

# InTouch® HMI 报警与事件指南

**Invensys Systems, Inc.**

修订版 A

上次修订日期：2007 年 8 月 6 日



## 版权声明

© 2007 Invensys Systems, Inc. 版权所有。保留所有权利。

保留所有权利。未经 Invensys Systems, Inc. 事先书面明确同意，不得通过任何手段（电子、机械、影印、录制或其它方式）复制、传输本文档中的任何部分，或是将其存储到检索系统。使用本文档所含信息不需承担任何相关的版权或专利责任。虽然在编制本文档的过程中已采取一切预防措施，但错误或疏漏在所难免，出版商与作者对此概不承担任何责任。对由于使用本文档所含信息而导致的任何损害，亦不承担任何赔偿责任。

本文档中的内容如有变更，恕不另行通知，这些内容亦不代表 Invensys Systems, Inc. 一方的承诺。本文所述软件系在遵守许可协议或保密协议的前提下提供。本软件的使用或复制必须遵守这些协议中的各项条款。

Invensys Systems, Inc.  
26561 Rancho Parkway South  
Lake Forest, CA 92630 U.S.A.  
(949) 727-3200

<http://www.wonderware.com>

对产品文档如有任何意见或建议，请发送电子邮件到 [productdocs@wonderware.com](mailto:productdocs@wonderware.com)。

## 商标

本文所提及且已知为商标或服务标志的所有专用名词均已采用适当的首字母大写形式。Invensys Systems, Inc. 无法证实此类信息的准确性。在本文档中使用某个专用名词不应视为会影响任何商标或服务标志的有效性。

Alarm Logger、ActiveFactory、ArchestrA、Avantis、DBDump、DBLoad、DT Analyst、FactoryFocus、FactoryOffice、FactorySuite、FactorySuite A<sup>2</sup>、InBatch、InControl、IndustrialRAD、IndustrialSQL Server、InTouch、MaintenanceSuite、MuniSuite、QI Analyst、SCADAAlarm、SCADASuite、SuiteLink、SuiteVoyager、WindowMaker、WindowViewer、Wonderware 以及 Wonderware Logger 均为 Invensys plc 及其子公司与附属公司的商标。所有其它品牌可能是其相应所有者的商标。

# 目录

欢迎.....	15
文档惯例.....	15
技术支持.....	16
第 1 章 报警与事件综述 .....	17
关于 InTouch 报警.....	18
报警优先级 .....	18
报警子状态 .....	19
报警确认.....	19
报警组 .....	19
关于 InTouch 事件.....	21
InTouch 报警的类型 .....	22
离散报警.....	22
模拟报警.....	23
InTouch 分布式报警系统.....	25
报警供应器与接收器.....	26
分布式报警组列表 .....	28
摘要报警与历史报警.....	29
报警禁用、约束及抑制.....	29
终端服务报警支持 .....	30
分布式报警系统数据储存 .....	30

<b>第 2 章 配置报警 .....</b>	<b>31</b>
定义报警层次结构 .....	31
创建报警组 .....	32
修改报警组 .....	34
删除报警组 .....	34
给标记配置报警条件 .....	34
配置离散报警 .....	35
配置值报警 .....	36
配置偏差报警 .....	37
配置变化率报警 .....	38
禁用报警 .....	39
约束报警 .....	39
为单独的标记设置事件属性 .....	41
配置报警与事件的全局设置 .....	41
配置报警缓冲区大小 .....	42
启用事件 .....	42
保留报警启用功能 .....	44
创建报警组列表文件 .....	45
<b>第 3 章 报警查询 .....</b>	<b>47</b>
示例报警查询 .....	49
获取更多有关 InTouch 查询的信息 .....	50
<b>第 4 章 查看当前报警 .....</b>	<b>51</b>
配置 Alarm Viewer 控件 .....	52
配置网格的外观 .....	52
配置显示字体 .....	56
选择与配置显示列 .....	56
控制用户在运行时可以访问哪些功能 .....	59
配置要显示哪些报警 .....	61
使用查询收藏夹创建自定义的保存查询 .....	62
为不同类型的报警记录颜色 .....	63
配置显示的报警记录时间格式 .....	64
配置报警记录的排序顺序 .....	66
在运行时使用 Alarm Viewer 控件 .....	67
理解状态栏信息 .....	68
在运行时使用查询收藏夹 .....	68

使用 Alarm Viewer 控件 ActiveX 属性 .....	71
配置 ActiveX 控件的颜色 .....	77
使用 Alarm Viewer 控件 ActiveX 方法 .....	78
确认报警 .....	78
抑制报警 .....	82
检索报警的有关信息 .....	87
运行查询 .....	87
移动与冻结显示 .....	90
给报警记录排序 .....	91
显示其它信息 .....	92
选择特定的报警 .....	93
显示上下文菜单 .....	96
使用方法与属性时的错误处理 .....	96
使用 ActiveX 事件触发脚本 .....	96
检测到新报警时运行脚本 .....	97

## 第 5 章 实时确认报警 ..... 99

理解报警确认模型 .....	100
条件确认报警模型 .....	100
扩展的摘要报警模型 .....	100
基于事件的报警模型 .....	101
在运行时检查标记的确认模型 .....	102
使用点域确认报警 .....	103
确认报警或报警组 .....	103
确认值报警 .....	105
确认离散报警 .....	106
确认偏差报警 .....	107
确认变化率报警 .....	108
使用脚本函数确认报警 .....	109
Ack() 函数 .....	109
标记值返回到正常时使用自动确认 .....	110
使用报警客户端确认报警 .....	110
使用报警与确认注释 .....	112

## 第 6 章 在运行时控制标记与组的报警属性..... 113

确定标记或报警组是否处在报警条件中 .....	120
\$NewAlarm 系统标记 .....	121
\$System 系统标记 .....	121
.Alarm 点域 .....	122
.Normal 点域 .....	123
.AlarmDsc 点域 .....	124
.AlarmDev 点域 .....	125
.AlarmROC 点域 .....	126
.LoStatus 点域 .....	127
.LoLoStatus 点域 .....	128
.HiStatus 点域 .....	129
.HiHiStatus 点域 .....	130
.MinorDevStatus 点域 .....	131
.MajorDevStatus 点域 .....	132
.ROCStatus 点域 .....	133
确定是否给标记设置了报警限 .....	134
.LoLoSet 点域 .....	134
.LoSet 点域 .....	135
.HiSet 点域 .....	135
.HiHiSet 点域 .....	136
.MinorDevSet 点域 .....	137
.MajorDevSet 点域 .....	138
.ROCSet 点域 .....	139
启用与禁用标记或报警组的报警 .....	140
启用 / 禁用所有报警 .....	140
启用 / 禁用 LoLo 报警 .....	142
启用 / 禁用 Low 报警 .....	144
启用 / 禁用 High 报警 .....	146
启用 / 禁用 HiHi 报警 .....	148
启用 / 禁用离散报警 .....	150
启用 / 禁用副偏差报警 .....	152
启用 / 禁用主偏差报警 .....	154
启用 / 禁用变化率报警 .....	156
更改标记的报警限 .....	158
.LoLoLimit 点域 .....	158
.LoLimit 点域 .....	159
.HiLimit 点域 .....	160

.HiHiLimit 点域 .....	161
.MinorDevPct 点域.....	162
.MajorDevPct 点域.....	163
.DevTarget 点域.....	164
.ROCPct 点域 .....	165
更改标记的报警死区.....	166
.AlarmValDeadband 点域.....	166
.AlarmDevDeadband 点域.....	167
更改与标记关联的报警注释.....	168
.AlarmComment 点域 .....	168
将用户自定义信息关联到报警实例 .....	169
.AlarmUserDefNumX 点域.....	169
.AlarmUserDefStr 点域 .....	170
确定标记或报警组的约束标记 .....	172
.AlarmDscInhibitor 点域 .....	172
.AlarmLoLoInhibitor 点域.....	173
.AlarmLoInhibitor 点域 .....	174
.AlarmHiInhibitor 点域 .....	175
.AlarmHiHiInhibitor 点域 .....	176
.AlarmMinDevInhibitor 点域 .....	177
.AlarmMajDevInhibitor 点域 .....	178
.AlarmROCIInhibitor 点域 .....	179
统计活动的或未确认的报警数 .....	180
.AlarmTotalCount 点域 .....	181
.AlarmUnAckCount 点域.....	182
.AlarmValueCount 点域 .....	183
.AlarmValueUnAckCount 点域.....	184
.AlarmDscCount 点域.....	185
.AlarmDscUnAckCount 点域.....	186
.AlarmDevCount 点域.....	187
.AlarmDevUnAckCount 点域 .....	188
.AlarmROCCount 点域 .....	189
.AlarmROCUnAckCount 点域.....	190

## 第 7 章 查看报警层次结构..... 191

配置 Alarm Tree Viewer 控件.....	192
配置外观与颜色.....	192
配置字体.....	194

配置自动刷新 .....	195
控制用户可以在运行时访问的功能 .....	195
配置要显示的供应器与组 .....	196
使用查询收藏夹创建自定义的保存查询 .....	198
配置报警组的排序顺序 .....	199
在运行时使用 Alarm Tree Viewer 控件 .....	200
理解状态栏信息 .....	201
使用查询收藏夹 .....	201
使用 AlarmTreeView 控件 ActiveX 属性 .....	202
使用 Alarm Tree Viewer 控件 ActiveX 方法 .....	204
检索有关控件的信息 .....	204
检索有关特定项目的信息 .....	205
冻结目录树 .....	210
从所选元素中创建查询字符串 .....	210
运行查询 .....	211
使用方法与属性时的错误处理 .....	212
使用 Alarm Tree Viewer 控件 ActiveX 事件触发脚本 .....	212

## 第 8 章 打印报警 ..... 213

配置报警打印与记录 .....	213
配置打印机设置 .....	214
配置要打印的报警 .....	215
配置打印与文件输出的格式 .....	217
配置报警的日志文件 .....	220
保存与加载配置文件 .....	222
打印报警 .....	223
将报警记录到文件 .....	224
使用特定的配置启动 Alarm Printer .....	225
使用脚本控制 Alarm Printer .....	225
停止与启动 Alarm Printer 实例或查询 .....	226
查询报警查询信息 .....	229
查询实例信息 .....	236
查询打印机信息 .....	241
设置报警查询信息 .....	243
处理 Alarm Printer 错误 .....	248



## 第 9 章 将报警记录到报警数据库.....249

Alarm DB Logger Manager 的 SQL Server 帐户 .....	250
使用 Alarm DB Logger Manager .....	250
配置报警数据库记录功能 .....	251
配置数据库连接 .....	251
配置要记录的报警 .....	252
配置记录间隔 .....	254
将 Alarm DB Logger 配置成服务 .....	255
启动与停止报警数据库记录功能.....	256
报警数据库视图.....	256
报警历史视图 .....	257
事件历史视图 .....	259
报警事件历史视图 .....	260
AlarmSuite 报警日志视图 .....	263
报警数据库存储过程.....	264
调用存储过程 .....	264
AlarmCounter 数据库存储过程.....	265
EventCounter 数据库存储过程 .....	266

## 第 10 章 查看记录的报警.....267

配置 Alarm DB View 控件 .....	268
配置数据库连接 .....	269
配置网格外观 .....	270
配置显示字体 .....	271
选择报警或事件数据.....	272
选择与配置显示列 .....	273
控制用户可以在运行时访问哪些功能 .....	276
配置为报警记录显示的时间格式与时区 .....	277
选择报警数据库中数据的时间段.....	279
创建自定义过滤器与使用过滤器收藏夹.....	281
配置不同类型报警记录的颜色 .....	286
配置报警记录的排序顺序 .....	287
在运行时使用 Alarm DB View 控件.....	288
给记录排序 .....	288
理解状态栏信息.....	288
使用 Alarm DB View ActiveX 属性 .....	289
AckAlmBackColor 属性 .....	289
AckAlmBackColorRange1 属性.....	289

AckAlmBackColorRange2 属性.....	290
AckAlmBackColorRange3 属性.....	290
AckAlmBackColorRange4 属性.....	291
AckAlmForeColor 属性.....	291
AckAlmForeColorRange1 属性 .....	291
AckAlmForeColorRange2 属性 .....	292
AckAlmForeColorRange3 属性 .....	292
AckAlmForeColorRange4 属性 .....	293
AckRtnBackColor 属性.....	293
AckRtnForeColor 属性.....	294
AlmRtnBackColor 属性 .....	294
AlmRtnForeColor 属性 .....	294
AutoConnect 属性 .....	295
ColorPriorityRange1 属性 .....	295
ColorPriorityRange2 属性 .....	296
ColorPriorityRange3 属性 .....	296
ColumnResize 属性.....	296
连接状态属性 .....	297
CustomMessage 属性.....	297
DatabaseName 属性 .....	297
DisplayMode 属性.....	298
DisplayedTimeZone 属性.....	298
Duration 属性.....	299
EndTime 属性 .....	300
EventBackColor 属性 .....	300
EventForeColor 属性 .....	300
FilterFavoritesFile 属性 .....	301
FilterMenu 属性.....	301
FilterName 属性 .....	301
FromPriority 属性.....	302
GroupExactMatch 属性 .....	302
GroupName 属性 .....	303
MaxRecords 属性 .....	303
Password 属性.....	303
PrimarySort 属性.....	304
ProviderExactMatch 属性 .....	304
ProviderName 属性 .....	305
QueryTimeZoneName 属性.....	305
RefreshMenu 属性 .....	305

ResetMenu 属性 .....	306
RowCount 属性 .....	306
RowSelection 属性 .....	306
SecondarySort 属性 .....	307
ServerName 属性 .....	307
ShowFetch 属性 .....	307
ShowGrid 属性 .....	308
ShowHeading 属性 .....	308
ShowMessage 属性 .....	308
ShowStatusBar 属性 .....	309
SilentMode 属性 .....	309
SortMenu 属性 .....	309
SortOrder 属性 .....	310
SpecificTime 属性 .....	310
StartTime 属性 .....	311
Time 属性 .....	311
ToPriority 属性 .....	311
TotalRowCount 属性 .....	312
UnAckAlmBackColor 属性 .....	312
UnAckAlmBackColorRange1 属性 .....	313
UnAckAlmBackColorRange2 属性 .....	313
UnAckAlmBackColorRange3 属性 .....	314
UnAckAlmBackColorRange4 属性 .....	314
UnAckAlmForeColor 属性 .....	315
UnAckAlmForeColorRange1 属性 .....	315
UnAckAlmForeColorRange2 属性 .....	316
UnAckAlmForeColorRange3 属性 .....	316
UnAckAlmForeColorRange4 属性 .....	317
UnAckOrAlarmDuration 属性 .....	317
UserID 属性 .....	317
使用 Alarm DB View ActiveX 方法 .....	318
控制数据库连接 .....	318
从数据库检索记录 .....	319
检索报警的有关信息 .....	321
给报警记录排序 .....	322
显示上下文菜单 .....	323
访问过滤器收藏夹 .....	323
显示其它信息 .....	323
使用方法与属性时的错误处理 .....	324
使用 Alarm DB View ActiveX 事件触发脚本 .....	324

## 第 11 章 分析标记间的报警分布..... 325

配置 Alarm Pareto ActiveX 控件 .....	325
配置与报警数据库的连接 .....	326
配置 Alarm Pareto 控件的外观与颜色 .....	327
配置显示字体 .....	329
配置用户可以在运行时访问的功能 .....	330
配置要分析的报警 .....	331
使用过滤器收藏夹创建自定义过滤器 .....	333
配置分析结果的显示 .....	337
在运行时使用 Alarm Pareto 控件 .....	338
理解状态栏上显示的信息 .....	338
使用 Alarm Pareto ActiveX 属性 .....	339
使用 Alarm Pareto ActiveX 方法 .....	342
控制数据库连接 .....	342
从数据库检索记录 .....	342
检索特定巴累托条的有关信息 .....	343
显示其它信息 .....	345
使用方法与属性时的错误处理 .....	346
使用 Alarm Pareto ActiveX 事件触发脚本 .....	346

## 第 12 章 维护报警数据库..... 347

配置清除或归档设置 .....	348
配置数据库连接 .....	348
配置要从服务器中清除的数据量 .....	349
配置清除的数据的归档 .....	350
配置日志文件设置 .....	352
手工清除与归档数据库 .....	354
设置自动清除计划 .....	356
恢复报警数据库 .....	357
配置数据库连接 .....	358
配置要恢复的文件 .....	359
开始数据库恢复操作 .....	360

## 第 13 章 通过冗余报警配置增强工厂安全性..... 361

理解热备份 .....	362
配置热备份对 .....	363
创建热备份对 .....	364

设置热备份对的报警关键码域 .....	366
创建报警记录映射文件 .....	368
导入报警记录映射文件 .....	371
热备份对示例 .....	373
确认同步示例 .....	377
有关热备份对的备注 .....	379

## 第 14 章 创建报警审核跟踪 ..... 381

## 附录 A 使用分布式报警显示对象 ..... 383

关于分布式报警显示对象 .....	383
分布式报警显示对象准则 .....	384
在设计时配置分布式报警显示对象 .....	385
创建分布式报警显示对象 .....	385
配置网格的外观 .....	386
控制用户可以在运行时访问的功能 .....	387
配置要显示哪些报警 .....	388
配置缺省报警注释 .....	389
配置报警记录的时间格式 .....	390
配置报警记录的字体 .....	391
配置报警记录的列 .....	392
配置报警记录的颜色 .....	394
配置显示类型 .....	395
使用分布式显示对象监视本地报警 .....	396
在运行时使用分布式报警显示对象 .....	396
可调整大小的显示列 .....	396
多重选择 .....	397
报警消息颜色 .....	397
状态栏 .....	397
快捷菜单 .....	398
选择与配置报警查询收藏夹 .....	399
使用函数与点域控制分布式报警显示对象 .....	402
获取或设置属性 .....	402
确认报警 .....	402
选择报警 .....	411
检索所选报警记录的有关信息 .....	416
设置报警查询 .....	432
检查当前查询属性 .....	435

检查分布式报警显示对象的更新 .....	441
抑制报警 .....	443
滚动报警显示 .....	455
显示报警统计与计数 .....	458
错误描述 .....	464
 <b>附录 B 从旧有的报警系统迁移 .....</b>	<b>465</b>
从标准报警系统迁移到分布式报警系统.....	465
迁移 AlarmSuite 数据库 .....	466
 <b>索引.....</b>	<b>469</b>

# 欢迎

本文介绍如何管理由 InTouch 应用程序生成的报警与事件。

您可以查看本文，也可以使用 Adobe Reader 中的打印功能来打印本文的部分或全部内容。

本文假设您了解如何使用 Microsoft Windows，包括浏览菜单、在应用程序之间切换，以及在屏幕上移动对象。如需有关这些任务的帮助，请参阅 Microsoft 文档。

## 文档惯例

本文采用以下惯例：

惯例	用于
首字母大写	路径与文件名。
<b>粗体</b>	菜单、命令、对话框名称以及对话框选项。
等宽字体	代码范例与显示文本。

## 技术支持

Wonderware 的“技术支持”部门提供多种技术支持方案，帮助解答有关 Wonderware 产品及其实施方案的任何疑问。

在与“技术支持”部门联系之前，请参阅本文中相关的章节，以寻求问题的可能解决方案。如果需要联系技术支持以获取帮助，请准备好以下信息：

- 使用的操作系统的类型与版本。
- 有关如何重现问题的详细说明。
- 看到的错误消息的准确内容。
- Log Viewer 或任何其它诊断应用程序提供的任何相关输出列表。
- 为解决问题所作的尝试及其结果的详细说明。
- 如果遇到仍然存在的已知问题，请提供指定给该问题的“Wonderware 技术支持”案例号。



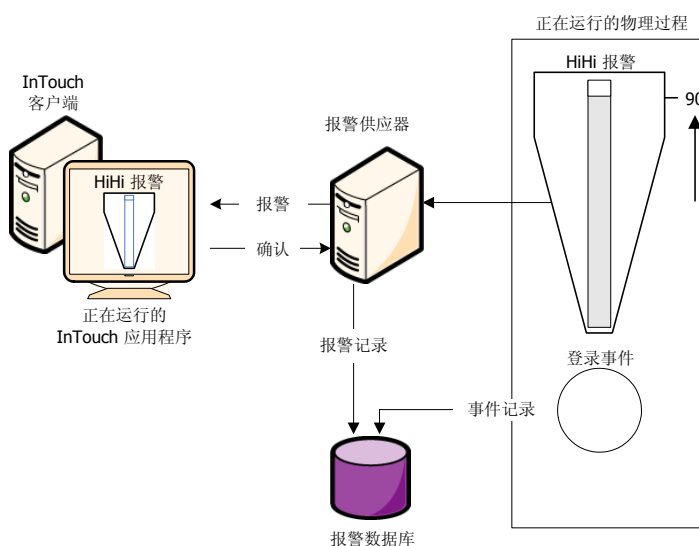
# 第 1 章

## 报警与事件综述

通过创建可生成报警与事件的 InTouch 应用程序，可以通知操作员有关生产过程活动的状态。

- 报警向运行时操作员警告可能导致潜在问题的过程条件。通常，您设置一个在过程值超过定义的极限时触发的报警。操作员通常必须确认报警。
- 事件代表正常的系统状态消息。通常事件在发生某种系统条件时触发，如操作员登录到 InTouch 应用程序。操作员不必确认事件。

下图显示 InTouch HMI 在应用程序运行期间如何处理报警与事件。报警与事件数据保存到报警数据库。



您可以配置任何标记进行事件监视。每次标记值改变时，都有一个事件消息记录到报警系统。事件消息包含：值是如何改变的，是操作员、I/O、脚本还是系统促使了这个改变。

## 关于 InTouch 报警

报警代表可能导致问题并要求操作员作出响应的过程条件警告。报警通常在过程值超过用户定义的极限时（如模拟值超过阈值上限）触发。这会触发一个报警，以通知操作员有问题发生。操作员确认报警之后，InTouch HMI 认为该报警已经得到确认。

您可以将 InTouch HMI 配置成即便引起报警的条件已消失但仍要求确认报警。这确保操作员可以了解到那些导致了临时报警状态但又已恢复正常的事件。

主要的报警状态在下表中介绍：

报警状态	条件
ACK	报警已确认。
ALM	报警已发生。
RTN	标记已从报警状态返回到正常状态。

## 报警优先级

您可以给报警指定一个优先级，或者说是严重度。例如，锅炉温度超出极限时，要求发出高优先级的报警，需引起立即关注。对于已到换班时间的报警，严重程度则可以低很多。报警优先级通常取决于环境 - 工厂应用、设备性质、安全性、备份系统的可用性、损害或停机的潜在成本等。

在定义标记时可以指定报警优先级。优先级范围可以从 1 到 999，其中 1 表示最严重。

您可以指定报警优先级范围来代表报警的分类。例如，如果某个过程要求使用四种严重程度级别，则可以创建四个优先级范围。

报警严重程度	优先级范围
关键	1 – 249
主要	250 – 499
次要	500 – 749
提示性	750 – 999

范围对于过滤报警非常有用。例如，您可以配置一个报警显示，从所有的报警中过滤出关键报警。您可以创建动画链接、确认脚本以及过滤的查看与打印，所有这些都可以基于报警优先级范围。

## 报警子状态

多状态报警包含一系列报警子条件。例如，模拟报警通常有多个极限。

- High 与 Low 阈值设置正常操作范围的边界。
- HiHi 与 LoLo 极限表示极度偏离正常值范围的偏差。

锅炉温度水平可以由于任何这些子状态之一而处在报警条件下。在继续处于整体报警条件时，锅炉温度还可以在任何两个子状态之间转换。

## 报警确认

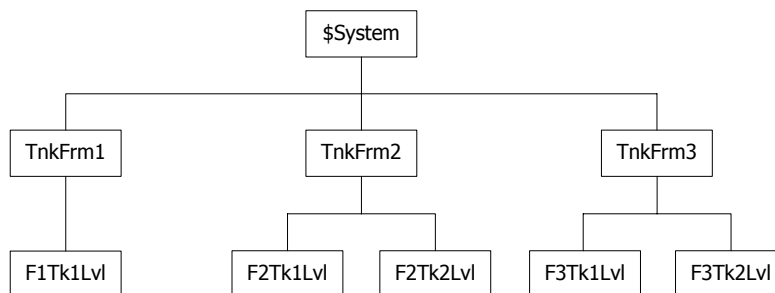
报警发生时，运行时操作员（或系统）必须确认报警。确认只是表示有人注意到该报警。这与采取修正操作没有关系，后者可能不会立即发生。它同报警条件是否返回到正常也没有什么关系 – 有时即便没有任何外界干预，它也可能自行恢复正常。

较高或中等优先级的报警通常要求立即确认，而优先级非常低的报警则可能不作要求。尽管生成报警的条件可能已消失（例如，温度上升得过高，然后又降了下来），但在确认之前，报警本身并不会被认为已得到解决。

## 报警组

您可以将报警分组，以便更轻松地进行跟踪与管理。报警组是工厂的不同区域、设备的各个部件、操作员的责任或生产过程的逻辑表示。

例如，下图显示贮油站应用的三层报警组层次结构。



对于过滤报警显示、Alarm Printer 以及确认脚本，“报警组”都非常有用。

每个标记都与一个报警组关联。缺省条件下，标记指定给 \$System 主组。您可以在 \$System 组下创建其它报警组层次结构，最多可达 32 级。

在“标记字典”中定义标记时，可以创建报警组，并将标记同它们关联起来。

报警组与组变量同 SmartSymbol 不兼容。您无法在 SmartSymbol 中使用对报警组或组变量的引用。

# 关于 InTouch 事件

事件是系统中发生的某些事情的可检测到的实例，它不一定要与报警关联。进入或脱离某种报警状态的转换过程便是事件的一种。事件也可以是操作员的动作、系统配置的改变或某种系统错误。

事件和条件不同。条件可以持续几分钟、几小时、几天，甚至几周。事件则是瞬时的；它在发生之后便立即结束。报警是一种条件；而报警通知则是一个事件。

事件代表正常的系统状态消息，不要求操作员作出响应。

定义标记进行事件监视时，可以选择在每次标记值改变时，都打印事件消息或将其记录到报警系统。事件消息包含：值是如何改变的，是操作员、I/O、脚本还是系统促使了这个改变。

事件可以是以下类型之一：

事件	条件
OPR	操作员使用“值输入”修改了标记值。
LGC	QuickScript 修改了标记值。
DDE	从 DDE 客户端插入了标记值。
SYS	发生了系统事件。
USER	\$Operator 发生了改变。

不论是否启用了任何标记的事件记录功能，系统都将生成 SYS 与 USER 事件。DDE、OPR 和 LGC 事件与标记值相关，只有那些已启用事件记录的标记才会生成这些事件。

## InTouch 报警的类型

在 InTouch HMI 中，报警根据其特性分为一些常见的类别。这些类别也就是所谓的“类”与“类型”。“分布式报警”系统将所有报警归类到五种基本“条件”下：“离散”、“值”、“偏差”、“变化率”以及 SPC。

报警条件	分布的类	分布的类型
离散	DSC	DSC
值 - LoLo	VALUE	LOLO
值 - Low	VALUE	LO
值 - High	VALUE	HI
值 - HiHi	VALUE	HIHI
偏差 - 主	DEV	MAJDEV
偏差 - 副	DEV	MINDEV
变化率	ROC	ROC
SPC	SPC	SPC

在定义标记时，您可以将每个 InTouch 标记关联到某个报警条件。根据标记的类型，可以给它定义一个或多个报警类或类型。

### 离散报警

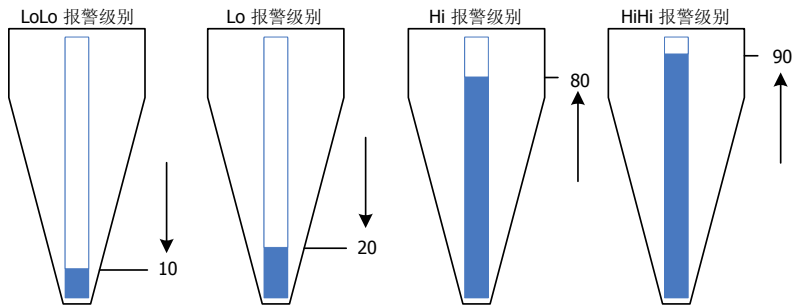
离散报警对应于有两种可能状态的离散标记。创建离散标记时，您可以将报警状态配置成对应于离散标记的“真”还是“假”状态。

## 模拟报警

模拟报警对应于同整数或实数关联的模拟标记。在模拟报警类型中，存在多个子类型：值、偏差及变化率。

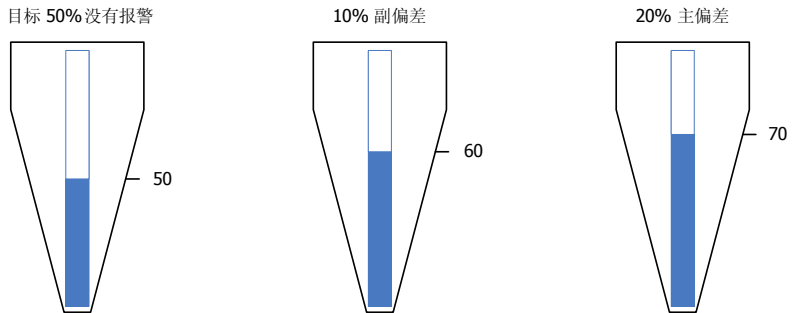
### 值报警

当前标记值与一个或多个预先确定的极限进行比较。如果值超过极限，则声明处于报警状态。您可以分别配置 "LoLo"、"Lo"、"Hi" 及 "HiHi" 等极限的值与优先级，并指明是否要使用每个极限。



### 偏差报警

当前标记值与目标值进行比较，然后将差异的绝对值与一个或多个极限进行比较，这些极限使用标记值范围的百分比来表示。



您可以分别配置副偏差极限与主偏差极限的值与优先级，并指明是否要使用每个极限。您也可以配置偏差死区的值，它也是使用标记范围的百分比来表示的。这可以控制标记值必须改变多少个百分点（占总计范围的）才能被认为处于报警状态。

例如，您可以按以下方式配置极限：

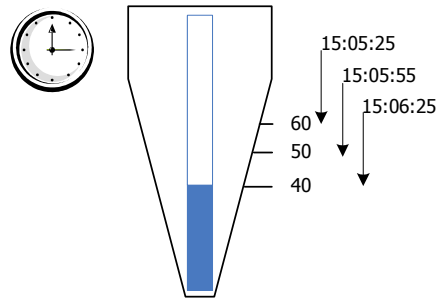
- 范围从 0 到 100
- 目标 50
- 副偏差 10%，即将副偏差阈值设置为 40 与 60。
- 主偏差 20%，即将主偏差阈值设置为 30 与 70。
- 死区 10%

如果标记值是 39，则发生副偏差报警。不过在评估并清除报警之前，标记值至少必须变为 50（40 加上死区值 10）。

如果标记值是 72，则发生主偏差报警。在该报警清除之前，标记值至少应该降到 61。

### 变化率报警

在一个测量周期中，比较标记的当前值与前面的值。如果变化率的绝对值超过极限，则声明处于报警状态。



您可以配置“变化率”极限的值与优先级，以及是否要使用该极限。该极限使用一段时间间隔 - 可以是每秒、每分钟或每小时 - 中标记值范围的百分比来表示。

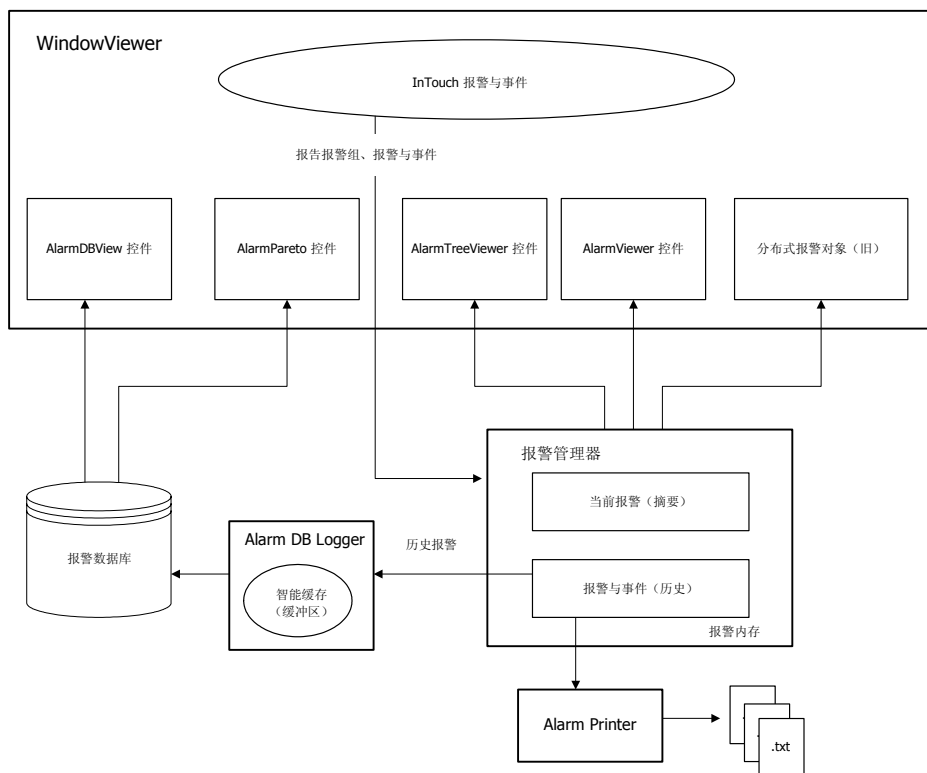


## InTouch 分布式报警系统

“分布式报警”系统由以下部分组成：

- “报警管理器”，管理当前活动报警（摘要报警）以及历史报警与事件。摘要与历史报警保存在 InTouch 内部报警内存中。
- Alarm DB Logger，将历史报警与事件存储到报警数据库。报警数据库是一个 SQL Server 数据库。
- Alarm Printer，打印历史报警与事件。
- 一组 ActiveX 控件，在运行时从内部报警内存或报警数据库中检索报警与事件。

下图显示系统概况。



运行时操作员可以使用“分布式报警”系统：

- 显示、记录及打印由本地 InTouch 应用程序以及其它网络应用程序的报警系统生成的报警与事件。
- 在本地或从远程网络节点上确认报警。
- 在 InTouch 应用程序中使用报警 ActiveX 控件显示预先配置的报警显示。
- 使用单独的报警注释字段提供关于报警的更多反馈。

作为应用程序开发人员，您可以：

- 通过点域来控制报警。
- 配置报警，以便在应用程序的完全控制下，直接或间接启用或禁用它们。您可以将报警抑制应用于单个报警类、标记或报警组，以禁止在特定的视图节点上显示报警信息。系统范围的禁用可以从源头阻止报警活动。
- 配置要记录到历史中的报警信息。Alarm DB Logger 可以作为 Windows 服务运行，也可以根据需要手工启动。报警记录功能使用 UTC (GMT) 时间标签，并提供夏令时与跨时区兼容性。
- 设置故障转移报警供应器。如果主报警供应器出现故障，“分布式报警”系统可以顺利地备份系统获取报警信息。重新连接主节点之后，在恢复使用主系统之前，“分布式报警”系统会确保已经重新同步报警确认。

“分布式报警”系统：

- 通过 Wonderware SuiteLink 协议发送数据，使用最少的 CPU 与网络资源。
- 在报警发生时，而不是在接收器接收报警时，给报警加上时间标签。时间标签包含毫秒。

## 报警供应器与接收器

在任何给定的节点上，都会有一组“报警供应器”（发布者）与“报警接收器”（预订者）。InTouch “分布式报警系统”提供通讯链接，在节点与软件组件之间传递报警信息。

### 报警供应器

报警供应器：

- 跟踪报警项 – 即可进入报警状态的项目 – 并向“分布式报警系统”提供这些项目的列表，包括有关任何项目分组层次结构的信息。

- 在报警项的状态改变时，通知“分布式报警系统”。状态改变包括项目是进入还是脱离报警状态，以及是否已确认最新的报警。
- 跟踪报警项是否已禁用。

InTouch HMI 支持外部报警供应器，如 SPCPro、QI Analyst、Archestra Galaxy，以及使用 Wonderware Alarm API 工具包构建的其它软件。这些报警记录的日期 / 时间标签由报警感应器提供，而不是由“分布式报警”系统生成。

## 报警接收器

报警接收器：

- 向“分布式报警”系统提供一组查询，确定它希望收到其通知的报警项。在报警接收器更改或删除查询之前，它保持活动状态，并指定报警供应器或报警组 - 这与使用“通配符”的 SQL 查询非常相似。只要报警供应器发出变化通知，“分布式报警”系统便检查与注册的任何查询匹配的报警，然后将更新传递给相应的报警接收器。
- 在收到更新时，显示或记录同项目的状态或其转换有关的信息。
- 确认报警。报警接收器发送确认通知给“分布式报警系统”，确定报警与报警供应器。该通知传递给报警供应器，后者随之将项目状态更新为已确认（如果适用），并进而通知“分布式报警系统”，从而确保将更新发布给所有感兴趣的报警接收器。

---

**备注** “分布式报警系统”中的大部分通讯活动是将报警查询与报警记录从一个节点发送到另一个节点。在节点内，报警查询与报警记录由内部报警内存进行跟踪并缓存，以最大限度减少网络流量。

---

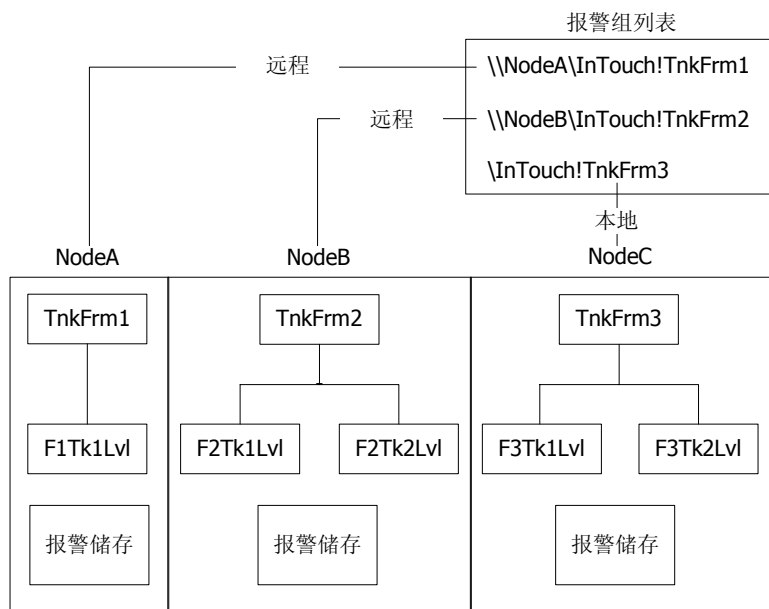
## 分布式报警组列表

“分布式报警系统”使用报警组将报警整理到本地目录树视图中。分布式报警显示使用目录树视图过滤报警。您可以从网络上的多个节点来查看这些报警组。

“分布式报警系统”使用一个报警组列表将本地与远程节点上的报警组合并到一起。报警组列表是一个有名称的列表，它由 InTouch 节点以及那些节点中的每一个上定义的报警组所组成。它也可以包含其它报警组列表名与本地报警组。报警接收器（如 Alarm Viewer 控件）使用此列表查询报警。

在查询别名下，Alarm Viewer 控件可以显示合并到一起的报警，这些报警来自属于该列表的各个组。这些报警可以在本地 InTouch 节点或是从网络中的远程节点上进行确认。

下图显示一个报警组列表，它将三个节点上的报警组合并到一起。报警组列表在 NodeC 本地进行定义。剩余的报警组都来自远程节点。



“分布式报警显示”显示对列表中所有的报警组进行查询所产生的报警。例如，如果有兴趣显示多个 InTouch 节点上所有的贮油站报警，则可以创建一个名称为 TankFarmAlarms 的列表。对于此列表，您可以将运行贮油站 InTouch 应用程序的所有节点上的报警组都添加到其中。

如需有关创建报警组的详细信息，请参阅第 32 页的“创建报警组”。

## 摘要报警与历史报警

摘要报警是当前活动的报警。历史报警是当前不活动并且通常存储在报警数据库中的报警。

例如，您可能希望查看在等待确认的所有当前报警的摘要，而所有其它报警信息则属于过去的历史，并非十分紧要。

## 报警禁用、约束及抑制

您可以“关闭”报警或者忽略它们，但并不实际删除报警配置。您可以禁用、约束或抑制某个报警。

报警禁用与约束在报警供应器上进行控制。抑制则在报警接收器上进行控制。如需有关供应器与接收器的详细信息，请参阅第 26 页的“报警供应器与接收器”。

- **禁用。**通过设置标帜将报警标记为已禁用，可以在报警供应器上禁用报警。无论发生什么报警条件，该项都绝不会进入报警状态。如需有关可用于禁用报警的点域的详细信息，请参阅第 140 页的“启用与禁用标记或报警组的报警”。

您可以一次禁用或启用某个标记的所有报警。此外，对于包含子状态的报警，也可以分别禁用每个子状态。

- **约束。**您可以按照以下方式约束报警：
  - a 在 WindowMaker 中将“约束”标记添加到报警配置。约束标记用于在运行时将报警标记为受约束。
  - b 在运行时将约束标记设置为“真”或“假”。约束标记为“假”时，报警按正常方式进行处理。约束标记为“真”时，该项目无法报警。

每个报警子状态都可以使用不同的标记进行约束，您可以不给一些子状态指定约束标记。

将标记指定为某个报警的约束标记时，会增加其交叉引用使用计数。

- **抑制。**抑制会导致报警接收器忽略特定的报警。如果某个报警与排除标准匹配，则它是不可见的。这就是说，它不出现在该特定报警接收器的显示中，也不能在其中进行打印或记录。

实际的报警生成完全不受抑制的影响。报警记录仍然可以记录到报警历史中。

如果某个报警变为禁用或有效约束状态，而项目仍处于报警状态，则该项目会被强制转换到一个不同（有效）的状态。具体的状态应取决于当前何种状态可用以及它们是否也被禁用。此活动由报警供应器根据报警的类型、极限值等进行处理。

受禁用或有效约束的报警不会等待确认。如果报警有子状态，它仅可以在仍旧可用的子状态上等待确认。

## 终端服务报警支持

通过在 **Terminal Services for InTouch** 中使用“分布式报警系统”，运行在不同终端会话上的报警客户端可以选择要显示哪些报警数据以及如何显示。

“报警供应器”通过唯一确定应用程序及应用程序实例的名称来识别它们。“报警供应器”或“报警接收器”向“分布式报警系统”注册之后，此信息将可供“分布式报警系统”使用。

运行“报警供应器”的节点则通过在系统中唯一确定该计算机节点的名称进行识别。在计算机节点上启动“分布式报警系统”的实例之后，此信息便可供它使用。

在记录报警事件时，节点与完整的“报警供应器”名可以确定报警的来源。

在“终端服务”环境中确认报警时，记录的“操作员节点”将是客户端机器的名称，相应的操作员从这些客户端机器中建立“终端服务”会话。如果节点名无法检索，将使用节点的 IP 地址代替。

终端服务器客户端会话不能是 **InTouch** 报警供应器。

## 分布式报警系统数据储存

“分布式报警系统”使用多种数据存储形式：

- 内部报警内存（缓冲区）

有关当前与最新报警的大多数信息都保存在各计算机节点的内存中。**InTouch** 使用两个内存位置：一个用于摘要（当前）报警，另一个用于历史报警与事件。此模型也用在“分布式报警系统”中。

摘要报警的内存可根据需要进行扩展，以容纳所有的当前报警，直至达到可用内存的极限。历史报警的内存只能增长到预设的极限。在历史内存达到此极限之后，随着新的报警记录添加进来，最旧的报警记录会被丢弃。在多节点环境中，各节点上的报警内存构成一个报警内存集合。

如需有关设置极限的详细信息，请参阅第 42 页的“配置报警缓冲区大小”。

- 报警数据库

**Alarm DB Logger** 创建一个数据库，以跟踪报警发生、子状态转换、确认及恢复正常的时间。实际上，这些记录会形成系统中的报警历史。

“分布式报警系统”以查询为基础，因而支持使用一个计算机节点来记录多个其它节点的报警。

## 第 2 章

# 配置报警

要配置报警，只要给标记配置报警条件即可。

如果需要，您也可以：

- 定义报警层次结构。
- 禁用与约束报警。
- 配置报警注释。
- 配置其它报警与事件属性。

## 定义报警层次结构

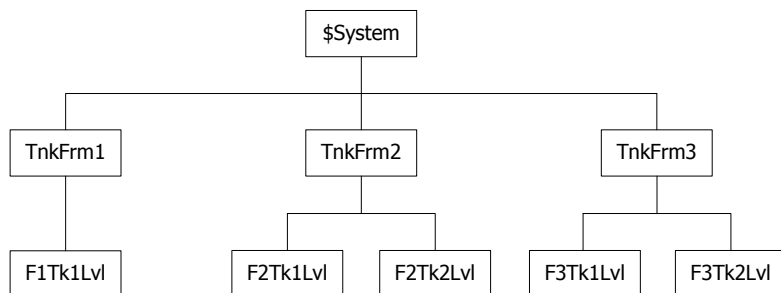
每个 InTouch 报警都属于某个报警组。对相关报警进行分组可以便于操作员过滤、显示及确认报警。如需有关详细信息，请参阅第 19 页的“报警组”。

“分布式报警系统”将报警组用作报警组列表的基础。如需有关创建“分布式报警系统”组列表的详细信息，请参阅第 45 页的“创建报警组列表文件”。

## 创建报警组

在开始创建报警组之前，首先计划报警组的组织方式与所需的报警组名。通过采用统一的组命名惯例，可以给层次结构中的组实现一种逻辑上的顺序。

在下图中，对于指定给层次结构中相同级别的组的名称，注意它们的相似之处。



同样请注意，下属组名通过包含父名的一部分来引用它们的父组。例如，第三级报警组名 **F1Tk1Lvl** 通过在组名中包含 **F1** 前缀来引用它的父报警组 **TnkFrm1**。制订一种命名惯例，以显示层次结构中不同级别的报警组之间的父子关系。

**备注** 虽然在 InTouch 许可机制中，报警组不计为标记，但它们在数据库中却确实计为标记。因此，报警组加上实际标记的总数不能超出 InTouch 许可证设置的最大限制。

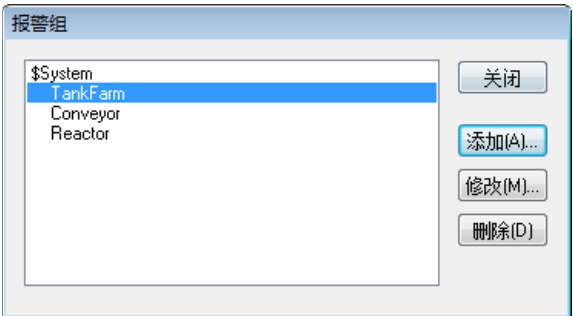
报警组名必须满足以下要求：

- 名称必须为 32 个字符或更少。
- 名称必须以字母数字字符（A-Z、a-z 或 0-9）开头。
- 名称从第二个字符位置开始，可以包含以下键盘字符 (@, #, \$, %, &, -, \_, ?, !, \)。
- 如果名称包含连字符 (-)，则必须以英文字符开头。
- 名称中不得包含空格。
- 名称至少必须包含一个英文字符。

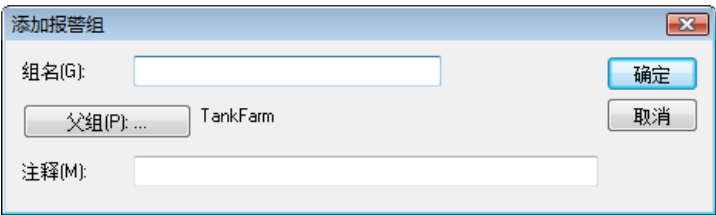


要创建报警组

- 1 在**特别**菜单上，单击**报警组**。此时出现**报警组**对话框。



- 2 单击**添加**。此时出现**添加报警组**对话框。



- 3 在**组名**框中，输入新报警组的名称。
- 4 要将报警组重新指定给另一个父组：
- a 单击**父组**以显示**报警组**对话框。如果这是为 InTouch 应用程序定义的第一个报警组，则该组自动指定给 \$System 父组。
  - b 从列表中选择一个新的父组，然后单击**关闭**。
- 5 在**注释**框中，为新报警组输入可选注释（最大长度是 49 个字符），然后单击**确定**。  
此时出现**报警组**对话框，显示已添加到列表中的新报警组。
- 6 单击**关闭**。

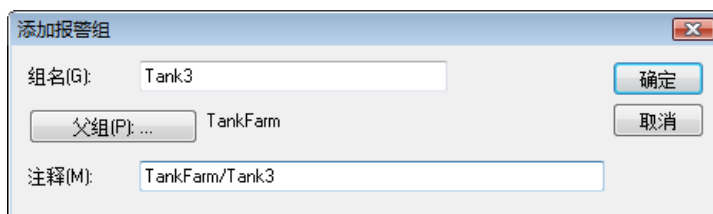
## 修改报警组

您可以修改报警组以：

- 给它重命名。
- 更改关联的注释。
- 将它重新指定给另一个组。

### 要修改报警组

- 1 在**特别**菜单上，单击**报警组**。此时出现**报警组**对话框。
- 2 选择要修改的报警组，然后单击**修改**。此时出现**修改报警组**对话框。



- 3 对报警组的名称或注释进行任何更改。
- 4 要将报警组重新指定给另一个父组：
  - a 单击**父组**以显示**报警组**对话框。
  - b 从列表中选择一个新的父组，然后单击**关闭**。
- 5 单击**确定**。

## 删除报警组

您可以删除某个报警组，并从层次结构中删除该组。属于所删除的报警组的报警与标记自动重新指定给层次结构中所删除的组的直接父组。同样，所删除的组的子组也将重新指定给所删除的组的直接父组。

### 要删除报警组

- 1 在**特别**菜单上，单击**报警组**。此时出现**报警组**对话框。
- 2 选择报警组，然后单击**删除**。消息出现时，单击**是**。
- 3 单击**关闭**。

## 给标记配置报警条件

通过指定报警类型与一个或多个报警阈值，您可以给任何标记配置报警。只要标记的值达到定义的阈值，便发生报警。标记值进入或脱离某种报警状态的任何转换都报告给“分布式报警系统”。

## 配置离散报警

离散报警对应于离散标记。您可以配置报警状态是对应于离散标记的“真”（打开、是、1）状态还是“假”（关闭、否、0）状态。

### 要为离散标记定义报警条件

- 1 打开“标记名字典”。
- 2 打开现有离散标记，或创建一个新的离散标记。
- 3 在**标记名字典**对话框的顶部，单击**报警**或**详细和报警**，以显示离散报警详细资料对话框。

The screenshot shows a configuration dialog for a discrete alarm. It has several sections:
 

- 确认模型 (Confirmation Model):** Three radio buttons: **条件(C)** (selected), **面向事件(E)**, and **扩展的摘要(E)**.
- 报警注释(O) (Alarm Annotation):** A text input field.
- 报警状态(A) (Alarm State):** Three radio buttons: **打开** (Open), **关闭** (Closed), and **无** (None).
- 优先级 (Priority):** A numeric input field with the value '1'.
- 报警约束标记 (Alarm Constraint Mark):** A dropdown menu.

- 4 在**确认模型**区域中，为标记选择报警确认模型。
  - 单击**条件**，确认会统计至确认时为止所有进入报警状态或子状态的转换次数。这是缺省确认模型。
  - 单击**面向事件**，确认仅针对进入报警状态或子状态的特定转换；只有指最近一次事务时才会接受确认。
  - 单击**扩展的摘要**，确认只针对特定的转换，无论是转入报警状态、子状态，还是返回到正常状态。从正常状态的每次转换都标志着一个新的“返回正常”(RTN)组的开始。RTN组中的所有转换都必须单独确认，之后整个RTN组才会视为已确认。
- 5 在**报警注释**框中，输入报警注释，最大长度为 131 个字符。
- 6 在**报警状态**区域中，选择活动报警状态是离散标记的**打开**还是**关闭**值。
- 7 在**优先级**框中，指定 1 到 999 之间的一个报警优先级。缺省优先级数字是 1，这是最高的报警优先级。
- 8 作为可选项，还可以为离散报警指定一个报警约束标记。
  - a 在**报警约束标记**框中，单击按钮以显示**选择标记**对话框，其中包含一系列已定义的标记。
  - b 从列表中选择一个标记，然后单击**确定**。选作约束标记的标记的名称出现在**报警约束标记**框中。

如需有关约束报警的详细信息，请参阅第 39 页的“约束报警”。
- 9 单击**保存**。
- 10 单击**关闭**以关闭**标记名字典**对话框。

## 配置值报警

值报警与整型或实型标记关联。您可以将报警设置成在标记值转换为超出一组预定义的阈值（从 LoLo 到 HiHi）时触发。您可以配置报警状态是否对应于该标记的任何值，以及与该报警关联的优先级。

### 要配置值报警

- 1 打开“标记名字典”。
- 2 选择现有的实型或整型标记，或创建一个新的标记。
- 3 在**标记名字典**对话框的顶部，单击**报警或详细和报警**，以显示报警详细资料对话框。

确认模型: <input checked="" type="radio"/> 条件(C) <input type="radio"/> 面向事件(T) <input type="radio"/> 扩展的摘要(X)			报警注释(A): 反映炉液面					
	报警值	优先级	报警约束标记		报警值	优先级	报警约束标记	值死区(V)
<input type="checkbox"/> LoLo(L)	0	1	...	<input checked="" type="checkbox"/> High(H)	1800	1	...	0
<input checked="" type="checkbox"/> Low(W)	200	1	...	<input type="checkbox"/> HiHi(I)	180	1	...	

- 4 在**确认模型**区域中，为标记选择报警确认模型。
  - 单击**条件**，确认会统计至确认时为止所有进入报警状态或子状态的转换次数。这是缺省的确认模型。
  - 单击**面向事件**，确认仅针对进入报警状态或子状态的特定转换。只有指最近一次事务时才会接受确认。
  - 单击**扩展的摘要**，确认只针对特定的转换，无论是转入报警状态、子状态，还是返回到正常状态。从正常状态的每次转换都标志着一个新的“返回正常”(RTN)组的开始。RTN组中的所有转换都必须单独确认，之后整个RTN组才会视为已确认。
- 5 在**报警注释**框中，输入缺省注释，最大长度为 131 个字符。注释指定给标记的 .AlarmComment 点域。
- 6 选择报警类型（LoLo、Low、High、HiHi），以检测何时标记值超出绝对极限值。
- 7 在**报警值**框中，为报警类型输入极限值。

例如，对于 LoLo 与 Low 报警，只要标记值小于**报警值**，便存在报警条件。对于 High 与 HiHi 报警，只要标记值超出**报警值**，便发生报警。您可以给这些极限值使用实数。
- 8 在**值死区**框中，输入工程单位数；标记值必须上升到报警值以上或下降到报警值以下这个数值，才能脱离报警状态。

例如，要从报警条件下返回到正常状态，标记值不仅要返回到报警限之内，还需要返回到指定的“值死区”范围内。“值死区”可以防止由于报警的不断反复（即标记值在极限附近上下浮动，从而持续进出报警状态）造成过度报警。
- 9 作为可选项，您可以为标记的报警类型（LoLo、Low、High、HiHi）指定报警约束标记。

- a 在**报警约束标记**区域中，单击按钮以显示**选择标记**对话框，其中包含一系列已定义的标记。
- b 从列表中选择一个标记，然后单击**确定**。选作约束标记的标记的名称出现在**报警约束标记**框中。

如需有关约束标记的详细信息，请参阅第 39 页的“约束报警”。

10 单击**保存**。

11 单击**关闭**以退出**标记名字典**对话框。

## 配置偏差报警

偏差报警与整型或实型标记关联。您可以通过以下方法来触发报警：将当前标记值与目标值进行比较，然后将差的绝对值与一个或多个极限进行比较，这些极限表示为标记值范围的百分比。

例如，以下这些值设置标记主、副偏差报警的条件：

最小值 = -1000

最大值 = 1000

副偏差百分比 = 10

主偏差百分比 = 15

目标值 = 500

将这些值用作示例，主、副偏差报警点按以下步骤进行计算：

- 1 计算标记的整个值范围。  
 $1000 - (-1000) = 2000$
- 2 将标记的整个值范围乘以主、副偏差百分比。  
 $2000 \times 0.10 = 200 = \text{副偏差限}$   
 $2000 \times 0.15 = 300 = \text{主偏差限}$
- 3 在目标值上加上与减去主、副偏差限。  
 $500 - 200 = 300 = \text{副偏差下限}$   
 $500 + 200 = 700 = \text{副偏差上限}$   
 $500 - 300 = 200 = \text{主偏差下限}$   
 $500 + 300 = 800 = \text{主偏差上限}$

### 要配置偏差报警

- 1 打开“标记名字典”。
- 2 选择现有的实型或整型标记，或创建一个新的标记。
- 3 在**标记名字典**对话框的顶部，单击**报警或详细和报警**，以显示报警详细资料对话框。

- 4 选择要使用的偏差（**主偏差**与**副偏差**）报警类型，它用于检测何时模拟型标记的值在指定目标值的主偏差或副偏差范围内。
- 5 在**% 偏差**框中，输入触发主偏差或副偏差报警条件时模拟标记要偏离目标值的百分比。它表示为标记范围的百分比。对于 I/O 标记，范围由标记的详细资料对话框中输入的**最小工程单位**与**最大工程单位**值定义。对于内存标记，范围由最小值与最大值定义。
- 6 在**目标**框中，输入标记参考值，主、副偏差百分比都基于这个参考值。
- 7 在**偏差死区百分比**框中，输入偏差百分比；标记值必须下降到极限值以下这个百分比，标记才能脱离报警条件。
- 8 单击**保存**。
- 9 单击**关闭**以关闭**标记名字典**对话框。

## 配置变化率报警

在指定时间间隔内报警值的变化超过指定范围时，变化率报警能检测到这个情况。只要标记值发生变化，就会对标记进行测试以确定是否发出变化率报警。变化率使用前一个值、上次更新时间、当前值以及当前时间来计算。

计算的值与为该标记指定的变化率百分比允许范围进行比较。如果变化率大于允许的百分比范围，则为标记设置报警条件。变化率报警持续有效，直至标记变化的速率降至报警限以下。

### 要配置变化率报警

- 1 打开“标记名字典”。
- 2 选择现有的实型或整型标记，或创建一个新的标记。
- 3 在**标记名字典**对话框顶部，单击**报警或详细和报警**，以显示报警详细资料对话框。下图仅显示适用于变化率报警的那些选项。

- 4 选择**变化率**框。

- 5 在 % 框中，输入允许的最大变化百分比极限。
- 6 选择秒、分或时作为时间间隔单位。
- 7 在优先级框中，输入 1 与 999 之间的一个数字，以设置报警优先级。
- 8 作为可选项，您可以为变化率报警指定一个报警约束标记。
  - a 在报警约束标记区域中，单击按钮以显示选择标记对话框，其中包含一系列已定义的标记。
  - b 从列表中选择标记，然后单击确定。选作约束标记的标记的名称出现在报警约束标记框中。

如需有关约束报警的详细信息，请参阅第 39 页的“约束报警”。
- 9 单击保存。
- 10 单击关闭以关闭标记名字典对话框。

## 禁用报警

您可以使用 `.AlarmEnabled` 或 `AlarmDisabled` 点域一次性禁用或启用标记的所有报警。对于具有子状态的报警，每个子状态都可以单独禁用。例如，模拟值报警可以启用 Hi 而禁用 HiHi。

在运行时，“报警供应器”不为禁用的报警或子状态生成报警。在运行时可以更改报警的禁用或启用状态。

只要报警从禁用转换为启用，检查逻辑便会确定“报警供应器”是否应该将该项放入报警状态。

如果在项目处于报警状态时，报警变为禁用或有效约束状态，该项目将强制转换到一个不同的（有效）状态。具体的状态应取决于当前有哪些可用的状态，以及它们是否也被禁用。此活动由“报警供应器”根据报警类型与极限值进行处理。

## 约束报警

作为可选项，您可以为每个报警或其子状态指定一个报警约束标记，以防止报警转换为活动状态。

- 约束标记值变为且保持为“真”（非零或非空）时，报警便会被约束。
- 类似地，报警约束标记变为且保持为“假”（零或空）时，报警不会被约束。

您只能在 WindowMaker 中更改约束标记。在运行时，可以更改约束标记的值。

您可以为单独的报警子状态指定约束标记。每个子状态可以由不同的标记来约束，某些子状态也可以不指定约束标记。

受约束的报警（对应的标记值为“真”）不等待确认。如果报警有子状态，它仅可以在仍旧可用的子状态上等待确认。

报警或子状态可以单独禁用、约束，也可以同时进行禁用与约束。仅当报警既已启用且未被有效约束时，该报警才可以变为活动状态。

如果某个报警或子状态未指定约束标记，则效果相当于它有一个总是为“假”的约束标记。

只要转换导致报警脱离有效约束状态，检查逻辑便会确定 InTouch 是否应该将该项放入报警状态。

如果在项目处于报警状态时报警变为有效约束状态，该项目必须强制转换到一个不同的（有效）状态。具体的状态应取决于当前有哪些可用的状态，以及它们是否也被禁用或有效约束。此活动由 InTouch 根据报警类型、极限值等进行处理。

如果报警（或报警子状态）在等待确认时变为有效约束状态，则该项目必须强行转换到一个不同的（有效）状态。与确定项目是否报警一样，InTouch 还必须确定应该处于什么状态。

报警约束标记包含在使用计数与许可证限制中。

使用以下只读标记点域获取报警约束标记的名称：

- AlarmDscInhibitor
- AlarmLoLoInhibitor
- AlarmLoInhibitor
- AlarmHiHiInhibitor
- AlarmHiInhibitor
- AlarmMajDevInhibitor
- AlarmMinDevInhibitor
- AlarmRocInhibitor

这些域返回标记的名称。因此，您可以在 InTouch QuickScript 的间接标记引用中使用该名称来获取报警约束标记的当前值，或更改报警约束标记的值。这样可以在运行时强制启用或有效约束报警组。



## 为单独的标记设置事件属性

定义标记以便进行事件监视时，每次标记的值发生变化，便会有 一条事件消息写入报警系统。事件消息记录值如何改变。例如， 是操作员、I/O、QuickScript，还是系统促使了改变。

- 1 打开“标记名字典”。
- 2 选择现有的标记，或创建一个新的标记，其数据将作为事件 进行记录。
- 3 选择**记录事件**。此时**优先级**框变为可用。为**优先级**输入的值确 定标记的事件优先级。



- 4 在**优先级**框中，指定 1 到 999 之间的一个数字作为事件优先 级。1 是最高的事件优先级，而 999 是最低的。
- 5 单击**保存**。
- 6 单击**关闭**以关闭**标记名字典**对话框。

## 配置报警与事件的全局设置

您可以配置以下全局设置，这些设置应用于应用程序生成的所有 报警与事件：

- 内部报警内存（缓冲区）大小。
- 报警返回到正常是否意味着确认。如需有关详细信息，请参 阅第 110 页的“标记值返回到正常时使用自动确认”。
- 事件记录。
- 报警启用状态在重新启动 WindowViewer 时是否保留。
- 是否将报警确认注释用作一般报警注释。如需有关详细信 息，请参阅第 112 页的“使用报警与确认注释”。

## 配置报警缓冲区大小

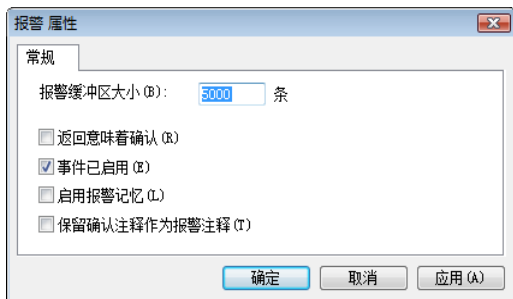
“分布式报警系统”中的通讯很大程度上是由在节点之间发送的报警查询与报警记录组成。在节点内部，报警查询与记录存储在 InTouch 内部报警内存（也称为报警缓冲区）中，以最大程度减少网络流量。报警缓冲区大小是节点可以为摘要或历史报警查询存储的最大报警数。报警缓冲区会删除最旧的记录以便为新记录腾出空间。

只有存储在内存中的报警事件才可以显示在应用程序窗口中。如果 InTouch 应用程序不显示任何报警状态，则可以将缓冲区大小设置为 1 以节省节点内存。

将一个很大的数值指定给报警缓冲区可能会影响节点性能。对于“分布式报警系统”，我们建议使用缺省值 500。

### 要配置报警缓冲区大小

- 1 在 WindowMaker 中打开 InTouch 应用程序。
- 2 在特别菜单上，指向配置，然后单击报警。此时出现报警属性对话框。



- 3 在报警缓冲区大小框中，输入内存报警缓冲区中可以为摘要或历史查询存储的最大报警项数量。
- 4 单击确定。

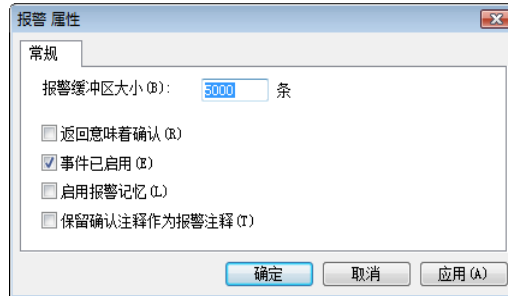
## 启用事件

您可以在应用程序中启用事件记录功能。事件代表由操作员动作、QuickScript 或 I/O 引起且已被认识到的应用程序数据的变化。

在标记的相关事件可以存储到内部报警内存或记录到报警数据库之前，必须从“标记名字典”中设置标记的记录事件属性。如需有关为标记指定事件记录的详细信息，请参阅第 41 页的“为单独的标记设置事件属性”。

### 要启用事件

- 1 在 WindowMaker 中打开 InTouch 应用程序。
- 2 在**特别**菜单上，指向**配置**，然后单击**报警**。此时出现**报警属性**对话框。



- 3 选择**事件已启用**复选框，以记录 InTouch 应用程序运行期间发生的所有事件。
- 4 单击**确定**。

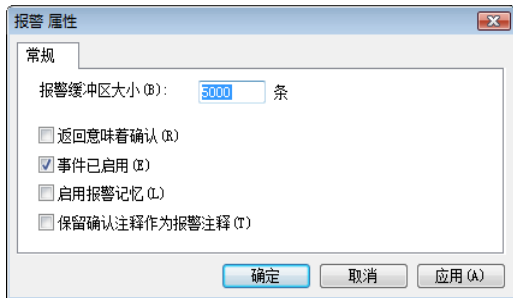
## 保留报警启用功能

您可以选择保留指定给标记的 `AlarmEnabled` 点域的当前值；在停止 InTouch 应用程序，然后再重新启动时，该值会保留下来。

指定给 `AlarmEnabled` 点域的值可以打开或关闭报警与事件记录功能。`.AlarmEnabled` 点域可以指定给标记或报警组。`.AlarmEnabled` 点域指定给报警组时，它确定是否记录与指定的报警组中的标记关联的所有报警。

### 要保留报警启用功能

- 1 在 WindowMaker 中打开 InTouch 应用程序。
- 2 在**特别菜单**上，指向**配置**，然后单击**报警**。此时出现**报警属性**对话框。



- 3 选择**启用报警记忆**复选框，以便重新启动 InTouch 应用程序时，`.AlarmEnabled` 点域的当前状态可以保留下来用作初始值。
- 4 单击**确定**。

## 创建报警组列表文件

您可以使用“分布式名称管理器”创建报警组列表。然后，将现有的报警组从本地与远程节点添加到该列表中。

下表显示从“分布式名称管理器”中指定报警组的语法。

节点	报警组语法
本地	<code>\InTouch!Group_Name</code> 或者 <code>.Group_Name</code>
远程	<code>\\Node_Name\InTouch!Group_Name</code> 或者 <code>Node_Name.Group_Name</code>

对于这些示例，*Node\_Name* 是 InTouch 远程节点的名称。*Group\_Name* 是报警组的名称。如果报警组是在定义报警组列表的本地节点上定义的，则只需输入报警组名，前面加上英文句点。例如，*.Group\_Name*。

### 要创建报警组列表

- 1 在**特别**菜单上，指向**配置**，然后单击**分布式名称管理器**。此时出现**分布式名称管理器**对话框。



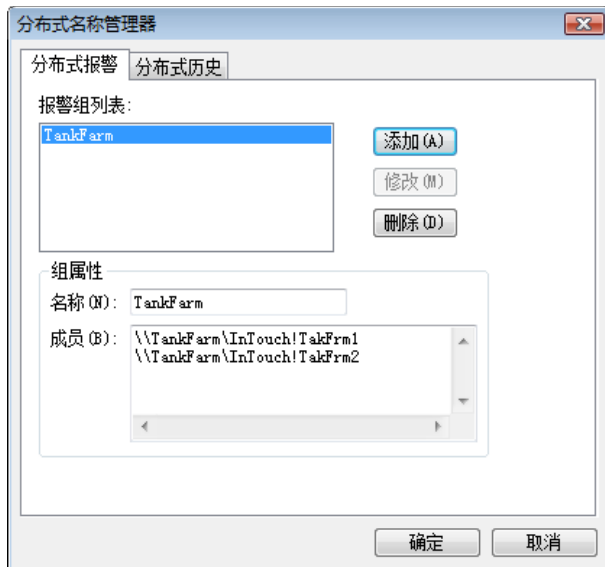
- 2 在**组属性**区域的**名称**框中，输入报警查询的名称。
- 3 在**成员**框中，输入要包含在查询中的 InTouch 节点与报警组的列表。

通过使用“标准的组输入”语法，或使用带英文句点的简短输入格式，可以输入节点名与报警组名。在保存报警组列表时，简短输入格式转换为“标准的组输入”格式。

**备注** Node.Group 与 .Group 语法仅可以用在此配置对话框中。在报警显示配置或任何报警 QuickScript 函数中，它是无效的。

- 单击**添加**将列表添加到报警组文件中。

此时成员的语法会自动转换。



- 单击**确定**。
- 将报警组列表的名称添加到 Alarm Viewer 控件的查询中。  
此时，Alarm Viewer 控件显示列表中指定的所有组的报警。

## 第 3 章

# 报警查询

报警查询检索以下内容之一：

- InTouch 内部报警内存或报警数据库中的报警与事件（历史报警）。
- InTouch 内部报警内存中的当前报警（摘要报警）。

配置 InTouch 报警 ActiveX 控件时，可以指定查询来源。您也可以选择用于过滤查询结果的查询选项。

下图显示 Alarm Viewer ActiveX 控件的**查询**选项卡。

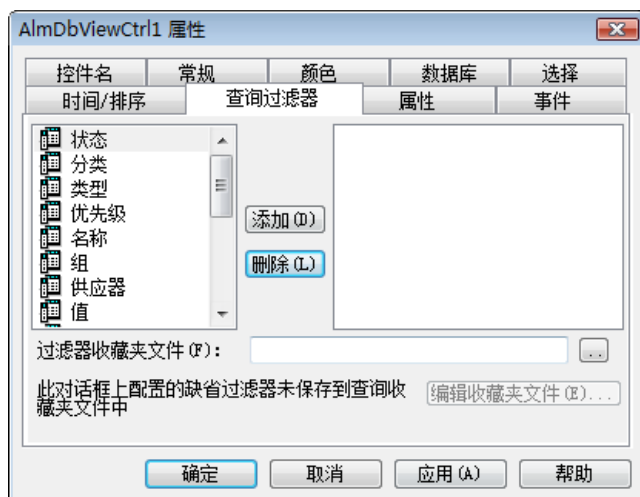


在本例中，您创建一个报警显示对象，显示根据以下准则选择的报警数据：

- 报警优先级 (1-999)
- 报警状态（全部、确认或未确认）
- 查询类型（摘要或历史）
- 报警组（本地或远程数据源）

您可以将查询保存到名称为“query favorites”的 .xml 文件。在运行时，通过运行另一个查询（使用保存到文件中的选择准则），可以使用新的报警数据来更新报警显示。

其它 InTouch 报警 ActiveX 控件提供更全面的查询准则。下图显示 Alarm DB Viewer 控件的**查询过滤器**选项卡。



通过从对话框左侧窗格显示的列表中选择报警或事件属性，您可以构建查询。然后，您给所选的属性指定一个值。最后，您可以通过使用布尔运算符将各个属性组合到一起设置查询过滤器条件。

您可以编写包含查询函数或点域的 QuickScript 从报警内存中选择报警与事件记录。以下 Alarm Viewer 控件语句使用 ApplyQuery() 方法查询报警内存。

```
#AlarmViewerCtrl1.ApplyQuery ("\\InTouch!$System",500,600,"All",
    "Historical");
```

此语句检索由 "\\InTouch!\$System" 查询指定的所有历史报警，优先级在 500 到 600 之间。所选的报警记录出现在 Alarm Viewer 控件显示中。



## 示例报警查询

本地节点上的报警查询遵循此语法：

`\Provider!AlarmGroup`

例如：

`\InTouch!$System`

远程节点使用以下查询语法：

`\\NodeName\Provider!AlarmGroup`

例如：

`\\ProdSvr\InTouch!$System`

查询 Galaxy 的报警时使用以下语法。此语法从特定计算机上特定区域中对象的特定属性获取报警。

`\\NodeName\Galaxy!Area.Object.Attribute`

以下语法从特定的区域获取所有报警：

`\\Galaxy!Area`

以下语法从两个区域获取报警：

`\\Galaxy!Area1    \\Galaxy!Area2`

以下语法从计算机节点（缺省条件下）的“平台”中获取所有报警：

`\\NodeName\Galaxy`

您也可以使用通配符。以下语法从 Area1、Area2、Area3 等中获取所有报警：

`\\Galaxy!Area*`

以下语法从 "Area" 区域中以字符 "Tank" 开始的所有对象中获取所有报警：

`\\Galaxy!Area.Tank*`

## 获取更多有关 InTouch 查询的信息

下表列出一些参考资料，供您获取有关每个 InTouch 查询来源的查询的详细信息。


查询来源	参阅
Alarm DB Logger Manager	第 252 页的 “配置要记录的报警”
Alarm Printer 实用程序	第 215 页的 “配置要打印的报警”
Alarm Viewer 控件	第 61 页的 “配置要显示哪些报警”
Alarm Tree Viewer 控件	第 196 页的 “配置要显示的供应器与组”
Alarm Pareto 控件	第 331 页的 “配置要分析的报警”
分布式报警显示对象	第 388 页的 “配置要显示哪些报警”
Hot Backup Manager	第 368 页的 “创建报警记录映射文件”

# 第 4 章

## 查看当前报警

使用 InTouch Alarm Viewer ActiveX 控件可以查看报警。  
Alarm Viewer 控件有滚动条、可调整大小的列、多个报警选项、更新状态栏、动态显示类型，并且可以根据报警类型显示不同的颜色。

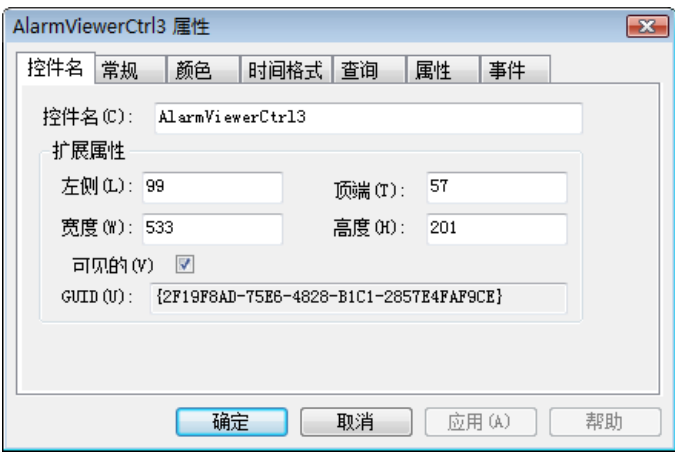
时间 /	状态	分类	类型	优先级
03/28/2007 09:57:38 上午	UNACK_RTN	VALUE	HI	1
03/28/2007 09:57:39 上午	UNACK	VALUE	HI	1

  正在显示第 1 到第 2 个: 默认查询 100 % 完成

我们建议使用 Alarm Viewer 控件来查看 InTouch 报警。不过，您可以继续使用“分布式报警”对象从 InTouch 7.1 以前版本所创建的应用程序中查看报警。

