
CS
TEKLYNX
CODESOFT
TEKLYNX
CODESOFT
Barcode Better
powered by:
TEKLYNX
Barcode Better

자습서

본 설명서에 포함된 정보는 계약 상의 성격을 띠지 않으며 사전 통지 없이 변경될 수 있습니다.

본 설명서에 설명된 소프트웨어는 라이센스 계약 하에 판매됩니다. 계약 조건 하에서만 소프트웨어를 사용, 복사 또는 재생산할 수 있습니다.

본 설명서의 어떠한 부분도 TEKLYNX Corporation SAS 의 서면 승인 없이 어떠한 형식으로도 또는 구매자 자신의 사용 용도 이외의 목적으로 복사, 재생산 또는 전송할 수 없습니다.

목차

설명서 소개

표기법

본 설명서에서는 다음과 같은 표기법을 사용하여 서로 다른 유형의 정보를 구분합니다.

- 명령과 같은 인터페이스 용어는 굵게 표시합니다.
- 키는 작은 대문자로 표시됩니다. 예를 들면 다음과 같습니다. "SHIFT 키를 누릅니다."
- 번호가 매겨진 목록은 사용자가 수행해야 할 절차가 있음을 나타냅니다.
- 문단 다음에 -또는-이라는 연결어가 오면, 해당 작업에 대한 다른 절차를 선택할 수 있다는 것을 의미합니다.
- 메뉴 명령에 하위 메뉴가 있는 경우에는 선택할 명령이 메뉴 이름 뒤에 굵은 글씨로 표시됩니다. 따라서, "**File(파일) > Open(열기)**로 이동"은 **File(파일)** 메뉴를 선택한 다음 **Open(열기)** 명령을 선택하는 것을 의미합니다.

제품에 대하여

본 설명서에 설명된 기능 중 일부가 사용자가 구입한 제품에 없을 수 있습니다.

소프트웨어에서 사용할 수 있는 특정 기능의 전체 목록은 제품과 함께 제공된 제원표를 참조하십시오.

데이터베이스 연결

몇 가지 참고 사항

이 장에서는 이 소프트웨어의 기능이 얼마나 강력한지를 보 여줍니다. 이제 ODBC 연결(Open Data Base Connectivity) 및 OLE DB(Object Linking and Embedding Database)를 사용하여 레이블(컨테이너)을 데이터베이스(내용)에 연결하려 고 합니다.

데이터베이스를 사용하면 데이터를 저장할 수 있으며 관계라는 2 차원 테이블로 구성되어 있습니다. 테이블의 각 행을 레코드라고 합니다. 레코드는 개체를 관리하기 위해 사용되며 개체의 등록 정보는 데이터베이스 테이블의 각 열에 필드 형태로 존재합니다.

데이터베이스에는 여러 테이블을 포함할 수 있습니다. 데이터베이스의 서로 다른 테이블을 연결하기 위해 조인을 사용 합니다. 이 장의 뒷부분에서 조인을 작성하는 방법을 구체적인 예를 들어서 보여줄 것입니다.

ODBC

데이터베이스 액세스 표준입니다. ODBC 는 이 레이블 디자인 소프트웨어와 같은 응용 프로그램을 서로 다른 여러 데이 터베이스에 연결하는 간편한 방법입니다.

이 소프트웨어에는 최신 데이터베이스에 액세스할 수 있는여러 개의 ODBC 드라이버가 포함되어 있습니다. 이러한 드 라이버 목록은 다음과 같습니다.

- Microsoft Access 드라이버(*.mdb)
- Microsoft Excel 드라이버(*.xls)
- Microsoft FoxPro 드라이버(*.dbf)
- ...

OLE DB

모든 데이터베이스 표준 및 메시징 시스템에 저장된 데이터 에 액세스하기 위한 연결 표준입니다.

ODBC 데이터 소스 설치 후 데이터 가져오기

예제를 통해 데이터베이스가 소프트웨어에 연결되어 있지 않을 때 연결 과정에 대해 확인하십시오.

ODBC 데이터 소스 설치

아래 설명된 예제는 직접 생성 모드를 이용합니다. 필요하다면 환경 메뉴에서 마법사를 선택하여 마법사를 이용할 수도 있습니다.

TKTraining.mdb 데이터베이스 연결하기

1. 코드소프트에서 **도구 > ODBC 관리자**를 선택합니다.
2. **시스템 데이터 소스** 탭(DSN)을 클릭 후 추가를 클릭합니다.

> **참고:** 시스템 데이터 소스 이름(DSNs)을 이용하여 데이터 소스를 정의할 수 있습니다. 이 데이터 소스의 경우 특정 사용자가 아닌 특정 컴퓨터에 연결됩니다. 필요한 권한을 소유한 경우 어느 사용자나 시스템 DSN 에 접속할 수 있습니다.

3. Microsoft Access Driver 를 클릭 후 마침을 클릭합니다.
4. 데이터 원본 이름 입력란에 "TK Training CS Level 2"라고 입력합니다.
5. 선택을 클릭 후 InstallDir\Samples\Forms\Tutorial 에 위치한 TKTraining.mdb 파일을 선택합니다.
6. 옵션 버튼 클릭 후 읽기 전용에 체크합니다. 이 옵션을 이용 어떠한 읽기/쓰기 문제 없이 Codesoft 와 동시에 데이터베이스를 열 수 있습니다.

7. **ODBC Microsoft Excel Setup** 대화 상자에서 **확인**을 누르십시오.

데이터 가져오기

데이터베이스가 소프트웨어에 연결되어 있다면 다음으로 문서에 연결해야 합니다.

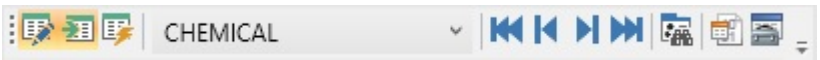
1. PRODUCT_WS3 라벨 디자인 파일을 엽니다.
2. **데이터 소스 > 데이터베이스 > 쿼리 작성/편집**을 선택합니다.
3. 데이터 소스 목록에서 TK Training CS Level 2 를 선택합니다.
4. 테이블 목록에서 "Fruits"를 선택합니다.

   데이터베이스 필드가 선택 필드 목록에 나타납니다.
5. "ProdName", "Origin", "Weight", "Reference" fields. 필드를 선택하십시오.
6.  버튼을 클릭하여 선택된 레코드를 알파벳 혹은 숫자로, 오름차순 혹은 내림차순으로 정렬할 수 있습니다.
7. **정렬 키**로 "Reference"를 선택 후 정렬 순서로 오름차순을 선택합니다.
8. 쿼리를 다음과 같이 저장합니다 -

   CInstallDir\Samples\Forms\Tutorial\PRODUCT_WS4_ODBC.CSQ.
9. 확인을 클릭합니다.

   변수가 자동으로 생성 후 데이터 소스 보기에 트리 형태로 목록이 나열됩니다.

생성한 개체가 필요로 하는 다른 값을 보거나 인쇄하려면 내비게이션 바를 이용하십시오. 쿼리 결과 창에서 인쇄해 볼 수도 있습니다.



변수 개체 생성

1. 테이터 소스 보기 창에 생성된 가변 데이터베이스 목록을 선택한 다음 화면으로 드래그합니다.
2. 메뉴에서 **문자를** 선택합니다.

Active Query Builder

고성능 쿼리 빌더는 쿼리 빌딩 과정을 시각 정보화하여 직관적인 인터페이스를 통해 사용자가 복잡한 SQL 쿼리 빌딩을 쉽게 사용할 수 있도록 해주는 도구입니다.

고성능 쿼리 빌더를 사용하려면 우선 SQL 의 개념에 대한 기본 지식이 있어야 합니다. 고성능 쿼리 빌더는 전문적이며 기술적인 세부사항은 드러내지 않은채 정확한 SQL 코드를 작성할 수 있도록 도와주며 SQL 원리에 대한 이해만으로 원하는 결과를 얻을 수 있도록 해줍니다.

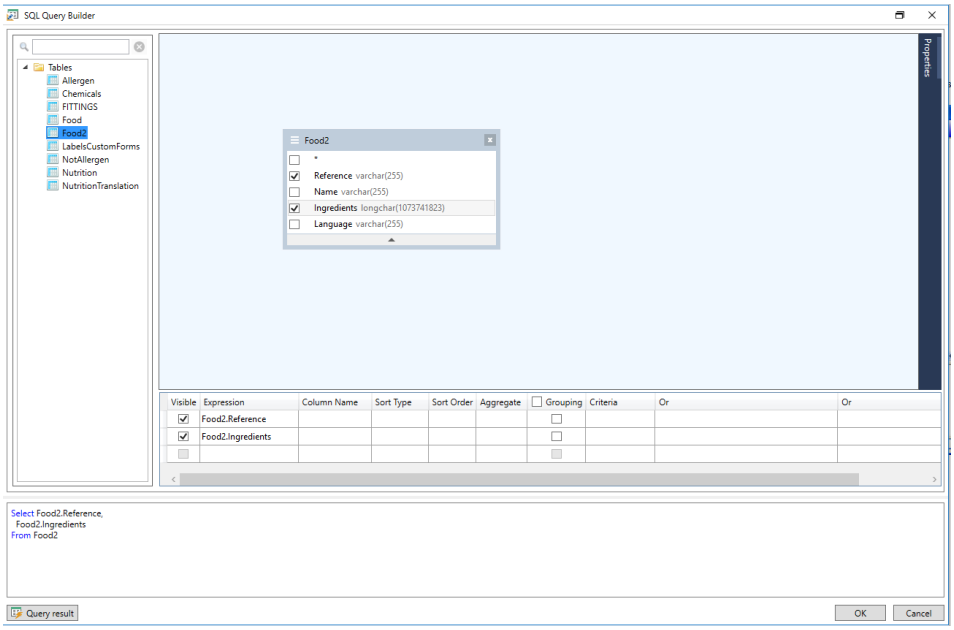
시작하기

고성능 쿼리 빌더를 시작하면 나타나는 화면입니다.

메인 창을 다음과 같은 부분으로 나눌 수 있습니다.

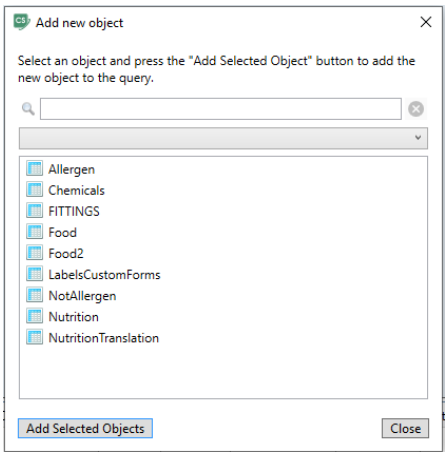
- **쿼리 작성 영역**은 쿼리를 시각적으로 표시할 주요 영역입니다. 이 영역에서 소스 데이터베이스 개체와 파생 테이블을 정의하고 이 들의 링크를 정의하며 테이블과 링크의 속성을 구성할 수 있습니다.
- **컬럼 창**은 쿼리 작성 영역 바로 아래에 있습니다. 이 창에서 쿼리 출력 컬럼과 표현식을 사용하여 모든 필요한 작업을 수행합니다. 필드 별칭 정의, 정렬 및 그룹핑, 기준 정의를 수행할 수 있습니다.
- **테이블 보기 영역**은 왼쪽에 있습니다. 여기에서 쿼리를 찾아보고 그 일부를 빠르게 확인할 수 있습니다. 창 상단의 **검색** 필드를 사용하여 검색을 지정합니다.
- 쿼리 작성 영역 위에 있는 페이지 컨트롤을 사용하여 메인 쿼리와 서브 쿼리를 전환할 수 있습니다.

- 쿼리 작성 영역의 구석에 "Q" 문자로 표시된 작은 영역은 유니온 서브 쿼리 처리 컨트롤입니다. 여기서 새로운 유니온 서브 쿼리를 추가하거나 이를 사용하여 필요한 모든 작업을 수행할 수 있습니다.



쿼리에 개체 추가

쿼리에 개체를 추가하려면 쿼리 작성 영역에서 오른쪽 마우스를 클릭하고 드롭 다운 메뉴에서 "개체 추가" 항목을 선택하십시오.

