

# **CRIMSON 3**

## 用户手册



---

版权所有 © 2003-2009 Red Lion Controls Inc.

全球范围内保留所有权利。

本手册包含的信息出于善意提供，但如有更改，恕不另行通知。本手册不提供任何保证，亦不代表 Red Lion Controls 方面的任何承诺。除非另有声明，否则用作示例的公司、名称和数据均为虚构。未经 Red Lion Controls Inc. 明确书面允许，不得通过任何电子方式或机械方式复制或传输本文档的任何部分。

Red Lion 徽标是 Red Lion Controls Inc. 的注册商标。

Crimson 和 Crimson 徽标是 Red Lion Controls Inc. 的注册商标。

所有其它商标均为其各自所有者的财产。

作者：Mike Granby、Jesse Benefiel。

---

---

# 目录

入门 .....	1
系统要求 .....	1
软件安装 .....	1
注册 .....	2
检查更新 .....	2
安装 USB 驱动程序 .....	2
故障排除 .....	4
后续步骤 .....	5
<b>CRIMSON 基础 .....</b>	<b>7</b>
提示框帮助 .....	7
窗口布局 .....	7
导航窗格 .....	8
资源窗格 .....	8
编辑窗格 .....	8
折叠窗格 .....	8
类别 .....	8
通信 .....	8
数据标记 .....	9
显示页面 .....	9
程序 .....	9
WEB 服务器 .....	9
数据记录器 .....	9
安全 .....	10
移动 .....	10
后退前进 .....	10
类别快捷方式 .....	10
项快捷方式 .....	11
导航列表 .....	11
文件夹操作 .....	12
列表和文件夹排序 .....	12
拖放操作 .....	12
在列表里搜索 .....	12
撤销和重做 .....	12
全局搜索 .....	13
数据库操作 .....	13
数据库标识符 .....	13
保存映像 .....	14
数据库保护 .....	14
转换数据库 .....	14

---

---

查找数据库错误 .....	15
下载至设备 .....	15
配置链接 .....	15
发送数据库 .....	16
提取数据库 .....	16
装载 COMPACTFLASH .....	17
格式化 COMPACTFLASH .....	18
发送时间和日期 .....	18
<b>使用通信 .....</b>	<b>20</b>
串行端口选择 .....	20
选择协议 .....	20
协议选项 .....	21
设备操作 .....	21
高级设置 .....	22
端口和设备使用 .....	23
网络配置 .....	23
以太网设置 .....	23
多个端口 .....	24
路由设置 .....	24
下载设置 .....	25
添加端口 .....	26
协议选择 .....	26
使用虚拟端口 .....	26
使用扩展卡 .....	27
从属协议 .....	28
选择协议 .....	28
添加网关块 .....	28
向块添加项 .....	29
访问单个位 .....	30
协议转换 .....	30
主与从属 .....	30
主与主 .....	31
哪种方法? .....	31
控制主块 .....	32
数据转换 .....	32
禁用通信 .....	33
<b>操作标记 .....</b>	<b>35</b>
关于标记 .....	35
数据源 .....	35
标记类型 .....	35
标记属性 .....	36
标记的优点 .....	36
编辑属性 .....	37

---

---

表达式属性.....	37
可译字符串.....	40
双向属性.....	41
操作属性.....	41
颜色属性.....	41
日志属性.....	42
创建标记.....	42
重复标记.....	43
编辑多个标记.....	43
使用复制自.....	43
使用选择性粘贴.....	43
属性选择.....	44
导入与导出.....	45
查找标记使用.....	45
数字标记.....	45
数据属性.....	45
格式属性.....	48
颜色属性.....	49
警报属性.....	49
触发器属性.....	51
绘图属性.....	52
安全属性.....	52
标志标记.....	53
数据属性.....	53
格式属性.....	55
颜色属性.....	56
警报属性.....	56
触发器属性.....	58
安全属性.....	58
字符串标记.....	58
数据属性.....	59
格式属性.....	61
颜色属性.....	61
安全属性.....	62
基本标记.....	62
标记数据流.....	62
数字标记读取流程.....	63
数字标记写入流程.....	63
使用写入时.....	63
数组标记属性.....	64
<b>使用格式.....</b>	<b>65</b>
格式类型.....	65
常规格式.....	66
数字格式.....	66

---

---

科学格式.....	67
时间和日期格式.....	68
IP 地址格式.....	69
双状态格式.....	69
多状态格式.....	69
<b>使用颜色.....</b>	<b>71</b>
着色类型.....	71
常规着色.....	71
固定着色.....	71
双状态着色.....	72
多状态着色.....	72
<b>创建显示页面.....</b>	<b>75</b>
编辑器基础.....	75
对页面进行操作.....	75
更改缩放级别.....	75
资源窗格.....	76
向页面添加项.....	77
对基元进行操作.....	77
选择基元.....	77
使用快速工具栏.....	78
在页面间移动基元.....	78
在数据库间移动基元.....	78
更改基元大小.....	78
使用布局控点.....	78
智能对齐.....	79
快速对齐.....	79
使用网格.....	80
对齐基元.....	80
为基元设置间距.....	81
为基元重新排序.....	81
重复基元.....	81
编辑多个基元.....	82
跳转至其它项.....	83
基元属性.....	84
显示或隐藏基元.....	84
定义基元颜色.....	84
定义闪烁颜色.....	86
定义双状态颜色.....	86
定义四状态颜色.....	87
定义混合颜色.....	87
定义颜色表达式.....	87
定义箱填充.....	88
定义填充格式.....	89
定义边缘格式.....	90

---

---

使用分组.....	90
创建和分解分组 .....	90
在分组内编辑.....	91
嵌套分组编辑.....	91
向基元添加文本 .....	92
向基元添加数据 .....	94
向基元添加操作 .....	98
保护操作 .....	98
启用操作 .....	99
前往页面操作.....	99
用户定义的操作 .....	100
按下按钮操作.....	101
更改值操作 .....	102
校正值操作 .....	102
播放音乐操作.....	103
登录用户操作.....	103
注销用户操作.....	103
向键添加操作 .....	104
编辑页面属性 .....	105
用户界面设置 .....	107
全局属性 .....	107
输入属性 .....	108
图像属性 .....	109
字体属性 .....	110
管理图像.....	110
管理字体.....	112
<b>基元类型.....</b>	<b>115</b>
核心基元.....	115
几何基元 .....	115
三维基元 .....	116
按钮基元 .....	116
文本和数据基元 .....	117
行基元.....	118
图像基元 .....	118
刻度尺基元.....	120
箭头 .....	122
多边形和星 .....	123
多边形.....	123
星 .....	123
提示框与标注 .....	124
半裁剪图.....	125
操作按钮.....	125
发光按钮.....	126
指示器 .....	127

---

---

双状态扳钮开关.....	128
三状态扳钮开关.....	130
双状态选择器开关.....	131
三状态选择器开关.....	131
旧版基元.....	132
椭圆片段.....	132
多信息滑块.....	132
系统基元.....	134
查看器格式.....	134
警报查看器.....	135
事件查看器.....	135
文件查看器.....	137
用户管理器.....	137
趋势查看器.....	138
触摸校准.....	140
触摸测试器.....	140
<b>本地化.....</b>	<b>141</b>
选择语言.....	141
配置自动翻译.....	142
翻译您的数据库.....	142
输入翻译.....	143
全局自动翻译.....	143
导出与导入.....	143
应用词典.....	143
预览翻译.....	144
切换语言.....	144
<b>使用小组件.....</b>	<b>145</b>
创建小组件.....	145
总结.....	149
这为何重要.....	150
细节.....	150
小组件数据定义.....	150
小组件存档.....	152
文件夹绑定.....	152
高级绑定.....	154
类匹配.....	154
绑定前缀.....	154
使用绑定至.....	155
详情小组件.....	155
启用详情创建.....	156
定义数据项.....	156
绑定的结果.....	156
多个详情页面.....	157

---



---

<b>使用数据记录器 .....</b>	<b>159</b>
创建数据日志 .....	159
批次日志记录 .....	160
控制批次 .....	161
日志文件存储 .....	161
日志记录流程 .....	162
访问日志文件 .....	162
<b>使用 WEB 服务器.....</b>	<b>165</b>
重要说明.....	165
WEB 服务器属性.....	165
添加网页.....	167
使用自定义网站 .....	168
创建站点 .....	168
嵌入数据 .....	168
部署站点 .....	168
<b>使用安全系统.....</b>	<b>169</b>
安全基础.....	169
基于对象的安全 .....	169
指定用户 .....	169
用户权限 .....	170
访问控制 .....	170
写入日志记录.....	170
默认访问权限.....	171
按需登录 .....	171
维护访问权限.....	171
操作前检查.....	171
安全设置.....	172
创建用户 .....	173
指定标记安全 .....	173
指定页面安全 .....	174
安全相关的函数 .....	174
<b>使用服务.....</b>	<b>175</b>
使用时间管理 .....	175
配置服务 .....	175
选择 SNTP 服务器 .....	177
时区配置 .....	177
使用 FTP 服务器 .....	178
配置服务 .....	178
FTP 安全 .....	178
使用文件同步 .....	179
配置服务 .....	179
使用电子邮件 .....	181
添加联系人.....	181

---

---

配置 SMTP .....	181
配置 SMS .....	183
<b>共享端口 .....</b>	<b>186</b>
启用 TCP/IP .....	186
共享所需端口 .....	186
通过另一个端口连接 .....	187
通过以太网连接 .....	187
纯粹虚拟端口 .....	189
限制 .....	189
<b>使用调制解调器 .....</b>	<b>191</b>
添加拨入连接 .....	191
添加拨出连接 .....	193
添加 SMS 连接 .....	195
SMS 消息处理 .....	195
检查调制解调器状态 .....	195
调制解调器通信故障排除 .....	197
使用多个接口 .....	197
<b>使用 USB 主机 .....</b>	<b>199</b>
内存条支持 .....	199
常规属性 .....	199
传输属性 .....	200
<b>使用程序 .....</b>	<b>201</b>
程序列表 .....	201
查找程序使用 .....	201
编辑程序 .....	201
获取帮助 .....	202
资源窗格 .....	202
程序数据类型 .....	203
程序属性 .....	203
添加注释 .....	205
返回值 .....	205
注意! .....	205
传递参数 .....	206
编程技巧 .....	206
多个操作 .....	206
IF 语句 .....	206
SWITCH 语句 .....	207
局部变量 .....	208
循环结构 .....	208
<b>编写表达式 .....</b>	<b>211</b>
数据值 .....	211
常量 .....	211
标记值 .....	213
标记属性 .....	213

---

---

页面属性 .....	213
通信引用 .....	213
简单运算.....	214
运算符优先级 .....	214
类型转换.....	214
比较值 .....	215
测试位 .....	215
多个条件.....	216
选择值 .....	216
操纵位 .....	216
与、或和异或.....	217
移位运算符.....	217
位非.....	217
索引数组.....	217
索引字符串 .....	218
相加字符串 .....	218
调用程序.....	218
使用函数.....	218
优先级总结 .....	218
<b>编写操作.....</b>	<b>221</b>
更换页面.....	221
更改数字值 .....	221
简单赋值 .....	221
复合赋值 .....	221
递增和递减.....	221
更改位值.....	221
运行程序.....	222
使用函数.....	222
运算符优先级 .....	222

---

---

# **CRIMSON 3**

## 用户手册

# 入门

欢迎使用 Crimson 3——Red Lion 公司广受赞誉的操作员界面配置软件的最新版本。如果您使用过较早版本的 Crimson，那么您很快就会注意到 Crimson 3 进行了大量改进，同时又保留了您早已习惯了的强大功能。如果您是第一次使用 Crimson，那么请确保至少阅读此手册开始的几章，从而了解软件如何运作。无论如何，您都能很快理解 Crimson 3 如何使设计强大美观的操作员界面系统更为简便快速。

## 系统要求

Crimson 3 可以在 Windows 2000 之后的各个版本的 Microsoft Windows 系统上运行。内存要求不高，任何满足其最低系统要求的系统均可流畅运行 Crimson。安装大约需要 100MB 的磁盘空间。最好使用分辨率足够的显示器，从而确保在编辑显示页面时无需滚动太多。推荐为 VGA 目标设备使用 XGVA PC。

## 软件安装

Crimson 3 的交付格式为 Windows Installer 软件包（msi 文件）。一般而言，您需要从 Red Lion 网站下载此文件，但是，如果您从其它来源下载了此文件，则请确保 Windows 信任软件包的数字签名，从而确保您使用的是正版 Red Lion 软件。

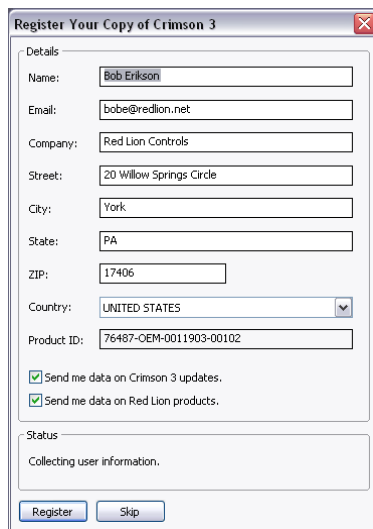


如上所示，发布者应显示为 Red Lion Controls Inc，您应该可以单击发布者名称，从而验证数字签名的完整性。如果您对软件包满意，请按下运行按钮，开始安装。

安装过程非常标准，除指定目标目录之外，应该无需过多人为干涉。安装过程结束之后，请查看开始菜单，找到 Red Lion Controls 文件夹。单击 Crimson 3.0 图标，启动软件。

## 注册

首次运行 Crimson 3 时，您会有机会注册软件。



Register Your Copy of Crimson 3

Details

Name: Bob Erikson

Email: bobe@redlion.net

Company: Red Lion Controls

Street: 20 Willow Springs Circle

City: York

State: PA

ZIP: 17406

Country: UNITED STATES

Product ID: 76487-OEM-0011903-00102

Send me data on Crimson 3 updates.

Send me data on Red Lion products.

Status

Collecting user information.

Register Skip

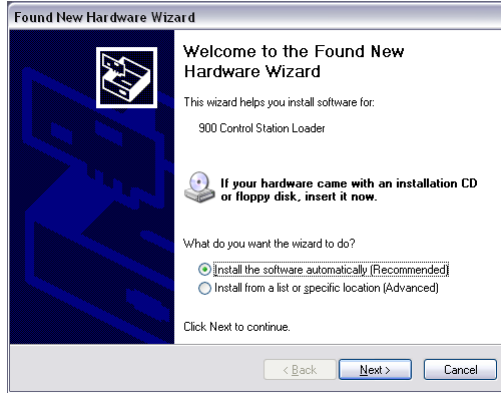
虽然注册是可选的，但是我们强烈推荐您抓住这个向我们提供您的联系方式的机会，这样我们就可以向您提供 Crimson 更新和相关产品的信息。因为注册需要 Internet 连接，所以如果您没有可用连接的话，您可以跳过这一步。如果您运行的是未注册版本的软件，Crimson 会定期提醒您。

## 检查更新

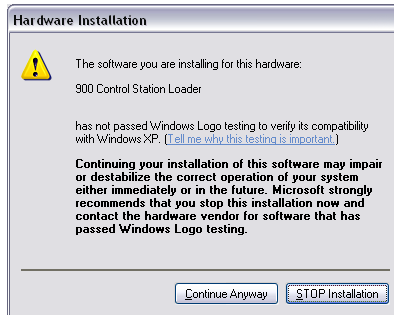
如果您有 Internet 连接，您可以使用帮助菜单里的检查更新命令来扫描 Red Lion 网站，查找是否存在新版本的 Crimson 3。如果发现了比您正在使用的版本更新的版本，Crimson 会询问是否应自动下载升级文件并更新您的软件。您也可以手动从 Red Lion 网站支持部分的下载页面下载升级文件。

## 安装 USB 驱动程序

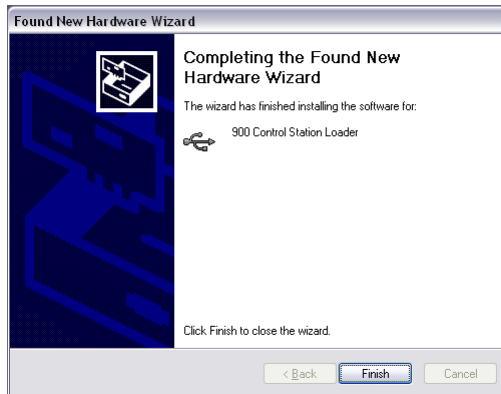
如果您遵照了目标硬件的配套指示，那么您应该还没有将硬件连接至您的 PC。现在您已经完成了 Crimson 3 安装，您可以安全地使用标准 USB 线缆将设备连接起来。一阵查找之后，Windows 应通过类似这样的对话框指示其已发现新硬件：



保留设置的默认状态，从而允许 Windows 自动定位设备驱动程序，然后按下下一步按钮。再经过一阵查找之后（在安装了大量软件的慢速机器上这可能需要一些时间），会出现一个让人胆怯的对话框，警告说 Crimson 设备驱动程序尚未经过数字签名。



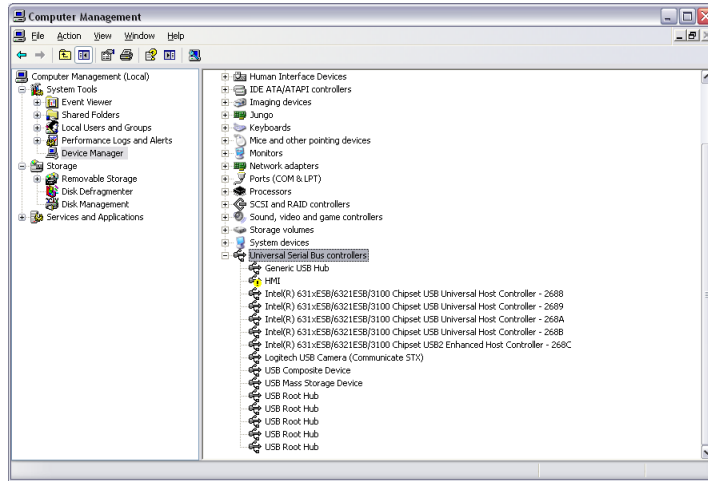
按下仍然继续按钮，很快您就会看到下图：



请注意，因为 Crimson 为启动加载器、Crimson 主应用程序和提供 CompactFlash 卡存储的复合驱动程序使用了不同的设备驱动程序，所以可能会要求您至多重复三次驱动程序安装过程。Crimson 3 不再要求为其连接的每个目标设备重新安装设备驱动程序，您注意到这一点时会松一口气。这意味着我们不能再期望一个 USB 段支持多个单一设备，但是我们确定您会发现这很值得。

## 故障排除

如果您没有遵照指示，而是在安装 Crimson 3 之前就将目标设备连至了您的 PC，或者看到那个吓人的未签名驱动程序对话框时紧张之下选择了停止，那么已中止的安装可能会使您无法正确安装驱动程序。要检查这一点，请找到我的电脑图标，右键单击后选择管理命令，打开 Windows 设备管理器。会出现类似于下图的窗口：

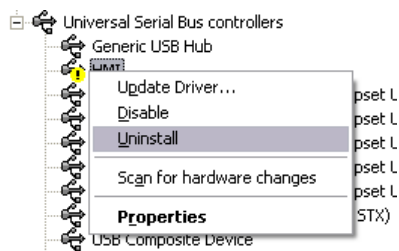


要做到这一点，各个操作系统上的精确过程可能有所不同，但基本理念是相同的：在桌面上或开始菜单里找到我的电脑图标，右键单击后选择管理。如果这样不起作用，请从控制面板里选择系统选项，然后从硬件选项卡打开设备管理器。

如果您的 USB 驱动程序有问题，您会在通用串行总线控制器类别下看到一个黄色的带有感叹号的图标。图标的名称可能是 HMI 或 Loader，或类似的名称。下图放大了显示了损坏的驱动程序：



要修复此问题，请右键单击损坏的设备，然后从菜单里选择卸载：



要求确认之后，Windows 会从您的系统里删除此设备。现在您可以关闭 Crimson 目标设备的电源。几秒之后，重新通电，Windows 会再次开始驱动程序安装过程。

如上所述，Crimson 实际上为启动加载器和 Crimson 运行时使用了不同的设备驱动程序。因此，您可能必须为每个驱动程序重复此修复过程，尽管如果安装失败的话，一切都不太可能越过启动加载器这一关。



## 后续步骤

因为 Crimson 3 使用了崭新的用户界面，所以我们建议现有 Crimson 用户至少快速浏览一下下一章。我们还建议您阅读与标记和显示页面配置相关的章节，因为 Crimson 3 简化了 Crimson 2 使用的一些概念，许多功能均可更易实现。如果您完全没有使用过 Crimson，那么请试着至少阅读至小组件章节。

祝您好运，使用愉快！

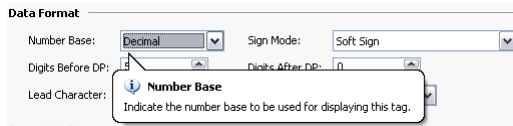


# CRIMSON 基础

要运行 Crimson，请选择开始菜单里的 Red Lion Controls 部分的 Crimson 3.0 图标。几秒之后 Crimson 会出现。您会注意到的第一件事情就是我们采用了全新的用户界面。这一全新界面允许更为快速的导航和更为迅速的数据库建设。我们希望它可以大大提高您的工作效率。

## 提示框帮助

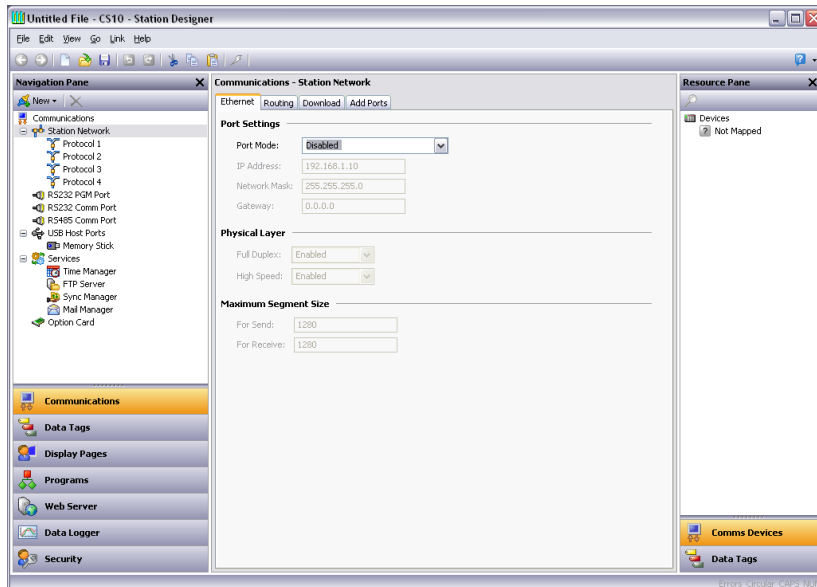
您需要了解的第一样东西就是叫做提示框帮助的有力功能：



此功能允许您查看 Crimson 内各个项的帮助信息。可以通过工具栏右侧边缘的图标或帮助菜单里的选项来控制此功能。默认模式允许按下 **F1** 键时显示帮助文本，如果您不确定如何进行给定字段的设置，帮助文本可以为您提供快速获取信息的方法，您的工作会变得更加简便！也可以选择鼠标悬停时模式，这样在一定时间段内鼠标指针悬停在特定字段时就会显示帮助；还可以选择选定模式，这样会始终为当前字段显示帮助。

## 窗口布局

Crimson 主窗口包含三个部分：



## 导航窗格

窗口的左侧部分叫做导航窗格，用于在 **Crimson** 配置文件的不同类别的项之间进行移动。每个类别均由窗格基部的一个栏代表，单击一个栏就会导航至相应部分。导航窗格的顶端部分显示了当前类别内的可用项，提供了用于操纵这些项的工具栏。如果您希望放大顶端部分，您可以选择并拖动分割顶端部分与类别栏的边框。

## 资源窗格

窗口的右侧部分叫做资源窗格，用于访问编辑当前类别时有用的各个项。它与导航窗格类似，包含了一些可以通过类别栏访问的类别。给定资源类别里的项可以拖放入您希望使用该项的位置。例如，可以从资源窗格内选择一个数据标记，然后将其放入配置字段，使该字段依赖于选定标记的值。许多项也都可以双击，从而为其设置当前字段。

## 编辑窗格

窗口的中间部分用于编辑当前选定的项。根据所选内容，它可能包含一些选项卡，每个选项卡都为选定项显示了一系列给定属性，也可能包含正在编辑的项特有的编辑器。

## 折叠窗格

导航窗格和资源窗格均可折叠至主窗口的边缘，从而为编辑窗格留出更多空间。要关闭窗格，请单击其标题栏左上角的‘x’。该窗格会被替换为标有箭头的较小的栏。单击此栏可以展开相关窗格。单击临时展开的窗格上的图钉图标可以将其锁定在其所在位置。

## 类别

**Crimson** 数据库内的主要类别如下：

### 通信



此类别指定了要在目标设备的串行端口和以太网端口上使用哪些协议。如果使用的是主协议（如 **Red Lion** 硬件初始化向或从远程设备的数据传输时所使用的协议），您可以使用此图标指定一个或多个要访问的设备。如果使用的是从属协议（如 **Red Lion** 硬件接受并响应来自其它系统的请求时所使用的协议），您可以指定要公布哪些数据项的读或写权限。您还可以使用此类别通过协议转换器在远程设备之间移动数据，并使用此类别配置扩展卡和服务。

## 数据标记



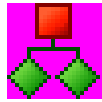
此类别定义了用于访问远程设备内的数据或在目标设备内存储信息的数据项。每个标记均有一系列属性，包括格式数据，指定了应如何在设备的显示或在网页之类的其它环境内应如何显示标记内的数据。通过在标记内指定此信息，Crimson 使您无需再在每次显示一个标记时重新输入格式数据。更多高级标记属性包括与标记相关的各种条件发生时激活的警报，或满足这些条件时会执行可编程操作的触发器。

## 显示页面



此类别用于创建和编辑显示页面。页面编辑器允许您显示各种叫做基元的图形项。这些可以是矩形和线之类的简单项，也可以是可绑定至特定标记的值或表达式的更为复杂的项，不一而足。默认情况下，这些基元使用标记创建时定义的格式信息，尽管如有需要可以重写此信息。您还可以使用编辑器定义按下、松开或按住键或基元时要执行何种操作。

## 程序



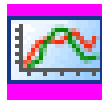
此类别用于使用 Crimson 独特的类似于 C 语言的编程语言创建和编辑程序。这些程序可以在系统内基于数据项执行复杂的决策或数据操纵操作。它们用于扩展 Crimson 软件内包含的功能，进而确保即使是最复杂的应用程序也可以轻松应对。程序可以调用各种系统功能，执行常见操作。

## WEB 服务器



此类别用于配置 Crimson 的 Web 服务器、创建和编辑网页。Web 服务器可以通过一系列机制提供对目标设备的远程访问。首先，您可以使用 Crimson 创建包含标记列表的自动网页，每个列表都会根据标记的属性设置格式。其次，您可以使用 Microsoft FrontPage 之类的第三方 HTML 编辑器创建自定义站点，然后将特殊文本包含在内，指示 Crimson 插入实时标记值。最后，您还可以启用 Crimson 独特的远程访问和控制功能，此功能允许 Web 浏览器查看目标设备的显示和控制其键盘。Web 服务器还可用于从数据记录器访问 CSV 文件。

## 数据记录器



此类别用于创建和管理数据日志，每个日志均可向目标设备的 CompactFlash 卡记录任意数目的变量。可以快至每秒一次地记录数据。记录的值将被存储在可快速导入 Microsoft Excel 之类的应用程序的 CSV（逗号分隔的变量）文件里。可以通过换出 CompactFlash 卡或将卡装载为连接至目标设备的 USB 端口的 PC 的驱动器访问这些文件，也可以使用以太网端口或调制解调器通过 Crimson 的 Web 或 FTP 服务器访问这些文件。

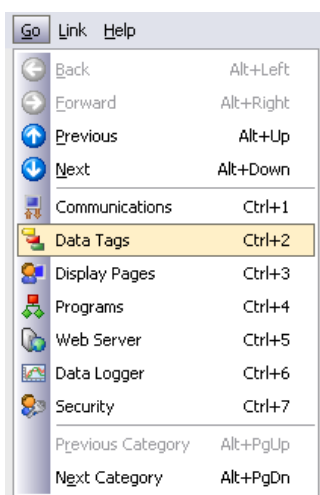
## 安全



此类别用于创建和管理目标设备的不同用户以及授予各个用户的访问权限。可以指定真实姓名，这样就使安全记录器不仅可以记录什么时候更改了什么数据，还可以记录是谁进行了更改。修改特定标记或访问页面的所需权限可以通过单个项的安全属性进行设置。还可以分配权限，从而允许或拒绝对 FTP 服务器或 Web 服务器的访问。

## 移动

要在 Crimson 数据库内移动，最简单的方法就是单击导航窗格里的类别栏，然后单击希望编辑的项。但是，还可以使用一系列快捷方式更快地进行移动，从而提高工作效率。这些快捷方式大多数都可以在前往菜单里或通过相关组合键使用：



### 后退前进

任务栏上的第一个图标或 **ALT+LEFT** 组合键可用于后退至先前选定的项。下一个图标或 **ALT+RIGHT** 组合键可用于再次前进，返回至最开始操作的项。在数据库类别之间进行切换时此功能非常有用。

### 类别快捷方式

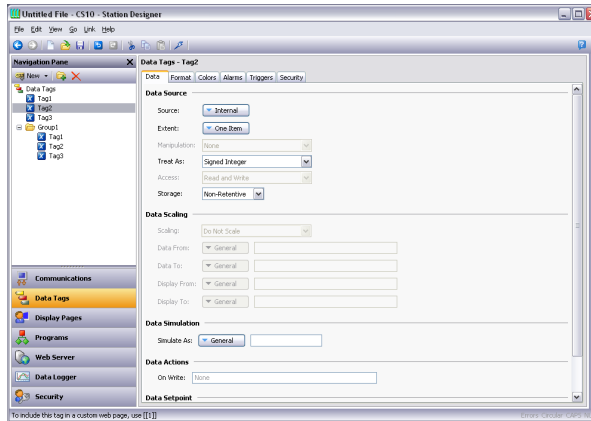
每个类别均分配有快捷键序列，其中包含了 **CTRL** 键和指示类别在导航窗格内的位置的数字。例如，可以直接使用 **CTRL+1** 组合键进入通信部分。您还可以使用 **ALT+PGUP** 和 **ALT+PGDN** 组合键在类别列表里上下移动。

## 项快捷方式

如果您在编辑窗格内操作，您可以使用 **ALT+UP** 和 **ALT+DOWN** 组合键在项之间进行切换。Crimson 将移动至项列表中的上一个或下一个项，并试图保持当前选定的数据字段不变。如果您希望在多个项上更改同一字段，这会非常有用，因为您无需再不断返回该字段或切换至导航窗格从而更改项。

## 导航列表

里的一些类别包含了项列表。例如，选择数据标记类别将使导航窗格显示数据库中全部数据标记的列表，允许对它们进行选择 and 编辑：



可以通过不同方法操纵这些导航列表里的项：

- 要快速查找一个项，请输入其名称开始的几个字符。Crimson 会选择匹配您输入的字符的第一个项。输入更多字符会使选择更加精确，而按下 **Esc** 会允许输入新的搜索字符。
- 要创建一个项，请单击导航窗格工具栏里的新建按钮。对于只支持单一类型的项的列表而言，您还可以使用 **ALT+INS** 组合键。工具栏上的新建按钮可以提供可用项的列表，允许您选择希望创建的项的类型。
- 要删除一个项，请使用导航窗格工具栏里的删除图标或按下 **ALT+DEL** 组合键。如果删除一个文件夹，则该文件夹里的全部项均将被删除。多次删除之前会出现警告，尽管它们始终可以通过撤销命令恢复。
- 要重命名一个项，请选择该项后按下 **F2**，然后，您可以输入新名称并按下 **ENTER**。您还可以选择该项后在其名称上再次单击，从而激活编辑，完成后同样按下 **ENTER**。

## 文件夹操作

一些列表支持将项分组进文件夹。可以使用导航窗格工具栏里的新文件夹图标创建文件夹，可以像常规项那样重命名和删除文件夹。为选定的文件夹创建项会将该项放入选定的文件夹。文件夹可嵌套至任何合理深度。

## 列表和文件夹排序

右键单击根项或文件夹，然后选择一个排序命令，即可对整个导航列表或文件夹内容进行排序。项可以按照字母升序或降序排列。无论使用何种排序方式，文件夹始终排在项前面。

## 拖放操作

导航列表里的项可以在列表内进行拖放，从而更改其位置或将其在文件夹之间进行移动。拖动时按住 **CTRL** 键会创建原始项的副本。项的从左到右的位置有时可用于决定在文件夹层次结构里的何处放置项。如果项被放入了错误的文件夹，请尝试将其向左或向右移动，从而获取正确位置。

通过打开两个 **Crimson** 副本，然后将相关项从源数据库的导航窗格拖入目标数据库的导航窗格，数据库项（如标记、显示页面或其它内容）也可在数据库文件之间进行拖动。如果尚未在目标数据库内选定合适的类别，临时按住一两秒正在拖入所需类别栏的项则会选定该类别，从而无需放弃并重复拖动操作。

## 在列表里搜索

虽然上文描述的快捷方式对于直接跳转至单一项而言非常有用，但是有时您也可能会希望查找名称中包含特定字符串的全部项。这可以通过使用导航窗格工具栏里的查找项命令实现。此命令将搜索当前列表，并将全部匹配项放入全局搜索结果列表。您可以使用 **F4** 键和 **SHIFT+F4** 组合键在此列表中移动，或按下 **F8** 键显示整个列表。请参阅本章后文了解更多关于全局搜索功能的信息。

## 撤销和重做

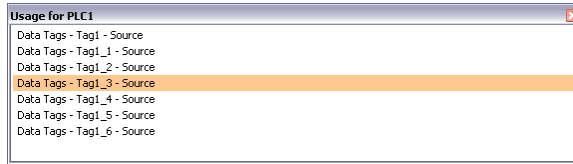
**Crimson 3** 使用了通用撤销和重做结构。这意味着您可以加载一个数据库，对其进行数小时的编辑，然后只需按住 **CTRL+Z** 组合键即可返回至数据库的原始状态。您还可以按住 **CTRL+Y** 组合键重新应用做出的更改。**Crimson** 会记住您的全部操作，撤销或重新应用更改时会自动在项和类别中进行导航。



## 全局搜索

Crimson 为在数据库内进行搜索提供了若干选项。从最简单的层面而言，您可以按下 **CTRL+SHIFT+F** 组合键在数据库中的任何位置搜索文本字符串。正如稍后您将看到的那样，您还可以搜索包含错误的表达式或引用标记或通信设备的项。这些搜索操作会将搜索结果放入全局搜索结果列表，允许您查看结果或在查找出的项之间前进或后退。

随时均可按下 **F8** 键显示结果列表：



窗口的标题栏描述了生成该列表的搜索操作，列表中的每一行都包含了对满足搜索条件的项的描述。上面的示例中，右键单击一个通信设备后选择查找使用命令列出了设备被引用的全部位置。双击给定项目将直接跳转至该项目，还可以使用 **F4** 键和 **SHIFT+F4** 组合键在列表中前进后退。也可以通过编辑菜单里的全局查找命令来使用与此功能相关的命令。

## 数据库操作

Crimson 将与特定配置相关的全部信息存储在叫做数据库文件的文件里。这些文件的扩展名为 `cd3`，尽管 Windows 资源管理器会在其默认设置下隐藏此扩展名。虽然 Crimson 3 数据库实质上是文本文件，但是它们经过了压缩，所以无法直接使用记事本之类的文本编辑器对其进行编辑。正如您所预期的一样，数据库通过文件菜单里的命令进行操纵。这些命令中的大部分都是标准的 Windows 应用程序命令，所以无需另行解释。

## 数据库标识符

Crimson 创建的每个数据库都分配有唯一的标识符。此标识符用于在下载新数据库时确定目标设备是否应清除其内部内存并删除记录在 CompactFlash 卡里的任何日志文件。如果标识符匹配设备中已经存在的数据库的标识符，则数据库仅被视作同一文件的不同版本，因此数据会被保留。相反，如果标识符不同，则数据会被清除。使用文件菜单里的另存为命令保存数据库文件的副本时，Crimson 会询问您是否希望分配新的标识符。如果这是新项目，请选择是；如果仅是在保存实质上同一数据库的备份副本，则请选择否。这会确保适当地清除或保留目标设备里的保持性数据。

## 保存映像

保存映像是文件菜单里为 Crimson 独有的一项。此命令允许创建可随后用于在终端上通过 CompactFlash 卡或可选的 USB 内存条更新数据库。文件包含了数据库的不可编辑版本以及运行所需的任何固件和启动加载器更新。将叫做 image.ci3 的映像文件放入目标设备的 CompactFlash 卡的根目录里，然后重设设备，会使用映像文件内容对启动加载器、固件和数据库进行更新。请注意，映像文件也可包含上传信息，允许从使用映像进行了更新的终端提取数据库文件的可编辑版本。

## 数据库保护

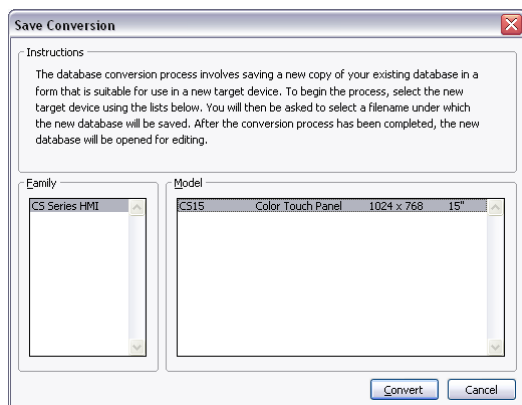
可以使用文件菜单里的保护命令对数据库进行密码保护：



默认访问权限参数用于定义在无需先输入数据库密码的情况下允许何种级别的访问权限。设置为只读访问权限将允许打开数据库，但是不允许进行和保存更改。设置为无访问权限将阻止没有密码时的全部访问。默认设置是完全访问权限，这允许打开数据库并进行编辑，而无需输入任何密码。Red Lion 收取象征性费用之后可以恢复遗忘的密码。

## 转换数据库

可以使用文件菜单里的保存转换命令将为一个目标设备设计的数据库转换成在另一个目标设备上使用的数据库。可以根据原始目标设备进行转换，但是大部分组合均可被支持。

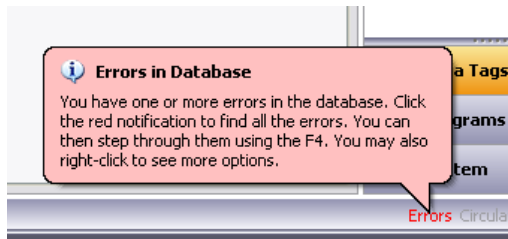


使用上面显示的对话框选择新目标设备之后，转换流程就会开始，然后会要求您输入新文件名。转换的数据库会存储在磁盘上。为避免意外破坏现有数据库，转换数据库时您只可以将数据库保存在新名称下。转换的数据库保存之后即会打开，以供编辑和查看。

转换流程会调整任何显示页面的大小，从而适应新的显示格式，并根据通信设备是否使用了 RS-232 或 RS-485 物理层而将其重新映射到新设备上的适当端口。如果新设备的通信端口比原始设备的少，则可能不能转换整个数据库。转换之后您可能必须进行一些调整。

### 查找数据库错误

某些操作可能会在数据库内生成错误。例如，您可能删除了一个通信设备，或者将一个标记设置为等于基于其自身的表达式，因此生成一个循环引用。Crimson 会通过显示在状态栏上方的红色提示框警告您出现了错误：



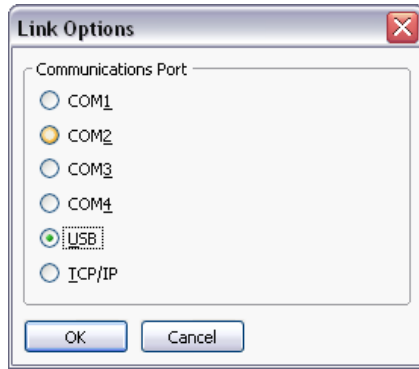
提示框会在几秒之后淡出，但是状态栏里的红色提示会一直存在，提醒您注意错误条件。单击指示器会搜索全部错误或循环引用，并将结果放入全局搜索结果列表，这样您就可以使用标准的 **F4** 键和 **SHIFT+F4** 组合键查看结果。您还可以右键单击指示器，从而访问重新编译整个数据库或优化设备通信组织方法的命令。因为一般而言 **Crimson** 无需用户干预即可执行必需步骤，所以很少需要手动重新编译数据库。

## 下载至设备

**Crimson** 数据库文件通过链接菜单下载至目标设备。一般而言，下载过程仅需几秒，但是，如果 **Crimson** 必须更新设备的固件或设备内没有当前数据库的较旧版本，则首次下载时可能需要更多时间。首次下载之后，**Crimson** 使用一种叫做增量下载的流程确保仅传输对数据库做出的更改。这意味着更新只需几秒即可完成，从而缩短开发周期，简化调试过程。

### 配置链接

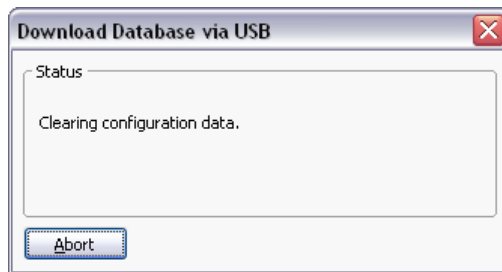
PC 和目标设备之间的编程链接可以使用 RS-232 端口、USB 端口或 TCP/IP 连接进行配置。虽然 TCP/IP 连接一般是通过面板的以太网端口配置的，但是也可以使用拨入链接来建立连接。下载之前，请使用链接-选项命令来确保选择了正确的方法：



请注意，使用 TCP/IP 进行下载时此对话框并不提供任何选择目标 IP 地址的方法。此信息现在存储在数据库文件里，通过网络配置项的下载选项卡进行配置。此方法使在多个数据库之间进行切换时更为简便，而无需每次重新配置目标 IP。

### 发送数据库

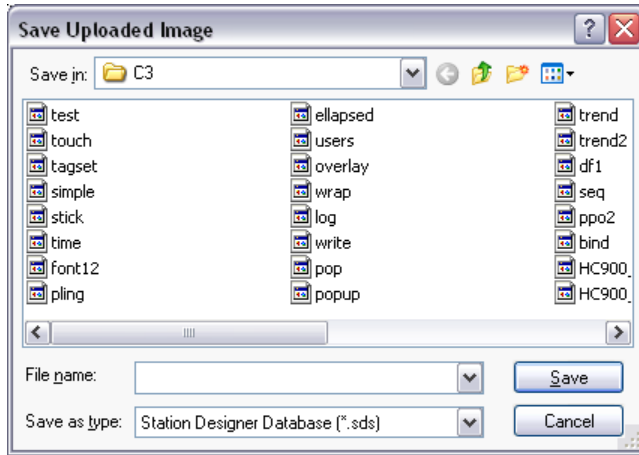
链接配置完成之后，就可以使用链接-发送命令或链接-更新命令来下载数据库。无论文件里的各个对象是否进行了更改，前者均会发送整个数据库。后者仅会发送更改，一般而言完成时间要短得多。因为如果增量下载出于任何原因而失败，Crimson 均将自动进行完整发送，所以您一般仅需使用更新命令。请注意，作为快捷方式，您还可以通过工具栏上的闪电标志或键盘上的 **F9** 键来使用链接-更新命令。



请注意，如果设备的固件要升级，则通过 TCP/IP 下载依赖于安装在面板里的 CompactFlash 卡。因为有时您可能希望进行这样的升级，所以强烈建议您在任何可能要进行 TCP/IP 下载的设备里安装 CompactFlash 卡。此外还请注意，必须通过通信类别里的网络设置启用 TCP/IP 下载。

### 提取数据库

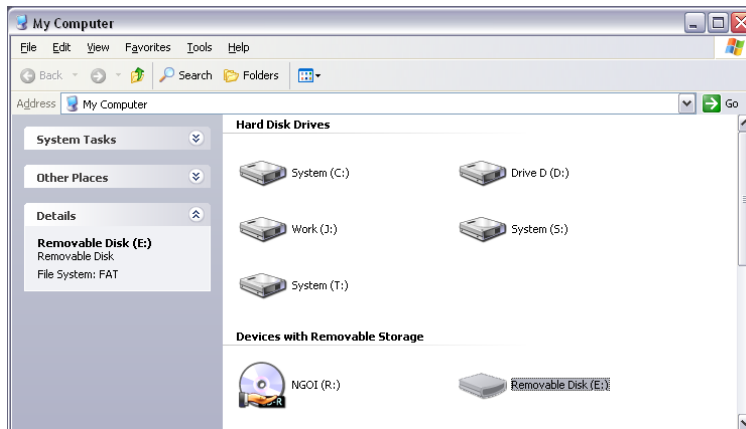
链接-支持上传命令可用于指示 Crimson 向目标设备发送数据库时是否应包含支持数据库上传所需的信息。此设置存储在数据库里，因此可以以单个文件为基础进行配置。支持上传会稍微减缓下载流程，对于包含嵌入图像的极大的数据库而言还可能失败，但是，其可以确保如果丢失了数据库文件，将能够从设备提取可编辑映像。



请注意，如果丢失了数据库文件，而且没有启用上传支持，则将不能重建文件，一切只能从零开始。要从面板提取数据库，请使用链接-提取命令。此命令将上传数据库，并要求输入保存文件要使用的名字。文件然后会被打开，以供编辑。如果数据库受到密码保护，则在打开文件之前还会要求输入密码。换言之，启用上传并不会避开密码保护。

## 装载 COMPACTFLASH

如果您通过 USB 端口连接至了合适的设备，您可以指示 Crimson 将设备的 CompactFlash 卡装载为 Windows 资源管理器内的驱动器。您可使用此功能将文件保存在卡里或从数据记录器读取信息。使用装载闪存命令和卸除闪存命令来装载和卸除驱动器。一旦发送了一个命令，则目标设备将重设，Windows 将刷新适当的资源管理器窗口。



请注意，装载 CompactFlash 卡时需要注意一些事项：

- 卡被装载时，目标设备将定期通知 PC 是否修改了卡上的数据。这意味着，如果在持续时间大于必需时间的数据日志记录操作期间进行卡装载，则 PC 和设备均将发生小幅性能降低。
- 如果从 PC 向 CompactFlash 卡写入，则 Windows 释放其在卡上的锁定之前目标设备将不能对卡进行访问。这可能需要几秒的时间，期间会限制数据日志记录操作，并阻止对自定义网页的访问。Crimson 将使用设备的 RAM

来确保不会丢失任何数据，但是，如果写入操作过多，导致卡被锁定超过四分钟，则数据可能会被放弃。

- 无论是通过资源管理器还是命令提示符，永远不应使用 Windows 来格式化通过 Crimson 装载的 CompactFlash 卡。格式化操作期间，Windows 无法正确地锁定卡，所以这种格式化可能并不可靠，进而导致后续数据丢失。关于如何可靠地对卡进行格式化，请参考下文。

## 格式化 COMPACTFLASH

唯一支持的卡格式化方法就是使用链接菜单里的格式化闪存命令。选择此命令会提示格式化流程将擦除 CompactFlash 卡里存储的全部数据，并提供取消操作的机会。如果选择继续，则会指示操作员面板对卡进行格式化。请注意，对于较大的卡而言，此过程可能持续数分钟。正在进行数据日志记录的面板上的慢速格式化可能会导致记录的数据中出现间隙。

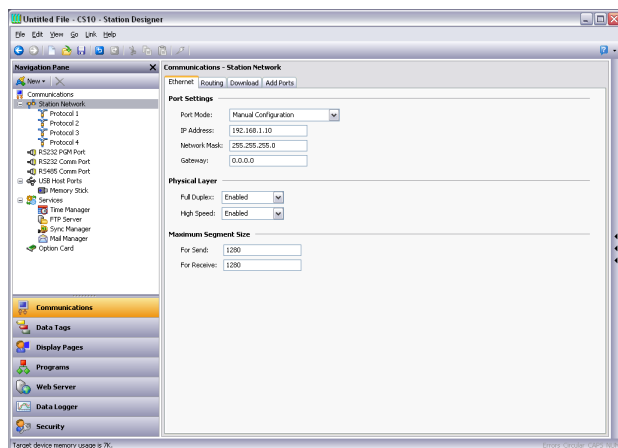
## 发送时间和日期

链接-发送时间命令可用于设置目标设备的时钟，从而匹配运行 Crimson 的 PC 上的时钟。此命令还会向目标设备发送当前时区和夏令时设置，允许使用时间管理器的高级功能。进行此操作之前，请确保 PC 的时间正确！



# 使用通信

创建 Crimson 数据库的第一阶段就是配置目标设备的通信端口，从而指示希望使用哪些协议以及访问哪些远程设备。这些操作通过通信类别进行。



正如可以看到的那样，通信类别以树状结构形式列出了单元的可用端口。上面显示的示例有三个主串行端口，还可以以扩展卡形式进一步添加两个端口。目标设备也可能提供一两个能够同时执行多个通信协议的以太网端口。

## 串行端口选择

决定目标设备的哪些串行端口用于通信时，请注意某些设备（和选件卡）会在多个端口之间多路复用单一串行控制器。这意味着如果一个端口用于从属协议，则另一个端口将不可用，如果使用了 DH-485 之类的令牌传递协议，则另一个端口同样将被禁用。如果您试图创建违反这些规则的配置，Crimson 会警告您。

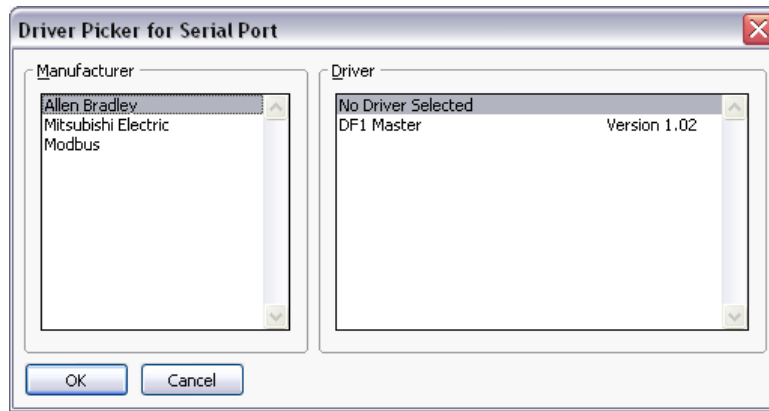
同时还要注意，目标设备的程序端口可用作额外的通信端口，但是这种情况下其不可用于下载。如果单元拥有用于下载的 USB 端口，那么这就不是问题。因此，如果希望通过编程端口连接设备，则强烈建议使用这种方法。如果未使用 USB，则必须提供通过执行响应用户操作的 `StopSystem()` 命令重新启用串行下载的方法。

## 选择协议

要为特定端口选择协议，请单击导航窗格里的端口图标，然后按下编辑窗格里的驱动程序字段旁边的选取按钮。



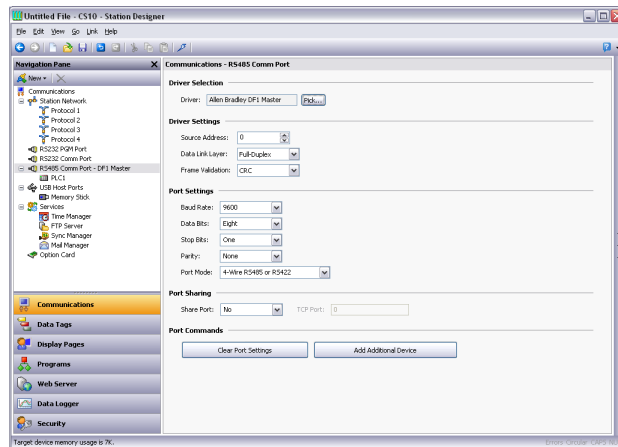
会出现下面的对话框：



选择适当的生产商和驱动程序，然后按下确定按钮，关闭对话框。然后会使用适当的协议配置端口，并在导航窗格里创建一个单一设备图标。如果配置的是串行端口，各个端口设置字段（波特率、数据位、停止位和奇偶校验）将被设为适用于相应程序的默认值。明显应检查这些设置，从而确保它们与要寻址的设备的设置相对应。

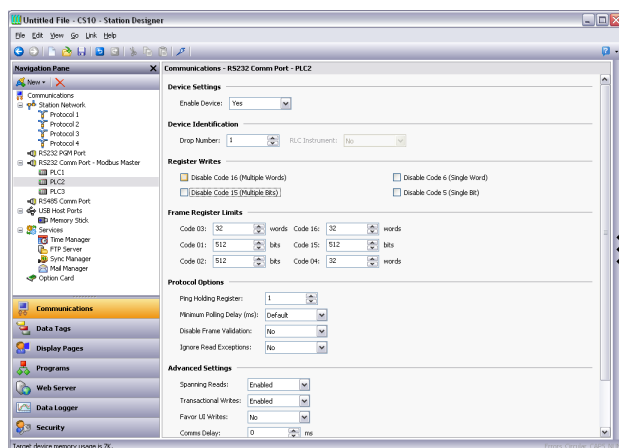
## 协议选项

某些协议需要进行独有的附加参数配置。选定相应端口图标时，这些会出现在编辑窗格里。下面的示例显示了 Allen-Bradley DF-1 驱动程序的附加参数，这些参数显示在编辑窗格的驱动程序设置部分下面：



## 设备操作

如上所述，选定通信协议时，会在相应的端口图标下创建一个单一设备。如果使用的是主协议，这代表了要通过协议寻址的第一个远程设备。如果协议支持对多个设备进行访问，则可使用编辑窗格里的添加附加设备按钮来添加更多目标设备，还可以使用导航窗格工具栏里的新通信设备命令。每个设备都由导航窗格里的一个图标表示，根据所用协议，还可以拥有一系列要配置的参数：



上面的示例中，选择了 Modbus Universal Master 协议，创建了两个附件设备，指示着一共有三个要访问的远程设备。编辑窗格显示了每个设备的属性。全部设备均显示有启用设备属性，其余字段则根据选定的协议进行显示。请注意，创建设备时 Crimson 为其指定了默认名称。可以通过选择导航窗格里的适当图标更改默认名称，按下 **F2** 键后输入新设备名称。

## 高级设置

除上述设备设置之外，某些主设备还提供一系列可用于优化通信行为的高级设置：

- **跨越读取** 选项指定了 Crimson 是否会通过读取数据块优化读取操作——即使这些块跨越了当前不在通信扫描上或未在数据库内引用的多个寄存器。例如，为引用寄存器 D1、D2 和 D4 寄存器的数据库启用跨越读取之后，会发出一个读取从 D1 开始的四个寄存器的单一通信命令。禁用跨越读取会为 D1 的两个寄存器进行一个读取操作，为 D4 的一个寄存器进行一个读取操作。
- **事务性写入** 选项指定了对 Crimson 里的数据值进行的一系列修改是否应导致一系列相应的写入操作，或是否仅应传输最后一个写入值。事务性写入使按键更换更为简便。
- **优先 UI 写入** 选项指定了是否应向用户操作直接导致的写入操作赋予优先级。对因协议转换或编程活动而执行大量后台通信的数据库进行操作时，这很有用。
- **通信延迟** 选项指定了将插入到此设备的任何两个通信事务之间的延迟。对性能低于 Crimson 性能的远程设备进行操作时，或要为设备赋予较低通信优先级时，这很有用。

## 端口和设备使用

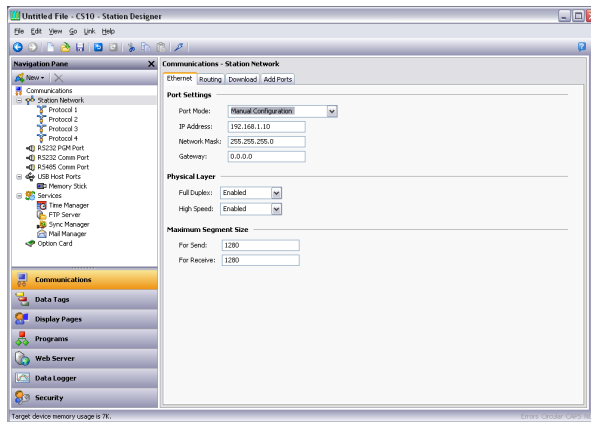
在导航窗格里右键单击给定通信设备或引用连接至特定端口的任何一个设备并选择查找使用命令，即可查找引用该设备的全部项。结果项将放入全局搜索结果列表，并可使用 **F4** 键和 **SHIFT+F4** 组合键进行访问。按下 **F8** 键可以隐藏或显示此列表。