## 配置 Alarm Viewer 控件

从 WindowMaker 中，您可以设置 Alarm Viewer 控件选项，也可以设置用户可以在 Alarm Viewer 控件的运行过程中修改的选项。您在 **AlarmViewerCtrl 属性** 对话框中设置这些选项。

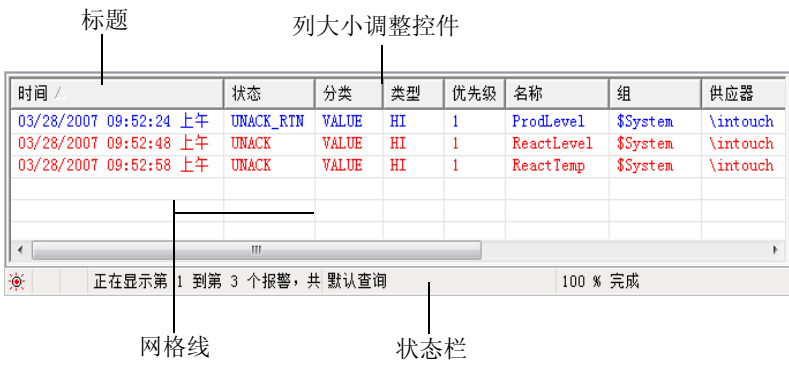


## 配置网格的外观

配置 Alarm Viewer 控件的视觉外观时，可以：

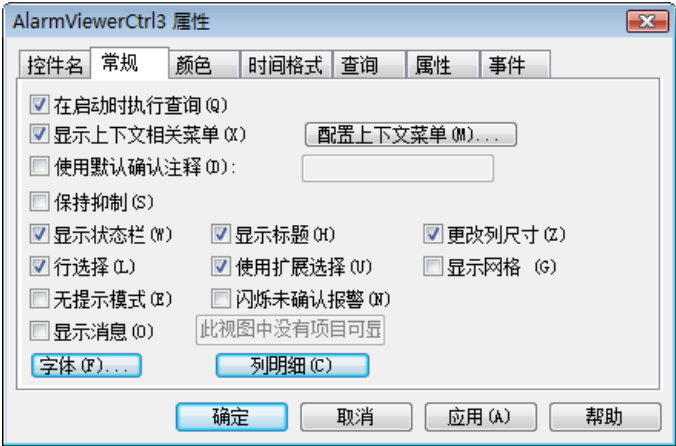
- 包含状态栏。
- 包含列标题。
- 包含显示行与列的水平与垂直网格线。
- 包含供用户调整列宽的运行时选项。
- 设置可视化元素的颜色。

下图显示所有可视化属性都处于活动状态时 Alarm Viewer 控件的外观。



要配置视觉外观

- 1 使用鼠标右键单击 Alarm Viewer 控件，然后单击**属性**。此时出现 **AlarmViewerCtrl 属性**对话框。
- 2 单击**常规**选项卡。



- 3 设置视觉外观。配置以下选项之一：

Option	描述
在启动时执行查询	使用缺省查询属性自动开始更新控件。如果查询未在应用程序启动时运行，则需要运行使用 Requery() 函数的脚本来更新网格。在运行时，网格的快捷菜单上也提供 <b>重新查询</b> 选项。
显示上下文相关菜单	启用运行时的鼠标右键快捷菜单。
使用默认确认注释	控制操作员确认报警时是否出现缺省注释。如果选择此框并且输入了一个字符串，则该字符串用作运行时的缺省注释。  如果未选择此框，则操作员确认报警时，会出现一个对话框，要求输入可选注释。该对话框可以填写，也可以留为空白。
保持抑制	更改报警查询时，在报警查询之间保持报警抑制。
显示状态栏	显示或隐藏 Alarm Viewer 控件底部的状态栏。
行选择	允许用户在运行时选择单独的行。每行代表一条报警记录。用户可以选择多个报警。

Option	描述
无提示模式	如果选择 <b>无提示模式</b> ，则 Alarm Viewer 控件在运行时不显示错误消息。如果未选择它，则“报警显示”显示弹出式错误消息。在任一种情况下，错误消息总是都发送到 ArchestrA Log Viewer。
显示消息	显示文本框中输入的消息。这是没有报警时显示的消息。
显示标题	显示或隐藏 Alarm Viewer 控件顶部的标题栏。
使用扩展选择	允许用户在按住 CTRL 或 SHIFT 键的情况下，结合鼠标按钮来同时选择多个报警。仅当选择了 <b>行选择</b> 复选框时，它才是可用的。
闪烁未确认报警	<p>使未确认的报警每秒闪烁一次，直至它们得到确认为止。</p> <p>在 WindowViewer 中冻结报警显示时，不会使未确认的报警停止闪烁。</p>
更改列尺寸	如果选择 <b>更改列尺寸</b> ，则用户可以在运行时调整列的宽度。否则列宽度是静态的，仅可以从 WindowMaker 中设置。
显示网格	如果选择 <b>显示网格</b> ，则 Alarm Viewer 控件显示分隔报警显示行与列的水平与垂直线。如果未选择它，则网格不可见。

4 单击应用。

5 单击颜色选项卡。



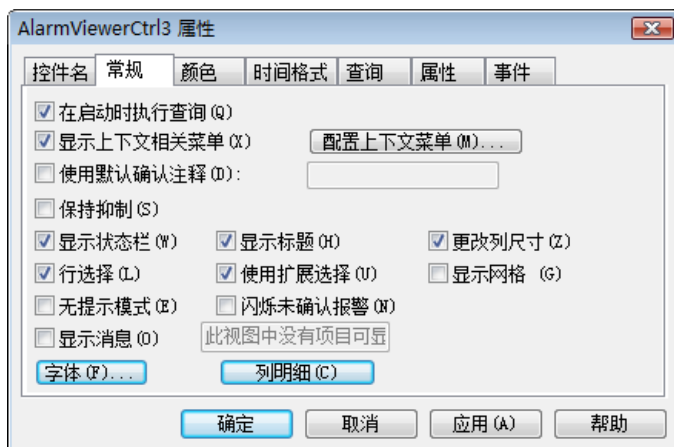
- 6 单击调色板按钮给 Alarm Viewer 控件的可视化元素指定颜色。
- 7 单击应用以保存选择的颜色。
- 8 单击确定。

## 配置显示字体

您可以为 Alarm Viewer 控件配置文本外观。

### 要配置字体属性

- 1 使用鼠标右键单击 Alarm Viewer 控件，然后单击**属性**。此时出现 **AlarmViewerCtrl 属性**对话框。
- 2 单击**常规**选项卡。



- 3 单击**字体**。此时出现标准的 Windows 字体对话框。配置字体，然后单击**确定**。
- 4 单击**确定**。

## 选择与配置显示列

对于 Alarm Viewer 控件，您可以：

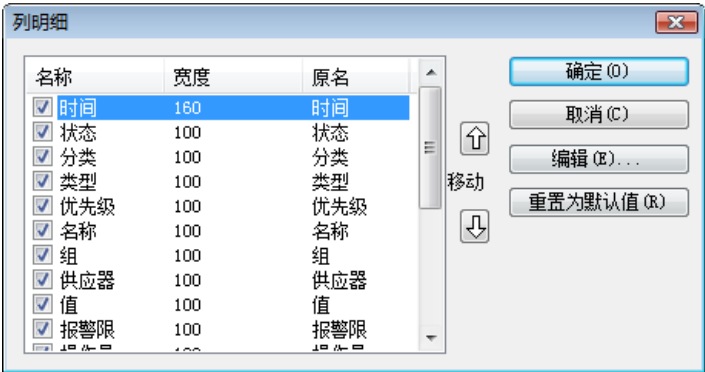
- 选择列并进行排序。
- 以像素为单位设置列的宽度。
- 给某个列重命名。

### 要配置显示列明细

- 1 使用鼠标右键单击 Alarm Viewer 控件，然后单击**属性**。此时出现 **AlarmViewerCtrl 属性**对话框。
- 2 单击**常规**选项卡。



3 单击**列明细**。此时出现**列明细**对话框。



4 在**名称**列中，对于希望出现的列，选择其名称旁边的复选框。您至少必须从列表中选择一列。

列	显示
时间	时间，格式在 <b>时间格式</b> 选项卡中指定。
状态	报警的状态。
分类	报警类别。
类型	报警类型。
优先级	报警优先级。
名称	标记名。
组	报警组名。
供应器	报警供应器的名称。
值	发生报警时标记的值。列宽应该足够大，以提供所需级别的精度。
报警限	标记的报警限值。列宽应该足够大，以提供所需级别的精度。
操作员	已登录且与报警条件关联的操作员的ID。
操作员全名	已登录的操作员的全名。
操作员节点	已登录且与报警条件关联的操作员的节点。  在“终端服务”环境中，这是操作员建立“终端服务”会话的客户端计算机的名称。如果无法检索节点名，则使用节点的IP地址代替。
操作员域	已登录且与报警条件关联的操作员的域。

列	显示
标记注释	标记的注释。
报警注释	与标记的报警关联的注释。此注释是定义标记的报警时在 <b>报警注释</b> 框中输入的。为报警引入某个确认注释时，新的注释更新到这个注释列中。
用户 1	报警的 <b>AlarmUserDefNum1</b> 属性的数值。
用户 2	报警的 <b>AlarmUserDefNum2</b> 属性的数值。
用户 3	报警的 <b>AlarmUserDefStr</b> 属性的字符串值。



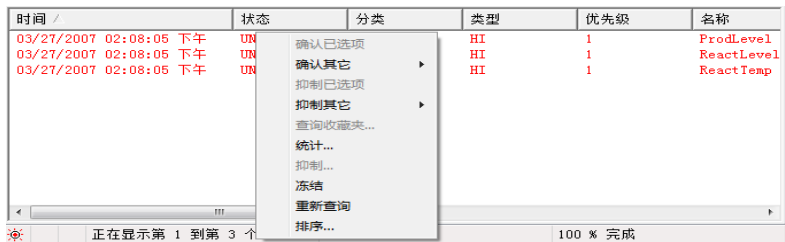
- 5 通过选择列名并使用上、下箭头，重新排列各个列。出现在**列明细**对话框顶部的列名是报警控件最左侧的列。
- 6 要更改某个列的名称或其宽度，请选择该列并单击**编辑**。此时出现**编辑**对话框。



- a 在**新名**框中，输入新的列名。
- a 在**新宽度**框中，输入列宽。列宽可以在从 1 到 999 个像素的范围内。
- b 单击**确定**。
- 7 单击**列明细**对话框中的**确定**。
- 8 单击**应用**。

## 控制用户在运行时可以访问哪些功能

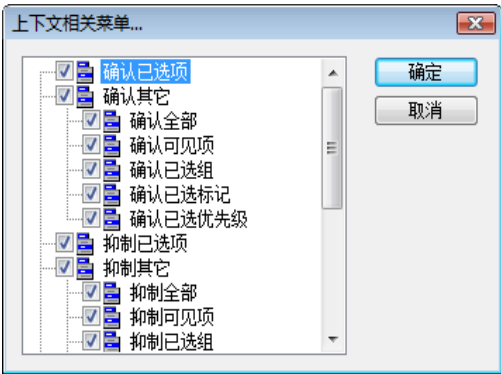
如果在运行时使用鼠标右键单击 Alarm Viewer 控件，则出现一个上下文相关（快捷）菜单。



您可以控制在快捷菜单上显示哪些菜单命令。

### 要配置快捷菜单

- 1 使用鼠标右键单击 Alarm Viewer 控件，然后单击**属性**。此时出现 **AlarmViewerCtrl 属性**对话框。
- 2 单击**常规**选项卡。
- 3 选择**显示上下文相关菜单**复选框。
- 4 单击**配置上下文菜单**。此时出现**上下文相关菜单**对话框。



此对话框显示一个层次结构化的命令列表，其中的命令都可以出现在 Alarm Viewer 控件的快捷菜单上。

- 5 配置快捷菜单选项。您至少必须选择一个快捷菜单命令。

此命令	允许运行时用户
确认已选项	确认所选报警。如果 <b>确认已选项</b> 与 <b>确认其它</b> 菜单项都未选择，则禁用使用 <b>默认确认注释</b> 复选框与文本框。
确认其它	通过其它方法确认报警。用户可以选择确认哪些报警。如果选择 <b>确认其它</b> ，则至少必须选择一个子菜单项。
确认全部	确认所有活动的报警。
确认可见项	确认可见的报警。

此命令	允许运行时用户
确认已选组	确认与所选的组具有相同的组名并且具有相同供应器名的所有报警。
确认已选标记	确认与所选标记具有相同的标记名并且具有相同供应器、组以及优先级的所有报警。
确认已选优先级	确认优先级与所选的一个或多个优先级相同并且具有相同供应器与组的所有报警。
抑制已选项	抑制所选报警。
抑制其它	通过快捷菜单中显示的其它方法抑制报警。
抑制全部	抑制所有报警。
抑制可见项	抑制可见的报警。
抑制已选组	抑制与所选组具有相同组名的所有报警。
抑制已选标记	抑制与所选标记具有相同标记名的所有报警。
抑制已选优先级	抑制优先级与所选的一个或多个优先级相同的所有报警。
全部撤销抑制	撤销抑制的所有报警。
查询收藏夹	打开 <b>报警查询</b> 对话框。
统计	打开 <b>报警统计</b> 对话框。
抑制	打开 <b>报警抑制</b> 对话框。
冻结	在 Alarm Viewer 控件的冻结 / 撤销冻结模式之间切换。
重新查询	重新运行报警查询。
排序	打开 <b>辅助排序</b> 对话框。

- 6 单击**确定**。
- 7 单击**行选择**，以允许用户在运行时从 Alarm Viewer 控件中选择某个行。
- 8 单击**使用扩展选择**，以允许用户使用 SHIFT 或 CTRL 键从 Alarm Viewer 控件中同时选择多条报警记录。
- 9 单击**应用**。

## 配置要显示哪些报警

Alarm Viewer 控件可以显示活动报警的摘要或历史报警的列表。

### 要设置常规报警查询属性

- 1 使用鼠标右键单击 Alarm Viewer 控件，然后单击**属性**。此时出现 **AlarmViewerCtrl 属性**对话框。
- 2 单击**常规**选项卡。
- 3 选择在**启动时执行查询**复选框，以便在应用程序启动时使用缺省查询属性自动更新 Alarm Viewer 控件。
- 4 选择**显示消息**复选框，以便在没有报警时显示缺省消息。在文本框中，输入要显示的消息。
- 5 单击**应用**。

### 要配置查询缺省值

- 1 使用鼠标右键单击 Alarm Viewer 控件，然后单击**属性**。此时出现 **AlarmViewerCtrl 属性**对话框。
- 2 单击**查询**选项卡。



- 3 在**从优先级**框中，输入最小报警优先级值（1 到 999）。
- 4 在**到优先级**框中，输入最大报警优先级值（1 到 999）。
- 5 单击**查询类型**箭头，然后选择**历史**或**摘要**作为缺省的运行时报警显示。

通过运行包含查询函数的 QuickScript，可以在运行时更改显示对象的缺省类型。例如，如果脚本包含 **Type** 参数设置为 "Summary" 的 ApplyQuery() 方法，则网格显示当前报警的摘要。相反地，如果相同的网格运行了 **Type** 参数设置为 "Historical" 的 ApplyQuery() 方法，则显示历史报警。QueryType 属性反映报警显示的当前状态。

如需有关 ApplyQuery() 方法的详细信息，请参阅第 88 页的“ApplyQuery() 方法”。

- 6 在**报警查询**框中，输入一个有效的报警查询。例如，输入  
\\InTouch!\$System 以查询属于缺省的 \$System 报警组的所有报警。
- 7 单击**确定**。

## 使用查询收藏夹创建自定义的保存查询

您可以配置一个查询收藏夹列表，供操作员从快捷菜单中进行选择。

### 要配置查询收藏夹文件

- 1 使用鼠标右键单击 Alarm Viewer 控件，然后单击**属性**。此时出现 **AlarmViewerCtrl 属性** 对话框。
- 2 单击**查询**选项卡。



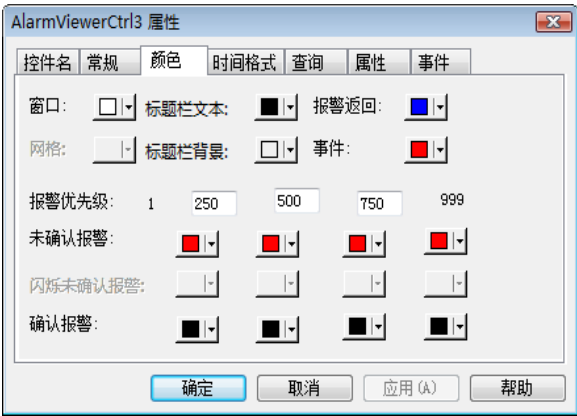
- 3 配置查询收藏夹文件。
  - a 在**查询收藏夹文件**框中，输入网络路径与文件名，或单击省略号按钮以浏览文件。
  - b 要编辑**过滤器收藏夹文件**，请单击**编辑查询收藏夹**按钮。此时打开**报警查询**窗口，可供您在收藏夹文件中添加、修改或删除过滤器。完成时单击**确定**，以保存更改并关闭窗口。
- 4 单击**确定**。

## 为不同类型的报警记录颜色

您可以为 Alarm Viewer 控件中出现的不同报警状态设置颜色选项。

### 要配置报警显示颜色

- 1 使用鼠标右键单击 Alarm Viewer 控件，然后单击**属性**。此时出现 **AlarmViewerCtrl 属性**对话框。
- 2 单击**颜色**选项卡。



- 3 单击每个颜色框以打开调色板。在与以下各项对应的调色板中单击要使用的颜色：

属性	描述
窗口	设置显示背景色。
标题栏文本	设置标题栏文本颜色（仅在选择“显示标题”选项时可用）。
报警返回	设置返回的报警（未经确认即返回到正常状态的报警）的颜色。
网格	设置网格的颜色。缺省情况下网格不显示。缺省的网格颜色是淡灰色。报警对象中网格的颜色自动设置为所选窗口颜色的对比色。
标题栏背景	设置标题栏背景色（仅在选择 <b>显示标题</b> 选项时可用）。
事件	设置事件的颜色。

- 4 在**报警优先级**框中，输入报警优先级数字，这些数字充当不同颜色的断点，这些颜色分别用于标识未确认的报警、确认的报警以及未确认的闪烁报警。
- 5 单击**未确认报警**与**确认报警**颜色框以打开调色板。在调色板中单击要使用的颜色。

- 6 要将报警查询配置成闪烁未确认的报警，请单击**常规**选项卡，选择**闪烁未确认报警**复选框，然后单击**颜色**选项卡并选择**闪烁未确认报警**颜色框。选择要用于每个报警优先级范围的颜色。

**备注** Alarm Viewer 控件无法显示少于一秒的时间内的变化。如果报警状态在一秒内变化了两次，则 Alarm Viewer 控件无法识别这种变化。

- 7 单击**应用**。

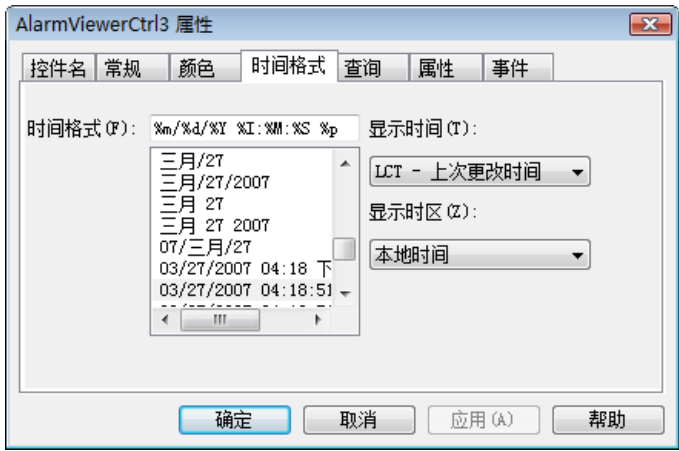
## 配置显示的报警记录时间格式

您可以为显示的报警记录配置时间格式。

对于 Alarm Viewer 控件，原始报警时间是报警发生时的日期 / 时间标签。如果标记为 I/O 标记，并且服务器能够传递时间标签，则它是来自 “I/O 服务器” 的时间标签。

### 要配置时间格式

- 1 使用鼠标右键单击 Alarm Viewer 控件，然后单击**属性**。此时出现 **AlarmViewerCtrl 属性** 对话框。
- 2 单击**时间格式**选项卡。



- 3 在**时间格式**列表中，单击所需的时间格式。**时间格式**框显示所选格式对应的字符组成的一组字符串，由 % 符号分隔。

字符串字符	描述
d	两位日期值。
b	三个英文字符的月份缩写。
Y	四位年份值。
m	两位月份值。
y	两位年份值。



字符串字符	描述
#x	完整的日期与星期。例如：2007 年 8 月 10 日星期五
B	完整的月份名。
H	以 24 小时时间格式表示的小时值。
M	分钟值。
P	AM 或 PM （用于 12 小时时间格式）。
S	秒值。
s	秒值的小数部分。
I	以 12 小时时间格式表示的小时值。

4 在显示时间列表中，选择显示的时间：

OAT	原始报警时间，指报警发生时的日期 / 时间标签。
LCT	上次更改时间，指最近一次报警实例状态变化的日期 / 时间标签：报警发生、子状态改变、返回到正常，或者是确认。
LCT 但确认时为 OAT	上次更改时间，但是确认时为原始报警时间。报警未确认时使用上次更改时间，确认报警后使用原始报警时间。

5 在显示时区列表中，选择时区：

GMT	“格林尼治标准时间”，也称作 “通用协调时间”、UTC 或 Zulu。
本地时间	已调整为本地时区的报警时间。
原始时间	已根据报警源的时区进行调整的报警时间。

6 单击应用。

## 配置报警记录的排序顺序

您可以对列表中的报警记录进行排序。缺省条件下，Alarm Viewer 控件按时间的升序列出报警记录。

您可以按主排序列与可选的辅助排序列的升序或降序对报警记录进行排序。

### 要配置报警记录的排序顺序

- 1 使用鼠标右键单击 Alarm Viewer 控件，然后单击**属性**。此时出现 **AlarmViewerCtrl 属性** 对话框。
- 2 单击**查询**选项卡。



- 3 通过完成以下操作选择排序选项：
  - a 从**排序列**列表中选择主排序列。只有可见的列才出现在**排序列**列表中。如果未看到所需的列，请转到**常规**选项卡，然后从**列明细**中选择该列。
  - b 从**辅助排序列**列表中选择辅助排序列。
  - c 如果已选择“时间”作为主排序列，则**自动滚动到新报警**复选框变为可用状态。如果希望自动滚动并且在发生新报警时显示它们，请选择此选项。
  - d 选择**升序**或**降序**作为排序方向。
- 4 单击**应用**。

## 在运行时使用 Alarm Viewer 控件

Alarm Viewer 控件包含一个快捷菜单，供操作员快速访问可以应用于以下对象的命令：一个或多个所选报警、报警组、标记以及优先级的显示对象。

下表显示 Alarm Viewer 控件快捷菜单上提供的命令：

- **确认已选项** - 确认所选报警。
- **确认其它** - 出现一个列出其它确认命令的子菜单。
  - **确认全部** - 确认当前报警查询中的所有报警。由于报警网格的显示区域有限，因此**确认全部**命令可能会确认网格中不可见的报警。
  - **确认可见项** - 仅确认当前报警网格中可见的那些报警。
  - **确认已选组** - 确认所有的特定报警，它们与一个或多个所选的报警来自相同的供应器，并且具有相同的组名。
  - **确认已选标记** - 确认所有特定的报警，它们具有来自相同供应器与组名的相同标记，并且与一个或多个所选报警具有相同的优先级。
  - **确认已选优先级** - 确认所有特定的报警，它们与一个或多个所选的报警来自相同的供应器与组名，并且具有相同的优先级。
- **抑制已选项** - 抑制所选的一个或多个报警。
- **抑制其它项** - 打开一个包含抑制命令的子菜单。
  - **抑制全部** - 抑制显示当前的所有报警以及将来要发生的这些报警。
  - **抑制可见项** - 抑制显示当前的任何可见报警以及将来要发生的这些报警。
  - **抑制已选组** - 抑制显示当前以及将来要发生的任何特定报警：它们与一个或多个所选的报警属于相同的组，并且具有相同的“供应器名”。
  - **抑制已选标记** - 抑制显示当前以及将来要发生的任何特定报警：它们与一个或多个所选报警属于相同的标记名，并且具有相同的“供应器名”、“组名”及“优先级范围”。
  - **抑制已选优先级** - 抑制显示当前以及将来要发生的任何特定报警：它们与一个或多个所选报警属于相同的优先级，并且具有相同的“供应器名”与“组名”。
  - **全部撤销抑制** - 清除抑制设置。
- **查询收藏夹** - 显示**报警查询**对话框以选择先前保存的报警查询。您也可以添加、修改及删除报警查询。
- **统计** - 显示**报警统计**对话框。
- **抑制** - 显示**报警抑制**对话框。

- **冻结** - 冻结当前显示。
- **重新查询** - 再次查询报警供应器。
- **排序** - 显示**辅助排序**对话框。

## 理解状态栏信息

如果从**常规**属性页中选择**显示状态栏**选项，则运行时在 Alarm Viewer 控件底部出现一个状态栏。



状态栏包含三个指示器：状态消息、当前报警查询以及进度条。这些指示器提供显示查询当前状态的概况，并提供 Alarm Viewer 控件中可用抑制的有关详细信息。冻结控件时状态栏的右侧窗格为红色，抑制一个或多个报警时状态栏的左侧窗格为红色。抑制生效时左侧窗格中显示“抑制”字样。

## 在运行时使用查询收藏夹

通过使用 Alarm Viewer 控件快捷菜单上的**查询收藏夹**命令，可以从先前定义的报警查询列表中快速选择报警查询。您也可以创建新的命名查询、编辑或删除现有的查询。

对报警查询所作的更改不自动应用于使用相同查询的其它 Alarm Viewer 控件。删除某个报警查询时，不会从使用相同查询的其它 Alarm Viewer 控件中自动删除该查询。

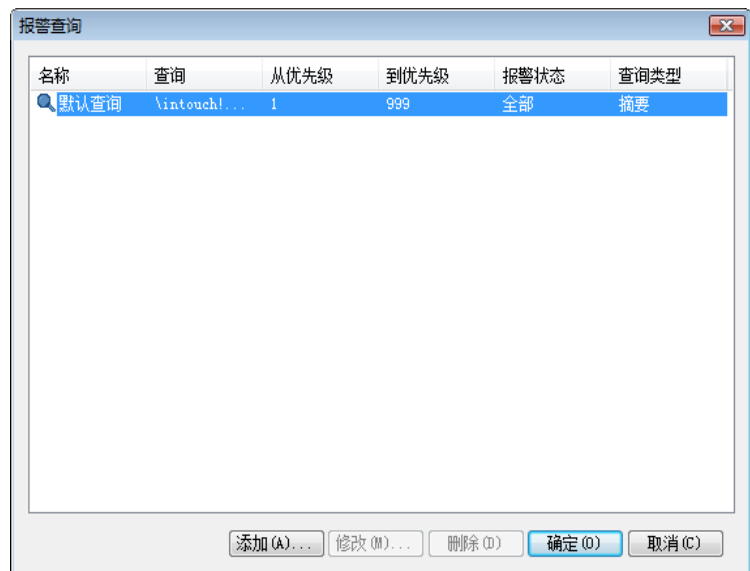
---

**备注** 对于 Alarm Viewer 控件中出现的多行报警查询，换行会出现“乱码”字符。但这不影响功能。

---

### 要在运行时选择报警查询

- 1 使用鼠标右键单击 Alarm Viewer 控件，然后单击**查询收藏夹**。此时出现**报警查询**对话框。



- 2 从当前定义的查询列表中，选择要显示的命名查询。
- 3 单击**确定**。现在 Alarm Viewer 控件显示查询检索的报警信息。

### 要在运行时添加新的命名查询

- 1 使用鼠标右键单击 Alarm Viewer 控件，然后单击**查询收藏夹**。此时出现**报警查询**对话框。
- 2 单击**添加**。此时出现**添加查询**对话框。



- 3 配置查询。执行以下操作：
  - a 在**名称**框中，输入要用于标识查询的名称。
  - b 在**查询**框中，输入希望执行的一组 InTouch 报警查询。您可以指定一个或多个“报警供应器”与组。
  - c 在**从优先级**框中，输入最小报警优先级值（1 到 999）。
  - d 在**到优先级**框中，输入最大报警优先级值（1 到 999）。
  - e 单击**报警状态**箭头，然后选择希望在报警查询中使用的报警状态（**全部**、**确认**、**未确认**）。
  - f 在**显示类型**区域中，选择**摘要**或**历史**作为要查询的记录类型。
- 4 单击**确定**以关闭**添加查询**对话框。
- 5 单击**报警查询**对话框中的**确定**，以便将查询添加到收藏夹。

要在运行时修改现有的命名查询

- 1 使用鼠标右键单击 Alarm Viewer 控件，然后单击**查询收藏夹**。此时出现**报警查询**对话框。
- 2 从当前定义的查询列表中，选择要修改的命名查询。
- 3 单击**修改**。此时出现**修改查询**对话框。
- 4 进行所需的修改，然后单击**确定**。
- 5 单击**报警查询**对话框中的**确定**。

要在运行时删除现有的命名查询

- 1 使用鼠标右键单击 Alarm Viewer 控件，然后单击**查询收藏夹**。此时出现**报警查询**对话框。
- 2 从当前定义的查询列表中，选择要删除的命名查询。
- 3 单击**删除**。出现消息时，单击**是**。
- 4 单击**报警查询**对话框中的**确定**。

## 使用 Alarm Viewer 控件 ActiveX 属性

您可以使用脚本直接设置 Alarm Viewer 控件属性的值，或是将它指定给 InTouch 标记或 I/O 引用。如需有关设置属性的详细信息，请参阅 *InTouch® HMI 脚本与逻辑指南* 中的第 8 章“编写 ActiveX 控件脚本”。

下表列出 Alarm Viewer 控件属性。

属性	类型	用途
AckAllMenu	离散	启用 / 禁用 <b>确认全部</b> 菜单项。
AckAlmColorRange1	整型	设置用于显示优先级范围从 1 到 ColorPriorityRange1 的已确认的报警的颜色。缺省优先级为 1 到 250。
AckAlmColorRange2	整型	设置用于显示优先级范围从 ColorPriorityRange1 到 ColorPriorityRange2 的已确认的报警的颜色。缺省优先级为 250 到 500。
AckAlmColorRange3	整型	设置用于显示优先级范围从 ColorPriorityRange2 到 ColorPriorityRange3 的已确认的报警的颜色。缺省优先级为 500 到 750。

属性	类型	用途
<b>AckAlmColorRange4</b>	整型	设置用于显示优先级范围从 <b>ColorPriorityRange3</b> 到 999 的已确认的报警的颜色。缺省优先级为 750 到 999。
<b>AckOthersMenu</b>	离散	启用 / 禁用 <b>确认其它</b> 菜单项。
<b>AckSelectedGroupsMenu</b>	离散	启用 / 禁用 <b>确认已选组</b> 菜单项。
<b>AckSelectedMenu</b>	离散	启用 / 禁用 <b>确认已选项</b> 菜单项。
<b>AckSelectedPrioritiesMenu</b>	离散	启用 / 禁用 <b>确认已选优先级</b> 菜单项。
<b>AckSelectedTagsMenu</b>	离散	启用 / 禁用 <b>确认已选标记</b> 菜单项。
<b>AckVisibleMenu</b>	离散	启用 / 禁用 <b>确认可见项</b> 菜单项。
<b>AlarmQuery</b>	消息	<p>设置初始报警查询。此域仅接受文本；它不接受标记。下例使用报警组的完整路径：</p> <p>\\Node\InTouch!Group</p> <p>本例使用本地报警组的完整路径：</p> <p>\InTouch!Group</p> <p>本例使用另一个“组列表”：</p> <p>GroupList</p>
<b>AlarmState</b>	消息	要查询的缺省报警状态（“全部”、“未确认”、“确认”）。
<b>AlmRtnColor</b>	整型	设置已返回到正常状态且未确认的报警的颜色。此颜色还用于从已确认状态返回到正常状态但尚未观察到确认状态转换的报警。
<b>AutoScroll</b>	离散	如果用户从开头滚动列表，这会自动跳到新报警。（新报警定义为当前未在显示对象中显示的那些报警）。
<b>ColorPriorityRange1</b>	整型	设置要显示的报警的优先级范围边界。此属性的值必须大于一旦小于 <b>ColorPriorityRange2</b> 的值。
<b>ColorPriorityRange2</b>	整型	设置要显示的报警的优先级范围边界。此属性的值必须大于 <b>ColorPriorityRange1</b> 的值，且小于 <b>ColorPriorityRange3</b> 的值



属性	类型	用途
<b>ColorPriorityRange3</b>	整型	设置要显示的报警的优先级范围边界。此属性的值必须大于 <b>ColorPriorityRange2</b> 的值且小于 999。
<b>ColumnResize</b>	离散	返回或设置一个值，确定是否可以在运行时调整列的大小。
<b>CustomMessage</b>	消息	没有报警时显示的缺省消息。
<b>DefaultAckComment</b>	消息	报警已确认且 <b>"UseDefaultAckComment"</b> 为“真”时用作注释。否则将提示用户输入注释。
<b>DisplayedTime</b>	消息	显示报警消息时间。值只能是 "OAT"、"LCT" 或 “LCT 但确认为 OAT”。
<b>DisplayedTimeZone</b>	消息	获取或设置当前时区字符串。值只能是 "GMT"、“原始时间”或“本地时间”。
<b>EventColor</b>	整型	设置事件的颜色。
<b>ExtendedSelection</b>	离散	允许您通过在按住 Ctrl 或 Shift 键的同时结合使用鼠标按钮来选择多个报警。缺省情况是通过单击它们来切换选择的报警（只有在选择了 <b>行选择</b> 复选框时才可用）。
<b>FlashUnAckAlarms</b>	离散	启用或禁用未确认报警的闪烁。它接受离散输入值 1 或 0。如果此属性设置为 1，未确认的报警每秒闪烁一次。如果此属性设置为 0，则未确认的报警不闪烁。此属性对应于 Alarm Viewer 控件 <b>常规</b> 选项卡上的 <b>闪烁未确认报警</b> 复选框。
<b>FlashUnackAlmColorRange1</b>	整型	为属于“报警优先级范围 1”的未确认报警设置闪烁颜色。
<b>FlashUnackAlmColorRange2</b>	整型	为属于“报警优先级范围 2”的未确认报警设置闪烁颜色。
<b>FlashUnackAlmColorRange3</b>	整型	为属于“报警优先级范围 3”的未确认报警设置闪烁颜色。
<b>FlashUnackAlmColorRange4</b>	整型	为属于“报警优先级范围 4”的未确认报警设置闪烁颜色。
<b>Font</b>	无	为控件中的记录与标题设置字体。
<b>FreezeMenu</b>	离散	启用 / 禁用 <b>冻结</b> 菜单项。
<b>FromPriority</b>	整型	设置缺省查询的最低优先级值。

属性	类型	用途
<b>GridColor</b>	整型	设置背景网格的颜色。
<b>NewAlarmEventMode</b>	整型	控制 NewAlarm 事件的触发。  0 = NewAlarm 事件无法触发。(缺省值)。  1 = NewAlarm 事件是活动的。  2 = NewAlarm 事件是活动的，并且在至少一个新的未确认报警到达时继续触发。
<b>QueryFavoritesFile</b>	消息	返回或设置查询收藏夹文件名。
<b>QueryFavoritesMenu</b>	离散	启用 / 禁用 “查询收藏夹” 菜单项。
<b>QueryName</b>	字符串	返回当前查询名。
<b>QueryStartup</b>	离散	如果设置此属性，则使用缺省查询属性自动开始更新网格。否则，您必须在脚本中运行 ApplyDefaultQuery 或 ApplyQuery 方法才能更新网格。
<b>QueryType</b>	消息	将显示类型设置为 “摘要” 或 “历史”。
<b>RequeryMenu</b>	离散	启用 / 禁用 “重新查询” 快捷菜单项。
<b>RetainSuppression</b>	离散	更改报警查询时在报警查询之间保持报警抑制。
<b>RowSelection</b>	离散	允许用户在运行时选择报警。
<b>SecondarySortColumn</b>	消息	返回或设置当前辅助排序列。
<b>SelectedCount</b>	整型	返回所选报警的总数。
<b>ShowContextMenu</b>	离散	启用快捷菜单功能。
<b>ShowGrid</b>	离散	返回或设置一个值，确定是否在控件中显示网格线。
<b>ShowHeading</b>	离散	显示控件的标题栏。
<b>ShowMessage</b>	离散	返回或设置一个值，确定是否为控件显示错误消息。
<b>ShowStatusBar</b>	离散	获取或设置一个值，确定是否显示状态栏。
<b>SilentMode</b>	离散	获取或设置一个值，确定控件是否处于无提示模式。
<b>SortColumn</b>	消息	获取或设置当前排序列。

属性	类型	用途
<b>SortMenu</b>	离散	启用 / 禁用 <b>排序</b> 菜单项。
<b>SortOrder</b>	离散	获取或设置排序方向。可能值有“升序”与“降序”，分别用 0 与 1 表示。
<b>StatsMenu</b>	离散	启用 / 禁用 <b>统计</b> 菜单项。
<b>SuppressAllMenu</b>	离散	启用 / 禁用 <b>抑制全部</b> 菜单项。
<b>SuppressedAlarms</b>	整型	获取抑制的报警的总数。
<b>SuppressionMenu</b>	离散	启用 / 禁用 <b>抑制</b> 菜单项。
<b>SuppressOthersMenu</b>	离散	启用 / 禁用 <b>抑制其它</b> 菜单项。
<b>SuppressSelectedGroupsMenu</b>	离散	启用 / 禁用 <b>抑制已选组</b> 菜单项。
<b>SuppressSelectedMenu</b>	离散	启用 / 禁用 <b>抑制已选项</b> 菜单项。
<b>SuppressSelectedPrioritiesMenu</b>	离散	启用 / 禁用 <b>抑制已选优先级</b> 菜单项。
<b>SuppressSelectedTagsMenu</b>	离散	启用 / 禁用 <b>抑制已选标记</b> 菜单项。
<b>SuppressVisibleMenu</b>	离散	启用 / 禁用 <b>抑制可见项</b> 菜单项。
<b>TimeFormat</b>	消息	设置报警时间标签的格式。
<b>TitleBackColor</b>	整型	设置标题栏背景色。
<b>TitleForeColor</b>	整型	设置标题栏前景色。
<b>ToPriority</b>	整型	设置报警查询的最高优先级。
<b>TotalAlarms</b>	整型	获取报警数。
<b>UnackAlarms</b>	整型	获取未确认的报警总数。
<b>UnAckAlmColorRange1</b>	整型	设置用于显示优先级范围从 1 到 ColorPriorityRange1 的未确认报警的颜色。
<b>UnAckAlmColorRange2</b>	整型	设置用于显示优先级范围从 ColorPriorityRange1 到 ColorPriorityRange2 的未确认报警的颜色。
<b>UnAckAlmColorRange3</b>	整型	设置用于显示优先级范围从 ColorPriorityRange2 到 ColorPriorityRange3 的未确认报警的颜色。

属性	类型	用途
UnAckAlmColorRange4	整型	设置用于显示优先级范围从 ColorPriorityRange3 到 999 的未确认报警的颜色。
UnsuppressAllMenu	离散	启用 / 禁用 “全部撤销抑制” 菜单项。
UseDefaultAckComment	离散	如果设置为 “真”，则确认报警时使用缺省确认注释。否则提示操作员输入注释。
WindowColor	整型	设置网格背景色。

## 配置 ActiveX 控件的颜色

您可以将颜色指定为整数值。报警 ActiveX 控制使用 ABGR 模型将颜色指定为 32 位整数，其中：

- A = 透明
- B = 蓝色
- G = 绿色
- R = 红色

报警 ActiveX 控件不支持透明度。高 8 位的任何值会被忽略。

例如，要将颜色设置为“蓝色”，使用以下 ABGR 值：

- A = 0
- B = 255
- G = 0
- R = 0

此颜色的十六进制值为 0x00FF0000。十进制值为 16711680。

下表显示可以使用的颜色示例

颜色	十六进制值	十进制值
白色	0x00FFFFFF	16777215
黑色	0x00000000	0
蓝色	0x00FF0000	16711680
红色	0x000000FF	225
绿色	0x0000FF00	652880

## 使用 Alarm Viewer 控件 ActiveX 方法

您可以在脚本中使用 Alarm Viewer 控件 ActiveX 方法来：

- 确认报警。
- 抑制报警。
- 获取报警的相关信息。
- 运行报警查询。
- 移动与冻结显示。
- 给报警记录排序。
- 选择特定的报警。
- 显示快捷菜单、关于对话框以及报警统计对话框。

如需有关调用方法的详细信息，请参阅 *InTouch® HMI 脚本与逻辑指南* 中的第 8 章 “编写 ActiveX 控件脚本”。

### 确认报警

使用以下方法在运行时确认报警。

- AckSelected() 方法
- AckAll() 方法
- AckVisible() 方法
- AckSelectedGroup() 方法
- AckSelectedTag() 方法
- AckSelectedPriority() 方法
- AckGroup() 方法
- AckPriority() 方法
- AckTag() 方法

### AckSelected() 方法

在运行时确认 Alarm Viewer 控件中所选的报警。

#### 语法

*Object.AckSelected (Comment)*

#### 参数

*Comment*

报警确认注释。

#### 示例

Tag1 定义为消息标记，控件名为 AlarmViewerCtrl1。

```
Tag1 = "Alarm Comment";  
#AlarmViewerCtrl1.AckSelected (Tag1);
```

### AckAll() 方法

确认当前报警查询中的所有报警。由于 Alarm Viewer 控件中的显示区域有限，AckAll() 方法可能也会确认未在显示对象中显示的报警。

#### 语法

*Object.AckAll (Comment)*

#### 参数

*Comment*

报警确认注释。

#### 示例

Tag1 定义为消息标记，控件名为 AlarmViewerCtrl1。

```
Tag1 = "Alarm Comment";  
#AlarmViewerCtrl1.AckAll (Tag1);
```

### AckVisible() 方法

仅确认 Alarm Viewer 控件中当前可见的那些报警。

#### 语法

*Object.AckVisible (Comment)*

#### 参数

*Comment*

报警确认注释。

#### 示例

Tag1 定义为消息标记，控件名为 AlarmViewerCtrl1。

```
Tag1 = "Alarm Comment";  
#AlarmViewerCtrl1.AckVisible (Tag1);
```

### AckSelectedGroup() 方法

确认与一个或多个所选报警具有相同组名的所有报警。

#### 语法

*Object.AckSelectedGroup (Comment)*

#### 参数

*Comment*

报警确认注释。

#### 示例

Tag1 定义为消息标记，控件名为 AlarmViewerCtrl1。

```
Tag1 = "Alarm Comment";  
#AlarmViewerCtrl1.AckSelectedGroup (Tag1);
```

### AckSelectedTag() 方法

确认与一个或多个所选报警具有相同标记、组名以及优先级的所有报警。

#### 语法

*Object.AckSelectedTag (Comment)*

#### 参数

*Comment*

报警确认注释。

#### 示例

Tag1 定义为消息标记，控件名为 AlarmViewerCtrl1。

```
Tag1 = "Alarm Comment";  
#AlarmViewerCtrl1.AckSelectedTag (Tag1);
```

### AckSelectedPriority() 方法

确认与一个或多个所选报警具有相同优先级范围的所有报警。

#### 语法

*Object.AckSelectedPriority (Comment)*

#### 参数

*Comment*

报警确认注释。

#### 示例

Tag1 定义为消息标记，控件名为 AlarmViewerCtrl1。

```
Tag1 = "Alarm Comment";  
#AlarmViewerCtrl1.AckSelectedPriority (Tag1);
```



## AckGroup() 方法

确认给定组名与供应器的所有报警。

### 语法

*Object.AckGroup(ApplicationName, GroupName, Comment)*

### 参数

*ApplicationName*

应用程序名，例如 \\node1\Intouch

*GroupName*

组名。例如， Turbine。

*Comment*

报警确认注释。

### 示例

控件名为 AlarmViewerCtrl1。

```
#AlarmViewerCtrl1.AckGroup ("\\Intouch", "Turbine", "Turbine  
acknowledgement Comment");
```

## AckPriority() 方法

确认指定优先级范围内具有相同供应器名与组名的所有报警。

### 语法

*Object.AckPriority(ApplicationName, GroupName, FromPriority,  
ToPriority, Comment)*

### 参数

*ApplicationName*

应用程序的名称。例如， \\node1\Intouch

*GroupName*

组名。例如， Turbine。

*FromPriority*

报警的开始优先级。例如， 100。

*ToPriority*

报警的结束优先级。例如， 900。

*Comment*

报警确认注释。

### 示例

控件名为 AlarmViewerCtrl1。

```
#AlarmViewerCtrl1.AckPriority ("\\Intouch", "Turbine", 100, 900,  
"Turbine acknowledgement Comment");
```

### AckTag() 方法

确认给定标记名的报警，这些报警在给定的优先级范围内并且具有相同的供应器名与组名。

#### 语法

*Object.AckTag(ApplicationName, GroupName, tag, FromPriority, ToPriority, Comment)*

#### 参数

*ApplicationName*

“应用程序”的名称，例如 \\node1\Intouch

*GroupName*

组名。例如， Turbine。

*tag*

报警标记的名称。例如， Valve1。

*FromPriority*

报警的开始优先级。例如， 100。

*ToPriority*

报警的结束优先级。例如， 900。

*Comment*

报警确认注释。

#### 示例

控件名为 AlarmViewerCtrl1。

```
#AlarmViewerCtrl1.AckTag ("\\Intouch", "Turbine", "Valve1", 100, 900, "Turbine acknowledgement Comment");
```

## 抑制报警

使用以下方法在运行时抑制报警：

- ShowSuppression() 方法
- SuppressSelected() 方法
- SuppressAll() 方法
- SuppressVisible() 方法
- SuppressSelectedGroup() 方法
- SuppressSelectedTag() 方法
- SuppressSelectedPriority() 方法
- UnSuppressAll() 方法
- SuppressGroup() 方法
- SuppressPriority() 方法

- SuppressTag() 方法

### ShowSuppression() 方法

显示抑制对话框，其中包含所有抑制的报警。

#### 语法

*Object*.ShowSuppression()

#### 示例

控件名为 AlarmViewerCtrl1。

```
#AlarmViewerCtrl1.ShowSuppression();
```

### SuppressSelected() 方法

抑制显示当前所选的报警以及将来要发生的这些报警。

#### 语法

*Object*.SuppressSelected()

#### 示例

控件名为 AlarmViewerCtrl1。

```
#AlarmViewerCtrl1.SuppressSelected();
```

### SuppressAll() 方法

抑制显示当前所有活动的报警以及将来要发生的这些报警。

#### 语法

*Object*.SuppressAll()

#### 示例

控件名为 AlarmViewerCtrl1。

```
#AlarmViewerCtrl1.SuppressAll();
```

### SuppressVisible() 方法

抑制显示当前的任何可见报警以及将来要发生的这些报警。

#### 语法

*Object*.SuppressVisible()

#### 示例

控件名为 AlarmViewerCtrl1。

```
#AlarmViewerCtrl1.SuppressVisible();
```

### SuppressSelectedGroup() 方法

抑制显示当前以及将来要发生的任何特定报警：它们与一个或多个所选的报警属于相同的“组”与“供应器”。

#### 语法

*Object*.SuppressSelectedGroup()

#### 示例

控件名为 AlarmViewerCtrl1。

```
#AlarmViewerCtrl1.SuppressSelectedGroup();
```

### SuppressSelectedTag() 方法

抑制显示当前以及将来要发生的任何特定报警：它们与一个或多个所选报警属于相同的标记名，并且具有相同的“组名”、“供应器名”、及“优先级范围”。

#### 语法

*Object*.SuppressSelectedTag()

#### 示例

控件名为 AlarmViewerCtrl1。

```
#AlarmViewerCtrl1.SuppressSelectedTag();
```

### SuppressSelectedPriority() 方法

抑制显示当前以及将来要发生的任何特定报警：它们与一个或多个所选报警具有相同的优先级范围。

#### 语法

*Object*.SuppressSelectedPriority()

#### 示例

控件名为 AlarmViewerCtrl1。

```
#AlarmViewerCtrl1.SuppressSelectedPriority();
```

### UnSuppressAll() 方法

清除报警抑制。

#### 语法

*Object*.UnSuppressAll()

#### 示例

控件名为 AlarmViewerCtrl1。

```
#AlarmViewerCtrl1.UnSuppressAll();
```

## SuppressGroup() 方法

抑制显示当前以及将来要发生的任何特定报警：它们属于给定的“组名”。

### 语法

*Object*.SuppressGroup(*ApplicationName*, *GroupName*)

### 参数

*ApplicationName*

“应用程序”的名称，例如 \\node1\Intouch

*GroupName*

组名。例如， Turbine。

### 示例

控件名为 AlarmViewerCtrl1。

```
#AlarmViewerCtrl1.SuppressGroup ("\\Intouch", "Turbine");
```

### SuppressPriority() 方法

抑制显示当前以及将来要发生的任何报警：它们具有指定的优先级范围，并且具有相同的“供应器名”与“组名”。

#### 语法

*Object.SuppressPriority(ApplicationName, GroupName, FromPriority, ToPriority)*

#### 参数

*ApplicationName*

“应用程序”的名称，例如 \\node1\Intouch

*GroupName*

组名。例如， Turbine。

*FromPriority*

报警的开始优先级。例如， 100。

*ToPriority*

报警的结束优先级。例如， 900。

#### 示例

控件名为 AlarmViewerCtrl1。

```
#AlarmViewerCtrl1.SuppressPriority ("\\Intouch", "Turbine", 100, 900);
```

### SuppressTag() 方法

抑制显示当前以及将来要发生的任何特定报警：它们由给定的标记名或组名发出，并且在指定的优先级范围内。

#### 语法

*Object.SuppressTag(ApplicationName, GroupName, tag, FromPriority, ToPriority)*

#### 参数

*ApplicationName*

“应用程序”的名称，例如 \\node1\Intouch。

*GroupName*

组名。例如， Turbine。

*tag*

报警标记的名称。例如， valve 1。

*FromPriority*

报警的开始优先级。例如， 100。

*ToPriority*

报警的结束优先级。例如， 900。

#### 示例

控件名为 AlarmViewerCtrl1。

```
#AlarmViewerCtrl1.SuppressTag ("\\Intouch", "Turbine", "Valve1", 100, 900);
```

## 检索报警的有关信息

使用 `GetItem()` 函数检索有关报警的信息。

### `GetItem()` 方法

返回一个字符串，它对应于指定的行与列上的报警。

#### 语法

*Object.GetItem(Row Number, Column Name)*

#### 参数

##### *RowNumber*

整型表达式，赋值为控件中特定的行。

##### *ColumnName*

字符串表达式，赋值为控件中的列名。

#### 示例

控件名为 `AlarmViewerCtrl1`，`Tag1` 定义为“内存消息”。

```
Tag1 = #AlarmViewerCtrl1.GetItem(1, "Group");
```

## 运行查询

使用以下方法运行查询。

- `ShowQueryFavorites()` 方法
- `Requery()` 方法
- `ApplyQuery()` 方法
- `ApplyDefaultQuery()` 方法
- `SetQueryByName()` 方法

### `ShowQueryFavorites()` 方法

如果 `QueryFavoritesFile` 属性包含有效查询收藏夹文件（采用 .xml 格式）的名称，则显示**报警查询**对话框。

#### 语法

*Object.ShowQueryFavorites()*

#### 示例

控件名为 `AlarmViewerCtrl1`。

```
#AlarmViewerCtrl1.ShowQueryFavorites();
```

### Requery() 方法

再次查询报警供应器。

#### 语法

*Object.Requery()*

#### 示例

控件名为 AlarmViewerCtrl1。

```
#AlarmViewerCtrl1.Requery();
```

### ApplyQuery() 方法

按照指定的“报警查询”、“从优先级”、“到优先级”、要查询报警的“状态”以及要检索报警的“类型”等参数来执行查询。

#### 语法

*Object.ApplyQuery(AlarmQuery, FromPriority, ToPriority, State, Type)*

#### 参数

##### *AlarmQuery*

报警查询。例如：\InTouch!\\$System

##### *FromPriority*

报警的开始优先级。例如， 100。

##### *ToPriority*

报警的结束优先级。例如， 900。

##### *State*

指定要显示报警的类型。例如，“未确认”或消息标记。有效状态包括“全部”、“未确认”或“确认”。

##### *Type*

指定查询的类型，例如 Historical（历史报警）或 Summary（摘要报警）。

#### 示例

控件名为 AlarmViewerCtrl1。

```
#AlarmViewerCtrl1.ApplyQuery ("\"InTouch!\$System",100,900,"All",  
    "Historical");
```

### ApplyDefaultQuery() 方法

按照设计时指定的 FromPriority、ToPriority、AlarmState、QueryType 以及 AlarmQuery 等属性来执行查询。缺省属性仅能在开发时更改，不能由其它报警查询来改写。

#### 语法

*Object.ApplyDefaultQuery()*

#### 示例

控件名为 AlarmViewerCtrl1。

```
#AlarmViewerCtrl1.ApplyDefaultQuery();
```



## SetQueryByName() 方法

设置由查询名指定的当前查询。查询名定义在查询收藏夹文件中。

### 语法

*Object*.SetQueryByName(*QueryName*)

### 参数

#### *QueryName*

使用“查询收藏夹”创建的查询的名称，例如 Turbine Queries。

### 示例

控件名为 AlarmViewerCtrl1。

```
#AlarmViewerCtrl1.SetQueryByName("Turbine Queries");
```

## 移动与冻结显示

使用以下函数移动或冻结显示：

- MoveWindow() 方法
- FreezeDisplay() 方法

### MoveWindow() 方法

在控件中按指定的方法滚动报警。

#### 语法

*Object*.MoveWindow(*Option*, *Repeat*)

#### 参数

##### *Option*

要执行的操作的类型。

类型	描述
LineDn	向下滚行。Repeat 参数控制要滚动的行数。
LineUp	向上滚行。Repeat 参数控制要滚动的行数。
PageDn	向下翻页。Repeat 参数控制要滚动的页数。
PageUp	向上翻页。Repeat 参数控制要滚动的页数。
Top	滚动到控件顶部。
Bottom	滚动到控件底部。
PageRt	向右翻页。Repeat 参数控制要滚动的页数。
PageLf	向左翻页。Repeat 参数控制要滚动的页数。
Right	向右滚动。Repeat 参数控制要滚动的列数。
Left	向左滚动。Repeat 参数控制要滚动的列数。
Home	滚动到控件顶行最左一列。

##### *Repeat*

应重复此操作的次数。

#### 示例

控件名为 AlarmViewerCtrl1。

```
#AlarmViewerCtrl1.MoveWindow ("Bottom", 0);  
#AlarmViewerCtrl1.MoveWindow ("LineUp", 3);  
#AlarmViewerCtrl1.MoveWindow ("PageLf", 7);
```

## FreezeDisplay() 方法

冻结显示。

### 语法

*Object*.FreezeDisplay(*Freeze*)

### 参数

*Freeze*

True = 冻结显示。

False = 撤销冻结显示。

### 示例

Tag1 定义为“内存离散”标记，控件名为 AlarmViewerCtrl1。

```
Tag1 = 1;
```

```
#AlarmViewerCtrl1.FreezeDisplay(Tag1);
```

## 给报警记录排序

使用以下函数给报警记录排序：

- ShowSort() 方法
- SetSort() 方法

## ShowSort() 方法

如果启用了 SortMenu 属性，则显示**第二位排序**对话框。

### 语法

*Object*.ShowSort()

### 示例

控件名为 AlarmViewerCtrl1。

```
#AlarmViewerCtrl1.ShowSort();
```

### SetSort() 方法

按照指定的 SortColumn 与 SortOrder 属性来设置排序准则。

#### 语法

*Object*.SetSort()

#### 示例

控件名为 AlarmViewerCtrl1。

```
#AlarmViewerCtrl1.SetSort();
```

## 显示其它信息

使用以下函数显示关于对话框与报警统计对话框：

- AboutBox() 方法
- ShowStatistics() 方法

### AboutBox() 方法

显示关于对话框。

#### 语法

*Object*.AboutBox

#### 示例

控件名为 AlarmViewerCtrl1。

```
#AlarmViewerCtrl1.AboutBox();
```

### ShowStatistics() 方法

显示报警统计对话框。

#### 语法

*Object*.ShowStatistics()

#### 示例

控件名为 AlarmViewerCtrl1。

```
#AlarmViewerCtrl1.ShowStatistics();
```

## 选择特定的报警

使用以下函数选择特定的报警：

- `SelectGroup()` 方法
- `SelectPriority()` 方法
- `SelectTag()` 方法
- `SelectAll()` 方法
- `SelectItem()` 方法
- `UnSelectAll()` 方法

### SelectGroup() 方法

选择包含相同报警组名与供应器名的所有报警。

#### 语法

*Object*.SelectGroup(*ApplicationName*, *GroupName*)

#### 参数

*ApplicationName*

“应用程序”的名称，例如 \\node1\Intouch

*GroupName*

组名。例如， Turbine。

#### 示例

控件名为 AlarmViewerCtrl1。

```
#AlarmViewerCtrl1.SelectGroup ("Intouch", "Turbine");
```

### SelectPriority() 方法

选择指定的优先级范围内具有相同供应器名与组名的所有报警。

#### 语法

*Object.SelectPriority(ApplicationName, GroupName, FromPriority, ToPriority)*

#### 参数

*ApplicationName*

“应用程序”的名称，例如 \\node1\Intouch

*GroupName*

组名。例如， Turbine。

*FromPriority*

报警的开始优先级。例如， 100。

*ToPriority*

报警的结束优先级。例如， 900。

#### 示例

控件名为 AlarmViewerCtrl1。

```
#AlarmViewerCtrl1.SelectPriority ("\\Intouch", "Turbine", 100, 900);
```

### SelectTag() 方法

选择来自特定“供应器 / 组 / 标记”的所有报警。您也可以指定“优先级范围”，或使用 1-999。

#### 语法

*Object.SelectTag(ApplicationName, GroupName, tag, FromPriority, ToPriority)*

#### 参数

*ApplicationName*

“应用程序”的名称，例如 \\node1\Intouch

*GroupName*

组名。例如， Turbine。

*tag*

报警标记的名称。例如， valve 1。

*FromPriority*

报警的开始优先级。例如， 100。

*ToPriority*

报警的结束优先级。例如， 900。

#### 示例

控件名为 AlarmViewerCtrl1。

```
#AlarmViewerCtrl1.SelectTag ("\\Intouch", "Turbine", "Valve1",100, 900);
```

### SelectAll() 方法

切换对显示对象中所有报警的选择。由于报警显示对象的显示区域有限，因此 SelectAll() 函数可能会选择到显示对象中不可见的报警。

#### 语法

*Object*.SelectAll()

#### 示例

```
#AlarmViewerCtrl1.SelectAll();
```

### SelectItem() 方法

切换对给定行上某个报警记录的选择。

#### 语法

*Object*.SelectItem(*RowNumber*)

#### 参数

*RowNumber*

整数值，指定要选择的报警记录的行号。控件中的第一行是 0。

#### 示例

控件名为 AlarmViewerCtrl1，Tag1 定义为内存整型。这会切换对 Alarm Viewer 控件中第十条报警记录的选择。

```
Tag1 = 9;
```

```
#AlarmViewerCtrl1.SelectItem (Tag1);
```

### UnSelectAll() 方法

取消选择所选的全部记录。

#### 语法

*Object*.UnSelectAll()

#### 示例

控件名为 AlarmViewerCtrl1。

```
#AlarmViewerCtrl1.UnSelectAll();
```

## 显示上下文菜单

使用 `ShowContext()` 方法在运行时显示快捷菜单。

### ShowContext() 方法

如果启用任何一个快捷菜单项，则显示快捷菜单。

#### 语法

```
Object.ShowContext()
```

#### 示例

控件名为 `AlarmViewerCtrl1`。

```
#AlarmViewerCtrl1.ShowContext();
```

## 使用方法与属性时的错误处理

您可以使用 `SilentMode` 属性隐藏运行时的错误。如果 `SilentMode` 属性设置为 1，则 `Alarm Viewer` 控件在运行时不显示错误消息。如果设置为 0，则 `Alarm Viewer` 控件显示错误消息。错误消息总是发送到 `Archestra Log Viewer`。

## 使用 ActiveX 事件触发脚本

您可以将 `QuickScript` 指定给 `Alarm Viewer` 控件事件（如鼠标单击或双击）。该事件发生时，此 `QuickScript` 便会运行。

`Alarm Viewer` 控件支持以下事件：

- Click
- DoubleClick
- NewAlarm
- ShutDown
- StartUp

Click 事件有一个参数 `ClicknRow`，它可以确定运行时所单击的行。

DoubleClick 事件有一个参数 `DoubleClicknRow`，它可以确定运行时所双击的行。

Click 与 DoubleClick 事件都是零基的。向用户发布 Click 与 / 或 DoubleClick 事件时，显示中的行计数从 0 开始。

---

**备注** `Alarm Viewer` 控件从 `StartUp` 事件调用方法时会忽略用户界面方法，原因在于控件尚不可见。这些包括：`ShowSort()`、`ShowContext()`、`GetSelectedItem()`、`GetNext()`、`GetPrevious()` 以及 `AboutBox()`。

---



如需有关编写 ActiveX 事件脚本的详细信息，请参阅 *InTouch® HMI 脚本与逻辑指南* 中的第 8 章“编写 ActiveX 控件脚本”。

## 检测到新报警时运行脚本

您可以将 Alarm Viewer 控件配置成在检测到新的未确认报警（也就是，由正常状态转到未确认状态，并且满足控件的查询与优先级过滤准则的任何报警）时运行 ActiveX 事件脚本。

NewAlarmEventMode 属性控制 NewAlarm 事件的触发。

- 如果将 NewAlarmEventMode 属性设置为 0，则不触发 NewAlarm 事件。这是缺省值。
- 如果将 NewAlarmEventMode 属性设置为 1，则在以下情况中会发出新报警：
  - 触发了事件。
  - 运行与 NewAlarm 事件关联的 ActiveX 事件脚本。
  - NewAlarmEventMode 属性设置为 0。

您必须将 NewAlarmEventMode 属性更改回 1 才能处理后续事件。

如果直到纠正或确认条件之后应用程序才应该再次执行某个操作，请使用此设置。例如，触发事件时，ActiveX 脚本可以发出报警声，直至报警得到确认为止。随后在下次收到新报警时，可以再次发出报警声。

- 如果将 NewAlarmEventMode 属性设置为 2，则 NewAlarm 事件是活动的，并且在至少一个新的未确认报警到达时继续触发。您不需要更改这个值便可以处理后续事件。新事件每秒最多触发一次，不论同一秒内是否还有其它新的报警到达。



## 第 5 章

### 实时确认报警

标记数据从正常转入报警状态时，生成一个新的报警实例。

“**InTouch** 分布式报警系统”通过以下状态跟踪每个报警实例：

- 标记第一次进入报警状态时
- 如果该报警是一个多状态报警，则是报警发生子状态转换时
- 报警返回到正常时
- 报警是否正在等待确认
- 确认报警时

与报警关联的标记值返回正常的非报警状态时，报警实例的生命周期便会结束。进入报警状态的后续转换会生成新的报警实例。

## 理解报警确认模型

InTouch HMI 支持三种报警确认模型。

- 对于面向条件的报警，在进行确认之前，确认过程会统计进入报警状态的所有项目。
- 对于扩展的摘要报警，确认只针对特定的转换，无论是进入报警状态、子状态，还是恢复到正常状态。进入不同报警子状态的所有转换都必须得到确认，之后整个报警才会视为已确认。
- 对于面向事件的报警（例如在 OPC 中的情况），仅当确认指最新的激活事件时，它才会被接受。

### 条件确认报警模型

对于面向条件的报警，在进行确认之前，确认过程会统计进入报警状态的所有项目。

确认针对的是报警的实例。报警实例在第一次进入报警状态时，便开始等待确认。如果报警得到确认，并随后转入新的报警子状态（例如从 "Hi" 转入 "HiHi"），则它开始等待下一次确认。只要得到确认，它都会被接受并应用于目前为止已发生的所有报警状态转换。

最新的实例得到确认时，报警会视为已确认。

### 扩展的摘要报警模型

对于扩展的摘要报警，确认只针对特定的转换，无论是进入报警状态、子状态，还是恢复到正常状态。进入不同报警子状态的所有转换都必须得到确认，之后整个报警才会视为已确认。

首次进入报警状态必须得到确认，返回到正常也必须单独确认。

进入新报警子状态的任何转换都视为必须确认的新实例，其返回到正常状态的转换也必须得到确认。子状态转换视为属于“返回正常组”，自项目从先前的正常状态首次进入报警状态开始。

如果项目恢复正常，随后再次进入报警状态，则会创建新的返回正常组。

每个转换都必须明确地单独进行确认，仅当项目恢复正常并且所有待返回正常组中的所有转换都得到确认时，报警才视为已确认。

---

**备注** “摘要”一词表示“等待确认”。此模型中的报警也称为“回铃”报警。

---

### 扩展的摘要报警记录

对于扩展的摘要报警，发生报警时，在报警显示对象中生成一条记录，显示已发生报警条件。此记录显示报警的日期与时间标签。此记录保持可见，直到操作员确认报警并发生返回到正常状态为止。如果在确认报警之前发生返回到正常状态，则报警对象中显示两条记录。

例如，锅炉温度超出 **high** 限状态并触发报警。随后，在操作员确认报警之前锅炉返回到正常温度范围。报警系统生成一条记录。指出发生了 **high** 限报警；同时还生成另一条记录，指出报警尚未确认。

### 使用扩展的摘要报警

定义标记并选择**扩展的摘要**作为确认模型时，操作员必须确认发生的报警，即便触发报警的状态已返回到正常。确认报警会更改报警项的颜色，但不会更改时间标签。仅当报警得到确认并已返回到正常状态时，报警才从显示中清除。

---

**备注** 使用扩展的摘要确认模式定义标记时，**报警属性**对话框中的**返回意味着确认**选项不应用于该标记。

---

## 基于事件的报警模型

对于面向事件的报警（例如在 OPC 中的情况），仅当确认指最新的激活事件时，它才会被接受。

报警实例第一次进入报警状态时，便开始等待确认。如果实例得到确认，并随后转入新的报警子状态，则它开始等待下一次确认。每个后续转换都会被指定一个序号，确认必须在其后附加与之对应的转换的序号。

确认只有在针对最近一次转换时才会被接受。如果被接受，它将应用于该报警目前为止发生的所有状态转换。最新的实例得到确认时，报警会视为已确认。

被拒绝的确认可能会记录下来以用于诊断目的，但系统中不会以其它方式进行跟踪。

面向事件的模型确保报警在不同的状态之间变化时，确认可以对应到当前的信息。在延迟时间较短的系统中，这可能看上去像是面向条件的报警。在其它环境（如 Internet）中，此模型的功能可能会变得非常重要。

## 在运行时检查标记的确认模型

使用 `.AlarmAckModel` 点域监视与标记关联的确认模型。

### **.AlarmAckModel** 点域

监视与标记关联的确认模型，具体如下：

0 = 条件（缺省值）

1 = 面向事件

2 = 扩展的摘要

#### 类别

报警

#### 用法

`Tagname.AlarmAckModel`

#### 参数

*Tagname*

任何离散、整型、实型、间接离散以及模拟标记。

#### 附注

此点域的缺省值为 0（条件确认模型）。

#### 数据类型

模拟（只读）

#### 有效值

0、1 或 2

#### 示例

如果 `PumpStation` 标记与事件报警关联，则会处理此 IF-THEN 语句的主体。

```
IF (PumpStation.AlarmAckModel == 1) THEN
    MyAlarmMessage="PumpStation is an Event
    alarm";
ENDIF;
```

#### 另请参阅

`.Alarm`, `.Ack`, `.UnAck`, `.AckDev`, `.AckDSC`, `.AckROC`

## 使用点域确认报警

您可以创建使用点域的脚本来确认所有的当前报警、所选报警或仅限特定类型的报警。

### 确认报警或报警组

您可以创建使用以下点域的脚本来确认指定的本地标记的报警，或指定的报警组中的报警。

- .Ack 点域
- .UnAck 点域

您无法使用这些点域来确认 Wonderware Application Server 产生的报警。

要确认正在运行的 InTouch 应用程序的所有本地报警，请将 \$System 报警组与适当的 .Ack 点域结合起来使用。

#### .Ack 点域

监视或控制所有类型的本地报警的报警确认状态。

##### 类别

报警

##### 用法

```
TagName.Ack=1;
```

##### 参数

*TagName*

任何离散、整型、实型、间接离散与模拟标记，或报警组。

##### 附注

将此点域设置为值 1 以确认与指定的标记或报警组关联的任何未确认报警。指定的标记是报警组时，与指定的组中的标记关联的所有未确认报警都会得到确认。指定的标记是报警组之外的任何其它类型时，则只有与该标记关联的未确认报警才会得到确认。将 .Ack 点域设置为 1 以外的值没有任何意义。

##### 数据类型

离散（可读写）

##### 有效值

1

##### 示例

以下语句确认与 Tag1 标记关联的报警：

```
Tag1.Ack=1;
```

下例用于确认 PumpStation 报警组内所有未确认的报警：

```
PumpStation.Ack=1;
```

---

**备注** .ACK 点域有一个逆向点域 .UnAck。出现未确认的报警时，.UnAck 设置为 1。随后 .UnAck 可以在动画链接或条件脚本中用于触发任何未确认报警的触发器。

---

**另请参阅**

Alarm, UnAck, AckDev, AckROC, AckDSC, AckValue, AlarmAckModel

**.UnAck 点域**

监视或控制本地报警的报警确认状态。

**类别**

报警

**用法**

```
TagName.UnAck=0;
```

**参数**

*TagName*

任何离散、整型、实型、间接离散与模拟标记，或报警组。

**附注**

将此点域设置为值 0 以确认与指定的标记或报警组关联的任何未确认报警。指定的标记是报警组时，与指定的组中的标记关联的所有未确认报警都会得到确认。指定的标记是任何其它类型时，只有与该标记关联的未确认报警才会得到确认。将此点域设置为 0 以外的值没有任何意义。

**数据类型**

仅限离散（只读 / 复位）

**有效值**

0

**示例**

以下语句确认与 Tag1 标记关联的任何报警。

```
Tag1.UnAck=0;
```

此语句确认报警组 PumpStation 内所有未确认的报警。

```
PumpStation.UnAck=0;
```

.UnAck 有一个逆向点域 .Ack。已确认报警之后，.Ack 点域的值设置为 1。

**另请参阅**

.Ack, Ack(), .Alarm, .AlarmAckModel



## 确认值报警

您可以创建使用 `.AckValue` 点域的脚本来确认指定的本地标记的所有值报警，或指定的报警组中的所有值报警。

### `.AckValue` 点域

监视或控制本地值报警的确认。

#### 用法

```
TagName.AckValue=1;
```

#### 参数

*TagName*

任何整型、实型、间接模拟标记或报警组。

#### 附注

将 `.AckValue` 点域设置为 1 以确认与指定的标记 / 组关联的任何未确认的值报警。指定的标记是报警组时，与指定的组中的标记关联的所有未确认的值报警都会得到确认。指定的标记是任何其它类型时，只有与该标记关联的未确认的值报警才会得到确认。

将此点域设置为 1 以外的值没有任何意义。

#### 数据类型

离散（可读写）

#### 有效值

1

#### 示例

以下语句确认与 `Tag1` 标记关联的值报警：

```
Tag1.AckValue=1;
```

本例确认报警组 `PumpStation` 内所有未确认的值报警：

```
PumpStation.AckValue=1;
```

间接报警组（使用“组变量”）可以用来确认值报警。例如，使用以下赋值语句：

```
StationAlarms.Name = "PumpStation";
```

其中，`StationAlarms` 定义为报警组类型的标记，然后与 `PumpStation` 关联。因此，下面的语句与上例类似，只是它用于确认 `PumpStation` 报警组中任何未确认的值报警，该报警组当前与 `StationAlarms` 报警组标记关联。

```
StationAlarms.AckValue=1;
```

#### 另请参阅

`.Alarm`, `.AlarmValue`, `.Ack`, `.UnAck`, `.AckDev`, `.AckDSC`, `.AckROC`, `.AlarmAckModel`

## 确认离散报警

您可以创建使用 `.AckDsc` 点域的脚本来确认指定的标记的离散报警，或指定的报警组的所有离散报警。

### `.AckDsc` 点域

确认指定的标记的离散报警，或指定的报警组的所有离散报警。

#### 用法

`TagName.AckDsc=1;`

#### 参数

*TagName*

指定给离散标记的名称，或报警组的名称。

#### 附注

设置为 1 以确认与指定的标记或报警组关联的活动离散报警。指定的标记是报警组时，与指定的组中的标记关联的所有未确认的离散报警都会得到确认。指定的标记是报警组之外的任何类型时，只有与该标记关联的未确认的离散报警才会得到确认。将此点域设置为 1 以外的值没有任何意义。

#### 数据类型

离散（可读写）

#### 有效值

0 或 1

#### 示例

以下语句验证 `Tag1` 是否有关联的活动离散报警：

```
IF (Tag1.AlarmDsc == 1) THEN
    MyAlarmMessage="The pumping station currently
    has an ALARM!";
ENDIF;
```

此点域未链接到 `.Ack` 或 `.UnAck` 点域。因此，即使已经确认某个活动的报警，此点域仍等于 1。

#### 另请参阅

`.Alarm`, `.AlarmDSC`, `.Ack`, `.UnAck`, `.AckDev`, `.AckDSC`,  
`.AckROC`, `.AckValue`, `.AlarmAckModel`

## 确认偏差报警

您可以创建使用 `.AckDev` 点域的脚本来确认指定的本地标记主、副偏差报警，或指定的报警组的偏差报警。

### `.AckDev` 点域

确认指定的本地标记的主、副偏差报警，或指定的报警组的所有偏差报警。

#### 类别

报警

#### 用法

```
TagName.AckDev=1;
```

#### 参数

*TagName*

任何整型、实型、间接模拟标记，或报警组。

#### 附注

将此点域设置为值 1 以确认与指定的标记或报警组关联的任何未确认的偏差报警。指定的标记是报警组时，与指定的组中的标记关联的所有未确认的偏差报警都会得到确认。将此点域设置为 1 以外的值没有任何意义。

#### 数据类型

离散（可读写）

#### 有效值

1

#### 示例

以下语句确认与 `Tag1` 标记关联的偏差报警：

```
Tag1.AckDev=1;
```

接下来的这个示例用于确认 `PumpStation` 报警组中所有未确认的偏差报警：

```
PumpStation.AckDev=1;
```

#### 另请参阅

`.Alarm`, `.AlarmDev`, `.Ack`, `.UnAck`, `.AckDSC`, `.AckROC`,  
`.AckValue`, `.AlarmAckModel`

## 确认变化率报警

您可以创建使用 `.AckROC` 点域的脚本来确认指定的本地标记的变化率报警，或指定的报警组的变化率报警。

### **.AckROC 点域**

确认指定的本地标记的变化率报警，或指定的报警组的所有变化率报警。

#### **用法**

```
TagName.AckROC=1;
```

#### **参数**

*TagName*

指定给整型、实型、间接模拟标记的名称，或报警组的名称。

#### **附注**

将此点域设置为值 1 以确认与指定的标记或报警组关联的任何未确认的变化率报警。指定的标记是报警组时，与指定的组中的标记关联的所有未确认的变化率报警都会得到确认。指定的标记是报警组之外的任何其它类型时，只有与该标记关联的未确认的变化率报警才会得到确认。将此点域设置为 1 以外的值没有任何意义。

#### **数据类型**

离散（可读写）

#### **有效值**

1

#### **示例**

以下语句确认与 `Tag1` 标记关联的变化率报警。

```
Tag1.AckROC=1;
```

接下来的这个示例确认 `PumpStation` 报警组内所有未确认的变化率报警：

```
PumpStation.AckROC=1;
```

#### **另请参阅**

`.Alarm`, `.AlarmROC`, `.Ack`, `.UnAck`, `.AckDev`, `.AckDSC`,  
`.AckValue`, `.AlarmAckModel`

## 使用脚本函数确认报警

您可以使用 Ack() 脚本函数确认标记或组的所有报警。

如果使用 Alarm Viewer 控件，则可以使用方法来确认报警。如需有关详细信息，请参阅第 78 页的“确认报警”。

如果使用“分布式报警显示”对象，则可以使用脚本函数来确认报警。如需有关详细信息，请参阅第 402 页的“确认报警”。

### Ack() 函数

确认任何未确认的 InTouch 报警。

#### 类别

报警

#### 语法

```
Ack TagName;
```

#### 参数

##### *TagName*

任何 InTouch 标记、报警组或组变量。

#### 附注

此函数可应用于标记或报警组。

#### 示例

以下语句可以用在按钮上，以确认任何未确认的报警：

```
Ack $System; {All alarms}
```

```
Ack Tagname;
```

```
Ack GroupName;
```

#### 另请参阅

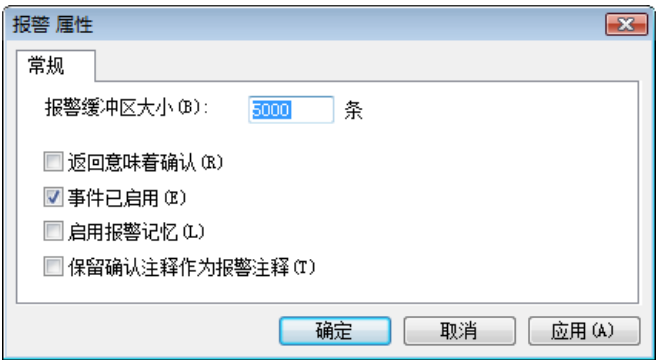
almAckAll(), almAckGroup() almAckTag(), almAckDisplay(), almAckRecent(), almAckPriority(). almAckSelect(), almAckSelectedGroup(), almAckSelectedPriority(), almAckSelectedTag()

## 标记值返回到正常时使用自动确认

InTouch HMI 可以在报警标记的值返回到正常时自动确认报警。此选项不应用于扩展的摘要报警。

### 要配置返回时的报警确认

- 1 在特别菜单上，指向**配置**，然后单击**报警**。此时出现**报警属性**对话框。



- 2 选择**返回意味着确认**复选框，以使 InTouch HMI 自动确认其值返回到正常状态 (RTN) 的报警。
- 3 单击**确定**。

## 使用报警客户端确认报警

操作员可以从 WindowViewer 中确认出现在 Alarm Viewer 控件中的报警。

在显示对象中，**状态**列显示所选报警记录的当前确认状态。此外，记录的文本也采用特定的颜色，以指出其确认状态。

时间 /	状态	分类	类型	优先级	名称	组
03/26/2007 02:59:50 下午	UNACK	VALUE	HI	1	ReactLevel	Reactor
03/26/2007 02:59:50 下午	UNACK	VALUE	HI	1	ProdLevel	Reactor
03/26	UNACK	VALUE	HI	1	ReactTemp	Reactor

确认已选项

确认其它

抑制已选项

抑制其它

查询收藏夹...