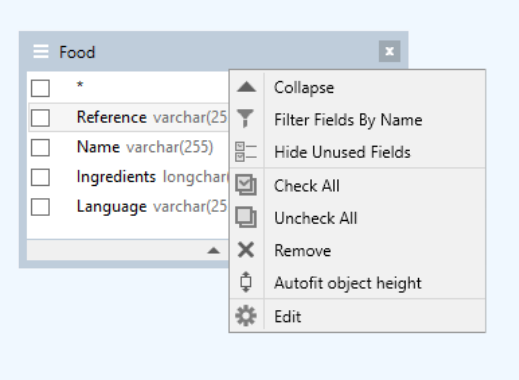


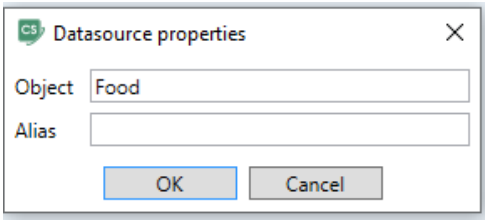
새 객체 추가창에서는 한 번에 여러 객체를 추가할 수 있습니다. 테이블 요소는 콤보 상자에 나열됩니다. **모든 객체 표시** 필터는 선택할 수 있는 모든 객체를 표시합니다. 창 상단의 **검색 필드**는 필요한 객체를 찾는 데 사용됩니다. **CTRL 키**를 누른 상태에서 하나 또는 여러 객체를 선택한 다음 **선택한 객체 추가**버튼을 눌러 이러한 객체를 쿼리에 추가할 수 있습니다. 이 작업을 여러 번 반복할 수 있습니다. 객체 추가를 마친 후 **닫기** 버튼을 클릭하여 이 창을 숨깁니다.

쿼리에서 객체를 제거하려면 객체 헤더에서 **닫기** 버튼을 클릭합니다.

개체 속성 편집

개체를 오른쪽 마우스를 클릭하고 드롭 다운 메뉴에서 **편집...** 항목을 선택하거나 단순히 개체 헤더를 마우스로 두 번 클릭하여 쿼리에 추가된 각 개체의 속성을 변경할 수 있습니다.

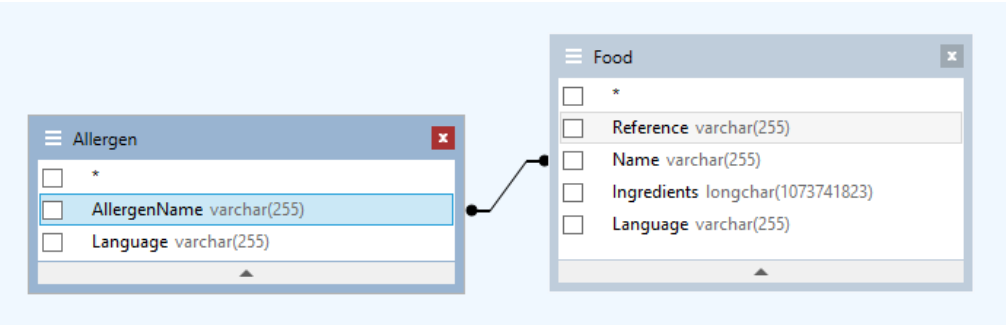




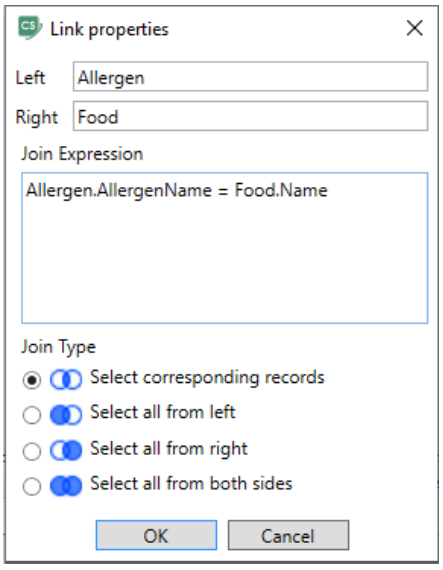
서버에 따라 **데이터 소스 속성** 대화 창은 다르게 나타날 수 있지만 모든 데이터 서버에서 최소한 별칭(Alias) 속성은 동일합니다.

테이블 조인

두 개체 사이의 링크를 만들려면 (예, 조인) 다른 개체와 링크하고자 하는 필드를 선택하고 이 필드를 다른 개체의 해당 필드에 드래그해야 합니다. 드래그를 마치면 링크된 필드를 연결하는 줄이 표시됩니다.



기본적으로 INNER JOIN 형식으로 조인됩니다. 즉, 두 테이블에서 일치하는 레코드만 결과 데이터 세트에 포함됩니다. 다른 형태의 조인을 정의하려면 링크를 오른쪽 마우스로 클릭하고 드롭 다운 메뉴에서 **편집...** 항목을 선택하거나 링크를 마우스로 두 번 클릭하여 **링크 속성** 대화 창을 열어야 합니다. 이 대화 창에서 조인 유형과 다른 링크 속성을 정의할 수 있습니다.



객체 간의 링크를 제거하려면 링크 선을 마우스 오른쪽 버튼으로 클릭하고 드롭다운 메뉴에서 **제거** 옵션을 선택하거나 링크 선을 선택하고 **삭제** 버튼을 누릅니다.

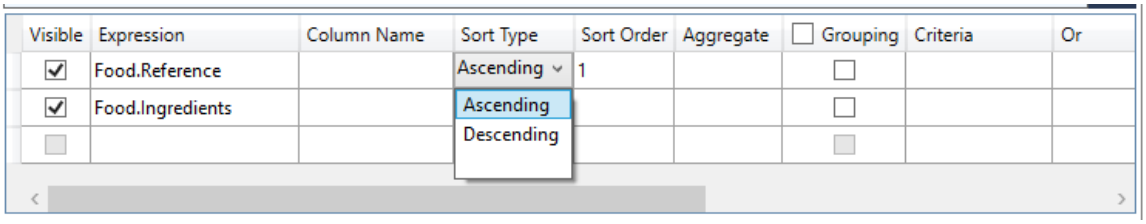
출력 필드 정렬

출력 쿼리 필드를 정렬하려면 **컬럼 창**의 **정렬 유형**과 **정렬 순서** 컬럼을 사용해야 합니다.

정렬 유형 컬럼은 필드를 정렬하는 방식을 지정할 수 있도록 합니다 - **오름** 또는 **내림** 차순.

정렬 순서 컬럼은 하나 이상의 필드를 정렬할 경우 필드가 정렬되는 순서를 설정할 수 있도록 합니다.

필드 정렬을 하지 않으려면 해당 필드의 **정렬 유형** 컬럼을 삭제해야 합니다.



조건 정의

컬럼 창에 열거된 표현식에 대한 조건을 정의하려면 **조건** 컬럼을 반드시 사용해야 합니다.

표현을 제외한 조건을 작성해야 합니다. 쿼리에서 다음과 같은 조건을 적용하려면

> **WHERE** (field >= 10) **AND** (field <= 20)
>
> 다음과 같이
>
> >= 10 **AND** <= 20 **조건** 컬럼에서 작성해야 합니다.

일반 표현식과 연산자를 편집 필드에 붙여 넣기 위해 **기준** 열의 **드롭다운 목록**을 사용할 수도 있습니다.

고급 기준을 생성하려면 **기준** 열에서 **표현식 편집기** 버튼을 사용합니다. **표현식 편집기** 대화 상자가 표시됩니다.

Or... 컬럼을 사용하여 하나의 표현식에 대해 여러 가지 조건을 지정할 수 있습니다. 이 조건은 **OR** 연산자를 사용하여 쿼리 내에서 연결됩니다.

매개 변수화된 쿼리 정의

Query Builder 는 매개 변수 값이 변수에 포함된 매개 변수화된 쿼리를 만들 수 있도록 합니다.

주: 사전에 변수를 만들어야 합니다.

1. 쿼리를 수행할 테이블을 끌어다 놓습니다.
2. 조건을 적용할 필드를 선택합니다.
3. **조건** 컬럼 또는 SQL 형식 편집 필드에서 검색 조건의 개체로 사용할 변수를 지정합니다.

   예: 이전에 작성한 변수 Var0 의 값을 검색하는 방법:

   - **SQL:**

     SELECT [Table].*

     FROM [Table]

     WHERE [Table].Field = APPLICATION.DOCUMENT.Var0

   - **조건 컬럼**

     = APPLICATION.DOCUMENT.Var0

4. **쿼리 결과** 버튼을 클릭하여 쿼리 결과를 표시합니다.

쿼리 결과 그리드

쿼리 결과 그리드에 액세스하려면 병합 데이터베이스 검색 도구 모음에서 또는 데이터 소스> 데이터베이스 메뉴를 통해 쿼리 정의 대화 상자에서  단추를 클릭합니다.

이 그리드를 사용하여 쿼리 결과를 표시하거나 특정 용어와 그 사용 예를 검색할 수 있고 또한 해당 레이블을 인쇄할 수 있습니다.

참고: "<바이너리 데이터>" 태그는 **BLOB 데이터 유형**(예: 이미지 파일)이 포함된 데이터베이스 필드에 대한 쿼리 결과 대화 상자에 표시됩니다.

쿼리 결과 그리드에는 다음이 포함됩니다.

- **검색 기능**

  검색할 쿼리를 입력할 수 있는 검색 필드

  검색 값을 입력할 수 있는 검색 데이터

   필드의 시작 부분 또는 그 어떤 부분에서도 값을 검색합니다.

- **쿼리 결과 레코드를 검색하는 네비게이션 기능**

  첫 번째 레코드 

  이전 레코드 

  다음 레코드 

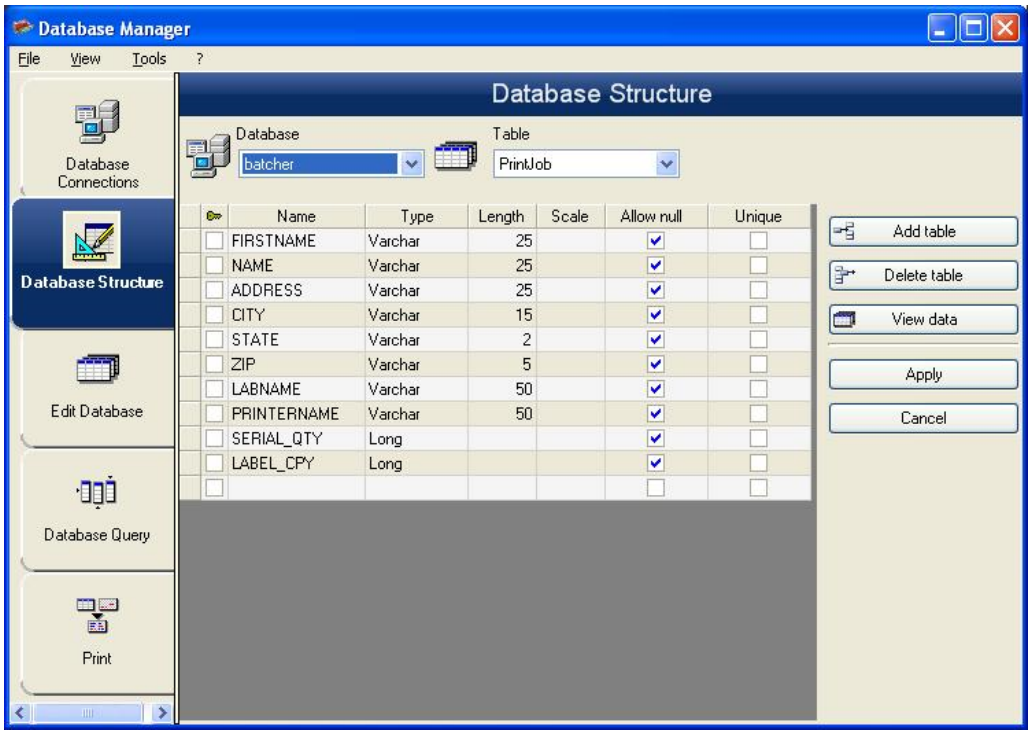
  마지막 레코드 

- **결과 그리드**

  쿼리 결과를 표시합니다.

Database Manager

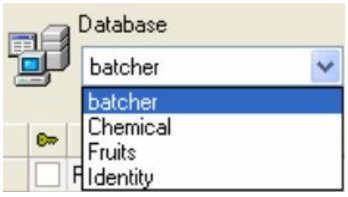
데이터베이스 관리자



데이터베이스 구조 창은 테이블/필드 등을 추가, 편집, 삭제하는 데이터베이스 파일 구조 관리에 사용됩니다.

연결 목록에서 데이터베이스 선택

1. 드롭다운 목록에서 **데이터베이스 (Database)**를 클릭합니다.
2. 필요한 데이터를 클릭합니다.

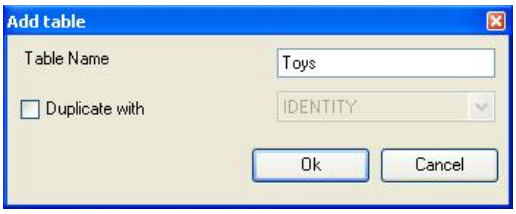


데이터베이스의 테이블 선택

1. **테이블 (Table)** 드롭다운 목록을 클릭합니다.
2. 필요한 데이터를 클릭합니다.

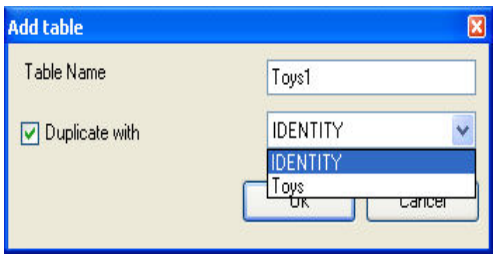
활성화된 데이터베이스에 테이블 추가

1. **테이블 추가를** (**Add Table**)클릭합니다.
2. 2. 새 테이블의 이름을 입력합니다.
3. 확인을 클릭합니다.