## 网络配置

目标设备的 IP 网络配置通过导航窗格里的网络图标进行编辑。选定此图标时，编辑窗格会显示一些选项卡，每个选项卡均允许配置一系列给定属性。

### 以太网设置

前一两个选项卡用于配置目标设备的以太网端口：



### 端口设置

端口模式字段控制了端口是否启用以及端口获取其 IP 配置的方法。如果选择了 DHCP 模式，目标设备将尝试从网络上的 DHCP 服务器获取 IP 地址和关联参数。如果 DHCP 失败，则会自动使用 APIPA 分配 IP 地址。（如果单元配置为使用从属协议或为网页服务，则仅在 DHCP 服务器配置为向单元的 MAC 地址分配已知 IP 地址时，此选项才有意义，因为若如不然，用户将无法确定如何为设备寻址！）

如果选择了更为常用的手动配置模式，则必须使用适当信息填写 IP 地址、网络掩码和网关字段。为这些字段提供的默认值几乎不可能适用于您的应用程序！选择适当的值时，请确保咨询您的网络管理员。将目标设备连接至您的网络之前，请确保输入并下载这些值。如果不这样做，就有可能在您的网络里造成问题，尽管这种可能性微乎其微。

选择仅 IEEE 802.3 模式将为低级别通信启用端口，但是不会分配 IP 地址，也不会允许 TCP 或 UDP 进行操作。仅在使用某些生成自动化协议之类的使用原始以太网的驱动程序时，此模式才有意义。

## 物理层

物理层选项控制了设备将尝试与其连接至的集线器或交换机协商的连接的类型。一般而言，这些选项可以留为默认状态，但是如果建立可靠连接时（尤其是在不使用集线器或交换机而直接连接至 PC 时）遇到了问题，则请考虑关闭全双工和高速操作，从而查看这是否能够解决问题。

## 最大段大小

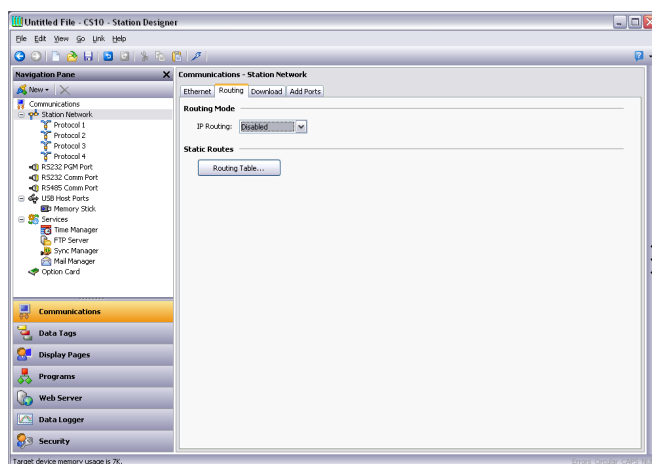
最大段大小选项为 TCP 发送和接收控制了 MSS 设置。一般而言，无需更改这些设置，因为默认值几乎适用于所有应用程序和网络。

## 多个端口

如果使用了多个端口，则请注意仅应有一个端口定义了默认网关，且每个端口均应拥有不同的网络地址。Crimson 仅将向一个接口发送给定 IP 数据包，因此，如果第一个以太网端口定义为 192.168.100.1，第二个以太网端口定义为 192.168.100.2，则这种配置将导致发送至 192.168.100.0 网络的全部数据包均进入第一个端口，进而导致第二个端口无法正确运行。

## 路由设置

第二个选项卡用于配置以太网路由选项：



## 路由模式

IP 路由选项用于启用或禁用接口之间的数据包路由。如果启用了此选项，则将按需转发以太网端口或调制解调器端口接收的目标为连接至另一个端口的设备的 IP 数据包。禁用此选项将阻止这种转发。所需设置依赖于您的网络拓扑。

## 路由表

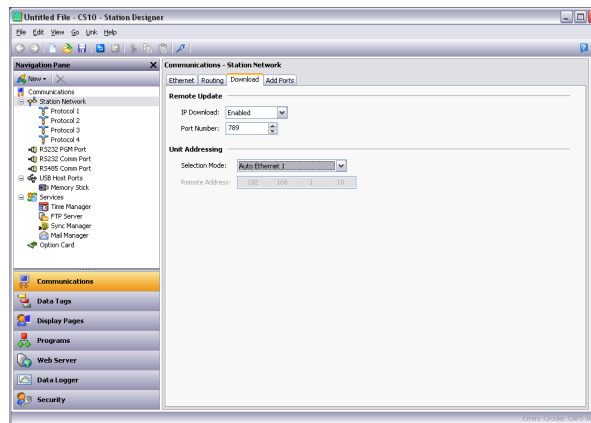
路由表为 Crimson 的 TCP/IP 堆栈定义了附加静态路由。



上面的示例中，定义了一个路由，指示 Crimson 将任何目标为以 192.168.3 开始的 IP 地址的数据包转发至地址为 192.9.200.8 的本地网络上的路由器。具体所需设置同样依赖于目标设备连接至的网络的拓扑。

## 下载设置

第三个选项卡用于配置通过 TCP/IP 下载至目标设备的下载：



## 远程更新

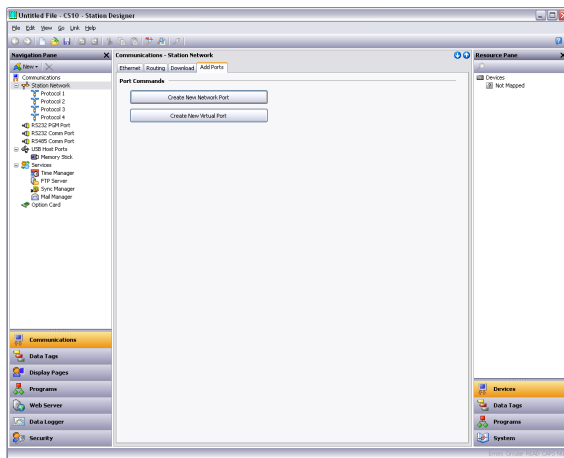
IP 下载选项用于启用或禁用 TCP/IP 下载，而端口号选项指定了要为这种下载使用哪个 TCP 端口。除非有更好的理由使用其它值，否则应使用默认值 789。

## 单元寻址

这些设置用于指定在链接-选项对话框里选择 TCP 下载方法时 Crimson 配置软件要使用的 IP 地址。自动模式会使用为选定的以太网端口配置的 IP 地址。（要让这有意义，显然必须手动配置端口。）手动模式允许通过远程地址字段输入 IP 地址。请注意，此信息被保存为数据库的一部分，允许方便地在同一网络上的不同单元之间进行切换。

## 添加端口

第四个选项卡可用于添加附加网络协议：



按下创建新网络端口按钮将进一步添加网络协议，数目最多为目标设备支持的端口数目。按下创建新虚拟端口按钮将执行类似的操作，但添加的是能够通过 TCP/IP 模拟串行端口的端口。在资源窗格里选择任一端口类型后按下 **ALT+DEL** 或选择删除工具栏选项，即可删除相应类型。

## 协议选择

网络配置之后，即可选择希望为通信使用的协议。可同时使用多个协议，许多协议均支持多个远程设备。这意味着决定如何混合协议和设备从而达成所需结果时可以有多种选择。

例如，假设您希望使用 Modbus 通过 TCP/IP 连接至两个远程从属设备。第一个选择是使用两个网络协议，将两者均配置为每个主机附加一个设备的 Modbus 主机。对于大多数协议而言，这将带来更高性能，因为这会允许两个设备同时进行通信。但是，这会使用两个可用协议，限制了在复杂应用程序里通过附加协议连接的能力。

第二个选择就是使用一个配置为 Modbus TCP/IP 主机的单一协议，但是还要进一步添加设备，这样两个从机均通过同一驱动程序进行访问。一般而言，这会带来略低的性能，因为 Crimson 会轮流轮询每个设备，而不是同时向两个设备通信。但是，这会节省网络协议，允许在不耗尽资源的情况下运行更复杂的应用程序。

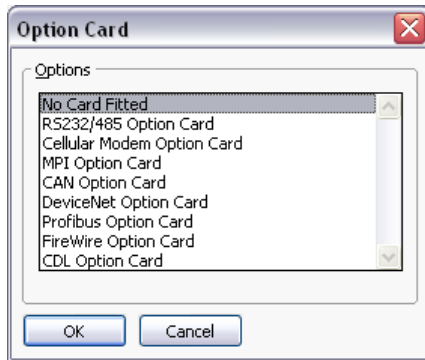
## 使用虚拟端口

如上所述，Crimson 支持向网络配置添加虚拟端口。虚拟端口像串行端口一样依赖于 Crimson 的通信系统，但是其数据通过 TCP/IP 链接发送和接收。虚拟端口可在主动模式或被动模式内进行配置。主动模式下，Crimson 将试图打开连接至指定远程设备的 TCP/IP 连接，而被动模式下，Crimson 会侦听特定 TCP/IP 端口的进站连接。一般而言，虚拟端口用于通过远程串行服务器与设备通信：使用了标准串行协议，但该协议的数据通过 TCP/IP 数据包进行封装。

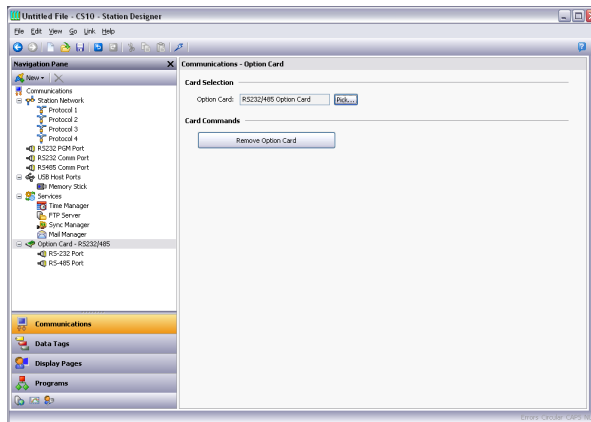
## 使用扩展卡

某些目标设备支持添加一个或多个扩展卡，从而提供更多通信功能。可使用多种卡，包括支持 CANOpen、Profibus 或 DeviceNet 之类的总线协议的型号。为每种卡均提供有安装说明，如需了解如何为设备挑选卡，请参阅提供的数据表。卡安装之后，选择导航窗格里的适当图标并单击选件卡属性旁边的选取按钮，即可进行配置。

将显示与下图类似的对话框：



选择适当的卡会向导航窗格添加一个或多个图标，代表通过卡可以使用的一个或多个附加端口。这些端口可以按照常规方式进行配置。下面的示例显示了一个串行扩展卡：



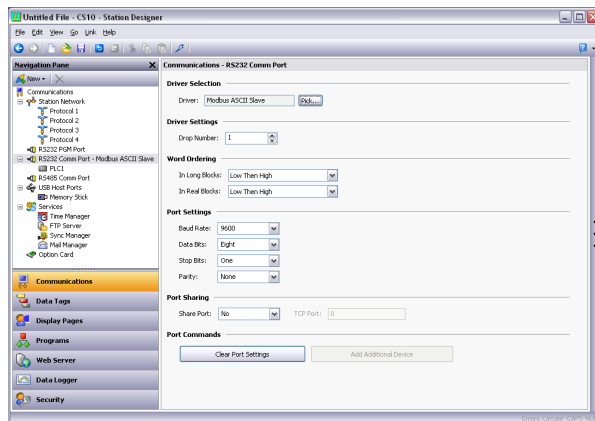
按照上文提供的说明可以配置附加端口。请注意，一个端口的可用驱动程序依赖于其支持的连接类型。例如，CANOpen 扩展卡显示了一个仅支持 CAN 通信标准专用的驱动程序的端口。

## 从属协议

对于主协议（如 Crimson 设备初始化通信所用的协议）而言，在通信类别下无需进一步进行配置。对于从属协议（如 Crimson 设备接收和相应远程请求所用的协议）而言，过程稍微复杂一些，因为必须指示希望公布哪些数据。

### 选择协议

对于主协议而言，第一个阶段是为希望使用的通信端口选择协议。下面的示例显示了目标设备的为使用 Modbus ASCII 从属协议而配置的 RS-232 端口：



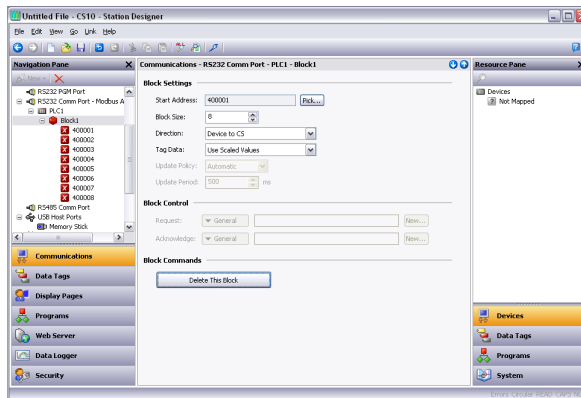
请注意，已为协议创建了一个设备。如果是主协议，这表示了 Crimson 将访问的远程设备。但是，这种情况下，设备表示了硬件自身作为的 Modbus 从机。这意味着只需要一个设备，并且硬件将响应的通信站号之类的事项通常通过端口设置而非设备设置进行配置。

### 添加网关块

协议配置之后，现在必须决定希望从属协议公布的地址范围。这个示例中，我们希望使用 Modbus 寄存器 40001 至 40008 来允许对数据库中的某些数据项进行读写访问。我们首先选择导航窗格里的设备图标，并单击编辑窗格里的添加网关块按钮。



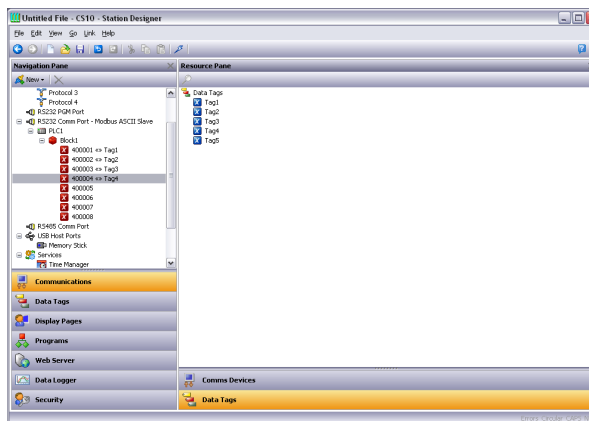
设备下会出现一个表示块的图标：



上面的示例中，我们将起始地址配置为 40001，指示这是我们希望块开始的地方。我们还将块大小配置为八，从而为每个我们希望公布的标记分配一个 Modbus 寄存器，并将方向配置为设备至 Crimson，从而指示我们希望远程设备能够读写通过此块公布的数据项。最后，我们将标记数据属性留为使用缩放值的默认设置，指示我们希望在标记数据传输至网关块之前为该数据应用缩放。

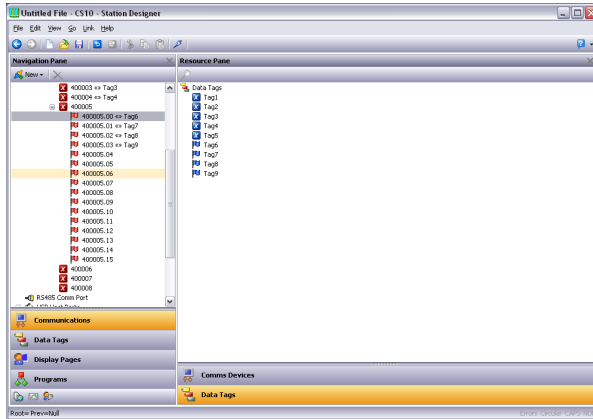
### 向块添加项

创建块并定义其大小之后，导航窗格里会出现一些项目，表示块公布至远程访问的各个寄存器。选定一个项目之后，一个展开的资源窗格会出现，提供对可用数据项的访问。这些项构成了来自数据库中的标记和来自自己配置的任何主通信设备的数据寄存器：



要指示希望网关块中的一个特定寄存器对应于一个特定数据项，只需从资源窗格将该项拖入导航窗格，然后放在适当的网关块项目上。上面的示例显示了块中的头四个寄存器如何映射至叫做 Tag1 至 Tag4 的标记，指示对 40001 至 40004 的访问应映射至相应变量。

### 访问单个位

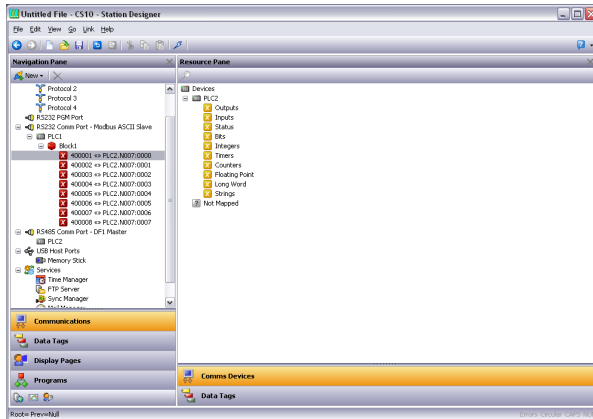


如果您的应用程序要求此项，您可以将网关块内的单个元素展开至其构成位。要完成该操作，请右键单击相关元素，然后从出现的菜单里选择展开位。导航窗格会更新为显示构成寄存器的单个位，这些位可以使用上文所述的拖放流程进行映射。

### 协议转换

除了通过从属协议公布内部数据标记之外，网关块还可用于公布从其它远程设备获取的数据，或在两个这样的主设备之间移动数据。这种独特的协议转换功能可使操作系统的元素之间整合更为紧密，即使使用的是简单的低成本设备。

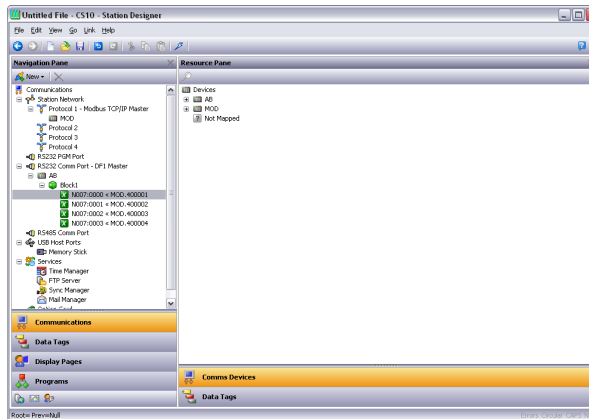
### 主与从属



通过从属协议从其它设备公布数据仅是上文所述的映射流程的扩展，只是这一次不是从资源窗格拖动标记，而是应选择通信设备类别，展开适当的主设备，然后拖动表示您希望公布的寄存器的图标。之后会要求您指定主设备的起始地址和要映射的寄存器的数目，会如示创建映射。

这个示例中，Allen-Bradley 控制器里的寄存器 N7:0 至 N7:7 通过 Modbus TCP/IP 作为寄存器 40001 至 40008 公布了访问权限。Crimson 会自动确保这些数据项从 Allen-Bradley PLC 读取，从而满足 Modbus 的请求，并会自动将向 Modbus 寄存器的写入转换为向 PLC 的写入。这种机制允许在以太网网络上连接即使是相对简单的 PLC。

## 主与主



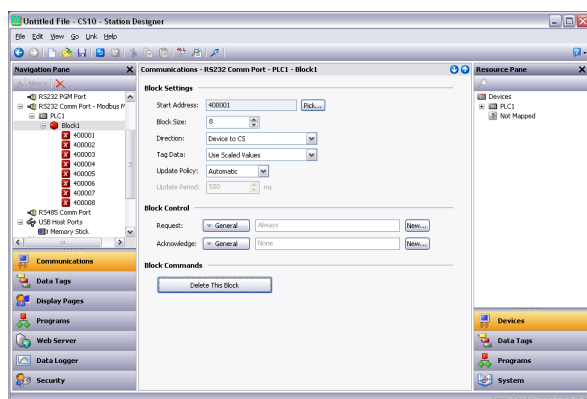
要在两个主设备之间移动数据，只需选择一个设备，并为该设备创建一个网关块。然后在从属协议上公布数据时可随意添加对其它设备的寄存器进行的引用。Crimson 同样会自动按需读取或写入数据，在设备之间透明移动数据。上面的示例显示了如何从 Modbus 设备向 SLC-500 移动数据。

### 哪种方法？

应如例所示在 Allen-Bradley 设备内创建网关块还是在 Modbus 设备内创建网关块？这是一个您可能想到的问题。第一件要注意的事情是要按单一方向进行传输仅需创建一个块。如果在 AB 内创建一个块去从 MOD 读取，并在 MOD 内创建一个块去向 AB 写入，那么这只是进行了两次传输，会降低性能！第二件要注意的事情是实质上可以随意决定哪个设备应拥有网关块。一般而言，创建块时，应尽量将数据库内的块数目降至最低。这意味着，如果 Allen-Bradley 内的寄存器在一个范围之内，但 Modbus 内的寄存器分散在整个 PLC 内，则网关块应在 Allen-Bradley 设备内创建，从而无需为访问 Modbus 地址空间的多个范围而创建多个块。

## 控制主块

主设备内的网关块有一些附加属性：



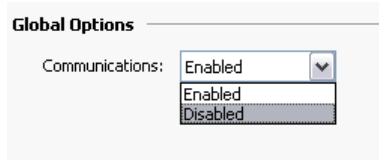
- 对于从属块而言，*标记数据* 属性选择了数据标记如何向块映射和从块映射。正如您将在下一章看到的那样，标记数据可能需经历各种转换阶段。此属性选择了转换过程中网关块将在哪儿获取和插入其数据。
- *更新策略* 属性用于定义块如何升级。默认设置为自动会导致读取块而不断更新，并写入块而仅传输更改了的值。设置为连续会导致全部块不断更新。设置为计时会导致全部块按照 *更新周期* 属性定义的速率进行更新，每次均会写入整个写入块的全部内容。
- *请求* 和 *确认* 属性用于通过标记或其它数据项控制块更新的计时。如果确认属性留空，则请求会作为启用字段，零值禁用块，非零值允许其运行。如果定义了确认，则请求和确认会作为标准的双线握手，块在请求的每个上升沿更新一次，处理完成之后则设为确认。

## 数据转换

还可以使用网关块进行 PLC 可能无法处理的数学操作。例如，您可能希望从 PLC 读取一个寄存器，对其进行缩放，并将平方根写回至另一个 PLC 寄存器。要完成此操作，请参阅数据标记部分，并创建一个表示将从设备读取的输入值的映射的标记。然后，创建一个表示输出值的标记，设置表达式，从而进行所需的数学计算。之后您可以创建一个目标为所需输出寄存器的网关块，拖动公式，从而指示 Crimson 将派生值写回 PLC。

## 禁用通信

Crimson 提供了一个使用通信类别顶层项包含的属性来禁用全部基于驱动程序的通信的选项：



开发期间您的当前位置没有可用的远程设备时，禁用通信可能很有用。在已禁用模式下操作时，Crimson 最初会将全部标记设置为等于其模拟值，然后会允许对它们进行更改，宛如它们被写入关联设备。如果您发现通信无缘无故停止，请确保您未将此设置为已禁用！



# 操作标记

为数据库配置了通信选项之后，下一步就是定义希望显示或操纵的数据项。这可通过选择编辑窗格里的数据标记类别来完成。可使用本手册前文提及的标准操作创建、删除或操纵标记。

## 关于标记

数据标记是表示数据项的命名实体。

### 数据源

标记可以从三种源获取数据：

- 标记可能被*映射*至远程设备里的一个或多个寄存器，这种情况下，引用或显示标记时，Crimson 会自动读取相应寄存器。同样，如果更改映射的标记，Crimson 也会自动向设备写入新值。
- 标记可能是*内部*的，这种情况下，它表示基于 Crimson 的设备里的一个或多个数据元素。内部标记可以标为保持性的，这样它们就会在动力循环之间保持值不变，也可标为非保持性，这样电源启动时它们会被重设为零。
- 标记可能是*表达式*，这种情况下，它表示基于其它数据项的计算，可选择使用数学运算符和一个或多个 Crimson 的内部函数或用户定义的函数。表达式标记用于为内部处理或向远程设备传输计算派生值。

### 标记类型

Crimson 支持三种主要类型的标记：

- *数字标记*表示整数或浮点值。
- *标志标记*表示开启或关闭值。
- *字符串标记*表示 Unicode 字符组成的字符串。

这三种主要类型的每一种均可表示一个单一值或一个值数组。数组是一组拥有类似属性的被分组在一起并通过索引值进行访问的项。映射的数组对应于目标设备内的多个寄存器。

第四种标记类型是 *基本标记*。这是一种仅能表示字符串或数字表达式的简化标记，没有标准标记的诸多强大功能，通常用于编码常量之类的简单数据项。

## 标记属性

Crimson 内的标记是可定义各种通用属性的多信息对象：

- 标记的 *标签* 是用于自动为引用数据项的数据字段进行标签的可译且可人工读取的字符串，还被 Web 服务器和数据记录器用于对关联的数据项进行标签。
- 标记的 *描述* 是用于对标记的目的进行注释的不可译字符串。目标设备的用户通常看不到描述，但是，出于诊断目的也可将描述显示出来。
- 标记的 *格式* 是定义标记数据的显示形式的一组设置。格式可设为常规，这种情况下，Crimson 会使用默认格式规则。格式也可设为多种格式类型中的一种。例如，数字值可以按科学格式进行显示，也可用于选择一系列不同的文本字符串。
- 标记的 *着色* 是用于定义如何显示标记的文本或要使用何种颜色表示标记状态的一组设置。存在一系列不同的颜色，允许标记根据不同条件更改其外观。前景色和背景色按对定义，可通过显示基元单独访问。
- 标记的 *安全描述符* 定义了更改标记时要使用的访问权限规则，以及是否应记录这些更改。

基本标记没有格式、着色和安全信息。此外，数字标记和标志标记还定义了警报和触发器，分别允许拉响警报和出现特定条件时采取操作。

## 标记的优点

因为 Crimson 允许直接将 PLC 寄存器放入显示页面，而无需定义数据标记，所以值得花一点时间指出使用标记时涉及的极少额外工作的益处：

- 标记允许您为数据项命名，这样您就可以知道正在引用 PLC 内的哪个数据项。此外，如果 PLC 内的数据移动了或您决定切换至一个完全不同的 PLC 系列，那么您就可以方便地重新映射标记，而无需对数据库进行其它更改。
- 标记使您无需多次输入相同信息。创建标记时，您可以指定如何显示标记。如果是数字标记，这意味着您指示 Crimson 应使用几个小数位，以及应向值附加什么单位（如有的话）。将标记放入显示页面时，Crimson 会知道如何为其设置格式，您无需再进行任何操作。同样，如果您决定更改格式，可能还会希望从一组单位切换至另一组，那么您可以一次完成这些操作，而无需轮流编辑每个显示页面。
- 标记用作对动画着色的基本方法。为标记定义的各种颜色可用于指定其它动画基元的显示方式。虽然还有其它方法，但更改显示基元的颜色时使用标记更为方便。
- 标记是实现从属协议的关键。Crimson 将这些协议视作在终端内用于公布数据项的机制。这允许通过多个端口访问相同数据，这样本地 SCADA 数据包和通过以太网从远程站点操作的类似的数据包就均可更改机器设置。如果没有标记，就没有要公布的内容，那么这种机制也就无法实现了！



- 标记在 **Crimson** 内用于实现许多高级功能。如果您希望使用警报、触发器、数据日志记录或 **Web** 服务器之类的功能，那么就必须使用标记。一般而言，这些功能都需要来自标记定义的格式数据，所以标记对它们的操作而言必不可少。

换言之，编程过程中，标记可以自动完成许多任务，大大节省时间。即使您决定不使用标记，本手册后面的许多章节也都参考了本章描述的概念。因此，继续之前请仔细阅读本章。

## 编辑属性

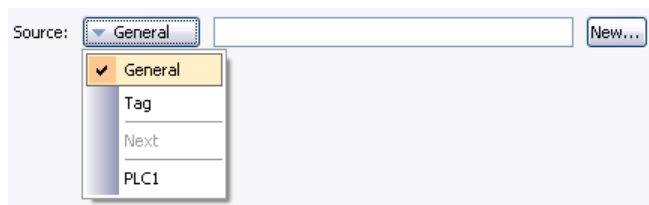
对于任何用过 **Windows** 操作系统的人而言，大多数属性的编辑方式都是不言自明的。例如，可能会要求您输入数字值或从下拉列表中选择一个项。但是，某些类型的属性提供有下面描述的更为复杂的编辑选项。

### 表达式属性

表达式属性可以设为：

- 常量值。
- 数据标记的内容。
- 远程通信设备里的寄存器的内容。
- 使用各种数学运算符连接在一起的这些项的组合。
- 本地程序的返回值。

默认状态下，紧跟属性标签之后的带箭头的按钮显示了字段处于常规模式之下。按钮右侧的编辑框可能会显示一个指示属性的默认行为的变灰的字符串。下面的示例显示了一个没有默认值的空表达式属性：



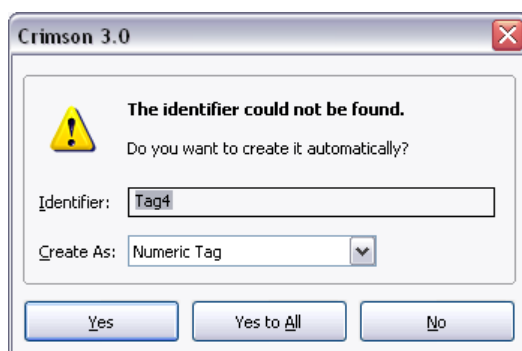
如果您熟悉 Crimson 的表达式语法（写入表达式章节详细描述了这种语法），您就可以直接在编辑框内输入表达式来编辑属性。

### 选择标记

要向现有标记设置表达式属性，共有四种方法。首先，可以在确保目标字段已选定之后在资源窗格内双击所需标记。其次，可以将标记从资源窗格拖放入目标字段。再次，可以从带箭头的按钮激活的下拉菜单中选择标记，但是请记住，从一开始就可以将目标拖入字段！最后，还可以依照老方法在表达式属性内输入标记名称。

### 创建标记

要向新标记设置表达式属性，同样也有四种方法。首先，对于定义数据项源的表达式而言，可以从带箭头的按钮激活的下拉菜单中选择新标记属性。其次，对于大多数其它表达式字段而言，可以按下常规模式下的编辑框旁边显示的新建按钮。再次，如果已经选定了标记，可以按下选取按钮后从出现的对话框中选择新标记。最后，可以输入作为表达式一部分的新标记的名称，然后 Crimson 会通过下面显示的对话框类似的对话框进行提示：



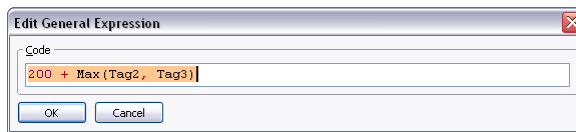
这个示例中，已输入了引用 Tag4 的表达式，但是数据库里不存在这个标记。Crimson 侦测到了错误，询问是否希望自动创建此标记。下拉列表可用于选择新标记的类型，会包含标记的使用情境中的适当选项。全部为是按钮用于指示 Crimson 使用默认数据类型来创建此表达式内缺少的任何其它标记，而且不会再次提示。

## 通信引用

要从通信设备选择寄存器，请从下拉菜单中选择设备。会显示一个对话框，允许您选择远程通信设备里的寄存器。菜单底部按照各个通信设备的创建顺序列出了这些设备。您还可以从下拉菜单中选择下一个选项，从而将当前标记设为等于最后一次使用的 PLC 寄存器加上映射至该地址的寄存器的数目。例如，将一个 32 位标记映射至 Modbus 寄存器 40001，然后选择下一个，会将后续标记映射至 40003。

## 编辑表达式

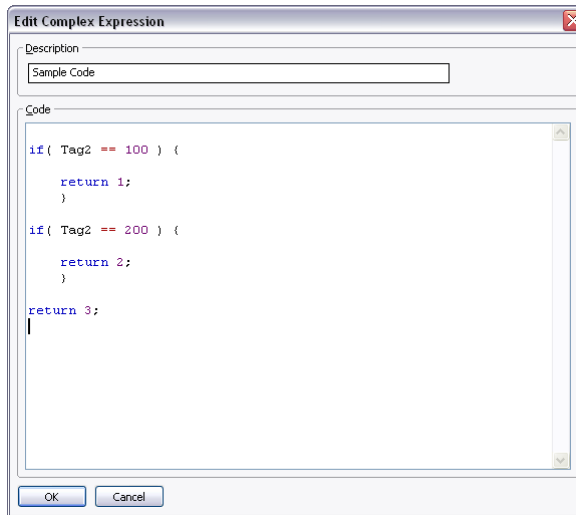
如上所述，常规表达式通常可直接在属性的编辑框内进行编辑。但是，要编辑常规表达式，还可以按下字段旁边的编辑按钮，进而激活一个允许显示表达式的更多内容的专用对话框：



此对话框中使用的编辑器与用于创建全局程序的编辑器相同，因此也提供了语法着色。将光标置于函数名称末尾后按下 **F1** 键，即可显示系统函数的相关帮助信息。

## 复杂表达式

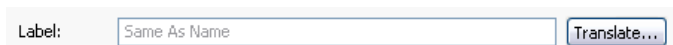
如果表达式太过复杂，无法显示在一行之内，那么可以从下拉菜单中选择复杂属性，从而允许创建一个局部程序：



Return 语句用于提供表达式的值，就像是调用了一个全局程序那样。请注意，使用了程序编辑器，提供语法着色和自动缩进功能，上文描述的 **F1** 机制可用于查看与系统函数相关的帮助信息。描述文本允许对程序的函数进行快速注释，这会显示在属性旁边，以供参考。请参阅本手册后面的章节，了解更多关于编写和编辑程序的信息。

## 可译字符串

Crimson 数据库可支持多语言操作，将显示给操作员面板用户的任何字符串均可藉此以多种不同语言之一进行显示。为允许您定义这些翻译，包含这种字符串的属性在其右侧拥有一个标注为翻译的按钮。

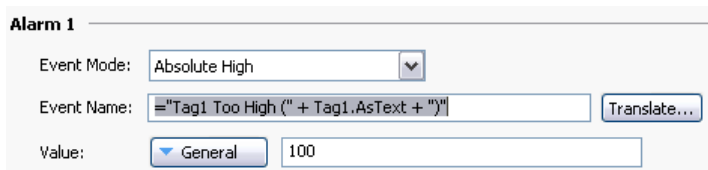


要输入翻译，请单击按钮，然后会出现下面的对话框：



对话框中列出的语言定义在数据库级别。请参阅本地化章节，了解如何选择这些语言、自动翻译功能操作和如何在运行时切换语言。请注意，如果未为特定语言输入翻译，但操作员随后选择了语言，那么 **Crimson** 将使用默认语言的文本。

可译字符串还可定义为等于表达式，进而允许其在运行时更改。例如，虽然通常会在配置过程中设置警报名称，但是数据库设计人员可能会希望警报包含触发警报的标记的值。如下例所示，就像编辑电子表格时那样，可通过在表达式前面加上一个等号来输入表达式：



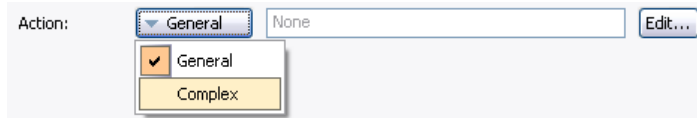
请注意，使用标记的 `AsText` 属性可允许其值根据其格式设置作为字符串进行访问。详情请参阅编写表达式章节。

## 双向属性

可以设为常量值或表达式的可译字符串之类的属性叫做双向属性。除了接受前面带有等号的表达式之外，这种属性还可通过将适当标记从资源窗格拖放入字段而设为标记值。

## 操作属性

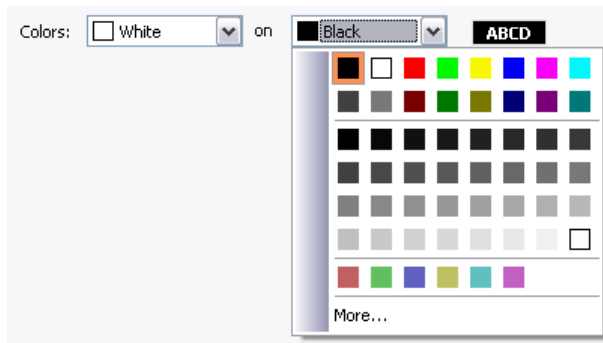
操作属性用于在标记内定义触发条件出现时要执行的操作。它们可通过与用于编辑表达式类似的下拉菜单和编辑框来进行编辑：



对于表达式而言，编辑按钮可用于调用较大的编辑窗口，可使用局部程序创建复杂操作。

## 颜色属性

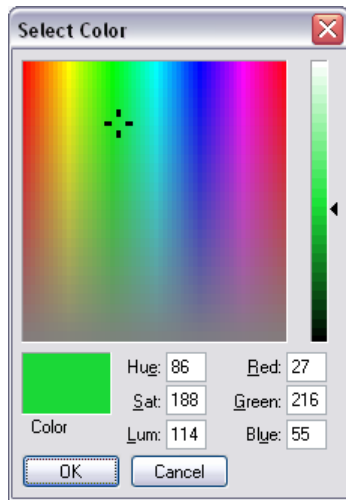
标记内的颜色属性表示了一对颜色，分别可用作以文本形式显示标记的状态时的前景色和背景色。下面的示例显示了一对正在编辑的颜色：



下拉菜单包含了下列颜色：

- 十六种标准 VGA 颜色。
- 三十二种黑色和白色之间的阴影灰色。
- 数据库中使用的任何其它颜色，可多达二十四种。

列表底部的更多选项可用于调用颜色选择对话框：



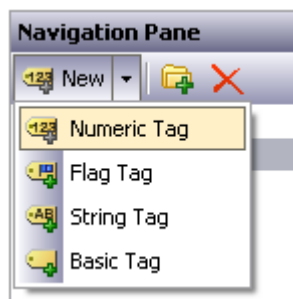
此对话框提供了若干种定义颜色的方式。可以从调色板选择，从“彩虹”窗口选择，或输入准确的 HSL 参数或 RGB 参数。如果选定的颜色尚未在数据库内使用过或不是标准颜色或灰色，则其将被添加至下拉菜单中显示的自定义颜色里。

## 日志属性

首次进入导航窗格的数据标记类别时，您会注意到与事件日志记录相关的一些属性。这些属性控制了是否以及如何将由标记或其警报生成的事件保存至 CompactFlash 卡。它们类似于数据日志定义的属性，因此，如需了解如何使用这些属性，请参考本手册后面的使用数据记录器章节。

## 创建标记

数据标记在导航窗格内通过常用方法创建和操纵。您会注意到，可以创建文件夹来组织标记，工具栏里的新建按钮有一个下拉箭头，允许您选择要插入的标记的类型。新建按钮左侧会创建与上一次创建的类型相同的标记，这样就无需使用下拉列表，使创建多个标记更为简便。



## 重复标记

编辑菜单里的智能重复命令可用于创建现有标记的新副本，将标记的数据源增量至引用下一个数据元素。

Crimson 可以选择通信设备里的下一个寄存器、数组的下一个成员或序列里的下一个标记，因此，“下一个”的定义依赖于数据元素的准确类型。例如，在一个映射至 Modbus 寄存器 40001 的 16 位标记上使用智能重复会生成一个映射至 40002 的标记，而在映射至 Array[2] 的标记上使用智能重复则会生成一个映射至 Array[3] 的标记。

此功能使创建引用顺序数据项的标记组更为简便。

## 编辑多个标记

您可能偶尔会希望同时编辑多个标记的属性。Crimson 支持这种操作，您可以编辑一个标记，然后将其它标记的属性设为等于首先编辑的那个标记的属性。Crimson 提供了两种进行这种操作的方法，这两种方法均依赖于相同的基础机制。

### 使用复制自

复制自命令可用于将给定标记的选定属性复制到导航列表里的一个或多个标记。要使用此命令，请选择目标标记，然后右键单击，打开上下文菜单。（请注意，通过使用常用的 **SHIFT+CTRL** 组合键，标记的导航列表可以支持多选。）选择一个复制自命令，然后光标会变化，允许选择一个要从其进行复制操作的标记。根据选定的命令，源标记的一个或多个属性会被应用到目标标记。

### 使用选择性粘贴

选择性粘贴命令可用于实现同样的结果，不过使用的是不同的方法，允许在多个数据库之间或多个 Crimson 运行实例之间复制属性。首先，选择源标记后使用复制命令将其放入剪贴板。然后，在导航列表里选择目标标记，请注意，可以进行多选。最后，右键单击所选内容，打开上下文菜单，选择选择性粘贴命令。会出现下面的对话框：



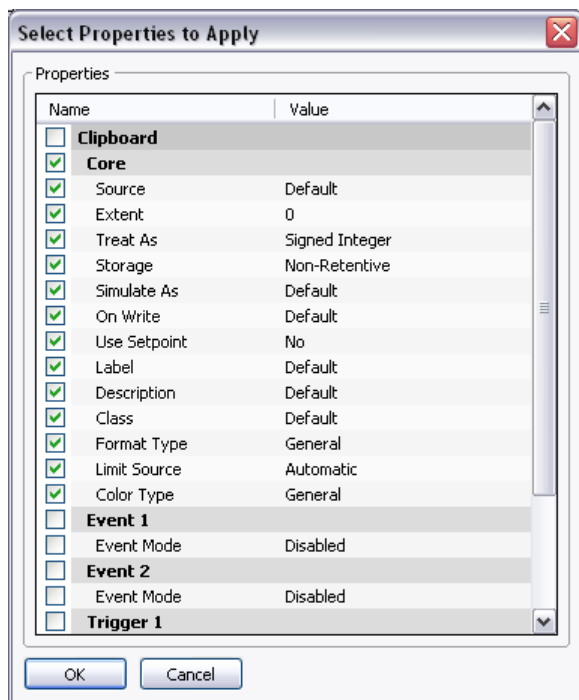
从源标记选择的属性会被应用到目标标记。

## 属性选择

上文描述的两种方法均允许定义要复制哪些属性：

- *映射*会复制源属性，还会复制控制该标记的通信选项（如范围、访问权限）的全部属性，以及数据源部分包含的其它全部属性。
- *缩放*会复制缩放至属性和关联的缩放限制。
- *格式*会复制格式类型和关联的格式对象。
- *着色*会复制着色类型和关联的着色对象。
- *警报*会复制警报 1 和警报 2 的全部属性。
- *触发器*会复制触发器 1 和触发器 2 的全部属性。
- *安全*会复制标记的安全页面的全部属性。

此外，*选择*选项可用于选择要复制的属性：



列表层次化地显示了源标记定义的全部属性，按照编辑标记时使用的布局对属性进行组织，并显示了分配给每个属性的值。可使用关联的勾选框选择或取消选择属性或属性组。仅会应用勾选的属性，提供对从一个标记向另一个标记复制何种内容的低层次控制。



## 导入与导出

选择导航列表里的数据标记项可以打开用于在数据库里导入和导出数据标记的按钮。标记可导出至 Unicode 文本文件或 ANSI 逗号分隔的变量文件，这两种文件均可使用 Microsoft Excel 之类的应用程序进行编辑。导出文件根据标记类型、格式类型和着色类型被分为多个部分，它们的含义可参阅下面的章节。

## 查找标记使用

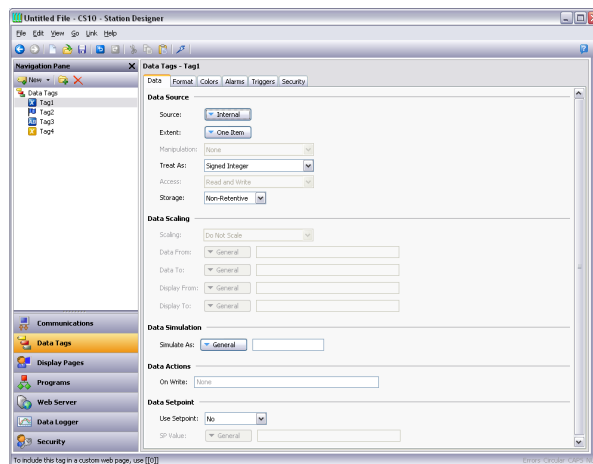
在导航窗格里右键单击给定标记并选择查找使用命令，即可查找引用该标记的全部项。结果项将放入全局搜索结果列表，并可使用 **F4** 键和 **SHIFT+F4** 组合键进行访问。按下 **F8** 键可以隐藏或显示此列表。

## 数字标记

数字标记表示了一个或多个整数值或浮点值。Crimson 进行全部内部计算时，均会使用 32 位有符号的整数或单精度浮点数，因此，进行处理之前，全部数据均会被转换为这两种形式之一。映射的数字标记支持在原始数据和将由 Crimson 使用的数据之间进行的多重转换。本章后面的内容详细定义了精确流程。

## 数据属性

数字标记在其数据选项卡里拥有下列属性：



## 数据源

- **源** 属性定义了标记从何处获取其数据。默认设置会使用内部标记，而下拉列表则可用于选择常规表达式、其它数据标记或远程设备的项。
- **范围** 属性用于在单一元素标记或数组之间进行选择。如果选择数组，则必须输入所需的元素数目。不允许为源是表达式的标记选择数组。对于映射的项而言，要从远程设备读取的寄存器的数目依赖于为地址定义的数据类型。例如，一个被映射至字作为长型类型的寄存器的有两个元素的数组会导致访问四个寄存器，每个长型值均需要两个字。一个映射至字作为字类型的数据的相似数组则只需要两个寄存器。

- *操作* 属性定义了通信数据传输进入映射的标记时要应用的第一阶段的转换。根据使用的确切数据类型，下列选项可能可用：

操纵	描述
无	数据不会更改。
颠倒位	数据中每个位的状态均会被颠倒。
反转位	数据中最有意义的位和最没有意义的位互换，中间位也会被同样处理。
反转字节	数据中最有意义的字节和最没有意义的字节互换，依此类推。仅大小大于等于 16 位的数据项才可使用此设置。
反转字	数据中最有意义的字和最没有意义的字互换，依此类推。仅大小大于等于 16 位的数据项可使用此选项。仅大小为 32 位的数据项才可使用此设置。

- *视为* 属性为内部标记定义了标记的数据类型。对于映射的标记而言，它定义了 *Crimson* 应如何解释操纵的数据。标记被映射时，属性将被设为合理的默认值，但是可以更改。请注意，对于映射的标记而言，视为属性并不能最终决定标记的实际数据类型，因为还可能会使用缩放属性对数据进行进一步转换。根据通信数据的确切数据类型，下列选项可能可用：

视为	描述
有符号整数	数据会被视为 32 位有符号的值，较小的数据值会经符号扩展。例如，一个 16 位的值 0x8000 会被转换为 0xFFFF8000。
无符号整数	数据会被视为 32 位有符号的值，较小的数据值会经零扩展。例如，一个 32 位的值 0x8000 会被转换为 0x00008000。仅大小小于 32 位的数据项才可使用此设置。
浮点	数据会被视为 32 位单精度浮点值。仅大小为 32 位的数据项才可使用此设置。

- *访问权限* 属性用于为映射的标记定义应允许何种通信操作。内部标记始终被设为读写访问权限，表达式标记则始终都是只读。
- *存储* 属性用于指示在目标设备的动力循环之间是否应保留标记。通常为内部标记使用此属性，但映射的只写标记也可能会保留其值。

## 数据缩放

- *缩放* 属性用于为映射的标记定义要在数据上进行的最终缩放步骤。无论 **Crimson** 如何看待操纵的通信数据，数据均可被缩放至整数或浮点。例如，整数值可以缩放至浮点值，这种情况下，**Crimson** 会将标记视为浮点。同样，浮点值也可以转换至整数，甚至无需更改其大小。
- *数据自* 和 *数据至* 属性定义了读取时发生的转换的域和写入时发生的转换的范围。值必须匹配视为属性指定的数据类型，只有这样，才能只为被视为浮点的数据的字段输入非整数值。读取时，超过这些限制的值仍会被接受，并被缩放至超过显示限制的相应值。
- *显示自* 和 *显示至* 属性定义了读取时发生的转换的范围和写入时发生的转换的域。值必须匹配缩放至属性指定的数据类型，只有这样，才能只为被缩放至浮点的数据的字段输入非整数值。读取时，超过这些限制的值仍会被接受，并被缩放至超过数据限制的相应值。

## 数据模拟

- *模拟为* 属性定义了要在页面编辑器里操作时要为标记使用的假定值。输入合理的值会更好展示页面可能的外观。如果全局禁用了通信，那么此值还会被目标设备用作标记的默认值。

## 数据操作

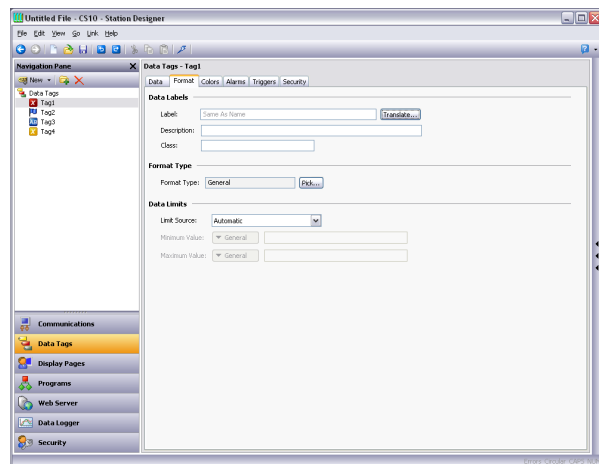
- *写入时* 属性定义了标记被更改时要调用的操作。写入发生时和操作执行时，系统变量 *数据* 将持有新数据值。本章稍后会介绍写入时属性的作用。

## 数据设定值

- *使用设定值* 属性用于为此标记启用或禁用设定值。
- *设定值* 属性定义了此标记名义上应跟随的表达式或其它标记。此设定值然后可用于在警报或基元里实现各种功能。

## 格式属性

数字标记在其格式选项卡里拥有下列属性：



## 数据标签

- 标签属性已在上文标记属性小节下描述过。
- 描述属性已在上文标记属性小节下描述过。
- 类属性被保留，以供将来扩展。

## 格式类型

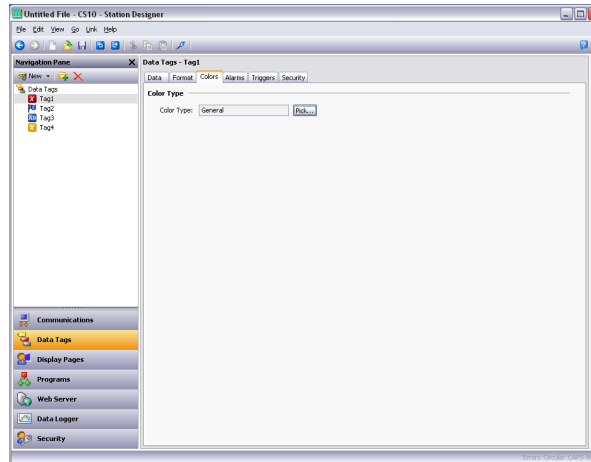
- 格式类型属性为此标记选择了格式。下章会详细描述各种格式类型和根据选定的格式类型可能出现的其它属性。

## 数据限制

- 限制源属性定义了如何定义标记的数据输入限制。默认设置为自动，会使数据选项卡里指定的显示范围被用作首要源，格式对象则用作回退。如果两种源均不能定义一个范围，则会使用标记的数据类型支持的最大范围。设置为根据格式可用于强制使用格式对象，而设置为用户定义则可用于允许手动输入限制。
- 最小值和最大值属性用于限制源设为用户定义时手动定义数据输入限制。

## 颜色属性

数字标记在其颜色选项卡里拥有下列属性：

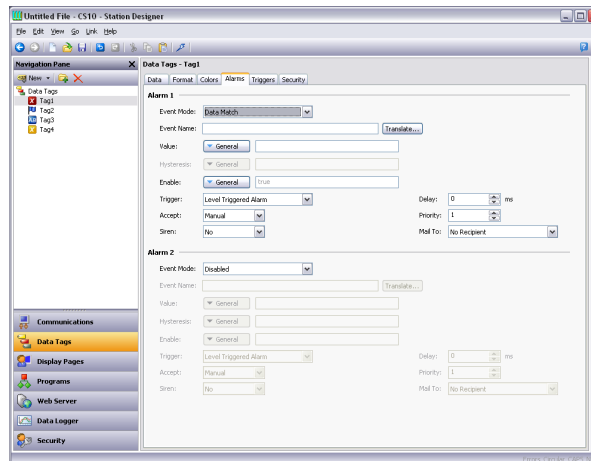


### 颜色类型

- **颜色类型** 属性定义了此标记的着色。下章会详细描述各种着色类型和根据选定的选项可能出现的其它属性。

## 警报属性

数字标记在其警报选项卡里拥有下列属性：



对于每个警报而言

- **事件模式** 属性用于指示用于决定是否应激活警报的逻辑。下表列出了可用的模式。

模式	下列条件出现时会激活警报：
数据匹配	标记的值等于警报的 <i>值</i> 。
数据不匹配	标记的值不等于警报的 <i>值</i> 。
绝对高	标记的值高于警报的 <i>值</i> 。

模式	下列条件出现时会激活警报：
绝对低	标记的值低于警报的 <i>值</i> 。
值升高	标记的值升高，升高量等于警报的 <i>值</i> 。
值降低	标记的值降低，降低量等于警报的 <i>值</i> 。
值更改	标记的值更改，更改量等于警报的 <i>值</i> 。

只有定义了设定值之后，下列模式才可用：

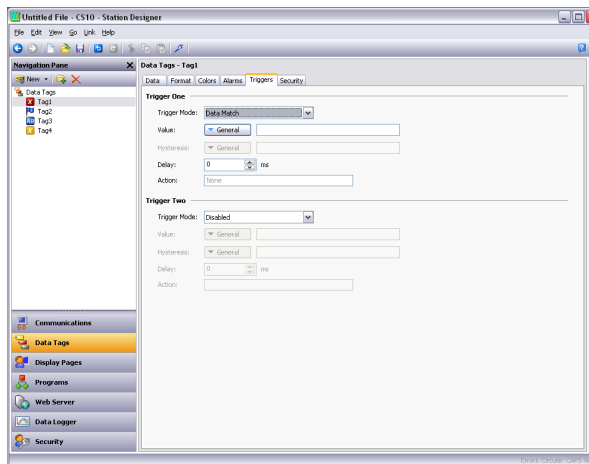
模式	下列条件出现时会激活警报：
偏差高	标记的值高于标记的 <i>设定值</i> ，高出量等于或大于警报的 <i>值</i> 。
偏差低	标记的值低于标记的 <i>设定值</i> ，低出量等于或大于警报的 <i>值</i> 。
带外	标记移出一个带，带的宽度等于警报的 <i>值</i> 的两倍且中心位于标记的 <i>设定值</i> 。
带内	标记移入一个带，带的宽度等于警报的 <i>值</i> 的两倍且中心位于标记的 <i>设定值</i> 。

- *事件名称* 属性定义了将显示在警报查看器里或引用此事件的事件日志里的名称。
- *值* 属性定义了警报将被激活的绝对值、从设定值的偏差或警报最后一次激活之后必须发生的值的更改。准确解释依赖于上文描述的事件模式。
- *滞后* 属性用于流程接近警报条件时阻止警报在开启状态和关闭状态之间摆动。例如，对于一个绝对高警报而言，标记高于警报的值时警报就会激活，但是只有在标记低于警报的值，且低出量大于或等于警报的滞后时，警报才会停用。请记住，警报一旦激活，属性就会始终用于维持该警报，不要在激活发生时进行修改。
- *启用* 属性定义了启用或禁用警报的表达式。非零值或空值会启用警报，零值则会禁用警报。
- *触发器* 属性用于指示警报应是边缘触发还是层次触发。在前一种情况下，事件模式指定的条件首次出现时，就会触发警报。在后一种情况下，只要条件存在，警报就会一直处于活跃状态下。此属性还可用于指示此警报仅应用作事件。这种情况下，警报会被边缘触发，但是不会导致出现警报条件。事件会被记录在内部内存或 CompactFlash 卡里（可选）。
- *延迟* 属性用于指示警报条件必须存在多久之后警报才会激活。如果是边缘触发的警报或事件，此属性还指定了后续重新激活发出进一步警报的信号之前警报条件必须不再存在的时间量。例如，如果一个警报设为速度开关指示一个马达未运行时激活，则此属性可用于在警报激活之前为马达提供加速时间。
- *接受* 属性用于指示不再显示警报之前是否应要求用户明确接受警报。边缘触发的警报必须始终手动接受。

- *优先级* 属性用于控制 *Crimson* 的警报查看器显示警报的顺序。优先级字段数字值越低，警报就会显示在越接近顶端的地方。
- *报警器* 属性用于指示激活此警报时是否也应激活目标设备的报警器。报警器活跃时，面板的显示也会闪烁，从而更好地吸引用户注意警报条件。
- *邮寄至* 属性指定了此警报激活时应向其发送消息的电子邮件地址簿条目。请参阅使用服务章节，了解电子邮件配置信息。

