 업데이트하면 변환의 모든 인스턴스에서 변경 내용을 상속합니다. 인스턴스는 변환의 포트, 식, 속성 및 이름 변경 내용을 상속합니다.

## 재사용 가능한 변환 편집

재사용 가능한 변환을 편집하면 해당 변환의 모든 인스턴스가 변경 내용을 상속합니다. 일부 변경의 경우 재사용 가능한 변환을 사용하는 매핑을 무효화할 수 있습니다.

편집기에서 변환을 열어 재사용 가능한 변환을 편집할 수 있습니다. 매핑에서 변환의 인스턴스를 편집할 수는 없습니다. 하지만 변환 런타임 속성은 편집할 수 있습니다.

재사용 가능한 변환에 대해 다음과 같은 변경을 할 경우 이 변환의 인스턴스를 사용하는 매핑이 유효하지 않게 될 수 있습니다.

- 변환에서 하나 이상의 포트를 삭제하면 매핑을 통한 데이터 흐름의 일부 또는 전부에서 인스턴스의 연결이 끊길 수 있습니다.
- 포트 데이터 유형을 변경하는 경우 해당 포트에서 호환되지 않는 데이터 유형을 사용하는 다른 포트로 데이터를 매핑할 수 없게 됩니다.
- 포트 이름을 변경하는 경우 해당 포트를 참조하는 식이 더 이상 유효하지 않습니다.
- 재사용 가능한 변환에서 유효하지 않은 식을 입력하는 경우 이 변환을 사용하는 매핑이 더 이상 유효하지 않습니다. 데이터 통합 서비스는 유효하지 않은 매핑을 실행할 수 없습니다.

## 재사용 가능 변환의 편집기 보기

편집기의 보기에서 재사용 가능 **Java** 변환에 대한 속성을 정의하고 **Java** 코드를 작성하십시오.

재사용 가능 변환의 경우 다음 보기를 사용할 수 있습니다.

### 개요

변환의 이름 및 설명을 입력하고, 입력 및 출력 포트를 작성 및 구성합니다.

### 고급

변환에 대한 고급 속성을 설정합니다.

# 재사용 불가능 변환

재사용 불가능 변환은 특정 매핑에서 생성하는 변환입니다. 다른 매핑에서는 해당 변환을 사용할 수 없습니다.

여러 변환을 포함하는 매핑을 생성하는 경우를 예로 들어 보겠습니다. 각 변환은 소스 데이터에 대해 계산을 수행합니다. 매핑 끝에서 결과를 처리하기 위한 재사용 불가능 집계 변환을 생성합니다. 재사용 불가능 변환을 생성할 때는 매핑에서 변환 간에 포트를 끌어 입력 포트를 생성할 수 있습니다.

Developer tool은 재사용 불가능 집계 변환을 매핑과 함께 보관하는 메타데이터로 저장합니다.

## 재사용 불가능 변환의 편집기 보기

재사용 불가능 변환의 속성은 편집기 보기에서 정의합니다.

재사용 불가능 변환에 대해 표시되는 보기는 다음과 같습니다.

### 일반

변환의 이름 및 설명을 입력합니다.

### 포트

입력 및 출력 포트를 작성하고 구성합니다.

### 고급

변환에 대한 고급 속성을 설정합니다.

## 변환 작성

재사용 가능한 변환을 작성하여 여러 매핑 또는 맵셋에서 재사용할 수 있습니다. 또는 재사용 불가능한 변환을 작성하여 매핑 또는 맵셋에서 한 번 사용할 수 있습니다.

재사용 가능한 변환을 작성하려면 **파일 > 새로 만들기 > 변환**을 클릭하고 마법사를 완료합니다.

매핑 또는 맵셋에서 재사용 불가능한 변환을 작성하려면 변환 색상표에서 변환을 선택하고 변환을 편집기로 끌어 놓습니다.

일부 변환의 경우 변환을 작성할 때 모드를 선택하거나 추가 구성을 수행해야 합니다. 예를 들어 조회 변환을 작성할 때 조회 소스로 사용할 데이터 개체를 선택해야 합니다.

변환을 작성하고 나면 변환이 편집기에 나타납니다. 일부 변환에는 미리 정의된 포트 및 그룹이 있습니다. 다른 변환의 경우에는 비어 있습니다.

## 제 2 장

# 변환 포트

이 장에 포함된 항목:

- [변환 포트 개요, 59](#)
- [포트 생성, 59](#)
- [포트 구성, 60](#)
- [포트 연결, 60](#)
- [포트 특성 전달, 63](#)
- [Excel에서 포트 복사, 66](#)

## 변환 포트 개요

변환 포트는 변환은 매핑이나 맵셋으로 연결하는 개별 데이터 열입니다. 변환은 입력 포트에서 데이터를 수신하고 출력 포트를 사용하여 데이터를 전송합니다. 입력/출력 포트는 데이터를 수신하여 변경되지 않은 상태로 전달합니다.

모든 입력 개체, 출력 개체, 맵셋 및 변환에는 포트 컬렉션이 있습니다. 입력 개체는 데이터를 제공하므로 출력 포트만 포함합니다. 출력 개체는 데이터를 수신하므로 입력 포트만 포함합니다. 맵셋은 입력 포트와 출력 포트만 포함합니다. 변환은 변환 및 그 적용에 따라 입력, 출력 및 입력/출력 포트의 조합을 포함합니다.

동적 포트는 업스트림 변환에서 하나 이상의 열을 수신할 수 있습니다. 동적 포트는 데이터 흐름을 기반으로 새 열 또는 변경된 열을 수신할 수 있습니다.

매핑에서 변환을 생성한 후 포트를 생성하고 포트 속성을 정의합니다. 포트를 통해 대상 및 다른 변환에 연결하여 매핑을 완료합니다. 포트 특성을 전달하여 변경된 특성을 매핑 전체에서 포트에 전달합니다.

## 포트 생성

일부 변환을 생성하는 경우 모든 포트를 수동으로 생성할 필요가 없습니다. 예를 들어 조회 변환을 생성하고 조회 테이블을 참조할 수 있습니다. 변환 포트를 보면 변환에 참조한 테이블의 각 열에 대한 출력 포트가 있는지 확인할 수 있습니다. 해당 포트는 정의할 필요가 없습니다.

다음과 같은 방법으로 포트를 생성합니다.

- **다른 변환에서 포트 끌기.** 다른 변환에서 포트를 끌어오면 디자이너가 동일한 속성을 포함한 포트를 생성하고 두 포트를 연결합니다. 레이아웃 > 열 복사를 클릭하여 포트 복사를 활성화합니다.
- **포트 탭에서 추가 단추를 클릭합니다.** 디자이너가 구성할 수 있는 빈 포트를 생성합니다.

## 포트 구성

변환 포트를 정의할 경우 포트 속성을 정의하십시오. 포트 속성에는 포트 이름, 데이터 유형, 포트 유형 및 기본값이 포함됩니다.

다음 포트 속성을 구성하십시오.

- **포트 이름.** 포트의 이름입니다. 포트 이름을 지정할 경우 다음 규칙을 사용하십시오.
  - 싱글바이트 또는 더블바이트 문자나 싱글바이트 또는 더블바이트 밑줄(\_)로 시작합니다.
  - 포트 이름에는 싱글바이트 또는 더블바이트 문자의 문자, 숫자, 밑줄(\_), \$, # 또는 @가 포함될 수 있습니다.
- **데이터 유형, 전체 자릿수 및 소수 자릿수.** 식 또는 조건을 입력할 경우 데이터 유형이 식의 반환 값과 일치하는지 확인하십시오.
- **포트 유형.** 변환에 입력, 출력, 입력/출력 및 변수 포트 유형의 조합이 포함될 수 있습니다.
- **기본값.** Null 값 또는 출력 변환 오류를 포함하는 포트에 대한 기본값을 할당합니다. 일부 포트의 기본값을 재정의할 수 있습니다.
- **설명.** 포트에 대한 설명입니다.
- **기타 속성.** 일부 변환에는 식 또는 그룹 기준 속성과 같이 해당 변환과 관련된 속성이 있습니다.

## 포트 연결

매핑에서 입력, 출력, 변환 및 맵렛 개체를 추가하고 구성한 후에는 매핑 개체 간에 포트를 연결하여 매핑을 완성합니다.

연결이 링크 유효성 검사 및 연결 요구 사항을 만족하는 경우에만 Developer 도구가 연결을 작성합니다.

연결되지 않은 상태로 포트를 유지할 수 있습니다. 데이터 통합 서비스는 연결되지 않은 포트를 무시합니다.

입력 개체, 변환, 맵렛, 출력 개체 간에 포트를 연결하면 다음과 같은 유형의 링크를 작성할 수 있습니다.

- 일대일
- 일대다

포트를 수동 또는 자동으로 연결할 수 있습니다.

### 일대일 링크

입력 개체 또는 변환의 포트 한 개를 출력 개체 또는 변환의 포트 한 개에 연결합니다.

### 일대다 링크

동일한 데이터를 여러 용도로 사용하려는 경우 포트를 연결하여 해당 데이터를 매핑의 여러 포트에 제공할 수 있습니다.

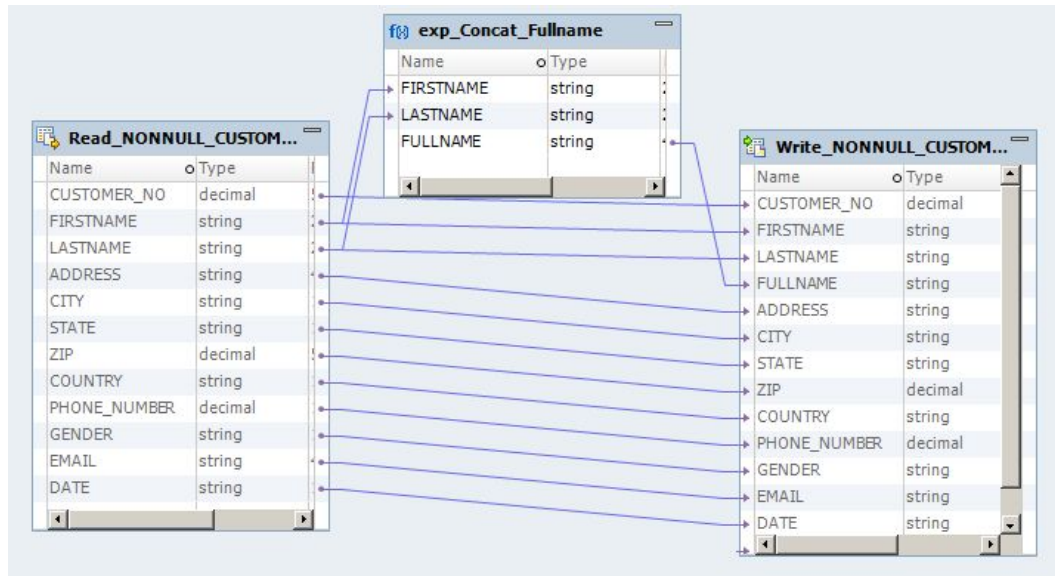
다음과 같은 방식으로 일대다 링크를 작성할 수 있습니다.

- 한 포트를 여러 변환 또는 출력 개체에 연결합니다.
- 한 변환의 여러 포트를 여러 변환 또는 출력 개체에 연결합니다.

예를 들어 급여 정보를 사용하여 집계 변환을 통해 은행 지점의 평균 연봉을 계산한다고 가정합니다. 각 직원의 월급을 계산하기 위해 구성된 식 변환에서 동일한 정보를 사용할 수 있습니다.



다음 그림은 일대다 링크를 사용하는 매핑의 예를 보여 줍니다.



## 수동으로 포트 연결

하나 또는 여러 개의 포트를 수동으로 연결할 수 있습니다.

입력 개체 또는 변환의 포트를 출력 개체 또는 변환의 포트로 끌어서 놓습니다.

Ctrl 또는 Shift 키를 사용하여 다른 변환 또는 출력 개체에 연결할 포트를 여러 개 선택할 수 있습니다.

**Developer** 도구가 맨 위의 쌍부터 포트를 연결하기 시작하고 유효성 검사 요구 사항을 만족하는 모든 포트를 연결합니다.

포트를 빈 포트로 끌어서 놓으면 **Developer** 도구가 포트를 복사하고 링크를 작성합니다.

## 자동으로 포트 연결

포트를 자동으로 연결하면 위치 또는 이름으로 연결할 수 있습니다.

이름을 사용하여 자동으로 포트를 연결하는 경우 포트를 연결할 접두사 또는 접미사를 지정할 수 있습니다. 접두사 또는 접미사를 사용하여 포트가 매핑에서 나타나는 위치를 나타냅니다.

### 이름별로 포트 연결

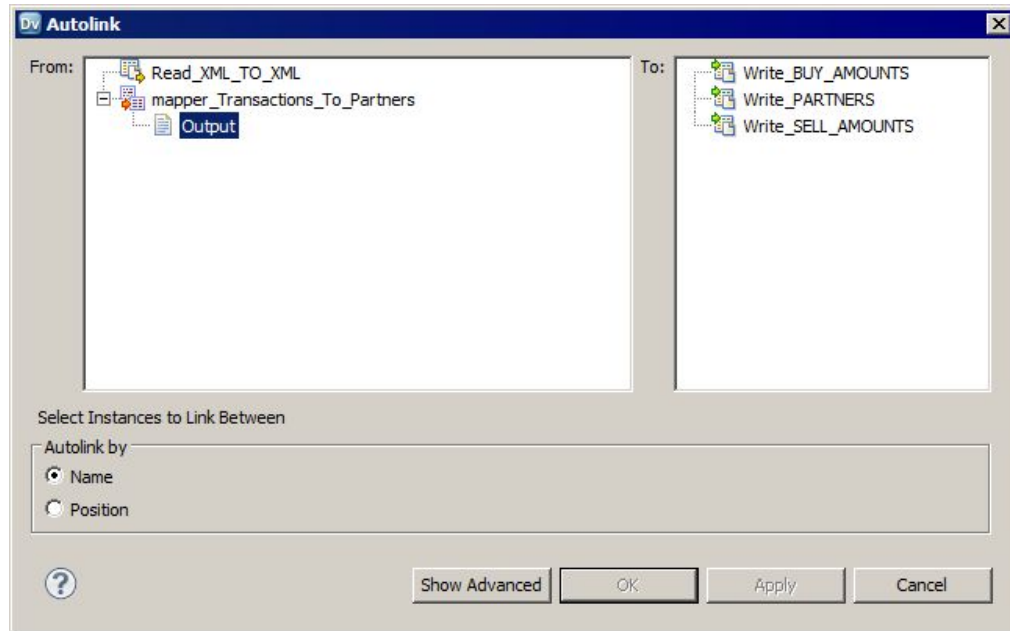
이름별로 포트를 연결할 경우 **Developer** 도구가 이름이 동일한 입력 포트와 출력 포트 간에 링크를 추가합니다. 전체 변환에서 동일한 포트 이름을 사용하는 경우 이름별로 연결합니다.

정의하는 접두사 및 접미사를 기반으로 포트를 연결할 수 있습니다. 접두사 또는 접미사를 사용하여 포트가 매핑에서 나타나는 위치를 나타냅니다. 포트 이름에서 접두사 또는 접미사를 사용하여 매핑 또는 맵렛에서 나타나는 위치를 구분하는 경우 이름 및 접두사 또는 접미사로 연결합니다.

이름별 연결은 대/소문자를 구분하지 않습니다.

1. **매핑 > 자동 연결**을 클릭합니다.

자동 연결 대화 상자가 나타납니다.



2. **시작** 창에서 연결할 개체를 선택합니다.
3. **끝** 창에서 연결할 개체를 선택합니다.
4. **이름**을 선택합니다.
5. 필요한 경우 **고급 항목 표시**를 클릭하여 접두사 또는 접미사를 기반으로 포트를 연결합니다.
6. **확인**을 클릭합니다.

## 위치별로 포트 연결

위치별로 연결할 경우 **Developer** 도구가 각 출력 포트를 해당 입력 포트에 연결합니다. 예를 들어 첫 번째 출력 포트는 첫 번째 입력 포트에 연결되고 두 번째 출력 포트는 두 번째 입력 포트에 연결됩니다. 동일한 순서로 관련 포트를 사용하여 변환을 작성할 경우 위치별로 연결합니다.

1. **매핑 > 자동 연결**을 클릭합니다.

자동 연결 대화 상자가 나타납니다.

2. **시작** 창에서 연결할 개체를 선택합니다.
3. **끝** 창에서 연결할 개체를 선택합니다.
4. **위치**를 선택하고 **확인**을 클릭합니다.

**Developer** 도구가 각 출력 포트를 해당 입력 포트에 연결합니다. 예를 들어 첫 번째 출력 포트는 첫 번째 입력 포트에 연결되고 두 번째 출력 포트는 두 번째 입력 포트에 연결됩니다.

## 포트 연결에 대한 규칙 및 지침

포트를 연결하는 경우 특정한 규칙 및 지침이 적용됩니다.

매핑 개체를 연결할 때 다음 규칙과 지침을 고려하십시오.

- 두 매핑 개체 간에 포트를 연결하려고 할 때 **Developer** 도구가 오류를 감지하면 포트를 연결할 수 없음을 나타내는 기호를 표시합니다.
- 매핑에서 데이터 흐름의 논리를 따릅니다. 다음과 같은 유형의 포트를 연결할 수 있습니다.
  - 수신 포트는 입력 포트 또는 입력/출력 포트여야 합니다.
  - 발신 포트는 출력 포트 또는 입력/출력 포트여야 합니다.
  - 입력 포트를 입력 포트에 연결하거나 출력 포트를 출력 포트에 연결할 수 없습니다.
- 입력 그룹의 포트 중 하나 이상을 업스트림 변환에 연결해야 합니다.
- 출력 그룹의 포트 중 하나 이상을 다운스트림 변환에 연결해야 합니다.
- 한 활성 변환 또는 활성 변환의 한 출력 그룹에서 다른 변환의 입력 그룹으로 포트를 연결할 수 있습니다.
- 활성 변환 및 수동 변환을 동일한 다운스트림 변환 또는 변환 입력 그룹으로 연결할 수 없습니다.
- 두 개 이상의 활성 변환을 동일한 다운스트림 변환 또는 변환 입력 그룹으로 연결할 수 없습니다.
- 원하는 수의 수동 변환을 동일한 다운스트림 변환, 변환 입력 그룹 또는 대상으로 연결할 수 있습니다.
- 동일한 변환에 있는 두 출력 그룹의 데이터가 정렬된 경우 이 두 그룹에서 정렬된 데이터에 대해 구성된 한 조이너 변환으로 포트를 연결할 수 있습니다.
- 데이터 유형이 호환 가능한 포트만 연결할 수 있습니다. **Developer** 도구는 두 데이터 유형을 연결하기 전에 두 유형 간에 매핑이 가능한지 확인합니다. 데이터 통합 서비스는 데이터 유형이 호환되지 않는 포트 간에 데이터를 변환할 수 없습니다.
- **Developer** 도구는 매핑이 데이터 흐름 유효성 검사를 위반할 경우 매핑을 올바르지 않은 것으로 표시합니다.

## 포트 특성 전달

포트 특성을 전달하여 변경된 특성을 매핑 전체에서 포트에 전달합니다.

1. 편집기에서 변환의 포트를 선택합니다.
2. **매핑 > 특성 전달**을 클릭합니다.  
**특성 전달** 대화 상자가 나타납니다.
3. 특성을 전달할 방향을 선택합니다.
4. 전달할 특성을 선택합니다.
5. 필요한 경우 결과를 미리 봅니다.
6. **적용**을 클릭합니다.  
**Developer** 도구가 포트 특성을 전달합니다.

## 종속성 유형

포트 특성을 전달하면 **Developer** 도구가 종속성을 업데이트합니다.

**Developer** 도구는 다음과 같은 종속성을 업데이트할 수 있습니다.

- 링크 경로 종속성
- 암시적 종속성

## 링크 경로 종속성

링크 경로 종속성은 전달된 포트와 해당 링크 경로의 포트 간 종속성입니다.

링크 경로의 종속성을 전달하면 **Developer** 도구가 정방향 링크 경로의 모든 입력 및 입력/출력 포트와 역방향 링크 경로의 모든 출력 및 입력/출력 포트를 업데이트합니다. **Developer** 도구는 다음과 같은 업데이트를 수행합니다.

- 전달된 포트의 링크 경로에 있는 모든 포트에 대해 포트 이름, 데이터 유형, 전체 자릿수, 소수 자릿수 및 설명을 업데이트합니다.
- 전달된 포트를 참조하는 모든 식 또는 조건을 변경된 포트 이름으로 업데이트합니다.
- 연결된 포트 이름이 변경되는 경우 동적 조회 변환의 연결된 포트 속성을 업데이트합니다.

## 암시적 종속성

암시적 종속성은 식 또는 조건을 기반으로 하는 두 포트 간의 변환 내 종속성입니다.

데이터 유형, 전체 자릿수, 소수 자릿수 및 설명을 암시적 종속성이 있는 포트에 전달할 수 있습니다. 또한 조건 및 식을 구문 분석하여 전달된 포트의 암시적 종속성을 확인할 수도 있습니다. 암시적 종속성이 있는 모든 포트는 출력 포트 또는 입력/출력 포트입니다.

조건을 포함하면 **Developer** 도구가 다음과 같은 종속성을 업데이트합니다.

- 링크 경로 종속성
- 동일한 조회 조건에서 전달된 포트에 사용된 출력 포트
- 전달된 포트와 연결된 동적 조회 변환의 연결된 포트
- 동일한 조인 조건에서 세부 포트에 사용된 마스터 포트

식을 포함하면 **Developer** 도구가 다음과 같은 종속성을 업데이트합니다.

- 링크 경로 종속성
- 전달된 포트를 사용하는 식을 포함하는 출력 포트

**Developer** 도구는 동일한 변환 내 암시적 종속성으로 전달하지 않습니다. 다른 변환에서 변경된 특성을 전달해야 합니다. 예를 들어 조회 조건에 사용된 포트의 데이터 유형을 변경하고 이 변경 사항을 조회 변환에서 전달하는 경우 **Developer** 도구는 동일한 조회 변환의 조건에 종속된 다른 포트에 변경 사항을 전달하지 않습니다.

## 변환별 전달된 포트 특성

Developer tool은 각 변환에 대해 종속성 및 특성을 전달합니다.

다음 표에서는 Developer tool이 각 변환에 대해 전달하는 종속성 및 특성에 대해 설명합니다.

변환	종속성	전달되는 특성
주소 유효성 검사기	없음.	없음. 이 변환에 포트 이름 및 데이터 유형이 미리 정의되어 있습니다.
집계	<ul style="list-style-type: none"> <li>- 링크 경로의 포트</li> <li>- 식</li> <li>- 암시적 종속성</li> </ul>	<ul style="list-style-type: none"> <li>- 포트 이름, 데이터 유형, 전체 자릿수, 소수 자릿수, 설명</li> <li>- 포트 이름</li> <li>- 데이터 유형, 전체 자릿수, 소수 자릿수</li> </ul>
연관	<ul style="list-style-type: none"> <li>- 링크 경로의 포트</li> </ul>	<ul style="list-style-type: none"> <li>- 포트 이름, 데이터 유형, 전체 자릿수, 소수 자릿수, 설명</li> </ul>
대/소문자 변환기	<ul style="list-style-type: none"> <li>- 링크 경로의 포트</li> </ul>	<ul style="list-style-type: none"> <li>- 포트 이름, 데이터 유형, 전체 자릿수, 소수 자릿수, 설명</li> </ul>
분류자	<ul style="list-style-type: none"> <li>- 링크 경로의 포트</li> </ul>	<ul style="list-style-type: none"> <li>- 포트 이름, 데이터 유형, 전체 자릿수, 소수 자릿수, 설명</li> </ul>
비교	<ul style="list-style-type: none"> <li>- 링크 경로의 포트</li> </ul>	<ul style="list-style-type: none"> <li>- 포트 이름, 데이터 유형, 전체 자릿수, 소수 자릿수, 설명</li> </ul>
통합자	없음.	없음. 이 변환에 포트 이름 및 데이터 유형이 미리 정의되어 있습니다.
데이터 마스킹	<ul style="list-style-type: none"> <li>- 링크 경로의 포트</li> </ul>	<ul style="list-style-type: none"> <li>- 포트 이름, 데이터 유형, 전체 자릿수, 소수 자릿수, 설명</li> </ul>
데이터 프로세서	<ul style="list-style-type: none"> <li>- 링크 경로의 포트</li> </ul>	<ul style="list-style-type: none"> <li>- 포트 이름, 데이터 유형, 전체 자릿수, 소수 자릿수, 설명</li> </ul>
결정	<ul style="list-style-type: none"> <li>- 링크 경로의 다운스트림 포트</li> </ul>	<ul style="list-style-type: none"> <li>- 포트 이름, 데이터 유형, 전체 자릿수, 소수 자릿수, 설명</li> </ul>
식	<ul style="list-style-type: none"> <li>- 링크 경로의 포트</li> <li>- 식</li> <li>- 암시적 종속성</li> </ul>	<ul style="list-style-type: none"> <li>- 포트 이름, 데이터 유형, 전체 자릿수, 소수 자릿수, 설명</li> <li>- 포트 이름</li> <li>- 데이터 유형, 전체 자릿수, 소수 자릿수</li> </ul>
필터	<ul style="list-style-type: none"> <li>- 링크 경로의 포트</li> <li>- 조건</li> </ul>	<ul style="list-style-type: none"> <li>- 포트 이름, 데이터 유형, 전체 자릿수, 소수 자릿수, 설명</li> <li>- 포트 이름</li> </ul>
계층-관계형	<ul style="list-style-type: none"> <li>- 링크 경로의 포트</li> </ul>	<ul style="list-style-type: none"> <li>- 포트 이름, 데이터 유형, 전체 자릿수, 소수 자릿수, 설명</li> </ul>
조이너	<ul style="list-style-type: none"> <li>- 링크 경로의 포트</li> <li>- 조건</li> <li>- 암시적 종속성</li> </ul>	<ul style="list-style-type: none"> <li>- 포트 이름, 데이터 유형, 전체 자릿수, 소수 자릿수, 설명</li> <li>- 포트 이름</li> <li>- 데이터 유형, 전체 자릿수, 소수 자릿수</li> </ul>
키 생성기	<ul style="list-style-type: none"> <li>- 링크 경로의 포트</li> </ul>	<ul style="list-style-type: none"> <li>- 포트 이름, 데이터 유형, 전체 자릿수, 소수 자릿수, 설명</li> </ul>
라벨러	<ul style="list-style-type: none"> <li>- 링크 경로의 포트</li> </ul>	<ul style="list-style-type: none"> <li>- 포트 이름, 데이터 유형, 전체 자릿수, 소수 자릿수, 설명</li> </ul>
조회	<ul style="list-style-type: none"> <li>- 링크 경로의 포트</li> <li>- 조건</li> <li>- 연결된 포트(동적 조회)</li> <li>- 암시적 종속성</li> </ul>	<ul style="list-style-type: none"> <li>- 포트 이름, 데이터 유형, 전체 자릿수, 소수 자릿수, 설명</li> <li>- 포트 이름</li> <li>- 포트 이름</li> <li>- 데이터 유형, 전체 자릿수, 소수 자릿수</li> </ul>
일치	<ul style="list-style-type: none"> <li>- 링크 경로의 포트</li> </ul>	<ul style="list-style-type: none"> <li>- 포트 이름, 데이터 유형, 전체 자릿수, 소수 자릿수, 설명</li> </ul>

변환	종속성	전달되는 특성
병합	- 링크 경로의 포트	- 포트 이름, 데이터 유형, 전체 자릿수, 소수 자릿수, 설명
노멀라이저	- 링크 경로의 포트	- 포트 이름
파서	- 링크 경로의 포트	- 포트 이름, 데이터 유형, 전체 자릿수, 소수 자릿수, 설명
순위	- 링크 경로의 포트 - 식 - 암시적 종속성	- 포트 이름, 데이터 유형, 전체 자릿수, 소수 자릿수, 설명 - 포트 이름 - 데이터 유형, 전체 자릿수, 소수 자릿수
읽기		
REST 웹 서비스 소비자	- 링크 경로의 포트	- 포트 이름, 데이터 유형, 전체 자릿수, 소수 자릿수, 설명
라우터	- 링크 경로의 포트 - 조건	- 포트 이름, 데이터 유형, 전체 자릿수, 소수 자릿수, 설명 - 포트 이름
시퀀스 생성기	- 링크 경로의 포트	- 포트 이름, 데이터 유형, 전체 자릿수, 소수 자릿수, 설명
분류기	- 링크 경로의 포트	- 포트 이름, 데이터 유형, 전체 자릿수, 소수 자릿수, 설명
SQL	- 링크 경로의 포트	- 포트 이름, 데이터 유형, 전체 자릿수, 소수 자릿수, 설명
표준화	- 링크 경로의 포트	- 포트 이름, 데이터 유형, 전체 자릿수, 소수 자릿수, 설명
합집합	- 링크 경로의 포트 - 암시적 종속성	- 포트 이름, 데이터 유형, 전체 자릿수, 소수 자릿수, 설명 - 데이터 유형, 전체 자릿수, 소수 자릿수
업데이트 전략	- 링크 경로의 포트 - 식 - 암시적 종속성	- 포트 이름, 데이터 유형, 전체 자릿수, 소수 자릿수, 설명 - 포트 이름 - 데이터 유형, 전체 자릿수, 소수 자릿수
가중치 평균	- 링크 경로의 포트	- 포트 이름, 데이터 유형, 전체 자릿수, 소수 자릿수, 설명
쓰기		

## Excel에서 포트 복사

포트 및 포트 속성을 Excel에서 구성하고 Developer tool의 변환 포트로 복사할 수 있습니다. 포트 속성에는 열 이름, 데이터 유형, 전체 자릿수 및 소수 자릿수가 포함됩니다. 포트가 많은 변환을 개발하거나 편집해야 하는 경우 이렇게 할 수 있습니다.

메타데이터를 다음 변환 유형으로 복사할 수 있습니다.

- 집계
- 식
- 필터
- Java

- 조이너
- 조회
- 노멀라이저
- 순위
- 읽기
- 라우터
- 시퀀스
- 분류기
- SQL
- 합집합
- 업데이트 전략
- 웹 서비스 소비자
- 창
- 쓰기

## Excel에서 변환 편집

변환의 많은 부분을 편집해야 하는 경우 Developer tool에서 모든 값을 변경할 필요가 없습니다. 대신에 변환 포트를 Excel에 복사하고 자동 채우기를 사용하여 모든 값을 동시에 변경한 다음 변환 포트를 Developer tool에 다시 붙여 넣습니다(바꿉니다).

1. Developer tool의 매핑 편집기에서 포트를 복사할 변환을 선택합니다.
2. Developer tool에서 원래 변환을 복사하려면 포트 내에서 마우스 오른쪽 단추를 클릭하고 **모두 선택**을 클릭합니다.
3. 포트를 Excel 스프레드시트에 복사합니다.
4. Excel 스프레드시트 내에서 데이터를 변경합니다. 메타데이터의 많은 부분을 변경하는 경우 Excel의 자동 채우기 기능을 사용할 수 있습니다. 이렇게 하면 채우기 핸들을 끌어서 인접한 셀을 기반으로 데이터를 채울 수 있습니다. 자세한 내용은 [“예: Excel에서 변환 편집” 페이지 68](#)을 참조하십시오.
5. Excel에서 메타데이터를 복사합니다.
6. 변환을 변경된 내용으로 업데이트하려면 포트 내에서 마우스 오른쪽 단추를 클릭하고 **붙여넣기(바꾸기)**를 클릭합니다.

## Developer tool에 메타데이터 복사

변환 포트를 Excel에서 생성한 다음 Developer tool에 복사할 수 있습니다.

1. 필요한 변환을 사용하여 Developer tool에서 매핑을 생성합니다.
2. Excel에서 변환에 대한 메타데이터를 정의합니다.
3. Excel에서 메타데이터를 복사합니다.
4. 메타데이터를 Developer tool의 변환으로 이동하려면 포트 내에서 마우스 오른쪽 단추를 클릭하고 **붙여넣기(바꾸기)**를 클릭합니다.

다음 이미지는 샘플 Excel 테이블 및 메타데이터를 Developer tool에 복사한 후 생성된 변환을 보여줍니다.

	A	B	C	D
1	Name	Type	Precision	Scale
2	EMPNO	decimal	10	0
3	ENAME	string	6	0
4	JOB	string	9	0
5	MGR	decimal	4	0
6	HIREDATE	string	19	0
7	SAL	decimal	7	0
8	COMM	string	7	0
9	DEPTNO	decimal	2	0

Sample Excel table

Properties		Data Viewer	Alerts
General			
Ports			
Name	Type	Precision	Scale
1 EMPNO	decimal	10	0
2 ENAME	string	6	0
3 JOB	string	9	0
4 MGR	decimal	4	0
5 HIREDATE	string	19	0
6 SAL	decimal	7	0
7 COMM	string	7	0
8 DEPTNO	decimal	2	0

After metadata is copied to the Developer tool

**참고:** 변환에 값을 복사하기 전에 각 셀의 값이 유효한지 반드시 확인해야 합니다. 예를 들어, 문자열 유형은 "0" 이외의 값을 가질 수 없습니다. 전체 자릿수 값은 단어일 수 없으며 유형 값은 숫자일 수 없습니다. 메타데이터가 잘못되면 오류 메시지가 표시됩니다.

## 예: Excel에서 변환 편집

변환을 개발하면서 모든 문자열 데이터 유형을 10진수로 변경해야 합니다. 각 필드를 개별적으로 변경하는 대신 Excel에서 전역 변경을 수행하여 Developer tool에 복사합니다.

1. 포트 내에서 마우스 오른쪽 단추를 클릭하고 **모두 선택**을 클릭하여 메타데이터를 Excel에 붙여 넣습니다.
2. 첫 번째 데이터 유형 값을 "문자열"에서 "10진수"로 변경한 다음 채우기 핸들을 사용하여 열의 나머지 셀을 자동으로 변경합니다.
3. 변경된 내용으로 변환을 업데이트하려면 Excel에서 메타데이터를 복사하고 포트 내에서 마우스 오른쪽 단추를 클릭하여 **붙여넣기(바꾸기)**를 클릭합니다.

Excel을 사용하면 각 필드를 개별적으로 변경하지 않아도 됩니다.

다음 이미지는 변환을 Excel로 이동하고 자동 채우기를 사용하여 특정한 값을 변경한 다음 변환을 Developer tool에 다시 복사하는 프로세스를 보여줍니다.

f Expression			
Name	Type	Length	Expression
All Ports (8)			
EMPNO	string	10	0
ENAME	string	6	0
JOB	string	9	0
MGR	string	4	0
HIREDATE	string	19	0
SAL	string	7	0
COMM	string	7	0
DEPTNO	string	2	0

Original transformation

	A	B	C	D	E
1	Name	Type	Precision	Scale	
2	EMPNO	decimal	10	0	
3	ENAME	decimal	6	0	
4	JOB	decimal	9	0	
5	MGR	decimal	4	0	
6	HIREDATE	decimal	19	0	
7	SAL	decimal	7	0	
8	COMM	decimal	7	0	
9	DEPTNO	decimal	2	0	

Values changed in Excel

f Expression			
Name	Type	Length	Expression
All Ports (8)			
EMPNO	decimal	10	0
ENAME	decimal	6	0
JOB	decimal	9	0
MGR	decimal	4	0
HIREDATE	decimal	19	0
SAL	decimal	7	0
COMM	decimal	7	0
DEPTNO	decimal	2	0

Copied back to mapping

## Excel에서 복사에 대한 규칙 및 지침

메타데이터를 Excel에서 Developer tool로 복사하는 경우 다음 규칙 및 지침을 고려하십시오.

- 변환에 값을 복사하기 전에 각 셀의 값이 유효한지 반드시 확인해야 합니다. 예를 들어, 문자열 유형은 "0" 이외의 값을 가질 수 없습니다. 전체 자릿수 값은 단어일 수 없으며 유형 값은 숫자일 수 없습니다. 메타데이터가 잘못되면 오류 메시지가 표시됩니다.
- 메타데이터를 복사할 수 있는 위치는 업데이트하는 변환의 유형에 따라 다릅니다. 예를 들어 변환의 속성 보기에서 마우스 오른쪽 단추를 클릭할 때 붙여넣기(바꾸기) 옵션을 사용할 수 없는 경우가 있습니다. 하지만 편집기에서 변환 포트 내부를 마우스 오른쪽 단추로 직접 클릭하면 이 옵션을 사용할 수 있습니다.



## 제 3 장

# 변환 캐시

이 장에 포함된 항목:

- [변환 캐시 개요, 69](#)
- [캐시 유형, 70](#)
- [캐시 파일, 70](#)
- [캐시 크기, 71](#)
- [데이터 통합 서비스에 의한 캐시 크기 증가, 72](#)
- [분할된 캐시의 캐시 크기, 73](#)
- [캐시 크기 최적화, 73](#)

## 변환 캐시 개요

데이터 통합 서비스는 매핑의 집계, 조이너, 조희, 순위 및 분류기 변환에 대해 캐시 메모리를 할당합니다. 데이터 통합 서비스는 집계, 조이너, 조희 및 순위 변환에 대해 인덱스 및 데이터 캐시를 작성합니다. 데이터 통합 서비스는 분류기 변환에 대해 하나의 캐시를 작성합니다.

이러한 변환에 대한 캐시 크기를 구성할 수 있습니다. 캐시 크기는 매핑 실행 시작 시 데이터 통합 서비스가 각 변환 캐시에 대해 할당하는 메모리의 양을 결정합니다.

캐시 크기가 시스템에서 사용 가능한 메모리보다 큰 경우 데이터 통합 서비스가 충분한 메모리를 할당하지 못하기 때문에 매핑 실행이 실패합니다.

캐시 크기가 변환을 실행하는 데 필요한 메모리 양보다 작은 경우 데이터 통합 서비스는 메모리에서 일부 변환을 처리하고 오버플로우 데이터를 캐시 파일에 저장합니다. 데이터 통합 서비스에서 캐시 파일을 디스크로 페이징할 때에는 처리 시간이 증가합니다. 최적의 성능을 위해, 데이터 통합 서비스가 메모리에서 전체 변환을 처리할 수 있도록 캐시 크기를 구성합니다.

기본적으로 데이터 통합 서비스는 서비스에서 할당할 수 있는 최대 메모리 양을 기반으로 런타임 시에 메모리 요구 사항을 자동으로 계산합니다. 자동 캐시 모드에서 매핑을 실행한 후 변환에 대한 캐시 크기를 조정할 수 있습니다. 매핑 로그의 변환 통계를 분석하여 최적의 성능에 필요한 캐시 크기를 결정한 다음 변환에 대한 특정 캐시 크기를 구성합니다.

## 캐시 유형

집계, 조이너, 조회 및 순위 변환에는 인덱스 캐시와 데이터 캐시가 필요합니다. 데이터 통합 서비스는 키 값을 인덱스 캐시에 저장하고 출력 값을 데이터 캐시에 저장합니다. 분류기 변환에는 단일 캐시가 필요합니다. 데이터 통합 서비스는 분류기 캐시에서 정렬될 정렬 키와 데이터를 저장합니다.

다음 테이블에는 데이터 통합 서비스가 각 캐시에 저장하는 정보 유형이 설명되어 있습니다.

매핑 개체	캐시 유형 및 설명
집계	<ul style="list-style-type: none"><li>- 인덱스. 그룹 기준 포트에 구성된 대로 그룹 값을 저장합니다.</li><li>- 데이터. 그룹 기준 포트를 기반으로 계산을 저장합니다.</li></ul>
조이너	<ul style="list-style-type: none"><li>- 인덱스. 고유한 키를 가진 조인 조건에 모든 마스터 행을 저장합니다.</li><li>- 데이터. 마스터 소스 행을 저장합니다.</li></ul>
조회	<ul style="list-style-type: none"><li>- 인덱스. 조회 조건 정보를 저장합니다.</li><li>- 데이터. 인덱스 캐시에 저장되지 않은 조회 데이터를 저장합니다.</li></ul>
순위	<ul style="list-style-type: none"><li>- 인덱스. 그룹 기준 포트에 구성된 대로 그룹 값을 저장합니다.</li><li>- 데이터. 그룹 기준 포트를 기반으로 순위 정보를 저장합니다.</li></ul>
분류기	<ul style="list-style-type: none"><li>- 분류기. 정렬 키 및 데이터를 저장합니다.</li></ul>

## 캐시 파일

매핑을 실행할 때 데이터 통합 서비스는 집계, 조이너, 조회, 순위 및 분류기 변환 각각에 대해 최소 하나의 캐시 파일을 작성합니다. 데이터 통합 서비스가 메모리에서 변환을 실행할 수 없는 경우 오버플로우 데이터를 캐시 파일에 씁니다.

다음 테이블에는 데이터 통합 서비스가 다른 매핑 개체에 대해 작성하는 캐시 파일 유형이 설명되어 있습니다.

매핑 개체	캐시 파일
집계, 조이너, 조회 및 순위 변환	데이터 통합 서비스가 다음 유형의 캐시 파일을 작성합니다. <ul style="list-style-type: none"><li>- 각 인덱스 캐시 및 데이터 캐시에 대한 헤더 파일 한 개</li><li>- 각 인덱스 캐시 및 데이터 캐시에 대한 데이터 파일 한 개</li></ul>
분류기 변환	데이터 통합 서비스가 하나의 분류기 캐시 파일을 작성합니다.

매핑을 실행하면 데이터 통합 서비스가 매핑 로그에 캐시 파일 이름 및 변환 이름을 나타내는 메시지를 씁니다. 매핑이 완료되면 데이터 통합 서비스가 캐시 메모리를 릴리스하고 일반적으로 캐시 파일을 삭제합니다. 다음 환경에서는 인덱스 및 데이터 캐시 파일을 캐시 디렉터리에서 찾을 수 있습니다.

- 지속형 캐시를 사용하도록 조회 변환을 구성합니다.
- 매핑이 성공적으로 완료되지 않습니다. 다음에 매핑을 실행할 때 데이터 통합 서비스가 기존 캐시 파일을 삭제하고 새 파일을 작성합니다.

캐시 파일에 쓰는 작업으로 인해 매핑 성능이 느려질 수 있기 때문에 메모리에서 변환을 실행하도록 캐시 크기를 구성합니다.

## 캐시 파일 디렉터리

데이터 통합 서비스는 집계, 조이너, 조희 및 순위 변환의 경우 캐시 디렉터리 속성에 지정된 디렉터리에서 캐시 파일을 작성하고 분류기 변환의 경우 작업 디렉터리 속성에 지정된 디렉터리에서 캐시 파일을 작성합니다.

데이터 통합 서비스 프로세스가 디렉터리를 찾지 못하면 매핑이 실패하고 캐시 파일을 작성할 수 없거나 열 수 없다는 메시지를 매핑 로그에 씁니다. 데이터 통합 서비스는 여러 개의 캐시 파일을 작성할 수 있습니다. 캐시 파일 수는 캐시 디렉터리에서 사용 가능한 디스크 공간 양으로 제한됩니다.

## 캐시 크기

캐시 크기는 매핑 실행 시작 시 데이터 통합 서비스가 각 변환 캐시에 대해 할당하는 메모리의 양을 결정합니다. 자동 캐시 모드 또는 특정 값을 사용하도록 변환 캐시 크기를 구성할 수 있습니다.

### 자동 캐시 크기

기본적으로 변환 캐시 크기는 "자동"으로 설정됩니다. 데이터 통합 서비스는 런타임 시 캐시 메모리 요구 사항을 자동으로 계산합니다. 사용자는 서비스에서 할당할 수 있는 최대 메모리 양을 정의합니다.

데이터 통합 서비스는 다음 지침을 사용하여 메모리를 자동으로 할당합니다.

**더 높은 처리 시간을 가진 변환에 더 많은 메모리를 할당합니다.**

데이터 통합 서비스는 일반적으로 더 높은 처리 시간을 가진 변환에 더 많은 메모리를 할당합니다. 예를 들어 데이터 통합 서비스는 분류기 변환을 실행하는 데 일반적으로 시간이 더 오래 걸리므로 분류기 변환에 더 많은 메모리를 할당합니다.

**인덱스 캐시보다 데이터 캐시에 더 많은 메모리를 할당합니다.**

집계, 조이너, 조희 및 순위 변환에는 인덱스 캐시와 데이터 캐시가 필요합니다. 데이터 통합 서비스가 변환에 대해 할당된 메모리를 인덱스 캐시와 데이터 캐시 전반에 분할할 때 더 많은 메모리가 데이터 캐시로 할당됩니다.

분류기 변환에는 단일 캐시가 필요합니다. 서비스는 변환에 대해 할당된 모든 메모리를 분류기 캐시에 할당합니다.

### 자동 캐시 크기의 최대 메모리

Administrator 도구의 데이터 통합 서비스 모듈에 대해 데이터 통합 서비스가 요청당 최대 메모리 속성에서 변환 캐시에 할당할 수 있는 최대 메모리 양을 정의합니다.

각 모듈은 메모리 요구 사항이 서로 다른 여러 유형의 요청을 실행합니다. 예를 들어, 매핑 및 프로파일 요청은 일반적으로 SQL 서비스 요청이나 웹 서비스 요청보다 더 많은 캐시 메모리를 필요로 합니다. 다음 데이터 통합 서비스 모듈에 대해 요청당 최대 메모리를 구성할 수 있습니다.

- 매핑 서비스 모듈
- 프로파일링 서비스 모듈
- SQL 서비스 모듈
- 웹 서비스 모듈

**참고:** 매핑 서비스 모듈 요청에는 워크플로우 내 매핑 태스크의 매핑과 매핑 실행이 포함됩니다.

프로파일링 서비스 모듈의 경우, 요청당 최대 메모리는 데이터 통합 서비스가 단일 프로파일 요청의 각 매핑 실행에 대해 할당할 수 있는 최대 메모리 양을 정의합니다.

나머지 모듈의 경우, 요청당 최대 메모리의 동작은 데이터 통합 서비스 구성에 종속됩니다. 동작은 데이터 통합 서비스의 작업 실행 옵션 속성 및 최대 메모리 크기 속성에 따라 다릅니다.

다음 테이블에는 데이터 통합 서비스 구성에 기반한 매핑, SQL 서비스 및 웹 서비스 모듈에 대한 요청당 최대 메모리의 동작이 설명되어 있습니다.

데이터 통합 서비스 구성	요청당 최대 메모리 동작
별도의 로컬 또는 원격 시스템 프로세스에서 작업을 실행합니다. 또는 최대 메모리 크기가 0입니다(기본 값).	<p>데이터 통합 서비스가 단일 요청에서 자동 캐시 모드를 사용하는 모든 변환에 할당할 수 있는 최대 메모리의 양(바이트)입니다.</p> <p>요청당 최대 메모리에 정의하는 값은 자동 캐시 모드를 사용하는 변환에만 영향을 미칩니다. 데이터 통합 서비스가 특정 캐시 크기를 구성하는 변환에는 별도로 메모리를 할당합니다. 요청에서 사용한 총 메모리는 요청당 최대 메모리 값을 초과할 수 있습니다.</p> <p>예를 들면, 요청당 최대 메모리가 800MB로 설정되었습니다. 한 매핑에 캐싱이 필요한 세 개의 변환이 있습니다. 두 개의 변환에서 자동 캐시 모드를 사용하도록 구성하고, 세 번째 변환에서 총 500MB의 캐시 크기를 사용하도록 구성합니다. 데이터 통합 서비스는 모든 변환 캐시에 대해 총 1,300MB의 메모리를 할당합니다.</p>
데이터 통합 서비스 프로세스에서 작업을 실행합니다. 최대 메모리 크기는 0보다 큽니다.	<p>데이터 통합 서비스에서 단일 요청에 할당할 수 있는 최대 메모리 양(바이트)입니다.</p> <p>요청당 최대 메모리 속성에 정의하는 값은 모든 변환에 영향을 미칩니다. 요청에서 사용한 총 메모리는 요청당 최대 메모리 값을 초과할 수 없습니다.</p>

자동 캐시 모드에 사용되는 최대 메모리의 양을 늘릴 때 모듈에 대한 모든 요청에 사용할 수 있는 최대 캐시 크기를 늘립니다. 캐시 파일이 디스크로 페이지되지 않도록 최대 메모리의 양을 늘릴 수 있습니다. 하지만 이 값은 모든 요청에 사용되기 때문에 데이터 통합 서비스가 일부 요청에 대해 필요한 것보다 더 많은 메모리를 할당할 수 있습니다.

## 특정 캐시 크기

변환에 대해 특정 캐시 크기를 구성할 수 있습니다. 데이터 통합 서비스는 매핑 실행 시작 시에 지정된 메모리 양을 변환 캐시에 할당합니다. 캐시 크기를 조정하려는 경우 특정 값을 바이트 단위로 구성합니다.

캐시 크기를 처음 구성할 때에는 자동 캐시 모드를 사용합니다. 매핑을 실행한 후 매핑 로그의 변환 통계를 분석하면 메모리에서 변환을 실행하는 데 필요한 캐시 크기를 결정할 수 있습니다. 매핑 로그에 지정된 값을 사용하도록 캐시 크기를 구성하는 경우 할당된 메모리가 낭비되지 않도록 할 수 있습니다. 하지만 최적의 캐시 크기는 소스 데이터의 크기에 따라 달라집니다. 후속 매핑이 실행된 다음 매핑 로그를 검토하여 캐시 크기에 대한 변경 사항을 모니터링합니다. 재사용 가능한 변환에 대해 특정 캐시 크기를 구성하는 경우 캐시 크기가 매핑의 각 변환 사용에 최적인지 확인해야 합니다.

특정 캐시 크기를 정의하려면 **Developer tool**의 변환 속성에서 캐시 크기 값을 구성합니다.

## 데이터 통합 서비스에 의한 캐시 크기 증가

데이터 통합 서비스는 구성된 캐시 크기를 기반으로 각 메모리 캐시를 작성합니다. 일부 경우 데이터 통합 서비스에 더 많은 캐시 메모리가 필요하여 데이터 통합 서비스가 구성된 캐시 크기를 늘릴 수 있습니다.

데이터 통합 서비스가 다음 이유 중 하나에 해당하는 경우 구성된 캐시 크기를 늘릴 수 있습니다.

**구성된 캐시 크기가 작업을 처리하는 데 필요한 최소 캐시 크기보다 작습니다.**

각 매핑을 초기화하기 위해 데이터 통합 서비스에 최소 메모리 양이 필요합니다. 구성된 캐시 크기가 필요한 최소 캐시 크기보다 작은 경우 최소 요구 사항을 충족하기 위해 데이터 통합 서비스가 구성된 캐시 크기를 늘립니다. 데이터 통합 서비스가 필요한 최소 메모리를 할당할 수 없는 경우 매핑이 실패합니다.

**구성된 캐시 크기가 캐시 페이지 크기의 배수가 아닙니다.**

데이터 통합 서비스는 캐시된 데이터를 캐시 페이지에 저장합니다. 캐시된 페이지는 캐시에 균일하게 맞아야 합니다. 예를 들어, 캐시 크기를 10MB(1,048,576바이트)로 구성한 상태에서 캐시 페이지 크기가 10,000바이트인 경우 데이터 통합 서비스는 10,000바이트 페이지 크기의 배수가 되도록 구성된 캐시 크기를 1,050,000바이트까지 늘립니다.

데이터 통합 서비스가 구성된 캐시 크기를 늘리는 경우 매핑이 계속 실행되고 다음과 같은 메시지를 매핑 로그에 씁니다.

```
INFO: MAPPING, TE_7212, Increasing [Index Cache] size for transformation <transformation name> from  
<configured cache size> to <new cache size>.  
INFO: MAPPING, TE_7212, Increasing [Data Cache] size for transformation <transformation name> from  
<configured cache size> to <new cache size>.
```

## 분할된 캐시의 캐시 크기

분할 옵션이 있는 경우 캐시 분할은 집계, 조이너, 순위, 조희 또는 분류기 변환을 실행하는 각 파티션에 대해 별도의 캐시를 작성합니다. 캐시 분할 중 각 파티션은 별도의 캐시에 서로 다른 데이터를 저장합니다. 데이터 통합 서비스가 이러한 변환에 캐시 분할을 사용하는 경우 데이터 통합 서비스는 전체 파티션 간에 할당된 캐시 크기를 나눕니다.

예를 들어 변환 캐시 크기를 100MB로 구성한다고 가정합니다. 데이터 통합 서비스는 4개의 파티션을 사용하여 변환을 실행합니다. 데이터 통합 서비스는 각 파티션이 최대 25MB를 캐시 크기로 사용하도록 캐시 크기 값을 나눕니다.

## 캐시 크기 최적화

최적의 매핑 성능을 위해, 데이터 통합 서비스가 메모리에서 전체 변환을 실행할 수 있도록 캐시 크기를 구성합니다.

최적의 캐시 크기를 구성하려면 다음 태스크를 수행하십시오.

1. 추적 수준을 자세한 정보 표시 초기화로 설정합니다.
2. 자동 캐시 모드에서 매핑을 실행합니다.
3. 매핑 로그의 캐싱 성능을 분석합니다.
4. 캐시 크기에 대해 특정 값을 구성합니다.

## 1단계. 추적 수준을 자세한 정보 표시 초기화로 설정합니다.

Developer tool에서, 추적 수준을 자세한 정보 표시 초기화로 설정하면 데이터 통합 서비스가 변환 통계를 매핑 로그에 쓸 수 있습니다. 변환 통계는 최적의 성능에 필요한 캐시 크기를 나열합니다. 기본적으로 추적 수준은 보통으로 설정됩니다.

다음 중 한 가지 방법으로 추적 수준을 자세한 정보 표시 초기화로 설정할 수 있습니다.

- 캐시를 사용하는 각 변환에 대한 고급 속성을 수정합니다.
- Developer tool에서 처음으로 매핑을 실행하려는 경우 기본 매핑 구성 속성을 수정합니다. 자세한 내용은 *Informatica Developer tool 가이드*를 참조하십시오.
- 명령줄에서 처음으로 배포된 매핑을 실행하려는 경우 매핑이 포함된 응용 프로그램에 대한 고급 속성을 수정합니다. 자세한 내용은 *Informatica Developer tool 가이드*를 참조하십시오.

## 2단계. 자동 캐시 모드에서 매핑 실행

매핑을 처음 실행하는 경우 변환 캐시 크기에 대해 자동 캐시 모드를 사용합니다.

Developer tool에서 매핑을 실행할 수 있습니다. 또는 응용 프로그램에 매핑을 추가한 다음 데이터 통합 서비스에 응용 프로그램을 배포하여 명령줄에서 매핑을 실행할 수도 있습니다.

## 3단계. 캐싱 성능 분석

자동 캐시 모드에서 매핑을 실행한 후 매핑 로그의 변환 통계를 분석하면 최적의 매핑 성능에 필요한 캐시 크기를 결정할 수 있습니다.

집계, 조이너, 조희 또는 순위 변환에서 디스크로 페이지할 때, 매핑 로그는 메모리에서 변환을 실행하는 데 필요한 인덱스 및 데이터 캐시 크기를 지정합니다. 예를 들어 AGG\_TRANS라는 집계 변환을 실행합니다. 매핑 로그는 다음과 같은 텍스트를 포함합니다.

```
CMN_1791, The index cache size that would hold [1098] aggregate groups of input rows for [AGG_TRANS], in memory, is [286720] bytes
CMN_1790, The data cache size that would hold [1098] aggregate groups of input rows for [AGG_TRANS], in memory, is [1774368] bytes
```

디스크에 페이지하지 않고 메모리에서 변환을 실행하려면 인덱스 캐시에는 286,720바이트가 필요하고 데이터 캐시에는 1,774,368바이트가 필요하다는 사항이 로그에 표시됩니다.

분류기 변환에서 디스크로 페이지할 때 매핑 로그는 데이터 통합 서비스가 소스 데이터에서 여러 패스를 만들었음을 알립니다. 정렬을 완료하기 위해 디스크로 페이지해야 하는 경우 데이터 통합 서비스가 데이터에서 여러 패스를 만듭니다. 메시지는 단일 패스에 필요한 바이트 수를 지정하고 이때 데이터 통합 서비스가 데이터를 한 번 읽고 디스크에 페이지하지 않고 메모리에서 정렬을 수행합니다.

예를 들어 SRT\_TRANS라는 분류기 변환을 실행합니다. 매핑 로그는 다음과 같은 텍스트를 포함합니다.

```
SORT_40427, Sorter Transformation [SRT_TRANS] required 2-pass sort (1-pass temp I/O: 13126221824 bytes). You may try to set the cache size to 14128 MB or higher for 1-pass in-memory sort.
```

데이터 통합 서비스가 데이터에서 하나의 패스를 만들려면 분류기 캐시에 14,128 MB가 필요하다는 사항이 로그에 표시됩니다.

## 4단계. 특정 캐시 크기 구성

최적의 성능을 위해서는 매핑 로그에 지정된 값을 사용하도록 변환 캐시 크기를 구성해야 합니다. Developer tool에서 인덱스 및 데이터 캐시 크기 변환 속성을 업데이트하십시오.

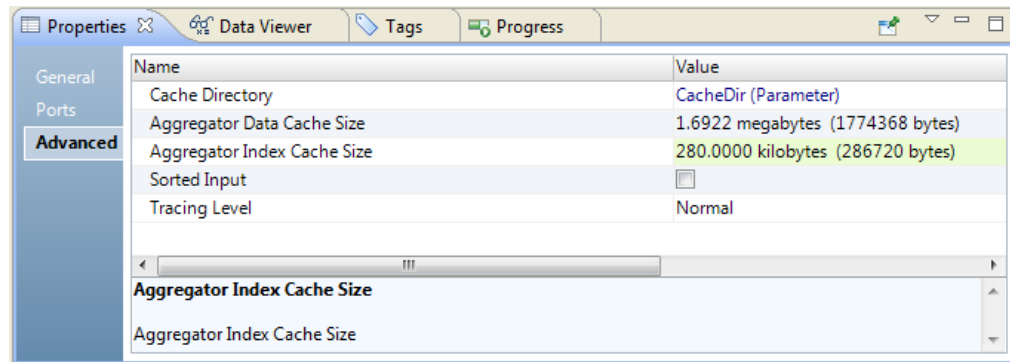
1. Developer tool에서 재사용 가능한 또는 재사용 불가능한 변환을 엽니다.

2. 다음과 같이 해당 변환 유형에 따라 캐시 크기 속성을 찾습니다.

옵션	설명
재사용 가능한 집계, 조이너, 순위 또는 분류기 변환	고급 보기를 클릭합니다.
재사용 불가능한 집계, 조이너, 순위 또는 분류기 변환	속성 보기에서 고급 탭을 클릭합니다.
재사용 가능한 조회 변환	런타임 보기를 클릭합니다.
재사용 불가능한 조회 변환	속성 보기에서 런타임 탭을 클릭합니다.

3. 맵핑 로그에서 권장하는 인덱스 및 데이터 캐시 크기 값을 바이트 단위로 입력합니다.

다음 이미지는 인덱스 및 데이터 캐시 크기에 대해 특정 값을 구성한 재사용 불가능한 집계 변환을 보여 줍니다.



4. 파일 > 저장을 클릭합니다.

## 제 4 장

# 주소 유효성 검사기 변환

이 장에 포함된 항목:

- [주소 유효성 검사기 변환 개요, 77](#)
- [주소 참조 데이터, 77](#)
- [모드 및 템플릿, 79](#)
- [포트 그룹 및 포트 선택, 79](#)
- [주소 유효성 검사기 변환 입력 포트 그룹, 79](#)
- [주소 유효성 검사기 변환 출력 포트 그룹, 80](#)
- [여러 인스턴스 포트, 83](#)
- [주소 유효성 검사 프로젝트, 84](#)
- [형식이 지정된 주소 및 우편 운송업체 표준, 85](#)
- [부분 주소 완료, 86](#)
- [주소 유효성 검사기 상태 포트, 86](#)
- [주소 유효성 검사기 변환의 일반 설정, 97](#)
- [기본 설정 창의 주소 유효성 검사 속성, 98](#)
- [주소 유효성 검사 고급 속성, 101](#)
- [인증 보고서, 120](#)
- [주소 유효성 검사기 변환 구성, 123](#)
- [주소 유효성 검사기 변환에 포트 추가, 124](#)
- [사용자 정의 템플릿 작성, 124](#)
- [주소 유효성 검사기 모델 정의, 124](#)
- [인증 보고서 정의, 125](#)
- [주소 유효성 검사기 변환 비원시 환경 내, 125](#)



## 주소 유효성 검사기 변환 개요

주소 유효성 검사기 변환은 입력 주소 데이터를 주소 참조 데이터와 비교하는 다중 그룹 변환입니다. 이 변환은 주소의 정확도를 결정하고 주소의 오류를 수정합니다. 또한 데이터 콘텐츠 및 구조에 대한 우편 배달 표준을 충족하는 레코드를 작성하고 상태 정보를 각 주소에 추가합니다.

주소 유효성 검사기 변환은 주소 데이터에 대해 다음 작업을 수행합니다.

- 소스 데이터의 주소 레코드를 주소 참조 데이터의 주소 정의와 비교합니다.
- 각 입력 주소의 유효성, 배달 가능 상태 및 주소에 포함된 오류 또는 모호성의 속성에 대한 세부적인 상태 보고서를 생성합니다.
- 오류를 수정하고 부분적인 주소 레코드를 완성합니다. 참조 데이터에서 확실히 일치하는 주소를 찾은 다음 해당 주소 참조 데이터에서 필요한 데이터 요소를 주소 레코드에 복사하여 주소를 수정합니다.
- 표준 주소에는 표시되지 않지만 우편 배달에 도움이 되는 정보를 추가합니다(예: 배달 지점 정보 및 좌표 정보).
- 데이터 프로젝트 및 우편 운송업체가 요구하는 형식으로 출력 주소를 기록합니다. 형식은 변환의 출력 포트를 선택할 때 사용자가 정의합니다.

## 주소 참조 데이터

주소 참조 데이터 집합은 국가의 국내 우편 운송업체가 인식할 수 있는 주소를 설명합니다. 주소 유효성 검사기 변환을 통해 주소 유효성 검사를 수행하기 전에 도메인의 **Informatica** 서비스 시스템에 주소 참조 데이터를 설치하십시오. **Informatica**에서 주소 참조 데이터를 구입 및 다운로드하십시오.

소스 주소 데이터가 식별하는 각 국가의 주소 참조 데이터 파일을 설치합니다. 인구가 많은 국가의 경우 여러 파일을 설치해야 할 수도 있습니다. 또한 주소 데이터를 보완하거나 보강하는 데이터 파일을 설치할 수 있습니다. 우편 운송업체는 보강된 데이터를 사용하여 주소의 정확성을 인증하고 우편을 신속하게 배달할 수 있습니다.

주소 유효성 검사를 수행할 경우 주소 유효성 검사기 변환에서 각 입력 레코드를 주소 참조 데이터와 비교합니다. 변환이 주소 참조 데이터에서 입력 주소를 찾은 경우 변환은 레코드를 완전하고 정확한 주소 데이터로 업데이트할 수 있습니다. 추가 참조 데이터 집합을 구입한 경우 변환이 주소 데이터를 보강할 수도 있습니다.

도메인의 **Informatica** 서비스 시스템에 있는 주소 참조 데이터 파일에 대한 정보를 확인하려면 **Developer tool**의 **기본 설정** 창을 사용하십시오.

## 주소 참조 데이터 유형

선택한 유효성 검사 모드는 변환이 입력 주소를 주소 참조 데이터에 비교하는 방식을 결정합니다.

주소 유효성 검사기 변환에서 다음 유형의 주소 참조 데이터를 읽을 수 있습니다.

### 우편 번호 조회 데이터

입력 포트의 코드 값에서 부분 주소 또는 전체 주소를 검색하려면 우편 번호 조회 데이터를 설치하십시오. 주소가 속하는 국가의 우편 번호 지원 수준에 따라 주소의 완성도가 결정됩니다. 입력 주소에서 우편 번호를 읽으려면 불연속 포트 그룹에서 국가에 해당하는 포트를 선택합니다.

다음 국가의 포트를 선택할 수 있습니다.

- 오스트리아. 건물 수준으로 주소를 반환합니다.
- 독일. 로컬리티, 지방자치제 또는 거리 수준으로 주소를 반환합니다.

- 일본. 고유한 우편함 수준으로 주소를 반환합니다.
- 남아프리카 공화국. 거리 수준으로 주소를 반환합니다.
- 대한민국. 고유한 우편함 수준으로 주소를 반환합니다.
- 세르비아. 거리 수준으로 주소를 반환합니다.
- 영국. 고유한 우편함 수준으로 주소를 반환합니다.

주소 유효성 검사기 변환이 우편 번호 조회 모드에서 실행되도록 구성한 경우 변환 시 우편 번호 조회 데이터를 읽습니다.

#### 일괄 데이터 및 대화형 데이터

주소 레코드 집합에서 주소 유효성 검사를 수행하려면 일괄 데이터 및 대화형 데이터를 설치합니다. 국내 우편 운송업체의 현재 우편 데이터를 기반으로 입력 주소가 완전하고 배달 가능한지 확인하려면 일괄 데이터와 대화형 데이터를 사용하십시오.

변환을 일괄 모드에서 실행되도록 구성하면 주소 유효성 검사기 변환은 각 입력 주소에 대해 주소를 하나만 반환합니다. 변환을 대화형 모드에서 실행되도록 구성하면 주소 유효성 검사기 변환은 각 입력 주소에 대해 주소를 하나 이상 반환합니다.

#### CAMEO 데이터

거주지 주소 레코드에 고객 세그먼트 데이터를 추가하려면 CAMEO 데이터를 설치합니다. 고객 세그먼트 데이터는 각 주소의 거주자에 대한 소득 수준 및 생활 방식 기본 설정을 표시합니다.

주소 유효성 검사기 변환이 일괄 모드 또는 인증 모드에서 실행되도록 구성한 경우 변환 시 CAMEO 데이터를 읽습니다.

#### 인증 데이터

우편 운송업체가 정의한 인증 표준과 주소 레코드가 일치하는지 확인하려면 인증 데이터를 설치합니다. 배달 지점 데이터 요소 같은 고유한 우편함을 식별할 수 있는 데이터 요소를 포함할 경우 주소가 인증 표준을 충족합니다. 주소가 인증 표준을 충족할 경우 우편 운송업체는 할인된 배달 요금을 청구합니다.

다음 국가는 인증 표준을 정의합니다.

- 오스트레일리아. AMAS(Address Matching Approval System) 표준에 따라 메일 인증.
- 캐나다. SERP(Software Evaluation And Recognition Program) 표준에 따라 메일 인증.
- 프랑스. SNA(National Address Management Service) 표준에 따라 메일 인증.
- 뉴질랜드. SendRight 표준에 따라 메일 인증.
- 미국. CASS(Coding Accuracy Support System) 표준에 따라 메일 인증.

주소 유효성 검사기 변환이 인증 모드에서 실행되도록 구성한 경우 변환 시 인증 데이터를 읽습니다.

#### 좌표 부여 데이터

좌표를 주소 레코드에 추가하려면 좌표 부여 데이터를 설치합니다. 좌표 부여는 위도 및 경도 좌표입니다.

주소 유효성 검사기 변환이 일괄 모드 또는 인증 모드에서 실행되도록 구성한 경우 변환 시 좌표 부여 데이터를 읽습니다.

#### 제안 목록 데이터

부분 주소 레코드의 유효한 대체 버전을 찾으려면 제안 목록 데이터를 설치하십시오. 주소 유효성 검사 매핑에서 주소 레코드를 실시간으로 하나씩 처리하도록 구성한 경우 제안 목록 데이터를 사용하십시오. 주소 유효성 검사기 변환에서 부분 주소의 데이터 요소를 사용하여 제안 목록 데이터를 중복 검사합니다. 변환 시 부분 주소에 정보가 포함된 유효한 주소를 반환합니다.

주소 유효성 검사기 변환이 제안 목록 모드에서 실행되도록 구성한 경우 변환 시 제안 목록 데이터를 읽습니다.

## 보조 데이터

우편 배달 시 우편 운송업체를 지원할 수 있는 데이터를 주소 레코드에 추가하려면 보조 데이터를 설치합니다. 주소를 포함하는 지리적 지역이나 우편 지역에 대한 상세 정보를 추가하려면 보조 데이터를 사용합니다. 일부 국가에서는 보조 데이터가 우편 시스템 내에서 우편함에 대한 고유한 식별자를 제공하기도 합니다.

주소 유효성 검사기 변환이 일괄 모드 또는 인증 모드에서 실행되도록 구성된 경우 변환 시 보조 데이터를 읽습니다.

**참고:** 국가 인식 모드 또는 구문 분석 모드에서는 변환이 주소 참조 데이터를 읽지 않습니다.

## 관련 항목:

- [“주소 유효성 검사기 변환의 일반 설정” 페이지 97](#)

# 모드 및 템플릿

주소 유효성 검사기 변환을 구성할 경우 변환에서 수행하는 유효성 검사 유형을 선택할 수 있습니다. 변환은 유효성 검사 유형을 모드로 정의합니다. 모드는 **일반 설정** 탭에서 또는 변환의 고급 설정으로 선택합니다.

포트 템플릿은 변환에서 생성할 수 있습니다. 템플릿은 하나 이상의 포트 그룹의 포트 하위 집합입니다. 템플릿을 사용하여 프로젝트에서 사용할 것으로 예상하는 포트를 구성할 수 있습니다.

# 포트 그룹 및 포트 선택

주소 유효성 검사기 변환에는 사용할 수 있는 입력 및 출력 포트가 들어 있는 미리 정의된 포트 그룹이 포함되어 있습니다. 주소 유효성 검사기 변환을 구성하는 경우 그룹을 찾고 필요한 포트를 선택합니다.

주소 입력 데이터 구조에 해당하는 입력 포트를 선택합니다. 프로젝트에 필요한 주소 데이터가 포함된 출력 포트를 선택합니다.

입력 및 출력 포트를 직접 변환에 추가하거나 입력 및 출력 포트가 포함된 기본 모델을 작성할 수 있습니다. 포트를 변환에 직접 추가하면 선택하는 포트가 해당 변환에만 적용됩니다. 기본 모델에 포트를 추가하면 선택한 포트가 이후 작성하는 주소 유효성 검사기 변환기에 적용됩니다.

주소 유효성 검사기 변환이 처리하지 않도록 하려는 열의 변환에 통과 포트를 추가할 수도 있습니다.

# 주소 유효성 검사기 변환 입력 포트 그룹

주소 데이터를 변환의 입력 포트에 연결하려면 먼저 입력 그룹을 찾아서 입력 데이터의 구조 및 콘텐츠에 해당하는 포트를 선택하십시오. 출력 그룹을 찾아보고 데이터 요구 사항에 일치하는 포트를 선택합니다.

주소 유효성 검사기 변환에서 기본 모델 및 고급 모델의 포트 그룹을 표시합니다. 기본 모델에서 포트 그룹을 사용하여 대부분의 주소를 정의할 수 있습니다. 주소가 매우 복잡할 경우 고급 모델에서 사용 가능한 추가 포트를 사용하십시오.

**참고:** 하나의 입력 포트 그룹에서만 포트를 선택하십시오.

변환에는 다음 입력 포트 그룹이 있습니다.

## 불연속

단일 데이터 요소에 대한 전체 정보(집 번호, 거리 또는 우편 번호 등)를 포함하는 데이터 열을 읽으려면 불연속 포트를 사용합니다. 불연속 그룹은 기본 모델 및 고급 모델에서 찾을 수 있습니다.

## 하이브리드

하나 이상의 데이터 요소에 대한 정보를 포함하는 데이터 열을 읽으려면 하이브리드 포트를 사용합니다. 하이브리드 그룹은 불연속 그룹 및 다중 선 그룹의 포트를 결합합니다. 우편 운송업체에 제출할 수 있는 주소 레코드를 생성하려면 하이브리드 포트를 사용합니다. 하이브리드 포트는 우편 운송업체 표준에 맞게 주소를 구성하고 각 행의 데이터 유형을 식별합니다. 하이브리드 그룹은 기본 모델 및 고급 모델에서 찾을 수 있습니다.

## 다중 선

여러 데이터 요소를 포함하는 데이터 열을 읽으려면 다중 선 포트를 사용합니다. 각 입력 열은 주소의 행에 해당합니다. 최적의 결과를 얻으려면 입력 데이터를 우편 운송업체가 요구하는 형식으로 정의하십시오. 인쇄 가능한 주소 레코드 집합을 생성하려면 다중 선 포트를 선택합니다.

각 다중 선 포트는 인쇄 가능한 주소의 행을 나타냅니다(예: 거리 데이터의 다음 행).

123 Main Street, Apartment 2

다중 선 포트는 각 주소 행에 표시되는 데이터 유형을 지정하지 않습니다. 다중 선 그룹은 기본 모델 및 고급 모델에서 찾을 수 있습니다.

## 단일 행

시/도 수준까지의 모든 주소 요소를 포함하고 요소 간의 구분 기호를 포함하지 않는 단일 데이터 열을 읽으려면 단일 행 포트를 사용합니다. 주소 요소를 제출하려면 포트 그룹에서 완전한 주소 포트를 사용합니다. 포트 그룹에는 주소에 대한 국가 정보를 읽을 때 사용되는 국가 포트도 포함되어 있습니다. 단일 행 그룹은 기본 모델 및 고급 모델에서 찾을 수 있습니다.

# 주소 유효성 검사기 변환 출력 포트 그룹

주소 유효성 검사기 변환을 다른 변환 또는 데이터 개체에 연결하려면 먼저 필요한 정보의 유형과 출력 주소에 사용할 구조를 결정해야 합니다.

출력 그룹을 찾아보고 데이터 요구 사항에 일치하는 포트를 선택합니다.

**참고:** 여러 출력 그룹에서 포트를 선택하고 공통된 기능을 포함하는 포트를 선택할 수 있습니다.

이 변환에는 다음과 같이 미리 정의된 출력 그룹이 포함됩니다.

## 주소 요소

집 번호, 아파트 번호 및 거리 이름 같은 거리 데이터 요소를 별도 포트에 씁니다. 기본 모델 및 고급 모델에서 주소 요소 그룹을 찾으십시오.

## AT 보조

건물 수준의 우편 번호 데이터처럼 우편 배달을 지원할 수 있는 데이터를 오스트리아의 주소에 씁니다. 기본 모델에서 AT 보조 그룹을 찾으십시오.

## AU 보조

오스트레일리아 통계청이 주소를 할당하는 지역을 식별하는 오스트레일리아 주소에 데이터를 씁니다. AU 보조 그룹은 기본 모델에서 찾을 수 있습니다.

#### 오스트레일리아에만 해당

주소가 오스트레일리아 우체국의 AMAS(주소 일치 승인 시스템) 표준을 충족할 수 있게 해주는 데이터를 오스트레일리아의 주소에 씁니다. 기본 모델 및 고급 모델에서 오스트레일리아에만 해당하는 그룹을 찾으십시오.

#### BE 보조

우편 배달을 지원할 수 있는 데이터를 벨기에의 주소에 씁니다. 데이터에는 벨기에 통계국의 지역 ID 코드와 로컬리티가 포함됩니다. 기본 모델에서 BE 보조 그룹을 찾으십시오.

#### BR 보조

IBGE(지리 통계 연구소)의 특별 지구 식별 코드 같이 우편 배달을 지원할 수 있는 데이터를 브라질의 주소에 씁니다. 기본 모델에서 BR 보조 그룹을 찾으십시오.

#### CAMEO

고객 세그먼트 분석에서 사용할 수 있는 인구 통계 및 소득 요약 데이터를 생성합니다. 기본 모델에서 CAMEO 그룹을 찾으십시오.

#### 캐나다에만 해당

주소가 캐나다 우체국의 SERP(소프트웨어 평가 및 인식 프로그램) 표준을 충족할 수 있게 해주는 데이터를 캐나다의 주소에 씁니다. 기본 모델에서 캐나다에만 해당 그룹을 찾으십시오.

#### CH 보조

확장된 우편 번호 데이터처럼 우편 배달을 지원할 수 있는 데이터를 스위스의 주소에 씁니다. 기본 모델에서 CH 보조 그룹을 찾으십시오.

#### CZ 보조

확장된 우편 번호 데이터처럼 우편 배달을 지원할 수 있는 데이터를 체코의 주소에 씁니다. CZ 보조 그룹은 기본 모델에서 찾을 수 있습니다.

#### 연락처 요소

이름, 인사말 및 직위 같은 개인 또는 연락처 데이터를 씁니다. 고급 모델에서 연락처 요소 그룹을 찾으십시오.

#### 국가

ISO(International Organization for Standardization)에서 정의한 국가 이름 또는 국가 코드를 씁니다. 기본 모델 및 고급 모델에서 국가 그룹을 찾으십시오.

#### DE 보조

지방 자치체 및 특별 지구 코드 데이터처럼 우편 배달을 지원할 수 있는 데이터를 독일의 주소에 씁니다. 기본 모델에서 DE 보조 그룹을 찾으십시오.

#### ES 보조

우편 배달을 지원할 수 있는 데이터를 스페인의 주소에 씁니다. 기본 모델에서 ES 보조 그룹을 찾으십시오.

#### 형식이 지정된 주소 행

인쇄 및 우편 작업을 위해 형식이 지정된 주소를 씁니다. 기본 모델 및 고급 모델에서 형식이 지정된 주소 행 그룹을 찾으십시오.

#### FR 보조

INSEE(National Institute of Statistics and Economic Studies)의 식별 코드 같이 우편 배달을 지원할 수 있는 데이터를 프랑스의 주소에 씁니다. 기본 모델에서 FR 보조 그룹을 찾으십시오.

#### 프랑스에만 해당

주소가 프랑스 우체국의 SNA(국가 주소 관리 서비스) 표준을 충족할 수 있게 해주는 데이터를 프랑스의 주소에 씁니다. 기본 모델에서 프랑스에만 해당 그룹을 찾으십시오.

## 좌표 부여

주소에 대해 위도 및 경도 좌표 같은 좌표 부여 데이터를 생성합니다. 기본 모델에서 좌표 부여 그룹을 찾으십시오.

## ID 요소

레코드 ID 및 트랜잭션 키 데이터를 씁니다. 고급 모델에서 ID 요소 그룹을 찾으십시오.

## IT 보조

우편 배달을 지원할 수 있는 데이터를 이탈리아의 주소에 씁니다. 기본 모델에서 IT 보조 그룹을 찾으십시오.

## JP 보조

Choumei Aza 코드처럼 우편 배달을 지원할 수 있는 데이터를 일본의 주소에 씁니다. 기본 모델에서 JP 보조 그룹을 찾으십시오.

## KR 보조

지정된 주소의 현재 및 이전 버전을 지정할 수 있는 고유한 식별자처럼 우편 배달을 지원할 수 있는 대한민국의 주소에 데이터를 씁니다. 기본 모델에서 KR 보조 그룹을 찾으십시오.

## 마지막 행 요소

국내 주소의 마지막 행에 표시될 수 있는 데이터를 씁니다. 기본 모델 및 고급 모델에서 마지막 행 요소를 찾으십시오.

## 뉴질랜드에만 해당

주소가 뉴질랜드 우체국의 SendRight 표준을 충족할 수 있게 해주는 데이터를 뉴질랜드의 주소에 씁니다. 기본 모델에서 뉴질랜드에만 해당 그룹을 찾으십시오.

## PL 보조

TERYT(Territorial Division) 데이터 같이 우편 배달을 지원할 수 있는 데이터를 폴란드의 주소에 씁니다. 기본 모델에서 PL 보조 그룹을 찾으십시오.

## 잔여

변환에서 다른 포트로 구문 분석할 수 없는 데이터 요소를 씁니다. 기본 모델 및 고급 모델에서 잔여 그룹을 찾으십시오.

## RS 보조

우편 번호 접미사 데이터처럼 우편 배달을 지원할 수 있는 데이터를 세르비아의 주소에 씁니다. 기본 모델에서 RS 보조 그룹을 찾으십시오.

## RU 보조

우편 배달을 지원할 수 있는 데이터를 러시아의 주소에 씁니다(예: 주소에 대한 연방 정보 주소 시스템 식별자). 기본 모델에서 RU 보조 그룹을 찾으십시오.

## 상태 정보

각 입력 및 출력 주소의 품질에 대한 상세한 데이터를 생성합니다. 기본 모델에서 상태 정보 그룹을 찾으십시오.

## 영국 보조

배달 지점 데이터, 육지 측량부 데이터처럼 우편 배달을 지원할 수 있는 데이터를 영국의 주소에 씁니다. 기본 모델에서 UK 보조 그룹을 찾으십시오.

## 미국에만 해당

주소가 미국 우체국의 CASS(Coding Accuracy Support System) 표준을 충족할 수 있게 해주는 데이터를 미국의 주소에 씁니다. 기본 모델에서 미국에만 해당 그룹을 찾으십시오.

## 미국 보조

미국 주소의 FIPS(연방 정보 처리 표준) 코드 같은 지리 및 인구 통계 데이터를 씁니다. 기본 모델에서 US 보조 그룹을 찾으십시오.

## XML

Address Verification 소프트웨어 라이브러리가 정의하는 XML 구조로 주소 레코드 데이터를 씁니다. 고급 모델에서 XML 그룹을 찾으십시오.

## ZA 보조

국가 주소 데이터베이스 데이터처럼 우편 배달을 지원할 수 있는 데이터를 남아프리카 공화국의 주소에 씁니다. 기본 모델에서 ZA 보조 그룹을 찾으십시오.

# 여러 인스턴스 포트

많은 유형의 주소 데이터가 주소에서 두 번 이상 발생할 수 있습니다. 주소에 여러 종류의 데이터 요소가 포함된 경우 포트의 여러 인스턴스를 선택할 수 있습니다.

다중 인스턴스 포트에는 최대 6개의 인스턴스가 포함될 수 있습니다. 많은 주소는 포함된 각 데이터 요소마다 1개 인스턴스의 포트를 사용합니다. 일부 주소에서는 포트의 두 번째 인스턴스를 사용합니다. 작은 주소 집합에서는 둘 이상의 포트 인스턴스를 사용합니다.

주로 포트의 첫 번째 인스턴스는 기본 이름이거나 포트가 식별하는 가장 큰 지역입니다. 선택한 포트의 포트 인스턴스 간 관계를 확인해야 합니다.

## 완전한 거리 포트 예제

영국 주소 레코드에는 두 개의 거리 이름이 포함될 수 있는데, 한 거리는 더 큰 거리 계획의 일부가 됩니다.

다음 테이블에는 완전한 거리 포트 2개를 사용하는 주소가 포함되어 있습니다.

포트	데이터
완전한 거리 번호 1	1A
완전한 거리 1	THE PHYGTLE
완전한 거리 2	SOUTH STREET
로컬리티 이름 1	NORFOLK
우편 번호 1	NR25 7QE

예제에서 완전한 거리 1의 거리 데이터는 완전한 거리 2의 거리 데이터에 종속됩니다. 완전한 거리 번호 1의 데이터는 완전한 거리 1의 데이터를 참조합니다.

**참고:** 완전한 거리 1이 우편함 위치를 지정하더라도 완전한 거리 2가 더 큰 거리가 될 수도 있습니다.

## 연락처 포트 예제

하나의 주소 레코드에 한 가정의 구성원 연락처를 나타내는 여러 연락처가 포함될 수 있습니다.

다음 테이블에는 연락처 이름 포트 2개를 사용하는 주소가 포함되어 있습니다.

포트	데이터
연락처 이름 1	MR. JOHN DOE
연락처 이름 2	MS. JANE DOE
형식이 지정된 주소 행 1	2 MCGRATH PLACE EAST
형식이 지정된 주소 행 2	ST. JOHN'S NL A1B 3V4
형식이 지정된 주소 행 3	CANADA

예제에서 조직은 연락처 이름 1 또는 연락처 이름 2를 적용할지 우선 순위를 결정할 수 있습니다. 주소 유효성 검사기 변환에서 연락처 데이터의 우선 순위를 지정하지 않습니다.

인쇄된 출력에 맞춰 주소의 형식을 지정할 경우 형식이 지정된 주소 행 포트의 여러 인스턴스를 사용할 수 있습니다. 형식이 지정된 주소 행 포트를 최대 12개까지 선택할 수 있습니다.

## 주소 유효성 검사 프로젝트

여러 유형의 프로젝트에서 주소 유효성 검사기 변환을 사용할 수 있습니다. 각 프로젝트 유형마다 다른 포트를 가진 주소 템플릿을 생성합니다.

다음과 같은 한두 가지 목표를 갖고 주소 유효성 검사 프로젝트를 정의할 수 있습니다.

### 우편 운송업체 표준을 준수하는 형식이 지정된 주소를 생성

우편 캠페인을 위해 큰 규모의 주소 레코드 집합을 작성할 수 있습니다. 우편 운송업체가 선호하는 형식으로 주소를 생성할 경우 우편 비용이 크게 줄어듭니다. 우편을 위한 주소를 작성할 때 형식이 지정된 주소의 각 행을 단일 포트에 쓰는 출력 포트를 선택하십시오. 연락처 이름, 거리 주소 행, 그리고 로컬리티 및 우편 번호 행에 대해 다른 포트를 선택할 수 있습니다.

### 소득 및 생활 방식 표시기로 주소를 구성

고객 세그먼트 데이터를 거주지 주소 레코드에 추가할 수 있습니다. 고객 세그먼트 데이터는 각 주소의 거주자에 대한 소득 수준 및 생활 방식 기본 설정을 표시합니다. 고객 세그먼트 데이터를 주소 레코드에 추가하려면 CAMEO 출력 그룹에서 포트를 선택합니다. 여러 소비자 시장을 대상으로 하는 우편 캠페인에서 고객 세그먼트 데이터를 사용할 수 있습니다.

### 우편 운송업체가 인증한 주소를 생성

오스트레일리아 우체국, 캐나다 우체국 또는 USPS(미국 우체국)에 대한 레코드 집합을 작성할 경우 각 주소의 배달 가능성을 확인하는 데이터를 추가할 수 있습니다.

주소 유효성 검사기 변환에서 주소 레코드가 각 우편 운송업체의 데이터 표준에 맞게 정확하고 완전한지 확인하는 보고서를 생성할 수 있습니다.

### 규정 요구 사항을 충족하는 주소 생성

조직에서 보유하고 있는 주소 레코드가 업계 또는 정부 규정에 맞게 정확한지 확인할 수 있습니다. 각 주소 데이터 요소를 별도 필드에 쓰는 출력 포트를 선택합니다. 또한 출력 데이터의 정확성 및 완전성에 대한 자세한 정보를 제공하는 주소 유효성 검사 상태 포트를 선택합니다.



## 부분 주소 완성

부분 주소를 입력하고 부분 주소와 일치하는 유효한 전체 주소를 참조 데이터에서 검색할 수 있습니다. 부분 주소를 완료하려면 변환이 제안 목록 모드 또는 대화형 모드에서 실행되도록 구성합니다. 완전한 주소 포트에서 입력 주소를 단일 행으로 입력할 수 있습니다.

## 주소의 데이터 품질 개선

다른 데이터 프로젝트와 병행하여 주소 데이터 집합의 구조 및 일반적인 데이터 품질을 개선할 수 있습니다. 예를 들어, 데이터 집합에 필요한 추가 열이 포함되거나, 여러 열에 동일한 유형의 데이터를 포함할 수 있습니다. 데이터 집합의 열 수를 줄이고, 다른 유형의 데이터에 사용하도록 열을 단순화할 수 있습니다.

# 형식이 지정된 주소 및 우편 운송업체 표준

우편 캠페인을 위한 주소 레코드를 생성할 때 우편 운송업체의 형식 표준과 일치하는 인쇄 가능한 주소 구조를 생성하십시오.

예를 들어, USPS에서는 미국 내 주소에 대해 다음과 같은 주소 형식을 유지 관리합니다.

Line 1	Person/Contact Data	JOHN DOE
Line 2	Street Number, Street, Sub-Building	123 MAIN ST NW STE 12
Line 3	Locality, State, ZIP Code	ANYTOWN NY 12345

주소의 각 행을 단일 포트에 쓰는 인쇄 가능한 주소 형식을 정의할 수 있습니다. 각 행에서 데이터 유형을 인식하는 포트를 사용하거나, 각 행의 데이터와 상관없이 주소 구조를 채우는 포트를 사용할 수 있습니다.

다음 테이블에서는 인쇄하기 위해 미국 주소의 형식을 지정할 수 있는 다양한 방법을 보여줍니다.

주소	사용할 포트	사용할 다른 포트
JOHN DOE	받는 사람 행 1	형식이 지정된 주소 행 1
123 MAIN ST NW STE 12	배달 주소 행 1	형식이 지정된 주소 행 2
ANYTOWN NY 12345	국가별 마지막 행 1	형식이 지정된 주소 행 3

데이터 집합에 회사 주소 및 거주지 주소 같이 다른 유형의 주소가 포함된 경우 형식이 지정된 주소 행 포트를 사용하십시오. 회사 주소의 경우 연락처 및 조직 데이터에 대한 3개의 주소 행이 필요할 수 있습니다. 주소 유효성 검사기 변환에서 필요한 경우에만 형식이 지정된 주소 행 포트를 사용하여 회사 주소 또는 거주지 주소의 형식이 올바르게 지정되었는지 확인합니다. 그러나 형식이 지정된 주소 행 포트에서 포함된 데이터 유형은 식별하지 않습니다.

모든 주소가 하나의 형식을 따를 경우 받는 사람 행, 배달 주소 행 및 국가별 마지막 행 포트를 사용하십시오. 받는 사람 행, 배달 주소 행 및 국가별 마지막 행 포트는 주소 데이터 요소를 정보 유형으로 구분하여 데이터 집합을 쉽게 이해할 수 있게 해줍니다.

**참고:** 다른 포트를 선택하여 이 주소를 처리할 수 있습니다. 이 예제에서는 인쇄 및 배달에 대한 주소 형식을 지정하는 포트에 초점을 맞춥니다.

## 지리 및 인구 통계 데이터

우편 캠페인을 위한 레코드 집합을 생성할 경우 주소에 표시되지 않을 수도 있는 여러 유형의 데이터를 추가할 수 있습니다. 우편 항목의 인구 통계 및 지리적 범위를 검토하려면 이 데이터를 사용하십시오.

예를 들어, 미국 주소가 속하는 의회 선거구를 식별할 수 있습니다. 대상 국가의 우편 시스템 참조 데이터에 좌표가 포함된 경우 위도 및 경도 좌표를 생성할 수도 있습니다.

## 부분 주소 완료

제안 목록 모드 또는 대화형 모드를 사용하는 경우 불완전한 주소를 입력한 후 참조 데이터에서 유효하고 완전한 주소를 검색할 수 있습니다.

주소가 확실하지 않을 때 유효한 주소 후보 목록을 보려면 제안 목록 모드를 선택합니다. 주소가 확실할 때 완전한 형식을 확인하려면 대화형 모드를 선택합니다. 어떤 경우든 주소 유효성 검사기 변환은 주소 참조 데이터를 검색한 다음 입력 데이터가 포함되는 모든 주소를 반환합니다.

변환이 제안 목록 모드 또는 대화형 모드에서 실행되도록 구성할 경우 다음 규칙 및 지침을 고려하십시오.

- 여러 포트에서 입력 주소를 정의하거나, 완전한 주소 입력 포트에서 모든 주소 요소를 입력할 수 있습니다.
- 제안 목록 모드에서 변환을 구성할 경우 불연속 입력 그룹에서 포트를 선택합니다. 또는 완전한 주소 포트를 선택하고 다중 선택 그룹에서 국가 이름 포트를 선택하십시오.
- 제안 목록 모드 및 대화형 모드에서 각 입력 주소에 대해 여러 주소를 반환할 수 있습니다. 최대 결과 수 속성에서 반환되는 주소 수에 대한 상한을 지정합니다. 일치하는 주소 수가 최대 결과 수 값보다 클 경우 초과 개수 포트에서 추가 주소 개수를 반환합니다.
- Informatica Address Verification에서는 제안 목록 모드를 빠른 완료 모드라고 부릅니다.

## 주소 유효성 검사기 상태 포트

주소 유효성 검사기 변환은 입력 및 출력 포트에서 읽고 쓰는 주소 요소에 대한 상태 정보를 씁니다. 상태 정보 포트를 사용하여 상태 정보를 볼 수 있습니다.

다음과 같은 상태 포트를 선택할 수 있습니다.

### 주소 확인 코드

주소에서 유효하지 않은 주소 요소를 설명합니다. 기본 모델의 상태 정보 포트 그룹에서 포트를 선택합니다.

### 주소 유형

우편 운송업체가 둘 이상의 주소 형식을 인식하는 경우 주소 유형을 나타냅니다. 기본 모델의 상태 정보 포트 그룹에서 포트를 선택합니다.

### 요소 입력 상태

입력 주소 요소와 참조 데이터 간의 유사성 수준을 설명합니다. 기본 모델의 상태 정보 포트 그룹에서 포트를 선택합니다.

### 요소 관련성

우편 운송업체가 주소에 대한 우편함을 식별하기 위해 필요한 주소 요소를 식별합니다. 기본 모델의 상태 정보 포트 그룹에서 포트를 선택합니다.

### 요소 결과 상태

입력 주소에 대해 주소 유효성 검사가 수행한 업데이트를 설명합니다. 기본 모델의 상태 정보 포트 그룹에서 포트를 선택합니다.

### 확장 요소 결과 상태

주소에 대해 추가 데이터가 있다는 것을 참조 데이터에 나타냅니다. 포트에 주소 유효성 검사가 주소에 대해 수행한 업데이트에 대한 상세한 정보를 포함할 수 있습니다. 기본 모델의 상태 정보 포트 그룹에서 포트를 선택합니다.

### 좌표 부여 상태

주소 유효성 검사가 주소에 대해 반환하는 좌표 부여 데이터 유형을 설명합니다. 기본 모델의 좌표 부여 포트 그룹에서 포트를 선택합니다.

### 편지 점수

우편 운송업체가 우편 항목을 주소로 배달할 수 있는 가능성을 나타냅니다. 기본 모델의 상태 정보 포트 그룹에서 포트를 선택합니다.

### 일치 코드

입력 주소에 대한 주소 유효성 검사 및 주소 수정 작업의 결과를 설명합니다. 기본 모델의 상태 정보 포트 그룹에서 포트를 선택합니다.

### 결과 백분율

입력 주소와 해당 출력 주소 간의 유사성 정도를 백분율 값으로 나타냅니다. 기본 모델의 상태 정보 포트 그룹에서 포트를 선택합니다.

## 요소 상태 코드 정의

요소 입력 상태, 요소 관련성, 요소 결과 상태 및 확장 요소 결과 상태 포트는 입력 및 출력 데이터 요소의 유효성에 대한 상태 정보를 제공합니다. 주소 유효성 검사 작업의 결과를 검토하려면 요소 포트를 선택합니다.

코드에는 다음 정보가 포함되어 있습니다.

- 요소 입력 상태 코드는 입력 주소 데이터와 주소 참조 데이터 간에 발견된 일치 항목의 품질을 나타냅니다.
- 요소 관련성 코드는 대상 국가에서 주소 배달하는 데 필요한 주소 요소를 식별합니다.
- 요소 결과 상태 코드는 처리 중 입력 데이터에 대한 변경사항을 설명합니다.
- 확장 요소 결과 상태 코드는 주소 참조 데이터에 주소 요소에 대한 추가 정보가 포함되어 있음을 나타냅니다.

각 포트는 20자의 코드를 반환합니다. 코드의 각 문자는 다른 주소 데이터 요소를 참조합니다. 요소 포트에서 출력 코드를 읽을 경우 각 문자가 참조하는 요소를 알고 있어야 합니다. 20개의 문자는 10개 쌍으로 구성되어 있습니다. 각 쌍의 두 개 코드가 주소 정보 유형을 나타냅니다. 예를 들어, 반환 코드의 첫 번째 위치는 기본 우편 번호 정보를 나타냅니다.

**참고:** 주소 확인 코드 포트는 요소 상태 포트와 동일한 주소 요소를 기반으로 20개 문자의 문자열을 반환합니다.

다음 테이블에서는 각 위치의 값이 식별하는 주소 요소를 설명합니다.

위치	주소 요소	설명	주소 요소 예제
1	우편 번호 수준 0	기본 우편 번호 정보(예: 5자리 우편 번호).	5자리 우편 번호 10118
2	우편 번호 수준 1	추가 우편 번호 정보(예: 우편 번호+4 코드의 마지막 4자리).	0110(우편 번호+4 코드의 경우 10118-0110)
3	로컬리티 수준 0	기본 위치(예: 구/군/시 또는 도시).	영국, 런던
4	로컬리티 수준 1	중속 로컬리티, 교외, 마을	이즐링턴, 런던

위치	주소 요소	설명	주소 요소 예제
5	시/도 수준 0	국가 내 기본 지역(예: 미국의 주 이름, 캐나다의 시/도 이름, 스위스의 주).	뉴욕주
6	시/도 수준 1	미국 카운티 이름.	퀸즈 카운티, 뉴욕주
7	거리 수준 0	기본 거리 정보.	사우스 그레이트 조지 스트리트
8	거리 수준 1	종속 거리 정보.	조지 아케이드, 사우스 그레이트 조지 스트리트
9	번호 수준 0	기본 거리와 연결된 건물 또는 집 번호.	460, 사우스 그레이트 조지 스트리트
10	번호 수준 1	종속 거리와 연결된 건물 또는 집 번호.	81, 조지 아케이드
11	배달 서비스 수준 0	사서함이나 우편 번호의 설명자 및 번호.	PO Box 111
12	배달 서비스 수준 1	배달을 담당하는 우체국 코드.	MAIN STN
13	건물 수준 0	건물 이름 또는 번호. 집 번호는 식별하지 않습니다.	앨리스 톨리 홀
14	건물 수준 1	추가 건물 이름 또는 번호.	스타 씨어터, 앨리스 톨리 홀
15	하위 건물 수준 0	아파트, 스위트 또는 층 이름 또는 번호.	80, 350 5번가, 80층
16	하위 건물 수준 1	하위 건물 수준 0 정보와 조합할 경우 아파트, 스위트 또는 층 정보.	80-18, 여기서 18은 스위트 번호이고 80은 층 번호
17	조직 수준 0	회사 이름.	AddressDoctor GmbH
18	조직 수준 1	추가 회사 정보(예: 모회사).	Informatica Corporation
19	국가 수준 0	국가 이름.	미국
20	국가 수준 1	자치령.	미국령 버진 아일랜드

포트 이름에 번호 접미사가 있는 경우 수준 0은 포트 번호 1의 데이터를 참조하고 수준 1은 포트 번호 2에서 6까지 포트의 데이터를 참조합니다.

인쇄된 주소에서 수준 0 정보는 수준 1 정보 앞 또는 뒤에 올 수 있습니다. 예를 들어, 우편 번호 수준 1은 우편 번호 수준 0 다음에 오고, 로컬리티 수준 1은 로컬리티 수준 0 앞에 옵니다.

## 주소 확인 코드 출력 포트 값

주소 확인 코드는 문자열의 각 문자가 서로 다른 입력 주소 요소를 나타내는 20자리 문자열입니다. 문자의 값은 주소의 해당하는 위치에서 유효하지 않은 모든 주소 요소를 설명합니다.

다음 표에는 주소 확인 코드 포트 값이 설명되어 있습니다.

코드	설명
2	배달을 위해서 주소 요소가 필요하지만 입력 주소에 없습니다. 주소 참조 데이터에 누락된 주소 요소가 포함되어 있습니다. 출력 2는 주소 요소가 없는 주소의 경우 배달에 유효하지 않음을 나타냅니다.
3	주소 요소가 주소의 유효한 범위를 벗어난 집 번호 또는 거리 번호입니다. 예를 들어, 주소 요소에 지정된 거리에 없는 집 번호가 포함되어 있습니다. 제안 목록 모드에서 대체 주소를 반환합니다.
4	입력 주소에 둘 이상의 요소 인스턴스가 포함되어 있기 때문에 주소 유효성 검사에서 주소 요소를 확인하거나 수정할 수 없습니다.
5	현재 주소에서 주소 요소가 모호하고 주소 참조 데이터에 대체 항목이 포함되어 있습니다. 주소 유효성 검사에서 입력 요소를 출력 요소에 복사합니다. 예를 들어, 주소 요소가 주소의 유효한 로컬리티와 일치하지 않는 유효한 우편 번호입니다.
6	주소 요소가 주소의 다른 요소와 충돌합니다. 주소 유효성 검사에서 주소에 대한 올바른 요소를 확인할 수 없습니다. 출력 주소가 입력 주소를 복사합니다.
7	주소를 여러 번 변경하지 않고 주소 요소를 수정할 수 없습니다. 주소 유효성 검사에서 주소를 수정할 수 있지만, 변경 횟수는 주소를 신뢰할 수 없음을 나타냅니다.
8	데이터가 우편 운송업체 유효성 검사 규칙을 준수하지 않습니다.

## 요소 입력 상태 출력 포트 값

요소 입력 상태는 문자열의 각 문자가 서로 다른 입력 주소 요소를 나타내는 20자리 문자열입니다. 각 문자의 값은 주소 요소에서 수행되는 처리 유형을 나타냅니다.

이 포트는 상태 정보 포트 그룹에서 찾을 수 있습니다.

다음 테이블에서는 일괄, 인증 또는 제안 목록 모드에서 요소 입력 상태가 출력 문자열의 각 위치에서 반환할 수 있는 코드에 대해 설명합니다.

코드	설명
0	현재 위치의 데이터가 입력 주소에 없습니다.
1	현재 위치의 데이터가 참조 데이터에 없습니다.
2	참조 데이터가 없어 데이터를 검사할 수 없습니다.
3	현재 위치의 데이터가 잘못되었습니다. 참조 데이터베이스가 숫자 또는 배달 서비스 값이 참조 데이터의 예상 범위 밖에 있음을 제안합니다. 일괄 및 인증 모드에서 변환이 현재 위치의 입력 데이터를 수정하지 않은 상태로 출력으로 전달합니다.

코드	설명
4	현재 위치의 데이터가 참조 데이터와 일치하지만 오류가 있습니다.
5	현재 위치의 데이터가 참조 데이터와 일치하지만 변환이 데이터를 수정했거나 표준화했습니다.
6	현재 위치의 데이터가 참조 데이터와 일치하며 오류가 없습니다.

다음 테이블에서는 구문 분석 모드에서 요소 입력 상태가 출력 문자열의 각 위치에서 반환할 수 있는 코드에 대해 설명합니다.

코드	설명
0	현재 위치의 데이터가 입력 주소에 없습니다.
1	변환이 현재 위치의 요소를 출력 주소에서 다른 위치로 이동했습니다.
2	현재 위치의 요소가 참조 데이터 값과 일치하지만 변환이 출력 주소에서 요소를 정규화했습니다.
3	현재 위치의 데이터가 올바릅니다.

## 요소 관련성 출력 포트 값

요소 관련성 값은 우편 배달에 주소 요소가 필요한지 여부를 나타냅니다. 이 포트는 상태 정보 포트 그룹에서 찾을 수 있습니다.

요소 관련성 값은 각 문자가 다른 유형의 주소 데이터를 나타내는 20자의 문자열입니다. 주소 유효성 검사 매핑을 실행한 후 포트 출력을 검토하여 각 주소에 필요한 주소 요소를 식별합니다. 결과를 사용하여 주소 데이터에 대해 올바른 출력 포트를 선택했는지 확인합니다. 관련 주소 데이터 요소에 대한 출력 포트를 선택하지 않은 경우 해당 주소의 출력이 유효하지 않습니다.

다음 테이블에는 출력 문자열의 각 위치에서 요소 관련성이 반환할 수 있는 코드가 설명되어 있습니다.

코드	설명
0	주소에 대한 배달과 관련이 없습니다.
1	주소에 대한 배달과 관련이 있습니다. 출력 문자열의 이 위치에 데이터가 없으면 국내 우편 운송업체가 주소에 배달할 수 없습니다.

**참고:** 일치 코드 값이 일괄 모드에서 Cx 또는 Vx이거나 대화형 모드에서 Cx, Vx, I3 또는 I4인 주소에 대해 요소 관련성 값을 사용할 수 있습니다. 요소 입력 상태, 요소 결과 상태, 확장 요소 결과 상태, 주소 확인 코드와 같은 기타 평가 코드는 일치 코드 값에 관계없이 값을 반환합니다.

## 요소 결과 상태 출력 포트 값

요소 결과 상태는 문자열의 각 문자가 서로 다른 입력 주소 요소를 나타내는 20자리 문자열입니다. 각 문자의 값은 유효성 검사 프로세스가 주소 요소에 수행하는 업데이트를 설명합니다.

이 포트는 상태 정보 포트 그룹에서 찾을 수 있습니다.

다음 표에는 요소 결과 상태 포트 값이 설명되어 있습니다.

코드	설명
0	현재 위치의 데이터가 출력 주소에 없습니다.
1	변환이 현재 위치의 데이터를 참조 데이터에서 찾지 못했습니다. 변환이 입력 데이터를 출력 데이터에 복사합니다.
2	현재 위치의 데이터가 검사되지 않았지만 표준화되었습니다.
3	현재 위치의 데이터가 확인되었지만 참조 데이터와 일치하지 않습니다. 참조 데이터에서 숫자 데이터가 유효한 범위가 아님을 제안합니다. 변환이 입력 데이터를 출력 포트에 복사합니다. 일괄 모드에서 적용합니다.
4	참조 데이터가 누락되어 있어 변환이 입력 데이터를 출력 데이터에 복사합니다.
5	현재 위치의 데이터 유효성이 검사되었지만 참조 데이터에 여러 일치 항목이 있기 때문에 데이터가 변경되지 않습니다. 일괄 모드에서 적용합니다.
6	데이터 유효성 검사에서 현재 위치의 입력 값을 삭제했습니다.
7	현재 위치의 데이터 유효성이 검사되었지만 입력 데이터에 맞춤법 오류가 있습니다. 유효성 검사에서 참조 데이터의 값을 사용하여 오류를 수정했습니다.
8	현재 위치의 데이터 유효성이 검사되었고 참조 데이터의 값으로 업데이트되었습니다. 값 8은 참조 데이터베이스에 입력 요소에 대한 추가 데이터가 있다는 것을 의미하기도 합니다. 예를 들어 유효성 검사에서 거리 이름 또는 건물 이름에 대한 완벽한 일치가 발견될 경우 건물 번호 또는 하위 건물 번호를 추가할 수 있습니다.
9	현재 위치의 데이터 유효성이 검사되었지만 변경되지 않았으며 배달 상태가 명확하지 않습니다. 예를 들어, DPV 값이 잘못되었습니다.
C	현재 위치의 데이터 유효성이 검사되고 확인되었지만 이름 데이터가 오래되었습니다. 유효성 검사에서 이름 데이터를 변경했습니다.
D	현재 위치의 데이터 유효성이 검사되고 확인되었지만 외국어 지명에서 공식 이름으로 변경되었습니다.
E	현재 위치의 데이터 유효성이 검사되고 확인되었습니다. 그러나 주소 유효성 검사에서 대/소문자 또는 언어를 표준화했습니다. 값이 대체 언어와 완전히 일치할 경우 주소 유효성 검사에서 언어를 변경할 수 있습니다. 예를 들어, 주소 유효성 검사에서 벨기에 주소의 "Brussels"을 "Bruxelles"로 변경할 수 있습니다.
F	현재 위치의 데이터 유효성이 검사되고 확인되었으며 참조 데이터와 완벽하게 일치하므로 변경되지 않았습니다.

출력 문자열의 19 및 20 위치는 국가 데이터와 관련되어 있습니다.

다음 테이블에서는 유효성 검사에서 19 및 20 위치에 대해 반환할 수 있는 값을 설명합니다.

코드	설명
0	현재 위치의 데이터가 출력 주소에 없습니다.
1	주소 유효성 검사에서 국가 데이터를 인식하지 않습니다.
4	주소 유효성 검사에서 주소 유효성 검사기 변환의 기본 국가 값을 사용하여 국가를 식별합니다.
5	참조 데이터에 여러 개의 일치 항목이 있어 주소 유효성 검사에서 국가를 확인할 수 없습니다.
6	주소 유효성 검사에서 스크립트를 사용하여 국가를 식별합니다.
7	주소 유효성 검사에서 주소 형식을 사용하여 국가를 식별합니다.
8	주소 유효성 검사에서 주요 도시 데이터를 사용하여 국가를 식별합니다.
9	주소 유효성 검사에서 시/도 데이터를 사용하여 국가를 식별합니다.
C	주소 유효성 검사에서 지방 데이터를 사용하여 국가를 식별합니다.
D	주소 유효성 검사에서 국가 이름을 사용하여 국가를 식별하지만 이름에 오류가 있습니다.
E	주소 유효성 검사에서 주소 데이터(예: ISO 코드 또는 국가 이름)를 사용하여 국가를 식별합니다.
F	주소 유효성 검사에서 주소 유효성 검사기 변환에 설정된 국가 강제 적용 값을 사용하여 국가를 식별합니다.

## 확장 요소 결과 상태 출력 포트 값

확장 요소 결과 상태는 문자열의 각 문자가 서로 다른 입력 주소 요소를 나타내는 20자리 문자열입니다. 확장 요소 결과 상태 출력 포트의 출력 코드는 요소 입력 상태 포트 및 요소 결과 상태 포트의 상태 데이터를 포함합니다. 이 포트의 출력 코드를 바탕으로 참조 데이터의 주소 요소에 대한 추가 정보가 있는지 여부를 확인할 수도 있습니다.

이 포트는 상태 정보 포트 그룹에서 찾을 수 있습니다.

다음 표에는 확장 요소 결과 상태 포트 값이 설명되어 있습니다.

코드	설명
1	주소 참조 데이터에는 주소 요소에 대한 추가 정보가 포함됩니다. 주소 유효성 검사에서 추가 정보를 요구하지 않습니다.
2	데이터 오류 또는 형식 오류를 해결하기 위해 주소 유효성 검사에서 주소 요소를 업데이트했습니다. 주소 유효성 검사에서 주소 요소를 확인하지 않았습니다.
3	데이터 오류 또는 형식 오류를 해결하기 위해 주소 유효성 검사에서 주소 요소를 업데이트했습니다. 주소 유효성 검사에서 주소 요소의 숫자 데이터를 확인했습니다.



코드	설명
4	형식 오류를 해결하기 위해 주소 유효성 검사에서 주소 요소를 다른 필드로 이동했습니다.
5	주소 참조 데이터에 주소 요소의 대체 버전(예: 기본 설정 로컬리티 이름)이 포함되어 있습니다.
6	주소 유효성 검사에서 주소 요소의 모든 부분을 확인하지 않았습니다. 주소 유효성 검사에서 확인할 수 없는 데이터가 요소에 포함되어 있습니다.
7	주소 유효성 검사 시 주소의 잘못된 위치에서 유효한 주소 요소가 발견되었습니다. 주소 유효성 검사에서 주소 요소를 올바른 위치로 이동했습니다.
8	주소 유효성 검사 시 잘못된 데이터 필드에서 유효한 주소 요소가 발견되었습니다. 주소 유효성 검사에서 주소 요소를 올바른 필드로 이동했습니다.
9	주소 유효성 검사에서 우편 운송업체 유효성 검사 규칙에 따라 출력 요소를 생성했습니다.
A	주소 유효성 검사에서 현재 위치에 사용 가능한 다른 주소 유형의 주소 요소를 발견했습니다. 주소 유효성 검사에서 대상 국가의 우편 운송업체 규칙을 준수하는 출력 주소 요소를 선택했습니다.
B	주소 유효성 검사에서 요소 관련성을 확인할 수 없습니다. 주소 유효성 검사에서 주소가 지정하는 국가에 대한 기본값을 반환합니다.
C	제안 목록 모드. 주소 유효성 검사에서 주소 요소에 대한 추가 주소 제안을 반환할 수 있습니다. 추가 제안을 반환하려면 주소 유효성 검사기 변환의 최대 결과 수 속성을 업데이트하십시오.
D	주소 유효성 검사에서 주소 요소에 숫자 데이터를 삽입했습니다.
E	주소 유효성 검사에서 기본 설정 언어로 주소 요소를 반환할 수 없습니다. 주소 유효성 검사에서 기본 언어로 요소를 반환합니다.
F	우편 번호 조회 모드. 입력 주소가 오래되었습니다.

## 편지 점수 출력 포트 값

배달 가능성 점수 값은 출력 주소의 배달 가능성에 대한 예측을 나타냅니다. 배달 가능성 점수를 주소의 배달 가능성을 나타내는 일반적인 지표로 사용할 수 있습니다. 이 포트는 상태 정보 포트 그룹에서 찾을 수 있습니다.

주소 유효성 검사기 변환은 배달 가능성 점수를 계산할 때 여러 요인을 고려합니다. 변환은 주로 주소에 대한 일치 코드 값과 요소 결과 상태 값을 근거로 계산을 수행합니다. 배달 가능성 점수에 영향을 미치는 다른 요인으로 주소 값의 우편 관련성과 국가에 대한 참조 데이터의 세분성이 있습니다.

배달 가능성 점수 포트 값은 주소의 배달 가능성에 대한 예측을 제공합니다. 이 점수는 주소의 배달 가능성을 나타내는 정확하거나 확정된 지표가 아닙니다.

다음 테이블에서는 배달 가능성 점수 출력 코드를 설명합니다.

값	요약	설명
5	완전 확실	주소의 유효성이 검사되었고 입력 주소의 모든 관련 요소가 확인되었음을 나타냅니다.
4	거의 확실	다음 시나리오 중 하나를 나타냅니다. <ul style="list-style-type: none"> <li>- 참조 데이터가 없기 때문에 하나 이상의 관련 주소 요소를 검사할 수 없습니다. 다른 주소 요소는 확인되었습니다.</li> <li>- 주소 유효성 검사를 통해 하나 이상의 관련 주소 요소가 매우 높은 신뢰도로 수정되었습니다. 이 시나리오는 주소 유효성 검사에서 입력 주소와 참조 데이터 사이에서 단일의 일치 항목이 발견되었고 변형 수준이 매우 낮은 경우 발생합니다.</li> </ul>
3	양호함	주소 유효성 검사에서 입력 주소의 관련 요소 중 하나 이상이 수정되었음을 나타냅니다. 주소 유효성 검사에서 입력 주소와 참조 데이터 사이에서 단일의 일치 항목이 발견되었고 변형 수준이 허용 가능한 수준입니다.
2	가능함	주소 유효성 검사에서 다음과 같은 이유 중 하나로 주소를 수정하거나 확인할 수 없음을 나타냅니다. <ul style="list-style-type: none"> <li>- 주소 유효성 검사에서 충분한 신뢰도로 참조 데이터에서 일치 후보를 식별할 수 없습니다.</li> <li>- 주소 유효성 검사에서 유사한 신뢰도 수준으로 다수의 일치 후보가 발견되었습니다.</li> </ul> 우편 운송업체는 이 주소로 우편을 배달할 수 있을지도 모릅니다.
1	위험함	주소 유효성 검사에서 입력 주소에 대해 부분적인 참조 데이터 일치만 찾을 수 있음을 나타냅니다.
0	배달할 수 없음	주소 유효성 검사에서 주소와 일치하는 항목을 참조 데이터에서 찾을 수 없음을 나타냅니다. 입력 주소에 누락된 항목이 너무 많거나 주소 유효성 검사에서 주소 요소의 대부분을 확인할 수 없습니다.

## 일치 코드 출력 포트 값

일치 코드 값은 입력 주소와 참조 데이터의 비교 결과를 요약합니다. 이 코드는 또한 변환이 주소에 수행한 모든 수정을 요약합니다. 이 포트는 상태 정보 포트 그룹에서 찾을 수 있습니다.

다음 테이블에는 일치 코드 출력 포트 값이 설명되어 있습니다.

코드	설명
A1	주소 코드 조회로 입력 코드에 대한 부분적 주소 또는 전체 주소를 찾았습니다.
A0	주소 코드 조회로 입력 코드에 대한 주소를 찾지 못했습니다.
C4	수정됨. 모든 우편 관련 요소가 확인되었습니다.
C3	수정됨. 일부 요소를 확인할 수 없습니다.
C2	수정되었지만 참조 데이터가 없으므로 배달 상태가 확실하지 않습니다.
C1	수정되었지만 사용자 표준화로 인한 오류가 유발되어 배달 상태가 확실하지 않습니다.

코드	설명
I4	데이터는 완전하게 수정될 수 없지만 참조 데이터에는 단일 주소와 일치하는 단일 항목이 있습니다.
I3	데이터는 완전하게 수정될 수 없으며 참조 데이터에는 여러 주소와 일치하는 여러 항목이 있습니다.
I2	데이터를 수정할 수 없습니다. 일괄 모드는 부분적인 제안 주소를 반환합니다.
I1	데이터를 수정할 수 없습니다. 일괄 모드는 주소를 제안할 수 없습니다.
N7	유효성 검사 오류가 발생했습니다. 단일 행 유효성 검사가 잠금 해제되지 않았으므로 유효성 검사가 수행되지 않았습니다.
N6	유효성 검사 오류가 발생했습니다. 대상 국가에 대한 단일 행 유효성 검사가 지원되지 않았으므로 유효성 검사가 수행되지 않았습니다.
N5	유효성 검사 오류가 발생했습니다. 참조 데이터베이스가 오래되어 유효성 검사가 수행되지 않았습니다.
N4	유효성 검사 오류가 발생했습니다. 참조 데이터가 손상되거나 잘못된 형식이므로 유효성 검사가 수행되지 않았습니다.
N3	유효성 검사 오류가 발생했습니다. 국가 데이터의 잠금이 해제될 수 없으므로 유효성 검사가 수행되지 않았습니다.
N2	유효성 검사 오류가 발생했습니다. 필요한 참조 데이터베이스를 사용할 수 없으므로 유효성 검사가 수행되지 않았습니다.
N1	유효성 검사 오류가 발생했습니다. 국가가 인식되지 않거나 지원되지 않아 유효성 검사가 수행되지 않았습니다.
Q3	제안 목록 모드. 주소 유효성 검사는 입력 주소에 해당하는 주소 참조 데이터에서 하나 이상의 완전한 주소를 검색할 수 있습니다.
Q2	제안 목록 모드. 주소 유효성 검사는 완전한 주소를 생성하기 위해 입력 주소 요소와 주소 참조 데이터의 요소를 결합할 수 있습니다.
Q1	제안 목록 모드. 주소 유효성 검사는 완전한 주소를 제안할 수 없습니다. 완전한 주소 제안을 생성하려면 데이터를 입력 주소에 추가하십시오.
Q0	제안 목록 모드. 제안을 생성하기에는 입력 데이터가 부족합니다.
RB	국가는 약어로 인식됩니다. ISO 두 문자 및 ISO 세 문자 국가 코드를 인식합니다. 독일의 "GER"과 같은 일반적 약어로도 국가를 인식할 수 있습니다.
RA	변환의 국가 강제 적용 설정에서 국가가 인식되었습니다.
R9	변환의 기본 국가 설정에서 국가가 인식되었습니다.
R8	국가 이름에서 국가가 인식되었습니다.
R7	국가명에서 국가가 인식되었으나 변환이 국가 데이터에서 오류를 식별했습니다.
R6	지역 데이터에서 국가가 인식되었습니다.

코드	설명
R5	시/도 데이터에서 국가가 인식되었습니다.
R4	주요 타운 데이터에서 국가가 인식되었습니다.
R3	주소 형식에서 국가가 인식되었습니다.
R2	스크립트에서 국가가 인식되었습니다.
R1	사용할 수 있는 일치 항목이 여러 개이므로 국가가 인식되지 않았습니다.
R0	국가가 인식되지 않았습니다.
S4	구문 분석 모드. 주소가 완전하게 구문 분석되었습니다.
S3	구문 분석 모드. 주소 구문 분석을 통해 다수의 결과가 도출되었습니다.
S1	구문 분석 모드. 입력 형식 불일치로 인해 구문 분석 오류가 발생했습니다.
V4	확인됨. 입력 데이터가 정확합니다. 주소 유효성 검사는 모든 우편 관련 요소를 확인했고 입력은 완전하게 일치되었습니다.
V3	확인됨. 입력 데이터는 올바르지만 일부 또는 모든 요소가 표준화되었거나 입력에 오래된 이름 또는 외국어 지명이 포함되어 있습니다.
V2	확인됨. 입력 데이터는 올바르지만 참조 데이터가 완전하지 않아 일부 요소를 확인할 수 없습니다.
V1	확인됨. 입력 데이터는 올바르지만 사용자 표준화가 배달 가능성에 부정적인 영향을 미쳤습니다. 예를 들어 우편 번호 길이가 너무 짧습니다.

## 좌표 상태 출력 포트 값

다음 표에는 좌표 상태 출력 포트 값이 설명되어 있습니다. 이 포트는 좌표 포트 그룹에서 찾을 수 있습니다. 입력 주소 국가에 대한 좌표 참조 데이터를 설치한 경우 이 포트를 선택하십시오.

값	설명
EGC0	해당 주소에 대한 좌표가 제공되지 않아 입력 주소에 좌표를 추가할 수 없습니다.
EGC1-3	나중에 사용하기 위해 예약됨.
EGC4	좌표가 우편 번호 수준까지 부분적으로 정확합니다.
EGC5	좌표가 우편 번호 수준까지 정확합니다.
EGC6	좌표가 로컬리티 수준까지 정확합니다.
EGC7	좌표가 거리 수준까지 정확합니다.
EGC8	좌표가 집 번호 수준까지 정확합니다. 좌표가 집 번호 위치를 예상하고 사서함이 포함된 쪽 거리에 대한 오프셋을 포함합니다.

값	설명
EGC9	좌표가 도착 지점 또는 옥상까지 정확합니다.
EGCA	좌표가 구획의 중심까지 정확합니다.
EGCC	좌표 데이터베이스가 손상되었습니다.
EGCN	좌표 데이터베이스를 찾을 수 없습니다.
EGCU	좌표 데이터베이스의 잠금이 해제되지 않았습니다.

**참고:** Informatica는 구획 중심 및 옥상 좌표에 대한 참조 데이터를 더 이상 발행하지 않습니다.

## 주소 유효성 검사기 변환의 일반 설정

일반 설정을 구성하여 주소 유효성 검사에 필요한 매개 변수를 설정합니다.

**일반 설정** 보기에서 다음 속성을 구성할 수 있습니다.

### 기본 국가

변환 시 입력 주소에서 대상 국가를 확인할 수 없는 경우에 사용할 주소 참조 데이터 집합을 지정합니다. 데이터에 국가 정보가 포함되는 경우 없음을 선택합니다.

기본 국가를 변환에서 고급 속성으로 설정할 수도 있습니다.

### 국가 강제 적용

선택적 속성입니다. 입력 주소의 국가 이름 또는 약어를 기본 국가의 이름 또는 약어로 바꿉니다. 입력 주소에서 국가를 식별하지 않으면 변환이 기본 국가 데이터를 주소에 추가합니다.

### 행 구분 기호

한 행으로 된 주소에서 데이터 필드를 구분할 구분자 기호를 지정합니다.

행 구분 기호를 변환에서 고급 속성으로 지정할 수도 있습니다.

### 대/소문자 구분 스타일

출력 데이터에 대한 대/소문자 스타일을 설정합니다. 첫 대문자에 대해 주소 참조 데이터 표준을 따르려면 혼합된 옵션을 선택합니다. 주소 참조 데이터가 사용하는 대/소문자 스타일로 주소를 기록하려면 유지됨 옵션을 선택합니다.

대/소문자 구분 스타일을 변환에서 고급 속성으로 설정할 수도 있습니다.

### 모드

변환이 수행하는 유효성 검사의 유형을 결정합니다.

다음 옵션 중 하나를 선택합니다.

모드 유형	설명
우편 번호 조회	우편 번호를 입력으로 제공하면 참조 데이터에서 부분 또는 전체 주소가 반환됩니다. 일부 국가는 주소의 로컬리티, 거리 이름, 건물 또는 고유한 사서함을 나타내는 우편 번호를 지원합니다.
일괄	데이터 집합의 레코드에 대해 주소 유효성 검사를 수행합니다. 일괄 유효성 검사는 주소의 완전성 및 배달 가능성을 중점으로 유효성을 검사합니다. 일괄 모드는 품질이 낮은 주소에 대한 제안을 반환하지 않습니다. 일괄이 기본 모드입니다.
인증됨	지정된 국가의 인증 표준을 기준으로 데이터 집합의 레코드에 대해 주소 유효성 검사를 수행합니다. 인증 표준을 충족하려면 각 주소에 고유한 사서함이 포함되어야 합니다. 인증된 주소 유효성 검사는 오스트레일리아, 프랑스, 뉴질랜드, 영국 및 미국의 주소에 수행할 수 있습니다.
국가 인식	우편 주소의 대상 국가를 결정합니다. 국가 인식 모드에서는 변환이 주소 유효성 검사를 수행하지 않습니다.
대화형	완전하지 않은 유효한 주소를 완전하게 만듭니다. 완전하지 않은 입력 주소와 일치하는 주소가 참조 데이터에 2개 이상 있는 경우 변환이 유효한 모든 주소를 최대 결과 수에 지정된 한도까지 반환합니다.
구문 분석	데이터를 주소 필드로 구문 분석합니다. 구문 분석 모드에서는 변환이 주소 유효성 검사를 수행하지 않습니다.
제안 목록	입력 주소에 부분적인 정보가 포함된 경우 참조 데이터에서 유효한 주소 목록을 반환합니다. 주소의 일부와 일치하는 주소가 참조 데이터에 2개 이상 있는 경우 변환이 유효한 모든 주소를 최대 결과 수에 지정된 한도까지 반환합니다.

모드를 변환에서 고급 속성으로 설정할 수도 있습니다.

## 기본 설정 창의 주소 유효성 검사 속성

주소 유효성 검사 엔진의 속성과, 엔진이 **Developer tool**에서 읽는 주소 참조 데이터 파일을 볼 수 있습니다. **Developer tool**에는 데이터 통합 서비스에서 주소 유효성 검사 매핑을 실행할 때 사용하는 엔진의 속성이 표시됩니다. **Developer tool**은 주소 유효성 검사 작업을 제어하는 콘텐츠 관리 서비스의 속성을 나열합니다.

속성을 검토하려면 **Developer tool**에서 **기본 설정** 창을 사용합니다. **기본 설정** 창의 **콘텐츠 상태** 옵션을 선택하면 현재 데이터 통합 서비스에서 사용하는 콘텐츠 관리 서비스를 식별할 수 있습니다. 속성을 보려면 해당 지역의 콘텐츠 관리 서비스를 선택합니다.

다음 속성을 볼 수 있습니다.

### 주소 유효성 검사 데이터

주소 유효성 검사 데이터 속성은 현재 콘텐츠 관리 서비스에서 데이터 통합 서비스에 제공할 수 있는 참조 데이터의 유형을 나열합니다. 또한 이 속성에는 참조 데이터가 적용되는 국가가 표시됩니다.

### 주소 유효성 검사 엔진

주소 유효성 검사 엔진 속성에는 현재 엔진 버전, 인증 구성 요소가 최근에 업데이트된 엔진 및 데이터 사전 로드 방법이 포함됩니다.

## 주소 유효성 검사 라이선스

주소 유효성 검사 라이선스 속성에는 현재 콘텐츠 관리 서비스에서 데이터 통합 서비스에 제공할 수 있는 참조 데이터에 대한 라이선스 정보가 포함됩니다.

## 주소 유효성 검사 데이터 속성

주소 유효성 검사 데이터 속성은 현재 콘텐츠 관리 서비스에서 데이터 통합 서비스에 제공할 수 있는 참조 데이터의 유형을 나열합니다. 또한 속성에는 참조 데이터가 적용되는 국가가 포함됩니다.

다음 테이블에는 **콘텐츠 상태** 보기에서 콘텐츠 관리 서비스를 선택할 때 표시되는 데이터 속성이 설명되어 있습니다.

속성	설명
국가 ISO	주소 참조 데이터 파일이 적용되는 국가입니다. 이 속성에는 해당 국가에 대한 ISO 세 문자 코드가 표시됩니다.
만료 날짜	현재 파일이 만료되는 날짜입니다. Informatica는 만료 날짜에 최신 파일을 릴리스합니다. 만료 날짜 후에 현재 주소 참조 데이터 파일을 사용할 수는 있지만 파일의 데이터가 더 이상 정확하지 않을 수 있습니다.
국가 유형	데이터로 수행할 수 있는 주소 유효성 검사 유형입니다. <b>일반 설정</b> 탭의 <b>모드</b> 옵션에서 처리 유형을 선택합니다. 선택하는 모드가 도메인의 주소 데이터 파일과 일치하지 않는 경우 주소 유효성 검사 매핑을 수행하지 못합니다.
만료 날짜 잠금 해제	라이선스가 만료되는 날짜입니다. 만료 날짜 잠금 해제 후에는 파일의 어떠한 버전도 사용할 수 없습니다. 주소 유효성 검사 라이선스 속성 보기에서 만료 날짜 잠금 해제 속성과 만료 날짜 속성은 동일한 정보를 나타냅니다.
시작 날짜 잠금 해제	국가 유형 속성이 식별하는 모드와 국가 ISO 속성이 식별하는 국가에 대해 라이선스가 발효되는 날짜입니다. 시작 날짜 잠금 해제 전에는 파일의 어떠한 버전도 사용할 수 없습니다.

## 주소 유효성 검사 라이선스 속성

주소 유효성 검사 라이선스 속성에는 현재 콘텐츠 관리 서비스에서 데이터 통합 서비스에 제공할 수 있는 참조 데이터에 대한 라이선스 정보가 포함됩니다.

다음 테이블에는 **콘텐츠 상태** 보기에서 콘텐츠 관리 서비스를 선택할 때 표시되는 라이선스 속성이 설명되어 있습니다.

속성	설명
잠금 해제 코드	코드 유형 속성이 식별하는 모드에 대한 참조 데이터를 잠금 해제하는 라이선스 코드입니다. Developer tool에 코드의 처음 4개 문자가 표시되며 다른 문자는 마스킹됩니다.
코드 유형	라이선스에 지정된 데이터를 사용하여 수행할 수 있는 주소 유효성 검사의 모드입니다. Informatica는 각 모드에 대한 단일 라이선스 코드를 발급합니다. 라이선스 코드는 하나 이상의 국가에 적용될 수 있습니다. <b>일반 설정</b> 탭의 <b>모드</b> 옵션에서 처리 유형을 선택합니다. 선택하는 모드가 도메인의 주소 데이터 파일과 일치하지 않는 경우 주소 유효성 검사 매핑을 수행하지 못합니다.
국가 목록	잠금 해제 코드가 잠금을 해제하는 참조 데이터의 국가입니다. 국가 목록 속성에는 각 국가에 대한 ISO 세 문자 코드가 하나 이상 포함되어 있습니다.

속성	설명
상태	라이선스 코드의 상태입니다. 라이선스 파일이 유효한 경우 이 속성은 확인을 반환합니다.
만료 날짜	라이선스가 만료되는 날짜입니다. 주소 유효성 검사 데이터 속성 보기의 만료 날짜 속성과 만료 날짜 잠금 해제 속성은 동일한 정보를 나타냅니다.

## 주소 유효성 검사 엔진 속성

주소 유효성 검사 엔진 속성에는 현재 엔진 버전, 인증 구성 요소가 최근에 업데이트된 엔진 및 데이터 사전 로드 방법이 포함됩니다.

다음 테이블에는 **콘텐츠 상태** 보기에서 콘텐츠 관리 서비스를 선택할 때 표시되는 엔진 속성이 설명되어 있습니다.

속성	값
엔진 버전	데이터 통합 서비스가 실행하는 주소 유효성 검사 엔진의 버전입니다.
CASS 버전	Informatica에서 가장 최근에 CASS 인증 구성 요소를 업데이트한 주소 유효성 검사 엔진의 버전입니다. 이 속성을 사용하면 CASS 인증 보고서의 엔진 버전을 식별할 수 있습니다. 엔진이 지원하는 CASS 인증 주기도 속성에 포함됩니다. 예를 들어 엔진은 인증 주기 N을 지원할 수 있습니다.
AMAS 버전	Informatica에서 가장 최근에 AMAS 인증 구성 요소를 업데이트한 주소 유효성 검사 엔진의 버전입니다. 이 속성을 사용하면 AMAS 인증 보고서의 엔진 버전을 식별할 수 있습니다.
SendRight 버전	Informatica에서 가장 최근에 SendRight 인증 구성 요소를 업데이트한 주소 유효성 검사 엔진의 버전입니다. 이 속성을 사용하면 SendRight 인증 보고서의 엔진 버전을 식별할 수 있습니다.
SERP 버전	Informatica에서 가장 최근에 SERP 인증 구성 요소를 업데이트한 주소 유효성 검사 엔진의 버전입니다. 이 속성을 사용하면 SERP 인증 보고서의 엔진 버전을 식별할 수 있습니다.
SNA 버전	Informatica에서 가장 최근에 SNA 인증 구성 요소를 업데이트한 주소 유효성 검사 엔진의 버전입니다. 이 속성을 사용하면 SNA 인증 보고서의 엔진 버전을 식별할 수 있습니다.
사전 로드 방법	데이터 통합 서비스에서 참조 데이터베이스를 메모리에 사전 로드할 때 사용하는 방법입니다. 데이터 통합 서비스에서 사전 로드하는 참조 데이터의 국가는 콘텐츠 관리 서비스 속성으로 지정됩니다. 가능한 값은 MAP 및 LOAD입니다. 기본값은 MAP입니다. MAP 방법과 LOAD 방법은 둘 다 메모리 블록을 할당한 다음 참조 데이터를 블록으로 읽습니다. 그러나 MAP 방법은 여러 프로세스 간에 참조 데이터를 공유할 수 있습니다.
캐시 크기	데이터 통합 서비스에서 사전 로드하지 않는 참조 데이터에 사용하는 데이터 캐시의 크기입니다. 가능한 값은 NONE, SMALL 및 LARGE입니다. 기본값은 LARGE입니다.
최대 메모리 사용량	주소 유효성 검사 엔진에서 할당할 수 있는 메모리(MB)입니다. 기본값은 4096입니다.
최대 주소 개체 개수	데이터 통합 서비스에서 동시에 실행할 수 있는 주소 유효성 검사 인스턴스의 최대 수입니다. 기본값은 3입니다.
최대 스레드 개수	주소 유효성 검사에서 사용할 수 있는 최대 스레드 개수입니다. 기본값은 2입니다.



속성	값
최대 결과 수	제안 목록 모드에서 매핑을 실행할 때 주소 유효성 검사에서 반환할 수 있는 최대 주소 수입니다. 기본값은 20입니다. 속성의 상한은 100입니다.
현재 날짜	현재 날짜입니다. Developer tool은 현재 날짜에 적용되는 속성 값을 반환합니다.
XML BOM 쓰기	데이터 통합 서비스에서 GetConfig.xml 파일에 바이트 순서 표식을 쓸지 여부를 나타냅니다. 가능한 값은 ALWAYS, IF_NECESSARY 및 NEVER입니다. 기본값은 IF_NECESSARY입니다.
XML 인코딩	주소 유효성 검사 엔진에서 데이터를 읽고 쓸 때 사용하는 XML 인코딩을 나타냅니다.

## 주소 유효성 검사 고급 속성

주소 유효성 검사기 변환 데이터에 대한 데이터 통합 서비스의 처리 방식을 결정하는 고급 속성을 구성할 수 있습니다.

### 별칭 로컬리티

주소 유효성 검사가 유효한 로컬리티 별칭을 공식적인 로컬리티 이름으로 대체할지 여부를 결정합니다.

로컬리티 별칭은 USPS가 배달 가능한 주소에서 요소로 인식하는 대체 로컬리티 이름입니다. 인증 모드에서 미국 주소 레코드에 대한 유효성을 검사하도록 주소 유효성 검사기 변환을 구성할 경우 이 속성을 사용할 수 있습니다.

다음 테이블에는 별칭 로컬리티 옵션이 설명되어 있습니다.

옵션	설명
꺼짐	별칭 로컬리티 속성을 비활성화합니다.
공식	대체 로컬리티 이름이나 로컬리티 별칭을 공식적인 로컬리티 이름으로 바꿉니다. 기본값 옵션.
유지	유효한 다른 로컬리티 이름 또는 로컬리티 별칭을 유지합니다. 입력 로컬리티 이름이 유효하지 않은 경우 주소 유효성 검사가 이름을 공식적인 이름으로 바꿉니다.

### 별칭 거리

주소 유효성 검사가 거리 별칭을 공식적인 거리 이름으로 대체할지 여부를 결정합니다.

거리 별칭은 USPS가 배달 가능한 주소의 요소로 인식하는 다른 거리 이름입니다. 인증 모드에서 미국 주소 레코드에 대한 유효성을 검사하도록 주소 유효성 검사기 변환을 구성할 경우 이 속성을 사용할 수 있습니다.

다음 테이블에는 별칭 거리 옵션이 설명되어 있습니다.

옵션	설명
꺼짐	해당 속성을 적용하지 않습니다.
공식	다른 거리 이름이나 거리 별칭을 모두 공식적인 거리 이름으로 바꿉니다. 기본값 옵션.
유지	유효한 다른 거리 이름 또는 거리 별칭을 유지합니다. 입력 거리 이름이 유효하지 않은 경우 주소 유효성 검사가 이름을 공식적인 이름으로 바꿉니다.

## 대/소문자 구분 스타일

변환이 출력 주소 데이터에 적용하는 대/소문자 구분 스타일을 지정합니다.

다음 테이블에는 대/소문자 구분 스타일 옵션이 설명되어 있습니다.

옵션	설명
매개 변수 할당	정의하는 매개 변수를 사용하여 대/소문자 구분 스타일을 설정합니다.
낮은 값	출력 주소를 소문자로 씁니다.
대/소문자	가능한 대상 국가에서 사용하는 대/소문자 구분 스타일을 사용합니다.
유지됨	주소 참조 데이터가 사용하는 대/소문자 구분 스타일을 적용합니다. 기본값 옵션입니다.
변경 없음	주소에 대/소문자 구분 스타일을 적용하지 않습니다. <b>참고:</b> 변경 없음 옵션은 출력 주소가 입력 주소의 대/소문자와 일치할 것으로 보장하지 않습니다. 주소 유효성 검사가 주소 요소를 참조 데이터의 요소로 바꾸는 경우 요소는 참조 데이터가 사용하는 대/소문자를 따릅니다.
대문자	출력 주소를 대문자로 씁니다.

또한 **일반 설정** 탭에서 대/소문자 구분 스타일을 구성할 수도 있습니다.

### 매개 변수 사용

다음 매개 변수 중 하나를 사용하여 대/소문자 구분 스타일을 지정할 수 있습니다.

- **LOWER.** 출력 주소를 소문자로 씁니다.
- **MIXED.** 가능한 대상 국가에서 사용하는 대/소문자 구분 스타일을 사용합니다.
- **NATIVE.** 주소 참조 데이터가 사용하는 대/소문자 구분 스타일을 적용합니다. 기본값 옵션입니다. **유지됨** 옵션과 일치합니다.
- **NOCHANGE.** 주소에 대/소문자 구분 스타일을 적용하지 않습니다.
- **UPPER.** 출력 주소를 대문자로 씁니다.

매개 변수 값을 대문자로 입력합니다.

## 출생지

주소 레코드가 우편으로 배달되는 국가를 식별합니다.

목록에서 국가를 선택합니다. 속성은 기본적으로 비어 있습니다.

## 국가 유형

완전한 주소 또는 형식이 지정된 주소 행 포트 출력 데이터에서 국가 이름 또는 약어 형식을 결정합니다. 변환은 선택하는 국가에 대한 표준 형식으로 국가 이름 또는 약어를 기록합니다.

다음 테이블에는 국가 유형 옵션이 설명되어 있습니다.

옵션	국가
ISO 2	ISO 두 문자 국가 코드
ISO 3	ISO 세 문자 국가 코드
ISO 번호	ISO 세 자리 국가 코드
약어	(나중에 사용하기 위해 예약됨)
CN	캐나다
DA	(나중에 사용하기 위해 예약됨)
DE	독일
EN	영국(기본값)
ES	스페인
FI	핀란드
FR	프랑스
GR	그리스
IT	이탈리아
JP	일본
HU	헝가리
KR	대한민국
NL	네덜란드
PL	폴란드
PT	포르투갈
RU	러시아

옵션	국가
SA	사우디아라비아
SE	스웨덴

## 기본 국가

주소 레코드가 대상 국가를 식별하지 않을 때 변환에서 사용하는 주소 참조 데이터 집합을 지정합니다.

목록에서 국가를 선택합니다. 주소 레코드에 국가 정보가 포함될 경우 기본 옵션을 사용합니다. 기본값은 **None**입니다.

또한 **일반 설정** 탭에서 기본 국가를 구성할 수도 있습니다.

### 매개 변수 사용

매개 변수를 사용하여 기본 국가를 지정할 수 있습니다. 매개 변수를 생성할 때 해당 국가에 대한 ISO 3166-1 alpha-3 코드를 매개 변수 값으로 입력합니다. 매개 변수 값을 입력할 때 대문자를 사용합니다. 예를 들어 모든 주소 레코드에 국가 정보가 포함되어 있을 경우 **NONE**를 입력합니다.

## 이중 주소 우선 순위

유효성 검사를 수행할 주소 유형을 결정합니다. 입력 주소 레코드에 두 개 이상의 유효한 주소 데이터 유형이 포함되어 있으면 해당 속성을 설정합니다.

예를 들어 주소 레코드에 사서함 요소 및 거리 요소가 모두 포함되어 있으면 해당 속성을 사용합니다. 주소 유효성 검사는 지정하는 주소 데이터 유형을 포함하는 데이터 요소를 읽습니다. 주소 유효성 검사는 주소에서 호환되지 않는 모든 데이터를 무시합니다.

다음 테이블에는 이중 주소 우선 순위 속성에 대한 옵션이 설명되어 있습니다.

옵션	설명
배달 서비스	사서함 요소 등 주소의 배달 서비스 데이터 요소에 대한 유효성을 검사합니다.
우편 관리	지역 우체국에서 필요로 하는 주소 요소에 대한 유효성을 검사합니다. 기본값 옵션.
거리	건물 번호 요소 및 거리 이름 요소 등 주소의 거리 데이터 요소에 대한 유효성을 검사합니다.

## 요소 약어

변환이 주소 요소의 축약된 형식을 반환하는지 결정합니다. 주소 참조 데이터에 약어가 포함되어 있는 경우 변환이 축약된 형식을 반환하도록 설정할 수 있습니다.

예를 들어 USPS(United States Postal Service)는 짧고 긴 형식의 많은 거리 및 로컬리티 이름을 유지합니다. HUNTSVILLE BROWNSFERRY RD의 짧은 형식은 HSV BROWNS FRY RD입니다. 거리 또는 로컬리티 값이 USPS에서 지정하는 최대 필드 길이를 초과하면 요소 약어 속성을 선택할 수 있습니다.

옵션은 기본적으로 지워져 있습니다. 축약된 주소 값을 반환하려면 속성을 **ON**으로 설정합니다. 일괄 모드에서 변환을 사용하면 속성은 축약된 로컬리티 이름 및 로컬리티 코드를 반환합니다. 인증 모드에서 변환을 사용하면 속성은 축약된 거리 이름, 로컬리티 이름 및 로컬리티 코드를 반환합니다.

## 실행 인스턴스

데이터 통합 서비스가 런타임 시 현재 변환에 대해 작성을 시도하는 스레드 수를 지정합니다. 변환이 포함된 맵핑에서 최대 병렬도 런타임 속성을 재정의하는 경우 데이터 통합 서비스가 실행 인스턴스 값을 고려합니다. 기본 실행 인스턴스 값은 1입니다.

데이터 통합 서비스는 여러 요인을 고려하여 변환에 할당할 스레드 수를 결정합니다. 사용자 요인은 실행 인스턴스 값, 매핑 및 도메인의 연결된 응용 프로그램 서비스의 값입니다.

데이터 통합 서비스는 변환에 대해 사용할 스레드 수를 계산할 때 다음 값을 읽습니다.

- 데이터 통합 서비스의 *최대 병렬도* 값. 기본값은 1입니다.
- 매핑 수준에서 설정하는 *최대 병렬도* 값. 기본값은 자동입니다.
- 변환의 *실행 인스턴스* 값. 기본값은 1입니다.

매핑 수준에서 최대 병렬도 값을 재정의하는 경우 데이터 통합 서비스가 속성에서 가장 낮은 값을 사용 시도하여 스레드 수를 결정합니다.

매핑 수준에서 기본 최대 병렬도 값을 사용하는 경우 데이터 통합 서비스가 실행 인스턴스 값을 무시합니다.

또한 데이터 통합 서비스는 작성할 스레드 수를 계산할 때 콘텐츠 관리 서비스의 *최대 주소 개체 개수* 속성을 고려합니다. *최대 주소 개체 개수* 속성은 매핑에서 동시에 실행할 수 있는 주소 유효성 검사 인스턴스의 최대 수를 결정합니다. *최대 주소 개체 개수* 속성 값은 데이터 통합 서비스의 *최대 병렬도* 값보다 크거나 같아야 합니다.

### 실행 인스턴스 속성에 대한 규칙 및 지침

실행 인스턴스 수를 설정할 때는 다음 규칙 및 지침을 고려합니다.

- 데이터 통합 서비스에서는 여러 사용자가 동시에 매핑을 실행할 수 있습니다. 올바른 스레드 수를 계산하려면 서비스가 액세스할 수 있는 중앙 처리 장치의 수를 동시에 매핑의 수로 나눕니다.
- PowerCenter에서 *AD50.cfg* 구성 파일은 매핑에서 동시에 실행할 수 있는 주소 유효성 검사 인스턴스의 최대 수를 지정합니다.
- 기본 실행 인스턴스 값 및 기본 최대 병렬도 값을 사용하는 경우 변환 작업은 분할 가능하지 않습니다.
- 실행 인스턴스 값을 1보다 큰 값으로 설정하는 경우 주소 유효성 검사기 변환을 수동 변환에서 활성 변환으로 변경합니다.

## 유동적 범위 확장

확장할 범위 속성을 설정할 때 주소 유효성 검사기 변환이 반환하는 주소 수에 대한 실질적인 한도를 지정합니다. 제안 목록 모드에서 실행할 변환을 구성할 때 확장할 범위 속성 및 유동적 범위 확장 속성을 설정할 수 있습니다.

확장할 범위 속성은 입력 주소에 집 번호 데이터가 포함되어 있지 않을 때 변환에서 주소 제안을 반환하는 방식을 결정합니다. 입력 주소에 전체 우편 번호와 같은 컨텍스트 데이터가 포함되어 있지 않을 경우 확장할 범위 속성이 매우 유사한 주소를 많이 생성할 수 있습니다. 유동적 범위 확장 속성은 단일 주소에 대해 확장할 범위 속성이 생성하는 주소 수를 제한합니다. 확장할 범위 속성이 All로 설정되어 있으면 유동적 범위 확장 속성을 On으로 설정합니다.

다음 테이블에는 유동적 범위 확장 속성에 대한 옵션이 설명되어 있습니다.

옵션	설명
켜짐	주소 유효성 검사는 확장할 범위 속성이 제안 목록에 추가되는 주소 수를 제한합니다. 기본값 옵션.
꺼짐	주소 유효성 검사는 확장할 범위 속성이 제안 목록에 추가되는 주소 수를 제한하지 않습니다.

**참고:** 주소 유효성 검사기 변환은 제안 목록으로 반환하는 모든 주소에 다른 방식으로 유동적 범위 확장 속성을 적용합니다. 변환은 목록에서 확장된 주소 수에 대한 고정 한도를 지정하지 않습니다. 또한 목록에 포함할 확장된 주소 수를 계산할 때 최대 결과 수 속성 설정도 고려합니다.

## 좌표 부여 데이터 유형

주소 유효성 검사기 변환이 주소에 대한 좌표 부여 데이터를 계산하는 방식을 결정합니다. 좌표 부여는 위도 및 경도 좌표입니다.

변환이 반환하는 좌표 부여 결과는 설치된 좌표 부여 참조 데이터에 따라 다릅니다. 좌표 부여 참조 데이터에 대한 자세한 내용은 **Informatica**에 문의하십시오.

다음과 같은 좌표 옵션 중 하나를 선택할 수 있습니다.

### 도착 지점

건물이나 구획의 입구에 대한 위도 및 경도 좌표를 반환합니다. 기본값 옵션입니다.

다음 국가의 주소에 대해 도착 지점 옵션을 선택할 수 있습니다.

오스트레일리아, 오스트리아, 캐나다, 크로아티아, 에스토니아, 핀란드, 프랑스, 독일, 헝가리, 이탈리아, 라트비아, 리히텐슈타인, 리투아니아, 룩셈부르크, 멕시코, 모나코, 네덜란드, 노르웨이, 폴란드, 슬로바키아, 슬로베니아, 스웨덴, 스위스 및 미국.

도착 지점 좌표를 지정한 경우에 주소 유효성 검사기 변환에서 주소의 좌표를 반환할 수 없으면 변환에서는 삽입된 좌표를 반환합니다.

### 구획 중심

지대 수준에서 구획의 지리적 중심에 대한 위도 및 경도 좌표를 반환합니다.

다음 국가의 주소에 대해 구획 중심 옵션을 선택할 수 있습니다.

오스트리아, 캐나다, 덴마크, 핀란드, 독일, 헝가리, 라트비아, 룩셈부르크, 네덜란드, 노르웨이, 슬로베니아, 스웨덴, 미국

구획 중심 좌표를 지정한 경우에 주소 유효성 검사기 변환에서 주소의 좌표를 반환할 수 없으면 변환에서는 좌표 데이터를 아예 반환하지 않습니다.

### 옥상

우편함이 있는 건물의 실제 중심을 식별하는 위도 및 경도 좌표를 반환합니다. 주소 유효성 검사기 변환은 아일랜드 및 영국 주소에 대해 옥상 좌표를 반환합니다.

옥상 좌표를 지정한 경우에 주소 유효성 검사기 변환에서 주소의 좌표를 반환할 수 없으면 변환에서는 좌표 데이터를 아예 반환하지 않습니다.

### 표준

건물이나 구획의 입구에 대한 예상 위도 및 경도 좌표를 반환합니다. 예상 좌표는 삽입된 좌표라고도 합니다.

주소 유효성 검사기 변환에서는 참조 데이터에서 사용 가능한 가장 가까운 좌표를 사용하여 주소의 좌표를 예상합니다.

**참고:** Informatica는 구획 중심 및 옥상 좌표에 대한 참조 데이터를 더 이상 발행하지 않습니다.

### 매개 변수 사용

매개 변수를 사용하여 좌표 부여 유형을 지정할 수 있습니다. **ARRIVAL\_POINT**, **PARCEL\_CENTROID**, **ROOFTOP** 또는 **NONE**을 입력합니다. 표준 좌표 부여를 반환하려면 **NONE**을 입력합니다.

매개 변수 값을 대문자로 입력합니다.

## 글로벌 최대 필드 길이

주소에서 모든 행의 최대 문자 수를 결정합니다. 주소 유효성 검사기 변환이 지정하는 것보다 더 많은 문자를 포함하는 출력 주소 행을 기록하는 경우 변환은 해당하는 행에서 주소 요소를 축약합니다.

이 속성을 사용하여 주소의 행 길이를 제어합니다. 예를 들어 **SNA** 표준에 따라 주소의 모든 행에는 38자 미만을 적용해야 합니다. **SNA** 표준에 대한 주소를 생성하는 경우 글로벌 최대 필드 길이를 38로 설정합니다.

기본값은 1024입니다.

### 매개 변수 사용

매개 변수를 사용하여 최대 주소 수를 지정할 수 있습니다. 매개 변수 값을 설정하려면 0 ~ 1024 사이의 정수를 입력합니다.

## 글로벌 기본 설정 설명자

주소 유효성 검사기 변환이 출력 데이터에 쓰는 건물 설명자, 하위 건물 설명자 및 거리 설명자의 형식을 결정합니다. 대상 국가의 주소 참조 데이터에 하나 이상의 데이터 요소에 대한 일련의 설명자가 포함되어 있는 경우 하나의 설명자를 선택합니다.

다음 테이블에는 속성에 대한 옵션이 설명되어 있습니다.

옵션	설명
데이터베이스	주소의 요소에 대해 참조 데이터베이스가 지정하는 설명자를 반환합니다. 데이터베이스에서 주소에 대한 설명자를 지정하지 않으면 변환이 입력 값을 출력 주소로 복사합니다. 데이터베이스가 기본값입니다.
긴 정수	설명자의 완전한 형식을 반환합니다(예: <i>Street</i> ).
입력 유지	설명자를 입력 주소에서 출력 주소로 복사합니다. 입력 설명자가 설명자의 유효한 버전이 아닌 경우 변환이 참조 데이터베이스에서 이와 동등한 유효한 설명자를 반환합니다.
짧게	설명자의 축약된 형식을 반환합니다(예: <i>St</i> ).

## 입력 형식 유형

필드가 지정되지 않은 입력 데이터에 포함된 가장 일반적인 정보 유형을 설명합니다. 입력 데이터를 완전한 주소 또는 형식이 지정된 주소 행 포트에 연결할 때 입력 형식 유형 속성을 사용합니다. 매핑 소스 데이터의 정보를 가장 잘 설명하는 옵션을 선택합니다.

다음 옵션 중 하나를 선택합니다.

- 모두
- 주소
- 조직
- 연락처
- 조직/연락처  
주소에는 조직 정보 및 연락처 정보가 포함됩니다.
- 조직/부서  
주소에는 조직 정보 및 부서 정보가 포함됩니다.

기본값은 All입니다.

## 국가가 포함된 입력 형식

입력에 국가 데이터가 포함되는지 여부를 지정합니다. 입력 데이터를 완전한 주소 또는 형식이 지정된 주소 행 입력 포트에 연결하고 데이터에 국가 정보가 포함되어 있는 경우 이 속성을 선택합니다.

옵션은 기본적으로 지워져 있습니다.

## 행 구분 기호

형식이 지정된 주소에서 줄 바꿈을 나타내는 구분자 기호를 지정합니다.

다음 옵션 중 하나를 선택합니다.

- 행 구분 기호를 식별하는 매개 변수 할당
- CR(캐리지 리턴)
- 쉼표
- LF(줄 바꿈)
- 없음
- 세미콜론
- 탭
- Windows 새 행(CRLF)

기본값은 Semicolon입니다.

또한 **일반 설정** 탭에서 행 구분 기호를 구성할 수도 있습니다.

### 매개 변수 사용

매개 변수를 사용하여 행 구분 기호를 지정할 수 있습니다. 매개 변수 값은 대/소문자를 구분합니다. 매개 변수 값을 대문자로 입력합니다.

다음 값 중 하나를 입력합니다.

- CR



- CRLF
- COMMA
- LF
- PIPE
- SEMICOLON
- SPACE
- TAB

## 일치 대체

주소 유효성 검사가 입력 주소에서 동의어 또는 역사적인 이름과 같은 대체 장소 이름을 인식하는지 여부를 결정합니다. 이 속성은 거리, 로컬리티 및 시/도 데이터에 적용됩니다.

**참고:** 일치 대체 속성은 유효성이 검사된 주소에서 대체 이름을 유지하지 않습니다.

다음 테이블에는 일치 대체 옵션이 설명되어 있습니다.

옵션	설명
모두	알려진 모든 대체 거리 이름 및 장소 이름을 인식합니다. 기본값 옵션.
보관만	역사적인 이름만 인식합니다. 예를 들어 주소 유효성 검사는 "콘스탄티노플"을 "이스탄불"의 역사적인 버전으로 유효성을 검사합니다.
없음	대체 거리 이름 또는 장소 이름을 인식하지 않습니다.
동의어만	동의어 및 외국어 지명만 인식합니다. 예를 들어 주소 유효성 검사는 "론드레스"를 "런던"의 외국어 지명으로 유효성을 검사합니다.

## 확장된 보관 일치

주소 유효성 검사가 오래된 일본 주소에 대한 고유한 배달 지점 코드를 반환하는지 여부를 결정합니다.

일본의 주소 참조 데이터 파일에는 해당하는 사서함에 대한 현재 주소와 함께 오래되거나 사용하지 않는 주소 관련 데이터가 포함되어 있습니다. 확장된 보관 일치 속성을 선택하면 주소 유효성 검사가 각 주소의 현재 버전에 대한 배달 지점 코드를 반환합니다. 또한 주소 유효성 검사는 입력 주소가 오래되었음을 나타내기 위해 확장 요소 결과 상태 포트에 값을 기록하기도 합니다.

주소 참조 데이터에서 현재 주소를 검색하려면 입력 요소로 주소 코드를 입력합니다.

다음 테이블에는 확장된 보관 일치 옵션이 설명되어 있습니다.

옵션	설명
꺼짐	해당 속성을 적용하지 않습니다.
켜짐	오래된 일본 주소의 현재 버전에 대한 주소 코드를 반환합니다.

확장된 보관 일치 속성은 일본에 대한 보조 데이터 및 주소 코드 조회 데이터를 사용합니다. 주소 유효성 검사에 이 속성을 적용하려면 주소 코드 조회 모드에서 실행하도록 변환을 구성해야 합니다.

## 일치 범위

주소 유효성 검사를 수행하는 동안 변환이 주소 참조 데이터를 기준으로 일치시키는 데이터 양을 결정합니다. 다음 테이블에는 일치 범위 옵션이 설명되어 있습니다.

옵션	설명
모두	선택한 모든 포트에 대한 유효성을 검사합니다. 기본값 옵션.
배달 지점	거리 옵션을 통해 유효성을 검사하는 데이터 외에 건물 및 하위 건물 주소 데이터에 대한 유효성도 검사합니다.
로컬리티	시/도, 로컬리티 및 우편 번호 데이터에 대한 유효성을 검사합니다.
거리	로컬리티 옵션을 통해 유효성을 검사하는 데이터 외에 거리 주소 데이터에 대한 유효성도 검사합니다.

## 최대 결과 수

주소 유효성 검사가 제안 목록 모드에서 반환할 수 있는 최대 주소 수를 결정합니다.

1에서 100까지 범위에서 최대 수를 설정할 수 있습니다. 기본값은 20입니다.

**참고:** 제안 목록 모드는 주소 참조 데이터를 기준으로 주소 확인을 수행하고 입력 주소와 일치 가능성이 있는 주소 목록을 반환합니다. 제안 목록 모드에서 주소를 확인하면 주소 유효성 검사는 가장 일치하는 항목을 먼저 반환합니다.

### 매개 변수 사용

매개 변수를 사용하여 최대 주소 수를 지정할 수 있습니다. 매개 변수 값을 설정하려면 0 ~ 100 사이의 정수를 입력합니다.

## 모드

변환이 수행하는 주소 분석 유형을 결정합니다. 또한 변환의 **일반 설정** 탭에서 이 모드를 구성할 수도 있습니다.

다음 테이블에는 모드 메뉴 옵션 및 설정할 수 있는 해당 매개 변수 값이 설명되어 있습니다.

모드 유형	설명
우편 번호 조회	우편 번호를 입력으로 제공하면 참조 데이터에서 부분 또는 전체 주소가 반환됩니다. 일부 국가는 주소의 로컬리티, 거리 이름, 건물 또는 고유한 사서함을 나타내는 우편 번호를 지원합니다.
일괄	데이터 집합의 레코드에 대해 주소 유효성 검사를 수행합니다. 일괄 유효성 검사는 주소의 완전성 및 배달 가능성을 중점으로 유효성을 검사합니다. 일괄 모드는 품질이 낮은 주소에 대한 제안을 반환하지 않습니다. 일괄이 기본 모드입니다.
인증됨	지정한 국가의 인증 표준을 기준으로 데이터 집합의 레코드에 대해 주소 유효성 검사를 수행합니다. 인증 표준을 충족하려면 각 주소에 고유한 사서함이 포함되어야 합니다. 인증된 주소 유효성 검사는 오스트레일리아, 프랑스, 뉴질랜드, 영국 및 미국의 주소에 수행할 수 있습니다.
국가 인식	우편 주소의 대상 국가를 결정합니다. 국가 인식 모드에서는 변환이 주소 유효성 검사를 수행하지 않습니다.

모드 유형	설명
대화형	완전하지 않은 유효한 주소를 완전하게 만듭니다. 완전하지 않은 입력 주소와 일치하는 주소가 참조 데이터에 2개 이상 있는 경우 변환이 유효한 모든 주소를 최대 결과 수에 지정된 한도까지 반환합니다.
구문 분석	데이터를 주소 필드로 구문 분석합니다. 구문 분석 모드에서는 변환이 주소 유효성 검사를 수행하지 않습니다.
제안 목록	입력 주소에 부분적인 정보가 포함된 경우 참조 데이터에서 유효한 주소 목록을 반환합니다. 주소의 일부와 일치하는 주소가 참조 데이터에 2개 이상 있는 경우 변환이 유효한 모든 주소를 최대 결과 수에 지정된 한도까지 반환합니다.

## 관련 항목:

- [“주소 유효성 검사기 변환의 일반 설정” 페이지 97](#)

## 최적화 수준

변환이 입력 주소 데이터와 주소 참조 데이터를 일치시키는 방식을 결정합니다. 이 속성은 변환이 주소 레코드를 업데이트하기 전에 입력 데이터와 참조 데이터 사이에서 찾아야 하는 일치 유형을 정의합니다.

다음 테이블에는 최적화 수준 옵션이 설명되어 있습니다.

옵션	설명
좁음	변환은 유효성 검사를 수행하기 전에 거리 정보에서 건물 번호 또는 집 번호를 구문 분석합니다. 또는 입력 포트 구조에 따라 입력 주소 요소에 대한 유효성을 엄격히 검사합니다. 좁음 옵션은 가장 빠른 주소 유효성 검사를 수행하지만 다른 옵션보다 덜 정확한 결과를 반환할 수 있습니다.
표준	변환은 유효성 검사를 수행하기 전에 입력 데이터에서 여러 가지 유형의 주소 정보를 구문 분석합니다. 표준 옵션을 선택하면 변환이 참조 데이터와 여러 개의 입력 값을 일치시킬 수 있는 경우 주소를 업데이트합니다. 기본값은 Standard입니다.
넓음	변환은 표준 구문 분석 설정을 사용하고 입력 데이터 전반에서 추가적인 구문 분석 작업을 수행합니다. 넓은 옵션을 선택하면 변환이 참조 데이터와 하나 이상의 입력 값을 일치시킬 수 있는 경우 주소를 업데이트합니다. 넓은 옵션은 매핑 실행 시간을 늘립니다.

## 매개 변수 사용

매개 변수를 사용하여 최적화 수준을 지정할 수 있습니다. NARROW, STANDARD 또는 WIDE를 입력합니다. 매개 변수 값을 대문자로 입력합니다.

## 출력 형식 유형

변환이 완전한 주소 또는 형식이 지정된 주소 행 출력 포트에 기록하는 가장 일반적인 정보 유형을 설명합니다. 출력 포트에서 예상되는 데이터를 가장 잘 설명한 옵션을 선택합니다.

다음 옵션 중 하나를 선택합니다.

- 모두

- 주소
  - 조직
  - 연락처
  - 조직/연락처  
주소에는 조직 정보 및 연락처 정보가 포함됩니다.
  - 조직/부서  
주소에는 조직 정보 및 부서 정보가 포함됩니다.
- 기본값은 All입니다.

## 국가가 포함된 출력 형식

변환이 완전한 주소 또는 형식이 지정된 주소 행 출력 포트에 국가 식별 데이터를 기록하는지 여부를 결정합니다.

옵션은 기본적으로 지워져 있습니다.

## 기본 설정 언어

참조 데이터 집합에 두 개 이상의 언어로 된 데이터가 포함되어 있을 때 주소 유효성 검사기 변환에서 주소 요소를 반환할 언어를 결정합니다. 벨기에, 캐나다, 중국, 핀란드, 홍콩, 아일랜드, 마카오, 스위스 및 대만의 주소에 대한 기본 설정 언어를 설정할 수 있습니다.

주소 유효성 검사기 변환은 다음 언어로 주소 데이터를 반환할 수 있습니다.

- 주소 참조 데이터의 주소에 대한 기본 언어. 기본 언어는 각 주소가 속하는 지역에서 주로 사용되는 언어입니다.
- 주소 참조 데이터가 주소에 대해 지원하는 기타 언어. 예를 들어 벨기에 참조 데이터는 플라망어, 프랑스어 및 독일어 주소 요소를 포함합니다.

주소 참조 데이터는 단일 주소 요소 또는 전체 주소에 대한 데이터를 여러 언어로 포함할 수 있습니다. 예를 들어 주소 유효성 검사는 아일랜드의 모든 주소 요소는 영어로 반환하고 거리, 로컬리티 및 시/도 정보는 아일랜드어로 반환할 수 있습니다. 또한 참조 데이터는 국가의 다른 지역 주소에 대해 다른 기본 주소를 지정할 수 있습니다. 예를 들어 스위스 참조 데이터의 기본 언어는 지역에 따라 프랑스어, 독일어, 이탈리아어로 변경됩니다.

다음 테이블에는 기본 설정 언어 속성에서 선택할 수 있는 옵션이 요약되어 있습니다.

옵션	설명
데이터베이스	각 주소가 주소 참조 데이터가 지정하는 언어로 반환됩니다. 주소 참조 데이터가 국가 내 서로 다른 지역의 주소에 대해 다른 언어를 지정할 수 있습니다. 데이터베이스가 기본 옵션입니다.
대체 1, 대체 2, 대체 3	참조 데이터의 대체 언어로 주소 요소를 반환합니다. 대체 언어는 주소가 속하는 국가에 따라 다릅니다.

옵션	설명
영어	참조 데이터가 데이터를 영어로 포함하는 경우 주소 요소를 영어로 반환합니다. 주소가 속하는 지역의 다른 주소 요소는 기본 언어로 반환됩니다.
입력 유지	주소 정보를 입력 언어로 반환합니다. 주소 참조 데이터에 입력 언어의 주소 정보가 포함되어 있는 경우 주소 유효성 검사에서 해당 언어가 유지됩니다. 주소 유효성 검사가 입력 주소에서 둘 이상의 지원되는 언어를 감지하는 경우 검사는 데이터베이스 언어로 주소를 반환합니다. 주소 유효성 검사에서 입력 언어로 요소를 반환할 수 없는 경우 데이터베이스 언어로 반환합니다.

**참고:** 주소 참조 데이터 집합에는 기본이 아닌 언어로 된 주소 요소가 일부 있을 수 있습니다(다른 주소 요소는 기본 언어 사용). 변환이 속성에서 지정하는 언어로 된 요소를 찾지 못하는 경우 변환은 해당 요소를 기본 언어로 반환합니다.

기본 설정 언어 옵션을 설정할 때 기본 설정 스크립트 속성이 지정하는 문자 집합이 예상되는 출력 주소 데이터와 호환되는지 확인합니다.

### 벨기에 주소에 대한 다국어 지원

다음 테이블에는 벨기에 주소에 지정할 수 있는 옵션과 언어가 설명되어 있습니다.

옵션	설명
데이터베이스	기본값입니다. 주소가 속한 지역의 주 언어로 주소를 반환합니다. 언어는 플라망어, 프랑스어 또는 독일어일 수 있습니다.
영어	주소 참조 데이터에 영어로 된 정보가 포함되는 경우 시/도, 로컬리티 및 거리 정보를 영어로 반환합니다. 주소가 속하는 지역의 다른 주소 요소는 주 언어로 반환됩니다.
대체 1	시/도, 로컬리티 및 거리 정보를 플라망어로 반환합니다.
대체 2	시/도, 로컬리티 및 거리 정보를 프랑스어로 반환합니다.
대체 3	시/도, 로컬리티 및 거리 정보를 독일어로 반환합니다.
입력 유지	주소 정보를 입력 언어로 반환합니다. 주소 참조 데이터에 입력 언어의 주소 정보가 포함되어 있는 경우 주소 유효성 검사에서 해당 언어가 유지됩니다. 주소 유효성 검사가 입력 주소에서 둘 이상의 지원되는 언어를 감지하는 경우 검사는 데이터베이스 언어로 주소를 반환합니다. 주소 유효성 검사에서 입력 언어로 요소를 반환할 수 없는 경우 데이터베이스 언어로 반환합니다.

### 캐나다 주소에 대한 다국어 지원

다음 테이블에는 캐나다 주소에 지정할 수 있는 옵션과 언어가 설명되어 있습니다.

옵션	설명
데이터베이스	기본값입니다. 퀘벡을 제외한 모든 지방에 대해 영어 주소를 반환합니다. 퀘벡 주소를 프랑스어로 반환합니다.
영어	모든 주소를 영어로 반환합니다.

옵션	설명
대체 1	모든 주소를 영어로 반환합니다.
대체 2	퀘벡 주소를 프랑스어로 반환합니다. 퀘벡이 아닌 지방의 경우 변환은 거리 설명자, 방향 정보 및 시/도 이름을 프랑스어로 반환하고 다른 주소 요소를 영어로 반환합니다.
입력 유지	주소 정보를 입력 언어로 반환합니다. 주소 참조 데이터에 입력 언어의 주소 정보가 포함되어 있는 경우 주소 유효성 검사에서 해당 언어가 유지됩니다. 주소 유효성 검사가 입력 주소에서 둘 이상의 지원되는 언어를 감지하는 경우 검사는 데이터베이스 언어로 주소를 반환합니다. 주소 유효성 검사에서 입력 언어로 요소를 반환할 수 없는 경우 데이터베이스 언어로 반환합니다.

## 중국 주소에 대한 다국어 지원

다음 테이블에는 중국 주소에 지정할 수 있는 옵션과 언어가 설명되어 있습니다.

옵션	설명
데이터베이스	기본값입니다. 모든 주소 정보를 중국어로 반환합니다.
영어	영어 버전의 거리 설명자 및 거리 방향 값을 반환합니다. 다른 모든 주소 정보는 중국어로 반환합니다. 영어 주소 요소는 "shi"와 같은 트랜스리터레이션 요소를 생략합니다.
대체 1	모든 주소 정보를 데이터베이스 언어로 반환합니다.
대체 2	모든 주소 정보를 데이터베이스 언어로 반환합니다.
대체 3	모든 주소 정보를 데이터베이스 언어로 반환합니다.
입력 유지	주소 정보를 입력 언어로 반환합니다. 주소 참조 데이터에 입력 언어의 주소 정보가 포함되어 있는 경우 주소 유효성 검사에서 해당 언어가 유지됩니다. 주소 유효성 검사가 입력 주소에서 둘 이상의 지원되는 언어를 감지하는 경우 검사는 데이터베이스 언어로 주소를 반환합니다. 주소 유효성 검사에서 입력 언어로 요소를 반환할 수 없는 경우 데이터베이스 언어로 반환합니다.

기본 설정 언어를 선택할 때는 다음 규칙 및 지침을 고려하십시오.

- 주소를 중국어로 반환하려면 데이터베이스, 대체 1, 대체 2 또는 대체 3을 선택합니다.  
주소를 중국어 문자 집합으로 반환하려면 기본 설정 스크립트 속성을 데이터베이스로 설정합니다.
- 거리 설명자 및 거리 방향 정보를 영어로 반환하려면 영어를 선택합니다.  
주소를 라틴 또는 ASCII 문자 집합으로 반환하려면 기본 설정 스크립트 속성을 LATIN 또는 ASCII 값으로 설정합니다.
- LATIN 또는 ASCII 값을 기본 설정 스크립트로 선택하고 데이터베이스를 기본 설정 언어로 선택하면 주소 유효성 검사에서 주소 데이터가 병음으로 반환됩니다.

## 핀란드 주소에 대한 다국어 지원

다음 테이블에는 핀란드 주소에 지정할 수 있는 옵션과 언어가 설명되어 있습니다.

옵션	설명
데이터베이스	기본값입니다. 모든 주소 정보를 핀란드어로 반환합니다.
대체 1	모든 주소 정보를 데이터베이스 언어로 반환합니다.
대체 2	거리, 로컬리티 및 시/도 정보를 스웨덴어로 반환합니다. 다른 모든 주소 정보는 핀란드어로 반환합니다.
입력 유지	주소 정보를 입력 언어로 반환합니다. 주소 참조 데이터에 입력 언어의 주소 정보가 포함되어 있는 경우 주소 유효성 검사에서 해당 언어가 유지됩니다. 주소 유효성 검사가 입력 주소에서 둘 이상의 지원되는 언어를 감지하는 경우 검사는 데이터베이스 언어로 주소를 반환합니다. 주소 유효성 검사에서 입력 언어로 요소를 반환할 수 없는 경우 데이터베이스 언어로 반환합니다.

## 이스라엘 주소에 대한 다국어 지원

다음 테이블에는 이스라엘 주소에 지정할 수 있는 옵션과 언어가 설명되어 있습니다.

옵션	설명
데이터베이스	기본값입니다. 모든 주소 정보를 히브리어로 반환합니다.
영어	모든 주소 정보를 영어로 반환합니다.
대체 1	모든 주소 정보를 히브리어로 반환합니다.
대체 2	모든 주소 정보를 영어로 반환합니다.
입력 유지	주소 정보를 입력 언어로 반환합니다. 주소 참조 데이터에 입력 언어의 주소 정보가 포함되어 있는 경우 주소 유효성 검사에서 해당 언어가 유지됩니다. 주소 유효성 검사가 입력 주소에서 둘 이상의 지원되는 언어를 감지하는 경우 검사는 데이터베이스 언어로 주소를 반환합니다. 주소 유효성 검사에서 입력 언어로 요소를 반환할 수 없는 경우 데이터베이스 언어로 반환합니다.

기본 설정 언어를 선택할 때는 다음 규칙 및 지침을 고려하십시오.

- 주소를 히브리어 문자 집합으로 반환하려면 기본 설정 스크립트 속성을 데이터베이스로 설정합니다.
- 주소를 라틴 또는 ASCII 문자 집합으로 반환하려면 기본 설정 스크립트 속성을 LATIN 또는 ASCII 값으로 설정합니다.
- 라틴 문자 집합을 기본 설정 스크립트로 선택하고 히브리어를 기본 설정 언어로 선택하는 경우 주소 유효성 검사는 히브리어 주소를 라틴 문자로 변환합니다. 라틴 문자 집합의 결과를 최적화하려면 영어를 기본 설정 언어로 선택합니다.

## 홍콩 주소에 대한 다국어 지원

다음 테이블에는 홍콩 주소에 지정할 수 있는 옵션과 언어가 설명되어 있습니다.

옵션	설명
데이터베이스	기본값입니다. 모든 주소 정보를 중국어로 반환합니다.
영어	모든 주소 정보를 영어로 반환합니다.
대체 1	모든 주소 정보를 데이터베이스 언어로 반환합니다.
대체 2	모든 주소 정보를 영어로 반환합니다.
대체 3	모든 주소 정보를 데이터베이스 언어로 반환합니다.
입력 유지	주소 정보를 입력 언어로 반환합니다. 주소 참조 데이터에 입력 언어의 주소 정보가 포함되어 있는 경우 주소 유효성 검사에서 해당 언어가 유지됩니다. 주소 유효성 검사가 입력 주소에서 둘 이상의 지원되는 언어를 감지하는 경우 검사는 데이터베이스 언어로 주소를 반환합니다. 주소 유효성 검사에서 입력 언어로 요소를 반환할 수 없는 경우 데이터베이스 언어로 반환합니다.

홍콩에 대한 기본 설정 언어를 선택할 때는 다음 규칙 및 지침을 고려하십시오.

- 주소를 중국어 문자 집합으로 반환하려면 기본 설정 스크립트 속성을 데이터베이스로 설정합니다.
- 주소를 라틴 또는 ASCII 문자 집합으로 반환하려면 기본 설정 스크립트 속성을 LATIN 또는 ASCII 값으로 설정합니다.
- 입력 데이터의 언어는 홍콩 주소에 대한 입력 유지 옵션의 작동에 영향을 줄 수 있습니다. 입력 데이터에 7비트 ASCII 문자가 사용되고 영어 설명자가 포함되는 경우 주소 유효성 검사에서 입력 언어를 영어로 식별합니다.

## 아일랜드 주소에 대한 다국어 지원

다음 테이블에는 아일랜드 주소에 지정할 수 있는 옵션과 언어가 설명되어 있습니다.

옵션	설명
데이터베이스	기본값입니다. 모든 주소 정보를 영어로 반환합니다.
영어	모든 주소 정보를 영어로 반환합니다.
대체 1	모든 주소 정보를 영어로 반환합니다.
대체 2	거리, 로컬리티 및 카운티 정보를 아일랜드어로 반환합니다. 다른 모든 주소 정보는 영어로 반환합니다.
입력 유지	주소 정보를 입력 언어로 반환합니다. 주소 참조 데이터에 입력 언어의 주소 정보가 포함되어 있는 경우 주소 유효성 검사에서 해당 언어가 유지됩니다. 주소 유효성 검사가 입력 주소에서 둘 이상의 지원되는 언어를 감지하는 경우 검사는 데이터베이스 언어로 주소를 반환합니다. 주소 유효성 검사에서 입력 언어로 요소를 반환할 수 없는 경우 데이터베이스 언어로 반환합니다.



## 마카오 주소에 대한 다국어 지원

다음 테이블에는 마카오 주소에 지정할 수 있는 옵션과 언어가 설명되어 있습니다.

옵션	설명
데이터베이스	기본값입니다. 모든 주소 정보를 중국어로 반환합니다.
대체 1	모든 주소 정보를 데이터베이스 언어로 반환합니다.
대체 2	모든 주소 정보를 포르투갈어로 반환합니다.
입력 유지	주소 정보를 입력 언어로 반환합니다. 주소 참조 데이터에 입력 언어의 주소 정보가 포함되어 있는 경우 주소 유효성 검사에서 해당 언어가 유지됩니다. 주소 유효성 검사가 입력 주소에서 둘 이상의 지원되는 언어를 감지하는 경우 검사는 데이터베이스 언어로 주소를 반환합니다. 주소 유효성 검사에서 입력 언어로 요소를 반환할 수 없는 경우 데이터베이스 언어로 반환합니다.

- 주소를 중국어 문자 집합으로 반환하려면 기본 설정 스크립트 속성을 데이터베이스로 설정합니다.
- 주소를 라틴 또는 ASCII 문자 집합으로 반환하려면 기본 설정 스크립트 속성을 LATIN 또는 ASCII 값으로 설정합니다.
- 입력 데이터의 언어는 마카오 주소에 대한 입력 유지 옵션의 작동에 영향을 줄 수 있습니다. 입력 데이터에 7 비트 ASCII 문자가 사용되고 포르투갈어 설명자가 포함되는 경우 주소 유효성 검사에서 입력 언어를 포르투갈어로 식별합니다.

## 스위스에 대한 다국어 지원

다음 테이블에는 스위스 주소에 지정할 수 있는 옵션과 언어가 설명되어 있습니다.

옵션	설명
데이터베이스	기본값입니다. 주소가 속하는 지역의 주 언어로 주소를 반환합니다. 예를 들어 주소 유효성 검사에서 취리히 주소는 독일어로 반환되고 제네바 주소는 프랑스어로 반환됩니다.
영어	참조 주소 데이터베이스에 영어로 된 정보가 포함되는 경우 로컬리티 및 시/도 정보를 영어로 반환합니다. 주소가 속하는 지역의 다른 주소 요소는 주 언어로 반환됩니다. 주소 유효성 검사에서 제네바 및 취리히 등 일부 로컬리티에 대한 로컬리티 정보는 영어로 반환됩니다.
대체 1	시/도 및 로컬리티 정보를 독일어로 반환합니다.
대체 2	시/도 및 로컬리티 정보를 프랑스어로 반환합니다.
대체 3	시/도 및 로컬리티 정보를 이탈리아어로 반환합니다.
입력 유지	주소 정보를 입력 언어로 반환합니다. 주소 참조 데이터에 입력 언어의 주소 정보가 포함되어 있는 경우 주소 유효성 검사에서 해당 언어가 유지됩니다. 주소 유효성 검사가 입력 주소에서 둘 이상의 지원되는 언어를 감지하는 경우 검사는 데이터베이스 언어로 주소를 반환합니다. 주소 유효성 검사에서 입력 언어로 요소를 반환할 수 없는 경우 데이터베이스 언어로 반환합니다.

**참고:** 또한 주소 유효성 검사에서 빌/비엔 주소에 대한 거리 정보가 구성된 대체 언어로 반환됩니다.

## 대만에 대한 다국어 지원

다음 테이블에는 대만 주소에 지정할 수 있는 옵션과 언어가 설명되어 있습니다.

옵션	설명
데이터베이스	기본값입니다. 모든 주소 정보를 중국어로 반환합니다.
영어	모든 주소 정보를 영어로 반환합니다.
입력 유지	주소 정보를 입력 언어로 반환합니다. 주소 참조 데이터에 입력 언어의 주소 정보가 포함되어 있는 경우 주소 유효성 검사에서 해당 언어가 유지됩니다. 주소 유효성 검사가 입력 주소에서 둘 이상의 지원되는 언어를 감지하는 경우 검사는 데이터베이스 언어로 주소를 반환합니다. 주소 유효성 검사에서 입력 언어로 요소를 반환할 수 없는 경우 데이터베이스 언어로 반환합니다.

기본 설정 언어를 선택할 때는 다음 규칙 및 지침을 고려하십시오.

- 주소를 중국어 문자 집합으로 반환하려면 기본 설정 스크립트 매개 변수를 데이터베이스로 설정합니다.
- 주소를 라틴 또는 ASCII 문자 집합으로 반환하려면 기본 설정 스크립트 매개 변수를 LATIN 또는 ASCII 값으로 설정합니다.
- 입력 데이터의 언어는 대만 주소에 대한 입력 유지 옵션의 작동에 영향을 줄 수 있습니다. 입력 데이터에 7비트 ASCII 문자가 사용되고 영어 설명자가 포함되는 경우 주소 유효성 검사에서 입력 언어를 영어로 식별합니다.

## 기본 설정 스크립트

주소 유효성 검사기 변환이 출력 데이터에 대해 사용하는 문자 집합을 결정합니다.

다음 테이블에는 속성에 대한 옵션이 설명되어 있습니다.

옵션	설명
ASCII(단순)	주소를 ASCII 문자로 반환합니다.
ASCII(확장)	주소를 ASCII 문자로 반환하고 주소의 특수 문자를 확장합니다. 예를 들어 Ö는 OE로 트랜스리터레이션합니다.
데이터베이스	주소 참조 데이터의 기본 언어로 사용되는 문자 집합으로 주소를 반환합니다. 데이터베이스가 기본값입니다.
라틴어	주소를 라틴어 문자 집합으로 반환합니다.
라틴어(Alt)	주소를 대체 라틴어 문자 집합으로 반환합니다. 예를 들어 한국 주소를 수정된 로마자화 트랜스리터레이션으로 반환하려면 라틴어를 지정합니다. 한국 주소를 이전의 ISO/TR 11941 트랜스리터레이션으로 반환하려면 라틴어(Alt)를 지정합니다.
우편 관리	주소를 해당 주소의 지역 우체국에 기본 설정된 스크립트로 반환합니다.
우편 관리(Alt)	주소를 해당 주소의 지역 우체국에서 승인한 대체 스크립트로 반환합니다.
입력 유지	주소 데이터를 입력 주소에서 사용하는 문자 집합으로 반환합니다.

변환은 여러 개의 언어 및 문자 집합으로 데이터를 포함하는 데이터 소스를 처리할 수 있습니다. 변환은 모든 입력 데이터를 **Unicode UCS-2** 문자 집합으로 변환하고 **UCS-2** 형식으로 데이터를 처리합니다. 변환은 데이터를 처리한 후 각 주소 레코드의 데이터를 속성에서 지정하는 문자 집합으로 변환합니다. 이 프로세스를 **트랜스리터레이션**이라고 합니다.

트랜스리터레이션에서 처리를 위해 문자를 변환하면 문자 집합에 있는 각 문자에 대한 숫자 표현을 사용할 수 있습니다. 또한 트랜스리터레이션은 문자에 대한 동일한 숫자 표현이 없는 경우 문자를 발음되는 대로 변환할 수도 있습니다. 주소 유효성 검사기 변환은 문자를 **UCS-2**로 매핑할 수 없는 경우 해당 문자를 공백으로 변환합니다.

**참고:** 변환의 기본 설정 언어 또는 기본 설정 스크립트를 업데이트하는 경우 선택한 언어와 문자 코드가 호환되는지 확인하십시오.

## 확장할 범위

주소 유효성 검사기 변환이 집 번호를 지정하지 않는 거리 주소에 대해 제안된 주소를 반환하는 방식을 결정합니다. 변환이 제안 목록 모드로 실행될 때 이 속성을 사용합니다.

주소 유효성 검사기 변환은 부분적이거나 불완전한 거리 주소를 제안 목록 모드로 읽습니다. 변환은 해당 주소를 주소 참조 데이터와 비교하고 최종 사용자에게 유사한 모든 주소를 반환합니다. 입력 주소에 집 번호가 포함되지 않는 경우 변환은 해당 거리에 대해 하나 이상의 집 번호 제안을 반환할 수 있습니다. 확장할 범위 속성은 변환이 주소를 반환하는 방식을 결정합니다.

변환은 단일 주소로 유효한 집 번호 범위를 반환할 수 있거나 유효한 각 집 번호에 대해 개별 주소를 반환할 수 있습니다. 또한 해당 거리에서 가장 낮은 집 번호부터 가장 높은 집 번호까지의 범위에서 각 번호에 대한 주소를 반환할 수도 있습니다.

다음 테이블에는 속성에 대한 옵션이 설명되어 있습니다.

옵션	설명
모두	주소 유효성 검사는 해당 거리에서 가능한 집 번호 범위에서 각 집 번호에 대해 제안된 주소를 반환합니다.
없음	주소 유효성 검사는 해당 거리에 대해 유효한 범위에서 가장 낮은 집 번호와 가장 높은 집 번호를 식별하는 단일 주소를 반환합니다.
올바른 항목만 포함	주소 유효성 검사는 주소 참조 데이터가 배달 가능한 주소로 인식하는 각 집 번호에 대해 제안된 주소를 반환합니다.

**참고:** 제안 목록 모드는 주소의 다른 요소를 사용하여 거리 번호에 대한 유효한 범위를 지정할 수 있습니다. 예를 들어 우편 번호가 주소 사서함을 포함하는 도시 블록을 식별할 수 있습니다. 주소 유효성 검사기 변환은 우편 번호를 사용하여 블록에서 가장 낮은 유효한 집 번호와 가장 높은 유효한 집 번호를 식별할 수 있습니다.

변환이 실질적인 한도 내에서 집 번호 범위를 결정할 수 없는 경우 제안된 주소 수가 사용할 수 없는 크기로 증대될 수 있습니다. 확장할 범위 속성이 생성하는 주소 수를 제한하려면 유동적 범위 확장 속성을 **On**으로 설정해야 합니다.

## 잘못된 주소 표준화

주소 유효성 검사 프로세스가 배달할 수 없는 주소의 데이터 값을 표준화하는지 여부를 결정합니다. 이 속성은 **11**에서 **14**까지의 범위에서 일치 코드 상태를 반환하는 주소 레코드에 적용됩니다.

데이터를 표준화하면 다운스트림 데이터 프로세스가 정확한 결과를 반환할 가능성이 높아집니다. 예를 들어, 중복 분석 매핑은 동일한 형식의 일반 주소 요소를 나타내는 두 개의 주소 레코드에 대해 높은 점수 일치를 반환할 수 있습니다.

주소 유효성 검사는 다음 주소 요소를 표준화할 수 있습니다.

- 도로 및 대로와 같은 거리 접미사 요소.
- 동, 서, 남, 북과 같은 방향을 가리키는 요소.
- 사서함과 같은 배달 서비스 요소.
- 아파트, 층, 스위트룸과 같은 하위 건물 요소.
- 시/도 이름. 표준화는 축약된 이름 형식을 반환합니다.

다음 테이블에는 속성에 대한 옵션이 설명되어 있습니다.

옵션	설명
꺼짐	주소 유효성 검사는 데이터 오류를 수정하지 않습니다. 기본값 옵션입니다.
켜짐	주소 유효성 검사는 데이터 오류를 수정합니다.

### 매개 변수 사용

데이터 오류에 대해 표준화 정책을 지정하기 위해 매개 변수를 할당할 수 있습니다. 매개 변수 값으로 OFF 또는 ON을 입력합니다. 값을 대문자로 입력합니다.

## 추적 수준

로그에 포함된 세부 정보의 양을 설정합니다.

로그의 추적 수준을 구성할 수 있습니다.

고급 탭에서 다음 속성을 구성합니다.

### 추적 수준

이 변환에 대해 로그에 표시되는 세부 정보의 양입니다. 간단, 보통, 자세한 정보 표시 초기화 또는 자세한 정보 표시 데이터를 선택할 수 있습니다. 기본값은 보통입니다.

## 인증 보고서

인증 모드에서 유효성 검사를 위해 제출하는 주소의 상태를 설명하는 보고서를 생성할 수 있습니다. 이 보고서는 주소 집합이 우편 운송업체에서 정의한 인증 표준을 충족하는지 확인합니다.

주소 유효성 검사기 변환은 다음 표준에 대한 보고서를 생성합니다.

### AMAS(주소 일치 승인 시스템)

오스트레일리아 우체국은 AMAS 인증 표준을 정의합니다.

### CASS(코딩 정확도 지원 시스템)

USPS는 CASS 인증 표준을 정의합니다.

### SendRight

뉴질랜드 우체국은 SendRight 인증 표준을 정의합니다.

### SERP(소프트웨어 평가 및 인식 프로그램)

캐나다 우체국은 SERP 인증 표준을 정의합니다.

**참고:** 주소 유효성 검사기 변환을 사용하여 프랑스 주소 레코드가 SNA(국가 주소 관리 서비스) 인증 표준에 맞는지 인증할 수 있습니다. 그러나 주소 유효성 검사기 변환의 SNA 주소 인증에 대한 보고서는 생성하지 않습니다.

우편 운송업체에 인증 표준을 충족하는 우편 항목을 제공할 때는 주소 집합이 포함된 인증 보고서를 제출합니다. 이 보고서에는 조직에 대한 데이터가 포함됩니다. 이 데이터는 주소 유효성 검사기 변환을 구성할 때 입력할 수 있습니다. 그러면 변환이 보고서 파일에 조직 데이터를 추가합니다.

## AMAS 보고서 필드

AMAS 보고서를 구성할 경우 인증된 주소 레코드 집합을 오스트레일리아 우체국에 제출하는 조직에 대한 정보를 제공하십시오. 보고서를 저장 또는 인쇄하고, 오스트레일리아 우체국에 제출한 주소 레코드를 보고서에 포함하십시오.

**보고서** 보기를 사용하여 정보를 입력합니다.

다음 표에는 입력하는 정보가 설명되어 있습니다.

필드	설명
보고서 파일 이름	주소 유효성 검사에서 작성되는 보고서의 이름 및 위치입니다. 기본적으로 주소 유효성 검사기는 데이터 통합 서비스 시스템의 bin 디렉터리에 보고서를 작성합니다. 보고서 파일을 데이터 통합 서비스 시스템의 다른 위치에 기록하려면 파일 경로 및 파일 이름을 입력하십시오. 정규화된 경로 또는 상대 경로를 입력할 수 있습니다. 상대 경로는 bin 디렉터리를 루트 디렉터리로 사용합니다. 주소 유효성 검사기 매핑을 실행하기 전에 지정한 디렉터리가 존재하는지 확인하십시오.
주소 목록 이름	오스트레일리아 우체국에 제출하는 주소 레코드 집합의 이름입니다.
목록 프로세서 이름	주소 레코드 집합을 제출하는 조직의 이름입니다.
목록 관리자/소유자 이름	조직의 주소 데이터 관리자 또는 소유자 이름입니다.
전화 번호	주소 레코드 집합을 제출하는 조직의 연락처 전화 번호입니다.
주소	주소 레코드 집합을 제출하는 조직의 주소입니다.

관련 항목:

- [“인증 보고서 정의” 페이지 125](#)

## CASS 보고서 필드

CASS 보고서를 구성할 때는 인증된 주소 레코드 집합을 USPS에 제출하는 조직에 대한 정보를 입력해야 합니다. 이 보고서를 저장 또는 인쇄하여 USPS에 제출하는 주소 레코드에 포함하십시오.

**보고서** 보기를 사용하여 정보를 입력합니다.

다음 표에는 입력하는 정보가 설명되어 있습니다.

필드	설명
보고서 파일 이름	주소 유효성 검사에서 작성되는 보고서의 이름 및 위치입니다. 기본적으로 주소 유효성 검사는 데이터 통합 서비스 시스템의 bin 디렉터리에 보고서를 작성합니다. 보고서 파일을 데이터 통합 서비스 시스템의 다른 위치에 기록하려면 파일 경로 및 파일 이름을 입력하십시오. 정규화된 경로 또는 상대 경로를 입력할 수 있습니다. 상대 경로는 bin 디렉터리를 루트 디렉터리로 사용합니다. 주소 유효성 검사기 매핑을 실행하기 전에 지정한 디렉터리가 존재하는지 확인하십시오.
목록 이름/ID	우편 운송업체에 제출하는 주소 목록의 이름 또는 ID 번호입니다.
목록 프로세서 이름	주소 유효성 검사를 수행하는 조직의 이름입니다.
이름/주소	주소 유효성 검사를 수행하는 조직의 우편 이름 및 주소입니다.

**참고:** 주소 유효성 검사기 변환은 미국에 대한 일괄 및 대화형 참조 데이터 파일에서 CASS 참조 데이터 파일에 대한 메타데이터를 읽습니다. CASS 보고서를 생성하려면 변환이 현재 CASS 참조 데이터 파일과 현재 일괄 및 대화형 참조 데이터 파일을 읽어야 합니다.

#### 관련 항목:

- [“인증 보고서 정의” 페이지 125](#)

## SendRight 보고서

SendRight 보고서를 구성하는 경우 인증된 주소 레코드 집합을 뉴질랜드 우체국에 제출하는 조직에 대한 정보를 제공합니다. 보고서를 저장하거나 인쇄하고 뉴질랜드 우체국에 제출하는 주소 레코드를 보고서에 포함시킵니다.

**보고서** 보기를 사용하여 정보를 입력합니다.

다음 표에는 입력하는 정보가 설명되어 있습니다.

필드	설명
고객 이름	주소 레코드 집합을 제출하는 조직의 이름입니다.
고객 NZP 번호	주소 레코드 집합을 제출하는 조직의 뉴질랜드 우체국 계정 번호입니다. 메일하우스가 조직을 대신해 레코드를 제출하는 경우 메일하우스 전송 ID(TPID) 번호를 입력합니다.
고객 데이터베이스	주소 레코드 집합이 포함된 파일의 이름입니다. 주소 유효성 검사기 변환이 콘텐츠 관리 서비스에서 지정하는 위치에 보고서를 생성합니다. Administrator 도구를 사용하여 위치를 설정합니다.
고객 주소	주소 레코드 집합을 제출하는 조직의 주소입니다.

## 관련 항목:

- [“인증 보고서 정의” 페이지 125](#)

## SERP 보고서 필드

SERP 보고서를 구성할 때는 인증된 주소 레코드 집합을 캐나다 우체국에 전송하는 조직에 대한 정보를 제공합니다. 보고서를 저장 또는 인쇄하고 캐나다 우체국에 제출한 주소 레코드를 보고서에 포함하십시오.

**보고서** 보기를 사용하여 정보를 입력합니다.

다음 테이블에는 입력하는 정보가 설명되어 있습니다.

필드	설명
보고서 파일 이름	주소 유효성 검사에서 작성되는 보고서의 이름 및 위치입니다. 기본적으로 주소 유효성 검사기는 데이터 통합 서비스 시스템의 bin 디렉터리에 보고서를 작성합니다. 보고서 파일을 데이터 통합 서비스 시스템의 다른 위치에 기록하려면 파일 경로 및 파일 이름을 입력하십시오. 정규화된 경로 또는 상대 경로를 입력할 수 있습니다. 상대 경로는 bin 디렉터리를 루트 디렉터리로 사용합니다. 주소 유효성 검사기 매핑을 실행하기 전에 지정한 디렉터리가 존재하는지 확인하십시오.
고객 CPC 번호	캐나다 우체국에서 주소 유효성 검사를 수행하는 조직에 대해 발급하는 고객 번호입니다.
고객 이름/주소	주소 유효성 검사를 수행하는 조직의 이름과 주소입니다.

## 관련 항목:

- [“인증 보고서 정의” 페이지 125](#)

## 주소 유효성 검사기 변환 구성

우편 주소 데이터 품질을 개선하고 유효성을 검사하려면 주소 유효성 검사기 변환을 사용하십시오.

주소 유효성 검사기 변환에서 주소 참조 데이터를 읽습니다. **Developer tool**이 필요한 주소 참조 데이터 파일에 액세스할 수 있는지 확인합니다.

1. 변환을 엽니다.
2. **일반 설정** 보기를 클릭하고 일반 속성을 구성합니다.
3. **템플릿** 보기를 클릭하여 입력 및 출력 포트를 추가합니다.
4. **보고서** 보기를 클릭하여 우편 서비스 주소 인증에 대한 보고서를 생성합니다.
5. **고급** 보기를 클릭하여 고급 주소 유효성 검사 속성을 구성합니다.
6. 입력 및 출력 포트를 연결합니다.

**참고:** 주소 변환에서 유효성을 검사하지 않도록 할 입력 포트를 **통과** 입력 포트 그룹에 연결합니다.

## 주소 유효성 검사기 변환에 포트 추가

주소 유효성 검사기 변환에 포트를 추가하려면 **템플릿** 보기를 사용하십시오.

1. **템플릿** 보기를 클릭합니다.
2. 템플릿을 확장합니다.
  - 일반 주소 필드를 추가하려면 **기본 모델** 템플릿을 선택합니다.
  - 특수한 주소 필드를 추가하려면 **고급 모델** 템플릿을 선택합니다.
3. 입력 데이터의 형식에 해당하는 입력 포트 그룹을 확장합니다. 입력 포트 그룹은 **불연속**, **다중 선** 및 **하이브리드**입니다.
4. 입력 포트를 선택합니다.

**팁:** CTRL 키를 클릭하여 여러 포트를 선택합니다.
5. 포트를 마우스 오른쪽 단추로 클릭하고 **변환에 포트 추가**를 선택합니다.
6. 필요한 필드가 포함된 출력 포트 그룹을 확장합니다.
7. 포트를 마우스 오른쪽 단추로 클릭하고 **변환에 포트 추가**를 선택합니다.
8. 유효성을 검사하지 않을 열에 대해 통과 포트를 추가하려면 **포트** 탭을 클릭하고 **통과** 입력 포트 그룹을 선택한 다음 **새로 만들기**를 클릭합니다.

## 사용자 정의 템플릿 작성

템플릿을 작성하여 재사용하려는 주소 포트를 그룹화합니다.

사용자 지정 템플릿을 작성하려면 기본 및 고급 템플릿에서 포트를 선택합니다. 후속 주소 유효성 검사기 변환을 작성할 때 사용자 지정 템플릿을 선택할 수 있습니다.

**참고:** 템플릿은 리포지토리 개체가 아닙니다. 템플릿은 템플릿을 작성할 때 사용한 시스템에 상주합니다.

1. **템플릿** 보기를 선택합니다.
2. **새로 만들기**를 클릭합니다.
3. 템플릿 이름을 입력합니다.
4. **기본 모델** 또는 **고급 모델** 템플릿을 확장하고 필요한 포트를 선택합니다.
5. **확인**을 클릭합니다.

## 주소 유효성 검사기 모델 정의

주소 유효성 검사기 모델은 주소 유효성 검사기 변환의 기본 입력 및 출력 포트를 정의합니다.

주소 유효성 검사기 변환에는 기본 입력 및 출력 포트가 포함되지 않습니다. 그러나 주소 유효성 검사기 변환에서 사용하는 입력 및 출력 포트를 지정하도록 모델을 정의할 수 있습니다.

**참고:** 모델은 리포지토리 개체가 아닙니다. 모델을 작성하기 위해 사용한 시스템에 모델이 있습니다.

주소 유효성 검사기 모델을 정의하려면 다음 단계를 수행하십시오.

1. **템플릿** 보기를 선택합니다.



2. 기본 모델 또는 고급 모델 템플릿을 확장하고 필요한 포트를 선택합니다.
3. 선택한 포트를 사용하여 기본 AV 모델 작성을 선택합니다.
4. 모델을 재설정하고 모든 포트를 제거하려면 기본 AV 모델 지우기를 선택합니다.

## 인증 보고서 정의

주소 유효성 검사기 변환에서 인증 보고서를 정의할 때 일반 설정 및 보고서 보기에서 옵션을 구성해야 합니다.

1. 일반 설정 보기에서 모드 옵션을 인증됨으로 설정합니다.
2. 보고서 보기에서 생성할 보고서 유형을 선택합니다. 다음 보고서 유형을 선택할 수 있습니다.

옵션	설명
AMAS 보고서	오스트레일리아 우체국에서 레코드 집합을 처리할 때 필요로 하는 정보가 포함됩니다.
CASS 보고서	USPS에서 레코드 집합을 처리할 때 필요로 하는 정보가 포함됩니다.
SendRight 보고서	뉴질랜드 우체국에서 레코드 집합을 처리할 때 필요로 하는 정보가 포함됩니다.
SERP 보고서	캐나다 우체국에서 레코드 집합을 처리할 때 필요로 하는 정보가 포함됩니다.

3. 선택한 각 보고서 유형에 대한 보고서 세부 정보를 입력합니다.

주소 유효성 검사기 변환을 사용하여 유효성을 검사한 주소 레코드 목록을 사용하여 보고서 파일을 우체국에 제출합니다.

### 관련 항목:

- [“AMAS 보고서 필드” 페이지 121](#)
- [“CASS 보고서 필드” 페이지 121](#)
- [“SendRight 보고서” 페이지 122](#)
- [“SERP 보고서 필드” 페이지 123](#)

## 주소 유효성 검사기 변환 비원시 환경 내

비원시 환경에서 주소 유효성 검사기 변환의 처리는 변환을 실행하는 엔진에 따라 다릅니다.

다음과 같은 비원시 런타임 엔진에 대한 지원을 고려하십시오.

- Blaze 엔진. 제한적으로 지원됩니다.
- Spark 엔진. 제한적으로 지원됩니다.
- Databricks Spark 엔진. 지원되지 않습니다.

## 주소 유효성 검사기 변환 Blaze 엔진 내

주소 유효성 검사기 변환은 다음과 같은 제한과 함께 지원됩니다.

- 주소 유효성 검사기 변환은 인증 보고서를 생성할 수 없습니다.
- 주소 유효성 검사기 변환이 대화형 모드 또는 제안 목록 모드에서 실행되도록 구성된 경우 유효성 검사에 실패합니다.

## 주소 유효성 검사기 변환 Spark 엔진 내

주소 유효성 검사기 변환은 다음과 같은 제한과 함께 지원됩니다.

- 주소 유효성 검사기 변환은 인증 보고서를 생성할 수 없습니다.
- 주소 유효성 검사기 변환이 대화형 모드 또는 제안 목록 모드에서 실행되도록 구성된 경우 유효성 검사에 실패합니다.

## 제 5 장

# 집계 변환

이 장에 포함된 항목:

- [집계 변환 개요, 127](#)
- [동적 매핑의 집계 변환, 128](#)
- [집계 변환 개발, 128](#)
- [집계 변환 포트, 128](#)
- [집계 식, 129](#)
- [그룹 기준 포트, 131](#)
- [집계 캐시, 133](#)
- [집계 변환을 위한 정렬된 입력, 134](#)
- [집계 변환 고급 속성, 135](#)
- [재사용 가능한 집계 변환 작성, 136](#)
- [재사용 불가능 집계 변환 작성, 136](#)
- [집계 변환에 대한 팁, 137](#)
- [집계 변환 문제 해결, 137](#)
- [집계 변환 - 비원시 환경, 138](#)

## 집계 변환 개요

집계 변환을 구성하여 데이터 그룹에 대해 평균 및 합계와 같은 집계 계산을 수행합니다. 집계 변환을 사용하여 중복 행을 제거할 수 있습니다. 집계 변환은 활성 변환입니다.

집계 변환을 구성하여 데이터 그룹에 대한 계산을 수행할 수 있으므로 집계 변환은 식 변환과 다릅니다. 식 변환은 행 기준으로 행에 대한 결과를 반환합니다.

예를 들어 조직 각 부서의 직원에 대한 평균 급여를 계산할 수 있습니다. 부서 번호로 그룹을 구성합니다. 평균 급여를 계산하고 각 고유 부서 번호에 대한 결과를 반환하도록 식을 구성합니다.

변환 언어를 사용하여 집계 식을 작성합니다.

데이터 통합 서비스는 데이터를 읽고 집계 캐시에 저장하여 집계 계산을 수행합니다. 입력 데이터를 정렬하여 성능을 향상시킬 수 있습니다. 입력 데이터를 정렬하는 경우 데이터 통합 서비스가 캐시를 작성하지 않습니다.

## 동적 매핑의 집계 변환

동적 매핑에서 집계 변환을 사용할 수 있습니다. 변환에서 동적 포트를 구성하고 생성된 포트를 참조할 수 있습니다.

다음 태스크를 수행하여 동적 매핑에서 집계 변환을 구성할 수 있습니다.

### 동적 포트 또는 생성된 포트를 그룹 기준 열로 참조

동적 포트 또는 생성된 포트를 그룹 기준 열로 포함할 수 있습니다. 동적 포트를 그룹 기준 열로 지정하는 경우 동적 포트에서 생성된 모든 포트가 그룹 기준 열로 지정됩니다. 이러한 이유로, 상위 동적 포트를 그룹 기준 열로 지정하는 경우 생성된 포트를 그룹 기준 열로 지정할 수 없습니다. 생성된 포트를 참조하고 런타임 시 생성된 포트가 없는 경우 매핑이 실패합니다. 그룹 기준 열을 매개 변수화하여 런타임 시 그룹화할 열을 나타낼 수 있습니다.

### 집계 변환의 생성된 포트 참조

집계 식에 생성된 포트를 포함할 수 있습니다. 런타임 시 포트가 없는 경우 매핑이 실패합니다. 집계 식의 동적 포트를 참조할 수 없습니다.

### 출력 포트에서 집계 식 작성

생성된 포트 또는 동적 포트에서 집계 식을 작성할 수 없습니다. 집계 식은 동적 식일 수 없습니다.

### 집계 식의 값 매개 변수화

집계 식에 매개 변수를 포함할 수 있습니다. 그러나 집계 식에서 식 매개 변수 또는 포트 매개 변수를 사용할 수 없습니다.

## 집계 변환 개발

집계 변환을 작성하는 경우 각 행에 대해 실행되도록 식을 정의합니다. 결과 기준을 반환하도록 그룹 기준 포트 목록을 정의합니다.

집계 변환을 작성하려면 다음 단계를 수행합니다.

1. 변환을 정의하고 포트를 작성합니다.
2. 변수 또는 출력 포트에 대한 집계 식을 구성합니다.
3. 집계 결과 기준을 반환하도록 포트 그룹을 정의합니다.

## 집계 변환 포트

집계 변환에는 그룹을 구성하고 식을 집계하는 데 사용할 수 있는 여러 포트 유형이 있습니다.

집계 변환 포트 보기에는 다음 필드가 있습니다.

### 이름

포트의 이름입니다.

### 유형

포트 데이터 유형입니다.

### 전체 자릿수

필드의 길이입니다.

### 소수 자릿수

숫자 데이터의 소수점 오른쪽에 있는 위치 수입니다.

### 입력

업스트림 변환의 데이터임을 나타냅니다.

### 출력

포트가 식의 값을 반환함을 나타냅니다. 식에는 비집계 식 및 조건절이 포함될 수 있습니다. 여러 개의 집계 출력 포트를 작성할 수 있습니다.

### 통과

포트가 데이터를 변경하지 않고 반환하는 입력-출력 포트임을 나타냅니다.

### 변수

포트가 식에 사용할 값 또는 계산을 저장할 수 있음을 나타냅니다. 변수 포트는 입력 또는 출력 포트일 수 없습니다. 변수 포트는 변환 내에서만 데이터를 전달합니다.

### 식

행 또는 행 그룹을 집계할 식입니다.

### 기본값

올바르지 않거나 Null 값이 있는 포트에 대한 기본값입니다.

### 입력 규칙

포트 이름 또는 데이터 유형에 따라 변환에 포트를 포함하거나 제외하도록 필터링하는 규칙 집합입니다. 동적 포트를 정의하는 경우 입력 규칙을 구성합니다.

## 집계 식

변수 포트 또는 집계 변환의 출력 포트에서 집계 식을 구성합니다. 포트가 입력 포트, 통과 포트 또는 동적 포트인 경우 식을 입력할 수 없습니다.

집계 식에는 집계 함수가 아닌 조건절 및 함수가 포함될 수 있습니다. 또한 집계 함수에는 다음과 같이 다른 집계 함수 내에 중첩된 집계 함수가 포함될 수 있습니다.

`MAX( COUNT( ITEM ))`

집계 식 결과는 변환의 그룹 기준 포트에 따라 다릅니다. 예를 들어 다음 집계 식은 판매한 항목의 전체 수량을 찾습니다.

`SUM( QUANTITY )`

그러나 동일한 식을 사용하고 **ITEM** 포트를 기준으로 그룹화하면 데이터 통합 서비스가 전체 수량을 항목별로 반환합니다.

모든 출력 포트에 집계 식을 작성하고 여러 집계 포트를 변환에 사용할 수 있습니다.

## 집계 함수

집계 변환 안에 집계 함수를 구성합니다. 다른 집계 함수 안에 하나의 집계 함수를 중첩할 수 있습니다.

변환 언어에는 다음 집계 함수가 포함됩니다.

- AVG
- COUNT
- FIRST
- LAST
- MAX
- MEDIAN
- MIN
- PERCENTILE
- STDDEV
- SUM
- VARIANCE

집계 변환의 식에 포트를 사용하고 집계 함수 내에서 포트를 사용하지 않는 경우 데이터 통합 서비스는 포트의 마지막 행을 사용하여 식을 처리합니다.

예를 들어 **COMMISSIONS** 및 **SALARY** 포트를 포함하는 집계 변환을 생성합니다. **SALARY** 포트는 그룹 기준 포트입니다.

출력 포트에 다음과 같은 식을 사용할 수 있습니다.

`SUM(COMMISSIONS)`

데이터 통합 서비스는 집계 함수를 처리하고 **COMMISSIONS** 포트 값의 합계를 출력 포트에 반환합니다.

이 식을 다음과 같은 식으로 수정할 수 있습니다.

`SUM(COMMISSIONS) + COMMISSIONS`

데이터 통합 서비스는 식을 처리할 때 **COMMISSIONS** 포트 값의 합계를 반환하고 **COMMISSIONS** 포트의 마지막 행 값을 출력 포트의 반환 값에 추가합니다.

다른 출력 포트의 경우 다음과 같은 식을 사용할 수 있습니다.

`SUM(COMMISSIONS) + SALARY`

데이터 통합 서비스는 식을 처리할 때 **COMMISSIONS** 포트 값의 합계를 반환하고 **SALARY** 포트의 마지막 행 값을 출력 포트의 반환 값에 추가합니다. **SALARY** 포트는 그룹 기준 포트이므로 **SALARY** 포트에서 각 행의 값은 동일합니다.

## 중첩 집계 함수

집계 변환에서는 여러 출력 포트에 여러 단일 수준 함수나 중첩 함수를 포함할 수 있습니다.

집계 변환에는 단일 수준 함수와 중첩 함수를 동시에 포함할 수 없습니다. 따라서 집계 변환의 출력 포트에 단일 수준 함수가 포함되어 있으면 해당 변환의 다른 포트에서 중첩 함수를 사용할 수 없습니다. 단일 수준 함수와 중첩 함수를 같은 집계 변환에 포함하면 **Developer tool**이 매핑 또는 맵셋을 잘못된 것으로 표시합니다. 단일 수준 함수와 중첩된 함수를 모두 작성해야 하는 경우 개별 집계 변환을 작성해야 합니다.

## 집계 식의 조건절

집계에 사용되는 행 수를 줄이려면 집계 식에 조건절을 사용합니다. TRUE 또는 FALSE로 평가되는 모든 절을 조건절로 사용할 수 있습니다.

예를 들어 분기별 할당량을 초과한 직원의 총 보수를 계산하려면 다음 식을 사용합니다.

```
SUM( COMMISSION, COMMISSION > QUOTA )
```

## 그룹 기준 포트

모든 입력 데이터에서 집계를 실행하는 대신 행 그룹을 정의하여 집계할 수 있습니다. 예를 들어 회사의 전체 매출을 계산하거나 지역별로 그룹화된 전체 매출을 찾을 수 있습니다.

집계 식의 그룹을 정의하려면 집계 변환에서 해당하는 입력, 입력/출력, 출력 및 변수 포트를 선택합니다. 여러 그룹 기준 포트를 선택하여 고유한 각 조합에 대한 새 그룹을 작성할 수 있습니다. 그러면 데이터 통합 서비스가 각 그룹에 정의된 집계를 수행합니다.

값을 그룹화하면 데이터 통합 서비스가 각 그룹에 대해 하나의 행을 생성합니다. 값을 그룹화하지 않은 경우 데이터 통합 서비스가 모든 입력 행에 대해 하나의 행을 반환합니다. 데이터 통합 서비스는 집계의 결과와 함께 각 그룹의 마지막 행을 반환합니다. 특정 행을 반환하도록 지정할 수 있습니다. 예를 들어 FIRST 집계 함수를 사용하는 경우 데이터 통합 서비스가 첫 번째 행을 반환합니다.

집계 변환에서 여러 그룹 기준 포트를 선택하면 데이터 통합 서비스가 포트 순서를 사용하여 그룹화 순서를 결정합니다. 그룹 순서는 결과에 영향을 미칠 수 있습니다. 적절한 그룹화가 이루어지도록 그룹 기준 포트의 순서를 지정하십시오. 그룹의 포트를 선택한 후 포트 순서를 변경할 수 있습니다.

예를 들어 Price\_Out이라는 출력 포트를 작성할 수 있습니다. Price\_Out에 대한 식은 SUM(수량 \* 가격)입니다. Store\_ID 및 항목을 그룹 기준 포트에 정의합니다. 변환이 상점 기준으로 각 항목에 대한 총 가격을 반환합니다.

입력 행에는 다음 데이터가 포함될 수 있습니다.

Store_ID	항목	수량	가격
101	배터리	3	2.99
101	배터리	1	3.19
101	배터리	2	2.59
101	AAA	2	2.45
201	배터리	1	1.99
201	배터리	4	1.59
301	배터리	1	2.45

데이터 통합 서비스는 다음 고유 그룹에 대해 집계 계산을 수행합니다.

Store_Id	항목
101	배터리
101	AAA
201	배터리
301	배터리

데이터 통합 서비스는 상점 기준 각 항목에 대한 (가격 \* 수량) 합계로 마지막 행의 Store\_ID, 항목, 수량 및 가격을 반환합니다.

Store_ID	항목	수량	가격	Price_Out
101	배터리	2	2.59	17.34
101	AAA	2	2.45	4.90

Store_ID	항목	수량	가격	Price_Out
201	배터리	4	1.59	8.35
301	배터리	1	2.45	2.45

## 그룹 기준 포트 구성

변환 **속성** 보기의 **그룹 기준** 탭에서 그룹 기준 포트를 정의합니다.

다음 이미지는 그룹 기준 탭을 보여 줍니다.

그룹 기준 탭에는 다음 옵션이 포함되어 있습니다.

### 지정 기준

**값** 또는 **매개 변수**를 선택합니다. 포트 이름을 사용하려면 **값**을 선택합니다. 포트 목록 매개 변수를 사용하려면 **매개 변수**를 선택합니다.

### 추가

수동으로 입력하는 포트 이름을 허용합니다. **추가**를 클릭하기 전에 올바른 포트 이름을 입력해야 합니다.

### 선택

**선택**을 클릭하여 그룹에 추가할 포트를 선택합니다. Developer tool에서는 선택할 변환의 포트 목록을 제공합니다.

### 위로 이동 및 아래로 이동

그룹의 포트 순서를 변경할 수 있습니다. 포트 이름을 선택한 다음 이동 단추 중 하나를 클릭하여 정렬 순서에서 위로 이동하거나 아래로 이동합니다.

## 그룹 기준 매개 변수

그룹에 포함할 하나 이상의 포트를 포함하는 포트 목록 매개 변수를 구성할 수 있습니다. 변환의 포트 목록에서 포트를 선택하여 포트 목록 매개 변수를 작성합니다.

다음 이미지는 매개 변수를 사용하여 그룹의 포트를 식별하는 경우 **그룹 기준** 탭을 보여 줍니다.



포트 목록 매개 변수를 찾거나 **새로 만들기**를 클릭하여 포트 목록 매개 변수를 작성할 수 있습니다. 포트 목록 매개 변수를 작성하도록 선택하는 경우 변환의 포트 목록에서 포트를 선택할 수 있습니다.

## 그룹 기준 포트의 기본값

그룹 기준 포트에 **Null** 값이 포함된 경우 데이터 통합 서비스가 그룹을 작성하지 않습니다. 그룹의 각 포트에 대한 기본값을 정의하여 **Null** 입력 값을 바꿀 수 있습니다. 그러면 데이터 통합 서비스가 집계 합계 행을 포함할 수 있습니다.

## 비집계 식

그룹 기준 포트에서 비집계 식을 사용하여 그룹을 수정하거나 바꿀 수 있습니다.

예를 들어 'AAA battery'를 그룹화하기 전에 바꾸려는 경우 다음 식을 사용하여 **CORRECTED\_ITEM**이라는 그룹 기준 출력 포트를 작성할 수 있습니다.

```
IIF( ITEM = 'AAA battery', battery, ITEM )
```

## 집계 캐시

집계 변환을 사용하는 매핑을 실행하면 데이터 통합 서비스가 인덱스 캐시 및 데이터 캐시를 메모리에 작성하여 변환을 실행합니다. 메모리 캐시에서 사용할 수 있는 공간보다 더 많은 공간이 데이터 통합 서비스에 필요한 경우 데이터 통합 서비스는 오버플로우 데이터를 캐시 파일에 저장합니다.

데이터 통합 서비스는 집계 변환에 대해 다음과 같은 캐시를 작성합니다.

- 그룹 기준 포트에 구성된 대로 그룹 값을 저장하는 인덱스 캐시.
- 그룹 기준 포트를 기반으로 계산을 저장하는 데이터 캐시.

데이터 통합 서비스는 정렬된 포트를 포함하는 집계 변환을 실행하는 데 캐시 메모리를 사용하지 않습니다. 정렬된 포트를 사용하는 집계 변환의 경우 캐시 메모리를 구성하지 않아도 됩니다.

## 집계 변환을 위한 정렬된 입력

정렬된 입력 옵션을 사용하여 집계 변환 성능을 향상시킬 수 있습니다.

정렬된 입력을 사용하는 경우 데이터 통합 서비스는 모든 데이터가 그룹별로 정렬된다고 가정하고 그룹에 대해 행을 읽을 때 집계 계산을 수행합니다. 필요한 경우 데이터 통합 서비스는 그룹 정보를 메모리에 저장합니다. 정렬된 입력 옵션을 사용하려면 정렬된 데이터를 집계 변환에 전달해야 합니다. 정렬된 입력을 사용하는 경우 집계 변환에서 정렬된 출력을 제공합니다.

정렬된 입력을 사용하지 않는 경우 데이터 통합 서비스는 읽을 때 집계 계산을 수행합니다. 데이터가 정렬되지 않았기 때문에 데이터 통합 서비스는 모든 집계 계산을 정확하게 하기 위해 전체 소스를 읽을 때까지 각 그룹의 데이터를 저장합니다.

예를 들어 하나의 집계 변환에 정렬된 입력 옵션과 함께 **STORE\_ID** 및 **ITEM** 그룹 기준 포트가 선택되어 있습니다. 다음 데이터를 집계 변환에 통과시키면 데이터 통합 서비스가 그룹 **201/battery**를 찾을 때 **101/battery** 그룹의 3개의 행에 대해 집계를 수행합니다.

STORE_ID	ITEM	QTY	PRICE
101	'battery'	3	2.99
101	'battery'	1	3.19
101	'battery'	2	2.59
201	'battery'	4	1.59
201	'battery'	1	1.99

정렬된 입력을 사용하면서 데이터를 제대로 미리 정렬하지 않으면 데이터 통합 서비스가 매핑 실행에 실패합니다.

### 정렬된 입력 조건

특정 조건으로 인해 정렬된 입력을 사용하지 못할 수 있습니다.

다음 조건 중 하나가 **true**인 경우 정렬된 입력을 사용할 수 없습니다.

- 집계 식에 중첩된 집계 함수가 포함됩니다.
- 소스 데이터가 데이터 기반입니다.

이 조건 중 하나가 **true**인 경우 데이터 통합 서비스는 사용자가 정렬된 입력을 사용하지 않는 것처럼 변환을 처리합니다.

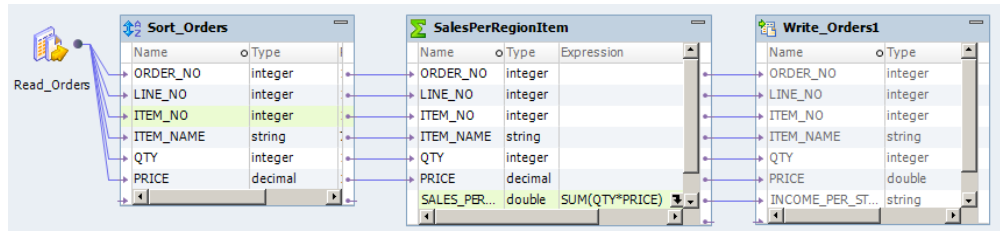
### 집계 변환에서 데이터 정렬

정렬된 입력을 사용하려면 집계 변환을 통해 정렬된 데이터를 전달합니다.

집계 변환에 나타나는 순서대로 집계 그룹 기준 포트별로 데이터를 정렬해야 합니다.

관계형 및 플랫폼 파일 입력의 경우 집계 변환에 전달하기 전에 분류기 변환을 사용하여 매핑의 데이터를 정렬합니다. 변환이 정렬된 데이터의 순서를 변경하지 않으면 분류기 변환을 집계 전에 매핑의 아무데나 배치할 수 있습니다. 집계 변환의 그룹 기준 열은 분류기 변환에 표시되는 것과 동일한 순서여야 합니다.

다음 매핑은 ITEM\_NO에 따라 오름차순으로 소스 데이터를 정렬하도록 구성된 분류기 변환을 보여줍니다.



분류기 변환이 다음과 같이 데이터를 정렬합니다.

ITEM_NO	ITEM_NAME	QTY	PRICE
345	Soup	4	2.95
345	Soup	1	2.95
345	Soup	2	3.25
546	Cereal	1	4.49
546	Cereal	2	5.25

정렬된 입력을 사용하면 집계 변환이 다음 결과를 반환합니다.

ITEM_NAME	QTY	PRICE	INCOME_PER_ITEM
Cereal	2	5.25	14.99
Soup	2	3.25	21.25

## 집계 변환 고급 속성

데이터 통합 서비스가 집계 변환용 데이터를 처리하는 방법을 결정할 수 있게 도와주는 속성을 구성합니다.

집계 변환에 대한 다음 고급 속성을 구성합니다.

### 캐시 디렉터리

데이터 통합 서비스가 인덱스 캐시 파일 및 데이터 캐시 파일을 작성하는 디렉터리입니다. 디렉터리가 존재하고 캐시 파일을 위한 디스크 공간이 충분한지 확인하십시오.

캐시 분할 중에 성능을 향상시키려면 여러 디렉터리를 세미콜론으로 구분하여 입력하십시오. 캐시 분할은 변환을 처리하는 각 파티션에 대해 별도의 캐시를 작성합니다.

기본값은 **CacheDir** 시스템 매개 변수입니다. 이 속성에 대해 다른 시스템 매개 변수를 구성하거나 사용자 정의 매개 변수를 구성할 수 있습니다.

### 데이터 캐시 크기

매핑 실행 시작 시 변환을 위해 데이터 통합 서비스가 데이터 캐시에 할당하는 메모리의 양입니다. "자동"을 선택하면 데이터 통합 서비스가 런타임 시 자동으로 메모리 요구 사항을 계산합니다. 캐시 크기를 조정하려는 경우 특정 값을 바이트 단위로 입력합니다. 기본값은 자동입니다.

### 인덱스 캐시 크기

매핑 실행 시작 시 변환을 위해 데이터 통합 서비스가 인덱스 캐시에 할당하는 메모리의 양입니다. "자동"을 선택하면 데이터 통합 서비스가 런타임 시 자동으로 메모리 요구 사항을 계산합니다. 캐시 크기를 조정하려는 경우 특정 값을 바이트 단위로 입력합니다. 기본값은 자동입니다.

### 정렬된 입력

입력 데이터가 그룹을 기준으로 미리 정렬되었음을 나타냅니다. 매핑에서 정렬된 데이터를 집계 변환에 전달하는 경우에만 이 옵션을 선택합니다.

### 추적 수준

이 변환에 대해 로그에 표시되는 세부 정보의 양입니다. 간단, 보통, 자세한 정보 표시 초기화 또는 자세한 정보 표시 데이터를 선택할 수 있습니다. 기본값은 보통입니다.

### 관련 항목:

- [“캐시 크기” 페이지 71](#)

## 재사용 가능한 집계 변환 작성

여러 매핑 또는 맵렛에서 사용할 수 있는 재사용 가능한 집계 변환을 작성합니다.

1. **개체 탐색기** 보기에서 프로젝트나 폴더를 선택합니다.
2. **파일 > 새로 만들기 > 변환**을 클릭합니다.  
새로 만들기 대화 상자가 나타납니다.
3. 집계 변환을 선택합니다.
4. **다음**을 클릭합니다.
5. 변환 이름을 입력합니다.
6. **마침**을 클릭합니다.  
변환이 편집기에 표시됩니다.
7. **추가** 단추를 클릭하여 포트를 변환에 추가합니다.
8. 포트를 편집하여 이름, 데이터 유형 및 전체 자릿수를 설정합니다.
9. 각 포트의 유형을 결정합니다. 입력, 출력, 통과 또는 변수 중에서 선택합니다.
10. 식 필드를 클릭하여 출력 포트에 대해 집계식을 구성합니다. 포트 및 매개 변수를 선택하여 집계 식을 정의할 수 있습니다.
11. **고급** 보기를 클릭하고 변환 속성을 편집합니다.

## 재사용 불가능 집계 변환 작성

매핑 또는 맵렛에서 재사용 불가능 집계 변환을 작성하십시오.

1. 매핑 또는 맵렛에서 집계 변환을 변환 팔레트에서 편집기로 끕니다.  
변환이 편집기에 표시됩니다.

2. **속성** 보기에서 변환 이름 및 설명을 편집합니다.
3. **포트** 탭에서 **새로 만들기** 단추를 클릭하여 포트를 변환에 추가합니다.
4. 포트를 편집하여 이름, 데이터 유형 및 전체 자릿수를 설정합니다.
5. 각 포트의 유형을 결정합니다. 입력, 출력, 통과 또는 변수 중에서 선택합니다.
6. 출력 포트에 대한 집계 식을 구성합니다.
7. **고급** 보기에서 변환 속성을 편집합니다.

## 집계 변환에 대한 팁

팁을 사용하면 집계 변환을 보다 효과적으로 사용할 수 있습니다.

### 정렬된 입력을 사용하여 집계 캐시의 사용을 줄입니다.

정렬된 입력은 매핑 실행 중에 캐시되는 데이터의 양을 줄여 성능을 향상시킵니다. 이 옵션을 분류기 변환과 함께 사용하면 정렬된 데이터가 집계 변환으로 전달됩니다.

### 연결된 입력/출력 또는 출력 포트를 제한합니다.

연결된 입력/출력 또는 출력 포트 수를 제한하여 집계 변환이 데이터 캐시에 저장하는 데이터의 양을 줄이십시오.

### 데이터를 집계하기 전에 필터링합니다.

매핑에서 필터 변환을 사용하는 경우 집계 변환 앞에 변환을 배치하여 불필요한 집계를 줄이십시오.

### 정렬된 집계 변환만 정렬된 출력을 제공합니다.

정렬되지 않은 입력을 사용하고 정렬된 출력을 생성하려는 경우 집계 변환 이후에 분류기 변환을 사용하십시오.

## 집계 변환 문제 해결

집계 변환의 문제를 해결할 수 있습니다.

### 정렬된 입력을 선택했지만 매핑에 이전과 동일한 시간이 걸립니다.

다음 조건 중 하나가 **true**인 경우 정렬된 입력을 사용할 수 없습니다.

- 집계 식에 중첩된 집계 함수가 포함됩니다.
- 소스 데이터가 데이터 기반입니다.

이 조건 중 하나가 **true**인 경우 데이터 통합 서비스는 사용자가 정렬된 입력을 사용하지 않는 것처럼 변환을 처리합니다.

### 집계 변환을 사용하는 매핑으로 인해 성능이 저하됩니다.

데이터 통합 서비스가 디스크로 페이지징할 수 있습니다. 변환 속성에서 인덱스 및 데이터 캐시 크기를 늘리면 성능을 향상시킬 수 있습니다.

## 집계 변환 - 비원시 환경

비원시 환경에서 집계 변환의 처리는 변환을 실행하는 엔진에 따라 다릅니다.

다음과 같은 비원시 런타임 엔진에 대한 지원을 고려하십시오.

- **Blaze** 엔진. 제한적으로 지원됩니다.
- **Spark** 엔진. 제한적으로 지원됩니다.
- **Databricks Spark** 엔진. 제한적으로 지원됩니다.

### 집계 변환 - Blaze 엔진

**Blaze** 엔진의 일부 처리 규칙은 데이터 통합 서비스의 처리 규칙과 다릅니다.

#### 매핑 유효성 검사

다음과 같은 상황에서는 매핑 유효성 검사가 실패합니다.

- 변환에 상태 저장 변수 포트가 포함되어 있습니다.
- 변환의 식에 지원되지 않는 함수가 포함되어 있습니다.

#### 집계 함수

집계 변환의 식에 포트를 사용하고 집계 함수 내에서 포트를 사용하지 않는 경우 런타임 엔진은 포트의 임의의 행을 사용하여 식을 처리할 수 있습니다.

런타임 엔진이 사용하는 행은 포트의 마지막 행이 아닐 수 있습니다. 처리가 분산되므로 런타임 엔진이 포트의 마지막 행을 결정하지 못할 수 있습니다.

#### 데이터 캐시 최적화

집계 변환을 통해 전달되는 이진 및 문자열 데이터 유형을 변수 길이를 사용하여 저장하도록 집계 변환의 데이터 캐시가 최적화됩니다. 최적화는 최대 **8MB**의 레코드 크기에 대해 활성화됩니다. 레코드 크기가 **8MB**를 초과하는 경우 변수 길이 최적화가 비활성화됩니다.

집계 변환을 통해 전달되는 데이터를 데이터 캐시에 저장할 때 변수 길이가 사용되는 경우 집계 변환은 정렬된 입력을 사용하도록 최적화되고 통과 분류기 변환은 런타임 매핑에서 집계 변환의 앞에 삽입됩니다.

