LV
TEKLYNX
LABELVIEW
TEKLYNX
LABELVIEW
Barcode Better
powered by:
TEKLYNX
Barcode Better

教程

2025版

本指南中包含的信息不具有合同性质，如有更改，恕不另行通知。

本指南中介绍的软件按许可协议销售。软件只能按照该协议的条款进行使用、复制或翻录。

未经 TEKLYNX Corporation SAS 的书面授权，不得以任何形式复制、翻录或传播本指南的任何部分，或用于购买人个人使用以外的任何用途。

© 2025 TEKLYNX Corporation SAS,

保留所有权利。

目录

关于本手册 6

印刷约定 6

关于您的产品 6

连接到数据库 7

一些提示 7

安装ODBC 数据源并导入数据 8

导入数据 8

创建变量对象 9

Active Query Builder 10

入门简介 10

添加对象至查询 11

编辑对象属性 12

加入表格 12

为输出字段排序 13

定义条件 14

定义参数化查询 14

查询结果 15

数据库管理器 17

数据库结构窗口 17

从连接列表中选择一个数据库 17

在数据库中选择一个表格 18

将表格添加到激活的数据库中 18

删除激活的数据库中的表格 19

查看/ 隐藏激活表格的数据 19

定义搜索字段 19

定义字段的内容类型 19
定义字段的最大容量 20
允许字段为空 20
编辑数据库窗口 20
根据它们的内容选择记录 21
选择所有恒等记录 21
选择一个恒等记录 21
创建新记录 22
修改记录 22
删除记录 22
数据库查询窗口 23
添加查询 23
选择/ 撤销选定一个或多个字段 23
修改所选字段的顺序 23
使用预定义数据创建过滤器 24
将一个逻辑运算符应用于多个过滤器 24
删除一个过滤器 24
修改SQL 中的过滤器 24
打印窗口 25
选择要打印的文件 25
选择一个现有的标签模版 26
从一个字段来选择文档 26
选择打印机 26
公式 27
公式数据源 27
关于函数 27
运算符 28
算术运算符 28

比较运算符 28
级联运算符 29
逻辑运算符 29
检查字符计算函数 29
转换函数 33
ATA 2000 转换函数 41
日期和时间函数 43
逻辑函数 51
数学函数 53
文本函数 57
定义公式数据源的属性 65
实践 - 计算特殊" 模数" 65
安装网络版 68
功能介绍 68
安装程序 69
网络配置 69
Network Manager 介绍 69
安装 Network and Users Utilities 69
配置 70
启动 License Service 71
在工作站上安装软件 72
在工作站上安装此软件 72

关于本手册

印刷约定

本手册通过以下印刷约定区分不同类型的信息：

- 从界面上摘录的术语，例如命令，以粗体显示。
- 按键名称以小型的大写字母显示。例如：「按 SHIFT 键」。
- 有编号的清单表示需要按照步骤执行操作。
- 当连词「-或者-」出现在一个段落后面时，表示可选择另一种方式来执行指定的任务。
- 当一个菜单命令包含子菜单时，菜单名称和紧随其后的命令名以粗体显示。因此，「转到 File（文件）> Open（打开）」表示选择 File（文件）菜单，再选择 Open（打开）命令。

关于您的产品

此手册中介绍的有些功能可能在您的产品中不可用。

有关软件中可用的特定功能的完整列表，请参见随产品提供的说明书。

连接到数据库

一些提示

本章可提供您一些概念，让您了解本软件的实际功能有多么 强大。现在我们将利用ODBC 连接（开放数据库连接）和 OLE DB （对像链接和嵌入式数据库）将您的标签（外壳） 和数据库（内容）连接起来。

数据库

数据库可以储存数据，这些数据将被编进称为关联的二维 表格。表格中的每行称为记录。记录的用途是管理对像，其属 性以字段形式被置于至一个表格中的不同列内。 数据库可以包括许多表格。如要在指定的数据库中连接不同 的表格，我们使用加入。本章接下来将以具体示例演示如何 创建加入。

ODBC

这是数据库的访问标准。ODBC 提供了将应用程序（如：您的 标签设计软件）连接于多个不同数据可的简明方法。

本软件提供了许多ODBC驱动程序，使您能访问最新数据库。驱动程序如下：

- Microsoft Access 驱动程序（*.mdb）
- Microsoft Excel 驱动程序（*.xls）
- Microsoft FoxPro 驱动程序（*.dbf）
- ...

OLE DB

它是访问信息系统中所有数据库标准和数据的链接标准。

安装ODBC 数据源并导入数据

在数据能够被访问之前，第一步必须先安装必需的数据源。

安装ODBC数据源

下面的一些步骤是采用直接创建模式. 如果您想以向导模式创建，请在 数据库 上单击鼠标右键并选择 向导来进行.

连接到TKTraining.mdb 数据库:

1. 在CODESOFT 中, 选择 工具 > 管理员ODBC.
2. 在User DSN（用户DNS）选项卡中.

注意: 您可以定义数据源为系统数据源名称 (DSNs). 这些数据源对于特定的电脑来说是唯一的，而对于特定用户非唯一性. 任何有权限的用户都可以访问此 DSN.

3. 选择Microsoft Access 驱动器（*.mdb）。单击 Finish （完成)。
4. 在Data source name（数据源名称）框中输入一"TK Training CS Level 2" 。
5. 点击 选择 并选择到数据库文件 TKTraining.mdb, 此文件位于InstallDir\Samples\Forms\Tutorial
6. 点击 选项 按钮. 勾选 只读 属性. 此选项可以避免在 CODESOFT 打开数据库文件时对数据库文件有任何的读/写操作.
7. 点击 确定 来退出 ODBC 连接对话框.

导入数据

安装了数据源，我们现在可以从数据库中导入数据并将它 插入标签中。

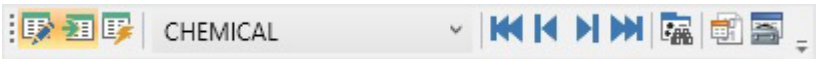
1. 打开标签文件 PRODUCT_WS3.lab

2. 选择**Data sources**（数据源）> **Database**（数据库）> **Create/Edit Query**（创建/ 编辑查询）...。
3. 从**Select a data source**（选择数据源）列表中选择 TK Training CS Level 2.
4. 选择 "Fruits" 字段在 选择表格 列表处. 此时数据库字段应该显示在 选择字段 列表处.
5. 分别选择字段 "ProdName", "Origin", "Weight" 以及 "Reference" .
6. 点击按钮 . 此按钮可以将数据库记录按照字母或数字进行升序或降序排序
7. 选择 "Reference" 作为 排序关键字 并将 "升序" 作为 排序顺序.
8. 保存查询 InstallDir\Samples\Forms\Tutorial\PRODUCT_WS4_ODBC.CSQ.
9. 单击**OK**（确认）。

变量在Document Browser（文档浏览器）的Data Sources（数据源）选项卡的Database（数据库）目录中列出。

*文件在C:\Documents and Settings\All Users\Documents\Teklynx\CODESOFT 9\Samples\Tutorial

来浏览或打印不同的数据库记录，请使用屏幕上方的导航条. 您也可以从 查看查询结果数据 窗口来打印数据.



创建变量对象

1. 从 数据库 字段列表中选择所需要创建的变量, 然后把它拖到标签设计区域松开鼠标左键.
2. 在弹出的菜单里面选择 文本.

Active Query Builder

查询生成器可以帮助您通过可视化的界面来完成复杂 SQL 查询语句的构建工作.

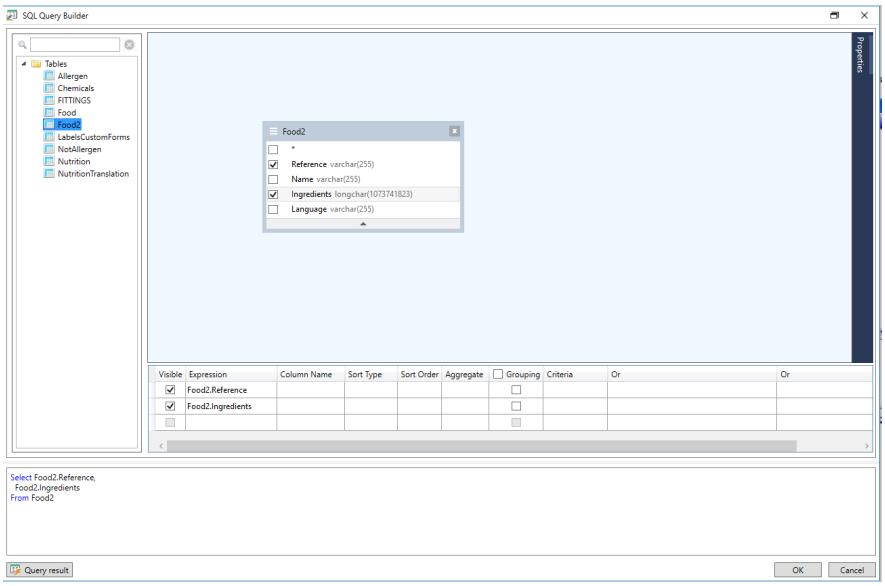
要使用查询生成器, 您必须有最基本的关于 SQL 方面的知识. 查询生成器将帮助您来创建正确的 SQL 语句, 了解最基本的SQL 语句将使您事半功倍.