统计...

抑制...

冻结

重新查询

排序...

正在显示第 1 到第 3 个报警，共有 默认查询 100 % 完成

此时确认的报警从显示对象中删除。

### 要确认所有报警

- 1 在显示对象中单击鼠标右键，指向**确认其它**，然后单击适当的命令：
  - 单击**确认全部**以确认所有当前报警。
  - 单击**确认可见项**以确认显示对象中所有的可见报警。此时出现**确认注释**对话框。
- 2 输入可选确认注释，然后单击**确定**。

### 要确认所选报警

- 1 选择一个或多个报警。
- 2 单击鼠标右键，然后单击**确认已选项**。此时出现**确认注释**对话框。
- 3 输入可选确认注释，然后单击**确定**。

### 要按组、标记或优先级确认报警

- 1 选择报警。
- 2 在显示对象中单击鼠标右键，指向**确认其它**，然后单击适当的命令：
  - 单击**确认已选组**，以确认属于所选报警组的所有报警。
  - 单击**确认已选标记**，以确认其所有标记的名称与所选报警的标记名相同的报警。
  - 单击**确认已选优先级**，以确认与所选报警具有相同的一个或多个优先级的所有报警。此时出现**确认注释**对话框。
- 3 输入可选确认注释，然后单击**确定**。

## 使用报警与确认注释

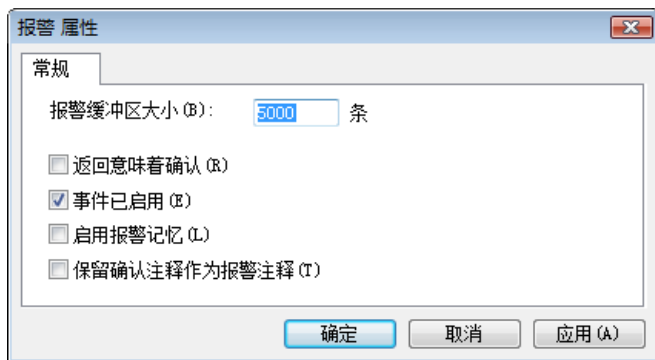
注释有两种类型：报警注释与确认注释。

- 报警注释在发生新的报警实例时设置。`.AlarmComment` 点域用作报警注释，并可以在 InTouch 脚本中进行设置或读取。您可以在“标记名字典”的标记定义中指定此注释的缺省值。报警注释最多包含 131 个字符。
- 确认注释由操作员在确认报警时提供。

您可以使用确认注释来更新标记数据库中的报警注释。

### 要允许报警确认注释更新 `.AlarmComment` 点域

- 1 在**特别菜单**上，指向**配置**，然后单击**报警**。此时出现**报警属性**对话框。



- 2 选择**保留确认注释作为报警注释**复选框，以使用随报警确认输入的注释来更新标记的 `.AlarmComment` 点域与“标记名字典”。

如果未选择此复选框，确认注释会随同确认的报警一起显示（在数据库、打印件以及显示对象中），但 `.AlarmComment` 不会改变。

- 3 单击**确定**。



# 第 6 章

## 在运行时控制标记与组的报警属性

您可以使用报警点域来动态管理报警条件。许多这样的点域都可以通过使用 I/O、表达式及脚本进行访问。通过 I/O 访问，您可以使用其它 Windows 应用程序（如 Excel 或远程节点上运行的 WindowViewer）来监视并控制特定标记的报警信息。

要访问与标记关联的点域，请使用此语法：

tag.dotfield

例如，如果希望允许在运行时更改 Analog\_tag 标记的 HiHi 报警限，则可以给某个按钮创建一个“模拟 - 用户输入”触动链接，并在该链接的对话框中输入 Analog\_tag.HiHiLimit 作为表达式。在运行期间，操作员只需单击该按钮，并为指定给 Analog\_tag 的 HiHi 报警限输入新的值即可。

下表简要介绍每个报警点域。

点域	描述
.Ack	监视 / 控制标记与报警组的报警确认状态。  .Ack 有一个逆向标记点域 .UnAck。出现未确认的报警时，.UnAck 设置为 1。 .UnAck 点域可以用在动画链接或条件脚本中，以触发任何未确认报警的触发器。
.AckDev	监视 / 控制模拟标记或报警组上活动的偏差报警的报警确认状态。

点域	描述
.AckDsc	监视 / 控制离散标记的当前确认状态。
.AckROC	监视 / 控制标记上活动的变化率报警的报警确认状态。
.AckValue	监视标记上活动的值报警的报警确认状态。
.Alarm	发出存在报警条件的信号。
.AlarmAckModel	<p>监视与标记关联的确认模型，具体如下：</p> <p>0 = 条件（缺省值）</p> <p>1 = 事件</p> <p>2 = 扩展</p> <p>应用于发生报警的离散或模拟标记。只读，但在 WindowMaker 中配置。</p>
.AlarmDev	发出存在偏差报警的信号。
.AlarmDevCount	统计给定的标记或报警组上活动的偏差报警总数。
.AlarmDevDeadband	监视 / 控制主、副偏差报警的偏差死区百分比。
.AlarmDevUnAckCount	统计给定的标记或报警组上未确认的活动偏差报警总数。
.AlarmDisabled	禁用 / 启用事件与报警。应用于发生报警的离散与模拟标记，或是应用于报警组。
.AlarmDsc	指出离散报警条件当前处于活动状态。
.AlarmDscCount	统计给定的标记或报警组上活动的离散报警总数。
.AlarmDscDisabled	指出标记是否可以生成离散报警。
	<p><b>备注</b> 对于离散标记，此点域与 .AlarmDisabled 点域相同。</p>

点域	描述
.AlarmDscEnabled	指出标记是否可以生成离散报警。  <b>备注</b> 对于离散标记, 此点域与 .AlarmEnabled 点域相同。
.AlarmDscInhibitor	返回指定给此标记的离散报警 (如果有) 的约束标记名。
.AlarmDscUnAckCount	统计给定的标记或报警组上未确认的活动离散报警总数。
.AlarmEnabled	禁用 / 启用事件与报警。
.AlarmHiDisabled	禁用 / 启用发生报警的模拟标记的 High 报警限。
.AlarmHiEnabled	禁用 / 启用发生报警的模拟标记的 High 报警限。
.AlarmHiHiDisabled	禁用 / 启用发生报警的模拟标记的 HiHi 报警限。
.AlarmHiHiEnabled	禁用 / 启用发生报警的模拟标记的 HiHi 报警限。
.AlarmHiHiInhibitor	返回 HiHi 报警限的约束标记引用。应用于发生报警的模拟标记。只读, 但可以在 WindowMaker 中配置。
.AlarmHiInhibitor	返回 High 报警限的约束标记引用。应用于发生报警的模拟标记。  只读, 但可以在 WindowMaker 中配置。
.AlarmLoDisabled	禁用 / 启用发生报警的模拟标记的 Low 报警限。
.AlarmLoEnabled	禁用 / 启用发生报警的模拟标记的 Low 报警限。

点域	描述
.AlarmLoInhibitor	<p>返回 Low 报警限的约束标记引用。应用于发生报警的模拟标记。</p> <p>只读，但可以在 WindowMaker 中配置。</p>
.AlarmLoLoDisabled	禁用 / 启用发生报警的模拟标记的 LoLo 报警限。
.AlarmLoLoEnabled	禁用 / 启用发生报警的模拟标记的 LoLo 报警限。
.AlarmLoLoInhibitor	<p>返回 LoLo 报警限的约束标记引用。应用于发生报警的模拟标记。</p> <p>只读，但可以在 WindowMaker 中配置。</p>
.AlarmMajDevDisabled	禁用 / 启用发生报警的模拟标记的“主偏差”报警限。
.AlarmMajDevEnabled	禁用 / 启用发生报警的模拟标记的“主偏差”报警限。
.AlarmMajDevInhibitor	<p>返回“主偏差”报警限的约束标记引用。应用于发生报警的模拟标记。</p> <p>只读，但可以在 WindowMaker 中配置。</p>
.AlarmMinDevDisabled	禁用 / 启用发生报警的模拟标记的“副偏差”报警限。
.AlarmMinDevEnabled	禁用 / 启用发生报警的模拟标记的“副偏差”报警限。
.AlarmMinDevInhibitor	<p>返回“副偏差”报警限的约束标记引用。应用于发生报警的模拟标记。</p> <p>只读，但可以在 WindowMaker 中配置。</p>
.AlarmROC	发出存在变化率报警的信号。
.AlarmROCCount	统计给定的标记或报警组上活动的变化率报警总数。

点域	描述
.AlarmROCDisabled	禁用 / 启用发生报警的模拟标记的变化率报警限。
.AlarmROCEnabled	禁用 / 启用发生报警的模拟标记的变化率报警限。
.AlarmROCIInhibitor	返回变化率报警限的约束标记引用。应用于发生报警的模拟标记。 只读，但可以在 <b>WindowMaker</b> 中配置。
.AlarmROCUnAckCount	统计给定的标记或报警组上未确认的变化率报警总数。
.AlarmTotalCount	统计给定的标记或报警组上活动的报警总数。
.AlarmUnAckCount	统计给定的标记或报警组上未确认的活动报警总数。
.AlarmUserDefNum1	可读写实型（浮点）值，缺省值为 0，且值未设置。应用于发生报警的离散标记、发生报警的模拟标记，或应用于报警组。  <b>备注</b> 此点域的值附加到报警上，但“仅”限于已设置（例如通过脚本或 <b>POKE</b> ）值的情况。
.AlarmUserDefNum1Set	可读写离散值。如果脚本为相应的标记定义了 <b>.AlarmUserDefNum1</b> ，则为 <b>TRUE</b> 。要取消与该标记 <b>.AlarmUserDefNum1</b> 值的关联，请将此点域设置为 <b>FALSE</b> 。缺省值为 <b>FALSE</b> 。

点域	描述
.AlarmUserDefNum2	<p>可读写实型（浮点）值，缺省值为 0，且值未设置。应用于发生报警的离散标记、发生报警的模拟标记，或应用于报警组。</p> <p><b>备注</b> 此点域的值附加到报警上，但“仅限于”已设置（例如通过脚本或 <b>POKE</b>）值的情况。</p>
.AlarmUserDefNum2Set	<p>可读写离散值。如果脚本为相应的标记定义了 .AlarmUserDefNum2，则为 TRUE。要取消与该标记 .AlarmUserDefNum2 值的关联，请将此点域设置为 FALSE。缺省值为 FALSE。</p>
.AlarmUserDefStr	<p>可读写文本字符串，缺省值为 ""，且值未设置。应用于发生报警的离散标记、发生报警的模拟标记，或应用于报警组。</p> <p><b>备注</b> 此点域的值附加到报警上，但“仅限于”已设置（例如通过脚本或 <b>POKE</b>）值的情况。</p>
.AlarmUserDefStrSet	<p>可读写离散值。如果脚本为相应的标记定义了 .AlarmUserDefStr，则为 TRUE。要取消与该标记 .AlarmUserDefStr 值的关联，请将此点域设置为 FALSE。缺省值为 FALSE。</p>
.AlarmValDeadband	<p>监视 / 控制报警值死区的值。</p>
.AlarmValueCount	<p>统计给定的标记名或报警组上活动的值报警总数。</p>
.AlarmValueUnAckCount	<p>统计给定的标记名或报警组上未确认的活动值报警总数。</p>
.DevTarget	<p>监视 / 控制主、副偏差报警的目标。</p>

点域	描述
.HiLimit, .HiHiLimit, .LoLimit, .LoLoLimit	可读写模拟标记名点域，用于监视 / 控制值报警检查的极限。这些点域只对整型与实型标记有效。
.HiStatus, .HiHiStatus, .LoStatus, .LoLoStatus	只读离散点域，确定是否存在指定类型的报警。这些点域只对整型与实型标记有效。
.MajorDevPct	可读写整型点域，用于监视或控制报警检查的主偏差百分比。
.MajorDevStatus	只读离散点域，确定指定的标记名是否存在主偏差报警。
.MinorDevPct	可读写整型点域，用于监视与 / 或控制报警检查的副偏差百分比。
.MinorDevStatus	只读离散点域，用于确定指定的标记名是否存在副偏差报警。
.Name	可读写消息点域，用于显示标记名的实际名称。例如，它可用于确定“组变量”所指向的“报警组”的名称，或 TagID 标记的名称。它也可以写入，以更改“组变量”所指向的“报警组”。
.Normal	只读离散点域，指定的标记名不存在报警时它等于 1。此点域对于“报警组”、“组变量”及常规标记而言是有效的。
.ROCPct	可读写点域，用于监视与 / 或控制报警检查的变化率。
.ROCStatus	只读离散点域，用于确定指定的标记是否存在变化率报警。

## 确定标记或报警组是否处在报警条件中

使用以下点域与系统标记确定正在运行的应用程序中的报警状态。您可以找出是否发生了新报警，标记或报警组处在报警还是正常状态。您也可以找出离散、偏差及变化率报警的状态。

- \$NewAlarm 系统标记
- \$System 系统标记
- .Alarm 点域
- .Normal 点域
- .AlarmDsc 点域
- .AlarmDev 点域
- .AlarmROC 点域
- .LoStatus 点域
- .LoLoStatus 点域
- .HiStatus 点域
- .HiHiStatus 点域
- .MinorDevStatus 点域
- .MajorDevStatus 点域
- .ROCStatus 点域



## \$NewAlarm 系统标记

如果正在运行的应用程序中发生新的报警，则设置为 1。

\$NewAlarm 系统标记不能针对远程报警进行设置。

### 语法

```
$NewAlarm=value;
```

### 数据类型

离散（可读写）

### 有效值

0 或 1

### 附注

将 \$NewAlarm 系统标记关联到应用程序窗口中的动画对象。例如，将该标记关联到某个确认按钮（操作员单击时可以将该标记的值重置为 0 并确认报警）。您也可以将 \$NewAlarm 系统标记链接到 PlaySound 逻辑函数，以便在发生报警时发出有声警告。

### 示例

将某个按钮添加到报警确认窗口，并给它附加在操作员单击按钮时运行的以下动作脚本。

```
Ack $System;  
$NewAlarm=0;  
HideSelf;
```

操作员单击该按钮时，会确认所有报警，\$NewAlarm 系统标记重置为 0，报警确认窗口会隐藏起来。

## \$System 系统标记

缺省报警组。

### 类别

报警

### 用法

```
$System
```

### 附注

缺省条件下，系统将标记指定给这个根报警组。定义的所有报警组都由 \$System 衍生而来。

### 数据类型

系统报警组

### 示例

```
$System.Ack = 1; {Acknowledges All Alarms}
```

## .Alarm 点域

指定的标记或报警组当前不处于报警状态时返回 0。报警发生时，.Alarm 点域返回 1。在报警条件消失之前，它保持为 1。  
.Alarm 点域有一个逆向点域 .Normal。

如果指定的标记是报警组的名称，则只要属于该组的任何标记处于报警状态，.Alarm 点域都返回 1。

### 类别

报警

### 用法

*TagName*.Alarm

### 参数

*TagName*

任何离散、整型、实型、间接离散与模拟标记，或报警组标记。

### 数据类型

离散（只读）

### 有效值

0 或 1

### 示例

以下语句验证 Tag1 是否有关联的活动报警：

```
IF (Tag1.Alarm == 1) THEN
```

如果 PumpStation 报警组内存在活动的报警，则处理此

IF-THEN 语句的主体。

```
IF (PumpStation.Alarm == 1) THEN
```

```
    MyAlarmMessage="The pumping station currently has an  
    ALARM!";
```

```
ENDIF;
```

此点域不链接到 .Ack 或 .UnAck 点域。因此，即使已经确认活动报警，.Alarm 仍等于 1。

## .Normal 点域

指定的标记不处于报警状态时返回 1。报警发生时，.Normal 点域返回 0。.Normal 点域有一个逆向点域 .Alarm。

### 类别

报警

### 语法

*TagName*.Normal

### 参数

*TagName*

任何离散、整型、实型、间接离散与模拟标记，或报警组标记。

### 数据类型

离散（只读）

### 有效值

0 或 1

### 示例

没有任何报警与 Tag1 标记关联时，运行以下 IF-THEN 语句。  
Tag1 有一个或更多活动报警时，运行 "ELSE" 主体。

```
IF (Tag1.Normal==1) THEN
    MyOperatorMessage="Tag1 is OK - No alarms associated
    with it";
ELSE
    MyOperatorMessage="Tag1 has one or more alarms
    active!";
ENDIF;
```

## .AlarmDsc 点域

指出指定的离散标记或报警组是否存在报警条件。缺省值为 0。指定的标记存在离散报警条件时，它设置为 1。在报警条件消失之前，值保持为 1。

如果指定的标记是报警组的名称，则只要该组中的任何标记处于活动离散报警状态，.AlarmDsc 点域都会设置为 1。

### 类别

报警

### 用法

*TagName*.AlarmDsc

### 参数

*TagName*

任何离散标记、间接离散标记，或报警组。

### 数据类型

离散（只读）

### 有效值

0 或 1

### 示例

以下语句验证 Tag1 是否有关联的活动离散报警：

```
IF (Tag1.AlarmDsc == 1) THEN
    MyAlarmMessage="The pumping station currently
    has an ALARM!";
ENDIF;
```

此点域不链接到 .Ack 或 .UnAck 点域。因此，即使已经确认活动报警，.AlarmDsc 点域仍等于 1。

### 另请参阅

.Ack, .UnAck, .Alarm, .AlarmDsc, .AckDsc

## **.AlarmDev 点域**

指出指定的标记或报警组的偏差报警何时变为活动状态。缺省值为 0。指定的标记存在偏差报警条件时，它设置为 1。在报警条件消失之前，值保持为 1。

如果指定的标记是报警组的名称，则只要该组中的任何标记处于活动报警状态，.AlarmDev 点域都会设置为 1。

### **类别**

报警

### **用法**

*TagName*.AlarmDev

### **参数**

*TagName*

任何整型、实型、间接模拟标记，或报警组。

### **数据类型**

离散（只读）

### **有效值**

0 或 1

### **示例**

以下语句验证 Tag1 是否有关联的活动偏差报警：

```
IF (Tag1.AlarmDev == 1) THEN
```

如果 PumpStation 报警组内存在活动的偏差报警，则处理此 IF-THEN 语句的主体。

```
IF (PumpStation.AlarmDev == 1) THEN  
    MyAlarmMessage="The pumping station currently  
    has an ALARM!";  
ENDIF;
```

此点域不链接到 .Ack 或 .UnAck 点域。因此，即使已经确认某个活动的报警，此点域仍等于 1。

### **另请参阅**

.Ack, .UnAck, .Alarm, .AckDev

## .AlarmROC 点域

指出指定的标记或报警组的变化率报警条件何时变为活动状态。缺省值为 0。指定的标记存在变化率报警条件时，它设置为 1。在变化率报警条件消失之前，值保持为 1。

如果指定的标记是报警组的名称，则只要该组中的任何标记处于变化率报警状态，.AlarmROC 点域都会设置为 1。

### 类别

报警

### 用法

*TagName*.AlarmROC

### 参数

*TagName*

任何整型、实型、间接模拟标记，或报警组。

### 数据类型

离散（只读）

### 有效值

0 或 1

### 示例

以下语句验证 Tag1 是否有关联的活动变化率报警：

```
IF (Tag1.AlarmROC == 1) THEN
```

如果 PumpStation 报警组内存在活动的变化率报警，则处理此 IF-THEN 语句的主体。

```
IF (PumpStation.AlarmROC == 1) THEN
    MyAlarmMessage="The pumping station currently
    has an ALARM!";
ENDIF;
```

此点域不链接到 .Ack 或 .UnAck 点域。因此，即使已经确认活动的变化率报警，此点域仍等于 1。

### 另请参阅

.Ack, .AckROC, .Alarm, .AlarmROCEnabled,  
.AlarmROCDisabled

## .LoStatus 点域

指出指定的标记或报警组的 Low 报警条件何时变为活动状态。缺省值为 0。指定的标记存在 Low 报警条件时，它设置为 1。在 Low 报警条件消失之前，值保持为 1。

此点域常常同 .Alarm 与 .Ack 点域结合使用，以确定特定标记的特定报警状态。

### 类别

报警

### 用法

*TagName*.LoStatus

### 参数

*TagName*

任何整型、实型或间接模拟标记。

### 数据类型

离散（只读）

### 有效值

0 或 1

### 示例

MyTag 标记的 .LoStatus（Low 报警条件）等于 1 时，运行以下 IF-THEN 语句。

```
IF (MyTag.LoStatus == 1) THEN
    OperatorMessage="MyTag has gone into Low
    Alarm";
ENDIF;
```

### 另请参阅

.Alarm, .AlarmValue, .Ack, .LoLimit, .LoSet,  
.AlarmDisabled, .AlarmEnabled, .AlarmLoDisabled,  
.AlarmLoEnabled, .AlarmLoInhibitor

## .LoLoStatus 点域

指出指定的标记或报警组的 LoLo 报警条件何时变为活动状态。缺省值为 0。指定的标记存在 LoLo 报警条件时，它设置为 1。在 LoLo 报警条件消失之前，值保持为 1。

此点域常常同 .Alarm 与 .Ack 点域结合使用，以确定系统内特定标记的报警状态的确切性质。

### 类别

报警

### 用法

*TagName*.LoLoStatus

### 参数

*TagName*

任何整型、实型或间接模拟标记。

### 数据类型

离散（只读）

### 有效值

0 或 1

### 示例

MyTag 标记的 .LoLoStatus（LoLo 报警限）等于 1 时，运行以下 IF-THEN 语句。

```
IF (MyTag.LoLoStatus == 1) THEN
    OperatorMessage="MyTag has gone into LoLo
    Alarm";
ENDIF;
```

### 另请参阅

.Alarm, .AlarmValue, .Ack, .LoLoLimit, .LoLoSet,  
.AlarmDisabled, .AlarmEnabled, .AlarmLoLoDisabled,  
.AlarmLoLoEnabled, .AlarmLoLoInhibitor



## .HiStatus 点域

指出指定的标记或报警组的 **High** 报警条件何时变为活动状态。缺省值为 0。指定的标记存在 **High** 报警条件时，它设置为 1。在 **High** 报警条件消失之前，值保持为 1。

此点域常常同 **.Alarm** 与 **.Ack** 点域结合使用，以确定标记的特定报警状态。

### 类别

报警

### 用法

*TagName*.HiStatus

### 参数

*TagName*

任何整型、实型或间接模拟标记。

### 数据类型

离散（只读）

### 有效值

0 或 1

### 示例

如果 **MotorAmps** 标记进入 **High** 报警限报警状态，则此脚本会调用另一个脚本，以停止泵浦马达输出。

```
IF (MotorAmps.HiStatus == 1) THEN
    CALL PumpShutdown( );
ENDIF;
```

### 另请参阅

**.Alarm**, **.AlarmValue**, **.Ack**, **.HiLimit**, **.HiSet**,  
**.AlarmDisabled**, **.AlarmEnabled**, **.AlarmHiDisabled**,  
**.AlarmHiEnabled**, **.AlarmHiInhibitor**

## .HiHiStatus 点域

指出指定的标记或报警组的 **HiHi** 报警条件何时变为活动状态。缺省值为 0。指定的标记存在 **HiHi** 报警条件时，它设置为 1。在 **HiHi** 报警条件消失之前，值保持为 1。

此点域常常同 **.Alarm** 与 **.Ack** 点域结合使用，以确定标记的特定报警状态。

### 类别

报警

### 用法

*TagName*.HiHiStatus

### 参数

*TagName*

任何整型、实型或间接模拟标记。

### 数据类型

离散（只读）

### 有效值

0 或 1

### 示例

**MyTag** 标记的 **.HiHiStatus**（**HiHi** 报警）为 1 时，运行以下 **IF-THEN** 语句。

```
IF (MyTag.HiHiStatus == 1) THEN
    OperatorMessage="MyTag has gone into HiHi
    Alarm";
ENDIF;
```

### 另请参阅

**.Alarm**, **.AlarmValue**, **.Ack**, **.HiHiLimit**, **.HiHiSet**,  
**.AlarmDisabled**, **.AlarmEnabled**, **.AlarmHiHiDisabled**,  
**.AlarmHiHiEnabled**, **.AlarmHiHiInhibitor**

## **.MinorDevStatus 点域**

指出指定的标记或报警组的副偏差报警何时变为活动状态。缺省值为 0。指定的标记存在副偏差报警条件时，它设置为 1。在副偏差报警条件消失之前，值保持为 1。

此点域常常同 .Alarm 与 .Ack 点域结合使用，以确定标记的特定报警状态。

### **类别**

报警

### **用法**

*TagName.MinorDevStatus*

### **参数**

*TagName*

任何整型、实型或间接模拟标记。

### **数据类型**

离散（只读）

### **有效值**

0 或 1

### **示例**

MyTag 标记的 .MinorDevStatus（副偏差报警）等于 1 时，运行以下 IF-THEN 语句。

```
IF (MyTag.MinorDevStatus == 1) THEN
    OperatorMessage="MyTag has gone into a Minor
    Deviation Alarm";
ENDIF;
```

### **另请参阅**

.AckDev, .AlarmDev, .AlarmMinDevDisabled,  
.AlarmMinDevEnabled, .AlarmMinDevInhibitor,  
.MinorDevPct, .MajorDevStatus

## .MajorDevStatus 点域

指出指定的标记或报警组的主偏差报警何时变为活动状态。缺省值为 0。存在主偏差报警条件时，指定的点域设置为 1。在主偏差报警条件消失之前，值保持为 1。

此点域常常同 .Alarm 与 .Ack 点域结合使用，以确定标记的特定报警状态。

### 类别

报警

### 用法

*TagName*.MajorDevStatus

### 参数

*TagName*

任何整型、实型或间接模拟标记。

### 数据类型

离散（只读）

### 有效值

0 或 1

### 示例

MyTag 标记的 .MajorDevStatus（主偏差报警）等于 1 时，运行以下 IF-THEN 语句。

```
IF (MyTag.MajorDevStatus == 1) THEN
    OperatorMessage="MyTag has gone into a Major
    Deviation Alarm";
ENDIF;
```

### 另请参阅

.AckDev, .AlarmDev, .AlarmMajDevDisabled,  
.AlarmMajDevEnabled, .AlarmMajDevInhibitor,  
.MajorDevPct, .MajorDevSet, .MinorDevStatus

## .ROCStatus 点域

指出指定的标记或报警组的变化率报警何时变为活动状态。缺省值为 0。指定的标记存在变化率报警条件时，它设置为 1。在变化率报警条件消失之前，值保持为 1。

此点域常常同 .Alarm 与 .Ack 点域结合使用，以确定标记的特定报警状态。

### 类别

报警

### 用法

*TagName*.ROCStatus

### 参数

*TagName*

任何整型、实型或间接模拟标记。

### 数据类型

离散（只读）

### 有效值

0 或 1

### 示例

MyTag 标记的 .ROCStatus（变化率报警）等于 1 时，运行以下 IF-THEN 语句。

```
IF (MyTag.ROCStatus == 1) THEN
    OperatorMessage="MyTag has gone into a
    Rate-Of-Change alarm";
ENDIF;
```

### 另请参阅

.ROCPct, .ROCSet

## 确定是否给标记设置了报警限

以下点域指出在应用程序运行期间是否给标记设置了报警限。

- .LoLoSet 点域
- .LoSet 点域
- .HiSet 点域
- .HiHiSet 点域
- .MinorDevSet 点域
- .MajorDevSet 点域
- .ROCSet 点域

### .LoLoSet 点域

指出是否已经给整型或实型标记设置了 LoLo 报警限。

**类别**

报警

**用法**

*TagName*.LoLoSet

**参数**

*TagName*

任何整型、实型或间接模拟标记。

**数据类型**

离散（只读）

**有效值**

0 或 1

**示例**

如果给 MyTag 标记设置了 LoLo 报警限，则执行 THEN 代码块：

```
IF (MyTag.LoLoSet== 1) THEN
    MsgTag="LoLo alarm limit has been set for
    MyTag";
ENDIF;
```

**另请参阅**

.Alarm, .AlarmValue, .Ack, .LoLoStatus, .LoLoLimit,  
.AlarmDisabled, .AlarmEnabled, .AlarmLoLoDisabled,  
.AlarmLoLoEnabled, .AlarmLoLoInhibitor

## .LoSet 点域

指出是否已经给整型或实型标记设置了 Low 报警限。

类别

报警

用法

*TagName*.LoSet

参数

*TagName*

任何整型、实型或间接模拟标记。

数据类型

离散（只读）

有效值

0 或 1

示例

如果给 MyTag 标记设置了 Low 报警限，则执行 THEN 代码块：

```
IF (MyTag.LoSet== 1) THEN
    MsgTag="Low alarm limit has been set for
    MyTag";
ENDIF;
```

另请参阅

.Alarm, .AlarmValue, .Ack, .LoStatus, .LoLimit,  
.AlarmDisabled, .AlarmEnabled, .AlarmLoDisabled,  
.AlarmLoEnabled, .AlarmLoInhibitor

## .HiSet 点域

指出是否已经给整型或实型标记设置了 High 报警限。

类别

报警

用法

*TagName*.HiSet

参数

*TagName*

任何整型、实型或间接模拟标记。

数据类型

离散（只读）

有效值

0 或 1

**示例**

如果给 MyTag 标记设置了 High 报警限，则执行 THEN 代码块：

```
IF (MyTag.HiSet== 1) THEN
    MsgTag="High alarm limit has been set for
    MyTag";
ENDIF;
```

**另请参阅**

.Alarm, .AlarmValue, .Ack, .HiHiStatus, .HiHiLimit,  
.AlarmDisabled, .AlarmEnabled, .AlarmHiHiDisabled,  
.AlarmHiHiEnabled, .AlarmHiHiInhibitor

## .HiHiSet 点域

指出是否已经给整型或实型标记设置了 HiHi 报警限。

**类别**

报警

**用法**

*TagName*.HiHiSet

**参数**

*TagName*

任何整型、实型或间接模拟标记。

**数据类型**

离散（只读）

**有效值**

0 或 1

**示例**

如果给 MyTag 标记设置了 HiHi 报警限，则执行 THEN 代码块：

```
IF (MyTag.HiHiSet== 1) THEN
    MsgTag="HiHi alarm limit has been set for
    MyTag";
ENDIF;
```

**另请参阅**

.Alarm, .AlarmValue, .Ack, .HiHiStatus, .HiHiLimit,  
.AlarmDisabled, .AlarmEnabled, .AlarmHiHiDisabled,  
.AlarmHiHiEnabled, .AlarmHiHiInhibitor



## **.MinorDevSet** 点域

指出是否已经给整型或实型标记设置了副偏差报警限。

**类别**

报警

**用法**

*TagName*.MinorDevSet

**参数**

*TagName*

任何整型、实型或间接模拟标记。

**数据类型**

离散（只读）

**有效值**

0 或 1

**示例**

如果给 MyTag 标记设置了副偏差百分比报警限，则执行 THEN 代码块：

```
IF (MyTag.MinorDevSet== 1) THEN
    MsgTag="Minor deviation alarm limit has been
    set for MyTag";
ENDIF;
```

**另请参阅**

.AckDev, .AlarmDev, .AlarmMinDevDisabled,  
.AlarmMinDevEnabled, .AlarmMinDevInhibitor,  
.MinorDevPct, .MinorDevStatus

## **.MajorDevSet 点域**

指出是否已经给整型或实型标记设置了主偏差报警限。

**类别**

报警

**用法**

*TagName*.MajorDevSet

**参数**

*TagName*

任何整型、实型或间接模拟标记。

**数据类型**

离散（只读）

**有效值**

0 或 1

**示例**

如果给 **MyTag** 标记设置了主偏差百分比报警限，则执行 **THEN** 代码块：

```
IF (MyTag.MajorDevSet== 1) THEN
    MsgTag="Major deviation alarm limit has been
    set for MyTag";
ENDIF;
```

**另请参阅**

.AckDev, .AlarmDev, .AlarmMajDevDisabled,  
.AlarmMajDevEnabled, .AlarmMajDevInhibitor,  
.MajorDevPct, .MajorDevStatus

## .ROCSet 点域

指出是否已经给整型或实型标记设置了变化率报警限。

**类别**

报警

**用法**

*TagName*.ROCSet

**参数**

*TagName*

任何整型、实型或间接模拟标记。

**数据类型**

离散（只读）

**有效值**

0 或 1

**示例**

如果给 MyTag 标记设置了变化率报警限，则执行 THEN 代码块：

```
IF (MyTag.ROCSet == 1) THEN
    MsgTag="Rate-of-change alarm limit has been
    set for MyTag";
ENDIF;
```

**另请参阅**

.Alarm, .Ack, .LoLimit, .LoLoLimit, .HiHiLimit, .HiLimit,  
.HiSet, .LoSet, .LoLoSet, .HiStatus, .HiHiStatus, .ROCPct,  
.ROCStatus

## 启用与禁用标记或报警组的报警

InTouch HMI 提供一组点域，可以在应用程序运行期间启用或禁用给标记设置的报警。

### 启用 / 禁用所有报警

.AlarmEnabled 与 .AlarmDisabled 点域可以根据标记或报警组各自的值来启用或禁用它们的报警。与这两个点域关联的报警状态互为反向。对于 .AlarmEnabled，值为 1 时给标记或报警组启用报警。 .AlarmDisabled 的值为 1 时给标记或报警组禁用报警。

其中任一个点域给报警组启用报警时，都会启用属于该组的所有标记的报警。其中任一个点域禁用报警时，会忽略所有的事件与报警。这些报警不存储在报警内存中，也不写入磁盘。

#### .AlarmEnabled 点域

启用或禁用标记或报警组的报警。

##### 类别

报警

##### 用法

*TagName*.AlarmEnabled

##### 参数

*TagName*

任何离散、整型、实型、间接离散、间接模拟标记，或报警组。

##### 附注

.AlarmEnabled 设置为 0 时，会忽略所有的事件与报警。它们不存储在报警内存中，也不写入磁盘。因此应尽可能重新激活事件 / 报警，以免丢失数据，这点非常重要。

指定的标记是一个报警组时，指定的报警组中与标记关联的所有报警都会启用。

##### 数据类型

离散（可读写）

##### 有效值

0 = 禁用报警

1 = 启用报警（缺省）

##### 示例

下例禁用 Tag1 标记的报警：

```
Tag1.AlarmEnabled=0;
```

##### 另请参阅

.AlarmDisabled

## **.AlarmDisabled 点域**

启用或禁用标记或报警组的报警。

### **类别**

报警

### **用法**

*TagName*.AlarmDisabled

### **参数**

*TagName*

任何离散、整型、实型、间接离散、间接模拟标记，或报警组。

### **附注**

.AlarmDisabled 设置为 1 时，会忽略所有的事件与报警。它们不存储在报警内存中，也不写入磁盘。因此应尽可能重新激活事件 / 报警，以免丢失数据，这点非常重要。

指定的标记是一个报警组时，指定的报警组中与标记关联的所有报警都会禁用。

这与 .AlarmEnabled 点域正好相反。

### **示例**

下例启用 Tag1 标记的报警。

```
Tag1.AlarmDisabled=0;
```

### **另请参阅**

.AlarmEnabled

## 启用 / 禁用 LoLo 报警

.AlarmLoLoEnabled 与 .AlarmLoLoDisabled 点域可以根据标记或报警组各自的值来启用或禁用它们的 LoLo 报警。与这两个点域关联的 LoLo 报警状态互为反向。对于

.AlarmLoLoEnabled，值为 1 时启用标记或报警组的 LoLo 报警。 .AlarmLoLoDisabled 的值为 1 时禁用标记或报警组的 LoLo 报警。

其中任一个点域启用报警组的 LoLo 报警时，都会启用属于该组的所有标记的 LoLo 报警。其中任一个点域禁用 LoLo 报警时，会忽略所有的 LoLo 报警。这些报警不存储在报警内存中，也不写入磁盘。

### .AlarmLoLoEnabled 点域

启用或禁用 LoLo 条件的事件与报警。

类别  
报警

用法

*TagName*.AlarmLoLoEnabled

参数

*TagName*

任何整型、实型、间接模拟标记，或报警组。

附注

.AlarmLoLoEnabled 设置为 0 时，会忽略所有 LoLo 条件的事件与报警。它们不存储在报警内存中，也不写入磁盘。因此应尽可能重新激活事件 / 报警，以免丢失数据，这点非常重要。

数据类型

离散（可读写）

有效值

0 = 禁用报警

1 = 启用报警（缺省）

示例

下例禁用 Tag1 标记的 LoLo 报警：

```
Tag1.AlarmLoLoEnabled=0;
```

另请参阅

.AlarmDisabled, .AlarmEnabled, .AlarmLoLoDisabled

## **.AlarmLoLoDisabled 点域**

启用或禁用 LoLo 条件的事件与报警。

### **类别**

报警

### **用法**

*TagName*.AlarmLoLoDisabled

### **参数**

*TagName*

任何整型、实型、间接模拟标记，或报警组。

### **附注**

.AlarmLoLoDisabled 设置为 1 时，会忽略所有 LoLo 条件的事件与报警。它们不存储在报警内存中，也不写入磁盘。因此应尽可能重新启用事件 / 报警，以免丢失数据，这点非常重要。

### **数据类型**

离散（可读写）

### **有效值**

1 = 禁用报警

0 = 启用报警（缺省）

### **示例**

下例启用 Tag2 标记的 LoLo 报警：

```
Tag2.AlarmLoLoDisabled=0;
```

### **另请参阅**

.AlarmDisabled, .AlarmEnabled, .AlarmLoLoEnabled

## 启用 / 禁用 Low 报警

.AlarmLoEnabled 与 .AlarmLoDisabled 点域可以根据标记或报警组各自的值来启用或禁用它们的 Low 报警。与这两个点域关联的 Low 报警状态互为反向。对于 .AlarmLoEnabled, 值为 1 时启用标记或报警组的 Low 报警。.AlarmLoDisabled 的值为 1 时禁用标记或报警组的 Low 报警。

其中任一个点域启用报警组的 Low 报警时, 都会启用属于该组的所有标记的 Low 报警。其中任一个点域禁用 Low 报警时, 会忽略所有的 Low 报警。这些报警不存储在报警内存中, 也不写入磁盘。

### .AlarmLoEnabled 点域

启用或禁用 Low 条件的事件与报警。

类别

报警

用法

*TagName*.AlarmLoEnabled

参数

*TagName*

任何整型、实型、间接模拟标记, 或报警组。

附注

.AlarmLoEnabled 设置为 0 时, 会忽略所有 Low 条件的事件与报警。它们不存储在报警内存中, 也不写入磁盘。因此应尽可能重新启用事件 / 报警, 以免丢失数据, 这点非常重要。

数据类型

离散 (可读写)

有效值

0 = 禁用报警

1 = 启用报警 (缺省)

示例

下例禁用 Tag1 标记的 Low 报警:

```
Tag1.AlarmLoEnabled=0;
```

另请参阅

.AlarmDisabled, .AlarmEnabled, .AlarmLoDisabled



## **.AlarmLoDisabled 点域**

启用或禁用 Low 条件的事件与报警。

### **类别**

报警

### **用法**

*TagName*.AlarmLoDisabled

### **参数**

*TagName*

任何整型、实型、间接模拟标记，或报警组。

### **附注**

.AlarmLoDisabled 设置为 1 时，会忽略所有 Low 条件的事件与报警。它们不存储在报警内存中，也不写入磁盘。因此应尽可能重新启用事件 / 报警，以免丢失数据，这点非常重要。

### **数据类型**

离散（可读写）

### **有效值**

1 = 禁用报警

0 = 启用报警（缺省）

### **示例**

下例启用 Tag2 标记的 Low 报警：

```
Tag2.AlarmLoDisabled=0;
```

### **另请参阅**

.AlarmDisabled, .AlarmEnabled, .AlarmLoEnabled

## 启用 / 禁用 High 报警

.AlarmHiEnabled 与 .AlarmHiDisabled 点域可以根据标记或报警组各自的值来启用或禁用它们的 High 报警。与这两个点域关联的 High 报警状态互为反向。对于 .AlarmHiEnabled, 值为 1 时启用标记或报警组的 High 报警。 .AlarmHiDisabled 的值为 1 时禁用标记或报警组的 High 报警。

其中任一个点域启用报警组的 High 报警时, 都会启用属于该组的所有标记的 High 报警。其中任一个点域禁用 High 报警时, 会忽略所有的 High 报警。 High 报警不存储在报警内存中, 也不写入磁盘。

### .AlarmHiEnabled 点域

启用或禁用 High 条件的事件与报警。

类别

报警

用法

*TagName*.AlarmHiEnabled

参数

*TagName*

任何整型、实型、间接模拟标记, 或报警组。

附注

.AlarmHiEnabled 设置为 0 时, 会忽略所有 High 条件的事件与报警。它们不存储在报警内存中, 也不写入磁盘。因此应尽可能重新激活事件 / 报警, 以免丢失数据, 这点非常重要。

这与 .AlarmHiDisabled 点域正好相反。

数据类型

离散 (可读写)

有效值

0 = 禁用报警

1 = 启用报警 (缺省)

示例

下例禁用标记 Tag1 的 High 报警:

```
Tag1.AlarmHiEnabled=0;
```

另请参阅

.AlarmHiDisabled, .AlarmEnabled

## **.AlarmHiDisabled 点域**

启用或禁用 High 条件的事件与报警。

**类别**

报警

**用法**

*TagName*.AlarmHiDisabled

**参数**

*TagName*

任何整型、实型、间接模拟标记，或报警组。

**附注**

.AlarmHiDisabled 设置为 1 时，会忽略所有 High 条件的事件与报警。它们不存储在报警内存中，也不写入磁盘。因此应尽可能重新激活事件 / 报警，以免丢失数据，这点非常重要。

这与 .AlarmHiEnabled 点域正好相反。

**数据类型**

离散（可读写）

**有效值**

1 = 禁用报警

0 = 启用报警（缺省）

**示例**

下例启用 Tag2 标记的 High 报警：

```
Tag2.AlarmHiDisabled=0;
```

**另请参阅**

.AlarmHiEnabled, .AlarmDisabled

## 启用 / 禁用 HiHi 报警

.AlarmHiHiEnabled 与 .AlarmHiHiDisabled 点域可以根据标记或报警组各自的值来启用或禁用它们的 HiHi 报警。与这两个点域关联的 HiHi 报警状态互为反向。对于

.AlarmHiHiEnabled，值为 1 时启用标记或报警组的 HiHi 报警。 .AlarmHiHiDisabled 的值为 1 时禁用标记或报警组的 HiHi 报警。

其中任一个点域启用报警组的 HiHi 报警时，都会启用属于该组的所有标记的 HiHi 报警。其中任一个点域禁用 HiHi 报警时，会忽略所有的 HiHi 报警。这些 HiHi 报警不存储在报警内存中，也不写入磁盘。

### .AlarmHiHiEnabled 点域

启用与 / 或禁用 HiHi 条件的事件与报警。

类别  
报警

用法

*TagName*.AlarmHiHiEnabled

参数

*TagName*

任何整型、实型、间接模拟标记，或报警组。

附注

.AlarmHiHiEnabled 设置为 0 时，会忽略所有 HiHi 条件的事件与报警。它们不存储在报警内存中，也不写入磁盘。因此应尽可能重新启用事件 / 报警，以免丢失数据，这点非常重要。

数据类型

离散（可读写）

有效值

0 = 禁用报警

1 = 启用报警（缺省）

示例

下例禁用 Tag1 标记的 HiHi 报警：

```
Tag1.AlarmHiHiEnabled=0;
```

另请参阅

.AlarmHiHiDisabled, .AlarmEnabled

## **.AlarmHiHiDisabled 点域**

启用或禁用 HiHi 条件的事件与报警。

### **类别**

报警

### **用法**

*TagName*.AlarmHiHiDisabled

### **参数**

*TagName*

任何整型、实型、间接模拟标记，或报警组。

### **附注**

.AlarmHiHiDisabled 设置为 1 时，会忽略所有 HiHi 条件的事件与报警。它们不存储在报警内存中，也不写入磁盘。因此应尽可能重新激活事件 / 报警，以免丢失数据，这点非常重要。

这与 .AlarmHiHiEnabled 点域正好相反。

### **数据类型**

离散（可读写）

### **有效值**

1 = 禁用报警

0 = 启用报警（缺省）

### **示例**

下例启用 Tag2 标记的 HiHi 报警：

```
Tag2.AlarmHiHiDisabled=0;
```

### **另请参阅**

.AlarmHiHiEnabled, .AlarmDisabled

## 启用 / 禁用离散报警

.AlarmDscEnabled 与 .AlarmDscDisabled 点域可以根据标记或报警组各自的值来启用或禁用它们的离散报警。与这两个点域关联的离散报警状态互为反向。对于 .AlarmDscEnabled，值为 1 时启用标记或报警组的离散报警。.AlarmDscDisabled 的值为 1 时禁用标记或报警组的离散报警。

其中任一个点域启用报警组的离散报警时，都会启用属于该组的所有标记的离散报警。其中任一个点域禁用离散报警时，会忽略所有的离散报警。这些离散报警不存储在报警内存中，也不写入磁盘。

### .AlarmDscEnabled 点域

指出标记是否可以生成离散报警。

#### 类别

报警

#### 用法

*TagName*.AlarmDscEnabled

#### 参数

*TagName*

任何离散标记、间接离散标记，或报警组。

#### 附注

.AlarmDscEnabled 设置为 0 时，会忽略所有离散条件的事件与报警。它们不存储在报警内存中，也不写入磁盘。因此应尽可能重新激活事件 / 报警，以免丢失数据，这点非常重要。

这与 .AlarmDscDisabled 点域正好相反。

#### 数据类型

离散（可读写）

#### 有效值

0 = 禁用报警

1 = 启用报警（缺省）

#### 示例

下例禁用 Tag1 标记的离散报警：

```
Tag1.AlarmDscEnabled=0;
```

#### 另请参阅

.AlarmDscDisabled

## **.AlarmDscDisabled 点域**

指出标记是否可以生成离散报警。

### **类别**

报警

### **用法**

*TagName*.AlarmDscDisabled

### **参数**

*TagName*

任何离散、间接离散标记，或报警组。

### **附注**

.AlarmDscDisabled 设置为 1 时，会忽略所有离散条件的事件与报警。它们不存储在报警内存中，也不写入磁盘。因此应尽可能重新激活事件 / 报警，以免丢失数据，这点非常重要。

这与 .AlarmDscEnabled 点域正好相反。

### **数据类型**

离散（可读写）

### **有效值**

1 = 禁用报警

0 = 启用报警（缺省）

### **示例**

下例启用 Tag2 标记的离散报警：

```
Tag2.AlarmDscDisabled=0;
```

## 启用 / 禁用副偏差报警

.AlarmMinDevEnabled 与 .AlarmMinDevDisabled 点域可以根据标记或报警组各自的值来启用或禁用它们的副偏差报警。与这两个点域关联的副偏差报警状态互为反向。对于

.AlarmMinDevEnabled, 值为 1 时启用标记或报警组的副偏差报警。 .AlarmMinDevDisabled 的值为 1 时禁用标记或报警组的副偏差报警。

其中任一个点域启用报警组的副偏差报警时, 都会启用属于该组的所有标记的副偏差报警。其中任一个点域禁用副偏差报警时, 会忽略所有的副偏差报警。这些副偏差报警不存储在报警内存中, 也不写入磁盘。

### .AlarmMinDevEnabled 点域

启用或禁用副偏差事件与报警。

#### 类别

报警

#### 用法

*TagName*.AlarmMinDevEnabled

#### 参数

*TagName*

任何整型、实型、间接模拟标记, 或报警组。

#### 附注

.AlarmMinDevEnabled 设置为 0 时, 会忽略所有的副偏差事件与报警。它们不存储在报警内存中, 也不写入磁盘。因此应尽可能重新启用事件 / 报警, 以免丢失数据, 这点非常重要。

#### 数据类型

离散 (可读写)

#### 有效值

0 = 禁用报警

1 = 启用报警 (缺省)

#### 示例

下例禁用 Tag1 标记的副偏差报警:

```
Tag1.AlarmMinDevEnabled=0;
```

#### 另请参阅

.AlarmDisabled, .AlarmEnabled, .AlarmMinDevDisabled



## **.AlarmMinDevDisabled 点域**

启用或禁用副偏差事件与报警。

### **类别**

报警

### **用法**

`TagName.AlarmMinDevDisabled`

### **参数**

*TagName*

任何整型、实型、间接模拟标记，或报警组。

### **附注**

.AlarmMinDevDisabled 设置为 1 时，会忽略所有的副偏差事件与报警。它们不存储在报警内存中，也不写入磁盘。因此应尽可能重新激活事件 / 报警，以免丢失数据，这点非常重要。

这与 .AlarmMinDevEnabled 点域正好相反。

### **数据类型**

离散（可读写）

### **有效值**

1 = 禁用报警

0 = 启用报警（缺省）

### **示例**

下例启用 Tag2 标记的副偏差报警：

```
Tag2.AlarmMinDevDisabled=0;
```

### **另请参阅**

.AlarmDisabled, .AlarmEnabled, .AlarmMinDevEnabled

## 启用 / 禁用主偏差报警

.AlarmMajDevEnabled 与 .AlarmMajDevDisabled 点域可以根据标记或报警组各自的值来启用或禁用它们的主偏差报警。与这两个点域关联的主偏差报警状态互为反向。对于

.AlarmMajDevEnabled, 值为 1 时启用标记或报警组的主偏差报警。.AlarmMajDevDisabled 的值为 1 时禁用标记或报警组的主偏差报警。

其中任一个点域启用报警组的主偏差报警时, 都会启用属于该组的所有标记的主偏差报警。其中任一个点域禁用主偏差报警时, 会忽略所有的主偏差报警。这些主偏差报警不存储在报警内存中, 也不写入磁盘。

### .AlarmMajDevEnabled 点域

启用或禁用主偏差事件与报警。

**类别**

报警

**用法**

*TagName*.AlarmMajDevEnabled

**参数**

*TagName*

任何整型、实型、间接模拟标记, 或报警组。

**附注**

.AlarmMajDevEnabled 设置为 0 时, 会忽略所有的主偏差事件与报警。它们不存储在报警内存中, 也不写入磁盘。因此应尽可能重新启用事件 / 报警, 以免丢失数据, 这点非常重要。

这与 .AlarmMajDevDisabled 点域正好相反。

**数据类型**

离散 (可读写)

**有效值**

0 = 禁用报警

1 = 启用报警 (缺省)

**示例**

下例禁用 Tag1 标记的主偏差报警:

```
Tag1.AlarmMajDevEnabled=0;
```

**另请参阅**

.AlarmDisabled, .AlarmEnabled, .AlarmMajDevDisabled

## **.AlarmMajDevDisabled 点域**

启用或禁用主偏差事件与报警。

### **类别**

报警

### **用法**

*TagName*.AlarmMajDevDisabled

### **参数**

*TagName*

任何整型、实型、间接模拟标记，或报警组。

### **附注**

.AlarmMajDevDisabled 设置为 1 时，会忽略所有的主偏差事件与报警。它们不存储在报警内存中，也不写入磁盘。因此应尽可能重新激活事件 / 报警，以免丢失数据，这点非常重要。

这与 .AlarmMajDevEnabled 点域正好相反。

### **数据类型**

离散（可读写）

### **有效值**

1 = 禁用报警

0 = 启用报警（缺省）

### **示例**

下例启用 Tag2 标记的主偏差报警：

```
Tag2.AlarmMajDevDisabled=0;
```

### **另请参阅**

.AlarmDisabled, .AlarmEnabled, .AlarmMajDevEnabled

## 启用 / 禁用变化率报警

.AlarmROCEnabled 与 .AlarmROCDisabled 点域可以根据标记或报警组各自的值来启用或禁用它们的变化率报警。与这两个点域关联的变化率报警状态互为反向。对于

.AlarmROCEnabled, 值为 1 时启用标记或报警组的变化率报警。 .AlarmROCDisabled 的值为 1 时禁用标记或报警组的变化率报警。

其中的任一个点域启用报警组的变化率报警时, 都会启用属于该组的所有标记的变化率报警。其中任一个点域禁用变化率报警时, 会忽略所有的变化率报警。这些变化率报警不存储在报警内存中, 也不写入磁盘。

### .AlarmROCEnabled 点域

启用或禁用变化率事件与报警。

#### 类别

报警

#### 用法

*TagName*.AlarmROCEnabled

#### 参数

*TagName*

任何整型、实型、间接模拟标记, 或报警组。

#### 附注

.AlarmROCEnabled 设置为 0 时, 会忽略所有的变化率条件事件与报警。它们不存储在报警内存中, 也不写入磁盘。因此应尽可能重新激活事件 / 报警, 以免丢失数据, 这点非常重要。

这与 .AlarmROCDisabled 点域正好相反。

#### 数据类型

离散 (可读写)

#### 有效值

0 = 禁用报警

1 = 启用报警 (缺省)

#### 示例

下例禁用 Tag1 标记的变化率报警:

```
Tag1.AlarmROCEnabled=0;
```

#### 另请参阅

.AlarmDisabled, .AlarmEnabled, .AlarmROCDisabled

## **.AlarmROCDisabled 点域**

禁用或启用变化率事件与报警。

### **类别**

报警

### **用法**

*TagName*.AlarmROCDisabled

### **参数**

*TagName*

任何整型、实型、间接模拟标记，或报警组。

### **附注**

.AlarmROCDisabled 设置为 1 时，会忽略所有的变化率事件与报警。它们不存储在报警内存中，也不写入磁盘。因此应尽可能重新激活事件 / 报警，以免丢失数据，这点非常重要。

这与 .AlarmROCEnabled 点域正好相反。

### **数据类型**

离散（可读写）

### **有效值**

1= 禁用报警

0= 启用报警（缺省）

### **示例**

下例启用 Tag2 标记的变化率报警：

```
Tag2.AlarmROCDisabled=0;
```

### **另请参阅**

.AlarmDisabled, .AlarmEnabled, .AlarmROCEnabled

## 更改标记的报警限

使用以下点域在应用程序运行期间更改标记的报警限。您可以更改 LoLo、Low、High 及 HiHi 报警的极限，更改主、副偏差的百分比与目标，以及变化率偏差。

- .LoLoLimit 点域
- .LoLimit 点域
- .HiLimit 点域
- .HiHiLimit 点域
- .MinorDevPct 点域
- .MajorDevPct 点域
- .DevTarget 点域
- .ROCPct 点域

### .LoLoLimit 点域

更改标记的 LoLo 报警限。

**类别**

报警

**用法**

*TagName*.LoLoLimit

**参数**

*TagName*

任何整型、实型或间接模拟标记。

**附注**

如果希望在有意或无意关闭 WindowViewer 之后，仍可以继续使用该点域的运行时值，请在“标记名字典”中选择**保留参数**选项。

**数据类型**

模拟（可读写）

**有效值**

必须在为指定的标记配置的值范围内。

**示例**

此语句按 10 递减 MyTag1 标记的 LoLo 报警限：

```
MyTag1.LoLoLimit=MyTag1.LoLoLimit - 10;
```

**另请参阅**

.Alarm, .AlarmValue, .Ack, .LoLoStatus, .LoLoSet,  
.AlarmDisabled, .AlarmEnabled, .AlarmLoLoDisabled,  
.AlarmLoLoEnabled, .AlarmLoLoInhibitor

## .LoLimit 点域

更改标记的 Low 报警限。

**类别**

报警

**用法**

*Tagname*.LoLimit

**参数**

*Tagname*

任何整型、实型或间接模拟标记。

**附注**

如果希望在有意或无意关闭 WindowViewer 之后，仍可以继续使用此点域的运行时值，请在“标记名字典”中选择**保留参数**选项。

**数据类型**

模拟（可读写）

**有效值**

必须在为指定的标记配置的值范围内。

**示例**

此语句按 10 递减 MyTag 标记的 Low 报警限：

```
MyTag.LoLimit=MyTag.LoLimit - 10;
```

**另请参阅**

.Alarm, .AlarmValue, .Ack, .LoStatus, .LoSet,  
.AlarmDisabled, .AlarmEnabled, .AlarmLoDisabled,  
.AlarmLoEnabled, .AlarmLoInhibitor

## .HiLimit 点域

更改标记的 High 报警限。

**类别**

报警

**用法**

*TagName*.HiLimit

**参数**

*TagName*

任何整型、实型或间接模拟标记。

**附注**

如果希望在有意或无意关闭 WindowViewer 之后，仍可以继续使用该点域的运行时值，请在“标记名字典”中选择**保留参数**选项。

**数据类型**

模拟（可读写）

**有效值**

必须在为指定的标记配置的值范围内。

**示例**

此语句将 PumpTemp 标记的 High 报警限设置为 212:

```
PumpTemp.HiLimit = 212;
```

**另请参阅**

.Alarm, .AlarmValue, .Ack, .HiHiStatus, .HiHiSet, .AlarmDisabled, .AlarmEnabled, .AlarmHiHiDisabled, .AlarmHiHiEnabled, .AlarmHiHiInhibitor



## .HiHiLimit 点域

更改标记的 HiHi 报警限。

### 类别

报警

### 用法

*TagName*.HiHiLimit

### 参数

*TagName*

任何整型、实型或间接模拟标记。

### 附注

如果希望在有意或无意关闭 WindowViewer 之后，仍可以继续使用此点域的运行时值，请在“标记名字典”中选择**保留参数**选项。

### 数据类型

模拟（可读写）

### 有效值

必须在为指定的标记配置的值范围内。

### 示例

以下语句按 5 递增 MyTag 标记的 HiHi 报警限：

```
MyTag.HiHiLimit=MyTag.HiHiLimit + 5;
```

### 另请参阅

.Alarm, .AlarmValue, .Ack, .HiHiStatus, .HiHiSet,  
.AlarmDisabled, .AlarmEnabled, .AlarmHiHiDisabled,  
.AlarmHiHiEnabled, .AlarmHiHiInhibitor

## **.MinorDevPct 点域**

更改标记的副偏差报警限。

**类别**

报警

**用法**

*TagName.MinorDevPct*

**参数**

*TagName*

任何整型、实型或间接模拟标记。

**附注**

如果希望在有意或无意关闭 WindowViewer 之后，仍可以继续使用该点域的运行时值，请在“标记名字典”中选择**保留参数**选项。

**数据类型**

实型（可读写）

**有效值**

0 到 100

**示例**

以下语句将 MyTag 标记的副偏差极限属性设置为 25%：

```
MyTag.MinorDevPct=25;
```

**另请参阅**

.AckDev, .AlarmDev, .AlarmMinDevDisabled,  
.AlarmMinDevEnabled, .AlarmMinDevInhibitor,  
.MinorDevSet, .MinorDevStatus

## **.MajorDevPct** 点域

更改标记的主偏差报警限。

**类别**

报警

**用法**

*TagName*.MajorDevPct

**参数**

*TagName*

任何整型、实型或间接模拟标记。

**附注**

如果希望在有意或无意关闭 WindowViewer 之后，仍可以继续使用此点域的运行时值，请在“标记名字典”中选择**保留参数**选项。

**数据类型**

实型（可读写）

**有效值**

0 到 100

**示例**

以下语句将 MyTag 标记的主偏差极限属性设置为 25%：

```
MyTag.MajorDevPct=25;
```

**另请参阅**

.AckDev, .AlarmDev, .AlarmMajDevDisabled,  
.AlarmMajDevEnabled, .AlarmMajDevInhibitor,  
.MajorDevSet, .MajorDevStatus

## .DevTarget 点域

更改标记的主、副偏差报警的目标。

### 类别

报警

### 用法

*TagName*.DevTarget

### 参数

*TagName*

任何整型、实型或间接模拟标记。

### 附注

如果希望在有意或无意关闭 WindowViewer 之后，仍可以继续使用该点域的运行时值，请在“标记名字典”中选择**保留参数**选项。

### 数据类型

实型（可读写）

### 有效值

必须在给标记指定的值范围内。

### 示例

以下语句将 MyTag 标记的偏差目标设置为 500：

```
MyTag.DevTarget=500;
```

### 另请参阅

.AckDev, .AlarmDev, .AlarmMajDevDisabled,  
.AlarmMajDevEnabled, .AlarmMajDevInhibitor,  
.MajorDevSet, .MajorDevStatus

## .ROCPct 点域

更改标记的变化率报警限。

### 类别

报警

### 用法

*TagName*.ROCPct

### 参数

*TagName*

任何整型、实型或间接模拟标记。

### 附注

此点域直接对应于“标记名字典”报警部分所配置的相同字段。

### 数据类型

整型（可读写）

### 有效值

0 到 100

### 示例

以下语句将 MyTag 标记的变化率报警限设置为 25%:

```
MyTag.ROCPct=25;
```

### 另请参阅

.ROCStatus, .ROCSet

## 更改标记的报警死区

使用以下点域在应用程序运行期间更改标记的报警死区范围：

- `.AlarmValDeadband` 点域
- `.AlarmDevDeadband` 点域

### `.AlarmValDeadband` 点域

在 InTouch 应用程序运行期间更改标记的死区值。

#### 类别

报警

#### 用法

`TagName.AlarmValDeadband`

#### 参数

*TagName*

任何整型、实型或间接模拟标记。

#### 附注

如果希望在有意或无意关闭 WindowViewer 之后，仍可以继续  
使用此点域的运行时值，请在“标记名字典”中选择**保留参数**选  
项。

#### 数据类型

模拟（可读写）

#### 有效值

必须在给标记指定的值范围内。

#### 示例

以下语句将 Tag1 标记的死区更改为 25：

```
Tag1.AlarmValDeadband=25;
```

#### 另请参阅

`.AlarmDevDeadband`

## .AlarmDevDeadband 点域

更改主、副偏差报警的偏差死区百分比。

### 类别

报警

### 用法

*TagName*.AlarmDevDeadband

### 参数

*TagName*

任何整型、实型或间接模拟标记。

### 附注

如果希望在有意或无意关闭 WindowViewer 之后，仍可以继续使用此点域的运行时值，请在“标记名字典”中选择**保留参数**选项。

### 数据类型

整型（可读写）

### 有效值

0 到 100

### 示例

以下语句将偏差死区百分比更改为 25%：

```
tag.AlarmDevDeadband=25;
```

### 另请参阅

.AlarmValDeadband, .AlarmDev

## 更改与标记关联的报警注释

.AlarmComment 点域返回与标记或报警组的报警关联的注释文本字符串。

### .AlarmComment 点域

返回与标记或报警组的报警关联的注释文本字符串。缺省条件下，在新应用程序中它是一个空字符串。

#### 类别

报警

#### 用法

*TagName*.AlarmComment

#### 参数

*TagName*

任何标记或报警组。

#### 数据类型

消息（可读写）

#### 有效值

文本

#### 示例

下例返回标记的报警注释，并将它放入另一个“内存消息”标记：

```
mTag1=Tag1.AlarmComment;
```

下例返回分布式报警对象 AlmObj\_1 中所选标记的报警注释，并将它放入 almComment 内存消息标记：

```
GetPropertyM(“AlmObj_1.AlarmComment”, almComment);
```

#### 另请参阅

GetPropertyM(), .AlarmAccess, .AlarmClass, .AlarmDate, .AlarmLimit, .AlarmName, .AlarmOprName, .AlarmOprNode, .AlarmPri, .AlarmProv, .AlarmState, .AlarmTime, .AlarmType, .AlarmValue



## 将用户自定义信息关联到报警实例

您可以给报警记录附加三个项目：两个数字和一个字符串。使用以下点域将信息添加到报警记录中。

- `.AlarmUserDefNumX` 点域
- `.AlarmUserDefStr` 点域

### `.AlarmUserDefNumX` 点域

要简化用户值的设置，可以在报警组及特定的标记上设置这些点域。例如，`InBatch` 可以在 `$System` 报警组的 `.AlarmUserDefNum1` 中设置批号，从而让所有的报警都附加上该批号。

`.AlarmUserDefNum1` 与 `.AlarmUserDefNum2` 点域分别对应于 Alarm Viewer 控件中的 User1 与 User2 列。

如果在某个报警组上设置 `.AlarmUserDefNum1`，则它会应用于该组及其任何子组中的所有报警。您也可以在标记上专门设置 `.AlarmUserDefNum1` 的值。在这种情况下，它只应用于该标记，并且会改写标记的报警组中 `.AlarmUserDefNum1` 的任何设置。

#### 类别

报警

#### 用法

`TagName.AlarmUserDefNum1`

`TagName.AlarmUserDefNum2`

#### 参数

*TagName*

任何离散、整型、实型、间接离散、间接模拟标记，或报警组。

#### 附注

这个用户自定义的点域可用于各种标记，尤其是离散标记、模拟标记以及报警组（无论是否给它们定义了报警）。针对任何单独的标记、组或父组，可以不设置这些项目、设置所有这些项目，或者只设置其中的一部分。

此点域的值附加到报警上，但“仅限于”已设置（例如通过脚本或 POKE）值的情况。

#### 数据类型

模拟（可读写）

#### 有效值

任何实数值或不设置（缺省）

**示例**

以下示例使用常数值。不过，您可以使用 **InTouch QuickScript** 将另一个标记的值复制到这些用户自定义点域中的任何一个。您也可以使用 **PtAcc** 来设置或检测它们，或者将 **InTouch** 用作“**I/O 服务器**”来获取或设置这些值。

```
$System.AlarmUserDefNum1 = 4;
GroupA.AlarmUserDefNum1 = 27649;
```

从概念上讲，将报警通知发送给“分布式报警系统”时，使用最低级别的设置。这就是说，如果标记的 **.AlarmUserDefNum1** 设置为某个值，则应该使用该设置来填写报警记录。不过，如果标记没有这样的值，则 **WindowViewer** 会检查该标记的报警组是否有值，如此逐层升级，直至到达 **\$System** 根组。如果在任何级别都没有发现任何设置，则报警记录中的输入项将保持空白（如果是数字，则为零；如果是字符串，则为空字符串）。

---

**备注** 这种层次结构化的搜索是根据每个项目分别进行处理的。因此，如果标记已设置 **.AlarmUserDefNum2**，而未设置 **AlarmUserDefNum1**，但其父组已设置 **.AlarmUserDefNum1**，则标记将从它的父组继承 **.AlarmUserDefNum1** 的设置。

---

**另请参阅**

**.AlarmUserDefStr**

## **.AlarmUserDefStr** 点域

**.AlarmUserDefStr** 点域会附加到报警数据库中由 **Alarm DB Logger** 记录的每个报警的信息中。**.AlarmUserDefStr** 点域对应于数据库字段 **User3**。您可以在 **SELECT** 语句中使用“用户自定义”列来选择用于数据库操作的特定报警集合。例如，如果将 **\$System.AlarmUserDefStr** 设置为“批次字符串”，并且它每次都会随着“批次”的更改而变化，则包含数据库字段 **User3** 的选择可用于选择特定批次的报警。

**类别**

报警

**用法**

*Tagname*.AlarmUserDefStr

**参数**

*Tagname*

任何离散、整型、实型、间接离散、间接模拟标记，或报警组。

**附注**

这个用户自定义的点域可用于各种标记，尤其是离散标记、模拟标记以及报警组（无论是否给它们定义了报警）。针对任何单独的标记、组或父组，可以不设置这些项目、设置所有这些项目，或者只设置其中的一部分。

此点域的值附加到报警上，但“仅”限于值已设置（例如通过脚本或 POKE）的情况。

**数据类型**

消息（可读写）

**有效值**

NULL 与任何有效的字符串

**示例**

本例使用常数值。不过，您可以使用 **InTouch QuickScript** 将另一个标记的值复制到这些用户自定义点域中的任何一个。您也可以使用 **PtAcc** 来设置或检测它们，或者将 **InTouch** 用作“I/O 服务器”来获取或设置这些值。

```
Tag04.AlarmUserDefStr = "Joe";
```

从概念上讲，将报警通知发送给“分布式报警系统”时，使用最低级别的设置。这就是说，如果将标记的 **.AlarmUserDefStr** 设置为某个值，则应该使用该设置来填写报警记录。不过，如果标记没有这样的值，则 **WindowViewer** 将检查该标记的报警组是否有，如此逐层升级，直至到达 **\$System** 根组。如果在任何级别都没有发现任何设置，则报警记录中的输入项将保持空白（如果是数字，则为零；如果是字符串，则为空字符串）。

此外请注意，这种层次结构化的搜索是根据每个项目分别进行处理的。因此，如果某个标记已设置为 **.AlarmUserDefNum1**，而未设置为 **.AlarmUserDefStr**，但其父组已设置 **.AlarmUserDefStr**，则该设置会用在报警记录中。

**另请参阅**

**.AlarmUserDefNumX**

## 确定标记或报警组的约束标记

使用以下点域确定各种报警类型的约束标记。

- `.AlarmDscInhibitor` 点域
- `.AlarmLoLoInhibitor` 点域
- `.AlarmLoInhibitor` 点域
- `.AlarmHiInhibitor` 点域
- `.AlarmHiHiInhibitor` 点域
- `.AlarmMinDevInhibitor` 点域
- `.AlarmMajDevInhibitor` 点域
- `.AlarmROCIInhibitor` 点域

### `.AlarmDscInhibitor` 点域

返回指定给离散报警的约束标记的名称。

**类别**

报警

**用法**

`TagName.AlarmDscInhibitor`

**参数**

*TagName*

任何离散标记或报警组。

**附注**

在 WindowMaker 中配置。不能在运行时更改。

**数据类型**

消息（只读）

**示例**

`.AlarmDSCInhibitor` 的使用方式如下：将 `.Name` 设置为等于 `.AlarmDscInhibitor` 标记值的某个间接标记，然后操纵该间接标记的值。

以下语句返回离散报警的报警约束标记的名称（假设 `SomeIndirectTag` 是一个模拟间接标记）：

```
SomeIndirectTag.Name =  
    AlarmedTag.AlarmDscInhibitor;
```

报警标记的约束状态可以通过设置间接标记的值来控制，具体如下：

```
SomeIndirectTag = 1;
```

打开约束。禁用 AlarmedTag 的离散报警

```
SomeIndirectTag = 0;
```

关闭约束

可以为 AlarmedTag 生成离散报警

## **.AlarmLoLoInhibitor** 点域

返回指定给 LoLo 报警的约束标记的名称。

**类别**

报警

**用法**

```
TagName.AlarmLoLoInhibitor
```

**参数**

*TagName*

任何整型、实型、间接模拟标记，或报警组标记。

**附注**

在 WindowMaker 中配置。不能在运行时更改。

**数据类型**

消息（只读）

**示例**

.AlarmLoLoInhibitor 点域的使用方式如下：将 .Name 设置为等于 .AlarmLoLoInhibitor 标记值的间接标记，然后操纵该间接标记的值。

以下语句返回 LoLo 报警限的报警约束标记的名称（假设 SomeIndirectTag 是一个模拟间接标记）：

```
SomeIndirectTag.Name =  
    AlarmedTag.AlarmLoLoInhibitor;
```

报警标记的约束状态可以通过设置间接标记的值来控制，具体如下：

```
SomeIndirectTag = 1;
```

打开约束

禁用 AlarmedTag 的 LoLo 报警

```
SomeIndirectTag = 0;
```

关闭约束

可以为 AlarmedTag 生成 LoLo 报警

**另请参阅**

.AlarmHiInhibitor, .AlarmHiHiInhibitor, .AlarmLoInhibitor

## **.AlarmLoInhibitor 点域**

返回指定给 Low 报警的约束标记的名称。

**类别**

报警

**用法**

*TagName*.AlarmLoInhibitor

**参数**

*TagName*

任何整型、实型、间接模拟标记，或报警组标记。

**附注**

在 WindowMaker 中配置。不能在运行时更改。

**数据类型**

消息（只读）

**示例**

.AlarmLoInhibitor 点域的使用方式如下：将 .Name 设置为等于 .AlarmLoInhibitor 标记值的间接标记，然后操纵该间接标记的值。

以下语句返回 Low 报警限的报警约束标记的名称（假设 SomeIndirectTag 是一个模拟间接标记）：

```
SomeIndirectTag.Name =  
    AlarmedTag.AlarmLoInhibitor;
```

报警标记的约束状态可以通过设置间接标记的值来控制，具体如下：

```
SomeIndirectTag = 1;
```

打开约束

禁用 AlarmedTag 的 Low 报警

```
SomeIndirectTag = 0;
```

关闭约束

可以为 AlarmedTag 生成 Low 报警

**另请参阅**

.AlarmHiInhibitor, .AlarmHiHiInhibitor,  
.AlarmLoLoInhibitor

## **.AlarmHiInhibitor 点域**

返回指定给 High 报警的约束标记的名称。

**类别**

报警

**用法**

*TagName*.AlarmHiInhibitor

**参数**

*TagName*

任何整型、实型、间接模拟标记，或报警组标记。

**附注**

在 WindowMaker 中配置。不能在运行时更改。

**数据类型**

消息（只读）

**示例**

.AlarmHiInhibitor 点域的使用方式如下：将 .Name 设置为等于 .AlarmHiInhibitor 标记值的间接标记，然后操纵该间接标记的值。

以下语句返回 High 报警限的报警约束标记的名称（假设 SomeIndirectTag 是一个模拟间接标记）：

```
SomeIndirectTag.Name =  
    AlarmedTag.AlarmHiInhibitor;
```