분류기 변환을 보려면 최적화된 매핑을 보거나 **Blaze** 유효성 검사 환경의 실행 계획을 봅니다.

데이터 캐시 최적화 중에는 집계 변환에 대한 데이터 캐시 및 인덱스 캐시가 자동으로 설정됩니다. 분류기 변환에 대한 분류기 캐시는 집계 변환에 대한 데이터 캐시와 동일한 크기로 설정됩니다. 분류기 캐시를 구성하려면 집계 변환에 대한 데이터 캐시의 크기를 구성해야 합니다.

### 집계 변환 - Spark 엔진

**Spark** 엔진의 일부 처리 규칙은 데이터 통합 서비스의 처리 규칙과 다릅니다.

#### 매핑 유효성 검사

다음과 같은 상황에서는 매핑 유효성 검사가 실패합니다.

- 변환에 상태 저장 변수 포트가 포함되어 있습니다.
- 변환의 식에 지원되지 않는 함수가 포함되어 있습니다.

### 집계 함수

집계 변환의 식에 포트를 사용하고 집계 함수 내에서 포트를 사용하지 않는 경우 런타임 엔진은 포트의 임의 행을 사용하여 식을 처리할 수 있습니다.

런타임 엔진이 사용하는 행은 포트의 마지막 행이 아닐 수 있습니다. 처리가 분산되므로 런타임 엔진이 포트의 마지막 행을 결정하지 못할 수 있습니다.

### 데이터 캐시 최적화

변수 길이를 사용하여 데이터를 저장하도록 변환에 대한 데이터 캐시를 최적화할 수 없습니다.

## 집계 변환 - Databricks Spark 엔진

Databricks Spark 엔진의 일부 처리 규칙은 데이터 통합 서비스의 처리 규칙과 다릅니다.

### 매핑 유효성 검사

다음과 같은 상황에서는 매핑 유효성 검사가 실패합니다.

- 변환에 상태 저장 변수 포트가 포함되어 있습니다.
- 변환의 식에 지원되지 않는 함수가 포함되어 있습니다.

### 집계 함수

집계 변환의 식에 포트를 사용하고 집계 함수 내에서 포트를 사용하지 않는 경우 런타임 엔진은 포트의 임의 행을 사용하여 식을 처리할 수 있습니다.

런타임 엔진이 사용하는 행은 포트의 마지막 행이 아닐 수 있습니다. 처리가 분산되므로 런타임 엔진이 포트의 마지막 행을 결정하지 못할 수 있습니다.

### 데이터 캐시 최적화

변수 길이를 사용하여 데이터를 저장하도록 변환에 대한 데이터 캐시를 최적화할 수 없습니다.

## 제 6 장

# 연관 변환

이 장에 포함된 항목:

- [연관 변환 개요, 140](#)
- [메모리 할당, 141](#)
- [연관 변환 고급 속성, 141](#)

## 연관 변환 개요

연관 변환은 일치 변환의 출력 데이터를 처리합니다. 연관 변환은 서로 다른 일치 클러스터에 할당된 중복 레코드를 연결하는 링크를 작성해 데이터 통합 및 마스터 데이터 관리 작업에서 이러한 중복 레코드를 서로 연관할 수 있도록 합니다.

연관 변환은 연관된 레코드 그룹의 각 행에 대한 **AssociationID** 값을 생성하고 이 ID 값을 출력 포트에 기록합니다.

연관 변환의 출력은 통합 변환이 읽습니다. 통합 변환은 공통된 연관 ID 값을 포함하는 레코드를 바탕으로 마스터 레코드를 작성할 때 사용합니다.

연관 변환에서는 입력 포트에 문자열 및 숫자 데이터 값을 사용할 수 있습니다. 다른 데이터 유형의 입력 포트를 추가하는 경우 입력 포트의 데이터 값이 문자열로 변환됩니다.

**AssociationID** 출력 포트는 정수 데이터를 기록합니다. Informatica Data Quality의 이전 버전에서 연관 변환을 구성한 경우 연관 변환은 문자열 데이터를 **AssociationID** 포트에 쓸 수 있습니다.

### Example: 일치 변환 출력 연관

다음 표에는 동일한 개인으로 식별될 수 있는 레코드 3개가 포함되어 있습니다.

ID	이름	주소	시	주	우편 번호	SSN
1	David Jones	100 Admiral Ave.	뉴욕	NY	10547	987-65-4321
2	Dennis Jones	1000 Alberta Ave.	뉴저지	NY	-	987-65-4321
3	D. Jones	Admiral Ave.	뉴욕	NY	10547-1521	-

일치 변환에 정의된 중복 분석 작업은 이 3개의 레코드를 서로의 중복 레코드로 식별하지 않습니다. 이유는 다음과 같습니다.

- 이름 및 주소 데이터에 대한 중복 검색을 정의하는 경우 레코드 1과 3은 중복 레코드로 식별되지만 레코드 2는 생략됩니다.



- 이름 및 사회 보장 번호 데이터에 대한 중복 검색을 정의하는 경우 레코드 1과 2는 중복 레코드로 식별되지만 레코드 3은 생략됩니다.
- 3개 특성(이름, 주소 및 사회 보장 번호)에 대한 중복 검색을 정의하는 경우 일치 변환은 어떤 레코드도 일치로 식별하지 않습니다.

연관 변환은 서로 다른 일치 클러스터의 데이터를 연결하므로 클러스터 ID를 공유하는 레코드에는 공통된 AssociationID 값이 할당됩니다. 이 예에서는 다음 표에 표시된 것과 같이 동일한 AssociationID가 3개 레코드에 할당됩니다.

ID	이름	주소	시	주	우편 번호	SSN	이름 및 주소 클러스터 ID	이름 및 SSN 클러스터 ID	연관 ID
1	David Jones	100 Admiral Ave.	뉴욕	NY	10547	987-65-4320	1	1	1
2	Dennis Jones	1000 Alberta Ave.	뉴저지	NY	-	987-65-4320	2	1	1
3	D. Jones	Alberta Ave.	뉴욕	NY	10547-1521	-	1	2	1

중복 레코드 데이터는 통합 변환에서 통합할 수 있습니다.

## 메모리 할당

연관 변환이 사용하는 최소 캐시 메모리의 양을 설정할 수 있습니다. 기본 설정은 400,000바이트입니다.

고급 탭에서 **캐시 파일 크기** 속성 값을 설정합니다.

기본값은 변환이 사용하는 최소 메모리 양을 나타냅니다. 연관 변환은 연결하는 포트 수에 따라 여러 기본값을 가져오려고 합니다. 변환은 다음 공식을 사용하여 캐시 메모리를 가져옵니다.

(연관 포트 수 + 1) x 기본 캐시 메모리

예를 들어 7개의 연관 포트를 구성하는 경우 변환이 캐시 메모리에 320만 바이트 또는 3.05MB를 할당하려고 합니다.

기본 설정을 변경하는 경우 변환은 추가 메모리를 가져오려고 하지 않습니다.

**참고:** 65536보다 낮은 캐시 메모리 값을 입력하면 연관 변환은 값을 메가바이트로 읽습니다.

## 연관 변환 고급 속성

연관 변환에는 캐시 메모리 동작 및 추적 수준을 결정하는 고급 속성이 포함됩니다.

다음과 같은 고급 속성을 구성할 수 있습니다.

### 캐시 파일 디렉터리

데이터 통합 서비스에서 현재 변환에 대해 임시 데이터를 쓸 디렉터리를 지정합니다. 입력 데이터의 양이 사용 가능한 시스템 메모리보다 클 경우 데이터 통합 서비스가 이 디렉터리에 임시 파일을 기록합니다. 데이터 통합 서비스는 매핑을 실행한 후에 임시 파일을 삭제합니다.

속성에 디렉터리 경로를 입력하거나 매개 변수를 사용하여 디렉터를 식별할 수 있습니다. 데이터 통합 서비스 시스템의 로컬 경로를 지정하십시오. 데이터 통합 서비스가 기록할 수 있는 디렉터를 지정해야 합니다. 기본값은 **CacheDir** 시스템 매개 변수입니다.

### 캐시 파일 크기

변환의 입력 데이터를 정렬하기 위해 데이터 통합 서비스에서 사용하는 시스템 메모리의 양을 결정합니다. 기본값은 **400,000**바이트입니다. 매개 변수를 사용하여 캐시 파일 크기를 지정할 수 있습니다.

데이터 통합 서비스는 데이터를 정렬하기 전에 사용자가 지정한 메모리 양을 할당합니다. 정렬 작업이 더 많은 데이터를 생성할 경우 데이터 통합 서비스는 초과 데이터를 캐시 파일 디렉터리에 기록합니다. 정렬 작업에 시스템 메모리 및 파일 저장소가 제공할 수 있는 것보다 더 많은 메모리가 필요할 경우 매핑이 실패합니다.

**참고:** 65536 이상의 값을 입력한 경우 일치 변환이 값을 바이트로 읽습니다. 이보다 낮은 값을 입력하면 일치 변환이 값을 메가바이트로 읽습니다.

### 추적 수준

이 변환에 대해 로그에 표시되는 세부 정보의 양입니다. 간단, 보통, 자세한 정보 표시 초기화 또는 자세한 정보 표시 데이터를 선택할 수 있습니다. 기본값은 보통입니다.

## 제 7 장

# 잘못된 레코드 예외 변환

이 장에 포함된 항목:

- [잘못된 레코드 예외 변환 개요, 143](#)
- [잘못된 레코드 예외 출력 레코드 유형, 144](#)
- [잘못된 레코드 예외 관리 프로세스 흐름, 144](#)
- [잘못된 레코드 예외 매핑, 145](#)
- [잘못된 레코드 예외 포트, 147](#)
- [잘못된 레코드 예외의 구성 보기, 148](#)
- [잘못된 레코드 예외 문제 할당, 150](#)
- [예외 변환의 고급 속성, 151](#)
- [잘못된 레코드 예외 변환 구성, 151](#)
- [잘못된 레코드 예외 매핑 예제, 152](#)

## 잘못된 레코드 예외 변환 개요

잘못된 레코드 예외 변환은 데이터 품질 프로세스의 출력을 읽고 수동 검토가 필요한 레코드를 식별하는 활성 변환입니다. 잘못된 레코드 예외 변환은 여러 그룹이 포함되는 변환입니다.

잘못된 레코드 예외 변환을 구성하면 레코드의 데이터 품질 문제를 식별하는 프로세스의 출력을 분석할 수 있습니다. 여기에서 데이터 품질 문제가 있어 추가 검토가 필요한 레코드는 예외 레코드가 됩니다.

잘못된 레코드 예외 변환은 다른 변환 또는 다른 매핑의 데이터 개체로부터 입력을 수신합니다. 잘못된 레코드 변환에 대한 입력에는 데이터 품질 문제의 텍스트 설명을 수신하는 하나 이상의 품질 문제 포트가 포함되어야 합니다. 잘못된 레코드 예외 변환에 대한 입력에는 각 레코드의 데이터 품질을 결정하는 데 사용되는 숫자 레코드 점수도 포함될 수 있습니다. 예외 변환에서 상한 및 하한 점수 임계값을 설정하면 레코드 점수에 따라 양호한 품질 및 잘못된 품질로 레코드를 분류할 수 있습니다. 잘못된 레코드 예외 변환은 예외 및 연결된 품질 문제 텍스트를 잘못된 레코드 테이블에 기록합니다.

고객에게 우편물을 보내기 전에 고객 주소의 유효성을 검사해야 하는 경우를 예로 들겠습니다. 개발자는 라벨러 변환을 통해 참조 테이블을 바탕으로 도시, 주 및 우편 번호의 유효성을 검사하는 매핑을 작성할 수 있습니다. 라벨러 변환은 필드의 유효성을 검사하고 결과에 따라 레코드 점수를 각 행에 추가합니다. 또한 라벨러 변환은 오류가 있는 각 레코드의 품질 문제를 설명하는 텍스트도 추가합니다. 라벨러 변환은 `city not valid` 또는 `ZIP code blank`와 같은 품질 문제 텍스트를 각 예외 레코드에 추가합니다. 잘못된 레코드 예외 변환은 수동 검토가 필요한 고객 레코드를 잘못된 레코드 테이블에 기록합니다. 그러면 데이터 분석가가 **Analyst** 도구에서 잘못된 레코드를 검토하고 수정합니다.

## 잘못된 레코드 예외 출력 레코드 유형

잘못된 레코드 예외에서 입력 레코드 점수를 검사하여 레코드 품질을 확인합니다. 그리고 레코드를 다른 출력 그룹에 반환합니다.

예외 변환에서 각 레코드 점수를 기반으로 다음 유형의 레코드를 식별합니다.

### 양호한 레코드

상한 임계값보다 크거나 같은 점수를 가진 레코드입니다. 양호한 레코드는 올바르게 때문에 검토할 필요가 없습니다. 예를 들어, 상한 임계값을 90으로 구성한 경우 90 이상의 점수를 가진 레코드는 검토할 필요가 없습니다.

### 잘못된 레코드

상한 임계값보다 작고 하한 임계값보다 크거나 같은 점수를 가진 레코드입니다. 잘못된 레코드는 **Analyst** 도구에서 검토해야 하는 예외입니다. 예를 들어, 하한 임계값이 40일 경우 40에서 90 사이의 점수를 가진 레코드는 수동으로 검토해야 합니다.

### 거부된 레코드

하한 임계값보다 적은 점수를 가진 레코드입니다. 거부된 레코드는 유효하지 않습니다. 기본적으로 예외 변환에서 거부된 레코드를 데이터 흐름에서 삭제합니다. 이 예제에서는 40점 이하의 레코드가 거부된 레코드입니다.

**참고:** 품질 문제 필드가 NULL일 경우 예외 레코드가 아닙니다. 품질 문제에 텍스트가 포함되어거나 빈 문자열이 포함된 경우 레코드가 예외입니다. 필드에 오류가 없는 경우 품질 문제 포트에 **Null** 값이 포함되어 있는지 확인하십시오. 품질 문제 포트에 **Null** 값 대신 공백이 포함된 경우 예외 변환에서 모든 레코드를 예외로 플래그 지정합니다. 사용자가 **Analyst** 도구에서 문제를 수정해야 할 경우 데이터 품질 문제별로 예외를 필터링할 수 없습니다.

레코드에 0보다 작거나 100보다 큰 점수가 있는 경우 행이 올바르지 않습니다. 데이터 통합 서비스에서 행이 올바르지 않다는 오류 메시지를 기록하고 레코드 처리를 건너뛵니다.

레코드 점수를 예외 변환에 대한 입력으로 연결하지 않은 경우 변환에서 품질 문제가 포함된 모든 레코드를 잘못된 레코드 테이블에 기록합니다.

잘못된 레코드 예외 변환을 매핑 태스크에 포함한 경우 예외의 수동 검토를 포함하는 휴먼 태스크를 동일한 워크플로우에 구성할 수 있습니다. 워크플로우에서 매핑 태스크가 종료되면 휴먼 태스크가 시작됩니다. 휴먼 태스크에서는 사용자가 **Analyst** 도구에 액세스하여 품질 문제를 확인해야 합니다. 사용자가 잘못된 레코드 테이블에 있는 각 레코드의 품질 상태를 변경하고 데이터를 업데이트할 수 있습니다.

## 잘못된 레코드 예외 관리 프로세스 흐름

예외 변환은 데이터 품질 변환에서 레코드 점수를 받아 다른 수준의 데이터 품질을 가진 레코드가 포함된 테이블을 작성합니다. 품질 문제를 찾고 각 행에 대한 레코드 점수를 제공하도록 데이터 품질 변환을 구성해야 합니다.

단일 매핑에서 데이터 품질 변환을 구성하거나 데이터 품질 프로세스의 다른 단계에 대한 매핑을 작성할 수 있습니다.

다음 잘못된 레코드 예외 관리 태스크를 수행하십시오.

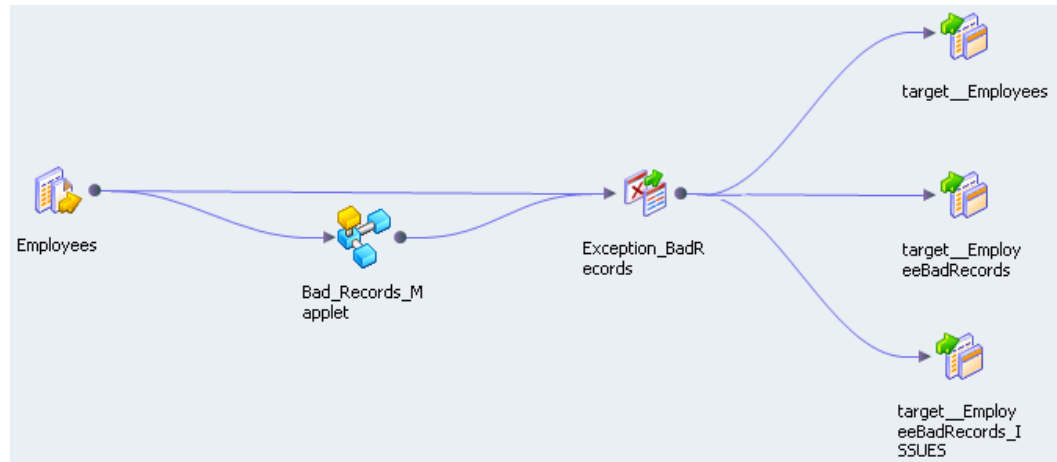
1. **Developer tool**에서 정의한 데이터 품질 문제를 기반으로 소스 데이터에 대한 점수 값을 생성하는 변환을 정의합니다. 소스 데이터의 품질을 설명하는 텍스트를 반환하는 변환을 정의합니다. 예를 들어, 소스 데이터를 참조 테이블에 대해 검사하도록 라벨러 변환을 구성한 다음 각 비교에 대한 설명 레이블을 작성할 수 있습니다. **IF/THEN** 규칙을 결정 변환에 정의하여 데이터 필드를 검사할 수 있습니다. 다른 데이터 품질 작업을 수행하는 여러 변환 및 맵셋을 정의할 수 있습니다.

2. 데이터 품질 작업에서 수신하는 레코드 점수를 분석하도록 예외 변환을 구성합니다. 레코드의 점수 값을 기반으로 데이터베이스 테이블에 레코드를 기록하도록 변환을 구성합니다. 양호한 레코드, 잘못된 레코드, 품질 문제 및 거부된 레코드에 대한 별도 테이블을 작성할 수 있습니다.
3. 잘못된 데이터를 포함하는 각 입력 포트에 품질 문제 포트를 할당합니다.
4. 필요에 따라 양호한 레코드 및 잘못된 레코드에 대한 대상 데이터 개체를 구성합니다. 매핑에서 예외 변환 출력 포트를 대상 데이터 개체에 연결합니다.
5. 잘못된 레코드에 대한 대상 데이터 개체를 작성합니다. 잘못된 레코드 테이블을 생성하여 매핑에 추가하도록 선택합니다. 잘못된 레코드 테이블을 생성할 경우 **Developer tool**에서도 품질 문제 테이블을 생성합니다. 품질 문제 테이블을 매핑에 추가합니다.
6. 매핑을 워크플로우에 추가합니다.
7. 잘못된 레코드의 수동 검토를 사용자에게 할당하도록 휴먼 태스크를 구성합니다. 사용자는 **Analyst** 도구에서 잘못된 레코드를 검토 및 업데이트할 수 있습니다.

## 잘못된 레코드 예외 매핑

잘못된 레코드 예외를 식별하는 매핑을 작성할 경우 레코드의 데이터 품질에 따라 하나 이상의 데이터베이스 대상에 레코드를 기록하는 매핑을 구성하십시오.

다음 그림에서는 잘못된 레코드 예외 매핑의 예를 보여 줍니다.



이 매핑에는 다음 개체가 포함됩니다.

### 데이터 소스

데이터 품질을 분석할 레코드가 포함된 직원 데이터 소스입니다.

### 맵릿

**Bad\_Records\_Maplet**에는 품질 문제를 검사 및 추가하고 점수를 소스 레코드에 기록하는 변환이 포함되어 있습니다. 규칙은 데이터를 분석하고 품질 문제를 찾는 변환입니다. 예를 들어, 입력 데이터를 참조 테이블과 비교하기 위해 라벨러 변환을 포함할 수 있습니다. 결과에 따라 품질 문제를 행에서 추가 열로 반환하도록 라벨러 변환을 구성할 수 있습니다. **IF, THEN, ELSE** 문을 사용하여 데이터를 검사하는 결정 변환을 구성하고 품질 문제를 적용하며 점수를 입력 데이터에 기록할 수 있습니다.

### 예외 변환

예외 변환에서 잘못된 레코드 테이블 및 문제 테이블을 포함한 데이터 대상에 기록할 레코드를 결정합니다.

### 양호한 레코드 테이블

예외 변환에서 모든 양호한 품질의 레코드를 `target_Employees` 테이블에 기록합니다.

### 잘못된 레코드 테이블

예외 변환에서 모든 잘못된 품질의 레코드를 `target_EmployeeBadRecords` 테이블에 기록합니다. 잘못된 레코드를 수동으로 검토해야 합니다.

### 문제 테이블

예외 변환에서 품질 문제를 `target_EmployeeBadRecords_ISSUES` 테이블에 기록합니다. `Analyst` 도구에서 잘못된 레코드를 조회할 경우 사용자 인터페이스가 품질 문제를 잘못된 레코드에 연결합니다.

필요에 따라 예외 변환에서 거부된 레코드를 거부된 레코드 테이블에 기록할 수 있습니다. 변환의 **구성** 보기에서 거부된 레코드에 대한 별도의 출력 그룹을 작성하도록 선택해야 합니다.

## 잘못된 레코드 예외 품질 문제

품질 문제는 낮은 레코드 점수를 발생시킨 데이터 품질 문제 유형을 설명하는 텍스트 문자열입니다. 잘못된 레코드 예외 변환은 낮은 레코드 점수가 포함된 각 소스 행과 연결된 품질 문제를 수신합니다. 품질 문제 및 레코드 점수를 결정하는 다른 유형의 변환을 구성할 수 있습니다.

예를 들어, 전화 번호를 검사하는 결정 변환을 작성할 수 있습니다. 결정 변환에서 전화 번호에 대한 레코드 점수 및 품질 문제를 생성합니다.

다음 결정 전략에서는 결정 변환에서 길이가 잘못된 전화 번호를 식별합니다.

```
IF LENGTH(Phone_Number) > 10 THEN
  Score:=50
  Phone_Quality_Issue:='Phone num too long'
ELSEIF LENGTH(Phone_Number) < 10 THEN
  Score:=50
  Phone_Quality_Issue:=' Phone num too short'
ELSE
  Score:=90
ENDIF
```

예외 변환을 구성할 경우 `Phone_Quality_Issue`를 `Phone_Number` 포트와 연결해야 합니다. 포트는 다른 입력 그룹의 포트입니다.

예외 변환이 결정 변환에서 생성한 점수를 읽고, 점수가 "50"인 레코드를 출력 포트의 잘못된 레코드 그룹에 할당합니다. 그리고 `Phone_Quality_Issue`를 출력 포트의 문제 그룹에 기록합니다.

## 휴먼 태스크

예외 변환을 포함하는 워크플로우를 구성할 때 매핑 태스크의 매핑을 포함할 수 있습니다. 이 동일한 워크플로우에 휴먼 태스크를 추가할 수도 있습니다. 휴먼 태스크가 포함되면 한 명 이상의 사용자가 `Analyst` 도구에서 예외 레코드를 수정해야 합니다.

매핑 태스크가 소스 데이터에서 해결되지 않은 데이터 품질 문제를 포함하는 레코드를 식별합니다. 데이터 분석가는 `Analyst` 도구를 사용하여 문제를 해결하고 각 레코드의 데이터 품질 상태를 업데이트할 수 있습니다.

휴먼 태스크를 구성할 때는 태스크 인스턴스와 태스크 단계를 각각 하나 이상 작성합니다. 태스크 인스턴스는 사용자가 작업해야 하는 데이터 집합을 나타냅니다. 태스크 단계는 사용자가 태스크 인스턴스의 레코드에 대해 수행하는 작업 유형을 나타냅니다. 여러 사용자가 `Analyst` 도구에서 데이터의 서로 다른 부분을 작업할 수 있도록 여러 개의 태스크 인스턴스를 작성할 수 있습니다.

사용자는 `Analyst` 도구에서 다음과 같은 방법으로 잘못된 레코드의 상태를 업데이트할 수 있습니다.

- 레코드가 유효한 경우 테이블 메타데이터를 업데이트하여 레코드가 데이터베이스에 영구적으로 저장되도록 합니다.

- 레코드가 유효하지 않으면 워크플로우의 이후 단계에서 레코드가 데이터베이스에서 제거되도록 테이블 메타데이터를 업데이트합니다.
- 레코드 상태를 확인할 수 없는 경우에는 레코드를 워크플로우로 반환하여 매핑 태스크에서 추가로 처리하도록 테이블 메타데이터를 업데이트합니다.

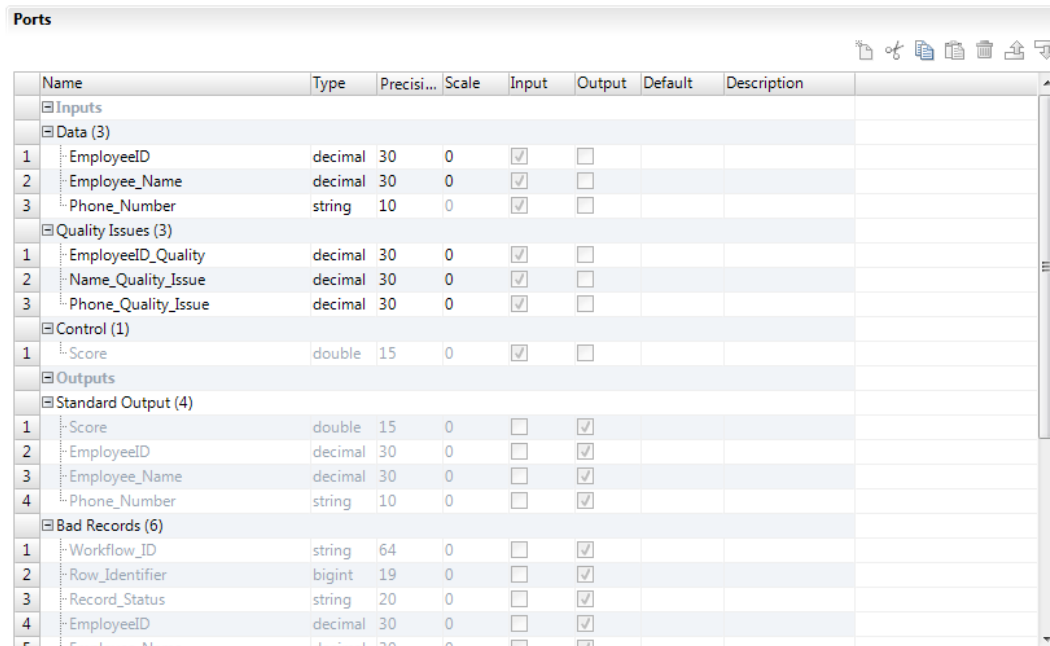
휴먼 태스크에 대한 자세한 내용은 *Informatica Developer 워크플로우 가이드*를 참조하십시오.

## 잘못된 레코드 예외 포트

잘못된 레코드 예외 변환의 **포트** 탭에서 입력 및 출력 포트를 구성합니다.

잘못된 레코드 예외 변환에는 입력 및 출력 포트 그룹이 포함됩니다.

다음 그림은 **포트** 탭을 보여 줍니다.



	Name	Type	Precisi...	Scale	Input	Output	Default	Description
<b>Inputs</b>								
<b>Data (3)</b>								
1	EmployeeID	decimal	30	0	<input checked="" type="checkbox"/>	<input type="checkbox"/>		
2	Employee_Name	decimal	30	0	<input checked="" type="checkbox"/>	<input type="checkbox"/>		
3	Phone_Number	string	10	0	<input checked="" type="checkbox"/>	<input type="checkbox"/>		
<b>Quality Issues (3)</b>								
1	EmployeeID_Quality	decimal	30	0	<input checked="" type="checkbox"/>	<input type="checkbox"/>		
2	Name_Quality_Issue	decimal	30	0	<input checked="" type="checkbox"/>	<input type="checkbox"/>		
3	Phone_Quality_Issue	decimal	30	0	<input checked="" type="checkbox"/>	<input type="checkbox"/>		
<b>Control (1)</b>								
1	Score	double	15	0	<input checked="" type="checkbox"/>	<input type="checkbox"/>		
<b>Outputs</b>								
<b>Standard Output (4)</b>								
1	Score	double	15	0	<input type="checkbox"/>	<input checked="" type="checkbox"/>		
2	EmployeeID	decimal	30	0	<input type="checkbox"/>	<input checked="" type="checkbox"/>		
3	Employee_Name	decimal	30	0	<input type="checkbox"/>	<input checked="" type="checkbox"/>		
4	Phone_Number	string	10	0	<input type="checkbox"/>	<input checked="" type="checkbox"/>		
<b>Bad Records (6)</b>								
1	Workflow_ID	string	64	0	<input type="checkbox"/>	<input checked="" type="checkbox"/>		
2	Row_Identifier	bigint	19	0	<input type="checkbox"/>	<input checked="" type="checkbox"/>		
3	Record_Status	string	20	0	<input type="checkbox"/>	<input checked="" type="checkbox"/>		
4	EmployeeID	decimal	30	0	<input type="checkbox"/>	<input checked="" type="checkbox"/>		
5	Employee_Name	decimal	30	0	<input type="checkbox"/>	<input checked="" type="checkbox"/>		

## 잘못된 레코드 예외 변환 입력 포트

잘못된 레코드 예외 변환에는 데이터, 품질 문제 및 레코드 점수에 대한 개별 입력 그룹이 있습니다.

잘못된 레코드 예외 변환에는 다음 입력 그룹이 있습니다.

### 데이터

소스 데이터 필드입니다.

### 품질 문제

레코드 품질 문제를 설명하는 포트가 포함되어 있습니다. 품질 문제 포트에는 "Excess\_Characters" 또는 "Bad\_Data\_Format" 같은 문자열이 포함되어 있습니다. 레코드마다 여러 품질 문제가 있을 수 있습니다.

**문제 할당** 보기에서 데이터 포트에 문제를 할당할 때까지 변환에서 품질 문제 그룹의 포트를 소스 데이터 필드에 연결하지 않습니다.

## 제어

레코드 점수입니다. 예외 변환에서 레코드 점수를 분석하여 입력 행이 예외인지 확인합니다. 점수 포트를 연결하지 않은 경우 품질 문제 포트에 데이터가 포함되어 있으면 예외 변환에서 행을 예외로 식별합니다.

## 잘못된 레코드 예외 변환 출력

잘못된 레코드 예외 변환에는 여러 개의 출력 그룹이 있습니다.

잘못된 레코드 예외 변환의 출력 그룹은 다음과 같습니다.

### 표준 출력

데이터 품질 문제를 검사할 필요가 없는 양호한 품질의 레코드입니다.

표준 출력 그룹의 각 레코드에는 레코드의 데이터 품질을 나타내는 점수 포트가 포함됩니다.

### 잘못된 레코드

데이터 품질 문제를 검사해야 하는 예외 레코드입니다.

잘못된 레코드 그룹의 각 레코드에는 워크플로우 ID, 행 식별자 및 레코드 상태 포트가 포함됩니다.

### 문제

잘못된 레코드 그룹의 레코드에 대한 품질 문제입니다. 품질 문제는 잘못된 레코드를 검토할 때 **Analyst** 도구에 표시되는 메타데이터 요소입니다.

문제 그룹의 각 레코드에는 워크플로우 ID 및 행 식별자 포트가 포함됩니다. 행 식별자 포트는 문제가 속하는 잘못된 레코드 행을 식별합니다.

### 거부된 레코드

데이터베이스에서 제거할 수 있는 레코드를 포함하는 선택적 그룹입니다. 거부된 레코드 그룹의 각 레코드는 점수 포트에 낮은 레코드 점수가 포함됩니다.

## 잘못된 레코드 예외의 구성 보기

**구성** 보기는 양호한 레코드와 잘못된 레코드를 식별하기 위해 변환에서 사용하는 상한 및 하한 임계값을 지정합니다. 또한 **구성** 보기는 임계값보다 높거나 낮은 점수를 가진 레코드의 대상 테이블을 식별합니다.

다음 그림은 예외 변환의 **구성** 보기를 보여 줍니다.



[illegible]

구성 보기에서 다음 속성을 구성할 수 있습니다.

잘못된 레코드 점수 범위의 하한입니다. 변환은 하한 임계값보다 낮은 점수를 가진 레코드를 거부된 레코드로 식별합니다.

잘못된 레코드 점수 범위의 상한입니다. 변환은 상한 임계값보다 높거나 같은 점수를 가진 레코드를 양호한 레코드로 식별합니다.

출력 레코드의 유형입니다. 기본 구성에서, 변환은 양호한 레코드를 표준 출력에 쓰고 잘못된 레코드를 잘못된 레코드 테이블에 씁니다. 변환은 기본적으로 거부된 레코드를 데이터베이스 테이블에 쓰지 않습니다.

변환에서 표준 출력 포트에 쓸 레코드 유형입니다. 기본값은 양호한 레코드입니다.

예외 변환에서 잘못된 레코드 출력 포트에 쓸 레코드 유형입니다. 기본값은 잘못된 레코드입니다.

거부된 레코드에 대해 별도의 출력 그룹을 작성합니다. 옵션은 기본적으로 지워져 있습니다.

잘못된 레코드 데이터를 포함할 데이터베이스 테이블을 작성합니다. 이 옵션을 선택하면 예외 변환이 데이터베이스 테이블을 작성하고, 데이터 개체를 모델 리포지토리에 추가하고, 개체 인스턴스를 매핑 캔버스에 추가합니다. 매핑의 예외 변환 인스턴스에 대한 잘못된 레코드 테이블을 생성할 수 있습니다. 잘못된 레코드 테이블을 생성하면 **Developer tool**도 레코드에 대한 설명 메타데이터를 저장하기 위해 문제 테이블을 작성합니다.

## 잘못된 레코드 테이블 및 문제 테이블 생성

매핑에 변환을 포함하는 경우 잘못된 레코드 테이블 및 문제 테이블을 생성할 수 있습니다. Developer tool은 테이블을 모델 리포지토리에 추가합니다.

1. **잘못된 레코드 테이블 생성**을 클릭하여 테이블을 생성합니다.

관계형 데이터 개체 작성 대화 상자가 표시됩니다.

2. 데이터베이스 연결을 찾습니다. 테이블을 포함할 데이터베이스에 대한 연결을 선택합니다.
3. 잘못된 레코드 테이블의 이름을 입력합니다. Developer tool이 입력한 이름을 잘못된 레코드 테이블과 문제 테이블에 적용합니다.

Developer tool이 다음 문자열을 문제 테이블 이름에 추가합니다.

`_ISSUE`

Oracle 데이터베이스에 연결하는 경우 잘못된 레코드 테이블 이름이 24자를 초과해서는 안 됩니다.

4. 잘못된 레코드 데이터 개체의 이름을 모델 리포지토리에 입력합니다.
5. **마침**을 클릭합니다.

Developer tool이 매핑 캔버스 및 모델 리포지토리에 테이블을 추가합니다.

## 잘못된 레코드 예외 문제 할당

데이터 품질 문제에 포트 및 우선 순위를 할당해야 합니다.

다음 그림에는 **문제 할당** 보기가 표시되어 있습니다.

Issue Assignment		
Assign ports and priorities to quality issues		
Quality Issue	Input	Issue...
EmployeeID_Quality_Issue	EmployeeID	1
Name_Quality_Issue	Employee_Name	1
Phone_Quality_Issue		1

**문제 할당** 보기에는 다음 필드가 포함되어 있습니다.

### 품질 문제

품질 문제 입력 그룹에서 정의한 품질 문제 포트가 **품질 문제** 열에 표시됩니다.

### 입력

**입력** 열에는 **문제 할당** 보기에서 품질 문제에 할당하는 데이터 포트가 포함되어 있습니다. 입력 포트를 품질 문제 포트와 연결합니다. 잘못된 품질 데이터를 포함하는 입력 포트에 문제 유형을 나타내는 해당하는 품질 문제 포트가 1개 이상 있어야 합니다. 예를 들어, **Phone\_Quality\_Issue**에 대해 **Phone\_Number** 포트를 선택할 수 있습니다. 둘 이상의 품질 문제에 포트를 할당할 수 있습니다.

### 문제 우선 순위

문제 우선 순위는 동일한 입력 포트를 여러 품질 문제에 할당할 때 가장 중요한 품질 문제를 결정합니다. 하나의 입력 필드에 대해 둘 이상의 품질 문제가 발생할 경우 데이터 통합 서비스는 가장 높은 우선 순위 품질 문제를 적용합니다. 하나의 입력 포트에 둘 이상의 품질 문제가 있고 문제 우선 순위가 동일한 경우, 데이터 통합 서비스는 목록에서 맨 위에 있는 품질 문제를 적용합니다. 1에서 99 사이의 우선 순위를 입력합니다. 여기서 1은 가장 높은 우선 순위를 나타냅니다.

Analyst 도구에서 레코드를 필터링하려면 문제 우선 순위를 정의하십시오.

## 품질 문제에 포트 할당

품질 문제와 연결할 포트를 할당하십시오. 사용자가 **문제 할당** 보기에서 추가한 각 연결마다 **Developer tool**이 문제 출력 그룹에 포트를 작성합니다.

1. 품질 문제마다 **입력** 필드를 클릭하여 입력 포트 목록을 표시합니다.
2. 품질 문제와 연결할 입력 포트를 선택합니다.  
둘 이상의 문제에 대해 동일한 포트를 선택할 수 있습니다.
3. **문제** 열을 클릭하고 품질 문제에 대한 우선 순위를 선택합니다.

## 예외 변환의 고급 속성

예외 변환 데이터에 대한 데이터 통합 서비스의 처리 방식을 결정하는 속성을 구성할 수 있습니다.

로그의 추적 수준을 구성할 수 있습니다.

**고급** 탭에서 다음 속성을 구성합니다.

### 추적 수준

이 변환에 대해 로그에 표시되는 세부 정보의 양입니다. 간단, 보통, 자세한 정보 표시 초기화 또는 자세한 정보 표시 데이터러를 선택할 수 있습니다. 기본값은 보통입니다.

## 잘못된 레코드 예외 변환 구성

잘못된 레코드 예외 변환을 구성할 때는 입력 포트와 각 포트에서 발생할 수 있는 품질 문제를 구성합니다. 데이터 품질을 결정하기 위한 상한 및 하한 임계값을 정의합니다. 예외 및 거부 레코드를 기록할 위치를 구성합니다.

1. 재사용 가능 또는 재사용 불가능한 잘못된 레코드 예외 변환을 작성합니다.
  - 재사용 가능 변환을 작성하려면 **파일 > 새로 만들기 > 변환**을 선택하고 잘못된 레코드 예외 변환을 선택합니다.
  - 재사용 불가능 변환을 작성하려면 매핑을 열고 매핑 캔버스에 예외 변환을 추가합니다. 마법사에서 잘못된 레코드 예외 변환을 선택합니다.
2. **다음**을 클릭하거나 **마침**을 클릭합니다.  
**다음**을 클릭하는 경우 변환을 작성하기 전에 기본 임계값 및 데이터 라우팅 옵션을 업데이트할 수 있습니다.
3. 입력 포트를 구성합니다.
  - 재사용 가능한 변환을 작성하는 경우 **포트** 탭을 선택하고 연결할 데이터의 포트를 변환에 추가합니다.
  - 재사용할 수 없는 변환을 작성하는 경우 다른 개체를 매핑 캔버스에 추가하고 입력 포트를 변환으로 끌어옵니다.
4. **구성** 보기를 선택합니다.
5. 상한 및 하한 점수 임계값을 구성합니다.
6. **데이터 라우팅 옵션** 섹션에서 표준 출력 및 예외 테이블 속성을 구성하여 변환이 각 유형의 레코드를 기록할 위치를 설정합니다.

양호한 레코드, 잘못된 레코드 및 거부된 레코드를 기록할 위치를 구성합니다. 표준 출력 또는 잘못된 레코드 테이블에 레코드를 기록할 수 있습니다.

7. **문제 할당** 보기를 엽니다. 데이터 품질 문제를 데이터 포트에 할당합니다.

각 문제에 우선 순위를 할당합니다. 포트에 여러 문제가 있는 값이 포함되는 경우 변환이 우선 순위가 가장 높은 문제를 표시합니다.

8. 잘못된 레코드 테이블을 생성하는 옵션을 선택합니다. 데이터베이스 연결 및 테이블 이름 정보를 입력합니다. 테이블은 기본 스키마에서 가져와야 합니다.

- 잘못된 레코드 테이블을 생성할 때는 레코드에 대한 테이블 및 레코드와 관련된 데이터 품질 문제에 대한 추가 테이블을 생성해야 합니다. 변환이 모델 리포지토리에 데이터베이스 개체를 작성합니다.

9. 변환 출력 포트를 하나 이상의 데이터 대상에 연결합니다. **구성** 보기에서 설정한 출력 옵션에 해당하는 데이터 개체에 출력 포트를 연결합니다.

- 재사용 가능한 변환을 작성하는 경우 변환을 매핑에 추가하고 출력 포트를 연결합니다.
- 재사용할 수 없는 변환을 작성하는 경우 변환이 포트를 잘못된 레코드 테이블에 연결합니다. 다른 데이터 대상에는 사용자가 출력 포트를 연결합니다.

## 잘못된 레코드 예외 매핑 예제

새 고객 데이터를 검토하는 데이터 프로젝트를 시행하는 조직이 있습니다. 이 조직은 고객 연락처 데이터가 유효한지 확인해야 합니다. 다음 예는 고객 레코드의 데이터 품질 분석을 수행하는 맵셋으로부터 레코드를 수신하는 잘못된 레코드 예외 변환을 정의하는 방법을 보여 줍니다.

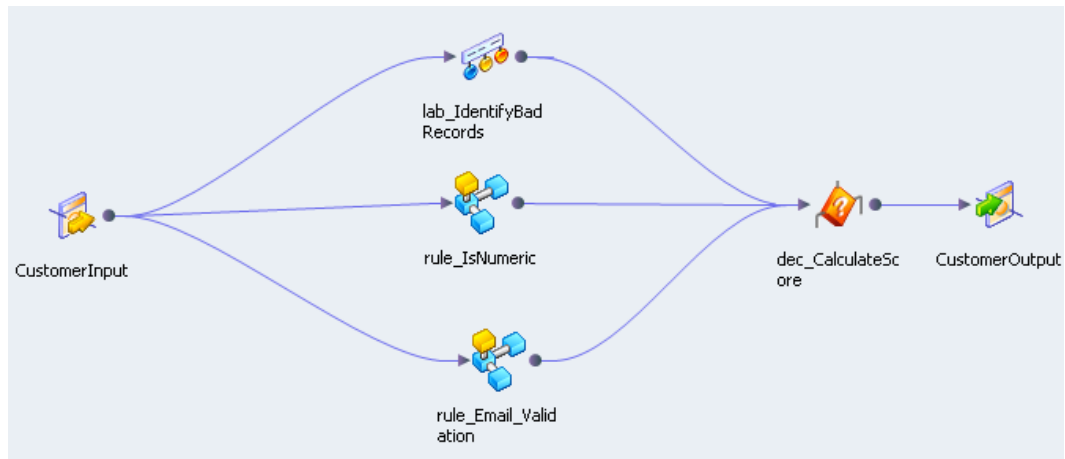
이 예에서는 고객 데이터의 형식 및 정확도를 평가하는 데이터 품질 변환이 포함된 맵셋을 작성합니다. 이 맵셋에는 데이터 품질 분석의 결과에 따라 레코드 점수를 생성하는 변환이 포함됩니다. 이 변환은 분석의 결과에 따라 데이터의 품질 문제도 정의합니다.

### 잘못된 레코드 예외 맵셋

데이터 품질 변환이 포함된 맵셋을 작성하여 특정 필드의 값을 검사합니다. 데이터 품질 변환은 참조 테이블 및 콘텐츠 집합을 검사하여 레코드의 필드가 유효한지 여부를 결정하고 결과에 따라 각 레코드에 레코드 점수를 적용합니다. 예외 변환은 이 맵셋으로부터 레코드를 수신하고 레코드 점수에 따라 각 레코드를 해당하는 출력으로 라우팅합니다.

맵셋은 라벨러 변환, 결정 변환 및 식 변환으로 구성됩니다.

다음 그림은 맵셋의 개체를 보여 줍니다.



이 맵렛은 다음 태스크를 수행합니다.

- 라벨러 변환이 입력 포트에서 수신한 로컬리티, 시/도, 국가 코드 및 우편 번호 데이터를 확인합니다. 라벨러 변환에는 각 포트에 대한 전략이 포함됩니다. 전략은 소스 데이터와 참조 테이블을 비교하고 유효하지 않은 값을 식별합니다.
- 식 변환 맵렛이 전화 번호가 숫자인지 확인하고 숫자가 10자리인지 확인합니다.
- 라벨러 변환 및 식 변환 맵렛이 전자 메일 주소가 유효한지 확인합니다. 식 변환은 전자 메일 문자열의 구조를 확인합니다. 라벨러 변환은 국제 IP 주소 접미사에 대한 참조 테이블을 기준으로 IP 주소를 검사합니다.
- 결정 변환이 변환 및 맵렛의 출력을 수신하고 고객 연락처 레코드에 대한 전체 레코드 점수를 계산합니다.

이러한 맵렛을 포함하는 잘못된 레코드 예외 매핑을 작성하십시오. 잘못된 레코드 예외 매핑에는 잘못된 레코드 데이터베이스 테이블에 예외를 기록하는 예외 변환이 포함됩니다. 데이터 분석가는 **Analyst** 도구에서 잘못된 레코드 테이블의 예외 레코드를 조사하고 업데이트할 수 있습니다.

## 잘못된 레코드 예외 예제 입력 그룹

예외 변환에는 3개 입력 그룹이 있습니다. 변환에는 소스 데이터를 받는 데이터 그룹이 있습니다. 데이터 품질 변환에서 찾은 데이터 품질 문제를 수신하는 품질 문제 그룹이 있습니다. 행에 대한 레코드 점수를 포함하는 제어 그룹도 있습니다.

다음 그림은 예외 변환의 입력 그룹을 보여 줍니다.

	Name	Type	Precision	Scale	Input	Output
	Inputs					
	Data (11)					
1	CUST_ID	decimal	20	0	<input checked="" type="checkbox"/>	<input type="checkbox"/>
2	COMPANY	string	100	0	<input checked="" type="checkbox"/>	<input type="checkbox"/>
3	CONTACT	string	100	0	<input checked="" type="checkbox"/>	<input type="checkbox"/>
4	TITLE	string	100	0	<input checked="" type="checkbox"/>	<input type="checkbox"/>
5	ADDR1	string	100	0	<input checked="" type="checkbox"/>	<input type="checkbox"/>
6	ADDR2	string	100	0	<input checked="" type="checkbox"/>	<input type="checkbox"/>
7	ADDR3	string	100	0	<input checked="" type="checkbox"/>	<input type="checkbox"/>
8	ADDR4	string	30	0	<input checked="" type="checkbox"/>	<input type="checkbox"/>
9	COUNTRY	string	20	0	<input checked="" type="checkbox"/>	<input type="checkbox"/>
10	PHONE	string	50	0	<input checked="" type="checkbox"/>	<input type="checkbox"/>
11	EMAIL	string	100	0	<input checked="" type="checkbox"/>	<input type="checkbox"/>
	Quality Issues (7)					
1	CompanyStatus	string	30	0	<input checked="" type="checkbox"/>	<input type="checkbox"/>
2	LocalityStatus	string	30	0	<input checked="" type="checkbox"/>	<input type="checkbox"/>
3	ProvinceStatus	string	30	0	<input checked="" type="checkbox"/>	<input type="checkbox"/>
4	CountryStatus	string	30	0	<input checked="" type="checkbox"/>	<input type="checkbox"/>
5	ZipStatus	string	30	0	<input checked="" type="checkbox"/>	<input type="checkbox"/>
6	PhoneStatus	string	30	0	<input checked="" type="checkbox"/>	<input type="checkbox"/>
7	EmailStatus	string	30	0	<input checked="" type="checkbox"/>	<input type="checkbox"/>
	Control (1)					
1	Score	double	15	0	<input checked="" type="checkbox"/>	<input type="checkbox"/>

## 잘못된 레코드 예외 예 구성

**구성** 보기에서 상한 및 하한 임계값을 정의합니다. 변환이 양호한 레코드, 잘못된 레코드 및 거부된 레코드를 기록하는 위치를 식별합니다.

양호한 레코드, 잘못된 레코드 및 문제 라우팅에 대한 기본 구성을 그대로 사용합니다.

다음 그림은 예외 변환의 **구성** 보기를 보여 줍니다.

**Manual Review Thresholds**

Lower Threshold :  10.00

Upper Threshold :  90.00

**Data Routing Options**

Type	Standard Output	Bad Record Table
Good Records (Above upper threshold)	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Bad Records (Between thresholds)	<input type="checkbox"/>	<input checked="" type="checkbox"/>
Rejected Records (Below lower threshold)	<input type="checkbox"/>	<input type="checkbox"/>

☐ Create separate output group for rejected records

다음 표에는 구성 설정이 설명되어 있습니다.

옵션	설정
하한 임계값	10
상한 임계값	90
양호한 레코드	표준 출력
잘못된 레코드	잘못된 레코드 테이블
거부된 레코드	-

잘못된 레코드 테이블 생성을 클릭하여 잘못된 레코드 및 문제 테이블을 작성합니다.

## 잘못된 레코드 예외 매핑 출력 예제

쓰기 변환을 매핑에 추가하고 이 데이터 개체에 표준 출력 포트를 연결합니다. 이 매핑에는 잘못된 레코드 데이터베이스 개체와 구성 보기에서 작성한 문제 데이터베이스 개체도 포함됩니다.

## 잘못된 레코드 테이블

잘못된 레코드 테이블에는 레코드 점수가 하한 임계값과 상한 임계값 사이인 예외가 포함됩니다.

다음 그림은 예외 변환이 반환하는 잘못된 레코드를 보여 줍니다.

Name: [excc\\_BadRecords.Bad\\_Output\\_DI](#)

	Workflow_ID	Row_Identifier	Record_Status	CUST_ID	COMPANY	CONTACT	TITLE	ADDR1	ADDR2
1	DummyWor ...	0	INVALID	7121657	WAITROSE	MRS LACI WINI...	DIRECTOR OF IS	380-394 N END...	FULHAM
2	DummyWor ...	1	INVALID	7121649	WAITROSE	MR NICHOLAS...	SENIOR PROJE...	EATON CNTR C...	EATON
3	DummyWor ...	2	INVALID	7121647	VARSHÉE SUPE...	MRS REUVEN C...	MANAGER	834 LONDON RD	THORNTON HE
4	DummyWor ...	3	INVALID	1002138	VICTORY SUPE...	MR RAY ARIAS	COMPUTER SPE...	22 FITCHBURG...	AYER
5	DummyWor ...	4	INVALID	1002137	VICTORY SUPE...	MR NEVILLE DE...	SR PROJ MANA...	21 TIMPANY BLV	GARDNER
6	DummyWor ...	5	INVALID	1002109	USDA FOREST...	MR MICHEAL F...	COMPUTER SPE...	1621 N KENT S...	ROSSLYN
7	DummyWor ...	6	INVALID	1002026	US ARMY CORP...	MR ROBERT EV...	PROGRAMMER...	20 MASS AVE NW	WASHINGTON
8	DummyWor ...	7	INVALID	1002062	US NAVY	MR JOHN WELCH	COMPUTER SPEC	DATA PROCESS...	OAK HARBOUR
9	DummyWor ...	8	INVALID	1062004	TRICOR INDUS...	MR MICHAEL G...	V.P.	8181 PROFESSI...	LANDOVER
10	DummyWor ...	9	INVALID	1001921	THE CORNER S...	MR DENIS LEE	MANAGER (\$\$...	102 MID STR S...	REDHILL
11	DummyWor ...	10	INVALID	7121217	THE CORNER S...	MRS TESSI SAN...	PROGRAMMER	192 BATTERSE...	BATTERSEA

잘못된 레코드 테이블에는 소스 레코드의 모든 필드가 포함됩니다. 잘못된 레코드에는 다음 필드도 포함됩니다.

#### Workflow\_ID

예외 변환이 포함된 워크플로우의 이름입니다. 이러한 워크플로우에는 예외 변환 매핑 태스크 및 문제 검토를 위한 휴먼 태스크가 포함됩니다. 워크플로우에 예외 변환이 없는 경우 Workflow\_ID에 DummyWorkflowID가 포함됩니다.

#### Row\_Identifier

각 행을 식별하는 고유 번호입니다.

#### Record\_Status

Analyst 도구에 대한 레코드 상태입니다. 잘못된 레코드 테이블의 각 레코드에는 Invalid 상태가 할당됩니다. Analyst 도구에서 레코드를 업데이트할 때 레코드 상태를 유지할 수 있습니다.

## 문제 테이블

문제 테이블에는 잘못된 레코드 테이블의 각 행에 대한 행이 1개 포함되어 있습니다. 각 행에는 데이터 품질 분석이 소스 레코드에 대해 발견한 문제가 포함되어 있습니다.

다음 그림에는 문제 테이블의 열이 표시되어 있습니다.

**Output**

Name: [excc\\_BadRecords.Issues\\_DI](#)

	Workflow_ID	Row_Identifier	COMPANY	DQAPRIORITY_...	ADDR2	DQAPRIORITY_...	ADDR3	DQAPRIORITY_...	COUNTRY	DQ...
1	DummyWor ...	0		1	invalid_locality	2		3		3
2	DummyWor ...	1		1	invalid_locality	2	invalid_province	3		3
3	DummyWor ...	2		1	invalid_locality	2		3		3
4	DummyWor ...	3		1		2		3		3
5	DummyWor ...	4		1		2		3		3
6	DummyWor ...	5		1	invalid_locality	2		3		3
7	DummyWor ...	6		1		2		3		3
8	DummyWor ...	7		1	invalid_locality	2		3		3
9	DummyWor ...	8		1	invalid_locality	2		3		3
10	DummyWor ...	9		1		2		3		3
11	DummyWor ...	10		1	invalid_locality	2		3		3

문제 테이블에는 다음 열이 포함되어 있습니다.

#### Workflow\_ID

레코드를 작성한 워크플로우를 식별합니다. 워크플로우에는 예외 변환 매핑 태스크와 문제점을 검토하는 휴먼 태스크가 포함되어 있습니다.

#### Row\_Identifier

데이터베이스 테이블에서 레코드 행을 식별합니다. 행 식별자는 문제 테이블의 행에 해당하는 잘못된 레코드 테이블의 행을 식별합니다.

#### 문제 필드 이름

필드 이름은 품질 문제가 될 수 있는 필드의 이름입니다. 필드에 오류가 포함된 경우 품질 문제 텍스트가 열 값이 됩니다. 위 그림에서 ADDR2 필드 이름에 invalid\_locality 품질 문제가 포함되어 있습니다.



## DQAPriority

문제 우선 순위입니다. 동일한 필드에 대해 여러 문제가 발생할 경우 가장 우선 순위가 높은 문제가 문제 필드 이름에 표시됩니다.

## 양호한 레코드 테이블

양호한 레코드 테이블의 각 레코드는 상한 임계값보다 높은 레코드 점수를 갖습니다. 이 예에서 상한 임계값은 90입니다.

다음 그림은 예외 변환이 반환하는 양호한 레코드를 보여 줍니다.

Output

Name: [exc\\_BadRecords.Output\\_DI](#)

	Score	CUST_ID	COMPANY	CONTACT	TITLE	ADDR1	ADDR2	ADDR3
1	100	7121669	YEOVALE STOR...	MRS OPPORTU...	OWNER/CONSULTANT	1 MARGROVE T...	BARNSTAPLE	DEVON
2	100	7121667	WHARFEDALE...	MRS PRAGER	GENERAL MANAGER	458 SOUTHCOA...	HULL	NORTH HI
3	100	1002195	WORLDSPAN	MR CHRISTOPH...	SENIOR ANALYST	N8J1-1 300 GA...	ATLANTA	GA
4	100	1002187	WINN DIXIE ST...	MS CORINA GA...	VICE PRESIDENT	1805 WAYNE M...	GOLDSBORO	NC
5	100	1002181	WINN DIXIE ST...	MR JENS GONY...	COMPUTER OPERATOR	506 W GANNO...	ZEBULON	NC
6	100	1002183	WINN DIXIE ST...	MS MERCIE VA...	MANAGER OF DBA	5715 GUNN HWY	TAMPA	FL
7	100	1000029	WINN DIXIE ST...	MR LYLE HICK...	MGR, DBA	3123 HWY 28 E	PINEVILLE	LA
8	100	1002178	WINN DIXIE ST...	MR HUSSEIN D...	AUTOMATED SYSTEMS MGR	2080 S FRONTA...	VICKSBURG	MS
9	100	1002177	WINN DIXIE ST...	MR REUVEN RH...	CHIEF DBA	695 S SEMORA...	ORLANDO	FL
10	100	1002180	WINN DIXIE ST...	MS GOLDI WEI...	MGR COORD	11957 5 APOPK...	ORLANDO	FL

양호한 레코드 테이블의 레코드에는 레코드 점수 및 소스 데이터 필드가 포함됩니다.

## 제 8 장

# 대/소문자 변환기 변환

이 장에 포함된 항목:

- [대/소문자 변환기 변환 개요, 158](#)
- [대/소문자 전략 속성, 158](#)
- [대/소문자 변환기 전략 구성, 159](#)
- [대/소문자 변환기 변환 고급 속성, 159](#)
- [대/소문자 변환기 변환 - 비원시 환경, 160](#)

## 대/소문자 변환기 변환 개요

대/소문자 변환기 변환은 입력 문자열에서 알파벳 문자의 대/소문자를 표준화하는 수동 변환입니다.

대문자, 소문자, 단어의 첫 글자를 대문자로 변환하거나 문장의 첫 글자를 대문자로 변환하는 등 대/소문자 변환 형식을 선택할 수 있습니다. 입력 데이터에서 각 문자의 현재 대/소문자를 반전할 수도 있습니다.

대/소문자 변환기 변환은 참조 테이블의 유효한 열에 있는 값을 사용하여 입력 문자의 대/소문자를 정의할 수 있습니다. 입력 값과 유효한 값 사이에 일치하는 항목이 발견되면 변환이 유효한 값의 대/소문자를 입력 값에 적용합니다. 참조 테이블은 대/소문자 변환 유형이 **단어의 첫 글자를 대문자로** 또는 **문장의 첫 글자를 대문자로**인 경우에 사용할 수 있습니다.

대/소문자 변환기 변환에서 여러 대/소문자 변환 전략을 생성할 수 있습니다. 각 전략은 하나의 변환 유형을 사용합니다.

## 대/소문자 전략 속성

대/소문자 변환 전략의 속성을 구성할 수 있습니다.

**전략** 보기에서 다음과 같은 대/소문자 변환 속성을 구성할 수 있습니다.

### 변환 유형

전략이 사용할 대/소문자 변환 방법을 정의합니다. 다음과 같은 대/소문자 변환 유형을 적용할 수 있습니다.

- **대문자.** 모든 글자를 대문자로 변환합니다.
- **문장의 첫 글자를 대문자로.** 필드 데이터 문자열의 첫 글자를 대문자로 변환합니다.
- **대/소문자 전환.** 소문자 글자를 대문자로 변환하고 대문자 글자를 소문자로 변환합니다.

- **단어의 첫 글자를 대문자로.** 각 하위 문자열의 첫 글자를 대문자로 변환합니다.
- **소문자.** 모든 글자를 소문자로 변환합니다.

기본 대/소문자 변환 방법은 대문자입니다.

#### 대문자 단어를 변경하지 않은 상태로 그대로 두기

대문자 문자열에 선택된 대문자 표시를 재정의합니다.

#### 구분자

단어의 첫 글자를 대문자로 변환할 때 대문자 표시가 적용되는 방식을 정의합니다. 예를 들어 "smith-jones"를 "Smith-Jones"로 변환하려면 대시를 구분자로 선택합니다. 기본 구분자는 공백 문자입니다.

#### 참조 테이블

참조 테이블에 지정된 대문자 표시 형식을 적용합니다. 대/소문자 변환 옵션이 **단어의 첫 글자를 대문자로** 또는 **문장의 첫 글자를 대문자로**인 경우에만 적용됩니다. 참조 테이블을 전략에 추가하려면 **새로 만들기**를 클릭합니다.

**참고:** 참조 테이블 일치가 토큰의 시작에서 발생하는 경우 토큰의 다음 문자가 대문자로 변경됩니다. 예를 들어 입력 문자열이 mcdonald이고 참조 테이블에 Mc에 대한 항목이 있는 경우 출력 문자열은 McDonald입니다.

## 대/소문자 변환기 전략 구성

입력 문자열의 대/소문자를 변경하려면 대/소문자 변환기 변환의 **전략** 보기에서 설정을 구성합니다.

1. **전략** 보기를 선택합니다.
2. **새로 만들기**를 클릭합니다.  
새 **전략** 마법사가 열립니다.
3. 필요에 따라 전략 이름과 설명을 편집합니다.
4. **입력** 및 **출력** 필드를 클릭하여 전략에 대한 포트를 선택합니다.
5. 전략 속성을 구성합니다. 기본 변환 전략은 **대문자**입니다.
6. **다음**을 클릭합니다.
7. 필요한 경우 참조 테이블을 추가하여 참조 테이블 항목과 일치하는 입력 데이터에 대한 대/소문자 옵션을 사용자 지정합니다. 참조 테이블 대/소문자 사용자 지정은 단어의 첫 글자를 대문자로 및 문장의 첫 글자를 대문자로 전략에만 적용됩니다.
8. **마침**을 클릭합니다.

## 대/소문자 변환기 변환 고급 속성

대/소문자 변환기 변환 데이터에 대한 데이터 통합 서비스의 처리 방식을 결정하는 데 도움이 되는 속성을 구성할 수 있습니다.

로그의 추적 수준을 구성할 수 있습니다.

**고급** 탭에서 다음 속성을 구성합니다.

### 추적 수준

이 변환에 대해 로그에 표시되는 세부 정보의 양입니다. 간단, 보통, 자세한 정보 표시 초기화 또는 자세한 정보 표시 데이터를 선택할 수 있습니다. 기본값은 보통입니다.

## 대/소문자 변환기 변환 - 비원시 환경

비원시 환경에서 대/소문자 변환기 변환의 처리는 변환을 실행하는 엔진에 따라 다릅니다.

다음과 같은 비원시 런타임 엔진에 대한 지원을 고려하십시오.

- **Blaze** 엔진. 제한 없이 지원됩니다.
- **Spark** 엔진. 제한 없이 지원됩니다.
- **Databricks Spark** 엔진. 지원되지 않습니다.

## 제 9 장

# 분류자 변환

이 장에 포함된 항목:

- [분류자 변환 개요, 161](#)
- [분류자 모델, 162](#)
- [분류자 알고리즘, 162](#)
- [분류자 변환 옵션, 162](#)
- [분류자 전략, 163](#)
- [분류자 변환의 고급 속성, 163](#)
- [분류자 전략 구성, 163](#)
- [분류자 분석 예제, 164](#)
- [분류자 변환 - 비원시 환경, 169](#)

## 분류자 변환 개요

분류자 변환은 입력 필드를 분석하고 각 필드의 정보 유형을 식별하는 수동 변환입니다. 입력 필드에 여러 텍스트 값이 포함된 경우 분류자 변환을 사용합니다.

분류자 변환을 구성하는 경우 분류자 모델 및 분류자 알고리즘을 선택합니다. 분류자 모델은 참조 데이터 개체의 유형입니다. 분류자 알고리즘은 문자열에서 유사한 단어 수 및 단어의 상대 위치를 계산하는 규칙 집합입니다. 변환이 알고리즘 분석과 분류자 모델의 콘텐츠를 비교합니다. 변환이 문자열의 지배적 정보 유형을 식별하는 모델 분류를 반환합니다.

분류자 변환은 상당한 길이의 문자열을 분석할 수 있습니다. 예를 들어 변환을 사용하여 이메일 메시지, 소셜 미디어 메시지 및 문서 텍스트의 콘텐츠를 분류할 수 있습니다. 각 문서 또는 메시지의 콘텐츠를 데이터 소스 열의 필드에 전달하고 열을 분류자 변환에 연결합니다. 각각의 경우 각 필드에 분석하려는 문자열 또는 문서의 전체 콘텐츠가 포함되도록 데이터 소스를 준비합니다.

## 분류자 모델

분류자 변환은 분류자 모델이라는 참조 데이터 개체를 사용하여 입력 데이터를 분석합니다. 분류자 변환을 구성할 때 분류자 모델을 선택합니다. 변환은 입력 데이터 열을 분류기 모델 데이터와 비교하고 각 입력 필드의 정보 유형을 설명하는 레이블을 반환합니다.

분류자 모델에는 참조 데이터 행 및 레이블 값이 포함됩니다. 행은 분류자 변환에 연결할 수 있는 포트의 입력 데이터를 나타냅니다. 레이블 값은 데이터 행에 포함되어 있는 정보의 유형을 설명합니다. 분류자 모델을 구성할 때 모델의 각 참조 데이터에 레이블을 할당합니다.

분류자 모델에서 참조 데이터 행을 레이블에 연결하려면 모델을 컴파일합니다. 컴파일 프로세스에서는 데이터 행과 레이블 값 간에 일련의 논리적 연관이 생성됩니다. 모델을 읽는 매핑을 실행하면 데이터 통합 서비스가 모델 논리를 분류기 변환 입력 데이터에 적용합니다. 데이터 통합 서비스는 각 입력 데이터 필드의 정보를 가장 정확히 설명하는 레이블을 반환합니다.

Developer tool에서 분류자 모델을 작성할 수 있습니다. 모델 리포지토리는 분류자 모델 개체를 저장합니다. Developer tool은 데이터 행, 레이블 및 컴파일 데이터를 Informatica 디렉터리 구조의 파일에 기록합니다.

## 분류자 알고리즘

변환 전략에 분류자 모델을 추가할 때는 분류자 알고리즘도 선택합니다. 분류자 알고리즘은 변환이 분류자 모델 데이터를 입력 데이터와 비교하는 방식을 결정합니다.

**Naive Bayes** 알고리즘 또는 **최대 엔트로피** 알고리즘을 선택할 수 있습니다.

알고리즘을 선택할 때 다음 요인을 고려하십시오.

- 최대 엔트로피 알고리즘은 **Naive Bayes** 알고리즘보다 정밀한 분석을 수행합니다.
- **Naive Bayes** 알고리즘을 사용하는 매핑은 동일한 데이터에서 최대 엔트로피 알고리즘을 사용하는 매핑보다 빠르게 실행됩니다.
- 코어 엑셀러레이터에 포함된 분류자 모델에는 최대 엔트로피 알고리즘을 선택하십시오.

## 분류자 변환 옵션

Developer tool에서 일련의 탭 또는 보기를 사용하여 분류자 변환의 옵션을 구성할 수 있습니다.

재사용 가능한 변환을 열면 변환 편집기의 일련의 탭에 옵션이 표시됩니다. 매핑에서 재사용할 수 없는 변환을 열면 매핑 편집기의 일련의 보기에 옵션이 표시됩니다. 매핑 속성 탭을 선택하면 재사용할 수 없는 변환의 보기를 볼 수 있습니다.

다음과 같은 보기를 선택할 수 있습니다.

### 일반

변환의 이름 및 설명을 보고 업데이트합니다.

### 포트

변환의 입력 및 출력 포트를 봅니다.

**참고:** 재사용 가능한 분류자 변환에서는 일반 보기 및 포트 보기가 **개요** 탭에 함께 표시됩니다.

### 전략

전략을 추가, 제거 또는 편집합니다.

### 종속성

각 전략의 입력 및 출력 포트를 봅니다.

### 고급

변환이 로그 파일에 기록하는 세부 정보의 수준을 설정합니다.

## 분류자 전략

변환이 입력 데이터에 대해 수행하는 데이터 분석 작업 집합이 전략입니다. 분류자 변환에서 하나 이상의 전략을 작성하십시오. 분류자 전략은 단일 입력 포트를 읽습니다.

전략에 하나 이상의 작업을 정의하십시오. 분류자 작업에서 입력 포트 데이터에 적용할 분류자 모델 및 분류자 알고리즘을 식별합니다. 각 작업에서 다른 출력 포트에 기록합니다. 다른 방법으로 입력 포트를 분석할 경우 전략에 여러 작업을 작성하십시오.

**참고:** 소스 데이터에 사용된 언어를 식별할 경우 분류자 작업에서 최대 엔트로피 알고리즘을 선택하십시오.

## 분류자 변환의 고급 속성

분류자 변환 데이터에 대한 데이터 통합 서비스의 처리 방식을 결정하는 데 도움이 되는 속성을 구성할 수 있습니다.

로그의 추적 수준을 구성할 수 있습니다.

**고급** 탭에서 다음 속성을 구성합니다.

### 추적 수준

이 변환에 대해 로그에 표시되는 세부 정보의 양입니다. 간단, 보통, 자세한 정보 표시 초기화 또는 자세한 정보 표시 데이터를 선택할 수 있습니다. 기본값은 보통입니다.

## 분류자 전략 구성

전략을 구성하여 데이터의 정보 유형을 식별할 수 있습니다. 각 전략은 입력 포트를 분석합니다.

재사용할 수 없는 변환에서는 전략을 구성하기 전에 입력 포트를 변환에 연결해야 합니다.

1. 변환을 열고 **전략** 보기를 선택합니다.
2. **새 전략**을 클릭합니다.  
전략 작성 마법사가 열립니다.
3. 전략 이름 및 선택적 설명을 입력합니다.
4. 입력 필드에서 입력 포트를 선택합니다.

5. 입력 포트의 전체 자릿수 값이 입력 포트의 모든 필드를 읽을 수 있을 만큼 많은지 확인합니다. 전체 자릿수 한도에 도달하면 포트가 입력 데이터를 잘라냅니다.
6. 점수 데이터를 전략 출력에 추가하는 옵션을 선택하거나 선택 취소합니다.
7. **다음**을 클릭합니다.
8. 분류자 작업 유형을 확인하고 **다음**을 클릭합니다.
9. 분류자 알고리즘을 선택합니다. 다음 알고리즘을 선택할 수 있습니다.
  - Naive Bayes
  - 최대 엔트로피

**참고:** 소스 데이터에 사용된 언어를 식별하려면 최대 엔트로피 알고리즘을 선택합니다.
10. 출력 포트를 확인합니다.
 

분류자 변환이 전략의 각 작업에 대한 단일 출력 포트를 작성합니다. 포트 이름 및 전체 자릿수를 편집할 수 있습니다.
11. 분류자 모델을 선택합니다.
 

모델 리포지토리의 분류자 모델 개체가 마법사에 나열됩니다.
12. **다음**을 클릭하여 다른 작업을 전략에 추가합니다. 또는 **마침**을 클릭합니다.

## 분류자 분석 예제

사용자가 새로운 스마트폰 응용 프로그램을 출시한 소프트웨어 회사의 데이터 스튜어드입니다. 회사는 응용 프로그램에 대한 대중의 반응과 이를 보도하는 매스컴에 대해 파악하고자 합니다. 회사에서 사용자와 사용자 팀에게 응용 프로그램에 대한 소셜 미디어 평가를 분석해달라고 요청합니다.

사용자는 스마트폰을 논의하는 트위터 피드의 데이터를 캡처하기로 결정합니다. 그래서 트위터 응용 프로그램 프로그래밍 인터페이스를 사용하여 트위터 데이터 스트림을 필터링합니다. 사용자는 분석할 트위터 데이터를 포함하는 데이터 소스를 작성합니다.

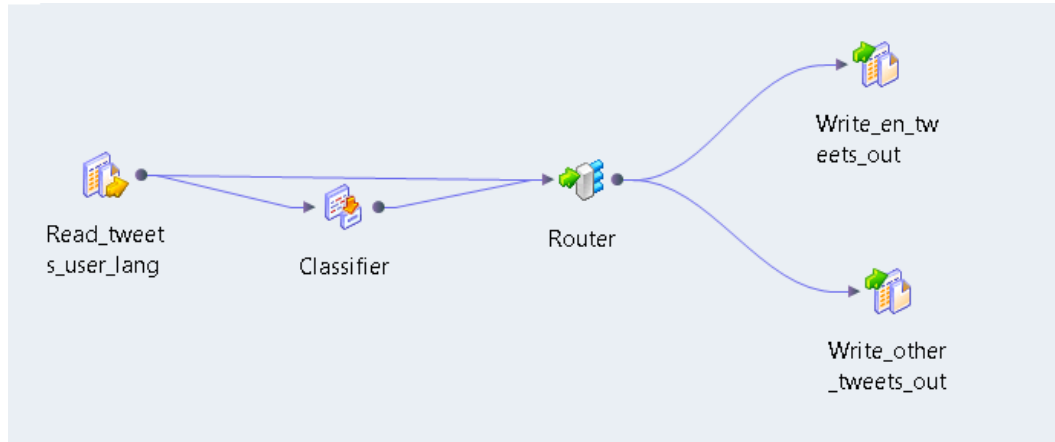
트위터 피드에는 여러 언어로 된 메시지가 포함되기 때문에 각 메시지에서 사용된 언어를 식별해야 합니다. 분류자 변환을 사용하여 언어를 분석하기로 결정합니다. 소스 데이터의 언어를 식별하는 매핑을 작성하고 트위터 메시지를 영어 및 비영어 데이터 대상에 기록하는 매핑을 작성합니다.



## 분류자 매핑 작성

데이터 소스를 읽고 데이터의 언어를 분류하고 포함된 언어에 따라 데이터를 대상에 기록하는 매핑을 작성합니다.

다음 이미지는 **Developer tool**의 매핑을 보여 줍니다.



작성하는 매핑에는 다음 개체가 포함됩니다.

개체 이름	설명
Read_tweet_user_lang	데이터 소스. Twitter 메시지가 포함됩니다.
분류자	분류자 변환. Twitter 메시지에 사용된 언어를 식별합니다.
라우터	라우터 변환. Twitter 메시지에 포함된 언어에 따라 메시지를 데이터 대상 개체로 라우팅합니다.
Write_en_tweets_out	데이터 대상. 영어 Twitter 메시지가 포함됩니다.
Write_other_tweets_out	데이터 대상. 영어가 아닌 Twitter 메시지가 포함됩니다.

## 입력 데이터 샘플

다음 데이터 조각은 맵핑에서 분석하는 **Twitter** 데이터의 샘플을 보여 줍니다.

### Twitter Message

```
RT @GanaphoneS3: Faltan 10 minutos para la gran rifa de un iPhone 5...
RT @Clarified: How to Downgrade Your iPhone 4 From iOS 6.x to iOS 5.x (Mac)...
RT @jerseyjazz: The razor was the iPhone of the early 2000s
RT @KrisiDevine: Apple Pie that I made for Thanksgiving. http://t.com/s9ImzFx0
RT @sophieHz: Dan yang punya 2 kupon undian. Masuk dalam kotak undian yang berhadiah Samsung
RT @IsabelFreitas: o galaxy tem isso isso isso e a bateria Á melhor que do iPhone
RT @PremiusIpad: Faltan 15 minutos para la gran rifa de un iPhone 5...
RT @payyton3: I want apple cider
RT @wiesteronder: Retweet als je iets van Apple, Nike, Adidas of microsoft hebt!
```

## 데이터 소스 구성

이 데이터 소스에는 단일 포트가 포함됩니다. 포트의 각 행에는 단일 **Twitter** 메시지가 포함됩니다.

다음 표에는 데이터 소스의 구성이 설명되어 있습니다.

포트 이름	포트 유형	전체 자릿수
텍스트	해당 없음	200

## 분류자 변환 구성

분류자 변환은 단일의 입력 포트와 출력 포트를 사용합니다. 분류자 변환의 입력 포트는 데이터 소스에서 텍스트 필드를 읽습니다. 출력 포트에는 텍스트 필드의 각 **Twitter** 메시지에서 식별된 언어가 포함됩니다. 분류자 변환은 ISO 국가 코드를 사용하여 언어를 식별합니다.

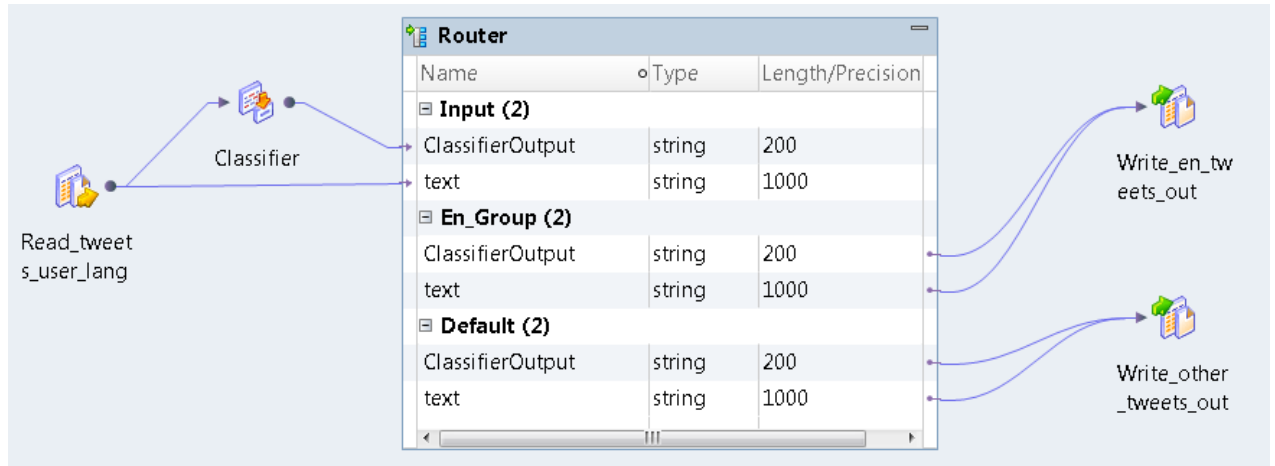
다음 표에는 분류자 변환의 구성이 설명되어 있습니다.

포트 이름	포트 유형	전체 자릿수	전략
text_input	입력	200	Classifier1
Classifier_Output	출력	2	Classifier1

## 라우터 변환 구성

라우터 변환은 2개의 입력 포트를 사용합니다. 이 변환은 데이터 소스에서 **Twitter** 메시지를 읽고 분류자 변환에서 **ISO** 국가 코드를 읽습니다. 라우터 변환은 사용자가 지정한 조건에 따라 입력 포트의 데이터를 서로 다른 출력 포트로 라우팅합니다.

다음 이미지는 라우터 변환의 포트 그룹 및 포트 연결을 보여 줍니다.



다음 표에는 라우터 변환의 구성이 설명되어 있습니다.

포트 이름	포트 유형	포트 그룹	전체 자릿수
Classifier_Output	입력	입력	2
텍스트	입력	입력	200
Classifier_Output	입력	기본값	2
텍스트	입력	기본값	200
Classifier_Output	입력	En_Group	2
텍스트	입력	En_Group	200

영어 메시지와 다른 언어의 메시지에 대한 데이터 스트림을 작성하도록 변환을 구성합니다. 데이터 스트림을 작성하려면 출력 포트 그룹을 변환에 추가합니다. 변환의 **그룹** 옵션을 사용하여 포트 그룹을 추가합니다.

변환이 각 데이터 스트림으로 데이터를 라우팅하는 방식을 결정하려면 포트 그룹에 조건을 정의해야 합니다. 조건은 포트를 식별하고 포트의 가능한 값을 지정합니다. 변환이 조건과 일치하는 입력 포트 값을 찾으면 입력 데이터를 포트 그룹으로 라우팅하고 포트 그룹이 조건을 적용합니다.

En\_Group에 다음 조건을 정의하십시오.

ClassifierOutput='en'

**참고:** 라우터 변환은 매핑의 두 개체로부터 데이터를 읽습니다. 라우터 변환은 데이터 개체에 정의된 행 시퀀스를 변경하지 않으므로 각 출력 그룹의 데이터를 결합할 수 있습니다.

## 데이터 대상 구성

매핑에는 영어 트위터 메시지에 대한 데이터 대상과 다른 언어로 된 메시지에 대한 대상이 포함되어 있습니다. 라우터 변환 출력 그룹의 포트를 데이터 대상에 연결하십시오.

다음 표에는 데이터 대상의 구성이 설명되어 있습니다.

포트 이름	포트 유형	전체 자릿수
텍스트	해당 없음	200
Classifier_Output	해당 없음	2

## 분류자 매핑 결과

매핑을 실행할 때 분류자 변환이 각 트위터 메시지의 언어를 식별합니다. 라우터 변환은 언어 분류에 따라 메시지 텍스트를 데이터 대상에 씁니다.

다음 데이터 조각은 영어 대상 데이터의 샘플을 보여줍니다.

ISO Country Code	Twitter Message
en	RT @Clarified: How to Downgrade Your iPhone 4 From iOS 6.x to iOS 5.x (Mac)...
en	RT @jerseyjazz: The razor was the iPhone of the early 2000s
en	RT @KrissiDevine: Apple Pie that I made for Thanksgiving. <a href="http://t.com/s9ImzFx0">http://t.com/s9ImzFx0</a>
en	RT @payyton3: I want apple cider

다음 데이터 조각은 다른 언어에 대해 식별된 대상 데이터의 샘플을 보여줍니다.

ISO Country Code	Twitter Message
es	RT @GanaphoneS3: Faltan 10 minutos para la gran rifa de un iPhone 5...
id	RT @sophieHz: Dan yang punya 2 kupon undian. Masuk dalam kotak undian yang berhadiah Samsung Champ.
pt	RT @IsabelFreitas: o galaxy tem isso isso isso e a bateria ã melhor que do iPhone
es	RT @PremiusIpad: Faltan 15 minutos para la gran rifa de un iPhone 5...
nl	RT @wiesteronder: Retweet als je iets van Apple, Nike, Adidas of microsoft hebt! <a href="http://t.co/Je6Ts00H">http://t.co/Je6Ts00H</a>

## 분류자 변환 - 비원시 환경

비원시 환경에서 분류자 변환의 처리는 변환을 실행하는 엔진에 따라 다릅니다.

다음과 같은 비원시 런타임 엔진에 대한 지원을 고려하십시오.

- **Blaze** 엔진. 제한 없이 지원됩니다.
- **Spark** 엔진. 제한 없이 지원됩니다.
- **Databricks Spark** 엔진. 지원되지 않습니다.

## 제 10 장

# 비교 변환

이 장에 포함된 항목:

- [비교 변환 개요, 170](#)
- [필드 일치 전략, 170](#)
- [ID 일치 전략, 173](#)
- [비교 전략 구성, 173](#)
- [비교 변환 고급 속성, 174](#)
- [비교 변환 - 비원시 환경, 174](#)

## 비교 변환 개요

비교 변환은 입력 문자열 쌍 간의 유사점을 평가하고 각 쌍의 유사점 정도를 숫자 점수로 계산하는 수동 변환입니다.

비교 변환을 구성할 때는 한 쌍의 입력 열을 선택하고 이 입력 열에 일치 전략을 할당해야 합니다.

비교 변환은 0~1 범위의 일치 점수를 출력합니다. 1은 완벽한 일치를 나타냅니다.

**참고:** 비교 변환에서 사용할 수 있는 전략은 일치 변환에서도 사용할 수 있습니다. 비교 변환은 일치 맵셋에 추가할 일치 비교 작업을 정의할 때 사용합니다. 여러 비교 변환을 맵셋에 추가할 수 있습니다. 일치 변환은 단일 변환의 일치 비교를 정의할 때 사용합니다. 일치 맵셋을 일치 변환에 포함할 수 있습니다.

## 필드 일치 전략

비교 변환에는 입력 데이터 필드 쌍을 비교하는 미리 정의된 필드 일치 전략이 포함되어 있습니다.

### Bigram

단일 필드에 입력된 우편 주소처럼 긴 텍스트 문자열을 비교하려면 **Bigram** 알고리즘을 사용하십시오.

**Bigram** 알고리즘은 두 문자열의 연속 문자 항목을 기반으로 두 데이터 문자열의 일치 점수를 계산합니다. 알고리즘에서 두 문자열에 공통된 연속 문자 쌍을 찾습니다. 두 문자열에서 일치하는 쌍 개수를 전체 문자 쌍 개수로 나눕니다.

## Bigram 예제

다음 문자열을 고려하십시오.

- larder
- lerder

이 문자열에서 다음 **Bigram** 그룹이 생성됩니다.

```
l a, a r, r d, d e, e r  
l e, e r, r d, d e, e r
```

"lerder" 문자열 내의 "e r" 문자열 두 번째 항목이 일치하지 않습니다. "larder" 문자열에서 "e r"의 해당하는 두 번째 항목이 없기 때문입니다.

**Bigram** 일치 점수를 계산하기 위해 변환에서 일치 쌍 개수(6)를 두 문자열의 전체 쌍 개수(10)로 나눕니다. 이 예에서 두 문자열은 60% 유사하며 일치 점수는 0.60입니다.

## Hamming 거리 측정법

**Hamming** 거리 측정법 알고리즘은 전화 번호, 우편 번호 또는 제품 코드 등의 숫자 또는 코드 필드와 같이 데이터 문자의 위치가 중요한 요인일 경우 사용합니다.

**Hamming** 거리 측정법 알고리즘은 두 데이터 문자열에서 문자가 다른 위치의 수를 계산하는 방법으로 일치 점수를 계산합니다. 문자열의 길이가 다를 경우 긴 문자열의 추가 문자는 다른 문자 수로 계산됩니다.

### Hamming 거리 측정법 예제

다음 문자열을 고려하십시오.

- Morlow
- Marlowes

강조 표시된 문자는 **Hamming** 알고리즘이 다른 문자로 식별하는 위치를 나타냅니다.

변환은 일치하는 문자의 수(5)를 가장 긴 문자열의 길이(8)로 나눠 **Hamming** 일치 점수를 계산합니다. 이 예에서 두 문자열은 62.5% 유사하며 일치 점수는 0.625입니다.

## 편집 거리 측정법

단어 또는 짧은 텍스트 문자열(예: 이름)을 비교하려면 편집 거리 측정법 알고리즘을 사용하십시오.

편집 거리 측정법 알고리즘에서 문자를 삽입, 삭제 또는 대체하여 하나의 문자열을 다른 문자열로 변환하는 최소 "비용"을 계산합니다.

### 편집 거리 측정법 예제

다음 문자열을 고려하십시오.

- Levenston
- Levenshtein

강조 표시된 문자는 문자열을 다른 문자열로 변환하는 데 필요한 작업을 표시합니다.

편집 거리 측정법 알고리즘에서 변경되지 않은 문자 수(8)를 가장 긴 문자열 길이(11)로 나눕니다. 이 예에서 두 문자열은 72.7% 유사하며 일치 점수는 0.727입니다.

## Jaro 거리 측정법

두 문자열을 비교할 때 비교 우선 순위가 첫 문자의 유사점인 경우 **Jaro** 거리 측정법 알고리즘을 사용합니다.

**Jaro** 거리 측정법의 일치 점수는 두 문자열의 첫 4개 문자가 갖는 유사점의 정도와 식별된 문자 전위의 수를 반영합니다. 첫 4개 문자 간의 일치에 대한 중요도는 페널티 속성에 입력한 값을 바탕으로 계산됩니다.

### Jaro 거리 측정법 속성

**Jaro** 거리 측정법 알고리즘을 구성할 때 다음 속성을 구성할 수 있습니다.

#### 페널티

두 문자열을 비교할 때 첫 4개 문자가 동일하지 않은 경우의 일치 점수 페널티를 결정합니다. 첫 번째 문자가 불일치하는 경우 변환이 전체 페널티 값을 뺍니다. 변환은 일치하지 않는 다른 문자의 위치에 따라 페널티의 일부를 뺍니다. 기본 페널티 값은 0.20입니다.

#### 대/소문자 구분

**Jaro** 거리 측정법 알고리즘으로 문자를 비교할 때 문자의 대/소문자를 고려할지 여부를 결정합니다.

### Jaro 거리 측정법 예제

다음 문자열을 고려하십시오.

- 391859
- 813995

기본 페널티 값인 0.20을 사용하여 이 문자열을 분석하는 경우 **Jaro** 거리 측정법 알고리즘은 0.513의 일치 점수를 반환합니다. 이 일치 점수는 해당 문자열이 51.3% 유사하다는 것을 나타냅니다.

## Hamming 거리 측정법 반전

**Hamming** 거리 측정법 반전 알고리즘을 사용하여 오른쪽에서 왼쪽으로 읽을 때 두 문자열 간의 문자 위치 차이 백분율을 계산합니다.

**Hamming** 거리 측정법 알고리즘은 두 데이터 문자열에서 문자가 다른 위치의 수를 계산하는 방법으로 일치 점수를 계산합니다. 길이가 다른 문자열의 경우 이 알고리즘은 가장 긴 문자열의 각 추가 문자를 문자열 간의 길이 차이로 계산합니다.

### Hamming 거리 측정법 반전 예제

왼쪽에서 오른쪽으로 쓰는 정렬을 사용하여 **Hamming** 측정법 반전 알고리즘을 시뮬레이트하는 다음 문자열을 살펴보십시오.

- 1-999-9999
- 011-01-999-9991

선택된 문자는 **Hamming** 거리 측정법 반전 알고리즘이 차이로 식별하는 위치를 나타냅니다.

**Hamming** 측정법 반전 일치 점수를 계산하기 위해 변환은 일치하는 문자의 수(9)를 가장 긴 문자열의 길이(15)로 나눕니다. 이 예제에서는 일치 점수가 0.6이므로 문자열 유사도가 60%인 것입니다.



## ID 일치 전략

비교 변환에는 개인, 주소 또는 회사 항목에 대한 일치를 찾는 데 사용할 수 있는 미리 정의된 ID 일치 전략이 포함되어 있습니다.

다음 표에는 각 ID 일치 전략에서 수행할 수 있는 일치 작업이 설명되어 있습니다.

ID 일치 전략	일치 작업
주소	주소 일치를 식별합니다.
연락처	조직 내에서 단일 위치의 연락처를 식별합니다.
회사 항목	법인 이름으로 조직을 식별합니다.
나누기	주소로 조직을 식별합니다.
가족	성과 주소 또는 전화 번호로 가족을 식별합니다.
필드	선택한 사용자 지정 필드를 식별합니다.
가정	동일한 거주지의 동일한 가족 구성원을 식별합니다.
개인	이름과 ID 또는 생일로 개인을 식별합니다.
조직	이름으로 조직을 식별합니다.
개인 이름	이름별로 개인을 식별합니다.
거주자	주소로 개인을 식별합니다.
광범위한 연락처	위치에 상관없이 조직 내 연락처를 식별합니다.
광범위한 가정	위치에 상관없이 동일한 가족 구성원을 식별합니다.

**참고:** ID 일치 전략에서는 **인구집단**이라는 참조 데이터 파일을 읽습니다. 시스템에 설치된 인구집단 데이터 파일에 대한 정보는 Informatica Administrator 사용자에게 문의하십시오.

## 비교 전략 구성

비교 전략을 구성하려면 비교 변환의 **전략** 보기에서 설정을 편집합니다.

1. **전략** 보기를 선택합니다.
2. **전략** 섹션에서 비교 전략을 선택합니다.
3. **필드** 섹션에서 **사용할 수 있는 필드** 열의 셀을 두 번 클릭하여 입력을 선택합니다.

**참고:** **입력 필드** 열에 입력 이름이 굵게 표시되는 각 행에 대해 입력을 선택해야 합니다.

## 비교 변환 고급 속성

데이터 통합 서비스가 비교 변환용 데이터를 처리하는 방법을 결정할 수 있게 도와주는 속성을 구성합니다.

로그의 추적 수준을 구성할 수 있습니다.

**고급** 탭에서 다음 속성을 구성합니다.

### 추적 수준

이 변환에 대해 로그에 표시되는 세부 정보의 양입니다. 간단, 보통, 자세한 정보 표시 초기화 또는 자세한 정보 표시 데이터를 선택할 수 있습니다. 기본값은 보통입니다.

## 비교 변환 - 비원시 환경

비원시 환경에서 비교 변환의 처리는 변환을 실행하는 엔진에 따라 다릅니다.

다음과 같은 비원시 런타임 엔진에 대한 지원을 고려하십시오.

- **Blaze** 엔진. 제한 없이 지원됩니다.
- **Spark** 엔진. 제한 없이 지원됩니다.
- **Databricks Spark** 엔진. 지원되지 않습니다.

## 제 11 장

# 통합 변환

이 장에 포함된 항목:

- [통합 변환 개요, 175](#)
- [통합 매핑, 176](#)
- [통합 변환 포트, 176](#)
- [통합 변환 보기, 176](#)
- [단순 전략, 179](#)
- [행 기반 전략, 179](#)
- [고급 전략, 180](#)
- [단순 통합 함수, 180](#)
- [행 기반 통합 함수, 184](#)
- [통합 매핑 예제, 188](#)
- [통합 변환 구성, 189](#)
- [통합 변환 - 비원시 환경, 189](#)

## 통합 변환 개요

통합 변환은 관련 레코드 그룹을 분석하고 각 그룹에 대한 통합된 레코드를 작성하는 활성 변환입니다. 키 생성기, 일치 및 연관 변환 같은 변환에서 생성한 레코드 그룹을 통합하려면 통합 변환을 사용하십시오.

통합 변환은 관련된 레코드 그룹에 전략을 적용하여 통합된 레코드를 생성합니다. 변환에는 어떤 레코드가 통합된 레코드인지 표시하는 출력 포트가 포함되어 있습니다. 통합된 레코드만 포함하도록 변환 출력을 제한하도록 선택할 수도 있습니다.

예를 들어, 일치 변환에서 생성하는 중복 직원 레코드 그룹을 통합할 수 있습니다. 통합 변환은 그룹의 모든 레코드에서 병합된 데이터를 포함하는 통합된 레코드를 작성할 수 있습니다.

통합 요구 사항에 따라 다른 유형의 전략을 사용하도록 통합 변환을 구성할 수 있습니다. 여러 레코드에서 통합된 레코드를 작성하려면 단순 전략을 사용하십시오. 단순 전략을 사용할 경우 각 포트마다 전략을 지정하십시오. 레코드 그룹의 행의 분석하고 한 행의 값을 가진 통합된 레코드를 작성하려면 행 기반 전략을 사용하십시오. 작성한 식을 적용하여 통합된 레코드를 작성하려면 고급 전략을 사용하십시오.

## 통합 매핑

레코드를 통합하려면 관련된 레코드 그룹을 작성하는 매핑을 작성하십시오. 매핑에 통합 변환을 추가하고 각 레코드 그룹을 단일 마스터 레코드로 통합하도록 변환을 구성합니다.

비즈니스 목표 및 데이터 요구 사항에 따라 통합 변환을 다른 변환에 연결합니다. 일치하는 레코드를 통합하기 위해 통합 변환을 일치 변환에 연결할 수 있습니다. 예외 레코드 관리의 일부로 레코드를 통합하려면 통합 변환을 예외 변환에 연결합니다. 키 생성기 변환을 사용하여 레코드를 그룹화하는 경우 통합 변환을 키 생성기 변환에 직접 연결할 수 있습니다. 통합 변환은 키 생성기 변환에서 작성한 각 그룹에 대해 통합된 레코드를 작성합니다.

### 원시 및 Hadoop 환경의 매핑 출력

원시 환경 및 Hadoop 환경에서 통합 매핑을 실행할 때 통합 변환은 서로 다른 결과를 생성할 수 있습니다. 매핑이 Hadoop의 여러 노드에서 실행되므로 입력 레코드는 원시 환경에서와는 다른 순서로 통합 변환을 입력할 수 있습니다. 그 결과, 변환은 동일한 입력 데이터 집합에 대해 각 환경에서 각기 다른 서바이버 레코드 집합을 생성할 수 있습니다. 변환 계산과 통합된 결과는 각 경우의 입력 행 순서에 대해 정확합니다.

원시 및 Hadoop 환경에서 동일한 서바이버 레코드를 생성하려면 다음과 같은 순서로 레코드가 정렬되도록 통합 변환을 구성합니다.

- 우선 그룹 기준 포트에서 레코드를 정렬합니다.
- 그런 다음, 입력 포트가 변환에 나타나는 순서대로 레코드를 정렬합니다.

## 통합 변환 포트

사용자가 추가한 각 입력 포트에 대한 출력 포트는 Developer tool에서 작성됩니다. 통합 변환에 수동으로 출력 포트를 추가할 수는 없습니다. 통합 변환에는 통합된 레코드를 나타내는 **IsSurvivor** 출력 포트도 포함됩니다.

통합 변환에 추가하는 입력 포트 중 하나에는 그룹 키가 포함되어야 합니다. 통합 전략이 전체 데이터 집합 대신 레코드 그룹을 처리하므로 통합 변환에 그룹 키 정보가 필요합니다.

입력 포트를 추가하면 Developer tool이 입력 포트 이름에 접미사 "1"을 추가하여 출력 포트 이름을 작성합니다. 통합 변환에는 레코드가 통합된 레코드인지 여부를 나타내는 **IsSurvivor** 출력 포트도 포함됩니다. 통합된 레코드인 경우 통합 변환이 **IsSurvivor** 포트에 문자열 "Y"를 기록합니다. 입력 레코드인 경우 통합 변환이 **IsSurvivor** 포트에 문자열 "N"을 기록합니다.

## 통합 변환 보기

통합 변환에는 포트, 전략 및 고급 속성에 대한 보기가 포함되어 있습니다.

### 통합 변환 전략 보기

전략 보기에는 단순, 행 기반 및 고급 전략에 대한 속성이 포함됩니다.

다음 목록에는 통합 전략의 유형이 설명되어 있습니다.

## 단순 전략

단순 전략은 레코드 그룹의 모든 포트 값을 분석하고 하나의 값을 선택합니다. 각 포트에 대한 단순 전략을 지정해야 합니다. 통합 변환은 모든 단순 전략에서 선택된 포트 값을 사용하여 통합된 레코드를 작성합니다. 단순 전략의 예에는 포트의 가장 자주 발생하는 값, 포트의 가장 긴 값 또는 포트의 공백을 제외하고 가장 자주 발생하는 값이 포함됩니다.

## 행 기반 전략

행 기반 전략은 레코드 그룹의 행을 분석하고 하나의 행을 선택합니다. 통합 변환은 이 행의 포트 값을 사용하여 통합된 레코드를 작성합니다. 행 기반 전략의 예에는 가장 많은 문자 수, 가장 작은 빈 필드 수 또는 가장 자주 발생하는 필드의 가장 많은 수가 포함됩니다.

## 고급 전략

고급 전략은 사용자가 정의한 전략을 사용하여 레코드 그룹을 분석합니다. 고급 전략은 통합 함수를 식에 사용하여 작성합니다. 통합 변환은 식의 출력을 바탕으로 통합된 레코드를 작성합니다. 작성 중인 이 식에 결합 변환에서 사용할 수 있는 모든 함수를 사용할 수 있습니다.

# 통합 변환의 고급 속성

통합 변환에는 정렬 동작, 출력 모드, 캐시 메모리 동작 및 추적 수준을 결정하는 고급 속성이 포함되어 있습니다. 다음과 같은 고급 속성을 구성할 수 있습니다.

## 정렬

변환이 **그룹 기준** 포트 데이터의 입력 행을 정렬할지 여부를 결정합니다. 이 속성은 기본적으로 활성화됩니다.

입력 행이 미리 정렬되어 있지 않은 경우 이 속성을 선택합니다.

## 대/소문자 구분 정렬

정렬 작업이 대/소문자를 구분하는지 여부를 결정합니다. 이 속성은 기본적으로 활성화됩니다.

## 출력 모드

변환이 모든 레코드를 출력으로 쓸지 아니면 통합된 레코드를 출력으로 쓸지 결정합니다. 기본값은 모두입니다.

## 캐시 파일 디렉터리

데이터 통합 서비스에서 현재 변환에 대해 임시 데이터를 쓸 디렉터리를 지정합니다. 입력 데이터의 양이 사용 가능한 시스템 메모리보다 클 경우 데이터 통합 서비스가 이 디렉터리에 임시 파일을 기록합니다. 데이터 통합 서비스는 매핑을 실행한 후에 임시 파일을 삭제합니다.

속성에 디렉터리 경로를 입력하거나 매개 변수를 사용하여 디렉터리를 식별할 수 있습니다. 데이터 통합 서비스 시스템의 로컬 경로를 지정하십시오. 데이터 통합 서비스가 기록할 수 있는 디렉터리를 지정해야 합니다. 기본값은 **CacheDir** 시스템 매개 변수입니다.

## 캐시 파일 크기

변환의 입력 데이터를 정렬하기 위해 데이터 통합 서비스에서 사용하는 시스템 메모리의 양을 결정합니다. 기본값은 400,000바이트입니다. 매개 변수를 사용하여 캐시 파일 크기를 지정할 수 있습니다.

데이터 통합 서비스는 데이터를 정렬하기 전에 사용자가 지정한 메모리 양을 할당합니다. 정렬 작업이 더 많은 데이터를 생성할 경우 데이터 통합 서비스는 초과 데이터를 캐시 파일 디렉터리에 기록합니다. 정렬 작업에 시스템 메모리 및 파일 저장소가 제공할 수 있는 것보다 더 많은 메모리가 필요할 경우 매핑이 실패합니다.

**참고:** 65536 이상의 값을 입력한 경우 일지 변환이 값을 바이트로 읽습니다. 이보다 낮은 값을 입력하면 일지 변환이 값을 메가바이트로 읽습니다.

## 추적 수준

이 변환에 대해 로그에 표시되는 세부 정보의 양입니다. 간단, 보통, 자세한 정보 표시 초기화 또는 자세한 정보 표시 데이터를 선택할 수 있습니다. 기본값은 보통입니다.

## 캐시 파일 크기

캐시 파일 크기 속성은 데이터 통합 서비스에서 정렬 작업을 위해 통합 변환에 할당하는 시스템 메모리의 양을 결정합니다. 이 속성은 데이터 통합 서비스 호스트 시스템의 **RAM**보다 작거나 같은 값으로 구성합니다.

최상의 성능을 얻으려면 최소 **16MB**의 캐시 파일 크기를 지정합니다.

데이터 통합 서비스에서는 정렬 작업을 시작하기 전에 캐시 파일 크기 속성에 지정된 메모리를 할당합니다. 데이터 통합 서비스에서는 모든 입력 데이터를 통합 변환에 전달한 다음 정렬 작업을 수행합니다.

입력 데이터의 양이 캐시 파일 크기보다 클 경우 데이터 통합 서비스에서 데이터를 캐시 파일 디렉터리에 기록합니다. 데이터를 캐시 파일 디렉터리에 쓸 경우 입력 데이터 양의 최소 2배에 해당하는 디스크 공간이 사용됩니다.

수신 데이터의 크기는 다음 수식을 사용하여 확인할 수 있습니다.

$$[\text{number\_of\_input\_rows} * (\text{Sum}(\text{column\_size}) + 16)]$$

다음 테이블에는 가능한 데이터 유형과 캐시 파일 데이터 계산에 적용되는 열 크기 값이 나열되어 있습니다.

데이터 유형	열 크기
이진	전체 자릿수 + 8 가까운 8의 배수로 반올림
날짜/시간	29
10진수, 많은 전체 자릿수 꺼짐(모든 전체 자릿수)	16
10진수, 많은 전체 자릿수 켜짐(전체 자릿수 <= 18)	24
10진수, 많은 전체 자릿수 켜짐(전체 자릿수 >18, <=28)	32
10진수, 많은 전체 자릿수 켜짐(전체 자릿수 > 28)	16
10진수, 많은 전체 자릿수 켜짐(음의 배율)	16
배정밀도	16
실수	16
정수	16
문자열, 텍스트	유니코드 모드: 2*(전체 자릿수 + 5) ASCII 모드: 전체 자릿수 + 9

## 단순 전략

단순 전략은 레코드 그룹의 포트를 분석하고 하나의 값을 반환합니다. 각 포트에 대한 단순 전략을 지정해야 합니다. 통합 변환은 모든 단순 전략에서 선택된 포트 값을 사용하여 통합된 레코드를 작성합니다.

변환의 **전략** 보기에서 전략을 구성하면 다음 텍스트가 전략의 통합 방법으로 표시됩니다.

*기본값 사용.*

기본 전략은 "값이 가장 큰 행 ID"입니다.

다음과 같은 단순 전략을 선택할 수 있습니다.

### 평균

레코드 그룹의 포트를 분석하고 전체 값의 평균을 반환합니다.

데이터 유형이 문자열 및 날짜/시간인 경우 가장 자주 발생하는 값이 반환됩니다.

### 최장

레코드 그룹의 포트를 분석하고 문자 수가 가장 많은 값을 반환합니다. 문자 수가 가장 많은 포트 값이 2개 이상인 경우 가장 먼저 조건을 충족한 값이 반환됩니다.

### 최대

레코드 그룹의 포트를 분석하고 가장 큰 값을 반환합니다.

데이터 유형이 문자열인 경우 가장 긴 문자열이 반환됩니다. 데이터 유형이 날짜/시간인 경우 가장 최근 날짜가 반환됩니다.

### 최소

레코드 그룹의 포트를 분석하고 가장 작은 값을 반환합니다.

데이터 유형이 문자열인 경우 가장 짧은 문자열이 반환됩니다. 데이터 유형이 날짜/시간인 경우 가장 이른 날짜가 반환됩니다.

### 가장 자주

레코드 그룹의 포트를 분석하고 가장 자주 발생하는 값을 반환합니다. 빈 값 또는 Null 값이 포함됩니다. 발생 수가 가장 큰 값이 2개 이상인 경우 가장 먼저 조건을 충족한 값이 반환됩니다.

### 가장 자주 나오고 비어 있지 않음

레코드 그룹의 포트를 분석하고 가장 자주 발생하는 값을 반환합니다. 빈 값 또는 Null 값은 제외됩니다. 빈 값을 제외하고 발생 수가 가장 큰 값이 2개 이상인 경우 가장 먼저 조건을 충족한 값이 반환됩니다.

### 최단

레코드 그룹의 포트를 분석하고 문자 수가 가장 작은 값을 반환합니다. 문자 수가 가장 작은 포트 값이 2개 이상인 경우 가장 먼저 조건을 충족한 값이 반환됩니다.

### 값이 가장 큰 행 ID

레코드 그룹의 포트를 분석하고 행 ID가 가장 큰 값을 반환합니다.

## 행 기반 전략

행 기반 전략은 레코드 그룹의 행을 분석하고 하나의 행을 선택합니다. 통합 변환은 이 행의 포트 값을 사용하여 통합된 레코드를 작성합니다. 기본 전략은 "대부분의 데이터"입니다.

다음 행 기반 전략 중에서 하나를 선택합니다.

### 대부분의 데이터

빈 문자 수가 가장 많은 행을 선택합니다. 문자 수가 가장 많은 행이 2개 이상인 경우 마지막으로 조건을 충족한 값이 반환됩니다.

### 대부분의 채움

빈 열을 제외하고 열 수가 가장 많은 행을 선택합니다. 빈 열을 제외하고 열 수가 가장 많은 행이 2개 이상인 경우 마지막으로 조건을 충족한 값이 반환됩니다.

### 모달 정확한 일치

빈 값을 제외하고 가장 자주 발생하는 값의 수가 가장 많은 행을 선택합니다. 예를 들어 레코드 그룹에서 가장 자주 발생하는 값이 포함된 포트 3개가 한 행에 있습니다. 이 행에서 가장 자주 발생하는 값의 수는 "3"입니다.

빈 값을 제외하고 가장 자주 발생하는 값의 수가 가장 많은 행이 2개 이상인 경우 마지막으로 조건을 충족한 값이 반환됩니다.

### 행 기반 전략 예제

다음 표에는 샘플 레코드 그룹이 표시되어 있습니다. 마지막 열에는 특정 행 기반 전략이 이 레코드 그룹에서 서로 다른 행을 선택하는 이유가 설명되어 있습니다.

제품 ID	이름	성	우편 번호	전략 선택
2106	Bartholomew		28516	대부분의 데이터 전략은 다른 행보다 이 행에 더 많은 문자가 포함되므로 이 행을 선택합니다.
2236	Bart	Smith	28579	대부분의 채움 전략은 다른 행보다 이 행에 빈 열이 더 많으므로 이 행을 선택합니다.
2236	<비어 있음>Smith		28516	모달 정확한 일치 전략은 가장 자주 발생하는 값의 수가 이 행에 가장 많이 포함되므로 이 행을 선택합니다.

## 고급 전략

고급 전략을 사용하면 미리 정의된 함수로부터 통합 전략을 작성할 수 있습니다. 통합 함수 및 기타 Informatica 함수를 사용할 수 있습니다.

단순 통합 함수 또는 행 기반 통합 함수를 포함하는 식을 작성할 수 있습니다. 단순 통합 함수는 레코드 그룹의 포트 값을 바탕으로 통합된 레코드를 작성할 때 사용합니다. 행 기반 통합 함수는 레코드 그룹에서 행을 선택할 때 사용합니다.

통합 식은 통합 변환의 모든 출력 포트를 채워야 합니다. 통합 식이 모든 출력 포트를 사용하지 않을 경우 변환이 매핑 실패를 야기합니다.

단순 또는 행 기반 전략을 고급 전략의 템플릿으로 사용할 수 있습니다. 단순 또는 행 기반 전략을 구성한 다음 고급을 선택합니다. 통합 변환이 전략을 수행하는 함수가 포함된 식을 생성합니다. 더 많은 함수를 추가하여 추가 요구 사항을 구현할 수 있습니다.

## 단순 통합 함수

단순 통합 함수는 레코드 그룹의 모든 포트 값에서 값을 선택합니다. 단순 통합 함수를 사용하는 경우 함수를 포트 및 그룹 기준 포트에 제공합니다.



## CONSOL\_AVG

레코드 그룹의 포트를 분석하고 전체 값의 평균을 반환합니다.

### 구문

`CONSOL_AVG(string, group by)`

다음 테이블에는 이 명령의 인수가 설명되어 있습니다.

인수	필수/선택	설명
문자열	필수	입력 포트 이름입니다.
그룹 기준	필수	그룹 식별자가 포함된 입력 포트의 이름입니다.

### 반환 값

포트에 있는 모든 값의 평균입니다.

문자열 및 날짜/시간 데이터 유형의 경우 함수에서 가장 자주 발생하는 값을 반환합니다.

### 예제

다음 식에서는 CONSOL\_AVG 함수를 사용하여 SalesTotal 입력 포트의 평균 값을 찾습니다.

`SalesTotal1:= CONSOL_AVG(SalesTotal, GroupKey)`

이 식에서는 CONSOL\_AVG 함수가 GroupKey 포트를 사용하여 레코드 그룹을 식별합니다. 레코드 그룹 내에서 함수가 SalesTotal 포트를 분석하고 평균 값을 반환합니다. 식에서 평균 값을 SalesTotal1 출력 포트에 기록합니다.

## CONSOL\_LONGEST

레코드 그룹의 포트를 분석하고 문자 수가 가장 많은 값을 반환합니다.

### 구문

`CONSOL_LONGEST(string, group by)`

다음 테이블에는 이 명령의 인수가 설명되어 있습니다.

인수	필수/선택	설명
문자열	필수	입력 포트 이름입니다.
그룹 기준	필수	그룹 식별자가 포함된 입력 포트의 이름입니다.

### 반환 값

문자 수가 가장 많은 포트 값이 반환됩니다.

문자 수가 가장 많은 포트 값이 2개 이상인 경우 가장 먼저 조건을 충족한 값이 반환됩니다.

### 예제

다음 식은 CONSOL\_LONGEST 함수를 사용하여 FirstName 입력 포트를 분석하고 문자 수가 가장 많은 값을 찾습니다.

`FirstName1:= CONSOL_LONGEST(FirstName, GroupKey)`

이 식에서 CONSOL\_LONGEST 함수는 GroupKey 포트를 사용하여 레코드 그룹을 식별합니다. 그런 다음 해당 레코드 그룹에서 FirstName 포트를 분석하고 가장 긴 값을 반환합니다. 이 값은 FirstName1 출력 포트에 기록됩니다.

## CONSOL\_MAX

레코드 그룹의 포트를 분석하고 가장 큰 값을 반환합니다.

### 구문

```
CONSOL_MAX(string, group by)
```

다음 테이블에는 이 명령의 인수가 설명되어 있습니다.

인수	필수/선택	설명
문자열	필수	입력 포트 이름입니다.
그룹 기준	필수	그룹 식별자가 포함된 입력 포트의 이름입니다.

### 반환 값

가장 큰 포트 값이 반환됩니다.

데이터 유형이 문자열인 경우 가장 긴 문자열이 반환됩니다. 데이터 유형이 날짜/시간인 경우 가장 최근 날짜가 반환됩니다.

### 예제

다음 식은 CONSOL\_MAX 함수를 사용하여 SalesTotal 입력 포트를 분석하고 가장 큰 값을 찾습니다.

```
SalesTotal1:= CONSOL_MAX(SalesTotal, GroupKey)
```

이 식에서 CONSOL\_MAX 함수는 GroupKey 포트를 사용하여 레코드 그룹을 식별합니다. 그런 다음 해당 레코드 그룹에서 SalesTotal 포트를 분석하고 가장 큰 값을 반환합니다. 이 값은 SalesTotal1 출력 포트에 기록됩니다.

## CONSOL\_MIN

레코드 그룹의 포트를 분석하고 가장 작은 값을 반환합니다.

### 구문

```
CONSOL_MIN(string, group by)
```

다음 테이블에는 이 명령의 인수가 설명되어 있습니다.

인수	필수/선택	설명
문자열	필수	입력 포트 이름입니다.
그룹 기준	필수	그룹 식별자가 포함된 입력 포트의 이름입니다.

### 반환 값

가장 작은 포트 값이 반환됩니다.

데이터 유형이 문자열인 경우 가장 짧은 문자열이 반환됩니다. 데이터 유형이 날짜/시간인 경우 가장 이른 날짜가 반환됩니다.

### 예제

다음 식은 CONSOL\_MIN 함수를 사용하여 SalesTotal 입력 포트를 분석하고 가장 작은 값을 찾습니다.

```
SalesTotal1:= CONSOL_MIN(SalesTotal, GroupKey)
```

이 식에서 **CONSOL\_MIN** 함수는 **GroupKey** 포트를 사용하여 레코드 그룹을 식별합니다. 그런 다음 해당 레코드 그룹에서 **SalesTotal** 포트를 분석하고 가장 작은 값을 반환합니다. 이 값은 **SalesTotal1** 출력 포트에 기록됩니다.

## CONSOL\_MOSTFREQ

레코드 그룹의 포트를 분석하고 가장 자주 발생하는 값을 반환합니다. 빈 값 또는 **Null** 값이 포함됩니다.

### 구문

**CONSOL\_MOSTFREQ**(*string*, *group by*)

다음 테이블에는 이 명령의 인수가 설명되어 있습니다.

인수	필수/선택	설명
<i>문자열</i>	필수	입력 포트 이름입니다.
<i>그룹 기준</i>	필수	그룹 식별자가 포함된 입력 포트의 이름입니다.

### 반환 값

빈 값 또는 **Null** 값을 포함하여 가장 자주 발생하는 값이 반환됩니다.

발생 수가 가장 큰 값이 2개 이상인 경우 가장 먼저 조건을 충족한 값이 반환됩니다.

### 예제

다음 식은 **CONSOL\_MOSTFREQ** 함수를 사용하여 **Company** 입력 포트를 분석하고 가장 자주 발생하는 값을 찾습니다.

**Company1** := **CONSOL\_MOSTFREQ**(**Company**, **GroupKey**)

이 식에서 **CONSOL\_MOSTFREQ** 함수는 **GroupKey** 포트를 사용하여 레코드 그룹을 식별합니다. 그런 다음 해당 레코드 그룹에서 **Company** 포트를 분석하고 가장 자주 발생하는 값을 반환합니다. 이 값은 **Company1** 출력 포트에 기록됩니다.

## CONSOL\_MOSTFREQ\_NB

레코드 그룹의 포트를 분석하고 가장 자주 발생하는 값을 반환합니다. 빈 값 또는 **Null** 값은 제외됩니다.

### 구문

**CONSOL\_MOSTFREQ\_NB**(*string*, *group by*)

다음 테이블에는 이 명령의 인수가 설명되어 있습니다.

인수	필수/선택	설명
<i>문자열</i>	필수	입력 포트 이름입니다.
<i>그룹 기준</i>	필수	그룹 식별자가 포함된 입력 포트의 이름입니다.

### 반환 값

공백 및 **Null** 값을 제외하고 가장 자주 발생하는 값입니다.

발생 수가 가장 큰 값이 2개 이상인 경우 가장 먼저 조건을 충족한 값이 반환됩니다.

## 예제

다음 식에서는 CONSOL\_MOSTFREQ\_NB 함수를 사용하여 회사 입력 포트를 분석하고 가장 자주 발생하는 값을 찾습니다.

```
Company1:= CONSOL_MOSTFREQ_NB(Company, GroupKey)
```

이 식에서는 CONSOL\_MOSTFREQ\_NB 함수가 GroupKey 포트를 사용하여 레코드 그룹을 식별합니다. 그런 다음 해당 레코드 그룹에서 Company 포트를 분석하고 가장 자주 발생하는 값을 반환합니다. 이 값은 Company1 출력 포트에 기록됩니다.

## CONSOL\_SHORTEST

레코드 그룹의 포트를 분석하고 문자 수가 가장 작은 값을 반환합니다.

### 구문

```
CONSOL_SHORTEST(string, group by)
```

다음 테이블에는 이 명령의 인수가 설명되어 있습니다.

인수	필수/선택	설명
문자열	필수	입력 포트 이름입니다.
그룹 기준	필수	그룹 식별자가 포함된 입력 포트의 이름입니다.

### 반환 값

문자 수가 가장 작은 포트 값이 반환됩니다.

문자 수가 가장 작은 포트 값이 2개 이상인 경우 가장 먼저 조건을 충족한 값이 반환됩니다.

## 예제

다음 식은 CONSOL\_SHORTEST 함수를 사용하여 FirstName 입력 포트를 분석하고 문자 수가 가장 작은 값을 찾습니다.

```
FirstName1:= CONSOL_SHORTEST(FirstName, GroupKey)
```

이 식에서 CONSOL\_SHORTEST 함수는 GroupKey 포트를 사용하여 레코드 그룹을 식별합니다. 그런 다음 해당 레코드 그룹에서 FirstName 포트를 분석하고 가장 짧은 값을 반환합니다. 이 값은 FirstName1 출력 포트에 기록됩니다.

## 행 기반 통합 함수

행 기반 통합 함수를 사용하여 레코드 그룹에서 레코드를 선택할 수 있습니다. 행 기반 통합 함수는 IF-THEN-ELSE 문에서 사용해야 합니다.

## CONSOL\_GETROWFIELD

행 기반 통합 함수에서 식별한 행을 읽고 지정한 포트에 대한 값을 반환합니다. 숫자 인수를 사용하여 포트를 지정하십시오.

다음 행 기반 통합 함수 중 하나와 함께 CONSOL\_GETROWFIELD 함수를 사용해야 합니다.

- CONSOL\_MODEXACT
- CONSOL\_MOSTDATA
- CONSOL\_MOSTFILLED

행 기반 통합 함수의 각 입력 포트에 대해 CONSOL\_GETROWFIELD 함수의 인스턴스 하나를 사용해야 합니다.

### 구문

CONSOL\_GETROWFIELD(*value*)

다음 테이블에는 이 명령의 인수가 설명되어 있습니다.

인수	필수/선택	설명
값	필수	행 기반 통합 함수의 입력 포트를 나타내는 숫자입니다. 함수에서 가장 왼쪽에 있는 포트를 지정하려면 "0"을 사용하십시오. 다른 포트를 표시하려면 후속 번호를 사용하십시오.

### 반환 값

지정한 포트의 값입니다. 행 기반 통합 함수에서 식별한 행에서 함수가 이 값을 읽습니다.

### 예제

다음 식에서는 CONSOL\_GETROWFIELD 함수를 CONSOL\_MOSTDATA 함수와 함께 사용합니다.

```
IF (CONSOL_MOSTDATA(First_Name,Last_Name,GroupKey,GroupKey))
THEN
First_Name1 := CONSOL_GETROWFIELD(0)
Last_Name1 := CONSOL_GETROWFIELD(1)
GroupKey1 := CONSOL_GETROWFIELD(2)
ELSE
First_Name1 := First_Name
Last_Name1 := Last_Name
GroupKey1 := GroupKey
ENDIF
```

이 식에서 CONSOL\_MOSTDATA 함수는 레코드 그룹의 행을 분석하고 단일 행을 식별합니다. CONSOL\_GETROWFIELD 함수는 연속 번호를 사용하여 해당 행의 포트 값을 읽고 이 값을 출력 포트에 기록합니다.

## CONSOL\_MODEXACT

가장 자주 발생하는 값의 수가 가장 많은 행을 식별합니다.

예를 들어 레코드 그룹에서 가장 자주 발생하는 값이 포함된 포트 3개가 한 행에 있습니다. 이 행에서 가장 자주 발생하는 값의 수는 "3"입니다.

이 함수는 CONSOL\_GETROWFIELD 함수와 함께 사용해야 합니다. CONSOL\_GETROWFIELD는 CONSOL\_MODEXACT 함수가 식별하는 행의 값을 반환합니다.

### 구문

CONSOL\_MODEXACT(*string1*, [*string2*, ..., *stringN*,]  
*group by*)

다음 테이블에는 이 명령의 인수가 설명되어 있습니다.

인수	필수/선택	설명
문자열	필수	입력 포트 이름입니다.
그룹 기준	필수	그룹 식별자가 포함된 입력 포트의 이름입니다.

## 반환 값

가장 자주 발생하는 필드의 수가 가장 많은 행의 경우 TRUE가 반환되고 다른 모든 행에 대해서는 FALSE가 반환됩니다.

## 예제

다음 식은 CONSOL\_MODELEXACT 함수를 사용하여 가장 자주 발생하는 필드의 수가 가장 많은 행을 찾습니다.

```
IF (CONSOL_MODELEXACT(First_Name,Last_Name,GroupKey,GroupKey))
THEN
First_Name1 := CONSOL_GETROWFIELD(0)
Last_Name1 := CONSOL_GETROWFIELD(1)
GroupKey1 := CONSOL_GETROWFIELD(2)
ELSE
First_Name1 := First_Name
Last_Name1 := Last_Name
GroupKey1 := GroupKey
ENDIF
```

이 식에서 CONSOL\_MODELEXACT 함수는 레코드 그룹의 행을 분석하여 행 하나를 식별합니다. CONSOL\_GETROWFIELD 함수는 연속 번호를 사용하여 해당 행의 포트 값을 읽고 이 값을 출력 포트에 기록합니다.

# CONSOL\_MOSTDATA

모든 포트에서 가장 많은 문자를 포함하는 행을 식별합니다.

이 함수는 CONSOL\_GETROWFIELD 함수와 함께 사용해야 합니다. CONSOL\_GETROWFIELD에서 CONSOL\_MOSTDATA 함수가 식별하는 행의 값을 반환합니다.

## 구문

```
CONSOL_MOSTDATA(string1, [string2, ..., stringN,]
group by)
```

다음 테이블에는 이 명령의 인수가 설명되어 있습니다.

인수	필수/선택	설명
문자열	필수	입력 포트 이름입니다.
그룹 기준	필수	그룹 식별자가 포함된 입력 포트의 이름입니다.

## 반환 값

모든 포트에서 가장 많은 문자를 포함하는 행의 경우에는 TRUE, 다른 모든 행의 경우에는 FALSE입니다.

## 예제

다음 식에서는 CONSOL\_MOSTDATA 함수를 사용하여 가장 많은 문자를 포함하는 행을 찾습니다.

```
IF (CONSOL_MOSTDATA(First_Name,Last_Name,GroupKey,GroupKey))
THEN
First_Name1 := CONSOL_GETROWFIELD(0)
```

```

Last_Name1 := CONSOL_GETROWFIELD(1)
GroupKey1 := CONSOL_GETROWFIELD(2)
ELSE
First_Name1 := First_Name
Last_Name1 := Last_Name
GroupKey1 := GroupKey
ENDIF

```

이 식에서 CONSOL\_MOSTDATA 함수는 레코드 그룹의 행을 분석하고 단일 행을 식별합니다. CONSOL\_GETROWFIELD 함수는 연속 번호를 사용하여 해당 행의 포트 값을 읽고 이 값을 출력 포트에 기록합니다.

## CONSOL\_MOSTFILLED

빈 필드를 제외하고 필드 수가 가장 많은 행을 식별합니다.

이 함수는 CONSOL\_GETROWFIELD 함수와 함께 사용해야 합니다. CONSOL\_GETROWFIELD는 CONSOL\_MOSTFILLED 함수가 식별하는 행의 값을 반환합니다.

### 구문

```

CONSOL_MOSTFILLED(string1, [string2, ..., stringN],
group by)

```

다음 테이블에는 이 명령의 인수가 설명되어 있습니다.

인수	필수/선택	설명
문자열	필수	입력 포트 이름입니다.
그룹 기준	필수	그룹 식별자가 포함된 입력 포트의 이름입니다.

### 반환 값

빈 필드를 제외하고 필드 수가 가장 많은 행의 경우 TRUE가 반환되고 다른 모든 행에 대해서는 FALSE가 반환됩니다.

### 예제

다음 식은 CONSOL\_MOSTFILLED 함수를 사용하여 문자 수가 가장 많은 행을 찾습니다.

```

IF (CONSOL_MOSTFILLED(First_Name,Last_Name,GroupKey,GroupKey))
THEN
First_Name1 := CONSOL_GETROWFIELD(0)
Last_Name1 := CONSOL_GETROWFIELD(1)
GroupKey1 := CONSOL_GETROWFIELD(2)
ELSE
First_Name1 := First_Name
Last_Name1 := Last_Name
GroupKey1 := GroupKey
ENDIF

```

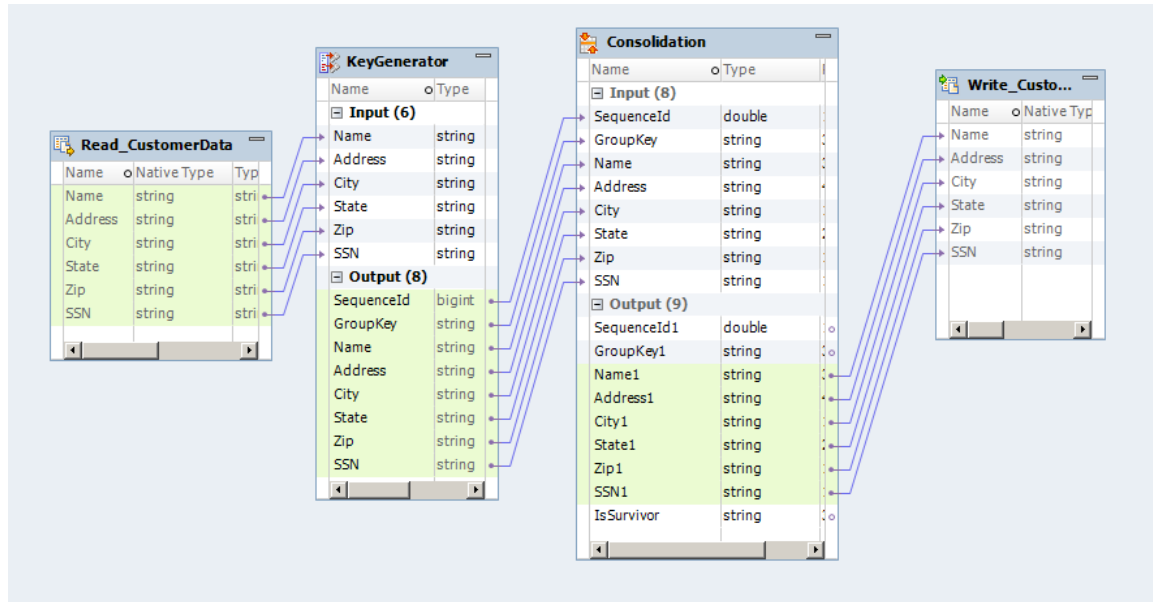
이 식에서 CONSOL\_MOSTFILLED 함수는 레코드 그룹의 행을 분석하여 행 하나를 식별합니다. CONSOL\_GETROWFIELD 함수는 연속 번호를 사용하여 해당 행의 포트 값을 읽고 이 값을 출력 포트에 기록합니다.

## 통합 매핑 예제

중복 고객 레코드를 통합해야 하는 조직이 있습니다. 이 조직의 사용자는 고객 레코드를 통합하기 위해 키 생성기 변환을 사용하여 데이터를 그룹화하고 통합 변환을 사용하여 레코드를 통합합니다.

사용자는 고객 레코드가 포함된 데이터 소스, 키 생성기 변환, 통합 변환 및 데이터 대상을 사용하여 매핑을 작성합니다. 이 매핑은 고객 레코드를 그룹화하고 그룹을 통합한 다음 하나의 통합된 레코드를 기록합니다.

다음 그림은 매핑을 보여 줍니다.



## 입력 데이터

분석하려는 입력 데이터에는 고객 정보가 포함됩니다.

다음 표에는 이 예의 입력 데이터가 포함되어 있습니다.

이름	주소	도시	상태	우편 번호	SSN
Dennis Jones	100 All Saints Ave	뉴욕	NY	10547	987-65-4320
Dennis Jones	1000 Alberta Rd	뉴욕	NY	10547	987-65-4320
D Jones	100 All Saints Ave	뉴욕	NY	10547-1521	

## 키 생성기 변환

키 생성기 변환을 사용하여 우편 번호 포트를 기준으로 입력 데이터를 그룹화합니다.

변환이 다음 데이터를 반환합니다.

SequenceId	GroupKey	이름	주소	도시	상태	우편 번호	SSN
1	10547	Dennis Jones	100 All Saints Ave	뉴욕	NY	10547	987-65-4320
2	10547	Dennis Jones	1000 Alberta Rd	뉴욕	NY	10547	
3	10547	D Jones	100 All Saints Ave	뉴욕	NY	10547-1521	987-65-4320



## 통합 변환

통합 변환을 사용하여 통합된 레코드를 생성합니다.

행 기반 전략 유형을 사용하는 통합 변환을 구성하십시오. 가장 자주 발생하는 값의 수가 가장 많은 행을 선택하는 모달 정확한 일치 전략을 선택합니다. 모달 정확한 일치 전략은 이 행의 값을 사용하여 통합된 레코드를 생성합니다. 통합된 레코드는 IsSurvivor 포트에서 "Y" 값을 포함하는 레코드입니다.

변환이 다음 데이터를 반환합니다.

GroupKey	이름	주소	도시	상태	우편 번호	SSN	IsSurvivor
10547	Dennis Jones	100 All Saints Ave	뉴욕	NY	10547	987-65-4320	N
10547	Dennis Jones	1000 Alberta Rd	뉴욕	NY	10547		N
10547	D Jones	100 All Saints Ave	뉴욕	NY	10547-1521	987-65-4320	N
10547	D Jones	100 All Saints Ave	뉴욕	NY	10547-1521	987-65-4320	연도

## 통합 매핑 출력

매핑 출력에 통합된 레코드만 포함되도록 통합 변환을 구성합니다.

이 예에서는 모달 정확한 일치 전략이 선택한 가장 자주 발생하는 값이 유효한 포트 값이라는 것을 확인할 수 있습니다. 통합된 레코드만 매핑 대상에 기록하려면 **고급** 보기를 선택하고 출력 모드를 "서바이버만"으로 설정합니다.

매핑을 실행하면 통합된 레코드만 매핑 출력에 포함됩니다.

## 통합 변환 구성

통합 변환을 구성할 때는 전략 유형을 선택하고 전략을 선택하거나 식을 기록하고 그룹화 포트를 선택하고 고급 옵션을 구성해야 합니다.

1. **통합** 보기를 선택합니다.
2. 전략 유형을 선택합니다.
3. 전략을 구성합니다.
  - 단순 전략 유형의 경우 각 포트에 대한 전략을 선택합니다.
  - 행 기반 전략 유형의 경우 전략을 선택합니다.
  - 고급 전략 유형의 경우 통합 함수를 사용하는 식을 작성합니다.
4. 그룹 기준 필드에서 그룹 식별자가 포함된 포트를 선택합니다.
5. 입력 데이터가 정렬되지 않은 경우 **고급** 보기에서 정렬을 활성화합니다.
6. 통합된 레코드 또는 모든 레코드를 포함하도록 출력을 구성합니다.

## 통합 변환 - 비원시 환경

비원시 환경에서 주소 유효성 검사기 변환의 처리는 변환을 실행하는 엔진에 따라 다릅니다.

다음과 같은 비원시 런타임 엔진에 대한 지원을 고려하십시오.

- Blaze 엔진. 제한적으로 지원됩니다.

- Spark 엔진. 제한적으로 지원됩니다.
- Databricks Spark 엔진. 지원되지 않습니다.

## 통합 변환 - Blaze 엔진

통합 변환은 다음과 같은 제한과 함께 지원됩니다.

- 각 환경에서 변환이 레코드를 처리하는 순서가 다를 수 있습니다.
- 각 환경에서 변환이 존속 레코드로 식별하는 레코드가 다를 수 있습니다.

## 통합 변환 - Spark 엔진

통합 변환은 다음과 같은 제한과 함께 지원됩니다.

- 각 환경에서 변환이 레코드를 처리하는 순서가 다를 수 있습니다.
- 각 환경에서 변환이 존속 레코드로 식별하는 레코드가 다를 수 있습니다.

## 통합 변환 - Databricks Spark 엔진

비원시 환경에서 통합 변환의 처리는 변환을 실행하는 엔진에 따라 다릅니다.

다음과 같은 비원시 런타임 엔진에 대한 지원을 고려하십시오.

- Databricks Spark 엔진. 지원되지 않습니다.

## 제 12 장

# 데이터 마스킹 변환

이 장에 포함된 항목:

- [데이터 마스킹 변환 개요, 191](#)
- [마스킹 기술, 192](#)
- [마스킹 규칙, 201](#)
- [특수 마스크 형식, 205](#)
- [기본값 파일, 209](#)
- [데이터 마스킹 변환 구성, 210](#)
- [데이터 마스킹 변환의 런타임 속성, 211](#)
- [데이터 마스킹 예제, 212](#)
- [데이터 마스킹 변환의 고급 속성, 214](#)
- [데이터 마스킹 변환 - 비원시 환경, 215](#)

## 데이터 마스킹 변환 개요

데이터 마스킹 변환은 중요한 프로덕션 데이터를 비프로덕션 환경의 실제와 같은 테스트 데이터로 변경합니다. 데이터 마스킹 변환은 사용자가 각 열에 구성된 마스킹 기술에 따라 소스 데이터를 수정합니다.

소프트웨어 개발, 테스트, 교육 및 데이터 마이닝을 위해 마스킹된 데이터를 작성할 수 있습니다. 마스킹된 데이터에서 데이터 관계를 유지하고 데이터베이스 테이블 간의 참조 무결성을 유지할 수 있습니다.

데이터 마스킹 변환은 소스의 데이터 유형 및 사용자가 열에 구성된 마스킹 기술에 따른 마스킹 규칙을 제공합니다. 문자열의 경우 문자열에서 바꿀 문자를 제한할 수 있습니다. 마스크에서 적용할 문자를 제한할 수 있습니다. 숫자 및 날짜의 경우 마스킹되는 데이터의 숫자 범위를 제공할 수 있습니다. 원래 숫자의 고정된 분산 또는 백분율 분산으로 범위를 구성할 수 있습니다. 데이터 통합 서비스는 사용자가 변환에 구성된 로컬에 따라 문자를 바꿉니다.

데이터 마스킹 변환을 통해 적용할 수 있는 마스킹 유형은 다음과 같습니다.

### 키 마스킹

동일한 소스 데이터, 마스킹 규칙 및 시드 값에 대한 고정 결과를 생성합니다. 고정 결과는 동일한 입력 값에 대해 반복 가능한 출력 값입니다.

### 무작위 마스킹

동일한 소스 데이터 및 마스킹 규칙에 대해 무작위의 반복 불가능한 결과를 생성합니다.

### 특수 마스크 형식

SSN, 신용 카드 번호, 전화 번호, URL, 전자 메일 주소 또는 IP 주소를 변경하는 특수 마스크 형식을 적용합니다.

## 마스킹 기술

마스킹 기술은 선택한 열에 적용할 데이터 마스킹의 유형입니다.

입력 열에 대해 다음 마스킹 기술 중 하나를 선택할 수 있습니다.

### 무작위

동일한 소스 데이터 및 마스킹 규칙에 대해 무작위의 반복 불가능한 결과를 생성합니다. 날짜, 숫자 및 문자열 데이터 유형을 마스킹할 수 있습니다. 무작위 마스킹에는 시드 값이 필요하지 않습니다. 무작위 마스킹의 결과는 비고정입니다.

### 식

식을 소스 열에 적용하여 데이터를 작성하거나 마스킹합니다. 모든 데이터 유형을 마스킹할 수 있습니다.

### 키

소스 데이터를 반복 가능 값으로 바꿉니다. 데이터 마스킹 변환은 동일한 소스 데이터, 마스킹 규칙 및 시드 값에 대한 확정 결과를 생성합니다. 날짜, 숫자 및 문자열 데이터 유형을 마스킹할 수 있습니다.

### 대체

사전에 있는 데이터 중 유사하지만 관계없는 데이터로 열의 데이터를 바꿉니다. 문자열 데이터 유형을 마스킹할 수 있습니다.

### 종속

다른 소스 열의 값을 기반으로 소스 열 한 개의 값을 바꿉니다. 문자열 데이터 유형을 마스킹할 수 있습니다.

### 토큰화

사용자 지정된 마스킹 조건에 따라 생성된 데이터로 소스 데이터를 바꿉니다. 데이터 마스킹 변환이 사용자 지정된 알고리즘에 지정된 규칙을 적용합니다. 문자열 데이터 유형을 마스킹할 수 있습니다.

### 특수 마스크 형식

신용 카드 번호, 전자 메일 주소, IP 주소, 전화 번호, SSN, SIN 또는 URL. 데이터 마스킹 변환이 기본 규칙을 적용하여 이러한 일반적인 유형의 중요한 데이터를 지능적으로 마스킹합니다.

### 마스킹 없음

데이터 마스킹 변환이 소스 데이터를 변경하지 않습니다.

기본값은 마스킹 없음입니다.

## 무작위 마스킹

무작위 마스킹은 무작위로 지정되지 않은 마스킹된 데이터를 생성합니다. 데이터 마스킹 변환은 동일한 소스 값이 다른 행에서 발생하는 경우 다른 값을 반환합니다. 마스킹 규칙을 정의하면 데이터 마스킹 변환이 반환하는 데이터 형식을 제어할 수 있습니다. 숫자, 문자열 및 날짜 값을 무작위 마스킹으로 마스킹합니다.

## 문자열 값 마스킹

문자열 열에 대한 무작위 출력을 생성하는 무작위 마스킹을 구성합니다. 출력 문자열의 각 문자에 대한 제한을 구성하려면 마스크 형식을 구성하십시오. 필터 문자를 구성하여 마스킹할 소스 문자 및 이러한 소스 문자를 사용하여 마스킹할 문자를 정의하십시오.

문자열 포트에는 다음 마스킹 규칙을 적용할 수 있습니다.

#### 범위

최소 및 최대 문자열 길이를 구성합니다. 데이터 마스킹 변환은 최소 문자열 길이와 최대 문자열 길이 사이의 무작위 문자로 구성된 문자열을 반환합니다.

#### 마스크 형식

입력 데이터의 각 문자를 대체할 문자 유형을 정의합니다. 각 문자를 알파벳, 숫자 또는 영숫자 문자 유형으로 제한할 수 있습니다.

#### 소스 문자열 문자

마스킹할 소스 문자열의 문자를 정의합니다. 예를 들어 입력 데이터에 발생하는 모든 숫자 기호(#) 문자를 마스킹할 수 있습니다. 소스 문자열 문자가 비어 있는 경우 데이터 마스킹 변환이 모든 입력 문자를 마스킹합니다.

#### 결과 문자열 대체 문자

대상 문자열의 문자를 결과 문자열 문자에 정의된 문자로 대체합니다. 예를 들어 다음 문자를 입력하면 각 마스크에 대문자 알파벳 문자(A~Z)가 포함되도록 구성됩니다.

ABCDEFGHIJKLMNOPQRSTUVWXYZ

## 숫자 값 마스킹

숫자 데이터를 마스킹할 경우 열에 대한 출력 값 범위를 구성할 수 있습니다. 데이터 마스킹 변환에서 포트 전체 자릿수에 따라 범위의 최소값 및 최대값 사이의 값을 반환합니다. 범위를 정의하려면 최소 및 최대 범위를 구성하거나 원본 소스 값의 분산을 기반으로 블러링 범위를 구성합니다.

숫자 데이터에 대해 다음 마스킹 매개 변수를 구성할 수 있습니다.

#### 범위

출력 값 범위를 정의합니다. 데이터 마스킹 변환에서 최소값과 최대값 사이의 숫자 데이터를 반환합니다.

#### 블러링 범위

소스 데이터의 고정된 분산 또는 백분율 분산 내에서 출력 값의 범위를 정의합니다. 데이터 마스킹 변환에서 소스 데이터 값에 가까운 숫자 데이터를 반환합니다. 범위 및 블러링 범위를 구성할 수 있습니다.

## 날짜 값 마스킹

무작위 마스킹으로 날짜 값을 마스킹하려면 출력 날짜의 범위를 구성하거나 분산을 선택합니다. 분산을 구성하는 경우 날짜에서 블러링할 부분을 선택합니다. 연도, 월, 일, 시간, 분 또는 초를 선택합니다. 데이터 마스킹 변환은 사용자가 구성한 범위의 날짜를 반환합니다.

날짜/시간 값을 마스킹하는 경우 다음 마스킹 규칙을 구성할 수 있습니다.

#### 범위

선택한 날짜/시간 값에 대해 반환할 최소값 및 최대값을 설정합니다.

#### 블러링

사용자가 날짜 단위에 적용한 분산을 바탕으로 날짜를 마스킹합니다. 데이터 마스킹 변환이 분산에 포함되는 날짜를 반환합니다. 연도, 월, 일, 시간, 분 또는 초를 블러링할 수 있습니다. 적용할 하한 및 상한 분산을 선택합니다.

## 식 마스킹

식 마스킹은 포트에 식을 적용하여 데이터를 변경하거나 새 데이터를 작성합니다. 식 마스킹을 구성하는 경우 식 편집기에서 식을 작성해야 합니다. 입력 및 출력 포트, 함수, 변수 및 연산자를 선택하여 식을 빌드합니다.

여러 포트의 데이터를 연결하여 다른 포트에 대한 값을 작성할 수 있습니다. 예를 들어 로그인 이름의 경우 다음과 같이 작성할 수 있습니다. 소스에 이름과 성 열이 있습니다. 조회 파일에서 이름과 성을 마스킹합니다. 데이터 마스킹 변환에서 **Login**이라는 이름의 다른 포트를 작성합니다. **Login** 포트에 대해 이름의 첫 글자와 성을 연결하는 다음과 같은 식을 구성합니다.

```
SUBSTR(FIRSTNM,1,1)||LASTNM
```

식을 작성할 때 실수를 최소화할 수 있도록 가리키고 클릭 인터페이스를 통해 함수, 포트, 변수 및 연산자를 선택하십시오.

식 편집기에는 식 마스킹이 구성되지 않은 출력 포트가 표시됩니다. 한 식의 출력을 다른 식의 입력으로 사용할 수는 없습니다. 출력 포트 이름을 식에 수동으로 추가할 경우 예기치 않은 결과를 얻을 수 있습니다.

식을 작성할 때는 해당 식이 포트의 데이터 유형과 일치하는 값을 반환하는지 확인해야 합니다. 식 포트의 데이터 유형이 숫자일 때 식의 데이터 유형이 동일하지 않으면 데이터 마스킹 변환이 **0**을 반환합니다. 식 포트의 데이터 유형이 문자열일 때 식의 데이터 유형이 동일하지 않으면 데이터 마스킹 변환이 **Null** 값을 반환합니다.

## 반복 가능한 식 마스킹

소스 열이 둘 이상의 테이블에서 나타나며 각 테이블의 열을 같은 값으로 마스킹해야 하는 경우 반복 가능 식 마스킹을 구성합니다.

반복 가능 식 마스킹을 구성하면 데이터 마스킹 변환은 식의 결과를 저장소 테이블에 저장합니다. 열이 다른 소스 테이블에서 나타나는 경우 데이터 마스킹 변환은 식이 아닌 저장소 테이블에서 마스킹된 값을 반환합니다.

## 사전 이름

반복 가능한 식 마스킹을 구성하는 경우 사전 이름을 입력해야 합니다. 사전 이름은 여러 데이터 마스킹 변환이 동일한 소스 값으로부터 동일하게 마스킹된 값을 생성할 수 있도록 하는 키입니다. 각 데이터 마스킹 변환에 동일한 사전 이름을 정의해야 합니다. 모든 텍스트를 사전 이름으로 입력할 수 있습니다.

## 저장소 테이블

저장소 테이블에는 세션 간 반복 가능 식 마스킹의 결과가 포함되어 있습니다. 저장소 테이블 행에는 소스 열 및 마스킹된 값 쌍이 들어 있습니다. 식 마스킹의 저장소 테이블은 대체 마스킹의 저장소 테이블과는 별개의 테이블입니다.

데이터 마스킹 변환이 반복 가능 식을 사용하여 값을 마스킹할 때마다 사전 이름, 로컬, 열 이름, 입력 값으로 저장소 테이블을 검색합니다. 저장소 테이블에서 행을 찾으면 저장소 테이블에서 마스킹된 값을 반환합니다. 데이터 마스킹 변환이 행을 찾지 못하면 열의 식에서 마스킹된 값을 생성합니다.

저장소에 암호화되지 않은 데이터가 있고 동일한 사전 이름을 키로 사용하는 경우 식 마스킹을 위해 저장소 테이블을 암호화해야 합니다.

## 식 마스킹에 대한 저장소 테이블 암호화

변환 언어 인코딩 함수를 사용하여 저장소 테이블을 암호화할 수 있습니다. 저장소 암호화를 활성화한 경우 저장소 테이블을 암호화해야 합니다.

1. **IDM\_EXPRESSION\_STORAGE** 저장소 테이블을 소스로 사용하는 매핑을 작성합니다.
2. 데이터 마스킹 변환을 작성합니다.
3. 마스킹된 값 포트에 식 마스킹 기술을 적용합니다.

4. MASKEDVALUE 포트에 다음 식을 사용합니다.

`Enc_Base64(AES_Encrypt(MASKEDVALUE, Key))`

5. 포트를 대상에 연결합니다.

## 예제

예를 들어 직원 테이블에는 다음 열이 포함됩니다.

FirstName  
LastName  
LoginID

데이터 마스킹 변환에서 **FirstName**과 **LastName**을 결합하는 식을 사용하여 **LoginID**를 마스킹합니다. 식 마스크를 반복 가능하도록 구성합니다. 반복 가능한 마스킹의 키로 사용할 사전 이름을 입력합니다.

**Computer\_Users** 테이블에는 **LoginID**가 포함되지만 **FirstName** 또는 **LastName** 열이 포함되지 않습니다.

Dept  
LoginID  
Password

**Employees** 테이블과 동일한 **LoginID**로 **Computer\_Users** 테이블의 **LoginID**를 마스킹하려면 **LoginID** 열에 대해 식 마스킹을 구성합니다. 반복 가능한 마스킹을 활성화하고 **LoginID** 직원 테이블에 정의한 것과 동일한 사전 이름을 입력합니다. 통합 서비스가 저장소 테이블에서 **LoginID** 값을 검색합니다.

통합 서비스가 저장소 테이블에서 **LoginID**에 대한 행을 찾지 못할 때 사용할 기본 식을 작성합니다.

**Computer\_Users** 테이블에 **FirstName** 또는 **LastName** 열이 없으므로 기본 식을 통해 작성되는 **LoginID**는 의미가 감소할 수 있습니다.

## 저장소 테이블 스크립트

**Informatica**는 저장소 테이블을 작성하기 위해 실행할 수 있는 스크립트를 제공합니다. 스크립트는 다음 위치에 있습니다.

`<PowerCenter installation directory>\client\bin\Extensions\DataMasking`

디렉터리에는 **Sybase**, **Microsoft SQL Server**, **IBM DB2** 및 **Oracle** 데이터베이스용 스크립트가 포함되어 있습니다. 각 스크립트의 이름은 `<Expression_<데이터베이스 유형>`입니다.

## 식 마스킹에 대한 규칙 및 지침

식 마스킹에 대해 다음 규칙 및 지침을 사용하십시오.

- 한 식의 출력을 다른 식의 입력으로 사용할 수는 없습니다. 출력 포트 이름을 식에 수동으로 추가할 경우 예기치 않은 결과를 얻을 수 있습니다.
- 가리키고 클릭 방법을 사용하여 식을 작성합니다. 식을 작성할 때 실수를 최소화할 수 있도록 가리키고 클릭 인터페이스를 통해 함수, 포트, 변수 및 연산자를 선택하십시오.
- 데이터 마스킹 변환이 반복 가능한 마스킹을 위해 구성되었지만 저장소 테이블이 존재하지 않는 경우 통합 서비스가 소스 데이터를 기본값으로 대체합니다.

## 키 마스킹

키 마스킹이 구성된 열은 소스 값과 시드 값이 동일할 경우 항상 고정된 데이터를 마스킹된 데이터로 반환합니다. 데이터 마스킹 변환은 이러한 열에 대해 고유한 값을 반환합니다.

열에 키 마스킹을 구성할 경우 데이터 마스킹 변환이 해당 열에 대한 시드 값을 작성합니다. 이 시드 값을 변경하여 서로 다른 데이터 마스킹 변환 간에 반복 가능 데이터를 생성할 수 있습니다. 예를 들어 키 마스킹을 구성하여

참조 무결성을 시행하려는 경우 한 테이블의 기본 키와 다른 테이블의 외래 키를 마스킹하는 데 동일한 시드 값을 사용합니다.

마스킹 규칙을 정의하면 데이터 마스킹 변환이 반환하는 데이터 형식을 제어할 수 있습니다. 키 마스킹은 문자열 및 숫자 값을 마스킹하는 데 사용할 수 있습니다.

## 문자열 값 마스킹

문자열에 대해 반복 가능한 출력을 생성하도록 키 마스킹을 구성할 수 있습니다. 마스크 형식을 구성하여 출력 문자열의 각 문자에 대한 제한을 정의하고 소스 문자열을 구성하여 마스킹할 소스 문자를 정의하고 결과 문자열 대체 문자를 구성하여 마스킹된 데이터를 특정 문자로 제한할 수 있습니다.

키 마스킹 문자열에 구성할 수 있는 마스킹 규칙은 다음과 같습니다.

### 시드

시드 값을 적용하면 열의 데이터가 고정 데이터로 마스킹됩니다. 1에서 1,000 사이의 숫자를 입력할 수 있습니다.

### 마스크 형식

입력 데이터의 각 문자를 대체할 문자 유형을 정의합니다. 각 문자를 알파벳, 숫자 또는 영숫자 문자 유형으로 제한할 수 있습니다.

### 소스 문자열 문자

마스킹할 소스 문자열의 문자를 정의합니다. 예를 들어 입력 데이터에 발생하는 모든 숫자 기호(#) 문자를 마스킹할 수 있습니다. 소스 문자열 문자가 비어 있는 경우 데이터 마스킹 변환이 모든 입력 문자를 마스킹합니다. 소스 문자열 문자의 수가 결과 문자열 문자의 수보다 작을 경우 데이터 마스킹 변환이 고유하지 않은 데이터를 반환할 수 있습니다.

### 결과 문자열 문자

대상 문자열의 문자를 결과 문자열 문자에 정의된 문자로 대체합니다. 예를 들어 다음 문자를 입력하면 각 마스크에 모든 대문자 알파벳 문자가 포함되도록 구성합니다.

ABCDEFGHIJKLMNOPQRSTUVWXYZ

## 숫자 값 마스킹

키 마스킹을 숫자 소스 데이터에 구성하면 확정 출력이 생성됩니다. 열에 숫자 키 마스킹을 구성할 경우 무작위 시드 값이 열에 할당됩니다. 데이터 마스킹 변환은 시드가 필요한 마스킹 알고리즘을 적용하여 소스 데이터를 마스킹합니다.

동일한 소스 값이 여러 열에서 발생하는 경우 반복 가능한 결과를 생성하도록 열의 시드 값을 변경할 수 있습니다. 예를 들어 두 테이블의 기본-외래 키 관계를 유지해야 하는 경우 각 데이터 마스킹 변환에서 기본 키 열의 시드 값과 외래 키 열의 시드 값을 동일한 값으로 입력하면 데이터 마스킹 변환이 동일한 숫자 값에 대해 고정 결과를 생성하고 두 테이블 간의 참조 무결성이 유지됩니다.

## 날짜/시간 값 마스킹

날짜/시간 값에 대해 키 마스킹을 구성하려면 데이터 마스킹 변환에서 무작위 숫자를 시드로 입력해야 합니다. 다른 열의 시드 값과 일치하도록 시드 값을 변경하면 열 간에 반복 가능한 날짜/시간 값이 반환됩니다.

데이터 마스킹 변환은 키 마스킹을 사용하여 1753년부터 2400년까지의 날짜를 마스킹할 수 있습니다. 소스 연도가 윤년인 경우 데이터 마스킹 변환이 윤년을 반환합니다. 소스 월의 일 수가 31일인 경우 데이터 마스킹 변환이 일 수가 31일인 월을 반환합니다. 소스 월이 2월인 경우 데이터 마스킹 변환이 2월을 반환합니다.

데이터 마스킹 변환은 항상 유효한 날짜를 생성합니다.



## 대체 마스크

대체 마스크는 유사하지만 관계 없는 데이터로 데이터 열을 바꿉니다. 대체 마스크를 사용하여 프로덕션 데이터를 실제 테스트 데이터로 바꿉니다. 대체 마스크를 구성하는 경우 대체 값이 포함된 사전을 정의합니다.

데이터 마스크 변환이 사용자가 구성한 사전에서 조회를 수행합니다. 데이터 마스크 변환이 소스 데이터를 사전의 데이터로 바꿉니다. 사전 파일에는 문자열 데이터, **datetime** 값, 정수 및 부동 소수점 숫자가 포함될 수 있습니다. **datetime** 값을 다음 형식으로 입력합니다.

mm/dd/yyyy

데이터를 반복 가능 또는 반복 불가능 값으로 대체할 수 있습니다. 반복 가능 값을 선택하는 경우 데이터 마스크 변환이 동일한 소스 데이터 및 시드 값에 대해 확정 결과를 생성합니다. 데이터를 확정 결과로 대체하도록 시드 값을 구성해야 합니다. 통합 서비스가 반복 가능 마스크를 위해 소스 및 마스크된 값의 저장소 테이블을 유지합니다.

두 이상의 데이터 열을 동일한 사전 행의 마스크된 값으로 대체할 수 있습니다. 한 입력 열의 대체 마스크를 구성합니다. 동일한 사전 행에서 마스크된 데이터를 수신하는 다른 열의 종속 데이터 마스크를 구성합니다.

## 사전

사전은 테이블의 각 행에 대한 대체 데이터와 일련 번호가 포함된 참조 테이블입니다. 플랫폼 파일의 대체 마스크를 위한 참조 테이블 또는 모델 리포지토리로 가져오는 관계형 테이블을 작성하십시오.

데이터 마스크 변환에서 일련 번호로 사전 행을 검색하는 숫자를 생성합니다. 데이터 마스크 변환에서 반복 가능한 대체 마스크에 대한 해시 키 또는 반복 불가능한 마스크에 대한 무작위 숫자를 생성합니다. 반복 가능한 대체 마스크를 구성한 경우 추가 조회 조건을 구성할 수 있습니다.

데이터 마스크 변환에서 두 이상의 포트를 마스크하도록 사전을 구성할 수 있습니다.

데이터 마스크 변환이 사전에서 대체 데이터를 검색할 경우, 변환 시 대체 데이터 값이 원본 값과 동일한지 검사하지 않습니다. 예를 들어, 데이터 마스크 변환에서 **John** 이름을 사전 파일의 동일한 이름(**John**)으로 대체할 수 있습니다.

다음 예제에서는 이름 및 성별이 포함된 사전 테이블을 보여 줍니다.

SNO	GENDER	FIRSTNAME
1	M	Adam
2	M	Adeel
3	M	Adil
4	F	Alice
5	F	Alison

이 사전에서 행의 첫 번째 필드는 일련 번호이고 두 번째 필드는 성별입니다. 통합 서비스는 항상 일련 번호로 사전 레코드를 조회합니다. 반복 가능한 마스크를 구성한 경우 성별을 조회 조건으로 추가할 수 있습니다. 통합 서비스는 해시 키를 사용하여 사전에서 행을 검색하고, 소스 데이터의 성별과 일치하는 성별을 가진 행을 찾습니다.

참조 테이블을 작성할 경우 다음 규칙과 지침을 사용하십시오.

- 테이블의 각 레코드에 일련 번호가 있어야 합니다.
- 일련 번호는 1부터 시작하는 순차적인 정수입니다. 일련 번호의 순서에서 번호가 누락될 수 없습니다.
- 일련 번호 열은 테이블 행의 아무 위치에나 있을 수 있습니다. 임의 레이블을 가질 수 있습니다.

플랫 파일 테이블을 사용하여 참조 테이블을 작성할 경우 다음 규칙과 지침을 사용하십시오.

- 플랫 파일 테이블의 첫 번째 행에는 각 레코드의 필드를 식별하는 열 레이블이 있어야 합니다. 필드는 쉼표로 구분됩니다. 첫 번째 행에 열 레이블이 없는 경우 통합 서비스는 첫 번째 행의 필드 값을 열 이름으로 가져옵니다.
- 플랫 파일 테이블은 \$PMLookupFileDir 조희 파일 디렉터리에 있어야 합니다. 기본적으로 이 디렉터리는 다음 위치에 있습니다.

<PowerCenter\_Installation\_Directory>\server\infa\_shared\LkpFiles

- Windows에서 플랫 파일 테이블을 작성하고 이를 UNIX 시스템에 복사한 경우 파일 형식이 UNIX에 적합한지 확인하십시오. 예를 들어, Windows와 UNIX는 행 마커 끝에 다른 문자를 사용합니다.

## 저장소 테이블

데이터 마스킹 변환은 세션 간 반복 가능 대체를 위해 저장소 테이블을 유지 관리합니다. 저장소 테이블 행에는 소스 열 및 마스킹된 값 쌍이 들어 있습니다. 데이터 마스킹 변환이 반복 가능 대체 값을 사용하여 값을 마스킹할 때마다 사전 이름, 로컬, 열 이름, 입력 값 및 시드로 저장소 테이블을 검색합니다. 행을 찾으면 저장소 테이블에서 마스킹된 값을 반환합니다. 데이터 마스킹 변환이 행을 찾지 못하면 해시 키를 사용하여 사전에서 행을 검색합니다.

플랫 파일 사전과 관계형 사전에 대한 저장소 테이블의 사전 이름 형식은 다릅니다. 플랫 파일 사전 이름은 파일 이름으로 식별됩니다. 관계형 사전 이름의 구문은 다음과 같습니다.

<Connection object>\_<dictionary table name>

Informatica는 관계형 저장소 테이블을 작성하기 위해 실행할 수 있는 스크립트를 제공합니다. 스크립트는 다음 위치에 있습니다.

<PowerCenter Client installation directory>\client\bin\Extensions\DataMasking

디렉터리에는 Sybase, Microsoft SQL Server, IBM DB2 및 Oracle 데이터베이스용 스크립트가 포함되어 있습니다. 각 스크립트의 이름은 Substitution\_<데이터베이스 유형>입니다. SQL 문 및 기본 키 제약 조건을 구성하면 다른 데이터베이스에서 테이블을 작성할 수 있습니다.

저장소에 암호화되지 않은 데이터가 있는 경우 대체 마스킹을 위해 저장소 테이블을 암호화하고, 동일한 시드 값 및 사전을 사용하여 동일한 열을 암호화해야 합니다.

## 대체 마스킹에 대한 저장소 테이블 암호화

변환 언어 인코딩 함수를 사용하여 저장소 테이블을 암호화할 수 있습니다. 저장소 암호화를 활성화한 경우 저장소 테이블을 암호화해야 합니다.

1. IDM\_SUBSTITUTION\_STORAGE 저장소 테이블을 소스로 사용하는 매핑을 작성합니다.
2. 데이터 마스킹 변환을 작성합니다.
3. 입력 값 및 마스킹된 값 포트에 대체 마스킹 기술을 적용합니다.
4. INPUTVALUE 포트에 다음 식을 사용합니다.

Enc\_Base64(AES\_Encrypt(INPUTVALUE, Key))

5. MASKEDVALUE 포트에 다음 식을 사용합니다.

Enc\_Base64(AES\_Encrypt(MASKEDVALUE, Key))

6. 포트를 대상에 연결합니다.

## 대체 마스크링 속성

대체 마스크링에 대해 다음과 같은 마스크링 규칙을 구성할 수 있습니다.

- **반복 가능 출력.** 세션 간에 확정 결과를 반환합니다. 데이터 마스크링 변환은 저장소 테이블에 마스크링된 값을 저장합니다.
- **시드 값.** 열에 대해 확정 마스크링된 데이터를 생성하기 위한 시드 값을 적용합니다. 1에서 1,000 사이의 숫자를 입력하십시오.
- **고유한 출력.** 데이터 마스크링 변환이 고유한 입력 값에 대해 고유한 출력 값을 생성하도록 강제 적용합니다. 두 입력 값이 같은 출력 값으로 마스크링되지는 않습니다. 사전에는 고유 출력을 활성화하기에 충분한 고유 행이 있어야 합니다.  
고유 출력을 비활성화하면 데이터 마스크링 변환이 입력 값을 고유 출력 값으로 마스크링할 수 없습니다. 사전에 행이 더 적게 포함될 수 있습니다.
- **고유한 포트.** 대체 마스크링에 대해 고유 레코드를 식별하는 데 사용되는 포트입니다. **Customer** 테이블의 이름을 마스크링하려는 경우를 예로 들어 보겠습니다. 이름을 고유한 포트에 포함하는 테이블 열을 선택하면 데이터 마스크링 변환이 중복 이름을 같은 마스크링된 값으로 바꿉니다. **Customer\_ID** 열을 고유한 포트에 선택하면 데이터 마스크링 변환은 각 이름을 고유한 값으로 바꿉니다.
- **사전 정보.** 대체 데이터 값을 포함하는 참조 테이블을 구성합니다. 참조 테이블을 선택하려면 **소스 선택**을 클릭합니다.
  - **사전 이름.** 선택하는 참조 테이블의 이름을 표시합니다.
  - **출력 열.** 데이터 마스크링 변환으로 반환할 열을 선택합니다.
- **조회 조건.** 대체 마스크링에 사용할 사전 행을 추가로 규정하는 조회 조건을 구성합니다. 조회 조건은 **SQL** 쿼리의 **WHERE** 절과 유사합니다. 조회 조건을 구성하는 경우 소스의 열 값을 사건의 열과 비교합니다.  
이름을 마스크링하려는 경우를 예로 들어 보겠습니다. 소스 데이터와 사전에는 이름 열과 성별 열이 있습니다. 각 여성의 이름을 사건의 여성 이름으로 바꾸는 조건을 추가할 수 있습니다. 조회 조건은 소스의 성별을 사건의 성별과 비교합니다.
  - **입력 포트.** 조회에서 사용할 소스 데이터 열입니다.
  - **사전 열.** 입력 포트를 비교할 사전 열입니다.

## 대체 마스크링에 대한 규칙 및 지침

대체 마스크링에 대해 다음 규칙 및 지침을 사용하십시오.

- 고유한 반복 가능 대체 마스크에 대해 저장소 테이블이 존재하지 않는 경우 세션이 실패합니다.
- 사전에 행이 없는 경우 데이터 마스크링 변환이 오류 메시지를 반환합니다.
- 데이터 마스크링 변환이 저장소 테이블의 로컬, 사전 및 시드를 사용하여 입력 값을 찾는 경우 행이 사전에 더 이상 존재하지 않는 경우에도 마스크링된 값을 검색합니다.
- 연결 개체를 삭제하거나 사전을 수정하는 경우 저장소 테이블이 잘립니다. 그렇지 않으면 예기치 않은 결과가 나타날 수 있습니다.
- 사건의 값 수가 소스 데이터의 고유한 값 수보다 적으면 데이터 마스크링 변환이 고유한 반복 가능 값을 사용하여 데이터를 마스크링할 수 없습니다. 데이터 마스크링 변환이 오류 메시지를 반환합니다.

## 종속 마스크링

종속 마스크링은 소스 데이터의 여러 열을 동일한 사전 행의 데이터로 대체합니다.

데이터 마스크링 변환에서 여러 열에 대해 대체 마스크링을 수행하면 마스크링된 데이터에 실제와 다른 필드 조합이 포함될 수 있습니다. 종속 마스크링을 구성하면 동일한 사전 행에서 여러 입력 열의 데이터를 대체할 수 있습니다. 마스크링된 데이터에는 "뉴욕, 뉴욕" 또는 "시카고, 일리노이"와 같이 유효한 조합이 포함됩니다.

종속 마스크를 구성할 때는 먼저 입력 열에 대체 마스크를 구성해야 합니다. 그런 다음 이 대체 열에 종속될 다른 입력 열을 구성합니다. 예를 들어 우편 번호 열에 대체 마스크를 선택하고 이 우편 번호 열에 종속될 열로 도시 및 주 열을 선택합니다. 종속 마스크는 대체된 도시 및 주 열이 대체된 우편 번호 값에 유효한 값인지 확인합니다.

**참고:** 대체 마스크를 먼저 구성하지 않은 열에는 종속 마스크를 구성할 수 없습니다.

열에 종속 마스크를 구성할 때 다음 마스크 규칙을 구성해야 합니다.

#### 종속 열

대체 마스크가 구성된 입력 열의 이름입니다. 데이터 마스크 변환은 해당 열의 마스크 규칙을 사용하여 사전에서 대체 데이터를 검색합니다. 대체 마스크가 구성된 열은 사전에서 마스크된 데이터를 검색할 때 키 열로 사용됩니다.

#### 출력 열

종속 마스크가 구성된 열의 값을 포함하는 사전 열의 이름입니다.

### 종속 마스크 예제

데이터 마스크 사전에는 다음 값을 포함하는 주소 행이 포함될 수 있습니다.

SNO	STREET	CITY	STATE	우편 번호	COUNTRY
1	32 Apple Lane	Chicago	IL	61523	KR
2	776 Ash Street	Dallas	TX	75240	KR
3	2229 Big Square	Atleeville	TN	38057	KR
4	6698 Cowboy Street	Houston	TX	77001	KR

주소 사전에 있는 도시, 주 및 우편 번호의 유효한 집합으로 소스 데이터를 마스크해야 합니다.

ZIP 포트에 대체 마스크를 구성합니다. ZIP 포트에 다음 마스크 규칙을 입력합니다.

규칙	값
사전 이름	주소
일련 번호 열	SNO
출력 열	우편 번호

City 포트에 종속 마스크를 구성합니다. City 포트에 다음 마스크 규칙을 입력합니다.

규칙	값
종속 열	우편 번호
출력 열	도시

State 포트에 종속 마스크를 구성합니다. State 포트에 다음 마스크 규칙을 입력합니다.

규칙	값
종속 열	우편 번호
출력 열	상태

데이터 마스크 변환은 우편 번호를 마스크할 때 우편 번호에 해당하는 유효한 도시 및 주를 사전 행에서 반환합니다.

## 토큰화 마스크

토큰화 마스크 기술을 사용하여 알고리즘에 지정하는 조건에 따라 소스 문자열 데이터를 마스크할 수 있습니다. 예를 들어 가짜 전자 메일 주소가 포함된 알고리즘을 작성하여 소스 데이터의 필드 항목을 바꿀 수 있습니다.

토큰화 마스크를 사용하여 마스크된 데이터의 형식을 구성할 수 있습니다. 토큰화 마스크를 사용할 수 있으려면 토큰나이저 이름을 마스크 알고리즘에 할당해야 합니다. 토큰나이저 이름은 사용된 마스크 알고리즘(JAR)을 참조합니다. 토큰화 마스크 기술을 적용하는 경우 토큰나이저 이름을 지정합니다.

### 토큰화 마스크 구성

토큰화 마스크 기술을 사용하기 전에 다음 작업을 수행하십시오.

1. 다음 경로에서 `tokenprovider` 디렉터리를 찾아봅니다. `<Informatica_home>\services\shared`.
2. 다음 XML 파일을 엽니다. `com.informatica.products.ilm.tx-tokenizerprovider.xml`.
3. 사용할 각 토큰나이저에 대한 클래스 파일의 토큰나이저 이름 및 정규화된 이름을 추가합니다. `tokenprovider` 디렉터리의 `com.informatica.products.ilm.tx-tokenprovider-<Build-Number>.jar` 클래스 안에 토큰나이저 클래스를 구현합니다. 각 토큰나이저에 대해 다음 예와 같이 XML 파일의 정보를 입력합니다.

```
<TokenizerProvider>
<Tokenizer Name="CCTokenizer"
ClassName="com.informatica.tokenprovider.CCTokenizer"/>
</TokenizerProvider>
```

여기서

- `Tokenizer Name`은 따옴표 안의 사용자 정의 이름입니다.
- `ClassName`은 `CLASSNAME` 특성에 대한 사용자 정의 이름입니다. 클래스 이름은 `com.informatica.products.ilm.tx-tokenprovider-<Build-Number>.jar` 안에서부터 구현하십시오.

구성이 완료되면 토큰화 마스크 기술을 사용할 수 있습니다. 토큰나이저 이름을 입력하여 매핑을 작성할 때 사용할 알고리즘을 지정합니다.

## 마스크 규칙

마스크 규칙은 마스크 기술을 선택한 후에 구성하는 옵션입니다.

무작위 또는 키 마스크 기술을 선택한 경우 마스크 형식, 소스 문자열 문자 및 결과 문자열 문자를 구성할 수 있습니다. 무작위 마스크를 선택한 경우 범위 또는 블러링을 구성할 수 있습니다.

다음 표에는 각 마스킹 기술에 대해 구성할 수 있는 마스킹 규칙이 설명되어 있습니다.

마스킹 규칙	설명	마스킹 기술	소스 데이터 유형
마스크 형식	출력 문자열의 각 문자를 알파벳, 숫자 또는 영숫자 문자로 제한하는 마스크입니다.	무작위 및 키	문자열
소스 문자열 문자	마스킹하거나 마스킹에서 제외할 소스 문자의 집합입니다.	무작위 및 키	문자열
결과 문자열 대체 문자	마스크에 포함하거나 제외할 문자의 집합입니다.	무작위 및 키	문자열
범위	출력 값의 범위입니다. - 숫자. 데이터 마스킹 변환이 최소값과 최대값 사이의 숫자 데이터를 반환합니다. - 문자열. 최소 문자열 길이와 최대 문자열 길이 사이의 무작위 문자로 구성된 문자열을 반환합니다. - 날짜/시간. 최소 날짜/시간과 최대 날짜/시간 사이의 날짜와 시간을 반환합니다.	무작위	숫자 문자열 날짜/시간
블러링	소스 데이터의 고정 또는 백분율 분산을 포함하는 출력 값의 범위입니다. 데이터 마스킹 변환이 소스 데이터의 값에 근사한 데이터를 반환합니다. 날짜/시간 열에는 고정 분산이 필요합니다. 열에는 고정 분산이 필요합니다.	무작위	숫자 날짜/시간

## 마스크 형식

마스크 형식을 구성하여 출력 열의 각 문자를 알파벳, 숫자 또는 영숫자 문자로 제한합니다. 다음 문자를 사용하여 마스크 형식을 정의하십시오.

A, D, N, X, +, R

**참고:** 마스크 형식에는 대문자가 포함됩니다. 소문자 마스크를 입력할 경우 데이터 마스킹 변환이 해당 문자를 대문자로 변환합니다.

다음 표에는 마스크 형식 문자가 설명되어 있습니다.

문자	설명
A	알파벳 문자. a~z 및 A~Z의 ASCII 문자를 예로 들 수 있습니다.
D	0~9의 자릿수. 데이터 마스킹 변환이 자릿수가 0~9가 아닌 문자에 대해 "X"를 반환합니다.
N	영숫자 문자. a~z, A~Z 및 0~9의 ASCII 문자를 예로 들 수 있습니다.
X	모든 문자. 영숫자 또는 기호를 예로 들 수 있습니다.
+	마스킹 없음.
R	나머지 문자. R을 지정하면 문자열의 나머지 문자가 모든 문자 유형이 될 수 있습니다. R은 마스크의 마지막 문자로 표시되어야 합니다.

예를 들어 부서 이름은 다음과 같은 형식을 가질 수 있습니다.

`nnn-<department_name>`

처음 3개 문자를 숫자로 바꾸고 부서 이름을 알파벳으로 바꾸고 출력에 대시를 유지하도록 마스크를 구성할 수 있습니다. 다음 마스크 형식을 구성하십시오.

`DDD+AAAAAAAAAAAAAAAA`

데이터 마스크 변환이 첫 3개 문자를 숫자 문자로 바꿉니다. 네 번째 문자는 바뀌지 않습니다. 데이터 마스크 변환이 나머지 문자를 알파벳 문자로 바꿉니다.

마스크 형식을 정의하지 않으면 데이터 마스크 변환이 각 소스 문자를 임의의 문자로 바꿉니다. 마스크 형식이 입력 문자열보다 긴 경우 데이터 마스크 변환이 마스크 형식에서 남는 문자를 무시합니다. 마스크 형식이 소스 문자열보다 짧은 경우 데이터 마스크 변환이 나머지 문자를 형식 R로 마스크합니다.

**참고:** 범위 옵션으로 마스크 형식을 구성할 수는 없습니다.

## 소스 문자열 문자

소스 문자열 문자는 마스크 또는 마스크하지 않기 위해 선택하는 소스 문자입니다. 소스 문자열에서 문자 위치는 중요하지 않습니다. 소스 문자는 대/소문자를 구분합니다.

원하는 수만큼 문자를 구성할 수 있습니다. 문자가 비어 있는 경우 데이터 마스크 변환이 열의 모든 소스 문자를 바꿉니다.

소스 문자열 문자에 대해 다음 옵션 중 하나를 선택합니다.

### 다음만 마스크

데이터 마스크 변환은 구성된 소스의 문자를 소스 문자열 문자로 마스크합니다. 예를 들어 문자 A, B 및 c를 입력하는 경우 문자가 소스 데이터에서 발생하면 데이터 마스크 변환이 A, B 또는 c를 다른 문자로 바꿉니다. A, B 또는 c가 아닌 소스 문자는 변경되지 않습니다. 마스크는 대/소문자를 구분합니다.

### 모두 마스크(다음 제외)

소스 문자열에서 발생하는 소스 문자열 문자를 제외하고 모든 문자를 마스크합니다. 예를 들어 필터 소스 문자 “-”를 입력하고 모두 마스크(다음 제외)를 선택하는 경우 소스 데이터에서 “-” 문자가 발생할 때 데이터 마스크 변환이 “-” 문자를 바꾸지 않습니다. 소수 문자의 나머지는 변경됩니다.

## 소스 문자열 예제

소스 파일에 Dependents라는 이름의 열이 있습니다. Dependents 열에는 쉼표로 구분된 이름이 2개 이상 있습니다. 사용자는 이 Dependents 열을 마스크하고 테스트 데이터에 쉼표를 유지하여 이름을 구분해야 합니다.

Dependents 열에 대해 소스 문자열 문자를 선택합니다. 마스크하지 않음을 선택하고 건너뛰 소스 문자로 “,”를 입력합니다. 따옴표는 입력하지 마십시오.

데이터 마스크 변환이 소스 문자열에서 쉼표를 제외한 모든 문자를 바꿉니다.

## 결과 문자열 대체 문자

결과 문자열 대체 문자는 마스크된 데이터에서 대체 문자로 선택하는 문자입니다. 결과 문자열 대체 문자를 구성할 때 데이터 마스크 변환은 소스 문자열의 문자를 결과 문자열 대체 문자로 바꿉니다. 다른 입력 값에 대해 같은 출력이 생성되지 않도록 하려면 광범위한 대체 문자를 구성하거나 소스 문자를 몇 개만 마스크하십시오. 문자열의 각 문자 위치는 중요하지 않습니다.

결과 문자열 대체 문자에 대해 다음 옵션 중 하나를 선택합니다.

## 다음만 사용

결과 문자열 대체 문자로 정의하는 문자만 사용하여 소스를 마스킹합니다. 예를 들어 A, B, c 문자를 입력하면 데이터 마스킹 변환은 소스 열의 모든 문자를 A, B, c로 바꿉니다. 예를 들어 "horse"라는 단어는 "BAcBA"로 바뀔 수 있습니다.

## 모두 사용(다음 제외)

결과 문자열 대체 문자로 정의하는 문자를 제외한 모든 문자를 사용하여 소스를 마스킹합니다. 예를 들어 결과 문자열 대체 문자로 A, B, c를 입력하면 마스킹된 데이터에는 A, B, c 문자가 포함되지 않습니다.

## 결과 문자열 대체 문자 예제

종속 열의 모든 쉼표를 세미콜론으로 대체하려면 다음 태스크를 수행하십시오.

1. 쉼표를 소스 문자열 문자로 구성하고 다음만 마스크를 선택합니다.  
종속 열에 쉼표가 있을 경우 데이터 마스킹 변환에서 쉼표만 마스킹합니다.
2. 세미콜론을 결과 문자열 대체 문자로 구성하고 다음만 사용을 선택합니다.  
데이터 마스킹 변환에서 종속 열의 모든 쉼표를 세미콜론으로 대체합니다.

## 범위

숫자, 날짜 또는 문자열 데이터의 범위를 정의합니다. 숫자 또는 날짜 값의 범위를 정의하면 데이터 마스킹 변환이 최소값과 최대값 사이에 있는 값을 사용하여 소스 데이터를 마스킹합니다. 문자열에 대한 범위를 구성하는 경우 문자열 길이 범위를 구성합니다.

### 문자열 범위

무작위 문자열 마스킹을 구성하는 경우 데이터 마스킹 변환이 소스 문자열의 길이가 다양한 문자열을 생성합니다. 필요에 따라 최소 및 최대 문자열 너비를 구성할 수 있습니다. 최소 또는 최대 너비로 입력하는 값은 양수여야 합니다. 각 너비는 포트 정밀도보다 작거나 같아야 합니다.

### 숫자 범위

숫자 열의 최소값과 최대값을 설정합니다. 최대값은 포트 전체 자릿수보다 작거나 같아야 합니다. 기본 범위는 1부터 포트 전체 자릿수 길이까지입니다.

### 날짜 범위

날짜/시간 값에 대한 최소값 및 최대값을 설정합니다. 최소값 및 최대값 필드에는 기본 최소 및 최대 날짜가 포함됩니다. 기본 날짜/시간 형식은 MM/DD/YYYY HH24:MI:SS입니다. 최대 날짜/시간은 최소 날짜/시간보다 나중이어야 합니다.

## 블러링

블러링은 소스 데이터 값의 고정 또는 백분율 분산에 포함되는 출력 값을 작성합니다. 블러링을 구성하면 원래 값에 근사한 무작위 값이 반환됩니다. 숫자 및 날짜 값을 블러링할 수 있습니다.

### 숫자 값 블러링

고정 또는 백분율 분산을 선택하여 숫자 소스 값을 블러링합니다. 낮은 블러링 값은 소스 값 미만의 분산입니다. 높은 블러링 값은 소스 값 이상의 분산입니다. 낮은 값과 높은 값은 0보다 크거나 같아야 합니다. 데이터 마스킹 변환이 마스킹된 데이터 반환할 때 사용자가 정의한 범위 내의 숫자 데이터가 반환됩니다.



다음 표에는 입력 소스 값이 66일 때 범위 값을 불러링한 경우에 대한 마스크 결과가 설명되어 있습니다.

블러링 유형	하한	상한	결과
고정	0	10	66와 76 사이
고정	10	0	56와 66 사이
고정	10	10	56와 76 사이
백분율	0	50	66와 99 사이
백분율	50	0	33와 66 사이
백분율	50	50	33와 99 사이

## 날짜 값 블러링

블러링을 구성하여 소스 날짜의 분산으로 날짜를 마스크합니다. 분산을 적용할 날짜의 단위를 선택합니다. 연도, 월, 일 또는 시간을 선택할 수 있습니다. 소스 날짜 단위의 위 및 아래 분산을 정의하는 하한 및 상한을 입력합니다. 데이터 마스크 변환은 이 분산을 적용하고 분산 내에 포함되는 날짜를 반환합니다.

예를 들어 마스크된 날짜를 소스 날짜에서 2년 내의 날짜로 제한하려면 연도를 단위로 선택합니다. 하한 및 상한으로 2를 입력합니다. 소스 날짜가 02/02/2006인 경우 데이터 마스크 변환이 02/02/2004에서 02/02/2008 사이의 날짜를 반환합니다.

기본 블러링 단위는 연도입니다.

## 특수 마스크 형식

특수 마스크 형식은 데이터의 일반 유형에 적용할 수 있는 마스크입니다. 특수 마스크 형식을 사용하면 데이터 마스크 변환이 실제 형식이지만 유효한 값이 아닌 마스크된 값을 반환합니다.

예를 들어 SSN을 마스크하는 경우 데이터 마스크 변환은 형식은 올바르지만 유효하지 않은 SSN을 반환합니다. 사회 보장 번호에 대해 반복 마스크를 구성할 수 있습니다.

다음 유형의 데이터에 대해 특수 마스크를 구성합니다.

- 사회 보장 번호
- 신용 카드 번호
- 전화 번호
- URL 주소
- 전자 메일 주소
- IP 주소
- 사회 보험 번호

소스 데이터 형식 또는 데이터 유형이 마스크에 대해 유효하지 않은 경우 데이터 통합 서비스는 기본 마스크를 데이터에 적용합니다. 통합 서비스는 기본값 파일에서 마스크된 값을 적용합니다. 기본값 파일을 편집하여 기본값을 변경할 수 있습니다.

## 신용 카드 번호 마스킹

데이터 마스킹 변환은 논리적으로 유효한 신용 카드 번호를 생성하여 유효한 신용 카드 번호를 마스킹합니다. 소스 신용 카드 번호의 길이는 13~19자리여야 합니다. 입력 신용 카드 번호에는 신용 카드 업계 규칙에 기반하는 유효한 체크섬이 포함되어야 합니다.

소스 신용 카드 번호에는 숫자, 공백 및 하이픈이 포함될 수 있습니다. 신용 카드에 잘못된 문자가 포함되거나 길이가 올바르지 않은 경우 통합 서비스가 세션 로그에 오류를 기록합니다. 통합 서비스는 소스 데이터가 잘못된 경우 기본 신용 카드 번호 마스크를 적용합니다.

데이터 마스킹 변환은 6자리 BIN(은행 식별 번호)을 마스킹하지 않습니다. 예를 들어 데이터 마스킹 변환은 신용 카드 번호 4539 1596 8210 2773을 4539 1516 0556 7067로 마스킹할 수 있습니다. 데이터 마스킹 변환은 유효한 체크섬을 포함하는 마스킹된 번호를 작성합니다.

## 전자 메일 주소 마스킹

데이터 마스킹 변환을 사용하여 문자열 값이 포함된 전자 메일 주소를 마스킹할 수 있습니다. 데이터 마스킹 변환은 전자 메일 주소를 무작위 ASCII 문자로 마스킹하거나 실제 전자 메일 주소로 바꿉니다.

다음과 같은 전자 메일 주소 마스킹 유형을 적용할 수 있습니다.

### 표준 전자 메일 마스킹

데이터 마스킹 변환이 무작위 ASCII 문자를 반환하여 전자 메일 주소를 마스킹합니다. 예를 들어 데이터 마스킹 변환을 Georgesmith@yahoo.com을 KtrlupQAPyk@vdSKh.BIC로 마스킹할 수 있습니다. 기본값은 표준입니다.

### 고급 전자 메일 마스킹

데이터 마스킹 변환이 변환 출력 포트 또는 사전 열에서 파생된 실제 전자 메일 주소로 전자 메일 주소를 마스킹합니다.

## 고급 전자 메일 마스킹

고급 전자 메일 마스킹 유형을 사용하면 전자 메일 주소를 실제와 같은 다른 전자 메일 주소로 마스킹할 수 있습니다. 데이터 마스킹 변환은 사전 열 또는 변환 출력 포트를 바탕으로 전자 메일 주소를 작성합니다.

전자 메일 주소의 로컬 부분은 매핑 출력 포트를 바탕으로 작성할 수 있습니다. 또는 관계형 테이블 또는 플랫폼 파일 열을 바탕으로 전자 메일 주소의 로컬 부분을 작성할 수도 있습니다.

데이터 마스킹 변환은 상수 값 또는 도메인 사전의 무작위 값으로부터 전자 메일 주소의 도메인 이름을 작성할 수 있습니다.

고급 전자 메일 마스킹은 다음 옵션을 사용하여 작성할 수 있습니다.

### 종속 포트 기반 전자 메일 주소

데이터 마스킹 변환 출력 포트를 바탕으로 전자 메일 주소를 작성할 수 있습니다. 이름 및 성 열에 대한 변환 출력 포트를 선택합니다. 데이터 마스킹 변환이 사용자가 이름 및 성 길이에 지정한 값을 바탕으로 이름, 성 또는 이름과 성을 마스킹합니다.

### 사전 기반 전자 메일 주소

사전의 열을 바탕으로 전자 메일 주소를 작성할 수 있습니다. 참조 테이블을 사전의 소스로 선택합니다.

이름 및 성에 대한 사전 열을 선택합니다. 데이터 마스킹 변환이 사용자가 이름 및 성 길이에 지정한 값을 바탕으로 이름, 성 또는 이름과 성을 마스킹합니다.

## 고급 전자 메일 주소 마스킹 유형의 구성 매개 변수

고급 전자 메일 주소 마스킹을 구성하는 경우 구성 매개 변수를 지정합니다.

다음 구성 매개 변수를 지정할 수 있습니다.

### 구분자

점, 하이픈 또는 밑줄과 같은 구분자를 선택하여 전자 메일 주소에서 이름과 성을 구분할 수 있습니다. 전자 메일 주소에서 이름과 성을 구분하지 않으려는 경우 구분자를 비워 둡니다.

### FirstName 열

전자 메일 주소의 이름을 마스킹할 데이터 마스킹 변환 출력 포트 또는 사전 열을 선택합니다.

### LastName 열

전자 메일 주소의 성을 마스킹할 데이터 마스킹 변환 출력 포트 또는 사전 열을 선택합니다.

### FirstName 또는 LastName 열의 길이

이름 및 성 열의 마스킹할 문자 길이를 제한합니다. 예를 들어 입력 데이터의 이름이 **Timothy**이고 성이 **Smith**인 경우가 있습니다. 이름 열의 길이로 **5**를 선택합니다. 성 열의 길이를 **1**로 선택하고 구분자로 점을 선택합니다. 데이터 마스킹 변환이 다음 전자 메일 주소를 생성합니다.

timot.s@<domain\_name>

### DomainName

gmail.com과 같은 상수 값을 도메인 이름에 사용할 수 있습니다. 또는 도메인 이름 목록이 포함되는 다른 사전 파일을 지정할 수 있습니다. 도메인 사전은 플랫폼 파일 또는 관계형 테이블이 될 수 있습니다.

## IP 주소 마스킹

데이터 마스킹 변환은 IP 주소를 마침표로 구분된 4개의 번호로 분할하여 다른 IP 주소로 마스킹합니다. 첫 번째 번호는 네트워크입니다. 네트워크 번호는 네트워크 범위 내에서 마스킹됩니다.

클래스 A IP 주소는 클래스 A IP 주소로 마스킹되고 10.x.x.x 주소는 10.x.x.x 주소로 마스킹됩니다. 클래스 및 개인 네트워크 주소는 마스킹되지 않습니다. 예를 들어 데이터 마스킹 변환은 11.12.23.34를 75.32.42.52로 마스킹하고 10.23.24.32를 10.61.74.84로 마스킹할 수 있습니다.

**참고:** 데이터 마스킹 변환은 IP 주소의 클래스 또는 개인 네트워크를 마스킹하지 않으므로 많은 수의 IP 주소를 마스킹하는 경우 고유하지 않은 값이 반환될 수 있습니다.

## 전화 번호 마스킹

데이터 마스킹 변환은 원래 전화 번호의 형식을 변경하지 않고 전화 번호를 마스킹합니다. 예를 들어 데이터 마스킹 변환이 전화 번호 (408)382 0658을 (607)256 3106으로 마스킹할 수 있습니다.

소스 데이터는 숫자, 공백, 하이픈 및 괄호를 포함할 수 있습니다. 통합 서비스는 영문자 또는 특수 문자를 마스킹하지 않습니다.

데이터 마스킹 변환은 마스크 문자열, 정수 및 bigint 데이터를 마스킹할 수 있습니다.

## 사회 보장 번호 마스킹

데이터 마스킹 변환은 사회 보장 관리의 최신 높은 그룹 목록에 따라 유효하지 않은 사회 보장 번호를 생성합니다. 높은 그룹 목록에는 사회 보장 관리가 제출한 유효한 숫자가 들어 있습니다.

기본 높은 그룹 목록은 다음 위치에 있는 텍스트 파일입니다.

<Installation Directory>\infa\_shared\SrcFiles\highgroup.txt

워크플로우에서 높은 그룹 목록 파일을 사용하려면 텍스트 파일을 데이터 통합 서비스에 대해 구성하는 소스 디렉터리에 복사합니다.

데이터 마스크 변환은 높은 그룹 목록에 없는 SSN 번호를 생성합니다. 사회 보장 관리에서 매달 높은 그룹 목록을 업데이트합니다. 다음 위치에서 최신 버전의 목록을 다운로드합니다.

<http://www.socialsecurity.gov/employer/ssns/highgroup.txt>

## 사회 보장 번호 형식

데이터 마스크 변환은 9개의 숫자가 포함된 SSN 형식을 허용합니다. 숫자는 문자 집합으로 구분될 수 있습니다. 예를 들어 데이터 마스크 변환은 다음 형식을 허용합니다. `+54-*9944$#789-,*()`.

## 지역 번호 요구 사항

데이터 마스크 변환은 소스와 동일한 형식에 유효하지 않은 사회 보장 번호를 반환합니다. SSN의 첫 3자리는 지역 번호를 정의합니다. 데이터 마스크 변환은 지역 번호를 마스크하지 않습니다. 그룹 번호와 일련 번호를 마스크합니다. 소스 SSN에는 유효한 지역 번호가 포함되어야 합니다. 데이터 마스크 변환은 높은 그룹 목록에서 지역 번호를 찾은 다음 사용되지 않은 번호 중에서 마스크된 데이터로 적용할 수 있는 번호의 범위를 결정합니다. SSN이 유효하지 않은 경우 데이터 마스크 변환이 소스 데이터를 마스크하지 않습니다.

## 반복 가능한 사회 보장 번호 마스크

사회 보장 번호에 대해 반복 마스크를 구성할 수 있습니다. 사회 보장 번호에 대해 반복 가능 마스크를 구성하려면 반복 가능 출력을 클릭하고 시드 값 또는 매핑 매개 변수를 선택합니다.

시드 값을 선택하면 디자이너가 난수를 시드로 할당합니다. 서로 다른 소스 데이터에서 같은 사회 보장 번호를 생성하려면 각 데이터 마스크 변환의 시드 값을 다른 변환의 사회 보장 번호에 대한 시드 값과 일치하도록 변경합니다. 매핑에서 데이터 마스크 변환을 정의한 경우 시드 값에 대한 매핑 매개 변수를 구성할 수 있습니다.

데이터 마스크 변환은 반복 가능 마스크를 사용하여 확정 사회 보장 번호를 반환합니다. 데이터 마스크 변환은 사회 보장 관리국에서 발급한 유효 사회 보장 번호는 반환할 수 없으므로 모든 고유 사회 보장 번호를 반환할 수 없습니다.

## URL 주소 마스크

데이터 마스크 변환은 `://` 문자열을 검색하고 그 오른쪽에 있는 하위 문자열을 구문 분석하여 URL을 구문 분석합니다. 소스 URL은 `://` 문자열을 포함해야 합니다. 소스 URL은 숫자 및 영문자를 포함할 수 있습니다.

데이터 마스크 변환은 URL의 프로토콜을 마스크하지 않습니다. 예를 들어 URL이 `http://www.yahoo.com`이면 데이터 마스크 변환이 `http://MgL.aHjCa.VsD/`를 반환할 수 있습니다. 데이터 마스크 변환이 유효하지 않은 URL을 생성할 수 있습니다.

**참고:** 데이터 마스크 변환은 항상 URL에 대해 ASCII 문자를 반환합니다.

## 사회 보험 번호 마스크

데이터 마스크 변환은 9개의 숫자로 이루어진 사회 보험 번호를 마스크합니다. 숫자는 문자 집합으로 구분될 수 있습니다.

번호에 구분자가 포함되지 않은 경우 마스크된 번호에 구분자가 포함되지 않습니다. 그렇지 않은 경우 마스크된 번호의 형식은 다음과 같습니다.

XXX-XXX-XXX

## 반복 가능 SIN 번호

반복 가능 SIN 값을 반환하도록 데이터 마스킹 변환을 구성할 수 있습니다. 반복 가능 SIN 마스킹에 대한 포트를 구성하는 경우 소스 SIN 값과 시드 값이 동일할 때마다 데이터 마스킹 변환이 확정적인 마스킹된 데이터를 반환합니다.

반복 가능 SIN 번호를 반환하려면 **반복 가능 값**을 활성화하고 시드 숫자를 입력합니다. 데이터 마스킹 변환은 각 SIN에 대해 고유한 값을 반환합니다.

## SIN 시작 자릿수

마스킹된 SIN의 첫 번째 숫자를 정의할 수 있습니다.

**시작 숫자**를 활성화하고 숫자를 입력합니다. 데이터 마스킹 변환이 입력하는 숫자로 시작하는 마스킹된 SIN 번호를 작성합니다.

# 기본값 파일

소스 데이터 형식 또는 데이터 유형이 마스크에 유효하지 않은 경우 데이터 통합 서비스가 기본 마스크를 데이터에 적용합니다. 통합 서비스는 기본값 파일에서 마스킹된 값을 적용합니다. 기본값 파일을 편집하여 기본값을 변경할 수 있습니다.

기본값 파일은 XML 파일이며 다음 위치에서 찾을 수 있습니다.

<설치 디렉터리>\infa\_shared\SrcFiles\defaultValue.xml

워크플로우에서 기본값 파일을 사용하려면 데이터 통합 서비스에 구성된 소스 디렉터리에 기본값 파일을 복사하십시오.

defaultValue.xml 파일에는 다음 이름-값 쌍이 포함됩니다.

```
<?xml version="1.0" standalone="yes" ?>
<defaultValue
  default_char = "X"
  default_digit = "9"
  default_date = "11/11/1111 00:00:00"
  default_email = "abc@xyz.com"
  default_ip = "99.99.9.999"
  default_url = "http://www.xyz.com"
  default_phone = "999 999 999 9999"
  default_ssn = "999-99-9999"
  default_cc = "9999 9999 9999 9999"
  default_sin = "999-999-999"
  default_seed = "500"/>
```

관련 항목:

- [“데이터 통합 서비스 구성” 페이지 210](#)

## 데이터 마스킹 변환 구성

다음 단계를 사용하여 데이터 마스킹 변환을 구성합니다.

1. 데이터 통합 서비스의 실행 옵션을 구성합니다.
2. 변환을 작성합니다.
3. 입력 포트를 정의합니다.
4. 변경할 각 포트에 대한 마스킹 규칙을 구성합니다.
5. 데이터를 미리 보고 결과를 확인합니다.

## 데이터 통합 서비스 구성

데이터 통합 서비스에 대한 실행 옵션은 Informatica Administrator(Administrator 도구)에서 구성할 수 있습니다.

실행 옵션을 구성하여 다음 기본 디렉터리를 설정하십시오.

- 홈 디렉터리. 소스 디렉터리 및 캐시 디렉터리를 포함합니다.
- 소스 디렉터리. 워크플로우의 소스 파일을 포함합니다. 예를 들어 소스 디렉터리에는 highgrp.txt 및 defaultvalue.xml 파일이 포함될 수 있습니다.
- 캐시 디렉터리. 대체 마스킹에 대한 캐시 파일을 포함합니다.

실행 옵션에 대한 값을 설정하려면 Administrator 도구를 열고 **도메인 탐색기**에서 데이터 통합 서비스를 선택합니다. **속성** 보기를 클릭한 후 **실행 옵션** 섹션에서 **편집**을 클릭합니다.

관련 항목:

- [“기본값 파일” 페이지 209](#)

## 데이터 마스킹 변환 작성

데이터 마스킹 변환은 Developer tool에서 작성합니다.

데이터 마스킹 변환을 작성하기 전에 소스를 작성합니다. 플랫폼 파일 또는 관계형 데이터베이스 테이블을 실제 데이터 개체로 가져옵니다.

1. **개체 탐색기** 보기에서 프로젝트나 폴더를 선택합니다.
2. **파일 > 새로 만들기 > 변환**을 클릭합니다.  
    **새로 만들기** 대화 상자가 나타납니다.
3. 데이터 마스킹 변환을 선택합니다.
4. **다음**을 클릭합니다.
5. 변환 이름을 입력합니다.
6. **마침**을 클릭합니다.  
    변환이 편집기에 표시됩니다.

## 포트 정의

**개요** 보기에서 데이터 마스킹의 입력 포트를 추가합니다. 입력 포트를 작성하면 Developer tool이 기본적으로 해당하는 출력 포트를 작성합니다. 출력 포트의 이름은 입력 포트와 동일합니다.

1. **개요** 보기에서 **새로 만들기**를 클릭하여 포트를 추가합니다.
2. 열의 데이터 유형, 전체 자릿수 및 배율을 구성합니다.  
열에 대한 마스킹 규칙을 정의하기 전에 열의 데이터 유형을 먼저 구성해야 합니다.
3. 포트에 대한 데이터 마스킹을 구성하려면 **개요** 보기의 마스킹 유형 열에서 화살표를 클릭합니다.

## 각 포트에 대한 데이터 마스킹 구성

데이터 마스킹 대화 상자에서 포트에 대한 마스킹 기술 및 해당하는 마스킹 규칙을 선택합니다. 데이터 마스킹 대화 상자는 **포트** 탭에서 데이터 마스킹 열을 클릭하면 표시됩니다.

1. 선택한 포트에 대해 마스킹을 구성하려면 **마스킹 적용**을 활성화합니다.  
마스킹하는 포트의 데이터 유형에 따라 Developer tool이 사용할 수 있는 마스킹 기술 목록을 표시합니다.
2. 목록에서 마스킹 기술을 선택합니다.  
선택한 마스킹 기술에 따라 서로 다른 마스킹 규칙이 표시됩니다. 일부 특수 마스크 형식의 경우 마스킹 없음 규칙을 구성해야 합니다.
3. 마스킹 규칙을 구성합니다.
4. **확인**을 클릭하여 포트에 대한 데이터 마스킹 구성을 적용합니다.  
포트에 대한 데이터 마스킹을 정의하면 Developer tool이 **out-<포트 이름>**이라는 이름의 출력 포트를 작성합니다. **<포트 이름>**은 입력 포트의 이름과 동일합니다. 데이터 마스킹 변환이 마스킹된 데이터를 **out-<포트 이름>** 포트에 반환합니다.

## 마스킹된 데이터 미리보기

**데이터 뷰어**에서 데이터 마스킹 변환 결과를 확인할 때 마스킹된 데이터를 원래 데이터와 비교할 수 있습니다.

1. 데이터 마스킹 변환 포트와 마스킹 규칙을 구성한 후에는 실제 데이터 개체 소스 및 데이터 마스킹 변환을 포함하는 매핑을 생성합니다.
2. 소스를 데이터 마스킹 변환에 연결합니다.
3. 소스의 데이터가 데이터 통합 서비스에서 액세스할 수 있는 공유 위치에 있는지 확인합니다.
4. 데이터 마스킹 변환을 클릭하여 매핑에서 해당 변환을 선택합니다.
5. **데이터 뷰어**를 클릭하고 **실행**을 클릭합니다.

Developer tool에 모든 데이터 마스킹 변환 출력 포트의 데이터가 표시됩니다. **출력** 접두사가 있는 포트에는 마스킹된 데이터가 포함되어 있습니다. **데이터 뷰어** 보기에서 마스킹된 데이터를 원래 데이터와 비교할 수 있습니다.

## 데이터 마스킹 변환의 런타임 속성

데이터 마스킹 변환의 런타임 속성을 구성하여 성능을 개선할 수 있습니다.

다음 런타임 속성을 구성합니다.

## 캐시 크기

기본 메모리의 사전 캐시 크기입니다. 메모리 크기를 늘리면 성능이 개선됩니다. 최소 권장 크기는 32MB(100,000레코드)입니다. Default is 8 MB.

## 캐시 디렉터리

사전 캐시의 위치입니다. 디렉터리에 대한 쓰기 권한이 있어야 합니다. 기본값은 CacheDir입니다.

## 공유 저장소 테이블

데이터 마스킹 변환의 인스턴스가 저장소 테이블을 공유할 수 있도록 합니다. 데이터 마스킹 변환의 인스턴스 2개가 데이터베이스 연결, 시드 값 및 로컬에 대해 동일한 사전 열을 사용하는 경우 공유 저장소 테이블을 활성화하십시오. 동일한 데이터 마스킹 변환의 포트 2개가 연결, 시드 및 로컬에 대해 동일한 사전 열을 사용하는 경우에도 공유 저장소 테이블을 활성화할 수 있습니다. 데이터 마스킹 변환 또는 포트가 사전 열을 공유하지 않는 경우에는 공유 저장소 테이블을 비활성화합니다. 기본값이 비활성화됩니다.

## 저장소 커밋 간격

한 번에 저장소 테이블에 커밋할 행의 수입니다. 값을 높이면 성능이 개선됩니다. 공유 저장소 테이블을 구성하지 않는 경우 커밋 간격을 구성하십시오. 기본값은 100,000입니다.

## 저장소 암호화

IDM\_SUBSTITUTION\_STORAGE 및 IDM\_EXPRESSION\_STORAGE와 같은 저장소 테이블을 암호화합니다. 저장소 암호화 속성을 활성화하기 전에 저장소 테이블의 데이터가 암호화되었는지 확인하십시오. 저장소 테이블을 암호화하지 않으려는 경우 이 옵션을 선택 취소합니다. 기본값이 비활성화됩니다.

## 저장소 암호화 키

데이터 마스킹 변환이 저장소 암호화 키를 바탕으로 저장소를 암호화합니다. 동일한 데이터 마스킹 변환 인스턴스의 각 세션 실행에 대해서도 동일한 암호화 키를 사용합니다.

## 대체 사전 소유자 이름

대체 마스킹 유형을 선택한 경우 대체 사전 테이블의 소유자 이름입니다. 데이터베이스 연결에 지정된 데이터베이스 사용자가 세션에서 대체 사전 테이블의 소유자가 아닌 경우 테이블 소유자를 지정해야 합니다.

## 저장소 소유자 이름

반복 가능한 식 또는 고유하게 반복 가능한 대체 마스킹 유형을 선택한 경우 IDM\_SUBSTITUTION\_STORAGE 또는 IDM\_EXPRESSION\_STORAGE에 대한 테이블 소유자의 이름입니다.

# 데이터 마스킹 예제

한 개발자가 고객 응용 프로그램에 대한 테스트 데이터를 작성하려고 합니다. 테스트 데이터에는 다른 개발자가 회사의 개발 환경에서 액세스할 수 있는 실제와 같은 고객 데이터가 포함되어야 합니다.

이 개발자는 고객 ID, 신용 카드 번호 및 소득과 같은 고객 데이터를 마스킹된 데이터로 반환하는 데이터 서비스를 작성합니다. 매핑에는 고객 데이터를 변환하는 데이터 마스킹 변환이 포함됩니다.

다음 그림은 매핑을 보여 줍니다.





매핑에는 다음 변환이 포함됩니다.

- **Read\_Customer\_Data.** 고객의 신용 카드 및 소득 정보를 포함합니다.
- **Customer\_Data\_Masking** 변환. **FirstName** 및 **LastName**을 제외한 모든 열을 마스킹합니다. 이 데이터 마스킹 변환은 마스킹된 열을 대상에 전달합니다.
- **Customer\_TestData.** 마스킹된 고객 데이터를 수신하는 출력 변환입니다.

## Read\_Customer 데이터

고객 데이터에는 다음과 같은 열이 포함됩니다.

열	데이터 유형
CustomerID	정수
LastName	문자열
FirstName	문자열
CreditCard	문자열
Income	정수
Join_Date	Datetime (MM/DD/YYYY)

다음 테이블에는 샘플 고객 데이터가 나와 있습니다

CustomerID	LastName	FirstName	CreditCard	Income	JoinDate
0095	Bergeron	Barbara	4539-1686-3069-3957	12000	12/31/1999
0102	Brosseau	Derrick	5545-4091-5232-8948	4000	03/03/2011
0105	Anderson	Lauren	1234-5678-9012-3456	5000	04/03/2009
0106	Boonstra	Pauline	4217-9981-5613-6588	2000	07/07/2007
0107	Chan	Brian	4533-3156-8865-3156	4500	06/18/1995

## 고객 데이터 마스킹 변환

이 데이터 마스킹 변환은 고객 행에서 이름과 성을 제외한 모든 열을 마스킹합니다.

이 데이터 마스킹 변환은 다음 유형의 마스킹을 수행합니다.

- 키 마스킹
- 무작위 마스킹
- 신용 카드 마스킹

다음 표에는 이 데이터 마스킹 변환의 각 포트에 대한 마스킹 규칙이 설명되어 있습니다.

입력 포트	마스킹 유형	마스킹 규칙	설명
CustomerID	키	시드는 934입니다. 고객 ID에 지정된 마스크 형식은 없습니다. 결과 문자열 대체 문자는 1234567890입니다.	CustomerID 마스크는 고정입니다. 마스킹된 고객 ID에는 숫자가 포함됩니다.
LastName	마스킹 없음	-	-
FirstName	마스킹 없음	-	-
CreditCard	CreditCard	-	이 데이터 마스킹 변환은 신용 카드 번호를 유효한 체크섬을 포함하는 다른 번호로 마스킹합니다.
Income	무작위	블러링 백분율 하한=1 상한=10	마스킹된 소득은 소스 소득의 10%에 포함됩니다.
JoinDate	무작위	블러링 단위=연도 하한=5 상한=5	마스킹된 날짜는 원래 날짜에서 5년 내에 포함됩니다.

## 고객 테스트 데이터 결과

Customer\_TestData 변환은 이 데이터 마스킹 변환으로부터 실제와 같은 고객 데이터를 수신합니다.

Customer\_TestData 대상이 수신하는 데이터는 다음과 같습니다.

out-CustomerID	out-LastName	outFirstName	out-CreditCard	out-Income	out-JoinDate
3954	Bergeron	Barbara	4539-1625-5074-4106	11500	03/22/2001
3962	Brosseau	Derrick	5545-4042-8767-5974	4300	04/17/2007
3964	Anderson	Lauren	1234-5687-2487-9053	5433	09/13/2006
3965	Boonstra	Pauline	4217-9935-7437-4879	1820	02/03/2010
3966	Chan	Brian	4533-3143-4061-8001	4811	10/30/2000

이 소득은 원래 소득의 10%에 포함됩니다. 입사 날짜는 원래 날짜에서 5년 내에 포함됩니다.

## 데이터 마스킹 변환의 고급 속성

데이터 마스킹 변환 데이터에 대한 데이터 통합 서비스의 처리 방식을 결정하는 데 도움이 되는 속성을 구성할 수 있습니다.

로그의 추적 수준을 구성할 수 있습니다.

고급 탭에서 다음 속성을 구성합니다.

## 추적 수준

이 변환에 대해 로그에 표시되는 세부 정보의 양입니다. 간단, 보통, 자세한 정보 표시 초기화 또는 자세한 정보 표시 데이터를 선택할 수 있습니다. 기본값은 보통입니다.

# 데이터 마스킹 변환 - 비원시 환경

비원시 환경에서 데이터 마스킹 변환의 처리는 변환을 실행하는 엔진에 따라 다릅니다.

다음과 같은 비원시 런타임 엔진에 대한 지원을 고려하십시오.

- **Blaze** 엔진. 제한적으로 지원됩니다.
- **Spark** 엔진. 제한적으로 지원됩니다.
- **Databricks Spark** 엔진. 지원되지 않습니다.

## 데이터 마스킹 변환 - Blaze 엔진

데이터 마스킹 변환은 다음과 같은 제한과 함께 지원됩니다.

다음과 같은 상황에서는 매핑 유효성 검사가 실패합니다.

- 변환이 반복 가능한 식 마스킹에 대해 구성되어 있습니다.
- 변환이 반복 가능한 고유 대체 마스킹에 대해 구성되어 있습니다.

**Blaze** 엔진에서 다음과 같은 마스킹 기술을 사용할 수 있습니다.

신용 카드

전자 메일

식

IP 주소

키

전화

무작위

SIN

SSN

토큰화

URL

무작위 대체

반복 가능한 대체

무작위 대체에 종속

반복 가능한 대체에 종속

## 데이터 마스킹 변환 - Spark 엔진

데이터 마스킹 변환은 다음과 같은 제한과 함께 지원됩니다.

다음과 같은 상황에서는 매핑 유효성 검사가 실패합니다.

- 변환이 반복 가능한 식 마스킹에 대해 구성되어 있습니다.
- 변환이 반복 가능한 고유 대체 마스킹에 대해 구성되어 있습니다.

Spark 엔진에서 다음과 같은 마스킹 기술을 사용할 수 있습니다.

신용 카드

전자 메일

식

IP 주소

키

전화

무작위

SIN

SSN

토큰화

URL

무작위 대체

반복 가능한 대체

무작위 대체에 종속

반복 가능한 대체에 종속

데이터 마스킹 변환의 성능을 최적화하려면 다음과 같은 Hadoop 연결에서 Spark 엔진 구성 속성을 구성합니다.

`spark.executor.cores`

각 실행자 프로세스가 Spark 엔진에서 태스크릿을 실행할 때 사용하는 코어 수를 나타냅니다.

`spark.executor.cores=1`로 설정합니다.

`spark.executor.instances`

각 실행자 프로세스가 Spark 엔진에서 태스크릿을 실행할 때 사용하는 인스턴스 수를 나타냅니다.

`spark.executor.instances=1`로 설정합니다.

`spark.executor.memory`

각 실행자 프로세스가 Spark 엔진에서 태스크릿을 실행할 때 사용하는 메모리 양을 나타냅니다.

`spark.executor.memory=3G`로 설정합니다.

## 제 13 장

# 데이터 프로세서 변환

이 장에 포함된 항목:

- [데이터 프로세서 변환 개요, 217](#)
- [데이터 프로세서 변환 보기, 218](#)
- [데이터 프로세서 변환 포트, 219](#)
- [시작 구성 요소, 221](#)
- [참조, 221](#)
- [데이터 프로세서 변환 설정, 222](#)
- [이벤트, 228](#)
- [로그, 229](#)
- [데이터 프로세서 변환 개발, 231](#)
- [데이터 프로세서 변환 가져오기 및 내보내기, 235](#)
- [데이터 프로세서 변환 비원시 환경 내, 237](#)

## 데이터 프로세서 변환 개요

데이터 프로세서 변환은 매핑에서 구조화되지 않은 파일 형식과 반구조화된 파일 형식을 처리합니다. 이 변환을 구성하여 메시징 형식, HTML 페이지, XML, JSON 및 PDF 문서를 처리하십시오. ACORD, HIPAA, HL7, EDI-X12, EDIFACT, SWIFT 등과 같은 구조화된 형식을 변환할 수도 있습니다.

매핑에서는 데이터 프로세서 변환을 사용하여 문서를 한 형식에서 다른 형식으로 변경합니다. 데이터 프로세서 변환은 매핑에 속한 모든 형식의 파일을 처리합니다. 데이터 프로세서 변환을 작성할 때 데이터를 변환하는 구성 요소를 정의합니다.

한 데이터 프로세서 변환이 데이터를 처리하는 여러 구성 요소를 포함할 수 있습니다. 각 구성 요소는 다른 구성 요소를 포함할 수 있습니다.

예를 들어 Microsoft Word 파일 형식으로 고객 송장을 받을 수 있습니다. 이 경우 각 Word 파일에서 데이터를 구문 분석하는 데이터 프로세서 변환을 구성합니다. 고객 데이터를 Customer 테이블로 추출하고, 주문 정보를 Orders 테이블로 추출합니다.

데이터 프로세서 변환을 작성할 때 XMap, 스크립트 또는 라이브러리를 정의합니다. XMap은 입력 계층 파일을 다른 구조의 출력 계층 파일로 변환합니다. 라이브러리는 산업 메시지 형식을 계층 구조의 XML 문서로 변환하거나 XML에서 산업 표준 형식으로 변환합니다. 스크립트는 소스 문서를 계층적 형식으로 구문 분석하거나, 계층적 형식을 다른 파일 형식으로 변환하거나, 계층적 문서를 다른 계층적 형식으로 매핑할 수 있습니다.

데이터 프로세서 변환 IntelliScript 편집기에서 스크립트를 정의합니다. 다음 유형의 스크립트를 정의할 수 있습니다.

- 파서. 소스 문서를 XML로 변환합니다. 파서의 출력은 항상 XML입니다. 입력은 텍스트, HTML, Word, PDF, HL7 등과 같은 형식일 수 있습니다.
- 직렬 변환기. XML 파일을 원하는 형식의 출력 문서로 변환합니다. 직렬 변환기의 출력은 텍스트 문서, HTML 문서, PDF 등과 같은 형식일 수 있습니다.
- 매퍼. XML 소스 문서를 다른 XML 구조나 스키마로 변환합니다. XMap에서와 같이 동일한 XML 문서를 변환할 수 있습니다.
- 변환기. 모든 형식의 데이터를 수정합니다. 텍스트를 추가, 제거, 변환 또는 변경합니다. 파서, 매퍼 또는 직렬 변환기와 함께 변환기를 사용합니다. 변환기를 독립 실행형 구성 요소로 실행할 수도 있습니다.
- 스트리머. 수 기가바이트 크기의 데이터 스트림과 같은 큰 입력 문서를 세그먼트로 분할합니다. 스트리머는 HIPAA 또는 EDI 파일과 같이 여러 메시지나 레코드를 포함하는 문서를 처리합니다.

자세한 내용은 *Data Transformation 사용자 가이드*를 참조하십시오.

## 데이터 프로세서 변환 보기

데이터 프로세서 변환에는 Developer tool에서 변환을 구성하고 실행할 때 액세스하는 다양한 보기가 있습니다.

일부 데이터 프로세서 변환 보기는 기본적으로 Developer tool에 나타나지 않습니다. 변환의 보기를 변경하려면 **창 > 보기 표시 > 기타 > Informatica**를 클릭합니다. 표시하려는 보기를 선택합니다.

데이터 프로세서 변환에는 다음과 같은 고정된 보기가 있습니다.

### 개요 보기

포트를 구성하고 시작 구성 요소를 정의합니다.

### 참조 보기

변환에서 스키마를 추가하거나 제거합니다.

### 설정 보기

인코딩, 출력 제어 및 XML 생성에 대한 변환 설정을 구성합니다.

### 개체 보기

변환에서 스크립트, XMap 및 라이브러리 개체를 추가, 수정 또는 삭제합니다.

데이터 프로세서 변환에서 다음 보기에 액세스할 수도 있습니다.

### 데이터 프로세서 16진수 소스 보기

입력 문서를 16진수 형식으로 표시합니다.

### 데이터 프로세서 이벤트 보기

Developer tool에서 변환을 실행할 때 발생하는 이벤트에 대한 정보를 표시합니다. 초기화, 실행 및 요약 이벤트를 표시합니다.

### 스크립트 도움말 보기

스크립트 편집기의 상황에 맞는 도움말을 표시합니다.

### 데이터 뷰어 보기

예제 입력 데이터를 보고, 변환을 실행하고, 출력 결과를 봅니다.

# 데이터 프로세서 변환 포트

변환 개요 보기에서 데이터 프로세서 변환 포트를 정의합니다.

데이터 프로세서 변환은 복잡한 파일 관독기의 파일, 버퍼 또는 스트리밍되는 버퍼에서 입력을 읽을 수 있습니다. 데이터 프로세서 변환은 복잡한 파일 관독기의 파일, 버퍼 또는 스트리밍되는 버퍼에서 입력을 읽을 수 있습니다. 데이터 프로세서 변환은 파일, 버퍼 또는 스트리밍되는 버퍼에서 입력을 읽을 수 있습니다. 전체 파일을 한 번에 읽기 위한 버퍼로 플랫폼 파일 관독기를 사용할 수 있습니다. 또한 데이터베이스로부터 입력 파일을 읽을 수 있습니다.

작성하는 출력 포트는 변환으로부터 문자열, 복잡한 파일 또는 관계형 데이터 행 중 무엇을 반환할지에 따라 달라집니다.

## 데이터 프로세서 변환 입력 포트

데이터 프로세서 변환을 작성하면 **Developer tool**이 기본 입력 포트를 작성합니다. 스크립트 시작 구성 요소에서 추가 입력 포트를 정의할 경우 **Developer tool**이 변환에서 추가 입력 포트를 작성합니다.

입력 유형은 데이터 통합 서비스가 데이터 프로세서 변환으로 전달하는 데이터의 유형을 결정합니다. 입력 유형은 입력이 데이터인지, 아니면 소스 파일 경로인지를 결정합니다.

다음 입력 유형 중 하나를 구성하십시오.

### 버퍼

데이터 프로세서 변환이 입력 포트에서 소스 데이터의 행을 수신합니다. 플랫폼 파일이나 **Informatica** 변환으로부터 데이터를 수신하는 변환을 구성할 경우 버퍼 입력 유형을 사용합니다.

### 파일

데이터 프로세서 변환이 입력 포트에서 소스 파일 경로를 수신합니다. 데이터 프로세서 시작 구성 요소가 소스 파일을 엽니다. **Microsoft Excel** 또는 **Microsoft Word** 파일 같은 이진 파일을 구문 분석하려면 파일 입력 유형을 사용합니다. 버퍼 입력 포트를 처리하려면 많은 양의 시스템 메모리가 필요할 수 있는 대용량 파일에도 파일 입력 유형을 사용할 수 있습니다.

### 서비스 매개 변수

데이터 프로세서 변환이 서비스 매개 변수 포트의 변수에 적용할 값을 수신합니다. 변수를 사용하여 입력 데이터를 수신하도록 선택한 경우 **Developer tool**이 각 변수에 대해 서비스 매개 변수 포트를 작성합니다.

### Output\_Filename

행 데이터 대신 파일 이름을 반환하도록 기본 출력 포트를 구성할 경우 **Developer tool**이 **Output\_Filename** 포트를 작성합니다. 매핑에서 **Output\_Filename** 포트에 파일 이름을 전달할 수 있습니다.

입력 포트를 정의할 때 포트의 예제 입력 파일 위치를 정의할 수 있습니다. 예제 입력 파일은 입력 파일의 작은 샘플입니다. 스크립트를 작성할 때 예제 입력 파일을 참조합니다. 또한 **데이터 뷰어** 보기에서 변환을 테스트할 때에도 예제 입력 파일을 사용합니다. **입력 위치** 필드에서 예제 입력 파일을 정의합니다.

## 서비스 매개 변수 포트

변수의 값을 수신하는 입력 포트를 작성할 수 있습니다. 변수는 문자열, 날짜 또는 숫자 등 모든 데이터 유형을 포함할 수 있습니다. 또한 변수는 소스 문서의 위치도 포함할 수 있습니다. 데이터 프로세서 구성 요소에서 변수를 참조할 수 있습니다.

변수에 대한 입력 포트를 작성하면 **Developer tool**이 선택할 수 있는 변수 목록을 표시합니다.

## 서비스 매개 변수 포트 작성

변수의 값을 수신하는 입력 포트를 작성할 수 있습니다. 또한 변수에서 작성한 포트를 제거할 수 있습니다.

1. 데이터 프로세서 변환 **개요** 보기를 엽니다.
2. **선택**을 클릭합니다.  
Developer tool에 변수 목록이 표시되며, 어떤 변수에 이미 포트가 있는지 나타냅니다.
3. 하나 이상의 변수를 선택합니다.  
Developer tool이 선택한 각 변수에 대해 버퍼 입력 포트를 작성합니다. 이 포트를 수정할 수 없습니다.
4. 변수에서 작성한 포트를 제거하려면 변수 목록에서 해당 선택 항목을 비활성화하십시오. 선택 항목을 비활성화하면 Developer tool이 해당 입력 포트를 제거합니다.

## 데이터 프로세서 변환 출력 포트

데이터 프로세서 변환에는 기본적으로 출력 포트 하나가 있습니다. 스크립트에서 추가 출력 포트를 정의하면 Developer tool이 데이터 프로세서 변환에 포트를 추가합니다. 관계형 데이터를 반환하도록 변환을 구성하는 경우 포트 그룹을 작성할 수 있습니다. 또한 서비스 매개 변수 포트와 통과 포트를 작성할 수 있습니다.

### 기본 출력 포트

데이터 프로세서 변환에는 기본적으로 출력 포트 하나가 있습니다. 관계형 출력을 작성할 때 기본 출력 포트 대신 관련 출력 포트의 그룹을 정의할 수 있습니다. 스크립트 구성 요소에서 추가 출력 포트를 정의하면 Developer tool이 변환에 추가적인 출력 포트를 추가합니다.

기본 출력 포트에 대해 다음 출력 유형 중 하나를 구성합니다.

#### 버퍼

데이터 프로세서 변환이 출력 포트를 통해 XML을 반환합니다. 데이터 프로세서 변환에서 XML을 다른 XML 문서에 매핑하거나 문서를 구문 분석할 때 버퍼 파일 유형을 선택합니다.

#### 파일

데이터 통합 서비스는 각 소스 인스턴스 또는 행에 대한 출력 포트에서 출력 파일 이름을 반환합니다. 데이터 프로세서 변환 구성 요소는 데이터 프로세서 변환 출력 포트를 통해 데이터를 반환하는 대신 출력 파일을 씁니다.

파일 출력 포트를 선택하면 Developer tool이 Output\_Filename 입력 포트를 작성합니다. 이 출력 파일 이름 포트로 파일 이름을 전달할 수 있습니다. 데이터 프로세서 변환은 이 포트에서 수신한 이름으로 출력 파일을 작성합니다.

출력 파일 이름이 비어 있으면 데이터 통합 서비스가 행 오류를 반환합니다. 오류가 발생하면 데이터 통합 서비스는 출력 포트에 null 값을 쓰고 행 오류를 반환합니다.

XML을 PDF 파일이나 Microsoft Excel 파일 같은 이진 데이터 파일로 변환할 때 파일 출력 유형을 선택합니다.

## 통과 포트

데이터 프로세서 변환에 통과 포트를 구성할 수 있습니다. 통과 포트는 입력 데이터를 수신한 후 데이터 변경 없이 매핑에 동일한 데이터를 반환하는 입력 및 출력 포트입니다.

매핑에 속한 데이터 프로세서 변환 인스턴스에서 통과 포트를 구성할 수 있습니다.

통과 포트를 추가하려면 매핑의 다른 변환에서 포트를 끌어다 놓습니다. **속성** 보기의 **포트** 탭에서 포트를 추가할 수도 있습니다. 통과 포트를 추가하려면 **새로 만들기**를 클릭합니다.



**참고:** 관계형 입력 및 계층 출력이 있는 데이터 프로세서 변환에 통과 포트를 추가할 경우 관계형 구조의 루트 그룹에 포트를 추가하십시오.

데이터 프로세서 변환에는 사용자 지정 데이터 유형인 통과 포트를 포함할 수 있습니다.

## 시작 구성 요소

시작 구성 요소는 데이터 프로세서 변환에서 처리를 시작하는 구성 요소입니다. **개요** 보기에서 시작 구성 요소를 구성합니다.

한 데이터 프로세서 변환이 데이터를 처리하는 여러 구성 요소를 포함할 수 있습니다. 각 구성 요소는 다른 구성 요소를 포함할 수 있습니다. 변환의 진입점이 되는 구성 요소를 식별해야 합니다.

데이터 프로세서 변환에서 시작 구성 요소를 구성할 경우 **XMap**, **Library** 또는 스크립트 구성 요소를 시작 구성 요소로 선택할 수 있습니다. **Script**의 경우 다음과 같은 유형의 구성 요소 중 하나를 선택할 수 있습니다.

- 파서. 소스 문서를 XML로 변환합니다. 입력은 텍스트, HTML, Word, PDF, HL7 등과 같은 형식일 수 있습니다.
- 매퍼. XML 소스 문서를 다른 XML 구조나 스키마로 변환합니다.
- 직렬 변환기. XML 파일을 원하는 형식의 출력 문서로 변환합니다.
- 스트리머. 수 기가바이트 크기의 데이터 스트림과 같은 큰 입력 문서를 세그먼트로 분할합니다.
- 변환기. 모든 형식의 데이터를 수정합니다. 텍스트를 추가, 제거, 변환 또는 변경합니다. 파서, 매퍼 또는 직렬 변환기와 함께 변환기를 사용합니다. 변환기를 독립 실행형 구성 요소로 실행할 수도 있습니다.

**참고:** 시작 구성 요소가 XMap 또는 Library가 아닐 경우 **개요** 보기 대신 **Script**에서 시작 구성 요소를 구성할 수도 있습니다.

## 참조

참조로 사용할 스키마 또는 맵렛을 선택하여 스키마 또는 맵렛 참조와 같은 변환 참조를 정의할 수 있습니다. 일부 데이터 프로세서 변환에는 변환의 관련 구성 요소에 대한 입력 또는 출력 계층을 정의하는 계층 스키마가 필요합니다. 변환에서 스키마를 사용하려면 변환에 대해 스키마 참조를 정의하십시오. **RunMapplet**이라는 이름의 특수화된 작업을 사용하여 데이터 프로세서 변환에서 맵렛을 호출할 수도 있습니다. 맵렛을 호출하려면 우선 변환에 대해 맵렛 참조를 정의해야 합니다.

변환 **참조** 보기에서 스키마 또는 맵렛 참조와 같은 변환 참조를 정의할 수 있습니다.

### 스키마 참조

데이터 프로세서 변환은 모델 리포지토리의 스키마 개체를 참조합니다. 변환을 작성하기 전에 스키마 개체가 리포지토리에 있을 수 있습니다. 또한 변환 **참조** 보기에서 스키마를 가져올 수도 있습니다.

스키마 인코딩은 직렬 변환기 또는 매퍼 개체의 입력 인코딩과 일치해야 합니다. 스키마 인코딩은 파서 또는 매퍼 개체의 출력 인코딩과 일치해야 합니다. 변환 **설정** 보기에서 작업 인코딩을 구성합니다.

스키마는 다른 스키마를 참조할 수 있습니다. **Developer tool**은 데이터 프로세서 변환이 참조하는 각 스키마의 네임스페이스와 접두사를 표시합니다. 빈 네임스페이스의 여러 스키마를 참조할 경우 변환이 올바르게 작동하지 않습니다.

## 맷렛 참조

RunMapplet 작업으로 데이터 프로세서 변환에서 맷렛을 호출할 수 있습니다. 데이터 프로세서 변환 구성 요소에 RunMapplet 작업을 추가하기 전에 우선 호출하려는 맷렛에 대한 참조를 정의해야 합니다.

# 데이터 프로세서 변환 설정

데이터 프로세서 변환 **설정** 보기에서 코드 페이지, XML 처리 옵션 및 로그 설정을 구성합니다.

## 문자 인코딩

문자 인코딩은 한 언어 또는 언어 그룹의 문자를 16진수 코드로 매핑하는 것입니다.

스크립트를 작성할 때 입력 문서의 인코딩과 출력 문서의 인코딩을 정의합니다. IntelliScript 편집기에서 문자를 표시하는 방법과 데이터 프로세서 변환에서 문자를 처리하는 방법을 정의하려면 작업 인코딩을 정의합니다.

## 작업 인코딩

작업 인코딩은 메모리의 데이터에 대한 코드 페이지와 사용자 인터페이스 및 작업 파일에 나타나는 데이터에 대한 코드 페이지입니다. 데이터 프로세서 변환에서 참조하는 스키마의 인코딩과 호환되는 작업 인코딩을 선택해야 합니다.

다음 테이블에서는 작업 인코딩 설정을 보여 줍니다.

설정	설명
데이터 프로세서 기본 코드 페이지 사용	데이터 프로세서 변환의 기본 인코딩을 사용합니다.
기타	목록에서 인코딩을 선택합니다.
XML 특수 문자 인코딩	XML 특수 문자의 표현 방식을 결정합니다. <b>없음</b> 또는 XML을 선택할 수 있습니다. - <b>없음.</b> &amp; &lt; &gt; &quot; &apos;로 유지 XML 특수 문자에 대한 항목 참조를 텍스트로 해석합니다. 예를 들어 > 문자가 다음으로 나타납니다. &gt; 기본값은 없음입니다. - <b>XML. &amp; &lt; &gt; " '로 변환</b> XML 특수 문자에 대한 항목 참조를 일반 문자로 해석합니다. 예를 들어 &gt;가 다음 문자로 나타납니다. >

## 입력 인코딩

입력 인코딩은 입력 문서에서 문자 데이터가 인코딩되는 방법을 결정합니다. 스크립트에서 추가 입력 포트에 대한 인코딩을 구성할 수 있습니다.

다음 테이블에는 **입력** 영역의 인코딩 설정이 설명되어 있습니다.

설정	설명
입력 문서에 지정된 인코딩 사용	XML 문서의 인코딩 특성과 같이 소스 문서에 정의된 코드 페이지를 사용합니다. 소스 문서에 인코딩 사양이 없는 경우 데이터 프로세서 변환은 <b>설정</b> 보기의 인코딩 설정을 사용합니다.
작업 인코딩 사용	작업 인코딩과 동일한 인코딩을 사용합니다.
기타	드롭다운 목록에서 입력 인코딩을 선택합니다.
XML 특수 문자 인코딩	XML 특수 문자의 표현 방식을 결정합니다. <b>없음</b> 또는 XML을 선택할 수 있습니다. <ul style="list-style-type: none"> <li>- <b>없음.</b>  &amp; &lt; &gt; &amp;quot; &amp;apos;로 유지  XML 특수 문자에 대한 항목 참조를 텍스트로 해석합니다. 예를 들어 &gt; 문자가 다음으로 나타납니다.  &amp;gt;  기본값은 없음입니다.</li> <li>- XML. &amp; &lt; &gt; " '로 변환  XML 특수 문자에 대한 항목 참조를 일반 문자로 해석합니다. 예를 들어 &amp;gt;가 다음 문자로 나타납니다.  &gt;</li> </ul>
바이트 순서	입력 문서에서 다중 바이트 문자가 나타나는 방법을 설명합니다. 다음 옵션을 선택할 수 있습니다. <ul style="list-style-type: none"> <li>- Little-endian. 최하위 바이트가 먼저 나타납니다. 기본값입니다.</li> <li>- Big-endian. 최상위 바이트가 먼저 나타납니다.</li> <li>- 이진 변환 없음.</li> </ul>

## 출력 인코딩

출력 인코딩은 기본 출력 문서에서 문자 데이터가 인코딩되는 방법을 결정합니다.