入门简介

这是"主动查询生成器"启动时的外观。

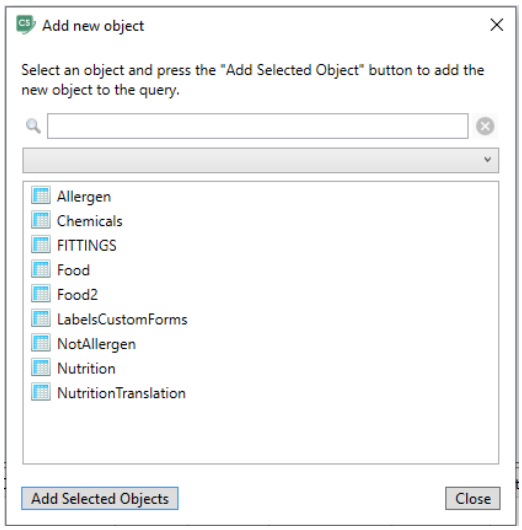
主窗口可以分成以下部分：

- **查询生成区**域是用来显示查询的可视表现形式的主要区域。通过此区域，您可以定义源数据库对象和衍生表，定义它们之间的连接并配置表格和连接的属性。
- **列窗格**位于查询生成区域下方。它通过使用查询输出列和表达式，执行所有的必需操作。在这里您可以定义字段别名、排序和分组以及定义条件。
- **表格**视图区域位于左侧。您可以在此处浏览查询，并快速找到它的任何部分。使用窗格顶部的**搜索**字段来指定搜索 。
- 查询生成区域上方的页面控制允许您在主查询和子查询之间切换。
- 查询生成区域角上的小块区域标有字母"Q"，这是并集子查询处理控件。在这里您可以添加新并集子查询并执行所有必需的相关操作。



添加对象至查询

要添加对象至查询，右键单击查询生成区域并从下拉菜单中选择"添加对象"项目。

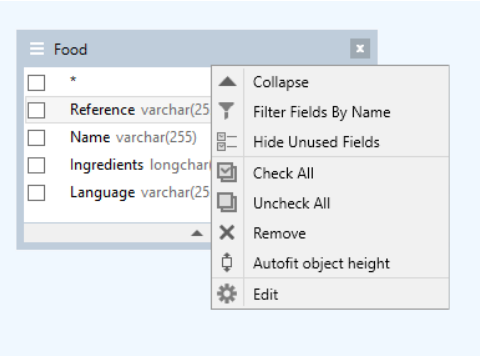


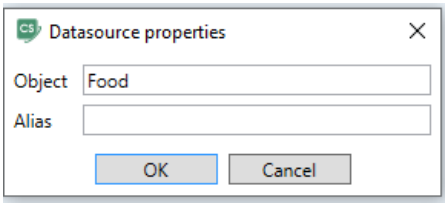
添加新对象窗口允许您一次添加多个对象。表格元素 将列在组合框中。**显示所有对象**筛选器将显示所有可供 选择的对象。位于窗口顶部的**搜索字段**用于查找所需的 对象。您可以通过按住 **CTRL 键**选择一个或多个对象，然后 按**添加选定对象**按钮，将这些对象添加到查询中。您可以重复执行 此操作多次。添加完对象后，单击**关闭**按钮 以隐藏此窗口。

要从查询中删除对象，请单击对象头中的关闭按钮。

编辑对象属性

您可以通过右键单击对象并从下拉菜单中选择编辑...项目或双击对象标题栏，就可以更改添加到查询中的每个对象的属性。

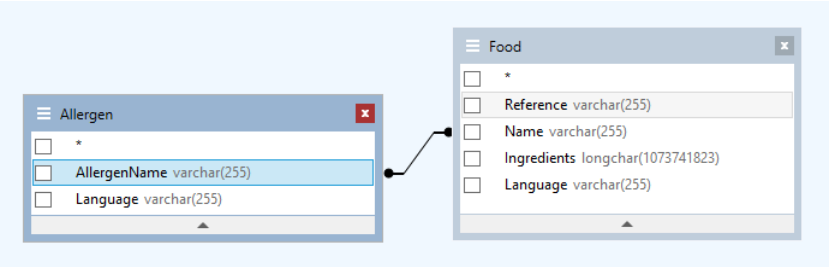




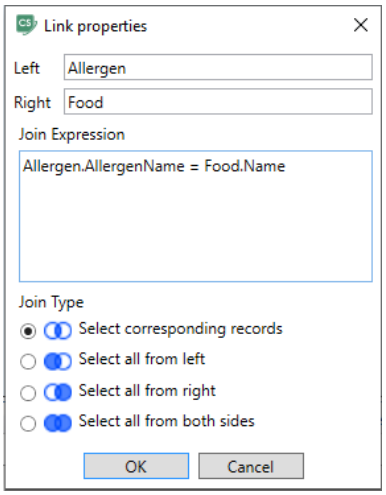
各服务器的数据源属性对话框可能有所不同，但至少所有数据库服务器的别名属性是相同的

加入表格

要在两个对象间创立连接（例如，加入它们），您应该选择想连接其中一个对象和另一个对象的字段，并将它拖到另一对象的相应字段中。完成拖放后，出现连接相互关联的字段的行。



根据默认创建的加入类型是"内部加入"，例如，只有与两个表格都匹配的记录才会包含在结果数据集中。要定义其他类型的加入，您必须右键单击连接并选择下拉菜单中的编辑...项目，或双击连接来打开连接属性对话框。此对话框可让您定义加入类型和其他连接属性



要删除对象之间的链接，请右键单击链接线，然后选择下拉菜单中的**删除**选项，或者选择链接线并按**删除按钮**。

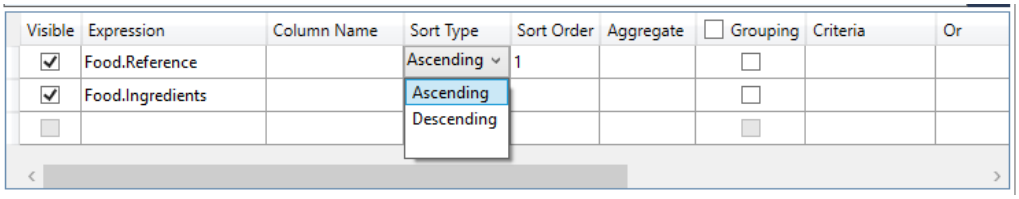
为输出字段排序

要启用输出查询字段的排序，您应该使用列窗格的排序类型和排序顺序列。

排序类型列可让您指定字段排序的方法 - 按升序或降序排列。

如果要排序多个字段，排序顺序列可让您设置要排序字段的顺序。

要禁用某字段的排序，您应该为该字段清除排序类型列。



定义条件

要定义在列窗格中列出的表达式条件，您必须使用条件列。

在该列中您应该写入条件，同时忽略表达式本身。要在您的查询中获得以下条件

> **WHERE**（字段 >= 10）**AND**（字段 <= 20）
>
> 您应该写入
>
> >= 10 **AND** <= 20 它们出现在条件列中。

您也可以使用**条件**列中的**下拉列表**，以将常用表达式和运算符粘贴到编辑字段中。对于高级条件创建，请使用**条件**列中的**表达式编辑器**按钮。**表达式编辑器**对话框即会显示 。

您可以使用Or...列来为单一的表达式指定多个条件。使用OR运算符将这些条件连接并入查询中。

定义参数化查询

查询生成器让您创建参数化查询，参数值保留在变量中。

请注意：您必须已经预先创建一个变量。

1. 拖放要执行查询的表格。
2. 选择条件适用的字段。
3. 在条件列或 SQL 格式编辑字段中，指定用作搜索条件对象的变量。

   例如：要寻找您预先创建变量 Var0 的值：

   - 在 SQL 中：

     SELECT [Table].*

     FROM [Table]