선택한 데이터베이스에서 미리 만들어놓은 테이블 구조를 복사할 수도 있습니다. 그렇게 하기 위해서는:

1. **복사** (Duplicate with)) 좌측의 박스를 체크합니다.
2. 드롭다운 목록을 클릭합니다.
3. 필요한 데이터를 클릭합니다.
4. **확인**을 클릭합니다.



활성화된 데이터베이스에서 테이블 삭제

1. **테이블 (Table)** 드롭다운 목록을 클릭합니다.
2. 필요한 데이터를 클릭합니다.
3. **삭제** (**Delete**)테이블을 클릭합니다.

활성화된 테이블의 데이터 보기/숨기기

1. **데이터 보기를** 클릭합니다.

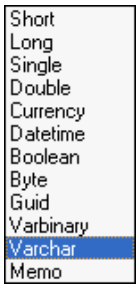
키 필드 정의

1. 필요한 필드 왼쪽의 박스를 체크합니다.

2. **적용 (Aplly)**을 클릭합니다.

내용의 필드 유형 설정

1. **Type (유형)** 컬럼에서 필요한 필드를 클릭합니다.
2. 드롭다운 목록 버튼을 클릭합니다.
3. 필요한 데이터를 클릭합니다.



4. **Apply (적용)**을 클릭합니다.

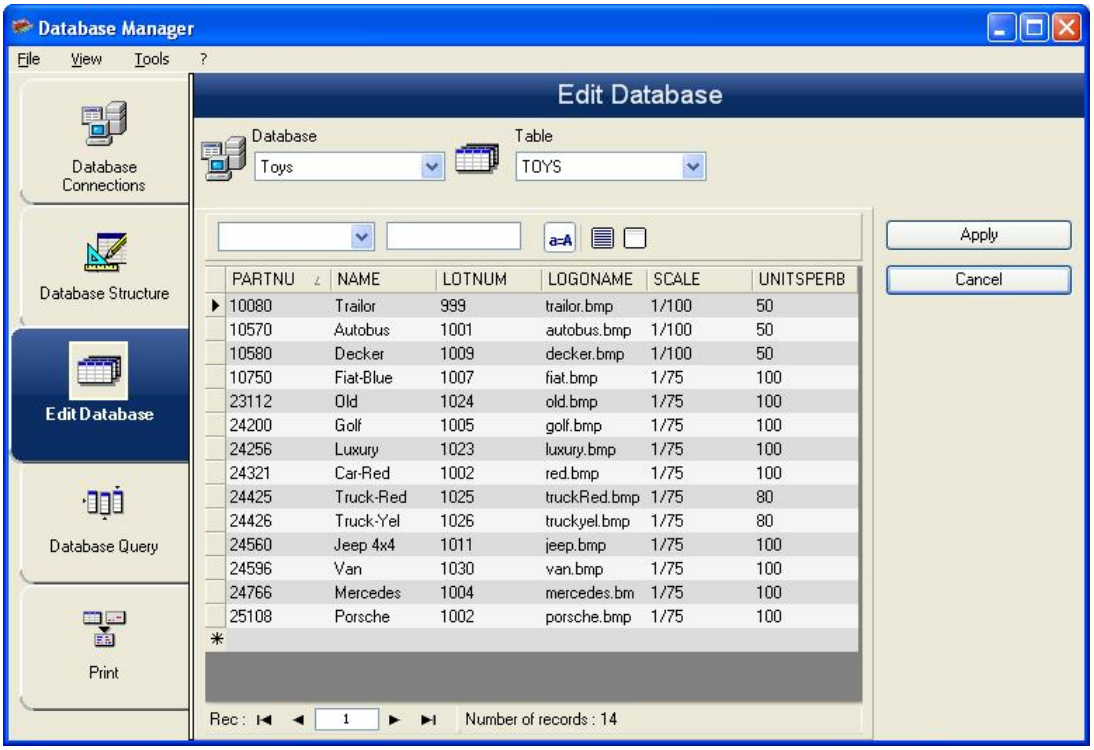
필드 최대 크기 정의

1. **Length(길이)** 컬럼에서 필요한 필드를 클릭합니다.
2. 필요한 값을 입력합니다.
3. **Apply (적용을)** 클릭합니다.

내용 없는 필드 추가

1. 필요한 필드의 **빈 상자를 (Allow Null box)** 박스 체크 합니다.
2. **Apply (적용)**을 클릭합니다.

데이터베이스 창 편집



데이터베이스 편집 창은 데이터베이스 파일의 내용을 추가, 편집 삭제하여 관리할 수 있습니다.

이는 데이터베이스 유형에 따라 다르며 예를 들어 엑셀 파일의 레코드는 수정이 불가능합니다.

내용에 따른 데이터 선택

레코드 찾기를 위해 필드 내용 이용하기

1. 드롭다운 목록 버튼을 클릭합니다.
2. 필요한 데이터를 클릭합니다.
3. 데이터 입력 필드를 클릭합니다.
4. 데이터 입력 필드에 필요한 값을 입력합니다.

동일한 모든 레코드 선택

최소한 하나의 레코드가 발견되어야 합니다.

1. 드롭다운 목록 버튼을 클릭합니다.
2. 필요한 데이터를 클릭합니다.
3. 데이터 입력 필드를 클릭합니다.
4. 데이터 입력 필드에 필요한 값을 입력합니다.
5. 모두 선택 버튼을 클릭합니다 ()

동일 레코드 선택.

적어도 하나의 레코드가 발견되어야 합니다. 검색 필드에는 다수의 동일한 내용이 있어야 합니다.

레코드 선택을 위해서는 검색 도구를 이용 하십시오. 1 (처음), 2 (이전, 3 (다음) or 4 (계속되는).

새 레코드 생성

1. *로 표시된 열의 필드를 클릭합니다.
2. 대응 필드의 필요한 값을 입력합니다.
3. **적용 (Apply)**을 클릭합니다 .

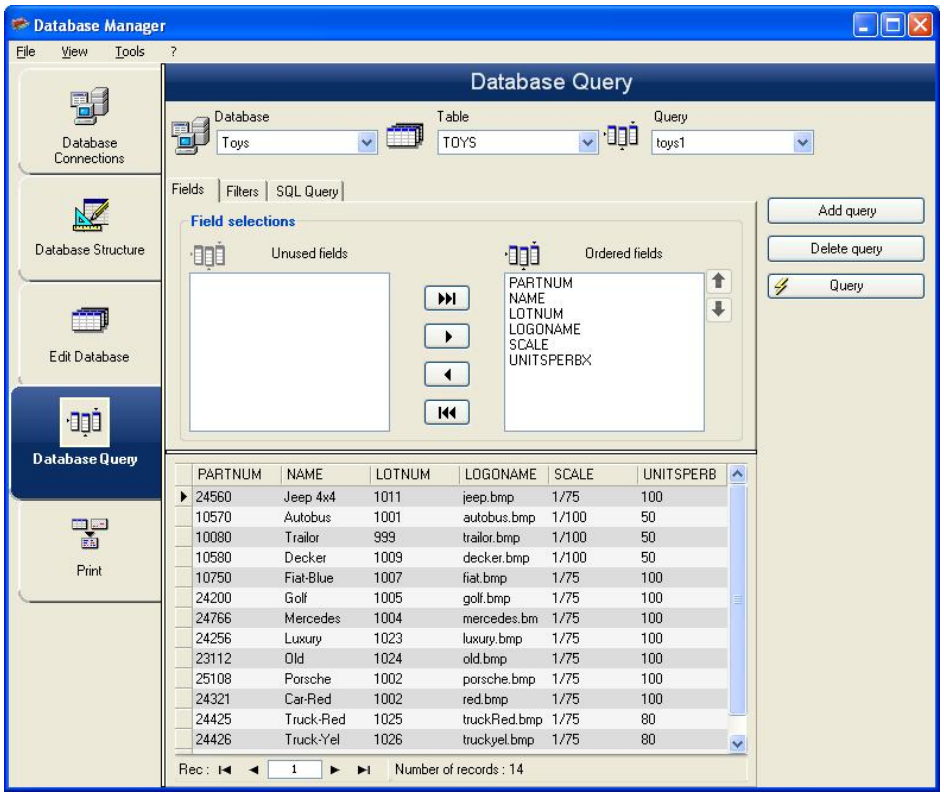
레코드 편집

1. 편집을 원하는 데이터를 클릭합니다.
2. 필요한 데이터를 입력합니다.
3. **적용 (Apply)**을 클릭합니다 .

레코드 삭제

1. 필요한 필드를 위해 데이터베이스 커서를 클릭합니다.
2. 필요한 필드에 데이터베이스 커서를 우클릭합니다.
3. 메뉴에서 **레코드 삭제를 (Delete Record)** 클릭합니다.

데이터베이스 쿼리 창



데이터베이스 쿼리 창은 다양한 필터를 생성 및 적용하는데 사용됩니다.

쿼리 추가

1. **필드 (Add query)** 탭에서 **쿼리 추가를 (Fields)** 클릭합니다 .
2. 쿼리 이름을 입력합니다.
3. **확인 (OK)** 을 클릭합니다 .

하나 혹은 그 이상의 필드 선택/선택해제

1. 내비게이션 도구를 이용합니다  .
2. 데이터베이스 미리보기를 새로고침하기 위해서는 **쿼리를 (Query)** 클릭합니다.

선택된 필드 순서 편집

1. **정렬된 필드 (Ordered fields)** 창의 필요한 필드를 클릭합니다.
2. 필요한 데이터를 찾기 위해 **위 혹**은 **아래** 화살표를 클릭합니다.
3. 데이터베이스 미리보기를 새로고침하기 위해서는 **쿼리를** (**Query)** 클릭합니다.

사전정의된 데이터를 이용하여 필터 생성

1. **필터** (**Filters)** 탭을 선택합니다.
2. **열 추가** 버튼을 클릭합니다 ().
3. **필드** (**Field)** 필드에서 필요한 데이터를 드롭 다운 목록으로부터 선택합니다.
4. 사용자 필드에서 필요한 사용자를 드롭 다운 목록으로부터 선택합니다.
5. **값** (**Value**)필드에서 필요한 값을 입력합니다.
6. 결과를 보기 위해서는 **쿼리를** (**Query**) 클릭합니다.

논리 사용자를 다수의 필터에 적용

1. 열 추가 버튼을 클릭합니다 ().
2. **논리** (**Logical)** 필드에서 필요한 데이터를 드롭 다운 목록으로부터 선택합니다. (AND 혹은 OR).
3. (위에 설명된 것처럼) 필터를 생성합니다.
4. 결과를 보기 위해서는 **쿼리를** (**Query**) 클릭합니다 .

필터 삭제

참고: 적어도 하나의 필터가 있어야 합니다.

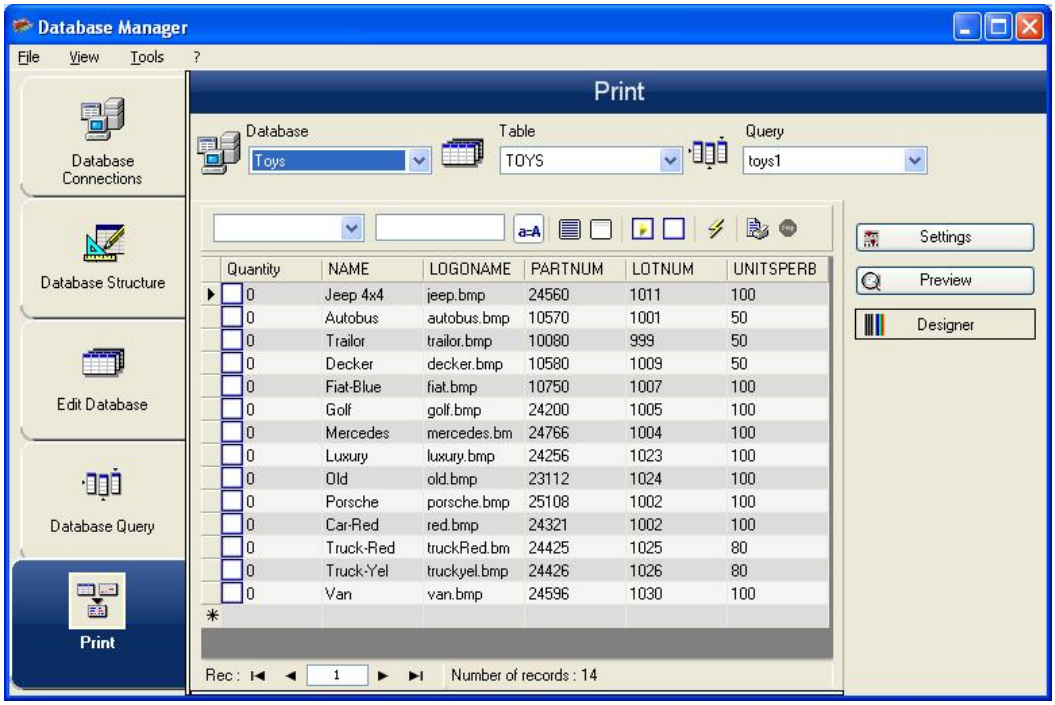
1. 필요한 필드에 데이터베이스 커서를 클릭합니다.
2. 열 삭제 버튼을 클릭합니다 ().