## 触发器属性

数字标记在其触发器选项卡里拥有下列属性：



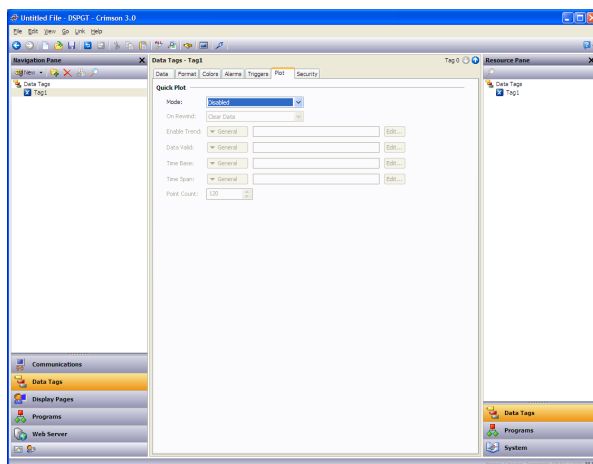
对于每个触发器而言

- *触发器模式* 属性与警报选项卡描述的相同。
- *值* 和 *滞后* 属性与警报选项卡描述的相同。
- *延迟* 属性与警报选项卡描述的相同。
- *操作* 属性用于指示触发器激活时应执行何种操作。请参阅编写操作章节，了解用于定义各种可用操作的语法。

## 绘图属性

快速绘图是一项新增至数字标记的功能，通过此功能可轻松以图形方式追踪标记值。启用并配置后，可从资源窗格内的“核心基元”类别将标记绘图添加至显示页面。单击并将“快速绘图”基元拖至所需的显示页面，并视需要调整大小。

绘图选项卡里拥有下列属性：



- *模式* 属性用于设置如何记录数据。*连续* 模式会在循环缓冲区内进行记录，同时放弃旧值。*连续* 模式是最常用的模式。*单次* 模式会在“启用”变为真值时开始记录，在缓冲区满或“启用”变为假值时停止记录。用于在槽内查找位置的时间值会与绘图开始的时间相关。*绝对单次* 模式与此相似，但所有时间值均从零开始之外。
- *后退时* 指定了时间倒退时有何操作。由于时间基准可能是变量，所有时间有可能倒退。这些选项会清理我们倒退至的时间之后的数据，或者将缓冲区内的所有数据移位，这样会保留旧数据，但会将其在时间内往后移位。
- *启用* 会开始和停止趋势。
- *数据有效* 可在不停止趋势的情况下记录数据内的间隙，进而在其重新启动时删除所有数据。
- 默认设置下，*时间基准* 为系统时间，用于定义时间基准。针对特殊应用（如从外部控制器记录斜变-恒温性能）而言，可使用外部时间基准。
- *时间跨度* 是在缓冲区内要记录的时间基准刻度线的数量。请注意，这通常大于点数，并与该变量共同定义了每个槽会占有多少刻度线。
- *点数* 是要存储在缓冲区内的点的数量，因为快速绘图专为不同时间的标记变更的基本显示而设计，所有这通常是较小的值，亦即小于显示内像素的数量。

## 安全属性

请参阅使用安全章节，了解安全详情。

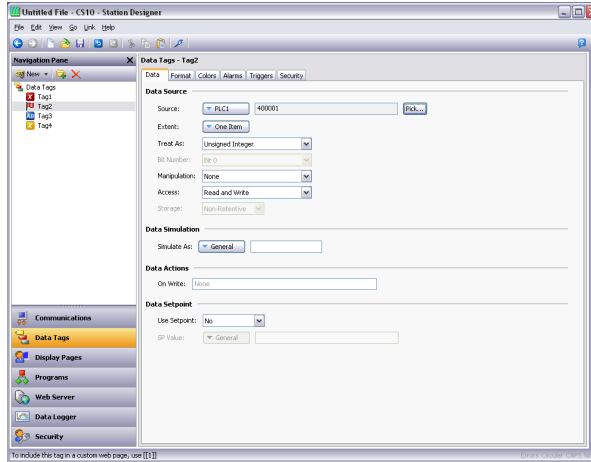


## 标志标记

标志标记表示了一个或多个开启或关闭值，无论基础数据类型如何，标志标记均被认为拥有整数内部数据类型。映射的标志标记允许在原始数据和将由 Crimson 使用的数据之间进行简单转换。

### 数据属性

标志标记在其数据选项卡里拥有下列属性：



### 数据源

- **源** 属性定义了标记从何处获取其数据。默认设置会使用内部标记，而下拉列表则可用于选择常规表达式、其它数据标记或远程设备的项。
- **范围** 属性用于在单一元素标记或数组之间进行选择。如果选择数组，则必须输入所需的元素数目。不允许为源是表达式的标记选择数组。对于映射的标记而言，要从远程设备读取的寄存器的精确数目依赖于标记映射至的寄存器的类型和视为设置。
- **视为** 属性用于为映射的标记定义如何从和向原始通信数据派生开启或关闭值。根据基础数据类型，下列设置可能可用：

视为	结果
无符号整数	如果数据非零，则标记为真，如果数据是零，则标记为假。真值会被作为整数值 1 写入，而假值则会被作为 0 写入。对于映射的数组而言，每个数组元素均对应于一个单一通信数据元素。任何大小大于等于 8 位的通信数据均可用此设置。
浮点	如果值非零，则标记为真，如果值是零，则标记为假。真值会被作为 32 位浮点值 1 写入，而假值则会被作为 0 写入。对于映射的数组而言，每个数组元素均对应于一个单一通信数据元素。大小等于 32 位的通信数据可用此设置。

视为	结果
位数组 Little Endian	从数据中提取一个单一位。对于单一元素而言，位数字字段选择了位，最没有意义的是位 0。对于数组而言，每个元素均是一个单一位，这样位就可以有效地在数据项里进行打包。数组的第一个元素是最没有意义的位，第二个元素是第二最没有意义的位，依次类推。因此，一个 PLC 里的映射至字节数据类型的 8 个元素的数组将从一个单一寄存器读取全部 8 个位。
位数组 Big Endian	除了位被反转之外，位数字字段位 0 读取最有意义的位，数组的第一个元素则作为最有意义的位的源，依此类推，其余与上面的相同。

- *位数字* 属性从多位数据项为映射的非数组标记提取一个单一位。此属性不用于其它配置。
- *操纵* 属性定义了读取数据时视为逻辑被执行之后或写入数据时视为逻辑被执行之前要应用至标记状态的转换。唯一可用的选项是插入标记的状态。对于单一位值而言，没有太多其它可做的事情！
- *访问权限* 属性用于为映射的标记定义应允许何种通信操作。内部标记始终被设为读写访问权限，表达式标记则始终都是只读。
- *存储* 属性用于指示在目标设备的动力循环之间是否应保留标记。通常为内部标记使用此属性，但映射的只写标记也可能会保留其值。

#### 数据模拟

- *模拟为* 属性定义了页面编辑器里操作时要为标记使用的假定值。输入合理的值会更好展示页面可能的外观。如果全局禁用了通信，那么此值还会被目标设备用作标记的默认值。

## 数据操作

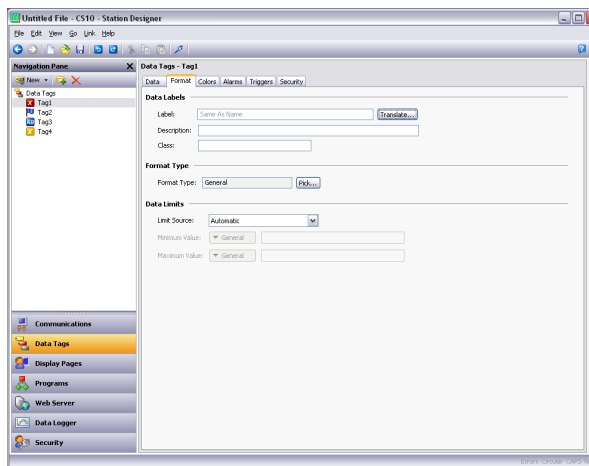
- *写入时* 属性定义了标记被更改时要调用的操作。写入发生时和操作执行时，系统变量 *数据* 将持有新数据值。本章稍后会介绍写入时属性的作用。

## 数据设定值

- *使用设定值* 属性用于为此标记启用或禁用设定值。
- *设定值* 属性定义了此标记名义上应跟随的表达式或其它标记。此设定值然后可用于在警报或基元里实现各种功能。

## 格式属性

标志标记在其格式选项卡里拥有下列属性：



## 数据标签

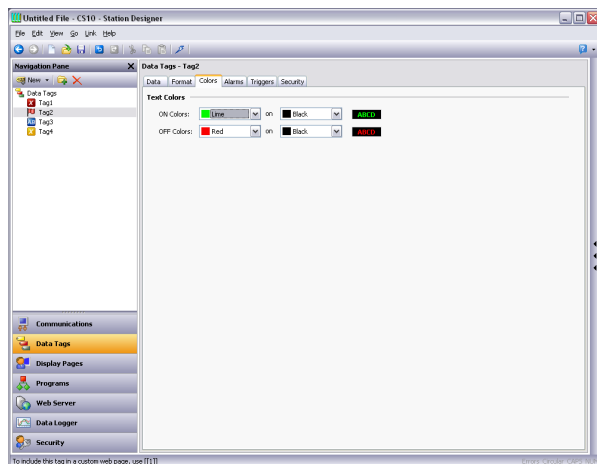
- *标签* 属性已在上文标记属性小节下描述过。
- *描述* 属性已在标记属性小节下描述过。
- 类属性被保留，以供将来扩展。

## 数据格式

- *开启状态* 和 *关闭状态* 属性是永久为标志标记选定的双状态格式对象的一部分。它们定义了用于显示标记位于相应状态下的文本。不支持其它格式类型。

## 颜色属性

标志标记在其颜色选项卡里拥有下列属性：

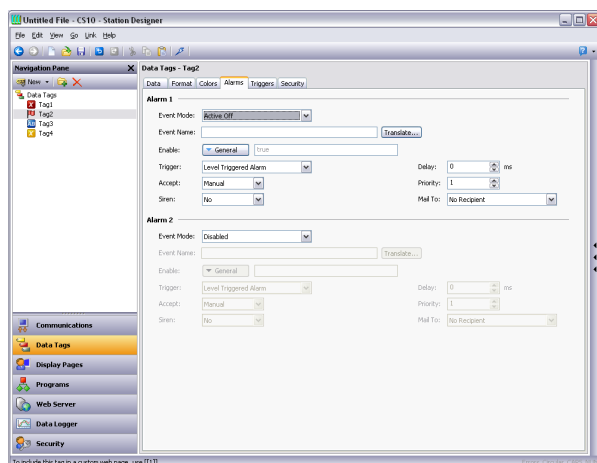


## 文本颜色

- 开启颜色 和 关闭颜色 属性是始终为标志标记使用的双状态着色的一部分。它们定义了可选的用于表示标记位于每种状态下的前景和背景颜色对。不支持其它着色。

## 警报属性

标志标记在其警报选项卡里拥有下列属性：



对于每个警报而言

- 事件模式 属性用于指示用于决定是否应激活警报的逻辑。下表列出了可用的模式。

模式	下列条件出现时会激活警报：
活跃开启	标记为真
活跃关闭	标记为假。

模式	下列条件出现时会激活警报：
状态更改	标记发生更改。

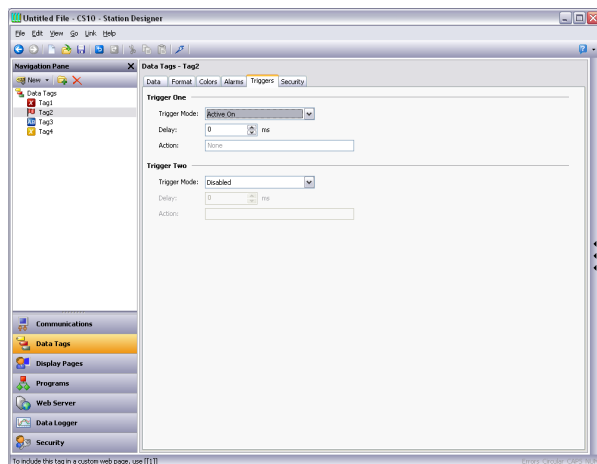
只有定义了设定值之后，下列模式才可用：

模式	下列条件出现时会激活警报：
不等于设定值	标记不等于其设定值。
设定值开启时关闭	标志未响应开启设定值。
设定值关闭时开启	标志未响应关闭设定值。
等于设定值	标记等于其设定值。

- **事件名称** 属性定义了将酌情显示在警报查看器里或事件日志里的名称。**Crimson** 会根据标记的标签和选定的事件模式建议默认名称。
- **启用** 属性定义了启用或禁用警报的表达式。非零值或空值会启用警报，零值则会禁用警报。
- **触发器** 属性用于指示警报应是边缘触发还是层次触发。在前一种情况下，事件模式指定的条件首次出现时，就会触发警报。在后一种情况下，只要条件存在，警报就会一直处于活跃状态下。此属性还可用于指示此警报仅应用作事件。这种情况下，警报会被边缘触发，但是不会导致出现警报条件。事件会被记录在内部内存或 **CompactFlash** 卡里（可选）。
- **延迟** 属性用于指示警报条件必须存在多久之后警报才会激活。如果是边缘触发的警报或事件，此属性还指定了后续重新激活发出进一步警报的信号之前警报条件必须不再存在的时间量。例如，如果一个警报设为速度开关指示一个马达未运行时激活，则此属性可用于在警报激活之前为马达提供加速时间。
- **接受** 属性用于指示不再显示警报之前是否应要求用户明确接受警报。边缘触发的警报必须始终手动接受。
- **优先级** 属性用于控制 **Crimson** 的警报查看器显示警报的顺序。优先级字段数字值越低，警报就会显示在越接近顶端的地方。
- **报警器** 属性用于指示激活此警报时是否也应激活目标设备的报警器。报警器活跃时，面板的显示也会闪烁，从而更好地吸引用户注意警报条件。
- **邮寄至** 属性指定了此警报激活时应向其发送消息的电子邮件地址簿条目。请参阅使用服务章节，了解电子邮件配置信息。

## 触发器属性

标志标记在其触发器选项卡里拥有下列属性：



对于每个触发器而言

- *触发器模式* 属性与警报选项卡描述的相同。
- *滞后* 属性与警报选项卡描述的相同。
- *操作* 属性用于指示触发器激活时应执行何种操作。请参阅编写操作章节，了解用于定义各种可用操作的语法。

## 安全属性

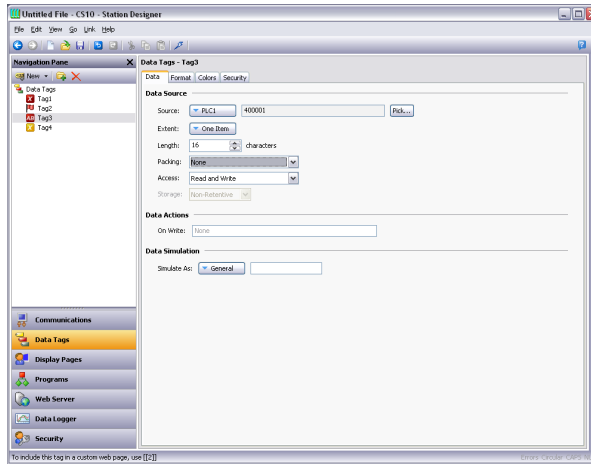
请参阅使用安全章节，了解安全详情。

## 字符串标记

字符串标记表示了由 Unicode 字符构成的一个或多个字符串。虽然 Crimson 3 完全在 Unicode 环境下操作，但也可以从 8 位源读写字符串。映射的字符串标记支持多种编码，允许从一个寄存器提取一个或多个字符。

## 数据属性

字符串标记在其数据选项卡里拥有下列属性：



## 数据源

- **源** 属性定义了标记从何处获取其数据。默认设置会使用内部标记，而下拉列表则可用于选择常规表达式、其它数据标记或远程设备的项。
- **范围** 属性用于在单一元素标记或数组之间进行选择。如果选择数组，则必须输入所需的元素数目。不允许为源是表达式的标记选择数组。对于映射的标记而言，要从远程设备读取的寄存器的精确数目依赖于标记映射至的寄存器的类型、长度和打包设置。
- **长度** 属性定义了字符串的长度。无需为非保持性内部字符串定义长度，因为它们可以存储任何合理长度的字符串。
- **打包** 属性用于为映射的标记定义如何从和向原始通信数据派生 Unicode 字符串值。根据基础数据类型，下列设置可能可用：

打包	结果
无	每个通信数据项均用于为字符串提供单个字符的源。8 位值会被视为 ASCII，而 16 位值或更大的值则会被视为 UNICODE。
ASCII Big Endian	数据项里的每个 8 位单元均用于为一个单一 ASCII 字符提供源，为第一个字符使用最有意义的 8 个位。仅大小大于等于 16 位的数据项才可使用此设置。
ASCII Little Endian	数据项里的每个 8 位单元均用于为一个单一 ASCII 字符提供源，为第一个字符使用最没有意义的 8 个位。仅大小大于等于 16 位的数据项才可使用此设置。
Unicode Big Endian	数据项里的每个 16 位单元均用于为一个单一 Unicode 字符提供源，为第一个字符使用最有意义的 16 个位。仅大小等于 32 位的数据项才可使用此设置。
Unicode Little Endian	数据项里的每个 16 位单元均用于为一个单一 Unicode 字符提供源，为第一个字符使用最没有意义的 16 个位。仅大小等于 32 位的数据项才可使用此设置。

打包	结果
十六进制字符串 Little Endian	数据项里的每个 4 位单元均用于在 '0'-'9' 和 'A'-'F' 的范围内为一个单一十六进制字符串提供源，首先使用最没有意义的 4 个位。不支持向使用此种打包方法的字符串进行写入。
十六进制字符串 Big Endian	数据项里的每个 4 位单元均用于在 '0'-'9' 和 'A'-'F' 的范围内为一个单一十六进制字符串提供源，首先使用最有意义的 4 个位。不支持向使用此种打包方法的字符串进行写入。

- *访问权限* 属性用于为映射的标记定义应允许何种通信操作。内部标记始终被设为读写访问权限，表达式标记则始终都是只读。
- *存储* 属性用于指示在目标设备的动力循环之间是否应保留标记。通常为内部标记使用此属性，但映射的只写标记也可能会保留其值。

#### 数据模拟

- *模拟为* 属性定义了页面编辑器里操作时要为标记使用的假定值。输入合理的值会更好展示页面可能的外观。如果全局禁用了通信，那么此值还会被目标设备用作标记的默认值。

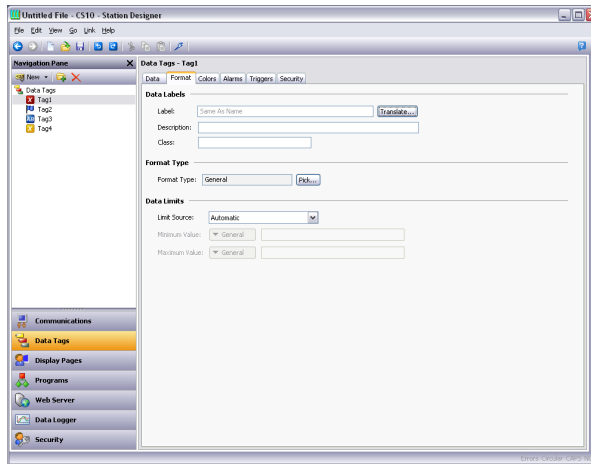
#### 数据操作

- *写入时* 属性定义了标记被更改时要调用的操作。写入发生时和操作执行时，系统变量 *数据* 将持有新数据值。本章稍后会介绍写入时属性的作用。



## 格式属性

字符串标记在其格式选项卡里拥有下列属性：

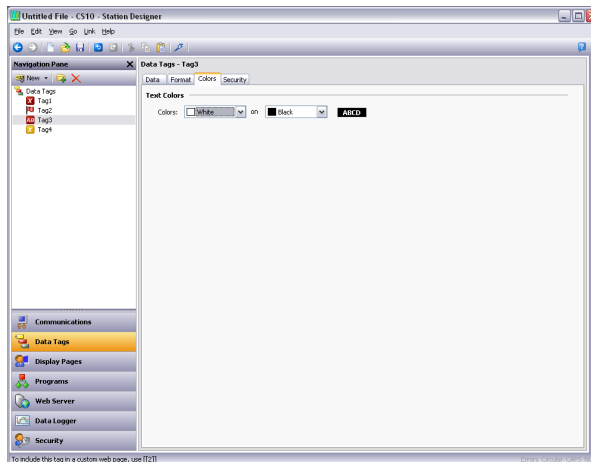


## 数据标签

- 标签属性已在上文标记属性小节下描述过。
- 描述属性已在标记属性小节下描述过。
- 类属性被保留，以供将来扩展。

## 颜色属性

字符串标记在其颜色选项卡里拥有下列属性：



## 文本颜色

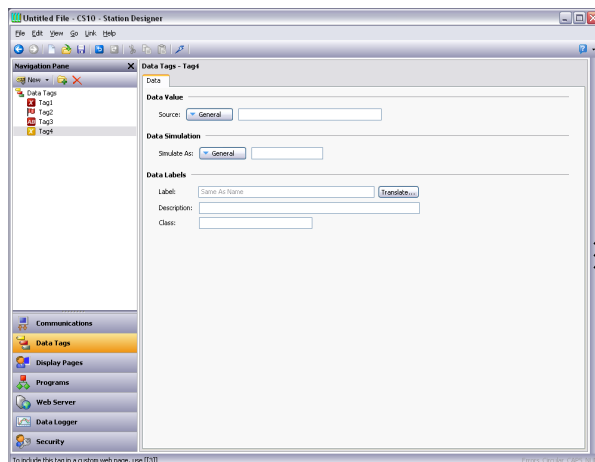
- 颜色属性是始终为字符串标记使用的固定着色的一部分。它们定义了可选的用于表示标记的前景和背景颜色对。不支持其它着色。

## 安全属性

请参阅使用安全章节，了解安全详情。

## 基本标记

基本标记用于表示常量或表达式：



### 数据值

- *数据值* 属性定义了标记的值，必须是表达式。标记本身将采用使用的表达式的数据类型。

### 数据模拟

- *模拟为* 属性定义了编辑页面时要为标记使用的默认值。输入合理的值会更好展示页面可能的外观。如果全局禁用了通信，那么此值还会被目标设备用作标记的默认值。

### 数据标签

- *标签* 属性已在上文标记属性小节下描述过。
- *描述* 属性已在标记属性小节下描述过。
- 类属性被保留，以供将来扩展。

## 标记数据流

正如您将注意到的那样，数字标记拥有发生在通信数据和 **Crimson** 实际使用的值之间的多种转换。可以配置这些转换，从而以任何希望使用的方式处理任何类型的数据，但是，值得进一步了解一下它们为数字标记进行操作的确切方式。

## 数字标记读取流程

从设备读取数据时，会发生下列步骤：

- 要求通信驱动程序基于为标记源定义的地址设置读取一个值。根据地址类型，驱动程序可能会合并多个寄存器，从而创建数据值。例如，读取一个单一字作为长型值会使驱动程序使用其对设备的字顺序的了解读取并合并两个寄存器。
- 然后根据相关标记的操纵属性对通信数据进行修改。这些处理会对数据进行位级或字节级更改，目的通常是为了说明应用程序不兼容性或数据不是通信驱动程序正常期待处理的形式其它情况。
- 然后结合标记的视为属性对操纵的数据进行解释，数据被酌情视为 32 位整数或 32 位单精度浮点值。小于 32 位的数据项会根据配置被零或符号扩展。如果没有定义缩放，则此步骤的结果定义了标记的最终值和数据类型。
- 如果定义了缩放，则会根据为标记定义的域和范围对解释的数据进行缩放。缩放结果的类型可能会与解释的数据的类型有所不同，因此，浮点值可能会缩放成整数，整数也可能缩放成浮点值。如果定义了缩放，则此步骤的结果定义了标记的最终值和数据类型。

## 数字标记写入流程

向设备写入数据时，会发生下列步骤：

- 如果定义了缩放，则会保留域和范围，将数据转回数据类型由视为属性定义的未缩放的值。
- 如果未缩放的数据比通信数据大，则会移除高顺序位，生成适用于下一步骤的裁剪版本的数据。
- 然后会根据操纵属性对裁剪的数据进行修改，反转上面应用的转换，生成通信数据。
- 然后通信驱动程序会根据地址类型将通信数据写入目标设备里的一个或多个寄存器。

## 使用写入时

标记的写入时属性含有对该标记进行更改时要执行的操作。操作执行时，叫做 Data 的系统属性会被设为新值，允许对新数据进行检查。此功能有三种典型用处：

- 常规读写标记的写入时属性可定义为允许按需执行操作。例如，数据库可能需要将标记的值存储为两种格式，一种是原始标记格式，另一种是转换的版本。可使用多种方法这样做，其中之一就是使用写入时属性，捕获写入，然后运行程序来计算和存储转换的版本。
- 可通过定义写入时属性将只读标记设为可写。虽然这看起来很奇怪，但是假设一个 PID 循环有一个指示其当前输出功率的只读属性，还有一个定义手动输出功率的读写属性。可以定义显示字段，从而允许在手动模式下输

入输出功率数据，并使用写入属性捕获数据，进而将值写入手动输出功率。

- 通过定义一个执行正向转换的表达式标记和一个执行反转的写入时操作，即可实现复杂转换。例如，一个标记可被设为 `Sqrt([40001])`，从而求出一个 Modbus PLC 里的一个值的平方根。因为这是一个表达式标记，所以其定义是只读，但是，可以通过定义一个等于 `[40001] = Data*Data` 的写入时，从而反转平方根计算，即可允许写入。

## 数组标记属性

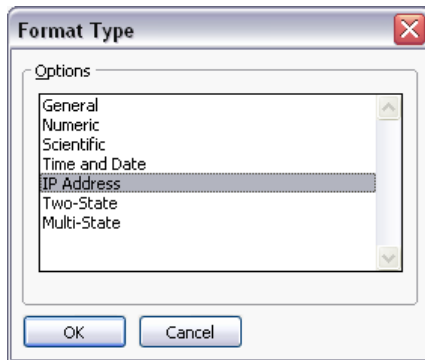
[TBA]

# 使用格式

可为数字标记选择多种数据格式中的一种，而标志标记和字符串标记的格式则只可分别固定为双状态和常规。每种格式类型都会取一个数据值，然后从或向文本字符串进行转换。

## 格式类型

支持下列格式：



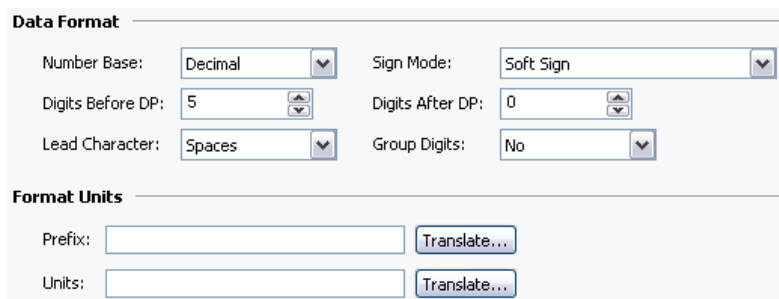
- *常规* 格式可为值设置简单格式，可将数字值转换为有符号的十进制值，还可在不进一步处理字符串的情况下将其传递。常规格式没有配置属性，是字符串标记的默认格式，还可隐式用于基本标记。
- *数字* 格式会取一个浮点值或整数值，然后通过使用指定数基并选择小数点前后所需的数字数目将其转换为字符串。它还可以向值添加前缀字符串和单位字符串，处理有符号值或无符号值。
- *科学* 格式会取一个浮点值或整数值，然后通过选择小数点后面所需的数字数目将其转换为指数格式。它还可以向值添加前缀字符串和单位字符串。
- *时间和日期* 格式会取一个整数值，并将其视为 1997 年 1 月 1 日之后经过的秒数。它可将结果显示为日期值、时间值或两者。支持选择日期格式和时间格式，从而允许使用各种国际标准。
- *IP 地址* 格式会取一个整数值，并将其显示为句点分隔的四个十进制字节，无需进一步配置，即可将 32 位数字显示为 IP 地址。
- *双状态* 格式会取一个数字值，并将其显示为基于值是否为零的两种字符串的一种。这是标志标记的永久定义格式。
- *多状态* 格式会取一个数字值，并将其与含有值和字符串的表进行比较。要么会显示与匹配数据值关联的字符串，要么格式可配置为显示值不高于上一个字符串的关联值的上一个字符串。

## 常规格式

常规格式没有属性。

## 数字格式

数字格式拥有下列属性：



Data Format	
Number Base:	Decimal
Sign Mode:	Soft Sign
Digits Before DP:	5
Digits After DP:	0
Lead Character:	Spaces
Group Digits:	No

Format Units	
Prefix:	<input type="text"/> <span>Translate...</span>
Units:	<input type="text"/> <span>Translate...</span>

### 数据格式

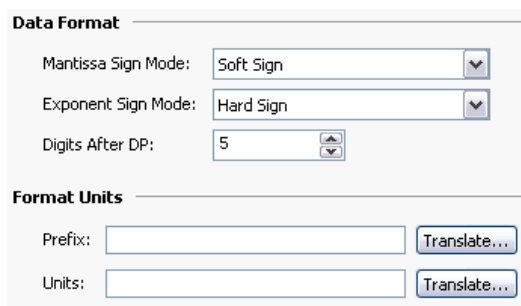
- **数基** 属性定义了显示的值的基数。密码设置以十进制进行操作，但是会使用星号将数字遮挡起来。使用非十进制模式时，许多其它选项将被禁用。
- **符号模式** 属性定义了如何处理数据以及如何显示符号。值设为无符号会将值作为 32 位无符号数字进行显示，从而允许显示和输入这样的值，即使 Crimson 无法在不能纳入 32 位有符号表示的值上进行任何数学计算。值设为软符号会为负数显示一个前导负号，为正数显示一个空格。值设为硬符号则会为正数显示一个前导正号，而不是空格。
- **小数点前面的数字** 属性定义了要显示在小数点前面的数字的数目。对于没有小数点的值而言，这是要显示的数字的总数目，因此控制了数据字段的大小。
- **小数点后面的数字** 属性定义了要显示在小数点前面的数字的数目。对于整数值而言，小数点会被插入整数表示，这样的话，如果此属性设为二，则 1234 就会被显示并输入为 12.34。零值会取消显示小数点。
- **前导字符** 属性定义了如何为带有前导零的值设置格式。可保留、用空格替换或完全移除前导零。移除前导零有时可能会导致显示的值发生令人不快的抖动，因为它们更改了数字的数目，如果值位于字段中间，则更为如此。
- **组数字** 属性可以为十进制数每隔三个数字插入一个逗号分隔符，亦可对其它数基进行类似操作。

### 格式单位

- *前缀* 属性定义了要显示在数字值前面的字符串。
- *单位* 属性定义了要显示在数字值后面的字符串。

## 科学格式

科学格式拥有下列属性：



The screenshot shows a dialog box titled "Data Format" with two main sections: "Data Format" and "Format Units".

**Data Format**

- Mantissa Sign Mode: Soft Sign (dropdown menu)
- Exponent Sign Mode: Hard Sign (dropdown menu)
- Digits After DP: 5 (spin box)

**Format Units**

- Prefix: [text input] [Translate... button]
- Units: [text input] [Translate... button]

### 数据格式

- *尾数符号模式* 属性定义了如何在尾数上显示符号。值设为软符号会为负数显示一个前导负号，为正数显示一个空格。值设为硬符号则会为正数显示一个前导正号，而不是空格。
- *指数符号模式* 属性定义了如何在指数上显示符号。值设为软符号会为负值显示一个前导负号，不会为正值显示任何内容。值设为硬符号则会为正值显示一个前导正号。
- *小数点后面的数字* 属性定义了要显示在小数点后面的数字的数目。按照定义，科学格式下小数点前面始终有一个数字。零值会取消显示小数点。

### 格式单位

- *前缀* 属性定义了要显示在数字值前面的字符串。
- *单位* 属性定义了要显示在数字值后面的字符串。

## 时间和日期格式

时间和日期格式拥有下列属性：

**Format Mode**

Field Contents: Time Then Date

**Time Format**

Time Format: 12 Hour (Civil)

Show Seconds: No

AM Suffix: AM Translate...

PM Suffix: PM Translate...

**Date Format**

Date Format: Locale Default

Show Year: As 2 Digits

Show Month: As Digits

### 格式模式

- *格式模式* 属性用于指示字段是否应显示时间、日期或两者。最后一种情况下，此属性还指示了两者的显示顺序。提供了多种选项，从而允许时间值被视为经过的时间量，而不是配以日期的时间。例如，在经过模式下，25.5 小时的值会显示为 25:30。在常规时间模式下，该值会显示为 00:30，因为系统会将其假定为 1997 年 1 月 2 日凌晨的时间。

### 时间格式

- *时间格式* 属性用于指示是否应使用 12 小时制（民用时间）或 24 小时制（军用时间）。配合其它属性，将此设置留为默认区域设置，会允许 Crimson 根据操作员面板内选择的语言选取合适格式。
- *AM 后缀* 和 *PM 后缀* 属性用于在 12 小时制模式下指示要根据上午和下午酌情附加在时间字段的文本。如果将属性留为未定义，Crimson 会使用默认设置。
- *显示秒* 属性用于指示时间字段是否应包含秒或仅应包含小时和分钟。

### 日期格式

- *日期格式* 属性用于指示各种日期元素（即日、月、年）的显示顺序。
- *显示年份* 属性用于指示日期字段是否应包含年份，如果应包含的话，还用于指示应为年份显示几个数字。
- *显示月份* 属性用于指示是否应将月份显示为数字（即 01 至 12）或显示为名称缩写（即 Jan 至 Dec）。

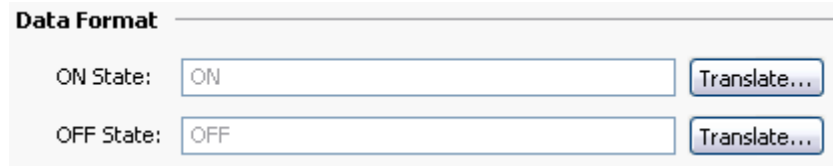


## IP 地址格式

IP 地址格式没有属性

## 双状态格式

双状态格式拥有下列属性：



**Data Format**

ON State:  [Translate...](#)

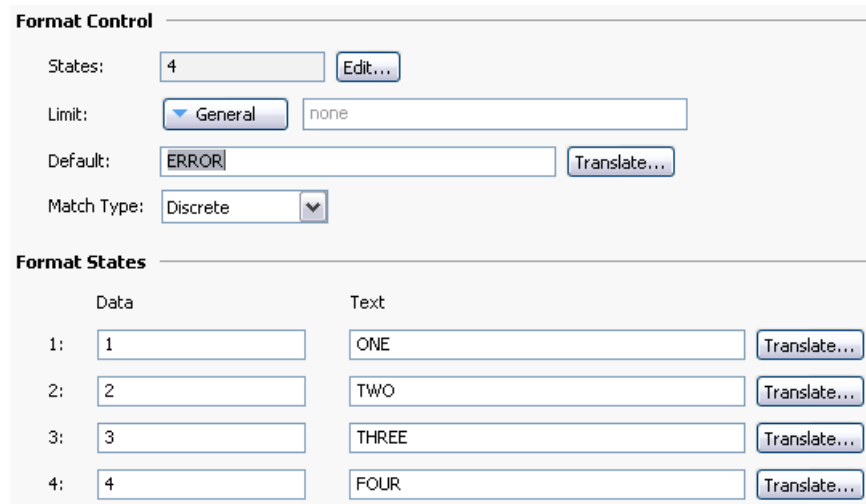
OFF State:  [Translate...](#)

日期格式

- 开启状态 属性定义了值不是零时要显示的文本。
- 关闭状态 属性定义了值是零时要显示的文本。

## 多状态格式

多状态格式拥有下列属性：



**Format Control**

States:  [Edit...](#)

Limit:

Default:  [Translate...](#)

Match Type:

**Format States**

	Data	Text	
1:	<input type="text" value="1"/>	<input type="text" value="ONE"/>	<a href="#">Translate...</a>
2:	<input type="text" value="2"/>	<input type="text" value="TWO"/>	<a href="#">Translate...</a>
3:	<input type="text" value="3"/>	<input type="text" value="THREE"/>	<a href="#">Translate...</a>
4:	<input type="text" value="4"/>	<input type="text" value="FOUR"/>	<a href="#">Translate...</a>

格式控制

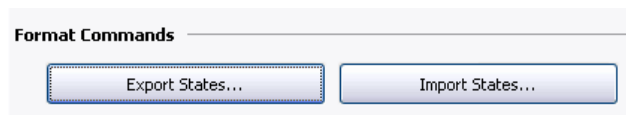
- 状态 属性定义了多状态格式应包含多少个状态，可多达 500 个项目。显示格式的窗口会进行更新，从而显示数据和文本属性的所需数目。
- 限制 属性定义了将数据与此格式进行匹配时会使用多少个状态。此属性可动态调整，而状态的绝对数目则被静态定义。在运行时填充状态字段时，此属性非常有用，因为它允许在数据输入过程中跳过未使用的字段。
- 默认 属性定义了数据无法与定义的状态匹配时要显示的字符串。如果未提供值，则会在括号里显示不匹配状态的数字表示。

- *匹配类型* 属性定义了数据如何与各种状态进行比较。如果选择了离散，那么标记数据必须匹配给定状态的数据值，这样才能使用该状态。如果选择了范围，则 **Crimson** 会假设状态数据值按数字升序排列，如果标记数据小于或等于一个状态的数据值但大于前一个状态的数据值，**Crimson** 就会使用该状态。数据输入过程中，范围格式对象会分配等于单个状态的实际数据值的值。

#### 格式状态

- *数据* 和 *文本* 属性为此格式下的每个状态定义了数据值和显示文本。没有文本字段的状态会被禁用并忽略。

#### 格式命令



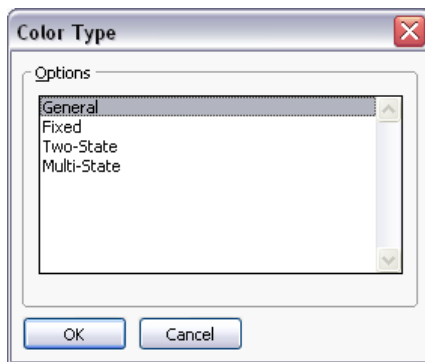
多状态格式对象还提供有允许从或向 Unicode 文本文件导入或导出其各种状态和关联属性的按钮。这些文件然后可以使用 Microsoft Excel 之类的应用程序进行编辑。

# 使用颜色

可为数字标记选择多种所谓的着色中的一种，而标志标记和字符串标记的颜色则只可分别固定为双状态和常规。每种着色都会取一个数据值，然后将其转换为前景和背景颜色对。

## 着色类型

支持下列着色：



- *常规*着色始终返回黑白色。
- *固定*着色始终返回固定颜色对。
- *双状态*着色会取一个数字值，并根据值是否为零选取两种颜色对中的一种。这是标志标记的永久定义着色。
- *多状态*着色会取一个数字值，并将其与含有数据值和颜色对的表进行比较。要么会显示与匹配数据值关联的颜色对，要么选择器可配置为使用其关联值不高于数据的上一个颜色对。

## 常规着色

常规着色没有属性。

## 固定着色

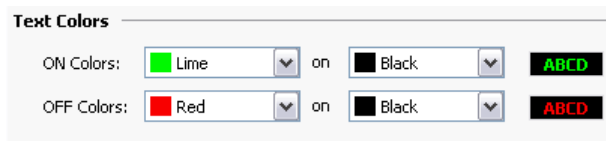
固定着色拥有下列属性：



- *颜色*属性定义了始终要使用的颜色。

## 双状态着色

双状态着色拥有下列属性：



- 开启颜色 属性定义了标记非零时要使用的颜色。
- 关闭颜色 属性定义了标记为零时要使用的颜色。

## 多状态着色

多状态着色拥有下列属性：



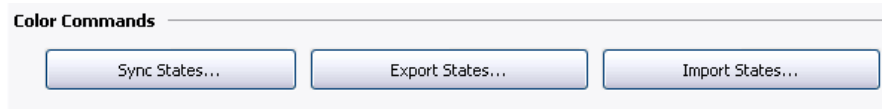
## 格式控制

- 状态 属性定义了多状态选择器应包含多少个状态，可多达 500 个项目。显示选择器的窗口会进行更新，从而显示数据和文本属性的所需数目。
- 默认颜色 属性定义了数据无法与定义的状态匹配时要使用的颜色。
- 匹配类型 属性定义了数据如何与各种状态进行比较。如果选择了离散，那么标记数据必须匹配给定状态的数据值，这样才能使用该状态。如果选择了范围，则 Crimson 会假设状态数据值按数字升序排列，如果标记数据小于或等于一个状态的数据值但大于前一个状态的数据值，Crimson 就会使用该状态。

## 格式状态

- 数据和颜色 属性为每个状态定义了数据值和颜色值。

## 颜色命令



多状态着色对象还提供有允许从或向 Unicode 文本文件导入或导出其各种状态和关联属性的按钮。这些文件然后可以使用 Microsoft Excel 之类的应用程序进行编辑。还提供有一个附加按钮，允许将着色的数据字段与为同一标记配置的多状态格式对象的数据字段进行同步，这样就无需再次输入同一值。

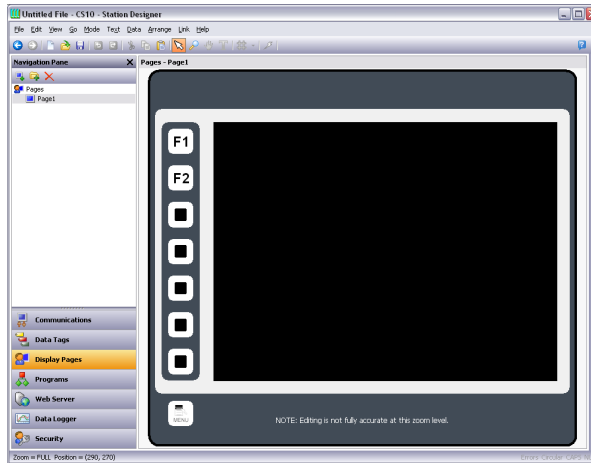


# 创建显示页面

选择导航窗格里的显示页面类别，可以打开全新的 **Crimson** 图形编辑器。使用此编辑器，即可快速有效地创建精美的显示，同时，编辑器还提供了最大限度的灵活性。

## 编辑器基础

图形编辑器的初始状态如下所示：



编辑窗格展示了资源设备，包括按键和显示区域本身。最低缩放级别下，会显示整个面板，即使这意味着为目标设备的显示内的每个像素分配的像素小于 PC 的显示内的一个像素。这种情况下，仍可查看页面并进行大多数编辑操作，但稍微会降低精度，因此，会显示与此相关的警告消息。

## 对页面进行操作

通过导航列表对页面进行操纵极为直观，与 **Crimson** 数据库内的任何其它项的操纵方式类似。尽管如此，还是值得重申一下一个事实——从数据库的导航窗格内选择页面，然后将其拖入目标数据库内的相应类别，即可在数据库之间复制页面。这使通过合并先前使用的页面设计来创建新数据库变得极为简便。

## 更改缩放级别

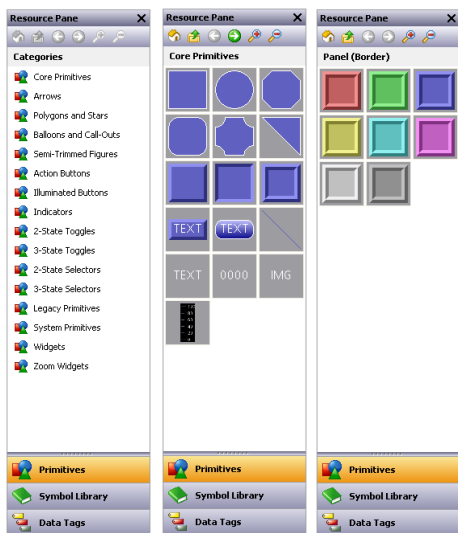
使用鼠标滚轮即可最为方便地进行放大或缩小。如果没有这样的鼠标，则可从工具栏选择放大镜，从而使用编辑器的缩放模式。此模式下，左键单击即可放大，右键单击或按住 **CTRL** 键时左键单击即可缩小。还可使用查看菜单里的缩放命令。

第一步缩放会对全面版视图进行 1:1 显示，将目标设备的显示放在编辑窗口的中央。随后进行的缩放会将数据保留在视图里的鼠标指针下面，从而便于选择希望详细检查的显示区域。

## 资源窗格

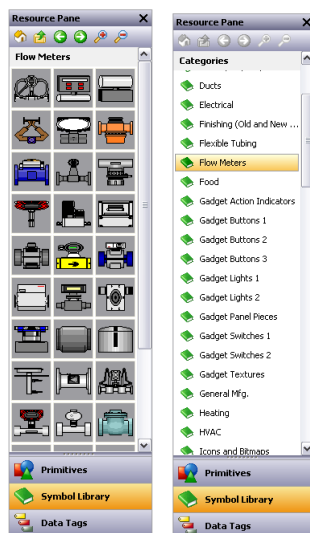
通常使用从资源窗格中拖动的项来生成显示页面。可通过单击窗口右侧带箭头的栏将资源窗格隐藏至窗口边沿，也可选择将资源窗格锁定在某个位置，可以最大化窗口，从而增加可用的工作空间。资源窗格有三个类别：

### 基元



基元类别用于访问用于组建显示页面的关键构建基块，在其各种状态之下显示在左侧。您会注意到顶层含有若干子类别，通过每个子类别均可访问若干基元。单击一个图标会显示一个子类别及其基元。单击一个给定基元会以预定义的颜色显示该基元的版本。工具栏里的图标可用于在子类别间进行移动、向较高层进行移动或更改每行显示的基元的数目。下章会详细描述各种基元。

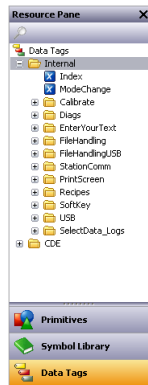
### 符号库



符号库类别的操作方式与基元类别极为相似，可通过该类别访问若干子类别，每个子类别均包含若干预定义的符号。单击一个给定符号会打开该符号的若干预着色版本，尽管此功能并不如在基元类别下常用。请花一些时间浏览一下符号库，因为它包含有成千上万个可能的图像。正确使用符号库可以构建更为精美易用的数据库。



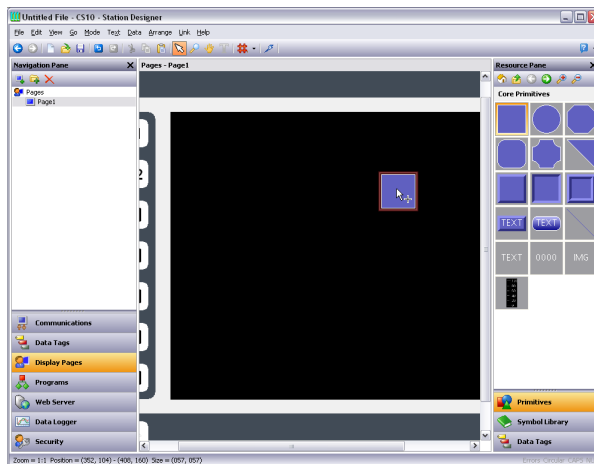
## 数据标记



数据标记类别包含了显示当前数据库内全部数据标记的树状视图，用于直接将标记拖放入显示页面，并用于在配置基元属性时提供对标记的访问。将一个标记拖放入页面会创建一个绑定至该标记的数据框，此数据框拥有按照标记本身定义的属性而设置的全部格式属性。还可按常规方式通过使用 **SHIFT** 键和 **CTRL** 键选择并拖动多个标记。这些功能使向页面添加数据变得极为快速简便。

## 向页面添加项

如上所述，资源窗格里的各项均可拖入编辑器，从而将其添加至显示页面。会为标记和图像创建合适的基元。下例显示了在基元类别里单击核心基元选择内容之后，如何将一个矩形基元拖入页面：



## 对基元进行操作

下面的章节描述了如何对基元进行常见操作。

### 选择基元

要选择一个显示基元，仅需将鼠标指针移至相关基元，然后单击。您会注意到，指针在基元上悬停时，会出现一个蓝色矩形，帮助显示将选定的内容。进行实际选择时，矩形会变成红色，并出现控点，从而允许您按需调整基元的大小。如果您希望选择的基元隐藏在另一个基元下面，请按下 **CTRL** 键进行选择。

要选择多个基元，可在希望选择的基元周围绘制一个选择矩形，或轮流选择每个基元，并按下 **SHIFT** 键来指示您希望每个基元都添加至选择内容。如果选定了多个基元，则红色矩形会环绕所选的全部基元，然后可使用控点调整基元的大小。可保留基元的相对大小和位置，只要 **Crimson** 这样做不违反最小大小要求。

## 使用快速工具栏

快速工具栏是一个浮动在当前选择内容右上方的工具栏：



工具栏最开始会以透明淡出形式显示，鼠标指针向其移动得越近，透明度就会越低。从工具栏移开指针会将其隐藏，重复选择流程或按下鼠标滚轮之前，此工具栏不会再次出现。使用快速工具栏，即可在尽量少移动鼠标的情况下使用一系列常用功能。可使用查看菜单里的命令启用或禁用快速工具栏。

## 在页面间移动基元

可按常规方式在显示页面内拖动基元，也可将基元从一个页面复制到另一个页面。要这样做，只需选择希望要复制的基元，然后将其拖入导航窗格。如果窗格是隐藏的，请悬停在带箭头的栏上，窗格即会滑入视图。悬停在目标页面，即会选择该页面。然后将基元拖回至编辑器，并将其放入新页面。按住 **CTRL** 键会将复制操作更改为移动操作，操作原理与在页面内移动时截然相反。

## 在数据库间移动基元

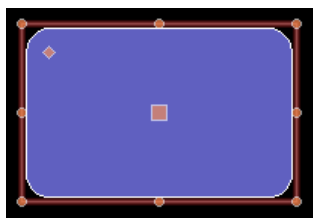
在数据库间拖动基元也同样简单。仅需选择希望复制的项，然后将其拖入另一个含有新数据库的 **Crimson** 运行实例。这可以对整个页面、基元组或单个项进行操作。

## 更改基元大小

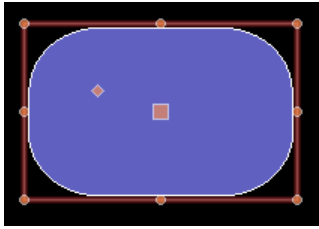
调整基元大小的方式极为直观——只需按住一个大小控点，然后将其向所需方向移动。可按住 **CTRL** 键，从而将大小调整操作限制为基元的宽度和高度相等。可按住 **SHIFT** 键，从而允许从基元中心而不是边缘开始进行大小调整操作。

## 使用布局控点

某些基元拥有内部控点，移动它们即可更改基元的布局。例如，下面显示的圆角矩形的左上角有一个布局控点。只要选定基元，该控点就会用菱形标记出来：



该例中，移动控点会更改矩形边角的半径：

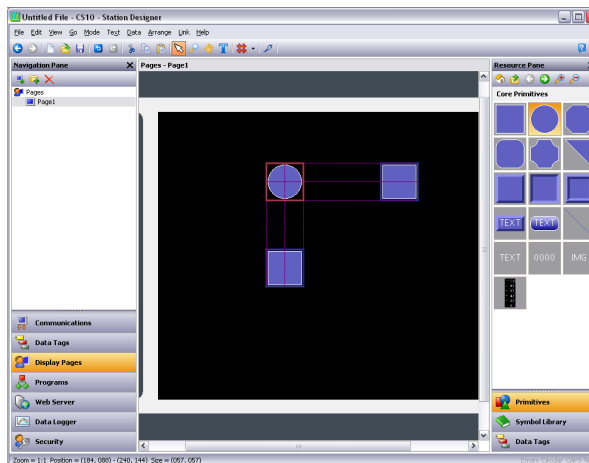


基元不同，控点的作用就也不同，但通常都很直观。

### 智能对齐

如果启用了查看菜单里的智能对齐功能，则进行移动或调整大小时 Crimson 就会提供参考线。这些可以帮助将基元与现有基元或显示中心对齐。只需稍微进行操作，创建基元时就可使用此功能方便地将基元对齐，而无需返回显示页面进行调整来对齐各种图形。

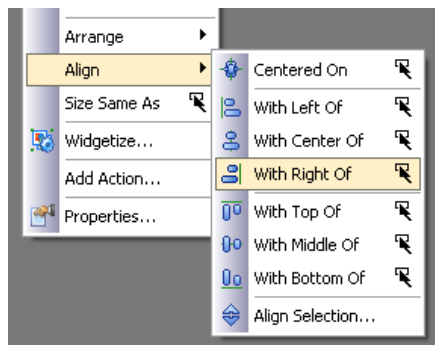
下面显示的示例中，正将一个圆形与两个正方形对齐：



参考线出现在图形的边缘和中心，显示了边缘和中心均被对齐。红色矩形标出了正在操纵的基元，而蓝色矩形则标出了已为其画出了参考线的基元。

### 快速对齐

Crimson 的快速对齐功能允许在不使用对话框的情况下将基元对齐至其它基元。要使用此功能，只需选择希望移动的基元，然后右键单击，打开上下文菜单。选择对齐子菜单，然后选择一个矩形和光标符号标示出的“...为”选项。鼠标指针会变化，从而指示现在需要单击希望对齐的基元。



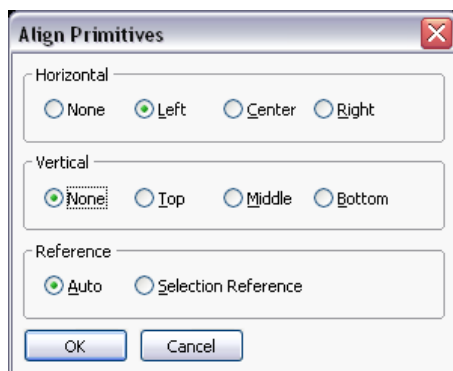
单击之后，就会进行对齐。

## 使用网格

工具栏上的网格按钮可用于控制对齐网格的行为和显示。单击按钮的左侧会显示或隐藏网格。单击下拉部分会允许配置要为其使用网格的操作。可单独为创建操作、大小调整操作和移动操作启用或禁用网格，也可使用全部选项或无选项来全局启用或禁用网格。还可控制在分组内进行编辑时是否使用网格。

## 对齐基元

虽然可使用上面讨论的智能对齐选项和快速对齐选项方便地进行多种对齐操作，但是有时您也可能会希望使用更为传统的方法。要这样做，请选择若干基元，然后使用排列菜单里的对齐所选内容命令来显示下列对话框：



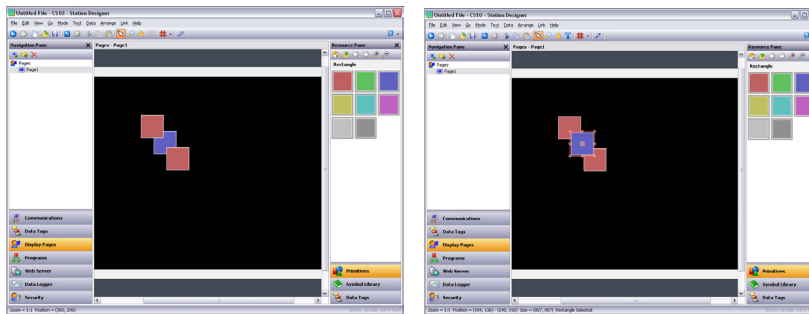
水平设置和垂直设置可用于指示要执行何种类型的对齐，而参考设置则定义了应使用哪个基元作为对齐操作的参考。上面的示例中，进行左对齐时，自动模式将使用最左侧的基元作为参考，而另一种模式则将使用选择的第一个项作为参考。可通过其中心较大的正方形来辨明此项。

## 为基元设置间距

如果希望在页面内为若干基元设置相等间距，则可使用排列菜单里的垂直等间距排列命令或水平等间距排列命令。这些命令会对当前选定的基元进行操作，并试图重新分配项之间的自由空间，从而实现相等间距。最外侧的两个基元会被留在当前位置。请注意，如果选定了一组不适当的基元，则命令可能失效，如果空间有限，则可能不会实现完美间距。

## 为基元重新排序

显示页面里的基元会按 Z 型顺序进行排列。这定义了基元的绘制顺序，并进而定义了一个给定基元是否应显示在另一个基元前面或后面。下面的第一个示例中，蓝色正方形显示在红色正方形后面，亦即位于 Z 型顺序的底部。第二个示例中，它被移到了顺序的顶层，显示在其它图形前面。