     WHERE [Table].Field = APPLICATION.DOCUMENT.Var0
   - 条件列

     = APPLICATION.DOCUMENT.Var0

4. 单击**查询结果按**钮以显示查询的结**果** 。

查询结果

如要进入查询结果，单击合并数据库浏览器工具栏的设定查询对话框中的 按钮或使用数据源 > 数据库菜单。

此网格可以显示查询结果、或搜索特殊术语及其所有事件，还可以打印相应的标签。

注意：针对包含 **BLOB 数据类型**（例如，图像文件）的数据库字段， **“<Binary data>”**标记将在“查询结果”对话框中显**示**。

查询结果包括：

- **搜索功能**

  搜索栏位，可以输入将要执行搜索的栏位

  搜索的数据，搜索要输入的值

  搜索栏位中或栏位 开始处任何位置的值。

- **浏览查询结果记录的导航功能**

  第一个记录

  上一个记录

  下一个记录

  最后一个记录

- **查询结果**

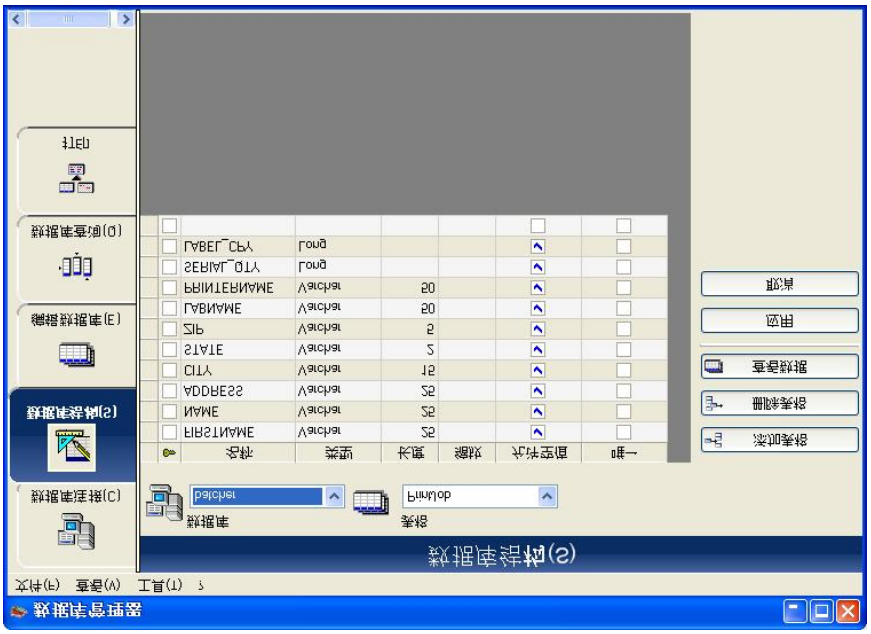
  显示产生的查询结果。

- **再次查询**

  再次查询请求，并更新网格。

数据库管理器

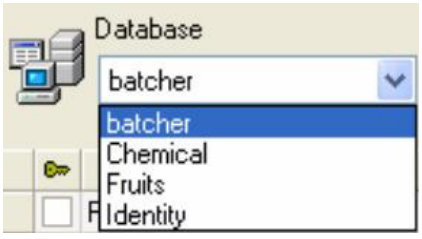
数据库结构窗口



数据库结构窗口用于管理数据库文件的结构：添加、修改或删除表格/ 字段等。

从连接列表中选择一个数据库

1. 单击Database （ 数据库 ）下拉列表.
2. 单击所需的数据.



在数据库中选择一个表格

1. 单击**Table**（表格）下拉列表.
2. 单击所需的数据.

将表格添加到激活的数据库中

1. 单击**Add table**（添加表格）.
2. 输入新表格的名称.
3. 单击**OK** (确定）.

可以从所选数据库中已经存在的表格中复制表格结构。如要进行此项操作：

1. 在**Duplicate with**（复制）旁边的框符中打勾.
2. 单击下拉列表.
3. 单击所需的数据.
4. 单击**OK**（确认）

删除激活的数据库中的表格

1. 单击Table （表格）下拉列表.
2. 单击所需的数据.
3. 单击Delete table （删除表格）.

查看/ 隐藏激活表格的数据

1. 单击View data （查看数据）.

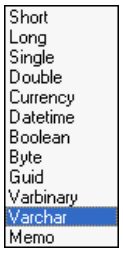
定义搜索字段

1. 在所需字段旁的框符中打勾.

2. 单击Apply （应用）.

定义字段的内容类型

1. 单击Type （类型）列中的所需字段.
2. 单击下拉列表按钮.
3. 单击所需的数据.



4. 单击Apply （应用）.

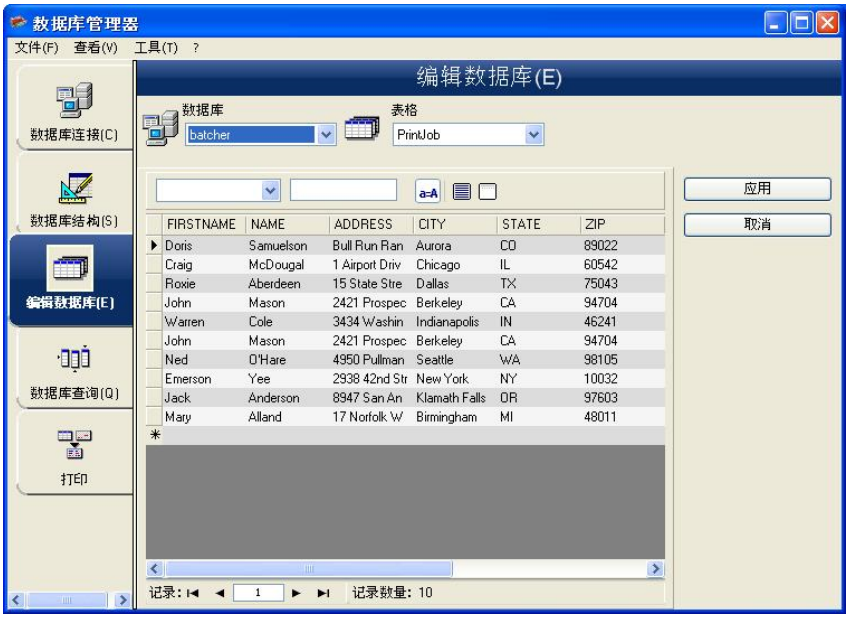
定义字段的最大容量

1. 单击**Length**（长度）列中的所需字段.
2. 输入所需值.
3. 单击**Apply**（应用）.

允许字段为空

1. 为所需字段在**Allow Null**（允许空白）框符中打勾.
2. 单击**Apply**（应用）..

编辑数据库窗口



编辑数据库窗口用于管理数据库文件的内容：添加、修改或删除数据。

这些动作取决于数据库的类型。因此不能修改Excel 文件中的记录。

根据它们的内容选择记录

使用字段目录查找记录.

1. 单击下拉列表按钮.
2. 单击所需的数据.
3. 单击数据输入字段.
4. 在数据输入字段中输入所需的值.

选择所有恒等记录

至少能找到一个数据库记录

1. 单击下拉列表按钮.
2. 单击所需的数据.
3. 单击数据输入字段.
4. 在数据输入字段中输入所需的值.
5. 单击Select all（选择所有）按钮 ( )

选择一个恒等记录

至少能找到一个数据库记录，并且在搜索字段框里有几个字段有相同的内容.

如要选择一个记录，使用搜索工具：单击1（第一个）、2（上一个）、3（下一个）或4（下列各项）。

创建新记录

1. 单击标有星号行中的字段.
2. 在相应字段中输入所需数值.
3. 单击Apply（应用）.

修改记录

1. 单击想要修改的数据.
2. 输入所需数据.
3. 单击Apply（应用）.

删除记录

1. 单击所需字段的数据库光标.
2. 右击所需字段的数据库光标.
3. 在快捷菜单中单击'Delete Record（删除记录）

数据库查询窗口

数据库查询窗口用于创建和应用各种过滤器。

添加查询

1. 单击**Add query**（添加查询）.
2. 输入查询名称.
3. 单击**OK**（确认）.

选择/ 撤销选定一个或多个字段

1. 如要选择或撤销选定一个或多个字段，单击导航工具 .
2. 单击**Query**（查询）.

修改所选字段的顺序

1. 单击**Ordered fields**（有序域）窗口中的所需字段.
2. 点击向上或向下箭头到达所需数据.
3. 单击**Query**（查询）.

使用预定义数据创建过滤器

1. 单击 **Filters**.
2. 单击**Add row** （添加行）按钮 ( )
3. 单击**Field** （字段）字段, 单击下拉列表按钮 单击所需的数据.
4. 单击**Operator** （运算符）字段 单击下拉列表按钮.
5. 单击**Value** （值）字段 输入所需值.
6. 单击**Query** （查询）.

将一个逻辑运算符应用于多个过滤器

1. 单击**Add row** （添加行）按钮 ( ).
2. 单击**Logical** （逻辑）字段 单击下拉列表按钮 单击所需的数据 (AND 或 OR).
3. 创建筛选器
4. 单击**Query** （查询）应用和查看修改

删除一个过滤器

注意: 至少必须存在一个过滤器。

1. 单击所需字段的数据库光标.
2. 单击**Remove row**（删除行）按钮 ( ).

修改SQL 中的过滤器

注意: 至少必须存在一个过滤器。

1. 选择 **SQL Query** 标签.
2. 在**Modify the query in SQL language** （修改SQL 语言查询）.

3. 单击Query （查询）.

打印窗口

打印窗口用于选择打印文件、指定打印机以及在运行打印前定义各种参数。

选择要打印的文件

1. 从文件中选一个文档
2. 检查文件名.

- 或者 -

1. 单击Create labels wizard （创建标签向导）按钮 ( ).
2. 遵守向导说明.

注意: 创建有关数据库的标签可以准确地定义每个数据库字段所需的元素。

选择一个现有的标签模版

1. 单击**Open an existing document** （打开现有文件）按钮 ( ).
2. 选择一个 .lbl 文件.
3. 单击**OK** （确认）.

注意: 当在其中一个激活数据库字段中定义了所需打印机后，使用选项Label name （标签名称）和Printer name （打印机名称）中的Field （字段）单选按钮 可以选择所需的标签或打印机。

从一个字段来选择文档

如果您的数据库中包含需要打印的标签名的字段,您可以定义这个字段作为标签名字段，让数据库管理器来通过这个字段抓取所需要打印的.lab 文件.

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

在本例里, 字段 'Labname' 就可以作为包含标签名称的字段.