다음 테이블에는 **출력** 영역의 인코딩 설정이 설명되어 있습니다.

설정	설명
작업 인코딩 사용	출력 인코딩이 작업 인코딩과 같습니다.
기타	사용자가 목록에서 출력 인코딩을 선택합니다.
XML 특수 문자 인코딩	XML 특수 문자의 표현 방식을 결정합니다. <b>없음</b> 또는 XML을 선택할 수 있습니다. <ul style="list-style-type: none"> <li>- <b>없음.</b>  &amp; &lt; &gt; &amp;quot; &amp;apos;로 유지  XML 특수 문자에 대한 항목 참조를 텍스트로 해석합니다. 예를 들어 &gt; 문자가 다음으로 나타납니다.  &amp;gt;  기본값입니다.</li> <li>- XML. &amp; &lt; &gt; " '로 변환  XML 특수 문자에 대한 항목 참조를 일반 문자로 해석합니다. 예를 들어 &amp;gt;가 다음 문자로 나타납니다.  &gt;</li> </ul>

설정	설명
입력 인코딩과 동일	출력 인코딩이 입력 인코딩과 같습니다.
바이트 순서	입력 문서에서 다중 바이트 문자가 나타나는 방법을 설명합니다. 다음 옵션을 선택할 수 있습니다. <ul style="list-style-type: none"> <li>- Little-endian. 최하위 바이트가 먼저 나타납니다. 기본값입니다.</li> <li>- Big-endian. 최상위 바이트가 먼저 나타납니다.</li> <li>- 이진 변환 없음.</li> </ul>

## 문자 인코딩에 대한 규칙 및 지침

인코딩 구성 시 다음 규칙과 지침을 따르십시오.

- 성능을 향상시키려면 작업 인코딩을 출력 문서와 동일한 인코딩으로 설정합니다.
- 입력 인코딩을 입력 문서와 동일한 인코딩으로 설정합니다.
- 출력 인코딩을 출력 문서와 동일한 인코딩으로 설정합니다.
- 다중 바이트 문자가 있는 언어인 경우 작업 인코딩을 UTF-8로 설정합니다. 입력 및 출력 인코딩의 경우 UTF-8 같은 유니코드 인코딩이나 Big5 또는 Shift\_JIS 같은 더블바이트 코드 페이지를 사용할 수 있습니다.

## 출력 설정

데이터 프로세서 변환이 이벤트 로그를 작성하고 출력 문서를 저장할지 여부를 제어하려면 출력 제어 설정을 구성합니다.

데이터 프로세서 변환이 디자인 타임 이벤트 로그에 쓰는 메시지 유형을 제어할 수 있습니다. 이벤트 로그와 함께 구문 분석된 입력 문서를 저장하면 **이벤트** 보기에서 오류가 발생한 컨텍스트를 볼 수 있습니다.

다음 테이블에는 **디자인 타임 이벤트** 영역의 설정이 설명되어 있습니다.

설정	설명
디자인 타임 이벤트 로깅	디자인 타임 이벤트 로그를 작성할지 여부를 결정합니다. 기본적으로 데이터 프로세서 변환은 디자인 타임 이벤트 로그에 알림, 경고 및 실패를 기록합니다. 다음 이벤트 유형을 제외할 수 있습니다. <ul style="list-style-type: none"> <li>- 알림</li> <li>- 경고</li> <li>- 실패</li> </ul>
구문 분석된 문서 저장	데이터 프로세서 변환이 구문 분석된 입력 문서를 저장하는 시점을 결정합니다. 다음 옵션을 선택할 수 있습니다. <ul style="list-style-type: none"> <li>- 항상</li> <li>- 사용 안 함</li> <li>- 실패 시</li> </ul> 기본값은 항상입니다.

다음 테이블에는 **런타임 이벤트** 영역의 설정이 설명되어 있습니다.

설정	설명
런타임 이벤트 로깅	매핑에서 변환을 실행할 때 이벤트 로그가 작성되는지 여부를 결정합니다. - 사용 안 함 - 실패 시 기본값은 사용 안 함입니다.

다음 테이블에는 **출력** 영역의 설정이 설명되어 있습니다.

설정	설명
자동 출력 비활성화	데이터 프로세서 변환이 출력을 표준 출력 파일에 쓸지 여부를 결정합니다. 다음과 같은 경우 표준 출력을 비활성화합니다. - 변환에서 출력 파일을 작성하기 전에 파서의 출력을 다른 구성 요소의 입력으로 전달합니다. - 출력 포트를 통해 데이터를 전달하는 대신 WriteValue 작업을 사용하여 스크립트에서 출력으로 직접 데이터를 씁니다.
값 압축 비활성화	데이터 프로세서 변환에서 값 압축을 사용하여 메모리 사용을 최적화할지 여부를 결정합니다. 중요: Informatica 글로벌 고객 지원 센터에서 값 압축을 비활성화하라고 요청하는 경우 이외에는 값 압축을 비활성화하지 마십시오.

다음 테이블에는 **이진 출력 포트 컬렉션 모드** 영역의 설정이 설명되어 있습니다. XML, Avro 또는 Parquet 출력이 있는 관계-계층형 변환의 이진 출력이나 Avro 또는 Parquet 출력이 있는 데이터 프로세서 변환 파서의 이진 출력에 다음 옵션 중 하나를 선택할 수 있습니다.

설정	설명
입력 행을 단일 출력으로 수집	데이터 프로세서 변환이 관계형 입력을 단일 이진 출력 포트에 누적할지 여부를 결정합니다.
크기 초과 시 출력 분할	명시된 최대 출력 크기를 기반으로 데이터 프로세서 변환이 출력을 청크로 분할할지 여부를 결정합니다.
각 행에 행 출력(수집 안 함)	데이터 프로세서 변환이 출력을 별도의 행에 전달하는지 여부를 결정합니다.

## 처리 설정

처리 설정은 데이터 프로세서 변환이 정의된 데이터 유형이 없는 요소를 처리하는 방식을 정의합니다. 이 설정은 스크립트에 영향을 줍니다. 그러나 XMap이 처리하는 요소에는 영향을 주지 않습니다.

다음 테이블에서는 스크립트의 XML 처리에 영향을 주는 처리 설정에 대해 설명합니다.

설정	설명
xs:string으로 처리	데이터 프로세서 변환이 유형이 없는 요소를 문자열로 처리합니다. XPath 선택 대화 상자에서 요소 또는 특성이 단일 노드로 표시됩니다.
xs:anyType으로 처리	데이터 프로세서 변환이 유형이 없는 요소를 anyType으로 처리합니다. XPath 선택 대화 상자에서 요소 또는 특성이 노드의 트리노드로 표시됩니다. 하나의 노드는 xs:string 유형이고 모든 명명된 복합 데이터 유형은 트리 노드로 표시됩니다.

다음 테이블에서는 **Streamer** 처리에 영향을 주는 처리 설정에 대해 설명합니다.

설정	설명
스트리머 청크 크기	이 설정은 Streamer가 입력 파일 스트림에서 매번 읽는 데이터의 양을 정의합니다. 데이터 프로세서 변환이 이 설정을 파일 입력이 있는 Streamer에 적용합니다.

다음 테이블에는 계층-관계형 변환 처리에 영향을 주는 처리 설정이 설명되어 있습니다.

설정	설명
엄격한 유효성 검사 적용	이 설정은 데이터 프로세서 변환이 계층형 입력에 대해 엄격한 유효성 검사를 수행하는지 여부를 결정합니다. 엄격한 유효성 검사가 적용되면 계층 입력 파일이 해당 스키마를 엄격하게 준수해야 합니다. 이 옵션은 데이터 프로세서 모드를 관계형 출력에 대한 출력 포트를 생성하는 <b>출력 매핑</b> 으로 설정할 때 적용할 수 있습니다. 이 옵션은 버전 10.2.1 이전 버전의 JSON 입력이 포함된 매핑에 적용되지 않습니다.
XML 입력 정규화	이 설정은 데이터 프로세서 변환이 XML 입력을 정규화할 것인지 결정합니다. 기본적으로 변환은 XML 입력에 대한 정규화를 수행합니다. 일부 경우 성능 향상을 위해 자동 정규화를 건너뛰도록 선택할 수 있습니다.

## XMap 설정

**XMap** 설정은 데이터 프로세서 변환이 출력 요소로 변환되지 않는 XMap 입력 요소를 처리하는 방법을 정의합니다. 읽지 않은 요소는 **XMap\_Unread\_Input\_Values**라는 전용 포트에 전달됩니다. 이 설정은 XMap이 시작 구성 요소로 선택된 경우에만 적용됩니다. 그러나 XMap이 처리하는 요소에는 영향을 주지 않습니다.

읽지 않은 XMap 요소를 전용 포트에 전달하려면 **추가 출력 포트에 읽지 않은 요소 쓰기** 설정을 활성화합니다.

## XML 출력 구성

XML 생성 설정은 XML 출력 문서의 특성을 정의합니다.

다음 테이블에서는 **스키마 제목** 영역의 XML 생성 설정에 대해 설명합니다.

설정	설명
스키마 위치	기본 출력 문서의 루트 요소에 대한 schemaLocation을 정의합니다.
네임스페이스 스키마 위치 없음	기본 출력 문서의 루트 요소에 대한 noNamespaceSchemaLocation 특성을 정의합니다.

XML 출력 모드 설정을 구성하여 데이터 프로세서 변환이 입력 XML 문서에서 누락된 요소 또는 특성을 처리하는 방법을 지정합니다. 다음 테이블에서는 **XML 출력 모드** 영역의 XML 생성 설정에 대해 설명합니다.

설정	설명
있는 그대로	빈 요소를 추가하거나 제거하지 않습니다. 기본값은 활성화됨입니다.
전체	출력 스키마에 정의된 모든 필수 요소와 선택적 요소를 출력에 씁니다. 콘텐츠가 없는 요소는 빈 요소로 씁니다.
압축	출력에서 빈 요소를 제거합니다. <b>요소에 추가</b> 가 활성화된 경우 데이터 프로세서 변환은 선택적 요소만 제거합니다. <b>요소에 추가</b> 가 비활성화된 경우 데이터 프로세서 변환은 모든 요소를 제거합니다. XML 출력이 올바르지 않을 수 있습니다.

다음 테이블에서는 **필요한 노드의 기본값** 영역의 XML 생성 설정에 대해 설명합니다.

설정	설명
요소에 추가	출력 스키마가 필수 요소에 대한 기본값을 정의하는 경우 출력에 요소와 기본값이 포함됩니다. 기본값은 활성화됨입니다.
특성에 추가	출력 스키마가 필수 특성에 대한 기본값을 정의하는 경우 출력에 특성과 해당 기본값이 포함됩니다. 기본값은 활성화됨입니다.
추가된 값 유효성 검사	데이터 프로세서 변환이 전체 모드 출력에서 추가된 빈 요소의 유효성을 검사할지 여부를 지정합니다. 기본값이 비활성화됩니다. <b>추가된 값 유효성 검사</b> 가 활성화되고 스키마가 빈 요소를 허용하지 않을 경우 XML 출력이 올바르지 않을 수 있습니다.

다음 테이블에서는 **처리 지침** 영역의 XML 생성 설정에 대해 설명합니다.

설정	설명
XML 처리 지침 추가	출력 문서의 문자 인코딩 및 XML 버전을 정의합니다. 기본값은 선택됨입니다.
XML 버전	XML 버전을 정의합니다. XML 버전 설정에는 다음과 같은 옵션이 있습니다. - 1.0 - 1.1 기본값은 1.0입니다.
인코딩	처리 지침에 지정되는 문자 인코딩을 정의합니다. 인코딩 설정에는 다음과 같은 옵션이 있습니다. - 출력 인코딩과 동일. 처리 지침의 출력 인코딩이 데이터 프로세서 변환 설정에 정의된 출력 인코딩과 동일합니다. - 사용자 지정. 처리 지침의 출력 인코딩을 정의합니다. 사용자가 필드에 값을 입력합니다.
사용자 지정 처리 지침 추가	출력 문서에 다른 처리 지침을 추가합니다. 출력 문서에 나타나는 대로 정확하게 처리 지침을 입력합니다. 기본값이 비활성화됩니다.

다음 테이블에서는 **XML 루트** 영역의 XML 생성 설정에 대해 설명합니다.

설정	설명
XML 루트 요소 추가	출력 문서에 루트 요소를 추가합니다. 출력 스키마에 정의된 루트 요소가 출력 문서에 두 번 이상 나타나는 경우 이 옵션을 사용합니다. 기본값이 비활성화됩니다.
루트 요소 이름	출력 문서에 추가할 루트 요소의 이름을 정의합니다.

## 이벤트

이벤트는 데이터 프로세서 변환에서 구성 요소의 처리 단계에 대한 레코드입니다. 스크립트 또는 라이브러리에서 각각의 앵커, 작업 또는 변환기가 이벤트를 생성합니다. **XMap**에서는 각 매핑 문이 이벤트를 생성합니다.

이러한 이벤트는 **데이터 프로세서 이벤트** 보기에서 볼 수 있습니다.

### 이벤트 유형

데이터 프로세서 변환은 로그 파일에 이벤트를 씁니다. 각 이벤트에는 이벤트가 성공했는지, 이벤트가 실패했는지 또는 이벤트가 실행되었지만 오류가 있는지 여부를 나타내는 이벤트 유형이 있습니다.

구성 요소가 하나 이상의 이벤트를 생성할 수 있습니다. 구성 요소는 이벤트가 성공했는지, 아니면 실패했는지에 따라 통과이거나 실패할 수 있습니다. 한 이벤트가 실패하면 구성 요소가 실패합니다.

다음 테이블에는 데이터 프로세서 변환이 생성하는 이벤트의 유형이 설명되어 있습니다.

이벤트 유형	설명
알림	정상적인 작업입니다.
경고	데이터 프로세서 변환이 실행되었지만 예기치 않은 상황이 발생했습니다. 예를 들어 데이터 프로세서 변환이 동일한 요소에 쓰기를 여러 번 했다고 가정합니다. 요소를 덮어쓸 때마다 데이터 프로세서 변환은 경고를 생성합니다.
실패	데이터 프로세서 변환이 실행되었지만 구성 요소가 실패했습니다. 예를 들어 필수 입력 요소가 비어 있습니다.
선택적 실패	데이터 프로세서 변환이 실행되었지만 선택적 구성 요소가 실패했습니다. 예를 들어 소스 문서에서 선택적 앵커가 누락되었습니다.
치명적 오류	심각한 오류 때문에 데이터 프로세서 변환이 실패했습니다. 예를 들어 입력 문서가 존재하지 않습니다.

### 데이터 프로세서 이벤트 보기

Developer tool에서 데이터 프로세서 변환을 실행한 경우 **데이터 프로세서 이벤트** 보기에 이벤트가 표시됩니다.

**데이터 프로세서 이벤트** 보기에는 **탐색** 패널과 **세부 정보** 패널이 있습니다. 탐색 패널에는 탐색 트리가 포함되어 있습니다. 탐색 트리에는 변환이 실행한 구성 요소가 시간 순서대로 나열됩니다. 각 노드에는 트리의 해당 노드 아래의 이벤트 중 가장 심각한 이벤트를 나타내는 아이콘이 표시됩니다. **탐색** 패널에서 노드를 선택하면 **세부 정보** 패널에 이벤트가 나타납니다.



탐색 트리에는 다음과 같은 최상위 노드가 포함되어 있습니다.

- 서비스 초기화. 데이터 프로세서 변환이 초기화하는 파일과 변수를 설명합니다.
- 실행. 스크립트, 라이브러리 또는 XMap이 실행한 구성 요소를 나열합니다.
- 요약. 처리 관련 통계를 표시합니다.

XMap을 실행하는 경우 탐색 패널의 각 노드 이름에 대괄호로 묶인 숫자가 나타납니다(예: [5]). 노드에 대한 이벤트를 생성한 문을 확인하려면 문 그리드에서 마우스 오른쪽 단추를 클릭하고 행 번호로 이동을 선택합니다. 노드 번호를 입력합니다.

스크립트를 실행하고 **탐색** 패널이나 **세부 정보** 패널에서 이벤트를 두 번 클릭하면 스크립트 편집기에서 해당 이벤트를 생성한 스크립트 구성 요소가 강조 표시됩니다. **데이터 뷰어** 보기의 **입력** 패널에는 이벤트를 생성한 예제 소스 문서 부분이 강조 표시됩니다.

## 로그

로그는 데이터 프로세서 변환의 레코드를 포함합니다. 데이터 프로세서 변환은 이벤트를 로그에 씁니다.

데이터 프로세서 변환은 다음과 같은 유형의 로그를 작성합니다.

### 디자인 타임 이벤트 로그

디자인 타임 이벤트 로그에는 **데이터 뷰어** 보기에서 데이터 프로세서 변환을 실행할 때 발생하는 이벤트가 포함되어 있습니다. 디자인 타임 로그는 **이벤트** 보기에서 볼 수 있습니다.

### 런타임 이벤트 로그

런타임 이벤트 로그에는 매핑에서 데이터 프로세서 변환을 실행할 때 발생하는 이벤트가 포함되어 있습니다. 텍스트 편집기에서 런타임 이벤트 로그를 보거나, 데이터 프로세서 변환의 **이벤트** 보기로 런타임 이벤트 로그를 끌어 놓을 수 있습니다.

### 사용자 로그

사용자 로그에는 스크립트에서 구성 요소에 대해 구성된 이벤트가 들어 있습니다. **데이터 뷰어** 보기에서 실행할 때와 매핑에서 실행할 때 데이터 프로세서 변환이 사용자 로그를 씁니다. 사용자 로그는 텍스트 편집기에서 볼 수 있습니다.

## 디자인 타임 이벤트 로그

디자인 타임 이벤트 로그에는 Developer tool의 **데이터 뷰어**에서 데이터 프로세서 변환을 실행할 때 발생하는 이벤트가 포함되어 있습니다.

**데이터 뷰어** 보기에서 데이터 프로세서 변환을 실행하는 경우 디자인 타임 이벤트 로그가 **데이터 프로세서 이벤트** 보기에 나타납니다. 기본적으로, 디자인 타임 이벤트 로그에는 알림, 경고 및 실패가 포함됩니다. 변환 설정에서 하나 이상의 이벤트 유형을 로그에서 제외하도록 데이터 프로세서 변환을 구성할 수 있습니다.

로그와 함께 입력 문서를 저장한 경우 **데이터 프로세서 이벤트** 보기에서 이벤트를 클릭하면 입력 문서 중 해당 이벤트를 생성한 위치를 찾을 수 있습니다. 데이터 프로세서 변환 설정을 구성할 때 실행할 때마다 또는 실패한 경우에만 입력 파일을 저장하도록 선택할 수 있습니다.

디자인 타임 이벤트 로그의 이름은 `events.cme`입니다. 다음 디렉터리에서 마지막으로 실행한 데이터 프로세서 변환에 대한 디자인 타임 이벤트 로그를 찾을 수 있습니다.

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

**데이터 뷰어**에서 변환을 실행할 때마다 데이터 프로세서 변환이 디자인 타임 이벤트 로그를 덮어씁니다. 디자인 타임 이벤트 로그를 나중에 변환을 실행한 후에 보려거나 서로 다른 실행의 로그를 비교하려면 디자인 타임 이벤트 로그의 이름을 변경하십시오. **Developer tool**을 닫으면 해당 위치에 아무 파일도 저장되지 않습니다.

## 런타임 이벤트 로그

런타임 이벤트 로그는 매핑에서 데이터 프로세서 변환을 실행할 때 발생하는 이벤트를 기록합니다.

데이터 프로세서 변환이 오류 없이 실행을 완료하면 이벤트 로그를 기록하지 않습니다. 실행에서 오류가 있을 경우 데이터 변환이 두 번째 실행되고 두 번째 실행 중에 이벤트 로그를 기록합니다. 런타임 이벤트 로그는 **events.cme**로 이름이 지정됩니다.

Windows 시스템에서 런타임 이벤트 로그는 다음 디렉터리에 있습니다.

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

Linux 또는 UNIX 시스템에서 루트 사용자의 런타임 이벤트 로그는 다음 디렉터리에 있습니다.

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

Linux 또는 UNIX 시스템에서 비 루트 사용자의 런타임 이벤트 로그는 다음 디렉터리에 있습니다.

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

구성 편집기를 사용하여 런타임 이벤트 로그의 위치를 변경할 수 있습니다.

## 데이터 프로세서 이벤트 보기에서 이벤트 로그 보기

**데이터 프로세서 이벤트** 보기에서 디자인 타임 이벤트 로그 또는 런타임 이벤트 로그를 확인할 수 있습니다.

Windows 탐색기를 열고 확인하려는 이벤트 로그 파일을 찾습니다. Windows 탐색기 창에서 **데이터 프로세서 이벤트** 보기로 로그를 끌어서 놓습니다. **데이터 프로세서 이벤트** 보기를 마우스 오른쪽 단추로 클릭하고 **찾기**를 선택하여 로그를 검색합니다.

**참고:** 가장 최근의 디자인 타임 이벤트 로그를 다시 로드하려면 **데이터 프로세서 이벤트** 보기를 마우스 오른쪽 단추로 클릭하고 **프로젝트 이벤트 다시 로드**를 선택합니다.

## 사용자 로그

사용자 로그에는 **Script**의 구성 요소 실패에 대해 구성하는 사용자 지정 메시지가 포함됩니다.

**Script**를 **데이터 뷰어** 보기에서 실행하고 매핑에서 실행할 때 데이터 프로세서 변환이 메시지를 사용자 로그에 기록합니다.

스크립트 구성 요소에 **on\_fail** 속성이 있을 경우 실패 시 사용자 로그에 메시지를 기록하도록 이 속성을 구성할 수 있습니다. **Script**에서 **on\_fail** 속성을 다음 값 중 하나로 설정합니다.

- LogInfo
- LogWarning
- LogError

**Script**를 실행할 때마다 새로운 사용자 로그가 생성됩니다. 사용자 로그 파일 이름에는 다음과 같이 변환 이름과 고유 GUID가 포함됩니다.

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

예: CalculateValue\_Aa93a9d14-a01f-442a-b9cb-c9ba5541b538.log

Windows 시스템의 경우 다음 디렉터리에서 사용자 로그를 찾을 수 있습니다.

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

Linux 또는 UNIX 시스템의 경우 다음 디렉터리에서 루트 사용자의 사용자 로그를 찾을 수 있습니다.

`/<Installation_Directory>/DataTransformation/UserLogs`

Linux 또는 UNIX 시스템의 경우 다음 디렉터리에서 비 루트 사용자의 사용자 로그를 찾을 수 있습니다.

`home/<Installation_Directory>/DataTransformation/UserLogs`

## 데이터 프로세서 변환 개발

새 변환 마법사를 사용하여 데이터 프로세서 변환을 자동 생성하거나 빈 데이터 프로세서 변환을 작성한 후 나중에 구성합니다. 빈 데이터 프로세서 변환을 작성하는 경우 변환에서 스크립트, XMap, 라이브러리 또는 유효성 검사 규칙 개체를 작성하도록 선택해야 합니다. 스크립트는 소스 문서를 계층적 형식으로 구문 분석하거나, 계층적 형식을 다른 파일 형식으로 변환하거나, 계층적 문서를 다른 계층적 형식으로 매핑할 수 있습니다. XMap은 입력 계층 파일을 다른 구조의 출력 계층 파일로 변환합니다. 라이브러리는 산업 메시지 형식을 계층 구조의 XML 문서로 변환하거나 XML에서 산업 표준 형식으로 변환합니다. 입력 또는 출력 계층을 정의하는 스키마를 선택합니다.

1. Developer tool에서 변환을 작성합니다.
2. 빈 데이터 프로세서 변환의 경우 다음 추가 단계를 수행합니다.
  - a. 입력 또는 출력 XML 계층을 정의하는 스키마 참조를 추가합니다.
  - b. 스크립트, XMap, 라이브러리 또는 유효성 검사 규칙 개체를 작성합니다.
3. 입력 및 출력 포트를 구성합니다.
4. 변환을 테스트합니다.

## 데이터 프로세서 변환 작성

Developer tool에서 데이터 프로세서 변환을 작성합니다. 빈 데이터 프로세서 변환을 작성하는 경우 변환에서 스크립트, XMap, 라이브러리 또는 유효성 검사 규칙 개체를 작성해야 합니다. 또는 새 변환 마법사를 사용하여 자동으로 데이터 프로세서 변환을 생성할 수 있습니다.

1. Developer tool에서 **파일 > 새로 만들기 > 변환**을 클릭합니다.
2. 데이터 프로세서 변환을 선택하고 **다음**을 클릭합니다.
3. 변환의 이름을 입력하고 변환을 배치할 모델 리포지토리 위치를 찾습니다.
4. 마법사를 사용하여 데이터 프로세서 변환을 작성할지 또는 빈 데이터 프로세서 변환을 작성할지 선택합니다.
5. 빈 데이터 프로세서 변환을 작성하도록 선택한 경우 **마침**을 클릭합니다.

Developer tool이 리포지토리에 빈 변환을 작성합니다. Developer tool에 **개요** 보기가 나타납니다.
6. 마법사를 사용하여 데이터 프로세서 변환을 작성하도록 선택한 경우 다음 단계를 수행합니다.
  - a. **다음**을 클릭합니다.
  - b. 입력 형식을 선택합니다.
  - c. COBOL, JSON 또는 ASN.1 등의 특정 입력 형식에 필요한 경우 스키마, copybook, 예제 파일 또는 사양 파일을 찾아 선택합니다.
  - d. 출력 형식을 선택합니다.
  - e. 출력 형식에 필요한 경우 스키마, copybook, 예제 파일 또는 사양 파일을 찾아 선택합니다.
  - f. **마침**을 클릭합니다. 마법사가 리포지토리에 변환을 작성합니다.

변환에 **Parser**, **Serializer** 또는 **Mapper**가 포함되거나 공통 구성 요소가 있는 개체가 포함될 수 있습니다. 스키마, **copybook**, 예제 파일 또는 사양 파일을 선택한 경우 마법사가 파일의 계층에 해당하는 스키마도 리포지토리에 작성합니다.

## 스키마 개체 선택

작성하려는 각 **XMap** 또는 스크립트 구성 요소에 대한 입력 또는 출력 계층을 정의하는 스키마 개체를 선택합니다.

참조 보기에서 스키마 참조를 추가할 수도 있고 **Script** 또는 **XMap** 개체를 작성할 때 스키마 참조를 추가할 수도 있습니다. 스키마 개체를 **Script** 또는 **XMap**에서 참조하려면 해당 개체가 모델 리포지토리에 있어야 합니다.

1. 데이터 프로세서 변환 **참조** 보기에서 **추가**를 클릭합니다.
2. 스키마 개체가 모델 리포지토리에 있으면 해당 스키마 개체를 찾아서 선택합니다.
3. 스키마 개체가 모델 리포지토리에 없을 경우에는 **새 스키마 개체 작성**을 클릭하고 계층 스키마 파일에서 스키마 개체를 가져옵니다.
4. 마침을 클릭하여 스키마 참조를 데이터 프로세서 변환에 추가합니다.

## 빈 데이터 프로세서 변환에서 개체 작성

데이터 프로세서 변환 **개체** 보기에서 스크립트, 라이브러리, **XMap** 또는 유효성 검사 규칙 개체를 작성합니다. 개체를 작성한 후 구성하기 위해 **개체** 보기에서 개체를 열 수 있습니다.

### 스크립트 작성

스크립트 개체를 작성하고, 작성할 스크립트 구성 요소의 유형을 정의할 수 있습니다. 필요에 따라 스키마 참조 및 예제 소스 파일을 정의할 수 있습니다.

1. 데이터 프로세서 변환 **개체** 보기에서 **새로 만들기**를 클릭합니다.
2. 스크립트의 이름을 입력하고 **다음**을 클릭합니다.
3. 파서 또는 직렬 변환기를 작성하도록 선택합니다. 매퍼, 변환기 또는 스트리머 구성 요소를 작성하려면 '기타'를 선택합니다.
4. 구성 요소의 이름을 입력합니다.
5. 구성 요소가 변환에서 데이터를 처리하는 첫 번째 구성 요소인 경우 **시작 구성 요소로 설정**을 활성화합니다.
6. 이 스크립트에 대한 스키마 참조를 입력하려면 **다음**을 클릭합니다. 스키마 참조를 사용하지 않으려면 **마침**을 클릭합니다.
7. 스키마 참조를 작성하도록 선택한 경우 **스키마 개체에 참조 추가**를 선택하고 모델 리포지토리에서 스키마 개체를 찾아봅니다. **새 스키마 개체 작성**을 클릭하여 모델 리포지토리에 스키마 개체를 작성합니다.
8. 예제 소스 참조 또는 예제 텍스트를 입력하려면 **다음**을 클릭합니다. 예제 소스를 정의하지 않으려면 **마침**을 클릭합니다.  
예제 소스를 사용하여 샘플 데이터를 정의하고 스크립트를 테스트합니다.
9. 예제 소스를 선택하도록 선택한 경우 **파일**을 선택하고 샘플 파일을 찾아봅니다.  
또한 샘플 텍스트를 **텍스트** 영역에 입력할 수도 있습니다. **Developer tool**은 텍스트를 사용하여 스크립트를 테스트합니다.
10. **마침**을 클릭합니다.  
**스크립트** 보기가 **Developer tool** 편집기에 나타납니다.

## XMap 작성

Data Transformation **개체** 보기에서 XMap을 작성할 수 있습니다. XMap을 작성할 때 입력 및 출력 계층 문서를 설명하는 스키마가 있어야 합니다. 입력 계층의 루트 요소인 요소를 스키마에서 선택합니다.

1. 데이터 프로세서 변환 **개체** 보기에서 **새로 만들기**를 클릭합니다.
2. XMap을 선택하고 **다음**을 클릭합니다.
3. XMap 이름을 입력합니다.
4. XMap 구성 요소가 변환에서 데이터를 처리하는 첫 번째 구성 요소인 경우 **시작 구성 요소로 설정**을 활성화합니다.  
**다음**을 클릭합니다.
5. 스키마 참조를 작성하도록 선택한 경우 **스키마 개체에 참조 추가**를 선택하고 모델 리포지토리에서 스키마 개체를 찾아봅니다.  
새 스키마 개체를 가져오려면 **새 스키마 개체 작성**을 클릭합니다.
6. XMap을 테스트하는 데 사용할 수 있는 샘플 계층 파일이 있는 경우 파일 시스템에서 해당 파일을 찾아 선택합니다.  
샘플 계층 파일을 변경할 수 있습니다.
7. 입력 계층의 루트를 선택합니다.  
**루트 요소 선택** 대화 상자에서 입력 계층 파일의 루트 요소인 요소를 스키마에서 선택합니다. 스키마에서 요소를 검색할 수 있습니다. 패턴 검색을 사용할 수 있습니다. 문자열에서 문자 수에 상관없이 일치시키려면 **\*<문자열>**을 입력합니다. 단일 문자를 일치시키려면 **?<문자>**를 입력합니다.
8. **마침**을 클릭합니다.  
작성하는 각 XMap에 대한 보기가 작성됩니다. 보기를 클릭하여 매핑을 구성합니다.

## 라이브러리 작성

Data Transformation **개체** 보기에서 라이브러리 개체를 작성합니다. 메시지 유형, 구성 요소 및 이름을 선택합니다. 필요에 따라 라이브러리 개체를 테스트하는 데 사용할 수 있는 샘플 메시지 유형 소스 파일을 정의할 수 있습니다.

데이터 프로세서 변환에서 라이브러리를 작성하기 전에 컴퓨터에 라이브러리 소프트웨어 패키지를 설치하십시오.

1. 데이터 프로세서 변환 **개체** 보기에서 **새로 만들기**를 클릭합니다.
2. 라이브러리를 선택하고 **다음**을 클릭합니다.
3. 메시지 유형을 찾아 선택합니다.
4. 파서 또는 직렬 변환기를 작성하도록 선택합니다.  
라이브러리 개체 입력이 메시지 유형이고 출력이 XML인 경우 파서를 작성합니다. 라이브러리 개체 입력이 XML이고 출력이 메시지 유형인 경우 직렬 변환기를 작성합니다.
5. 라이브러리가 데이터 프로세서 변환에서 데이터를 처리하는 첫 번째 구성 요소인 경우 **시작 구성 요소로 설정**을 활성화합니다.  
**다음**을 클릭합니다.
6. 라이브러리를 테스트하는 데 사용할 수 있는 샘플 메시지 유형 소스 파일이 있는 경우 파일 시스템에서 해당 파일을 찾아 선택합니다.  
샘플 파일을 변경할 수 있습니다.

7. **마침**을 클릭합니다.

작성한 각 메시지 유형에 대해 **Developer tool**이 보기를 작성합니다. 매핑에 액세스하려면 보기를 클릭합니다.

## 유효성 검사 규칙 작성

데이터 프로세서 변환 **개체** 보기에서 유효성 검사 규칙 개체를 작성합니다.

1. 데이터 프로세서 변환 **개체** 보기에서 **새로 만들기**를 클릭합니다.
2. 유효성 검사 규칙을 선택하고 **다음**을 클릭합니다.
3. 유효성 검사 규칙의 이름을 입력합니다.
4. 유효성 검사 규칙을 테스트하는 데 사용할 수 있는 샘플 XML 파일이 있는 경우 파일 시스템에서 해당 파일을 찾아 선택합니다.  
샘플 XML 파일을 변경할 수 있습니다.
5. **마침**을 클릭합니다.  
**Developer tool**이 유효성 검사 규칙 개체를 작성하고 유효성 검사 규칙 편집기에서 엽니다.

## 예제 소스 추가

예제 소스를 선택하여 작성하려는 스크립트, XMap, 라이브러리 또는 유효성 검사 규칙을 테스트합니다.

스크립트, XMap, 라이브러리 또는 유효성 검사 규칙을 작성할 때 예제 소스를 추가할 수 있습니다. 예제 소스를 선택하면 예제 소스가 모델 리포지토리에 추가됩니다. 모델 리포지토리 제한으로 인해 예제 소스 파일 크기는 5MB로 제한됩니다.

예제 소스를 변경할 수 있습니다.

## 포트 작성

**개요** 보기에서 입력 및 출력 포트를 구성합니다.

**Script**에서 추가 입력 또는 출력 포트를 구성하는 경우 **Developer tool**이 기본적으로 추가 입력 포트와 추가 출력 포트를 변환에 추가합니다. 따라서 **개요** 보기에서 입력 포트를 추가하지 않습니다.

1. XML 대신 출력 데이터 행을 반환하려면 **관계형 출력**을 활성화합니다.  
관계형 출력을 활성화하면 **Developer tool**이 기본 출력 포트를 제거합니다.
2. 입력 포트 데이터 유형, 포트 유형, 전체 자릿수 및 소수 자릿수를 선택합니다.
3. 관계형 출력 포트를 정의하지 않는 경우 출력 포트 데이터 유형, 포트 유형, 전체 자릿수 및 소수 자릿수를 선택합니다.
4. **Script**에 추가 입력 포트가 있을 경우 포트에 대한 예제 입력 파일의 위치를 정의할 수 있습니다. **입력 위치** 필드에서 **열기** 단추를 클릭하여 파일을 찾습니다.
5. 관계형 출력을 활성화했을 경우 **출력 매핑**을 클릭하여 출력 포트를 작성합니다.
6. 포트 보기에서 **계층 출력** 영역의 노드를 **관계형 포트** 영역의 필드에 매핑합니다.

## 변환 테스트

데이터 뷰어 보기에서 데이터 프로세서 변환을 테스트합니다.

변환을 테스트하기 전에 시작 구성 요소를 정의했는지 확인합니다. Script에서 시작 구성 요소를 정의하거나 개요 탭에서 시작 구성 요소를 선택할 수 있습니다. 또한 테스트하는 데 사용할 예제 입력 파일도 선택한 상태여야 합니다.

1. 데이터 뷰어 보기를 엽니다.
2. 실행을 클릭합니다.

Developer tool이 변환의 유효성을 검사합니다. 오류가 없으면 Developer tool이 입력 영역에 예제 파일을 표시합니다. 출력 결과는 출력 패널에 나타납니다.

3. 이벤트 표시를 클릭하여 데이터 프로세서 이벤트 보기를 표시합니다.
4. Script 편집기에서 이벤트를 디버깅하기 위해 데이터 프로세서 이벤트 보기에서 이벤트를 두 번 클릭합니다.
5. 여러 구성 요소를 편집하는 경우 각각 서로 다른 예제 입력 파일로 입력 파일을 변경하도록 편집기와 동기화를 클릭합니다.

파일 시스템에서 예제 파일 콘텐츠를 수정하면 변경 사항이 입력 영역에 표시됩니다.

## 데이터 프로세서 변환 가져오기 및 내보내기

데이터 프로세서 변환을 서비스로 내보내 Data Transformation 리포지토리에 저장할 수 있습니다. Data Transformation 서비스를 Developer tool로 가져올 수도 있습니다. Data Transformation 서비스를 가져오면 Developer tool이 해당 서비스에서 데이터 프로세서 변환을 작성합니다.

**참고:** Data Transformation 서비스를 모델 리포지토리로 가져오면 Developer tool이 연결된 스키마를 리포지토리로 가져옵니다. 리포지토리에 스키마를 수정하면 변경 사항이 변환 스키마 참조에 나타나지 않는 경우가 있습니다. 모델 리포지토리 연결을 닫고 다시 열거나 Developer tool을 닫고 다시 열면 스키마 변경 사항이 변환에 나타나게 할 수 있습니다.

## 데이터 프로세서 변환을 서비스로 내보내기

데이터 프로세서 변환을 Data Transformation 서비스로 내보낼 수 있습니다. 서비스를 실행하려는 시스템의 파일 시스템 리포지토리로 서비스를 내보냅니다. PowerCenter, 사용자 정의 응용 프로그램 또는 Data Transformation CM\_console 명령으로 서비스를 실행할 수 있습니다.

1. Object Explorer 보기에서 내보내려는 데이터 프로세서 변환을 마우스 오른쪽 단추로 클릭한 다음 내보내기를 선택합니다.  
내보내기 대화 상자가 표시됩니다.
2. Informatica > 데이터 프로세서 변환 내보내기를 선택하고 다음을 클릭합니다.  
선택 페이지가 표시됩니다.
3. 다음을 클릭합니다.  
서비스 이름 및 대상 폴더 선택 페이지가 표시됩니다.
4. 대상 폴더를 선택합니다.
  - Developer tool을 호스팅하는 시스템에서 서비스를 내보내려면 서비스 폴더를 클릭합니다.



- 다른 시스템에 서비스를 배포하려면 **폴더**를 클릭합니다. 서비스를 배포하려는 시스템의 \ServiceDB 디렉터리를 찾아 선택합니다.
5. **마침**을 클릭합니다.

## 여러 Data Transformation 서비스 가져오기

Data Transformation 서비스의 디렉터리를 저장한 시스템에서 디렉터리를 가져올 수 있습니다. Data Transformation 서비스를 Developer 모델 리포지토리로 가져오면 Developer tool은 .cmw 파일과 함께 변환, 스키마 및 예제 데이터를 가져옵니다. 여러 서비스를 가져와야 하는 경우 한 번에 한 서비스를 가져오는 대신 서비스의 디렉터리를 가져오면 됩니다.

1. **파일 > 가져오기**를 클릭합니다.  
가져오기 대화 상자가 표시됩니다.
2. **Informatica Data Transformation 서비스(폴더) 가져오기**를 선택하고 **다음**을 클릭합니다.  
**Data Transformation 서비스 가져오기** 페이지가 표시됩니다.
3. 가져오려는 디렉터리를 찾아 선택합니다.
4. 변환을 저장하려는 리포지토리의 위치를 찾아 선택하고 **마침**을 클릭합니다.  
Developer tool은 .cmw 파일과 함께 변환, 스키마 및 예제 데이터를 가져옵니다.

## Data Transformation 서비스 가져오기

Data Transformation 서비스 .cmw 파일을 모델 리포지토리로 가져와 데이터 프로세서 변환을 생성할 수 있습니다. Developer tool은 .cmw 파일과 함께 변환, 스키마 및 예제 데이터를 가져옵니다.

1. **파일 > 가져오기**를 클릭합니다.  
가져오기 대화 상자가 표시됩니다.
2. **Informatica Data Transformation 서비스(단일) 가져오기**를 선택하고 **다음**을 클릭합니다.  
**Data Transformation 서비스 가져오기** 페이지가 표시됩니다.
3. 가져오려는 서비스 .cmw 파일을 찾아 선택합니다.  
Developer tool이 서비스 파일 이름에 따라 변환에 이름을 지정합니다. 이 이름을 변경할 수 있습니다.
4. 변환을 저장하려는 리포지토리의 위치를 찾아 선택하고 **마침**을 클릭합니다.  
Developer tool은 .cmw 파일과 함께 변환, 스키마 및 예제 데이터를 가져옵니다.
5. 변환을 편집하려면 **Object Explorer** 보기에서 변환을 두 번 클릭합니다.

## 데이터 프로세서 변환이 포함된 매핑을 PowerCenter로 내보내기

데이터 프로세서 변환이 포함된 매핑을 PowerCenter로 내보내는 경우 개체를 로컬 파일로 내보낸 다음 PowerCenter로 매핑을 가져올 수 있습니다. 또는, 매핑을 직접 PowerCenter 리포지토리로 내보낼 수 있습니다.

1. **개체 탐색기** 보기에서 내보낼 매핑을 선택합니다. 마우스 오른쪽 단추를 클릭하고 **내보내기**를 선택합니다.  
내보내기 대화 상자가 표시됩니다.
2. **Informatica > PowerCenter**를 선택합니다.
3. **다음**을 클릭합니다.  
**PowerCenter로 내보내기** 대화 상자가 나타납니다.



4. 프로젝트를 선택합니다.
5. PowerCenter 릴리스를 선택합니다.
6. 내보내기 위치, PowerCenter 가져오기 XML 파일 또는 PowerCenter 리포지토리를 선택합니다.
7. 내보내기 옵션을 지정합니다.
8. 다음을 클릭합니다.

Developer tool에서 내보낼 개체를 선택하라는 메시지가 표시됩니다.

9. 내보낼 개체를 선택하고 마침을 클릭합니다.  
Developer tool이 개체를 선택한 위치로 내보냅니다. 매핑을 한 위치로 내보낸 경우 Developer tool은 매핑의 데이터 프로세서 변환을 지정된 위치의 폴더에 서비스로 내보냅니다.
10. 매핑을 PowerCenter 리포지토리로 내보낸 경우 서비스는 다음 디렉터리 경로로 내보내집니다. %temp%\DTServiceExport2PC\  
내보내기 기능은 다음 이름을 사용하여 각 서비스에 대해 별도의 폴더를 작성합니다.  
<date><serviceFullName>  
변환에 관계형 매핑이 포함된 경우 관계형-계층 매핑에 대한 폴더와 계층-관계형 매핑에 대한 별도의 폴더가 작성됩니다.
11. 파일을 내보낸 로컬 위치에서 데이터 프로세서 변환 서비스가 포함된 폴더를 PowerCenter ServiceDB 폴더로 복사합니다.
12. 매핑을 PowerCenter 가져오기 XML 파일로 내보낸 경우 매핑을 PowerCenter로 가져옵니다.  
PowerCenter로 개체를 가져오는 방법에 대한 자세한 내용은 *PowerCenter 9.6.0 리포지토리 가이드*를 참조하십시오.

## 데이터 프로세서 변환 비원시 환경 내

비원시 환경에서 데이터 프로세서 변환의 처리는 변환을 실행하는 엔진에 따라 다릅니다.

다음과 같은 비원시 런타임 엔진에 대한 지원을 고려하십시오.

- Blaze 엔진. 제한적으로 지원됩니다.
- Spark 엔진. 지원되지 않습니다.
- Databricks Spark 엔진. 지원되지 않습니다.

## 데이터 프로세서 변환 Blaze 엔진 내

데이터 프로세서 변환은 다음과 같은 제한과 함께 지원됩니다.

- 데이터 프로세서 모드 변환이 입력 매핑 또는 서비스 및 입력 매핑으로 설정된 경우 매핑 유효성 검사가 실패합니다.

## 제 14 장

# 결정 변환

이 장에 포함된 항목:

- [결정 변환 개요, 238](#)
- [결정 변환 함수, 238](#)
- [결정 변환의 조건문, 241](#)
- [결정 변환 연산자, 241](#)
- [결정 변환의 NULL 처리, 242](#)
- [결정 전략 구성, 243](#)
- [결정 변환의 고급 속성, 243](#)
- [결정 변환 - 비원시 환경, 244](#)

## 결정 변환 개요

결정 변환은 입력 데이터의 조건을 평가하고 이러한 조건의 결과에 따라 출력을 작성하는 수동 변환입니다.

입력 필드의 값에 따라 다른 값을 생성하도록 결정 변환을 구성할 수 있습니다. 예를 들어 고객 수익이 특정 금액을 초과하는 경우 해당 고객의 이름에 문자열 "우선 순위"를 추가할 수 있습니다.

결정 변환에 여러 결정 전략을 추가할 수 있습니다. 각 전략은 **IF-THEN-ELSE** 조건문을 평가합니다. 이 조건문 안에 **ELSEIF** 조건을 사용하거나 추가 **IF-THEN-ELSE** 문을 중첩할 수 있습니다.

결정 변환은 조건문 및 함수를 사용하여 소스 데이터를 테스트할 수 있다는 점에서 식 변환과 유사합니다. 그러나 결정 변환은 다음과 같은 점에서 식 변환과 다릅니다.

- 결정 변환은 **IF-THEN-ELSE** 문을 사용하여 조건을 평가합니다. 식 변환은 **IIF** 문을 사용합니다.
- 결정 변환에는 식 변환에서 사용할 수 없는 함수가 포함됩니다.
- 각 결정 전략은 여러 개의 출력을 생성할 수 있습니다.

## 결정 변환 함수

결정 변환은 결정 전략을 정의할 때 사용하는 미리 정의된 함수에 대한 액세스를 제공합니다.

결정 변환의 식 편집기에는 **Decision** 폴더가 있습니다. 이 폴더에는 결정 변환과 관련된 함수가 포함됩니다. 결정 변환의 식 편집기에는 식 변환 함수에 대한 액세스를 제공하는 다른 폴더도 있습니다.

식 편집기에서 함수를 클릭하면 함수의 사용 및 데이터 유형과 함께 함수의 기능에 대한 설명이 표시됩니다.

**참고:** 모든 식 변환 함수가 결정 변환과 호환되는 것은 아닙니다. 결정 변환은 호환되는 식 변환 함수에 대한 액세스만 제공합니다.

### 결정 변환 함수의 목록

- ABS
- ADD\_TO\_DATE
- ASCII
- CEIL
- CHOOSE
- CHR
- CHRCODE
- CONCAT
- CONTAINS
- CONVERT\_BASE
- COS
- COSH
- CRC32
- CUME
- CURDATE
- CURTIME
- DATE\_COMPARE
- DATE\_DIFF
- DATECONVERT
- EXP
- FLOOR
- FV
- GET\_DATE\_PART
- GREATEST
- IN
- INDEXOF
- INITCAP
- INSTR
- IS\_DATE
- IS\_NUMBER
- ISNULL
- LAST\_DAY
- LEAST
- LEFTSTR

- LENGTH
- LN
- LOG
- LOWER
- LPAD
- LTRIM
- MAKE\_DATE\_TIME
- MAX
- MD5
- METAPHONE
- MIN
- MOD
- MONTHCOMPARE
- MOVINGAVG
- MOVINGSUM
- NPER
- PMT
- POWER
- PV
- RAND
- RATE
- REG\_EXTRACT
- REG\_MATCH
- REG\_REPLACE
- REPLACECHR
- REPLACESTR
- REVERSE
- RIGHTSTR
- ROUND
- RPAD
- RTRIM
- SET\_DATE\_PART
- SIGN
- SIN
- SINH
- SOUNDEX
- SQRT
- SUBSTR

- TAN
- TANH
- TIMECOMPARE
- TO\_CHAR
- TO\_DATE
- TO\_FLOAT
- TO\_INTEGER
- TRUNC
- UPPER
- XOR

**참고:** 상수 값을 사용하여 결정 변환에서 CURDATE, DATE\_COMPARE, DATECONVERT 및 MONTHCOMPARE 함수의 날짜 형식을 정의합니다.

## 결정 변환의 조건문

결정 변환은 IF-THEN-ELSE 조건문을 사용하여 입력 데이터를 평가합니다.

이러한 조건문 안에 ELSEIF 조건을 사용하거나 추가 IF-THEN-ELSE 문을 중첩할 수 있습니다. 결정 변환의 조건문은 다음 형식을 사용합니다.

```
// Primary condition
IF <Boolean expression>
THEN <Rule Block>
// Optional - Multiple ELSEIF conditions
ELSEIF <Boolean expression>
THEN <Rule Block>
// Optional ELSE condition
ELSE <Rule Block>
ENDIF
```

규칙 블록 안에 추가 조건문을 중첩할 수 있습니다.

## 결정 변환 연산자

결정 변환 연산자를 사용하여 결정 전략을 정의합니다.

다음 표에는 결정 변환 연산자가 설명되어 있습니다.

연산자 유형	연산자	설명
할당	:=	값을 포트에 할당합니다.
부울	AND	필수 논리 조건을 추가합니다. 상위 부울 식이 true가 되려면 이 연산자에 연결된 모든 논리 조건이 true여야 합니다.

연산자 유형	연산자	설명
부울	OR	논리 조건을 추가합니다. 상위 부울 식이 true가 되려면 이 연산자에 연결된 최소 1개의 논리 조건이 true여야 합니다.
부울	NOT	부정 논리 조건을 지정합니다. 상위 부울 식이 true가 되려면 이 연산자가 지정하는 부정 조건이 true여야 합니다.
결정	=	비교 항목이 동일한지 테스트합니다. 문자열 또는 숫자 데이터 유형과 함께 사용합니다.
결정	<>	비교 항목이 동일하지 않은지 테스트합니다. 문자열 또는 숫자 데이터 유형과 함께 사용합니다.
결정	<	값이 다른 값보다 작은지 테스트합니다. 숫자 데이터 유형과 함께 사용합니다.
결정	<=	값이 다른 값보다 작거나 같은지 테스트합니다. 숫자 데이터 유형과 함께 사용합니다.
결정	>	값이 다른 값보다 큰지 테스트합니다. 숫자 데이터 유형과 함께 사용합니다.
결정	>=	값이 다른 값보다 크거나 같은지 테스트합니다. 숫자 데이터 유형과 함께 사용합니다.
숫자	-	빼기
숫자	NEG	부정
숫자	+	더하기
숫자	*	곱하기
숫자	/	나누기
숫자	%	모듈로. 한 숫자를 다른 숫자로 나눈 나머지를 반환합니다.
문자열		문자열을 연결합니다.

## 결정 변환의 NULL 처리

NULL 처리는 결정 변환의 NULL 값 데이터에 대한 데이터 통합 서비스의 처리 방식을 결정합니다.

NULL 처리를 활성화하면 결정 변환이 NULL 입력 데이터의 원래 형태를 유지하고 NULL 입력 값을 사용하여 함수를 평가합니다.

NULL 처리를 비활성화하면 결정 변환이 NULL 입력 데이터에 기본값을 할당하고 기본값을 사용하여 함수를 평가합니다. 예를 들어 정수 유형의 입력 필드에 NULL 값이 있는 경우 결정 변환이 0 값을 입력 필드에 할당하고 입력 값 0을 사용하여 함수를 평가합니다.

결정 변환에서 NULL 처리는 기본적으로 활성화되지 않습니다. **전략** 탭에서 NULL 처리를 활성화할 수 있습니다. 변환의 전략을 구성한 후에 NULL 처리를 활성화할 수 있습니다.

## 결정 전략 구성

결정 전략을 구성하려면 소스 데이터를 결정 변환에 연결하고 변환 보기에서 속성을 편집하십시오.

1. 결정 변환을 엽니다.
2. 변환에 입력 및 출력 포트가 포함되어 있는지 확인합니다.
3. **결정** 보기를 선택합니다.
4. **추가**를 클릭합니다.
5. 전략 이름을 입력합니다.
6. 식 영역에서 IF-THEN-ELSE 조건부 문을 입력합니다.
7. 식에 함수를 추가하려면 **함수** 탭에서 함수를 찾아 함수 이름을 두 번 클릭합니다.  
**팁:** 함수를 빠르게 입력하려면 함수 이름의 첫 자를 입력하고 CTRL-Space를 선택합니다.
8. 식에 포트를 추가하려면 **포트** 탭에서 포트를 찾습니다. 포트 이름을 두 번 클릭하여 식에 추가합니다. 필요에 따라 **출력 포트 편집**을 클릭하여 출력 포트 설정을 편집하거나 출력 포트를 추가합니다.
9. 필요에 따라 "/" 다음에 설명을 입력하여 설명 줄을 추가합니다.
10. **유효성 검사**를 클릭하여 결정 식이 유효한지 확인합니다.
11. **확인**을 클릭하여 전략을 저장합니다.
12. 필요에 따라 다른 전략을 추가합니다. 전략마다 고유한 출력 포트를 사용해야 합니다. 전략은 출력 포트를 공유할 수 없습니다.

## 결정 변환의 고급 속성

결정 변환 데이터에 대한 데이터 통합 서비스의 처리 방식을 결정하는 데 도움이 되는 속성을 구성할 수 있습니다.

결정 변환에 대해 다음 고급 속성을 구성하십시오.

### 추적 수준

이 변환에 대해 로그에 표시되는 세부 정보의 양입니다. 간단, 보통, 자세한 정보 표시 초기화 또는 자세한 정보 표시 데이터를 선택할 수 있습니다. 기본값은 보통입니다.

### 분할 가능

여러 스레드를 사용하여 변환을 처리할 수 있습니다. 데이터 통합 서비스가 변환을 처리하는 데 하나의 스레드를 사용하도록 하려면 이 옵션을 선택 취소합니다. 데이터 통합 서비스는 여러 스레드를 사용하여 나머지 매핑 파이프라인 단계를 처리할 수 있습니다.

변환에서 CUME, MOVINGSUM 또는 MOVINGAVG와 같은 숫자 함수 중 하나를 사용하는 경우 결정 변환에 대한 분할을 비활성화할 수 있습니다. 이러한 함수는 행별로 실행 집계 및 평균을 계산합니다. CUME, MOVINGSUM 또는 MOVINGAVG 함수를 사용하는 분할된 변환이 각 매핑 실행에서 동일한 계산 결과를 반환하지 않을 수 있습니다.

변환에서 CUME, MOVINGSUM 또는 MOVINGAVG 함수를 사용하지 않는 경우 변환에 대해 분할을 활성화하여 성능을 최적화하십시오.

## 결정 변환 - 비원시 환경

비원시 환경에서 결정 변환의 처리는 변환을 실행하는 엔진에 따라 다릅니다.

다음과 같은 비원시 런타임 엔진에 대한 지원을 고려하십시오.

- **Blaze** 엔진. 제한 없이 지원됩니다.
- **Spark** 엔진. 제한적으로 지원됩니다. 결정 변환 속성을 분할 가능하도록 구성해야 합니다.
- **Databricks Spark** 엔진. 지원되지 않습니다.



## 제 15 장

# 중복 레코드 예외 변환

이 장에 포함된 항목:

- [중복 레코드 예외 변환 개요, 245](#)
- [중복 레코드 예외 프로세스 흐름, 246](#)
- [중복 레코드 예외, 246](#)
- [중복 레코드 예외 구성 보기, 246](#)
- [포트, 248](#)
- [중복 레코드 예외 변환의 고급 속성, 250](#)
- [중복 레코드 예외 매핑 예제, 251](#)
- [중복 레코드 예외 변환 작성, 256](#)

## 중복 레코드 예외 변환 개요

중복 레코드 예외 변환은 데이터 품질 프로세스의 출력을 읽고 수동 검토가 필요한 중복 레코드를 식별하는 활성 변환입니다. 중복 레코드 예외 변환은 여러 그룹이 포함되는 변환입니다.

중복 레코드 예외 변환은 다른 변환 또는 다른 매핑의 데이터 개체로부터 입력을 수신합니다. 중복 레코드 예외 변환에 대한 입력에는 레코드의 중복 여부를 결정하는 데 사용되는 숫자 일치 점수가 포함되어야 합니다. 중복 레코드 예외 변환에는 상한 및 하한 일치 점수 임계값을 설정합니다.

중복 레코드 예외 변환은 다음 작업 중 하나를 수행합니다.

- 일치 점수가 상한 임계값보다 크거나 같은 경우 변환이 레코드를 중복 레코드로 처리하고 데이터베이스 대상에 기록합니다.
- 일치 점수가 상한 임계값보다 작고 하한 임계값보다 큰 경우 변환이 레코드를 가능한 중복 레코드로 처리하고 수동 검토를 위해 다른 대상에 기록합니다. 레코드가 클러스터에 속하는 경우 변환이 클러스터의 모든 레코드를 대상에 기록합니다.
- 클러스터에 하한 임계값보다 작은 일치 점수가 있는 경우 변환이 클러스터의 모든 레코드를 고유한 레코드 출력 그룹으로 이동합니다. 크기가 1인 클러스터는 일치 점수에 관계없이 고유한 그룹으로 라우팅됩니다. 기본적으로 중복 레코드 예외 변환은 고유한 레코드를 대상에 기록하지 않습니다. 사용자는 고유한 레코드를 반환하도록 변환을 구성할 수 있습니다.
- 클러스터에 0~100 범위가 아닌 일치 점수가 있는 경우 예외 변환이 클러스터의 모든 행을 무시합니다. 데이터 통합 서비스가 클러스터 ID를 포함하는 메시지를 기록합니다.

## 중복 레코드 예외 프로세스 흐름

중복 레코드 예외 변환은 다른 데이터 품질 변환의 출력을 분석하고 테이블을 작성하여 데이터 품질 수준이 다른 레코드를 이 테이블에 포함합니다.

데이터 품질 변환을 단일 매핑에 구성하거나 프로세스의 여러 단계에 대한 매핑을 작성할 수 있습니다.

수동 검토가 필요한 중복 레코드는 **Analyst** 도구에서 검토하고 업데이트할 수 있습니다.

**Developer tool**에서 다음 태스크를 수행하십시오.

1. 데이터 품질 문제에 대한 점수 값을 생성하는 매핑을 작성합니다.
2. 일치 변환을 클러스터 모드에서 사용하여 중복 레코드 예외에 대한 점수 값을 생성합니다.
3. 일치 변환의 출력을 읽도록 중복 레코드 예외 변환을 구성합니다. 레코드의 일치 점수 값에 따라 레코드를 데이터베이스 테이블에 기록하도록 변환을 구성합니다.
4. 자동 통합 레코드에 대한 대상 데이터 개체를 구성합니다.
5. **중복 레코드 테이블 생성** 옵션을 클릭하여 중복 레코드 테이블을 작성하고 매핑 캔버스에 추가합니다.
6. 매핑을 워크플로우에 추가합니다.
7. 가능한 중복 레코드에 대한 수동 검토를 사용자에게 할당하는 휴먼 태스크를 구성합니다. 사용자는 **Analyst** 도구에서 레코드를 검토하고 업데이트할 수 있습니다.

## 중복 레코드 예외

중복 레코드 예외 변환을 사용하여 수동 검토가 필요한 중복 데이터의 클러스터를 식별할 수 있습니다. 중복 가능성은 클러스터에 있는 레코드의 일치 점수에 따라 결정됩니다. 변환에서 일치 점수에 대한 상한 및 하한 임계값을 구성할 수 있습니다. 상한 및 하한 임계값은 유사점의 정도를 정의합니다.

클러스터에는 일치 작업을 통해 그룹화된 관련 레코드가 포함됩니다. 일치 변환은 중복 분석 작업 및 ID 확인 작업을 통해 클러스터를 작성합니다. 클러스터의 각 레코드는 동일한 클러스터 ID를 갖습니다. 클러스터에서 가장 낮은 일치 점수가 상한 임계값과 하한 임계값 사이인 경우 중복 레코드 예외 변환이 해당 클러스터를 중복 레코드 예외 클러스터로 식별합니다. 일치 변환은 클러스터 ID 값 열을 모든 레코드에 추가합니다. 중복 레코드에는 동일한 클러스터 ID가 추가됩니다.

클러스터에서 점수 일치치가 가장 낮은 레코드가 클러스터 유형을 결정합니다. 예를 들어 클러스터에 일치 점수가 0.95인 레코드가 11개 있고 일치 점수가 0.79인 레코드가 1개 있습니다. 상한 임계값이 0.9이고 하한 임계값이 0.8인 경우 예외 변환은 해당 레코드를 고유한 레코드 테이블에 기록합니다.

## 중복 레코드 예외 구성 보기

일치 점수 임계값을 정의하고 중복 레코드 예외 변환이 다른 유형의 출력 데이터를 기록할 위치를 구성합니다.

다음 그림은 구성할 수 있는 속성을 보여 줍니다.

Manual Review Thresholds

Lower Threshold : 0.80
Upper Threshold : 0.95

Data Routing Options

Type	Standard Output	Duplicate Record Table
Automatic Consolidation (Above upper threshold)	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Manual Consolidation (Between thresholds)	<input type="checkbox"/>	<input checked="" type="checkbox"/>
Unique Records (Below lower threshold)	<input type="checkbox"/>	<input type="checkbox"/>

☐ Create separate output group for unique records

Generate duplicate record table

다음과 같은 속성을 구성할 수 있습니다.

### 하한 임계값

중복 레코드 점수 범위의 하한입니다. 일치 점수가 이 값보다 낮은 레코드는 고유한 레코드로 처리됩니다. 하한 임계값은 0에서 1 사이의 숫자입니다.

### 상한 임계값

중복 레코드 점수 범위의 상한입니다. 일치 점수가 상한 임계값보다 크거나 같은 레코드는 중복 레코드로 처리됩니다. 상한 임계값은 하한 임계값의 숫자보다 큰 숫자입니다.

### 자동 통합

모든 레코드의 일치 점수가 상한 임계값보다 큰 클러스터입니다. 자동 통합 클러스터는 검토가 필요하지 않으며, 중복 레코드를 포함합니다. 이러한 레코드는 통합 변환을 사용하여 결합할 수 있습니다. 기본적으로 중복 레코드 예외 변환은 자동 통합 클러스터를 표준 출력 포트에 기록합니다.

### 수동 통합

모든 레코드의 일치 점수가 하한 임계값보다 크거나 같고 최소 1개 레코드의 일치 점수가 상한 임계값보다 작은 클러스터입니다. 이러한 클러스터는 수동 검토를 수행하여 중복 레코드의 포함 여부를 결정해야 합니다. 기본적으로 중복 레코드 예외 변환은 수동 통합 레코드를 중복 레코드 테이블에 기록합니다.

### 고유한 통합

클러스터 크기가 1이거나 일치 점수가 하한 임계값보다 낮은 레코드를 포함하는 클러스터입니다. 고유한 레코드 클러스터는 중복이 아닙니다. 기본적으로 중복 레코드 예외 변환은 고유한 레코드를 출력 테이블에 기록하지 않습니다.

### 표준 출력

표준 출력 포트에 기록되는 레코드 유형입니다.

기본값은 자동 통합 레코드입니다.

### 중복 레코드 테이블

중복 레코드 출력 포트에 기록되는 레코드 유형입니다. 기본값은 수동 통합 레코드입니다.

### 고유한 레코드에 대해 별도의 출력 그룹 작성

고유한 레코드에 대해 별도의 출력 그룹을 작성합니다. 고유한 레코드에 대해 별도의 테이블을 작성하지 않는 경우 고유한 레코드를 다른 그룹 중 하나에 기록하도록 변환을 구성할 수 있습니다. 또는 고유한 레코드를 출력 테이블에 기록하는 작업을 건너뛸 수 있습니다. 기본값은 비활성화됩니다.

## 중복 레코드 테이블 생성

중복 레코드 클러스터 데이터를 포함할 데이터베이스 개체를 작성합니다. 이 옵션을 선택하면 **Developer tool**이 데이터베이스 개체를 작성합니다. **Developer tool**은 이 개체를 모델 리포지토리에 추가하고 개체의 인스턴스를 매핑 캔버스에 추가하고 포트를 개체에 연결합니다.

## 중복 레코드 테이블 생성

매핑의 중복 레코드 예외 변환 인스턴스로부터 중복 레코드 테이블을 생성할 수 있습니다.

1. 구성 보기에서 **중복 레코드 테이블 생성**을 클릭하여 테이블을 생성합니다.  
**관계형 데이터 개체 작성** 대화 상자가 표시됩니다.
2. 테이블을 포함할 데이터베이스에 대한 연결을 찾아보고 선택합니다.
3. 중복 레코드 테이블의 이름을 데이터베이스에 입력합니다.
4. 중복 레코드 테이블 개체의 이름을 모델 리포지토리에 입력합니다.
5. **마침**을 클릭합니다.  
**Developer tool**이 새 테이블을 매핑 캔버스에 추가합니다.

## 포트

중복 레코드 예외 변환에는 여러 그룹의 입력 및 출력 포트가 있습니다.

다음 그림은 입력 및 출력 포트의 예를 보여줍니다.

Ports								
	Name	Type	Precision	Scale	Input	Output	Default	Description
Inputs								
Data (3)								
1	Employee	decimal	10	0	<input checked="" type="checkbox"/>	<input type="checkbox"/>		
2	Name	string	50	0	<input checked="" type="checkbox"/>	<input type="checkbox"/>		
3	Addr1	string	50	0	<input checked="" type="checkbox"/>	<input type="checkbox"/>		
Control (3)								
1	Score	double	15	0	<input checked="" type="checkbox"/>	<input type="checkbox"/>		
2	Row_Identifier	string	25	0	<input checked="" type="checkbox"/>	<input type="checkbox"/>		
3	Cluster_ID	integer	10	0	<input checked="" type="checkbox"/>	<input type="checkbox"/>		
Outputs								
Standard Output (6)								
1	Score	double	15	0	<input type="checkbox"/>	<input checked="" type="checkbox"/>		
2	Row_Identifier	string	25	0	<input type="checkbox"/>	<input checked="" type="checkbox"/>		
3	Cluster_ID	integer	10	0	<input type="checkbox"/>	<input checked="" type="checkbox"/>		
4	Employee	decimal	10	0	<input type="checkbox"/>	<input checked="" type="checkbox"/>		
5	Name	string	50	0	<input type="checkbox"/>	<input checked="" type="checkbox"/>		
6	Addr1	string	50	0	<input type="checkbox"/>	<input checked="" type="checkbox"/>		
Cluster Data (9)								
1	Row_Identifier	bigint	19	0	<input type="checkbox"/>	<input checked="" type="checkbox"/>		
2	Sequential_Cl...	bigint	19	0	<input type="checkbox"/>	<input checked="" type="checkbox"/>		
3	Cluster_ID	integer	10	0	<input type="checkbox"/>	<input checked="" type="checkbox"/>		

## 중복 레코드 예외 변환 입력 포트

중복 레코드 예외 변환에는 입력 포트의 데이터 그룹 및 제어 그룹이 있습니다.

**데이터** 그룹에는 소스 데이터를 수신하는 사용자 정의 포트가 포함되어 있습니다.

**제어** 포트는 일치 변환이 소스 데이터에 추가하는 메타데이터를 수신합니다. 다음 표에는 **제어** 포트가 설명되어 있습니다.

포트	설명
점수	0과 1 사이의 10진수 값입니다. 레코드를 클러스터에 연결한 레코드의 유사점 정도를 식별합니다.
Row_Identifier	레코드의 고유한 식별자입니다.
Cluster_ID	레코드가 속하는 일치 클러스터의 ID입니다.

## 중복 레코드 예외 변환의 출력 포트

중복 레코드 예외 변환에는 여러 개의 출력 그룹이 있습니다. 기본적으로 변환은 중복 레코드를 **표준 출력** 그룹에 기록합니다. 가능한 일치는 **클러스터 데이터** 그룹에 기록합니다. 고유한 레코드에 대한 출력 그룹을 추가할 수 있습니다.

**구성** 보기의 기본 설정을 변경하여 변환이 출력 포트에 기록하는 레코드 유형을 변경할 수 있습니다.

다음 표에는 **표준 출력** 그룹에 대한 출력 포트가 설명되어 있습니다.

포트	설명
점수	0과 1 사이의 10진수 값입니다. 클러스터에 있는 레코드 간의 유사점 정도를 식별합니다.
Row_Identifier	레코드의 고유한 식별자입니다.
Cluster_ID	일치 변환이 레코드를 할당한 클러스터의 ID입니다.
사용자 정의 포트	소스 데이터 필드입니다.

다음 표에는 **클러스터 데이터** 그룹의 출력 포트가 설명되어 있습니다.

포트	설명
Row_Identifier	레코드의 고유한 식별자입니다.
Sequential_Cluster_ID	휴먼 태스크의 클러스터를 식별합니다. 시퀀스 클러스터 ID는 워크플로우가 휴먼 태스크의 인스턴스에 클러스터를 할당할 때 사용됩니다.
Cluster_ID	레코드가 속한 클러스터를 식별합니다. 일치 변환은 모든 레코드에 클러스터 ID를 할당합니다.
점수	0과 1 사이의 10진수 값입니다. 레코드를 클러스터에 연결한 레코드의 유사점 정도를 식별합니다.
Is_Master	레코드가 클러스터의 기본 설정 레코드인지 여부를 나타내는 문자열 값입니다. 기본적으로 클러스터의 첫 번째 행이 기본 설정 레코드입니다. 값은 Y 또는 N입니다.

포트	설명
Workflow_ID	태스크의 레코드에 대한 워크플로우를 식별하는 ID입니다. 워크플로우에 포함되지 않는 매핑을 실행하는 경우 워크플로우 ID는 DummyWorkflowID입니다.
사용자 정의 포트	소스 데이터 포트입니다.

## 포트 작성

각 입력 포트를 데이터 그룹에 추가합니다. 입력 포트를 추가하면 **Developer tool**이 동일한 이름의 출력 포트를 표준 출력 그룹, 클러스터 데이터 그룹 및 고유한 레코드 그룹에 추가합니다.

1. 데이터 입력 그룹을 선택합니다.  
이 그룹이 강조 표시됩니다.
2. **새로 만들기(삽입)**을 클릭합니다.  
**Developer tool**이 데이터 그룹, 표준 출력 그룹, 클러스터 데이터 그룹 및 고유한 레코드 그룹에 필드를 추가합니다.
3. 필요에 따라 필드의 이름을 변경합니다.  
**Developer tool**이 다른 그룹의 필드 이름을 변경합니다.
4. 데이터 소스에 대해 추가해야 하는 나머지 포트를 입력합니다.

## 중복 레코드 예외 변환의 고급 속성

중복 레코드 예외 변환에는 정렬 동작, 캐시 메모리 동작 및 추적 수준을 결정하는 고급 속성이 포함되어 있습니다.

다음과 같은 고급 속성을 구성할 수 있습니다.

### 정렬

변환이 **클러스터 ID** 포트 데이터의 입력 행을 정렬할지 여부를 결정합니다. 이 속성은 기본적으로 활성화됩니다.

입력 행이 미리 정렬되어 있지 않은 경우 이 속성을 선택합니다.

### 캐시 파일 디렉터리

데이터 통합 서비스에서 현재 변환에 대해 임시 데이터를 쓸 디렉터리를 지정합니다. 입력 데이터의 양이 사용 가능한 시스템 메모리보다 클 경우 데이터 통합 서비스가 이 디렉터리에 임시 파일을 기록합니다. 데이터 통합 서비스는 매핑을 실행한 후에 임시 파일을 삭제합니다.

속성에 디렉터리 경로를 입력하거나 매개 변수를 사용하여 디렉터리를 식별할 수 있습니다. 데이터 통합 서비스 시스템의 로컬 경로를 지정하십시오. 데이터 통합 서비스가 기록할 수 있는 디렉터리를 지정해야 합니다. 기본값은 **CacheDir** 시스템 매개 변수입니다.

### 캐시 파일 크기

변환의 입력 데이터를 정렬하기 위해 데이터 통합 서비스에서 사용하는 시스템 메모리의 양을 결정합니다. 기본값은 **400,000**바이트입니다.

데이터 통합 서비스는 데이터를 정렬하기 전에 사용자가 지정한 메모리 양을 할당합니다. 정렬 작업이 더 많은 데이터를 생성할 경우 데이터 통합 서비스는 초과 데이터를 캐시 파일 디렉터리에 기록합니다. 정렬 작업

에 시스템 메모리 및 파일 저장소가 제공할 수 있는 것보다 더 많은 메모리가 필요할 경우 매핑이 실패합니다.

**참고:** 65536 이상의 값을 입력한 경우 일치 변환이 값을 바이트로 읽습니다. 이보다 낮은 값을 입력하면 일치 변환이 값을 메가바이트로 읽습니다.

#### 추적 수준

이 변환에 대해 로그에 표시되는 세부 정보의 양입니다. 간단, 보통, 자세한 정보 표시 초기화 또는 자세한 정보 표시 데이터를 선택할 수 있습니다. 기본값은 보통입니다.

## 중복 레코드 예외 매핑 예제

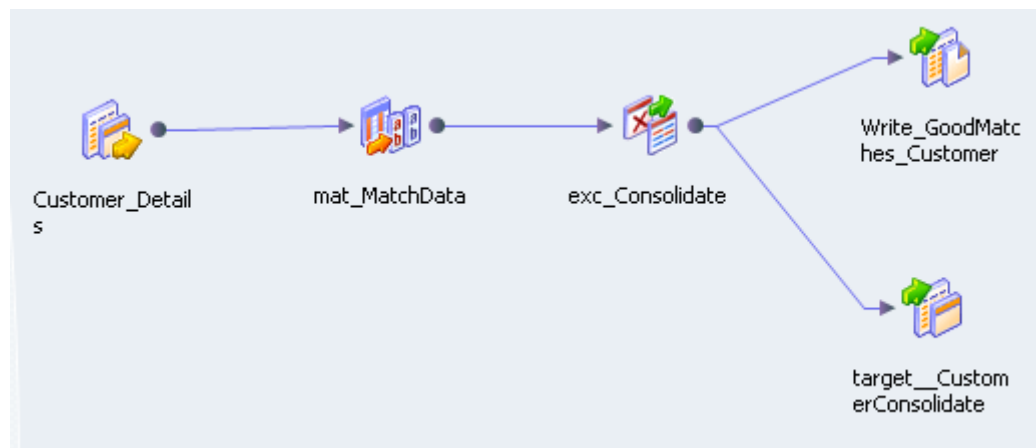
고객 데이터를 검토하는 데이터 프로젝트를 실행하는 조직이 있습니다. 이 조직은 고객 데이터에 여러 중복 레코드가 포함되는지 여부를 확인해야 합니다. 또한 중복일 수 있는 일부 레코드를 수동으로 검토해야 합니다.

이를 위해 중복 고객 레코드를 식별하는 데이터 품질 매핑을 작성합니다. 매핑에는 일치 변환이 포함됩니다. 중복 레코드 예외 변환은 일치 변환의 결과를 수신합니다. 중복 레코드 예외 변환은 상태가 분명하지 않은 각 레코드 클러스터를 데이터베이스 테이블에 기록합니다. 데이터 분석가는 **Analyst** 도구에서 데이터를 검토하고 중복 레코드를 결정합니다.

### 중복 레코드 예외 매핑

고객 레코드를 검사하고 중복 레코드를 찾는 중복 레코드 예외 매핑을 구성합니다.

다음 그림은 중복 레코드 예외 매핑을 보여 줍니다.



이 매핑에는 다음 개체가 포함됩니다.

#### Customer\_Details

중복 레코드가 포함되었을 수 있는 데이터 소스입니다.

#### mat\_MatchData

고객 데이터를 검사하여 레코드 일치 여부를 결정하는 일치 변환입니다. 이 일치 변환은 두 열 값 간의 유사점 정도를 나타내는 숫자 점수를 작성합니다. 알고리즘은 0~1 범위의 10진수 값으로 일치 점수를 계산합니다. 두 열 값이 동일할 경우 점수 1이 할당됩니다.

## exc\_Consolidate

레코드 중에서 가능한 중복 고객 레코드, 알려진 중복 고객 레코드 또는 고유한 중복 레코드를 결정하는 중복 레코드 예외 변환합니다.

## Write\_GoodMatches\_Customer 테이블

수동 검토가 필요하지 않은 모든 레코드를 수신하는 테이블입니다. 중복 레코드 예외 변환은 중복 레코드와 고유한 레코드를 이 테이블에 기록합니다.

## Target\_CustomerConsolidate 테이블

중복 레코드 예외 변환은 가능한 중복 레코드를 Target\_CustomerConsolidate 테이블에 기록합니다. 이 테이블의 레코드는 Analyst 도구에서 수동으로 검토해야 합니다.

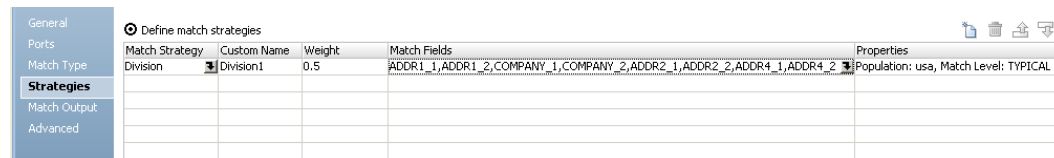
## 일치 변환

일치 변환은 고객 데이터를 수신하고 ID 일치 분석을 수행합니다.

일치 변환의 출력 유형을 클러스터 - 모두 일치로 구성하십시오. 일치 변환은 일치하는 레코드를 클러스터로 반환합니다. 클러스터의 각 레코드는 클러스터에서 최소 1개의 다른 레코드와 일치해야 하며 점수 일치가 일치 임계값보다 크거나 같아야 합니다. 일치 임계값은 0.75입니다.

일치 변환의 전략 탭에서 부서 일치 전략을 선택하십시오. 부서 전략은 주소 필드를 바탕으로 조직을 식별하는 미리 정의된 일치 전략입니다. 일치 변환의 전략 탭에서 일치 검사를 수행할 입력 포트를 선택하고 전략 가중치를 0.5로 구성하십시오.

다음 그림은 일치 변환의 부서 전략 구성을 보여 줍니다.



Match Strategy	Custom Name	Weight	Match Fields	Properties
Division	Division1	0.5	ADDR1_1, ADDR1_2, COMPANY_1, COMPANY_2, ADDR2_1, ADDR2_2, ADDR4_1, ADDR4_2	Population: usa, Match Level: TYPICAL

일치 변환은 클러스터 정보를 각 출력 레코드에 추가합니다. 또한 고유한 RowID를 각 레코드에 추가합니다.

## 중복 레코드 예외 입력 그룹

중복 레코드 예외 변환에는 2개의 입력 그룹이 있습니다. 고객 데이터를 수신하는 데이터 그룹과 행의 일치 점수, 행 식별자 및 클러스터 ID를 포함하는 제어 그룹입니다.

다음 그림은 예외 변환의 입력 그룹을 보여 줍니다.



	Name	Type	Precision	Scale	Input	Output	Default	Description
Inputs								
Data (11)								
1	CUST_ID	decimal	20	0	<input checked="" type="checkbox"/>	<input type="checkbox"/>		
2	COMPANY	string	200	0	<input checked="" type="checkbox"/>	<input type="checkbox"/>		
3	CONTACT	string	200	0	<input checked="" type="checkbox"/>	<input type="checkbox"/>		
4	TITLE	string	200	0	<input checked="" type="checkbox"/>	<input type="checkbox"/>		
5	ADDR1	string	200	0	<input checked="" type="checkbox"/>	<input type="checkbox"/>		
6	ADDR2	string	100	0	<input checked="" type="checkbox"/>	<input type="checkbox"/>		
7	ADDR3	string	100	0	<input checked="" type="checkbox"/>	<input type="checkbox"/>		
8	ADDR4	string	50	0	<input checked="" type="checkbox"/>	<input type="checkbox"/>		
9	COUNTRY	string	50	0	<input checked="" type="checkbox"/>	<input type="checkbox"/>		
10	PHONE	string	100	0	<input checked="" type="checkbox"/>	<input type="checkbox"/>		
11	EMAIL	string	100	0	<input checked="" type="checkbox"/>	<input type="checkbox"/>		
Control (3)								
1	Score	double	15	0	<input checked="" type="checkbox"/>	<input type="checkbox"/>		
2	Row_Identifier	string	25	0	<input checked="" type="checkbox"/>	<input type="checkbox"/>		
3	Cluster_ID	integer	10	0	<input checked="" type="checkbox"/>	<input type="checkbox"/>		

데이터 그룹에는 고객 데이터가 포함됩니다. 고객 데이터에는 고객 ID, 연락처, 직위 및 주소 필드가 포함됩니다. 제어 그룹은 일치 변환이 각 고객 레코드에 추가한 추가 메타데이터입니다. 제어 그룹에는 일치 점수, 행 ID 및 클러스터 ID가 포함됩니다.

## 중복 레코드 예외 예의 구성 보기

구성 보기에서 상한 및 하한 임계값을 정의합니다. 변환이 중복 고객 레코드, 가능한 중복 레코드 및 고유한 고객 레코드를 기록할 위치를 식별합니다.

다음 그림은 중복 레코드 예외 변환의 구성 보기를 보여 줍니다.

Manual Review Thresholds

Lower Threshold : 
Upper Threshold :

Data Routing Options

Type	Standard Output	Duplicate Record Table
Automatic Consolidation (...)	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Manual Consolidation (Be ...)	<input type="checkbox"/>	<input checked="" type="checkbox"/>
Unique Records (Below lo...	<input checked="" type="checkbox"/>	<input type="checkbox"/>

☐ Create separate output group for unique records

Generate duplicate record table

다음 표에는 구성 설정이 설명되어 있습니다.

옵션	설정
하한 임계값	.80
상한 임계값	.95
자동 통합	표준 출력 테이블
수동 통합	중복 레코드 테이블
고유한 레코드	표준 출력 테이블

**중복 레코드 테이블 생성**을 클릭하여 중복 레코드 테이블을 작성합니다. 고유한 레코드에 대한 별도의 테이블을 작성하지 마십시오. 고유한 레코드는 표준 출력 테이블에 기록됩니다.

## 표준 출력 테이블 레코드

Write\_GoodMatches\_Customer 대상 테이블은 표준 출력 그룹에서 행을 받습니다. 테이블은 고유 레코드와 중복 레코드를 모두 받습니다. 이러한 레코드는 수동으로 검토할 필요가 없습니다.

다음 그림에는 예외 변환이 반환하는 표준 출력 레코드가 나와 있습니다.

Output										
Name: <a href="#">lexc.Consolidate.Good.Records</a>										
Score	Row_Identifier	Cluster_ID	CUST_ID	COMPANY	CONTACT	TITLE	ADDR1	ADDR2	ADDR3	
1	1 - 1001590	1	1001590	SHOP'N SAVE	MR DANIEL COWLEY	SR COMPUTER ...	1755 WABASH...	SPRINGFIELD	<null>	
2	1 - 1001599	2	1001599	SHOP'N SAVE	MS VERENE TEKAUTZ	NETWORK SYS...	45 GRAVOIS BL...	FENTON	MO	
3	1 - 1001604	3	1001604	SHOP'N SAVE	MR REUBEN HENDRICKS	GENERAL MAN...	800 CARLYLE A...	BELEVILLE	IL	
4	1 - 1001622	4	1001622	SHOPRITE	MS JACKY DAGENHART	INFO SYSTEMS...	60 BEAVER BR...	LINCOLN PARK	NJ	
5	1 - 7121564	5	7121564	SPAR	MR BRADLY THOMAS	COMP SPEC	FORE STR	ST DENNIS	CORNWALL	
6	0.850000000000...	1 - 7121565	5	7121565	SPAR	MS PEARL GARRISON	SURVEYING TE...	FORE STR	ST AUSTELL	BUGLE, CORN...
7	0.840000000000...	1 - 7121567	5	7121567	SPAR	MR EARL WILLIS	TECHNICIAN (C...	FORE STREET,...	BUNGLE	CORNWALL
8	0.779999999999...	1 - 7121597	5	7121597	SPAR	MRS JERRY FIGURES	LEAD SYS ANAL...	6 FORE STR	BUDLEIGH SAL...	DEVON
9	0.77	1 - 7121590	5	7121590	SPAR	MRS MINNA HASE	SR RESEARCH...	42 FORE STR	NEWTON ABBOT	BOVEY TRACEY.
10	1 - 7121570	6	7121570	SPAR	MR GERMAYNE HANKS	SENIOR PROJE...	1 2 CHRISTINA...	TOTNESS	DEVON	
11	1 - 7121571	7	7121571	SPAR	MRS CHERIE ALBERSON	CONTROLLER	1 N STR	ASHBURTON	DEVON	
12	0.96	1 - 7121608	7	7121608	SPAR	MRS CHERIE ALBERSON	CONTROLLER	N STR	ASHBURTON	DEVON
13	1 - 1001658	9	1001658	SPARTAN STOR...	MS MARTHA CHASSE	PROJECT MANA...	5161 W MAIN S...	KALAMAZOO	MI	
14	1 - 1001660	10	1001660	SPARTAN STOR...	MR DESMOND WINTERS	OWNER / CON...	1824 PORTAGE...	KALAMAZOO	MI	
15	1 - 1001659	10	1001659	SPARTAN STOR...	MR DESMOND GRINDLEY	SYSTEMS ANAL...	1824 PORTAGE...	KALAMAZOO	MI	
16	1 - 1001691	11	1001691	SPARTAN STOR...	MS SLUZY TRAPANI	MGR ENG. SAL...	2545 W CADILA...	FARWELL	MI	
17	1 - 1001664	12	1001664	SPARTAN STOR...	MR HARDEN DALTON	OWNER / CON...	438 LINCOLN A...	IONIA	MI	
18	1 - 1001694	13	1001694	SPARTAN STOR...	MR REAMONN PELZER	CHIEF DBA	24445 DRAKE RD	FARMINGTON...	MI	
19	1 - 1001729	14	1001729	SUN MART	MS ZELDA DECHICK	MIS MANAGER	1100 13TH AVE E	WEST FARGO	ND	
20	1 - 1001724	15	1001724	SUN MART	MS MINERVA BAKER	COMPUTER SPE...	1921 W A STR	NORTH PLATTE	NE	
21	1 - 1001732	16	1001732	SUPER FRESH	MS MARGA JANOWSKI	PRESIDENT	2105 PHILADEL...	CLAYMONT	DE	
22	1 - 1001736	17	1001736	SUPER FRESH	MS BETTY EASTIP	ASSISTANT	2465 S BROAD...	HAMILTON TO...	NJ	
23	1 - 1001738	18	1001738	SUPER FRESH	MS MINICA HOCH	MGR COORD	4330 48TH STR...	WASHINGTON	DC	
24	1 - 1001758	19	1001758	SUPER STOP &...	MR ANCILIN SAMSON	AUTOMATED S...	150 NEW PK AVE	HARTFORD	CT	
25	1 - 1001767	20	1001767	SUPERPETZ	MR EZARRAS DELAHANTY	LEAD PROGRA...	250 SHOUP MIL...	DAYTON	OH	
26	1 - 1001796	21	1001796	SUPERPETZ	MR RUSTICUS WHITACRE	DATABASE AD...	1275 W PATRIC...	FREDERICK	MD	
27	1 - 1001798	22	1001798	SUPERPETZ	MS TIMMIE NICE	NETWORK SYS...	2420 EASTERN...	YORK	PA	

이 레코드에는 다음 필드가 포함됩니다.

### 점수

클러스터에 있는 레코드 간의 유사점 정도를 나타내는 일치 점수입니다. 일치 점수가 1인 레코드는 검토할 필요가 없는 중복 레코드입니다. 레코드의 일치 점수가 하한 임계값보다 낮은 클러스터는 중복 클러스터가 아닙니다.

### Row\_Identifier

테이블의 각 행을 고유하게 식별하는 행 번호입니다. 이 예제에서 행 식별자에는 고객 ID가 포함됩니다.

## 클러스터 ID

클러스터에 대한 고유한 식별자입니다. 클러스터의 각 레코드는 같은 클러스터 ID를 받습니다. 샘플 출력 데이터의 처음 4개 레코드는 고유합니다. 각 레코드의 클러스터 ID는 고유합니다. 행 5~9는 클러스터 5에 속합니다. 이 클러스터의 각 레코드는 주소 필드가 유사하므로 중복 레코드입니다.

## 소스 데이터 필드

표준 출력 테이블 그룹은 모든 소스 데이터 필드도 받습니다.

# 클러스터 출력

Target\_CustomerConsolidate 테이블은 클러스터 출력 그룹의 레코드를 수신합니다. 클러스터 출력 그룹은 중복 레코드일 수 있는 레코드를 반환합니다. Target\_CustomerConsolidate 테이블의 레코드는 Analyst 도구에 서 수동으로 검토해야 합니다.

다음 이미지는 Target\_CustomerConsolidate 테이블의 일부 레코드와 필드를 보여 줍니다.

Output												
Name: <a href="#">excc_Consolidate_Cluster_Data</a>												
	Ro...	Sequential...	Cluster_ID	Score	Is_Master	Workflow...	CUST_ID	COMPANY	CONTACT	TITLE	ADDR1	ADDR2
1	0	0	8	1	Y	Dummy/W...	7121580	SPAR	MR MICHAEL AVELINO	CONTROLLER	219 VIRGEN BELLA FLOR...	LEPE
2	1	0	8	1	N	Dummy/W...	7121580	SPAR	MR MICHAEL AVELINO	CONTROLLER	219 VIRGEN BELLA FLOR...	LEPE
3	2	0	8	0.81	N	Dummy/W...	7121585	SPAR	MR ROBERT ADAMS	MANAGER OF DBA	3 VIRGEN BELLA FDL	LEPE
4	3	1	26	1	Y	Dummy/W...	1001921	THE CORNER SHOP	MR DENIS LEE	MANAGER (\$\$ OF DBA	102 MID STR 5 NUTFIELD	REDHILL
5	4	1	26	1	N	Dummy/W...	1001921	THE CORNER SHOP	MR DENIS LEE	MANAGER (\$\$ OF DBA	102 MID STR 5 NUTFIELD	REDHILL
6	5	1	26	0.88	N	Dummy/W...	7121633	THE CORNER SHOP	MR BRADLY WYATT	MGR, DBA	102 MID STR 5 NUTFIELD	REDHILL
7	6	1	26	0.81	N	Dummy/W...	7121634	THE CORNER SHOP	MR BRADLY LOPEZ	OWNER/CONSULTANT	971 MID STR 5 NUTFIELD	REDHILL
8	7	2	74	1	Y	Dummy/W...	1001922	TOPS MARKETS	MS SARAD SACKRIDER	PRESIDENT	22777 ROCKSIDE RD	BEDFORD
9	8	2	74	1	N	Dummy/W...	1001922	TOPS MARKETS	MS SARAD SACKRIDER	PRESIDENT	22777 ROCKSIDE RD	BEDFORD
10	9	2	74	0.860000000000...	N	Dummy/W...	1001640	TOPS MARKETS	MS SARAH ALLEN	INDUSTRIAL ENGINEER	77 ROCKSIDE RD	BEDFORD
11	10	3	81	1	Y	Dummy/W...	1777777	WILLIAM WRIGLEY...	MR JUAN WISNIEWSKI	<null>	C. MIS 410 N MICHIGAN...	CHICAGO
12	11	3	81	1	N	Dummy/W...	1777777	WILLIAM WRIGLEY...	MR JUAN WISNIEWSKI	<null>	C. MIS 410 N MICHIGAN...	CHICAGO
13	12	3	81	0.93	N	Dummy/W...	1002174	WILLIAM WRIGLEY...	MR JUAN WISNIEWSKI	LEAD PROGRAMMER A...	CORPORATE MIS 410 N M...	CHICAGO
14	13	3	81	0.93	N	Dummy/W...	1002153	WILLIAM WRIGLEY...	MR JOHN WISNIEWSKI	MGR COORD	CORPORATE MIS 410 N M...	CHICAGO
15	14	3	81	1	N	Dummy/W...	1002211	WILLIAM WRIGLEY...	MR JON WISNIEWSKI	MGR COORD	C. MIS 410 N MICHIGAN...	CHICAGO
16	15	3	81	0.93	N	Dummy/W...	1002210	WILLIAM WRIGLEY...	MR JOHNNY WISNIEWSKI	BUSINESS ANALYST	CORPORATE MIS 410 N M...	CHICAGO
17	16	3	81	0.93	N	Dummy/W...	1002142	WILLIAM WRIGLEY...	MR JOHN WISNIEWSKI	MGR COORD	CORPORATE MIS 410 N M...	CHICAGO
18	17	4	106	1	Y	Dummy/W...	1000051	A&P	MS SARA ONEAL	BUSINESS MANAGER	120 MAIN RD	MANVILLE
19	18	4	106	1	N	Dummy/W...	1000051	A&P	MS SARA ONEAL	BUSINESS MANAGER	120 MAIN RD	MANVILLE
20	19	4	106	0.850000000000...	N	Dummy/W...	1000089	A&P	MR BOBBY SMYTHE	PARTNER	120 N MAYNE	MANVILLE
21	20	4	106	0.99	N	Dummy/W...	1000096	A&P	MS JENNY BEASLEY	LEAD SYSTEM ANALYST	120 N MAIN RD	MANVILLE
22	21	5	135	1	Y	Dummy/W...	1001628	SIMS DELTEC INC	MR KEN YOUNGQUIST	CONTROLLER	1265 GRAY FOX STR	SAINT PAUL
23	22	5	135	1	N	Dummy/W...	1001628	SIMS DELTEC INC	MR KEN YOUNGQUIST	CONTROLLER	1265 GRAY FOX STR	SAINT PAUL

이 레코드에는 다음 필드가 포함됩니다.

## Row\_Identifier

테이블의 각 행을 고유하게 식별하는 숫자입니다.

## 시퀀스 클러스터 ID

휴먼 태스크로 검토해야 하는 각 클러스터에 대한 시퀀스 식별자입니다. 중복 레코드 예외 변환은 클러스터 데이터 출력 그룹의 레코드에 시퀀스 클러스터 ID를 추가합니다.

## 클러스터 ID

클러스터에 대한 고유한 식별자입니다. 일치 변환은 모든 출력 레코드에 클러스터 ID를 할당합니다. 중복 레코드 및 가능한 중복 레코드는 클러스터 ID를 공유합니다. 고유한 레코드에도 클러스터 ID가 추가되지만 다른 레코드와 ID 번호를 공유하지 않습니다.

## 점수

클러스터에 있는 레코드 간의 유사점 정도를 나타내는 일치 점수입니다. 수동 검토가 필요한 레코드의 일치 점수는 0.95보다 작거나 0.80보다 큼니다.

## 마스터 여부

레코드가 클러스터에서 기본 설정 레코드인지 여부를 나타냅니다.

## WorkflowID

변환이 워크플로우에 포함되지 않으므로 WorkflowID는 DummyWorkflowID입니다.

## 레코드 필드

레코드의 다른 필드에는 고객 소스 데이터가 포함됩니다.

# 중복 레코드 예외 변환 작성

중복 레코드 예외 변환을 구성할 때는 입력 포트를 구성해야 합니다. 일치를 결정하는 상한 및 하한 임계값을 정의하고, 중복 레코드 및 고유한 레코드를 기록할 위치도 구성합니다.

1. 재사용 가능 또는 재사용 불가능한 중복 레코드 예외 변환을 작성합니다.
  - 재사용 가능 변환을 작성하려면 **파일 > 새로 만들기 > 변환**을 선택하고 중복 레코드 예외 변환을 선택합니다.
  - 재사용할 수 없는 변환을 작성하려면 매핑을 열고 매핑 캔버스에 변환을 추가합니다. 마법사에서 중복 레코드 예외 변환을 선택합니다.
2. **다음**을 클릭하거나 **마침**을 클릭합니다.

**마침**을 클릭하는 경우 변환을 작성하기 전에 기본 임계값 및 데이터 라우팅 옵션을 업데이트할 수 있습니다.
3. 구성 보기에서 상한 및 하한 일치 점수 임계값을 구성합니다.
4. **데이터 라우팅 옵션** 섹션에서 표준 출력 및 예외 테이블 속성을 구성하여 변환이 각 유형의 레코드를 기록할 위치를 설정합니다.

필요한 경우 중복 레코드, 검토할 중복 레코드 및 고유한 레코드를 기록할 위치를 수정합니다.
5. 필요한 경우 고유한 레코드 테이블을 생성합니다. 새 테이블에 대한 데이터베이스 연결 및 테이블 이름 정보를 입력합니다. 고유한 레코드 테이블을 생성하면 변환이 모델 리포지토리에 데이터베이스 개체를 작성합니다.
6. 입력 포트를 구성합니다. 입력 포트를 추가하면 **Developer tool**이 동일한 포트 이름을 출력 그룹에 추가합니다.
  - 재사용 가능한 변환을 작성하는 경우 **포트** 탭을 선택하고 연결할 데이터의 포트를 변환에 추가합니다.
  - 재사용할 수 없는 변환을 작성하는 경우 다른 개체를 매핑 캔버스에 추가하고 입력 포트를 변환으로 끌어옵니다.
7. 변환 출력 포트를 하나 이상의 데이터 대상에 연결합니다. **구성** 보기에서 설정한 출력 옵션에 해당하는 데이터 개체에 출력 포트를 연결합니다.
  - 재사용 가능한 변환을 작성하는 경우 변환을 매핑에 추가하고 출력 포트를 연결합니다.
  - 재사용할 수 없는 변환을 작성하는 경우 변환이 포트를 클러스터 데이터 테이블에 연결합니다. 다른 데이터 대상에는 사용자가 출력 포트를 연결해야 합니다.

## 제 16 장

# 식 변환

이 장에 포함된 항목:

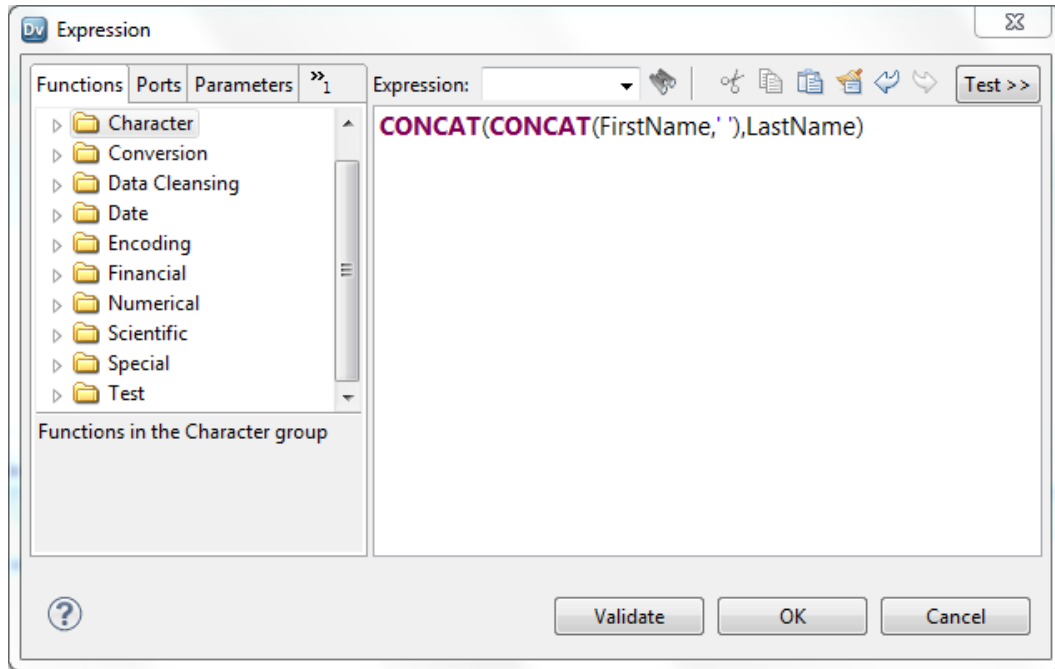
- [식 변환 개요, 257](#)
- [식 변환 포트, 258](#)
- [식 테스트, 259](#)
- [포트 선택기, 260](#)
- [창 작업, 263](#)
- [동적 식, 267](#)
- [식 변환 고급 속성, 271](#)
- [식 변환 - 비원시 환경, 272](#)

## 식 변환 개요

식 변환은 행에서 계산을 수행하거나 조건문을 테스트하는 데 사용할 수 있는 수동 변환입니다. 재사용 불가능 식 변환에서 매핑 출력을 정의하는 경우 집계할 매핑 출력 식을 정의할 수 있습니다.

단일 행에서 직원 급여를 조정하거나 이름과 성을 연결하거나 문자열을 숫자로 변환하는 식을 작성해야 할 수 있습니다.

다음 그림에서는 이름, 공백 및 성을 연결하는 식 변환의 식을 보여 줍니다.



각 출력 포트마다 식을 작성하여 식 변환에 여러 식을 입력할 수 있습니다. 예를 들어 각 직원 급여에서 지방 및 국가 소득세와 같은 여러 유형의 세금을 계산하려고 할 수 있습니다. 두 세금 계산에는 직원 급여와 세율이 필요합니다. 각 계산에 대해 별도의 출력 포트를 정의합니다. 각 출력 포트에 대해 서로 다른 식을 정의합니다. 포트 값은 변경되지 않으므로 급여 및 세율에 대해 통과 포트를 정의할 수 있습니다.

## 식 변환 포트

식 변환에는 식을 정의할 때 참조할 수 있는 여러 포트 유형이 있습니다.

식 변환에는 다음 포트 유형이 있습니다.

### 입력

업스트림 변환의 데이터를 수신합니다. 식 변환이 포트 값을 변경하지 않는 경우 입력 포트 대신 통과 포트를 정의할 수 있습니다.

### 출력

식의 반환 값을 포함합니다. 출력 포트에 대한 구성 옵션으로 식을 입력합니다. 각 포트에 대한 기본값을 구성할 수도 있습니다.

**참고:** 식이 숫자 오류(예: 0으로 나누기 또는 음수의 SQRT)를 야기하는 경우 무한값 또는 NaN 값이 반환됩니다.

### 통과

통과 포트를 정의하여 값을 변경하지 않고 변환을 통해 데이터를 전달합니다. 계산에서 통과 포트를 참조할 수 있지만 통과 포트의 값을 변경할 수는 없습니다.

### 변수

식에서 사용할 데이터를 임시로 저장합니다. 여러 행에 데이터를 저장할 수 있습니다. 식을 정의하여 값을 변수 포트에 반환할 수 있습니다.

## 동적 포트

동적 매핑의 포트를 받거나 반환합니다. 동적 포트는 업스트림 변환에서 한 개 이상의 열을 받고 각 열에 대해 생성된 포트를 작성할 수 있습니다. 동적 출력 포트는 한 개 이상의 생성된 포트를 반환할 수 있습니다. 입력 규칙을 정의하여 동적 포트에서 받는 열을 확인할 수 있습니다. 동적 출력 포트에는 여러 출력 포트를 생성하는 식이 포함될 수 있습니다.

## 생성된 포트

동적 포트 내 단일 열을 나타내는 포트입니다. 식 변환의 생성된 포트는 식 변환이 업스트림 변환에서 받는 열에 따라 변경될 수 있습니다.

# 식 테스트

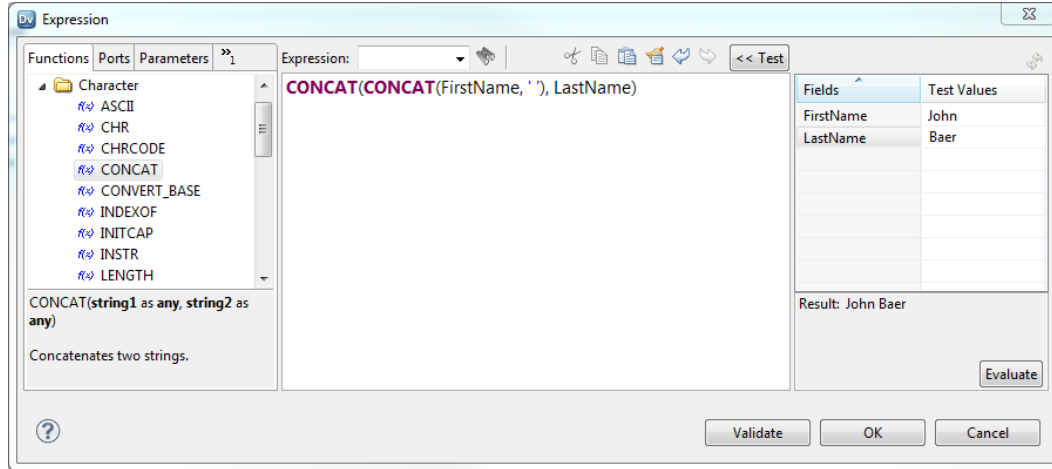
식 편집기에서 구성하는 식을 테스트할 수 있습니다. 식을 테스트하는 경우 샘플 데이터를 입력한 다음 식을 평가합니다.

식을 구성할 때 다음 방법으로 식을 테스트할 수 있습니다.

- 식 변환의 출력 또는 변수 포트에서
- 변환을 매핑에 추가한 후 식 변환의 매핑 출력 보기에서

예를 들어 이름, 공백, 성을 연결하는 식을 구성한 후 포트에 대한 샘플 데이터를 입력한 다음 식을 평가하여 결과를 확인할 수 있습니다.

다음 이미지는 샘플 이름 및 성을 연결하는 식의 결과를 보여 줍니다.



## 샘플 데이터에 대한 날짜 형식 문자열



날짜/시간 또는 시간대가 포함된 타임스탬프 데이터 유형인 포트에 사용하는 식을 테스트하는 경우 필수 날짜 형식 문자열을 사용하여 포트에 대한 샘플 데이터를 입력해야 합니다.

날짜/시간 데이터 유형인 포트에 대해 샘플 데이터를 입력하려면 MM/DD/YYYY HH24:MI:SS 형식을 사용합니다. 식을 평가하면 식에서 지정한 형식을 사용하여 식 편집기에 결과가 표시됩니다. 식에서 형식 문자열을 생략한 경우 동일한 형식 MM/DD/YYYY HH24:MI:SS를 사용하여 식 편집기에 결과가 표시됩니다.

시간대가 포함된 타임스탬프 데이터 유형인 포트에 대한 샘플 데이터를 입력하려면 MM/DD/YYYY HH24:MI:SS TZR 형식을 사용합니다. 식을 평가하면 YYYY-MM-DD HH24:MI:SS.NS TZR 형식을 사용하여 식 편집기에 결과가 표시됩니다.

## 식 테스트

식 편집기에서 식을 테스트하여 식을 평가하고 결과를 확인합니다.

1. 다음 중 한 가지 방법으로 식 편집기를 엽니다.
  - 식 변환에서 출력 포트 또는 변수 포트에 대한 식 열의  열기 단추( )를 클릭합니다.
  - 매핑에 포함된 식 변환을 선택합니다. 매핑 출력 보기에서 출력에 대한 식 열의  열기 단추( )를 클릭합니다.
2. 식을 구성합니다.
3. 테스트 >>를 클릭하여 테스트 패널을 엽니다.
4. 테스트 값 열의 각 필드에 샘플 데이터를 입력합니다.  
식에 포함된 각 포트 또는 매개 변수에 대한 테스트 값을 입력할 수 있습니다.
5. 평가를 클릭합니다.  
테스트 패널 아래쪽에 식 결과가 표시됩니다.

## 포트 선택기

변환이 포트를 생성한 경우 생성된 포트가 변경될 때 실행되도록 변환을 구성해야 합니다. 포트 선택기를 사용하여 동적 식, 조회 조건 또는 조이너 조건에서 사용할 포트를 결정할 수 있습니다.

포트 선택기는 식에서 참조할 수 있는 정렬된 포트 목록입니다. 생성된 포트가 동적 매핑에서 변경되는 경우 포트 선택기에 다른 포트가 포함될 수 있습니다.

예를 들어 다음 식이 동적 매핑에서 생성된 포트를 참조합니다.

`Salary * 12`

동적 소스를 사용하도록 매핑을 구성하지만 각 소스 파일의 급여 정보가 포함된 열에 다른 이름이 있습니다. 열 이름은 `Salary`, `Monthly_Salary` 또는 `Base_Salary`입니다.

다음 태스크를 수행하여 다른 열 이름을 조정합니다.

1. "Salary\_PortSelector"라는 포트 선택기를 작성합니다.
2. 접미사가 "Salary"인 모든 포트 이름을 허용하는 선택 규칙을 작성합니다.
3. 급여 열 이름 대신 포트 선택기 이름을 포함하도록 식을 구성합니다. 식에는 다음 구문이 있습니다.

`Salary_PortSelector * 12`

급여 포트 이름으로 식이 실행됩니다.

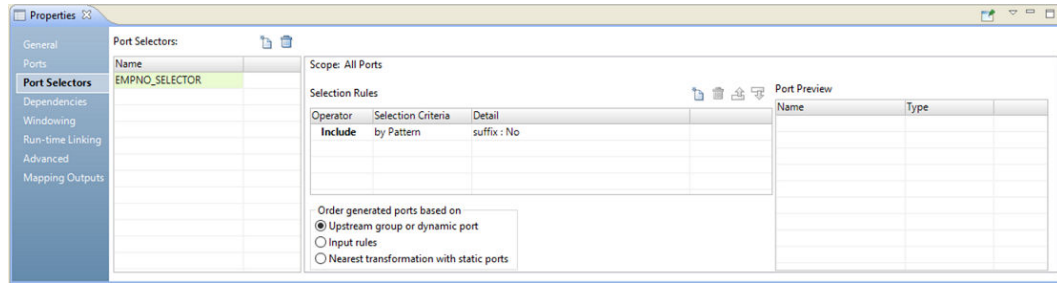
## 포트 선택기 구성

포트 선택기를 구성하는 경우 선택 규칙을 정의하여 포함할 생성된 포트를 결정합니다. 선택 규칙은 동적 포트에 대해 구성할 수 있는 입력 규칙과 유사합니다.

포트 선택기는 정적 포트 또는 생성된 포트를 포함할 수 있습니다. 포트 선택기 탭에서 포트 선택기를 구성합니다.



다음 이미지는 **포트 선택기** 탭을 보여 줍니다.



포트 선택기에 대해 다음 속성을 구성합니다.

### 이름

포트 선택기를 식별합니다. 변환에서 여러 포트 선택기를 작성하고 식에서 참조할 수 있습니다.

### 범위

포트 선택기가 적용되는 포트 그룹을 식별합니다. 조이너 또는 조희 변환에 대한 포트 선택기를 작성하는 경우 범위를 선택해야 합니다. 이러한 변환에는 여러 입력 그룹이 있습니다. 조이너 변환에는 마스터 또는 세부 정보 범위가 있습니다. 조희 변환에는 가져오기 또는 조희 범위가 있습니다. 식 변환에는 입력 그룹 하나가 있습니다. 범위는 항상 모든 포트입니다.

### 선택 규칙

포트 선택기에 포함될 포트를 결정합니다. 선택 규칙을 작성하면 **포트 미리보기** 패널에 현재 입력 포트에 적합한 포트가 표시됩니다. 이러한 포트는 변경될 수 있습니다. 선택 규칙을 구성하여 여러 소스에서 포트를 포함합니다.

## 선택 규칙

포트 선택기와 연결된 선택 규칙은 포트 선택기에 포함될 포트를 결정합니다.

선택 규칙을 작성하면 **포트 미리보기** 패널에 현재 입력 포트에 적합한 포트가 표시됩니다. 이러한 포트는 변경될 수 있습니다. 선택 규칙을 구성하여 여러 소스에서 포트를 포함합니다.

다음 조건에 따라 선택 규칙을 작성합니다.

### 연산자

선택 규칙이 반환하는 포트를 포함하거나 제외합니다. 기본값은 포함입니다. 포트를 제외하려면 포트를 포함해야 합니다.

### 선택 조건

작성할 선택 규칙 유형입니다. 열 이름, 포트 유형, 패턴 또는 복합 데이터 유형 정의를 기반으로 규칙을 생성할 수 있습니다. 열 이름에 따라 포트를 포함하려면 특정 이름을 검색하거나 이름의 문자 패턴을 검색합니다.

### 세부 정보

선택 조건에 적용할 값입니다. 선택 조건이 열 이름 기준인 경우 검색할 문자열 또는 이름을 구성합니다. 선택 조건이 포트 유형인 경우 포함할 포트 유형을 선택합니다.

다음 표에는 선택 조건과 해당 조건에 대한 세부 정보를 지정하는 방법이 설명되어 있습니다.

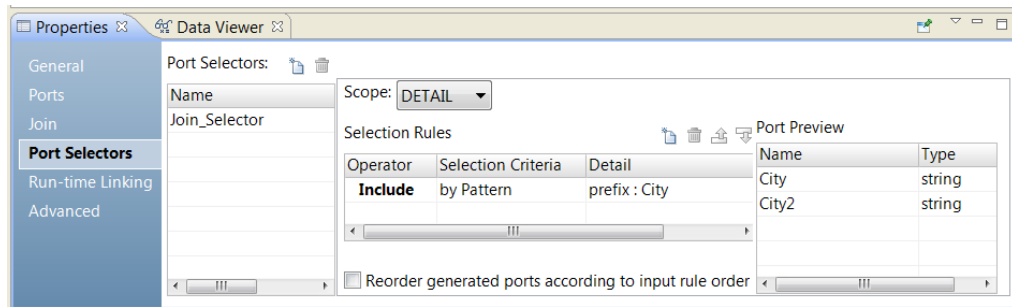
선택 조건	설명	세부 정보
모두	모든 포트를 포함합니다.	세부 정보가 필요하지 않습니다.
이름	포트 이름에 따라 포트를 필터링합니다.	값 목록에서 포트 이름을 선택하거나 포트 또는 포트 목록 유형인 매개 변수를 사용합니다.
유형	각 포트의 데이터 유형에 따라 포트를 필터링합니다.	목록에서 데이터 유형을 선택합니다.
패턴	이름의 문자 문자열 또는 정규식으로 포트를 필터링합니다.	접두사, 접미사 또는 정규식을 포트 이름에 대한 패턴 유형으로 선택합니다. 그런 다음 패턴에 대한 값을 입력하거나 문자열 유형인 매개 변수를 사용합니다.
복합 데이터 유형 정의	복합 데이터 유형 정의를 기준으로 포트를 필터링합니다.	접두사, 접미사 또는 정규식을 복합 데이터 유형 정의에 대한 패턴 유형으로 선택합니다. 그런 다음 패턴에 대한 값을 입력하거나 문자열 유형인 매개 변수를 사용합니다.

## 포트 선택기 작성

포트 선택기를 작성하여 동적 식, 조회 조건 또는 조이너 조건에서 사용할 포트를 결정합니다.

1. **포트 선택기** 탭을 클릭합니다.
2. **포트 선택기** 영역에서 **새로 만들기**를 클릭합니다.  
Developer tool이 모든 포트를 포함하는 기본 선택 규칙으로 포트 선택기를 작성합니다.
3. **포트 선택기** 영역에서 포트 선택기 이름을 고유 이름으로 변경합니다.
4. 조이너 변환 또는 조회 변환에 대해 작업하는 경우 범위를 선택합니다.  
사용 가능한 포트는 선택하는 포트 그룹에 따라 변경됩니다.
5. **선택 규칙** 영역에서 **연산자**를 선택합니다.
  - 포함. 포트 선택기에 대한 포트를 포함하는 규칙을 작성합니다. 포트를 제외하려면 포트를 포함해야 합니다.
  - 제외. 포트 선택기에서 특정 포트를 제외하는 규칙을 작성합니다.
6. **선택 조건**을 선택합니다.
  - 이름별. 이름으로 특정 포트를 선택합니다. 범위의 포트 목록에서 포트 이름을 선택할 수 있습니다.
  - 유형별. 유형으로 포트를 선택합니다. 데이터 유형을 하나 이상 선택할 수 있습니다.
  - 패턴별. 포트 이름의 글자 패턴으로 포트를 선택합니다. 특정 글자로 검색하거나 정규식을 작성할 수 있습니다.

다음 이미지는 포트 선택기 탭을 보여 줍니다.



7. 세부 정보 열을 클릭합니다.  
입력 규칙 세부 정보 대화 상자가 나타납니다.
8. 포트 기준으로 필터링할 값을 선택합니다.
  - 이름별. 선택하여 값 또는 매개 변수 기준으로 포트 목록을 작성합니다. 선택을 클릭하여 목록에서 포트를 선택합니다.
  - 유형별. 목록에서 데이터 유형을 하나 이상 선택합니다. **포트 미리보기** 영역에 선택하는 유형의 포트가 표시됩니다.
  - 패턴별. 선택하여 포트 이름의 접두사 또는 접미사에서 문자의 특정 패턴을 검색합니다. 또는 선택하여 검색할 정규식을 작성합니다. 매개 변수를 구성하거나 검색할 패턴을 구성합니다.

**포트 미리보기** 영역에는 규칙을 구성한 포트 선택기의 포트가 표시됩니다..
9. 포트 선택기에서 포트 순서를 다시 지정하려면 **생성된 포트의 순서를 입력 규칙 순서에 따라 다시 지정**을 선택합니다.

## 창 작업

변환에 창 함수가 포함되는 경우 창 작업 속성을 구성해야 합니다. 창 작업은 Spark 엔진의 변환에만 사용할 수 있습니다.

창 함수는 행 그룹에 작동하며 모든 입력 행에 대한 반환 값을 계산합니다.

식 변환에서 창 함수를 정의하기 전에 창 작업 속성을 구성하여 창을 설명해야 합니다. 창 작업 속성에는 프레임 사양, 파티션 키 및 순서 키가 포함됩니다. 프레임 사양은 현재 행에 대한 전체 계산에 포함되는 행을 설명합니다. 파티션 키는 동일한 파티션에 포함할 행을 결정합니다. 순서 키는 파티션에서 행을 정렬하는 방법을 결정합니다.

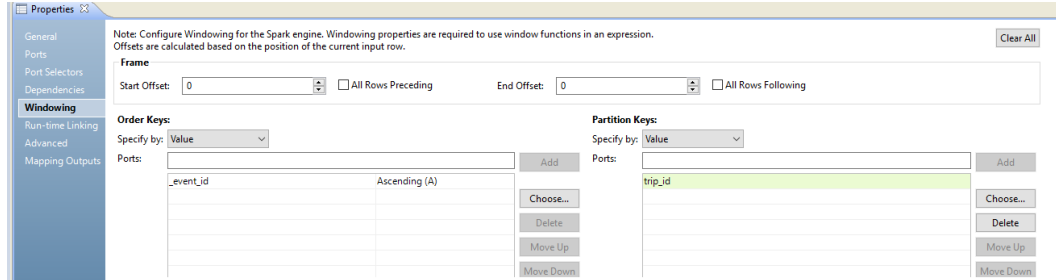
창 작업 속성을 구성한 후 식 변환에서 창 함수를 정의합니다. Informatica는 창 함수 LEAD 및 LAG를 지원합니다. 또한 집계 함수를 식 변환에서 창 함수로 사용할 수도 있습니다.

## 창 작업 구성

식 변환에 창 함수를 포함하는 경우 함수에 연결된 창 작업 속성을 구성합니다. 창 작업 속성은 특정 입력 행에 연결된 분할, 정렬 및 프레임 경계를 정의합니다.

창 작업 탭에서 창 작업에 대한 변환을 구성합니다.

다음 이미지는 창 작업 탭을 보여 줍니다.



창 작업 탭에서 다음 속성 그룹을 구성할 수 있습니다.

### 프레임

현재 입력 행의 위치로부터의 실제 오프셋에 따라 현재 입력 행에 대한 프레임에 포함되는 행을 정의합니다.

집계 함수를 창 함수로 사용하는 경우 프레임을 구성합니다. 창 함수 LEAD 및 LAG는 개별 행을 참조하며 프레임 사양을 무시합니다.

### 파티션 키

입력 행을 다른 파티션으로 분할합니다. 파티션 키를 정의하지 않으면 모든 행이 단일 파티션에 속합니다.

### 순서 키

파티션의 행을 정렬하는 방법을 정의합니다. 선택한 포트에 따라 파티션 내의 행 위치가 결정됩니다. 순서 키는 오름차순 또는 내림차순일 수 있습니다. 순서 키를 정의하지 않으면 행에 특정 순서가 적용되지 않습니다.

## 프레임

프레임은 현재 행에 대한 상대 위치를 기준으로 현재 입력 행에 대한 계산에 포함할 행을 결정합니다.

LEAD 또는 LAG 대신 집계 함수를 사용하는 경우 창 프레임을 지정해야 합니다. LEAD 및 LAG는 개별 행을 참조하며 프레임 사양을 무시합니다.

시작 오프셋과 종료 오프셋은 현재 입력 행의 앞과 뒤에 표시되는 행 수를 설명합니다. 오프셋 "0"은 현재 입력 행을 나타냅니다. 예를 들어 시작 오프셋이 -3이고 종료 오프셋이 0인 경우 프레임에 현재 입력 행이 포함되고 현재 입력 행의 앞에 3개의 행이 있음을 나타냅니다.

다음 이미지는 시작 오프셋이 -1이고 종료 오프셋이 1인 프레임을 보여 줍니다.

Type	Category	Revenue	
Action	Video game	1000	
Arcade	Video game	1000	← 1 PRECEDING
Sports	Video game	2000	
Adventure	Video game	3000	← 1 FOLLOWING
Strategy	Video game	4000	

Current input row →

모든 입력 행에 대해 함수는 프레임 내의 행에 집계 작업을 수행합니다. 이전 프레임을 사용하여 SUM 같은 집계 식을 구성하는 경우 이 식은 프레임 안에 있는 값의 합계를 계산하고 입력 행에 대해 값 6000을 반환합니다.

현재 입력 행을 포함하지 않는 프레임을 지정할 수도 있습니다. 예를 들어 시작 오프셋이 10이고 종료 오프셋이 15인 경우 전체 행이 6개이고 현재 행 뒤에 열 번째 행부터 열 다섯 번째 행이 포함되는 프레임을 나타냅니다.

**참고:** 시작 오프셋은 종료 오프셋보다 작거나 같아야 합니다.

**이전의 모든 행 및 이후의 모든 행** 오프셋은 파티션의 첫 번째 행과 파티션의 마지막 행을 나타냅니다. 예를 들어 시작 오프셋이 이전의 모든 행이고 종료 오프셋이 -1인 경우 프레임에는 현재 행의 앞에 행 1개가 포함되고 그 뒤에 모든 행이 포함됩니다.

다음 그림은 시작 오프셋이 0이고 종료 오프셋이 이후의 모든 행이 프레임임을 보여 줍니다.

Genre	Recordings	Revenue
Jazz	233	5000
Gospel	214	1000
Country	145	2000
Ethnic	154	9000
Pop	317	4000
Rock	237	2100
Classical	221	3200
EDM	153	950
Hip Hop	839	2300
Punk	415	7650

Current input row →

All Rows Following

## 파티션 및 순서 키

행 그룹을 형성하도록 파티션 및 순서 키를 구성하고 각 파티션 안에 있는 행의 순서 또는 시퀀스를 정의합니다.

다음 키를 사용하여 창에서 행을 그룹화하고 순서를 지정하는 방법을 지정합니다.

### 파티션 키

모든 입력에 대한 계산을 수행하지 않고 파티션 경계를 정의하도록 파티션 키를 구성합니다. 창 함수는 현재 행과 동일한 파티션에 속하는 행 전체에 작동합니다.

값 또는 매개 변수로 파티션 키를 지정할 수 있습니다. 포트 이름을 사용하려면 **값**을 선택합니다. 정렬 키 목록 매개 변수를 사용하려면 **매개 변수**를 선택합니다. 정렬 키 목록 매개 변수에는 정렬할 포트 목록이 포함됩니다. 파티션 키를 지정하지 않으면 모든 데이터가 동일한 파티션에 포함됩니다.

### 순서 키

순서 키를 사용하여 파티션의 행을 정렬하는 방법을 결정할 수 있습니다. 순서 키는 파티션 안에서 특정 행의 위치를 정의합니다.

값 또는 매개 변수로 순서 키를 지정할 수 있습니다. 포트 이름을 사용하려면 **값**을 선택합니다. 정렬 키 목록 매개 변수를 사용하려면 **매개 변수**를 선택합니다. 정렬 키 목록 매개 변수에는 정렬할 포트 목록이 포함됩니다. 또한 데이터를 오름차순 또는 내림차순으로 정렬할지 선택해야 합니다. 순서 키를 지정하지 않으면 파티션의 모든 행이 특정 순서로 정렬되지 않습니다.

### 예

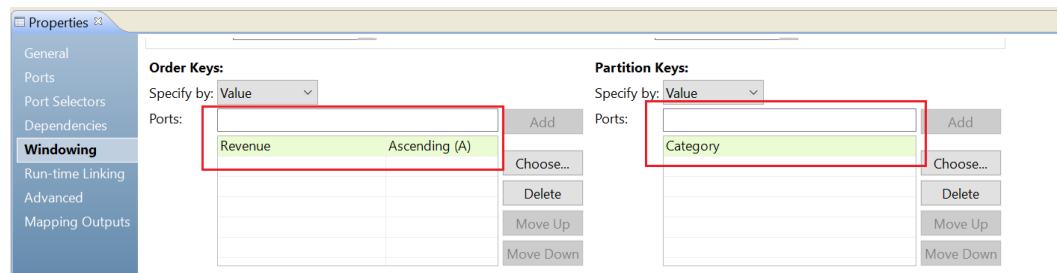
커피와 차를 파는 매장의 소유자가 있습니다. 이 소유자는 가장 많이 팔린 커피 및 차 제품과 두 번째로 많이 팔린 커피 및 차 제품을 계산하려고 합니다.

다음 테이블에는 제품, 해당하는 제품 범주 및 각 제품의 매출이 나열되어 있습니다.

Product	Category	Revenue
Espresso	Coffee	600
Black	Tea	550
Cappuccino	Coffee	500
Americano	Coffee	600
Oolong	Tea	250
Macchiato	Coffee	300
Green	Tea	450
White	Tea	650

소유자는 데이터를 범주로 분할하고 매출의 내림차순으로 데이터를 정렬합니다.

다음 이미지는 창 작업 탭에서 구성하는 속성을 보여 줍니다.



다음 테이블에는 범주에 따라 2개의 파티션으로 그룹화된 데이터가 나와 있습니다. 각 파티션 내에서 매출은 내림차순으로 정렬되어 있습니다.

Product	Category	Revenue
Espresso	Coffee	600
Americano	Coffee	600
Cappuccino	Coffee	500
Macchiato	Coffee	300
White	Tea	650
Black	Tea	550
Green	Tea	450
Oolong	Tea	250

소유자는 분할 및 정렬 사양을 바탕으로 가장 많이 팔린 커피 2개가 에스프레소와 아메리카노이고 가장 많이 팔린 차 2개가 백차와 홍차라는 것을 알게 됩니다.

## 창 작업 구성에 대한 규칙 및 지침

변환에서 창 작업을 구성할 때는 특정 지침이 적용됩니다.

창 함수에 대한 창 작업 속성을 구성하는 경우 다음 규칙 및 지침을 고려하십시오.

- 프레임을 구성하는 경우 시작 오프셋은 종료 오프셋보다 작거나 같아야 합니다. 그렇지 않으면 프레임이 올바르게 표시되지 않습니다.
- 집계 함수를 창 함수로 사용하는 경우 프레임 사양을 구성합니다. LEAD 및 LAG는 오프셋 값을 기준으로 작동하며 프레임 사양을 무시합니다.
- 복합 포트를 파티션 또는 순서 키로 사용할 수 없습니다.
- 런타임 오류를 방지하려면 파티션 및 순서 키에 고유한 포트 이름을 할당합니다.
- 파티션 및 순서 키에는 동적 포트와 동일한 동적 포트에서 생성된 하나 이상의 포트를 동시에 사용할 수 없습니다. 동적 포트를 선택하거나 생성된 포트를 선택해야 합니다.

## 동적 식

동적 출력 포트에서 식을 구성하면 해당 식이 동적 식이 됩니다. 동적 식은 여러 출력 포트를 생성할 수 있습니다.

동적 식의 포트 선택기 또는 동적 포트를 참조할 수 있습니다. 포트 선택기 또는 동적 포트가 여러 포트를 포함하는 경우 동적 식이 각 포트에 대해 실행됩니다.

동적 식을 구성하면 **Developer tool**이 생성된 포트가 식에 대해 올바른 유형인지 유효성을 검사하지 않습니다. 예를 들어 문자열 유형이 필요한 식에 10진수 유형 포트가 포함된 포트 선택기를 참조하는 경우 식이 디자인 타임 시 올바른 것으로 나타납니다.

### 예제

식 변환에는 다음 생성된 입력 포트가 있습니다.

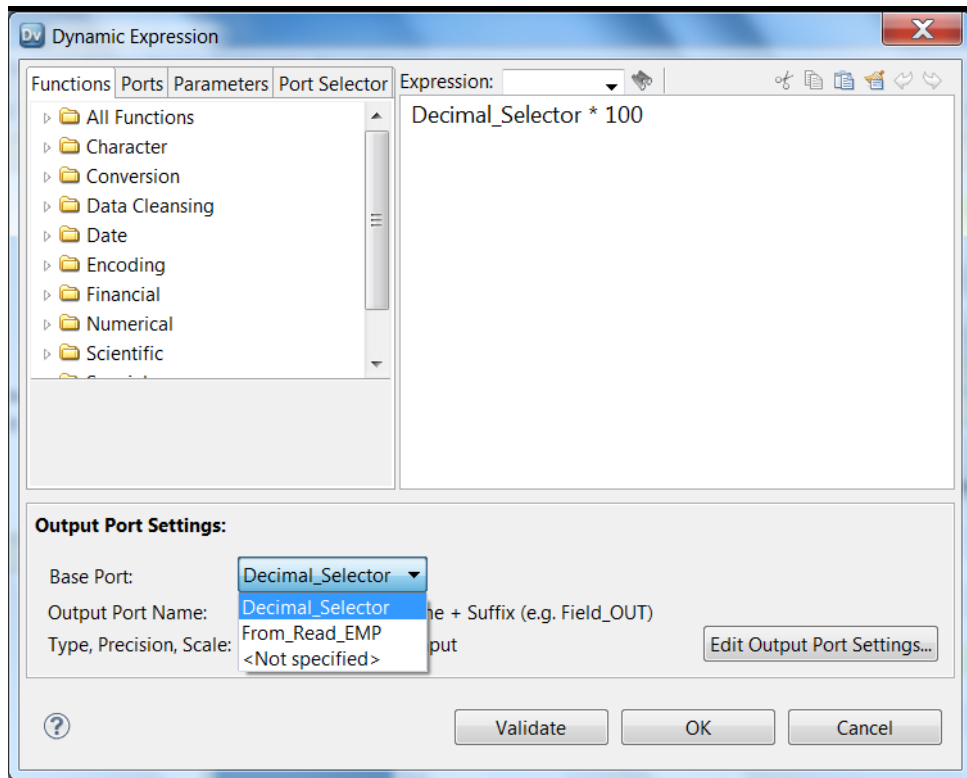
```
EMPNO  Decimal
NAME   String
SALARY Decimal
DEPTNO Decimal
```

변환은 **MyDynamicPort**라는 동적 출력 포트를 포함합니다. 해당 출력 포트는 동적 식의 결과를 반환합니다. 동적 식은 포트 선택기의 각 포트 값을 100으로 곱합니다. 해당 식은 포트 선택기의 각 포트에 대해 한 번씩 실행됩니다. 각 인스턴스는 서로 다른 결과를 반환할 수 있습니다. 식 변환은 각 결과에 대해 개별 출력 포트를 생성합니다.

**Decimal\_Selector** 포트 선택기에는 10진수 데이터 유형인 포트를 포함하는 선택 규칙이 있습니다.

```
EMPNO  Decimal
SALARY Decimal
DEPTNO Decimal
```

다음 이미지는 **Decimal\_Selector** 포트 선택기를 참조하는 동적 식을 보여 줍니다.



출력 포트 설정을 편집하여 출력 포트 이름 및 출력 포트 속성을 변경합니다. 기본 포트를 선택할 수도 있습니다.

## 출력 포트 설정

동적 식에 대해 입력으로 사용하려는 포트를 나타낼 수 있습니다. **기본 포트** 영역에서 포트를 선택합니다.

기본 포트로 **Decimal\_Selector** 포트 선택기를 선택하는 경우 동적 식이 10진수 유형 포트를 반환합니다. **NAME** 포트는 문자열이므로 동적 식이 포트를 생성하지 않습니다.

다음 이미지는 변환의 생성된 포트를 보여 줍니다.

Properties Data Viewer Tags Notifications								
General								
<b>Ports</b>								
Port Selectors								
Dependencies								
Parameters								
Run-time Linking								
Advanced								
Mapping Outputs								
Name	Type	Precis...	Scale	Input	Output	Varia...	Expression	
1 From_Read_EMP	dynamic		0	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	From_Read_EMP	
1 EMPNO	decimal	3	0	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		
2 NAME	string	10	0	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		
3 SALARY	decimal	4	0	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		
4 DEPTNO	decimal	4	0	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		
2 MyDynamicPort	dynamic		0	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	Decimal_Selector * 100	
1 EMPNO_OUT	decimal	3	0	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		
2 SALARY_OUT	decimal	4	0	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		
3 DEPTNO_OUT	decimal	4	0	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		

**From\_Read\_Emp** 동적 포트가 입력-출력 포트이지만 변환은 **MyDynamicPort** 동적 출력 포트의 포트만 반환합니다.

출력 포트의 이름 지정 방법을 구성할 수 있습니다. 기본 출력 포트 이름은 입력 포트 이름이고 접미사는 **\_OUT**입니다.

포트 선택기의 기본 포트를 변경할 수 있습니다.



다음 이미지는 식 편집기에서 출력 포트 설정을 보여 줍니다.

**Output Port Settings:**

Base Port: From\_Read\_EMP ▾

Output Port Name: Primary input port name + Suffix (e.g. Field\_O

Type, Precision, Scale: Inherit from primary input
Edit Output Port Settings...

기본 포트를 **From\_Read\_EMP**로 구성하는 경우 생성된 모든 입력 포트를 포함하는 동적 포트를 선택합니다. 데이터 통합 서비스가 **From\_Read\_EMP**의 모든 포트에 대해 동적 식을 실행합니다.

다음 이미지는 **From\_Read\_Emp** 입력에 따른 생성된 출력 포트를 보여 줍니다.

General								
Ports								
	Name	Type	Precis...	Scale	Input	Output	Varia...	Expression
1	From_Read_EMP	dynamic		0	✓	✓		From_Read_EMP
1	EMPNO	decimal	3	0				
2	NAME	string	10	0				
3	SALARY	decimal	4	0				
4	DEPTNO	decimal	4	0				
2	MyDynamicPort	dynamic		0		✓		Decimal_Selector * 100
1	EMPNO_OUT	decimal	3	0				
2	NAME_OUT	string	10	0				
3	SALARY_OUT	decimal	4	0				
4	DEPTNO_OUT	decimal	4	0				

생성된 출력 포트에는 문자열 유형인 **NAME\_OUT**이라는 출력 포트가 포함됩니다.

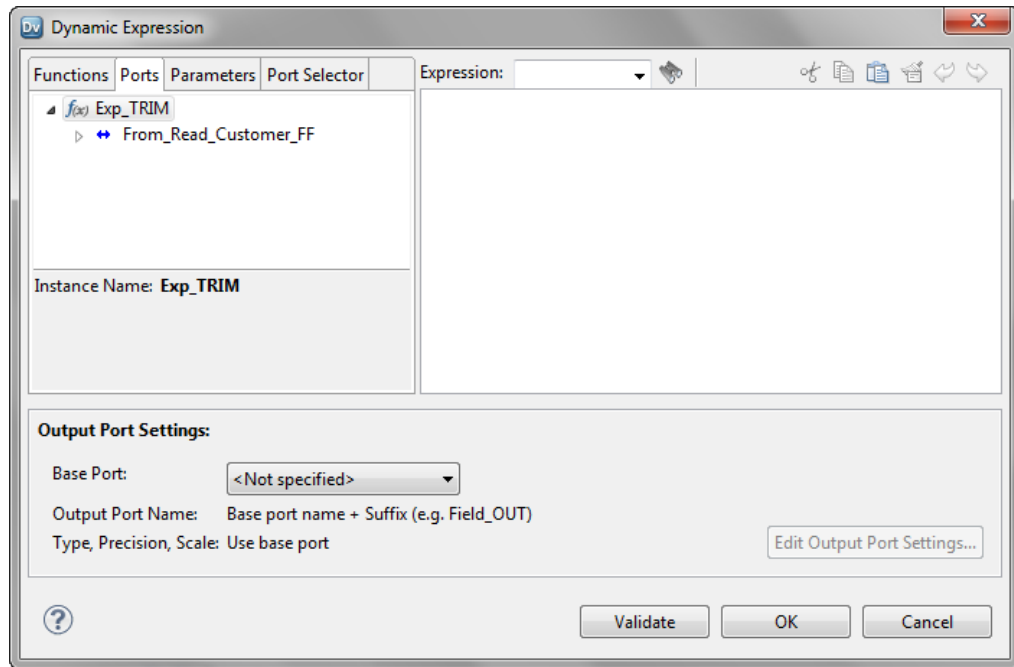
데이터 통합 서비스가 각 동적 식에 대해 출력 포트를 생성합니다. 포트 15개를 생성하는 동적 식을 작성하고 포트 5개를 생성하는 또 다른 동적 식을 정의하는 경우 데이터 통합 서비스가 출력 포트 20개를 생성합니다. 각 동적 출력 포트는 다른 포트 그룹을 생성합니다.

## 동적 식 작성

식 변환에서 동적 식을 작성하여 동적 포트 또는 포트 선택기의 각 포트에 대해 식을 한 번 실행합니다. 동적 식은 각 인스턴스에 대한 개별 생성된 포트에 결과를 반환합니다.

- 식 변환에서 **속성** 보기로 이동하여 **포트** 탭을 클릭합니다.
- 새 동적 포트**를 클릭합니다.  
Developer tool이 기본 속성으로 동적 포트를 작성합니다.
- 동적 포트의 이름을 바꾸고 입력 옵션을 비활성화합니다.  
동적 포트는 출력 포트여야 합니다.
- 동적 출력 포트에 대한 **식** 열에서 **열기** 단추(🔑)를 클릭합니다.

동적 식 대화 상자가 나타납니다.

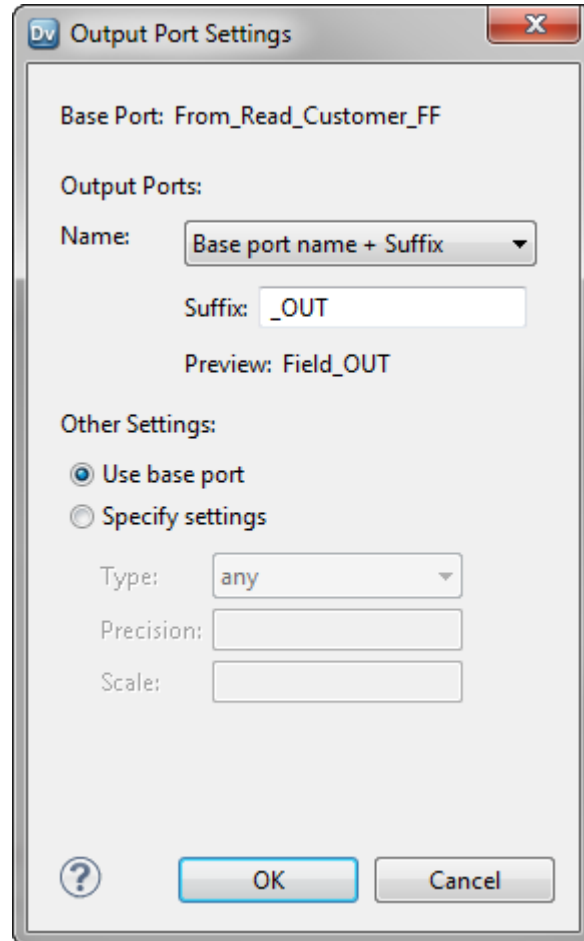


5. 식 편집기에서 식을 입력합니다. 식에는 포트 선택기 또는 동적 포트가 포함될 수 있습니다.  
예를 들어 Dynamic\_Customer가 동적 포트인 LTRIM(RTRIM(Dynamic\_Customer))입니다.
6. **유효성 검사**를 클릭하여 식의 유효성을 검사합니다.
7. **확인**을 클릭하여 **식 유효성 검사** 대화 상자를 종료합니다.
8. **출력 포트 설정** 영역에서 동적 출력 포트를 **기본 포트** 목록에서 선택하거나 식에서 참조한 포트 선택기를 선택합니다.  
Developer tool이 선택한 항목에 따라 출력 포트를 생성합니다.

9. 다음 단계를 사용하여 출력 포트의 이름을 바꿉니다.

a. **출력 포트 설정 편집**을 클릭합니다.

출력 포트 설정 대화 상자가 나타납니다.



b. 이름 목록에서 옵션 중 하나를 선택하고 접두사 또는 접미사에 대한 값을 입력합니다. **고정 문자열 + 자동 번호**를 선택한 경우 출력 포트 이름에 대한 텍스트를 입력합니다. 예를 들어 출력 포트 이름에 TRIM을 입력하는 경우 출력 포트 이름은 TRIM1, TRIM2, TRIM3으로 나타납니다.

c. 필요한 경우 **기타 설정** 영역에서 **설정 지정**을 선택하여 출력 포트에 대한 유형, 전체 자릿수 및 소수 자릿수를 변경합니다. 기본적으로 출력 포트는 기본 포트의 설정을 사용합니다.

d. **확인**을 클릭합니다.

10. **확인**을 클릭하여 **동적 식 편집기**를 종료합니다.

## 식 변환 고급 속성

데이터 통합 서비스에서 식 변환에 대한 데이터를 처리하는 방법을 결정할 수 있게 도와주는 속성을 구성하십시오.

식 변환에 대한 다음 고급 속성을 구성합니다.

## 추적 수준

이 변환에 대해 로그에 표시되는 세부 정보의 양입니다. 간단, 보통, 자세한 정보 표시 초기화 또는 자세한 정보 표시 데이터를 선택할 수 있습니다. 기본값은 보통입니다.

## 행 순서 유지

변환에 대한 입력 데이터의 행 순서를 유지합니다. 데이터 통합 서비스에서 행 순서를 변경할 수 있는 어떤 최적화도 수행하지 않는 경우 이 옵션을 선택합니다.

데이터 통합 서비스가 최적화를 수행하는 경우 이전에 매핑에 설정된 순서를 잃게 될 수 있습니다. 정렬된 플랫 파일 소스, 정렬된 관계형 소스 또는 분류기 변환을 사용하여 매핑에서 순서를 설정할 수 있습니다. 행 순서를 유지하기 위해 변환을 구성하는 경우 데이터 통합 서비스는 매핑에 대해 최적화를 수행할 때 이 구성을 고려합니다. 데이터 통합 서비스는 순서를 유지할 수 있을 때 변환에 대해 최적화를 수행합니다. 데이터 통합 서비스는 최적화가 행 순서를 변경할 수 있는 경우 변환에 대해 최적화를 수행하지 않습니다.

# 식 변환 - 비원시 환경

비원시 환경에서 식 변환의 처리는 변환을 실행하는 엔진에 따라 다릅니다.

다음과 같은 비원시 런타임 엔진에 대한 지원을 고려하십시오.

- Blaze 엔진. 제한적으로 지원됩니다.
- Spark 엔진. 제한적으로 지원됩니다.
- Databricks Spark 엔진. 제한적으로 지원됩니다.

## 식 변환 - Blaze 엔진

다음과 같은 상황에서는 매핑 유효성 검사가 실패합니다.

- 변환에 상태 저장 변수 포트가 포함되어 있습니다.
- 변환의 식에 지원되지 않는 함수가 포함되어 있습니다.

사용자 정의 함수가 포함된 식 변환은 함수에 예외 오류가 있는 행에 대해 null 값을 반환합니다.

## 식 변환 - Spark 엔진

다음과 같은 상황에서는 매핑 유효성 검사가 실패합니다.

- 변환에 상태 저장 변수 포트가 포함되어 있습니다.
- 변환의 식에 지원되지 않는 함수가 포함되어 있습니다.

**참고:** 식이 숫자 오류(예: 0으로 나누기 또는 음수의 SQRT)를 야기하는 경우 null 값이 반환되고 출력에 행이 표시되지 않습니다. 원시 환경에서는 식에 무한값 또는 NaN 값이 반환됩니다.

## 식 변환 - Databricks Spark 엔진

다음과 같은 상황에서는 매핑 유효성 검사가 실패합니다.

- 변환에 상태 저장 변수 포트가 포함되어 있습니다.
- 변환의 식에 지원되지 않는 함수가 포함되어 있습니다.

**참고:** 식이 숫자 오류(예: 0으로 나누기 또는 음수의 SQRT)를 야기하는 경우 null 값이 반환되고 출력에 행이 표시되지 않습니다. 원시 환경에서는 식에 무한값 또는 NaN 값이 반환됩니다.

## 제 17 장

# 필터 변환

이 장에 포함된 항목:

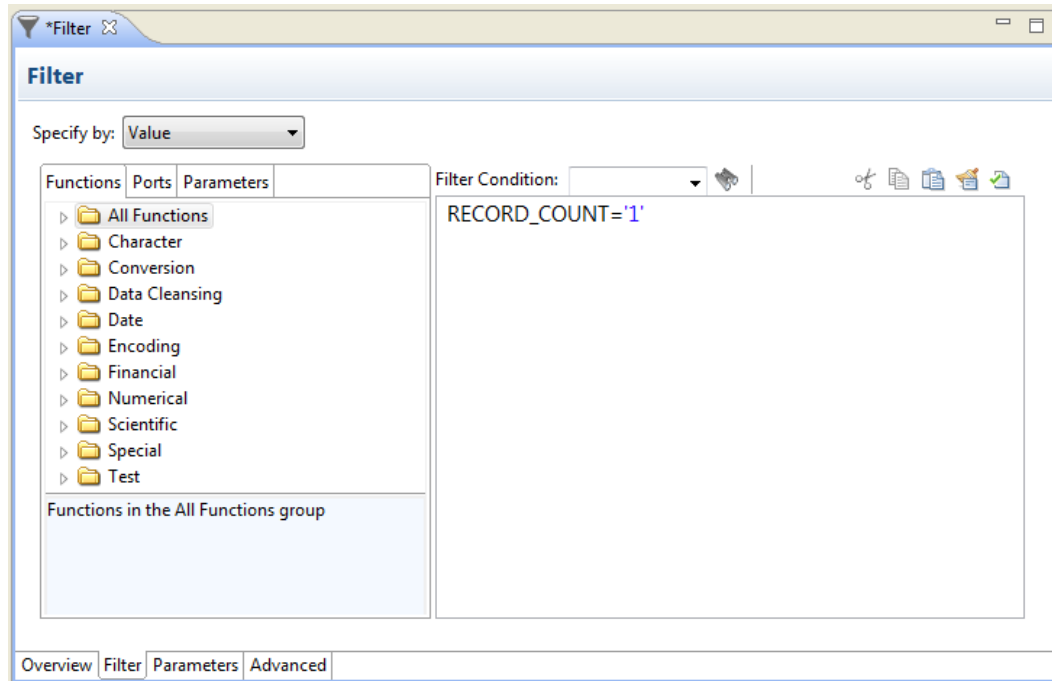
- [필터 변환 개요, 274](#)
- [동적 매핑의 필터 변환, 275](#)
- [필터 조건, 275](#)
- [필터 변환 고급 속성, 277](#)
- [필터 변환 성능 팁, 278](#)
- [필터 변환 - 비원시 환경, 278](#)

## 필터 변환 개요

매핑의 행을 필터링하려면 필터 변환을 사용하십시오. 활성 변환인 필터 변환은 해당 변환을 통과하는 행의 수를 변경할 수 있습니다.

필터 변환은 지정된 필터 조건을 충족하는 행의 통과를 허용합니다. 조건을 충족하지 못하는 행은 삭제됩니다. 하나 이상의 조건을 기반으로 데이터를 필터링할 수 있습니다.

다음 이미지는 필터 변환의 필터 조건을 보여 줍니다.



행이 지정된 조건을 충족하는지 여부에 따라 데이터 통합 서비스가 평가하는 행마다 필터 조건이 **TRUE** 또는 **FALSE**를 반환합니다. **TRUE**를 반환하는 행의 경우 데이터 통합 서비스가 변환을 통과합니다. **FALSE**를 반환하는 행의 경우 데이터 통합 서비스가 삭제하고 메시지를 로그에 기록합니다.

둘 이상 변환의 포트를 필터 변환에 연결할 수 없습니다. 필터의 입력 포트는 단일 변환에서 제공되어야 합니다.

## 동적 매핑의 필터 변환

동적 매핑에서 필터 변환을 사용할 수 있습니다. 변환에서 동적 포트를 구성하고 필터 조건의 생성된 포트를 참조할 수 있습니다.

전체 필터 조건을 매개 변수화할 수 있습니다. 전체 식을 포함하는 기본값으로 식 매개 변수를 구성합니다.

**Developer tool**은 매개 변수 기본값에서 필터 조건의 유효성을 검사하지 않습니다.

필터 조건의 동적 포트를 참조할 수 있습니다. 동적 포트는 여러 생성된 포트를 포함할 수 있습니다. 데이터 통합 서비스가 각 생성된 포트를 포함하도록 필터 조건을 확장합니다. 각 생성된 포트는 식에 포함할 올바른 유형이어야 합니다.

필터 조건의 생성된 포트를 참조할 수 있습니다. 그러나 런타임 시 생성된 포트가 없는 경우 매핑이 실패합니다.

## 필터 조건

필터 조건은 **TRUE** 또는 **FALSE**를 반환하는 식입니다.

식 편집기에 조건을 입력합니다. 필터 조건은 대/소문자를 구분합니다.

단일 값을 필터로 반환하는 모든 식을 사용할 수 있습니다. 예를 들어 급여가 \$30,000 이하인 직원의 행을 필터링하려면 다음 조건을 입력하십시오.

```
SALARY > 30000
```

AND 및 OR 논리 연산자를 사용하여 조건의 여러 구성 요소를 지정할 수 있습니다. 월급이 \$30,000 미만인 직원과 \$100,000보다 많은 직원을 필터링하려면 다음 조건을 입력하십시오.

```
SALARY > 30000 AND SALARY < 100000
```

필터 조건에서 포트, 매개 변수, 동적 포트 및 생성된 포트를 사용할 수 있습니다. 식 편집기에서 포트 및 매개 변수를 선택합니다.

필터 조건에서 동적 포트를 사용하는 경우 필터 조건이 확장되어 동적 포트에서 생성된 모든 포트가 포함됩니다. 예를 들어 동적 포트 **MyDynamicPort**에는 다음 10진수 포트 3개가 포함됩니다.

```
Salary  
Bonus  
Stock
```

다음 필터 조건을 구성하는 경우

```
MyDynamicPort > 100
```

필터 조건이 다음 식으로 확장됩니다.

```
Salary > 100 AND Bonus > 100 AND Stock > 100
```

필터 조건에 대한 상수를 입력할 수 있습니다. FALSE에 해당하는 숫자는 영(0)입니다. 영(0)이 아닌 값은 TRUE에 해당합니다. 예를 들어 숫자 데이터 유형을 가진 **NUMBER\_OF\_UNITS**라는 포트가 변환에 포함되어 있습니다. **NUMBER\_OF\_UNITS** 값이 영(0)일 경우 FALSE를 반환하도록 필터 조건을 구성하십시오. 그렇지 않을 경우 조건에서 TRUE를 반환합니다.

**참고:** 단일 포트 선택기 또는 동적 포트를 부울 값으로 사용할 수 없습니다.

식에서 TRUE 또는 FALSE를 값으로 지정할 필요는 없습니다. TRUE 및 FALSE는 설정된 조건의 암시적인 반환 값입니다. 필터 조건이 NULL로 평가될 경우 행은 FALSE입니다.

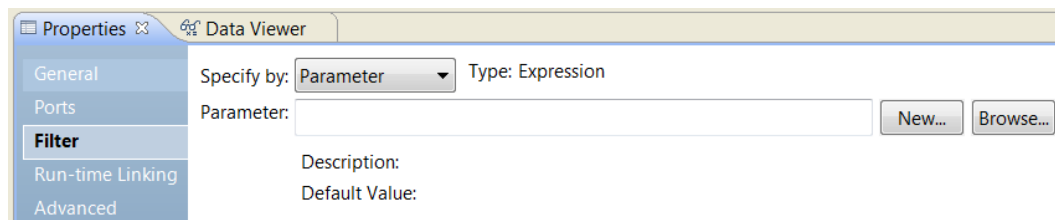
## 필터 조건 매개 변수화

식 매개 변수를 구성하여 필터 조건을 정의할 수 있습니다. 식 매개 변수는 전체 식을 포함합니다.

필터 변환이 동적 매핑에 있는 경우 필터 조건을 매개 변수화해야 할 수 있습니다. 필터 조건은 런타임 시 변환의 생성된 포트에 따라 변경될 수 있습니다.

필터 조건에 대해 식 매개 변수를 사용하려면 필터 변환 속성의 **필터** 탭에서 **매개 변수로 지정**을 선택합니다.

다음 이미지는 매개 변수로 필터 조건을 지정하는 경우 **필터** 탭을 보여 줍니다.

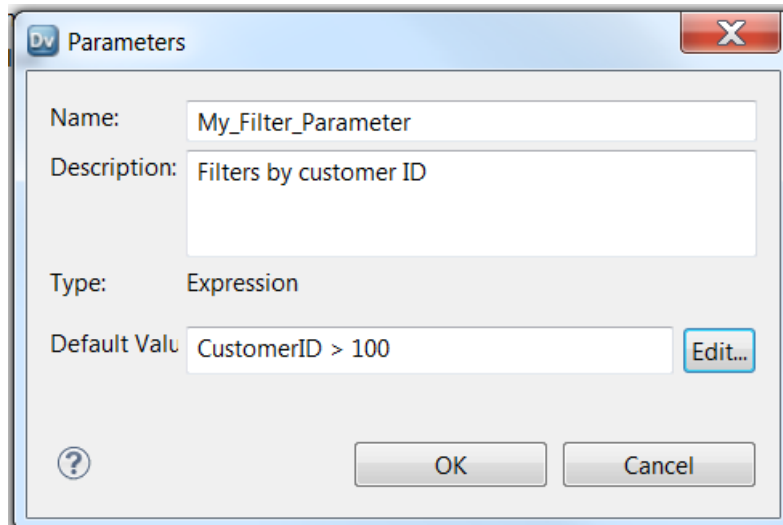


이미 작성한 식 매개 변수를 찾아 선택할 수 있습니다. 또는 식 매개 변수를 작성할 수 있습니다.

식 매개 변수를 작성하려면 **새로 만들기**를 클릭합니다. 매개 변수 이름, 설명 및 기본 식 값을 입력합니다.

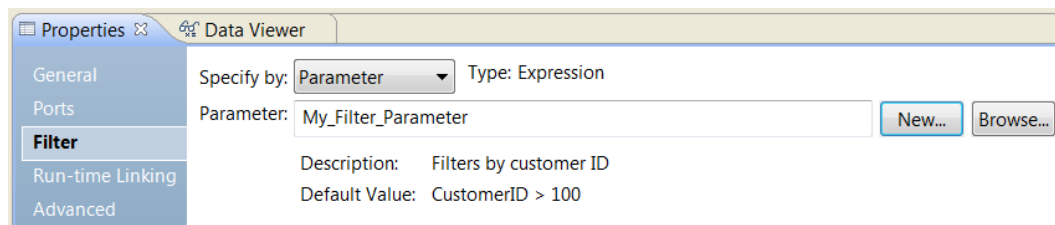


다음 이미지는 매개 변수를 입력하는 위치를 보여 줍니다.



매개 변수 대화 상자에서 기본 식을 입력할 수 있습니다. 식 편집기를 사용하려는 경우 편집을 클릭합니다. 식 편집기를 사용하는 경우 식에서 사용할 함수 및 포트를 선택할 수 있습니다. 식의 유효성을 검사할 수 있습니다.

다음 이미지는 필터 조건에 대한 매개 변수가 있는 필터 탭을 보여 줍니다.



식 매개 변수는 매핑 매개 변수입니다. 매개 변수 집합 또는 런타임 시 매개 변수 파일에서 매개 변수를 재정의할 수 있습니다.

## Null 값을 가진 행 필터링

Null 값 또는 공백을 포함하는 행을 필터링하려면 ISNULL 및 IS\_SPACES 함수를 사용하여 포트 값을 테스트하십시오.

예를 들어, FIRST\_NAME 포트에 NULL 값을 포함하는 행을 필터링할 경우 다음 조건을 사용하십시오.

```
IIF(ISNULL(FIRST_NAME),FALSE,TRUE)
```

이 조건은 FIRST\_NAME 포트가 NULL일 경우 반환 값이 FALSE이고 행이 무시됨을 기술합니다. 그렇지 않은 경우 행이 다음 변환으로 전달됩니다.

## 필터 변환 고급 속성

데이터 통합 서비스가 필터 변환에 대한 데이터 처리 방법을 결정할 수 있게 도와주는 속성을 구성하십시오.

로그의 추적 수준을 구성할 수 있습니다.

고급 탭에서 다음 속성을 구성합니다.

### 추적 수준

이 변환에 대해 로그에 표시되는 세부 정보의 양입니다. 간단, 보통, 자세한 정보 표시 초기화 또는 자세한 정보 표시 데이터를 선택할 수 있습니다. 기본값은 보통입니다.

## 필터 변환 성능 팁

필터 변환 성능을 개선하는 팁을 사용하십시오.

**매핑에서 조기에 필터 변환을 사용하십시오.**

필터 변환은 매핑에서 가능한 소스에 가까이 두십시오. 매핑을 통해 무시할 행을 전달하는 대신, 소스에서 대상까지의 데이터 흐름 초기에 원치 않는 데이터를 필터링할 수 있습니다.

## 필터 변환 - 비원시 환경

비원시 환경에서 필터 변환의 처리는 변환을 실행하는 엔진에 따라 다릅니다.

다음과 같은 비원시 런타임 엔진에 대한 지원을 고려하십시오.

- **Blaze** 엔진. 제한적으로 지원됩니다.
- **Spark** 엔진. 제한 없이 지원됩니다.
- **Databricks Spark** 엔진. 제한 없이 지원됩니다.

### 필터 변환 - Blaze 엔진

매핑에 **Hive** 소스의 분할된 열에 대한 필터 변환이 포함되는 경우 **Blaze** 엔진은 필터 조건을 만족하는 데이터가 있는 파티션만 읽을 수 있습니다. 필터를 **Hive** 소스로 푸시하려면 매핑에서 소스 다음에 위치하도록 필터 변환을 구성합니다.

## 제 18 장

# 계층-관계형 변환

이 장에 포함된 항목:

- [계층-관계형 변환 개요, 279](#)
- [예제 - 계층-관계형 변환, 279](#)
- [출력 관계형 포트 및 개요 보기, 281](#)
- [계층-관계형 변환 포트, 281](#)
- [스키마 참조, 282](#)
- [포트 구성, 282](#)
- [계층-관계형 변환 개발, 282](#)

## 계층-관계형 변환 개요

계층-관계형 변환에서는 XML 또는 JSON 계층 입력을 처리하여 이를 관계형 출력으로 변환합니다. 계층-관계형 변환은 입력 포트에서 계층 입력을 읽어 변환 출력 포트의 관계형 출력으로 데이터를 변환합니다. 계층 입력을 관계형 출력으로 변환하려면 스키마 파일을 사용하여 계층 데이터를 정의합니다.

계층-관계형 변환 마법사를 사용하여 데이터를 자동으로 매핑할 수 있습니다. 변환 **개요** 보기에서 매핑을 관계형 출력 포트에 구성할 수 있습니다.

마법사에서 변환을 생성하면 데이터를 관계형 출력 포트에서 매핑의 다른 변환으로 전달할 수 있습니다.

## 예제 - 계층-관계형 변환

Harrinder Shipping이라는 회사의 물류 부서에서는 발송 데이터를 처리해야 합니다. 그들은 재고 및 고객 데이터를 계층 형식에서 데이터베이스 테이블에 저장할 수 있는 관계형 데이터로 변환해야 합니다.

이를 위해서는 계층 데이터를 관계형 데이터로 변환하는 매핑을 작성해야 합니다. 회사의 재고 시스템에서 발송 재고 데이터를 계층 형식으로 생성합니다. 매핑에서 발송 데이터를 입력하고 사용 가능한 관계형 형식으로 세부 정보를 출력하는 계층-관계형 변환을 사용해야 합니다.

발송 입력은 계층 형식으로 되어 있습니다. 발송 요소에는 각 발송에 대해 고객 및 재고 데이터가 들어 있는 하위 요소가 포함되어 있습니다.

Shipments  
Shipment  
Items

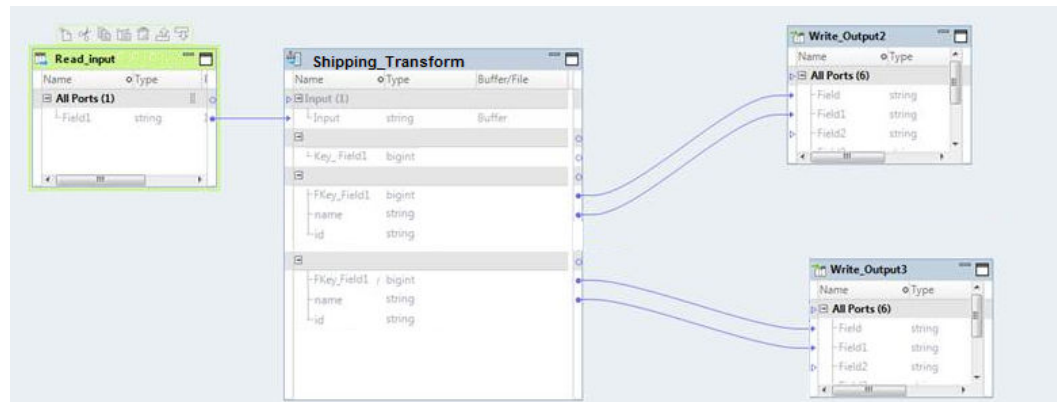
Item\_Name  
Inventory\_ID  
Customer  
Customer\_Name  
Customer\_ID  
Customer\_Address

관계형 출력에서 Customer\_ID 요소는 고객 테이블에서 기본 키이고 발송 테이블에서 외래 키입니다.

Customer_ID	Customer_Name	Customer_Address
3543766	Tony Birch	6 Moby Drive
6342562	Sujita Man	22 Dan Street
6471862	Dwayne Horace	7 Jafendar Boulevard
7265204	Carmela Perez	23 Dan Street
4559672	Delilah Soraya	28 Jafendar Boulevard

Shipment_ID	Inventory_Item	Customer_ID
9173327437	908274	7265204
9174562342	553439	7265204
8484526471	546584	3543766
7023847265	908274	3543766
9174596725	553439	3543766

다음 이미지는 이 예제의 매핑을 보여 줍니다.



이 매핑에는 다음 개체가 포함됩니다.

#### Read\_input

계층 데이터가 있는 파일에 대한 경로가 들어 있는 소스입니다. XML 파일에서 청구 데이터를 읽습니다.

#### Shipping\_Transform

XML 입력을 관계형 출력으로 변환하는 계층-관계형 변환입니다.

#### Write\_Output2

변환된 데이터의 일부인 고객 테이블을 관계형 형식으로 저장하는 대상입니다.

#### Write\_Output3

변환된 데이터의 또 다른 일부인 발송 테이블을 관계형 형식으로 저장하는 두 번째 대상입니다.

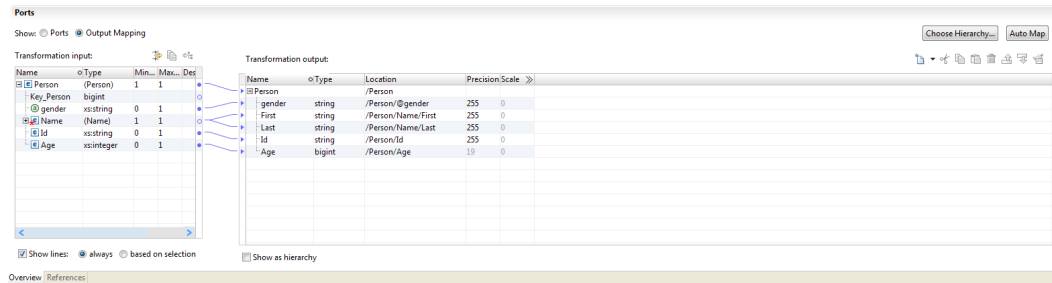
매핑에서 **Read\_input** 플랫폼 파일을 사용하여 계층 입력에 대한 대상 경로를 입력합니다. 매핑에서 **Shipping\_Transform** 변환을 사용하여 데이터를 처리하고 변환합니다. 그런 다음 출력을 두 개의 출력 대상에 저장합니다.

## 출력 관계형 포트 및 개요 보기

계층-관계형 변환에서 계층 데이터를 관계형 출력으로 변환하기 위해 마법사는 계층 노드와 관계형 포트 간 링크를 생성합니다. 관계형 포트와 계층 포트 간 링크를 보려면 **개요** 보기를 사용합니다. 계층 출력의 노드를 포트 그룹에 연결하여 출력 포트 그룹을 작성할 수도 있습니다.

관계형 그룹에 대한 매핑을 보려면 **개요** 보기를 사용합니다. **출력 매핑**을 선택합니다. **포트** 패널이 **개요** 보기에 나타납니다.

다음 이미지는 **포트** 패널을 보여 줍니다.



**변환 입력** 영역은 계층 스키마를 보여 주며 왼쪽에 있습니다. **변환 출력** 영역은 관계형 출력 포트를 보여 주며 오른쪽에 있습니다.

**변환 출력** 영역에서 관계형 출력 포트를 정의하고 스키마의 노드를 포트에 연결할 수 있습니다. 또한 스키마의 노드에서 **변환 출력** 영역의 빈 필드로 포인터를 끌어서 놓아 포트를 작성할 수도 있습니다. 출력 스키마의 노드를 포트에 끌어서 놓으면 **Developer** 도구가 둘 사이 링크를 표시합니다.

## 계층-관계형 변환 포트

변환 **개요** 보기에 계층-관계형 변환 포트가 정의되어 있습니다.

계층-관계형 변환은 파일 또는 버퍼에서 입력을 읽습니다. 출력 포트는 변환에서 관계형 데이터를 반환합니다.

계층-관계형 변환을 작성하면 **Developer tool**이 기본 입력 포트를 작성합니다. 입력 유형은 데이터 통합 서비스가 계층-관계형 변환으로 전달하는 데이터의 유형을 결정합니다. 입력 유형은 입력이 데이터인지, 아니면 소스 파일 경로인지를 결정합니다.

다음 입력 유형 중 하나를 구성하십시오.

### 버퍼

계층-관계형 변환이 입력 포트의 소스 데이터 행을 받습니다. 변환이 **Informatica** 변환으로부터 데이터를 받도록 구성하는 경우 버퍼 입력 유형을 사용합니다.

### 파일

계층-관계형 변환이 입력 포트의 소스 파일 경로를 받습니다. 계층-관계형 변환이 소스 파일을 엽니다. 버퍼 입력 포트를 처리하려면 많은 양의 시스템 메모리가 필요할 수 있는 대용량 파일에도 파일 입력 유형을 사용할 수 있습니다.

새 변환 마법사로 변환을 작성하는 경우 예제 입력 파일을 정의할 수 있습니다. 예제 입력 파일은 입력 파일의 작은 샘플입니다. 계층-관계형을 작성하는 경우 예제 입력 파일을 참조합니다. 또한 **데이터 뷰어** 보기에서 변환을 테스트할 때에도 예제 입력 파일을 사용합니다.

변환은 관계형 데이터를 반환하는 한 개 이상의 포트 그룹을 포함합니다.

## 스키마 참조

계층-관계형 변환에는 변환에서 입력 계층을 정의하기 위한 계층 스키마가 필요합니다. 변환에서 스키마를 사용하려면 스키마 참조를 정의합니다.

변환 **참조** 보기에서 변환 스키마 참조를 정의할 수 있습니다.

계층-관계형 변환은 모델 리포지토리의 스키마 개체를 참조합니다. 변환을 작성하기 전에 스키마 개체가 리포지토리에 있을 수 있습니다. 또한 변환 **참조** 보기에서 스키마를 가져올 수도 있습니다.

스키마는 다른 스키마를 참조할 수 있습니다. **참조** 보기는 계층-관계형 변환에서 참조하는 각 스키마에 대한 네임스페이스와 접두사를 표시합니다. 빈 네임스페이스의 여러 스키마를 참조할 경우 변환이 올바르게 작동하지 않습니다.

## 포트 구성

**포트** 패널에서, 변환은 계층 스키마 노드와 관계형 포트 간 매핑을 보여 줍니다. 변환은 스키마를 사용하여 계층 입력을 정의합니다. 스키마에 루트 요소가 될 수 있는 요소가 둘 이상 있는 경우 루트 요소가 될 노드를 선택합니다.

마법사는 계층 스키마 노드와 관계형 포트 간 링크를 생성합니다. 생성된 링크를 변경하려는 경우 **포트** 패널을 사용하여 링크를 추가, 삭제 또는 편집할 수 있습니다. 노드를 포트에 연결하고 포트를 작성할 수 있습니다.

노드를 **변환 출력** 영역에 연결할 때 **Developer tool**이 위치 필드를 계층의 노드 위치로 업데이트합니다. 수동으로 포트를 작성하는 경우 노드를 포트에 매핑해야 합니다. **위치** 열을 업데이트하고 목록에서 노드를 선택합니다.

여러 번 발생하는 노드를 상위 요소를 포함하는 그룹에 연결할 때 포함할 하위 요소 발생 수를 구성할 수 있습니다. 또는 상위 그룹을 변환 출력에서 여러 번 발생하는 하위 그룹으로 대체할 수 있습니다.

그룹을 생성하려면 노드를 **변환 출력** 영역의 빈 열에 연결합니다. 여러 번 발생하는 하위 노드를 빈 입력 또는 출력 열에 연결하면 **Developer tool**이 그룹을 다른 출력 그룹과 연결할지 묻는 메시지를 표시합니다. 그룹을 선택하면 **Developer tool**이 그룹을 연결하는 키를 작성합니다.

**변환 출력** 영역에서 관련 그룹의 출력 포트를 구성합니다. 출력 그룹을 연결할지 묻는 메시지가 나타나면 **Developer** 도구는 키를 그룹에 추가합니다. 키를 나타내는 포트를 수동으로 추가할 수도 있습니다.

## 계층-관계형 변환 개발

새 변환 마법사를 사용하여 계층-관계형 변환을 자동 생성합니다. 스키마 또는 계층 샘플 파일을 선택하여 입력 계층을 정의합니다.

1. **Developer tool**에서 변환을 작성합니다.
2. 입력 포트 및 매핑을 구성합니다.
3. 변환을 테스트합니다.

### 계층-관계형 변환 작성

1. **Developer tool**에서 **파일 > 새로 만들기 > 변환**을 클릭합니다.
2. 계층-관계형 변환을 선택하고 **다음**을 클릭합니다.

3. 변환의 이름을 입력하고 변환을 배치할 모델 리포지토리 위치를 찾은 다음 **다음**을 클릭합니다.
4. 스키마를 선택하려면 다음 방법 중 하나를 선택합니다.
  - 모델 리포지토리의 스키마를 사용하여 입력 계층을 정의하려면 **스키마 개체** 필드 근처에서 리포지토리의 스키마 파일을 찾아 선택합니다.
  - 스키마 파일을 가져오려면 **새 스키마 개체 작성**을 클릭합니다. **새 스키마 개체** 창에서 스키마 파일을 찾아 선택하거나 샘플 계층 파일에서 스키마를 작성하도록 선택할 수 있습니다.
5. 출력 계층의 루트를 선택합니다. **계층 루트** 대화 상자에서 출력 계층 파일에 대한 루트 요소인 스키마의 요소를 선택합니다. 루트 개체 선택에 도움이 되도록 샘플 계층 파일을 추가할 수 있습니다. 샘플 파일을 추가하려면 **샘플 파일** 필드 근처에서 파일 시스템의 파일을 찾아 선택합니다.
6. **마침**을 클릭합니다.  
마법사가 리포지토리에 변환을 작성합니다.

## 포트 및 매핑 구성

**개요** 보기에서 입력 및 출력 포트를 구성합니다.

1. 입력 포트 데이터 유형, 포트 유형, 전체 자릿수 및 소수 자릿수를 선택합니다.
2. 매핑을 보려면 **개요** 보기 **포트** 영역에서 **출력 매핑**을 선택합니다.
3. **포트** 그리드의 트리를 확장합니다. 왼쪽에 있는 **변환 입력** 패널에는 예상 계층 입력이 표시되고 오른쪽에 있는 **변환 출력** 패널에는 관계형 출력이 표시됩니다.
4. 노드를 루트로 정의하려면 **계층 선택**을 클릭합니다.  
Developer tool에는 루트 수준 및 루트 수준 아래의 노드만 **변환 입력** 영역에 표시합니다.
5. 포트를 계층 노드와 연결하는 선을 보려면 **선 표시**를 클릭합니다. 모든 연결 선을 표시하거나 선택한 포트의 선만 표시하도록 선택합니다.
6. 입력 그룹 또는 포트를 **변환 출력** 영역에 추가하려면 다음 방법 중 하나를 사용합니다.
  - **변환 입력** 영역에서 단순 또는 복합 요소를 **변환 출력** 영역의 빈 열로 끌어옵니다. 노드가 그룹 노드인 경우 Developer 도구가 포트 없이 관계형 그룹을 추가합니다.
  - 관계형 그룹을 추가하려면 행을 선택하고 마우스 오른쪽 단추를 클릭하여 **새로 만들기 > 그룹**을 선택합니다.
  - 관계형 포트를 추가하려면 마우스 오른쪽 단추를 클릭하고 **새로 만들기 > 필드**를 선택합니다.
7. 포트의 위치에 대한 계층 노드 설정을 지우려면 다음 방법 중 하나를 사용합니다.
  - **변환 입력** 영역에서 하나 이상의 노드를 선택하고 마우스 오른쪽 단추를 클릭하여 **지우기**를 선택합니다.
  - 관계형 포트를 계층 노드에 연결하는 하나 이상의 선을 선택하고 마우스 오른쪽 단추를 클릭한 다음 **삭제**를 선택합니다.
8. 계층에서 출력 포트를 표시하려면 **계층으로 보기**를 클릭합니다. 각 하위 그룹은 상위 그룹 아래에 나타납니다.

## 변환 테스트

**데이터 뷰어** 보기에서 계층-관계형 변환을 테스트합니다.

변환을 테스트하기 전에 파일 입력 위치를 정의했는지 확인합니다. DIS 시스템의 입력 위치를 **개요** 보기에서 **포트** 패널의 입력 위치 열에 정의합니다.

1. **데이터 뷰어** 보기를 엽니다.

2. **실행**을 클릭합니다.

Developer tool이 변환의 유효성을 검사합니다. 오류가 없으면 Developer tool이 **출력** 패널에 계층 파일 콘텐츠를 표시합니다.



## 제 19 장

# Java 변환

이 장에 포함된 항목:

- [Java 변환 개요, 285](#)
- [Java 변환 디자인, 289](#)
- [Java 변환 포트, 289](#)
- [Java 변환의 고급 속성, 290](#)
- [Java 코드 개발, 293](#)
- [Java 변환의 Java 속성, 297](#)
- [Java 변환을 통한 필터 최적화, 299](#)
- [Java 변환 작성, 301](#)
- [Java 변환 컴파일, 302](#)
- [Java 변환 문제 해결, 303](#)
- [구조체 데이터로 변환의 예, 304](#)
- [Java 변환 - 비원시 환경, 307](#)

## Java 변환 개요

Java 변환을 사용하여 Developer tool의 기능을 확장할 수 있습니다.

Java 변환은 Java 프로그래밍 언어가 포함된 변환 기능을 정의할 수 있는 단순한 원시 프로그래밍 인터페이스를 제공합니다. Java 변환을 사용하면 Java 프로그래밍 언어 또는 외부 Java 개발 환경에 대한 고급 지식이 없어도 단순하거나 적당히 복잡한 변환 기능을 정의할 수 있습니다. Java 변환은 활성 또는 수동 변환입니다.

Developer tool은 JDK(Java 개발 키트)를 사용하여 Java 코드를 정의하고 변환에 대한 바이트 코드를 생성합니다. Developer tool은 바이트 코드를 모델 리포지토리에 저장합니다.

데이터 통합 서비스는 런타임 시 생성된 바이트 코드를 JRE(Java Runtime Environment)를 사용하여 실행합니다. 데이터 통합 서비스는 Java 변환이 포함된 매핑을 실행할 때 JRE를 사용하여 바이트 코드를 실행하고 입력 행을 처리하고 출력 행을 생성합니다.

Java 변환을 작성하려면 변환 논리를 정의하는 Java 코드 조각을 작성해야 합니다. Java 변환에 대한 변환 동작은 다음과 같은 이벤트를 바탕으로 정의합니다.

- 변환이 입력 행을 수신합니다.
- 변환이 모든 입력 행을 처리했습니다.

매핑이 Spark 엔진에서 실행되는 경우 Java 변환에서 복합 데이터 유형을 사용하여 계층 데이터를 처리할 수 있습니다. 복합 데이터 유형을 사용하면 Spark 엔진이 Avro, Parquet 및 JSON 복합 파일의 계층 데이터를 직접 읽고 처리하고 씁니다.

## 재사용 가능 및 재사용 불가능 Java 변환

재사용 가능 또는 재사용 불가능 Java 변환을 작성할 수 있습니다.

재사용 가능 변환은 여러 매핑에 존재할 수 있습니다. 재사용 불가능 변환은 단일 매핑 내에 존재합니다.

속성을 정의하고 Java 코드를 작성하는 편집기의 보기는 재사용 가능 또는 재사용 불가능 Java 변환을 작성하는 지 여부에 따라 다릅니다.

## 활성 및 수동 Java 변환

Java 변환을 작성할 때는 변환 유형을 활성 또는 수동으로 정의해야 합니다.

변환 유형을 설정한 후에는 변경할 수 없습니다.

Java 변환은 사용자가 **입력 시입력 행에서** 탭에서 입력 데이터의 각 행에 대해 한 번 정의한 Java 코드를 실행합니다.

Java 변환은 다음과 같이 변환 유형에 따라 출력 행을 처리합니다.

- 수동 Java 변환은 변환의 각 입력 행을 처리한 후에 각 입력 행에 대해 1개의 출력 행을 생성합니다.
- 활성 Java 변환은 변환의 각 입력 행에 대해 여러 개의 출력 행을 생성합니다.

각 출력 행을 생성하려면 `generateRow` 메서드를 사용합니다. 예를 들어 변환에 시작 날짜 및 종료 날짜를 나타내는 입력 포트 2개가 포함되는 경우 `generateRow` 메서드를 사용하여 시작 날짜와 종료 날짜 사이의 각 날짜에 대한 출력 행을 생성할 수 있습니다.

## 데이터 유형 변환

Java 변환은 PowerCenterDeveloper tool 데이터 유형을 Java 변환의 포트 유형에 따라 Java 데이터 유형으로 변환합니다.

Java 변환은 입력 행을 읽고 입력 포트 데이터 유형을 Java 데이터 유형으로 변환합니다.

Java 변환이 출력 행을 기록할 때는 Java 데이터 유형을 출력 포트의 데이터 유형으로 변환합니다.

예를 들어 Java 변환에서 Integer 데이터 유형을 포함하는 입력 포트는 다음과 같이 처리됩니다.

- Java 변환이 입력 포트의 Integer 데이터 유형을 Java 기본 Integer 데이터 유형으로 변환합니다.
- Java 변환에서 Java 변환이 입력 포트의 값을 Java 기본 Integer 데이터 유형으로 처리합니다.
- Java 변환이 Java 기본 Integer 데이터 유형을 Integer 데이터 유형으로 변환하여 출력 행을 생성합니다.

다음 테이블에는 Java 변환이 PowerCenterDeveloper tool 데이터 유형을 Java 기본 및 복합 데이터 유형으로 매핑하는 방식이 나와 있습니다.

PowerCenter 데이터 유형	Java 데이터 유형
Char	String
Binary	byte[]
Long(INT32)	int

PowerCenter 데이터 유형	Java 데이터 유형
Double	double
Decimal	double BigDecimal
BIGINT	long
Date/Time	BigDecimal long(1970년 1월 1일 00:00:00.000 GMT 이후의 시간(밀리초))

Developer tool 데이터 유형	Java 데이터 유형
array*	java.util.List
bigint	long
binary	byte[]
date/time	나노초 처리를 활성화한 경우 나노초 전체 자릿수의 BigDecimal 나노초 처리를 비활성화한 경우 밀리초 전체 자릿수의 long(1970년 1월 1일 00:00:00.000 GMT 이후의 시간(밀리초))
decimal	많은 전체 자릿수 처리를 비활성화한 경우 전체 자릿수 15의 고정밀도 많은 전체 자릿수를 활성화한 경우 BigDecimal
double	double
integer	int
map*	java.util.Map
string	String
struct*	struct 필드 요소에 대한 getter 및 setter로 사용자 지정된 JavaBean 클래스
text	String
* Spark 엔진에서만 지원됩니다.	

Java에서 `java.util.List`, `java.util.Map`, `String`, `byte[]` 및 `BigDecimal` 데이터 유형은 복합 데이터 유형입니다. `double`, `int` 및 `long` 데이터 유형은 기본 데이터 유형입니다.

Developer tool에서 `array`, `struct` 및 `map` 데이터 유형은 복합 데이터 유형입니다.

**참고:** Java 변환은 기본 데이터 유형의 Null 값을 0으로 설정합니다. 입력 행에서 입력 시 탭에서 `isNull` 및 `setNull` API 메서드를 사용하여 입력 포트의 Null 값을 출력 포트의 Null 값으로 설정할 수 있습니다. 예는 [“setNull” 페이지 319](#)에서 확인하십시오.

## 복합 데이터 유형 변환 - Spark 엔진

Spark 엔진에서 실행되는 매핑의 Java 변환에 복합 포트를 추가할 수 있습니다. 복합 포트는 복합 데이터 유형에 할당되는 포트입니다. Java 변환은 복합 데이터 유형을 Java의 유사한 복합 데이터 유형으로 변환합니다. 또한 복합 데이터 유형의 요소를 유사한 Java 데이터 유형, 즉 기본 데이터 유형의 박스화된 버전으로 변환합니다.

### Array 데이터 유형

Java 변환은 입력 행을 읽을 때 Array 데이터 유형을 Java의 List 데이터 유형으로 변환합니다. 변환은 Array 요소의 데이터 유형을 Java 데이터 유형의 박스화된 버전으로 변환합니다.

예를 들어 Java 변환은 Developer tool 복합 데이터 유형인 `array<integer>`를 Java 데이터 유형인 `List<Integer>`로 변환합니다.

Java 변환이 출력 행을 쓸 때는 Java List 데이터 유형을 Array 데이터 유형으로 변환합니다. 변환은 List 요소의 데이터 유형을 Developer tool 데이터 유형의 초기 버전으로 변환합니다.

### Struct 데이터 유형

Java 변환은 Struct 데이터 유형의 입력 행을 읽을 때 상응하는 Java Bean 클래스를 생성합니다. Java Bean 클래스의 이름은 Struct 데이터 유형의 이름과 동일합니다. 변환은 구조체 요소의 데이터 유형을 Java 데이터 유형의 박스화된 버전으로 변환합니다.

Struct 데이터 유형에 특수 문자 또는 Java 예약 키워드(예: `final` 또는 `private`)가 사용되는 경우 Java 변환은 클래스 이름의 특수 문자를 밑줄(\_)로 대체합니다. 변환 속성 탭의 Java 보기에서 전체 코드를 열어 생성된 클래스를 확인할 수 있습니다.

Java 변환은 밑줄(\_)이 접두사로 추가된 클래스 멤버 필드 이름을 생성합니다. 또한 멤버 필드에 대한 `getter` 및 `setter`를 생성합니다.

생성된 Java Bean 클래스에는 유형 정의 라이브러리 이름의 네임스페이스가 외부 클래스로 포함됩니다. 외부 클래스의 이름은 유형 정의 라이브러리의 이름과 동일합니다. 구조체 포트의 Java 데이터 유형 이름에는 다음 형식이 사용됩니다.

```
type_library_name.struct_type_name
```

예를 들어 유형 라이브러리 이름이 `m_Type_Definition_Library`인 경우 구조체 포트의 복합 데이터 유형 정의는 다음과 같습니다.

```
Customer {
  name string
  age integer
}
```

Java 변환은 멤버 필드에 대한 `getter` 및 `setter`를 사용하여 Java Bean 클래스를 생성합니다. 다음 코드 조각은 외부 클래스 및 내부 클래스를 보여 줍니다.

```
public static final class m_Type_Definition_Library {
  public static final class Customer implements Serializable {
    private String _name;
    private Integer _age;

    public Customer() {}
```

다음 코드 조각은 문자열 유형의 구조체 요소 이름에 대한 `getter` 및 `setter`를 보여 줍니다.

```
public String get_name() {
  return _name;
}

public void set_name(String _name) {
  this._name = _name;
}
```

Java 변환은 Struct 데이터 유형의 출력 행을 쓸 때 Java Bean 클래스를 Struct 데이터 유형으로 변환합니다. 변환은 클래스의 멤버 필드를 구조체 요소로 변환합니다. 멤버 필드의 데이터 유형은 Developer tool 데이터 유형의 초기 버전으로 변환됩니다.

### Map 데이터 유형

Java 변환은 입력 행을 읽을 때 Map 데이터 유형을 Java의 Map 데이터 유형으로 변환합니다. 변환은 맵 요소의 데이터 유형을 Java 데이터 유형의 박스화된 버전으로 변환합니다.

예를 들어 Developer tool의 복합 데이터 유형인 map<string, bigint>는 Java 데이터 유형인 Map<String, long>으로 변환됩니다.

Java 변환이 출력 행을 쓸 때는 Java Map 데이터 유형을 Map 데이터 유형으로 변환합니다. 변환은 Map 요소의 데이터 유형을 Developer tool 데이터 유형의 초기 버전으로 변환합니다.

## Java 변환 디자인

Java 변환을 디자인할 때는 작성할 변환의 유형과 같은 요인을 고려해야 합니다.

Java 변환을 디자인하는 경우 다음과 같은 질문을 고려하십시오.

- 활성 또는 수동 Java 변환 중 어느 변환을 작성해야 하나?  
수동 Java 변환은 변환의 각 입력 행에 대해 1개의 출력 행을 생성합니다.  
활성 Java 변환은 변환의 각 입력 행에 대해 여러 개의 출력 행을 생성합니다.
- Java 변환에서 함수를 정의해야 하나? 그렇다면 각 함수에 어떤 식을 포함할 것입니까?  
예를 들어 입력 또는 출력 포트의 값을 조회하거나 Java 변환 변수의 값을 조회하는 식을 호출하는 함수를 정의할 수 있습니다.
- 재사용 가능 또는 재사용 불가능 Java 변환 중 어느 변환을 작성할 것입니까?  
재사용 가능 변환은 여러 매핑에 존재할 수 있습니다.  
재사용 불가능 변환은 단일 매핑에만 존재할 수 있습니다.

## Java 변환 포트

Java 변환에는 입력 포트와 출력 포트가 있습니다.

재사용 불가능 Java 변환의 포트를 작성하고 편집하려면 편집기의 **포트** 탭을 사용합니다. 재사용 가능한 Java 변환의 포트를 작성하고 편집하려면 편집기의 **개요** 보기를 사용합니다.

포트의 기본값을 지정할 수 있습니다. 변환에 포트를 추가한 후 해당 포트 이름을 Java 코드 조작의 변수로 사용할 수 있습니다.

### 그룹 및 포트 작성

Java 변환을 작성하면 입력 그룹 1개와 출력 그룹 1개가 Java 변환에 포함됩니다.

Java 변환에는 항상 1개의 입력 그룹과 1개의 출력 그룹이 포함됩니다. 입력 또는 출력 그룹이 여러 개인 Java 변환은 유효하지 않습니다. 그룹 헤더에 그룹 이름을 입력하여 기존 그룹 이름을 변경할 수 있습니다. 그룹을 삭제한 경우 입력 그룹 작성 또는 출력 그룹 작성 아이콘을 클릭하여 새 그룹을 추가할 수 있습니다.

포트를 작성하면 **DesignerDeveloper tool**이 현재 선택된 행 또는 그룹 아래에 포트를 추가합니다. 입력 그룹 아래에 표시되는 입력/출력 포트는 출력 그룹에도 포함됩니다. 출력 그룹 아래에 표시되는 입력/출력 포트는 입력 그룹에도 포함됩니다.

## 기본 포트 값 설정

Java 변환에서 포트에 대한 기본값을 정의할 수 있습니다.

Java 변환은 포트의 데이터 유형에 따라 기본 포트 값을 사용하여 포트 변수를 초기화합니다.

### 입력 및 출력 포트

Java 변환은 Java 코드 조각에서 할당된 값이 없는 연결되지 않은 입력 포트 또는 출력 포트의 값을 초기화합니다.

Java 변환은 다음 Java 데이터 유형에 따라 포트를 초기화합니다.

#### 기본 데이터 유형

포트에 대해 **null**이 아닌 기본값을 정의한 경우 변환에서 포트 변수 값을 기본값으로 초기화합니다. 그렇지 않은 경우 포트 변수 값을 **0**으로 초기화합니다.

#### 복합 데이터 유형

포트에 대한 기본값을 정의한 경우 변환에서 새 개체를 생성하고 개체를 기본값으로 초기화합니다. 그렇지 않은 경우 변환에서 포트 변수를 **Null**로 초기화합니다. 예를 들어 **string** 포트에 대한 기본값을 정의한 경우 변환이 새 **String** 개체를 생성하고 **String** 개체를 기본값으로 초기화합니다.

**참고:** Java 코드에서 **Null** 값을 가진 입력 포트 변수에 액세스할 경우 **NullPointerException**이 발생합니다.

입력 포트를 파티션 키 및 정렬 키로 활성화할 수 있으며, 정렬 방향을 할당할 수 있습니다. 데이터 통합 서비스는 데이터를 분할하고 정렬 키와 정렬 방향을 기준으로 각 파티션의 데이터를 정렬합니다. 변환 범위가 모든 입력으로 설정된 경우 파티션 키 및 정렬 키가 사용될 수 있습니다.

데이터 파티션 및 정렬을 위해 다음 속성을 사용하십시오.

#### 파티션 키

동일한 파티션으로 그룹화할 데이터 행을 결정하는 입력 포트입니다.

#### 정렬 키

각 파티션 내에서 정렬 조건을 결정하는 입력 포트입니다.

#### 방향

오름차순 또는 내림차순 순서입니다. 기본값은 오름차순입니다.

## Java 변환의 고급 속성

Java 변환에는 변환 코드 및 변환을 위한 고급 속성이 포함됩니다.

Java 변환을 매핑에 사용하는 경우 변환 속성을 재정의할 수 있습니다.

고급 탭에서 Java 변환에 대한 다음 고급 속성을 정의할 수 있습니다.

#### 추적 수준

이 변환에 대해 로그에 표시되는 세부 정보의 양입니다. 간단, 보통, 자세한 정보 표시 초기화 또는 자세한 정보 표시 데이터를 선택할 수 있습니다. 기본값은 보통입니다.

## 분할 가능

여러 스레드를 사용하여 변환을 처리할 수 있습니다. 데이터 통합 서비스가 변환을 처리하는 데 하나의 스레드를 사용하도록 하려면 이 옵션을 선택 취소합니다. 데이터 통합 서비스는 여러 스레드를 사용하여 나머지 매핑 파이프라인 단계를 처리할 수 있습니다.

Java 코드를 사용하려면 변환이 한 개의 스레드로 처리되어야 하는 경우 Java 변환에 대해 분할을 비활성화합니다.

## 높은 정밀도 설정

전체 자릿수가 38보다 작거나 같은 10진수 데이터 유형 포트를 Java BigDecimal 데이터 유형 포트로 처리합니다.

10진수 데이터 유형 포트를 Java Double 데이터 유형 포트로 처리하려면 많은 전체 자릿수를 비활성화합니다.

다음 표는 많은 전체 자릿수 옵션의 활성화 또는 비활성화에 따라 Java 변환이 10진수 데이터 유형 입력 포트의 값을 처리하는 방법을 보여 줍니다.

예	높은 정밀도 처리가 활성화됨	높은 정밀도 처리가 비활성화됨
10진수 유형의 입력 포트에 값 40012030304957666903이 수신됩니다.	Java 변환이 값을 그대로 유지합니다.	Java 변환이 값을 다음 값으로 변환합니다. $4.00120303049577 \times 10^{19}$

Java 변환에 10진수 포트 또는 10진수 데이터 유형의 요소가 있는 복합 포트가 포함되는 경우 변환에는 매핑과 동일한 전체 자릿수 모드가 사용되어야 합니다. 예를 들어 Java 변환에서 많은 전체 자릿수를 사용하는 경우 매핑에서 많은 전체 자릿수를 활성화해야 합니다.

## 날짜/시간에 나노초 사용

날짜/시간 데이터 유형 포트를 나노초 전체 자릿수의 Java BigDecimal 데이터 유형 포트로 변환합니다.

나노초 처리를 비활성화하면 생성된 Java 코드가 날짜/시간 데이터 유형 포트를 밀리초 전체 자릿수의 Java 긴 정수 데이터 유형 포트로 변환합니다.

## 클래스 경로

가져오기 탭에서 가져오는 비표준 Java 패키지에 연결된 jar 또는 클래스 파일 디렉터리에 대한 클래스 경로를 설정합니다.

Java 코드를 컴파일하려면 Developer tool 클라이언트 시스템이 jar 또는 클래스 파일 디렉터리에 액세스할 수 있어야 합니다.

운영 체제에 따라 다음과 같이 클래스 경로 항목을 구분하십시오.

- UNIX의 경우 콜론을 사용하여 클래스 경로 항목을 구분합니다.
- Windows의 경우 세미콜론을 사용하여 클래스 경로 항목을 구분합니다.

예를 들어 가져오기 탭에서 Java 변환기 패키지를 가져오고 converter.jar에 패키지를 정의하는 경우 Java 변환에 대한 Java 코드를 컴파일하기 전에 converter.jar 파일의 위치를 클래스 경로에 추가해야 합니다.

**참고:** 기본 제공 Java 패키지에 대한 클래스 경로는 설정하지 않아도 됩니다. 예를 들어 java.io는 기본 제공 Java 패키지이므로 java.io에 대한 클래스 경로를 설정하지 않아도 됩니다.

## 활성 여부

Java 변환에서 각 입력 행에 대해 2개 이상의 출력 행을 생성할 수 있습니다.

이 속성은 **Java** 변환을 작성한 후에 변경할 수 없습니다. 이 속성을 변경하려면 새 **Java** 변환을 작성해야 합니다.

#### 변환 범위

데이터 통합 서비스가 수신 데이터에 **Java** 변환 논리를 적용할 때 사용하는 방법을 정의합니다. 다음 값 중 하나를 선택할 수 있습니다.

- **행**. 한 번에 데이터의 한 행에 변환 논리를 적용합니다. 데이터의 단일 행에 따라 프로시저의 결과가 결정되는 경우 행을 선택합니다.
- **트랜잭션**. 트랜잭션의 모든 행에 변환 논리를 적용합니다. 다른 트랜잭션의 행이 아닌 동일한 트랜잭션의 모든 행에 따라 프로시저의 결과가 결정되는 경우 트랜잭션을 선택합니다. 트랜잭션을 선택할 경우 모든 입력 그룹을 동일한 트랜잭션 제어점에 연결해야 합니다.
- **모든 입력**. 모든 수신 데이터에 변환 논리를 적용합니다. 모든 입력을 선택하면 데이터 통합 서비스가 트랜잭션 경계를 제거합니다. 소스 데이터의 모든 행에 따라 프로시저의 결과가 결정되는 경우 모든 입력을 선택합니다.

**참고:** 변환 범위 속성은 Hive 환경에서만 유효합니다.

#### 상태 비저장

변환에 대한 입력 데이터의 행 순서를 유지합니다. 데이터 통합 서비스에서 행 순서를 변경할 수 있는 어떤 최적화도 수행하지 않는 경우 이 옵션을 선택합니다.

데이터 통합 서비스가 최적화를 수행하는 경우 이전에 매핑에 설정된 순서를 잃게 될 수 있습니다. 정렬된 플랫폼 파일 소스, 정렬된 관계형 소스 또는 분류기 변환을 사용하여 매핑에서 순서를 설정할 수 있습니다. 행 순서를 유지하기 위해 변환을 구성하는 경우 데이터 통합 서비스는 매핑에 대해 최적화를 수행할 때 이 구성을 고려합니다. 데이터 통합 서비스는 순서를 유지할 수 있을 때 변환에 대해 최적화를 수행합니다. 데이터 통합 서비스는 최적화가 행 순서를 변경할 수 있는 경우 변환에 대해 최적화를 수행하지 않습니다.

## PowerCenterDeveloper tool 클라이언트에 대한 클래스 경로 구성

jar 파일 또는 클래스 파일 디렉터리를 **PowerCenter** 클라이언트Developer tool 클라이언트 클래스 경로에 추가할 수 있습니다.

**PowerCenter** 클라이언트Developer tool 클라이언트의 클라이언트가 실행되는 시스템에 대한 클래스 경로를 설정하려면 다음 작업을 완료하십시오.

- **CLASSPATH** 환경 변수를 구성합니다. **CLASSPATH** 환경 변수는 **PowerCenter** 클라이언트Developer tool 클라이언트의 클라이언트 시스템에서 설정합니다. 이 설정은 시스템에서 실행되는 모든 **Java** 프로세스에 적용됩니다.
- 재사용 불가능 **Java** 변환의 경우Java 변환의 설정고급 속성에서 클래스 경로를 구성합니다. 이 설정은 이 **Java** 변환을 포함하는 세션매핑에 적용됩니다. **PowerCenter** 클라이언트Developer tool 클라이언트의 클라이언트는 **Java** 코드를 컴파일할 때 이 클래스 경로에 파일을 포함합니다.

jar 또는 클래스 파일 디렉터리를 **Java** 변환의 클래스 경로에 추가하려면 다음 단계를 완료하십시오.

1. **Java 코드고급** 탭에서 **설정** 링크를 클릭**클래스 경로** 옆의 값 열에 있는 아래쪽 화살표 아이콘을 클릭합니다.  
**설정클래스 경로 편집** 대화 상자가 표시됩니다.
2. 클래스 경로를 추가하려면 다음 단계를 완료하십시오.
  - a. **추가**를 클릭합니다.  
**다른 이름으로 저장** 창이 표시됩니다.
  - b. **다른 이름으로 저장** 창에서 jar 파일이 위치한 디렉터리로 이동합니다.



c. **확인**을 클릭합니다.

**클래스 경로 편집** 대화 상자에 클래스 경로가 표시됩니다.

3. **클래스 경로 추가** 아래에서 찾아보기를 클릭하고 가져온 패키지에 대한 **jar** 파일 또는 클래스 파일 디렉터리를 선택합니다. **확인**을 클릭합니다.

4. **추가**를 클릭합니다.

**jar** 또는 클래스 파일 디렉터리가 변환의 **jar** 및 클래스 파일 디렉터리 목록에 표시됩니다.

5. **jar** 파일 또는 클래스 파일 디렉터를 제거하려면 **jar** 또는 클래스 파일 디렉터를 선택하고 **제거**를 클릭합니다.

디렉터리 목록에서 디렉터리가 사라집니다.

## 데이터 통합 서비스에 대한 클래스 경로 구성

런타임 시 데이터 통합 서비스 노드의 클래스 경로에 필요한 **jar** 또는 클래스 파일 디렉터를 추가할 수 있습니다.

런타임 중에 필요한 **jar** 파일을 데이터 통합 서비스 노드의 다음 디렉터리에 배치하십시오.

`$INFA_HOME/services/shared/jars`

이 위치의 **jar** 파일은 동적으로 로드됩니다. 런타임 시 개별 매핑에 필요한 모든 클래스 파일을 이 디렉터리에서 찾아 로드할 수 있습니다.

**참고:** Java 변환은 이 디렉터리의 **jar** 파일을 매핑 수준 클래스 경로에 추가합니다.

## Java 코드 개발

특정 변환 이벤트에 대한 변환 동적을 정의하는 **Java** 코드를 기록하고 컴파일하려면 **Java** 보기의 코드 항목 탭을 사용합니다.

코드 항목 탭의 코드 조각은 원하는 순서로 개발할 수 있습니다. **전체 코드** 탭에서는 전체 **Java** 코드를 볼 수 있지만 편집할 수는 없습니다.

코드 조각을 개발한 후에 코드 조각 또는 전체 **Java** 코드를 컴파일하고 **Java** 보기에 있는 **컴파일** 속성의 **결과** 창에서 결과를 볼 수 있습니다.

각 코드 항목 탭에는 **Java** 코드를 기록하고 보고 컴파일할 때 사용하는 구성 요소가 포함됩니다.

## 코드 속성

Java 변환 API 메시드와 같이 Java 코드를 보고 입력할 때 사용하는 컨트롤을 제공합니다. 다음 표에는 코드 속성에서 제공되는 컨트롤이 설명되어 있습니다.

제어	설명
탐색기	<p>입력 포트, 출력 포트 및 호출 가능한 Java 변환 API 메시드를 표시합니다.</p> <p>탐색기에서 항목을 클릭하면 항목의 설명이 표시됩니다.</p> <p>항목을 Java 코드 창에 추가하려면 항목을 두 번 클릭합니다. 또는 탐색기의 항목을 Java 코드 창으로 끌어옵니다.</p> <p>탐색기는 다음 코드 항목 탭에서 사용할 수 있습니다.</p> <ul style="list-style-type: none"> <li>- 도우미</li> <li>- 입력 시</li> <li>- 끝에서</li> </ul>
Java 코드 창	<p>변환의 Java 코드를 보거나 입력할 수 있습니다. Java 코드 창은 기본 Java 구문 강조 표시를 사용하여 Java 코드를 표시합니다.</p> <p><b>참고:</b> 전체 코드 탭에서는 Java 변환의 전체 클래스 코드를 볼 수 있지만 편집할 수는 없습니다.</p> <p>Java 코드 창은 다음 코드 항목 탭에서 사용할 수 있습니다.</p> <ul style="list-style-type: none"> <li>- 가져오기</li> <li>- 도우미</li> <li>- 입력 시</li> <li>- 끝에서</li> <li>- 함수</li> <li>- 최적화 프로그램 인터페이스</li> <li>- 전체 코드</li> </ul>
새 함수 명령	<p>Java 식을 호출하는 함수를 정의할 때 사용하는 함수 정의 대화 상자를 엽니다.</p> <p>함수 명령은 함수 탭에서 사용할 수 있습니다.</p>
편집 도구 모음	<p>잘라내기, 복사 및 붙여넣기 등 Java 코드 편집을 위한 도구 아이콘을 클릭할 수 있습니다.</p> <p>편집 도구 모음은 다음 코드 항목 탭에서 사용할 수 있습니다.</p> <ul style="list-style-type: none"> <li>- 가져오기</li> <li>- 도우미</li> <li>- 입력 시</li> <li>- 끝에서</li> <li>- 함수</li> <li>- 최적화 프로그램 인터페이스</li> </ul>

## 컴파일 속성

Java 코드의 컴파일 및 디버그를 위한 컨트롤을 제공합니다. 다음 표에는 컴파일 속성의 컨트롤이 설명되어 있습니다.

제어	설명
컴파일 명령	변환의 Java 코드를 컴파일합니다.
결과 창	<p>Java 변환 클래스의 컴파일 결과를 확인하고 코드에서 오류의 소스를 찾을 수 있습니다.</p> <p>코드의 오류를 찾으려면 결과 창에서 오류 메시지를 마우스 오른쪽 단추로 클릭하고 조각 코드 또는 전체 코드의 오류를 선택하여 봅니다.</p> <p>결과 창의 오류 메시지를 두 번 클릭하여 오류의 소스를 볼 수도 있습니다.</p>

## Java 코드 조각 작성

Java 코드 조각을 작성하여 변환 동작을 정의하려면 **Java 코드** 탭의 **코드** 탭의 **코드** Java 코드를 사용합니다.

1. 해당하는 코드 항목 탭을 클릭합니다.

다음 표에는 **Java** 보기의 코드 항목 탭에서 완료할 수 있는 태스크가 설명되어 있습니다.

탭	설명
가져오기	활성 또는 수동 Java 변환에 대한 타사, 기본 제공 및 사용자 지정 Java 패키지를 가져옵니다. 가져온 패키지를 다른 코드 항목 탭에서 사용할 수 있습니다.
도우미	활성 또는 수동 Java 변환의 Java 변환 클래스에 대한 사용자 정의 변수 및 메서드를 선언합니다. 선언된 변수 및 메서드는 <b>가져오기</b> 탭을 제외한 다른 코드 항목 탭에서 사용할 수 있습니다.
입력 시	활성 또는 수동 Java 변환이 입력 행을 수신할 때의 동작을 정의합니다. 이 탭에서 정의한 Java 코드는 각 입력 행에 대해 한 번 실행됩니다. 이 탭에서 입력 및 출력 포트 데이터, 변수 및 Java 변환 API 메서드에 액세스하고 사용할 수 있습니다.
끝에서	활성 또는 수동 Java 변환이 모든 입력 데이터를 처리한 후의 변환 동작을 정의합니다. 이 탭에서 활성 변환의 출력 데이터를 설정하고 Java 변환 API 메서드를 호출할 수도 있습니다.
함수	Java 프로그래밍 언어가 포함된 Java 변환에서 식을 호출하는 함수를 정의합니다. 예를 들어 입력 또는 출력 포트의 값을 조회하거나 Java 변환 변수의 값을 조회하는 식을 호출하는 함수를 정의할 수 있습니다. 함수 탭에서 수동으로 함수를 정의하거나 새 함수를 클릭하고 함수 정의 대화 상자를 호출하여 함수를 쉽게 정의할 수 있습니다.
최적화 프로그램 인터페이스	초기 선택 또는 푸시인 필터 최적화를 정의합니다. 탐색기에서 최적화 방법을 선택합니다. 코드 조각을 업데이트하여 최적화를 활성화합니다. 필터 논리를 푸시할 입력 포트 및 연결된 출력 포트를 정의합니다.
전체 코드	읽기 전용. 이 탭에서 Java 변환의 전체 클래스 코드를 보고 컴파일할 수 있습니다.

2. 조각의 입력 또는 출력 열 변수에 액세스하려면 탐색기에서 포트 이름을 두 번 클릭탐색기의 **입력** 또는 **출력** 목록을 확장하고 포트의 이름을 두 번 클릭합니다.
3. 조각의 Java 변환 API를 호출하려면 탐색기에서 API 이름을 두 번 클릭합니다. 필요한 경우 적절한 API 입력 값을 구성합니다.탐색기에서 **호출 가능 API** 목록을 확장하고 메서드의 이름을 두 번 클릭합니다. 필요한 경우 메서드의 적절한 입력 값을 구성합니다.
4. 코드 조각코드 항목 탭 유형을 바탕으로 적절한 Java 코드를 작성합니다.  
전체 코드Java 코드 창(전체 코드 탭의)에서 Java 변환의 전체 클래스 코드를 봅니다.

## Java 패키지 가져오기

패키지 가져오기가져오기 탭에서 활성 또는 수동 Java 변환의 Java 패키지를 가져올 수 있습니다.

타사, 기본 제공 또는 사용자 지정 Java 패키지를 가져올 수 있습니다. Java 패키지를 가져온 후 다른 코드 항목 탭에서 가져온 패키지를 사용할 수 있습니다.

**참고:** 패키지 가져오기가져오기 탭에서는 정적 변수, 인스턴스 변수 또는 사용자 메서드를 선언할 수 없습니다.

PowerCenter 클라이언트 Developer tool에서 Java 변환이 포함된 메타데이터를 내보내거나 가져오는 경우 Java 변환에 필요한 타사 또는 사용자 지정 패키지를 포함하는 jar 또는 클래스 파일은 내보내기 또는 가져오기에 포함되지 않습니다.

Java 변환이 포함된 메타데이터를 가져오는 경우에는 필요한 타사 또는 사용자 지정 패키지가 포함된 jar 또는 클래스 파일을 PowerCenter 클라이언트 및 통합 서비스 노드 Developer tool 클라이언트 및 데이터 통합 서비스 노드에 복사해야 합니다.

예를 들어 Java I/O 패키지를 가져오려면 **패키지 가져오기** 탭에서 다음 코드를 입력합니다.

```
import java.io.*;
```

비표준 Java 패키지를 가져오는 경우 패키지 또는 클래스를 Java 변환 설정의 클래스 경로에 추가합니다.

## 도우미 코드 정의

**도우미 코드** 도우미 탭에서 활성 또는 수동 Java 변환의 Java 변환 클래스에 대한 사용자 정의 변수 및 메서드를 선언할 수 있습니다.

**도우미 코드** 도우미 탭에서 선언한 변수 및 메서드는 **패키지 가져오기** 탭을 제외한 모든 코드 항목 탭에서 사용할 수 있습니다.

**도우미 코드** 도우미 탭에서 다음 유형의 코드, 변수 및 메서드를 선언할 수 있습니다.

- 정적 코드 및 정적 변수.

정적 블록 안에 정적 변수 및 정적 코드를 선언할 수 있습니다. 세션의 모든 파티션 및 매핑에서 재사용 가능한 Java 변환의 모든 인스턴스는 정적 코드 및 변수를 공유합니다. 정적 코드는 Java 변환의 다른 코드보다 먼저 실행됩니다.

예를 들어 다음 코드는 매핑에서 Java 변환의 모든 인스턴스에 대한 오류 임계값을 저장하는 정적 변수를 선언합니다.

```
static int errorThreshold;
```

변환의 오류 임계값을 저장하고 세션의 모든 파티션 및 매핑의 모든 Java 변환 인스턴스에서 오류 임계값에 액세스하려면 이 변수를 사용합니다.

**참고:** 여러 파티션 세션 또는 재사용 가능한 Java 변환의 정적 변수를 동기화해야 합니다.

- 인스턴스 변수.

파티션 수준 인스턴스 변수를 선언할 수 있습니다. 세션의 여러 파티션 또는 매핑에서 재사용 가능한 Java 변환의 여러 인스턴스는 인스턴스 변수를 공유하지 않습니다. 충돌을 방지하고 비기본 인스턴스 변수를 초기화하려면 접두사를 사용하여 인스턴스 변수를 선언하십시오.

예를 들어 다음 코드는 부울 변수를 사용하여 출력 행의 생성 여부를 결정합니다.

```
// boolean to decide whether to generate an output row
// based on validity of input
private boolean generateRow;
```

- 사용자 정의 정적 메서드 또는 인스턴스 메서드.

Java 변환의 기능을 확장합니다. **도우미 코드** 도우미 탭에서 선언된 Java 메서드는 출력 변수 또는 로컬로 선언된 인스턴스 변수를 사용하거나 수정할 수 있습니다. **도우미 코드** 도우미 탭에서는 Java 메서드의 입력 변수에 액세스할 수 없습니다.

예를 들어 두 정수를 더하는 함수를 선언하려면 **도우미 코드** 도우미 탭에서 다음 코드를 사용합니다.

```
private int myTXAdd (int num1,int num2)
{
    return num1+num2;
}
```

# Java 변환의 Java 속성

특정 변환 이벤트에 대한 변환 동적을 정의하는 Java 코드를 기록하고 컴파일하려면 **Java** 보기의 코드 항목 탭을 사용합니다.

다음 탭은 코드 항목 탭입니다.

- 가져오기
- 도우미
- 입력 시
- 끝에서
- 함수
- 최적화 프로그램 인터페이스

Java 변환의 전체 클래스 코드는 **전체 코드** 탭에서 볼 수 있습니다.

## 가져오기 탭

**가져오기** 탭에서는 활성 또는 수동 Java 변환의 타사, 기본 제공 또는 사용자 지정 Java 패키지를 가져올 수 있습니다.

Java 패키지를 가져오려면 **가져오기** 탭에 있는 **코드** 속성의 **Java 코드** 창에 패키지를 가져오는 코드를 입력합니다.

예를 들어 `java.io` 패키지를 가져오려는 경우 다음 코드를 입력할 수 있습니다.

```
import java.io.*;
```

Java 패키지를 가져오는 코드를 컴파일하려면 **가져오기** 탭의 **컴파일** 속성에서 **컴파일**을 클릭합니다. 컴파일 결과는 **가져오기** 탭의 **결과** 창에 표시됩니다.

가져온 Java 패키지는 다른 코드 항목 탭에서 사용할 수 있습니다.

## 도우미 탭

**도우미** 탭에서는 활성 또는 수동 Java 변환의 Java 변환 클래스에 대한 사용자 정의 변수 및 메서드를 선언할 수 있습니다.

사용자 정의 변수 및 메서드를 선언하려면 **도우미** 탭에 있는 **코드** 속성의 **Java 코드** 창에 코드를 입력합니다.

Java 변환의 도우미 코드를 컴파일하려면 **도우미** 탭의 **컴파일** 속성에서 **컴파일**을 클릭합니다. 컴파일 결과는 **도우미** 탭의 **결과** 창에 표시됩니다.

선언된 변수 및 메서드는 **가져오기** 탭을 제외한 다른 코드 항목 탭에서 사용할 수 있습니다.

## 입력 시 탭

**입력 시** 탭에서는 활성 또는 수동 Java 변환이 입력 행을 받을 때 동작하는 방식을 정의합니다. 이 탭에서 입력 및 출력 포트 데이터, 변수 및 Java 변환 API 메서드에 액세스하고 사용할 수 있습니다.

이 탭에서 정의한 Java 코드는 각 입력 행에 대해 한 번 실행됩니다.

Java 변환이 입력 행을 받을 때 동작하는 방식을 정의하려면 **입력 시** 탭에서 **코드** 속성의 **Java 코드** 창에 코드를 입력합니다.

**입력 시** 탭의 탐색기에서 다음 변수 및 API 메서드를 액세스 및 정의할 수 있습니다.

- 입력 및 출력 포트 변수입니다. 포트 이름을 변수 이름으로 사용하여 입력 및 출력 포트 데이터에 변수로 액세스합니다. 예를 들어 "in\_int"가 정수 입력 포트이면 Java 기본 데이터 유형 `int`가 포함된 변수 "in\_int"로 참조하여 이 포트의 데이터에 액세스할 수 있습니다. 입력 및 출력 포트를 변수로 선언할 필요는 없습니다.

입력 포트 변수에는 값을 할당하지 마십시오. **입력 시** 탭에서 입력 변수에 값을 할당하면 현재 행에서 해당 포트에 대한 입력 데이터를 가져올 수 없습니다.

- 인스턴스 변수 및 사용자 정의 메서드. **도우미** 탭에서 선언한 인스턴스나 정적 변수 또는 사용자 정의 메서드를 사용합니다.

활성 Java 변환에 정수 데이터 유형의 두 입력 포트 `BASE_SALARY` 및 `BONUSES`와 정수 데이터 유형의 출력 포트 `TOTAL_COMP` 하나가 포함되는 경우를 예로 들어 보겠습니다. 이 경우 **도우미** 탭에서 두 정수를 추가하고 결과를 반환하는 사용자 정의 메서드인 `myTXAdd`를 생성합니다. 입력 포트의 전체 값을 출력 포트에 할당하고 출력 행을 생성하려면 **입력 시** 탭에서 다음 Java 코드를 사용합니다.

```
TOTAL_COMP = myTXAdd (BASE_SALARY,BONUSES);
generateRow();
```

Java 변환은 입력 행을 받으면 `BASE_SALARY` 및 `BONUSES` 입력 포트의 값을 추가하고 `TOTAL_COMP` 출력 포트에 해당 값을 할당한 다음 출력 행을 생성합니다.

- Java 변환 API 메서드. Java 변환에서 제공하는 API 메서드를 호출할 수 있습니다.

Java 변환의 코드를 컴파일하려면 **입력 시** 탭의 **컴파일** 속성에서 **컴파일**을 클릭합니다. 컴파일 결과가 **입력 시** 탭의 **결과** 창에 표시됩니다.

## 끝에서 탭

**끝에서** 탭에서는 활성 또는 수동 Java 변환이 모든 입력 데이터를 처리한 후의 변환 동작을 정의할 수 있습니다. 이 탭에서 활성 변환의 출력 데이터를 설정하고 Java 변환 API 메서드를 호출할 수도 있습니다.

Java 변환이 모든 입력 데이터를 처리한 후의 변환 동작을 정의하려면 **끝에서** 탭에 있는 **코드** 속성의 **Java 코드** 창에 코드를 입력합니다.

**끝에서** 탭에서 다음 변수 및 API 메서드를 정의하고 액세스할 수 있습니다.

- 출력 포트 변수. **포트** 탭에서 정의한 모든 출력 포트의 이름을 변수로 사용하거나 활성 Java 변환에 대한 출력 데이터를 설정할 수 있습니다.
- 인스턴스 변수 및 사용자 정의 메서드. **도우미** 탭에서 선언한 모든 인스턴스 변수 또는 사용자 정의 메서드를 사용할 수 있습니다.
- Java 변환 API 메서드. Java 변환이 제공하는 API 메서드를 호출합니다.

예를 들어 데이터의 끝에 도달할 때 로그에 정보를 기록하려면 다음 Java 코드를 사용합니다.

```
logInfo("Number of null rows for partition is: " + partCountNullRows);
```

Java 변환의 코드를 컴파일하려면 **끝에서** 탭의 **컴파일** 속성에서 **컴파일**을 클릭합니다. 컴파일 결과는 **끝에서** 탭의 **결과** 창에 표시됩니다.

## 함수 탭

**함수** 탭에서는 Java 프로그래밍 언어가 포함된 Java 변환에서 식을 호출하는 함수를 정의합니다.

예를 들어 입력 또는 출력 포트의 값을 조회하거나 Java 변환 변수의 값을 조회하는 식을 호출하는 함수를 정의할 수 있습니다.

함수를 정의하려면 **함수** 탭에 있는 **코드** 속성의 **Java 코드** 창에서 수동으로 함수를 정의하거나 새 함수를 클릭하여 **함수 정의** 대화 상자를 호출하고 간편하게 함수를 정의합니다.

코드를 컴파일하려면 **함수** 탭에 있는 **컴파일** 속성에서 **컴파일**을 클릭합니다. 컴파일 결과는 **함수** 탭의 **결과** 창에 표시됩니다.

## 전체 코드 탭

**전체 코드** 탭에서는 Java 변환의 전체 클래스 코드를 볼 수 있지만 편집할 수 없고 컴파일할 수는 있습니다.

**코드** 속성의 **Java 코드** 창에서 전체 클래스 코드를 볼 수 있습니다.

Java 변환의 전체 코드를 컴파일하려면 **전체 코드** 탭의 **컴파일** 속성에서 **컴파일**을 클릭합니다. 컴파일 결과는 **전체 코드** 탭의 **결과** 창에 표시됩니다.

## Java 변환을 통한 필터 최적화

데이터 통합 서비스는 활성 Java 변환에 필터 최적화를 적용할 수 있습니다. Java 변환을 정의할 때 Java 변환 **최적화 프로그램 인터페이스** 탭에서 필터 최적화를 위한 코드를 추가합니다.

### Java 변환의 초기 선택 최적화

Java 변환에 부작용이 없는 경우 초기 선택 최적화에 대해 능동 또는 수동 Java 변환을 활성화할 수 있습니다. 최적화 프로그램은 Java 변환을 통해 필터 논리를 전달하고 필요에 따라 필터 조건을 수정합니다.

초기 선택 최적화를 위한 코드 조각을 보려면 **최적화 프로그램 인터페이스** 탭의 탐색기에서 **PredicatePushOptimization**을 선택합니다.

#### allowPredicatePush

부울. 초기 선택을 활성화합니다. 초기 선택을 활성화하려면 함수를 **true** 결과와 메시지를 반환하도록 변경합니다. 기본값은 **false**이며 함수에서 최적화가 지원되지 않는다는 메시지가 반환됩니다.

```
public ResultAndMessage allowPredicatePush(boolean ignoreOrderOfOp) {
    // To Enable PredicatePushOptimization, this function should return true
    //return new ResultAndMessage(true, "");
    return new ResultAndMessage(false, "Predicate Push Optimization Is Not Supported");
}
```

#### canGenerateOutputFieldEvalError

부울. Java 변환이 0으로 나누기 오류 같은 출력 필드 오류를 반환할 수 있는지 여부를 나타냅니다. Java 변환이 출력 필드 오류를 생성하지 않는 경우 **false**를 반환하도록 함수를 변경하십시오. Java 변환이 실패 오류를 생성할 수 있는 경우에는 데이터 통합 서비스가 초기 선택 최적화를 사용할 수 없습니다.

```
public boolean canGenerateOutputFieldEvalError() {
    // If this Java transformation can never generate an output field evaluation error,
    // return false.
    return true;
}
```

#### getInputExpr

입력 필드의 입력 값 중에서 출력 필드를 구성하는 입력 값을 설명하는 Informatica 식을 반환합니다. 최적화 프로그램이 변환을 통해 필터 논리를 푸시하려면 어떤 입력 필드가 출력 필드를 구성하는지 알아야 합니다.

```
public InfaExpression getInputExpr(TransformationField field,
    TransformationDataInterface group) {
    // This should return an Informatica expression for output fields in terms of input fields
    // We will only push predicate that use fields for which input expressions are defined.
    // For example, if you have two input fields in0 and in1 and three output fields out0, out1, out2
    // out0 is the pass-through of in1, out2 is sum of in1 and in2, and out3 is unknown, the code should
    be:
    //if (field.getName().equals("out0"))
    //    return new InfaExpression("in0", instance);
}
```

```

        //else if (field.getName().equals("out1"))
        //    return new InfaExpression("in0 + in1", instance);
        //else if (field.getName().equals("out2"))
        //    return null;
        return null;
    }

```

예를 들어 매핑에 필터 식 "out0 > 8"이 포함되어 있다고 가정합니다. out0은 Java 변환에서 out0 출력 포트의 값입니다. out0의 값을 in0 입력 포트 + 5의 값으로 정의할 수 있습니다. 최적화 프로그램이 다음 식 "(in0 + 5) > 8"을 초기 선택 최적화를 사용하는 Java 변환으로 푸시할 수 있습니다. 출력 필드에 입력 필드 식이 없는 경우에는 NULL을 반환할 수 있습니다. 이 경우 최적화 프로그램이 필터 식을 입력 식이 없는 출력 필드에 푸시하지 않습니다.

다음과 같은 코드를 포함할 수 있습니다.

```

    if (field.getName().equals("out0"))
        return new InfaExpression("in0 + 5", instance);
    else if (field.getName().equals("out2"))
        return null;

```

## inputGroupsPushPredicateTo

필터 논리를 받을 수 있는 그룹 목록을 반환합니다. Java 변환에는 입력 그룹 하나가 있습니다. Java 변환의 경우 이 함수를 수정하지 마십시오.

```

public List<TransformationDataInterface> inputGroupsPushPredicateTo(
    List<TransformationField> fields) {
    // This functions returns a list of input data interfaces to push predicates to.
    // Since JavaTx only has one input data interface, you should not have to modify this function
    AbstractTransformation tx = instance.getTransformation();
    List<DataInterface> dis = tx.getDataInterfaces();
    List<TransformationDataInterface> inputDIs = new ArrayList<TransformationDataInterface>();
    for (DataInterface di : dis){
        TransformationDataInterface tdi = (TransformationDataInterface) di;
        if (tdi.isInput())
            inputDIs.add(tdi);
    }
    if(inputDIs.size() == 1)
        return inputDIs;
    else
        return null;
}

```

## Java 변환의 푸시인 최적화

Java 변환에 부작용이 없으며 최적화가 매핑 결과에 영향을 미치지 않는 경우 푸시인 최적화에 대한 능동 Java 변환을 활성화할 수 있습니다.

Java 변환에 대한 푸시인 최적화를 구성할 때 Java 변환이 최적화 프로그램에서 받는 필터 조건을 저장하는 방법을 정의합니다. 필터 조건을 검사하는 코드를 추가합니다. Java 변환은 필터 논리를 포함할 수 있으면 true 조건을 최적화 프로그램에 전달합니다. 최적화 프로그램은 최적화된 매핑에서 필터 변환을 제거합니다.

Java 변환을 구성할 때, 최적화 중에 필터 조건을 변환 메타데이터로 저장하는 코드를 작성합니다. 런타임에 필터 조건을 검색하고 필터 논리에 따라 행을 삭제하는 코드도 작성합니다.

Java 변환을 정의할 때 Java 변환 **최적화 프로그램 인터페이스** 탭에서 푸시인 최적화를 위한 코드를 추가합니다. 푸시인 최적화를 위한 코드 조각에 액세스하려면 변환 **최적화 프로그램 인터페이스** 탭의 탐색기에서 **FilterPushdownOptimization**을 선택합니다.

개발자 도구에 푸시인 최적화를 활성화하고 최적화 프로그램에서 필터 조건을 검색하는 코드 조각이 표시됩니다. 최적화를 활성화하고 필터 논리를 변환 메타데이터로 저장하도록 코드 조각을 업데이트합니다.

## isFilterSupported

푸시인 최적화를 활성화하려면 true를 반환합니다. 푸시인 최적화를 비활성화하려면 false를 반환합니다.



푸시인 최적화를 활성화하려면 **true**를 반환하도록 함수를 변경합니다.

```
public ResultAndMessage isFilterSupported() {
    // To enable filter push-into optimization this function should return true
    // return new ResultAndMessage(true, "");
    return new ResultAndMessage(false, "Filter push-into optimization is not supported");
}
```

## pushFilter

최적화 프로그램에서 필터 조건을 받습니다.

필터를 검사하고 변환에서 필터 논리를 사용할 수 있는지 여부를 확인하는 코드를 추가합니다. 변환은 필터를 포함할 수 있으면 다음 메서드를 사용하여 필터 조건을 변환 메타데이터로 저장합니다.

storeMetadata(문자열 키, 문자열 데이터)

키는 메타데이터에 대한 식별자입니다. 원하는 문자열을 키로 정의할 수 있습니다. 데이터는 런타임에 삭제할 행을 결정하기 위해 저장하려는 데이터입니다. 예를 들어 **Java** 변환이 최적화 프로그램에서 받는 필터 조건이 데이터일 수 있습니다.

```
public ResultAndMessage pushFilter(InfaExpression condition) {
    // Add code to absorb the filter
    // If filter is successfully absorbed return new ResultAndMessage(true, ""); and the optimizer
    // will remove the filter from the mapping
    // If the filter is not absorbed, return new ResultAndMessage(false, msg);
    return new ResultAndMessage(false, "Filter push-into optimization is not supported");
}
```

# Java 변환 작성

Developer tool에서 재사용 가능하거나 재사용 불가능 **Java** 변환을 작성할 수 있습니다.

## 재사용 가능 Java 변환 작성

재사용 가능 변환은 여러 매핑 내에 존재할 수 있습니다.

Developer tool에서 재사용 가능한 **Java** 변환을 작성합니다.

1. **개체 탐색기** 보기에서 프로젝트나 폴더를 선택합니다.
2. **파일 > 새로 만들기 > 변환**을 클릭합니다.  
    **새로 만들기** 대화 상자가 나타납니다.
3. **Java** 변환을 선택합니다.
4. **다음**을 클릭합니다.
5. 변환 이름을 입력합니다.
6. 활성 변환을 작성하려면 **활성으로 작성** 옵션을 선택합니다.
7. **마침**을 클릭합니다.

변환이 편집기에 표시됩니다.

8. **포트** 보기에서 **새로 만들기** 단추를 클릭하여 포트를 변환에 추가합니다.
9. 포트를 편집하여 이름, 데이터 유형 및 전체 자릿수를 설정합니다.  
    포트 이름을 **Java** 코드 조각의 변수로 사용합니다.
10. **Java** 보기에서 코드 항목 탭을 사용하여 변환의 **Java** 코드를 작성하고 컴파일합니다.

11. **Java** 보기에서 **함수** 탭을 사용하여 식을 호출하는 함수를 정의합니다.
12. 코드 항목 탭에서 **컴파일** 속성의 **결과** 창에 표시된 오류 메시지를 두 번 클릭하여 변환의 **Java** 코드에 대한 컴파일 오류를 찾고 수정합니다.
13. **고급** 보기에서 변환 속성을 편집합니다.

## 재사용 불가능 Java 변환 작성

재사용 불가능 변환은 단일 매핑에만 존재합니다.

Developer tool에서 재사용 불가능 **Java** 변환을 작성합니다.

1. 매핑 또는 맵셋에서 변환 색상표의 **Java** 변환을 편집기로 끌어옵니다.
2. **새 Java 변환** 대화 상자에서 변환 이름을 입력합니다.
3. 활성 변환을 작성하려면 **활성으로 작성** 옵션을 선택합니다.
4. **마침**을 클릭합니다.  
변환이 편집기에 표시됩니다.
5. **일반** 탭에서 변환 이름 및 설명을 편집합니다.
6. **포트** 탭에서 **새로 만들기** 단추를 클릭하여 포트를 변환에 추가합니다.
7. 포트를 편집하여 이름, 데이터 유형 및 전체 자릿수를 설정합니다.  
포트 이름을 **Java** 코드 조각의 변수로 사용합니다.
8. **Java** 보기에서 코드 항목 탭을 사용하여 변환의 **Java** 코드를 작성하고 컴파일합니다.
9. **Java** 보기에서 **함수** 탭을 사용하여 식을 호출하는 함수를 정의합니다.
10. 코드 항목 탭에서 **컴파일** 속성의 **결과** 창에 표시된 오류 메시지를 두 번 클릭하여 변환의 **Java** 코드에 대한 컴파일 오류를 찾고 수정합니다.
11. **고급** 보기에서 변환 속성을 편집합니다.

## Java 변환 컴파일

PowerCenter 클라이언트Developer tool은 **Java** 컴파일러를 사용하여 **Java** 코드를 컴파일하고 변환의 바이트 코드를 생성합니다.

**Java** 컴파일러는 **Java** 코드를 컴파일하고 **출력 창결과** 창(코드 항목 탭의 **컴파일** 속성에 위치)에 컴파일 결과를 표시합니다. **Java** 컴파일러는 PowerCenter 클라이언트Developer tool과 함께 **java/bin** 디렉터리에 설치됩니다.