要移动 Z 型顺序里的项，请选择项，然后使用排列菜单里的各种命令。向前移动命令和向后移动命令会将所选内容向指定的方向移动一步，而移至顶层命令和移至底层命令会将所选内容移至 Z 型顺序的指定末端。如果您拥有配备了滚轮的鼠标，那么还可通过按住 **CTRL** 键并滚动滚轮来移动所选内容。向上滚动滚轮会将所选内容移至底层，向下滚动滚轮则会将所选内容移至顶层。

## 重复基元

**CTRL+D** 组合键和编辑菜单里的智能重复命令可用于复制当前基元，同时调整其属性，这样基元就可以从下一个数据项获取控制数据。“下一个”的定义依赖于数据的准确类型，因为 **Crimson** 可以选择通信设备里的下一个寄存器、数组的下一个成员或序列里的下一个标记。例如，重复为一个映射至 `Array[0]` 的按钮使用智能重复，会生成一个按钮序列，其中的按钮依次映射至 `Array[1]`、`Array[2]`，并依次类推，直至填满整个屏幕。

## 编辑多个基元

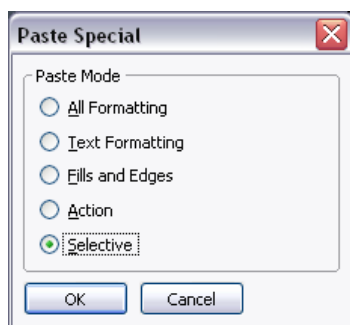
您可能偶尔会希望编辑多个基元的属性。Crimson 支持这种操作，您可以编辑一个基元，然后将若干其它基元的属性设为等于首先编辑的那个基元的属性。Crimson 提供了两种进行这种操作的方法，这两种方法均依赖于相同的基础机制。

### 使用复制自

复制自命令可用于将给定基元的选定属性复制到一个或多个其它基元。要使用此命令，请选择所需目标，然后右键单击，打开关联的上下文菜单。选择一个复制自命令，然后光标会变化，允许选择一个要从其进行复制操作的基元。根据选定的命令，源的一个或多个属性会被应用到目标基元。

### 使用选择性粘贴

选择性粘贴命令可用于实现同样的结果，不过使用的是不同的方法，允许在多个数据库之间或多个 Crimson 运行实例之间复制属性。首先，选择源基元后使用复制命令将其放入剪贴板。然后，选择所需目标基元，右键单击所选内容，选择选择性粘贴命令。会出现下面的对话框：



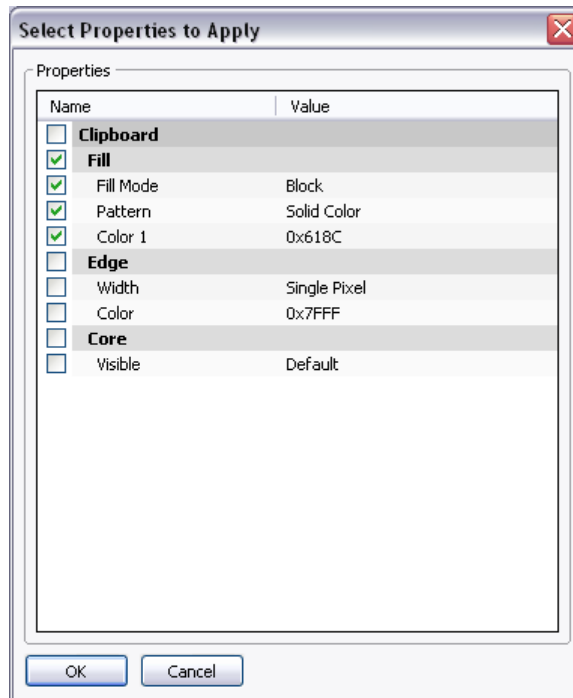
从源基元选择的属性会被应用到目标基元。

### 属性选择

上文描述的两种方法均允许定义要复制哪些属性：

- *全部格式* 会复制除文本、数据项和操作之外的全部事项。
- *文本格式* 会复制文本或数据项的字体、对齐和边距。
- *填充与边缘* 会从图形选项卡复制填充和边缘。
- *操作* 会复制指定给基元的任何操作。

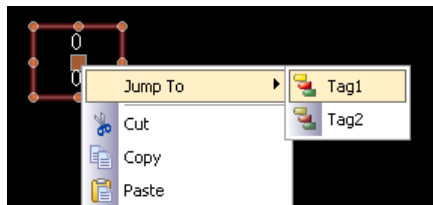
此外，*选择*选项可用于选择要复制的属性：



列表层次化地显示了源基元定义的全部属性，按照编辑基元时使用的布局对属性进行组织，并显示了分配给每个属性的值。可使用关联的勾选框选择或取消选择每个属性或属性组。会应用勾选的属性，提供对从一个基元向另一个基元复制何种内容的低层次控制。

### 跳转至其它项

如果一个基元引用了标记、显示页面或其它项，则会在其上下文菜单里出现一个跳转子菜单。选择此菜单，从而查看引用的项的列表。选择那些项中的一个，从而直接跳转至数据库的那部分。下例显示了一个引用了两个标记的基元：



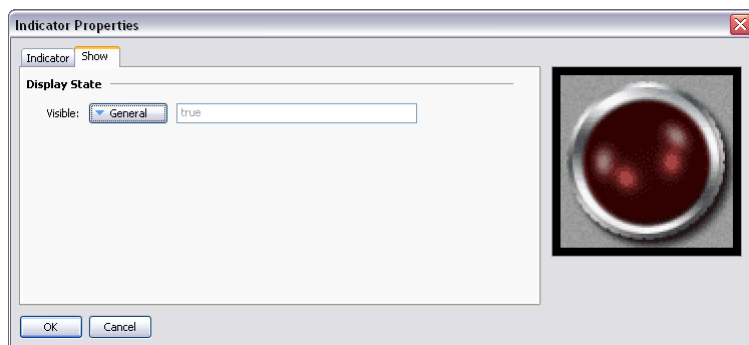
对标记进行了任意更改之后，可使用工具栏上的后退按钮或 **ALT+LEFT** 组合键来返回刚才编辑的显示页面。请注意导航期间保留所选内容的方式，这种方式使查看或编辑引用的对象和恢复显示创建流程更为简便。

## 基元属性

可通过双击基元或使用基元的上下文菜单里的属性命令来编辑基元的属性。还可以选择基元，然后按下 **ALT+ENTER** 组合键。基元的属性对话框会包含若干选项卡，某些选项卡只会在向基元添加了附加项（如文本、数据或操作）的情况下出现。属性对话框显示了当前基元的实时预览，允许在确认更改之前查看更改的效果。

### 显示或隐藏基元

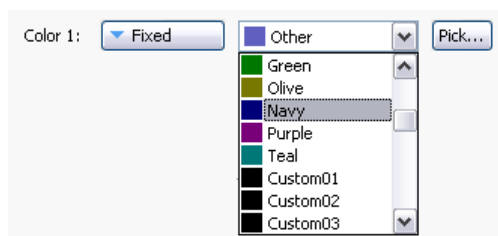
所有基元均在其属性对话框里有一个显示选项卡：



*可见* 属性可设置为整数表达式，从而在运行时显示或隐藏关联的基元。零值会隐藏基元，非零值则会允许显示基元。默认设置下所有基元均为可见。

### 定义基元颜色

基元内的颜色使用与下面显示的字段类似的字段进行编辑：



您会注意到，颜色属性通过下拉菜单按钮、下拉列表和选取按钮来显示。下拉菜单可选择下列之一的颜色动态显示模式：

- *固定* 模式下，颜色不会变化，从下拉列表中或通过按下选取按钮调用颜色选择对话框来进行选择。
- *标记文本* 模式下，颜色会动态显示，从而匹配特定标记定义的前景色。可按下选取按钮来选择特定标记。
- *标记背景* 下，颜色会动态显示，从而匹配特定标记定义的前景色。可按下选取按钮来选择特定标记。
- *闪烁* 模式下，颜色会动态显示，按特定频率在两种颜色之间交替显示，禁用闪烁时显示另一种颜色。

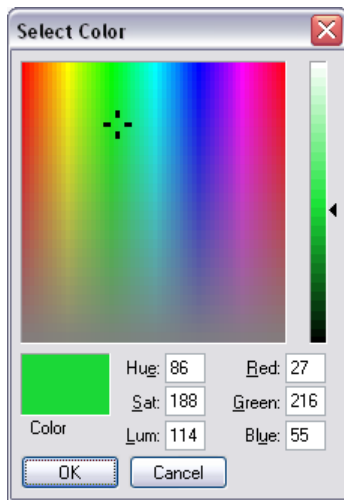


- 双状态 模式下，颜色会动态显示，根据一个标记或其它数据项的值在两种颜色之间进行切换。
- 四状态 模式下，颜色会动态显示，根据两个标记或其它数据项的值在四种颜色之间进行切换。
- 混合 模式下，颜色会动态显示，根据与特定最小值和最大值相关的一个标记或其它数据项的值，从一种颜色平滑过渡到另一种颜色。
- 表达式 模式下，可输入一个用于决定要显示的颜色数字表达式。详情请参阅下面章节。
- 复杂 模式下，可编写返回整数值的局部程序来定义要显示的颜色。详情请参阅下面章节。

下拉菜单包含了下列颜色

- 十六种标准 VGA 颜色。
- 三十二种黑色和白色之间的阴影灰色。
- 数据库中使用的任何其它颜色，可多达二十四种。

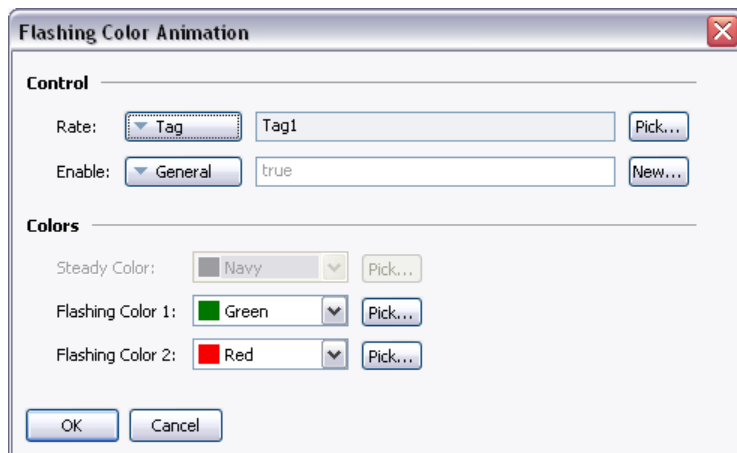
列表底部的更多选项可用于调用颜色选择对话框：



此对话框提供了若干种定义颜色的方式。可以从调色板选择，从“彩虹”窗口选择，或输入准确的 HSL 参数或 RGB 参数。如果选定的颜色尚未在数据库内使用过或不是标准颜色或灰色，则其将被添加至下拉菜单中显示的自定义颜色里。

## 定义闪烁颜色

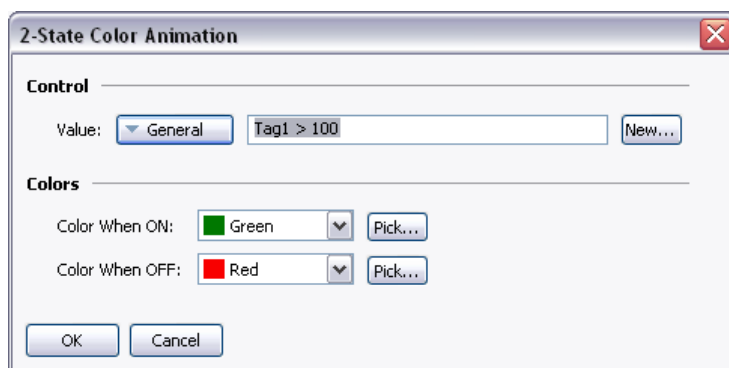
通过下列对话框定义闪烁颜色：



- *频率* 属性定义了闪烁的发生频率。值设为 1 会导致闪烁频率为 1 赫兹，每种颜色闪烁 500 毫秒。不推荐使用超过 4 赫兹的频率，因为目标设备的显示的更新频率可能会产生令人不快的“跳动”效果。
- *启用* 属性定义了可用于启用或禁用闪烁的可选表达式。禁用闪烁时，会显示平稳颜色。
- *颜色* 属性允许定义要使用的颜色。

## 定义双状态颜色

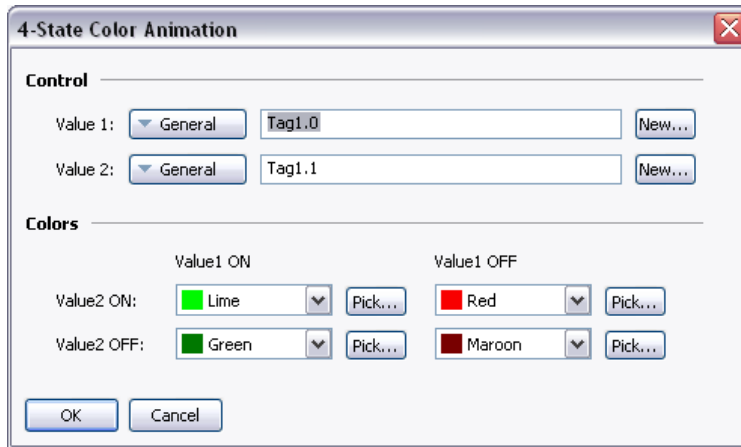
通过下列对话框定义双状态颜色：



- *值* 属性用于选择要显示的颜色。
- *颜色* 属性允许定义要使用的颜色。

## 定义四状态颜色

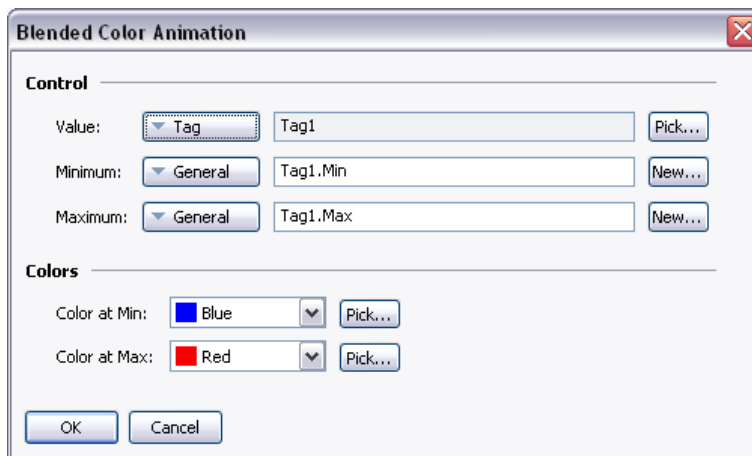
通过下列对话框定义四状态颜色：



- 值属性用于选择要显示的颜色。
- 颜色属性允许定义要使用的颜色。

## 定义混合颜色

通过下列对话框定义混合颜色：



- 值、最小和最大属性用于定义要显示的颜色。示例中，标记位于或低于其最小值时，颜色为蓝色，而标记位于或高于其最大值时，颜色为红色，标记在其极限之间更改时，会从蓝色平滑过渡到红色。
- 颜色属性允许定义要使用的颜色。

## 定义颜色表达式

如上所述，可通过整数表达式或返回整数值的局部程序来定义颜色属性。这些机制用在标准颜色动态显示方法不足的情况下。因为极少需要使用这些功能，所以，如果感觉此部分太过复杂，则可随意略过。

Crimson 使用 15 位的颜色值，最低的五个位表示红色，下五个位表示绿色，最高的五个位表示蓝色。可像操纵任何其它整数值一样来操纵颜色值。

### 创建颜色

可使用 `ColGetRGB(r,g,b)` 函数来从其红色、绿色和蓝色成分创建颜色值。虽然 Crimson 使用了包含三个 5 位值的 15 位的颜色值，但是传递至此函数的参数会被因子 8 缩放，所以，参数应位于 0 到 255 的范围之内。因此，`ColGetRGB(128, 0, 64)` 将返回一个类似于紫色的颜色，其红色值为 16，没有绿色成分，蓝色值为 8。

### 分离颜色

可使用 `ColGetRed(rgb)` 函数、`ColGetGreen(rgb)` 函数和 `ColGetBlue(rgb)` 函数来访问一个颜色值的单个颜色成分。为了与 `ColGetRed(rgb)` 使用的约定保持一致，这些函数返回的值会被缩放至 0 到 255 之间。

### 选择颜色

可使用 `ColPick2()` 函数根据表达式的值在两种颜色之间选择。例如，如果 `Flag1` 非零，表达式 `ColPick2(Flag1, Col1, Col2)` 会返回 `Col1`，如果 `Flag1` 是零，则会返回 `Col2`。如有需要，可通过调用 `ColGetRGB()` 函数来替换第一个和第二个颜色参数。

### 混合颜色

可使用 `ColBlend()` 函数通过混合两种颜色来生成用户定义的颜色。例如，如果 `Data` 是 0，表达式 `ColBlend(Data, 0, 100, Col1, Col2)` 会返回 `Col1`，如果 `Data` 是 100，则会返回 `Col2`。中间值是两种颜色的适当混合，允许从一种颜色平滑过渡到另一种颜色。同样可通过调用 `ColGetRGB()` 函数来替换颜色参数。

### 响应触摸

如果已触摸了当前基元，则系统变量 `IsPressed` 等于真，否则便等于假。它可与颜色选择函数同时使用，从而根据基元的触摸状态对基元进行动态显示。请注意，除非为基元定义了操作或基元支持固有操作，否则无法为触摸启用基元。

### 定义箱填充

许多几何基元都支持一个所谓的“箱填充”选项，通过此选项，会基于标记的内容将图填充至给定水平。此功能可用于实现简单条形图或填充更为复杂的形状。

下例显示了一个由下到上箱填充设为 60% 的六角星：



通过使用基元的填充行为属性来定义箱填充：

The 'Fill Behavior' dialog box contains the following fields:

- Fill Mode:** A dropdown menu currently showing 'Fill from Bottom'.
- Value:** A dropdown menu showing 'Tag' and a text input field containing 'Tag1', with a 'Pick...' button to its right.
- Minimum:** A dropdown menu showing 'General' and a text input field containing 'Tag1.Min'.
- Maximum:** A dropdown menu showing 'General' and a text input field containing 'Tag1.Max'.

- **填充模式** 属性定义了是否应绘制箱填充，以及填充应从哪个方向开始。填充可从基元的任何边缘开始，允许创建复杂动态显示。块模式会使用单一样式填充图形，从而禁用箱填充。
- **值** 属性选择了用于计算填充水平的值。如果输入了一个标记，则会自动使用标记属性表达式语法将最小限制和最大限制设为该标记的数据输入限制。值属性可以是整数值或浮点值。填充水平计算始终使用浮点来进行。
- **最大和最小值** 定义了对值属性进行缩放来计算填充水平时要使用的限制。

### 定义填充格式

基元的填充格式属性定义了如何填充基元的内部：

The 'Fill Format' dialog box contains the following fields:

- Pattern:** A dropdown menu showing 'Graduated Fill 2'.
- Color 1:** A dropdown menu showing 'Fixed' and a color selection area with 'Other' selected, and a 'Pick...' button.
- Color 2:** A dropdown menu showing 'Fixed' and a color selection area with 'White' selected, and a 'Pick...' button.
- Color 3:** A dropdown menu showing 'Fixed' and a color selection area with 'Gray06' selected, and a 'Pick...' button.

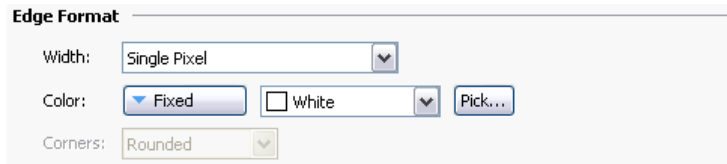
- **样式** 属性在多种填充属性之间进行选择。常用选项是纯色，但也可选择多种抖动样式和阴影线样式。还有若干种可用的渐变填充：

样式	描述
渐变填充 1	基元顶部和底部使用颜色 1，垂直向中心更改至颜色 2。
渐变填充 2	基元顶部使用颜色 1，垂直向底部更改至颜色 2。
渐变填充 3	基元左侧和右侧使用颜色 1，水平向中间更改至颜色 2。
渐变填充 4	基元左侧使用颜色 1，水平向右侧更改至颜色 2。

- **颜色 1** 属性定义了要为填充使用的第一种颜色。
- **颜色 2** 属性定义了要为填充使用的可选的第二种颜色。
- **颜色 3** 属性定义了要为箱填充使用的背景色。如果使用了块填充，则不需要此颜色。如果当前基元不支持箱填充，则可能不会出现此属性。

## 定义边缘格式

基元的边缘格式属性定义了如何为基元绘制边缘：



- *宽度* 属性指定了边缘的厚度。可选择无值来显示边缘。Crimson 当前仅支持奇数边缘大小，宽度最大可达九个像素。
- *颜色* 属性定义了边缘的颜色。
- *边角* 属性仅矩形才有，定义了绘制边缘时是否应使用圆形或正方形。其它基元全部默认使用圆形边角。

## 使用分组

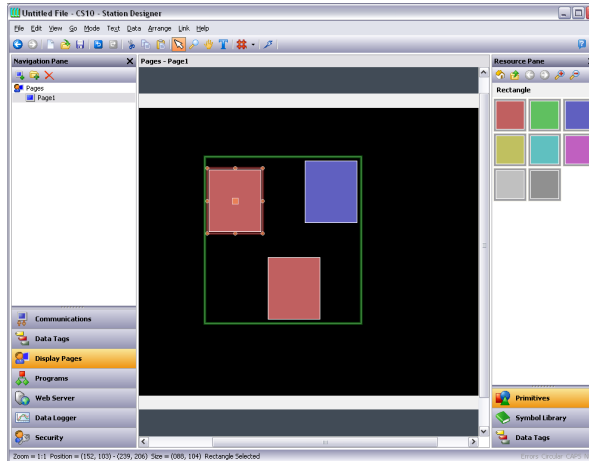
分组是被视为单一对象的基元组。

### 创建和分解分组

如果希望以这种方式处理若干基元，则可如上所述将其选定，然后使用排列菜单里的分组命令。也可按下 **CTRL+G** 组合键来进行同样的操作。分组创建之后，即可像单一对象一样进行移动、大小调整和复制。选择分组然后使用取消分组命令或按下 **CTRL+U** 组合键，即可将其分解为组件基元。请注意，分组可以包含基元和可嵌套多达任何合理限制的其它分组。

## 在分组内编辑

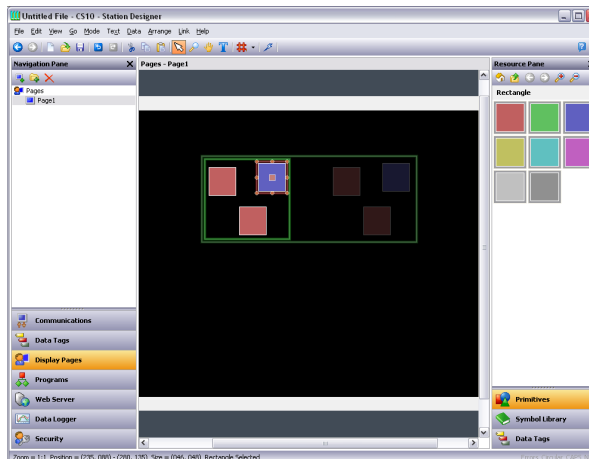
分组创建之后，您可能会希望在不分解分组的情况下编辑其内容。如果创建的是嵌套分组，那么这就特别有用，因为重新分组流程会非常困难。要在分组内编辑，首先请选择分组，然后单击分组的一个成员。（请不要单击分组对象的中央控点，因为它用于将分组作为整体进行移动或选择。）选定分组成员之后，Crimson 会切换到如下所示的分组编辑模式：



请注意显示在正在编辑的分组周围的绿色矩形。除了项不能移出分组边界之外，在分组内编辑的方法与在页面内编辑类似。项可被复制、粘贴、调整大小和删除。事实上，可进行任何有用的操作。甚至还可以从资源窗格将新项拖放入分组。要退出分组编辑模式，请在分组外单击或按下 **Esc** 键。

## 嵌套分组编辑

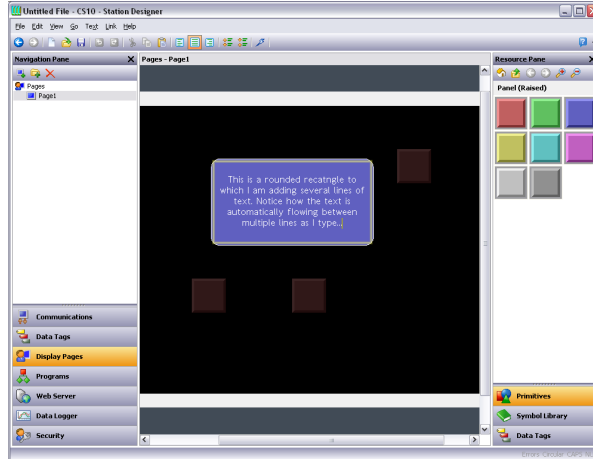
Crimson 还允许在本身就位于分组内的分组里进行编辑：



要激活此功能，首先请在外侧的分组内开始编辑，然后选择内侧分组，并单击内侧分组里的成员。请注意在上面的示例中一系列淡出矩形如何用于显示分组层次。还请注意当前分组外的项如何显示为淡出颜色，从而更便于分辨分组在哪里结束。使用 **Esc** 键退出嵌套分组编辑时，每按一下键就会向上移一层。

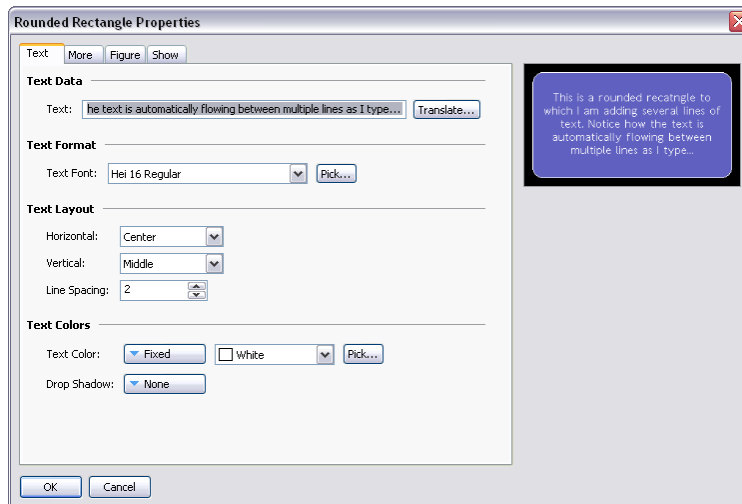
## 向基元添加文本

Crimson 里的大多数基元都可支持添加文本。要向基元添加文本，只需选择基元，按下 **F2** 键，然后开始键入。还可以右键单击基元，然后从出现的菜单中选择添加文本命令。下例显示了正在输入到一个圆角矩形中的文本：



首先，请注意如何为基元显示黄色边界矩形以及其它全部基元如何淡出。其次，请注意文本编辑器如何在多行之间自动分隔文本。试着调整包含文本的基元的大小，您就会了解 Crimson 如何重新调整文本从而使其适合于新形状。

文本编辑过程中，工具栏会发生变化，从而提供修改文本对齐、放大或缩小行间距的命令。通过选择基元的上下文菜单里的文本属性或在文本编辑模式下按下 **ALT+ENTER** 组合键，即可编辑更多高级文本属性：

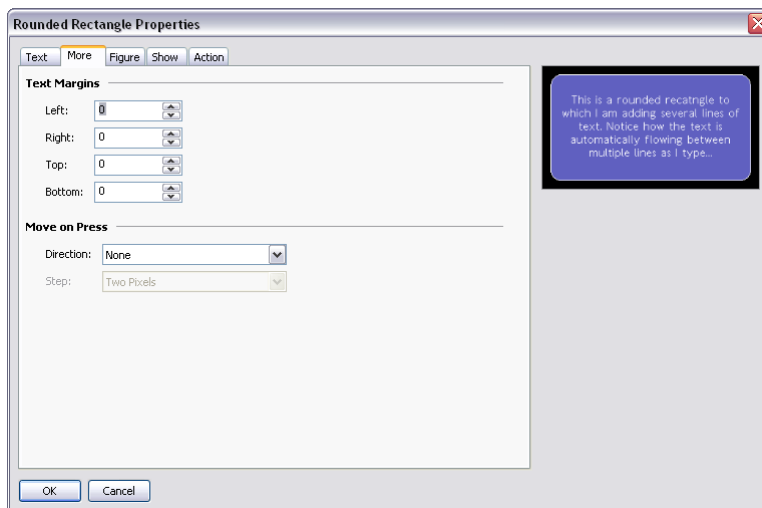




## 文本属性

- 文本 属性包含了要显示的文本。可使用垂直条字符强制进行换行。因为此字段是可译字符串，所以可编辑多语言版本。这还暗示了此属性可设为表达式，从而允许动态更改其内容。Crimson 支持完全动态重调整，可以使用复杂的、引人注目的展示选项。
- 文本字体 属性允许选择所需字体。Crimson 的全新默认字体为 Hei，这是一种支持简体中文和大多数其它语言的 Unicode 字体。可使用选取按钮来调用字体选择对话框，允许将您的系统内安装的任何字体转换为可被目标设备使用的形式。请注意，需由您自己负责确保您已获得使用此类字体的授权。
- 水平属性定义了文本的水平对齐。
- 垂直属性定义了文本的垂直对齐。
- 行距属性定义了像素之间的附加行距。
- 文本颜色 属性选择了文本的颜色。
- 投影 属性用于启用文本右侧和底部的可选阴影。试图将文本与其背景明确区分开来时，尤其是背景是含有多种颜色的图像时，此效果非常有用。

## 更多属性

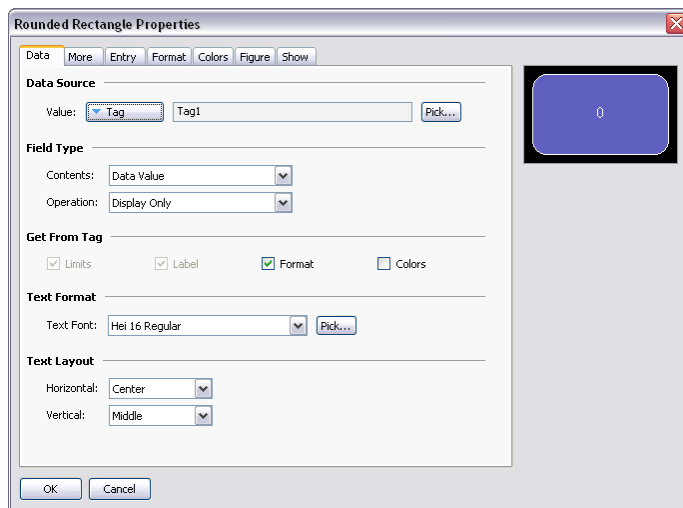


- 文本边距 属性用于控制文本周围相对基元提供的文本边界框而言的边距。使用在字符上下为发音符或下行字母留有大量空间的字体时，这些属性可用于获取更好的视觉居中显示效果。
- 方向 属性定义了按下关联的基元时文本的移动方向。只有向基元分配了操作或基元是按钮之类的拥有与其关联的固有操作时，此属性才能启用。创建触摸时应进行反馈的自定义按钮时，此选项十分有用。
- 步长 属性指示了按下基元时应将文本移动多远。可根据所需效果选择一至三个像素。

## 向基元添加数据

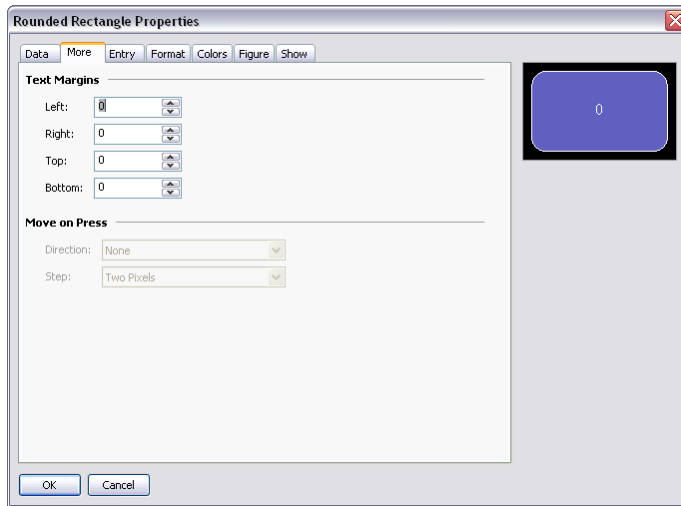
支持添加文本的基元还支持显示实时数据，并可为数据输入对其进行配置。要向基元添加数据，请右键单击基元，然后选择出现的菜单里的添加数据命令。还可选择基元，然后按下 **F3** 键。会显示基元的属性对话框，其中拥有若干可用于定义所需数据项及其行为的附加选项卡。

### 数据属性



- *值* 属性定义了要显示的数据值。
- *内容* 属性定义了字段是否应显示数据值、显示数据值及其关联标签，或仅显示标签。
- *操作* 属性定义了字段是否应仅显示值或同时提供数据输入功能。显然，只有所选数据值是可写的时，才可进行数据输入。
- *从标记获取* 属性定义了某些数据字段的属性是否被局部定义或被链接至正在显示的标记的属性。只有在值里指定了标记时，这些选项才可用。
- *文本字体* 属性允许选择所需字体。Crimson 的全新默认字体为 Hei，这是一种支持简体中文和大多数其它语言的 Unicode 字体。可使用选取按钮来调用字体选择对话框，允许将您的系统内安装的任何字体转换为可被目标设备使用的形式。请注意，需由您自己负责确保您已获得使用此类字体的授权。
- *水平* 属性定义了文本的水平对齐。
- *垂直* 属性定义了文本的垂直对齐。

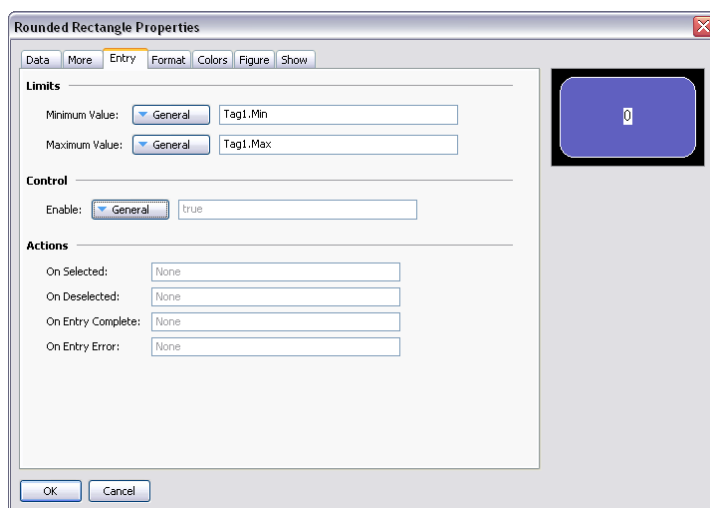
## 更多属性



- **文本边距** 属性用于控制文本周围相对基元提供的文本边界框而言的边距。使用在字符上下为发音符或下行字母留有大量空间的字体时，这些属性可用于获取更好的视觉居中显示效果。
- **方向** 属性定义了按下关联的基元时文本的移动方向。只有向基元分配了操作或基元是按钮之类的拥有与其关联的固有操作时，此属性才能启用。创建触摸时应进行反馈的自定义按钮时，此选项十分有用。
- **步长** 属性指示了按下基元时应将文本移动多远。可根据所需效果选择一至三个像素。

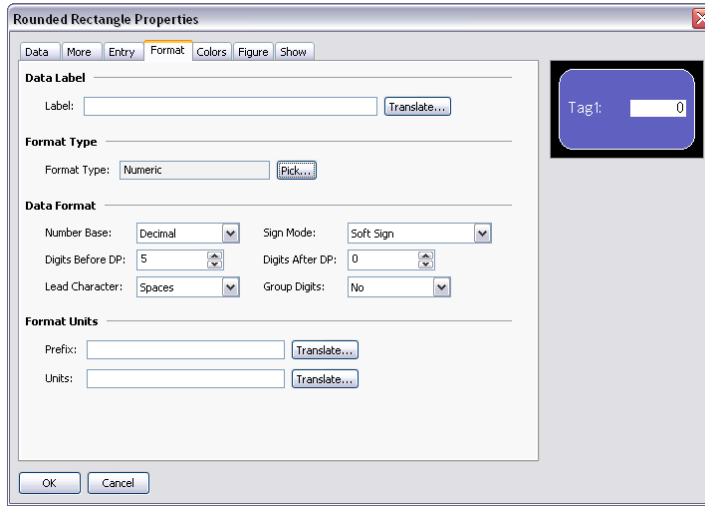
## 输入选项

这些属性仅在启用了数据输入时才可用：



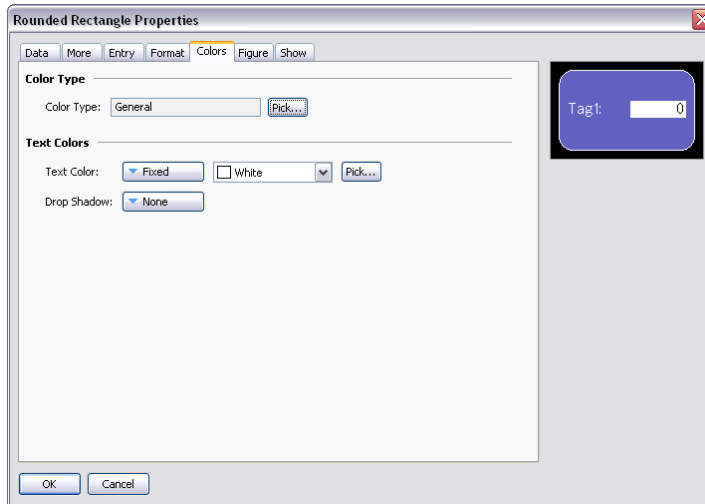
- *最大值* 和 *最小值* 属性定义了数据输入限制。如果字段设置为从控制标记获取其数据输入限制，则这些属性将不可用。并非全部格式类型均允许进行这些设置，如果其数据限制已隐式定义，则更是如此。
- *启用* 属性用于提供启用或禁用数据输入的表达式。禁用的数据输入字段仅能用作只显示字段。
- *选定时* 属性指定了用户按下数据输入字段时在数据输入开始之前要执行的操作。
- *取消选定时* 属性指定了数据输入因为值写入、页面更改或用户按下取消输入流程的按钮而结束时要执行的操作。
- *输入完成时* 属性指定了数据输入成功完成时要执行的操作。
- *输入错误时* 属性指定了用户输入无效值时要执行的操作。

## 格式属性



- **标签** 属性定义了要应用到此字段的标签。如果标签不会被显示，或字段配置为从控制标记获取其标签，则此属性可能会不可用。
- **格式类型** 字段指定了显示和可选编辑数据值时要显示的格式类型。同样，如果是从控制标记获取格式，则此属性可能会不可用。
- 其它属性则只针对选定的数据格式可用。请参阅使用格式章节，详细了解每种格式的属性。

## 颜色属性

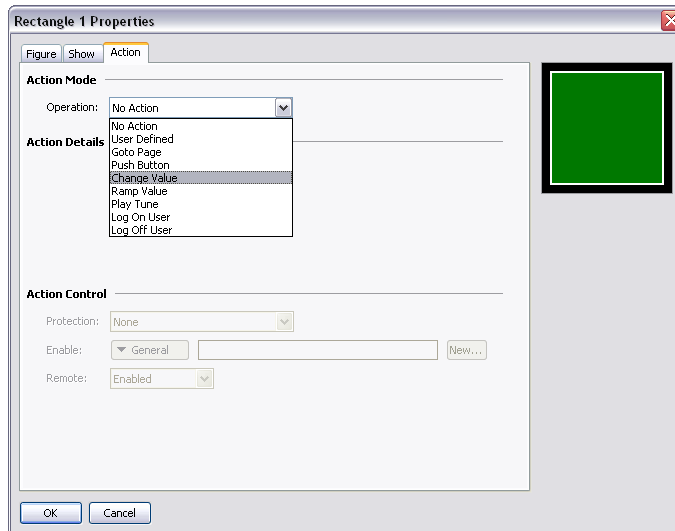


- **颜色类型** 字段指定了显示数据值时要使用的着色。如果从控制标记获取着色，则此字段可能会不可用。
- **文本颜色** 属性用于在使用常规着色时重写文本的颜色。
- **投影** 属性用于启用文本右侧和底部的可选阴影。试图将文本与其背景明确区分开来时，尤其是背景是含有多种颜色的图像时，此效果非常有用。此属性仅可用于常规着色。

- 其它属性则只针对选定的着色可用。请参阅使用着色章节，详细了解每种着色的属性。

## 向基元添加操作

可向不执行自身隐式操作的基元添加操作员按下或松开触摸屏时要执行的自定义操作。选择基元的上下文菜单里的添加操作命令，或选择基元后按下 **CTRL+I** 组合键，即可添加操作。会为基元的属性添加一个操作选项卡，并会出现属性对话框：



## 保护操作

操作的保护属性可用于防止意外调用操作。此功能配合了安全系统提供的其它保护功能，在关联的操作开始之前被调用。可用保护模式如下：

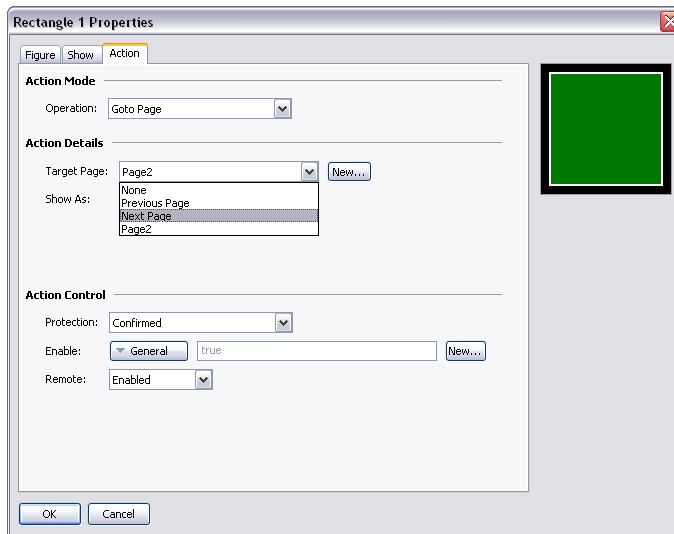
- **确认** 模式会显示一个确认操作的弹出窗口，如果用户指示操作应继续，则会执行操作。
- **锁定** 模式会显示一个说明操作已被锁定的弹出窗口。如果用户指示操作应继续，则其会被解锁，用户必须再次激活操作，从而真正执行操作。选择另一个操作会锁定前一个操作，并会等待全局超时。
- **硬锁定** 模式原理与锁定模式类似，只是在操作执行之后会重新锁定操作，每次均需进行解锁。

## 启用操作

如果希望特定操作只在某条件为真时才能执行，请在 *启用* 字段内为该条件输入表达式。该表达式可直接引用一个标志标记，也可使用编写表达式章节定义的任何比较运算符或逻辑运算符。如果需要更为复杂的逻辑（如若干操作之一根据更为复杂的决策来执行），请为用户定义模式配置键，然后使用该键来调用实现所需逻辑的程序。还可使用 *远程* 属性来阻止 Web 服务器对此操作的访问。

## 前往页面操作

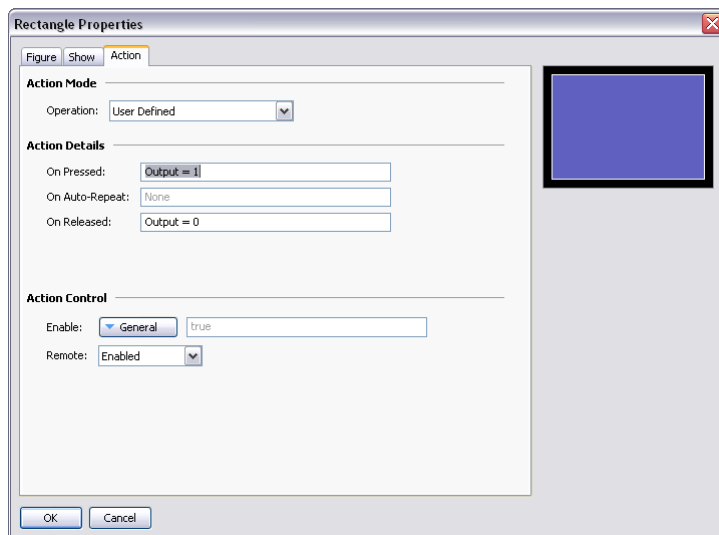
此操作用于指示目标设备显示一个新页面：



- *目标页面* 属性用于指示要显示哪个页面。除了数据库里包含的页面之外，还可以选择上一页或下一页来在页面历史列表中导航。使用新建按钮，无需离开对话框即可创建新页面。
- *显示为* 属性用于指示如何显示页面。选择正常页面会使页面按常规方式选定，而弹出窗口选项会使新页面上的基元显示在当前页面顶上的矩形弹出窗口里。可通过执行 `HidePopup()` 函数来关闭弹出窗口。

## 用户定义的操作

此操作用于执行一个或多个用户定义的操作：



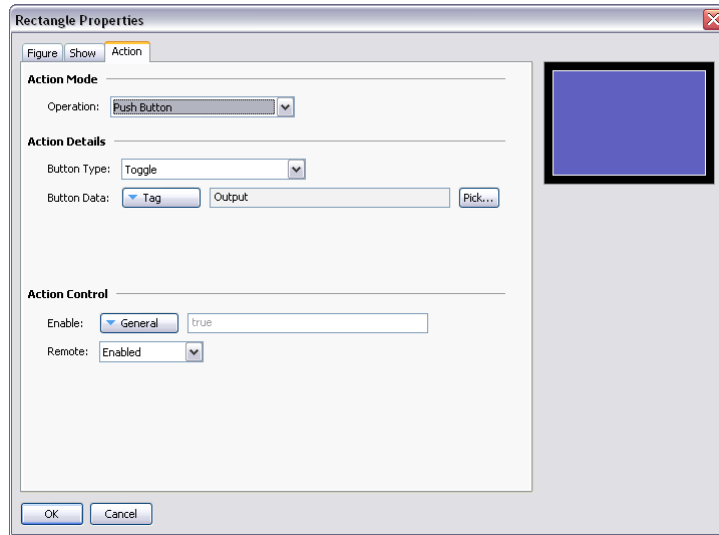
- *按下时* 属性定义了按下基元时要执行的操作。此操作可调用函数参考里的任何函数或编写操作章节里的数据修改运算符，还可运行一个程序来执行更为复杂的操作。
- *自动重复时* 属性定义了按下后按住基元时要执行的操作。最初按下时和随后自动重复时，均会发生操作，因此，无需同时定义此属性和按下时属性。此操作可调用函数参考里的任何函数或编写操作章节里的数据修改运算符，还可运行一个程序。
- *松开时* 属性定义了松开基元时要执行的操作。此操作可调用函数参考里的任何函数或编写操作章节里的数据修改运算符，还可运行一个程序。

上面的示例中，使用了一个用户定义的操作来实现一个临时按键。



## 按下按钮操作

此操作用于模拟一个按键：



- *按钮类型* 属性选择了所需的行为：

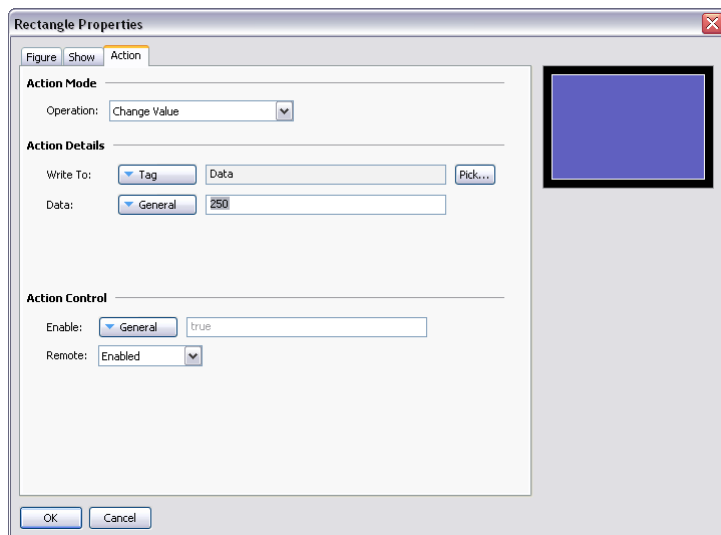
按钮类型	基元行为
扳钮	按下基元时更改数据状态。
常开瞬时	按下基元时将数据设为 1。 松开基元时将数据设为 0。
常闭瞬时	按下基元时将数据设为 0。 松开基元时将数据设为 1。
开启	按下基元时将数据设为 1。
关闭	按下基元时将数据设为 0。

- *按钮数据* 属性定义了要被键更改的数据。

上面的示例中触摸基元会切换输出标记的值。

## 更改值操作

此操作用于向数据项写入数字值：

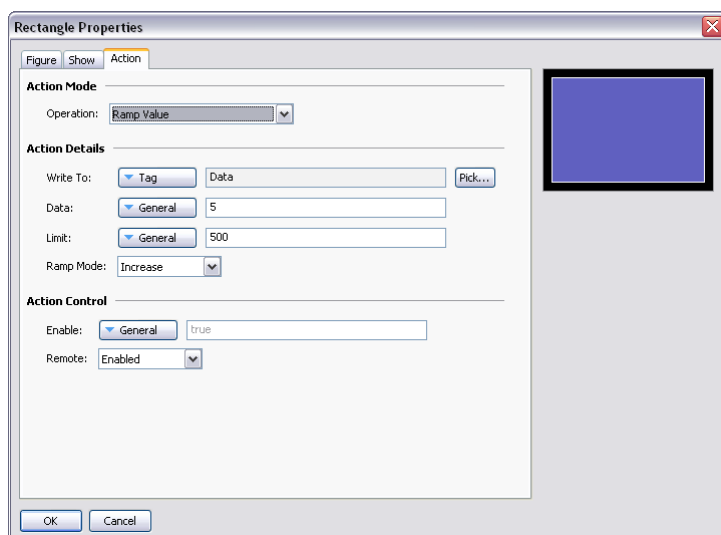


- 写入至属性定义了要更改的数据项。
- 数据属性定义了要写入的数据。

上面的示例中，触摸基元会将数据标记设为 250。请注意，此操作支持浮点值和整数。数据属性的类型必须适合于写入至属性定义的数据项。

## 校正值操作

此操作用于提高或降低数据项。选项如下所示：



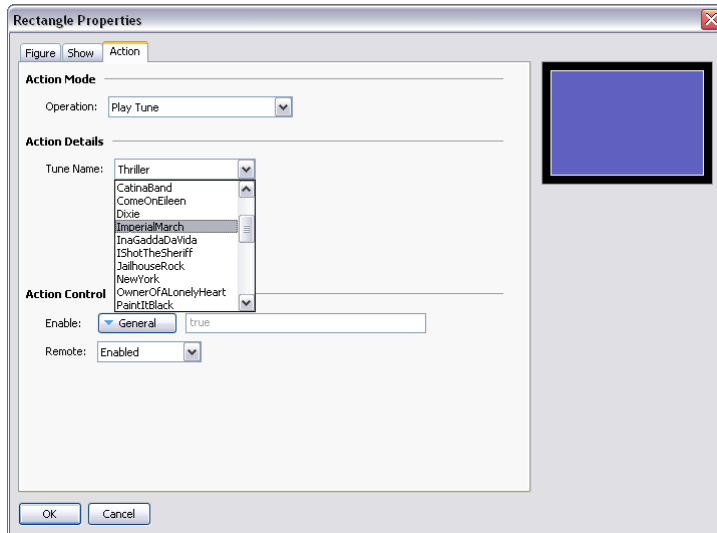
- 写入至属性定义了要更改的数据项。
- 数据属性定义了要将项提高或降低的步长。
- 限制属性定义了最小数据值或最大数据值。

- **校正模式** 属性定义了是否提高或降低项。

上面的示例中，按下并按住基元会重复提高数据标记，每次提高 5，直至数据达到 500。请注意，此操作支持浮点值和整数值。数据属性和限制属性的类型必须适合于写入至属性定义的数据项。

### 播放音乐操作

此操作使用目标设备的内部报警器播放选定的音乐。



- **音乐名称** 选择了要播放的音乐。

可使用 `PlayRTTTL()` 函数来播放自定义音乐。

### 登录用户操作

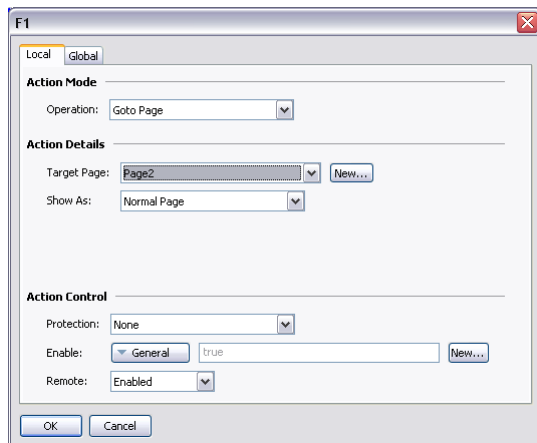
此操作会在目标设备上激活登录屏幕，没有选项。

### 注销用户操作

此操作会将当前用户从目标设备注销，没有选项。

## 向键添加操作

可向目标设备的键添加操作。请进行缩小，直至看到键，然后双击一个键，打开其属性：



您将注意到此对话框含有两个选项卡，每个选项卡均定义了一个操作。第一个选项卡定义了显示当前页面时要被此键执行的操作，第二个选项卡则定义了要在每个页面上执行的操作。这些操作分别叫做局部操作和全局操作。

用于显示键的颜色会根据定义了哪些操作发生变化：



如果键显示为紫色，则为此页面定义了一个局部操作。



如果键显示为绿色，则定义了一个全局操作。



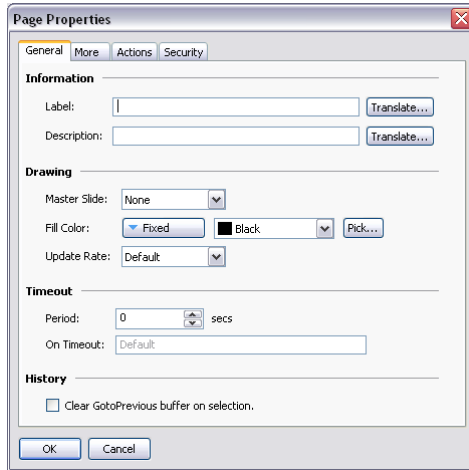
如果键显示为蓝色，则定义了局部操作和全局操作。

定义了一个操作之后，即可右键单击键，使用出现的菜单来选择设为全局或设为局部，从而更改操作类型。如果已定义了这两种类型的操作，则这些选项将不可用。

## 编辑页面属性

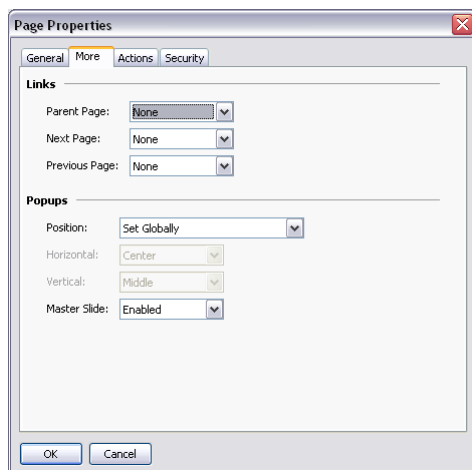
在编辑窗格里的任何基元之外的部分右键单击，就会激活上下文菜单，允许选择属性命令来编辑显示页面的属性：