1. 检查标签名称组.
2. 选择所需要的字段.

选择打印机

点击**Add or remove a printer** (添加或删除打印机）按钮.

公式

公式数据源

命令：**Data source（数据源）> Formula（公式）> Add（添加）**

Formula（公式）数据源包含已创建数据源的列表。这些数据源由运算符、常量、数据源、控制变量、公式和函数填充。数据可能是数字或字母数字。

要在文档中执行计算，您必须先创建公式数据源。

此数据源具有特定对话框，您可在该对话框中为给定公式定义所需的函数。

关于函数

注意：Windows系统的区域设置可能会影响某些公式的输出。

函数是使用叫做自变量

函数用于操作数字、字符串或逻辑值、计算或操作的结果。

公式定义中有组函数公式：

- 检查字符计算函数
- 转换函数
    - ATA 2000 转换函数
- 转换函数
- 日期和时间函数

- 逻辑函数
- 数学函数
- 字符串函数

运算符

此程式包括数学、比较、级联和逻辑运算符。

算术运算符

运算符	目的
*	将两个数字相乘
+	将两个数字相加
-	从一个数字中减去另一个数字，或为运算对像指定一个负值
/	两个数字相除
^	指数幂的一个乘方数
%	模数

注：有些公式不会像通常那样返回错误。例如：1/0, 0/0, 0.0/0.0, 1/000以及0^(-2)。这是有意为之的，以避免打印过程受阻。用户需要定义所需的行为，如果需要还可以添加额外的控制。

比较运算符

运算符	目的
<	小于
<=	小于或等于

>	大于
>=	大于或等于
=	等于
<>	差异

级联运算符

用于组合两个字符串

运算符	目的
&	两个字符串级联

逻辑运算符

（同样参见逻辑函数）

运算符	目的
!	非逻辑

检查字符计算函数

注意：Windows系统的区域设置可能会影响某些公式的输出。

addmodulo10（字符串）：返回整数，该整数末尾添加有 Modulo 10 检查字符。

addmodulo10_212(字符串)：返回 modulo 10_212 检查字符（标准 modulo 10 检查字符的编码），并包括当前文本字符串和字符串末尾的检验和。

addmodulo43（字符串）：返回整数，该整数末尾添加有 Modulo 43 检查字符。

bimodulo 11（字符串）：提供 2 个 modulo 11 检查字符（Code 11）。

canadacustomscd（字符串）：返回 Canada Customs Standard 检查字符。

check103(字符串): 此函数计算 Mod103.

check 128（字符串）：返回Code 128 和 EAN-128 检查字符。

checkPZN(字符串): 计算 PZN 条形码的校验数位PZN 条形码基于德国用于药品的 Code 39。

checkSum2Mod47(字符串): 此函数计算 Mod47.

checkSumMod10(字符串):此函数计算 Mod10.

checkSumMod10_Codabar(字符串): 此函数计算 Codabar 的 Mod10 校验和.

checkSumMod10_MSI(字符串): 此函数计算 MSI 条形码的 Mod10 校验和.

checkSumMod11_3Suisse(字符串): 此函数计算 3Suisses 的 Mod11 校验和.

checkSumMod34(字符串): 此函数计算 Mod34.

checkupce（字符串）：返回 UPC-E（6 个指令字符）的检查字符。

CRC16(字符串):返回 CRC-16 值

示例：

CRC16("123456789") = 47933

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

modulo 10（字符串）：返回 2/5 interlaced、EAN-13、EAN-8、EAN-128 K-Mart 和 VPCA 的 modulo 10 检查字符。

modulo10_212(字符串) 返回一个 modulo 10 检查字符变码

modulo10IBM(字符串)：返回 IBM modulo 10 检查字符（标准 modulo 10 检查字符的变码）

modulo10UPS(字符串)：返回 UPS modulo 10 检查字符（标准 modulo 10 检查字符的变码）

modulo 11（字符串）：返回 modulo 11（Code 11）检查字符。

modulo11IBM(字符串)：返回 IBM modulo 11 检查字符（标准 modulo 11 检查字符的变码）

modulo 16（字符串）：返回 modulo 16 查字符。

modulo 24（字符串）：返回 modulo 24（编码 PSA）检查字符。

modulo 32（值）：返回 modulo 32（意大利制药业）检查字符。

modulo 43（字符串）：返回 modulo 43（Code 39）检查字符。

modulo 47（字符串）：返回两个 modulo 47（Code 93）检查字符。

Plessey（字符串）：提供两个检查字符（Plessey 编码）。

pricecd（字符串）：返回 4 个 UPC Random Price 检查字符。

pricecd5（字符串）：返回 5 个 UPC Random Price 检查字符。

StringToExt39(字符串)：此函数用于准备在条形码中使用的 Extended 39.

UPSCheckDigit (字符串)：返回 UPS 校验字符。该方法的输入值为字符串。使用该方法时，您可确定跟踪编号的其他内容。该方法采用一个包含 15 个字符的序列，并使用这个序列计算校验数位。

1 – 前两个字符必须为“1Z”。

2 – 随后的 6 个字符使用 UPS 帐号“XXXXXX”。

3 – 随后的 2 个字符指示服务类型：

–“01”表示次日空运。

–“02”表示第二日空运。

–“03”表示地面运输。

4 – 随后的 5 个字符为发票编号（我们的发票采用 6 位编号；在此需省去第一位，例如：发票编号为 123456 时，使用 23456 这 5 个字符）。

5 – 随后的 2 个字符为包装编号，用 0 补位。例如：包装编号为 1 时使用“01”，包装编号为 2 时使用“02”。

6 – 最后一个字符为校验数位。

注意：上述序列共有 17 个字符，但计算校验数位时仅需 15 个字符。因此，应去掉“1Z”，仅使用该方法中的后 15 个字符。

转换函数

注意：Windows系统的区域设置可能会影响某些公式的输出。

AI253 (字符串)：此函数专门用于准备应用标识符 (253) 字符串.

AI8003 (字符串)：此函数专门用于准备应用标识符 (8003) 字符串.

ASCII(字符串)：返回字符串自变量第一个字符的 ASCII 编码。

示例：

ascii("A") = 65

char(值)：

- 对于介于 128 到 255 之间的值：返回与 ASCII 表中的整数自变量对应的字符。
- 对于所有负值和介于 0 到 127 以及大于 255 的值：返回与整数自变量对应的 Unicode 字符。

示例：

char (65) = "A"

CodabarData(«data», «start», «stop»)： 调整 CodabarData 的数据。

Code128CData(字符串)： 调整 Code 128 C 的数据。

Currencytoeuro(值)：将当前国家货币转换为欧洲货币。

DatabarData(«data», «min», «max», «hasComposite»)： 调整 DatabarData 的数据。

DBCSToUnicode(字符串, «codepage»)：公式需要使用两个参数 – 待使用的数据 «string»，以及所使用的代码页 «codepage»。代码页参数使用语言名称（Thai、Japanese、ChineseGBK、Korean、ChineseBig5、European、Eastern、Cyrillic、Greek、Turkish、Hebrew、Arabic、Baltic、Vietnamese、UTF-8、ACP）或数字值表示（请在 Internet 上搜索代码页标识符文档）。

示例:

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

对于来自非 Unicode 文件或数据源（如数据库）的数据，需要使用此功能。

注释：此功能对 UnicodeToDBCS 方法具有反作用。

dollar (值)：修改栏位来表示价格。

示例：

如果名为 PRICE 的栏位显示值 199，则：

dollar (PRICE) 将显示 $199.00。

Ean128Data(«字符串_1»,字符串_2»)： Adjust "字符串_1" data for EAN 128 barcode according to template "字符串_2"

Eurotocurrency(值)：将当前欧洲货币转换为国家货币。

注意：必须通过选择**显示小数**框在**公式**变量的**输出**选项卡中设定要显示的小数数量。在**选项**对话框的**其他**选项卡中设定使用的兑换率。

FileToData(«fileName», «errorData», «maxSize»)： 返回从文件读取的数据。

fixed（值,num_decimals,non_sep)：返回一串格式化的字符，该字符串的小数数量等于 *num_decimals*，有或没有千位分隔符。

示例：

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

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

注意：必须通过选择**显示小数**框在**公式**变量的**输出**选项卡中设定要显示的千位分隔符。

FormatDate(«date», «dateFormat», «localeID»)：**将 «date» 转换为采用 «dateFormat» 格式的字符串。**

返回值	字符串	字符串格式的日期值，已根据格式进行格式化
日期	字符串	字符串格式的日期值。
dateFormat	字符串/数字	基本日期值的格式。 - 数字编码预定义的格式。（例如 1- "dd/mm", 2- "dd mmmm" 等,详细请见下方列表） 可能的值可在这里找到。 - 0 或者负数表示正在使用当前系统的日期格式 - 字符串值直接编码格式。（例如 "yyyy/mm/dd"）
LocaleID	数字	参数可指定用于显示日期值的区域设置。区域设置 ID 为可选参数（默认：0，此时将使用系统区域设置）。您可在 http://msdn.microsoft.com/en-us/goglobal/bb964664.aspx 上找到可能的值（数字）

FormatMoney(«amount», «use separation characters(T/F)», «price characters», «force leading zaro (T/F)», «location for price character», «# of decimal places»)： 此函数允许添加货币符号和点、强制使用前导零，以及使用分隔符号