报警标记的约束状态可以通过设置间接标记的值来控制，具体如下：

```
SomeIndirectTag = 1;
```

打开约束

禁用 AlarmedTag 的 High 报警

```
SomeIndirectTag = 0;
```

关闭约束

可以为 AlarmedTag 生成 High 报警

**另请参阅**

.AlarmHiHiInhibitor, .AlarmLoInhibitor,  
.AlarmLoLoInhibitor

## .AlarmHiHiInhibitor 点域

返回指定给 HiHi 报警条件的约束标记的名称。

**类别**

报警

**用法**

*TagName*.AlarmHiHiInhibitor

**参数**

*TagName*

任何整型、实型、间接模拟标记，或报警组标记。

**附注**

在 WindowMaker 中配置。不能在运行时更改。

**数据类型**

消息（只读）

**示例**

.AlarmHiHiInhibitor 点域的使用方式如下：将 .Name 设置为等于 .AlarmHiHiInhibitor 标记值的间接标记，然后操纵该间接标记的值。以下语句返回 HiHi 报警限的报警约束标记的名称（假设 SomeIndirectTag 是一个模拟间接标记）：

```
SomeIndirectTag.Name =  
    AlarmedTag.AlarmHiHiInhibitor;
```

报警标记的约束状态可以通过设置间接标记的值来控制，具体如下：

```
SomeIndirectTag = 1;
```

打开约束

禁用 AlarmedTag 的 HiHi 报警

```
SomeIndirectTag = 0;
```

关闭约束

可以为 AlarmedTag 生成 HiHi 报警

**另请参阅**

.AlarmHiInhibitor, .AlarmLoInhibitor, .AlarmLoLoInhibitor



## .AlarmMinDevInhibitor 点域

返回与副偏差报警条件关联的报警约束标记的名称。

### 类别

报警

### 用法

`TagName.AlarmMinDevInhibitor`

### 参数

*TagName*

任何整型、实型、间接模拟标记，或报警组标记。

### 附注

在 WindowMaker 中配置。不能在运行时更改。

### 数据类型

消息（只读）

### 示例

.AlarmMinDevInhibitor 点域的使用方式如下：将 .Name 设置为等于 .AlarmMinDevInhibitor 标记值的间接标记，然后操纵该间接标记的值。以下语句返回副偏差报警限的报警约束标记的名称（假设 SomeIndirectTag 是一个模拟间接标记）：

```
SomeIndirectTag.Name =  
    AlarmedTag.AlarmMinDevInhibitor;
```

报警标记的约束状态可以通过设置间接标记的值来控制，具体如下：

```
SomeIndirectTag = 1;
```

打开约束

禁用 AlarmedTag 的副偏差报警

```
SomeIndirectTag = 0;
```

关闭约束

可以为 AlarmedTag 生成副偏差报警

### 另请参阅

.AlarmMajDevInhibitor

## **.AlarmMajDevInhibitor** 点域

返回与主偏差报警条件关联的报警约束标记的名称。

**类别**

报警

**用法**

*TagName*.AlarmMajDevInhibitor

**参数**

*TagName*

任何整型、实型、间接模拟标记，或报警组标记。

**数据类型**

消息（只读）

**示例**

.AlarmMajDevInhibitor 点域的使用方式如下：将 .Name 设置为等于 .AlarmMajDevInhibitor 标记值的间接标记，然后操纵该间接标记的值。以下语句返回主偏差报警限的报警约束标记的名称（假设 SomeIndirectTag 是一个模拟间接标记）：

```
SomeIndirectTag.Name =  
    AlarmedTag.AlarmMajDevInhibitor;
```

报警标记的约束状态可以通过设置间接标记的值来控制，具体如下：

```
SomeIndirectTag = 1;
```

打开约束

禁用 AlarmedTag 的主偏差报警

```
SomeIndirectTag = 0;
```

关闭约束

可以为 AlarmedTag 生成主偏差报警

**另请参阅**

.AlarmMinDevInhibitor

## **.AlarmROCIInhibitor 点域**

返回与变化率报警条件关联的报警约束标记的名称。

**类别**

报警

**用法**

*TagName*.AlarmROCIInhibitor

**参数**

*TagName*

任何整型、实型、间接模拟标记，或报警组标记。

**数据类型**

消息（只读）

**示例**

.AlarmROCIInhibitor 点域的使用方式如下：将 .Name 设置为等于 .AlarmROCIInhibitor 标记值的间接标记，然后操纵该间接标记的值。以下语句返回变化率报警限的报警约束标记的名称（假设 SomeIndirectTag 是一个模拟间接标记）：

```
SomeIndirectTag.Name =  
    AlarmedTag.AlarmROCIInhibitor;
```

报警标记的约束状态可以通过设置间接标记的值来控制，具体如下：

```
SomeIndirectTag = 1;
```

打开约束

禁用 AlarmedTag 的变化率报警

```
SomeIndirectTag = 0;
```

关闭约束

可以为 AlarmedTag 生成变化率报警

## 统计活动的或未确认的报警数

使用以下点域找出应用程序运行期间活动的或未确认的报警的数量。

点域	描述
.AlarmTotalCount 点域	统计与标记或报警组关联的报警数。
.AlarmUnAckCount 点域	统计与标记或报警组关联的未确认报警数。
.AlarmValueCount 点域	统计与标记关联的值报警数。
.AlarmValueUnAckCount 点域	统计与标记关联的未确认的值报警数。
.AlarmDscCount 点域	统计离散报警数。
.AlarmDscUnAckCount 点域	统计未确认的离散报警数。
.AlarmDevCount 点域	统计偏差报警数。
.AlarmDevUnAckCount 点域	统计未确认的偏差报警数。
.AlarmROCCount 点域	统计变化率报警数。
.AlarmROCUnAckCount 点域	统计未确认的变化率报警数。

## **.AlarmTotalCount** 点域

统计指定的标记或报警组的活动报警总数。

### **类别**

报警

### **用法**

*TagName*.AlarmTotalCount

### **参数**

*TagName*

任何类型的标记或报警组。

### **附注**

该计数包括值、偏差、变化率及离散报警。它包括已确认与未确认的报警数。

### **数据类型**

整型（只读）

### **有效值**

0 或任何正整数

### **示例**

Tag1 是给报警配置的模拟标记。ATC 也是一个模拟标记，它获取 Tag1 中存在的所有活动报警（同时包含“未确认”与“确认”的）的总数。

```
ATC = Tag1.AlarmTotalCount;
```

**另请参阅**

.AlarmDevCount, .AlarmDevUnAckCount,  
.AlarmDSCCount, .AlarmDSCUnAckCount,  
.AlarmValueCount, .AlarmUnAckCount,  
.AlarmValueUnAckCount, .AlarmROCCount,  
.AlarmROCUAckCount

## **.AlarmUnAckCount 点域**

统计指定的标记或报警组的未确认报警总数。

**类别**

报警

**用法**

*TagName*.AlarmUnAckCount

**参数**

*TagName*

任何类型的标记或报警组。

**附注**

该计数包括未确认的值、偏差、变化率及离散报警。

**数据类型**

整型（只读）

**有效值**

0 或任何正整数

**示例**

Tag1 是给报警配置的模拟或离散标记。AUC 是一个模拟标记，它获取 Tag1 中存在的未确认报警总数。

```
AUC = Tag1.AlarmUnAckCount;
```

**另请参阅**

.AlarmDevCount, .AlarmDevUnAckCount, .AlarmDscCount,  
.AlarmDscUnAckCount, .AlarmValueCount,  
.AlarmTotalCount, .AlarmValueUnAckCount,  
.AlarmROCCount, .AlarmROCUAckCount

## **.AlarmValueCount 点域**

统计指定的标记或报警组的活动值报警总数。

### **类别**

报警

### **用法**

*TagName*.AlarmValueCount

### **参数**

*TagName*

任何整型、实型、间接模拟标记，或报警组。

### **附注**

这包括 HiHi、High、Low 及 LoLo 报警的计数。它包括已确认与未确认的报警数。对于非扩展的摘要报警标记，此计数不超过 1。不过，此计数可能会因报警组的不同而不同。

### **数据类型**

整型（只读）

### **有效值**

0 或任何正整数

### **示例**

Tag1 是给值报警配置的模拟标记。AVC 也是一个模拟标记，它获取 Tag1 中存在的所有报警值的总数。

```
AVC = Tag1.AlarmValueCount;
```

### **另请参阅**

.AlarmDevCount, .AlarmDevUnAckCount, .AlarmDscCount,  
.AlarmDscUnAckCount, .AlarmROCCount,  
.AlarmTotalCount, .AlarmValueUnAckCount,  
.AlarmROCUAckCount, .AlarmUnAckCount

## **.AlarmValueUnAckCount** 点域

统计指定的标记或报警组的未确认值报警总数。这包括 HiHi、High、Low 及 LoLo 报警的计数。

### **类别**

报警

### **用法**

*TagName*.AlarmValueUnAckCount

### **参数**

*TagName*

任何整型、实型、间接模拟标记，或报警组。

### **数据类型**

整型（只读）

### **有效值**

0 或任何正整数

### **示例**

Tag1 是给值报警配置的模拟标记。AVUC 也是一个模拟标记，它获取 Tag1 中存在的所有未确认的值报警总数。

```
AVUC = Tag1.AlarmValueUnAckCount;
```

### **另请参阅**

.AlarmDevCount, .AlarmDevUnAckCount, .AlarmDscCount,  
.AlarmDscUnAckCount, .AlarmROCCount,  
.AlarmTotalCount, .AlarmValueCount,  
.AlarmROCUAckCount, .AlarmUnAckCount



## **.AlarmDscCount 点域**

统计指定的标记或报警组的活动离散报警总数。

### **类别**

报警

### **用法**

```
TagName.AlarmDscCount;
```

### **参数**

*TagName*

任何离散标记、间接离散标记，或报警组。

### **附注**

指定给 .AlarmDscCount 点域的计数同时包括已确认和未确认的报警。对于非扩展的摘要报警标记，此计数始终等于 1。不过，此计数可能会因报警组的不同而不同。

### **数据类型**

整型（只读）

### **有效值**

0 或任何正整数

### **示例**

Tag1 是给离散报警配置的离散标记。ADC 是模拟标记，它获取 Tag1 中存在的活动离散报警（同时包括已确认和未确认的）的总数。

```
ADC = Tag1.AlarmDSCCount;
```

### **另请参阅**

.AlarmDevCount, .AlarmDevUnAckCount,  
.AlarmValueCount, .AlarmROCUAckCount,  
.AlarmTotalCount, .AlarmDscUnAckCount,  
.AlarmValueUnAckCount, .AlarmROCUAckCount,  
.AlarmUnAckCount

## .AlarmDscUnAckCount 点域

统计指定的标记或报警组的未确认离散报警总数。

**类别**

报警

**用法**

*TagName*.AlarmDscUnAckCount

**参数**

*TagName*

任何离散标记、间接离散标记，或报警组。

**数据类型**

整型（只读）

**有效值**

0 或任何正整数

**示例**

Tag1 是给离散报警配置的离散标记。ADUC 是模拟标记，它获取 Tag1 中存在的未确认离散报警的总数。

```
ADUC = Tag1.AlarmDscUnAckCount;
```

**另请参阅**

.AlarmDevCount, .AlarmDevUnAckCount, .AlarmDscCount,  
.AlarmValueCount, .AlarmROCCount, .AlarmTotalCount,  
.AlarmValueUnAckCount, .AlarmROCUnAckCount,  
.AlarmUnAckCount

## **.AlarmDevCount** 点域

统计指定的标记或报警组的活动偏差报警总数。

### 类别

报警

### 用法

*TagName*.AlarmDevCount

### 参数

*TagName*

任何实型、整型、间接模拟标记，或报警组。

### 附注

这包括主、副偏差报警的计数。它包括已确认与未确认的报警数。对于非扩展的摘要报警标记，此计数始终等于 1。不过，此计数可能会因报警组的不同而不同。

### 数据类型

模拟（只读）

### 有效值

0 或任何正整数

### 示例

Tag1 是给“偏差”报警配置的模拟标记。ADC 也是一个模拟标记，它获取 Tag1 中存在的活动偏差报警（同时包含已确认和未确认的）的总数。

```
ADC=Tag1.AlarmDevCount;
```

### 另请参阅

.AlarmDSCCount, .AlarmValueCount,  
.AlarmROCUAckCount, .AlarmTotalCount,  
.AlarmDSCUnAckCount, .AlarmValueUnAckCount,  
.AlarmDevUnAckCount, .AlarmROCUAckCount,  
.AlarmUnAckCount

## **.AlarmDevUnAckCount 点域**

统计指定的标记或报警组的未确认偏差报警的总数。这包括主、副偏差报警的计数。

**类别**

报警

**用法**

*TagName*.AlarmDevUnAckCount

**参数**

*TagName*

任何实型、整型、间接模拟标记，或报警组。

**数据类型**

模拟（只读）

**有效值**

0 或任何正整数

**示例**

Tag1 是给“偏差”报警配置的模拟标记。ADUC 也是一个模拟标记，它获取 Tag1 中存在的未确认偏差报警的总数。

```
ADUC = Tag1.AlarmDevUnAckCount;
```

**另请参阅**

.AlarmDevCount, .AlarmDSCCount, .AlarmValueCount,  
.AlarmROCUAckCount, .AlarmTotalCount,  
.AlarmDSCUnAckCount, .AlarmValueUnAckCount,  
.AlarmROCUAckCount, .AlarmUnAckCount

## **.AlarmROCCount** 点域

统计指定的标记或报警组的活动变化率报警总数。它包括已确认与未确认的报警数。对于非扩展的摘要报警标记，此计数将始终等于 1。不过，此计数可能会因报警组的不同而不同。

### **类别**

报警

### **用法**

*TagName*.AlarmROCCount

### **参数**

*TagName*

任何实型、整型、间接模拟标记，或报警组。

### **数据类型**

整型（只读）

### **有效值**

0 或任何正整数

### **示例**

Tag1 是给变化率报警配置的模拟标记。ARC 也是一个模拟标记，它获取 Tag1 中存在的活动变化率报警（同时包含已确认和未确认）的总数。

```
ARC = Tag1.AlarmROCCount;
```

### **另请参阅**

.AlarmDevCount, .AlarmDevUnAckCount, .AlarmDscCount, .AlarmDscUnAckCount, .AlarmValueCount, .AlarmTotalCount, .AlarmValueUnAckCount, .AlarmROCUAckCount, .AlarmUnAckCount

## **.AlarmROCUAckCount 点域**

统计指定的模拟标记或报警组的未确认变化率报警总数。

**类别**

报警

**用法**

*TagName*.AlarmROCUAckCount

**参数**

*TagName*

任何实型、整型、间接模拟标记，或报警组。

**数据类型**

整型（只读）

**有效值**

0 或任何正整数

**示例**

Tag1 是给变化率报警配置的模拟标记。ADUC 也是一个模拟标记，它获取 Tag1 中存在的未确认变化率报警的总数。

```
ARUC = Tag1.AlarmROCUAckCount;
```

**另请参阅**

.AlarmDevCount, .AlarmDevUnAckCount, .AlarmDscCount,  
.AlarmDscUnAckCount, .AlarmValueCount,  
.AlarmTotalCount, .AlarmValueUnAckCount,  
.AlarmROCCount, .AlarmUnAckCount

## 第 7 章

### 查看报警层次结构

Alarm Tree Viewer ActiveX 控件显示报警查询所选的报警供应器的报警组层次结构。Alarm Tree Viewer 控件中出现的项目包括报警供应器、节点以及组。



您可以通过使用 Alarm Tree Viewer 控件来提高 Alarm Viewer 控件的有用性。您可以创建一个脚本，使操作员在 Alarm Tree Viewer 控件中选择报警供应器时，Alarm Viewer 控件会查询新的报警供应器。

您可以配置 Alarm Tree Viewer 控件的显示方式以及显示的数据。如需有关详细信息，请参阅第 192 页的“配置 Alarm Tree Viewer 控件”。

完成 Alarm Tree Viewer 控件的配置时，可以通过以下操作来修改要查看的数据：

- 按名称给数据排序。
- 更新目录树。
- 执行其它查询。

如需有关 ActiveX 控件的详细信息，请参阅 *InTouch® HMI 可视化指南* 中的第 6 章“ActiveX 控件”。

## 配置 Alarm Tree Viewer 控件

您可以为 Alarm Tree Viewer 控件配置以下属性：

- 一般控件外观，包括颜色
- 文本字体
- 自动刷新
- 用户可以在运行时访问的功能
- 要显示的供应器与组
- 自定义的保存查询
- 报警组的排序顺序

您可以从 WindowMaker 中以及在 Alarm Tree Viewer 控件运行期间配置这些选项。

### 配置外观与颜色

配置 Alarm Tree Viewer 控件的视觉外观时，可以：

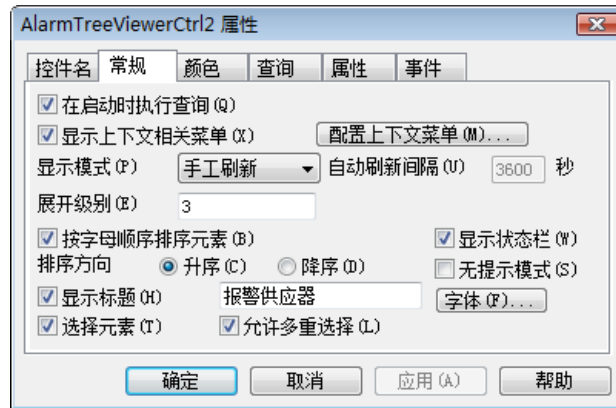
- 包含状态栏。
- 包含列标题。
- 设置可视化元素的颜色。

#### 要配置外观

- 1 使用鼠标右键单击 Alarm Tree Viewer 控件，然后单击属性。此时出现 **AlarmTreeViewCtrl 属性** 对话框。



## 2 单击常规选项卡。



## 3 配置 Alarm Tree Viewer 控件呈现给运行时用户的方式。执行以下任何操作：

- 选择**在启动时执行查询**复选框，以使用缺省查询属性自动更新目录树。否则用户必须运行**刷新**命令才能更新目录树。
- 选择**显示上下文相关菜单**复选框以激活快捷菜单。单击**配置上下文菜单**以配置要在菜单上出现的命令。如需有关详细信息，请参阅第 195 页的“控制用户可以在运行时访问的功能”。
- 在**显示模式**列表中，单击所需的目录树刷新方式。对于自动刷新，请在**自动刷新间隔**框中输入刷新间隔。范围是 5 到 32767 秒。
- 在**展开级别**框中，输入目录树的展开级数。这确定手工刷新控件时报警目录树打开的报警组分支级别。值为 1 时仅显示供应器，值为 2 时显示供应器的直接报警组，依此类推。
- 选择**按字母顺序排列元素**复选框，以便按字母顺序给目录树元素排序。单击**升序**或**降序**作为排序方向。
- 选择**显示标题**复选框，以便在层次结构上显示标题。在方框中，输入标题栏文本。
- 选择**显示状态栏**复选框，以便在 Alarm Tree Viewer 控件底部显示状态栏。
- 单击**字体**以配置目录树的字体属性。此时出现标准的 Windows 字体对话框。
- 选择**选择元素**复选框，使用户可以选择目录树中的元素。
- 选择**允许多重选择**复选框，使用户可以使用 CTRL 与 SHIFT 键选择一个或多个元素。
- 选择**无提示模式**复选框，以防止 Alarm Tree Viewer 控件显示运行时错误消息。错误消息总是发送到 Logger 中。

## 4 单击应用。

- 5 单击**颜色**选项卡。



- 6 单击调色板按钮，以便给 Alarm Tree Viewer 控件的可视化元素指定颜色。

您可以设置标题栏文本、窗口背景、所选元素文本以及所选元素背景的颜色。

- 7 单击**应用**。

## 配置字体

您可以为 Alarm Tree Viewer 控件配置文本外观。

### 要配置字体

- 1 使用鼠标右键单击 Alarm Tree Viewer 控件，然后单击**属性**。此时出现 **AlarmTreeViewCtrl 属性** 对话框。
- 2 单击**常规**选项卡。



- 3 单击**字体**。此时出现标准的 Windows 字体对话框。配置字体，然后单击**确定**。
- 4 单击**确定**。

## 配置自动刷新

您可以将 Alarm Tree Viewer 控件配置为在运行时自动刷新。否则，操作员必须手工刷新 Alarm Tree Viewer 控件。

### 要配置自动刷新

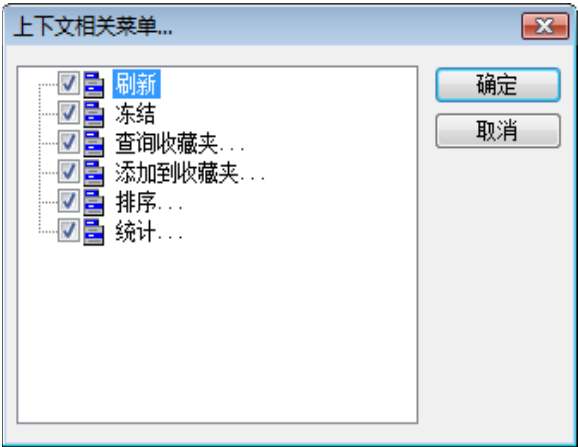
- 1 使用鼠标右键单击 Alarm Tree Viewer 控件，然后单击属性。此时出现 AlarmTreeViewCtrl 属性对话框。
- 2 单击常规选项卡。
- 3 在显示模式列表中，单击所需的目录树刷新方式，即手工刷新或自动刷新。对于自动刷新，请在自动刷新闻隔框中输入目录树刷新闻隔。范围是 5 到 32767 秒。
- 4 单击应用。

## 控制用户可以在运行时访问的功能

Alarm Tree Viewer 控件包含一个快捷菜单，通过使用鼠标右键单击控件，操作员可以在运行时打开它。您可以配置要在菜单上出现的命令。

### 要配置运行时快捷菜单

- 1 使用鼠标右键单击 Alarm Tree Viewer 控件，然后单击属性。此时出现 AlarmTreeViewCtrl 属性对话框。
- 2 单击常规选项卡。
- 3 选择显示上下文相关菜单复选框以激活快捷菜单。
- 4 单击配置上下文菜单。此时出现上下文相关菜单对话框。



- 5 选择要在快捷菜单中出现的每个命令对应的复选框。您至少必须选择一个快捷菜单命令。

命令	描述
刷新	刷新 Alarm Tree Viewer 控件中显示的数据。

命令	描述
冻结	可用于切换目录树的冻结 / 撤销冻结模式。
查询收藏夹	显示 <b>报警查询</b> 对话框，以便从可用列表中选择一个查询收藏夹。
添加到收藏夹	可用于从 <b>添加查询</b> 对话框中添加新的查询。
排序	显示 <b>排序</b> 对话框，以按照升序或降序给 Alarm Tree Viewer 控件数据排序。
统计	显示 <b>报警统计</b> 对话框，列出 Alarm Tree Viewer 控件中显示的内容占当前所检索的报警供应器的百分比。

- 6 单击**确定**以关闭上下文相关菜单对话框。
- 7 单击**应用**。

## 配置要显示的供应器与组

您可以为属于 Alarm Tree Viewer 控件的报警供应器与组配置报警查询。报警查询是由空格分隔的一个或多个报警供应器的列表。报警供应器的有效语法如下。

报警供应器的完整路径：

\\Node\ProviderName

本地报警供应器的路径：

\ProviderName

对于多个查询，请使用空格分隔每个查询。例如：

\InTouch \\Node17\InTouch \\MyNode\InTouch

缺省的报警查询是 \InTouch。不得将标记用于报警查询。

如果查询多个报警组，并且稍后取消部署了一个或多个组，则 Alarm Tree Viewer 控件不自动更新以便从视图中删除这些组。您必须停止并重新启动报警供应器，才能取消它的注册。

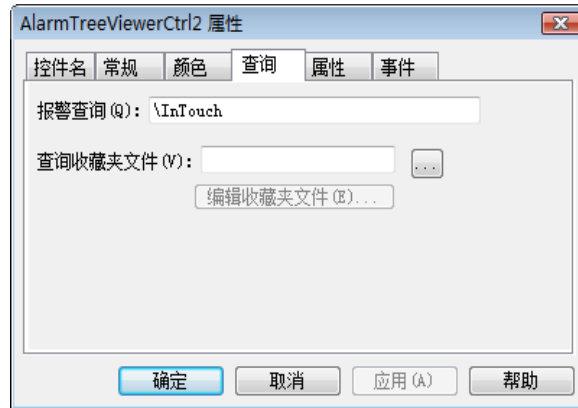
如果通过在报警查询中指定 \Galaxy 来查询 ArchestrA Galaxy，则显示 Galaxy 中部署的所有 InTouch 报警供应器。例如：

\\Node\Galaxy!Area[name]

如果从具有多个报警供应器的节点查询信息，并且这些供应器包含同名的组，则目录树中显示最后一个报警供应器的记录。

### 要配置报警查询

- 1 使用鼠标右键单击 Alarm Tree Viewer 控件，然后单击属性。此时出现 AlarmTreeViewCtrl 属性对话框。
- 2 单击查询选项卡。



- 3 在报警查询框中，输入初始报警查询的路径。
- 4 单击应用。

## 使用查询收藏夹创建自定义的保存查询

您可以配置一个查询收藏夹列表，供操作员从快捷菜单中进行选择。

查询文件可以保存到任何文件夹，而不必保存到 InTouch 应用程序文件夹中。报警查询文件是 .xml 文件。

### 要配置查询收藏夹文件

- 1 使用鼠标右键单击 Alarm Tree Viewer 控件，然后单击**属性**。此时出现 **AlarmTreeViewCtrl 属性** 对话框。
- 2 单击**查询**选项卡。



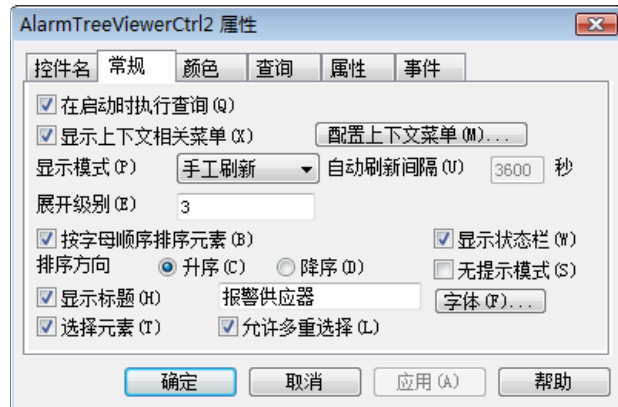
- 3 配置查询收藏夹文件。
  - a 在**查询收藏夹文件**框中，输入网络路径与文件名，或单击省略号按钮以浏览文件。
  - b 要编辑**查询收藏夹文件**，请单击**编辑查询收藏夹**按钮。此时打开**报警查询**窗口，可供您在收藏夹文件中添加、修改或删除过滤器。完成时单击**确定**，以保存更改并关闭窗口。
- 4 单击**确定**。

## 配置报警组的排序顺序

在 Alarm Tree Viewer 控件中，节点与报警组可以按字母顺序的升序或降序显示。

### 要配置报警组的排序顺序

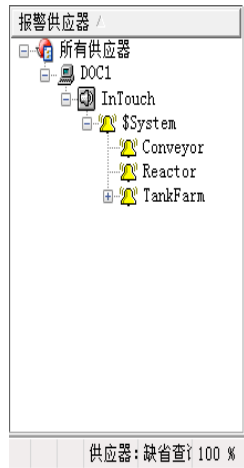
- 1 使用鼠标右键单击 Alarm Tree Viewer 控件，然后单击**属性**。此时出现 **AlarmTreeViewCtrl 属性** 对话框。
- 2 单击**常规**选项卡。



- 3 选择**按字母顺序排序元素**复选框，以便按字母顺序列出报警组。
- 4 单击**升序**或**降序**以指定排序方向。
- 5 单击**确定**。

## 在运行时使用 Alarm Tree Viewer 控件

Alarm Tree Viewer 控件可用于浏览报警供应器与报警组的层次结构。



Alarm Tree Viewer 控件可以显示多个节点与报警供应器。

- 节点由计算机图标表示。
- 报警供应器由扬声器图标表示。
- 报警组由钟型图标表示。

选择一个或多个报警组时，可以生成报警查询，以用在 Alarm Tree Viewer 控件与 Alarm DB View 控件中。要选择多个报警组，请在按住 Shift 键的同时单击某个组。要取消选择所有的组，请单击空白区域。

根据配置控件的方式，运行时快捷菜单上会出现一个或多个以下命令：

- **刷新** – 强制手工更新报警。
- **冻结** – 停止更新报警。
- **查询收藏夹** – 打开报警查询对话框，在其中可以从先前定义的报警查询列表中选择某个报警查询。
- **添加到收藏夹** – 打开添加查询对话框，其中已根据所选的“组”（如果有）输入了查询字符串。
- **排序** – 打开排序对话框，其中包含一些选项，可以按字母顺序的升序或降序给报警组排序。
- **统计** – 打开报警统计对话框，以显示检索的报警供应器的百分比。



## 理解状态栏信息

Alarm Tree Viewer 控件状态栏出现在窗口底部，显示以下信息：

- 当前查询的名称
- 当前查询的完成百分比状态

## 使用查询收藏夹

通过使用 Alarm Tree Viewer 控件快捷菜单上的**查询收藏夹**命令，可以从先前定义的报警查询列表中快速选择并运行报警查询。您也可以创建新的命名查询、编辑或删除现有的查询。

### 要选择并运行报警查询

- 1 在运行时使用鼠标右键单击 Alarm Tree Viewer 控件。
- 2 单击**查询收藏夹**。此时出现**报警查询**对话框。
- 3 从当前定义的查询列表中，选择要显示的命名查询。
- 4 单击**确定**。此时 Alarm Tree Viewer 控件显示所选查询的报警组信息。

## 使用 AlarmTreeView 控件 ActiveX 属性

您可以使用脚本直接设置 Alarm Tree Viewer 控件属性的值，或是将它指定给 InTouch 标记或 I/O 引用。如需有关设置属性的详细信息，请参阅 *InTouch® HMI 脚本与逻辑指南* 中的第 8 章“编写 ActiveX 控件脚本”。

下表列出 Alarm Tree Viewer 控件的所有属性。如需有关设置颜色值的详细信息，请参阅第 77 页的“配置 ActiveX 控件的颜色”。

属性名称	用途
AddtoFavoritesMenu	启用或禁用 <b>添加到收藏夹</b> 快捷菜单命令。
AlarmQuery	显示初始报警查询并允许更改查询。有效语法为 \\<node>\<provider> 或 \<provider>。
ElementSelection	控制操作员在运行时是否可以选目录树中的元素。
ExpansionLevel	设置手工刷新控件时报警目录树打开的分支级别。值为 1 时仅显示供应器，值为 2 时显示供应器的直接报警组，依此类推。
Font	获取或设置控件中显示的记录与标题的字体。
FreezeMenu	启用或禁用 <b>冻结</b> 菜单命令。
HeaderText	获取或设置出现在 AlarmTreeView 控件的标题中的文本。
MultiSelection	允许在 AlarmTreeView 控件中选择多个元素。
QueryFavoritesFile	获取或设置查询收藏夹文件名。
QueryFavoritesMenu	启用或禁用 <b>查询收藏夹</b> 菜单命令。
QueryStartup	如果选择此属性，则使用缺省查询属性自动更新 AlarmTreeView 控件。如果没有选择，则必须重新查询才能更新 Alarm Tree Viewer 控件。
RefreshInterval	获取或设置以秒为单位的控件自动刷新闻隔。
RefreshMenu	获取或设置一个值，确定快捷菜单中是否出现 <b>刷新</b> 命令。
SelTextBackColor	获取或设置所选元素的背景色。
SelTextColor	获取或设置所选元素的文本颜色。
ShowContextMenu	启用或禁用快捷菜单。
ShowHeading	显示或隐藏 Alarm Tree Viewer 控件的标题栏。
ShowStatusBar	获取或设置一个值，确定是否显示状态栏。
SilentMode	获取或设置一个值，确定控件是否处于“无提示”模式。
SortElements	启用或禁用 Alarm Tree Viewer 控件中的排序功能。

属性名称	用途
SortMenu	启用或禁用 <b>排序</b> 菜单命令。
SortOrder	获取或设置排序方向。可能值有“升序”与“降序”，分别用 0 与 1 表示。
StatsMenu	启用或禁用 <b>统计</b> 菜单命令。
TextColor	获取或设置 Alarm Tree Viewer 控件的文本颜色。
TitleBackColor	获取或设置标题栏背景色。仅在设置了 ShowHeading 属性时可用。
TitleForeColor	获取或设置标题栏前景色。仅在设置了 ShowHeading 属性时可用。
WindowColor	获取或设置 Alarm Tree Viewer 控件的窗口背景色。

## 使用 Alarm Tree Viewer 控件 ActiveX 方法

您可以在脚本中使用 Alarm Tree Viewer 控件方法来：

- 检索有关控件的信息。
- 检索有关报警层次结构中特定项目的信息。
- 冻结控件。
- 创建查询字符串。
- 运行查询。

如需有关调用方法的详细信息，请参阅 *InTouch® HMI 脚本与逻辑指南* 中的第 8 章 “编写 ActiveX 控件脚本”。

### 检索有关控件的信息

您可以使用这些方法检索有关 Alarm Tree Viewer 控件的信息。

- AboutBox() 方法
- GetElementCount() 方法

#### AboutBox() 方法

显示 AlarmTreeView 关于对话框。

##### 语法

*Object*.AboutBox()

##### 示例

控件名为 AlarmTreeViewCtrl1。

```
#AlarmTreeViewCtrl1.AboutBox();
```

#### GetElementCount() 方法

获取目录树中元素的总数。

##### 语法

*Object*.GetElementCount()

##### 示例

控件名为 AlarmTreeViewCtrl1， nTag1 是整型或实型标记。

```
nTag1 = #AlarmTreeViewCtrl1.GetElementCount();
```

## 检索有关特定项目的信息

您可以使用一组方法来检索 Alarm Tree Viewer 控件窗口中所显示的元素的相关信息。

- CheckElementMembership() 方法
- GetElementCount() 方法
- GetElementName() 方法
- GetElementPath() 方法
- GetSelectedElementCount() 方法
- GetSelectedElementName() 方法
- GetSelectedElementPath() 方法
- GetSubElementCount() 方法
- GetSubElementName() 方法
- GetSubElementPath() 方法

### CheckElementMembership() 方法

检查子目录树元素是否为父目录树元素的一部分。

#### 语法

*Object*.CheckElementMembership(*PathName*,  
*DescendantElementName*, *AncestorElementName*)

#### 参数

##### *PathName*

路径的名称。例如， \InTouch 或 \\NodeName。

##### *DescendantElementName*

子元素的名称。例如， GroupA。

##### *AncestorElementName*

父元素的名称。例如 ,GroupB。

### GetElementCount() 方法

获取目录树中元素的总数。

#### 语法

*Object*.GetElementCount()

#### 示例

控件名为 AlarmTreeViewCtrl1， nTag1 是整型或实型标记。

```
nTag1 = #AlarmTreeViewCtrl1.GetElementCount();
```

### GetElementName() 方法

获取与索引对应的元素名。

#### 语法

*Object*.GetElementName(*ElementIndex*)

#### 参数

*ElementIndex*

元素的索引。

#### 示例

控件名为 AlarmTreeViewCtrl1， StrTag 是消息标记。

```
StrTag = #AlarmTreeViewCtrl1.GetElementName(3);
```

### GetElementPath() 方法

获取索引对应的元素路径，一直到指定的展开级别。

#### 语法

*Object*.GetElementPath(*ElementIndex*, *ExpansionLevel*)

#### 参数

*ElementIndex*

元素的索引。

*ExpansionLevel*

展开的级别。

#### 示例

控件名为 AlarmTreeViewCtrl1， StrTag 是消息标记，返回索引为 17 的元素的路径，最多 4 个展开级别。

```
StrTag = #AlarmTreeViewCtrl1.GetElementPath(17,  
4);
```

### GetSelectedElementCount() 方法

获取目录树中所选元素的数量。

#### 语法

*Object*.GetSelectedElementCount()

#### 示例

控件名为 AlarmTreeViewCtrl1， nTag1 是整型或实型标记。

```
nTag1 = #AlarmTreeViewCtrl1.GetSelectedElementCount();
```

### GetSelectedElementName() 方法

获取 Alarm Tree Viewer 控件上所选元素的名称。

#### 语法

*Object*.GetSelectedElementName()

**示例**

控件名为 AlarmTreeViewCtrl1， StrTag 是消息标记。

```
StrTag =
    #AlarmTreeViewCtrl1.GetSelectedElementName();
```

**GetSelectedElementPath() 方法**

获取到指定的展开级别的所选元素的路径。

**语法**

*Object*.GetSelectedElementPath(*ExpansionLevel*)

**参数**

*ExpansionLevel*

展开的级别。

**示例**

控件名为 AlarmTreeViewCtrl1， StrTag 是消息标记。

```
StrTag =
    #AlarmTreeViewCtrl1.GetSelectedElementPath(3)
    ;
```

**GetSubElementCount() 方法**

获取指定的元素中的子元素总数。

**语法**

*Object*.GetSubElementCount(*Path*, *ElementName*)

**参数**

*Path*

路径的名称。例如：

\\NodeName\\InTouch

如果 Path 参数为空，则 Alarm Tree Viewer 控件查找与指定的元素名匹配的第一个目录树元素。

*ElementName*

元素的名称。例如， Group1。

**示例**

控件名为 AlarmTreeViewCtrl1， nTag1 是整型或实型标记。

```
nTag1 =
    #AlarmTreeViewCtrl1.GetSubElementCount("",
    "Group1" );
nTag1 = #AlarmTreeViewCtrl1.GetSubElementCount(
    "\\NodeName", "Group1" );
nTag1 = #AlarmTreeViewCtrl1.GetSubElementCount(
    "\\InTouch", "Group1" );
nTag1 = #AlarmTreeViewCtrl1.GetSubElementCount(
    "\\NodeName\\InTouch", "Group1" );
```

### GetSubElementName() 方法

对于指定的元素，获取相应索引上的子元素的名称。

#### 语法

*Object*.GetSubElementName(*Path*, *ElementName*, *ElementIndex*)

#### 参数

##### *Path*

路径的名称。例如：

\\NodeName\InTouch

如果 *Path* 参数为空，则 Alarm Tree Viewer 控件查找与指定的元素名匹配的第一个目录树元素。

##### *ElementName*

元素的名称。例如， Group1。

##### *ElementIndex*

元素的索引。

#### 示例

控件名为 AlarmTreeViewCtrl1， StrTag 是消息标记。

```
StrTag =  
    #AlarmTreeViewCtrl1.GetSubElementName("",  
        "Group1", 1);  
  
StrTag =  
    #AlarmTreeViewCtrl1.GetSubElementName("\\Node  
    Name", "Group1", 1);  
  
StrTag =  
    #AlarmTreeViewCtrl1.GetSubElementName("\\InTou  
    ch", "Group1", 1);  
  
StrTag =  
    #AlarmTreeViewCtrl1.GetSubElementName("\\Node  
    Name\InTouch", "Group1", 1);
```



## GetSubElementPath() 方法

根据元素名的索引来获取子元素的路径（到指定的展开级别）。

### 语法

*Object*.GetSubElementPath(*Path*, *ElementName*, *ElementIndex*, *ExpansionLevel*)

### 参数

#### *Path*

路径的名称。例如：

\\NodeName\InTouch

如果 *Path* 参数为空，则 Alarm Tree Viewer 控件查找与指定的元素名匹配的第一个目录树元素。

#### *ElementName*

元素的名称。例如，Group1。

#### *ElementIndex*

元素的索引。

#### *ExpansionLevel*

展开的级别。

### 示例

控件名为 AlarmTreeViewCtrl1，StrTag 是消息标记。

```
StrTag =  
    #AlarmTreeViewCtrl1.GetSubElementPath("",  
        "Group1", 1, 3);  
  
StrTag =  
    #AlarmTreeViewCtrl1.GetSubElementPath("\\Node  
        Name", "Group1", 1, 3);  
  
StrTag =  
    #AlarmTreeViewCtrl1.GetSubElementPath("\\InTou  
        ch", "Group1", 1, 3);  
  
StrTag =  
    #AlarmTreeViewCtrl1.GetSubElementPath("\\Node  
        Name\InTouch", "Group1", 1, 3);
```

## 冻结目录树

您可以使用 **Freeze()** 方法防止使用任何进一步的更改来更新 Alarm Tree Viewer 控件目录树。

### Freeze() 方法

冻结 Alarm Tree Viewer 控件目录树。

#### 语法

*Object*.Freeze(Frozen)

#### 参数

*Frozen*

控制是否可以更新目录树。

1 = 冻结目录树。

0 = 撤销冻结目录树。

#### 示例

Tag1 定义为内存离散标记，控件名为 AlarmTreeViewCtrl1。

```
Tag1 = 1;  
#AlarmTreeViewCtrl1.Freeze(Tag1);
```

## 从所选元素中创建查询字符串

您可以使用 GetAlarmQueryFromSelection() 方法从 AlarmTreeView 控件的所选元素中检索查询字符串。查询字符串随后可以在 Alarm Viewer 控件中动态使用。

### GetAlarmQueryFromSelection() 方法

从 Alarm Tree Viewer 控件的所选元素中返回报警查询字符串。

#### 语法

*Object*.GetAlarmQueryFromSelection()

#### 示例

控件名为 AlarmTreeViewCtrl1，StrTag 是消息标记。例如，StrTag 设置为 \\NodeName\InTouch\GroupA。

```
StrTag =  
#AlarmTreeViewCtrl1.GetAlarmQueryFromSelection();
```

## 运行查询

您可以使用一些方法为 Alarm Tree Viewer 控件运行查询，这些方法可以检索查询收藏夹文件中保存的现有查询，或设置指定新报警供应器集合的字符串。

- SetQueryByName() 方法
- SetQueryByString() 方法

### SetQueryByName() 方法

将当前查询设置为传递的查询名所指定的查询。查询必须在查询收藏夹文件中。

#### 语法

*Object*.SetQueryByName(*QueryName*)

#### 参数

*QueryName*

使用查询收藏夹创建的查询的名称。例如，Turbine Queries。

#### 示例

控件名为 AlarmTreeViewCtrl1。

```
#AlarmTreeViewCtrl1.SetQueryByName("Turbine  
Queries");
```

### SetQueryByString() 方法

将当前查询设置为一个新字符串，以指定新的“报警供应器”集合。

#### 语法

*Object*.SetQueryByString(*NewQuery*)

#### 参数

*NewQuery*

包含报警查询的字符串。例如：

\\MasterNode\InTouch

#### 示例

控件名为 AlarmTreeViewCtrl1。

```
#AlarmTreeViewCtrl1.SetQueryByString("\\MasterN  
ode\InTouch");
```

## 使用方法与属性时的错误处理

所有 Alarm Tree Viewer 控件错误消息都发送到 Logger。如果将 Alarm Tree Viewer 控件配置成在无提示模式下运行，则不显示运行时错误。

## 使用 Alarm Tree Viewer 控件 ActiveX 事件触发脚本

您可以将 QuickScript 指定给 Alarm Tree Viewer 控件事件（如鼠标单击或双击）。该事件发生时，此 QuickScript 便会运行。

Alarm Tree Viewer 控件支持以下事件：

- Click
- DoubleClick
- ShutDown
- StartUp

Click 事件有一个参数 ClicknElementID，它可以确定在运行时所单击的目录树中的元素。

DoubleClick 事件有一个参数 DoubleClicknElementID，它可以确定在运行时所双击的目录树中的元素。

对于 Click 与 DoubleClick 事件，对于“全部供应器”节点，返回的 ElementID 是 -1。

---

**备注** Alarm Tree Viewer 控件从 StartUp 事件调用方法时会忽略用户界面方法，原因在于控件尚不可见。这些方法包括：  
AboutBox()、CheckElementMembership()、Freeze()、  
GetAlarmQueryFromSelection()、GetElementCount()、  
GetElementName()、GetElementPath()、  
GetSelectedElementCount()、GetSelectedElementName()、  
GetSelectedElementPath()、GetSubElementCount()、  
GetSubElementName()、GetSubElementPath() 以及 Refresh()。

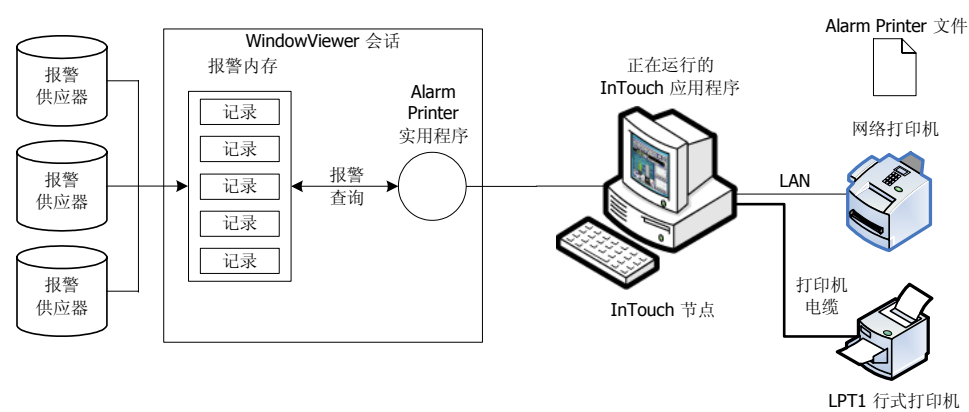
---

如需有关编写 ActiveX 事件脚本的详细信息，请参阅 *InTouch® HMI 脚本与逻辑指南* 中的第 8 章“编写 ActiveX 控件脚本”。

# 第 8 章

## 打印报警

InTouch Alarm Printer 实用程序可以打印来自多个节点的报警。您可以使用专门的行式打印机或网络打印机来按照事件打印报警内存中存储的报警记录。同样，您也可以使用 Alarm Printer 将报警记录保存到文件。



您可以将“分布式报警”系统配置成发生特定的事件时在行式打印机上打印它们。典型情况下，您会立即打印报警，以便在万一发生灾难性故障时将信息记录下来。通常您会使用点阵打印机，它通过串行或并行端口直接连接到运行 InTouch 应用程序的计算机上。由于 Windows 网络打印机与激光打印机在实际打印页面之前将整个页面保存在内存中，因此它们一般不适合灾难性事件的数据采集。

## 配置报警打印与记录

您可以运行 Alarm Printer 的多个实例。Alarm Printer 的每个实例都必须配置成打印到不同的打印机，并且必须使用单独的报警查询来配置。

您可以配置 Alarm Printer 各个单独的实例，以打印特定优先级范围内的报警。例如，一个 Alarm Printer 实例仅打印高优先级的报警，而另一个实例则仅打印低优先级的报警。与此类似，您也可以使用 Alarm Printer 的一个实例来打印工厂某个区域的报警，而使用另外一个实例来打印另一个区域的报警。

您可以将 Alarm Printer 配置保存到扩展名为 .alc 的文件中。您可以根据需要创建任意多个配置文件。对于每个正在运行的 Alarm Printer 实例，Alarm Printer 均使用单独的配置文件。

## 配置打印机设置

您必须指定供 Alarm Printer 使用的打印机。同样，您必须选择是否在报警发生时打印它们。

打印报警时应该使用专用打印机。

打印机可以连接到本地机器或网络上。任何 Windows 打印机都可以用作 Alarm Printer 的输出设备。


### 要配置打印机设置

- 1 打开 Alarm Printer 实用程序。执行以下操作：
  - a 在 WindowMaker 工具视图中，展开**应用程序**。
  - b 双击 **Alarm Printer**。
- 2 在菜单栏上，单击**配置**。此时出现**配置设置**对话框。
- 3 单击**打印**选项卡。



- 4 在**打印到**区域中，选择与 Alarm Printer 的连接。
  - 单击**无**不使用打印机。
  - 单击**LPT1-3**以使用某个打印机，它通过并行端口连接到正在运行 InTouch 应用程序的计算机。
  - 单击**COM1-4**以使用某个打印机，它通过串行端口连接到正在运行 InTouch 应用程序的计算机。单击**端口配置**以显示**COM 属性**对话框，并更改指定给所选 COM 端口的缺省值。
  - 单击**打印机**以使用某个打印机，它通过网络连接到正在运行 InTouch 应用程序的计算机。在方框中，输入打印机的名称，或单击**浏览**以选择可用的打印机。

**备注** 如果没有出现要使用的打印机，请使用“Windows 添加打印机”向导添加打印机。

- 5 选择**打印时删除尾部空格**复选框，以防止打印机打印空行或空页。
-  6 选择**启用打印**复选框以打印报警。
- 7 选择**禁用实时报警打印**复选框，以防止 Alarm Printer 在报警发生时打印它们。
- 8 单击**确定**。

## 配置要打印的报警

Alarm Printer 查询报警内存来选择要打印或保存到日志文件的记录。查询基于以下条件从内部报警内存中选择记录：

- 报警优先级
- 当前报警状态（未确认 / 确认）
- 报警组成员关系

每个报警都有一个指定的优先级编号，代表报警的严重程度。报警优先级的范围是从 1 到 999。最严重的报警所指定的优先级是 1。最不严重的报警所指定的优先级是 999。

如果网络或打印机连接失败，Alarm Printer 不会重新打印所有的报警。Alarm Printer 仅打印连接失败之前尚未打印的那些报警。

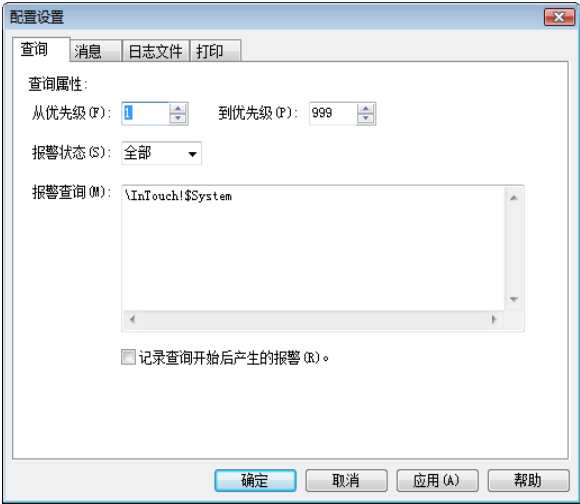
### 要配置所要打印的报警

- 1 打开 Alarm Printer 实用程序。执行以下操作：
  - a 在 WindowMaker 工具视图中，展开**应用程序**。
  - b 双击**Alarm Printer**。



- 2 在菜单栏上，单击**配置**。此时出现**配置设置**对话框。

3 单击**查询**选项卡。



4 在**从优先级**框中，输入最高优先级的报警值（1 到 999）。

5 在**到优先级**框中，输入最低优先级的报警值（1 到 999）。

6 在**报警状态**列表中，单击报警状态。

单击	显示
全部	所有报警。
确认	仅限已确认的报警。
未确认	仅限未确认的报警。

7 在**报警查询**框中，输入一个或多个报警查询。您可以指定一个或多个报警供应器与组。使用空格分隔多个查询。

8 选择**记录查询开始后产生的报警**复选框，以便仅包含查询开始之后发生的报警。此时 Alarm Printer 忽略报警内存中的报警记录，这些报警在 Alarm Printer 开始查询之前便已经触发。

9 单击**确定**。



## 配置打印与文件输出的格式

选择的每个选项在打印输出中都显示为一个单独的字段。包含数据的字段超出指定的最大字段长度时，将按照这些限制进行截断。

### 要配置报警打印与文件输出的格式

- 1 打开 Alarm Printer 实用程序。执行以下操作：
  - a 在 WindowMaker 工具视图中，展开应用程序。
  - b 双击 **Alarm Printer**。
- 2 在菜单栏上，单击**配置**。此时出现**配置设置**对话框。
- 3 单击**消息**选项卡。



- 4 在日期 / 时间区域中，选择**日期**复选框，然后从列表中选择一种日期格式。列出的日期格式包含以下组成部分：

选项	描述
DD	两位日期值 (01-31)
MM	两位月份值 (01-12)
YY	年份的后两位
YYYY	四位年份值
MMM	三个英文字符的月份缩写

- 5 选择时间复选框，然后从列表中选择一种时间格式。列出的格式包含时间的以下组成部分：

选项	描述
AP	选择“上午 / 下午”时间格式。例如，下午 3:00 显示为 3:00 PM。未指定此项的时间缺省使用 24 小时军用时间格式。例如，下午 3:00 显示为 15:00。
HH	发生报警 / 事件时的两位小时值（00-23 或 01-12）。
MM	发生报警 / 事件时的两位分钟值（00-59）。
SS	发生报警 / 事件时的两位秒值（00-59）。
SSS	发生报警 / 事件时的三位毫秒值。

- 6 根据报警发生的时间选择报警出现在报警记录中的顺序：

选项	描述
OAT	（原始报警时间）发生报警时的日期 / 时间标签。
LCT	（上次更改时间）报警实例最近一次状态改变的日期 / 时间标签：报警发生、子状态改变、返回正常，或者是确认。
LCT 但确认为 OAT	（“上次更改时间”，但确认时使用“原始报警时间”）报警未确认时使用上次更改时间，确认报警之后使用原始报警时间。

- 7 选择要打印的报警信息。

选项	描述
报警状态	打印报警状态。例如，UnAck、Ack。
报警类	打印报警类。例如，VALUE、DEV、ROC。
报警类型	打印报警类型。例如，HIHI、LO、MAJDEV。
优先级	打印报警优先级（1-999）。
7.1 默认值（按钮）	选择 InTouch 7.1 版 Alarm Printer 实用程序的缺省设置。

选项	描述
<b>7.11 默认值（按钮）</b>	选择 InTouch 7.11 版 Alarm Printer 实用程序的缺省设置。
<b>删除尾部空格</b>	实际字段值的长度小于为该字段配置的值时，从打印的字段中删除额外的尾部空格。
<b>最小列间距</b>	减少各个列的间距，使打印的页面上可以显示更多的字段。
<b>报警名</b>	打印报警名（标记）。在 <b>长度</b> 框中，输入报警名的字符数（最多 64 个字符）。
<b>组名</b>	打印报警组的名称。在 <b>长度</b> 框中，输入报警组名的字符数（最多 64 个字符）。
<b>报警供应器</b>	打印报警供应器的名称。在 <b>长度</b> 框中，输入报警供应器名的字符数（最多 64 个字符）。
<b>报警值</b>	打印标记的值。在 <b>长度</b> 框中，输入报警值的字符数（最多 32 个字符）。
<b>极限</b>	打印标记的报警限。在 <b>长度</b> 框中，输入允许报警限使用的字符数（最多 32 个字符）。此数字应该足够大，以提供所需级别的精度。
<b>操作员节点</b>	打印与报警条件关联的操作员节点。在 <b>长度</b> 框中，输入允许操作员节点使用的字符数（最多 64 个字符）。 在“终端服务”环境中，这是相应的操作员建立“终端服务”会话的客户端计算机的名称。如果无法检索节点名，则使用节点的 IP 地址代替。
<b>操作员名称</b>	打印与报警条件关联的操作员名。在 <b>长度</b> 框中，输入允许操作员名使用的字符数（最多 16 个字符）。
<b>注释</b>	打印与标记关联的报警注释。在 <b>长度</b> 框中，输入允许注释使用的字符数（最多 131 个字符）。

选项	描述
用户 1	打印与报警对应的“用户自定义数字 1”的数值。
用户 2	打印与报警对应的“用户自定义数字 2”的数值。
用户 3	打印与报警关联的用户自定义字符串属性的字符串值。最多 131 个字符。

8 单击应用。

## 配置报警的日志文件

您可以将报警记录保存到文件中。缺省条件下，Alarm Printer 基于以下命名惯例给日志文件指定名称：

YYMMDDHH.ALG

记号	描述
YY	创建日志文件时的年份的后两位数字。
MM	创建日志文件时的两位数字的月份 (01-12)。
DD	创建日志文件时的两位数字的日期 (01-31)。
HH	创建日志文件时的两位数字的小时值 (00-23)。

### 要配置报警记录日志

- 1 打开 Alarm Printer 实用程序。执行以下操作：
  - a 在 WindowMaker 工具视图中，展开应用程序。
  - b 双击 Alarm Printer。



- 2 在菜单栏上，单击配置。此时出现配置设置对话框。

### 3 单击日志文件选项卡。



- 4 选择**启用报警文件记录**复选框，以便将报警记录保存到日志文件。

- 5 在**目录**框中，输入要保存报警日志文件的路径，或浏览到某个文件夹位置。

- 6 在**循环文件名小时数**框中，输入要在单独的日志文件中保存多少个小时的报警记录。

有效输入为 1 到 24。如果 InTouch 应用程序持续运行，请选择一个适当的文件记录间隔，以便创建一组时间长度相等的日志文件。例如，将**循环文件名小时数**设置为 6 时，可以创建 4 个时间长度相等的日志文件。

- 7 在**开始小时值**框中，输入每天开始将报警记录保存到文件的起始小时值。

有效输入为 0 到 23。

例如，某个炼油厂按照每天三班的方式运转。第一班从 06:00 开始。管理人员希望记录每一班的报警记录。要做到这一点，请给**循环文件名小时数**选项输入 8。给**开始小时值** (0-23) 选项输入 6。Alarm Printer 从 06:00 到 14:00 创建一个日志文件，从 14:00 到 22:00 创建另一个，从 22:00 到 06:00 创建第三个。

- 8 在**保留日志文件时间**框中，输入要保留日志文件的天数。

Alarm Printer 将日志记录保存到指定的天数（加上当天）。Alarm Printer 删除超过保留期限的日志文件。要无限期保存日志文件，请输入 0。

- 9 在**日志文件扩展名**框中，接受缺省的 ALG 文件扩展名，或者给日志文件指定其它三个英文字符的扩展名。

如果使用 .csv 扩展名，则可以将日志文件直接导入 Excel 或“记事本”。

- 10 要删除日志文件中每个项目末尾的空格，请选择**日志条目中删除尾部空格**复选框。

您也可以指定一个字段分隔符，放到日志文件中每个记录的末尾。

- 11 选择**按原始列排序**复选框，以便让日志文件记录保持报警显示中的相同顺序。

- 12 单击**应用**。

## 保存与加载配置文件

从 WindowMaker 中启动 Alarm Printer 时，**Alarm Printer** 对话框显示缺省的配置设置。这些配置设置保存在报警配置文件 (.alc) 中。

您可以将打印机配置设置保存到某个文件，在打印报警之前可以加载该文件。

如果在运行时从命令提示符中或通过双击 \*.alc 文件名来打开特定的报警配置文件，则可以显示它。

### 要创建新的 Alarm Printer 配置文件

- 1 打开 Alarm Printer 实用程序。执行以下操作：

- a 在 WindowMaker 工具视图中，展开**应用程序**。

- b 双击 **Alarm Printer**。



- 2 在**文件**菜单上，单击**新建**以显示带缺省值的 Alarm Printer。

- 3 在菜单栏上，单击**配置**。此时出现**配置设置**对话框。

- 4 配置报警设置。



- 5 在**文件**菜单上，选择**保存**。

### 要编辑现有的 Alarm Printer 配置文件



- 1 在**文件**菜单上，选择**打开**。

- 2 选择要编辑的 Alarm Printer 配置文件。

- 3 编辑该文件。



- 4 在**文件**菜单上，选择**保存**。选择**另存为**将更改保存到新的文件，而不更改现有的文件。

## 打印报警

每个查询记录当前打开的 Alarm Printer 配置文件 (.alc) 中指定的所有报警。如果未指定文件，则使用配置 Alarm Printer 期间当前所选的设置。

您可以使用 Alarm Printer 运行多个查询。每个查询都使用不同的参数，并且与单独的 Alarm Printer 实例关联。如果 Alarm Printer 的两个实例正在运行相同的查询，则查询结果将完全相同。

Alarm Printer 正在运行时，可以手工启动或停止查询。务必确保启用了打印。

### 要启动报警查询



- ◆ 在**查询**菜单上，单击**开始 / 停止**。

### 要停止报警查询




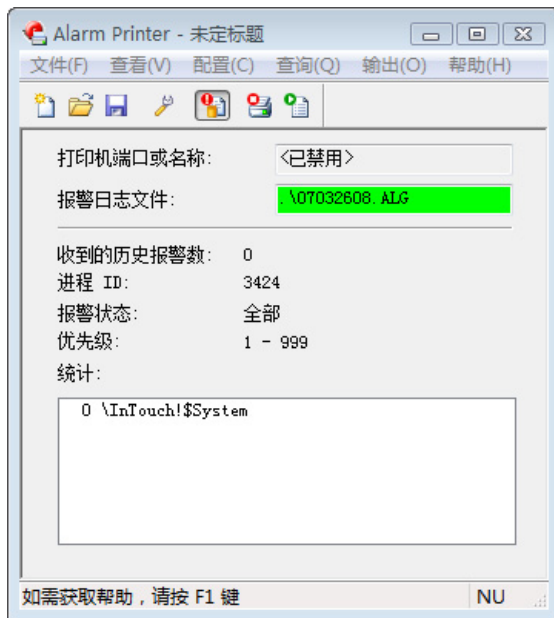
- ◆ 在**查询**菜单上，单击**开始 / 停止**。

## 将报警记录到文件

您可以使用 Alarm Printer 将报警记录保存到文件中。您应该已经从**配置设置**对话框中配置好记录功能。

### 要将报警记录到文件

- 1 打开 Alarm Printer 实用程序。
  - a 在 WindowMaker 工具视图中，展开**应用程序**。
  - b 双击 **Alarm Printer**。
- 2 根据需要配置查询以采集要记录的报警记录。
-  3 单击**文件记录**按钮。
- 4 运行报警查询。



此时由查询采集的报警记录写入所配置的日志文件中。



## 使用特定的配置启动 Alarm Printer

通过在批处理文件中使用以下命令，可以在启动系统时自动启动 Alarm Printer 并打开特定的文件。

```
ALMPRT.EXE MYQUERY.ALC
```

其中，MYQUERY.ALC 是打开的 Alarm Printer 配置文件的名。指定 .exe 文件扩展名属于可选项。

确保批处理文件切换到安装 InTouch HMI 的文件夹。

要防止因系统意外关机并重新启动而导致丢失任何查询数据，可以通过从批处理文件中运行以下命令来自动启动 Alarm Printer 并自动运行特定的查询。

```
ALMPRT.EXE -q MYQUERY.ALC
```

通过在命令中使用 ñq，查询会在系统启动时自动运行。

## 使用脚本控制 Alarm Printer

您可以在脚本中使用 Alarm Printer 函数来控制报警打印。

Alarm Printer 函数返回一个整型错误码，指出函数是否成功完成。下表显示各个错误码。

错误码	错误消息
0	成功
1	实例未找到或未在运行
2	接口未初始化
3	无法访问虚拟内存
4	错误码无效
5	已在运行的实例太多
6	结果字符串太长
7	传递给函数的实例索引无效
8	无法将消息发送到 Alarm Printer 应用程序
9	未等到 Alarm Printer 应用程序的响应
20	“到优先级”必须大于或等于“从优先级”
21	优先级值无效
22	报警状态无效
23	由于查询正在运行，无法执行命令

错误码	错误消息
24	查询字符串无效
25	查询处理状态无效
26	打印状态选择器无效
27	Alarm Printer 窗口收到的命令无法识别
28	查询无法启动

## 停止与启动 Alarm Printer 实例或查询

使用以下函数启动与停止 Alarm Printer 实例与报警查询。

- APUStartInstance() 函数
- APUStartQuery() 函数
- APUSStopInstance() 函数
- APUSStopQuery() 函数

### APUStartInstance() 函数

使用配置文件中指定的值在最小化状态下启动 Alarm Printer 的实例。

类别  
View

语法

```
[Result=] APUStartInstance (sFilePath, iTagInstance) ;
```

参数

*sFilePath*

配置文件的完整路径（输入字符串）。

*iTagInstance*

整型标记。如果函数成功执行，则给它返回实例号。

附注

您最多可以启动十六个 Alarm Printer 实例。此函数将实例号 (0 - 15) 写入 iTagInstance 参数。实例号在启动新的 Alarm Printer 实例时进行递增。

此实例号可以由其它 Alarm Printer 函数用于识别 Alarm Printer 实例。

此函数返回 0；如果发生错误，则返回错误码。

Alarm Printer 程序不自动开始处理报警内存中的报警。使用 APUStartQuery() 函数开始处理该实例报警内存中的数据。

### 示例

```
Status =  
    APUStartInstance("c:\MyAlarmCfg\Area1Alarms.alc",  
    Inst);
```

### 另请参阅

APUStartQuery(), APUStopInstance(), APUStopQuery()

## APUStartQuery() 函数

设置要从报警内存中处理的记录的日期与时间限制，然后启动查询。

类别

View

### 语法

```
[Result=] APUStartQuery(iInstance,iYear,iMonth,iDay,  
    iHour,iMinute);
```

### 参数

*iInstance*

Alarm Printer 的实例（0 到 15）

*iYear*

年份数字。

*iMonth*

月份数字。

*iDay*

日期值。

*iHour*

小时值。

*iMinute*

分钟值。

### 附注

如果在查询已经运行时试图启动另一个查询，则会发生错误。

如果将所有的日期与时间值都设置为 0，则打印所有报警。这是由于 0 被解释为 1900 年 1 月 1 日午夜。指定的日期与时间采用本地时间格式。年份值为 -1 时，将日期设置为处理命令时的当前时间。

返回整型错误码。

### 示例

```
Status = APUStartQuery(Inst,2007,4,16,22,12);
```

### 另请参阅

APUStartInstance(), APUStopInstance(), APUStopQuery()

### APUStopInstance() 函数

停止指定的 Alarm Printer 实例。此时将停止进一步添加任何要打印的记录，终止当前正在执行的任何打印查询，并关闭程序的实例。

类别  
View

语法

```
[Result=] APUStopInstance(iInstance) ;
```

参数

*iInstance*

Alarm Printer 的实例（0 到 15）。

附注

返回整型错误码。

示例

```
Status = APUStopInstance(5) ;
```

另请参阅

APUStartInstance(), APUStartQuery(), APUStopQuery()

### APUStopQuery() 函数

请求指定的实例停止运行查询。应用程序保持运行状态，但是不处理任何查询。调用 APUStartQuery() 可以使实例启动查询。

类别  
View

语法

```
[Result=] APUStopQuery(iInstance) ;
```

参数

*iInstance*

Alarm Printer 的实例（0 到 15）。

附注

返回整型错误码。

示例

```
Status = APUStopQuery(5) ;
```

另请参阅

APUStartInstance(), APUStartQuery(), APUStopInstance()

## 查询报警查询信息

使用以下函数根据 Alarm Printer 配置文件中设置的标记优先级与数值来检索查询参数。

- APUGetAlarmGroupText() 函数
- APUGetQueryFromPriority() 函数
- APUGetQueryToPriority() 函数
- APUGetConfigurationFilePath() 函数
- APUGetPrinterJobCount() 函数
- APUGetQueryAlarmState() 函数
- APUGetQueryProcessingState() 函数

### APUGetAlarmGroupText() 函数

获取 “报警查询” 报警组文本。

类别

View

语法

```
[Result=] APUGetAlarmGroupText (iInstance,sTagGroup) ;
```

参数

*iInstance*

Alarm Printer 的实例（0 到 15）。

*sTagGroup*

文本 - 报警组

附注

初始报警组文本从指定的 Alarm Printer 实例正在使用的 .alc 配置文件中读取。报警组文本作为 sTagGroup 参数传递到 InTouch 消息标记中。

返回整型错误码。

示例

TagGroup 消息标记在运行函数之后可能包含以下值：  
 \intouch!\$system

```
Status = APUGetAlarmGroupText(Inst,TagGroup);
```

另请参阅

APUGetConfigurationFilePath(), APUGetPrinterJobCount(), APUGetQueryAlarmState(), APUGetQueryFromPriority(), APUGetQueryProcessingState(), APUGetQueryToPriority()

### APUGetQueryFromPriority() 函数

获取查询的“从优先级”值。

#### 类别

View

#### 语法

```
[Result=] APUGetQueryFromPriority(iInstance, iTagPriority);
```

#### 参数

*iInstance*

Alarm Printer 的实例（0 到 15）。

*iTagPriority*

整型标记，用于接收“从优先级”值。

#### 附注

初始优先级从指定的 Alarm Printer 实例正在使用的 .alc 配置文件中读取。“从优先级”值作为 *iTagPriority* 参数传递给某个 InTouch 整型标记。

返回整型错误码。

#### 示例

在本例中，**FromPri** 是包含“从优先级”值（如 1）的整型标记。

```
Status = APUGetQueryFromPriority(Inst, FromPri);
```

#### 另请参阅

APUGetAlarmGroupText(),  
APUGetConfigurationFilePath(), APUGetPrinterJobCount(),  
APUGetQueryAlarmState(),  
APUGetQueryProcessingState(), APUGetQueryToPriority()

## APUGetQueryToPriority() 函数

获取查询的“到优先级”。

### 类别

View

### 语法

```
[Result=] APUGetQueryToPriority(iInstance,iPriority );
```

### 参数

*iInstance*

Alarm Printer 的实例（0 到 15）。

*iPriority*

整型标记，用于接收“到优先级”值。

### 附注

其它查询不得与包含 APUGetQueryToPriority() 函数的脚本同时运行。“到优先级”值写入函数的 iPriority 参数，它是一个整型标记。

返回整型错误码。

### 示例

ToPri 整型标记接收“到优先级”值，如 999。

```
Status = APUGetQueryToPriority(Inst, ToPri);
```

### 另请参阅

APUGetAlarmGroupText(),  
APUGetConfigurationFilePath(), APUGetPrinterJobCount(),  
APUGetQueryAlarmState(), APUGetQueryFromPriority(),  
APUGetQueryProcessingState()

### APUGetConfigurationFilePath() 函数

返回用于查询的 .alc 配置文件的完整文件路径。

类别  
View

语法

```
[Result=] APUGetConfigurationFilePath(iInstance,  
                                         sTagFilePath) ;
```

参数

*iInstance*

Alarm Printer 的实例（0 到 15）。

*sTagFilePath*

消息标记，用于检索 Alarm Printer 实例正在使用的配置文件的文件路径名。

附注

文件路径文本返回给一个消息标记，该标记可以指定为此函数的 *sTagFilePath* 参数。

返回整型错误码。

示例

CfgFilePath 消息标记接收特定配置文件的文件路径，该配置文件与 Inst 整型标记中包含的 Alarm Printer 实例关联。例如：

c:\MyAlarmCfg\Area1Alarms.alc

```
Status = APUGetConfigurationFilePath(Inst, CfgFilePath);
```

另请参阅

APUGetAlarmGroupText(), APUGetPrinterJobCount(),  
APUGetQueryAlarmState(), APUGetQueryFromPriority(),  
APUGetQueryProcessingState(), APUGetQueryToPriority()



## APUGetPrinterJobCount() 函数

为此实例使用的打印机返回最新的作业计数这个 Windows 打印机状态。

### 类别

View

### 语法

```
[Result=] APUGetPrinterJobCount(iInstance, iTagCount);
```

### 参数

#### *iInstance*

Alarm Printer 的实例（0 到 15）。

#### *iTagCount*

整型标记，用于接收计数值。

### 附注

此函数在 *iTagCount* 参数中返回计数值，该参数是一个整型标记。

除非查询正在运行，否则结果不是最新的。除非报警已经打印，否则结果不是最新的 - 作业计数通常在初次打开打印机时更新，然后在每打印一行报警时更新。

返回的作业计数值仅对 Windows 打印机有效，对于与并行或串行端口关联的打印机则没有多少意义。

返回整型错误码。

### 示例

PJCount 是一个整型标记，用于接收指定的实例的计数值。

```
Status = APUGetPrinterJobCount(Inst, PJCount);
```

### 另请参阅

APUGetAlarmGroupText(),  
APUGetConfigurationFilePath(),  
APUGetQueryAlarmState(), APUGetQueryFromPriority(),  
APUGetQueryProcessingState(), APUGetQueryToPriority()

### APUGetQueryAlarmState() 函数

返回查询的报警状态。

#### 类别

View

#### 语法

```
[Result=] APUGetQueryAlarmState(iInstance, iTagState);
```

#### 参数

*iInstance*

Alarm Printer 的实例（0 到 15）。

*iTagState*

整型标记，用于接收同指定的实例关联的查询的报警状态。值的含义如下：

0 = 全部

1 = 已确认

2 = 未确认

#### 附注

初始报警状态从 .alc 文件中读取。此函数在 *iTagCount* 参数中返回它，该参数是一个整型标记。

返回整型错误码。

#### 示例

AlmState 是包含 0、1 或 2 的整型标记。

```
Status = APUGetQueryAlarmState(Inst, AlmState);
```

#### 另请参阅

APUGetAlarmGroupText(),  
APUGetConfigurationFilePath(), APUGetPrinterJobCount(),  
APUGetQueryFromPriority(),  
APUGetQueryProcessingState(), APUGetQueryToPriority()

## APUGetQueryProcessingState() 函数

返回报警查询的处理状态。

### 类别

View

### 语法

```
[Result=] APUGetQueryProcessingState (iInstance,  
iTagState) ;
```

### 参数

*iInstance*

Alarm Printer 的实例（0 到 15）。

*iTagState*

整型标记，用于从函数中接收处理状态。它的含义如下：

0 = 停止

1 = 启动

### 附注

此函数给 iTagState 参数返回一个整数值，该参数是一个整型标记。

返回整型错误码。

### 示例

ProcState 是一个整型标记，查询不在运行时设置为 0，查询正在运行时设置为 1。

```
Status = APUGetQueryProcessingState (Inst,  
ProcState) ;
```

### 另请参阅

APUGetAlarmGroupText(),  
APUGetConfigurationFilePath(), APUGetPrinterJobCount(),  
APUGetQueryAlarmState(), APUGetQueryFromPriority(),  
APUGetQueryToPriority()

## 查询实例信息

使用以下函数返回 Alarm Printer 实例的当前状态。

- APUFindAlarmGroupInstance() 函数
- APUFindFileInstance() 函数
- APUFindPrinterInstance() 函数
- APUGetInstanceCount() 函数
- APUIsInstanceUsed() 函数

### APUFindAlarmGroupInstance() 函数

返回使用指定的报警组字符串的第一个 Alarm Printer 实例。

#### 类别

View

#### 语法

```
[Result=] APUFindAlarmGroupInstance(sGroup,  
                                     iInstance);
```

#### 参数

*sGroup*

要在各实例中查找的报警组的名称。

*iInstance*

整型标记，对于查找到的使用指定组名的实例，此标记接收它的值。

#### 附注

此函数将实例号作为 *iInstance* 参数返回给某个整型标记。初始报警组字符串从 .alc 文件中读取。如果未找到任何实例，则函数返回 1 作为错误码（没有可用的实例），并将 0 写入整型标记参数。

返回整型错误码。

#### 示例

FoundInstance 是一个整型标记，用于接收所找到的第一个实例的编号，该实例将 \$System 用作其查询。

```
Status = APUFindAlarmGroupInstance("$System", FoundInstance);
```

#### 另请参阅

APUFindFileInstance(), APUFindPrinterInstance(),  
APUGetInstanceCount(), APUIsInstanceUsed()

## APUFindFileInstance() 函数

查找使用指定的 .alc 配置文件的第一个 Alarm Printer 实例。

类别

View

语法

```
[Result=] APUFindFileInstance(sFilePath,iInstance);
```

参数

*sFilePath*

要查找的实例的 .alc 配置文件路径。

*iInstance*

整型标记，用于接收实例编号。

附注

此函数将实例号作为 *iInstance* 参数返回给某个整型标记。使用此函数获取所需的 Alarm Printer 实例。匹配文件路径字符串时不区分大小写。

如果未找到任何实例，则函数返回 1 作为错误码（没有可用的实例），并将 0 写入整型标记参数。

返回整型错误码。

示例

InstFound 是一个整型标记，它接收使用

c:\MyAlarmCfg\Area1Alarms.alc 配置文件的第一个实例的编号。

```
Status = APUFindFileInstance("c:\MyAlarmCfg\Area1Alarms.alc",  
    InstFound);
```

另请参阅

APUFindAlarmGroupInstance(), APUFindPrinterInstance(),  
APUGetInstanceCount(), APUIsInstanceUsed()

### APUFindPrinterInstance() 函数

查找使用指定的打印机名称或端口的第一个 Alarm Printer 实例。

**类别**  
View

**语法**

```
[Result=] APUFindPrinterInstance(sPrinter,iInstance);
```

**参数**

*sPrinter*

要查找的实例的打印机名称。

*iInstance*

整型标记，用于接收实例编号。

**附注**

此函数将实例号作为 *iInstance* 参数返回给某个整型标记。使用此函数获取所需的 Alarm Printer 实例。打印机名存储在 .alc 文件中，并可以从中读取。匹配打印机名字符串时不区分大小写。如果未找到任何实例，则函数返回 1 作为错误码（没有可用的实例），并将 0 写入整型标记参数。