SQL 에서 필터 편집

참고: 적어도 하나의 필터가 있어야 합니다.

1. **SQL 쿼리** (**SQL Query**) 탭을 선택합니다.
2. SQL 쿼리를 활성화한 후 수동 변경을 가능케하려면 **SQL 언어에서 쿼리 편집 (Modify the query in SQL language)** 하기를 체크합니다.
3. 결과를 보기 위해서는 **쿼리를** (**Query**) 클릭합니다 .

인쇄 창



인쇄 창은 인쇄에 필요한 파일을 선택하고 프린터를 지정하고 인쇄 작업 전 다양한 매개 변수를 정의하는데 사용됩니다.

인쇄할 문서 선택

1. 파일에서 문서를 선택합니다.

2. 라벨 이름 그룹의 파일 박스를 체크합니다.

   - 혹은 -

1. **라벨 생성 마법사**버튼을 클릭합니다 ().
2. 마법사의 지시에 따르십시오.

참고: 데이터베이스와 연결된 라벨을 생성하는 것은 각 데이터베이스 필드 배치에 정확히 어떤 요소가 필요로 하는지 정의해야 합니다.

기존 라벨 템플릿 선택

1. 기존 문서 열기 버튼을 클릭합니다 ().
2. .lab 파일을 선택합니다.
3. **확인 (OK)** 을 클릭합니다.

참고: 옵션의 « 라벨 이름 » 및 « 프린터 이름 » 그룹 내 « 필드 » 라디오 버튼을 이용하여 필요한 라벨 혹은 프린터를 선택할 수 있으며 후자는 활성화된 데이터베이스의 필드 중 하나에 정의됩니다.

필드에서 문서 선택

데이터베이스가 필드 중 하나에 인쇄 작업에 필요한 라벨 이름을 포함하고 있다면 이 필드를 데이터베이스 관리자가 .lab 파일을 선택하는 공간으로 정의할 수 있습니다.

Ref	Designation	Qt	Code	Labname
6574	Ref1	1	9876546321	Label1
6354	Ref2	2	1236478855	Label2
6987	Ref3	3	6987456321	Label1
3684	Ref4	4	3698745632	Label3

이 예제에서는 Labname 필드가 Labname 필드로 사용됩니다.

1. 라벨 이름 그룹의 필드 박스를 체크합니다.
2. 필요한 필드를 선택합니다.

프린터 선택

추가 혹은 프린터 삭제를 클릭합니다.

공식 데이터 소스

공식 데이터 소스

명령: **Data source(데이터 소스) > Formula(수식) > Add(추가)**

수식 데이터 소스에는 생성한 데이터 소스의 목록이 있습니다. 이러한 데이터 소스는 연산자, 상수, 데이터 소스, 제어 변수, 수식 및 함수의 조합으로 채워집니다. 데이터는 숫자 또는 영숫자입니다.

문서 내에서 계산을 수행하려면 수식 데이터 소스를 먼저 생성해야 합니다.

이 데이터 소스에는 주어진 수식에 필요한 함수를 정의할 수 있는 특정 대화 상자가 있습니다.

함수 정보

함수란 인수로 불리는 특정 값을 사용하여 구문으로 불리는 일정 순서에 따라 계산을 수행하는 사전정의된 공식입니다.

함수들은 계산 또는 작동의 결과인 숫자, 문자열 또는 논리 값을 반환하는데 사용합니다.

공식 정의로 다음과 같은 6 개의 함수 그룹이 있습니다.

- 검사 문자 계산 함수
- 변환 함수
    - ATA 2000 변환 함수
- 날짜 및 시간 함수

- 논리 함수
- 수학 함수
- 문자열 함수

연산자

프로그램에 수학 연산자, 비교 연산자, 연결 연산자 및 논리 연산자가 포함됩니다.

산술 연산자

연산자	용도
*	두 숫자 곱하기
+	두 숫자 더하기
-	하나에서 다른 하나를 빼기 또는 피연산자에 마이너스 값을 할당하기
/	하나의 수를 다른 수로 나누기
^	수의 제곱 지수를 높이기
%	Modulo

비교 연산자

연산자	의미
<	~보다 작다
<=	~보다 작거나 같다
>	~보다 크다

>=	~보다 크거나 같다
=	~와 같다
<>	~와 같지 않다

연결 연산자

두 열을 조합하는 데 사용됩니다.

연산자	의미
&	두 열의 연결

논리 연산자

(논리 함수 참조)

연산자	의미
!	논리적이지 않음

검사 문자 계산 함수

addmodulo10(string): 끝에 추가된 Modulo 10 검사 문자와 함께 정수를 반환합니다.

addmodulo10_212(): modulo 10_212 검사 문자(표준 modulo 10 검사 문자의 이형)를 반환하고 현재 텍스트 열과 열 끝에 검사합계를 포함합니다.

addmodulo43(string): 끝에 추가된 Modulo 43 검사 문자와 함께 정수를 반환합니다.

bimodulo_11(string): 두 개의 modulo 11 검사 문자(코드 11)를 산출합니다.

canadacustomscd(string): 캐나다 관세 기준 검사 문자를 반환합니다.

Check103(string): 이 함수는 Mod103 을 계산합니다.

check_128(string): 코드 128 및 EAN-128 검사 문자를 반환합니다.

checkPZN(string): PZN 바코드를 위한 검사 숫자를 계산합니다. PZN 바코드는 독일 내 제약 제품에 사용되는 코드 39 에 기반합니다.

CheckSum2Mod47(string): 이 함수는 Mod47 을 계산합니다.

CheckSumMod10(string): 이 함수는 Mod10 을 계산합니다.

CheckSumMod10_Codabar(string): 이 함수는 Codabar 에 대한 특정 Mod10 체크섬을 계산합니다.

CheckSumMod10_MSI(string): 이 함수는 MSI 바코드에 대한 Mod10 체크섬을 계산합니다.

CheckSumMod11_3Suisse(string): 이 함수는 3Suisses 에 대한 Mod10 체크섬을 계산합니다.

CheckSumMod34(string): 이 함수는 Mod34 를 계산합니다.

checkupce(string): UPC-E 에 대한 검사 문자(6 개 필수 문자)를 반환합니다.

CRC16(string): Returns CRC-16 value.

Examples:

CRC16("123456789") = 47933

"0x" & hex(CRC16("123456789"), 4) = 0xBB3D

modulo_10(value): 인터레이스 2 of 5 코드(Code 2 of 5 interlaced), EAN-13, EAN-8, EAN-128 K-Mart, VPCA 의 modulo 10 검사 문자를 반환합니다.

modulo10_212(): modulo 10 검사 문자 이형을 반환합니다.

modulo10IBM(): IBM modulo 10 검사 문자(표준 modulo 10 검사 문자의 이형)를 반환합니다.

modulo10UPS(string): UPS modulo 10 검사 문자(표준 modulo 10 검사 문자의 이형)를 반환합니다.

modulo_11(string): modulo 11(코드 11) 검사 문자를 반환합니다.

modulo11IBM(): IBM modulo 11 검사 문자(표준 modulo 11 검사 문자의 이형)를 반환합니다.

modulo16(string): modulo 16 검사 문자를 반환합니다.

modulo_24(string): modulo 24(코드 PSA) 검사 문자를 반환합니다.

modulo_32(value): modulo 32(이탈리아 의약업) 검사 문자를 반환합니다.

modulo_43(string): modulo 43(코드 39) 검사 문자를 반환합니다.

modulo_47(string): 두 개의 modulo 47(코드 93) 검사 문사를 반환합니다.

Plessey(string): 두 개 검사 문자를 산출합니다(Plessey 코드).

pricecd(string): 4 UPC Random Price 검사 문자를 반환합니다.

pricecd5(string): 5 UPC Random Price 검사 문사를 반환합니다.

StringToExt39(string): 이 함수는 Extended 39 바코드에 사용할 문자열을 준비합니다.

UPSCheckDigit(string): UPS 확인 문자를 반환합니다. 메서드에 대한 입력이 문자열입니다. 이 메서드를 사용하면 사용자가 나머지 추적 번호를 그대로 둘 수 있습니다.

이는 15 자 시퀀스로 사용되며, 다음 시퀀스로 확인 번호를 계산합니다.

1 - 처음 두 개의 문자는 "1Z"여야 합니다.

2 - 다음 6 개의 문자는 UPS 계정 번호 "XXXXXX"(으)로 채웁니다.

3- 그 다음 2 개의 문자는 다음과 같은 서비스 유형을 나타냅니다.

- "01" - 다음 날 도착 항공 배송

- "02" - 2 일 후 도착 항공 배송

- "03" - 육로 배송

4- 그 다음 5 개의 문자는 송장 번호입니다. 송장 번호는 6 자리이지만 첫 번째 자리는 생략합니다. 예를 들어, 123456 송장의 경우 23456 자로 산출됩니다.

5- 그 다음 2 자리는 패키지 번호이며, 0 으로 채워집니다. 예를 들어, 패키지 1 은 "01"이고, 2 는 "02"입니다.

6- 마지막 최종 문자는 확인 번호입니다.

참고: 위에 설명된 시퀀스는 17 자이며, 이 중 확인 번호를 계산하는 데 15 자만 필요합니다. 이를 수행하려면 "1Z" 부분을 생략하고 메서드에 마지막 15 자만 사용합니다.

변환 함수

AI253 («string»): 응용 프로그램 식별자 253 에 대한 문자열을 준비하기 위한 특정 함수.

AI8003 («string»): 응용 프로그램 식별자 8003 에 대한 문자열을 준비하기 위한 특정 함수.

ASCII(string): 문자열 인수에서 첫 번째 문자의 ASCII 코드를 반환합니다.

예:

ascii("A") = 65

char(integer)

- 128 ~ 255 사이의 값을 위한 것입니다: ASCII 표 안의 정수 주장에 해당하는 문자를 제공합니다.
- 모든 음수 값 및 0~127 사이의 값 그리고 255 보다 큰 값의 경우: 정수 주장에 해당하는 유니코드 문자를 반환합니다.

예:

char (65) = "A"

참고: 이 응용 프로그램에서는 확장된 ASCII 문자를 지원합니다.

CodabarData(«data», «start», «stop»): CodabarData 에 대한 데이터를 조정합니다.

Code128CData(«string»): 코드 128 C 에 대한 데이터를 조정합니다.

Currencytoeuro(value): 해당 국가 현지 통화를 유로로 환전합니다.

DatabarData(«data», «min», «max», «hasComposite»): DatabarData 에 대한 데이터를 조정합니다.

DBCSToUnicode(«string», «codepage»): 공식은 변환할 데이터 <<string>>과 사용된 코드페이지 <<codepage>>의 두 가지 파라미터를 이용합니다. 코드페이지 파라미터는 언어명(태국어, 일본어, 중국어 GBK, 한국어, 중국어 Big5, 유럽, 동부, 키릴, 그리스어, 터키어, 히브리어, 아랍어, 발틱, 베트남어, UTF-8, ACP)이나 숫자값(인터넷에서 코드 페이지 식별자 문서 검색)으로 대표됩니다.

예:

DBCSToUnicode(unicodetoDBCS("傍傍傍傍", "UTF-8"), "UTF-8") =傍傍傍傍

이 기능은 유니코드가 아닌 파일이나 데이터 소스(예: 데이터베이스)에서 온 데이터에 필요합니다.

주: 이 기능은 UnicodeToDBCS 방식의 반전을 갖추고 있습니다.

dollar(value): 값을 나타내는 필드를 수정합니다.

예:

PRICE 필드에 값이 199 로 표시된 경우:

dollar(PRICE)에 $199.00 로 표시됩니다.

Ean128Data («string_1»,«string_2»): Adjust "string_1" data for EAN 128 barcode according to template "string_2"

Eurotocurrency(value): 유로를 해당 국가 현지 통화로 환전합니다.

참고: 표시할 십진법의 수는 십진법 표시 상자에 체크표시하여 공식 변수의 출력 탭에서 정의해야 합니다. 사용되는 환전 비율은 옵션 대화 상자의 기타 탭에서 정의됩니다.

fixed(value, num_decimals, non_sep): 는 num_decimals 와 같은 십진법 수와 함께 포맷된 문자열을 반환하며 여기에는 천 단위 분리 기호를 포함할 수도 있고 포함하지 않을 수도 있습니다.