示例：

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

注：价格字符定位负责价格字符位置，在价格字符和数量之间添加空格。如果小于结果字符串的长度，该值将被忽略。

FormatNumber(值)：此函数允许您对数字字段进行格式化，其中英镑符号 (#) 表示仅在有值时显示，零 (0) 表示始终显示。

示例：

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»)：返回环境变量的值 «name»（字符串）。

示例：

GetEnv("OS") = " Windows_NT"。

注意：此函数可处理用户定义的变量

GS1AIData(«AIName», «AIData», «isLastAI»)：调整 GS1 AI 的数据

GS1HRData(«AIName», «AIData», «calcCheckDigit»)：调整 GS1 Human readable 的数据

GS1Norm(字符串)：调整 GS1 Norm 的数据

int(值)：返回小于或等于*值*自变量的最大整数。

示例：

int (-5.863) = -6

int (5.863) = 5

LmChar(整数)：返回对应于ANSI表中整数参数的字符。该表不依赖于本地系统。
值大小为0到255之间。

MaxiCodeData(«Mode»,«PostalCode»,«CountryCode»,«ClassOfService»,«TrackingNumber»,«UpsShipperNumber»,«JulianDate»,«ShipmentID»,«PackageNumber»,«TotalPackages»,«PackageWeight»,«AdressValidation»,«ShipToAdress»,«ShipToCity»,«ShipToState»)：从输入 **«string»** 创建 maxicode 数据。

text（值,格式）：返回一个*值*自变量，该自变量的格式是由*格式*自变量强制实施的，与表单变量说明中的格式一致。

示例：

text(012.345, "######,(###)") = 12,(345)

text(1012.345, "######,(###)") = 1012,(345)

trunc（值）：返回*值*自变量的整数部分。

示例：

trunc (123.45) = 123

unicodetoDBCS（数据,代码页）：公式采用 2 个参数 - 要转换的数据以及使用的代码页。代码页可以下列任何一种作为值：

Thai

Japanese

Chinese GBK

Korean

Chinese Big5

European eastern

Greek

Turkish

Hebrew

Arabic

Baltic

Vietnamese

注意： 代码页参数不区分大小写。

value（字符串）：提供字符串的数字值。

示例：

value ("123") = 123

value ("0.00:48:00")-value ("12:00:00") 等于 "16:48:00"-"12:00:00" 等于 0.00， 4 小时和 48 分钟的序列号。

注意：通常无需使用公式中的**值**函数，因为在必要时，应用程式会自动将文本转换为数字。

ValueEx（字符串）：将 字符串 转换为数字值。

VoiceCode(字符串, 字符串, 字符串)：VoiceCode 是使用 GTIN、批号以及 PTI 可选日期计算出来的 4 位数字。

Return value	string	计算 VoiceCode 值。
GTIN	string	14 位 GTIN 编号。只能使用数字。如果使用了非数字值，则返回错误。如果数据长度超过 14 位，则取前 14 位数字。如果数据长度不足 14 位，则在前面补 0。
LotNumber	string	数据最多由 20 位字母数字符号组成。如果数据长度超过 20 位，则取前 20 个符号。
Date	string	此项为可选参数。使用 6 位数字，以年月日的格式表示日期。 如果数据长度错误、使用非数字值或日期错误，则使用公式计算时返回错误。

该计算按下述方式执行：

1. 计算 PlainText。

   a. PlainText 为后附批号和日期（如果有）的 14 位 GTIN。

b. 请勿使用应用标识符前缀或括号。

c. GTIN、批号以及日期字段之间不留空格。

d. 以 YYMMDD 格式表示日期（如果有），且不能使用“/”字符。

2. 使用根据生成多项式 X16 + X15 + X2 + 1 得到的标准 ANSI CRC-16 哈希值来计算 PlainText ASCII 码的 ANSI CRC-16 哈希值
   请参见 CRC16 函数。
3. 取哈希值的 4 位最低有效位，将其转换成十进制形式（哈希值 mod 为 10000）作为 VoiceCode。

例如:

GTIN = (01) 10850510002011

批号 = (10) 46587443HG234

PlainText = 10850510002011146587443HG234

CRC-16 Hash = 26359

VoiceCode = 6359

大位数 = 59

小位数 = 63

示例:

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

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

ATA 2000 转换函数

注意：Windows系统的区域设置可能会影响某些公式的输出。

Bin2Hex(«value», «pad»)

将二进制值转换为十六进制输出值。“pad”参数用于通过在结果的左侧添加 0 字符来定义整个输出的长度。

示例：

Bin2Hex("0010100111101001101",0) 将返回 14F4D

Bin2Hex("0010100111101001101",0) 将返回 00014F4D

CRC16_CCITT(«string»)

计算 CRC16 CCITT 校验和。

在针对 TOC 的 ATA2000 规则中使用此函数来计算 CRC。

示例：

CRC16_CCITT ("A") 将返回 B915

CRC16_CCITT ("AB") 将返回 4B74

CRC16_CCITT ("ABC") 将返回 F508

CRC32_CCITT(«string»)

计算 CRC32 CCITT 校验和。

在针对 TOC 的 ATA2000 规则中使用此函数来计算 CRC。

示例：

CRC32_CCITT ("A") 将返回 D3D99E8B

CRC32_CCITT ("AB") 将返回 30694C07

CRC32_CCITT ("123456789") 将返回 CBF43926

Dec2Bin(«value», «pad»)

将十进制值转换为二进制输出值。“pad”参数用于通过在结果的左侧添加 0 字符来定义整个输出的长度。

示例：

Dec2Bin("5", 0) 将返回 101

Dec2Bin("5", 4) 将返回 0101

Dec2Hex(«value», «pad»)

将十进制值转换为十六进制输出值。“pad”参数用于通过在结果的左侧添加 0 字符来定义整个输出的长度。

示例：

Dec2Hex(510, 0) 将返回 1FE

Dec2Hex(510, 4) 将返回 01FE

String2Hex(«value», «pad»)

将字符串值转换为十六进制输出值。“pad”参数用于通过在结果的右侧添加 0 字符来定义输出的对齐方式（以字节为单位）。

在 RFTAG 中，数据通常以使用词对齐（2 个字节）的十六进制进行编码。

示例：

String2Hex("A", 0) 将返回 41（**A** 是十进制中的 AINSI 代码 **65**，在十六进制中是 **41**）

String2Hex("ABC", 2) 将返回 414243

String2Hex("ABC", 4) 将返回 41424300

下面是 RFID ATA2000 出生记录的样本：

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)

将返回

4D46522053303637312A534551204D333747584239322A504E4F20505137565A342A50445420434F4E54524F4C4C45522A49

String6BitsEncoding(«value», «pad»)

将字符串值转换为 6 位 ASCII 编码输出值。

“pad”参数用于通过在结果的右侧添加 0 字符来定义输出的对齐方式（以字节为单位）。

在 ATA2000 中，数据可以使用 6 位（而不是 8 位）进行编码以便压缩。

示例：

String6BitsEncoding("11", 0) 将返回 C710

VDAString6BitsEncoding(«字符串)

将字符串值转换为6位ASCII编码 输出值采用德国汽车工业协会（VDA）的要求。在ATA2000中，数据可以用6位（而非8位）编码，以便压缩。 用于转换的表格可在此处获取。 更多详情，请参阅技术规格VDA5500及VDA4994。

示例：

VDAString6BitsEncoding("11") 将返回 C71860820820

日期和时间函数

以下日期变量表示系统日期和时间，而不是程式设定的日期变量。

注意：Windows系统的区域设置可能会影响某些公式的输出。

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

您可以根据键盘输入的值（而不是当前日期）计算出有效期。

返回值	字符串	以字符串格式显示的日期值，根据基本日期和偏移量计算
date	字符串	以字符串格式显示的基本日期值
dateFormat	字符串/数字	基本日期值的格式。 - 数字编码预定义的格式。（例如 1- "dd/mm", 2- "dd mmmm" 等,详细请见下方列表） 可能的值可在这里找到。 - 0 或者负数表示正在使用当前系统的日期格式 - 字符串值直接编码格式。（例如 "yyyy/mm/dd"）
offset	数字	添加到基本日期中的日期单位数。
offsetUnit	字符串/数字	offset 参数的含义。 - 1 或 "或" 代表日期 - 2 或 "或" 代表月份 - 3 或 "或" 代表年份
changeMonth	数字	可选（默认：真） - 0 代表假 - 1 代表真 如果计算得出的日期在月份中不存在（例如 2 月 30 日） 则"真"返回在下个月的第一天（即 3 月 1 日） 而"假"返回当前月份的最后一天（即 2 月 28 日）

outputFormat	字符串/数字	可选（默认：与 dateFormat 相同） 该参数指定了计算得出的日期的输出格式。可能得出的值与 dateFormat 中的相同。
localeID	数字	可选（默认：0，此时将使用系统区域设置） 此参数可指定用于显示日期值的区域设置。

示例:

BestBefore *(«date» , «dateFormat» , «offset» , «offsetUnit» , «changeMonth» , «outputFormat»)*

BestBefore("14/06/2012" ,"dd/mm/yyyy","18","m", 1, "dd/mm/yyyy") will return 14/12/2013.