**Java** 변환의 전체 코드를 컴파일하려면 **컴파일** 속성에서 **컴파일**을 클릭합니다(**Java 코드전체 코드** 탭 사용).

작성된 **Java** 변환에는 **Java** 변환의 기본 기능을 정의하는 **Java** 클래스가 포함됩니다. **Java** 클래스의 전체 코드에는 변환의 템플릿 클래스 코드와 사용자가 코드 항목 탭에서 정의하는 **Java** 코드가 포함됩니다.

**Java** 변환을 컴파일하면 PowerCenter 클라이언트Developer tool이 코드 항목 탭의 코드를 변환의 템플릿 클래스에 추가하여 변환의 전체 클래스 코드를 생성합니다. 그런 다음 PowerCenter 클라이언트Developer tool은 **Java** 컴파일러를 호출하여 전체 클래스 코드를 컴파일합니다. **Java** 컴파일러는 변환을 컴파일하여 변환의 바이트 코드를 생성합니다.

컴파일 결과는 **출력결과** 창에 표시됩니다. 컴파일 결과를 사용하여 **Java** 코드 오류를 식별하고 찾을 수 있습니다.

**참고:** **Java** 변환에서 **확인**을 클릭하여 변환을 컴파일할 수도 있습니다.

# Java 변환 문제 해결

코드 항목 탭에서 **컴파일** 속성의 **결과** 창에서 **Java** 코드 오류를 찾고 수정할 수 있습니다.

**Java** 변환의 오류는 **Java** 변환 클래스의 전체 코드 또는 코드 항목 탭의 코드 오류로 인해 발생할 수 있습니다.

**Java** 변환 문제를 해결하려면 다음 고급 단계를 완료합니다.

1. 변환에 대한 전체 클래스 코드 또는 **Java** 조각 코드에서 오류의 소스를 찾습니다.
2. 오류 유형을 식별합니다. **결과** 창의 컴파일 결과 및 오류 위치를 사용하여 오류 유형을 식별합니다.
3. 코드 항목 탭에서 **Java** 코드를 수정합니다.
4. 변환을 다시 컴파일합니다.

## 컴파일 오류의 소스 찾기

컴파일 오류의 소스를 찾으려면 코드 항목 탭 또는 **전체 코드** 탭에서 **컴파일** 속성의 **결과** 창에 표시되는 컴파일 결과를 사용합니다.

**결과** 창의 오류 메시지를 두 번 클릭하면 오류를 야기한 소스 코드가 코드 항목 탭 또는 **전체 코드** 탭의 **Java 코드** 창에 강조 표시됩니다.

**전체 코드** 탭에서 오류를 찾을 수 있지만 **Java** 코드는 **전체 코드** 탭에서 편집할 수 없습니다. **전체 코드** 탭에서 찾은 오류를 수정하려면 해당하는 코드 항목 탭에서 코드를 수정해야 합니다. 변환에 대한 전체 클래스 코드에 사용자 코드를 추가하여 발생한 오류를 보려면 **전체 코드** 탭을 사용해야 할 수 있습니다.

## 코드 항목 탭 또는 전체 코드 탭에서 오류 찾기

코드 항목 탭 또는 **전체 코드** 탭에서 컴파일 오류를 찾을 수 있습니다.

1. 코드 항목 탭 또는 **전체 코드** 탭에서 **컴파일** 속성의 **결과** 창에 표시된 오류 메시지를 마우스 오른쪽 단추로 클릭합니다.
2. **표시 위치 > 조각** 또는 **표시 위치 > 전체 코드** 탭을 클릭합니다.

Developer tool이 오류의 소스를 선택한 탭에 강조 표시합니다.

**참고:** **전체 코드** 탭에서는 오류를 볼 수 있지만 수정할 수는 없습니다. 오류를 수정하려면 해당하는 코드 항목 탭으로 이동해야 합니다.

## 컴파일 오류의 소스 식별

사용자 코드에 오류가 있을 경우 컴파일 오류가 발생할 수 있습니다.

사용자 코드에 오류가 있을 경우 클래스에 대한 비사용자 코드에도 오류가 발생할 수 있습니다. 컴파일 오류는 **Java** 변환에 대한 사용자 및 비사용자 코드에서 발생합니다.

## 사용자 코드 오류

코드 입력 탭의 사용자 코드에서 오류가 발생할 수 있습니다. 사용자 코드 오류에는 표준 **Java** 구문 및 언어 오류가 포함됩니다.

PowerCenter 클라이언트 Developer tool이 코드 입력 탭에서 전체 클래스 코드에 사용자 코드를 추가할 때도 사용자 코드 오류가 발생할 수 있습니다.

Java 변환에 이름이 `int1`인 입력 포트와 정수 데이터 유형이 포함되는 경우를 예로 들어 보겠습니다. 클래스의 전체 코드는 다음 코드를 사용하여 입력 포트 변수를 선언합니다.

```
int int1;
```

그러나 **입력 시 행입력 시** 탭에서 같은 변수를 사용하는 경우에는 Java 컴파일러가 변수 다시 선언에 대해 오류를 표시합니다. 이 오류를 해결하려면 **입력 시 행입력 시** 탭에서 변수 이름을 바꿉니다.

## 비사용자 코드 오류

코드 항목 탭의 사용자 코드에서 사용자가 아닌 코드로 인해 오류가 발생할 수 있습니다.

예를 들어 Java 변환에는 정수 데이터 유형을 사용하는 입력 포트 및 출력 포트, `int1` 및 `out1`이 있습니다. 다음 코드를 **입력 행에서입력 시** 코드 항목 탭에 쓰면 입력 포트 `int1`에 대해 이자를 계산하고 출력 포트 `out1`에 할당합니다.

```
int interest;
interest = CallInterest(int1); // calculate interest
out1 = int1 + interest;
}
```

변환을 컴파일하면 PowerCenter ClientDeveloper tool이 **입력 행에서입력 시** 코드 항목 탭의 코드를 변환을 위해 전체 클래스 코드에 추가합니다. Java 컴파일러가 Java 코드를 컴파일할 때 일치하지 않는 괄호가 있으면 전체 클래스 코드의 메시드가 중간에 종료되고 Java 컴파일러에서 오류가 발생합니다.

## 구조체 데이터로 변환의 예

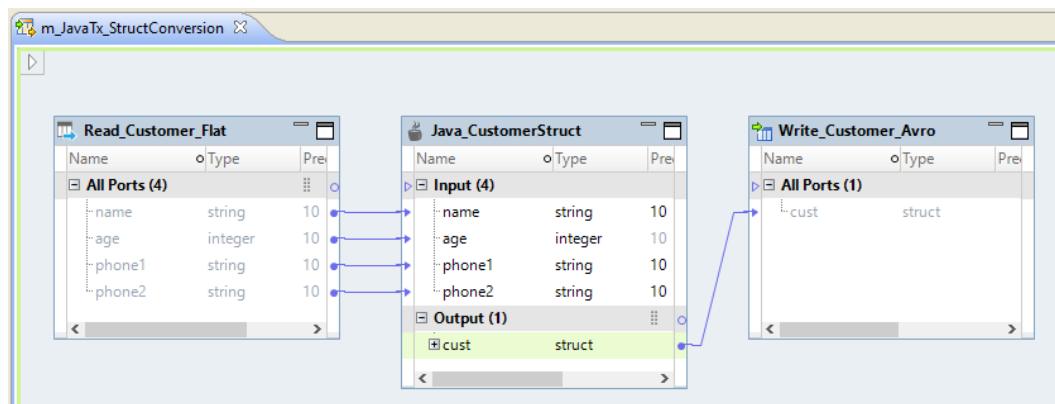
플랫 파일 형식의 대용량 고객 데이터를 구조체 데이터로 변환하고 Avro 파일에 써야 한다고 가정하겠습니다. 입력 파일에는 이름, 연령, 및 전화 번호 같은 고객 세부 정보가 포함됩니다. 입력 파일에서 고객 이름이 null인 경우 이 고객의 세부 정보를 출력 파일에 추가할 필요가 없습니다.

이 경우 변환 기능을 정의하는 Java 변환이 포함된 매핑을 개발할 수 있습니다. Hadoop 환경에서는 Spark 엔진에서 매핑을 실행하여 데이터를 변환하고 구조체 데이터를 Avro 파일에 씁니다.

매핑을 생성하고 다음과 같은 변환을 구성합니다.

- 플랫 파일 소스에서 고객 정보를 읽는 읽기 변환
- 플랫 데이터를 구조체 데이터로 변환하고 일치하지 않는 데이터를 제거하는 활성 Java 변환
- 구조체 데이터를 Avro 파일에 쓰는 쓰기 변환

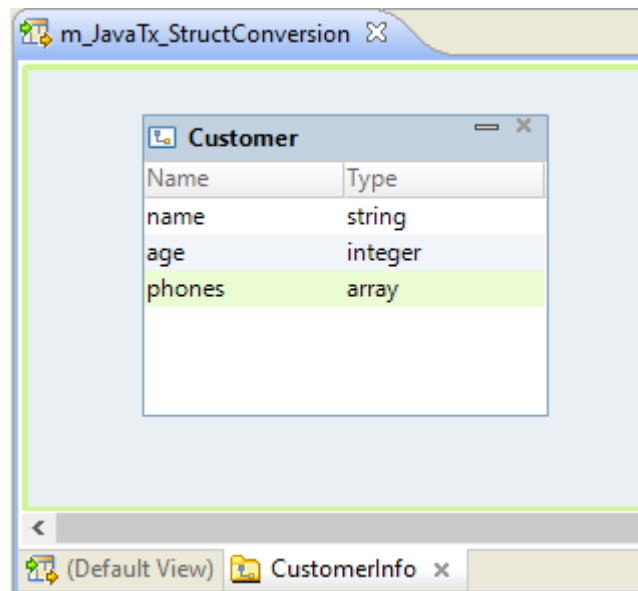
다음 이미지는 읽기 변환, Java 변환 및 쓰기 변환이 포함된 매핑을 보여 줍니다.



매핑 편집기의 유형 정의 라이브러리 탭에서 복합 데이터 유형 정의인 **Customer**를 생성합니다. 이 복합 데이터 유형 정의는 구조체 데이터의 스키마를 나타냅니다. 유형 정의 라이브러리의 이름을 **CustomerInfo**로 바꿉니다. 복합 데이터 유형 정의에 다음과 같은 요소를 추가합니다.

- 문자열 유형의 이름
- 정수 유형의 연령
- 문자열 요소가 포함된 배열 유형의 전화 번호

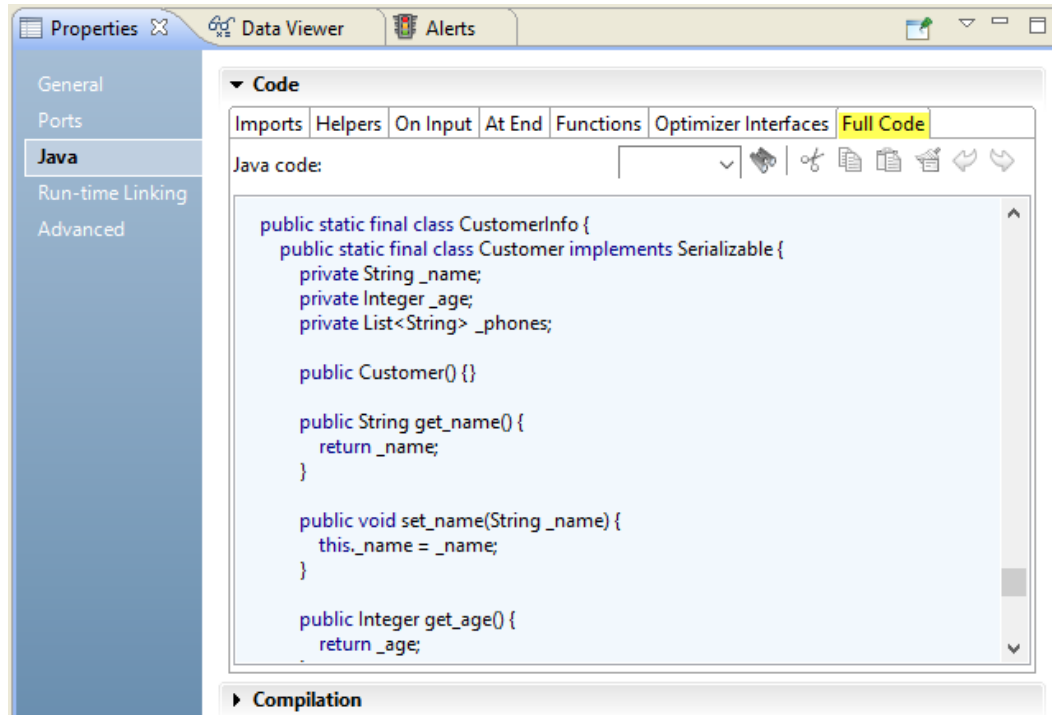
다음 이미지는 유형 정의 라이브러리의 복합 데이터 유형 정의를 보여 줍니다.



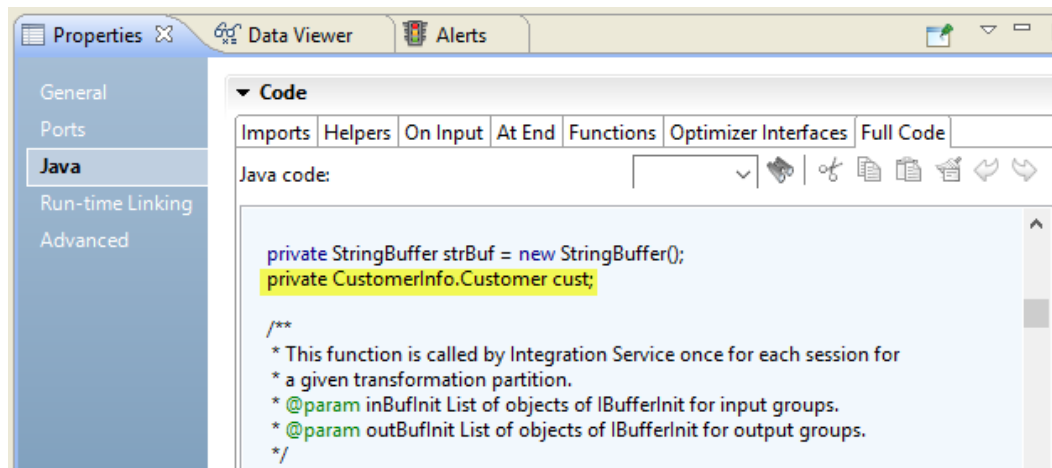
Java 변환에서 구조체 출력 포트를 추가하고 생성된 복합 데이터 유형 정의를 참조하도록 포트의 유형 구성을 지정합니다. 이 Java 변환은 멤버 필드를 읽고 설정하는 **setter** 및 **getter**가 포함된 **Customer** 클래스를 생성합니다. 이 클래스에는 다음과 같은 멤버 필드가 포함됩니다.

- `_name`
- `_age`
- `_phones`

다음 이미지는 **Java** 보기의 전체 코드 탭에 나와 있는 구조체 포트에 대해 생성된 클래스를 보여 줍니다.

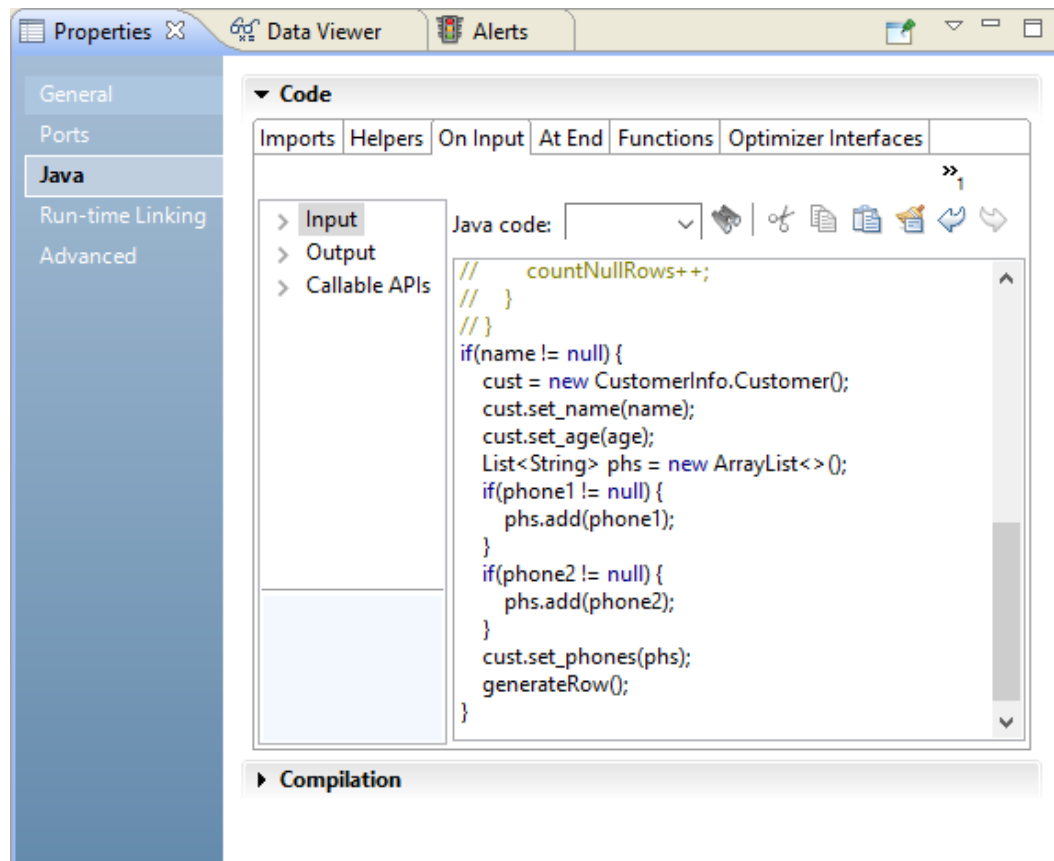


구조체 포트에 대한 Java 데이터 유형은 유형 정의 라이브러리의 이름과 복합 데이터 유형 정의를 사용합니다. 다음 이미지에는 생성된 코드의 `cust` 필드에 대한 `CustomerInfo.Customer` Java 데이터 유형 이름이 나와 있습니다.



Java 변환의 **Java** 보기에서 변환에 필요한 타사, 기본 제공 또는 사용자 지정 Java 패키지를 가져옵니다. 플랫폼 데이터를 구조체 데이터로 변환하고 고객 이름이 null인 경우 고객 행을 제거하는 Java 코드를 작성하고 컴파일합니다.

다음 이미지는 **입력 시** 탭의 코드를 보여 줍니다.



매핑의 유효성을 검사하고 Spark 엔진에서 매핑을 실행하여 변환된 데이터를 Avro 파일 출력에 기록합니다.

## Java 변환 - 비원시 환경

비원시 환경에서 Java 변환의 처리는 변환을 실행하는 엔진에 따라 다릅니다.

다음과 같은 비원시 런타임 엔진에 대한 지원을 고려하십시오.

- Blaze 엔진. 제한적으로 지원됩니다.
- Spark 엔진. 제한적으로 지원됩니다.
- Databricks Spark 엔진. 지원되지 않습니다.

## Java 변환 - Blaze 엔진

Java 변환에서 외부 .jar 파일을 사용하려면 다음 단계를 수행합니다.

1. 데이터 통합 서비스 시스템에서 다음 위치의 Informatica 설치 디렉터리에 외부 .jar 파일을 복사합니다.  
<Informatica 설치 디렉터리>/services/shared/jars. 그런 다음 데이터 통합 서비스를 재사용합니다.

2. Java 변환이 포함된 매핑을 개발하고 실행하는 Developer tool을 호스팅하는 시스템에서 다음을 수행합니다.
  - a. 로컬 시스템의 디렉터리에 외부 .jar 파일을 복사합니다.
  - b. 로컬 .jar 파일을 가리키는 가져오기 문을 포함하도록 Java 변환을 편집합니다.
  - c. Java 변환의 클래스 경로를 업데이트합니다.
  - d. 변환을 컴파일합니다.

## Java 변환 - Spark 엔진

복합 데이터 유형을 사용하여 계층 데이터를 처리할 수 있습니다.

Spark 엔진의 일부 처리 규칙은 데이터 통합 서비스의 처리 규칙과 다릅니다.

### 일반 제한

Spark 엔진에서 Java 변환은 다음과 같은 제한과 함께 지원됩니다.

- 변환 논리를 Hadoop으로 푸시하는 경우 변환의 Java 코드는 표준 출력에 출력을 쓸 수 없습니다. Java 코드는 로그 파일에 나타나는 표준 오류에 출력을 쓸 수 있습니다.
- 날짜/시간 값의 경우 Spark 엔진은 최대 마이크로초의 전체 자릿수를 지원합니다. 날짜/시간 값에 나노초가 포함되는 경우 뒤의 자릿수가 잘립니다.

### 분할

Java 변환에서 분할을 사용하는 경우 다음과 같은 제한이 적용됩니다.

- 분할 가능한 속성은 Java 변환에서 활성화되어야 합니다. 단일 파티션에서 변환을 실행할 수 없습니다.
- 변환 범위 속성에는 다음과 같은 제한이 적용됩니다.
  - 트랜잭션을 변환 범위의 값으로 사용할 수 없습니다.
  - 파티션 키에 대한 입력 포트를 활성화하는 경우 변환 범위를 모든 입력으로 설정해야 합니다.
  - 변환 범위가 행인 경우 상태 비저장을 활성화해야 합니다.

### 매핑 유효성 검사

다음과 같은 상황에서는 매핑 유효성 검사가 실패합니다.

- Java 변환 내의 식에서 연결되지 않은 조회 변환을 참조합니다.
- 복합 데이터 유형의 포트를 파티션 또는 정렬 키로 선택합니다.
- 날짜/시간에서 나노초 처리를 활성화하고, Java 변환에 날짜/시간 유형의 요소가 포함된 복합 데이터 유형의 포트가 있습니다. 예를 들어 날짜/시간에서 나노초 처리를 활성화하는 경우 `array<data/time>` 유형의 포트는 올바르지 않습니다.
- 많은 전체 자릿수를 활성화하는 경우 Java 변환에 10진수 포트 또는 10진수 데이터 유형의 요소가 있는 복합 포트를 사용하는 식이 있고 포트에 연산자가 사용되면 유효성 검사 오류가 발생합니다.  
예를 들어 변환에 `decimal_port`라는 10진수 포트가 있는 경우 `decimal_port + 1`이라는 식을 사용하면 유효성 검사 오류가 발생합니다.



다음과 같은 상황에서는 매핑이 실패합니다.

- **Java** 변환에 10진수 포트 또는 10진수 데이터 유형의 요소가 있는 복합 포트가 포함되고 **Java** 변환과 매핑에 사용되는 전체 자릿수 모드가 다릅니다.

매핑에서 많은 전체 자릿수가 활성화된 경우라도 매핑에 10진수 데이터 유형의 요소가 있는 복합 포트가 포함된 경우 매핑이 스트리밍 매핑인 경우 등의 일부 상황에서는 매핑이 작은 전체 자릿수 모드로 데이터를 처리합니다. **Java** 변환과 매핑에 대해 많은 전체 자릿수가 활성화되어 있지만 매핑이 작은 전체 자릿수 모드로 데이터를 처리하면 매핑이 실패합니다.

### 외부 .jar 파일 사용

**Java** 변환에서 외부 .jar 파일을 사용하려면 다음 단계를 수행합니다.

1. 데이터 통합 서비스 시스템에서 다음 위치의 **Informatica** 설치 디렉터리에 외부 .jar 파일을 복사합니다.  
<Informatica 설치 디렉터리>/services/shared/jars
2. 데이터 통합 서비스를 재사용합니다.
3. **Java** 변환이 포함된 매핑을 개발하고 실행하는 **Developer tool**을 호스팅하는 시스템에서 다음을 수행합니다.
  - a. 로컬 시스템의 디렉터리에 외부 .jar 파일을 복사합니다.
  - b. 로컬 .jar 파일을 가리키는 가져오기 문을 포함하도록 **Java** 변환을 편집합니다.
  - c. **Java** 변환의 클래스 경로를 업데이트합니다.
  - d. 변환을 컴파일합니다.

## 제 20 장

# Java 변환 API 참조

이 장에 포함된 항목:

- [Java 변환 API 메서드 개요, 310](#)
- [커밋, 311](#)
- [defineJExpression, 312](#)
- [failSession, 312](#)
- [generateRow, 313](#)
- [getInRowType, 314](#)
- [getMetadata, 314](#)
- [incrementErrorCount, 315](#)
- [invokeJExpression, 315](#)
- [isNull, 316](#)
- [logError, 317](#)
- [logInfo, 318](#)
- [resetNotification, 318](#)
- [롤백, 319](#)
- [setNull, 319](#)
- [setOutRowType, 320](#)
- [storeMetadata, 321](#)

## Java 변환 API 메서드 개요

Java 변환의 **Java 코드** 탭편집기의 **Java** 보기에 있는 코드 항목 탭에서 Java 코드에 API 메서드를 추가하여 변환 동작을 정의할 수 있습니다.

API 메서드를 코드에 추가하려면 코드 항목 탭의 탐색기에서 **호출 가능 API** 목록을 확장한 다음, 코드에 추가할 메서드 이름을 두 번 클릭합니다.

또는 메서드를 탐색기에서 **Java** 코드 조각으로 끌거나 **Java** 코드 조각에 API 메서드를 수동으로 입력할 수 있습니다.

Java 변환에서 다음 API 메서드를 Java 코드에 추가할 수 있습니다.

### 커밋

트랜잭션을 생성합니다.

**defineJExpression**

Java 식을 정의합니다.

**failSession**

오류 메시지와 함께 예외가 발생하고 세션매핑이 실패합니다.

**generateRow**

활성 Java 변환에 대한 출력 행을 생성합니다.

**getInRowType**

변환에서 현재 행의 입력 유형을 반환합니다.

**incrementErrorCount**

세션매핑에 대한 오류 수를 증가합니다.

**invokeJExpression**

**defineJExpression** 메서드를 사용하여 정의한 Java 식을 호출합니다.

**isNull**

입력 열에 Null 값이 있는지 확인합니다.

**logError**

오류 메시지를 세션 로그에 기록합니다.

**logInfo**

정보 메시지를 세션 로그에 기록합니다.

**resetNotification**

데이터 통합 서비스 시스템이 다시 시작 모드에서 실행되는 경우 매핑 실행 후 Java 코드에서 사용되는 변수를 다시 설정합니다.

**롤백**

롤백 트랜잭션을 생성합니다.

**setNull**

활성 또는 수동 Java 변환의 출력 열 값을 Null로 설정합니다.

**setOutRowType**

출력 행에 대한 업데이트 전략을 설정합니다. 삽입, 업데이트 또는 삭제 플래그를 행에 지정할 수 있습니다.

## 커밋

트랜잭션을 생성합니다.

패키지 가져오기 탭이나 Java 식 코드 항목 탭을 제외한 모든 탭에서 커밋을 사용하십시오. 트랜잭션을 생성하도록 구성된 활성 변환에서만 커밋을 사용할 수 있습니다. 트랜잭션을 생성하도록 구성되지 않은 활성 변환에서 커밋을 사용할 경우 통합 서비스에서 오류가 발생하고 세션이 실패합니다.

다음 구문을 사용합니다.

```
commit();
```

다음 Java 코드를 사용하여 Java 변환에서 처리한 100개 행마다 트랜잭션을 생성한 다음 rowsProcessed 카운터를 0으로 설정합니다.

```
if (rowsProcessed==100) {
    commit();
    rowsProcessed=0;
}
```

## defineJExpression

식 문자열 및 입력 매개 변수를 포함하여 식을 정의합니다. `defineJExpression` 메서드의 인수에는 입력 매개 변수와 식 구문을 정의하는 문자열 값을 포함하는 `JExprParamMetadata` 개체의 배열이 포함됩니다.

다음 구문을 사용합니다.

```
defineJExpression(
    String expression,
    Object[] paramMetadataArray
);
```

다음 표에는 매개 변수가 설명되어 있습니다.

매개 변수	유형	데이터 유형	설명
식	입력	문자열	식을 나타내는 문자열입니다.
paramMetadataArray	입력	Object[]	식의 입력 매개 변수를 포함하는 JExprParamMetadata 개체의 배열입니다.

`defineJExpression` 메서드는 가져오기 및 함수 탭을 제외한 모든 코드 항목 탭의 Java 코드에 추가할 수 있습니다.

`defineJExpression` 메서드를 사용하려면 식의 입력 매개 변수를 나타내는 `JExprParamMetadata` 개체의 배열을 인스턴스화해야 합니다. 매개 변수에 대한 메타데이터 값을 설정하고 배열을 `defineJExpression` 메서드에 매개 변수로 전달하십시오.

예를 들어 다음 Java 코드는 두 문자열의 값을 조화하는 식을 작성합니다.

```
JExprParamMetadata params[] = new JExprParamMetadata[2];
params[0] = new JExprParamMetadata(EDatatype.STRING, 20, 0);
params[1] = new JExprParamMetadata(EDatatype.STRING, 20, 0);
defineJExpression(":lkp.mylookup(x1,x2)",params);
```

**참고:** 식에 전달하는 매개 변수에는 문자 `x`로 시작되는 연속 번호를 지정해야 합니다. 예를 들어 매개 변수 3개를 식에 전달하는 경우 매개 변수의 이름을 `x1`, `x2` 및 `x3`으로 지정합니다.

## failSession

오류 메시지와 함께 예외가 발생하고 세션매핑이 실패합니다.

다음 구문을 사용합니다.

```
failSession(String errorMessage);
```

다음 표에는 매개 변수가 설명되어 있습니다.

매개 변수	매개 변수 유형	데이터 유형	설명
errorMessage	입력	문자열	오류 메시지 문자열입니다.

세션매핑을 종료하려면 `failSession` 메서드를 사용하십시오. 코드 항목 탭의 `try/catch` 블록에서 `failSession` 메서드를 사용하지 마십시오.

**가져오기패키지 가져오기** 및 **함수Java** 식 탭을 제외한 코드 항목 탭에서 Java 코드에 `failSession` 메서드를 추가할 수 있습니다.

다음 Java 코드에서 `input1` 입력 포트를 Null 값에 대해 테스트하고 Null일 경우 세션매핑을 실패하는 방법을 보여 줍니다.

```
if(isNull("input1")) {  
    failSession("Cannot process a null value for port input1.");  
}
```

## generateRow

활성 Java 변환에 대한 출력 행을 생성합니다.

다음 구문을 사용합니다.

```
generateRow();
```

`generateRow` 메서드를 호출할 경우 Java 변환에서 출력 포트 변수의 현재 값을 사용하여 출력 행을 생성합니다. 입력 행에 해당하는 여러 행을 생성할 경우 각 입력 행마다 두 번 이상 `generateRow` 메서드를 호출할 수 있습니다. 활성 Java 변환에서 `generateRow` 메서드를 사용하지 않을 경우 변환에서 출력 행을 생성하지 않습니다.

**가져오기패키지 가져오기** 및 **함수Java** 식 탭을 제외한 모든 코드 항목 탭에서 `generateRow` 메서드를 Java 코드에 추가할 수 있습니다.

활성 변환에서만 `generateRow` 메서드를 호출할 수 있습니다. 수동 변환에서 `generateRow` 메서드를 호출할 경우 세션데이터 통합 서비스에서 오류를 생성합니다.

다음 Java 코드를 사용하여 출력 행을 1개 생성하고, 출력 포트의 값을 수정한 다음, 다른 출력 행을 생성하십시오.

```
// Generate multiple rows.  
if(!isNull("input1") && !isNull("input2"))  
{  
    output1 = input1 + input2;  
    output2 = input1 - input2;  
}  
generateRow();  
// Generate another row with modified values.  
output1 = output1 * 2;  
output2 = output2 * 2;  
generateRow();
```

## getInRowType

변환에서 현재 행의 입력 유형을 반환합니다. 메서드에서 삽입, 업데이트, 삭제 또는 거부 값을 반환합니다.

다음 구문을 사용합니다.

```
rowType getInRowType();
```

다음 표에는 매개 변수가 설명되어 있습니다.

매개 변수	매개 변수 유형	데이터 유형	설명
rowType	출력	문자열	다음 값 중 하나인 업데이트 전략 유형을 반환합니다. <ul style="list-style-type: none"><li>- DELETE</li><li>- INSERT</li><li>- REJECT</li><li>- UPDATE</li></ul>

**입력 시 입력 행에서** 코드 항목 탭에서 `getInRowType` 메서드를 Java 코드에 추가할 수 있습니다.

업데이트 전략을 설정하도록 구성된 활성 변환에서 `getInRowType` 메서드를 사용할 수 있습니다. 업데이트 전략을 설정하도록 구성되지 않은 활성 변환에서 이 메서드를 호출할 경우 세션데이터 통합 서비스에서 오류를 생성합니다.

다음 Java 코드를 사용하여 다음 작업을 수행하십시오.

- 현재 입력 행 유형을 출력 행에 전달합니다.
- `input1` 입력 포트의 값이 100보다 큰 경우 출력 행 유형을 DELETE로 설정합니다.

```
// Set the value of the output port.  
output1 = input1;  
// Get and set the row type.  
String rowType = getInRowType();  
setOutRowType(rowType);  
// Set row type to DELETE if the output port value is > 100.  
if(input1 > 100  
    setOutRowType(DELETE);
```

## getMetadata

런타임 시 Java 변환 메타데이터를 검색합니다. `getMetadata` 메서드가 `storeMetadata` 메서드와 함께 저장한 메타데이터를 검색합니다(예: 최적화 프로그램이 `pushFilter` 함수에서 Java 변환에 전달하는 필터 조건).

다음 구문을 사용합니다.

```
getMetadata (String key);
```

다음 표에는 매개 변수가 설명되어 있습니다.

매개 변수	매개 변수 유형	데이터 유형	설명
키	입력	문자열	메타데이터를 식별합니다. <code>getMetadata</code> 메서드는 키를 사용하여 검색할 메타데이터를 결정합니다.

다음 코드 항목 탭에서 `getMetadata` 메서드를 Java 코드에 추가할 수 있습니다.

- 도우미
- 입력 시
- 끝에서
- 최적화 프로그램 인터페이스
- 함수

푸시인 최적화에 대한 필터 조건을 검색하도록 `getMetadata` 메서드를 구성할 수 있습니다. `getMetadata` 메서드가 최적화 프로그램에서 사용자가 저장한 각 필터 조건을 검색할 수 있습니다.

```
// Retrieve a filter condition
String mydata = getMetadata ("FilterKey");
```

## incrementErrorCount

세션에 대한 오류 수를 증가합니다. 오류 수가 세션에 대한 오류 임계값에 도달한 경우 세션매핑이 실패합니다.

다음 구문을 사용합니다.

```
incrementErrorCount(int nErrors);
```

다음 표에는 매개 변수가 설명되어 있습니다.

매개 변수	매개 변수 유형	데이터 유형	설명
nErrors	입력	정수	세션에 대한 오류 수가 증가하는 숫자입니다.

**가져오기 패키지 가져오기** 및 **함수 Java** 식 탭을 제외한 코드 항목 탭에서 Java 코드에 `incrementErrorCount` 메서드를 추가할 수 있습니다.

다음 Java 코드는 변환의 입력 포트에 Null 값이 있을 경우 오류 수를 증가하는 방법을 보여 줍니다.

```
// Check if input employee id and name is null.
if (isNull ("EMP_ID_INP") || isNull ("EMP_NAME_INP"))
{
    incrementErrorCount(1);
    // if input employee id and/or name is null, don't generate a output row for this input row
    generateRow = false;
}
```

## invokeJExpression

식을 호출하고 식의 값을 반환합니다.

다음 구문을 사용합니다.

```
(datatype)invokeJExpression(
    String expression,
    Object[] paramMetadataArray);
```

`invokeJExpression` 메서드의 입력 매개 변수는 식을 나타내는 문자열 값과 식 입력 매개 변수를 포함하는 개체의 배열입니다.

다음 표에는 매개 변수가 설명되어 있습니다.

매개 변수	매개 변수 유형	데이터 유형	설명
식	입력	문자열	식을 나타내는 문자열입니다.
<code>paramMetadataArray</code>	입력	<code>Object[]</code>	식의 입력 매개 변수를 포함하는 개체의 배열입니다.

**가져오기 및 함수패키지 가져오기 및 Java 식** 탭을 제외한 모든 코드 항목 탭의 **Java** 코드에 `invokeJExpression` 메서드를 추가할 수 있습니다.

다음 규칙 및 지침에 따라 `invokeJExpression` 메서드를 사용하십시오.

- **반환 데이터 유형.** `invokeJExpression` 메서드의 반환 데이터 유형은 개체입니다. 함수의 반환 값을 적절한 데이터 유형으로 캐스팅해야 합니다.  
정수, 배열, 문자열 및 `byte[]` 데이터 유형으로 값을 반환할 수 있습니다.
- **행 유형.** `invokeJExpression` 메서드의 반환 값에 대한 행 유형은 `INSERT`입니다.  
반환 값에 서로 다른 행 유형을 사용하려면 고급 인터페이스를 사용합니다.
- **Null 값.** Null 값을 매개 변수로 전달하거나 `invokeJExpression` 메서드의 반환 값이 `NULL`인 경우 이 값은 Null 표시기로 처리됩니다.  
예를 들어 식의 반환 값이 `NULL`이고 반환 데이터 유형이 문자열인 경우 Null 값과 함께 문자열이 반환됩니다.
- **날짜 데이터 유형.** 날짜 데이터 유형의 입력 매개 변수는 문자열 데이터 유형으로 변환해야 합니다.  
식의 문자열을 날짜 데이터 유형으로 사용하려면 `to_date()` 함수를 사용하여 문자열을 날짜 데이터 유형으로 변환해야 합니다.  
또한 날짜 데이터 유형을 반환하는 모든 식의 반환 유형을 문자열 데이터 유형으로 캐스팅해야 합니다.

다음 예제에서는 "John" 및 "Smith" 문자열을 연결하여 "John Smith" 문자열을 반환합니다.

```
(String)invokeJExpression("concat(x1,x2)", new Object [] { "John ", "Smith" });
```

**참고:** 식에 전달하는 매개 변수에는 문자 `x`로 시작되는 연속 번호를 지정해야 합니다. 예를 들어 매개 변수 3개를 식에 전달하는 경우 매개 변수의 이름을 `x1`, `x2` 및 `x3`으로 지정합니다.

## isNull

입력 열 값에 Null 값이 있는지 확인합니다.

다음 구문을 사용합니다.

```
Boolean isNull(String satrColName);
```



다음 표에는 매개 변수가 설명되어 있습니다.

매개 변수	매개 변수 유형	데이터 유형	설명
strColName	입력	문자열	입력 열의 이름입니다.

**입력 시 입력 행에서** 코드 항목 탭에서 `isNull` 메서드를 Java 코드에 추가할 수 있습니다.

다음 Java 코드에서는 `SALARY` 입력 열 값을 `totalSalaries` 인스턴스 변수에 추가하기 전에 해당 값이 `Null`인지 확인하는 방법을 보여 줍니다.

```
// if value of SALARY is not null
if (!isNull("SALARY")) {
    // add to totalSalaries
    TOTAL_SALARIES += SALARY;
}
```

또는 다음 Java 코드를 사용하여 동일한 결과를 얻으십시오.

```
// if value of SALARY is not null
String strColName = "SALARY";
if (!isNull(strColName)) {
    // add to totalSalaries
    TOTAL_SALARIES += SALARY;
}
```

## logError

오류 메시지를 세션 로그에 기록합니다.

다음 구문을 사용합니다.

```
logError(String msg);
```

다음 표에는 매개 변수가 설명되어 있습니다.

매개 변수	매개 변수 유형	데이터 유형	설명
msg	입력	문자열	오류 메시지 문자열입니다.

**가져오기 패키지 가져오기** 및 **함수 Java** 식 탭을 제외한 모든 코드 항목 탭에서 `logError` 메서드를 Java 코드에 추가할 수 있습니다.

다음 Java 코드에서는 입력 포트가 `Null`일 경우 오류를 로그하는 방법을 보여 줍니다.

```
// check BASE_SALARY
if (isNull("BASE_SALARY")) {
    logError("Cannot process a null salary field.");
}
```

코드가 실행되면 다음 메시지가 세션 로그에 표시됩니다.

```
[JTX_1013] [ERROR] Cannot process a null salary field.
```

## logInfo

정보 메시지를 세션 로그에 기록합니다.

다음 구문을 사용합니다.

```
logInfo(String msg);
```

다음 표에는 매개 변수가 설명되어 있습니다.

매개 변수	매개 변수 유형	데이터 유형	설명
msg	입력	문자열	정보 메시지 문자열입니다.

**가져오기 패키지 가져오기** 및 **함수 Java** 식 탭을 제외한 코드 항목 탭에서 Java 코드에 logInfo 메서드를 추가할 수 있습니다.

다음 Java 코드에서는 Java 변환이 1000개 행의 메시지 임계값을 처리한 후 세션 로그에 메시지를 기록하는 방법을 보여 줍니다.

```
if (numRowsProcessed == messageThreshold) {  
    logInfo("Processed " + messageThreshold + " rows.");  
}
```

## resetNotification

데이터 통합 서비스 시스템이 다시 시작 모드에서 실행되는 경우 매핑 실행 후 Java 코드에서 사용되는 변수를 다시 설정합니다.

다시 시작 모드에서 데이터 통합 서비스는 초기화가 취소되지는 않지만 요청 후에 재설정되므로 데이터 통합 서비스가 다음 요청을 처리할 수 있습니다.

Java 변환의 경우 매핑 실행 후 resetNotification 메서드를 사용하여 Java 코드에서 변수를 재설정할 수 있습니다.

다음 구문을 사용합니다.

```
public int resetNotification(IGroup group) {  
    return EStatus.value;  
}
```

다음 표에는 매개 변수가 설명되어 있습니다.

매개 변수	매개 변수 유형	데이터 유형	설명
int	출력	EStatus.값	값을 반환합니다. 여기서 값은 다음 값 중 하나입니다. <ul style="list-style-type: none"><li>- SUCCESS. 성공</li><li>- FAILURE. 실패</li><li>- NOIMPL. 구현되지 않음</li></ul>
그룹	입력	IGroup	입력 그룹입니다.

resetNotification 메서드를 **도우미** 탭의 코드 항목 탭에서 Java 코드에 추가할 수 있습니다.

resetNotification 메서드는 호출 가능 API 목록에 나타나지 않습니다.

예를 들어 Java 코드가 `out5_static`이라는 정적 변수를 선언하고 1로 초기화한다고 가정하면 다음 Java 코드는 다음 번 매핑을 실행한 후 `out5_static` 변수를 1로 재설정합니다.

```
public int resetNotification(IGroup group) {
    out5_static=1;
    return EStatus.SUCCESS;
}
```

이 메서드는 필수가 아닙니다. 하지만 데이터 통합 서비스가 다시 시작 모드에서 실행되고 매핑에 `resetNotification` 메서드를 구현하지 않은 Java 변환이 포함된 경우 `JSDK_42075` 경고 메시지가 로그에 표시됩니다.

## 롤백

롤백 트랜잭션을 생성합니다.

패키지 가져오기 또는 Java 식 코드 항목 탭을 제외한 탭에서 롤백을 사용합니다. 트랜잭션을 생성하도록 구성된 활성 변환에서만 롤백을 사용할 수 있습니다. 트랜잭션을 생성하도록 구성되지 않은 활성 변환에서 롤백을 사용하는 경우 통합 서비스가 오류를 생성하고 세션을 실행하지 못합니다.

다음 구문을 사용합니다.

```
rollback();
```

다음 코드를 사용하여 입력 행에 잘못된 조건이 있으면 롤백 트랜잭션을 생성하고 세션을 실행시키지 않거나 처리할 행 수가 100이면 트랜잭션을 생성합니다.

```
// If row is not legal, rollback and fail session.
if (!isRowLegal()) {
    rollback();
    failSession("Cannot process illegal row.");
} else if (rowsProcessed==100) {
    commit();
    rowsProcessed=0;
}
```

## setNull

활성 또는 수동 Java 변환에서 출력 열의 값을 null로 설정합니다.

다음 구문을 사용합니다.

```
setNull(String strColName);
```

다음 표에는 매개 변수가 설명되어 있습니다.

매개 변수	매개 변수 유형	데이터 유형	설명
<code>strColName</code>	입력	문자열	출력 열의 이름입니다.

`setNull` 메서드는 활성 또는 수동 Java 변환에서 출력 열의 값을 null로 설정합니다. 출력 열을 null로 설정한 후에는 출력 행을 생성할 때까지 값을 수정할 수 없습니다.

가져오기패키지 가져오기 및 함수Java 식 탭을 제외한 모든 코드 입력 탭에서 Java 코드에 `setNull` 메서드를 추가할 수 있습니다.

다음 Java 코드는 입력 열 값을 확인하고 출력 열의 해당 값을 `null`로 설정하는 방법을 보여 줍니다.

```
// check value of Q3RESULTS input column
if(isNull("Q3RESULTS")) {
    // set the value of output column to null
    setNull("RESULTS");
}
```

다음 Java 코드를 사용해도 같은 결과를 얻을 수 있습니다.

```
// check value of Q3RESULTS input column
String strColName = "Q3RESULTS";
if(isNull(strColName)) {
    // set the value of output column to null
    setNull(strColName);
}
```

## setOutRowType

출력 행에 대한 업데이트 전략을 설정합니다. `setOutRowType` 메서드는 삽입, 업데이트 또는 삭제를 위해 행에 플래그를 지정할 수 있습니다.

입력 행에서 코드 항목 탭에서만 `setOutRowType`을 사용할 수 있습니다. 업데이트 전략을 설정하도록 구성된 활성 변환에서만 `setOutRowType`을 사용할 수 있습니다. 업데이트 전략을 설정하도록 구성되지 않은 활성 변환에서 `setOutRowType`을 사용하는 경우 세션이 오류를 생성하고 세션이 실패합니다.

다음 구문을 사용합니다.

```
setOutRowType(String rowType);
```

다음 테이블에는 이 메서드의 인수가 설명되어 있습니다.

인수	데이터 유형	입력/출력	설명
rowType	문자열	입력	업데이트 전략 유형. 값은 INSERT, UPDATE 또는 DELETE 일 수 있습니다.

다음 Java 코드를 사용하여 행 유형이 UPDATE 또는 INSERT이고 입력 포트 `input1`의 값이 100 미만이면 현재 행의 입력 유형을 전파하고 `input1`의 값이 100보다 크면 출력 유형을 DELETE로 설정합니다.

```
// Set the value of the output port.
output1 = input1;

// Get and set the row type.
String rowType = getInRowType();
setOutRowType(rowType);

// Set row type to DELETE if the output port value is > 100.
if(input1 > 100)
    setOutRowType(DELETE);
```

# storeMetadata

getMetadata 메서드를 사용하여 런타임에 검색할 수 있는 Java 변환 메타데이터를 저장합니다.

다음 구문을 사용합니다.

```
storeMetadata (String key String data);
```

다음 표에는 매개 변수가 설명되어 있습니다.

매개 변수	매개 변수 유형	데이터 유형	설명
키	입력	문자열	메타데이터를 식별합니다. storeMetadata 메서드를 사용하려면 메타데이터를 식별할 키가 필요합니다. 키를 임의의 문자열로 정의합니다.
데이터	입력	문자열	Java 변환 메타데이터로 저장할 데이터입니다.

storeMetadata 메서드를 Java 코드의 다음 코드 입력 탭에 추가할 수 있습니다.

- 도우미
- 입력 시
- 끝에서
- 최적화 프로그램 인터페이스
- 함수

푸시인 최적화를 위한 필터 조건을 허용하도록 활성 변환에서 storeMetadata 메서드를 구성할 수 있습니다.

storeMetadata 메서드는 최적화 프로그램이 매핑에서 Java 변환으로 푸시하는 필터 조건을 저장합니다.

```
// Store a filter condition
storeMetadata ("FilterKey", condition);
```

## 제 21 장

# Java 식

이 장에 포함된 항목:

- [Java 식 개요, 322](#)
- [식 정의함수 정의 대화 상자를 사용하여 식 정의, 323](#)
- [단순 인터페이스 작업, 325](#)
- [고급 인터페이스 작업, 326](#)
- [JExpression 클래스 API 참조, 331](#)

## Java 식 개요

Java 프로그래밍 언어를 사용하여 Java 변환에서 PowerCenter 식을 호출할 수 있습니다.

식을 사용하면 Java 변환의 기능을 확대할 수 있습니다. 예를 들어 Java 변환에서 식을 호출하여 입력 또는 출력 포트의 값을 조회하거나 Java 변환 변수의 값을 조회할 수 있습니다.

Java 변환에서 식을 호출하려면 Java 코드를 생성하거나 Java 변환 API 메서드를 사용하여 식을 호출합니다. 식을 호출한 다음 식의 결과를 적절한 코드 항목 탭에 사용합니다. 식을 호출하는 Java 코드를 생성하거나 API 메서드를 사용하여 식을 호출하는 Java 코드를 쓸 수 있습니다.

다음 표에는 Java 변환에서 식을 작성하고 호출할 때 사용할 수 있는 방법이 설명되어 있습니다.

메서드	설명
식 정의함수 정의 대화 상자	식을 호출하는 함수를 작성하고 식의 코드를 생성할 수 있습니다.
단순 인터페이스	단일 API 메서드를 호출하여 식을 호출하고 식의 결과를 얻을 수 있습니다.
고급 인터페이스	식을 정의하고 식을 호출하고 식의 결과를 사용할 수 있습니다. 개체 중심 프로그래밍에 익숙한 경우 식 호출에 대한 더 많은 제어 기능을 사용하려면 고급 인터페이스를 사용하십시오.

## 식 함수 유형

식 편집기를 사용하거나 식 정의 대화 상자에서 식을 작성하거나 함수 정의 대화 상자를 사용하거나 단순 또는 고급 인터페이스를 사용하여 Java 변환에 대한 식을 작성할 수 있습니다.

Java 코드의 입력 또는 출력 포트 변수를 입력 매개 변수로 사용하는 식을 입력할 수 있습니다.

**식 정의** 대화 상자를 사용할 경우 **식 편집기**를 사용하여 **Java** 변환에서 식을 사용하기 전에 식의 유효성을 검사할 수 있습니다.

**함수 정의** 대화 상자를 사용할 경우 **Java** 변환에서 식을 사용하기 전에 식의 유효성을 검사할 수 있습니다.

**Java** 변환에서 다음 유형의 식 함수를 호출할 수 있습니다.

식 함수 유형	설명
변환 언어 함수	일반 식을 처리하도록 설계된 SQL 같은 함수입니다.
사용자 정의 함수	변환 언어 함수를 기반으로 PowerCenterDeveloper tool에서 작성하는 함수입니다.
사용자 지정 함수	사용자 지정 함수 API를 사용하여 작성한 함수입니다.

연결되지 않은 변환 및 ,기본 제공 변수, 사용자 정의 매핑 및 워크플로우 변수, 미리 정의된 워크플로우 변수를 식에서 사용할 수도 있습니다. 예를 들어, 식에서 연결되지 않은 조회 변환을 사용할 수 있습니다.

## 식 정의함수 정의 대화 상자를 사용하여 식 정의

**Java** 식을 정의하는 경우 함수를 구성하고 식을 작성한 다음 식을 호출하는 코드를 생성합니다.

**식 정의함수 정의** 대화 상자에서 함수를 정의하고 식을 작성할 수 있습니다.

식 함수를 작성하고 **Java** 변환에서 식을 사용하려면 다음 고급 탭을 완료합니다.

1. 함수 이름, 설명 및 매개 변수를 포함하여 식을 호출하는 함수를 구성합니다. 식을 작성할 때 함수 매개 변수를 사용합니다.
2. 식 구문을 작성하고 식의 유효성을 검증합니다.
3. 식을 호출하는 **Java** 코드를 생성합니다.

디자이너가 변환 개발자의 **Java** 식 코드 항목 탭에 코드를 배치합니다.

개발자가 **함수** 코드 항목 탭에 코드를 배치합니다.

**Java** 코드를 생성한 다음 적절한 코드 항목 탭에서 생성된 함수를 호출하여 식을 호출하거나 단순 또는 고급 인터페이스를 사용하는지 여부에 따라 **JExpression** 개체를 가져옵니다.

**참고:** 식을 작성할 때 식의 유효성을 검증하려면 **식 정의함수 정의** 대화 상자를 사용해야 합니다.

### 1단계. 함수 구성

식을 호출하는 **Java** 함수에 대해 함수 이름, 설명 및 입력 매개 변수를 구성합니다.

함수를 구성할 때 다음 규칙과 지침을 따르십시오.

- 변환의 기존 **Java** 함수 또는 예약된 **Java** 키워드와 충돌하지 않는 고유한 함수 이름을 사용합니다.
- 매개 변수 이름, **Java** 데이터 유형, 전체 자릿수 및 소수 자릿수를 구성해야 합니다. 입력 매개 변수는 변환을 위해 **Java** 코드의 함수를 호출할 때 전달하는 값입니다.
- 날짜 데이터 유형을 식으로 전달하려면 입력 매개 변수에 문자열 데이터 유형을 사용합니다.

식이 날짜 데이터 유형을 반환하는 경우 단순 인터페이스에서는 반환 값을 문자열 데이터 유형으로 사용하고 고급 인터페이스에서는 문자열 또는 긴 정수 데이터 유형을 사용할 수 있습니다.

## 2단계. 식 작성 및 유효성 검사

식을 작성하는 경우 함수에 대해 구성된 매개 변수를 사용합니다.

또한 식에서 변환 언어 함수, 사용자 지정 함수 또는 다른 사용자 정의 함수를 사용할 수 있습니다. **함수 정의** 대화 상자 **식 정의** 대화 상자 또는 **식 편집기** 대화 상자에서 식을 작성하고 유효성을 검사할 수 있습니다.

## 3단계. 식에 대한 Java 코드 생성

함수 및 함수 매개 변수를 구성하여 식을 정의하고 유효성을 검사한 후 식을 호출하는 **Java** 코드를 생성할 수 있습니다.

디자이너개발자가 생성된 **Java** 코드를 **Java 식함수** 코드 항목 탭에 배치합니다. 생성된 **Java** 코드를 사용하여 변환 개발자의 코드 항목 탭에서 식을 호출하는 함수를 호출합니다. 단순 또는 고급 **Java** 코드를 생성할 수 있습니다.

식을 호출하는 **Java** 코드를 생성한 다음에는 식을 편집하거나 유효성을 다시 검사할 수 없습니다. 코드를 생성한 다음 식을 수정하려면 식을 다시 작성해야 합니다.

## 식 정의함수 정의 대화 상자를 사용한 식 작성 및 Java 코드 생성

**식 정의함수 정의** 대화 상자에서 식을 호출하는 함수를 작성할 수 있습니다.

식을 호출하는 함수를 작성하려면 다음 단계를 완료하십시오.

1. 변환 개발자에서 **Java** 변환을 열거나 새 **Java** 변환을 작성합니다.
2. **Java 코드** 탭에서 **새 함수 식 정의** 링크를 클릭합니다.  
**식 정의함수 정의** 대화 상자가 표시됩니다.
3. 함수 이름을 입력합니다.
4. 필요한 경우 식의 설명을 입력합니다.  
최대 2,000자까지 입력할 수 있습니다.
5. 함수에 대한 매개 변수인수를 작성합니다.  
매개 변수인수를 작성할 때 매개 변수인수 이름, 데이터 유형, 전체 자릿수 및 배율을 구성합니다.
6. **편집기 실행**을 클릭하여 작성한 매개 변수로 식을 작성합니다. **식** 탭에서 작성한 인수를 포함하는 식을 작성합니다.
7. 식의 유효성을 검사하려면 **유효성 검사**를 클릭합니다.
8. 필요한 경우 **식** 상자에 식을 입력합니다. 그런 다음 **유효성 검사**를 클릭하여 식의 유효성을 검사합니다.
9. 고급 인터페이스를 사용하여 **Java** 코드를 생성하려면 **고급 코드 생성** 옵션을 선택합니다. 그런 다음 **생성**을 클릭합니다.

DesignerDeveloper가 식을 호출하는 함수를 **Java 식함수** 코드 항목 탭에 생성합니다.

## Java 식 템플릿

식에 대한 단순 또는 고급 **Java** 코드를 사용하여 식에 대한 **Java** 코드를 생성할 수 있습니다.

식의 템플릿을 기반으로 식에 대한 **Java** 코드가 생성됩니다.

다음 예제에는 단순 **Java** 코드에 생성된 **Java** 식 템플릿이 표시되어 있습니다.

```
Object function_name (Java datatype x1[,  
                        Java datatype x2 ...] )  
{  
    throws SDK Exception
```



```

return (Object)invokeJExpression( String expression,
                                   new Object [] { x1[, x2, ... ]} );
}

```

다음 예제에는 고급 인터페이스를 사용하여 생성된 Java 식 템플릿이 표시되어 있습니다.

```

JExpression function_name () throws SDKException
{
    JExprParamMetadata params[] = new JExprParamMetadata[number of parameters];
    params[0] = new JExprParamMetadata (
        EDataType.STRING, // data type
        20, // precision
        0 // scale
    );

    ...
    params[number of parameters - 1] = new JExprParamMetadata (
        EDataType.STRING, // data type
        20, // precision
        0 // scale
    );

    ...
    return defineJExpression(String expression,params);
}

```

## 단순 인터페이스 작업

invokeJExpression Java API 메서드를 사용하여 단순 인터페이스에서 식을 호출할 수 있습니다.

### invokeJExpression

식을 호출하고 식의 값을 반환합니다.

다음 구문을 사용합니다.

```

(datatype)invokeJExpression(
    String expression,
    Object[] paramMetadataArray);

```

invokeJExpression 메서드의 입력 매개 변수는 식을 나타내는 문자열 값과 식 입력 매개 변수를 포함하는 개체의 배열입니다.

다음 표에는 매개 변수가 설명되어 있습니다.

매개 변수	매개 변수 유형	데이터 유형	설명
식	입력	문자열	식을 나타내는 문자열입니다.
paramMetadataArray	입력	Object[]	식의 입력 매개 변수를 포함하는 개체의 배열입니다.

가져오기 및 함수패키지 가져오기 및 Java 식 탭을 제외한 모든 코드 항목 탭의 Java 코드에 invokeJExpression 메서드를 추가할 수 있습니다.

다음 규칙 및 지침에 따라 `invokeJExpression` 메서드를 사용하십시오.

- 반환 데이터 유형. `invokeJExpression` 메서드의 반환 데이터 유형은 개체입니다. 함수의 반환 값을 적절한 데이터 유형으로 캐스팅해야 합니다.

정수, 배열, 문자열 및 바이트[] 데이터 유형으로 값을 반환할 수 있습니다.

- 행 유형. `invokeJExpression` 메서드의 반환 값에 대한 행 유형은 `INSERT`입니다.

반환 값에 서로 다른 행 유형을 사용하려면 고급 인터페이스를 사용합니다.

- Null 값. Null 값을 매개 변수로 전달하거나 `invokeJExpression` 메서드의 반환 값이 Null인 경우 이 값은 Null 표시기로 처리됩니다.

예를 들어 식의 반환 값이 Null이고 반환 데이터 유형이 문자열인 경우 Null 값과 함께 문자열이 반환됩니다.

- 날짜 데이터 유형. 날짜 데이터 유형의 입력 매개 변수는 문자열 데이터 유형으로 변환해야 합니다.

식의 문자열을 날짜 데이터 유형으로 사용하려면 `to_date()` 함수를 사용하여 문자열을 날짜 데이터 유형으로 변환해야 합니다.

또한 날짜 데이터 유형을 반환하는 모든 식의 반환 유형을 문자열 데이터 유형으로 캐스팅해야 합니다.

**참고:** 식에 전달하는 매개 변수에는 문자 `x`로 시작되는 연속 번호를 지정해야 합니다. 예를 들어 매개 변수 3개를 식에 전달하는 경우 매개 변수의 이름을 `x1`, `x2` 및 `x3`으로 지정합니다.

## 단순 인터페이스 예제

도우미 코드도우미 및 입력 행에서 입력 시 코드 항목 탭에서 `invokeJExpression` API 메서드를 사용하는 식을 정의하고 호출할 수 있습니다.

다음 예는 Java 변환에서 NAME 및 ADDRESS 입력 포트에 대해 조회를 완료하고 반환 값을 COMPANY\_NAME 출력 포트에 할당하는 방법을 보여줍니다.

입력 행에서 입력 시 코드 항목 탭에서 다음 코드를 입력합니다.

```
COMPANY_NAME = (String)invokeJExpression("lkp.my_lookup(X1,X2)", new Object [] {str1 ,str2} );
generateRow();
```

## 고급 인터페이스 작업

고급 인터페이스에서는 개체 지향 API 메서드를 사용하여 식을 정의 및 호출하고 식 결과를 가져올 수 있습니다.

다음 테이블에는 고급 인터페이스에서 사용할 수 있는 클래스 및 API 메서드가 설명되어 있습니다.

클래스 또는 API 메서드	설명
EDatatype 클래스	식의 데이터 유형을 열거합니다.
JExprParamMetadata 클래스	식의 각 매개 변수에 대한 메타데이터를 포함합니다. 매개 변수 메타데이터에는 데이터 유형, 전체 자릿수 및 배열이 포함됩니다.
defineJExpression API 메서드	식을 정의합니다. PowerCenter 식 문자열 및 매개 변수를 포함합니다.

클래스 또는 API 메서드	설명
invokeJExpression API 메서드	식을 호출합니다.
JExpression 클래스	메타데이터 생성/호출/가져오기, 식 결과 가져오기, 반환 데이터 유형 확인을 위한 메서드를 포함합니다.

## 고급 인터페이스를 사용하여 식 호출

고급 인터페이스를 사용하여 식을 정의하고 호출하고 결과를 얻을 수 있습니다.

1. **도우미 코드** 또는 **입력 행에서도우미** 또는 **입력 시** 코드 항목 탭에 식의 각 매개 변수인수에 대한 JExprParamMetadata 클래스의 인스턴스를 작성하고 메타데이터의 값을 설정합니다. 필요한 경우 defineJExpression 메서드의 JExprParamMetadata 개체를 인스턴스화할 수 있습니다.
2. defineJExpression 메서드를 사용하여 식의 JExpression 개체를 가져옵니다.
3. 적절한 코드 항목 탭에서 invokeJExpression 메서드를 사용하여 식을 호출합니다.
4. isResultNull 메서드를 사용하여 반환 값의 결과를 검사합니다.
5. getResultDataType 및 getResultMetadata 메서드를 사용하면 반환 값의 데이터 유형 또는 메타데이터를 가져올 수 있습니다.
6. 적절한 API 메서드를 사용하여 식의 결과를 가져옵니다. getInt, getDouble, getStringBuffer 및 getBytes 메서드를 사용할 수 있습니다.

## 고급 인터페이스 작업에 대한 규칙 및 지침

고급 인터페이스 작업을 수행할 때 규칙 및 지침을 알고 있어야 합니다.

다음 규칙 및 지침을 사용하십시오.

- NULL 값을 매개 변수로 전달하거나 식의 결과가 NULL인 경우 값이 NULL 표시기로 처리됩니다. 예를 들어 식의 결과가 NULL이고 반환 데이터 유형이 문자열인 경우 문자열이 NULL 값으로 반환됩니다. isResultNull 메서드를 사용하면 식의 결과를 확인할 수 있습니다.
- 식에서 사용할 수 있으려면 날짜 데이터 유형의 입력 매개 변수를 문자열로 변환해야 합니다. 식의 문자열을 날짜 데이터 유형으로 사용하려면 to\_date() 함수를 사용하여 문자열을 날짜 데이터 유형으로 변환해야 합니다.

날짜 데이터 유형을 문자열 또는 긴 정수 데이터 유형으로 변환하는 식 결과를 얻을 수 있습니다.

날짜 데이터 유형을 문자열 데이터 유형으로 변환하는 식 결과를 얻으려면 getStringBuffer 메서드를 사용합니다. 날짜 데이터 유형을 긴 정수 데이터 유형으로 변환하는 식 결과를 얻으려면 getLong 메서드를 사용합니다.

## EDataType 클래스

식에 사용된 **Java** 데이터 유형을 열거합니다. 식의 반환 데이터 유형을 가져오거나 **JExprParamMetadata** 개체의 매개 변수에 대한 데이터 유형을 할당합니다. **EDataType** 클래스는 인스턴스화가 필요하지 않습니다.

다음 표에는 식의 **Java** 데이터 유형에 대한 열거 값이 나열되어 있습니다.

데이터 유형	열거 값
INT	1
DOUBLE	2
STRING	3
BYTE_ARRAY	4
DATE_AS_LONG	5

다음 예는 **EDataType** 클래스를 사용하여 문자열 데이터 유형을 **JExprParamMetadata** 개체에 할당하는 **Java** 코드를 보여 줍니다.

```
JExprParamMetadata params[] = new JExprParamMetadata[2];
params[0] = new JExprParamMetadata (
    EDataType.STRING, // data type
    20, // precision
    0 // scale
);
...
```

## JExprParamMetadata 클래스

식의 매개 변수를 나타내고 매개 변수에 대한 메타데이터를 설정하는 개체를 인스턴스화합니다.

**JExprParamMetadata** 개체의 배열을 **defineJExpression** 메서드에 대한 입력으로 사용하여 입력 매개 변수에 대한 메타데이터를 설정하십시오. **JExprParamMetadata** 개체의 인스턴스는 **Java 식함수** 코드 항목 탭 또는 **defineJExpression**에서 작성할 수 있습니다.

다음 구문을 사용합니다.

```
JExprParamMetadata paramMetadataArray[] = new JExprParamMetadata[numberOfParameters];
paramMetadataArray[0] = new JExprParamMetadata(datatype, precision, scale);
...
paramMetadataArray[numberOfParameters - 1] = new JExprParamMetadata(datatype, precision, scale);
```

다음 표에는 인수가 설명되어 있습니다.

인수	인수 유형	인수 데이터 유형	설명
데이터 유형	입력	EDataType	매개 변수의 데이터 유형입니다.
전체 자릿수	입력	정수	매개 변수의 전체 자릿수입니다.
배율	입력	정수	매개 변수의 배율입니다.

예를 들어 데이터 유형이 문자열이고 전체 자릿수가 20이며 배열이 0인 JExprParamMetadata 개체 2개의 배열을 인스턴스화하려면 다음 Java 코드를 사용합니다.

```
JExprParamMetadata params[] = new JExprParamMetadata[2];
params[0] = new JExprParamMetadata(EDatatype.STRING, 20, 0);
params[1] = new JExprParamMetadata(EDatatype.STRING, 20, 0);
return defineJExpression("LKP.LKP_addresslookup(X1,X2)",params);
```

## defineJExpression

식 문자열 및 입력 매개 변수를 포함하여 식을 정의합니다. **defineJExpression** 메서드의 인수에는 입력 매개 변수와 식 구문을 정의하는 문자열 값을 포함하는 JExprParamMetadata 개체의 배열이 포함됩니다.

다음 구문을 사용합니다.

```
defineJExpression(
    String expression,
    Object[] paramMetadataArray
);
```

다음 표에는 매개 변수가 설명되어 있습니다.

매개 변수	유형	데이터 유형	설명
식	입력	문자열	식을 나타내는 문자열입니다.
paramMetadataArray	입력	Object[]	식의 입력 매개 변수를 포함하는 JExprParamMetadata 개체의 배열입니다.

**defineJExpression** 메서드는 가져오기 및 함수 탭을 제외한 모든 코드 항목 탭의 Java 코드에 추가할 수 있습니다.

**defineJExpression** 메서드를 사용하려면 식의 입력 매개 변수를 나타내는 JExprParamMetadata 개체의 배열을 인스턴스화해야 합니다. 매개 변수에 대한 메타데이터 값을 설정하고 배열을 **defineJExpression** 메서드에 매개 변수로 전달하십시오.

예를 들어 다음 Java 코드는 두 문자열의 값을 조회하는 식을 작성합니다.

```
JExprParamMetadata params[] = new JExprParamMetadata[2];
params[0] = new JExprParamMetadata(EDatatype.STRING, 20, 0);
params[1] = new JExprParamMetadata(EDatatype.STRING, 20, 0);
defineJExpression("lkp.mylookup(x1,x2)",params);
```

**참고:** 식에 전달하는 매개 변수에는 문자 x로 시작되는 연속 번호를 지정해야 합니다. 예를 들어 매개 변수 3개를 식에 전달하는 경우 매개 변수의 이름을 x1, x2 및 x3으로 지정합니다.

## JExpression 클래스

식을 작성 및 호출하고 식의 값을 반환하고 반환 데이터 유형을 검사하는 메서드가 포함됩니다.

다음 표에는 JExpression 클래스의 메서드가 나열되어 있습니다.

메서드 이름	설명
invoke	식을 호출합니다.
getResultDataType	식 결과의 데이터 유형을 반환합니다.
getResultMetadata	식 결과의 메타데이터를 반환합니다.

메서드 이름	설명
isResultNull	식 결과의 결과 값을 검사합니다.
getInt	식의 결과 값을 정수 데이터 유형으로 반환합니다.
getDouble	식의 결과 값을 배정밀도 데이터 유형으로 반환합니다.
getStringBuffer	식의 결과 값을 문자열 데이터 유형으로 반환합니다.
getBytes	식의 결과 값을 바이트[] 데이터 유형으로 반환합니다.

## 고급 인터페이스 예제

고급 인터페이스를 사용하여 Java 변환에서 조회 식을 작성하고 호출할 수 있습니다.

다음 예는 식을 호출하는 함수를 작성하고 식을 호출하여 반환 값을 얻는 Java 코드를 보여 줍니다. 이 예에서는 데이터 유형이 문자열인 두 입력 포트 NAME 및 COMPANY의 값이 myLookup 함수에 전달됩니다. myLookup 함수는 조회 식을 사용하여 ADDRESS 출력 포트에 대한 값을 조회합니다.

**참고:** 이 예에서는 LKP\_addresslookup이라는 매핑에 연결되지 않은 조회 변환이 있는 것으로 가정합니다.

**도우미 코드**도우미 탭(변환 개발자)에서 다음 Java 코드를 사용합니다.

```
JExpression addressLookup() throws SDKException
{
    JExprParamMetadata params[] = new JExprParamMetadata[2];
    params[0] = new JExprParamMetadata (
        EDataType.STRING,    // data type
        50,                  // precision
        0                    // scale
    );
    params[1] = new JExprParamMetadata (
        EDataType.STRING,    // data type
        50,                  // precision
        0                    // scale
    );
    return defineJExpression("LKP.LKP_addresslookup(X1,X2)",params);
}
JExpression lookup = null;
boolean isJExprObjCreated = false;
```

**입력 행에서 입력 시** 탭에 다음 Java 코드를 사용하면 식이 호출되고 ADDRESS 포트의 값이 반환됩니다.

```
...
if(!isJExprObjCreated)
{
    lookup = addressLookup();
    isJExprObjCreated = true;
}
lookup = addressLookup();
lookup.invoke(new Object [] {NAME,COMPANY}, ERowType.INSERT);
EDataType addressDataType = lookup.getResultDataType();
if(addressDataType == EDataType.STRING)
{
    ADDRESS = (lookup.getStringBuffer()).toString();
} else {
    logError("Expression result datatype is incorrect.");
}
...
```

# JExpression 클래스 API 참조

JExpression 클래스에는 식 작성 및 호출, 식의 값 반환 및 반환 데이터 유형 검사를 위한 API 메서드가 포함됩니다.

JExpression 클래스 포함되는 API 메서드는 다음과 같습니다.

- `getBytes`
- `getDouble`
- `getInt`
- `getLong`
- `getResultDataType`
- `getResultMetadata`
- `getStringBuffer`
- `invoke`
- `isResultNull`

## getBytes

식의 결과 값을 바이트[] 데이터 유형으로 반환합니다. AES\_ENCRYPT 함수로 데이터를 암호화하는 식의 결과를 가져옵니다.

다음 구문을 사용합니다.

```
objectName.getBytes();
```

다음 예는 AES\_ENCRYPT 함수를 사용하여 이진 데이터를 암호화하는 식의 결과를 가져오는 Java 코드입니다. 여기에서 JExprEncryptData는 JExpression 개체입니다.

```
byte[] newBytes = JExprEncryptData.getBytes();
```

## getDouble

식의 결과 값을 배정밀도 데이터 유형으로 반환합니다.

다음 구문을 사용합니다.

```
objectName.getDouble();
```

다음 예는 급여 값을 배정밀도로 반환하는 식의 결과를 가져오는 Java 코드입니다. 여기에서 JExprSalary는 JExpression 개체입니다.

```
double salary = JExprSalary.getDouble();
```

## getInt

식의 결과 값을 정수 데이터 유형으로 반환합니다.

다음 구문을 사용합니다.

```
objectName.getInt();
```

예를 들어 다음 Java 코드를 사용하면 직원 ID 번호를 정수로 반환하는 식의 결과를 얻을 수 있습니다. 여기에서 findEmpID는 JExpression 개체입니다.

```
int empID = findEmpID.getInt();
```

## getLong

식 결과 값을 긴 정수 데이터 유형으로 반환합니다. 날짜 데이터 유형을 사용하는 식 결과를 가져옵니다.

다음 구문을 사용합니다.

```
objectName.getLong();
```

날짜 값을 긴 정수 데이터 유형으로 반환하는 식 결과를 가져오려면 다음 예제 Java 코드를 사용하십시오. 여기서 JExprCurrentDate는 JExpression 개체입니다.

```
long currDate = JExprCurrentDate.getLong();
```

## getResultDataType

식 결과의 데이터 유형을 반환합니다. EDataType 값을 반환합니다.

다음 구문을 사용합니다.

```
objectName.getResultDataType();
```

다음 예는 식을 호출하고 결과의 데이터 유형을 변수 데이터 유형에 할당하는 Java 코드를 보여 줍니다.

```
myObject.invoke(new Object[] { NAME,COMPANY }, ERowType INSERT);  
EDataType dataType = myObject.getResultDataType();
```

## getResultMetadata

식 결과의 메타데이터를 반환합니다. getResultMetadata를 사용하여 식 결과의 전체 자릿수, 배율 및 데이터 유형을 가져올 수 있습니다. 식의 반환 값의 메타데이터를 JExprParamMetadata 개체에 할당할 수 있습니다. 결과 메타데이터를 검색하려면 getScale, getPrecision 및 getDataType 개체를 사용합니다.

다음 구문을 사용합니다.

```
objectName.getResultMetadata();
```

다음 예는 myObject의 반환 값의 배율, 전체 자릿수 및 데이터 유형을 변수에 할당하는 Java 코드를 보여 줍니다.

```
JExprParamMetadata myMetadata = myObject.getResultMetadata();  
int scale = myMetadata.getScale();  
int prec = myMetadata.getPrecision();  
int datatype = myMetadata.getDataType();
```

**참고:** getDataType 개체 메서드는 데이터 유형의 정수 값을 EDataType으로 열거하여 반환합니다.

## getStringBuffer

식의 결과 값을 문자열 데이터 유형으로 반환합니다.

다음 구문을 사용합니다.

```
objectName.getStringBuffer();
```

두 개의 연결된 문자열을 반환하는 식 결과를 얻으려면 다음 예제 Java 코드를 사용하십시오. 여기서 JExprConcat는 JExpression 개체입니다.

```
String result = JExprConcat.getStringBuffer();
```



## invoke

식을 호출합니다. `invoke`의 인수에는 입력 매개 변수 및 행 유형을 정의하는 개체가 포함됩니다. `invoke` 메서드를 사용하려면 먼저 `JExpression` 개체를 인스턴스화해야 합니다. 행 유형으로는 `ERowType.INSERT`, `ERowType.DELETE` 및 `ERowType.UPDATE`를 사용합니다.

다음 구문을 사용합니다.

```
objectName.invoke(  
    new Object[] { param1[, ... paramN ]},  
    rowType  
);
```

다음 표에는 인수가 설명되어 있습니다.

인수	데이터 유형	입력/ 출력	설명
objectName	JExpression	입력	JExpression 개체 이름입니다.
매개 변수	-	입력	식의 입력 값을 포함하는 개체 배열입니다.

예를 들어 식을 나타내는 `JExpression` 개체를 반환하는 `address_lookup()`이라는 **Java** 식함수 코드 항목 탭에 함수를 작성합니다. 다음 코드를 사용하면 입력 포트 `NAME` 및 `COMPANY`를 사용하는 식이 호출됩니다.

```
JExpression myObject = address_lookup();  
myObject.invoke(new Object[] { NAME,COMPANY }, ERowType INSERT);
```

## isResultNull

식 결과의 값을 검사합니다.

다음 구문을 사용합니다.

```
objectName.isResultNull();
```

다음 예는 식을 호출하고 식의 반환 값이 `Null`이 아닐 경우 반환 값을 변수 주소에 할당하는 **Java** 코드입니다.

```
JExpression myObject = address_lookup();  
myObject.invoke(new Object[] { NAME,COMPANY }, ERowType INSERT);  
if(!myObject.isResultNull()) {  
    String address = myObject.getStringBuffer();  
}
```

## 제 22 장

# 조이너 변환

이 장에 포함된 항목:

- [조이너 변환 개요, 334](#)
- [조이너 변환의 고급 속성, 335](#)
- [조이너 캐시, 336](#)
- [조이너 변환의 포트, 337](#)
- [동적 매핑의 조이너 변환, 337](#)
- [조이너 변환의 포트 선택기, 338](#)
- [조인 조건 정의, 340](#)
- [조인 유형, 343](#)
- [조이너 변환의 정렬된 입력, 346](#)
- [동일한 소스의 데이터 조인, 349](#)
- [소스 파이프라인 차단, 351](#)
- [조이너 변환 성능 팁, 352](#)
- [조이너 변환에 대한 규칙 및 지침, 352](#)
- [조이너 변환 - 비원시 환경, 353](#)

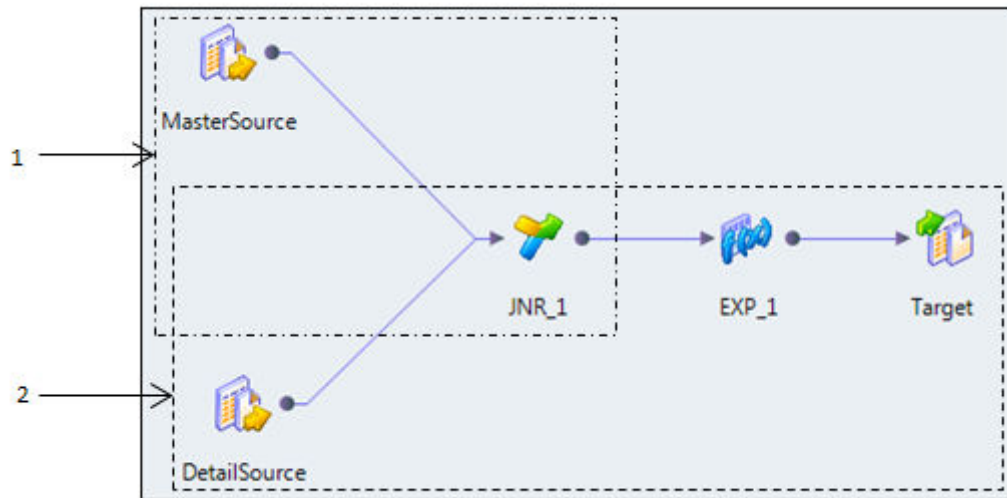
## 조이너 변환 개요

조이너 변환은 다른 위치 또는 다른 파일 시스템에 상주하는 유형이 다른 관련 소스 2개의 소스 데이터를 조인합니다. 동일한 소스의 데이터를 조인할 수도 있습니다. 조이너 변환은 여러 그룹이 포함되는 활성 변환입니다.

조이너 변환은 소스와 최소 1개 이상의 일치하는 열을 조인합니다. 조이너 변환은 두 소스에서 하나 이상의 열 쌍을 일치시키는 조건을 사용합니다.

두 입력 파이프라인에는 마스터 파이프라인 및 세부 파이프라인 또는 마스터 및 세부 분기가 포함됩니다. 마스터 파이프라인은 조이너 변환에서 종료되고 세부 파이프라인은 대상까지 계속됩니다.

다음 그림은 조이너 변환이 포함된 매핑의 마스터 및 세부 파이프라인을 보여 줍니다.



1. 마스터 파이프라인
2. 세부 파이프라인

매핑에서 3개 이상의 소스를 조인하려면 조이너 변환의 출력을 다른 소스 파이프라인과 조인해야 합니다. 모든 소스 파이프라인이 조인될 때까지 조이너 변환을 매핑에 추가합니다.

## 조이너 변환의 고급 속성

조이너 변환 데이터에 대한 데이터 통합 서비스의 처리 방식을 결정하는 데 도움이 되는 속성을 구성할 수 있습니다.

**고급** 탭에서 다음 속성을 구성합니다.

### 조이너 데이터 캐시 크기

매핑 실행 시작 시 변환을 위해 데이터 통합 서비스가 데이터 캐시에 할당하는 메모리의 양입니다. "자동"을 선택하면 데이터 통합 서비스가 런타임 시 자동으로 메모리 요구 사항을 계산합니다. 캐시 크기를 조정하려는 경우 특정 값을 바이트 단위로 입력합니다. 기본값은 자동입니다.

### 조이너 인덱스 캐시 크기

매핑 실행 시작 시 변환을 위해 데이터 통합 서비스가 인덱스 캐시에 할당하는 메모리의 양입니다. "자동"을 선택하면 데이터 통합 서비스가 런타임 시 자동으로 메모리 요구 사항을 계산합니다. 캐시 크기를 조정하려는 경우 특정 값을 바이트 단위로 입력합니다. 기본값은 자동입니다.

### 캐시 디렉터리

데이터 통합 서비스가 인덱스 캐시 파일 및 데이터 캐시 파일을 작성하는 디렉터리입니다. 디렉터리가 존재하고 캐시 파일을 위한 디스크 공간이 충분한지 확인하십시오.

캐시 분할 중에 성능을 향상시키려면 여러 디렉터리를 세미콜론으로 구분하여 입력하십시오. 캐시 분할은 변환을 처리하는 각 파티션에 대해 별도의 캐시를 작성합니다.

기본값은 **CacheDir** 시스템 매개 변수입니다. 이 속성에 대해 다른 시스템 매개 변수를 구성하거나 사용자 정의 매개 변수를 구성할 수 있습니다.

### 정렬된 입력

입력 데이터가 그룹을 기준으로 미리 정렬되었음을 나타냅니다. 정렬된 데이터를 조인하려면 정렬된 입력을 선택합니다. 정렬된 입력을 사용하면 성능이 향상됩니다.

### 마스터 정렬 순서

마스터 소스 데이터의 정렬 순서를 지정합니다. 마스터 소스 데이터가 오름차순인 경우 오름차순을 선택합니다. 오름차순을 선택한 경우 정렬된 입력도 활성화합니다. 기본값은 자동입니다.

### 추적 수준

이 변환에 대해 로그에 표시되는 세부 정보의 양입니다. 간단, 보통, 자세한 정보 표시 초기화 또는 자세한 정보 표시 데이터를 선택할 수 있습니다. 기본값은 보통입니다.

### 관련 항목:

- [“캐시 크기” 페이지 71](#)

## 조이너 캐시

조이너 변환을 사용하는 매핑을 실행하면 데이터 통합 서비스가 인덱스 캐시 및 데이터 캐시를 메모리에 작성하여 변환을 실행합니다. 메모리 캐시에서 사용할 수 있는 공간보다 더 많은 공간이 데이터 통합 서비스에 필요한 경우 데이터 통합 서비스는 오버플로우 데이터를 캐시 파일에 저장합니다.

조이너 변환을 사용하는 매핑을 실행하는 경우 데이터 통합 서비스가 마스터 및 세부 소스의 행을 동시에 읽고 마스터 행을 기반으로 인덱스 및 데이터 캐시를 작성합니다. 데이터 통합 서비스가 세부 소스 데이터 및 캐시된 마스터 데이터를 기반으로 조인을 수행합니다.

조이너 변환의 유형은 데이터 통합 서비스가 캐시에 저장하는 행의 수를 결정합니다.

다음 테이블에는 데이터 통합 서비스가 다른 유형의 조이너 변환에 대해 캐시에 저장하는 정보가 설명되어 있습니다.

조이너 변환 유형	인덱스 캐시	데이터 캐시
정렬되지 않은 입력	고유한 인덱스 키가 포함된 조인 조건에 모든 마스터 행을 저장합니다.	모든 마스터 행을 저장합니다.
다른 소스가 포함된 정렬된 입력	고유한 인덱스 키가 포함된 조인 조건에 100개의 마스터 행을 저장합니다.	인덱스 캐시에 저장된 행에 해당하는 마스터 행을 저장합니다. 마스터 데이터에 동일한 키를 가진 여러 행이 포함된 경우 데이터 통합 서비스가 데이터 캐시에 100개보다 많은 행을 저장합니다.
동일한 소스가 포함된 정렬된 입력	고유한 키가 포함된 조인 조건에 모든 마스터 또는 세부 행을 저장합니다. 데이터 통합 서비스가 마스터 파이프라인보다 빠른 세부 파이프라인을 처리하는 경우 세부 행을 저장합니다. 그렇지 않은 경우 마스터 행을 저장합니다. 저장하는 행수는 마스터 및 세부 파이프라인의 처리 비율에 따라 다릅니다. 하나의 파이프라인이 다른 파이프라인보다 행을 더 빨리 처리하는 경우 데이터 통합 서비스는 이미 처리된 모든 행을 캐시합니다. 서비스는 다른 파이프라인이 행 처리를 마칠 때까지 행을 캐시된 상태로 유지합니다.	인덱스 캐시에 저장된 행의 데이터를 저장합니다. 인덱스 캐시가 마스터 파이프라인의 키를 저장하는 경우 데이터 캐시는 마스터 파이프라인의 데이터를 저장합니다. 인덱스 캐시가 세부 파이프라인의 키를 저장하는 경우 데이터 캐시는 세부 파이프라인의 데이터를 저장합니다.

## 조이너 변환의 포트

조이너 변환에는 데이터 통합 서비스의 조인 수행 방식을 결정하는 여러 포트 유형이 포함됩니다.

조이너 변환의 포트 유형은 다음과 같습니다.

### 마스터

매핑의 마스터 소스에 연결하는 포트입니다.

### 세부 정보

매핑의 세부 소스에 연결하는 포트입니다.

### 동적 포트

동적 매핑의 포트를 받거나 반환합니다. 동적 포트는 업스트림 변환에서 한 개 이상의 열을 받고 각 열에 대해 생성된 포트를 작성할 수 있습니다. 동적 출력 포트는 한 개 이상의 생성된 포트를 반환할 수 있습니다. 입력 규칙을 정의하여 동적 포트에서 받는 열을 확인할 수 있습니다.

포트를 마스터 포트에서 세부 포트로 변경할 수 있습니다. 또한 포트를 세부 포트에서 마스터 포트로 변경할 수도 있습니다. 한 포트의 포트 유형을 변경하는 경우 모든 포트의 포트 유형을 변경해야 합니다. 따라서 마스터 포트를 세부 포트로 변경하는 경우 모든 마스터 포트를 세부 포트로 변경하고 모든 세부 포트를 마스터 포트로 변경해야 합니다.

## 동적 매핑의 조이너 변환

동적 매핑에서 조이너 변환을 사용할 수 있습니다. 조인 조건의 동적 포트 및 생성된 포트를 참조할 수 있습니다.

동적 매핑은 런타임 시 소스, 대상 및 변환 논리가 변경될 수 있는 매핑입니다. 매개 변수 및 규칙을 설정하여 데이터의 구조를 변경할 수 있습니다. 동적 매핑에서 조이너 변환을 사용하는 경우 소스의 구조가 변경될 수 있습니다. 조인 조건의 입력 포트 및 포트도 변경될 수 있습니다.

다음 태스크를 수행하여 동적 매핑에서 조이너 변환을 구성할 수 있습니다.

### 동적 포트 정의

동적 포트 및 생성된 포트를 정의하여 동적 소스의 다른 입력 열을 포함합니다. 조인 조건에 동적 포트 또는 생성된 포트를 포함할 수 있습니다.

### 포트 선택기 정의

조인 조건에서 사용할 포트가 포함된 포트 선택기를 정의합니다. 포트 선택기에 포함할 포트를 결정하는 선택 규칙을 구성합니다. 런타임 시 특정 포트를 포함하도록 포트 선택기를 매개 변수화할 수 있습니다.

### 조인 조건 매개 변수화

전체 조인 조건을 매개 변수화할 수 있습니다. 필요할 수 있는 각 조인 조건에 대한 식 매개 변수를 구성합니다.

동적 매핑에 대한 자세한 내용은 *Informatica Developer 매핑 가이드*를 참조하십시오.

## 조이너 변환의 포트 선택기

조이너 변환에 생성된 포트가 있으면 런타임 시 변환에 다른 생성된 포트가 있는 경우 올바른 조인 조건을 구성해야 합니다.

예를 들어 동적 매핑이 다음 조인 조건으로 조이너 변환을 포함할 수 있습니다.

`CustomerID = CustomerNo`

`CustomerID`는 조이너 변환의 생성된 포트입니다. 매핑에 동적 소스가 있으므로 매핑이 여러 다른 소스 파일 형식을 실행할 수 있습니다. 고객 번호를 포함하는 열에 각 소스 파일의 다른 이름 `CustomerID`, `CustomerNum` 또는 `CustNO`가 있습니다.

조이너 변환에서 포트 선택기를 작성하여 동적 소스로부터 다른 고객 열 이름을 포함할 수 있습니다. 접두사 "`Cust`"가 있는 포트 이름을 포함하는 선택 규칙으로 포트 선택기를 구성합니다.

그런 다음 `CustomerID` 열 이름 대신 포트 선택기 이름을 포함하도록 조인 조건을 구성합니다.

`Customer_PortSelector = CustomerNo`

조인 조건이 "`Cust`"로 시작하는 포트 이름에 올바르게 나타납니다.

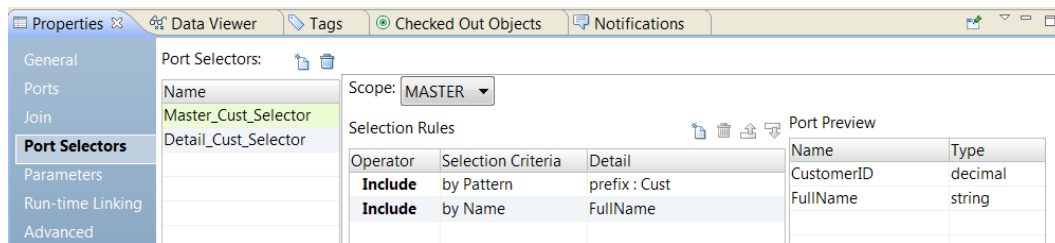
포트 선택기에는 하나 이상의 포트가 있을 수 있습니다. 조인 조건의 마스터 그룹 및 세부 정보 그룹에 동일한 포트 수가 포함되는 경우 조인 조건이 여러 포트를 포함할 수 있습니다.

## 선택 규칙

포트 선택기를 구성하는 경우 선택 규칙을 정의하여 포함할 생성된 포트를 결정합니다. 선택 규칙은 동적 포트에 대해 구성할 수 있는 입력 규칙과 유사합니다.

포트 선택기는 포트 또는 생성된 포트를 포함할 수 있습니다. **포트 선택기** 탭에서 포트 선택기를 구성합니다.

다음 이미지는 **포트 선택기** 탭을 보여 줍니다.



포트 선택기에 대해 다음 속성을 구성합니다.

### 이름

포트 선택기를 식별합니다. 변환에서 여러 포트 선택기를 작성하고 식에서 참조할 수 있습니다.

### 범위

포트 선택기가 적용되는 포트 그룹을 식별합니다. 마스터 또는 세부 정보 범위를 선택합니다.

### 선택 규칙

포트 선택기에 포함될 포트를 결정합니다. 선택 규칙을 작성하면 **포트 미리보기** 패널에 현재 입력 포트에 적합한 포트가 표시됩니다. 이러한 포트는 변경될 수 있습니다. 선택 규칙을 구성하여 여러 소스에서 포트를 포함합니다.

다음 조건에 따라 선택 규칙을 작성할 수 있습니다.

## 연산자

선택 규칙이 반환하는 포트를 포함하거나 제외합니다. 기본값은 포함입니다. 포트를 제외하려면 포트를 포함해야 합니다.

## 선택 조건

작성할 선택 규칙 유형입니다. 포트 유형 또는 열 이름에 따라 규칙을 작성할 수 있습니다. 열 이름에 따라 포트를 포함하려면 특정 이름을 검색하거나 이름의 문자 패턴을 검색합니다.

## 세부 정보

선택 조건에 적용할 값입니다. 선택 조건이 열 이름 기준인 경우 검색할 문자열 또는 이름을 구성합니다. 선택 조건이 포트 유형인 경우 포함할 포트 유형을 선택합니다.

다음 표에는 선택 조건과 해당 조건에 대한 세부 정보를 지정하는 방법이 설명되어 있습니다.

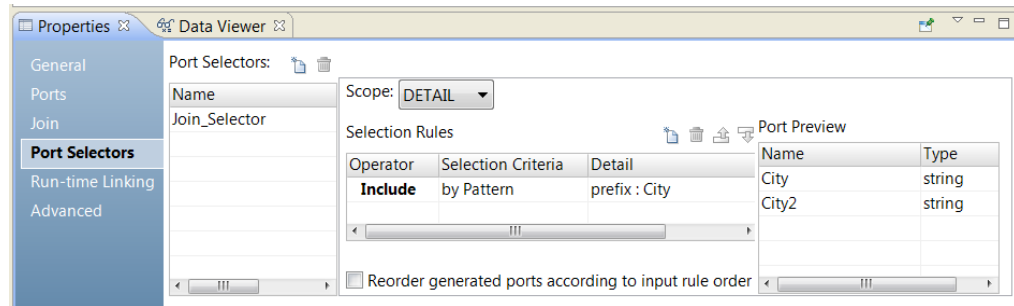
선택 조건	설명	세부 정보
모두	모든 포트를 포함합니다.	세부 정보가 필요하지 않습니다.
이름	포트 이름에 따라 포트를 필터링합니다.	값 목록에서 포트 이름을 선택하거나 포트 또는 포트 목록 유형인 매개 변수를 사용합니다.
유형	각 포트의 데이터 유형에 따라 포트를 필터링합니다.	목록에서 데이터 유형을 선택합니다.
패턴	이름의 문자 문자열 또는 정규식으로 포트를 필터링합니다.	접두사, 접미사 또는 정규식을 포트 이름에 대한 패턴 유형으로 선택합니다. 그런 다음 패턴에 대한 값을 입력하거나 문자열 유형인 매개 변수를 사용합니다.

## 포트 선택기 작성

포트 선택기를 작성하여 동적 식, 조회 조건 또는 조이너 조건에서 사용할 포트를 결정합니다.

1. **포트 선택기** 탭을 클릭합니다.
2. **포트 선택기** 영역에서 **새로 만들기**를 클릭합니다.  
Developer tool이 모든 포트를 포함하는 기본 선택 규칙으로 포트 선택기를 작성합니다.
3. **포트 선택기** 영역에서 포트 선택기 이름을 고유 이름으로 변경합니다.
4. 조이너 변환 또는 조회 변환에 대해 작업하는 경우 범위를 선택합니다.  
사용 가능한 포트는 선택하는 포트 그룹에 따라 변경됩니다.
5. **선택 규칙** 영역에서 **연산자**를 선택합니다.
  - 포함. 포트 선택기에 대한 포트를 포함하는 규칙을 작성합니다. 포트를 제외하려면 포트를 포함해야 합니다.
  - 제외. 포트 선택기에서 특정 포트를 제외하는 규칙을 작성합니다.
6. **선택 조건**을 선택합니다.
  - 이름별. 이름으로 특정 포트를 선택합니다. 범위의 포트 목록에서 포트 이름을 선택할 수 있습니다.
  - 유형별. 유형으로 포트를 선택합니다. 데이터 유형을 하나 이상 선택할 수 있습니다.
  - 패턴별. 포트 이름의 글자 패턴으로 포트를 선택합니다. 특정 글자로 검색하거나 정규식을 작성할 수 있습니다.

다음 이미지는 포트 선택기 탭을 보여 줍니다.



7. 세부 정보 열을 클릭합니다.

입력 규칙 세부 정보 대화 상자가 나타납니다.

8. 포트 기준으로 필터링할 값을 선택합니다.

- 이름별. 선택하여 값 또는 매개 변수 기준으로 포트 목록을 작성합니다. 선택을 클릭하여 목록에서 포트를 선택합니다.
- 유형별. 목록에서 데이터 유형을 하나 이상 선택합니다. **포트 미리보기** 영역에 선택하는 유형의 포트가 표시됩니다.
- 패턴별. 선택하여 포트 이름의 접두사 또는 접미사에서 문자의 특정 패턴을 검색합니다. 또는 선택하여 검색할 정규식을 작성합니다. 매개 변수를 구성하거나 검색할 패턴을 구성합니다.

**포트 미리보기** 영역에는 규칙을 구성한 포트 선택기의 포트가 표시됩니다..

9. 포트 선택기에서 포트 순서를 다시 지정하려면 **생성된 포트의 순서를 입력 규칙 순서에 따라 다시 지정**을 선택합니다.

## 조인 조건 정의

조인 조건에는 데이터 통합 서비스가 두 행을 조인할 때 사용하는 두 입력 소스의 포트가 포함됩니다.

선택된 조인 유형에 따라 데이터 통합 서비스는 행을 결과 집합에 추가하거나 행을 삭제합니다. 조이너 변환은 조인 유형, 조건 및 입력 데이터 소스를 바탕으로 결과 집합을 생성합니다.

조인 조건을 정의하기 전에 마스터 소스 및 세부 소스가 최적의 성능을 제공하도록 구성되었는지 확인하십시오. 매핑을 실행하면 데이터 통합 서비스가 마스터 소스의 각 행을 세부 소스와 비교합니다. 정렬되지 않은 조이너 변환의 성능을 개선하려면 행 수가 더 적은 소스를 마스터 소스로 사용합니다. 정렬된 조이너 변환의 성능을 개선하려면 중복 키 값이 더 적은 소스를 마스터 소스로 사용합니다.

조인 조건에는 조이너 변환의 입력 소스의 포트를 하나 이상 사용합니다. 포트를 추가하면 두 소스를 조인하는 데 필요한 시간이 늘어납니다. 조건에서 포트의 순서는 조이너 변환의 성능에 영향을 미칠 수 있습니다. 조인 조건에 여러 포트를 사용하는 경우 데이터 통합 서비스는 사용자가 지정한 순서로 포트를 비교합니다.

**Char**와 **Varchar** 데이터 유형을 조인하는 경우 데이터 통합 서비스는 **Char** 값을 채우는 공백을 문자열 수에 포함하여 계산합니다.

```
Char(40) = "abcd"
Varchar(40) = "abcd"
```

여기에서 **Char** 값은 36개의 공백으로 채워진 "abcd"이고 데이터 통합 서비스는 **Char** 필드에 후행 공백이 포함되기 때문에 두 필드를 조인하지 않습니다.



**참고:** 조이너 변환은 Null 값을 일치시키지 않습니다. 예를 들어 EMP\_ID1과 EMP\_ID2에 Null 값이 있는 행이 포함되는 경우 이러한 행은 일치로 간주되지 않으므로 두 행이 조인되지 않습니다. Null 값이 포함된 행을 조인하려면 Null 입력을 기본값으로 바꾼 다음 이 기본값에 조인을 수행해야 합니다.

단순 또는 고급 조건 유형을 정의할 수 있습니다. 또한 식 매개 변수를 정의할 수 있습니다. 식 매개 변수는 조인 식이 포함된 매개 변수입니다. 런타임 시 매핑 매개 변수로 매개 변수 값을 변경할 수 있습니다.

## 단순 조건 유형

정렬 또는 정렬되지 않은 조이너 변환에 대해 단순 조건 유형을 정의합니다.

단순 조건에는 지정한 마스터 및 세부 소스를 비교하는 조건이 하나 이상 포함됩니다. 단순 조건은 다음 형식을 사용해야 합니다.

`<master_port> operator <detail_port>`

정렬된 조이너 변환의 경우 조건은 같은 연산자를 사용해야 합니다.

정렬되지 않은 조이너 변환의 경우 조건은 =, !=, >, >=, <, <= 연산자를 사용할 수 있습니다.

예를 들어 EMPLOYEE\_AGE 및 EMPLOYEE\_POSITION 테이블이 포함된 두 소스에 모두 직원 ID 번호가 포함되어 있으면 다음 조건은 두 소스에 나열된 직원이 들어 있는 행의 일치 여부를 확인합니다.

`EMP_ID1 = EMP_ID2`

Developer tool은 단순 조건에서 데이터 유형의 유효성을 검사합니다. 조건에서는 두 포트의 데이터 유형이 같아야 합니다. 데이터 유형이 일치하지 않는 두 포트를 조건에서 사용해야 하는 경우에는 데이터 유형을 일치하도록 변환하십시오.

단순 조건에서 조인 조건 목록을 구성할 수 있습니다. 여러 조인 조건을 구성하는 경우 조인하려면 모든 조건이 true여야 합니다.

예를 들어 단순 조건에서 다음 문을 구성할 수 있습니다.

`StoreID = StoreNO  
Dept = Department  
Salary > Commission`

동일한 문을 고급 조건으로 보는 경우 조인 조건이 다음 식과 같이 나타납니다.

`StoreID = StoreNO AND Dept = Department AND (Salary > Commission)`

## 고급 조건 유형

고급 조건 유형은 정렬되지 않은 조이너 변환에 정의합니다.

고급 조건에는 부울 또는 숫자 값으로 평가되는 모든 식이 포함될 수 있습니다. 고급 조건에는 =, !=, >, >=, <, <= 과 같은 연산자가 포함될 수 있습니다.

조인 조건에 대한 상수를 입력할 수 있습니다. FALSE에 해당하는 숫자는 영(0)입니다. 영(0)이 아닌 값은 TRUE에 해당합니다. 예를 들어 숫자 데이터 유형을 가진 NUMBER\_OF\_UNITS라는 포트가 변환에 포함되어 있습니다. NUMBER\_OF\_UNITS 값이 영(0)일 경우 FALSE를 반환하도록 필터 조건을 구성하십시오. 그렇지 않을 경우 조건에서 TRUE를 반환합니다.

**참고:** 조인 조건에 대한 부울 값으로 단일 동적 포트 또는 포트 선택기를 사용할 수 없습니다.

조인 조건에 식을 입력하려면 **조인** 탭에서 고급 조건 유형을 선택합니다. 식 편집기를 사용하여 포트, 매개 변수, 식, 포트 선택기 및 연산자를 조건에 포함합니다. 생성된 포트를 사용할 수 있습니다. 포트 유형이 숫자인 경우 식 편집기에 단일 포트를 입력할 수 있습니다. 하지만 하나의 포트 선택기를 식으로 입력할 수 없습니다.

예를 들어 직원의 전체 이름을 일치시켜 소스를 조인하려고 합니다. 마스터 소스에 **FirstName** 및 **LastName** 포트가 포함됩니다. 세부 소스에는 **FullName** 포트가 포함됩니다. 이러한 경우 마스터 포트를 연결하고 두 소스의 전체 이름을 일치시키는 다음 조건을 정의할 수 있습니다.

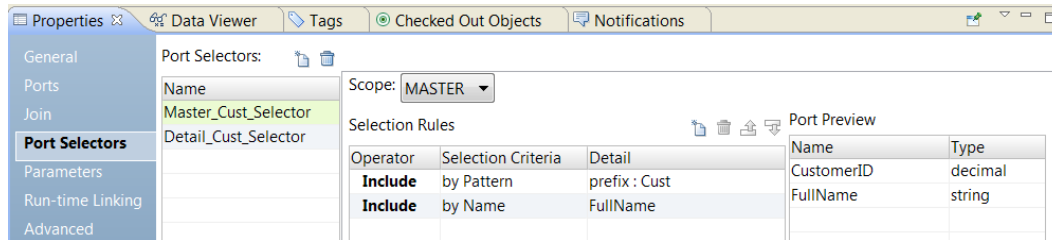
CONCAT(FirstName, LastName) = FullName

## 조인 조건의 포트 선택기

조인 조건에서 포트 선택기를 포함할 수 있습니다. 조인 조건은 마스터 그룹의 포트 선택기 및 세부 정보 그룹의 포트 선택기를 참조해야 합니다.

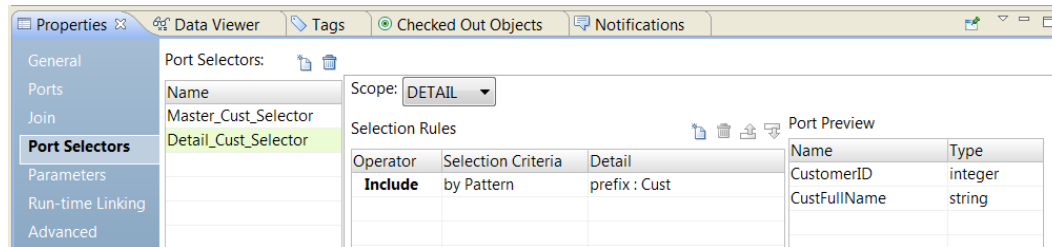
예를 들어 조이너 변환에는 동적 포트가 있습니다. 조인 조건의 여러 생성된 포트를 비교해야 할 수 있습니다.

다음 이미지는 마스터 그룹에 대한 포트 선택기의 필드를 보여 줍니다.



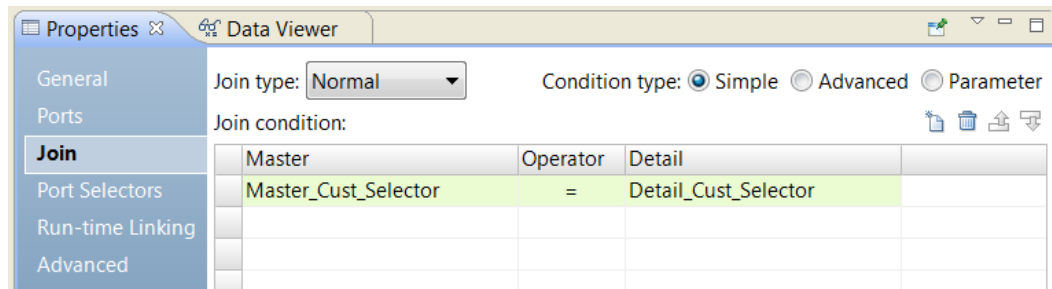
Master\_Cust\_Selector에 CustomerID 및 FullName 포트가 포함되어 있습니다.

다음 이미지는 세부 정보 그룹에 대한 포트 선택기의 필드를 보여 줍니다.



Detail\_Cust\_Selector에 CustomerNo 및 CustFullName 포트가 포함되어 있습니다. 이러한 포트에는 접두가 Cust가 있습니다.

다음 단순 조인 조건을 작성합니다.



조인 조건이 Master\_Cust\_Selector의 각 포트를 Detail\_Cust\_Selector와 비교합니다. 조인 조건은 CustomerID = CustomerNo AND FullName = CustFullName입니다.

각 포트 선택기는 동일한 포트 수를 포함해야 합니다. 포트는 동일한 유형이어야 합니다.

**참고:** 포트 선택기의 범위를 변경하고 단순 유형 조인 조건이 더 이상 올바르지 않은 경우 Developer tool이 조인 유형을 고급으로 전환할 수 있습니다. **조인** 탭에서 조인 조건 유형을 단순 유형으로 다시 전환할 수 있습니다.

## 조인 조건의 동적 포트

포트 선택기의 동적 포트를 참조할 수 있습니다.

동적 포트는 한 개 이상의 생성된 포트를 포함할 수 있습니다. 조인 조건이 동적 포트를 포함하는 경우 마스터 포트의 수는 세부 정보 포트의 수와 동일해야 합니다.

예를 들어 동적 포트 A에는 2개의 생성된 포트가 있습니다.

```
CustomerID  
OrderID
```

동적 포트 B에도 2개의 생성된 포트가 있습니다.

```
CustomerNo  
OrderNo
```

다음 조인 조건은 올바릅니다.

`DynamicPortA = DynamicPortB`

조인 조건이 다음 식으로 확장됩니다.

`CustomerID = CustomerNo AND OrderID = OrderNo`

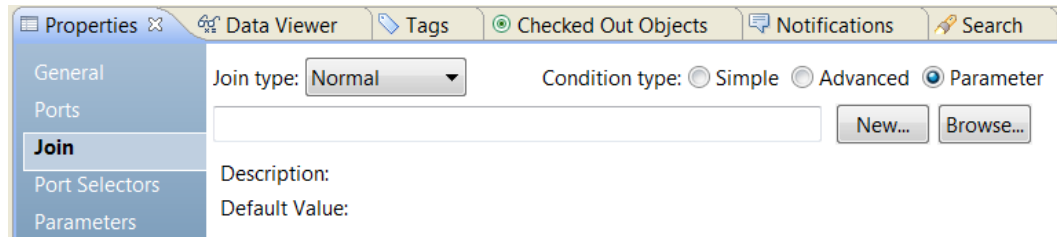
포트 선택기가 동적 포트와 동일한 수의 포트를 포함하는 경우 조인 조건에서 포트 선택기 및 동적 포트를 참조할 수 있습니다.

## 식 매개 변수

조인 조건을 포함하는 식 매개 변수를 정의할 수 있습니다. 조이너 변환에서 매개 변수를 조인 조건으로 선택할 수 있습니다.

매개 변수를 조인 조건에 대해 사용하려면 조인 탭에서 매개 변수 조건 유형을 선택합니다.

다음 이미지는 매개 변수 조건 유형을 선택할 위치를 보여 줍니다.



기존 매개 변수를 찾거나 매개 변수를 작성할 수 있습니다. 매개 변수를 작성하려면 **새로 만들기**를 클릭하고 매개 변수를 정의합니다. 식 편집기에서 식을 작성합니다.

**참고:** 식 매개 변수에는 다른 매개 변수가 포함될 수 없습니다. 식 매개 변수에 매개 변수를 포함하는 경우 데이터 통합 서비스가 런타임 유효성 검사 오류를 표시합니다.

## 조인 유형

조이너 변환에서 조인은 서로 다른 유형의 소스에서 시작될 수 있습니다.

조이너 변환은 다음과 같은 유형의 조인을 지원합니다.

- 보통

- 마스터 외부
- 세부 외부
- 전체 외부

**참고:** 일반 또는 마스터 외부 조인은 전체 외부 또는 세부 외부 조인보다 빠르게 수행됩니다.

두 소스 중 어떤 소스의 데이터도 포함하지 않는 필드가 결과 집합에 포함되는 경우 조이너 변환은 빈 필드를 Null 값으로 채웁니다. NULL을 반환하는 필드가 있다는 것을 알고 NULL이 대상에 삽입되는 것을 원하지 않는 경우에는 해당하는 포트에 대한 기본값을 설정할 수 있습니다.

## 일반 조인

일반 조인을 사용하는 경우 데이터 통합 서비스는 조건에 따라 마스터 및 세부 소스에서 일치하지 않는 모든 데이터 행을 무시합니다.

자동차 부품에 대한 두 데이터 소스인 PARTS\_SIZE 및 PARTS\_COLOR가 있는 경우를 예로 들어 보겠습니다.

마스터 소스인 PARTS\_SIZE 데이터 소스는 다음 데이터를 포함합니다.

PART_ID1	DESCRIPTION	SIZE
1	Seat Cover	Large
2	Ash Tray	Small
3	Floor Mat	Medium

세부 소스인 PARTS\_COLOR 데이터 소스는 다음 데이터를 포함합니다.

PART_ID2	DESCRIPTION	COLOR
1	Seat Cover	Blue
3	Floor Mat	Black
4	Fuzzy Dice	Yellow

두 소스에서 PART\_ID 일치 여부를 확인하여 두 테이블을 조인하려면 다음과 같이 조건을 설정합니다.

PART\_ID1 = PART\_ID2

일반 조인을 사용하여 이러한 테이블을 조인하면 결과 집합에는 다음 데이터가 포함됩니다.

PART_ID	DESCRIPTION	SIZE	COLOR
1	Seat Cover	Large	Blue
3	Floor Mat	Medium	Black

다음 예는 동일한 SQL 문을 보여 줍니다.

```
SELECT * FROM PARTS_SIZE, PARTS_COLOR WHERE PARTS_SIZE.PART_ID1 = PARTS_COLOR.PART_ID2
```

## 마스터 외부 조인

마스터 외부 조인은 세부 소스 데이터의 모든 행 및 마스터 소스의 일치하는 행을 유지합니다. 이 조인은 마스터 소스의 일치하지 않는 행을 무시합니다.

동일한 조건으로 마스터 외부 조인을 사용하여 샘플 테이블을 조인하면 결과 집합에 다음 데이터가 포함됩니다.

PART_ID	DESCRIPTION	SIZE	COLOR
1	Seat Cover	Large	Blue
3	Floor Mat	Medium	Black
4	Fuzzy Dice	NULL	Yellow

Fuzzy Dice에 크기가 지정되지 않았기 때문에 데이터 통합 서비스가 이 필드를 NULL로 채웁니다.

다음 예는 동일한 SQL 문을 보여 줍니다.

```
SELECT * FROM PARTS_SIZE RIGHT OUTER JOIN PARTS_COLOR ON (PARTS_COLOR.PART_ID2 = PARTS_SIZE.PART_ID1)
```

## 세부 외부 조인

세부 외부 조인은 마스터 소스 데이터의 모든 행 및 세부 소스의 일치하는 행을 유지합니다. 이 조인은 세부 소스의 일치하지 않는 행을 무시합니다.

동일한 조건으로 세부 외부 조인을 사용하여 샘플 테이블을 조인하면 결과 집합에 다음 데이터가 포함됩니다.

PART_ID	DESCRIPTION	SIZE	COLOR
1	Seat Cover	Large	Blue
2	Ash Tray	Small	NULL
3	Floor Mat	Medium	Black

Ash Tray에 색상이 지정되지 않았기 때문에 데이터 통합 서비스가 이 필드를 NULL로 채웁니다.

다음 예는 동일한 SQL 문을 보여 줍니다.

```
SELECT * FROM PARTS_SIZE LEFT OUTER JOIN PARTS_COLOR ON (PARTS_SIZE.PART_ID1 = PARTS_COLOR.PART_ID2)
```

## 전체 외부 조인

전체 외부 조인은 마스터 소스 데이터와 세부 소스 데이터의 모든 행을 유지합니다.

동일한 조건으로 전체 외부 조인을 사용하여 샘플 테이블을 조인하면 결과 집합에 다음 데이터가 포함됩니다.

PARTED	DESCRIPTION	SIZE	Color
1	Seat Cover	Large	Blue
2	Ash Tray	Small	NULL
3	Floor Mat	Medium	Black
4	Fuzzy Dice	NULL	Yellow

Ash Tray에 색상이 지정되지 않았고 Fuzzy Dice에 크기가 지정되지 않았기 때문에 데이터 통합 서비스가 이러한 필드를 NULL로 채웁니다.

다음 예는 동일한 SQL 문을 보여 줍니다.

```
SELECT * FROM PARTS_SIZE FULL OUTER JOIN PARTS_COLOR ON (PARTS_SIZE.PART_ID1 = PARTS_COLOR.PART_ID2)
```

## 조이너 변환의 정렬된 입력

정렬된 입력 옵션을 사용하면 조이너 변환 성능을 향상시킬 수 있습니다. 데이터가 정렬된 경우 정렬된 입력을 사용합니다.

정렬된 데이터를 사용하도록 조이너 변환을 구성하는 경우 데이터 통합 서비스가 디스크 입력 및 출력을 최소화하여 성능을 향상시킵니다. 대규모 데이터 집합으로 작업할 때 가장 크게 성능이 향상되는 것을 볼 수 있습니다.

정렬된 데이터를 사용하도록 매핑을 구성하려면 조이너 변환을 처리할 때 데이터 통합 서비스가 정렬된 데이터를 사용할 수 있도록 매핑에서 정렬 순서를 설정하고 유지 관리합니다. 매핑을 구성하려면 다음 단계를 완료하십시오.

1. 조인하려는 데이터의 정렬 순서를 구성합니다.
2. 정렬된 데이터의 순서를 유지 관리하는 변환을 추가합니다.
3. 정렬된 데이터를 사용하도록 조이너 변환을 구성하고 정렬 출처 포트를 사용하도록 조인 조건을 구성합니다. 정렬 출처는 정렬된 데이터의 소스를 나타냅니다.

## 정렬 순서 구성

데이터 통합 서비스가 정렬된 데이터를 조이너 변환에 전달할 수 있도록 정렬 순서를 구성합니다.

정렬 순서를 구성하려면 다음 방법 중 하나를 사용합니다.

- 정렬된 플랫폼 파일을 사용합니다. 플랫폼 파일에 정렬된 데이터가 포함되는 경우 정렬 열의 순서가 각 소스 파일에서 일치하는지 확인하십시오.
- 정렬된 관계형 데이터를 사용합니다. 관계형 데이터 개체의 정렬된 포트를 사용하여 소스 데이터베이스의 열을 정렬합니다. 정렬된 포트의 순서는 각 관계형 데이터 개체에서 동일하게 구성해야 합니다.
- 분류기 변환을 사용하여 관계형 또는 플랫폼 파일 데이터를 정렬합니다. 분류기 변환은 마스터 및 세부 파이프라인에 배치합니다. 각 분류기 변환이 동일한 순서의 정렬 키 포트 및 정렬 순서 방향을 사용하도록 구성하십시오.

정렬된 데이터를 사용하도록 구성된 조이너 변환에 정렬되지 않거나 잘못 정렬된 데이터를 전달할 경우 매핑 실행이 실패하고 데이터 통합 서비스가 로그 파일에 오류를 기록합니다.

## 매핑에 변환 추가

조이너 변환에서 정렬된 데이터의 순서를 유지하는 매핑에 변환을 추가합니다.

정렬 출처 바로 뒤에 조이너 변환을 배치하면 정렬된 데이터를 유지할 수 있습니다.

정렬 출처와 조이너 변환 사이에 변환을 추가할 때 정렬된 데이터를 유지하려면 다음 지침을 사용하십시오.

- 정렬 출처와 조이너 변환 사이에는 다음 변환을 배치하지 마십시오.
  - 순위
  - 합집합

- 정렬되지 않은 집계
- 위의 변환 중 하나를 포함하는 맵셋
- 정렬 출처와 조이너 변환 사이에 정렬된 집계 변환을 배치하려면 다음 지침을 사용하십시오.
  - 정렬된 입력을 사용하도록 집계 변환을 구성합니다.
  - 집계 변환의 그룹 기준 열에 정렬 출처의 포트와 동일한 포트를 사용합니다.
  - 그룹 기준 포트는 정렬 출처의 포트와 동일한 순서여야 합니다.
- 조이너 변환의 결과 집합을 다른 파이프라인과 조인하는 경우 첫 번째 조이너 변환의 데이터 출력이 정렬되었는지 확인하십시오.

## 조인 조건에 대한 규칙 및 지침

정렬된 조이너 변환에 대한 조인 조건을 작성할 경우 특정한 규칙 및 지침이 적용됩니다.

조인 조건을 작성할 경우 다음 지침을 사용하십시오.

- 같은 연산자(=)를 사용하는 단순 조건 유형을 정의해야 합니다.
- 정렬 출처 및 조이너 변환 간에 정렬된 집계 변환을 사용할 경우 조인 조건을 정의할 때 정렬된 집계 변환을 정렬 출처로 처리하십시오.
- 조인 조건에서 사용하는 포트가 정렬 출처의 포트와 일치해야 합니다.
- 여러 조인 조건을 구성할 경우 첫 번째 조인 조건의 포트가 정렬 출처의 첫 번째 포트와 일치해야 합니다.
- 여러 조건을 구성할 경우 조건 순서가 정렬 출처의 포트 순서와 일치해야 하고 어떤 포트도 건너뛰면 안 됩니다.
- 정렬 출처의 정렬된 포트 수는 조인 조건의 포트 수보다 크거나 같을 수 있습니다.
- 포트를 10진수 데이터 유형과 조인할 경우 각 포트의 전체 자릿수가 동일한 전체 자릿수 범위에 속해야 합니다.

다음 올바른 전체 자릿수 범위 중 하나를 사용할 수 있습니다.

- 10진수 0-18
- 10진수 19-28
- 10진수 29 이상
- 10진수 29-38
- 10진수 39 이상
- 10진수 29-38
- 10진수 39 이상

예를 들어, `DecimalA = DecimalB` 조건을 정의하고 여기서 `DecimalA`의 전체 자릿수는 15이고 `DecimalB`의 전체 자릿수는 25일 경우 조건이 올바르지 않습니다.

## 조인 조건 및 정렬 순서의 예제

이 예에서는 정렬된 포트를 사용하여 마스터 파이프라인과 세부 파이프라인을 조인하는 조이너 변환을 보여 줍니다.

다음과 같은 정렬된 포트를 사용하여 마스터 파이프라인과 세부 파이프라인에 분류기 변환을 구성하십시오.

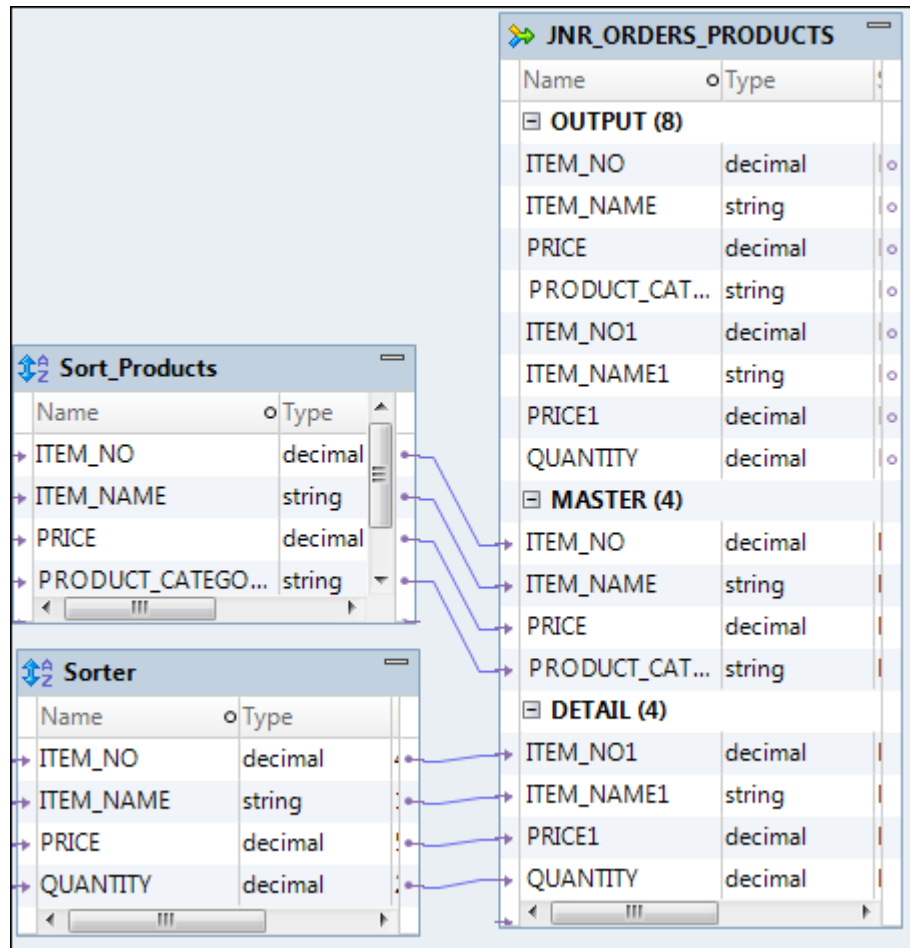
- ITEM\_NO
- ITEM\_NAME
- PRICE

조인 조건을 구성할 때는 다음 지침을 사용하여 정렬 순서를 유지합니다.

- ITEM\_NO를 첫 번째 조인 조건에 사용해야 합니다.
- 두 번째 조인 조건을 추가하는 경우 ITEM\_NAME을 사용해야 합니다.
- PRICE를 사용하려는 경우 ITEM\_NAME도 두 번째 조인 조건에 사용해야 합니다.

ITEM\_NAME을 건너뛰고 ITEM\_NO와 PRICE에 조인을 수행하면 정렬 순서가 유지되지 않고 데이터 통합 서비스에서 매핑 실행이 실패합니다.

다음 그림은 ITEM\_NO, ITEM\_NAME 및 PRICE 포트에서 정렬 및 조인을 수행하도록 구성된 매핑을 보여 줍니다.



조이너 변환을 사용하여 마스터 파이프라인과 세부 파이프라인을 조인하는 경우 다음 조인 조건 중 하나를 구성할 수 있습니다.

ITEM\_NO = ITEM\_NO

또는

ITEM\_NO = ITEM\_NO1

ITEM\_NAME = ITEM\_NAME1

또는

ITEM\_NO = ITEM\_NO1

ITEM\_NAME = ITEM\_NAME1

PRICE = PRICE1



## 동일한 소스의 데이터 조인

일부 데이터에 계산을 수행한 다음 변환된 데이터를 원래 데이터와 조인하려는 경우와 같이 동일한 소스의 데이터를 조인할 수 있습니다.

동일한 소스의 데이터를 조인하는 경우 하나의 매핑 내에서 원래 데이터를 유지하고 해당 데이터의 일부를 변환할 수 있습니다. 동일한 소스의 데이터는 다음과 같이 조인할 수 있습니다.

- 동일한 파이프라인의 분기 2개를 조인합니다.
- 동일한 소스의 인스턴스 2개를 조인합니다.

### 동일한 파이프라인의 분기 2개 조인

동일한 소스의 데이터를 조인할 경우 파이프라인의 분기 2개를 작성할 수 있습니다.

파이프라인의 분기를 만드는 경우 최소 1개의 파이프라인 분기에서 매핑 입력과 조이너 변환 사이에 변환을 추가해야 합니다. 정렬된 데이터를 조인하고 조이너 변환을 정렬된 입력에 대해 구성해야 합니다.

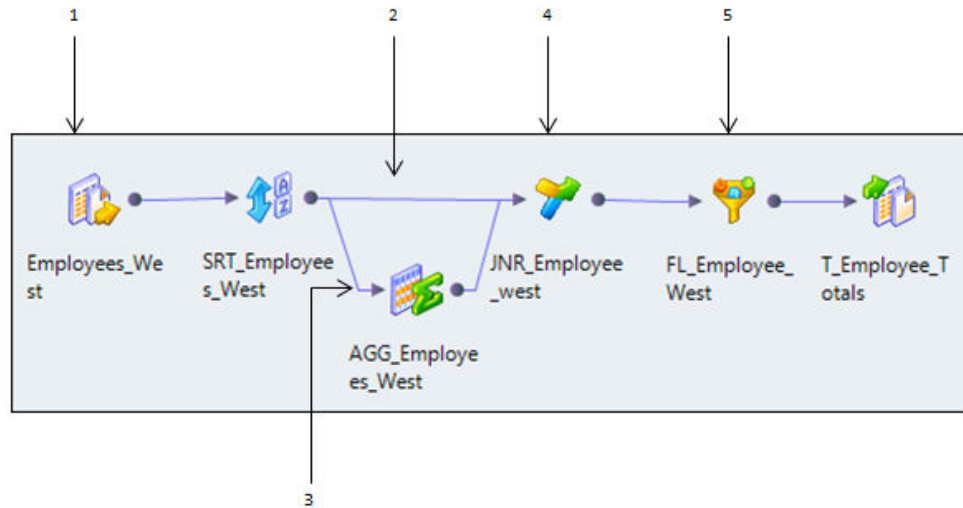
예를 들어 다음 포트를 포함하는 소스가 있습니다.

- 직원
- 부서
- 총 매출액

대상에는 부서의 평균 매출액을 초과하는 매출액을 달성한 직원을 표시하려고 합니다. 그러려면 다음 변환을 포함하는 매핑을 작성해야 합니다.

- 분류기 변환. 데이터를 정렬합니다.
- 정렬된 집계 변환. 매출액 데이터의 평균값을 계산하고 부서를 기준으로 그룹화합니다. 이 집계를 수행하면 개별 직원에 대한 데이터가 손실됩니다. 직원 데이터를 유지하려면 파이프라인의 분기를 집계 변환에 전달하고 동일한 데이터가 포함된 분기를 조이너 변환에 전달하여 원래 데이터를 유지해야 합니다. 파이프라인의 두 분기를 조인할 때는 집계된 데이터를 원래 데이터와 조인합니다.
- 정렬된 조이너 변환. 정렬된 집계 데이터를 원래 데이터와 조인합니다.

- 필터 변환. 평균 매출액 데이터를 각 직원의 매출액 데이터와 비교한 다음 평균 매출액 이하의 매출액을 달성한 직원을 필터링하여 제외합니다.



1. Employees\_West 소스
2. 파이프라인 분기 1
3. 파이프라인 분기 2
4. 정렬된 조이너 변환
5. 평균 매출액 이하의 매출액을 달성한 직원을 제외하는 필터링

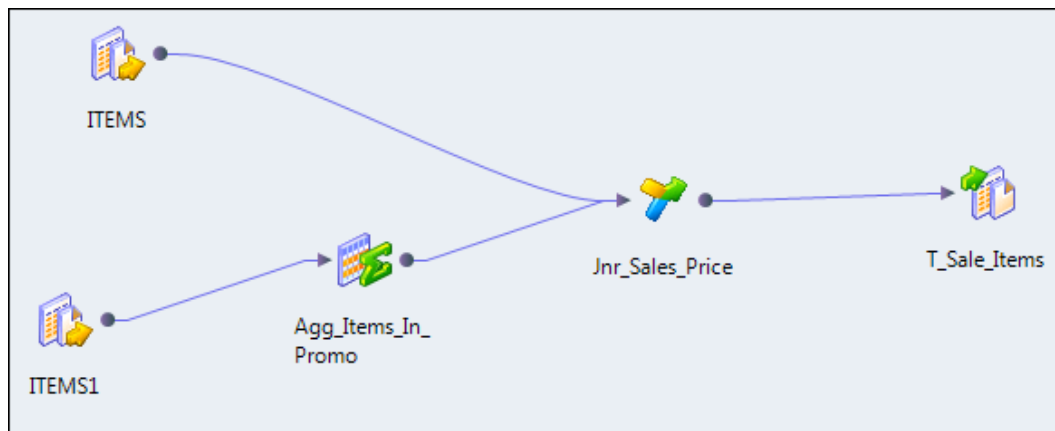
두 분기를 조인할 때 조이너 변환이 각 분기에서 데이터를 수신하는 간격이 길어지면 성능이 저하될 수 있습니다. 조이너 변환은 첫 번째 분기의 모든 데이터를 캐시하고 캐시가 채워지면 디스크에 해당 캐시를 기록합니다. 그런 다음 두 번째 분기의 데이터를 수신할 때 조이너 변환은 해당 데이터를 디스크에서 읽어야 합니다.

## 동일한 소스의 인스턴스 2개 조인

소스의 두 번째 인스턴스를 작성하여 동일한 소스의 데이터를 조인할 수 있습니다.

두 번째 소스 인스턴스를 작성한 후에 두 소스 인스턴스의 파이프라인을 조인할 수 있습니다. 정렬되지 않은 데이터를 조인하려는 경우 동일한 소스의 인스턴스 2개를 작성하고 파이프라인을 조인해야 합니다.

다음 그림은 조이너 변환을 통해 조인된 동일한 소스의 인스턴스 2개를 보여 줍니다.



동일한 소스의 인스턴스 2개를 조인하면 데이터 통합 서비스가 각 소스 인스턴스에 대한 소스 데이터를 읽습니다. 따라서 파이프라인의 분기 2개를 조인하는 것보다 성능이 느려질 수 있습니다.

## 동일한 소스의 데이터 조인 지침

파이프라인의 분기를 조인할지 소스의 인스턴스 2개를 조인할지를 결정할 때는 특정 지침이 적용됩니다.

다음 지침을 바탕으로 파이프라인의 분기를 조인할지 소스의 인스턴스 2개를 조인할지를 결정하십시오.

- 소스가 크거나 소스 데이터를 한 번만 읽을 수 있는 경우 파이프라인의 분기 2개를 조인합니다.
- 정렬된 데이터를 사용하는 경우 파이프라인의 분기 2개를 조인합니다. 소스 데이터가 정렬되지 않아 분류기 변환을 사용하여 데이터를 정렬하는 경우 데이터를 정렬한 후에 파이프라인의 분기를 만듭니다.
- 파이프라인에서 소스와 조이너 변환 사이에 차단 변환을 추가해야 하는 경우 소스의 인스턴스 2개를 조인합니다.
- 한 파이프라인이 다른 파이프라인보다 느리게 처리되는 경우 소스의 인스턴스 2개를 조인합니다.
- 정렬되지 않은 데이터를 조인해야 하는 경우 소스의 인스턴스 2개를 조인합니다.

## 소스 파이프라인 차단

조이너 변환을 포함하는 매핑을 실행할 때 데이터 통합 서비스는 매핑 구성 및 정렬된 입력에 대한 조이너 변환 구성에 따라 소스 데이터를 차단하고 차단 해제합니다.

### 정렬되지 않은 조이너 변환

데이터 통합 서비스가 정렬되지 않은 조이너 변환을 처리하는 경우 세부 정보 행을 읽기 전에 모든 마스터 행을 읽습니다. 데이터 통합 서비스는 마스터 소스에서 행을 캐시하는 동안 세부 소스를 차단합니다.

데이터 통합 서비스가 모든 마스터 행을 읽고 캐시한 후 세부 소스의 차단을 해제하고 세부 정보 행을 읽습니다. 정렬되지 않은 조이너 변환이 포함된 일부 매핑은 데이터 흐름 유효성 검사에 위반됩니다.

### 정렬된 조이너 변환

데이터 통합 서비스는 정렬된 조이너 변환을 처리할 때 매핑 구성에 따라 데이터를 차단합니다. 조이너 변환에 대한 마스터 및 세부 입력이 서로 다른 소스에서 생성되는 경우 차단 논리를 사용할 수 있습니다.

데이터 통합 서비스는 대상 로드 순서 그룹의 모든 소스를 동시에 차단하지 않고도 조이너 변환을 처리할 수 있으면 차단 논리를 사용하여 해당 변환을 처리합니다. 그렇지 않으면 차단 논리를 사용하지 않습니다. 대신 캐시에 더 많은 행을 저장합니다.

데이터 통합 서비스는 차단 논리를 사용하여 조이너 변환을 처리할 수 있으면 캐시에 더 적은 행을 저장하므로 성능이 개선됩니다.

### 마스터 행 캐싱

데이터 통합 서비스는 조이너 변환을 처리할 때 두 소스의 행을 동시에 읽고 마스터 행을 기반으로 인덱스 및 데이터 캐시를 빌드합니다.

그런 다음 세부 소스 데이터 및 캐시 데이터를 바탕으로 조인을 수행합니다. 데이터 통합 서비스가 캐시에 저장하는 행의 수는 소스 데이터 및 조이너 변환의 정렬된 입력 구성에 따라 다릅니다.

정렬되지 않은 조이너 변환의 성능을 높이려면 행 수가 더 적은 소스를 마스터 소스로 지정하십시오. 정렬된 조이너 변환의 성능을 높이려면 중복 키 값이 더 적은 소스를 마스터로 지정하십시오.

## 조이너 변환 성능 팁

팁을 사용하여 조이너 변환의 성능을 향상시킬 수 있습니다.

조이너 변환은 런타임 시 중간 결과를 유지하기 위해 추가 공간을 필요로 하므로 성능을 저하시킬 수 있습니다. 조이너 성능 카운터 정보를 확인하여 조이너 변환에 최적화가 필요한지 여부를 결정할 수 있습니다.

다음 팁을 사용하여 조이너 변환의 성능을 개선하십시오.

### 마스터 소스를 중복 키 값이 더 적은 소스로 지정합니다.

데이터 통합 서비스가 정렬된 조이너 변환을 처리할 때 한 번에 백 개의 고유 키에 대한 행을 캐싱합니다. 마스터 소스에 키 값이 같은 행이 많이 포함된 경우 데이터 통합 서비스가 더 많은 행을 캐싱해야 하므로 성능이 저하될 수 있습니다.

### 마스터 소스를 행이 더 적은 소스로 지정합니다.

조이너 변환이 마스터 소스에 대해 세부 소스의 각 행을 비교합니다. 마스터의 행 수가 적을수록 조인 비교 횟수가 줄어들고 조인 프로세스가 빨라집니다.

### 가능한 경우 데이터베이스에서 조인을 수행합니다.

데이터베이스에서 조인을 수행하는 것이 매핑 실행 중에 조인을 수행하는 것보다 빠릅니다. 사용하는 데이터베이스 조인 유형이 성능에 영향을 미칠 수 있습니다. 일반 조인이 외부 조인보다 빠르며 결과 행 수도 적습니다. 서로 다른 두 데이터베이스나 플랫폼 파일 시스템의 테이블을 조인하는 경우와 같이, 데이터베이스에서 조인을 수행할 수 없는 경우도 있습니다.

### 가능한 경우 정렬된 데이터를 조인하십시오.

정렬된 입력을 사용하도록 조이너 변환을 구성합니다. 데이터 통합 서비스는 디스크 입력과 디스크 출력을 최소화하여 성능을 향상시킵니다. 대규모 데이터 집합으로 작업할 때 가장 큰 성능 향상을 얻을 수 있습니다. 비정렬 조이너 변환의 경우 행 수가 더 적은 소스를 마스터 소스로 지정하십시오.

### 조인 조건을 최적화합니다.

데이터 통합 서비스는 작은 그룹에서 행을 읽고 큰 그룹에서 일치하는 행을 찾는 다음 조인 작업을 수행하는 방식으로 한 조인 피연산자의 데이터 집합 크기를 줄입니다. 데이터 집합의 크기를 줄이면 데이터 통합 서비스가 더 이상 큰 그룹 소스에서 불필요한 행을 읽지 않게 되므로 매핑 성능이 향상됩니다. 데이터 통합 서비스는 조인 조건을 큰 그룹 소스로 이동하고 작은 그룹과 일치하는 행만 읽습니다.

### 반 조인 최적화 방법을 사용합니다.

한 입력 그룹에 다른 입력 그룹보다 훨씬 많은 행이 있으며 조인 조건을 기준으로 큰 그룹의 많은 행이 작은 그룹의 행과 일치하지 않는 경우 반 조인 최적화 방법을 사용하여 매핑 성능을 향상시킵니다.

## 조이너 변환에 대한 규칙 및 지침

조이너 변환을 사용할 때는 특정 규칙과 지침이 적용됩니다.

조이너 변환은 대다수 변환의 입력을 허용합니다. 그러나 입력 파이프라인 중 하나가 업데이트 전략 변환을 포함할 때는 조이너 변환을 사용할 수 없습니다.

## 조이너 변환 - 비원시 환경

비원시 환경에서 조이너 변환의 처리는 변환을 실행하는 엔진에 따라 다릅니다.

다음과 같은 비원시 런타임 엔진에 대한 지원을 고려하십시오.

- **Blaze** 엔진. 제한적으로 지원됩니다.
- **Spark** 엔진. 제한적으로 지원됩니다.
- **Databricks Spark** 엔진. 제한적으로 지원됩니다.

### 조이너 변환 - Blaze 엔진

다음과 같은 상황에서는 매핑 유효성 검사가 실패합니다.

- 변환에 같지 않은 조인이 포함되어 있고 맵 측면 조인이 비활성화되어 있습니다.
- 조이너 변환 식이 연결되지 않은 조회 변환을 참조합니다.

조이너 변환에 세부 외부 조인 또는 전체 외부 조인이 구성되어 있고 맵 측면 조인이 비활성화되어 있습니다.

### 조이너 변환 - Spark 엔진

다음과 같은 상황에서는 매핑 유효성 검사가 실패합니다.

- 대/소문자 구분이 비활성화되어 있습니다.
- 조인 조건의 데이터 유형이 이진이거나 조인 조건에 바이너리 식이 포함되어 있습니다.

### 조이너 변환 - Databricks Spark 엔진

다음과 같은 상황에서는 매핑 유효성 검사가 실패합니다.

- 대/소문자 구분이 비활성화되어 있습니다.
- 조인 조건의 데이터 유형이 이진이거나 조인 조건에 바이너리 식이 포함되어 있습니다.

## 제 23 장

# 키 생성기 변환

이 장에 포함된 항목:

- [키 생성기 변환 개요, 354](#)
- [Soundex 전략, 355](#)
- [문자열 전략, 355](#)
- [NYSIIS 전략, 356](#)
- [키 생성기 출력 포트, 356](#)
- [그룹화 전략 구성, 357](#)
- [키 작성 속성, 357](#)
- [키 생성기 변환 고급 속성, 358](#)
- [키 생성기 변환 - 비원시 환경, 358](#)

## 키 생성기 변환 개요

키 생성기 변환은 사용자가 선택한 열의 데이터 값을 기반으로 레코드를 그룹으로 구성하는 활성 변환입니다. 레코드를 일치 변환에 전달하기 전에 이 변환을 사용하여 레코드를 정렬하십시오.

키 생성기 변환에서 그룹화 전략을 사용하여 선택된 열에 대한 그룹 키를 작성합니다. 전략은 문자열, **Soundex** 및 **NYSIIS**입니다. 선택된 필드에 공통 값이 있는 레코드는 공통 그룹 키 값을 갖습니다. 일치 변환에서는 공통 그룹 키 값을 가진 레코드를 함께 처리합니다. 이렇게 하면 일치 변환에서 중복 분석을 더 빠르게 실행할 수 있습니다.

일치 변환에서 수행해야 하는 비교 작업 수는 데이터 집합의 레코드 개수에 따라 기하급수적으로 증가합니다. 이렇게 기하급수적으로 증가할 경우 많은 계산 리소스를 소모할 수 있습니다. 그룹 키를 작성하면 키 생성기 변환을 통해 일치 변환에서 레코드를 더 작은 그룹으로 비교할 수 있게 되고 처리 시간이 줄어들 수 있습니다.

필드 일치를 수행할 경우 그룹 키를 생성하기 위한 열을 선택합니다. 그러면 일치에 필요한 유용한 그룹을 제공할 수 있습니다. 예를 들어, 성 열에서 이름 열보다 더 의미 있는 그룹 키 데이터를 제공할 수 있습니다. 그러나 일치 변환에서 중복 분석을 위해 해당 열을 선택할 경우에는 성 열을 사용하지 마십시오.

키 생성기 변환에서 각 레코드에 대해 고유한 ID를 작성할 수도 있습니다. 일치 변환에 들어가는 레코드마다 고유한 ID가 있어야 합니다. 데이터에 대한 ID가 없는 경우 키 생성기 변환을 사용하여 ID를 작성하십시오.

# Soundex 전략

Soundex 전략은 단어를 분석하고 단어 발음을 나타내는 코드를 사용하여 그룹 키를 생성합니다.

Soundex 코드는 단어의 첫 번째 문자로 시작하고 연속 자음을 나타내는 일련의 숫자가 뒤에 옵니다. Soundex 전략을 사용하여 동일한 코드를 유사하게 소리나는 단어에 할당합니다. 전략이 반환하는 영숫자 문자 수를 정의하도록 Soundex 깊이를 구성합니다.

이 전략은 철자 대신 단어의 발음에 초점을 맞추기 때문에 대체 철자 및 사소한 철자 변화를 그룹화할 수 있습니다. 예를 들어 Smyth와 Smith의 Soundex 코드는 동일합니다.

또한 Soundex 전략은 잘못 발음한 단어를 그룹화할 수 있습니다. 예를 들어 이름이 Edmonton과 Edmonson인 Soundex 코드는 동일합니다.

## Soundex 전략 속성

Soundex 전략 속성을 구성하여 키 생성기 변환이 그룹 키를 생성하는 데 사용하는 Soundex 설정을 결정합니다.

다음 테이블에는 Soundex 전략 속성이 설명되어 있습니다.

속성	설명
Soundex 깊이	Soundex 전략에서 반환하는 영숫자 문자의 수를 결정합니다. 기본 깊이는 3입니다. 이 깊이를 사용하는 경우 문자열의 첫 번째 문자와 다음 두 고유 자음을 나타내는 2개의 숫자로 구성된 Soundex 코드가 생성됩니다.

관련 항목:

- [“문자열 전략 속성” 페이지 356](#)
- [“키 작성 속성” 페이지 357](#)
- [“그룹화 전략 구성” 페이지 357](#)

## 문자열 전략

문자열 전략은 입력 데이터의 하위 문자열에서 그룹 키를 작성합니다.

입력 열 내에서 하위 문자열의 길이 및 위치를 지정할 수 있습니다. 예를 들어 입력 문자열의 처음 네 문자에서 키를 작성하도록 이 전략을 구성할 수 있습니다.

## 문자열 전략 속성

문자열 전략 속성을 구성하여 키 생성기 변환이 그룹 키를 생성하는 데 사용하는 하위 문자열을 결정합니다. 다음 테이블에는 문자열 전략 속성이 설명되어 있습니다.

속성	설명
왼쪽에서 시작	입력 필드를 왼쪽에서 오른쪽으로 읽도록 변환을 구성합니다.
오른쪽에서 시작	입력 필드를 오른쪽에서 왼쪽으로 읽도록 변환을 구성합니다.
시작 위치	건너뛰길 문자 수를 지정합니다. 예를 들어 <b>시작 위치</b> 로 3을 입력하면 하위 문자열이 입력 필드의 지정한 쪽에서 4번째 문자부터 시작됩니다.
길이	그룹 키로 사용할 문자열 길이를 지정합니다. 전체 입력 필드를 사용하려면 0을 입력합니다.

관련 항목:

- [“Soundex 전략 속성” 페이지 355](#)
- [“키 작성 속성” 페이지 357](#)
- [“그룹화 전략 구성” 페이지 357](#)

## NYSIIS 전략

NYSIIS 전략이 단어를 분석하고 단어 발음을 나타내는 문자를 사용하여 그룹 키를 생성합니다.

Soundex 전략이 문자열에서 첫 번째 모음만 고려하는 반면 NYSIIS 전략은 문자열 전체에서 모음을 분석합니다. NYSIIS 전략은 6개 문자 중 하나로 모든 문자를 변환하고 대부분의 모음을 문자 A로 변환합니다.

## 키 생성기 출력 포트

키 생성기 변환 출력 포트는 일치 변환에서 레코드를 처리하는 데 사용하는 ID와 그룹 키를 생성합니다.

다음 테이블에서는 키 생성기 변환의 출력 포트에 대해 설명합니다.

속성	설명
SequenceID	소스 데이터 집합의 각 레코드를 식별하는 ID를 생성합니다.
GroupKey	일치 변환이 레코드를 처리하는 데 사용하는 그룹 키를 생성합니다.

재사용 가능 키 생성기 변환을 생성할 때는 **개요** 보기를 통해 포트를 확인합니다. 재사용 불가능 변환을 매핑에 추가할 때는 **속성** 보기의 **포트** 탭을 통해 포트를 확인합니다.



## 그룹화 전략 구성

그룹화 전략을 구성하려면 **전략** 보기에서 속성을 편집하십시오.

키 생성기 전략을 구성하기 전에 입력 포트를 키 생성기 변환에 추가합니다.

1. **전략** 보기를 선택합니다.
2. **새로 만들기** 단추를 클릭합니다.
3. 그룹화 전략을 선택합니다.
4. **확인**을 클릭합니다.
5. **입력** 열에서 입력 포트를 선택합니다.
6. 속성 필드에서 선택 화살표를 클릭하여 전략 속성을 구성합니다.
7. 키 작성 속성을 구성합니다.

관련 항목:

- [“Soundex 전략 속성” 페이지 355](#)
- [“문자열 전략 속성” 페이지 356](#)
- [“키 작성 속성” 페이지 357](#)

## 키 작성 속성

분석하는 데이터에 해당하는 키 작성 속성을 구성합니다.

다음 표에는 키 작성 속성이 설명되어 있습니다.

속성	설명
결과 정렬	GroupKey 필드를 사용하여 키 생성기 변환 출력을 정렬합니다. 필드 일치 작업의 경우 이 옵션을 선택하거나 일치 변환에 정렬된 데이터를 제공하는지 확인해야 합니다. ID 일치 작업의 경우 이 옵션을 선택하지 마십시오.
자동으로 시퀀스 키 생성	입력 데이터의 순서를 사용하여 시퀀스 키 필드를 생성합니다.
필드를 시퀀스 키로 사용	지정한 열에 대한 시퀀스 필드를 생성합니다.
시퀀스 키 필드	시퀀스 키 필드의 이름을 지정합니다.

관련 항목:

- [“Soundex 전략 속성” 페이지 355](#)
- [“문자열 전략 속성” 페이지 356](#)
- [“그룹화 전략 구성” 페이지 357](#)

## 키 생성기 변환 고급 속성

키 생성기 변환에는 캐시 메모리 동작 및 추적 수준을 결정하는 고급 속성이 포함됩니다.

다음과 같은 고급 속성을 구성할 수 있습니다.

### 캐시 파일 디렉터리

데이터 통합 서비스에서 현재 변환에 대해 임시 데이터를 쓸 디렉터리를 지정합니다. 입력 데이터의 양이 사용 가능한 시스템 메모리보다 클 경우 데이터 통합 서비스가 이 디렉터리에 임시 파일을 기록합니다. 데이터 통합 서비스는 매핑을 실행한 후에 임시 파일을 삭제합니다.

속성에 디렉터리 경로를 입력하거나 매개 변수를 사용하여 디렉터리를 식별할 수 있습니다. 데이터 통합 서비스 시스템의 로컬 경로를 지정하십시오. 데이터 통합 서비스가 기록할 수 있는 디렉터리를 지정해야 합니다. 기본값은 **CacheDir** 시스템 매개 변수입니다.

### 캐시 파일 크기

변환의 입력 데이터를 정렬하기 위해 데이터 통합 서비스에서 사용하는 시스템 메모리의 양을 결정합니다. 매개 변수를 사용하여 캐시 파일 크기를 지정할 수 있습니다.

데이터 통합 서비스는 데이터를 정렬하기 전에 사용자가 지정한 메모리 양을 할당합니다. 정렬 작업이 더 많은 데이터를 생성할 경우 데이터 통합 서비스는 초과 데이터를 캐시 디렉터리에 씁니다. 정렬 작업에 시스템 메모리 및 파일 저장소가 제공할 수 있는 것보다 더 많은 메모리가 필요할 경우 매핑이 실패합니다.

캐시 파일 크기를 지정하지 않으면 변환이 데이터 통합 서비스 실행 옵션의 최대 메모리 값을 적용합니다.

**참고:** 65536 이상의 값을 입력한 경우 일치 변환이 값을 바이트로 읽습니다. 이보다 낮은 값을 입력하면 일치 변환이 값을 메가바이트로 읽습니다.

### 추적 수준

이 변환에 대해 로그에 표시되는 세부 정보의 양입니다. 간단, 보통, 자세한 정보 표시 초기화 또는 자세한 정보 표시 데이터를 선택할 수 있습니다. 기본값은 보통입니다.

## 키 생성기 변환 - 비원시 환경

비원시 환경에서 키 생성기 변환의 처리는 변환을 실행하는 엔진에 따라 다릅니다.

다음과 같은 비원시 런타임 엔진에 대한 지원을 고려하십시오.

- Blaze 엔진. 제한 없이 지원됩니다.
- Spark 엔진. 제한 없이 지원됩니다.
- Databricks Spark 엔진. 지원되지 않습니다.

## 제 24 장

# 라벨러 변환

이 장에 포함된 항목:

- [라벨러 변환 개요, 359](#)
- [라벨러 변환 사용 시기, 360](#)
- [라벨러 변환의 참조 데이터 사용, 361](#)
- [라벨러 변환 전략, 362](#)
- [라벨러 변환 포트, 363](#)
- [문자 레이블 지정 속성, 364](#)
- [토큰 레이블 지정 속성, 366](#)
- [문자 레이블 지정 전략 구성, 368](#)
- [토큰 레이블 지정 전략 구성, 369](#)
- [라벨러 변환의 고급 속성, 369](#)
- [라벨러 변환 - 비원시 환경, 370](#)

## 라벨러 변환 개요

라벨러 변환은 입력 포트 필드를 분석하고 각 필드의 데이터를 설명하는 텍스트 레이블을 작성하는 수동 변환입니다.

포트에 포함된 정보 유형을 파악하고자 할 경우 라벨러 변환을 사용하십시오. 포트의 정보 유형을 모를 경우 또는 포트에 예상한 정보 유형이 포함되지 않은 레코드를 식별하고자 할 경우 라벨러 변환을 사용하십시오.

레이블은 입력 문자열을 설명하는 하나 이상의 문자가 포함된 문자열입니다. 문자열에 포함된 데이터를 기반으로 입력 문자열에 레이블을 할당하도록 라벨러 변환을 구성하십시오.

변환을 구성한 경우 검색할 문자 또는 문자열의 유형을 지정하고, 변환에서 연결된 문자 또는 문자열을 찾으려면 출력으로 작성하는 레이블을 지정하십시오. 레이블 지정 작업을 구성할 때 검색할 문자 및 문자열 유형과 사용할 레이블을 입력하십시오. 또는 참조 데이터 개체를 사용하여 문자, 문자열 및 레이블을 지정합니다.

문자 레이블 지정 또는 토큰 레이블 지정을 수행하도록 변환을 구성하십시오.

### 문자 레이블 지정

마침표 및 공백을 포함하여 입력 문자열의 문자 구조를 설명하는 레이블을 작성합니다. 변환에서 열의 각 행에 대한 단일 레이블을 작성합니다. 예를 들어, 라벨러 변환에서 우편 번호 10028을 "nnnnn"으로 레이블 지정할 수 있습니다. 여기서 "n"은 숫자 문자를 나타냅니다.

## 토큰 레이블 지정

입력 문자열의 정보 유형을 설명하는 레이블을 작성합니다. 변환에서 입력 데이터에 식별된 각 토큰에 대한 레이블을 작성합니다. 예를 들어, "John J. Smith" 문자열을 "Word Init Word" 토큰을 사용하여 레이블 지정하도록 라벨러 변환을 구성할 수 있습니다.

토큰은 입력 문자열의 구분된 값입니다.

라벨러가 사용자가 지정한 레이블과 일치하는 문자 또는 문자열을 찾은 경우 레이블을 이름을 새 출력 포트에 기록합니다.

라벨러 변환에서 참조 데이터를 사용하여 문자 및 토큰을 식별합니다. 라벨러 전략에서 작업을 구성한 경우 참조 데이터 개체를 선택하십시오.

# 라벨러 변환 사용 시기

라벨러 변환은 포트의 각 값에 대한 설명 레이블을 씁니다.

다음 예는 라벨러 변환을 사용하여 수행할 수 있는 일부 분석 유형을 설명합니다.

## 연락처 데이터가 포함된 레코드 찾기

이름 목록이 포함된 참조 테이블을 사용하여 변환을 구성합니다. 토큰 레이블 지정 전략을 작성하여 참조 테이블의 값과 일치하는 문자열에 레이블을 지정합니다. 출력 데이터를 검토할 때 해당 레이블이 포함된 레코드는 사람을 나타낼 가능성이 있습니다.

## 비즈니스 레코드 찾기

Inc, Corp 및 Ltd와 같은 비즈니스 접미사 목록이 포함된 토큰 집합을 사용하여 변환을 구성합니다. 토큰 레이블 지정 전략을 작성하여 참조 테이블의 값과 일치하는 문자열에 레이블을 지정합니다. 출력 데이터를 검토할 때 해당 레이블이 포함된 레코드는 비즈니스를 나타낼 가능성이 있습니다.

**참고:** 비즈니스 이름을 식별하려는 비즈니스 접미사의 토큰 집합을 사용합니다. 테이블에 식별하려는 모든 비즈니스가 포함되어 있다고 확인하는 경우 비즈니스 이름의 참조 테이블을 사용할 수 있습니다. 예를 들어 뉴욕 증권 거래소의 회사가 나열된 참조 테이블을 사용할 수 있습니다.

## 전화 번호 데이터 찾기

전화 번호의 문자 구조를 정의하는 문자 집합을 사용하여 변환을 구성합니다. 예를 들어 미국 전화 번호와 같이 여러 패턴의 구두점 기호 및 숫자를 인식하는 문자 집합을 사용할 수 있습니다. 데이터를 검토하여 전화 번호에 대해 올바른 숫자가 포함되지 않은 레코드를 찾을 수 있습니다.

열 데이터를 분석하기 위해 문자 레이블에 다음 문자를 사용할 수 있습니다.

c=punctuation character n=digit s=space

다음 테이블은 동일한 전화 번호 구조를 보여줍니다.

문자 구조	전화 번호
cnnncsnnnncnnnnnnnn	(212) 555-1212
nnnnnnnnnnnn	2125551212
cnnncnnnncnnnn	+212-555-1212

# 라벨러 변환의 참조 데이터 사용

Informatica Developer를 설치할 때는 라벨러 변환에서 사용할 수 있는 여러 유형의 참조 데이터 개체가 함께 설치됩니다. 참조 데이터 개체를 생성할 수도 있습니다.

라벨러 변환 전략에 참조 데이터 개체를 추가할 때 변환은 전략의 입력 데이터에서 참조 데이터 개체의 값을 검색합니다. 그리고 발견된 값을 참조 데이터 개체의 유효한 값이나 지정한 값으로 바꿉니다.

다음 테이블에는 사용할 수 있는 참조 데이터 유형이 설명되어 있습니다.

참조 데이터 유형	설명
문자 집합	문자, 숫자, 구두점 기호 등의 여러 문자 유형을 식별합니다. 문자 레이블 지정 작업에서 사용됩니다.
확률 모델	토큰 레이블 작업에 유사 항목 일치 기능을 추가합니다. 변환은 확률 모델을 사용하여 문자열의 정보 유형을 유추할 수 있습니다. 유사 항목 일치 기능을 활성화하려면 Developer tool에서 확률 모델을 컴파일합니다. 토큰 레이블 지정 작업에서 사용됩니다.
참조 테이블	데이터베이스 테이블의 항목과 일치하는 문자열을 찾습니다. 토큰 레이블 지정 및 문자 레이블 지정 작업에서 사용됩니다.
정규식	정의하는 조건과 일치하는 문자열을 식별합니다. 정규식을 사용하면 더 큰 문자열에서 문자열을 찾을 수 있습니다. 토큰 레이블 지정 작업에서 사용됩니다.
토큰 집합	포함된 정보 유형에 따라 문자열을 식별합니다. 토큰 레이블 지정 작업에서 사용됩니다. Informatica는 단어, 전화 번호, 우편 번호 및 제품 코드 정의와 같이 여러 유형의 토큰 정의를 가진 토큰 집합과 함께 설치됩니다.

## 문자 집합

문자 집합에는 특정 문자 및 문자 범위를 식별하는 식이 포함됩니다. 라벨러 변환 및 토큰 구문 분석 모드를 사용하는 파서 변환에서 문자 집합을 사용할 수 있습니다.

문자 범위는 순차적 문자 코드 범위를 지정합니다. 예를 들어 문자 범위 "[A-C]"는 대문자 "A", "B" 및 "C"와 일치합니다. 이 문자 범위는 소문자 "a", "b" 또는 "c"와 일치하지 않습니다.

문자 집합을 사용하여 토큰 구문 분석 및 레이블 지정 작업의 일부로 특정 문자 또는 문자 범위를 식별할 수 있습니다. 예를 들어 전화 번호가 포함된 열에서 모든 숫자에 레이블을 지정할 수 있습니다. 숫자에 레이블을 지정한 후 파서 변환에서 패턴을 식별하여 문제가 있는 패턴을 별도의 출력에 쓸 수 있습니다.

## 확률 모델

확률 모델은 포함된 정보 유형 및 입력 문자열에서 차지하는 위치에 따라 토큰을 식별합니다.

확률 모델에는 참조 데이터 값과 레이블 값이 포함됩니다. 참조 데이터 값은 변환에 연결하는 입력 포트의 데이터를 나타냅니다. 레이블 값은 참조 데이터 값에 포함되는 정보의 유형을 설명합니다. 사용자는 모델의 각 참조 데이터 값에 레이블을 할당합니다.

확률 모델에서 참조 데이터 값을 레이블에 연결하려면 모델을 컴파일합니다. 컴파일 프로세스에서는 데이터 값과 레이블 간의 일련의 논리적 연관이 생성됩니다. 모델을 읽는 매핑을 실행하면 데이터 통합 서비스가 모델 논리를 변환 입력 데이터에 적용합니다. 데이터 통합 서비스는 입력 데이터 값을 가장 정확히 설명하는 레이블을 반환합니다.

Developer tool에서 확률 모델을 작성할 수 있습니다. 모델 리포지토리는 확률 모델 개체를 저장합니다. Developer tool은 데이터 값, 레이블 및 컴파일 데이터를 Informatica 디렉터리 구조의 파일에 씁니다.

## 참조 테이블

참조 테이블은 두 개 이상의 열이 포함된 데이터베이스 테이블입니다. 한 열에는 표준 또는 필요한 버전의 데이터 값이 들어 있고 다른 열에는 대체 버전 값이 들어 있습니다. 참조 테이블을 변환에 추가하면 변환에서 입력 포트 데이터를 검색하여 참조 테이블에도 나타나는 값을 찾습니다. 작업하는 데이터 프로젝트에 유용한 모든 데이터를 사용하여 테이블을 작성할 수 있습니다.

## 정규식

레이블 지정 작업의 컨텍스트에서 정규식은 입력 데이터의 특정 문자열을 식별하는 데 사용할 수 있는 식입니다. 토큰 레이블 지정 모드를 사용하는 라벨러 변환에서 정규식을 사용할 수 있습니다.

라벨러 변환에서는 정규식을 사용하여 입력 패턴을 일치시키고 단일 레이블을 작성합니다. 출력이 여러 개인 정규식이 여러 레이블을 생성하지는 않습니다.

## 토큰 집합

토큰 집합에는 특정 토큰을 식별하는 식이 포함됩니다. 토큰 레이블 지정 모드를 사용하는 라벨러 변환에서 토큰 집합을 사용할 수 있습니다.

토큰 집합을 사용하여 토큰 레이블 지정 작업의 일부로 특정 토큰을 식별할 수 있습니다. 예를 들어 토큰 집합을 사용하여 "AccountName@DomainName" 형식을 사용하는 모든 전자 메일 주소에 레이블을 지정할 수 있습니다. 토큰에 레이블을 지정하면 파서 변환을 사용하여 전자 메일 주소를 지정하는 출력 포트에 쓸 수 있습니다.

Developer tool에 광범위한 패턴 범위를 식별하기 위해 사용할 수 있는 시스템 정의 토큰 집합이 포함되어 있습니다. 다음은 시스템 정의 토큰 집합의 예입니다.

- 단어
- 숫자
- 전화 번호
- 전자 메일 주소
- 우편 번호
- 사회 보장 번호 등의 국가 ID 번호
- 신용 카드 번호

## 라벨러 변환 전략

레이블을 입력 데이터에 할당하려면 레이블 지정 전략을 사용합니다. 레이블 지정 전략을 구성하려면 라벨러 변환의 전략 보기에서 설정을 편집합니다.

레이블 지정 전략을 작성할 때는 하나 이상의 작업을 추가합니다. 각 작업은 특정 레이블 지정 태스크를 구현합니다.

전략은 라벨러 변환이 제공하는 마법사에서 작성할 수 있습니다. 레이블 지정 전략을 작성할 때 문자 레이블 지정 또는 토큰 레이블 지정 모드 중 원하는 모드를 선택하십시오. 그런 다음 선택한 레이블 지정 모드에 관련된 작업을 추가합니다.

**중요:** 작업 및 전략의 순서를 변경할 수 있습니다. 각 작업은 이전 작업의 결과를 읽으므로 전략 안의 작업 순서를 변경하면 전략의 출력이 변경될 수 있습니다.

## 문자 레이블 지정 작업

데이터의 문자 패턴을 설명하는 레이블을 작성하려면 문자 레이블 지정 작업을 사용하십시오.

다음 유형의 작업을 문자 레이블 지정 전략에 추가할 수 있습니다.

### 문자 집합을 사용하여 문자 레이블 지정

숫자 또는 영문자 같은 미리 정의된 문자 집합을 사용하여 문자 레이블을 지정합니다. 유니코드 및 유니코드 가 아닌 문자 집합을 선택할 수 있습니다.

### 참조 테이블을 사용하여 문자 레이블 지정

참조 테이블의 사용자 지정 레이블을 사용하여 문자 레이블을 지정합니다.

## 토큰 레이블 지정 작업

토큰 레이블 지정 작업을 사용하여 데이터의 문자열을 설명하는 레이블을 작성할 수 있습니다.

라벨러 변환은 입력 문자열에서 여러 토큰을 식별하고 레이블을 지정할 수 있습니다. 예를 들어 **US** 전화 번호 및 전자 메일 주소 토큰 집합을 사용하도록 라벨러 변환을 구성할 수 있습니다. 라벨러 변환이 입력 문자열 "555-555-1212 someone@somewhere.com"을 처리하는 경우 출력 문자열은 "USPHONE EMAIL"입니다.

다음 유형의 토큰 레이블 지정 작업을 레이블 지정 전략에 추가할 수 있습니다.

### 참조 테이블을 사용하여 레이블 지정

참조 테이블 항목과 일치하는 문자열에 레이블을 지정합니다.

### 토큰 집합을 사용하여 토큰 레이블 지정

토큰 집합 데이터 또는 확률 모델 데이터와 일치하는 문자열 패턴에 레이블을 지정합니다.

## 라벨러 변환 포트

변환에 구성한 레이블 지정 작업에 필요한 입력 및 출력 포트를 선택해야 합니다.

라벨러 변환은 다음 포트를 사용합니다.

### 입력 포트

업스트림 개체의 문자열 입력을 읽습니다.

### 레이블이 지정된 출력 포트

변환 작업이 정의한 레이블을 기록합니다.

### 토큰화된 출력 포트

출력의 각 레이블에 해당하는 입력 문자열을 전달합니다. 라벨러 변환의 파서 변환 다운스트림을 맵셋 또는 매핑에 추가하고 파서 변환이 패턴 기반 구문 분석 모드에서 실행되도록 구성하려는 경우 이 포트를 선택합니다. 파서 변환은 토큰 레이블 지정 출력을 토큰화된 출력 포트의 데이터에 연결합니다.

### 점수 출력 포트

토큰 레이블 지정 작업에서 확률 일치 기술이 생성한 점수 값을 기록하려면 이 옵션을 선택합니다.

확률 모델을 사용하는 토큰 레이블 지정 작업을 실행하면 레이블이 지정된 각 문자열에 대한 숫자 점수가 생성됩니다. 이 점수는 입력 문자열과 확률 모델에 정의된 패턴 간의 유사 정도를 나타냅니다.

## 문자 레이블 지정 속성

문자 레이블 지정 작업에 대한 속성은 라벨러 변환의 **전략** 보기에서 구성합니다.

### 일반 속성

일반 속성은 전략에서 정의하는 모든 문자 레이블 지정 작업에 적용됩니다. 일반 속성은 전략의 이름을 지정하고 입력 및 출력 포트를 지정할 때 사용합니다.

다음 표에는 일반 속성이 설명되어 있습니다.

속성	설명
이름	전략 이름을 제공합니다.
입력	전략 작업이 읽을 수 있는 입력 포트를 식별합니다.
출력	전략 작업이 쓸 수 있는 출력 포트를 식별합니다.
설명	전략의 텍스트 설명을 제공합니다. 이 속성은 선택 사항입니다.

### 참조 테이블 속성

문자 레이블 지정 전략을 정의할 때 문자 집합 및 참조 테이블을 사용하여 레이블을 지정하는 작업을 추가할 수 있습니다. 변환이 참조 테이블이 사용하는 방식을 지정하려면 참조 테이블 속성을 사용합니다.

다음 표에는 참조 테이블 속성이 설명되어 있습니다.

속성	설명
이름	작업 이름을 제공합니다.
참조 테이블	변환이 문자 레이블 지정에 사용할 참조 테이블을 지정합니다.
레이블	참조 테이블 항목과 일치하는 입력 문자에 대한 대체 텍스트를 지정합니다.
전략에 있는 다른 레이블 재정의	이 레이블 지정 작업으로 다른 레이블 지정 작업을 재정의할지 여부를 결정합니다.



## 문자 집합 속성

문자 레이블 지정 전략을 정의할 때 문자 집합 및 참조 테이블을 사용하여 레이블을 지정하는 작업을 추가할 수 있습니다. 문자 집합 속성을 사용하여 변환이 문자 집합을 사용하는 방식을 정의합니다.

다음 표에는 문자 집합 속성이 설명되어 있습니다.

속성	설명
이름	작업 이름을 제공합니다.
문자 집합 선택	변환이 문자열 레이블 지정에 사용할 문자 집합을 지정합니다. 문자 집합에 일치하는 입력 문자열에 대한 대체 텍스트를 재정의할 수 있습니다. <b>레이블</b> 열의 선택 화살표를 클릭하여 사용자 지정 대체 텍스트를 입력합니다.
필터 텍스트	사용자가 입력한 문자 또는 와일드카드를 사용하여 문자 집합의 목록을 필터링합니다.
문자 집합 추가	사용자 지정 문자 집합을 정의하려면 선택합니다.
편집	사용자 지정 문자 집합의 콘텐츠를 편집합니다.
가져오기	콘텐츠 집합에 저장된 문자 집합의 재사용할 수 없는 사본을 작성할 수 있습니다. 원래 문자 집합을 변경하는 경우 라벨러 변환에 저장된 사본은 업데이트되지 않습니다.
제거	사용자 지정 문자 집합을 삭제합니다.
실행 순서 지정	데이터에 토큰 집합이 적용되는 순서를 설정합니다. 위쪽 및 아래쪽 화살표를 사용하여 순서를 변경합니다.

## 필터 속성

레이블 지정 작업 중에 건너뛰 값을 지정할 수 있습니다. 레이블 지정 작업을 적용하지 않을 값을 지정하려면 **텍스트 무시** 속성을 사용합니다.

다음 테이블에는 필터 속성이 설명되어 있습니다.

속성	설명
검색 용어	변환이 레이블 지정 작업을 수행하기 전에 필터링하는 문자열을 지정합니다. 이 기능을 사용하면 정의된 레이블 지정 전략에 대한 예외를 지정할 수 있습니다.
대/소문자 구분	필터링된 문자열을 검색 용어의 대/소문자와 일치할지 여부를 결정합니다.
대문자	필터링된 문자열을 대문자로 변환합니다.
시작	필터링된 문자열에 대한 검색을 시작할 문자 위치를 지정합니다.
끝	필터링된 문자열에 대한 검색을 중지할 문자 위치를 지정합니다.

# 토큰 레이블 지정 속성

라벨러 변환의 전략 보기에서 토큰 레이블 지정 작업에 대한 속성을 구성합니다.

## 일반 속성

전략에서 정의하는 모든 토큰 레이블 지정 작업에는 일반 속성이 적용됩니다. 일반 속성을 사용하여 전략 이름, 입력 및 출력 포트, 그리고 전략이 확률 일치 기술을 활성화하는지 여부를 지정합니다.

다음 표에는 일반 속성이 설명되어 있습니다.

속성	설명
이름	전략 이름을 제공합니다.
입력	전략 작업에서 읽을 수 있는 출력 포트를 식별합니다.
출력	전략 작업에서 쓸 수 있는 출력 포트를 식별합니다.
설명	전략에 대한 설명입니다. 이 속성은 선택 사항입니다.
확률 일치 기술 사용	전략이 확률 모델을 사용하여 토큰 유형을 식별할 수 있음을 지정합니다.
반전 활성화	전략이 오른쪽에서 왼쪽으로 입력 데이터를 읽음을 나타냅니다. 확률 일치에 대해서는 이 속성이 비활성화됩니다.
구분자	변환이 입력 데이터의 하위 문자열을 평가하는 데 사용하는 문자를 지정합니다. 기본값은 공백입니다. 확률 레이블 지정에 대해서는 속성이 비활성화됩니다.
토큰화된 출력 필드	전략이 여러 레이블을 출력 포트에 쓸 수 있도록 합니다. 파서 변환에서 패턴 기반 구문 분석용 입력 데이터를 생성하려면 이 필드를 선택합니다.
점수 출력 필드	확률 일치에서 생성되는 점수 값이 포함된 필드를 식별합니다. 확률 일치 기술 사용 옵션을 선택할 때 점수 출력 필드를 설정합니다.
출력 구분자	출력 포트에서 데이터 값을 구분하는 문자를 지정합니다. 기본값은 콜론입니다.

## 토큰 집합 속성

토큰 집합을 사용하도록 레이블 지정 작업을 구성하는 경우 토큰 집합 속성이 적용됩니다.

다음 표에는 일반 속성이 설명되어 있습니다.

속성	설명
토큰 집합 선택	변환이 문자열 레이블 지정에 사용할 토큰 집합을 지정합니다.
필터 텍스트	토큰 집합 또는 정규식의 목록을 필터링합니다. 필터로 텍스트 문자와 와일드카드 문자를 사용합니다.
토큰 집합 추가	사용자 지정 토큰 집합을 정의하는 데 사용합니다.

속성	설명
정규식 추가	입력 패턴이 일치하는 정규식을 정의하는 데 사용합니다.
편집	사용자 지정 토큰 집합 또는 정규식의 콘텐츠를 편집합니다.
가져오기	모델 리포지토리의 폴더에서 정규식 또는 토큰 집합의 재사용 불가능한 사본을 가져옵니다. 토큰 집합 또는 정규식의 소스 개체를 업데이트하는 경우 데이터 통합 서비스는 재사용 불가능한 사본을 업데이트하지 않습니다.
제거	사용자 지정 토큰 집합 또는 정규식을 제거합니다.
실행 순서 지정	작업이 토큰 집합 또는 정규식을 데이터에 적용하는 순서를 설정합니다. 위쪽 및 아래쪽 화살표를 사용하여 순서를 변경합니다.

## 사용자 지정 레이블 속성

토큰 레이블 작업을 구성하는 경우 **사용자 지정 레이블** 보기를 선택하여 특정 검색 용어에 대한 레이블을 작성할 수 있습니다.

다음 테이블에는 사용자 지정 레이블 속성이 설명되어 있습니다.

속성	설명
검색 용어	검색하려는 문자열을 식별합니다.
대/소문자 구분	입력 데이터가 검색 용어의 대/소문자와 일치해야 하는 지 여부를 지정합니다.
사용자 지정 레이블	적용할 사용자 지정 레이블을 식별합니다.

## 확률 일치 속성

확률 일치 기술을 사용하는 옵션을 선택하는 경우 확률 모델을 레이블 지정 작업에 추가할 수 있습니다. 확률 모델을 토큰 집합 또는 참조 테이블을 사용하는 전략에는 추가할 수 없습니다.

다음 테이블에는 확률 일치와 관련된 속성이 설명되어 있습니다.

속성	설명
이름	작업 이름을 제공합니다.
필터 텍스트	입력하는 문자 또는 와일드카드를 사용하여 리포지토리의 확률 모델 목록을 필터링합니다.
확률 모델	작업에서 사용하기 위한 확률 모델을 식별합니다.

## 참조 테이블 속성

참조 테이블을 사용하도록 레이블 지정 작업을 구성하는 경우 참조 테이블 속성이 적용됩니다.

다음 표에는 참조 테이블 속성이 설명되어 있습니다.

속성	설명
이름	작업 이름을 제공합니다.
참조 테이블	토큰의 레이블을 지정하는 데 작업이 사용하는 참조 테이블을 지정합니다.
레이블	입력 문자열이 참조 테이블 항목과 일치하는 경우 작업이 새 포트에 기록하는 텍스트를 지정합니다.
대/소문자 구분	입력 문자열이 참조 테이블 항목의 대/소문자와 일치해야 하는지 여부를 결정합니다.
유효한 값으로 일치 항목 대체	레이블이 지정된 문자열을 참조 테이블의 Valid 열의 항목으로 바꿉니다.
모드	토큰 레이블 지정 방법을 결정합니다. 참조 테이블 항목과 일치하는 입력 문자열의 레이블을 지정하려면 포괄적을 선택합니다. 참조 테이블 항목과 일치하지 않는 입력 문자열의 레이블을 지정하려면 제외를 선택합니다.
우선 순위 설정	전략에서 참조 테이블 레이블 지정 작업이 토큰 집합 레이블 지정 작업보다 우선하는지 여부를 결정합니다. 이 속성을 설정하면 변환이 토큰 집합 레이블 지정에 앞서 참조 테이블 레이블 지정을 수행하고 토큰 집합 분석이 참조 테이블 레이블 분석을 덮어쓸 수 없습니다.

## 문자 레이블 지정 전략 구성

레이블 지정 전략을 구성하려면 라벨러 변환의 **전략** 보기에서 설정을 편집합니다.

1. **전략** 보기를 선택하고 **새로 만들기**를 클릭하여 전략을 작성합니다.  
전략 마법사가 열립니다.
2. 전략 이름을 입력합니다.
3. **입력** 및 **출력** 필드를 클릭하여 전략에 대한 포트를 정의합니다.
4. 필요한 경우 전략에 대한 설명을 입력합니다.
5. 문자 레이블 지정 모드를 선택합니다.
6. **다음**을 클릭합니다.
7. 구성할 문자 레이블 지정 작업의 유형을 선택합니다. 다음 작업을 구성할 수 있습니다.
  - 참조 테이블을 사용하여 문자 레이블 지정.
  - 문자 집합을 사용하여 문자 레이블 지정.
8. **다음**을 클릭합니다.
9. 작업 속성을 구성하고 **다음**을 클릭합니다.
10. 필요한 경우 **텍스트 무시** 속성을 구성합니다.
11. **다음**을 클릭하여 더 많은 작업을 전략에 추가하거나 **마침**을 클릭합니다.

변환이 전략 및 작업을 처리하는 순서를 변경할 수 있습니다. **전략** 보기에서 전략 또는 작업을 선택하고 **위로 이동** 또는 **아래로 이동**을 클릭합니다.

## 토큰 레이블 지정 전략 구성

레이블 지정 전략을 구성하려면 라벨러 변환의 **전략** 보기에서 설정을 편집합니다.

1. **전략** 보기를 선택하고 **새로 만들기**를 클릭하여 전략을 작성합니다.  
전략 마법사가 열립니다.
2. 전략 이름을 입력합니다.
3. **입력** 및 **출력** 필드를 클릭하여 전략에 대한 포트를 정의합니다.
4. 필요한 경우 전략에 대한 설명을 입력합니다.
5. 토큰 레이블 지정 모드를 선택합니다.  
선택한 모드의 속성을 확인하거나 편집합니다.
6. **다음**을 클릭합니다.
7. 구성할 토큰 레이블 지정 작업의 유형을 선택합니다. 다음 작업을 구성할 수 있습니다.
  - 토큰 집합을 사용하여 토큰 레이블을 지정합니다.
  - 참조 테이블을 사용하여 토큰 레이블을 지정합니다.
  - 확률 일치를 사용하여 토큰 레이블을 지정합니다.
8. **다음**을 클릭합니다.
9. 작업 속성을 구성하고 **다음**을 클릭합니다.  
확률 일치를 사용하도록 전략을 구성한 경우 잔여 데이터로 식별된 토큰에 사용할 레이블을 입력합니다.
10. 필요에 따라 **사용자 지정 레이블** 속성을 구성합니다.
11. **다음**을 클릭하여 더 많은 작업을 전략에 추가하거나 **마침**을 클릭합니다.

변환이 전략 및 작업을 처리하는 순서를 변경할 수 있습니다. **전략** 보기에서 전략 또는 작업을 선택하고 **위로 이동** 또는 **아래로 이동**을 클릭합니다.

## 라벨러 변환의 고급 속성

라벨러 변환 데이터에 대한 데이터 통합 서비스의 처리 방식을 결정하는 데 도움이 되는 속성을 구성할 수 있습니다.

로그의 추적 수준을 구성할 수 있습니다.

**고급** 탭에서 다음 속성을 구성합니다.

### 추적 수준

이 변환에 대해 로그에 표시되는 세부 정보의 양입니다. 간단, 보통, 자세한 정보 표시 초기화 또는 자세한 정보 표시 데이터를 선택할 수 있습니다. 기본값은 보통입니다.

## 라벨러 변환 - 비원시 환경

비원시 환경에서 라벨러 변환의 처리는 변환을 실행하는 엔진에 따라 다릅니다.

다음과 같은 비원시 런타임 엔진에 대한 지원을 고려하십시오.

- **Blaze** 엔진. 제한 없이 지원됩니다.
- **Spark** 엔진. 제한 없이 지원됩니다.
- **Databricks Spark** 엔진. 지원되지 않습니다.

## 제 25 장

# 조회 변환

이 장에 포함된 항목:

- [조회 변환 개요, 371](#)
- [연결된 조회 및 연결되지 않은 조회, 372](#)
- [조회 변환 개발, 374](#)
- [조회 쿼리, 374](#)
- [조회 소스 필터, 377](#)
- [조회 조건, 378](#)
- [조회 캐시, 381](#)
- [쿼리 속성, 381](#)
- [동적 매핑의 조회 변환, 382](#)
- [동적 포트 정의, 382](#)
- [조회 소스 변경, 382](#)
- [포트 선택기, 387](#)
- [런타임 속성, 392](#)
- [고급 속성, 393](#)
- [재사용 가능한 조회 변환 작성, 395](#)
- [재사용 불가능한 조회 변환 작성, 396](#)
- [연결되지 않은 조회 변환 작성, 397](#)
- [연결되지 않은 조회 예제, 397](#)
- [조회 변환 - 비원시 환경, 399](#)

## 조회 변환 개요

조회 변환은 플랫폼 파일, 논리적 데이터 개체, 참조 테이블 또는 관계형 테이블에서 데이터를 조회하는 수동 또는 활성 변환입니다. 조회 변환은 조회에서 행을 1개 또는 여러 개 반환할 수 있습니다.

조회 변환을 작성하기 전에 조회 소스를 작성하십시오. 플랫폼 파일 또는 관계형 데이터베이스 테이블을 실제 데이터 개체로 가져옵니다. 또는 조회 소스로 사용할 논리적 데이터 개체 또는 참조 테이블을 작성하십시오. 조회 변환을 작성할 경우 **Developer tool**에서 데이터 개체 또는 참조 테이블의 열을 변환에 조회 포트에 추가합니다. 변환을 작성한 후 조회 결과를 반환하는 하나 이상의 출력 포트를 구성하십시오. 조회 조건을 구성하고 다른 조회 속성을 구성하십시오.

매핑 또는 미리보기 데이터를 실행할 경우 통합 서비스에서 조회 소스를 쿼리합니다. 통합 서비스는 변환의 조회 포트, 조회 속성 및 조회 조건을 기반으로 조회 소스를 쿼리합니다. 조회 변환에서 조회 결과를 대상 또는 다른 변환에 반환합니다.

연결되거나 연결되지 않은 조회 변환을 구성할 수 있습니다. 연결된 변환은 매핑의 다른 변환에 연결됩니다. 연결되지 않은 변환은 다른 변환의 :LKP 식에서 입력을 수신합니다. 조회 변환이 논리적 데이터 개체에서 조회를 수행할 경우 연결된 조회 변환을 구성해야 합니다. 조회 변환 입력 포트를 업스트림 변환 또는 업스트림 소스에 연결합니다. 출력 포트를 다운스트림 변환 또는 다운스트림 대상에 연결합니다.

매핑에서 여러 조회 변환을 사용할 수 있습니다.

조회 변환과 함께 다음 태스크를 수행할 수 있습니다.

- 관련 값 가져오기. 입력 데이터의 값을 기반으로 조회 소스에서 값을 검색합니다. 예를 들어, 입력 데이터에 직원 ID가 포함되어 있습니다. 조회 소스에서 직원 ID로 직원 이름을 검색합니다.
- 조회 소스에서 여러 행을 검색합니다.
- 계산 수행. 조회 테이블에서 값을 검색하고 계산에서 값을 사용합니다. 예를 들어, 판매세를 검색하고, 세금을 계산한 다음 대상에 세금을 반환합니다.
- 식을 허용하는 변환에서 :LKP 식과 함께 연결되지 않은 조회를 수행합니다. 변환의 다른 식으로 결과를 필터링합니다.
- 동적 매핑에서 조회 변환을 사용하려면 조회 소스 및 조회 조건을 매개 변수화합니다.

## 연결된 조회 및 연결되지 않은 조회

연결된 조회 변환 또는 연결되지 않은 조회 변환을 구성할 수 있습니다. 연결된 조회 변환은 매핑의 다른 변환에 연결되는 입력 및 출력 포트가 있는 변환입니다. 연결되지 않은 조회 변환은 매핑에 표시되지만 다른 변환에 연결되지 않습니다.

연결되지 않은 조회 변환은 식 변환 또는 집계 변환 같은 변환에서 :LKP 식 결과의 입력을 수신합니다. :LKP 식은 조회 변환에 인수를 전달하고 조회 변환으로부터 결과를 다시 수신합니다. :LKP 식은 조회 결과를 식 변환 또는 집계 변환의 다른 식에 전달하여 결과를 필터링할 수 있습니다.

다음 표에는 연결된 조회와 연결되지 않은 조회 간 차이점이 나열되어 있습니다.

연결된 조회	연결되지 않은 조회
파이프라인에서 직접 입력 값을 받습니다.	다른 변환의 :LKP 식 결과에서 입력 값을 받습니다.
동적 또는 정적 캐시를 사용합니다.	정적 캐시를 사용합니다.
캐시에는 조회 조건의 조회 소스 열과 출력 포트인 조회 소스 열이 포함됩니다.	캐시에는 조회 조건의 모든 조회 및 출력 포트와 조회/반환 포트가 포함됩니다.
동일한 행 또는 삽입의 여러 열을 동적 조회 캐시에 반환합니다.	각 행의 열 1개를 반환 포트에 반환합니다.
조회 조건에 일치하는 값이 없으면 통합 서비스는 모든 출력 포트에 대해 기본값을 반환합니다. 동적 캐시를 구성한 경우 통합 서비스는 행을 캐시에 삽입하거나 변경하지 않은 상태로 그대로 둡니다.	조회 조건에 일치하는 값이 없으면 통합 서비스는 NULL을 반환합니다.

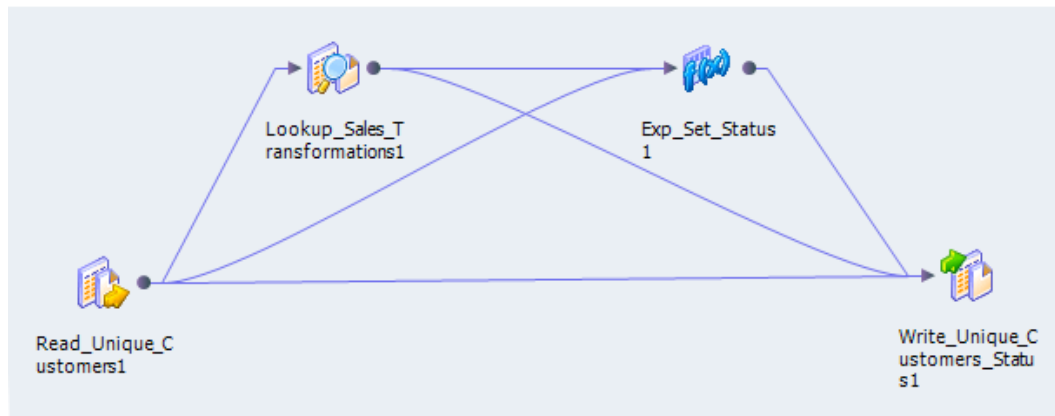


연결된 조회	연결되지 않은 조회
조회 조건에 일치하는 값이 있으면 통합 서비스는 모든 조회/출력 포트에 대해 조회 조건 결과를 반환합니다. 동적 캐싱을 구성한 경우 통합 서비스는 캐시의 행을 업데이트하거나 행을 변경되지 않은 상태로 그대로 둡니다.	조회 조건에 대한 일치가 발생할 경우 통합 서비스는 조회 조건 결과를 반환 포트에 반환합니다.
여러 출력 값을 다른 변환에 전달합니다. 조회/출력 포트를 다른 변환에 연결합니다.	출력 값 1개를 다른 변환에 반환합니다. 조회 변환 반환 포트에서 다른 변환의 LKP 식을 포함하는 포트에 값을 전달합니다.
사용자 정의 기본값을 지원합니다.	사용자 정의 기본값을 지원하지 않습니다.

## 연결된 조회

연결된 조회 변환은 매핑의 소스 또는 대상에 연결된 조회 변환입니다.

다음 그림에서는 연결된 조회 변환이 포함된 매핑을 보여 줍니다.



연결된 조회 변환이 포함된 매핑을 실행할 경우 통합 서비스에서 다음 단계를 수행합니다.

1. 통합 서비스에서 다른 변환의 값을 조회 변환의 입력 포트에 전달합니다.
2. 각 입력 행마다 통합 서비스에서 변환의 조회 포트 및 조회 조건을 기반으로 조회 소스 또는 캐시를 쿼리합니다.
3. 변환이 캐시되지 않거나 변환에서 정적 캐시를 사용할 경우 통합 서비스가 조회 쿼리의 값을 반환합니다.  
변환에서 동적 캐시를 사용할 경우 통합 서비스가 캐시에서 행을 찾지 못하면 캐시에 행을 삽입합니다. 통합 서비스가 캐시에서 행을 찾은 경우 캐시의 행을 업데이트하거나 변경되지 않은 상태로 그대로 둡니다. 행을 삽입, 업데이트 또는 변경 없음으로 플래그 지정합니다.
4. 통합 서비스가 쿼리의 데이터를 반환하여 이를 매핑의 다음 변환에 전달합니다.  
변환에서 동적 캐시를 사용할 경우 행을 필터 또는 라우터 변환에 전달하여 새 행을 대상으로 필터링할 수 있습니다.

**참고:** 이 장에서는 달리 지정하지 않는 한 연결된 조회 변환을 설명합니다.

## 연결되지 않은 조회

연결되지 않은 조회 변환은 매핑의 소스 또는 대상에 연결되지 않은 조회 변환입니다. 식을 허용하는 변환에서 **:LKP** 식을 사용하여 조회를 호출합니다.

연결되지 않은 조회 변환은 일반적으로 느린 변경 차원 테이블을 업데이트할 때 사용됩니다. 느린 변경 차원 테이블에 대한 자세한 내용은 Informatica 기술 자료(<http://mysupport.informatica.com>)를 참조하십시오.

조회 식에 대한 구문은 **:LKP lookup\_transformation\_name(argument, argument, ...)**입니다.

각 인수를 나열하는 순서는 조회 변환의 조회 조건 순서와 일치해야 합니다. 조회 변환은 조회 변환 반환 포트를 통해 쿼리 결과를 반환합니다. 조회를 호출하는 변환은 **:LKP** 식이 포함된 포트에서 조회 결과 값을 수신합니다. 조회 쿼리가 값을 반환하지 못하는 경우 포트가 **Null** 값을 수신합니다.

연결되지 않은 조회를 수행하는 경우 동일한 조회를 매핑에서 여러 번 수행할 수 있습니다. 다른 식에서 조회 결과를 테스트하고 결과에 따라 행을 필터링할 수 있습니다.

연결되지 않은 조회 변환이 포함된 매핑을 실행하는 경우 통합 서비스가 다음 단계를 수행합니다.

1. 연결되지 않은 조회 변환은 집계 변환, 식 변환 또는 업데이트 전략 변환과 같은 다른 변환의 **:LKP** 식 결과에서 입력 값을 수신합니다.
2. 통합 서비스가 조회 변환의 조건 및 조회 포트에 따라 조회 소스 또는 캐시를 쿼리합니다.
3. 통합 서비스가 조회 변환의 반환 포트를 통해 값을 반환합니다.
4. 통합 서비스가 반환 값을 **:LKP** 식이 포함된 포트에 전달합니다.

## 조회 변환 개발

조회 변환을 개발할 경우 조회 소스 유형 및 조회 조건 같은 요인을 고려해야 합니다.

조회 변환을 개발할 경우 다음 요인을 고려하십시오.

- 플랫폼 파일, 논리적 데이터 개체, 참조 테이블 또는 관계형 데이터 개체에서 변환을 작성할지 여부. 조회 변환을 작성하기 전에 조회 소스를 작성하십시오. 플랫폼 파일 또는 관계형 데이터베이스 테이블을 실제 데이터 개체로 가져옵니다. 또는 조회 소스로 사용할 논리적 데이터 개체 또는 참조 테이블을 작성하십시오.
- 변환의 출력 포트.
- 변환의 조회 조건.
- 통합 서비스가 조회 데이터를 캐시하도록 할지 여부. 통합 서비스는 플랫폼 파일, 참조 테이블 또는 관계형 데이터 개체에 대한 데이터를 캐시할 수 있습니다.

## 조회 쿼리

통합 서비스는 조회 변환에서 구성한 포트 및 속성을 기반으로 조회를 쿼리합니다. 통합 서비스는 첫 번째 행이 조회 변환에 들어올 때 기본 조회 쿼리를 실행합니다.

관계형 테이블에 대해 관계형 조회 또는 파이프라인 조회를 사용할 경우 조회 쿼리를 재정의할 수 있습니다. 관계형 테이블에 대해 조회를 사용할 경우 조회 쿼리를 재정의할 수 있습니다. 재정의의 사용하여 **ORDER BY** 절을 변경하거나, **WHERE** 절을 추가하거나, 캐시되기 전에 조회 데이터를 변환할 수 있습니다. 재정의의 사용하여 **WHERE** 절을 추가하거나 조회 데이터가 캐시되기 전에 변환할 수 있습니다.

SQL 재정의의를 구성하고 조회 쿼리에서 필터링할 경우 통합 서비스가 필터를 무시합니다.

## 기본 조회 쿼리

기본 조회 쿼리에 다음 문이 포함되어 있습니다.

SELECT

SELECT 문에는 매핑의 모든 조회 포트가 포함되어 있습니다. 조회 쿼리에 대한 SELECT 문을 보려면 조회 SQL 재정의 속성을 선택합니다.

SELECT

SELECT 문에는 매핑의 모든 조회 포트가 포함되어 있습니다. 조회 쿼리에 대한 SELECT 문을 보려면 사용자 지정 쿼리 사용 속성을 선택합니다.

ORDER BY

ORDER BY 절은 조회 변환에 표시되는 것과 동일한 순서로 열이 표시되도록 합니다. 통합 서비스에서 ORDER BY 절을 생성합니다. 기본 SQL을 생성할 경우 이 절이 표시되지 않습니다.

## 조회 쿼리에 대한 SQL 재정의

관계형 조회에 대해 조회 쿼리를 재정의할 수 있습니다. WHERE 절을 추가하고 조회 데이터가 캐시되기 전에 조회 데이터를 변환할 수 있습니다.

테이블 이름과 열 이름에 예약어와 슬래시를 사용할 수 있습니다.

쿼리를 입력해 기본 조회 쿼리를 완전히 재정의할 수 있습니다. 또는 기본 조회 쿼리를 보고 편집할 수 있습니다. 기본 조회 쿼리에는 조회 포트, 출력 포트 및 반환 포트가 포함됩니다.

SQL 재정의의 사용하는 경우 쿼리에 ORDER BY 1이 추가됩니다. 이 절은 다른 절에 대한 첫 번째 값과 마지막 값을 확실히 제공하도록 데이터를 정렬합니다.

**참고:** Hive 명령줄 유틸리티에서 다음 쿼리를 실행하여 SQL 유효성을 수동으로 검사할 수 있습니다.

```
CREATE VIEW <table name> (<port list>) AS <SQL>
```

설명:

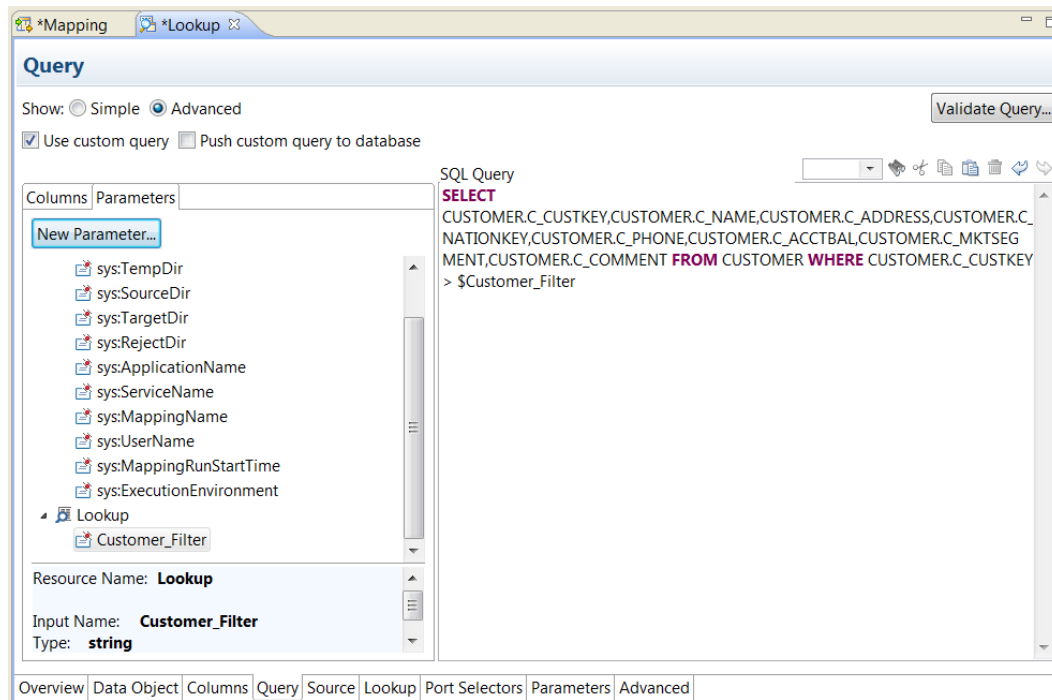
- <table name>은 선택한 이름입니다.
- <port list>는 소스의 쉼표로 구분된 포트 목록입니다.
- <SQL>은 유효성을 검사할 쿼리입니다.

## SQL 재정의 쿼리의 매개 변수

조회 변환의 조회 쿼리에서 시스템 매개 변수 또는 사용자 정의 매개 변수를 사용할 수 있습니다. SQL 편집기는 선택할 수 있는 시스템 매개 변수 및 사용자 정의 매개 변수 목록을 제공합니다.

조회 변환의 **쿼리** 탭에서 사용자 정의 매개 변수를 찾거나 작성할 수 있습니다. 각 매개 변수에 대한 기본값을 정의합니다. 조회 변환을 매핑에 추가한 후 맵셋 또는 매핑 매개 변수를 변환 매개 변수에 바인딩하여 기본값을 재정의할 수 있습니다.

다음 이미지는 조회 변환 쿼리 탭을 보여 줍니다.



## 예약어

조희 이름 또는 열 이름에 MONTH, YEAR 등의 데이터베이스 예약어가 포함되어 있으면 통합 서비스가 데이터베이스에 대해 SQL을 실행할 때 매핑이 실패하며 데이터베이스 오류가 발생합니다.

통합 서비스 설치 디렉터리에서 예약어 파일 **reswords.txt**를 생성하고 유지 관리해야 합니다. 통합 서비스는 세션을 초기화할 때 **reswords.txt** 파일을 검색하고 예약어를 따옴표로 묶은 다음 소스, 대상 및 조희 데이터베이스에 대해 SQL을 실행합니다.

통합 서비스 설치 디렉터리에서 예약어 파일 **reswords.txt**를 생성하고 유지 관리해야 합니다. 통합 서비스는 매핑을 초기화할 때 **reswords.txt** 파일을 검색하고 예약어를 따옴표로 묶은 다음 소스, 대상 및 조희 데이터베이스에 대해 SQL을 실행합니다.

Microsoft SQL Server 및 Sybase와 같은 일부 데이터베이스가 따옴표로 묶은 식별자에 대해 SQL-92 표준을 사용하도록 설정해야 할 수 있습니다. 연결 환경 SQL을 사용하여 명령을 실행합니다. 예를 들어 Microsoft SQL Server의 경우 다음 명령을 사용합니다.

Microsoft SQL Server 및 Sybase와 같은 일부 데이터베이스가 따옴표로 묶은 식별자에 대해 SQL-92 표준을 사용하도록 설정해야 할 수 있습니다. 환경 SQL을 사용하여 명령을 실행합니다. 예를 들어 Microsoft SQL Server의 경우 다음 명령을 사용합니다.

```
SET QUOTED_IDENTIFIER ON
```

## 조희 쿼리 재정의에 대한 지침

조희 쿼리를 재정의할 경우 특정한 규칙 및 지침이 적용됩니다.

조희 SQL 쿼리를 재정의하는 경우 다음 지침을 고려하십시오.

- 관계형 조희에 대한 조희 SQL 쿼리를 재정의할 수 있습니다.
- 기본 쿼리를 생성한 다음 재정의의 구성합니다. 이렇게 해야 모든 조희/출력 포트가 쿼리에 포함됩니다. SELECT 문의 포트를 추가하거나 뺄 경우 세션이 실패합니다.

- 소스 조회 필터를 추가하여 조회 캐시에 추가되는 행을 필터링합니다. 이렇게 하면 통합 서비스가 **WHERE** 절에 일치하는 동적 캐시 및 대상 테이블에 행을 삽입합니다.
- 여러 조회 변환이 조회 캐시를 공유하는 경우 동일한 조회 **SQL** 재정의의 각 조회 변환에 사용합니다.
- 모든 행을 반환하는 조회 변환을 구성한 경우 통합 서비스는 정렬된 키를 사용하여 조회 캐시를 작성합니다. 조회 변환이 조회의 모든 행을 검색하는 경우 통합 서비스가 정렬된 순서의 키를 사용하여 데이터 캐시를 작성합니다. 행이 정렬되지 않은 경우 통합 서비스가 캐시의 모든 행을 검색할 수 없습니다. 데이터가 키로 정렬되지 않은 경우 예기치 않은 결과가 나올 수 있습니다.
- **ORDER BY** 절에는 조회 조건에 표시되는 동일한 순서로 조건 포트가 포함되어야 합니다.
- **ORDER BY** 절을 재정의하는 경우 설명 표기법을 사용하여 조회 변환이 생성하는 **ORDER BY** 절을 억제하십시오.
- 푸시다운 최적화를 사용하는 경우 **ORDER BY** 절을 재정의하거나 생성된 **ORDER BY** 절을 설명 표기법으로 억제할 수 없습니다.
- 조회 쿼리의 테이블 이름 또는 열 이름에 예약어가 포함되는 경우 예약어를 따옴표로 묶어야 합니다.
- 캐시되지 않은 조회에 대한 조회 쿼리를 재정의하려면 통합 서비스가 여러 일치점을 찾을 때 값을 반환하도록 선택하십시오.
- 기본 **SQL** 문의 열은 추가하거나 삭제할 수 없습니다.
- **SQL** 재정의에는 매개 변수나 변수를 포함할 수 없습니다.
- **Developer tool**은 **SQL** 쿼리의 구문 유효성을 검사하지 않습니다. 연결되지 않은 조회 쿼리의 **SQL** 재정의가 올바르지 않은 경우 매핑이 실패합니다.

## 조회 쿼리 재정의

기본 조회 **SQL** 쿼리를 재정의하여 조회 소스에서 사용자 지정된 쿼리를 작성할 수 있습니다.

1. **속성** 보기에서 **쿼리** 탭을 선택합니다.
2. **고급**을 선택합니다.
3. **사용자 지정 쿼리 사용**을 선택합니다.
4. **SQL 쿼리 영역**에서 조회 쿼리를 편집합니다.  
테이블 이름, 열 이름 또는 매개 변수를 두 번 클릭하면 쿼리에 추가할 수 있습니다.
5. **쿼리 유효성 검사**를 클릭하여 조회 쿼리의 유효성을 검사합니다.
6. **사용자 지정 쿼리를 데이터베이스로 푸시**를 선택하여 데이터베이스에서 조회 쿼리를 실행합니다.

## 조회 소스 필터

캐싱이 활성화된 관계형 조회 변환에 대한 조회 소스 필터를 구성할 수 있습니다. 통합 서비스가 조회 소스 테이블에서 수행하는 조회 수를 제한하려면 조회 소스 필터를 추가합니다.

조회 소스 필터를 구성한 경우 통합 서비스가 필터 문의 결과를 기반으로 조회를 수행합니다. 예를 들어, **ID**가 **510**보다 큰 모든 직원을 성을 검색해야 할 경우가 있습니다.

**EmployeeID** 열에서 다음 조회 소스 필터를 구성하십시오.

```
EmployeeID >= 510
```

EmployeeID는 조회 변환의 입력 포트입니다. 통합 서비스가 소스 행을 읽을 때 EmployeeID 값이 510보다 클 경우 캐시에서 조회를 수행합니다. EmployeeID가 510보다 작거나 같을 경우 조회 변환에서 성을 검색하지 않습니다.

푸시다운 최적화용으로 구성된 매핑의 조회 쿼리에 조회 소스 필터를 추가할 경우 통합 서비스에서 SQL 재정의 를 나타내는 보기를 작성합니다. 통합 서비스가 이 보기에 대해 SQL 쿼리를 실행하여 데이터베이스에 변환 논리 를 푸시합니다.

## 조회에서 소스 행 필터링

캐싱이 활성화된 관계형 조회 변환에 대한 조회 소스 필터를 구성할 수 있습니다. 조회 소스 테이블에서 통합 서비스가 수행하는 조회 수를 제한하도록 조회의 소스 행을 필터링합니다.

1. **속성** 보기에서 **쿼리** 탭을 선택합니다.
2. **필터** 옵션에서 **편집**을 클릭합니다.
3. SQL 편집기에서 입력 포트를 선택하거나 필터링할 조회 변환 포트를 입력합니다.
4. 필터 조건을 입력합니다.

필터 조건에 **WHERE** 키워드를 포함하지 않습니다. 문자열 매핑 매개 변수 및 변수를 문자열 식별자로 묶습니다.

5. **쿼리 유효성 검사**를 클릭하여 필터 조건 구문의 유효성을 검사합니다.

## 조회 조건

데이터 통합 서비스가 조회 조건을 기반으로 조회 소스에서 데이터를 조회합니다. 조회 변환에 조회 조건을 구성한 경우 소스 데이터에 있는 하나 이상의 열 값을 조회 소스 또는 캐시의 값과 비교하십시오.

예를 들어 소스 데이터는 employee\_number를 포함합니다. 조회 소스 테이블은 employee\_ID, first\_name 및 last\_name을 포함합니다. 다음 조회 조건을 구성합니다.

```
employee_ID = employee_number
```

employee\_number마다 데이터 통합 서비스가 조회 소스의 employee\_ID, last\_name 및 first\_name 열을 반환합니다.

데이터 통합 서비스가 조회 소스에서 둘 이상의 행을 반환할 수 있습니다. 다음 조회 조건을 구성합니다.

```
employee_ID > employee_number
```

데이터 통합 서비스가 소스 직원 번호보다 큰 모든 employee\_ID 번호의 행을 반환합니다.

### 데이터 개체 조회의 Null 값

조회 조건의 입력이 NULL인 경우 데이터 개체 조회 변환이 Null 값이 있는 단일 행을 출력 전용 포트에 반환하고 입력 행의 값을 통과 포트에 반환합니다.

예를 들어 다음 조회 조건은 employee\_ID에 대한 값이 NULL이 하나 이상의 행이 포함된 데이터 소스에 대한 조회를 수행합니다.

```
employee_ID = employee_number
```

이 예에서 다음 데이터로 조회 테이블을 사용합니다.

EMPLOYEE_ID	LAST_NAME
1294765	Hara
1356356	Carver
1407207	NULL
1570348	Draper
NULL	Limonov

다음 데이터 소스의 입력 값을 조회 테이블과 비교합니다.

```
EMPLOYEE_NUMBER
-----
1294765
1356356
1407207
1648246
NULL
```

이 예에서 조회 조건은 다음 결과를 생성합니다.

```
1294765,Hara
1356356,Carver
1407207,NULL
NULL,NULL
NULL,NULL
```

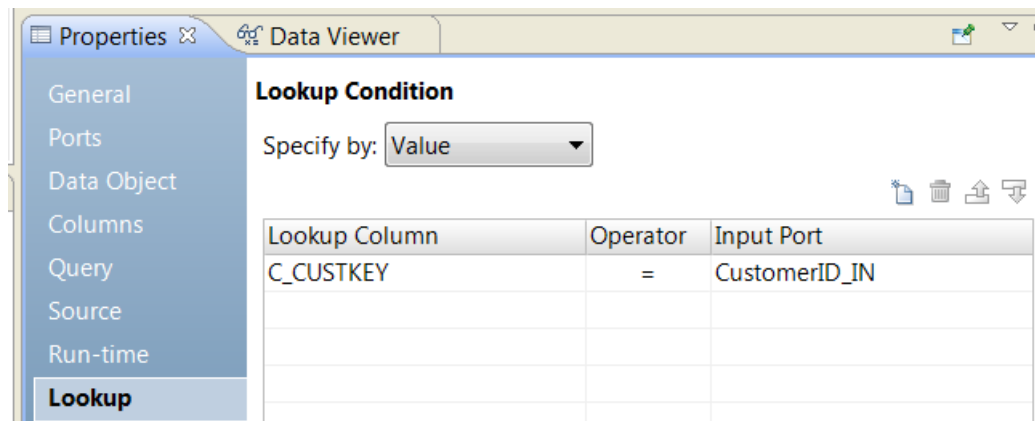
조회 조건이 첫 번째 두 개 행에 대한 EMPLOYEE\_ID 및 EMPLOYEE\_NUMBER 간 일치를 찾습니다. 세 번째 행의 경우 조회 소스에 조회 조건에 관여하지 않는 NULL 값이 있는 행이 포함됩니다. 해당 행은 조회 조건을 일치시키고 비조회 열에 대한 NULL 값이 있는 결과를 반환합니다.

네 번째 및 다섯 번째 행의 경우 조회 조건이 일치하지 않고 두 값에 대해 NULL을 반환합니다. 다섯 번째 행의 경우 NULL이 NULL을 포함하여 일치하지 않으므로 일치를 다시 찾지 않습니다.

## 조회 조건 구성

조회 조건은 조회 소스에서 검색할 행을 결정하는 식입니다. **속성** 보기의 **조회** 탭에서 조회 조건을 구성합니다.

다음 이미지는 고객 번호로 조회를 수행하기 위한 조회 조건을 보여 줍니다.



**조회** 탭에서 다음 옵션을 구성할 수 있습니다.

## 지정 기준

값을 선택하여 조회 열 및 입력 포트 이름을 선택합니다. 매개 변수를 선택하여 조회 조건을 정의하기 위한 식 매개 변수를 구성합니다.

## 조회 열

입력 행의 열과 일치시킬 조회 소스의 열입니다. 조회 조건에 여러 열을 포함할 수 있습니다.

## 연산자

조회 열 및 입력 포트 간 검색할 조건을 결정하는 연산자입니다. 연산자에는 =, !=, >, <, >=, <=이 포함됩니다.

## 입력 포트

조회 소스에서 검색할 값을 포함하는 입력 포트입니다. 한 개 이상의 입력 포트를 조회 소스의 포트와 비교할 수 있습니다.

# 조회 변환 조건의 규칙 및 지침

조회 변환에 대해 조건을 입력할 때는 특정 규칙과 지침이 적용됩니다.

조회 변환에 대해 조건을 입력할 때는 다음 규칙과 지침을 고려하십시오.

- 조회 조건의 열 데이터 유형이 일치해야 합니다.
- 조회 조건의 각 조회 포트에 대해 입력 포트를 하나씩 사용합니다. 변환에서 둘 이상의 조건에 같은 입력 포트를 사용할 수 있습니다.
- 조회 조건에서 포트 선택기 또는 동적 포트를 사용하는 경우 조회 조건이 식의 모든 포트를 고려합니다.
- 동적 입력 포트 또는 포트 선택기를 조회 조건의 입력 포트로 사용할 수 있습니다. 입력 포트의 생성된 포트 수는 조회 열의 포트 수와 같아야 합니다.
- 조회 조건이 여러 개인 조회 변환을 처리할 때 통합 서비스는 모든 조회 조건과 일치하는 행을 반환합니다.
- 식 매개 변수를 작성하여 재사용 불가능 조회 변환에서 조회 조건을 매개 변수화할 수 있습니다.
- 조회 성능을 높이려면 다음 순서로 조건을 입력하십시오.
  - 같음(=)
  - 보다 작음(<), 보다 큼(>), 작거나 같음(<=), 크거나 같음(>=)
  - 같지 않음(!=)
- 조회 조건을 작성할 때는 =, >, <, >=, <=, 또는 != 연산자 중 하나를 사용합니다.
- 통합 서비스는 조회 변환을 구성할 때 사용하는 항목(동적 조회 캐시, 정적 조회 캐시, 캐시되지 않은 조회)에 따라 조회 일치 여부를 다르게 처리합니다.
- 통합 서비스는 null 값의 일치 여부를 확인합니다. 예를 들어 통합 서비스는 조회 포트와 입력 포트가 모두 null 값을 포함하는 경우 이 두 포트가 같다고 간주합니다.
- 조회 조건의 열이 10진수 데이터 유형이면 각 열의 전체 자릿수가 같은 전체 자릿수 범위에 속해야 합니다. 유효한 전체 자릿수 범위에는 다음이 포함됩니다.
  - 10진수 0-18
  - 10진수 19-28
  - 10진수 29 이상
  - 10진수 29-38
  - 10진수 39 이상
  - 10진수 29-38



- 10진수 39 이상

예를 들어 **DecimalA**의 전체 자릿수는 15이고 **DecimalB**의 전체 자릿수는 25인 **DecimalA = DecimalB** 조건을 정의하는 경우 조회 조건은 유효하지 않습니다.

## 조회 캐시

큰 조회 소스 또는 작은 조회 테이블을 캐싱하여 성능을 개선할 수 있습니다. 조회 소스를 캐시할 경우 통합 서비스는 각 입력 행에 대한 조회 소스를 쿼리하는 대신 조회 캐시를 쿼리합니다.

비즈니스 요구 사항에 따라 다른 유형의 조회 캐시를 작성할 수 있습니다. 정적 캐시 또는 동적 캐시를 작성할 수 있습니다. 지속형 또는 비지속형 캐시를 작성할 수 있습니다. 여러 조회 변환 간에 캐시를 공유할 수 있습니다.

조회 변환이 동적 매핑에 있는 경우 지속형 또는 비지속형 캐시가 있을 수 있습니다. 캐시를 지속하고 매개 변수로 조회 소스를 변경하는 경우 매핑이 실패합니다. 플랫폼 파일 조회 소스에 대한 제어 파일을 변경하는 경우에도 매핑이 실패합니다.

**참고:** 조회 변환에 동적 포트 또는 매개 변수화된 조회 소스가 포함된 경우 동적 조회 캐시 또는 지속형 조회 캐시를 사용할 수 없습니다.

## 쿼리 속성

관계형 조회 테이블에서 조회 쿼리를 보거나 변경하려면 쿼리 속성을 구성합니다. 조회에 대해 필터를 적용하거나 조회 쿼리를 사용자 지정할 수 있습니다.

다음 테이블에서는 관계형 조회를 수행하는 조회 변환의 쿼리 속성에 대해 설명합니다.

속성	설명
단순	조회 쿼리를 보고 조회에 대해 필터를 적용하려면 선택합니다.
고급	쿼리를 보거나 사용자 지정하거나 관계형 조회 테이블이 포함된 데이터베이스에서 쿼리를 실행하려면 선택합니다.
필터	통합 서비스가 쿼리하는 행 수를 줄이려면 필터를 입력합니다. <b>단순</b> 옵션을 선택해야 이 옵션이 표시됩니다.
사용자 지정 쿼리 사용	조회 쿼리를 재정의하려면 선택합니다. <b>고급</b> 옵션을 선택해야 이 옵션이 표시됩니다.
사용자 지정 쿼리를 데이터베이스로 푸시	관계형 조회 테이블이 포함된 데이터베이스에서 쿼리를 실행하려면 선택합니다. <b>고급</b> 옵션을 선택해야 이 옵션이 표시됩니다.
SQL 쿼리	조회를 수행하는 데 사용되는 SQL 쿼리를 표시합니다. SQL 쿼리를 사용자 지정할 수 있습니다. <b>고급</b> 옵션을 선택해야 이 옵션이 표시됩니다.

## 동적 매핑의 조회 변환

동적 매핑에서 조회 변환을 사용할 수 있습니다. 소스 데이터에 따라 여러 포트를 받고 반환하도록 동적 포트를 구성할 수 있습니다. 조회 소스 및 조회 조건을 매개 변수화하여 여러 포트에 따라 조회를 수행할 수 있습니다.

동적 매핑은 런타임 시 소스, 대상 및 변환 논리가 변경될 수 있는 매핑입니다. 매개 변수 및 규칙을 설정하여 데이터의 구조를 변경할 수 있습니다. 동적 매핑에서 조회 변환을 사용하는 경우 소스 데이터에 따라 조회 변환의 입력 포트가 변경될 수 있습니다. 조회 소스의 구조 및 조회 조건의 포트가 변경될 수 있습니다.

**참고:** 조회 변환이 동적 포트 또는 매개 변수화된 조회 소스를 포함하는 경우 조회 캐시를 지속시킬 수 없습니다. 또한 동적 캐시를 구성할 수 없습니다.

조회 변환이 동적 매핑의 변환을 사용하도록 다음 태스크를 수행할 수 있습니다.

### 동적 포트 정의

입력 열에 대한 변경을 포함하도록 동적 포트 및 생성된 포트를 정의합니다.

### 조회 소스 매개 변수화

조회 소스를 정의하는 데이터 개체에 대한 매개 변수를 할당합니다. 재사용 불가능 조회 변환에서 조회 소스를 매개 변수화할 수 있습니다.

### 포트 선택기 정의

조회 조건에서 포트를 사용하도록 지정하는 포트 선택기를 정의합니다. 재사용 불가능 조회 변환에서 포트 선택기 포트를 매개 변수화할 수 있습니다.

### 조회 조건 매개 변수화

식 매개 변수를 작성하고 전체 식을 포함하는 기본값을 정의합니다.

동적 매핑에 대한 자세한 내용은 *Informatica Developer 매핑 가이드*를 참조하십시오.

## 동적 포트 정의

조회 변환에서 동적 포트를 정의할 수 있습니다.

조회 조건의 입력 열에 있는 동적 포트를 참조할 수 있습니다. 동적 포트에 생성된 포트가 여러 개 있는 경우 조회 조건의 조회 열 요소에 대해 포트 선택기를 사용할 수 있습니다. 동적 입력 포트에는 조회 조건의 포트 선택기와 동일한 수의 포트가 포함되어야 합니다.

동적 포트에 단일 값만 포함된 경우 조회 조건의 조회 열 요소에서 단일 포트를 사용할 수 있습니다.

조회 조건의 생성된 포트를 참조할 수 있습니다. 그러나 동적 매핑의 소스가 변경되는 경우 생성된 포트가 존재하지 않을 수 있습니다. 매핑이 실패합니다.

## 조회 소스 변경

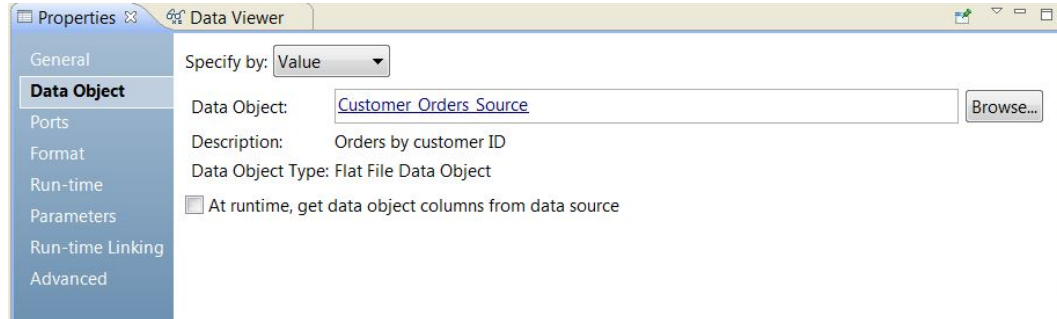
재사용 가능 조회 변환에 대한 조회 소스인 데이터 개체를 변경할 수 있습니다. 런타임 시 조회 소스로 사용할 데이터 개체를 결정하는 매개 변수를 구성할 수 있습니다.

실제 데이터 개체에서 변환을 작성하는 경우 변환 속성의 **데이터 개체** 탭에 데이터 개체에 대한 정보가 나타납니다. 데이터 개체 이름을 클릭하여 모델 리포지토리에서 실제 데이터 개체 정의를 볼 수 있습니다.

모델 리포지토리에서 다른 실제 데이터 개체를 찾아 변환에 대한 데이터 개체를 변경할 수 있습니다. 데이터 개체를 변경하는 경우 변환은 선택하는 데이터 개체의 런타임 속성 및 고급 속성을 사용합니다.

데이터 소스의 변경에 따라 런타임 시 데이터 개체의 구조를 업데이트할 수 있습니다. 데이터 소스는 데이터 개체가 나타내는 실제 파일 또는 데이터베이스 테이블입니다. 데이터 통합 서비스를 활성화하여 데이터 소스에서 데이터 열을 가져오는 경우 데이터 통합 서비스가 데이터 소스의 구조를 검사합니다. 데이터 통합 서비스는 데이터 소스에 따라 변환 인스턴스의 데이터 개체 포트를 업데이트합니다. 데이터 통합 서비스는 모델 리포지토리의 실제 데이터 개체 정의를 변경하지 않습니다.

다음 이미지는 **데이터 개체** 탭을 보여 줍니다.



**데이터 개체** 탭에는 다음 필드가 있습니다.

#### 지정 기준

값을 선택하여 특정 데이터 개체 이름을 입력합니다. **매개 변수**를 선택하여 데이터 개체를 매개 변수화합니다.

#### 데이터 개체

모델 리포지토리의 데이터 개체 이름입니다. **데이터 개체** 링크를 클릭하여 리포지토리에서 데이터 개체 정의를 열 수 있습니다. 또한 모델 리포지토리의 다른 데이터 개체를 찾아볼 수 있습니다.

#### 설명

리포지토리의 데이터 개체 설명입니다. 읽기 전용입니다.

#### 데이터 개체 유형

플랫 파일 데이터 개체, 관계형 테이블 개체 또는 사용자 지정된 데이터 개체와 같은 데이터 개체의 유형을 설명합니다.

#### 런타임 시 데이터 소스에서 데이터 개체 열을 가져옵니다.

데이터 통합 서비스는 런타임 시 데이터 개체가 참조하는 데이터 파일 또는 테이블에서 메타데이터 및 데이터 정의 변경을 가져온 후 변환 인스턴스에 대한 데이터 개체의 구조를 업데이트합니다.

런타임 시 데이터 통합 서비스가 메타데이터 및 데이터 정의 변경을 가져오는 방법을 미리 보려면 확인된 매개 변수와 함께 매핑을 표시합니다.

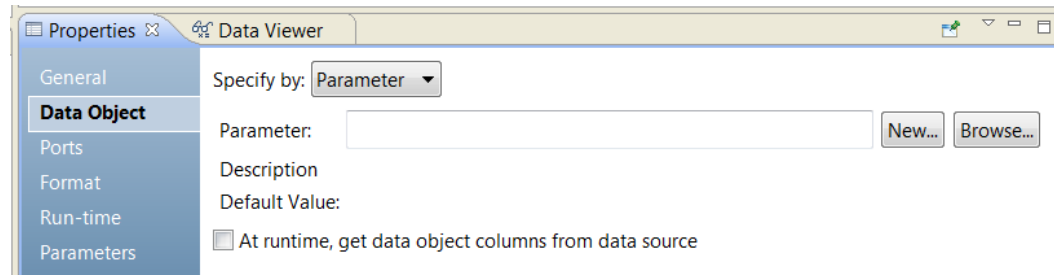
## 조회 소스 매개 변수화

재사용 불가능 조회 변환에서 조회 소스에 대한 매개 변수를 구성할 수 있습니다.

데이터 개체를 매개 변수화하려면 **데이터 개체** 탭에서 **매개 변수로 지정**을 선택합니다. **데이터 개체** 탭의 속성이 변경됩니다.

데이터 개체를 매개 변수화하려면 리소스 유형 매개 변수를 작성하거나 이미 작성한 리소스 매개 변수를 찾습니다. 매개 변수 기본값은 모델 리포지토리의 실제 데이터 개체 이름입니다. 기본 매개 변수 값을 작성하는 경우 리포지토리의 데이터 개체 목록에서 실제 데이터 개체 이름을 선택합니다.

다음 이미지는 매개 변수로 데이터 개체를 지정하는 경우 **데이터 개체** 탭을 보여 줍니다.



**데이터 개체** 탭에는 매개 변수 기준 다음 옵션이 있습니다.

#### 매개 변수

데이터 개체로 구성된 리소스 매개 변수의 이름입니다. 읽기 전용입니다.

#### 설명

매개 변수에 대한 설명입니다. 읽기 전용입니다.

#### 새로 만들기

리소스 매개 변수를 작성합니다. 모델 리포지토리에서 매개 변수 기본값에 대한 데이터 개체를 찾아 선택합니다.

#### 찾아보기

리소스 매개 변수를 찾아 매개 변수를 선택합니다.

#### 기본값

데이터 개체에 대해 구성된 리소스 매개 변수의 기본값입니다. 기본값은 모델 리포지토리의 실제 데이터 개체 이름 및 개체에 대한 경로입니다. 읽기 전용입니다.

## 조회 포트와 포트 이름 충돌

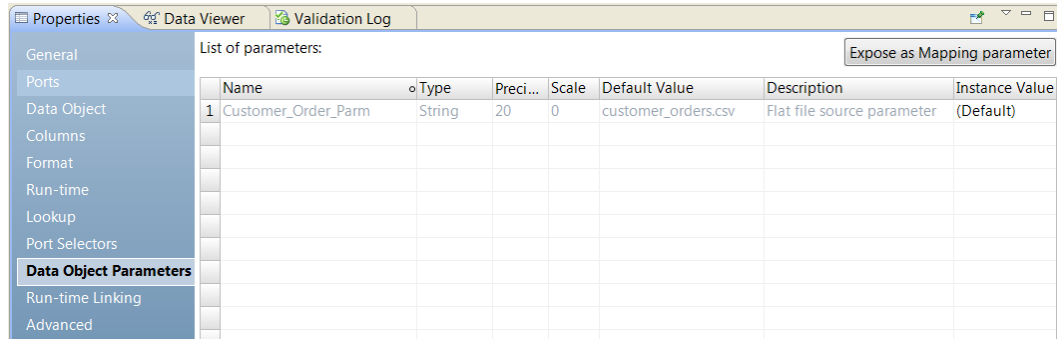
조회 서비스를 매개 변수화할 때 조회 변환 입력 포트가 조회 소스의 포트와 이름이 충돌할 수 있습니다.

조회 변환 입력 포트가 조회 소스의 조회 포트와 이름이 충돌하는 경우 **Developer tool**은 두 포트의 이름을 바꾸지 않습니다. **Developer tool**은 유효성 검사 오류를 표시합니다. 조회 변환의 입력 포트 이름을 변경하거나 해당 포트를 변환에서 제거해야 합니다.

## 매개 변수를 포함하는 조회 소스

매개 변수를 포함하는 실제 데이터 개체에서 조회 소스를 작성할 수 있습니다. 실제 데이터 개체를 매핑에 추가하면 매개 변수가 **데이터 개체 매개 변수** 탭에 나타납니다.

다음 이미지는 조회 변환의 **데이터 개체 매개 변수** 탭을 보여 줍니다.



Customer\_Order\_Parm이 표시되어 있습니다. Customer\_Order\_Parm은 플랫폼 파일 데이터 개체의 소스 파일 이름에 대한 매개 변수입니다. 매핑에서 소스 파일 이름을 재정의하려면 Customer\_Order\_Parm을 매핑의 매개 변수로 바인딩합니다. **매개 변수로 노출**을 클릭하여 매핑에서 중복 매개 변수를 작성합니다.

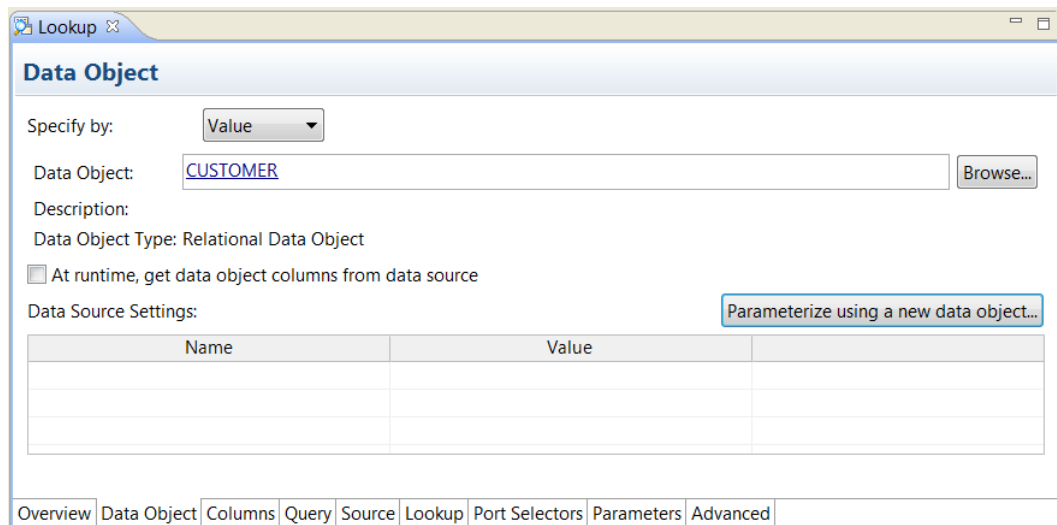
## 중복 데이터 개체의 매개 변수 구성

리포지토리에서 중복 데이터 개체를 작성하고 해당 실제 데이터 개체에 대한 속성을 매개 변수화할 수 있습니다. 연결, 리소스 이름, 테이블 소유자 또는 제어 파일 이름과 같은 속성에 대한 기본값을 정의합니다.

관계형 데이터 개체 및 플랫폼 파일 데이터 개체에 대한 중복 데이터 개체를 작성할 수 있습니다. 재사용 가능 조회 변환 및 재사용 불가능 조회 변환에서 중복 데이터 개체를 작성할 수 있습니다.

조회 변환 **데이터 개체** 탭에서 중복 데이터 개체를 작성합니다. 데이터 개체를 값으로 지정하는 경우 중복 개체를 작성할 수 있습니다. 중복 데이터 개체를 작성하는 경우 조회 변환의 데이터 개체 이름을 중복 데이터 개체 이름으로 바꿉니다. Developer tool에서 데이터 개체 속성에 대한 매개 변수를 작성합니다. Developer tool에서 매개 변수에 대한 기본값을 입력하라는 메시지가 표시됩니다. 중복 데이터 개체 이름 구문은 <원본 개체 이름>\_Param입니다.