예:

fixed(1234.5678, 3, TRUE) = "1234.568"

fixed(1234.5678, 3, FALSE) = "1,234.567"

참고: 표시할 천 단위 분리 기호는 십진법 표시 상자에 체크표시하여 공식 변수의 출력 탭에서 정의되어야 합니다.

FormatDate(«date», «dateFormat», «localeID»): «date»를 «dateFormat»에 따라 형식이 지정된 문자열로 변환합니다.

반환 값	문자열	문자열 형식의 날짜 값으로 형식에 따라 서식이 지정됩니다.
date	문자열	문자열 형식의 날짜 값입니다.
dateFormat	문자열 / 숫자	기본 날짜 값의 형식 - 숫자가 사전 정의된 형식을 인코딩합니다. (예: 1 dd/mm, 2 dd mmmm 등등) 가능한 값은 여기에서 사용할 수 있습니다. - 0 또는 음수값은 현재 시스템 날짜 형식을 사용함을 의미합니다. - 문자열 값은 형식을 직접 인코딩합니다. (예:"yyyy/mm/dd")
LocaleID	숫자	localeID 파라미터는 데이터 값을 나타내는 로케일을 지정합니다. localeID 는 옵션입니다(기본설정: 0, 시스템 로케일 이용). http://msdn.microsoft.com/en-us/goglobal/bb964664.aspx 에서 사용 가능한 값(숫자)을 볼 수 있습니다.

FormatMoney(«amount», «use separation characters(T/F)», «price characters», «force leading zaro (T/F)», «location for price character», «# of decimal places»):

이 함수는 통화 기호, 점 및 강제 선행 제로를 추가하고 정보 분리 문자를 사용하도록 허용합니다.

예제:

FormatMoney("123456.234", "T","$","F",0,2) = $123,456.23

FormatMoney("1234","F","$","T",8,2) = $1234.00

FormatMoney("0.234","F","$","F",0,2) = $.23

FormatMoney("0.234","F","$","T",0,2) = $0.23

참고: 가격 문자의 위치는 가격 문자 위치에 원인이 됩니다, 공백은 가격 문자와 합계 사이에 추가됩니다. 값이 결과 문자열 길이보다 적은 경우, 이 값은 무시됩니다.

FormatNumber(number): 이 함수는 값이 있고 0 이 항상 표시를 의미하는 경우 파운드 기호(#)가 표시만을 의미하는 숫자 필드의 포맷을 설정하도록 허용합니다.

Examples:

FormatNumber(123.45, "US$ #,###,###.00") = US$ 123.45
FormatNumber(123.45, "US$ 0,000,000.00") = US$ 0,000,123.45
FormatNumber(.45, "#,##0.00") = 0.45
FormatNumber(.45, "#,###.00") = 45
FormatNumber(7188302335, "(###) ###-####") = (718) 830-2335
FormatNumber(123.45, "00.00") = 23.45
FormatNumber(123.567, "###,##0.00") = 123.57

GetEnv(name): 환경 변수의 **«name»**(문자열) 값을 반환합니다.

예제:

GetEnv("OS") = " Windows_NT".

참고: 이 기능은 사용자 정의 변수와 함께 작동합니다.

GS1AIData(«AIName», «AIData», «isLastAI»): GS1 AI 에 대한 데이터를 조정합니다.

GS1HRData(«AIName», «AIData», «calcCheckDigit»): GS1 Human readablel 에 대한 데이터를 조정합니다.

GS1Norm(«string»): GS1 Norm 에 대한 데이터를 조정합니다.

int(value): 는 **value** 인수보다 작거나 동일한 최대 정수를 반환합니다.

예:

int(-5.863) = -6

int(5.863) = 5

LmChar («정수») : ANSI 테이블에 정수 인수에 해당하는 문자를 반환합니다. 이 테이블은 로케일(locale) 속성에 의존하지 않습니다. 값은 0 에서 255 사이입니다..

MaxiCodeData(«Mode»,«PostalCode»,«CountryCode»,«ClassOfService»,«TrackingNumber»,«UpsShipperNumber»,«JulianDate»,«ShipmentID»,«PackageNumber»,«TotalPackages»,«PackageWeight»,«AdressValidation»,«ShipToAdress»,«ShipToCity»,«ShipToState»): 입력 «string»에서 Maxicode 데이터를 만듭니다.

text(value, format): 는 형식 변수의 설명에서 설명된 형식에 해당하는 format 인수가 부과한 형식으로 쓰인 value 인수를 반환합니다.

예:

text (012345678,"###-##-####") = 012-24-5678

text (2125551212,"(###)###-####") = (212)555-1212

trunc(value): 는 값 인수의 정수 부분을 반환합니다.

예:

trunc (123.45) = 123

unicodetoDBCS(Data, Codepage): 이 공식에는 2 개 매개변수, 즉 전환할 데이터와 사용된 Codepage 가 필요합니다. Codepage 매개변수에는 다음 중 하나를 값으로 사용할 수 있습니다.

Thai

Japanese

Chinese GBK

Korean

Chinese Big5

European eastern

Greek

Turkish

Hebrew

Arabic

Baltic

Vietnamese

참고: Codepage 매개변수는 대/소문자를 구분하지 않습니다.

value(string): 는 한 문자열의 숫자 값을 산출합니다.

예:

value("123") = 123

value("0.00:48:00")-value("12:00:00") = "16:48:00"-"12:00:00" = 일련번호 0.00 (4 시간 48 분에 해당).

참고: 응용 프로그램은 필요한 경우 텍스트를 숫자로 자동 전환하기 때문에 공식에서 숫자 기능을 반드시 사용할 필요는 없습니다.

ValueEx(«string»): «string»을 숫자 값으로 변환합니다.

VoiceCode(«string», «string», «string»): VoiceCode 는 GTIN, Lot 및 PTI 의 선택적 날짜를 사용하여 계산되는 4 자리 숫자입니다.

Return value	string	VoiceCode 값을 계산합니다.
GTIN	string	14 자리 GTIN 숫자입니다. 숫자만 허용됩니다. 숫자가 아닌 값을 사용하면 오류가 반환됩니다. 데이터 길이가 14 자리보다 긴 경우에는 왼쪽 부분에 14 자리가 허용되고 데이터 길이가 14 자리보다 작은 경우에는 왼쪽에 '0'이 추가됩니다.
LotNumber	string	최대 20 개의 영숫자 기호 데이터입니다. 데이터가 20 자리보다 긴 경우 왼쪽 부분에 20 개의 기호가 허용됩니다.
Date	string	이는 선택적 매개 변수입니다. YYMMDD 형식으로 데이터를 표시하는 6 자리 데이터입니다. 길이가 올바르지 않거나, 값이 숫자가 아니거나, 날짜가 잘못된 경우에는 수식이 오류를 반환합니다.

이 계산은 다음과 같이 수행됩니다.

1. PlainText 를 계산합니다.

a. PlainText 는 Lot 코드와 날짜(있는 경우)에 의해 추가된 14 자리 GTIN 입니다.

b. 응용 프로그램 식별자 접두사 또는 괄호를 포함하지 마십시오.

c. GTIN, Lot 및 날짜 필드 사이에는 공백이 없습니다.

d. 날짜가 있는 경우 0 패킹을 사용하고 '/' 문자가 없는 상태의 YYMMDD 로 표시됩니다.

2. X16 + X15 + X2 + 1 다항식이 포함된 표준 ANSI CRC-16 해시를 사용하여 PlainText ASCII 바이트의 ANSI CRC-16 해시를 계산합니다.

   CRC16 기능을 참조하십시오.

3. 4 자리 최하위 숫자를 10 진수 형식(해시 mod 10000)으로 사용하여 해시에서 VoiceCode 를 계산합니다.

예:

GTIN = (01) 10850510002011

Lot = (10) 46587443HG234

PlainText = 10850510002011146587443HG234

CRC-16 Hash = 26359

VoiceCode = 6359

큰 숫자 = 59

작은 숫자= 63

예:

VoiceCode("10850510002011", "46587443HG234") = 6359

VoiceCode("65457886676767", "2", "100126") = 5836

ATA 2000 변환 함수

Bin2Hex(«value», «pad»)

이진 값을 16 진수 출력 값으로 변환합니다. ‘pad’ 매개 변수는 결과 왼쪽에 0 문자를 추가하여 전체 출력 길이를 정의하는 데 사용됩니다.

예:

Bin2Hex("0010100111101001101",0) returns 14F4D
Bin2Hex("0010100111101001101",0) returns 00014F4D

CRC16_CCITT(«string»)

CRC16 CCITT 체크섬을 계산합니다.

CRC 계산용 TOC 에 대한 ATA2000 규정에 사용됩니다.

예:

CRC16_CCITT ("A") returns B915
CRC16_CCITT ("AB") returns 4B74
CRC16_CCITT ("ABC") returns F508

CRC32_CCITT(«string»)

Calculates CRC32 CCITT checksum.

This is used in ATA2000 regulation for TOC for CRC calculation.

Example:

CRC32_CCITT ("A") returns D3D99E8B

CRC32_CCITT ("AB") returns 30694C07

CRC32_CCITT ("123456789") returns CBF43926

Dec2Bin(«value», «pad»)

십진수 값을 이진 출력 값으로 변환합니다. ‘pad’ 매개 변수는 결과 왼쪽에 0 문자를 추가하여 전체 출력 길이를 정의하는 데 사용됩니다.

예:

Dec2Bin("5", 0) returns 101
Dec2Bin("5", 4) returns 0101

Dec2Hex(«value», «pad»)

십진수 값을 16 진수 출력 값으로 변환합니다. ‘pad’ 매개 변수는 결과 왼쪽에 0 문자를 추가하여 전체 출력 길이를 정의하는 데 사용됩니다.

예:

Dec2Hex(510, 0) returns 1FE
Dec2Hex(510, 4) returns 01FE

String2Hex(«value», «pad»)

문자열 값을 16 진수 출력 값으로 변환합니다. ‘pad’ 매개 변수는 결과 오른쪽에 0 문자를 추가하여 출력에 대한 바이트 단위 정렬을 정의하는 데 사용됩니다.

RFTAG 에서 데이터는 단어 정렬(2 바이트)을 사용하여 16 진수로 인코딩되는 경우가 많습니다.

예:

String2Hex("A", 0) return 41(**A** 는 십진수로 AINSI 코드 **65** 이며 16 진수로 **41** 입니다.)

String2Hex("ABC", 2) return 414243
String2Hex("ABC", 4) return 41424300

RFID ATA2000 Birth 레코드 샘플은 다음과 같습니다.

String2Hex("MFR S0671*SEQ M37GXB92*PNO PQ7VZ4*PDT CONTROLLER*ICC 456789*UIC 2*DMF 20160701*UNT KG*HAZ UN0003*ESD 1*LOT 123456*CNT FR*PMLPML123456789", 2) returns

4D46522053303637312A534551204D333747584239322A504E4F20505137565A342A504454204
34F4E54524F4C4C45522A49

String6BitsEncoding(«value», «pad»)

문자열 값을 6 비트 ASCII 인코딩 출력 값으로 변환합니다.

‘pad’ 매개 변수는 결과 오른쪽에 0 문자를 추가하여 출력에 대한 바이트 단위 정렬을 정의하는 데

사용됩니다.

ATA2000 에서 데이터는 압축을 목적으로 6 비트(8 비트 대신)로 인코딩될 수 있습니다.

예:

String6BitsEncoding("11", 0) return C710

VDAString6BitsEncoding(«string»)

Converts a string value to a 6-Bit ASCII encoding output value using the German Association of the Automotive Industry (VDA) requirements.
In ATA2000 data can be encoded in 6bits (instead of 8bits) for compression purposes.
The table used for conversion is available here. For more details refer to specifications VDA5500 and VDA4994.

예:

VDAString6BitsEncoding("11") returns C71860820820

날짜 및 시간 함수

아래 날짜 변수는 시스템 날짜 및 시간이며 프로그램이 정의한 날짜 변수는 아닙니다.

Note: 30/12/1899 = your application's reference date. If you want the formula to return the current date, the solution is to calculate the number of days elapsed since the reference date.

BestBefore(date , dateFormat , offset , offsetUnit , changeMonth , outputFormat, localeID)

키보드 입력 값을 기반(현재 날짜 기반이 아님)으로 Best Before(최적의 이전) 날짜를 계산할 수 있습니다.