### 常规属性



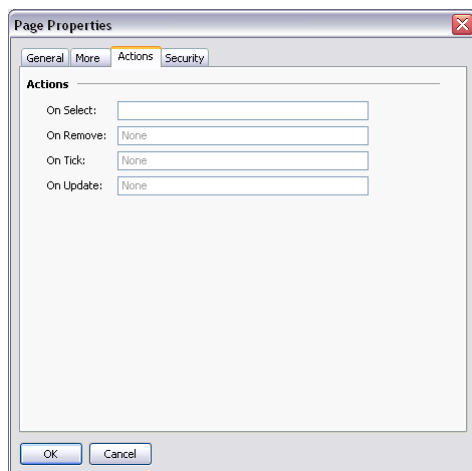
- **标签** 和 **描述** 属性定义了可在其它地方使用 **Crimson** 的属性提取语法访问的常规目的可译字符串。详情请参阅编写表达式章节。
- **主幻灯片** 属性允许选择用作当前页面的背景的另一个页面。这允许在一个单一页面上绘制时钟、警报状态指示器等常见用户界面元素，然后将其包含在若干其它页面里。
- **填充颜色** 属性定义了没有使用主幻灯片时的页面的背景颜色。应避免动态显示背景颜色，因为颜色变化需要硬件在页面上重新绘制所有项，可能会影响系统性能。
- **更新频率** 属性定义了页面的更新频率。正常情况下不应使用超频设置。默认设置即为当前标准设置。
- **超时** 属性定义了超时行为。如果经过了等于 **时间量** 的时间，且没有任何用户活动，则会执行 **超时时** 操作。请参阅编写操作章节，了解可能的操作。
- **清除 GotoPrevious 缓冲区** 属性指示了选定此页面时应清除 `GotoPrevious()` 和 `GotoNext()` 保存的历史缓冲区。通常会在数据库的主菜单页面上设置此属性，从而使用户无法从该点后退。

## 更多属性



- **链接** 属性组允许在一个显示页面上通过标准操作选择若干页面。父页属性定义了发生超时且没有定义操作时要选择的页面。下一页属性定义了启用输入导航且焦点移到页面上最后一个字段之外时要选择的页面。上一页属性定义了焦点移到页面上第一个字段之外时要选择的页面。
- **位置** 属性允许重写此页面的弹出窗口的全局定义位置。如果启用了局部设置，则水平和垂直属性用于指定位置。
- **主幻灯片** 属性用于指示显示弹出窗口时是否应保持主幻灯片活跃。默认设置为启用，允许使用主幻灯片上的按钮，即使出现弹出窗口时实际页面上的按钮被禁用。如果希望主幻灯片上的全局导航选项始终可用，那么这就非常有用。

## 操作属性



- **选定时** 属性定义了显示页面时要执行的操作。
- **移除时** 属性定义了取消选择页面时要执行的操作。
- **滴答时** 属性定义了每秒要执行一次的操作。

- *更新时* 属性定义了每次页面更新时要执行的操作。

### 安全属性

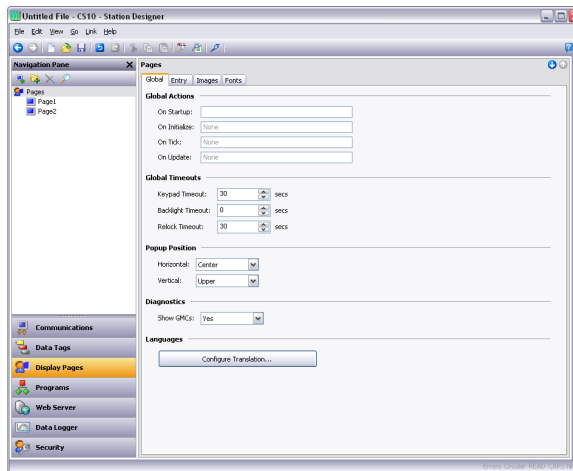
请参阅使用安全章节，了解安全描述符。

## 用户界面设置

选择导航列表里的根项即可访问用户界面设置。

### 全局属性

全局选项卡包含了可在整个数据库内应用的各种常规设置：



### 全局操作

- *启动时* 属性定义了系统启动时要执行的操作。
- *初始化时* 属性定义了稍后要执行的操作。<sup>1</sup>
- *滴答时* 属性定义了每秒要执行一次的操作。
- *更新时* 属性定义了每次页面更新时要执行的操作。

### 全局超时

- *键盘超时* 属性定义了一个没有用户操作的时间量，此时间量过后，任何数据输入操作均将被取消，关联的弹出键盘也将被从页面移除。
- *背光超时* 属性定义了一个没有用户操作的时间量，此时间量过后，显示背光将被关闭，从而节省动力，延长显示寿命。默认设置为零，会禁用此功能。
- *重锁超时* 属性定义了一个时间量，此时间量过后，通过锁定和硬锁定方法保护的任何操作均将被自动重锁，这样一来，使用之前用户必须再次对其进行解锁。

<sup>1</sup>这两个属性之间的区别非常细微，大多数用户无需关注。

## 弹出位置

- *水平* 和 *垂直* 属性定义了弹出显示页面和弹出键盘的默认位置。如果希望使用页面自身的属性来指定新值，则可在页面级别将这些属性重写。

## 诊断

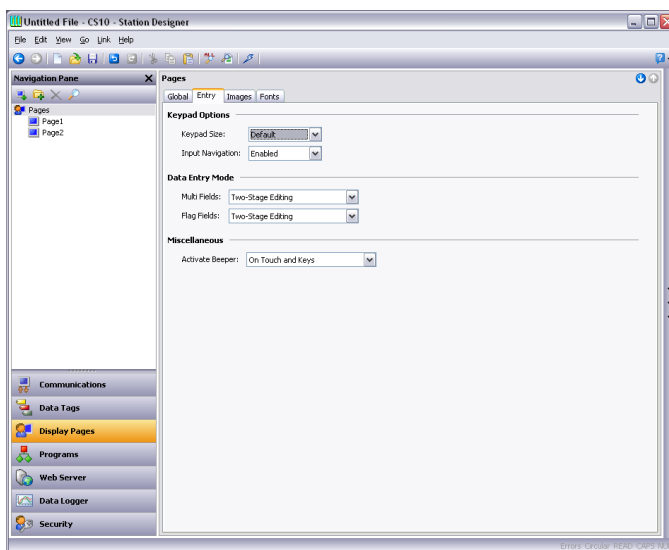
- *显示 GMC* 属性用于启用或禁用运行时系统错误之后显示某些诊断信息。这些信息对于 Red Lion 修正软件问题而言非常有用，但是可能会使用户感觉混乱。

## 语言

- *配置翻译* 按钮用于配置要在系统内使用的语言。详情请参阅本地化章节。

## 输入属性

输入选项卡包含了应用至数据输入的全局设置：



## 键盘选项

- *键盘大小* 属性用于选择数据输入键盘的大小。各种设置会逐渐增加键盘尺寸，而设置为最大会使键盘占据屏幕的大部分，可在操作员戴着厚重手套等情况之下使用。
- *输入导航* 属性用于在各种键盘上显示或隐藏上一个键和下一个键。这些键可用于在无需关闭键盘的情况下在输入字段之间进行移动。



## 数据输入模式

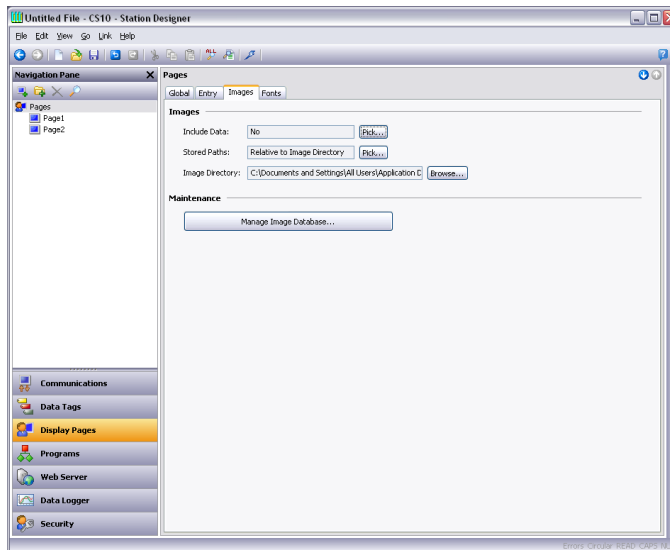
- **多数据输入** 属性用于控制为多状态格式对象使用的数据输入模式。双阶段编辑下，必须按下**回车键**，从而确认更改；单阶段编辑下，只要使用**升高**或**降低**来进行更改，新数据就会被写入关联的数据项。单阶段编辑速度更快，但是更改多状态设置时可能会导致写入中间值。
- **标志数据输入** 属性用于控制为双状态格式对象使用的数据输入模式。操作方式与上面的属性相同。

## 杂项

- **激活蜂鸣器** 属性用于按需开启或关闭目标设备的蜂鸣器。蜂鸣器可以为键盘激活和触摸屏激活提供反馈，但开发流程中可能会变得很烦人。

## 图像属性

图像选项卡用于管理数据库里的图像：



## 图像

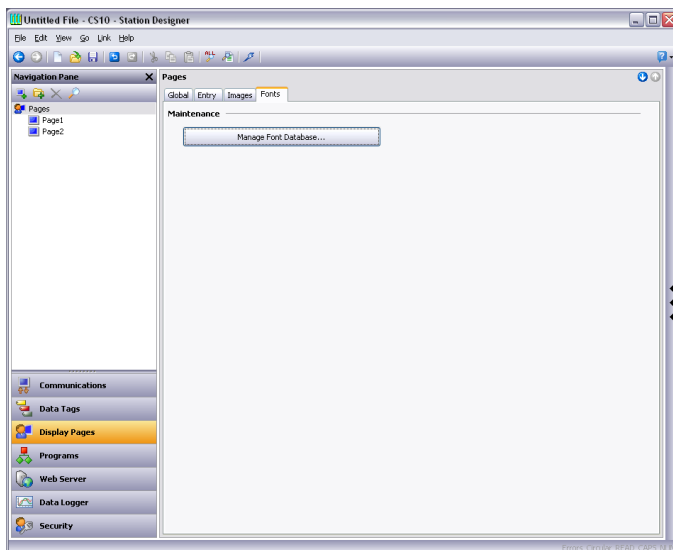
- **包含数据** 属性指示了拖入显示页面的外部图像是否应被存储为源位置的指示点，或是否应在数据库文件内包含真正的图像数据。包含图像数据通常会使数据库变得极大，如果不扩充目标设备的内存，就无法使用支持上传功能。
- **存储路径** 属性定义了如何存储图像链接。绝对模式会存储包括驱动器号在内的完整路径。两种相对模式则会存储并解释相对于数据库或相对于 **Crimson** 图像目录的图像路径，允许在不同机器之间移动数据库和图像文件，而无需过多担心绝对路径位置。
- **图像目录** 属性定义了上面所说的图像路径。

## 维护

- *管理图像数据库* 按钮用于调用图像管理器，从而查看并操纵数据库里使用的图像。请参阅下列章节，详细了解此功能。

## 字体属性

字体选项卡用于管理数据库里的字体：



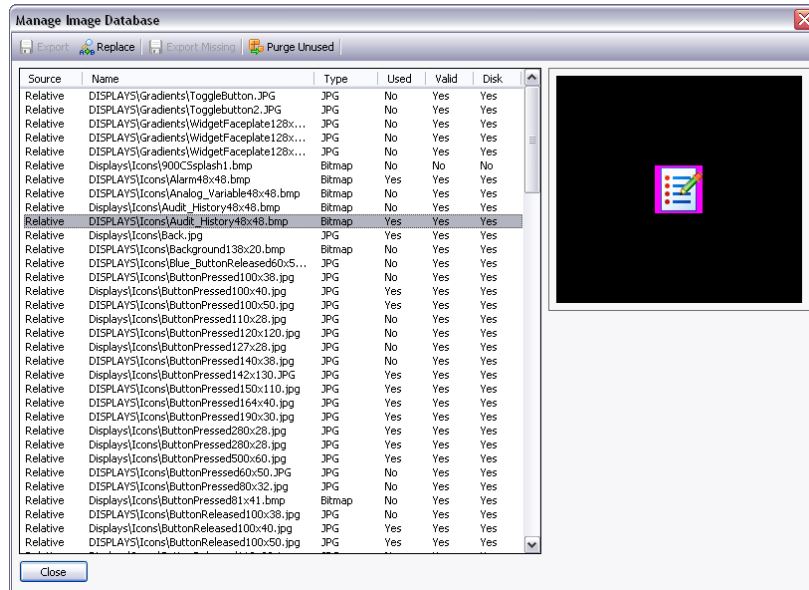
## 维护

- *管理字体数据库* 按钮用于调用字体管理器，从而查看并操纵数据库里使用的字体。请参阅下列章节，详细了解此功能。

## 管理图像

可从用户界面设置的图像选项卡里调用图像管理器。它包含了数据库里引用的全部图像及其属性的列表。它允许查看图像，并对图像的存储和使用方式进行某些更改。

下面的示例显示了一个复杂数据库里的图像管理器：



主列表视图显示了各种图像的属性：

- **源** 列指示了是否通过固定路径或相对路径从文件获取图像、从符号库获取图像，或从在图像从其它源传递入或拖入时存储的内部数据获取图像。
- **名称** 列显示了存储在文件里的图像的文件名，以及从符号库获取的图像的相关符号信息。
- **类型** 列显示了图像数据的文件类型。
- **已使用** 列指示了是否在数据库里使用了图像。
- **有效** 列指示了有效图像数据是否可用。如果图像是从不再可用的磁盘文件获取的，且数据库未配置为通过上文描述的包含数据属性持有自身图像数据，则可将此列设为否。
- **磁盘** 列指示了磁盘里是否存在图像。直接传递入或拖入编辑器的图像可能根本不会在磁盘里存在过，对于从文件获取且存储在数据库里的图像而言，如果文件不再可用，那么这些图像就会缺失。

窗口顶部的工具栏允许执行各种命令：

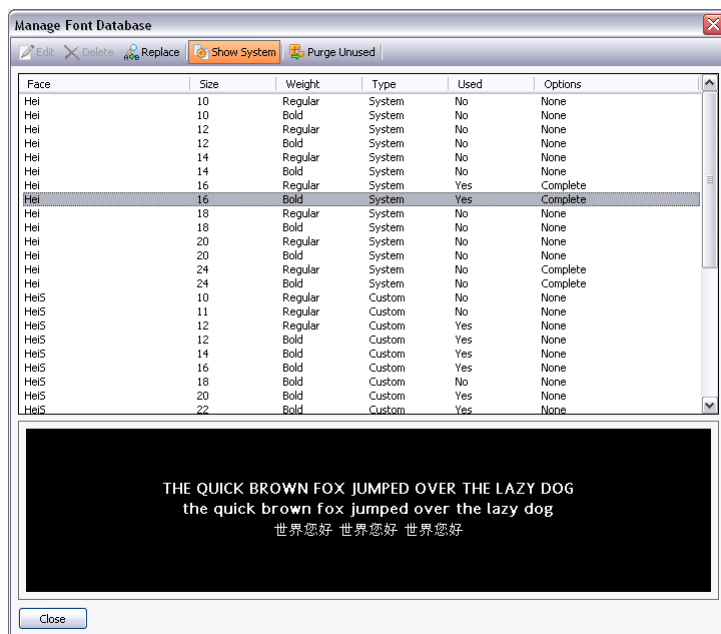
- **导出** 命令会将当前可用的但却未存储在磁盘里的图像保存至文件。如果已为选定的图像定义了文件名，则会使用该名称。否则，会提醒您选择文件名。
- **替换** 命令允许将一个给定图像替换为另一个图像。数据库里对该图像的全部引用也会被更新，从而反映更改。
- **导出全部** 命令会导出当前可用的但却未存储在磁盘里并且已定义了文件名的全部图像。此命令可用于确保关闭包含数据功能之前将全部图像存储在外部文件里。

- *清除未使用的* 命令用于移除未在数据库里使用的全部图像，从而在将数据库保存至磁盘时节省磁盘空间。使用此命令也可减少目标设备里的内存使用。

## 管理字体

可从用户界面设置的字体选项卡里调用字体管理器。它包含了数据库里引用的全部字体及其属性的列表。它允许查看字体，并对字体的存储和使用方式进行某些更改。

下面的示例显示了一个复杂数据库里的字体管理器：



主列表视图显示了各种字体的属性：

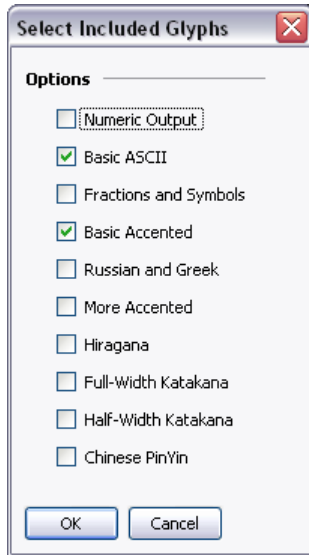
- *正面* 属性显示了字体的名称。
- *大小* 属性以像素为单位显示了字体的高度。
- *粗细* 属性指示了字体是否为粗体。
- *类型* 属性指示了字体是否为系统字体或自定义字体。
- *已使用* 属性指示了是否在数据库里使用了字体。
- *选项* 属性列出了为字体选定的选项。

窗口顶部的工具栏允许执行各种命令：

- *编辑* 按钮允许编辑自定义字体的属性。
- *删除* 按钮允许删除未使用的字体。字体一旦删除，就不会再出现在用于选择字体的下拉列表里面，但是可使用关联的选取按钮来重新创建。
- *替换* 按钮允许将一个字体替换为另一个字体。数据库里对该字体的全部引用也会被更新，从而反映更改。

- *显示系统* 按钮控制了是否在列表里显示系统字体。
- *清除未使用的* 按钮用于移除未在数据库里使用的全部字体，从而减少目标设备里的内存使用量。与删除的字体类似，清除的字体不会再出现在用于选择字体的下拉列表里面，但是可使用关联的选取按钮来重新创建。

编辑自定义字体的属性时会出现下列对话框：



各种选项允许在为目标设备创建的和下载的字体图像里包含特殊字符集。将字符限制为您的应用程序需要的字符，可以节省内存，对于较大字体而言尤其如此。请注意，数字输出选项可单独用于将字体限制为数字、小数点和其它用于呈现常规数字、科学数字或十六进制数字的字符。

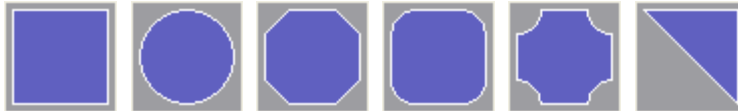


# 基元类型

本章描述了 Crimson 提供的每个基元。

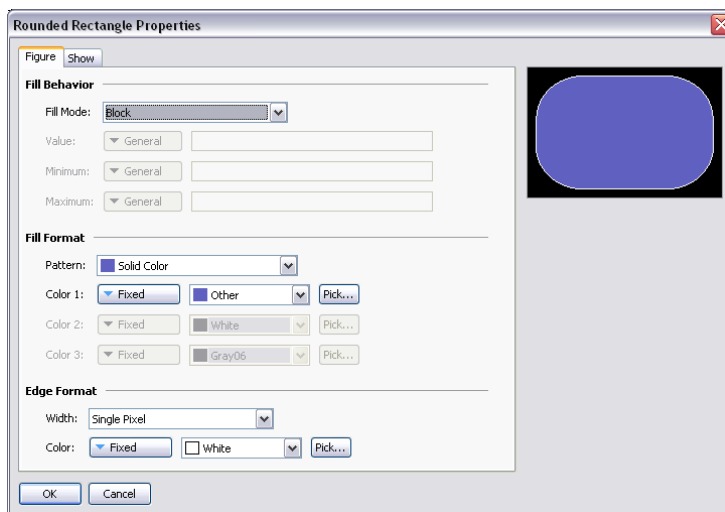
## 核心基元

### 几何基元



几何基元表示了简单的形状：矩形、圆形、裁剪矩形、圆角矩形、缺角矩形和楔形。所有这些基元均支持箱填充，因此可用于实现条形图之类的效果。它们还支持添加文本或数据，因此可用于创建文本显示或数据显示，或提供数据输入。最后，它们还支持添加操作，因此可用于实现交互式显示元素。

这些基元的基元特定属性选项卡如下所示：



请参阅上一章，详细了解标准填充和边缘设置。请注意，楔形拥有一个附加属性，亦即位置属性，用于指定边界矩形里的三角楔形的方向。

裁剪矩形、圆角矩形和缺角矩形都有布局控点，可用于指定边角效果的半径。它们在零半径简并形式下，均会变成简单矩形。

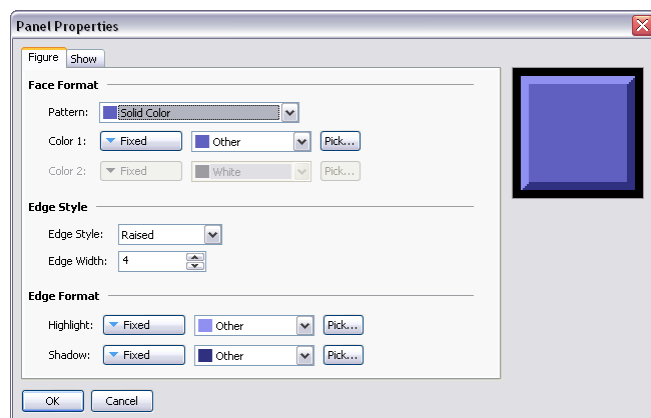
虽然几何基元都很简单，但是它们支持箱填充、数据、文本和操作，这就意味着，事实上仅使用矩形或圆角矩形就可创建大多数数据库的绝大部分。

## 三维基元



各种三维基元表示了拥有三维边框的矩形。虽然资源窗格里显示了三种版本，但是它们实际上全部只是简单矩形的预配置变体。这些基元支持箱填充，因此可用于实现条形图之类的效果。它们还支持添加文本或数据，因此可用于创建文本显示或数据显示，或提供数据输入。最后，它们还支持添加操作，因此可用于实现交互式显示元素。

这些基元的基元特定属性选项卡如下所示：



请参阅上一章，详细了解标准填充和边缘设置。*边缘样式* 选择了要绘制的边缘的类型，从而有效地在上面显示的三种预定义版本之间进行选择。*边缘宽度* 属性定义了要分配给每个边缘元素的像素数目。拥有边框边缘样式的基元的边缘大小是定义的边缘宽度的两倍。

与几何基元类似，三维基元支持箱填充、数据、文本和操作，可用于创建标准数据库的绝大部分。

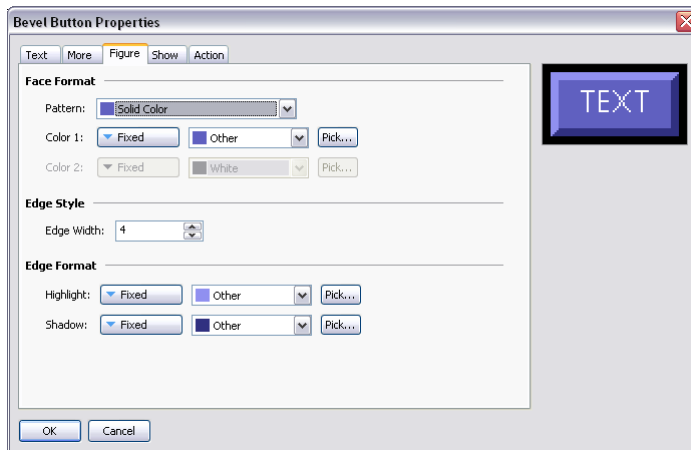
## 按钮基元



按钮基元实现了斜面按钮或渐变按钮。文本已预配置，允许对按钮进行标签，但是可以移除，从而允许添加实时数据。默认提供了一个操作选项卡，但是，如果为数据输入添加并配置了实时数据，此选项卡就会被禁用。拥有数据输入字段的按钮使用按钮按下动作来激活编辑。

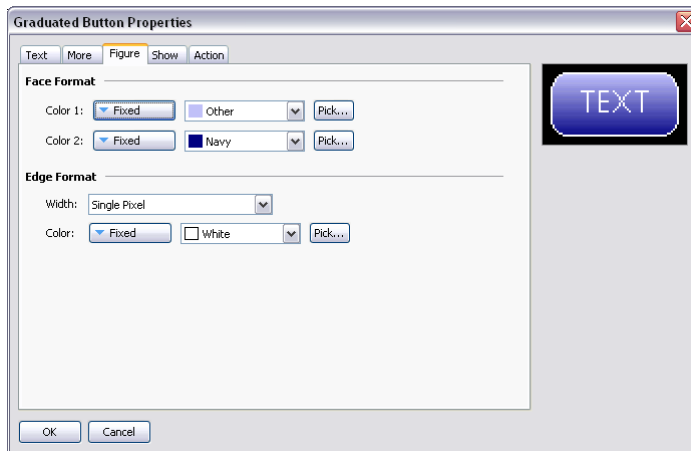


斜面按钮的基元特定属性选项卡如下所示：



请参阅上一章，详细了解标准设置。*边缘宽度* 属性定义了要分配给每个边缘元素的像素数目。因为此基元始终使用边框边缘样式，所以边缘大小始终是定义的边缘宽度的两倍。

渐变按钮的基元特定属性选项卡如下所示：



请参阅上一章，详细了解标准设置。

## 文本和数据基元



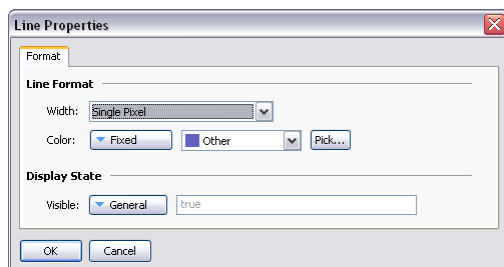
事实上，文本框和数据框就是拥有预定义的文本项和数据项且没有定义的填充和边缘颜色的矩形。它们之所以存在，就是为了使添加文本和数据元素更为简便，还为了向那些不习惯于使用简单几何基元来构造整个数据库的用户提供方便！它们还可用于向基元添加或构造分组时添加第二个数据或文本元素。

请参阅上一章，详细了解标准设置。

## 行基元



行基元实现了简单的行。属性对话框如下所示：



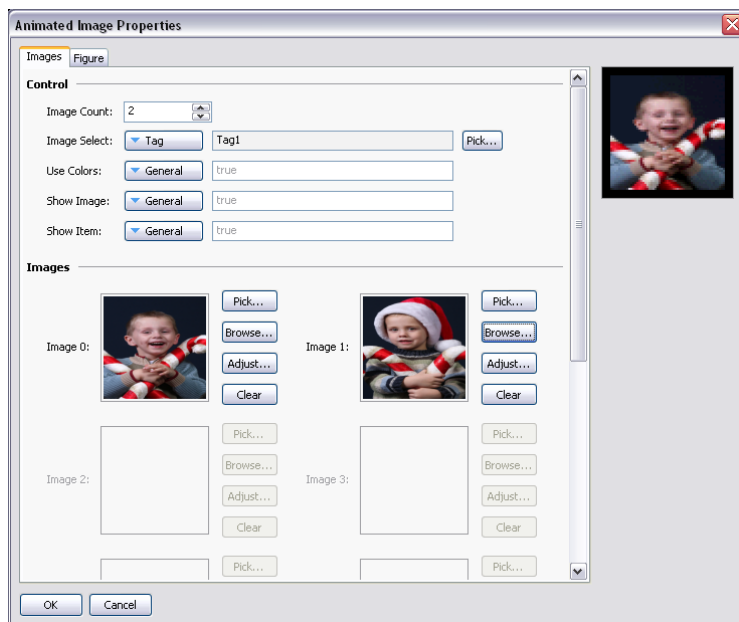
请参阅上一章，详细了解标准设置。

## 图像基元



图像基元用于显示可基于数字值从一系列图像中选择的图像。基元支持显示位图、JPEG、图元文件和多种其它图像类型，可使用透明背景或填充背景，可定义围绕图像周围的边缘，还支持添加数据、文本和操作，从而允许构造更为复杂的元素。

一个动态显示的图像基元的图像选项卡如下所示：



- *图像计数* 属性为此基元定义了图像槽的数目。基于图像选择属性的值，随时均会从这些图像中选择一个进行显示。

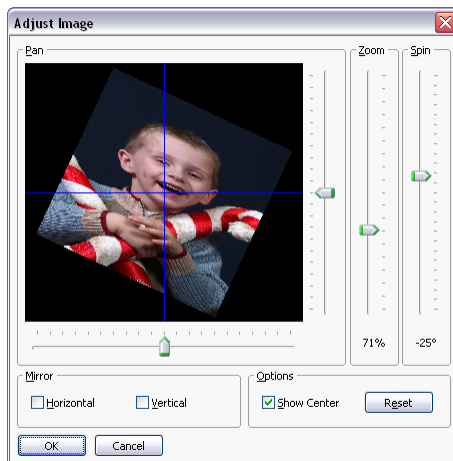
- *图像选择* 属性选择了所需的图像。它被视作基于零的值，以图像计数为模进行减少。换言之，如果定义了四个图像，则值 0、4、8 等将显示第一个图像，值 1、5、9 等将显示第二个图像，依此类推。
- *使用颜色* 属性用于将图像更改为黑白色或保留其本身颜色。求值为非零值的表达式或空表达式会使用彩色图像。零值会将图像更改为使用标准 RGB 亮度权重的灰度模式。显示按钮上的图像的禁用状态时，此选项十分有用。
- *显示图像* 属性用于显示或隐藏图像。如果没有为基元定义边缘或填充，则此属性功能上等同于 *显示项* 属性，否则，此属性将根据配置显示边缘或背景。
- *显示项* 属性用于显示或隐藏整个基元。

### 定义图像

对话框的*图像*部分为每个槽定义了图像。按下每个图像旁边的选取按钮均将显示一个对话框，提醒您可以简便地将图像拖入该字段。此图像可从资源窗格的符号库类别拖入、从 Windows 资源管理器的文件夹拖入，或从任何其它支持拖放的应用程序拖入。浏览按钮可用于打开含有合适的图像格式的文件，并将该文件加载入此槽。如上所述。支持 JPEG、图元文件、位图和多种其它文件类型。

### 调整图像

图像旁边的调整按钮可用于修改图像：



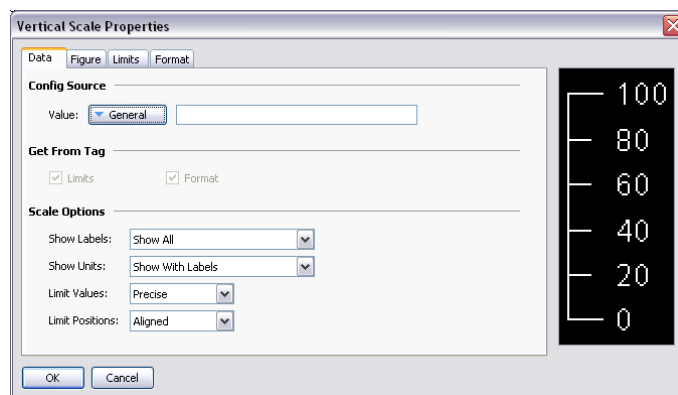
各种滑块可用于平铺、缩放和旋转图像，而水平和垂直勾选框则可用于对图像进行水平或垂直镜面显示。显示中心勾选框会显示或隐藏标出图像中心的蓝线，而重设按钮则可用于将图像恢复到原始状态。操纵选项有时用于修改图像，从而为动态显示创建各种不同状态。

## 刻度尺基元



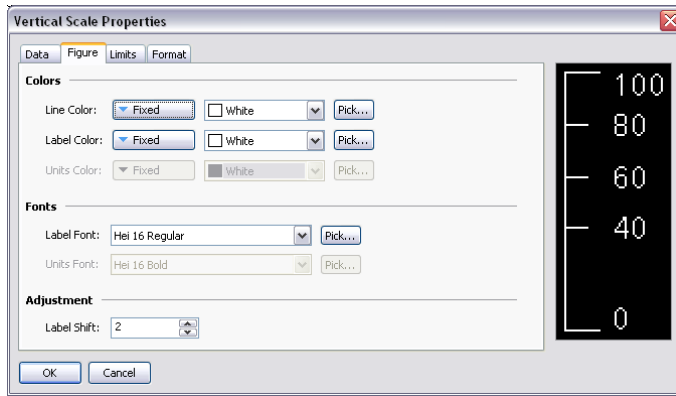
刻度尺基元用于绘制一个垂直刻度尺。刻度尺的限制可定义为常量，也可根据特定表达式而有所改变。可对刻度尺进行标签或不进行标签，而任何标签均基于可从标记获取的特定格式对象。

### 数据属性



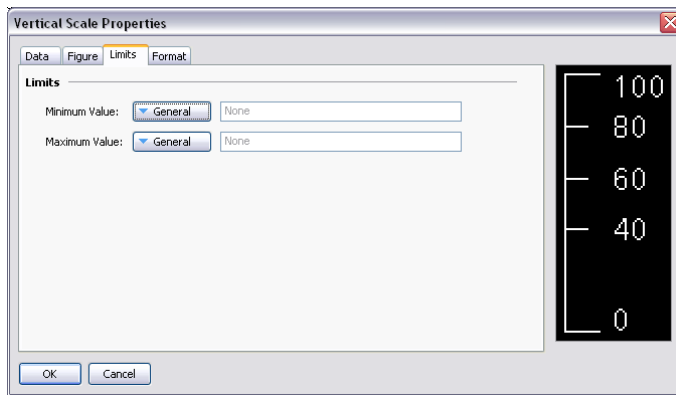
- **值** 属性定义了用于为刻度尺获取限制和格式信息的可选标记。值本身并不被基元使用，而标记则用作更多信息的源。
- **从标记获取** 属性用于指示在值属性内可选定义的标记是否应用作相关数据的源。
- **显示标签** 属性用于显示或隐藏数字刻度尺标签。
- **显示单位** 属性用于显示或隐藏数字数据格式定义的单位。单位可添加至每个刻度尺标签，也可绘制在刻度尺边缘。
- **限制值** 属性指定了如何确定刻度尺的顶值和底值。如果指定了精确设置，则将精确使用限制值，即使这会产生并不完全对应于自动选择的刻度线间距的限制。这可能会产生不规则刻度尺标签，但是会确保精确地按照刻度尺基元的需要绘制置于刻度尺旁边的箱填充和同一标记的边界。设置为圆角允许刻度尺基元自动调整限制，从而获取规则的刻度线间距。
- **限制位置** 属性指定了刻度尺的限制如何与单元标签进行关联。设置为对齐会将刻度线和标签准确对齐，但可能会将外侧的刻度线从基元边缘向内移动。设置为移位会将外侧的两个标签相对刻度线进行移动，但是允许最小和最大刻度线与基元边缘对齐，使其更易于与，比如说箱填充，对齐。

## 图属性



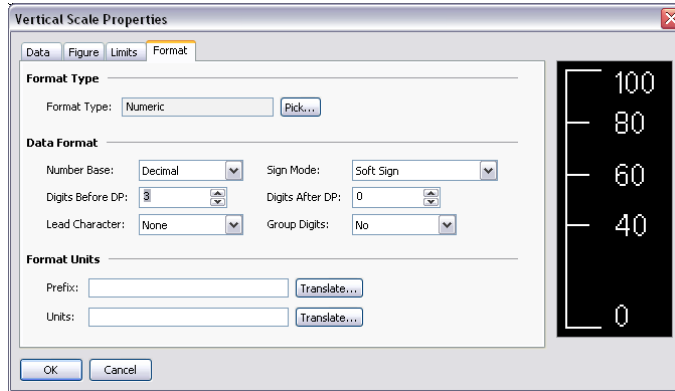
此页面上的属性定义了为刻度尺使用的颜色和字体。请参阅上一章，详细了解标准设置。**标签移位**属性可用于相对于刻度线向上或向下移动标签，使用在字符上面或下面添加间距的字体时，此属性可产生更为美观的效果。

## 限制属性



此页面上的属性用于定义要由刻度尺显示的最小值和最大值。可以指定表达式，这种情况下 **Crimson** 将在运行时动态更新刻度尺，选择适用于新值的刻度线和标签位置。如果选择了标记作为限制值的源，则这些设置可能会不可用。

## 格式属性



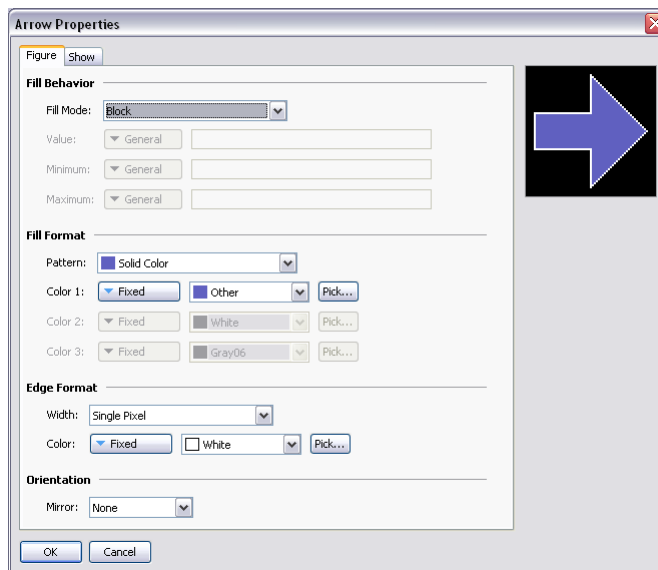
**格式类型** 字段指定了绘制刻度尺标签时要使用的格式类型。仅支持常规格式和数字格式。如果是从标记获取格式，则选择可能会不可用。请参阅使用格式章节，详细了解选择数字数据格式时显示的各种其它属性。

## 箭头



四个箭头基元事实上是一个简单基元的预定义版本。此基元支持箱填充。它还支持添加文本或数据，因此可用于创建文本显示或数据显示，或提供数据输入。最后，它还支持添加操作，因此可用于实现交互式显示元素。

基元特定属性选项卡如下所示：



请参阅上一章，详细了解标准设置。**镜面** 属性用于控制箭头的方向。正是这个属性用于生成资源窗格里显示四个预定义版本。

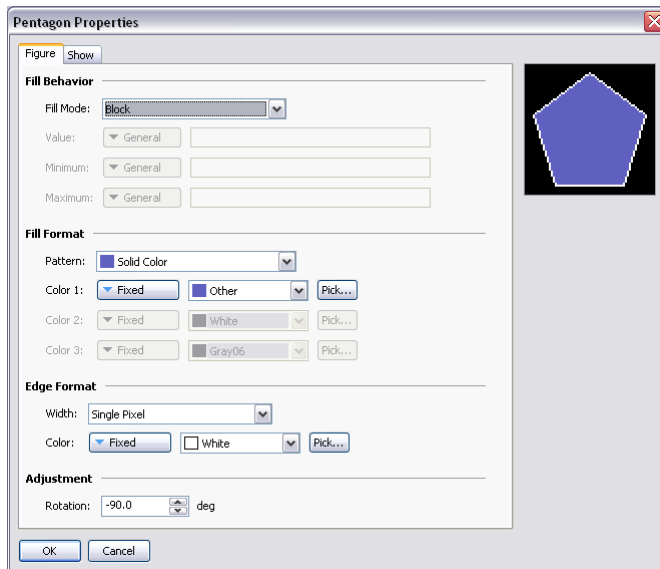
## 多边形和星

### 多边形



这些基元用于显示常规多边形：三角形、五边形、八边形和十边形。它们全部支持箱填充。它们还支持添加文本或数据，因此可用于创建文本显示或数据显示，或提供数据输入。最后，它们还支持添加操作，因此可用于实现交互式显示元素。

这些基元的基元特定属性选项卡如下所示：



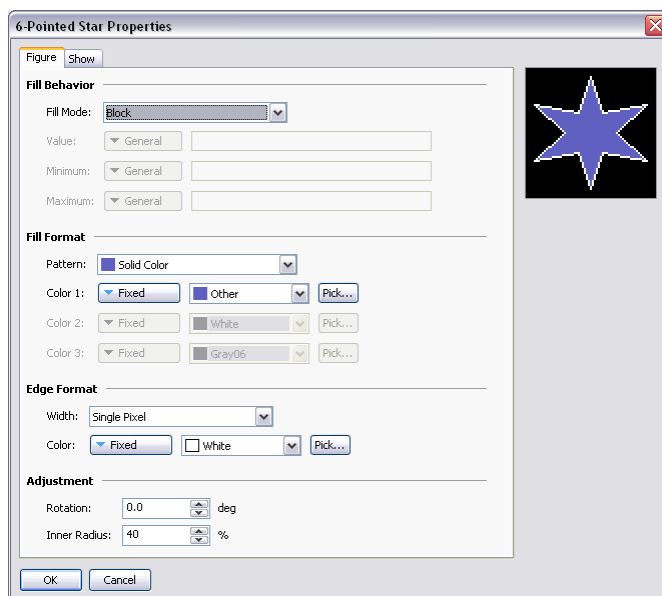
请参阅上一章，详细了解标准设置。*旋转* 属性可用于在边界矩形内旋转多边形。X 轴和 Y 轴均会被缩放，这样多边形的整体宽度和高度即可填满矩形。

### 星



这些基元表示了常规的四角星、五角星、六角星和八角星。所有这些基元均支持箱填充。它们还支持添加文本或数据，因此可用于创建文本显示或数据显示，或提供数据输入。最后，它们还支持添加操作，因此可用于实现交互式显示元素。

这些基元的基元特定属性选项卡如下所示：



请参阅上一章，详细了解标准设置。旋转属性可用于在边界矩形内旋转多边形。X 轴和 Y 轴均会被缩放，这样多边形的整体宽度和高度即可填满矩形。内半径属性用于更改星的棱角。（通过取一个有  $2n$  个侧面的多边形，然后在绘制多边形时更改其棱角，即可创建星。此属性控制了半径比。）

## 提示框与标注



提供的提示框基元可用于对页面上的项进行标签或为操作员提供帮助。它支持添加文本和数据，还可配置为显示箱填充。它还支持添加操作，因此可用于实现交互式显示元素。

提示框的精确样式通过若干布局控点进行控制：



左上角的控点控制了边角的半径。中心的控点控制了提示框主体相对于提示框尾部的高度。底部的控点控制了提示框尾部的位置。移动控点时，会自动重新调整提示框内的文本。



## 半裁剪图



半裁剪图是圆角矩形、裁剪矩形和缺角矩形的移除了两个边角的版本。创建标题栏和基元组边缘上的其它效果时，它们很有用。每个图均有四个可用方向。

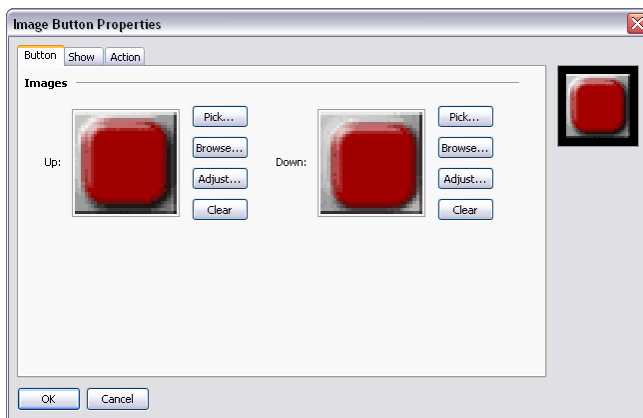
## 操作按钮



操作按钮使用从符号库中预选定的图像来创建按下时会执行给定操作的按钮。除了上面显示的之外，还提供了多种其它版本。单击资源窗格里的给定按钮会显示可用的不同颜色的变体。例如，正方形按钮可用红色、绿色或黑色：



使用操作按钮时，首先应使用属性对话框的操作选项卡来按照上文的描述定义操作。按钮选项卡可用于调整按钮图像或定义自定义版本：



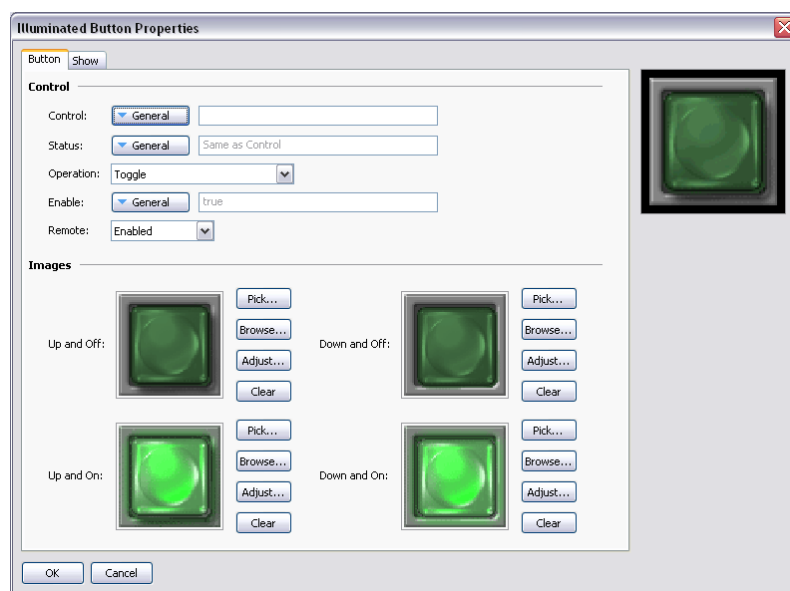
## 发光按钮



发光按钮使用从符号库中预选定的图像来创建控制标记的按钮，该按钮会根据标记的状态或其它表达式的状态而发光。除了上面显示的之外，还提供了多种其它版本。单击资源窗格里的给定按钮会显示可用的不同颜色的变体。例如，糖果按钮可用红色、绿色、黄色、蓝色或灰色：



这些基元的基元特定属性选项卡如下所示：



- **控制** 属性定义了按下或松开按钮时要写入的值。此值必须为可写，将根据为按钮定义的确切操作而设为一或零。
- **状态** 属性用于控制按钮发光。如果此属性留空，则会默认等于控制值。如果需要更为复杂的逻辑，则可使用不同的值。

- *操作* 属性选择了所需行为:

操作	按钮行为
扳钮开关	按下按钮时更改数据状态。
闭锁	如果数据是 0，则按下按钮时将其设为 1。 如果数据是 1，则松开按钮时将其设为 0。
常开瞬时	按下按钮时将数据设为 1。 松开按钮时将数据设为 0。
常闭瞬时	按下按钮时将数据设为 0。 松开按钮时将数据设为 1。
开启	按下按钮时将数据设为 1。
关闭	按下按钮时将数据设为 0。

请注意，相对于非零值设回为零的点而言，闭锁和扳钮开关稍微有些不同。按下按钮时，扳钮开关会进行全部更改，而松开按钮时，闭锁会将值关闭。这会产生与真实闭锁按钮的行为更为一致的结果。

请参阅上一章，详细了解*保护*、*启用*和*远程*属性。请参阅本章前面部分，详细了解如何更改或调整各种按钮图像。正如上面的示例中所看到的那样，需要四个图像来表示按钮的状态。

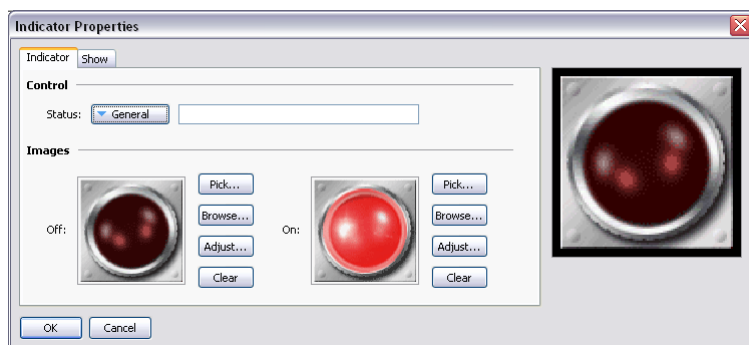
## 指示器



指示器使用从符号库中预选定的图像来显示数据值的开启/关闭状态。除了上面显示的之外，还提供了多种其它版本。单击资源窗格里的给定按钮会显示可用的不同颜色的变体。例如，指示开关指示器可用红色、绿色、黄色、蓝色或白色：



指示器的属性集非常简单：

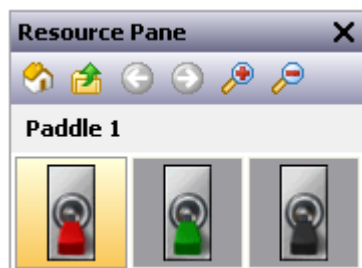


状态 属性控制了要绘制的图像。其它属性全部是标准属性。

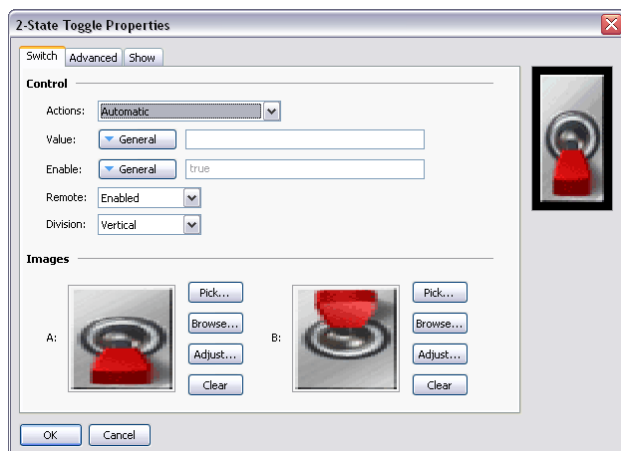
## 双状态扳钮开关



双状态扳钮开关使用从符号库中预选定的图像来实现拥有上下位置的扳钮开关。除了上面显示的之外，还提供了多种其它版本。单击资源窗格里的给定扳钮开关会显示可用的不同颜色的变体。例如，浆形开关可用红色、绿色或黑色：



## 开关属性

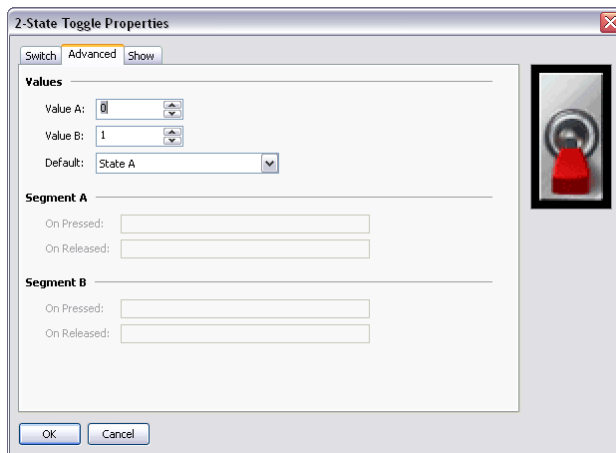


- *操作* 属性控制了开关的行为。三种自动模式以常规或有偏重的扳钮开关为模型，而用户定义模式则允许指定按下或松开扳钮开关的每半部分时会发生的更为复杂的操作。
- *值* 属性在自动模式下使用，开关更改时会被写入与状态 A 和 B 关联的数据值。默认设置下，零表示状态 A，一表示状态 B，但是，这些值可以使用此基元的高级设置进行更改。
- *分割* 属性定义了是否垂直扳动或水平扳动开关，进而定义了 **Crimson** 解释用户触摸时应如何分割基元。

请参阅上一章，详细了解*保护*、*启用*和*远程*属性。

请参阅本章前面部分，详细了解如何更改或调整各种开关图像。

### 高级属性



- *值 A* 和 *值 B* 属性定义了自动模式下用于表示开关的两个状态的数据值。从值属性读取的值将与这两个值进行比较，从而决定显示哪个状态，同样，更改开关会写入适当值。
- *默认* 属性选择了从值属性读取的数据与值 A 和值 B 均不匹配时要显示的状态。
- *按下时* 和 *松开时* 属性定义了用户按下或松开开关的 A 部分或 B 部分时要执行的自定义行为。对于垂直开关而言，A 是下半部分，B 是上半部分。对于水平开关而言，A 是左半部分，B 是右半部分。

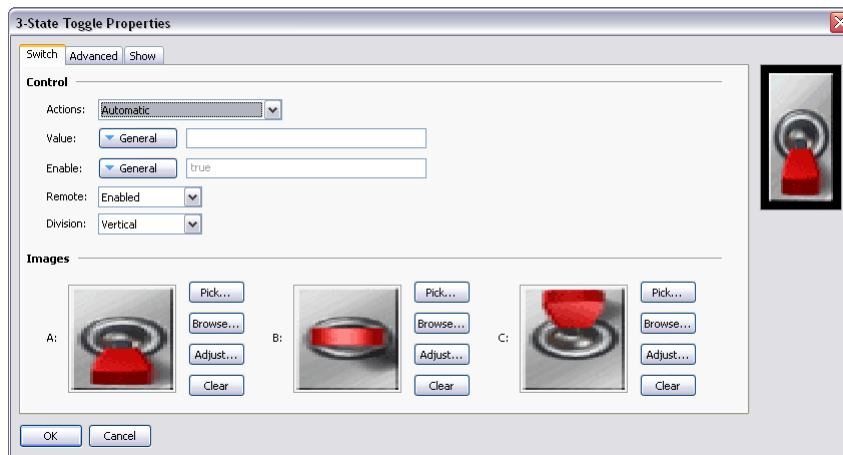
## 三状态扳钮开关



三状态扳钮开关使用从符号库中预选定的图像来实现拥有上中下位置的扳钮开关。除了上面显示的之外，还提供了多种其它版本。单击资源窗格里的给定扳钮开关会显示可用的不同颜色的变体。例如，浆形开关可用三种颜色：



### 开关属性

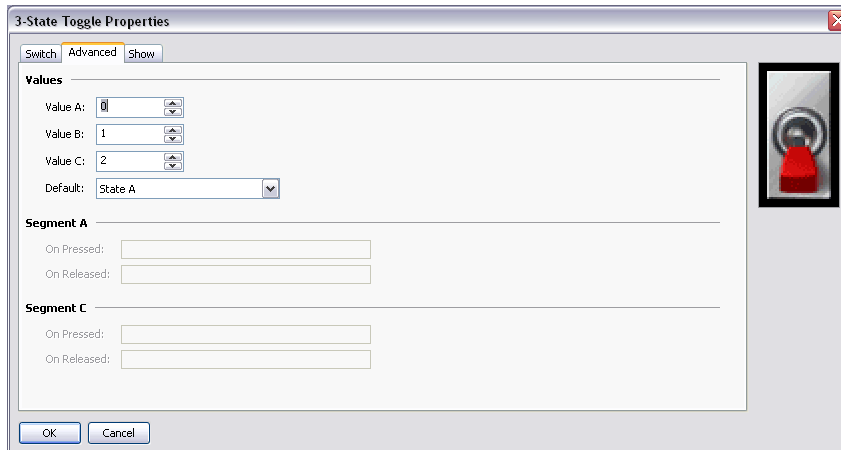


- **操作** 属性控制了开关的行为。四种自动模式以常规或有偏重的扳钮开关为模型，而用户定义模式则允许指定按下或松开扳钮开关的每半部分时会发生的更为复杂的操作。请注意，开关一次只能移动一个位置，因此，如果要从状态 A 移动到状态 C，就必须先移动到状态 B，这与真实的扳钮开关一样。
- **值** 属性在自动模式下使用，开关更改时会被写入与状态 A、B 或 C 关联的数据值。默认设置下，零表示状态 A，一表示状态 B，二表示状态 C，但是，这些值可以使用此基元的高级设置进行更改。
- **分割** 属性定义了是否垂直扳动或水平扳动开关，进而定义了 Crimson 解释用户触摸时应如何分割基元。

请参阅上一章，详细了解**保护**、**启用**和**远程**属性。

请参阅本章前面部分，详细了解如何更改或调整各种开关图像。

## 高级属性



- *值 A*、*值 B* 和 *值 C* 属性定义了自动模式下用于表示开关的三个状态的数据值。从值属性读取的值将与这些值进行比较，从而决定显示哪个状态，同样，更改开关会写入适当值。
- *默认* 属性选择了从值属性读取的数据与值 A、值 B 和值 C 均不匹配时要显示的状态。
- *按下时* 和 *松开时* 属性定义了用户按下或松开开关的 A 部分或 C 部分时要执行的自定义行为。对于垂直开关而言，A 是下半部分，C 是上半部分。对于水平开关而言，A 是左半部分，C 是右半部分。