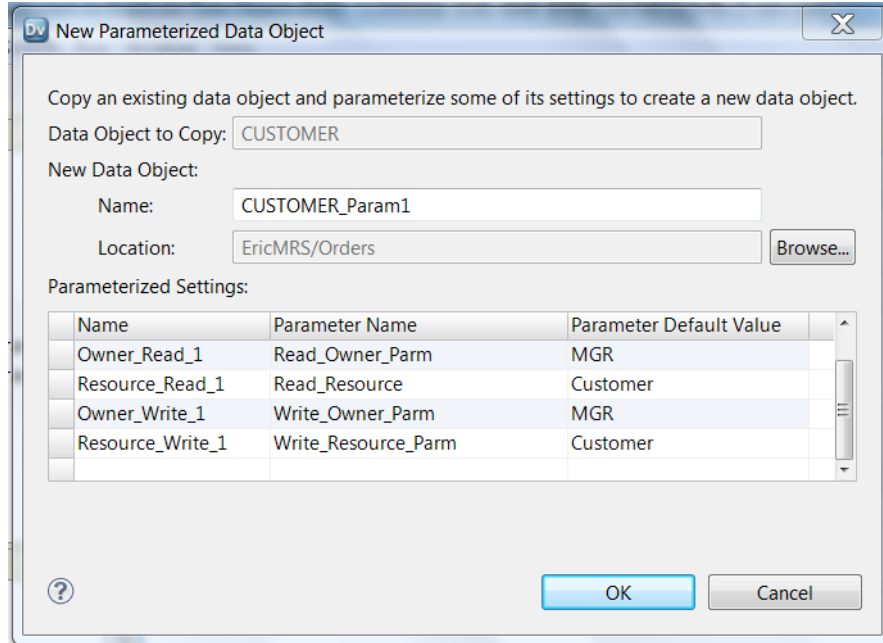
다음 이미지는 관계형 및 플랫폼 파일 데이터 개체에 대한 **데이터 개체** 탭의 새 데이터 개체를 사용하여 매개 변수화 단추를 보여 줍니다.



중복 데이터 개체를 작성하여 데이터 개체를 매개 변수화하면 Developer tool이 데이터 개체에 대한 매개 변수 집합을 작성합니다. 데이터 개체가 플랫 파일 또는 관계형 데이터 개체인지 여부에 따라 Developer tool이 다른 매개 변수를 작성합니다.

중복 데이터 개체를 작성하는 경우 새 매개 변수화된 데이터 개체 대화 상자에서 기본 매개 변수 값을 구성합니다.

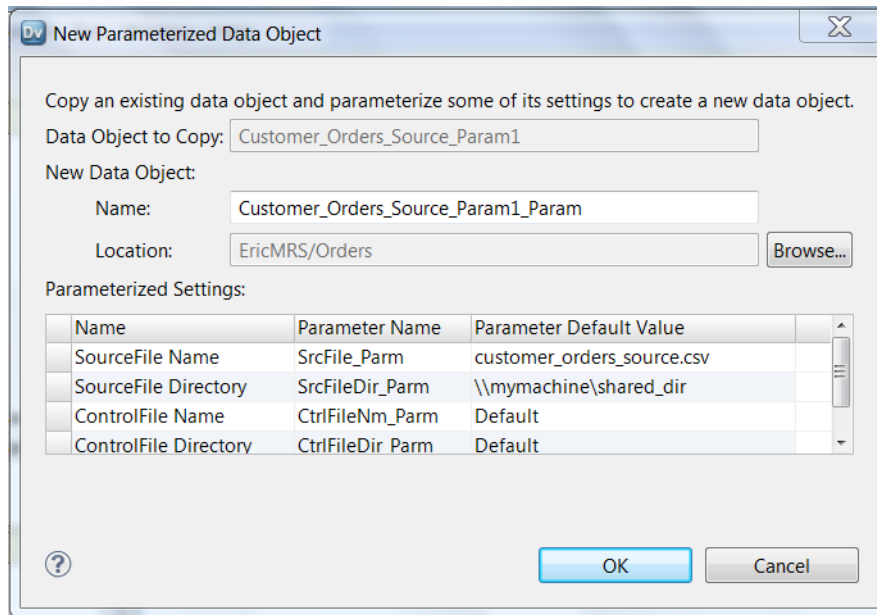
다음 이미지는 관계형 데이터 개체에 대한 새 매개 변수화된 데이터 개체 대화 상자를 보여 줍니다.



데이터 개체의 이름을 변경할 수 있습니다. 소유자 및 리소스 매개 변수에 대한 매개 변수 기본값을 입력합니다.

원본 데이터 개체가 매개 변수화된 경우 Developer tool이 원본 데이터 개체의 매개 변수를 중복 데이터 개체에 복사합니다. 원본 속성이 매개 변수화되지 않은 경우 Developer tool이 중복 데이터 개체에서 해당하는 매개 변수를 작성합니다. Developer tool에서는 원본 속성 값을 중복 데이터 개체의 기본 매개 변수 값으로 사용합니다. Developer tool에서 원본 속성 값을 확인하지 못하는 경우 Developer tool은 매개 변수 유형에 따라 기본값을 사용하여 매개 변수를 작성합니다.

다음 이미지는 플랫 파일 데이터 개체에 대한 새 매개 변수화된 데이터 개체 대화 상자를 보여 줍니다.



소스 파일 및 소스 파일 디렉터리에 대한 기본 매개 변수 값을 구성합니다. 플랫폼 파일에 제어 파일이 있는 경우 제어 파일 이름 및 디렉터를 구성합니다.

기본값을 구성한 후 **Developer tool**이 중복 데이터 개체를 작성합니다. 조회 변환의 **데이터 개체** 탭에 중복 데이터 개체 이름이 나타납니다. **개체 탐색기**에 중복 데이터 개체가 나타납니다.

데이터 개체를 작성한 후 데이터 개체에 대한 매개 변수 값을 변경하려면 **개체 탐색기**에서 실제 데이터 개체를 엽니다. **매개 변수** 탭을 클릭합니다.

## 포트 선택기

조회 변환이 생성된 포트를 포함하는 경우 조회 조건을 작성할 수 있습니다. 조회 조건에서 동적 포트 또는 포트 선택기를 참조할 수 있습니다. 또한 식 매개 변수를 사용하여 전체 조회 식을 매개 변수화할 수 있습니다.

동적 포트가 여러 생성된 포트를 포함하는 경우 조회 조건에서 생성된 포트를 필터링하도록 포트 선택기를 정의할 수 있습니다. 동적 매핑에서 조회 소스가 변경될 수 있습니다. 조회 열에 대해 사용할 포트를 필터링하도록 포트 선택기를 구성할 수 있습니다. 조회 소스 및 포트 선택기는 입력 열 포트 선택기와 동일한 수의 포트를 포함해야 합니다.

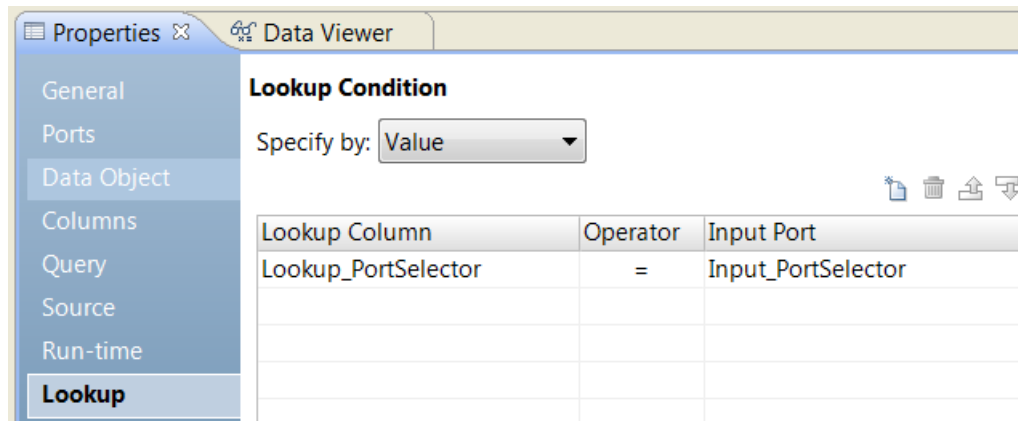
예를 들어 **Lookup\_PortSelector**는 다음 포트를 포함합니다.

```
C_CustKey
C_OrderKey
```

**Input\_PortSelector**는 다음 입력 포트를 포함합니다.

```
CustomerID_IN
OrderID_IN
```

다음 이미지는 포트 선택기가 포함된 조회 조건을 보여 줍니다.



조회 조건이 다음 식으로 확장됩니다.

$C\_CustKey = CustomerID\_IN \text{ AND } C\_OrderKey = OrderID\_IN$

조회 조건이 여러 포트를 포함하는 경우 한 개의 연산자를 구성할 수 있습니다. 예를 들어 연산자를 보다 큼(>)으로 변경할 수 있습니다. 조회 조건이 다음 식으로 확장됩니다.

$C\_CustKey > CustomerID\_IN \text{ AND } C\_OrderKey > OrderID\_IN$

동적 포트를 포함하는 조회 조건을 작성할 수 있습니다.

$Lookup\_PortSelector = Dynamic\_Input\_Port$

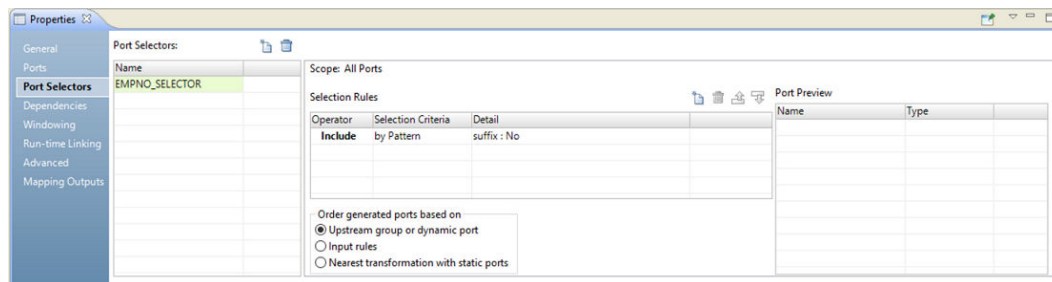
동적 포트에는 포트 선택기와 동일한 수의 포트가 포함되어야 합니다.

## 포트 선택기 구성

포트 선택기를 구성하는 경우 선택 규칙을 정의하여 포함할 생성된 포트를 결정합니다. 선택 규칙은 동적 포트에 대해 구성할 수 있는 입력 규칙과 유사합니다.

포트 선택기는 정적 포트 또는 생성된 포트를 포함할 수 있습니다. **포트 선택기** 탭에서 포트 선택기를 구성합니다.

다음 이미지는 **포트 선택기** 탭을 보여 줍니다.



포트 선택기에 대해 다음 속성을 구성합니다.

### 이름

포트 선택기를 식별합니다. 변환에서 여러 포트 선택기를 작성하고 식에서 참조할 수 있습니다.

### 범위

포트 선택기가 적용되는 포트 그룹을 식별합니다. 조이너 또는 조회 변환에 대한 포트 선택기를 작성하는 경우 범위를 선택해야 합니다. 이러한 변환에는 여러 입력 그룹이 있습니다. 조이너 변환에는 마스터 또는 세



부 정보 범위가 있습니다. 조회 변환에는 가져오기 또는 조회 범위가 있습니다. 식 변환에는 입력 그룹 하나가 있습니다. 범위는 항상 모든 포트입니다.

### 선택 규칙

포트 선택기에 포함될 포트를 결정합니다. 선택 규칙을 작성하면 **포트 미리보기** 패널에 현재 입력 포트에 적합한 포트가 표시됩니다. 이러한 포트는 변경될 수 있습니다. 선택 규칙을 구성하여 여러 소스에서 포트를 포함합니다.

## 선택 규칙

포트 선택기와 연결된 선택 규칙은 포트 선택기에 포함될 포트를 결정합니다.

선택 규칙을 작성하면 **포트 미리보기** 패널에 현재 입력 포트에 적합한 포트가 표시됩니다. 이러한 포트는 변경될 수 있습니다. 선택 규칙을 구성하여 여러 소스에서 포트를 포함합니다.

다음 조건에 따라 선택 규칙을 작성합니다.

### 연산자

선택 규칙이 반환하는 포트를 포함하거나 제외합니다. 기본값은 포함입니다. 포트를 제외하려면 포트를 포함해야 합니다.

### 선택 조건

작성할 선택 규칙 유형입니다. 열 이름, 포트 유형, 패턴 또는 복합 데이터 유형 정의를 기반으로 규칙을 생성할 수 있습니다. 열 이름에 따라 포트를 포함하려면 특정 이름을 검색하거나 이름의 문자 패턴을 검색합니다.

### 세부 정보

선택 조건에 적용할 값입니다. 선택 조건이 열 이름 기준인 경우 검색할 문자열 또는 이름을 구성합니다. 선택 조건이 포트 유형인 경우 포함할 포트 유형을 선택합니다.

다음 표에는 선택 조건과 해당 조건에 대한 세부 정보를 지정하는 방법이 설명되어 있습니다.

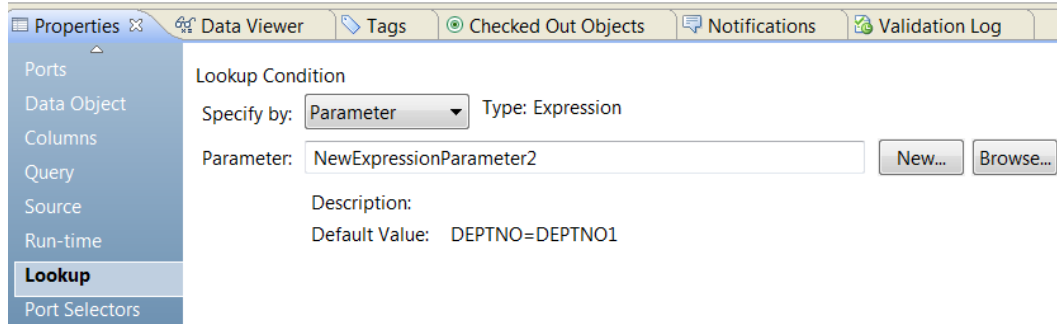
선택 조건	설명	세부 정보
모두	모든 포트를 포함합니다.	세부 정보가 필요하지 않습니다.
이름	포트 이름에 따라 포트를 필터링합니다.	값 목록에서 포트 이름을 선택하거나 포트 또는 포트 목록 유형인 매개 변수를 사용합니다.
유형	각 포트의 데이터 유형에 따라 포트를 필터링합니다.	목록에서 데이터 유형을 선택합니다.
패턴	이름의 문자 문자열 또는 정규식으로 포트를 필터링합니다.	접두사, 접미사 또는 정규식을 포트 이름에 대한 패턴 유형으로 선택합니다. 그런 다음 패턴에 대한 값을 입력하거나 문자열 유형인 매개 변수를 사용합니다.
복합 데이터 유형 정의	복합 데이터 유형 정의를 기준으로 포트를 필터링합니다.	접두사, 접미사 또는 정규식을 복합 데이터 유형 정의에 대한 패턴 유형으로 선택합니다. 그런 다음 패턴에 대한 값을 입력하거나 문자열 유형인 매개 변수를 사용합니다.

## 조회 조건 매개 변수화

조회 조건을 정의하는 식 매개 변수를 구성할 수 있습니다. 식 매개 변수는 식 편집기에서 작성하는 전체 식을 포함합니다. 매핑 매개 변수를 정의하여 런타임 시 식 매개 변수를 재정의할 수 있습니다.

매개 변수를 사용하여 조회 조건을 지정하는 경우 식 매개 변수를 찾거나 매개 변수를 작성할 수 있습니다.

다음 이미지는 조회 조건에 대한 식 매개 변수를 구성하는 위치를 보여 줍니다.



매개 변수를 작성하려면 **새로 만들기**를 클릭합니다. 매개 변수 이름을 정의하고 기본값을 편집합니다. 식 매개 변수 기본값은 조회 조건을 정의하기 위한 전체 식입니다. 식의 생성된 포트, 동적 포트 및 포트 선택기를 사용할 수 있습니다.

**참고:** 식을 작성하는 경우 조회 열은 항상 첫 번째 값이고 입력 열은 두 번째 값입니다.

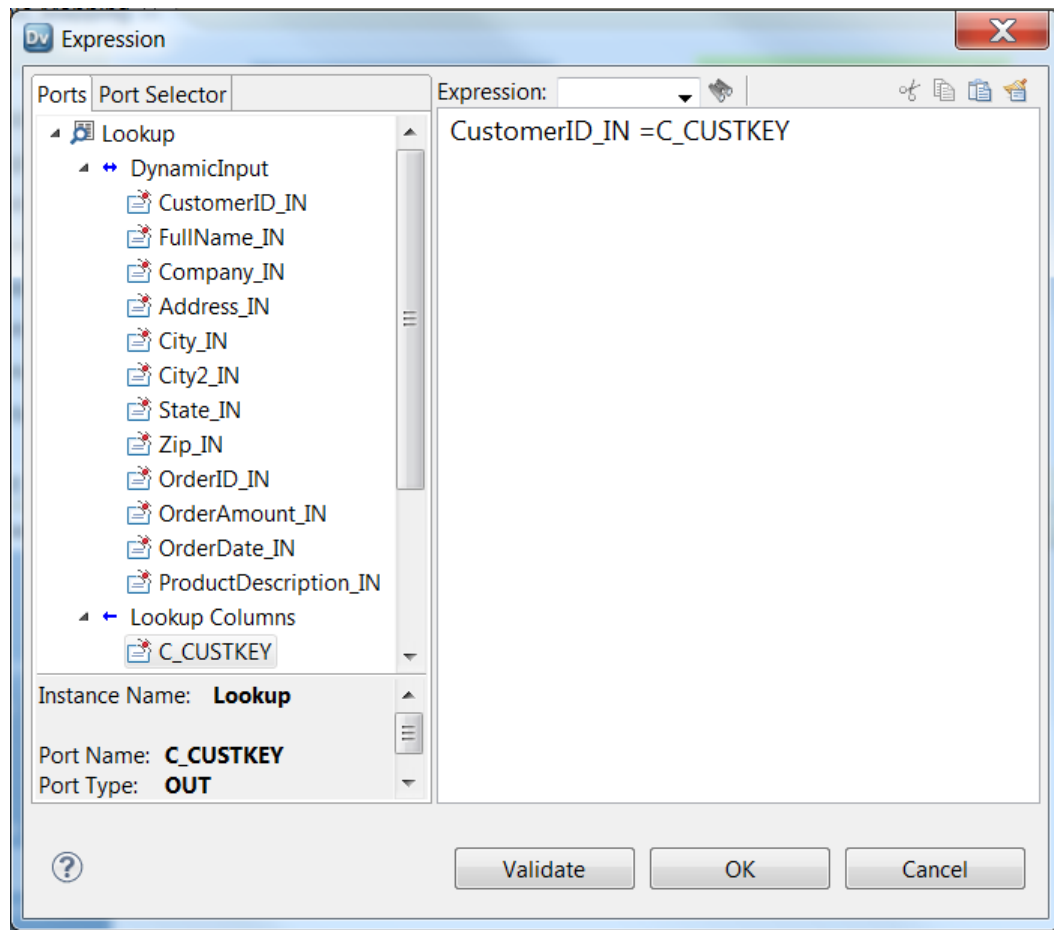
예를 들어 식 매개 변수에서 다음 조회 조건을 작성합니다.

CustomerID\_IN = C\_CUSTKEY

CustomerID\_IN은 조회 열입니다.

C\_CUSTKEY는 입력 열입니다.

다음 이미지는 식 편집기에서 조회 식을 보여 줍니다.



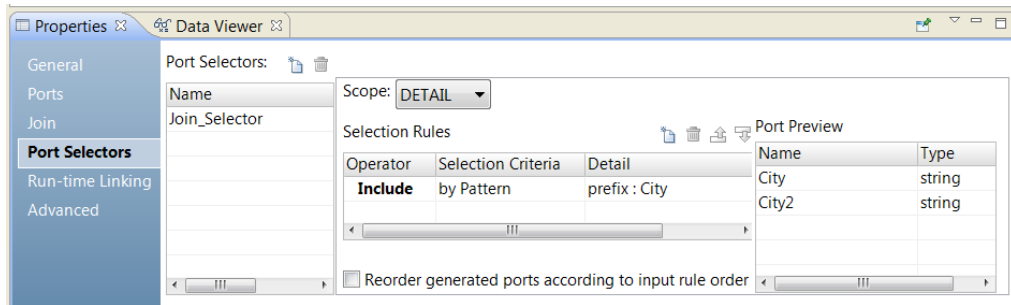
## 포트 선택기 작성

포트 선택기를 작성하여 동적 식, 조회 조건 또는 조이너 조건에서 사용할 포트를 결정합니다.

1. **포트 선택기** 탭을 클릭합니다.
2. **포트 선택기** 영역에서 **새로 만들기**를 클릭합니다.  
Developer tool이 모든 포트를 포함하는 기본 선택 규칙으로 포트 선택기를 작성합니다.
3. **포트 선택기** 영역에서 포트 선택기 이름을 고유 이름으로 변경합니다.
4. 조이너 변환 또는 조회 변환에 대해 작업하는 경우 범위를 선택합니다.  
사용 가능한 포트는 선택하는 포트 그룹에 따라 변경됩니다.
5. **선택 규칙** 영역에서 **연산자**를 선택합니다.
  - 포함. 포트 선택기에 대한 포트를 포함하는 규칙을 작성합니다. 포트를 제외하려면 포트를 포함해야 합니다.
  - 제외. 포트 선택기에서 특정 포트를 제외하는 규칙을 작성합니다.
6. **선택 조건**을 선택합니다.
  - 이름별. 이름으로 특정 포트를 선택합니다. 범위의 포트 목록에서 포트 이름을 선택할 수 있습니다.
  - 유형별. 유형으로 포트를 선택합니다. 데이터 유형을 하나 이상 선택할 수 있습니다.

- 패턴별. 포트 이름의 글자 패턴으로 포트를 선택합니다. 특정 글자로 검색하거나 정규식을 작성할 수 있습니다.

다음 이미지는 포트 선택기 탭을 보여 줍니다.



- 세부 정보 열을 클릭합니다.  
입력 규칙 세부 정보 대화 상자가 나타납니다.
- 포트 기준으로 필터링할 값을 선택합니다.
  - 이름별. 선택하여 값 또는 매개 변수 기준으로 포트 목록을 작성합니다. **선택**을 클릭하여 목록에서 포트를 선택합니다.
  - 유형별. 목록에서 데이터 유형을 하나 이상 선택합니다. **포트 미리보기** 영역에 선택하는 유형의 포트가 표시됩니다.
  - 패턴별. 선택하여 포트 이름의 접두사 또는 접미사에서 문자의 특정 패턴을 검색합니다. 또는 선택하여 검색할 정규식을 작성합니다. 매개 변수를 구성하거나 검색할 패턴을 구성합니다.

**포트 미리보기** 영역에는 규칙을 구성한 포트 선택기의 포트가 표시됩니다..
- 포트 선택기에서 포트 순서를 다시 지정하려면 **생성된 포트의 순서를 입력 규칙 순서에 따라 다시 지정**을 선택합니다.

## 런타임 속성

런타임 속성을 설정하여 조회 캐싱을 활성화하고 구성합니다. 런타임 조회 속성을 구성하려면 먼저 조회 변환을 매핑에 추가해야 합니다.

다음 테이블에는 플랫폼 파일, 참조 테이블 또는 관계형 조회를 수행하는 조회 변환에 대한 런타임 속성이 설명되어 있습니다.

속성	설명
조회 캐싱 설정	<p>통합 서비스가 조회 값을 캐시하는지 여부를 나타냅니다.</p> <p>조회 캐싱을 활성화하는 경우 통합 서비스는 조회 소스를 한 번 쿼리하고 값을 캐시하고 캐시의 값을 조회합니다. 조회 값을 캐싱하면 큰 조회 테이블에서 성능이 향상될 수 있습니다.</p> <p>캐싱을 비활성화하는 경우 행이 변환에 전달될 때마다 통합 서비스가 조회 값의 조회 소스에 대해 select 문을 실행합니다.</p> <p>통합 서비스는 항상 플랫폼 파일 조회를 캐시합니다.</p>
조회 데이터 캐시 크기	<p>매핑 실행 시작 시 변환을 위해 데이터 통합 서비스가 데이터 캐시에 할당하는 메모리의 양입니다. "자동"을 선택하면 데이터 통합 서비스가 런타임 시 자동으로 메모리 요구 사항을 계산합니다. 캐시 크기를 조정하려는 경우 특정 값을 바이트 단위로 입력합니다. 기본값은 자동입니다.</p>

속성	설명
조회 인덱스 캐시 크기	매핑 실행 시작 시 변환을 위해 데이터 통합 서비스가 인덱스 캐시에 할당하는 메모리의 양입니다. "자동"을 선택하면 데이터 통합 서비스가 런타임 시 자동으로 메모리 요구 사항을 계산합니다. 캐시 크기를 조정하려는 경우 특정 값을 바이트 단위로 입력합니다. 기본값은 자동입니다.
캐시 파일 이름 접두사	캐시 파일의 접두사입니다. 지속형 조회 캐시에 대해 캐시 파일 이름 접두사를 지정할 수 있습니다.
사전 빌드 조회 캐시	<p>조회 변환이 데이터를 수신하기 전에 통합 서비스가 조회 캐시를 작성하도록 할 수 있습니다. 통합 서비스가 동시에 여러 조회 캐시 파일을 작성하면 성능을 향상시킬 수 있습니다.</p> <p>다음 옵션 중 하나를 구성합니다.</p> <ul style="list-style-type: none"> <li>- 자동. 통합 서비스가 값을 결정합니다.</li> <li>- 항상 허용. 조회 변환이 데이터를 수신하기 전에 통합 서비스가 조회 캐시를 작성하도록 할 수 있습니다. 통합 서비스가 동시에 여러 조회 캐시 파일을 작성하면 성능을 향상시킬 수 있습니다.</li> <li>- 항상 허용 안 함. 조회 변환이 첫 번째 행을 수신하기 전에 통합 서비스가 조회 캐시를 작성할 수 없습니다.</li> </ul>
조회 캐시 디렉터리 이름	<p>조회 소스를 캐시하도록 조회 변환을 구성하는 경우 조회 캐시 파일을 작성하는 데 사용되는 디렉터리입니다.</p> <p>기본값은 CacheDir 시스템 매개 변수입니다. 이 속성에 대해 다른 시스템 매개 변수를 구성하거나 사용자 정의 매개 변수를 구성할 수 있습니다.</p>
조회 소스에서 다시 캐시	조회 캐시를 재작성하여 지속형 캐시를 조회 테이블과 동기화합니다. 조회 변환에 지속형 조회 캐시가 있고 조회 테이블이 자주 변경되는 경우 데이터베이스에서 조회를 다시 캐시합니다.

## 관련 항목:

- [“캐시 크기” 페이지 71](#)

## 고급 속성

고급 속성에서 지속형 조회 캐시 및 관계형 데이터베이스 연결을 구성하십시오. 조회 소스 유형에 따라 표시되는 속성입니다.

다음 표에는 각 조회 소스 유형에 대한 고급 속성이 설명되어 있습니다.

속성	조회 소스 유형	설명
조회 캐시 지속형	플랫 파일, 참조 테이블, 관계형 조회	통합 서비스가 2개 이상의 캐시 파일로 구성된 지속형 조회 캐시를 사용하는지 여부를 나타냅니다. 조회 변환이 지속형 조회 캐시에 대해 구성되고 지속형 조회 캐시 파일이 없는 경우, 통합 서비스에서 파일을 작성합니다.
대/소문자 구분 문자열 비교	플랫 파일	문자열 열에 대한 조회를 수행할 경우 통합 서비스가 대/소문자 구분 문자열 비교를 사용합니다.

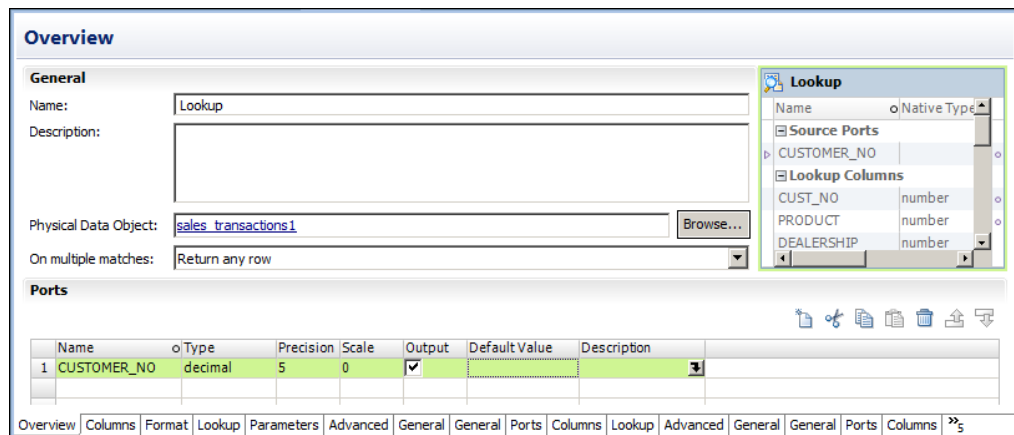
속성	조회 소스 유형	설명
Null 순서	플랫 파일	통합 서비스에서 Null 값 순서를 지정하는 방법을 결정합니다. Null 값을 높게 또는 낮게 정렬하도록 선택할 수 있습니다. 기본적으로 통합 서비스는 Null 값을 높게 정렬합니다. 이 경우 통합 서비스 구성을 재정의하여 비교 연산자의 Null을 높게, 낮게 또는 Null로 처리합니다. 관계형 조회의 경우 Null 순서가 데이터베이스 기본값을 기반으로 합니다.
추적 수준	플랫 파일, 논리적 데이터 개체, 참조 테이블, 관계형 조회	이 변환에 대한 로그에 나타나는 세부 정보 양을 설정합니다. 간단, 보통, 자세한 정보 표시 초기화 또는 자세한 정보 표시 데이터를 선택할 수 있습니다. 기본값은 보통입니다.
업데이트 기타 항목 삽입	참조 테이블, 관계형 조회	동적 조회 캐시에만 적용됩니다. 조회 변환에 입력되는 행 유형이 업데이트이고, 행이 인덱스 캐시에 있으며, 캐시 데이터가 기존 행과 다를 경우 통합 서비스가 캐시의 행을 업데이트합니다. 새로운 행일 경우 통합 서비스가 캐시에 행을 삽입합니다.
삽입 기타 항목 업데이트	참조 테이블, 관계형 조회	동적 조회 캐시에만 적용됩니다. 조회 변환에 입력되는 행 유형이 삽입이고 새로운 행일 경우 통합 서비스가 행을 캐시에 삽입합니다. 행이 인덱스 캐시에 있지만 데이터 캐시가 현재 행과 다른 경우 통합 서비스가 데이터 캐시에 해당 행을 업데이트합니다.
업데이트 시 이전 값 출력	참조 테이블, 관계형 조회	통합 서비스가 행을 업데이트하기 전에 캐시에 있던 값을 출력합니다. 그렇지 않은 경우 통합 서비스가 캐시에 쓰는 업데이트된 값을 출력합니다.
동적 캐시 조건 업데이트	참조 테이블, 관계형 조회	동적 조회 캐시에만 적용됩니다. 동적 캐시를 업데이트할지 여부를 나타내는 식입니다. 조건이 true이고 캐시에 데이터가 있는 경우 통합 서비스가 캐시를 업데이트합니다. 기본값은 true입니다.
연결	참조 테이블, 관계형 조회	관계형 조회 소스를 포함하는 관계형 데이터베이스에 대한 연결입니다. 연결에 대해 매개 변수를 사용할 수 있습니다. 참조 테이블 조회의 경우 이 필드가 읽기 전용입니다.
정렬된 입력	플랫 파일	입력 데이터가 그룹을 기준으로 미리 정렬되었음을 나타냅니다.
날짜/시간 형식	플랫 파일	날짜/시간 형식 및 필드 너비를 정의합니다. 밀리초, 마이크로초 또는 나노초 형식의 필드 너비는 29입니다. 포트에 대한 날짜/시간 형식을 선택하지 않은 경우 날짜/시간 형식을 입력할 수 있습니다. 기본값은 YYYY-MM-DD HH24:MI:SS입니다. 날짜/시간 형식은 포트 크기를 변경하지 않습니다. 이 필드는 읽기 전용입니다.
1000 단위 구분 기호	플랫 파일	값은 없음입니다. 이 필드는 읽기 전용입니다.
소수 구분 기호	플랫 파일	값은 마침표(.)입니다. 이 필드는 읽기 전용입니다.

## 재사용 가능한 조회 변환 작성

플랫 파일, 논리적 데이터 개체, 참조 테이블 또는 관계형 데이터 개체에서 데이터를 조회하려면 조회 변환을 작성하십시오.

1. 개체 탐색기 보기에 프로젝트나 폴더를 선택합니다.
2. 파일 > 새로 만들기 > 변환을 클릭합니다.
3. 조회 마법사를 찾습니다.
4. 플랫 파일 데이터 개체 조회, 논리적 데이터 개체 조회, 참조 테이블 조회 또는 관계형 데이터 개체 조회를 선택합니다.
5. 플랫 파일 데이터 개체 조회, 논리적 데이터 개체 조회 또는 관계형 데이터 개체 조회를 선택합니다.
6. 다음을 클릭합니다.  
새 조회 변환 대화 상자가 나타납니다.
7. Developer tool에서 실제 데이터 개체 또는 참조 테이블을 선택합니다.
8. Developer tool에서 실제 데이터 개체를 선택합니다.
9. 변환 이름을 입력합니다.
10. 일치하는 항목이 여러 개인 경우에서 조회 조건과 일치하는 여러 행을 찾은 경우 조회 변환에서 반환하는 행을 결정합니다.
11. 마침을 클릭합니다.  
조회 변환이 편집기에 표시됩니다.
12. 개요 보기의 포트 섹션에서 출력 포트를 변환에 추가합니다.

다음 그림에서는 조회 변환의 CUSTOMER\_NO 출력 포트를 보여 줍니다.



13. 속성 보기의 런타임 탭에서 조회 캐싱 설정을 선택하여 조회 캐싱을 활성화합니다.  
참고: 런타임 조회 속성을 구성하려면 먼저 조회 변환을 매핑에 추가해야 합니다.
14. 속성 보기의 조회 탭에서 하나 이상의 조회 조건을 추가합니다.
15. 속성 보기의 고급 탭에서 추적 수준, 동적 조회 캐시 속성 및 런타임 연결을 구성합니다.
16. 변환을 저장합니다.

## 재사용 불가능한 조회 변환 작성

매핑 또는 맵렛에서 재사용 불가능한 조회 변환을 작성하십시오.

1. 매핑 또는 맵렛에서 조회 변환을 변환 팔레트에서 편집기로 끄니다.  
새로 만들기 대화 상자가 나타납니다.
2. 플랫폼 파일 데이터 개체 조회, 논리적 데이터 개체 조회, 참조 테이블 조회 또는 관계형 데이터 개체 조회를 선택합니다.
3. 플랫폼 파일 데이터 개체 조회, 논리적 데이터 개체 조회 또는 관계형 데이터 개체 조회를 선택합니다.
4. 다음을 클릭합니다.  
새 조회 변환 대화 상자가 나타납니다.
5. Developer tool에서 실제 데이터 개체 또는 참조 테이블을 선택합니다.
6. Developer tool에서 실제 데이터 개체를 선택합니다.
7. 변환 이름을 입력합니다.
8. 일치하는 항목이 여러 개인 경우에서 조회 조건과 일치하는 여러 행을 찾은 경우 조회 변환에서 반환하는 행을 결정합니다.
9. 마침을 클릭합니다.  
조회 변환이 편집기에 표시됩니다.
10. 편집기에서 조회 변환을 선택합니다.  
변환 위에 도구 모음이 표시됩니다.
11. 속성 보기의 포트 탭에서 출력 포트를 변환에 추가합니다.  
다음 그림에서는 조회 변환의 CUSTOMER\_NO 출력 포트를 보여 줍니다.

Properties							
Data Viewer Tags Validation Log Navigator							
General							
Ports	Name	Type	Precision	Scale	Output	DefaultValue	Description
1	CUSTOMER_NO	decimal	5	0	<input checked="" type="checkbox"/>		

12. 속성 보기의 런타임 탭에서 조회 캐싱 설정을 선택하여 조회 캐싱을 활성화합니다.  
참고: 런타임 조회 속성을 구성하려면 먼저 조회 변환을 매핑에 추가해야 합니다.
13. 속성 보기의 조회 탭에서 하나 이상의 조회 조건을 추가합니다.
14. 속성 보기의 고급 탭에서 추적 수준, 동적 조회 캐시 속성 및 런타임 연결을 구성합니다.
15. 변환을 저장합니다.



## 연결되지 않은 조회 변환 작성

식에서 조회를 수행할 경우 연결되지 않은 조회 변환을 작성합니다. 플랫폼 파일, 참조 테이블 또는 관계형 데이터 개체에서 재사용 가능하거나 재사용 불가능한 연결되지 않은 조회 변환을 작성할 수 있습니다.

1. **개체 탐색기** 보기에서 프로젝트나 폴더를 선택합니다.
2. **파일 > 새로 만들기 > 변환**을 클릭합니다.
3. 조회 마법사를 찾습니다.
4. **플랫 파일 데이터 개체 조회**, **참조 테이블 조회** 또는 **관계형 데이터 개체 조회**를 선택합니다.
5. **다음**을 클릭합니다.  
새 **조회** 대화 상자가 나타납니다.
6. **Developer tool**에서 실제 데이터 개체 또는 참조 테이블을 선택합니다.
7. 변환 이름을 입력합니다.
8. **일치하는 항목이 여러 개인 경우**에서 조회 조건과 일치하는 여러 행을 찾은 경우 조회 변환에서 반환하는 행을 결정합니다. 연결되지 않은 조회에 대해 **모든 행 반환**을 선택하지 마십시오.
9. **마침**을 클릭합니다.  
조회 변환이 편집기에 표시됩니다.
10. **개요** 보기의 **포트** 섹션에서 포트를 변환에 추가합니다.  
:LKP 식의 각 인수에 대한 입력 포트를 작성합니다. 작성하는 조회 조건마다 입력 포트를 작성합니다. 여러 조건에서 하나의 입력 포트를 사용할 수 있습니다.
11. **개요** 보기의 **포트** 섹션에서 포트 1개를 반환 포트에 구성합니다.
12. **조회** 보기에서 변환 입력 값을 조회 소스 또는 캐시의 값과 비교하려면 하나 이상의 조회 조건을 추가합니다.  
조건이 **True**일 경우 조회에서 반환 포트에 값을 반환합니다. 조회 조건이 **false**일 경우 조회에서 **NULL**을 반환합니다.
13. 식을 허용하는 변환(예: 집계 변환, 식 변환 또는 업데이트 전략 변환)에서 포트에 대한 :LKP 식을 작성합니다.
14. 매핑을 작성할 경우 편집기에서 연결되지 않은 조회 변환을 매핑에 추가하십시오. 그러나 포트를 매핑의 다른 변환에 연결하지는 마십시오.

## 연결되지 않은 조회 예제

캘리포니아의 소매업체가 주 내에서 고객에게 판매하는 품목의 각 가격에 주 판매세를 추가합니다. 세금액은 고객의 거주 카운티를 기준으로 합니다. 판매세를 검색하려면 카운티 이름을 받은 다음 해당 카운티에 대한 판매세액을 반환하는 조회 변환을 생성합니다. 판매세를 부과하지 않는 카운티의 경우 조회 변환은 **NULL**을 반환합니다. 식 변환에서 조회를 호출합니다.

카운티 기준 판매세의 연결되지 않은 조회를 구성하려면 다음 단계를 완료하십시오.

1. 카운티별 판매세액을 포함하는 플랫폼 파일 실제 데이터 개체를 가져옵니다.
2. 연결되지 않은 조회 변환을 생성합니다.
3. 입력 포트를 조회 변환에 추가합니다.
4. 반환 포트를 정의합니다.

5. 조회 조건을 생성합니다.
6. 식 변환에서 조회를 호출합니다.

### 1단계. 판매세 조회 소스를 모델 리포지토리로 가져옵니다.

조회 변환을 생성하기 전에 판매세 파일을 모델 리포지토리에 저장해야 합니다. 이 시나리오에서 판매세 파일에는 Sales\_County 및 County\_SalesTax의 두 필드가 포함되어 있습니다. 카운티는 카운티 이름을 포함하는 문자열입니다. County\_SalesTax는 카운티의 세율을 포함하는 10진수 필드입니다. 판매세 파일이 조회 소스입니다.

### 2단계. 연결되지 않은 조회 변환 생성

판매세 플랫폼 파일 데이터 개체를 사용하여 재사용 가능 플랫폼 파일 조회 변환을 생성합니다. 이 시나리오의 변환 이름은 Sales\_Tax\_Lookup입니다. 여러 항목이 일치하는 경우에는 첫 번째 행 반환을 선택합니다.

### 3단계. 조회 변환 포트 정의

속성 보기의 포트 탭에서 조회 변환 포트를 정의합니다.

포트 유형	이름	유형	길이	배율
입력	In_County	문자열	25	
출력	SalesTax	10진수	3	3

### 4단계. 조회 변환 반환 포트 구성

반환 포트는 조회에서 검색하는 플랫폼 파일의 필드입니다. 열 탭에서 반환 포트는 County\_SalesTax 열입니다.

조회 결과가 true이면 통합 서비스는 플랫폼 파일 소스에서 카운티를 찾습니다. 통합 서비스는 반환 포트에서 판매세 값을 반환합니다. 통합 서비스에서 카운티를 찾을 수 없으면 조회 결과는 false가 되며 통합 서비스는 반환 포트에서 NULL을 반환합니다.

### 5단계. 조회 조건 정의

조회 보기에서 입력 값을 조회 소스의 값과 비교할 조회 조건을 정의합니다.

조회 조건을 추가하려면 조회 열을 클릭합니다.

조회 조건의 구문은 다음과 같습니다.

SALES\_COUNTY = IN\_COUNTY

### 6단계. 식 변환 생성

플랫폼 파일에서 판매 레코드를 받는 식 변환을 생성합니다. 식 변환은 고객 번호, 판매액 및 판매 카운티를 받습니다. 이 변환은 고객 번호, 판매액 및 판매세를 반환합니다.

식 변환에는 다음과 같은 포트가 있습니다.

포트 유형	이름	유형	길이	전체 자릿수	기본값
입력	카운티	문자열	25	10	
통과	Customer	문자열	10		
통과	SalesAmt	10진수	10	2	
출력	SalesTax	10진수	10	2	0

SalesTax 포트는 :LKP 식을 포함합니다. 식은 Sales\_Tax\_Lookup 변환을 포함하며 카운티 이름을 매개 변수로 전달합니다. Sales\_Tax\_Lookup 변환은 판매세율을 식에 반환합니다. 식 변환은 세율에 판매액을 곱합니다.

SalesTax 포트에 대해 다음 식을 입력합니다.

(:LKP.Sales\_Tax\_Lookup(County) \* SalesAmt)

SalesTax 포트는 식 결과를 포함합니다. 조회가 실패하면 조회 변환은 NULL을 반환하며 SalesTax 포트는 null 값을 포함합니다.

SalesTax 포트에 null 값을 확인할 식을 추가할 수 있습니다. SalesTax가 NULL이면 0을 반환하도록 SalesTax 포트를 구성할 수 있습니다. null 값을 확인하고 0을 반환하도록 하려면 조회 식에 다음 텍스트를 추가합니다.

```
IIF(ISNULL(:LKP.Sales_Tax_Lookup(County) * SalesAmt),0, SalesTax)
```

## 조회 변환 - 비원시 환경

비원시 환경에서 조회 변환의 처리는 변환을 실행하는 엔진에 따라 다릅니다.

다음과 같은 비원시 런타임 엔진에 대한 지원을 고려하십시오.

- **Blaze** 엔진. 제한적으로 지원됩니다.
- **Spark** 엔진. 제한적으로 지원됩니다.
- **Databricks Spark** 엔진. 제한적으로 지원됩니다.

### 조회 변환 - Blaze 엔진

Blaze 엔진의 일부 처리 규칙은 데이터 통합 서비스의 처리 규칙과 다릅니다.

다음과 같은 상황에서는 매핑 유효성 검사가 실패합니다.

- 캐시가 공유, 명명, 지속, 동적 또는 캐시되지 않도록 구성되어 있습니다. 캐시는 정적 캐시여야 합니다.

Sqoop를 조회 변환으로 사용하는 데이터 개체를 매핑에 추가하는 경우 데이터 통합 서비스는 Sqoop를 통해 매핑을 실행하지 않습니다. JDBC를 통해 매핑을 실행합니다.

### 조회 변환 - Spark 엔진

Spark 엔진의 일부 처리 규칙은 데이터 통합 서비스의 처리 규칙과 다릅니다.

#### 매핑 유효성 검사

다음과 같은 상황에서는 매핑 유효성 검사가 실패합니다.

- 대/소문자 구분이 비활성화되어 있습니다.
- 조회 변환의 조회 조건에 **Binary** 데이터 유형이 포함되어 있습니다.
- 캐시가 공유, 명명, 지속, 동적 또는 캐시되지 않도록 구성되어 있습니다. 캐시는 정적 캐시여야 합니다.

다음과 같은 상황에서는 매핑이 실패합니다.

- 변환이 연결되지 않았고 조이너 또는 **Java** 변환과 함께 사용됩니다.

#### 여러 개의 일치 항목

여러 개의 일치 항목에서 첫 번째 값, 마지막 값 또는 임의의 값을 반환하도록 선택하는 경우 조회 변환은 임의의 값을 반환합니다.

여러 개의 일치 항목에 대해 오류를 보고하도록 변환을 구성하는 경우 **Spark** 엔진은 중복 행을 삭제하고 이러한 행을 로그에 포함하지 않습니다.

## 조희 변환 - Databricks Spark 엔진

Databricks Spark 엔진의 일부 처리 규칙은 데이터 통합 서비스의 처리 규칙과 다릅니다.

### 매핑 유효성 검사

다음과 같은 상황에서는 매핑 유효성 검사가 실패합니다.

- 대/소문자 구분이 비활성화되어 있습니다.
- 조희 변환의 조희 조건에 이진 데이터 유형이 포함되어 있습니다.
- 캐시가 공유, 명명, 지속, 동적 또는 캐시되지 않도록 구성되어 있습니다. 캐시는 정적 캐시여야 합니다.
- 조희 소스가 Microsoft Azure SQL 데이터 웨어하우스가 아닙니다.

다음과 같은 상황에서는 매핑이 실패합니다.

- 변환이 연결되지 않았고 조이너 변환과 함께 사용됩니다.

### 여러 개의 일치 항목

여러 개의 일치 항목에서 첫 번째 값, 마지막 값 또는 임의의 값을 반환하도록 선택하는 경우 조희 변환은 임의의 값을 반환합니다.

여러 개의 일치 항목에 대해 오류를 보고하도록 변환을 구성하는 경우 Spark 엔진은 중복 행을 삭제하고 이러한 행을 로그에 포함하지 않습니다.

## 제 26 장

# 조회 캐시

이 장에 포함된 항목:

- [조회 캐시 개요, 401](#)
- [조회 캐시 유형, 402](#)
- [캐싱되지 않은 조회, 402](#)
- [정적 조회 캐시, 403](#)
- [지속형 조회 캐시, 403](#)
- [동적 조회 캐시, 404](#)
- [공유 조회 캐시, 404](#)
- [캐시 비교, 406](#)
- [조회에 대한 캐시 분할, 406](#)

## 조회 캐시 개요

조회 변환을 구성하여 관계형 또는 플랫폼 파일 조회 소스를 캐시할 수 있습니다. 크기가 큰 조회 테이블 또는 파일에서 조회 캐싱을 활성화하면 조회 성능이 향상됩니다.

통합 서비스는 캐시된 조회 변환에서 데이터의 첫 번째 행을 처리할 때 캐시를 메모리에 빌드합니다. 통합 서비스는 소스 행이 조회 변환을 시작할 때 캐시를 작성합니다. 통합 서비스는 사용자가 조회 변환에 구성한 양에 따라 캐시의 메모리를 할당합니다. 통합 서비스는 조건 값을 인덱스 캐시에 저장하고 출력 값을 데이터 캐시에 저장합니다. 통합 서비스는 조회 변환을 시작하는 각 행에 대해 캐시를 쿼리합니다.

메모리 캐시가 부족할 경우 통합 서비스가 캐시 파일에 오버플로우 값을 저장합니다. 통합 서비스는 캐시 파일을 캐시 디렉터리에 작성합니다. 기본적으로 통합 서비스는 **CacheDir** 시스템 매개 변수에 지정된 디렉터리에 캐시 파일을 작성합니다. 매핑이 완료되면 지속형 캐시를 사용하도록 조회 변환을 구성한 경우를 제외하고 통합 서비스가 캐시 메모리를 해제하고 캐시 파일을 삭제합니다.

플랫폼 파일 조회를 사용하는 경우 통합 서비스는 조회 소스를 캐시합니다. 정렬된 입력을 사용하도록 플랫폼 파일 조회를 구성한 경우 조건 열이 그룹화되지 않으면 통합 서비스가 조회를 캐시하지 못합니다. 열이 그룹화되었지만 정렬되지 않은 경우 통합 서비스는 정렬된 입력이 구성되지 않은 것처럼 조회를 처리합니다.

조회 변환에 캐싱을 구성하지 않은 경우 통합 서비스가 각 입력 행에 대해 조회 소스를 쿼리합니다. 조회 소스의 캐시 여부와 관계없이 조회 쿼리 및 처리 결과는 동일합니다. 그러나 조회 캐싱을 활성화하면 크기가 큰 조회 소스의 조회 성능이 개선될 수 있습니다.

## 조회 캐시 유형

다른 유형의 조회 캐시를 구성할 수 있습니다. 예를 들어, 동일한 매핑의 여러 조회 변환 사이에서 캐시를 공유할 경우 공유 캐시를 구성할 수 있습니다.

다음 유형의 조회 캐시를 구성할 수 있습니다.

### 정적 캐시

정적 캐시는 통합 서비스가 조회를 처리하는 동안 변경되지 않습니다. 통합 서비스가 조회를 처리할 때마다 정적 캐시를 다시 작성합니다. 기본적으로 조회 변환에 대한 캐싱을 활성화한 경우 통합 서비스에서 정적 캐시를 작성합니다. 통합 서비스가 첫 번째 조회 요청을 처리할 때 캐시를 작성합니다. 조회 변환에 수신되는 각 행의 캐시에서 값을 조회합니다. 조회 조건이 **True**일 경우 통합 서비스가 조회 캐시의 값을 반환합니다.

다음과 같은 이유에는 정적 캐시를 사용합니다.

- 매핑이 실행되는 동안 조회 소스가 변경되지 않습니다.
- 조회가 연결되지 않은 조회입니다. 연결되지 않은 조회에 대해 정적 캐시를 사용해야 합니다.
- 성능을 개선하고자 합니다. 통합 서비스에서 조회 변환을 처리하는 동안 캐시를 업데이트하지 않기 때문에, 통합 서비스는 동적 캐시가 있는 조회 변환보다 정적 캐시가 있는 조회 변환을 더 빠르게 처리합니다.
- 조회 조건이 **false**일 경우 통합 서비스가 연결된 변환에 대해서는 기본값을 반환하고 연결되지 않은 변환에 대해서는 **NULL**을 반환하도록 하려고 합니다.

### 지속형 캐시

지속형 캐시는 통합 서비스가 조회를 처리할 때마다 변경되지 않습니다. 통합 서비스가 조회 캐시 파일을 저장하고 다음에 캐시를 사용하도록 구성된 조회 변환을 처리할 때 조회 캐시 파일을 다시 사용합니다. 조회 소스가 변경되지 않는 경우 지속형 캐시를 사용합니다.

필요한 경우 지속형 조회 캐시를 다시 작성하도록 조회 변환을 구성할 수 있습니다.

### 동적 캐시

통합 서비스가 조회를 처리하는 동안 동적 조회 캐시는 변경됩니다. 통합 서비스가 첫 번째 조회 요청을 처리할 때 동적 조회 캐시를 작성합니다. 통합 서비스가 각 행을 처리할 때 데이터를 조회 캐시에 동적으로 삽입 또는 업데이트하고 해당 데이터를 대상에 전달합니다. 동적 캐시는 대상과 동기화됩니다.

새로운 레코드와 변경된 레코드를 기반으로 대상을 업데이트할 경우 동적 캐시를 사용하십시오. 매핑에 대상 데이터에 대한 조회가 필요하지만 대상 연결이 느릴 경우에도 동적 캐시를 사용할 수 있습니다.

### 공유 캐시

동일한 매핑의 여러 조회 변환에서 공유 캐시를 사용할 수 있습니다. 매핑 성능을 개선하려면 공유 캐시를 사용하십시오. 각 조회 변환마다 별도의 조회 캐시를 생성하는 대신, 통합 서비스에서 하나의 캐시를 생성합니다.

## 캐싱되지 않은 조회

캐싱되지 않은 조회는 통합 서비스가 조회 소스를 캐시하지 않는 경우 발생합니다. 기본적으로 통합 서비스는 조회 변환에 조회 캐시를 사용하지 않습니다.

통합 서비스는 조회 캐시를 구성하고 쿼리하는 대신 조회 소스를 쿼리하는 경우를 제외하고 캐시된 조회를 처리하는 것과 동일한 방식으로 캐싱되지 않은 조회를 처리합니다.

조회 조건이 **true**이면 통합 서비스가 조회 소스에서 값을 반환합니다. 통합 서비스가 연결된 조회 변환을 처리하는 경우 조회/출력 포트가 나타내는 값을 반환합니다. 통합 서비스가 연결되지 않은 조회 변환을 처리하는 경우 반환 포트가 나타내는 값을 반환합니다.

조건이 **true**가 아닌 경우 통합 서비스가 **NULL** 또는 기본값을 반환합니다. 통합 서비스가 연결된 조회 변환을 처리하는 경우 조건이 충족되지 않으면 출력 포트의 기본값을 반환합니다. 연결되지 않은 조회 변환을 처리하는 경우 조건이 충족되지 않으면 통합 서비스가 **NULL**을 반환합니다.

## 정적 조회 캐시

정적 조회 캐시는 통합 서비스에서 조회 변환을 처리할 때 업데이트하지 않는 캐시입니다. 캐싱에 대해 조회 변환을 구성할 때 통합 서비스는 기본적으로 정적 조회 캐시를 생성합니다.

통합 서비스가 첫 번째 조회 요청을 처리할 때 캐시를 작성합니다. 변환에 전달되는 각 행의 조회 조건을 바탕으로 캐시를 쿼리합니다.

조회 조건이 **true**이면 통합 서비스는 정적 조회 캐시에서 값을 반환합니다. 통합 서비스가 연결된 조회 변환을 처리하는 경우 조회/출력 포트가 나타내는 값을 반환합니다. 통합 서비스는 연결되지 않은 조회 변환을 처리할 때 반환 포트에 표시되는 값을 반환합니다.

조건이 **true**가 아닌 경우 통합 서비스가 **NULL** 또는 기본값을 반환합니다. 통합 서비스는 연결된 조회 변환을 처리할 때 조건이 충족되지 않으면 출력 포트의 기본값을 반환합니다. 통합 서비스는 연결되지 않은 조회 변환을 처리할 때 조건이 충족되지 않으면 **NULL**을 반환합니다.

## 지속형 조회 캐시

지속형 조회 캐시는 통합 서비스가 동일한 매핑의 여러 실행에 대해 재사용하는 캐시입니다. 매핑 실행 사이에 조회 소스가 변경되지 않는 경우 지속형 조회 캐시를 사용합니다.

기본적으로 통합 서비스는 조회 변환에서 조회 캐시가 활성화된 경우 비지속형 캐시를 사용합니다. 매핑이 완료되면 통합 서비스가 캐시 파일을 삭제합니다. 다음에 매핑을 실행할 때 통합 서비스는 조회 소스에서 메모리 캐시를 작성합니다.

지속형 조회 캐시를 사용하도록 조회 변환을 구성하는 경우 통합 서비스가 여러 번의 매핑 실행을 위해 캐시 파일을 저장하고 재사용합니다. 지속형 캐시를 사용하면 조회 테이블을 읽고 조회 캐시를 재작성하는 데 필요한 시간이 소요되지 않습니다.

통합 서비스가 지속형 조회 캐시와의 매핑을 처음 실행할 때 통합 서비스는 캐시 파일을 디스크에 저장합니다. 다음에 매핑을 실행할 때 통합 서비스는 캐시 파일에서 메모리 캐시를 작성합니다.

원래 조회 소스가 변경되는 경우 지속형 조회 캐시를 재작성하도록 통합 서비스를 구성할 수 있습니다. 캐시를 재작성하는 경우 통합 서비스가 새 캐시 파일을 작성하고 메시지를 통합 서비스 로그에 씁니다.

## 지속형 조회 캐시 재작성

지속형 조회 캐시를 재작성하도록 통합 서비스를 구성할 수 있습니다. 지속형 조회 캐시를 재작성하도록 통합 서비스를 구성하지 않는 경우에도 통합 서비스가 지속형 조회 캐시를 재작성하는 경우도 있습니다.

지속형 조회 캐시를 재작성하는 경우 다음 규칙 및 지침을 고려하십시오.

- 통합 서비스가 캐시를 마지막으로 작성한 이후 조회 소스가 변경된 경우 지속형 조회 캐시를 재작성합니다.
- 매핑이 캐시를 공유하는 하나 이상의 조회 변환을 포함하는 경우 캐시를 재작성할 수 있습니다.

- 조회 테이블이 세션 간에 변경되지 않는 경우 지속형 조회 캐시를 사용하도록 조회 변환을 구성합니다. 통합 서비스가 캐시 파일을 저장하고 재사용하면 조회 테이블을 읽는 데 필요한 시간이 소요되지 않습니다.
- 조회 테이블이 매핑 실행 간에 변경되지 않는 경우 지속형 조회 캐시를 사용하도록 조회 변환을 구성합니다. 통합 서비스가 캐시 파일을 저장하고 재사용하면 조회 테이블을 읽는 데 필요한 시간이 소요되지 않습니다.
- 조회 캐시를 재작성하도록 후속 조회 변환을 구성하는 경우 통합 서비스가 후속 조회 변환을 처리할 때 캐시를 재작성하는 대신 캐시를 공유합니다.
- 매핑에 두 가지 지속형 조회가 포함되어 있고 캐시를 재작성하도록 두 번째 조회 변환을 구성하는 경우 통합 서비스가 두 조회 변환에 대해 지속형 조회 캐시를 모두 재작성합니다.

통합 서비스는 다음 시나리오에서 지속형 조회 캐시를 재작성합니다.

- 통합 서비스가 캐시 파일을 찾을 수 없습니다.
- 통합 서비스가 캐시를 재사용할 수 없습니다. 이 인스턴스에서 조회 캐시를 재작성하거나 매핑이 실패합니다.
- 통합 서비스에 대해 많은 전체 자릿수를 활성화하거나 비활성화합니다.
- 조회 변환 또는 매핑을 편집합니다.

**참고:** 변환 설명을 편집하는 경우 통합 서비스가 캐시를 재작성하지 않습니다.

- 파티션 수를 변경합니다.
- 조회 소스에 액세스하기 위해 사용되는 파일 위치 또는 데이터베이스 연결을 변경합니다.
- 유니코드 모드에서 정렬 순서를 변경합니다.
- 통합 서비스 코드 페이지를 변경합니다.

## 동적 조회 캐시

동적 캐시는 통합 서비스가 각 행을 처리할 때 업데이트하는 캐시입니다. 동적 조회 캐시를 사용하면 캐시와 대상의 동기화 상태를 유지할 수 있습니다.

관계형 조회 및 플랫폼 파일 조회와 함께 동적 캐시를 사용할 수 있습니다. 통합 서비스가 첫 번째 조회 요청을 처리할 때 캐시를 작성합니다. 조회 변환에 전달된 각 행의 조회 조건을 기반으로 캐시를 쿼리합니다. 통합 서비스는 각 행을 처리할 때 조회 캐시를 업데이트합니다.

조회 쿼리 결과, 행 유형 및 조회 변환 속성을 기반으로 통합 서비스가 캐시에 행을 삽입 또는 업데이트하거나, 캐시를 변경하지 않습니다.

## 공유 조회 캐시

공유 조회 캐시는 매핑의 여러 조회 변환에서 공유되는 정적 조회 캐시입니다. 공유 조회 캐시를 사용하여 캐시를 작성하는 데 필요한 시간을 줄일 수 있습니다.

기본적으로 통합 서비스는 호환되는 캐시 구조가 있는, 매핑의 조회 변환에 대한 캐시를 공유합니다. 예를 들어 매핑이 동일한 재사용 가능 조회 변환의 두 인스턴스를 한 매핑에 포함하고 두 인스턴스에 대해 동일한 출력 포트를 사용하는 경우 기본적으로 조회 변환이 조회 캐시를 공유합니다.

통합 서비스가 첫 번째 조회 변환을 처리할 때 캐시를 작성합니다. 동일한 캐시를 사용하여 캐시를 공유하는 후속 조회 변환을 처리합니다. 통합 서비스가 조회 캐시를 공유하는 경우 통합 서비스 로그에 메시지를 씁니다.



통합 서비스가 첫 번째 조회 변환에 대해 데이터 캐시 메모리 및 인덱스 캐시 메모리를 할당합니다. 조회 캐시를 공유하는 후속 조회 변환에 대해서는 추가 메모리를 할당하지 않습니다.

변환 또는 캐시 구조가 공유를 허용하지 않는 경우 통합 서비스가 새 캐시를 작성합니다.

## 조회 캐시 공유에 대한 규칙 및 지침

조회 캐시를 공유하는 경우 다음 규칙 및 지침을 고려하십시오.

- 하나 이상의 정적 캐시를 동적 조회와 공유할 수 있습니다. 동적 조회가 동일한 매핑에서 정적 조회가 있는 캐시를 공유하는 경우 정적 조회는 동적 조회에서 생성된 캐시를 재사용합니다.
- 동적 조회 간에는 캐시를 공유할 수 없습니다.
- 지속형 조회 캐시를 재작성하도록 여러 조회 변환을 구성하는 경우 통합 서비스가 첫 번째 조회 변환에 대한 캐시를 작성한 다음 후속 조회 변환에 대한 지속형 조회 캐시를 공유합니다.
- 지속형 조회 캐시를 재작성하도록 첫 번째 조회 변환을 구성하지 않고 캐시를 재작성하도록 후속 조회 변환을 구성하는 경우 변환이 캐시를 공유할 수 없습니다. 통합 서비스가 각 조회 변환을 처리할 때 캐시를 작성합니다.
- 후속 조회 변환에 대한 조회/출력 포트는 통합 서비스가 캐시를 작성하기 위해 사용하는 조회 변환의 포트 하위 집합과 일치하거나 하위 집합이어야 합니다. 포트 순서는 일치하지 않아도 됩니다.
- 통합 서비스가 해시 자동 키 분할을 사용하는 경우 각 변환의 조회/출력 포트는 일치해야 합니다.
- 통합 서비스가 해시 자동 키 분할을 사용하지 않는 경우 첫 번째 공유 변환의 조회/출력 포트는 후속 변환의 조회/출력 포트의 상위 집합과 일치하거나 상위 집합이어야 합니다.
- 캐시를 공유하는 조회 변환은 다음과 같은 특징이 있어야 합니다.
  - 조회 변환은 조회 조건에서 동일한 포트를 사용해야 합니다.
  - SQL 재정의가 사용되는 경우 조회 변환은 동일한 SQL 재정의를 사용해야 합니다.
  - 조회 캐시가 모든 조회 변환에서 활성화되어야 합니다.
  - 조회 변환이 동일한 조회 소스 유형을 사용해야 합니다.
  - 모든 관계형 조회 변환이 동일한 데이터베이스 연결을 사용해야 합니다.
  - 조회 변환이 동일한 조회 테이블 이름을 사용해야 합니다.
  - 모든 조회 변환에 대한 캐시 구조가 호환 가능해야 합니다.

## 캐시 비교

통합 서비스는 구성된 조회 캐시의 유형에 따라 다른 작업을 수행합니다.

다음 표에는 조회 변환의 캐시되지 않은 조회, 정적 캐시 및 동적 캐시가 비교되어 있습니다.

캐시되지 않음	정적 캐시	동적 캐시
통합 서비스가 캐시를 삽입하거나 업데이트하지 않습니다.	통합 서비스가 캐시를 삽입하거나 업데이트하지 않습니다.	통합 서비스가 행을 대상에 전달할 때 캐시에 행을 삽입하거나 업데이트할 수 있습니다.
관계형 조회를 사용할 수 있습니다.	관계형, 플랫 파일 또는 파이프라인 조회를 사용할 수 있습니다. 관계형 또는 플랫 파일 조회를 사용할 수 있습니다.	관계형, 플랫 파일 또는 소스 한정자 조회를 사용할 수 있습니다. 관계형 또는 플랫 파일 조회를 사용할 수 있습니다.
조건이 true인 경우 통합 서비스가 조회 테이블 또는 캐시의 값을 반환합니다. 조건이 true가 아닌 경우 통합 서비스가 연결된 변환에 대해 기본값을 반환하고 연결되지 않은 변환에 대해 NULL을 반환합니다.	조건이 true인 경우 통합 서비스가 조회 테이블 또는 캐시의 값을 반환합니다. 조건이 true가 아닌 경우 통합 서비스가 연결된 변환에 대해 기본값을 반환하고 연결되지 않은 변환에 대해 NULL을 반환합니다.	조건이 true인 경우 통합 서비스가 행 유형에 따라 캐시의 행을 업데이트하거나 캐시를 변경되지 않은 상태로 유지합니다. 조건이 true인 경우 행이 캐시 및 대상 테이블에 있다는 것을 나타냅니다. 업데이트된 행을 대상에 전달할 수 있습니다. 조건이 true가 아닌 경우 통합 서비스가 행 유형에 따라 행을 캐시에 삽입하거나 캐시를 변경되지 않은 상태로 유지합니다. 조건이 true가 아닌 경우 행이 캐시 또는 대상에 없다는 것을 나타냅니다. 삽입된 행을 대상 테이블에 전달할 수 있습니다.

## 조회에 대한 캐시 분할

캐시 분할은 집계, 조이너, 순위 또는 조회 변환을 처리하는 각 파티션에 대해 별개의 캐시를 작성합니다. 캐시 분할은 각 파티션에서 별개의 캐시를 병렬로 쿼리하기 때문에 매핑 성능을 향상시킵니다.

통합 서비스는 매핑에 대한 파티션을 작성할 때 분할된 조회 변환에 캐시 분할을 사용할 수 있습니다.

통합 서비스는 다음 조건이 true인 경우 연결된 조회 변환에 캐시 분할을 사용합니다.

- 조회 조건에 같은 연산자만 포함된 경우
- 연결된 조회 변환이 관계형 테이블에서 데이터를 조회하고 데이터베이스가 대/소문자 구분 비교에 대해 구성된 경우.

통합 서비스는 연결되지 않은 조회 변환에 캐시 분할을 사용하지 않습니다.

통합 서비스가 조회 변환에 캐시 분할을 사용하지 않을 경우 조회 변환의 모든 파티션은 동일한 캐시를 공유합니다. 각 파티션은 동일한 캐시를 순차적으로 쿼리합니다.

## 제 27 장

# 동적 조회 캐시

이 장에 포함된 항목:

- [동적 조회 캐시 개요, 407](#)
- [동적 조회 캐시 사용, 408](#)
- [동적 조회 캐시 속성, 409](#)
- [동적 조회 캐시 및 출력 값, 411](#)
- [조회 변환 값, 411](#)
- [SQL 재정의 및 동적 조회 캐시, 414](#)
- [동적 조회 캐시의 매핑 구성, 414](#)
- [조건부 동적 조회 캐시 업데이트, 417](#)
- [식 결과로 동적 캐시 업데이트, 418](#)
- [동적 조회 캐시 예제, 419](#)
- [동적 조회 캐시에 대한 규칙 및 지침, 420](#)

## 동적 조회 캐시 개요

동적 조회 캐시를 사용하면 캐시와 대상의 동기화 상태를 유지할 수 있습니다. 동적 캐시를 관계형 조회, 플랫폼 파일 조회 또는 파이프라인 조회와 함께 사용할 수 있습니다. 동적 캐시를 관계형 조회 또는 플랫폼 파일 조회와 함께 사용할 수 있습니다.

통합 서비스는 첫 번째 조회 요청을 처리할 때 동적 조회 캐시를 빌드합니다. 변환에 전달되는 각 행의 조회 조건을 바탕으로 캐시를 쿼리합니다. 통합 서비스는 각 행을 처리할 때 조회 캐시를 업데이트합니다.

통합 서비스는 소스에서 행을 읽을 때 조회 쿼리의 결과, 행 유형 및 조회 변환 속성에 따라 다음과 같은 동적 조회 캐시 작업 중 하나를 수행합니다.

### 행을 캐시에 삽입

행이 캐시에 없을 때 행을 캐시에 삽입하도록 조회 변환이 구성된 경우 통합 서비스가 행을 캐시에 삽입합니다. 입력 포트 또는 생성된 시퀀스 ID를 바탕으로 행을 캐시에 삽입하도록 변환을 구성할 수 있습니다. 삽입된 행에는 삽입 플래그가 지정됩니다.

### 캐시의 행을 업데이트

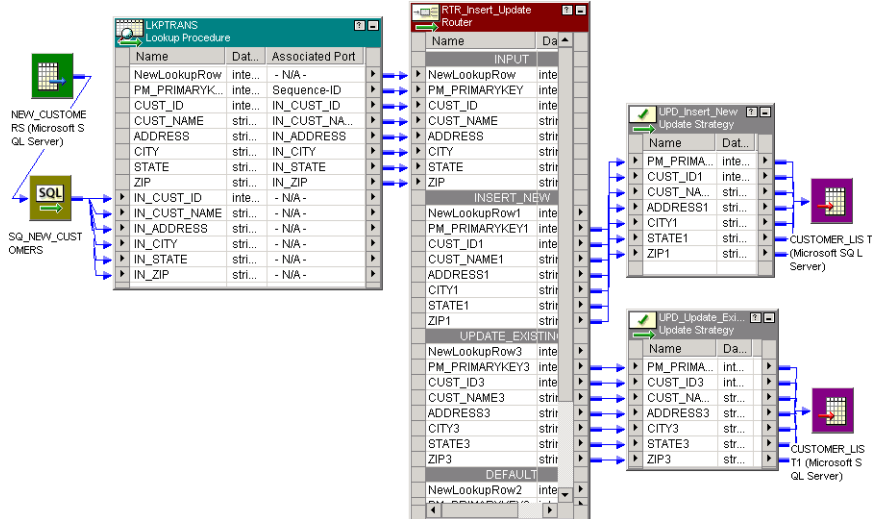
행이 캐시에 있을 때 캐시의 행을 업데이트하도록 조회 변환이 구성된 경우 통합 서비스가 캐시의 행을 업데이트합니다. 통합 서비스는 입력 포트를 바탕으로 캐시의 행을 업데이트합니다. 업데이트된 행에는 업데이트 플래그가 지정됩니다.

## 캐시를 변경하지 않음

행이 캐시에 있을 때 새 행만 삽입하도록 조회 변환이 구성된 경우 통합 서비스가 캐시를 변경하지 않습니다. 행이 캐시에 없을 때 기존 행만 업데이트하도록 지정한 경우 또는 행이 캐시에 있지만 조회 조건에 따라야 하는 경우에도 캐시가 변경되지 않습니다. 변경되지 않은 행에는 변경되지 않음 플래그가 지정됩니다.

NewLookupRow의 값에 따라 동적 조회 변환과 함께 라우터 또는 필터 변환을 구성하여 삽입 또는 업데이트 행을 대상 테이블로 라우팅할 수 있습니다. 변경되지 않은 행을 다른 대상 테이블 또는 플랫폼 파일로 라우팅하거나 삭제할 수 있습니다.

다음 그림은 동적 조회 캐시를 사용하는 조회 변환이 포함된 매핑을 보여 줍니다.



## 동적 조회 캐시 사용

동적 조회 캐시가 포함된 조회 변환이 조회 소스의 변경 내용을 기준으로 캐시를 업데이트하도록 구성할 수 있습니다.

동적 조회 캐시를 사용하는 이유는 다음과 같습니다.

**새로운/업데이트된 고객 정보로 마스터 고객 테이블을 업데이트합니다.**

예를 들어 조회 변환을 사용해 고객 테이블에서 조회를 수행하여 고객이 대상에 있는지를 확인할 수 있습니다. 캐시는 고객 테이블을 나타냅니다. 조회 변환에서는 행을 대상으로 전달할 때 캐시의 행을 삽입 및 업데이트합니다.

**여러 실시간 세션에서 마스터 고객 테이블에 행을 삽입합니다.**

각 세션에서 조회 변환을 사용하여 같은 고객 테이블에 대해 조회를 수행합니다. 각 조회 변환은 고객 테이블의 동적 조회 캐시에 행을 삽입합니다. 여러 세션 간에 동적 캐시를 동기화하는 방법에 대한 자세한 내용은 [GUID-8D12D72A-5FD1-4EF5-B9E8-81A8FEC5DEFA](#)를 참조하십시오.

**느린 변경 차원 테이블과 팩트 테이블로 데이터를 로드합니다.**

파이프라인 두 개를 생성하고 차원 테이블에 대한 조회를 수행하는 조회 변환을 구성합니다. 동적 조회 캐시를 사용하여 차원 테이블로 데이터를 로드합니다. 정적 조회 캐시를 사용하여 팩트 테이블로 데이터를 로드하고 첫 번째 파이프라인에서 동적 캐시의 이름을 지정합니다.

**관계형 테이블이 아닌 내보낸 플랫폼 파일을 조회 소스로 사용합니다.**

데이터베이스 연결 속도가 느린 경우 관계형 테이블 콘텐츠를 플랫폼 파일로 내보낸 다음 해당 파일을 조회 소스로 사용할 수 있습니다. 예를 들어 데이터베이스에 대한 ODBC 연결 속도가 느리면 이 방법을 사용해야 할 수 있습니다. 데이터베이스 테이블을 매핑의 관계형 대상으로 구성하고 조회 캐시 변경 내용을 데이터베이스 테이블로 다시 전달할 수 있습니다.

## 동적 조회 캐시 속성

동적 조회 속성을 구성하여 동적 조회 캐시를 실행하고 캐시의 업데이트 방식을 구성할 수 있습니다. 예를 들어 동적 캐시에 삽입되고 업데이트되는 값을 구성할 수 있습니다.

동적 조회 캐시를 활성화하는 경우 다음 속성을 구성하십시오.

### **일치하는 항목이 여러 개인 경우**

오류 보고로 설정합니다.

### **동적 조회 캐시**

동적 조회 캐시를 활성화합니다.

이 옵션은 조회 캐싱을 활성화한 후에 사용할 수 있습니다.

### **업데이트 기타 항목 삽입**

이 옵션은 조회 변환을 시작하는 업데이트 유형의 행에 적용됩니다. 활성화한 경우 통합 서비스가 캐시의 기존 행을 업데이트하고 새 행을 삽입합니다. 비활성화한 경우 통합 서비스가 새 행을 삽입하지 않습니다.

이 옵션은 동적 캐싱을 활성화한 후에 사용할 수 있습니다.

### **삽입 기타 항목 업데이트**

이 옵션은 조회 변환을 시작하는 삽입 유형의 행에 적용됩니다. 활성화한 경우 통합 서비스가 캐시에 행을 삽입하고 기존 행을 업데이트합니다. 비활성화한 경우 통합 서비스가 기존 행을 업데이트하지 않습니다.

이 옵션은 동적 캐싱을 활성화한 후에 사용할 수 있습니다.

### **업데이트 시 이전 값 출력**

조회 변환이 캐시의 기존 값 또는 새 값을 출력할 수 있습니다. 활성화한 경우 통합 서비스가 캐시의 값을 업데이트하기 전에 조회/출력 포트의 기존 값을 출력합니다. 통합 서비스는 캐시의 행을 업데이트할 때 조회 캐시의 값을 출력한 다음 입력 데이터를 바탕으로 행을 업데이트합니다. 통합 서비스가 캐시에 행을 삽입할 때는 null 값이 출력됩니다.

조회/출력 및 입력/출력 포트에서 동일한 값을 전달하려면 통합 서비스에 대해 이 속성을 비활성화하십시오. 이 속성은 기본적으로 활성화됩니다.

이 옵션은 동적 캐싱을 활성화한 후에 사용할 수 있습니다.

### **동적 캐시 조건 업데이트**

활성화된 경우 통합 서비스는 조건 식을 사용하여 동적 캐시의 업데이트 여부를 결정합니다. 조건이 **true**이고 캐시에 데이터가 존재하는 경우 통합 서비스가 캐시를 업데이트합니다.

조건 식은 조회 포트 또는 입력 포트를 사용하여 작성합니다. 조건 식에는 입력 값 또는 조회 캐시의 값이 포함될 수 있습니다. 기본값은 **true**입니다.

이 옵션은 동적 캐싱을 활성화한 후에 사용할 수 있습니다.

### **NewLookupRow**

디자이너는 이 포트를 동적 캐시가 구성된 조회 변환에 추가합니다.

NewLookupRow 속성에는 다음 값 중 하나가 포함될 수 있습니다.

- 0 = 캐시를 업데이트하지 않습니다.
- 1 = 행을 캐시에 삽입합니다.
- 2 = 캐시의 행을 업데이트합니다.

조회 캐시와 대상 테이블의 동기화를 유지하려면 NewLookupRow 값이 1 또는 2일 때 행을 대상에 전달하십시오.

Developer tool은 이 포트를 동적 캐시가 구성된 조회 변환에 추가합니다.

NewLookupRow 속성에는 다음 값 중 하나가 포함될 수 있습니다.

- 0 = 캐시를 업데이트하지 않습니다.
- 1 = 행을 캐시에 삽입합니다.
- 2 = 캐시의 행을 업데이트합니다.

조회 캐시와 대상 테이블의 동기화를 유지하려면 NewLookupRow 값이 1 또는 2일 때 행을 대상에 전달하십시오.

#### 연결된 포트

통합 서비스는 연결된 포트의 값을 사용하여 캐시의 데이터를 업데이트합니다. 통합 서비스는 입력 포트와 조회 조건에 지정된 조회 소스 포트를 연결합니다. 동적 조회의 나머지 조회 소스 포트에 대해서도 연결된 포트를 구성해야 합니다. 동적 조회의 모든 조회 소스 포트에 대해 연결된 포트를 구성하지 않을 경우 매핑 유효성 검사가 실패합니다.

조회 소스 포트를 다음 개체와 연결할 수 있습니다.

개체	설명
입력 포트	입력 포트의 값을 바탕으로 캐시를 업데이트합니다.
연결된 식	식을 입력하려면 선택합니다. 통합 서비스가 식의 결과를 바탕으로 캐시를 업데이트합니다.
시퀀스 ID	조회 캐시에 삽입된 행에 대한 기본 키를 생성합니다. 시퀀스 ID는 bigint 및 int 열에만 연결할 수 있습니다.

#### 업데이트에 대한 Null 입력 무시

동적 캐시를 사용하도록 조회 변환을 구성할 경우 디자이너가 조회/출력 포트에 대해 이 포트 속성을 활성화합니다. 통합 서비스에서 null 입력 값으로 캐시의 열을 업데이트하지 않도록 하려면 이 속성을 선택합니다.

동적 캐시를 사용하도록 조회 변환을 구성할 경우 Developer tool이 조회/출력 포트에 대해 이 포트 속성을 활성화합니다. 통합 서비스에서 null 입력 값으로 캐시의 열을 업데이트하지 않도록 하려면 이 속성을 선택합니다.

#### 비교 시 무시

동적 캐시를 사용하도록 조회 변환을 구성할 경우 디자이너가 조회 조건에 사용되지 않는 조회/출력 포트에 대해 이 포트 속성을 활성화합니다. 통합 서비스는 기본적으로 모든 조회 포트의 값을 연결된 포트의 값과 비교합니다. 통합 서비스에서 행을 업데이트하기 전에 값을 비교할 때 포트를 무시하도록 하려면 이 속성을 선택합니다. 이 속성을 사용하면 비교 성능이 향상됩니다.

동적 캐시를 사용하도록 조회 변환을 구성할 경우 Developer tool이 조회 조건에 사용되지 않는 조회/출력 포트에 대해 이 포트 속성을 활성화합니다. 통합 서비스는 기본적으로 모든 조회 포트의 값을 연결된 포트의

값과 비교합니다. 통합 서비스에서 행을 업데이트하기 전에 값을 비교할 때 포트를 무시하도록 하려면 이 속성을 선택합니다. 이 속성을 사용하면 비교 성능이 향상됩니다.

## 동적 조회 캐시 및 출력 값

동적 조회 캐시를 활성화한 경우 출력 포트 값은 동적 조회 캐시의 구성 방식에 따라 다릅니다. 조회/출력 포트의 출력 값은 통합 서비스가 행을 업데이트할 때 사용자가 출력하도록 선택한 값(이전 값 또는 새 값)에 따라 다릅니다.

**업데이트 시 이전 값 출력** 속성을 구성하여 조회/출력 포트에 다음과 같은 유형의 출력 값 중 하나를 지정할 수 있습니다.

- 업데이트 시 이전 값 출력. 통합 서비스가 행을 업데이트하기 전에 캐시에 존재하는 값을 출력합니다.
- 업데이트 시 새 값 출력. 통합 서비스가 캐시에 기록하는 업데이트된 값을 출력합니다. 조회/출력 포트 값은 출력 포트 값과 일치합니다.
- 업데이트 시 새 값 출력. 통합 서비스가 캐시에 기록하는 업데이트된 값을 출력합니다. 조회/출력 포트 값은 출력 포트 값과 일치합니다.

## 조회 변환 값

조회 변환에는 입력 포트, 조회 및 출력 포트에 대한 값이 포함됩니다. 동적 조회 캐시를 활성화한 경우 출력 포트 값은 동적 조회 캐시의 구성 방식에 따라 다릅니다.

조회 변환에는 다음 유형의 값이 포함됩니다.

### 입력 값

통합 서비스가 조회 변환에 전달하는 값입니다.

### 조회 값

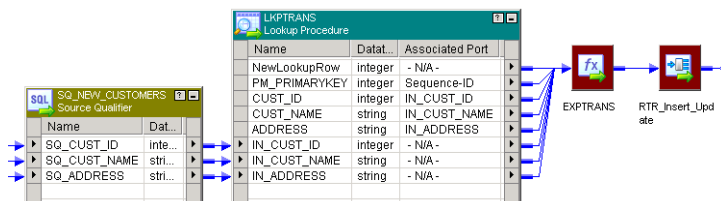
통합 서비스가 캐시에 삽입하는 값입니다.

### 출력 값

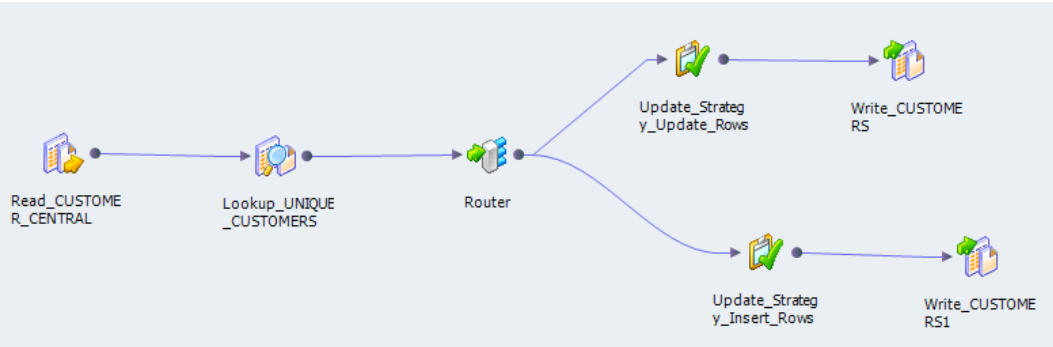
통합 서비스가 조회 변환의 출력 포트에서 전달하는 값입니다. 조회/출력 포트의 출력 값은 통합 서비스가 행을 업데이트할 때 사용자가 출력하도록 선택한 값(이전 값 또는 새 값)에 따라 다릅니다.

## 조회 변환 값 예제

예를 들어, 다음 개체를 가진 매핑을 작성합니다.



예를 들어, 다음 개체를 가진 매핑을 작성합니다.



조희 변환의 경우 동적 조희 캐싱을 활성화하고 다음 조희 조건을 정의하십시오.

`IN_CUST_ID = CUST_ID`

기본적으로 조희 변환에 입력되는 모든 행의 행 유형이 삽입됩니다. 캐시 및 대상 테이블에서 삽입과 업데이트를 모두 수행하려면 조희 변환에서 **삽입 기타 항목 업데이트** 속성을 선택합니다.

### 초기 캐시 값

세션을 실행할 경우 통합 서비스가 대상 테이블에서 조희 캐시를 작성합니다.

매핑을 실행할 경우 통합 서비스가 대상 테이블에서 조희 캐시를 작성합니다.

다음 표에는 조희 캐시의 초기 값이 표시되어 있습니다.

PK_PRIMARYKEY	CUST_ID	CUST_NAME	ADDRESS
100001	80001	Marion James	100 Main St.
100002	80002	Laura Jones	510 Broadway Ave.
100003	80003	Shelley Lau	220 Burnside Ave.

### 입력 값

대상 테이블에 있는 행과 대상 테이블에 없는 행이 소스에 포함되어 있습니다. 통합 서비스가 소스 행을 조희 변환에 전달합니다.

다음 표에는 소스 행이 표시되어 있습니다.

SQ_CUST_ID	SQ_CUST_NAME	SQ_ADDRESS
80001	Marion Atkins	100 Main St.
80002	Laura Gomez	510 Broadway Ave.
99001	Jon Freeman	555 6th Ave.

### 조희 값

통합 서비스가 조희 조건을 기준으로 캐시의 값을 조희합니다. 기존 고객 ID 80001 및 80002에 대해 캐시의 행을 업데이트합니다. 고객 ID 99001에 대해 캐시에 행을 삽입합니다. 통합 서비스가 새 행에 대해 새 키 (PK\_PRIMARYKEY)를 생성합니다.



다음 표에는 조회에서 반환된 행과 값이 표시되어 있습니다.

PK_PRIMARYKEY	CUST_ID	CUST_NAME	ADDRESS
100001	80001	Marion Atkins	100 Main St.
100002	80002	Laura Gomez	510 Broadway Ave.
100004	99001	Jon Freeman	555 6th Ave.

## 출력 값

통합 서비스가 동적 캐시에서 수행한 삽입 및 업데이트를 기반으로 조회 변환의 행에 플래그 지정합니다. 결국 통합 서비스가 삽입 행에 대한 분기와 업데이트 행에 대한 분기를 달리 작성하는 라우터 변환에 행을 전달합니다. 각 분기에는 업데이트 전략 변환이 포함되어 있습니다. 업데이트 전략 변환은 **NewLookupRow** 포트 값을 기반으로 삽입 또는 업데이트에 대한 행을 플래그 지정합니다.

통합 서비스가 행을 업데이트할 때 이전 값 또는 새 값을 출력하도록 선택하는지 여부에 따라 조회/출력 및 입력/출력 포트의 출력 값이 달라집니다. 그러나 새 행과 업데이트된 행의 경우 시퀀스 ID를 사용하는 **NewLookupRow** 포트 및 조회/출력 포트의 출력 값이 동일합니다.

새 값을 출력하도록 선택한 경우 조회/출력 포트에서 다음 값을 출력합니다.

NewLookupRow	PK_PRIMARYKEY	CUST_ID	CUST_NAME	ADDRESS
2	100001	80001	Marion Atkins	100 Main St.
2	100002	80002	Laura Gomez	510 Broadway Ave.
1	100004	99001	Jon Freeman	555 6th Ave.

이전 값을 출력하도록 선택한 경우 조회/출력 포트에서 다음 값을 출력합니다.

NewLookupRow	PK_PRIMARYKEY	CUST_ID	CUST_NAME	ADDRESS
2	100001	80001	Marion James	100 Main St.
2	100002	80002	Laura Jones	510 Broadway Ave.
1	100004	99001	Jon Freeman	555 6th Ave.

통합 서비스가 조회 캐시의 행을 업데이트하는 경우 캐시 및 대상 테이블의 행에 대해 기본 키 (**PK\_PRIMARYKEY**) 값을 사용합니다.

통합 서비스는 캐시에서 찾지 않는 대상 고객에 대해 시퀀스 ID를 사용하여 기본 키를 생성합니다. 통합 서비스는 조회 캐시에 기본 키 값을 삽입하고 값을 조회/출력 포트에 반환합니다.

통합 서비스는 입력 값과 일치하는 입력/출력 포트의 값을 출력합니다.

**참고:** 입력 값이 NULL이고 연결된 입력 포트에 대해 Null 무시 속성을 선택한 경우 입력 값이 조회 값 또는 입력/출력 포트의 값과 같지 않습니다. Null 무시 속성을 선택한 경우 Null 값을 대상에 전달하면 조회 캐시 및 대상 테이블이 동기화되지 않을 수 있습니다. Null 값을 대상에 전달하지 않는지 확인해야 합니다.

## SQL 재정의 및 동적 조회 캐시

조회 쿼리에서 **WHERE** 절을 추가하여 캐시를 작성하는 데 사용되는 레코드를 필터링하고 캐시되지 않은 조회에 대한 데이터베이스 테이블에서 조회를 수행할 수 있습니다. 그러나 행을 동적 캐시에 삽입하면 통합 서비스가 **WHERE** 절을 사용하지 않습니다.

동적 캐시를 사용하는 조회 변환에 **WHERE** 절을 추가할 때는 조회 변환 전에 필터 변환을 연결하여 캐시 또는 대상 테이블에 삽입하지 않을 행을 필터링합니다. 필터 변환을 포함하지 않으면 캐시 및 대상 테이블 간에 결과가 일관되지 않을 수 있습니다.

직원 테이블 EMP에 대해 동적 조회를 수행하여 EMP\_ID를 기준으로 일치 행을 표시하도록 조회 변환을 구성하는 경우를 예로 들어 보겠습니다. 이 경우 다음과 같은 조회 SQL 재정의의 정의를 정의합니다.

```
SELECT EMP_ID, EMP_STATUS FROM EMP ORDER BY EMP_ID, EMP_STATUS WHERE EMP_STATUS = 4
```

세션매핑을 처음 실행할 때 통합 서비스는 조회 SQL 재정의의 기준으로 대상 테이블에서 조회 캐시를 빌드합니다. 캐시의 모든 행은 **WHERE** 절의 조건(EMP\_STATUS = 4)과 일치합니다.

예를 들어 통합 서비스는 지정한 조회 조건과 일치하는 소스 행을 읽지만 EMP\_STATUS 값은 2입니다. 대상에는 EMP\_STATUS가 2인 행이 있을 수도 있지만 통합 서비스는 SQL 재정의로 인해 캐시에서 해당 행을 찾지 않습니다. 통합 서비스는 캐시에 행을 삽입한 다음 대상 테이블에 행을 전달합니다. 통합 서비스가 대상 테이블에 이 행을 삽입할 때 해당 행이 이미 있으면 결과가 일관되지 않을 수 있습니다. 또한 캐시의 모든 행이 SQL 재정의에서 **WHERE** 절의 조건과 일치하지는 않습니다.

**WHERE** 절과 일치하는 행만 캐시에 삽입되는지 확인하려면 조회 변환 전에 필터 변환을 추가하고 필터 조건을 조회 SQL 재정의에서 **WHERE** 절의 조건으로 정의합니다.

위의 예제에서는 SQL 재정의의 **WHERE** 절 및 필터 변환에 다음 필터 조건을 입력합니다.

```
EMP_STATUS = 4
```

## 동적 조회 캐시의 매핑 구성

동적 캐시를 포함하는 조회를 사용하는 경우 동적 조회 캐시를 업데이트하고 변경된 행을 대상에 기록하도록 매핑을 구성해야 합니다.

다음 단계를 완료하여 동적 조회 캐시의 매핑을 구성하십시오.

**조회 변환의 입력 행에 삽입 또는 업데이트 플래그를 지정합니다.**

기본적으로 모든 입력 행의 행 유형은 삽입입니다. 조회 변환 앞에 업데이트 전략 변환을 추가하여 입력 행에 여러 행 유형을 지정합니다.

**통합 서비스가 동적 캐시의 입력 행을 처리하는 방식을 지정합니다.**

삽입 기타 항목 업데이트 또는 업데이트 기타 항목 삽입 옵션을 선택하여 삽입 또는 업데이트 플래그가 지정된 행을 처리합니다.

**대상에 삽입하고 업데이트할 행에 대한 별도의 매핑 파이프라인을 작성합니다.**

조회 변환 뒤에 필터 또는 라우터 변환을 추가하여 삽입 및 업데이트 행을 개별 매핑 분기로 라우팅합니다. NewLookupRow의 값을 사용하여 각 행에 적절한 분기를 결정합니다.

**조회 변환의 출력 행에 대한 행 유형을 구성합니다.**

행에 삽입 또는 업데이트 플래그를 지정하는 업데이트 전략 변환을 추가합니다.

## 삽입 기타 항목 업데이트

동적 조회 캐시에서 행 유형이 삽입인 기존 행을 업데이트하려면 **삽입 기타 항목 업데이트** 속성을 사용합니다.

이 속성은 조회 변환을 시작하는 삽입 유형의 행에만 적용됩니다. 업데이트와 같은 다른 모든 유형의 행이 조회 변환을 시작하는 경우 **삽입 기타 항목 업데이트** 속성은 통합 서비스의 행 처리 방식에 영향을 미치지 않습니다.

**삽입 기타 항목 업데이트**를 선택하고 조회 변환을 시작하는 행 유형이 삽입인 경우 통합 서비스는 해당 행이 새 행일 경우 캐시에 삽입합니다. 인덱스 캐시에 존재하지만 데이터 캐시에는 존재하지 않는 행이 현재 행과 다른 경우 통합 서비스가 데이터 캐시에 해당 행을 업데이트합니다.

**삽입 기타 항목 업데이트**를 선택하지 않고 조회 변환을 시작하는 행 유형이 삽입인 경우 통합 서비스는 해당 행이 새 행일 경우 캐시에 삽입하고 이미 존재하는 행일 경우 캐시를 변경하지 않습니다.

다음 표에는 조회 변환을 시작하는 행의 유형이 삽입일 때 통합 서비스가 조회 캐시를 변경하는 방식이 설명되어 있습니다.

삽입 기타 항목 업데이트 옵션	행이 캐시에 있음	데이터 캐시가 다름	조회 캐시 결과	NewLookupRow 값
선택 취소 - 삽입만	예	-	변경 없음	0
선택 취소 - 삽입만	아니오	-	삽입	1
선택됨	예	예	업데이트	2 <sup>1</sup>
선택됨	예	아니오	변경 없음	0
선택됨	아니오	-	삽입	1

<sup>1</sup> 조회 조건에 없는 모든 조회 포트에 대해 Null 무시를 선택한 경우 이러한 모든 포트에 null 값이 포함되면 통합 서비스가 캐시를 변경하지 않으며 NewLookupRow 값은 0과 동일합니다.

## 업데이트 기타 항목 삽입

**업데이트 기타 항목 삽입** 속성을 사용하여 행 유형이 업데이트인 경우 동적 조회 캐시에 새 행을 삽입할 수 있습니다.

조회 변환에서 **업데이트 기타 항목 삽입** 속성을 선택할 수 있습니다. 이 속성은 행 유형이 업데이트인 조회 변환에 입력되는 행에만 적용됩니다. 삽입과 같은 기타 유형의 행이 조회 변환에 입력되면 이 속성은 통합 서비스가 행을 처리하는 방법에 영향을 주지 않습니다.

조회 변환에 입력되는 행 유형이 업데이트일 때 이 속성을 선택하면 통합 서비스는 행이 인덱스 캐시에 있으며 캐시 데이터가 기존 행과 다른 경우 캐시의 행을 업데이트합니다. 새로운 행일 경우 통합 서비스가 캐시에 행을 삽입합니다.

조회 변환에 입력되는 행 유형이 업데이트일 때 이 속성을 선택하지 않으면 통합 서비스는 행이 있는 경우 캐시에서 행을 업데이트하며 새 행의 경우에는 캐시를 변경하지 않습니다.

조회 조건에 없는 모든 조회 포트에 null 값이 포함된 경우 해당 포트에 대해 **Null 무시**를 선택하면 통합 서비스는 캐시를 변경하지 않으며 NewLookupRow 값은 0이 됩니다.

다음 테이블에서는 조회 변환에 입력되는 행의 유형이 업데이트일 때 통합 서비스가 조회 캐시를 변경하는 방법을 설명합니다.

업데이트 기타 항목 삽입 옵션	행이 캐시에 있음	데이터 캐시가 다름	조회 캐시 결과	NewLookupRow 값
지워짐(업데이트만 해당)	예	예	업데이트	2
지워짐(업데이트만 해당)	예	아니오	변경 없음	0
지워짐(업데이트만 해당)	아니오	-	변경 없음	0
선택됨	예	예	업데이트	2
선택됨	예	아니오	변경 없음	0
선택됨	아니오	-	삽입	1

## 동적 조회 캐시 및 대상 동기화

동적 조회 캐시와 대상을 동기화하려면 다운스트림 변환을 구성합니다.

동적 조회 캐시를 사용하는 경우 통합 서비스는 조회 캐시에 먼저 기록한 다음 대상 테이블에 기록합니다. 통합 서비스가 데이터를 대상에 기록하지 않을 경우 조회 캐시와 대상 테이블이 동기화되지 않을 수 있습니다. 예를 들어 대상 데이터베이스가 데이터를 거부할 수 있습니다.

조회 캐시를 조회 테이블과 동기화 상태로 유지하려면 다음 지침을 고려하십시오.

- NewLookupRow 값이 1 또는 2일 경우 행을 캐시된 대상에 전달하는 라우터 변환을 사용합니다.
- NewLookupRow 값이 0일 경우 행을 삭제하는 라우터 변환을 사용합니다. 또는 행을 다른 대상에 출력합니다.
- 행을 대상에 삽입 또는 업데이트하도록 플래그 지정하려면 조회 변환 후 업데이트 전략 변환을 사용합니다.
- 세션을 실행할 때 오류 임계값을 1로 설정합니다. 오류 임계값을 1로 설정한 경우 첫 번째 오류가 발생하면 세션이 실패합니다. 통합 서비스가 새 캐시 파일을 디스크에 기록하지 않습니다. 대신, 원본 캐시 파일이 있는 경우 이 파일을 복원합니다. 또한 세션 이전 대상 테이블을 대상 데이터베이스에 복원해야 합니다.
- 통합 서비스가 조회 캐시에 기록하는 동일한 값을 조회 변환이 대상에 출력하는지 확인하십시오. 업데이트 시 새 값을 출력하도록 선택한 경우 입력/출력출력 포트 대신 조회/출력 포트만 대상 테이블에 연결하십시오. 업데이트 시 이전 값을 출력하도록 선택한 경우 조회 변환 후 그리고 라우터 변환 전에 식 변환을 추가하십시오. 대상 테이블의 각 포트마다 식 변환에서 출력 포트를 추가하고 대상에 Null 입력 값을 출력하지 않도록 확인하는 식을 작성하십시오.
- 세션 속성에서 소스 행을 다음으로 처리 속성을 데이터 구동으로 설정하십시오.
- 업데이트 전략 대상 테이블 옵션을 정의할 경우 삽입 및 업데이트를 업데이트로 선택합니다. 이 경우 통합 서비스가 업데이트 표시된 행을 업데이트하고 삽입 표시된 행을 삽입하게 됩니다. 세션 속성에서 매핑 탭의 변환 보기에서 이 옵션을 선택합니다.

## 조건부 동적 조회 캐시 업데이트

부울 식의 결과를 기반으로 동적 조회 캐시를 업데이트할 수 있습니다. 통합 서비스는 식이 **True**일 경우 캐시를 업데이트합니다.

대상 테이블에 제품 번호, 재고 수량 및 타임스탬프 열이 있는 경우를 예로 들겠습니다. 재고 수량을 최신 소스 값으로 업데이트해야 합니다. 소스 데이터의 타임스탬프가 동적 캐시의 타임스탬프보다 큰 경우 재고 수량을 업데이트할 수 있습니다. 조회 변환에서 다음과 유사한 식을 작성하십시오.

```
lookup_timestamp < input_timestamp
```

식에는 조회 및 입력 포트가 포함될 수 있습니다. 기본 제공 변수, 매핑 변수 및 매개 변수의 변수에 액세스할 수 있습니다. 사용자 정의 함수를 포함하고 연결되지 않은 변환을 참조할 수 있습니다.

식에서 **true**, **false** 또는 **NULL**을 반환합니다. 식 결과가 **NULL**일 경우 식이 **false**입니다. 통합 서비스에서 캐시를 업데이트하지 않습니다. 식 결과를 **true**로 변경해야 할 경우 식의 **NULL** 값 확인을 추가할 수 있습니다. 기본 식 값은 **true**입니다.

변환 개발자를 사용하여 식을 작성하십시오. 세션 수준에서 **동적 캐시 업데이트 조건**을 재정의할 수 없습니다.

## 조건부 동적 조회 캐시 처리

통합 서비스의 동적 조회 캐시 업데이트 여부를 결정하는 조건을 작성할 수 있습니다. 조건이 **false** 또는 **NULL**일 경우 통합 서비스가 동적 조회 캐시를 업데이트하지 않습니다.

이 조건이 **false** 또는 **NULL**인 경우 조회 변환의 속성에 관계없이 **NewLookupRow** 값은 0이고 통합 서비스가 행 삽입 또는 업데이트로 동적 조회 캐시를 업데이트하지 않습니다.

행이 캐시에 있고 다른 행 삽입 또는 업데이트 기타 항목 삽입이 활성화된 경우 **NewLookupRow** 값은 1이고 통합 서비스가 새 행을 캐시에 업데이트합니다.

행이 캐시에 없고 다른 행 삽입 또는 업데이트 기타 항목 삽입이 활성화된 경우 **NewLookupRow** 값은 2이고 통합 서비스가 새 행을 캐시에 삽입합니다.

동적 캐시를 동기화하도록 조회 변환을 구성한 경우 통합 서비스가 행을 캐시에 삽입하면 조회 소스에도 행이 삽입됩니다.

## 조건부 동적 조회 캐시 구성

통합 서비스의 동적 조회 캐시 업데이트 여부를 결정하는 식을 구성할 수 있습니다.

1. 조회 변환을 작성합니다.
2. **속성** 보기의 **런타임** 탭에서 **조회 캐싱 설정**을 선택합니다.
3. **속성** 보기의 **고급** 탭에서 **동적 조회 캐시**를 선택합니다.
4. 조건을 입력하려면 **동적 캐시 조건 업데이트** 속성의 아래쪽 화살표를 클릭합니다.  
식 편집기가 표시됩니다.
5. 식 조건을 정의합니다.  
식의 입력 포트, 조회 포트 및 함수를 선택할 수 있습니다.
6. **유효성 검사**를 클릭하여 식이 유효한지 확인합니다.
7. **확인**을 클릭합니다.
8. 해당하는 경우 동적 조회 캐시에 적용되는 다른 고급 속성을 구성합니다.

## 식 결과로 동적 캐시 업데이트

조회 변환에서 동적 조회 캐시 값을 식 결과로 업데이트할 수 있습니다.

예를 들어, 제품 테이블 대상에 주문 개수를 포함하는 숫자 열이 있습니다. 조회 변환에서 제품에 대한 주문을 받을 때마다 다음 식 결과로 동적 캐시 `cache order_count`를 업데이트합니다.

```
order_count = order_count + 1
```

조회 변환에서 `order_count`를 반환합니다.

통합 서비스에서 식이 Null로 평가되는 경우를 처리하는 방법을 구성할 수 있습니다.

### Null 식 값

식 값 중 하나가 NULL인 경우 식이 NULL을 반환합니다. 하지만 NULL이 아닌 값을 반환하도록 식을 구성할 수 있습니다.

식이 조회 포트를 참조하지만 소스 데이터가 신규인 경우 조회 포트에 기본값이 포함됩니다. 기본값은 NULL일 수 있습니다. IsNull 식을 구성하면 NULL 값을 확인할 수 있습니다.

예를 들어 다음 식은 `lookup_column`이 NULL인지 확인합니다.

```
if (isnull(lookup_column), input_port, user_expression)
```

열이 NULL인 경우 `input_port` 값을 반환합니다. 그렇지 않은 경우 식의 값을 반환합니다.

### 식 처리

통합 서비스는 식 결과에 따라 동적 조회 캐시에 행을 삽입 및 업데이트할 수 있습니다. 식의 결과는 조회 포트 값이 NULL인지 여부와 식에 포함되었는지 여부에 따라 달라질 수 있습니다.

삽입 기타 항목 업데이트를 활성화한 경우 데이터가 캐시에 없으면 통합 서비스가 식 결과에 따라 행을 삽입합니다. 데이터가 캐시에 없는 경우 조회 포트 값은 NULL입니다. 식이 조회 포트 값을 참조하는 경우 통합 서비스가 식의 기본 포트 값을 대체합니다. 삽입 기타 항목 업데이트를 활성화한 경우 데이터가 캐시에 있으면 통합 서비스가 식 결과에 따라 행을 업데이트합니다.

업데이트 기타 항목 삽입을 활성화한 경우 데이터가 캐시에 있으면 통합 서비스가 식 결과에 따라 캐시를 업데이트합니다. 데이터가 캐시에 없는 경우에는 통합 서비스가 식 결과를 포함하는 행을 삽입합니다. 식이 조회 포트 값을 참조하는 경우 통합 서비스가 식의 기본 포트 값을 대체합니다.

동적 캐시를 동기화하도록 조회 변환을 구성한 경우 통합 서비스는 식 결과에 따라 행을 조회 캐시에 삽입합니다. 통합 서비스는 식 결과에 따라 행을 조회 소스에 삽입합니다.

### 동적 캐시 업데이트를 위한 식 구성

동적 캐시 조회 업데이트를 위한 식을 구성할 수 있습니다.

조건부 식을 작성하려면 먼저 동적 조회를 수행하는 조회 변환을 활성화해야 합니다.

1. 조회 변환을 작성합니다.
2. 속성 보기의 런타임 탭에서 **조회 캐싱 설정**을 선택합니다.
3. 속성 보기의 고급 탭에서 **동적 조회 캐시**를 선택합니다.
4. 해당하는 경우 동적 조회 캐시에 적용되는 다른 고급 속성을 구성합니다.
5. 식을 작성하려면 속성 보기에서 **열** 탭을 선택합니다.
6. 업데이트할 조회 포트의 **연결된 포트** 열에서 드롭다운 화살표를 클릭합니다.

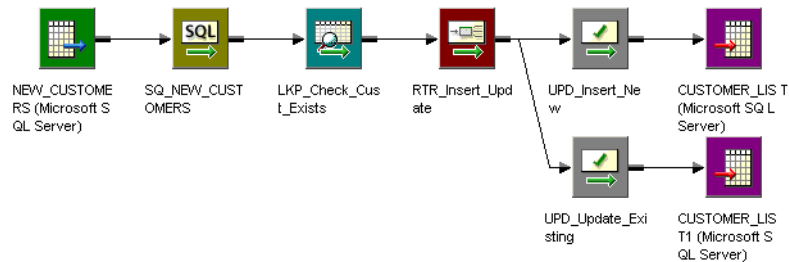
7. 드롭다운 목록에서 **연결된 식**을 선택하고 **Enter**를 클릭합니다.  
식 편집기가 표시됩니다.
8. 식을 정의합니다.  
식의 입력 포트, 조회 포트 및 함수를 선택할 수 있습니다. 식 반환 값이 조회 포트의 데이터 유형과 일치해야 합니다.
9. **유효성 검사**를 클릭하여 식이 유효한지 확인합니다.
10. **확인**을 클릭합니다.

## 동적 조회 캐시 예제

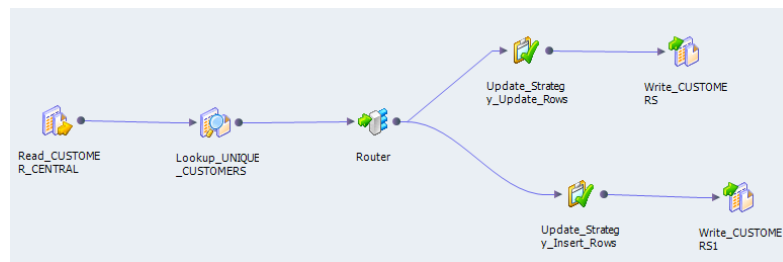
동적 조회 캐시를 사용하여 대상에 행을 삽입 및 업데이트할 수 있습니다. 동적 조회 캐시를 사용하는 경우 대상에 삽입하고 업데이트하는 동일한 행을 캐시에 삽입 및 업데이트할 수 있습니다.

예를 들어 고객 데이터가 포함된 테이블을 업데이트해야 합니다. 소스 데이터에는 대상에 삽입하거나 업데이트할 고객 데이터의 행이 포함됩니다. 대상을 나타내는 동적 캐시를 작성합니다. 이 동적 캐시의 고객을 조회하는 조회 변환을 구성합니다.

다음 그림은 동적 조회 캐시를 사용하는 조회 변환이 포함된 매핑을 보여 줍니다.



다음 그림은 동적 조회 캐시를 사용하는 조회 변환이 포함된 매핑을 보여 줍니다.



라우터 변환은 두 분기로 분할됩니다. 라우터 변환은 삽입 행을 한 분기로 전달하고 업데이트 행을 다른 분기로 전달합니다. 각 분기에는 행을 대상에 기록하는 업데이트 전략 변환이 포함됩니다. 두 분기에는 동일한 대상이 포함됩니다.

세션을 시작하면 통합 서비스가 **Customer\_List** 테이블에서 조회 캐시를 작성합니다. **Customer\_List** 테이블은 매핑의 대상이기도 합니다. 통합 서비스가 조회 캐시에 없는 행을 읽으면 해당 행을 캐시에 삽입합니다.

매핑을 시작하면 통합 서비스가 고객 대상 테이블에서 조회 캐시를 작성합니다. 통합 서비스가 조회 캐시에 없는 행을 읽으면 해당 행을 캐시에 삽입합니다.



조회 변환은 각 행을 라우터 변환에 반환합니다. 라우터 변환은 행의 표시(삽입 또는 업데이트)에 따라 업데이트 전략 변환 중 하나로 행을 전달합니다. 라우터 변환은 **NewLookupRow** 속성을 바탕으로 행의 표시(삽입 또는 업데이트)를 결정합니다. 업데이트 전략 변환은 각 행을 삽입 또는 업데이트로 표시한 다음 대상에 전달합니다.

세션을 실행하면 **Customer\_List** 테이블이 변경됩니다. 통합 서비스가 조회 캐시에 새 행을 삽입하고 기존 행을 업데이트합니다. 통합 서비스는 조회 캐시와 **Customer\_List** 테이블의 동기화 상태를 유지합니다.

매핑을 실행하면 고객 대상 테이블이 변경됩니다. 통합 서비스가 조회 캐시에 새 행을 삽입하고 기존 행을 업데이트합니다. 통합 서비스는 조회 캐시와 고객 대상 테이블의 동기화 상태를 유지합니다.

대상에 대한 키를 생성하려면 연결된 포트의 **Sequence-ID**를 사용합니다. 통합 서비스는 이 시퀀스 ID를 대상 테이블에 삽입된 새 행의 기본 키로 사용합니다.

동적 조회 캐시를 사용하면 데이터베이스에서 캐시를 한 번만 작성하므로 세션 성능이 개선됩니다.

## 동적 조회 캐시에 대한 규칙 및 지침

동적 조회 캐시를 사용하는 경우 다음 지침을 고려하십시오.

- 동적 조회 캐시를 사용하는 경우 **일치하는 항목이 여러 개인 경우** 속성을 보고서 오류로 설정해야 합니다. 속성을 재설정하려면 동적 조회를 정적 조회로 변경하고 속성을 변경한 다음 정적 조회를 동적 조회로 변경합니다.
- 동일한 대상 로드 순서 그룹에서 동적 조회 변환과 정적 조회 변환 간에 캐시를 공유할 수 없습니다.
- 관계형, 플랫 파일 또는 소스 한정자 변환 조회의 경우 동적 조회 캐시를 활성화할 수 있습니다.
- 관계형, 플랫 파일 조회의 경우 동적 조회 캐시를 활성화할 수 있습니다.
- 조회 변환은 연결된 변환이어야 합니다.
- 지속형 또는 비지속형 캐시를 사용할 수 있습니다.
- 동적 캐시가 비지속형인 경우 **조회 소스에서 다시 캐시**를 활성화하지 않는 경우에도 통합 서비스가 항상 데이터베이스에서 캐시를 재작성합니다.
- 동적 캐시가 비지속형인 경우 **조회 소스에서 다시 캐시**를 활성화하지 않는 경우에도 통합 서비스가 항상 데이터베이스에서 캐시를 재작성합니다.
- 동적 캐시 파일을 조회 소스 테이블과 동기화할 때 조회 변환이 행을 조회 소스 테이블 및 동적 조회 캐시에 삽입합니다. 소스 행이 업데이트 행인 경우 조회 변환이 동적 조회 캐시만 업데이트합니다.
- 같은 조회 조건만 작성할 수 있습니다. 동적 캐시에서는 데이터 범위를 조회할 수 없습니다.
- 조회 조건에 없는 각 조회 포트는 입력 포트, 시퀀스 ID 또는 연결된 식과 연결해야 합니다.
- **NewLookupRow** 값이 1 또는 2일 경우 행을 캐시된 대상에 전달하는 라우터 변환을 사용합니다.
- **NewLookupRow** 값이 0일 경우 행을 삭제하는 라우터 변환을 사용합니다. 그렇지 않으면 행을 다른 대상에 출력할 수 있습니다.
- 통합 서비스가 동일한 값을 조회 캐시에 쓰는 대상에 출력하는지 확인합니다. 업데이트 시 새 값을 출력하기로 선택하는 경우 입력/출력 포트 대신 조회/출력 포트만 대상 테이블에 연결합니다. 업데이트 시 이전 값을 출력하도록 선택한 경우 조회 변환 후 그리고 라우터 변환 전에 식 변환을 추가하십시오. 대상 테이블의 각 포트마다 식 변환에서 출력 포트를 추가하고 대상에 **Null** 입력 값을 출력하지 않도록 확인하는 식을 작성하십시오.
- 조회 SQL 재정의의 사용하는 경우 조회를 위해 적절한 대상에 올바른 열을 매핑합니다.
- **WHERE** 절을 조회 SQL 재정의에 추가하는 경우 조회 변환 전에 필터 변환을 사용합니다. 이렇게 하면 통합 서비스가 **WHERE** 절에 일치하는 동적 캐시 및 대상 테이블에 행을 삽입합니다.



- 동적 캐시를 사용하도록 재사용 가능 조회 변환을 구성하는 경우 매핑에서 **동적 조회 캐시** 속성을 비활성화하거나 조건을 편집할 수 없습니다.
- 조회 변환 후 업데이트 전략 변환을 사용하여 대상의 삽입 또는 업데이트를 위해 행에 플래그를 지정합니다.
- 조회 변환에서 업데이트 기타 항목 삽입 속성을 사용하려면 조회 변환 전에 업데이트 전략 변환을 사용하여 일부 또는 모든 행을 업데이트로 정의합니다.
- 세션 속성에서 행 유형을 데이터 구동으로 설정합니다.
- 세션 속성에서 대상 테이블 옵션에 대해 삽입 및 업데이트 시 업데이트를 선택합니다.

## 제 28 장

# 일치 변환

이 장에 포함된 항목:

- [일치 변환 개요, 422](#)
- [일치 분석, 423](#)
- [일치 점수 계산, 426](#)
- [마스터 데이터 분석, 429](#)
- [ID 일치 분석 및 지속형 인덱스 데이터, 430](#)
- [일치 매핑의 성능, 431](#)
- [ID 분석의 일치 성능, 433](#)
- [일치 변환 보기, 436](#)
- [일치 변환 포트, 437](#)
- [일치 맵렛, 441](#)
- [일치 분석 작업 구성, 443](#)
- [일치 변환 - 비원시 환경, 443](#)

## 일치 변환 개요

일치 변환은 레코드 간의 유사점 수준을 분석하는 활성 변환입니다. 하나의 데이터 집합 또는 두 개의 데이터 집합 간에 중복 정보가 포함된 레코드를 찾으려면 일치 변환을 사용합니다.

일치 변환에서 입력 포트의 값을 분석하고 값 사이의 유사점 정도를 나타내는 숫자 점수 집합을 생성합니다. 여러 포트를 선택하여 입력 레코드 간 전체 유사점 수준을 확인할 수 있습니다. 최소 점수를 임계값으로 지정하여 중복 정보를 포함할 가능성이 높은 레코드를 식별합니다.

다음 데이터 프로젝트에서 일치 변환을 사용할 수 있습니다.

- 고객 관계 관리. 예를 들어, 상점에서 우편 캠페인을 계획하고 있고 고객 데이터베이스에서 중복되는 고객 레코드를 확인해야 합니다.
- 인수 합병. 예를 들어, 한 은행이 동일한 지역의 다른 은행을 인수하는데 두 은행의 고객 중 일부는 공통된 고객입니다.
- 규정 준수 이니셔티브. 예를 들어, 모든 데이터 시스템에 중복 레코드가 없도록 요구하는 정부 또는 산업 규정 하에서 영업하고 있는 기업이 있습니다.
- 재무 위험 관리. 예를 들어, 은행이 예금주 간의 관계를 검색하고자 할 수 있습니다.

- 마스터 데이터 관리. 예를 들어, 소매점 체인에 고객 레코드의 마스터 데이터베이스가 있고, 체인의 각 소매점은 정기적으로 마스터 데이터베이스에 레코드를 제출합니다.
- 데이터 집합에서 중복 레코드를 식별해야 하는 프로젝트.

## 일치 분석

서로 다른 유형의 중복 분석을 일치 변환에 정의할 수 있습니다. 정의하는 중복 분석 작업은 매핑의 데이터 소스 수와 소스에 포함된 정보 유형에 따라 다릅니다.

일치 변환을 구성하는 경우 다음과 같은 요인을 고려하십시오.

- 데이터 집합에서 열 하나를 선택하거나 여러 개의 열을 선택할 수 있습니다.
- 데이터 소스 하나의 열을 분석하거나 2개의 데이터 소스를 분석할 수 있습니다.
- 입력 포트 필드의 원시 데이터를 분석하는 일치 변환을 구성하거나 데이터의 ID 정보를 분석하는 일치 변환을 구성할 수 있습니다.
- 서로 다른 유형의 출력을 기록하도록 일치 변환을 구성할 수 있습니다. 선택한 출력 유형에 따라 변환이 기록하는 레코드 수와 레코드의 순서가 결정됩니다.
- 성능을 높이려면 입력 레코드를 그룹으로 정렬한 다음 일치 분석을 수행합니다.

## 열 분석

일치 변환을 구성할 때 분석에 대해 하나 이상의 열을 선택하십시오.

일치 변환은 열을 쌍으로 분석합니다. 분석할 열을 하나만 선택할 경우 일치 변환이 열의 임시 사본을 작성한 다음 소스 열과 임시 열을 비교합니다. 분석할 열을 2개 선택할 경우 변환에서 선택한 두 열의 값을 비교합니다. 변환에서 한 열의 각 값을 다른 열의 모든 값과 비교합니다. 변환은 분석하는 각 값 쌍에 대해 일치 점수를 반환합니다.

일치 변환에서 전략을 구성할 때 분석할 열을 선택합니다. 전략은 분석할 열과 해당 열에 적용할 알고리즘을 지정합니다. 알고리즘은 각 값 쌍 간의 유사성 수준을 계산합니다. 변환의 다른 알고리즘은 다른 기준을 사용하여 값 간의 유사성 수준을 측정합니다. 한 변환에 여러 개의 전략을 정의하고 각 전략에 다른 열을 할당할 수 있습니다.

### 열 분석 예제

성 데이터의 열 값을 비교하기 위해 데이터 소스 및 일치 변환을 포함하는 매핑을 작성하고 *Surname* 포트를 일치 변환에 연결합니다. 변환은 매핑이 실행될 때 *Surname* 포트에서 데이터의 임시 사본을 작성합니다.

다음 이미지는 성 데이터의 조각을 보여줍니다.

	A	B
1	Surname	Surname_1
2	Annan	Annan
3	Baker	Baker
4	Barker	Barker
5	Edwards	Edwards
6	Parker	Parker
7	Smith	Smith
8	Smith	Smith
9	Zhang	Zhang

매핑은 다음 값이 중복되었다는 사실을 나타내는 일치 점수 집합을 생성합니다.

- Baker, Baker
- Barker, Parker
- Smith, Smith

데이터를 검토할 때 *Baker, Baker* 및 *Parker*가 중복 값이 아님을 결정합니다. 또한 *Smith*와 *Smith*는 중복 값을 결정합니다.

## 단일 소스 분석 및 이중 소스 분석

하나 또는 두 개의 데이터 소스에서 데이터를 분석하도록 일치 변환을 구성할 수 있습니다. 변환에서 전략을 정의할 때 각 데이터 소스에서 포트를 선택하십시오.

단일 소스 분석을 수행하도록 변환을 구성하는 경우 단일 데이터 집합에서 하나 이상의 포트를 선택합니다. 이중 소스 분석을 수행하도록 변환을 구성하는 경우 각 데이터 집합에서 하나 이상의 포트를 선택합니다. 포트를 쌍으로 선택합니다. 선택한 각 포트 쌍에 대해 변환이 한 포트의 각 값을 다른 포트의 모든 값과 비교합니다. 단일 열의 데이터에서 단일 소스 분석을 수행하는 경우 변환은 선택한 포트에 대한 임시 사본을 작성합니다.

**참고:** ID 일치 분석을 수행할 때 데이터 소스를 이전 매핑에서 작성한 ID 데이터의 지속형 인덱스와 비교할 수 있습니다. **일치 유형** 옵션을 사용하여 지속형 인덱스가 있는 ID 분석을 지정합니다.

## 필드 일치 분석 및 ID 일치 분석

필드 일치 분석 또는 ID 일치 분석을 수행하도록 일치 분석을 구성할 수 있습니다.

필드 일치 분석에서 일치 변환은 변환을 시작한 소스 데이터를 분석합니다. 모든 유형의 데이터에 대해 필드 일치 분석을 수행할 수 있습니다. ID 일치 분석에서 일치 변환은 입력 데이터로부터 대체 데이터 값의 인덱스를 생성하고 인덱스 데이터를 분석합니다. 입력 포트에 ID 데이터가 포함된 경우 ID 일치 분석에 대해 일치 변환을 구성하십시오. ID는 개인 또는 조직을 식별하는 데이터 값의 그룹입니다.

데이터 집합은 여러 가지 다른 방법으로 하나의 ID를 나타낼 수 있습니다. 예를 들어 다음 데이터 값은 John Smith라는 이름을 나타냅니다.

- John Smith
- Smith, John
- jsmith@email.com
- SMITHJMR

일치 변환은 레코드에서 ID 데이터를 읽고 ID의 가능한 대체 버전을 계산합니다. 그런 다음 ID의 현재 버전과 대체 버전을 포함하는 인덱스를 작성합니다. 일치 변환은 입력 레코드의 값이 아닌 인덱스 값을 분석합니다.

## ID 인구집단 파일

ID 일치 작업은 인구집단이라고 하는 참조 데이터 파일을 읽습니다. 인구집단 파일은 ID 데이터의 가능한 변형을 정의합니다. 이 파일은 Informatica 응용 프로그램을 통해 설치되지 않습니다. Informatica에서 인구집단 데이터 파일을 구입하여 다운로드해야 합니다.

인구집단 파일은 콘텐츠 관리 서비스가 액세스할 수 있는 위치에 설치해야 합니다. 콘텐츠 관리 서비스의 위치는 Informatica Administrator를 사용하여 설정할 수 있습니다.

## 일치 분석의 그룹

변환에서 수행해야 하는 데이터 수 비교로 인해 일치 분석 매핑을 실행하는 데 시간이 좀 걸릴 수 있습니다. 비교 수는 선택한 포트의 데이터 값 수와 관련됩니다.

다음 테이블에는 단일 포트의 다른 데이터 값 수에 대해 매핑이 수행하는 계산의 수를 보여줍니다.

데이터 값의 수	비교 수
10,000	5천만
100,000	5십억
1백만	5천억

매핑 실행 시간을 줄이려면 입력 데이터 레코드를 그룹에 할당하십시오. 그룹은 지정된 포트의 동일한 값을 포함하는 레코드의 집합입니다. 그룹화된 데이터에서 일치 분석을 수행하면 일치 변환에서 각 그룹 내의 레코드를 분석합니다. 변환은 한 그룹의 레코드를 다른 그룹의 레코드와 비교하지 않습니다. 그룹을 사용하면 변환이 매핑 분석 시 정확하게 비교해야 하는 레코드의 총 수가 줄어듭니다.

데이터를 그룹으로 구성할 때 다음 규칙 및 지침을 고려하십시오.

- 데이터를 그룹화할 포트는 그룹 키 포트입니다. 그룹 키 포트에는 주소 데이터 집합의 도시 또는 주 이름과 같이 다양한 중복 값이 있어야 합니다. 매핑 데이터에 사용 가능한 그룹 키 포트가 없는 경우 키 생성기를 사용하여 현재 매핑 데이터에서 포트를 작성합니다. 키 생성기 변환의 그룹 키 출력 포트를 일치 변환에 연결합니다.
- 또한 키 생성기 변환을 사용하여 시퀀스 식별자를 매핑 데이터에 추가할 수 있습니다.
- 필드 일치 작업에서 그룹 키 포트를 지정해야 합니다. ID 분석에 대해 일치 변환을 구성한 경우 그룹 키 포트를 선택하지 마십시오. ID 분석에서 ID 인덱스 데이터에 대한 그룹 키를 생성합니다.
- 일치 분석에서 사용하려는 그룹 키 포트를 지정하지 마십시오.
- 그룹을 작성할 때는 그룹의 크기가 유효한지 확인해야 합니다. 그룹이 너무 작으면 일치 분석이 데이터 집합에서 중복 데이터를 찾지 못할 수 있습니다. 그룹이 너무 크면 일치 분석에서 잘못된 중복을 반환할 수 있습니다. 평균 크기가 레코드 10,000개 정도인 그룹을 작성하는 그룹 키를 선택하십시오.
- 그룹은 매핑 데이터 집합에서 레코드 위치 순서를 다시 지정하지 않습니다.

## 일치 쌍 및 클러스터

일치 변환에서 다른 개수의 입력 행과 출력 행을 읽고 기록할 수 있으며, 출력 행의 순서를 변경할 수 있습니다. 일치 분석의 결과에 대한 출력 형식을 결정하십시오.

변환에서 다음 형식으로 행을 기록할 수 있습니다.

### 일치하는 쌍

변환이 일치 임계값을 충족하는 점수와 일치하는 모든 레코드 쌍에 대한 행을 씁니다. 변환이 각 레코드 쌍을 단일 행에 씁니다.

레코드가 하나 이상의 다른 레코드와 일치할 수 있으므로 레코드가 하나 이상의 출력 행에 나타날 수 있습니다.

### 가장 잘 일치

변환에서 데이터 집합의 각 레코드에 대한 행을 쓰고 다른 데이터 집합에서 가장 유사한 레코드를 동일한 행에 추가합니다.

### 클러스터

변환이 레코드 간 유사점 수준에 따라 출력 레코드를 클러스터에 할당합니다. 클러스터는 각 레코드가 일치 임계값을 충족하는 점수를 바탕으로 하나 이상의 다른 레코드와 일치하는 레코드 집합입니다. 변환이 각 레코드를 단일 행에 씁니다.

클러스터의 각 레코드는 클러스터에서 하나 이상의 다른 레코드와 일치해야 합니다. 따라서 클러스터에는 서로 일치하지 않는 레코드 쌍이 포함될 수 있습니다. 레코드가 다른 레코드와 일치하지 않는 경우 클러스터에 단일 레코드가 포함될 수 있습니다.

**참고:** 필드 분석의 클러스터 옵션은 ID 분석의 클러스터 - 모두 일치 옵션에 해당합니다. ID 분석의 클러스터 - 가장 잘 일치 옵션은 클러스터 계산 및 일치하는 쌍 계산을 결합합니다.

변환의 **일치 출력** 보기에서 출력 옵션을 구성하십시오.

### 관련 항목:

- [“클러스터 출력 옵션” 페이지 427](#)

## 일치 점수 계산

일치 점수는 두 열 값 사이의 유사 정도를 나타내는 숫자 값입니다. 알고리즘은 0~1 범위의 10진수 값으로 일치 점수를 계산합니다. 두 열 값이 동일할 경우 알고리즘에서 점수 1을 할당합니다.

분석을 위해 여러 열 쌍을 선택한 경우 변환에서 선택된 열의 점수를 기반으로 평균 점수를 계산합니다. 기본적으로 변환에서 각 열 쌍의 점수에 동일한 가중치를 할당합니다. 변환은 데이터 집합에서 열 데이터의 상대적인 중요도를 유추하지 않습니다.

변환에서 일치 점수를 계산하기 위해 사용하는 가중치 값을 편집할 수 있습니다. 데이터 집합의 열에 더 높거나 더 낮은 우선 순위를 할당할 경우 가중치 값을 편집하십시오.

변환이 열에서 **null** 값을 찾은 경우 적용할 점수를 설정할 수도 있습니다. 기본적으로 변환에서는 **null** 값을 데이터 오류로 처리하고 **null** 값을 포함하는 값 쌍에 낮은 일치 점수를 할당합니다.

**참고:** 선택한 알고리즘에서 두 값 사이의 일치 점수를 결합합니다. 알고리즘에서 두 값에 대한 단일 점수를 생성합니다. 일치 점수는 일치 출력 유형이나 사용자가 선택한 평가 방법 유형에 종속되지 않습니다.

## 가중치 점수

일치 분석을 위해 여러 열을 선택하는 경우 변환은 열의 점수에 따라 각 레코드의 평균 점수를 계산합니다. 평균 점수는 각 열에 비교 알고리즘에 적용하는 가중치 값을 포함합니다.

기본적으로 모든 알고리즘은 가중치 값으로 0.5를 사용합니다. 선택한 열이 중복 정보를 포함할 가능성이 높은 경우 이 값을 늘릴 수 있습니다. 선택한 열의 중복 값이 레코드 간의 진짜 중복 정보를 나타낼 가능성이 적은 경우 가중치 값을 줄일 수 있습니다. 마스크 변환은 평균 점수를 각 쌍의 레코드에 대한 단일 일치 점수로 사용합니다.

## Null 일치 점수

일치 알고리즘은 값 쌍의 값 하나 또는 두 값이 모두 null일 때 미리 정의된 일치 점수를 해당 값 쌍에 적용합니다. 필드 일치 알고리즘이 null 값에 적용하는 일치 점수를 편집할 수 있습니다.

### Null 일치 점수 및 필드 일치 알고리즘

필드 일치 알고리즘을 구성할 때는 알고리즘이 null 데이터에 적용하는 일치 점수 값을 확인하십시오. 필드 일치 알고리즘은 두 값을 비교할 때 값 중 하나 또는 두 값이 모두 null이면 기본 점수인 0.5를 적용합니다. 점수가 0.5이면 데이터 값 간의 유사점 수준이 낮은 것입니다.

null 일치 점수를 확인할 때는 다음 규칙 및 지침을 고려하십시오.

- 알고리즘이 기본 키 또는 기타 중요 데이터를 포함하는 열을 분석할 때는 기본 점수를 편집하지 마십시오. 이 경우 null 값은 데이터 오류를 나타내며 데이터에는 기본 점수를 사용하는 것이 적절합니다.
- 알고리즘이 선택적으로 데이터를 포함할 수 있는 열을 분석할 때는 null 일치 점수 값을 일치 임계값과 같은 값으로 업데이트하십시오. null 일치 점수를 일치 임계값으로 설정하면 일치 분석에서 null 값을 사용해도 아무런 영향을 주지 않습니다.

### Null 일치 점수 및 ID 일치 알고리즘

ID 일치 알고리즘은 두 값을 비교할 때 값 중 하나 또는 두 값이 모두 null이면 일치 점수 0을 적용합니다. ID 일치 분석에서는 null 일치 점수가 포함된 레코드를 고유 레코드 클러스터에 할당하고 클러스터 크기 값 1을 기록합니다. ID 일치 알고리즘이 null 데이터에 적용하는 점수는 편집할 수 없습니다.

## 클러스터 출력 옵션

출력 데이터에서 유사하거나 동일한 레코드를 구성하려는 경우 클러스터 출력 옵션을 선택합니다.

클러스터 출력 옵션을 선택하면 변환이 클러스터 ID 값을 각 출력 레코드에 추가합니다. 클러스터 ID 값으로 레코드를 정렬할 수 있습니다. 변환 출력은 모든 레코드에 대한 행을 포함합니다. 레코드가 일치 임계값을 충족하는 점수의 다른 레코드와 일치하지 않는 경우 변환이 고유 클러스터 ID를 레코드에 할당합니다. **일치 출력** 보기를 사용하여 클러스터 출력 옵션을 선택하거나 업데이트합니다.

다음 클러스터 출력 옵션을 선택할 수 있습니다.

#### 클러스터

옵션을 선택하여 클러스터 ID 값을 출력 레코드로 할당합니다.

#### 클러스터 - 가장 잘 일치

옵션을 선택하여 일치 점수가 가장 높은 레코드 쌍을 클러스터에 추가합니다. 레코드가 한 개 이상의 가장 잘 일치하는 다른 레코드를 나타낼 수 있으므로 한 개 이상의 레코드 쌍이 클러스터 ID 값을 공유할 수 있습니다.

#### 클러스터 - 모두 일치

클러스터 - 모두 일치 옵션은 클러스터 옵션과 같은 방식으로 작동합니다.

변환은 ID 일치 분석에서 클러스터 - 모두 일치 및 클러스터 - 가장 잘 일치를 옵션 이름으로 사용합니다.

**참고:** 데이터 통합 서비스에서 여러 개의 일치 변환을 동시에 실행하는 경우 데이터 통합 서비스는 각 변환의 출력에 대해 고유한 클러스터 ID 값을 생성합니다. 따라서 각 변환에서 생성하는 레코드의 클러스터 ID 값은 비연속적일 수 있습니다.

### 클러스터 옵션 및 클러스터 - 모두 일치 옵션

필드 일치 분석에서 클러스터 옵션을 선택합니다. ID 일치 분석에서 클러스터 - 모두 일치 옵션을 선택합니다.

일치 변환은 다음 규칙을 사용하여 클러스터를 작성합니다.

- 두 레코드의 일치 점수가 일치 임계값을 충족하는 경우 일치 변환이 레코드를 클러스터에 추가합니다.
- 데이터 집합의 레코드가 클러스터의 레코드와 일치할 경우 일치 변환이 레코드를 클러스터에 추가합니다.
- 한 클러스터의 레코드가 다른 클러스터의 레코드와 일치할 경우 클러스터가 병합됩니다.
- 일치 변환은 모든 레코드가 클러스터에 속할 때까지 일치 결과의 반복 스윙을 수행합니다.
- 레코드가 데이터 집합의 다른 레코드와 일치하지 않는 경우 변환이 고유 클러스터 ID 값을 레코드에 할당합니다.

### 클러스터 - 가장 잘 일치 옵션

ID 일치 분석에서 클러스터 - 가장 잘 일치 옵션을 선택합니다.

변환은 다음 규칙을 사용하여 클러스터를 작성합니다.

- 일치 변환은 현재 레코드와의 일치 점수가 가장 높은 레코드를 식별합니다. 일치 점수가 임계값을 충족하는 경우 일치 변환이 레코드 쌍을 클러스터에 추가합니다.
- 일치하는 레코드 중 하나가 클러스터에 있는 경우 일치 변환이 다른 레코드를 현재 클러스터에 추가합니다.
- 일치 변환은 모든 레코드가 클러스터에 속할 때까지 일치 점수 결과의 반복 스윙을 수행합니다.
- 레코드가 데이터의 다른 레코드와 일치하지 않는 경우 클러스터에 단일 레코드가 포함될 수 있습니다.

**참고:** 일치 출력 보기에서 일치 속성을 사용하여 변환이 단일 데이터 소스를 지속형 데이터 저장소와 비교하는 방법을 지정할 수 있습니다. 일치 속성은 변환이 소스 데이터 또는 지속형 데이터 저장소 내의 중복을 찾을지 여부를 결정합니다.

### 관련 항목:

- [“일치 쌍 및 클러스터” 페이지 425](#)

## 클러스터 분석의 드라이버 점수 및 링크 점수

일치 변환에서 클러스터 출력 옵션을 선택한 경우 링크 점수 및 드라이버 점수 데이터를 출력에 추가할 수 있습니다.

링크 점수는 레코드를 동일한 클러스터의 멤버로 식별하는 두 레코드 간의 점수입니다. 레코드 간의 링크에 따라 클러스터의 구성이 결정됩니다. 모든 레코드는 동일한 클러스터 내 다른 레코드에 연결할 수 있습니다.

드라이버 점수는 클러스터 내에서 가장 높은 시퀀스 ID 값을 가진 레코드와 동일한 클러스터 내 다른 레코드 간의 점수입니다. 드라이버 점수는 클러스터의 모든 레코드를 단일 레코드에 대해 평가할 수 있는 수단을 제공합니다. 드라이버 점수를 일치 출력에 추가할 경우 모든 클러스터가 완료될 때까지 일치 변환에서 드라이버 점수를 계산할 수 없기 때문에 매핑이 더 느리게 실행됩니다.

**참고:** 일치 분석에서는 사용자가 정의한 각 전략에 대한 단일 점수 집합을 생성합니다. 드라이버 점수와 링크 점수는 각 클러스터의 다른 레코드 쌍에 대한 일치 점수를 나타냅니다. 드라이버 점수와 링크 점수는 레코드에서 변환을 입력하는 순서에 종속됩니다. 드라이버 점수는 일치 임계값보다 낮을 수 있습니다.

### 클러스터 분석 예제

성 데이터 열을 분석하기 위한 필드 일치 전략을 구성합니다. 전략에 0.825의 일치 임계값을 설정합니다. 클러스터된 출력 형식을 선택하고, 변환에서 데이터 뷰어를 실행합니다.



다음 표에는 데이터 뷰어에서 표시하는 데이터가 나와 있습니다.

성	시퀀스 ID	클러스터 ID	클러스터 크기	드라이버 ID	드라이버 점수	링크 ID	링크 점수
SMITH	1	1	2	1 - 6	1	1 - 1	1
SMYTH	2	2	2	1 - 3	0.833333	1 - 2	1
SMYTHE	3	2	2	1 - 3	1	1 - 2	0.833333
SMITT	4	3	1	1 - 4	1	1 - 4	1
SMITS	5	4	1	1 - 5	1	1 - 5	1
SMITH	6	1	2	1 - 6	1	1 - 1	1

데이터 뷰어에는 성 데이터에 대한 다음 정보가 포함되어 있습니다.

- **SMITT** 및 **SMITS**는 일치 임계값을 충족하는 점수를 가진 레코드와 일치하지 않습니다. 일치 변환에서 레코드가 데이터 집합에서 고유한지 확인합니다.  
**SMITT** 및 **SMITS**의 클러스터 크기는 1입니다. 클러스터 출력에서 고유한 레코드를 찾으려면 단일 레코드를 포함하는 클러스터를 검색하십시오.
- **SMITH** 및 **SMITH**의 링크 점수는 1입니다. 일치 변환에서 레코드가 동일한지 확인합니다. 변환에서 레코드를 단일 클러스터에 추가합니다.
- **SMYTH** 및 **SMYTHE**의 링크 점수는 0.833333입니다. 점수가 일치 임계값을 초과합니다. 따라서 변환에서 레코드를 단일 클러스터에 추가합니다.

## 마스터 데이터 분석

일치 변환에서 2개의 데이터 소스를 분석하는 경우 한 소스를 마스터 데이터 집합으로 식별해야 합니다. 일치 변환은 지정된 데이터 집합에 있는 각 레코드의 데이터 값을 두 번째 데이터 집합에 있는 모든 행의 해당하는 값과 비교합니다.

대부분의 조직에서 마스터 데이터 집합은 영구적이고 가치가 높은 데이터 저장소로 간주됩니다. 마스터 데이터 집합에 레코드를 추가하기 전에 일치 변환을 사용하면 해당 레코드로 인해 마스터 데이터에 중복 정보가 추가되는 않는지를 확인할 수 있습니다.

### 마스터 데이터 예제

고객의 계정 레코드에 대한 마스터 데이터 집합을 유지하는 은행이 있습니다. 이 은행은 마스터 데이터 집합을 새 고객 계정을 식별하는 레코드로 매일 업데이트합니다. 은행은 중복 분석 매핑을 사용하여 새 레코드로 인해 마스터 데이터 집합의 고객 정보가 중복되지 않는지 확인합니다. 마스터 데이터 집합과 새 계정 테이블은 공통된 구조에 기반하며 테이블은 동일한 유형의 데이터베이스를 사용합니다. 따라서 은행은 마스터 데이터 집합을 업데이트할 때마다 중복 분석 매핑을 재사용할 수 있습니다.

### 마스터 데이터 집합 분석의 방향

일치 변환은 단일 데이터 집합의 레코드를 다른 레코드와 단일 방향으로 비교합니다. 일치 변환은 마스터 데이터 집합의 각 레코드를 두 번째 데이터 집합의 모든 레코드와 비교합니다. 일치 변환은 두 번째 데이터 집합의 각 레코드를 마스터 데이터 집합의 모든 레코드와 비교하지 않습니다. 따라서 마스터 데이터 집합의 선택이 일치 분석의 결과에 영향을 줄 수 있습니다.

다음 표에서는 ID 일치 분석에서 비교할 수 있는 두 개의 데이터 집합을 보여 줍니다.

데이터 집합 1	데이터 집합 2
Alex Bell	Alexander Bell
Alexander Graham Bell	Thomas Edison
Alva Edison	Nicola Tesla
Marie Curie	Irene Joliot Curie
Dorothy Crowfoot	Dorothy Hodgkin

데이터 집합 1을 마스터 데이터 집합으로 선택하고 **가장 잘 일치** 출력 옵션을 선택하는 경우 출력에 다음 레코드가 포함됩니다.

- Alex Bell, Alexander Bell
- Alexander Graham Bell, Alexander Bell

데이터 집합 2를 마스터 데이터 집합으로 선택하고 **가장 잘 일치** 출력 옵션을 선택하는 경우 출력에 다음 레코드가 포함됩니다.

- Alexander Bell, Alex Bell

데이터 집합 2가 마스터 데이터 집합인 경우 출력 데이터에서 Alexander Bell이 Alex Bell과 이미 일치하므로 변환이 Alexander Bell과 Alexander Graham Bell을 일치시킬 수 없습니다.

## 매핑 재사용

마스터 데이터 집합에 정기적으로 데이터를 추가하는 경우 재사용할 수 있는 중복 분석 매핑을 구성하십시오. 마스터 데이터 집합과 비교한 데이터 소스에서 포트 구성이 변경되지 않은 경우 매핑을 재사용할 수 있습니다.

매핑을 실행한 경우 변환에서 마스터 데이터 및 더 최근 데이터를 지정했는지 확인하십시오. 다른 구성을 업데이트하지 않고 매핑을 실행할 수 있습니다.

## ID 일치 분석 및 지속형 인덱스 데이터

ID 정보를 분석하는 매핑을 실행하면 일치 변환이 데이터 집합의 대체 ID 버전을 저장하는 인덱스를 생성합니다. 기본적으로 일치 변환은 인덱스 데이터를 임시 파일에 기록합니다. 인덱스 데이터를 데이터베이스 테이블에 저장하도록 변환을 구성할 수 있습니다.

생성한 인덱스 테이블은 후속 매핑에서 재사용할 수 있는 데이터 저장소를 나타냅니다. 인덱스 테이블을 데이터 소스와 비교하고, 원하는 경우 데이터 소스의 인덱스 데이터로 인덱스 테이블을 업데이트할 수 있습니다. 변환에서 인덱스 테이블을 재생성하지 않기 때문에 후속 매핑은 더 빨리 실행됩니다. 추가로 인덱스 테이블은 ID 데이터의 신뢰할 수 있는 데이터 저장소를 나타냅니다.

## 관련 항목:

- [“ID 분석의 일치 성능” 페이지 433](#)

## 지속형 인덱스 데이터에 대한 규칙 및 지침

마스터 데이터 집합의 ID 정보를 분석하도록 일치 변환을 구성하는 경우 다음 규칙 및 지침을 고려하십시오.

- 마스터 데이터 집합에 대해 재사용 가능 인덱스를 생성하려면 인덱스 데이터를 데이터베이스 테이블에 쓰도록 변환을 구성합니다. 데이터베이스 테이블은 인덱스 데이터의 지속형 저장소로 구성됩니다.
- 인덱스 데이터 저장소에 설정된 다른 데이터 집합의 ID를 비교하려면 데이터 집합을 매핑 데이터 소스로 구성합니다. 데이터 소스 및 인덱스 데이터 저장소를 읽도록 일치 변환을 구성합니다. 지정하는 데이터베이스 연결의 기본 스키마에서 인덱스 테이블을 선택합니다.
- 일치 변환에서 입력 레코드의 시퀀스 식별자 값을 레코드에 해당하는 인덱스 데이터 행에 추가합니다. *SequenceID* 입력 포트에는 시퀀스 식별자가 포함됩니다. 변환은 시퀀스 식별자를 사용하여 일치 분석의 여러 단계에 걸쳐 인덱스 데이터를 추적합니다. 시퀀스 ID 포트 연결을 끊지 마십시오.

- 일치 변환을 인덱스 저장소에 연결하는 경우 변환이 저장소를 작성한 변환의 인구집단, 키 수준, 키 유형 및 키 필드 속성 값을 재사용합니다. 또한 변환은 저장소에 작성된 변환의 포트 구성을 재사용합니다.

변환 속성이 일치하지 않는 경우 ID 분석이 매핑 소스 데이터 및 인덱스 데이터를 제대로 비교할 수 없습니다.

- 일치 변환은 키 필드로 선택한 입력 포트의 데이터를 사용하여 ID 인덱스를 생성합니다. 변환은 다른 포트의 데이터를 인덱스에 쓸 수도 있습니다. 변환의 키가 아닌 필드 데이터 포트 연결을 끊는 경우 매핑을 실행할 때 해당 인덱스 열의 데이터를 지웁니다. 인덱스 테이블의 입력 포트 데이터를 유지하려면 입력 데이터 포트 연결을 끊지 마십시오.
- 데이터 집합에 대한 인덱스 테이블을 생성할 때 일치 변환의 일치 분석을 비활성화할 수 있습니다. 예를 들어 데이터 집합에 대해 인덱스 저장소를 작성할 때 일치 분석을 비활성화할 수 있습니다. 일치 분석을 비활성화하면 매핑이 더 빨리 실행됩니다.  
일치 분석을 비활성화할 때 일치 변환은 지속성 상태 코드 및 지속성 상태 설명을 생성할 수 있습니다. 변환은 일치 점수 또는 일치 분석의 결과와 연관된 기타 데이터를 생성하거나 표시하지 않습니다. 예를 들어 레코드를 클러스터에 할당하도록 변환을 구성하고 일치 분석을 비활성화하는 경우 변환은 클러스터 ID 값을 생성하거나 표시하지 않습니다.
- 일치 변환에서 인덱스 저장소를 매핑 소스의 데이터로 업데이트할지 여부를 결정하십시오. 일치 변환은 시퀀스 식별자를 사용하여 인덱스 저장소 및 매핑 데이터의 행이 동일한 레코드를 나타내는지 결정합니다.

## 일치 매핑의 성능

일치 변환이 포함된 매핑을 실행하기 전에 일치 변환의 성능을 결정하는 데이터 요소를 미리 볼 수 있습니다. 매핑을 실행하는 데 필요한 리소스가 시스템에 충분한지 확인할 수 있습니다. 입력 데이터의 유사성 수준을 측정하도록 변환을 올바르게 구성했는지 확인할 수도 있습니다.

**일치 성능 분석** 옵션을 사용하여 시스템에 필요한 리소스가 있는지 확인합니다. **일치 클러스터 분석** 옵션을 사용하여 매핑이 입력 데이터의 유사성 수준을 정확하게 측정할 수 있는지 확인합니다.

단일 데이터 소스를 읽는 모든 일치 변환에서 일치 성능 분석 및 일치 클러스터 분석을 실행할 수 있습니다. 이중 소스 필드 일치 분석을 수행하는 일치 변환에는 일치 성능 분석을 실행합니다. 인덱스 테이블에 연결된 ID 일치 전략에서 일치 성능 분석 또는 일치 클러스터 분석을 실행하지 마십시오.

## 일치 성능 분석 드릴다운

일치 분석 데이터를 드릴다운하여 일치 임계값을 충족하거나 초과하는 레코드 쌍을 볼 수 있습니다. **세부 정보** 보기에서 레코드를 두 번 클릭하고 데이터 뷰어를 사용하여 선택한 레코드와 일치하는 레코드를 봅니다. 데이터 뷰어에는 단일 행의 각 레코드 쌍에 대한 데이터가 표시됩니다. 행에는 각 레코드 쌍에 대한 행 식별자가 포함됩니다.

## 일치 클러스터 분석 드릴다운

클러스터 분석 데이터를 드릴다운하여 각 클러스터의 레코드를 볼 수 있습니다. **세부 정보** 보기에서 클러스터를 두 번 클릭하고 데이터 뷰어에서 데이터를 읽습니다. 데이터 뷰어는 한 번에 하나의 클러스터를 표시합니다. 클러스터 데이터에는 드라이버 점수, 링크 점수, 드라이버 식별자 또는 링크 식별자 등 사용자가 선택한 점수 옵션이 포함됩니다.

## 일치 변환 로깅

일치 변환을 사용하는 매핑을 실행하면 **Developer tool**의 로그가 매핑이 수행하는 비교 수 계산을 추적합니다. 로그 데이터를 보려면 데이터 뷰어에서 **로그 표시** 옵션을 선택합니다.

로그는 매핑의 비교 수가 100,000번이 될 때마다 업데이트됩니다.

# 일치 클러스터 분석 데이터 보기

변환이 작성할 수 있는 클러스터의 통계 데이터를 볼 수 있습니다. 클러스터 통계에는 현재 매핑 구성에 따른 데이터 집합의 레코드 중복 수준이 요약되어 있습니다.

데이터를 보려면 매핑 캔버스에서 일치 변환을 마우스 오른쪽 버튼으로 클릭하고 **일치 클러스터 분석**을 선택합니다.

해석을 실행하기 전에 변환이 들어 있는 매핑의 유효성을 검사합니다.

일치 클러스터 분석이 다음 속성에 대해 데이터를 표시합니다.

속성	설명
소스	입력 데이터 행 수입입니다.
최종 실행	분석 날짜 및 시간입니다.
검색된 총 클러스터 수	매핑이 실행될 때 일치 분석이 생성하는 클러스터 수입입니다.
최소 클러스터 크기	가장 적은 레코드가 포함된 클러스터의 레코드 수입입니다. 최소 클러스터 크기가 1인 경우 데이터 집합은 최소 한 개의 고유한 레코드를 포함합니다.
최대 클러스터 크기	가장 많은 레코드가 포함된 클러스터의 레코드 수입입니다. 이 값이 평균 클러스터 크기를 많이 초과하는 경우 가장 큰 클러스터에 잘못된 중복이 포함되어 있을 수 있습니다.
고유한 레코드 수	다른 레코드가 일치 임계값을 충족하는 점수와 일치하지 않는, 데이터 집합의 레코드 수입입니다.
중복 레코드 수	다른 레코드가 일치 임계값을 충족하는 점수와 일치하는, 데이터 집합의 레코드 수입입니다.
총 비교 수	매핑이 수행하는 비교 작업 수입입니다.
평균 클러스터 크기	클러스터의 평균 레코드 수입입니다.

## 일치 성능 분석 데이터 보기

매핑이 입력 데이터로 읽는 레코드 그룹에 대한 통계 데이터를 볼 수 있습니다.

데이터를 보려면 매핑 캔버스에서 일치 변환을 마우스 오른쪽 버튼으로 클릭하고 **일치 성능 분석**을 선택합니다.

해석을 실행하기 전에 변환이 들어 있는 매핑의 유효성을 검사합니다.

일치 성능 분석이 다음 속성에 대해 데이터를 표시합니다.

속성	설명
소스	입력 데이터 행 수입입니다.
최종 실행	분석 날짜 및 시간입니다.
검색된 총 그룹 수	선택한 그룹 키 값에 따라 데이터 집합에 대해 정의된 그룹 수입입니다.
처리량 (분당 레코드 수)	일치 분석의 속도를 예측하는 변수 값입니다. 이 값을 설정합니다. 이 값을 사용하여 일치 분석을 실행하는 데 필요한 시간을 예측합니다.
레코드 일치에 걸리는 예상 시간	일치 변환 구성에 따라 데이터 집합의 모든 레코드를 분석하는 데 걸린 시간입니다.
생성된 총 쌍 수	입력 데이터 행 수 및 그룹 수에 따라 변환이 수행해야 하는 비교 수입입니다.
최소 그룹 크기	그룹이 포함할 수 있는 최소 레코드 수를 나타내는 변수 값입니다. 이 값을 설정합니다. 이 값을 사용하여 매핑이 사용 가능한 크기의 그룹을 작성하는지 확인합니다. <b>참고:</b> 최소 그룹 크기 값은 매핑이 실행될 때 작성된 그룹의 크기를 확인하지 않습니다.
최소 임계값 아래의 그룹 수	최소 그룹 크기 값보다 적은 레코드가 포함된 그룹의 수입입니다. 최소 그룹 크기 미만인 그룹이 여러 개 있는 경우 변환을 편집하고 다른 그룹 키를 선택해야 할 수 있습니다.
최대 그룹 크기	그룹이 포함할 수 있는 최대 레코드 수를 나타내는 변수 값입니다. 성능 분석을 위해 이 값을 설정합니다. 이 값을 사용하여 매핑이 사용 가능한 크기의 그룹을 작성하는지 확인합니다. <b>참고:</b> 이 값은 매핑이 실행될 때 작성된 그룹의 크기를 확인하지 않습니다.
최대 임계값 위의 그룹 수	최대 그룹 크기 값보다 많은 레코드가 포함된 그룹의 수입입니다. 최대 그룹 크기를 초과하는 그룹이 여러 개 있는 경우 변환을 편집하고 다른 그룹 키를 선택해야 할 수 있습니다.

## ID 분석의 일치 성능

두 데이터 집합에 대한 ID 분석을 수행할 때 매핑 성능을 개선하려면 데이터베이스 테이블에서 ID 인덱스 데이터를 읽도록 일치 변환을 구성하십시오. 매핑을 실행하여 마스터 데이터 집합에 대한 인덱스 테이블을 작성합니다. 매핑을 다시 실행하여 인덱스 데이터와 다른 데이터 소스를 비교합니다.

**일치 유형** 보기의 옵션을 사용하여 인덱스 데이터를 저장하는 데이터베이스 테이블을 식별합니다. 인덱스 데이터를 다른 소스의 데이터와 비교하도록 변환을 구성할 때 동일한 옵션을 사용하여 인덱스 테이블도 선택합니다.

인덱스 데이터를 데이터베이스 테이블에 기록하려면 다음 태스크를 수행하십시오.

1. ID 정보의 데이터 소스를 읽는 매핑을 작성합니다.

2. 인덱스 데이터를 데이터베이스에 기록하는 일치 변환을 매핑에 구성합니다.
3. 매핑을 실행하여 인덱스 데이터를 생성합니다. 인덱스 데이터는 재사용할 수 있는 데이터 저장소를 나타냅니다.

데이터베이스 테이블에서 인덱스 데이터를 읽으려면 다음 태스크를 수행하십시오.

1. 다른 ID 데이터 소스를 읽는 매핑을 작성합니다.
2. 이전에 지정한 데이터베이스에서 인덱스 데이터를 읽는 일치 변환을 매핑에 구성합니다.  
매핑 데이터 소스와 인덱스 데이터가 공통된 구조를 공유하는 경우 인덱스 데이터를 생성한 매핑을 재사용할 수 있습니다.
3. 매핑을 실행하여 데이터 소스와 인덱스 데이터를 비교합니다.  
이 매핑은 데이터 소스에 대한 인덱스 데이터를 생성합니다. 더 큰 데이터 집합에 대한 인덱스 데이터를 생성할 필요가 없으므로 두 데이터 집합에 대한 인덱스 데이터를 생성하는 이중 소스 매핑보다 빠르게 실행됩니다.

## 관련 항목:

- [“ID 일치 분석 및 지속형 인덱스 데이터” 페이지 430](#)

## ID 인덱스 데이터에 대한 데이터 저장소 작성

ID 정보가 포함된 데이터 소스를 읽는 매핑을 구성합니다. 일치 변환을 사용하여 인덱스 데이터를 데이터베이스에 기록합니다.

1. 매핑을 작성하고 데이터 소스를 매핑 캔버스에 추가합니다.
2. 일치 변환을 매핑 캔버스에 추가합니다.
3. 데이터 소스에서 ID 정보가 포함된 포트를 선택합니다.
  - ID 정보 포트를 일치 변환에 연결합니다.
  - 시퀀스 식별자 값이 포함된 포트를 일치 변환에 연결합니다.
4. 일치 변환에서 **일치 유형** 보기를 선택합니다.
5. 일치 유형을 **지속형 레코드 ID로 ID 일치**로 설정합니다.
6. 인덱스 데이터 저장소를 작성하려면 다음 옵션을 구성하십시오.
  - 지속성 방법을 **데이터베이스를 새 ID로 업데이트**로 설정합니다.
  - 일치 프로세스 값을 확인합니다. 기본값은 **활성화**입니다. 일치 프로세스를 비활성화하면 매핑에서 ID 인덱스 테이블을 작성하지만 데이터에 대해 일치 분석을 수행하지 않습니다.
  - DB 연결 메뉴에서 인덱스 테이블에 대한 데이터베이스를 선택합니다.
  - 지속형 저장소 메뉴에서 **신규 작성**을 선택합니다. **저장소 테이블 작성** 대화 상자에서 인덱스 이름을 입력합니다.
7. 일치 변환에 필요한 다른 구성 단계를 수행합니다. 예를 들어 변환 전략을 구성합니다.
8. 대상 데이터 개체를 매핑에 추가합니다.
9. 일치 변환의 출력 포트를 대상 데이터 개체에 연결합니다.
10. 매핑을 실행합니다.

매핑이 데이터 소스에 대한 인덱스 데이터를 지정한 데이터베이스 테이블에 기록합니다.

## 단일 소스 분석에서 인덱스 데이터 저장소 사용

ID 정보의 데이터 소스를 읽는 매핑을 구성합니다. 일치 변환을 사용하여 데이터 소스를 마스터 데이터 집합의 인덱스 데이터 저장소와 비교합니다.

매핑을 구성하기 전에 데이터 소스에 시퀀스 식별자 값 열이 포함되어 있는지 확인합니다.

시간을 절약하기 위해 인덱스 데이터 저장소를 작성한 매핑을 복사하거나 재사용할 수 있습니다.

1. 인덱스 데이터 저장소를 생성한 매핑을 엽니다.  
또는 매핑의 사본을 엽니다.
2. 매핑에서 데이터 소스를 확인합니다.  
필요한 경우 데이터 소스를 현재 데이터가 포함된 소스로 교체합니다.  
**참고:** 데이터 소스를 삭제하는 경우 일치 변환에서 포트 연결도 삭제합니다.
3. ID 정보가 포함된 데이터 소스 포트를 식별합니다.
  - ID 정보 포트를 일치 변환에 연결합니다.
  - 시퀀스 식별자 값이 포함된 포트를 일치 변환에 연결합니다.  
입력 포트 및 입력 포트 순서는 인덱스 테이블에 작성된 변환의 입력 포트와 일치해야 합니다.
4. 일치 변환에서 **일치 유형** 보기를 선택합니다.
5. 일치 유형을 **지속형 레코드 ID로 ID 일치**로 설정합니다.
6. 인구집단, 키 수준, 키 유형 및 키 필드 값을 확인합니다.
7. 인덱스 데이터 저장소를 식별하는 옵션을 구성합니다.
  - 지속성 방법을 설정합니다. 예를 들어 **데이터베이스를 업데이트하지 않음**을 선택하여 인덱스 테이블의 현재 데이터를 유지합니다.
  - 일치 프로세스를 **활성화**로 설정합니다.
  - DB 연결 메뉴에서 인덱스 테이블이 포함된 데이터베이스를 선택합니다.
  - 지속형 저장소 메뉴에서 인덱스 데이터를 포함하는 테이블을 찾습니다.
8. 일치 출력 보기에서 다음 속성을 구성합니다.
  - 일치. 변환이 데이터베이스 테이블에서 인덱스 데이터를 읽을 때 분석할 레코드를 식별합니다.
  - 출력. 변환이 출력으로 쓰는 레코드를 필터링합니다.
9. 일치 변환에 필요한 다른 구성 단계를 수행합니다.  
예를 들어 변환 전략을 구성합니다.
10. 매핑에서 데이터 대상을 확인합니다.  
필요한 경우 데이터 대상을 다른 데이터 개체로 교체합니다.
11. 일치 변환의 출력 포트를 대상 데이터 개체에 연결합니다.
12. 매핑을 실행합니다.

매핑에서 데이터 소스 레코드와 인덱스 데이터 저장소를 비교합니다. 변환이 데이터 소스의 인덱스 데이터를 데이터 저장소에 씁니다.

# 일치 변환 보기

일치 변환에서 사용자가 일련의 보기에 구성할 수 있는 옵션을 구성합니다. 재사용 가능한 변환에서는 보기가 **Developer tool** 캔버스에 탭으로 표시됩니다. 보기를 열려면 탭을 클릭합니다. 재사용 불가능한 변환에서는 보기가 캔버스에 목록으로 표시됩니다. 보기를 열려면 목록에서 항목을 클릭합니다.

일치 분석 작업을 구성한 경우 다음 보기를 구성할 수 있습니다.

## 일반

재사용 불가능한 일치 변환의 이름 및 설명을 업데이트하려면 일반 보기를 사용합니다. 설명은 선택 사항입니다.

## 포트

재사용 불가능한 일치 변환에서 입력 포트 및 출력 포트를 확인하려면 포트 보기를 사용합니다.

## 개요

재사용 가능 변환의 이름 및 설명을 업데이트하려면 개요를 사용합니다. 설명은 선택 사항입니다. 또한 재사용 가능 변환에서 입력 포트 및 출력 포트를 작성하려면 개요를 사용합니다.

## 일치 유형

변환에서 수행하는 중복 분석 유형을 선택하려면 일치 유형 보기를 사용합니다. 필드 일치 분석 또는 ID 일치 분석을 선택할 수 있습니다. 단일 데이터 소스 또는 두 개의 데이터 소스를 지정할 수 있습니다.

## 전략

입력 데이터를 분석하는 전략을 하나 이상 정의하려면 전략 보기를 사용합니다. 각 전략에서 두 개의 데이터 열을 선택하고, 일치 분석 알고리즘을 열에 할당합니다.

## 일치 출력

출력 데이터의 구조와 형식을 지정하려면 일치 출력 보기를 사용합니다.

## 매개 변수

변환을 포함하는 매핑을 실행할 경우 데이터 통합 서비스가 변환에 적용할 수 있는 매개 변수를 정의하려면 매개 변수 보기를 사용합니다.

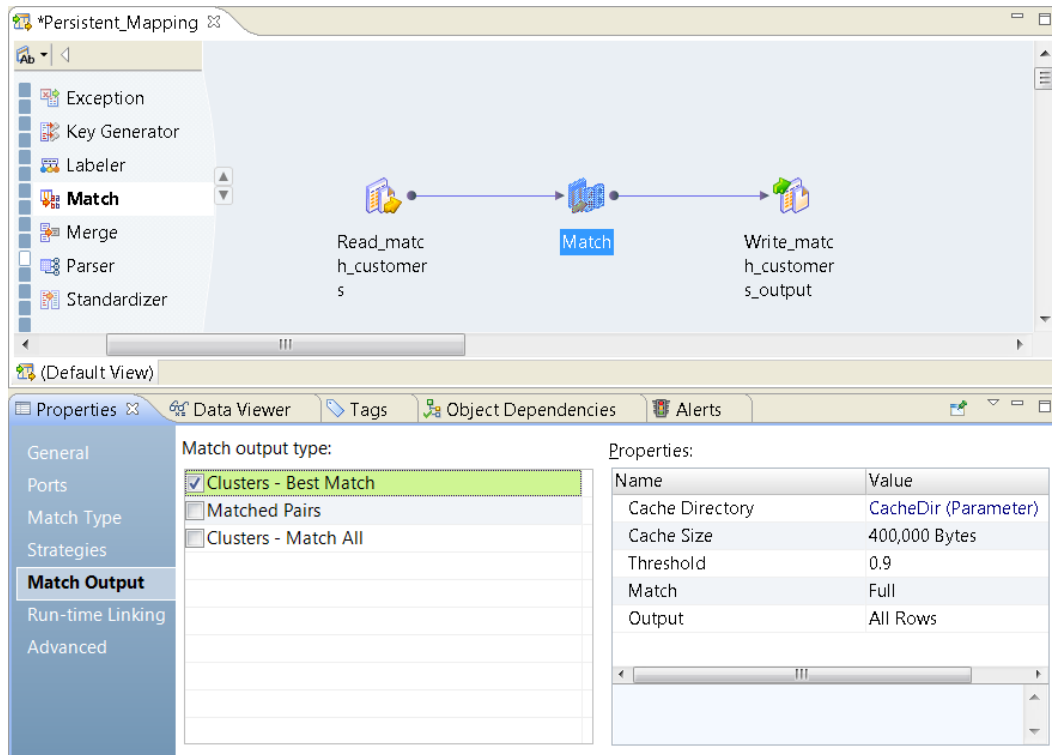
## 고급

다음 속성을 지정하려면 고급 보기를 사용합니다.

- 매핑에서 변환에 대해 기록하는 로그 메시지의 세부 정보 수준입니다.
- 변환을 포함하는 매핑을 실행할 경우 ID 일치 작업에서 사용하는 프로세스 수입니다.
- 변환에서 동일한 레코드의 데이터를 직접 출력 포트에 전달하는지 여부입니다. 클러스터된 출력을 기록 하도록 변환을 구성할 경우 동일한 레코드를 필터링할 수 있습니다.



다음 이미지는 지속형 인덱스 저장소를 사용하여 ID 분석에 대해 구성하는 일치 변환의 보기를 보여 줍니다.



## 일치 변환 포트

일치 변환에는 미리 정의된 입력 포트 및 출력 포트 집합이 포함됩니다. 여기에는 사용자가 정의한 일치 분석 작업을 위한 메타데이터가 포함됩니다. 일치 변환은 사용자가 구성한 일치 유형 및 일치 출력 옵션에 따라 이 포트를 선택하거나 선택 취소합니다.

일치 변환을 구성할 때는 메타데이터 포트를 검토하십시오. 일치 변환을 매핑에 추가하는 경우 업스트림 및 다운스트림 매핑 개체의 유효한 포트에 메타데이터 포트를 연결해야 합니다.

## 일치 변환 입력 포트

미리 정의된 입력 포트에는 변환에서 일치 분석하는 데 필요한 메타데이터가 포함되어 있습니다.

일치 변환을 작성한 후 다음 입력 포트를 구성할 수 있습니다.

### Sequenceld

매핑 소스 데이터 집합에서 각 레코드의 고유한 식별자입니다. 입력 데이터 집합의 모든 레코드에 고유한 시퀀스 식별자가 포함되어 있습니다. 데이터 집합에 중복된 시퀀스 식별자가 포함된 경우 일치 변환에서 중복 레코드를 제대로 식별할 수 없습니다. 데이터에 고유한 식별자가 없는 경우 키 생성기 변환을 사용하여 고유한 식별자를 작성하십시오.

ID 데이터에 대한 인덱스 데이터 저장소를 작성한 경우 일치 변환에서 각 레코드의 시퀀스 식별자를 데이터 저장소에 추가합니다. 변환에서 데이터 소스를 인덱스 데이터 저장소와 비교하도록 구성한 경우 변환이 두 데이터 집합에서 공통된 시퀀스 식별자를 찾을 수 있습니다. 시퀀스 식별자가 각 데이터 집합에서 고유한 경우 변환에서 시퀀스 식별자를 분석할 수 있습니다.

## GroupKey

레코드가 속하는 그룹을 식별하는 키 값입니다.

**참고:** 매핑 성능을 개선하려면 **GroupKey** 입력 포트 및 동일한 전체 자릿수 값을 사용하여 입력 포트에 연결하는 출력 포트를 구성하십시오.

## 일치 변환 출력 포트

미리 정의된 출력 포트에는 일치 변환이 수행하는 분석에 대한 메타데이터가 포함됩니다.

일치 변환을 작성한 후에 다음 출력 포트를 구성할 수 있습니다.

### GroupKey

레코드가 속한 그룹을 식별하는 키 값입니다.

연관 변환과 같은 다운스트림 변환은 그룹 키 값을 읽을 수 있습니다.

### ClusterId

레코드가 속하는 클러스터의 식별자입니다. 클러스터 출력에 사용됩니다.

### ClusterSize

레코드가 속하는 클러스터의 레코드 수입니다. 클러스터에 1개의 고유한 레코드가 포함되는 경우 클러스터 크기는 1입니다. 클러스터 출력에 사용됩니다.

### RowId 및 RowId1

레코드의 고유한 행 식별자입니다. 일치 변환은 일치 분석 작업 중에 행 식별자를 사용하여 행을 식별합니다. 식별자는 입력 데이터의 행 번호와 일치하지 않을 수 있습니다.

### DriverId

클러스터의 드라이버 레코드에 대한 행 식별자입니다. 클러스터 출력에 사용됩니다. 드라이버 레코드는 **SequenceId** 입력 포트 값이 가장 큰 클러스터의 레코드입니다.

### DriverScore

일치 변환은 일치하는 쌍 출력 및 클러스터된 출력의 드라이버 점수를 할당합니다. 일치하는 쌍에서 드라이버 점수는 레코드 쌍 간의 일치 점수입니다. 클러스터에서 드라이버 점수는 클러스터의 현재 레코드와 드라이버 레코드 간의 일치 점수입니다.

### LinkId

현재 레코드와 일치하고 현재 레코드를 클러스터에 연결한 레코드의 행 식별자입니다. 클러스터 출력에 사용됩니다.

### LinkScore

클러스터의 작성 또는 클러스터에 대한 레코드 추가를 야기하는 두 레코드 간의 일치 점수입니다. **LinkId** 포트는 현재 레코드와 링크 점수를 공유하는 레코드를 식별합니다. 클러스터 출력에 사용됩니다.

### PersistenceStatus

입력 레코드에 대한 일치 분석의 결과를 나타내는 8자리 코드입니다. 데이터 소스를 인덱스 데이터 저장소와 비교하는 변환의 단일 소스 ID 분석에 사용됩니다.

일치 변환은 코드의 처음 3개 문자를 채웁니다. 각 자리에는 서로 다른 문자가 반환될 수 있습니다. 4번째부터 8번째 자리에는 0이 반환됩니다.

일치하는 쌍 출력을 생성하도록 변환을 구성할 경우 변환이 **PersistenceStatus** 포트 및 **PersistenceStatus1** 포트를 작성합니다.

## PersistenceStatusDesc

지속성 상태 코드 값의 텍스트 설명입니다. 데이터 소스를 인덱스 데이터 저장소와 비교하는 변환의 단일 소스 ID 분석에 사용됩니다.

일치하는 쌍 출력을 생성하도록 변환을 구성할 경우 변환이 PersistenceStatusDesc 포트 및 PersistenceStatusDesc1 포트를 작성합니다.

## 지속성 상태 코드 및 지속성 상태 설명

지속성 상태 코드 및 지속성 상태 설명은 일치 변환에서 분석한 다른 인덱스 데이터 유형 간의 관계를 설명합니다. 지속형 ID 데이터 저장소를 읽도록 변환을 구성한 경우 변환에서 상태 코드 및 상태 설명을 생성합니다.

변환에서 지속성 상태 코드를 PersistenceStatus 포트에 기록합니다. 코드에는 8자가 포함됩니다. 변환은 문자열의 처음 3개 위치를 코드 값으로 채웁니다. 4번째부터 8번째 자리에는 0이 반환됩니다.

변환에서 지속성 상태 설명을 PersistenceStatusDesc 포트에 기록합니다. 설명에는 지속성 상태 코드의 처음 3개 위치에 있는 값이 설명되며 쉼표로 구분된 텍스트 문자열이 포함되어 있습니다.

변환은 소스 데이터 레코드의 시퀀스 식별자 값을 사용하여 두 데이터 집합에 대한 인덱스 데이터를 비교합니다.

다음 테이블에는 변환에서 상태 설명 및 상태 코드의 각 위치에 기록한 정보 유형이 설명됩니다.

위치	설명
1	레코드를 포함하는 데이터 집합을 식별합니다.
2	레코드의 중복 상태를 표시합니다. 변환에서 변환 입력 데이터와 인덱스 데이터 저장소 간의 공통 시퀀스 식별자를 찾습니다.
3	변환이 데이터에서 수행하는 모든 작업을 설명합니다.
4-8	상태 코드의 각 위치에 0이 포함됩니다. 상태 설명에는 다음 위치에 대한 텍스트가 포함되지 않습니다.

## 상태 코드 값 및 상태 설명 값

지속성 상태 코드 및 지속성 상태 설명은 변환 입력 레코드와 데이터 저장소가 나타내는 레코드 간의 관계를 설명합니다. 변환은 시퀀스 식별자 값을 사용하여 레코드를 식별하고 데이터 집합에서 레코드 간의 관계를 결정합니다.

지속성 상태 코드 및 지속성 상태 설명은 공통 구조를 가집니다. 상태 코드 및 상태 설명은 출력 데이터 문자열의 각 위치에 동일한 정보를 가집니다.

## 데이터 집합 상태

상태 코드 및 상태 설명의 첫 번째 값은 레코드를 포함하는 데이터 집합을 식별합니다.

다음 테이블에는 변환이 첫 번째 위치에서 반환할 수 있는 상태 코드 및 상태 설명이 설명됩니다.

상태 코드	상태 설명
S	저장소. 현재 레코드가 인덱스 데이터 저장소에서 생성됩니다.
I	입력입니다. 현재 레코드가 변환 입력 데이터에서 생성됩니다.

### 중복 레코드 상태

상태 코드 및 상태 설명의 두 번째 값은 변환 인덱스 데이터 및 지속형 데이터 저장소 간의 관계를 설명합니다.

다음 테이블에는 변환이 두 번째 위치에서 반환할 수 있는 상태 코드 및 상태 설명이 설명됩니다.

상태 코드	상태 설명
A	부재. 인덱스 데이터 저장소에는 현재 레코드에 대한 데이터가 포함되지 않습니다.
E	존재. 현재 레코드는 인덱스 데이터 저장소 및 변환 입력 데이터에 있습니다.
I	올바르지 않음. 변환에서 현재 레코드를 분석할 수 없습니다. 예를 들어 일치 유형 탭의 키 필드가 레코드 데이터와 호환되지 않아서 변환에서 해당 레코드에 대한 인덱스 데이터를 생성할 수 없습니다.
N	신규. 레코드가 데이터 소스에 있습니다.
O	[대시] 레코드가 인덱스 데이터 저장소에 있습니다.

### 데이터 저장소 상태

상태 코드 및 상태 설명에 있는 세 번째 값은 변환이 인덱스 데이터 테이블에서 수행하는 모든 작업을 설명합니다.

다음 테이블에는 변환이 세 번째 위치에서 반환할 수 있는 상태 코드 및 상태 설명이 설명됩니다.

상태 코드	상태 설명
A	추가됨. 변환에서 현재 입력 레코드의 인덱스 데이터를 지속형 데이터 저장소에 추가합니다. 변환 입력 데이터 및 지속형 인덱스 데이터는 다른 시퀀스 식별자를 가집니다.
I	무시됨. 변환에서 현재 입력 레코드의 어떤 인덱스 데이터도 지속형 데이터 저장소에 추가하지 않습니다.

상태 코드	상태 설명
N	변환이 다음 설명 중 하나를 반환합니다. <ul style="list-style-type: none"> <li>- 변경 내용 없음. 현재 레코드가 지속형 데이터 저장소에서 생성되며 변환은 작업을 수행하지 않습니다.</li> <li>- 추가되지 않음. 정의한 일치 정책 때문에 변환이 지속형 데이터 저장소를 현재 입력 레코드의 데이터로 업데이트하지 않습니다.</li> </ul>
R	제거됨. 변환이 인덱스 데이터 저장소에서 레코드 인덱스 데이터를 제거합니다.
U	업데이트됨. 변환이 지속형 데이터 저장소의 행을 변환 입력 레코드의 인덱스 데이터로 업데이트합니다. 변환 입력 데이터 및 지속형 인덱스 데이터는 공통 시퀀스 식별자를 가집니다.

### 지속성 상태 설명 예제

지속성 상태 코드 **INA00000**에는 다음과 같은 지속성 상태 설명이 있습니다.

*입력, 신규, 추가됨*

상태 코드 및 상태 설명은 다음 레코드 정보를 포함합니다.

- 레코드가 변환 입력 데이터에서 생성됩니다.
- 지속형 데이터 저장소에는 레코드 사본이 없습니다.
- 변환이 레코드의 인덱스 데이터를 지속형 데이터 저장소에 추가합니다.

## 출력 포트 및 일치 출력 선택

선택하는 일치 출력 옵션에 따라 변환에서 출력 포트가 결정됩니다. 예를 들어 클러스터된 출력 유형을 선택하면 변환이 **ClusterId** 포트 및 **ClusterSize** 포트를 생성합니다.

필요한 변환 출력 유형을 선택하고 변환에서 포트를 검토합니다.

일치 출력 유형을 업데이트하는 경우, 업데이트한 후 변환에서 출력 포트 구성을 확인합니다. 매핑에서 변환을 사용하는 경우 출력 포트를 매핑의 다운스트림 개체에 다시 연결해야 할 수 있습니다.

## 일치 맵렛

일치 맵렛은 일치 변환에서 생성하고 포함하는 맵렛 유형입니다.

일치 맵렛을 생성하려면 일치 변환 구성을 일치 맵렛으로 저장합니다. 일치 맵렛을 생성할 때는 일치 변환 설정을 비교 및 가중치 평균 변환으로 변환합니다.

일치 맵렛을 생성한 후에는 변환을 추가하여 일치 분석을 사용자 지정할 수 있습니다. 예를 들어 식 변환을 추가해 두 전략의 링크 점수를 평가한 다음 최고 점수를 선택할 수 있습니다.

일치 변환과 달리 일치 맵렛은 수동이므로 **Analyst** 도구 내에서 규칙으로 사용할 수 있습니다. **Analyst** 도구에서 일치 맵렛을 사용하여 데이터 프로파일링 프로세스의 일부분으로 레코드 일치 여부를 확인합니다.

일치 변환은 일치 변환 내에서 생성하는 일치 맵렛만 읽을 수 있습니다.

## 일치 맵렛 작성

일치 맵렛을 작성하여 여러 변환을 사용하는 일치 분석 작업을 정의합니다.

1. 일치 변환을 편집기에서 열고 **전략** 보기를 선택합니다.
2. **일치 규칙 사용**을 선택합니다.
3. **이름 필드**에서 **새로 작성**을 선택합니다.  
새 맵렛 창이 열립니다.
4. 새 맵렛 창에서 맵렛의 이름을 입력하고 맵렛을 저장할 위치를 선택합니다.
5. 필요한 경우 **일치 변환의 전략 재사용**을 선택하여 현재 일치 변환의 입력, 전략 및 가중치를 일치 맵렛에 복사합니다.  
**참고:** 일치 변환에 현재 정의된 일치 기능을 복제하는 일치 맵렛을 빠르게 작성하려는 경우 이 설정을 사용하는 것이 좋습니다.
6. **마침**을 클릭합니다.  
편집기에서 일치 맵렛이 열립니다.
7. 필요한 경우 일치 맵렛에 비교 변환 및 가중치 평균 변환을 추가하고 구성하여 일치 작업을 작성합니다.
8. **파일 > 저장**을 클릭하여 맵렛을 저장합니다.
9. 맵렛을 닫고 일치 변환이 포함된 편집기를 선택합니다. 작성한 맵렛이 **이름 필드**에 표시되는지 확인합니다.
10. 필요한 경우 **일치 필드** 단추를 클릭하여 맵렛의 일치 필드를 구성합니다.  
일치 규칙 구성 창이 열립니다.
11. **입력 필드** 및 **사용 가능한 입력** 열의 필드를 두 번 클릭하여 입력 포트를 일치 입력에 할당합니다.
12. **파일 > 저장**을 클릭하여 변환을 저장합니다.

## 일치 맵렛 사용

일치 변환에서 이전에 정의한 일치 맵렛을 선택하고 구성할 수 있습니다.

1. 일치 변환을 편집기에서 열고 **전략** 보기를 선택합니다.
2. **일치 규칙 사용**을 선택합니다.
3. **이름 필드**에서 **기존 항목 사용**을 선택합니다.  
일치 규칙 구성 창이 열립니다.
4. **찾아보기**를 클릭하여 리포지토리에서 일치 맵렛을 찾습니다.  
**중요:** 일치 변환에서 작성한 맵렛만 선택할 수 있습니다.  
일치 맵렛 선택 창이 열립니다.
5. 일치 맵렛을 선택하고 **확인**을 클릭합니다.
6. **입력 필드** 및 **사용 가능한 입력** 열의 필드를 두 번 클릭하여 입력 포트를 일치 입력에 할당합니다.
7. **확인**을 클릭합니다.  
일치 규칙 구성 창이 닫힙니다.
8. **파일 > 저장**을 클릭하여 일치 변환을 저장합니다.

## 일치 분석 작업 구성

일치 작업을 구성하려면 소스 데이터를 일치 변환에 연결하고 변환 보기에서 속성을 편집하십시오.

1. 일치 변환을 작성하고 소스 데이터를 변환에 연결합니다.
2. **일치 유형** 보기를 선택하고 일치 유형을 선택합니다.
3. 선택한 일치 작업 유형에 대한 속성을 구성합니다.  
이중 소스 일치 유형을 선택한 경우 **마스터 데이터 집합** 속성을 구성합니다.
4. **전략** 보기를 선택하고 **일치 전략 정의**를 선택합니다.
5. **새로 만들기**를 클릭합니다.  
**새 일치 전략** 마법사가 열립니다.
6. 일치 전략을 선택하고 **다음**을 클릭합니다.
7. 필요에 따라 가중치 및 Null 일치 설정을 편집합니다. **다음**을 클릭합니다.
8. 사용 가능한 열에서 셀을 두 번 클릭하여 분석할 입력 포트를 선택합니다.  
**다음**을 클릭하여 다른 전략을 구성하거나 **마침**을 클릭하여 마법사를 종료합니다.  
**참고:** 전략 구성을 편집하려면 **전략** 보기에서 해당 전략의 셀에 있는 화살표를 클릭합니다.
9. **일치 출력** 보기를 선택합니다.  
일치 출력 유형을 선택하고 속성을 구성합니다.

**참고:** **전략** 보기에서 일치 맵셋을 선택 또는 작성하여 일치 전략을 구성할 수도 있습니다. 일치 맵셋은 일치 변환에 포함할 수 있는 맵셋 유형입니다.

## 일치 변환 - 비원시 환경

비원시 환경에서 일치 변환의 처리는 변환을 실행하는 엔진에 따라 다릅니다.

다음과 같은 비원시 런타임 엔진에 대한 지원을 고려하십시오.

- Blaze 엔진. 제한적으로 지원됩니다.
- Spark 엔진. 제한적으로 지원됩니다.
- Databricks Spark 엔진. 지원되지 않습니다.

### 일치 변환 - Blaze 엔진

ID 인덱스 데이터를 데이터베이스 테이블에 쓰도록 일치 변환이 구성된 경우 매핑 유효성 검사가 실패합니다.

일치 변환은 원시 환경과 비원시 환경에서 다른 방식으로 클러스터 ID 값을 생성합니다. 비원시 환경에서 변환은 그룹 ID 값을 클러스터 ID에 추가합니다.

### 일치 변환 - Spark 엔진

ID 인덱스 데이터를 데이터베이스 테이블에 쓰도록 일치 변환이 구성된 경우 매핑 유효성 검사가 실패합니다.

일치 변환은 원시 환경과 비원시 환경에서 다른 방식으로 클러스터 ID 값을 생성합니다. 비원시 환경에서 변환은 그룹 ID 값을 클러스터 ID에 추가합니다.

## 제 29 장

# 필드 분석의 일치 변환

이 장에 포함된 항목:

- [필드 일치 분석, 444](#)
- [필드 일치 분석을 위한 프로세스 흐름, 444](#)
- [필드 일치 유형 옵션, 445](#)
- [필드 일치 전략, 445](#)
- [필드 일치 출력 옵션, 448](#)
- [필드 일치의 고급 속성, 449](#)
- [필드 일치 분석 예제, 450](#)

## 필드 일치 분석

하나의 데이터 집합 또는 두 개의 데이터 집합 간에 유사하거나 중복 레코드를 찾으려면 필드 일치 분석을 수행하십시오.

필드 일치 분석을 위한 일치 변환을 구성한 경우 다음 보기에서 옵션을 설정하십시오.

- 일치 유형
- 전략
- 일치 출력

필요한 경우 매개 변수 및 고급 보기의 옵션을 설정합니다.

## 필드 일치 분석을 위한 프로세스 흐름

다음 프로세스 흐름은 필드 일치 분석을 위해 일치 변환을 구성할 때 사용하는 단계를 요약합니다. 일치 변환만 사용하는 프로세스를 정의할 수도 있고 일치 변환과 다른 변환을 사용하는 프로세스를 정의할 수도 있습니다.

**참고:** 일치 변환을 필드 일치 분석의 매핑에 추가하는 경우 업스트림 키 생성기 변환을 매핑에 추가합니다.

일치 변환을 위해 데이터를 준비하려면 다음 단계를 수행합니다.

1. 소스 데이터 레코드를 그룹으로 구성합니다.

키 생성기 변환을 사용하여 그룹 키 값을 각 레코드에 할당합니다. 그룹 할당은 일치 변환이 수행해야 하는 계산 수를 줄입니다.



2. 데이터 소스 레코드에 고유 시퀀스 식별자 값이 들어 있는지 확인합니다. 키 생성기 변환을 사용하여 값을 작성할 수 있습니다.

일치 변환에서 다음 단계를 수행합니다.

1. 일치 유형으로 필드 분석을 지정하고 데이터 소스 수를 지정합니다.  
두 개의 데이터 집합을 분석하도록 변환을 구성하는 경우 마스터 데이터 집합을 선택합니다.  
**일치 유형** 보기를 사용하여 유형 및 데이터 소스 수를 설정합니다.
2. 일치 분석 전략을 정의합니다. 알고리즘을 선택하고 한 쌍의 열을 알고리즘에 할당합니다.  
**전략** 보기를 사용하여 전략을 정의합니다.
3. 일치 분석 결과를 생성하기 위해 변환이 사용하는 방법을 지정합니다.
4. 일치 임계값을 설정합니다. 일치 임계값은 두 레코드를 서로에 대한 중복으로 식별할 수 있는 최소 점수입니다.

**일치 출력** 보기를 사용하여 출력 방법 및 일치 임계값을 선택합니다.

**참고:** 일치 변환 또는 가중치 평균 변환에서 일치 임계값을 설정할 수 있습니다. 일치 맵셋을 작성하는 경우 가중치 평균 변환을 사용합니다.

## 필드 일치 유형 옵션

일치 유형 보기는 이중 소스 일치 분석에 적용되는 단일 옵션을 포함합니다. 옵션에서 마스터 데이터 집합을 식별합니다. 일치 유형 보기는 단일 소스 일치 분석을 위한 옵션을 포함하지 않습니다.

두 데이터 집합을 분석하는 경우 하나를 마스터 데이터 집합으로 선택해야 합니다. 프로젝트 또는 조직에서 마스터 데이터 집합을 나타내는 데이터 집합이 없는 경우 더 큰 데이터 집합을 마스터 데이터 집합으로 선택합니다.

마스터 데이터 집합을 지정하려면 **마스터 데이터 집합** 옵션을 사용합니다.

## 필드 일치 전략

전략 보기에는 입력 데이터에 정의한 전략이 나열됩니다.

변환에서 데이터 소스 레코드 간의 유사점 및 차이를 측정하는 방법을 전략에서 결정합니다.

## 필드 일치 알고리즘

일치 변환에는 두 열에서 데이터 값을 비교하는 알고리즘이 포함되어 있습니다. 각 알고리즘은 다른 방법으로 데이터 값의 차이 정도를 계산합니다.

선택하는 열에서 찾으려고 하는 데이터 차이의 유형을 측정할 수 있는 알고리즘을 선택합니다.

### Bigram

단일 필드에 입력된 우편 주소처럼 긴 텍스트 문자열을 비교하려면 **Bigram** 알고리즘을 사용하십시오.

**Bigram** 알고리즘은 두 문자열의 연속 문자 항목을 기반으로 두 데이터 문자열의 일치 점수를 계산합니다. 알고리즘에서 두 문자열에 공통된 연속 문자 쌍을 찾습니다. 두 문자열에서 일치하는 쌍 개수를 전체 문자 쌍 개수로 나눕니다.

## Bigram 예제

다음 문자열을 고려하십시오.

- larder
- lerder

이 문자열에서 다음 **Bigram** 그룹이 생성됩니다.

```
l a, a r, r d, d e, e r
l e, e r, r d, d e, e r
```

"lerder" 문자열 내의 "e r" 문자열 두 번째 항목이 일치하지 않습니다. "larder" 문자열에서 "e r"의 해당하는 두 번째 항목이 없기 때문입니다.

**Bigram** 일치 점수를 계산하기 위해 변환에서 일치 쌍 개수(6)를 두 문자열의 전체 쌍 개수(10)로 나눕니다. 이 예에서 두 문자열은 60% 유사하며 일치 점수는 0.60입니다.

## Hamming 거리 측정법

**Hamming** 거리 측정법 알고리즘은 전화 번호, 우편 번호 또는 제품 코드 등의 숫자 또는 코드 필드와 같이 데이터 문자의 위치가 중요한 요인일 경우 사용합니다.

**Hamming** 거리 측정법 알고리즘은 두 데이터 문자열에서 문자가 다른 위치의 수를 계산하는 방법으로 일치 점수를 계산합니다. 문자열의 길이가 다를 경우 긴 문자열의 추가 문자는 다른 문자 수로 계산됩니다.

### Hamming 거리 측정법 예제

다음 문자열을 고려하십시오.

- Morlow
- Marlowes

강조 표시된 문자는 **Hamming** 알고리즘이 다른 문자로 식별하는 위치를 나타냅니다.

변환은 일치하는 문자의 수(5)를 가장 긴 문자열의 길이(8)로 나눠 **Hamming** 일치 점수를 계산합니다. 이 예에서 두 문자열은 62.5% 유사하며 일치 점수는 0.625입니다.

## 편집 거리 측정법

단어 또는 짧은 텍스트 문자열(예: 이름)을 비교하려면 편집 거리 측정법 알고리즘을 사용하십시오.

편집 거리 측정법 알고리즘에서 문자를 삽입, 삭제 또는 대체하여 하나의 문자열을 다른 문자열로 변환하는 최소 "비용"을 계산합니다.

### 편집 거리 측정법 예제

다음 문자열을 고려하십시오.

- Levenston
- Levenshtein

강조 표시된 문자는 문자열을 다른 문자열로 변환하는 데 필요한 작업을 표시합니다.

편집 거리 측정법 알고리즘에서 변경되지 않은 문자 수(8)를 가장 긴 문자열 길이(11)로 나눕니다. 이 예에서 두 문자열은 72.7% 유사하며 일치 점수는 0.727입니다.

## Jaro 거리 측정법

두 문자열을 비교할 때 비교 우선 순위가 첫 문자의 유사점인 경우 **Jaro** 거리 측정법 알고리즘을 사용합니다.

**Jaro** 거리 측정법의 일치 점수는 두 문자열의 첫 4개 문자가 갖는 유사점의 정도와 식별된 문자 전위의 수를 반영합니다. 첫 4개 문자 간의 일치에 대한 중요도는 페널티 속성에 입력한 값을 바탕으로 계산됩니다.

### Jaro 거리 측정법 속성

**Jaro** 거리 측정법 알고리즘을 구성할 때 다음 속성을 구성할 수 있습니다.

#### 페널티

두 문자열을 비교할 때 첫 4개 문자가 동일하지 않은 경우의 일치 점수 페널티를 결정합니다. 첫 번째 문자가 불일치하는 경우 변환이 전체 페널티 값을 뺍니다. 변환은 일치하지 않는 다른 문자의 위치에 따라 페널티의 일부를 뺍니다. 기본 페널티 값은 0.20입니다.

#### 대/소문자 구분

**Jaro** 거리 측정법 알고리즘으로 문자를 비교할 때 문자의 대/소문자를 고려할지 여부를 결정합니다.

### Jaro 거리 측정법 예제

다음 문자열을 고려하십시오.

- 391859
- 813995

기본 페널티 값인 0.20을 사용하여 이 문자열을 분석하는 경우 **Jaro** 거리 측정법 알고리즘은 0.513의 일치 점수를 반환합니다. 이 일치 점수는 해당 문자열이 51.3% 유사하다는 것을 나타냅니다.

## Hamming 거리 측정법 반전

**Hamming** 거리 측정법 반전 알고리즘을 사용하여 오른쪽에서 왼쪽으로 읽을 때 두 문자열 간의 문자 위치 차이 백분율을 계산합니다.

**Hamming** 거리 측정법 알고리즘은 두 데이터 문자열에서 문자가 다른 위치의 수를 계산하는 방법으로 일치 점수를 계산합니다. 길이가 다른 문자열의 경우 이 알고리즘은 가장 긴 문자열의 각 추가 문자를 문자열 간의 길이 차이로 계산합니다.

### Hamming 거리 측정법 반전 예제

왼쪽에서 오른쪽으로 쓰는 정렬을 사용하여 **Hamming** 측정법 반전 알고리즘을 시뮬레이트하는 다음 문자열을 살펴보세요.

- 1-999-9999
- 011-01-999-9991

선택된 문자는 **Hamming** 거리 측정법 반전 알고리즘이 차이로 식별하는 위치를 나타냅니다.

**Hamming** 측정법 반전 일치 점수를 계산하기 위해 변환은 일치하는 문자의 수(9)를 가장 긴 문자열의 길이(15)로 나눕니다. 이 예제에서는 일치 점수가 0.6이므로 문자열 유사도가 60%인 것입니다.

## 필드 일치 전략 속성

**전략** 보기에서 **전략** 마법사를 열고 각 필드 일치 전략에 대한 속성을 구성합니다.

필드 일치 전략을 구성하는 경우 다음 속성을 구성할 수 있습니다.

#### 이름

이름별로 전략을 식별합니다.

## 가중치

레코드의 전체 점수를 계산할 때 일치 점수에 할당된 상대 우선 순위를 결정합니다. 기본값은 0.5입니다.

## 단일 필드 Null

한 개의 값이 Null일 때 알고리즘이 데이터 값 쌍에 적용하는 일치 점수를 정의합니다. 기본값은 0.5입니다.

## 두 필드 모두 Null

두 값이 모두 Null일 때 알고리즘이 데이터 값 쌍에 적용하는 일치 점수를 정의합니다. 기본값은 0.5입니다.

**참고:** 일치하는 열 값 중 하나 또는 두 값이 모두 Null이면 일치 알고리즘이 일치 점수를 계산하지 않습니다. 알고리즘이 Null 일치 속성에서 정의된 점수를 적용합니다. Null 일치 속성을 지울 수 없습니다.

# 필드 일치 출력 옵션

**일치 출력** 옵션을 구성하여 필드 일치 분석에 대한 출력 형식을 정의할 수 있습니다.

이 옵션은 **일치 출력 유형** 영역 및 **속성** 영역에서 구성합니다.

## 일치 출력 유형

일치 출력 보기에는 출력 데이터 형식을 지정하는 옵션이 포함됩니다. 레코드를 클러스터 또는 일치하는 쌍으로 기록하도록 변환을 구성할 수 있습니다.

다음 일치 출력 유형 중 하나를 선택하십시오.

### 가장 잘 일치

두 번째 데이터 집합에서 가장 잘 일치 항목을 나타내는 레코드가 있는 마스터 데이터 집합에 각 레코드를 기록합니다. 일치 작업에서는 마스터 레코드에 대해 일치 점수가 가장 높은 두 번째 데이터 집합의 레코드를 선택합니다. 두 개 이상의 레코드가 가장 높은 점수를 반환할 경우 일치 작업에서 두 번째 데이터 집합의 첫 번째 레코드를 선택합니다. 가장 잘 일치에서는 각 레코드 쌍을 단일 행에 기록합니다.

이중 소스 분석용 변환을 구성한 경우 **가장 잘 일치**를 선택할 수 있습니다.

### 클러스터

일치 임계값을 충족하는 일치 점수를 가지고 서로 연결된 레코드 집합을 포함한 클러스터를 기록합니다. 각 레코드가 일치 임계값을 충족하는 점수를 가진 클러스터의 다른 레코드 하나 이상과 일치해야 합니다.

단일 소스 분석 및 이중 소스 분석에 대해 변환을 구성하는 경우 **클러스터**를 선택할 수 있습니다.

### 일치하는 쌍

일치 임계값을 충족하는 점수와 서로 일치하는 모든 레코드 쌍을 씁니다. 변환이 각 쌍을 단일 행에 쓰고 각 쌍에 대한 일치 점수를 각 행에 추가합니다. 레코드가 하나 이상의 다른 레코드와 일치하는 경우 변환이 각 레코드 쌍에 대해 행을 씁니다.

단일 소스 분석 및 이중 소스 분석에 대해 변환을 구성하는 경우 **일치하는 쌍**을 선택할 수 있습니다.

## 일치 출력 속성

일치 출력 보기에는 캐시 메모리 동작, 점수 일치 임계값, 변환 출력에 나타나는 점수 일치를 지정하는 속성이 포함되어 있습니다.

변환이 일치 점수 값을 출력 레코드에 추가하는 방식도 일치 출력 속성을 사용하여 지정할 수 있습니다.

일치 출력 유형을 선택한 후에 다음 속성을 구성하십시오.

## 캐시 디렉터리

데이터 통합 서비스에서 필드 일치 분석 중에 임시 데이터를 쓸 디렉터리를 지정합니다. 일치 분석이 생성하는 데이터 양이 사용 가능한 시스템 메모리보다 클 경우 데이터 통합 서비스가 이 디렉터리에 임시 파일을 기록합니다. 데이터 통합 서비스는 매핑을 실행한 후에 임시 파일을 삭제합니다.

속성에 디렉터리 경로를 입력하거나 매개 변수를 사용하여 디렉터리를 식별할 수 있습니다. 데이터 통합 서비스 시스템의 로컬 경로를 지정하십시오. 데이터 통합 서비스가 쓰기 권한이 있는 디렉터리를 지정해야 합니다. 기본값은 **CacheDir** 시스템 매개 변수입니다.

## 캐시 크기

데이터 통합 서비스에서 필드 일치 분석에 할당하는 시스템 메모리 양을 결정합니다. 기본값은 **400,000**바이트입니다.

데이터 통합 서비스는 데이터를 정렬하기 전에 사용자가 지정한 메모리 양을 할당합니다. 일치 분석이 더 많은 데이터를 생성할 경우 데이터 통합 서비스는 초과 데이터를 캐시 디렉터리에 기록합니다. 일치 분석이 시스템 메모리 및 파일 저장소가 제공할 수 있는 것보다 많은 메모리가 필요할 경우 매핑이 실패합니다.

**참고:** 65536 이상의 값을 입력한 경우 일치 변환이 값을 바이트로 읽습니다. 이보다 낮은 값을 입력하면 일치 변환이 값을 메가바이트로 읽습니다.

## 임계값

두 레코드를 서로의 잠재적 중복으로 식별하는 최소 일치 점수를 설정합니다.

매개 변수를 임계값에 할당할 수 있습니다. 10진수 값을 0에서 1 사이의 범위로 설정합니다.

## 평가 방법

변환 출력에 표시되는 일치 점수 값을 결정합니다. 클러스터 출력의 평가 방법을 선택합니다.

다음 테이블에는 평가 방법 옵션이 설명되어 있습니다.

평가 방법 옵션	설명
모두	링크 점수 및 드라이버 점수를 클러스터의 각 레코드에 추가합니다.
링크 점수	링크 점수를 클러스터의 각 레코드에 추가합니다. 기본값 옵션입니다.
드라이버 점수	드라이버 점수를 클러스터의 각 레코드에 추가합니다.
없음	점수 일치를 클러스터의 레코드에 추가하지 않습니다.

**참고:** 드라이버 점수를 레코드에 추가하면 매핑 실행 시간이 길어집니다. 매핑이 모든 클러스터가 완료된 후에 드라이버 점수 값을 레코드에 추가하기 때문입니다.

# 필드 일치의 고급 속성

변환에는 실행 인스턴스 수, 변환이 동일한 행을 분석하는 방법 및 로그 데이터의 추적 수준을 결정하는 고급 속성이 포함됩니다.

다음과 같은 고급 속성을 구성할 수 있습니다.

## 실행 인스턴스

변환에서 런타임 시 사용하는 스레드 수를 결정합니다.

일치 변환은 필드 일치 분석에서 단일의 실행 인스턴스를 사용합니다. 실행 인스턴스의 수는 일치 변환에 ID 일치 분석을 구성할 때 편집할 수 있습니다.

### 정확한 일치 항목 필터링

변환에서 일치 전략의 비교 알고리즘을 입력 데이터의 동일한 레코드 쌍에 적용할지 여부를 결정합니다.

변환이 동일한 레코드 쌍을 발견할 경우 알고리즘을 통해 레코드 간의 유사점 수준을 분석하지 않아도 됩니다. 변환에서 추가 분석 없이 레코드를 바로 출력 단계로 전달할 수 있습니다. 동일한 레코드를 바로 출력 단계로 전달하도록 변환을 구성하려면 정확한 일치 항목 필터링을 선택합니다. 입력 데이터에 다수의 동일 행이 포함될 경우 비교 알고리즘이 수행하는 계산 수가 줄어들므로 매핑이 더 빠르게 실행됩니다.

이 옵션은 입력 데이터에 다수의 동일 행이 포함될 경우에 선택합니다. 입력 데이터에 다수의 동일 행이 포함되지 않을 경우 이 옵션을 선택하면 변환이 더 느리게 실행될 수 있으니 선택하지 마십시오.

**참고:** 이 옵션을 선택하거나 선택 취소할 경우 변환 출력에 동일한 레코드 데이터가 포함됩니다. 이 옵션을 선택하고 선택 취소할 경우 변환이 출력 레코드에 서로 다른 링크 점수 및 드라이버 점수를 할당할 수 있습니다.

### 추적 수준

이 변환에 대해 로그에 표시되는 세부 정보의 양입니다. 간단, 보통, 자세한 정보 표시 초기화 또는 자세한 정보 표시 데이터를 선택할 수 있습니다. 기본값은 보통입니다.

## 필드 일치 분석 예제

한 은행의 데이터 스튜어드가 지난 7일 동안 은행 계좌를 개설한 고객에 대한 계정 레코드 집합을 받았습니다. 이 데이터 스튜어드는 데이터 집합에 중복 레코드가 포함되지 않았는지 확인하기 위해 레코드에서 중복 데이터를 검색하는 매핑을 설계합니다.

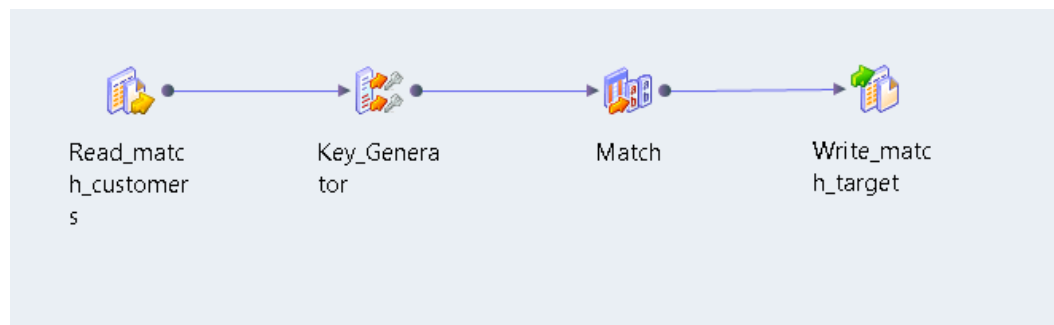
### 매핑 작성

여러 필드의 중복 데이터를 찾는 매핑을 작성합니다.

이 매핑은 다음 태스크를 수행합니다.

- 데이터 소스를 읽습니다.
- 소스 레코드에 그룹 키 값 및 시퀀스 식별자 값을 추가합니다.
- 레코드의 필드 데이터를 분석합니다.
- 결과를 데이터 대상에 기록합니다.

다음 이미지는 Developer tool의 매핑을 보여 줍니다.



작성하는 매핑에는 다음 개체가 포함됩니다.

개체 이름	설명
Read_Match_Customers	데이터 소스 이름, 주소 및 계좌 번호 등 예금주의 세부 정보를 포함합니다.
Key_Generator	키 생성기 변환입니다. 소스 데이터에 그룹 키 값 및 시퀀스 식별자 값을 추가합니다.
일치	일치 변환. 계좌 데이터의 중복 수준을 분석합니다.
Write_match_target	데이터 대상. 필드 분석의 결과를 포함합니다.

## 입력 데이터 샘플

데이터 집합에는 고객에 대한 계정 번호, 이름 주소 및 고용주가 포함되어 있습니다. 모델 리포지토리의 데이터 집합에서 데이터 소스를 작성하십시오. 데이터 소스를 매핑에 추가합니다.

다음 데이터 조각에서 고객 계정 데이터의 샘플을 보여 줍니다.

CustomerID	Lastname	도시	상태	우편 번호
15954467	JONES	SCARSDALE	NY	10583
10110907	JONES	MINNEAPOLIS	MN	55437
19131127	JONES	INDIANAPOLIS	IN	46240
10112097	JONES	HOUSTON	TX	77036
19133807	JONES	PLANTATION	FL	33324
10112447	JONES	SCARSDALE	NY	10583
15952487	JONES	HOUSTON	TX	77002
10112027	JONES	OAKLAND	CA	94623

## 키 생성기 변환 구성

키 생성기 변환을 구성할 때는 분석할 데이터 소스 포트를 연결해야 합니다. 그룹 키 데이터가 포함된 포트를 지정합니다. 레코드에 고유한 식별자가 포함되지 않는 경우 시퀀스 ID 포트를 사용하여 고유한 식별자를 레코드에 추가합니다.

그룹 키 포트를 지정할 때는 다음 지침을 고려하십시오.

- 포트 데이터에서 정기적으로 반복되는 값이 포함된 포트를 선택합니다. 가능한 경우 크기가 비슷한 그룹을 작성하는 포트를 선택합니다.
- 중복 분석과 관련이 없는 포트를 선택합니다.

현재 예에서는 도시 포트를 그룹 키로 선택합니다. 계정 이름이 도시에서 두 번 이상 나타나는 계정에는 중복 데이터가 포함될 수 있습니다. 계정 이름이 서로 다른 도시에서 두 번 이상 나타나는 계정은 중복이 될 가능성이 낮습니다.

**Tip:** 그룹 키 포트를 선택하기 전에 데이터 소스에 열 프로필을 실행하십시오. 프로필 결과를 바탕으로 각 값이 포트에 나타나는 횟수를 확인할 수 있습니다.

## 일치 변환 구성

재사용할 수 없는 일치 변환을 매핑에 추가하여 필드 분석을 수행합니다.

다음 태스크를 완료하여 일치 변환을 구성하십시오.

1. 수행할 일치 분석의 유형을 선택합니다.
2. 입력 포트를 변환에 연결합니다.
3. 레코드 데이터를 비교하는 전략을 구성합니다.
4. 변환에서 작성할 일치 출력 데이터의 유형을 선택합니다.
5. 출력 포트를 데이터 대상에 연결합니다.

### 일치 작업의 유형 선택

**일치 유형** 보기의 옵션을 사용하여 일치 작업을 선택합니다. 계정 데이터를 비교하려면 단일 소스 필드 분석을 수행하도록 변환을 구성하십시오.

### 입력 포트 연결

고객 계정 데이터 포트를 일치 변환에 연결합니다.

일치 변환은 미리 설정된 입력 포트를 사용하여 레코드 처리 순서를 결정합니다. 이 변환은 시퀀스 식별자를 사용하여 입력 포트부터 출력으로 기록하는 일치하는 쌍 또는 클러스터까지의 레코드를 추적합니다. 일치 변환이 처리하는 레코드는 그룹 키를 사용하여 정렬됩니다.

다음과 같은 일치 변환의 미리 설정된 포트를 키 생성기 변환의 미리 설정된 출력 포트에 연결합니다.

- SequenceID
- GroupKey

### 필드 분석 전략 구성

전략을 구성하려면 **전략** 보기의 옵션을 사용하십시오. 변환이 레코드 데이터에 대해 수행하는 분석 유형을 전략에서 결정합니다.

다음 전략을 작성하십시오.

- 고객 ID 번호를 분석하려면 편집 거리 측정법 알고리즘을 사용하는 전략을 작성합니다. CustomerID\_1 및 CustomerID\_2 포트를 선택합니다.
- 성 데이터를 분석하려면 Jaro 거리 측정법 알고리즘을 사용하는 전략을 작성합니다. Lastname\_1 및 Lastname\_2 포트를 선택합니다.

**참고:** Jaro 거리 측정법 알고리즘에서는 다른 문자로 시작하는 유사한 문자열에 추가 페널티를 적용합니다. 따라서 Jaro 거리 측정법 알고리즘은 높은 일치 점수를 PATTON 및 PATTEN에 적용할 수 있지만 BAYLOR 및 TAYLOR에는 더 낮은 일치 점수를 적용합니다.

- 우편 번호 데이터를 분석하려면 Hamming 거리 측정법 반전 알고리즘을 사용하는 전략을 작성합니다. Zip\_1 및 Zip\_2 포트를 선택합니다.



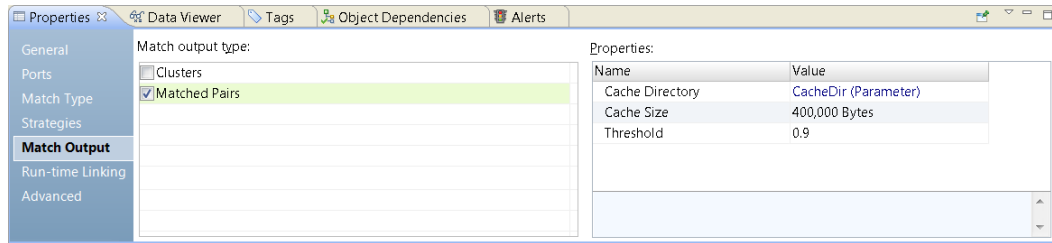
## 관련 항목:

- [“필드 일치 알고리즘” 페이지 445](#)

## 일치 출력 유형 선택

**일치 출력** 보기의 옵션을 사용하여 일치 분석의 결과에 대한 출력 형식을 정의합니다.

다음 이미지는 단일 소스 필드 분석에 대한 일치 출력 보기를 보여 줍니다.



출력 레코드가 일치하는 쌍으로 구성되도록 변환을 구성하십시오. 변환이 레코드를 쌍으로 반환하므로 분석 결과에는 고유한 레코드가 포함되지 않습니다.

## 출력 포트 연결

일치 변환의 출력 포트를 매핑의 데이터 대상에 연결합니다. 데이터 대상에 쓸 레코드 데이터가 포함된 포트를 선택하십시오.

변환에는 각 일치 쌍의 레코드를 식별하는 사전 설정된 포트가 포함되어 있습니다. 사전 설정된 포트 이름은 **RowId** 및 **RowId1**입니다. 각 행 ID 값은 출력 데이터의 레코드를 식별합니다.

행 ID 값은 일치 전략에서 선택한 포트에 해당합니다. 전략을 구성할 경우 접미사 **\_1** 또는 **\_2**가 포함된 포트 이름을 선택합니다. **RowId** 값은 **\_1** 접미사가 있는 포트가 포함된 레코드를 식별합니다. **RowId1** 값은 **\_2** 접미사가 있는 포트가 포함된 레코드를 식별합니다.

다른 출력 포트를 사용하여 레코드 간 관계를 검토할 수 있습니다. 링크 포트 값 및 드라이버 포트 값은 각 클러스터에 있는 레코드 간의 유사 정도를 나타냅니다.

현재 예에서는 모든 포트를 데이터 대상에 연결합니다. 포트의 출력 데이터를 보려면 데이터 뷰어를 실행합니다.

## 데이터 뷰어 실행

데이터 뷰어를 실행하여 일치 분석의 결과를 검토합니다. 기본적으로 데이터 뷰어에는 일치 변환의 모든 출력 포트가 표시됩니다. 매핑을 실행하면 데이터 대상이 출력 포트의 데이터로 업데이트됩니다.

다음 이미지는 데이터 뷰어의 출력 데이터를 보여 줍니다.

The screenshot shows a workflow in the Field Match tool with four components: Read\_match\_customer\_s\_13, Key\_Generator, Match, and Write\_match\_target. Below the workflow is the Data Viewer tab, which displays a table of matched records. The table has columns for various fields including Lastname, Firstname, Company, Address, City, State, Zip, and Customer ID. The first four rows of data are shown, with Row 1 to 4 indicated at the bottom.

	Lastname_1	Firstname_1	Company_1	Address1_1	City_1	State_1	Zip_1	Customer1...	Lastname_2	Firstname_2	Company_2	Address1_2	City_2
1	Chan	David	SANDS BR...	515 FOLS...	SAN FRAN...	CA	94105	15954522	Chan	Roy	PG&E (PA...	77 BEALE...	SAN FRAN...
2	Andersen	Keith	CHARLES...	1100 N.E....	SAN ANT...	TX	78209	10110428	Andersen	Keith	CHARLES...	1100 N.E....	SAN ANT...
3	Abrary	David	AIRFINAN...	488 MADL...	NEW YORK	NY	10022	15954929	Abrary	Julianne	TRANSOC...	39 EAST 5...	NEW YORK
4	Chan	Agusta	J. STREICH...	86 TRINIT...	NEW YORK	NY	10006	19132518	Chan	Carmen	AMERICA...	86 TRINIT...	NEW YORK

데이터 뷰어는 고객 계정 데이터에 하나 이상의 중복 레코드가 포함되는지 확인합니다.

데이터 뷰어에서 다음 데이터를 고려하십시오.

- 일치 변환은 **Augusta Chan**과 **Carmen Chan**의 성과 주소 데이터가 같으므로 두 레코드가 동일한 정보를 포함할 수 있다고 판단합니다. 이 레코드를 검토한 데이터 스튜어드는 레코드가 데이터 집합에서 고유하다고 결정합니다. 그러나 레코드가 공통된 고객 ID 값을 공유한다는 사실을 알게 됩니다. 이 데이터 집합의 기본 키가 고객 ID 였으므로 데이터 스튜어드는 뉴욕 사무실에 연락합니다. 뉴욕 사무실에서는 이 오류를 해결합니다.
- 일치 변환은 **Keith Anderson**에 대한 레코드가 동일한 정보를 포함할 수 있다고 판단합니다. 레코드를 검토한 데이터 스튜어드는 두 레코드가 동일한 계정을 나타내는 것을 확인합니다. 그러나 해당 레코드의 고객 ID 값이 다르다는 것을 알게 됩니다. 한 고객 계정에는 단일의 ID 값이 있어야 하므로 데이터 스튜어드는 샌안토니오 사무실에 연락합니다. 샌안토니오 사무실에서는 이 오류를 해결합니다.

## 결론

이 일치 분석의 결과를 보면 고객 계정 데이터 집합에 최소 1쌍 이상의 중복 레코드가 포함되는 것을 알 수 있습니다. 데이터 스튜어드는 확인이 필요한 계정에 대한 내용을 해당 지역의 지점에 문의하고 데이터 집합의 다른 레코드가 고유하게 고객 계정을 식별하는지 확인합니다.

## 제 30 장

# ID 분석의 일치 변환

이 장에 포함된 항목:

- [ID 일치 분석, 455](#)
- [ID 일치 분석의 프로세스 흐름, 456](#)
- [ID 일치 유형 속성, 456](#)
- [ID 일치 전략, 460](#)
- [ID 일치 출력 옵션, 462](#)
- [ID 일치의 고급 속성, 465](#)
- [지속형 인덱스 사례 연구, 466](#)
- [ID 일치 분석 예제, 468](#)

## ID 일치 분석

ID 일치 분석을 수행하면 단일 데이터 세트 또는 데이터 세트 2개에서 유사하거나 중복된 ID를 찾을 수 있습니다.

유사한 ID가 2개 이상의 레코드에 존재하는 것은 레코드 중복을 나타낼 수 있습니다. 유사한 ID는 공유 가족 ID 또는 공유 직원 ID와 같은 레코드 간의 연결을 나타낼 수도 있습니다.

ID 일치 분석에 대해 일치 변환을 구성하는 경우 다음 보기에서 옵션을 설정해야 합니다.

- 일치 유형
- 전략
- 일치 출력

필요한 경우 매개 변수 및 고급 보기의 옵션을 설정합니다.

ID 일치 분석에 대해 일치 변환을 구성하려면 입력 포트를 ID 전략의 모든 기본 필수 필드에 연결해야 합니다. 대부분의 ID 전략에는 기본 필수 필드가 포함됩니다. 일부 전략에는 보조 필수 필드도 포함됩니다. 최소 1개 이상의 보조 필수 필드에 입력 포트를 연결해야 합니다.

## ID 일치 분석의 프로세스 흐름

다음 프로세스 흐름에는 ID 일치 분석을 실행하도록 일치 변환을 구성하기 위해 수행하는 단계가 요약되어 있습니다. 일치 변환만 사용하는 프로세스를 정의할 수도 있고 일치 변환과 다른 변환을 사용하는 프로세스를 정의할 수도 있습니다.

일치 변환을 업스트림 데이터 개체에 연결하기 전에 레코드가 고유한 시퀀스 식별자 값을 포함하는지 확인합니다. 키 생성기 변환을 사용하여 값을 작성할 수 있습니다. ID 일치 분석을 수행할 때는 필요에 따라 입력 데이터를 그룹으로 구성할 수 있습니다.

일치 변환에서 다음 단계를 수행합니다.

1. ID 분석을 일치 유형으로 지정하고 데이터 소스 수를 지정합니다.

두 개의 데이터 집합을 분석하도록 변환을 구성하는 경우 마스터 데이터 집합을 선택합니다.

**일치 유형** 보기를 사용하여 유형 및 데이터 소스 수를 설정합니다.

2. 인덱스 데이터를 저장할 위치를 식별합니다. 변환은 인덱스 데이터를 임시 파일에 쓰거나 데이터베이스 테이블에 저장할 수 있습니다.

**일치 유형** 보기를 사용하여 인덱스 데이터 저장소를 지정합니다.

3. 일치 분석 전략을 정의합니다. 인구집단과 비교 알고리즘을 선택하고 한 쌍의 열을 알고리즘에 할당합니다. 인구집단은 선택할 열 쌍을 나타냅니다.

**전략** 보기를 사용하여 전략을 정의합니다.

4. 일치 분석 결과를 생성하기 위해 변환이 사용하는 방법을 지정합니다.

5. 일치 임계값을 설정합니다. 일치 임계값은 두 레코드를 서로에 대한 중복으로 식별할 수 있는 최소 점수입니다.

**일치 출력** 보기를 사용하여 출력 방법 및 일치 임계값을 선택합니다.

**참고:** 일치 변환 또는 가중치 평균 변환에서 일치 임계값을 설정할 수 있습니다. 일치 맵셋을 작성하는 경우 가중치 평균 변환을 사용합니다.

## ID 일치 유형 속성

일치 유형 보기를 사용하여 일치 변환이 수행하는 분석 유형을 지정하고 분석을 정의하는 속성을 설정합니다. 단일 소스 분석 또는 이중 소스 분석을 지정할 수 있습니다. ID 인덱스 데이터에 대해 지속형 데이터 저장소를 지정할 수도 있습니다.

구성할 속성은 선택한 분석 유형에 따라 달라집니다. 여러 옵션이 모든 분석 유형에 대해 공통으로 적용됩니다.

### 공통 속성

다음 속성은 모든 ID 분석 유형에 공통으로 적용됩니다.

#### 인구집단

변환이 사용하는 인구집단 파일을 식별합니다. 인구집단 파일에는 인덱스 키를 생성하는 키 작성 알고리즘이 포함되어 있습니다.

#### 키 수준

ID 알고리즘이 생성하는 키 수를 결정합니다. 기본 설정은 표준입니다. 제한 설정을 선택하면 적은 수의 키가 생성되고 정확성은 개선되지만 처리 시간이 길어집니다. 확장 설정을 선택하면 많은 수의 키가 생성되고 정확성은 떨어지지만 처리 시간이 짧아집니다.

## 키 유형

키 필드가 포함하는 정보의 유형을 설명합니다. ID 분석은 사람 이름, 조직 및 주소에 대한 키를 생성할 수 있습니다. **키 필드** 속성에서 지정한 열을 가장 잘 설명하는 키 유형을 선택하십시오.

## 검색 수준

변환이 일치 분석에 적용하는 검색 깊이 및 검색 속도 간의 균형을 식별합니다. 검색 깊이는 반환되는 일치 수에 반비례합니다. 예를 들어 **포괄적** 옵션은 적은 수의 일치를 반환합니다.

## 키 필드

일치 변환이 인덱스 키 데이터를 생성하는 데 사용하는 열을 지정합니다. 선택한 열에 **키 유형** 속성에서 지정한 정보 유형이 포함되는지 확인하십시오.

## 인덱스 디렉터리

데이터 통합 서비스가 현재 변환에 대한 인덱스 키 데이터를 쓸 디렉터리를 지정합니다. 이 속성은 기본적으로 비어 있습니다. 인덱스 디렉터리를 지정하지 않으면 데이터 통합 서비스는 사용자가 콘텐츠 관리 서비스에서 지정한 위치를 사용합니다.

디렉터리 경로를 지정하거나 매개 변수를 사용하여 디렉터리를 식별할 수 있습니다. 데이터 통합 서비스 시스템의 로컬 경로를 지정하십시오. 데이터 통합 서비스가 쓰기 권한이 있는 디렉터리를 지정해야 합니다.

## 캐시 디렉터리

데이터 통합 서비스가 ID 일치 분석의 인덱스 작성 단계 중에 임시 데이터를 쓸 디렉터리를 식별합니다. 속성을 업데이트하여 현재 변환의 데이터에 대해 위치를 지정하십시오. 이 속성은 기본적으로 비어 있습니다. 캐시 디렉터리를 지정하지 않으면 데이터 통합 서비스는 사용자가 콘텐츠 관리 서비스에서 지정한 위치를 사용합니다.

디렉터리 경로를 지정하거나 매개 변수를 사용하여 디렉터리를 식별할 수 있습니다. 데이터 통합 서비스 시스템의 로컬 경로를 지정하십시오. 데이터 통합 서비스가 쓰기 권한이 있는 디렉터리를 지정해야 합니다.

## 캐시 크기

데이터 통합 서비스에서 ID 인덱스 작성에 할당하는 시스템 메모리 양을 결정합니다. 기본값은 400,000바이트입니다.

인덱스 작성 작업이 많은 양의 데이터를 생성할 경우 데이터 통합 서비스는 초과 데이터를 캐시 디렉터리에 씁니다. 작업에 시스템 메모리 및 파일 저장소가 제공할 수 있는 것보다 더 많은 메모리가 필요할 경우 매핑이 실패합니다.

**참고:** 65536 이상의 값을 입력한 경우 일치 변환이 값을 바이트로 읽습니다. 이보다 낮은 값을 입력하면 일치 변환이 값을 메가바이트로 읽습니다.

## 이중 소스 속성

이중 소스 분석에 대해 변환을 구성할 때 공통 속성과 더불어 다음 속성도 설정하십시오.

## 마스터 데이터 집합

마스터 데이터가 포함된 데이터 소스를 식별합니다. 마스터 데이터 집합은 이중 소스 분석에서 지정합니다.

## 지속형 데이터 스토리지 속성

지속형 인덱스 데이터 저장소를 사용하도록 변환을 구성할 때 공통 속성과 더불어 다음 속성도 설정하십시오.

## 지속성 방법

변환이 현재 인덱스 테이블을 매핑 데이터 소스의 인덱스 데이터로 업데이트할지 여부를 지정합니다. 다음 옵션 중 하나를 선택합니다.

- 데이터베이스를 새 ID로 업데이트.  
변환은 인덱스 데이터 중에서 시퀀스 식별자가 중복되지 않는 인덱스 데이터에 모든 행을 추가합니다. 변환은 인덱스의 현재 행을 업데이트하지 않습니다.  
기본적으로 이 옵션을 선택하면 변환은 일치 분석을 수행합니다. 일치 프로세스 옵션을 사용하여 일치 분석을 활성화하거나 비활성화할 수 있습니다.
- 데이터베이스를 업데이트하지 않음.  
변환이 매핑 데이터 소스의 인덱스 데이터로 인덱스 테이블을 업데이트하지 않습니다.  
이 옵션을 선택하면 변환은 일치 분석을 수행합니다.
- 데이터베이스에서 ID를 제거합니다.  
행이 시퀀스 식별자를 매핑 소스 데이터와 공유하는 경우 변환은 인덱스 테이블에서 행을 삭제합니다. 사용자가 이 옵션을 선택하는 경우 변환은 일치 분석을 수행하지 않습니다.
- 데이터베이스에서 현재 ID를 업데이트합니다.  
행이 시퀀스 식별자를 공유하는 경우 변환은 인덱스 테이블의 행을 매핑 소스 데이터의 행으로 바꿉니다. 변환은 인덱스에 행을 추가하지 않습니다.  
기본적으로 이 옵션을 선택하면 변환은 일치 분석을 수행합니다. 일치 프로세스 옵션을 사용하여 일치 분석을 활성화하거나 비활성화할 수 있습니다.

기본 지속성 방법은 **데이터베이스를 새 ID로 업데이트**입니다.

## 일치 프로세스

현재 변환이 ID 분석을 수행하는지 여부를 결정합니다.

지속성 방법 속성에서 선택한 옵션에 따라 일치 프로세스 속성의 옵션이 결정됩니다.

## DB 연결

인덱스 테이블을 포함하는 데이터베이스를 식별합니다.

## 지속형 저장소

지정한 데이터베이스 안의 인덱스 테이블을 식별합니다.

## 관련 항목:

- [“지속형 인덱스 사례 연구” 페이지 466](#)
- [“지속성 방법 매개 변수” 페이지 459](#)

# 인덱스 디렉터리 및 캐시 디렉터리 속성

인덱스 디렉터리와 캐시 디렉터리는 ID 분석 중에 일치 변환에서 생성하는 임시 데이터를 저장합니다.

인덱스 데이터를 데이터베이스 테이블에 쓰도록 변환을 구성하지 않는 경우 데이터 통합 서비스는 인덱스 디렉터리에 데이터를 씁니다. 캐시 크기가 지정하는 양보다 더 많은 메모리 양이 ID 분석에 필요한 경우 데이터 통합 서비스는 캐시 디렉터리에 데이터를 씁니다. 기본적으로 속성은 일치 유형 보기에서 비어 있습니다. 인덱스 디렉터리나 캐시 디렉터리를 지정하지 않는 경우 데이터 통합 서비스는 콘텐츠 관리 서비스에서 디렉터리 경로를 읽습니다.

일치 유형 보기에서 인덱스 디렉터리나 캐시 디렉터리를 지정하는 경우 데이터 통합 서비스 시스템에 로컬 경로를 입력합니다. 정규화된 경로 또는 상대 경로를 입력할 수 있습니다. 상대 경로를 입력할 경우 마침표로 경로를 시작하십시오. 경로는 데이터 통합 서비스 시스템의 **tomcat/bin** 디렉터리에 상대적입니다.

다음 테이블은 캐시 디렉터리 또는 인덱스 디렉터리 속성의 상대 경로를 보여 주며 속성이 나타내는 정규화된 경로를 식별합니다.

상대 경로	정규화된 경로
./ch	[Informatica_installation_directory]/tomcat/bin/ch

인덱스 디렉터리 속성과 캐시 디렉터리 속성을 구성하여 동일한 디렉터리를 식별할 수 있습니다. 각 속성에서 동일한 디렉터리를 지정하는 경우 데이터 통합 서비스는 사용자가 지정한 디렉터리에 디렉터리를 작성합니다. 데이터 통합 서비스는 인덱스 데이터에 대해 인덱스 디렉터리를, 캐시 데이터에 대해 캐시 디렉터리를 작성합니다. 각 속성에서 서로 다른 디렉터리를 지정하는 경우 데이터 통합 서비스는 사용자가 지정한 디렉터리에 데이터를 씁니다.

데이터 통합 서비스는 매핑 실행 후 디렉터리에서 인덱스 파일과 캐시 파일을 삭제합니다. 디렉터리는 삭제하지 않습니다. 일치 변환이 디렉터리를 식별하는 경우 데이터 통합 서비스가 다른 매핑의 디렉터리를 다시 사용할 수 있습니다.

### 콘텐츠 관리 서비스 속성

콘텐츠 관리 서비스는 인덱스 디렉터리와 캐시 디렉터리에 대해 다음 기본 위치를 지정합니다.

- ./identityIndex. ID 인덱스 데이터의 기본 디렉터리입니다.
- ./identityCache. ID 캐시 데이터의 기본 디렉터리입니다.

일치 변환에서 속성을 설정하지 않으면 ID 일치 매핑을 실행할 때 데이터 통합 서비스가 기본 디렉터리를 작성합니다. 데이터 통합 서비스는 tomcat/bin 디렉터리에 디렉터리를 작성합니다.

## 지속성 방법 매개 변수

ID 일치 분석에서 지속성 데이터 인덱스를 선택하면 매개 변수를 사용하여 지속성 방법을 식별할 수 있습니다. 문자열을 사용하여 매개 변수 값을 정의합니다.

다음 표에는 **일치 유형** 탭에서 선택할 수 있는 지속성 방법 및 해당 방법에 대해 설정할 수 있는 매개 변수 값이 나열되어 있습니다.

지속성 방법	매개 변수
데이터베이스를 새 ID로 업데이트	ignore
데이터베이스를 업데이트하지 않음	addNone
데이터베이스에서 ID 제거	remove
데이터베이스에서 현재 ID 업데이트	update

매개 변수 값은 대/소문자를 구분합니다.

관련 항목:

- [“ID 일치 유형 속성” 페이지 456](#)

## ID 일치 전략

전략 보기는 ID 데이터에 대해 정의한 전략을 나열합니다. 이 전략은 일치 변환이 ID 인덱스 데이터 간의 유사성 및 차이점을 측정하는 방식을 결정합니다.