반환 값	문자열	문자열 형식으로 된 날짜 값으로, 기본 날짜 및 오프셋에 의해 계산됨
date	문자열	문자열 형식의 기본 날짜 값
dateFormat	문자열 / 숫자	기본 날짜 값의 형식 - 숫자가 사전 정의된 형식을 인코딩합니다. (예: 1 dd/mm, 2 dd mmmm 등등) 가능한 값은 여기에서 사용할 수 있습니다. - 0 또는 음수값은 현재 시스템 날짜 형식을 사용함을 의미합니다. - 문자열 값은 형식을 직접 인코딩합니다. (예:"yyyy/mm/dd")
offset	숫자	기본 날짜에 추가할 날짜 단위의 수
offsetUnit	문자열 / 숫자	offset 매개 변수의 의미입니다. - 1, "또는"D"는 일 수를 의미합니다. - 2, "또는 "M"은 월 수를 의미합니다. - 3, " 또는 "Y"는 햇수를 의미합니다.

changeMonth	숫자	선택 사항(기본값: True) - 0: False - 1: True 계산된 날짜가 해당 월에 존재하지 않는 경우(예: 2 월 30 일) True 는 다음 달의 첫 일(예: 3 월 1 일)을 반환합니다. False 는 현재 달의 마지막 날을 반환합니다(예: 2 월 28 일).
outputFormat	문자열 / 숫자	선택 사항(기본값: dateFormat 과 동일) 이 매개 변수는 계산된 날짜의 출력 형식을 지정합니다. 가능한 값은 dateFormat 의 경우와 동일합니다.
localeID	숫자	옵션(기본 설정: 0, 시스템 로케일 이용) 이 파라미터는 데이터 값을 나타내는 로케일을 지정합니다.

예:

BestBefore("14/06/2012" ,"dd/mm/yyyy","18","m", 1, "mmmm dd, yyyy", 1033) will return December 14, 2013.

BestBefore("14/06/2012" ,"dd/mm/yyyy","18","m", 1, "mmmm dd, yyyy", 1036) will return Décembre 14, 2013.

예: Formula with a date data source.

Create a date variable name date0 with the date of the day, for example 26/06/2012 (dd/mm/yy format).

BestBefore(date0 ,"dd/mm/yy","18","m", 1, "dd/mm/yyyy") will return 14/12/2013.

CFIA_Month(«date»):

Returns month name used by Canadian Food Inspection Agency (CFIA): JA, FE, MR, AL, MA, JN, JL, AU, SE, OC, NO, DE.

Example:

CFIA_Month ("03/01/2021") returns MR

CFIA_Month (today()) returns month name based on today's date, if today is 01/01/2021 - the return value is JA

CFIA_Month ("06/01") returns JN

Note: «date» argument format depends on the environment settings applied for the system.

DateOffset(date , offset, offsetUnit , changeMonth)

반환 값	날짜	지정된 기본 날짜에 날짜 간격을 추가합니다.
date	날짜	기본 날짜 값입니다.
offset	숫자	기본 날짜에 추가할 날짜 단위의 수
offsetUnit	문자열 / 숫자	offset 매개 변수의 의미입니다. - 1, "또는"D"는 일 수를 의미합니다. - 2, "또는 "M"은 월 수를 의미합니다. - 3, "또는"Y"는 햇수를 의미합니다
changeMonth	숫자	선택 사항(기본값: True) - 0: False - 1: True 계산된 날짜가 해당 월에 존재하지 않는 경우(예: 2 월 30 일) True 는 다음 달의 첫 일(예: 3 월 1 일)을 반환합니다. False 는 현재 달의 마지막 날을 반환합니다(예: 2 월 28 일).

예

DateOffset("26/06/2012",1,"d",1) will return 27/06/2012

-OR-

DateOffset(today(),1,"D", 0)

DateValue(formattedDate , format)

반환 값	날짜	formattedDate 매개 변수의 날짜 값을 반환합니다
formattedDate	문자열	날짜 값으로 변환할 텍스트 형식의 날짜입니다
format	문자열	선택 사항(기본값: 시스템 날짜 형식) 지정되어 있는 경우 직접적인 형식 문자열입니다(예: "dd/mm/yy").

예

DateValue(22062012,"ddmmyyyy") will return 22/06/2012.

day(date)는 date 인수의 날을 산출합니다.

FiscalDate(«fiscalStartDate», «outputFormat»)

날짜를 사용자의 회계 연도로 계산할 수 있도록 해줍니다 (yyyy.mm.dd).

예 (현재 날짜는 2009 년 11 월 24 일인 경우):

FiscalDate("2009.01.01", 1) = 09

FiscalDate("2009.01.01", "yyyy") = 2009

FiscalDate("2009.01.01", 3) = 47

FiscalDate("2009.01.01", "day") = 328

hour(date)는 date 인수의 시간을 산출합니다.

minute(date)는 date 인수의 분을 산출합니다.

month(date)는 date 인수의 달을 산출합니다.

now()는 현재 날짜와 시간을 산출합니다.

second (): 날짜 인수의 초를 조정합니다.

shiftcode(items)

Shift Code 는 현재 시간을 기반으로, 변경해야 할 레이블에 있는 필드의 데이터 소스로서 사용됩니다.

시프트는 정의되고 이름이 지정된 시간의 분량을 나타냅니다.

반환 값	문자열	현재 Shift Code 값을 반환합니다.
항목	문자열	다음 형식의 Shift Code 항목: 시작시간:시작분-중지시간:중지분-값|…| 시작시간:시작분-중지시간:중지분-값

예를 들어 시프트는 다음과 같이 정의할 수 있습니다.

- 7:00 ~ 15:00 = 낮
- 15:00 ~ 23:00 = 저녁
- 23:00 ~ 7:00 = 밤

참고:

1. 시간 값 정의에는 24 시간 형식이 사용됩니다. 0:00 은 자정이며 12:00 은 정오입니다.

2. 입력 내용에서 시간이 겹치는 경우, 먼저 허용 가능한 값이 반환됩니다.

3. 현재 시간에 대한 시프트가 없으면 빈 문자열이 반환됩니다.

예(현재 시간이 16:00 인 경우):

shiftcode("7:00~15:00-낮|15:00~23:00-저녁|23:00~7:00-밤") ="저녁"

SpecificDateFormat("[date format]", "+/[offset data][date interval]")

SpecificDateFormat 기능을 사용하면 특정 간격으로 날짜를 오프셋팅하여 날짜 스탬프를 맞춤화하고 이 맞춤화한 날짜를 공식 표현으로 사용할 수 있습니다. SpecificDateFormat 기능은 기본 날짜 형식을 사용하거나 선택한 다른 형식을 사용하는 시스템 시계에 기반해 현재 날짜를 오프셋팅합니다.

- [date format] [날짜 형식] 사용할 날짜 형식을 지정합니다. 이 파라미터는 괄호 안에 있어야 합니다. 유효한 날짜 형식 표를 보십시오:

- +/[offset data] 이 값은 날짜를 오프셋할 값을 지정합니다. 간격은 현재 날짜에서 더하거나 빼기를 원함에 따라 더하기(+) 또는 빼기(-)를 앞에 붙여야 합니다.
- [date interval] 날짜 간격에는 일은 d, 주는 w, 달은 m, 년도는 y 를 사용합니다.

다음 표는 다른 형식으로 표현에 사용되는 SpecificDateFormat 기능의 예를 보여줍니다. (참고: 아래의 예에서는 현재 날짜가 2013 년 7 월 29 일입니다)

SpecificDateFormat("mm dd yy" , "+1y") = 7 29 14 (현재 날짜로부터 1 년 후)

SpecificDateFormat("ww/m/yyyy" , "30w) = 1/12/2012 (현재 날짜로부터 30 주 전)

SpecificDateFormat("ddddd / wwww /mmmm" , "") = 41484/5928/1363 (1900 년 이후의 일수, 주, 개월 수)

TimeOffset(FormatDate(Now(), "mm/dd/yyyy hh:nn:ss"),"time interval +/offset time")

TimeOffset 기능을 사용하면 특정 간격으로 시간을 오프셋팅하여 시간 스탬프를 맞춤화하고 이 맞춤화한 시간을 공식 표현으로 사용할 수 있습니다. TimeOffset 기능은 기본 시간 형식을 사용하는 시스템 시계에 기반해 현재 시간을 오프셋팅합니다.

표현 안에서 사용하는 경우 TimeOffset 기능은 다음 파라미터를 사용해 작성되어야 합니다:

TimeOffset(FormatDate(Now(), "mm/dd/yyyy hh:nn:ss"),"시간 간격 +/ 오프셋 시간")

- "mm/dd/yyyy hh:nn:ss"의 기본 형식만 사용할 수 있습니다. 다른 형식을 입력하려면 TimeOffset 을 DateValue 기능으로 래핑한 다음 다른 시간 형식을 지정할 수 있는 FormatDate 기능 안에 래핑해야 합니다.
- “+/offset time” 이 값은 시간을 오프셋할 값을 지정합니다. 오프셋은 현재 시간에서 더하거나 빼기를 원함에 따라 더하기(+) 또는 빼기(-)를 앞에 붙여야 합니다.
- "시간 간격" - 시간 간격은 초를 의미하는 s, 분을 의미하는 m, 시간을 의미하는 h 여야 합니다.

다음 표는 다른 형식으로 표현에 사용되는 TimeOffset 기능의 예를 보여줍니다. (참고: 아래의 예에서는 현재 시간이 2013 년 7 월 30 일 오후 4:12:10 입니다)

TimeOffset(FormatDate(Now(), "mm/dd/yyyy hh:nn:ss"),"h+5") = 07/30/2013 21:12:10 (현재 시간으로부터 5 시간 후)

FormatDate(DateValue(TimeOffset(FormatDate(Now(), "mm/dd/yyyy hh:nn:ss"), "h+10"), "mm/dd/yyyy hh:nn:ss"), "hhnn") = 16-22 (현재 시간으로부터 10 분 후)

today()는 현재 날짜를 산출합니다.

Week(date)는 date 인수에서 그 주의 수를 반환합니다.

weekday(date)은 date 인수에서 그 주일의 요일을 반환합니다.

참고: 일요일은 한 주의 첫째 날로 간주됩니다.:

예: "Tuesday November 20, 1999"에서 weekday(now()) = 3

WeekISO8601(Date, DateFormat)

ISO8601 을 지원하는 수식을 작성할 수 있습니다. ISO 8601 은날짜 및 시간 관련 데이터 변환을 위한 국제 표준입니다.

반환 값	날짜	지정된 날짜의 주를 반환합니다.
Date	문자열	날짜 값으로 변환할 텍스트 형식의 날짜입니다.
DateFormat	문자열	기본 날짜 값의 형식 - 숫자는 사전 정의된 형식을 인코딩합니다. (예: 1-dd/mm/yy, 2-dd/mm/yyyy 등) - 0 또는 음수값은 현재 시스템 날짜 형식을 사용함을 의미합니다. - 문자열 값은 형식을 직접 인코딩합니다. (예: "yyyy/mm/dd")

예:

WeekISO8601(" 03/01/2010 ",0) = 53

WeekISO8601(" 02/01/2011 "," dd/mm/yyyy ") = 52

WeekISO8601(" 01/01/2011 ", " dd/mm/yyyy ") = 52

year(date)는 date 인수의 연도를 산출합니다.

예:

minute(now())는 현재 시와 분을 제공합니다.

year(today())는 현재 날짜의 연도를 제공합니다.

참고:분 및 시에 대해서는 now() 인수만 허용됩니다. 기타 모든 기능에 대해서 now() 및 today()가 허용됩니다.

논리 함수

논리 함수를 사용하여 하나 이상의 조건이 수행됐는지 여부를 검사할 수 있습니다.

참고: TRUE 는 1 이고 FALSE 는 0 입니다.

and (expr_1, expr_2)

그리고 (expr_1, expr_2)은 두 인수가 참인 경우 TRUE 를 반환하고 최소 하나가 거짓인 경우 FALSE 를 반환합니다. 인수들은 논리 값으로부터 계산되어야 합니다.

예:

and(exact("string","string"),exact("string","string")) = 0

and(exact("string","string"),exact("string","string")) = 1

exact (string_1, string_2)는 두 열이 동일한 경우 TRUE 를 반환하고 그렇지 않은 경우 FALSE 를 반환합니다. 이 기능은 대/소문자를 구분합니다.

예:

exact("software","software") = 1

exact("software","software") = 0

if (expr, Val_if_true, Val_if_false)은 Expr(식)가 참인 경우 Val_if_true 값을 반환하고 Expr 가 거짓인 경우 Val_if_false 인수를 반환합니다.

예:

if(exact("string", "string"), "true", "false")) = 거짓

if(exact("string", "string"), "true", "false")) = 참

not (logical)은 logical 인수의 반대입니다.

예:

not(exact("string", "string")) = 1

not(exact("string", "string")) = 0

not(False) = 1 또는 not(O) = 1

not(True) = 0 또는 not(1) = 0

not(1+1=2) = 0

or **(expr_1, expr_2)**는 두 인수가 거짓인 경우 두 인수 중 하나가 사실이면 TRUE 를 반환하고 두 인수가 모두 거짓이면 FALSE 를 반환합니다. 인수는 논리 값으로부터 계산되어야 합니다.

예:

or(exact("string", "string"),exact("string", "string")) = 0

or(exact("string", "string"),exact("string", "string")) = 1

or(true,true) = 1 또는 or(1,1) = 1

or(true,false) = 1 또는 or(1,0) = 1

or(false,false)= 0 또는 or(0,0) = 0

수학 함수

Abs(data): 이 함수는 데이터의 절대값(양수)을 제공합니다. 숫자 다음에 문자 사용을 허용합니다.