## 双状态选择器开关



双状态选择器开关使用从符号库中预选定的图像来实现拥有两个状态的旋转选择器开关。它们的行为与双状态扳钮开关相同，事实上，它们是使用同样的基元来实现的。

## 三状态选择器开关



三状态选择器开关使用从符号库中预选定的图像来实现拥有三个状态的旋转选择器开关。它们的行为与三状态扳钮开关相同，事实上，它们是使用同样的基元来实现的。

## 旧版基元

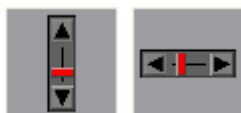
为了与其它软件程序包兼容，特提供了这些基元。

### 椭圆片段



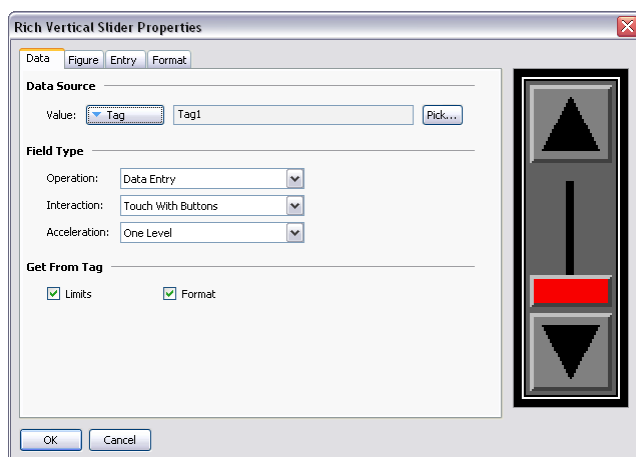
这些基元表示了一个椭圆的四分之一或二分之一。它们的属性是常规属性。

### 多信息滑块



多信息滑块基元允许通过模拟滑块来调整标记值。虽然它们可能非常有用，但是却可能会被新版 Crimson 里更为强大的基元代替，所以将它们归入了旧版子类别。

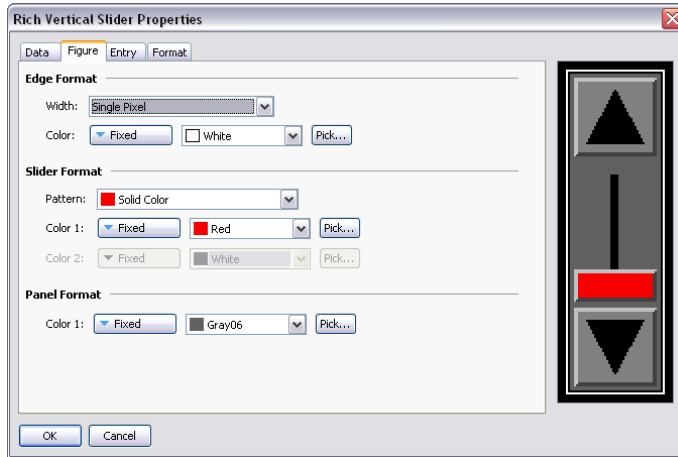
### 数据属性



- *值* 属性指定了要编辑其值的数据。
- *操作* 属性用于指示是否启用数据输入。默认值启用了输入，因为只读滑块容易出现误导。
- *交互* 属性指定了用户如何与基元进行交互，可通过按钮、通过直接操纵滑块或两者同时使用。
- *加速* 属性指定了数据输入过程中应提供几级加速。进行适当数目的步长之后，加速会逐渐更快地移动滑块。一级以上的加速可能会导致不经意间做出较大更改。
- *从标记获取* 属性用于指示是否从值属性里提供的标记来获取滑块限制和数据格式，或是否手动输入。

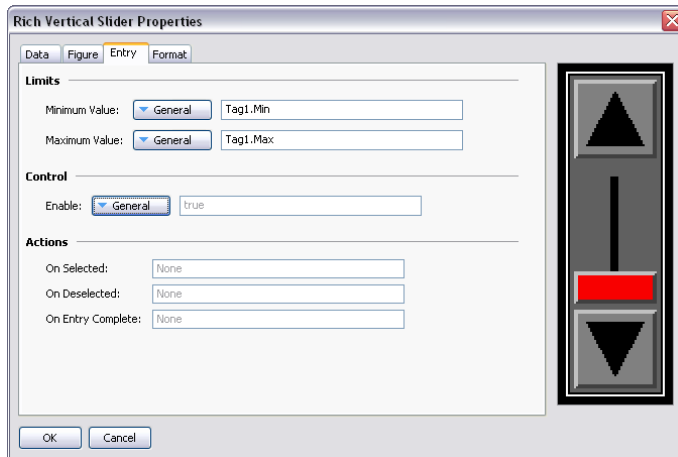


### 图属性



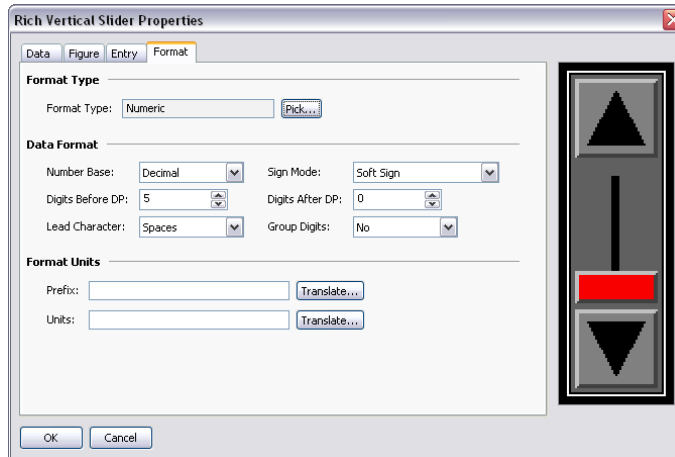
请参阅上一章，详细了解标准填充和边缘设置。

### 输入属性



请参阅上一章，详细了解标准数据输入属性。

## 格式属性

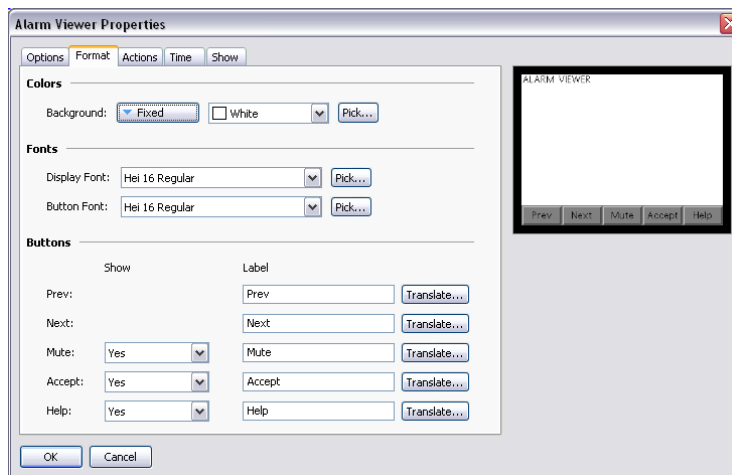


格式选项卡定义了基元使用的数据格式。既然实际上基元并不显示任何数据，所以您可能会思考为什么需要它，而答案正是加速：数据输入加速依赖于对正在编辑的数据的数基和任何小数点的位置的了解。其它设置则被忽略。请注意，如果是从控制标记获取格式，则格式选择可能会不可用。

## 系统基元

### 查看器格式

大多数系统基元都显示或操纵 **Crimson** 创建的或存取的数据。每个查看器都含有一个查看区域，查看区域下面有若干按钮。基于列表的查看器的外观通过属性对话框的格式选项卡进行控制：

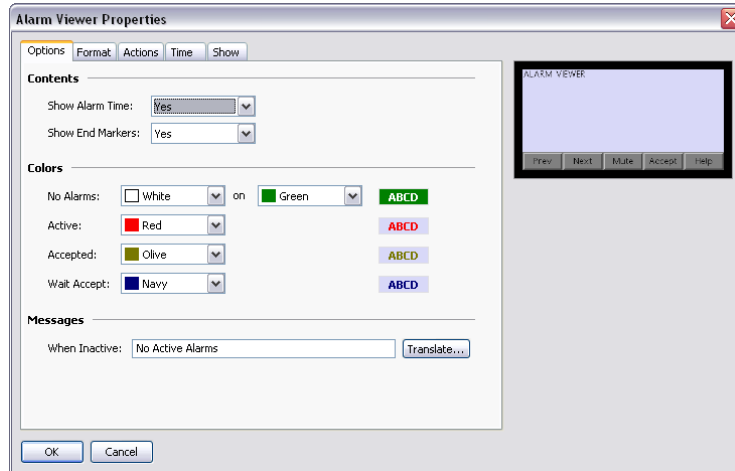


颜色和字体可按照常规方式指定。“按钮”允许禁用查看器底部的一些按钮，还允许对那些按钮的标签进行编辑或翻译，以供国际应用程序所用。请记住，可译字符串可设为表达式，这就意味着，按钮上的标签可在运行时进行自定义。

## 警报查看器

警报查看器用于在系统内显示和可选接受警报。

### 选项属性



- *显示警报时间* 属性用于指示是否应在每个警报前面添加发生时间和日期。要使用的具体时间格式在时间选项卡里指定。
- *显示结尾标记* 属性用于指示是否应在列表里包含标记，从而标识出第一个项和最后一个项，进而使用户更易于知晓他们是否位于列表开头或结尾。
- *颜色* 属性组指定了显示警报的不同状态时要使用的文本颜色。“无警报”消息允许为其定义专用背景色，而各种状态特定的颜色则使用基元本身的背景色。
- *不活动时* 属性定义了或可能翻译了没有活跃警报时要由基元显示的字符串。

### 操作属性

如果通过格式选项卡启用了查看器底部的帮助按钮，则 *帮助时* 操作定义了按下按钮时要执行的操作。

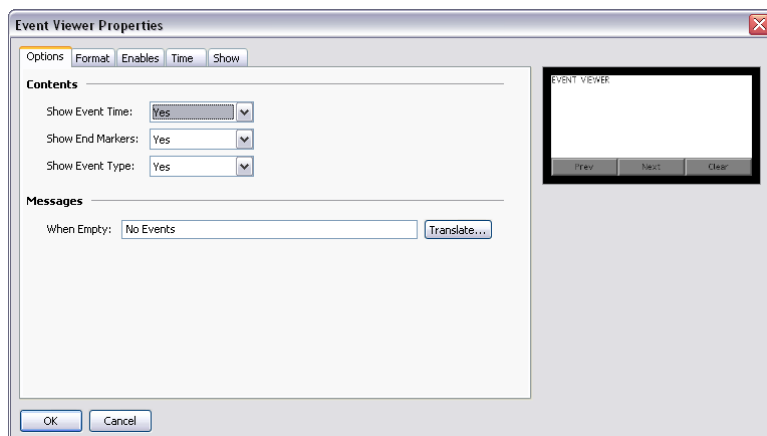
### 时间属性

时间选项卡定义了指示警报发生时间和日期时要使用的格式。详情请参阅使用格式章节。

## 事件查看器

事件查看器用于查看和可选清除系统响应数据标记生成的警报或事件时记录的事件。

## 选项属性



- *显示事件时间* 属性用于指示是否应在每个事件前面添加发生时间和日期。要使用的具体时间格式在时间选项卡里指定。
- *显示结尾标记* 属性用于指示是否应在列表里包含标记，从而标识出第一个项和最后一个项，进而使用户更易于知晓他们是否位于列表开头或结尾。
- *显示事件类型* 属性用于指示是否应对每次输入进行标记，从而指示其是否为警报发生、警报接受或警报清除，或其是否只表示一个简单事件。如果警报正在使用，未启用此设置则可能会产生较为混乱的显示。
- *为空时* 属性定义了或可能翻译了日志中没有事件时要由基元显示的字符串。

## 启用属性

如果通过格式选项卡启用了查看器底部的清除按钮，则*启用清除* 属性用于启用或禁用清除事件日志。

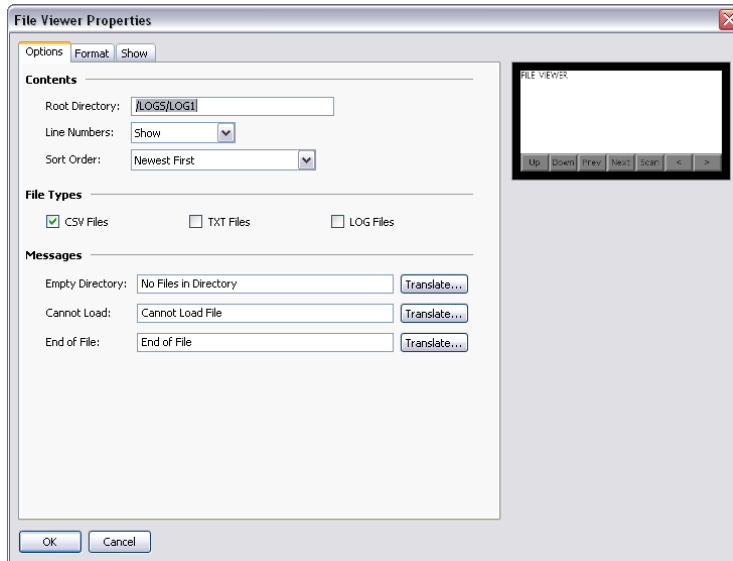
## 时间属性

时间选项卡指定了指示事件发生时间和日期时要使用的格式。详情请参阅使用格式章节。

## 文件查看器

文件查看器用于允许用户查看 CompactFlash 卡里的文本文件。

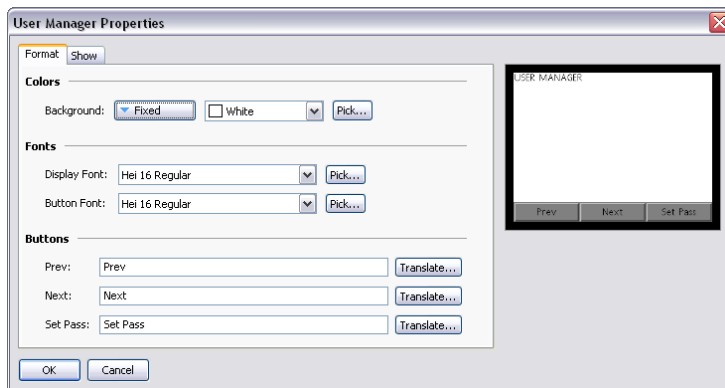
### 选项属性



- *根目录* 属性指定了要显示的目录。
- *行数* 属性用于在文件里显示或隐藏行数。
- *排序顺序* 属性用于指示如何访问文件。
- *文件类型* 属性组用于指示应设为可用以供查看的文件的类型。请注意，只可显示文本文件。
- *消息* 属性组定义了且可能翻译了文件查看器使用的各种消息。

## 用户管理器

用户管理器用于允许在运行时更改密码：

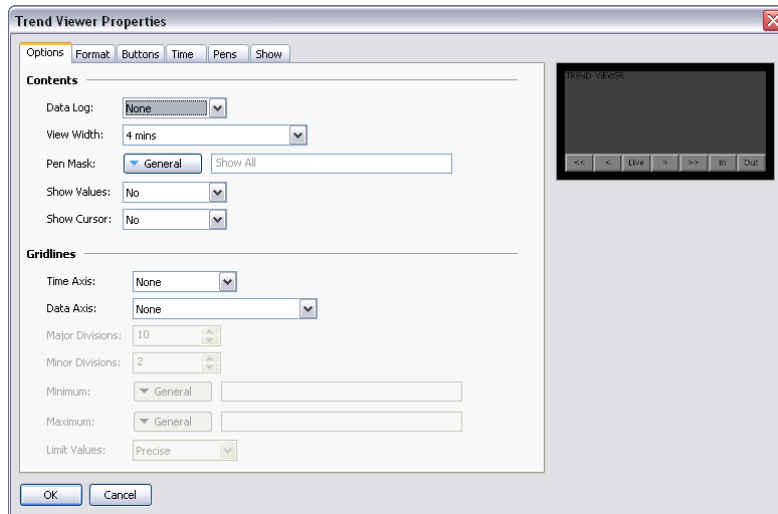


其核心属性包含在一个单一选项卡之内，全部是常规属性。

## 趋势查看器

趋势查看器允许显示来自数据记录器的信息。

### 选项属性



- **数据记录器** 属性选择了要显示的数据日志。
- **视图宽度** 属性用于指示要在窗口内显示的数据的初始量。用户随后可使用查看器底部的按钮进行缩小和放大。
- **笔蒙板** 属性用于提供一个 32 位整数值，从而选择性地启用或禁用显示特定通道。位 0 对应于数据日志的第一个通道，位 1 对应第二个通道，以此类推。值 1 会显示通道，值 0 则会隐藏通道。留空则会提供默认行为，显示全部通道。
- **显示值** 属性在实时模式下或使用光标前后滚动时启用或禁用显示与数据日志的每个通道关联的数据值。
- **显示光标** 属性用于启用或禁用在查看器里显示光标。用户可激活光标，从而允许精确确定特定时间点，并可选允许显示关联的历史数据值。
- **时间轴** 属性定义了是否应为时间轴显示网格线。Crimson 会根据查看器跨越的时间量自动决定网格线的间距。
- **数据轴** 属性用于为数据轴控制网格线显示。可指定仅主要分度或主要分度和次要分度，从而手动定义网格线；也可为数据轴指定最小值和最大值并由 Crimson 计算最佳网格线样式，从而自动定义网格线。
- **主分度** 和 **次分度** 属性定义了使用手动定义的网格线时要绘制的分度的数目。
- **最小** 和 **最大** 属性用于指示使用自动网格线时要显示的数据的范围。Crimson 会使用这些值来决定要采用的最佳网格线样式。数据值也会被缩放至这些值，这与数据值被缩放至其本身数据限制恰恰相反。

- *限制值* 属性指定了如何确定刻度尺的顶值和底值。如果指定了精确设置，则将精确使用最小值和最大值，即使这会产生并不完全对应于自动选择的刻度线间距的限制。这可能会产生不规则网格线间距。设置为圆角允许刻度尺基元自动调整限制，从而获取规则的刻度线间距。

### 格式属性

这些属性用于指定颜色和字体。它们的操作都是常规操作。

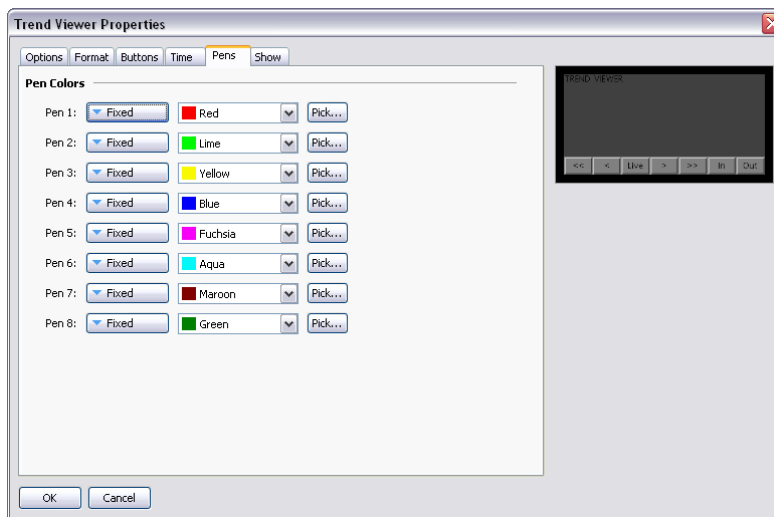
### 按钮属性

这些属性用于编辑和可选翻译各种按钮标签。

### 时间属性

时间选项卡用于设置提供与数据日志相关的时间和日期信息时要使用的时间的格式。详情请参阅使用格式章节。

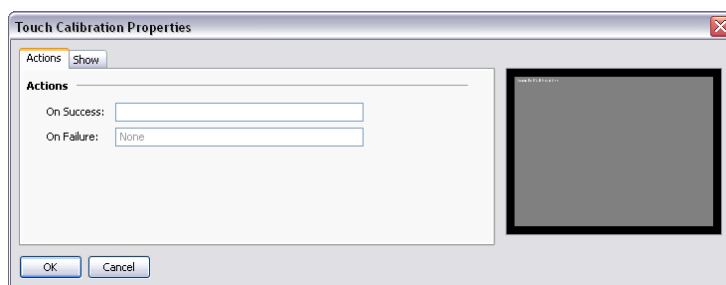
### 笔属性



这些属性用于指定绘制数据时使用的八种颜色。这些颜色循环使用，这样第九个通道就会返回到第一种颜色。不推荐绘制过多通道，因为这可能会产生非常混乱的显示。

## 触摸校准

触摸校准基元用于校准触摸屏：



其基元特定属性定义了校准成功和失败时分别要执行的操作。这些属性通常配置为返回主屏幕或从校准页面前进。

## 触摸测试器

触摸测试器基元允许用户检查触摸屏性能和校准。每次触摸都会在屏幕上产生一个点，并且显示有前面的触摸留下的痕迹。除了可见性控制之外，它没有其它可配置的属性。

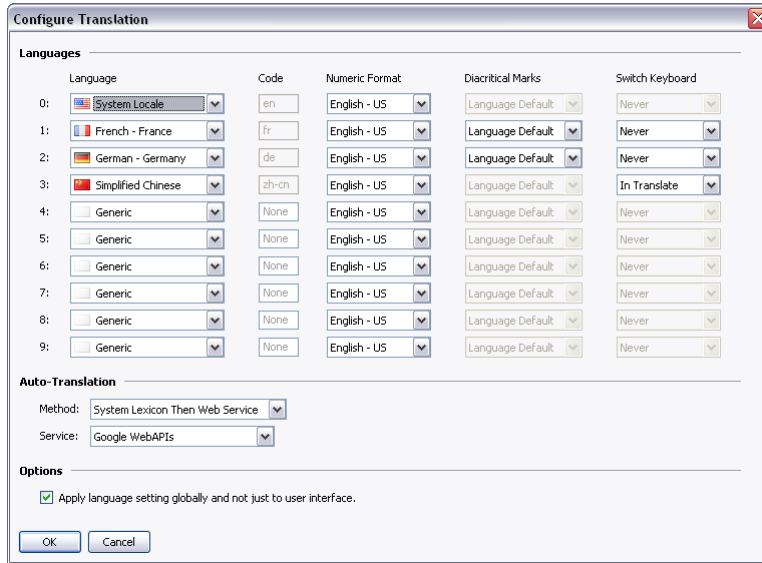


# 本地化

Crimson 3 支持一些本地化功能，这些功能允许对数据库进行修改，从而在多语言环境下使用。本章描述了如何使用这些功能，以及如何轻松创建可在全球范围内使用的数据库。

## 选择语言

创建多语言数据库的第一阶段就是配置要在项目内使用的语言。按下用户界面属性的全局页面上的配置翻译按钮，就会显示下列对话框：



对话框的顶端部分为每种语言定义了若干属性：

- **语言** 属性用于选择所需语言。根据使用国家，一种语言可能拥有多种变体。可为 Crimson 不直接支持的语言使用通用设置。
- **代码** 属性用于显示或输入选定的语言的两个字符的代码。此属性会在自动翻译过程中传递给 Web 翻译服务，并会用于定义词典文件的标题行。必须为 Crimson 事先并不知晓的通用语言手动输入代码。
- **数字格式** 属性用于定义 Crimson 是否应将数字设置为美国格式或当前所选语言的特定格式。数字格式选项包括使用逗号代替小数点，以及使用数字分组符号。
- **发音符** 属性用于重写语言在大写字符上的音符方面的默认设置。例如，法国法语（与加拿大法语相反）会在大写字符上应用音符，这可能会使使用某些字体时难以呈现这些字符。为此设置选择仅小写，会重写此默认行为。
- **切换键盘** 属性用于选择 Crimson 配置软件会将键盘布局切换为特定语言所用的布局的情况。使用翻译对话框时，无论是否正在使用该语言对文本进

行编辑，均会发生切换。默认为简体中文之类的语言启用翻译对话框里的键盘切换，从而确保调用适当的输入法编辑器。

下面的部分控制了自动翻译，详细描述如下。最下面的属性选择了是否将当前语言设置应用到 Web 服务器和数据记录器之类的服务，或这些服务是否应始终使用系统默认语言。

## 配置自动翻译

Crimson 拥有强大的自动翻译功能，帮助您修改您的数据库，以供国际使用。自动翻译包含两个组件，分别是系统词典和基于 Web 的翻译服务。

系统词典是一个 Unicode 文本文件，里面包含了工业自动化和流程控制领域使用的许多标准单词和短语，以及它们在若干种常用语言里的翻译。翻译过程中可参阅该词典，这样就可以极为迅速准确地翻译常见的多次出现的文本。

基于 Web 的部分则会使用两种服务中的一种。Google WebAPIs 通常提供更为快速的翻译，因为它不受带宽限制的限制。相反，Microsoft Translator 提供更为准确的翻译，但是它不够快速，因为它会限制每分钟可以提交的次数。可从上面图示的配置翻译对话框里选择一种服务。

自动翻译可配置为使用一种或两种方法。如果拥有 Internet 连接，则通常最好先使用词典，然后使用基于 Web 的服务。在某些情况下也可只使用词典，从而避免出现基于 Web 的服务可能产生的误译。

## 翻译您的数据库

可使用若干方法来完成数据库翻译。

## 输入翻译

按下数据库里每个可译字符串旁边的翻译按钮，即可进行手动翻译。会显示一个对话框，允许输入翻译文本，或只为此字符串调用自动翻译：



像这样的局部自动翻译允许您检查翻译的准确性。

## 全局自动翻译

文件菜单里的实用工具子菜单含有一个为数据库里的每个字符串应用自动翻译的命令。执行此命令时可能需要一些时间，如果使用的是限制带宽的翻译服务，则更是如此。使用全局字符串时，应格外小心，因为如果系统词典里不存在的字符串包含了技术术语或行业特定的行话，那么就可能会出现误译。

## 导出与导入

实用工具子菜单还含有导出和导入数据库里含有全部可译文本的文本文件的命令。这些文件可以使用 Microsoft Excel 之类的应用程序进行编辑，从而允许手动输入翻译。使用第三方翻译服务时，此功能尤其有用。文件格式包含了若干指示每个字符串的源的列，允许根据上下文对不同情况下的同一给定字符串进行不同翻译。

## 应用词典

除了上文描述的系统词典之外，您还可以从零开始或使用实用工具子菜单里的导出词典命令来创建您自己的词典。词典文件是 Unicode 文本文件，文件标题行含有制表符分割的语言代码，这些代码就是配置翻译对话框里定义的代码属性里使用的代码。标题行下面，每一行都含有定义的每一种语言里的一个单词或短语。

下面显示了词典文件示例：

en	fr	de
one	un	eins
two	deux	zwei
three	trois	drie

请注意，除了始终使用大写的特定术语（如德语名词等）之外，输入的文本始终应使用小写。使用小写可以允许 **Crimson** 形成自身的大写并标注大小写变体。

## 预览翻译

在从工具栏里的标志图标打开的下拉菜单中选择适当语言，即可在图形编辑器里预览翻译。对文本进行的任何直接编辑也会应用到当前选定的语言，其它语言则不会被更改。在对话框里编辑会继续被限制为默认语言，从翻译按钮访问的其它语言则如常。

## 切换语言

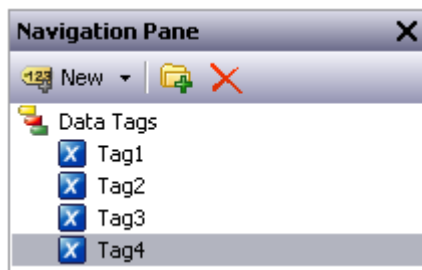
目标设备使用的语言通过调用 `SetLanguage()` 函数进行控制，函数的参数是 0 到 9 之间的选择所需选项的数字。例如，在上面的示例中调用 `SetLanguage(1)` 会选择法语，而自定义操作 `SetLanguage(2)` 则会选择德语。可使用 `GetLanguage()` 函数来确定当前语言。

# 使用小组件

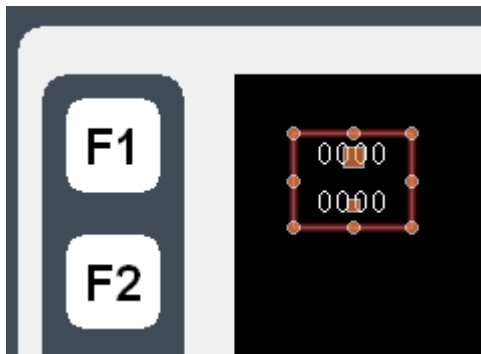
Crimson 3 支持一项强大的新功能，能够将普通基元分组转换为叫做小组件的功能强大的整体。除了组件基元之外，小组件还包含有用户可定义的数据项，这些数据项可在分组级别进行编辑，但是会被小组件的组件基元引用。本章解释了如何创建和使用小组件。

## 创建小组件

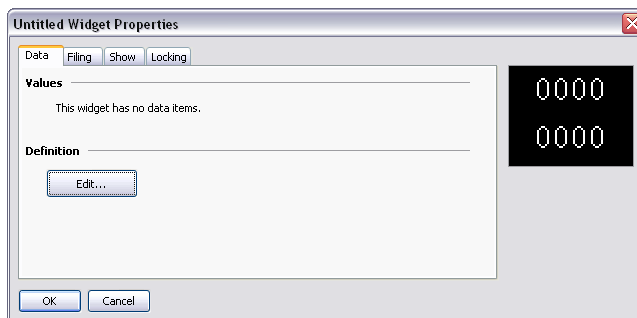
要理解小组件，最简单的方法就是创建一个小组件。让我们从创建一个空数据库然后添加四个数字标记开始。将标记的属性留为默认设置，会生成四个内部整数值，亦即 Tag1 到 Tag4。



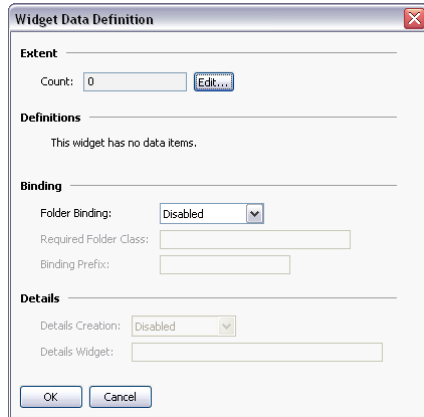
切换到显示页面部分，然后向页面添加两个数据框基元：



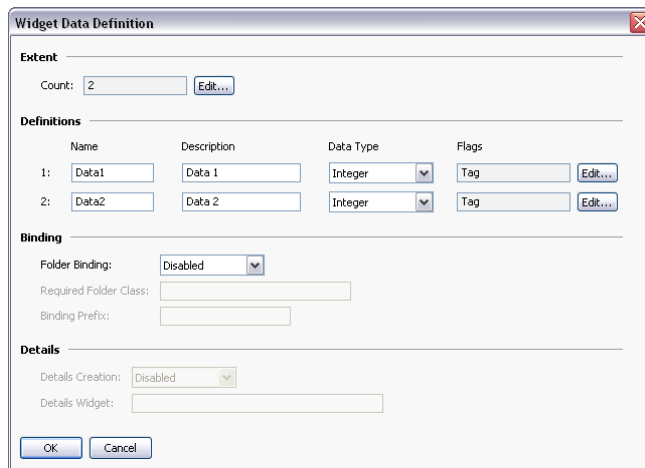
现在先将它们的属性留为默认值，然后选择这两个项。右键单击所选内容，然后从上下文菜单中选择完美命名的“小组件化”命令。这两个项会被绑定到一个分组，还会出现下列对话框：



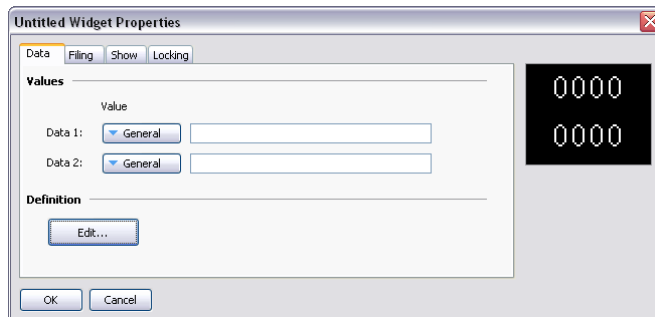
小组件创建之后，此对话框将用于编辑小组件的数据项，但是现在我们还没有定义任何内容。单击定义部分的编辑按钮，允许我们定义一些数据项：



单击计数字段旁边的编辑按钮会让我们创建两个属性：



如上所示完成数据字段，请注意，一定要正确设置数据类型，并将标志字段修改为指示每个数据项均应是标记。（可使用标志字段旁边的编辑按钮来编辑标志字段。）按下确定，关闭对话框。请注意小组件本身现在如何在自己的属性对话框里显示数据项：

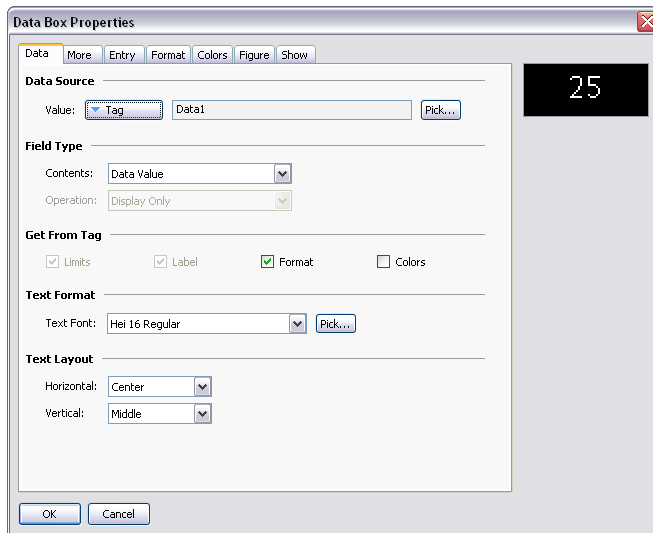


现在请先忽略这些，按下确定，关闭此对话框。

小组件仍应在图形编辑器里选定，所以请单击小组件里包含的数据框之一，进入分组编辑模式。请记住，绿色矩形标示出了我们正在编辑的分组，红色矩形显示出了该分组内选定的项：



双击数据框，打开其属性：

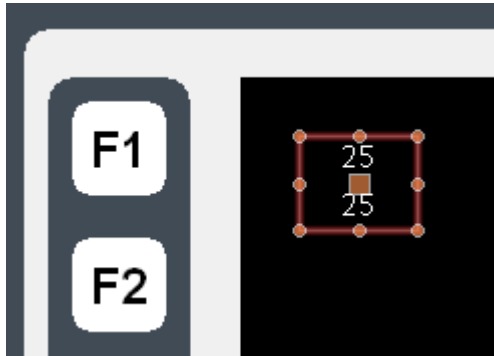


在值字段输入 `Data1`，并注意结果。

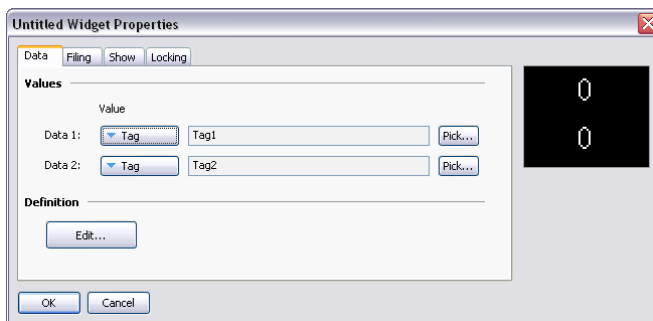
`Crimson` 会接受这作为标记名称，即使我们的数据库里事实上并没有叫做 `Data1` 的标记。此值实际等于小组件里定义的数据项之一，将表示我们返回并编辑小组件数据时分配的任何标记。（预览窗口里显示的值 `25` 是为未映射至任何内容的小组件数据项使用的默认值。）既然 `Data1` 被标示为标记，所以我们可以访问它的属性，将它用作格式信息的源，或对其进行任何可以对标记进行的所需操作。

为第二个数据框重复此步骤，这次将其值属性设为 `Data2`。

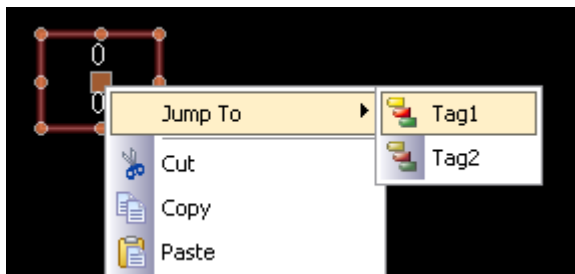
按下 **Esc** 键，直至只选定了小组件。如果您操作过度清除了选择，只需单击小组件即可，并确保它周围出现一个红色矩形：



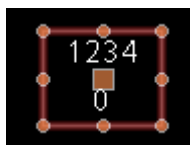
现在打开小组件属性，这次为数据项输入值：



如上所示输入值，将数据项分别设置为 `Tag1` 和 `Tag2`。请注意现在预览如何显示为值 0，因为小组件里的数据框现在分别是从 `Tag1` 和 `Tag2` 获取数据。如果想要使事情变得更加有趣，那么可以右键单击小组件，然后打开跳转菜单：

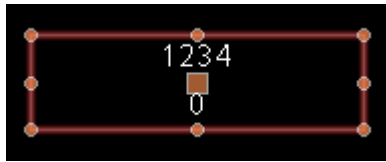


选择 `Tag1`，跳转至该标记，然后在模拟为属性里输入值 1234。使用 **ALT+LEFT** 组合键或工具栏上的后退按钮，请注意小组件如何继续追踪标记数据：

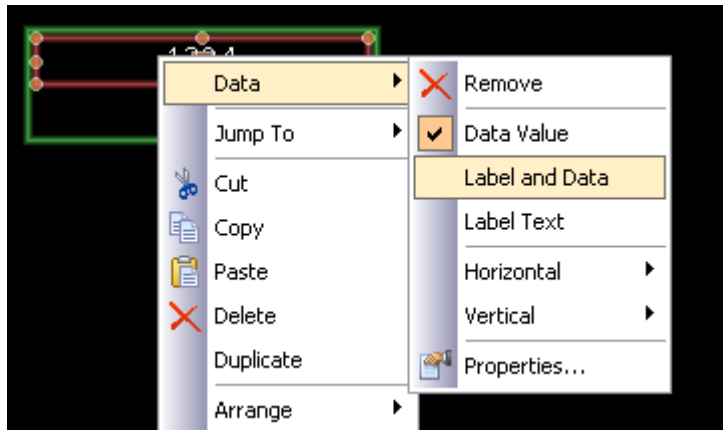




接下来，抓住右侧的控点，然后稍微拉宽小组件：



左键单击这两个数据框的上部，进入分组编辑模式，然后右键单击同一数据框，打开其上下文菜单：



选择数据子菜单，然后选择标签和数据命令，将此数据框配置为显示标记标签及其数据值。请注意小组件的新外观：



正如您所看到的那样，数据框显示的是来自 Tag1 的标签，指示了我们在数据框的值属性里输入的 Data1 的值完全对等于数据项随后被配置为的标记。我们将把小组件项设置为标记的流程称作将该数据项绑定至该标记。可以使用更为复杂的方式来进行绑定，这些方法我们稍后会看到。

## 总结

现在让我们概括一下我们进行的操作：

- 我们将基元放入显示，然后将它们分入一种叫做小组件的特殊分组。在编辑和其它方面，小组件的操作方式与普通分组类似，但是，小组件还有一些附加属性。
- 我们为小组件编辑了数据定义，创建了两个数据项，为每个数据项分配了名称、描述、数据类型和若干标志。
- 我们使用分组编辑编辑了小组件的内容，将它们的属性设置为小组件自己的数据项，使用数据项名称来引用它们。

- 我们修改了小组件的数据项，将它们绑定至标记，进而为我們的小组件提供真实的标记及其关联信息。

## 这为何重要

那么小组件为什么很重要？我们本可以轻松地创建数据框并将其直接绑定至标记，那么我们何苦还要进行这些额外的步骤呢？您试图创建更为复杂的小组件时，答案就会变得很明显：

- 小组件允许数据项在多个地方使用，小组件里的多个元素依赖于同一个标记，而无需在多个地方选择标记名称。
- 小组件可以封装复杂设计和功能，允许在多个数据库间对其进行复制和重复使用。事实上，小组件允许用户创建复杂基元。
- 小组件可保存至磁盘，也可添加至资源窗格，或通过电子邮件进行分发，进而使 Crimson 用户能够更轻松地和其用户或技术支持进行合作。

## 细节

下一部分会更为详细地重新描述上面的主题。

还描述了一些能够使小组件更加强大的魔法。

## 小组件数据定义

赋予小组件强大功能的正是它们的数据项。只需打开小组件的属性，单击数据页面里的定义部分的编辑按钮，即可编辑小组件的数据定义。

	Name	Description	Data Type	Flags	
1:	Data1	Data 1	Integer	Tag	Edit...
2:	Data2	Data 2	Integer	Tag	Edit...

- **范围** 属性定义了此小组件需要多少数据项。此值可随时更改，但是将其缩小会导致数据项及其值丢失。最多可定义八十个数据项。
- 每个数据项的 **名称** 属性均用于从小组件里包含的基元引用该项。因此，其必须满足标记名称的全部要求，不能包含空格和标点，必须以字母打头。

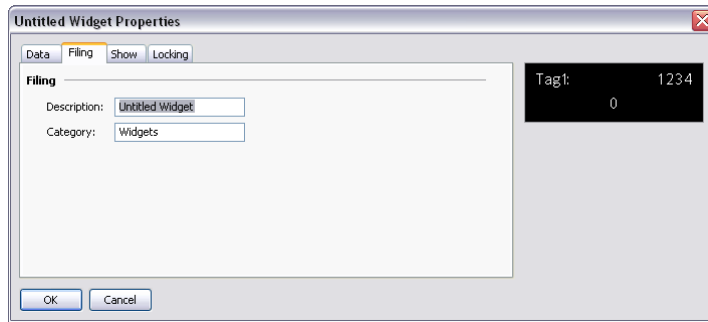
- 每个数据项的 *描述* 属性均用于提供更为友好的名称版本，这次则是为了显示在数据项编辑对话框里。此字段的内容设置没有限制。
- 每个数据项的 *数据类型* 属性均定义了所需的数据类型。数据项在小组件的属性对话框里的显示方式依赖于所选设置。真实数据类型、整数数据类型和字符串数据类型对应于表达式值，而颜色数据类型、页面数据类型和操作数据类型则允许创建更为复杂的项。页面和操作项可被视为来自小组件的基元内部的页面名称和程序。
- 每个数据项的 *标志* 属性均用于修改数据类型为真实、整数或字符串的项。它支持下列设置：

设置	描述
标记	为数据项输入的值必须为标记。小组件里的基元可将数据项视为标记，并可访问其属性、数据格式等。
可写	为数据项输入的值必须为可写。同样允许小组件里的基元向数据项进行写入。
数组	为数据项输入的值必须为数组的名称。小组件里的基元会将数据项视为数组，并且必须使用索引操作符来访问单个值。
元素	为数据项输入的值必须为数组元素。小组件里的基元会将数据项视为元素，并且会可以将其传递至需要此类型的参数的函数。
未绑定	<b>Crimson</b> 不会向此属性应用文件夹绑定，从而允许其用于存储预定义值，而不会由绑定而生成错误。文件夹绑定详情请参阅后面章节。

- *绑定* 属性组用于控制叫做文件夹绑定的高级功能。下文会详细介绍此功能。
- *详情* 属性组用于控制叫做详情页面创建的高级功能。下文会详细介绍此功能。

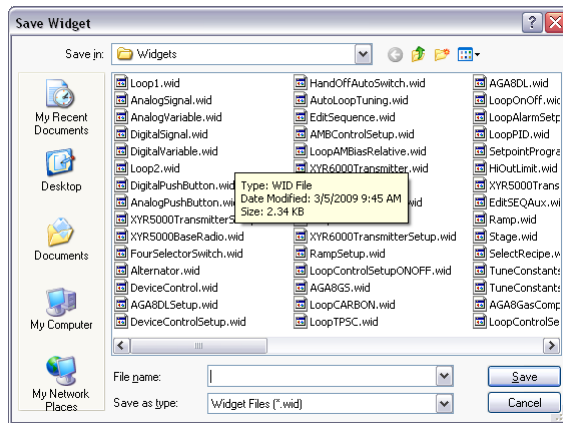
## 小组件存档

每个小组件在其属性对话框里均有一个存档选项卡：



*描述* 和 *类别* 属性用于控制小组件保存之后如何在资源窗格里进行显示。同一类别的全部小组件会被分在基元类别里的同一个子类别之内，用户悬停在项上时，即会显示小组件描述。

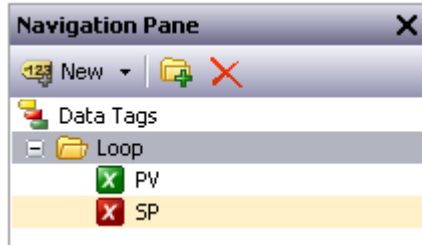
要保存一个小组件，只需选定它，然后从编辑菜单里选择保存小组件，或按下 **CTRL+Q** 组合键。会打开一个标准的保存对话框，允许将小组件保存为 **Crimson** 小组件目录里的 wid 文件：



只要向此目录添加了小组件，资源窗格就会自动进行更新。无论是通过 **Crimson** 进行了更改，还是通过 **Windows** 资源管理器直接将 wid 文件放入目录，均会发生更新。请注意，小组件文件是独立文件，可在安装在不同机器上的 **Crimson** 之间进行传递。这提供了一种功能强大的机制，可用于共享用户界面元素，或用于多人共同从事同一项目时与其它工程师交换项。

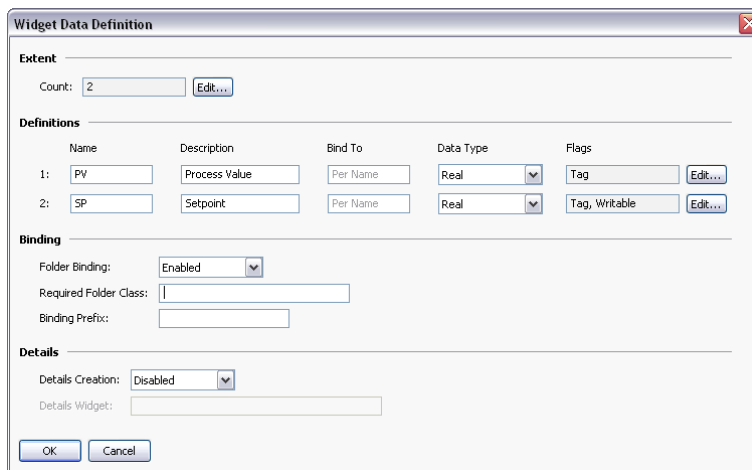
## 文件夹绑定

**Crimson** 能够将标记组织进文件夹，这使得一种面向对象的设计成为可能，通过这种设计，可将表示一个对象的属性的不同标记分入一个表示该对象本身的文件夹。考虑一下下面的示例：



这里创建了一个文件夹来表示一个 PID 循环，创建了若干标记来引用该循环的流程值和设定值。标记用代码 `Loop.PV` 和 `Loop.SP` 来指代，并使用了在使用嵌套项方面的标准 **Crimson** 规则。

文件夹绑定允许创建一个反映已在标记内创建的对象和属性结构的小组件。考虑一下下列数据定义：



这里我们创建了两个数据项，它们的名称与构成一个 PID 循环的标记的名称匹配。我们为它们提供了人工可读的名称，还将这两个数据项的标志设为标记。我们还将设定值定义为可写。请注意，这一次每个数据项都出现了一个叫做**绑定至**的新属性——讨论高级文件夹绑定时我们会回到这里。

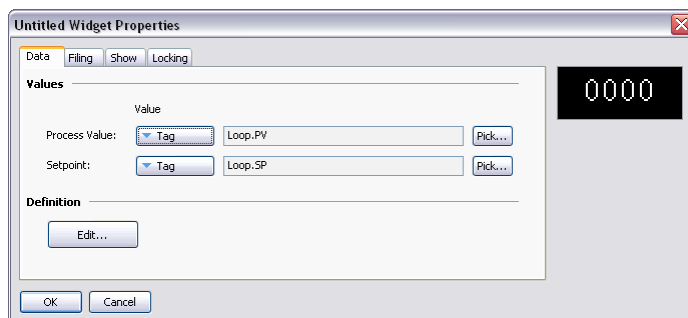
在绑定部分，我们启用了文件夹绑定。这指示了我们希望 **Crimson** 支持将全部数据项自动绑定至来自一个单一源文件夹的标记。保存这些更改并选择小组件的上下文菜单之后，我们将看到一个叫做**绑定小组件**的新命令，此命令允许执行绑定操作。

选择该命令或按下 **CTRL+B** 组合键，就会显示下列对话框：



如果我们从资源窗格将 Loop 文件夹拖放至目标，那么小组件的数据项就会被自动绑定至该文件夹内的相应标记。

打开小组件的属性会显示结果：



换言之，每个数据项都被绑定至了选定文件夹里名称与其自身数据项名称相等的标记。请想象一下这有多么强大：您可以定义多个属性，然后将它们绑定至一个单一操作，从而减少设计时间，并且可以更好地使用之前设计的项。

## 高级绑定

文件夹绑定支持若干高级选项。

### 类匹配

第一个也是最简单的一个就是小组件属性里的 *所需文件夹类* 设置。这可用于限制绑定期间将接受的文件夹，从而避免何种量和不同对象类型之间的不匹配。小组件上的指定类必须匹配要绑定的文件夹上的类，否则就会发生错误。

### 绑定前缀

*绑定前缀* 属性可在嵌套小组件时用于允许将子小组件绑定至父小组件绑定至的文件夹的子文件夹。例如，假设创建一个双循环小组件，它将绑定至一个文件夹，该文件夹包含两个 PID 文件夹，分别叫做 Loop1 和 Loop2。通过将每个子小组件的绑定前缀设置为循环名称之一，就可以确保它们被绑定至拖放到父小组件上的文件夹中的不同子文件夹。例如，如果第一个子小组件的绑定前缀是 Loop1，它的父小组件绑定至了一个叫做 Dual 的文件夹，那么该子小组件的属性就会分别绑定至表达式 Dual.Looped.PV 和 Dual.Looped.SP。

## 使用绑定至

数据项的 *绑定至* 属性可用于修改数据项绑定至的表达式。最简单的选项就是输入一个与数据项的名称不同的名称，这种情况下，将使用该名称来选择要绑定至的标记。

### 使用周期

还可以输入包含周期的名称。这些会在源文件夹的子文件夹里选择标记。例如，输入 `Remote.SP` 会在绑定至 `Loop` 文件夹时将相关数据项绑定至表达式 `Loop.Remote.SP`。

### 使用脱字符号

要升序文件夹树，可为名称添加脱字符号，一个脱字符号会往上移动一级。一个绑定至 `Dual.Loop` 的小组件内的绑定至属性设置为 `^Name` 的数据项会绑定至表达式 `Dual.Name`。

### 特殊名称

还可以使用若干特殊 *绑定至* 名称之一：

名称	结果
<code>::Path</code>	此小组件绑定至的标记的完整路径，包括任何父文件夹。
<code>::Name</code>	此小组件绑定至的 标记的名称，不包括任何父文件夹。
<code>::TopPath</code>	根小组件在嵌套绑定操作中绑定至的标记的完整路径。对等于非嵌套绑定中的 <code>::Path</code> 。
<code>::TopName</code>	根小组件在嵌套绑定操作中绑定至的标记的名称。对等于非嵌套绑定中的 <code>::Name</code> 。

请注意，这些特殊名称中的每一个都会求值为等于所需名称而不是真实标记本身的字符串常量。它们通常用于为用户提供与小组件或其根小组件绑定至的文件夹相关的信息。

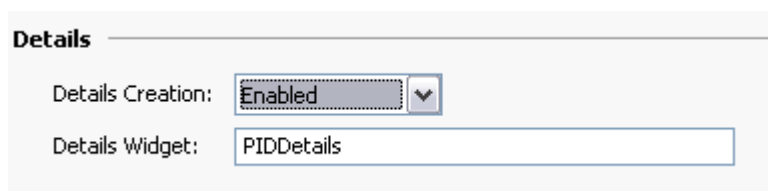
## 详情小组件

假设您创建了一个 `PID` 小组件，但是希望在用户按下该小组件内的按钮时显示更为详细的状态信息。最简单的方法就是创建一个更为复杂（也可能更大）的小组件，然后将其绑定至同一个循环。您会将此小组件放入另一个页面，然后从原始概览小组件中选择该页面，或许会使用数据项来指示小组件应使用哪个页面。

但是，详情小组件创建会自动执行所有这些步骤！

## 启用详情创建

此功能通过小组件的数据定义里的 *详情创建* 属性进行控制：



**Details**

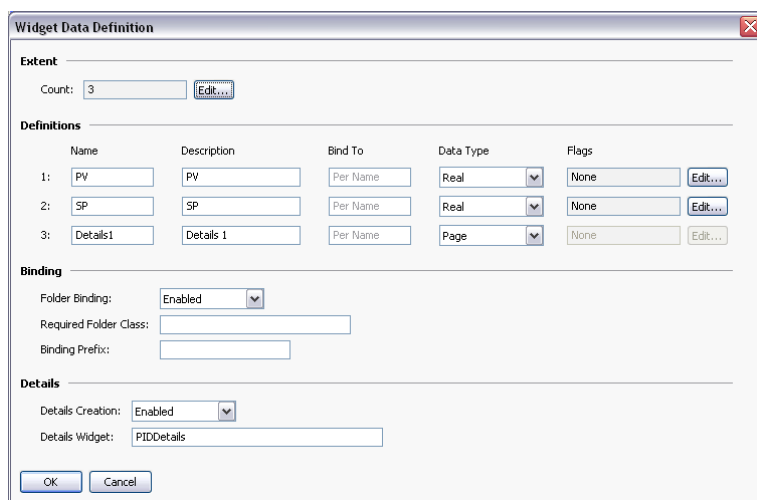
Details Creation:  ▼

Details Widget:

*详情创建* 属性用于提供一个或多个希望放置在其自身页面的详情小组件的逗号分隔的列表。每个小组件均通过分配用于保存的文件名来进行指定。上面的示例中，我们从 Crimson 的小组件目录中的一个文件中提取了一个叫做 PIDDetails.wid 的详情小组件。

## 定义数据项

我们还必须提供概览小组件中的数据项，这样我们就可以访问为详情小组件创建的页面的名称。这些属性必须命名为 Details1、Details2，依此类推，且详情小组件列表中的每个元素均拥有一个数据项。每个数据项的数据类型均应为页面数据类型。下面的示例中，我们创建了一个单一的这类属性，来持有单一详情页面的页面名称：



**Widget Data Definition**

Extent

Count:

**Definitions**

	Name	Description	Bind To	Data Type	Flags
1:	<input type="text" value="PV"/>	<input type="text" value="PV"/>	<input type="text" value="Per Name"/>	<input type="text" value="Real"/> ▼	<input type="text" value="None"/> <input type="button" value="Edit..."/>
2:	<input type="text" value="SP"/>	<input type="text" value="SP"/>	<input type="text" value="Per Name"/>	<input type="text" value="Real"/> ▼	<input type="text" value="None"/> <input type="button" value="Edit..."/>
3:	<input type="text" value="Details1"/>	<input type="text" value="Details 1"/>	<input type="text" value="Per Name"/>	<input type="text" value="Page"/> ▼	<input type="text" value="None"/> <input type="button" value="Edit..."/>

**Binding**

Folder Binding:  ▼

Required Folder Class:

Binding Prefix:

**Details**

Details Creation:  ▼

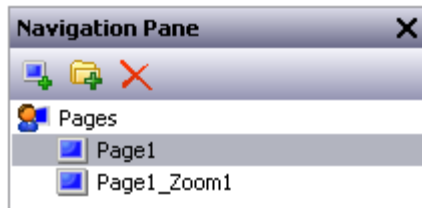
Details Widget:

## 绑定的结果

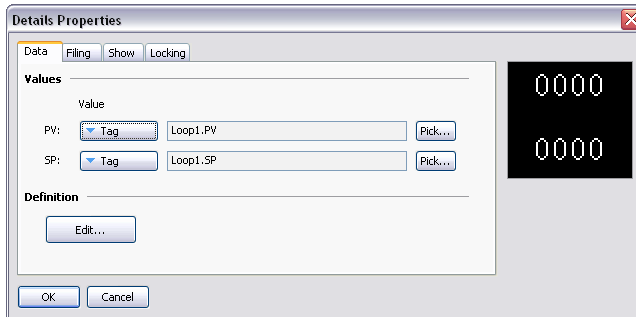
我们将概览小组件绑定至 PID 循环时，会创建一个新页面来持有详情小组件。新页面的名称根据包含概览小组件的页面的名称而定，但是会添加一个“Zoom”后缀和一个使名称唯一的数字。



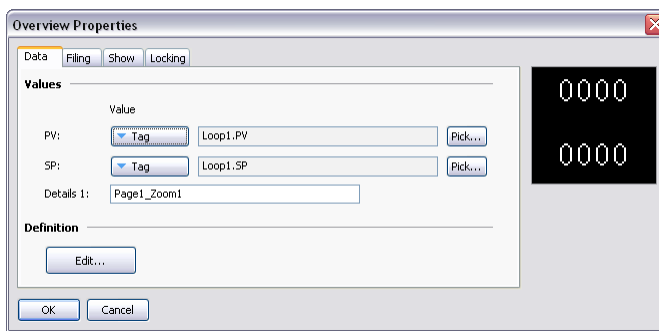
此页面会被放置在导航列表内的当前页面下面：



此页面内创建的详情会被绑定至循环：



并且概览小组件的属性会被修改如下：



可以轻松地在概览小组件里定义一个按钮，然后使此按钮调用一个 ShowPopup(Details1) 操作，从而显示关联的详情小组件。详情小组件自身可通过调用 HidePopup() 来关闭弹出。