返回整型错误码。

**示例**

FoundInst 是一个整型标记，对于在关联的 .alc 文件中将 LPT1 用作打印机名的第一个实例，它用于接收该实例的编号。

```
Status = APUFindPrinterInstance("LPT1",  
    FoundInst);
```

**另请参阅**

APUFindAlarmGroupInstance(), APUFindFileInstance(),  
APUGetInstanceCount(), APUIsInstanceUsed()

## APUGetInstanceCount() 函数

返回正在运行的 Alarm Printer 实例的数量，最多 16 个。

### 类别

View

### 语法

```
[Result=] APUGetInstanceCount (iCount) ;
```

### 参数

*iCount*

整型标记，用于接收实例编号。

### 附注

此函数将实例数量作为参数返回给某个整型标记。前十六个同时运行的实例之外的任何其它实例都不是动态控制的，也无法获取其状态。

返回整型错误码。

### 示例

ICount 是一个整型标记。运行时的值为 7 时，表示当前有七个 Alarm Printer 实例正在运行。

```
Status = APUGetInstanceCount (iCount) ;
```

### 另请参阅

APUFindAlarmGroupInstance(), APUFindFileInstance(),  
APUFindPrinterInstance(), APUIsInstanceUsed()

### APUIsInstanceUsed() 函数

返回一个离散值，指出当前是否正在使用某个实例。

类别  
View

#### 语法

```
[Result=] APUIsInstanceUsed(iInstance) ;
```

#### 参数

*iInstance*

Alarm Printer 实例的编号（0 到 15）。

#### 附注

结果为 0 或 1：

0 = 实例未使用。

1 = 实例在使用。

#### 示例

InUse 是一个离散标记，如果实例 5 正在使用，则该标记设置为真 (1)；如果实例 5 不在使用，则设置为假 (0)。

```
InUse = APUIsInstanceUsed(5) ;
```

#### 另请参阅

APUFindAlarmGroupInstance(), APUFindFileInstance(),  
APUFindPrinterInstance(), APUGetInstanceCount()



## 查询打印机信息

使用以下函数返回 Alarm Printer 的状态。

- APUGetPrinterName() 函数
- APUGetPrinterStatus() 函数

### APUGetPrinterName() 函数

返回此实例使用的打印机的 Windows 打印机名或端口名。

类别

View

语法

```
[Result=] APUGetPrinterName(iInstance, sTagPrinter) ;
```

参数

*iInstance*

Alarm Printer 实例编号（0 到 15）。

*sTagPrinter*

消息标记，它接收与指定实例关联的配置的打印机名或端口名。

附注

如果未给某个实例配置打印机，则此函数返回 **NONE**。打印机名存储在 .alc 文件中，并可以从中读取。此函数将打印机名或端口名作为 **sTagPrinter** 参数返回给某个消息标记。

返回整型错误码。

示例

**PrtName** 是一个消息标记，它接收与 Alarm Printer 实例 3 关联的配置的打印机名或端口名。

```
Status = APUGetPrinterName(3, PrtName) ;
```

另请参阅

APUGetPrinterStatus()

## APUGetPrinterStatus() 函数

返回此实例使用的 Windows 打印机的最新状态。

### 类别

View

### 语法

```
[Result=] APUGetPrinterStatus(iInstance,iSelector,  
                                iTagStatus);
```

### 参数

#### *iInstance*

Alarm Printer 的实例（0 到 15）。

#### *iSelector*

整数值，指定如下内容：

0 = 获取 “Alarm Printer 错误” 的状态

1 = 获取 “Alarm Printer 缺纸” 的状态

2 = 获取 “Alarm Printer 离线” 的状态

3 = 获取 “Alarm Printer 溢出” 的状态

#### *iTagStatus*

整型或实型标记，它接收某个打印机的状态，该打印机与指定的实例号以及 *iSelector* 参数所选择的类型关联。

### 附注

此函数将打印机状态作为 *iTagStatus* 参数返回给某个整型或实型标记。除非查询正在运行并且已经打印报警，否则结果不是最新的。状态通常在初次打开打印机时更新，然后在每打印一行报警时更新。

此状态信息是根据 Microsoft 或 Windows 的驱动程序标准向打印机查询的。并非所有的打印机制造商都遵循这些标准。因此，并非所有打印机都会返回状态信息。

返回整型错误码。

### 示例

PrtStat 是一个整型标记，它从同实例 5 关联的打印机接收 “打印机离线” 状态。

```
Status = APUGetPrinterStatus(5, 2, PrtStat);
```

### 另请参阅

APUGetPrinterName()

## 设置报警查询信息

使用以下函数设置 Alarm Printer 查询中使用的参数。

- APUSetAlarmGroupText() 函数
- APUSetQueryAlarmState() 函数
- APUSetQueryFromPriority() 函数
- APUSetQueryToPriority() 函数
- APUSetTimeoutValues() 函数

### APUSetAlarmGroupText() 函数

设置“报警查询”报警组文本。

**类别**

View

**语法**

```
[Result=] APUSetAlarmGroupText (iInstance,sGroup) ;
```

**参数**

*iInstance*

Alarm Printer 的实例（0 到 15）。

*sGroup*

报警组文本。

**附注**

查询不在运行时此函数才能成功。

返回整型错误码。

**示例**

本例将 Alarm Printer 实例 1 的查询设置为  
\InTouch!GroupA。

```
Status = APUSetAlarmGroupText (1,  
    "\intouch!GroupA") ;
```

**另请参阅**

APUSetQueryAlarmState(), APUSetQueryFromPriority(),  
APUSetQueryToPriority(), APUSetTimeoutValues()

## APUSetQueryAlarmState() 函数

设置查询的报警状态。

### 类别

View

### 语法

```
[Result=] APUSetQueryAlarmState(iInstance, iState);
```

### 参数

*iInstance*

Alarm Printer 的实例（0 到 15）。

*iState*

具有以下可能值的整数：

0 = 全部

1 = 已确认

2 = 未确认

### 附注

运行使用 APUSetQueryAlarmState() 函数的脚本时，无法同时运行查询。

返回整型错误码。

### 示例

本例设置 Alarm Printer 实例 3 的查询，使其仅查询已确认的报警。

```
Status = APUSetQueryAlarmState(3, 1);
```

### 另请参阅

APUSetAlarmGroupText(), APUSetQueryFromPriority(),  
APUSetQueryToPriority(), APUSetTimeoutValues()

## APUSetQueryFromPriority() 函数

设置报警查询的下边界或 “从优先级”。

### 类别

View

### 语法

```
[Result=]  
APUSetQueryFromPriority(iInstance,iPriority) ;
```

### 参数

#### *iInstance*

Alarm Printer 的实例（0 到 15）。

#### *iPriority*

表示 “从优先级” 的整数值（1 到 999）。

### 附注

查询不在运行时，此函数才能成功。

返回整型错误码。

### 示例

本例设置某个查询的 “从优先级” 值，该查询与 Inst 整型标记的实例值以及 FromPri 整型标记的值关联。

```
Status = APUSetQueryFromPriority(Inst, FromPri);
```

### 另请参阅

APUSetAlarmGroupText(), APUSetQueryAlarmState(),  
APUSetQueryToPriority(), APUSetTimeoutValues()

## APUSetQueryToPriority() 函数

设置查询的“到优先级”。

### 类别

View

### 语法

```
[Result=] APUSetQueryToPriority(iInstance,iPriority);
```

### 参数

*iInstance*

Alarm Printer 的实例（0 到 15）。

*iPriority*

表示“到优先级”的整数值，范围从 1 到 999。它应该设置为大于或等于指定的实例中相同查询的“从优先级”值。

### 附注

“到优先级”必须大于或等于“从优先级”，才能设置有效的报警优先级范围。包含 APUSetQueryToPriority() 函数的脚本无法与查询同时运行。

返回整型错误码。

### 示例

本例将同实例 0 关联的查询的“到优先级”值设置为 240。

```
Status = APUSetQueryToPriority(0,240);
```

### 另请参阅

APUSetAlarmGroupText(), APUSetQueryAlarmState(),  
APUSetQueryFromPriority(), APUSetTimeoutValues()

## APUSetTimeoutValues() 函数

APUSetTimeoutValues() 函数设置以秒为单位的超时间隔。超时间隔控制：程序正在运行时，观察到由于内存访问或无法获取有效的响应所导致的错误数量。

缺省内存访问超时为两秒钟。缺省短响应等待时间为 10 秒，缺省长响应等待时间为 20 秒。

### 类别

View

### 语法

```
[Result=] APUSetTimeoutValues (iMemory,iShort, iLong) ;
```

### 参数

#### *iMemory*

整型 - 访问超时

#### *iShort*

短响应等待时间（整型）

#### *iLong*

长响应等待时间（整型）

### 示例

```
Status =  
    APUSetTimeoutValues (iMemory, iShort, iLong) ;
```

### 另请参阅

APUSetAlarmGroupText(), APUSetQueryAlarmState(),  
APUSetQueryFromPriority(), APUSetQueryToPriority()

## 处理 Alarm Printer 错误

使用 `APUTranslateErrorCode()` 函数将 Alarm Printer 错误码转换为错误消息。

### APUTranslateErrorCode() 函数

将 APU 函数之一返回的错误码转换为简要描述错误码的中文字符串。

#### 类别

View

#### 语法

```
[Result=]  
APUTranslateErrorCode(iErrorCode,sTagMessage);
```

#### 参数

##### *iErrorCode*

整型错误码，通常由其它大多数 APU 函数返回。

##### *sTagMessage*

消息标记，用于接收错误消息。

#### 附注

此函数将错误消息作为 `sTagMessage` 参数返回给某个消息标记。如果传递了未知的错误码，此函数可能会失败。

返回整型错误码。

#### 示例

如果 Alarm Printer 的实例 15 当前不在运行，则本例将消息标记 `errmsg` 设置为“没有可用的实例”。

```
Status =  
APUTranslateErrorCode(APUSetAlarmGroupText(15,"  
$system"), ErrMsg);
```

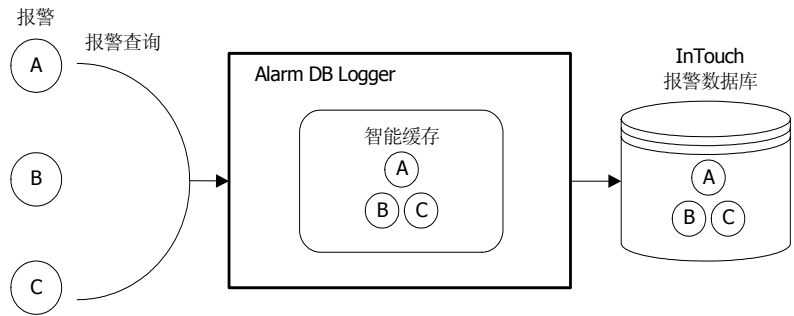


# 第 9 章

## 将报警记录到报警数据库

“InTouch 分布式报警系统”包含 Alarm DB Logger 实用程序，用于将报警与事件记录到报警数据库。

Alarm DB Logger 是一个报警接收器。您可以给它配置一个或多个查询，以便从 InTouch 报警供应器中选择报警。这些查询选择的报警存储在临时内存缓冲区（也称为“智能缓存”）中。Alarm DB Logger 按周期性的间隔将“智能缓存”中的内容作为报警与事件记录写入报警数据库。



Alarm DB Logger 可以自动重新连接。丢失与数据库的连接时，Logger 按固定的间隔检查数据库连接。重新建立与报警数据库的连接之后，记录功能恢复执行。

Alarm DB Logger 会报告所有的错误，不论是以服务方式还是作为普通的 ArchestrA Logger 应用程序来运行。

## Alarm DB Logger Manager 的 SQL Server 帐户

Alarm DB Logger Manager 在 SQL Server 数据库中创建并使用以下帐户：

帐户	口令
wwAdmin	wwAdmin
wwPower	wwPower
wwUser	wwUser

## 使用 Alarm DB Logger Manager

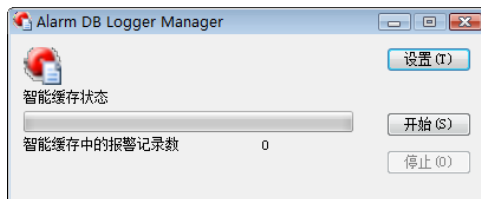
使用 Alarm DB Logger Manager 启动与停止记录操作。Alarm DB Logger Manager 可以作为服务或普通的应用程序来启动。

您可以使用 Alarm DB Logger 配置向导来配置报警记录功能，该向导可以从 Alarm DB Logger Manager 中启动。

Alarm DB Logger Manager 显示报警记录占用“智能缓存”的百分比。SQL Server 连接丢失或报警发生速度比 Alarm DB Logger 可以将它们记录到报警数据库的速度更快时，这些报警会进行缓存。

**要打开 Alarm DB Logger Manager**

- 1 在工具视图中，展开应用程序。
- 2 双击 **Alarm DB Logger Manager**。此时出现 **Alarm DB Logger Manager**。



## 配置报警数据库记录功能

要将 InTouch 报警与事件数据记录到报警数据库，请从 Alarm DB Logger Manager 中执行以下操作：

- 配置与报警数据库的连接
- 选择要记录到报警数据库的报警
- 设置将记录保存到报警数据库的间隔
- 选择运行 Alarm DB Logger 的方法

### 配置数据库连接

您必须在 Alarm DB Logger 与报警数据库之间建立连接。

Alarm DB Logger 仅支持 SQL Server 身份验证，并且 SQL Server 身份验证必须设置为混合模式。

#### 要配置数据库连接

- 1 打开 Alarm DB Logger Manager。执行以下操作：
  - a 在工具视图中，展开应用程序。
  - b 双击 **Alarm DB Logger Manager**。
- 2 单击**设置**。此时出现 **Alarm DB Logger Manager - 配置**向导。



- 3 配置数据库连接。执行以下操作：
  - a 在**服务器名**框中，输入安装了报警数据库的计算机的节点名。
  - b 在**数据库**框中，输入 InTouch 报警数据库的名称。
  - c 在**用户名**框中，输入为报警数据库创建的用户帐户名。
  - d 在**口令**框中，输入与报警数据库用户帐户关联的口令。
- 4 在**记录模式**区域中，配置存储记录的方式。执行以下操作之一：
  - 单击**详细**，以便为每个报警条件（处于报警状态、已确认、已返回正常）存储一条单独的记录。
  - 单击**合并**，以便将报警的所有状态（处于报警状态、已确认、已返回正常）存储在一条记录中，并包含每次转换的时间标签。
- 5 如果需要，单击**创建**以创建数据库。
- 6 单击**测试连接**以验证与报警数据库的连接。此时出现一条消息，指出已成功连接到数据库。
- 7 单击**下一步**以配置所要记录的报警。请参阅第 252 页的“配置要记录的报警”。

## 配置要记录的报警

在 Alarm DB Logger Manager 配置向导的第二个页面中，可以定义一个或多个查询，以便从 InTouch 或 Galaxy 报警供应器中选择报警。您也可以选择作为数据库查询一部分的报警优先级值范围。

远程节点使用以下查询语法：

\\nodeName\Provider!AlarmGroup

本地节点使用以下查询语法：

\Provider!AlarmGroup

示例：

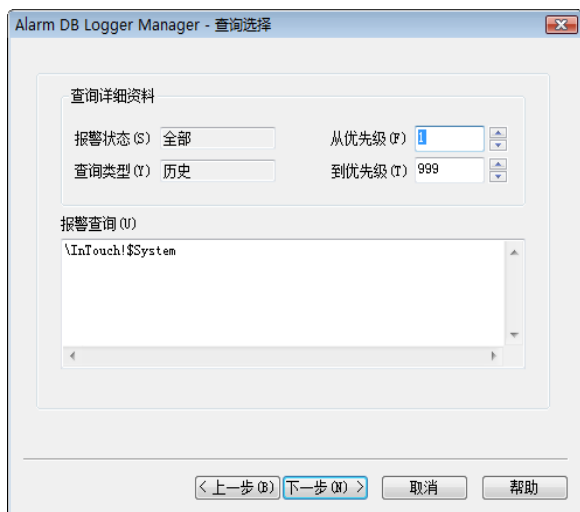
\\ProdSvr\InTouch!\$System

以下是一个使用 Galaxy 报警供应器的示例。此查询提供特定区域的所有报警。

\\Galaxy!Area

### 要配置所要记录的报警

- 1 打开 Alarm DB Logger Manager 并启动配置向导。执行以下操作：
  - a 在工具视图中，展开应用程序。
  - b 双击 **Alarm DB Logger Manager**。
  - c 单击**设置**。此时出现 **Alarm DB Logger Manager - 配置向导**。
- 2 单击**下一步**。此时出现 **Alarm DB Logger Manager - 查询选择** 页面。



只读的**报警状态**框显示要记录的报警状态。只读的**查询类型**框显示查询类型。

- 3 在**从优先级**框中，输入报警优先级范围的起始值。
- 4 在**到优先级**框中，输入报警优先级范围的结束值。
- 5 在**报警查询**框中，输入要用于在报警数据库中存储或检索数据的报警查询。
- 6 单击**下一步**以配置记录间隔。请参阅第 254 页的“配置记录间隔”。

## 配置记录间隔

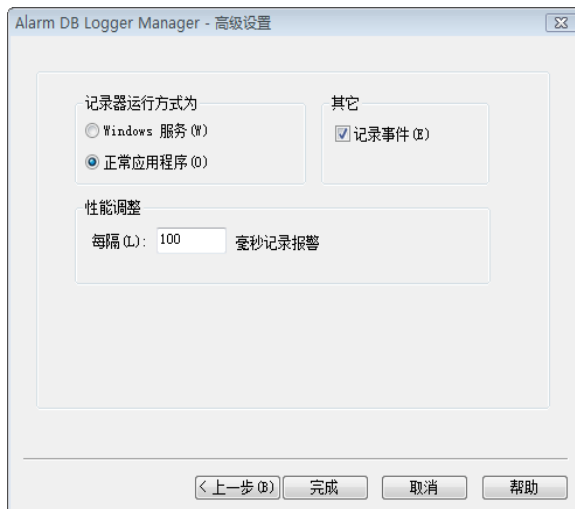
在 Alarm DB Logger Manager 配置向导的第三个页面中，可以配置高级查询设置。您可以将事件记录到报警数据库中。通过设置将报警记录从内部报警内存写入数据库的频率，您也可以调整报警记录功能的性能。

记录间隔不同于 SQL Server 的重新连接速率。记录间隔是读取报警的间隔。重新连接速率取决于同 SQL Server 尝试建立连接的超时值。

将记录间隔设置得过低会影响系统性能。

### 要配置记录间隔

- 1 打开 Alarm DB Logger Manager 并启动配置向导。执行以下操作：
  - a 在工具视图中，展开应用程序。
  - b 双击 Alarm DB Logger Manager。
  - c 单击设置。此时出现 Alarm DB Logger Manager - 配置向导。
- 2 单击下一步。此时出现 Alarm DB Logger Manager - 查询选择页面。
- 3 单击下一步。此时出现 Alarm DB Logger Manager - 高级设置页面。

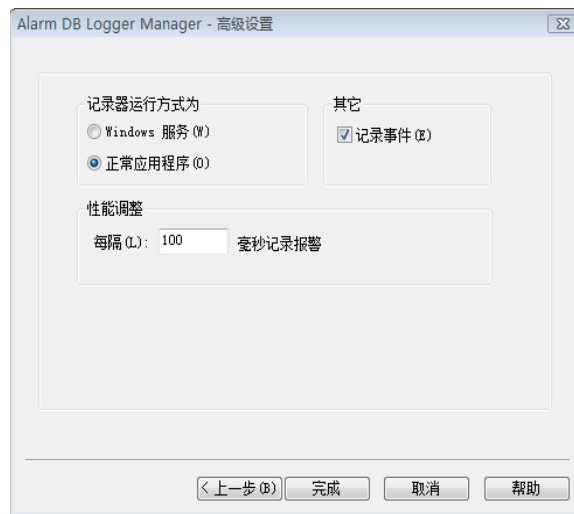


- 4 如果希望将 InTouch 事件记录存储到报警数据库，请选择**记录事件**复选框。您还可以储存来自 ArchedrA Galaxy 的事件。
- 5 在**性能调整**区域中，以毫秒为单位输入将报警记录写入报警数据库的间隔。
- 6 单击**完成**。

## 将 Alarm DB Logger 配置成服务

您可以将 Alarm DB Logger 配置成作为服务来运行。

- 1 作为管理员登录到计算机。
- 2 打开 Alarm DB Logger Manager 并启动配置向导。执行以下操作：
  - a 在工具视图中，展开**应用程序**。
  - b 双击 **Alarm DB Logger Manager**。
  - c 单击**设置**。此时出现 **Alarm DB Logger Manager - 配置向导**。
- 3 单击**下一步**。此时出现 **Alarm DB Logger Manager - 查询选择**页面。
- 4 单击**下一步**。此时出现 **Alarm DB Logger Manager - 高级设置**页面。



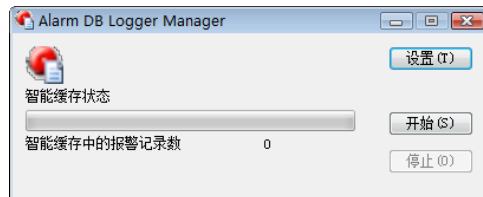
- 5 在**记录器运行方式为**区域中，单击 **Windows 服务**或**正常应用程序**。
- 6 单击**完成**。

## 启动与停止报警数据库记录功能

Alarm DB Logger 将记录写入报警数据库。它既可以作为服务，也可以作为普通的应用程序来启动与运行（具体取决于配置 Alarm DB Logger 时选择的运行模式）。

### 要启动或停止报警记录功能

- 1 在工具视图中，展开应用程序。
- 2 双击 **Alarm DB Logger Manager**。此时出现 **Alarm DB Logger Manager**。



智能缓存状态显示报警记录占用内存缓冲区的百分比。

- 3 单击**开始**以开始报警记录过程。
- 4 单击**停止**以结束报警记录过程。

## 报警数据库视图

您可以使用一组 SQL Server 数据库视图轻松查询过去与当前发生的报警与事件。

数据库视图只是一个逻辑表，它将多个底层数据库表中的信息合并到一起。数据库视图通常称为虚拟数据库表，原因在于它们可以使用标准的 SQL 语句来查询，好像它们是真实存在的表格那样。查询视图时，它返回一组记录（或称作行）。每行有多列信息，包含的是记录数据。

您可以使用报警数据库视图来执行复杂的查询。例如，您可以检索工厂的特定区域在特定的时间跨度内发生的所有 HiHi 报警的报警记录。或者，您也可以检索某个特定的报警供应器节点所记录的所有数据改变事件。

视图中的所有字符串都采用 Unicode 编码。



## 报警历史视图

v\_AlarmHistory 视图包含所选时间范围内发生的所有历史报警与报警转换事件。查询指定开始和结束日期与时间（EventStamp 或 EventStampUTC 列）。返回的记录包括报警发生、报警确认、报警启用、报警禁用以及报警返回正常等事件。

下表介绍视图中的各个列：

列名	数据类型	描述
EventStamp	Datetime	报警事件的日期与时间（采用数据库本地时间）
AlarmState	nChar	报警状态：UNACK、UNACK_RTN、ACK、ACK_RTN 之一。
Tagname	nChar	生成报警的对象的名称，如 TIC101。
Description	nVarchar	报警的描述字符串。可以缺省为对象描述（或 InTouch HMI 中的注释）。或者是已确认的记录的确认证释。
Area	nChar	报警的“区域”或“组”的名称。
Type	nChar	报警类型，例如 Hi、HiHi、ROC、PV.HiAlarm。
Value	nChar	报警时报警变量的值。
CheckValue	nChar	报警时报警限的值。
Priority	Integer	报警优先级。
Category	nChar	报警类或报警类别。例如“值”、“偏差”、“变化率”、“过程”、“批次”、“系统”，等等。
Provider	nChar	报警的供应商：节点 /InTouch，或 GalaxyName。
Operator	nChar	操作员的姓名。
DomainName	nChar	域的名称。
UserFullName	nChar	操作员用户的全名（例如，Joseph P. Smith）。
UNACKDuration	Float	最近一次报警转换（报警或子状态）与确认之间的时间（如果有）。
User1	Float	用户自定义数字 1。
User2	Float	用户自定义数字 2。
User 3	nChar	用户自定义字符串字段。
EventStampUTC	DateTime	报警事件的 UTC 日期 / 时间。

列名	数据类型	描述
Millisec	Small integer	事件标签秒值的小数部分，以 0.1 毫秒为增量。
OperatorNode	nvarchar(32)	操作员在其中确认报警的节点的名称。 v_AlarmHistory2 视图是 v_AlarmHistory 视图的变体。
列名	数据类型	描述
EventStamp	Datetime	报警事件的日期与时间（采用数据库本地时间）
AlarmState	nChar	报警状态：UNACK、UNACK_RTN、ACK、ACK_RTN、ACK_ALM、UNACK_ALM 之一
TagName	nChar	生成报警的对象的名称，如 TIC101。
Description	nVarchar	报警的描述字符串。可以缺省为对象描述（或 InTouch 中的注释）。或者是已确认的记录的确 认注释。
Area	nChar	报警的“区域”或“组”的名称。
Type	nChar	报警类型，例如 Hi、HiHi、ROC、PV.HiAlarm。
Value	nChar	报警时报警变量的值。
CheckValue	nChar	报警时报警限的值。
Priority	Integer	报警优先级。
Category	nChar	报警类或报警类别。例如“值”、“偏差”、“变化率”、“过程”、“批次”、“系统”，等等。
Provider	nChar	报警的供应器：节点 /InTouch，或 GalaxyName。
Operator	nChar	操作员名称：JoeR（如果有）。
DomainName	nChar	域的名称。
UserFullName	nChar	操作员用户的全名（例如，Joseph P. Smith）。
AlarmDuration	Float	报警发生与返回正常之间的时间。
User1	Float	用户自定义数字 1。
User2	Float	用户自定义数字 2。
User 3	nChar	用户自定义字符串字段。
EventStampUTC	Date Time	报警事件的 UTC 日期 / 时间。
Millisec	Small integer	事件标签秒值的小数部分，以 0.1 毫秒为增量。

示例查询 - 报警历史视图

本例从“报警历史”视图中选择所有记录：

```
SELECT * FROM v_AlarmHistory
```

本例从“报警历史”视图中选择优先级大于 100 的所有记录。

```
SELECT * FROM v_AlarmHistory WHERE Priority >100
```

事件历史视图

v\_EventHistory 数据库视图列出所选时间范围内发生的所有历史事件。查询客户端指定开始和结束日期与时间范围。返回的记录包括所有的非报警事件。

列名	数据类型	描述
EventStamp	Datetime	事件的日期与时间。
TagName	nChar	生成事件的对象的名称，如 Pump1。
Description	nVarChar	事件的描述字符串。可以缺省为对象描述，或 InTouch 中的注释。
Area	nChar	事件的“区域”或“组”的名称。
Type	nChar	事件类型，如“操作员数据改变”、“启动时”，等等。
Value	nChar	新值（如果有）。
CheckValue	nChar	旧值（如果有）。
Category	nChar	事件类别或类，如“值”、“过程”、“批次”、“系统”，等等。
Provider	nChar	事件发生器，如节点 /InTouch，或用于更改用户的“视图引擎”的名称。
Operator	nChar	operator1 的名称：JoeR（如果有）。
DomainName	nChar	域的名称。
UserFullName	nChar	操作员用户的全名（例如，Joseph P. Smith）。
User1	Float	用户自定义数字 1。
User2	Float	用户自定义数字 2。
User 3	nChar	用户自定义字符串字段。
EventStampUTC	DateTime	事件的 UTC 日期 / 时间。
Millisec	Small integer	事件标签秒值的小数部分，以 0.1 毫秒为增量。

## 报警事件历史视图

`v_AlarmEventHistory` 数据库视图提供所选时间范围内发生的所有事件与报警的历史列表。查询客户端指定开始和结束日期与时间。返回的记录包括所有的报警与事件。该视图将报警视图与事件视图合而为一，代表这两个视图中的记录的集合。

列名	数据类型	描述
EventStamp	Datetime	事件的日期与时间。
AlarmState	nChar	报警状态：UNACK、UNACK_RTN、ACK、ACK_RTN 之一。不适用于事件。
TagName	nChar	生成报警的对象的名称，如 TIC101。
Description	nVarchar	报警 / 事件的描述字符串。可以缺省为对象描述（或 InTouch 中的注释）。或者是已确认的记录的确认注释。
Area	nChar	报警的“区域”或“组”的名称。
Type	nChar	报警或事件的类型，例如 Hi、HiHi、“变化率”、PV.HiAlarm、“操作员数据改变”，等等。
Value	nChar	报警时报警变量的值。
CheckValue	nChar	报警时报警限的值，或事件的旧值。
Priority	Integer	报警优先级。
类别	nChar	报警或事件的类，或报警类别，如“值”、“过程”、“批次”、“系统”，等等。
Provider	nChar	报警的供应器，如节点 /InTouch，或 GalaxyName。
Operator	nChar	确认操作员或数据改变操作员的名称。
DomainName	nChar	域的名称。
UserFullName	nChar	操作员用户的全名（例如，Joseph P. Smith）。
UNACKDuration	Float	从最近一次报警转换到“确认”所经过的毫秒数。
User1	Float	用户自定义数字 1。
User2	Float	用户自定义数字 2。
User 3	nChar	用户自定义字符串字段。
EventStampUTC	DateTime	事件的 UTC 日期 / 时间。
Millisec	Small integer	事件标签秒值的小数部分，以 0.1 毫秒为增量。

v\_AlarmEventHistory2 视图是 v\_AlarmEventHistory 视图的变体。

列名	数据类型	描述
EventStamp	Datetime	事件的日期与时间。
AlarmState	nChar	报警状态：UNACK、UNACK_RTN、ACK、ACK_RTN 之一。不适用于事件。
TagName	nChar	生成报警的对象的名称，如 TIC101。
Description	nVarchar	报警 / 事件的描述字符串。可以缺省为对象描述（或 InTouch 中的注释）。或者是已确认的记录的确认证释。
Area	nChar	报警的“区域”或“组”的名称。
Type	nChar	报警或事件的类型，例如 Hi、HiHi、“变化率”、PV.HiAlarm、“操作员数据改变”，等等。
Value	nChar	报警时报警变量的值。
CheckValue	nChar	报警时报警限的值，或事件的旧值。
Priority	Integer	报警优先级。
Category	nChar	报警或事件的类，或报警类别，如“值”、“过程”、“批次”、“系统”，等等。
Provider	nChar	报警的供应器，如节点 /InTouch，或 GalaxyName。
Operator	nChar	确认操作员或数据改变操作员的名称。
DomainName	nChar	域的名称。
UserFullName	nChar	操作员用户的全名（例如，Joseph P. Smith）。
AlarmDuration	Float	从报警发生到返回到正常 (RTN) 所经过的毫秒数。
User1	Float	用户自定义数字 1。
User2	Float	用户自定义数字 2。
User 3	nChar	用户自定义字符串字段。
EventStampUTC	Date Time	事件的 UTC 日期 / 时间。
Millisec	Small integer	事件标签秒值的小数部分，以 0.1 毫秒为增量。

### 示例查询 - 报警事件历史视图

对于标记名为 **MyTag1**、报警状态为 **ACK\_RTN** 或 **ACK** 的所有记录，下例选择这些记录中 **TagName**、**Area** 以及 **Type** 列中的数据。结果按报警供应器排序。

```
SELECT TagName, Area, Type FROM v_AlarmEventHistory
WHERE TagName='MyTag1'
      AND (AlarmState='ACK_RTN' OR AlarmState='ACK')
ORDER BY Provider
```

## AlarmSuite 报警日志视图

v\_AlarmSuiteAlarmLog 数据库视图提供历史报警与事件的视图，它返回的表与 AlarmSuite 实际的表 AlarmLog 具有相同的列。适用于 AlarmSuite 的 AlarmLog 表的查询也同样适用于此视图，只是这些查询中的表名必须更改为视图名。

列名	数据类型	描述
EventStamp	Datetime	事件的日期与时间。
EventType	NChar	由 AlarmSuite 定义的事件类型。
AlarmType	NChar	报警的类型。
AlarmState	NChar	报警的确认状态。
NodeName	NChar	报警的节点。
TagName	NChar	发出报警的对象的名称。
GroupName	NChar	报警组或区域的名称。
Comment	NChar	确认注释（如果有）。
Value	Float	报警变量的值。
Limit	Float	报警时报警限的值。
ValueString	NChar	请参阅 AlarmSuite 表格。
Operator	NChar	操作员用户名。
Priority	Integer	优先级。
Units	NChar	在 7.11 中返回 ""。

### 示例查询 - AlarmSuite 报警日志视图

下例从视图中选择所有不同的标记名：

```
SELECT Distinct TagName FROM v_AlarmSuiteAlarmLog
```

## 报警数据库存储过程

您可以使用一组 **SQL Server** 存储过程轻松检索信息，以便用于分析以及报警与事件的报告。

存储过程是 **SQL** 语句的集合，用于执行特定的功能并从数据库返回数据。存储过程可以接受一些输入参数，控制它们如何工作，如何以结果集的形式返回数据。

返回的结果是一组记录，以非常类似于 **SQL Server** 数据库视图的 **SQL** 记录集的面目出现。存储过程在数据库中以内嵌 **SQL** 语句的形式存在，可减少同客户端之间的往返，因此可以提高性能。

如需有关存储过程的详细信息（包括如何查看它们的定义），请参阅 **Microsoft SQL Server** 文档。

## 调用存储过程

使用 **EXECUTE** 语句调用存储过程。

```
EXECUTE sp_AlarmCounter @StartDate='2007-01-01',  
    @EndDate='2007-03-31', @Tagname = 'tag1', @Type =  
    'LO', @Provider = 'WW21353\InTouch', @Comment =  
    'SSAADD'
```

在本例中，**StartDate** 与 **EndDate** 参数是必需的。其余参数属于可选项。如果不提供某个参数的值，则它不会用于过滤结果集。

如果存储过程包含日期 / 时间变量，则可以使用 **SQL Server** 文档中指定的任何有效格式。



## AlarmCounter 数据库存储过程

此存储过程返回一定时间间隔内发生独特报警的次数。独特的报警由 TagName、Provider、AlarmType 及 Category 来标识。

此计数器仅适用于报警发生次数（不适用于确认或返回到正常之类的转换）。因此举例而言，如果发生某个报警，然后确认该报警，再然后该报警返回到正常状态，则该报警的计数只有 1（而不是 3）。

查询必须指定开始和结束日期与时间。它有五个可选参数：TagName、Class、Type、Provider 以及 Comment。

此视图可回答的问题的例子有：供应器 Node1|InTouch（供应器）上的对象 TIC101（标记名）在某个时间跨度上发生了多少次 HiHi（类型）值报警（类别）？

**备注** AlarmCounter 存储过程仅适用于“详细”记录模式，不受“合并”记录模式的支持。

sp\_AlarmCounter

列名	数据类型	描述
TagName	Nchar	生成报警的对象的名称，如 TIC101。
GroupName	Nchar	报警的区域或组的名称。
AlarmType	Nchar	报警类型，例如 Hi、HiHi、ROC、PV、HiAlarm。
AlarmClass	Nchar	报警类或报警类别，如“值”、“过程”、“批次”等。
AlarmCount	Integer	一定时间范围内报警发生的次数。如果报警发生在开始日期与时间之前，则它不包括在计数内。
Priority	Integer	报警优先级。
Provider	Nchar	报警的供应器，如节点 /InTouch，或 GalaxyName。
Comment	Nchar	报警注释。

示例查询：

```
EXECUTE sp_AlarmCounter @StartDate='2007-01-01
23:23:23', @EndDate='2007-03-31 23:23:23', @Tagname =
'$NewAlarm'
```

## EventCounter 数据库存储过程

此数据库存储过程提供一定时间间隔内某个特定标记上发生的某一特定类型的事件数。查询客户端必须指定开始和结束日期与时间。它有三个可选参数：TagName、Provider 以及 Comment。此计数器仅适用于非报警事件。此视图的用途是显示每个事件的频率。例如，泵浦打开了多少次？TagName、Provider、Category 及 Type 用于唯一确定事件并执行计数。

sp\_EventCounter

列名	数据类型	描述
TagName	NChar	生成报警的对象的名称，如 TIC101
Area	NChar	报警的“区域”或“组”的名称。
Type	NChar	事件类型。
Category	NChar	报警类或报警类别，如“值”、“过程”、“批次”等。
EventCount	Integer	在指定的时间范围内 TagName（标记名）此 Type（类型）的事件发生的次数。
Provider	NChar	事件的供应器：节点 /InTouch，或 GalaxyName。
Comment	NChar	事件注释。

示例查询：

```
EXECUTE sp_EventCounter @StartDate='2007-01-01
23:23:23', @EndDate='2007-03-31 23:23:23', @Tagname =
'$NewAlarm'
```

## 第 10 章

### 查看记录的报警

您可以使用 Alarm DB View ActiveX 控件以可视化方式显示报警数据库中的数据。使用此控件可以显示 InTouch 应用程序在运行时生成的所有报警与事件信息。

对于此控件，您可以配置：

- 上下文相关菜单功能
- 显示模式
- 列表控制选项
- 不同属性的颜色
- 字体类型、样式及大小
- 数据库规格（服务器名、用户 ID 及口令）
- 查询过滤器
- 列管理
- 排序

在应用程序运行期间，运行时用户可以更改一些选项来选择要查看的数据。它们可以：

- 给列中的信息排序
- 更新显示对象
- 执行查询
- 调整列宽

如需有关 ActiveX 控件的详细信息，请参阅 *InTouch® HMI 可视化指南* 中的第 6 章 “ActiveX 控件”。

## 配置 Alarm DB View 控件

配置 Alarm DB View 控件时，可以：

- 配置与报警数据库的连接。
- 配置网格外观。
- 选择显示字体。
- 配置视觉外观。
- 设置用户在应用程序运行期间可以访问的功能。
- 配置要显示的报警。
- 创建自定义过滤器。
- 设置各类报警记录的颜色。
- 配置显示的时间格式。
- 配置显示的报警记录的排序顺序。

## 配置数据库连接

您必须配置 Alarm DB View 控件与报警数据库之间的连接。

使用对报警数据库有适当的只读访问权限的帐户，而不是系统管理员帐户。

### 要配置与报警数据库的连接

- 1 使用鼠标右键单击 Alarm DB View 控件，然后单击**属性**。此时出现 **AlmDbViewCtrl 属性** 对话框。
- 2 单击**数据库**选项卡。



- 3 配置与报警数据库的连接。执行以下操作：
  - a 在**服务器名**框中，输入安装了报警数据库的计算机的节点名。
  - b 在**数据库名**框中，输入数据库名。
  - c 在**用户 ID**框中，输入有效的用户帐户名。
  - d 在**口令**框中，输入与用户帐户关联的口令。
- 4 选择**自动连接**复选框，使 Alarm DB View 控件在 InTouch 应用程序开始运行时自动连接到报警数据库。

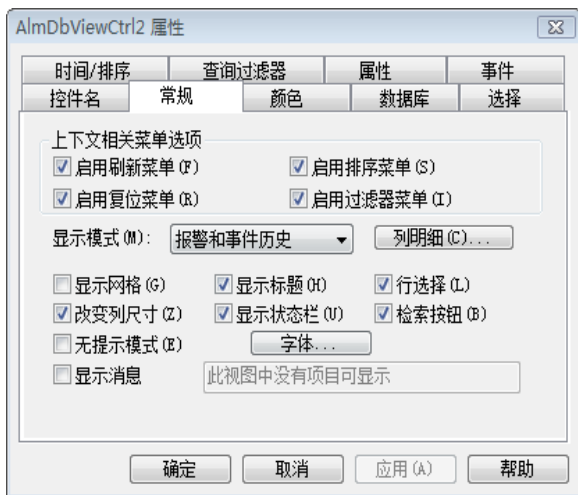
如果不选择**自动连接**复选框，则必须通过明确调用 **Connect()** 方法将 Alarm DB View 控件配置为连接到报警数据库。如需有关 **Connect** 方法的详细信息，请参阅第 318 页的“**Connect()** 方法”。
- 5 单击**测试连接**以验证与报警数据库的连接。此时出现一条消息，指出连接成功。
- 6 单击**应用**。

## 配置网格外观

您可以配置用于确定 Alarm DB View 控件视觉外观的属性。

### 要配置网格外观

- 1 使用鼠标右键单击 Alarm DB View 控件，然后单击**属性**。  
此时出现 **AlmDbViewCtrl 属性**对话框。
- 2 单击**常规**选项卡。



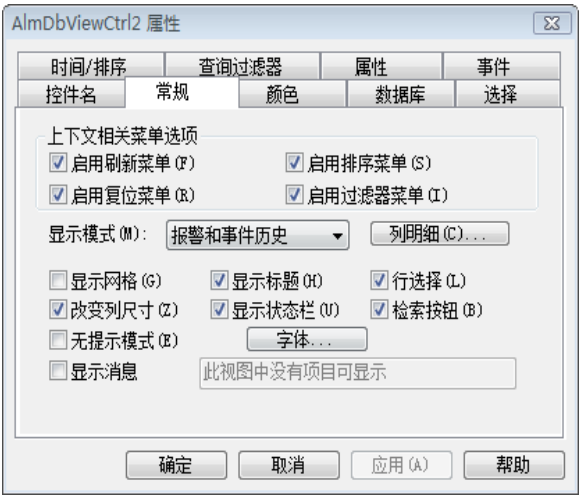
- 3 配置常规外观。执行以下操作：
  - 单击**显示网格**复选框以显示网格。
  - 单击**无提示模式**复选框，以防止控件在运行时显示任何错误消息。
  - 单击**显示消息**复选框，以便在报警查询未返回任何记录的情况下显示一条消息。在方框中输入要显示的消息。
  - 单击**显示标题**复选框以显示控件标题。
  - 单击**显示状态栏**复选框以显示状态栏。
- 4 单击**应用**。

## 配置显示字体

您可以选择 Alarm DB View 控件中出现的所有文本的字体。

### 要配置字体属性

- 1 使用鼠标右键单击 Alarm DB View 控件，然后单击**属性**。  
此时出现 **AlmDbViewCtrl 属性**对话框。
- 2 单击**常规**选项卡。



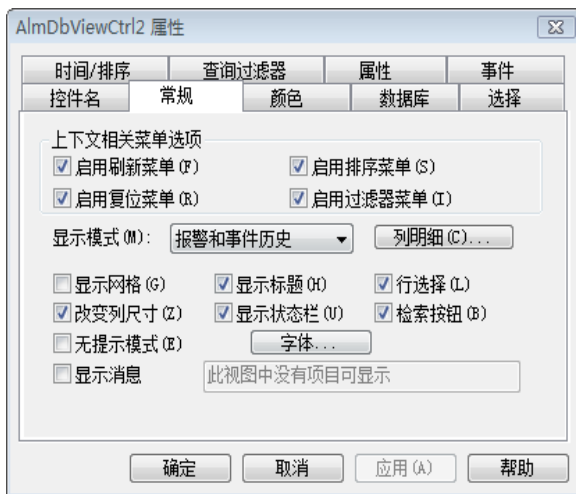
- 3 单击**字体**。此时出现标准的 Windows 字体对话框。配置字体，然后单击**确定**。
- 4 单击**应用**。

## 选择报警或事件数据

您可以配置在 Alarm DB View 控件中显示报警记录、事件记录，或同时显示这两者。

### 要选择数据类型

- 1 使用鼠标右键单击 Alarm DB View 控件，然后单击**属性**。此时出现 **AlmDbViewCtrl 属性** 对话框。
- 2 单击**常规**选项卡。



- 3 在**显示模式**列表中，单击历史记录的类型：
  - 单击**报警和事件历史**，以同时显示报警与事件的历史数据库记录。
  - 单击**报警历史**以便仅显示历史报警记录。
  - 单击**事件历史**以便仅显示历史事件记录。
- 4 单击**应用**。



## 选择与配置显示列

对于 Alarm DB View 控件，可以：

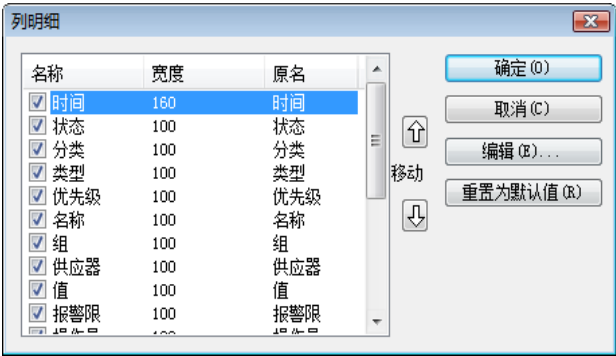
- 选择各个列并进行排序。
- 以像素为单位设置列宽。
- 给某个列重命名。

您至少必须选择一个列。

**重要** 如果显示模式发生更改，则列名会重置为缺省的列名。在更改列名之前，最好先选择显示模式。

### 要配置显示列明细

- 1 使用鼠标右键单击 Alarm DB View 控件，然后单击**属性**。此时出现 **AlmDbViewCtrl 属性**对话框。
- 2 单击**常规**选项卡。
- 3 单击**列明细**。此时出现**列明细**对话框。



- 4 在**名称**列中，对于希望出现在 Alarm DB View 控件中的列，选择其名称旁边的复选框。您至少必须从列表中选择一列。

列名	描述
时间	显示报警发生的时间。
状态	显示报警的状态。
分类	显示报警类别。
类型	显示报警类型。
优先级	显示报警优先级。
名称	显示导致报警或事件的标记或数据源。
组	显示报警组名。
供应器	显示报警供应器的名称。

列名	描述
值	显示报警发生时标记的值。
限制	显示标记的报警限值。
操作员	显示与报警条件关联且已登录的操作员的 ID。
操作员全名	显示已登录的操作员的全名。
操作员节点	显示与报警条件关联且已登录的操作员的节点。  在“终端服务”环境中，这是操作员建立“终端服务”会话的客户端计算机的名称。如果无法检索节点名，则使用节点的 IP 地址代替。
操作员域	显示与报警条件关联且已登录的操作员的域。
报警注释	显示标记的注释。此注释是在数据库中定义标记的报警时，在 <b>报警注释</b> 框中输入的。为报警引入某个确认注释时，新的注释更新到这个注释列中。
用户 1	显示与报警对应的用户自定义数字 1 的数值。
用户 2	显示与报警对应的用户自定义数字 2 的数值。
用户 3	显示与报警关联的用户定义字符串属性的字符串值。
持续时间	显示未确认持续时间或报警持续时间，具体取决于操作员所作的选择。
UTC 时间	显示 UTC 格式的报警时间。



- 5 通过选择列名并使用上、下箭头，重新排列各个列。出现在**列明细**对话框顶部的列名是报警控件最左侧的列。
- 6 要更改某个列的名称或其宽度，请选择该列并单击**编辑**。此时出现**编辑**对话框。



- a 在**新名**框中，输入新的列名。

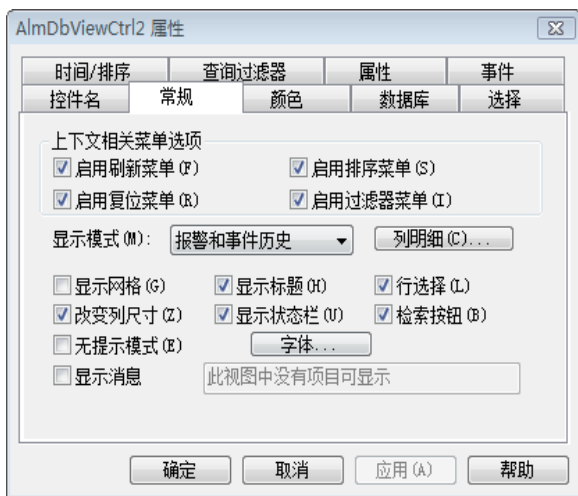
- a 在**新宽度**框中，输入列宽。列宽可以在从 1 到 999 个像素的范围内。
  - b 单击**确定**。
- 7 单击**重置为默认值**，以恢复缺省的列明细设置。
- 8 单击**列明细**对话框中的**确定**。
- 9 单击**应用**。

## 控制用户可以在运行时访问哪些功能

您可以启用与禁用此控件中可用的快捷菜单功能。

### 要配置运行时功能

- 1 使用鼠标右键单击 Alarm DB View 控件，然后单击**属性**。  
此时出现 **AlmDbViewCtrl 属性**对话框。
- 2 单击**常规**选项卡。



- 3 在**上下文相关菜单选项**区域中，配置在运行时可用的菜单命令。
  - 选择**启用刷新菜单**复选框，以便在运行时启用控件快捷菜单中的**刷新**菜单选项。**刷新**菜单根据数据库来刷新控件，如果连接成功，它显示 1 到 **MaxRecords** 属性所定义的数字这个范围内的一组记录。
  - 选择**启用复位菜单**复选框，以便在运行时启用控件快捷菜单中的**重置**菜单选项。“重置”菜单将所有的列还原为设计时保存的设置。
  - 选择**启用排序菜单**复选框，以便在运行时启用控件快捷菜单中的**排序**菜单选项。此菜单显示**第二位排序**菜单，用于设置用户自定义的列排序方式。
  - 选择**启用过滤器菜单**复选框，以便在运行时启用控件快捷菜单上的**过滤器**菜单选项。此菜单显示过滤器菜单，用于设置用户自定义的过滤准则。
- 4 单击**改变列尺寸**复选框以调整列宽。
- 5 选择**行选择**复选框，使您可以选择报警记录。
- 6 选择**检索按钮**复选框，以便在控件右侧显示检索按钮。
- 7 单击**应用**。

## 配置为报警记录显示的时间格式与时区

您可以为 Alarm DB View 控件中显示的报警记录配置时间格式与时区。

### 要配置时间格式

- 1 使用鼠标右键单击 Alarm DB View 控件，然后单击**属性**。此时出现 **AlmDBViewCtrl 属性** 对话框。
- 2 单击**时间 / 排序**选项卡。



- 3 在**时间格式**列表中，单击某个时间格式：

字符串字符	描述	示例
d	两位数字的日期	31
b	三个英文字母的月份缩写	八月
Y	四位数字的年份	2007
m	两位数字的月份	11
/	日期分隔符	06/2007
y	两位数字的年份	07
#x	完整的日期与星期	2002 年 8 月 9 日星期五
B	完整的月份名	九月
-	连字符日期分隔符	06-07
.	句点日期分隔符	06.07
,	逗号日期分隔符	八月 09, 2007
H	24 小时制时间	22:15

字符串字符	描述	示例
:	时间分隔符，如 4:41	
M	分钟	00:41
p	显示的“上午”或“下午”	
S	秒钟	16:41:07
s	秒钟的小数部分	16:41:07.390
I	指定 AM/PM 的 12 小时制时间	04:41 下午

您也可以使用自定义文本与格式字符在列表框中手工输入自己的格式字符串。下面是一些时间格式字符串的示例：

时间格式字符串	显示
%d %b	09 八月
%m/%d/%Y	08/09/2002
%#x	2002 年 8 月 9 日星期五
%Y-%m-%d	2002-08-09
%m/%d/%Y %H:%M %p	08/09/2002 16:56 下午
%m/%d/%Y %H:%M:%s %p	08/09/2002 16:56:38.07 下午
%I:%M %p	04:56 下午

- 4 在**显示时区**列表中，单击所需的时区：

时区	描述
GMT	报警时间标签使用“格林尼治标准时间”。
本地时间	报警时间标签显示时使用 Alarm DB View 控件所在客户端的本地时间。
原始时间	报警时间标签显示时使用报警供应器的本地时间。

- 5 单击**应用**。

## 选择报警数据库中数据的时间段

您可以设置查询值来根据所选的时间选择记录。您也可以配置要查看的最大记录数、报警查询的开始与结束时间，以及查询时区。

### 要选择数据的时间段

- 1 使用鼠标右键单击 Alarm DB View 控件，然后单击**属性**。此时出现 **AlmDBViewCtrl 属性**对话框。
- 2 单击**选择**选项卡。



- 3 要使用预定义的时间间隔（查询数据时总是使用 UTC 时区），请单击**持续时间**列表中的某个间隔。
- 4 要使用特定的开始时间与结束时间，请单击**使用指定时间**，然后配置详细资料。
  - a 在**开始时间**框中，输入检索报警记录的开始时间。字符串必须采用 YYYY/MM/DD HH:MM:SS 格式。使用任何时区中自 1970 年 1 月 1 日午夜到 2038 年 1 月 18 日 19:14:07 之间的任何日期。
  - b 在**结束时间**框中，输入停止检索报警记录的结束时间。字符串必须采用 YYYY/MM/DD HH:MM:SS 格式。使用任何时区中自 1970 年 1 月 1 日午夜到 2038 年 1 月 18 日 19:14:07 之间的任何日期。
  - c 在**查询时区**区域中，单击 **UTC** 或**原始时间**。UTC 时间是“格林尼治标准时间”，也称作“通用协调时间”或 Zulu。原始时间指操作员所在时区的当前时间。
- 5 在**持续时间列**区域中，配置显示哪种类型的持续时间。持续时间以毫秒为单位计算。
  - 单击**未确认持续时间**，以显示最近的报警转换（报警或子状态）与确认（如果有）之间的时间。

- 单击**报警持续时间**，以显示报警最初的实例与它返回到正常的时间之间经过的时间量。
- 6 在**最大记录数**框中，输入在一个实例上可以从控件中查看的记录数。最大记录数的有效范围是从 1 到 1000。
  - 7 单击**应用**。

### 关于查询时区

“查询时区”选项有**原始时间**与**UTC**。

- 原始时间指操作员所在时区的本地时间。
- UTC 时间是“格林尼治标准时间”，也称作“通用协调时间”或 Zulu。

如果选择“使用指定时间”，则可以在两个查询时区选项之间进行选择。如果未选择“使用指定时间”，则所选的“持续时间”预定义的时间间隔查询数据时总是使用 UTC 时区。

要避免“夏令时”转换期间的问题，建议在“查询时区”中总是使用 UTC。如果使用“本地时间”，则从“夏令时”到“正常时间”的转换期间的报警记录可能丢失。

如果运行多台使用不同时区设置的计算机，并且它们都记录到相同的报警数据库，则每条记录都将获得 UTC 时间标签，加上所需的时区偏差以及夏令时调整，将该时间标签转换成相应的“原始时间”时需要这样。结果，数据库中的每一项都有两个时间标签：UTC 时间与执行记录的计算机的“原始时间”。这使检索速度更快。在表格项中，UTC 时间标识为“转换时间”；而“原始时间”标识为"EventStamp"。



## 创建自定义过滤器与使用过滤器收藏夹

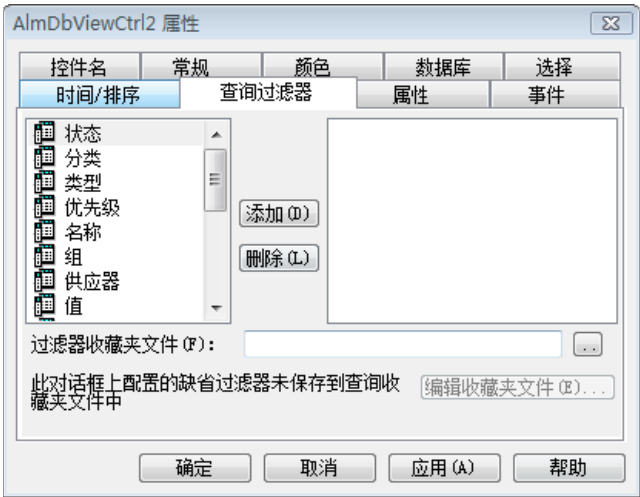
您可以选择要在查询结果中包含哪些记录。例如，您可以按记录日期或报警状态来选择过滤器。您可以选择多个字段来限制或扩展查询结果。

如果没有定义自定义的过滤器，则返回所有的记录。

如果已翻译语言切换字符串，便无法将这些字符串用作过滤器准则。翻译的字符串储存在报警数据库之外的地方。

### 要创建自定义的过滤器

- 1 使用鼠标右键单击 Alarm DB View 控件，然后单击**属性**。此时出现 **AlmDBViewCtrl 属性**对话框。
- 2 单击**查询过滤器**选项卡。



- 3 在左侧窗格中，选择过滤器字段，然后单击**添加**，以便将它们包含到右侧窗格显示的过滤器中。这些过滤器字段在下表中介绍：

字段名称	查询过滤准则：
状态	报警状态。如需有关详细信息，请参阅第 285 页“状态列的值”。
类	报警类。
类型	报警类型。
优先级	报警优先级。
名称	报警名。
组	报警组名。
供应器	报警供应器。

字段名称	查询过滤准则：
值	报警值。“值”列中的值显示为字母数字值。在“查询过滤器”中，这些值的比较是当作字符串比较来完成的。
限制	报警限。值为字母数字字符。在“查询过滤器”中，这些值的比较是当作字符串比较来完成的。
操作员	操作员。
操作员全名	操作员的全名。
操作员节点	与报警关联的操作员的节点名。
操作员域	与报警关联的操作员的域名。
报警注释	报警注释。
用户 1	报警用户自定义数值 1。
用户 2	报警用户自定义数值 2。
用户 3	报警用户自定义字符串值。
持续时间	未确认与报警持续时间。“持续时间”列设置为零时，不会在查询中产生任何包含空值的记录。

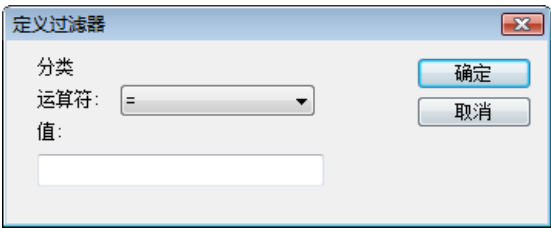
- 4 要从过滤器窗格中删除某个字段，请单击要删除的字段，然后单击**删除**。删除过滤器的操作无法撤消。出现消息时，单击**是**。
- 5 配置每个过滤器字段的准则。如需有关详细信息，请参阅第 282 页的“定义列过滤器准则”。
- 6 配置过滤器的运算符与组合。如需有关详细信息，请参阅第 284 页的“组合报警列”。
- 7 配置查询收藏夹文件。执行以下操作：
  - a 在**过滤器收藏夹文件**框中，输入网络路径与文件名，或单击省略号按钮以浏览文件。
  - b 要编辑**过滤器收藏夹文件**，请单击**编辑收藏夹文件**按钮。此时打开**过滤器收藏夹**窗口，可供您在收藏夹文件中添加、修改或删除过滤器。完成时单击**确定**，以保存更改并关闭窗口。
- 8 单击**应用**。

定义列过滤器准则

对于查询中包含的每个列过滤器，都必须配置过滤器准则。例如，您可能只希望查看特定操作员的报警。

要定义列过滤器

- 1 使用鼠标右键单击字段，然后单击**编辑过滤器**。此时出现定义过滤器对话框。



- 2 在**运算符**列表中，选择所需的运算符。
- 3 在**值**框中，输入必须匹配的准则。**值**框不接受所选的查询无法处理的值。给字母数字列名使用 **Like** 与 **Not Like** 过滤器运算符时，**值**框接受以下通配符：

字符	描述
%	包含零个或多个字符的任何字符串。
_	任何单个字符。
[]	指定的范围（如 [a-f]）或集合（如 [abcdef]）内的任何单个字符。
[^]	指定的范围（如 [^a-f]）或集合（如 [^abcdef]）之外的任何单个字符。

“值”框的以下限制适用于不同的字段：

字段	限制
所有字段	除“用户 1”、“用户 2”以及“优先级”之外，接受所有的字母数字字符。
优先级	接受从 1 到 999 的整数值。
用户 1、用户 2	仅接受负数、正数或小数。

- 4 单击**确定**。

### 组合报警列

定义多个字段时，各个列使用布尔运算符进行合并。

- AND 运算符返回满足全部所选字段值的记录。
- OR 运算符返回满足任何所选字段值的记录。

要使用 AND/OR 运算符来设置过滤器选择准则，必须将相应的字段组合到一起。对于过滤器窗格中的某一项，只能创建一个过滤器表达式。如果需要多个表达式，则必须将该项目再次添加到过滤器窗格中。

缺省条件下，组合的字段使用 AND 运算符。

AND 与 OR 运算符是父节点。每个父节点下的所选字段都是子节点。您无法将字段从父节点拖放到子节点上。

#### 要组合多个报警列

- 1 使用鼠标右键单击字段，然后单击**组合**。
- 2 将一个字段拖放到另一个字段上。

### 复制或移动查询过滤器

如果有多个 Alarm DB View 控件的实例，并且希望给多个实例使用相同的过滤器，则可以将定义的过滤器从一个实例复制或剪切到另一个过滤器中。

#### 要复制过滤器

- 1 在 Alarm DB View 控件的第一个实例中定义过滤器。
- 2 使用鼠标右键单击这些过滤器，然后单击**复制**。要移动这些过滤器，请单击**剪切**。
- 3 关闭 Alarm DB View 控件的第一个实例。
- 4 打开 Alarm DB View 控件的下一个实例，然后单击**查询过滤器**选项卡。
- 5 将箭头放在右侧窗格中。使用鼠标右键单击所选的过滤器，然后单击**粘贴**。

状态列的值

将**状态**列添加到过滤器查询时，可以从**定义过滤器**对话框的**值**菜单中指定值。可用的值在下表中介绍：

值	描述
ACK	查询所有的系统确认。
ACK_ALM	查询所有已确认的报警。
UNACK_ALM	查询所有未确认的报警。
ACK_RTN	查询已返回到正常状态且已确认的所有报警。
UNACK_RTN	查询已返回到正常状态但尚未确认的报警。
All UNACK Records	查询未确认的所有记录。
All ACK Records	查询所有已确认的记录。
All ALM Records	查询所有的报警记录。
All RTN Records	查询已返回到正常状态的所有报警。

**备注** 扩展的摘要报警模式中的标记用于创建某个报警（该报警在确认主报警之后返回到正常状态）时，将创建两条记录。第一条记录是“已确认并返回到正常状态”记录，因为新报警已经在返回到正常状态下。第二条记录是已确认记录，它对应于确认主报警。以前对 **ACK\_ALM** 的实现已经更改为 **ACK**。



- 5 通过单击颜色框以打开调色板，为以下每一个选项配置颜色。

选项	描述
未确认报警前景色	给未确认的报警的每个颜色优先级范围设置前景色。
未确认报警背景色	给未确认的报警的每个颜色优先级范围设置背景色。
确认报警前景色	给已确认的报警的每个颜色优先级范围设置前景色。
确认报警背景色	给已确认的报警的每个颜色优先级范围设置背景色。

- 6 单击应用。

配置报警记录的排序顺序

您可以控制报警记录在控件中的排序方式。

- 1 使用鼠标右键单击 Alarm DB View 控件，然后单击属性。此时出现 AlmdbViewCtrl 属性对话框。
- 2 单击时间 / 排序选项卡。



- 3 在主排序列列表中，单击主排序列的名称。只有可见的列才出现在排序列列表中。如果没有看到所需的列，请单击常规选项卡，然后从列明细中选择该列。
- 4 在辅助排序列列表中，单击辅助排序列的名称。
- 5 选择升序或降序作为排序方向。
- 6 单击确定。

## 在运行时使用 Alarm DB View 控件

根据控件的配置，您可以：

- 检索与刷新数据。
- 给数据排序。
- 过滤数据。
- 选择行。
- 通过拖动列来更改它们的顺序。
- 将所有的列还原为设计时保存的设置。

### 给记录排序

您可以给显示对象中的记录排序。单击标题对所有的行进行排序。

使用鼠标右键单击控件，然后单击**排序**以打开**第二位排序**对话框，从中可以按升序或降序对单个及多个列进行排序。

要指定所要排序的列，请选择列名旁边的复选框。使用**排序顺序**方向键给各个列重新排序。

如果多个报警事件具有相同的时间标签，则它们可能不会按期望的顺序出现。

例如，如果所需的排序方式是先按照报警状态的降序进行排序，则需要：

- 1 选择**日期与状态**复选框。
- 2 选择**状态**行。
- 3 单击向上排序顺序箭头。
- 4 在**排序类型**区域中，单击**降序**。
- 5 单击**确定**。

### 理解状态栏信息

状态栏显示控件的当前状态。

- 左侧框架显示服务器名与连接的数据库。
- 中间框架显示查询返回的总记录数中显示了多少条记录。
- 右侧框架显示与服务器的连接状态。



## 使用 Alarm DB View ActiveX 属性

您可以使用脚本直接设置 Alarm DB View 控件属性的值，或者将它指定给一个 InTouch 标记或 I/O 引用。如需有关设置属性的详细信息，请参阅 *InTouch® HMI 脚本与逻辑指南* 中的第 8 章“编写 ActiveX 控件脚本”。

如需有关设置颜色值的详细信息，请参阅第 77 页的“配置 ActiveX 控件的颜色”。

### AckAlmBackColor 属性

获取或设置已确认报警的背景色。此设置会覆盖已确认报警的各个单独范围的颜色设置（AckAlmBackColorRange1 到 AckAlmBackColorRange4）。

#### 类型

整型

#### 缺省值

白色

#### 语法

*Object*.AckAlmBackColor [= color]

#### 值

*color*

确定背景色的值或常数。

### AckAlmBackColorRange1 属性

获取或设置已确认报警的背景色。此颜色应用于控件中显示的特定记录，其状态为 ACK\_ALM，且优先级在 1 到 ColorPriorityRange1 范围之内。

#### 类型

整型

#### 缺省值

白色

#### 语法

*Object*.AckAlmBackColorRange1 [= color]

#### 值

*color*

确定背景色的值或常数。

## AckAlmBackColorRange2 属性

获取或设置已确认报警的背景色。此颜色应用于控件中显示的特定记录，其状态为 ACK\_ALM，且优先级在 ColorPriorityRange1 到 ColorPriorityRange2 范围之内。

### 类型

整型

### 缺省值

白色

### 语法

*Object*.AckAlmBackColorRange2 [= color]

### 值

*color*

确定背景色的值或常数。

## AckAlmBackColorRange3 属性

获取或设置已确认报警的背景色。此颜色应用于控件中显示的特定记录，其状态为 ACK\_ALM，且优先级在 ColorPriorityRange2 到 ColorPriorityRange3 范围之内。

### 类型

整型

### 缺省值

白色

### 语法

*Object*.AckAlmBackColorRange3 [= color]

### 值

*color*

确定背景色的值或常数。

## AckAlmBackColorRange4 属性

获取或设置已确认报警的背景色。此颜色应用于控件中显示的特定记录，其状态为 ACK\_ALM，且优先级在 ColorPriorityRange3 到 999 范围之内。

### 类型

整型

### 缺省值

白色

### 语法

*Object*.AckAlmBackColorRange4 [= color]

### 值

*color*

确定背景色的值或常数。

## AckAlmForeColor 属性

获取或设置已确认报警的前景色。此设置会覆盖已确认报警的各个单独范围的颜色设置（AckAlmForeColorRange1 到 AckAlmForeColorRange4）。

### 类型

整型

### 缺省值

黑色

### 语法

*Object*.AckAlmForeColor [= color]

### 值

*color*

确定文本颜色的值或常数。

## AckAlmForeColorRange1 属性

获取或设置已确认报警的前景色。此颜色应用于控件中显示的特定记录，其状态为 ACK\_ALM，且优先级在 1 到 ColorPriorityRange1 范围之内。

### 类型

整型

### 缺省值

黑色

### 语法

*Object*.AckAlmForeColorRange1 [= color]

值

*color*

确定文本颜色的值或常数。

## AckAlmForeColorRange2 属性

获取或设置已确认报警的前景色。此颜色应用于控件中显示的特定记录，其状态为 ACK\_ALM，且优先级在 ColorPriorityRange1 到 ColorPriorityRange2 范围之间。

**类型**

整型

**缺省值**

黑色

**语法**

*Object*.AckAlmForeColorRange2 [= color]

值

*color*

确定文本颜色的值或常数。

## AckAlmForeColorRange3 属性

获取或设置已确认报警的前景色。此颜色应用于控件中显示的特定记录，其状态为 ACK\_ALM，且优先级在 ColorPriorityRange2 到 ColorPriorityRange3 范围之间。

**类型**

整型

**缺省值**

黑色

**语法**

*Object*.AckAlmForeColorRange3 [= color]

值

*color*

确定文本颜色的值或常数。

## AckAlmForeColorRange4 属性

获取或设置已确认报警的前景色。此颜色应用于控件中显示的特定记录，其状态为 ACK\_ALM，且优先级在 ColorPriorityRange3 到 999 范围之内。

### 类型

整型

### 缺省值

黑色

### 语法

*Object*.AckAlmForeColorRange4 [= color]

### 值

*color*

确定文本颜色的值或常数。

## AckRtnBackColor 属性

获取或设置返回到正常的已确认报警 (ACK\_RTN) 的背景色。

### 类型

整型

### 缺省值

白色

### 语法

*Object*.AckRtnBackColor [= color]

### 值

*color*

确定背景色的值或常数。

## AckRtnForeColor 属性

获取或设置返回到正常的已确认报警 (ACK\_RTN) 的文本颜色。

**类型**

整型

**缺省值**

蓝色

**语法**

*Object*.AckRtnForeColor [= color]

**值**

*color*

确定文本颜色的值或常数。

## AlmRtnBackColor 属性

获取或设置返回到正常的报警的背景色。此颜色应用于显示的状态为 ALM\_RTN 的记录。

**类型**

整型

**缺省值**

白色

**语法**

*Object*.AlmRtnBackColor [= color]

**值**

*color*

确定背景色的值或常数。

## AlmRtnForeColor 属性

获取或设置已返回的报警的前景色。此颜色应用于显示的状态为 ALM\_RTN 的记录。

**类型**

整型

**缺省值**

蓝色

**语法**

*Object*.AlmRtnForeColor [= color]

**值**

*color*

确定文本颜色的值或常数。

## AutoConnect 属性

获取或设置一个值，确定控件在运行时是否自动连接到数据库。

### 数据类型

整型

### 缺省值

False

### 语法

*Object*.AutoConnect [= *Integer*]

### 值

*Integer*

整型表达式，指定控件在运行时是否连接到数据库。

True = 连接到数据库。

False = （缺省值）不连接到数据库。

### 附注

您必须明确调用 Connect() 方法才能连接到数据库。

## ColorPriorityRange1 属性

设置要显示的报警的优先级范围边界。此属性的值必须大于 1 且小于 ColorPriorityRange2 的值。

### 类型

整型

### 缺省值

250

### 语法

*Object*.ColorPriorityRange1 [= integer or priority]

## ColorPriorityRange2 属性

设置要显示的报警的优先级范围边界。此属性的值必须大于 ColorPriorityRange1 的值且小于 ColorPriorityRange3 的值。

**类型**

整型

**缺省值**

500

**语法**

*Object.ColorPriorityRange2* [= integer or priority]

## ColorPriorityRange3 属性

设置要显示的报警的优先级范围边界。此属性的值必须大于 ColorPriorityRange2 的值且小于 999。

**类型**

整型

**缺省值**

750

**语法**

*Object.ColorPriorityRange3* [= integer or priority]

## ColumnResize 属性

返回或设置一个值，确定是否可以调整列宽。

**类型**

离散

**缺省值**

True

**语法**

*Object.ColumnResize* [= *Discrete*]

**值**

*Discrete*

True = （缺省值）列宽在运行时可以调整。

False = 列宽无法调整。



## 连接状态属性

返回连接的状态。此属性为只读。

### 数据类型

消息

### 语法

*Object*.ConnectStatus

### 值

已连接 = 控件已连接到数据库。

未连接 = 控件未连接到数据库。

进行中 = 控件正在连接到数据库。

### 示例

控件名为 AlmDbView1， Tagname 是一个消息标记。

Tagname = #AlmDbView1.ConnectStatus;

## CustomMessage 属性

获取或设置报警数据库中检索不到报警记录时 Alarm DB View 控件显示的消息。

### 类型

消息

### 缺省值

此视图中没有项目可显示。

### 语法

*Object*.CustomMessage [= string]

## DatabaseName 属性

指定要连接的数据库。

### 类型

消息

### 语法

*Object*.DatabaseName [= text]

## DisplayMode 属性

返回控件的显示模式，确定仅显示报警、事件，还是两者都显示。此属性为只读。

### 类型

字符串

### 缺省值

Alarms & Events History（报警与事件历史）

### 语法

*Object*.DisplayMode

### 附注

可能的值有：

Alarms & Events History（报警与事件历史）

Alarms History（报警历史）

Events History（事件历史）

### 示例

控件名为 AlmDbView1，tag 是一个消息标记。

```
tag = #AlmDbView1.DisplayMode;
```

## DisplayedTimeZone 属性

获取或设置显示的时区。

### 类型

字符串

### 缺省值

Local Time（本地时间）

### 语法

*Object*.DisplayedTimeZone [= message]

### 附注

可能的值有：

GMT - 报警时间标签使用“格林尼治标准时间”。

Local Time（本地时间）- 报警时间标签显示时使用 Alarm DB View 控件所在客户端的本地时间。

Origin Time（原始时间）- 报警时间标签显示时使用报警供应器的本地时间。

## Duration 属性

获取或设置用于设置开始与结束时间的持续时间。

### 类型

消息

### 缺省值

"Last Hour"（最近一小时）

### 语法

*Object*.Duration [= text]

### 值

*text*

包含持续时间的字符串表达式。

此属性必须包含以下字符串之一：

Last Minute（最近 1 分钟）

Last 5 Minutes（最近 5 分钟）

Last 15 Minutes（最近 15 分钟）

Last Half Hour（最近 30 分钟）

Last Hour（最近 1 小时）

Last 2 Hours（最近 2 小时）

Last 4 Hours（最近 4 小时）

Last 8 Hours（最近 8 小时）

Last 12 Hours（最近 12 小时）

Last Day（最近 1 天）

Last 2 Days（最近 2 天）

Last 3 Days（最近 3 天）

Last Week（最近 1 星期）

Last 2 Weeks（最近 2 星期）

Last 30 days（最近 30 天）

Last 90 days（最近 90 天）

## EndTime 属性

返回或设置结束时间。

**类型**

消息

**语法**

*Object*.EndTime [= *text*]

**值**

*text*

赋结束时间值的字符串表达式。返回的字符串总是采用 (MM/DD/YYYY HH:MM:SS) 格式。设置字符串值时也要求使用相同的格式。此属性可以处理任何时区中自 1970 年 1 月 1 日午夜到 2038 年 1 月 18 日 19:14:07 的日期。

## EventBackColor 属性

获取或设置事件报警的背景色。此颜色应用于控件中显示的状态为 EVT\_EVT 的记录。

**类型**

整型

**缺省值**

白色

**语法**

*Object*.EventBackColor [= *color*]

**值**

*color*

确定背景色的值或常数。

## EventForeColor 属性

获取或设置事件报警的前景色。此颜色应用于控件中显示的状态为 EVT\_EVT 的记录。

**类型**

整型

**缺省值**

红色

**语法**

*Object*.EventForeColor [= *color*]

**值**

*color*

确定文本颜色的值或常数。

## FilterFavoritesFile 属性

获取或设置过滤器收藏夹文件。此文件由**过滤器收藏夹**对话框用于读取或写入过滤器收藏夹。

**类型**

字符串

**缺省值**

空值

**语法**

*Object*.FilterFavoritesFile [= String]

## FilterMenu 属性

获取或设置一个值，确定是否在快捷菜单中显示**过滤器**菜单项。

**类型**

离散

**缺省值**

True

**语法**

*Object*.FilterMenu [= Discrete]

**值**

True = 显示**过滤器**菜单项（缺省值）。

False = 不显示**过滤器**菜单项。

## FilterName 属性

返回当前过滤器的名称（如果有）。

**类型**

字符串（只读）

**缺省值**

Null

**语法**

*Object*.FilterName [= String]

## FromPriority 属性

获取或设置控件的“从优先级”值。

### 类型

整型

### 缺省值

1

### 语法

*Object*.FromPriority [= integer]

### 附注

您可以使用此属性来过滤要显示哪些报警记录。例如，如果将此属性设置为 760，则仅显示优先级范围在 760 到 ToPriority 属性值之间的报警。

## GroupExactMatch 属性

GroupExactMatch 属性为真时，仅显示报警组名与 GroupName 属性值完全匹配的报警。当它为假时，组名只需指定要过滤的报警组名的一部分。

### 类型

离散

### 缺省值

False

### 语法

*Object*.GroupExactMatch [= discrete]