예

Abs(-5) = 5

Abs(5) = 5

base10tobaseX(string_1,string_2)는 기수 10 의 string_2 을 기수 string_1 로 전환합니다.

예

기수 16 으로 이름 붙여진 필드에 "0123456789ABCDEF" 열이 포함된 경우

BASE10TOBASEX(Base 16, 12)는 C

BASE10TOBASEX(Base16,10)는 A

BASE10TOBASEX("012345","9")는 13 을 산출합니다.

baseXtobase10**(string_1,string_2)**는 기수 string_1 의 string_2 를 기수 10 으로 전환합니다.

예

기수 16 에 열 "0123456789ABCDEF"이 포함된 경우

BASEXTOBASE10(Base16, "E")는 14

BASEXTOBASE10(Base16,10)는 A

BASEXTOBASE10("012345","9")은 13 을 산출합니다.

Ceil(data):이 함수는 데이터를 다음 정수로 반올림합니다. 숫자 다음에 문자 사용을 허용합니다.

예

Ceil(3.234) = 4

Ceil(7.328) = 8

Decimals(data1, data2): 이 함수는 data1 의 data2 소수점 이하의 자릿수를 사용합니다. 숫자 다음에 문자 사용을 허용합니다.

예

Decimals(4, 2) = 4.00

Decimals(3.524, 1) = 3.5

eval_add(«string»,«string»): 매개 변수의 덧셈을 반환합니다.

예

eval_add(5,5)=10

eval_div(«string»,«string»):매개 변수의 나눗셈을 반환합니다.

예

eval_div(20,2)=10

eval_mult(«string»,«string»): 매개 변수의 곱셈을 반환합니다.

예

eval_mult(5,2)=10

eval_sub(«string»,«string»): 매개 변수의 뺄셈을 반환합니다.

예

eval_sub(20,10)=10

Floor(data)**:** 이 함수는 데이터를 다음 정수로 반올림합니다. 숫자 다음에 문자 사용을 허용합니다.

예

Floor(3.234)= 3

Floor(7.328)= 7

hex(val_1,val_2)**는** val_1 십진 수를 총계 값 val_2 16 진법 형식으로 전환합니다.

예

hex(2, 8) = 00000002

int(value)**는** value 인수보다 작거나 동일한 최대 정수를 반환합니다.

예:

int (-5.863) = -6

int (5.863) = 5

max(data1, data2):이 함수는 가장 낮은 값을 표시합니다. 숫자 다음에 문자 사용을 허용합니다.

예:

Max(5, 12) = 12

min(data1, data2): 이 함수는 데이터의 절대값(양수)을 제공합니다. 숫자 다음에 문자 사용을 허용합니다.

예:

Min(5, 12) = 5

mod(val_1, val_2)는 val_1 인수를 val_2 인수로 나눴을 때 그 나머지를 반환합니다. 그 결과는 제수와 동일한 기호를 갖습니다.

예:

mod (7,2) = 1

mod (-7,2) = 1

mod (7,-2) = -1

mod (-7,-2) = -1

quotient(val_1, val_2)는 val_1 인수를 val_2 인수로 나눴을 때 그 정수 결과를 반환합니다.

예

quotient(10, 2) = 5

round(val_1, val_2)는 val_2 에 의해 지시된 숫자로 반올림된 수 val_1 을 반환합니다.

- val_2 가 0 보다 큰 경우 val_1 은 지시된 십진법 수로 반올림됩니다.
- val_2 가 0 인 경우 val_1 은 가장 가까운 정수로 반올림됩니다.
- val_2 가 0 보다 작은 경우 val_1 은 소수점의 왼쪽으로 반올림됩니다.

예:

round (4.25,1) = 4.3

round (1.449, 1) = 1.4

round (42.6,-1) = 40

텍스트 함수

문자열은 각 상자마다 하나의 문자가 포함되는 경우 하나의 표처럼 만들 수 있습니다. 길이(공백을 포함한 열에서 총 문자 수)에 따라 정의됩니다. 해당 열에서 문자의 위치는 표에서의 그 위치에 해당하는 것으로 즉 첫 번째 문자는 위치 1 에 있습니다.

예: 위치 3 은 열의 세 번째 문자에 해당합니다.

cyclebasex ( ) 으로 카운팅이 데이터베이스 카운팅 시스템에서 일어나도록 할 수 있습니다. 넘버링 시스템은 링크된 식 내에서 정의되어야 합니다. 또한 각 번호에 대해 시작 값, 각 증분 값 및 인쇄 매수를 지정해야 합니다. 이 모든 값은 레이블의 다른 필드로 링크될 수 있지만 필드명은 인용 부호로 표시하지 말아야 합니다.

예:

기수 16 필드에 문자열 0123456789ABCDEF 이 포함된 경우:

cyclebasex(base16, "8", 1 ,1) = 8,9,A,B,C;

cyclebasex(base16, "F", -1,1) = F,E,D,C,B,A 9,8,7;

cyclebasex(base16, "B0 ", 1,1) = B0, B1, B2;

cyclebasex("012345", "4",1,2) = 4,4,5,5,10,10,11,11;

cyclechar () 는 완전 주기를 위해 사용자가 정의한 문자 세트를 만듭니다.

예:

cyclechar("A", "C") = A B C A B C A B C ;

cyclechar("A", "C", 1,2) = A A B B C C A A B B ;

cyclenumber( ) 를 사용하여 전형적인 일련의 숫자 또는 문자(0,1,2; 또는 A,B,C;) 대신에 자신만의 일련 번호를 설정할 수 있습니다.

예:

cyclenumber(1,3)의 레이블 출력 순서: 1 2 3 1 2 3 1 2 3...

cyclenumber(1,3,1,2)의 레이블 출력 순서: 1 1 2 2 3 3 1 1 2 2 3 3...

cyclestring **( )** 으로 완전 주기를 하나의 증분 필드로 사용하는 단어 또는 문자 그룹을 만들 수 있습니다. 전체 열은 인용 부호(" ") 안에 표시해야 하며 각 단어 또는 문자 그룹은 세미콜론(;)으로 그 나머지로부터 분리시켜야 합니다.

예:

cyclestring("Mon ; Tue ; Wed ; Thu ; Fri ; Sat ; Sun") = Mon Tue Wed Thu Fri Sat Sun

다음 예는 O 와 I 를 제외한 알파벳으로 된 모든 글자들을 사용하는 레이블의 경우입니다.

cyclestring("A;B;C;D;E;F;G;H;J;K;L;M;N;P;Q;R;S;T;U;V;W;X;Y;Z")

exact **(string_1, string_2)** 는 두 열이 동일한 경우 TRUE 를 반환하고 그렇지 않은 경우 FALSE 를 반환합니다.

예:

exact("software","software") = 1

exact("software","software") = 0

Extract **(string, sep, pos)** 함수는 구분자 <<sep>>에 의해 구분되는 데이터를 포함하고 있으며, 지정된 위치 <<pos>>에 있는 <<string>> 문자열에서 하부 문자열을 반환합니다.

예:

Extract("1;2;3;4", ";", 3) = 3

find (string, key, start) 는 string 인수에서 key 인수가 처음 사용된 위치를 반환합니다. string 인수의 검색은 start 인수(start >= 1)에 의해 반환된 위치에서 시작합니다. key 인수가 전혀 발견되지 않는 경우 그 함수는 0 으로 재설정됩니다. 이 함수는 대문자와 소문자를 구별합니다.

예:

find("Peter McPeepert","P",1) = 1

find("Peter McPeepert","p",1) = 12

left (string, num_char) 는 해당 string 인수에서 뽑은 문자열을 반환합니다. 이 열은 string 인수의 위치 1 에서 시작하고 그 길이는 num_char 인수과 동일합니다.

예:

left("Peter McPeepert",1) = P

left("Peter McPeepert ",5) = Peter

len (string) 은 string 인수의 길이를 산출합니다. 스페이스는 문자로 계산됩니다.

예:

len("Paris, New York") = 15

len("") = 0

len(" ") = 1

lower (string) 는 텍스트열에서 모든 대문자를 소문자로 전환합니다.

예:

lower("Paris, New York") = paris, new york

LTrim (string) 이 함수는 왼쪽 데이터의 숫자 앞이나 뒤에 있는 공백을 자동으로 제거합니다.

예:

LTrim(" No."): No

mid **(string, start, num_char)** 는 해당 string 인수에서 뽑은 문자열을 반환합니다. 이 열은 start 인수(start >=1)에 해당하는 위치에서 시작하고 그 길이는 num_char 인수와 동일합니다.

예:

mid("Paris, New York",8,8) = New York

pad **()** 는 필드의 왼쪽에 문자를 추가하여 전체 입력에 사전정의된 길이를 할당합니다. 모든 문자는 패딩 문자로 선택될 수 있습니다.

예:

GREETING 필드에 값 HELLO 가 표시된 경우:

pad(GREETING,8) = 000HELLO

pad(5,3) = 005

pad("Nine",6,a) = aaNine

replace **(string, start, num_char, new_string)** 는 전환된 해당 string 인수를 반환합니다. start 인수에서 정의된 위치에서 문자 수(num_char 인수에 해당)가 new_string 인수에 의해 대체됩니다.

예:

replace("Paris, New York",8,8,"Singapore") = Paris, Singapore

ReplaceString **(string, old_string, new_string)**: 함수는 구분자 <<sep>>에 의해 구분되는 데이터를 포함하고 있으며, 지정된 위치 <<pos>>에 있는 <<string>> 문자열에서 하부 문자열을 반환합니다.

예:

ReplaceString("abc12def12", "12", "") = abcdef

rept (string, num_char) 는 string 인수가 num_char 인수에서 그 숫자만큼 반복됩니다.

예:

rept("Ah Paris! ",2) = Ah Paris! Ah Paris!

right (string, num_char) 는 해당 string 의 마지막 문자들로 구성된 문자열을 산출하며 그 길이는 num_char 인수와 동일합니다.

예:

right("Purchase order",8) = order

RTrim (string) 이 함수는 오른쪽 데이터의 숫자 앞이나 뒤에 있는 공백을 자동으로 제거합니다.

예:

RTrim("Part ") :Part

search (string, key, start) 는 해당 string 인수에서 key 인수가 처음 사용된 위치를 산출합니다. 검색은 start 인수(start >= 1)에 의해 정의된 위치에서 시작합니다. key 인수가 사용된 예가 없는 경우 함수는 0 으로 재설정됩니다.

예:

search("Purchase order","order,1) = 8

search("Purchase order","c",1) = 8

StrAfter (data, start after, length) 이 함수는 문자 다음에 지정된 시작 이후 문자 길이와 정확하게 같은 문자열이 됩니다.

예:

StrAfter("1234-5678", '-', 3)= 대시 다음에 문자 3 개를 즉시 가져감 (567)

StrAfter("1234-5678", '-')= 대시 다음에 모든 문자를 가져감 (5678)

StrBefore **(data, start before, length)**:이 함수는 문자 앞에서 지정된 시작 전 문자 길이와 정확하게 같은 문자열이 됩니다.

예:

StrBefore("1234-5678", '-', 2)= 대시 앞에 문자 2 개를 즉시 가져감 (34)

StrBefore("1234-5678", '-')= 대시 앞에 모든 문자를 가져감 (1234)

SuppressBlankRows **(string)** 건너뛴 빈 라인과 함께 문자열을 반환합니다. 객체를 생성하고 원하는 필드를 배치하며 비어 있는 필드를 표시하지 않도록 할 수 있습니다.

예:

SuppressBlankRows ({Var0} & char(10) & {Var1} & char(10) & {Var2})(Var0, Var1 및 Var2 는 변수이며 char(10)은 새 라인을 의미하는 "\n" 기호입니다.)

Variables/Values	Formula	Result
Var0 = "Doris" Var1 = "Bull Run Ranch" Var2 = "Aurora"	*SuppressBlankRows({Var0} & char(10) & {Var1} & char(10) & {Var2})*	Doris Bull Run Ranch Aurora
Var0 = "Craig" Var1 = "" Var2 = "Chicago"	*SuppressBlankRows({Var0} & char(10) & {Var1} & char(10) & {Var2})*	Craig Chicago
Var0 = "Craig" Var1 = "" Var2 = "Chicago"	*{Var0} & char(10) & {Var1} & char(10) & {Var2}*	Craig Chicago

trim (string) 는 전환된 string 인수를 반환합니다. 열 처음과 끝에 있는 모든 스페이스는 삭제됩니다. 두 단어 사이에 포함된 스페이스 수를 하나로 줄입니다.

예:

trim(" Purchase order") = Purchase order

trimall (string) 는 전환된 string 인수를 반환합니다. 모든 스페이스는 삭제합니다.

예:

trimall("Paris / New York / Rome") = Paris/NewYork/Rome

upper (string) 는 대문자로 전환된 string 을 산출합니다.

예:

upper("Purchase order") = PURCHASE ORDER

ztrim( ) 는 전체가 숫자로 된 필드에서 왼쪽에서 시작되는 모든 0 을 삭제합니다.