示例: Formula with a date data source.

Create a date variable name date0 with the date of the day: 26/06/2012 for this exemple with dd/mm/yy format.

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

CFIA_Month(«日期 »):

返回加拿大食品检验局（CFIA）使用的月份名称：JA、FE、MR、AL、MA、JN、JL、AU、SE、OC、NO、DE。

例如：

CFIA_Month ("03/01/2021") 返回MR

CFIA_Month (today()) 根据今天的日期返回月份名称，如果今天是01/01/2021 - 返回值为JA an>NO

CFIA_Month ("06/01") 返回JN

注：有关通过RDP运行标签软件时使用本地打印机的说明，请参阅管理员指南。

DateOffset(date , offset, offsetUnit , changeMonth)

返回值	**日期**	**为指定的基本日期添加日期间隔。**
date	日期	基本日期值。
offset	数字	添加到基本日期中的日期单位数。
offsetUnit	字符串/数字	offset 参数的含义。 - 1 或 ”或" 代表日 - 2 或 "或 " 代表月份 - 3 或 " 或 " 代表年份
changeMonth	数字	可选（默认：真） - 0 代表假 - 1 代表真 如果计算得出的日期在月份中不存在（例如 2 月 30 日） 则"真"返回下个月的第一天（即 3 月 1 日） 而"假"返回在当前月份的最后一天（即 2 月 28 日）

示例:

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

-OR-

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

示例:

Create a date variable name Date0 with the date of the day. : 26/06/2012 for this exemple with dd/mm/yy format. You should ensure that date variable is in default system format (otherwise you

can use DateValue() function to get the date value from the date variable).

DateOffset(Date0,1,"D", 0) will return 27/06/2012

DateValue(formattedDate , format)

返回值	日期	返回 formattedDate 参数的日期值
formattedDate	字符串	文本格式的日期，将转化为日期值。
format	字符串	可选（默认：系统日期格式） 如果指定，则显示直接输入的格式字符串，例如 "dd/mm/yy"

示例:

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

day（日期）提供日期自变量的日数。

FiscalDate(«fiscalStartDate», «outputFormat»)

让你计算财政年度的开始日期（yyyy.mm.dd）。

例子（如果当前的日期是2009年11月24号）：

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（日期）提供日期自变量的小时数。

minute（日期）提供日期自变量的分钟数。

month（日期）提供日期自变量的月数。

now() 提供当前日期。

second() 指定日期参数的秒数。

shiftcode(«items», «defaultValue»)

用作需要根据当前时间变化的标签上字段的数据源。移位指的是已定义和命名的一段时间。

返回值	字符串	返回当前 shift code 的值。
项目	字符串	用以下格式表示的 shift code 项目：起始小时:起始分钟-停止小时:停止分钟-值|...| 起始小时:起始分钟-停止小时:停止分钟-值

例如，可按如下方式定义移位：

- 7:00 到 15:00 = 白天
- 15:00 到 23:00 = 傍晚
- 23:00 到 7:00 = 夜晚

注意：

1. 24 小时制用于定义时间值。0:00 是子夜；12:00 是正午。
2. 如果输入的时间发生重叠，则将返回第一个可接受的值。
3. 如果当前时间没有移位，则将返回一个空字符串。

示例（如果当前时间为 16:00）：

shiftcode（"7:00-15:00-白天|15:00-23:00-傍晚|23:00-7:00-夜晚"）="傍晚"

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

SpecificDateFormat函数允许您自定义日期戳，方法是将日期向前或向后调整一定间隔并在表达式中使用该自定义日期。
SpecificDateFormat 函数根据系统时钟按照默认日期格式或您选择的其他格式向前或向后调整当前日期。

- [date format] 指定要使用的日期格式。该参数必须用括号括起。请参看有效日期格式表
- +/[offset data] 此值指定日期的偏移量。间隔之前必须有加号 (+) 或减号 ()，具体取决于您要在当前日期的基础上增加时间还是减少时间。
- [date interval] 日期间隔必须为天 (d)、周 (w)、月 (m) 或年 (y)。

下表中是 SpecificDateFormat

函数在表达式中的不同格式的示例。（注意：对于这些示例，我们假定当前日期为 2013 年 7 月 29 日）

SpecificDateFormat("mm dd yy" , "+1y") = 7 29 14 （当前日期后的 1 年）

SpecificDateFormat("ww/m/yyyy" , "30w) = 1/12/2012 (当前日期之前的 30 周）

SpecificDateFormat("ddddd / wwww /mmmm" , "") = 41484/5928/1363 （从 1900 年开始计算的天数、周数和月数）

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

TimeOffset 函数允许您自定义时间戳，方法是将时间向前或向后调整一定间隔并在表达式中使用该自定义时间。TimeOffset 函数根据系统时钟按照默认时间格式向前或向后调整当前时间。

在表达式中使用 TimeOffset 函数时，必须使用以下参数书写函数：

- 仅可使用默认的“mm/dd/yyyy hh:nn:ss”格式。要以其他格式输入，需要将 TimeOffset 嵌入到 DateValue 函数，然后再整体嵌入到可以指定其他时间格式的 FormatDate 函数中。

- “+/offset time”此值指定时间的偏移量。偏移量之前必须有加号 (+) 或减号 ()，具体取决于您要在当前时间的基础上增加时间还是减少时间。
- "time interval" 时间间隔必须是秒 (s)、分钟 (m) 或小时 (h)。

下表中是 TimeOffset 函数在表达式中的不同格式的示例。（注意：对于这些示例，我们假定当前时间为 2013 年 7 月 30 日下午 4 点 12 分 10 秒）

TimeOffset(FormatDate(Now(), "mm/dd/yyyy hh:nn:ss"),"h+5") = 07/30/2013 21:12:10 （当前时间后的 5 小时）

FormatDate(DateValue(TimeOffset(FormatDate(Now(), "mm/dd/yyyy hh:nn:ss"), "n+10"), "mm/dd/yyyy hh:nn:ss"), "hhnn") = 16-22（当前时间后的 10 分钟）

today () 提供当前日期。

Week（日期）提供日期自变量的周数。

weekday（日期）提供日期自变量的每周日数。

注意：周日为每周的第一天。

示例：

1999 年 11 月 20 日周二， weekday(now()) = 3

WeekISO8601(Date, DateFormat)

允许您创建支持 ISO8601 的公式。ISO 8601 是一项针对日期和时间相关数据交换的国际标准。

返回值	日期	返回指定日期中的星期
Date	字符串	一种文本格式的日期，用于转化为日期值。

DateFormat	字符串	基本日期值的格式。 - 数字编码预定义格式。 （例如 1 dd/mm/yy，2 dd/mm/yyyy 等） - 0 或负值表示正在使用当前系统的日期格式 - 字符串值可直接对格式进行编码。（例如 "yyyy/mm/dd"）

示例：

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

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

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

year（日期）提供日期自变量的年份。

示例：

minute (now()) 提供当前的小时和分钟。

year (today()) 提供当前日期的年份。

逻辑函数

逻辑函数使您能检查是否符合一个或多个条件。

注意：Windows系统的区域设置可能会影响某些公式的输出。

注意：TRUE 等于 1，FALSE 等于 0

如果两个自变量都为真，则 and (expr_1,expr_2) 返回 TRUE，如果至少一个为假，则返回 FALSE。必须从逻辑值中计算自变量。

示例：

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

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

如果两个字符串相同，exact (string_1, string_2) 返回 TRUE，否则返回 FALSE。此函数区分大小写。

示例：

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

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

if (expr,Val_if_true,Val_if_false) 返回 Val_if_true 值，如果 Expr 为真；如果 Expr为假，则返回Val_if_false 自变量。

示例：

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

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

not (logical) 提供逻辑自变量的对立项。

示例：not(exact("string","string")) = 1

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

not(False) = 1 或 not(O) = 1

not(True) = 0 或 not(1) = 0

not(1+1=2) = 0

如果两个自变量中的一个为真，or (expr_1,expr_2) 将返回 TRUE，如果两个自变量均为假，则返回 FALSE。必须从逻辑值中计算自变量。

示例：

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

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

or(true,true) = 1 或 or(1,1) = 1

or(true,false) = 1 或 or(1,0) = 1

or(false,false)= 0 或 or(0,0) = 0

数学函数

以数值执行运算并得出数值结果。值可以是变量或常量。

注意：Windows系统的区域设置可能会影响某些公式的输出。

Abs(data): 此函数计算数据的绝对值（正数）。允许在数字之后使用字母。

示例

Abs(-5) = 5

Abs(5) = 5

base10tobaseX(string_1,string_2) 将 string_2 从基数 10 转换为基数 string_1

示例

如果名为基数 16 的栏位含有字符串 "0123456789ABCDEF"

BASE10TOBASEX(Base16,12) 生成 C

BASE10TOBASEX(Base16,10) 生成 A

BASE10TOBASEX("012345","9") 生成 13

注意：此公式中字符串2参数不能为负数

baseXtobase10(string_1,string_2) 将 string_2 从基数 string_1 转换为基数 10

示例

如果名为 16 的栏位包含字符串 "0123456789ABCDEF"