## 多个详情页面

如果创建了多个详情页面，那么您会想起，概览小组件里叫做 Details1、Details2 并以此类推的数据项将持有这些页面的名称。这些数据线还可在详情小组件上被定义，并将被设为创建的页面的名称。如果您希望允许第一个详情页面导航至第二个页面并以此类推，从而将这些页面链接在一起，那么这就很有用。详情小组件还可以定义一个叫做 DetailsP 的特殊数据项，此数据项将被设为等于持有概览小组件的页面。这可用于返回概览——提供了多个详情页面时，这无法通过简单的 GotoPrevious() 来实现。



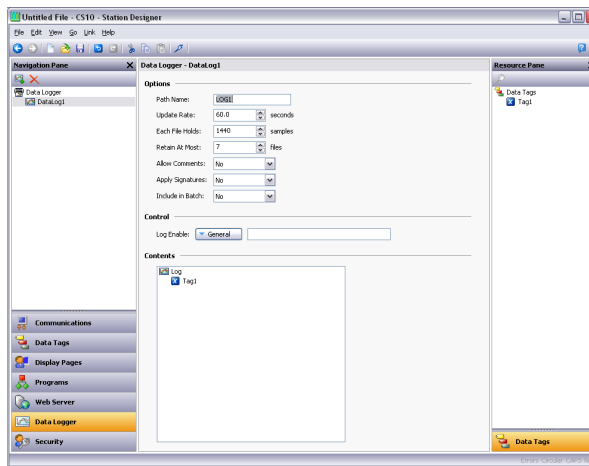
# 使用数据记录器

既然您已经配置了应用程序的核心，那么您可能会决定使用 **Crimson** 的数据记录器来将某些标记值记录进入 **CompactFlash** 卡。通过这种方式记录的数据会被存储在行业标准的逗号分隔的变量文件 (CSV) 中，并可使用多种方法轻松导入 **Excel** 之类的应用程序。要配置数据日志记录，请选择导航窗格里的数据记录器类别。

## 创建数据日志

可按常规方式在导航列表里创建数据日志。

每个日志均拥有下列属性：



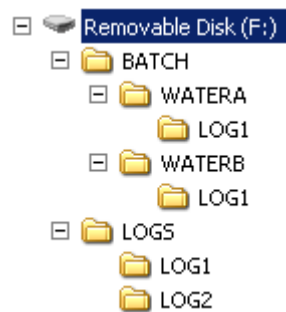
- **路径名称** 属性允许修改用于保存日志的目录。默认设置下，日志被保存在名称与日志名称相同的目录里。此属性允许按照与 8.3 命名约定不兼容的方式对日志重命名，同时仍使用有效的日志记录目录。
- **更新频率** 属性用于指示 **Crimson** 提取要记录的数据项的样本的频率。虽然可输入小数位，但是采样仅精确到 200 毫秒。最快采样频率为一秒，但是请注意，使用如此高的值将会生成极为大量的数据。会按照同一频率对日志里的全部标记进行采样。
- **每个文件持有** 属性用于指示每个日志文件将包含多少个样本。记录这么多样本之后，会使用不同名称来创建新的日志文件。通常而言，此值的设置标准为使每个日志文件包含合理量的数据。例如，上面所示的日志被配置为每天使用一个新日志文件。
- **保留最多** 属性用于指示最旧日志文件删除之前应在 **CompactFlash** 卡里保留多少个日志文件。此属性的设置标准为使使用记录信息的任何内容能够在信息被删除之前从 **Crimson** 设备提取数据。上面显示的日志被配置为保留一星期量的数据。
- **允许注释** 属性用于启用或禁用通过 `LogComment()` 函数向数据日志添加注释。请参阅参考手册，详细了解如何使用此函数。

- *包含在批次内* 属性用于在批次日志记录系统内包含或排除此日志。请参阅下文，详细了解如何进行批次日志记录。
- *日志启用* 属性用于允许或禁止日志记录。如果输入的表达式为真，则会启用日志记录。如果表达式为假，则会禁用日志记录。如果未输入表达式，则会默认启用日志记录。
- *内容* 属性用于指示应将哪些标记包含在数据日志之内。标记可从资源窗格拖入列表，并使用标准拖放技巧在列表内上下移动。

## 批次日志记录

第一次访问数据记录器时，您会注意到一个启用或禁用批次日志记录的全局设置。对于正常数据日志记录操作而言，数据记录器会将日志文件保存在为每个日志指定的文件夹名称之下。另一方面，批次日志记录也会将这样配置的全部日志保存在依照当前生产批次命名的目录里。这允许将与特定批次相关的全部日志作为一组来进行访问和操纵。

为了说明这一点，请看下面的目录结构：



此示例取自一个启用了批次日志记录并配置了两个数据日志的目标设备。第一个数据日志被设置为包含在批次之内，而第二个则没有。请注意，日志文件被默认存储在名称为 LOGS/LOG1 和 LOGS/LOG2 的目录内，但是还请注意，第一个日志还被放在了 BATCH 目录的子目录之下。每个子目录都包含了在该批次开始时间和结束时间之间采样的数据。

## 控制批次

批次日志记录通过一系列函数来进行控制。`NewBatch(name)` 将创建一个名称为 *name* 的文件夹，结束当前批次并开始一个新批次。此命令之后记录的文件将保存在新文件夹里。`EndBatch()` 函数将停止当前批次，而 `GetBatch()` 则将返回当前活跃批次的名称。更多信息请参阅参考手册。

## 日志文件存储

如上所述，数据日志将它们的数据存储在目标设备的 **CompactFlash** 卡上的一系列文件里。文件被放在日志的属性里指定的子目录里，而此目录则会存储在叫做 **LOGS** 的根目录项内。

日志文件按照日志计划开始的时间和日期来进行命名。如果文件包含的信息大于一个小时的量，那么文件就会被命名为 `YYMMDDhh.CSV`，其中 `YY` 表示文件的年份，`MM` 表示月份，`DD` 表示日期，`hh` 表示小时。如果文件包含的信息不足一个小时的量，那么文件则会被命名为 `MMDDhhmm.CSV`，前面六个字符的含义与上文描述的相同，结尾的 `mm` 则表示日志开始的分钟。这些规则确保了每个日志文件的名称都依赖于创建时间而独一无二。

文件的长度依赖于 *更新频率* 和 *每个文件持有* 属性。例如，将更新频率设为 5 秒，将样本数目设为 360，那么每个文件就会持有  $(5 \times 360) / 60 = 30$  分钟的量的数据，因此会使用 `MMDDhhmm.CSV` 文件名格式。每隔 30 分钟就会在整点或半点创建一个新文件。

## 日志记录流程

Crimson 的数据记录器使用两个单独的流程来进行操作。第一个流程会按照每个日志的属性指定的频率对每个数据点进行采样，然后将数据放入目标设备的 RAM 里的缓冲区。第二个流程则会每隔两分钟执行一次，将数据从内存写入 CompactFlash 卡。

此结构拥有若干优势：

- 可以确保向 CompactFlash 卡进行的写入以两分钟为界，亦即精确在小时过后的 2 分钟、4 分钟、6 分钟等。这意味着，如果目标设备支持热插拔 CF 卡，那么就可以等待下一个写入操作开始，而且，CompactFlash 卡的活动 LED 停止闪烁之后，下次写入之前您就拥有至少两分钟来移除卡，而无需担心数据损坏。只要在四分钟之内插入新卡，数据就不会丢失。
- 无需为每个样本连续更新 CompactFlash 卡的文件系统数据结构，所以对卡进行的写入操作可以获取更高性能。对于配置为使用较高频率进行采样的日志而言，常规 CompactFlash 卡并不允许不经缓冲流程便可靠地写入数据。

请注意，因为最多有两分钟不会向 CompactFlash 卡写入数据，所以如果终端断电，则可能会丢失最多此量的日志数据。此外，如果写入过程中目标设备断电，则 CompactFlash 卡可能会损坏。为了确保不会发生永久损坏，Crimson 使用了一种日记系统，此系统会将写入缓存至终端内的附加稳定内存。如果设备侦测到断电时中止了写入，那么重新通电时写入就会被重复，从而修改任何数据损坏并修复 CompactFlash 卡。

如果想从正在进行数据日志记录的面板上移除 CompactFlash 卡，那么必须遵循上文描述与活动 LED 相关的步骤，只有在活动停止之后才可以断电。如果不确定终端是否正确断电，请重新通电，完成一个 CompactFlash 卡写入序列，再按照正确步骤断电。然后就可以安全地移除卡。

因为移除 CompactFlash 卡的步骤较为复杂，所以 Crimson 提供了一系列访问日志文件的其它机制，从而无需进行移除。下面描述了这些方法。

## 访问日志文件

有五种方法可以访问日志文件：

- 可以按照本手册开始时描述的方法将 CompactFlash 卡装载为 PC 的驱动器。这允许通过 Windows 资源管理器复制日志。在 CompactFlash 卡首次装载时 Windows 可对卡设置的负荷量方面，此方法有一些不足之处。
- 可以使用下一章描述的 Web 服务器。启用 Web 服务器之后，使用 Web 浏览器（如 Microsoft Internet Explorer），或使用通过 Crimson 提供的 WebSync 实用工具实现的自动流程，即可通过 TCP/IP 连接对日志文件进行访问。
- 可以使用 FTP 服务器允许远程客户端连接至 Crimson 设备后下载日志。详情请参阅使用服务章节。

- 可以使用同步管理器定期将文件推送至 FTP 服务器。详情请参阅使用服务章节。
- 可以配置通信类别里的内存条选项，从而将日志文件自动复制到 USB 内存设备。详情请参阅本手册的使用通信章节。





# 使用 WEB 服务器

通过使用调制解调器或目标设备的以太网端口，Crimson 的 Web 服务器可用于通过 TCP/IP 连接公开各种数据。这允许对诊断信息或数据记录器记录的值进行远程访问。通过在导航窗格内选择 Web 服务器类别，可以配置 Web 服务器。

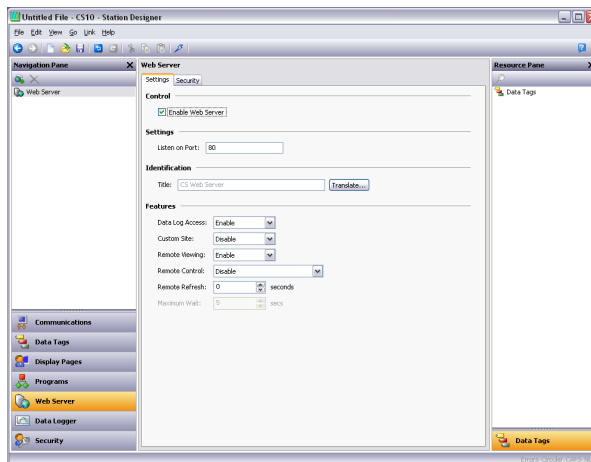
## 重要说明

虽然 Crimson 提供了一系列保护机制来限制对 Web 服务器的访问，但设计系统时仍应运用优秀的工程实践。也就是说，您应避免通过 Web 服务器进行任何与安全相关的操作。最好使用外部防火墙，防止在 Crimson 自身的安全防护受到破坏时出现未经授权的访问。根本而言，安全还是您自己的责任，而且 Red Lion Controls 也不建议您完全依赖于 Crimson 自身的安全措施。

## WEB 服务器属性

Web 服务器的属性可以从导航列表的根条目内访问。

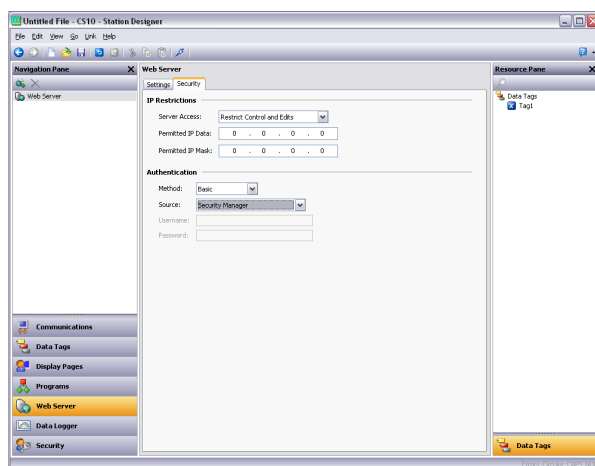
### 设置属性



- **启用 Web 服务器** 属性用于启用或禁用 Web 服务器。如果启用了服务器，面板会等待传入请求，然后按需满足请求。如果禁用了服务器，则该端口的连接将被拒绝。请记住，为使服务器正常运作，必须使用通讯类别配置一个 TCP/IP 连接。
- **侦听端口** 属性指示了 Web 服务器将侦听的 TCP 端口号。端口 80 是 HTTP 协议使用的标准端口，可能最适用于您的应用程序。
- **标题** 属性用于提供显示在 Web 服务器菜单内的标题。该名称可用于区分同一网络上的若干终端，因此可以确保访问正确的终端。
- **数据记录访问** 属性用于启用或禁用对数据记录器创建的文件进行的 Web 访问。如果远程客户端程序要使用 Web 服务器自动同步数据记录，则必须启用该功能。

- **远程查看** 属性用于启用或禁用一项功能，通过该功能，可以使用 Web 浏览器查看目标设备显示内的当前内容。对操作员可能遇到的与操作员面板或其控制的机器有关的问题进行远程诊断时，该功能十分有用。
- **远程控制** 属性用于启用或禁用一项选项，通过该选项，远程查看功能得以延伸，从而允许使用 Web 浏览器模拟按下按键或显示基元，进而允许对面板或其控制的机器进行远程控制。虽然该功能极为有用，但是必须使用各种安全参数来避免对机器进行越权篡改。如果可以通过 Internet 访问面板，则强烈推荐使用外部防火墙。
- **自定义站点** 属性用于启用或禁用一项功能，通过该功能，存储在 CompactFlash 卡 \WEB 目录内的文件通过 Web 服务器进行公开。下文详细描述了该功能。
- **远程刷新** 属性表示了连接至 Web 服务器的 Web 浏览器刷新远程查看页面的频率。值设为 0 时，将尽可能快地进行刷新。较高的值会减少带宽使用，可能较为适用于调制解调器连接。

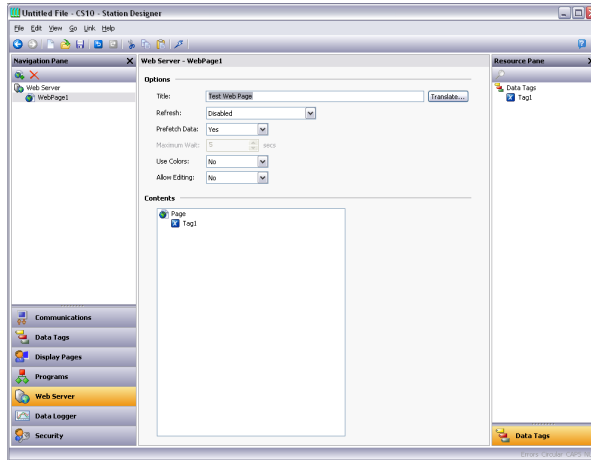
## 安全属性



- **IP 限制** 组用于限制 Web 服务器访问其 IP 地址与指示的掩码和数据匹配的主机。可以限制所有访问，也可以使用筛选来限制远程控制或数据编辑功能。
- **验证** 组定义了是否要求连接至 Web 服务器的用户提供用户名和密码信息，还定义了如何验证信息。**方法** 定义了所使用的算法，其中摘要为推荐选择。**源** 属性用于指示是否直接在 Web 服务器属性内输入所需的用户名和密码，或者是否在 Crimson 的安全系统内创建用户并授予其 Web 服务器访问权。

## 添加网页

除上述功能之外，Web 服务器还支持显示通用网页，各个网页均含有一组预定义的标记值。这些页面在导航窗格内按照常规方式创建。每个网页均有下列属性：



- **标题** 属性用于标识通过 Web 浏览器展示给用户的菜单中的网页。虽然标题是可译的，但当前版本的 Crimson 仅使用美国版本的文本。
- **刷新** 属性用于标示是否应指示 Web 浏览器自动刷新页面内容。支持 1 到 8 秒之间的更新速率。请注意，Web 浏览器的闪烁量会根据所用的精确包和机器性能而有所变化。更新可能不会为完全无闪烁。
- **使用颜色** 属性用于指示呈现页面时是否应使用标记着色定义的颜色。如果启用了该属性，则 Web 浏览器内显示的颜色会根据标记状态而改变。详情请参阅使用数据标记章节。
- **允许编辑** 属性用于启用通过此页面对数据标记进行编辑。如果启用了该属性，则每一个数据值均会显示编辑按钮，允许用户更改值。如果为标记定义了安全设置，则登入 Web 服务器的用户必须拥有足够权限方可修改标记。使用该功能时，建议同时使用验证。
- **内容** 属性用于指示应将哪些标记包含在页面内。可以从资源窗格内将标记拖入列表，并使用标准拖放技巧在列表内上下移动标记。

## 使用自定义网站

虽然通过标准网页可以迅速简便地访问终端内的数据，但是您可能会发现，由于无法编辑数据的精确格式，从某种程度而言，您的艺术能力无法发挥得淋漓尽致。因此，您可以通过 **Crimson** 的自定义站点功能使用您最喜爱的第三方 **HTML** 编辑器创建完全自定义的网站，还可通过在设备的 **CompactFlash** 卡内插入某些特殊序列和存储结果文件来使用目标设备的 **Web** 服务器发布该站点。

### 创建站点

网站可以使用您的浏览器所支持的任何 **HTML** 功能，但是不能使用 **ASP**、**CGI** 或其它服务器端工具。**HTML** 文件和关联图形使用的文件名必须符合 8.3 命名约定。这就意味着，文件扩展名应该是 **HTM** 而不是 **HTML**，应该是 **JPG** 而不是 **JPEG**。这还意味着文件名主体不得超过 8 个字符，而且不能依赖于大小写字母之间的区别来区分不同页面。只要确保目录符合 8.3 命名约定且不依赖于大小写来进行区分，即可使用任何目录结构。

### 嵌入数据

要在网页内嵌入标记数据，请插入序列 `[[N]]`，并将 **N** 替换为标记相应的索引编号。在数据标记类别内选定了一个标记之后，该索引编号显示在状态栏之内，并与标记的创建顺序大致对应。包含该序列的网页受到服务时，该序列将由按照标记属性来编排格式的标记当前值进行替换。

### 部署站点

要部署自定义网站，请将其复制到将安装至目标设备上的 **CompactFlash** 卡的 `\WEB` 目录内。要复制文件，请将卡装载为本手册先前章节所描述的驱动器，或者使用连至 **PC** 的适当写卡器。在 **Web** 服务器的属性内启用自定义站点之后，站点将出现在主 **Web** 菜单之内。选定站点之后，将显示 `\WEB` 目录内一个叫做 `DEFAULT.HTM` 的文件。在此之后，导航将根据站点内的链接进行。

# 使用安全系统

Crimson 拥有强大的安全功能，这些功能允许定义哪些操作员可以访问哪些显示页面，并限制那些操作员只能更改特定数据。软件还拥有安全日志记录功能，此功能可用于记录对数据值进行的更改，指示更改何时发生和由谁进行。

## 安全基础

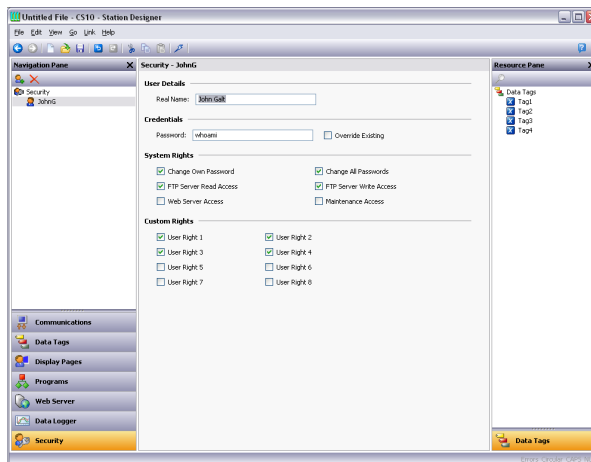
下面的章节详述了安全系统使用的一些基础概念。

### 基于对象的安全

Crimson 的安全系统以对象为基础。这意味着安全特征被应用至显示页面或标记，而不是访问页面或更改标记的用户界面元素。另一种基于主体的方法则通常意味着，将安全设置应用至每个可能更改受限数据的用户界面元素时，必须极为小心。Crimson 的方法避免了这种重复，确保了一旦决定保护一个标记，那么它就会在数据库里始终受保护。

### 指定用户

Crimson 支持创建任何数目的用户，每个用户均将拥有一个用户名、一个真实姓名和一个密码。用户名是一个不区分大小写、不能有空格的字符串，用于登录时确定用户身份，而真实姓名则通常是一个较长的字符串，用于在日志文件中记录进行更改的用户的人工可读的身份。请注意，只要适用于您的应用程序，就可随意使用这些字段。例如，您可以创建代表一组用户或职务的用户，如操作员、监督员和经理。您还可决定使用真实姓名来保留时钟号码之类的项，从而将用户身份绑定至您的 MRP 系统。



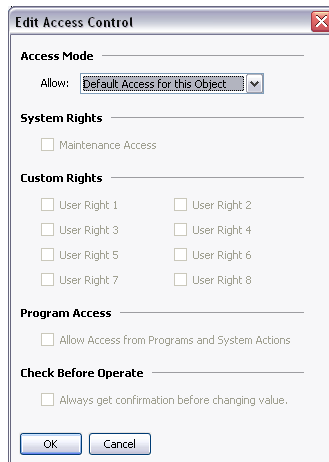
## 用户权限

会授予每个用户零个或更多权限。无权限的用户仅能访问那些不需要记录用户身份的对象，而拥有更多权限的用户则可访问需要那些权限才能访问的对象。权限分为系统权限和用户权限，前者控制 **Crimson** 软件内各个功能的访问权限，后者则供一般使用。例如，用户权限 1 可用于在数据库内控制对生产目标的访问。仅会为希望能够更改这些事项的用户授予此权限。

## 访问控制

需受安全约束的对象拥有关联的 *访问权限* 控制属性。

编辑此属性会显示下列窗口：

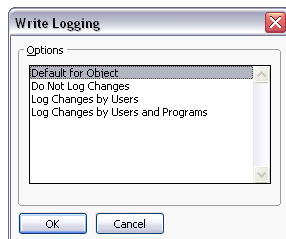


这些设置允许指定项是否可被任何人访问，或可被身份已知的操作员访问，或可被拥有特定用户权限的用户访问。还可指定标记是否可被用户操作之外的其它事项调用的程序更改。此功能可确保敏感数据不会发生后台更改，即使编程错误试图进行此类更改。

## 写入日志记录

标记还拥有 *写入日志记录* 属性。

编辑此属性会显示下列窗口：



所选内容指示了是否应记录由用户或程序对标记进行的更改。此功能允许创建记录对系统进行的更改的审计线索，从而简化错误查找流程，并提供流程配置的质量控制信息。请注意，记录由程序进行的更改时，必须十分小心，因为某些数据库可能会在某些情况下记录量不能管理的数据。

## 默认访问权限

为了加速配置流程，Crimson 还可以指定默认访问权限，为映射的标记、内部标记和显示页面写入日志记录参数。对于必须记录对外部数据进行的更改而无需创建审计线索即可操纵 Crimson 的内部数据的系统而言，映射的标记和未映射的标记之间的区别十分重要。

## 按需登录

Crimson 的安全系统支持常规登录和按需登录。使用按钮之类的用户界面元素来激活登录用户操作或调用 `UserLogOn()` 函数时，就会发生常规登录。如果用户尝试进行没有足够访问权限的操作，或失败的登录尝试没有在同一操作内发生，则会发生按需登录。例如，用户可能会按下运行一个程序从而重设若干值的按钮，只要程序试图更改需要安全访问权限的值，系统就会提示输入登录凭据。这种方法减少了操作员交互量，使系统反应更为灵敏。

## 维护访问权限

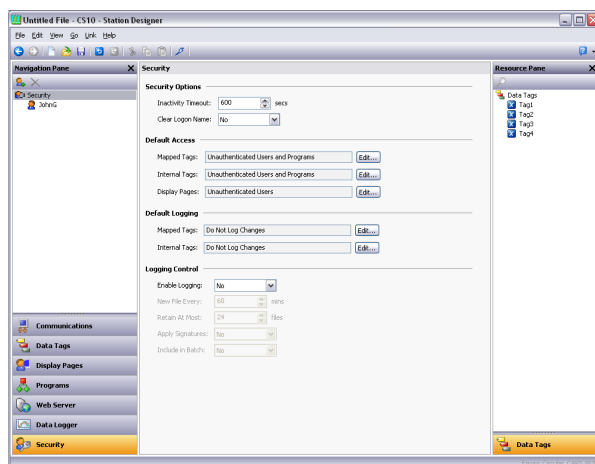
系统还提供了一种叫做维护模式的功能，允许在进行系统调试时重写用户不活动超时。如果一个显示页面使用维护访问权限标注为可访问，并且当前用户因为该权限而获取了该页面的访问权限，那么就会激活此模式。使用此模式，调试系统时就无需重复登录。

## 操作前检查

操作前检查功能允许强制用户确认对特定敏感数据项做出的每个更改。选择数据标记的安全描述符上的适当设置，即可启用此功能。对启用了此功能的标记进行更改时，会出现一个弹出窗口，显示新值和旧值，并要求在允许更改之前进行确认。无论用户当前是否登录，此功能均会起作用，且独立于进行更改所需的任何用户权限之外，还独立于创建用户界面时定义的保护操作之外。

## 安全设置

可通过安全类别里的根项来访问安全系统设置：



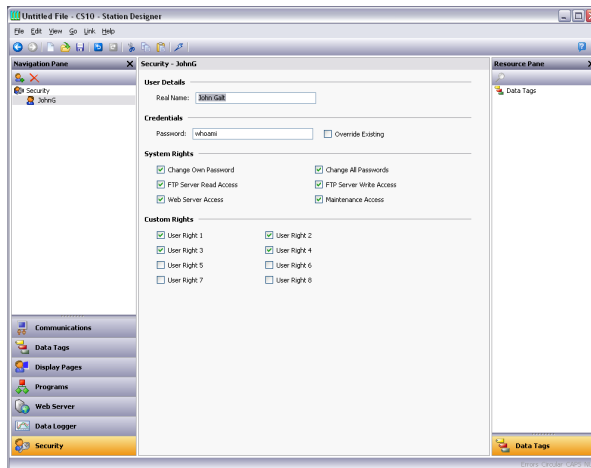
可用属性如下：

- **不活动超时** 属性用于指示自动注销当前用户之前必须经过多少没有用户输入的时间。为此属性设置过高的值会使系统不够安全，而过低的值则会使操作员难以操作系统。
- **清除登录名称** 属性用于指示要求操作员登录之前是否应清除用户名。如果禁用了此设置，就会显示先前的用户名，只需重新输入密码即可。启用此功能会使系统更为安全，而且在某些行业内可能必须启用此功能。
- **默认访问权限** 属性用于指示没有为该项定义特定访问权限时要提供给各种对象的访问权限。各种设置已经在上文的访问控制部分进行了说明。
- **默认日志记录** 属性用于指示没有为标记定义日志记录标准时是否应记录对映射的和未映射的标记做出的更改。默认情况下，不能记录编程访问，因为进行这种日志记录时必须小心谨慎，从而避免过量日志记录活动。
- **日志记录控制** 属性定义了是否以及如何创建安全日志。请参阅使用数据记录器章节，详细了解如何写入数据和命名文件。



## 创建用户

可通过常规方法在导航列表内创建并操纵用户：



每个用户均拥有下列属性：

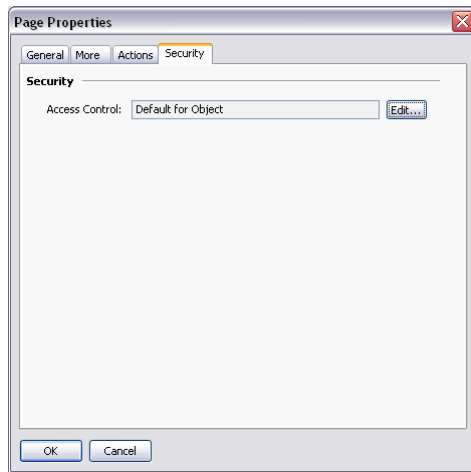
- **真实姓名** 属性用于在安全日志中记录用户的身份，还会显示在用于在运行时更改密码的安全管理器基元内。如果要求了最高安全，那么应使用不易于从真实姓名猜测出的用户名。
- **密码** 属性为此用户指定了初始密码。密码区分大小写，包含了字母数字字符。请注意，如果勾选了 **重写现有** 勾选框，那么下载此数据库时，会重写从目标设备对此密码进行的任何更改。
- **系统权限** 属性用于授予用户执行某些系统操作的权限。与密码更改相关的属性不言自明，而维护模式的使用则已在上文说明。
- **自定义权限** 属性用于授予用户某些权限，这些权限随后可在数据库内用于允许对标记分组或显示页面分组进行访问。这些权限的具体用途由系统设计者指定。

## 指定标记安全

每个标记都有一个叫做安全的选项卡，此选项卡定义了标记的访问控制和写入日志记录设置。如果未定义特定设置，则系统将根据标记是否映射至外部数据来使用适当的默认设置。

## 指定页面安全

显示页面的访问控制设置通过它们的属性对话框来定义：



同样，如果未定义设置，则会使用默认设置。

## 安全相关的函数

请参阅参考手册，详细了解函数 `UserLogOn()`、`UserLogOff()` 和 `TestAccess()`。从一个程序内对许多值进行更改时，最后一个函数非常有用，因为它允许在代码早期强制进行访问权限检查，从而避免做出会使后期操作由于用户权限不足而失败的更改。

# 使用服务

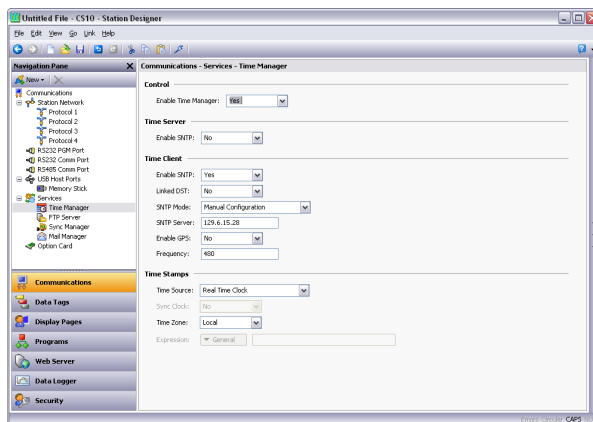
除了此文件前文描述的核心功能外，您还可在通信类别中配置多种服务。这些服务显示在导航窗格内的服务图标之下，每种服务均在下文进行了说明。

## 使用时间管理

Crimson 拥有一些功能，可使用它们将目标设备的时间和日期设置与多种源进行同步。时间管理器还可以保存与设备的时区和当前是否启用了夏令时相关的信息。事实上，拥有准确的时区信息对于正确进行同步而言至关重要，因为各种同步方法均只能通用协调时间（UTC，也叫做格林威治标准时间）。Crimson 可用作客户端从其它基于 Crimson 的设备请求时间，也可用作服务器这些设备提供时间。请注意，服务器实现目前尚不支持第三方客户端。

## 配置服务

时间管理器通过导航窗格里关联的图标进行配置：



*启用时间管理器* 属性用于控制对其它功能的访问。如果未勾选此属性，则 Crimson 仅将按照本地时区运行，而且无法获取时区和其它时间管理信息。

## 时间服务器

适当配置时间服务器部分的启用 SNTP 属性会指示 Crimson 用作 SNTP 服务器。这会允许其它 Crimson 设备将自己的时钟与此单元的时钟进行同步。请注意，Crimson 的 SNTP 实现并不完全与 RFC 相容，也不能用作第三方客户端的同步源。

## 时间客户端

在时间客户端部分的启用 SNTP 属性内选择是指示 Crimson 运行其 SNTP 客户端。然后，Crimson 会尝试将其时钟与另一个基于 Crimson 的设备的时钟进行同步，或与另一个接入网络的 SNTP 时间源（如运行 Windows XP 的电脑）进行同步。时间客户端拥有下列附加属性：

- *链接 DST* 属性用于指示 SNTP 客户端尝试从 SNTP 服务器读取当前的夏令时设置。因为此功能不是 SNTP 协议的标准构成部分，所以，只有在指定

另一个 **Crimson** 设备作为服务器时，它才有用。此功能非常有用，因为它允许通过工厂网络中的一个单一设备来调整夏令时，其它设施然后将遵循中央设置。

- *SNTP 模式* 和 *SNTP 服务器* 属性用于配置简单网络时间服务服务器的 IP 地址。如果选择了 *通过 DHCP 配置*，则为了使用 DHCP，必须至少配置一个以太网端口，并且服务器必须配置为通过选项 42 指定一个服务器。
- *启用 GPS* 属性用于指示时间客户端使用通过 NMEA-0183 连接的 GPS 单元作为替代方法来获取当前时间。可使用适当驱动程序将单元连接到任何串行端口。
- *频率* 属性指定了尝试 **Crimson** 使用以上启用的方法尝试同步其时间的频率。**Crimson** 将始终在通电后二十秒内尝试同步，然后按照此属性的规定进行同步。如果给定同步尝试失败，则单元将每隔 30 秒重试一次，直至成功找到适当的时间源。

## 时间戳

**Crimson** 可在目标设备的 CompactFlash 卡上记录各种日志文件，每个日志项目都有一个时间戳。默认情况下，时间戳来自本地实时时钟，并使用本地时区。可通过下列属性改变行为：

- *时间源* 属性用于指示应该从哪里获取时间戳。默认设置会从单元自己的实时时钟获取时间，而替代设置则允许使用一个表达式来定义当前时间。此表达式通常是对连接设备中的一个数据项的引用，允许将该设备的时钟用于数据日志记录。表达式必须在 *表达式* 属性中输入。
- *同步时钟* 属性用于指示是否应将本地实时时钟与上文指定的替代时间源进行同步。如果启用了此选项，则本地时钟将在启动时同步，并在之后定期同步；如果替换源由于通信问题而不可用，则本地始终将被用作时间戳源。
- *时区* 属性用于指示用于时间戳的时区。仅在本地实时时钟被配置为时间戳源时，此属性方可适用。选择本地将使用本地时区，选择 UTC 则将使用通用协调时间。第二种设置可以生成能够轻易在时区之间移植的日志文件，并且日志文件不会在切入和切出夏令时时出现不连贯。

## 选择 SNTP 服务器

配置 SNTP 客户端时，有若干选项可以选择服务器。

如果您的网络基础设施内有一个基于 Windows 或 Unix 的时间服务器，那么最终您应该与此源进行同步，从而确保企业范围内的正确同步。但是，如果在同一网络上有多台 Crimson 设备，那么您会发现，最好的方法就是将其中一台指定为主设备来设置夏令时，然后将该设备单独同步至企业时间源。然后您可以配置其它设备与主设备进行同步，并启用链接 DST 功能将夏令时设置在整个工厂。

如果没有可用的企业时间源，则可指定一个单一的 Crimson 设备，将其作为操作员进行时间设置的点，然后将其它设备与该源进行同步。此外，如果您的装置通过以太网或调制解调器连接提供了对 Internet 的 TCP/IP 访问权限，那么您可以将 SNTP 客户端配置为与公共时间服务器进行同步。例如，可以使用 192.6.15.28，这是由 NIST 提供的公共时间服务器的当前 IP 地址。

下列网站里显示了其它一些服务器：

<http://support.microsoft.com/kb/262680>

请注意，因为 Crimson 使用 IP 地址而不是主机名来引用 SNTP 服务器，所以如果一个服务器被重定位到新网络地址，则 Crimson 将失去与该服务的连接。虽然此类重定位非常少见，但是它们不受您和 Red Lion 的控制。因此，我们认为使用通过 DNS 访问其自身来源的企业时间源更为可取！

## 时区配置

如上所述，如果 Crimson 设备要使用高级时间管理，则其必须知晓当前时区。此信息可以通过两种方式提供。最简单的方法就是使用 Crimson 配置软件的链接菜单内的发送时间命令。除了设置时钟之外，此命令还会发送 PC 的当前时区和夏令时的状态。Crimson 将在稳定内存中存储这些数据，并从该点开始使用这些数据。很显然，在将时间和日期信息发送至单元之前，必须确保 PC 上拥有有效的时间和日期信息！

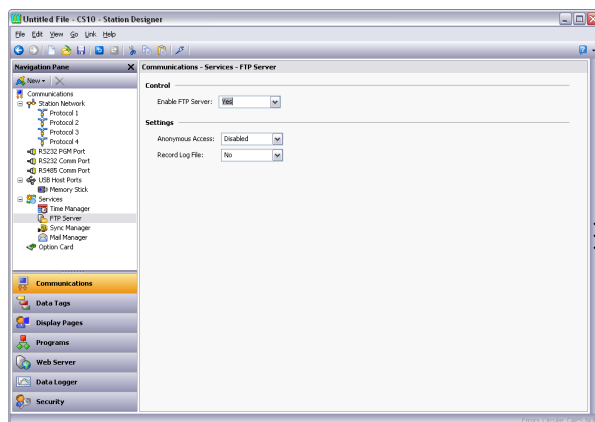
另一种方法则是使用系统变量 TimeZone 和 UseDST。第一个变量会持有当地时区和 UTC 相差的小时数，可能是正数，也可能是负数。例如，设置为 -5 就对应于美国的东部标准时间。第二个变量根据是否使用了夏令时而包含值 1 或值 0。通过用户界面编辑任意变量均会使单元的时钟按照新的设置进行更改。例如，启用夏令时会将时钟往前拨一个小时，而禁用夏令时则会将时钟往后拨一个小时。典型数据库仅需要公开 UseDST 供用户编辑，而且如果已使用了上文描述的链接 DST 功能，则甚至连这都不需要。

## 使用 FTP 服务器

Crimson 的 FTP 服务器提供了一种在 Crimson 设备和运行了 FTP 客户端应用程序的远程计算机之间交换文件的方法。Crimson 设备将起到服务器的作用，等待客户端应用程序连接并下载或上传文件。

### 配置服务

FTP 服务器通过导航窗格里关联的图标进行配置：



可配置下列属性：

- **匿名访问** 属性定义了要授予使用匿名 FTP 访问服务器的用户的权限（如有）。设置为已禁用会阻止匿名访问。设置为只读会允许用户从 CompactFlash 卡里下载文件，但会阻止上传。设置为读写会允许上传和下载。
- 启用 **记录日志文件**，会在 CompactFlash 卡的根目录里记录所有 FTP 交互活动的日志。调试 FTP 操作时，此文件可能非常有用，但是这有可能会略微降低性能。

### FTP 安全

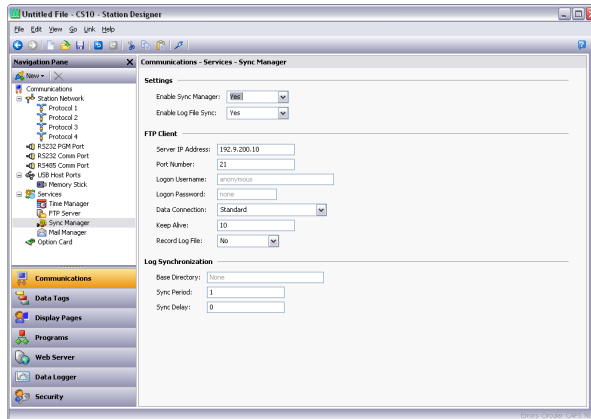
因为 FTP 服务器可以提供对 CompactFlash 卡的完全访问权限，所以强烈建议使用安全管理器来定义特定用户名和密码组合，从而授予那些用户适当的访问权限。总体而言，您应避免授予匿名访问权限，尤其应避免允许匿名写入。

## 使用文件同步

可以使用同步管理器在 **Crimson** 设备和 **FTP** 服务器之间交换文件。此功能可用于自动或按需将日志文件与服务器电脑进行同步，从而提供通过 **Web** 服务器访问日志文件的替代方法，并允许从多个通信站向中心点进行无人参与的文件传输。（请注意，虽然由于历史原因它被称为同步管理器，但是此服务实际上是基于一个常规目的 **FTP** 客户端的，无论是否正在进行文件同步，该客户端均还可用于进行其它 **FTP** 传输。）

## 配置服务

同步管理器通过导航窗格里关联的图标进行配置：



## FTP 客户端

下列属性与 **FTP** 客户端相关：

- **启用同步管理器** 属性用于启用 **FTP** 客户端。客户端可以在实际并未启用同步的情况下启用，这就允许将其用于通过 `FtpFilePut()` 和 `FtpFileGet()` 函数进行手动文件传输。
- **启用日志文件同步** 属性用于启用实际同步。其它与此功能相关的设置详见下一章节。
- **服务器 IP 地址** 属性用于指示服务器的 **IP** 地址。
- **端口号** 属性用于指示 **FTP** 客户端服务将试图连接的 **TCP** 端口。默认值适用于大多数应用程序，因为大多数服务器都会侦听端口 **21**。
- **登陆用户名** 和 **登录密码** 是建立连接时提交给服务器的凭据。尽管根据服务器实现而定，但两者通常都区分大小写。进行匿名登录时，请将用户名留为默认值，或将密码留空，或按照所提供的服务器的要求输入您的电子邮件地址。
- **数据连接** 用于在标准模式和 **PASV** 模式之间进行选择。您可以启用 **PASV** 模式，从而使 **FTP** 客户端发起所有数据连接，而不是等待来自服务器的入站连接。在非 **FTP** 感知的防火墙后或通过某些类型的网络地址转换进行操作时，可能会需要此模式。通常而言，通过 **GPRS** 调制解调器连接进行操作时也可使用此模式。

- *保持活动状态* 时间是 FTP 连接应保持活动状态以防需要进一步传输的时间量。零值会在当前传输完成后立即关闭连接。传输多个文件时，非零值会使操作更为高效。
- *记录日志文件* 属性可用于在 CompactFlash 卡的根目录里记录所有 FTP 交互活动的日志。调试 FTP 操作时，此文件可能非常有用，但是这有可能会略微降低性能。

## 日志同步

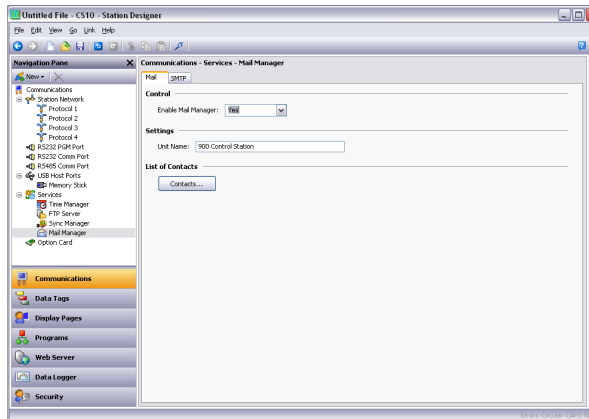
下列属性与日志文件同步尤其相关：

- *基目录* 属性定义了服务器上放置日志文件的目录。此目录与 FTP 服务器的文件夹空间有关，与服务器自身的文件系统的基础目录结构无关。通常而言，您应为每个同步至给定服务器的 Crimson 设备指定一个不同的基目录。
- *同步周期* 属性指定了 FTP 客户端连接至服务器并传输其文件的频率。周期以小时为单位，始终从凌晨十二点开始，这样一来，选择值三就会在凌晨十二点、3:00 am、6:00 am 等时间进行传输。
- *同步延迟* 属性定义了文件传输发生的标准时间之后以分钟为单位的延迟。此属性可用于允许多个终端与一台服务器对话，并确保不会因所有文件传输同时发生而导致超出目标设备荷载。



## 使用电子邮件

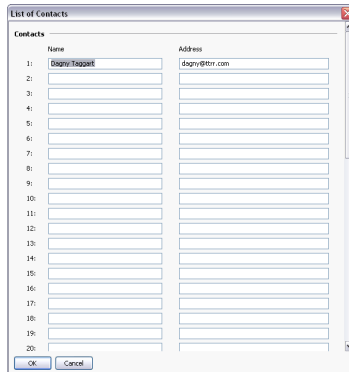
Crimson 可配置为在出现警报条件或需要通知系统内的其它事件时发送电子邮件消息。邮件传输和电子邮件地址通讯簿通过邮件管理器进行配置：



邮件选项卡上的属性用于启用或禁用邮件管理器，并用于为运行 Crimson 的设备提供名称。此名称将在电子邮件消息中用于识别消息的发起方。通常而言，应用程序将使用设备所连接的机器的名称，或使用其所监测的站点的名称。

### 添加联系人

联系人按钮可用于访问 Crimson 的通讯簿：

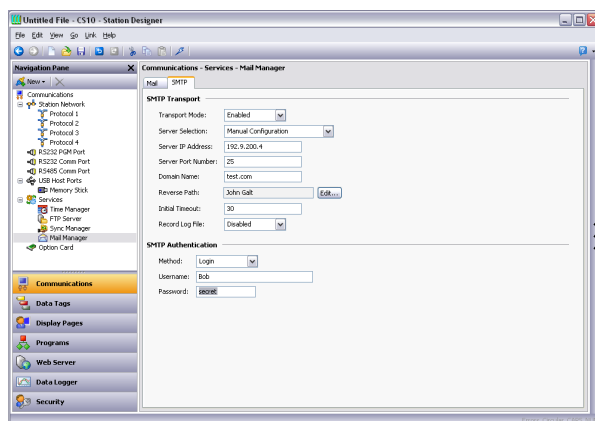


每个条目均允许输入显示名称和地址。地址的格式应适合于所需传输。例如，SMTP 名称应该是常用的 name@domain 格式，而 SMS 名称则应按照国际电话号码的格式输入，但无需前导加号。多个电子邮件地址可以使用分号隔开输入，从而创建简单邮件列表。

### 配置 SMTP

SMTP 选项卡用于配置简单邮件传输协议。这是用于在 Internet 或其它 TCP/IP 网络上发送电子邮件的标准协议。SMTP 地址需遵照常见的 name@domain 标准。

SMTP 传输的配置选项如下所示：



## SMTP 传输

- **传输模式** 属性用于启用或禁用传输。请注意，启用 SMTP 传输之前，必须通过邮件选项卡启用邮件管理器。还请注意，为了使邮件管理器能够传送消息，必须启用至少一个传输。
- **服务器选择** 属性定义了传输定位 SMTP 服务器的方式。如果使用了手动选择，则应使用 **服务器 IP 地址** 属性来手动指定服务器。如果选择了 **通过 DHCP 配置**，则必须将至少一个以太网端口配置为使用 DHCP，且服务器必须配置为通过选项 69 指定 SMTP 服务器。
- **服务器 IP 地址** 用于在启用了手动服务器选择时指定 SMTP 服务器。服务器必须配置为接受来自面板的邮件，并在应用程序要求时中继消息。
- **服务器端口号** 属性定义了用于 SMTP 会话的 TCP 端口号。默认值是 25。此值适用于大多数应用程序，只有在 SMTP 服务器被配置为使用其它端口时才需要调整。
- **域名** 属性指定了要在 HELO 命令或 EHLO 命令里传递到 SMTP 服务器的域名。绝大多数 SMTP 服务器都会忽略此字符串。如果您的 SMTP 服务器试图进行 DNS 检查从而确认客户端身份，您就可能需要在 DNS 配置里输入一些适当内容，但这种情况并不多见。

- **反转路径** 属性指定了用作目标设备发送的消息的发起方的电子邮件地址。属性包含了一个显示名称和一个电子邮件地址。由于 **Crimson** 不能接收消息，所以电子邮件地址经常会被设置为发送回复后返回“无法投递”消息的内容。
- **初始超时** 属性指定了邮件客户端等待 **SMTP** 服务器发送欢迎横幅的秒数。一些 **Microsoft** 服务器会试图与邮件客户端协商 **Microsoft** 特定的身份验证，因此会延迟显示横幅的时间点。在与这样的服务器进行操作时，您可能会希望将此时间量延长至 2 分钟或更多。
- **记录日志文件** 属性可用于在 **CompactFlash** 卡的根目录里记录所有 **SMTP** 交互活动的日志。调试 **SMTP** 操作时，此文件可能非常有用，但是始终启用此属性有可能会略微降低性能。

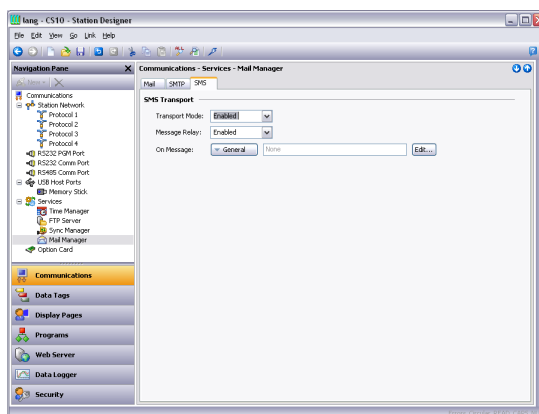
### SMTP 身份验证

- **方法** 属性指示了客户端要尝试的身份验证的类型。选择“摘要式”会始终使用一种以加密形式发送密码的身份验证技术，如果服务器不支持此方法，则会跳过身份验证。选择“基本”会尝试使用安全技术，但会在必要时恢复至细致编码的传输。选择“无”将不会尝试进行身份验证。您的服务器可能会要求进行身份验证，也可能会不要求。请联系您的网络管理员或邮件提供商，详细了解应为您的服务器使用何种设置。
- **用户名和密码** 属性为上述身份验证过程提供了可选凭据。

### 配置 SMS

**SMS** 选项卡用于配置与目标设备一起使用 **GPRS** 调制解调器时支持的短消息服务传输。**SMS** 传输的地址使用国际电话号码格式，但无需前导加号。例如，地址 17175551111 会发送消息给美国境内号码为 (717) 555-1111 的手机或其它 **GSM** 设备。

**SMS** 传输的配置选项如下所示：



- **传输模式** 属性用于启用或禁用传输。
- **消息中继** 属性用于启用或禁用 **Crimson** 的 **SMS** 中继功能。如果启用了此功能，则收到发送给若干收件人的消息的用户可以回复该消息，并且可以

使 **Crimson** 设备运行时将消息中继给其他收件人。这提供了一种在多个消息收件人之间进行简单会议的功能。

- *收到消息时* 属性用于定义每次收到消息时要执行的操作。操作中会定义一个叫做数据的局部系统变量，允许访问消息本身。SMS 的来源号码会放置在消息前面，并使用分号将其与消息分开。



# 共享端口

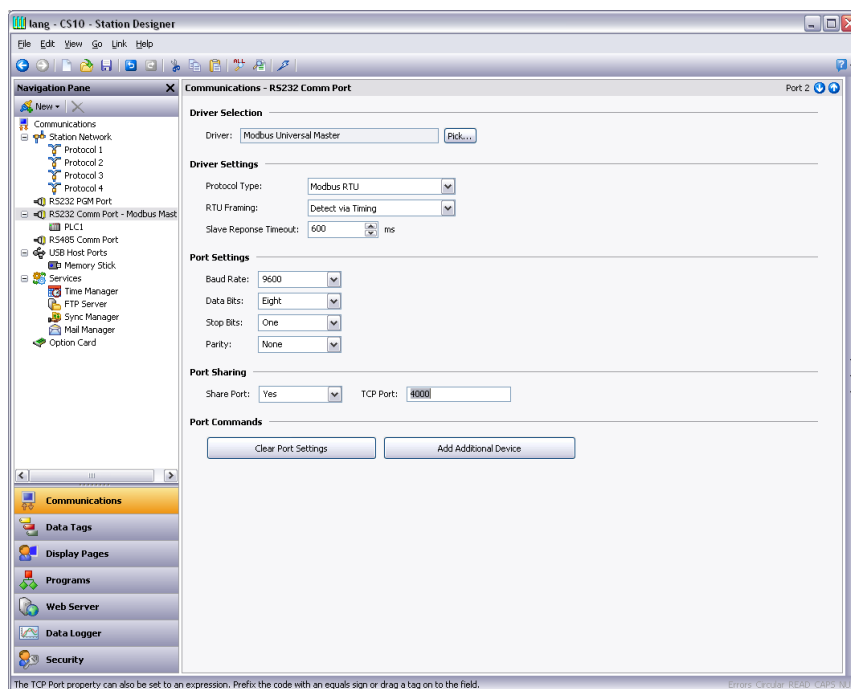
Crimson 支持端口共享功能，此功能允许向任何串行连接的设备进行物理或虚拟串行连接。例如，您可能正在使用配备有可编程控制器的操作员面板，但是因为 PLC 只有一个串行端口，所以修改梯形程序时，您可能需要多次插拔线缆。通过共享通信端口来连接至 PLC，就可以通过另外一个串行端口或使用 TCP/IP 链接创建的连接将数据直接发送到控制器。

## 启用 TCP/IP

使用端口共享时，配置的第一步就是按本手册所述启用以太网端口。虽然可以不选择使用虚拟串行端口功能，但是即使是本地端口共享，也是以 TCP/IP 协议为基础的，除非启用至少一个网络接口，否则 TCP/IP 协议将不可用。

## 共享所需端口

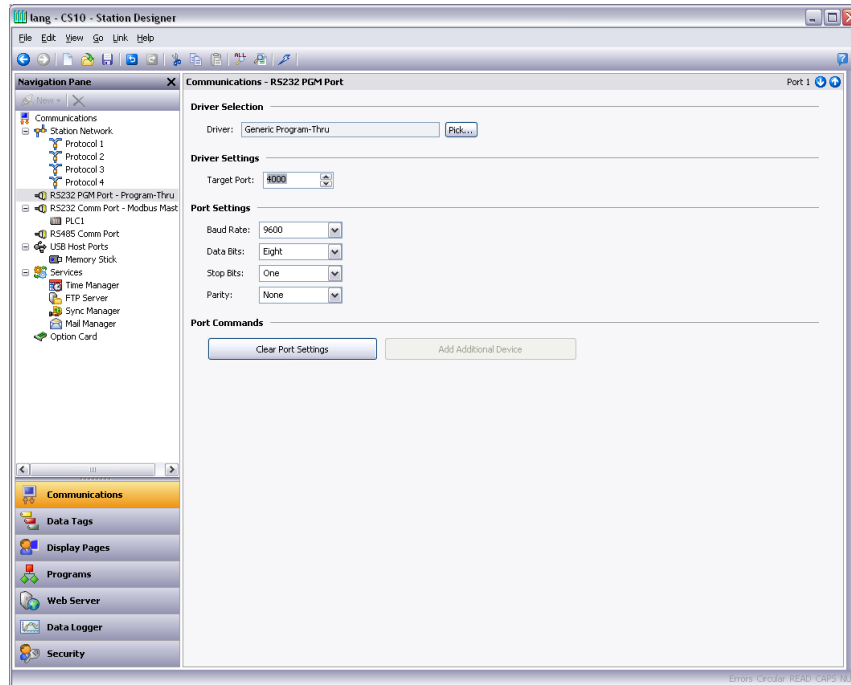
第二步就是共享所需端口，在共享端口属性里选择是，然后输入合适的 TCP/IP 端口号来精确指示应如何为虚拟端口寻址，即可完成此步骤：



如果将端口设置留为零，则会使用 4000 加上端口的逻辑索引。也可以使用任何未被其它 TCP/IP 协议使用的端口号。如果您不知道如何设置，那么我们推荐使用 4000 到 4099 之间的端口号。

## 通过另一个端口连接

如果希望使用目标设备上的另一个端口来将数据路由到共享端口，则必须为该端口选择“通用程序通过”驱动程序，并使用共享端口的 TCP/IP 端口号来配置此驱动程序。下面的示例中，我们正在将来自编程端口的数据路由至通过 RS-232 2 通信端口连接的 PLC：



请注意，大多数情况下，波特率和其它端口设置无需与我们正在共享的端口的设置相同，因为 Crimson 会进行转换。唯一例外就是一个设备传输大量数据但另一个设备没有回应的情况。这种情况下，进行大量传输的设备使用的波特率不得高于接收设备的波特率，否则 Crimson 等待数据被传输时可能会没有足够内存来缓存数据。

上面的示例中，为了使用共享端口，您应将 PC 上的一个备用串行端口连接至目标设备的编程端口，然后将 PLC 编程软件配置为与此 COM 端口会话。只要 PC 开始向 PLC 发送数据，Crimson 和 PLC 之间的所有通信就会被挂起，而且目标设备的两个端口将在软件中连接，这样一来，PC 就会直接与 PLC 对话。如果超过一分钟以上还未传输任何数据，则 Crimson 和 PLC 之间的通信将恢复。