예:

WEIGHT 필드에 값이 000200 으로 표시된 경우:

ztrim(weight) = 200

공식 변수의 등록 정보 정의하

명령: **Data source(데이터 소스) > Formula(수식) > Properties(속성).**

문서 검색의 데이터 소스 탭에서 변수 등록 정보 정의하기.

1. **Edit(편집)** 상자에 직접 수식을 입력합니다.

   -또는-

   원하는 요소를 선택한 후 **Insert(삽입)**를 클릭합니다.
2. **OK(확인)**를 클릭합니다.

힌트: 두 번 클릭하면 요소를 삽입할 수 있습니다

참고: 수식에서 사용할 변수에 &+-*/<>=^%,!\" 문자 중 하나를 포함하는 이름이 있는 경우에는, {} 괄호로 묶어야 합니다.

참고: 출력 페이지에 정의된 서식을 포함하여 현재 수식 계산 결과를 나타내는 동적 미리 보기입니다. 오류가 발생한 경우 미리 보기가 빨간색으로 표시됩니다. 구한 값이 잘린 경우에는 Output(출력) 탭에 지정된 최대 길이를 수정해야 합니다.

실제 연습: 특정 모듈 생성

이 연습 과정에서는 "Formula_4_NewCustCode" 함수 데이터 소스를 이용하여 "Customer_Code" EAN8 바코드를 2/5 Interleaved 바코드로 변환할 것입니다.

바코드 속성은 다음과 같습니다:

- 심볼로지 : 프린터,

- 높이: 4mm,
- 좁은 바 너비: 1mm,
- 비율: 2,
- 사람이 판독 가능: 바코드 하단 중앙 정렬,
- 바와의 거리: 0mm,
- 문자 글꼴: 프린터 글꼴

1. ORDER_WS2 라벨을 엽니다.

무게 계산

Formula_1_Weighted 수식을 생성합니다. Customer_Code 의 첫 번째 캐릭터는 1 로 곱해지고 두 번째 캐릭터는 2 로 곱해지고 세 번째는 1 로 그리고 네 번째는 다시 2 로 곱해지며 이와 같이 계속된다는 사실을 기억하십시오.

변수의 최대 출력 길이는 6 입니다

Formula_1_Weighted:

mid(Customer_Code,1,1)*1&mid(Customer_Code,2,1)*2&mid(Customer_Code,3,1)

1&mid(Customer_Code,4,1)*2

무게 계산 결과 추가

다음 단계에는 이전 수식의 결과 값을 더하는 것이 포함되어 있습니다. 최대로 허가된 캐릭터 열 길이는 2 라는 점을 기억하십시오.

두 번째 수식을 생성 후 이름을 Formula_2_Sum 이라고 지정합니다.

체크 디지트 계산:

이전 결과를 이용하여 이제 체크 디지트 값을 계산할 것입니다.

세 번째 수식 생성 후 이름을 Formula_3_CheckDigit 라고 지정합니다

표현식은 다음과 같습니다:

if ((Formula_2_Sum % 10) > 0,10 - Formula_2_Sum % 10,0)

인코딩 될 데이터 계산:

바코드 생성 시 인코딩 될 데이터가 포함되어야 합니다. 예를 들면 변수 Customer_Code 값은 체크 디지크 값(Formula_3_CheckDigit)과 연결되어야 합니다.

네 번째 수식 생성 후 이름을 Formula_4_NewCustCode 라고 지정합니다. 이 수식이 Customer_Code 와 Formula_3_CheckDigit 의 결과 값입니다.

바코드 생성:

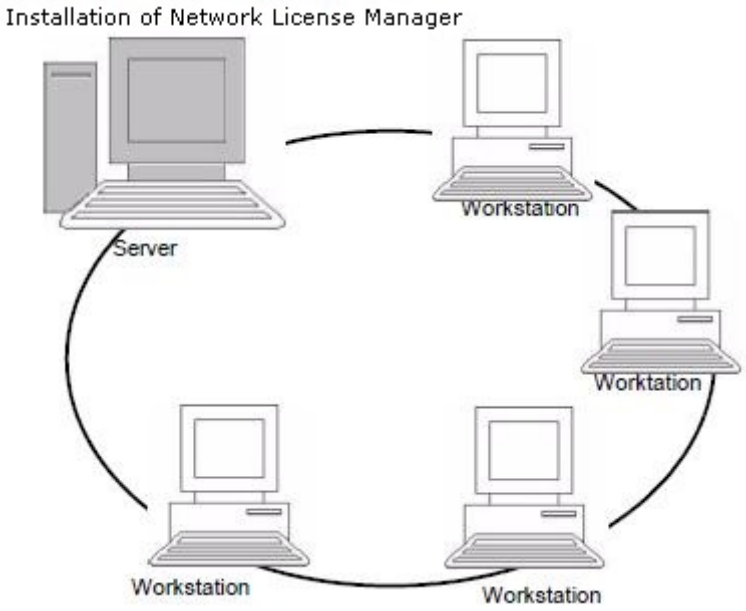
1. Formula_4_NewCustCode 함수 선택 후 Customer_Code 바코드 위로 드래그 합니다.
2. 바코드 속성을 정의합니다.

네트워크 버전 설치

기능 설명

네트워크 (여러 사용자용) 패키지를 사용하면 네트워크를 통해 레이블 제작 소프트웨어에 대한 접속 제어할 수 있습니다. 이 유틸리티를 사용하면 네트워크에서 위치에 관계 없이 레이블 제작 소프트웨어에 여러 사용자가 동시에 접속하도록 할 수 있습니다.

레이블 제작 소프트웨어의 네트워크 버전을 사용하려면 서버나 서버 역할을 할 장치에 **Network License Manager** 를 설치한 다음 각 워크스테이션에 레이블 제작 소프 트웨어를 설치해야 합니다.



네트워크 설치 절차

네트워크 구성

소프트웨어를 설치하려면 먼저 네트워크 관리자가 특히 다음과 같은 사용자 그룹의 네트워크 구조를 정의해야 합니다 .

- **Network License Manager** 와 동글을 설치할 라이센스 서버 정의
- 레이블 제작 소프트웨어를 사용할 워크스테이션 또는 클라이언트 워크스테이션 정의

네트워크 관리자 설명

Network License Manager 를 사용하면 레이블 제작 소프트 웨어의 네트워크 구성을 사용할 수 있습니다. 네트워크 관리 자에는 다음이 포함됩니다

- **Network License Manager** (라이센스 서비스) (**License Service**)
- **네트워크 설정 마법사: 네트워크 설정 마법사**는 네트워크 구성 정의를 도와줍니다.
- 사용자 관리자: 사용자 관리자는 **Network License Manager** 와 함께 설치되므로 네트워크 설정에서 레이블 제작 소프트웨어에 대한 액세스 권한을 정의할 수 있습니다.

서버에 네트워크 사용권 관리자 설치

레이블 제작 소프트웨어를 사용할 모든 워크스테이션에 설치 하기 전에 먼저 서버에 License Service 유틸리티를 설치하여 네트워크를 구성해야 합니다 .

서버에 **Network License Manager** 를 설치하려면 다음 절차 를 따르십시오 .

1. 설치용 DVD 를 DVD 드라이브에 넣습니다 .

   설치 창이 표시됩니다 .

   DVD 이 자동으로 시작되지 않으면 다음 절차를 따르십 시오 . Windows 익스플로러로 이동하여

   DVD 드라이브 문자를확장하십시오. index.hta 를 더블클릭하십시오. (예 : D:\index.hta).

2. **Network License Manager** 를 선택합니다. 여기에는 라이센스 관리자 및 사용자 관리자가

   포함됩니다. 다음 설치 단추를 클릭합니다

3. 화면에 나타나는 지시 사항을 따릅니다 .
4. 네트워크 구성의 설정을 정의하려면 서버에서 네트워크 설정 마법사를 시작합니다. 구성을

   수정하지 않은 경우기본적으로 각 워크스테이션은 자체 설정을 가집니다 .

구성

네트워크 버전을 구성하는 데 필요한 모든 도구는 Windows 작업 표시 줄 (Systray)에서 액세스 할 수있는 **Network License Manager toolbar** 에서 사용할 수 있습니다.

창 작업 표시 줄에서  아이콘을 두 번 클릭하면 **Network toolbar** 가 표시됩니다.

네트워크 설정 마법사는 네트워크 버전의 설정을 정의하는데 도움이 됩니다 .

1. 네트워크 설정 마법사를 시작하려면 아이콘을 클릭하십시오  .
2. 마법사의 단계 1 에서 설정 모드를 일반, 사용자별 또는 장치별 중에서 선택합니다 .

   - **일반** : 모든 사용자가 모든 워크스테이션에서 동일한 설정을 사용합니다 . (user.ini)

- **사용자별**: 각 사용자가 어느 워크스테이션에서나 자신의 설정에 액세스할 수 있습니다. (user name.ini)
- **장치별** : 각 워크스테이션에 고유 설정이 있습니다 .(station.ini)

3. 단계에서는 이러한 설정을 저장할 위치를 지정합니다 . 이러한 설정을 다양한 워크스테이션과 공유하려는 경우에는 모든 워크스테이션에 접속할 수 있는 네트워크 경로를 지정하십시오. (예 : TKDongle).
4. 단계에서는 공유 데이터(변수, 목록, 인쇄 로그 파일 등)를 저장할 위치를 지정합니다 . 모든 사용자가 이 폴더에 정확하게 접속할 수 있어야 합니다. .

사용자 관리자를 구성하려면

모든 레이블 제작 소프트웨어 사용자를 위한 네트워크 접속 권한은 설치 중에 정의되야 합니다 (사용자 관리자 도움말 참조). ï 네트워크 도구 모음에 있는 사용자 관리자 아이콘을 클릭합니다.

라이센스 관리자 시작

레이블 제작 소프트웨어를 모든 워크스테이션에 설치하기 전에 라이센스 관리자를 시작해야 합니다.

라이센스 관리자는 서비스로 설치되어 있습니다. 수동으로 시작할 필요는 없습니다 . 실제로 , 서비스는 워크스테이션이 켜질 때 시작되고 워크스테이션이 켜져 있는 한 백그라운드 작업으로 계속 실행됩니다 .

서비스 제어를 시작하려면

ï 네트워크 도구 모음에 있는 아이콘을 클릭합니다 .

- 또는 -

TkxWebLicenseServerController.exe 파일을 두 번 클릭합니다 .

- 또는 -

Windows 작업 표시줄에서 **네트워크 표시줄** 아이콘을 마우스 오른쪽 버튼으로 클릭하고 **라이선스 서비스 컨트롤러**를 선택합니다.

워크스테이션에 소프트웨어 설치

레이블 제작 소프트웨어는 이 소프트웨어를 사용할 모든 워 크스테이션에 설치해야 합니다 .

워크스테이션에 소프트웨어를 설치하려면

1. 설치용 DVD 를 DVD 드라이브에 넣습니다 .

   설치 창이 표시됩니다 .

   DVD 가 자동으로 시작되지 않으면 다음 절차를 따르십시오 .

   Windows 익스플로러로 이동하여 DVD 드라이브 문자를 확장하십시오. index.hta 를 더블 클릭하십시오.(예 : **D:\index.hta** 입력)

2. 설치할 제품을 선택하고 **설치** 단추를 클릭한 후 화면에 나 타나는 지시 사항을 따릅니다 .
3. 레이블 제작 소프트웨어를 시작합니다. 도구 메뉴에서 네트워크 관리를 선택합니다

   - 또는 -

   Windows 시작 메뉴에서 레이블링 소프트웨어 그룹에서 네트워크 관리 바로 가기를 선택하십시오.
4. 네트워크 라이센스 사용을 활성화합니다.
5. **Network License 유형**을 선택합니다.

- **공유 폴더 라이선스**는 Windows 의 파일 공유 기능을 사용하여 소프트웨어와 라이선스 관리자 사이를 통합합니다.

- **웹 라이선스**는 소프트웨어와 라이선스 관리자 사이에서 http/https 통신을 사용합니다.

6. **웹 라이선스** 유형을 선택했으면 **서버 포트**를 지정합니다.
7. 수정을 클릭하여 라이센스 관리자와 동글이 설치된 서버를 선택합니다.

   - 또는 -

   찾아보기를 클릭하여 라이센스 관리자가 설치된 서버를 자동으로 검색합니다.

   네트워크 설정이 이미 구성된 경우에는 현재 네트워크 구성을 사용할지 여부를 묻는 메시지가 표시됩니다 .
8. 네트워크 설정을 수정하거나 구성하려는 경우에는 **네트워크 설정 마법사** 단추를 클릭합니다
9. **확인**을 클릭합니다 .
10. 프로그램을 다시 시작합니다 .

France
+33 (0) 562 601 080

Germany
+49 (0) 2103 2526 0

Singapore
+65 6908 0960

United States
+1 (414) 837 4800

www.teklynx.com

Microsoft Partner
Gold Independent Software Vendor (ISV)