BASEXTOBASE10(Base16,"E") 生成 14

BASEXTOBASE10(Base16,10) 生成 A

BASEXTOBASE10("012345","9") 生成 13

Ceil(data): 此函数计算大于指定数的最小整数。允许在数字之后使用字母。

示例

Ceil(3.234) = 4

Ceil(7.328) = 8

Decimals(data1, data2): 此函数为 data1 显示 data2 个小数点位数。允许在数字之后使用字母。

示例

Decimals(4, 2) = 4.00

Decimals(3.524, 1) = 3.5

eval_add(«string»,«string»): 返回参数的总和。

示例

eval_add(5,5)=10

eval_div(«string»,«string»):返回参数的商数。

示例

eval_div(20,2)=10

eval_mult(«string»,«string»): 返回参数的乘积。

示例

eval_mult(5,2)=10

eval_sub(«string»,«string»): 返回参数的差值。

示例

eval_sub(20,10)=10

Floor(data): 此函数计算小于指定数的最大整数。允许在数字之后使用字母。

示例

Floor(3.234)= 3

Floor(7.328)= 7

hex(val_1,val_2) 利用 val_ 总值将 val_1 十进制数字转换为十六进制格式。

示例

hex(2,8) = 00000002

注意：此公式中val_1参数不能为负数

int(值）返回小于或等于值自变量的最大整数。

示例：

int (-5.863) = -6

int (5.863) = 5

max(data1, data2): 此函数显示的最大值。允许在数字之后使用字母。

示例：

Max(5, 12) = 12

min(data1, data2): 此函数显示最小值。允许在数字之后使用字母。

示例：

Min(5, 12) = 5

mod(val_1,val_2) 返回 val_1 自变量除以 val_2 自变量所得的余数。结果与除数有同样的符号。

示例：

mod (7,2) = 1

mod (-7,2) = 1

mod (7,-2) = -1

mod (-7,-2) = -1

quotient(val_1,val_2) 返回 val_1 自变量除以 val_2 自变量的整数结果。

示例

quotient(10,2) = 5

round(val_1,val_2) 返回自变量 val_1，该自变量四舍五入到 val_2 显示的位数。

- 如果 val_2 大于 0，val_1 则四舍五入到说明的小数位数。
- 如果 val_2 等于 0，val_1 则四舍五入到最接近的整数。
- 如果 val_2 小于 0，val_1 则四舍五入到小数点左侧。

示例：

round (4.25,1) = 4.3

round (1.449,1) = 1.4

round (42.6,-1) = 40

文本函数

注意：Windows系统的区域设置可能会影响某些公式的输出。

如果每个框中都含有一个字符，字符串可以被同化到表格中。它由自身长度设定（字符串中字符的总数，包括空格。）字符串中字符的位置与表格中字符的位置向对应，即：第一个字符在位置 1 上。

示例：位置 3 对应于字符串中的第三个字符。

cyclebasex ( ) 可以在任何类型的数据库计数系统中进行计数。必须在链接的表达式中设定编号系统。还必须相互指定起始值、每个增量的值以及副本数量。所有这些值都可以链接于标签中的其他栏位，但不能使用引号将栏位名括起。

示例：

如果名为基数 16 的栏位包含字符串 0123456789ABCDEF，则：

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 ( ) 为完整的循环创建一组的用户设定字符。

示例：

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 ( ) 可以使用您自己设置的数字顺序，而无需使用数字或字母的正常顺序（0,1,2 或 A,B,C）。

示例：

cyclenumber(1,3) 将会产生以下顺序的标签：1 2 3 1 2 3 1 2 3...

cyclenumber(1,3,1,2) 将会产生以下顺序的标签： 1 1 2 2 3 3 1 1 2 2 3 3 1 1...

cyclestring ( ) *可以使用完整的循环将一组字或字符创建为增量栏位。整个字符串必须使用引号 (" ")括起，并且必须使用分号 (;) 将每个字或每组字符与其他字或字符组隔开。*

示例：

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

以下示例用于使用除 O 和 I 以外所有字母的标签。

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

exact (string_1, string_2) 如果两个字符串相同，返回 TRUE，否则返回 FALSE。

示例：

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

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

extract(«string», «sep», «pos») 从字符串 «string» 中返回处于指定位置 «pos» 并由字符串 «sep» 分隔的数据构成的子字符串。

例如：

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

find（字符串,按键,起始开始）返回自变量字符串中第一个按键自变量产生的位置。字符串自变量中的搜索是从起始自变量（起始 >= 1）返回的位置开始的。如果没有产生按键自变量，函数将复位为 0。此函数区分字母大小写。

示例：

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

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

A0;

left (string,num_char) 返回从字符串自变量中提取的字符串。此字符串在字符串自变量的位置 1 处开始，其长度与 num_char 自变量相等。

示例：

left("Peter McPeepert",1) = P

left("Peter McPeepert ",5) = Peter

len（字符串）提供字符串自变量的长度。空格作为字符计算。

示例：

len("Paris,New York") = 15

len("") = 0

len(" ") = 1

lower（字符串）将文本字符串中的所有大写字母转换为小写字母。

示例：

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

LTrim(«string»): 此函数自动删除右侧数据开头和结尾的空格。

示例：

LTrim(" No."): No

mid（字符串,num_char）返回从字符串自变量中提取的字符串。此字符串从起始自变量（起始 >=1）的值相对应的位置开始，其长度等于 num_char 自变量的长度。

示例：

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

pad ( ) *向栏位左侧添加字符，从而指定整个输入的预定长度。任何字符都可以选作填充字符。*

示例：

如果名为 GREETING 的栏位显示值 HELLO，则：

pad(GREETING,8) = 000HELLO

pad(5,3,0) = 005

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

replace（字符串,起始,num_char,new_string）返回转换的字符串自变量。来自在起始自变量中设定的位置上的字符数（等于 num_char 自变量）已被 new_string 自变量替代。

示例：

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

replacestring(«string», «old_string», «new_string») 使用指定的 «new_string» 在字符串 «string» 中替换另外指定的所有 «old_string»。

例如：

replacestring("ABC12DEF12", "12", "") = ABCDEF

rept（字符串,num_char）给出一个字符串自变量重复num_char自变量数值的字符串。

示例：

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

right（字符串,num_char）提供构成字符串最后字符的字符串，其长度等于 num_char 自变量。

示例：

right("Purchase order",5) = order

RTrim («string»): 此函数自动删除左侧数据开头和结尾的空格。

示例：

RTrim("Part ") :Part

search（字符串,按键,起始）提供字符串自变量中第一次产生按键自变量的位置。搜索从起始自变量（起始 >= 1）设定的位置开始。如果没有产生按键自变量，函数将复位为 0。

示例：

search("Purchase order","order",1) = 10

search("Purchase order","c",1) = 4

Sort(«列表», «分隔符», «方向»): 返回已排序的项目列表。排序是通过字符串比较完成的。列表是一个字符串参数，包含多个项目，由逗号或分号作为分隔符。方向是一个可选整数，用于指定排序方向，0 = 升序，1 = 降序。如果未指定方向，则默认为升序。

例如：

Sort(“3;5;6;1”, “;”) = “1;3;5;6”

Sort("one;two;three", ";", 1) =“two;three;one”

StrAfter(«data»,«start after», «length»): 此函数返回指定字符之后的指定长度的字符串。

示例:

StrAfter("1234-5678", '-', 3)= 取紧靠连字符之后的 3 个字符 (567)

StrAfter("1234-5678", '-')= 取连字符之后的所有字符 (5678)

StrBefore(«data»,«start before», «length»): 此函数返回指定字符之前的指定长度的字符串。

示例:

StrBefore("1234-5678", '-', 2)= 取紧靠连字符之前的 2 个字符 (34)

StrBefore("1234-5678", '-')= 取连字符之前的所有字符 (1234)

SuppressBlankRows («string»): 将返回一个带有跳过空行的字符串。它可让 您创建对象，在其中放置您想要的字段，并禁止空 字段。

示例:

SuppressBlankRows({Var0} & char(10) & {Var1} & char(10) & {Var2}) （ Var0、Var1 和 Var2 是变量，char(10) 是表示新行的"\n"符号 ）

Variables/Values	Formula	将显示
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

SuppressEmpty(«列表», «分隔符»)： 删除列表中的空白项目。列表是一个字符串参数，包含多个项目，由逗号或分号作为分隔符。

例如：

SuppressEmpty("3;;;1", ";") = "3;1"

trim（字符串）返回转换的字符串自变量。删除字符串起始和末尾处的所有空格。两个字之间的空格数减为 1 个。

示例：

trim(" Purchase order") = Purchase order

trimall（*字符串*）*返回转换的字符串自变量。删除所有空格。*

示例：

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

Unique(«列表», «分隔符», «比较符»): 删除项目列表中的重复条目。列表是一个字符串参数，包含多个项目，由逗号或分号作为分隔符。可选的比较符表明是否区分大小写，0 = 区分大小写，1 = 忽略大小写。如果未指定比较器，则默认为区分大小写。

例如：

Unique(“3;5;3;1”, “;”) = “3;5;1”

Unique("one;ONE;two", ";", 1) = “one;two”

upper（字符串）返回转换的为大写字母的字符串。

示例：

upper("Purchase order") = PURCHASE ORDER

ztrim（）*在完全为数字的栏位中删除从左侧开始出现的所有 0。*

示例：

如果名为 WEIGHT 的栏位显示值 000200，则：

ztrim(weight) = 200

定义公式数据源的属性

命令：**Data source（数据源）> Formula（公式）> Properties（属性）。**

1. 在 Edit（编辑）框中直接输入公式。

   -或者-

   选择所需元素，然后单击 Insert（插入）。

2. 单击 OK（确定）。

提示：可通过双击某个元素来插入该元素。

注意：如果公式中使用的某个变量具有包含 &+-*/<>=^%,!\" 之中某个字符的名称，则必须使用括号 {} 将其引起来。

注意：动态预览，表示包括“输出”页面中所定义格式在内的当前公式计算结果。 如果出现错误，预览将显示为红色。如果获得的值被截断，则必须在修改 Output（输出）选项卡中指定的最大长度。

实践 - 计算特殊" 模数"

在此练习中我们将 EAN8 条形码"Customer_Code"转换为一个 2/5 Interleaved 条形码，我们使用公式”r;Formula_4_NewCustCode”来完成这个转换.