## 通过以太网连接

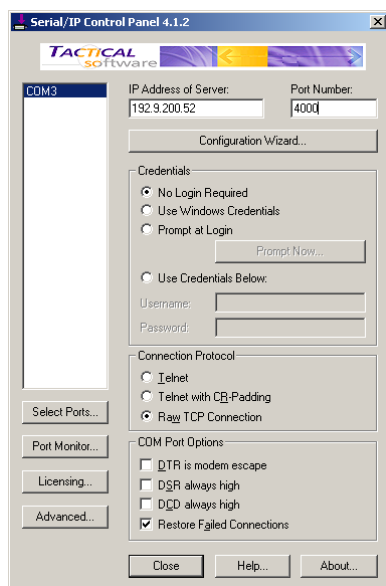
除了使用 PC 上的和基于 Crimson 的设备上的附加串行端口之外，还可以使用第三方实用工具在 PC 上创建虚拟串行端口。这些端口的应用与物理 COM 端口相同，但是它们事实上只是通过 TCP/IP 向和从远程设备发送和接收数据。通过安装一个这样的实用工具，然后将其配置为从基于 Crimson 的设备寻址，就可以在不使用附加线缆的情况下串行访问连接至该设备的任何设备。事实上，PC 上根本无需拥有任何物理串行端口，使用笔记本电脑时，这种功能极有价值，因为 COM 端口选件通常都很昂贵。

有许多可用的第三方虚拟串行端口实用工具。在免费软件阵营里，HW Group (<http://www.hw-group.com>) 提供了一种叫做 HW Virtual Serial Port 的实用工具。还有另

外一些可用的免费端口驱动程序，大多数都是基于相同的源代码。在收费软件阵营里，Tactical Software (<http://www.tacticalsoftware.com>) 提供了 Serial/IP，每个端口大约 100 美元。

虽然各种免费驱动程序无疑都拥有许多满意的用户，但是我们发现这些驱动程序在某些 PC 上偶尔可能会出现稳定性问题。因此，我们仅能支持 Tactical Software 的 Serial/IP，下列信息均假定您正在使用此软件。

要创建一个虚拟串行端口，请打开 Serial/IP 的配置屏幕，然后选择希望定义的 COM 端口的名称。这通常是那些分配给物理端口和安装在 PC 上的调制解调器的 COM 端口之后的第一个自由 COM 端口。接下来，请输入基于 Crimson 的设备的 IP 地址，并输入共享端口时分配的 TCP/IP 端口号。下面的示例按照上文示例的需要进行了配置。最后，请确保选择 Raw TCP Connection，然后关闭 Serial/IP 对话框。



现在您可以将任何基于 Windows 的软件配置为使用新创建的 COM 端口进行下载。软件打开连接时，Crimson 将挂起共享端口上的通信，数据会在 PC 软件和远程 PLC 之间进行交换，就像它们是直接连接在一起那样！端口关闭时，或超过一分钟以上未传输数据时，通信将恢复。

请注意，假设您已经购买了适当数量的 Serial/IP 许可证，那么您就可以创建足够多的虚拟端口。这意味着，您可以从同一台 PC 连接至多个设备，通过各自的编程模块进行下载，而无需插拔任何线缆。使用拥有许多设备的复杂系统时，此功能极为有用。



### 纯粹虚拟端口

某些情况下，您可能会希望使用基于 **Crimson** 的设备上的一个备用串行端口来提供对未在数据库内引用的远程设备的访问，有效地将备用端口用作远程串行服务器。要这样做，请为该端口选择虚拟串行端口驱动程序，并按照常规方式配置端口。然后，如上所述将端口共享，通过 **TCP/IP** 将其公开。虚拟串行端口驱动程序本身不会进行任何通信活动，但仍允许为设备共享远程访问。

### 限制

请注意，某些 **PLC** 编程模块可能无法使用物理共享端口或虚拟共享端口进行操作。下列问题应该多加注意：如果超时过快，**Crimson** 就没有时间将数据中继到 **PLC**；对发送中断信号或操纵硬件握手行过于依赖；如果使用 **DOS** 格式的端口访问，则模块将无法看到虚拟端口。幸运的是，这些问题极少发生，而且大多数模块都会很好地进行通信，就像是直接连接到了 **PLC** 那样。



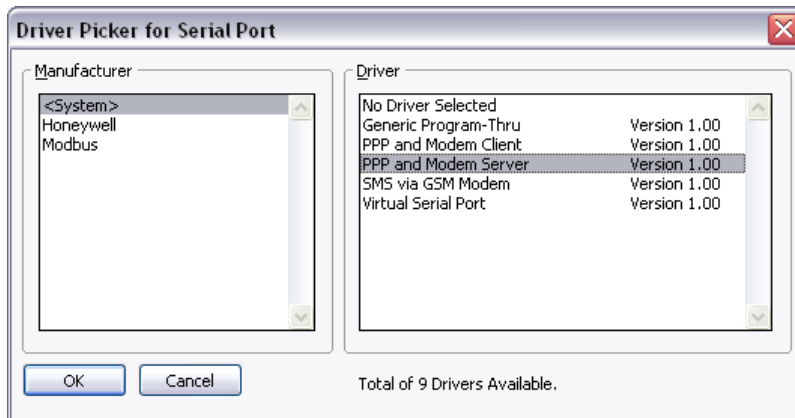
# 使用调制解调器

本章解释了如何对 **Crimson** 进行配置，从而使其能够通过调制解调器来操作，或通过直接串行连接至运行 **Windows** 操作系统的电脑来操作。

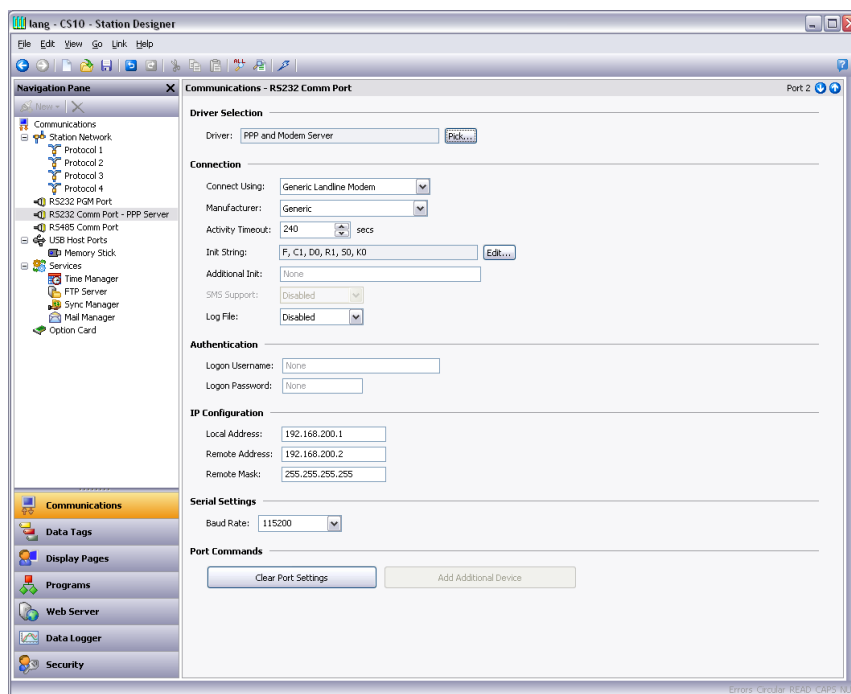
请注意，**Crimson** 的调制解调器支持完全基于点对点协议(**PPP**)。虽然 **Modbus** 之类的协议允许任何两个设备之间发生一个单一对话，但是 **PPP** 更类似于一个以太网连接，因为它允许在一个单一物理链接上存在不限数目的逻辑连接。因此，一个单一 **PPP** 连接可允许同时访问面板的 **TCP/IP** 下载功能、**Web** 服务器、共享串行端口和定义的任何 **TCP/IP** 协议。

## 添加拨入连接

要向数据库添加拨入连接，请选择通信类别，然后导航至要通过其建立连接的串行端口。单击驱动程序的选取按钮，然后从系统部分选择 **PPP 和调制解调器服务器** 驱动程序：



编辑窗格现在会显示调制解调器配置:



调制解调器拥有下列配置选项:

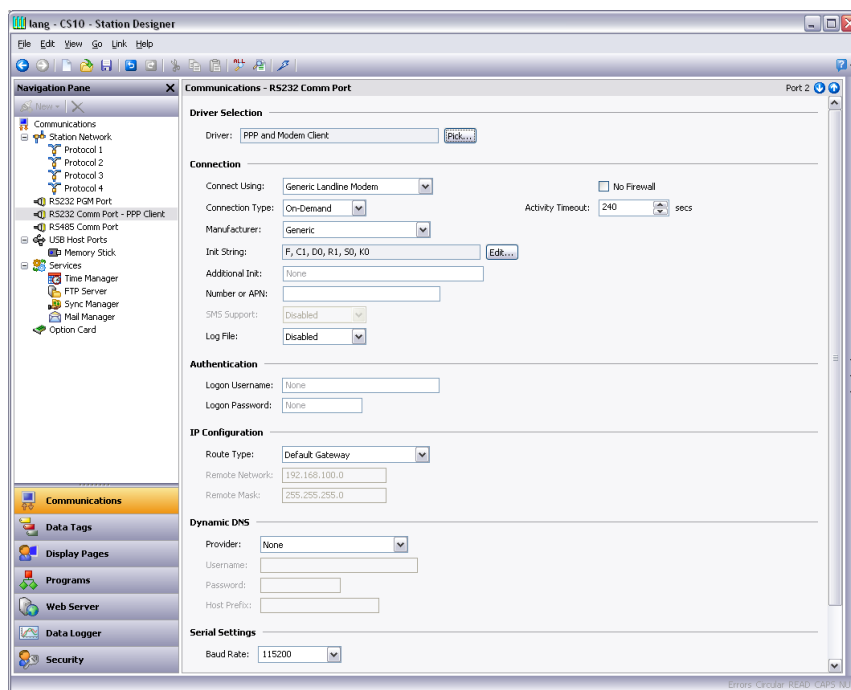
- **连接使用** 属性用于选择要用于进行连接的物理设备。支持的设备为直接串行连接至运行 Microsoft Windows 操作系统的电脑、实现 Hayes 命令集的通用电话线路调制解调器和实现行业标准 GSM 命令的 GSM 调制解调器。对于拨入连接而言，GSM 设备必须配置为在电路交换数据模式下。
- **生产商** 属性用于从为其开发了并在 Crimson 内存储了特定配置的生产商或型号中进行选择。将此设置留为通用，会允许自定义与初始化字符串等相关的设置。关于任何特定调制解调器所需的设置，请咨询技术支持。
- **活动超时** 属性用于定义 G3 未通过 PPP 链接发送任何数据包而导致连接终止的时间量。对于拨入连接而言，假定连接设备是友好设备，所以不会为了过滤可能导致链接长时间活跃的可选数据包而费力。请注意，即使想要永久连接，也必须输入合适的超时，从而允许侦测无效链接。这意味着所谓的永久连接也可能偶尔失效，但是无论在何种情况下都会立刻重新建立。
- **初始化字符串** 属性用于启用或禁用初始化序列中的某些命令。如果在生产商属性里输入了特定设置，则会自动配置此属性。
- **附加初始化** 字符串用于非直接链接，提供了一系列用于初始化调制解调器的 AT 命令。不需要初始 AT 前缀。只需将一个命令放在另一个命令之后，即可将若干命令组合在一起。您的调制解调器所需的精确字符串依赖于其内部软件，所以，如果您联系技术支持寻求帮助，请确保提供精确的品牌和型号信息。

- *SMS 支持* 属性用于在使用 GSM 调制解调器时启用短消息服务(SMS)。为使 SMS 消息传送正常操作，还必须在本手册描述的邮件服务器里启用 SMS 传输。
- *登录用户名* 和 *登录密码* 属性用于定义远程客户端为了连接至此设备而必须提供的凭据。用户名不区分大小写，而密码则区分大小写。Crimson 的 PPP 执行会要求其对等机使用 CHAP 验证，从而避免传送或接受纯文本密码，但是，如果远程客户端不支持 CHAP，则会返回至使用 PAP。
- *本地地址* 属性用于定义要分配给连接的本地端的 IP 地址。这将是 G3 的此链接的 IP 地址。请注意，这不能与 G3 的以太网端口的 IP 地址相同，因为每个物理 IP 接口均必须拥有一个独一无二的 IP 地址。默认值在大多数情况下都能使用，除非您的网络设计需要使用不同的设置。
- *远程地址* 属性用于定义要分配给连接的远程端的 IP 地址。它与 *远程掩码* 属性一起用于确定哪些数据包将被路由至此连接。对于大多数应用程序而言，将使用掩码 255.255.255.255，从而指示 Crimson 仅通过此接口发送目标直接为远程客户端的数据包。相反，掩码 0.0.0.0 则允许向远程客户端转发未特定匹配另一个接口的全部数据包，可能是为了将来向预期主机进行转发。可使用中间掩码来控制具体要发送哪些数据包。

### 添加拨出连接

除了应为所需端口选择 *PPP* 和 *调制解调器客户端* 驱动程序之外，拨出连接的添加方法与上文描述的完全相同。

此调制解调器的配置选项如下所示：



此调制解调器拥有下列与拨入连接不同的属性：

- **连接使用** 属性与拨入连接的相同，但是添加了对通过 GSM 调制解调器进行 GPRS 连接的支持。这些连接与 CSD 连接不同，因为它们的速度更快，并通常基于传递了多少数据而不是进行了多久连接来收费。因此，GPRS 连接可配置为永久连接，除非需要提供停机时间来允许传输 SMS 消息。
- **无防火墙** 属性用于关闭为拨出连接提供的防火墙保护。这种保护阻止了对此接口进行的入站连接，并阻止了 G3 发送某些诊断数据包，这些数据包可能会向黑客提供系统信息，也可能被攻击者用于在没有实际数据传输时保持连接活跃。如果您正通过此连接直接连至了 Internet，则不应关闭防火墙。 仅应为向公司网络或其它受控环境进行的连接禁用防火墙。
- **连接类型** 属性用于指示您是否希望永久维持此连接，或是否希望试图通过此接口向可到达的主机传输数据时自动建立此连接。如果选择按需连接，则必须指定超时，此超时之后如果 G3 没有传递任何数据包，链接将终止。
- **登录用户名** 和 **登录密码** 属性用于定义试图初始化此连接时要传递给远程服务器的凭据。用户名不区分大小写，而密码则区分大小写。Crimson 的 PPP 执行会要求其对等机使用 CHAP 验证，从而避免传送或接受纯文本密码，但是，如果远程客户端不支持 CHAP，则会返回至使用 PAP。
- **路由类型** 属性用于定义将通过此接口传输的数据。对于按需连接而言，这有效地定义了何时激活连接。如果选择了默认网关，则不匹配以太网连接的地址和网络掩码的数据包将被发送至此接口。请注意，此模式下，以太网端口的网关设置必须为 0.0.0.0，否则它将带走全部数据包，而不会留下

任何数据包来激活调制解调器！如果选择了*特定网络*，则必须提供定义数据包将路由至的网络的地址和网络掩码。

### 添加 SMS 连接

需要文本消息传送功能但没有建立拨入和拨出连接时，就会使用 SMS 连接。除了应为所需端口选择*通过 GSM 调制解调器的 SMS* 设备之外，它们的配置方法与上文描述的相同。

此驱动程序的属性是拨入连接的属性的子集。始终为此驱动程序启用 SMS 支持，但是请再次注意，为使 SMS 消息传送正常操作，必须在邮件管理器里启用 SMS 传输。

### SMS 消息处理

启用了 SMS 消息传送之后，Crimson 将指示 GSM 调制解调器每隔五秒检查一次新入站和出站消息。入站消息会被转发至邮件管理器，管理器可根据其配置将消息转发至其它用户。请注意，调制解调器连接至 CSD 或 GPRS 会话时，无法检查消息，所以，使用 SMS 时，应避免使用永久连接。还请注意，如果配置了多个 GSM 调制解调器，则它们均可以接收消息，但是只会使用最后一个调制解调器进行发送。

### 检查调制解调器状态

为了协助调试调制解调器连接，Crimson 提供了 `GetInterfaceStatus()` 函数。此函数会取一个单一参数，此参数是所需接口的数字索引。接口零为内部环回接口，接下来是任何启用的以太网接口，最后则是 PPP 接口。例如，在使用单一以太网端口的系统内，第一个 PPP 接口的索引是 1。

此函数会返回一个可根据下表进行解释的字符串：

状态	含义
CLOSED	接口尚未初始化。此状态只会在系统启动期间短暂出现。
INIT	调制解调器正在初始化。如果连接停留在此状态之内，则正在发送给调制解调器的初始化字符串中可能存在错误。
IDLE	链接现在闲置。GSM 调制解调器将在字符串结尾返回一个数字，指示信号强度。下一个表格说明了如何解释这些值。
SMS	调制解调器正在发送 SMS 消息，或正在轮询调制解调器，从而查看是否有可用的新 SMS 消息。如果为调制解调器启用了 SMS 消息传送，则每隔五秒此状态就会短暂出现一次。
CONNECTING	调制解调器正在建立连接。通常只有客户端连接会出现此状态，指示了正在建立通话。
LISTENING	调制解调器正在等待通话。只有服务器连接会出现此状态。请注意，GSM 调制解调器等待通话时也将返回一个 IDLE 状态，从而显示信号强度。

状态	含义
ANSWER	调制解调器正在答复通话，并试图为连接协商波特率。只要服务器连接会出现此状态。如果已建立连接，调制解调器将进入 CONNECTED 状态。
CONNECTED	调制解调器已建立连接。此状态只会短暂出现，因为稍后会很快开始 LCP 协商流程。
NEG LCP	连接正在协商 LCP 选项。此流程决定了客户端和服务端均可接受的链接协议设置集。
AUTH	连接正在进行验证流程，从而确保使用了适当的用户凭据。
NEG IPCP	连接正在协商 IPCP 选项。此流程决定了客户端和服务端均可接受的网路协议设置集。
UP	连接现在活跃，且 IP 数据可被交换。
HANGING UP	调制解调器正在断开连接。此状态只会在调制解调器返回至 IDLE 状态之前短暂出现。

GSM 调制解调器返回的信号强度值含义如下：

值	信号强度
0	-113dBm 或更小。
1	-111dBm。
2-30	-109dBm 至 -52dBm，以 2dBm 递增。
31	-51dBm 或更大。
99	无法确定信号强度。



显示信号强度时，移动电话通常如下对这些值进行解释：

值	强度	信号栏数目
5 或更小。	-103dBm 或更小。	一
6 至 9。	-101dBm 至 -95dBm	二
10 至 14。	-93dBm 至 -85dBm	三
15 或更大。	-83dBm 或更大。	四

### 调制解调器通信故障排除

各种调制解调器驱动程序均提供一个 *日志文件* 属性，会将与调制解调器进行的交换记录在 CompactFlash 卡上的文件里。调制解调器初始设置期间或试图查找适当配置选项时，此文件供调试之用。创建正确的调制解调器配置序列之后，请确保禁用此功能。

### 使用多个接口

Crimson 支持最多两个调制解调器独立连接。与目标设备提供的一个或两个以太网端口结合使用时，这可以实现最多四个不同的 IP 接口，全部都会根据为每个连接定义的配置参数进行操作。本章节说明了这些多个接口如何交互，以及 Crimson 如何决定向哪里发送每个数据包。

#### 接口选择

每个接口都有一个 IP 地址和一个网络掩码，它们用于决定是否将数据包转发至该接口。例如，如果一个以太网端口的 IP 地址配置为 192.168.1.0，网络掩码配置为 255.255.255.0，则目标为以 192.168.1 开头的 IP 地址的任何数据包均将被发送至此接口。同样，如果一个按需调制解调器连接的远程 IP 地址是 192.168.2.2，网络掩码是 255.255.255.255，则向地址 192.168.2.2 发送一个数据包会导致建立连接。

请注意，这种机制仅将向一个单一接口发送一个数据包。这意味着不同接口应该拥有不同的网络地址，网络地址通过对接口的 IP 地址和网络掩码进行与运算来定义。如果违反了此要求，数据包将不会被路由至拥有该网络地址的第二个接口，而且该端口上的通信将失败。例如，不得将一个以太网端口配置为 192.168.100.1 且将另外一个配置为 192.168.100.2，因为目标为 192.168.100.0 网络的数据包只会被发送至第一个端口。

## 默认路由

此外，一个单一接口还可定义一个默认路由，此路由将用于处理未特定匹配任何其它接口的数据包。接口类型不同，则路由配置方法也会不同，请参阅下表：

接口	要定义默认路由
以太网	请为 <i>网关</i> 属性输入一个非零值。
拨入	请为 <i>远程掩码</i> 输入 0.0.0.0。
拨出	请为 <i>路由类型</i> 属性选择默认网关。

只有一个单一接口可定义一个默认路由。例如，同一个操作员面板可能被连接至多个使用 IP 地址 192.168.1.0 和网络掩码 255.255.255.0 且未定义网关的以太网设备。可配置一个按需调制解调器连接来访问 Internet 服务提供商，从而发送警报电子邮件。它的 *路由类型* 被设为默认网关，使它成为任何目标为不匹配为以太网端口定义的网络的数据包的路由。SMTP 服务器被配置为 24.104.0.39，这样一来，试图发送消息时，就会建立一个拨出连接。

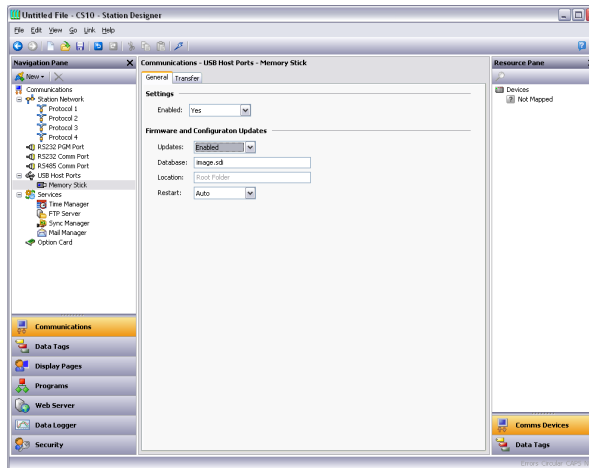
# 使用 USB 主机

如果您的目标设备拥有一个或多个 USB 主机端口，则通信类别内的相应图标可用于配置它将支持的设备。当前版本的 Crimson 仅支持 USB 内存设备。

## 内存条支持

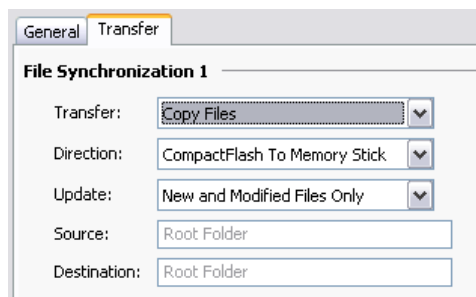
USB 内存设备通过内存条图标进行配置：

### 常规属性



- *启用* 属性用于全局启用或禁用内存条支持。
- *更新* 属性用于配置将新固件和数据库自动传输至 CompactFlash 卡的根目录。
- *数据库* 属性定义了要复制到 CompactFlash 卡上的 image.ci3 文件里的数据库映像的名称。此设置允许将多个文件放入同一个内存条，每个 Crimson 设备将复制适用于其自身应用程序的文件。
- *位置* 属性指定了内存条里可存放上文指定的数据库映像的位置。
- *重新启动* 属性用于指示文件复制完成之后是否应自动重新启动。启用此属性将允许 Crimson 立即加载数据库映像里的信息。

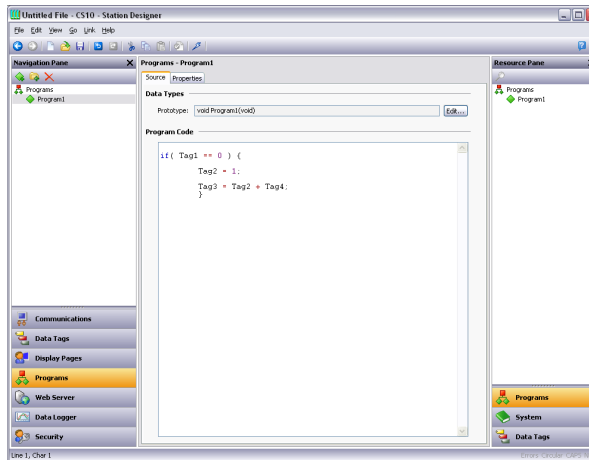
## 传输属性



- 每个同步分组的 *传输* 属性均定义了要执行的函数。信息可复制或移动，操作可应用至指定文件夹里的文件，也可以递归为基础附加应用至子文件夹及其内容。
- *方向* 属性指定了传输的方向。
- *更新* 属性用于指示是否应复制目标设备内业已存在的文件，或是否应只传输新文件和修改了的文件。Crimson 使用文件的时间戳和大小来确定是否应处理文件。
- *源* 和 *目标* 属性用于指示源和目标设备上存放文件的文件夹。

# 使用程序

本手册前面的章节介绍了使用操作来响应按键或按下触摸屏或数据标记更改。如果要执行无法显示在一行之内的或需要复杂决策逻辑的操作，那么就可以使用编程类别来创建并操纵程序。



## 程序列表

导航窗格里的程序列表是一个常规的导航列表，可用于创建、删除、重命名和组织程序。请注意，程序可分组进文件夹，每个程序的图标可显示三种状态：绿色指示程序已翻译并验证；黄色指示程序已编辑但尚未翻译；红色指示程序包含一个或多个错误。

## 查找程序使用

在导航窗格里右键单击给定程序并选择查找使用命令，即可查找引用该程序的全部项。结果项将放入全局搜索结果列表，并可使用 **F4** 键和 **SHIFT+F4** 组合键进行访问。按下 **F8** 键可以隐藏或显示此列表。

## 编辑程序

要编辑一个程序，只需使用编辑窗格里显示的源选项卡来编辑程序文本。您会注意到，只要开始键入，程序的图标就会变成黄色，指示您进行了尚未翻译的更改。您还会注意到，Crimson 的程序编辑器还会执行语法着色、自动缩进和其它一系列适用于代码编辑器的功能。通过右键单击编辑窗格然后从出现的菜单中选择适当命令，即可配置编辑器选项。

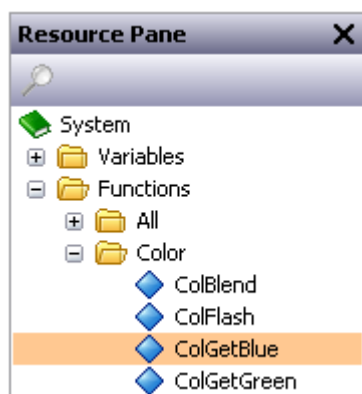
程序编写完成之后，请按下 **CTRL+T** 组合键或选择工具栏上的翻译按钮。然后会检查程序中是否存在错误。如果发现了错误，则会出现一个对话框，并且程序的图标会变成红色。光标也会被移到错误出现的位置。如果没有错误，会响起钟声，并且程序的图标会变成绿色，指示程序已翻译成适于在目标设备内执行的形式。

## 获取帮助

在编辑窗格内操作时，可以使用一个快捷方式来获取与系统函数相关的帮助。将光标放在函数名称中间或末尾，然后按下 **F1** 键，就可以显示与函数操作、参数和返回类型相关的信息。也可以在键入函数名称后按下 **F1** 键，从而访问这些信息。

## 资源窗格

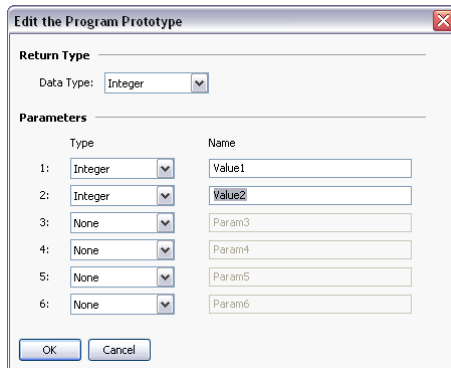
程序编辑器显示的资源窗格包含了一系列可以拖入您的代码的项。数据标记类别和程序类别不言自明，允许将项名称插入编辑器，可快速访问各个项。通过系统类别可以访问 Crimson 的广泛的系统变量和函数库：



正如您看到的那样，变量和函数被分组进了类别。选定一个函数是，它的返回类型和参数类型会显示在状态栏上。将一个函数放入您的代码会输入适当文本，并将文本光标放入函数名称后面的括号内，从而允许您输入所需参数。

## 程序数据类型

程序编辑器上的字段可用于编辑程序的数据类型：

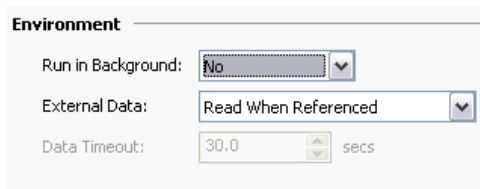


- **数据类型** 属性用于指示此程序是否应仅执行一系列操作，或是否应执行一次计算并将计算值返回给调用方。会返回值的程序不能在后台运行。
- **参数** 属性部分定义了程序将接受的最多六个参数。每个参数都有一个名称和一个数据类型。此示例中，程序接受了两个参数，第一个叫做 Value1，第二个叫做 Value 2，两个都是 32 位有符号整数。

下面会详细讨论值的返回和参数的传递。

## 程序属性

编辑器的第二个选项卡定义了程序的执行环境：



- **后台运行** 属性用于指示 Crimson 是否应等待程序执行完成之后再继续执行其它任务。例如，如果此属性设为否，那么为响应按键而运行程序会导致显示更新出现暂停，直至程序完成。（因为大多数程序的执行时间都极为短暂，所以这可能根本注意不到。）如果此属性设为是，则显示更新会继续，程序则会以低优先级在后台执行。一次只能执行一个后台程序，所以会将后续请求排入队列，稍后执行。请注意，会返回值的程序不能在后台运行，因为如果后台运行，则调用方将无法得到它们的返回值！

- *外部数据* 和 *超时* 属性用于控制程序如何就程序引用的外部数据项与 Crimson 的通信基础架构进行交互。您可以回想到 Crimson 仅会读取使用的数据项。此属性用于控制此规律在程序方面的确切解释：

模式	行为
被引用时读取	只要程序被引用，程序使用的外部数据就会被添加到通信扫描。如果程序被显示页面引用，则显示页面时将读取数据；如果程序被全局操作或触发器引用，则将始终读取数据。这是默认模式，可适用于除了使用极大量外部数据的程序之外的所有程序。
始终读取	无论程序是否被引用，程序使用的外部数据始终均将被读取。这意味着程序可随时运行，而且操作员不会看到其它模式下程序第一次被引用时会出现的“未就绪”消息。此模式的缺点则是如果程序引用了大量数据，通信性能就可能会降低。
被执行时读取	只有程序被调用时才会读取程序内使用的外部数据。程序会等待超时属性内定义的时间量，从而等待这些数据可用。如果数据无法被读取（可能是由于设备离线），则程序就不会执行。使用大量数据的全局引用程序通常使用此模式，如果使用其它模式，就会降低通信扫描速度。
读取并仍然执行	会按照与始终读取模式相同的方式来处理外部数据，但是无论数据是否被成功读取，程序都将执行。因此，操作员永远都不会看到“未就绪”消息。但是，如果设备离线，就不能保证程序的数据项包含有效数据。



## 添加注释

有两种方法可以向程序添加注释。首先，可以使用“//”序列来引导在当前行内继续的一个单行注释。其次，可以使用“/\*”序列来引导一个单行或多行注释。“\*/”序列出现之前，此注释会一直继续。下面的示例显示了这两种注释方式：

```
// 这是一个单行注释

/* 这是注释的第 1 行
   这是注释的第 2 行
   这是注释的第 3 行*/
```

单行注释也可放在程序代码行后面。

## 返回值

如上所述，程序可以返回值。此类程序可在数据库内任何地方被其它程序或表达式调用。例如，如果希望按照马达的不同状态来执行特别复杂的解码并返回指示当前状态的值，那么可以创建一个与下面示例类似的返回整数的程序：

```
if( MotorRunning )
    return 1;
else {
    if( MotorTooHot )
        return 2;
    if( MotorTooCold )
        return 3;
    return 0;
}
```

然后可以配置一个标记来调用此程序，并使用多状态格式来提供各种状态的名称。将标记的值属性设为 Program()，即可执行调用，其中 Program 是相关程序的名称。括号用于指示函数调用，不能省略。

### 注意！

请注意，使用程序来返回值时，必须小心谨慎。尤其应避免使用持续时间较长的循环，或执行在调用函数的上下文中没有意义的操作。例如，如果上面的代码片段调用 GotoPage 函数来更换页面，则每次程序被调用时显示均会更改。但是请想象一下，如果试图从关联的标记记录数据，那么会产生什么样的结果？之后您就会意识到这不是明智之举，因此，会返回值的程序一定要简单，并应始终考虑程序执行的上下文。如果存在疑问，就尽量使用简单的数学计算和 if 语句。

## 传递参数

如上所述，程序可以接受参数。假设您希望编写一个叫做 `FindMean` 的程序，来求两个整数值的平均数。程序会被配置为接受两个整数参数，亦即 `a` 和 `b`。程序还会被配置为返回一个整数。程序代码会被定义为：

```
return (a+b)/2;
```

创建并翻译此程序之后，您就可以输入一个 `FindMean(Tag1, Tag2)` 之类的表达式，来使用适当的参数来调用此程序。这种情况下，表达式将等于 `Tag1` 和 `Tag2` 的平均数。

## 编程技巧

下面的部分概述了 `Crimson` 支持的编程结构。使用的基本语法就是 `C` 编程语言的语法。请注意，我们的目的并不是要教您怎么变成程序员，也不是要教您掌握 `C` 语言的精妙。本手册并不会讲述这些话题。我们的目的是提供可用功能的快速概览，这样有兴趣的用户就可以进一步研究。

### 多个操作

类型最简单的程序由多个操作组成，每个操作独占一行，以分号结尾。可使用编写操作章节内定义的各种操作。如果多个操作结合成的单一操作定义不可读取，就会使用下面这样的简单编程。

下面的示例设置了若干变量，然后会更改显示页面：

```
Motor1 = 0;  
Motor2 = 1;  
Motor3 = 0;  
  
GotoPage(Page1);
```

这些操作会按顺序执行，然后程序会返回调用方。

### IF 语句

此类型的语句用于在程序内进行判断。它包括一个包含在括号内的 `if` 语句的条件，后面则是条件为真时要执行的一个（或多个）操作。如果指定了多个操作，则每个操作应独占一行，并应使用大括号来标出整个语句。还可使用可选的 `else` 子句来提供条件为假时要执行的代码。

下面的示例显示了一个有一个操作的 `if` 语句：

```
if( TankFull )
    StartPump = 1;
```

下面的示例显示了一个有两个操作的 `if` 语句：

```
if( TankEmpty ) {
    StartPump = 0;
    OpenValue = 1;
}
```

下面的示例显示了一个有 `else` 子句的 `if` 语句：

```
if( MotorHot )
    StartFan = 1;
else
    StartFan = 0;
```

请注意，一定要在语句的 `if` 部分或 `else` 部分中要执行的操作分组两侧添加大括号。如果省略了大括号，**Crimson** 可能就无法理解您希望哪些操作依赖于 `if` 条件。虽然推荐在各个操作之间进行换行，但是换行无法用于指定哪些操作要包含在条件语句内。

## SWITCH 语句

`Switch` 语句用于将一个整数值与多个可能的常量进行比较，然后根据匹配的值执行一个操作。除了下面的示例中显示的之外，精确语法还支持许多其它选项。但是，对于大多数应用程序而言，下面的简单形式均可适用。

下面的示例会启动 `MotorIndex` 标记中的值选定的马达：

```
switch( MotorIndex ) {  
  
    case 1:  
        MotorA = 1;  
        break;  
  
    case 2:  
    case 3:  
        MotorB = 1;  
        break;  
  
    case 4:  
        MotorC = 1;  
        break;  
  
    default:  
        MotorD = 1;  
        break;  
  
}
```

值 1 会启动马达 A，值 2 或 3 会启动马达 B，值 4 会启动马达 C，未明确列表的其它值则会启动马达 D。与所用语法相关的一些信息值得注意：在 `case` 语句两侧要使用大括号，要使用 `break` 来结束每个条件块，要使用两个连续的 `case` 语句来匹配多个值，可使用可选的 `default` 语句来指示指定的值与控制表达式的值均不匹配时要执行的操作。（如果这些语法看起来太过复杂，则可使用一系列 `if` 语句来产生同样的效果，但是性能会略微降低，而且可靠性也会降低。）

## 局部变量

一些程序会使用变量来存储中间值，或控制下面描述的循环结构。您无需定义标记来持有这些值，可以使用下面显示的语法来声明局部变量：

```
int     a;           // Declare local integer 'a'  
float   b;           // Declare local real   'b'  
cstring c;          // Declare local string 'c'
```

声明局部变量时，在变量名称后面添加 `=` 和要分配的值，就可以初始化局部变量。未通过此方式初始化的变量会按需被设为零或空字符串。

请注意，局部变量在范围和生存时间内均为局部。这意味着它们不能在程序之外被引用，而且在多次函数调用之间不能保留值。如果函数被按递归方式调用，则每次调用都会拥有自己的变量。

## 循环结构

三种不同循环可用于在条件为真时执行给定部分的代码。`while` 循环在代码执行之前检查条件，`do` 循环则在之后检查条件。`for` 循环是定义 `while` 循环的快捷方式，允许将三个基本元素结合到一个语句之内。

请注意，在程序内使用循环时应该小心谨慎，因为可能会出现导致循环无法终结的编程错误。根据程序被调用的不同情况，这可能会导致终端的用户界面活动或通信中断。循环次数过多也可能导致调用循环的子系统性能降低。

### WHILE 循环

只要 `while` 语句里的条件仍然为真，此类循环就会不断重复它后面跟随的操作。如果条件永远不为真，则操作永远都不会被执行。如果希望在循环内包含多个操作，请确保使用大括号括起多个语句，这与 `if` 语句相同。下面的示例初始化了一对局部变量，然后使用第一个变量在数组的内容中进行循环，将前十个元素相加，并将总值返回给调用方：

```
int i=0, t=0;

while( i < 10 ) {
    t = t + Data[i];
    i = i + 1;
}

return t;
```

下面的示例显示了同一个程序，但是以压缩格式进行了重写。因为现在循环语句只控制了一个单一操作，所以省略了大括号：

```
int i=0, t=0;

while( i < 10 )
    t += Data[i++];

return t;
```

### FOR 循环

您可以注意到上面显示的 `while` 循环拥有四个元素：

1. 初始化循环控制变量。
2. 执行确定循环是否应继续的检查。
3. 执行要由循环执行的操作。
4. 向控制变量做出更改。

for 循环允许将元素 1、2 和 4 结合到一个单一语句之内，这样一来，语句后面的操作就只需要执行元素 3。此语法可实现与 BASIC 语言和其它语言中的 FOR-NEXT 循环类似的结果。

使用此语句之后，上面给出的示例可以重写为：

```
int i, t;

for( i=t=0; i<10; i++ )
    t += Data[i];

return t;
```

您会注意到，for 语句包含三个不同的元素，元素之间使用分号隔开。第一个元素是初始化步骤，循环首次开始时会被执行一次；第二个是条件，会在每次循环调用开始时被检查，从而确定循环是否应继续；第三个则是归纳步骤，用于对控制变量进行更改，从而进入下一次循环调用。同样，如果希望在循环内包含多个操作，请确保使用大括号！

## Do 循环

除了条件是在循环的结尾检查的之外，此类型的循环与 while 循环类似。这意味着循环始终会执行最少一次。

下面的示例显示了使用 do 循环重写的上面的示例：

```
int i=0, t=0;

do {
    t += Data[i];
} while( ++i < 10 );

return t;
```

## 循环控制

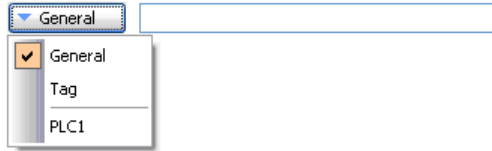
有两个附加语句可在循环里使用。Break 语句可用于提前结束循环，continue 语句则可用于跳过循环主体的剩余部分，不执行后续代码即开始另一次调用。这两个语句必须与 if 语句合用才有意义。

下面的示例显示了一个另一个程序返回真值时就会提前结束的循环：

```
for( i=0; i<10; i++ ) {
    if( LoopAbort() )
        break;
    LoopBody();
}
```

# 编写表达式

本手册前面的章节内我们讲到 **Crimson** 内的许多字段都配置为表达式，而且这些字段通过与下面类似的用户界面元素来进行编辑：



许多情况下，您会将这些属性设定为等于标记的值，或等于远端通讯设备中的寄存器的内容。这些情况下，只需从资源窗格拖动项，或单击下拉菜单中的适当选项，然后从出现的对话框中选则项即可。

但是，某些情况下，您也可能会希望使一个属性依赖于更为复杂的数据项集合，也许是运用一些数学计算来组合或比较它们的值。这时您就需要通过表达式来处理这些情况，只要在下拉菜单中选定了常规模式，就可以在属性的编辑框中输入表达式。

## 数据值

所有表达式都包含最少一个数据值。因此，最简单的表达式就是对单一常量、单一标记或单一寄存器进行的引用。如果输入了后两种选项之一，那么 **Crimson** 就会自动切换至适当的属性模式，从而简化编辑过程。例如，如果在常规模式下输入一个标记名称，那么 **Crimson** 就会切换至标记模式，在选择字段内显示标记名称。

### 常量

常量表示常数或常量字符串。

#### 整数常量

整数常量表示一个有符号的 32 位数。它们可以是十进制、二进制、八进制或十六进制。下面的示例显示了同一个数字在四种不同进制中的表达：

进制	示例
十进制	123
二进制	0b1111011
八进制	0173
十六进制	0x7B

不会使用早期软件版本支持的‘U’和‘L’。

## 字符常量

字符常量表示一个编码在 32 位数的低 16 位中的单一 Unicode 字符。一个字符常量由包含在单引号内的一个字符组成，这样一来，'A' 就可以表示值 65。某些无法打印或表示的字符则可使用转义序列进行编码，每个转义序列都由一个反斜杠引导：

序列	值	ASCII
<code>\a</code>	十六进制 0x07, 十进制 7	BEL
<code>\t</code>	十六进制 0x09, 十进制 9	TAB
<code>\n</code>	十六进制 0x0A, 十进制 10	LF
<code>\f</code>	十六进制 0x0C, 十进制 12	FF
<code>\r</code>	十六进制 0x0D, 十进制 13	CR
<code>\e</code>	十六进制 0x1B, 十进制 27	ESC
<code>\xnn</code>	<i>nn</i> 表示十六进制值。	-
<code>\unnnn</code>	<i>nnnn</i> 表示十六进制值。	-
<code>\nnn</code>	<i>nnn</i> 表示八进制值。	-
<code>\\</code>	一个反斜杠字符。	-
<code>\'</code>	一个单引号字符。	-
<code>\"</code>	一个双引号字符。	-

## 逻辑常量

逻辑常量表示用于指示一个是非表达式的真或假的 1 值或 0 值。一个表示 PLC 数字输出的标记就可分配为等于逻辑常量。可简单使用 1 或 0 来输入逻辑常量，也可使用关键字 true 或 false 来输入。

## 浮点常量

浮点常量表示一个 32 位单精度浮点值。它们由整数部分、整数部分后面的小数点和小数点后面的小数部分组成。通过指定一个尾数值然后添加一个'E'和一个指数，还可支持科学计数法。

## 字符串常量

字符串常量表示字符序列。它们由包含在双引号内的要表示的字符组成。例如，字符串"ABCD"表示了一个四个字符的字符串，包含了值 65、66、67 和 68。（事实上，使用了五个 16 位的字来存储字符串，而且在字符串末尾添加了一个空值。）上文讨论的各种转义序列也可在字符串内使用。



## 标记值

标记的值由标记名称以表达式来表示。被组织进文件夹的标记由标记的路径名称来表示，每对元素之间由句点隔开。因此，叫做 **Loop** 的文件夹里的叫做 **PV** 的标记会被引用为 **Loop.PV**。请注意，查找所需标记时，不区分大小写。表达式输入之后，对标记名称进行的任何更改均会修改引用该标记的全部表达式。

## 标记属性

数据标记拥有一些属性，在标记名称后面加上一个句点和所需属性的名称，就可以访问这些属性。定义了下列属性：

属性	描述	数据类型
Name	标记的名称。	字符串。
AsText	标记的格式被设为文本的值。	字符串。
Label	标记的标签属性。	字符串。
Desc	标记的描述属性。	字符串。
Prefix	标记的格式定义的前缀。	字符串。
Units	标记的格式定义的单元。	字符串。
SP	标记的设定值属性。	与标记相同。
Min	标记的数据输入底限。	与标记相同。
Max	标记的数据输入上限。	与标记相同。
Fore	标记的当前前景色。	整数。
Back	标记的当前背景色。	整数。

## 页面属性

显示页面也有一些可以通过相同方式来访问的属性：

属性	描述	数据类型
Name	页面的名称。	字符串。
Label	页面的标签属性。	字符串。
Desc	页面的描述属性。	字符串。

## 通信引用

通过使用包含左方括号、寄存器名称和右方括号的语法，即可将对主通信设备内的寄存器的引用输入到表达式中。还可在寄存器名称前面添加设备名称，并用句点隔开。引用数据库内的唯一设备时，无需输入设备名称。

下面的示例使用了这种语法：

示例	含义
[D100]	第一个设备里的寄存器 <b>D100</b> 。
[AB.N7:0]	设备 <b>AB</b> 里的寄存器 <b>N7:0</b> 。

示例	含义
[FX.D100]	设备 FX 里的寄存器 D100。

## 简单运算

如上所述，表达式经常包含多个数据值，它们的值通过数学运算符结合起来。这些表达式中最简单的可以添加一对值，而更为复杂的表达式则可获取三个值的平均数。这些运算使用与 Excel 等应用程序中类似的语法来进行。下面的示例显示了一些可进行的基础运算：

运算符	优先级	示例
加	分组 4	<b>Tag1 + Tag2</b>
减	分组 4	<b>Tag1 - Tag2</b>
乘	分组 3	<b>Tag1 * Tag2</b>
除	分组 3	<b>Tag1 / Tag2</b>
余	分组 3	<b>Tag1 % Tag2</b>

虽然上面的示例中运算符两侧都有空格，但是事实上并不需要空格。

## 运算符优先级

您可能已经注意到上表中的优先级列。从代数知识中可以知道，同时使用多个运算符时，会按照固定顺序来执行这些运算符。例如，乘法始终在加法之前运算。Crimson 通过运算符优先级来实现这样的顺序。每个运算符均被放在一个分组之内，然后按照分组数字从低到高应用运算符。除非文本中另有说明，否则同一分组内的运算符的应用顺序为从左到右。可以使用括号来重写默认运算顺序。

## 类型转换

Crimson 通常会自动决定何时从使用整数来运算表达式切换为使用浮点来运算表达式。例如，如果用一个浮点值来除一个整数值，进行除法之前，整数会被转换为浮点数。但是，某些情况下你也许会希望强制进行转换。

例如，假设对三个表示三个箱中的液位的整数值进行相加，然后使用箱总数来除总值，从而求出平均液位。如果使用  $(\text{Tank1} + \text{Tank2} + \text{Tank3}) / 3$  之类的表达式，那么所得结果可能不够精确，因为除法会使用整数来进行，所以求出的平均数不会包含任何小数位。要强制 Crimson 使用浮点来求值，最简单的方法就是将 3 更改为 3.0。稍微有些复杂的方法则是使用 `float(Tank1+Tank2+Tank3)/3` 之类的语法。这会调用术语上称之为“类型转换”的操作，手动将其转换为浮点值。

类型转换也可用于将浮点值转换为整数值，可能是为了在将中间值存入 PLC 寄存器内之前降低其精度。例如，表达式 `int(cos(Theta)*100)` 会计算出一个角度的余弦，然后使用浮点将求出值乘以 100，将其转换为整数，省略小数点后的数字。

## 比较值

有时您可能会希望将一个数据的值与另一个进行比较，然后根据结果进行决策。例如，您可能会希望定义一个标志公式，来显示液位是否超过了特定值，或在程序里使用一个 `if` 语句在马达达到所需速度时执行一些代码。提供了下列比较运算符：

运算符	优先级	示例
等于	分组 7	<b>Data == 100</b>
不等于	分组 7	<b>Data != 100</b>
大于	分组 6	<b>Data &gt; 100</b>
大于等于	分组 6	<b>Data &gt;= 100</b>
小于	分组 6	<b>Data &lt; 100</b>
小于等于	分组 6	<b>Data &lt;= 100</b>

每个运算符均会根据其检查的条件生成一个 0 值或 1 值。运算符可用于整数、浮点值或字符串之上。如果要比较字符串，则比较不区分大小写，所以“abc”与“ABC”相同。

## 测试位

**Crimson** 只允许使用位选择运算符来测试数据值内的一个位的值，此运算符由一个句点表示。运算符左侧应为要测试位的值，右侧应为指示要测试的位数的表达式。右侧的值必须在 0 到 31 之间。运算符求出的结果根据相关位的值等于 0 或 1。

运算符	优先级	示例
位选择	分组 1	<b>Input.2</b>

上面显示的示例会测试指示的标记中的位 2（亦即值为 4 的位）。

如果希望测试等于零的位，那么可以使用逻辑非运算符：

运算符	优先级	示例
逻辑非	分组 2	<b>!Input.2</b>

如果指示的标记的位 2 等于 0，则此示例等于 1，反之亦然。

## 多个条件

如果希望定义一个若干条件均 为真时才为真的表达式，那么可以使用逻辑与运算符。同样，如果希望定义一个若干条件中任何 一个为真时就为真的表达式，则可使用逻辑或运算符。下面的示例显示了这两种运算符：

运算符	优先级	示例
逻辑与	分组 11	<b>A&gt;10 &amp;&amp; B&gt;10</b>
逻辑或	分组 12	<b>A&gt;10    B&gt;10</b>

当且仅当左右两侧的表达式均为真时，逻辑与运算符才会产生值 1。当任何一个表达式为真时，逻辑或运算符就会产生值 1。请注意，逻辑运算符与本章节描述的位运算符有所不同，只要知道答案是什么之后，逻辑运算符就会停止运算。这意味着，在上面的逻辑与示例中，只有 A 大于 10 时，运算符右侧才会执行，因为如若不然，则与运算符的结果必然已经等于零。虽然此属性在上面的示例中并不会带来什么不同，但是如果左侧或右侧的表达式会调用程序或对数据值进行更改，那么就必须要考虑这一点。

## 选择值

有些情况下，您可能会希望在两个值之间根据某一条件的值进行选择，这两个值可以是整数、浮点值或字符串。例如，您可能会希望将一个马达的速度设为根据一个标志标记而等于 500 rpm 或 2000 rpm。此操作可使用 ?: 运算符来进行，此运算符很独特，因为它会如下例所示取三个参数：

运算符	优先级	示例
选择	分组 13	<b>Fast ? 2000 : 500</b>

如果 Fast 为真，则此示例会求值为 2000，否则则求值为 500。此运算符可被认为与 Microsoft Excel 之类的应用程序里的 IF 函数相同。

## 操纵位

Crimson 还提供了一些执行操作的运算符，它们不会讲整数视为数字值，而是视为位序列。这些运算符叫做位运算符。

## 与、或和异或

这三个位运算符均会产生一个结果，结果里每个位都被定义为等于运算符左右两侧的值中的相应位，这些位使用一个特殊的真值表结合起来：

运算符	优先级	示例
位与	分组 8	<code>Data &amp; Mask</code>
位或	分组 9	<code>Data   Mask</code>
位异或	分组 10	<code>Data ^ Mask</code>

下表显示了关联的真值表：

A	B	A & B	A   B	A ^ B
0	0	0	0	0
0	1	0	1	1
1	0	0	1	1
1	1	1	1	0

## 移位运算符

Crimson 还提供了将整数向左或向右移动  $n$  个位的运算符：

运算符	优先级	示例
左移	分组 5	<code>Data &lt;&lt; 2</code>
右移	分组 5	<code>Data &gt;&gt; 2</code>

每个示例均会将 `Data` 向指定方向移动两个位。

## 位非

最后，Crimson 还提供了一个对值中的位取反的位非运算符：

运算符	优先级	示例
Bitwise NOT	Group 2	<code>~Mask</code>

此示例会产生一个值，其中每个位都等于其在 `Mask` 中的值的反值。

## 索引数组

在数组名称后面加上一对包含一个索引数组的方括号，即可选择数组标记内的元素。此表达式的范围必须在 0 到数组内元素数目减一之间。例如，如果创建一个 10 个元素的数组，则第一个元素将是 `Name[0]`，最后一个则是 `Name[9]`。

## 索引字符串

方括号还可用于选择字符串内的字符。例如，如果有一个叫做 `Text` 的标记包含了字符串“ABCD”，则表达式 `Text[0]` 会返回值 `65`，等于第一个字符的 Unicode 值。超出字符串结尾的索引值会返回值零。

## 相加字符串

与相加数字类似，加法运算符可用于连接多个字符串。因此，表达式 `"AB"+"CD"` 会求值为“ABCD”。还可使用加法运算符来向字符串添加整数，这种情况下，会在字符串里的数据之后添加一个等于整数的 Unicode 表达形式的字符。

## 调用程序

在会返回值的程序的名称后面添加一对括号，就可以在表达式内调用程序。例如，`Program1()*10` 会调用关联的程序，并将返回值乘以 10。要使这有意义，则 `Program1` 的返回类型显然必须应设置为整数或浮点。

## 使用函数

Crimson 提供了一些预定义的函数，这些函数可用于访问系统信息，或执行常见数学运算。函数参考内详细定义了这些函数。可使用与程序所用的类似的语法来调用函数，函数的任何参数均包含在括号内。例如，`cos(0)` 会调用余弦函数，其中参数为 `0`，会返回值 `+1.0`。

## 优先级总结

下表显示了本章节定义的全部运算符的优先级：

分组	运算符
分组 1	.
分组 2	! ~
分组 3	* / %
分组 4	+ -
分组 5	<< >>
分组 6	< > <= >=
分组 7	== !=
分组 8	&
分组 9	
分组 10	^
分组 11	&&
分组 12	
分组 13	?:

会首先应用数字较低的分组内的运算符。





# 编写操作

表达式定义了值，操作则定义了事件发生时要执行的操作。数据库内的大多数操作都与基元交换或键盘交互有关。因为 **Crimson** 提供了一种简单的方法来定义常见操作，所以您经常无需手动编写操作。但是，如果您希望使用触发器、编写程序或在用户定义模式下使用键或基元，则需要编写操作。

## 更换页面

要创建一个更改显示在面板显示内的页面的操作，请使用语法 `GotoPage(Name)`，其中 `Name` 是相关显示页面的名称。会移除当前页面，然后显示新页面。

## 更改数字值

**Crimson** 提供了若干种更改数据值的方法。

### 简单赋值

要创建一个向通信设备里的标记或寄存器分配新值的操作，请使用语法 `Data=Value`，其中 `Data` 是要更改的数据项，`Value` 是要分配的值。请注意，`Value` 不仅可以是常数值，还可以是任何正确类型的有效表达式。请参阅上文，详细了解如何编写表达式。例如，代码 `[N7:0]=Tank1+Tank2` 可用于将两个液位相加，然后将总量存入 PLC 寄存器。

### 复合赋值

要创建一个操作，用于将数据值设为等于通过上文定义的任何运算符将当前值与另一个值结合，那么请使用语法 `Dataop=Value`，其中 `Data` 是要更改的标记，`Value` 是要被运算符使用的值，`op` 是任何可用的运算符。例如，代码 `Tag+=10` 会将 `Tag` 的值增加 10，而 `Tag*=10` 会将当前值乘以 10。

### 递增和递减

要创建一个为数据值增加一的操作，请使用语法 `Data++`。要创建一个为数据值减少一的操作，请使用语法 `Data--`。请注意，`++` 或 `--` 运算符放在相关数据值前后均可。放在前面时，`++Data` 表示的表达式值等于增加之后的 `Data` 的值。放在后面时，表达式等于更改之前的值。

## 更改位值

要更改标记内的一个位，请使用 `Data.Bit=1` 或 `Data.Bit=0` 语法，按需设置或清除位，其中 `Data` 是相关标记，`Bit` 是基于零的位数。请再次注意，`=` 操作符右侧的值可以是表达式（如需），这样一来，`Data.1=(Level>10)` 就可用于根据液位是否超过预设值来设置或清除一个位。

## 运行程序

在程序名称后添加一对括号，就可以在操作内调用程序。例如，`Program1()` 会调用关联的程序。根据程序属性的定义，程序会在前台或后台运行。

## 使用函数

Crimson 提供了一些可用于执行各种操作的预定义函数。函数参考中详细定义了这些函数。可使用与程序所用的类似的语法来调用函数，函数的任何参数均包含在括号内。例如，`SetLanguage(1)` 会将终端的语言设为 1。

## 运算符优先级

全部赋值运算符均位于分组 14 之内。换言之，操作中它们会在其它所有运算符之后执行。在分组中它们是从右到左执行的，这就是说，代码 `Tag1=Tag2=Tag3=0` 可用于一次清除全部三个标记。