### ID 일치 알고리즘

일치 변환에 ID 인덱스의 데이터 값을 비교하는 사전 정의된 ID 알고리즘이 포함됩니다. 데이터 집합에서 ID 데이터 유형을 가장 잘 나타내는 알고리즘을 선택합니다.

다음 테이블은 알고리즘을 설명하고 각 알고리즘에 대해 선택하는 입력을 식별합니다.

ID 알고리즘	설명
주소	주소를 공유하는 레코드를 식별합니다. 알고리즘에 다음 기본 입력이 필요합니다. <ul style="list-style-type: none"><li>- 주소</li></ul> 알고리즘에 보조 입력이 필요하지 않습니다.
연락처	단일 조직 위치에 있는 연락처를 공유하는 레코드를 식별합니다. 알고리즘에 다음 기본 입력이 필요합니다. <ul style="list-style-type: none"><li>- Person_Name</li><li>- Organization_Name</li><li>- Address_Part1</li></ul> 알고리즘에 보조 입력이 필요하지 않습니다.
회사 항목	통합 ID 데이터를 공유하는 레코드를 식별합니다. 필요에 따라 이 알고리즘을 선택하고 주소 및 전화 데이터를 분석합니다. 알고리즘에 다음 기본 입력이 필요합니다. <ul style="list-style-type: none"><li>- Organization_Name</li></ul> 알고리즘에 보조 입력이 필요하지 않습니다.
나누기	조직 내의 사무실 위치를 공유하는 레코드를 식별합니다. 알고리즘에 다음 기본 입력이 필요합니다. <ul style="list-style-type: none"><li>- Organization_Name</li><li>- Address_Part1</li></ul> 알고리즘에 보조 입력이 필요하지 않습니다.
가족	가족에 속하는 개인을 식별합니다. 이름, 주소 및 전화 번호 데이터를 분석합니다. 알고리즘에 다음 기본 입력이 필요합니다. <ul style="list-style-type: none"><li>- Person_Name</li></ul> 알고리즘에 다음 보조 입력 중 하나가 필요합니다. <ul style="list-style-type: none"><li>- Address_Part1</li><li>- Telephone_Number</li></ul>



ID 알고리즘	설명
필드	선택하는 포트에서 데이터를 공유하는 레코드를 식별합니다. 알고리즘이 필요한 입력을 지정하지 않습니다. 중복 ID 데이터를 포함할 수 있는 포트를 선택합니다.
가정	가정에 속하는 개인을 식별합니다. 이름 데이터 및 주소 데이터를 분석합니다. 알고리즘에 다음 기본 입력이 필요합니다. - Person_Name - Address_Part1
개인	중복된 개인을 식별합니다. 이름, 생년월일와 사회 보장 번호, 계정 번호 및 차량 ID 번호와 같은 개인 ID 데이터를 분석합니다. 알고리즘에 다음 기본 입력이 필요합니다. - Person_Name 알고리즘에 다음 보조 입력 중 하나가 필요합니다. - 날짜 - ID
조직	조직 데이터를 공유하는 레코드를 식별합니다. 알고리즘에 다음 기본 입력이 필요합니다. - Organization_Name 알고리즘에 보조 입력이 필요하지 않습니다.
개인 이름	개인에 대한 정보를 공유하는 레코드를 식별합니다. 알고리즘에 다음 기본 입력이 필요합니다. - Person_Name 알고리즘에 보조 입력이 필요하지 않습니다.
거주자	주소에서 중복된 개인을 식별합니다. 필요에 따라 개인 ID 데이터를 분석하도록 이 전략을 구성합니다. 알고리즘에 다음 기본 입력이 필요합니다. - Person_Name - Address_Part1 알고리즘에 보조 입력이 필요하지 않습니다.
광범위한 연락처	조직에 있는 연락처를 공유하는 레코드를 식별합니다. 알고리즘에 다음 기본 입력이 필요합니다. - Person_Name - Organization_Name 알고리즘에 보조 입력이 필요하지 않습니다.
광범위한 가정	동일한 가정에 속하는 개인을 식별합니다. 알고리즘에 다음 기본 입력이 필요합니다. - Address_Part1 알고리즘에 보조 입력이 필요하지 않습니다.

## ID 일치 전략 속성

각 ID 전략에 대해 속성을 구성합니다.

ID 전략을 구성할 때는 다음 전략 속성을 구성할 수 있습니다.

## 인구집단

ID 분석에 적용할 인구집단을 결정합니다. 인구집단에는 특정 로캘과 언어에 대한 키 빌드 알고리즘이 포함됩니다.

## 일치 수준

검색 품질과 검색 속도의 균형을 결정합니다. 반환되는 일치 항목의 수가 많을수록 검색 속도는 느려집니다. 느슨한 설정을 사용하는 검색에서는 일치 항목이 더 적게 반환되고 보수적 설정을 사용하는 검색에서는 일치 항목이 더 많이 반환됩니다.

# ID 일치 출력 옵션

일치 출력 보기에는 출력 데이터 형식을 지정하는 옵션이 포함됩니다. 레코드를 클러스터 또는 일치하는 쌍으로 기록하도록 변환을 구성할 수 있습니다. 또한 지속형 인덱스 데이터 저장소에 대해 ID 분석을 수행할 때 다른 범주의 ID를 포함하거나 제외하도록 구성할 수도 있습니다.

이 옵션은 **일치 출력 유형** 영역 및 **속성** 영역에서 구성합니다.

## 일치 출력 유형

일치 출력 보기에는 출력 데이터 형식을 지정하는 옵션이 포함됩니다. 레코드를 클러스터 또는 일치하는 쌍으로 기록하도록 변환을 구성할 수 있습니다.

다음 일치 출력 유형 중 하나를 선택하십시오.

### 가장 잘 일치

두 번째 데이터 집합에서 가장 잘 일치 항목을 나타내는 레코드가 있는 마스터 데이터 집합에 각 레코드를 기록합니다. 일치 작업에서는 마스터 레코드에 대해 일치 점수가 가장 높은 두 번째 데이터 집합의 레코드를 선택합니다. 두 개 이상의 레코드가 가장 높은 점수를 반환할 경우 일치 작업에서 두 번째 데이터 집합의 첫 번째 레코드를 선택합니다. 가장 잘 일치에서는 각 레코드 쌍을 단일 행에 기록합니다.

이중 소스 분석용 변환을 구성한 경우 **가장 잘 일치**를 선택할 수 있습니다.

### 클러스터 - 가장 잘 일치

동일한 데이터 집합 또는 두 데이터 집합 간 한 레코드와 다른 레코드 간 가장 잘 일치를 나타내는 클러스터를 씁니다. 두 레코드 간의 일치 점수가 일치 임계값을 충족해야 합니다. 레코드가 하나 이상의 다른 레코드와 가장 잘 일치를 나타내는 경우 가장 잘 일치 클러스터에 두 개 이상의 레코드가 포함될 수 있습니다.

ID 분석 유형에서 **클러스터 - 가장 잘 일치**를 선택할 수 있습니다.

**참고:** 사용자가 선택한 인덱스 데이터 저장 방법은 **클러스터 - 가장 잘 일치** 모드에서 클러스터 출력 콘텐츠에 영향을 미칠 수 있습니다. 인덱스 테이블에 연결되는 변환은 동일한 레코드의 인덱스 데이터를 임시 파일에 저장하는 변환과 다른 클러스터를 작성할 수 있습니다. 인덱스 데이터 저장 방법은 변환이 레코드 쌍에 대해 생성하는 일치 점수에 영향을 미치지 않습니다.

### 클러스터 - 모두 일치

일치 임계값을 충족하는 점수와 일치하는 레코드 클러스터를 씁니다. 각 레코드는 클러스터에서 하나 이상의 다른 레코드와 일치해야 합니다.

ID 분석 유형에서 **클러스터 - 모두 일치**를 선택할 수 있습니다.

## 일치하는 쌍

일치 임계값을 충족하는 점수와 서로 일치하는 모든 레코드 쌍을 씁니다. 변환이 각 쌍을 단일 행에 쓰고 각 쌍에 대한 일치 점수를 각 행에 추가합니다. 레코드가 하나 이상의 다른 레코드와 일치하는 경우 변환이 각 레코드 쌍에 대해 행을 씁니다.

ID 분석 유형에서 **일치하는 쌍**을 선택할 수 있습니다.

## 일치 출력 속성

일치 출력 보기에는 캐시 메모리 동작 및 점수 일치 임계값을 지정하는 속성이 포함됩니다. 이 속성을 사용하면 변환이 분석할 데이터 저장소 레코드를 선택하고 데이터 저장소 레코드를 출력으로 기록하는 방식도 결정할 수 있습니다.

일치 출력 유형을 선택한 후에 다음 속성을 구성하십시오.

### 캐시 디렉터리

데이터 통합 서비스에서 ID 일치 분석 중에 임시 데이터를 쓸 디렉터리를 지정합니다. 일치 분석이 생성하는 데이터 양이 사용 가능한 시스템 메모리보다 클 경우 데이터 통합 서비스가 이 디렉터리에 임시 파일을 기록합니다. 데이터 통합 서비스는 매핑을 실행한 후에 임시 파일을 삭제합니다.

속성에 디렉터리 경로를 입력하거나 시스템 매개 변수를 사용하여 디렉터리를 식별할 수 있습니다. 데이터 통합 서비스 시스템의 로컬 경로를 지정하십시오. 데이터 통합 서비스가 기록할 수 있는 디렉터리를 지정해야 합니다. 기본값은 **CacheDir** 시스템 매개 변수입니다.

### 캐시 크기

데이터 통합 서비스에서 ID 일치 분석에 할당하는 시스템 메모리 양을 결정합니다. 기본값은 **400,000**바이트입니다.

일치 분석이 더 많은 데이터를 생성할 경우 데이터 통합 서비스는 초과 데이터를 캐시 디렉터리에 기록합니다. 일치 분석에 시스템 메모리 및 파일 저장소가 제공할 수 있는 것보다 많은 메모리가 필요할 경우 매핑이 실패합니다.

**참고:** 65536 이상의 값을 입력한 경우 일치 변환이 값을 바이트로 읽습니다. 이보다 낮은 값을 입력하면 일치 변환이 값을 메가바이트로 읽습니다.

### 일치

변환이 데이터베이스 테이블에서 인덱스 데이터를 읽을 때 분석할 레코드를 식별합니다. **일치 유형** 보기의 옵션을 사용하여 인덱스 테이블을 식별합니다.

기본적으로 변환은 데이터 소스 및 인덱스 데이터베이스 테이블의 모든 레코드를 분석합니다. 일치 속성을 구성하여 중복 분석에 대한 레코드 하위 집합을 지정하십시오.

### 출력

인덱스 데이터베이스 테이블을 읽는 변환을 구성할 때 변환이 출력으로 기록하는 레코드를 필터링합니다.

**일치 유형** 보기의 옵션을 사용하여 인덱스 테이블을 식별합니다.

기본적으로 일치 변환은 데이터 소스 및 인덱스 데이터베이스 테이블의 모든 레코드를 출력으로 기록합니다. 입력 데이터의 모든 레코드를 검토하지 않아도 되는 경우 **출력** 속성을 구성하십시오.

### 임계값

두 레코드를 서로의 잠재적 중복으로 식별하는 최소 일치 점수를 설정합니다.

매개 변수를 임계값에 할당할 수 있습니다. 10진수 값을 0에서 1 사이의 범위로 설정합니다.

## 일치 속성 구성

**일치 출력** 보기의 **일치** 속성을 사용하여 변환이 분석할 입력 데이터를 선택하는 방식을 지정합니다. 인덱스 데이터의 지속형 저장소를 읽도록 일치 변환을 구성할 때 속성도 구성합니다. 일치 속성은 **일치 유형** 보기에서 설정한 옵션을 구체화합니다.

일치 속성을 구성하여 다음 유형의 분석을 수행할 수 있습니다.

### 데이터 소스 레코드와 인덱스 데이터 레코드 비교

데이터 소스와 인덱스 데이터 테이블 간의 중복 레코드를 찾으려면 **제외**를 선택합니다.

제외 옵션을 선택하면 일치 변환이 데이터 소스 레코드와 인덱스 데이터 저장소를 비교합니다. 데이터 소스 또는 데이터 저장소 안에 있는 레코드는 분석되지 않습니다.

인덱스 데이터 저장소에 중복 레코드가 없고 데이터 소스에도 중복 레코드가 없는 경우 **제외**를 선택합니다.

### 데이터 소스 레코드와 인덱스 데이터 레코드를 비교하고 각 데이터 소스 레코드를 서로 비교

데이터 소스에서 중복 레코드를 찾고 데이터 소스와 인덱스 테이블 간의 중복 레코드를 찾으려면 **부분**을 선택합니다.

변환이 데이터 소스 레코드와 인덱스 데이터 저장소를 비교합니다. 또한 데이터 소스 안의 레코드를 서로 비교합니다.

인덱스 데이터 저장소에 중복 레코드가 없다는 것을 알지만 데이터 소스에서 중복 분석을 수행하지 않은 경우 **부분**을 선택합니다.

### 데이터 소스 및 인덱스 테이블의 모든 레코드를 단일 데이터 집합으로 비교

데이터 소스와 인덱스 테이블 간의 중복 레코드를 찾고 데이터 소스 및 인덱스 테이블 안에 있는 중복 레코드를 찾으려면 **전체**를 선택합니다. 기본 옵션은 전체입니다.

변환이 데이터 소스 및 데이터 저장소를 단일 데이터 집합으로 분석하고 이 데이터 집합 안의 모든 레코드 데이터를 비교합니다.

데이터 집합에 중복 레코드가 없는지 확인할 수 없는 경우 **전체** 옵션을 선택합니다.

## 출력 속성 구성

**일치 출력** 보기의 **출력** 속성을 사용하여 변환이 출력으로 쓰는 레코드를 필터링합니다. 인덱스 데이터 테이블을 지정하고 클러스터링된 출력 형식을 선택할 때 속성을 구성합니다. 데이터 소스에서 하나 이상의 레코드가 포함된 클러스터로 출력을 제한하려면 레코드를 필터링합니다.

다음과 같은 방식으로 출력 데이터를 필터링할 수 있습니다.

### 데이터 소스 또는 인덱스 테이블의 레코드가 포함된 모든 클러스터 쓰기

**모든 행**을 선택합니다. 변환이 데이터 소스 또는 인덱스 데이터 저장소의 레코드가 하나 이상 포함된 모든 클러스터를 씁니다. 기본값은 모든 행입니다.

클러스터는 레코드 하나를 포함할 수 있으므로 출력에는 모든 레코드가 포함됩니다.

### 데이터 소스의 레코드가 포함된 모든 클러스터 쓰기

**새 행 및 연결된 행**을 선택합니다. 변환에서 데이터 소스의 레코드를 하나 이상 포함하는 모든 클러스터를 씁니다.

클러스터에 단일 레코드가 포함될 수 있기 때문에 출력에는 데이터 소스의 모든 레코드가 포함됩니다. 클러스터는 인덱스 테이블의 레코드도 포함할 수 있습니다.

### 데이터 소스의 모든 클러스터 쓰기

**새 행만**을 선택합니다. 변환이 데이터 소스의 레코드가 포함된 클러스터를 씁니다. 출력에는 인덱스 테이블의 레코드가 포함되지 않습니다.

# ID 일치의 고급 속성

변환에는 실행 인스턴스 수, 동일 행에 대한 변환의 분석 여부 및 로그 데이터의 추적 수준을 결정하는 고급 속성이 포함됩니다.

다음과 같은 고급 속성을 구성할 수 있습니다.

## 실행 인스턴스

데이터 통합 서비스가 런타임 시 현재 변환에 대해 작업을 시도하는 스레드 수를 지정합니다. 변환이 포함된 매핑에서 최대 병렬도 런타임 속성을 재정의하는 경우 데이터 통합 서비스가 실행 인스턴스 값을 고려합니다. 기본 실행 인스턴스 값은 자동입니다.

데이터 통합 서비스는 여러 요인을 고려하여 변환에 할당할 스레드 수를 결정합니다. 사용자 요인은 실행 인스턴스 값, 매핑 및 도메인의 연결된 응용 프로그램 서비스의 값입니다.

매핑을 실행하는 데이터 통합 서비스는 다음 값을 읽어 변환에 대해 사용할 스레드 수를 결정합니다.

- 데이터 통합 서비스의 *최대 병렬도* 값. 기본값은 1입니다.
- 매핑 수준에서 설정하는 *최대 병렬도* 값. 기본값은 자동입니다.
- 변환의 *실행 인스턴스* 값. 기본값은 자동입니다.

매핑 수준에서 최대 병렬도 값을 재정의하는 경우 데이터 통합 서비스가 속성에서 가장 낮은 값을 사용 시도하여 스레드 수를 결정합니다.

매핑 수준에서 기본 최대 병렬도 값을 사용하는 경우 데이터 통합 서비스가 실행 인스턴스 값을 무시합니다.

실행 인스턴스 수를 설정할 때는 다음 규칙 및 지침을 고려합니다.

- 데이터 통합 서비스에서는 여러 사용자가 동시 매핑을 실행할 수 있습니다. 올바른 스레드 수를 계산하려면 데이터 통합 서비스가 액세스할 수 있는 중앙 처리 장치의 수를 동시 매핑의 수로 나눕니다.
- 기본 실행 인스턴스 값 및 기본 최대 병렬도 값을 사용하는 경우 변환 작업은 분할 가능하지 않습니다.

## 정확한 일치 항목 필터링

변환에서 일치 전략의 비교 알고리즘을 입력 데이터의 동일한 레코드 쌍에 적용할지 여부를 결정합니다.

변환이 동일한 레코드 쌍을 발견할 경우 알고리즘을 통해 레코드 간의 유사점 수준을 분석하지 않아도 됩니다. 변환에서 추가 분석 없이 레코드를 바로 출력 단계로 전달할 수 있습니다. 동일한 레코드를 바로 출력 단계로 전달하도록 변환을 구성하려면 정확한 일치 항목 필터링을 선택합니다. 입력 데이터에 다수의 동일 행이 포함될 경우 비교 알고리즘이 수행하는 계산 수가 줄어들므로 매핑이 더 빠르게 실행됩니다.

이 옵션은 입력 데이터에 다수의 동일 행이 포함될 경우에 선택합니다. 입력 데이터에 다수의 동일 행이 포함되지 않을 경우 이 옵션을 선택하면 변환이 더 느리게 실행될 수 있으니 선택하지 마십시오.

**참고:** 이 옵션을 선택하거나 선택 취소할 경우 변환 출력에 동일한 레코드 데이터가 포함됩니다. 이 옵션을 선택하고 선택 취소할 경우 변환이 출력 레코드에 서로 다른 링크 점수 및 드라이버 점수를 할당할 수 있습니다.

## 추적 수준

이 변환에 대해 로그에 표시되는 세부 정보의 양입니다. 간단, 보통, 자세한 정보 표시 초기화 또는 자세한 정보 표시 데이터를 선택할 수 있습니다. 기본값은 보통입니다.

## 지속형 인덱스 사례 연구

사용자가 여러 분기를 가진 은행의 데이터 스튜어드입니다. 모든 분기에 있는 고객 계정 레코드의 마스터 데이터 집합을 관리합니다. 인덱스 데이터베이스 테이블 집합을 사용하여 고객 계정 데이터베이스에 중복 데이터가 포함되어 있지 않은지 확인합니다.

인덱스 데이터 저장소를 작성하고 관리하려면 다음 작업을 수행하십시오.

- 데이터 저장소를 작성합니다.
- 데이터 저장소를 은행 분기의 최신 데이터로 업데이트합니다.  
계정 데이터를 데이터 저장소에 추가하거나 데이터 저장소의 현재 데이터를 업데이트할 수 있습니다.
- 데이터 저장소에서 더 이상 사용되지 않는 레코드를 제거합니다.

각 작업을 수행하면 데이터 저장소에 중복 레코드가 작성될 수 있다는 사실을 알아 두십시오. 데이터를 마스터 데이터 저장소 데이터에 추가하기 전에 분기 데이터를 분석할 정책을 개발할지 결정합니다. ID 일치 분석을 사용하여 분기 데이터를 분석하고 데이터가 데이터 저장소에 중복 ID를 작성하지 않았는지 확인합니다. 일치 변환에서 지속형 인덱스 옵션을 구성하여 분기 데이터와 데이터 저장소를 분석합니다.

### 지속형 인덱스 데이터 관리에 대한 정책 개발

데이터 스튜어드로서 고객 계정 데이터 저장소에 중복 ID가 포함될 수 없다고 표시하는 비즈니스 규칙을 정의합니다. ID 일치 매핑을 설계하여 준비 데이터베이스의 분기 데이터를 분석한 후 데이터를 데이터 저장소에 추가합니다.

분기 데이터를 데이터 저장소에 추가하는 작업은 다음의 경우 중복 ID를 작성할 수 있습니다.

- 분기 데이터에 중복 ID가 있습니다.
- 분기 데이터에 인덱스에도 포함된 ID가 있습니다.
- 분기 데이터가 데이터 저장소의 더 새로운 ID 버전을 포함하고 더 새로운 버전이 인덱스의 다른 ID와 일치합니다.

준비 데이터베이스를 데이터 저장소와 비교할 때 분기 데이터의 중복 레코드 상태를 반영하는 지속형 인덱스 옵션을 선택합니다. 데이터 저장소를 업데이트하기 전에 분기 데이터를 인덱스 데이터와 비교할지 결정할 수 있습니다.

**참고:** 일부 옵션에서 일치 분석을 활성화하거나 비활성화할 수 있습니다. 일치 분석을 활성화하면 매핑 데이터를 분석하거나 인덱스 데이터 저장소를 매핑 데이터와 비교합니다. 데이터 분석이 필요 없는 경우 일치 분석을 비활성화합니다. 일치 출력 탭의 일치 속성을 사용해도 일치 분석에서 데이터를 포함하거나 제외할 수 있습니다.

### 매핑 데이터 소스를 인덱스 데이터 저장소와 비교

매핑 입력 데이터를 인덱스 데이터 저장소와 비교하고 데이터 저장소를 변경하지 않으려면 다음 옵션을 선택하십시오.

- 데이터베이스를 업데이트하지 않음

매핑에서 입력 데이터를 인덱스 데이터 저장소와 비교합니다. 매핑에서 어떤 데이터도 인덱스 데이터 저장소에 추가, 제거 또는 업데이트하지 않습니다.

이 옵션을 선택하면 ID 일치 분석을 비활성화할 수 없습니다.

인덱스 데이터를 업데이트하지 않으므로 저장소에 중복 행을 작성할 수 없습니다. 일치 출력 탭의 일치 속성에서 데이터 프로젝트의 현재 요구 조건과 일치하는 옵션을 선택합니다. 예를 들어 **전체** 옵션을 선택합니다. **전체** 옵션은 매핑 데이터에 중복 항목이 포함되지 않았는지 확인하고 매핑 데이터가 중복 항목을 데이터 저장소에 추가하지 않는지 확인합니다.

**참고:** 해당 옵션을 사용하여 매핑 데이터와 데이터 저장소를 비교한 후 데이터 저장소를 업데이트합니다. 매핑 출력에 매핑 데이터가 중복 항목을 데이터 저장소에 추가하지 않는다고 표시되면 매핑을 다시 실행합니다. 매핑을 다시 실행할 때 데이터베이스를 업데이트하도록 옵션을 선택합니다.

## 데이터 저장소 작성 및 데이터 저장소에 행 추가

데이터 저장소를 작성하고 매핑 데이터의 행을 데이터 저장소에 추가하려면 다음 옵션을 선택하십시오.

- 데이터베이스를 새 ID로 업데이트

행이 시퀀스 식별자를 데이터 저장소의 행과 공유하지 않는 경우 매핑에서 행을 데이터 저장소에 추가합니다. 매핑은 인덱스 테이블의 어떤 행도 덮어쓰지 않습니다. 빈 데이터베이스 테이블을 지정하면 매핑에서 모든 매핑 인덱스 데이터를 테이블에 기록합니다.

이 옵션을 선택하면 ID 일치 분석을 활성화하거나 비활성화할 수 있습니다. 이 옵션은 기본적으로 일치 분석을 활성화합니다.

인덱스 행을 업데이트하지 않기 때문에 일치 출력 탭의 일치 속성에서 **제외** 옵션이나 **부분** 옵션을 선택합니다. 이전 프로세스에서 매핑 데이터 행의 고유성이 확인된 경우 **제외** 옵션을 사용합니다.

## 데이터 저장소에서 행 업데이트

데이터 저장소의 현재 행을 매핑 데이터로 업데이트하려면 다음 옵션을 선택하십시오.

- 데이터베이스에서 현재 ID 업데이트

레코드에서 시퀀스 식별자를 매핑 데이터의 레코드와 공유하는 경우 매핑에서 데이터 저장소의 현재 레코드를 업데이트합니다. 매핑은 어떤 행도 인덱스 테이블에 추가하지 않습니다.

이 옵션을 선택하면 ID 일치 분석을 활성화하거나 비활성화할 수 있습니다. 이 옵션은 기본적으로 일치 분석을 비활성화합니다.

인덱스 행을 인덱스 테이블에 추가하지 않기 때문에 일치 출력 탭의 일치 속성에서 **전체** 옵션을 선택합니다.

**참고:** 데이터 저장소의 행을 업데이트하는 경우 매핑 소스 데이터와 데이터 저장소 간에 중복 항목이 있을 수 있습니다. **전체** 옵션을 선택하여 데이터 저장소에 추가한 ID 데이터가 저장소의 현재 데이터와 일치하지 않는지 확인합니다.

## 데이터 저장소에서 행 제거

데이터 저장소에서 행을 제거하려면 다음 옵션을 선택하십시오.

- 데이터베이스에서 ID 제거

행이 시퀀스 식별자를 매핑 데이터의 레코드와 공유하는 경우 매핑은 데이터 저장소에서 해당 행을 삭제합니다.

이 옵션을 선택하면 ID 일치 분석을 활성화하거나 비활성화할 수 있습니다. 이 옵션은 기본적으로 일치 분석을 비활성화합니다.

**참고:** 데이터 저장소에서 데이터를 제거한 경우 저장소 행 간의 관계를 변경하십시오. 저장소에 중복 ID가 포함된 경우 클러스터의 연결된 레코드 데이터 또는 드라이버 레코드 데이터를 제거해야 할 수 있습니다. 또는 일치된 쌍에서 가장 잘 일치하는 항목에 대한 데이터를 제거해야 할 수 있습니다. 매핑을 다시 실행하면 매핑에서 다른 클러스터 또는 중복 쌍을 생성할 수 있습니다. 중복 레코드를 포함하지 않는 데이터 저장소에서 행을 제거하면 레코드의 중복 상태를 변경할 수 없습니다. 행을 제거한 후 매핑을 실행하면 매핑에서 데이터 집합에 남아 있는 ID에 대해 동일한 일치 점수를 생성합니다.

## ID 일치 분석 예제

사용자는 여러 도시에 개발 센터가 있는 소프트웨어 조직의 인사부 관리자입니다. 조직은 직원의 인사 레코드를 본사 데이터베이스에 저장합니다. 개발 센터에서 정기적으로 직원을 채용하고 채용한 직원의 인사 데이터를 사용자에게 보냅니다.

인사 레코드를 스프레드시트 파일로 추가하고 파일 데이터를 사용하여 직원 데이터베이스를 업데이트합니다. 현재 파일에 중복 ID가 포함되어 있는지 확인하고자 합니다.

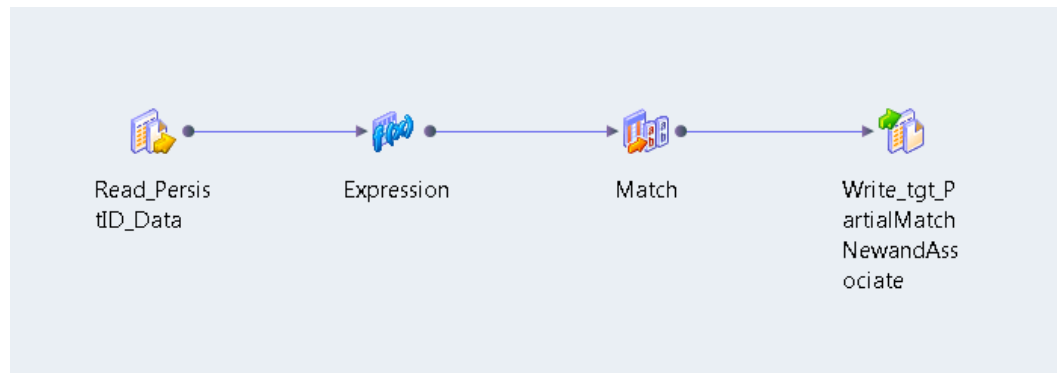
직원 레코드에 대한 ID 분석을 수행하는 매핑을 설계합니다. 일치 변환을 구성하여 스프레드시트 파일의 중복 ID를 검색합니다. 또한 파일 데이터가 마스터 데이터베이스의 어떤 직원 데이터와도 중복되지 않는지 확인해야 합니다. 일치 변환을 구성하여 조직 직원을 위해 저장한 마스터 데이터와 파일 데이터를 비교합니다.

데이터베이스가 마스터 데이터 집합이기 때문에 직원 레코드의 인덱스 데이터를 지속형 데이터 저장소에 저장합니다.

### 매핑 작성

중복 ID를 찾는 매핑을 작성하십시오. 매핑에서 데이터 소스를 읽고, 시퀀스 식별자를 소스 레코드에 추가하며, ID 분석을 수행하고, 결과를 데이터 대상에 기록합니다.

다음 이미지는 **Developer tool**의 매핑을 보여 줍니다.



작성하는 매핑에는 다음 개체가 포함됩니다.

개체 이름	설명
Read_PersistID_Data	데이터 소스 직원 이름 및 세부 정보가 포함되어 있습니다.
식	식 변환입니다. 시퀀스 식별자 값을 소스 데이터에 추가합니다.
일치	일치 변환. 소스 데이터 ID의 중복 수준을 분석합니다.
Write_tgt_PartialMatchNewandAssociate	데이터 대상. ID 분석 결과가 포함되어 있습니다.

**참고:** 매핑에서 키 생성기 변환을 사용하지 않습니다. ID 일치 분석에서 키 생성기 변환은 선택 사항입니다.



## 입력 데이터 샘플

직원의 이름, 직원이 근무하는 도시 및 직원에게 지정된 역할이 포함된 직원 파일이 있습니다. 이 직원 파일에 기반하는 데이터 소스를 모델 리포지토리에 작성합니다. 데이터 소스를 매핑에 추가합니다.

다음은 직원 파일에 있는 직원 데이터 샘플을 보여주는 데이터 조각입니다.

이름	도시	역할
Chaithra	방갈로르	SE
Ramanan	첸나이	SSE
Ramesh	첸나이	SSE
Ramesh	첸나이	팀장
Sunil	방갈로르	부장
Venu	하이데라바드	부장
Harish	방갈로르	SE
Sachin	방갈로르	SSE

## 식 변환 구성

식 변환을 구성하는 경우 매핑 출력에 포함할 모든 데이터 소스 포트를 연결해야 합니다. 포트를 통과 포트로 연결합니다. 시퀀스 ID 값을 포트에 추가하는 식을 작성합니다.

다음 식은 변수 *Init3*를 작성하고 정수 값 1267을 각 시퀀스 식별자에 추가합니다.

*Init3+1267*

다음 표에는 직원 데이터 소스를 읽는 식 변환의 포트가 설명되어 있습니다.

이름	포트 유형	포트 그룹
SEQID	bigint	출력만
이름	문자열	통과
도시	문자열	통과
역할	문자열	통과
Init3	정수	변수

## 일치 변환 구성

재사용할 수 없는 일치 변환을 매핑에 추가하여 ID 분석을 수행합니다.

다음 태스크를 수행하여 변환을 구성하십시오.

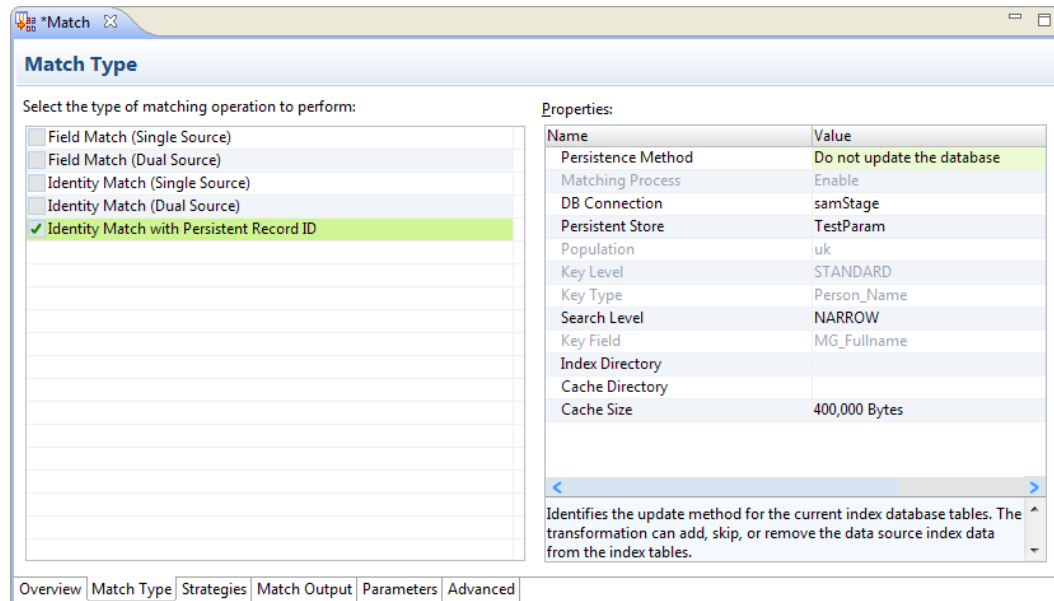
1. 수행할 일치 분석의 유형을 선택합니다.
2. 입력 포트를 변환에 연결합니다.

3. 레코드 데이터를 비교하는 전략을 구성합니다.
4. 변환에서 작성할 일치 출력 데이터의 유형을 선택합니다.
5. 출력 포트를 데이터 대상에 연결합니다.

## 일치 작업의 유형 선택

일치 유형 보기의 옵션을 사용하여 일치 작업을 선택합니다.

다음 이미지는 일치 유형 보기를 보여 줍니다.



데이터 소스의 인덱스 데이터를 마스터 데이터 집합의 인덱스 데이터와 비교하려면 **지속형 레코드 ID로 ID 일치**를 선택합니다. 일치 분석이 데이터를 인덱스 테이블에 추가하지 않도록 지속성 방법을 업데이트합니다. 매핑 결과를 검토한 후 인덱스 테이블을 업데이트할지 결정할 수 있습니다.

**DB 연결** 옵션을 사용하여 인덱스 데이터가 포함된 데이터베이스를 식별합니다. **지속형 저장소** 옵션을 사용하여 인덱스 테이블을 선택합니다.

**참고:** 일치 변환은 인덱스 데이터베이스 테이블의 메타데이터에서 ID 인구집단, 키 수준, 키 유형 및 키 필드 속성 값을 읽습니다. 이 값은 인덱스 데이터 저장소에 작성된 변환의 관련 속성과 일치합니다.

## 입력 포트 연결

입력 데이터 포트를 변환에 연결합니다. 포트 이름, 포트 순서, 데이터 유형 및 전체 자릿수가 데이터 저장소를 작성한 변환의 포트 구성과 일치하는지 확인합니다.

일치 변환은 미리 설정된 입력 포트를 사용하여 레코드 처리 순서를 결정합니다. 이 변환은 시퀀스 식별자를 사용하여 입력 포트부터 출력으로 기록하는 일치하는 쌍 또는 클러스터까지의 레코드를 추적합니다. 일치 변환이 처리하는 레코드는 그룹 키를 사용하여 정렬됩니다.

사전 설정된 포트를 식 변환의 다음 포트에 연결합니다.

- SequenceID. 식 변환의 SEQID 포트에 연결합니다.
- GroupKey. 식 변환의 City 포트에 연결합니다.

## ID 분석에 대한 전략 구성

전략 보기의 옵션을 사용하여 전략을 구성합니다. 전략은 변환이 레코드 데이터에 수행하는 분석 유형을 결정합니다.

레코드 데이터에 대해 **Person\_Name** 알고리즘을 선택합니다. 분석할 **Name** 입력 포트를 선택합니다. 변환이 포트 데이터의 사본을 작성하므로 **Name\_1** 포트 및 **Name\_2** 포트를 선택합니다.

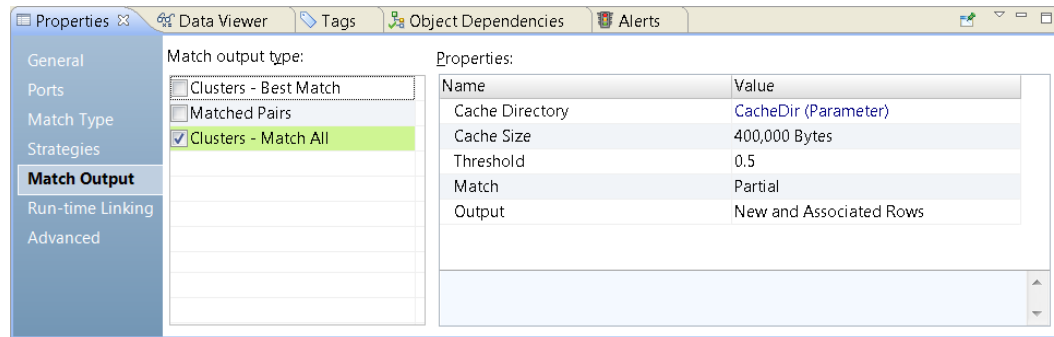
### 관련 항목:

- [“ID 일치 알고리즘” 페이지 460](#)

## 일치 출력 유형 선택

일치 분석의 결과에 대한 출력 형식을 정의하려면 일치 출력 보기의 옵션을 사용하십시오.

다음 이미지에서는 단일 소스 ID 분석에 대한 일치 출력 보기를 보여 줍니다.



출력 레코드를 클러스터로 구성하도록 변환을 구성하십시오. 각 클러스터에는 사용자가 지정한 일치 속성을 기반으로 하나 이상의 다른 레코드와 일치하는 모든 레코드가 포함되어 있습니다. 일치 속성에서 변환이 데이터 소스 레코드를 인덱스 레코드에 비교하는 방법을 결정합니다.

다음 표에는 직원 레코드 데이터를 분석하기 위해 지정하는 일치 속성 옵션이 설명되어 있습니다.

일치 속성	옵션	옵션 설명
일치	일부	변환이 데이터 소스 레코드와 인덱스 데이터 저장소를 비교합니다. 또한 데이터 소스 안의 레코드를 서로 비교합니다.
출력	새 행 및 연결된 행	변환에서 데이터 소스의 레코드를 하나 이상 포함하는 모든 클러스터를 기록합니다. 클러스터에 인덱스 데이터 저장소의 레코드가 포함될 수 있습니다. 클러스터에 단일 레코드가 포함될 수 있기 때문에 출력에는 데이터 소스의 모든 레코드가 포함됩니다.

### 관련 항목:

- [“일치 속성 구성” 페이지 464](#)
- [“출력 속성 구성” 페이지 464](#)

## 출력 포트 연결

일치 변환의 출력 포트를 매핑의 데이터 대상에 연결합니다. 데이터 대상에 쓸 레코드 데이터가 포함된 포트를 선택하십시오.

이 변환에는 클러스터된 데이터를 위한 일련의 미리 설정된 포트가 포함됩니다. 레코드의 중복 상태를 나타내는 미리 설정된 포트를 선택하고 각 레코드를 저장하는 데이터 소스를 식별하십시오.

다음 포트에는 중복 레코드를 찾고 소스 또는 레코드를 결정할 때 사용할 수 있는 데이터가 포함됩니다.

- **ClusterSize** 포트는 클러스터의 레코드 수를 나타냅니다. 클러스터 크기가 1보다 큰 클러스터에 레코드가 속하는 경우 변환은 해당 레코드를 다른 레코드의 중복으로 간주합니다.
- **ClusterID** 포트는 레코드가 속하는 클러스터를 식별합니다. ClusterID 데이터를 사용하면 현재 레코드의 중복인 레코드를 찾을 수 있습니다.
- **PersistenceStatus** 포트는 코드 값을 사용하여 매핑 소스의 인덱스 데이터와 데이터 저장소의 인덱스 데이터 간의 관계를 설명합니다.
- **PersistenceStatusDesc** 포트는 PersistenceStatus 포트 코드 값에 대한 텍스트 설명을 반환합니다.

다른 포트를 사용하여 클러스터 레코드 간의 관계를 검토할 수도 있습니다. 링크 포트 값 및 드라이버 포트 값은 각 클러스터에 있는 레코드 간의 유사 정도를 나타냅니다.

현재 예에서는 모든 포트를 데이터 대상에 연결합니다. 포트의 출력 데이터를 보려면 데이터 뷰어를 실행합니다.

### 관련 항목:

- [“지속성 상태 코드 및 지속성 상태 설명” 페이지 439](#)
- [“상태 코드 값 및 상태 설명 값” 페이지 439](#)

## 데이터 뷰어 실행

데이터 뷰어를 실행하여 일치 분석의 결과를 검토합니다. 기본적으로 데이터 뷰어에는 일치 변환의 모든 출력 포트가 표시됩니다. 매핑을 실행하면 데이터 대상이 출력 포트의 데이터로 업데이트됩니다.

다음 이미지는 데이터 뷰어의 출력 데이터를 보여 줍니다.

The screenshot shows a workflow window titled '\*PartialMatchNewandAssociated\_MatchAll'. The workflow consists of four steps: 'Read\_IncreP', 'Expression', 'Match', and 'Write\_tgtLP'. Below the workflow, the 'Data Viewer' tab is active, displaying a table of output data. The table has columns: Name, ClusterSize, PersistenceStatusDesc, Designation, GroupKey, LinkId, ClusterId, LinkScore, and PersistenceSt.. The table contains 16 rows of data, showing various names and their associated cluster information.

Name	ClusterSize	PersistenceStatusDesc	Designation	GroupKey	LinkId	ClusterId	LinkScore	PersistenceSt..
1 Chaith	2	Input, Exists, Ignored	SE	Bangalore	S - 1	000000001	1	IEI00000
2 Chaith	2	Store, -, No change	SE	Bangalore	1 - 1267	000000001	1	SON00000
3 Ramana	2	Input, Exists, Ignored	SSE	Chennai	S - 2	000000002	1	IEI00000
4 Ramana	2	Store, -, No change	SSE	Chennai	1 - 2534	000000002	1	SON00000
5 Ramesh	4	Input, Exists, Ignored	SSE	Chennai	S - 3	000000003	1	IEI00000
6 Ramesh	4	Input, Exists, Ignored	Lead	Chennai	S - 3	000000003	1	IEI00000
7 Ramesh	4	Store, -, No change	SSE	Chennai	1 - 3801	000000003	1	SON00000
8 Ramesh	4	Store, -, No change	Lead	Chennai	1 - 3801	000000003	1	SON00000
9 Sunil	2	Input, Exists, Ignored	Principal	Bangalore	S - 5	000000004	1	IEI00000
10 Sunil	2	Store, -, No change	Principal	Bangalore	1 - 6335	000000004	1	SON00000
11 Venu	2	Input, Exists, Ignored	Principal	Hydrabad	S - 6	000000005	1	IEI00000
12 Venu	2	Store, -, No change	Principal	Hydrabad	1 - 7602	000000005	1	SON00000
13 Harish	2	Input, Exists, Ignored	SE	Bangalore	S - 7	000000006	1	IEI00000
14 Harish	2	Store, -, No change	SE	Bangalore	1 - 8869	000000006	1	SON00000
15 Sachin	2	Input, Exists, Ignored	SSE	Bangalore	S - 8	000000007	1	IEI00000
16 Sachin	2	Store, -, No change	SSE	Bangalore	1 - 10136	000000007	1	SON00000

이 데이터 뷰어에서는 파일에 마스터 데이터 집합의 데이터와 중복되는 레코드가 포함되는지 확인합니다.

데이터 뷰어의 다음 데이터 값을 고려하십시오.

- 데이터 집합의 각 레코드가 둘 이상의 레코드가 포함된 클러스터에 속합니다. 따라서 각 레코드는 최소한 다른 1개 레코드의 중복입니다. 고유한 레코드에는 클러스터 크기 1이 할당됩니다. 이 데이터 소스에는 마스터 데이터 집합에 포함되지 않는 레코드가 없습니다.
- **PersistenceStatusDesc** 데이터는 레코드의 출처를 식별하고 일치 변환이 레코드를 인덱스 테이블에 추가하는지 여부를 나타냅니다. 이 열은 각 입력 레코드가 마스터 데이터 집합에 존재한다는 것을 나타냅니다. 따라서 일치 변환은 데이터를 마스터 데이터 인덱스에 추가하지 않습니다.

## 결론

이 일치 분석의 결과를 보면 직원 레코드 파일에 포함된 레코드 중 마스터 데이터 집합에 포함되지 않은 레코드는 없다는 것을 알 수 있습니다. 또한 지속성 상태의 설명을 보면 매핑이 데이터 소스의 데이터로 인덱스 테이블을 업데이트하지 않는다는 것을 알 수 있습니다. 그러므로 이 직원 레코드 파일을 폐기할 수 있습니다.

지역 사무실에서 다른 업데이트를 보내면 다른 파일을 작성하여 마스터 데이터 집합과 비교할 수 있습니다. 또한 이 매핑과 인덱스 테이블을 재사용할 수 있습니다. 마스터 데이터 집합에 대한 인덱스 데이터가 데이터베이스 테이블에 저장되므로 인덱스 데이터를 다시 생성하지 않아도 됩니다.

## 제 31 장

# 병합 변환

이 장에 포함된 항목:

- [병합 변환 개요, 475](#)
- [병합 전략 구성, 475](#)
- [병합 변환 고급 속성, 476](#)
- [병합 변환 - 비원시 환경, 476](#)

## 병합 변환 개요

병합 변환은 여러 입력 열에서 데이터 값을 읽고 단일 출력 열을 생성하는 수동 변환입니다.

병합 변환을 사용하여 기본 형식으로 데이터를 생성합니다. 예를 들어 `Customer_Firstname` 및 `Customer_Surname` 필드를 결합하여 `Customer_FullName` 필드를 생성할 수 있습니다.

병합 변환 내에서 여러 병합 전략을 생성할 수 있습니다. 병합 변환에서는 전략을 생성하는 데 사용할 수 있는 마법사를 제공합니다.

## 병합 전략 구성

병합 전략을 구성하려면 병합 변환의 **전략** 보기에서 설정을 편집하십시오.

1. **전략** 보기를 선택합니다.
2. **새로 만들기**를 클릭합니다.  
새 **전략** 마법사가 열립니다.
3. **입력** 필드를 클릭하여 전략에 대한 입력 포트를 선택합니다.
4. 병합된 항목 사이에 배치할 병합 문자를 정의하려면 **선택**을 클릭합니다. 병합 문자를 선택하지 않은 경우 병합 변환에서 기본적으로 공백 문자를 사용합니다.
5. 필요에 따라 **병합된 출력에 빈 문자열 포함**을 선택하여 빈 입력 문자열을 출력에 포함합니다.
6. **마침**을 클릭합니다.

## 병합 변환 고급 속성

데이터 통합 서비스가 병합 변환에 대해 데이터를 처리하는 방법을 결정할 수 있는 속성을 구성합니다.

로그의 추적 수준을 구성할 수 있습니다.

**고급** 탭에서 다음 속성을 구성합니다.

### 추적 수준

이 변환에 대해 로그에 표시되는 세부 정보의 양입니다. 간단, 보통, 자세한 정보 표시 초기화 또는 자세한 정보 표시 데이터를 선택할 수 있습니다. 기본값은 보통입니다.

## 병합 변환 - 비원시 환경

비원시 환경에서 병합 변환의 처리는 변환을 실행하는 엔진에 따라 다릅니다.

다음과 같은 비원시 런타임 엔진에 대한 지원을 고려하십시오.

- **Blaze** 엔진. 제한 없이 지원됩니다.
- **Spark** 엔진. 제한 없이 지원됩니다.
- **Databricks Spark** 엔진. 지원되지 않습니다.



## 제 32 장

# 노멀라이저 변환

이 장에 포함된 항목:

- [노멀라이저 변환 개요, 477](#)
- [여러 번 발생하는 필드, 478](#)
- [여러 번 발생하는 레코드, 478](#)
- [입력 계층 정의, 480](#)
- [노멀라이저 변환 출력 그룹 및 포트, 487](#)
- [출력 그룹에 대한 키 생성, 491](#)
- [노멀라이저 변환 고급 속성, 491](#)
- [노멀라이저 변환 작성, 492](#)
- [업스트림 소스에서 노멀라이저 변환 작성, 492](#)
- [노멀라이저 매핑 예제, 493](#)
- [노멀라이저 변환 - 비원시 환경, 496](#)

## 노멀라이저 변환 개요

노멀라이저 변환은 소스 행 하나를 여러 대상 행으로 변환하는 활성 변환입니다. 노멀라이저 변환은 여러 번 발생하는 데이터가 포함된 행을 받으면 여러 번 발생하는 데이터의 각 인스턴스에 대해 행을 반환합니다.

노멀라이저 변환은 여러 번 발생하는 데이터를 개별 출력 행으로 구문 분석합니다. 관계형 소스 행에 4개의 영업 분기가 포함된 경우를 예로 들어 보겠습니다. 이 경우 노멀라이저 변환은 각 영업 발생에 대해 개별 출력 행을 생성합니다.

노멀라이저 변환을 정의할 때는 소스 데이터 구조를 설명하는 입력 행 계층을 구성합니다. 필요에 따라 변환 입력 계층에서 레코드를 정의할 수 있습니다. 레코드는 필드 그룹의 컨테이너입니다. 소스 데이터에서 필드 그룹이 여러 번 발생하면 레코드를 정의합니다. 입력 계층에 따라 변환 출력 그룹을 구성할 수 있는 방법이 결정됩니다.

노멀라이저 변환은 관계형 테이블이나 플랫폼 파일 소스에서 데이터를 변환합니다.

## 여러 번 발생하는 필드

소스 데이터에서 여러 번 반복되는 필드는 입력 행 계층에서 여러 번 발생하는 필드로 정의할 수 있습니다. 노멀라이저 변환은 소스의 필드 그룹 또는 여러 번 발생하는 필드의 각 발생에 대해 개별 행을 반환할 수 있습니다.

소스 행에 매장별 4개 영업 분기가 포함된 경우를 예로 들어 보겠습니다.

매장	Sales(1)	Sales(2)	Sales(3)	Sales(4)
Store1	100	300	500	700
Store2	250	450	650	850

노멀라이저 입력 계층을 정의할 때 이 4개의 영업 필드를 여러 번 발생하는 필드 하나로 결합할 수 있습니다.

Qtr\_Sales와 같은 필드 이름을 정의하고 소스에서 해당 필드가 4번 발생하도록 구성합니다.

출력 그룹에 매장 데이터와 영업 데이터가 포함되어 있으면 노멀라이저 변환은 각 Store 및 Qtr\_Sales 결합에 대해 행을 반환합니다. 출력 행은 해당 출력 행에 있는 Qtr\_Sales 인스턴스를 식별하는 인덱스를 포함합니다.

변환에서는 다음 행을 반환합니다.

매장	Qtr_Sales	Qtr (GCID)
Store1	100	1
Store1	300	2
Store1	500	3
Store1	700	4
Store2	250	1
Store2	450	2
Store2	650	3
Store2	850	4

출력 그룹에 단일 발생 열과 여러 번 발생하는 열이 포함되어 있으면 노멀라이저는 각 출력 행에서 단일 발생 열에 대해 중복 데이터를 반환합니다. 예를 들어 Store1과 Store2가 각 Qtr\_Sales 인스턴스에 대해 반복됩니다.

소스 행은 여러 번 발생하는 데이터의 수준을 둘 이상 포함할 수 있습니다. 입력 계층을 정의하는 방법에 따라 각 수준에서 개별 행을 반환하도록 노멀라이저 변환을 구성할 수 있습니다.

## 생성된 열 ID

노멀라이저 변환은 다중 발생 필드의 각 인스턴스에 대해 GCID(생성된 열 ID) 출력 포트를 반환합니다.

생성된 열 ID 포트는 다중 발생 데이터의 인스턴스에 대한 인덱스입니다. 예를 들어 소스 레코드에서 필드가 4번 발생하는 경우 Developer tool은 다중 발생 데이터가 행에서 발생하는 인스턴스에 따라 생성된 열 ID 포트에 값 1, 2, 3 또는 4를 반환합니다.

## 여러 번 발생하는 레코드

노멀라이저 변환 소스 데이터에서 여러 번 발생하는 레코드를 정의할 수 있습니다. 레코드는 필드 그룹입니다. 여러 번 발생하는 소스 필드 그룹을 정의해야 할 때는 노멀라이저 변환에서 레코드를 정의합니다.

### 여러 번 발생하는 레코드 예제

다음 Customer 행에는 집 주소 정보와 회사 주소 정보를 포함하는 고객 정보가 들어 있습니다.

CustomerID  
FirstName

```
LastName
Home_Street
Home_City
Home_State
Home_Country
Business_Street
Business_City
Business_State
Business_Country
```

노멀라이저 변환을 구성할 때는 **Customer** 필드와 여러 번 발생하는 주소 레코드를 포함하는 입력 구조를 정의할 수 있습니다. 주소 레코드는 두 번 발생합니다. 노멀라이저 변환 출력 그룹을 구성할 때는 **CustomerID**, **FirstName** 및 **LastName** 필드가 아닌 다른 대상으로 **Address** 레코드를 반환할 수 있습니다.

다음 예제에서는 여러 번 발생하는 주소 레코드가 포함된 입력 구조를 보여 줍니다.

```
CustomerID
FirstName
LastName
Address (occurs twice)
  Street
  City
  State
  Country
```

하위 레코드는 레코드 내의 레코드입니다. 레코드와 하위 레코드를 정의할 때는 소스 행에서 필드의 입력 계층을 정의합니다. 각 레코드는 변환 출력을 정의할 때 참조할 수 있는 계층의 노드입니다.

각 주소 유형에 대해 소스 행에 전화 번호 여러 개가 포함되는 경우를 예로 들어 보겠습니다.

```
CustomerID
FirstName
LastName
Home_Street
Home_City
Home_State
Home_Country
Telephone_No
Cell_Phone_No
Alternate_Phone_No
Business_Street
Business_City
Business_State
Business_Country
Business_Telephone_No
Business_Cell_Phone_No
Business-Alternate_Phone1
```

**Address**가 **Phone**의 상위 항목으로 지정되면 입력 계층을 정의합니다. 노멀라이저 변환 출력을 정의할 때는 주소와 전화 번호를 반환하여 대상을 고객 정보와 구분할 수 있습니다.

다음 예제와 같은 입력 계층을 정의합니다.

```
CustomerID
FirstName
LastName
Address (occurs twice)
  Street
  City
  State
  Country
  Phone
    Telephone_No (occurs three times)
```

# 입력 계층 정의

노멀라이저 변환을 생성할 때는 소스의 필드와 레코드를 설명하는 입력 계층을 정의합니다. 변환의 **노멀라이저** 보기에서 입력 계층을 정의합니다.

**Developer tool**은 입력 계층에서 정의하는 필드에 따라 변환 입력 포트를 생성합니다. 변환 출력 그룹을 정의하기 전에 입력 그룹 구조를 정의합니다.

입력 계층을 정의할 때는 소스 데이터의 구조에 해당하는 입력 구조를 정의해야 합니다. 소스 데이터는 여러 번 발생하는 필드의 그룹을 둘 이상 포함할 수 있습니다. 구조를 정의하려는 경우 소스의 다른 레코드와 같은 수준에서 발생하는 레코드를 구성할 수 있습니다. 또는 다른 레코드 내에서 발생하는 레코드를 정의할 수 있습니다.

## 입력 계층 예제

다음 소스 행은 두 번 발생하는 **Address** 레코드와 **Customer** 필드를 포함합니다.

```
CustomerID
FirstName
LastName
Address
  Street
  City
  State
  Country
Address1
  Street1
  City1
  State1
  Country1
```

**노멀라이저** 보기에서 입력 구조를 정의할 때는 **CustomerID**, **FirstName** 및 **LastName**을 필드로 추가할 수 있습니다. **Address** 레코드를 정의하고 주소에 **Street**, **City**, **State** 및 **Country** 필드를 포함합니다. **Address Occurs** 값을 2로 변경합니다.

다음 이미지에는 **노멀라이저** 보기의 입력 계층이 나와 있습니다.

Normalizer						
Name	Level	Occurs	Type	Precision	Scale	
CustomerID	1	1	string	10	0	
FirstName	1	1	string	10	0	
LastName	1	1	string	10	0	
Address	1	2				
Street	2	1	string	10	0	
City	2	1	string	10	0	
State	2	1	string	10	0	
Country	2	1	string	10	0	

**노멀라이저** 보기의 **Occurs** 열은 소스 행의 레코드나 필드 인스턴스 수를 식별합니다. 여러 번 발생하는 필드나 레코드에 대해 **Occurs** 열의 값을 변경합니다. 이 예제에서 **Customer** 필드는 한 번 발생하고 **Address** 레코드는 두 번 발생합니다.

**노멀라이저** 보기의 **Level** 열은 입력 계층에서 필드나 레코드가 표시되는 위치를 나타냅니다. **Customer** 필드는 계층의 수준 1에 있습니다. **Address** 레코드도 수준 1에 있습니다.

## 노멀라이저 변환 입력 포트

노멀라이저 보기의 입력 계층을 정의할 때 Developer tool이 노멀라이저 변환 입력 포트를 생성합니다. 입력 계층에서 필드를 변경하면 Developer tool이 입력 포트를 변경합니다.

**개요** 보기에서 노멀라이저 변환 입력 포트를 확인합니다. **개요** 보기에서 입력 포트의 순서를 다시 지정할 수 있습니다. 입력 포트를 변경하려면 **노멀라이저** 보기에서 입력 계층을 업데이트합니다.

입력 계층에서 필드를 여러 번 발생하도록 정의하면 Developer tool은 여러 번 발생하는 필드의 각 인스턴스에 대해 입력 포트를 하나씩 생성합니다. 레코드가 여러 번 발생하는 경우 Developer tool은 레코드의 각 필드 인스턴스에 대해 입력 포트를 생성합니다.

### 입력 포트 예제

다음 이미지에는 Developer tool에서 고객 데이터 및 여러 번 발생하는 주소 데이터에 대해 생성하는 입력 포트가 나와 있습니다.

Ports					
	Name	Type	Precision	Scale	Location
	Input				
1	CustomerID	string	10	0	CustomerID
2	FirstName	string	10	0	FirstName
3	LastName	string	10	0	LastName
4	Street	string	10	0	Address.Street
5	Street1	string	10	0	Address.Street
6	City	string	10	0	Address.City
7	City1	string	10	0	Address.City
8	State	string	10	0	Address.State
9	State1	string	10	0	Address.State
10	Country	string	10	0	Address.Country
11	Country1	string	10	0	Address.Country

## 필드 병합

노멀라이저 보기에서 비슷한 데이터 필드를 여러 번 발생하는 필드 하나로 병합할 수 있습니다. 다른 개체에서 포트를 끌어 매핑에서 노멀라이저 변환을 생성할 때는 필드를 병합해야 할 수 있습니다.

소스 행은 Base\_Salary, Bonus\_Pay, Sales\_Commissions 등 서로 다른 유형의 급여 데이터가 들어 있는 여러 필드를 포함할 수 있습니다. 필드를 병합하여 3번 발생하는 Salary 필드 하나를 생성할 수 있습니다.

다음 이미지에는 노멀라이저 보기에서 3개의 급여 유형을 선택한 직원 행이 나와 있습니다.

Normalizer					
Name	Level	Occurs	Type	Preci...	Scale
EmployeeID	1	1	string	10	0
Base_Salary	1	1	decimal	10	0
Bonus_Pay	1	1	decimal	10	0
Sales Commissions	1	1	decimal	10	0

3가지 급여 데이터 유형을 3번 발생하는 Salary 필드에 병합할 수 있습니다.

다음 이미지에는 Salary 필드가 나와 있습니다.

Normalizer					
Name	Level	Occurs	Type	Preci...	Scale
EmployeeID	1	1	string	10	0
Salary	1	3	decimal	10	0

## 필드 병합

노멀라이저 보기에서 같은 유형의 필드를 여러 번 발생하는 필드 하나로 병합합니다.

1. 노멀라이저 보기를 클릭합니다.
2. 병합할 필드를 선택합니다.
3. 병합 단추를 클릭합니다.  
필드 병합 대화 상자가 표시됩니다.
4. 병합 필드의 이름, 유형, 전체 자릿수, 배율 및 여러 번 발생하는 필드의 발생 횟수를 입력합니다.
5. 확인을 클릭합니다.

## 필드 플랫폼화

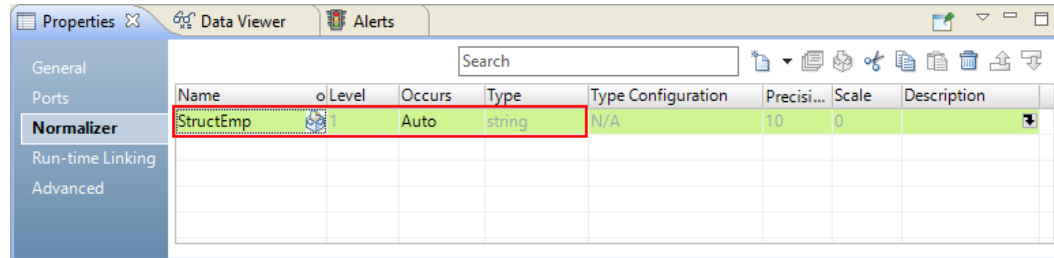
Spark 엔진에서 실행되는 매핑에서 복합 데이터 유형의 필드를 플랫폼화할 수 있습니다. 복합 포트를 통해 전달되는 계층 데이터를 수정하려면 노멀라이저 보기에서 필드를 플랫폼화합니다.

플랫폼화 작업의 출력은 복합 데이터 유형에 따라 다릅니다. Array 또는 Struct 데이터 유형을 플랫폼화하는 경우 노멀라이저 변환은 복합 데이터 유형의 각 요소에 대한 행을 생성합니다. Map 데이터 유형을 플랫폼화하는 경우 노멀라이저 변환은 맵 키 및 맵 값 요소에 대한 열 2개를 생성합니다.

중첩된 데이터 유형의 플랫폼화 작업은 첫 번째 수준에서 요소를 추출합니다. 중첩된 데이터 유형을 모든 수준에서 플랫폼화하려면 Developer tool에서 복합 포트 플랫폼화 계층 변환 마법사를 사용합니다. 모두 플랫폼화 옵션은 각 수준에서 요소를 추출하고 기본 데이터 유형의 관계형 데이터를 반환합니다. 계층 변환 마법사에 대한 자세한 내용은 Informatica Big Data Management 사용자 가이드를 참조하십시오.

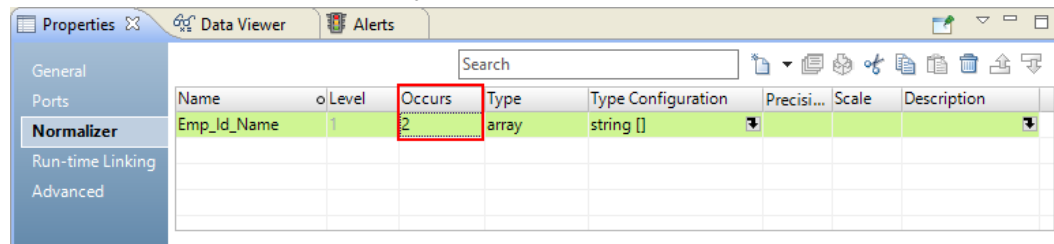
플랫화 작업은 노멀라이저 보기에서 **Occurs** 열의 값을 **Auto**로 변경하고 플랫화된 필드 옆에 플랫화 아이콘을 추가합니다. **Auto** 값은 변환이 복합 데이터 유형의 모든 요소를 플랫화함을 나타냅니다.

다음 이미지에는 문자열 필드로 플랫화된 구조체가 나와 있습니다. 플랫화 아이콘이 옆에 표시되어 있고 **Occurs** 값은 **Auto**로 변경되었습니다.



다중 발생 필드는 플랫화할 수 없습니다. 예를 들어 **Occurs** 값이 2인 배열 필드는 플랫화할 수 없습니다.

다음 이미지에는 플랫화할 수 없는 **Array** 데이터 유형의 다중 발생 필드가 나와 있습니다.



## 배열 플랫화

노멀라이저 변환은 1차원 배열을 기본 데이터 유형으로 플랫화하고 n차원 배열을 (n-1)차원 배열로 플랫화합니다. 변환을 통해 생성되는 행 수는 배열 크기와 동일합니다.

예를 들어 10개의 문자열 요소가 있는 배열 포트를 플랫화하는 경우 출력은 10개의 문자열 포트를 반환합니다. 3차원 배열을 플랫화하는 경우 출력은 2차원 배열을 반환합니다.

테이블에 문자열 포트인 **Name**과 배열 포트인 **Phones**가 있습니다. 이 배열 포트를 플랫화하려고 합니다. 테이블에 포함된 값은 다음과 같습니다.

Name	Phones
Adams	[205-128-6478, 722-515-2889, 650-213-4020]
Jane	[650-321-4506]

배열 포트를 플랫화하는 경우 출력은 다음과 같습니다.

Name	Phones	GCID_Phones
Adams	205-128-6478	1
Adams	722-515-2889	2
Adams	650-213-4020	3
Jane	650-321-4506	1

플랫화된 필드의 **Occurs** 값을 편집하여 배열에 포함된 특정 수의 요소를 추출할 수 있습니다. 값은 1보다 큰 양의 정수여야 합니다. 이 값은 추출할 요소의 수를 결정합니다. 예를 들어 **Occurs** 값을 2로 변경하여 배열의 첫 번째 2개 요소를 추출할 수 있습니다. 출력은 다음과 같습니다.

Name	Phones	GCID_Phones
Adams	205-128-6478	1
Adams	722-515-2889	2
Jane	650-321-4506	1

## 구조체 플랫화

변환은 구조체를 구조체 요소의 데이터 유형 필드로 플랫화합니다. **Struct** 데이터 유형을 플랫화하려면 모든 구조체 요소의 데이터 유형이 동일해야 합니다. 변환은 **Struct** 데이터 유형의 각 요소에 대한 행을 생성합니다.

예를 들어 다음 구조체 필드를 플랫화하려고 합니다.

```
customer_address{
  city : string
  state : string
  zip : string
}
```

테이블에 포함된 값은 다음과 같습니다.

Name	customer_address
Clara	{       New York       NY       10032     }

구조체 포트를 플랫화하는 경우 출력은 다음과 같습니다.

Name	customer_address	GCID_customer_address
Clara	New York	1
Clara	NY	2
Clara	10032	3

구조체 요소의 데이터 유형이 다르고 처음 2개 이상의 요소에 대한 데이터 유형이 동일한 경우 데이터 유형이 동일한 연속적인 요소에 대한 구조체 데이터를 플랫화할 수 있습니다. 동일한 데이터 유형의 연속적인 구조체 요소를 추출하려면 **Occurs** 값을 편집합니다. 값은 1보다 큰 양의 정수여야 합니다. 예를 들어 **emp\_address** 구조체에 다음과 같은 요소가 포함되어 있습니다.

```
emp_address{
  city : string
  state : string
  zip : int
  country : string
}
```

**Occurs** 값을 2로 정의하여 **City** 및 **State** 구조체 요소를 추출할 수 있습니다. 값을 3 또는 4로 정의하면 매핑 유효성 검사가 실패합니다.



## 맵 플랫폼화

변환은 맵의 키 및 값 요소에 대한 2개 필드로 맵을 플랫폼화합니다. 플랫폼화한 맵 필드의 경우 Occurs 값을 Auto에서 정수 값으로 변경할 수 없습니다.

예를 들어 다음 emp\_sal 맵 필드를 문자열 키 및 정수 값의 배열로 플랫폼화하려고 합니다.

```
<emp_name -> [base_sal, bonus, commision]>
```

다음 이미지에는 노멀라이저 보기에서 플랫폼화하려는 맵 필드가 나와 있습니다.

Name	oLevel	Occurs	Type	Type Configuration	Precisi...	Scale	Description
emp_id	1	1	string	N/A	10	0	
emp_sal	1	1	map	< string, string [] >			

테이블에 포함된 값은 다음과 같습니다.

emp_id	emp_sal
12200	<Greg -> [4000, 1000, 500]>
12201	<Patricia -> [3800, 1500, 1000]>

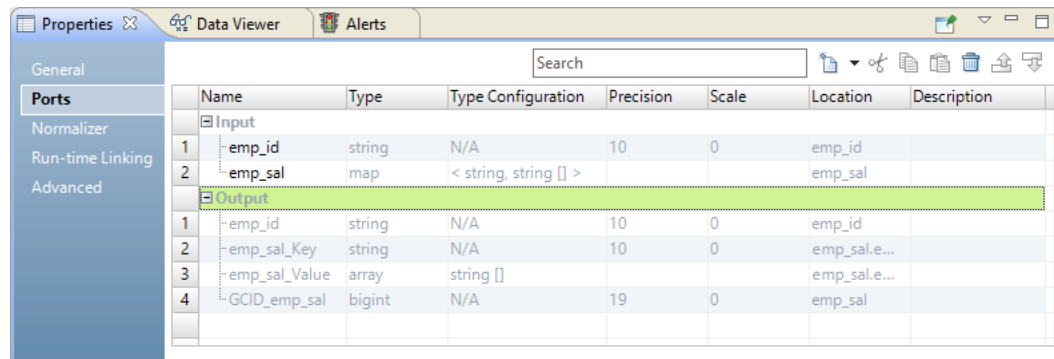
맵 포트를 플랫폼화하는 경우 출력은 맵 키에 대한 문자열 필드와 맵 값에 대한 배열 필드를 다음과 같이 반환합니다.

emp_id	emp_sal_Key	emp_sal_Value	GCID_emp_salary
12200	Greg	[4000, 1000, 500]	1
12201	Patricia	[3800, 1500, 1000]	1

다음 이미지는 노멀라이저 보기에서 문자열 키 필드 및 배열 값 필드로 플랫폼화된 맵 필드를 보여 줍니다.

Name	oLevel	Occurs	Type	Type Configuration	Precisi...	Scale	Description
emp_id	1	1	string	N/A	10	0	
emp_sal	1	Auto		N/A			
emp_sal_Key	2	1	string	N/A	10	0	
emp_sal_Value	2	1	array	string []			

다음 이미지는 포트 보기의 출력 그룹을 보여 줍니다.



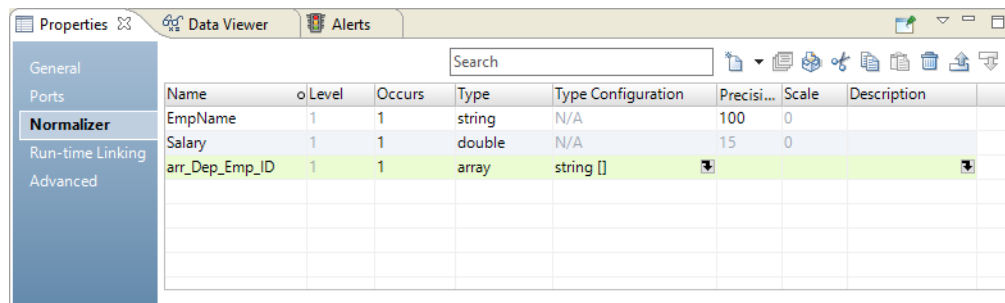
	Name	Type	Type Configuration	Precision	Scale	Location	Description
<b>Input</b>							
1	emp_id	string	N/A	10	0	emp_id	
2	emp_sal	map	< string, string [] >			emp_sal	
<b>Output</b>							
1	emp_id	string	N/A	10	0	emp_id	
2	emp_sal_Key	string	N/A	10	0	emp_sal.e...	
3	emp_sal_Value	array	string []			emp_sal.e...	
4	GCID_emp_sal	bigint	N/A	19	0	emp_sal	

## 필드 플랫폼화

복합 데이터 유형의 필드를 플랫폼화하여 계층 데이터를 수정하거나 관계형 데이터로 변환할 수 있습니다.

1. 노멀라이저 보기를 클릭합니다.
2. 복합 데이터 유형의 필드를 선택합니다.

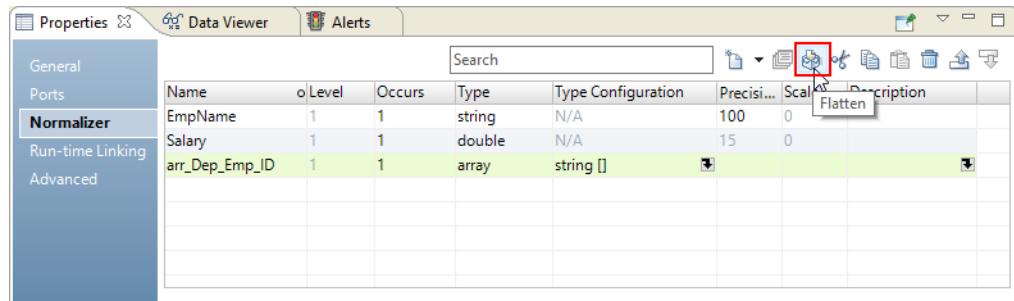
다음 이미지는 문자열 요소를 포함하는 배열 유형의 필드를 보여 줍니다.



	Name	o Level	Occurs	Type	Type Configuration	Precisi...	Scale	Description
	EmpName	1	1	string	N/A	100	0	
	Salary	1	1	double	N/A	15	0	
	arr_Dep_Emp_ID	1	1	array	string []			

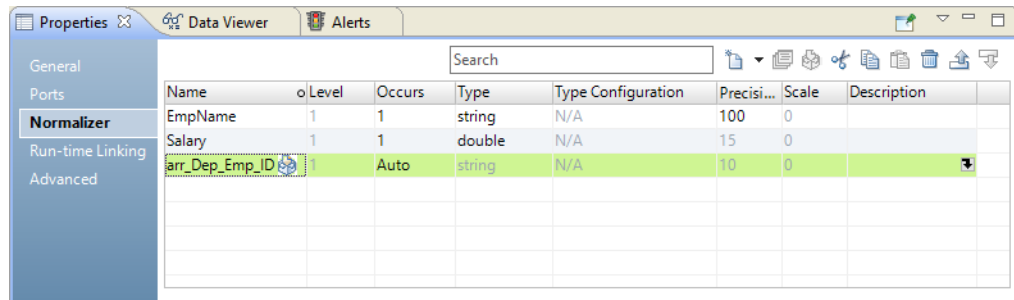
### 3. 플랫폼 단추를 클릭합니다.

다음 이미지는 플랫폼 단추를 보여 줍니다.

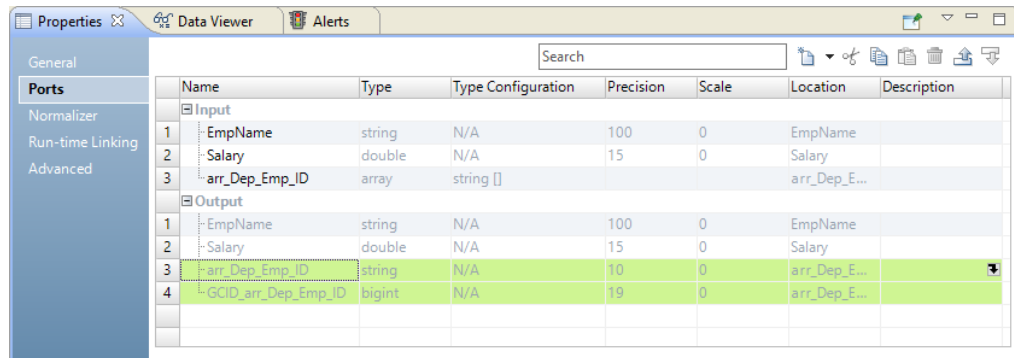


플랫화 작업은 복합 데이터 유형의 필드를 플랫화된 필드로 바꾸고 **Occurs** 값을 Auto로 변경합니다. 플랫화된 필드의 데이터 유형은 플랫화한 복합 데이터 유형에 따라 결정됩니다.

다음 이미지는 문자열 유형의 플랫화된 필드를 보여 줍니다.



다음 이미지는 **포트** 보기에 표시된 플랫화된 문자열 출력 포트와 GCID 출력 포트를 보여 줍니다.



## 노멀라이저 변환 출력 그룹 및 포트

노멀라이저 변환의 개요 보기에서 출력 그룹 및 포트를 정의합니다. 변환 입력 계층을 정의한 후에 출력 그룹을 정의할 수 있습니다.

Developer tool은 기본적으로 출력 그룹을 하나 이상 생성합니다. 출력 그룹은 입력 포트의 모든 수준 1 필드와 첫 번째 여러 번 발생하는 필드를 포함합니다. 노멀라이저 보기에서 여러 번 발생하는 필드를 두 개 이상 정의하면 Developer tool은 각각의 추가 여러 번 발생하는 필드에 대해 출력 그룹을 생성합니다. 첫 번째 수준 출력 그룹 생성 고급 속성을 비활성화하여 Developer tool이 추가 출력 그룹 생성을 중지하도록 할 수 있습니다.

다음 소스 행에는 고객 데이터, 여러 번 발생하는 Sales 필드 및 여러 번 발생하는 Phone 필드가 포함되어 있습니다.

```
CustomerID
LastName
FirstName
Sales (occurs 4 times)
Phone (occurs 3 times)
```

Developer tool은 입력 구조에서 출력 그룹 두 개를 생성합니다.

```
Output
CustomerID
LastName
FirstName
Sales

Output1
Phone
GCID_Phone
```

소스 데이터에 여러 번 발생하는 필드가 둘 이상 포함되어 있으므로 Developer tool은 Output1 그룹을 생성합니다. 레코드에서 정의하는 필드에 대해서는 그룹이 생성되지 않습니다. 레코드를 정의할 때는 레코드의 필드를 포함하는 출력 그룹을 정의해야 합니다.

다음 소스 행은 두 번 발생하는 Address 레코드와 Customer 필드를 포함합니다.

```
CustomerID
  FirstName
  LastName
  Address
    Street
    City
    State
    Country
  Address1
    Street1
    City1
    State1
    Country1
```

Developer tool은 다음 필드를 포함하는 출력 그룹을 생성합니다.

```
CustomerID
FirstName
LastName
```

Developer tool은 수준 1 Customer 필드에 대한 기본 출력 그룹을 생성합니다. 기본 출력 그룹에는 Address 레코드가 포함되지 않습니다. 출력에서 Address 데이터를 반환할 방법을 구성해야 합니다.

출력 행의 구조를 지정해야 하는 방법에 따라 출력 그룹을 생성합니다. 소스 행에 고객 데이터와 주소 데이터가 포함되어 있으면 Customer 필드에 대해 출력 그룹을 생성할 수 있습니다. 주소 필드에 대해 다른 출력 그룹을 생성할 수 있습니다. 또는 기본 출력 그룹을 업데이트하여 주소 필드를 추가할 수 있습니다. 다음 예제에서는 출력 그룹 구성에 따라 달라지는 출력 결과를 보여 줍니다.

## 출력 그룹 작성

노멀라이저 변환의 개요 보기에서 출력 그룹을 작성합니다.

노멀라이저 변환의 개요 보기를 열면 Developer tool이 입력 계층에서 작성한 기본 그룹이 표시됩니다.

새 출력 그룹을 작성하면 입력 계층의 필드 및 레코드 목록이 대화 상자에 표시됩니다. 그룹에 포함할 필드 또는 레코드를 선택하십시오.

### 출력 그룹 예제

다음 이미지는 새 출력 그룹 대화 상자를 보여 줍니다.

### New Output Group

Select the Normalizer fields to add to the output group.

Name	Occurs	Type	Precision	Scale	
<input type="checkbox"/> CustomerID	1	string	10	0	
<input type="checkbox"/> FirstName	1	string	10	0	
<input type="checkbox"/> LastName	1	string	10	0	
<input checked="" type="checkbox"/> Address	1				
<input checked="" type="checkbox"/> Street	1	string	10	0	
<input checked="" type="checkbox"/> City	1	string	10	0	
<input checked="" type="checkbox"/> State	1	string	10	0	
<input checked="" type="checkbox"/> Country	1	string	10	0	

Address 레코드를 선택하면 Developer tool이 Address 레코드의 필드에 해당하는 출력 포트 그룹을 작성합니다. Output1 그룹에는 Street, City, State 및 Country 포트가 포함됩니다. 출력 그룹의 포트를 변경할 수 있습니다.

다음 이미지는 개요 보기의 출력 그룹 및 Output1 그룹을 보여 줍니다.

Output					
1	CustomerID	string	10	0	CustomerID
2	FirstName	string	10	0	FirstName
3	LastName	string	10	0	LastName
Output1					
1	Street	string	10	0	Address.Street
2	State	string	10	0	Address.State
3	Country	string	10	0	Address.Country
4	GCID_Address	bigint	19	0	Address

Output 그룹의 행을 Customer 테이블로 반환하도록 노멀라이저 변환을 구성할 수 있습니다.

Customer 테이블은 다음 행과 유사한 데이터를 수신합니다.

```
100, Robert, Bold
200, James, Cowan
```

Output1 그룹의 행을 Address 테이블로 반환할 수 있습니다. Address 테이블은 Street, City, State, Country 및 GCID를 수신합니다.

Address 테이블은 다음 행과 유사한 데이터를 수신합니다.

```
100 Summit Dr, Redwood City, CA, United States,1
41 Industrial Way, San Carlos, CA, United States,2
85 McNulty Way, Los Angeles, CA, United States,1
55 Factory Street, Los Vegas, NV, United States,2
```

GCID는 출력 행에 있는 고객 주소의 인스턴스를 식별합니다. 이 예에서 노멀라이저 변환은 Address 레코드의 인스턴스 2개를 반환합니다. 각 출력 행에는 1또는 2의 GCID 값이 포함됩니다.

## 출력 그룹 업데이트

노멀라이저 변환 출력 그룹을 업데이트할 수 있습니다. 그룹에서 필드를 추가하거나 제거할 수 있습니다.

입력 계층을 정의할 때 기본적으로 **Developer tool**은 수준 1 출력 그룹을 생성합니다. 레코드는 그룹에 포함되지 않습니다. 기본 출력 그룹을 업데이트하여 레코드를 추가할 수 있습니다.

출력 그룹을 업데이트하려면 그룹 이름을 선택하고 **새로 만들기 > 그룹 업데이트**를 클릭합니다. **출력 그룹 편집** 대화 상자에는 입력 계층의 필드가 표시됩니다. 그룹에 포함할 필드를 선택합니다.

### 출력 그룹 업데이트 예제

위의 예제에서 **Developer tool**은 CustomerID, FirstName 및 LastName 필드가 포함된 기본 출력 그룹을 생성했습니다.

다음 이미지에는 기본 출력 그룹이 나와 있습니다.

Output					
1	CustomerID	string	10	0	CustomerID
2	FirstName	string	10	0	FirstName
3	LastName	string	10	0	LastName

기본 출력 그룹을 업데이트하여 **Address** 레코드를 추가할 수 있습니다.

다음 이미지에는 **출력 그룹 편집** 대화 상자가 나와 있습니다.

### Edit Output Group

Select the Normalizer fields to include in the output group.

Name	Occurs	Type	Precision	Scale
<input checked="" type="checkbox"/> CustomerID	1	string	10	0
<input checked="" type="checkbox"/> FirstName	1	string	10	0
<input checked="" type="checkbox"/> LastName	1	string	10	0
<input checked="" type="checkbox"/> Address	2			
<input checked="" type="checkbox"/> Street	1	string	10	0
<input checked="" type="checkbox"/> State	1	string	10	0
<input checked="" type="checkbox"/> Country	1	string	10	0

이 예제에서 CustomerID, FirstName 및 LastName은 수준 1 노드입니다. Address 레코드도 수준 1 노드입니다. 노멀라이저 변환은 고객 데이터와 같은 행의 Address를 반환할 수 있습니다. Address는 여러 번 발생하므로 **Developer tool**은 출력 그룹에 GCID\_Address 인덱스를 추가합니다.

다음 이미지에는 출력 그룹의 포트가 나와 있습니다.

Output					
1	CustomerID	string	10	0	CustomerID
2	FirstName	string	10	0	FirstName
3	LastName	string	10	0	LastName
4	Street	string	10	0	Address.Street
5	State	string	10	0	Address.State
6	Country	string	10	0	Address.Country
7	GCID_Address	bigint	19	0	Address

Customer 필드와 여러 번 발생하는 Address 필드가 출력 그룹에 있으면 노멀라이저 변환은 각 주소 데이터 인스턴스에 대해 같은 Customer 필드를 반환합니다.

다음 예제에서는 노멀라이저 변환이 출력 그룹에서 생성하는 행을 보여 줍니다.

```
100, Robert, Bold, 100 Summit Dr, Redwood City, CA, United States,1
100, Robert, Bold, 41 Industrial Way, San Carlos, CA, United States,2
200, James, Cowan, 85 McNulty Way, Los Angeles, CA, United States,1
200, James, Cowan, 55 Factory Street, Los Vegas, NV, United States,2
```

GCID 포트는 Address 인스턴스 번호를 포함합니다. GCID 값은 1 또는 2입니다.

## 출력 그룹에 대한 키 생성

시퀀스 생성기 변환을 구성하여 노멀라이저 변환이 동일한 소스 행에서 반환하는 각 출력 행을 연결하는 키를 생성할 수 있습니다.

매핑에서 노멀라이저 변환 앞에 시퀀스 생성기 변환을 추가할 수 있습니다. 시퀀스 생성기 변환은 각 소스 행에 시퀀스 번호를 추가합니다. 노멀라이저 변환이 동일한 소스 행에서 여러 개의 출력 그룹 또는 행을 반환하는 경우 각 출력 행에는 동일한 시퀀스 번호가 추가됩니다. 대상 테이블 간의 기본 키-외래 키 관계에서 이 번호를 키로 사용할 수 있습니다.

예를 들어 노멀라이저 변환이 한 출력 그룹에서 고객 정보를 반환하고 다른 출력 그룹에서 주문 정보를 반환하는 경우 시퀀스 번호를 사용하여 한 테이블의 고객 정보를 다른 테이블의 주문 정보에 연결할 수 있습니다.

## 노멀라이저 변환 고급 속성

**고급** 탭에서 노멀라이저 변환에 대한 속성을 구성합니다.

**고급** 탭에서 다음 속성을 구성합니다.

### 첫 번째 수준의 출력 그룹을 자동으로 생성

노멀라이저 변환에 여러 번 발생하는 그룹이 둘 이상 포함되어 있으면 출력 그룹을 자동으로 생성합니다.

### 추적 수준

이 변환에 대해 로그에 표시되는 세부 정보의 양입니다. 간단, 보통, 자세한 정보 표시 초기화 또는 자세한 정보 표시 데이터를 선택할 수 있습니다. 기본값은 보통입니다.

## 첫 번째 수준의 출력 그룹 생성

Developer tool에서 기본적으로 노멀라이저 변환 첫 번째 수준의 출력 그룹을 작성합니다. 사용자가 노멀라이저 보기에서 다중 발생 필드를 하나 이상 정의할 경우 Developer tool에서 추가 출력 그룹을 작성합니다.

Developer tool에서 모든 수준 1 단일 발생 필드와 첫 번째 수준 1 다중 발생 필드를 포함하는 출력 그룹을 작성합니다. 변환에 다중 발생 필드가 둘 이상 포함된 경우 Developer tool에서 추가 출력 그룹을 작성합니다.

Developer tool에서 수준 1 레코드에 대한 출력 그룹을 작성하지 않습니다. 수준 1 레코드를 정의할 경우 출력에서 배치할 위치를 구성해야 합니다.

Developer tool이 다중 발생 필드에 대한 출력 그룹을 작성하지 못하도록 중지하려면 **첫 번째 수준의 출력 그룹을 자동으로 생성** 고급 속성을 비활성화하십시오.

## 노멀라이저 변환 작성

재사용 가능 또는 재사용 불가능한 노멀라이저 변환을 작성할 수 있습니다. 재사용 가능 변환은 여러 매핑에 존재할 수 있습니다. 재사용 불가능 변환은 단일 매핑에만 존재합니다.

1. 변환을 작성하려면 다음 방법 중 하나를 사용합니다.

옵션	설명
재사용 가능	<b>개체 탐색기</b> 보기에서 프로젝트나 폴더를 선택합니다. <b>파일 &gt; 새로 만들기 &gt; 변환</b> 을 클릭합니다. 노멀라이저 변환을 선택하고 <b>다음</b> 을 클릭합니다.
재사용 불가능	매핑 또는 맵렛에서 <b>변환</b> 색상표의 노멀라이저 변환을 편집기로 끌어 옵니다. 소스 데이터 개체 또는 매핑의 변환에서 <b>포트</b> 를 끌어 변환을 정의할 수 있습니다.

**새 노멀라이저 변환** 마법사가 표시됩니다.

2. 변환 이름을 입력합니다.

3. **다음**을 클릭합니다.

**노멀라이저 정의** 페이지가 표시됩니다.

4. 레코드를 추가하려면 **새로 만들기** 단추를 클릭한 다음 **레코드**를 선택합니다.

5. 필드를 추가하려면 **새로 만들기** 단추를 클릭한 다음 **필드**를 선택합니다.

필드를 레코드에 추가하려면 필드를 추가하기 전에 레코드를 선택해야 합니다.

6. 필요한 경우 **발생** 열의 값을 두 번 클릭하여 필드 또는 레코드의 발생을 변경할 수 있습니다.

7. **다음**을 클릭합니다.

**노멀라이저 포트** 페이지가 표시됩니다.

8. 출력 그룹을 추가하려면 **새로 만들기** 단추를 클릭한 다음 **새 출력 그룹**을 선택합니다.

9. 출력 그룹을 편집하려면 편집할 출력 그룹을 선택합니다. **새로 만들기** 단추를 클릭한 다음 **출력 그룹 편집**을 선택합니다.

10. **마침**을 클릭합니다.

변환이 편집기에 표시됩니다.

## 업스트림 소스에서 노멀라이저 변환 작성

빈 노멀라이저 변환을 작성하고 소스 데이터 개체 또는 변환에서 노멀라이저 변환으로 포트를 끌어서 입력 및 포트를 작성할 수 있습니다.

1. 소스 데이터를 노멀라이저 변환으로 전달하려면 소스 또는 변환을 포함하는 매핑을 작성하십시오.

2. 노멀라이저 변환을 작성하려면 변환 팔레트에서 노멀라이저 변환을 선택하고 변환을 편집기로 끄십시오. 노멀라이저 대화 상자가 표시됩니다.

3. **마침**을 클릭하여 비어 있는 변환을 작성합니다.

4. 매핑의 소스 또는 변환에서 포트를 선택하고 해당 포트를 노멀라이저 변환으로 끕니다.

입력 및 출력 포트가 노멀라이저 변환에 표시됩니다. **Developer tool**에서 입력 그룹과 출력 그룹을 하나씩 작성합니다.



5. **노멀라이저** 보기를 열어서 기본 그룹을 업데이트하고 필요에 따라 필드를 레코드로 구성합니다.
6. 여러 필드를 하나의 다중 발생 필드로 병합하려면 **노멀라이저** 보기에서 필드를 선택하고 **병합** 옵션을 클릭합니다.  
다중 발생 필드의 이름을 선택합니다.

## 노멀라이저 매핑 예제

소매업체가 각 매장의 총 판매액 데이터를 받는다고 가정해 보겠습니다. 해당 업체는 매장 정보와 판매액 4개가 포함된 데이터 행을 받습니다. 각 판매액은 연도 내 분기당 총 판매액을 나타냅니다.

다음 예제에서는 **Store** 대상 및 **Sales** 대상으로 판매 데이터를 반환하도록 노멀라이저 변환을 정의하는 방법을 보여 줍니다. **Store** 대상은 각 매장에 대해 행 하나를 받습니다. **Sales** 대상은 각 매장에서 행 4개를 받습니다. 각 행에는 단일 분기의 판매 데이터가 포함됩니다.

시퀀스 생성기 변환은 각 매장에 대해 고유한 ID를 생성합니다. 노멀라이저 변환은 각 출력 행에 **StoreID**를 반환합니다.

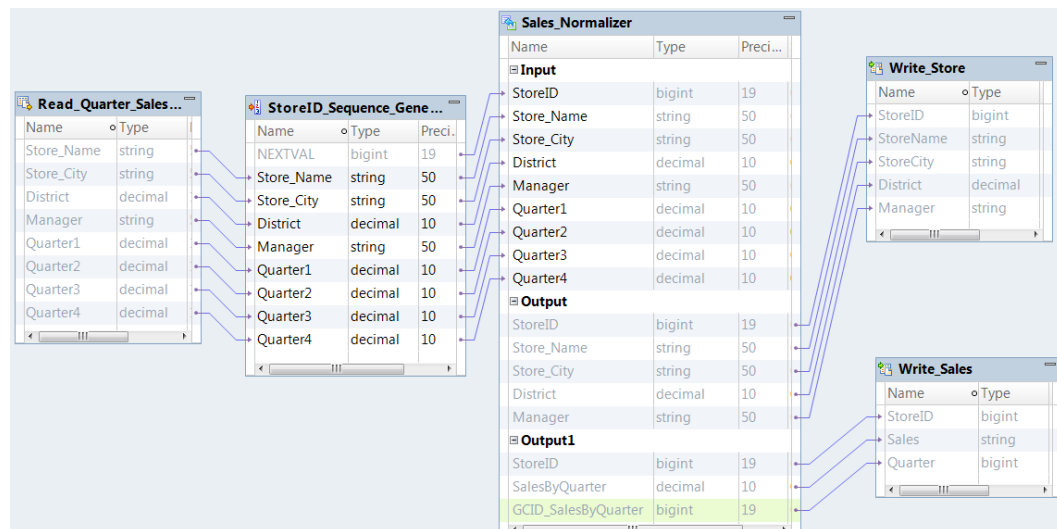
읽기 변환, 시퀀스 생성기 변환, 노멀라이저 변환 및 쓰기 변환(2개)을 사용하여 매핑을 생성합니다.

## 노멀라이저 예제 매핑

플랫 파일 소스에서 여러 번 발생하는 분기별 판매 데이터를 정규화하려면 노멀라이저 변환이 포함된 매핑을 생성합니다.

노멀라이저 변환은 각 판매 분기에 대해 개별 출력 행을 생성하고 정규화된 판매액을 **Sales** 대상에 씁니다. 노멀라이저 변환은 **Store** 정보를 **Store** 대상에 씁니다.

다음 그림에는 노멀라이저 변환 매핑이 나와 있습니다.



이 매핑에는 다음 개체가 포함됩니다.

**Read\_STORE**

여러 번 발생하는 필드가 포함된 데이터 소스입니다.

### StoreID 시퀀스 생성기 변환

Store 테이블을 Sales 테이블에 연결하기 위한 storeID 키를 생성하는 시퀀스 생성기 변환입니다.

### Sales\_Normalizer

여러 번 발생하는 판매 데이터를 정규화하는 노멀라이저 변환입니다.

### Write\_Store

노멀라이저 변환에서 매장 정보를 받는 대상입니다.

### Write\_Sales

노멀라이저 변환에서 판매액 값을 받는 대상입니다.

## 노멀라이저 예제 정의

소스는 매장 정보와 분기별 판매액 데이터를 포함하는 플랫폼 파일입니다. **노멀라이저** 보기에서 소스 데이터의 구조를 정의합니다.

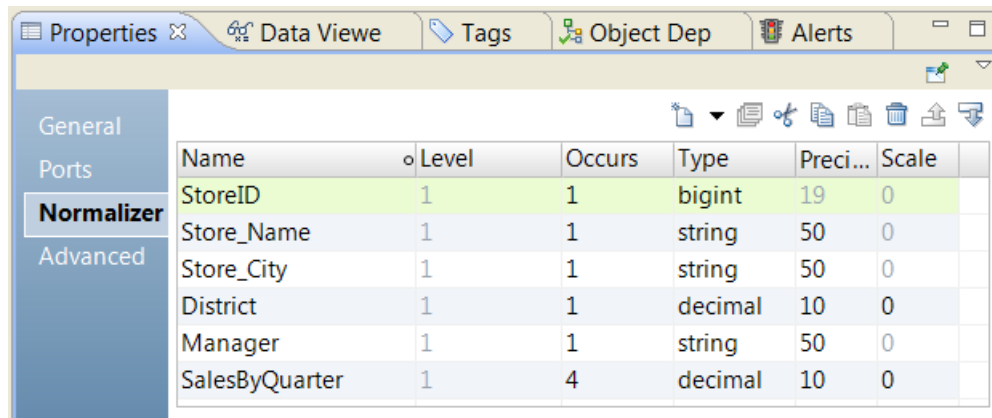
STORE 플랫폼 파일은 다음 소스 데이터를 포함합니다.

StoreID	Store_Name	Store_City	지구	Manager	Quarter1	Quarter2	Quarter3	Quarter4
1	BigStore	뉴욕	동부	Robert	100	300	500	700
2	SmallStore	Phoenix	서부	Radhika	250	450	650	850

플랫폼 파일을 매핑에 읽기 변환으로 추가한 다음 빈 노멀라이저 변환을 생성합니다. Read\_STORE 데이터 개체에서 노멀라이저 변환으로 포트를 끌어 노멀라이저 정의를 생성합니다.

노멀라이저 보기에는 Store\_Name, Store\_City, District 및 Manager 필드의 인스턴스가 하나씩 포함됩니다. 노멀라이저 보기에는 QUARTER라는 필드의 인스턴스 4개가 포함됩니다. QUARTER 필드를 병합하여 4번 발생하는 SalesByQuarter 필드 하나를 생성합니다.

다음 그림에는 Quarter 필드가 병합된 노멀라이저 정의가 나와 있습니다.



Name	Level	Occurs	Type	Preci...	Scale
StoreID	1	1	bigint	19	0
Store_Name	1	1	string	50	0
Store_City	1	1	string	50	0
District	1	1	decimal	10	0
Manager	1	1	string	50	0
SalesByQuarter	1	4	decimal	10	0

## 노멀라이저 예제 입력 및 출력 그룹

입력 계층을 수정하고 나면 노멀라이저 변환에는 입력 그룹과 기본 출력 그룹이 하나씩 포함됩니다. 출력 포트를 두 그룹으로 다시 구성해야 합니다. Store 정보가 포함된 그룹과 Sales 정보가 포함된 그룹이 하나씩 필요합니다.

입력 그룹은 소스의 각 필드에 대한 포트를 포함합니다. 출력 그룹은 여러 번 발생하는 SalesByQuarter 필드에 대한 포트와 Store 필드에 대한 포트를 포함합니다. 출력 그룹에는 여러 번 발생하는 SalesByQuarter 필드에 해당하는 생성된 열 ID인 GCID\_SalesByQuarter도 포함됩니다.

분기별 판매액을 다른 대상으로 반환하려면 **개요** 보기에서 새 그룹을 생성합니다. **Output1** 그룹에서 다음 필드를 추가합니다.

StoreID  
SalesByQuarter  
GCID\_SalesByQuarter

기본 출력 그룹을 업데이트합니다. 다음 필드를 제거합니다.

SalesByQuarter  
GCID\_SalesByQuarter

다음 이미지에는 **개요** 보기의 입력 그룹과 출력 그룹이 나와 있습니다.

	Name	Type	Precision	Scale	Location
<b>Input</b>					
1	StoreID	bigint	19	0	StoreID
2	Store_Name	string	50	0	Store_N...
3	Store_City	string	50	0	Store_City
4	District	decimal	10	0	District
5	Manager	string	50	0	Manager
6	Quarter1	decimal	10	0	SalesBy...
7	Quarter2	decimal	10	0	SalesBy...
8	Quarter3	decimal	10	0	SalesBy...
9	Quarter4	decimal	10	0	SalesBy...
<b>Output</b>					
1	StoreID	bigint	19	0	StoreID
2	Store_Name	string	50	0	Store_N...
3	Store_City	string	50	0	Store_City
4	District	decimal	10	0	District
5	Manager	string	50	0	Manager
<b>Output1</b>					
1	StoreID	bigint	19	0	StoreID
2	SalesByQuarter	decimal	10	0	SalesBy...
3	GCID_SalesByQuarter	bigint	19	0	SalesBy...

StoreID는 Store 정보를 Sales 정보와 연결하는 생성된 키입니다. 두 출력 그룹이 모두 StoreID를 반환하는지 확인합니다.

## 노멀라이저 예제 매핑 출력

매핑에 쓰기 변환을 추가하고 해당 데이터 개체에 노멀라이저 변환 출력 포트를 연결합니다.

매핑을 실행하면 노멀라이저 변환이 다음 행을 Store 대상에 씁니다.

StoreID	Store_Name	Store_City	지구	Manager
1	BigStore	뉴욕	동부	Robert
2	SmallStore	Phoenix	서부	Radhika

노멀라이저 변환은 다음 행을 Sales 대상에 씁니다.

StoreID	SalesByQuarter	GCID_SalesByQuarter
1	100	1
1	300	2

StoreID	SalesByQuarter	GCID_SalesByQuarter
1	500	3
1	700	4
2	250	1
2	450	2
2	650	3
2	850	4

## 노멀라이저 변환 - 비원시 환경

비원시 환경에서 노멀라이저 변환의 처리는 변환을 실행하는 엔진에 따라 다릅니다.

다음과 같은 비원시 런타임 엔진에 대한 지원을 고려하십시오.

- **Blaze** 엔진. 제한 없이 지원됩니다.
- **Spark** 엔진. 제한 없이 지원됩니다.
- **Databricks Spark** 엔진. 제한 없이 지원됩니다.

## 제 33 장

# 파서 변환

이 장에 포함된 항목:

- [파서 변환 개요, 497](#)
- [파서 변환 모드, 498](#)
- [파서 변환 사용 시기, 498](#)
- [파서 변환에서 참조 데이터 사용, 499](#)
- [토큰 구문 분석 작업, 501](#)
- [토큰 구문 분석 포트, 501](#)
- [토큰 구문 분석 속성, 502](#)
- [패턴 기반 구문 분석 모드, 505](#)
- [토큰 구문 분석 전략 구성, 505](#)
- [패턴 구문 분석 전략 구성, 506](#)
- [파서 변환 고급 속성, 507](#)
- [파서 변환 - 비원시 환경, 507](#)

## 파서 변환 개요

파서 변환은 입력 데이터 값을 새 포트로 구문 분석하는 수동 변환입니다. 변환은 값이 포함하는 정보 유형 및 입력 문자열의 값 위치에 따라 새 포트에 값을 씁니다.

데이터 집합 구조를 변경하려는 경우 파서 변환을 사용합니다. 파서 변환은 열을 데이터 집합에 추가하고 데이터 값을 새 열에 쓸 수 있습니다. 데이터 열의 단일 열에 다중 값이 포함되어 있고 포함된 정보 유형에 따라 개별 열에 데이터 값을 쓰려는 경우 파서 변환을 사용합니다.

파서 변환은 데이터 값을 정의된 출력 포트로 구문 분석합니다. 변환이 입력 데이터 값을 식별할 수 있지만 정의된 출력 포트를 사용할 수 없는 경우 변환이 오버플로우 포트에 값을 씁니다. 변환이 입력 데이터 값을 식별할 수 없는 경우 구문 분석되지 않은 데이터 포트에 값을 씁니다.

## 파서 변환 모드

파서 변환을 작성할 경우 토큰 구문 분석 모드 또는 패턴 기반 구문 분석 모드를 선택하십시오.

다음 모드 중 하나를 선택하십시오.

- 토큰 구문 분석 모드. 토큰 집합, 정규식, 확률 모델 및 참조 테이블 같은 참조 데이터 개체의 값과 일치하는 입력 값을 구문 분석하려면 이 모드를 사용하십시오. 변환에서 여러 토큰 구문 분석 전략을 사용할 수 있습니다.
- 패턴 기반 구문 분석 모드. 패턴 집합의 값과 일치하는 입력 값을 구문 분석하려면 이 모드를 사용하십시오.

## 파서 변환 사용 시기

열의 데이터 필드에 두 개 이상의 정보 유형이 포함되어 있고 필드 값을 새 열로 이동하려는 경우 파서 변환을 사용합니다. 파서 변환을 통해 데이터 집합의 각 정보 유형에 대해 새 열을 작성할 수 있습니다.

다음 예에서는 파서 변환을 사용하여 수행할 수 있는 일부 구조 변경 유형에 대해 설명합니다.

### 연락처 데이터에 대해 새 열 작성

이름 데이터를 단일 열에서 여러 열로 구문 분석하는 데이터 구조를 작성할 수 있습니다. 예를 들어 호칭, 이름, 중간 이름 및 성에 대한 열을 작성할 수 있습니다.

입력 포트에 사람 이름의 구조를 나타내는 확률 모델을 사용하여 변환을 구성합니다. 입력 포트 데이터 샘플을 사용하여 모델을 정의합니다.

확률 모델을 입력 포트에 적용하고 이름 값을 새 열에 쓰는 토큰 구문 분석 전략을 작성합니다. 변환이 입력 문자열에서 각 값의 위치 및 값이 나타내는 이름 유형에 따라 이름 값을 새 열에 씁니다.

**참고:** 또한 패턴 기반 구문 분석 전략을 사용하여 연락처 데이터를 구문 분석할 수 있습니다. 패턴 기반 구문 분석 전략을 구성하는 경우 입력 포트의 이름 구조를 나타내는 패턴을 정의합니다.

### 주소 열 작성

단일 열의 주소 데이터를 배달 가능 주소를 설명하는 여러 열로 구문 분석하는 데이터 구조를 작성할 수 있습니다.

우편 번호, 시/도 이름, 구/군/시 이름과 같은 인식 가능한 주소 요소가 포함된 참조 테이블을 사용하여 변환을 구성합니다. 각 주소 요소를 새 포트에 쓰는 토큰 구문 분석 전략을 작성합니다.

거리 이름 및 숫자 데이터는 너무 일반적이어서 참조 테이블에서 캡처할 수 없기 때문에 참조 테이블을 사용하여 입력 문자열의 거리 주소 데이터를 구문 분석할 수 없습니다. 그러나 오버플로우 포트를 사용하면 이 데이터를 캡처할 수 있습니다. 주소에서 모든 구/군/시, 시/도 및 우편 번호 데이터를 구문 분석한 경우 나머지 데이터는 거리 정보를 포함합니다.

예를 들어 토큰 구문 분석 전략을 사용하여 다음 주소를 주소 요소로 분할합니다.

123 MAIN ST NW STE 12 ANYTOWN NY 12345

구문 분석 전략은 주소 요소를 다음 열에 쓸 수 있습니다.

열 이름	데이터
오버플로우	123 MAIN ST NW STE 12
도시	ANYTOWN
상태	NY
우편 번호	12345

### 제품 데이터 열 작성

단일 열의 제품 데이터를 제품 인벤토리 세부 정보를 설명하는 여러 열로 구문 분석하는 데이터 구조를 작성할 수 있습니다.

차원, 색상 및 가중치와 같은 인벤토리 요소가 포함된 토큰 집합을 사용하여 변환을 구성합니다. 각 인벤토리 요소를 새 포트에 쓰는 토큰 구문 분석 전략을 작성합니다.

예를 들어 토큰 구문 분석 전략을 사용하여 다음 설명을 개별 인벤토리 요소로 분할합니다.

500ML Red Matt Exterior

구문 분석 전략은 주소 요소를 다음 열에 쓸 수 있습니다.

열 이름	데이터
크기	500ML
색상	빨간색
스타일	Matt
외부	연도

## 파서 변환에서 참조 데이터 사용

Informatica Developer는 파서 변환에서 사용할 수 있는 여러 참조 데이터 개체와 함께 설치됩니다. 또한 Developer tool에서 참조 데이터 개체를 작성할 수 있습니다.

참조 데이터 개체를 파서 변환에 추가하는 경우 변환이 개체의 값과 일치하는 문자열을 사용자가 지정한 새 열에 씁니다.

다음 테이블에는 사용할 수 있는 참조 데이터 유형이 설명되어 있습니다.

참조 데이터 유형	설명
패턴 집합	문자열에서 각 값의 상대적 위치에 따라 데이터 값을 식별합니다.
확률 모델	유사 항목 일치 기능을 토큰 구문 분석 작업에 추가합니다. 변환은 확률 모델을 사용하여 문자열의 정보 유형을 유추할 수 있습니다. 유사 항목 일치 기능을 활성화하려면 Developer tool에서 확률 모델을 컴파일합니다.
참조 테이블	데이터베이스 테이블의 항목과 일치하는 문자열을 찾습니다.
정규식	정의하는 조건과 일치하는 문자열을 식별합니다. 정규식을 사용하면 더 큰 문자열에서 문자열을 찾을 수 있습니다.
토큰 집합	포함된 정보 유형에 따라 문자열을 식별합니다. Informatica는 단어, 전화 번호, 우편 번호 및 제품 코드 정의와 같이 여러 유형의 토큰 정의를 가진 토큰 집합과 함께 설치됩니다.

## 패턴 집합

패턴 집합에는 토큰 레이블 지정 작업의 출력에서 데이터 패턴을 식별하는 식이 포함됩니다. 패턴 집합을 사용하여 토큰화된 데이터 출력 포트를 분석하고 일치하는 문자열을 하나 이상의 출력 포트에 쓸 수 있습니다. 패턴 구문 분석 모드를 사용하는 파서 변환에서 패턴 집합을 사용합니다.

예를 들어 이름과 이니셜을 식별하는 패턴 집합을 사용하도록 파서 변환을 구성할 수 있습니다. 이 변환에서는 패턴 집합을 사용하여 토큰 레이블 지정 모드에서 라벨러 변환의 출력을 분석합니다. 출력의 이름과 이니셜을 별도의 포트에 쓰도록 파서 변환을 구성할 수 있습니다.

## 확률 모델

확률 모델은 포함된 정보 유형 및 입력 문자열을 차지하는 위치에 따라 토큰을 식별합니다.

확률 모델에는 참조 데이터 값과 레이블 값이 포함됩니다. 참조 데이터 값은 변환에 연결하는 입력 포트의 데이터를 나타냅니다. 레이블 값은 참조 데이터 값에 포함되는 정보의 유형을 설명합니다. 사용자는 모델의 각 참조 데이터 값에 레이블을 할당합니다.

확률 모델에서 참조 데이터 값을 레이블에 연결하려면 모델을 컴파일합니다. 컴파일 프로세스에서는 데이터 값과 레이블 간의 일련의 논리적 연관이 생성됩니다. 모델을 읽는 매핑을 실행하면 데이터 통합 서비스가 모델 논리를 변환 입력 데이터에 적용합니다. 데이터 통합 서비스는 입력 데이터 값을 가장 정확히 설명하는 레이블을 반환합니다.

Developer tool에서 확률 모델을 작성할 수 있습니다. 모델 리포지토리는 확률 모델 개체를 저장합니다. Developer tool은 데이터 값, 레이블 및 컴파일 데이터를 Informatica 디렉터리 구조의 파일에 씁니다.

**참고:** 확률 모델을 토큰 구문 분석 작업에 추가한 다음 확률 모델에서 레이블 구성을 편집하는 경우 작업이 무효화됩니다. 확률 모델에서 레이블 구성을 업데이트할 때는 해당 모델을 사용하는 구문 분석 작업을 다시 생성하십시오.

## 참조 테이블

참조 테이블은 두 개 이상의 열이 포함된 데이터베이스 테이블입니다. 한 열에는 표준 또는 필요한 버전의 데이터 값이 들어 있고 다른 열에는 대체 버전 값이 들어 있습니다. 참조 테이블을 변환에 추가하면 변환에서 입력 포트 데이터를 검색하여 참조 테이블에도 나타나는 값을 찾습니다. 작업하는 데이터 프로젝트에 유용한 모든 데이터를 사용하여 테이블을 작성할 수 있습니다.



## 정규식

구문 분석 작업의 컨텍스트에서 정규식은 입력 데이터에서 하나 이상의 문자열을 식별하는 데 사용할 수 있는 식입니다. 파서 변환은 식별된 문자열을 하나 이상의 출력 포트에 씁니다. 토큰 구문 분석 모드를 사용하는 파서 변환에서 정규식을 사용할 수 있습니다.

파서 변환에서는 정규식을 사용하여 입력 데이터에서 패턴을 일치시키고 모든 일치하는 문자열을 하나 이상의 출력으로 구문 분석합니다. 예를 들어 정규식을 사용하여 입력 데이터에서 모든 전자 메일 주소를 식별하고 각각의 전자 메일 주소 구성 요소를 서로 다른 출력으로 구문 분석할 수 있습니다.

## 토큰 집합

토큰 집합에는 특정 토큰을 식별하는 식이 포함됩니다. 토큰 구문 분석 모드를 사용하는 파서 변환에서 토큰 집합을 사용할 수 있습니다.

토큰 집합을 사용하여 구문 분석 작업의 일부로 특정 토큰을 식별할 수 있습니다. 예를 들어 토큰 집합을 사용하여 "AccountName@DomainName" 형식을 사용하는 모든 전자 메일 주소를 구문 분석할 수 있습니다.

## 토큰 구문 분석 작업

토큰 구문 분석 모드에서 파서 변환은 토큰 집합, 정규식, 확률 모델 또는 참조 테이블 항목의 데이터와 일치하는 문자열을 구문 분석합니다.

토큰 구문 분석을 수행하려면 변환의 **전략** 보기에서 전략을 추가합니다. 각 전략에 작업을 하나 이상 추가할 수 있습니다. 변환에서는 전략을 생성하는 데 사용할 수 있는 마법사를 제공합니다.

다음과 같은 유형의 작업을 토큰 구문 분석 전략에 추가할 수 있습니다.

### 토큰 집합을 사용하여 구문 분석

미리 정의된/사용자 정의 토큰 집합을 사용하여 입력 데이터 구문을 분석합니다. 토큰 집합 작업에서는 하나 이상의 출력에 데이터를 쓰는 사용자 지정 정규식을 사용할 수 있습니다.

확률 모델을 사용하여 입력 데이터 값을 식별하고 구문 분석할 수도 있습니다.

### 참조 테이블을 사용하여 구문 분석

참조 테이블을 사용하여 입력 데이터 구문을 분석합니다.

변환은 작업을 전략에 표시되는 순서대로 수행합니다.

## 토큰 구문 분석 포트

데이터에 적합한 설정으로 토큰 구문 분석 포트를 구성합니다.

토큰 구문 분석 모드의 파서 변환에는 다음과 같은 포트 유형이 있습니다.

### 입력

파서 변환으로 전달하는 데이터를 포함합니다. 변환은 **전략** 탭에 지정된 **입력 조인 문자**를 사용하여 모든 입력 포트를 결합된 데이터 문자열로 병합합니다. 입력 조인 문자를 지정하지 않으면 변환은 기본적으로 공백 문자를 사용합니다.

### 구문 분석된 출력 포트

정상적으로 구문 분석된 문자열을 포함하는 사용자 정의 출력 포트입니다. 여러 구문 분석 전략이 같은 출력을 사용하는 경우 변환은 **전략** 탭에 지정된 **출력 조인 문자**를 사용하여 출력을 결합된 데이터 문자열로 병합합니다. 출력 조인 문자를 지정하지 않으면 변환은 기본적으로 공백 문자를 사용합니다.

### 오버플로우

변환에 정의된 출력 수에 맞지 않는 정상적으로 구문 분석된 문자열을 포함합니다. 예를 들어 변환에 "WORD" 출력 두 개만 포함되어 있는 경우 "John James Smith" 문자열이 있으면 "Smith" 오버플로우 출력이 발생합니다. 파서 변환은 추가하는 각 전략에 대해 오버플로우 포트를 생성합니다.

자세한 오버플로우 옵션을 선택하면 변환이 모델의 각 레이블에 대해 오버플로우 포트를 생성합니다.

### 구문 분석되지 않음

변환에서 정상적으로 구문 분석할 수 없는 문자열을 포함합니다. 파서 변환은 추가하는 각 전략에 대해 구문 분석되지 않음 포트를 생성합니다.

### 확률 일치의 출력 포트

확률 일치 기술을 사용하도록 구문 분석 전략을 구성하면 파서 변환이 각 출력 포트에 대한 일치 점수를 저장할 포트를 추가합니다.

다음 테이블에서는 포트 유형에 대해 설명합니다.

포트 유형	확률 일치에서 생성된 포트
구문 분석된 출력 포트	[레이블 이름] 출력 [레이블 이름] 점수 출력
오버플로우 데이터 포트	[오버플로우 데이터] 출력 [오버플로우 데이터] 점수 출력
구문 분석되지 않은 데이터 포트	[구문 분석되지 않은 데이터] 출력 [구문 분석되지 않은 데이터] 점수 출력

## 토큰 구문 분석 속성

파서 변환의 **전략** 보기에서 토큰 구문 분석 작업에 대한 속성을 구성합니다.

### 일반 속성

전략에서 정의하는 모든 토큰 구문 분석 지정 작업에는 일반 속성이 적용됩니다. 일반 속성을 사용하여 전략 이름, 입력 및 출력 포트, 그리고 전략이 확률 일치 기술을 활성화하는지 여부를 지정합니다.

다음 테이블에는 일반 속성이 설명되어 있습니다.

속성	설명
이름	전략 이름을 제공합니다.
입력	전략 작업이 읽을 수 있는 입력 포트를 식별합니다.

속성	설명
출력	전략 작업이 쓸 수 있는 출력 포트를 식별합니다.
설명	전략에 대한 설명입니다. 이 속성은 선택 사항입니다.
확률 일치 기술 사용	전략이 확률 모델을 사용하여 토큰을 식별할 수 있음을 지정합니다.
입력 조인 문자	입력 데이터 포트를 조인하는 데 사용되는 문자를 지정합니다. 변환에서는 모든 입력 포트를 결합된 데이터 문자열로 병합하고 전체 문자열을 구문 분석합니다.
출력 조인 문자	여러 구문 분석 작업에서 같은 출력을 사용할 때 출력 데이터 값을 조인하는 데 사용되는 문자를 지정합니다.
반전 활성화	데이터를 오른쪽에서 왼쪽으로 구문 분석하도록 전략을 구성합니다. 확률 일치에 대해서는 이 속성이 비활성화됩니다.
오버플로우 반전 활성화	오버플로우 데이터를 오른쪽에서 왼쪽으로 구문 분석하도록 전략을 구성합니다. 확률 일치에 대해서는 이 속성이 비활성화됩니다.
자세한 오버플로우 활성화	각 구문 분석 작업에 대해 고유한 오버플로우 필드를 생성합니다.
구분자	입력 데이터를 개별 토큰으로 구분하는 구분자를 지정합니다. 기본값은 공백입니다.

## 확률 모델 속성

토큰 구문 분석 전략을 구성할 때 토큰 집합 대신 확률 모델을 선택할 수 있습니다. **토큰 집합을 사용하여 구문 분석** 작업을 선택하고 확률 일치 기술을 사용하는 옵션을 선택합니다.

다음 테이블에는 확률 모델 속성이 설명되어 있습니다.

속성	설명
이름	작업 이름을 제공합니다.
필터 텍스트	입력하는 문자 또는 와일드카드를 사용하여 토큰 집합, 확률 모델 또는 정규식 목록을 필터링합니다.
확률 모델	선택한 확률 모델을 식별합니다.

## 참조 테이블 속성

참조 테이블을 사용하도록 레이블 지정 작업을 구성하는 경우 참조 테이블 속성이 적용됩니다.

다음 표에는 참조 테이블 속성이 설명되어 있습니다.

속성	설명
이름	작업 이름을 제공합니다.
참조 테이블	입력 값을 구문 분석하기 위해 작업이 사용하는 참조 테이블을 지정합니다.

속성	설명
대/소문자 구분	입력 문자열이 참조 테이블 항목의 대/소문자와 일치해야 하는지 여부를 결정합니다.
유효한 값으로 일치 항목 대체	구문 분석된 데이터를 참조 테이블의 Valid 열의 데이터로 바꿉니다.
출력	구문 분석된 데이터의 출력 포트를 지정합니다.

## 토큰 집합 속성

토큰 집합을 사용하도록 구문 분석 작업을 구성하는 경우 토큰 집합 속성이 적용됩니다.

**토큰 집합을 사용하여 구문 분석** 작업을 선택하고 토큰 집합을 사용하여 입력을 구문 분석합니다. 확률 일치 기술을 사용하려면 옵션을 지웁니다.

다음 테이블에는 토큰 집합 속성이 설명되어 있습니다.

속성	설명
이름	작업 이름을 제공합니다.
토큰 집합(단일 출력만)	작업이 데이터를 구문 분석하기 위해 사용하는 토큰 집합을 지정합니다. 작업이 데이터를 단일 포트에 씁니다.
정규식(단일 또는 다중 출력)	작업이 데이터를 구문 분석하기 위해 사용하는 정규식을 지정합니다. 입력 필드에서 여러 문자열을 찾는 경우 작업이 데이터를 다중 포트에 씁니다.
출력	작업이 쓰는 출력 포트를 식별합니다.

토큰 집합 또는 정규식을 추가, 편집, 가져오기 또는 제거할 수 있습니다. 토큰 집합의 목록을 필터링할 수도 있습니다.

다음 테이블에는 이러한 태스크를 수행하기 위해 사용하는 속성이 설명되어 있습니다.

속성	설명
필터 텍스트	토큰 집합 또는 정규식의 목록을 필터링합니다. 필터로 텍스트 문자와 와일드카드 문자를 사용합니다.
추가	사용자 지정 토큰 집합 또는 정규식을 정의하는 데 사용합니다.
편집	사용자 지정 토큰 집합의 콘텐츠를 편집합니다.
가져오기	모델 리포지토리의 폴더에서 정규식 또는 토큰 집합의 재사용 불가능한 사본을 가져옵니다. 토큰 집합 또는 정규식의 소스 개체를 업데이트하는 경우 데이터 통합 서비스는 재사용 불가능한 사본을 업데이트하지 않습니다.
제거	사용자 지정 토큰 집합 또는 정규식을 삭제합니다.

## 패턴 기반 구문 분석 모드

패턴 기반 구문 분석 모드에서 파서 변환은 여러 문자열로 이루어진 패턴을 구문 분석합니다.

다음 방법을 사용하여 패턴 기반 구문 분석 모드에서 패턴을 정의할 수 있습니다.

- 참조 테이블에 정의된 패턴을 사용하여 입력 데이터를 구문 분석합니다. 토큰 레이블 지정 모드를 사용하는 라벨러 변환의 프로파일링된 출력에서 패턴 참조 테이블을 작성할 수 있습니다.
- 정의하는 패턴을 사용하여 입력 데이터를 구문 분석합니다.
- 모델 리포지토리의 재사용 가능 패턴 집합에서 가져오는 패턴을 사용하여 입력 데이터를 구문 분석합니다. 재사용 가능 패턴 집합을 변경 해도 파서 변환에서 추가하는 데이터는 업데이트되지 않습니다.

"+" 및 "\*" 와일드카드를 사용하여 패턴을 정의할 수 있습니다. 어떤 문자열이든 일치시키려면 "\*" 문자를 사용하고 이전 문자열에서 하나 이상의 인스턴스를 일치시키려면 "+" 문자를 사용합니다. 예를 들어 한 단어 토큰의 여러 연속 인스턴스를 찾으려면 "WORD+"를 사용하고 어떤 유형이든 하나 이상의 토큰이 뒤따라오는 단어 토큰을 찾으려면 "WORD \*"를 사용합니다.

이러한 방법의 여러 인스턴스를 파서 변환에서 사용할 수 있습니다. 변환은 **구성** 보기에 나열된 순서로 인스턴스를 사용합니다.

**참고:** 패턴 기반 구문 분석 모드에서 파서 변환을 사용하려면 토큰 레이블 지정 모드를 사용하는 라벨러 변환의 출력이 필요합니다. 패턴 기반 구문 분석 모드를 사용하는 파서 변환을 작성하기 전에 라벨러 변환을 작성하고 구성합니다.

## 패턴 기반 구문 분석 포트

데이터에 적합한 설정을 사용하여 패턴 기반 구문 분석 포트를 구성합니다.

패턴 기반 구문 분석 모드를 사용하는 파서 변환에는 다음 포트 유형이 있습니다.

### Label\_Data

이 포트를 토큰 레이블 지정 모드를 사용하는 라벨러 변환의 Labeled\_Output 포트에 연결합니다.

### Tokenized\_Data

이 포트를 토큰 레이블 지정 모드를 사용하는 라벨러 변환의 Tokenized\_Data 출력 포트에 연결합니다.

### Parse\_Status

입력 패턴에 대해 일치 항목이 있는 경우 이 포트가 일치됨 값을 출력합니다. 일치 항목이 없는 경우 일치되지 않음을 출력합니다.

### 오버플로우

성공적으로 구문 분석되었지만 변환에 정의된 출력 수에 맞지 않는 문자열입니다. 예를 들어 두 "WORD" 출력만 정의된 경우 문자열 "John James Smith"는 기본적으로 "Smith"가 오버플로우 출력이 됩니다.

### 구문 분석됨

사용자 정의된 포트에서 성공적으로 구문 분석된 문자열입니다.

## 토큰 구문 분석 전략 구성

토큰 구문 분석 전략을 구성하려면 토큰 구문 분석 모드에서 파서 변환을 열고 **전략** 보기를 선택합니다.

1. **전략** 보기를 선택합니다.

2. **새로 만들기**를 클릭합니다.  
새 전략 마법사가 열립니다.
3. **입력** 필드를 클릭하여 전략에 대한 포트를 선택합니다.
4. 전략 속성을 구성하고 **다음**을 클릭합니다.
5. 작업을 선택하고 **다음**을 클릭합니다.
6. 작업 속성을 구성하고 성공적으로 구문 분석된 데이터에 대한 출력 포트를 선택합니다.
7. 필요한 경우 **다음**을 클릭하여 더 많은 작업을 전략에 추가합니다.
8. 모든 작업을 전략에 추가한 후에 **마침**을 클릭합니다.
9. 필요한 경우 더 많은 전략을 변환에 추가합니다.
10. 필요에 따라 변환이 전략 및 작업을 처리하는 순서를 변경합니다. 전략 또는 작업을 선택하고 **위로 이동** 또는 **아래로 이동**을 클릭합니다.

## 패턴 구문 분석 전략 구성

패턴 구문 분석 전략을 구성하려면 패턴 구문 분석 모드에서 파서 변환을 열고 **패턴** 보기를 선택합니다.

패턴을 구문 분석하도록 변환을 구성하기 전에 **패턴** 보기에 필요한 출력 포트 이름이 표시되는지 확인합니다. 파서 변환은 토큰을 선택하는 출력 포트로 구문 분석합니다. 필요한 경우 추가 출력 포트를 작성합니다.

1. **패턴** 보기를 선택합니다.
2. 전략에 패턴을 하나 이상 추가합니다. 다음과 같은 방식으로 패턴을 추가할 수 있습니다.
  - 패턴 작성을 위한 데이터 값을 입력합니다. **새로 만들기**를 클릭하고 **새 패턴**을 선택합니다.  
새 패턴을 선택하는 경우 **여기에 패턴 입력**을 클릭하고 토큰 유형을 하나 이상 입력합니다. 입력 데이터 필드의 토큰 구조와 일치하는 토큰을 입력해야 합니다. 입력 포트의 토큰 구조를 설명하는 데 필요한 패턴을 추가합니다.
  - 참조 테이블에서 데이터 값을 가져옵니다. **새로 만들기**를 클릭하고 **새 참조 테이블**을 선택합니다.  
새 참조 테이블을 선택하는 경우 모델 리포지토리를 찾아보고 토큰 구조 목록이 포함된 참조 테이블을 선택합니다. 참조 테이블에는 두 개의 열이 포함되어야 합니다. 참조 테이블의 두 번째 열에는 숫자 값이 포함되어야 합니다.
  - 패턴 집합에서 데이터 값을 가져옵니다. **가져오기**를 클릭하고 모델 리포지토리에서 재사용 가능 패턴 집합을 선택합니다.  
가져오기를 선택하는 경우 모델 리포지토리의 콘텐츠 집합을 찾은 다음 재사용 가능 패턴 집합을 선택합니다.

**참고:** 필터 텍스트 필드를 사용하여 참조 테이블 및 패턴 집합 목록을 필터링할 수 있습니다.

패턴 열에는 패턴 집합과 참조 테이블을 함께 포함할 수 있습니다.

3. 패턴 열의 각 토큰을 출력 포트에 할당합니다.
  - 토큰을 출력 포트에 할당하려면 포트 열을 두 번 클릭하고 메뉴에서 토큰 이름을 선택합니다.
  - 여러 토큰을 단일 출력으로 구문 분석하려면 포트 열을 두 번 클릭하고 **사용자 지정**을 선택합니다. 포트에 토큰을 할당하고 사용할 구분자를 선택합니다.

패턴의 각 행에 포함된 토큰을 하나 이상의 출력 포트에 할당합니다.
4. 변환을 저장합니다.

## 파서 변환 고급 속성

데이터 통합 서비스가 파서 변환의 데이터를 처리하는 방법을 결정하는 데 사용할 수 있는 속성을 구성합니다.

로그의 추적 수준을 구성할 수 있습니다.

**고급** 탭에서 다음 속성을 구성합니다.

### 추적 수준

이 변환에 대해 로그에 표시되는 세부 정보의 양입니다. 간단, 보통, 자세한 정보 표시 초기화 또는 자세한 정보 표시 데이터를 선택할 수 있습니다. 기본값은 보통입니다.

## 파서 변환 - 비원시 환경

비원시 환경에서 파서 변환의 처리는 변환을 실행하는 엔진에 따라 다릅니다.

다음과 같은 비원시 런타임 엔진에 대한 지원을 고려하십시오.

- **Blaze** 엔진. 제한 없이 지원됩니다.
- **Spark** 엔진. 제한 없이 지원됩니다.
- **Databricks Spark** 엔진. 지원되지 않습니다.

## 제 34 장

# Python 변환

이 장에 포함된 항목:

- [Python 변환 개요, 508](#)
- [Python 변환 포트, 510](#)
- [Python 변환 고급 속성, 510](#)
- [Python 변환 구성 요소, 510](#)
- [규칙 및 지침, 512](#)
- [Python 변환 생성, 512](#)
- [Python 변환 사용 사례, 513](#)
- [Python 변환 - 비원시 환경, 514](#)

## Python 변환 개요

Spark 엔진에서 실행되는 매핑에서 Python 코드를 실행하려는 경우 Python 변환을 사용할 수 있습니다.

Python 변환은 Python 코드를 사용하여 변환 기능을 정의할 수 있는 인터페이스를 제공하는 수동 변환입니다. Python 변환 안에는 Python 코드와 Python 코드에 사용되는 리소스 파일이 참조됩니다.

Python 변환을 사용하면 변환에 전달하는 데이터에 대한 시스템 모델을 구현할 수 있습니다. 예를 들어 Python 변환을 사용하여 미리 학습된 모델을 로드하는 Python 코드를 쓸 수 있습니다. 입력 데이터를 분류하거나 예측을 생성하도록 미리 학습된 모델을 사용할 수 있습니다.

Python 변환을 사용하려면 먼저 데이터 통합 서비스 시스템에 Python을 설치하고 Hadoop 연결에서 해당하는 Spark 고급 속성을 구성해야 합니다.

Python 설치에 대한 자세한 내용은 *Informatica Big Data Management 통합 가이드*를 참조하십시오.

## 데이터 유형 변환

Python 변환은 Developer tool 데이터 유형을 Python 변환의 포트 유형에 따라 Python 데이터 유형으로 변환합니다.

Python 변환은 입력 행을 읽고 입력 포트 데이터 유형을 Python 데이터 유형으로 변환합니다. Python 변환이 출력 행을 기록할 때는 Python 데이터 유형을 출력 포트의 데이터 유형으로 변환합니다.

예를 들어 Python 변환에서 배정밀도 데이터 유형을 포함하는 입력 포트는 다음과 같이 처리됩니다.

- Python 변환은 입력 포트의 배정밀도 데이터 유형을 Python 부동 소수점 수 데이터 유형으로 변환합니다.



- 변환은 입력 포트의 값을 Python 부동 소수점 수 데이터 유형의 값으로 사용합니다.
- Python 변환은 출력 행을 생성할 때 Python 부동 소수점 수 데이터 유형을 배정밀도 데이터 유형으로 변환합니다.

다음 테이블에는 Python 변환이 Developer tool 데이터 유형을 Python 데이터 유형으로 변환하는 방법이 나와 있습니다.

Developer tool 데이터 유형	Python 데이터 유형
정수	Int
10진수	부동 소수점 수
배정밀도	부동 소수점 수
이진*	PyJArray
타임스탬프	날짜/시간
문자열	Str
<b>연기 공지:</b> Python 변환의 이진 포트에 대한 지원이 연기되었습니다. 지원은 향후 릴리스에서 복구됩니다. <i>Python 변환은 이 테이블에 나열되지 않은 데이터 유형을 지원하지 않습니다.</i>	

Python 변환에서 코드를 작성하는 경우 Python 변환의 출력 포트 데이터 유형은 Python 코드의 데이터 유형과 호환되어야 합니다. 따라서 Python 변환의 출력 포트를 배정밀도 데이터 유형으로 구성하는 경우 Python 코드에서 해당하는 변수는 부동 소수점 수여야 합니다.

## 입력 및 출력 포트의 데이터 유형

Python 변환에서 해당하는 입력 및 출력 포트의 데이터 유형은 동일해야 합니다. 데이터 유형이 동일하지 않은 경우 Python 코드의 데이터 유형을 변환합니다.

예를 들어 정수 데이터 유형을 사용하는 입력 포트와 문자열 데이터 유형을 사용하는 출력 포트를 생성합니다. 그런 다음 입력 포트의 데이터를 처리하고 출력 포트에 데이터를 쓰는 Python 코드를 정의합니다. Python 코드에서 Python 함수 `str()`을 사용하여 입력 포트의 정수 데이터 유형을 변환하고 출력 포트에 문자열 데이터 유형으로 출력을 쓸 수 있습니다.

**참고:** 이진 데이터 유형을 Python 변환으로 전달하면 Python 변환이 이진 데이터 유형을 PyJArray로 변환합니다. Python 코드에서 PyJArray를 다른 Python 데이터 유형(예: `byte`, `bytearray` 또는 `struct`)으로 변환하여 코드에 사용할 수 있습니다. 출력 변수를 정의할 때는 Python 데이터 유형을 Python 변환에서 지원되는 데이터 유형으로 변환해야 합니다.

**연기 공지:** Python 변환의 이진 포트에 대한 지원이 연기되었습니다. 지원은 향후 릴리스에서 복구됩니다.

# Python 변환 포트

Python 변환에는 입력 포트와 출력 포트가 있습니다.

재사용 불가능한 Python 변환의 포트를 생성하고 편집하려면 편집기의 **포트** 탭을 사용합니다. 재사용 가능한 Python 변환의 포트를 생성하고 편집하려면 편집기의 **개요** 보기를 사용합니다. 변환에 포트를 추가한 후 해당 포트 이름을 Python 코드의 변수로 사용할 수 있습니다.

다음 규칙을 사용하여 입력 및 출력 포트를 생성합니다.

- 포트 이름에는 ASCII 문자만 사용할 수 있습니다.
- 포트 이름은 Python 키워드일 수 없습니다. 예를 들어 `import`, `global` 또는 `class` 같은 포트 이름을 사용하지 마십시오.
- 포트 이름은 `resourceJepFile`일 수 없습니다.

Python 변환의 출력 포트에 사용자 정의 기본값을 구성할 수 없습니다. 사용자 정의 기본값은 Python 코드에서 설정할 수 있습니다.

예를 들어 `output_port` 출력 포트에 대한 'value' 기본값을 설정하려면 `output_port = 'value'`를 작성합니다.

## Python 변환 고급 속성

Python 변환에는 변환에 대한 고급 속성이 포함됩니다.

고급 탭에서 Python 변환에 대한 다음 고급 속성을 정의할 수 있습니다.

### 추적 수준

이 변환에 대해 로그에 표시되는 세부 정보의 양입니다. 간단, 보통, 자세한 정보 표시 초기화 또는 자세한 정보 표시 데이터를 선택할 수 있습니다. 기본값은 보통입니다.

## Python 변환 구성 요소

Python 변환에는 변환에 전달하는 데이터에서 Python 스크립트를 실행할 때 사용되는 구성 요소가 포함됩니다.

Python 변환에는 다음과 같은 구성 요소가 포함됩니다.

### 리소스 파일

Python 코드에서 액세스하는 리소스가 포함된 플랫폼 파일입니다.

이 파일은 Developer tool 외부에서 대용량 데이터 집합에 대해 학습된 미리 학습된 모델일 수 있습니다. 미리 학습된 모델을 사용하여 데이터를 분류하거나 Python 변환에 전달하는 데이터를 기준으로 예측을 생성할 수 있습니다. Python 코드에서 미리 학습된 모델에 액세스할 수 있습니다.

### Python 코드

Python 변환이 Python 변환으로 전달되는 데이터를 처리할 때 사용하는 Python 스크립트입니다. Python 코드를 작성하면 입력 변수를 다시 구성하고, 미리 학습된 모델을 로드하고, 출력 변수를 정의할 수 있습니다.

## 리소스 파일

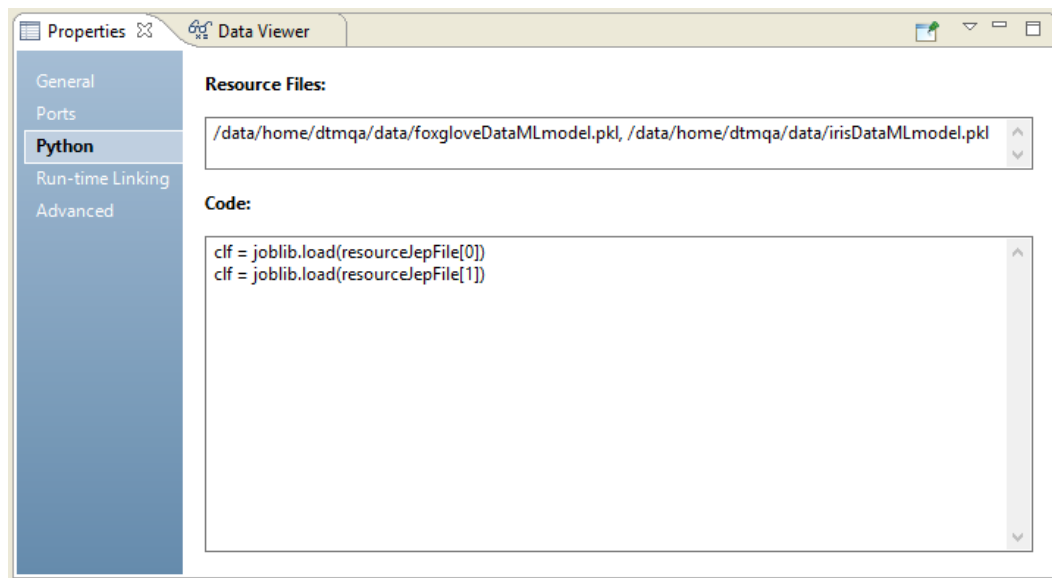
리소스 파일은 Python 코드에서 사용하는 리소스가 포함된 플랫폼 파일입니다. 미리 학습된 모델을 사용하는 경우 Python 변환에서 미리 학습된 모델을 리소스 파일로 지정합니다.

Python 변환 안에 데이터 통합 서비스 시스템의 리소스 파일 경로를 나열합니다. 리소스 파일 경로가 여러 개인 경우 쉼표를 사용하여 구분합니다.

Python 코드에서 리소스 파일에 액세스할 때는 resourceJepFile 목록을 참조합니다. 목록을 참조하려면 Python 변환에서 리소스 파일을 지정해야 합니다. 목록을 참조할 때는 Python 변환에서 리소스 파일이 나타나는 순서에 따라 리소스 파일의 위치를 찾는 인덱스를 지정합니다.

예를 들어 Python 변환 안에 여러 개의 리소스 파일을 지정합니다. 이 경우 Python 코드의 첫 번째 리소스 파일을 참조하려면 resourceJepFile[0] 변수를 사용할 수 있습니다.

다음 이미지는 Python 변환에서 리소스 파일을 지정하고 Python 코드에서 리소스 파일에 액세스하는 방법을 보여 줍니다.



foxgloveDataMLmodel.pkl 리소스 파일과 irisDataMLmodel.pkl 리소스 파일은 데이터 통합 서비스 시스템의 리소스 파일 경로를 사용하여 나열되어 있습니다. Python 코드는 리소스 파일에 액세스하여 관련된 Python 개체를 다시 구성합니다. 첫 번째 리소스 파일에 액세스하는 경우 Python 코드는 resourceJepFile[0]을 사용하여 리소스 파일을 참조합니다. 두 번째 리소스 파일인 irisDataMLmodel.pkl에 액세스하는 경우 Python 코드는 resourceJepFile[1]을 사용하여 리소스 파일을 참조합니다.

## Python 코드

Python 코드는 Python 변환에서 변환의 데이터 처리 방법을 정의하기 위해 작성하는 Python 스크립트입니다. Python 코드를 작성하면 입력 변수를 다시 구성하고, 미리 학습된 모델을 로드하고, 출력 변수를 정의할 수 있습니다.

Python 코드를 작성할 때는 다음 규칙을 사용합니다.

- 입력 포트에 액세스하려면 입력 포트 이름을 호출합니다.
- 출력 포트를 설정하려면 출력 포트를 값으로 설정합니다. 출력 포트를 Python 변환에서 정의한 각 출력 포트의 값으로 설정해야 합니다.

- 변환이 입력 포트의 데이터를 출력 포트에 쓰는 방법을 정의하려면 출력 포트를 입력 포트의 값으로 설정합니다.

예를 들어 `input_port` 입력 포트의 데이터를 `output_port` 출력 포트에 쓰려면 `output_port = input_port`를 작성합니다.

- 리소스 파일 경로에 액세스하려면 `resourceJepFile` 변수를 사용합니다. `resourceJepFile[0]` 같은 인덱스를 사용하여 리소스 파일을 지정합니다.

Python 변환을 실행할 때 데이터 통합 서비스는 Python 코드의 유효성을 검사하지 않습니다.

## 규칙 및 지침

Python 변환을 사용하는 경우 다음 규칙 및 지침을 고려하십시오.

- Python 변환은 수동 변환입니다. Spark 엔진은 Python 변환을 행별로 처리합니다.
- 데이터 통합 서비스는 Python 변환의 Python 코드에 대한 유효성을 검사하지 않습니다.
- 복합 포트를 Python 변환으로 전달할 수 없습니다.

## Python 변환 생성

Developer tool에서 재사용 가능하거나 재사용 불가능한 Python 변환을 생성할 수 있습니다.

### 재사용 가능한 Python 변환 생성

재사용 가능한 Python 변환을 생성하여 여러 매핑에서 이 변환을 사용할 수 있습니다. Developer tool에서 재사용 가능한 Python 변환을 생성합니다.

1. **개체 탐색기** 보기에서 프로젝트 또는 폴더를 선택합니다.
2. **파일 > 새로 만들기 > 변환**을 클릭합니다.  
새로 만들기 대화 상자가 나타납니다.
3. Python 변환을 선택합니다.
4. **다음**을 클릭합니다.
5. 변환 이름을 입력합니다.
6. **마침**을 클릭합니다.  
변환이 편집기에 표시됩니다.
7. **개요** 보기에서 **새로 만들기** 단추를 클릭하여 포트를 변환에 추가합니다.
8. 포트를 편집하여 이름, 데이터 유형 및 전체 자릿수를 설정합니다.  
포트 이름을 Python 코드의 변수로 사용합니다.
9. **Python** 보기에서 리소스 파일을 지정하고 변환에 사용할 Python 코드를 작성합니다.
10. **고급** 보기에서 변환에 대한 고급 속성을 편집합니다.

## 재사용 불가능한 Python 변환 생성

재사용 불가능한 Python 변환을 생성하여 단일 매핑에서 이 변환을 사용할 수 있습니다. Developer tool에서 재사용 불가능한 Python 변환을 생성합니다.

1. 매핑 또는 맵렛에서 매핑 색상표의 Python 변환을 편집기로 끌어 옵니다.  
변환이 편집기에 표시됩니다.
2. **일반** 탭에서 변환 이름 및 설명을 편집합니다.
3. **포트** 탭에서 **새로 만들기** 단추를 클릭하여 포트를 변환에 추가합니다.
4. 포트를 편집하여 이름, 데이터 유형 및 전체 자릿수를 설정합니다.  
포트 이름을 Python 코드의 변수로 사용합니다.
5. **Python** 탭에서 리소스 파일을 지정하고 변환에 사용할 Python 코드를 컴파일합니다.
6. 런타임 시 포트가 변경되는 경우 **런타임 연결** 탭에서 매핑의 Python 변환과 업스트림 또는 다운스트림 변환 간의 런타임 링크를 생성합니다.
7. **고급** 탭에서 변환에 대한 고급 속성을 편집합니다.

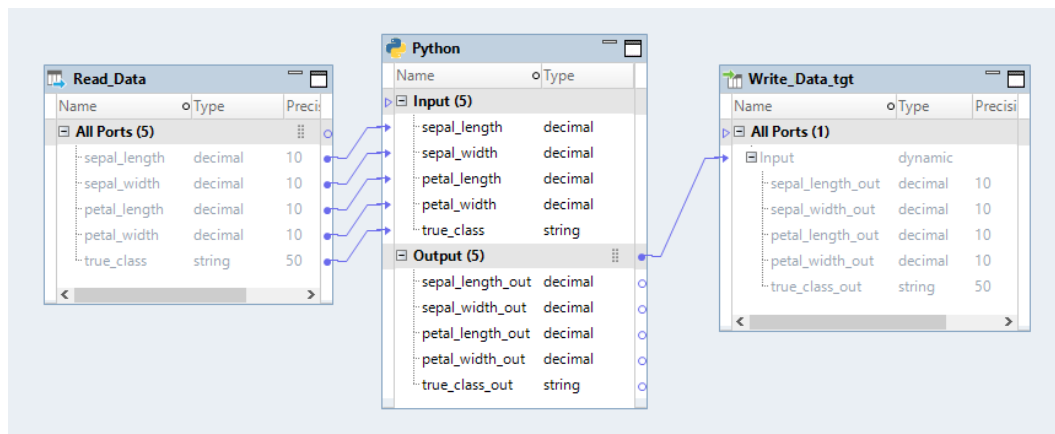
## Python 변환 사용 사례

제약 회사에서 심장 질환의 치료를 개선하기 위해 디기탈리스의 화기 형성에 대한 데이터를 연구하는 한 연구원이 있습니다. 이 연구원은 일반적인 디기탈리스인 *Digitalis purpurea*와 털이 있는 디기탈리스인 *Digitalis lanata* 중에서 어느 디기탈리스가 질병의 진행을 늦추는 데 효과적인지를 알아내려고 합니다.

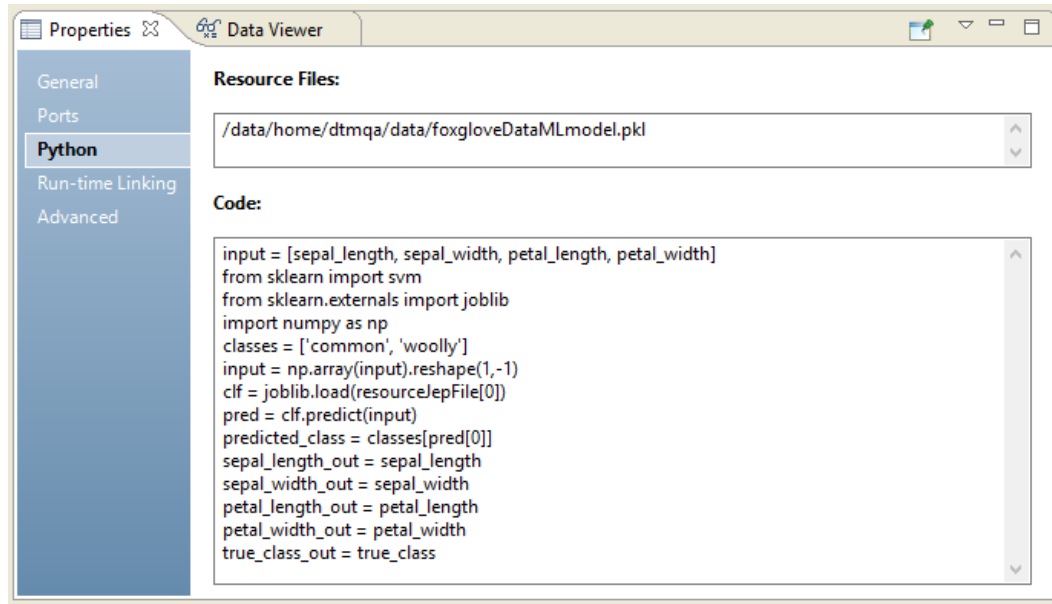
이 연구원은 연구를 위해 꽃의 종류별로 꽃받침과 꽃잎의 길이와 너비에 대한 데이터를 분류해야 합니다. 데이터를 분류하기 위해 연구원은 Developer tool 외부에서 미리 학습된 모델을 개발했습니다.

이제 Developer tool에서 미리 학습된 모델을 작동 가능한 상태로 만들어야 합니다. Developer tool에서 Python 변환이 포함된 매핑을 생성합니다. 미리 학습된 모델을 리소스 파일 형태로 Python 변환에 나열합니다. 미리 학습된 모델에 액세스하는 Python 스크립트를 작성합니다. 꽃받침에 대한 데이터를 Python 변환에 전달하여 디기탈리스의 종에 따라 데이터를 분류합니다.

다음 이미지는 연구원이 생성할 수 있는 매핑을 보여 줍니다.



다음 이미지는 Python 변환의 미리 학습된 모델에 액세스하기 위해 연구원이 작성할 수 있는 Python 코드를 보여 줍니다.



Python 변환은 Python 스크립트에 따라 입력 포트의 데이터를 처리하고 분류된 데이터를 출력 포트에 씁니다.

## Python 변환 - 비원시 환경

비원시 환경에서 Python 변환의 처리는 변환을 실행하는 엔진에 따라 다릅니다.

다음과 같은 비원시 런타임 엔진에 대한 지원을 고려하십시오.

- Blaze 엔진. 지원되지 않습니다.
- Spark 엔진. 제한적으로 지원됩니다.
- Databricks Spark 엔진. 지원되지 않습니다.

## Python 변환 - Spark 엔진

출력 포트에 사용자 정의 기본값이 할당된 경우 매핑 유효성 검사가 실패합니다.

다음과 같은 상황에서는 매핑이 실패합니다.

- 출력 포트에 Python 코드의 값이 할당되지 않았습니다.
- 해당하는 입력 및 출력 포트의 데이터 유형이 동일하지 않으며 Python 코드가 입력 포트의 데이터 유형을 출력 포트의 데이터 유형으로 변환하지 않습니다.
- Python 변환에 10진수 포트가 포함되어 있고 매핑에 많은 전체 자릿수가 활성화되어 있습니다.

**참고:** 데이터 통합 서비스는 Python 코드의 유효성을 검사하지 않습니다.

## 제 35 장

# 순위 변환

이 장에 포함된 항목:

- [순위 변환 개요, 515](#)
- [동적 매핑의 순위 변환, 516](#)
- [순위 변환 포트, 516](#)
- [순위 포트, 518](#)
- [그룹 기준 포트 정의, 518](#)
- [순위 캐시, 519](#)
- [순위 변환 고급 속성, 520](#)
- [순위 변환 - 비원시 환경, 520](#)

## 순위 변환 개요

순위 변환은 레코드를 상위 또는 하위 범위로 제한하는 활성 변환입니다. 순위 변환을 사용하여 포트 또는 그룹에서 가장 크거나 가장 작은 숫자 값을 반환할 수 있습니다. 또는 순위 변환을 사용하여 매핑 정렬 순서의 상위 또는 하위에 있는 문자열을 반환할 수 있습니다.

매핑을 실행하는 동안 데이터 통합 서비스가 순위 계산을 수행할 수 있을 때까지 입력 데이터를 캐시합니다.

순위 변환은 변환 함수 **MAX** 및 **MIN**과 다릅니다. 순위 변환은 하나의 값이 아닌 상위 또는 하위 값 그룹을 반환합니다. 예를 들어 순위 변환을 사용하여 지정된 지역에서 상위 10명의 판매자를 선택할 수 있습니다. 또는 재무 보고서 생성하기 위해 순위 변환을 사용하여 급여 및 오버헤드에서 가장 낮은 비용을 사용하는 3개의 부서를 식별할 수 있습니다. **SQL** 언어가 데이터 그룹을 처리하기 위해 설계된 여러 함수를 제공하지만 표준 **SQL** 함수를 사용해서는 행 집합에서 상위 또는 하위 계층을 식별할 수 없습니다.

동일한 행 집합을 나타내는 모든 포트를 변환에 연결합니다. 변환을 구성할 때 설정하는 몇 가지 측정에 따라 해당 순위에 속하는 행이 순위 변환을 통과합니다.

활성 변환인 순위 변환에서는 통과하는 행 수를 변경할 수 있습니다. 100개의 행을 순위 변환에 전달하지만 상위 10개의 행만 순위를 지정하도록 선택할 수 있습니다. 상위 10개의 행이 순위 변환에서 다른 변환으로 전달됩니다.

한 변환에서 순위 변환으로 포트를 연결할 수 있습니다. 또한 로컬 변수를 작성하고 비집계 식을 쓸 수도 있습니다.

## 문자열 값 순위 지정

문자열 포트의 상위 또는 하위 값을 반환하도록 순위 변환을 구성할 수 있습니다. 배포된 매핑에 대해 선택된 정렬 순서에 따라 데이터 통합 서비스가 문자열을 정렬합니다.

매핑을 포함하는 응용 프로그램을 구성하는 경우 데이터 통합 서비스가 매핑을 실행하는 데 사용하는 정렬 순서를 선택합니다. 이진 또는 프랑스어나 독일어 같은 특정 언어를 선택할 수 있습니다. 이진을 선택하는 경우 데이터 통합 서비스가 각 문자열의 이진 값을 계산하고 이진 값을 사용하여 문자열을 정렬합니다. 언어를 선택하는 경우 데이터 통합 서비스가 언어의 정렬 순서를 사용하여 알파벳 순서에 따라 문자열을 정렬합니다.

## 순위 변환 속성

순위 변환을 작성하는 경우 다음 속성을 구성할 수 있습니다.

- 캐시 디렉터리를 입력합니다.
- 상위 또는 하위 순위를 선택합니다.
- 순위를 결정하는 데 사용되는 값이 포함된 입력/출력 포트를 선택합니다. 한 포트만 선택하여 순위를 정의할 수 있습니다.
- 순위를 지정하려는 행 수를 선택합니다.
- 각 제조업체에 대해 10개의 가장 저렴한 제품과 같이 순위를 위한 그룹을 정의합니다.

## 동적 매핑의 순위 변환

동적 매핑에서 순위 변환을 사용할 수 있습니다. 변환에서 동적 포트를 구성하고 생성된 포트를 참조할 수 있습니다.

순위 변환에서 생성된 포트를 참조하고 런타임 시 생성된 포트가 없는 경우 매핑이 실패합니다.

동적 포트를 순위 포트에 지정하는 경우 동적 포트에는 한 개의 생성된 포트만 포함될 수 있습니다.

동적 포트를 그룹 기준 포트에 지정하는 경우 데이터 통합 서비스가 생성된 모든 포트를 그룹 기준 포트에 고려합니다. 생성된 포트를 그룹 기준 포트에 지정하고 상위 동적 포트를 순위 포트 또는 그룹 기준 포트에 지정하는 경우 매핑은 올바르게 실행되지 않습니다.

순위 포트 및 그룹 기준 포트를 매개 변수화할 수 있습니다. 순위 포트에 대해 포트 유형 매개 변수를 사용합니다. 그룹 기준 포트에 대해 포트 목록 유형 매개 변수를 사용합니다.

## 순위 변환 포트

순위 변환에는 매핑의 다른 변환에 연결된 입력, 입력/출력 또는 출력 포트가 포함됩니다. 또한 변환에는 통과 및 변수 포트가 포함됩니다.

순위 변환에는 다음 포트 유형이 있습니다.

### 입력

업스트림 변환의 데이터를 수신합니다. 입력 포트를 입력/출력 포트에 지정할 수 있습니다. 변환에는 입력 포트가 하나 이상 있어야 합니다.



## 동적 포트

여러 열을 받아 동적 수의 생성된 포트를 작성할 수 있는 포트입니다. 생성된 포트는 단일 열을 나타내는 동적 포트 내 포트입니다. 입력, 출력 및 변수 동적 포트를 작성할 수 있습니다.

## 출력

데이터를 다운스트림 변환에 전달합니다. 출력 포트를 입력/출력 포트로 지정할 수 있습니다 변환에는 출력 포트가 하나 이상 있어야 합니다.

## 통과

변경되지 않은 데이터를 전달합니다.

## 변수

로컬 변수에 사용됩니다. 식에서 사용할 값 또는 계산을 변수 포트에 저장할 수 있습니다. 변수 포트는 입력 또는 출력 포트일 수 없습니다. 변수 포트는 변환 내에서 데이터를 전달합니다.

# 순위 인덱스

Developer tool은 각 순위 변환에 대해 RANKINDEX 포트를 작성합니다. 데이터 통합 서비스는 순위 인덱스 포트를 사용하여 그룹의 각 행에 대한 순위 위치를 저장합니다.

예를 들어 회사에서 가장 높은 급여를 받는 50명의 직원을 식별하기 위한 순위 변환을 작성할 수 있습니다. SALARY 열을 순위를 측정하기 위해 사용되는 입력/출력 포트로 식별하고 상위 50명을 제외한 모든 행을 필터링하도록 변환을 구성합니다.

순위 변환이 상위 또는 하위 순위에 속하는 모든 행을 식별한 다음 순위 인덱스 값을 할당합니다. 급여로 측정된 상위 50명의 직원의 경우 가장 높은 급여를 받는 직원이 순위 인덱스 1을 받습니다. 다음으로 높은 급여를 받는 직원이 순위 인덱스 2를 받습니다. 이런 식으로 계속 진행됩니다. 재고에서 10개의 가장 싼 제품과 같이 하위 순위를 측정하는 경우 순위 변환이 가장 낮은 값에서 가장 높은 값으로 순위 인덱스를 할당합니다. 그러므로 가장 싼 항목이 순위 인덱스 1을 받습니다.

두 개의 순위 값이 일치하는 경우에는 순위 인덱스에서 동일한 값을 받고 변환은 다음 값을 건너뜁니다. 예를 들어 국가에 상위 5개의 소매점을 표시하려는데 두 상점의 판매량이 동일한 경우 반환 데이터가 다음과 유사하게 표시될 수 있습니다.

RANKINDEX	SALES	STORE
1	10000	Orange
1	10000	Brea
3	90000	Los Angeles
4	80000	Ventura

RANKINDEX는 출력 포트 전용입니다. 순위 인덱스를 매핑의 다른 변환에 전달하거나 대상에 직접 전달할 수 있습니다.

## 순위 포트

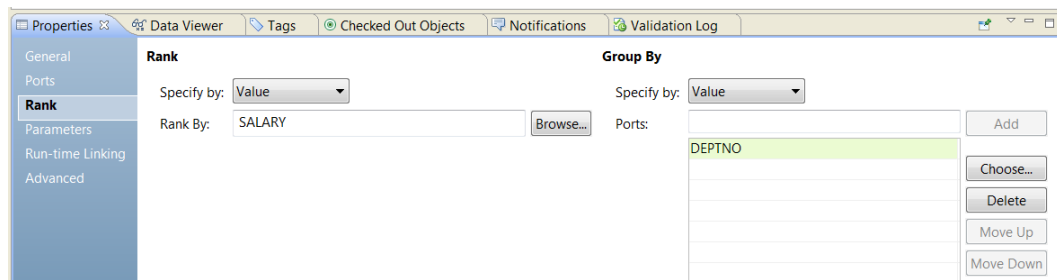
순위 포트는 열을 순위 값 기준으로 결정합니다.

하나의 입력/출력 또는 출력 포트를 순위 포트로 지정해야 합니다. 예를 들어 순위 변환을 작성하여 급여에 따라 각 부서 최고 직원의 순위를 지정합니다. 급여 포트에는 각 직원의 급여가 포함됩니다. 급여 입력/출력 포트를 순위 포트를 지정합니다.

**속성** 보기의 **순위** 탭에서 순위 포트를 선택합니다. 순위 포트에 대해 매개 변수를 사용할 수 있습니다. 매개 변수를 사용하려면 **매개 변수로 지정**을 선택합니다. 포트 매개 변수를 찾거나 포트 매개 변수를 작성합니다. 매개 변수 기본값은 포트 또는 생성된 포트의 이름입니다.

순위 포트를 다른 변환에 연결해야 합니다.

다음 이미지는 **순위** 탭을 보여 줍니다.



## 그룹 기준 포트 정의

순위 지정된 행에 대한 그룹을 작성하도록 순위 변환을 구성할 수 있습니다.

예를 들어, 제조업체별로 가장 비싼 항목 10개를 선택할 경우 먼저 각 제조업체에 대해 그룹을 정의합니다. **순위** 탭의 **그룹 기준** 패널에서 입력, 입력/출력 또는 출력 포트 중 하나를 그룹 기준 포트로 설정할 수 있습니다.

그룹 포트의 고유한 값마다 변환에서 순위 정의에 해당하는 행 그룹을 작성합니다(위 또는 아래 및 각 순위의 특정 번호).

순위 변환에서 행 수를 다음 두 가지 방법으로 변경합니다. 맨 위 또는 맨 아래 순위에 해당하는 거의 모든 행을 필터링하여 변환을 통과하는 행 수를 줄이십시오. 그룹을 정의하여 각 그룹에 대해 순위 지정된 행 집합을 1개 작성합니다.

예를 들어, 분기별로 그룹화된 상위 5명의 영업사원 순위를 지정하는 순위 변환을 작성한 경우, 순위 인덱스에서 각 분기마다 1에서 5까지 영업사원 번호를 지정합니다.

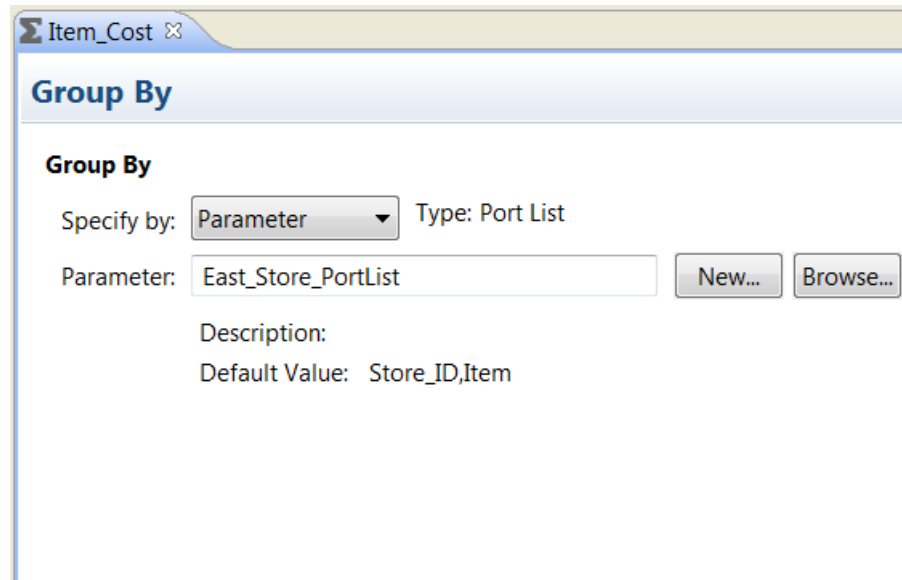
RANKINDEX	SALES_PERSON	SALES	QUARTER
1	Sam	10,000	1
2	Mary	9,000	1
3	Alice	8,000	1
4	Ron	7,000	1
5	Alex	6,000	1

속성 보기의 **고급** 탭에서 순위에 포함할 행 수를 설정합니다.

## 그룹 기준 매개 변수

그룹에 포함할 하나 이상의 포트를 포함하는 포트 목록 매개 변수를 구성할 수 있습니다. 변환의 포트 목록에서 포트를 선택하여 포트 목록 매개 변수를 작성합니다.

다음 이미지는 매개 변수를 사용하여 그룹의 포트를 식별하는 경우 **그룹 기준** 탭을 보여 줍니다.



포트 목록 매개 변수를 찾거나 **새로 만들기**를 클릭하여 포트 목록 매개 변수를 작성할 수 있습니다. 포트 목록 매개 변수를 작성하도록 선택하는 경우 변환의 포트 목록에서 포트를 선택할 수 있습니다.

## 순위 캐시

순위 변환을 사용하는 매핑을 실행하면 데이터 통합 서비스가 인덱스 캐시 및 데이터 캐시를 메모리에 작성하여 변환을 실행합니다. 메모리 캐시에서 사용할 수 있는 공간보다 더 많은 공간이 데이터 통합 서비스에 필요한 경우 데이터 통합 서비스는 오버플로우 데이터를 캐시 파일에 저장합니다.

순위 변환을 사용하는 매핑을 실행하면 데이터 통합 서비스가 입력 행을 데이터 캐시의 행과 비교합니다. 입력 행이 캐시된 행의 순위 밖에 있는 경우 데이터 통합 서비스가 캐시된 행을 입력 행으로 바꿉니다. 행을 그룹화하도록 순위 변환을 구성하는 경우 데이터 통합 서비스가 각 그룹에서 행의 순위를 지정합니다.

데이터 통합 서비스가 순위 변환에 대해 다음 캐시를 작성합니다.

- 그룹 기준 포트에 구성된 대로 그룹 값을 저장하는 인덱스 캐시.
- 그룹 기준 포트를 기반으로 정보를 저장하는 데이터 캐시.

# 순위 변환 고급 속성

데이터 통합 서비스가 순위 변환에 대해 데이터를 처리하는 방법을 결정할 수 있는 속성을 구성합니다.

**고급** 탭에서 다음 속성을 구성합니다.

## 상위/하위

열에 대해 상위 또는 하위 순위를 지정할지 여부를 지정합니다.

## 순위 수

상위 또는 하위 순위에 포함하려는 행의 수입니다.

## 대/소문자 구분 문자열 비교

데이터 통합 서비스가 문자열의 순위를 지정할 때 대/소문자 구분 문자열 비교를 사용할지 여부를 지정합니다. 데이터 통합 서비스가 문자열의 대/소문자 구분을 무시하도록 하려면 이 옵션을 지웁니다. 기본적으로 이 옵션은 선택되어 있습니다.

## 캐시 디렉터리

데이터 통합 서비스가 인덱스 캐시 파일 및 데이터 캐시 파일을 작성하는 디렉터리입니다. 디렉터리가 존재하고 캐시 파일을 위한 디스크 공간이 충분한지 확인하십시오.

캐시 분할 중에 성능을 향상시키려면 여러 디렉터리를 세미콜론으로 구분하여 입력하십시오. 캐시 분할은 변환을 처리하는 각 파티션에 대해 별도의 캐시를 작성합니다.

기본값은 **CacheDir** 시스템 매개 변수입니다. 이 속성에 대해 다른 시스템 매개 변수를 구성하거나 사용자 정의 매개 변수를 구성할 수 있습니다.

## 순위 데이터 캐시 크기

매핑 실행 시작 시 변환을 위해 데이터 통합 서비스가 데이터 캐시에 할당하는 메모리의 양입니다. "자동"을 선택하면 데이터 통합 서비스가 런타임 시 자동으로 메모리 요구 사항을 계산합니다. 캐시 크기를 조정하려는 경우 특정 값을 바이트 단위로 입력합니다. 기본값은 자동입니다.

## 순위 인덱스 캐시 크기

매핑 실행 시작 시 변환을 위해 데이터 통합 서비스가 인덱스 캐시에 할당하는 메모리의 양입니다. "자동"을 선택하면 데이터 통합 서비스가 런타임 시 자동으로 메모리 요구 사항을 계산합니다. 캐시 크기를 조정하려는 경우 특정 값을 바이트 단위로 입력합니다. 기본값은 자동입니다.

## 추적 수준

이 변환에 대해 로그에 표시되는 세부 정보의 양입니다. 간단, 보통, 자세한 정보 표시 초기화 또는 자세한 정보 표시 데이터를 선택할 수 있습니다. 기본값은 보통입니다.

## 관련 항목:

- [“캐시 크기” 페이지 71](#)

# 순위 변환 - 비원시 환경

비원시 환경에서 순위 변환의 처리는 변환을 실행하는 엔진에 따라 다릅니다.

다음과 같은 비원시 런타임 엔진에 대한 지원을 고려하십시오.

- **Blaze** 엔진. 제한적으로 지원됩니다.

- Spark 엔진. 제한적으로 지원됩니다.
- Databricks Spark 엔진. 제한적으로 지원됩니다.

## 순위 변환 - Blaze 엔진

Blaze 엔진의 일부 처리 규칙은 데이터 통합 서비스의 처리 규칙과 다릅니다.

순위 변환을 통해 전달되는 이진 및 문자열 데이터 유형을 변수 길이를 사용하여 저장하도록 순위 변환의 데이터 캐시가 최적화됩니다. 최적화는 최대 **8MB**의 레코드 크기에 대해 활성화됩니다. 레코드 크기가 **8MB**를 초과하는 경우 변수 길이 최적화가 비활성화됩니다.

순위 변환을 통해 전달되는 데이터를 데이터 캐시에 저장할 때 변수 길이가 사용되는 경우 순위 변환은 정렬된 입력을 사용하도록 최적화되고 통과 분류기 변환은 런타임 매핑에서 순위 변환의 앞에 삽입됩니다.

분류기 변환을 보려면 최적화된 매핑을 보거나 **Blaze** 유효성 검사 환경의 실행 계획을 봅니다.

데이터 캐시 최적화 중에는 순위 변환에 대한 데이터 캐시 및 인덱스 캐시가 자동으로 설정됩니다. 분류기 변환에 대한 분류기 캐시는 순위 변환에 대한 데이터 캐시와 동일한 크기로 설정됩니다. 분류기 캐시를 구성하려면 순위 변환에 대한 데이터 캐시의 크기를 구성해야 합니다.

## 순위 변환 - Spark 엔진

Spark 엔진의 일부 처리 규칙은 데이터 통합 서비스의 처리 규칙과 다릅니다.

### 매핑 유효성 검사

다음과 같은 상황에서는 매핑 유효성 검사가 실패합니다.

- 대/소문자 구분이 비활성화되어 있습니다.
- 순위 포트의 데이터 유형이 이진입니다.

### 데이터 캐시 최적화

변수 길이를 사용하여 데이터를 저장하도록 변환에 대한 데이터 캐시를 최적화할 수 없습니다.

## 순위 변환 - Databricks Spark 엔진

Databricks Spark 엔진의 일부 처리 규칙은 데이터 통합 서비스의 처리 규칙과 다릅니다.

### 매핑 유효성 검사

다음과 같은 상황에서는 매핑 유효성 검사가 실패합니다.

- 대/소문자 구분이 비활성화되어 있습니다.
- 순위 포트의 데이터 유형이 이진입니다.

### 데이터 캐시 최적화

변수 길이를 사용하여 데이터를 저장하도록 변환에 대한 데이터 캐시를 최적화할 수 없습니다.

## 제 36 장

# 읽기 변환

이 장에 포함된 항목:

- [읽기 변환 개요, 522](#)
- [읽기 변환 속성, 522](#)
- [관계형 데이터 개체 동기화, 525](#)
- [소스 데이터 개체 변경, 526](#)
- [읽기 변환 매개 변수, 528](#)
- [제약 조건, 529](#)
- [읽기 변환 작성, 529](#)

## 읽기 변환 개요

읽기 변환은 소스의 데이터를 읽는 수동 변환입니다. 읽기 변환은 재사용 불가능합니다.

실제 데이터 개체 또는 논리적 데이터 개체에서 읽기 변환을 작성할 수 있습니다. PowerExchange® 어댑터 소스에서 가져온 실제 데이터 개체에서 읽기 변환을 생성하려는 경우 매핑 편집기에서 데이터 개체를 사용하여 읽기 변환을 생성하기 전에 읽기 작업을 지정하라는 메시지가 표시될 수 있습니다.

변환을 작성하는 데 사용한 데이터 개체의 유형에 따라 읽기 변환에 대해 다른 속성을 구성할 수 있습니다. 예를 들어 관계형 데이터 개체에서 읽기 변환을 작성하는 경우 SQL 재정의의 구성하고 제약 조건을 정의할 수 있습니다. 구성할 수 있는 속성은 변환에 대해 매개 변수를 구성했는지 여부에 따라 다릅니다.

읽기 변환에는 동적 소스가 포함될 수 있습니다. 읽기 변환이 포트, 메타데이터 및 다른 속성을 동적으로 업데이트하도록 구성할 수 있습니다. 동적 소스 구성에 대한 자세한 내용은 *Informatica Developer 매핑 가이드*의 "동적 매핑" 장을 읽어 보십시오.

## 읽기 변환 속성

읽기 변환을 작성한 후 변환에 대한 속성을 구성할 수 있습니다.

속성 탭에서 읽기 변환 속성을 구성합니다. 사용할 수 있는 탭은 읽기 변환이 나타내는 소스의 유형에 따라 다릅니다.

다음 표에는 각 속성 탭이 설명되어 있고 탭에 대해 사용하는 소스 유형이 식별되어 있습니다.

속성 탭	설명	소스 유형
일반	변환 속성 및 동작을 지정합니다. 관계형 및 사용자 지정된 데이터 개체 소스의 경우 변환 입력 포트를 소스와 동기화합니다.	모두
데이터 개체	변환 데이터 소스를 지정합니다.	- 관계형 - 플랫폼 파일 - 사용자 지정된 데이터 개체
포트	연결된 데이터 개체로 포트 정의를 설정합니다.	- 관계형 - 사용자 지정된 데이터 개체 - 논리적 데이터 개체
형식	플랫폼 파일 데이터 소스에 대한 설정을 입력합니다.	플랫폼 파일
쿼리	소스에 쿼리를 지정합니다.	- 관계형 - 사용자 지정된 데이터 개체 - 논리적 데이터 개체
런타임	런타임 동작을 정의합니다.	- 관계형 - 플랫폼 파일 - 사용자 지정된 데이터 개체
소스	소스 테이블을 선택하고 소스 세부 정보를 구성합니다.	- 관계형 - 사용자 지정된 데이터 개체
데이터 개체 매개 변수	매개 변수 속성을 설정합니다.	- 플랫폼 파일 - 사용자 지정된 데이터 개체 - 논리적 데이터 개체
런타임 링크	런타임 시 사용할 포트를 확인하는 데 매개 변수, 연결 정책 또는 둘 다를 사용하는 변환 간 그룹-그룹 연결을 구성합니다.	모두
고급	추적 수준 및 행 순서를 설정합니다. 관계형 소스의 경우 옵션을 설정하여 런타임 시 대상 테이블을 작성하거나 바꿉니다.	모두

## 일반 속성

읽기 변환의 이름 및 설명을 구성할 수 있습니다. 또한 다음 속성을 구성할 수 있습니다.

### 열 메타데이터 변경 시

관계형 소스에서 사용 가능합니다. 다음 옵션 중 하나를 선택합니다.

- 출력 포트 동기화. **Developer tool**이 모델 리포지토리가 데이터 개체에 대해 저장하는 메타데이터 변경으로 읽기 변환 출력 포트를 업데이트합니다.
- 동기화 안 함. 읽기 변환이 데이터 개체의 메타데이터 변경을 표시하지 않습니다.

### 실제 데이터 개체

플랫폼 파일 및 사용자 지정된 소스에서 사용 가능합니다. 변환을 작성하는 데 사용된 개체입니다.

데이터 개체 이름을 선택하고 해당 속성을 구성할 수 있습니다.

## 데이터 개체 속성

데이터 개체 탭에서 동적으로 읽기 변환 소스를 지정하거나 변경하고 관계형, 플랫폼 파일 및 사용자 지정된 데이터 개체 소스를 만들 수 있습니다.

다음과 같은 속성을 구성할 수 있습니다.

### 지정 기준

읽기 변환에 대한 소스 열 및 메타데이터를 지정하려면 다음 옵션 중 하나를 선택합니다.

- 값. 읽기 변환이 연결된 데이터 개체를 사용하여 소스 열 및 메타데이터를 지정합니다.
- 매개 변수. 읽기 변환이 매개 변수를 사용하여 소스 열 및 메타데이터를 지정합니다.

### 데이터 개체

기존 데이터 개체에서 읽기 변환을 작성한 경우 필드에 개체의 이름이 표시됩니다. **찾아보기**를 클릭하여 읽기 변환과 연결할 데이터 개체를 변경합니다.

### 런타임 시 데이터 소스에서 데이터 개체 열을 가져옵니다.

이 옵션을 활성화하면 데이터 통합 서비스가 소스 테이블에서 메타데이터 및 데이터 정의 변경을 읽기 변환으로 가져옵니다.

## 쿼리 속성

관계형 리소스 또는 사용자 지정된 데이터 개체에 SQL 쿼리를 구성합니다.

쿼리 탭에서 속성을 구성하는 경우 단순 또는 고급 속성을 구성하도록 선택합니다.

**단순** 속성 보기에서 고유 정의 문으로 구성하고 문에 대한 힌트, 조인, 필터 및 정렬 조건을 편집하기 위해 기본 SQL 문을 구성합니다.

**고급** 속성 보기에서 사용자 지정 SQL 쿼리를 정의할 수 있습니다. 연결된 데이터 개체 또는 매개 변수에서 열을 선택하거나 새 매개 변수를 작성하여 데이터 개체를 나타낼 수 있습니다.

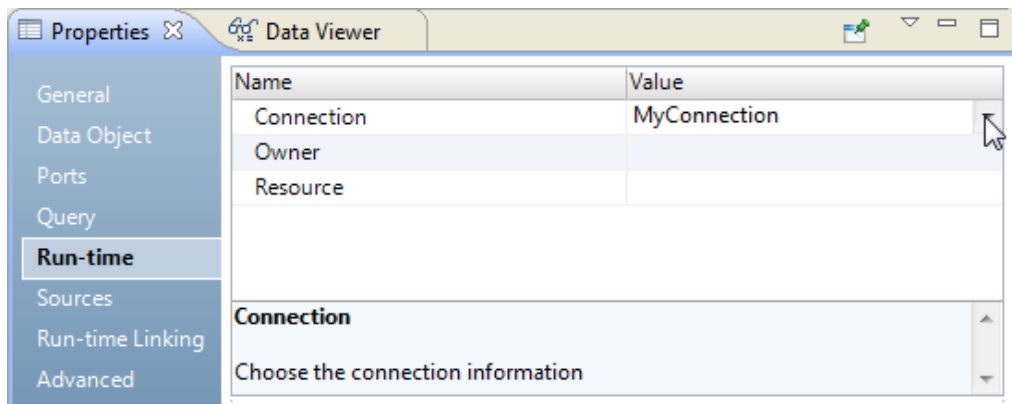
## 런타임 속성

런타임 탭에서 다음 읽기 변환 속성을 구성할 수 있습니다.

### 연결

관계형 소스에서 사용 가능합니다. 변환에서 사용된 연결입니다. 필드 오른쪽을 클릭하여 연결을 변경합니다.

다음 이미지는 클릭할 드롭다운 단추의 위치를 보여 줍니다.





## 소스 속성

관계형 리소스 및 사용자 지정된 데이터 개체에 대한 소스 세부 정보를 구성합니다. 리포지토리로 가져온 후 관계형 데이터 개체 정의를 변경할 수 있습니다. 포트를 추가 및 제거하고, 기본 키를 정의하고, 리포지토리의 여러 관계형 데이터 개체 간에 관계를 구성할 수 있습니다.

소스 탭에서 다음 설정을 구성할 수 있습니다.

### 모든 소스

추가 및 제거 단추를 사용하여 변환에 추가 소스를 추가하거나 제거합니다.

### 일반 탭

선택한 소스의 이름 및 설명을 변경합니다. 소스 이름을 클릭하여 다른 세부 정보를 변경합니다.

### 키 탭

리소스 열을 키로 지정합니다.

### 관계 탭

여러 관계형 리소스 간 관계를 추가 및 제거합니다.

## 고급 속성

고급 속성을 구성하여 데이터 통합 서비스가 읽기 변환에 대해 데이터를 처리하는 방법을 결정합니다.

고급 탭에서 다음 속성을 구성합니다.

### 추적 수준

매핑 로그 파일의 세부 정보의 양을 제어합니다.

### PreSQL

소스를 읽기 전에 소스 데이터베이스에 대해 데이터 통합 서비스가 실행하는 SQL 명령입니다.

Developer tool은 SQL의 유효성을 검사하지 않습니다.

### PostSQL

대상에 쓴 후에 소스 데이터베이스에 대해 데이터 통합 서비스가 실행하는 SQL 명령입니다.

Developer tool은 SQL의 유효성을 검사하지 않습니다.

### 제약 조건

테이블 수준 참조 무결성 제약 조건에 대한 SQL 문입니다. 관계형 소스에만 적용됩니다.

## 관계형 데이터 개체 동기화

실제 데이터 개체의 소스가 변경되는 경우 해당 개체를 동기화할 수 있습니다. 실제 데이터 개체를 동기화하면 Developer tool이 선택된 소스에서 개체 메타데이터를 다시 가져옵니다.

모든 실제 데이터 개체를 동기화할 수 있습니다. 관계형 데이터 개체 또는 사용자 지정된 데이터 개체를 동기화하는 경우 Developer tool에서 정의하는 키 관계를 유지하거나 덮어쓸 수 있습니다.

매핑 개체를 동기화하는 여러 방법 중 하나를 선택합니다.

## 관계형 리소스 동기화

실제 데이터 개체를 동기화하려면 **Object Explorer** 보기에서 개체를 마우스 오른쪽 단추로 클릭하고 **동기화**를 선택합니다.

## 변환 포트를 실제 데이터 개체와 동기화

변환의 데이터 개체 탭에서 **런타임 시 데이터 소스에서 데이터 개체 열을 가져옵니다**. 옵션을 선택합니다.

런타임 시 데이터 통합 서비스가 데이터 소스의 메타데이터 및 데이터 정의 변경 내용을 가져오고 모델 리포지토리의 데이터 개체 정의를 새로 고칩니다.

데이터 통합 서비스가 메타데이터 및 데이터 정의 변경을 가져오는 방법을 미리 보려면 확인된 매개 변수와 함께 매핑을 표시합니다.

## 메타데이터 변경 시 포트 동기화

변환의 일반 탭에서 포트 동기화 옵션을 선택합니다. 이 옵션의 정확한 레이블은 구성하는 변환의 유형에 따라 다릅니다. 예를 들어 읽기 변환의 경우 해당 옵션은 **메타데이터 변경 시 출력 포트 동기화**입니다.

매핑이 실행되면 데이터 통합 서비스가 변환의 열 메타데이터를 데이터 소스의 메타데이터와 동기화합니다.

# 소스 데이터 개체 변경

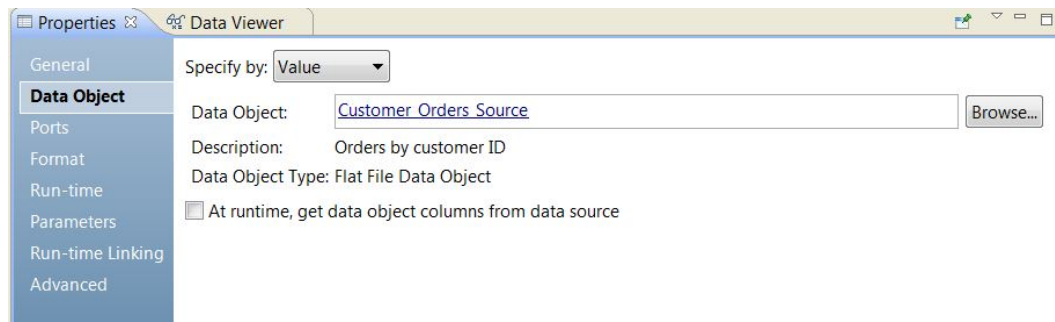
읽기 변환은 모델 리포지토리의 실제 데이터 개체 또는 논리적 데이터 개체를 기반으로 합니다. 읽기 변환을 구성하는 경우 데이터 개체를 변경할 수 있습니다. 데이터 개체를 매개 변수화하여 런타임 시 데이터 개체를 변경할 수 있습니다. 예를 들어 프로덕션 매핑 실행에 사용하는 소스 파일 대신 다른 소스 파일로 매핑을 테스트할 수 있습니다.

실제 데이터 개체에서 변환을 작성하는 경우 변환 속성의 **데이터 개체** 탭에 데이터 개체에 대한 정보가 나타납니다. 데이터 개체 이름을 클릭하여 모델 리포지토리에서 실제 데이터 개체 정의를 볼 수 있습니다.

모델 리포지토리에서 다른 실제 데이터 개체를 찾아 변환에 대한 데이터 개체를 변경할 수 있습니다. 데이터 개체를 변경하는 경우 변환은 선택하는 데이터 개체의 런타임 속성 및 고급 속성을 사용합니다.

데이터 소스의 변경에 따라 런타임 시 데이터 개체의 구조를 업데이트할 수 있습니다. 데이터 소스는 데이터 개체가 나타내는 실제 파일 또는 데이터베이스 테이블입니다. 데이터 통합 서비스를 활성화하여 데이터 소스에서 데이터 열을 가져오는 경우 데이터 통합 서비스가 데이터 소스의 구조를 검사합니다. 데이터 통합 서비스는 데이터 소스에 따라 변환 인스턴스의 데이터 개체 포트를 업데이트합니다. 데이터 통합 서비스는 모델 리포지토리의 실제 데이터 개체 정의를 변경하지 않습니다.

다음 이미지는 **데이터 개체** 탭을 보여 줍니다.



**데이터 개체** 탭에는 다음 필드가 있습니다.

## 지정 기준

값을 선택하여 특정 데이터 개체 이름을 입력합니다. 매개 변수를 선택하여 데이터 개체를 매개 변수화합니다.

## 데이터 개체

모델 리포지토리의 데이터 개체 이름입니다. 데이터 개체 링크를 클릭하여 리포지토리에서 데이터 개체 정의를 열 수 있습니다. 또한 모델 리포지토리의 다른 데이터 개체를 찾아볼 수 있습니다.

## 설명

리포지토리의 데이터 개체 설명입니다. 읽기 전용입니다.

## 데이터 개체 유형

플랫 파일 데이터 개체, 관계형 테이블 개체 또는 사용자 지정된 데이터 개체와 같은 데이터 개체의 유형을 설명합니다.

## 런타임 시 데이터 소스에서 데이터 개체 열을 가져옵니다.

데이터 통합 서비스는 런타임 시 데이터 개체가 참조하는 데이터 파일 또는 테이블에서 메타데이터 및 데이터 정의 변경을 가져온 후 변환 인스턴스에 대한 데이터 개체의 구조를 업데이트합니다.

런타임 시 데이터 통합 서비스가 메타데이터 및 데이터 정의 변경을 가져오는 방법을 미리 보려면 확인된 매개 변수와 함께 매핑을 표시합니다.

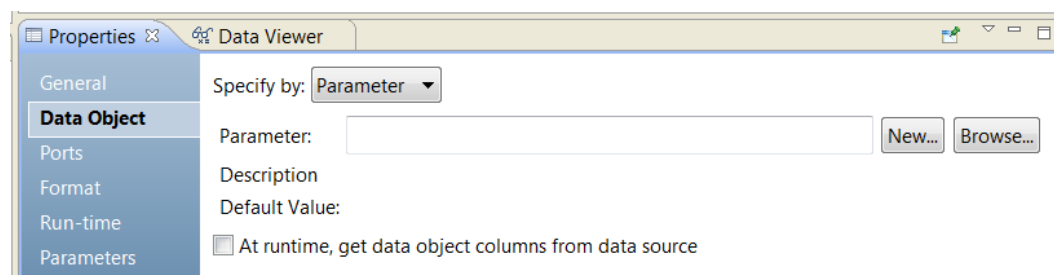
# 읽기 변환 매개 변수화

읽기 변환을 매개 변수화하고 런타임 시 데이터 개체를 변경할 수 있습니다.

데이터 개체를 매개 변수화하려면 데이터 개체 탭에서 매개 변수로 지정을 선택합니다. 데이터 개체 탭의 속성이 변경됩니다.

데이터 개체를 매개 변수화하려면 리소스 유형 매개 변수를 작성하거나 이미 작성한 리소스 매개 변수를 찾습니다. 매개 변수 기본값은 모델 리포지토리의 실제 데이터 개체 이름입니다. 기본 매개 변수 값을 작성하는 경우 리포지토리의 데이터 개체 목록에서 실제 데이터 개체 이름을 선택합니다.

다음 이미지는 매개 변수로 데이터 개체를 지정하는 경우 데이터 개체 탭을 보여 줍니다.



데이터 개체 탭에는 매개 변수 기준 다음 옵션이 있습니다.

## 매개 변수

데이터 개체로 구성된 리소스 매개 변수의 이름입니다. 읽기 전용입니다.

## 설명

매개 변수에 대한 설명입니다. 읽기 전용입니다.

## 새로 만들기

리소스 매개 변수를 작성합니다. 모델 리포지토리에서 매개 변수 기본값에 대한 데이터 개체를 찾아 선택합니다.

## 찾아보기

리소스 매개 변수를 찾아 매개 변수를 선택합니다.

## 기본값

데이터 개체에 대해 구성된 리소스 매개 변수의 기본값입니다. 기본값은 모델 리포지토리의 실제 데이터 개체 이름 및 개체에 대한 경로입니다. 읽기 전용입니다.

# 읽기 변환 매개 변수

읽기 변환을 생성하는 위치에서 읽기 변환의 일부 속성 및 재사용 가능 실제 데이터 개체의 일부 속성을 매개 변수화할 수 있습니다.

실제 데이터 개체를 생성하는 경우 읽기 및 쓰기 속성을 구성합니다. 데이터 개체를 매핑에 추가하는 경우 **데이터 개체 매개 변수** 탭에 실제 데이터 개체의 읽기 속성에 대해 구성하는 매개 변수가 나타납니다.

실제 데이터 개체의 다음 읽기 속성에 대해 매개 변수를 구성할 수 있습니다.

- 제어 파일 디렉터리
- 제어 파일 이름
- 기본 소수 자릿수
- 구분자
- 플랫폼 파일 구분자
- 병합 파일 디렉터리
- 소스 파일 이름
- 소스 파일 디렉터리

매핑에 실제 데이터 개체를 추가한 후 읽기 변환의 **데이터 개체 매개 변수** 탭에서 매개 변수를 볼 수 있습니다. 이러한 매개 변수를 매핑 매개 변수로 노출하여 런타임 시 매개 변수를 재정의할 수 있습니다.

**참고:** 매개 변수화된 소스 내에 사용자 정의 매개 변수를 중첩할 수 없습니다. 소스 데이터 개체가 매개 변수화된 경우 사용자 정의 매개 변수를 매핑 매개 변수로 표시하여 런타임 시 매개 변수 값을 재정의할 수 없습니다. 매핑에는 기본값이 사용됩니다.

읽기 변환에 대해 다음 매핑 매개 변수를 구성할 수 있습니다.

- 연결(관계형)
- 데이터 개체
- 링크 확인 순서
- 리소스 이름(관계형)
- 테이블 소유자 이름(관계형)

매핑 **데이터 개체 매개 변수** 탭에서 이러한 매개 변수를 볼 수 있습니다.

## 제약 조건

제약 조건은 조건부 식으로, 데이터 행의 값이 이 식을 충족시켜야 합니다.

제약 조건을 설정할 경우 각 데이터 행에 대해 TRUE로 평가되는 식을 입력합니다.

데이터 통합 서비스는 관계형 소스, 논리적 데이터 개체, 실제 데이터 개체 또는 가상 테이블에서 제약 조건을 읽을 수 있습니다. 재사용 가능한 실제 데이터 개체에 대한 제약 조건을 설정하려면 사용자 지정 데이터 개체를 생성하십시오.

데이터 통합 서비스가 제약 조건을 읽으면서, 적용된 최적화 방법에 기반하는 데이터 행에 대해 TRUE로 평가되지 않는 행을 삭제할 수 있습니다.

제약 조건을 설정하기 전에 소스 데이터가 제약 조건에 의해 설정된 조건을 충족하는지 확인해야 합니다. 예를 들어 소스 데이터베이스에 AGE 열이 있으며, AGE < 70인 행만 있는 것으로 나타납니다. 이 경우 소스 데이터베이스에 대해 AGE < 70인 제약 조건을 설정할 수 있습니다. 데이터 통합 서비스는 제약 조건 AGE < 70을 사용하여 소스 데이터베이스에서 레코드를 읽습니다. 데이터 통합 서비스가 AGE >= 70인 레코드를 읽은 경우 AGE >= 70인 행을 삭제합니다.

데이터베이스에 연결할 때 데이터베이스 환경에 대한 제약 조건을 설정하는 SQL 명령을 데이터베이스에서 사용할 수 있습니다. 데이터 통합 서비스에서 데이터베이스에 연결할 때마다 연결 환경 SQL을 실행합니다.

## 읽기 변환 작성

읽기 변환을 작성하는 경우 변환을 작성하는 리소스 위치에 따라 다음 방법 중 하나를 선택합니다.

**모델 리포지토리의 데이터 개체에서 변환을 작성합니다.**

다음 단계를 수행하여 모델 리포지토리의 데이터 개체에서 읽기 변환을 작성합니다.

1. 편집기에서 매핑을 엽니다.
2. 데이터 개체를 **Object Explorer** 보기에서 편집기로 끌어옵니다.
3. 읽기를 선택하고 **확인**을 클릭합니다.  
매핑의 읽기 변환에는 데이터 개체의 포트 및 속성이 포함됩니다.

**매핑 편집기를 사용하여 변환을 작성합니다.**

세부 읽기 변환 설정을 구성하려면 이 방법을 사용합니다. 읽기 변환을 매개 변수에 기반하도록 하려는 경우에도 이 방법을 사용할 수 있습니다.

매핑 편집기에서 읽기 변환을 작성하려면 [“매핑 편집기에서 읽기 변환 작성” 페이지 529](#)를 참조하십시오.

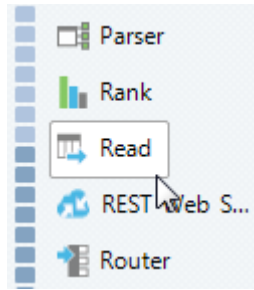
## 매핑 편집기에서 읽기 변환 작성

읽기 변환을 작성하여 매핑의 데이터 소스, 열 메타데이터 및 속성을 나타낼 수 있습니다.

다음 단계를 수행합니다.

1. 다음 방법 중 하나를 사용하여 읽기 변환을 작성합니다.
  - 매핑 편집기에서 마우스 오른쪽 단추를 클릭하고 **변환 추가**를 선택합니다.  
**변환 추가** 대화 상자가 열립니다.  
읽기 변환을 선택하고 **다음**을 클릭합니다.
  - 매핑 팔레트에서 아래로 스크롤하여 읽기 변환 아이콘을 찾아 두 번 클릭합니다.

다음 이미지는 읽기 변환 아이콘을 보여 줍니다.



새 읽기 변환 대화 상자가 열립니다.

2. 플랫폼 파일, 관계형 리소스 또는 사용자 지정된 데이터 개체를 소스로 사용하려면 다음 단계를 수행합니다.
  - a. **실제 데이터 개체**를 데이터 개체 유형으로 선택합니다.
  - b. **찾아보기**를 클릭하여 플랫폼 파일, 관계형 리소스 또는 사용자 지정된 데이터 개체를 선택합니다.  
데이터 개체 선택 창이 열립니다.
  - c. 데이터 개체를 선택하고 **확인**을 클릭합니다.
  - d. 필요한 경우 런타임 시 소스에서 데이터 개체 열을 가져오도록 변환을 구성합니다. **런타임 시 데이터 소스에서 데이터 개체 열 가져오기**를 선택합니다.  
매핑 실행 시 데이터 통합 서비스가 읽기 변환에 대한 열 메타데이터를 새로 고칩니다.
3. 매개 변수를 소스로 사용하려면 다음 단계를 수행합니다.
  - a. **매개 변수를 사용하여 작성**을 선택합니다.
  - b. **새로 만들기**를 클릭하여 새 매개 변수를 작성하거나 **찾아보기**를 클릭하여 기존 매개 변수를 선택합니다.
  - c. 매개 변수를 선택하고 **확인**을 클릭합니다.
  - d. 필요한 경우 런타임 시 소스에서 데이터 개체 열을 가져오도록 변환을 구성합니다. **런타임 시 데이터 소스에서 데이터 개체 열 가져오기**를 선택합니다.  
매핑 실행 시 데이터 통합 서비스가 읽기 변환에 대한 열 메타데이터를 새로 고칩니다.
4. 논리적 데이터 개체를 소스로 사용하려면 다음 단계를 수행합니다.
  - a. **논리적 데이터 개체**를 데이터 개체 유형으로 선택합니다.
  - b. **찾아보기**를 클릭하여 데이터 개체를 선택한 다음 **확인**을 클릭합니다.
5. 필요한 경우 읽기 변환에 대한 이름을 입력합니다.
6. **마침**을 클릭합니다.

## 제 37 장

# 관계형-계층 변환

이 장에 포함된 항목:

- [관계형-계층 변환 개요, 531](#)
- [예제 - 관계형-계층 변환, 531](#)
- [입력 관계형 포트 및 개요 보기, 533](#)
- [관계형-계층 변환 포트, 534](#)
- [스키마 참조, 534](#)
- [관계형-계층 변환 개발, 535](#)

## 관계형-계층 변환 개요

관계형-계층 변환은 관계형 입력을 처리하여 계층 출력으로 변환합니다. 관계형-계층 변환은 입력 포트에서 관계형 입력을 읽고 변환 출력 포트에서 데이터를 계층 출력으로 변환합니다. 관계형 입력을 계층 출력으로 변환하려면 스키마 개체를 사용하여 계층 구조를 정의합니다.

관계형-계층 변환 마법사를 사용하여 관계형 입력 포트를 반영하는 계층 구조를 작성할 수 있습니다. 변환 **개요** 보기에서 계층 출력 포트에 대한 매핑을 볼 수 있습니다.

변환을 작성한 후에는 데이터를 계층 출력 포트에서 매핑의 다른 변환으로 전달할 수 있습니다.

관계형 모델에서 각 테이블 스키마는 각 행을 고유하게 식별하기 위해 기본 키라는 열을 식별합니다. 테이블의 각 행과 다른 테이블의 행 사이 관계는 외래 키를 사용하여 식별합니다. 마법사는 변환을 작성할 때 키를 생성합니다. 자동 생성된 변환을 변경하고 포트를 추가, 편집 또는 삭제할 수 있습니다.

입력 관계형 포트를 계층의 노드에 연결할 수 있습니다. 계층의 관련 요소 또는 특성에서 입력의 관계형 그룹으로 기본 키를 연결합니다. 기본 키는 관계형 테이블에서 각 행을 식별합니다.

계층의 관련 요소 또는 특성에서 입력의 관계형 그룹으로 외래 키를 연결합니다. 관계형 입력의 외래 키는 한 테이블에서 다른 테이블의 기본 키를 가리키는 열입니다.

입력 관계형 포트 및 계층 노드는 호환되는 데이터 유형이어야 합니다.

## 예제 - 관계형-계층 변환

Electronics Superstore라는 회사의 회계 부서에서는 직원들의 급여를 처리해야 합니다. 그들은 관계형 데이터베이스에 저장된 직원 데이터를 해당 지불 시스템에서 처리할 수 있는 계층 형식으로 변환해야 합니다.

매핑에서 직원 이름, 직원 ID, 직원 주소, 직원 은행 계좌 데이터와 같은 직원들의 세부 정보를 입력하고 이러한 세부 정보를 사용 가능한 계층 형식으로 출력하는 관계형-계층 변환을 사용해야 합니다.

관계형 입력에서 **Bank\_ID** 요소는 직원 테이블에서 기본 키이고 은행 테이블에서 외래 키입니다.

Employee_ID	Last_Name	First_Name	주소	Bank_ID	Bank Account
9173327437	Sandrine	Jacques	74 Mobile Avenue	74845	8723487234
9174562342	Race	Tom	266 Crouse St.	9234734	45324734
8484526471	Jones	Charles	3815 LaValle Boulevard	389236	234638437
7023847265	Smith	Delilah	193 Short Drive	74845	8723463432
9174596725	Frederick	George	17 Serenity Road	9234734	6342636699

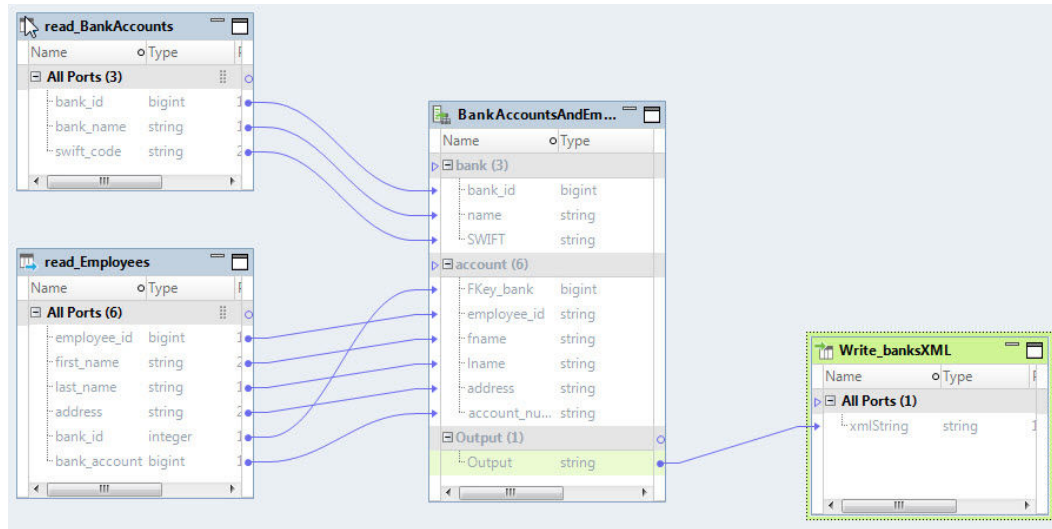
Bank_ID	Bank_Name	SWIFT_Code
74845	National Bank	9173327
9234734	International Bank	9174562
389236	Star National Bank	8484526

계층 형식의 지불 출력에서, 요소는 테이블에서 결합되어 있습니다.

```
<banks>
  <bank name="National Bank" SWIFT="9173327">
    <account id="8723487234">
      <employee_id>9173327437</employee_id>
      <fname>Sandrine</fname>
      <lname>Jacques</lname>
      <address>74 Mobile Avenue</address>
    </account>
    <account id="8723463432">
      <employee_id>9082745558</employee_id>
      <fname>Delilah</fname>
      <lname>Smith</lname>
      <address>193 Short Drive</address>
    </account>
  </bank>
  <bank name="International Bank" SWIFT="9174562">
    <accounts>
      <account id="45324734">
        <employee_id>5534398889</employee_id>
        <fname>Race</fname>
        <lname>Tom</lname>
        <address>266 Crouse St.</address>
      </account>
      <account id="6342636699">
        <employee_id>9174596725</employee_id>
        <fname>Frederick</fname>
        <lname>George</lname>
        <address>17 Serenity Road</address>
      </account>
    </accounts>
  </bank>
  <bank name="Star National Bank" SWIFT="8484526">
    <accounts>
      <account id="234638437">
        <employee_id>8484526471</employee_id>
        <fname>Jones</fname>
        <lname>Charles</lname>
        <address>3815 LaValle Boulevard</address>
      </account>
    </accounts>
  </bank>
</banks>
```



다음 이미지는 이 예제의 매핑을 보여 줍니다.



이 매핑에는 다음 개체가 포함됩니다.

**Read\_BankAccounts**

은행 데이터가 들어 있는 소스입니다.

**Read\_Employees**

직원 데이터가 들어 있는 소스입니다.

**BankAccountsAndEmployees\_To\_PaymentsSystemXML**

직원 정보와 은행 계좌 정보가 들어 있는 관계형 입력을 지불 시스템에서 처리할 수 있는 XML 형식으로 변환하는 관계형-계층 변환입니다.

**Write\_BanksXML**

매핑을 실행할 때마다 변환된 데이터를 저장하는 파일에 대한 대상 경로입니다.

매핑에서 **Read\_BankAccount** 및 **Read\_Employees** 파일을 사용하여 관계형 입력을 제공합니다. 매핑에서 **BankAccountsAndEmployees\_To\_PaymentsSystemXML** 변환을 사용하여 데이터를 처리하고 변환합니다. 그런 다음 매핑에서 **Write\_BanksXML** 플랫폼 파일에 나열된 대상 경로에 출력을 저장합니다.

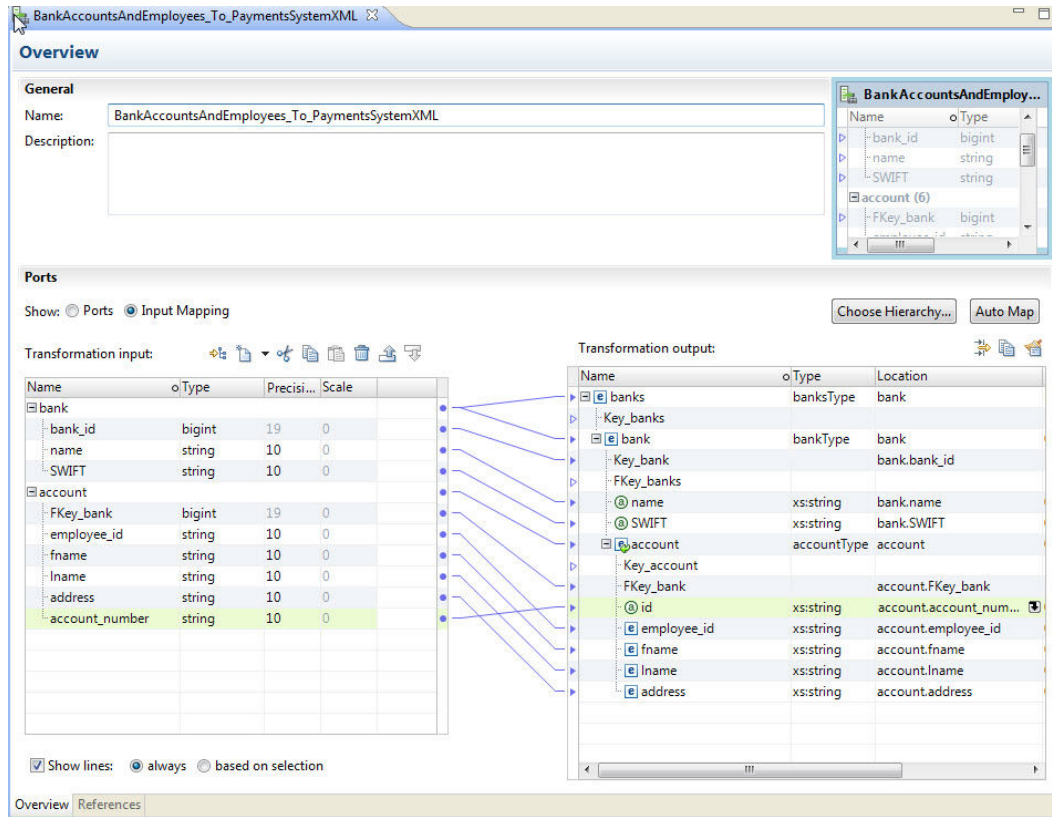
## 입력 관계형 포트 및 개요 보기

관계형 데이터를 계층 데이터로 변환하기 위해 마법사는 관계형 입력 포트를 반영하는 계층 구조를 작성합니다.

**개요** 보기를 사용하여 관계형 포트를 계층 포트에 연결할 수 있습니다.

관계형 입력과 계층 출력 간 링크를 보려면 **개요** 보기를 사용합니다. **입력 매핑**을 선택합니다. **포트** 패널이 **개요** 보기에 나타납니다.

다음 이미지는 **포트** 패널을 보여 줍니다.



포트 패널의 왼쪽에는 관계형 요소와 그룹이 포함된 **변환 입력** 영역이 있습니다. 오른쪽에는 계층 스키마 노드가 포함된 **변환 출력** 영역이 있습니다.

**변환 입력** 영역에서 포트를 작성하고 관계형 요소를 스키마 노드와 연결할 수 있습니다. 또한 스키마의 노드에서 **변환 입력** 영역의 빈 필드로 포인터를 끌어서 놓아 포트를 작성할 수도 있습니다. 관계형 포트를 스키마 노드에 연결하면 Developer 도구가 둘 사이 링크를 표시합니다.

## 관계형-계층 변환 포트

관계형-계층 변환 포트는 변환 **개요** 보기에 정의되어 있습니다.

관계형-계층 변환은 버퍼에서 관계형 데이터 출력을 읽을 수 있습니다. 출력 포트는 계층 데이터를 버퍼로 반환합니다.

## 스키마 참조

관계형-계층 변환에는 변환에서 출력 계층을 정의하기 위한 계층 스키마가 필요합니다. 변환에서 스키마를 사용하려면 스키마 참조를 정의합니다.

변환 **참조** 보기에서 변환 스키마 참조를 정의할 수 있습니다.

관계형-계층 변환은 모델 리포지토리의 스키마 개체를 참조합니다. 변환을 작성하기 전에 스키마 개체가 리포지토리에 있을 수 있습니다.

스키마는 다른 스키마를 참조할 수 있습니다. **참조** 보기는 관계형-계층 변환에서 참조하는 각 스키마에 대한 네임스페이스와 접두사를 표시합니다.

## 관계형-계층 변환 개발

새 변환 마법사를 사용하여 관계형-계층 변환을 자동 생성합니다. 스키마 또는 계층 샘플 파일을 선택하여 출력 계층을 정의합니다.

### 관계형-계층 변환 작성

1. Developer tool에서 **파일 > 새로 만들기 > 변환**을 클릭합니다.
2. 관계형-계층 변환을 선택하고 **다음**을 클릭합니다.
3. 변환의 이름을 입력하고 변환을 배치할 모델 리포지토리 위치를 찾은 다음 **다음**을 클릭합니다.
4. 스키마를 선택하려면 다음 방법 중 하나를 선택합니다.
  - 모델 리포지토리의 스키마를 사용하여 출력 계층을 정의하려면 **스키마 개체** 필드 근처에서 리포지토리의 스키마 파일을 찾아 선택합니다.
  - 스키마 파일을 가져오려면 **새 스키마 개체 작성**을 클릭합니다. **새 스키마 개체** 창에서 스키마 파일을 찾아 선택하거나 샘플 계층 파일에서 스키마를 작성하도록 선택할 수 있습니다.
5. 출력 계층의 루트를 선택합니다. **계층 루트** 대화 상자에서 출력 계층 파일에 대한 루트 요소인 스키마의 요소를 선택합니다. 루트 개체 선택에 도움이 되도록 샘플 계층 파일을 추가할 수 있습니다. 샘플 파일을 추가하려면 **샘플 파일** 필드 근처에서 파일 시스템의 파일을 찾아 선택합니다.
6. **마침**을 클릭합니다.  
마법사가 리포지토리에 변환을 작성합니다.

### 포트 작성

**개요** 보기에서 포트를 구성합니다.

1. 매핑을 보려면 **개요** 보기 **포트** 영역에서 **입력 매핑**을 선택합니다.
2. 입력 포트 유형, 전체 자릿수 및 소수 자릿수를 선택합니다.
3. **포트** 그리드의 트리를 확장합니다. 왼쪽의 **변환 입력** 패널에는 관계형 입력이 표시되고 오른쪽의 **변환 출력** 패널에는 예상 계층 출력이 표시됩니다.
4. 노드를 루트로 정의하려면 **계층 선택**을 클릭합니다.  
Developer tool에는 루트 수준 및 루트 수준 아래의 노드만 **변환 입력** 영역에 표시합니다.
5. 포트를 계층 노드와 연결하는 선을 보려면 **선 표시**를 클릭합니다. 모든 연결 선을 표시하거나 선택한 포트의 선만 표시하도록 선택합니다.
6. 입력 그룹 또는 포트를 **변환 입력** 영역에 추가하려면 다음 방법 중 하나를 사용합니다.
  - **변환 출력** 영역에서 단순 또는 복합 요소를 **변환 입력** 영역의 빈 열로 끌어옵니다. 노드가 그룹 노드인 경우 Developer tool이 포트 없이 관계형 그룹을 추가합니다.

- 관계형 그룹을 추가하려면 행을 선택하고 마우스 오른쪽 단추를 클릭하여 **새로 만들기 > 그룹**을 선택합니다.
  - 관계형 포트를 추가하려면 마우스 오른쪽 단추를 클릭하고 **새로 만들기 > 필드**를 선택합니다.
7. 포트의 위치에 대한 계층 노드 설정을 지우려면 다음 방법 중 하나를 사용합니다.
- **변환 출력** 영역에서 하나 이상의 노드를 선택하고 마우스 오른쪽 단추를 클릭한 다음 **지우기**를 선택합니다.
  - 관계형 입력 포트를 계층 노드에 연결하는 하나 이상의 선을 선택하고 마우스 오른쪽 단추를 클릭한 다음 **삭제**를 선택합니다.
8. 계층에서 출력 포트를 표시하려면 **계층으로 보기**를 클릭합니다. 각 하위 그룹은 상위 그룹 아래에 나타납니다.

## 제 38 장

# REST 웹 서비스 소비자 변환

이 장에 포함된 항목:

- [REST 웹 서비스 소비자 변환 개요, 537](#)
- [REST 웹 서비스 소비자 변환 구성, 539](#)
- [HTTP 메서드, 540](#)
- [REST 웹 서비스 소비자 변환 포트, 543](#)
- [REST 웹 서비스 소비자 변환 입력 매핑, 545](#)
- [REST 웹 서비스 소비자 변환 출력 매핑, 547](#)
- [REST 웹 서비스 소비자 변환의 고급 속성, 549](#)
- [REST 웹 서비스 소비자 변환 작성, 550](#)
- [배열이 포함된 JSON 응답 메시지 구문 분석, 551](#)

## REST 웹 서비스 소비자 변환 개요

REST 웹 서비스 소비자 변환은 웹 서비스 클라이언트로 REST 웹 서비스에 연결하여 데이터에 액세스하거나 데이터를 변환하는 활성 변환입니다. REST 웹 서비스에 연결하려면 REST 웹 서비스 소비자 변환을 사용합니다. REST 웹 서비스 소비자 변환은 REST 웹 서비스에 요청을 전송하고 REST 웹 서비스의 응답을 수신합니다.

REST 웹 서비스 소비자 변환은 변환 또는 HTTP 연결에서 정의한 URL을 통해 웹 서비스에 연결합니다. HTTPS 연결도 사용할 수 있습니다. REST 웹 서비스 소비자 변환은 TLS 1.2, TLS 1.1 또는 TLS 1.0을 사용할 수 있습니다.

REST 웹 서비스에는 웹 서비스가 지원하는 각 작업에 대한 HTTP 메서드가 포함됩니다. 데이터 통합 서비스는 REST 웹 서비스에 연결하여 데이터를 가져오거나 게시하거나 배치하거나 삭제하는 요청을 전송할 수 있습니다. 요청은 개별 리소스 또는 리소스 컬렉션에 따라 작동할 수 있습니다. 데이터 통합 서비스는 요청 메시지를 전송한 후에 웹 서비스로부터 응답 메시지를 수신합니다.

요청 및 응답 메시지에는 계층을 형성할 수 있는 요소가 있는 XML 또는 JSON 데이터가 포함됩니다. 요청 또는 응답 메시지에 여러 번 발생하는 요소가 포함되는 경우 요소의 그룹에 의해 XML 또는 JSON 계층의 수준이 형성됩니다. 그룹은 한 수준이 다른 수준 안에 중첩될 경우에 연결됩니다.

REST 웹 서비스 소비자 변환에서 메서드 입력과 메서드 출력은 요청 및 응답 메시지의 구조를 정의합니다. 메서드 입력 및 메서드 출력에는 메시지 요소를 입력 및 출력 포트에 매핑하는 방법을 정의하는 매핑이 포함됩니다.

REST 웹 서비스 소비자 변환은 프록시 서버를 지원합니다. Microsoft SharePoint 응용 프로그램을 REST 웹 서비스 소비자 변환에 연결할 수도 있습니다.

## 예

한 온라인 매장에서 제품 데이터베이스에 대한 리소스를 정의합니다. 이 데이터베이스는 각 제품을 제품 번호로 식별합니다.

웹 서비스 클라이언트는 REST 웹 서비스를 통해 제품 세부 정보에 액세스합니다. 웹 서비스는 다음 URL을 사용합니다.

`http://www.HypoStores.com/products/ProductDetails`

이때 설명 및 단위 가격 등 특정 제품에 대한 세부 정보를 검색하고 세부 정보를 매핑의 변환 다운스트림에 전달하려면 제품에 대한 세부 정보를 검색하고 이러한 세부 정보를 다른 변환에 전달하는 REST 웹 서비스 소비자 변환을 작성해야 합니다.

다음 테이블에는 구성해야 하는 변환 세부 정보가 표시되어 있습니다.

변환 세부 정보	값
HTTP 메서드	Get
기본 URL	<code>http://www.HypoStores.com/products/ProductDetails</code>
입력 인수 포트	Part_No
출력 포트	설명, Unit_Price
메서드 출력	<응답 메시지의 구조입니다.>

메서드 출력에는 응답 메시지의 요소를 출력 포트에 매핑하는 방법을 정의하는 출력 매핑이 포함됩니다.

데이터 통합 서비스는 웹 서비스에 요청을 전송할 때 인수 포트의 값을 기본 URL에 추가합니다. 예를 들어 제품 0716에 대한 세부 정보를 검색하려는 경우 데이터 통합 서비스는 다음 URL을 사용합니다.

`http://www.HypoStores.com/products/ProductDetails?Part_No=0716`

데이터 통합 서비스는 응답을 수신할 때 응답 메시지의 제품 설명 및 단위 가격을 출력 포트에 대한 데이터로 변환합니다.

Part\_No를 매개 변수로 전달하여 매핑을 실행하는 중간에 값을 대체할 수도 있습니다.

## REST 웹 서비스 소비자 변환 프로세스

REST 웹 서비스 소비자 변환은 입력 포트 및 메서드 입력의 데이터를 바탕으로 요청 메시지를 작성합니다. 또한 메서드 출력에 따라 응답 메시지의 요소를 출력 포트에 대한 데이터로 변환합니다.

REST 웹 서비스 소비자 변환의 입력 포트에는 매핑의 업스트림 변환으로부터 수신한 관계형 데이터가 포함됩니다. 데이터 통합 서비스는 메서드 입력을 사용하여 입력 포트의 데이터를 요청 메시지의 요소로 변환합니다.

데이터 통합 서비스는 변환 속성 또는 HTTP 연결에서 구성된 기본 URL을 읽어 웹 서비스에 연결합니다. 또한 URL 포트 또는 인수 포트의 값을 기본 URL에 추가하여 사용자가 가져오거나 게시하거나 배치하거나 삭제하려는 리소스를 식별합니다.

데이터 통합 서비스는 응답을 수신한 다음 응답 메시지의 데이터를 변환의 출력 포트에 전달합니다. 데이터 통합 서비스는 메서드 출력의 구성에 따라 데이터를 전달합니다. 출력 포트에는 관계형 데이터가 포함됩니다. 데이터 통합 서비스는 출력 포트의 데이터를 매핑의 다운스트림 변환 또는 대상으로 전송합니다.

# REST 웹 서비스 소비자 변환 구성

REST 웹 서비스 소비자 변환을 작성할 때 HTTP 메서드를 선택하고 메서드 입력 및 출력을 정의합니다. Get 메서드를 선택한 경우에는 메서드 입력을 정의하지 않습니다.

HTTP 요청 메시지의 입력 요소는 입력 포트에 매핑됩니다. HTTP 요청 메시지의 출력 요소는 출력 포트에 매핑됩니다. Developer tool에서는 첫 번째 수준 요소에 대한 포트가 작성됩니다.

변환을 구성할 때 다음 태스크를 완료하십시오.

1. HTTP 메서드를 선택합니다.
2. 요청 및 응답 메시지의 헤더 및 본문에 있는 요소를 나타내는 포트를 구성합니다.
3. 입력 매핑을 구성합니다.
4. 출력 매핑을 구성합니다.
5. 웹 서비스에 대한 연결 및 URL과 같은 고급 속성을 구성합니다.

REST 웹 서비스에 인증이 필요한 경우 HTTP 연결 개체를 작성합니다.

## 메시지 구성

데이터 통합 서비스는 REST 웹 서비스 소비자 변환에서 구성한 메서드 입력 및 출력과 포트에 따라 요청 메시지를 생성하고 응답 메시지를 해석합니다.

입력 포트는 요청 메시지의 서로 다른 부분을 나타냅니다. 검색하거나 변경할 리소스를 식별하는 입력 포트를 추가할 수 있습니다. 또한 요청 메시지에서 HTTP 헤더, 쿠키 정보 및 요소를 나타내는 입력 포트를 추가할 수도 있습니다.

출력 포트는 매핑의 다운스트림 변환 또는 대상으로 전송할 응답 메시지의 요소를 나타냅니다. 응답 메시지에서 HTTP 헤더, 쿠키 정보, 응답 코드 및 요소를 나타내는 출력 포트를 추가할 수 있습니다.

## 리소스 식별

데이터 통합 서비스는 HTTP 요청의 리소스를 식별하기 위해 특정 입력 포트의 값을 기본 URL에 추가합니다. 기본 URL은 HTTP 연결 또는 변환 속성에 정의합니다. URL 또는 인수 포트를 사용하여 특정 리소스를 식별할 수 있습니다.

웹 서비스가 고유한 문자열을 통해 리소스를 식별하는 경우 URL 포트를 사용합니다.

예를 들어 HypoStores REST 웹 서비스는 다음 URL을 통해 제품 번호로 제품을 식별합니다.

`http://www.HypoStores.com/products/ProductDetails/<Part_No>`

제품을 식별하려면 다음 변환 세부 정보를 정의합니다.

1. 기본 URL을 다음 URL로 설정합니다.  
`http://www.HypoStores.com/products/ProductDetails`
2. URL 포트를 정의하고 URL 포트를 통해 제품 번호를 변환에 전달합니다.

매핑이 제품 번호 500을 URL 포트에 전달할 경우 데이터 통합 서비스는 다음 URL을 요청 메시지에 사용합니다.

`http://www.HypoStores.com/products/ProductDetails/500`

웹 서비스가 인수를 통해 리소스의 위치를 식별하는 경우에는 인수 포트를 사용합니다.

예를 들어 제품 번호를 "Part\_No" 인수를 통해 HypoStores REST 웹 서비스에 전달하려고 합니다.

제품을 식별하려면 다음 변환 세부 정보를 정의합니다.

1. 기본 URL을 다음 URL로 설정합니다.

`http://www.HypoStores.com/products/ProductDetails`

2. 인수 이름 "Part\_No"를 사용하여 인수 포트를 작성하고 이 인수 포트를 사용하여 제품 번호를 변환에 전달합니다.

매핑이 제품 번호 600을 URL 포트에 전달할 경우 데이터 통합 서비스는 다음 URL을 요청 메시지에 사용합니다.

`http://www.HypoStores.com/products/ProductDetails?Part_No=600`

여러 인수를 정의하려면 여러 인수 포트를 작성합니다. 데이터 통합 서비스에서는 앰퍼샌드(&)를 사용하여 각 인수를 구분합니다.

예를 들어 REST 웹 서비스에서 직원 세부 정보를 검색하고 직원의 이름과 성을 "First\_Name" 및 "Last\_Name" 인수를 통해 전달하려고 합니다. 그러려면 "First\_Name" 및 "Last\_Name"이라는 인수 이름을 사용하여 인수 포트를 작성합니다. 매핑이 이름 "John Smith"를 변환에 전달하면 데이터 통합 서비스는 다음과 같은 URL을 요청 메시지에 사용합니다.

`http://www.HypoStores.com/employees/EmpDetails?First_Name=John&Last_Name=Smith`

URL 또는 인수 포트를 지정하지 않을 경우 데이터 통합 서비스는 변환 속성 또는 HTTP 연결의 기본 URL을 사용하여 리소스를 식별합니다. HTTP 연결의 기본 URL은 변환의 기본 URL을 재정의합니다.

## HTTP 메서드

REST 웹 서비스 소비자 변환을 작성할 때는 데이터 통합 서비스에서 요청 메시지에 사용할 HTTP 메서드를 선택해야 합니다. 변환을 작성한 후에는 HTTP 메서드를 변경할 수 없습니다.

다음 HTTP 메서드 중 하나를 사용하도록 변환을 구성합니다.

### Get

웹 서비스에서 리소스 또는 리소스 컬렉션을 검색합니다. 예를 들어 제품의 테이블을 검색하거나 단일 제품에 대한 정보를 검색할 수 있습니다.

### Post

데이터를 웹 서비스로 전송합니다. Post 메서드는 리소스 또는 리소스 컬렉션을 작성할 때 사용됩니다. 예를 들어 신규 매장의 트랜잭션 세부 정보를 추가할 수 있습니다.

### Put

리소스 또는 리소스 컬렉션을 바꿉니다. 데이터가 존재하지 않는 경우에는 Put 메서드가 데이터를 게시합니다. 예를 들어 고객의 배송 주소를 업데이트할 수 있습니다.

### Delete

리소스 또는 리소스 컬렉션을 삭제합니다. 예를 들어 조직에서 더 이상 일하지 않는 직원의 레코드를 삭제할 수 있습니다.

## HTTP Get 메서드

데이터 통합 서비스에서는 REST 웹 서비스에서 데이터를 검색할 때 HTTP Get 메서드를 사용합니다. Get 메서드는 리소스 또는 리소스 컬렉션을 검색할 때 사용됩니다.

Get 메서드를 사용하도록 REST 웹 서비스 소비자 변환을 구성하는 경우 입력 포트, 메서드 출력 및 출력 포트를 구성해야 합니다. 메서드 입력은 구성하지 않습니다.



## 예

HypoStores 제품 데이터베이스에서 제품 번호 500에 대한 설명 및 가격을 검색하려고 합니다. 웹 서비스에서는 다음 URL을 사용하여 제품을 식별합니다.

`http://www.HypoStores.com/products/ProductDetails?Part_No=<Part_No>`

다음 기본 URL을 입력합니다.

`http://www.HypoStores.com/products/ProductDetails`

다음 테이블에는 정의할 수 있는 입력 포트가 표시되어 있습니다.

포트 유형	인수 이름	입력 값
인수	Part_No	500

다음 테이블에는 정의할 수 있는 출력 포트가 표시되어 있습니다.

포트 유형	포트 이름	반환 값
출력	Part_Desc	...<desc>ACME 볼펜, 12pk, 검정, 0.7mm</desc>...
출력	Price_USD	...<price>9.89</price>...

## HTTP Post 메서드

데이터 통합 서비스에서는 REST 웹 서비스에 데이터를 전송할 때 HTTP Post 메서드를 사용합니다. Post 메서드가 수행하는 실제 기능은 웹 서비스를 통해 결정됩니다. Post 메서드를 사용하여 리소스 또는 리소스 컬렉션을 작성할 수도 있습니다.

Post 메서드를 사용하도록 REST 웹 서비스 소비자 변환을 구성하는 경우 입력 포트, 메서드 입력, 메서드 출력 및 출력 포트를 구성해야 합니다.

## 예

HypoStores 제품 데이터베이스에 새 제품 501을 게시하려고 합니다. 웹 서비스에서는 제품 501에 대해 다음 URL을 사용합니다.

`http://www.HypoStores.com/products/ProductDetails/501`

다음 기본 URL을 입력합니다.

`http://www.HypoStores.com/products/ProductDetails`

다음 테이블에는 정의할 수 있는 입력 포트가 표시되어 있습니다.

포트 유형	포트 이름	입력 값
URL	URL_Part_No	501
입력	Part_Desc	ACME 볼펜, 12pk, 검정, 0.5mm
입력	Price_USD	9.89

다음 테이블에는 정의할 수 있는 출력 포트가 표시되어 있습니다.

포트 유형	포트 이름	반환 값
출력	응답	<웹 서비스에서 반환하는 응답>

## HTTP Put 메서드

데이터 통합 서비스에서는 REST 웹 서비스를 통해 데이터를 업데이트할 때 HTTP Put 메서드를 사용합니다.

Put 메서드는 리소스 또는 리소스 컬렉션을 업데이트할 때 사용됩니다. 데이터가 존재하지 않을 경우에는 데이터 통합 서비스가 리소스 또는 리소스 컬렉션을 작성합니다.

Put 메서드를 사용하도록 REST 웹 서비스 소비자 변환을 구성하는 경우 입력 포트, 메서드 입력, 메서드 출력 및 출력 포트를 구성해야 합니다.

### 예

HypoStores 제품 데이터베이스의 제품 501에 대한 단위 가격을 업데이트하려고 합니다. 웹 서비스에서는 제품 501에 대해 다음 URL을 사용합니다.

`http://www.HypoStores.com/products/ProductDetails/501`

다음 기본 URL을 입력합니다.

`http://www.HypoStores.com/products/ProductDetails`

다음 테이블에는 정의할 수 있는 입력 포트가 표시되어 있습니다.

포트 유형	포트 이름	입력 값
URL	URL_Part_No	501
입력	Price_USD	9.99

다음 테이블에는 정의할 수 있는 출력 포트가 표시되어 있습니다.

포트 유형	포트 이름	반환 값
출력	응답	<웹 서비스에서 반환하는 응답>

## HTTP Delete 메서드

데이터 통합 서비스에서는 REST 웹 서비스를 통해 데이터를 제거할 때 HTTP Delete 메서드를 사용합니다.

Delete 메서드는 리소스 또는 리소스 컬렉션을 제거할 때 사용됩니다.

Delete 메서드를 사용하도록 REST 웹 서비스 소비자 변환을 구성하는 경우 입력 포트, 메서드 입력, 메서드 출력 및 출력 포트를 구성해야 합니다.

### 예

HypoStores 제품 데이터베이스에서 제품 번호 502를 삭제하려고 합니다. 웹 서비스에서는 다음 URL을 사용하여 제품을 식별합니다.

`http://www.HypoStores.com/products/ProductDetails?Part_No=<Part_No>`

다음 기본 URL을 입력합니다.

<http://www.HypoStores.com/products/ProductDetails>

다음 테이블에는 정의할 수 있는 입력 포트가 표시되어 있습니다.

포트 유형	인수 이름	입력 값
인수	Part_No	502

다음 테이블에는 정의할 수 있는 출력 포트가 표시되어 있습니다.

포트 유형	포트 이름	반환 값
출력	응답	<웹 서비스에서 반환하는 응답>

## REST 웹 서비스 소비자 변환 포트

REST 웹 서비스 소비자 변환에는 여러 개의 입력 포트와 여러 개의 출력 포트가 포함될 수 있습니다. XML 또는 JSON 계층의 구조에 따라 포트를 그룹으로 작성합니다.