### 附注

将此属性与 GroupName 属性配合使用来过滤 Alarm DB View 控件。

### 示例

例如：

```
#AlarmDBViewCtrl1.GroupName = "Group"
#AlarmDBViewCtrl1.GroupExactMatch = 0;
#AlarmDBViewCtrl1.Refresh();
```

## GroupName 属性

获取或设置当前 Alarm DB View 控件报警组名过滤器。

### 类型

字符串

### 缺省值

(无)

### 语法

*Object*.GroupName [= GroupName]

### 附注

将此属性设置为 "GroupA"，重新查询时，显示对象中仅出现属于 GroupA 报警组的标记。

## MaxRecords 属性

返回或设置要检索的最大记录数。

### 类型

整型

### 缺省值

100

### 语法

*Object*.MaxRecords [=integer]

### 值

*integer*

整型表达式，指定要在给定的时间检索的记录数。最大记录数可以在从 1 到 1000 的范围内。为获取最佳性能，应根据需要尽可能将此值设置得小一些。

## Password 属性

返回或设置检索数据的 SQL Server 口令。

### 类型

消息

### 语法

*Object*.Password [= text]

### 值

*text*

赋口令值的字符串表达式。

## PrimarySort 属性

获取或设置用于给报警显示对象排序的主列名。

**类型**

消息

**缺省值**

(无)

**语法**

*Object*.PrimarySort [= message]

## ProviderExactMatch 属性

ProviderExactMatch 属性为真时，仅显示报警供应器名与 ProviderName 属性值完全匹配的的报警。当它为假时，供应器名只需指定要过滤的报警供应器名的一部分。

**类型**

离散

**缺省值**

False

**语法**

*Object*.ProviderExactMatch [= discrete]

**附注**

将此属性与 ProviderName 属性配合使用来过滤 Alarm DB View 控件。

**示例**

例如：

```
#AlarmDBViewCtrl1.ProviderName = "Provider"
```

```
#AlarmDBViewCtrl1.ProviderExactMatch = 0;
```

```
#AlarmDBViewCtrl1.Refresh();
```



## ProviderName 属性

获取或设置当前 Alarm DB View 控件的报警供应器名过滤器。

**类型**

字符串

**缺省值**

(无)

**语法**

*Object*.ProviderName [= ProviderName]

**附注**

如果某个标记属于 Provider1 报警供应器，则将此属性设置为 "Provider1" 并重新查询时，显示对象中仅出现属于 Provider1 报警供应器的标记。

## QueryTimeZoneName 属性

获取或设置给查询使用特定时间时的时区。

**类型**

离散

**缺省值**

False

**语法**

*Object*.QueryTimeZone [= Discrete]

**值**

True = GMT

False = 原始时间，即报警供应器的本地时间。

## RefreshMenu 属性

获取或设置一个值，确定是否在快捷菜单中显示刷新菜单项。

**类型**

离散

**缺省值**

True

**语法**

*Object*.RefreshMenu [= Discrete]

**值**

True = 显示刷新菜单项（缺省值）。

False = 不显示刷新菜单项。

## ResetMenu 属性

获取或设置一个值，确定是否在快捷菜单中显示**重置**菜单项。

**类型**

离散

**缺省值**

True

**语法**

*Object*.ResetMenu [= Discrete]

**值**

True = 显示**重置**菜单项（缺省值）。

False = 不显示**重置**菜单项。

## RowCount 属性

返回控件中显示的记录数。此属性为只读。

**类型**

整型

**语法**

*Object*.RowCount

**示例**

控件名为 AlmDbView1，tagname 是一个整型标记。

tagname = #AlmDbView1.RowCount;

## RowSelection 属性

返回或设置一个值，确定在运行时是否可以选择行。

**类型**

离散

**缺省值**

True

**语法**

*Object*.RowSelection [= Discrete]

**值**

*Discrete*

True = （缺省值）允许选择行。

False = 不允许选择行。

**附注**

如果不允许选择行，则不会生成 Click 或 DoubleClick 事件。

## SecondarySort 属性

获取或设置用于给报警显示对象排序的辅助列名。

**类型**

消息

**缺省值**

(无)

**语法**

*Object.SecondarySort [= text]*

## ServerName 属性

返回或设置控件检索数据时连接的服务器的名称。

**类型**

消息

**语法**

*Object.ServerName [= text]*

## ShowFetch 属性

返回或设置一个值，确定是否显示检索按钮。

**类型**

离散

**缺省值**

True

**语法**

*Object.ShowFetch [= Discrete]*

**值**

*Discrete*

True = (缺省值) 显示检索按钮。

False = 不显示检索按钮。

## ShowGrid 属性

返回或设置一个值，确定是否显示网格线。

**类型**

离散

**缺省值**

False

**语法**

*Object.ShowGrid [= Discrete]*

**值**

*Discrete*

True = 显示网格线。

False = （缺省值）不显示网格线。

## ShowHeading 属性

返回或设置一个值，确定是否显示列标题。

**类型**

离散

**缺省值**

True

**语法**

*Object.ShowHeading [= Discrete]*

**值**

*Discrete*

True = （缺省值）显示列标题。

False = 不显示列标题。

## ShowMessage 属性

确定报警数据库中是否有记录时是否显示自定义消息 “此视图没有项目可显示”。

**类型**

离散

**缺省值**

False

**语法**

*Object.ShowMessage [= discrete]*

## ShowStatusBar 属性

返回或设置一个值，确定是否显示状态栏。

**类型**

离散

**缺省值**

True

**语法**

*Object.ShowStatusBar [= Discrete]*

**值**

*Discrete*

True = （缺省值）显示状态栏。

False = 不显示状态栏。

## SilentMode 属性

获取或设置一个值，确定控件是否处于“无提示”模式。

**类型**

离散

**缺省值**

False

**语法**

*Object.SilentMode [= Discrete]*

**值**

True = 提示模式打开无。

False = 无提示模式关闭（缺省值）。

## SortMenu 属性

返回或设置一个值，确定是否在快捷菜单中显示**排序**菜单项。

**类型**

离散

**缺省值**

True

**语法**

*Object.SortMenu [= Discrete]*

**值**

离散表达式。

True = （缺省值）显示**排序**菜单项。

False = 不显示**排序**菜单项。

## SortOrder 属性

获取或设置报警的排序顺序，这取决于要排序的列（主排序列）。

**类型**

离散

**缺省值**

True

**语法**

*Object.SortOrder [= discrete]*

**值**

离散表达式。

True = 升序。

False = 降序。

## SpecificTime 属性

返回或设置一个值，确定控件是使用 StartTime 与 EndTime 属性，还是根据 Duration 属性的值来计算开始时间与结束时间。

**类型**

离散

**缺省值**

False

**语法**

*Object.SpecificTime [= Discrete]*

**值**

True = 使用 StartTime 与 EndTime 属性。

False = （缺省值）根据 "Duration" 属性计算 StartTime 与 EndTime。

## StartTime 属性

返回或设置开始时间。

**类型**

消息

**语法**

*Object*.StartTime [= *text*]

**值**

*text*

赋 StartTime 值的字符串表达式。返回的字符串总是采用 (MM/DD/YYYY HH:MM:SS) 格式。设置字符串值时也要求使用相同的格式。此属性可以处理任何时区中自 1970 年 1 月 1 日午夜到 2038 年 1 月 18 日 19:14:07 的日期。

## Time 属性

获取与设置要在显示对象中使用的时间格式。

**类型**

消息

**缺省值**

%m/%d/%Y %I:%M:%S %p

**语法**

*Object*.Time [= *message*]

**附注**

如需有关时间格式字符串的详细信息，请参阅第 277 页的“配置为报警记录显示的时间格式与时区”。

## ToPriority 属性

获取或设置控件的“到优先级”值。

**类型**

整型

**缺省值**

999

**语法**

*Object*.ToPriority [= *integer*]

**附注**

使用此属性过滤要显示哪些报警记录。例如，如果将此属性设置为 900，则仅显示优先级范围从 FromPriority 属性值到 900 之间的报警。

## TotalRowCount 属性

返回当前查询的总记录数。此属性为只读。

### 类型

整型

### 语法

*Object*.TotalRowCount

### 附注

行计数指当前查询中返回的行数，通常与 MaxRecords 属性相同，除非检索到的记录数小于 MaxRecords 属性。例如，如果符合特定准则的记录有 950 条，而 MaxRecords 属性为 100，则最后一页有 50 条记录，行数为 50。在相同的示例中，TotalRowCount 属性始终是 950。

### 示例

控件名为 AlmDbView1，tagname 是一个整型标记。

```
tagname = #AlmDbView1.TotalRowCount;
```

## UnAckAlmBackColor 属性

获取或设置未确认的报警的背景色。此颜色应用于控件中显示的状态为 UNACK\_ALM 的所有记录。它会覆盖

UnAckAlmBackColorRange1 到

UnAckAlmBackColorRange4 属性值所作的任何设置。

### 类型

整型

### 缺省值

白色

### 语法

*Object*.UnAckAlmBackColor [= color]

### 值

*color*

确定特定对象颜色的值或常数。



## UnAckAlmBackColorRange1 属性

获取或设置未确认的报警的背景色。此颜色应用于控件中显示的特定记录，其状态为 UNACK\_ALM，且优先级在 1 到 ColorPriorityRange1 范围之内。

### 类型

整型

### 缺省值

白色

### 语法

*Object*.UnAckAlmBackColorRange1 [= color]

### 值

*color*

确定特定对象颜色的值或常数。

## UnAckAlmBackColorRange2 属性

获取或设置未确认的报警的背景色。此颜色应用于控件中显示的特定记录，其状态为 UNACK\_ALM，且优先级在 ColorPriorityRange1 到 ColorPriorityRange2 范围之内。

### 类型

整型

### 缺省值

白色

### 语法

*Object*.UnAckAlmBackColorRange2 [= color]

### 值

*color*

确定特定对象颜色的值或常数。

## UnAckAlmBackColorRange3 属性

获取或设置未确认的报警的背景色。此颜色应用于控件中显示的特定记录，其状态为 UNACK\_ALM，且优先级在 ColorPriorityRange2 到 ColorPriorityRange3 范围之间。

### 类型

整型

### 缺省值

白色

### 语法

*Object*.UnAckAlmBackColorRange3 [= color]

### 值

*color*

确定特定对象颜色的值或常数。

## UnAckAlmBackColorRange4 属性

获取或设置未确认的报警的背景色。此颜色应用于控件中显示的特定记录，其状态为 UNACK\_ALM，且优先级在 ColorPriorityRange3 到 999 范围之间。

### 类型

整型

### 缺省值

白色

### 语法

*Object*.UnAckAlmBackColorRange4 [= color]

### 值

*color*

确定特定对象颜色的值或常数。

## UnAckAlmForeColor 属性

获取或设置未确认的报警的文本颜色。此颜色应用于控件中显示的状态为 UNACK\_ALM 的所有记录。它会覆盖

UnAckAlmForeColorRange1 到 UnAckAlmForeColorRange4 属性值所作的任何设置。

### 类型

整型

### 缺省值

红色

### 语法

*Object*.UnAckAlmBackColor [= color]

### 值

*color*

确定文本颜色的值或常数。

## UnAckAlmForeColorRange1 属性

获取或设置未确认的报警的前景色。此颜色应用于控件中显示的特定记录，其状态为 UNACK\_ALM，且优先级在 1 到

ColorPriorityRange1 范围之内。

### 类型

整型

### 缺省值

红色

### 语法

*Object*.UnAckAlmForeColorRange1 [= color]

### 值

*color*

确定特定对象颜色的值或常数。

## UnAckAlmForeColorRange2 属性

获取或设置未确认的报警的前景色。此颜色应用于控件中显示的特定记录，其状态为 UNACK\_ALM，且优先级在 ColorPriorityRange1 到 ColorPriorityRange2 范围之间。

### 类型

整型

### 缺省值

红色

### 语法

*Object*.UnAckAlmForeColorRange2 [= color]

### 值

*color*

确定特定对象颜色的值或常数。

## UnAckAlmForeColorRange3 属性

获取或设置未确认的报警的前景色。此颜色应用于控件中显示的特定记录，其状态为 UNACK\_ALM，且优先级在 ColorPriorityRange2 到 ColorPriorityRange3 范围之间。

### 类型

整型

### 缺省值

红色

### 语法

*Object*.UnAckAlmForeColorRange3 [= color]

### 值

*color*

确定特定对象颜色的值或常数。

## UnAckAlmForeColorRange4 属性

获取或设置未确认的报警的前景色。此颜色应用于控件中显示的特定记录，其状态为 UNACK\_ALM，且优先级在 ColorPriorityRange3 到 999 范围之内。

### 类型

整型

### 缺省值

红色

### 语法

*Object*.UnAckAlmForeColorRange4 [= color]

### 值

*color*

确定特定对象颜色的值或常数。

## UnAckOrAlarmDuration 属性

持续时间列显示“未确认持续时间”或“报警持续时间”。如果为 FALSE (0)，则显示“未确认持续时间”；如果为 TRUE (1)，则显示“报警持续时间”。

### 类型

整型

### 缺省值

False

### 语法

*Object*.UnAckOrAlarmDuration [= integer]

## UserID 属性

返回或设置用于连接到 SQL Server 以检索数据的用户 ID。

### 类型

消息

### 语法

*Object*.UserID [= text]

### 值

*text*

赋用户 ID 值的字符串表达式。

## 使用 Alarm DB View ActiveX 方法

使用 Alarm DB View ActiveX 方法可以：

- 控制数据库连接。
- 从报警数据库中检索记录。
- 检索报警的有关信息。
- 重置显示外观。
- 给报警记录排序。
- 显示上下文菜单。
- 访问过滤器收藏夹。

如需有关调用方法的详细信息，请参阅 *InTouch® HMI 脚本与逻辑指南* 中的第 8 章 “编写 ActiveX 控件脚本”。

### 控制数据库连接

使用 `Connect()` 方法连接到报警数据库，使用 `Disconnect()` 方法断开连接。

#### Connect() 方法

将控件连接到数据库；如果连接成功，则显示范围从 1 到 `MaxRecords` 之间的一组记录。

##### 语法

`Object.Connect`

##### 示例

控件名为 `AlmDbView1`。

```
#AlmDbView1.Connect();
```

#### Disconnect() 方法

断开控件与数据库的连接。

##### 语法

`Object.Disconnect`

##### 示例

控件名为 `AlmDbView1`。

```
#AlmDbView1.Disconnect();
```

## 从数据库检索记录

使用以下方法选择查询、从数据库中检索记录以及刷新显示对象：

- `SelectQuery()` 方法
- `GetPrevious()` 方法
- `GetNext()` 方法
- `Refresh()` 方法

### SelectQuery() 方法

将当前显示对象设置为 .xml 文件中指定的查询名。

#### 语法

```
Object.SelectQuery( "QueryName" );
```

#### 参数

##### QueryName

过滤器收藏夹文件中定义的查询的名称。

#### 示例

本例应用 "HighPriority" 查询定义的过滤器准则，此查询保存在当前与 AlmDbView1 控件关联的过滤器收藏夹文件中。

```
#AlmDbView1.SelectQuery("HighPriority");
```

### GetPrevious() 方法

从数据库中检索上一组记录（如果有）。

#### 语法

```
Object.GetPrevious();
```

#### 示例

控件名为 AlmDbView1。

```
#AlmDbView1.GetPrevious();
```

### GetNext() 方法

从数据库中检索下一组记录（如果有）。

#### 语法

*Object*.GetNext

#### 示例

控件名为 AlmDbView1。

```
#AlmDbView1.GetNext();
```

### Refresh() 方法

使用数据库中的数据来刷新控件；如果连接成功，则显示范围从 1 到 MaxRecords 之间一组记录。

#### 语法

*Object*.Refresh

#### 附注

通过调用 Alarm DB View 控件的 Refresh() 方法启动刷新操作之后，RowCount 与 TotalRowCount 属性的值更改为 -1，直至完成刷新（也就是，从数据库中检索到所有相关的记录之后）。完成刷新时，这两个属性都使用正确的当前行数进行更新。

Refresh() 方法采用异步工作方式 - 它立即将控制权返回给发出调用得脚本，然后在后台继续工作。这表示，在调用 Refresh() 方法之后查询 RowCount 与 TotalRowCount 的值时，很可能会返回 -1；原因在于，这是在尚未完成刷新的情况下查询它们的值。获取正确值的一种方法是：使用脚本确定属性值何时从 -1 更改为其它值，这样就可以知道现在有正确的值提供。

#### 示例

控件名为 AlmDbView1。

```
#AlmDbView1.Refresh();
```



## 检索报警的有关信息

使用以下方法从数据库中检索与特定报警有关的记录：

- GetItem() 方法
- GetSelectedItem() 方法

### GetItem() 方法

将指定的行与列的数据作为字符串来返回。

#### 语法

*Object.GetItem(Integer, message)*

#### 参数

##### *Integer*

整型表达式，赋值为给控件中特定的行。

##### *Message*

字符串表达式，赋值为控件中的列名。

#### 示例

控件名为 AlmDbView1， tag 定义为消息标记。

```
tag = #AlmDbView1.GetItem(1, "Group");
```

### GetSelectedItem() 方法

将所选行、给定列的数据作为字符串来返回。

#### 语法

*Object.GetSelectedItem(message)*

#### 参数

##### *Message*

赋控件中列名值的字符串表达式。

#### 示例

控件名为 AlmDbView1， tag 定义为消息标记。

```
tag = #AlmDbView1.GetSelectedItem("State");
```

## 给报警记录排序

使用以下方法给报警记录排序及重置列宽的调整：

- SortOnCol() 方法
- ShowSort() 方法
- Reset() 方法

### SortOnCol() 方法

在显示的报警记录上执行主排序。

#### 语法

*Object.SortOnCol(message, Integer)*

#### 参数

*Message*

赋控件中列名值的字符串表达式。

*Integer*

要使用的排序方向。0 = 升序，1 = 降序。

#### 示例

控件名为 AlmDbView1。

```
tag = #AlmDbView1.SortOnCol("Name",1);
```

### ShowSort() 方法

如果启用 SortMenu 属性，则显示**第二位排序**对话框。

#### 语法

*Object.ShowSort*

#### 示例

控件名为 AlmDbView1。

```
#AlmDbView1.ShowSort();
```

### Reset() 方法

将所有的列还原为设计时保存的设置。

#### 语法

*Object.Reset*

#### 示例

控件名为 AlmDbView1。

```
#AlmDbView1.Reset();
```

## 显示上下文菜单

使用 ShowContext() 方法显示快捷菜单。

### ShowContext() 方法

如果启用 RefreshMenu、ResetMenu 或 SortMenu 属性中的任何一个，则显示快捷菜单。

#### 语法

*Object*.ShowContext

#### 示例

控件名为 AlmDbView1。

```
#AlmDbView1.ShowContext();
```

## 访问过滤器收藏夹

使用 ShowFilter() 方法显示过滤器收藏夹对话框。

### ShowFilter() 方法

显示过滤器收藏夹对话框。

#### 语法

*Object*.ShowFilter

#### 示例

控件名为 AlmDbView1。

```
#AlmDbView1.ShowFilter();
```

## 显示其它信息

使用 AboutBox() 方法显示关于对话框。

### AboutBox() 方法

显示关于对话框。

#### 语法

*Object*.AboutBox

#### 示例

控件名为 AlmDbView1。

```
#AlmDbView1.AboutBox();
```

## 使用方法与属性时的错误处理

使用 `SilentMode` 属性确定控件是否处于无提示模式。控件处于无提示模式时，不显示错误消息。要查看错误，请调用 `GetLastError()` 方法来返回错误消息。

### GetLastError() 方法

如果 Alarm DB View 控件处于无提示模式，则返回上一条错误消息。

#### 语法

`Object.GetLastError()`

#### 示例

控件名为 `AlmDbView1`，`Tagname` 定义为变量或字符串。

```
Tagname = #AlmDbView1.GetLastError();
```

## 使用 Alarm DB View ActiveX 事件触发脚本

您可以将 `QuickScript` 指定给 Alarm DB View 控件事件（如鼠标单击或双击）。该事件发生时，此 `QuickScript` 便会运行。

Alarm DB View 控件支持以下事件：

- Click
- DoubleClick
- ShutDown
- StartUp

Click 事件有一个参数 `ClicknRow`，它可以确定运行时所单击的行。

DoubleClick 事件有一个参数 `DoubleClicknRow`，它可以确定运行时所双击的行。

Click 与 DoubleClick 事件都是零基的。发布 Click 与 / 或 DoubleClick 事件时，显示对象中的条形计数从零开始。

---

**备注** Alarm DB View 控件从 `StartUp` 事件调用方法时会忽略用户界面方法，原因在于控件尚不可见。这些方法包括：`ShowSort()`、`ShowContext()`、`GetSelectedItem()`、`GetNext()`、`GetPrevious()` 及 `AboutBox()`。

---

如需有关编写 ActiveX 事件脚本的详细信息，请参阅 *InTouch® HMI 脚本与逻辑指南* 中的第 8 章“编写 ActiveX 控件脚本”。

# 第 11 章

## 分析标记间的报警分布

通过使用 Alarm Pareto ActiveX 控件，可以分析给定的生产系统中有哪些报警与事件最经常发生。您也可以按报警发生的时段来分析报警频率。

Alarm Pareto 控件的分析功能可以确定生产系统中最大的问题。Alarm Pareto 控件帮助确定应该将精力集中到哪些地方才能取得最显著的改善。

Alarm Pareto 控件显示一个代表报警活动的条形图。

如需有关 ActiveX 控件的详细信息，请参阅 *InTouch® HMI 可视化指南* 中的第 6 章 “ActiveX 控件”。

### 配置 Alarm Pareto ActiveX 控件

配置 Alarm Pareto 控件时，可以指定：

- 与报警数据库的连接。
- 巴累托控件的外观，包括颜色与字体。
- 用户可以在运行时访问的功能。
- 图表中显示的报警及其显示方式。

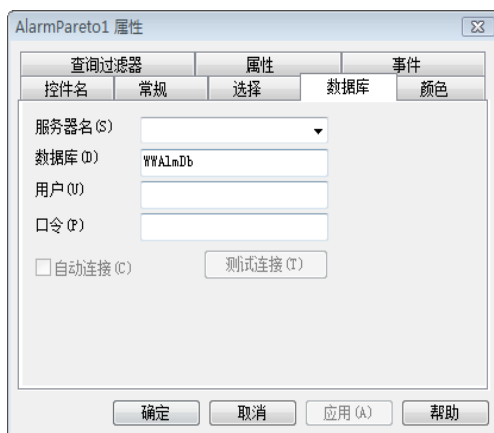
## 配置与报警数据库的连接

您必须配置 Alarm Pareto 控件与报警数据库之间的连接。

使用对报警数据库有适当的只读访问权限的帐户，而不是系统管理员帐户，这是一种很好的做法。

### 要配置与报警数据库的连接

- 1 使用鼠标右键单击 Alarm Pareto 控件，然后单击**属性**。此时出现 **AlarmPareto 属性** 对话框。
- 2 单击**数据库**选项卡。



- 3 配置连接。执行以下操作：
  - a 在**服务器名**框中，输入安装了报警数据库的计算机的节点名。
  - b 在**数据库**框中，输入报警数据库的名称。
  - c 在**用户**框中，输入报警数据库的有效用户帐户名。
  - d 在**口令**框中，输入与报警数据库用户帐户关联的口令。
- 4 如果希望 Alarm Pareto 控件在 WindowViewer 启动时立即自动连接到报警数据库，请选择**自动连接**复选框。  
如果没有选择**自动连接**复选框，则必须明确调用 Connect() 方法，将 Alarm Pareto 控件配置为连接到报警数据库。如需有关 Connect 方法的详细信息，请参阅第 342 页的“Connect() 方法”。
- 5 单击**测试连接**以验证与报警数据库的连接。此时出现一条消息，指出连接成功。
- 6 单击**应用**。

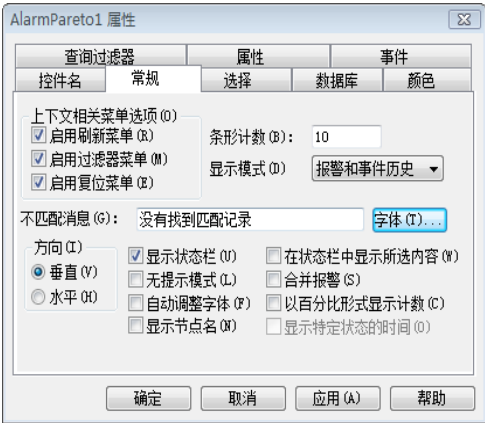
## 配置 Alarm Pareto 控件的外观与颜色

您可以配置 Alarm Pareto 控件的视觉外观。您可以：

- 包含状态栏。
- 设置“巴累托”条形图的方向。
- 包含条形图的描述。
- 选择 Alarm Pareto 图表的颜色。

### 要设置 Alarm Pareto 控件的外观

- 1 使用鼠标右键单击 Alarm Pareto 控件，然后单击**属性**。此时出现 **AlarmPareto 属性**对话框。
- 2 单击**常规**选项卡。

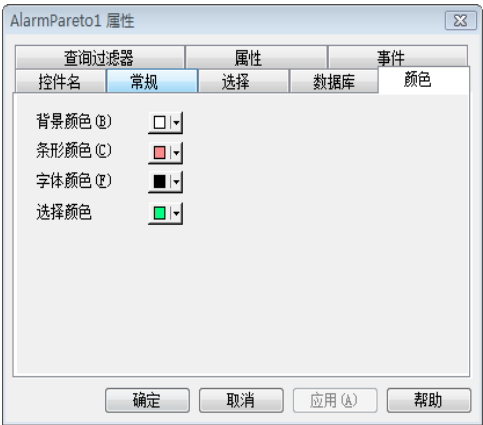


- 3 配置常规选项。执行以下操作：

属性	描述
条形计数	设置要在 Alarm Pareto 控件中查看的条形数。
显示模式	此列表显示可用的视图选项。选项有“报警和事件历史”、“报警历史”以及“事件历史”。
不匹配消息	设置在 Alarm Pareto 控件中没有要处理的数据时显示的消息。
垂直	沿垂直方向显示条形。
水平	沿水平方向显示条形。
显示状态栏	启用状态栏。

属性	描述
无提示模式	Alarm Pareto 控件不显示运行时错误消息。如果未选择它，则报警显示对象显示错误消息。错误消息总是发送到 <b>Logger</b> 中。
自动调整字体	可用空间不足而无法在所选的条形上正确显示文本时，“自动调整字体”功能会隐藏文本，并且仅在选择该条形时才显示文本。
显示节点名	在条形图上显示节点名。
在状态栏中显示所选内容	在状态栏上显示所选条形的描述。
合并报警	将报警状态合并为两个状态。例如，如果某个模拟标记有三个报警状态：Hi、HiHi 及 Normal，则 Hi 与 HiHi 报警状态会划分为一个状态。
以百分比形式显示计数	根据计数除以总数得到的百分比来显示条形。
显示特定状态的时间	根据每个标记处于报警状态的时间来显示 Alarm Pareto 控件。仅当显示模式设置为“报警历史”时，此选项才会启用。

- 4 单击**应用**。
- 5 单击**颜色**选项卡。



- 6 单击每个颜色框以打开调色板。



7 单击要给以下每个图表属性指定的颜色：

属性	描述
背景颜色	设置 Alarm Pareto 图的背景色
条形颜色	设置图表的条形颜色
字体颜色	设置图表中出现的文本的颜色。
选择颜色	设置所选条形的颜色

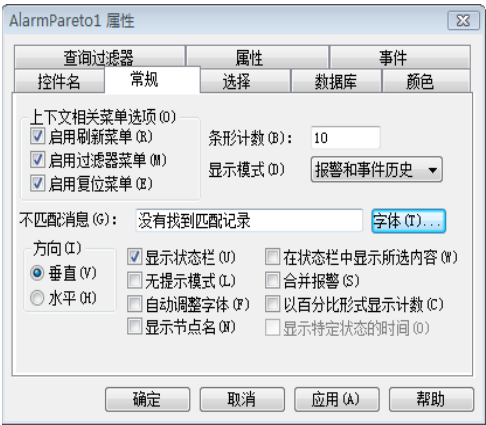
8 单击应用。

配置显示字体

您可以给 Alarm Pareto 图中出现的文本指定字体属性。

要配置字体属性

- 1 使用鼠标右键单击 Alarm Pareto 控件，然后单击**属性**。此时出现 **AlarmPareto 属性**对话框。
- 2 单击**常规**选项卡。



- 3 单击**字体**。此时出现标准的 Windows 字体对话框。配置字体，然后单击**确定**。
- 4 单击**确定**。

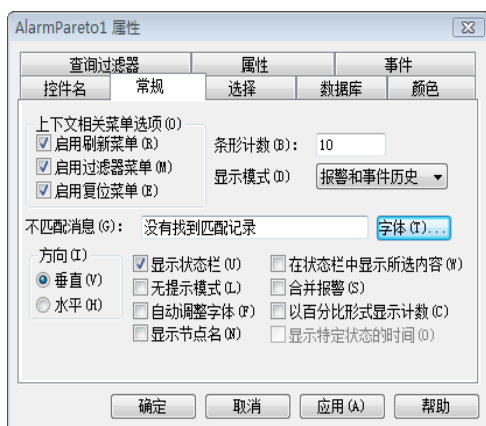
## 配置用户可以在运行时访问的功能

Alarm Pareto 图包含一个快捷菜单。操作员在运行时使用鼠标右键单击趋势时，会出现一个菜单，其中包含的选项可动态更新趋势中显示的数据。



### 要配置运行时功能

- 1 使用鼠标右键单击 Alarm Pareto 控件，然后单击**属性**。此时出现 **AlarmPareto 属性** 对话框。
- 2 单击**常规**选项卡。



- 3 在上下文相关菜单选项区域中，配置菜单命令：
  - a 选择**启用刷新菜单**复选框，以允许运行时用户刷新 Alarm Pareto 趋势中显示的数据，并显示范围在 1 到 MaxRecords 属性所定义的数字之间的记录。
  - b 选择**启用过滤器菜单**复选框，以允许用户显示**过滤器收藏夹**对话框，用于选择包含 Alarm Pareto 趋势数据库查询值的文件。
  - c 选择**启用复位菜单**复选框，以允许用户将运行时的 Alarm Pareto 图恢复为从 WindowMaker 中指定的原始值。操作员在运行时所作的全部更改都会恢复成设计时的原始值。
- 4 单击**应用**。

## 配置要分析的报警

您可以配置要使用 Alarm Pareto 图表来分析的报警。您可以指定：

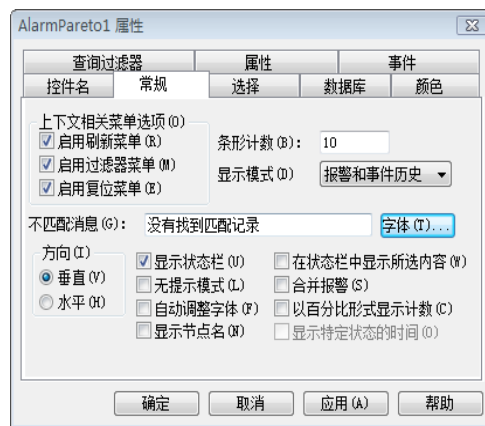
- 数据库数据的类型（报警或事件数据）
- 从中选择记录的时间段
- 过滤数据的准则

### 选择报警或事件数据

您可以配置在 Alarm Pareto 图中显示报警记录、事件记录，还是这两者都显示。

#### 要选择数据类型

- 1 使用鼠标右键单击 Alarm Pareto 控件，然后单击**属性**。此时出现 **AlarmPareto 属性** 对话框。
- 2 单击**常规**选项卡。



- 3 在**显示模式**列表中，配置记录类型。执行以下任何操作。
  - 单击**报警和事件历史**，以同时显示报警与事件的历史数据库记录。
  - 单击**报警历史**以便仅显示历史报警记录。
  - 单击**事件历史**以便仅显示历史事件记录。
- 4 单击**应用**。

## 选择时段

您可以设置查询值来根据所选的时间选择记录。您也可以配置要查看的最大记录数、报警查询的开始与结束时间，以及查询时区。

### 要选择数据的时段

- 1 使用鼠标右键单击 Alarm Pareto 控件，然后单击**属性**。此时出现 **AlarmPareto 属性** 对话框。
- 2 单击**选择**选项卡。



- 3 要使用预定义的时间间隔（查询数据时总是使用 UTC），请单击**持续时间**列表中的某个间隔。
- 4 要使用特定的开始时间与结束时间，请单击**使用指定时间**，然后配置详细资料。
  - a 在**开始时间**框中，输入检索报警记录的开始时间。字符串必须采用 YYYY/MM/DD HH:MM:SS 格式。使用任何时区中自 1970 年 1 月 1 日午夜到 2038 年 1 月 18 日 19:14:07 之间的任何日期。
  - b 在**结束时间**框中，输入停止检索报警记录的结束时间。字符串必须采用 YYYY/MM/DD HH:MM:SS 格式。使用任何时区中自 1970 年 1 月 1 日午夜到 2038 年 1 月 18 日 19:14:07 之间的任何日期。
  - c 在**查询时区**区域中，单击 **UTC** 或**原始时间**。UTC 时间是“格林尼治标准时间”，也称作“通用协调时间”或 Zulu。原始时间指操作员所在时区的当前时间。
- 5 在**最大记录数**框中，输入在一个实例上可以从控件中查看的记录数。最大记录数的有效范围是从 0 到 1000000。
- 6 单击**应用**。

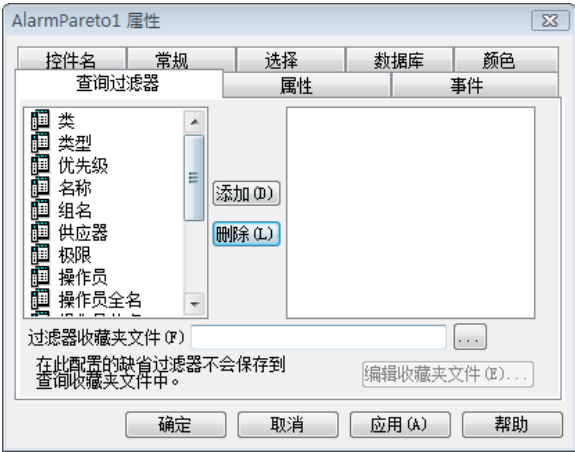
## 使用过滤器收藏夹创建自定义过滤器

您可以选择要在查询结果中包含哪些记录。例如，您可以按记录日期或报警状态来选择过滤器。您可以选择多个字段来限制或扩展查询结果。

如果没有定义自定义过滤器，则使用查询所有记录的缺省过滤器。

### 要创建自定义过滤器

- 1 使用鼠标右键单击 Alarm Pareto 控件，然后单击**属性**。此时出现 **AlarmPareto 属性**对话框。
- 2 单击**查询过滤器**选项卡。



- 3 在左侧窗格中，选择过滤器字段，然后单击**添加**，以便将它们包含到右侧窗格显示的过滤器中。这些过滤器字段在下表中介绍：

字段名	查询过滤准则：
类	报警类。
类型	报警类型。
优先级	报警优先级。
名称	报警名。
组名	报警组名。
供应器	报警供应器。
限制	报警限。这些值是字母数字字符。在“查询过滤器”中，这些值的比较是当作字符串比较来完成的。
操作员	操作员。
操作员全名	操作员的全名。
操作员节点	与报警关联的操作员的节点名。

字段名	查询过滤准则：
操作员域	与报警关联的操作员的域名。
注释	报警注释。
用户 1	用户自定义的报警数值 1。
用户 2	用户自定义的报警数值 2。
用户 3	用户自定义的报警字符串值。
持续时间	未确认与报警持续时间。“持续时间”列设置为零时，不会在查询中产生任何包含空值的记录。

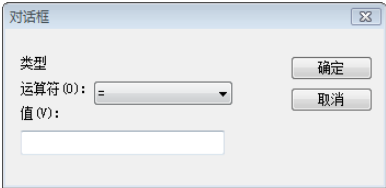
- 4 要从过滤器窗格中删除某个字段，请单击要删除的字段，然后单击**删除**。删除过滤器的操作无法撤消。出现消息时，单击**是**。
- 5 配置每个过滤器字段的准则。如需有关详细信息，请参阅第 335 页的“定义列过滤器准则”。
- 6 配置运算符与过滤器的组合。如需有关详细信息，请参阅第 336 页的“组合报警列”。
- 7 配置过滤器收藏夹文件。
  - a 在**过滤器收藏夹文件**框中，输入网络路径与文件名，或单击省略号按钮以浏览文件。
  - b 要编辑**过滤器收藏夹文件**，请单击**编辑收藏夹文件**按钮。此时打开**过滤器收藏夹**窗口，可供您在收藏夹文件中添加、修改或删除过滤器。完成时单击**确定**，以保存更改并关闭窗口。
- 8 单击**应用**。

定义列过滤器准则

对于查询中包含的每个列过滤器，都必须配置过滤器准则。例如，您可能只希望查看特定操作员的报警。

要定义列过滤器

- 1 使用鼠标右键单击字段，然后单击**编辑过滤器**。此时出现对话框对话框。



- 2 在**运算符**列表中，选择所需的运算符。
- 3 在**值**框中，输入必须匹配的准则。**值**框不接受所选的查询无法处理的值。给字母数字列名使用 **Like** 与 **Not Like** 过滤器运算符时，**值**框接受以下通配符：

字符	查找
%	包含零个或多个字符的任何字符串
_	任何单个字符
[]	指定的范围（如 [a-f]）或集合（如 [abcdef]）内的任何单个字符。
[^]	指定的范围（如 [^a-f]）或集合（如 [^abcdef]）之外的任何单个字符。

“值”框的以下限制适用于不同的字段：

字段	限制
所有字段	除“用户 1”、“用户 2”以及“优先级”之外，接受所有的字母数字字符。
优先级	接受从 1 到 999 的整数值。
用户 1、 用户 2	仅接受负数、正数或小数。

- 4 单击**确定**。

### 组合报警列

定义多个字段时，各个列使用布尔运算符进行合并。

- AND 运算符返回满足全部所选字段值的记录。
- OR 运算符返回满足任何所选字段值的记录。

要使用 AND/OR 运算符来设置过滤器选择准则，必须将相应的字段组合到一起。对于过滤器窗格中的某一项，只能创建一个过滤器表达式。如果需要多个表达式，则必须将该项目再次添加到过滤器窗格中。

缺省条件下，组合的字段使用 AND 运算符。

AND 与 OR 运算符是父节点。每个父节点下的所选字段都是子节点。您无法将字段从父节点拖放到子节点上。

### 要组合多个报警列

- 1 使用鼠标右键单击字段，然后单击**组合**。
- 2 将一个字段拖放到另一个字段上。

### 复制或移动查询过滤器

如果有多个 Alarm Pareto 控件的实例，并且希望给多个实例使用相同的过滤器，则可以将定义的过滤器从一个实例复制或剪切到另一个过滤器中。

### 要复制过滤器

- 1 在 Alarm Pareto 控件的第一个实例中定义过滤器。
- 2 使用鼠标右键单击这些过滤器，然后单击**复制**。要移动这些过滤器，请单击**剪切**。
- 3 关闭 Alarm Pareto 控件的第一个实例。
- 4 打开 Alarm Pareto 控件的下一个实例，然后单击**查询过滤器**选项卡。
- 5 将箭头放在右侧窗格中。使用鼠标右键单击所选的过滤器，然后单击**粘贴**。

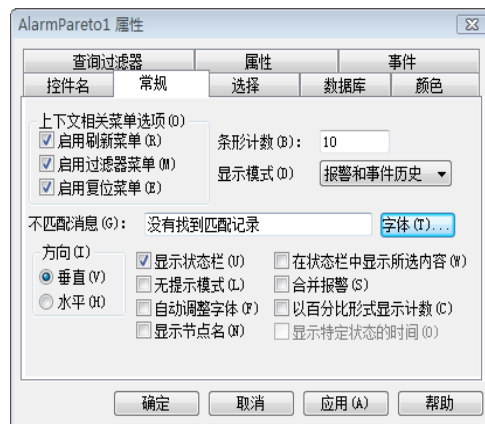


## 配置分析结果的显示

您可以配置查询结果中报警的显示。

### 要配置显示

- 1 使用鼠标右键单击 Alarm Pareto 控件，然后单击**属性**。此时出现 **AlarmPareto 属性** 对话框。
- 2 单击**常规**选项卡。



- 3 选择**合并报警**复选框，以便将多状态报警中的报警合并为单时间标签的记录。
- 4 选择**以百分比形式显示计数**复选框，以便为图中显示的每个条形添加占总数的百分比。
- 5 单击**应用**。

## 在运行时使用 Alarm Pareto 控件

在运行时使用鼠标右键单击 Alarm Pareto 控件打开一个快捷菜单。下表列出该快捷菜单中出现的所有可能的选项。

菜单选项	描述
刷新	刷新显示对象。
过滤器	可用于编辑过滤器以更改巴累托控件接收的数据。只有在设置了过滤器收藏夹文件的情况下，此菜单项才可用。
复位	将图形复位成缺省查询。

### 理解状态栏上显示的信息

Alarm Pareto 控件的状态栏显示：

- 控件与报警数据库之间的数据库连接状态。
- 刷新图形中显示的数据的图形更新状态。

## 使用 Alarm Pareto ActiveX 属性

您可以使用脚本直接设置 Alarm Pareto 控件属性的值，或是将它指定给 InTouch 标记或 I/O 引用。如需有关设置属性的详细信息，请参阅 *InTouch® HMI 脚本与逻辑指南* 中的第 8 章“编写 ActiveX 控件脚本”。

下表列出 Alarm Pareto 属性。如需有关设置颜色值的详细信息，请参阅第 77 页的“配置 ActiveX 控件的颜色”。

属性名	用途
<b>AutoConnect</b>	返回或设置一个值，确定控件处于运行时模式之后是否立即连接到数据库。
<b>AutoFont</b>	可用空间不足而无法在所选的条形上正确显示文本时，“自动调整字体”功能会隐藏文本，并且仅当选择该条形时才显示文本。
<b>BackGndColor</b>	设置 Alarm Pareto 图的背景色
<b>BarColor</b>	设置 Alarm Pareto 控件的条形颜色。
<b>BarCount</b>	设置 Alarm Pareto 控件上出现的条形数。
<b>BarSelectColor</b>	设置 Alarm Pareto 控件中所选条形的颜色。
<b>Connected</b>	确定 Alarm Pareto 控件是否连接到数据库。
<b>ConsolidatedAlarms</b>	将多个报警状态合并到两个状态。例如，如果某个模拟标记有三个报警状态：Hi、HiHi 及 Normal，则将 Hi 与 HiHi 报警状态划分为一个状态。
<b>DatabaseName</b>	设置 Alarm Pareto 控件要连接的数据库的名称。
<b>DisplayMode</b>	设置显示模式。显示模式选项包括“报警和事件历史”、“报警历史”以及“事件历史”。
<b>Duration</b>	返回或设置由控件用于设置“开始时间”与“结束时间”的持续时间。

属性名	用途
<b>EnableRefresh</b>	启用或禁用可刷新 Alarm Pareto 控件的上下文菜单。
<b>EnableReset</b>	启用或禁用可重置 Alarm Pareto 控件的上下文菜单。
<b>EnableSilentMode</b>	启用或禁用“无提示模式”。如果禁用“无提示模式”，则 Alarm Pareto 控件显示错误消息框。如果启用“无提示模式”，则错误消息框不会出现。错误信息写入 logger 中。
<b>EndTime</b>	返回或设置结束日期与时间。
<b>FilterMenu</b>	启用或禁用某个上下文菜单项，供您编辑过滤器来更改 Pareto 控件接收的数据。仅当设置了“过滤器收藏夹文件”，此属性才会启用。
<b>FilterFavoritesFile</b>	按字符串格式指定过滤器收藏夹文件。
<b>Font</b>	设置控件中记录与标题的字体。
<b>FontColor</b>	设置 Alarm Pareto 控件中记录视图的字体颜色。
<b>HorizontalChart</b>	将图表显示为水平条形。如果禁用 HorizontalChart，则图表显示为垂直条形。
<b>MaxRecords</b>	返回或设置一个值，指定要在给定的时间检索的最大记录数。
<b>NoMatchMessage</b>	设置在 Alarm Pareto 控件中没有要处理的数据时显示的消息。
<b>QueryTimeZone</b>	将时区设置为“UTC 时间”或“原始时间”。
<b>ServerName</b>	返回当前服务器名。
<b>ShowCountPercentage</b>	如果选择此项，则每个条形的计数显示为占总数的百分比。如果未选择，则显示每个条形的实际计数。

属性名	用途
<b>ShowNodeName</b>	将 Alarm Pareto 控件设置为除显示其它的信息之外，还在条形上显示节点名。
<b>ShowSelectedInStatusBar</b>	启用或禁用在状态栏上显示所选条形相关信息的功能。
<b>ShowStatusBar</b>	返回或设置一个值，确定是否显示状态栏。
<b>ShowTimeinState</b>	返回或设置一个值，确定 Alarm Pareto 控件是否根据标记处于报警状态的时间来显示条形。如果禁用，则控件基于报警的发生次数来显示条形。
<b>SpecificTime</b>	返回或设置一个值，确定控件是使用“开始时间”与“结束时间”属性，还是根据“持续时间”属性的值来计算开始时间与结束时间。
<b>StartTime</b>	返回或设置开始日期与时间。
<b>User</b>	返回或设置由控件用于连接到 SQL Server 的“用户”。

## 使用 Alarm Pareto ActiveX 方法

使用 Alarm Pareto 的 ActiveX 方法可以：

- 控制数据库连接。
- 从数据库检索记录。
- 检索特定巴累托条的有关信息。

如需有关调用方法的详细信息，请参阅 *InTouch® HMI 脚本与逻辑指南* 中的第 8 章“编写 ActiveX 控件脚本”。

### 控制数据库连接

使用 Connect() 方法连接到报警数据库。

#### Connect() 方法

连接到从 Alarm Pareto 控件属性的**数据库**选项卡中配置的数据库。

##### 语法

*Object*.Connect()

##### 示例

控件名为 AlarmPareto1。

```
#AlarmPareto1.Connect();
```

### 从数据库检索记录

使用以下函数从数据库中检索记录：

- Refresh() 方法
- SelectQuery() 方法

#### Refresh() 方法

使用数据库中的数据来刷新控件；如果连接成功，则显示范围从 1 到 MaxRecords 属性定义的数字之间的一组记录。

##### 语法

*Object*.Refresh()

##### 示例

```
#AlarmPareto1.Refresh();
```

### SelectQuery() 方法

选择配置为查询收藏夹文件的过滤器。

#### 语法

*Object*.SelectQuery(Filter)

#### 参数

*Filter*

查询过滤器的名称。

#### 示例

控件名为 AlarmPareto1。

```
#AlarmPareto1.SelectQuery("MyFilter");
```

## 检索特定巴累托条的有关信息

使用以下函数检索特定巴累托条的有关信息：

- GetItemAlarmName() 方法
- GetItemAlarmType() 方法
- GetItemCount() 方法
- GetItemTotalTime() 方法
- GetItemEventType() 方法
- GetItemProviderName() 方法

### GetItemAlarmName() 方法

获取特定条形的报警的名称。

#### 语法

*Object*.GetItemAlarmName(*BarIndex*)

#### 参数

*BarIndex*

条形的索引。

#### 示例

控件名为 AlarmPareto1。

```
#AlarmPareto1.GetItemAlarmName(1);
```

### GetItemAlarmType() 方法

获取特定条形的报警的类型。

#### 语法

*Object*.GetItemAlarmType(*BarIndex*)

#### 参数

*BarIndex*

条形的索引。

#### 示例

控件名为 AlarmPareto1。

```
#AlarmPareto1.GetItemAlarmType(1);
```

### GetItemCount() 方法

获取条形中的报警数量。

#### 语法

*Object*.GetItemCount(*BarIndex*)

#### 参数

*BarIndex*

条形的索引。

#### 示例

控件名为 AlarmPareto1。

```
#AlarmPareto1.GetItemCount(1);
```

### GetItemTotalTime() 方法

获取标记处于报警状态的总计时间（以秒为单位）。

#### 语法

*Object*.GetItemTotalTime(*BarIndex*)

#### 参数

*BarIndex*

条形的索引。

#### 示例

控件名为 AlarmPareto1。

```
#AlarmPareto1.GetItemTotalTime(1);
```



### GetItemEventType() 方法

获取特定条形的事件的类型。

#### 语法

*Object*.GetItemEventType(*BarIndex*)

#### 参数

*BarIndex*

条形的索引。

#### 示例

控件名为 AlarmPareto1。

```
#AlarmPareto1.GetItemEventType(1);
```

### GetItemProviderName() 方法

获取为特定的条形生成的报警的供应器名。

#### 语法

*Object*.GetItemProviderName(*BarIndex*)

#### 参数

*BarIndex*

条形的索引。

#### 示例

控件名为 AlarmPareto1。

```
#AlarmPareto1.GetItemProviderName(1);
```

## 显示其它信息

使用 AboutBox() 方法显示关于对话框。

### AboutBox() 方法

显示关于对话框。

#### 语法

*Object*.AboutBox()

#### 示例

控件名为 AlarmPareto1。

```
#AlarmPareto1.AboutBox();
```

## 使用方法与属性时的错误处理

Alarm Pareto 控件基于**无提示模式**选项处理运行时错误消息。如需有关详细信息，请参阅第 327 页的“配置 Alarm Pareto 控件的外观与颜色”。

如果选择**无提示模式**，则 Alarm Pareto 控件不显示运行时错误消息。如果未选择它，则报警显示对象显示错误消息。所有的 Alarm Pareto 错误消息都发送到 Logger 中。

## 使用 Alarm Pareto ActiveX 事件触发脚本

您可以将 QuickScript 指定给 Alarm Pareto 控件事件（如鼠标单击或双击）。该事件发生时，此 QuickScript 便会运行。

Alarm Pareto 控件支持以下事件：

- Click
- DoubleClick
- ShutDown
- StartUp

Click 事件有一个参数 ClicknBarIndex，它可以确定在运行时所单击的条形的索引。

DoubleClick 事件有一个参数 DoubleClicknBarIndex，它可以确定在运行时所双击的条形的索引。

Click 与 DoubleClick 事件都是零基的。发布 Click 与 / 或 DoubleClick 事件时，显示对象中的条形计数从 0 开始。

---

**备注** Alarm Pareto 控件从 StartUp 事件调用方法时会忽略用户界面方法，原因在于控件尚不可见。这些方法包括 AboutBox() 与 Refresh()。

---

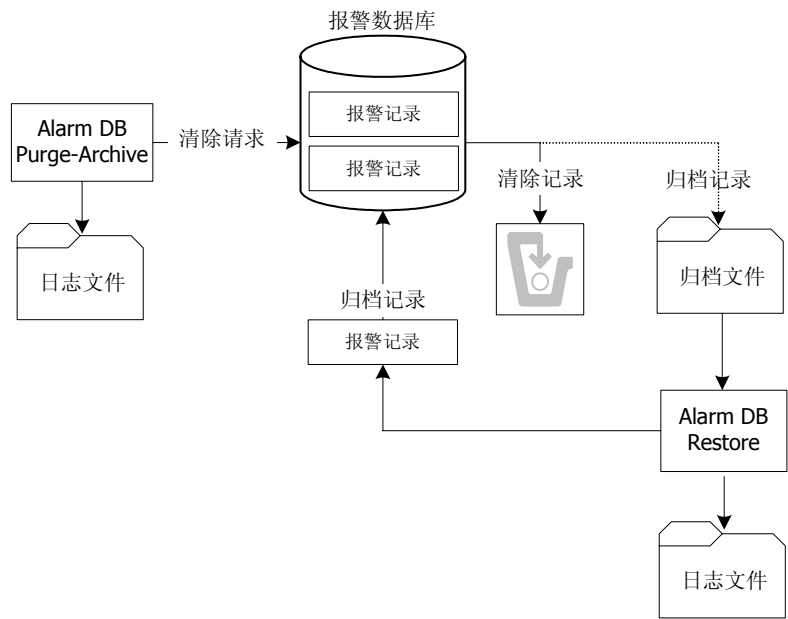
如需有关编写 ActiveX 事件脚本的详细信息，请参阅 *InTouch® HMI 脚本与逻辑指南* 中的第 8 章“编写 ActiveX 控件脚本”。

# 第 12 章

## 维护报警数据库

您可以使用两个 InTouch 实用程序来管理报警数据库。Alarm DB Purge-Archive 实用程序可用于从数据库中永久删除记录或将记录归档到文件中。如果数据库已毁坏，则可以使用 Alarm DB Restore 实用程序来恢复归档的记录。

下图显示这两个实用程序如何清除 / 归档记录，然后再将它们恢复到数据库。



您必须作为管理员登录到计算机才能使用 Alarm DB Purge-Archive 实用程序。

## 配置清除或归档设置

使用 Alarm DB Purge-Archive 实用程序可以：

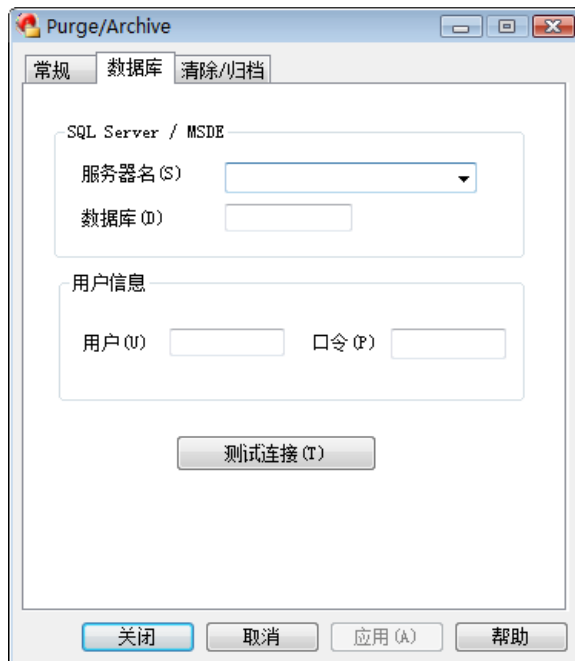
- 选择要从报警数据库中清除的记录的类型。
- 按每天、每星期或每月的日程自动清除记录。
- 将清除的数据库记录归档到文件（可选）。
- 将归档或清除操作的状态保存到日志文件中，以便就出现的问题进行疑难排解。
- 显示清除或归档操作的状态。

## 配置数据库连接

在可以使用 Alarm DB Purge-Archive 实用程序之前，必须先连接到报警数据库。

**要配置数据库连接**

- 1 打开 Alarm DB Purge-Archive 实用程序。执行以下操作：
  - a 在工具视图中，展开**应用程序**。
  - b 双击 **Alarm DB Purge-Archive**。
- 2 单击**数据库**选项卡。



- 3 配置数据库连接。执行以下操作：
  - a 在**服务器名**列表中，单击服务器的节点名。
  - b 在**数据库**框中，输入报警数据库的名称。
  - c 在**用户信息**区域中，输入报警数据库用户帐户的用户名与口令。
- 4 单击**测试连接**以测试与数据库的连接。此时出现一条消息，指出与报警数据库的连接是否成功。单击**确定**。
- 5 单击**应用**。

## 配置要从服务器中清除的数据量

您可以：

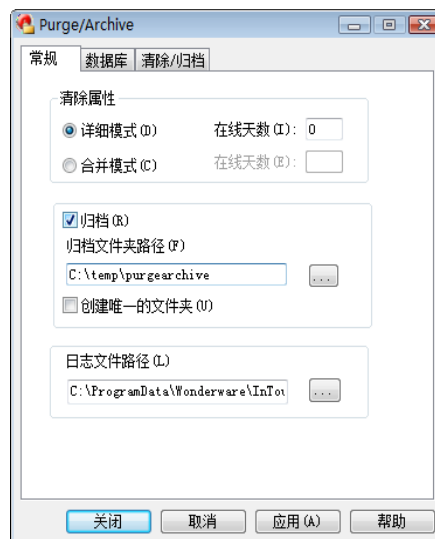
- 选择要从报警数据库中清除的报警记录的类型。
- 将从报警数据库中清除的记录归档到文件中（可选）。
- 选择要存储清除日志文件的文件夹位置。

您可以选择需要清除的表的类型，即 **AlarmDetail** 或 **AlarmConsolidated** 表。

在指定日期之前的所有数据都会清除掉。有效输入为 0 到 9999。如果选择 0，则从报警数据库中清除除当天的记录之外的所有记录。

### 选择要清除的记录

- 1 打开 Alarm DB Purge-Archive 实用程序。执行以下操作：
  - a 在工具视图中，展开**应用程序**。
  - b 双击 **Alarm DB Purge-Archive**。
- 2 单击**常规**选项卡。



- 3 在**清除属性**区域中，配置要清除的记录的类型。执行以下操作之一：
  - 单击**详细模式**，以清除数据库中按“详细”模式保存的报警记录。
  - 单击**合并模式**，以清除数据库中按“合并”模式保存的报警记录。
- 4 在**在线天数**框中，输入要在报警数据库中保存记录的天数。
- 5 单击**应用**。

## 配置清除的数据的归档

您可以归档从报警数据库中清除的记录，然后使用 Alarm DB Restore 实用程序来恢复它们。

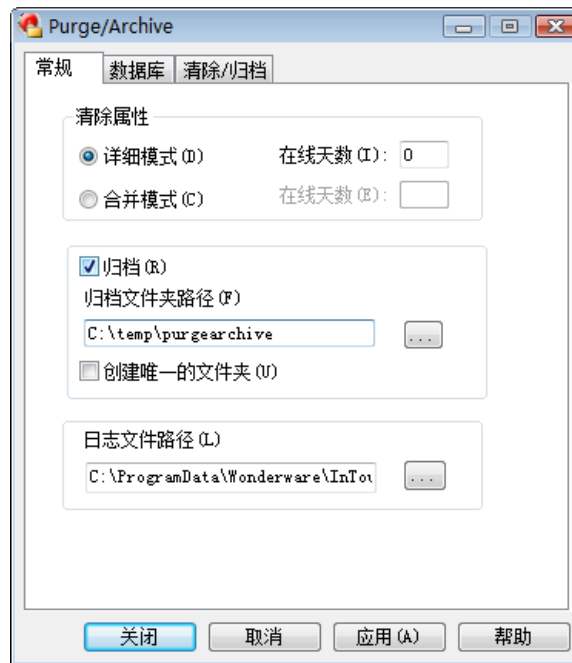
清除报警数据库时，Alarm DB Purge-Archive 实用程序自动创建一组九个归档文件，这些文件对应于已清除的报警数据库表。每个文件都包含单张表中清除的记录。

Alarm DB Purge-Archive 实用程序根据表名、以及清除操作发生的日期与时间给归档文件指定名称。例如，对于 2007 年 6 月 22 日下午 5:30 清除的 AlarmMaster 表，其归档文件的名称格式如下：

AlarmMaster\_06222007\_1730.txt

### 要配置归档

- 1 打开 Alarm DB Purge-Archive 实用程序。执行以下操作：
  - a 在工具视图中，展开应用程序。
  - b 双击 **Alarm DB Purge-Archive**。
- 2 单击**常规**选项卡。



- 3 选择**归档**复选框。
- 4 在**归档文件夹路径**框中，输入要保存归档文件的文件夹位置，或单击省略号按钮来浏览到该位置。
- 5 如果希望将归档文件放置在归档文件所在文件夹下单独的子文件夹中，请选择**创建唯一的文件夹**复选框。
- 6 单击**应用**。

## 配置日志文件设置

Alarm DB Purge-Archive 实用程序在清除操作期间生成状态消息。您可以从该实用程序的状态窗口中在线查看这些消息。

Alarm DB Purge-Archive 实用程序还将清除消息写入清除日志文件 WWAlmPurge.log 中。

下例显示清除操作成功之后日志文件中存储的消息。

清除开始时间

正在开始事务 ....

正在归档 ProviderSession 表 ...

正在归档 Query 表 ...

正在归档 Cause 表 ...

正在归档 AlarmMaster 表 ...

正在归档 OperatorDetails 表 ...

正在归档 AlarmDetail 表 ...

正在归档 Comment 表 ...

正在归档 Events 表 ...

正在归档 TagStatus 表 ...

正在清除数据库中的记录 ...

正在提交 ...

清除结束时间

144 条记录（从 AlarmMaster 中）已连同其它表格中的相关记录一起清除。

缺省条件下，清除日志文件存储在此文件夹中：C:\Documents and Settings\All Users\Application

Data\Wonderware\InTouch。对于运行 Microsoft Windows Vista 操作系统的计算机，缺省应用程序文件夹是：

C:\Users\UserName\Documents\我的 InTouch 应用程序。

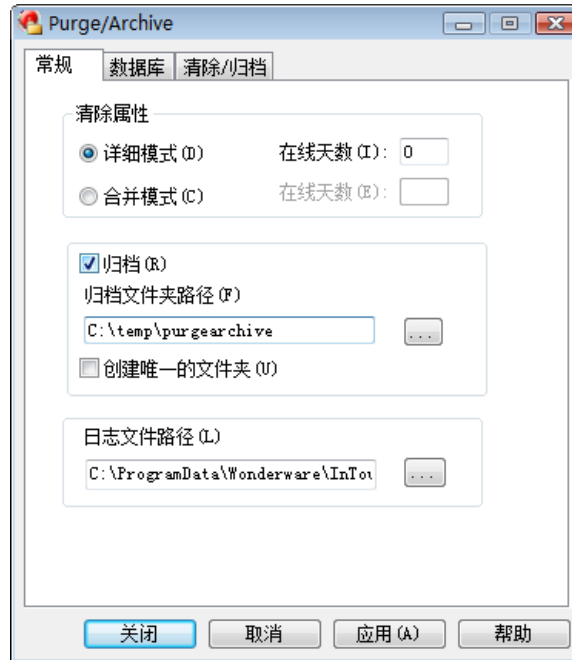
您可以更改清除日志文件的存储位置。

每次发生清除时，Alarm DB Purge-Archive 实用程序都会将新消息追加到日志文件中。



### 要设置归档记录

- 1 打开 Alarm DB Purge-Archive 实用程序。执行以下操作：
  - a 在工具视图中，展开应用程序。
  - b 双击 **Alarm DB Purge-Archive**。
- 2 单击**常规**选项卡。



- 3 在**日志文件路径**框中，输入要放置清除日志文件的文件夹位置，或单击省略号按钮来浏览该位置。
- 4 单击**应用**。

## 手工清除与归档数据库

您可以手工清除与归档报警数据库。此操作会覆盖激活时间并立即开始清除与归档。

清除操作检查是否存在归档文件，并将内容追加到相同的文件中。如果归档文件不存在，则按照命名惯例创建一个文件，然后将它用于归档。

清除操作不删除特定的表中的项目，如 **ProviderSession**、**Query** 及 **Cause** 表，这些表通过外部关键字约束条件链接到 **AlarmMaster** 等主表。这些表中的相关记录会写入文件以保持数据的一致性，同时也会保留在数据库中。

**注意** 仅当 **Alarm DB Logger** 服务停止时，才可以手工清除所有记录（**立即全清**选项）。如果清除操作在 **Alarm DB Logger** 服务运行期间成功提交，则 **Alarm DB Logger** 服务会停止记录并开始缓存记录。

### 要从报警数据库中手工清除与归档记录

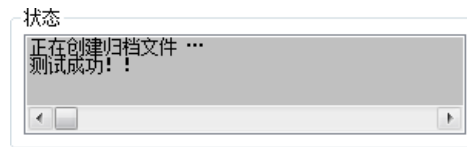
- 1 打开 **Alarm DB Purge-Archive** 实用程序。执行以下操作：
  - a 在工具视图中，展开**应用程序**。
  - b 双击 **Alarm DB Purge-Archive**。
- 2 单击**清除 / 归档**选项卡。



- 3 单击**立即测试**以执行清除测试，以便验证与数据库的连接及归档位置。

清除测试会在指定的归档文件夹中创建一些空的归档文件。

**状态**区域显示一条消息，指出测试成功。



仅当选择归档清除的记录时，**立即测试**按钮才可用。**归档**选项位于**常规**选项卡。

- 4 清除数据库中的记录。执行以下操作之一：
  - 单击**立即清除**以清除所选的记录。
  - 单击**立即全清**以清除所有的记录。
- 5 要停止清除，请单击**取消清除**。如果取消清除，则报警数据库会回滚到原始状态。

## 设置自动清除计划

Alarm DB Purge-Archive 可以按计划的间隔从报警数据库中自动清除或归档记录。您可以执行清除测试，以便验证同数据库的连接及目标位置，并可以启动与停止清除。

### 要设置自动清除计划

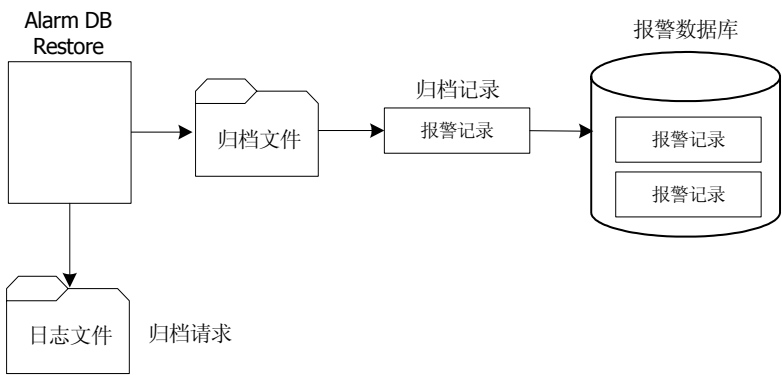
- 1 打开 Alarm DB Purge-Archive 实用程序。执行以下操作：
  - a 在工具视图中，展开应用程序。
  - b 双击 **Alarm DB Purge-Archive**。
- 2 单击**清除 / 归档**选项卡。



- 3 在**时间间隔**区域中，选择一个清除间隔（每天、每星期或每月）。  
如果单击**每星期**或**每月**，则在**激活时间**区域中出现**日期**框，供您指定星期几或月中的日期。  
如果单击**每天**，则在**时间**框中，可以配置希望开始清除 / 归档操作的时间。
- 4 在**运行方式**区域中，单击**应用程序**使 Purge-Archive 实用程序作为应用程序运行，或单击**服务**使其作为服务运行。
- 5 单击**应用**以保存清除与归档设置。
- 6 单击**启动**以实施 Alarm DB Purge-Archive 实用程序的自动清除计划。
- 7 单击**关闭**。

## 恢复报警数据库

Alarm DB Restore 实用程序可以将归档文件中归档的报警记录恢复到报警数据库中。下图概要介绍将报警记录恢复到数据库的步骤。



要恢复数据库，必须：

- 配置与报警数据库的连接。
- 选择要恢复到报警数据库中的记录。
- 将归档的记录恢复到报警数据库。

最小化时， Alarm DB Restore 实用程序显示为系统工具栏中的一个图标。使用鼠标右键单击该图标时，会出现显示以下命令的菜单：

命令	描述
恢复	开始恢复过程。
取消恢复	取消恢复过程。
清除状态	清除状态窗口。
隐藏窗口	最小化 Alarm DB Restore 实用程序，使之成为系统工具栏中的一个图标。
显示窗口	打开并最大化 Alarm DB Restore 实用程序。
退出	关闭 Alarm DB Restore 实用程序。

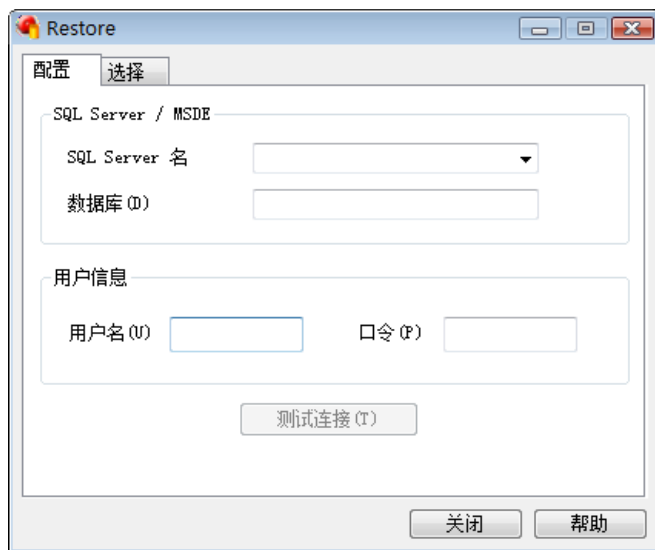
如果在 Alarm DB Restore 实用程序中单击鼠标右键，则会出现相同的菜单。

## 配置数据库连接

您必须选择要用于恢复归档的数据的数据库。如果服务器上不存在指定的数据库，则程序会提示您使用缺省的服务器参数来创建一个新的数据库。

### 要配置用于恢复的数据库

- 1 打开 Alarm DB Restore 实用程序。执行以下操作：
  - a 在工具视图中，展开应用程序。
  - b 双击 **Alarm DB Restore**。
- 2 单击**配置**选项卡。



- 3 配置与报警数据库的连接。执行以下操作：
  - a 在 **SQL Server** 名列表中，单击存放报警数据库的服务器的节点名。
  - a 在**数据库名**框中，输入报警数据库的名称。
  - b 在**用户信息**区域中，将报警数据库用户名与口令输入相应的方框。
  - c 单击**测试连接**以测试与数据库的连接。此时出现一条消息，指出与报警数据库的连接是否成功。单击**确定**。
- 4 单击**关闭**。

## 配置要恢复的文件

您可以选择要恢复的记录的时间段，以及是否希望重建数据库表。

如果取消恢复，数据库会回滚到原始状态。

**注意** 如果试图恢复数据库中已经存在的归档报警，则不会恢复这些归档的记录。这可以避免在数据库中出现重复的报警 / 事件项。与记录关联的“报警 GUID”或“事件 GUID”确定数据库中是否已经存在报警或事件。

### 选择要恢复的数据库记录

- 1 打开 Alarm DB Restore 实用程序。执行以下操作：
  - a 在工具视图中，展开应用程序。
  - b 双击 **Alarm DB Restore**。
- 2 单击**选择**选项卡。



- 3 在**归档文件目录**框中，输入归档文件位置的完整路径（最多 255 个字母数字字符），或单击按钮以查找并选择存储归档文件的文件夹。
- 4 在**恢复此后的文件（日期 / 时间）**区域中，选择开始将记录恢复到数据库的日期与时间。  
缺省条件下，开始日期与时间设置为当前日期与时间。
- 5 在**日志文件目录**框中，输入要创建并存储日志文件的文件夹的完整路径（最多 255 个字母数字字符），或是单击按钮以查找并选择某个文件夹。
- 6 如果选择**重建表**复选框，则会重建指定的报警数据库的表。根据为归档文件中包含的报警记录选择的记录类型，选择：
  - **详细** - 在详细模式下重建报警数据库表。

- 合并 - 在合并模式下重建报警数据库表。

**重要** 重建这些表时，会覆盖报警数据库中当前存储的所有记录。

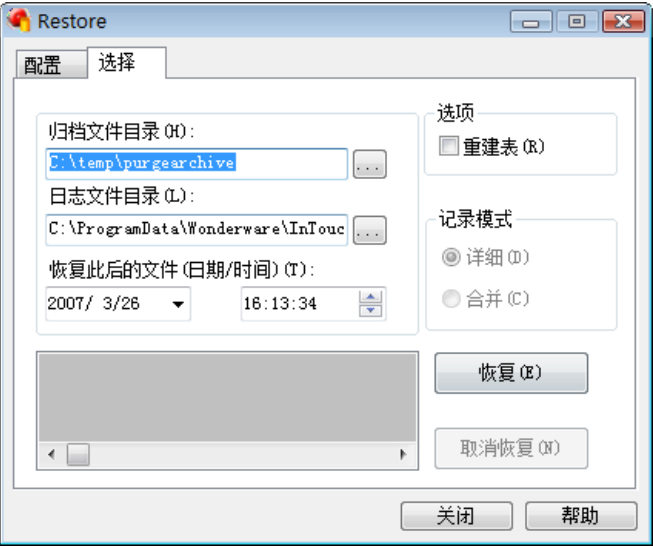
7 单击**恢复**。

## 开始数据库恢复操作

在建立数据库连接并指定归档文件所在的文件夹及时间过滤器之后，便可以恢复归档的数据库记录。

要从归档文件中恢复数据库记录

- 1 打开 Alarm DB Restore 实用程序。执行以下操作：
  - a 在工具视图中，展开**应用程序**。
  - b 双击 **Alarm DB Restore**。
- 2 单击**选择**选项卡。



- 3 单击**恢复**。此时出现一条消息，显示恢复是否成功，以及已恢复到数据库的记录数。



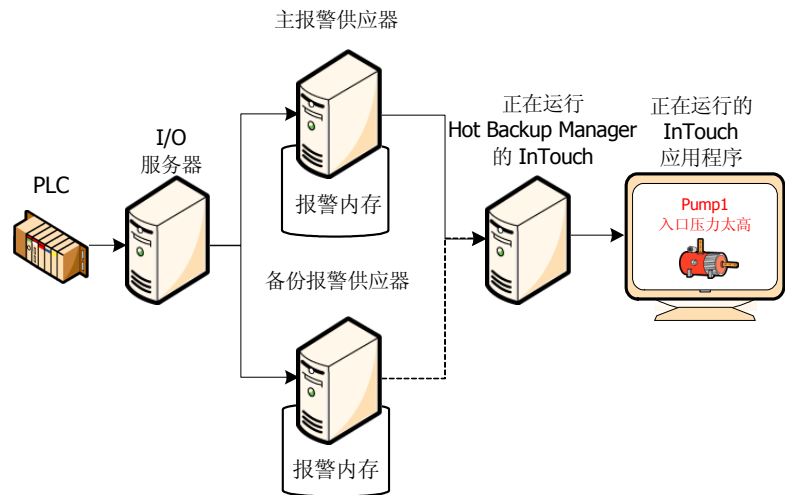


## 第 13 章

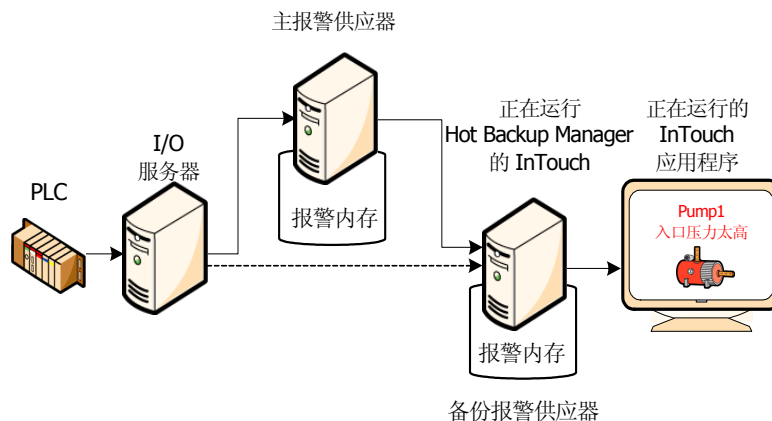
# 通过冗余报警配置增强工厂安全性

“InTouch 分布式报警”系统发出通知，并从网络中的远程节点上运行的应用程序接收报警确认。报警供应器应用程序将报警数据存储在它们的内存中。报警接收器应用程序作为客户端在其它节点上的运行，可以远程查询、显示及确认报警供应器中的报警。

您可以使用 Alarm Hot Backup Manager 来创建重复的报警供应器。下图显示 Hot Backup Manager 如何将当前的辅助报警储备库用作备份供应器。



您也可以在相同的节点上运行 Hot Backup Manager 与备份供应器，如下图所示：

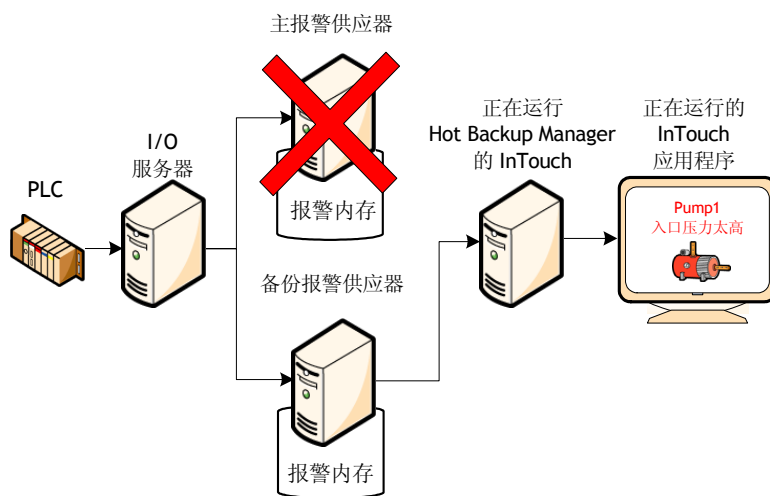


## 理解热备份

热备份提供一个名称（热备份对名），指向两个报警供应器，即主报警供应器与备份报警供应器（热备份对）。InTouch HMI 的报警接收器（如 Alarm Viewer 控件）可以引用这个热备份对名，并从主报警供应器或备份报警供应器中检索报警。