条形码有如下的一些属性:

- 符号体系: 打印机,
- 高度: 4毫米,
- 窄条宽度: 1毫米,
- 比率: 2,
- 人工识别符: 下面 / 居中,
- 与条形码之间的距离: 0毫米,
- 字符字体: 打印机字体..

打开标签文件 ORDER_WS2.LAB .

计算重量

创建一个公式并命名为 Formula_1_Weighted. 计算规则为：变量 Customer_Code 的第一个字符乘以1, 第二个乘以2, 第三个乘以1, 第四个乘以2, 等等.

变量的最大输出长度为 6.

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

将计算出的重量结果加起来:

接下来的一步是将前面公式中得到的结果加起来. 最大的输入长度为2.

创建第二个公式并命名为 Formula_2_Sum.

计算校验位:

利用前面的结果，我们来计算校验位的值.

创建第三个公式并命名为 Formula_3_CheckDigit.

表达式如下:

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

将原来条码数值与校验位组合起来:

当创建条形码的时候必须包括原有数值和校验位 (Formula_3_CheckDigit).

创建第四个公式并且命名为 Formula_4_NewCustCode. 此公式将变量 Customer_Code 与校验码 Formula_3_CheckDigit 串接起来.

创建条形码:

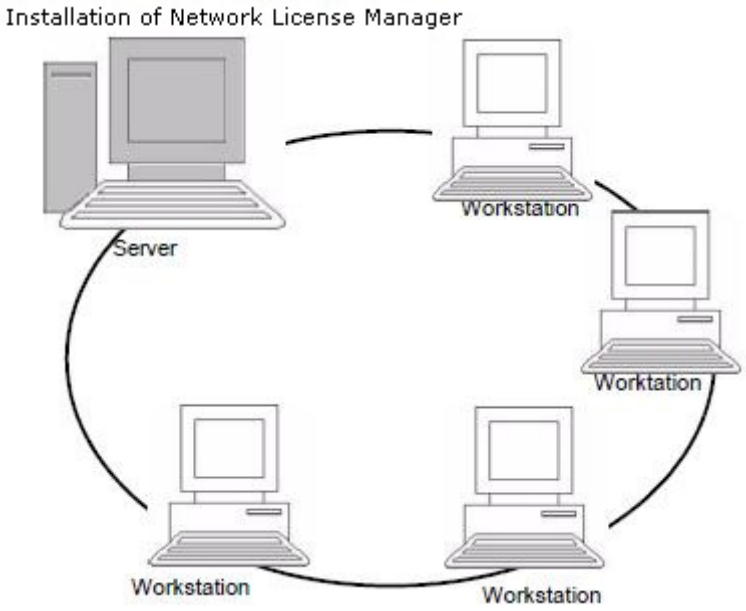
1. 选择公式 Formula_4_NewCustCode 并将它拖到设计区域 Customer_Code 条形码的位置.
2. 设置条形码的属性.

安装网络版

功能介绍

利用网络（多用户）软件包，您可以通过网络控制对标签设计软件许可证的访问权限。 使用此实用程序，您可以让很多用户同时从网络中的任意位置访问标签设计软件。

若要使用此标签软体的网路/ 多重使用者版，您必须在伺服器或作为伺服器的工作站上安装Network License Manager， 然后在每个工作站上，安装标签设定软体。



安装程序

网络配置

安装软件之前，网络管理员必须先为用户组定义网络结构，特别是：

- 定义要在其中安装 **Network License Manager** 及加密狗的许可证服务器。
- 定义要使用标签设计软件的工作站或客户端工作站

Network Manager 介绍

NNetwork License Manager 可让您使用标签设计软件的网络 配置。**Network License Manager** 包括：

- Network License Manager (**License Service**)
- **Network Settings Wizard**： **Network Settings Wizard** 可帮 助您定义网络配置。

安装 Network and Users Utilities

在将使用标签设计软件的所有工作站上安装此软件之前，必须先在服务器上安装 License Service 实用程序，以对网络进行配置。

在服务器上安装 **Network License Manager**。

1. 将安装 DVD 插入适当的驱动器中。

   显示安装窗口。

   如果 DVD 光盘不能自动运行： 转到 Windows 资源管理器并**展开** DVD 驱动器盘符。双击 **index.hta** 。

2. 选择 **Network License Manager**，此实用程序包含 **License Service** 和 用户管理器。 然后单击安装按钮。
3. 按照屏幕上的说明操作。
4. 如果要定义网络配置的设置，请启动服务器上的 Network Settings Wizard。默认情况下，如果不修改配置，每个 工作站将拥有自己的设置。

配置

配置网络版所需的所有工具都可以从 **Network toolbar**（从 Windows 任务栏 (Systray) 访问）获得。

从 Windows 任务栏，双击图标 以显示 **Network toolbar**。

Network Settings Wizard 可帮助您定义网络版本的设置。

1. 要启动 Network Settings Wizard，请单击图标 。

2. 在向导的步骤 1 中，选择一个设置模式： 普通、根据用户或根据工作站。

    - 普通： 所有用户将在所有工作站上使用相同的设置。(user.ini)
    - 根据用户： 每个用户可以在任何工作站上访问各自的设置。 (user name.ini)
    - 根据工作站： 每个工作站具有各自的设置(station.ini)

3. 在步骤 2 中，指定您要存储这些设置的位置。如果想要在各个工作站之间共享这些设置，应指定所有工作站都可以访问的网络路径。（示例： TKDongle）。

4. 在步骤 3 中，指定您要存储共享数据（变量、列表、打印日志文件等）的位置。 请确保所有用户对这些文件夹都具有正确的访问权限。

启动 License Service

必须先确保已启动 License Service，然后才能在所有工作站上安装标签设计软件。

License Service 已安装为服务。 您无需启动它。实际上，服务会在工作站启动时启动，而且只要工作站处于打开状态，它就会作为一项后台任务运行。

如果将网络服务器安装为受软件密钥保护，则您必须 对许可证进行激活，然后它才会自动启动。

启动服务控制器

单击网络工具栏上的图标

- 要么 -

椭鬟双击 TkxWebLicenseServerController.exe 文件。(C:\PROGRAM FILES\TEKLYNX\NETWORK\TOOLS\DONGLE)

- 要么 -

右键单击 Windows 任务栏中的**网络工具栏**图标，然后选择**许可证服务控制器**

在工作站上安装软件

必须将标签设计软件安装在要使用该软件的所有工作站上。

在工作站上安装此软件

1. 将安装 DVD 插入适当的驱动器中。

   将显示安装窗口。

   如果 DVD 不能自动运行：转到 Windows 资源管理器并展开 DVD 驱动器盘符。双击 index.hta

   。 。

2. 选择要安装的产品，然后单击安装按钮，并按照屏幕上的说明进行操作。
3. 启动标签设计软件。 从工具菜单中，选择网络管理。

   或 -

   从 Windows“开始”菜单，选择标签软件组中的“网络管理”快捷方式。
4. 启用使用网络许可证。
5. 选择网络许可证类型。

   - 共享文件夹许可证将使用 Windows 的文件共享功能，在软件和许可证管理器之间进行通信。

   - Web 许可证将在软件和许可证管理器之间使用 http/https 通信。
6. 如果选择了 Web 许可证类型，请指定服务器端口。
7. 单击修改，选择安装了 License Service 和加密狗的服务 器。

   梆鹭

   单击浏览，自动搜索已安装 License Service 的服务器。

   如果已经配置了网络设置，则会显示一条消息，询问您是否要使用当前的网络配置。

8. 如果您要修改或配置网络设置，请单击 Network Settings Wizard 按钮。
9. 单击确定。
10. 重新启动该程序。

France
+33 (0) 562 601 080

Germany
+49 (0) 2103 2526 0

Singapore
+65 6908 0960

United States
+1 (414) 837 4800

Copyright 2025 TEKLYNX Corporation SAS. All rights reserved. LABEL MATRIX, LABELVIEW, CODESOFT, LABEL ARCHIVE, SENTINEL, PRINT MODULE, BACKTRACK, TEKLYNX CENTRAL, TEKLYNX, and Barcode Better are trademarks or registered trademarks of TEKLYNX Corporation SAS or its affiliated companies. All other brands and product names are trademarks and/or copyrights of their respective owners.

www.teklynx.com