변환 포트를 볼 때 XML 또는 JSON 계층을 보지 않아도 될 경우에는 포트를 표시합니다. 포트를 표시할 때 그룹을 정의하고 포트를 정의하고 메서드 입력 및 출력의 요소를 입력 및 출력 포트에 매핑할 수 있습니다.

REST 웹 서비스 소비자 변환에는 여러 개의 입력 그룹과 여러 개의 출력 그룹이 포함될 수 있습니다. 포트를 작성할 때 그룹을 작성하고 포트를 그룹에 추가합니다. XML 또는 JSON의 입력 또는 출력 계층 구조에 따라 그룹 계층의 포트를 정의합니다. 키를 추가하여 하위 그룹을 상위 그룹에 연결합니다.

계층의 최하위 그룹을 제외한 모든 그룹에는 기본 키가 있어야 합니다. 루트 그룹을 제외한 계층의 모든 그룹에는 외래 키가 있어야 합니다.

변환에는 RequestInput이라는 루트 입력 그룹이 포함됩니다. 이 루트 입력 그룹에 기본 키를 추가해야 합니다. 키는 문자열, bigint 또는 정수여야 합니다. 루트 입력 그룹의 모든 포트를 통과 포트 구성할 수 있습니다.

요소를 포트에 매핑하려면 위치 열에서 필드를 클릭하고 위치 선택 대화 상자에서 계층을 확장합니다. 그런 다음 계층에서 요소를 선택합니다.

### 입력 포트

입력 포트는 웹 서비스에 전달할 업스트림 변환 또는 소스의 데이터를 나타냅니다. 여러 입력 포트를 구성할 수 있습니다. 각 입력 포트는 요청 메시지의 요소에 매핑됩니다.

입력 포트를 추가하려면 입력 그룹을 선택하고 새로 만들기 단추 옆의 화살표를 클릭한 다음 필드를 선택합니다.

### 출력 포트

출력 포트는 다운스트림 변환 또는 대상에 전달할 응답 메시지의 요소를 나타냅니다. 여러 출력 포트를 구성할 수 있습니다. 각 출력 포트는 응답 메시지의 요소에 매핑됩니다.

출력 포트를 추가하려면 출력 그룹을 선택하고 새로 만들기 단추 옆의 화살표를 클릭한 다음 필드를 선택합니다.

## 통과 포트

통과 포트는 데이터를 변경하지 않고 변환을 통해 데이터를 전달합니다. 루트 입력 그룹의 모든 포트를 통과 포트 구성할 수 있습니다.

통과 포트를 추가하려면 포트를 루트 입력 그룹에 추가합니다. 그런 다음 마우스 오른쪽 단추로 포트를 클릭하고 **매핑**을 선택합니다.

## 인수 포트

리소스에 대한 URL이 인수를 사용할 경우 인수 포트를 사용하여 리소스를 식별할 수 있습니다. 인수 포트는 루트 입력 그룹에 추가합니다.

인수 포트에는 포트 이름과 인수 이름이 포함됩니다. 인수 이름에 포트 이름으로 허용되지 않는 문자가 포함되는 경우 포트 이름과 다른 인수 이름을 입력합니다. 예를 들어 인수 "Cust-ID"를 웹 서비스에 전달하려고 합니다. 그런데 데이터 통합 서비스에서는 포트 이름에 대시 문자(-)를 사용할 수 없습니다. 이 경우 인수 이름으로 "Cust-ID"를 입력하고 포트 이름으로는 "CustID"를 입력합니다.

그러면 데이터 통합 서비스가 인수 이름과 각 인수 포트의 값을 이름=값 쌍으로 기본 URL에 추가합니다. 여러 인수 포트를 구성할 수 있습니다. 데이터 통합 서비스는 요청의 여러 인수를 앰퍼샌드 문자(&)로 분리합니다.

예:

```
http://www.HypoStores.com/customers/CustDetails?Last_Name=Jones&First_Name=Mary
```

인수 포트와 URL 포트를 변환에서 정의할 경우 데이터 통합 서비스가 URL 포트 값을 기본 URL에 추가하고 인수 이름과 값을 그 뒤에 추가합니다.

인수 포트를 추가하려면 루트 입력 그룹을 마우스 오른쪽 단추로 클릭하고 **새로 만들기 > 인수 포트**를 선택합니다. 인수 이름과 포트 이름을 입력합니다.

## URL 포트

URL 포트를 사용하면 정적 URL을 통해 리소스를 식별할 수 있습니다. 리소스를 식별하기 위해 데이터 통합 서비스에서 URL 포트 값을 기본 URL에 추가합니다.

예:

```
http://www.HypoStores.com/products/ProductDetails/<URL_port_value>
```

URL 포트를 루트 입력 그룹에 추가합니다.

여러 URL 포트를 구성할 수 있습니다. 데이터 통합 서비스에서는 슬래시(/) 문자를 사용하여 각 URL 포트의 값을 구분합니다. 변환에 URL 포트 및 인수 포트를 정의한 경우 데이터 통합 서비스가 URL 포트 값을 기본 URL에 추가하고, 그 다음에 인수 이름과 값이 나옵니다.

URL 포트를 추가하려면 루트 입력 그룹에서 마우스 오른쪽 단추를 클릭하고 **새로 만들기 > URL 포트**를 선택합니다.

## HTTP 헤더 포트

HTTP 헤더 포트는 요청 메시지의 HTTP 헤더를 나타냅니다. 여러 개의 HTTP 헤더 포트를 구성할 수 있습니다.

요청에서 헤더 정보를 웹 서비스에 전달하려면 포트를 루트 입력 그룹에 추가합니다. 루트 입력 그룹에 대해 하나의 HTTP 헤더 포트를 구성할 수 있습니다. 루트 입력 그룹에 HTTP 헤더를 추가하는 경우 통과 포트 구성할 수 있습니다.

HTTP 헤더 포트에는 포트 이름과 HTTP 헤더 이름이 포함됩니다. HTTP 헤더 이름에 포트 이름에서 허용되지 않는 문자가 포함될 경우 포트 이름과 다른 HTTP 헤더 이름을 입력하십시오. 예를 들어 헤더 이름 "Content-

Type"을 웹 서비스에 전달하려고 합니다. 그런데 데이터 통합 서비스에서는 포트 이름에 대시 문자(-)를 사용할 수 없습니다. 이 경우 "Content-Type"을 HTTP 헤더 이름으로 입력하되 포트 이름으로는 "ContentType"을 입력하십시오.

HTTP 헤더 포트를 추가하려면 루트 입력 그룹을 마우스 오른쪽 단추로 클릭하고 **새로 만들기 > HTTP 헤더**를 선택합니다. 헤더 이름 및 포트 이름을 입력합니다.

## 쿠키 포트

쿠키 인증을 사용하도록 REST 웹 서비스 소비자 변환을 구성할 수 있습니다. 원격 웹 서비스 서버에서는 쿠키를 기반으로 웹 서비스 소비자 사용자를 추적합니다. 매핑이 웹 서비스를 여러 번 호출하는 경우 성능을 향상시킬 수 있습니다.

쿠키 정보를 요청의 웹 서비스로 전달하려면 포트를 루트 입력 그룹에 추가합니다. 루트 입력 그룹에 대해 하나의 쿠키 포트를 구성할 수 있습니다. 쿠키 포트를 루트 입력 그룹에 추가할 경우 해당 쿠키 포트를 통과 포트로 구성할 수 있습니다.

응답에서 쿠키 정보를 추출하려면 쿠키 포트를 출력 그룹에 추가합니다. 각 출력 그룹에 대해 하나의 쿠키 포트를 구성할 수 있습니다.

쿠키 포트를 웹 서비스 요청 메시지에 연결하면 웹 서비스 공급자가 응답 메시지에 쿠키 값을 반환합니다. 쿠키 값을 매핑의 다른 변환 다운스트림에 전달하거나 파일에 쿠키 값을 저장할 수 있습니다. 쿠키 값을 파일에 저장할 경우 쿠키를 REST 웹 서비스 소비자 변환에 대한 입력으로 구성할 수 있습니다.

쿠키 포트를 추가하려면 루트 입력 그룹을 마우스 오른쪽 단추로 클릭하고 **새로 만들기 > 기타 포트**를 선택합니다. 그런 다음 **쿠키**를 선택하고 **확인**을 클릭합니다.

## 출력 XML 포트

출력 XML 포트는 웹 서비스의 응답을 나타냅니다. 출력 XML 포트는 문자열 포트입니다.

출력 XML 포트를 출력 그룹에 추가합니다. 각 출력 그룹에 대해 하나의 출력 XML 포트를 구성할 수 있습니다.

루트 입력 그룹을 마우스 오른쪽 단추로 클릭하고 **새로 만들기 > 기타 포트**를 선택합니다. 그런 다음 **출력 XML**을 선택하고 **확인**을 클릭합니다.

## 응답 코드 포트

응답 코드 포트는 웹 서비스의 HTTP 응답 코드를 나타냅니다. 응답 코드 포트는 정수 포트입니다.

응답 코드 포트는 출력 그룹에 추가합니다. 각 출력 그룹에 대해 하나의 응답 코드 포트를 구성할 수 있습니다.

응답 코드 포트를 추가하려면 출력 그룹을 선택하고 루트 입력 그룹을 마우스 오른쪽 단추로 클릭한 다음 **새로 만들기 > 기타 포트**를 선택합니다. 그런 다음 **응답 코드**를 선택하고 **확인**을 클릭합니다.

# REST 웹 서비스 소비자 변환 입력 매핑

변환 포트를 볼 때 메서드 입력 계층을 보려면 입력 매핑을 표시합니다. 입력 매핑을 표시하면 입력 그룹을 정의하고 입력 포트를 정의하고 입력 포트를 메서드 입력 요소에 매핑할 수 있습니다.

입력 매핑에는 다음 영역이 포함됩니다.

### 포트

**포트** 영역에서 변환 입력 그룹 및 입력 포트를 작성합니다.

## 메서드 입력

**메서드 입력** 영역에는 REST 웹 서비스 소비자 변환이 웹 서비스에 전송하는 요청 메시지의 요소가 표시됩니다. 스키마 개체를 사용하여 변환을 작성하는 경우 스키마 개체가 메서드 입력 계층을 정의합니다.

입력 포트를 작성한 후 **포트** 영역의 입력 포트를 **메서드 입력** 영역의 요소에 매핑합니다. 입력 포트를 메서드 입력의 요소에 매핑하면 포트의 위치가 **메서드 입력** 영역의 위치 열에 나타납니다.

입력 계층의 첫 번째 수준을 매핑하기로 선택한 경우 **Developer tool**이 메서드 입력의 첫 번째 수준에 있는 노드를 입력 포트에 매핑합니다. **Developer tool**은 매핑을 수행할 포트도 작성합니다. 하나 이상의 다중 발생 하위 요소가 있는 다중 발생 상위 요소가 계층의 첫 번째 수준에 포함될 경우 **Developer tool**이 계층의 첫 번째 수준을 매핑하지 않습니다.

입력 포트를 메서드 입력의 요소에 연결하는 행을 표시하도록 선택할 수 있습니다.

## 입력 포트를 요소에 매핑하기 위한 규칙 및 지침

입력 포트를 메서드 입력 계층의 요소에 매핑하는 경우 다음 규칙을 검토합니다.

- 입력 포트는 계층의 한 요소에 매핑할 수 있습니다. 동일한 포트는 계층의 여러 키에 매핑할 수 있습니다.
- 입력 포트 및 요소의 데이터 유형은 호환되어야 합니다.
- 한 입력 그룹의 포트를 메서드 입력의 여러 계층 수준에 매핑할 수 있습니다.
- 입력 포트를 메서드 입력의 키에 매핑해야 합니다. 키에 매핑하는 포트는 문자열, 정수 또는 **bigint** 데이터 유형이어야 합니다. 요청 메시지에 포함하는 계층 수준 이상의 메서드 입력에 있는 모든 수준의 키에 데이터를 매핑하십시오. 매핑하는 수준을 포함하여 그 이상의 모든 수준에 대한 외래 키를 포함하십시오.

**참고:** 메서드 입력 계층의 가장 낮은 수준만 매핑할 경우 입력 포트를 키에 매핑할 필요가 없습니다.

- **RequestInput** 루트 요소는 메서드 입력 정의에 대한 **Rest\_Consumer\_input** 그룹의 하위 요소에 매핑해야 합니다.
- 여러 문자열, **bigint** 또는 정수 입력 포트를 **메서드 입력** 영역의 키에 매핑하여 복합 키를 작성할 수 있습니다. 복합 키의 **위치** 필드를 클릭하면 입력 포트의 순서를 다시 정렬하거나 포트 중 하나를 제거할 수 있습니다.
- 웹 서비스가 **JSON** 문서를 생성할 경우 응답 계층의 첫 번째 노드는 **xmlRoot**여야 합니다. **JSON** 응답이 포함된 웹 서비스의 첫 번째 노드가 **xmlRoot**가 아닐 경우 **null** 값이 나타날 수 있습니다.

## 메서드 입력에 입력 포트 매핑

변환 입력 매핑을 표시할 때 입력 그룹을 정의하고 입력 포트를 정의하고 입력 포트를 메서드 입력 요소에 매핑할 수 있습니다.

1. REST 웹 서비스 소비자 변환을 엽니다.
2. **포트** 보기에서 입력 매핑을 표시합니다.
3. 루트 입력 그룹에 대한 기본 키를 정의합니다.
4. 입력 그룹 또는 포트를 **포트** 영역에 추가하려면 다음 방법 중 하나를 사용합니다.

방법	설명
요소를 끌어서 놓습니다.	<b>메서드 입력</b> 영역에서 그룹 또는 하위 요소를 <b>포트</b> 영역의 빈 열로 끌어서 놓습니다. 그룹을 <b>포트</b> 영역으로 끌 경우 <b>Developer tool</b> 이 포트 없이 그룹을 추가합니다.
그룹 또는 포트를 수동으로 추가합니다.	그룹을 추가하려면 <b>새로 만들기</b> 단추 옆의 화살표를 클릭한 다음 <b>그룹</b> 을 클릭합니다. 포트를 추가하려면 <b>새로 만들기</b> 단추 옆의 화살표를 클릭한 다음 <b>필드</b> 를 클릭합니다.

방법	설명
다른 변환에서 포트를 끌어서 놓습니다.	편집기에서 다른 변환의 포트를 REST 웹 서비스 소비자 변환으로 끌어서 놓습니다.
포트를 복사합니다.	다른 변환에서 포트를 선택하여 <b>포트</b> 영역에 복사합니다. 포트를 복사하기 위해 키보드 바로 가기를 사용하거나 Developer tool의 <b>복사</b> 및 <b>붙여넣기</b> 단추를 사용할 수 있습니다.
계층의 첫 번째 수준 매핑을 선택합니다.	Developer tool이 메서드 입력의 첫 번째 수준에 있는 요소를 입력 포트 및 그룹에 매핑합니다. Developer tool은 매핑을 수행할 입력 포트 및 그룹도 작성합니다.

- 수동으로 포트를 작성하거나 다른 변환의 포트를 복사하는 경우 **메서드 입력** 영역에서 **위치** 열을 클릭하고 목록에서 포트를 선택합니다.
- 입력 포트를 복합 키로 매핑하려면 다음 방법 중 하나를 사용하십시오.

방법	설명
입력 포트를 끌어서 놓습니다.	두 개 이상의 입력 포트를 선택하고 메서드 입력 계층의 키로 끌어서 놓습니다.
<b>위치 선택</b> 대화 상자에서 입력 포트를 선택합니다.	메서드 입력 계층에 있는 키의 <b>위치</b> 열을 클릭한 다음 입력 포트를 선택합니다.

- 요소의 위치를 지우려면 다음 방법 중 하나를 사용합니다.

방법	설명
<b>지우기</b> 를 클릭합니다.	<b>메서드 입력</b> 영역에서 하나 이상의 요소를 선택하고 <b>지우기</b> 를 클릭합니다.
포트를 요소에 연결하는 행을 삭제합니다.	입력 포트를 메서드 입력의 요소에 연결하는 하나 이상의 행을 선택하고 <b>삭제</b> 를 누릅니다.

## REST 웹 서비스 소비자 변환 출력 매핑

변환 포트를 볼 때 메서드 출력 계층을 보려면 출력 매핑을 표시합니다. 출력 매핑을 표시하면 출력 그룹을 정의하고 출력 포트를 정의하고 메서드 출력 요소를 출력 포트에 매핑할 수 있습니다.

출력 매핑에는 다음 영역이 포함됩니다.

### 메서드 출력

**메서드 출력** 영역에는 웹 서비스가 REST 웹 서비스 소비자 변환에 반환하는 응답 메시지의 요소가 표시됩니다. 스키마 개체를 사용하여 변환을 작성하는 경우 스키마 개체가 메서드 출력 계층을 정의합니다.

### 포트

**포트** 영역에서 변환 출력 그룹 및 포트를 작성합니다.

출력 포트를 작성한 후 **메서드 출력**의 요소를 **포트** 영역의 포트에 매핑합니다. 메서드 출력의 요소를 출력 포트에 매핑하면 요소의 위치가 **포트** 영역의 **위치** 열에 나타납니다.

출력 계층의 첫 번째 수준을 매핑하기로 선택한 경우 Developer tool이 메서드 출력의 첫 번째 수준에 있는 요소를 출력 포트에 매핑합니다. Developer tool은 매핑을 수행할 포트도 작성합니다. 하나 이상의 다중 발생 하위

요소가 있는 다중 발생 상위 요소가 계층의 첫 번째 수준에 포함될 경우 **Developer tool**이 계층의 첫 번째 수준을 매핑하지 않습니다.

계층에 출력 포트를 표시하도록 선택할 수 있습니다. 각 하위 그룹은 상위 그룹 아래에 나타납니다. 메서드 출력의 요소를 출력 포트에 연결하는 행을 표시하도록 선택할 수도 있습니다.

연결된 스키마 개체가 리포지토리에서 삭제된 경우 출력 매핑의 메서드 요소에 대한 위치는 **Developer tool**에 유지됩니다. 출력 매핑을 표시할 때 메서드 요소의 위치는 여전히 **포트** 영역에서 출력 포트에 대한 **위치** 열에 표시됩니다. 다른 스키마를 변환에 연결할 경우 **Developer tool**이 각 위치가 올바른지 확인합니다. 위치가 더 이상 올바르지 않은 경우 **Developer tool**이 출력 매핑의 **포트** 영역에서 메서드 요소의 위치를 지웁니다.

## 요소를 출력 포트에 매핑하기 위한 규칙 및 지침

메서드 출력 계층의 요소를 출력 포트에 매핑하는 경우 다음 규칙을 검토합니다.

- 메서드 출력 요소 및 출력 포트의 데이터 유형은 호환되어야 합니다.
- 그룹의 출력 포트 2개 이상에 요소를 매핑할 수 없습니다.
- 통과 포트를 제외한 모든 출력 포트에는 올바른 위치가 있어야 합니다.
- 여러 번 발생하는 하위 요소를 빈 출력 포트에 끝 경우 그룹을 다른 출력 그룹에 연결해야 합니다. 그룹을 선택하면 **Developer tool**이 그룹을 연결하는 키를 작성합니다.
- 여러 번 발생하는 요소를 상위 요소가 포함된 그룹으로 끝면 포함할 하위 요소 발생 수를 구성할 수 있습니다. 또는 상위 그룹을 변환 출력에서 여러 번 발생하는 하위 그룹으로 대체할 수 있습니다.
- 웹 서비스가 **JSON** 문서를 생성할 경우 응답 계층의 첫 번째 노드는 **xmlRoot**여야 합니다. **JSON** 응답이 포함된 웹 서비스의 첫 번째 노드가 **xmlRoot**가 아닐 경우 출력 포트에 **null** 값이 나타날 수 있습니다.

## 보기 사용자 지정의 옵션

**메서드 출력** 영역에 쿠키 포트, 통과 포트 및 키가 표시되도록 메서드 출력 계층을 변경할 수 있습니다. 요소 정렬 방법을 정의하는 그룹화 구성을 표시할 수도 있습니다.

**메서드 출력** 영역에서 **보기 사용자 지정**을 클릭합니다. 다음 옵션을 활성화합니다.

### 시퀀스, 선택 및 모두

요소 정의가 시퀀스, 선택 또는 모두인지 여부를 나타내는 행을 표시합니다.

시퀀스 그룹의 요소는 계층에 지정된 순서여야 합니다.

선택 그룹의 요소 하나 이상이 응답 메시지에 표시되어야 합니다.

모든 그룹의 요소가 응답 메시지에 포함되어야 합니다.

### 키

**메서드 출력** 영역에서 키를 봅니다. **메서드 출력** 영역에는 각 그룹의 키가 포함됩니다. **포트** 영역의 출력 포트에 키를 추가할 수 있습니다.

### 통과 포트

**메서드 출력** 영역에 통과 포트가 표시됩니다. 통과 포트는 데이터를 변경하지 않고 변환을 통해 데이터를 전달하는 포트입니다. 메서드 출력의 통과 포트를 모든 **REST** 웹 서비스 소비자 변환 출력 그룹에 연결할 수 있습니다. 통과 포트는 데이터를 한 번 수신하므로 응답 메시지의 루트 수준에 위치합니다.



## 출력 포트에 메서드 출력 매핑

변환 출력 매핑을 표시할 때 출력 그룹을 정의하고 출력 포트를 정의하고 메서드 출력 요소를 출력 포트에 매핑할 수 있습니다.

1. REST 웹 서비스 소비자 변환을 엽니다.
2. **포트** 보기에서 출력 매핑을 표시합니다.
3. 출력 그룹 또는 포트를 **포트** 영역에 추가하려면 다음 방법 중 하나를 사용합니다.

방법	설명
요소를 끌어서 놓습니다.	<b>메서드 출력</b> 영역에서 그룹 또는 하위 요소를 <b>포트</b> 영역의 빈 열로 끌어서 놓습니다. 그룹을 <b>포트</b> 영역으로 끌 경우 Developer tool이 포트 없이 그룹을 추가합니다.
그룹 또는 포트를 수동으로 추가합니다.	그룹을 추가하려면 <b>새로 만들기</b> 단추 옆의 화살표를 클릭한 다음 <b>그룹</b> 을 클릭합니다. 포트를 추가하려면 <b>새로 만들기</b> 단추 옆의 화살표를 클릭한 다음 <b>필드</b> 를 클릭합니다.
다른 변환에서 포트를 끌어서 놓습니다.	편집기에서 다른 변환의 포트를 REST 웹 서비스 소비자 변환으로 끌어서 놓습니다.
포트를 복사합니다.	다른 변환에서 포트를 선택하여 <b>포트</b> 영역에 복사합니다. 포트를 복사하기 위해 키보드 바로 가기를 사용하거나 Developer tool의 <b>복사</b> 및 <b>붙여넣기</b> 단추를 사용할 수 있습니다.

4. 포트를 수동으로 작성하거나 다른 변환의 포트를 복사하는 경우 **포트** 영역에서 **위치** 열을 클릭하고 목록에서 요소를 선택합니다.
5. 포트의 위치를 지우려면 다음 방법 중 하나를 사용합니다.

방법	설명
<b>지우기</b> 를 클릭합니다.	<b>포트</b> 영역에서 하나 이상의 포트를 선택하고 <b>지우기</b> 를 클릭합니다.
요소를 포트에 연결하는 행을 삭제합니다.	메서드 출력의 요소를 출력 포트에 연결하는 하나 이상의 행을 선택하고 <b>삭제</b> 를 누릅니다.

## REST 웹 서비스 소비자 변환의 고급 속성

데이터 통합 서비스가 REST 웹 서비스 소비자 변환의 데이터를 처리하는 방법을 결정하는 속성을 구성할 수 있습니다.

**고급** 탭에서 다음 속성을 구성합니다.

### 추적 수준

이 변환에 대해 로그에 표시되는 세부 정보의 양입니다. 간단, 보통, 자세한 정보 표시 초기화 또는 자세한 정보 표시 데이터를 선택할 수 있습니다. 기본값은 보통입니다.

### 연결

웹 서비스에 연결할 HTTP 연결 개체를 식별합니다. HTTP 연결은 Developer tool에서 작성하고 편집합니다. HTTP 연결을 구성할 때 기본 URL, 웹 서비스에 필요한 보안 유형 및 연결 제한 시간 기간을 구성합니다.

REST 웹 서비스 소비자 변환은 URL을 사용하여 웹 서비스에 연결합니다. 이 URL은 변환 속성 또는 HTTP 연결에서 정의할 수 있습니다.

HTTP 연결은 다음과 같은 상황에서 구성합니다.

- URL 입력 포트를 사용하지 않습니다.
- 웹 서비스에 HTTP 인증 또는 SSL 인증서가 필요합니다.
- 기본 연결 제한 시간 기간을 변경해야 합니다.

#### XML 스키마 유효성 검사

런타임 시 응답 메시지의 유효성을 검사합니다. **올바르지 않은 XML에 대한 오류** 또는 **유효성 검사 없음**을 선택합니다.

#### 정렬된 입력

데이터 통합 서비스에서 모든 입력 데이터를 처리하지 않고도 출력을 생성할 수 있습니다. 입력 데이터를 XML 입력 계층의 키별로 정렬한 경우 정렬된 입력을 활성화합니다.

#### URL

REST 웹 서비스의 기본 URL입니다. 이 값은 HTTP 연결의 기본 URL로 재정의됩니다.

#### 형식

웹 서비스 응답의 형식입니다. 웹 서비스 응답에 따라 **XML** 또는 **JSON**을 선택합니다.

## REST 웹 서비스 소비자 변환 작성

재사용 가능하거나 재사용 불가능한 REST 웹 서비스 소비자 변환을 작성할 수 있습니다. 재사용 가능 변환은 여러 매핑에 존재할 수 있습니다. 재사용 불가능 변환은 단일 매핑에만 존재합니다.

REST 웹 서비스 소비자 변환을 작성할 때 요소 및 XML 계층을 수동으로 정의하거나 스키마 개체에서 요소 및 계층을 가져올 수 있습니다. 스키마 개체는 XML 파일 또는 텍스트 파일이 될 수 있습니다.

### REST 웹 서비스 소비자 변환 작성

REST 웹 서비스 소비자 변환을 작성할 때는 메서드를 선택하고 선택한 메서드에 따라 메서드 입력 및 메서드 출력을 정의합니다.

1. REST 웹 서비스 소비자 변환을 작성하려면 다음 방법 중 하나를 사용합니다.

방법	설명
재사용 가능	개체 탐색기 보기에서 프로젝트나 폴더를 선택합니다. <b>파일 &gt; 새로 만들기 &gt; 변환</b> 을 클릭합니다. REST 웹 서비스 소비자 변환을 선택하고 <b>다음</b> 을 클릭합니다.
재사용 불가능	매핑 또는 맵렛에서 변환 색상표의 REST 웹 서비스 소비자 변환을 매핑 또는 맵렛 편집기로 끌어서 놓습니다.

2. 변환 이름을 입력하고 위치 및 HTTP 메서드를 선택합니다.
3. **다음**을 클릭합니다.

- 메서드 입력을 정의하려면 다음 방법 중 하나를 사용합니다.

방법	설명
빈 항목으로 생성	XML 요소 및 계층을 수동으로 정의합니다.
스키마 개체의 요소에서 작성	스키마 개체에서 XML 요소 및 계층을 가져옵니다.

**메서드 입력 정의** 영역에 변환 입력 그룹 및 입력 포트가 표시됩니다. **입력 매핑** 영역에 요청 메시지 계층이 표시됩니다.

- 입력 그룹 및 입력 포트를 정의하고 입력 포트를 입력 요소로 매핑합니다.
- 다음을 클릭합니다.
- 메서드 출력을 정의하려면 **빈 항목으로 작성** 또는 **스키마 개체의 요소에서 작성**을 선택합니다.  
**메서드 출력 정의** 영역에 변환 입력 그룹 및 입력 포트가 표시됩니다. **출력 매핑** 영역에 요청 메시지 계층이 표시됩니다.
- 출력 그룹 및 출력 포트를 정의하고 요소를 출력 포트에 매핑합니다.
- 마침을 클릭합니다.

## 배열이 포함된 JSON 응답 메시지 구문 분석

요소가 복합 유형의 하위 요소이고 이 요소의 최대 발생 수가 바인딩되지 않음인 경우 스키마가 올바르지 않습니다. JSON 파서는 요소의 여러 인스턴스를 추출하는 것을 제한합니다.

복합 유형에서 하위 요소의 최대 발생 수는 0 또는 1이어야 하고 스키마의 복합 유형에 대해 순서 표시기를 선택해야 합니다. 스키마 유효성 검사를 위해 최대 발생 수를 1로 변경하는 경우 한 번에 하나의 요소 인스턴스를 추출할 수 있습니다.

스키마의 복합 유형 선택 순서 표시기에서 최대 발생 수를 바인딩되지 않음으로 사용할 수 있습니다.

### JSON 응답 메시지 예제

복합 유형 요소 `xmlRoot`의 요소 이름이 `Likes`이고 최대 발생 수가 바인딩되지 않은 다음과 같은 스키마가 있습니다.

```
<xs:schema attributeFormDefault="unqualified" elementFormDefault="qualified" xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="xmlRoot">
    <xs:complexType>
      <xs:all>
        <xs:element type="xs:byte" name="Age"/>
        <xs:element type="xs:string" name="FirstName"/>
        <xs:element type="xs:string" name="Likes" maxOccurs="unbounded" minOccurs="0"/>
        <xs:element type="xs:string" name="FamilyName"/>
      </xs:all>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

이 경우 다음과 같은 형식으로 JSON 응답을 변경할 수 있습니다.

```
<xs:schema attributeFormDefault="unqualified" elementFormDefault="qualified" xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="xmlRoot">
    <xs:complexType>
```

```

<xs:choice maxOccurs="unbounded">
  <xs:element type="xs:byte" name="Age"/>
  <xs:element type="xs:string" name="FirstName"/>
  <xs:element type="xs:string" name="Likes" />
  <xs:element type="xs:string" name="FamilyName"/>
</xs:choice>
</xs:complexType>
</xs:element>
</xs:schema>

```

<xs:choice maxOccurs="unbounded">는 콘텐츠를 한 번 이상 원하는 순서로 반복할 수 있습니다.

## 응답 메시지의 명명되지 않은 배열

REST 웹 서비스 소비자 변환은 명명되지 않은 배열을 요청 메시지가 아닌 응답 메시지에서만 지원합니다. 메서드 출력 정의에 정의된 명명되지 않은 배열 스키마를 구문 분석하려면 **complexType**의 상위 요소 또는 단순 유형 배열 요소 이름이 **xmlRoot**여야 합니다.

REST 웹 서비스 소비자 변환에서는 **xmlRoot**를 최대 발생이 바운드되지 않음으로 설정된 **xmlRoot** 요소의 하위 요소로 정의하고 명명되지 않은 배열의 요소를 **xmlRoot** 요소의 하위 요소로 정의해야 합니다.

다음 이미지는 명명되지 않은 배열에 대해 정의된 메서드 출력을 보여 줍니다.

☐ Ports ☐ Method input ☒ Method output

Show: ☒ Method output definition ☐ Output mapping

Method output definition

Name	Type	Min...	Ma...	Description	>>
Rest_Consume...	(Rest_Cons...				
xmlRoot	(xmlRoot)	1	1		
xmlRoot	(xmlRoot)	1	Un...		
emp	xs:string	1	1		
empid	xs:string	1	1		

## 제 39 장

# 라우터 변환

이 장에 포함된 항목:

- [라우터 변환 개요, 553](#)
- [동적 매핑의 라우터 변환, 554](#)
- [그룹 작업, 554](#)
- [포트 작업, 558](#)
- [매핑에서 라우터 변환 연결, 559](#)
- [라우터 변환 고급 속성, 559](#)
- [라우터 변환 - 비원시 환경, 559](#)

## 라우터 변환 개요

라우터 변환은 하나 이상의 조건에 따라 데이터를 여러 출력 그룹으로 라우팅하는 활성 변환입니다. 출력 그룹을 서로 다른 여러 변환 또는 매핑의 여러 대상으로 라우팅합니다.

라우터 변환과 필터 변환은 모두 조건을 사용하여 데이터를 테스트하므로 서로 비슷합니다. 필터 변환은 조건 하나로 데이터를 테스트하여 조건을 충족하지 않는 데이터 행은 삭제합니다. 라우터 변환은 하나 이상의 조건으로 데이터를 테스트하며 조건을 하나라도 충족하지 않는 데이터 행을 기본 출력 그룹으로 라우팅할 수 있습니다.

여러 조건에 따라 같은 입력 데이터를 테스트해야 하는 경우에는 같은 태스크를 수행하기 위해 여러 필터 변환을 작성하는 대신 매핑에서 라우터 변환을 사용하십시오. 라우터 변환이 더 효율적입니다. 예를 들어 3개의 조건에 따라 데이터를 테스트하려는 경우 필터 변환 3개 대신 라우터 변환 하나를 사용할 수 있습니다. 매핑에서 라우터 변환을 사용할 때 데이터 통합 서비스는 들어오는 데이터를 한 번 처리합니다. 매핑에서 여러 필터 변환을 사용할 때 데이터 통합 서비스는 각 변환에 대해 들어오는 데이터를 처리합니다.

라우터 변환은 입력/출력 그룹, 입력/출력 포트, 그룹 필터 조건 및 **Developer tool**에서 구성하는 고급 속성으로 구성됩니다.

다음 그림에는 샘플 라우터 변환과 해당 구성 요소가 나와 있습니다.

Name	Type	Length
<b>Input (4)</b>		
COUNTRY	string	13
CUSTOMER_NO	decimal	10
FIRSTNAME	string	10
LASTNAME	string	10
<b>Default (4)</b>		
COUNTRY	string	13
CUSTOMER_NO	decimal	10
FIRSTNAME	string	10
LASTNAME	string	10
<b>France (4)</b>		
COUNTRY	string	13
CUSTOMER_NO	decimal	10
FIRSTNAME	string	10
LASTNAME	string	10
<b>Japan (4)</b>		
COUNTRY	string	13
CUSTOMER_NO	decimal	10
FIRSTNAME	string	10
LASTNAME	string	10
<b>USA (4)</b>		
COUNTRY	string	13
CUSTOMER_NO	decimal	10
FIRSTNAME	string	10
LASTNAME	string	10

1. 입력 그룹
2. 입력 포트
3. 기본 출력 그룹
4. 사용자 정의 출력 그룹

## 동적 매핑의 라우터 변환

동적 매핑에서 라우터 변환을 사용할 수 있습니다. 변환에서 동적 포트를 구성하고 그룹 필터 조건의 생성된 포트를 참조할 수 있습니다.

그룹 필터 조건에서 동적 포트를 사용하는 경우 동적 포트에는 한 개 이상의 생성된 포트가 포함될 수 있습니다. 그룹 필터 조건이 각 생성된 포트를 포함하도록 확장됩니다. 각 생성된 포트는 식에 대해 올바른 유형이어야 합니다.

그룹 필터 조건을 매개 변수화할 수 있습니다. 식 유형 매개 변수를 사용하여 필터를 지정합니다.

## 그룹 작업

라우터 변환에는 다음과 같은 유형의 그룹이 있습니다.

- 입력
- 출력

## 입력 그룹

순위 변환에는 단일의 입력 그룹이 포함됩니다. 입력 그룹에는 사용자가 변환에 추가하는 모든 입력 포트가 포함됩니다.

## 출력 그룹

순위 변환에는 다음과 같은 유형의 출력 그룹이 포함됩니다.

### 사용자 정의 그룹

들어오는 데이터를 기준으로 조건을 테스트하려면 사용자 정의 그룹을 생성합니다. 사용자 정의 그룹은 출력 포트와 그룹 필터 조건으로 구성됩니다. **Developer tool**을 사용하여 **그룹** 탭에서 사용자 정의 그룹을 생성하고 편집할 수 있습니다. 지정하려는 각 조건에 대해 사용자 정의 그룹을 하나씩 생성합니다.

데이터 통합 서비스는 조건을 사용하여 들어오는 데이터의 각 행을 평가합니다. 그리고 기본 그룹을 처리하기 전에 각 사용자 정의 그룹의 조건을 테스트합니다. 데이터 통합 서비스는 연결된 출력 그룹의 순서에 따라 각 조건의 평가 순서를 결정합니다. 데이터 통합 서비스는 매핑의 대상이나 변환에 연결된 사용자 정의 그룹을 처리합니다.

행이 둘 이상의 그룹 필터 조건을 충족하면 데이터 통합 서비스는 해당 행을 여러 번 전달합니다.

### 기본 그룹

사용자 정의 그룹을 하나 생성하고 나면 **Developer tool**에서 기본 그룹을 생성합니다. **Developer tool**에서 기본 그룹을 편집하거나 삭제할 수는 없습니다. 이 그룹에는 그룹 필터 조건이 연결되어 있지 않습니다. 모든 그룹 조건이 **FALSE**로 평가되면 데이터 통합 서비스는 행을 기본 그룹으로 전달합니다. 데이터 통합 서비스가 기본 그룹의 모든 행을 삭제하도록 하려면 매핑에서 변환 또는 대상에 기본 그룹을 연결하지 마십시오.

목록에서 마지막 사용자 정의 그룹을 삭제하면 **Developer tool**에서 기본 그룹을 삭제합니다.

**Developer tool**은 입력 그룹의 입력 포트에서 속성 정보를 복사하여 각 출력 그룹에 대해 출력 포트 집합을 생성합니다. 출력 포트 또는 해당 속성은 변경하거나 삭제할 수 없습니다.

## 그룹 필터 조건 사용

하나 이상의 그룹 필터 조건에 따라 데이터를 테스트할 수 있습니다. 식 편집기를 사용하여 **그룹** 탭에서 그룹 필터 조건을 작성합니다.

단일 값을 반환하는 식을 입력할 수 있습니다. 조건에 상수를 지정할 수도 있습니다. 그룹 필터 조건은 행이 지정된 조건을 충족하는지 여부에 따라, 변환을 통과하는 각 행에 대해 **TRUE** 또는 **FALSE**를 반환합니다. 영(0)은 **FALSE**와 같습니다. 영(0)이 아닌 값은 **TRUE**에 해당합니다. 단일 숫자 포트를 필터 조건으로 사용할 수 있습니다. 데이터 통합 서비스가 **TRUE**로 평가되는 데이터 행을 각 사용자 정의 그룹과 연결된 각 변환 또는 대상에 전달합니다.

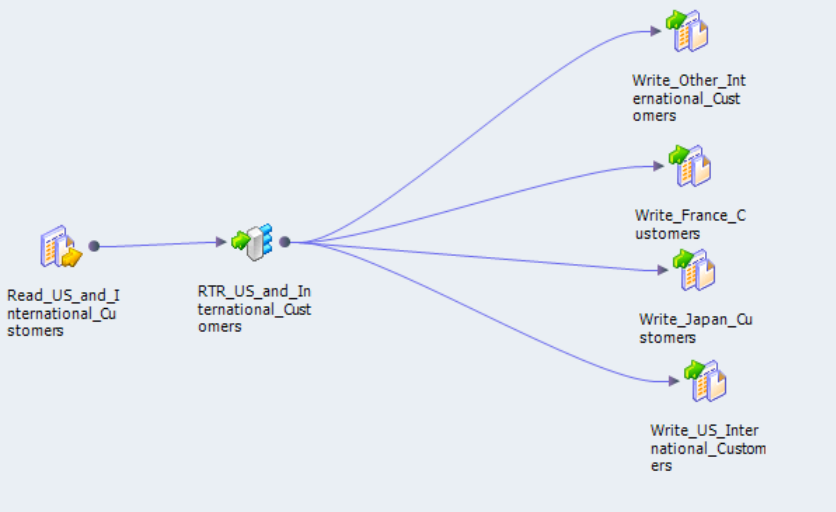
**참고:** 단일 동적 포트를 사용하여 부울 값을 반환할 수 없습니다.

예를 들어 9개 국가에 고객이 있는 상태에서 3개 국가의 데이터에 서로 다른 계산을 수행하려고 합니다. 매핑에서 라우터 변환을 사용하면 이 데이터를 3개의 다른 식 변환으로 필터링할 수 있습니다.

그룹 필터 조건에서 매개 변수를 요소로 사용할 수 있습니다. 시스템 매개 변수 또는 사용자 정의 매개 변수를 사용할 수 있습니다. 식 편집기에서 매개 변수를 작성하고 식에 추가할 수 있습니다.

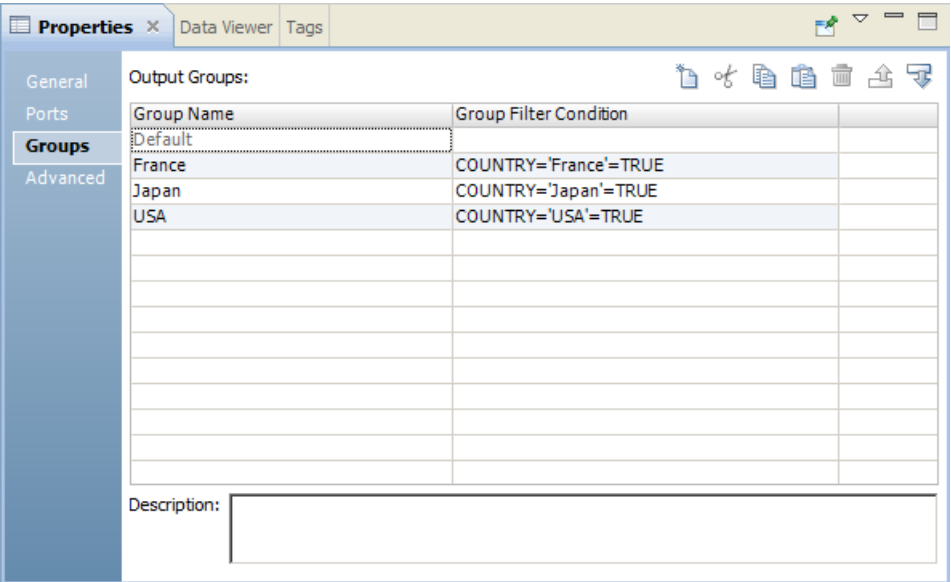
기본 그룹에는 그룹 필터 조건이 없습니다. 하지만 식 변환을 작성하면 나머지 6개 국가의 데이터를 기반으로 계산을 수행할 수 있습니다.

다음 그림은 여러 조건에 따라 데이터를 필터링하는 라우터 변환을 사용하는 매핑을 보여줍니다.



3개의 다른 국가의 데이터에 따라 여러 계산을 수행하기 위해 그룹 탭에서 3개의 사용자 정의 그룹을 작성하고 3개의 그룹 필터 조건을 지정합니다.

다음 그림은 고객 데이터를 필터링하는 그룹 필터 조건을 보여줍니다.



다음 테이블은 고객 데이터를 필터링하는 그룹 필터 조건을 보여줍니다.

그룹 이름	그룹 필터 조건
프랑스	customer_name='프랑스'=TRUE
일본	customer_name='일본'=TRUE
USA	customer_name='USA'=TRUE

매핑에서 데이터 통합 서비스가 TRUE로 평가되는 데이터 행을 일본, 프랑스 및 미국과 같은 각 사용자 정의 그룹과 연결된 각 변환 또는 대상에 전달합니다. 모든 조건이 FALSE로 평가되면 데이터 통합 서비스가 기본 그룹으로 행을 전달합니다. 그런 다음 데이터 통합 서비스가 다른 6개 국가의 데이터를 기본 그룹과 연결된 대상 또는



변환으로 전달합니다. 데이터 통합 서비스가 기본 그룹의 모든 행을 삭제하도록 하려면 매핑에서 변환 또는 대상에 기본 그룹을 연결하지 마십시오.

라우터 변환이 조건을 충족하는 각 그룹을 통해 데이터를 전달합니다. 데이터가 3개의 출력 그룹 조건을 충족하는 경우 라우터 변환이 3개의 출력 그룹을 통해 데이터를 전달합니다.

예를 들어 라우터 변환에서 다음 그룹 조건을 구성합니다.

그룹 이름	그룹 필터 조건
출력 그룹 1	employee_salary > 1000
출력 그룹 2	employee_salary > 2000

라우터 변환이 **employee\_salary=3000**을 사용하여 입력 행 데이터를 처리하면 출력 그룹 1 및 2를 통해 데이터를 라우팅합니다.

## 그룹 필터 조건의 동적 포트

그룹 필터 조건에서 동적 포트를 사용할 수 있습니다. 데이터 통합 서비스가 동적 포트의 각 생성된 포트에 필터 조건을 적용합니다.

예를 들어 동적 포트 **MyDynamicPort**에는 10진수 포트 3개가 포함됩니다.

Salary  
Bonus  
Stock

다음 그룹 필터 조건을 구성할 수 있습니다.

**MyDynamicPort > 100**

그룹 필터 조건이 다음 식으로 확장됩니다.

**Salary > 100 AND Bonus > 100 AND Stock > 100**

각 생성된 포트는 식에 대해 올바른 유형이어야 합니다.

## 그룹 필터 매개 변수화

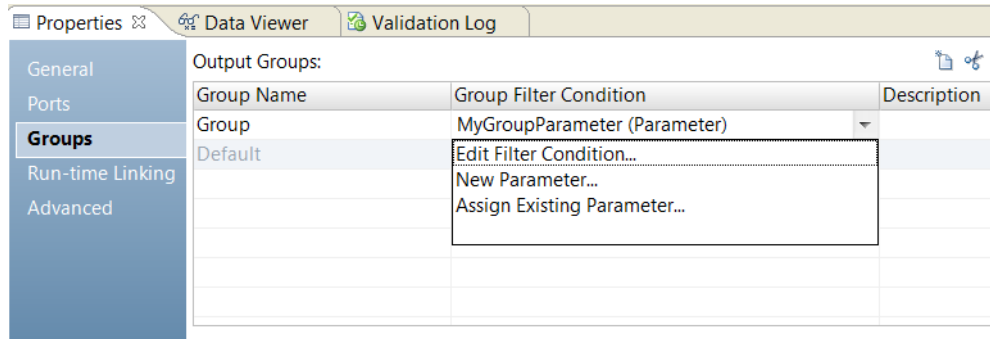
식 매개 변수를 사용하여 그룹 필터를 정의할 수 있습니다. 식 매개 변수는 전체 식이 포함된 매개 변수입니다.

예를 들어 식 매개 변수에는 다음 식이 포함될 수 있습니다.

**Employee\_Salary > 1000.**

그룹 필터를 매개 변수화하려면 그룹 탭의 **그룹 필터 조건** 필드에서 **새 매개 변수**를 선택합니다. 식 편집기에서 매개 변수를 정의하고 식을 입력합니다. 식 매개 변수에는 다른 매개 변수가 포함될 수 없습니다.

다음 이미지는 변환 속성의 **그룹** 탭을 보여 줍니다.



식 매개 변수를 사용하는 경우 **Developer tool**이 식의 유효성을 검사할 수 없습니다. 런타임 시 식이 올바르지 않은 경우 매핑이 실패할 수 있습니다.

## 그룹 추가

그룹을 추가하면 **Developer tool**이 입력 포트의 속성 정보를 출력 포트에 복사합니다.

1. **그룹** 탭을 클릭합니다.
2. **새로 만들기** 단추를 클릭합니다.
3. **그룹 이름** 섹션에 그룹 이름을 입력합니다.
4. **그룹 필터 조건** 필드를 클릭하여 **식 편집기**를 엽니다.
5. 그룹 필터 조건을 입력합니다.
6. **유효성 검사**를 클릭하여 조건의 구문을 검사합니다.
7. **확인**을 클릭합니다.

## 포트 작업

라우터 변환에는 입력 포트 및 출력 포트가 있습니다. 입력 포트는 입력 그룹에 있고 출력 포트는 출력 그룹에 있습니다.

**포트** 탭에서 수동으로 작성하거나 다른 변환에서 복사하여 입력 포트를 작성할 수 있습니다.

**Developer tool**은 입력 포트에서 다음 속성을 복사하여 출력 포트를 작성합니다.

- 포트 이름
- 데이터 유형
- 전체 자릿수
- 배율
- 기본값

입력 포트를 변경하는 경우 **Developer tool**이 출력 포트를 업데이트하여 이러한 변경 내용을 반영합니다. 출력 포트는 편집하거나 삭제할 수 없습니다.

**Developer tool**은 입력 포트 이름에 따라 출력 포트 이름을 작성합니다. 각 입력 포트에 대해 **Developer tool**은 각 출력 그룹에 해당 출력 포트를 작성합니다.

## 매핑에서 라우터 변환 연결

매핑에서 변환을 라우터 변환에 연결할 경우 다음 규칙을 고려하십시오.

- 하나의 그룹을 하나의 변환 또는 대상에 연결할 수 있습니다.
- 그룹의 출력 포트 1개를 여러 변환 또는 대상에 연결할 수 있습니다.
- 1개 그룹의 여러 출력 포트를 여러 변환 또는 대상에 연결할 수 있습니다.
- 하나 이상의 그룹을 대상 1개 또는 단일 입력 그룹 변환에 연결할 수 없습니다.
- 각 출력 그룹을 다른 입력 그룹에 연결할 때 조이너 변환을 제외한 여러 입력 그룹 변환에 하나 이상의 그룹을 연결할 수 없습니다.

## 라우터 변환 고급 속성

데이터 통합 서비스가 라우터 변환에 대해 데이터를 처리하는 방법을 결정할 수 있는 고급 속성을 구성합니다.

로그의 추적 수준을 구성할 수 있습니다.

고급 탭에서 다음 속성을 구성합니다.

### 추적 수준

이 변환에 대해 로그에 표시되는 세부 정보의 양입니다. 간단, 보통, 자세한 정보 표시 초기화 또는 자세한 정보 표시 데이터를 선택할 수 있습니다. 기본값은 보통입니다.

## 라우터 변환 - 비원시 환경

비원시 환경에서 라우터 변환의 처리는 변환을 실행하는 엔진에 따라 다릅니다.

다음과 같은 비원시 런타임 엔진에 대한 지원을 고려하십시오.

- Blaze 엔진. 제한 없이 지원됩니다.
- Spark 엔진. 제한 없이 지원됩니다.
- Databricks Spark 엔진. 제한 없이 지원됩니다.

## 제 40 장

# 시퀀스 생성기 변환

이 장에 포함된 항목:

- [시퀀스 생성기 변환 개요, 560](#)
- [시퀀스 생성기 포트, 561](#)
- [시퀀스 생성기 변환 속성, 566](#)
- [시퀀스 데이터 개체, 571](#)
- [시퀀스 생성기 변환 작성, 573](#)
- [시퀀스 생성기 변환 작성, 574](#)
- [FAQ, 575](#)
- [시퀀스 생성기 변환 - 비원시 환경, 576](#)

## 시퀀스 생성기 변환 개요

시퀀스 생성기 변환은 숫자 값을 생성하는 수동 변환입니다. 시퀀스 생성기 변환을 사용하여 고유한 기본 키 값을 생성하거나 누락된 기본 키를 바꾸거나 순차적 숫자 범위를 반복합니다.

시퀀스 생성기 변환은 연결된 변환입니다. 이 변환은 하나 이상의 변환에 연결할 수 있는 두 출력 포트를 포함합니다. 통합 서비스는 행 블록이 연결된 변환에 입력될 때마다 시퀀스 번호 블록을 생성합니다. **CURRVAL**을 연결하면 통합 서비스가 각 블록에서 행 하나를 처리합니다. **NEXTVAL**을 다른 변환의 입력 포트에 연결하면 통합 서비스는 숫자 시퀀스를 생성합니다. **CURRVAL**을 다른 변환의 입력 포트에 연결하면 통합 서비스는 **NEXTVAL** 값과 증분 범위 값을 생성합니다.

시퀀스 생성기는 통과 포트와 출력 포트를 포함합니다. **NEXTVAL** 포트는 다른 변환의 입력 포트에 연결합니다. 통합 서비스는 매핑 실행 시 시퀀스를 증분합니다.

단일 매핑에서 사용할 시퀀스 생성기 변환을 생성할 수도 있고 여러 매핑에서 사용할 재사용 가능 시퀀스 생성기 변환을 생성할 수도 있습니다. 재사용 가능 시퀀스 생성기 변환은 시퀀스 생성기 변환 인스턴스를 사용하는 각 매핑에서 시퀀스의 무결성을 유지 관리합니다.

새 시퀀스 또는 시퀀스 데이터 개체를 바탕으로 시퀀스 생성기 변환을 생성할 수 있습니다. 시퀀스 데이터 개체는 값 시퀀스를 생성하고 유지 관리하는 개체입니다.

시퀀스 생성기 변환을 재사용 가능하도록 지정하여 여러 매핑에서 사용할 수 있습니다. 단일 대상에 대한 여러 로드를 수행할 때 시퀀스 생성기 변환을 재사용할 수 있습니다.

예를 들어 큰 입력 파일을 병렬로 실행되는 3개 세션으로 구분하는 경우 시퀀스 생성기 변환을 사용하여 기본 키 값을 생성합니다. 다른 시퀀스 생성기 변환을 사용하는 경우에는 통합 서비스가 중복 키 값을 생성할 수 있습니다. 대신 3개 세션에 대해 모두 재사용 가능 시퀀스 생성기 변환을 사용하여 각 대상 행에 고유한 값을 제공합니다.

# 시퀀스 생성기 포트

시퀀스 생성기 변환에는 NEXTVAL 및 CURRVAL의 두 가지 출력 포트가 있습니다. 이러한 포트는 편집하거나 삭제할 수 없습니다. 마찬가지로 포트를 변환에 추가할 수 없습니다.

시퀀스 생성기 변환에는 통과 포트와 하나의 출력 포트, NEXTVAL가 있습니다. 출력 포트는 편집하거나 삭제할 수 없습니다.

## 통과 포트

시퀀스 생성기 변환에 포트를 통과 포트로 추가할 수 있습니다. 통과 포트는 입력 데이터를 수신한 후 데이터 변경 없이 매핑에 동일한 데이터를 반환하는 입력 및 출력 포트입니다.

NEXTVAL 및 CURRVAL 출력 포트를 대상에 연결하기 전에 입력 포트를 하나 이상 변환에 추가한 다음 업스트림 소스 또는 변환에 연결해야 합니다. 변환에 통과 포트를 추가하려면 매핑의 업스트림 소스 또는 변환에서 시퀀스 생성기 변환으로 포트를 끕니다.

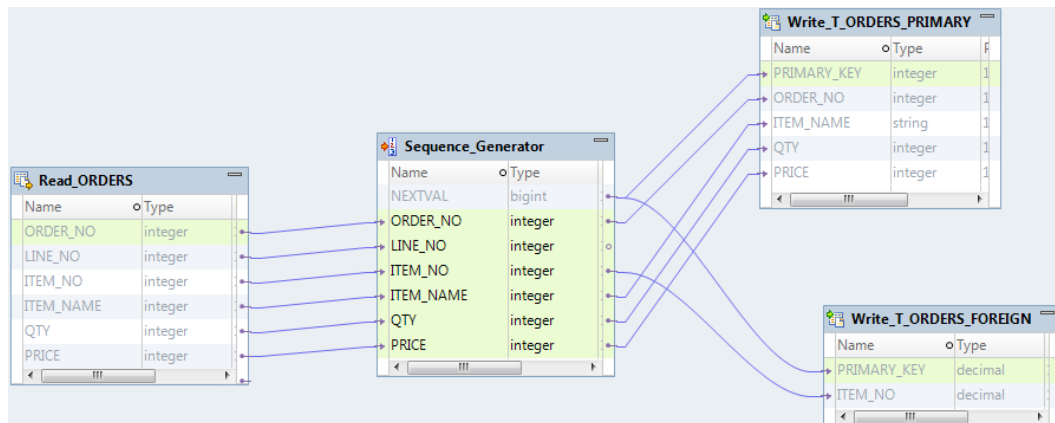
NEXTVAL 출력 포트를 대상에 연결하기 전에 입력 포트를 하나 이상 변환에 추가한 다음 업스트림 소스 또는 변환에 연결해야 합니다. 변환에 통과 포트를 추가하려면 매핑의 업스트림 소스 또는 변환에서 시퀀스 생성기 변환으로 포트를 끕니다.

## NEXTVAL 포트

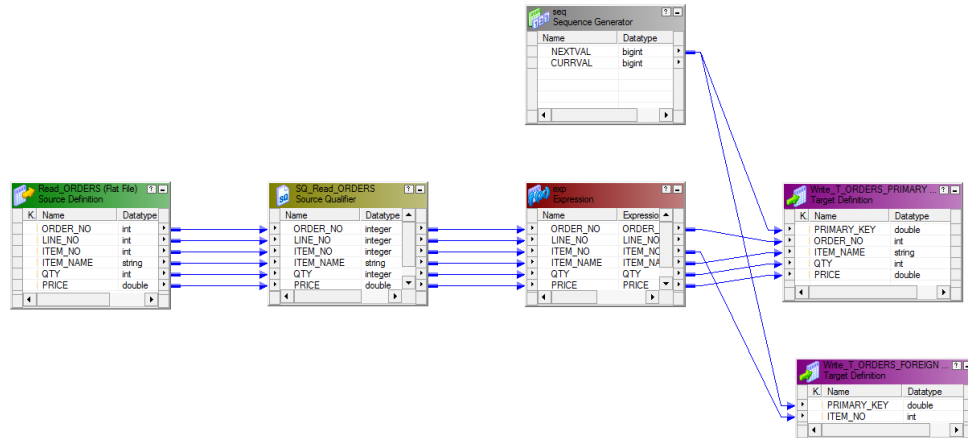
NEXTVAL을 변환에 연결하여 변환의 각 행에 대해 고유한 값을 생성할 수 있습니다. NEXTVAL 포트를 다운스트림 변환 또는 대상에 연결하여 숫자 시퀀스를 생성합니다. NEXTVAL을 여러 변환에 연결하는 경우 통합 서비스가 각 변환에 대해 동일한 숫자 시퀀스를 생성합니다.

NEXTVAL 포트를 연결하여 시작 값 및 증분값 속성에 따라 시퀀스를 생성합니다. 시퀀스 생성기가 시퀀스를 반복하도록 구성되지 않은 경우 NEXTVAL 포트가 구성된 끝 값까지 시퀀스 번호를 생성합니다.

다음 이미지는 기본 및 외래 키 값을 생성하기 위해 소스 및 두 대상에 연결된 시퀀스 생성기 변환 NEXTVAL 포트와의 매핑을 보여줍니다.



다음 이미지는 기본 및 외래 키 값을 생성하기 위해 두 대상에 연결된 시퀀스 생성기 변환 NEXTVAL 포트와의 매핑을 보여줍니다.



시작 값 = 1과 증분 값 = 1을 사용하여 시퀀스 생성기 변환을 구성하는 경우 통합 서비스가 T\_ORDERS\_PRIMARY 및 T\_ORDERS\_FOREIGN 대상 테이블에 대해 동일한 기본 키 값을 생성합니다.

NEXTVAL을 여러 변환에 연결하여 각 변환의 각 행에 대해 고유한 값을 생성합니다. NEXTVAL 포트를 다운스트림 변환 또는 대상에 연결하여 시퀀스 번호를 생성하려면 NEXTVAL 포트를 사용합니다. NEXTVAL 포트를 연결하여 현재 값 및 증분 범위 속성에 따라 시퀀스를 생성합니다. 시퀀스 생성기가 시퀀스를 반복하도록 구성되지 않은 경우 NEXTVAL 포트가 구성된 끝 값까지 시퀀스 번호를 생성합니다.

예를 들어 NEXTVAL을 매핑의 두 대상에 연결하여 고유한 기본 키 값을 생성할 수 있습니다. 통합 서비스가 각 대상 테이블에 대해 고유한 기본 키 값 열을 생성합니다. 고유한 기본 키 값 열은 시퀀스 번호 블록으로 하나의 대상 테이블에 전송됩니다. 첫 번째 대상이 시퀀스 번호 블록을 수신한 다음 다른 대상이 시퀀스 생성기 변환에서 시퀀스 번호 블록을 수신합니다.

예를 들어 다음과 같은 시퀀스 생성기 변환을 구성합니다. 현재 값 = 1, 증분 범위 = 1. 통합 서비스가 T\_ORDERS\_PRIMARY 및 T\_ORDERS\_FOREIGN 대상 테이블에 대해 다음 기본 키 값을 생성합니다.

**T\_ORDERS\_PRIMARY TABLE:**  
**PRIMARY KEY**

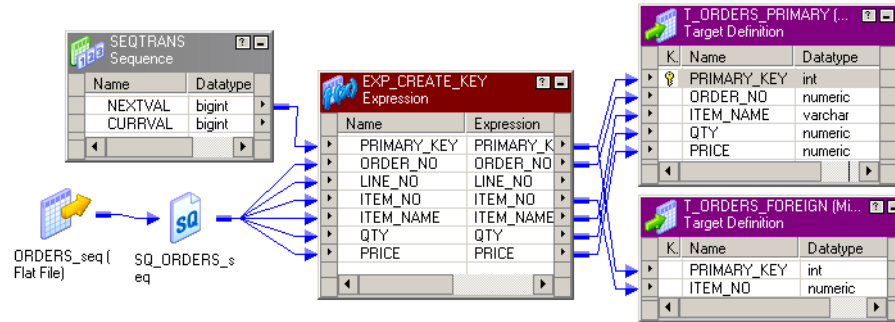
1  
2  
3  
4  
5

**T\_ORDERS\_FOREIGN TABLE:**  
**PRIMARY KEY**

6  
7  
8  
9  
10

단일 변환에서 데이터를 수신하는 둘 이상의 대상에 동일한 값을 전송하려면 시퀀스 생성기 변환을 그 이전 변환에 연결하면 됩니다. 통합 서비스가 값을 시퀀스 번호 블록으로 처리합니다. 이를 통해 통합 서비스가 고유한 값을 변환에 전달한 다음 변환에서 대상으로 행을 라우팅할 수 있습니다.

다음 그림은 고유한 값을 식 변환에 전달하는 시퀀스 생성기와의 매핑을 보여줍니다.



식 변환이 두 대상을 동일한 기본 키 값으로 채웁니다.

예를 들어 다음과 같은 시퀀스 생성기 변환을 구성합니다. 현재 값 = 1, 증분 범위 = 1. 통합 서비스가 T\_ORDERS\_PRIMARY 및 T\_ORDERS\_FOREIGN 대상 테이블에 대해 다음 기본 키 값을 생성합니다.

#### T\_ORDERS\_PRIMARY TABLE: PRIMARY KEY

1  
2  
3  
4  
5

#### T\_ORDERS\_FOREIGN TABLE: PRIMARY KEY

1  
2  
3  
4  
5

**참고:** 그리드에서 분할된 세션을 실행하는 경우 시퀀스 생성기 변환은 각 파티션의 행 숫에 따라 값을 건너뛰니다.

## 키 작성

시퀀스 생성기 변환에서 NEXTVAL 포트를 대상 또는 다운스트림 변환에 연결하여 기본 또는 외래 키 값을 작성할 수 있습니다. 1에서 9,223,372,036,854,775,807 범위의 값을 최소 간격 1로 사용할 수 있습니다.

시퀀스 생성기 변환에서 NEXTVAL 포트를 대상 또는 다운스트림 변환에 연결하여 기본 또는 외래 키 값을 작성할 수 있습니다. 1에서 9,223,372,036,854,775,807 범위의 값을 최소 간격 1로 사용할 수 있습니다. 복합 키를 작성하여 테이블의 각 행을 식별할 수도 있습니다.

기본 또는 외래 키를 작성할 때는 주기 옵션을 사용하여 통합 서비스가 중복 기본 키를 작성하지 않도록 하십시오. 이렇게 하려면 세션 속성에서 대상 테이블 잘라내기 옵션을 선택하거나 복합 키를 작성합니다.

복합 키를 작성하려면 통합 서비스의 주기를 더 작은 값 집합으로 구성합니다. 예를 들어 세 곳의 매장에서 주문 번호가 생성되는 경우 시퀀스 생성기 변환의 주기 값을 1에서 3으로 구성하고 1씩 증분되도록 구성합니다. ORDER\_NO 포트를 시퀀스 생성기 변환에 연결하면 생성된 값에서 고유한 복합 키가 작성됩니다.

다음 예는 복합 키 및 주문 순서를 보여 줍니다.

#### COMPOSITE\_KEY

1  
2

#### ORDER\_NO

12345  
12345

## COMPOSITE\_KEY

3  
1  
2  
3

## ORDER\_NO

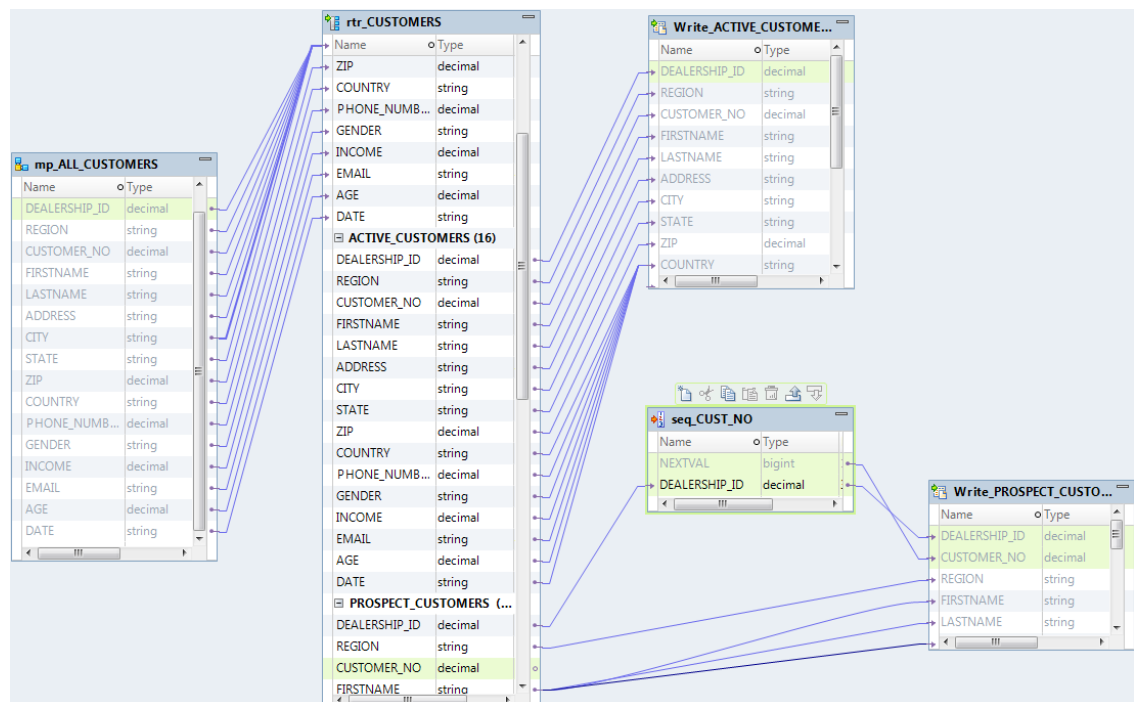
12345  
12346  
12346  
12346

## 누락된 값 바꾸기

시퀀스 생성기 변환을 사용하여 누락된 키를 바꾸는 경우 라우터 변환을 사용하여 값이 할당된 열에서 Null 값을 필터링할 수도 있습니다. 라우터 변환을 시퀀스 생성기 변환에 연결하고 NEXTVAL을 사용하여 Null 값을 채우기 위한 숫자 값 시퀀스를 생성합니다.

예를 들어 CUSTOMER\_NO 열에서 Null 값을 바꾸려면 고객 데이터가 포함된 소스와의 매핑을 작성합니다. 라우터 변환을 추가하여 Null 값을 가진 고객에서 할당된 고객 번호를 가진 고객을 필터링합니다. 시퀀스 생성기 변환을 추가하여 고유한 CUSTOMER\_NO 값을 생성합니다. 데이터를 쓸 고객 대상을 추가합니다.

다음 이미지는 CUSTOMER\_NO 열에서 Null 값을 바꾸는 매핑을 보여줍니다.



시퀀스 생성기 변환을 사용하여 IIF 및 ISNULL 함수와 함께 NEXTVAL을 사용하여 누락된 키를 바꿀 수 있습니다.

예를 들어 ORDER\_NO 열에서 Null 값을 바꾸려면 속성을 사용하여 시퀀스 생성기 변환을 작성하고 NEXTVAL 포트를 식 변환으로 끕니다. 식 변환에서 ORDER\_NO 포트를 다른 필요한 포트와 함께 변환으로 끕니다. 그런 다음 출력 포트, ALL\_ORDERS를 작성합니다.

그런 다음 ALL\_ORDERS에서 다음 식을 입력하여 Null 순서를 바꿉니다.

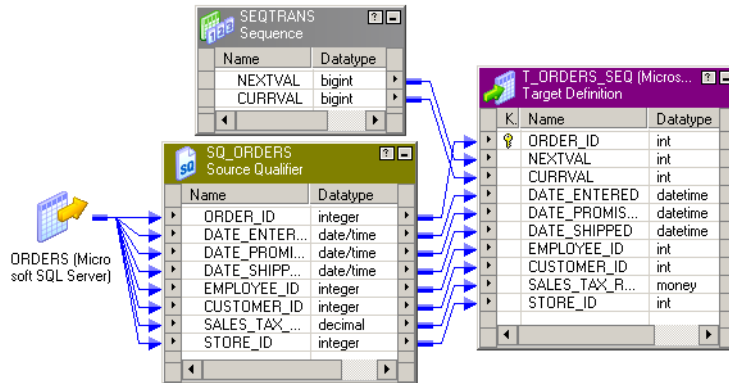
```
IIF( ISNULL( ORDER_NO ), NEXTVAL, ORDER_NO )
```



## CURRVAL

CURRVAL은 NEXTVAL에 증분 값을 더한 것입니다. CURRVAL 포트는 NEXTVAL 포트가 이미 다운스트림 변환에 연결된 경우에만 연결합니다. 행이 CURRVAL 포트에 연결된 변환으로 들어가면 통합 서비스가 마지막으로 작성된 NEXTVAL 값에 1을 더한 값을 전달합니다.

다음 그림은 CURRVAL 및 NEXTVAL 포트를 대상에 연결하는 작업을 보여 줍니다.



예를 들어 다음과 같은 시퀀스 생성기 변환을 구성합니다. 현재 값 = 1, 증분 범위 = 1. 이 경우 통합 서비스는 NEXTVAL 및 CURRVAL에 대해 다음 값을 생성합니다.

### NEXTVAL

1  
2  
3  
4  
5

### CURRVAL

2  
3  
4  
5  
6

NEXTVAL 포트를 연결하지 않고 CURRVAL 포트를 연결하면 통합 서비스가 각 행에 대해 상수 값을 전달합니다. 시퀀스 생성기 변환에서 CURRVAL 포트를 연결하면 통합 서비스가 각 블록에서 1개의 행을 전달합니다. 매핑에서 NEXTVAL 포트만 연결하면 성능을 최적화할 수 있습니다.

**참고:** 그리드에서 분할된 세션을 실행하는 경우 시퀀스 생성기 변환이 각 파티션의 행 수에 따라 값을 건너뛰을 수도 있습니다.

# 시퀀스 생성기 변환 속성

통합 서비스가 순차적 값을 생성하는 데 사용하는 변환 속성을 구성합니다.

다음 테이블에는 시퀀스 데이터 개체 및 새 시퀀스에 구성하는 속성이 나열되어 있습니다.

속성	설명
시작 값	주기 옵션을 사용하는 경우 통합 서비스가 사용할 생성된 시퀀스의 시작 값입니다. 주기를 선택하는 경우 끝 값에 도달하면 통합 서비스가 이 값으로 다시 순환합니다. 기본값은 0입니다. 최대값은 9,223,372,036,854,775,806입니다.
끝 값	통합 서비스가 생성하는 최대값입니다. 세션 중 통합 서비스가 이 값에 도달하고 시퀀스가 순환하도록 구성되지 않은 경우 세션이 실패합니다. 최대값은 9,223,372,036,854,775,807입니다.
증분 값	NEXTVAL 포트의 두 연속 값 사이의 차이입니다. 기본값은 1입니다. 양의 정수여야 합니다. 최대값은 2,147,483,647입니다.
주기	활성화된 경우 통합 서비스가 시퀀스 범위를 순환하고 시작 값을 사용하여 다시 시작합니다. 비활성화된 경우 통합 서비스가 구성된 끝 값에서 시퀀스를 중지합니다. 끝 값에 도달했지만 여전히 처리할 행이 있는 경우 통합 서비스에서 오버플로우 오류가 발생하고 세션을 실행하지 못합니다.
재설정	활성화된 경우 통합 서비스가 매핑 실행이 완료될 때 시퀀스 데이터 개체를 시작 값으로 재설정합니다. 비활성화된 경우 통합 서비스가 매핑 실행이 완료된 후 현재 값을 증분하고 다음 매핑 실행에서 해당 값을 사용합니다. 재사용 가능 시퀀스 생성기 변환 및 재사용 가능 시퀀스 데이터 개체를 사용하는 재사용 불가능 시퀀스 생성기 변환의 경우 이 속성이 비활성화됩니다.
추적 수준	통합 서비스가 매핑 로그에 쓰는 변환에 대한 세부 정보 수준입니다. 간단, 보통, 자세한 정보 표시 초기화 또는 자세한 정보 표시 데이터를 선택할 수 있습니다. 기본값은 보통입니다.
행 순서 유지	변환에 대한 입력 데이터의 행 순서를 유지합니다. 통합 서비스에서 행 순서를 변경할 수 있는 어떤 최적화도 수행하지 않아야 하는 경우 이 옵션을 선택합니다. 기본값은 false입니다.

다음 목록에는 구성할 수 있는 시퀀스 생성기 변환 속성이 설명되어 있습니다.

## 시작 값

주기 옵션을 사용하는 경우 통합 서비스가 사용할 생성된 시퀀스의 시작 값입니다. 주기를 선택하는 경우 끝 값에 도달하면 통합 서비스가 이 값으로 다시 순환합니다.

기본값은 0입니다.

최대값은 9,223,372,036,854,775,806입니다.

## 증분 범위

NEXTVAL 포트의 두 연속 값 사이의 차이입니다.

기본값은 1입니다.

양의 정수여야 합니다.

최대값은 2,147,483,647입니다.

#### 끝 값

통합 서비스가 생성하는 최대값입니다. 세션 중 통합 서비스가 이 값에 도달하고 시퀀스가 순환하도록 구성되지 않은 경우 세션이 실패합니다.

최대값은 9,223,372,036,854,775,807입니다.

NEXTVAL 포트를 다운스트림 정수 포트에 연결한 경우 끝 값을 정수 최대값보다 작은 값으로 설정하십시오. NEXTVAL이 다운스트림 포트의 데이터 유형 최대값을 초과할 경우 세션이 실패합니다.

#### 현재 값

시퀀스의 현재 값입니다. 통합 서비스에서 사용하도록 할 값을 시퀀스의 첫 번째 값으로 입력합니다. 일련의 값이 순환하게 하려면 값이 시작 값보다 크거나 같고 끝 값보다 작아야 합니다.

총 캐시된 값이 0으로 설정된 경우 통합 서비스는 현재 값을 업데이트하여 세션+1에 대해 마지막으로 생성된 값을 반영한 다음 업데이트된 현재 값을 이 세션의 다음 번 실행에 대한 기준으로 사용합니다. 그러나 재설정 옵션을 사용하면 통합 서비스는 각 세션 이후 이 값을 원래 값으로 재설정합니다.

**참고:** 이 설정을 편집하는 경우 시퀀스를 새 설정으로 재설정합니다. 현재 값을 10으로 재설정하고 증분이 1이면 다음 번에 세션을 사용할 때 통합 서비스가 첫 번째 값, 10을 생성합니다.

최대값은 9,223,372,036,854,775,806입니다. 현재 값이 최대값을 초과하면 통합 서비스가 값을 NULL로 설정합니다.

#### 주기

활성화된 경우 통합 서비스가 시퀀스 범위를 순환하고 시작 값을 사용하여 다시 시작합니다.

비활성화된 경우 통합 서비스가 구성된 끝 값에서 시퀀스를 중지합니다. 끝 값에 도달했지만 여전히 처리할 행이 있는 경우 통합 서비스에서 오버플로우 오류가 발생하고 세션을 실행하지 못합니다.

#### 총 캐시된 값

한 번에 통합 서비스가 캐시하는 순차적 값 수입니다. 여러 세션이 동시에 동일한 재사용 가능 시퀀스 생성기를 사용하는 경우 이 옵션을 사용하여 각 세션이 고유한 값을 수신하게 할 수 있습니다. 통합 서비스가 각 값을 캐시할 때 리포지토리를 업데이트합니다. 0으로 설정되면 통합 서비스가 값을 캐시하지 않습니다.

기본값은 0입니다.

재사용 가능 시퀀스 생성기에 대한 기본값은 1,000입니다.

최대값은 9,223,372,036,854,775,807입니다.

#### 재설정

활성화된 경우 통합 서비스가 각 세션의 원래 현재 값에 따라 값을 생성합니다. 비활성화된 경우 통합 서비스가 현재 값을 업데이트하여 세션+1에 대해 마지막으로 생성된 값을 반영한 다음 업데이트된 현재 값을 다음 세션 실행에 대한 기준으로 사용합니다.

이 속성은 재사용 가능 시퀀스 생성기 변환에 대해 비활성화됩니다.

#### 추적 수준

통합 서비스가 세션 로그에 쓰는 변환에 대한 세부 정보 수준입니다.

## 시작 값

주기를 사용하여 반복 시퀀스를 생성할 수 있습니다(예: 1년의 달에 해당하도록 1부터 12까지의 숫자).

1. 통합 서비스가 시작 값으로 사용하게 하려는 시퀀스의 가장 낮은 값을 입력합니다.

2. 끝 값에 사용할 가장 높은 값을 입력합니다.
3. 주기를 선택합니다.

순환하는 동안 통합 서비스가 시퀀스의 구성된 끝 값에 도달하면 구성된 시작 값부터 시작하여 주기를 다시 시작합니다.

## 증분 범위

통합 서비스에서 NEXTVAL 포트의 시퀀스는 시퀀스 생성기 변환의 현재 값 및 증분 범위 속성을 바탕으로 생성됩니다.

현재 값 속성은 통합 서비스가 각 세션에 대한 시퀀스 작성을 시작할 때의 값입니다. 증분 범위는 통합 서비스가 시퀀스의 새 값을 작성할 때 기존 값에 추가하는 정수입니다. 기본적으로 현재 값은 1로 설정되고 증분 범위는 1로 설정됩니다.

예를 들어 현재 값이 1,000이고 증분 범위가 10인 시퀀스 생성기 변환을 작성할 수 있습니다. 매핑을 통해 행 3개를 전달하는 경우 통합 서비스는 다음과 같은 값 집합을 생성합니다.

```
1000
1010
1020
```

## 끝 값

통합 서비스에서 생성하도록 할 최대값이 끝 값입니다. 통합 서비스가 끝 값에 도달하고 시퀀스 생성기가 시퀀스를 반복하도록 구성되지 않은 경우 세션이 오버플로우 오류와 함께 실패합니다.

통합 서비스에서 생성하도록 할 최대값이 끝 값입니다. 통합 서비스가 끝 값에 도달하고 시퀀스 생성기가 시퀀스를 반복하도록 구성되지 않은 경우 매핑 실행이 오버플로우 오류와 함께 실패합니다.

끝 값을 1과 9,233,372,036,854,775,807 사이의 정수로 설정하십시오. NEXTVAL 포트를 다운스트림 정수 포트에 연결한 경우 끝 값을 정수 최대값보다 작은 값으로 설정하십시오. 예를 들어, NEXTVAL 포트를 작은 정수 포트에 연결한 경우 끝 값을 최대 32,767로 설정하십시오. NEXTVAL이 다운스트림 포트의 데이터 유형 최대값을 초과할 경우 세션이 실패합니다.

끝 값을 1과 9,233,372,036,854,775,807 사이의 정수로 설정하십시오. NEXTVAL 포트를 다운스트림 정수 포트에 연결한 경우 끝 값을 정수 최대값보다 작은 값으로 설정하십시오. 예를 들어, NEXTVAL 포트를 작은 정수 포트에 연결한 경우 끝 값을 최대 32,767로 설정하십시오. NEXTVAL이 다운스트림 포트의 데이터 유형 최대값을 초과할 경우 매핑이 실패합니다.

## 증분값

통합 서비스에서 NEXTVAL 포트의 시퀀스는 시퀀스 생성기 변환의 현재 값 및 증분 범위 속성을 바탕으로 생성됩니다.

현재 값 속성은 통합 서비스가 각 세션에 대한 시퀀스 작성을 시작할 때의 값입니다. 증분 범위는 통합 서비스가 시퀀스의 새 값을 작성할 때 기존 값에 추가하는 정수입니다. 기본적으로 현재 값은 1로 설정되고 증분 범위는 1로 설정됩니다.

예를 들어 현재 값이 1,000이고 증분 범위가 10인 시퀀스 생성기 변환을 작성할 수 있습니다. 매핑을 통해 행 3개를 전달하는 경우 통합 서비스는 다음과 같은 값 집합을 생성합니다.

```
1000
1010
1020
```

## 값 범위 반복

시퀀스 생성기 변환에 대한 값 범위를 설정할 수 있습니다. 반복 옵션을 사용할 경우 시퀀스 생성기 변환에서 끝 값에 도달하면 범위를 반복합니다.

예를 들어, 시퀀스 범위가 10에서 시작하여 50에서 끝나도록 설정하고 증가 값을 10으로 설정한 경우 시퀀스 생성기 변환에서 10, 20, 30, 40, 50 값을 작성합니다. 시퀀스가 다시 10에서 시작됩니다.

## 현재 값

통합 서비스는 각 세션에 생성되는 값에 대한 기준으로 현재 값을 사용합니다. 통합 서비스가 시퀀스 생성기 변환을 처음 사용할 때 사용할 값을 표시하려면 해당 값을 현재 값으로 입력해야 합니다. 시퀀스 생성기 변환을 사용하여 일련의 값을 반복하려면 현재 값이 시작 값보다 크거나 같고 끝 값보다 작아야 합니다.

통합 서비스는 각 매핑 실행에 대해 생성되는 값의 기준으로 현재 값을 사용합니다. 통합 서비스는 항상 시작 값을 시퀀스 생성기 변환의 현재 값으로 사용합니다.

각 세션이 끝나면 통합 서비스는 캐시된 값의 시퀀스 생성기 번호가 0일 경우 세션에 생성된 마지막 값에 1을 더한 값으로 현재 값을 업데이트합니다. 예를 들어, 통합 서비스가 생성된 값 101로 세션을 종료한 경우 리포지토리의 시퀀스 생성기 현재 값을 102로 업데이트합니다. 다음에 시퀀스 생성기가 사용될 때 통합 서비스는 다음 생성되는 값의 기준으로 102를 사용합니다. 시퀀스 생성기 증분 범위가 1일 경우 통합 서비스가 시퀀스 생성기를 사용하여 다른 세션을 시작하면 첫 번째 생성되는 값은 102입니다.

각 매핑 실행이 끝날 때 통합 서비스는 캐시된 값의 시퀀스 생성기 번호가 0일 경우 세션에 생성된 마지막 값에 1을 더한 값으로 현재 값을 업데이트합니다. 예를 들어, 통합 서비스가 생성된 값 101로 세션을 종료한 경우 리포지토리의 시퀀스 생성기 현재 값을 102로 업데이트합니다. 다음에 시퀀스 생성기가 사용될 때 통합 서비스는 다음 생성되는 값의 기준으로 102를 사용합니다. 시퀀스 생성기 증분 범위가 1일 경우 통합 서비스가 시퀀스 생성기를 사용하여 다른 세션을 시작하면 첫 번째 생성되는 값은 102입니다.

여러 버전의 시퀀스 생성기 변환이 있는 경우 통합 서비스는 세션을 실행할 때 모든 버전의 현재 값을 업데이트합니다. 통합 서비스는 사용자가 시퀀스 생성기 변환 또는 상위 매핑을 체크 아웃했는지 여부와 상관없이 버전에서 현재 값을 업데이트합니다. 두 개 값이 다를 경우 업데이트된 현재 값이 시퀀스 생성기 변환의 편집된 현재 값을 재정의합니다.

예를 들어, 사용자 1이 시퀀스 생성기 변환을 작성하고 현재 값 10을 시퀀스 생성기 버전 1에 저장한 상태에서 체크 인합니다. 사용자 1이 시퀀스 생성기 변환을 체크 아웃하고 새로운 현재 값 100을 시퀀스 생성기 버전 2에 입력합니다. 사용자 1은 시퀀스 생성기 변환을 계속 체크 아웃합니다. 반면 사용자 2는 시퀀스 생성기 변환 버전 1을 사용하는 세션을 실행합니다. 통합 서비스는 사용자 2가 세션을 실행할 때 체크 인 값 10을 현재 값으로 사용합니다. 세션이 완료되면 현재 값은 150입니다. 통합 서비스는 사용자 1이 시퀀스 생성기 변환을 체크 아웃했다라도 시퀀스 생성기 변환의 버전 1과 버전 2에 대해 현재 값을 150으로 업데이트합니다.

세션을 실행한 후 매핑을 열 경우 현재 값이 세션에 생성된 마지막 값에 1을 더한 값으로 표시됩니다. 통합 서비스가 현재 값을 사용하여 각 세션의 첫 번째 값을 결정하기 때문에 시퀀스를 재설정할 경우에만 현재 값을 편집해야 합니다.

여러 버전의 시퀀스 생성기 변환이 있고 시퀀스를 재설정할 경우, 현재 값을 수정한 후 매핑 또는 재사용 가능한 시퀀스 생성기 변환을 체크 인해야 합니다.

**참고:** 시퀀스 생성기를 재설정하도록 구성한 경우 통합 서비스는 각 세션의 첫 번째 생성되는 값의 기준으로 현재 값을 사용합니다.

## 총 캐시된 값

총 캐시된 값에 따라 통합 서비스가 한 번에 캐시하는 값 수가 결정됩니다. 총 캐시된 값이 0보다 큰 경우 통합 서비스가 구성된 값 수를 캐시하고 값을 캐시할 때마다 현재 값을 업데이트합니다.

여러 세션이 동시에 동일한 재사용 가능 시퀀스 생성기 변환을 사용하는 경우 여러 인스턴스의 시퀀스 생성기 변환이 있을 수 있습니다. 각 세션에 대해 동일한 값을 생성하지 않도록 방지하려면 총 캐시된 값을 구성하여 각 세션에 대해 시퀀스 값 범위를 예약합니다.

**팁:** 그리드에서 세션을 실행할 때 성능을 향상시키려면 시퀀스 생성기 변환에 대해 총 캐시된 값을 늘립니다. 그러면 마스터 및 작업자 DTM 프로세스와 리포지토리 사이에 필요한 통신이 감소합니다.

## 재사용 불가능 시퀀스 생성기

재사용 불가능 시퀀스 생성기 변환에서는 총 캐시된 값이 0으로 기본 설정되며 통합 서비스는 세션 중에 값을 캐시하지 않습니다. 통합 서비스는 값을 캐시하지 않을 때 세션 시작 시 리포지토리에서 현재 값에 액세스합니다. 그런 다음 시퀀스에 대해 값을 생성합니다. 세션이 종료되면 통합 서비스는 리포지토리에서 현재 값을 업데이트합니다.

총 캐시된 값을 0보다 크게 설정하면 통합 서비스는 세션 중에 값을 캐시합니다. 세션이 시작될 때 통합 서비스는 리포지토리에서 현재 값에 액세스하여 구성된 값의 수를 캐시한 다음 그에 따라 현재 값을 업데이트합니다. 통합 서비스는 캐시의 모든 값을 사용하는 경우 리포지토리에서 다음 값 집합에 액세스하여 현재 값을 업데이트합니다. 세션이 종료되면 통합 서비스는 캐시에서 나머지 값을 무시합니다.

재사용 불가능 시퀀스 생성기 변환에서는 총 캐시된 값을 0보다 크게 설정하면 통합 서비스가 세션 중에 리포지토리에 액세스하는 횟수가 증가할 수 있습니다. 또한 각 세션이 종료될 때 사용되지 않은 캐시된 값이 무시되므로 특정 값 섹션을 건너뜁니다.

시퀀스 생성기 변환을 다음과 같이 구성하는 경우를 예로 들어 보겠습니다. 총 캐시된 값 = 50, 현재 값 = 1, 증분 범위 = 1. 통합 서비스는 세션을 시작할 때 세션에 대해 값 50개를 캐시하고 리포지토리에서 현재 값을 50으로 업데이트합니다. 통합 서비스는 세션에 대해 값 1~39를 사용하고 사용되지 않은 값인 40~49를 무시합니다. 통합 서비스는 해당 세션을 다시 실행할 때 리포지토리에서 현재 값인 50을 확인합니다. 그런 후 다음 50개의 값을 캐시하고 현재 값을 100으로 업데이트합니다. 세션 중에는 값 50~98이 사용됩니다. 두 세션에 대해 생성되는 값은 1~39 및 50~98입니다.

## 재사용 가능 시퀀스 생성기

여러 세션에 재사용 가능 시퀀스 생성기 변환이 있고 동시에 해당 세션이 실행될 때 총 캐시된 값을 사용하여 각 세션이 시퀀스에서 고유한 값을 수신하도록 할 수 있습니다. 기본적으로 재사용 가능 시퀀스 생성기에 대해 총 캐시된 값이 1000으로 설정됩니다.

여러 세션이 동시에 동일한 시퀀스 생성기 변환을 사용하는 경우 각 세션에 대해 동일한 값을 생성할 위험이 있습니다. 이를 방지하려면 총 캐시된 값을 구성하여 통합 서비스가 각 세션에 대해 총 값 집합을 캐시하게 합니다.

예를 들어 재사용 가능 시퀀스 생성기 변환을 다음과 같이 구성합니다. 총 캐시된 값 = 50, 현재 값 = 1, 증분 범위 = 1. 두 개의 세션이 시퀀스 생성기를 사용하고 거의 동시에 실행하도록 예약되어 있습니다. 통합 서비스가 첫 번째 세션을 시작할 때 세션에 대해 50개의 값을 캐시하고 리포지토리에서 현재 값을 50으로 업데이트합니다. 통합 서비스가 세션에서 1~50의 값을 사용하여 시작합니다. 통합 서비스가 두 번째 세션을 시작할 때 현재 값(50으로 설정)에 대해 리포지토리를 확인합니다. 그런 후 다음 50개의 값을 캐시하고 현재 값을 100으로 업데이트합니다. 그런 다음 두 번째 세션에서 51~100의 값을 사용합니다. 어떤 세션이든 해당 캐시된 값을 모두 사용하면 통합 서비스가 새 값 집합을 캐시하고 현재 값을 업데이트하여 이러한 값이 시퀀스 생성기에 대해 고유하게 유지되도록 합니다.

재사용 가능 시퀀스 생성기 변환의 경우 총 캐시된 값을 줄이면 삭제되는 값을 최소화할 수 있지만 1보다는 커야 합니다. 총 캐시된 값을 줄이는 경우 세션 중에 값을 캐시하기 위해 통합 서비스가 리포지토리에 액세스하는 횟수를 늘릴 수 있습니다.

## 재설정

재사용 불가능 시퀀스 생성기 변환에 대해 재설정을 선택하면 통합 서비스는 세션을 시작할 때마다 원래 시작 값을 기준으로 값을 생성합니다. 그렇지 않으면 통합 서비스는 마지막으로 생성된 값에 증분 값을 더한 결과를 반영하도록 현재 값을 업데이트하고 다음번에 시퀀스 생성기 변환을 사용할 때 업데이트된 값을 사용합니다.

재설정 속성을 사용하도록 재사용 불가능 시퀀스 생성기 변환을 구성하면 통합 서비스가 각 매핑 실행에 대해 원래 시작 값을 사용합니다. 그렇지 않으면 통합 서비스는 현재 값을 증분하고 다음 매핑 실행에서 해당 값을 사용합니다.

1씩 증가하는 1에서 1,000까지의 값을 생성하도록 시퀀스 생성기 변환을 구성하는 경우를 예로 들어 보겠습니다. 이 경우 시작 값을 1로 재설정하도록 선택합니다. 첫 번째 세션 실행 중에 통합 서비스는 숫자 1~234를 생성합니다. 각 후속 매핑 실행에서 통합 서비스는 초기 값 1부터 숫자를 다시 생성합니다.

예를 들어 시작 값을 1로 설정하여 1씩 증가하는 1에서 1,000까지의 값을 생성하도록 시퀀스 생성기 변환을 구성하고 재설정을 선택한다고 가정해 보겠습니다. 첫 번째 매핑 실행 중에 통합 서비스는 숫자 1~234를 생성합니다. 각 후속 매핑 실행에서 통합 서비스는 시작 값 1부터 숫자를 다시 생성합니다.

재설정을 수행하지 않으면 통합 서비스는 첫 번째 실행 종료 시에 현재 값을 235로 업데이트합니다. 그러면 다음 번에 시퀀스 생성기 변환을 사용할 때 처음으로 생성되는 값이 235가 됩니다.

**참고:** 재사용 가능 시퀀스 생성기 변환에 대해서는 재설정이 비활성화됩니다.

## 행 순서 유지

변환에 대한 입력 데이터의 행 순서를 유지합니다. 통합 서비스에서 행 순서를 변경할 수 있는 어떤 최적화도 수행하지 않아야 하는 경우 이 옵션을 선택합니다.

통합 서비스가 최적화를 수행하면 이전에 매핑에 설정된 순서를 잃게 될 수 있습니다. 정렬된 플랫폼 파일 소스, 정렬된 관계형 소스 또는 분류기 변환을 사용하여 매핑에서 순서를 설정할 수 있습니다. 행 순서를 유지하기 위해 변환을 구성할 경우 통합 서비스는 매핑에 대해 최적화를 수행할 때 이 구성을 고려합니다. 통합 서비스는 순서를 유지할 수 있는 경우에 변환에 대해 최적화를 수행합니다. 통합 서비스는 최적화가 행 순서를 변경할 수 있는 경우에는 변환에 대해 최적화를 수행하지 않습니다.

## 시퀀스 데이터 개체

시퀀스 데이터 개체는 숫자 값 시퀀스를 작성하고 유지합니다. 시퀀스 생성기 변환이 시퀀스 데이터 개체를 사용하여 변환에 대한 값을 생성합니다.

여러 시퀀스 생성기 변환에서 재사용 가능 시퀀스 데이터 개체를 사용할 수 있습니다. 동일한 통합 서비스에서 실행되는 경우 동일한 시퀀스 데이터 개체를 사용하는 모든 시퀀스 생성기 변환은 동일한 시퀀스 값을 사용합니다. 재사용 불가능 시퀀스 생성기 변환에서 재사용 가능 시퀀스 데이터 개체를 사용할 수도 있습니다. 재사용 불가능 시퀀스 데이터 개체는 재사용 불가능 시퀀스 생성기 변환에서 사용할 수 있습니다.

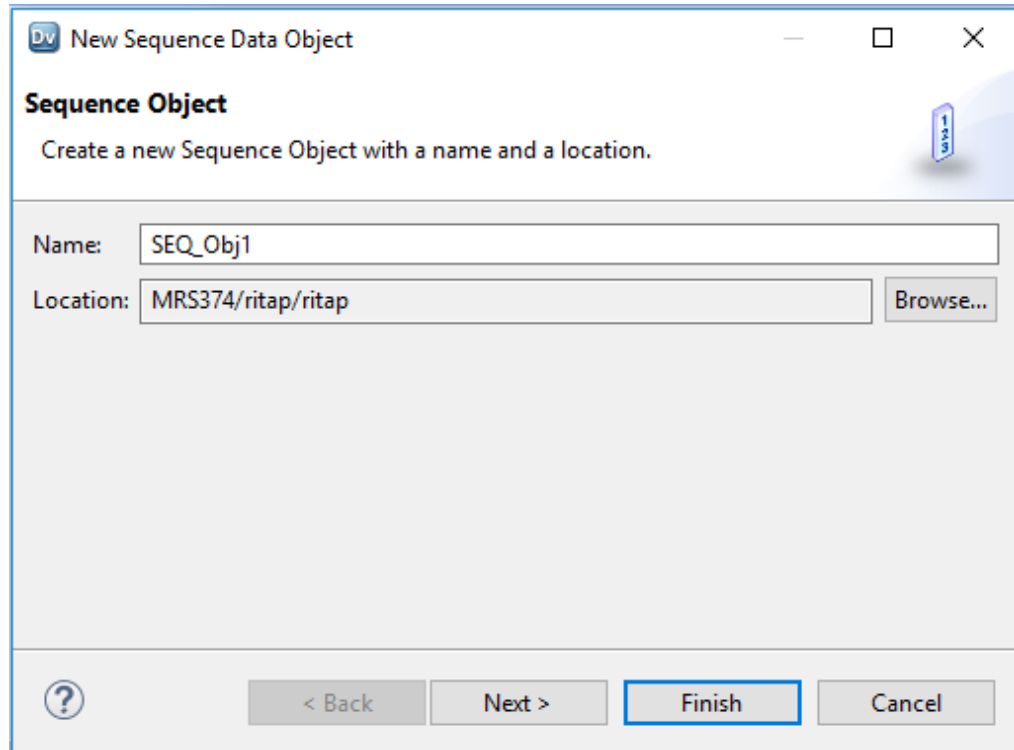
예를 들어 관계형 테이블의 동일한 기본 키 필드에 쓰는 여러 매핑을 작성합니다. 각 매핑은 동일한 재사용 가능 시퀀스 데이터 개체를 사용하고 동일한 통합 서비스에서 실행되는 동일한 재사용 가능 시퀀스 생성기 변환을 사용합니다. 각 매핑이 고유한 값을 기본 키 필드에 씁니다.



## 시퀀스 데이터 개체 생성

시퀀스 데이터 개체를 사용하여 시퀀스 생성기 변환을 생성하려면 시퀀스 데이터 개체를 생성하고, 개체 속성을 구성하고, 시퀀스 생성기 변환 대화 상자에서 개체를 선택합니다.

1. 매핑 편집기에서 매핑 팔레트를 아래로 스크롤하여 시퀀스 생성기 변환을 찾고 매핑으로 끌어서 놓습니다.  
새 변환 마법사가 열립니다.
2. 새 시퀀스 데이터 개체를 클릭합니다.  
새 데이터 개체 마법사가 열립니다.

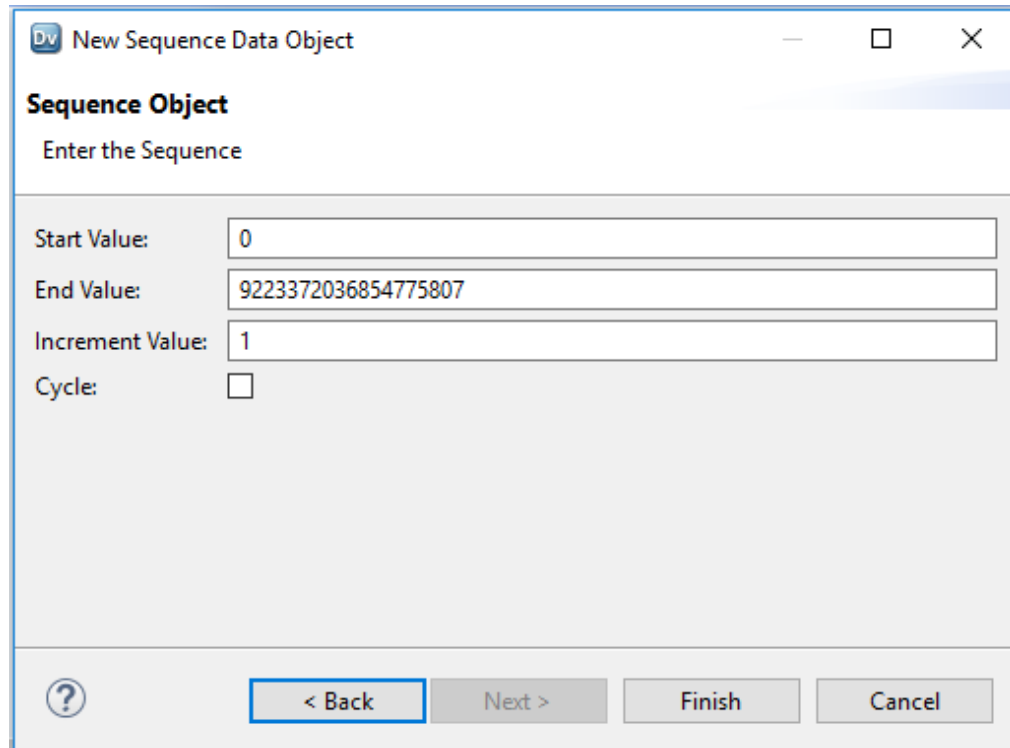


The image shows a dialog box titled "New Sequence Data Object" with a "Dv" icon in the top-left corner. The dialog has standard window controls (minimize, maximize, close) in the top-right. Below the title bar, the text "Sequence Object" is displayed in bold, followed by the instruction "Create a new Sequence Object with a name and a location." To the right of this text is a small icon of a data object. The main area of the dialog contains two input fields: "Name:" with the text "SEQ\_Obj1" and "Location:" with the text "MRS374/ritap/ritap". To the right of the "Location:" field is a "Browse..." button. At the bottom of the dialog, there is a row of buttons: a help button (question mark icon), "< Back", "Next >", "Finish" (which is highlighted with a blue border), and "Cancel".

3. 시퀀스 데이터 개체의 이름을 입력합니다.  
시퀀스 데이터 개체의 이름 지정 규칙은 SEQ\_<데이터 개체 이름>입니다.
4. 다음을 클릭하여 시퀀스 데이터 개체 속성을 구성합니다.



개체에서 시퀀스 생성기 변환을 생성하는 경우 데이터 개체에 대해 입력한 속성이 변환에 사용됩니다. 다음 이미지는 구성할 수 있는 속성을 보여 줍니다.



The image shows a dialog box titled "New Sequence Data Object". It has a "Sequence Object" section with the instruction "Enter the Sequence". Below this, there are four input fields: "Start Value" with the value "0", "End Value" with the value "9223372036854775807", "Increment Value" with the value "1", and "Cycle" with an unchecked checkbox. At the bottom, there is a help icon (question mark) and four buttons: "< Back", "Next >", "Finish", and "Cancel".

5. 데이터 개체 속성을 구성한 후 시퀀스 데이터 개체를 사용하여 시퀀스 생성기 변환을 생성할 수 있습니다. 변환을 생성할 때 시퀀스 생성기 변환의 이름을 지정하고 **기존 시퀀스 개체 선택**을 선택합니다. 데이터 개체로 이동하고 **확인**을 클릭합니다.

매핑 편집기에 NEXTVAL 출력 전용 포트와 함께 시퀀스 생성기 변환이 표시됩니다. NEXTVAL 포트를 다운스트림 변환 또는 대상에 연결하여 숫자 시퀀스를 생성할 수 있습니다.

## 시퀀스 생성기 변환 작성

시퀀스 생성기 변환을 매핑에서 사용하려면 변환을 매핑에 추가하고 변환 속성을 구성한 다음 NEXTVAL 또는 CURRVAL을 하나 이상의 변환에 연결합니다.

시퀀스 생성기 변환을 작성하려면 다음을 수행하십시오.

1. 매핑 디자이너에서 변환 > 작성을 클릭합니다. 시퀀스 생성기 변환을 선택합니다.  
시퀀스 생성기 변환의 이름 지정 규칙은 `SEQ_TransformationName`입니다.
2. 시퀀스 생성기의 이름을 입력하고 작성을 클릭합니다. 완료를 클릭합니다.  
디자이너가 시퀀스 생성기 변환을 작성합니다.
3. 변환의 제목 표시줄을 두 번 클릭합니다.
4. 변환의 설명을 입력합니다.
5. 속성 탭을 선택합니다. 설정을 입력합니다.

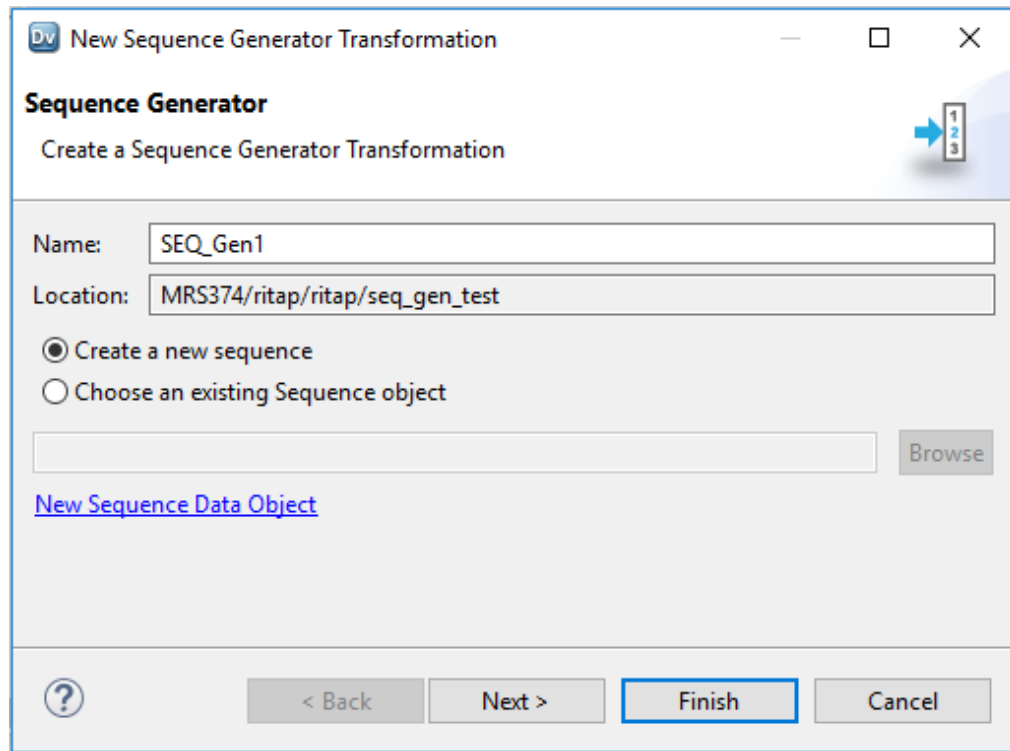
**참고:** 시퀀스 생성기 변환 속성을 세션 수준에서 재정의할 수 없습니다. 이는 생성된 시퀀스 값의 무결성을 보호하기 위한 것입니다.

6. 확인을 클릭합니다.
7. 세션 중에 새 시퀀스를 생성하려면 매핑에서 최소 1개의 변환에 NEXTVAL 포트를 연결합니다.  
다른 변환의 식에서 NEXTVAL 또는 CURRVAL 포트를 사용합니다.

## 시퀀스 생성기 변환 작성

시퀀스 생성기 변환을 매핑에서 사용하려면 변환을 매핑에 추가하고 변환 속성을 구성한 다음 NEXTVAL을 하나 이상의 변환에 연결합니다.

1. 매핑 편집기에서 매핑 팔레트를 아래로 스크롤하여 시퀀스 생성기 변환을 찾고 매핑으로 끌어서 놓습니다.  
**새 변환** 마법사가 열립니다.



2. 시퀀스 생성기 변환의 이름을 입력합니다.  
시퀀스 생성기 변환의 이름 지정 규칙은 SEQ\_<변환 이름>입니다.
3. 새 시퀀스를 생성하거나 기존 시퀀스 개체를 사용하도록 선택합니다.

- 새 시퀀스를 생성하려면 **새 시퀀스 생성**을 선택합니다. **다음**을 클릭하여 시퀀스 속성을 구성합니다. 다음 이미지는 구성할 수 있는 속성을 보여 줍니다.

- 기존 시퀀스 개체를 사용하려면 **기존 시퀀스 개체 선택**을 선택합니다. 시퀀스 개체로 이동하고 **확인**을 클릭합니다.

매핑 편집기에 **NEXTVAL** 출력 전용 포트와 함께 시퀀스 생성기 변환이 표시됩니다. **NEXTVAL** 포트를 다운 스트림 변환 또는 대상에 연결하여 숫자 시퀀스를 생성할 수 있습니다.

## FAQ

### 재사용 불가능 시퀀스 생성기 변환을 재사용 가능하도록 변경할 수 있습니까?

변환을 재사용 가능으로 만들 수는 없지만 변환을 변경하여 시퀀스 데이터 개체를 사용할 수는 있습니다. 시퀀스 데이터 개체는 이것을 사용하는 변환 수와 관계 없이 시퀀스의 무결성을 유지합니다.

### 재사용 불가능 시퀀스 생성기 변환을 맵릿에 배치할 수 있습니까?

아니오, 없습니다. 맵릿은 재사용 가능 개체이므로 맵릿의 모든 개체도 재사용 가능해야 합니다. 대신 재사용 가능 시퀀스 생성기 변환을 사용하십시오.

## 시퀀스 생성기 변환 - 비원시 환경

비원시 환경에서 시퀀스 생성기 변환의 처리는 변환을 실행하는 엔진에 따라 다릅니다.

다음과 같은 비원시 런타임 엔진에 대한 지원을 고려하십시오.

- **Blaze** 엔진. 제한적으로 지원됩니다.
- **Spark** 엔진. 제한적으로 지원됩니다.
- **Databricks Spark** 엔진. 지원되지 않습니다.

### 시퀀스 생성기 변환 - Blaze 엔진

다음 조건이 참인 경우 시퀀스 생성기 변환이 포함된 매핑은 많은 양의 리소스를 소비합니다.

- 변환의 **행 순서 유지** 속성을 *true*로 설정합니다.
- 매핑이 단일 파티션에서 실행됩니다.

### 시퀀스 생성기 변환 - Spark 엔진

시퀀스 생성기 변환은 출력 데이터에서 행 순서를 유지하지 않습니다. 변환에서 **행 순서 유지** 속성을 활성화하는 경우 데이터 통합 서비스는 이 속성을 무시합니다.

## 제 41 장

# 분류기 변환

이 장에 포함된 항목:

- [분류기 변환 개요, 577](#)
- [동적 매핑의 분류기 변환, 578](#)
- [분류기 변환 개발, 578](#)
- [분류기 변환 포트, 578](#)
- [정렬 탭, 579](#)
- [정렬 키 구성, 579](#)
- [분류기 변환 고급 속성, 581](#)
- [분류기 캐시, 582](#)
- [분류기 변환 작성, 582](#)
- [분류기 변환 예제, 583](#)
- [분류기 변환 - 비원시 환경, 584](#)

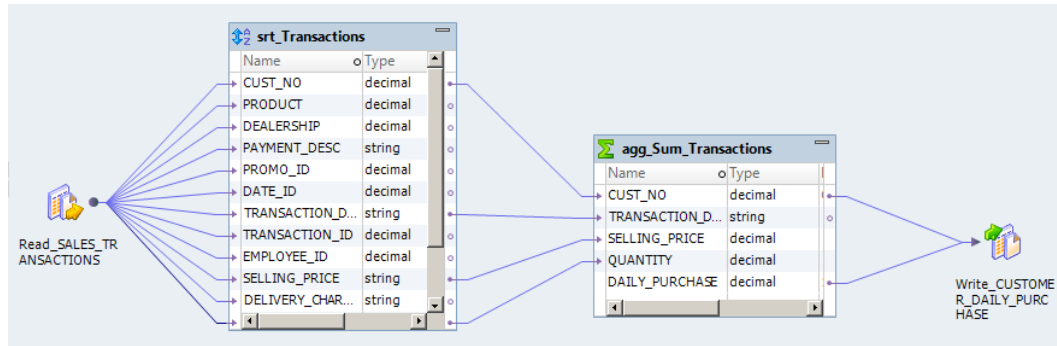
## 분류기 변환 개요

분류기 변환을 사용하여 지정된 정렬 키에 따라 오름차순 또는 내림차순으로 데이터를 정렬합니다. 대/소문자 구분 정렬 및 고유 출력을 위해 분류기 변환을 구성할 수 있습니다. 분류기 변환은 활성 변환입니다.

분류기 변환을 생성할 때는 포트를 정렬 키로 지정하고 각 정렬 키 포트를 오름차순 또는 내림차순으로 정렬하도록 구성합니다. 정렬 키에 대해 여러 포트를 지정하면 데이터 통합 서비스가 각 포트를 순차적으로 정렬합니다.

고객 데이터베이스에서 전체 고객 판매에 대한 송장을 생성해야 하는 경우를 예로 들어 보겠습니다. 이 경우 고객 판매 테이블에서 분류기 변환을 사용하여 고객 번호에 따라 데이터를 내림차순으로 정렬합니다. 분류기 변환의 결과를 집계 변환의 입력으로 사용합니다. 정렬된 입력 옵션을 사용하여 집계 변환 성능을 향상시킬 수 있습니다.

다음 그림은 매핑을 보여 줍니다.



## 동적 매핑의 분류기 변환

동적 매핑에서 분류기 변환을 사용할 수 있습니다. 변환에서 동적 포트를 구성하고 생성된 포트를 참조할 수 있습니다.

분류기 변환에서 동적 포트 또는 생성된 포트를 참조할 수 있습니다. 그러나 런타임 시 생성된 포트가 없는 경우 매핑이 실패합니다.

동적 포트를 정렬 키로 사용하는 경우 데이터 통합 서비스가 동적 포트의 생성된 모든 포트 및 생성된 포트 순서를 고려합니다.

정렬 키를 매개 변수화할 수 있습니다. 정렬 키에 대해 정렬 목록 매개 변수를 사용합니다.

## 분류기 변환 개발

분류기 변환을 개발할 때는 정렬 키 포트, 고유한 출력 행 및 대/소문자 구분 정렬 조건과 같은 요인을 고려해야 합니다.

분류기 변환을 개발하는 경우 다음과 같은 요인을 고려하십시오.

- 정렬 키 및 정렬 방향으로 구성할 포트.
- 대/소문자 구분 정렬 사용 여부.
- Null 값을 정렬 우선 순위로 고려할지 여부.
- 고유 출력 행을 사용할지 여부.
- 설정할 분류기 캐시 크기 값.

## 분류기 변환 포트

분류기 변환에서 포트를 작성하는 경우 기본값으로 입력 및 출력 포트를 작성합니다. 분류기 변환이 입력 포트와 동일한 출력 포트를 반환합니다.

분류기 변환에서 동적 포트를 정의할 수 있습니다. 동적 포트는 매핑의 업스트림 변환으로부터 여러 데이터 열을 받을 수 있습니다. 이를 통해 분류기 변환이 여러 열이 포함된 행을 정렬할 수 있습니다.

속성 보기의 **정렬** 탭에서 정렬 키를 정의합니다.

## 정렬 탭

분류기 변환 **속성** 보기의 **정렬** 탭에서 정렬 키를 정의합니다. 정렬 조건으로 사용할 포트를 하나 이상 선택합니다.

데이터 통합 서비스는 분류기 탭의 포트 순서에 따라 데이터를 정렬합니다. 데이터를 오름차순 또는 내림차순으로 정렬하도록 구성합니다. 기본값은 오름차순입니다.

고유 출력 행에 대해 분류기 변환을 구성하는 경우 **Developer tool**은 모든 포트를 정렬 키의 일부분으로 구성합니다. 데이터 통합 서비스는 정렬 작업 중에 중복 행을 무시합니다.

## 정렬 키 구성

변환 **속성** 보기의 **정렬** 탭에서 정렬 키를 정의합니다.

다음은 개념의 시작 부분입니다.

다음 이미지는 **정렬** 탭을 보여 줍니다.

**Sort**

Output: ☒ All rows ☐ Distinct rows only

**Sort Keys**

Specify by: Value

Ports:		
Department	Ascending (A)	
Employee	Ascending (A)	

Buttons: Add, Choose..., Delete, Move Up, Move Down

**정렬** 탭에는 다음 옵션이 포함되어 있습니다.

## 출력

정렬한 모든 행을 반환할지 또는 중복 행을 무시할지 여부를 선택합니다. 중복 행은 모든 열 값이 동일한 행입니다.

## 지정 기준

값 또는 매개 변수를 선택합니다. 포트 이름을 사용하려면 값을 선택합니다. 정렬 목록 매개 변수를 사용하려면 매개 변수를 선택합니다.

## 추가

수동으로 입력하는 포트 이름을 허용합니다. 추가를 클릭하기 전에 올바른 포트 이름을 입력해야 합니다.

## 선택

선택을 클릭하여 정렬 키에 추가할 포트를 선택합니다. Developer tool에서는 선택할 변환의 포트 목록을 제공합니다.

## 위로 이동 및 아래로 이동

그룹의 포트 순서를 변경할 수 있습니다. 포트 이름을 선택한 다음 이동 단추 중 하나를 클릭하여 정렬 순서에서 위로 이동하거나 아래로 이동합니다.

# 정렬 키 매개 변수화

정렬 키에 대한 포트 목록을 포함하는 정렬 목록 매개 변수를 작성할 수 있습니다.

정렬 변환이 동적 매핑에 있는 경우 정렬 변환에 생성된 포트가 포함될 수 있습니다. 정렬 키를 매개 변수화할 수 있습니다. 정렬할 포트 목록을 포함하는 정렬 목록 매개 변수를 작성합니다.

변환 속성의 정렬 탭에서 매개 변수로 지정을 선택합니다. 새로 만들기를 클릭하여 매개 변수를 작성합니다.

다음 이미지는 매개 변수 대화 상자를 보여 줍니다.

The image shows a 'Parameters' dialog box with the following fields and controls:

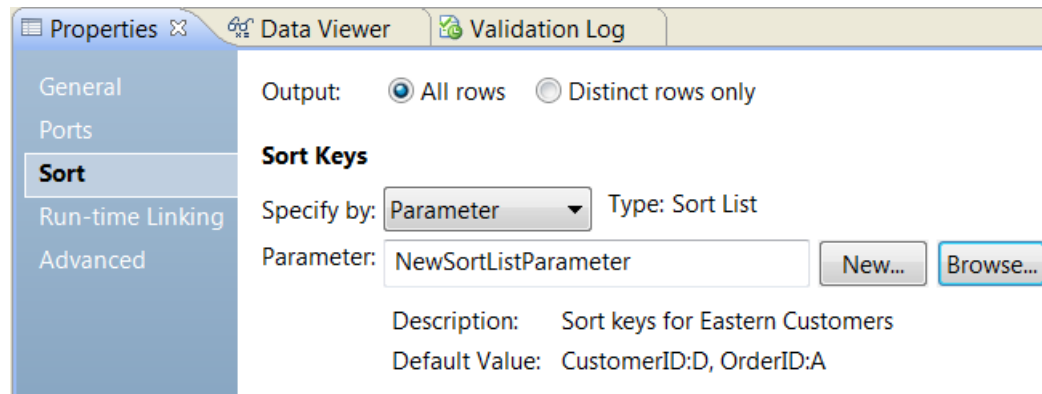
- Name:** NewSortListParameter
- Description:** Sort keys for Eastern Customers
- Type:** Sort List
- Default Value:** A table with two columns: CustomerID (Descending (D)) and OrderID (Ascending (A)).
- Buttons:** Add, Choose..., Delete, Move Up, Move Down, OK, Cancel.

CustomerID	Descending (D)
OrderID	Ascending (A)



정렬 키에 대한 포트 또는 생성된 포트를 선택합니다. 오름차순 또는 내림차순 정렬 유형을 선택할 수 있습니다. 수동으로 포트 이름을 입력할 수 있습니다. **기본값** 필드에 포트 이름을 입력하고 **추가**를 클릭합니다. Developer tool이 포트 이름을 정렬 목록에 추가합니다.

다음 이미지는 정렬 키에 대한 매개 변수를 구성한 후 **정렬** 탭을 보여 줍니다.



## 분류기 변환 고급 속성

분류기 변환 고급 속성에서 추가 정렬 조건을 지정할 수 있습니다. 데이터 통합 서비스는 모든 정렬 키 포트에 속성을 적용합니다. 분류기 변환 속성은 데이터 통합 서비스가 데이터를 정렬할 때 할당하는 시스템 리소스도 결정합니다.

다음 섹션에서는 분류기 변환의 고급 속성에 대해 설명합니다.

### 대/소문자 구분

데이터 통합 서비스가 데이터를 정렬할 때 대/소문자를 고려하는지 여부를 결정합니다. 대/소문자 구분 속성을 활성화하면 데이터 통합 서비스는 대문자를 소문자보다 높게 정렬합니다. Developer tool은 기본적으로 대/소문자 구분을 설정합니다.

### Null을 Low로 처리

null 값을 다른 모든 값보다 낮게 처리합니다. 데이터 통합 서비스가 정렬 작업을 수행할 때 null 값을 다른 모든 값보다 낮게 처리하도록 하려면 속성을 활성화합니다.

### 분류기 캐시 크기

매핑 실행 시작 시 정렬 작업을 수행하기 위해 데이터 통합 서비스에서 할당하는 메모리의 양입니다. 데이터 통합 서비스는 정렬 작업을 수행하기 전에 모든 들어오는 데이터를 분류기 변환으로 전달합니다. "자동"을 선택하면 데이터 통합 서비스가 런타임 시 자동으로 메모리 요구 사항을 계산합니다. 캐시 크기를 조정하려는 경우 특정 값을 바이트 단위로 입력합니다. 기본값은 자동입니다.

### 작업 디렉터리

수신 데이터의 양이 분류기 캐시 크기보다 큰 경우 데이터 통합 서비스가 데이터를 임시로 저장하는 디렉터리입니다. 임시 파일은 데이터가 정렬된 후에 삭제됩니다.

캐시 분할 중에 성능을 향상시키려면 여러 디렉터리를 세미콜론으로 구분하여 입력하십시오. 캐시 분할은 변환을 처리하는 각 파티션에 대해 별도의 캐시를 작성합니다.

기본값은 TempDir 시스템 매개 변수입니다. 다른 시스템 매개 변수를 구성하거나 이 필드에서 사용자 정의 매개 변수를 구성할 수 있습니다.

## 추적 수준

이 변환에 대해 로그에 표시되는 세부 정보의 양입니다. 간단, 보통, 자세한 정보 표시 초기화 또는 자세한 정보 표시 데이터를 선택할 수 있습니다. 기본값은 보통입니다.

## 관련 항목:

- [“캐시 크기” 페이지 71](#)

# 분류기 캐시

데이터 통합 서비스는 분류기 변환을 실행하기 위해 메모리에 캐시를 작성합니다. 데이터 통합 서비스는 정렬 작업을 수행하기 전에 모든 들어오는 데이터를 분류기 변환으로 전달합니다. 데이터 통합 서비스에 메모리 캐시에 사용할 수 있는 공간보다 더 많은 공간이 필요한 경우 데이터 통합 서비스는 데이터를 분류기 변환 작업 디렉터리에 임시로 저장합니다.

메모리의 모든 데이터를 정렬하도록 캐시 크기를 구성하지 않는 경우 데이터 통합 서비스가 소스 데이터에서 여러 패스를 만들었다는 것을 알리는 경고가 세션 로그에 나타납니다. 정렬을 완료하기 위해 정보를 디스크에 페이징해야 하는 경우 데이터 통합 서비스가 데이터에서 여러 패스를 만듭니다. 메시지는 단일 패스에 필요한 메모리 양을 지정하고 이때 데이터 통합 서비스가 데이터를 한 번 읽고 디스크에 페이징하지 않고 메모리에서 정렬을 수행합니다. 매핑 성능을 최적화하려면 데이터 통합 서비스가 데이터에서 하나의 패스를 만들도록 캐시 크기를 구성합니다.

수신 데이터의 양이 분류기 캐시 크기보다 큰 경우 데이터 통합 서비스는 데이터를 분류기 변환 작업 디렉터리에 임시로 저장합니다. 작업 디렉터리에 데이터를 저장할 경우 데이터 통합 서비스에 적어도 수신 데이터 양의 두 배에 이르는 디스크 공간이 필요합니다.

최상의 성능을 얻으려면 매핑을 실행하는 시스템에서 사용 가능한 물리적 메모리의 양과 같거나 작은 값으로 분류기 캐시 크기를 구성합니다. 분류기 변환을 사용하여 데이터를 정렬하려면 최소 16MB(16,777,216바이트)의 물리적 메모리를 할당합니다. 기본적으로 분류기 캐시 크기는 자동으로 설정됩니다.

## 분류기 캐시 최적화

분류기 변환을 통해 전달되는 이진 및 문자열 데이터 유형을 변수 길이를 사용하여 저장하도록 분류기 캐시가 최적화됩니다.

변수 길이를 사용하면 데이터 통합 서비스가 분류기 캐시에 저장하는 데이터의 크기와 데이터 통합 서비스 시스템의 디스크 공간 소비량이 줄어듭니다.

예를 들어 고객에 대한 데이터를 저장합니다. 일부 고객의 이름은 다른 고객의 이름보다 깁니다. 데이터 통합 서비스가 고정된 길이를 사용하여 고객 이름에 대한 데이터를 저장한다면 각 이름에 대해 20자의 데이터를 저장해야 할 수 있습니다. 데이터 통합 서비스가 변수 길이를 사용하면 평균 10자 길이의 데이터를 저장할 수 있습니다.

# 분류기 변환 작성

재사용 가능하거나 재사용 불가능한 분류기 변환을 작성할 수 있습니다.

## 재사용 가능한 분류기 변환 작성

여러 매핑 또는 맵렛에서 사용하기 위한 재사용 가능한 분류기 변환을 작성하십시오.

1. **Object Explorer** 보기에서 프로젝트 또는 폴더를 선택합니다.
2. **파일 > 새로 만들기 > 변환**을 클릭합니다.  
새로 만들기 대화 상자가 나타납니다.
3. 분류기 변환을 선택합니다.
4. **다음**을 클릭합니다.
5. 변환 이름을 입력합니다.
6. **마침**을 클릭합니다.  
변환이 편집기에 표시됩니다.
7. **새로 만들기**를 클릭하여 변환에 포트를 추가합니다.
8. 포트를 편집하여 이름, 데이터 유형 및 전체 자릿수를 설정합니다.
9. **정렬** 탭에서 정렬할 포트를 선택하거나 정렬 목록 매개 변수를 선택합니다.
10. **고급** 보기를 클릭하고 변환 속성을 편집합니다.

## 재사용 불가능 분류기 변환 작성

재사용 불가능 분류기 변환을 매핑 또는 맵렛에 작성합니다.

1. 매핑 또는 맵렛에서 변환 색상표의 분류기 변환을 편집기로 끌어 옵니다.  
변환이 편집기에 표시됩니다.
2. **속성** 보기에서 변환 이름 및 설명을 편집합니다.
3. **포트** 탭에서 **새로 만들기**를 클릭하여 포트를 변환에 추가합니다.
4. 포트를 편집하여 이름, 데이터 유형 및 전체 자릿수를 설정합니다.
5. **키**를 선택하여 포트를 정렬 키로 표시합니다.
6. **고급** 탭을 클릭하고 변환 속성을 편집합니다.

## 분류기 변환 예제

고객의 모든 주문에 대한 정보가 포함된 데이터베이스 테이블 **PRODUCT\_ORDERS**가 있습니다.

ORDER_ID	ITEM_ID	ITEM	QUANTITY	PRICE
43	123456	ItemA	3	3.04
41	456789	ItemB	2	12.02
43	000246	ItemC	6	34.55
45	000468	ItemD	5	0.56
41	123456	ItemA	4	3.04
45	123456	ItemA	5	3.04

ORDER_ID	ITEM_ID	ITEM	QUANTITY	PRICE
45	456789	ItemB	3	12.02

PRODUCT\_ORDERS에 대해 분류기 변환을 사용하고 ORDER\_ID를 정렬 키로, 방향을 내림차순으로 지정합니다.

데이터를 정렬하고 나면 데이터 통합 서비스가 정렬 변환에서 다음 행을 전달합니다.

ORDER_ID	ITEM_ID	ITEM	QUANTITY	PRICE
45	000468	ItemD	5	0.56
45	123456	ItemA	5	3.04
45	456789	ItemB	3	12.02
43	123456	ItemA	3	3.04
43	000246	ItemC	6	34.55
41	456789	ItemB	2	12.02
41	123456	ItemA	4	3.04

각 주문의 총 금액과 항목 수량을 확인해야 합니다. 분류기 변환의 결과를 집계 변환의 입력으로 사용할 수 있습니다. 집계 변환의 정렬된 입력을 사용하여 성능을 개선합니다.

정렬된 입력을 사용하지 않는 경우 데이터 통합 서비스는 읽을 때 집계 계산을 수행합니다. 데이터 통합 서비스는 모든 집계 계산이 정확하도록 전체 소스를 읽을 때까지 각 그룹에 대한 데이터를 저장합니다. 정렬된 입력을 사용하는 경우 데이터를 올바르게 사전 분류하지 않으면 예기치 않은 결과가 발생합니다.

집계 변환에는 포트별 ORDER\_ID 그룹이 있으며 정렬된 입력 옵션이 선택되어 있습니다. 분류기 변환에서 데이터를 전달하면 집계 변환에서 ORDER\_ID를 그룹화하여 각 주문의 총 금액을 계산합니다.

ORDER_ID	SUM
45	54.06
43	216.42
41	36.2

## 분류기 변환 - 비원시 환경

비원시 환경에서 분류기 변환의 처리는 변환을 실행하는 엔진에 따라 다릅니다.

다음과 같은 비원시 런타임 엔진에 대한 지원을 고려하십시오.

- Blaze 엔진. 제한적으로 지원됩니다.
- Spark 엔진. 제한적으로 지원됩니다.
- Databricks Spark 엔진. 제한적으로 지원됩니다.

## 분류기 변환 - Blaze 엔진

Blaze 엔진의 일부 처리 규칙은 데이터 통합 서비스의 처리 규칙과 다릅니다.

### 매핑 유효성 검사

다음과 같은 상황에서는 매핑 유효성 검사가 실패합니다.

- 행 순서를 유지하도록 대상이 구성되어 있고 분류기 변환이 플랫폼 파일 대상에 직접 연결되지 않았습니다.

### 병렬 정렬

데이터 통합 서비스는 다음과 같은 제한과 함께 병렬 정렬을 활성화합니다.

- 매핑의 분류기 변환과 대상 사이에 다른 변환이 포함되지 않습니다.
- 정렬 키의 데이터 유형은 분류기 변환과 대상 간에 변경되지 않습니다.
- 분류기 변환의 각 정렬 키는 대상의 열에 연결되어야 합니다.

### 글로벌 정렬

다음 조건이 참인 경우 **Blaze** 엔진은 글로벌 정렬을 수행할 수 있습니다.

- 분류기 변환이 플랫폼 파일 대상에 직접 연결되어 있습니다.
- 대상이 행 순서를 유지하도록 구성되어 있습니다.
- 정렬 키의 데이터 유형이 이진인 아닙니다.

이러한 조건 중 하나가 참이 아닌 경우 **Blaze** 엔진은 로컬 정렬을 수행합니다.

### 데이터 캐시 최적화

집계 또는 순위 데이터 캐시 최적화를 위해 집계 또는 순위 변환 앞에 분류기 변환이 삽입된 경우 분류기 캐시의 크기는 집계 또는 순위 변환의 데이터 캐시와 동일합니다. 분류기 캐시를 구성하려면 집계 또는 순위 변환에 대한 데이터 캐시의 크기를 구성해야 합니다.

## 분류기 변환 - Spark 엔진

Spark 엔진의 일부 처리 규칙은 데이터 통합 서비스의 처리 규칙과 다릅니다.

### 매핑 유효성 검사

대/소문자 구분이 비활성화되어 있으면 매핑 유효성 검사가 실패합니다.

다음과 같은 상황에서 데이터 통합 서비스는 경고를 기록하고 분류기 변환을 무시합니다.

- 대상과 분류기 변환 정렬 키의 유형이 일치하지 않습니다.
- 대상에 연결되지 않은 정렬 키가 변환에 포함되어 있습니다.
- 쓰기 변환이 행 순서를 유지하도록 구성되지 않았습니다.
- 변환이 쓰기 변환에서 바로 위쪽에 있지 않습니다.

### Null 값

null 값을 높음으로 처리하도록 변환을 구성한 경우에도 데이터 통합 서비스는 null 값을 낮음으로 처리합니다.

### 데이터 캐시 최적화

변수 길이를 사용하여 데이터를 저장하도록 분류기 캐시를 최적화할 수 없습니다.

## 병렬 정렬

데이터 통합 서비스는 다음과 같은 제한과 함께 병렬 정렬을 활성화합니다.

- 매핑의 분류기 변환과 대상 사이에 다른 변환이 포함되지 않습니다.
- 정렬 키의 데이터 유형은 분류기 변환과 대상 간에 변경되지 않습니다.
- 분류기 변환의 각 정렬 키는 대상의 열에 연결되어야 합니다.

## 분류기 변환 - Databricks Spark 엔진

Databricks Spark 엔진의 일부 처리 규칙은 데이터 통합 서비스의 처리 규칙과 다릅니다.

### 매핑 유효성 검사

대/소문자 구분이 비활성화되어 있으면 매핑 유효성 검사가 실패합니다.

다음과 같은 상황에서 데이터 통합 서비스는 경고를 기록하고 분류기 변환을 무시합니다.

- 대상과 분류기 변환 정렬 키의 유형이 일치하지 않습니다.
- 대상에 연결되지 않은 정렬 키가 변환에 포함되어 있습니다.
- 쓰기 변환이 행 순서를 유지하도록 구성되지 않았습니다.
- 변환이 쓰기 변환에서 바로 위쪽에 있지 않습니다.

### Null 값

null 값을 높음으로 처리하도록 변환을 구성한 경우에도 데이터 통합 서비스는 null 값을 낮음으로 처리합니다.

### 데이터 캐시 최적화

변수 길이를 사용하여 데이터를 저장하도록 분류기 캐시를 최적화할 수 없습니다.

## 병렬 정렬

데이터 통합 서비스는 다음과 같은 제한과 함께 병렬 정렬을 활성화합니다.

- 매핑의 분류기 변환과 대상 사이에 다른 변환이 포함되지 않습니다.
- 정렬 키의 데이터 유형은 분류기 변환과 대상 간에 변경되지 않습니다.
- 분류기 변환의 각 정렬 키는 대상의 열에 연결되어야 합니다.

## 제 42 장

# SQL 변환

이 장에 포함된 항목:

- [SQL 변환 개요, 587](#)
- [SQL 변환 포트, 588](#)
- [SQL 변환 고급 속성, 592](#)
- [SQL 변환 쿼리, 593](#)
- [입력 행 대 출력 행의 카디널리티, 595](#)
- [SQL 변환을 통한 필터 최적화, 598](#)
- [SQL 쿼리가 포함된 SQL 변환 예제, 600](#)
- [저장 프로시저, 603](#)
- [SQL 변환 연결, 608](#)
- [수동으로 SQL 변환 작성, 609](#)
- [저장 프로시저에서 SQL 변환 작성, 610](#)

## SQL 변환 개요

SQL 변환에서는 매핑의 SQL 쿼리 미드스트림을 처리합니다. SQL 변환에서 SQL 쿼리를 실행할 수도 있고 데이터베이스에서 저장 프로시저를 실행하도록 SQL 변환을 구성할 수도 있습니다.

쿼리 또는 저장 프로시저의 매개 변수에 입력 포트 값을 전달할 수 있습니다. 변환에서는 데이터베이스의 행을 삽입, 삭제, 업데이트 및 검색할 수 있습니다. SQL DDL 문을 실행하여 테이블을 생성하거나 매핑의 테이블 미드스트림을 삭제할 수 있습니다. SQL 변환은 활성 변환입니다. 변환에서는 각 입력 행에 대해 여러 행을 반환할 수 있습니다.

데이터베이스에서 SQL 변환으로 저장 프로시저를 가져올 수 있습니다. 저장 프로시저를 가져올 때 **Developer tool**은 저장 프로시저의 매개 변수에 해당하는 변환 포트를 생성합니다. **Developer tool**에서는 저장 프로시저 호출도 자동으로 생성합니다.

저장 프로시저를 실행하도록 SQL 변환을 구성하려면 다음 태스크를 수행하십시오.

1. 연결하려는 데이터베이스 유형을 포함하여 변환 속성을 정의합니다.
2. 저장 프로시저를 가져와 포트를 정의하고 저장 프로시저 호출을 생성합니다.
3. 실행해야 하는 추가 저장 프로시저 또는 결과 집합에 대해 포트를 수동으로 정의합니다.
4. SQL 편집기에서 저장 프로시저 호출을 더 추가합니다.

변환 SQL 편집기에서 SQL 쿼리를 구성할 수 있습니다. SQL 변환을 실행하면 해당 변환이 쿼리를 처리하고 행을 반환하며 데이터베이스 오류를 반환합니다.

쿼리를 실행하도록 SQL 변환을 구성하려면 다음 태스크를 수행하십시오.

1. 연결하려는 데이터베이스 유형을 포함하여 변환 속성을 정의합니다.
2. 입력 및 출력 포트를 정의합니다.
3. SQL 편집기에서 SQL 쿼리를 생성합니다.

변환을 구성한 후에는 매핑에서 SQL 변환을 구성하고 업스트림 포트를 연결합니다. 데이터를 미리 보고 결과를 확인합니다.

## SQL 변환 포트

SQL 변환을 생성할 때는 Developer tool에서 기본적으로 **SQLError** 포트를 생성합니다. **포트** 보기에서 입력 포트, 출력 포트 및 통과 포트를 추가합니다.

SQL 변환에는 다음과 같은 유형의 포트가 있습니다.

### 입력

SQL 쿼리에서 사용할 수 있는 소스 데이터를 받습니다.

### 출력

SQL SELECT 쿼리에서 데이터베이스 데이터를 반환합니다.

### 통과

소스 데이터를 변경하지 않고 변환을 통해 전달하는 입력-출력 포트입니다.

### SQLError

데이터베이스에서 SQL 오류를 반환합니다. 오류가 발생하지 않은 경우 **NULL**을 반환합니다.

### NumRowsAffected

입력 행에 대한 **INSERT**, **DELETE** 및 **UPDATE** 쿼리 문의 영향을 받는 총 데이터베이스 행 수를 반환합니다. 출력 행에 업데이트 통계를 포함하도록 선택하면 Developer tool이 이 포트를 작성합니다.

### 반환 값

저장 프로시저에서 반환 값을 받습니다.

## 입력 포트

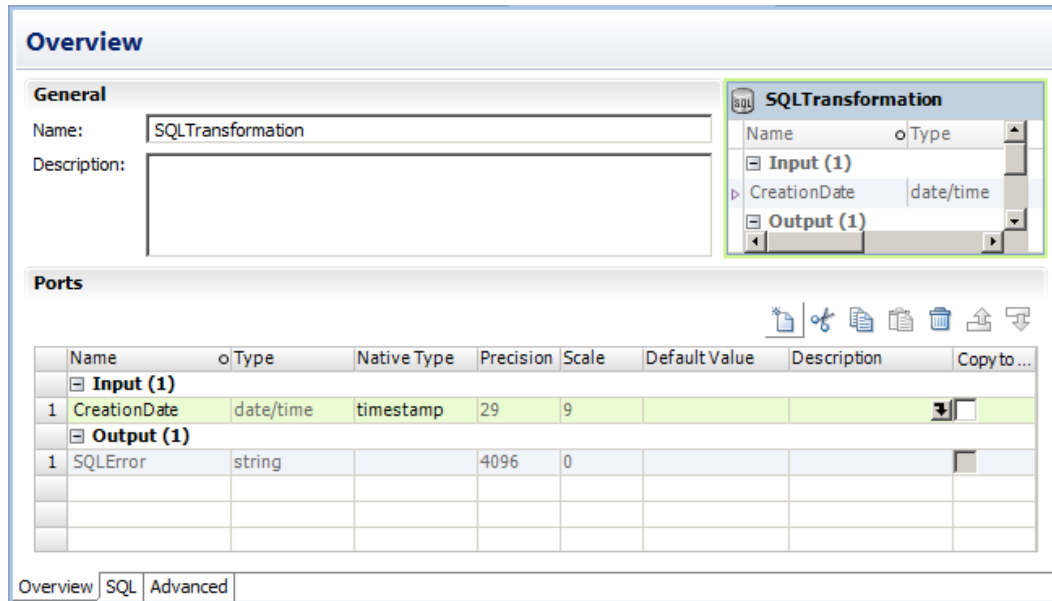
모든 유형의 SQL 문 또는 저장 프로시저에서 매개 변수 바인딩이 포함된 SQL 변환 입력 포트를 참조할 수 있습니다. 출력 포트에 전달하지 않을 데이터의 경우 SQL 변환에서 입력 포트를 작성할 수 있습니다.

입력 매개 변수가 있는 SQL 쿼리를 구성할 경우 수동으로 포트를 추가해야 합니다. 저장 프로시저를 SQL 변환에 가져올 경우 SQL 변환에서 입력 포트를 작성합니다. 데이터를 변경하지 않고 변환에 전달하기 위해 통과 포트를 추가할 수 있습니다.

**개요** 보기에서 포트를 추가할 수 있습니다. 포트를 추가할 경우 포트에 대한 원시 데이터 유형을 입력합니다. 원시 데이터 유형은 사용자가 연결된 데이터베이스에 유효한 데이터 유형입니다. 원시 데이터 유형을 구성한 경우 변환 데이터 유형이 표시됩니다. 행을 SQL 변환으로 끌 경우 Developer tool에서 사용자가 연결된 데이터베이스에 유효한 데이터 유형을 기반으로 원시 데이터 유형을 설정합니다. 쿼리에서 사용자가 사용하고 있는 열의 데이터 유형이 데이터베이스의 열과 동일한 데이터 유형인지 확인하십시오.



다음 그림에서는 재사용 가능한 SQL 변환의 **CreationDate** 입력 포트를 보여 줍니다.



입력 포트를 추가하려면 **포트** 패널에서 **입력**을 클릭합니다. **새로 만들기**를 클릭합니다.

**참고:** 포트에 대해 **출력에 복사**를 선택한 경우 입력 포트가 통과 포트가 됩니다. 통과 포트는 **포트 보기**의 **입력** 및 **출력** 섹션에 표시됩니다.

## 출력 포트

SQL 변환 출력 포트는 쿼리 문이나 저장 프로시저에서 값을 반환합니다.

SQL 변환을 수동으로 구성할 때는 출력 포트를 정의해야 합니다. **SELECT** 문이 반환하는 각 포트나 각 저장 프로시저 출력 매개 변수에 대해 출력 포트를 정의합니다.

저장 프로시저를 가져올 때 **Developer tool**은 프로시저가 반환하는 각 출력 매개 변수에 대해 출력 포트를 생성합니다. 프로시저가 결과 집합을 반환하는 경우에는 결과 집합에서 출력 포트를 수동으로 정의해야 합니다. 저장 프로시저는 결과 집합을 반환할 수 있으며 같은 실행의 결과 집합에 포함되지 않은 출력 매개 변수를 반환할 수 있습니다. 결과 집합 필드와 출력 매개 변수에 대해 출력 포트를 정의해야 합니다.

출력 포트를 구성할 때는 포트의 원시 데이터 유형을 선택합니다. 출력 포트 원시 데이터 유형은 데이터베이스에서 해당 열의 데이터 유형과 일치해야 합니다. 원시 데이터 유형을 구성할 때 **Developer tool**은 포트에 대한 변환 데이터 유형을 정의합니다.

예를 들어 SQL 변환에는 **Oracle** 데이터베이스에 대한 다음 SQL 쿼리가 포함됩니다.

```
SELECT FirstName, LastName, Age FROM EMPLOYEES
```

SQL 변환에서 다음 출력 포트 및 원시 데이터 유형을 구성할 수도 있습니다.

출력 포트	원시 데이터 유형	변환 데이터 유형
FirstNm	varchar2	문자열
LastNm	varchar2	문자열
나이	숫자	배정밀도

출력 포트의 수와 순서는 쿼리 또는 저장 프로시저가 반환하는 열의 수 및 순서와 일치해야 합니다. 출력 포트의 수가 쿼리 또는 저장 프로시저의 열 수보다 많으면 추가 포트는 null 값을 반환합니다. 출력 포트의 수가 SQL의 열 수보다 적으면 데이터 통합 서비스에서 행 오류를 생성합니다.

변환이 연결하는 데이터베이스 유형을 변경하면 Developer tool이 출력 포트의 원시 유형을 변경합니다. Developer tool이 모든 포트에 대해 올바른 데이터 유형을 선택하지 않을 수도 있습니다. 데이터베이스 유형을 변경하는 경우 각 출력 포트의 원시 데이터 유형이 데이터베이스의 열과 같은 데이터 유형인지 확인하십시오. 예를 들어 Developer tool은 Oracle 데이터베이스 열에 대해 nVarchar2를 선택할 수 있습니다. 데이터 유형을 varchar2로 변경해야 할 수 있습니다.

SQL 변환 개요 보기에서 출력 포트를 구성합니다.

## 통과 포트

통과 포트는 데이터를 변경하지 않고 변환을 통해 데이터를 전달하는 입력-출력 포트입니다. SQL 변환은 SQL 쿼리가 행을 반환하는지 여부에 관계없이 통과 포트의 데이터를 반환합니다.

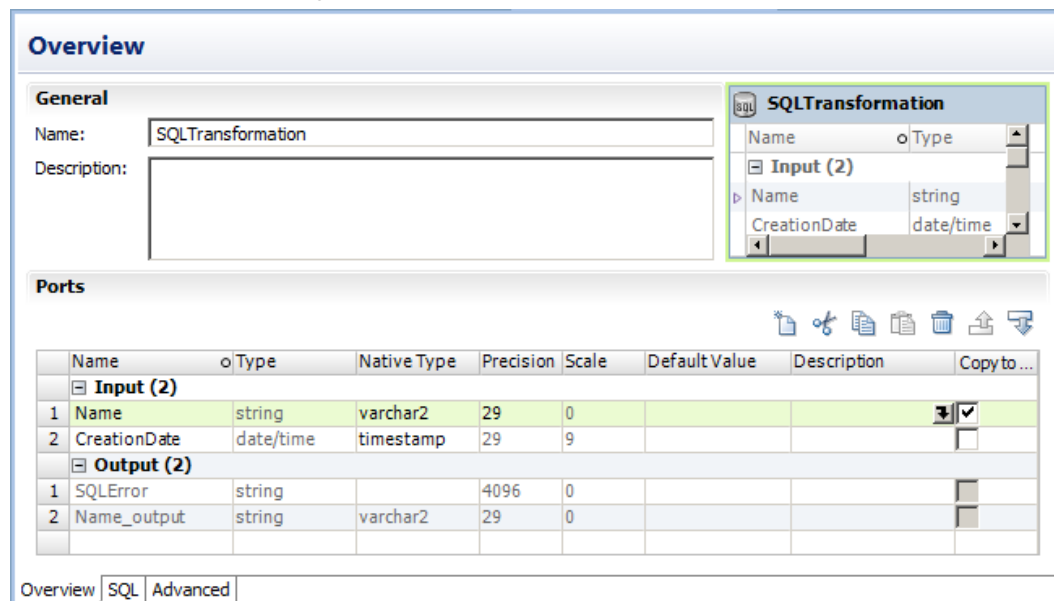
입력 행에 SELECT 쿼리 문이 포함되어 있으면 SQL 변환은 데이터베이스에서 반환하는 각 행에 대해 통과 포트의 데이터를 반환합니다. 쿼리 결과에 행이 여러 개 포함되어 있으면 SQL 변환은 각 행에서 통과 데이터를 반복합니다.

쿼리가 행을 반환하지 않으면 SQL 변환은 출력 열에 null 값이 포함된 통과 열 데이터를 반환합니다. 예를 들어 INSERT, UPDATE 및 DELETE 문이 포함된 쿼리는 행을 반환하지 않습니다. 쿼리에 오류가 있으면 SQL 변환은 출력 포트의 통과 열 데이터, SQLError 메시지 및 null 값을 반환합니다.

SELECT 쿼리에서 데이터를 반환하도록 통과 포트를 구성할 수는 없습니다.

통과 포트를 생성하려면 입력 포트를 생성한 다음 **출력에 복사**를 선택합니다. Developer tool에서는 출력 포트를 생성한 다음 포트 이름에 "\_output" 접미사를 추가합니다. Developer tool이 통과 포트용으로 생성하는 출력 포트는 변경할 수 없습니다. "\_output" 접미사를 사용하여 출력 포트를 생성할 수는 없습니다.

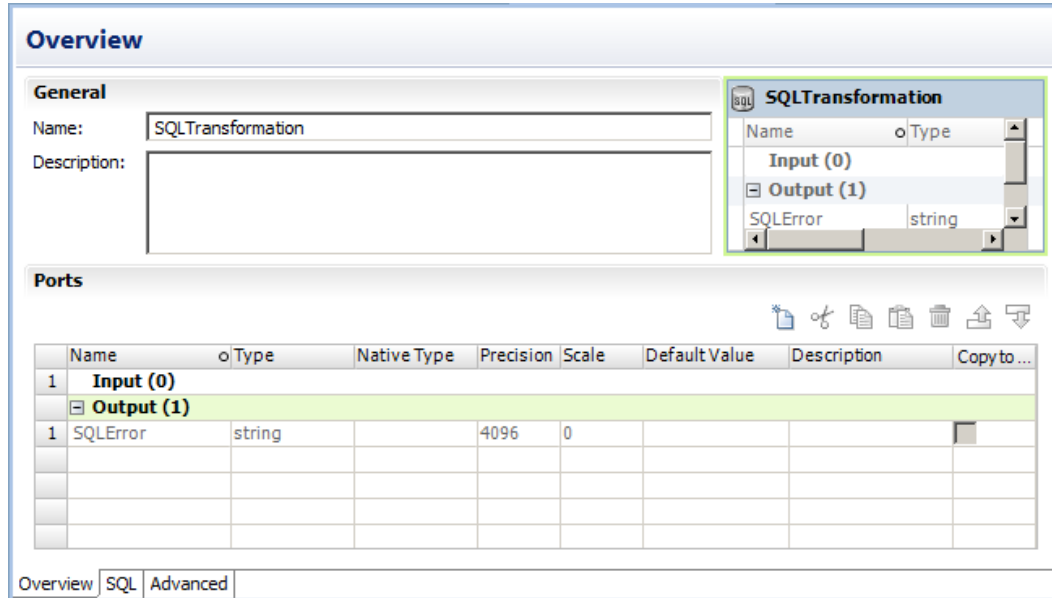
다음 그림에는 재사용 가능 SQL 변환의 Name 통과 포트가 나와 있습니다.



## SQLException 포트

SQLException 포트는 저장 프로시저 또는 SQL 쿼리의 데이터베이스에서 SQL 오류를 반환합니다.

다음 그림에는 재사용 가능 SQL 변환의 SQLException 포트가 나와 있습니다.



SQL 쿼리에 구문 오류가 있으면 SQLException 포트에는 데이터베이스의 오류 텍스트가 포함됩니다. 예를 들어 다음 SQL 쿼리는 Oracle 데이터베이스에서 행 오류를 생성합니다.

```
SELECT Product_ID FROM Employees
```

직원 테이블에는 Product\_ID가 없습니다. 데이터 통합 서비스는 행을 하나 생성합니다. SQLException 포트의 한 행에 오류 텍스트가 포함되어 있습니다.

```
ORA-0094: "Product_ID": invalid identifier Database driver error... Function Name: Execute SQL Stmt:  
SELECT Product_ID from Employees Oracle Fatal Error
```

SQL 쿼리에서 여러 쿼리 문을 구성하거나 여러 저장 프로시저를 호출할 수 있습니다. SQL 오류가 발생해도 계속 진행되도록 SQL 변환을 구성하면 SQL 변환이 특정 쿼리 문에 대해서는 행을 반환하지만 다른 쿼리 문에 대해서는 데이터베이스 오류를 반환할 수 있습니다. SQL 변환은 데이터베이스 오류를 개별 행에 반환합니다.

## 영향을 받는 행 수

각 입력 행에 대해 INSERT, UPDATE 또는 DELETE 쿼리 문이 변경하는 행의 수를 NumRowsAffected 출력 포트가 반환하도록 설정합니다. SQL 쿼리에 대해 NumRowsAffected 출력 포트를 구성할 수 있습니다.

데이터 통합 서비스는 쿼리의 각 문에 대해 NumRowsAffected를 반환합니다. NumRowsAffected는 기본적으로 비활성화됩니다.

SQL 쿼리에 INSERT, UPDATE 또는 DELETE 문이 포함되어 있지 않은 상태에서 NumRowsAffected를 활성화하면 각 출력 행에서 NumRowsAffected가 0이 됩니다.

SQL 쿼리에 여러 문이 포함되어 있으면 데이터 통합 서비스는 각 문에 대해 NumRowsAffected를 반환합니다. NumRowsAffected는 INSERT, UPDATE 및 DELETE 문이 입력 행에 대해 변경하는 행의 합을 포함합니다.

예를 들어 쿼리는 다음 문을 포함합니다.

```
DELETE from Employees WHERE Employee_ID = '101';  
SELECT Employee_ID, LastName from Employees WHERE Employee_ID = '103';  
INSERT into Employees (Employee_ID, LastName, Address)VALUES ('102', 'Gein', '38 Beach Rd')
```

DELETE 문은 행 하나에 적용됩니다. SELECT 문은 행에 적용되지 않습니다. INSERT 문은 행 하나에 적용됩니다.

데이터 통합 서비스는 DELETE 문에서 행 하나를 반환합니다. NumRowsAffected는 1과 같습니다. 데이터 통합 서비스는 SELECT 문에서 행 하나를 반환하며 NumRowsAffected는 0입니다. 데이터 통합 서비스는 INSERT 문에서 행 하나를 반환하며 NumRowsAffected는 1입니다.

## SQL 변환 고급 속성

언제든지 SQL 변환 속성을 변경할 수 있습니다. 기본 데이터베이스 유형은 Oracle입니다. 연결해야 하는 데이터베이스가 다른 데이터베이스 유형이면 변환에 포트를 추가하기 전에 데이터베이스 유형을 변경합니다.

고급 탭에서 다음 속성을 구성합니다.

### 추적 수준

이 변환에 대해 로그에 표시되는 세부 정보의 양입니다. 간단, 보통, 자세한 정보 표시 초기화 또는 자세한 정보 표시 데이터를 선택할 수 있습니다. 기본값은 보통입니다. SQL 변환 추적 수준을 자세한 정보 표시 데이터로 구성하면 데이터 통합 서비스는 준비한 각 SQL 쿼리를 매핑 로그에 씁니다.

### 연결 유형

데이터 통합 서비스가 데이터베이스에 연결하는 방법을 설명합니다. 연결 유형이 정적입니다. 데이터 통합 서비스가 데이터베이스에 한 번 연결합니다. SQL 변환에서 데이터베이스 연결 개체를 선택하십시오. 읽기 전용입니다.

### DB 유형

SQL 변환이 연결되는 데이터베이스의 유형입니다. 목록에서 데이터베이스 유형을 선택하십시오. Oracle, Microsoft SQL Server, IBM DB2 또는 ODBC를 선택할 수 있습니다. 데이터베이스 유형은 포트 탭에서 할당할 수 있는 데이터 유형에 영향을 줍니다. 데이터베이스 유형을 변경하면 Developer tool이 입력, 출력 및 통과 포트의 포트 데이터 유형을 변경합니다.

### 행 내의 오류에 대해 계속

SQL 오류가 발생한 후 쿼리에서 남은 SQL 문 처리를 계속합니다.

### 출력으로 통계 포함

NumRowsAffected 출력 포트를 추가합니다. 포트가 입력 행에 대한 INSERT, DELETE 및 UPDATE 쿼리 문이 업데이트하는 총 데이터베이스 행 수를 반환합니다.

### 최대 출력 행 개수

SQL 변환이 SELECT 쿼리에서 출력할 수 있는 최대 행 수를 정의합니다. 무제한 행 수를 구성하려면 [최대 출력 행 개수]를 0으로 설정하십시오.

### 쿼리 설명

변환에서 정의하는 SQL 쿼리의 설명입니다.

### SQL 모드

SQL 쿼리가 외부 스크립트인지 아니면 쿼리가 변환에서 정의되는지를 결정합니다. SQL 모드가 쿼리입니다. SQL 변환은 SQL 편집기에서 정의하는 쿼리를 실행합니다. 읽기 전용입니다.

### SQL 쿼리

SQL 편집기에서 구성하는 SQL 쿼리를 표시합니다.

## 부작용 있음

SQL 변환이 행을 반환함과 동시에 함수를 수행함을 나타냅니다. SQL 쿼리가 데이터베이스를 업데이트할 때 SQL 변환에서 부작용이 발생합니다. SQL 쿼리에 CREATE, DROP, INSERT, UPDATE, GRANT, REVOKE 등의 문이 포함되어 있으면 **부작용 있음**을 활성화합니다.

변환이 결과가 없는 SELECT 문에 대해 NULL 행을 반환하는 경우에도 SQL 변환에서 부작용이 발생합니다. 행이 통과 포트 값, SQL 오류 정보 또는 NUMRowsAffected 필드를 포함할 수 있습니다.

푸시인 최적화 또는 초기 선택 최적화를 허용하기 위해 **부작용 있음** 속성을 비활성화합니다. 기본값은 활성화됨입니다.

## 데이터베이스 출력만 반환

SQL 변환은 결과 0개를 반환하는 SELECT 문에 대한 행, INSERT/UPDATE/DELETE/COMMIT 등의 기타 문에 대한 행 또는 null 행은 생성하지 않습니다.

## 푸시인 최적화 활성화

데이터 통합 서비스가 매핑의 필터 변환에서 SQL 변환의 SQL로 논리를 푸시하도록 설정합니다.

## 행 순서 유지

변환에 대한 입력 데이터의 행 순서를 유지합니다. 데이터 통합 서비스에서 행 순서를 변경할 수 있는 어떤 최적화도 수행하지 않는 경우 이 옵션을 선택합니다.

데이터 통합 서비스가 최적화를 수행하는 경우 이전에 매핑에 설정된 순서를 잃게 될 수 있습니다. 정렬된 플랫폼 파일 소스, 정렬된 관계형 소스 또는 분류기 변환을 사용하여 매핑에서 순서를 설정할 수 있습니다. 행 순서를 유지하기 위해 변환을 구성하는 경우 데이터 통합 서비스는 매핑에 대해 최적화를 수행할 때 이 구성을 고려합니다. 데이터 통합 서비스는 순서를 유지할 수 있을 때 변환에 대해 최적화를 수행합니다. 데이터 통합 서비스는 최적화가 행 순서를 변경할 수 있는 경우 변환에 대해 최적화를 수행하지 않습니다.

## 분할 가능

여러 스레드를 사용하여 변환을 처리할 수 있습니다. 데이터 통합 서비스가 변환을 처리하는 데 하나의 스레드를 사용하도록 하려면 이 옵션을 선택 취소합니다. 데이터 통합 서비스는 여러 스레드를 사용하여 나머지 매핑 파이프라인 단계를 처리할 수 있습니다.

SQL 쿼리를 사용하려면 변환이 한 개의 스레드로 처리되어야 하는 경우 SQL 변환에 대해 분할을 비활성화합니다. 또는 데이터베이스에 하나만 연결되도록 SQL 변환에 대해 분할을 비활성화하려고 할 수 있습니다.

# SQL 변환 쿼리

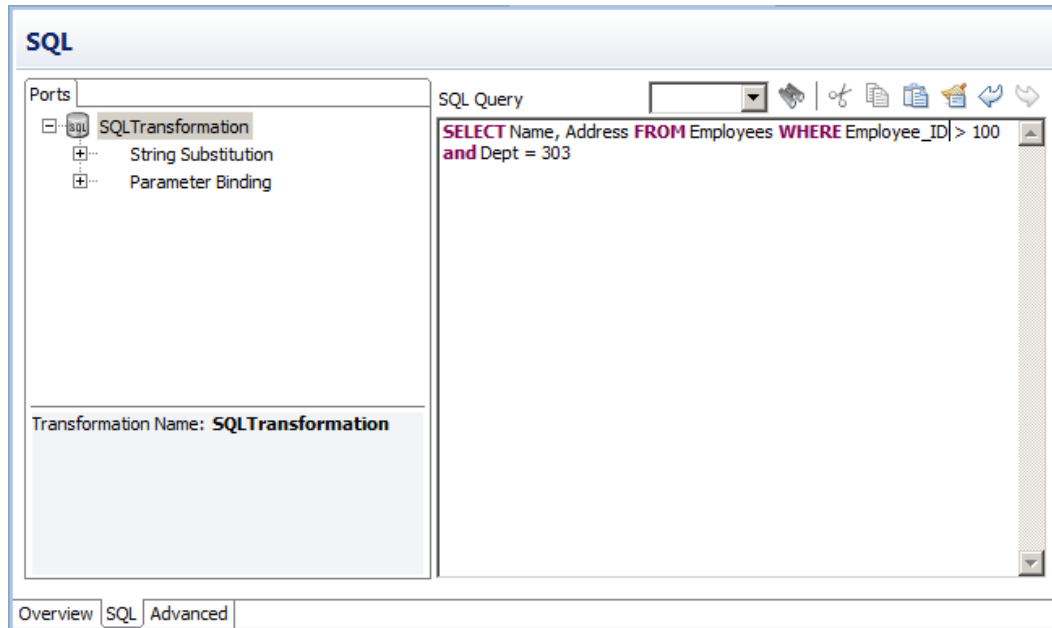
데이터베이스의 행을 검색하거나 데이터베이스를 업데이트하기 위해 SQL 편집기에서 SQL 쿼리를 생성합니다.

쿼리를 생성하려면 SQL 편집기의 SQL 보기에서 쿼리 문을 입력하십시오. SQL 쿼리 문에는 최대 32767자를 입력할 수 있습니다. SQL 편집기는 쿼리에서 참조할 수 있는 변환 포트의 목록을 제공합니다. 포트 이름을 두 번 클릭하여 쿼리 매개 변수로 추가할 수 있습니다.

쿼리를 생성하면 SQL 편집기가 쿼리에서 포트 이름의 유효성을 검사합니다. 또한 문자열 대체에 사용하는 포트가 문자열 데이터 유형인지도 확인합니다. SQL 편집기는 SQL 쿼리의 구문 유효성은 검사하지 않습니다.

SQL 쿼리에서 상수를 사용할 수 있습니다. 각 문자열은 작은따옴표(')로 묶습니다.

다음 그림에는 샘플 SQL 쿼리가 나와 있습니다.



정적 SQL 쿼리를 생성할 수 있습니다. 쿼리 문은 변경되지 않지만 매개 변수를 포함하여 값을 변경할 수 있습니다. 데이터 통합 서비스가 각 입력 행에 대해 쿼리를 실행합니다.

## SQL 쿼리 정의

입력 행마다 동일한 쿼리 문을 실행하는 SQL 쿼리를 정의하십시오. 행의 입력 포트 값을 기반으로 쿼리 열 또는 테이블을 변경할 수 있습니다. 또한 입력 포트 값을 기반으로 WHERE 절의 값을 변경할 수도 있습니다.

입력 행마다 WHERE 절의 데이터 값을 변경하려면 매개 변수 바인딩을 구성하십시오.

입력 포트 값을 기반으로 쿼리 열을 변경하거나 테이블을 변경하려면 문자열 대체를 사용하십시오.

### 매개 변수 바인딩

쿼리의 데이터를 변경하려면 쿼리 매개 변수를 구성하고 변환의 입력 포트에 쿼리 매개 변수를 바인딩합니다. 입력 포트에 매개 변수를 바인딩할 때는 쿼리의 이름을 기준으로 포트를 식별합니다. SQL 편집기에서는 포트 이름을 물음표(?)로 묶습니다. 쿼리 데이터는 포트의 데이터 값에 따라 변경됩니다.

다음 쿼리에서는 매개 변수 바인딩을 사용합니다.

```
DELETE FROM Employee WHERE Dept = ?Dept?
INSERT INTO Employee(Employee_ID, Dept) VALUES (?Employee_ID?, ?Dept?)
UPDATE Employee SET Dept = ?Dept? WHERE Employee_ID > 100
```

다음 SQL 쿼리에는 SQL 변환의 Employee\_ID 및 Dept 입력 포트에 바인딩되는 쿼리 매개 변수가 있습니다.

```
SELECT Name, Address FROM Employees WHERE Employee_Num =?Employee_ID? and Dept = ?Dept?
```

소스에는 다음 행이 포함될 수 있습니다.

Employee_ID	Dept
100	Products
123	HR
130	Accounting

데이터 통합 서비스는 행에서 다음 쿼리 문을 생성합니다.

```
SELECT Name, Address FROM Employees WHERE Employee_ID = '100' and DEPT = 'Products'
SELECT Name, Address FROM Employees WHERE Employee_ID = '123' and DEPT = 'HR'
SELECT Name, Address FROM Employees WHERE Employee_ID = '130' and DEPT = 'Accounting'
```

## 문자열 대체

문자열 변수를 사용하여 쿼리 문의 구성 요소를 바꿀 수 있습니다. 예를 들어 문자열 변수를 사용하여 쿼리에서 테이블 이름을 바꿀 수 있습니다. 또는 **SELECT** 문에서 열 이름을 바꿀 수도 있습니다.

테이블 이름을 대체하려면 각 입력 행에서 테이블 이름을 수신하도록 입력 포트를 구성합니다. **SQL** 편집기에서, 포트의 **문자열 대체** 목록에서 포트를 선택합니다. **Developer tool**이 쿼리에서 이름으로 입력 포트를 식별하고 이름을 물결표(~)로 묶습니다.

다음 쿼리는 문자열 변수, ~Table\_Port~를 포함합니다.

```
SELECT Emp_ID, Address from ~Table_Port~ where Dept = 'HR'
```

소스가 다음 값을 **Table\_Port** 열에 전달할 수 있습니다.

### Table\_Port

Employees\_USA

Employees\_England

Employees\_Australia

데이터 통합 서비스가 ~Table\_Port~ 변수를 입력 포트의 테이블 이름 값으로 바꿉니다.

```
SELECT Emp_ID, Address from Employees_USA where Dept = 'HR'
SELECT Emp_ID, Address from Employees_England where Dept = 'HR'
SELECT Emp_ID, Address from Employees_Australia where Dept = 'HR'
```

# 입력 행 대 출력 행의 카디널리티

데이터 통합 서비스가 **SELECT** 쿼리를 실행하면 **SQL** 변환이 검색하는 각 행에 대해 행을 반환합니다. 쿼리에서 데이터가 검색되지 않을 경우 **SQL** 변환은 각 입력 행에 대해 0 또는 1개의 행을 반환합니다.

## 쿼리 문 처리

**SELECT** 쿼리가 성공할 경우 **SQL** 변환이 여러 행을 검색할 수 있습니다. 쿼리에 다른 문이 포함되는 경우 데이터 통합 서비스가 **SQL** 오류 또는 영향을 받는 행 수를 포함하는 행을 생성할 수 있습니다.

## 포트 구성

**NumRowsAffected** 출력 포트에는 **UPDATE**, **INSERT** 또는 **DELETE** 문이 하나의 입력 행에 대해 변경하는 행의 수가 포함됩니다. **SQL** 변환은 쿼리의 각 문에 대해 영향을 받는 행 수를 반환합니다. **SQL** 변환에 통과 포트가 포함되는 경우에는 각 소스 행에 대해 최소 한 번 이상 열 데이터가 반환됩니다.

## 최대 행 수 구성

최대 출력 행 개수는 **SQL** 변환이 **SELECT** 쿼리에서 반환하는 행 수를 제한합니다.

## 오류 행

데이터 통합 서비스는 연결 오류 또는 구문 오류가 발생할 경우 행 오류를 반환합니다. **SQL** 변환은 오류를 **SQLException** 포트에 반환합니다.

## SQL 오류에 대해 계속

SQL 문에 오류가 있는 경우 처리를 계속하도록 SQL 변환을 구성할 수 있습니다. SQL 변환이 행 오류를 생성하지 않습니다.

## 쿼리 문 처리

SQL 쿼리 유형에 따라 SQL 변환이 반환하는 행수가 결정됩니다. SQL 변환은 0, 1 또는 여러 행을 반환할 수 있습니다. 쿼리에 **SELECT** 문이 포함된 경우 SQL 변환이 데이터베이스의 각 열을 출력 포트로 반환합니다. 변환이 모든 인가된 행을 반환합니다.

다음 테이블에는 쿼리 모드에서 오류가 발생하지 않는 경우 SQL 변환이 다른 유형의 쿼리 문에 대해 생성하는 출력 행이 나열되어 있습니다.

쿼리 문	출력 행
UPDATE, INSERT, DELETE만	쿼리의 각 문에 대해 하나의 행
하나 이상의 SELECT 문	검색한 데이터베이스 총 행 수.
CREATE, DROP, TRUNCATE 등의 DDL 쿼리	쿼리의 각 문에 대해 하나의 행

## 포트 구성

출력으로 통계 포함을 활성화하면 Developer tool이 NumRowsAffected 포트를 작성합니다. 데이터 통합 서비스가 SQL 쿼리 문에 따라 NumRowsAffected를 사용하여 최소 한 개의 행을 반환합니다.

다음 테이블에는 NumRowsAffected를 활성화하는 경우 SQL 변환이 생성하는 출력 행이 나열되어 있습니다.

쿼리 문	출력 행
UPDATE, INSERT, DELETE만	문에 NumRowsAffected를 사용하여 각 문에 대해 한 행씩
하나 이상의 SELECT 문	검색한 데이터베이스 총 행 수. 각 행의 NumRowsAffected는 0입니다.
CREATE, DROP, TRUNCATE 등의 DDL 쿼리	NumRowsAffected가 0인 한 개의 행

## 최대 출력 행 개수

SQL 변환에서 **SELECT** 쿼리에 대해 반환하는 행의 수를 제한할 수 있습니다. 행의 수를 제한하려면 **최대 출력 행 개수** 속성을 구성합니다. 쿼리에 여러 **SELECT** 문이 포함되어 있으면 SQL 변환은 모든 **SELECT** 문의 총 행 개수를 제한합니다.

**최대 출력 행 개수**를 100으로 설정하는 경우를 예로 들어 보겠습니다. 쿼리에는 다음의 두 **SELECT** 문이 포함되어 있습니다.

```
SELECT * FROM table1; SELECT * FROM table2;
```

첫 번째 **SELECT** 문이 행 200개를 반환하고 두 번째 **SELECT** 문이 행 50개를 반환하면 SQL 변환은 첫 번째 **SELECT** 문에서 행 100개를 반환합니다. 두 번째 문에서는 행이 반환되지 않습니다.

무제한 출력 행을 구성하려면 **최대 출력 행 개수**를 0으로 설정하십시오.



## 오류 행

데이터 통합 서비스는 연결 오류 또는 구문 오류가 발생할 경우 행 오류를 반환합니다. SQL 변환은 SQL 오류를 SQLError 포트에 반환합니다.

통과 포트 또는 NumRowsAffected 포트를 구성한 경우 SQL 변환이 각 소스 행에 대해 최소 1개의 행을 반환합니다. 쿼리에서 데이터가 반환되지 않을 경우 SQL 변환은 통과 데이터 및 NumRowsAffected 값을 반환하지만 출력 포트의 Null 값도 반환합니다. 필터 변환을 통해 출력 행을 전달하면 Null 값을 포함하는 행을 제거할 수 있습니다.

다음 표에는 SQL 변환이 UPDATE, INSERT 또는 DELETE 쿼리 문에 대해 생성하는 행이 설명되어 있습니다.

NumRowsAffected 포트 또는 통과 포트 구성	SQLError	행 출력
포트 둘 다 구성되지 않음	아니오	SQLError 포트의 NULL 값을 포함하는 행 1개가 생성됩니다.
포트 둘 다 구성되지 않음	예	SQLError 포트의 오류를 포함하는 행 1개가 생성됩니다.
둘 중 한 포트가 구성됨	아니오	NumRowsAffected 또는 통과 열 데이터를 포함하는 각 쿼리 문에 대한 행 1개가 생성됩니다.
둘 중 한 포트가 구성됨	예	SQLError 포트의 오류, NumRowsAffected 포트 또는 통과 포트 데이터를 포함하는 행 1개가 생성됩니다.

다음 표에는 SQL 변환이 SELECT 문에 대해 생성하는 출력 행의 수가 설명되어 있습니다.

NumRowsAffected 포트 또는 통과 포트 구성	SQLError	행 출력
포트 둘 다 구성되지 않음	아니오	각 SELECT 문에서 반환된 행에 따라 1개 이상의 행이 생성됩니다.
포트 둘 다 구성되지 않음	예	성공한 문에 대한 출력 행의 합계보다 큰 행 1개가 생성됩니다. 마지막 행에는 SQLError 포트의 오류가 포함됩니다.
둘 중 한 포트가 구성됨	아니오	각 SELECT 문에 대해 반환된 행에 따라 1개 이상의 행이 생성됩니다. <ul style="list-style-type: none"> <li>- NumRowsAffected가 활성화된 경우 각 행에 값이 0인 NumRowsAffected 열이 포함됩니다.</li> <li>- 통과 포트가 구성된 경우 각 행에 통과 열 데이터가 포함됩니다. 쿼리에서 여러 행이 반환되는 경우 각 행에 통과 열 데이터가 중복됩니다.</li> </ul>
둘 중 한 포트가 구성됨	예	각 SELECT 문에 대해 반환된 행에 따라 1개 이상의 행이 생성됩니다. 마지막 행에는 SQLError 포트의 오류가 포함됩니다. <ul style="list-style-type: none"> <li>- NumRowsAffected가 활성화된 경우 각 행에 값이 0인 NumRowsAffected 열이 포함됩니다.</li> <li>- 통과 포트가 구성된 경우 각 행에 통과 열 데이터가 포함됩니다. 쿼리에서 여러 행이 반환되는 경우 각 행에 통과 열 데이터가 중복됩니다.</li> </ul>

다음 표에는 SQL 변환이 CREATE, DROP 또는 TRUNCATE와 같은 DDL 쿼리에 대해 생성하는 출력 행의 수가 설명되어 있습니다.

NumRowsAffected 포트 또는 통과 포트 구성	SQLError	행 출력
포트 둘 다 구성되지 않음	아니오	SQLError 포트의 NULL 값을 포함하는 행 1개가 생성됩니다.
포트 둘 다 구성되지 않음	예	SQLError 포트의 오류를 포함하는 행 1개가 생성됩니다.
둘 중 한 포트가 구성됨	아니오	값이 0인 NumRowsAffected 열 및 통과 열 데이터를 포함하는 행 1개가 생성됩니다.
둘 중 한 포트가 구성됨	예	SQLError 포트의 오류, 값이 0인 NumRowsAffected 열 및 통과 열 데이터를 포함하는 행 1개가 생성됩니다.

## SQL 오류에 대해 계속

쿼리 문에서 발생하는 SQL 오류를 무시하도록 선택할 수 있습니다. **행 내의 SQL 오류에 대해 계속**을 활성화합니다. 데이터 통합 서비스가 행의 나머지 SQL 문을 계속해서 실행합니다.

데이터 통합 서비스가 행 오류를 생성하지 않습니다. 그러나 SQLError 포트에는 실패한 SQL 문 및 오류 메시지가 포함됩니다.

예를 들어 쿼리에 다음 문이 포함될 수 있습니다.

```
DELETE FROM Persons WHERE FirstName = 'Ed';
INSERT INTO Persons (LastName, Address) VALUES ('Gein', '38 Beach Rd')
```

DELETE 문이 실패한 경우 SQL 변환이 데이터베이스에서 오류 메시지를 반환합니다. 데이터 통합 서비스는 INSERT 문을 계속해서 처리합니다.

데이터베이스 오류를 해결하고 오류를 야기한 쿼리 문에 오류를 연결하려면 **SQL 오류에 대해 계속** 옵션을 비활성화합니다.

## SQL 변환을 통한 필터 최적화

필터 조건이 통과 포트만 참조하고 SQL 변환에 부작용이 없는 경우 데이터 통합 서비스가 SQL 변환을 통해 필터 최적화를 적용할 수 있습니다.

다음과 같은 상황에서는 SQL 변환에 부작용이 있습니다.

- SQL 쿼리가 데이터베이스를 업데이트합니다. SQL 쿼리에 CREATE, DROP, INSERT, UPDATE, GRANT 또는 REVOKE 문이 포함됩니다.
- 변환이 반환할 결과가 없는 SELECT 문에 대해 NULL을 반환합니다. 행이 통과 포트 값, SQL 오류 정보 또는 NumRowsAffected 필드를 포함할 수 있습니다.

데이터 통합 서비스는 SQL 변환을 통해 초기 선택 및 푸시인 최적화 방법을 적용할 수 있습니다.

## SQL 변환의 초기 선택 최적화

필터 조건이 통과 포트만 참조하고 SQL 변환에 부작용이 없는 경우 데이터 통합 서비스가 SQL 변환에서 초기 선택 최적화를 수행할 수 있습니다.

다음과 같은 상황에서는 SQL 변환에 부작용이 있습니다.

- SQL 쿼리가 데이터베이스를 업데이트합니다. SQL 쿼리에 CREATE, DROP, INSERT, UPDATE, GRANT 또는 REVOKE 문이 포함됩니다.
- 변환이 반환할 결과가 없는 SELECT 문에 대해 NULL을 반환합니다. 행이 통과 포트 값, SQL 오류 정보 또는 NUMRowsAffected 필드를 포함할 수 있습니다.

## SQL 변환의 초기 선택 최적화 활성화

SQL 변환에 부작용이 없는 경우 SQL 변환에서 초기 선택 최적화를 활성화합니다.

1. SQL 변환 고급 속성에서 데이터베이스 출력만 반환 옵션을 활성화합니다.
2. 변환 고급 속성에서 부작용 있음을 선택 취소합니다.
3. 변환에 NumAffectedRows 포트가 있는 경우 해당 포트를 제거합니다.

## SQL 변환의 푸시인 최적화

푸시인 최적화에서는 데이터 통합 서비스가 매핑의 필터 변환에서 SQL 변환의 쿼리로 필터 논리를 푸시합니다.

SQL 변환에서 푸시인 최적화를 활성화할 경우 다음 규칙과 지침을 따르십시오.

- 변환 SQL 쿼리에 SELECT 문만 포함되어야 합니다.
- 변환 SQL 쿼리가 유효한 하위 쿼리여야 합니다.
- 필터 조건이 SQL 오류 또는 NumRowsAffected 필드를 참조할 수 없습니다.
- 출력 포트의 이름과 SQL SELECT 문의 열 이름이 일치해야 합니다. 필터 조건에서 출력 포트를 참조하는 경우 데이터 통합 서비스가 해당 포트 이름을 SQL 쿼리로 푸시합니다. 쿼리의 열이 출력 포트 이름과 일치하지 않는 경우 SQL에 별칭을 추가할 수 있습니다. 예를 들어 SELECT mycolname1 AS portname1, mycolname2 AS portname2와 같이 별칭을 추가합니다.
- 변환에 부작용이 있을 수 없습니다.

## SQL 변환의 푸시인 최적화 예제

SQL 변환이 고객 ID로 주문을 검색합니다. SQL 변환 후에 나타나는 필터 변환은 주문 액수가 1000보다 큰 행만 반환합니다.

데이터 통합 서비스가 다음과 같은 필터를 SQL 변환의 SELECT 문에 푸시합니다.

```
orderAmount > 1000
```

SQL 쿼리의 각 문이 필터를 포함하는 SELECT 문의 개별 하위 쿼리가 됩니다.

다음 쿼리 문은 원래 쿼리 문을 SELECT 문의 하위 쿼리로 보여 줍니다.

```
SELECT <customerID>, <orderAmount>, ... FROM (원래 쿼리 문) ALIAS WHERE <orderAmount> > 1000
```

SQL 쿼리에 여러 개의 문이 있는 경우 각 문이 별도의 하위 쿼리에 포함됩니다. 하위 쿼리는 WHERE 절이 포함된 동일한 구문을 갖습니다.

customerID 및 orderAmount 포트는 SQL 변환에서 출력 포트의 이름입니다. 하위 쿼리는 통과 포트, SQL 오류 또는 SQL 통계 포트를 포함하지 않습니다. SQL 변환에 여러 개의 필터를 푸시하는 경우 WHERE 절에 모든 필터가 포함됩니다.

## SQL 변환의 푸시인 최적화 활성화

SQL 변환 고급 속성 탭에서 속성을 구성하여 푸시인 최적화를 활성화합니다.

1. 부작용 있음을 선택 취소합니다.
2. 데이터베이스 출력만 반환을 활성화합니다.
3. 최대 출력 행 개수를 0으로 설정합니다.
4. 푸시인 최적화를 활성화합니다.

## SQL 쿼리가 포함된 SQL 변환 예제

여러분이 Hypostores Corporation의 인사 부서 소속 개발자라고 가정하겠습니다. Hypostores는 직원 급여 정보를 인사 관련 직원 데이터와 별도의 데이터베이스에서 유지 관리합니다. 인사 부서에서는 여러 지역에 걸쳐 직원 및 급여의 단일 보기를 쿼리해야 합니다.

직원 논리적 데이터 개체에서 직원 데이터 및 급여 데이터의 단일 보기를 표시하는 논리적 데이터 개체 매핑을 생성하려고 합니다.

직원 데이터 소스를 사용하여 논리적 데이터 개체 매핑을 생성합니다. 급여 데이터베이스에서 급여 및 입사일을 검색하기 위한 SQL 변환을 포함합니다.

### 논리적 데이터 개체 매핑

논리적 데이터 개체 매핑에는 다음 개체가 포함됩니다.

#### 직원 테이블

인적 자원 데이터베이스에 있는 직원 데이터의 입력 관계형 테이블입니다.

#### 급여 테이블

직원의 급여 및 고용 날짜가 포함된 급여 명세서 데이터베이스의 테이블입니다. 이 데이터베이스는 Oracle 데이터베이스입니다.

#### SQL 변환

각 직원 행에 대한 고용 날짜 및 급여를 검색하는 변환입니다. 이 변환은 급여 명세서 데이터베이스에 연결하여 데이터베이스의 급여 테이블에 대한 SQL 쿼리를 실행합니다.

#### 논리적 데이터 개체

직원 및 급여 데이터의 결합된 보기가 포함됩니다. 논리적 데이터 개체는 SQL 변환의 출력을 수신합니다.

#### SQLErrors 파일

SQLErrors 파일은 데이터베이스의 모든 SQL 오류가 포함되는 플랫폼 파일입니다. 데이터 통합 서비스는 각 입력 행에 대해 최소 1개 이상의 행을 SQLErrors 파일에 기록합니다. SQL 오류가 발생하지 않은 경우 SQLError 열에는 NULL이 포함됩니다. SQLErrors 파일을 검토하여 오류를 해결합니다.

### 급여 테이블

급여 테이블은 급여 명세서 데이터베이스의 관계형 테이블입니다. 이 테이블에는 급여 명세서 부서에서 유지하는 직원 데이터가 포함됩니다. SQL 변환은 급여 테이블에서 고용 날짜와 직원 급여를 검색합니다.

다음 표에서는 급여 테이블의 일부 행을 보여 줍니다.

Employee_Num	HireDate	급여
10	1997년 5월 3일	232000
11	2001년 9월 11일	444000
12	1989년 10월 17일	656000
13	2007년 8월 13일	332100

## 직원 테이블

소스는 인적 자원 데이터베이스의 직원 테이블입니다.

다음 표에서는 직원 테이블의 샘플 행을 보여 줍니다.

EmpID	LastName	FirstName	DeptId	전화
10	Smith	Martha	FIN	(415) 552-1623
11	Jones	Cynthia	ENG	(415) 552-1744
12	Russell	Cissy	SLS	(415) 552-1656
13	Goyal	Girish	FIN	(415) 552-1656

## SQL 변환

SQL 변환은 급여 명세서 데이터베이스의 급여 테이블에서 직원의 고용 날짜와 급여를 검색합니다. 급여 테이블은 Oracle 데이터베이스에 있습니다.

다음 단계를 사용하여 SQL 변환을 구성합니다.

1. SQL 변환 속성을 구성합니다.
2. 포트를 정의합니다.
3. SQL 쿼리를 작성합니다.
4. SQL 변환에 대한 데이터베이스 연결을 구성합니다.

### SQL 변환 속성 정의

고급 속성 보기에서 SQL 변환 속성을 구성합니다.

다음 속성을 구성합니다.

#### 데이터베이스 유형

데이터베이스 유형은 Oracle입니다. 포트를 정의할 때 Oracle에 해당하는 포트 데이터 유형을 선택할 수 있습니다.

#### 행 내의 오류에 대해 계속

비활성화. 행에서 SQL 오류가 발생할 경우 처리를 중지합니다.

## 출력으로 통계 포함

비활성화. NumRowsAffected 출력 포트를 작성하지 않습니다.

## 포트 정의

직원 소스 테이블에서 각 열에 대한 입력 포트를 정의하십시오. 입력 포트를 열의 통과 포트로 변경하려면 **출력에 복사**를 선택합니다. **출력에 복사**를 선택하면 Developer tool에서 사용자가 복사한 각 포트에 대한 해당하는 출력 포트를 작성합니다.

다음 입력 통과 포트를 작성하십시오.

이름	유형	원시 유형	전체 자릿수	배율	출력에 복사
EmpID	10진수	number(p,2)	4	0	x
LastName	문자열	varchar2	30	0	x
FirstName	문자열	varchar2	20	0	x
DeptID	문자열	varchar2	4	0	x
전화	문자열	varchar2	16	0	x

SQL 변환에는 다음 출력 포트가 있습니다.

이름	유형	원시 유형	전체 자릿수	배율
EmpID	10진수	number(p,s)	4	0
LastName	문자열	varchar2	30	0
FirstName	문자열	varchar2	20	0
DeptID	문자열	varchar2	4	0
전화	문자열	varchar2	16	0
HireDate	날짜/시간	타임스탬프	29	0
급여	10진수	number(p,s)	8	2

**출력에 복사**를 선택하면 Developer tool이 자신이 작성한 각 출력 포트에 "\_output" 접미사를 추가합니다.

채용 일자 및 급여 열에 대한 출력 포트를 수동으로 정의합니다. SQL 변환이 포트의 급여 테이블에서 채용 일자 및 급여 열을 반환합니다.

## SQL 쿼리 정의

급여 테이블에서 각 직원의 채용 일자 및 급여를 선택하는 SQL 쿼리를 작성하십시오.

SQL 변환 SQL 보기에서 쿼리를 정의하십시오.

SQL 편집기에서 다음 쿼리를 입력하십시오.

```
select HIREDATE,SALARY,from Salary where EMPLOYEE_NUM =?EmpID?
```

채용 일자, 급여 및 **Employee\_Num**은 급여 테이블의 열 이름입니다.

?EMPID?는 EmpID 포트 값을 포함하는 매개 변수입니다.

## 데이터베이스 연결 정의

런타임 보기에서 **SQL 변환**이 연결하는 데이터베이스의 데이터베이스 연결 개체를 선택합니다. **Oracle** 데이터베이스 연결 개체를 선택합니다.

## 출력

**SQLException** 포트 및 **EmpID\_output** 포트를 **SQLExceptions** 플랫폼 파일에 연결합니다. **SQL** 오류가 발생하지 않으면 **SQLException** 포트에는 **null** 값이 포함됩니다.

**EmpID** 및 기타 출력 포트를 논리적 데이터 개체에 연결합니다.

**SQL 변환**은 직원 테이블의 데이터를 포함하는 행을 반환하고 급여 테이블의 입사일 및 급여를 포함합니다.

다음 테이블에는 논리적 데이터 개체의 일부 행이 나와 있습니다.

EmpID	LastName	FirstName	DeptID	전화	HireDate	급여
10	Smith	Martha	FIN	(415) 552-1623	19970303 00:00:00	2320.00
11	Jones	Cynthia	ENG	(415) 552-1744	20010911 00:00:00	4440.00
12	Russell	Cissy	SLS	(415) 552-1656	19891017 00:00:00	6560.00
13	Goyal	Girish	FIN	(415) 552-1660	20070813 00:00:00	3210.00

## 저장 프로시저

**SQL 변환**에서 저장 프로시저를 호출할 수 있습니다. 저장 프로시저를 사용하여 관계형 데이터베이스에서 태스크를 자동화할 수 있습니다. 저장 프로시저는 사용자 정의 변수, 조건부 문 및 표준 **SQL** 문이 지원하지 않는 기타 기능을 허용합니다.

**SQL 변환**은 관계형 데이터베이스에 연결하여 저장 프로시저를 실행합니다. **SQL 변환**은 **Oracle**, **IBM DB2**, **Microsoft SQL Server**, **Sybase** 및 **ODBC**에서 저장 프로시저를 호출할 수 있습니다. 저장 프로시저는 데이터베이스에 보관되며 데이터베이스에서 실행됩니다.

**Sybase** 데이터베이스에서 저장 프로시저를 호출하려면 **ODBC** 연결을 생성합니다. **Windows**가 아닌 운영 체제의 **Microsoft SQL Server** 데이터베이스에서 저장 프로시저를 호출하려는 경우에도 **ODBC** 연결을 생성해야 합니다.

저장 프로시저는 **Transact-SQL**, **PL-SQL** 또는 기타 데이터베이스 프로시저 문의 미리 컴파일된 컬렉션입니다. 저장 프로시저 구문은 데이터베이스에 따라 다릅니다.

저장 프로시저를 사용하여 다음 태스크를 완료할 수도 있습니다.

- 대상 데이터베이스에 데이터를 로드하기 전에 해당 데이터베이스의 상태를 확인합니다.
- 데이터베이스에 충분한 공간이 있는지 확인합니다.
- 특수화된 계산을 수행합니다.

- 특정 값을 기준으로 데이터를 검색합니다.
- 인덱스를 삭제한 후 다시 생성합니다.

저장 프로시저를 사용하여 일반적으로는 변환에 포함해야 하는 쿼리나 계산을 수행할 수 있습니다. 예를 들어 판매세 계산을 위한 효율적으로 테스트된 저장 프로시저가 있으면 식 변환에서 같은 계산을 다시 작성하는 대신 저장 프로시저를 사용하여 해당 계산을 수행할 수 있습니다.

저장 프로시저는 입력을 허용한 다음 결과 행 집합을 반환할 수 있습니다. 저장 프로시저는 입력이 필요하지 않으며 출력을 반환하지 않는 DDL 태스크를 실행할 수 있습니다.

둘 이상의 저장 프로시저를 실행하도록 SQL 변환을 구성할 수 있습니다. 구성하는 각 저장 프로시저에 대해 저장 프로시저 매개 변수와 일치하도록 변환 포트를 구성합니다. 각 저장 프로시저는 데이터를 출력 포트로 다시 전달할 수 있습니다.

저장 프로시저를 포함하는 데이터베이스에는 사용자 사용 권한이 있습니다. 데이터베이스에 대해 저장 프로시저를 실행할 사용 권한이 있어야 합니다.

**참고:** 저장 함수는 단일 값을 반환하는 함수라는 점을 제외하면 저장 프로시저와 비슷합니다. SQL 변환은 저장 함수를 실행할 수 있습니다.

## 저장 프로시저용 SQL 변환 포트

SQL 변환 입력 및 출력 포트는 저장 프로시저의 입력 및 출력 매개 변수에 해당합니다.

저장 프로시저를 가져오면 **Developer tool**은 데이터베이스 연결에서 데이터베이스 유형을 확인합니다.

**Developer** 도구는 저장 프로시저의 매개 변수에서 SQL 변환에 입력 및 출력 포트를 생성합니다. **Developer tool**은 저장 프로시저의 매개 변수에서 각 포트의 원시 데이터 유형을 확인합니다.

SQL 변환을 수동으로 구성할 때는 변환에서 입력 및 출력 포트를 구성해야 합니다. 데이터베이스 유형을 구성할 때 SQL 변환은 입력하는 데이터베이스 유형에 따라 각 포트에 대해 원시 데이터 유형을 변경합니다.

SQL 변환과 저장 프로시저 간에 다음 유형의 데이터를 전달할 수 있습니다.

### 입력 및 출력 매개 변수

SQL 변환은 저장 프로시저로 매개 변수를 보내며 입력 및 출력 포트의 저장 프로시저에서 매개 변수를 받습니다.

### 반환 값

저장 프로시저에서 반환 값을 전달하면 **Developer tool**은 반환 값 포트를 생성합니다.

### SQL 오류

SQL 변환은 **SQL\_Error** 포트의 저장 프로시저에서 오류를 반환합니다.

## 입력 및 출력 매개 변수

SQL 변환에서 저장 프로시저를 호출할 경우 호출 문에서 참조하는 각 필드는 입력 또는 출력 포트를 식별합니다. 저장 프로시저를 가져올 경우 **Developer tool**에서 저장 프로시저 호출 문을 생성합니다. 그렇지 않으면 호출 문을 수동으로 구성해야 합니다.

변환의 **SQL** 보기에서 호출 문을 편집할 수 있습니다.

호출 문의 형식은 다음과 같습니다.

```
?RETURN_VALUE? = call <저장 프로시저 이름>(?Field1?, ?Field2?, . . . )
```

포트 이름을 물음표로 묶습니다. 포트 이름이 저장 프로시저의 매개 변수 이름과 일치할 필요는 없습니다. 출력 포트는 **SELECT** 쿼리의 매개 변수와 동일한 순서여야 합니다.



INOUT 매개 변수를 포함하는 저장 프로시저를 사용할 수 있습니다. SQL 변환에서 입력 포트 이름으로 INOUT 매개 변수를 식별합니다. 출력 포트에는 output\_ 접두사가 있습니다. 데이터 통합 서비스에서 입력 포트와 출력 포트를 동일한 매개 변수에 바인딩합니다.

결과 집합을 반환하도록 SQL 변환을 구성할 수 있습니다. 저장 프로시저에서 결과 집합을 반환할 경우 Developer tool에서 결과 집합의 열에 대한 출력 포트를 작성할 수 없습니다. 저장 프로시저를 가져올 경우 수동으로 포트를 입력하고 저장 프로시저 호출을 구성해야 합니다.

## 반환 값

반환 값은 저장 프로시저 상태를 정의하는 코드 또는 텍스트 문자열입니다. 저장 프로시저에 반환 값이 있으면 SQL 변환은 **반환 값** 포트를 포함합니다.

대부분의 데이터베이스는 저장 프로시저를 실행한 후 반환 값을 전달할 수 있습니다. 반환 값은 정수를 포함할 수도 있고 저장 프로시저에서 정의하는 값을 포함할 수도 있습니다. 예를 들어 저장 프로시저가 성공하면 "Success"가 반환될 수 있습니다.

저장 프로시저가 단일 반환 값이 아닌 결과 집합을 반환하면 SQL 변환은 프로시저의 첫 번째 반환 값을 받습니다.

## 저장 프로시저 결과 집합

저장 프로시저에서 결과 집합을 받도록 SQL 변환을 구성할 수 있습니다. 저장 프로시저가 결과 집합의 여러 행을 반환합니다. SQL 변환은 각 행을 매핑에 반환할 수 있습니다.

다음 저장 프로시저는 결과 집합을 반환합니다.

```
CREATE OR REPLACE FUNCTION fetchEMPinfo
(p_State IN VARCHAR2 )
return types.cursortype
AS
my_cursor types.cursortype;
BEGIN
OPEN my_cursor FOR SELECT EMP_ID, NAME, CITY FROM EMP WHERE STATE = p_State ORDER BY EMP_ID;
RETURN my_cursor;
END;
```

저장 프로시저를 가져오면 Developer tool이 다음 구문과 비슷한 저장 프로시저 호출 문을 생성합니다.

**호출** FETCHEMPINFO (?P\_STATE?)

입력 매개 변수는 p\_state입니다. Developer tool은 출력 포트를 자동으로 생성하지 않습니다. 저장 프로시저 매개 변수와 같은 데이터 유형으로 출력 포트를 수동으로 생성해야 합니다.

결과 집합에 EMP\_ID, EMPNAME 및 CITY 열이 포함되는 경우를 예로 들어 보겠습니다. 이러한 열에 대해 출력 포트를 생성합니다.

또한 다음 구문을 사용하여 출력 열로 SQL 호출을 수동으로 업데이트해야 합니다.

(?EMP\_ID?,?EMPNAME?,?CITY?) = **호출** FETCHEMPINFO (?P\_STATE?)

## 다른 데이터베이스가 포함된 결과 집합

데이터베이스 유형에 따라 다른 구문을 사용하여 결과 집합을 반환하도록 저장 프로시저를 구성합니다.

### Oracle

Oracle 저장 함수가 커서를 포함하는 결과를 반환합니다.

```
create or replace function sp_ListEmp return types.cursortype
```

```

as
    l_cursor    types.cursorType;
begin
    open l_cursor for select ename, empno from emp order by ename;
    return l_cursor;
end;

```

또한 Oracle은 커서를 입력 매개 변수로 허용합니다. SQL 변환을 사용하여 커서를 입력 매개 변수로 구성할 수 있습니다.

## Microsoft SQL Server

Microsoft SQL Server 저장 프로시저는 프로시저 본문의 **select** 문이나 테이블로 명시적으로 선언된 반환 유형을 포함하는 결과 집합 저장 프로시저를 반환합니다.

```

Create PROCEDURE InOut(
@inout varchar(100) OUT
)
AS
BEGIN
set @inout = concat(@inout, '__')
select * from mytable;
END

```

## IBM DB2

IBM DB2 저장 프로시저는 개방형 커서를 포함하는 결과 집합을 반환할 수 있습니다. 반환되는 결과 집합의 수는 **RESULT SET** 절에서 선언됩니다. 저장 프로시저는 커서를 열어서 반환합니다. 아래 예제에서는 개방형 커서 2개를 반환합니다.

```

CREATE PROCEDURE TESTMULTIRS
(IN i_cmacct CHARACTER(5))
RESULT SETS 2
LANGUAGE SQL
BEGIN

DECLARE csnum INTEGER;

--Declare serial cursors to consume less resources
--You do not need a rollable cursor.

DECLARE getDeptNo CHAR(50); --Be careful with the estimated length.
DECLARE getDeptName CHAR(200);
DECLARE c1 CURSOR WITH RETURN FOR s1;
SET getDeptNo = 'SELECT DEPTNO FROM DEPT';
SET getDeptName = 'SELECT DEPTNAME FROM DEPT';

PREPARE s1 FROM getDeptNo;
OPEN c1;

END;

```

## Sybase

Sybase 저장 프로시저는 프로시저 본문의 **select** 문이나 테이블로 명시적으로 선언된 반환 유형을 포함하는 결과 집합 저장 프로시저를 반환합니다.

```

CREATE PROCEDURE FETCHEMPINFO
(
    @p_State VARCHAR(5),
    @e_id INT OUTPUT,
    @e_name VARCHAR(50) OUTPUT
)
AS
BEGIN
SET NOCOUNT ON
SELECT EMP_ID, NAME FROM EMP WHERE STATE = @p_State ORDER BY EMP_ID
SET NOCOUNT OFF

```

```

SELECT @e_id AS EMP_ID, @e_name AS NAME
RETURN
END
GO

```

--Configure the following variables to execute the procedure.

```

DECLARE @p_State VARCHAR(5)
DECLARE @EMPID int
DECLARE @EMPNAME varchar(50)

SET @p_State = 'CA'
exec FETCHEMPINFO @p_State, @e_id = @EMPID, @e_name = @EMPNAME
GO

```

## 결과 집합 행

일부 저장 프로시저는 결과 집합 행 외에 출력 매개 변수도 반환합니다. SQL 변환은 마지막 행에서 출력 매개 변수를 반환합니다. 단일 발생 출력 매개 변수는 결과 집합 행에 포함되지 않습니다.

직원 ID를 받아 출력 매개 변수 1에서는 직원 이름을 반환하고 출력 매개 변수 2에서는 부서를 반환하는 저장 프로시저를 기록하는 경우를 예로 들어 보겠습니다. 이 저장 프로시저는 직원이 연간 사용하는 각 병가의 행동도 반환합니다. 행에는 날짜, 시간 및 휴가 이유가 포함됩니다.

결과 집합에는 직원별로 각기 다른 수의 행이 포함됩니다. 결과 집합의 각 행에는 빈 직원 이름 및 부서가 포함됩니다. SQL 변환은 결과 집합 뒤에 직원 이름과 부서를 반환합니다. 직원 이름과 부서는 마지막 행에 표시됩니다.

## 저장 프로시저 예제

SQL 변환으로 데이터를 반환하는 저장 프로시저를 호출할 수 있습니다.

다음 저장 프로시저는 직원 번호를 받으며 직원 번호와 직원 이름이 포함된 행 하나를 반환합니다.

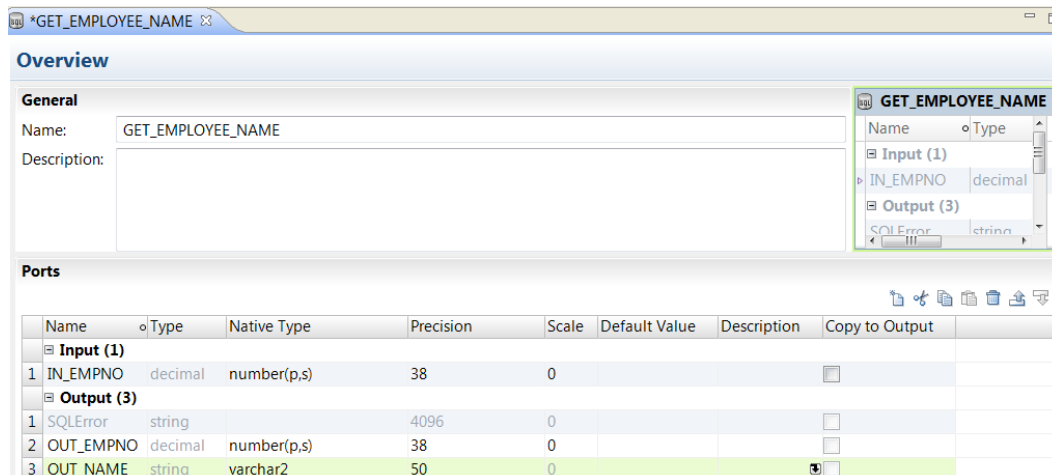
```

CREATE OR REPLACE PROCEDURE SP_GETNAME
(IN_EMPNO IN NUMBER, OUT_EMPNO NUMBER, OUT_NAME OUT STRING)
AS
BEGIN
SELECT EMP_KEY,EMP_NAME into OUT_EMPNO , OUT_NAME from EMP_TABLE where EMP_KEY=IN_EMPNO;
END;/"

```

SQL 변환을 생성하려면 저장 프로시저를 가져옵니다. Developer tool이 입력 포트와 출력 포트를 생성합니다. 포트 이름은 저장 프로시저의 매개 변수 이름과 같습니다.

다음 그림에는 SQL 변환의 포트가 나와 있습니다.



Developer tool에서는 직원 이름을 검색하기 위해 다음 저장 프로시저 호출을 생성합니다.

**호출** SP\_GETNAME (?IN\_EMPNO?,?OUT\_EMPNO?,?OUT\_NAME?)

SQL 편집기에서 저장 프로시저 호출을 확인할 수 있습니다. 모든 SQL 오류는 **SQLError** 포트에 표시됩니다.

## SQL 변환 연결

변환 런타임 속성에서 **SQL 변환 연결**을 구성하십시오. 변환을 작성할 때 연결을 지정하지 않은 경우 런타임 연결을 구성해야 할 수도 있습니다.

SQL 변환을 작성할 때 선택한 연결이 아닌 다른 연결 이름을 정의할 수 있습니다. SQL 변환 **고급** 속성에 있는 데이터베이스 유형과 동일한 데이터베이스 유형의 연결을 선택해야 합니다.

SQL 변환 연결 이름에 대한 매개 변수를 구성할 수 있습니다. 매핑의 **매개 변수** 보기에서 매개 변수를 정의한 후 런타임 연결에 해당 매개 변수를 할당해야 합니다.

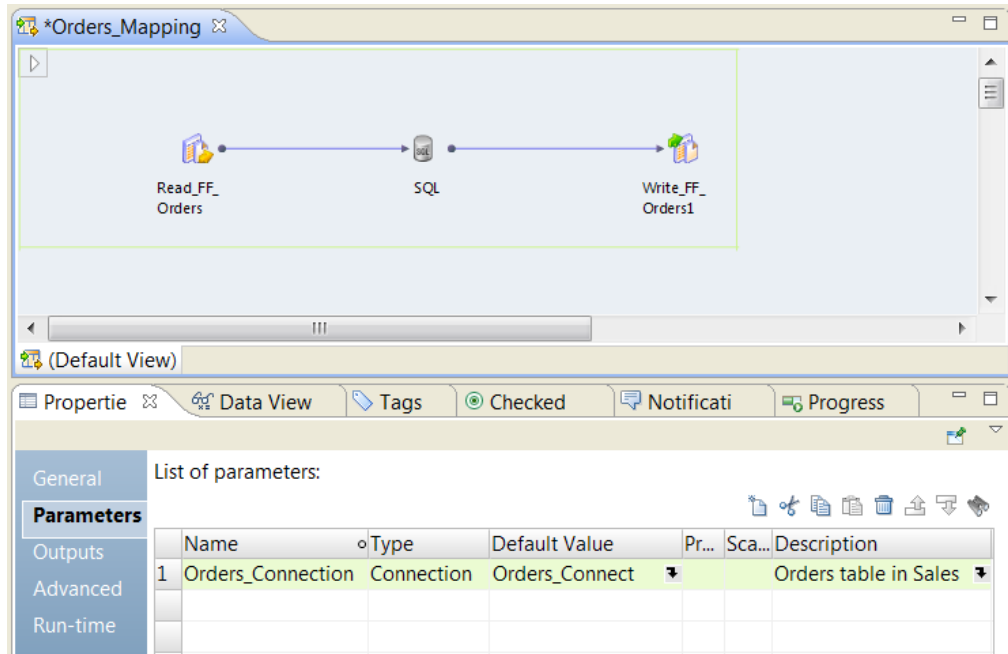
### 연결 이름 매개 변수 작성

SQL 변환에 대한 런타임 연결 이름에 사용자 정의 매개 변수를 지정할 수 있습니다. Developer tool이 변환 매개 변수 대신 연결에 대한 매핑 매개 변수를 작성합니다.

1. SQL 변환을 포함하는 매핑을 작성합니다.
2. SQL 변환 **런타임** 탭을 클릭합니다.
3. **연결 이름**에서 선택 화살표를 클릭하고 **매개 변수 할당**을 선택합니다.
4. **매개 변수 할당** 대화 상자에서 **새로 만들기**를 클릭합니다.
5. **매개 변수** 대화 상자에서 연결 매개 변수 이름 및 매개 변수에 대한 설명을 입력합니다. 매개 변수 유형이 연결에 대한 기본값으로 지정됩니다.
6. **매개 변수 할당** 대화 상자의 매개 변수 대화 상자에서 **확인**을 클릭합니다.

Developer tool이 매핑 매개 변수를 작성하고 연결 이름에 할당합니다.. 런타임 속성에 매개 변수 이름이 나타납니다.

- 매핑 매개 변수 목록을 보려면 편집기 안쪽을 클릭한 다음 **매개 변수** 탭을 클릭합니다.



## 수동으로 SQL 변환 작성

SQL 변환을 수동으로 작성할 수 있습니다. SQL 쿼리를 실행하는 변환을 구성하려는 경우 변환을 수동으로 작성할 수 있습니다. 저장 프로시저를 가져올 수 없는 경우에도 저장 프로시저를 호출하는 변환을 수동으로 작성할 수 있습니다. 변환을 수동으로 작성하는 경우 입력 및 출력 포트를 구성하고 SQL 문을 SQL 편집기에 입력합니다.

- 개체 탐색기 보기에 프로젝트나 폴더를 선택합니다.
- 파일 > 새로 만들기 > 변환을 클릭합니다.  
새로 만들기 대화 상자가 나타납니다.
- SQL 변환을 선택합니다.
- 다음을 클릭합니다.
- 빈 항목으로 작성을 선택합니다.
- 변환의 이름을 입력하고 변환의 리포지토리 위치를 입력합니다.
- 마침을 클릭합니다.
- 개요 보기를 클릭하여 포트를 변환에 추가합니다.
- 입력 포트를 추가하려면 포트 패널에서 **입력**을 클릭하여 포트를 추가할 위치를 표시합니다. **새로 만들기** 단추를 클릭하고 포트 이름, 원시 유형 및 전체 자릿수를 입력합니다.  
기본 데이터베이스 유형은 **Oracle**입니다. 사용자가 **고급** 보기에 데이터베이스 유형을 변경하지 않는 한 **Oracle** 데이터베이스에 대한 원시 유형이 **Developer tool**에 표시됩니다.
- 출력 포트를 추가하려면 포트 패널에서 **출력**을 클릭한 다음 포트를 추가합니다. **새로 만들기** 단추를 클릭하고 포트 이름, 원시 유형 및 전체 자릿수를 입력합니다.  
기본적으로 첫 번째 출력 포트는 **SQLException** 포트입니다.

11. **고급** 보기에서 **SQL 변환**에 연결할 데이터베이스 유형을 선택합니다. 오류 처리에 대한 다른 고급 속성 및 기타 선택적 속성을 구성합니다.  
데이터베이스 유형을 선택하면 **Developer tool**이 **개요** 보기에 표시되는 포트의 원시 데이터 유형을 변경합니다.
12. **SQL** 보기에서 **SQL 쿼리** 또는 저장 프로시저 호출을 입력합니다. **SQL 편집기**에서 매개 변수 바인딩 또는 문자열 대체에 대한 포트를 선택합니다.  
저장 프로시저가 결과 집합을 반환하는 경우 다음 구문과 유사한 구문을 사용하여 저장 프로시저 호출을 입력해야 합니다. (?Field1?,?Field2?,?Field3?) = **호출** Stored\_Procedure\_Name (?Input\_Parm?).

#### 관련 항목:

- [“SQL 쿼리 정의” 페이지 602](#)

## 저장 프로시저에서 SQL 변환 작성

데이터베이스 연결에서 저장 프로시저를 가져와 SQL 변환을 구성할 수 있습니다.

1. **개체 탐색기** 보기에서 프로젝트나 폴더를 선택합니다.
2. **파일 > 새로 만들기 > 변환**을 클릭합니다.  
**새로 만들기** 대화 상자가 나타납니다.
3. **SQL 변환**을 선택합니다.
4. **다음**을 클릭합니다.
5. **기존의 저장 프로시저에서 작성**을 선택합니다.
6. 데이터베이스 연결을 찾아보고 선택합니다.
7. 가져올 저장 프로시저를 찾아보고 선택합니다.
8. 변환의 이름 및 위치를 입력합니다.
9. **마침**을 클릭합니다.  
**Developer tool**이 포트 및 저장 프로시저 호출을 작성합니다.
10. 저장 프로시저가 결과 집합을 반환할 경우 수동으로 출력 포트를 추가한 다음 저장 프로시저 호출을 다시 구성해야 합니다.
  - a. **개요** 보기에서 **포트** 패널의 **출력** 을 클릭합니다. **새로 만들기** 단추를 클릭하고 출력 포트 이름, 원시 유형 및 전체 자릿수를 입력합니다.
  - b. **SQL** 보기에서 다음 구문을 사용하도록 저장 프로시저 호출을 변경합니다. (?Field1?,?Field2?,?Field3?) = **호출** Stored\_Procedure\_Name (?Input\_Parm?)  
**SQL 편집기**의 **매개 변수 바인딩** 포트 목록에서 입력 및 출력 매개 변수를 선택할 수 있습니다.

## 제 43 장

# 표준화 변환

이 장에 포함된 항목:

- [표준화 변환 개요, 611](#)
- [표준화 전략, 611](#)
- [표준화 속성, 612](#)
- [표준화 전략 구성, 613](#)
- [표준화 변환 고급 속성, 613](#)

## 표준화 변환 개요

표준화 변환은 입력 문자열을 검사하고 해당 문자열의 표준화된 버전을 작성하는 수동 변환입니다.

표준화 변환은 표준화된 버전의 입력 문자열이 포함된 열을 작성합니다. 이러한 열을 작성할 때 변환은 입력 데이터의 문자열을 대체하거나 제거할 수 있습니다.

예를 들어 표준화 변환을 사용하여 문자열 **Street**, **St.** 및 **STR**이 포함된 주소 데이터 열을 검사할 수 있습니다. 이러한 문자열의 모든 인스턴스를 문자열 **St**로 바꿀 수 있습니다.

표준화 변환에서 여러 표준화 전략을 작성할 수 있습니다. 각 전략은 여러 표준화 작업을 포함할 수 있습니다. 표준화 변환은 전략을 작성하는 데 사용하는 마법사를 제공합니다.

## 표준화 전략

표준화 전략을 사용하여 표준화된 버전의 입력 문자열이 들어 있는 열을 작성할 수 있습니다.

표준화 전략을 구성하는 경우 하나 이상의 작업을 추가합니다. 각 작업은 특정 표준화 태스크를 구현합니다.

다음 유형의 작업을 표준화 전략에 추가할 수 있습니다.

### 참조 테이블 일치 항목을 올바른 값으로 바꾸기

참조 테이블 값과 일치하는 문자열을 참조 테이블의 "Valid" 값으로 바꿉니다.

### 사용자 지정 문자열로 참조 테이블 일치 항목 바꾸기

참조 테이블 값과 일치하는 문자열을 사용자 정의 대체 문자열로 바꿉니다.

### 참조 테이블 일치 항목 제거

참조 테이블 값과 일치하는 문자열을 제거합니다.

### 사용자 지정 문자열 바꾸기

사용자 정의 문자열을 사용자 정의 대체 문자열로 바꿉니다.

### 사용자 지정 문자열 제거

사용자 정의 문자열을 제거합니다.

**중요:** 작업의 순서를 변경할 수 있습니다. 각 작업은 이전 작업의 결과를 읽기 때문에 작업 순서로 전략의 출력이 변경될 수 있습니다.

## 표준화 속성

표준화 전략 및 작업에 대한 속성을 구성하려면 표준화 변환에서 **전략** 보기를 선택합니다.

### 전략 속성

전략 속성은 전략 내의 모든 작업에 적용됩니다. 다음과 같은 전략 속성을 구성할 수 있습니다.

#### 다중 공백 제거

다중 연속 공백을 하나의 공백으로 바꿉니다.

#### 후행 및 선행 공백 제거

데이터 문자열의 시작과 끝에서 공백을 제거합니다.

#### 구분자

검색 토큰을 정의하는 구분자를 결정합니다. 예를 들어 "세미콜론"을 선택하면 표준화 변환이 문자열 "oranges;apples"를 검색한 다음 문자열 "oranges" 및 "apples"를 찾습니다. 변환은 기본적으로 공백 구분자를 사용합니다.

### 작업 속성

다음 유형의 표준화 작업에 대한 속성을 구성할 수 있습니다.

#### 참조 테이블 작업

참조 테이블 작업은 다음 속성을 포함합니다.

- **참조 테이블.** 데이터를 표준화하기 위해 사용하는 참조 테이블을 결정합니다. **찾아보기**를 클릭하고 참조 테이블을 선택합니다.
- **대/소문자 구분.** 입력 문자열이 참조 테이블 항목의 대/소문자와 일치해야 하는지 여부를 결정합니다.
- **교체할 내용.** 참조 테이블 항목과 일치하는 입력 문자열을 제공하는 텍스트와 바꿉니다. 대체 작업에만 적용합니다.
- **범위.** 참조 테이블 값을 포함하는 입력 문자열 일부를 지정합니다.

#### 사용자 지정 문자열 작업

사용자 지정 문자열 작업은 다음 속성을 포함합니다.

- **다음과 토큰 일치.** 입력 문자 내에서 찾으려는 검색 문자열을 정의합니다.
- **교체할 내용.** 지정하는 검색 문자열과 일치하는 입력 문자열을 바꿉니다. 대체 작업에만 적용합니다.
- **범위.** 검색할 입력 문자열 일부를 지정합니다.



## 표준화 전략 구성

표준화 전략을 구성하려면 표준화 변환의 **전략** 보기에서 설정을 편집합니다.

1. **전략** 보기를 선택합니다.
2. **새로 만들기**를 클릭합니다.  
새 **전략** 마법사가 열립니다.
3. **입력** 필드를 클릭하여 전략에 대한 포트를 선택합니다.
4. 전략 속성을 구성하고 **다음**을 클릭합니다.
5. 작업을 선택하고 **다음**을 클릭합니다.
6. 작업 속성을 구성합니다.
7. 필요한 경우 **다음**을 클릭하여 더 많은 작업을 전략에 추가합니다.
8. 모든 작업을 전략에 추가한 후에 **마침**을 클릭합니다.
9. 필요한 경우 더 많은 전략을 변환에 추가합니다.
10. 필요한 경우 변환이 전략 또는 작업을 처리하는 순서를 변경합니다. 전략 또는 작업을 선택하고 **위로 이동** 또는 **아래로 이동**을 클릭합니다.

## 표준화 변환 고급 속성

데이터 통합 서비스가 표준화 변환에 대해 데이터를 처리하는 방법을 결정할 수 있는 속성을 구성합니다.

로그의 추적 수준을 구성할 수 있습니다.

**고급** 탭에서 다음 속성을 구성합니다.

### 추적 수준

이 변환에 대해 로그에 표시되는 세부 정보의 양입니다. 간단, 보통, 자세한 정보 표시 초기화 또는 자세한 정보 표시 데이터를 선택할 수 있습니다. 기본값은 보통입니다.

## 제 44 장

# 합집합 변환

이 장에 포함된 항목:

- [합집합 변환 개요, 614](#)
- [그룹 및 포트, 615](#)
- [합집합 변환 고급 속성, 615](#)
- [합집합 변환 처리, 616](#)
- [합집합 변환 작성, 616](#)
- [합집합 변환 - 비원시 환경, 617](#)

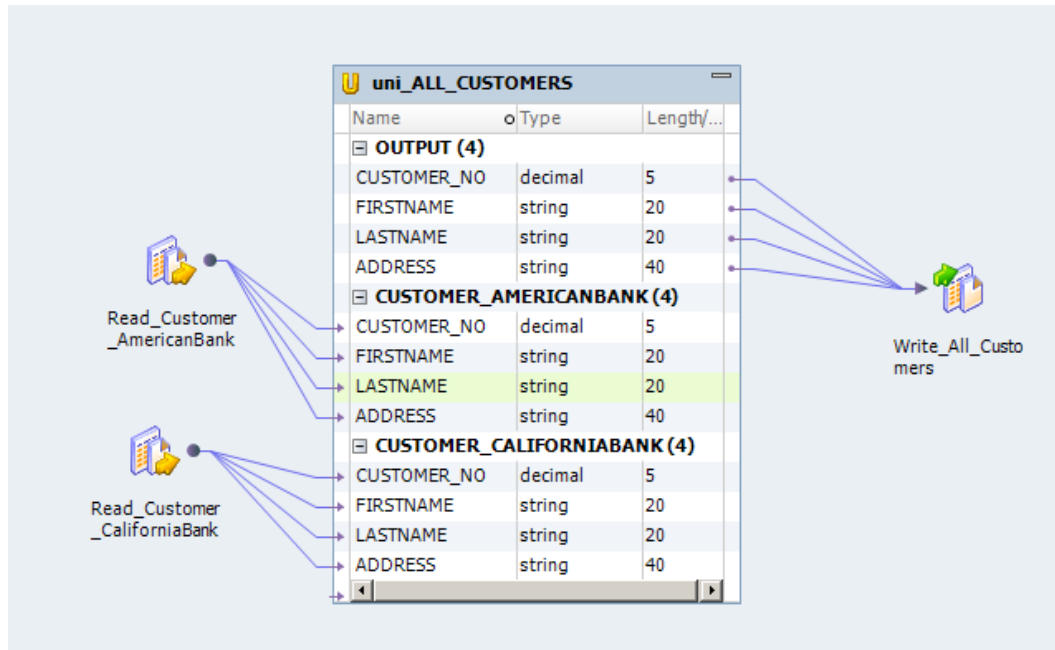
## 합집합 변환 개요

합집합 변환을 사용하여 여러 파이프라인 또는 파이프라인 분기의 데이터를 하나의 파이프라인 분기로 병합할 수 있습니다.

합집합 변환은 여러 입력 그룹과 하나의 출력 그룹을 사용하는 활성 변환입니다. 소스를 일치하는 포트와 병합하고 데이터를 입력 그룹과 동일한 포트 구조를 가진 출력 그룹을 통해 전달합니다. **Developer tool**에서 합집합 변환을 사용하여 중복 행을 제거하지 않고 여러 소스의 데이터를 병합할 수 있습니다.

예를 들어 **American Bank**와 **California Bank**의 고객 계좌 데이터를 결합하려고 합니다. 합집합 변환이 포함된 맵핑을 작성하여 소스 개체의 데이터를 병합하고 대상 개체에 쓸 수 있습니다.

다음 그림은 합집합 변환과의 매핑을 보여줍니다.



## 그룹 및 포트

합집합 변환에는 여러 개의 입력 그룹과 1개의 출력 그룹이 있습니다. 사용자는 하나 이상의 입력 그룹을 작성할 수 있습니다. 하나의 출력 그룹은 **Developer tool**이 작성합니다. 사용자는 출력 그룹을 작성, 편집 또는 삭제할 수 없습니다. 각 그룹에는 일치하는 포트가 있어야 합니다.

포트를 작성하려면 변환의 포트를 복사하거나 수동으로 포트를 작성합니다. 포트를 작성하면 **Developer tool**이 각 입력 그룹의 입력 포트 및 출력 그룹의 출력 포트를 작성합니다. **Developer tool**은 사용자가 각 입력 및 출력 포트에 지정한 출력 포트 이름을 사용합니다. 또한 데이터 유형, 전체 자릿수 및 배열과 같은 각 포트에 대한 동일한 메타데이터를 사용합니다.

단일 파이프라인의 서로 다른 분기 또는 서로 다른 소스 파이프라인의 입력 그룹을 연결할 수 있습니다. 합집합 변환을 매핑에 추가할 때는 모든 입력 그룹의 동일한 포트를 연결해야 합니다. 한 입력 그룹의 포트를 연결하고 다른 입력 그룹에 있는 동일한 포트는 연결하지 않을 경우 데이터 통합 서비스가 연결되지 않은 포트에 **NULL**을 전달합니다.

## 합집합 변환 고급 속성

데이터 통합 서비스가 합집합 변환에 대해 로그 세부 정보를 표시하는 방법을 결정할 수 있는 속성을 구성합니다.

로그의 추적 수준을 구성할 수 있습니다.

**고급** 탭에서 다음 속성을 구성합니다.

## 추적 수준

이 변환에 대해 로그에 표시되는 세부 정보의 양입니다. 간단, 보통, 자세한 정보 표시 초기화 또는 자세한 정보 표시 데이터를 선택할 수 있습니다. 기본값은 보통입니다.

# 합집합 변환 처리

합집합 변환을 사용하여 여러 파이프라인 또는 파이프라인 분기의 데이터를 하나의 파이프라인 분기로 병합할 수 있습니다. 데이터 통합 서비스는 모든 입력 그룹을 병렬로 처리합니다. 합집합 변환에 연결된 소스를 동시에 읽은 다음 데이터 블록을 변환의 입력 그룹으로 전달합니다. 합집합 변환은 데이터 통합 서비스에서 데이터 블록을 받는 순서에 따라 블록을 처리합니다. 합집합 변환은 입력 그룹의 입력 데이터를 차단하지 않습니다.

# 합집합 변환 작성

재사용 가능하거나 재사용 불가능한 합집합 변환을 작성할 수 있습니다.

## 재사용 가능한 합집합 변환 작성

여러 맵핑 또는 맵렛을 사용하려면 재사용 가능한 합집합 변환을 작성하십시오.

1. **개체 탐색기** 보기에서 프로젝트나 폴더를 선택합니다.
2. **파일 > 새로 만들기 > 변환**을 클릭합니다.  
    **새로 만들기** 대화 상자가 나타납니다.
3. 합집합 변환을 선택합니다.
4. **다음**을 클릭합니다.
5. 변환 이름을 입력합니다.
6. **마침**을 클릭합니다.  
    변환이 편집기에 표시됩니다.
7. **추가** 단추를 클릭하여 포트를 변환에 추가합니다.
8. 포트를 편집하여 이름, 데이터 유형 및 전체 자릿수를 설정합니다.
9. **그룹** 보기를 선택합니다.
10. **새로 만들기** 단추를 클릭하여 입력 그룹을 추가합니다.
11. **고급** 보기를 클릭하고 변환 속성을 편집합니다.

## 재사용 불가능 합집합 변환 작성

재사용 불가능 합집합 변환을 맵핑 또는 맵렛에 작성합니다.

1. 맵핑 또는 맵렛에서 변환 색상의 합집합 변환을 편집기로 끌어옵니다.  
    변환이 편집기에 표시됩니다.
2. **일반** 탭에서 변환 이름 및 설명을 편집합니다.
3. 업스트림 변환의 모든 포트를 선택하여 합집합 변환으로 끌어옵니다. 모든 포트가 합집합 변환의 입력 그룹 및 출력 그룹의 포트에 표시됩니다.

4. **속성** 보기의 **그룹** 탭에서 **새로 만들기**를 클릭하여 입력 그룹을 추가합니다.  
기존의 입력 그룹과 유사한 포트를 포함하는 다른 입력 그룹이 표시됩니다.
5. 합집합 변환의 출력 그룹에 있는 포트를 선택하여 매핑의 다운스트림 변환으로 끌어옵니다.

## 합집합 변환 - 비원시 환경

비원시 환경에서 합집합 변환의 처리는 변환을 실행하는 엔진에 따라 다릅니다.

다음과 같은 비원시 런타임 엔진에 대한 지원을 고려하십시오.

- Blaze 엔진. 제한 없이 지원됩니다.
- Spark 엔진. 제한 없이 지원됩니다.
- Databricks Spark 엔진. 제한 없이 지원됩니다.

## 합집합 변환 - Databricks Spark 엔진

## 제 45 장

# 업데이트 전략 변환

이 장에 포함된 항목:

- [업데이트 전략 변환 개요, 618](#)
- [동적 매핑의 업데이트 전략 변환, 619](#)
- [매핑 내 행에 플래그 지정, 619](#)
- [개별 대상의 업데이트 옵션 지정, 620](#)
- [업데이트 전략 변환 - 비원시 환경, 621](#)

## 업데이트 전략 변환 개요

업데이트 전략 변환은 삽입, 업데이트, 삭제 또는 거부를 위해 행에 플래그를 지정하는 활성 변환입니다. 업데이트 전략 변환을 사용하여 적용하는 조건에 따라 대상에 있는 기존 행에 대한 변경 사항을 제어할 수 있습니다.

활성 변환인 업데이트 전략 변환은 통과하는 행 수를 변경할 수 있습니다. 업데이트 전략 변환은 각 행을 테스트하여 특정 조건을 충족하는지 표시한 다음 그에 따라 행에 플래그를 지정합니다. 변환은 삽입, 업데이트 또는 삭제를 위해 플래그를 지정한 행을 다음 변환에 전달합니다. 거부를 위해 플래그가 지정된 행을 다음 변환에 전달하거나 해당 행을 삭제하도록 변환을 구성할 수 있습니다.

예를 들어 매일 주소가 변경된 경우 업데이트 전략 변환을 사용하여 모든 고객 행에 업데이트 플래그를 지정할 수 있습니다. 또는 회사에서 더 이상 근무하지 않는 사람에 대해 모든 직원 행에 거부 플래그를 지정할 수 있습니다.

Spark 엔진에서 매핑을 실행하는 경우 업데이트 전략 변환을 사용하여 관계형 데이터베이스 대상에 결과를 쓸 수 있습니다. 매핑에는 JDBC 연결 문자열이 사용됩니다.

## 업데이트 전략 설정

업데이트 전략을 정의하려면 다음 단계를 완료합니다.

1. 매핑에서 삽입, 업데이트, 삭제 또는 거부를 위해 행에 플래그를 지정하는 방식을 제어하려면 업데이트 전략 변환을 매핑에 추가합니다. 업데이트 전략 변환을 사용하여 서로 다른 데이터베이스 작업의 동일한 대상을 목적으로 하는 행에 플래그를 지정하거나 행을 거부할 수 있습니다.
2. 매핑을 구성할 때 개별 대상에 대해 삽입, 업데이트 및 삭제 옵션을 정의합니다. 대상별로 삽입 또는 삭제를 위해 플래그가 지정된 모든 행에 삽입 및 삭제를 허용하거나 허용하지 않을 수 있습니다. 업데이트를 위해 플래그가 지정된 모든 행에 대해 업데이트를 처리하기 위한 여러 가지 방법을 선택할 수 있습니다.
3. 매핑이 Spark 런타임 엔진에서 실행되는 경우 업데이트 전략 변환 고급 속성에서 Hive 병합 사용을 선택할 수 있습니다. 쿼리에 INSERT, UPDATE 또는 DELETE 문 대신 MERGE 문을 사용하면 일반적으로 처리 효율이 개선됩니다.

## 동적 매핑의 업데이트 전략 변환

동적 매핑에서 업데이트 전략 변환을 사용할 수 있습니다. 변환에서 동적 포트를 구성하고 생성된 포트를 참조할 수 있습니다.

업데이트 전략 변환에서 동적 포트 또는 생성된 포트를 참조할 수 있습니다. 그러나 런타임 시 생성된 포트가 없는 경우 매핑이 실패합니다.

업데이트 전략 식에서 동적 포트를 사용하는 경우 동적 포트에는 한 개의 생성된 포트만 포함될 수 있습니다.

업데이트 전략 식을 매개 변수화할 수 있습니다. 식 유형 매개 변수를 사용하고 전체 식을 기본 매개 변수 값으로 입력합니다.

## 매핑 내 행에 플래그 지정

업데이트 전략 변환을 매핑에 추가하여 개별 행에 삽입, 업데이트, 삭제 또는 거부 플래그를 지정할 수 있습니다.

각 행이 특정 조건을 충족하는지 여부를 테스트하는 업데이트 전략 식을 정의하십시오. 그런 다음 각 행에 숫자 코드를 할당하여 특정 데이터베이스 작업에 대한 플래그를 행에 지정합니다.

다음 표에는 각 데이터베이스 작업에 대한 상수 및 해당하는 숫자 값이 나열되어 있습니다.

작업	상수	숫자 값
삽입	DD_INSERT	0
업데이트	DD_UPDATE	1
삭제	DD_DELETE	2
거부	DD_REJECT	3

데이터 통합 서비스는 다른 모든 값을 삽입으로 처리합니다.

## 업데이트 전략 식

식 편집기에 업데이트 전략 식을 입력합니다.

업데이트 전략 식은 변환 언어에서 IIF 또는 DECODE 함수를 사용하여 각 행을 테스트합니다. 예를 들어 다음 IIF 문은 시작 날짜가 적용 날짜보다 뒤에 오는 경우 거부를 위해 행에 플래그를 지정합니다. 그렇지 않은 경우 문이 업데이트를 위해 행에 플래그를 지정합니다.