如果这两个供应器节点都正常工作，则报警接收器从主供应器接收报警数据。但如果主供应器发生故障，则报警接收器从备份供应器接收报警数据。

下图显示报警接收器如何在主报警供应器发生故障之后仍能接收报警。报警接收器仍然引用热备份对，但是由备份供应器提供报警数据。



Hot Backup Manager 对主供应器与备份供应器之间的报警确认进行同步。如果主供应器上的某个报警得到确认，则备份供应器上的相同报警也会同时得到确认。

Hot Backup Manager:

- 提供用于创建备份对的配置实用程序。
- 提供某个配置实用程序，用于在热备份对的主供应器与备份供应器之间映射报警记录。
- 同步备份对之间的报警确认。
- 在“分布式报警系统”启动与停止时，在属于一个“热备份”系统的所有节点之间建立通讯。

## 配置热备份对

Alarm Hot Backup Manager 使用运行供应器应用程序的两个主机节点来创建一个备份对。您可以从 WindowMaker 中启动 Hot Backup Manager。要配置热备份对：

- 创建热备份对。
- 设置报警记录的关键码域。
- 映射报警记录关键码域。
- 将报警记录映射导入 Hot Backup Manager。

## 创建热备份对

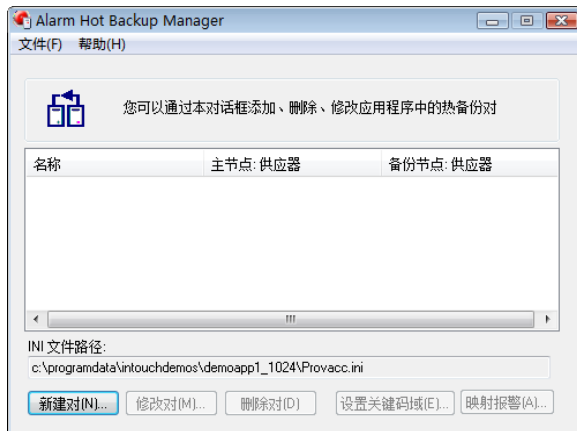
要创建热备份对，您可以：

- 给热备份对指定一个名称。
- 确定主报警供应器。
- 确定备份报警供应器。

您也可以指定一个包含配置信息的 Provacc.ini 文件。

### 要配置热备份对


- 1 打开 Alarm Hot Backup Manager。执行以下操作：
  - a 在工具视图中，展开应用程序。
  - b 双击 **Alarm Hot Backup Manager**。



- 2 在文件菜单上，单击**打开**。选择 Provacc.ini 文件，然后单击**确定**。

缺省条件下，Alarm Hot Backup Manager 在最近打开的 InTouch 应用程序文件夹中查找 Provacc.ini 文件。您应该使用 InTouch 应用程序文件夹中的 Provacc.ini 文件。否则，您可以在另一个指定的文件夹位置创建一份 Provacc.ini 文件，然后选择它供 Hot Backup Manager 使用。

- 3 单击**新建对**。此时出现**添加新对**对话框。

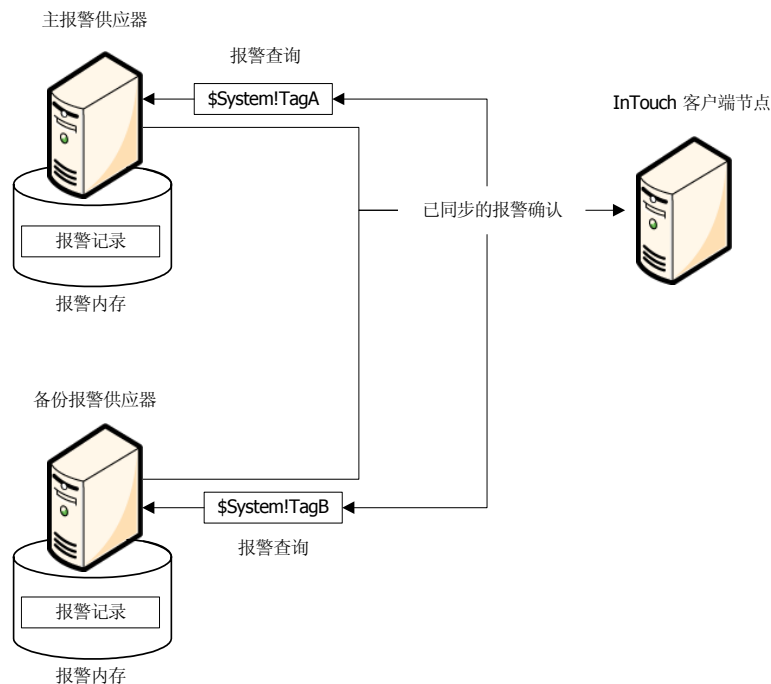


**备注** 您无法更改 InTouch 静态供应器名。

- 4 在**热备份对名**框中，为新备份对输入唯一的名称。  
对名的最大长度为 32 个字母数字字符。您可以在对名中使用下划线字符。
- 5 在**主节点**区域中，配置主节点。执行以下操作：
- a 在**名称**框中，输入运行主供应器应用程序的计算机的节点名。节点名对于 **Hot Backup Manager** 而言必须是唯一的。如果输入一个不存在的节点名，或节点已经用在其它热备份对中，则出现一条错误消息。
  - b 在**组**框中，输入从主供应器查询报警的报警组的名称。
- 6 在**备份节点**区域中，配置备份节点。执行以下操作：
- a 在**名称**框中，输入运行备份供应器应用程序的计算机的节点名。这可以是运行 **Hot Backup Manager** 的相同节点。
  - b 在**组**框中，输入从备份供应器中查询报警的报警组的名称。
- 7 单击**确定**。
- 8 在**文件**菜单上，单击**保存**。
- 9 重新启动 WindowMaker。

## 设置热备份对的报警关键码域

要同步主供应器与备份供应器之间的报警确认，必须确定标记报警记录字段的组合。此字段组合为每个供应器当前报警储备库中存储的成对报警记录生成一个唯一的映射关键码。下图显示标准查询中基于报警记录字段的同步报警确认请求。



映射关键码可以是设计时与运行时报警记录的组合。设计时报警记录基于从“标记名字典”中定义标记时的报警属性。

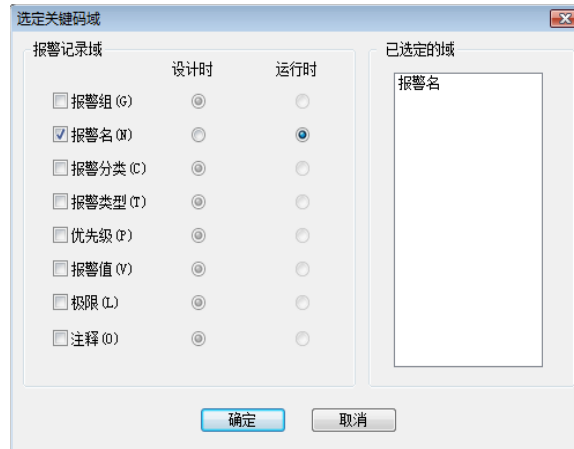
例如，报警名字段在设计时是已知的，因为它采用主节点应用程序与备份节点应用程序中定义的标记名。QuickScript 或操作员动作定义或修改在应用程序运行期间作为记录来存储的报警属性。

您可以使用设计时或运行时报警记录字段的任意组合来创建映射关键码。映射关键码必须仅从每个供应器的当前报警储备库中选择一条记录。关键码域必须创建唯一的查询。

您可以使用 Hot Backup Manager 从报警记录字段中创建映射关键码列表。

### 要为热备份对创建报警字段映射列表

- 1 打开 Alarm Hot Backup Manager。执行以下操作：
  - a 在工具视图中，展开应用程序。
  - b 双击 **Alarm Hot Backup Manager**。
- 2 从列表选择一个热备份对。
- 3 单击**设置关键码域**。此时出现**选定关键码域**对话框。



- 4 在**报警记录域**区域中，选择要包含在映射关键码列表中的报警记录字段。  
所选的报警记录字段出现在**已选定的域**列表框中。
- 5 为所选的报警记录字段选择**设计时**或**运行时**。
- 6 单击**确定**。
- 7 重新启动 WindowMaker。

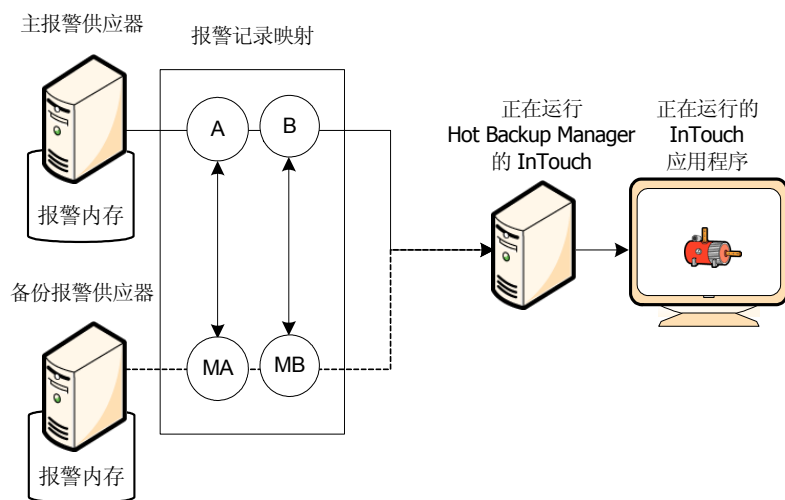
## 创建报警记录映射文件

只要主报警供应器与备份报警供应器在运行不同的应用程序，便必须映射热备份对的报警记录。报警记录映射在主供应器与备份供应器的不同报警记录之间建立起对应关系。例如，您可以基于指定的 InTouch 标记名来映射报警记录。尽管它们的名称可能不同，但是在两个供应器报警储备库之间，这些报警记录在逻辑上是一致的。

**备注** 如果主供应器与备份供应器正在运行相同的应用程序，则不需要创建报警记录映射文件。如果没有提供映射文件，则“分布式报警系统”假设主供应器与备份供应器正在运行具有相同报警记录的相同应用程序。

映射可用于在运行不同应用程序的供应器之间进行报警确认的同步。“分布式报警系统”确认供应器上的报警时，它同样知道要确认另一个供应器上的哪个报警。

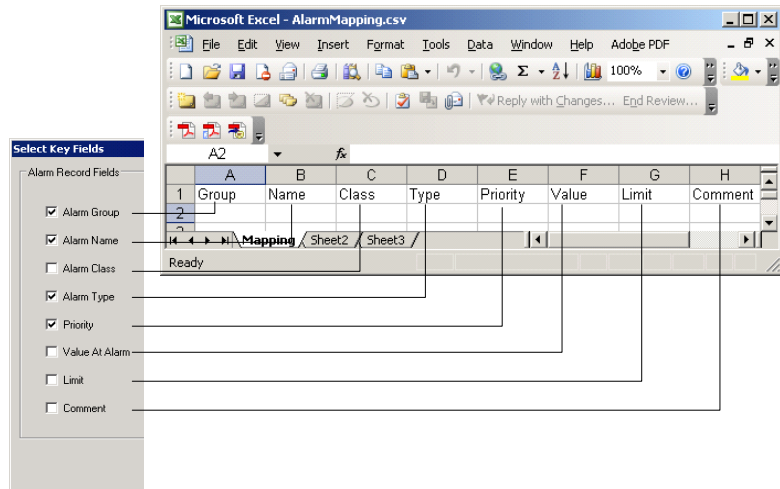
下图显示某个热备份对的两个供应器之间的报警记录映射。在本例中，主供应器的 A、B 报警记录映射为备份供应器 MA 与 MB 中相应的报警记录。



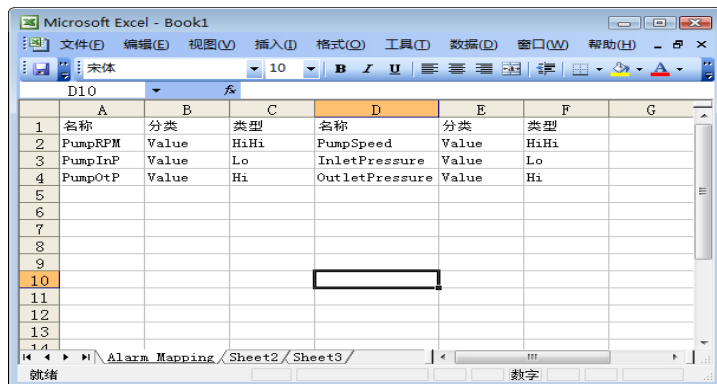


Hot Backup Manager 从使用 Microsoft Excel 或文本编辑器（如“记事本”）创建的逗号分隔值 (CSV) 文件中导入报警记录映射。映射文件包含一个排序过的报警记录字段列表，这些字段关联到主供应器与备份供应器的相应报警记录。

您必须将标记报警记录字段指定为映射文件的标题。文件中标题的顺序必须与**选定关键码域**对话框中显示的报警记录字段匹配。下图显示某个 Excel 文件的列标题，其顺序与**选定关键码域**对话框中报警记录字段的顺序匹配。



您可以创建一个映射文件，其中仅包含用于生成映射关键码的所选报警记录字段的标题。下图显示一个仅包含“名称”、“分类”以及“类型”等标题的 Excel 文件。添加标题时，它们的顺序必须总是与**选定关键码域**对话框中报警记录字段的顺序匹配。



在左侧的列集合中，指定主供应器的报警记录字段。同样地，在右侧的列集合中指定备份供应器的相同记录。

映射文件列标题	指定给报警记录字段的值
组	指定了该标记的报警组的名称。报警组名不得包含空格。
名称	映射其报警记录的标记的名称。标记名不得包含空格。
分类	指定给该标记的报警的分类。可能的“分类”值有： <ul style="list-style-type: none"><li>• VALUE，代表值报警。</li><li>• DEV，代表偏差报警。</li><li>• ROC，代表变化率报警。</li><li>• DSC，代表离散报警。</li></ul>
类型	与报警类关联的报警条件的类型。 <ul style="list-style-type: none"><li>• 对于值报警，有 LOLO、LO、HI 以及 HIHI</li><li>• 对于偏差报警，有 MinDev 与 MajDev</li><li>• 对于变化率报警，有 ROC</li><li>• 对于离散报警，有 DSC</li></ul>
优先级	指定给报警条件的优先级。优先级必须是 1 到 999 之间的一个数字。
值	请参阅以下备注。
报警限	请参阅以下备注。
注释	请参阅以下备注。

- “值”、“报警限”以及“注释”列：
- 特定节点中特定记录的“分类”或“类型”值未知时，“值”与“报警限”列的值可以是 Null 之外的任何值。
  - 特定节点中特定记录的“分类”值已知为 Value、Dev 或 ROC 时，“值”与“报警限”列的值只能接受 1234567890.-+eE 等字符。
  - 特定节点中特定记录的“类型”值已知为 LOLO、LO、HI、HIHI、MinDev、MajDev 或 ROC 时，“值”与“报警限”列的值只能接受 1234567890.-+eE 等字符。

- 特定节点中特定记录的“分类”或“类型”值中任何一个已知为 DSC 时，“值”与“报警限”列的值可以是 Null 之外的任何值。
- “注释”列的值没有任何限制。
- 映射文件中的所有记录都应该是唯一的。Hot Backup Manager 在导入过程中会跳过任何重复的记录。您可以在导入过程完成之后查看详细信息。

您可以合并报警记录的字段值 - 如“组”、“名称”以及“优先级” - 以生成唯一确定报警记录的“组合映射关键码”。

“InTouch 报警供应器”将“名称”字段的值赋给生成报警的标记名。因此，给定热备份对时，可以使用报警组名与标记名的组合来生成映射关键码。

例如：

供应器节点	备份节点
\$System!TagA	\$System!TagB

如果某个供应器将名称字段与注释字段合到一起用作唯一的字段，则映射关键码可以是名称与注释的组合。

供应器节点	备份节点
tagA!CommentA	tagB!CommentB

对于第三个供应器的任何其它字段组合，此方法也适用。

## 导入报警记录映射文件

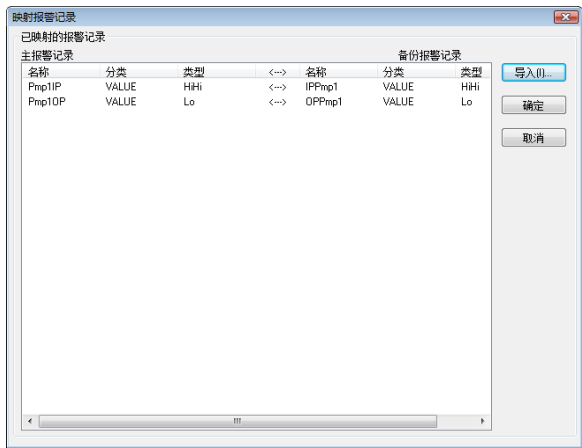
您可以将报警记录映射文件的内容导入 Hot Backup Manager 中。

导入映射文件时，不会在“报警分类”与“报警类型”之间进行交叉验证。

### 要从映射文件中导入报警记录

- 1 打开 Alarm Hot Backup Manager。执行以下操作：
  - a 在工具视图中，展开应用程序。
  - b 双击 **Alarm Hot Backup Manager**。
- 2 从列表中选择热备份对。

3 单击**映射报警**。此时出现**映射报警记录**对话框。



4 单击**导入**。此时出现**打开**对话框。选择映射文件，然后单击**打开**。

此时 Hot Backup Manager 开始从该文件中导入记录。

5 导入所有的映射记录之后，单击**确定**。

6 在**文件**菜单上，单击**保存**。

7 重新启动 WindowMaker。

### 报警映射文件导入问题疑难排解

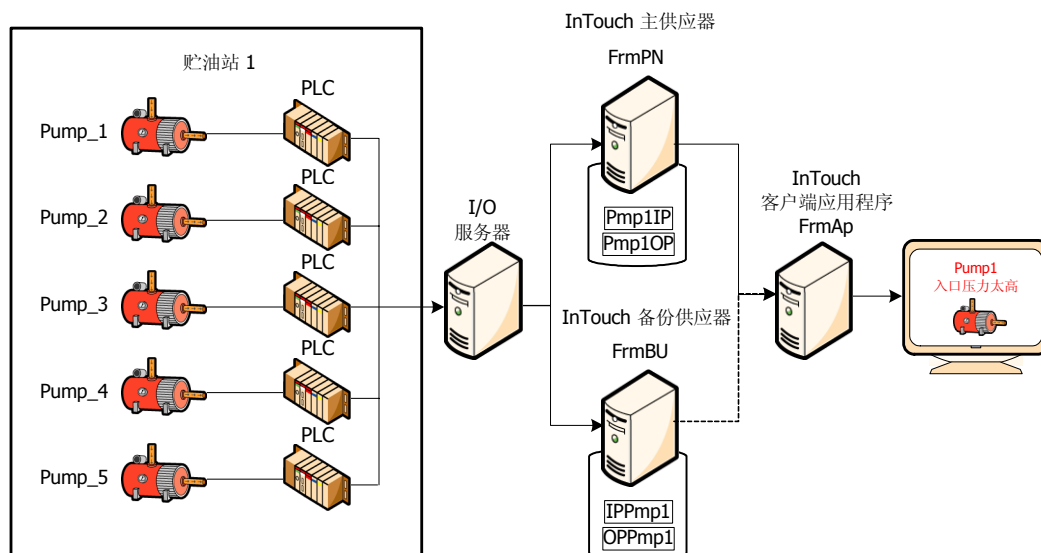
以下情况会导致无法导入文件：

- 对于导入文件中的所有记录，应该为要求填写的列填入一个值。任何记录都不得缺少或多出一些值。
- 导入文件的标题应该与**选定关键码域**对话框中的标题相同，并且顺序也应该相同。

如果导入的记录包含错误项，则提示您跳过这个特定的记录号或中止导入过程。

## 热备份对示例

本节介绍需要设置报警备份对的典型工作情形。下图显示 InTouch 贮油站应用程序的热备份对配置。在本例中，InTouch 应用程序监视泵浦压力，并将它用作一个报警条件。



所有三台计算机都在运行 InTouch HMI。热备份对包含作为主供应器的 FrmPN 以及作为备份供应器的 FrmBU。这两个节点用作 “InTouch 分布式报警系统” 中的报警供应器。

Hot Backup Manager 在 FrmAp 节点上运行。InTouch 客户端应用程序在 FrmAp 上运行，并且从热备份对的两个供应器中接收报警。

FrmPN 上运行的 InTouch 应用程序为泵浦入口压力与出口压力生成两个摘要报警。这两个报警属于 TnkFrm1 报警组。

FrmBU 上运行的 InTouch 应用程序为泵浦压力生成逻辑上等价的两个报警。

设置冗余热备份对时，您可以：

- 创建热备份对。
- 设置报警记录关键码域。
- 创建报警记录映射文件。
- 导入报警记录映射文件。

### 要创建热备份对

- 1 在 WindowMaker 中打开 InTouch 应用程序。在本例中，应用程序在 FrmAp 节点上运行。
- 2 打开 Alarm Hot Backup Manager。执行以下操作：
  - a 在工具视图中，展开应用程序。
  - b 双击 Alarm Hot Backup Manager。
- 3 单击**新建对**。此时出现**添加新对**对话框。
- 4 如下图所示，完成**添加新对**对话框中的选项。

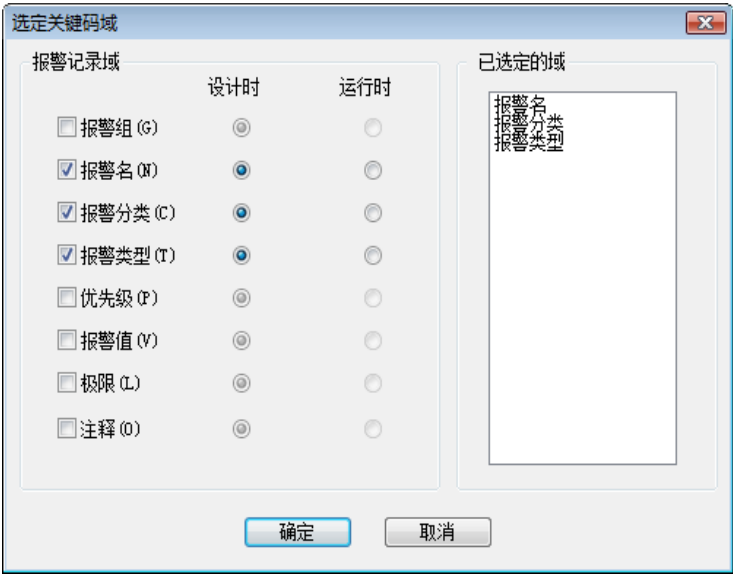


- 5 单击**确定**以返回 Alarm Hot Backup Manager 对话框。
- 6 在 WindowMaker 中保持打开 Alarm Hot Backup Manager 对话框。

此时已完成第一个步骤，即创建热备份对。接下来要完成以下操作程序，以便为每个供应器的报警储备库中存储的成对报警记录生成唯一的映射密钥码。

要映射报警记录关键码域

- 1 选择列表中的热备份对。
- 2 单击**设置关键码域**。此时出现**选定关键码域**对话框。  
如下图所示，完成**选定关键码域**对话框中的选项。主供应器与备份供应器之间的报警在逻辑上是一致的，但是指定了不同的名称并且属于不同的报警组。通过选择**报警组**、**报警名**、**报警分类**以及**报警类型**作为**设计时**选项，可以为每个供应器的报警储备库中存储的记录生成唯一的映射关键码。



- 3 单击**确定**。出现消息时，单击**是**。  
此时已完成第二个步骤，即创建报警记录映射关键码。
- 在此情形中，所有三个节点都在运行 InTouch 应用程序。两个供应器节点生成等价的报警，但使用不同的标记名。主供应器在泵浦的入口与出口压力过高时生成两个摘要报警。备份供应器为相同的泵浦压力报警条件生成两个逻辑上等价的报警。接下来要完成以下操作程序以便创建映射文件，将每个供应器的报警储备库中存储的等价记录关联起来。

### 要创建报警映射文件

- 1 使用 Excel 或文本编辑器（如“记事本”）创建一个 .csv 文件。
- 2 按**选定关键词域**对话框中报警记录字段选项的相同顺序输入文件标题的名称。  
在本例中，文件标题应该按报警组、报警名、报警分类以及报警类型的顺序排列。
- 3 在文件的每一行上映射两个供应器之间的报警。
- 4 下面的 Excel 文件示例显示应该如何为热备份对的两个供应器指定标题与报警条件。将映射文件保存到客户端节点上运行的 Hot Backup Manager 可以访问到的位置。

	A1	B	C	D	E	F
1	名称	分类	类型	名称	分类	类型
2	Pmp1IP	VALUE	HiHi	IPPmp1	VALUE	HiHi
3	Pmp1OP	VALUE	Lo	OPPmp1	VALUE	Lo
4						
5						
6						

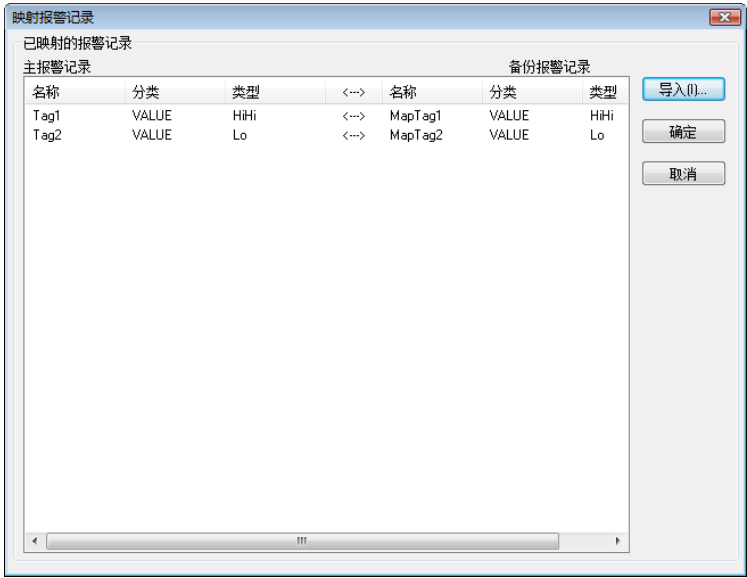
此时已完成第三个步骤，即创建报警记录映射文件。

在最后一个步骤，需要将报警映射文件的内容导入 Hot Backup Manager 中。在本例中，客户端应用程序知道要在两个报警供应器之间确认哪些泵浦压力报警记录。



要导入 .csv 报警记录映射文件

- 1 打开 Alarm Hot Backup Manager。  
此时应列出先前创建的报警备份对。
- 2 单击映射报警。出现映射报警记录对话框。
- 3 单击导入。此时出现打开对话框。
- 4 选择映射文件，然后单击打开。此时映射报警记录对话框列出文件中的报警映射记录。



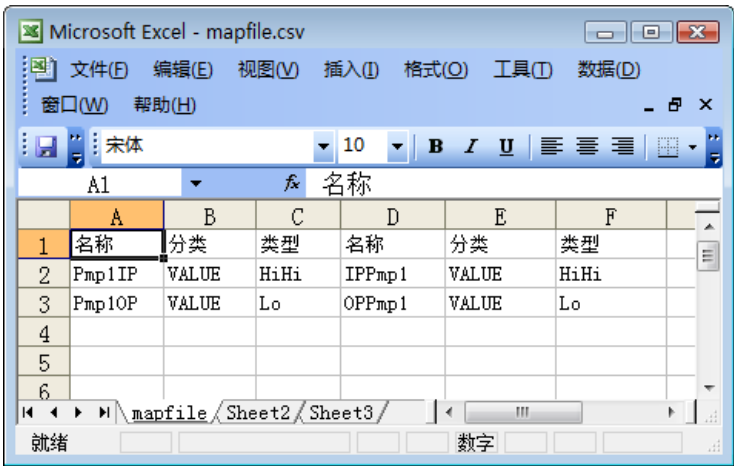
- 5 单击确定。此时 Hot Backup Manager 开始从该文件中导入记录。
- 6 导入所有的映射记录之后，单击确定。  
现在您便可以运行热备份应用程序。

确认同步示例

在本例中：

- “报警名”、“报警分类”以及“报警类型”选作设计时关键码域。
- “报警组”选作运行时字段。

以下报警记录映射文件是使用 Microsoft Excel 创建的。



Pmp1IP 报警映射到 IPPmp1 报警。两者的“分类”都是 VALUE，“类型”都是 HIHI。

Pmp1OP 报警映射到 OPPmp1 报警。两者的“分类”都是 VALUE，“类型”都是 Lo。

- 只要两个节点上的报警组名保持相同，在主报警节点上确认 Pmp1IP 的 HiHi 报警时，便也会确认辅助供应器节点上 IPPmp1 的 HiHi 报警。
- 只要两个节点上的报警组名保持相同，在主报警节点上确认 Pmp1OP 的 Low 报警时，便也会确认辅助供应器节点上 OPPmp1 的 Low 报警。

仅当设计时与运行时映射相互匹配时，才会发生确认同步。

您可以选择设计时与运行时报警记录字段的任何组合来进行映射。不过请确保映射不会导致多次引用。

例如，如果选择了两个报警记录字段（如“分类”与“优先级”），则很可能有多个报警与这个准则匹配。在这种情况下，无法保证“热备份”同步能够正常工作。在传播确认的过程中，也可能会遇到符合该准则的随机报警，而其它匹配的报警则可能得不到确认。

## 有关热备份对的备注

- 只有对于 InTouch 7.11 及更高版本的客户端，才可以使用热备份。
- 扩展的摘要报警或面向事件的报警不受支持。
- 如果“分布式报警显示”对象查询热备份对，然后再次单独查询主供应器，则“分布式报警显示”对象会显示重复的记录。
- 不要将供应器配置为多个热备份对的主供应器或辅助供应器。
- 如果主供应器上的记录得到确认，而稍后才启动辅助供应器（确认发生时是停工的），则确认的记录的时间标签应该与主供应器中的记录相同。
- 查询热备份对的报警接收器显示供应器的单独节点名。
- 您可以选择设计时与运行时报警记录字段的任何组合来进行映射。不过请确保映射不会导致多次引用。
- 映射“值”与“报警限”关键码域时，值会舍入到小数点后四位，然后再进行映射。
- 没有特定的设计时与运行时映射组合的报警记录会使用缺省的运行时映射。



## 第 14 章

# 创建报警审核跟踪

将 InTouch 报警供应器配置为使用操作系统或 ArchestrA 身份验证，并且发生报警时，假设操作员已登录，则报警显示对象会在“操作员全名”列中包含操作员的全名。

例如，如果某个用户注册在 PLANT\_FLOOR 域中，用户 ID 为 JohnS，全名为 John Smith，则“操作员全名”列包含 John Smith 字样。如果随后确认了该报警，并且执行确认的节点设置为使用操作系统或 ArchestrA 安全性，则“操作员全名”列会进行更新，以显示确认操作员的全名。否则，报警显示对象显示与 \$Operator 标记中的任何内容进行串联的计算机名。

InTouch 安全性可以在报警确认中包含操作员的全名。这在与报警检测有关的记录上也是可能的。在大多数机构中，登录 ID 并非某个人的全名，而是缩写或者是角色分类。

给供应器与接收器 InTouch 节点配置操作系统身份验证时：

- 报警显示对象在生成报警与执行确认时显示全名。
- Alarm Printer 在生成报警与执行确认时打印全名。
- Alarm DB Logger 随每个报警记录在 Operator 与 AckOperator 字段中记录域名、登录用户 ID 以及用户的全名。这样，即便某个机构中有两个员工的全名完全相同，也可以作出唯一的判断。
- “操作员”字段以“域名\用户名”的格式显示用户帐户。

# 附录 A

## 使用分布式报警显示对象

此版本的 InTouch HMI 包含 “分布式报警显示” 对象，以支持使用 InTouch 7 及更早版本开发的应用程序。对于更新版本的 InTouch HMI，您应该使用 Alarm Viewer 控件来创建报警显示。

### 关于分布式报警显示对象

“分布式报警显示” 对象提供单个显示对象来同时显示本地与远程报警。

日期	时间	状态	类	类型	优...	名称	组	供应器
03/27	17:43	UNACK_RTN	VALUE	HI	1	ReactLevel	Reactor	\InTouch
03/27	17:43	UNACK	VALUE	HI	1	ReactTemp	Reactor	\InTouch
03/27	17:43	UNACK_RTN	VALUE	HI	1	ProdLevel	Reactor	\InTouch

更新成功

默认查询

此显示对象的功能包括内置滚动条、可调整大小的列、选择多个报警、状态栏、快捷菜单，以及基于报警优先级与状态的颜色。

“分布式报警显示”对象包含一些属性，可设置报警显示对象（包括显示的信息）的外观、用于各种报警条件的颜色，以及显示的报警组与报警优先级。

如需有关显示对象的详细信息，请参阅第 396 页的“在运行时使用分布式报警显示对象”。

## 分布式报警显示对象准则

使用“分布式报警显示”对象时，需要遵循以下准则：

- 每个显示对象都必须有一个标识符，以便任何关联的 QuickScript 函数知道要修改哪个显示对象。此标识符是在报警配置对话框的显示名框中输入的；对于每个显示对象，它都必须保持唯一。
- 显示对象不应该覆盖其它的 InTouch 对象，如窗口控件或图形对象等。通过单击 WindowMaker 中的“分布式报警显示”对象，并检查该显示对象的“手柄”，可以很方便地验证这点。这些手柄不应接触到屏幕上其它的对象。
- 这些显示对象应尽量少用。在一个屏幕上放置大量的显示对象可能会导致系统性能下降。如果可能，请限制窗口上的显示对象数，可以进一步调用包含其它显示对象的窗口。



## 在设计时配置分布式报警显示对象

您可以配置：

- 常规功能，如状态栏、滚动条等。
- 列与排序顺序。
- 用于检索报警记录的报警查询。
- 显示报警记录的时间格式。
- 所显示的报警记录的字体与颜色。
- 运行时用户可以在显示对象中执行的操作，如调整列的大小、选择报警、访问快捷菜单等。

## 创建分布式报警显示对象

您可以像对待向导那样创建“分布式报警显示”对象。

**要创建“分布式报警显示”对象**

- 1 单击**向导 /ActiveX 工具栏**中的**向导工具**。此时出现**向导选择**对话框。
- 2 从向导列表中选择**报警显示**。
- 3 双击**分布式报警显示**向导。此时该对话框关闭，您的窗口再次出现，且光标处于“粘贴”模式。
- 4 在窗口中单击以粘贴“分布式报警显示”对象。要调整向导大小，请拖曳选择手柄。

现在您便可以开始配置该显示对象。

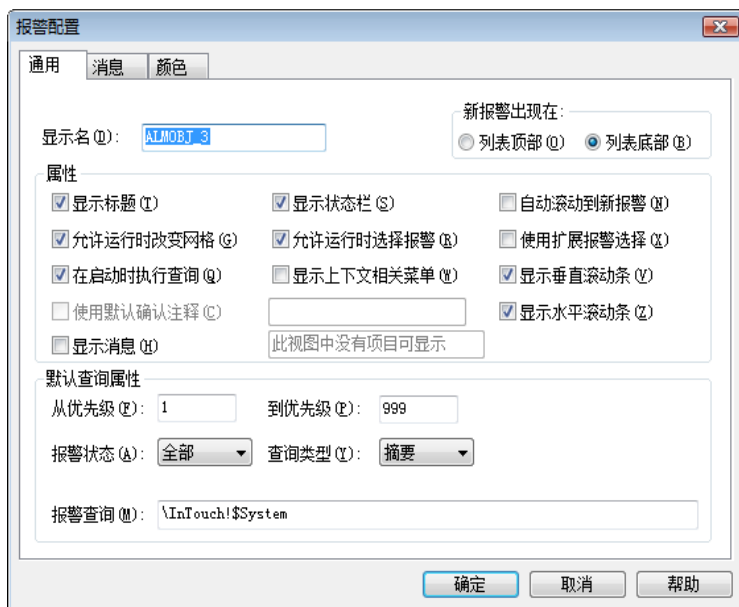
## 配置网格的外观

对于显示网格，您可以配置：

- 标题栏、状态栏及滚动条。
- 新报警出现在网格中的位置以及是否自动滚动到它们所在的位置。
- 要在没有报警记录可显示时显示的缺省消息。

### 要配置网格外观

- 1 使用鼠标右键单击“分布式报警显示”对象，然后单击属性。此时出现报警配置对话框。



- 2 在显示名框中，输入报警显示对象的名称。对于使用的每个报警显示对象，此名称必须保持唯一。此名称会在整个系统中用于引用此对象，以执行报警确认与查询之类的任务。
- 3 在新报警出现在区域中，配置希望新报警在对象中出现的位置：
  - 单击列表顶部，以便在列表顶部显示最新的报警。
  - 单击列表底部，以便在列表底部显示最新的报警。
- 4 在属性区域中，配置标题栏、状态栏及滚动条。执行以下任何操作：
  - 选择显示标题复选框以显示报警消息标题栏。
  - 选择显示状态栏复选框以显示状态栏。
  - 选择显示垂直滚动条复选框，以显示垂直滚动条。
  - 选择显示水平滚动条复选框，以显示水平滚动条。

- 5 选择**显示消息**复选框，以便在没有报警记录可显示时显示缺省消息。在方框中输入该消息。
- 6 选择**自动滚动到新报警**复选框，以便让选择项自动跳到新报警处。新报警定义为当前未在显示对象中显示的那些报警。
- 7 单击**确定**。

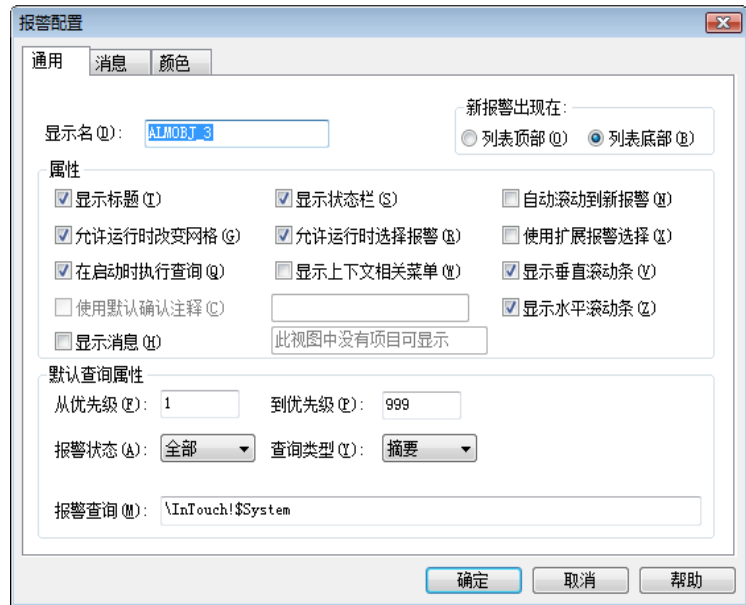
## 控制用户可以在运行时访问的功能

您可以允许运行时用户更改列设置、选择报警及打开快捷菜单。

**备注** 您可以使用脚本函数来运行快捷菜单上出现的命令。

### 要配置运行时功能

- 1 使用鼠标右键单击“分布式报警显示”对象，然后单击**属性**。此时出现**报警配置**对话框。



- 2 配置运行时可用的选项。执行以下任何操作：
  - 选择**允许运行时改变网格**复选框，以允许用户更改列设置。
  - 选择**显示上下文相关菜单**复选框以启用快捷菜单。
  - 选择**允许运行时选择报警**复选框以允许用户选择报警。
  - 选择**使用扩展报警选择**复选框，以便允许用户在按住 Ctrl 或 Shift 键的同时结合使用鼠标来选择多个报警。缺省情况是通过单击它们来切换选择的报警。
- 3 单击**确定**。

## 配置要显示哪些报警

您可以配置“分布式报警显示”对象根据以下内容来显示报警：

- 优先级。
- 状态，如已确认或未确认。
- 类型，摘要或历史。

配置报警查询时，只能使用文本。您无法使用标记。示例查询如下。

“报警组”的完整路径：

\\Node\InTouch!Group

本地“报警组”的完整路径

\InTouch!Group

另一个“组列表”：

GroupList

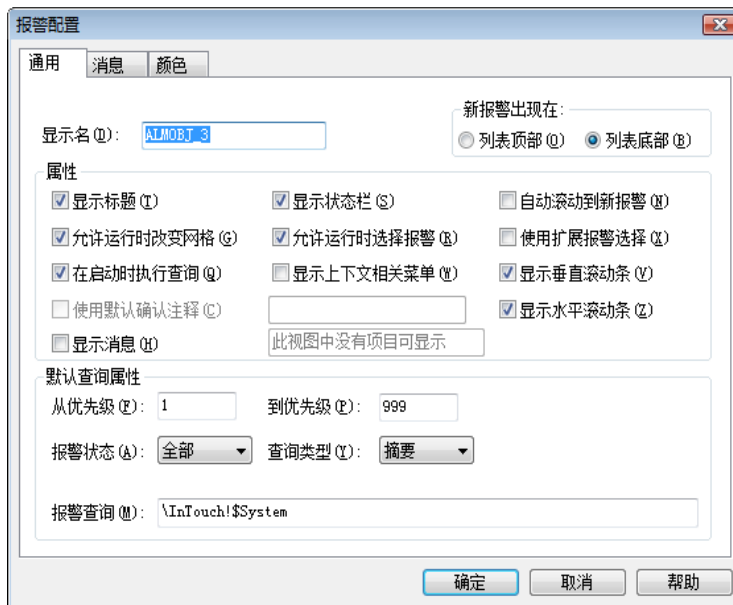
要执行多个查询，请使用空格分隔每个查询。例如：

\InTouch!Group GroupList

仅在选择在启动时执行查询复选框，或执行 almDefQuery() 函数的情况下，才使用缺省的查询属性。

### 要配置所要显示的报警

- 1 使用鼠标右键单击“分布式报警显示”对象，然后单击属性。此时出现报警配置对话框。



- 2 选择启动时执行查询复选框。

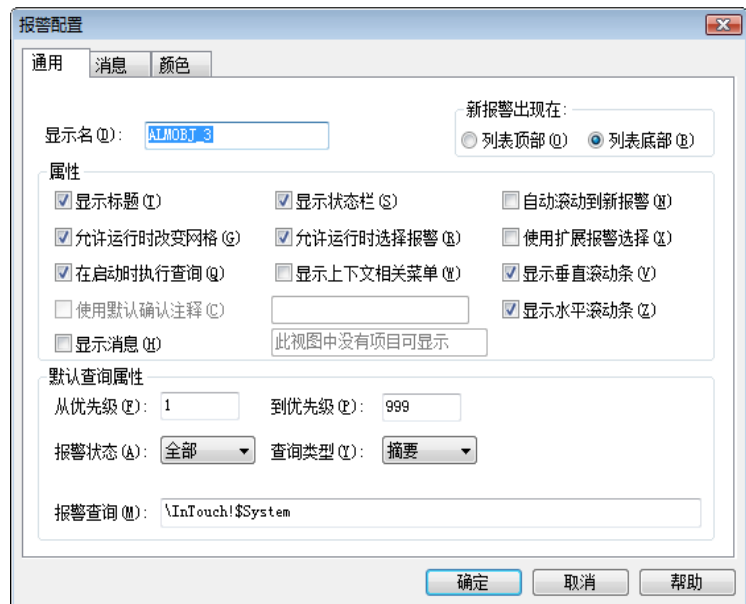
- 3 在**默认查询属性**区域中，配置对象的缺省查询。
  - 在**从优先级**框中，输入缺省的最小报警优先级。
  - 在**到优先级**框中，输入缺省的最大报警优先级。
  - 在**报警状态**列表中，单击要查询的缺省报警状态（“全部”、“未确认”、“确认”）。
  - 在**查询类型**列表中，单击“摘要”或“历史”报警类型。
  - 在**报警查询**框中，输入初始报警查询。
- 4 单击**确定**。

## 配置缺省报警注释

您可以配置要在操作员确认报警时使用的缺省注释。如果没有配置缺省注释，则在操作员确认报警时会出现一个对话框，要求操作员输入注释。该对话框可以填写，也可以留为空白。

### 要配置缺省注释

- 1 使用鼠标右键单击“分布式报警显示”对象，然后单击**属性**。此时出现**报警配置**对话框。



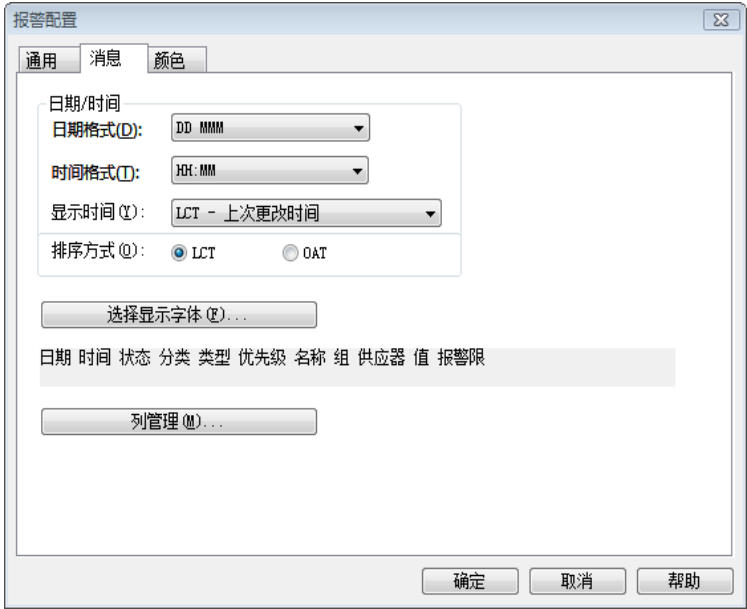
- 2 选择**显示上下文相关菜单**复选框。
- 3 选择**使用默认确认注释**复选框，然后在框中输入注释文本。
- 4 单击**确定**。

## 配置报警记录的时间格式

原始报警时间是报警发生时的日期 / 时间标签。如果标记为 I/O 标记，并且服务器能够传递时间标签，则它是来自 “I/O 服务器” 的时间标签。

### 要配置报警显示时间格式

- 1 使用鼠标右键单击 “分布式报警显示” 对象，然后单击属性。此时出现报警配置对话框。
- 2 单击消息选项卡。



- 3 在日期格式列表中，单击日期的格式。可用的格式包括：

选项	显示	选择	显示
DD MMM	28 二月	MM/DD	02/28
DD MMM YYYY	28 二月 2007	MM/DD/YY	02/28/07
DD/MM	28/07	MMM DD	二月 28
DD/MM/YY	28/02/07	MMM DD YYYY	二月 28 2007
YY/MM/DD	07/02/28	YYYY/MM/DD	2007/02/28

- 4 在**时间格式**列表中，单击时间的格式。将此列表中的值用作指定时间格式的模板。例如，要将时间指定为 10:24:30 上午，请使用 HH:MM:SS AP。模板字符如下：

字符	描述
AP	选择“上午 / 下午”格式。例如，下午三点钟显示为 03:00 下午。未指定此项的时间缺省使用 24 小时军用时间格式。例如，下午三点钟显示为 15:00。
HH	显示报警 / 事件发生时的小时值。
MM	显示报警 / 事件发生时的分钟值。
SS	显示报警 / 事件发生时的秒钟值。
SSS	显示报警 / 事件发生时的毫秒值。

- 5 在**排序方式**区域中，配置希望报警在对象中的排序方式：
- 单击 **OAT** 以使用原始报警时间，即报警发生时的日期 / 时间标签。
  - 单击 **LCT** 以使用上次报警变化时间，即报警实例最近一次改变状态的日期 / 时间标签：报警发生、子状态改变、恢复到正常或确认。
- 6 单击**确定**。

## 配置报警记录的字体

您可以设置控件中记录与标题的字体。

### 要配置字体属性

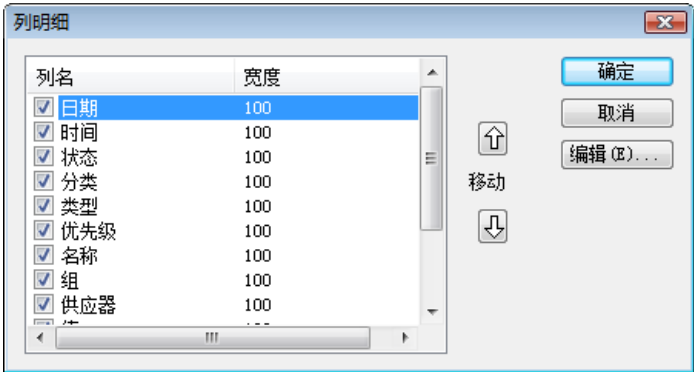
- 1 使用鼠标右键单击“分布式报警显示”对象，然后单击**属性**。此时出现**报警配置**对话框。
- 2 单击**消息**选项卡。
- 3 单击**选择显示字体**。此时出现标准的 Windows **字体**对话框。
- 4 配置字体，然后单击**确定**。

## 配置报警记录的列

您可以选择要显示的列、指定列的顺序，以及设置列名与宽度。

### 要配置显示列明细

- 1 使用鼠标右键单击“分布式报警显示”对象，然后单击属性。此时出现报警配置对话框。
- 2 单击消息选项卡。
- 3 单击列管理。此时出现列明细对话框。



- 4 选择列名列表中的复选框，以便在“分布式报警显示”对象中显示该列。您至少必须选择一列。下表介绍各个列：

列	描述
日期	按照从消息选项卡中选择的格式显示日期。
时间	按照从消息选项卡中选择的格式显示时间。
状态	显示报警的状态。
类	显示报警的类别。
类型	显示报警类型。
优先级	显示报警优先级。
名称	显示报警 / 标记名。
组	显示“报警组”名。
供应器	显示报警供应器的名称。
值	显示报警发生时标记名的值。
极限	显示标记名的报警限值。
操作员	显示与报警条件关联且已登录的操作员 ID。
注释	显示标记名注释。此注释是在数据库中定义标记名的报警时，在报警注释框中输入的。



- 5 要重新排列各个列，请选择列名，然后使用“上移”与“下移”箭头按钮。**列明细**对话框顶部出现的列名是在报警显示对象的最左侧显示的列。
- 6 要编辑列名与宽度，请选择列名，然后单击**编辑**。此时出现该列的**编辑**对话框。



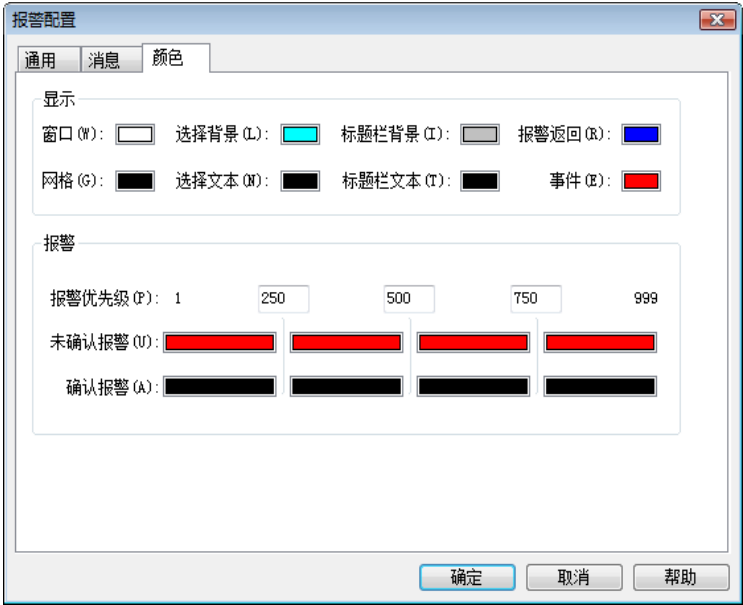
- a 如果要显示与缺省列名不同的列名，请在**新名称**框中输入新的名称。
  - b 在**新宽度**框中，输入列宽。列宽可以在从 1 到 999 个像素的范围内。缺省列宽是 100 个像素。
  - c 单击**编辑**对话框中的**确定**。
- 7 单击**列明细**对话框中的**确定**。
- 8 单击**确定**。

## 配置报警记录的颜色

您可以为“分布式报警显示”对象的各个部分配置颜色，如背景颜色与所选记录的颜色。您也可以为不同范围的报警记录配置不同的颜色。

### 要配置报警显示颜色

- 1 使用鼠标右键单击“分布式报警显示”对象，然后单击属性。此时出现报警配置对话框。
- 2 单击颜色选项卡。



- 3 在显示区域中，单击每个颜色框以打开 InTouch 调色板。单击希望以下每个部分使用的颜色：

Option	描述
窗口	设置显示对象背景颜色。
网格	设置网格颜色。
选择背景	设置突出显示的文本背景颜色。
选择文本	设置突出显示的文本颜色。
标题栏背景	设置标题栏背景颜色（仅当显示标题选项为开时设置的效果才可见）。
标题栏文本	设置标题栏文本颜色（仅当显示标题选项为开时设置的效果才可见）。
报警返回	设置返回的报警（未经确认即返回到正常的报警）的颜色。
事件	设置事件报警的颜色。

- 4 在报警优先级框中，输入报警显示对象的断点值。

- 5 单击**未确认报警**与**确认报警**颜色框，以打开 InTouch 调色板。在调色板中单击要使用的颜色。
- 6 单击**确定**。

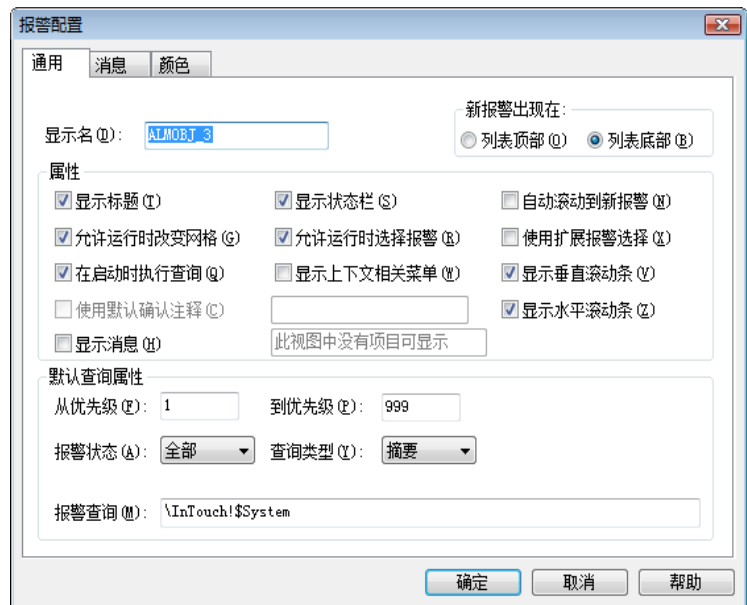
## 配置显示类型

“分布式报警显示”对象可以显示活动报警的摘要或历史报警的列表。

您也可以在运行时动态地更改显示类型。例如，这可以通过运行 `almQuery()` 脚本函数来实现。`almQuery()` 脚本函数使用的参数可以用于将指定的“分布式报警显示”对象（例如：“AlmObj\_1”）设置为指定的显示类型（例如：“摘要”）。如需有关详细信息，请参阅第 433 页的“`almQuery()` 函数”。

### 要配置查询缺省值

- 1 使用鼠标右键单击“分布式报警显示”对象，然后单击**属性**。此时出现**报警配置**对话框。



- 2 在**查询类型**列表中，单击希望用作运行时缺省值的报警显示类型。
- 3 单击**确定**。

## 使用分布式显示对象监视本地报警

您可以使用“分布式报警显示”对象来显示与确认本地及远程报警。

### 要设置仅监视本地报警的显示对象

- 1 使用鼠标右键单击“分布式报警显示”对象，然后单击**属性**。此时出现**报警配置**对话框。
- 2 在**报警查询**框中，输入：`\InTouch!$System`  
您可以将 `$System` 替换成任何有效的报警组。您也可以定义一个仅包含 `\InTouch!$System` 的报警组列表，然后使用这个组列表而不是直接引用。
- 3 根据显示类型与应用程序要求的任何过滤设置来配置其它参数。

## 在运行时使用分布式报警显示对象

“分布式报警显示”对象使用网格来显示报警消息。此网格允许动态调整列宽，只要选择列手柄并将其拖动到所需的宽度即可。此功能仅在运行时提供。

网格列的更改不会保存；因此如果在更改网格列之后关闭包含该报警显示对象的窗口，然后再打开该窗口，则网格列又将使用它们的缺省宽度。您可以在 **WindowMaker** 中调整缺省列宽。

网格允许您在列表框中选择一个或多个报警。所选的报警可通过使用 `almAckSelect()` **QuickScript** 函数进行确认。在配置“分布式报警显示”对象时，也可以定义选择行为，以便允许切换单项选择（逐项）或多项选择（在按住 **CTRL** 或 **SHIFT** 键的同时，通过单击鼠标来选择多个报警）。您可以关闭运行时选择功能。

根据报警的优先级以及它是否已确认，您也可以为显示的每个报警消息配置多达八种不同的颜色。

如果在配置时打开滚动条，则可以使用滚动条。

根据控件的配置，报警显示对象可能有用于按页滚动报警记录的控件。

## 可调整大小的显示列

“分布式报警显示”对象使用网格来存放报警消息。在运行时，只要选择一个列并拖动它，便能动态调整列宽。您也可以双击垂直网格线来自动调整列宽。此功能仅在运行时提供。列宽调整功能必须在配置时打开。

关闭包含该报警显示对象的窗口时，网格列的更改不会保存。

## 多重选择

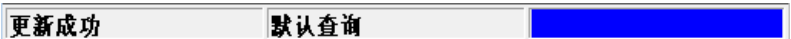
根据报警显示对象的配置，您可以在列表框中选择一个或多个报警。

## 报警消息颜色

根据报警的优先级以及报警是否已确认，最多可以给显示的每个报警消息使用八种不同的颜色。

## 状态栏

根据报警对象的配置，状态栏可以显示状态消息、当前报警查询及进度条。



状态栏的左侧部分显示控件的当前状态。  
这些指示器提供显示查询当前状态的概况，并提供“分布式报警显示”对象中可用抑制的有关详细信息。冻结生效时状态栏的右侧窗格是红色的；抑制生效时状态栏的左侧窗格是红色的。抑制生效时左侧窗格中显示“抑制”字样。

功能	描述
状态消息	状态栏左端的状态消息提供更详尽的当前查询状态的说明。
报警查询	“报警查询”提供当前报警查询的可视化指示。
进度条	状态栏右端的更新进度条提供当前查询进度的可视化指示。

状态消息内容如下：

状态消息	状态 / 指示器	进度条
无	没有查询	无
未完全更新	查询未完成	蓝色 / 绿色
更新成功	查询完成	红色
抑制	查询名	纯蓝
冻结	查询名	红色

## 快捷菜单

根据报警对象的配置，您可以使用鼠标右键单击对象来打开常用命令的快捷菜单：

单击此命令	完成此操作
确认已选项	确认所选的报警。
确认其它	确认显示对象中的所有报警，或仅确认可见的报警、所选组、所选标记名及优先级。
抑制已选项	抑制所选报警。
抑制其它	抑制显示对象中的所有报警，或仅抑制可见的报警、所选组、所选标记名及所选优先级。
查询收藏夹	打开 <b>报警查询</b> 对话框，在其中可以选择要使用的查询。
统计	打开 <b>报警统计</b> 对话框。
抑制	打开 <b>报警抑制</b> 对话框。
冻结	冻结当前显示对象。

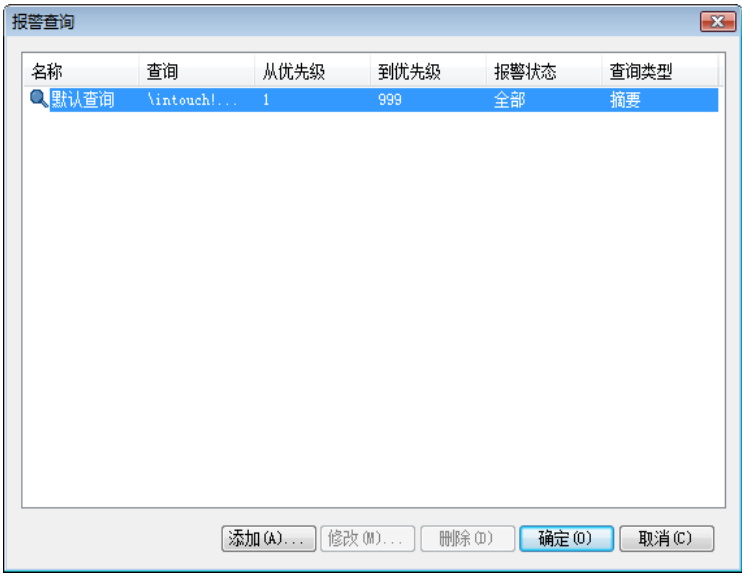
## 选择与配置报警查询收藏夹

使用快捷菜单上的**查询收藏夹**命令从先前定义的报警查询列表中快速选择某个报警查询。您也可以创建新的命名查询、编辑或删除现有的查询。

**备注** 对于“分布式报警显示”中出现的多行报警查询，换行会出现“乱码”字符。但这不影响功能。

### 要为显示对象选择报警查询

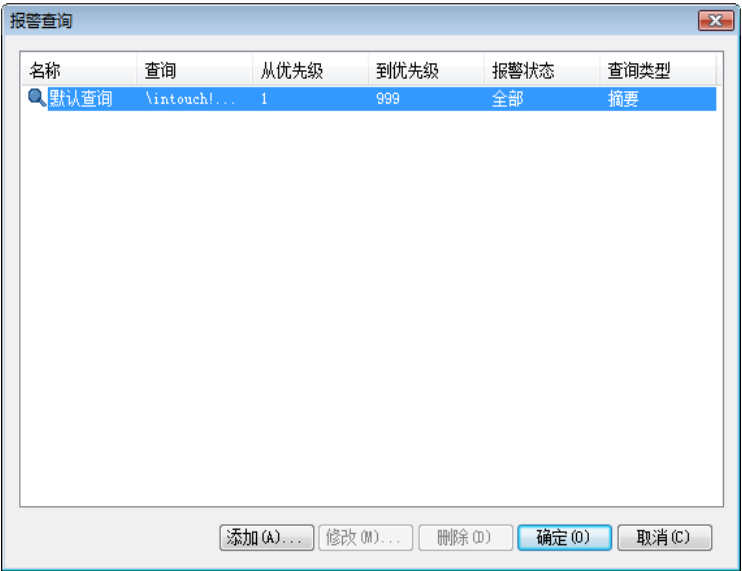
- 1 在运行时，使用鼠标右键单击“分布式报警显示”，然后单击**查询收藏夹**。此时出现**报警查询**对话框。



- 2 从当前定义的查询列表中，选择要使用的命名查询。
- 3 单击**确定**。此时“分布式报警显示”对象显示所选查询的报警信息。

要添加新的命名查询

- 1 在运行时，使用鼠标右键单击“分布式报警显示”，然后单击**查询收藏夹**。此时出现**报警查询**对话框。



- 2 单击**添加**。此时出现**添加查询**对话框。



- 3 配置查询。执行以下操作：
- 在**名称**框中，输入查询的名称。
  - 在**查询**框中，输入希望执行的一组 InTouch 报警查询。您可以指定一个或多个报警供应器与组。
  - 在**从优先级**框中，输入最小的报警优先级值（1 到 999）。在**到优先级**框中，输入最大的报警优先级值（1 到 999）。
  - 在**报警状态**列表中，单击要在报警查询中使用的报警状态。
  - 在**显示类型**区域中，单击要显示的报警类型。如需有关详细信息，请参阅第 29 页的“摘要报警与历史报警”。
- 4 单击**确定**以关闭**添加查询**对话框。
- 5 单击**报警查询**对话框中的**确定**。



### 要修改现有的命名查询

- 1 在运行时，使用鼠标右键单击“分布式报警显示”，然后单击**查询收藏夹**。此时出现**报警查询**对话框。
- 2 从当前定义的查询列表中，选择要修改的命名查询。
- 3 单击**修改**。此时出现**修改查询**对话框。
- 4 进行所需的修改，然后单击**确定**关闭**修改查询**对话框。
- 5 单击**报警查询**对话框中的**确定**。

---

**备注** 对于正在使用被修改的报警查询的其它“分布式报警显示”对象，修改内容不会自动应用于它们。

---

### 要删除现有的命名查询

- 1 在运行时，使用鼠标右键单击“分布式报警显示”，然后单击**查询收藏夹**。此时出现**报警查询**对话框。
- 2 从当前定义的查询列表中，选择要修改的命名查询。
- 3 单击**删除**。出现消息时，单击**是**。
- 4 单击**报警查询**对话框中的**确定**。

---

**备注** 对于正在使用要删除的报警查询的其它“分布式报警显示”对象，删除操作不会自动应用于它们。

---

## 使用函数与点域控制分布式报警显示对象

您可以使用函数与点域在运行时控制“分布式报警显示”对象。

如需函数返回的错误号的列表，请参阅第 464 页的“错误描述”。

### 获取或设置属性

属性可以通过 `GetPropertyX()` 函数进行访问，其中 **X** 是数据类型（**D** 代表离散、**I** 代表整型、**M** 代表消息）。例如：

```
GetPropertyM(ControlName.Property, MsgTag)
```

如需有关 `GetPropertyX()` 函数的详细信息，请参阅 *InTouch® HMI 脚本与逻辑指南* 中的第 6 章“内置函数”。

运行包含 `GetPropertyX()` 函数的脚本时，属性值保存到 **MsgTag**。如果选择了多个行，指定给 **MsgTag** 的属性是选择的多个行中第一行的标记值。

### 确认报警

“分布式报警显示”对象能够确认它可以查询到的任何报警（仅限于摘要显示对象）。“分布式报警显示”对象包含一些报警确认函数。对于用来确认本地报警与报警组的 **.Ack** 点域而言，这些函数可以起到补充作用。使用这些函数确认所有的报警、显示的报警以及所选的报警。

您也可以按照特性来确认它们。例如，组成员关系、优先级、应用程序名以及标记名。

- `almAckAll()` 函数
- `almAckDisplay()` 函数
- `almAckGroup()` 函数
- `almAckPriority()` 函数
- `almAckRecent()` 函数
- `almAckTag()` 函数
- `almAckSelect()` 函数
- `almAckSelectedGroup()` 函数
- `almAckSelectedPriority()` 函数
- `almAckSelectedTag()` 函数

### almAckAll() 函数

确认当前查询中的所有报警，包括摘要模式下“分布式报警显示”对象中当前未显示的那些报警。

类别

报警

语法

```
[Result=] almAckAll (ObjectName,Comment) ;
```

参数

*ObjectName*

报警对象的名称。例如， AlmObj\_1。

*Comment*

报警确认注释。

示例

```
MessageTag = "Acknowledge All by" + $Operator;  
almAckAll ("AlmObj_1",MessageTag);
```

另请参阅

Ack(), almAckGroup(), almAckTag(), almAckDisplay(),  
almAckRecent(), almAckSelect(), almAckSelectedGroup(),  
almAckSelectedPriority(), almAckSelectedTag()

### almAckDisplay() 函数

仅确认摘要模式下“分布式报警显示”对象中当前可见的那些报警。

类别

报警

语法

```
[Result=] almAckDisplay (ObjectName,Comment) ;
```

参数

*ObjectName*

报警对象的名称。例如， AlmObj\_1。

*Comment*

报警确认注释。

示例

```
almAckDisplay ("AlmObj_1", "Display  
Acknowledgement");
```

另请参阅

Ack(), almAckAll(), almAckGroup(), almAckTag(),  
almAckRecent(), almAckSelect(), almAckSelectedGroup(),  
almAckSelectedPriority(), almAckSelectedTag()

### almAckGroup() 函数

确认命名的“分布式报警”对象中与特定的供应器及组名匹配的所有报警。

类别

报警

语法

```
[Result=]almAckGroup( "ObjectName",  
    ApplicationName, GroupName, Comment);
```

参数

*ObjectName*

报警对象的名称。例如， AlmObj\_1。

*ApplicationName*

“应用程序” 的名称，例如 \\node1\Intouch

*GroupName*

Intouch 报警组的名称，如 \$System。

*Comment*

报警确认注释。

示例

```
MessageTag = "Acknowledge group, Turbines, by " +  
    $Operator;  
almAckGroup("AlmObj_1", "\\Intouch", "Turbine",  
    MessageTag);
```

## almAckPriority() 函数

确认指定的“分布式报警”对象中显示的所有报警，这些报警是上次的查询结果，该查询与报警的应用程序名、报警组及优先级范围匹配。

### 类别

报警

### 语法

```
[Result=]almAckPriority(ObjectName, ApplicationName,  
    GroupName, FromPri, ToPri, Comment);
```

### 参数

#### *ObjectName*

报警对象的名称。例如， AlmObj\_1。

#### *ApplicationName*

“应用程序”的名称，例如 \\node1\Intouch

#### *GroupName*

“组”的名称，例如 \$System

#### *FromPri*

报警优先级范围的起始数值。例如， 100。

#### *ToPri*

报警优先级范围的结束数值。例如， 900。

#### *Comment*

报警确认注释。

### 示例

```
almAckPriority("AlmObj_1", "\\node1\Intouch",  
    "Turbines", 10, 100, "Range 10 to 100  
    acknowledged");
```

## almAckRecent() 函数

确认最近发生的报警。

### 语法

```
[Result=]almAckRecent(ObjectName, Comment)
```

### 参数

#### *ObjectName*

报警对象的名称。例如， AlmObj\_1。

#### *Comment*

报警确认注释。

### 示例

```
almAckRecent("AlmObj_1", $DateString);
```

## almAckTag() 函数

确认指定的“分布式报警显示”对象中显示的所有报警，这些报警是上次的查询结果。报警必须与查询指定的应用程序名、组名、标记名及优先级范围匹配。

### 类别

报警

### 语法

```
[Result=]almAckTag("ObjectName", ApplicationName,  
    GroupName, TagName, FromPri, ToPri, Comment);
```

### 参数

#### *ObjectName*

报警对象的名称。例如，AlmObj\_1。

#### *ApplicationName*

应用程序的名称。例如，\\node1\Intouch。

#### *GroupName*

报警组的名称。例如，\$System。

#### *TagName*

值处于报警状态的标记的名称。

#### *FromPri*

报警优先级范围的起始数值。例如，100。

#### *ToPri*

优先级范围的结束数值。例如，900。

#### *Comment*

报警确认注释。

### 示例

```
almAckTag("AlmObj_1", "\\node1\Intouch",  
    "Turbines", "Valve1", 10, 100, "Acknowledged for  
    Valve1");
```

### 另请参阅

Ack(), almAckAll(), almAckGroup(), almAckDisplay(),  
almAckRecent(), almAckSelect(), almAckSelectedGroup(),  
almAckSelectedPriority(), almAckSelectedTag()

## almAckSelect() 函数

仅确认摘要模式下“分布式报警显示”对象中所选的那些报警。

### 类别

报警

### 语法

```
[Result=]almAckSelect (ObjectName,Comment) ;
```

### 参数

#### *ObjectName*

报警对象的名称。例如， AlmObj\_1。

#### *Comment*

报警确认注释。

### 示例

本例仅确认白班或夜班中发生的那些报警。

```
IF ($Hour >= 0 and $Hour < 8) THEN
    AckTag = "Night Shift";
ELSE
    AckTag = "Day Shift";
ENDIF;

almAckSelect ("AlmObj_1",AckTag);
```

### 另请参阅

Ack(), almAckAll(), almAckGroup(), almAckTag(),  
almAckDisplay(), almAckRecent(), almAckSelectedGroup(),  
almAckSelectedPriority(), almAckSelectedTag()

### almAckSelectedGroup() 函数

确认满足此条件的所有报警：这些报警具有相同的供应器名与组名，并且它们与指定的“分布式报警显示”对象中所选的一个或多个报警具有相同的组名。

#### 类别

报警

#### 语法

```
[Result=]almAckSelectedGroup(ObjectName,Comment);
```

#### 参数

##### *ObjectName*

报警对象的名称。例如， AlmObj\_1。

##### *Comment*

报警确认注释。

#### 示例

```
MessageTag = "Acknowledge selected groups by " +  
    $Operator;  
almAckSelectedGroup ("AlmObj_1", MessageTag);
```

#### 另请参阅

Ack(), almAckAll(), almAckGroup(), almAckTag(),  
almAckDisplay(), almAckRecent(), almAckSelect(),  
almAckSelectedPriority(), almAckSelectedTag()



## almAckSelectedPriority() 函数

确认满足此条件的所有报警：这些报警具有相同的供应器名与组名，并且它们与指定的“分布式报警显示”对象中所选的一个或多个报警具有相同的优先级值。优先级是根据所选报警记录的最小与最大优先级计算得出的。

### 类别

报警

### 语法

```
[Result=]almAckSelectedPriority(ObjectName,  
                                Comment) ;
```

### 参数

#### *ObjectName*

报警对象的名称。例如， AlmObj\_1。

#### *Comment*

报警确认注释。

### 示例

```
MessageTag = "Acknowledge selected priorities by "  
            + $Operator;  
almAckSelectedPriority ("AlmObj_1", MessageTag);
```

### 另请参阅

Ack(), almAckAll(), almAckGroup(), almAckTag(),  
almAckDisplay(), almAckRecent(), almAckSelect(),  
almAckSelectedGroup(), almAckSelectedTag()

### almAckSelectedTag() 函数

确认满足此条件的所有报警：这些报警具有相同的“标记名”、来自相同的供应器与组名，并且它们与指定的“分布式报警显示”对象中所选的一个或多个报警具有相同的优先级。

#### 类别

报警

#### 语法

```
[Result=]almAckSelectedTag (ObjectName,Comment) ;
```

#### 参数

##### *ObjectName*

报警对象的名称。例如， AlmObj\_1。

##### *Comment*

报警确认注释。

#### 示例

```
MessageTag = "Acknowledge selected tagnames by " +  
    $Operator;
```

```
almAckSelectedTag ("AlmObj_1", MessageTag);
```

#### 另请参阅

Ack(), almAckAll(), almAckGroup(), almAckTag(),  
almAckDisplay(), almAckRecent(), almAckSelect(),  
almAckSelectedGroup(), almAckSelectedPriority()

## 选择报警

您可以创建脚本从“分布式报警显示”对象中选择报警。您可以选择所有的报警，也可以仅选择所选的报警，或者是获取当前的报警数。

- almSelectAll() 函数
- almUnselectAll() 函数
- almSelectionCount() 函数
- almSelectGroup() 函数
- almSelectItem() 函数
- almSelectPriority() 函数
- almSelectTag() 函数

您也可以根据数据源、报警优先级以及 InTouch 标记来选择特定的报警。

### almSelectAll() 函数

选择 / 取消选择指定的“分布式报警显示”对象中的所有报警。

类别

报警

语法

```
[Result=]almSelectAll(ObjectName);
```

参数

*ObjectName*

报警对象的名称。例如， AlmObj\_1。

示例

```
If $AccessLevel > 8000 THEN  
    almSelectAll("AlmObj_1");  
    almAckSelect("AlmObj_1", "Ack Selected by a  
    Manager");  
ENDIF;
```

另请参阅

almSelectItem(), almSelectGroup(), almSelectPriority(),  
almSelectTag(), almUnselectAll()

### almUnselectAll() 函数

取消选择指定的“分布式报警显示”对象中所选的全部报警。

类别

报警

语法

```
[Result=]almUnselectAll(ObjectName);
```

参数

*ObjectName*

报警对象的名称。例如， AlmObj\_1。

示例

```
If $AccessLevel == 9999 THEN
    almAckSelect("AlmObj_1", "Comment");{This
    alarm can be acknowledged by only
    Administrator}
ELSE
    almUnselectAll("AlmObj_1");
ENDIF;
```

另请参阅

almSelectAll(), almSelectItem(), almSelectGroup(),  
almSelectPriority(), almSelectTag()