```
IIF( ( ENTRY_DATE > APPLY_DATE), DD_REJECT, DD_UPDATE)
```

업데이트 전략 식에서 매개 변수를 구성할 수 있습니다. 식 편집기에서 매개 변수를 작성하거나 매개 변수를 찾습니다.

변환이 동적 매핑에 있는 경우 변환의 생성된 필드가 변경될 수 있습니다. 전체 업데이트 전략 식을 매개 변수화할 수 있습니다. 매개 변수를 사용하여 식을 정의하는 경우 **Developer tool**이 식의 유효성을 검사할 수 없습니다. 식 매개 변수에는 다른 매개 변수가 포함될 수 없습니다.

## 업데이트 전략 변환 고급 속성

데이터 통합 서비스가 업데이트 전략 변환의 데이터를 처리하는 방법을 결정하려면 고급 속성을 구성합니다.

고급 탭에서 업데이트 전략 변환에 대해 다음 고급 속성을 정의할 수 있습니다.

### 거부된 행 전달

업데이트 전략 변환이 거부된 행을 다음 변환으로 전달하는지 아니면 삭제하는지를 결정합니다. 기본적으로 데이터 통합 서비스는 거부된 행을 다음 변환으로 전달합니다. 데이터 통합 서비스가 행을 거부하도록 플래그를 지정하고 거부 파일에 씁니다. 거부된 행 전달을 선택하지 않으면 데이터 통합 서비스는 거부된 행을 삭제하고 매핑 로그 파일에 씁니다.

### Hive 병합 사용

매핑이 **Spark** 엔진에서 실행되는 경우 업데이트 전략 변환에서 **Hive MERGE**를 사용하여 **Hive** 대상의 업데이트를 수행할지 여부를 결정합니다. 쿼리에 **INSERT**, **UPDATE** 또는 **DELETE** 문 대신 **MERGE** 문을 사용하면 처리 효율이 개선됩니다.

다음과 같은 시나리오에서 매핑은 **Hive MERGE** 옵션을 무시하고 데이터 통합 서비스는 **INSERT**, **UPDATE** 및 **DELETE**를 사용하여 작업을 수행합니다.

- 매핑이 **Blaze** 또는 **Hive**에서 실행됩니다.
- 특정 **Hadoop** 배포에서 **Hive** 구현에 의해 **MERGE**가 제한됩니다.

제한이 결과에 영향을 미치는지 여부를 포함한 작업 결과가 매핑 로그에 포함됩니다.

업데이트가 분할 또는 버킷 구성 열에 영향을 미치는 경우 열 업데이트가 생략됩니다.

**참고:** **Developer tool** 및 데이터 통합 서비스는 유효성 검사 시 이 제한을 고려하지 않습니다. 업데이트 전략 식이 이러한 제한을 위반하는 경우 매핑에서 예기치 않은 결과가 나올 수 있습니다.

### 추적 수준

이 변환에 대해 로그에 표시되는 세부 정보의 양입니다. 간단, 보통, 자세한 정보 표시 초기화 또는 자세한 정보 표시 데이터를 선택할 수 있습니다. 기본값은 보통입니다.

## 집계 및 업데이트 전략 변환

집계 및 업데이트 전략 변환을 동일한 파이프라인의 일부로 연결하는 경우 집계 변환을 업데이트 전략 변환 앞에 배치합니다. 이 순서에서 데이터 통합 서비스는 집계 계산을 수행한 다음 이 계산의 결과가 포함된 행에 삽입, 업데이트, 삭제 또는 거부 플래그를 지정합니다.

업데이트 전략 변환을 집계 변환 앞에 배치하는 경우 집계 변환이 다른 작업에 대한 플래그가 지정된 행을 처리하는 방식을 고려해야 합니다. 이 순서에서 데이터 통합 서비스는 행에 삽입, 업데이트, 삭제 또는 거부 플래그를 지정한 다음 집계 계산을 수행합니다. 행에 지정된 플래그에 따라 집계 변환이 계산에 사용된 행의 값을 처리하는 방식이 결정됩니다. 예를 들어 삭제 플래그가 지정된 행을 함께 계산에 사용하는 경우 데이터 통합 서비스는 이 행의 값을 뺍니다. 거부 플래그가 지정된 행을 함께 계산에 사용하는 경우 데이터 통합 서비스는 이 행의 값을 포함하지 않습니다. 삽입 또는 업데이트 플래그가 지정된 행을 함께 계산에 사용하는 경우 데이터 통합 서비스는 이 행의 값을 합계에 더합니다.

## 개별 대상의 업데이트 옵션 지정

업데이트 전략 변환을 사용하여 특정 데이터베이스 작업의 각 행에 플래그를 지정한 후에는 매핑에서 각 대상에 대한 삽입, 업데이트 및 삭제 옵션을 정의합니다. 삽입 또는 삭제 플래그가 지정된 행에 대해 삽입이나 삭제를



용하지 않을 수 있습니다. 업데이트를 위해 플래그가 지정된 모든 행에 대해 업데이트를 처리하기 위한 여러 가지 방법을 선택할 수 있습니다.

매핑의 대상 데이터 개체 고급 속성에서 업데이트 전략 옵션을 정의합니다. 다음 업데이트 전략 옵션을 설정할 수 있습니다.

#### 삽입

삽입 플래그가 지정된 모든 행을 대상에 삽입합니다. 기본값은 활성화됩니다.

#### 삭제

삭제 플래그가 지정된 모든 행을 대상에서 삭제합니다. 기본값은 활성화됩니다.

#### 업데이트 전략

기존 행에 대한 업데이트 전략입니다. 다음 전략 중 하나를 선택합니다.

- **업데이트 시 업데이트.** 업데이트 플래그가 지정된 모든 행을 업데이트합니다. 이것이 기본값입니다.
- **삽입 시 업데이트.** 업데이트 플래그가 지정된 모든 행을 삽입합니다.
- **업데이트 기타 항목 삽입.** 업데이트 플래그가 지정된 모든 행이 대상에 있으면 해당 행을 업데이트한 다음 삽입하도록 표시된 나머지 행을 삽입합니다.

#### 데이터를 잘라내기

데이터를 로드하기 전에 대상을 잘라냅니다. 기본값이 비활성화됩니다.

## 업데이트 전략 변환 - 비원시 환경

비원시 환경에서 업데이트 전략 변환의 처리는 변환을 실행하는 엔진에 따라 다릅니다.

다음과 같은 비원시 런타임 엔진에 대한 지원을 고려하십시오.

- **Blaze 엔진.** 제한적으로 지원됩니다.
- **Spark 엔진.** 제한적으로 지원됩니다.
- **Databricks Spark 엔진.** 지원되지 않습니다.

**참고:** 업데이트 전략 변환은 Hive ACID를 지원하는 Hadoop 배포에서만 지원됩니다.

## 업데이트 전략 변환 - Blaze 엔진

Hive ACID를 지원하는 Hadoop 배포에서 업데이트 전략 변환을 사용할 수 있습니다.

Blaze 엔진의 일부 처리 규칙은 데이터 통합 서비스의 처리 규칙과 다릅니다.

#### 일반 제한

업데이트 전략 변환이 동일한 기본 키 값에 대한 다수의 업데이트 행을 수신하는 경우 변환은 행 하나를 무작위로 선택하여 대상을 업데이트합니다.

다수의 업데이트 전략 변환이 동일한 대상의 여러 인스턴스에 쓰는 경우 대상 데이터를 예측하지 못할 수 있습니다.

Blaze 엔진은 삭제, 업데이트, 삽입의 순서로 작업을 실행합니다. 업데이트 전략 변환이 행을 수신하는 순서와 동일한 순서로 행을 처리하지 않습니다.

Hive 대상은 항상 업데이트를 업데이트 작업으로 수행합니다. Hive 대상은 업데이트 기타 항목 삽입이나 삽입 시 업데이트를 지원하지 않습니다.

### 매핑 유효성 검사 및 컴파일 유효성 검사

다음과 같은 상황에서는 매핑 유효성 검사가 실패합니다.

- 업데이트 전략 변환이 둘 이상의 대상에 연결되어 있습니다.
- 업데이트 전략 변환이 대상의 바로 앞에 위치하지 않습니다.
- 업데이트 전략 대상이 Hive 대상이 아닙니다.
- 업데이트 전략 변환 대상이 외부 ACID 테이블입니다.
- 대상에 기본 키가 없습니다.
- 런타임 시 대상 테이블을 잘라내는 Hive 대상 속성이 활성화되어 있습니다.
- 런타임 시 대상 테이블을 생성하거나 바꾸는 Hive 대상 속성이 활성화되어 있습니다.

다음과 같은 상황에서는 매핑이 실패합니다.

- 대상이 ORC 버킷으로 구성되지 않았습니다.
- Hive 대상이 실제 테이블보다 적은 행을 포함하도록 수정되었습니다.

다음과 같은 상황에서는 컴파일 유효성 검사 오류가 발생하고 매핑 실행이 중지됩니다.

- Hive 버전이 0.14 이전입니다.
- 대상 테이블에 트랜잭션이 활성화되지 않았습니다.

### Hive 대상 테이블 사용

업데이트 전략 변환에서 Hive 대상 테이블을 사용하려면 Hive 데이터 정의 언어의 다음 절이 포함된 Hive 대상 테이블을 생성해야 합니다. TBLPROPERTIES ("transactional"="true").

업데이트 전략 변환에서 Hive 대상을 사용하려면 Hadoop 연결에 연결된 hive-site.xml 구성 집합에 다음 속성이 구성되어 있어야 합니다.

```
hive.support.concurrency      true
hive.enforce.bucketing       true
hive.exec.dynamic.partition   nonstrict
hive.txn.manager              org.apache.hadoop.hive.ql.lockmgr.DbTxnManager
hive.compactor.initiator.on    true
hive.compactor.worker.threads 1
```

## 업데이트 전략 변환 - Spark 엔진

Hive ACID를 지원하는 Hadoop 배포에서 업데이트 전략 변환을 사용할 수 있습니다.

업데이트 전략 변환을 사용하여 매핑 결과를 JDBC 호환 관계형 대상에 쓸 수도 있습니다.

Spark 엔진의 일부 처리 규칙은 데이터 통합 서비스의 처리 규칙과 다릅니다.

### Hive 대상에 대한 일반 제한

대상이 Hive 테이블 또는 JDBC 호환 테이블인 경우 업데이트 전략 변환은 거부된 행을 다음 변환으로 전달하지 않습니다.

업데이트 전략 변환이 동일한 기본 키 값에 대한 다수의 업데이트 행을 수신하는 경우 변환은 행 하나를 무작위로 선택하여 대상을 업데이트합니다.

다수의 업데이트 전략 변환이 동일한 대상의 여러 인스턴스에 쓰는 경우 대상 데이터를 예측하지 못할 수 있습니다.

Spark 엔진에서 매핑을 실행하는 경우 **Hive** 병합 사용 옵션을 선택할 수 있습니다. 이 옵션에는 다음과 같은 제한이 있습니다.

- **Blaze** 또는 **Hive**를 사용하여 매핑을 실행하는 경우 **Hive MERGE** 옵션이 무시됩니다. 데이터 통합 서비스는 이전 구현을 사용하여 변환을 실행합니다.
- 삭제 또는 업데이트에 대한 단일 행을 대상의 여러 행과 일치할 수 없습니다. 매핑이 이 제한을 위반하는 경우 런타임 오류로 인해 매핑이 실패합니다.
- 분할 또는 버킷 구성 열을 업데이트하는 업데이트 전략 식을 구성하는 경우 매핑이 **Hive MERGE** 옵션을 무시합니다. 또한 매핑이 열을 업데이트하지 않습니다.

**참고:** Developer tool 및 데이터 통합 서비스는 유효성 검사 시 이러한 제한을 고려하지 않습니다. 식 또는 매핑이 이러한 제한을 위반하는 경우 매핑이 실행될 수 있지만 예상과 다른 결과가 발생합니다.

**Hive** 대상은 항상 업데이트를 업데이트 작업으로 수행합니다. **Hive** 대상은 업데이트 기타 항목 삽입이나 삽입 시 업데이트를 지원하지 않습니다.

### 매핑 유효성 검사

다음과 같은 상황에서는 매핑 유효성 검사가 실패합니다.

- 업데이트 전략 변환이 둘 이상의 대상에 연결되어 있습니다.
- 업데이트 전략 변환이 대상의 바로 앞에 위치하지 않습니다.
- 업데이트 전략 변환 대상이 외부 **ACID** 테이블입니다.
- 대상에 연결된 기본 키가 없습니다.
- 런타임 시 **Hive** 또는 관계형 대상 테이블이 잘라내기를 활성화하는 속성이 선택되어 있습니다.
- 런타임 시 **Hive** 또는 관계형 대상 테이블에 대한 다음과 같은 대상 전략 중 하나가 선택되어 있습니다.
  - 대상 테이블 생성 또는 바꾸기
  - ApplyNewColumns
  - ApplyNewSchema
  - 실패

대상이 **Hive** 대상인 경우 다음과 같은 상황에서 매핑이 실패합니다.

- 대상 테이블에 트랜잭션이 활성화되지 않았습니까.
- 대상이 **ORC** 버킷으로 구성되지 않았습니까.

### Hive 대상 테이블 사용

업데이트 전략 변환에서 **Hive** 대상 테이블을 사용하려면 **Hive** 데이터 정의 언어의 다음 절이 포함된 **Hive** 대상 테이블을 생성해야 합니다. **TBLPROPERTIES ("transactional"="true")**.

업데이트 전략 변환에서 **Hive** 대상을 사용하려면 **Hadoop** 연결에 연결된 **hive-site.xml** 구성 집합에 다음 속성이 구성되어 있어야 합니다.

```
hive.support.concurrency      true
hive.enforce.bucketing       true
hive.exec.dynamic.partition.mode nonstrict
hive.txn.manager              org.apache.hadoop.hive.ql.lockmgr.DbTxnManager
hive.compactor.initiator.on   true
hive.compactor.worker.threads 1
```

### JDBC 대상 테이블 사용

Spark 엔진에서 실행할 매핑의 업데이트 전략 변환에 대해 **JDBC** 호환 관계형 데이터베이스 대상을 구성할 수 있습니다.

## 제 46 장

# 웹 서비스 소비자 변환

이 장에 포함된 항목:

- [웹 서비스 소비자 변환 개요, 624](#)
- [WSDL 선택, 626](#)
- [웹 서비스 소비자 변환 포트, 627](#)
- [웹 서비스 소비자 변환 입력 매핑, 628](#)
- [웹 서비스 소비자 변환 출력 매핑, 631](#)
- [웹 서비스 소비자 변환 고급 속성, 634](#)
- [필터 최적화, 638](#)
- [웹 서비스 소비자 변환 작성, 640](#)
- [웹 서비스 소비자 변환 예제, 642](#)

## 웹 서비스 소비자 변환 개요

웹 서비스 소비자 변환은 웹 서비스 클라이언트로 웹 서비스에 연결하여 데이터를 액세스하거나 변환합니다. 웹 서비스 소비자 변환은 다중 그룹 변환입니다.

웹 서비스는 **SOAP**, **WSDL** 및 **XML** 같은 공개된 표준을 사용합니다. **SOAP**는 웹 서비스의 통신 프로토콜입니다. 웹 서비스 클라이언트 요청 및 웹 서비스 응답은 **SOAP** 메시지입니다. **WSDL**은 웹 서비스 작업의 프로토콜, 형식 및 서명을 설명하는 **XML** 스키마입니다.

웹 서비스 작업에는 정보 요청, 데이터 업데이트 요청 또는 태스크 수행 요청이 포함됩니다. 예를 들어, 웹 서비스 소비자 변환에서 **getCustomerOrders**라는 웹 서비스 작업을 실행하는 **SOAP** 요청을 전송합니다. 변환에서 요청을 통해 고객 **ID**를 전달합니다. 웹 서비스 클라이언트가 고객 및 주문 정보를 검색합니다. 웹 서비스가 **SOAP** 응답으로 변환에 정보를 반환합니다.

웹 서비스 소비자 변환에서 **WSDL**, 웹 서비스 연결 또는 끝점 **URL** 입력 포트에 정의된 끝점 **URL**을 사용하여 웹 서비스에 연결합니다. 웹 서비스 연결에서 웹 서비스에 대한 보안을 활성화합니다.

## SOAP 메시지

웹 서비스 소비자 변환에서는 **SOAP(Simple Object Access Protocol)**를 사용하여 웹 서비스 공급자와 정보를 교환하고 웹 서비스를 요청합니다. **SOAP**는 웹 서비스 요청 및 응답 메시지에 대한 형식을 정의합니다.

웹 서비스 소비자 변환을 사용하여 데이터를 변환할 경우 변환 시 **SOAP** 요청이 생성되고 변환이 웹 서비스에 연결됩니다. **WSDL** 개체, 웹 서비스 연결 또는 엔드포인트 **URL** 입력 포트에 정의된 엔드포인트 **URL**을 사용하여 변환이 웹 서비스에 연결합니다. **SOAP** 요청에는 요청된 작업을 실행하기 위해 웹 서비스에 필요한 정보가 포함되

어 있습니다. 웹 서비스 작업에서 SOAP 응답의 변환에 데이터를 반환합니다. 변환에서 SOAP 응답의 데이터를 매핑하고 출력 포트에 데이터를 반환합니다.

웹 서비스 소비자 변환은 ISO-8859-1 형식으로 SOAP 메시지 헤더를 인코딩합니다.

변환에서 문서/리터럴 인코딩이 포함된 SOAP 메시지를 처리할 수 있습니다. 문서/리터럴 스타일에는 SOAP 메시지를 설명하기 위한 XML 스키마가 필요합니다. SOAP 메시지는 XML로 구성됩니다. SOAP 메시지에 다중 발생 요소가 포함된 경우 요소 그룹이 XML 계층의 수준을 구성합니다. 한 수준이 다른 수준 안에 중첩될 경우 그룹이 연결됩니다.

SOAP 요청 메시지에 계층 데이터가 포함될 수 있습니다. 예를 들어, 웹 서비스 소비자 변환에서 고객 주문을 영업 데이터베이스에 추가하는 요청을 전송합니다. 변환에서 2개의 데이터 그룹을 SOAP 요청 메시지를 통해 전달합니다. 한 그룹에는 고객 ID와 이름이 포함되어 있고, 다른 그룹에는 주문 정보가 포함되어 있습니다. 주문 정보가 여러 번 발생합니다.

SOAP 응답 메시지에 계층 데이터가 포함될 수 있습니다. 예를 들어, 웹 서비스 소비자 변환에서 고객 주문에 대한 SOAP 요청을 생성합니다. 웹 서비스에서 주문 헤더 및 다중 발생 주문 세부 정보 요소를 SOAP 응답을 통해 반환합니다.

## WSDL 파일

WSDL 파일에는 웹 서비스에 전달되는 데이터에 대한 설명이 포함되므로 보내는 사람과 받는 사람이 교환되는 데이터를 알 수 있습니다. 웹 서비스 소비자 변환을 작성하려면 먼저 WSDL 파일을 리포지토리에 가져와야 합니다.

WSDL에서 데이터에서 수행할 작업과 프로토콜 바인딩 또는 전송을 설명하므로 웹 서비스 소비자가 요청 메시지를 올바른 형식으로 전송할 수 있습니다. WSDL에서 웹 서비스에 연결하는 네트워크 주소를 설명합니다.

WSDL에는 SOAP 요청 및 응답 메시지를 인코딩하는 방법에 대한 정보가 포함되어 있습니다. SOAP 인코딩에서 SOAP 메시지 본문의 형식을 결정합니다. 이는 웹 서비스 소비자와 통신하기 위해 웹 서비스에서 사용하는 요청 및 응답 메시지의 형식을 설명합니다. 웹 서비스 개발자는 다양한 토크를 사용하여 웹 서비스를 작성할 수 있습니다. 토크에서 SOAP 메시지를 인코딩하는 다양한 방법을 지원합니다.

웹 서비스 소비자 변환에서는 문서/리터럴 SOAP 인코딩 스타일을 지원합니다. 웹 서비스 소비자 변환과 함께 WSDL 1.1을 사용할 수 있습니다. MIME, DIME 및 MTOM 메시지 같은 WSDL 첨부 파일은 사용할 수 없습니다.

## 작업

웹 서비스에는 웹 서비스가 지원하는 각 작업에 대한 작업이 포함됩니다.

예를 들어 웹 서비스에 이름이 `getcustomerid`인 작업이 포함될 수 있습니다. 이 작업은 고객 세부 정보를 사용하여 고객 이름과 응답을 검색합니다. 작업 입력에는 고객 이름에 대한 요소가 포함됩니다. 작업 출력에는 고객 이름에 따라 고객 세부 정보에 대한 요소가 포함됩니다.

웹 서비스 소비자 변환을 구성할 때는 변환이 데이터를 작업 입력에 매핑하는 방법과 변환이 작업 출력의 데이터를 매핑하는 방법을 정의합니다. 변환에서 다음 정보를 구성해야 합니다.

### 입력 매핑

변환 입력 포트를 웹 서비스 작업 입력 노드에 매핑하는 방법을 정의합니다. 작업 입력은 작업에 대한 SOAP 요청의 요소를 정의합니다.

### 출력 매핑

웹 서비스 작업 출력 노드를 변환 출력 포트에 매핑하는 방법을 정의합니다. 작업 출력은 작업에 대한 SOAP 응답의 요소를 정의합니다.

## 웹 서비스 보안

웹 서비스 연결에서 웹 서비스에 대한 보안을 활성화합니다. 다음과 같은 유형의 보안을 구성할 수 있습니다.

### 웹 서비스 보안

데이터 통합 서비스가 SOAP 요청을 웹 서비스 공급자에게 전송할 경우 웹 서비스 보안 헤더를 포함할 수 있습니다. 웹 서비스 보안 헤더에는 인증 정보가 포함되므로 웹 서비스 공급자가 데이터 통합 서비스를 인증할 수 있습니다.

웹 서비스 소비자 변환에서 사용자 이름 토큰을 제공합니다. 데이터 통합 서비스가 SOAP 요청에 별도의 보안 SOAP 헤더를 작성하고 요청을 웹 서비스 공급자에게 전달합니다.

웹 서비스 연결에서 다음 유형의 웹 서비스 보안을 사용할 수 있습니다.

- **PasswordText.** 데이터 통합 서비스가 WS-Security SOAP 헤더의 암호를 변경하지 않습니다.
- **PasswordDigest.** 데이터 통합 서비스가 암호에 임시 값과 타임스탬프를 결합시킵니다. 데이터 통합 서비스가 암호에 SHA 해시를 적용하고 base64 인코딩 형식으로 인코딩한 후 SOAP 헤더에서 해당 인코딩된 암호를 사용합니다.

### TLS(Transport Layer Security)

SSL(Secure Sockets Layer)을 사용하여 TCP/IP의 전송 계층(TCP 계층) 위에 구현된 보안입니다. 웹 서비스가 보안 메시지 전송을 위해 HTTPS(Hypertext Transfer Protocol over SSL)를 웹 주소로 사용합니다. 웹 서비스 소비자 변환은 TLS 1.2, TLS 1.1 또는 TLS 1.0을 사용할 수 있습니다. 전송 계층 보안과 함께 HTTP 인증, 프록시 서버 인증 및 SSL 인증서를 사용할 수 있습니다.

### SSL 인증

HTTPS 프로토콜을 통해 연결할 때 SSL 인증을 사용할 수 있습니다.

다음 유형의 SSL 인증을 사용할 수 있습니다.

- 단방향 SSL 인증
- 양방향 SSL 인증

### HTTP 인증

HTTP 프로토콜을 통해 연결할 때 HTTP 인증을 사용할 수 있습니다.

다음 HTTP 인증 방법을 사용할 수 있습니다.

- 기본 인증
- 다이제스트 인증
- NTLM(NT LAN Manager) 인증

## WSDL 선택

웹 서비스 소비자 변환을 작성하기 전에 WSDL 파일을 모델 리포지토리에 가져와야 합니다. WSDL에서 사용자가 실행하려는 웹 서비스의 작업 서명을 정의합니다. WSDL을 가져올 경우 Developer tool에서 다른 변환에 재사용할 수 있는 실제 데이터 개체를 작성합니다.

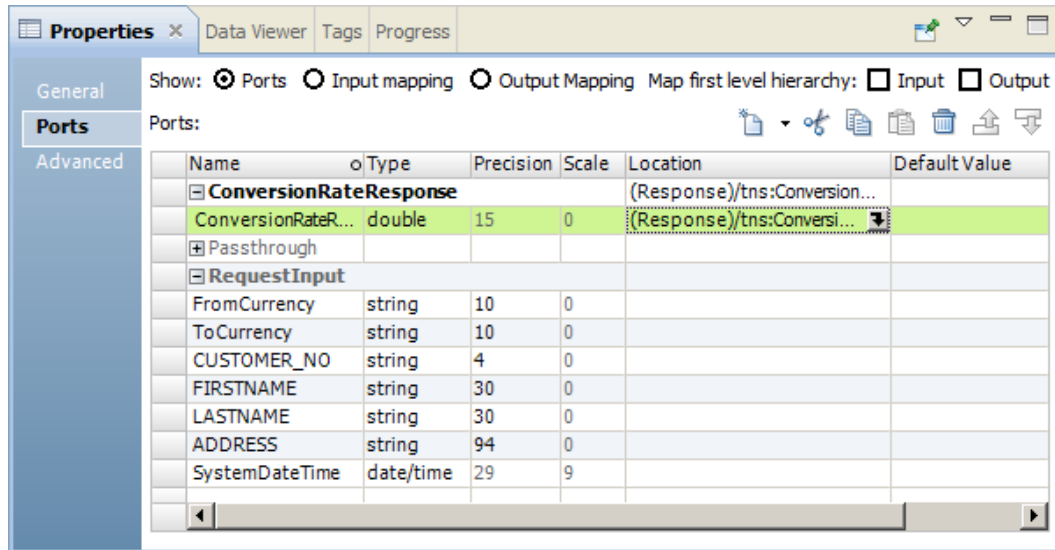
WSDL에서 여러 작업을 정의할 수 있습니다. 웹 서비스 소비자 변환을 작성할 경우 실행할 작업을 선택합니다. 웹 서비스 소비자 변환의 작업 입력 및 작업 출력 계층을 볼 수 있습니다. 계층에서 SOAP 요청 메시지 및 SOAP 응답 메시지의 구조를 정의합니다.

단방향 입력 작업으로 WSDL을 가져올 수도 있습니다. 단방향 입력 작업으로 WSDL을 가져올 경우 더미 출력 포트를 작성해야 합니다.

## 웹 서비스 소비자 변환 포트

변환 포트를 볼 때 작업 계층을 볼 필요가 없는 경우 포트를 표시하십시오. 포트를 표시할 경우 그룹을 정의하고, 포트를 정의하며, 작업 출력에서 출력 포트에 노드를 매핑할 수 있습니다.

다음 그림에서는 재사용 불가능한 웹 서비스 소비자 변환에 대한 포트를 보여 줍니다.



웹 서비스 소비자 변환에는 다중 입력 그룹 및 다중 출력 그룹이 있을 수 있습니다. 포트를 작성할 때 그룹을 작성하고 포트를 그룹에 추가합니다. 작업 입력 또는 작업 출력 계층의 구조를 기반으로 포트를 그룹 계층으로 정의합니다. 키를 추가하여 하위 그룹을 상위 그룹에 연결합니다. 계층의 최하위 그룹을 제외한 모든 그룹에는 기본 키가 있어야 합니다. 루트 그룹을 제외한 계층의 모든 그룹에는 외래 키가 있어야 합니다.

변환에는 **RequestInput**이라는 루트 입력 그룹이 포함됩니다. 이 루트 입력 그룹에 기본 키를 추가해야 합니다. 키는 문자열, **bigint** 또는 정수여야 합니다.

다른 통과 포트를 루트 입력 그룹에 추가할 수 있습니다. 통과 포트는 데이터를 수정하지 않고 변환에 전달합니다. 통과 포트는 입력 데이터에서 한 번만 발생할 수 있습니다. 통과 포트를 모든 출력 그룹에 추가할 수 있습니다. 출력 포트를 입력 포트에 연결합니다. SOAP 요청을 통해 전달하는 입력 값은 SOAP 응답의 출력 행에서 반복됩니다.

또한 HTTP 헤더, 쿠키 포트, 동적 URL 포트 및 웹 서비스 보안 인증 포트를 루트 입력 그룹에 추가할 수 있습니다. 루트 그룹의 데이터는 한 번 발생합니다.

작업 출력 노드를 출력 포트에 매핑하려면 **위치** 열의 필드를 클릭하고 **위치 선택** 대화 상자에서 계층을 확장합니다. 그런 다음, 계층에서 노드를 선택합니다.

## HTTP 헤더 입력 포트

웹 서비스에 추가 HTTP 헤더가 필요할 수 있습니다. 루트 입력 그룹에 입력 포트를 작성하여 추가 헤더 정보를 웹 서비스 공급자에게 전달할 수 있습니다.

HTTP 헤더 및 HTTP 포트를 추가하려면 루트 입력 그룹을 선택하고 **새로 만들기** 단추 옆의 화살표를 클릭합니다. 그런 다음 **HTTP 헤더**를 클릭합니다. 헤더 이름 및 포트 이름을 입력합니다.

여러 개의 HTTP 헤더를 작성할 수 있습니다.



## 기타 입력 포트

미리 정의된 입력 포트를 웹 서비스 소비자 변환에 추가할 수 있습니다.

다음과 같은 미리 정의된 입력 포트를 추가할 수 있습니다.

### 쿠키 포트

쿠키 인증을 사용하도록 웹 서비스 소비자 변환을 구성할 수 있습니다. 원격 웹 서비스 서버는 쿠키를 기반으로 웹 서비스 소비자 사용자를 추적합니다. 매핑이 웹 서비스를 여러 번 호출하는 경우 성능을 향상시킬 수 있습니다.

쿠키 포트를 웹 서비스 요청 메시지에 연결하면 웹 서비스 공급자가 응답 메시지에 쿠키 값을 반환합니다. 쿠키 값을 매핑의 다른 변환 다운스트림에 전달하거나 파일에 쿠키 값을 저장할 수 있습니다. 쿠키 값을 파일에 저장할 때 쿠키를 웹 서비스 소비자 변환의 입력으로 구성할 수 있습니다.

쿠키 출력 포트를 웹 서비스 소비자 변환 출력 그룹 중 하나에 연결할 수 있습니다.

### 끝점 URL 포트

웹 서비스 소비자 변환이 끝점 URL을 사용하여 웹 서비스에 연결합니다. WSDL 파일, 웹 서비스 연결 또는 끝점 URL 입력 포트에서 끝점 URL을 정의할 수 있습니다. 변환이 포트에서 동적으로 URL을 수신할 경우 데이터 통합 서비스가 WSDL 파일 또는 웹 서비스 연결에서 정의된 URL을 재정의합니다.

웹 서비스 소비자 변환에는 각 웹 서비스 요청에 대한 하나의 URL 포트 값이 포함될 수 있습니다. 끝점 URL 포트는 루트 입력 그룹에 추가합니다.

### WS-Security 포트

웹 서비스 보안은 웹 서비스 연결에서 활성화합니다. 웹 서비스 보안을 활성화할 경우 웹 서비스 연결 또는 WS-Security 입력 포트에서 사용자 이름 및 암호를 정의해야 합니다.

WS-Security 포트를 추가할 경우 변환의 입력 포트를 통해 사용자 이름과 암호를 전달해야 합니다. 변환이 포트에서 동적으로 사용자 이름과 암호를 수신할 경우 데이터 통합 서비스가 웹 서비스 연결에서 정의된 값을 재정의합니다.

**참고:** 웹 서비스 연결에는 HTTP 및 WS-Security 인증에 대한 하나의 사용자 이름과 암호가 포함됩니다.

미리 정의된 입력 포트를 추가하려면 **포트** 영역에서 루트 입력 그룹을 클릭합니다. **새로 만들기** 단추 옆에 있는 화살표를 클릭한 다음 **기타 포트**를 클릭합니다. 추가할 포트를 선택합니다.

## 웹 서비스 소비자 변환 입력 매핑

변환 포트를 보는 경우 작업 입력 계층을 보려면 입력 매핑을 표시합니다. 입력 매핑을 표시할 경우 입력 그룹 및 입력 포트를 정의하고 입력 포트를 작업 입력 노드에 매핑할 수 있습니다.

입력 매핑에는 다음 영역이 포함됩니다.

### 포트

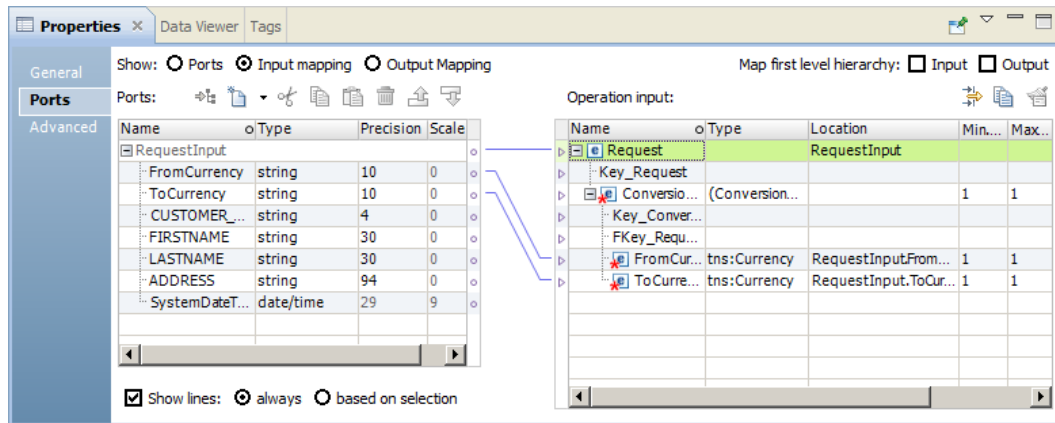
**포트** 영역에서 변환 입력 그룹 및 입력 포트를 작성합니다.

### 작업 입력

**작업 입력** 영역에는 웹 서비스 소비자 변환이 웹 서비스에 전송하는 SOAP 요청 메시지의 노드가 표시됩니다. 변환을 작성하기 위해 사용하는 WSDL 데이터 개체에서 작업 입력 계층을 정의합니다.



다음 그림에서는 재사용 불가능한 웹 서비스 소비자 변환에 대한 입력 매핑을 보여 줍니다.



입력 포트를 작성한 후 **포트** 영역의 입력 포트를 **작업 입력** 영역의 노드에 매핑합니다. 입력 포트를 작업 입력의 노드에 매핑할 경우 포트의 위치가 **작업 입력** 영역의 **위치** 열에 표시됩니다.

입력 계층의 첫 번째 수준을 매핑하기로 선택한 경우 **Developer tool**에서 작업 입력의 첫 번째 수준에 있는 노드를 입력 포트에 매핑합니다. **Developer tool**은 매핑을 수행할 포트도 작성합니다. 계층의 첫 번째 수준에 하나 이상의 다중 발생 하위 노드가 있는 다중 발생 상위 노드가 포함된 경우 **Developer tool**은 계층의 첫 번째 수준을 매핑하지 않습니다.

하나의 문자열 또는 텍스트 입력 포트에서 전체 **SOAP** 요청 메시지로 **XML** 데이터를 매핑할 수 있습니다. **XML** 데이터를 전체 **SOAP** 요청으로 매핑할 경우 포트를 작업 입력의 노드에 매핑할 수 없습니다.

입력 포트를 작업 입력의 노드에 연결하는 행을 표시하도록 선택할 수도 있습니다.

## 관련 항목:

- [“웹 서비스 SOAP 메시지 생성 개요” 페이지 653](#)

## 입력 포트를 노드에 매핑하기 위한 규칙 및 지침

입력 포트를 작업 입력 계층의 노드에 매핑하는 경우 다음 규칙을 검토합니다.


- 입력 포트를 계층의 노드 하나에 매핑할 수 있습니다. 동일한 포트는 계층의 여러 키에 매핑할 수 있습니다.
- 입력 포트 및 노드의 데이터 유형은 호환되어야 합니다.
- 한 입력 그룹의 포트를 작업 입력의 여러 계층 수준에 매핑할 수 있습니다.
- 입력 포트를 작업 입력의 키에 매핑해야 합니다. 키에 매핑하는 포트는 문자열, 정수 또는 **bigint** 데이터 유형이어야 합니다. **SOAP** 메시지에 포함하는 계층 수준 이상의 작업 입력에 있는 모든 수준의 키에 데이터를 매핑하십시오. 매핑하는 수준을 포함하여 그 이상의 모든 수준에 대한 외래 키를 포함하십시오.

**참고:** 작업 입력 계층의 가장 낮은 수준만 매핑할 경우 입력 포트를 키에 매핑할 필요가 없습니다.

- 여러 문자열, **bigint** 또는 정수 입력 포트를 **작업 입력** 영역의 키에 매핑하여 복합 키를 작성할 수 있습니다. 복합 키의 **위치** 필드를 클릭하면 입력 포트의 순서를 다시 정렬하거나 포트 중 하나를 제거할 수 있습니다.

## 보기 사용자 지정의 옵션

**작업 입력** 영역에 키가 표시되도록 작업 입력 계층을 변경할 수 있습니다. 노드 정렬 방법을 정의하는 그룹화 구성을 표시할 수도 있습니다.

보기 사용자 지정 단추( )를 **작업 입력** 영역에서 클릭합니다. 다음 옵션을 활성화합니다.

## 시퀀스, 선택 및 모두

요소 정의가 시퀀스, 선택 또는 모두인지 여부를 나타내는 행을 표시합니다.

모든 그룹의 노드가 SOAP 메시지에 모두 포함되어야 합니다.

시퀀스 그룹의 노드는 WSDL에 지정된 순서여야 합니다.

선택 그룹의 노드 하나 이상이 SOAP 메시지에 표시되어야 합니다.

## 키

**작업 입력** 영역에서 키를 봅니다. **작업 입력** 영역에는 각 그룹의 키가 포함됩니다. **포트** 영역의 입력 포트에 키를 추가할 수 있습니다.

## 작업 입력에 입력 포트 매핑

변환 입력 매핑을 표시할 때 입력 그룹을 정의하고 입력 포트를 정의하고 입력 포트를 작업 입력 노드에 매핑할 수 있습니다.

1. 웹 서비스 소비자 변환을 엽니다.
2. 변환 입력 매핑을 표시하려면 다음 방법 중 하나를 사용합니다.
  - 재사용 가능한 변환의 경우 **개요** 보기를 클릭합니다. 입력 매핑을 표시하도록 선택합니다.
  - 재사용 불가능한 변환의 경우 **속성** 보기에서 **포트** 탭을 클릭합니다. 입력 매핑을 표시하도록 선택합니다.
3. 루트 입력 그룹에 대한 기본 키를 정의합니다.
4. 입력 그룹 또는 포트를 **포트** 영역에 추가하려면 다음 방법 중 하나를 사용합니다.

옵션	설명
노드 끌기	<b>작업 입력</b> 영역에서 그룹 노드 또는 하위 노드를 <b>포트</b> 영역의 빈 열로 끌어서 놓습니다. 노드가 그룹 노드일 경우 Developer tool에서 포트 없이 그룹을 추가합니다.
그룹 또는 포트를 수동으로 추가	그룹을 추가하려면 <b>새로 만들기</b> 단추 옆의 화살표를 클릭한 다음 <b>그룹</b> 을 클릭합니다. 포트를 추가하려면 <b>새로 만들기</b> 단추 옆의 화살표를 클릭한 다음 <b>필드</b> 를 클릭합니다.
다른 변환에서 포트 끌기	편집기에서 다른 변환의 포트를 웹 서비스 소비자 변환으로 끌어서 놓습니다.
포트 복사	다른 변환에서 포트를 선택하여 <b>포트</b> 영역에 복사합니다. 포트를 복사하기 위해 키보드 바로가기를 사용하거나 Developer tool의 복사 및 붙여넣기 단추를 사용할 수 있습니다.
계층의 첫 번째 수준 매핑 선택	<b>계층의 첫 번째 수준 매핑</b> 을 선택합니다. Developer tool이 작업 입력의 첫 번째 수준에 있는 노드를 입력 포트 및 그룹에 매핑합니다. Developer tool은 매핑을 수행할 입력 포트 및 그룹도 작성합니다.

5. 포트를 수동으로 작성하거나 다른 변환의 포트를 복사하는 경우 **작업 입력** 영역에서 **위치** 열을 클릭하고 목록에서 포트를 선택합니다.
6. 입력 포트를 복합 키로 매핑하려면 다음 방법 중 하나를 사용하십시오.

옵션	설명
입력 포트 끌기	두 개 이상의 입력 포트를 선택하고 작업 입력 계층의 키로 끌어서 놓습니다.

옵션	설명
<b>위치 선택</b> 대화 상자에서 입력 포트 선택	작업 입력 계층에서 키의 <b>위치</b> 열을 클릭한 다음 입력 포트를 선택합니다.

7. 노드의 위치를 지우려면 다음 방법 중 하나를 사용하십시오.

옵션	설명
<b>지우기</b> 클릭	<b>작업 입력</b> 영역에서 하나 이상의 노드를 선택하고 <b>지우기</b> 를 클릭합니다.
포트를 노드에 연결하는 행 삭제	입력 포트를 작업 입력의 노드에 연결하는 하나 이상의 행을 선택하고 <b>삭제</b> 를 누릅니다.

8. 연결된 WSDL 데이터 개체에 **anyType** 요소, 임의 요소, **anyAttribute** 특성, 파생된 유형 요소 또는 대체 그룹이 포함되는 경우 **작업 입력** 영역에서 개체를 선택합니다. 노드의 **유형** 열에서 **선택**을 클릭한 다음, 목록에서 하나 이상의 유형, 요소 또는 특성을 선택합니다.
9. 문자열 또는 텍스트 입력 포트의 XML 데이터를 전체 SOAP 요청에 매핑하려면 포트를 마우스 오른쪽 단추로 클릭하고 **XML로 매핑**을 선택합니다.

## 웹 서비스 소비자 변환 출력 매핑

변환 포트를 보는 경우 작업 출력 계층을 보려면 출력 매핑을 표시하십시오. 출력 매핑을 표시할 경우 출력 그룹을 정의하고, 출력 포트를 정의하며, 작업 출력 노드를 출력 포트에 매핑할 수 있습니다.

출력 매핑에는 다음 영역이 포함됩니다.

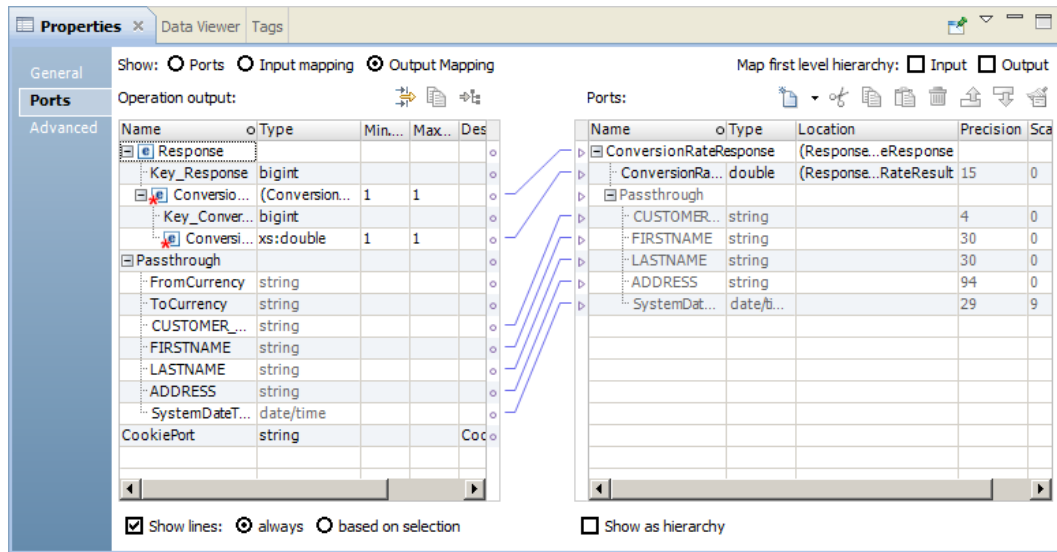
### 작업 출력

**작업 출력** 영역에는 웹 서비스가 웹 서비스 소비자 변환에 반환하는 SOAP 응답 메시지의 노드가 표시됩니다. 변환을 작성하기 위해 사용하는 WSDL 데이터 개체에서 작업 출력 계층을 정의합니다.

### 포트

**포트** 영역에서 변환 출력 그룹 및 포트를 작성합니다.

다음 그림에서는 재사용 불가능한 웹 서비스 소비자 변환에 대한 출력 매핑을 보여 줍니다.



출력 포트를 작성한 후 **작업 출력** 영역의 노드를 **포트** 영역의 포트에 매핑합니다. 작업 입력에서 출력 포트에 노드를 매핑할 경우 노드 위치가 **포트** 영역의 **위치** 열에 표시됩니다.

출력 계층의 첫 번째 수준을 매핑하기로 선택한 경우 **Developer tool**에서 작업 출력의 첫 번째 수준에 있는 노드를 출력 포트에 매핑합니다. **Developer tool**은 매핑을 수행할 포트도 작성합니다. 계층의 첫 번째 수준에 하나 이상의 다중 발생 하위 노드가 있는 다중 발생 상위 노드가 포함된 경우 **Developer tool**은 계층의 첫 번째 수준을 매핑하지 않습니다.

계층에 출력 포트를 표시하도록 선택할 수 있습니다. 각 하위 그룹은 상위 그룹 아래에 나타납니다. 작업 출력의 노드를 출력 포트에 연결하는 행을 표시하도록 선택할 수도 있습니다.

연결된 WSDL 데이터 개체가 리포지토리에서 삭제된 경우 **Developer tool**이 작업 노드의 위치를 출력 매핑에 유지합니다. 출력 매핑을 표시할 경우 **포트** 영역에서 출력 포트의 **위치** 열에 작업 노드의 위치를 계속 표시합니다. 다른 WSDL을 변환과 연결할 경우 **Developer tool**에서 각 위치가 유효한지 확인합니다. 위치가 더 이상 유효하지 않을 경우 **Developer tool**이 출력 매핑의 **포트** 영역에서 작업 노드의 위치를 지웁니다.

## 관련 항목:

- [“웹 서비스 SOAP 메시지 구문 분석 개요” 페이지 645](#)

## 노드를 출력 포트에 매핑하기 위한 규칙 및 지침

작업 출력 계층의 노드를 출력 포트에 매핑할 경우 다음 규칙을 검토하십시오.

- 작업 출력 노드 및 출력 포트에는 호환 가능한 데이터 유형이 있어야 합니다.
- 그룹의 둘 이상의 출력 포트에 노드를 매핑할 수 없습니다.
- 통과 포트를 제외한 모든 출력 포트에는 올바른 위치가 있어야 합니다.
- 다중 발생 하위 노드를 빈 출력 포트에 끝 경우 그룹을 다른 출력 그룹에 연결해야 합니다. 그룹을 선택하면 **Developer tool**이 그룹을 연결하는 키를 작성합니다.
- 여러 번 발생하는 요소를 상위 요소가 포함된 그룹으로 끝면 포함할 하위 요소 발생 수를 구성할 수 있습니다. 또는 상위 그룹을 변환 출력에서 여러 번 발생하는 하위 그룹으로 대체할 수 있습니다.

## SOAP 메시지를 XML로 매핑

데이터를 개별 출력 포트로 반환하지 않고 전체 SOAP 메시지를 XML로 매핑할 수 있습니다.


SOAP 메시지를 XML로 매핑하면 데이터 통합 서비스에서 전체 SOAP 메시지를 하나의 포트에 반환합니다. 출력 포트를 작성하지 마십시오.

전체 메시지를 매핑하려면 **작업 출력** 영역의 루트 그룹을 마우스 오른쪽 단추로 클릭합니다. **XML로 매핑**을 선택합니다.

Developer tool에서 문자열 출력 포트가 작성됩니다. 전체 자릿수는 65535바이트입니다.

## 보기 사용자 지정의 옵션

**작업 출력** 영역에 쿠키 포트, 통과 포트 및 키가 표시되도록 작업 출력 계층을 변경할 수 있습니다. 노드 정렬 방법을 정의하는 그룹화 구성을 표시할 수도 있습니다.

**보기 사용자 지정** 단추(  )를 **작업 출력** 영역에서 클릭합니다. 다음 옵션을 활성화합니다.

### 시퀀스, 선택 및 모두

요소 정의가 시퀀스, 선택 또는 모두인지 여부를 나타내는 행을 표시합니다.

모든 그룹의 노드가 SOAP 메시지에 모두 포함되어야 합니다.

시퀀스 그룹의 노드는 WSDL에 지정된 순서여야 합니다.

선택 그룹의 노드 하나 이상이 SOAP 메시지에 표시되어야 합니다.

### 키

**작업 출력** 영역에서 키를 봅니다. **작업 출력** 영역에는 각 그룹의 키가 포함됩니다. **포트** 영역의 출력 포트에 키를 추가할 수 있습니다.

### 통과 포트

**작업 출력** 영역에 통과 포트가 표시됩니다. 통과 포트는 데이터를 변경하지 않고 변환을 통해 데이터를 전달하는 포트입니다. 작업 출력의 통과 포트를 모든 웹 서비스 소비자 변환 출력 그룹에 연결할 수 있습니다. 통과 포트는 데이터를 한 번 수신하므로 응답 메시지의 SOAP 수준에 위치합니다.

### 쿠키 포트

쿠키 포트를 표시합니다. 쿠키 인증을 구성한 경우 원격 웹 서비스 서버가 쿠키를 바탕으로 웹 서비스 소비자 사용자를 추적합니다. 요청 메시지에서 웹 서비스 쿠키를 연결한 경우 웹 서비스가 응답 메시지에서 쿠키를 반환합니다. 작업 출력의 쿠키를 모든 웹 서비스 소비자 변환 출력 그룹에 연결할 수 있습니다.

## 출력 포트에 작업 출력 매핑

변환 출력 매핑을 표시할 때 출력 그룹을 정의하고 출력 포트를 정의하고 작업 출력 노드를 출력 포트에 매핑할 수 있습니다.

1. 웹 서비스 소비자 변환을 엽니다.
2. 변환 출력 매핑을 표시하려면 다음 방법 중 하나를 사용합니다.
  - 재사용 가능한 변환의 경우 **개요** 보기를 클릭합니다. 출력 매핑을 표시하도록 선택합니다.
  - 재사용 불가능한 변환의 경우 **속성** 보기에서 **포트** 탭을 클릭합니다. 출력 매핑을 표시하도록 선택합니다.

3. 출력 그룹 또는 포트를 **포트** 영역에 추가하려면 다음 방법 중 하나를 사용합니다.

옵션	설명
노드 끌기	<b>작업 출력</b> 영역의 그룹 노드 또는 하위 노드를 <b>포트</b> 영역의 빈 열로 끌어서 놓습니다. 노드가 그룹 노드일 경우 Developer tool에서 포트 없이 그룹을 추가합니다.
그룹 또는 포트를 수동으로 추가	그룹을 추가하려면 <b>새로 만들기</b> 단추 옆의 화살표를 클릭한 다음 <b>그룹</b> 을 클릭합니다. 포트를 추가하려면 <b>새로 만들기</b> 단추 옆의 화살표를 클릭한 다음 <b>필드</b> 를 클릭합니다.
다른 변환에서 포트 끌기	편집기에서 다른 변환의 포트를 웹 서비스 소비자 변환으로 끌어서 놓습니다.
포트 복사	다른 변환에서 포트를 선택하여 <b>포트</b> 영역에 복사합니다. 포트를 복사하기 위해 키보드 바로 가기를 사용하거나 Developer tool의 복사 및 붙여넣기 단추를 사용할 수 있습니다.
계층의 첫 번째 수준 매핑 선택	<b>계층의 첫 번째 수준 매핑</b> 을 선택합니다. Developer tool이 작업 출력의 첫 번째 수준에 있는 노드를 출력 포트 및 그룹에 매핑합니다. Developer tool은 매핑을 수행할 출력 포트 및 그룹도 작성합니다.

4. 포트를 수동으로 작성하거나 다른 변환의 포트를 복사하는 경우 **포트** 영역에서 **위치** 열을 클릭하고 목록에서 노드를 선택합니다.
5. 포트의 위치를 지우려면 다음 방법 중 하나를 사용합니다.

옵션	설명
지우기 클릭	<b>포트</b> 영역에서 하나 이상의 포트를 선택하고 <b>지우기</b> 를 클릭합니다.
노드를 포트에 연결하는 행 삭제	작업 출력의 노드를 출력 포트에 연결하는 하나 이상의 행을 선택하고 <b>삭제</b> 를 누릅니다.

6. 연결된 WSDL 데이터 개체에 anyType 요소, 임의 요소, anyAttribute 특성, 파생된 유형 요소 또는 대체 그룹이 포함되는 경우 **작업 출력** 영역에서 개체를 선택합니다. 노드의 **유형** 열에서 **선택**을 클릭한 다음, 목록에서 하나 이상의 유형, 요소 또는 특성을 선택합니다.
7. 전체 SOAP 응답 메시지를 XML로 매핑하려면 **작업 출력** 영역에서 루트 그룹을 마우스 오른쪽 단추로 클릭하고 **XML로 매핑**을 선택합니다.

## 웹 서비스 소비자 변환 고급 속성

웹 서비스 소비자 변환 고급 속성에는 추적 수준, 일반 결합 포트, 웹 서비스 연결 및 동시 웹 서비스 요청 메시지가 포함되어 있습니다.

고급 탭에서 웹 서비스 소비자 변환에 대한 다음 고급 속성을 정의할 수 있습니다.

### 추적 수준

이 변환에 대해 로그에 표시되는 세부 정보의 양입니다. 간단, 보통, 자세한 정보 표시 초기화 또는 자세한 정보 표시 데이터를 선택할 수 있습니다. 기본값은 보통입니다.

### SOAP 작업

WSDL에 정의된 SOAP 작업 값을 웹 서비스 소비자 변환에 대한 상수 값으로 재정의합니다.

## 일반 SOAP 결함 처리 활성화

WSDL에 정의되지 않은 결함 메시지를 반환합니다. **GenericFault** 출력 그룹에 출력 포트를 작성하여 결함 코드 및 메시지를 처리합니다.

다음 테이블에는 SOAP 1.1 및 SOAP 1.2에 대한 결함 출력 포트가 설명되어 있습니다.

SOAP 1.1용 결함 출력 포트	SOAP 1.2용 결함 출력 포트	설명
결함 코드	코드*	결함 ID 코드를 반환합니다.
결함 문자열	이유*	결함 메시지에 오류 설명을 반환합니다.
결함 세부 정보	세부 정보	일반 결함 메시지를 통해 웹 서비스 공급자가 웹 서비스 소비자 변환에 전달하는 사용자 지정 정보를 반환합니다.
결함 작업자	역할	결함을 발생시킨 개체에 대한 정보를 반환합니다.
-	노드	결함을 생성한 SOAP 노드의 URI를 반환합니다.
* 코드 및 이유 출력 포트는 계층적입니다.		

**참고:** 코드 결함 출력 포트를 확장하여 SubCode 결함 출력 포트를 한 수준 위로 추출할 수 있습니다.

## HTTP 오류 처리 활성화

웹 서비스에서 HTTP 오류를 반환합니다. **GenericFault** 출력 그룹에 HTTP 오류 출력 포트를 작성합니다.

### 결함을 오류로 처리

결함 메시지를 매핑 로그에 추가합니다. 결함이 발생하면 데이터 통합 서비스가 매핑에 대한 오류 수를 증분합니다. 초기 선택 및 푸시인 최적화를 허용하려면 이 속성을 비활성화합니다. 기본값은 활성화됩니다.

## 연결

웹 서비스에 연결할 웹 서비스 연결 개체를 식별합니다. **Developer tool**에서 웹 서비스 연결을 작성합니다. **Developer tool** 또는 **Administrator** 도구에서 웹 서비스 연결을 편집합니다. 웹 서비스 연결을 구성할 경우 끝점 URL, 웹 서비스에 필요한 보안 유형 및 연결 제한 시간을 구성합니다.

웹 서비스 소비자 변환이 끝점 URL을 사용하여 웹 서비스에 연결합니다. WSDL 파일, 웹 서비스 연결 또는 끝점 URL 입력 포트에서 끝점 URL을 정의할 수 있습니다.

웹 서비스 연결을 구성하는 경우를 확인하려면 다음 지침을 사용하십시오.

- WSDL 파일의 URL과 다른 끝점 URL을 사용할 경우, 그리고 끝점 URL 입력 포트를 사용하지 않을 경우 연결을 구성합니다.
- 연결하는 웹 서비스에 웹 서비스 보안, HTTP 인증 또는 SSL 인증서가 필요한 경우 연결을 구성합니다.
- 기본 연결 제한 시간을 변경할 경우 연결을 구성합니다.

**참고:** 리포지토리의 WSDL 데이터 개체를 웹 서비스 연결과 연관시킬 수 있습니다. 연관된 연결은 해당 WSDL에서 작성하는 모든 웹 서비스 소비자 변환의 기본 연결이 됩니다.

## 압축 활성화

GZIP 압축 방법을 사용하여 SOAP 요청의 코딩을 활성화하고 GZIP 또는 deflate를 사용하여 SOAP 응답의 디코딩을 활성화합니다.

## XML 스키마 유효성 검사

런타임 시 SOAP 응답 메시지의 유효성을 검사합니다. **올바르지 않은 XML에 대한 오류** 또는 **유효성 검사 없음**을 선택합니다.

## 정렬된 입력

데이터 통합 서비스에서 모든 입력 데이터를 처리하지 않고도 출력을 생성할 수 있습니다. 입력 데이터를 작업 입력 계층의 키별로 정렬한 경우 정렬된 입력을 활성화합니다.

## 푸시인 최적화

푸시인 최적화를 활성화합니다. **푸시인 최적화** 속성의 **열기** 단추를 클릭하여 필터 값을 수신하는 필터 포트를 선택합니다. 각 필터 포트에 대해 필터링된 열을 웹 서비스 응답에 포함하는 출력 포트를 선택합니다.

## 부작용 있음

웹 서비스가 행 반환 이외의 다른 기능을 수행함을 나타내는 확인란입니다. 웹 서비스가 행 반환 이외에 개체를 수정하거나 다른 개체 또는 함수와 상호 작용할 경우 웹 서비스 소비자 변환에 부작용이 있습니다. 웹 서비스에서 데이터베이스를 수정하거나, 합계에 추가하거나, 예외를 발생시키거나, 이메일을 작성하거나, 부작용이 있는 다른 웹 서비스를 호출할 수 있습니다. 푸시인 최적화 또는 초기 선택 최적화를 허용하기 위해 **부작용 있음** 속성을 비활성화합니다. 기본값은 활성화됩니다.

## 동시 실행 활성화

웹 서비스 소비자 변환을 활성화하여 여러 웹 서비스 요청을 병렬로 보내도록 웹 서비스에 대한 여러 동시 연결을 작성할 수 있습니다. 웹 서비스 소비자 변환을 활성화하여 웹 서비스에 대한 여러 동시 연결을 작성할 때 메모리 소비 제한 및 동시 연결 수 제한을 설정할 수 있습니다.

다음 테이블에는 옵션이 설명되어 있습니다.

옵션	설명
동시 실행 활성화	웹 서비스에 대한 여러 동시 연결을 작성합니다.
동시 실행 연결 제한	동시 실행 웹 서비스 연결 수입니다. 기본값은 20입니다.
총 동시 실행 메모리 제한(MB)	모든 동시 실행 연결에 대한 총 메모리 할당 한도입니다. 기본값이 100MB입니다.

## 웹 서비스 오류 처리

SOAP 결함 및 HTTP 오류 다운스트림을 매핑에서 전달하도록 웹 서비스 소비자 변환을 구성할 수 있습니다. 결함이 발생할 경우 오류 수를 증분할 수 있습니다. 변환 고급 속성에서 웹 서비스 오류 처리를 구성하십시오.

웹 서비스에서 응답 메시지를 반환하거나 결함을 반환합니다. 결함은 오류입니다. 웹 서비스는 발생하는 오류를 기반으로 다른 결함을 생성할 수 있습니다.

웹 서비스 소비자 변환에서 다음 유형의 결함을 반환할 수 있습니다.

### SOAP 결함

WSDL에서 정의하는 SOAP 오류입니다. 웹 서비스 응답 메시지에서 결함을 반환하는 출력 오류 포트를 구성하십시오. SOAP 1.1 바인딩의 경우 데이터 통합 서비스가 결함에 대한 결함 메시지, 결함 코드, 결함 문자열 및 결함 작업자 요소를 반환합니다. SOAP 1.2 바인딩에 대해서는 데이터 통합 서비스가 결함에 대한 결함 메시지, 코드, 이유, 노트 및 역할 요소를 반환합니다.

### 일반 SOAP 결함

웹 서비스가 런타임 시 일반 SOAP 결함을 생성합니다. SOAP 1.1 바인딩 및 SOAP 1.2 바인딩의 경우 결함 요소가 다릅니다. WSDL에서 일반 SOAP 결함을 정의하지 않습니다. 일반 SOAP 결함에는 인증 실패 및 SOAP 요청 오류가 포함됩니다.



## HTTP 오류

변환에서 HTTP 오류 처리를 활성화한 경우 **Developer tool**이 HTTP 결함 출력 포트를 추가합니다. 데이터 통합 서비스가 단일 문자열 포트를 통해 웹 서비스의 HTTP 오류를 반환합니다. HTTP 오류에는 오류 코드와 메시지가 포함됩니다.

웹 서비스의 **SOAP** 응답에 유효하지 않은 **XML** 데이터가 있는 경우 웹 서비스 소비자 변환에서 오류를 반환합니다.

**SOAP** 결함을 오류로 처리할지 구성할 수 있습니다. 결함을 오류로 처리를 활성화하고 **SOAP** 오류가 발생한 경우 데이터 통합 서비스가 매핑에 대한 오류 수를 증분합니다. 결함이 메시지 로그에 표시됩니다.

## 메시지 압축

**SOAP** 메시지 압축을 활성화하면 웹 서비스 소비자 변환이 웹 서비스 요청 메시지를 압축하고 압축된 웹 서비스 응답 메시지를 수신합니다.

웹 서비스 소비자 변환이 **GZip** 압축으로 **SOAP** 요청을 인코딩합니다. 또한 **GZip** 또는 **deflate** 압축으로 인코딩된 응답 메시지를 허용합니다.

데이터 통합 서비스는 웹 서비스에서 응답을 받을 때 **SOAP** 메시지의 **Content-Encoding** HTTP 헤더를 확인하고 메시지를 디코딩합니다.

기본값은 압축 인코딩 없음입니다. 웹 서비스는 **SOAP** 응답을 압축하지 않습니다.

다음 테이블에는 압축을 켜거나 끌 때의 요청 및 응답 메시지의 헤더가 표시되어 있습니다.

압축	헤더
켜짐	Content-Encoding 헤더: GZip Accept-Encoding 헤더: GZip, deflate
꺼짐	빈 Content-Encoding 헤더 빈 Accept-Encoding 헤더

웹 서비스에서 기본 압축으로 응답 메시지를 인코딩하는 경우도 간혹 있습니다. 웹 서비스 소비자 변환은 **GZip** 또는 **deflate**로 인코딩된 메시지를 디코딩합니다. 웹 서비스에서 예기치 않게 응답 메시지를 인코딩할 경우 웹 서비스 소비자 변환이 매핑 로그에 메시지를 기록합니다.

압축은 변환의 고급 속성에서 활성화할 수 있습니다.

## 동시 실행

웹 서비스 소비자 변환을 활성화하여 여러 웹 서비스 요청을 병렬로 보내도록 웹 서비스에 대한 여러 동시 연결을 작성할 수 있습니다.

예를 들어 은행 정보를 쿼리하는 동안 웹 서비스 소비자 변환을 동시 실행에 대해 구성하여 여러 행을 병렬로 보낼 수 있습니다. 입력 행이 20개인 경우 20개 요청을 동시에 보내 처리를 가속화할 수 있습니다.

웹 서비스 소비자 변환에서 동시 실행을 활성화할 경우 총 메모리 사용 제한을 구성할 수 있습니다.

웹 서비스 소비자 변환에서 동시 실행을 활성화할 경우 동시 웹 서비스 연결 수를 구성할 수 있습니다.

## 동시 실행에 대한 규칙 및 지침

동시 실행을 사용하는 동안에는 다음 규칙 및 지침을 사용합니다.

- 동시 실행은 정렬된 입력 행을 웹 서비스에 대한 여러 개의 동시 연결로 지원합니다. 순서가 지정된 출력 행은 지원되지 않습니다.
- 데이터 집합이 100개 행을 초과할 경우 동시 실행을 사용합니다.
- 동시 웹 서비스 연결의 수를 늘리지 않는 것이 좋습니다. 동시 웹 서비스 연결 수는 운영 체제에서 사용하는 소켓 수에 연결됩니다. 소켓 수를 늘리면 비용이 증가합니다.
- 동시 실행 기능을 사용하는 동안 성능을 최적화하려면 최소 100MB 이상의 RAM이 포함된 멀티코어 프로세서 시스템을 사용합니다.
- 동시 실행 메모리 제한은 웹 서비스를 호출하는 동안 동시 작업 흐름에서 사용하는 메모리를 나타냅니다.
- 웹 서비스 소비자 변환에서 동시 실행을 활성화할 경우 메모리 소비 제한을 구성할 수 있습니다. 메모리 사용량이 서버의 실제 RAM을 초과하지 않도록 하십시오.

## 동시 실행을 위한 모범 사례

동시 실행을 사용하는 동안 성능을 최적화하려면 다음 모범 사례를 따르십시오.

- 총 동시 실행 메모리 제한 및 동시 실행 연결 제한의 기본 값을 변경하지 않습니다.
- 행이 100개 미만인 데이터 집합에 대해 동시 실행을 사용하지 않습니다.
- 동시 실행을 사용하는 동안 매핑에 통과 포트를 사용하지 않습니다.

# 필터 최적화

필터 최적화는 매핑을 통과하는 행 수를 줄여 성능을 향상시킵니다. 데이터 통합 서비스에서는 초기 선택 최적화 또는 푸시인 최적화를 적용할 수 있습니다.

데이터 통합 서비스는 필터 최적화 방법을 적용할 때 매핑에서 가능한 소스에 근접한 위치로 필터를 이동합니다. 데이터 통합 서비스가 매핑에서 변환 앞으로 필터를 이동할 수 없는 경우에는 필터 논리를 변환으로 푸시할 수 있습니다.

## 웹 서비스 소비자 변환의 초기 선택 최적화 활성화

웹 서비스 소비자 변환에 부작용이 없고 변환이 결함을 오류로 처리하지 않을 경우 변환에 대한 초기 선택 최적화를 활성화합니다.

1. 웹 서비스 소비자 변환의 **고급 속성** 보기를 엽니다.
2. **결함을 오류로 처리**를 선택 취소합니다.
3. **부작용 있음**을 선택 취소합니다.

## 웹 서비스 소비자 변환의 푸시인 최적화

변환이 SQL 데이터 서비스의 가상 테이블에 있는 경우 웹 서비스 소비자 변환을 사용하여 푸시인 최적화를 구성할 수 있습니다. 웹 서비스 변환이 매핑에 있을 때 변환을 사용하여 푸시인 최적화를 구성할 수 있습니다.

매핑은 웹 서비스를 호출하여 최종 사용자 SQL 쿼리의 문에 따라 데이터 집합 또는 데이터의 하위 집합을 검색합니다. 최종 사용자 SQL 쿼리에는 선택적 필터 조건이 포함됩니다.

웹 서비스 소비자 변환은 푸시인 최적화를 통해 필터 포트의 필터 값을 검색합니다. 필터 포트는 푸시인 최적화를 구성할 때 필터 포트로 인식하는 연결되지 않은 입력 포트입니다. 필터 포트에는 최종 사용자 쿼리에 필터가 포함되지 않을 경우 웹 서비스가 모든 행을 반환하도록 하는 기본값이 있습니다. 필터 포트는 통과 포트가 아닙니다.

**참고:** 필터 필드는 웹 서비스 요청에서 루트 그룹의 일부여야 합니다.

필터 포트를 구성할 때는 웹 서비스 소비자 변환에서 웹 서비스 응답의 열 데이터를 수신할 출력 포트를 식별해야 합니다. 예를 들어 필터 포트가 **EmployeeID**라는 입력 포트일 경우 응답의 출력 포트는 **EmployeeNum**이라는 포트가 될 수 있습니다. **Developer tool**에서 가상 테이블 읽기의 필터 논리를 웹 서비스 소비자 요청으로 푸시하려면 입력 필터 포트와 출력 포트가 연결되어야 합니다. 일반적으로 웹 서비스 요청에 대한 입력 포트는 웹 서비스 응답의 출력 포트와 다릅니다.

필터 포트를 구성할 때는 웹 서비스 소비자 변환에서 웹 서비스 응답의 열 데이터를 수신할 출력 포트를 식별해야 합니다. 예를 들어 필터 포트가 **EmployeeID**라는 입력 포트일 경우 응답의 출력 포트는 **EmployeeNum**이라는 포트가 될 수 있습니다. **Developer tool**에서 테이블 읽기의 필터 논리를 웹 서비스 소비자 요청에 푸시하려면 입력 필터 포트와 출력 포트가 연결되어야 합니다. 일반적으로 웹 서비스 요청에 대한 입력 포트는 웹 서비스 응답의 출력 포트와 다릅니다.

필터 필드는 통과 포트가 될 수 없습니다. 필터 포트를 구성하면 포트의 기본값이 필터 조건의 값으로 변경되므로 통과 출력 포트 값도 변경됩니다. 출력 통과 포트에 기반하는 필터는 예기치 않은 결과를 반환합니다.

여러 필터 식을 웹 서비스 소비자 변환에 푸시할 수 있습니다. 각 필터 조건은 다음 형식이어야 합니다.

<필드> = <상수>

필터 조건은 **AND**로 조인되어야 합니다. 조건을 **OR**로 조인할 수 없습니다.

## 웹 서비스 소비자 변환의 푸시인 최적화 예제

SQL 데이터 서비스는 사용자로부터 수신한 SQL 쿼리에 따라 모든 고객에 대한 주문을 반환하거나 특정 고객에 대한 주문을 반환합니다.

데이터 서비스에는 다음 구성 요소의 논리적 데이터 개체가 포함됩니다.

### Customer 테이블

고객 정보가 포함된 Oracle 데이터베이스 테이블입니다.

### 웹 서비스 소비자 변환

웹 서비스를 호출하여 고객의 최신 주문을 검색하는 변환입니다. 웹 서비스 소비자 변환에는 **customerID** 및 **orderNum**에 대한 입력 포트가 있습니다. 이 변환에는 **Customer** 테이블로부터 수신한 고객 데이터가 포함된 통과 포트가 있습니다. **orderNum** 포트는 필터 포트이고 연결되지 않습니다. **orderNum**의 기본값은 "\*"입니다. 웹 서비스에서 웹 서비스 요청을 통해 이 값을 수신할 경우 모든 주문이 반환됩니다.

### Orders 가상 테이블

웹 서비스의 고객 및 주문 데이터를 수신하는 가상 테이블입니다. 최종 사용자는 이 테이블을 쿼리합니다. **Orders**에는 고객 열, **orderID** 열 및 고객 및 주문 데이터가 포함됩니다.

최종 사용자는 다음 SQL 쿼리를 SQL 데이터 서비스에 전달합니다.

```
SELECT * from OrdersID where customer = 23 and orderID = 56
```

데이터 통합 서비스가 쿼리를 분할하여 매핑을 최적화합니다. 데이터 통합 서비스는 초기 선택 최적화를 사용하여 필터 논리 **customer = 23**을 **Customer** 테이블 읽기로 이동합니다. 데이터 통합 서비스는 푸시인 최적화를 사용하여 필터 논리 **orderID = 56**을 웹 서비스 소비자 변환 필터 포트에 푸시합니다. 웹 서비스 소비자 변환이 고객 **23**에 대한 **ordersID 56**을 검색합니다.

## 웹 서비스 소비자 변환의 푸시인 최적화 활성화

웹 서비스 소비자 변환에 부작용이 없고 변환이 결함을 오류로 처리하지 않을 경우 변환에 대한 푸시인 최적화를 활성화합니다.

1. 웹 서비스 소비자 변환의 **고급 속성** 보기를 엽니다.
2. **결함을 오류로 처리**를 선택 취소합니다.
3. **부작용 있음**을 선택 취소합니다.
4. **푸시인 최적화** 속성에서 **열기** 단추를 클릭합니다.
5. 입력 최적화 대화 상자에서 필터 포트 이름을 선택합니다.  
여러 필터 포트를 선택할 수 있습니다.
6. **출력 열**을 클릭합니다.
7. 각 필터 포트에 대해 필터링된 열을 웹 서비스 응답에 포함하는 출력 포트를 선택합니다.
8. 각 필터 포트에 대한 기본값을 입력합니다.

**참고:** 웹 서비스 소비자 포트가 필터 포트가 아닌 경우 기본값을 구성할 수 없습니다.

## 웹 서비스 소비자 변환 작성

재사용 가능하거나 재사용 불가능한 웹 서비스 소비자 변환을 작성할 수 있습니다. 재사용 가능 변환은 여러 매핑에 존재할 수 있습니다. 재사용 불가능 변환은 단일 매핑에만 존재합니다.

단일 WSDL 개체에서 SOAP 1.1 바인딩 및 SOAP 1.2 바인딩에 대한 웹 서비스 소비자 변환을 작성할 수 있습니다.

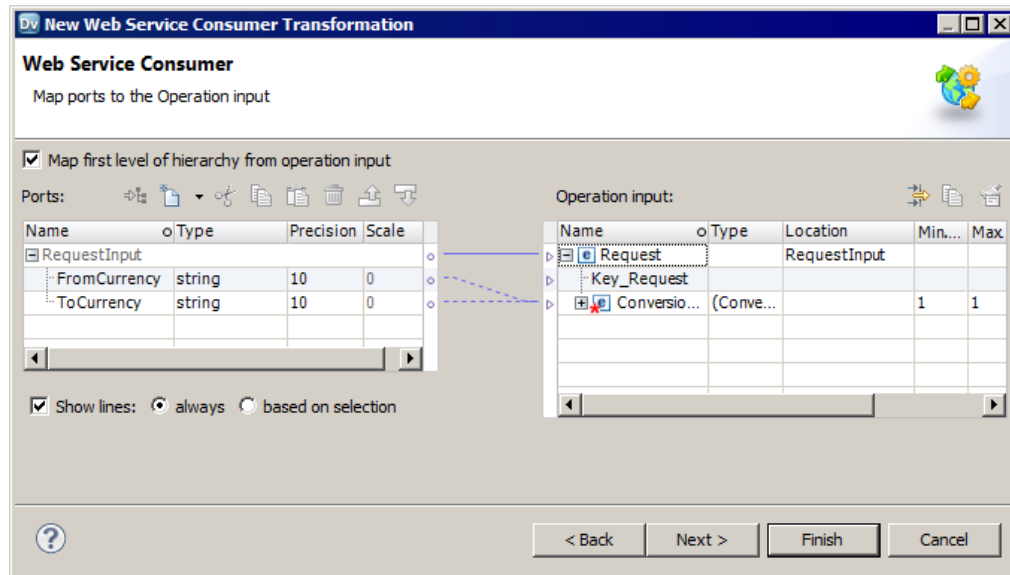
1. 변환을 작성하려면 다음 방법 중 하나를 사용합니다.

옵션	설명
재사용 가능	<b>개체 탐색기</b> 보기에서 프로젝트나 폴더를 선택합니다. <b>파일 &gt; 새로 만들기 &gt; 변환</b> 을 클릭합니다. 웹 서비스 소비자 변환을 선택하고 <b>다음</b> 을 클릭합니다.
재사용 불가능	매핑 또는 맵렛에서 변환 색상표의 웹 서비스 소비자 변환을 편집기로 끌어서 놓습니다.

새 웹 서비스 소비자 변환 대화 상자가 나타납니다.

2. WSDL 데이터 개체를 찾아보고 선택하여 웹 서비스 요청 및 응답 메시지를 정의합니다.  
WSDL이 리포지토리에 없는 경우 새 웹 서비스 소비자 변환 대화 상자에서 WSDL을 가져올 수 있습니다.
3. WSDL에서 작업을 찾아보고 선택합니다.  
SOAP 1.1 바인딩 또는 SOAP 1.2 바인딩이 있는 작업을 선택할 수 있습니다.
4. **다음**을 클릭합니다.

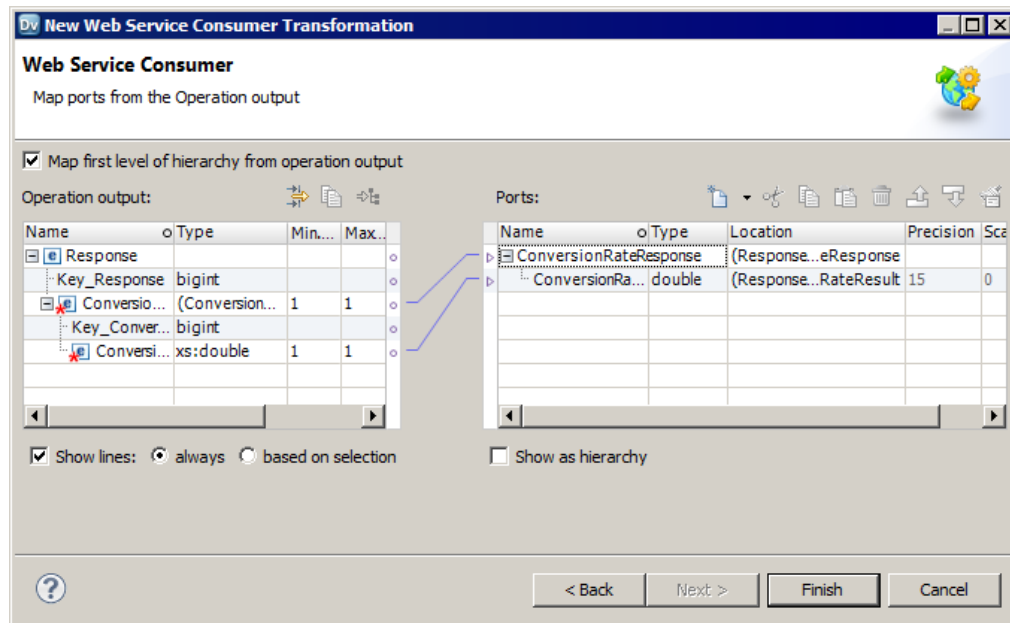
작업 입력에서 포트 매핑 화면이 나타납니다. 포트 영역에 변환 입력 그룹 및 입력 포트가 표시됩니다. 작업 입력 영역에 요청 메시지 계층이 표시됩니다.



5. 입력 그룹 및 입력 포트를 정의하고 입력 포트를 작업 입력 노드로 매핑합니다.

6. 다음을 클릭합니다.

작업 출력에서 포트 매핑 화면이 나타납니다. 작업 출력 영역에 응답 메시지 계층이 표시됩니다. 포트 영역에 변환 출력 그룹 및 출력 포트가 표시됩니다.



7. 출력 그룹 및 출력 포트를 정의하고 작업 출력 노드를 출력 포트에 매핑합니다.

8. 마침을 클릭합니다.

9. 고급 보기를 클릭하여 변환 속성 및 웹 서비스 연결을 구성합니다.

관련 항목:

- [“작업 입력에 입력 포트 매핑” 페이지 630](#)
- [“출력 포트에 작업 출력 매핑” 페이지 633](#)
- [“웹 서비스 소비자 변환 고급 속성” 페이지 634](#)

## 웹 서비스 소비자 변환 예제

회사에서는 RT100 제품군에 대한 주문 정보를 판매 조직에 공개해야 합니다. 판매 팀에서는 매일 주문 요약 및 주문 세부 정보를 쿼리해야 합니다.

이 경우 가상 테이블로 일일 주문 정보를 공개하는 논리적 데이터 개체를 작성합니다. 읽기 매핑에는 최신 RT100 주문을 반환하는 웹 서비스 소비자를 포함합니다. 웹 서비스 소비자 변환은 RT100 제품군에 대한 일일 주문 요약 및 주문 세부 정보를 반환하는 웹 서비스를 사용합니다.

### 입력 파일

입력 파일은 제품 행 번호가 포함된 플랫폼 파일입니다.

실제 데이터 개체를 작성하여 입력 파일을 정의합니다. 이 파일에는 Product\_Line 필드 1개가 포함됩니다. 필드 값은 RT100입니다. **런타임 속성** 보기에서 실제 데이터 개체의 위치를 정의합니다.

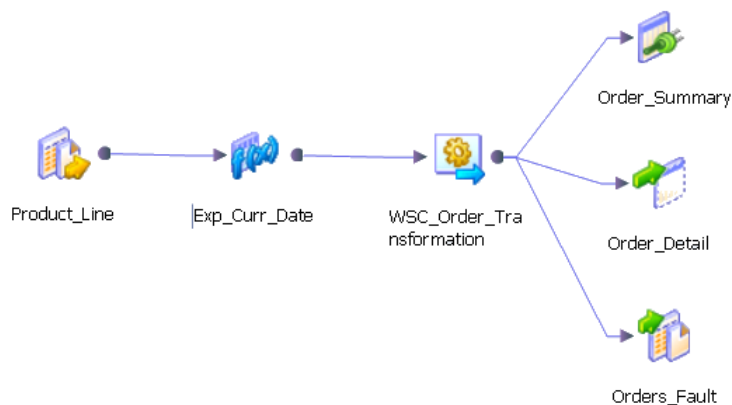
### 논리적 데이터 개체 모델

조직의 비즈니스 분석가는 주문 요약 및 주문 세부 테이블 구조를 설명하는 논리적 데이터 모델을 작성합니다. 논리적 데이터 모델에는 Order\_Summary 및 Order\_Detail 논리적 데이터 개체가 포함됩니다.

분석가는 논리적 데이터 모델을 정의하는 모델링 도구에서 스키마를 작성합니다. 사용자는 이 스키마에서 논리적 데이터 모델을 가져와 Order\_Summary 및 Order\_Detail 논리적 데이터 개체를 작성합니다.

### 논리적 데이터 개체 매핑

논리적 데이터 개체 매핑은 논리적 데이터 개체를 통해 데이터에 액세스하는 방법을 설명합니다.



읽기 매핑에는 다음 개체가 포함됩니다.

Product\_Line

제품 행 번호가 포함된 입력 플랫폼 파일입니다.

Exp\_Curr\_Date 변환

웹 서비스 소비자 변환의 루트 수준 입력 그룹에 대한 현재 날짜 및 기본 키를 반환하는 식 변환입니다.

WSC\_Order 변환

웹 서비스를 사용하여 주문 정보를 검색하는 웹 서비스 소비자 변환입니다. 변환은 제품 행과 현재 날짜를 요청 메시지로 웹 서비스에 전달합니다. 변환은 웹 서비스의 주문 정보를 응답 메시지로 수신합니다.

Order\_Summary 테이블

Order\_No, Customer\_Id, Qty 및 Order\_Date와 같은 주문 정보가 포함된 논리적 데이터 개체입니다.

Order\_Detail 테이블

Order\_No, Product\_Id, Qty 및 Status와 같은 주문 세부 정보가 포함된 논리적 데이터 개체입니다.

Orders\_Fault

일반 결함 메시지를 수신하는 출력 플랫폼 파일입니다.

## 웹 서비스 소비자 변환

웹 서비스 소비자 변환에서 제품군, 날짜 및 일련 번호를 입력으로 수신합니다. 변환에서 **Get\_Order\_Info** 웹 서비스 작업을 사용하여 주문 정보를 검색합니다.

웹 서비스 소비자 변환을 작성할 경우 요청 및 응답 웹 서비스 메시지를 설명하는 **WSDL** 데이터 개체를 선택합니다. 웹 서비스 메시지에는 **XML** 요소의 계층 그룹이 포함되어 있습니다. 요소에 다른 요소가 포함될 수 있습니다. 일부 요소는 여러 번 발생합니다. 리포지토리의 **Order\_Info WSDL** 개체에서 변환을 작성합니다.

변환 입력 포트를 구성하고 포트를 작업 입력 계층에 매핑합니다. 작업 출력 계층의 노드를 출력 포트에 매핑합니다. 웹 서비스 연결 및 런타임 속성을 정의합니다.

### 변환 입력 매핑

**포트** 보기에 입력 매핑을 표시할 경우 입력 포트를 정의하고 작업 입력의 노드에 매핑할 수 있습니다.

변환 **포트** 영역에는 루트 그룹과 주문 그룹이 있습니다. 루트 그룹이 요청 입력 그룹입니다. 포트 하나를 요청 입력 그룹에 추가하여 기본 키를 나타냅니다.

주문 그룹에는 **Select\_Date** 및 **Select\_Product\_Line** 입력 포트가 있습니다.

입력 포트를 **작업 입력** 영역의 **Order\_Date** 및 **Product\_Line** 노드에 매핑합니다.

웹 서비스 소비자 변환이 웹 서비스에 전달하는 요청 메시지를 **작업 입력** 영역에서 정의합니다. 노드는 기본적으로 **작업 입력** 영역에 표시됩니다.

### 변환 출력 매핑

**포트** 보기에서 출력 매핑을 표시할 경우 작업 출력의 노드를 변환 출력 그룹에 매핑하여 출력 포트를 정의할 수 있습니다.

웹 서비스가 웹 서비스 응답 메시지를 통해 다음 계층을 반환합니다.

```
Response
  Orders
    Order
      Key_Order
      Order_ID
      Order_Date
```

```

Customer_ID
Total_Qty
Order_Details
  Order_Detail
    Product_ID
    Description
    Qty
    Status

```

웹 서비스에서 여러 주문을 반환합니다. 주문은 주문 수준에서 다중 발생 노드입니다. 각 주문마다 웹 서비스가 여러 주문 세부 정보를 반환할 수 있습니다. **Order\_Detail**은 **Order\_Details** 수준에서 다중 발생 노드입니다.

**참고:** Developer tool이 사용자 인터페이스에서 **Key\_Order** 노드를 추가합니다. 키를 출력 그룹에 매핑하여 그룹 간 관계를 정의할 수 있습니다. 이 예제에서는 **Order\_ID**가 **Order**에서 기본 키이고 **Order\_Details**에서 외래 키입니다.

**포트** 영역에서 다음 출력 그룹을 작성합니다.

```

Order
  Order_ID
  Order_Date
  Customer_ID
  Total_Qty

Order_Detail
  Order_ID
  Product_ID
  Description
  Qty
  Status

```

**Order\_ID** 값이 변경될 때마다 데이터 통합 서비스가 **Order** 그룹의 행을 기록합니다.

**Order\_ID** 및 **Product\_ID**의 값이 변경될 때마다 데이터 통합 서비스가 **Order\_Detail** 그룹의 행을 기록합니다.

## 변환 고급 속성

웹 서비스 소비자 변환에 대한 다음 고급 속성을 구성합니다.

### 일반 SOAP 결함 처리 활성화

SOAP 결함 메시지를 수신하는 출력 포트를 추가합니다.

### 연결

웹 서비스 액세스를 위한 웹 서비스 연결을 선택합니다.

### 압축 활성화

웹 서비스 소비자 변환이 웹 메시지를 GZIP로 압축합니다.



## 제 47 장

# 웹 서비스 SOAP 메시지 구문 분석

이 장에 포함된 항목:

- [웹 서비스 SOAP 메시지 구문 분석 개요, 645](#)
- [변환 사용자 인터페이스, 645](#)
- [다중 발생 출력 구성, 646](#)
- [anyType 요소 구문 분석, 649](#)
- [파생된 유형 구문 분석, 650](#)
- [QName 요소 구문 분석, 651](#)
- [대체 그룹 구문 분석, 651](#)
- [SOAP 메시지의 XML 구성 구문 분석, 651](#)

## 웹 서비스 SOAP 메시지 구문 분석 개요

데이터 통합 서비스는 웹 서비스 변환에서 SOAP 메시지를 구문 분석할 때 행 데이터를 생성합니다.

웹 서비스 입력 변환 및 웹 서비스 소비자 변환은 SOAP 메시지를 구문 분석하는 웹 서비스 변환입니다.

SOAP 메시지를 구문 분석하는 변환을 구성하려면 SOAP 메시지 계층과 유사한 구조의 출력 포트를 작성합니다. SOAP 메시지 계층의 노드를 포트에 매핑합니다.

출력 포트의 정규화된 그룹, 비정규화된 그룹 및 포트의 피벗된 그룹을 구성할 수 있습니다. SOAP 메시지에 파생된 유형, anyType 요소 또는 대체 그룹이 포함되는 경우 SOAP 메시지 인스턴스에서 발생할 수 있는 유형에 따라 서로 다른 출력 그룹을 구성할 수 있습니다.

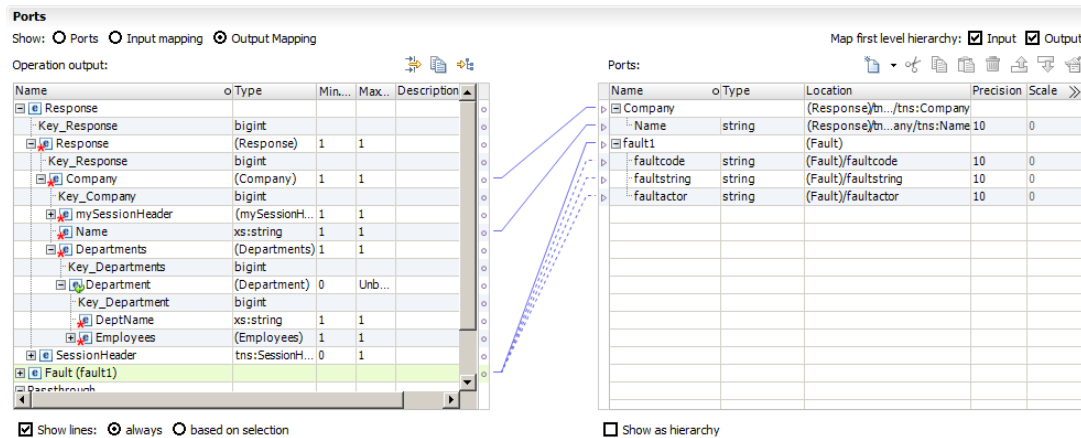
관련 항목:

- [“웹 서비스 소비자 변환 출력 매핑” 페이지 631](#)

## 변환 사용자 인터페이스

웹 서비스 소비자 변환 및 웹 서비스 입력 변환에서 SOAP 메시지의 데이터를 변환 출력 포트에 매핑하는 데 사용할 수 있는 사용자 인터페이스를 제공합니다.

다음 그림에서는 웹 서비스 소비자 변환에서 SOAP 1.1 메시지 노드 및 출력 포트 간 매핑을 보여 줍니다.



## 작업 영역

작업 영역에는 SOAP 메시지 계층이 포함되어 있습니다. 복합 노드 또는 다중 발생 노드는 구조의 계층 수준을 정의합니다. Developer tool에서 수준 간 상위-하위 관계를 정의하는 키를 수준에 추가합니다.

이전 그림에서 SOAP 메시지 계층에는 다음 수준이 있습니다.

### 응답 또는 요청

응답 또는 요청 메시지의 루트를 나타내는 수준입니다.

### 회사

요청 데이터의 최상위 수준입니다.

### 부서

회사 내 다중 발생 부서입니다.

### 직원

직원은 부서 내 복합 요소입니다.

### 결합 그룹

오류 메시지를 수신하는 결합 메시지 그룹입니다.

## 포트 영역

각 SOAP 메시지 수준의 데이터를 출력 포트에 매핑할 수 있습니다. 출력 포트의 각 그룹이 기본 외래 키 관계를 가진 다른 출력 그룹에 연결될 수 있습니다.

이전 그림에서 변환에는 SOAP 메시지의 노드 그룹에 해당하는 출력 포트 그룹이 있습니다.

# 다중 발생 출력 구성

입력 변환 또는 웹 서비스 소비자 변환에서 다중 발생 데이터를 반환할 경우 출력 포트를 서로 다른 구성으로 구성할 수 있습니다.

정규화된 출력 데이터, 피벗된 출력 데이터 또는 비정규화된 출력 데이터를 구성할 수 있습니다.

예를 들어 SOAP 메시지에 Department 및 Employee 복합 요소가 포함되어 있습니다. 각 Department에는 다수의 Employee가 포함됩니다. Department는 Employee의 상위 요소입니다.

SOAP 메시지에는 다음 요소 계층이 포함됩니다.

```
Departments
  Department_ID
  Department_Name
Employees
  Employee_ID
  Employee_Name
```

## 정규화된 관계형 출력

정규화된 출력 데이터를 생성할 경우 데이터 값이 출력 그룹에서 반복되지 않습니다. SOAP 메시지의 계층 수준과 포트의 출력 그룹 사이에 일대일 관계를 작성합니다.

SOAP 메시지에 **Department** 상위 계층 수준과 **Employee** 하위 계층 수준이 포함될 경우 다음 포트 그룹을 작성할 수 있습니다.

```
Departments
  Department_Key
  Department_ID
  Department_Name

Employees
  Department_Key
  Employee_ID
  Employee_Name
```

Department\_Key는 Employee 출력 그룹을 Department 그룹에 연결하는 생성된 키입니다.

## 생성된 키

출력 그룹을 추가하면 Developer tool에서 생성된 키가 있는 다른 출력 그룹에 해당 출력 그룹을 연결합니다. Developer tool에서는 상위 그룹과 하위 그룹에 **bigint** 키를 추가합니다. 런타임 시 데이터 통합 서비스에서는 생성된 키에 대한 키 값을 작성합니다.

예

SOAP 계층에는 다음 노드가 포함됩니다.

```
Departments
  Dept_Key
  Dept_Num
  Dept_Name

Employees
  Dept_FK
  Employee_Num
  Employee_Name
```

Departments에 대한 포트의 출력 그룹을 작성하는 경우 Departments 노드를 포트 영역의 빈 필드에 매핑해야 합니다. Developer tool은 다음 출력 그룹을 작성합니다.

```
Departments
  Dept_Num
  Dept_Name
```

Employees 노드를 Ports 영역의 빈 필드에 매핑하면 Developer tool에서 Employees 그룹을 Departments 그룹에 연결하라는 메시지를 표시합니다. Employees 그룹을 둘 이상의 그룹에 연결할 수 있습니다. Developer tool이 각 그룹에 키를 추가합니다.

Developer tool은 다음 그룹 및 생성된 키를 작성합니다.

```
Departments
  Key_Departments
```

```

Dept_Num
Dept_Name

Employees
  Key_Departments
    Employee_Num
    Employee_Name

```

**참고:** 노드를 생성된 키에 매핑할 필요는 없습니다. 데이터 통합 서비스에서 런타임 시 키 값을 작성합니다.

Developer tool은 하나의 출력 그룹에 여러 수준으로 생성된 키를 작성할 수 있습니다. Employees 그룹에는 다음 포트가 포함될 수 있습니다.

```

Employees
  Key_Employees
  Key_Departments
  Key_Managers
  Employee_Num
  Employee_Name

```

Key\_Departments 및 Key\_Managers는 상위 그룹을 가리키는 생성된 키입니다. Key\_Employees는 Employees 그룹에 대한 생성된 키입니다. Key\_Employees는 하위 그룹을 Employees 그룹에 연결할 때 나타냅니다.

## 비정규화된 관계형 출력

관계형 출력을 비정규화할 수 있습니다. 출력 데이터를 비정규화하는 경우 상위 그룹의 요소 값이 각 하위 요소에 대해 반복됩니다.

출력 데이터를 비정규화하려면 상위 계층 수준의 노드를 출력 포트의 하위 그룹에 매핑합니다.

다음 예제는 Employees 출력 그룹의 Department\_ID 및 Department\_Name을 보여 줍니다.

```

Employees
  Department_ID
  Department_Name
  Employee_ID
  Employee_Name

```

Department\_ID 및 Department\_Name은 부서의 각 직원에 대해 반복됩니다.

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

## 피벗된 관계형 출력

여러 번 발생하는 요소를 특정 개수만큼 출력 그룹에 포함할 수 있습니다.

여러 번 발생하는 요소를 피벗하려면 여러 번 발생하는 하위 요소를 출력 포트의 상위 그룹에 매핑합니다.

Developer tool에서 상위에 포함할 하위 요소 수를 정의하라는 메시지가 표시됩니다.

다음 예제는 Departments 상위 그룹의 두 Employee\_ID 인스턴스를 보여 줍니다.

```

Departments
  Department_ID
  Department_Name
  Employee_ID1
  Employee_ID2

```

# anyType 요소 구문 분석

anyType 요소는 WSDL 또는 스키마에서 모든 글로벌 유형의 선택을 나타냅니다. Developer tool에서 노드를 포트에 매핑할 때 anyType 요소에 대한 SOAP 메시지에 표시할 유형을 선택해야 합니다. SOAP 메시지의 anyType 요소를 복합 유형 또는 xs:string으로 바꿔야 합니다. 선택한 각 유형에 대한 포트 그룹을 작성합니다.

데이터를 출력 포트에 매핑하려면 유형을 선택해야 합니다. WSDL 또는 스키마에 글로벌 유형이 포함되지 않는 경우 Developer tool이 anyType 요소를 xs:string으로 바꿉니다.

작업 영역에서 요소 유형을 선택하려면 anyType 요소의 **유형** 열에서 **선택**을 클릭합니다. 사용 가능한 복합 유형 및 xs:string의 목록이 나타납니다.

anyType 요소를 파생된 유형으로 바꿀 경우 데이터 통합 서비스에서는 한 번에 한 유형에 대한 요소를 채웁니다. SOAP 메시지에는 기본 유형 및 파생된 유형의 데이터가 동시에 포함되지 않습니다.

## 파생된 유형 예제

WSDL에 anyType 요소가 포함되어 있습니다. 이 요소를 AddressType과 파생된 유형 USAddressType으로 바꿉니다. SOAP 메시지 계층에는 다음 그룹이 포함됩니다.

```
Address:AddressType (base type)
  Address: AddressType
    Street
    City

Address:USAddressType (derived type)
  Street
  City
  State
  ZipCode
```

SOAP 메시지에는 다음 데이터가 포함됩니다.

```
<address xsi: type ="AddressType">
<street>1002 Mission St.</street>
<city>san jose</city>
</address>

<address xsi:type="USAddressType">
<street>234 Fremont Blvd</street>
<city>Fremont</city>
<zip>94556</zip>
<state>CA</state>
</address>
```

데이터 통합 서비스는 xsi: AddressType에 대해 하나의 행을 반환합니다.

Street	City
1002 Mission St.	San Jose

데이터 통합 서비스는 파생된 유형 xsi: USAddressType에 대해 하나의 행을 반환합니다.

Street	City	State	Zip
234 Fremont Blvd.	Sunnyvale	CA	94556

유형이 xsi: USAddressType일 경우 데이터 통합 서비스가 AddressType을 채우지 않습니다.

## 파생된 유형 구문 분석

파생된 유형이 포함된 SOAP 메시지를 구문 분석할 수 있습니다. SOAP 메시지의 데이터를 수신하는 포트를 정의할 때는 SOAP 메시지에 표시될 수 있는 유형을 선택합니다. 선택한 유형의 요소에 따라 작성해야 하는 포트가 결정됩니다.

예를 들어 WSDL에는 AddressType과 이름이 USAddressType인 파생된 유형이 포함될 수 있습니다. Developer tool의 작업 영역에서 다음 그룹을 작성할 수 있습니다.

```
Address
  Address: AddressType
    Street
    City

  Address:USAddressType
    Street
    City
    State
    ZipCode
```

SOAP 메시지에는 다음 데이터가 포함될 수 있습니다.

```
<address>
<street>1002 Mission St.</street>
<city>san jose</city>
</address>

<address xsi:type="USAddressType">
<street>234 Fremont Blvd</street>
<city>Fremont</city>
<zip>94556</zip>
<state>CA</state>
</address>

<address xsi:type="USAddressType">
<street>100 Cardinal Way</street>
<city>Redwood City</city>
<zip>94536</zip>
<state>CA</state>
</address>

<address>
<street>100 El Camino Real</street>
<city>Sunnyvale</city>
</address>
```

데이터 통합 서비스는 기본 유형인 Address에 대해 다음 행을 반환합니다.

Street	City
1002 Mission St.	San Jose
234 Fremont Blvd	Sunnyvale
100 Cardinal Way	Redwood City
100 El Camino Real	Sunnyvale

데이터 통합 서비스는 파생된 유형인 USAddress에 대해 다음 행을 반환합니다.

Street	City	State	Zip
234 Fremont Blvd.	Sunnyvale	CA	94556
100 Cardinal Way	Redwood City	CA	94536

데이터 통합 서비스는 기본 유형의 모든 주소를 반환합니다. 데이터 통합 서비스는 파생된 유형의 모든 미국 주소를 반환합니다. 파생된 유형에는 USAddressType이 기본 유형으로부터 상속한 Street 및 City 요소가 포함됩니다.

## QName 요소 구문 분석

데이터 통합 서비스는 SOAP 메시지의 QName 요소를 구문 분석할 때 스키마에 정의된 네임스페이스 접두사를 사용하기 위해 스키마의 네임스페이스에 속하는 QName 값을 업데이트합니다. 이 외에는 요소의 값을 업데이트하지 않습니다.

예를 들어 네임스페이스 "http://user/test"에 대해 정의된 네임스페이스 접두사 tns가 스키마에 있습니다. SOAP 메시지에는 동일한 네임스페이스에 대해 정의된 네임스페이스 접두사 mytns가 있습니다. 데이터 통합 서비스는 QName 값 mytns:myelement를 구문 분석할 때 값을 tns:myElement로 변경합니다.

데이터 통합 서비스는 SOAP 메시지에 QName 요소를 생성할 때 요소의 값을 업데이트하지 않습니다.

## 대체 그룹 구문 분석

대체 그룹은 한 요소를 동일한 그룹의 다른 요소로 바꿉니다. 대체 그룹은 각 요소 정의에 대체 그룹 이름이 포함 된다는 점을 제외하고 파생된 유형과 유사합니다.

대체 그룹의 특정 유형으로부터 요소를 수신하는 포트의 출력 그룹을 구성할 수 있습니다. 대체 그룹의 다른 유형으로부터 요소를 수신하는 포트의 출력 그룹을 여러 개 작성할 수 있습니다.

## SOAP 메시지의 XML 구성 구문 분석

SOAP 메시지에 선택, 목록 및 합집합 요소 같은 XML 구성이 포함될 수 있습니다.

웹 서비스 변환은 일부 제한이 있지만 이 구성을 가진 SOAP 메시지를 구문 분석할 수 있습니다.

### 선택 요소

선택 요소는 하위 요소를 <선택> 선언의 요소 중 하나로 제한합니다.

다음 텍스트는 직원 또는 계약자인 사람 요소를 나타냅니다.

```
<xs:element name="person">
  <xs:complexType>
    <xs:choice>
      <xs:element name="employee" type="employee"/>
      <xs:element name="contractor" type="contractor"/>
    </xs:choice>
  </xs:complexType>
</xs:element>
```

다음 방법을 사용하여 선택 요소를 매핑할 수 있습니다.

- 각 선택 요소에 대한 출력 포트를 출력 그룹에 작성합니다. 일부 요소의 경우 출력 행에 null 값이 포함됩니다.
- 각 선택에 대한 출력 그룹을 작성합니다. 위의 예의 경우 직원 그룹과 계약자 그룹을 작성합니다. 데이터 통합 서비스에서는 SOAP 메시지에 나타나는 요소를 바탕으로 행을 생성합니다.

## 목록 요소

목록은 "Monday Tuesday Wednesday"와 같이 여러 단순 유형 값을 포함할 수 있는 XML 요소입니다.

데이터 통합 서비스에서는 목록을 문자열 값으로 반환할 수 있습니다. SOAP 메시지에 목록이 포함될 경우 목록의 항목을 개별 출력 행으로 매핑할 수 없습니다. 매핑에서 분리해야 하는 경우 목록의 요소를 분리하는 식 변환을 구성할 수 있습니다.

## 합집합 요소

합집합 요소는 둘 이상의 유형이 결합된 단순 유형입니다.

다음 텍스트는 단순 유형, `size_no` 및 `size_string`의 합집합인 크기 요소를 보여 줍니다.

```
<xs:element name="Size">
  <xs:simpleType>
    <xs:union memberTypes="size_no size_string" />
  </xs:simpleType>
</xs:element>
```

크기를 출력 포트에 매핑하려면 크기에 대한 포트 하나를 작성합니다. 출력 포트를 문자열로 구성합니다. 데이터를 다른 유형으로 변환하도록 매핑에서 다른 변환을 구성할 수 있습니다.



## 제 48 장

# 웹 서비스 SOAP 메시지 생성

이 장에 포함된 항목:

- [웹 서비스 SOAP 메시지 생성 개요, 653](#)
- [변환 사용자 인터페이스, 654](#)
- [포트 및 계층 수준 관계, 655](#)
- [키, 656](#)
- [포트 매핑, 657](#)
- [여러 번 발생하는 포트 피벗, 659](#)
- [비정규화된 데이터 매핑, 660](#)
- [파생된 유형 및 요소 대체, 661](#)
- [SOAP 메시지의 XML 구성 생성, 663](#)

## 웹 서비스 SOAP 메시지 생성 개요

데이터 통합 서비스에서는 SOAP 메시지를 생성할 때 입력 데이터 그룹으로부터 XML 데이터를 생성합니다. 웹 서비스 소비자 변환, 웹 서비스 출력 변환 또는 결합 변환을 작성할 때는 SOAP 메시지 계층에 매핑할 입력 포트를 구성해야 합니다.

SOAP 메시지를 생성하도록 변환을 구성하려면 입력 포트의 그룹을 작성하고 각 그룹을 SOAP 메시지 계층의 그룹에 매핑합니다. SOAP 메시지의 구조는 WSDL 또는 스키마를 통해 정의됩니다.

비정규화된 입력 데이터를 바탕으로 SOAP 메시지의 데이터 그룹을 구성할 수 있습니다. 또한 다중 발생 입력 데이터를 SOAP 메시지의 다중 발생 노드에 피벗할 수도 있습니다.

데이터를 SOAP 메시지의 파생된 유형, anyType 요소 또는 대체 그룹에 매핑할 수 있습니다. 변환을 정의할 때 SOAP 메시지에서 발생할 수 있는 유형을 선택해야 합니다. 선택한 유형에 따라 작성해야 하는 입력 포트가 결정됩니다.

Developer tool에서 SOAP 메시지 계층을 보면 계층에 키가 포함됩니다. 키는 SOAP 메시지에 표시되지 않습니다. 키는 데이터 통합 서비스에서 SOAP 메시지에 있는 그룹 간의 상위-하위 관계를 정의할 때 사용됩니다. 키 값을 구성하려면 입력 데이터를 SOAP 메시지의 키에 매핑합니다.

관련 항목:

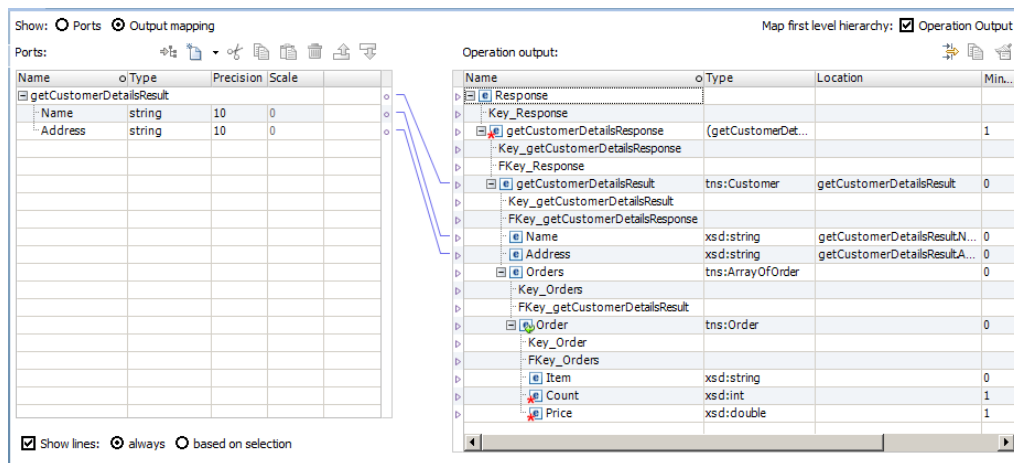
- “웹 서비스 소비자 변환 입력 매핑” 페이지 628

## 변환 사용자 인터페이스

웹 서비스 출력 변환, 결합 변환 및 웹 서비스 소비자 변환에는 SOAP 메시지를 구성하는 데 사용하는 사용자 인터페이스가 포함되어 있습니다.

SOAP 메시지를 생성하도록 변환을 구성하려면 SOAP 메시지 계층과 유사한 구조로 입력 포트를 작성하십시오. WSDL 또는 스키마에서 계층의 구조를 정의합니다. 각 입력 포트를 SOAP 메시지의 노드에 매핑합니다.

다음 그림에서는 웹 서비스 출력 변환에서 입력 포트 및 SOAP 메시지 노드 간의 매핑의 보여 줍니다.



## 입력 포트 영역

**입력 포트** 영역에서 입력 포트의 그룹을 작성합니다. 매핑해야 하는 SOAP 메시지 계층의 각 수준에 대한 입력 포트를 포함합니다.

응답 또는 요청 입력 그룹과 데이터를 수신하는 하위 그룹을 작성해야 합니다.

입력 포트 그룹을 작성할 때 각 상위 그룹의 기본 키를 정의합니다. 각 하위 그룹의 외래 키를 정의합니다. 외래 키는 그룹을 상위 그룹에 연결합니다.

응답 수준 또는 WSDL 루트 수준에 대한 키는 WSDL 루트 수준에서 데이터를 전달하는 경우를 제외하고 정의할 필요가 없습니다. 예를 들어 루트 수준에는 HTTP 헤더가 포함될 수 있습니다.

고객 및 주문에 대한 다음 그룹과 유사한 포트 그룹을 작성할 수 있습니다.

```
Response
  Response_Key

Customer_Details_Root
  Key_Cust_Det
  FK_Response_Key

Customer
  Customer_ID
  FK_Cust_Det
  Name
```

```

Address
  Orders
    Order_Num
    FK_Cust_ID
  Order_Items
    Order_Num
    Item
    Count
    Price

```

## 작업 영역

**작업** 영역에는 WSDL 또는 스키마를 통해 정의된 SOAP 메시지 계층의 요소가 표시됩니다. WSDL 또는 스키마의 모든 요소가 SOAP 메시지에 포함될 필요는 없습니다. 메시지는 사용자가 입력 포트에서 매핑한 데이터가 포함됩니다.

다중 발생 노드 및 복합 노드는 SOAP 메시지 구조의 계층 수준을 정의합니다. Developer tool은 키를 수준에 추가하여 수준 간의 상위-하위 관계를 작성합니다. 리프 수준을 제외한 계층의 모든 수준에는 기본 키가 있습니다. 각 하위 수준에는 상위 수준에 대한 외래 키가 있습니다. SOAP 메시지 계층에 나타나는 키는 SOAP 메시지 인스턴스에는 나타나지 않습니다. 데이터 통합 서비스에서는 SOAP 메시지를 생성할 때 키의 값을 사용하여 데이터의 수준을 연결합니다.

**위치** 열에는 SOAP 메시지의 요소에 대한 데이터가 포함된 그룹 이름 및 입력 포트가 포함됩니다. **위치** 열은 입력 포트를 노드에 매핑하기 전까지 비어 있습니다.

이전 그림의 SOAP 메시지에는 단일 인스턴스의 고객 세부 정보 및 주문이 포함되어 있습니다. **Orders** 그룹에는 이름이 **Order**인 다중 발생 요소가 포함되어 있습니다. SOAP 메시지 계층에는 키로 연결된 다음 수준이 포함됩니다.

```

Response
  GetCustomerDetailsResponse
    GetCustomerDetailsResult
      Orders
        Order

```

응답 수준은 응답 메시지의 루트를 나타냅니다. 데이터 통합 서비스는 이 수준을 사용하여 헤더를 SOAP 메시지에 연결합니다.

GetCustomerDetailsResponse 수준은 메시지의 루트입니다.

## 포트 및 계층 수준 관계

입력 포트를 SOAP 메시지 계층에 매핑하는 경우 입력 그룹과 SOAP 메시지 계층 수준 간의 관계가 유지 관리됩니다. 예를 들어 **Department**와 **Employee**라는 입력 그룹 2개가 있습니다.

**Department** 입력 그룹은 다음 행을 수신합니다.

Dept_num	Name	Location
101	HR	New York
102	Product	California

**Employee** 입력 그룹은 다음 행을 수신합니다.

Dept_num	Employee
101	Alice

Dept_num	Employee
101	Bob
102	Carol
102	Dave

Employee 그룹의 부서 번호를 Department 그룹과 Employee 그룹 간의 관계를 설정하는 외래 키로 매핑합니다. 부서 번호는 부서 계층 수준에서 발생하지만 직원 수준에서는 발생하지 않습니다.

SOAP 메시지에는 다음 XML 구조가 포함됩니다.

```
<department>
  <dept_num>101</dept_num>
  <name>HR</name>
  <location>New York</location>

  <employee>
    <name>Alice</name>
  </employee>

  <employee>
    <name>Bob</name>
  </employee>
</department>

<department>
  <dept_num>102</dept_num>
  <name>Product</name>
  <location>California</location>

  <employee>
    <name>Carol</name>
  </employee>

  <employee>
    <name>Dave</name>
  </employee>
</department>
```

## 키

SOAP 메시지 계층에는 키가 포함됩니다. 데이터 통합 서비스에서 SOAP 메시지에 XML 계층을 구성하려면 키 값이 필요합니다.

입력 포트 데이터를 SOAP 메시지 계층의 키에 매핑해야 합니다. 데이터를 제공하는 각 수준의 키에 데이터를 매핑합니다. 다중 발생 노드가 있는 경우 노드를 상위 그룹에 연결해야 합니다.

키는 유형 없이 SOAP 메시지에 나타납니다. 키에 매핑하는 포트는 문자열, 정수 또는 **bigint** 데이터 유형이어야 합니다. 상위 그룹의 기본 키와 각 하위 그룹의 외래 키는 데이터 유형, 전체 자릿수 및 소수 자릿수가 같아야 합니다. 생성된 키를 SOAP 메시지 키에 매핑할 수 있습니다.

포트를 노드에 매핑하고 동일한 계층 수준의 키에 매핑할 수 있습니다. 예를 들어 **Employee\_ID**를 SOAP 메시지의 노드에 매핑하고 **Employee** 수준의 키에 매핑할 수 있습니다.

계층의 그룹 노드 2개가 상위-하위 관계에 있는 경우 다음 태스크를 완료하십시오.

- 포트를 상위 노드 그룹의 기본 키에 매핑합니다.
- 포트를 하위 노드 그룹의 외래 키에 매핑합니다.

기본 키를 입력 포트에 매핑하여 기본 키가 null이거나 기본 키가 중복된 행을 제거할 수도 있습니다.

여러 포트를 동일한 키에 매핑하여 SOAP 메시지에 복합 키를 작성할 수 있습니다. 복합 키는 데이터를 비정규화하고 일부 다중 발생 값 조합에 대해 고유한 키를 유지 관리해야 하는 경우에 사용됩니다. 문자열, bigint 또는 정수 값을 포함하는 복합 키를 작성할 수 있습니다.

**참고:** 식 변환을 작업 매핑에 포함하여 키 값을 생성할 수 있습니다.

### 복합 키 예

다음 포트 그룹에서 고유한 Division-Department 키를 구성합니다.

```
Company
  Company_Num
  Company_Name

  Division
    Company_Num
    Division_Num
    Division_Name

    Department
      Division_Num
      Dept_Num
      Dept_Name
      Location
```

Dept\_Num은 Division 내에서 고유하지만 Company의 모든 Division에 대해 고유하지는 않습니다.

Division 및 Department 정보를 포함하는 Department 그룹을 구성할 수 있습니다. 그러면 Division 번호와 Department 번호를 복합 키의 일부로 구성합니다.

```
Department
  Division_Num + Dept_Num (key)
  Dept_Name
  Location
```

키 값은 포트를 매핑하는 순서에 따라 결정됩니다.

## 포트 매핑

입력 포트를 작성한 후에 각 입력 포트를 SOAP 메시지 계층에 매핑합니다. 포트의 위치는 작업 영역에서 노드 옆에 표시됩니다.

다음 유형의 노드에 포트를 매핑할 수 있습니다.

### 원자성 노드

하위 항목이 없고 나눌 수 없는 단순 요소 또는 특성입니다.

### 다중 발생 원자성 노드

계층의 동일한 위치에서 여러 번 발생하는 단순 요소 또는 특성입니다.

### 복합 노드

다른 요소가 포함된 요소입니다.

상위 노드에 위치가 없는 경우 상위 노드가 입력 그룹 이름을 위치로 수신합니다. 상위 노드에 위치가 있는 경우 계층 수준의 각 노드에는 동일한 위치의 출력 위치가 있어야 합니다.

입력 그룹 이름을 계층 수준의 상위 노드에 매핑할 수 있습니다. **Developer tool**이 계층의 상위 노드에 대한 위치 필드를 업데이트합니다. **Developer tool**은 계층의 그룹에 속하는 하위 노드를 업데이트하지 않습니다. 입력 포트를 하위 노드에 매핑할 경우 각 입력 포트 위치는 위치 상위 노드와 동일한 위치여야 합니다.

입력 그룹을 계층 수준에 매핑한 후 나중에 변경할 수 있습니다. **지우기**를 클릭하거나 포트와 작업 영역 간의 행을 삭제할 수 있습니다. 행을 삭제하려면 행의 포인터를 끌어 선택합니다. **삭제**를 클릭합니다.

## 포트 매핑

포트를 SOAP 메시지의 노드에 매핑할 경우 포트를 매핑한 노드의 유형에 따라 **Developer tool**에서 제공하는 결과가 달라집니다.

다음 테이블에는 **작업** 영역에서 단일 포트를 서로 다른 대상 노드에 매핑할 때의 결과가 설명되어 있습니다.

대상 노드	결과
원자성 노드	단일 포트를 노드에 매핑하고 상위 노드에 위치가 없는 경우 노드가 포트의 위치를 수신합니다. 상위 노드 위치는 단일 포트에 대한 입력 그룹의 위치를 수신합니다. 단일 포트를 노드에 매핑하고 상위 노드에 이미 위치가 있는 경우 상위 노드의 위치를 변경하고 동일한 수준의 다른 하위 노드에 대한 위치를 지울 수 있습니다. 계층 수준 위치는 포트의 그룹 이름으로 변경됩니다.
다중 발생 원자성 노드 또는 다중 발생 원자성 노드의 기본 키	단일 포트를 다중 발생 원자성 노드에 매핑할 경우 <b>Developer tool</b> 이 원자성 노드의 위치를 선택한 포트의 그룹으로 설정합니다.
복합 노드	단일 포트를 복합 노드에 매핑할 경우 <b>Developer tool</b> 이 복합 노드의 위치를 포트가 포함된 그룹의 위치로 설정합니다. 단일 발생 원자성 노드에 포트를 할당하라는 메시지가 표시됩니다. 모든 단일 발생 원자성 노드에 위치가 있는 경우 복합 노드를 매핑할 수 없습니다.

## 그룹 매핑

입력 그룹을 SOAP 메시지의 노드에 매핑할 경우 포트를 매핑한 노드의 유형에 따라 **Developer tool**에서 제공하는 결과가 달라집니다.

다음 테이블에는 **작업** 영역에서 그룹을 노드에 매핑할 때의 결과가 설명되어 있습니다.

대상 노드	결과
원자성 노드	원자성 노드에는 그룹을 매핑할 수 없습니다.
다중 발생 원자성 노드	입력 그룹의 포트를 선택하여 노드 및 기본 키의 위치를 업데이트해야 합니다.
다중 발생 복합 노드	<b>Developer tool</b> 이 복합 노드의 위치를 그룹의 위치로 설정합니다.

## 여러 포트 매핑

여러 포트를 SOAP 메시지의 노드에 매핑할 경우 포트를 매핑한 노드의 유형에 따라 Developer tool에서 제공하는 결과가 달라집니다. 동일한 그룹에서 매핑하는 경우 여러 포트를 동시에 매핑할 수 있습니다.

다음 테이블에는 여러 포트를 노드에 매핑할 때의 결과가 설명되어 있습니다.

대상 노드	결과
단일 원자성 노드	여러 포트를 단일 포트에 매핑할 때는 <b>작업</b> 영역에서 단일 원자성 노드 2개 이상에 대한 위치를 업데이트해야 합니다. 계층의 수준에 업데이트할 충분한 노드가 없는 경우 Developer tool이 사용 가능한 노드에 대한 포트만 매핑합니다.
다중 발생 원자성 노드	여러 포트를 다중 발생 원자성 노드에 매핑할 경우 포트를 노드의 다중 발생에 피벗해야 합니다. Developer tool은 매핑한 포트 수에 따라 노드의 인스턴스를 작성합니다. 연결한 포트의 수를 설명하는 메시지가 표시됩니다.
다중 발생 복합 노드	여러 포트를 복합 노드에 매핑할 경우 업데이트할 단일 발생 노드 원자성 노드를 선택해야 합니다. 또한 포트를 노드의 다중 발생으로 피벗해야 합니다. Developer tool은 매핑한 포트 수에 따라 노드의 인스턴스를 작성합니다.

## 여러 번 발생하는 포트 피벗

여러 입력 포트를 SOAP 메시지의 여러 번 발생하는 노드에 매핑할 수 있습니다. Developer tool이 입력 데이터를 SOAP 메시지의 여러 노드에 피벗합니다.

피벗할 요소의 수를 변경하려면 **매핑 옵션** 대화 상자에서 **기존 피벗 재정의**를 선택합니다.

**포트** 영역에서 피벗된 포트 인스턴스 중 하나를 제거할 경우 Developer tool이 **작업** 영역에서 모든 인스턴스를 제거합니다.

### 피벗 예제

입력 그룹에 다음 행이 포함될 수 있습니다.

Num	Name	Location	emp_name1	emp_name2	emp_name3
101	HR	New York	Alice	Tom	Bob
102	Product	California	Carol	Tim	Dave

각 행에는 부서 번호와 세 명의 직원 이름이 포함되어 있습니다.

Employee는 SOAP 메시지 계층에서 여러 번 발생하는 노드입니다. Employee의 모든 인스턴스를 입력 행에서 SOAP 메시지 계층에 매핑할 수 있습니다. Employee의 모든 발생을 선택합니다. **매핑**을 클릭합니다. **매핑 옵션** 대화 상자에 목록에서 노드를 선택하라는 메시지가 표시됩니다.

Developer tool이 SOAP 메시지 계층의 여러 이름 노드를 포함하도록 Employee 노드를 변경합니다.

```

Department
  num
  name
  location
  Employee (unbounded)
    emp_name1
    emp_name2
    emp_name3
  
```

SOAP 메시지는 다음 계층을 반환합니다.

```
<department>
  <num>101</num>
  <name>HR</name>
  <location>New York</location>
  <employee>
    <emp_name>Alice</name>
  </employee>
  <employee>
    <emp_name>Tom</name>
  </employee>
  <employee>
    <emp_name>Bob</name>
  </employee>
</department>

<department>
  <num>102</num>
  <name>Product</name>
  <location>California</location>
  <employee>
    <emp_name>Carol</name>
  </employee>
  <employee>
    <emp_name>Tim</name>
  </employee>
  <employee>
    <emp_name>Dave</name>
  </employee>
</department>
```

## 비정규화된 데이터 매핑

비정규화된 데이터를 매핑하고 SOAP 메시지의 비정규화된 노드에 전달할 수 있습니다.

비정규화된 데이터를 매핑할 때는 하나의 입력 그룹의 데이터를 SOAP 메시지 계층의 여러 노드에 전달합니다. 다음 관계와 유사한 유형의 그룹 관계를 SOAP 메시지에 작성할 수 있습니다.

### 선형 노드 관계

노드 A는 노드 B의 상위 노드입니다. 노드 B는 노드 C의 상위 노드입니다. 노드 C는 노드 D의 상위 노드입니다.

### 계층 노드 관계

노드 A는 노드 B의 상위 노드입니다. 노드 A는 또한 노드 C의 상위 노드입니다. 노드 B와 노드 C는 관계가 없습니다.

다음 테이블에는 비정규화된 Division 및 Department 데이터가 포함된 입력 행이 표시되어 있습니다.

Division	Dept_Num	Dept_Name	Phone	Employee_Num	Employee_Name
01	100	Accounting	3580	2110	Amir
01	100	Accounting	3580	2113	Robert
01	101	Engineering	3582	2114	Stan
01	101	Engineering	3582	2115	Jim
02	102	Facilities	3583	2116	Jose

입력 데이터에는 고유한 직원 번호와 이름이 포함됩니다. Division 및 Department 데이터는 동일한 Division 및 Department의 각 직원에 대해 반복됩니다.



## 선형 그룹 관계

포트를 구성할 때 **Division**, **Department** 및 **Employee**에 대한 개별 그룹을 구성할 수 있습니다. **Division**은 **Department**의 상위 그룹이고 **Department**는 **Employee**의 상위 그룹입니다. 다음과 같은 선형 구조로 그룹을 구성할 수 있습니다.

```
Division
  Division_Key
  Division_Num
  Division Name

  Department
    Department_Key
    Division_FKey
    Dept_Num
    Dept_Name
    Phone

    Employee
      Department_Fkey
      Employee_Num
      Employee_Name
```

**Division\_Num** 및 **Dept\_Num**이 입력 데이터에서 반복되지만 SOAP 메시지에는 **Division** 및 **Department**의 고유한 인스턴스가 포함됩니다. **Division\_Num**을 **Division** 그룹의 기본 키로 정의합니다. **Dept\_Num**을 **Department** 그룹의 기본 키로 정의합니다.

## 계층 그룹 관계

**Division** 상위 그룹과 **Department** 및 **Employee** 하위 그룹이 포함된 그룹 계층을 작성할 수 있습니다. **Department**와 **Employee**에는 기본 키-외래 키 관계가 없습니다. **Department** 및 **Employee**는 **Division**의 하위 그룹입니다. 다음과 같은 구조로 그룹을 구성할 수 있습니다.

```
Division
  Division_Key
  Division_Num
  Division_Name

  Department
    Division_FKey
    Dept_Num
    Dept_Name

  Employee
    Division_FKey
    Employee_Num
    Employee_Name
```

# 파생된 유형 및 요소 대체

입력 포트를 SOAP 메시지의 파생된 복합 유형, **anyType** 요소 및 대체 그룹에 매핑할 수 있습니다. SOAP 메시지에는 기본 유형 및 파생된 유형의 요소가 포함될 수 있습니다.

유형 관계에서 기본 유형은 다른 유형을 파생하는 유형입니다. 파생된 유형은 기본 유형의 요소를 상속받습니다. 확장된 복합 유형은 기본 유형에서 요소를 상속받는 파생된 유형이며 추가 요소를 포함합니다. 제한된 복합 유형은 기본 유형에서 일부 요소를 제한하는 파생된 유형입니다.

## 파생된 유형 생성

WSDL 또는 스키마에 파생된 유형이 포함될 경우 SOAP 메시지에 포함할 유형을 선택해야 합니다.

예를 들어 WSDL에서 기본 유형인 `AddressType`을 정의합니다. WSDL에는 파생된 `AddressType`인 `USAddressType` 및 `UKAddressType`도 포함됩니다.

각 유형에는 다음 요소가 포함됩니다.

- `AddressType`: `street`, `city`
- `USAddressType`(`AddressType` 확장): `state`, `zipCode`
- `UKAddressType`(`AddressType` 확장): `postalCode`, `country`

작업 영역에서 `USAddressType`을 선택하면 Developer tool이 `USAddressType` 요소에 대한 그룹을 SOAP 메시지에 작성합니다. 이 그룹에는 기본 주소의 `street` 및 `city`와 `USAddress`의 `state` 및 `zipCode`가 포함됩니다. 기본 유형을 확장하는 파생된 유형에는 항상 기본 유형의 요소가 포함됩니다.

SOAP 메시지의 사용 가능한 모든 파생된 유형을 선택할 경우 Developer tool은 다음과 유사한 그룹을 SOAP 계층에 작성합니다.

```
Address
  Address: Address
    Street
    City

  Address:USAddressType
    Street
    City
    State
    ZipCode

  Address: UKAddressType
    Street
    City
    PostalCode
    Country
```

사용자는 `Address`, `USAddress` 및 `UKAddress`에 대한 입력 포트 그룹을 정의해야 합니다.

## anyType 요소 및 특성 생성

일부 스키마 요소 및 특성의 경우 SOAP 메시지에서 모든 데이터 유형을 사용할 수 있습니다.

`anyType` 요소는 전역적으로 알려진 모든 유형의 선택을 나타냅니다. SOAP 메시지의 `anyType` 요소에 포트를 매핑하기 전에 사용 가능한 복합 유형 또는 `xs:string`을 선택하십시오. WSDL 또는 스키마에 복합 유형이 포함되지 않을 경우 Developer tool이 `anyType` 요소 유형을 `xs:string`으로 바꿉니다.

작업 영역에서 요소 유형을 선택하려면 `anyType` 요소의 **유형** 열에서 **선택**을 클릭합니다. 사용 가능한 복합 유형 및 `xs:string`의 목록이 나타납니다.

다음 요소 및 특성의 경우 데이터의 모든 유형을 사용할 수 있습니다.

### anyType 요소

요소가 연결된 XML 파일의 모든 데이터 유형이 될 수 있습니다.

### anySimpleType 요소

요소가 연결된 XML 파일의 모든 `simpleType`이 될 수 있습니다.

### ANY 콘텐츠 요소

요소가 스키마에서 정의된 모든 글로벌 요소가 될 수 있습니다.

## anyAttribute 특성

요소가 스키마에서 이미 정의된 모든 특성이 될 수 있습니다.

## 대체 그룹 생성

대체 그룹은 SOAP 메시지에서 한 요소를 다른 요소로 바꿀 때 사용합니다. 대체 그룹은 요소 정의에 대체 그룹 이름이 포함된다는 점을 제외하고 파생된 유형과 유사한 방식으로 작동합니다.

예를 들어 기본 유형인 Address와 파생된 유형인 USAddress 및 UKAddress가 있습니다.

```
xs:element name="Address" type="xs:string"/>
<xs:element name="USAddress" substitutionGroup="Address"/>
<xs:element name="UKAddress" substitutionGroup="Address"/>
```

SOAP 메시지 계층을 구성할 때 SOAP 메시지에서 Address 유형을 대체할 요소를 선택할 수 있습니다.

## SOAP 메시지의 XML 구성 생성

WSDL 또는 스키마에는 선택, 목록 또는 합집합 요소가 포함될 수 있습니다. 웹 서비스 변환을 사용하면 이러한 요소가 포함되는 SOAP 메시지를 생성할 수 있습니다.

## 선택 요소

선택 요소는 하위 요소를 <선택> 선언의 요소 중 하나로 제한합니다.

선택 요소가 포함된 SOAP 메시지에 포트를 매핑하려면 선택 구성의 모든 요소를 포함하는 하나의 입력 그룹을 작성합니다. 예를 들어 항목 설명은 차원 또는 가중치일 수 있습니다.

item: description, choice {dimension, weight}

차원일 경우 설명은 길이, 너비 및 높이를 포함하는 복합 유형입니다.

가중치일 경우 설명은 단순 문자열 유형입니다.

입력 데이터에는 다음과 같은 열 및 행이 포함됩니다.

설명	길이	너비	높이	가중치
Box	20cm	18cm	15cm	Null
Coffee	Null	Null	Null	500g

SOAP 메시지에는 차원 또는 가중치 설명이 포함된 Item 그룹이 포함됩니다.

```
Item
  Description
    Dimension
      Length
      Width
      Height
    Weight
```

입력 데이터의 NULL 값은 XML 출력에서 누락된 요소가 됩니다.

SOAP 메시지에는 다음 데이터가 포함됩니다.

```
<item>
  <desc>box</desc>
  <dimension>
    <length>20cm</length>
    <width>18cm</width>
```

```

        <height>15cm</height>
    </dimension>
</item>

<item>
    <desc>coffee</desc>
    <weight>500g</weight>
</item>

```

## 목록 요소

목록은 동일한 요소 또는 특성의 여러 단순 유형 값이 포함될 수 있는 XML 요소입니다. 데이터 통합 서비스에서는 입력 데이터의 목록이 통합된 문자열 데이터로 표시될 경우 목록을 처리할 수 있습니다.

목록의 각 항목이 **ClassDates1**, **ClassDates2** 및 **ClassDates3**과 같은 개별 요소일 경우 데이터 통합 서비스에서 항목을 목록으로 처리할 수 없습니다. SOAP 메시지의 목록을 반환해야 하는 경우 식 변환을 사용하여 항목을 문자열로 결합할 수 있습니다.

다음 입력 행에는 요일이 포함된 **ClassDates**라는 목록 요소가 포함됩니다.

CourseID	Name	ClassDates
Math 1	Beginning Algebra	Mon Wed Fri
History 1	World History	Tue Thu

데이터 통합 서비스에서는 다음 XML 구조의 SOAP 메시지를 반환할 수 있습니다.

```

<class>
  <courseId>Math 1</courseId>
  <name>Beginning Algebra</name>
  <classDates>Mon Wed Fri</classDates>
</class>
<class>
  <courseId>History 1</courseId>
  <name>World History</name>
  <classDates>Tue Thu</classDates>
</class>

```

## 합집합 요소

합집합 요소는 둘 이상의 유형이 결합된 단순 유형입니다. SOAP 메시지에 합집합 요소가 포함된 경우 문자열의 데이터를 포함하는 단일 입력 포트를 매핑해야 합니다.

예를 들어, SOAP 메시지에 크기라는 요소가 포함되어 있습니다. 크기는 정수 및 문자열의 합집합입니다.

```

<xs:element name="size">
  <xs:simpleType>
    <xs:union memberTypes="size_no size_string" />
  </xs:simpleType>
</xs:element>

```

입력 행에 설명 및 크기가 포함된 항목이 들어 있습니다. 항목에는 숫자 크기(예: 42)가 있을 수 있습니다. 또는 항목에 문자열 값(라지, 미디엄, 스몰)의 크기가 있을 수 있습니다.

다음 테이블에는 숫자 크기와 문자열 크기를 가진 입력 행이 표시되어 있습니다.

설명	크기
신발	42
셔츠	라지

항목 크기에 대해 하나의 포트를 작성합니다. 포트를 문자열로 매핑합니다. **SOAP** 메시지에 다음 요소가 포함되어 있습니다.

```
<item>
  <desc>shoes</desc>
  <size>42</size>
</item>

<item>
  <desc>shirt</desc>
  <size>large</size>
</item>
```

## 제 49 장

# 가중치 평균 변환

이 장에 포함된 항목:

- [가중치 평균 변환 개요, 666](#)
- [가중치 평균 변환 구성, 666](#)
- [가중치 일치 점수 예제, 667](#)
- [가중치 평균 변환 고급 속성, 667](#)
- [가중치 평균 변환 - 비원시 환경, 667](#)

## 가중치 평균 변환 개요

가중치 평균 변환은 여러 일치 작업에서 일치 점수를 읽고 단일 일치 점수를 생성하는 수동 변환입니다.

숫자 가중치를 가중치 평균 변환을 시작하는 각 점수에 적용할 수 있습니다. 가중치는 0과 1 사이의 값입니다. 각 입력 점수에 적용된 가중치를 편집하여 출력 점수에 대한 기여도를 늘리거나 줄일 수 있습니다. 중복 분석에서 각 데이터 열의 상대적 중요도를 반영하는 가중치를 적용합니다.

비교 변환을 매핑 또는 맵셋에 추가하는 경우 가중치 평균 변환을 사용합니다.

**참고:** 또한 일치 변환에서 가중치를 할당할 수 있습니다. 일치 변환을 사용하여 일치 전략을 구성하고 단일 변환에서 가중치를 할당할 수 있습니다. 일치 맵셋을 일치 변환에 포함할 수 있습니다.

## 가중치 평균 변환 구성

매핑이 일련의 일치 분석 작업에 대해 생성하는 전체 일치 점수를 가중치 평균 변환을 사용하여 조정할 수 있습니다. 소스 데이터 집합에 정의한 데이터 비교의 우선 순위를 반영하도록 각 입력 포트의 상대 가중치를 편집할 수 있습니다. 가중치 평균 변환의 각 입력 포트는 비교 변환 전략의 일치 점수 출력을 나타냅니다.

다음 단계는 재사용할 수 없는 가중치 평균 변환을 비교 변환을 사용하는 맵셋 또는 매핑에 구성하는 프로세스를 설명합니다.

1. 일치 분석 맵셋 또는 매핑을 열고 비교 변환의 가중치 평균 변환 다운스트림을 추가합니다.
2. 비교 변환의 점수 출력을 가중치 평균 입력 포트에 연결합니다.  
맵셋 또는 매핑의 다른 비교 변환에도 이 단계를 반복합니다.
3. 가중치 평균 변환의 **포트** 탭을 선택합니다.

4. 각 입력에 대한 **가중치** 필드를 두 번 클릭하고 0.001에서 1 사이의 가중치 값을 입력합니다. 가중치 값은 변환의 다른 입력과 비교한 입력 점수의 상대적 중요도를 반영한 것이어야 합니다.
5. 맵렛 또는 매핑을 저장합니다.

## 가중치 일치 점수 예제

고객 데이터베이스에서 중복된 고객 이름 수를 확인하기 위해 일치 분석 매핑을 작성합니다. 데이터 집합의 ZIP 코드 및 성 열에 대한 일치 점수를 생성하기 위해 두 개의 비교 변환을 추가합니다.

여러 레코드에 일치하는 ZIP 코드가 있지만 성이 일치하는 레코드는 훨씬 적습니다. 이러한 일치 점수의 평균을 계산할 때 더 고유한 일치의 중요성을 강조해야 합니다.

성 일치 점수의 중요성을 강조하기 위해 더 높은 가중치를 성 일치 점수에 적용합니다.

예를 들어 성 점수 입력의 **가중치** 값을 0.8로 설정하고 ZIP 코드 점수 입력의 **가중치** 값을 0.4로 설정합니다.

## 가중치 평균 변환 고급 속성

가중치 평균 변환에 대해 데이터 통합 서비스가 데이터를 처리하는 방법을 결정할 수 있는 속성을 구성합니다.

로그의 추적 수준을 구성할 수 있습니다.

**고급** 탭에서 다음 속성을 구성합니다.

### 추적 수준

이 변환에 대해 로그에 표시되는 세부 정보의 양입니다. 간단, 보통, 자세한 정보 표시 초기화 또는 자세한 정보 표시 데이터를 선택할 수 있습니다. 기본값은 보통입니다.

## 가중치 평균 변환 - 비원시 환경

비원시 환경에서 가중치 평균 변환의 처리는 변환을 실행하는 엔진에 따라 다릅니다.

다음과 같은 비원시 런타임 엔진에 대한 지원을 고려하십시오.

- Blaze 엔진. 제한 없이 지원됩니다.
- Spark 엔진. 제한 없이 지원됩니다.
- Databricks Spark 엔진. 지원되지 않습니다.

## 제 50 장

# 쓰기 변환

이 장에 포함된 항목:

- [쓰기 변환 개요, 668](#)
- [쓰기 변환 속성, 668](#)
- [쓰기 변환 작성, 674](#)

## 쓰기 변환 개요

쓰기 변환은 수동 변환입니다. 매핑은 쓰기 변환을 사용하여 데이터를 대상에 기록합니다. 쓰기 변환은 재사용 불가능합니다.

실제 데이터 개체, 논리적 데이터 개체 또는 매개 변수에서 쓰기 변환을 작성할 수 있습니다. **PowerExchange** 어댑터 소스에서 가져온 실제 데이터 개체에서 쓰기 변환을 생성하려는 경우 매핑 편집기에서 데이터 개체를 사용하여 쓰기 변환을 생성하기 전에 쓰기 작업을 지정하라는 메시지가 표시될 수 있습니다.

쓰기 변환에 대해 구성할 수 있는 속성은 변환을 작성하는 데 사용한 데이터 개체의 유형에 따라 다릅니다.

쓰기 변환은 동적 대상을 나타낼 수 있습니다. 쓰기 변환이 포트, 메타데이터 및 다른 속성을 동적으로 업데이트하도록 구성할 수 있습니다. 동적 대상 구성에 대한 자세한 내용은 *Informatica Developer 매핑 가이드*를 참조하십시오.

## 쓰기 변환 속성

쓰기 변환을 작성한 후 변환에 대한 속성을 구성할 수 있습니다.

변환의 **속성** 보기에 있는 탭에서 쓰기 변환 속성을 구성합니다. 구성할 수 있는 탭은 쓰기 변환이 나타내는 대상의 유형에 따라 다릅니다.



다음 표에는 각 속성 탭이 설명되어 있고 탭에 대해 사용하는 대상 유형이 식별되어 있습니다.

속성 탭	설명	대상 유형
일반	변환 속성 및 동작을 지정합니다. 관계형 및 사용자 지정된 데이터 개체 소스의 경우 변환 입력 포트를 소스와 동기화합니다.	모두
데이터 개체	변환 데이터 소스를 지정합니다. 관계형 및 사용자 지정된 데이터 개체 소스의 경우 런타임에 데이터 소스에서 데이터 개체 열을 가져옵니다.	플랫 파일 관계형 사용자 지정된 데이터 개체
형식	플랫 파일 데이터 소스에 대한 설정을 입력합니다.	플랫 파일
포트	연결된 데이터 개체 또는 매핑 흐름에 따라 포트 정의를 설정합니다.	모두
런타임	런타임에 대상에 데이터를 쓸 때 데이터 통합 서비스가 사용하는 속성입니다 (예: 거부 파일을 보낼 대상). 플랫 파일 대상의 경우 거부 파일 이름 및 디렉터리입니다.	플랫 파일 관계형
데이터 개체 매개 변수	데이터 개체 매개 변수를 봅니다. 매핑에 대한 매개 변수 값을 구성하거나 매개 변수를 매핑 매개 변수에 바인딩합니다.	플랫 파일 사용자 지정된 데이터 개체 논리적 데이터 개체
런타임 연결	새 런타임 링크를 작성하고 링크 속성을 봅니다.	모두
고급	추적 수준 및 행 순서를 설정합니다. 관계형 또는 Hive 대상의 경우 대상 스키마 전략을 설정합니다.	플랫 파일 관계형 사용자 지정된 데이터 개체 논리적 데이터 개체

## 일반 속성

쓰기 변환의 이름 및 설명을 구성할 수 있습니다. 또한 다음 속성을 구성할 수 있습니다.

### 열 메타데이터 변경 시

관계형 및 사용자 지정된 대상에서 사용 가능합니다. 다음 옵션 중 하나를 선택합니다.

- 입력 포트 동기화. **Developer tool**은 모델 리포지토리가 데이터 개체에 대해 저장하는 메타데이터 변경으로 쓰기 변환 입력 포트를 업데이트합니다.
- 동기화 안 함. **Developer tool**이 데이터 개체의 메타데이터 변경을 표시하지 않습니다.

기본값은 입력 포트 동기화 옵션입니다.

### 실제 데이터 개체

플랫 파일 및 사용자 지정된 대상에서 사용 가능합니다. 변환을 작성하는 데 사용된 개체입니다.

데이터 개체 이름을 선택하고 해당 속성을 구성할 수 있습니다.

## 데이터 개체 속성

데이터 개체 탭에서 동적으로 쓰기 변환 대상을 지정하거나 변경하고 관계형 및 사용자 지정된 데이터 개체 대상을 만들 수 있습니다.

다음과 같은 속성을 구성할 수 있습니다.

### 지정 기준

쓰기 변환에 대한 대상 열 및 메타데이터를 지정하려면 다음 옵션 중 하나를 선택합니다.

- 값. 쓰기 변환이 연결된 데이터 개체를 사용하여 대상 열 및 메타데이터를 지정합니다.
- 매개 변수. 쓰기 변환이 매개 변수를 사용하여 대상 열 및 메타데이터를 지정합니다.

기본값은 값 옵션입니다.

### 데이터 개체

기본 데이터 개체에서 쓰기 변환을 작성한 경우 필드에 개체의 이름이 표시됩니다. **찾아보기**를 클릭하여 쓰기 변환과 연결할 데이터 개체를 변경합니다.

### 매개 변수

쓰기 변환과 연결할 매개 변수를 선택하거나 작성합니다.

### 런타임 시 데이터 소스에서 데이터 개체 열을 가져옵니다.

이 옵션을 활성화하면 데이터 통합 서비스가 대상 테이블에서 메타데이터 및 데이터 정의 변경을 쓰기 변환으로 가져옵니다.

데이터 통합 서비스가 메타데이터 및 데이터 정의 변경을 가져오는 방법을 미리 보려면 확인된 매개 변수와 함께 매핑을 표시합니다.

## 포트 속성

포트 탭에서 다음 속성을 구성할 수 있습니다.

### 열 정의 기준

다음 옵션 중 하나를 선택하여 쓰기 변환 열을 정의합니다.

- 연결된 데이터 개체. 데이터 개체 탭에서 데이터 개체의 열 이름, 메타데이터 및 다른 속성을 사용합니다.
- 매핑 흐름. 매핑이 매핑의 업스트림 개체에서 열 이름, 메타데이터 및 다른 속성을 가져옵니다.

기본값은 연결된 데이터 개체 옵션입니다.

### 열 리소스 속성

플랫 파일 및 사용자 지정된 대상에서 사용 가능합니다. 각 열의 리소스는 열이 이름, 메타데이터 및 다른 속성을 가져오는 데이터 개체입니다. 리소스 이름을 선택하여 리소스 속성을 변경합니다.

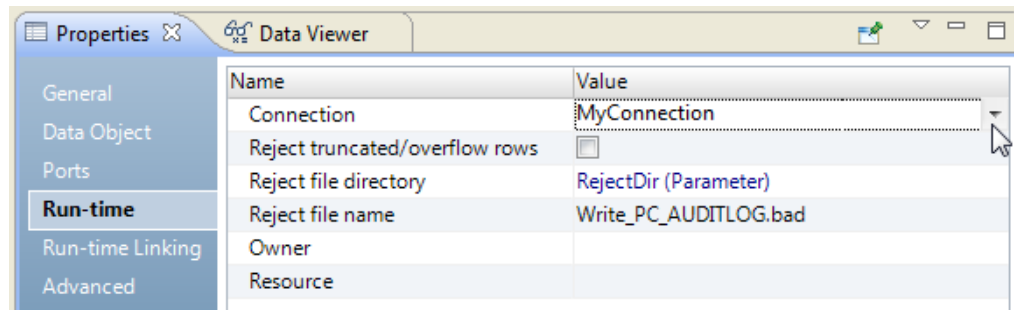
## 런타임 속성

런타임 탭에서 다음 쓰기 트랜잭션 속성을 구성할 수 있습니다.

### 연결

관계형 대상에서 사용 가능합니다. 변환에서 사용된 연결입니다. 필드 오른쪽을 클릭하여 연결을 변경합니다.

다음 이미지는 클릭할 드롭다운 단추의 위치를 보여 줍니다.



### 잘린/오버플로우 행 거부

관계형 및 사용자 지정된 대상에서 사용 가능합니다.

Developer tool을 통해 포트 간에 데이터를 전달하여 데이터를 변환할 수 있습니다. 변환으로 인해 때때로 숫자 데이터 오버플로우 또는 문자가 포함된 열의 문자열 잘림이 발생합니다. 예를 들어, **Decimal(28, 2)**에서 **Decimal(19, 2)** 포트로 데이터를 전달하면 숫자 오버플로우가 발생합니다. 마찬가지로 **String(28)** 포트에서 **String(10)** 포트로 데이터를 전달하면 데이터 통합 서비스가 문자열을 10자로 잘라냅니다.

변환으로 인해 오버플로우가 발생할 경우 기본적으로 데이터 통합 서비스가 행을 건너뛵니다. 데이터 통합 서비스는 거부 파일에 데이터를 쓰지 않습니다. 문자열의 경우 데이터 통합 서비스는 문자열을 잘라내고 해당 문자열을 다음 변환에 전달합니다.

이 옵션을 선택하면 마지막 변환과 대상 간에 잘리거나 오버플로우된 데이터를 모두 세션 거부 파일에 포함합니다. 잘린/오버플로우된 행 거부를 선택한 경우 데이터 통합 서비스는 모든 잘린 행 및 오버플로우된 행을 세션 구성 방법에 따라 세션 거부 파일 또는 행 오류 로그에 전송합니다.

### 거부 파일 디렉터리

거부 파일이 있는 디렉터리입니다. 기본값은 **RejectDir** 시스템 매개 변수입니다.

### 거부 파일 이름

거부 파일의 파일 이름입니다. 기본값은 **<output\_file\_name>.bad**입니다.

여러 파티션이 플랫폼 파일 대상에 쓰는 경우 각 파티션이 **<output\_file\_name><partition\_number>.bad**라는 별도의 거부 파일에 씁니다.

## 런타임 링크 속성

**런타임 링크** 탭에서 런타임 링크를 작성하고 구성합니다. 런타임 링크는 런타임 시 사용할 포트를 확인하는 데 매개 변수, 링크 정책 또는 둘 다를 사용하는 변환 간 그룹-그룹 연결입니다. 런타임 링크는 매핑 편집기에서 두꺼운 선으로 나타납니다.

다음 경우에 쓰기 변환에 런타임 링크를 작성하고 구성합니다.

- 쓰기 변환의 대상 데이터 개체가 매개 변수를 사용합니다.
- 업스트림 변환의 포트가 런타임 시 변경될 수 있습니다.

**참고:** 매핑 흐름에 따라 대상 열을 정의하는 경우 쓰기 변환에 런타임 링크를 작성하지 마십시오.

**런타임 링크** 탭에서 다음 태스크를 수행할 수 있습니다.

### 런타임 링크 작성

**링크** 영역에서 **새로 만들기** 단추를 클릭하고 새 링크 대화 상자에서 런타임 시 포트를 쓰기 변환에 연결할 변환을 선택합니다.

## 런타임 링크 속성 구성

링크 속성 영역에서 다음 런타임 링크 속성을 구성합니다.

### 매개 변수

매핑 실행 간 포트 이름이 변경될 수 있고 포트 이름 값을 아는 경우 이 옵션을 선택합니다. 입력 링크 설정 유형의 매개 변수를 사용하여 매핑 실행 간 이름 값으로 포트를 연결합니다. 입력 링크 설정 매핑 매개 변수에 대한 구문은 쉼표로 구분된 포트 쌍으로 구성됩니다. Afield1->Bfield2, Afield3->Bfield4 .

### 링크 정책

일치하는 이름이 있는 포트를 자동으로 연결하려는 경우 이 옵션을 선택합니다. 예를 들어 매핑 개체에 SALARY라는 포트가 포함된 경우 데이터 통합 서비스가 해당 개체를 연결합니다. 포트 이름의 접두사 및 접미사는 무시할 수 있습니다.

런타임 시 데이터 통합 서비스는 다음 순서로 포트 간 링크의 연결을 설정하고 확인합니다.

- 매핑 편집기에서 수동으로 작성한 링크
- 런타임 링크에 대해 구성한 매개 변수에 따른 링크
- 런타임 링크에 대해 구성한 링크 정책에 따른 링크

런타임 링크에 대한 자세한 내용은 *Informatica Developer 매핑 가이드*를 참조하십시오.

## 고급 속성

고급 속성을 구성하여 데이터 통합 서비스가 쓰기 변환에 대해 데이터를 처리하는 방법을 결정합니다.

고급 탭에서 다음 속성을 구성합니다.

### 추적 수준

매핑 로그 파일의 세부 정보의 양을 제어합니다.

### 대상 로드 유형

대상 로드의 유형입니다. 일반 또는 대량을 선택합니다. 관계형 리소스 또는 사용자 지정된 데이터 개체에 대한 대상 로드 유형을 설정할 수 있습니다.

일반을 선택할 경우 데이터 통합 서비스가 대상을 일반적으로 로드합니다. DB2, Sybase, Oracle 또는 Microsoft SQL Server에 로드하는 경우 대량을 선택할 수 있습니다. 다른 데이터베이스 유형에 대해 대량을 선택하면 데이터 통합 서비스가 일반 로드로 되돌립니다. 대량 로드를 사용하면 매핑 성능을 향상시킬 수 있지만 데이터베이스 로깅이 발생하지 않으므로 복구 기능이 제한됩니다. 대량 로드를 사용하여 Oracle 대상에 쓰는 경우 Oracle 데이터베이스에서 제약 조건을 비활성화하여 성능을 최적화할 수 있습니다.

매핑에 업데이트 전략 변환이 포함되는 경우에는 일반 모드를 선택하십시오. 일반 모드를 선택하고 Microsoft SQL Server 대상 이름에 공백이 포함되는 경우 다음 환경 SQL을 연결 개체에 구성합니다.

```
SET QUOTED_IDENTIFIER ON
```

### 업데이트 재정의

대상에 대한 기본 UPDATE 문을 재정의합니다.

### 삭제

삭제 플래그가 지정된 모든 행을 삭제합니다.

기본값은 활성화됩니다.

### 삽입

삽입 플래그가 지정된 모든 행을 삽입합니다.

기본값은 활성화됩니다.

## 대상 스키마 전략

관계형 또는 Hive 대상 테이블에 대한 대상 스키마 전략의 유형입니다.

다음과 같은 대상 스키마 전략 중 하나를 선택할 수 있습니다.

- **RETAIN** - 기존 대상 스키마 유지. 데이터 통합 서비스는 기존 대상 스키마를 유지합니다.
- **CREATE** - 런타임 시 테이블 생성 또는 바꾸기. 데이터 통합 서비스가 식별하는 대상 테이블에 따라 런타임 시 대상 테이블을 삭제하고 테이블로 바꿉니다.
- 매개 변수 할당. 대상 스키마 전략의 값을 나타내는 매개 변수를 할당한 후 런타임 시 매개 변수를 변경할 수 있습니다.

## 생성 또는 바꾸기에 대한 DDL 쿼리

정의한 DDL 쿼리에 따라 런타임 시 대상 테이블을 생성하거나 바꿉니다. **CREATE - 런타임 시 테이블 생성 또는 바꾸기** 대상 스키마 전략 옵션을 선택하는 경우 적용 가능합니다.

## 대상 테이블 잘라내기

데이터를 로드하기 전에 대상을 잘라냅니다.

기본값은 활성화됩니다.

## 대상 파티션 잘라내기

데이터를 로드하기 전에 내부 또는 외부의 분할된 Hive 대상을 잘라냅니다. 이 옵션을 선택하기 전에 **대상 테이블 잘라내기**를 선택해야 합니다.

기본값은 비활성화됩니다.

## 업데이트 전략

기존 행에 대한 업데이트 전략입니다. 다음 전략 중 하나를 선택할 수 있습니다.

- 업데이트 시 업데이트. 데이터 통합 서비스가 업데이트 플래그가 지정된 모든 행을 업데이트합니다.
- 삽입 시 업데이트. 데이터 통합 서비스가 업데이트 플래그가 지정된 모든 행을 삽입합니다. **삽입** 대상 옵션도 선택해야 합니다.
- 업데이트 기타 항목 삽입. 업데이트 플래그가 지정된 행이 대상에 존재하는 경우 데이터 통합 서비스가 해당 행을 업데이트한 다음 삽입이 표시된 나머지 행을 삽입합니다. **삽입** 대상 옵션도 선택해야 합니다.

## PreSQL

소스를 읽기 전에 대상 데이터베이스에 대해 데이터 통합 서비스가 실행하는 SQL 명령입니다.

Developer tool은 SQL의 유효성을 검사하지 않습니다.

## PostSQL

대상에 쓴 후에 대상 데이터베이스에 대해 데이터 통합 서비스가 실행하는 SQL 명령입니다.

Developer tool은 SQL의 유효성을 검사하지 않습니다.

## 행 순서 유지

대상에 대한 입력 데이터의 행 순서를 유지합니다. 데이터 통합 서비스에서 행 순서를 변경할 수 있는 어떤 최적화도 수행하지 않는 경우 이 옵션을 선택합니다.

데이터 통합 서비스가 최적화를 수행하는 경우 이전에 매핑에 설정된 행 순서를 잃게 될 수 있습니다. 정렬된 플랫폼 파일 소스, 정렬된 관계형 소스 또는 분류기 변환을 사용하여 매핑에서 행 순서를 설정할 수 있습니다. 대상이 행 순서를 유지하도록 구성하는 경우 데이터 통합 서비스가 대상에 대한 최적화를 수행하지 않습니다.

## 제약 조건

테이블 수준 참조 무결성 제약 조건에 대한 SQL 문입니다. 관계형 대상에만 적용됩니다.

## 쓰기 변환 작성

쓰기 변환을 작성하는 경우 변환을 작성하는 리소스 위치에 따라 다음 방법 중 하나를 선택합니다.

### 모델 리포지토리의 데이터 개체에서 변환을 작성합니다.

쓰기 변환 메타데이터를 데이터 개체에 기반하도록 하려는 경우 이 방법을 사용합니다.

매핑 편집기에서 쓰기 변환을 작성하려면 **새 쓰기 변환** 마법사를 사용하거나 **Object Explorer** 보기에서 데이터베이스를 끌어와 쓰기를 변환 유형으로 선택합니다.

### 매핑 개체의 흐름에서 변환 작성

쓰기 변환의 포트를 업스트림 매핑 변환의 메타데이터 흐름에 기반하도록 하려면 이 방법을 사용합니다.

### 매개 변수를 사용하여 변환 작성

쓰기 변환이 매개 변수가 나타내는 개체에서 포트를 상속하도록 하려는 경우 이 방법을 사용합니다.

### 매핑에서 다른 변환을 사용하여 변환 작성

쓰기 변환에 소스 변환과 동일한 포트가 있도록 하려는 경우 이 방법을 사용합니다.

## 데이터 개체에서 쓰기 변환 작성

매핑 편집기에서 쓰기 변환을 작성하고 구성할 수 있습니다.

변환에 대한 특정 속성을 구성하려는 경우 이 방법을 사용하고자 할 수 있습니다. 예를 들어 변환을 작성한 다음 런타임 시 데이터 소스로부터 열 메타데이터를 가져오도록 구성할 수 있습니다.

1. 매핑 편집기에서 마우스 오른쪽 단추를 클릭하고 **변환 추가**를 선택합니다.  
**변환 추가** 대화 상자가 열립니다.
2. 쓰기를 선택하고 **확인**을 클릭합니다.  
**새 쓰기 변환** 마법사가 열립니다.
3. **실제 데이터 개체** 또는 **논리적 데이터 개체**를 선택한 다음 **찾아보기**를 클릭합니다.  
**데이터 개체 선택** 대화 상자가 열립니다.
4. 데이터 소스를 선택한 다음 **확인**을 클릭합니다.
5. 연결된 데이터 개체로 쓰기 변환 열을 정의하려면 **연결된 데이터 개체**를 선택합니다.  
쓰기 변환 포트가 연결된 데이터 개체의 열과 동일합니다.
6. 런타임 시 대상 파일의 변경으로 대상 개체 열을 동적으로 업데이트하려면 **런타임 시 데이터 소스에서 데이터 개체 열 가져오기**를 선택합니다.  
매핑 실행 시 데이터 통합 서비스가 쓰기 변환에 대한 열 메타데이터를 새로 고칩니다.
7. **마침**을 클릭합니다.

## 매핑 흐름에서 쓰기 변환 작성

매핑 편집기에서 쓰기 변환을 작성하고 구성할 수 있습니다.

변환에 대한 특정 속성을 구성하려는 경우 이 방법을 사용하고자 할 수 있습니다. 예를 들어 변환을 작성한 다음 매핑 흐름에 따라 열을 정의하고 런타임 시 데이터 소스로부터 열 메타데이터를 가져오도록 구성할 수 있습니다.

1. 매핑 편집기에서 마우스 오른쪽 단추를 클릭하고 **변환 추가**를 선택합니다.  
**변환 추가** 대화 상자가 열립니다.
2. 쓰기를 선택하고 **확인**을 클릭합니다.

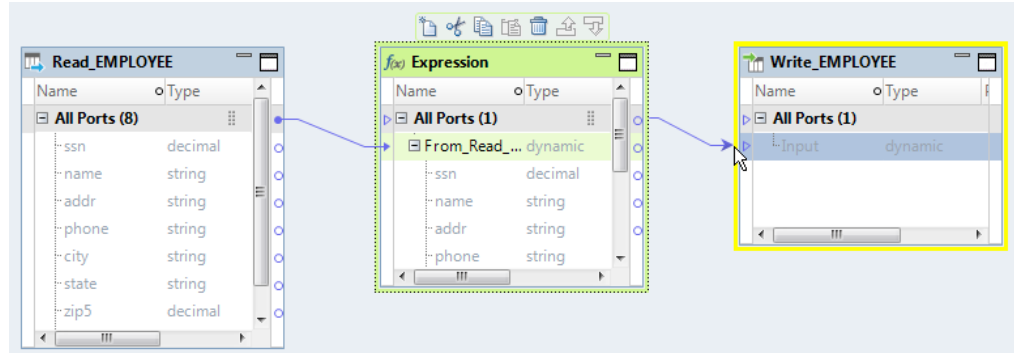
새 쓰기 변환 마법사가 열립니다.

3. 실제 데이터 개체 또는 논리적 데이터 개체를 선택한 다음 **찾아보기**를 클릭합니다.

데이터 개체 선택 대화 상자가 열립니다.

4. 데이터 소스를 선택하고 **확인**을 클릭합니다.
5. 매핑 흐름을 선택한 다음 **마침**을 클릭합니다.
6. 업스트림의 모든 포트 포트로부터 쓰기 변환의 **입력** 포트에 포트를 끌어옵니다.

다음 이미지는 업스트림 포트를 쓰기 변환의 **입력** 포트에 연결하는 방법을 보여 줍니다.



쓰기 변환이 업스트림 매핑 개체로부터 열 정의를 받습니다.

7. 런타임 시 대상 파일의 변경으로 대상 개체 열을 동적으로 업데이트하려면 **런타임 시 데이터 소스에서 데이터 개체 열 가져오기**를 선택합니다.

매핑 실행 시 데이터 통합 서비스가 쓰기 변환에 대한 열 메타데이터를 새로 고칩니다.

## 매개 변수에서 쓰기 변환 작성

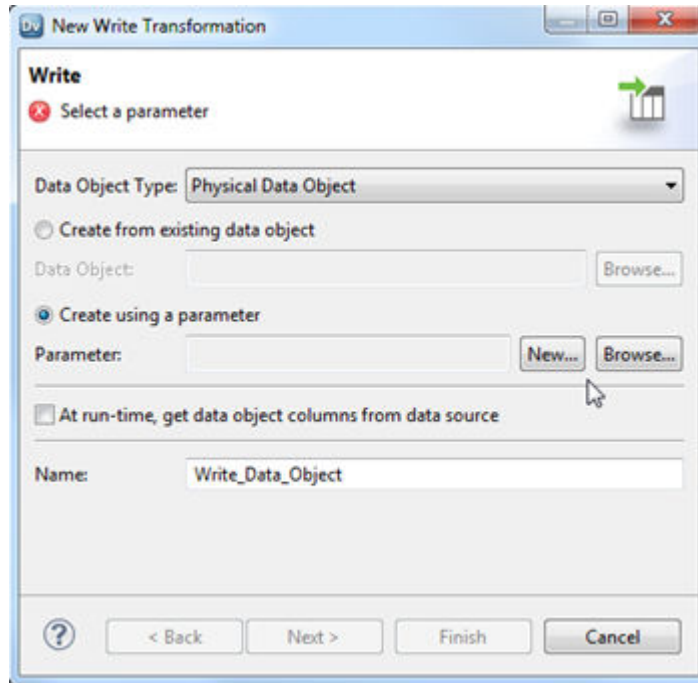
매개 변수는 매핑 실행 간에 변경할 수 있는 상수 값입니다.

동적 매핑에서 매개 변수를 사용하여 소스 및 대상을 변경할 수 있습니다. 또한 입력 규칙, 선택 규칙, 런타임 링크 및 변환 속성에 대해 매개 변수를 사용할 수 있습니다. 매개 변수의 값을 변경하면 데이터 통합 서비스가 매개 변수에 지정된 값에 따라 대상을 작성하거나 다시 작성합니다.

1. 편집기에서 마우스 오른쪽 단추를 클릭하고 **변환 추가**를 선택합니다.
2. 변환 목록에서 쓰기를 선택한 다음 **확인**을 클릭합니다.

**팁:** 변환의 첫 글자를 입력하여 목록을 필터링합니다.

새 쓰기 변환 대화 상자가 열립니다.



3. 매개 변수를 사용하여 작성을 선택합니다.
4. 매개 변수를 찾아 선택하거나 새로 만들기를 클릭하여 새 매개 변수를 작성한 다음 매개 변수를 작성할 데이터 개체를 찾아 선택합니다.
5. 필요한 경우 런타임 시 데이터 소스로부터 데이터 개체 열 가져오기 옵션을 선택하여 런타임 시 변환 열을 새로 고칩니다.
6. 필요한 경우 다음을 클릭하여 변환 열을 정의할 방법을 선택합니다.
7. 마침을 클릭합니다.

## 기존 변환에서 쓰기 변환 생성

매핑에 포함된 변환 안에 동일한 포트를 사용하여 쓰기 변환을 생성할 수 있습니다.

복합 파일, 플랫폼 파일 또는 관계형 리소스에 대한 대상을 생성할 수 있습니다. 기존 변환의 포트 중 하나에 복합 포트가 포함되는 경우 복합 파일 대상을 생성하고 이름별로 포트를 연결하거나 런타임 시 연결 정책에 따라 포트를 연결해야 합니다. Developer tool에서는 Avro, Parquet, ORC 또는 JSON 복합 파일 대상을 생성할 수 있습니다.

1. 편집기에서 매핑을 엽니다.
2. 매핑 편집기에서 변환을 마우스 오른쪽 단추로 클릭하고 **대상 생성**을 선택합니다.

**대상 생성** 창이 열립니다.

3. 복합 파일, 플랫폼 파일 또는 관계형 데이터 개체 유형을 선택합니다.
4. 링크 유형을 선택합니다.

다음 링크 유형 중에서 선택할 수 있습니다.

### 이름별로 포트 연결

쓰기 변환에 있는 포트가 소스에 있는 포트와 일치하며 이름이 같습니다.



### 매핑 흐름에 기반하여 동적 포트 연결

쓰기 변환에 매핑 흐름의 업스트림 개체를 기반으로 하는 동적 포트가 포함되어 있습니다.

### 링크 정책에 기반하여 런타임에 포트 연결

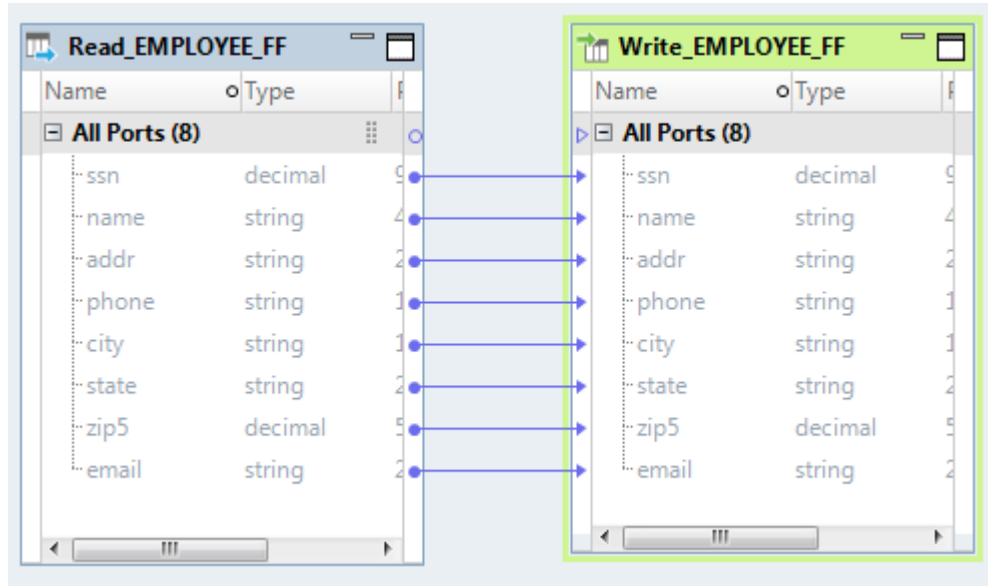
포트가 쓰기 변환의 런타임 링크 탭에서 구성된 링크 정책에 기반하여 런타임 시 대상에 생성됩니다.

동적 포트 및 런타임 링크 구성에 대한 자세한 내용은 *Informatica Developer 매핑 가이드*를 참조하십시오.

5. 새 데이터 개체의 이름을 지정합니다.
6. 필요한 경우 **찾아보기**를 클릭하여 데이터 개체의 위치를 선택합니다.
7. 복합 파일 대상을 선택하거나 생성하는 경우 **리소스 형식** 드롭다운 상자에서 복합 파일 형식을 Avro 또는 Parquet로 선택합니다.
8. **마침**을 클릭합니다.

Developer tool은 다음과 같은 태스크를 수행합니다.

- 매핑에 쓰기 변환을 추가합니다.  
다음 이미지는 읽기 변환에서 생성된 쓰기 변환을 보여 줍니다.



- 포트를 연결합니다.
- 실제 데이터 개체를 생성합니다.  
관리자는 실제 데이터 개체 속성을 구성할 수 있습니다. 예를 들어 복합 파일 데이터 개체에 사용할 HDFS 연결을 지정해야 합니다.

## 부록 A

# 변환 구분자

이 부록에 포함된 항목:

- [변환 구분자 개요, 678](#)

## 변환 구분자 개요

변환 구분자는 데이터 문자열에서 나누기를 지정합니다.

다음 테이블에는 변환이 데이터 문자열을 구문 분석하고 기록하기 위해 사용하는 구분자가 나열되어 있습니다.

구분자 이름	구분자 기호
@ 기호	@
쉼표	,
대시	-
큰따옴표	"
슬래시	/
마침표	.
해시	#
파이프	
세미콜론	;
작은따옴표	'
공백	[스페이스바]
탭	[Tab 키]
밑줄	_

# 인덱스

16진수 소스 보기, 데이터 프로세서  
설명 [218](#)

## A

ABORT 함수  
사용 [51](#)  
anyAttribute 특성  
웹 서비스 소비자 변환 [628](#), [631](#)  
anyType  
포트 매핑 [662](#)  
anyType 요소  
구문 분석 [649](#)  
웹 서비스 소비자 변환 [628](#), [631](#)  
API 메서드  
Java 변환 [310](#)

## B

패키지 가져오기 탭  
Java 변환 [295](#)  
편집기와 동기화  
데이터 프로세서 변환 [235](#)  
포트  
Excel에서 복사 [66](#)  
Java 변환 [289](#)  
SOAP 메시지에 매핑 [657](#)  
구성 [60](#)  
기본값 개요 [49](#)  
라우터 변환 [558](#)  
변수 포트 [47](#)  
비정규화된 웹 서비스 입력 [660](#)  
생성 [59](#)  
시퀀스 생성기 변환 [561](#)  
잘못된 레코드 예외 변환 [147](#)  
중복 레코드 예외 변환 [248](#)  
평가 순서 [48](#)  
포트 값  
Java 변환 [290](#)  
포트 목록 매개 변수  
그룹 기준 탭 [132](#), [519](#)  
포트 보기  
SQL 변환 출력 [589](#)  
포트 선택기  
선택 규칙 [260](#), [261](#), [388](#), [389](#)  
동적 식 내 [267](#)  
선택 규칙 [260](#), [261](#), [388](#), [389](#)  
작성 [262](#), [339](#), [391](#)  
조이너 변환 [338](#)  
조이너 변환 내 [342](#)  
포트 특성  
전달 [63](#)

포트 할당  
품질 문제 [151](#)  
표준 출력 그룹  
중복 레코드 예외 변환 [249](#)  
표준 출력 포트  
설명 [148](#)  
표준화 변환  
개요 [611](#)  
푸시인 최적화  
SQL 변환 [599](#)  
SQL 변환에서 활성화 [600](#)  
웹 서비스 소비자 변환 [638](#)  
품질 문제  
포트 할당 대상 [151](#)  
품질 문제 포트  
빈 값과 NULL [144](#)  
설명 [147](#)  
피벗된 데이터  
SOAP 메시지 [659](#)  
피벗된 출력  
SOAP 메시지 구문 분석 [648](#)  
필드 일치 분석  
정의된 필드 분석 [424](#)  
프로세스 흐름 [444](#)  
필터 변환  
Blaze 엔진 [278](#)  
null 값을 포함하는 행 [277](#)  
개요 [274](#)  
고급 속성 [277](#)  
동적 매핑 [275](#)  
비원시 환경 [278](#)  
성능 팁 [278](#)  
필터 조건 [275](#)  
필터 조건 매개 변수 [276](#)  
필터 포트  
웹 서비스 소비자 변환 [638](#)  
하위 레코드  
노멀라이저 변환 [478](#)  
하한 임계값  
구성 [148](#), [246](#)  
함수 탭  
Java 변환 [298](#)  
합집합 변환  
Databricks Spark 엔진 [617](#)  
개요 [614](#)  
비원시 환경 [617](#)  
합집합 요소  
SOAP 메시지 구문 분석 [652](#)  
설명 [664](#)  
행  
업데이트 플래그 지정 [619](#)  
행 순서 유지  
시퀀스 생성기 변환 [571](#)  
현재 값  
시퀀스 생성기 변환 [569](#)

현재 값(속성)  
시퀀스 생성기 변환 [566](#)  
활성 변환  
Java [285](#), [286](#)  
설명 [33](#)  
순위 [515](#)  
휴먼 태스크  
잘못된 레코드 예외 [146](#)

## C

CURVAL 포트  
시퀀스 생성기 변환 [565](#)

## D

Data Transformation 서비스  
모델 리포지토리로 가져오기 [236](#)  
여러 서비스 가져오기 [236](#)  
defineJExpression  
Java 식 API 에서드 [329](#)  
defineJExpression 에서드  
Java 변환 [312](#)

## E

EDatatype 클래스  
Java 식 [328](#)  
ERROR 함수  
사용 [51](#)  
Excel  
Developer tool에 복사 [67](#)  
변환 편집 [68](#)  
변환 포트 구성 [67](#)  
포트 복사 [66](#)  
복사 규칙 및 지침 [68](#)

## F

failSession 에서드  
Java 변환 [312](#)

## G

GCID  
노멀라이저 변환 예제 [488](#)  
설명 [478](#)  
generateRow 에서드  
Java 변환 [313](#)  
getBytes 에서드  
Java 변환 [331](#)  
getDouble 에서드  
Java 변환 [331](#)  
getInRowType 에서드  
Java 변환 [314](#)  
getInt 에서드  
Java 변환 [331](#)  
getLong 에서드  
Java 변환 [332](#)  
getMetada 에서드  
Java 변환 [314](#)

getResultDataType 에서드  
Java 변환 [332](#)  
getResultMetadata 에서드  
Java 변환 [332](#)  
getStringBuffer 에서드  
Java 변환 [332](#)  
GZip  
SOAP 메시지 압축 [637](#)

## H

Hive 병합 사용  
옵션 [620](#)  
HTTP 연결  
REST 웹 서비스 [549](#)  
HTTP 오류 출력  
웹 서비스 소비자 변환에서 활성화 [634](#)  
HTTP 응답 코드  
REST 웹 서비스 소비자 변환 [545](#)  
HTTP 헤더  
REST 웹 서비스 소비자 변환에 추가 [544](#)  
웹 서비스 소비자 변환에 추가 [627](#)

## I

ID 인덱스 데이터의 지속형 저장소 [430](#)  
ID 일치 분석  
마스터 데이터 집합 [430](#)  
인덱스 데이터 분석의 출력 속성 [464](#)  
인덱스 데이터의 지속형 저장소 [430](#)  
정의된 ID 분석 [424](#)  
지속성 상태 설명 [439](#)  
지속성 상태 코드 [439](#)  
지속형 인덱스 데이터에 대한 규칙 및 지침 [431](#)  
프로세스 흐름 [456](#)  
IIF 함수  
시퀀스 생성기 변환으로 누락된 키 바꾸기 [564](#)  
incrementErrorCount 에서드  
Java 변환 [315](#)  
invoke  
Java 식 API 에서드 [333](#)  
invokeJExpression  
API 에서드 [325](#)  
invokeJExpression 에서드  
Java 변환 [315](#)  
isNull 에서드  
Java 변환 [316](#)  
isResultNull 에서드  
Java 변환 [333](#)

## J

Java 기본 데이터 유형  
Java 변환 [286](#)  
Java 변환  
API 에서드 [310](#)  
Blaze 엔진 [307](#)  
defineJExpression 에서드 [312](#)  
failSession 에서드 [312](#)  
generateRow 에서드 [313](#)  
getInRowType 에서드 [314](#)  
getMetadata 에서드 [314](#)  
incrementErrorCount 에서드 [315](#)  
invokeJExpression 에서드 [315](#)

## Java 변환 (계속)

- isNull 메서드 [316](#)
- Java 기본 데이터 유형 [286](#)
- Java 코드 [293](#)
- Java 코드 조각 작성 [295](#)
- logError 메서드 [317](#)
- logInfo 메서드 [318](#)
- null 값 검사 [316](#)
- null 값 설정 [319](#)
- resetNotification 메서드 [318](#)
- setNull 메서드 [319](#)
- Spark 엔진 [308](#)
- storeMetadata 메서드 [321](#)
- 가져오기 탭 [295](#), [297](#)
- 개요 [285](#)
- 고급 속성 [290](#)
- 그룹 작성 [289](#)
- 기본 포트 값 [290](#)
- 끝에서 탭 [298](#)
- 나노초 처리 [290](#)
- 높은 정밀도 처리 [290](#)
- 데이터 유형 변환 [286](#)
- 도우미 코드 탭 [296](#)
- 도우미 탭 [296](#), [297](#)
- 디자인 [289](#)
- 로그 [317](#), [318](#)
- 매핑 수준 클래스 경로 [293](#)
- 매핑 실패 위치 [312](#)
- 메타데이터 검색 [314](#)
- 메타데이터 저장 [321](#)
- 문제 해결 [303](#)
- 변수 재설정 [318](#)
- 변환 범위 [290](#)
- 비사용자 코드 오류 [304](#)
- 비원시 환경 [307](#)
- 사용자 코드 오류 [303](#)
- 상태 비저장 [290](#)
- 세션 로그 [318](#)
- 세션 실패 위치 [312](#)
- 수동 [286](#)
- 입력 시 탭 [297](#)
- 입력 포트 [290](#)
- 입력 행 유형 가져오기 [314](#)
- 작성 [301](#), [302](#)
- 전체 코드 탭 [299](#)
- 출력 포트 [290](#)
- 출력 행 유형 설정 [320](#)
- 컴파일 [302](#)
- 컴파일 오류 [303](#)
- 컴파일 오류의 소스 식별 [303](#)
- 패키지 가져오기 탭 [295](#)
- 포트 작성 [289](#)
- 함수 탭 [298](#)
- 활성 [286](#)

## Java 변환 API 메서드

- setOutRowType [320](#)
- 롤백 [319](#)
- 커밋 [311](#)

## Java 식

- EDatatype 클래스 [328](#)
- invokeJExpression API 메서드 [325](#)
- Java 변환 [322](#)
- Java 코드 생성 [324](#)
- JExpression 클래스 [329](#), [331](#)
- JExprParaMetadata 클래스 [328](#)
- 고급 인터페이스 [326](#)
- 고급 인터페이스 예제 [330](#)

## Java 식 (계속)

- 고급 인터페이스를 사용하여 호출 [327](#)
- 구성 [323](#)
- 규칙 및 지침 [325](#), [327](#)
- 단순 인터페이스 [325](#)
- 단순 인터페이스 예제 [326](#)
- 단순 인터페이스를 사용하여 호출 [325](#)
- 변환 언어 함수 사용 [322](#)
- 사용자 정의 함수 사용 [322](#)
- 사용자 지정 함수 사용 [322](#)
- 생성 [323](#)
- 식 정의 대화 상자에서 작성 [324](#)
- 식 함수 유형 [322](#)
- 작성 [324](#)
- 함수 구성 [323](#)
- 함수 정의 대화 상자에서 작성 [324](#)
- 호출 [315](#)
- 호출 규칙 및 지침 [315](#)

## Java 식 API 메서드

- defineJExpression [329](#)
- getBytes [331](#)
- getDouble [331](#)
- getInt [331](#)
- getLong [332](#)
- getResultDataType [332](#)
- getResultMetadata [332](#)
- getStringBuffer [332](#)
- invoke [333](#)
- isResultNull [333](#)

## Java 코드

- Java 변환 [293](#)
- 오류 찾기 [303](#)

## Java 코드 생성

- Java 식 [324](#)

## Java 코드 조각

- Java 변환에 대해 작성 [295](#)

## Java 코드 컴파일

- 전체 코드 탭 [299](#)

## Java 패키지

- 가져오기 [295](#)

## JDK

- Java 변환 [285](#)

## JExpression 클래스

- Java 식 [329](#), [331](#)

## JExprParaMetadata 클래스

- Java 식 [328](#)

## JRE

- Java 변환 [285](#)

## L

### logError 메서드

- Java 변환 [317](#)

### logInfo 메서드

- Java 변환 [318](#)

## N

### NEXTVAL 포트

- 시퀀스 생성기 [561](#)

### null 값

- Java 변환에 대한 설정 [319](#)

- Java 변환에서 검사 [316](#)

- 건너뛰기 [52](#)

- 상수로 바꾸기 [51](#)

null 일치 점수  
일치 변환 [427](#)  
NumRowsAffected  
행 출력 [597](#)  
나노초 처리  
Java 변환 [290](#)  
날짜 값  
무작위 데이터 마스킹 [193](#)  
날짜/시간 값  
데이터 마스킹 [196](#)  
노멀라이저 변환  
GCID [478](#)  
개요 [477](#)  
고급 속성 [491](#)  
다른 개체의 포트를 끌기 [492](#)  
매핑 예제 [493](#)  
매핑 출력 예제 [495](#)  
비원시 환경 [496](#)  
여러 번 발생하는 레코드 [478](#)  
여러 번 발생하는 필드 [478](#)  
예제 사용 사례 [493](#)  
입력 계층 [480](#)  
입력 및 출력 그룹 예제 [494](#)  
입력 포트 [481](#)  
작성 [492](#)  
정의 예제 [494](#)  
출력 그룹 [487](#), [488](#)  
출력 그룹 편집 [490](#)  
키 생성 [491](#)  
필드 병합 [481](#), [482](#)  
하위 레코드 [478](#)  
노멀라이저 보기  
설명 [480](#)  
높은 정밀도 처리  
Java 변환 [290](#)  
누락된 값  
시퀀스 생성기로 바꾸기 [564](#)  
다중 그룹  
변환 [42](#)  
단순 인터페이스  
Java 변환 API 메서드 [325](#)  
Java 식 [325](#)  
예제 [326](#)  
단순 조건 유형  
조이너 변환 [341](#)  
단일 소스 일치 분석 [424](#)  
대/소문자 변환기 변환  
개요 [158](#)  
비원시 환경 [160](#)  
대상  
관계형 [618](#)  
대체 그룹  
SOAP 메시지 구문 분석 [651](#)  
웹 서비스 [663](#)  
웹 서비스 소비자 변환 [628](#), [631](#)  
대체 마스킹  
마스킹 속성 [198](#)  
설명 [197](#)  
데이터  
임시 저장 [47](#)  
데이터 개체  
매개 변수화 [526](#)  
중복 매개 변수화 [385](#)  
데이터 개체 탭  
새 데이터 개체를 사용하여 매개 변수화 [385](#)  
설명 [526](#)  
필드 설명 [382](#)

데이터 마스킹 변환  
IP 주소 마스킹 [207](#)  
URL 마스킹 [208](#)  
고유한 출력 [211](#)  
공유 저장소 테이블 [211](#)  
기본값 파일 [209](#)  
날짜 값 마스킹 [204](#)  
대체 마스킹 [197](#)  
대체 마스킹 속성 [198](#), [199](#)  
대체 마스킹에 대한 사전 [197](#)  
런타임 속성 [211](#)  
마스킹 형식 [202](#)  
마스킹 기술 [192](#)  
무작위 마스킹 [192](#)  
반복 가능 SIN 번호 [209](#)  
반복 가능 SSN [208](#)  
반복 가능한 식 마스킹 [194](#)  
범위 [204](#)  
블러링 [204](#)  
비원시 환경 [215](#)  
사전 이름 식 마스킹 [194](#)  
사회 보장 번호 마스킹 [207](#), [210](#)  
사회 보장 번호 마스킹 [208](#)  
설명 [191](#)  
소스 문자열 문자 [203](#)  
식 마스킹 [194](#)  
식 마스킹 지침 [195](#)  
신용 카드 마스킹 [206](#)  
저장소 커밋 간격 [211](#)  
저장소 테이블 [194](#), [198](#)  
전자 메일 주소 마스킹 [206](#)  
전화 번호 마스킹 [207](#)  
중속 데이터 마스킹 [199](#)  
캐시 디렉터리 [211](#)  
캐시 크기 [211](#)  
특수 마스크 형식 [205](#)  
데이터 유형  
Java 변환 [286](#)  
데이터 처리 변환  
시작 구성 요소 [221](#)  
데이터 캐시  
변환 [70](#)  
데이터 통합 서비스  
다시 시작 모드 [318](#)  
데이터 프로세서 16진수 소스 보기  
설명 [218](#)  
데이터 프로세서 변환  
인코딩 설정 [222](#)  
Blaze 엔진 [237](#)  
XMap 설정 [226](#)  
XML 설정 [226](#)  
데이터 뷰어에서 테스트 [235](#)  
보기 [218](#)  
비원시 환경 [237](#)  
사용자 로그 [230](#)  
서비스 매개 변수 포트 [219](#), [220](#)  
서비스로 내보내기 [235](#)  
설명 [217](#), [218](#)  
설정 [222](#)  
입력 포트 [219](#)  
작성 [231](#)  
처리 설정 [225](#)  
출력 제어 설정 [224](#)  
출력 포트 [220](#)  
포트 [219](#)  
데이터 프로세서 이벤트 보기  
데이터 프로세서 변환 [228](#)

데이터 프로세서 이벤트 보기 (계속)

이벤트 로그 보기 [230](#)

도우미 코드 탭

Java 변환 [296](#)

도우미 탭

Java 변환 [296](#), [297](#)

동기화

사용자 지정된 데이터 개체 [525](#)

실제 데이터 개체 [525](#)

동시 웹 서비스 요청 메시지

웹 서비스 소비자 변환에서 활성화 [634](#)

동적 URL

웹 서비스 소비자 변환 [628](#)

동적 관계형 데이터 개체

읽기 변환 작성 [529](#)

동적 매핑

포트 선택기 [261](#), [389](#)

라우터 변환 [554](#)

분류기 변환 [578](#)

선택 규칙 [261](#), [389](#)

순위 변환 [516](#)

업데이트 전략 변환 [619](#)

조이너 변환 [337](#)

조회 변환 [382](#)

조회 조건 [387](#)

집계 변환 [128](#)

필터 변환 [275](#)

동적 식

예제 [267](#)

개요 [267](#)

작성 [269](#)

출력 포트 설정 [268](#)

동적 조회 캐시

사용 사례 [408](#)

설명 [407](#)

조회 SQL 재정의 [414](#)

플랫 파일 소스 사용 [407](#)

동적 포트

조인 조건 [343](#)

조회 조건 [382](#)

드라이버 점수

일치 변환 [428](#)

디자인

Java 변환 [289](#)

디자인 타임 이벤트 로그

설명 및 위치 [229](#)

라벨러 변환

개요 [359](#)

비원시 환경 [370](#)

라우터 변환

개요 [553](#)

고급 속성 [559](#)

그룹 [554](#)

그룹 필터 조건 [555](#)

동적 매핑 [554](#)

동적 포트 [557](#)

매개 변수 [557](#)

매핑에서 연결 [559](#)

비원시 환경 [559](#)

예제 [555](#)

포트 [558](#)

라이브러리

데이터 프로세서 변환에서 작성 [233](#)

런타임 보기

SQL 변환 [603](#)

런타임 이벤트 로그

데이터 프로세서 변환 [230](#)

레코드

노멀라이저 변환 [478](#)

로그

Java 변환 [317](#), [318](#)

정의 [229](#)

로그, 디자인 타임 이벤트

설명 및 위치 [229](#)

로컬 변수

개요 [47](#)

롤백

Java 변환 API 에서 [319](#)

롤백 행 생성

Java 변환 [319](#)

리포지토리 보기

데이터 프로세서 변환 [218](#)

링크

일대다 [60](#)

일대일 [60](#)

링크 점수

일치 변환 [428](#)

마스크 형식

문자열 값 마스크 [202](#)

특수 마스크 형식 [205](#)

마스킹 규칙

결과 문자열 대체 문자 [203](#)

마스크 형식 [202](#)

범위 [204](#)

블러링 [204](#)

소스 문자열 문자 [203](#)

특수 마스크 형식 [205](#)

마스킹 기술

데이터 마스크 [192](#)

매개 변수

라우터 변환 [557](#)

필터 조건 [276](#)

매개 변수 바인딩

SQL 변환 [594](#)

매개 변수 옵션

조이너 변환 [343](#)

매퍼

설명 [217](#)

매핑

라우터 변환 사용 [559](#)

행에 업데이트 플래그 지정 [619](#)

매핑 매개 변수

조회 SQL 재정의 [375](#)

매핑 변수

조회 SQL 재정의 [375](#)

매핑 실패

Java 변환 [312](#)

맵렛

참조 [221](#)

메서드

Java 변환 API [310](#)

모든 그룹

REST 웹 서비스 소비자 변환에서 보기 [548](#)

웹 서비스 소비자 변환에서 보기 [629](#), [633](#)

모든 요소

웹 서비스 소비자 변환 [628](#), [631](#)

목록 요소

SOAP 메시지 구문 분석 [652](#)

설명 [664](#)

무작위 마스크

날짜 값 마스크 [193](#)

문자열 값 마스크 [192](#)

숫자 값 [192](#)

문자열  
  순위 [516](#)  
문자열 값  
  사용자 지정 데이터 마스킹 [192](#)  
  키 데이터 마스킹 [196](#)  
문자열 대체  
  SQL 변환 [595](#)  
문제 테이블  
  생성 [150](#)  
문제 할당 보기  
  잘못된 레코드 예외 변환 [150](#)  
문제 해결  
  Java 변환 [303](#)  
바인딩  
  WSDL 파일 요소 [625](#)  
버퍼 입력 포트  
  데이터 프로세서 변환 [219](#)  
버퍼 출력 포트  
  데이터 프로세서 변환 [220](#)  
범위  
  마스터 및 세부 정보 [338](#)  
  숫자 값 마스킹 [204](#)  
  포트 선택기 [260, 388](#)  
변수  
  Java 변환 [296](#)  
  개요 [47](#)  
  저장 프로시저 결과, 캡처 [48](#)  
  초기화 [49](#)  
  포트 평가 순서 [48](#)  
변수 길이  
  분류기 변환 내 [582](#)  
변수 포트  
  개요 [47](#)  
  식 변환 [258](#)  
변환  
  Excel에서 포트 구성 [67](#)  
  메타데이터 복사 [67](#)  
  Excel에서 편집 [68](#)  
  Hadoop 환경 [37](#)  
  Java [285](#)  
  개발 [42](#)  
  개요 [33](#)  
  다중 그룹 [42](#)  
  시퀀스 생성기 [560](#)  
  식 [43](#)  
  식 유효성 검사 [45](#)  
  연결 끊김 [34](#)  
  연결됨 [34](#)  
  오류 처리 [53](#)  
  작성 [58](#)  
  재사용 가능 [56, 57](#)  
  재사용 가능 편집 [57](#)  
  재사용 불가능 [58](#)  
  캐시 [69](#)  
  캐시 분할 [73](#)  
  캐시 크기 [71, 72](#)  
  캐시 크기 최적화 [73](#)  
  캐시 파일 [70](#)  
  활성 [33](#)  
변환 범위  
  Java 변환 [290](#)  
변환 언어  
  Java 식 사용 [322](#)  
변환 포트  
  개요 [59](#)  
변환기  
  설명 [217](#)

병합 변환  
  개요 [475](#)  
  비원시 환경 [476](#)  
복합 키  
  REST 웹 서비스 소비자 변환 [546](#)  
  시퀀스 생성기 변환을 사용하여 작성 [563](#)  
  웹 서비스 소비자 변환 [628](#)  
부작용  
  SQL 변환 [599](#)  
  웹 서비스 소비자 변환 [638](#)  
분류기  
  변수 길이 [582](#)  
분류기 변환  
  Blaze 엔진 [585](#)  
  Databricks Spark 엔진 [586](#)  
  Spark 엔진 [585](#)  
  개요 [577](#)  
  동적 매핑 [578](#)  
  비원시 환경 [584](#)  
  정렬 탭 [579](#)  
  캐시 [582](#)  
  캐시 크기 [69](#)  
분류자 변환  
  개요 [161](#)  
  분류자 모델 [162](#)  
  분류자 알고리즘 [162](#)  
  비원시 환경 [169](#)  
블러링  
  날짜 값 [205](#)  
  숫자 값 [204](#)  
비교 변환  
  개요 [170](#)  
  비원시 환경 [174](#)  
비사용자 코드 오류  
  Java 변환 [304](#)  
비정규화된 입력  
  웹 서비스 포트 [660](#)  
비정규화된 출력  
  SOAP 메시지 구문 분석 [648](#)  
사용자 로그  
  데이터 프로세서 변환 [230](#)  
사용자 정의 그룹  
  라우터 변환 [554](#)  
사용자 정의 메서드  
  Java 변환 [296](#)  
사용자 정의 함수  
  Java 식 사용 [322](#)  
사용자 지정 함수  
  Java 식 사용 [322](#)  
사용자 코드 오류  
  Java 변환 [303](#)  
사전  
  대체 데이터 마스킹 [197](#)  
  반복 가능한 식 마스킹 [194](#)  
사전 정보  
  데이터 마스킹 변환 [199](#)  
사회 보장 번호  
  반복 가능 데이터 마스킹 [208](#)  
  지역 번호 마스킹 [208](#)  
삼입 기타 항목 업데이트(속성)  
  설명 [415](#)  
상수  
  null 값을 바꾸기 [51](#)  
상태 비저장  
  Java 변환 [290](#)  
상한 임계값  
  구성 [148, 246](#)



- 샘플 소스 파일
  - 데이터 프로세서 변환에서 정의 [232](#)
- 생성
  - Python 변환 [512](#), [513](#)
- 생성된 키
  - 웹 서비스 출력 그룹 [647](#)
- 서비스
  - WSDL 파일 요소 [625](#)
- 서비스 매개 변수 포트
  - 데이터 프로세서 변환 [219](#)
- 선택 규칙
  - 동적 매핑 [261](#), [389](#)
  - 조이너 변환 [338](#)
  - 포트 선택기 [260](#), [388](#)
- 선택 요소
  - REST 웹 서비스 소비자 변환에서 보기 [548](#)
  - SOAP 메시지 구문 분석 [651](#)
  - 설명 [663](#)
  - 웹 서비스 소비자 변환에서 보기 [629](#), [633](#)
- 선택 조건
  - 포트 선택기 [260](#), [388](#)
- 선택적 실패 이벤트
  - 데이터 프로세서 변환 [228](#)
- 설정 보기
  - 데이터 프로세서 변환 [222](#)
- 성능
  - 변수를 사용하여 성능 개선 [47](#)
- 세션 로그
  - Java 변환 [318](#)
- 세션 실패
  - Java 변환 [312](#)
- 소스 문자열 문자
  - 데이터 마스킹 변환 [203](#)
- 소스 최적화
  - 제약 조건 [529](#)
- 소스 행 필터링
  - 조화 변환 [377](#)
- 속성
  - 쓰기 변환
    - 속성 [668](#)
  - 읽기 변환
    - 속성 [522](#)
- 수동 변환
  - Java [285](#), [286](#)
  - 개요 [34](#)
  - 시퀀스 생성기 [560](#)
- 수동 통합
  - 중복 레코드 예외 변환 [246](#)
- 순위
  - 데이터 그룹 [518](#)
  - 문자열 값 [516](#)
- 순위 변환
  - Blaze 엔진 [521](#)
  - Databricks Spark 엔진 [521](#)
  - RANKINDEX 포트 [517](#)
  - Spark 엔진 [521](#)
  - 개요 [515](#)
  - 고급 속성 [520](#)
  - 그룹 정의 대상 [518](#)
  - 동적 매핑 [516](#)
  - 변수 사용 [47](#)
  - 비원시 환경 [520](#)
  - 순위 포트 [518](#)
  - 옵션 [516](#)
  - 캐시 [519](#)
  - 캐시 크기 [69](#)
  - 포트 [516](#)

- 순위 포트
  - 순위 변환 [518](#)
- 숫자 값
  - 무작위 마스킹 [193](#)
  - 키 마스킹 [196](#)
- 스크립트
  - 데이터 프로세서 변환에서 작성 [232](#)
- 스크립트 도움말 보기
  - 데이터 프로세서 변환 [218](#)
- 스키마
  - 계층-관계형 변환 [282](#)
  - 관계형-계층 변환 [534](#)
  - 데이터 프로세서 변환 [221](#)
- 스트리머
  - 설명 [217](#)
- 시작 값
  - 시퀀스 생성기 변환 [567](#)
- 시작 값(속성)
  - 시퀀스 생성기 변환 [566](#)
- 시작 구성 요소
  - 데이터 프로세서 변환 [221](#)
- 시작 자릿수
  - 사회 보험 번호 [209](#)
- 시퀀스 그룹
  - REST 웹 서비스 소비자 변환에서 보기 [548](#)
- 시퀀스 데이터 개체
  - 속성 [571](#)
  - 작성 [572](#)
- 시퀀스 생성기 변환
  - Blaze 엔진 [576](#)
  - CURRVAL 포트 [565](#)
  - IIF 함수를 사용하여 누락된 키 바꾸기 [564](#)
  - NEXTVAL 포트 [561](#)
  - Spark 엔진 [576](#)
  - 값 범위 [569](#)
  - 개요 [560](#)
  - 복합 키 작성 [563](#)
  - 비원시 환경 [576](#)
  - 속성 [566](#), [571](#)
  - 시작 값 [567](#)
  - 작성 [573](#), [574](#)
  - 재사용 가능 [570](#)
  - 재사용 불가능 [570](#)
  - 재설정 [571](#)
  - 주기 [567](#), [569](#)
  - 중분 범위 속성 [568](#)
  - 총 개시된 값 [569](#)
  - 포트 [561](#)
  - 행 순서 유지 [571](#)
  - 현재 값 [569](#)
- 식
  - Java 변환 [322](#)
  - 단순화 [47](#)
  - 변환 내 [43](#)
  - 설명 추가 [45](#)
  - 유효성 검사 [45](#)
  - 입력 [44](#)
  - 포트에 추가 [44](#)
- 식 마스킹
  - 규칙 및 지침 [195](#)
  - 반복 가능 마스킹 [194](#)
  - 반복 가능한 마스킹 예 [195](#)
  - 설명 [194](#)
- 식 매개 변수
  - 조이너 변환의 [343](#)
  - 조화 조건 [390](#)

## 식 변환

Blaze 엔진 [272](#)  
Databricks Spark 엔진 [272](#)  
Spark 엔진 [272](#)  
개요 [257](#)  
고급 속성 [271](#)  
동적 식 [267](#)  
변수 사용 [47](#)  
비원시 환경 [272](#)  
출력 포트 설정 [268](#)  
테스트 [259](#)  
포트 유형 [258](#)

## 식 편집기

Java 식 사용 [324](#)  
설명 [44](#)  
식 유효성 검사 [45](#)  
식 테스트 [259](#)

## 실제 데이터 개체

동기화 [525](#)

## 실패 이벤트

데이터 프로세서 변환 [228](#)

## 쓰기 변환

개요 [668](#)

## 알림 이벤트

데이터 프로세서 변환 [228](#)

## 업데이트 기타 항목 삽입(속성)

설명 [415](#)

## 업데이트 전략 변환

Blaze 엔진 [621](#)  
Spark 엔진 [622](#)  
개요 [618](#)  
거부된 행 전달 [620](#)  
고급 속성 [620](#)  
구성 단계 [618](#)  
동적 매핑 [619](#)  
비원시 환경 [621](#)  
식 [619](#)  
작성 [619](#)  
집계 조합 [620](#)

## 여러 번 발생하는 필드

노멀라이저 변환 [478](#)

## 연결

REST 웹 서비스 [549](#)

웹 서비스 [634](#)

## 연결되지 않은 변환

조희 변환 [374](#)

## 연결되지 않은 조회

개요 [374](#)

설명 [372](#)

## 연결된 변환

Java [285](#)

순위 [515](#)

시퀀스 생성기 [560](#)

## 연결된 조회

개요 [373](#)

설명 [372](#)

## 연관 변환

개요 [140](#)

고급 속성 [141](#)

추적 수준 [141](#)

## 영향을 받는 행 수

SQL 변환 [591](#)

## 예

파티션 및 순서 키 [265](#)

## 예약어

조회 쿼리 [376](#)

## 예외 변환

문제 할당 보기 [150](#)

## 예제

동적 식 [267](#)

## 예제 소스

데이터 프로세서 변환에서 작성 [234](#)

## 예제 입력 파일

데이터 프로세서 변환 [219](#)

## 오류

Java 변환의 임계값 증가 [315](#)

처리 [53](#)

## 오류 수

Java 변환에 대해 증분 [315](#)

## 외래 키

시퀀스 생성기 변환을 사용하여 작성 [563](#)

## 요소

합집합 [664](#)

## 웹 서비스

anyType에 포트 매핑 [662](#)

대체 그룹 [663](#)

파생된 유형 [662](#)

## 웹 서비스 변환

위치 열 [655](#)

## 웹 서비스 소비자 변환

HTTP 오류 출력 활성화 [634](#)

HTTP 헤더 추가 [627](#)

SOAP 메시지 [624](#)

SOAP 압축 [637](#)

TLS(Transport Layer Security) [626](#)

개요 [624](#)

고급 속성 [634](#)

끝점 URL [628](#)

동시 웹 서비스 요청 메시지 [634](#)

동적 WS-Security 이름 [628](#)

동적 웹 서비스 URL [628](#)

보안 [626](#)

오류 처리 [636](#)

일반 SOAP 결함 [636](#)

일반 결함 출력 활성화 [634](#)

입력 매핑 [628](#)

입력 포트 매핑 [628](#)

작성 [640](#)

작업 [625](#)

초기 선택 최적화 [638](#)

출력 노드 매핑 [631](#)

출력 매핑 [631](#)

쿠키 인증 [628](#)

키 보기 [629](#), [633](#)

푸시인 최적화 [638](#)

푸시인 최적화 활성화 [640](#)

필터 최적화 [638](#)

## 웹 서비스 연결

개요 [634](#)

## 위치 열

웹 서비스 변환 [655](#)

## 유효성 검사

기본값 [53](#), [56](#)

## 유효성 검사 규칙 개체

데이터 프로세서 변환에서 작성 [234](#)

## 유효성 검사 단추

변환 [53](#), [56](#)

## 응답 코드

REST 웹 서비스 소비자 변환 [545](#)

## 이벤트

데이터 프로세서 변환 [228](#)

## 이벤트 로그

보기 [230](#)

이벤트 로그, 디자인 타임  
  설명 및 위치 [229](#)  
이벤트 보기  
  데이터 프로세서 변환 [228](#)  
이벤트 보기, 데이터 프로세서  
  이벤트 로그 보기 [230](#)  
이벤트 유형  
  데이터 프로세서 변환 [228](#)  
이중 소스 일치 분석 [424](#)  
인덱스 캐시  
  변환 [70](#)  
인스턴스 변수  
  Java 변환 [296](#)  
인코딩 설정  
  데이터 프로세서 변환 [222](#)  
인코딩 지침  
  데이터 프로세서 변환 [224](#)  
일반 SOAP 결함  
  웹 서비스 소비자 변환 [636](#)  
일반 결함 출력  
  웹 서비스 소비자 변환에서 활성화 [634](#)  
일치 변환  
  Blaze 엔진 [443](#)  
  ID 분석 프로세스 흐름 [456](#)  
  ID 인덱스 테이블 [430](#)  
  null 일치 점수 [427](#)  
  Spark 엔진 [443](#)  
  그룹화된 데이터 [425](#)  
  단일 소스 분석 [424](#)  
  드라이버 점수 [428](#)  
  링크 점수 [428](#)  
  미리 정의된 입력 포트 [437](#)  
  미리 정의된 출력 포트 [438](#)  
  비원시 환경 [443](#)  
  사용 사례 [422](#)  
  샘플 일치 전략 [252](#)  
  성능 데이터 보기 [433](#)  
  성능 요인 [431](#)  
  이중 소스 분석 [424](#)  
  일치 분석 작업 구성 [443](#)  
  일치 분석에 대한 맵렛 작성 [442](#)  
  일치 분석의 개념 [423](#)  
  일치 출력 보기 속성 [448](#)  
  일치 출력 보기 옵션 [448](#)  
  일치 출력 보기의 일치 속성 [464](#)  
  일치 출력 보기의 출력 속성 [464](#)  
  정의된 ID 분석 [424](#)  
  정의된 필드 분석 [424](#)  
  지속성 상태 설명 [439](#)  
  지속성 상태 코드 [439](#)  
  출력 형식 [462](#)  
  클러스터 분석 데이터 보기 [432](#)  
  클러스터 점수 옵션 [427](#)  
  필드 분석 프로세스 흐름 [444](#)  
일치 분석의 그룹 [425](#)  
일치 전략  
  중복 레코드 예외 변환 [252](#)  
읽기 데이터 개체  
  매개 변수화 [527](#)  
읽기 변환  
  개요 [522](#)  
  관계형 데이터 개체에서 작성 [529](#)  
입력 계층  
  노멀라이저 변환 [480](#)  
입력 매핑  
  REST 웹 서비스 소비자 변환 [545](#)  
  웹 서비스 소비자 변환 [628](#)

입력 시 탭  
  Java 변환 [297](#)  
입력 포트  
  Java 변환 [289](#), [290](#)  
  기본값 [49](#)  
입력 포트 영역  
  SOAP 메시지 생성 [654](#)  
입력 행  
  행 유형 가져오기 [314](#)  
자동 캐시 크기  
  설명 [71](#)  
자동 통합  
  중복 레코드 예외 변환 [246](#)  
작성  
  Java 변환 [301](#), [302](#)  
  시퀀스 데이터 개체 [572](#)  
  시퀀스 생성기 변환 [573](#), [574](#)  
작업  
  WSDL 파일 요소 [625](#)  
작업 영역  
  웹 서비스 변환 [655](#)  
작업 입력 영역  
  웹 서비스 소비자 변환 사용자 지정 [629](#)  
작업 출력 영역  
  웹 서비스 소비자 변환 사용자 지정 [633](#)  
잘못된 레코드 예외 변환  
  개요 [143](#)  
  고급 속성 [151](#)  
  구성 [151](#)  
  구성 보기 [148](#)  
  매핑 [145](#)  
  빈 품질 문제 포트 [144](#)  
  양호한 레코드 예외의 출력 [157](#)  
  예제 [152](#)  
  예제의 맵렛 [152](#)  
  예제의 문제 출력 [156](#)  
  예제의 잘못된 레코드 출력 [155](#)  
  예제의 출력 [155](#)  
  입력 포트 [147](#)  
  출력 그룹 [144](#)  
  출력 포트 [148](#)  
  포트 그룹 [147](#)  
  품질 문제 [146](#)  
  프로세스 흐름 [144](#)  
잘못된 레코드 테이블 생성  
  잘못된 레코드 예외 변환 [150](#)  
재사용 가능  
  시퀀스 생성기 변환 [571](#)  
재사용 가능한 변환  
  설명 [56](#)  
  편집 [57](#)  
재사용 불가능 변환  
  설명 [58](#)  
재설정  
  시퀀스 생성기 변환 [571](#)  
재설정(속성)  
  시퀀스 생성기 변환 [566](#)  
저장 프로시저  
  SQL 변환 [603](#)  
  결과 집합 예제 [605](#)  
  매개 변수 [604](#)  
  반환 값 [605](#)  
  반환 값 포트 [604](#)  
  변수에 기록 [48](#)  
  예제 [607](#)  
  입력 및 출력 포트 [604](#)

- 저장소 암호화
  - 데이터 마스킹 변환 [211](#)
- 저장소 암호화 키
  - 데이터 마스킹 변환 [211](#)
- 저장소 커밋 간격
  - 데이터 마스킹 변환 [211](#)
- 저장소 테이블
  - 대체 데이터 마스킹 [198](#)
  - 식 마스킹 [194](#)
- 전체 코드 탭
  - Java 변환 [299](#)
  - Java 컴파일 오류 [303](#)
- 정렬 속성
  - 중복 레코드 예외 변환 [250](#)
  - 통합 변환 [177](#)
- 정렬 키
  - 구성 [579](#)
- 정렬 탭
  - 분류기 변환 [579](#)
- 정적 변수
  - Java 변환 [296](#)
- 정적 조회 캐시
  - 개요 [403](#)
- 정적 코드
  - Java 변환 [296](#)
- 제약 조건
  - 소스 최적화 [529](#)
- 조건
  - 라우터 변환 [555](#)
  - 조이너 변환 [340](#)
- 조건 유형
  - 단순 조이너 변환 [341](#)
  - 조이너 변환의 고급 조건 [341](#)
- 조이너 변환
  - Blaze 엔진 [353](#)
  - Databricks Spark 엔진 [353](#)
  - Spark 엔진 [353](#)
  - 개요 [334](#)
  - 고급 속성 [335](#)
  - 고급 조건 유형 [341](#)
  - 규칙 및 지침 [352](#)
  - 단순 조건 유형 [341](#)
  - 동일한 소스의 데이터 조인 [349](#)
  - 동적 매핑 [337](#)
  - 동적 포트 [343](#)
  - 마스터 외부 조인 [345](#)
  - 마스터 행 캐싱 [351](#)
  - 비원시 환경 [353](#)
  - 선택 규칙 [338](#)
  - 성능 [352](#)
  - 세부 외부 조인 [345](#)
  - 소스 파이프라인 차단 [351](#)
  - 식 매개 변수 [343](#)
  - 일반 조인 [344](#)
  - 전체 외부 조인 [345](#)
  - 정렬 순서 구성 [346](#)
  - 정렬되지 않음 [351](#)
  - 정렬된 입력 [346](#)
  - 정렬됨 [351](#)
  - 조건 [340](#)
  - 조건 유형 [341](#)
  - 조인 유형 [343](#)
  - 캐시 [336](#)
  - 캐시 크기 [69](#)
  - 포트 [337](#)
  - 포트 선택기 [338](#)
  - 포트 선택기 사용 [342](#)

- 조인 조건
  - 개요 [340](#)
- 조회
  - 캐싱되지 않음 [402](#)
- 조회 SQL 재정의
  - 동적 캐시 [414](#)
- 조회 변환
  - Blaze 엔진 [399](#)
  - Databricks Spark 엔진 [400](#)
  - Spark 엔진 [399](#)
  - 개발 [374](#)
  - 개요 [371](#)
  - 고급 속성 [393](#)
  - 관계형 매개 변수화된 데이터 개체 [385](#)
  - 기본 쿼리 [374](#)
  - 동적 매핑 [382](#)
  - 동적 캐시 [407](#)
  - 동적 포트 [382](#)
  - 런타임 속성 [392](#)
  - 매핑 매개 변수 및 변수 [375](#)
  - 비원시 환경 [399](#)
  - 연결 끊김 [374](#)
  - 연결되지 않은 조회 작성 [397](#)
  - 연결되지 않음 [372](#)
  - 연결됨 [372, 373](#)
  - 재사용 가능 변환 작성 [395](#)
  - 재사용 불가능 변환 작성 [396](#)
  - 조회 재정의 [377](#)
  - 조회 조건 [378](#)
  - 조회 조건 규칙 및 지침 [380](#)
  - 조회 쿼리 재정의 [375](#)
  - 지속형 캐시 [403](#)
  - 캐시 [401](#)
  - 캐시 크기 [69](#)
  - 캐시 크기, 줄이기 [375](#)
  - 캐싱 [381](#)
  - 캐싱되지 않은 조회 [402](#)
  - 쿼리 속성 [381](#)
  - 포트 이름 충돌 [384](#)
- 조회 소스
  - 매개 변수화 [385](#)
- 조회 소스 필터
  - 조회 제한 [377](#)
- 조회 재정의
  - 지침 [376](#)
- 조회 조건
  - 데이터 마스킹 변환 [199](#)
  - 매개 변수로 지정 [390](#)
  - 생성된 포트 사용 [387](#)
  - 설명 [379](#)
- 조회 캐시
  - 개요 [381, 401](#)
  - 동적 [407](#)
  - 정의 [401](#)
  - 정적 [403](#)
  - 지속형 [403](#)
- 조회 쿼리
  - ORDER BY [375](#)
  - 기본 쿼리 [375](#)
  - 설명 [374](#)
  - 예약어 [376](#)
  - 재정의 [375, 377](#)
  - 재정의 지침 [376](#)
- 종속 마스킹
  - 설명 [199](#)
- 종속 열
  - 데이터 마스킹 [199](#)

- 종속성
  - 링크 경로 [64](#)
  - 암시적 [64](#)
- 주기
  - 시퀀스 생성기 변환 속성 [567](#)
- 주기(속성)
  - 시퀀스 생성기 변환 속성 [566](#)
- 주소 유효성 검사기 변환
  - 비원시 환경 [125](#)
- 중복 레코드 예외 변환
  - 개요 [245](#)
  - 고급 속성 [250](#)
  - 구성 [256](#)
  - 구성 보기 [246](#)
  - 레코드 정렬 [250](#)
  - 매핑 예제 [251](#)
  - 샘플 출력 [254](#)
  - 예제 [251](#)
  - 예제의 구성 설정 [253](#)
  - 자동 통합 [246](#)
  - 중복 레코드 테이블 생성 [248](#)
  - 추적 수준 [250](#)
  - 출력 그룹 [249](#)
  - 클러스터 [246](#)
  - 클러스터 출력 예제 [255](#)
  - 포트 [248](#)
  - 표준 출력 그룹 [249](#)
- 증분
  - 시퀀스 간격 설정 [568](#)
- 증분 범위(속성)
  - 시퀀스 생성기 변환 속성 [566](#)
- 지속형 조회 캐시
  - 개요 [403](#)
- 집계 변환
  - Blaze 엔진 [138](#)
  - Databricks Spark 엔진 [139](#)
  - Spark 엔진 [138](#)
  - 개발 [128](#)
  - 개요 [127](#)
  - 고급 속성 [135](#)
  - 그룹 기준 포트 [131](#)
  - 데이터 정렬 [134](#)
  - 동적 매핑 [128](#)
  - 문제 해결 [137](#)
  - 변수 사용 [47](#)
  - 비원시 환경 [138](#)
  - 비집계 식 [133](#)
  - 업데이트 전략 조합 [620](#)
  - 재사용 가능 변환 작성 [136](#)
  - 재사용 불가능 변환 작성 [136](#)
  - 정렬된 입력 [134](#)
  - 중첩 집계 함수 [130](#)
  - 집계 식 [129](#)
  - 집계 함수 [130](#)
  - 캐시 [133](#)
  - 캐시 크기 [69](#)
  - 팁 [137](#)
  - 포트 [128](#)
- 차단 변환
  - 설명 [42](#)
- 참조 보기
  - 계층-관계형 변환 [282](#)
  - 관계형-계층 변환 [534](#)
  - 데이터 프로세서 변환 [221](#)
- 창 작업
  - 속성 [264](#)
- 창 작업 속성
  - 규칙 및 지침 [267](#)
  - 순서 [264](#), [265](#)
  - 파티션 [264](#), [265](#)
  - 프레임 [264](#)
- 찾기
  - Java 코드의 오류 [303](#)
- 초기 선택 최적화
  - SQL 변환 [599](#)
  - 웹 서비스 소비자 변환 [638](#)
- 초기화 중
  - 변수 [49](#)
- 총 캐시된 값
  - 시퀀스 생성기 변환 속성 [569](#)
  - 시퀀스 생성기 속성 값 [566](#)
- 최대 출력 행 개수
  - SQL 변환 [595](#), [596](#)
- 추적 수준
  - 시퀀스 생성기 변환 속성 [566](#)
  - 연관 변환 [141](#)
  - 중복 레코드 예외 변환 [250](#)
  - 키 생성기 변환 [358](#)
  - 통합 변환 [177](#)
- 출력 그룹
  - 노멀라이저 변환 [488](#)
  - 잘못된 레코드 예외 변환 [144](#)
- 출력 그룹 편집 대화 상자
  - 노멀라이저 변환 [490](#)
- 출력 매핑
  - REST 웹 서비스 소비자 변환 [547](#)
  - 웹 서비스 소비자 변환 [631](#)
- 출력 제어 설정
  - 데이터 프로세서 변환 [224](#)
- 출력 포트
  - Java 변환 [289](#), [290](#)
  - 기본값 [49](#)
  - 오류 처리 [49](#)
- 출력 포트 설정
  - 설명 [268](#)
- 출력 행
  - Java 변환에서 행 유형 설정 [320](#)
- 출력 행 생성
  - Java 변환 [313](#)
- 치명적 오류 이벤트
  - 데이터 프로세서 변환 [228](#)
- 캐시
  - 개요 [69](#)
  - 데이터 [70](#)
  - 데이터 통합 서비스에 의해 증가된 크기 [72](#)
  - 동적 조회 캐시 [407](#)
  - 변환 [69](#)
  - 분할 [73](#)
  - 유형 [70](#)
  - 인덱스 [70](#)
  - 정적 조회 캐시 [403](#)
  - 조회 변환 [381](#), [401](#)
  - 캐시 파일 [69](#)
  - 크기 [71](#)
  - 크기 최적화 [73](#)
  - 파일 디렉터리 [71](#)
- 캐시 디렉터리
  - 데이터 마스킹 변환 [211](#)
- 캐시 크기
  - 데이터 마스킹 변환 [211](#)
  - 자동 [71](#)
- 캐시 파일
  - 개요 [70](#)

- 캐시 파일 (계속)
  - 디렉터리 [71](#)
- 캐시 파일 디렉터리
  - 연관 변환 [141](#)
  - 중복 레코드 예외 변환 [250](#)
  - 키 생성기 변환 [358](#)
  - 통합 변환 [177](#)
- 캐시 파일 크기
  - 연관 변환 [141](#)
  - 중복 레코드 예외 변환 [250](#)
  - 키 생성기 변환 [358](#)
  - 통합 변환 [177](#)
- 커밋
  - Java 변환 API 메서드 [311](#)
- 컴파일
  - Java 변환 [302](#)
- 컴파일 오류
  - Java 변환의 소스 식별 [303](#)
- 코드 조각
  - Java 변환에 대해 작성 [295](#)
- 쿠키 인증
  - REST 웹 서비스 소비자 변환 [545](#)
  - 웹 서비스 소비자 변환 [628](#)
- 쿼리
  - 조회 변환 [374](#)
  - 조회 재정의 [375](#)
- 클래스 경로
  - 매핑 속성 [293](#)
- 클러스터
  - 중복 레코드 예외 변환 [246](#)
- 클러스터 데이터
  - 출력 그룹 [249](#)
- 키
  - SOAP 메시지 계층 [656](#)
  - 시퀀스 생성기 변환을 사용하여 작성 [563](#)
- 키 마스킹
  - 날짜/시간 값 마스킹 [196](#)
  - 문자열 값 마스킹 [196](#)
  - 설명 [195](#)
  - 숫자 값 [195](#)
  - 숫자 값 마스킹 [196](#)
- 키 생성기 변환
  - 개요 [354](#)
  - 고급 속성 [358](#)
  - 비원시 환경 [358](#)
  - 추적 수준 [358](#)
- 통과 포트
  - SQL 변환 [590](#)
  - 기본값 [49](#)
  - 식 변환 [258](#)
- 통합 변환
  - 개요 [175](#)
  - 고급 속성 [177](#)
  - 레코드 정렬 [177](#)
  - 비원시 환경 [189](#)
  - 추적 수준 [177](#)
- 트랜잭션
  - 생성 [311](#)
- 트랜잭션 생성
  - Java 변환 [311](#)
- 특수 형식 마스킹
  - IP 주소 [207](#)
  - URL [208](#)
  - 반복 가능 SIN 번호 [209](#)
  - 사회 보장 번호 [207](#)
  - 사회 보험 번호 [208](#)
  - 신용 카드 번호 [206](#)

- 특수 형식 마스킹 (계속)
  - 전자 메일 주소 [206](#)
  - 전화 번호 [207](#)
- 파생된 유형
  - SOAP 메시지 구문 분석 [650](#)
  - 웹 서비스 [662](#)
- 파생된 유형 요소
  - 웹 서비스 소비자 변환 [628](#), [631](#)
- 파서
  - 설명 [217](#)
- 파서 변환
  - 개요 [497](#)
  - 비원시 환경 [507](#)
- 파일 입력 포트
  - 데이터 프로세서 변환 [219](#)
- 파일 출력 포트
  - 데이터 프로세서 변환 [220](#)

## O

- ORDER BY
  - 재정의 [375](#)
  - 조회 쿼리 [375](#)

## P

- Python
  - 고급 속성 [510](#)
  - 구성 요소 [510](#)
  - 규칙 [512](#)
  - 데이터 유형 [508](#)
  - 리소스 파일 [510](#), [511](#)
  - 미리 학습된 모델 [510](#)
  - 사용 사례 [513](#)
  - 생성 [512](#)
  - 스크립트 [510](#)
  - 예 [513](#)
    - 입력 포트
      - 데이터 유형 [509](#)
  - 지침 [512](#)
    - 출력 포트
      - 데이터 유형 [509](#)
  - 코드 [511](#)
- Python 변환
  - Spark 엔진 [514](#)
  - 개요 [508](#)
  - 비원시 환경 [514](#)
  - 작성 [512](#), [513](#)

## Q

- Qname 요소
  - SOAP 메시지 구문 분석 [651](#)

## R

- resetNotification 메서드
  - Java 변환 [318](#)
- REST 웹 서비스 소비자 변환
  - Delete 메서드 [542](#)
  - Get 메서드 [540](#)
  - HTTP 메서드 [540](#)
  - HTTP 헤더 포트 [544](#)

## REST 웹 서비스 소비자 변환 (계속)

- Post 메서드 [541](#)
- Put 메서드 [542](#)
- RequestInput 포트 [543](#)
- TLS(Transport Layer Security) [626](#)
- URL 포트 [544](#)
- XML 스키마 유효성 검사 [549](#)
- 개요 [537](#)
- 고급 속성 [549](#)
- 구성 [539](#)
- 기본 URL 설정 [549](#)
- 리소스 식별 [539](#)
- 매핑 입력 [537](#)
- 매핑 출력 [537](#)
- 메시지 구성 [539](#)
- 보안 [626](#)
- 연결 속성 [549](#)
- 응답 코드 포트 [545](#)
- 인수 포트 [544](#)
- 인터넷 미디어 유형 [549](#)
- 입력 매핑 [545](#)
- 입력 매핑 규칙 [546](#)
- 입력 포트 [543](#)
- 입력 포트 매핑 [546](#)
- 작성 [550](#)
- 재사용 가능 [550](#)
- 재사용 불가능 [550](#)
- 정렬된 입력 [549](#)
- 추적 수준 [549](#)
- 출력 XML 포트 [545](#)
- 출력 매핑 [547](#)
- 출력 매핑 규칙 [548](#)
- 출력 매핑 보기 사용자 지정 [548](#)
- 출력 포트 [543](#)
- 출력 포트 매핑 [549](#)
- 쿠키 포트 [545](#)
- 통과 포트 [544](#)
- 포트 [543](#)
- 포트에 요소 매핑 [543](#)
- 프로세스 [538](#)
- 프록시 서버 지원 [537](#)

## S

- setNull 메서드
  - Java 변환 [319](#)
- setOutRowType
  - Java 변환 API 메서드 [320](#)
- SIN 번호
  - 반복 가능 데이터 마스킹 [209](#)
  - 사회 보험 번호 마스킹 [208](#)
- SOAP 계층
  - 입력 포트에 대한 관계 [655](#)
- SOAP 메시지
  - anyType 요소 구문 분석 [649](#)
  - 개요 [624](#)
  - 다중 발생 노드 매핑 [646](#)
  - 대체 그룹 구문 분석 [651](#)
  - 데이터 피벗 [659](#)
  - 목록 요소 구문 분석 [652](#)
  - 목록 요소 매핑 [664](#)
  - 선택 요소 구문 분석 [651](#)
  - 선택 요소 매핑 [663](#)
  - 여러 입력 포트 매핑 [659](#)
  - 키 [656](#)
  - 포트 매핑 [657](#)

## SOAP 메시지 (계속)

- 포트를 합집합 요소에 매핑 [664](#)
- SOAP 메시지 구문 분석
  - Qname 요소 [651](#)
  - 비정규화된 출력 [648](#)
  - 설명 [645](#)
  - 정규화된 출력 [647](#)
  - 파생된 유형 [650](#)
  - 피벗된 출력 [648](#)
  - 합집합 요소 [652](#)
- SOAP 압축
  - 웹 서비스 소비자 변환 [637](#)
- SOAP 작업
  - 웹 서비스 소비자 변환에서 재정의 [634](#)
- SOAP 작업 재정의
  - 웹 서비스 소비자 변환 [634](#)
- SQL 변환
  - INOUT 매개 변수 [604](#)
  - SQL 오류에 대해 계속 [598](#)
  - SQLException 포트 [591](#)
  - 개요 [587](#)
  - 결과 집합 [605](#)
  - 고급 속성 보기 [592](#)
  - 데이터베이스 연결 정의 [603](#)
  - 런타임 연결 [608](#)
  - 매개 변수 바인딩 [594](#)
  - 반환 값 포트 [604](#)
  - 빈 항목으로 작성 [609](#)
  - 영향을 받는 행 수 [591](#)
  - 예제 [600](#)
  - 입력 포트 설명 [588](#)
  - 입력 행 대 출력 행의 카디널리티 [595](#)
  - 저장 프로시저 [603](#)
  - 저장 프로시저 매개 변수 [604](#)
  - 저장 프로시저 호출 [604](#)
  - 저장 프로시저에서 작성 [610](#)
  - 초기 선택 최적화 [599](#)
  - 출력 포트 정의 [589](#)
  - 출력 행 제한 [596](#)
  - 쿼리 문 [602](#)
  - 쿼리 문자열 대체 [595](#)
  - 쿼리 정의 [594](#)
  - 통과 포트 [590](#)
  - 포트 [588](#)
  - 푸시인 최적화 [599](#)
  - 푸시인 최적화 속성 [600](#)
  - 행 출력 수 [597](#)
- SQL 오류에 대해 계속
  - SQL 변환 [595](#), [598](#)
- SQL 입력 포트
  - SQL 변환 [588](#)
- SQL 쿼리
  - SQL 변환 [594](#)
- SQLException 포트
  - SQL 변환 [591](#)
- storeMetadata 메서드
  - Java 변환 [321](#)

## T

- TLS(Transport Layer Security)
  - REST 웹 서비스 소비자 변환 [626](#)
  - 웹 서비스 소비자 변환 [626](#)

## W

WS-Security 사용자 이름

동적 포트 [628](#)

WSDL 파일

바인딩 요소 [625](#)

서비스 요소 [625](#)

작업 요소 [625](#)

포트 요소 [625](#)

## X

XMap 개체

데이터 프로세서 변환에서 작성 [233](#)

## ㄱ

가져오기 탭

Java 변환 [295](#), [297](#)

가중치 평균 변환

비원시 환경 [667](#)

개의 포트

변환의 전달된 특성 [65](#)

수동으로 연결 [61](#)

연결 [60](#)

연결 규칙 및 지침 [63](#)

위치별로 연결 [62](#)

이름별로 연결 [61](#)

자동 연결 [61](#)

거부 파일

업데이트 전략 [620](#)

거부된 레코드

잘못된 레코드 예외 변환 [148](#)

거부된 행 전달

구성 [620](#)

옵션 [620](#)

결과 문자열 대체 문자

데이터 마스킹 변환 [203](#)

결과 집합

SQL 변환 [605](#)

샘플 저장 프로시저 [605](#)

출력 매개 변수 배치 [607](#)

결정 변환

개요 [238](#)

고급 속성 [243](#)

비원시 환경 [244](#)

품질 문제 샘플 전략 [146](#)

결함을 오류로 처리

웹 서비스 소비자 변환에서 활성화 [634](#)

경고 이벤트

데이터 프로세서 변환 [228](#)

계산

변수 사용 [48](#)

계층-관계형 변환

개발 [282](#)

데이터 뷰어에서 테스트 [283](#)

설명 [279](#)

예제 [279](#)

작성 [282](#)

포트 [281](#)

포트 구성 [282](#)

포트 작성 [283](#)

고급 속성

Java 변환 [290](#)

REST 웹 서비스 소비자 변환 [549](#)

고급 속성 (계속)

SQL 변환 [592](#)

연관 변환 [141](#)

웹 서비스 소비자 변환 [634](#)

잘못된 레코드 예외 변환 [151](#)

중복 레코드 예외 변환 [250](#)

키 생성기 변환 [358](#)

통합 변환 [177](#)

고급 인터페이스

EDatatype 클래스 [328](#)

Java 식 [326](#)

Java 식 호출 [327](#)

JExpression 클래스 [329](#), [331](#)

JExprParamMetadata 클래스 [328](#)

예제 [330](#)

고급 조건 유형

조이너 변환 [341](#)

고려 사항

Java 변환 [289](#)

고유 출력

분류기 변환 [579](#)

고유한 레코드 테이블

작성 [256](#)

고유한 출력

데이터 마스킹 변환 [199](#)

공유 저장소 테이블

데이터 마스킹 변환 [211](#)

공유 조회 캐시

분할, 지침 [405](#)

지침 [405](#)

관계형 데이터 개체

동적 읽기 변환 작성 [529](#)

관계형-계층 변환

개발 [535](#)

설명 [531](#)

예제 [531](#)

작성 [535](#)

포트 [534](#)

포트 작성 [535](#)

구성 보기

잘못된 레코드 예외 변환 [148](#)

잘못된 레코드 예외 변환 예제 [154](#)

중복 레코드 예외 변환 [246](#)

구성 요소 보기

데이터 프로세서 변환 [218](#)

규칙

기본값 [55](#)

규칙 및 지침

창 작업 속성 [267](#)

그룹

라우터 변환 [554](#)

라우터 변환에 추가 [558](#)

사용자 정의 [554](#)

그룹 기준 탭

설명 [132](#)

포트 목록 매개 변수 [132](#), [519](#)

그룹 필터 조건

라우터 변환 [555](#)

기본 그룹

라우터 변환 [554](#)

기본 키

시퀀스 생성기 변환을 사용하여 작성 [563](#)

기본값

개요 [49](#)

규칙 [55](#)

데이터 마스킹 [209](#)

사용자 정의 [50](#)



기본값 (계속)

유효성 검사 [53](#), [56](#)

입력 [53](#), [56](#)

입력 포트 [49](#), [50](#)

출력 포트 [49](#), [50](#)

통과 포트 [49](#), [50](#)

## ㄱ

끝 값(속성)

시퀀스 생성기 변환 [566](#)

끝에서 탭

Java 변환 [298](#)

끝점 URL

REST 웹 서비스 소비자 변환 [544](#)

웹 서비스 소비자 변환 [628](#)