### almSelectionCount() 函数

返回操作员在“分布式报警显示”对象中选择的报警的数量。

类别

报警

语法

```
[Result=]almSelectionCount(ObjectName);
```

参数

*ObjectName*

报警对象的名称。例如， AlmObj\_1。

示例

将操作员在“分布式报警显示”对象中选择的报警数指定给 AlarmCount 标记。

```
AlarmCount = almSelectionCount("AlmObj_1");
```

## almSelectGroup() 函数

选择 / 取消选择指定的“分布式报警显示”对象中包含的所有报警，这些报警是上次的查询结果，并且产生的报警包含相同的报警组名。

### 类别

报警

### 语法

```
[Result=]almSelectGroup(ObjectName, ApplicationName, GroupName);
```

### 参数

#### *ObjectName*

报警对象的名称。例如， AlmObj\_1。

#### *ApplicationName*

“应用程序”的名称。例如， \\node1\Intouch。

#### *GroupName*

“组”的名称。例如， \$System。

### 示例

```
almSelectGroup("AlmObj_1", "\\Intouch", "Turbine");
```

### 另请参阅

almSelectAll(), almSelectItem(), almSelectPriority(),  
almSelectTag(), almUnSelectAll()

## almSelectItem() 函数

选择 / 取消选择报警显示对象中上次所选或取消选择的项目。

### 语法

```
[Result=]almSelectItem(ObjectName);
```

### 参数

#### *ObjectName*

报警对象的名称。例如， AlmObj\_1。

### 示例

```
almSelectItem("AlmObj_1");
```

### 另请参阅

almSelectAll(), almSelectGroup(), almSelectPriority(),  
almSelectTag(), almUnSelectAll()

### almSelectPriority() 函数

选择 / 取消选择指定的“分布式报警显示”对象中的所有报警，这些报警是上次的查询结果，并且产生的报警在指定的优先级范围内。

#### 类别

报警

#### 语法

```
[Result=]almSelectPriority( "objectName", ApplicationName,  
    GroupName, FromPri, ToPri );
```

#### 参数

##### *ObjectName*

报警对象的名称。例如， AlmObj\_1。

##### *ApplicationName*

“应用程序”的名称。例如， \\node1\Intouch。

##### *GroupName*

“组”的名称。例如， \$System。

##### *FromPri*

报警的开始优先级。例如， 100 或整型标记。

##### *ToPri*

报警的结束优先级。例如， 900 或整型标记。

#### 示例

```
almSelectPriority("AlmObj_1", "\\node1\Intouch",  
    "Turbines", 10, 100);
```

#### 另请参阅

almSelectAll(), almSelectItem(), almSelectGroup(),  
almSelectTag(), almUnSelectAll()

## almSelectTag() 函数

选择 / 取消选择指定的“分布式报警显示”对象中的所有报警，这些报警是上次的查询结果，并且它们具有指定的标记名。

### 类别

报警

### 语法

```
[Result=]almSelectTag (ObjectName, ApplicationName,  
                        GroupName, TagName, FromPri, ToPri);
```

### 参数

#### *ObjectName*

报警对象的名称。例如， AlmObj\_1。

#### *ApplicationName*

“应用程序”的名称。例如， \\node1\Intouch。

#### *GroupName*

“组”的名称。例如， \$System。

#### *TagName*

报警标记的名称。

#### *FromPri*

报警的开始优先级。例如， 100 或整型标记。

#### *ToPri*

报警的结束优先级。例如， 900 或整型标记。

### 示例

```
almSelectTag("AlmObj_1", "\\node1\Intouch",  
            "Turbines", "Valve1", 10, 100);
```

### 另请参阅

almSelectAll(), almSelectItem(), almSelectGroup(),  
almSelectPriority(), almUnSelectAll()

## 检索所选报警记录的有关信息

您可以创建脚本来返回所选报警的有关信息。在脚本中使用以下点域：

- .AlarmTime 点域
- .AlarmDate 点域
- .AlarmName 点域
- .AlarmValue 点域
- .AlarmClass 点域
- .AlarmType 点域
- .AlarmState 点域
- .AlarmLimit 点域
- .AlarmPri 点域
- .AlarmGroupSel 点域
- .AlarmAccess 点域
- .AlarmProv 点域
- .AlarmOprName 点域
- .AlarmOprNode 点域
- .AlarmComment 点域



### .AlarmTime 点域

返回报警发生的时间。该报警必须在摘要模式下的“分布式报警显示”对象中处于选择状态。

类别

报警

用法

```
[ErrorNumber=]GetPropertyM(  
    "ObjectName.AlarmTime", TagName);
```

参数

*ObjectName*

“分布式报警显示”对象的名称。例如，*AlmObj\_1*。

*TagName*

任何消息标记。

数据类型

字符串（只读）

示例

在本例中，*AlmObj\_1* 是“分布式报警显示”对象的名称，*almTime* 是内存消息标记。

```
GetPropertyM("AlmObj_1.AlarmTime", almTime);
```

如果用在“触动按钮”QuickScript 中，此语句将发生报警的时间返回到 *almTime* 标记中。

另请参阅

GetPropertyM(), .AlarmAccess, .AlarmClass,  
.AlarmComment, .AlarmDate, .AlarmLimit, .AlarmName,  
.AlarmOprName, .AlarmOprNode, .AlarmPri, .AlarmProv,  
.AlarmState, .AlarmType, .AlarmValue

### .AlarmDate 点域

返回与所选的报警关联的日期。该报警必须通过单击摘要模式下的“分布式报警显示”对象来进行选择。

类别

报警

用法

```
[ErrorNumber=]GetPropertyM("ObjectName.AlarmDate",  
    TagName);
```

参数

*ObjectName*

“分布式报警显示”对象的名称。例如， *AlmObj\_1*。

*TagName*

任何消息标记。

数据类型

字符串（只读）

示例

如果用在“触动按钮” QuickScript 中，此语句将日期返回给 almDate 标记。

```
GetPropertyM("AlmObj_1.AlarmDate",almDate);
```

AlmObj\_1 是“分布式报警显示”对象的名称， almDate 是内存消息标记，它用于检索与所选报警关联的标记的日期。

另请参阅

GetPropertyM(), .AlarmAccess, .AlarmClass,  
.AlarmComment, .AlarmLimit, .AlarmName,  
.AlarmOprName, .AlarmOprNode, .AlarmPri, .AlarmProv,  
.AlarmState, .AlarmTime, .AlarmType, .AlarmValue

### .AlarmName 点域

返回与所选报警关联的标记的名称。该报警必须通过单击摘要模式下的“分布式报警显示”对象来进行选择。

#### 类别

报警

#### 用法

```
[ErrorNumber=]GetPropertyM("ObjectName.AlarmName", TagName);
```

#### 参数

##### *ObjectName*

“分布式报警显示”对象的名称。例如， *AlmObj\_1*。

##### *TagName*

任何消息标记。

#### 数据类型

字符串（只读）

#### 示例

如果用在“触动按钮” QuickScript 中，此语句将报警名称返回给 *almName*。

```
GetPropertyM("AlmObj_1.AlarmName", almName);
```

其中， *AlmObj\_1* 是“分布式报警显示”对象的名称， *almName* 是内存消息标记，它用于检索与所选报警关联的标记的名称。

#### 另请参阅

`GetPropertyM()`, `.AlarmAccess`, `.AlarmClass`,  
`.AlarmComment`, `.AlarmDate`, `.AlarmLimit`,  
`.AlarmOprName`, `.AlarmOprNode`, `.AlarmPri`, `.AlarmProv`,  
`.AlarmState`, `.AlarmTime`, `.AlarmType`, `.AlarmValue`

### **.AlarmValue 点域**

返回与所选报警关联的标记的报警值。该报警必须通过单击摘要模式下的“分布式报警显示”对象来进行选择。

**类别**

报警

**用法**

```
[ErrorNumber=]GetPropertyM("ObjectName.AlarmValue,  
    TagName);
```

**参数**

*ObjectName*

“分布式报警显示”对象的名称。例如， *AlmObj\_1*。

*TagName*

任何消息标记。

**数据类型**

字符串（只读）

**附注**

此函数使用消息标记来检索数值。这是由于 `GetProperty` 函数不支持实数。您可以使用 `StringToReal()` 函数将结果指定给实型标记。

**示例**

如果用在“触动按钮” QuickScript 中，此语句将值返回给 `almValue`。

```
GetPropertyM("AlmObj_1.AlarmValue", almValue);
```

其中， `AlmObj_1` 是“分布式报警显示”对象的名称，`almValue` 是内存消息标记，它包含与所选报警关联的标记的报警值。

**另请参阅**

`GetPropertyM()`, `.AlarmAccess`, `.AlarmClass`,  
`.AlarmComment`, `.AlarmDate`, `.AlarmLimit`,  
`.AlarmOprName`, `.AlarmOprNode`, `.AlarmPri`, `.AlarmProv`,  
`.AlarmState`, `.AlarmTime`, `.AlarmType`, `.AlarmValue`

## .AlarmClass 点域

返回与所选报警关联的标记的报警类。该报警必须从摘要模式下的“分布式报警显示”对象中进行选择。

类别

报警

用法

```
[ErrorNumber=]GetPropertyM("ObjectName.AlarmClass"  
    ,TagName);
```

参数

*ObjectName*

“分布式报警显示”对象的名称。例如，*AlmObj\_1*。

*TagName*

任何消息标记。

数据类型

字符串（只读）

示例

以下语句返回与所选报警关联的报警类。

```
GetPropertyM("AlmObj_1.AlarmClass",almClass);
```

*AlmObj\_1* 是“分布式报警显示”对象的名称，*almClass* 是内存消息标记，它包含与所选报警关联的标记的报警类。

如果用在“触动按钮”QuickScript 中，此语句将报警的报警类返回给 *almClass* 标记。

另请参阅

GetPropertyM(), .AlarmAccess, .AlarmComment,  
.AlarmDate, .AlarmLimit, .AlarmName, .AlarmOprName,  
.AlarmOprNode, .AlarmPri, .AlarmProv, .AlarmState,  
.AlarmTime, .AlarmType, .AlarmValue

### **.AlarmType 点域**

返回与所选报警关联的标记的报警类型。该报警必须通过单击摘要模式下的“分布式报警显示”对象来进行选择。

#### **类别**

报警

#### **用法**

```
[ErrorNumber=]GetPropertyM("ObjectName.AlarmType",  
    TagName);
```

#### **参数**

*ObjectName*

“分布式报警显示”对象的名称。例如，*AlmObj\_1*。

*TagName*

任何消息标记。

#### **数据类型**

字符串（只读）

#### **示例**

如果用在“触动按钮” QuickScript 中，则操作员确认报警时，此语句将所选报警的类型返回给 *almType* 标记。

```
GetPropertyM("AlmObj_1.AlarmType",almType);
```

其中，*AlmObj\_1* 是“分布式报警显示”对象的名称，*almType* 是内存消息标记，它包含与所选报警关联的标记的报警类型。

#### **另请参阅**

*GetPropertyM()*, *.AlarmAccess*, *.AlarmClass*,  
*.AlarmComment*, *.AlarmDate*, *.AlarmLimit*, *.AlarmName*,  
*.AlarmOprName*, *.AlarmOprNode*, *.AlarmPri*, *.AlarmProv*,  
*.AlarmState*, *.AlarmTime*, *.AlarmValue*

### .AlarmState 点域

返回所选报警的状态。该报警必须通过单击摘要模式下的“分布式报警显示”对象来进行选择。

类别

报警

用法

```
[ErrorNumber=]GetPropertyM(  
    "ObjectName.AlarmState", TagName);
```

参数

*ObjectName*

“分布式报警显示”对象的名称。例如，*AlmObj\_1*。

*TagName*

任何消息标记。

数据类型

字符串（只读）

示例

如果用在“触动按钮”QuickScript 中，此语句将所选报警的状态返回给 almState 标记。

```
GetPropertyM("AlmObj_1.AlarmState", almState);
```

其中，AlmObj\_1 是“分布式报警显示”对象的名称，almState 是内存消息标记，它包含与所选报警关联的标记的报警状态。

另请参阅

GetPropertyM(), .AlarmAccess, .AlarmClass,  
.AlarmComment, .AlarmDate, .AlarmLimit, .AlarmName,  
.AlarmOprName, .AlarmOprNode, .AlarmPri, .AlarmProv,  
.AlarmTime, .AlarmType, .AlarmValue

### **.AlarmLimit 点域**

返回与所选报警关联的标记的报警限。该报警必须通过单击摘要模式下的“分布式报警显示”对象来进行选择。

#### **类别**

报警

#### **用法**

```
[ErrorNumber=]GetPropertyM  
("ObjectName.AlarmLimit", TagName);
```

#### **参数**

##### *ObjectName*

“分布式报警显示”对象的名称。例如， *AlmObj\_1*。

##### *TagName*

任何消息标记。

#### **数据类型**

字符串（只读）

#### **附注**

此函数使用消息标记来检索数值。这是由于 **GetProperty** 函数不支持实数。您可以使用 **StringToReal()** 函数将结果指定给实型标记。

#### **示例**

如果用在触动按钮 **QuickScript** 中，此语句将所选报警的限制返回给 **almLimit** 标记。

```
GetPropertyM("AlmObj_1.AlarmLimit", almLimit);
```

其中， **AlmObj\_1** 是“分布式报警显示”对象的名称，**almLimit** 是内存消息标记，它包含与所选报警关联的标记的报警限。

#### **另请参阅**

**GetPropertyM()**, **.AlarmAccess**, **.AlarmClass**,  
**.AlarmComment**, **.AlarmDate**, **.AlarmName**,  
**.AlarmOprName**, **.AlarmOprNode**, **.AlarmPri**, **.AlarmProv**,  
**.AlarmState**, **.AlarmTime**, **.AlarmType**, **.AlarmValue**



### .AlarmPri 点域

返回与所选报警关联的标记的优先级 (1-999)。该报警必须通过单击摘要模式下的“分布式报警显示”对象来进行选择。

#### 类别

报警

#### 用法

```
[ErrorNumber=]GetPropertyM("ObjectName.AlarmPri",  
    TagName);
```

#### 参数

*ObjectName*

“分布式报警显示”对象的名称。例如，*AlmObj\_1*。

*TagName*

任何消息标记。

#### 数据类型

字符串（只读）

#### 示例

如果用在“触动按钮”QuickScript 中，此语句将报警优先级返回给 almPrilvl 标记。

```
GetPropertyM("AlmObj_1.AlarmPri",almPrilvl);
```

其中，AlmObj\_1 是“分布式报警显示”对象的名称，almPrilvl 是内存消息标记，它包含与所选报警关联的标记的优先级。

#### 另请参阅

GetPropertyM(), .AlarmAccess, .AlarmClass, .AlarmComment, .AlarmDate, .AlarmLimit, .AlarmName, .AlarmOprName, .AlarmOprNode, .AlarmProv, .AlarmState, .AlarmTime, .AlarmType, .AlarmValue

### **.AlarmGroupSel 点域**

返回与所选报警关联的标记的报警组。该报警必须通过单击摘要模式下的“分布式报警显示”对象来进行选择。

#### **类别**

报警

#### **用法**

```
[ErrorNumber=]GetPropertyM(  
    "ObjectName.AlarmGroupSel", TagName);
```

#### **参数**

*ObjectName*

“分布式报警显示”对象的名称。例如， *AlmObj\_1*。

*TagName*

任何消息标记。

#### **数据类型**

字符串（只读）

#### **示例**

如果用在“触动按钮” QuickScript 中，此语句将所选报警组的名称返回给 almGroup 标记。

```
GetPropertyM("AlmObj_1.AlarmGroupSel", almGroup);
```

其中， AlmObj\_1 是“分布式报警显示”对象的名称，  
almGroup 是内存消息标记，它包含与所选报警关联的标记的报警组。

#### **另请参阅**

GetPropertyM(), .AlarmGroup, .AlarmName

## .AlarmAccess 点域

返回与所选报警关联的标记的“访问名”。该报警记录必须通过单击摘要模式下的“分布式报警显示”对象来进行选择。

### 类别

报警

### 用法

```
GetPropertyM("Objectname.AlarmAccess", TagName);
```

### 参数

#### *ObjectName*

“分布式报警显示”对象的名称。例如， *AlmObj\_1*。

#### *TagName*

任何消息标记。

### 数据类型

字符串（只读）

### 示例

如果用在“触动按钮” QuickScript 中，此语句将与报警关联的标记的“访问名”返回给 almAccess 标记。

```
GetPropertyM("AlmObj_1.AlarmAccess", almAccess);
```

其中， AlmObj\_1 是“分布式报警显示”对象的名称，  
almAccess 是内存消息标记，它包含与所选报警关联的标记的“访问名”。

### 另请参阅

GetPropertyM(), .AlarmClass, .AlarmComment, .AlarmDate,  
.AlarmLimit, .AlarmName, .AlarmOprName,  
.AlarmOprNode, .AlarmPri, .AlarmProv, .AlarmState,  
.AlarmTime, .AlarmType

### **.AlarmProv 点域**

返回与所选报警关联的标记的报警供应器。该报警必须通过单击摘要模式下的“分布式报警显示”对象来进行选择。

#### **类别**

报警

#### **用法**

```
[ErrorNumber=]GetPropertyM(  
    "ObjectName.AlarmProv", TagName);
```

#### **参数**

*ObjectName*

“分布式报警显示”对象的名称。例如， *AlmObj\_1*。

*TagName*

任何消息标记。

#### **数据类型**

字符串（只读）

#### **示例**

如果用在“触动按钮” QuickScript 中，此语句将供应器名返回给 almProv 标记。

```
GetPropertyM("AlmObj_1.AlarmProv", almProv);
```

AlmObj\_1 是“分布式报警显示”对象的名称， almProv 是内存消息标记，它包含与所选报警关联的标记的供应器名。

#### **另请参阅**

GetPropertyM(), .AlarmAccess, .AlarmClass,  
.AlarmComment, .AlarmDate, .AlarmLimit, .AlarmName,  
.AlarmOprName, .AlarmOprNode, .AlarmPri, .AlarmState,  
.AlarmTime, .AlarmType, .AlarmValue

### .AlarmOprName 点域

返回确认所选报警且已登录的操作员的名称。该报警必须通过单击摘要模式下的“分布式报警显示”对象来进行选择。

类别

报警

用法

```
[ErrorNumber=]GetPropertyM(  
    "ObjectName.AlarmOprName", TagName);
```

参数

*ObjectName*

“分布式报警显示”对象的名称。例如， *AlmObj\_1*。

*TagName*

任何消息标记。

数据类型

字符串（只读）

示例

```
GetPropertyM("AlmObj_1.AlarmOprName", almOprName);
```

其中， *AlmObj\_1* 是“分布式报警显示”对象的名称， *almOprName* 是内存消息标记，它包含与标记关联的报警所对应的操作员的名称。

如果用在“触动按钮” QuickScript 中，此语句将操作员的名称返回给 *almOprName* 标记。

另请参阅

GetPropertyM(), .AlarmAccess, .AlarmClass,  
.AlarmComment, .AlarmDate, .AlarmLimit, .AlarmName,  
.AlarmOprNode, .AlarmPri, .AlarmProv, .AlarmState,  
.AlarmTime, .AlarmType, .AlarmValue

### **.AlarmOprNode 点域**

返回与所选报警关联的标记的操作员节点。该报警必须通过单击摘要模式下的“分布式报警显示”对象来进行选择。

在“终端服务”环境下确认报警时，“操作员节点”是相应的操作员建立“终端服务”会话的客户端机器的名称。如果无法检索节点名，则使用节点的 IP 地址代替。

#### **类别**

报警

#### **用法**

```
[ErrorNumber=]GetPropertyM(  
    "ObjectName.AlarmOprNode", TagName);
```

#### **参数**

##### *ObjectName*

“分布式报警显示”对象的名称。例如， *AlmObj\_1*。

##### *TagName*

任何消息标记。

#### **数据类型**

字符串（只读）

#### **示例**

```
GetPropertyM("AlmObj_1.AlarmOprNode", almOprNode);
```

其中， *AlmObj\_1* 是“分布式报警显示”对象的名称，  
*almOprNode* 是内存消息标记，它包含与所选报警关联的标记的操作员节点名。

如果用在“触动按钮” QuickScript 中，此语句将操作员节点返回给 *almOprNode* 标记。

#### **另请参阅**

*GetPropertyM()*, *.AlarmAccess*, *.AlarmClass*,  
*.AlarmComment*, *.AlarmDate*, *.AlarmLimit*, *.AlarmName*,  
*.AlarmOprName*, *.AlarmPri*, *.AlarmProv*, *.AlarmState*,  
*.AlarmTime*, *.AlarmType*, *.AlarmValue*

## **.AlarmComment 点域**

返回报警注释，这是描述报警（而非标记）的可读写文本字符串。缺省条件下，此注释在新应用程序中为空。

不过，在将旧有的 InTouch 应用程序转换为 InTouch 7.11 或更高的版本时，为保证后向兼容性，会将标记注释复制到 .AlarmComment 点域中。

### **类别**

报警

### **用法**

```
[ErrorNumber=]GetPropertyM(  
    "ObjectName.AlarmComment", TagName);
```

### **参数**

*ObjectName*

“分布式报警显示”对象的名称。例如，*AlmObj\_1*。

*TagName*

任何消息标记。

### **数据类型**

字符串（只读）

### **示例**

下例返回“分布式报警显示”对象 *AlmObj\_1* 中所选标记的报警注释，并将它放入 *almComment* 标记中：

```
GetPropertyM("AlmObj_1.AlarmComment",  
    almComment);
```

### **另请参阅**

*GetPropertyM()*, *.AlarmAccess*, *.AlarmClass*, *.AlarmDate*,  
*.AlarmLimit*, *.AlarmName*, *.AlarmOprName*,  
*.AlarmOprNode*, *.AlarmPri*, *.AlarmProv*, *.AlarmState*,  
*.AlarmTime*, *.AlarmType*, *.AlarmValue*

## 设置报警查询

使用以下查询函数从报警内存中检索记录。

- almDefQuery() 函数
- almQuery() 函数
- almSetQueryByName() 函数

### almDefQuery() 函数

使用缺省属性执行查询，以更新指定的“分布式报警显示”对象。

**类别**

报警

**语法**

```
[Result=]almDefQuery(ObjectName);
```

**参数**

*ObjectName*

“分布式报警显示”对象的名称。例如， AlmObj\_1。

**附注**

缺省查询属性在 WindowMaker 中开发“分布式报警显示”对象时指定。

**示例**

```
almDefQuery(“AlmObj_1”);
```

**另请参阅**

almQuery(), almSetQueryByName()



## almQuery() 函数

执行查询以更新指定的“分布式报警显示”对象，并使用指定的参数。

### 类别

报警

### 语法

```
[Result=]almQuery(ObjectName,AlarmList,FromPri,ToPri,State,Type);
```

### 参数

#### *ObjectName*

报警对象的名称。例如， AlmObj\_1。

#### *AlarmList*

设置“报警查询/名称管理器”的别名以执行查询，例如“\intouch!\$System”或消息标记。

#### *FromPri*

要显示的报警的开始优先级。例如， 100 或整型标记。

#### *ToPri*

要显示的报警的结束优先级。例如， 900 或整型标记。

#### *State*

指定要显示的报警类型。例如，“UnAck”或“消息”标记。有效状态包括“全部”、“未确认”或“确认”。

#### *Type*

更新的显示对象中出现的报警记录的类型：

"Hist" = 历史报警

"Summ" = 摘要报警

### 示例

此语句检索 MyAlarmListGroup 中指定的、优先级为 500 到 600 的所有历史报警。这些报警出现在 AlmObj\_1 报警显示对象中。

```
almQuery("AlmObj_1","MyAlarmListGroup",500,600,
  "All","Hist");
```

在本例中， MyAlarmListGroup 是从“名称管理器”设置中配置的报警列表。

### 另请参阅

almDefQuery(), almSetQueryByName()

### almSetQueryByName() 函数

使用用户自定义查询收藏夹文件的参数，对“分布式报警显示”对象的指定实例启动新的报警查询。

#### 类别

报警

#### 语法

```
[Result=]almSetQueryByName (ObjectName, QueryName);
```

#### 参数

##### *ObjectName*

报警对象的名称。例如，AlmObj\_1。

##### *QueryName*

使用“查询收藏夹”创建的查询的名称。

#### 附注

这是“分布式报警显示”对象特定实例的查询。屏幕上可能有多个这样的显示对象，每个都有自己的查询。

#### 示例

本例使用名为 "Turbine Queries" 的查询中的参数开始新的查询。

```
almSetQueryByName ( "AlmObj_1" , "Turbine Queries" );
```

#### 另请参阅

almQuery(), almDefQuery()

## 检查当前查询属性

使用以下点域返回报警内存查询的状态。

- .AlarmGroup 点域
- .QueryType 点域
- .QueryState 点域
- .Successful 点域
- .PriFrom 点域
- .PriTo 点域

### .AlarmGroup 点域

包含用于填充“分布式报警显示”对象的当前查询。

**类别**

报警

**用法**

```
[ErrorNumber=]GetPropertyM(  
    "ObjectName.AlarmGroup", TagName);
```

**参数**

*ObjectName*

报警对象的名称。例如， AlmObj\_1。

*TagName*

任何消息标记。

**附注**

此只读点域包含指定的“分布式报警显示”对象使用的当前报警查询。此查询可以是报警组的列表，也可以是报警供应器的直接引用。

**数据类型**

字符串（只读）

**示例**

此语句将“分布式报警显示”对象 AlmObj\_1 使用的当前报警查询返回给 CurrentQuery 标记：

```
GetPropertyM("AlmObj_1.AlarmGroup", CurrentQuery);
```

**另请参阅**

GetPropertyM(), .AlarmGroupSel, .AlarmName

### **.QueryType 点域**

显示当前报警查询的类型。

#### **类别**

报警

#### **用法**

```
[ErrorNumber=]GetPropertyI(  
    "ObjectName.QueryType",TagName);
```

#### **参数**

*ObjectName*

报警对象的名称。例如， AlmObj\_1。

*TagName*

任何整型标记

#### **附注**

此只读点域包含指定的“分布式报警显示”对象当前使用的查询类型。

#### **数据类型**

整型（只读）

#### **有效值**

1 = 历史

2 = 摘要

#### **示例**

以下语句将“分布式报警显示”对象 AlmObj\_1 的当前查询类型返回给 AlmQueryType 标记：

```
GetPropertyI("AlmObj_1.QueryType",AlmQueryType);
```

#### **另请参阅**

GetPropertyI().QueryState

## **.QueryState 点域**

显示当前报警状态查询过滤器。

### **类别**

报警

### **用法**

```
[ErrorNumber=]GetPropertyI(  
    "ObjectName.QueryState", TagName);
```

### **参数**

#### *ObjectName*

报警对象的名称。例如， AlmObj\_1。

#### *TagName*

任何整型标记

### **附注**

此只读点域包含指定的“分布式报警显示”对象当前使用的查询过滤器。

### **数据类型**

整型（只读）

### **有效值**

0 = 全部

1 = 未确认

2 = 已确认

### **示例**

以下语句将“分布式报警显示”对象 AlmObj\_1 的当前查询过滤器返回给 AlmQueryState 标记：

```
GetPropertyI("AlmObj_1.QueryState",  
    AlmQueryState);
```

### **另请参阅**

GetPropertyI(), .QueryType

**.Successful 点域**

指出当前查询是否成功。

**类别**

报警

**用法**

```
[ErrorNumber=]GetPropertyD(  
    "ObjectName.Successful", TagName);
```

**参数****ObjectName**

报警对象的名称。例如， AlmObj\_1。

**TagName**

离散标记，处理函数时用于存放属性值。

**附注**

此只读点域包含指定的“分布式报警显示”对象使用的上一个查询的状态。

**数据类型**

离散（只读）

**有效值**

0 = 查询错误

1 = 查询成功

**示例**

以下语句将“分布式报警显示”对象 AlmObj\_1 的上一个查询的状态返回给 AlmFlag 标记。

```
GetPropertyD("AlmObj_1.Successful", AlmFlag);
```

**另请参阅**

GetPropertyD()

### **.PriFrom 点域**

返回当前查询使用的报警优先级范围的最小值。

#### **类别**

报警

#### **用法**

```
[ErrorNumber=]GetPropertyI("ObjectName.PriFrom",  
    Tagname);
```

#### **参数**

*ObjectName*

“分布式报警显示”对象的名称。例如， *AlmObj\_1*。

*TagName*

任何整型标记。

#### **数据类型**

整型（只读）

#### **示例**

以下语句将“分布式报警显示”对象 *AlmObj\_1* 所使用的查询的最小优先级值返回给 *MinPri* 整型标记：

```
GetPropertyI("AlmObj_1.PriFrom",MinPri);
```

#### **另请参阅**

`GetPropertyI()`, `.PriTo`, `.AlarmPri`

### **.PriTo 点域**

包含当前查询使用的报警优先级范围的最大值。

#### **用法**

```
[ErrorNumber=]GetPropertyI("ObjectName.PriTo",  
    Tagname);
```

#### **参数**

*ObjectName*

“分布式报警显示”对象的名称。例如， *AlmObj\_1*。

*TagName*

整型标记，处理函数时用于存放属性值。

#### **数据类型**

整型（只读）

#### **示例**

以下语句将“分布式报警显示”对象 *AlmObj\_1* 所使用的查询的最大优先级值返回给 *MaxPri* 整型标记：

```
GetPropertyI("AlmObj_1.PriTo",MaxPri);
```

#### **另请参阅**

`GetPropertyI()`, `.PriFrom`, `.AlarmPri`



## 检查分布式报警显示对象的更新

使用以下点域找出“分布式报警显示”对象是否包含当前的所有报警，是否存在待处理的更新。

- .ListChanged 点域
- .PendingUpdates 点域

### .ListChanged 点域

指出“分布式报警显示”对象是否有任何新的报警或更新。

**类别**

报警

**用法**

```
[ErrorNumber=]GetPropertyD(
  "ObjectName.ListChanged°±, TagName);
```

**参数**

*ObjectName*

报警对象的名称。例如， AlmObj\_1。

*TagName*

离散标记，处理函数时用于存放属性值。

**附注**

此只读点域包含有关以下状态的信息：是否存在任何更改需要在“分布式报警显示”对象中进行更新。此属性在读取时自动重置。

**数据类型**

离散（只读）

**有效值**

0 = 显示对象没有新的报警或更新

1 = 显示对象有新的更新

**示例**

以下语句将“分布式报警显示”对象 AlmObj\_1 的任何新报警或更新的状态返回给 AlmDispStat 标记：

```
GetPropertyD( "AlmObj_1.ListChanged"
, AlmDispStat);
```

**另请参阅**

GetPropertyD()

### **.PendingUpdates 点域**

指出“分布式报警显示”对象待处理的更新数。冻结显示并创建新的报警记录时，通常有待处理的更新。这些更新不显示，但是待处理更新计数会增加。

#### **类别**

报警

#### **用法**

```
[ErrorMessage=]GetPropertyI(  
    "ObjectName.PendingUpdates" , TagName);
```

#### **参数**

##### *ObjectName*

报警对象的名称。例如，AlmObj\_1。

##### *TagName*

整型标记，处理函数时用于存放属性值。

#### **附注**

此只读点域包含指定的“分布式报警显示”对象中待处理的更新数。任何大于零的数值都表示“分布式报警显示”对象有新的报警数据。每次刷新对象时，此值都会重置。

#### **数据类型**

整型（只读）

#### **示例**

如果“分布式报警显示”对象 AlmObj\_1 有任何待处理的更新，则以下语句会将整数 1 或更大的数值返回给

AlarmPendingUpdates 标记：

```
GetPropertyI("AlmObj_1.PendingUpdates", AlarmPendingUpdates);
```

#### **另请参阅**

GetPropertyI()

## 抑制报警

在与排除标准匹配的报警接收器上，“分布式报警显示”对象可以抑制一个或多个报警。如果某个报警与排除标准匹配，则它不会出现在显示对象的实例中。

您可以使用 QuickScript 函数来抑制报警。

- almSuppressAll() 函数
- almUnsuppressAll() 函数
- almSuppressDisplay() 函数
- almSuppressGroup() 函数
- almSuppressPriority() 函数
- almSuppressTag() 函数
- almSuppressSelected() 函数
- almSuppressSelectedGroup() 函数
- almSuppressSelectedPriority() 函数
- almSuppressSelectedTag() 函数
- almSuppressRetain() 函数
- .SuppressRetain 点域

### almSuppressAll() 函数

抑制显示当前查询中的当前以及将来要发生的这些报警，包括摘要模式下“分布式报警显示”对象中当前未显示的那些报警。

#### 语法

```
[Result=] almSuppressAll (ObjectName);
```

#### 参数

##### *ObjectName*

报警对象的名称。例如， AlmObj\_1。

#### 附注

此函数类似于 almAckAll()，通过确定希望在显示对象中查看的所有报警，并上下滚动来查看所有这些报警，以确定要抑制哪些报警。

#### 示例

```
almSuppressAll( "AlmObj_1" );
```

#### 另请参阅

almSuppressGroup(), almSuppressTag(),  
almSuppressDisplay(), almSuppressPriority(),  
almSuppressRetain(), almSuppressSelected(),  
almSuppressSelectedGroup(),  
almSuppressSelectedPriority(), almSuppressSelectedTag(),  
almUnSuppressAll()

## almUnsuppressAll() 函数

撤销抑制所有被抑制的报警。

### 语法

```
[Result=] almUnsuppressAll (ObjectName) ;
```

### 参数

#### ObjectName

报警对象的名称。例如， AlmObj\_1。

### 示例

```
almUnsuppressAll ( “AlmObj_1” ) ;
```

### 另请参阅

almSuppressAll(), almSuppressGroup(), almSuppressTag(),  
almSuppressDisplay(), almSuppressPriority(),  
almSuppressRetain(), almSuppressSelected(),  
almSuppressSelectedGroup(),  
almSuppressSelectedPriority(), almSuppressSelectedTag()

## almSuppressDisplay() 函数

抑制显示摘要模式下 “分布式报警显示” 对象中当前的可见报警以及将来要发生的这些报警。

### 语法

```
[Result=] almSuppressDisplay (ObjectName) ;
```

### 参数

#### ObjectName

报警对象的名称。例如， AlmObj\_1。

### 附注

此函数类似于相应的 almAckDisplay() 函数，通过确定当前显示的所有报警来确定要抑制的报警。

### 示例

```
almSuppressDisplay ( “AlmObj_1” ) ;
```

### almSuppressGroup() 函数

抑制显示当前以及将来要发生的任何报警，它们具有指定的供应器与组名。

#### 语法

```
[Result=]almSuppressGroup(ObjectName,  
    ApplicationName,GroupName);
```

#### 参数

##### *ObjectName*

报警对象的名称。例如， AlmObj\_1。

##### *ApplicationName*

应用程序的名称。例如， \\node1\InTouch

##### *GroupName*

报警组的名称。例如， \$System

#### 示例

```
almSuppressGroup("AlmObj_1","\\InTouch","Turbines"  
);
```

#### 另请参阅

almSuppressAll(), almSuppressTag(), almSuppressDisplay(),  
almSuppressPriority(), almSuppressRetain(),  
almSuppressSelected(), almSuppressSelectedGroup(),  
almSuppressSelectedPriority(), almSuppressSelectedTag(),  
almUnSuppressAll()

## almSuppressPriority() 函数

抑制显示当前以及将来要发生的任何特定报警：它们在指定的优先级范围内，并且具有相同的供应器名与“组名”。

### 语法

```
[Result=]almSuppressPriority(ObjectName,  
                             ApplicationName, GroupName, FromPri, ToPri) ;
```

### 参数

#### *ObjectName*

报警对象的名称。例如， AlmObj\_1。

#### *ApplicationName*

应用程序的名称。例如， \\node1\InTouch

#### *GroupName*

“组”的名称。例如， \$System

#### *FromPri*

报警的开始优先级。例如， 100 或整型标记。

#### *ToPri*

报警的结束优先级。例如， 900 或整型标记。

### 示例

```
almSuppressPriority("AlmObj_1", "\\node1\Intouch",  
                  "Turbines", 10, 100) ;
```

### 另请参阅

almSuppressAll(), almSuppressGroup(), almSuppressTag(),  
almSuppressDisplay(), almSuppressRetain(),  
almSuppressSelected(), almSuppressSelectedGroup(),  
almSuppressSelectedPriority(), almSuppressSelectedTag(),  
almUnSuppressAll()

## almSuppressTag() 函数

抑制显示当前以及将来要发生的任何报警：它们属于指定的标记名，并且具有相同的供应器名、组名以及优先级范围。

### 类别

报警

### 语法

```
[Result=]almSuppressTag (ObjectName, ApplicationName,  
    GroupName, TagName, FromPri, ToPri, AlarmClass,  
    AlarmType) ;
```

### 参数

#### ObjectName

报警对象的名称。例如， AlmObj\_1。

#### ApplicationName

应用程序的名称。例如， \\node1\InTouch

#### GroupName

“组” 的名称。例如， \$System

#### TagName

报警标记的名称。

#### FromPri

报警的开始优先级。例如， 100 或整型标记。

#### ToPri

报警的结束优先级。例如， 900 或整型标记。

#### AlarmClass

报警的类。例如，“Value”。

#### AlarmType

报警的类型。例如，“HiHi”。

### 示例

```
almSuppressTag ("AlmObj_1", "\\node1\Intouch",  
    "Turbines", "Valve1", 10, 100, "Value", "LoLo") ;
```

### 另请参阅

almSuppressAll(), almSuppressGroup(),  
almSuppressDisplay(), almSuppressPriority(),  
almSuppressRetain(), almSuppressSelected(),  
almSuppressSelectedGroup(),  
almSuppressSelectedPriority(), almSuppressSelectedTag(),  
almUnSuppressAll()



## almSuppressSelected() 函数

抑制显示摘要模式下“分布式报警显示”对象中当前所选的报警以及将来要发生的这些报警。

### 类别

报警

### 语法

```
[Result=]almSuppressSelected(ObjectName) ;
```

### 参数

*ObjectName*

报警对象的名称。例如， AlmObj\_1。

### 附注

此函数类似于 almAckSelect() 函数，根据显示对象中所选的报警来确定要抑制的报警。

### 示例

```
almSuppressSelected("AlmObj_1") ;
```

### 另请参阅

almSuppressAll(), almSuppressGroup(), almSuppressTag(),  
almSuppressDisplay(), almSuppressPriority(),  
almSuppressSelectedGroup(),  
almSuppressSelectedPriority(), almSuppressSelectedTag(),  
almUnSuppressAll()

### almSuppressSelectedGroup() 函数

抑制显示当前以及将来要发生的特定报警：它们与一个或多个所选的报警属于相同的组，并且在指定的“分布式报警显示”对象中具有相同的供应器名。

#### 类别

报警

#### 语法

```
[Result=]almSuppressSelectedGroup(ObjectName);
```

#### 参数

##### *ObjectName*

报警对象的名称。例如， AlmObj\_1。

#### 附注

此函数类似于 almAckSelectedGroup() 函数，首先确定所选的报警，然后确定这些报警所属的组，并抑制那些组中的报警将来的实例。

#### 示例

```
almSuppressSelectedGroup("AlmObj_1");
```

#### 另请参阅

almSuppressAll(), almSuppressGroup(), almSuppressTag(),  
almSuppressDisplay(), almSuppressSelected(),  
almSuppressSelectedPriority(), almSuppressSelectedTag()

## almSuppressSelectedPriority() 函数

抑制显示当前以及将来要发生的特定报警：它们与一个或多个所选的报警属于相同的优先级，并且在指定的“分布式报警显示”对象中具有相同的供应器名与“组”标记。

### 类别

报警

### 语法

```
[Result=]almSuppressSelectedPriority(ObjectName);
```

### 参数

#### *ObjectName*

报警对象的名称。例如， AlmObj\_1。

### 附注

优先级是根据所选报警记录的最小与最大优先级计算得出的。

此函数类似于 almAckSelectedPriority() 函数，首先确定显示对象中的所选报警，然后确定那些报警相应的优先级，并抑制将来要发生的具有相同优先级的报警。

### 示例

```
almSuppressSelectedPriority("AlmObj_1");
```

### 另请参阅

almSuppressAll(), almSuppressGroup(),  
almSuppressTagName(), almSuppressDisplay(),  
almSuppressSelected(), almSuppressSelectedGroup(),  
almSuppressSelectedTag(), almAckSelectedPriority()

### almSuppressSelectedTag() 函数

抑制显示当前以及将来要发生的任何特定报警：它们与一个或多个所选的报警具有相同的“标记名”，并且具有相同的供应器名、组名以及优先级范围。

#### 类别

报警

#### 语法

```
[Result=]almSuppressSelectedTag(ObjectName);
```

#### 参数

##### *ObjectName*

报警对象的名称。例如， AlmObj\_1。

#### 示例

```
almSuppressSelectedTag("AlmObj_1");
```

#### 另请参阅

almSuppressAll(), almSuppressGroup(), almSuppressTag(),  
almSuppressDisplay(), almSuppressSelectedAlarm(),  
almSuppressSelectedGroup(),  
almSuppressSelectedPriority(), almAckSelectedTag(),  
almUnSuppressAll()

## almSuppressRetain() 函数

抑制后续查询产生的所有报警。

类别

报警

语法

```
[Result=]almSuppressRetain(ObjectName,SuppressionRetainFlag);
```

参数

*ObjectName*

报警对象的名称。例如， AlmObj\_1。

*SuppressionRetainFlag*

任何离散或模拟标记、0 或非零值。如果为后续查询保留抑制信息，则为 TRUE；否则为 FALSE。

附注

如果标识为 0，则报警查询改变时，会删除抑制过滤器。

示例

```
almSuppressRetain("AlmObj_1", 0);
```

另请参阅

almSuppressAll(), almSuppressGroup(), almSuppressTag(),  
almSuppressDisplay(), almSuppressPriority(),  
almSuppressSelected(), almSuppressSelectedGroup(),  
almSuppressSelectedPriority(), almSuppressSelectedTag(),  
almUnSuppressAll()

### **.SuppressRetain 点域**

读取 / 写入 “分布式报警显示” 对象保持抑制的功能的状态。

#### **类别**

报警

#### **用法**

```
[ErrorNumber=]GetPropertyD(  
    "ObjectName.SuppressRetain", TagName);  
[ErrorNumber=] SetPropertyD(  
    "ObjectName.SuppressRetain", TagName);
```

#### **参数**

##### *ObjectName*

“分布式报警显示” 对象的名称。例如, *AlmObj\_1*。

##### *Tagname*

离散标记, 处理脚本时用于存放属性值。

#### **数据类型**

离散 (可读写)

#### **有效值**

0 = 关闭保持

1 = 打开保持

#### **示例**

以下语句使用 SupRtn 离散标记设置 “AlmObj\_1” 抑制保持器的状态。

```
SetPropertyD("AlmObj_1.SuppressRetain", SupRtn);
```

#### **另请参阅**

GetPropertyD(), SetProperty()

## 滚动报警显示

使用以下函数与点域垂直或水平滚动“分布式报警显示”对象中的报警列表。您也可以冻结显示对象。

- almMoveWindow() 函数
- .Freeze 点域
- .PrevPage 点域
- .NextPage 点域

### almMoveWindow() 函数

垂直或水平滚动“分布式报警显示”对象的报警列表。

类别  
报警

语法

[Result=] almMoveWindow ( *ObjectName*, *Option*, *Repeat* ) ;

参数

*ObjectName*

报警对象的名称。例如， AlmObj\_1。

*Option*

要执行的滚动操作的类型：

类型	描述
LineDn	下移一行。
LineUp	上移一行。
PageDn	下翻一页。
PageUp	上翻一页。
Top	滚动到列表顶部。
Bottom	滚动到列表底部。
PageRt	右移一页。
PageLf	左移一页。
Right	滚动到列表末尾（右侧）。
Left	滚动到列表开头（左侧）。

*Repeat*

应重复此操作的次数。

**示例**

```
almMoveWindow("AlmObj_1", "Bottom", 0);  
almMoveWindow("AlmObj_1", "LineDn", 3);  
almMoveWindow("AlmObj_1", "PageUp", 0);
```

**.Freeze 点域**

.Freeze 点域读取冻结状态，或冻结 / 撤销冻结 “分布式报警显示” 对象。

**类别**

报警

**用法**

```
[ErrorNumber=]GetPropertyD("ObjectName.Freeze",  
    TagName);  
[ErrorNumber=]SetPropertyD("ObjectName.Freeze",  
    TagName);
```

**参数****ObjectName**

报警对象的名称。例如， AlmObj\_1。

**TagName**

离散标记，处理函数时用于存放属性值。

**附注**

包含或更改 “分布式报警显示” 对象冻结状态的可读写点域。冻结报警显示对象时，显示的报警无法更新，也无法添加新的报警。冻结对报警是否闪烁没有影响。

**数据类型**

离散（可读写）

**有效值**

0 = 关闭冻结

1 = 打开冻结

**示例**

以下语句使用 AlmFreeze 离散标记设置 “AlmObj\_1” 的 Freeze 属性。

```
SetPropertyD("AlmObj_1.Freeze", AlmFreeze);
```

**另请参阅**

GetPropertyD(), SetPropertyD()



### .PrevPage 点域

将“分布式报警显示”对象向上滚动一页（一整屏的报警）。

**类别**

报警

**用法**

```
[ErrorNumber=] SetPropertyD ("ObjectName.PrevPage", 0)  
;
```

**参数**

*ObjectName*

报警对象的名称。例如， AlmObj\_1。

**附注**

设置此属性时，“分布式报警显示”对象显示上一页。显示上一页之后，变量自动设置为 1，除非已到达列表顶部。这种情况下，值保持为 0。

**数据类型**

离散（可读写）

**另请参阅**

GetPropertyD(), SetPropertyD(), .NextPage, .PageNum, .TotalPages

### .NextPage 点域

将“分布式报警显示”对象向下滚动一页（一整屏的报警）。

**类别**

报警

**用法**

```
[ErrorNumber=] SetPropertyD ("ObjectName.NextPage", 0)  
;
```

**参数**

*ObjectName*

报警对象的名称。例如， AlmObj\_1。

**附注**

设置此属性时，报警显示对象显示下一页。

**数据类型**

离散（可读写）

**另请参阅**

GetPropertyD(), SetPropertyD(), .PrevPage, .PageNum, .TotalPages

## 显示报警统计与计数

使用以下函数与点域显示当前“分布式报警显示”对象的有关统计信息。

- almShowStats() 函数
- .PageNum 点域
- .TotalPages 点域
- .NumAlarms 点域
- .ProvidersReq 点域
- .ProvidersRet 点域

### almShowStats() 函数

显示指定的“分布式报警显示”对象的**报警统计**对话框。

类别

报警

语法

```
[Result=]almShowStats(ObjectName) ;
```

参数

*ObjectName*

报警对象的名称。例如， AlmObj\_1。

示例

```
almShowStats( “AlmObj_1” );
```

### .PageNum 点域

包含报警对象中显示的当前页码。

#### 类别

报警

#### 语法

```
[ErrorNumber=]GetPropertyI("ObjectName.PageNum",  
    TagName);
```

#### 参数

*ObjectName*

“分布式报警显示”对象的名称。例如， *AlmObj\_1*。

*TagName*

整型标记，存放“分布式报警显示”对象中当前显示的页面的编号。

#### 附注

此只读点域返回指定的“分布式报警显示”对象中当前显示的页面的编号。

#### 数据类型

整型（只读）

#### 示例

以下语句将“分布式报警显示”对象 *AlmObj\_1* 中当前显示的页面的编号返回给 *AlarmPage* 整型标记：

```
GetPropertyI("AlmObj_1.PageNum", AlarmPage);
```

#### 另请参阅

*GetPropertyI()*, *.NextPage*, *.PrevPage*, *.TotalPages*

### **.TotalPages** 点域

包含“分布式报警显示”对象中的总页数。

#### **类别**

报警

#### **语法**

```
[ErrorNum=]GetPropertyI("ObjectName.TotalPages",  
    TagName);
```

#### **参数**

##### *ObjectName*

“分布式报警显示”对象的名称。例如，*AlmObj\_1*。

##### *TagName*

整型标记，检索指定的“分布式报警显示”对象中包含的总报警页面数。

#### **附注**

此点域返回指定的“分布式报警显示”对象中包含的总报警页面数。一个页面即任何给定时刻屏幕上的对象中显示的所有报警。

#### **数据类型**

整型（只读）

#### **示例**

以下语句将“分布式报警显示”对象 *AlmObj\_1* 中包含的总页数返回给 *AlmTotalPages* 整型标记：

```
GetPropertyI(  
    "AlmObj_1.TotalPages",AlmTotalPages);
```

#### **另请参阅**

*GetPropertyI()*, *NextPage*, *PrevPage*, *PageNum*

## .NumAlarms 点域

包含“分布式报警显示”对象中的报警数。

### 类别

报警

### 语法

```
[ErrorNum=]GetPropertyI("ObjectName.NumAlarms",  
    Tagname);
```

### 参数

*ObjectName*

“分布式报警显示”对象的名称。例如，*AlmObj\_1*。

*TagName*

整型标记，存放指定的“分布式报警显示”对象中注册的当前报警数。这不仅包括那些已显示的报警，还包括已注册的所有报警。

### 附注

此只读点域返回“分布式报警显示”对象中的总报警数。

### 数据类型

整型（只读）

### 示例

以下语句将“分布式报警显示”对象 *AlmObj\_1* 所使用的当前报警数返回给 *AlarmCount* 整型标记：

```
GetPropertyI("AlmObj_1.NumAlarms",AlarmCount);
```

### 另请参阅

[GetPropertyI\(\)](#)

### **.ProvidersReq 点域**

包含指定的“分布式报警显示”对象所使用的当前查询所需的报警供应器数。

#### **类别**

报警

#### **语法**

```
[ErrorNumber=]GetPropertyI(  
    "ObjectName.ProvidersReq", TagName);
```

#### **参数**

*ObjectName*

“分布式报警显示”对象的名称。例如，*AlmObj\_1*。

*TagName*

整型标记，存放指定的“分布式报警显示”对象中注册的当前报警供应器数。这不仅包括那些已显示的报警，还包括已注册的所有报警。

#### **数据类型**

整型（只读）

#### **示例**

以下语句返回“分布式报警显示”对象 "AlmObj\_1" 所使用的当前查询所需的报警供应器数。此值写入 **TotalProv** 整型标记：

```
GetPropertyI("AlmObj_1.ProvidersReq", TotalProv);
```

#### **另请参阅**

GetPropertyI(), .ProvidersRet

## **.ProvidersRet** 点域

包含指定的“分布式报警显示”对象所使用的当前查询返回的报警供应器数。

### **类别**

报警

### **用法**

```
[ErrorNumber=]GetPropertyI  
    ("ObjectName.ProvidersRet",TagName);
```

### **参数**

*ObjectName*

“分布式报警显示”对象的名称。例如，*AlmObj\_1*。

*TagName*

整型标记，存放已经成功将其报警返回给指定的“分布式报警显示”对象的报警供应器数。

### **附注**

此只读点域包含指定的“分布式报警显示”对象所使用的当前查询返回的报警供应器数。

### **数据类型**

整型（只读）

### **示例**

以下语句返回特定的报警供应器的数量，这些供应器已成功将其报警返回给“分布式报警显示”对象 *AlmObj\_1*。此值写入 *RetProv* 整型标记：

```
GetPropertyI("AlmObj_1.ProvidersRet",RetProv);
```

### **另请参阅**

`GetPropertyI()`, `.ProvidersReq`

## 错误描述

下表介绍各个错误号。如果返回此表中没有的某个编号，则该错误是未知错误。

错误号	描述
0	成功
-1	一般性错误
-2	可用内存不足
-3	属性为只读
-4	指定的项目已经存在
-5	未知对象名
-6	未知属性名



## 附录 B

# 从旧有的报警系统迁移

您可以迁移使用“标准报警系统”或 AlarmSuite 构建的应用程序。

## 从标准报警系统迁移到分布式报警系统

从“标准报警系统”迁移到“分布式报警系统”时，主 / 从应用程序中的所有“标准报警显示”都会迁移为“分布式报警显示”对象。

颜色、字体、表达式及报警查询等设置不迁移。新的“分布式报警显示”对象具有以下缺省查询，其中 `nodename` 是主节点的名称：

```
\\nodename\intouch!$system
```

确认与报警状态点域能够像过去一样继续使用。根据 I/O 标记是为 NetDDE 还是为 SuiteLink 而配置，有可能需要启用 NetDDE。不过，由于现在可以使用“分布式报警显示”对象来确认报警，因此您可能会决定不再需要使用单独的控件来发出确认。

## 迁移 AlarmSuite 数据库

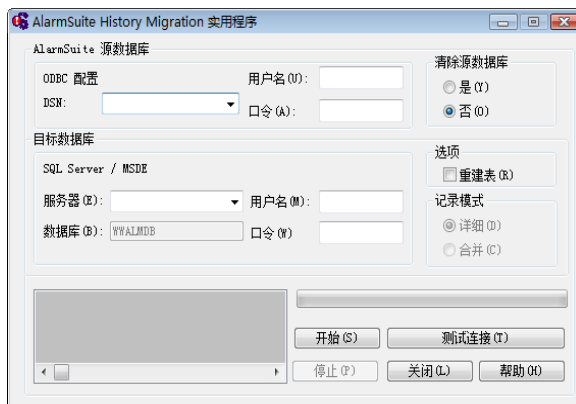
您可以使用“AlarmSuite History Migration 实用程序”将数据从 AlarmSuite 数据库迁移到 SQL Server 报警数据库。

迁移期间，源数据库或目标数据库都不应记录新的数据。如果在迁移期间进行记录，则可能导致失去数据完整性。

**重要** 在迁移期间，您可以选择在目标数据库中重建表。重建表将导致丢失目标数据库中现有的数据。

### 要迁移 AlarmSuite 数据

- 1 在 Windows 开始菜单上，指向程序，指向 Wonderware，指向 InTouch，然后单击 AlarmSuite History Migration。此时出现 AlarmSuite History Migration 实用程序对话框。



- 2 在 AlarmSuite 源数据库区域中，配置与 AlarmSuite 数据库的连接。执行以下操作：
  - a 在 DSN 列表中，单击 AlarmSuite 数据库的数据源名。
  - b 在用户名框中，输入该数据源名的用户名。
  - c 在口令框中，输入该数据源名的用户口令。
  - d 在清除源数据库区域中，单击是以便在数据迁移结束时清除源数据库。创建表是事务处理的一部分，但创建数据库不是。如果选择是，则从源数据库中迁移数据之后提交清除事务。  
单击否以便在完成数据迁移之后不清除源数据库。
- 3 在目标数据库区域中，配置与“分布式报警系统”数据库的连接。执行以下操作：
  - a 在服务器列表中，单击要使用的数据库服务器的名称。  
只读的数据库框显示数据库名。缺省值为 WWALMDB。
  - b 在用户名框中，输入 SQL Server 用户名。
  - c 在口令框中，输入与该 SQL Server 用户帐户关联的口令。

- d 选择**重建表**复选框，以便在目标数据库中重建表。如果重建表，则会删除现有的信息。如果不选择此选项，数据会附加到现有的表中。
  - e 在**记录模式**区域中，单击**详细**以便仅恢复详细表，或单击**合并**以便仅恢复合并表。
- 4 单击**测试连接**以测试与源数据库及目标数据库的连接。
  - 5 单击**开始**，以便开始将数据从源数据库迁移到目标数据库中。
  - 6 单击**停止**以中止正在进行的迁移过程。这会回滚事务处理。（单击**关闭**相当于单击**停止**）。

---

**备注** 创建表是事务处理的一部分，但创建数据库不是。如果将**清除源数据库**选项设置为**是**，则从源数据库中迁移数据之后会提交清除事务。

---

- 7 迁移完成之后，单击**关闭**。



# 索引

## 符号

\$NewAlarm 系统标记 121

\$System 系统标记 121

## A

AboutBox() 方法 92, 204, 323, 345

.Ack 点域 103

Ack() 函数 109

AckAll() 方法 79

AckAlmBackColor 属性 289

AckAlmBackColorRange1 属性 289

AckAlmBackColorRange2 属性 290

AckAlmBackColorRange3 属性 290

AckAlmBackColorRange4 属性 291

AckAlmForeColor 属性 291

AckAlmForeColorRange1 属性 291

AckAlmForeColorRange2 属性 292

AckAlmForeColorRange3 属性 292

AckAlmForeColorRange4 属性 293

.AckDev 点域 107

.AckDsc 点域 106

AckGroup() 方法 81

AckPriority() 方法 81

.AckROC 点域 108

AckRtnBackColor 属性 293

AckRtnForeColor 属性 294

AckSelected() 方法 79

AckSelectedGroup() 方法 80

AckSelectedPriority() 方法 80

AckSelectedTag() 方法 80

AckTag() 方法 82

.AckValue 点域 105

AckVisible() 方法 79

ActiveX 方法

Alarm DB View 控件 318

Alarm Pareto 控件 342

Alarm Tree Viewer 控件 204

Alarm Viewer 控件 78

ActiveX 事件

Alarm DB View 控件 324

Alarm Pareto 控件 346

Alarm Tree Viewer 控件 212

ActiveX 属性

Alarm DB View 控件 289

Alarm Pareto 控件 339

Alarm Tree Viewer 控件 202

Alarm Viewer 控件 71

Alarm DB Logger

AlarmCounter 存储过程 265

AlarmSuite 报警日志视图 263

- 报警存储过程 264
- 报警记录功能 251, 256
- 报警历史视图 257, 259
- 报警事件历史视图 262
- 报警数据库视图 256
- 存储过程 264
- EventCounter 存储过程 266
- 关于 249
- 记录间隔 254
- SQL Server 帐户 250
- 事件历史视图 259, 260
- 使用 250
- 数据库连接 251
- Windows 服务 255
- 要记录的报警 252
- Alarm DB Logger 报警数据库视图 256
- Alarm DB Logger 的 SQL Server 帐户 250
- Alarm DB View 控件
  - ActiveX 方法 318
  - ActiveX 事件 324
  - ActiveX 属性 289
  - 报警或事件数据 272
  - 查看记录的报警 267
  - 查询过滤器 284
  - 错误处理 324
  - 访问过滤器收藏夹 323
  - 给记录排序 288, 322
  - 过滤器收藏夹 281
  - 过滤器准则 282
  - 检索报警信息 321
  - 检索记录 319
  - 列 273
  - 排序顺序 287
  - 配置 268
  - 时间段 279
  - 时间格式 277
  - 时区 277, 280
  - 数据库连接 269, 318
  - 网格 270
  - 显示上下文菜单 323
  - 显示“关于”框 323
  - 颜色 286
  - 运行时功能 276
  - 在运行时使用 288
  - 状态栏 288
  - 状态列的值 285
  - 字体 271
  - 组合多个列 284
- Alarm DB View 控件的时区 277, 280
- .Alarm 点域 122
- Alarm Pareto 控件
  - ActiveX 方法 342
  - ActiveX 事件 346
  - ActiveX 属性 339
  - 错误处理 346
  - 分析结果的显示 337
  - 复制或移动查询过滤器 336
  - 关于 325
  - 过滤器收藏夹 333
  - 过滤器准则 335
  - 检索记录 342
  - 检索特定巴累托条的有关信息 343
  - 配置 325
  - 时段 332
  - 数据库连接 326, 342
  - 外观与颜色 327
  - 显示“关于”框 345
  - 选择报警或事件数据 331
  - 要分析的报警 331
  - 运行时功能 330
  - 在运行时使用 338
  - 状态栏 338
  - 字体 329
  - 组合多个列 336
- Alarm Pareto 控件的分析结果 337
- Alarm Printer 查询
  - 启动 226
  - 停止 226
- Alarm Printer 实例
  - 启动 226
  - 停止 226
- Alarm Tree Viewer 控件
  - ActiveX 方法 204
  - ActiveX 事件 212
  - ActiveX 属性 202

- 查看报警层次结构 191
- 查询收藏夹 198, 201
- 创建查询字符串 210
- 错误处理 212
- 冻结目录树 210
- 供应器与组 196
- 检索信息 204, 205
- 排序顺序 199
- 配置 192
- 刷新 195
- 外观与颜色 192
- 运行查询 211
- 运行时功能 195
- 在运行时使用 200
- 状态栏 201
- 字体 194
- Alarm Tree Viewer 控件查询字符串 210
- Alarm Viewer 控件
  - ActiveX 方法 78
  - ActiveX 事件 96
  - ActiveX 属性 71
  - 查看当前报警 51
  - 查询收藏夹 62, 68
  - 错误处理 96
  - 给报警记录排序 91
  - 检测到新报警时运行脚本 97
  - 检索有关报警的信息 87
  - 列 56
  - 排序顺序 66
  - 配置 52
  - 确认报警 78
  - 时间格式 64
  - 网格 52
  - 显示上下文菜单 96
  - 显示统计信息 92
  - 显示“关于”框 92
  - 选择特定的报警 93
  - 颜色 63, 77
  - 要显示的报警 61
  - 移动与冻结显示 90
  - 抑制报警 82
  - 运行查询 87
  - 运行时功能 59
  - 在运行时使用 67
  - 状态栏 68
  - 字体 56
- Alarm Viewer 控件的统计信息 92
- .AlarmAccess 点域 427
- .AlarmAckModel 点域 102
- .AlarmClass 点域 421
- .AlarmComment 点域 168, 431
- AlarmCounter 存储过程 265
- .AlarmDate 点域 418
- .AlarmDev 点域 125
- .AlarmDevCount 点域 187
- .AlarmDevDeadband 点域 167
- .AlarmDevUnAckCount 点域 188
- .AlarmDisabled 点域 141
- .AlarmDsc 点域 124
- .AlarmDscCount 点域 185
- .AlarmDscDisabled 点域 151
- .AlarmDscEnabled 点域 150
- .AlarmDscInhibitor 点域 172
- .AlarmDscUnAckCount 点域 186
- .AlarmEnabled 点域 140
- .AlarmGroup 点域 435
- .AlarmGroupSel 点域 426
- .AlarmHiDisabled 点域 147
- .AlarmHiEnabled 点域 146
- .AlarmHiHiDisabled 点域 149
- .AlarmHiHiEnabled 点域 148
- .AlarmHiHiInhibitor 点域 176
- .AlarmHiInhibitor 点域 175
- .AlarmLimit 点域 424
- .AlarmLoDisabled 点域 145
- .AlarmLoEnabled 点域 144
- .AlarmLoInhibitor 点域 174
- .AlarmLoLoDisabled 点域 143
- .AlarmLoLoEnabled 点域 142
- .AlarmLoLoInhibitor 点域 173
- .AlarmMajDevDisabled 点域 155
- .AlarmMajDevEnabled 点域 154
- .AlarmMajDevInhibitor 点域 178
- .AlarmMinDevDisabled 点域 153
- .AlarmMinDevEnabled 点域 152
- .AlarmMinDevInhibitor 点域 177

- .AlarmName 点域 419
- .AlarmOprName 点域 429
- .AlarmOprNode 点域 430
- .AlarmPri 点域 425
- .AlarmProv 点域 428
- .AlarmROC 点域 126
- .AlarmROCCount 点域 189
- .AlarmROCDisabled 点域 157
- .AlarmROCEnabled 点域 156
- .AlarmROCIhibitor 点域 179
- .AlarmROCUnAckCount 点域 190
- .AlarmState 点域 423
- AlarmSuite 报警日志视图 263
- .AlarmTime 点域 417
- .AlarmTotalCount 点域 181
- .AlarmType 点域 422
- .AlarmValDeadband 点域 166
- .AlarmValue 点域 420
- .AlarmValueCount 点域 183
- .AlarmValueUnAckCount 点域 184
- .AlarmUnAckCount 点域 182
- .AlarmUserDefNumX 点域 169
- .AlarmUserDefStr 点域 170
- almAckAll() 函数 403
- almAckDisplay() 函数 403
- almAckGroup() 函数 404
- almAckPriority() 函数 405
- almAckRecent() 函数 405
- almAckSelect() 函数 407
- almAckSelectedGroup() 函数 408
- almAckSelectedPriority() 函数 409
- almAckSelectedTag() 函数 410
- almAckTag() 函数 406
- almDefQuery() 函数 432
- almMoveWindow() 函数 455
- almQuery() 函数 433
- AlmRtnBackColor 属性 294
- AlmRtnForeColor 属性 294
- almSelectAll() 函数 411
- almSelectGroup() 函数 413
- almSelectionCount() 函数 412
- almSelectItem() 函数 413
- almSelectPriority() 函数 414
- almSelectTag() 函数 415
- almSetQueryByName() 函数 434
- almShowStats() 函数 458
- almSuppressAll() 函数 444
- almSuppressDisplay() 函数 445
- almSuppressGroup() 函数 446
- almSuppressPriority() 函数 447
- almSuppressRetain() 函数 453
- almSuppressSelected() 函数 449
- almSuppressSelectedGroup() 函数 450
- almSuppressSelectedPriority() 函数 451
- almSuppressSelectedTag() 函数 452
- almSuppressTag() 函数 448
- almUnselectAll() 函数 412
- almUnsuppressAll() 函数 445
- ApplyDefaultQuery() 方法 88
- ApplyQuery() 方法 88
- APUFindAlarmGroupInstance() 函数 236
- APUFindFileInstance() 函数 237
- APUFindPrinterInstance() 函数 238
- APUGetAlarmGroupText() 函数 229
- APUGetConfigurationFilePath() 函数 232
- APUGetInstanceCount() 函数 239
- APUGetPrinterJobCount() 函数 233
- APUGetPrinterName() 函数 241
- APUGetPrinterStatus() 函数 242
- APUGetQueryAlarmState() 函数 234
- APUGetQueryFromPriority() 函数 230
- APUGetQueryProcessingState() 函数 235
- APUGetQueryToPriority() 函数 231
- APUIsInstanceUsed() 函数 240
- APUSetAlarmGroupText() 函数 243
- APUSetQueryAlarmState() 函数 244
- APUSetQueryFromPriority() 函数 245
- APUSetQueryToPriority() 函数 246
- APUSetTimeoutValues() 函数 247
- APUStartInstance() 函数 226
- APUStartQuery() 函数 227
- APUStopInstance() 函数 228
- APUStopQuery() 函数 228
- APUTranslateErrorCode() 函数 248



AutoConnect 属性 295

## B

### 报警

- 变化率报警 24
- 打印 215
- 分析 331
- 供应器 26
- 关于 17, 18
- 记录 252
- 接收器 26
- 禁用 39
- 类型 22
- 离散报警 22
- 模拟报警 23
- 配置 31
- 偏差报警 23
- 启用或禁用 140
- 确认 19, 103
- 确认模型 100
- 冗余 361
- 冗余配置的同步 377
- 审核跟踪 381
- 统计 180
- 维护数据库 347
- 显示 388
- 优先级 18
- 约束 39
- 值报警 23
- 子状态 19
- 组 19
- 报警标记的约束标记 172
- 报警层次结构
  - Alarm Tree Viewer 控件 191
  - 定义 31
- 报警查询 47
- 报警打印与记录 213
- 报警供应器 26
- 报警接收器 26, 27
- 报警历史视图 257, 259
- 报警审核跟踪 381
- 报警事件历史视图 262
- 报警实例与用户自定义信息 169
- 报警数据库

- Alarm DB Logger 251, 256
- 存储过程 264
- 归档清除的数据 350
- 恢复 357, 360
- 计划自动清除 356
- 清除或归档设置 348
- 日志文件设置 352
- 手工归档数据库 354
- 数据库连接 348, 358
- 维护 347
- 要恢复的文件 359
- 要清除的数据 349
- 报警数据库归档 354
- 报警数据库记录间隔 254
- 报警数据库维护 347
- 报警条件
  - 报警组 120
  - 标记 120
- 报警限
  - 标记 134
  - 更改 158
- 报警抑制 29
- 报警优先级 18
- 报警与事件的全局设置 41
- 报警注释 112
  - 分布式报警显示对象 389
  - 更改标记的 168
- 报警子状态 19
- 报警组 19
  - 报警条件 120
  - 创建 32
  - 删除 34
  - 修改 34
- 报警组的约束标记 172
- 保留报警记录功能 44
- 保留记录功能 44
- 变化率报警
  - 关于 24
  - 配置 38
  - 启用或禁用 156
  - 确认 108
- 标记
  - 报警条件 34, 120
  - 报警限 134

更改报警注释 168

事件属性 41

标准报警系统 465

## C

CheckElementMembership() 方法 205

ColorPriorityRange1 属性 295

ColorPriorityRange2 属性 296

ColorPriorityRange3 属性 296

ColumnResize 属性 296

Connect() 方法 318, 342

ConnectStatus 属性 297

CustomMessage 属性 297

查询过滤器

Alarm DB View 控件 284

Alarm Pareto 控件 336

查询收藏夹

Alarm Tree Viewer 控件 198, 201

Alarm Viewer 控件 62, 68

分布式报警显示对象 399

创建报警组列表文件 45

从 WindowViewer 中确认报警 110

存储过程 264

AlarmCounter 存储过程 265

报警数据库 264

EventCounter 存储过程 266

错误

Alarm DB View 控件 324

Alarm Pareto 控件 346

Alarm Tree Viewer 控件 212

打印报警 248

分布式报警显示对象 464

## D

DatabaseName 属性 297

.DevTarget 点域 164

Disconnect() 方法 318

DisplayedTimeZone 属性 298

DisplayMode 属性 298

Duration 属性 299

打印报警

保存与加载配置文件 222

报警查询信息 229, 243

报警打印与记录 213

错误 248

打印报警 213, 223

打印机设置 214

打印与文件输出的格式 217

记录信息 241

将报警记录到文件 220, 224

启动 Alarm Printer 225

启动 Alarm Printer 查询 226

启动 Alarm Printer 实例 226

实例信息 236

使用脚本控制 225

停止 Alarm Printer 查询 226

停止 Alarm Printer 实例 226

要打印的报警 215

打印报警的报警查询信息 229, 243

打印报警的格式 217

打印报警的记录信息 241

打印报警的配置文件 222

打印报警的日志文件 220

打印报警的实例信息 236

冻结

Alarm Tree Viewer 控件 210

Alarm Viewer 控件 90

## E

EndTime 属性 300

EventBackColor 属性 300

EventCounter 存储过程 266

EventForeColor 属性 300

## F

FilterFavoritesFile 属性 301

FilterMenu 属性 301

FilterName 属性 301

.Freeze 点域 456

Freeze() 方法 210

FreezeDisplay() 方法 91

FromPriority 属性 302

分布式报警系统

关于 25

数据储存 30

分布式报警系统数据储存 30

分布式报警显示对象

报警统计与计数 458

- 报警注释 389
- 查询收藏夹 399
- 创建 385
- 错误 464
- 多重选择 397
- 关于 383
- 滚动 455
- 获取或设置属性 402
- 检查当前查询属性 435
- 监视本地报警 396
- 检索报警记录 416
- 快捷菜单 398
- 列 392
- 排序顺序 392
- 配置 385
- 确认报警 402
- 时间格式 390
- 使用脚本进行控制 402
- 使用准则 384
- 网格外观 386
- 显示类型 395
- 显示列 396
- 选择报警 411
- 颜色 394, 397
- 要显示的报警 388
- 抑制报警 443
- 运行时功能 387
- 在运行时使用 396
- 状态栏 397
- 字体 391
- 分布式报警显示对象的报警统计与计数 458
- 分布式报警显示对象的快捷菜单 398
- 分布式报警显示对象的显示类型 395
- 分布式报警组列表 28
- GetItemCount() 方法 344
- GetItemEventType() 方法 345
- GetItemProviderName() 方法 345
- GetItemTotalTime() 方法 344
- GetLastError() 方法 324
- GetNext() 方法 320
- GetPrevious() 方法 319
- GetSelectedElementCount() 方法 206
- GetSelectedElementName() 方法 206
- GetSelectedElementPath() 方法 207
- GetSelectedItem() 方法 321
- GetSubElementCount() 方法 207
- GetSubElementName() 方法 208
- GetSubElementPath() 方法 209
- GroupExactMatch 属性 302
- GroupName 属性 303
- 给记录排序
  - Alarm DB View 控件 288, 322
  - Alarm Viewer 控件 91
- 更改报警死区 166
- 供应器
  - Alarm Tree Viewer 控件 196
  - 报警 26
- 关于框
  - Alarm DB View 控件 323
  - Alarm Pareto 控件 345
  - Alarm Viewer 控件 92
- 归档清除的数据 350
- 滚动“分布式报警显示”对象 455
- 过滤器收藏夹
  - Alarm DB View 控件 281, 323
  - Alarm Pareto 控件 333
- 过滤器准则
  - Alarm DB View 控件 282
  - Alarm Pareto 控件 335

## G

- GetAlarmQueryFromSelection() 方法 210
- GetElementCount() 方法 204, 205
- GetElementName() 方法 206
- GetElementPath() 方法 206
- GetItem() 方法 87, 321
- GetItemAlarmName() 方法 343
- GetItemAlarmType() 方法 344

## H

- .HiHiLimit 点域 161
- .HiHiSet 点域 136
- .HiHiStatus 点域 130
- .HiLimit 点域 160
- .HiSet 点域 135
- .HiStatus 点域 129
- 缓冲区大小 42

恢复

报警数据库 357, 360

报警数据库文件 359

获取或设置“分布式报警显示”对象的属性 402

## J

记录报警

打印报警 224

文件设置 352

基于事件的报警模型 101

检查“分布式报警显示”对象的查询属性 435

检查“分布式报警显示”对象的更新 441

监视本地报警的“分布式报警显示”对象 396

检索记录

Alarm DB View 控件 319

Alarm Pareto 控件 342

分布式报警显示对象 416

检索信息

Alarm DB View 控件 321

Alarm Pareto 控件 343

Alarm Tree Viewer 控件 204

Alarm Viewer 控件 87

脚本

Alarm Viewer 控件 97

打印报警 225

分布式报警显示对象 402

确认报警 109

禁用 29

报警 140

变化率报警 156

副偏差报警 152

High 报警 146

HiHi 报警 148

LoLo 报警 142

Low 报警 144

离散报警 150

主偏差报警 154

禁用或启用 HiHi 报警 148

## K

扩展的摘要报警

模型 100

使用 101

扩展的摘要报警模型 100

## L

.ListChanged 点域 441

.LoLimit 点域 159

.LoLoLimit 点域 158

.LoLoSet 点域 134

.LoLoStatus 点域 128

.LoSet 点域 135

.LoStatus 点域 127

离散报警

报警 22

配置 35

启用或禁用 150

确认报警 106

历史报警 29

联系技术支持 16

列

Alarm DB View 控件 273, 284, 285

Alarm Pareto 控件 336

Alarm Viewer 控件 56

分布式报警显示对象 392, 396

## M

.MajorDevPct 点域 163

.MajorDevSet 点域 138

.MajorDevStatus 点域 132

MaxRecords 属性 303

.MinorDevPct 点域 162

.MinorDevSet 点域 137

.MinorDevStatus 点域 131

MoveWindow() 方法 90

模拟报警 23

## N

.NextPage 点域 457

.Normal 点域 123

.NumAlarms 点域 461

## P

- .PageNum 点域 459
- Password 属性 303
- .PendingUpdates 点域 442
- .PrevPage 点域 457
- .PriFrom 点域 439
- PrimarySort 属性 304
- .PriTo 点域 440
- ProviderExactMatch 属性 304
- ProviderName 属性 305
- .ProvidersReq 点域 462
- .ProvidersRet 点域 463
- 排序顺序
  - Alarm DB View 控件 287
  - Alarm Tree Viewer 控件 199
  - 分布式报警显示对象 392
  - 配置 Alarm Viewer 控件 66
- 配置 ActiveX 控件的颜色 77
- 配置报警缓冲区大小 42
- 配置报警数据库的清除或归档设置 348
- 偏差报警
  - 报警 23
  - 配置 37
  - 确认报警 107

## Q

- 启动 Alarm Printer 225
- .QueryState 点域 437
- QueryTimeZoneName 属性 305
- .QueryType 点域 436
- 启用
  - 报警 140
  - 变化率报警 156
  - 副偏差报警 152
  - High 报警 146
  - HiHi 报警 148
  - LoLo 报警 142
  - Low 报警 144
  - 离散报警 150
  - 事件 42
  - 主偏差报警 154
- 启用或禁用 High 报警 146
- 启用或禁用 LoLo 报警 142

- 启用或禁用 Low 报警 144
- 启用或禁用副偏差报警 152
- 启用或禁用主偏差报警 154
- 迁移
  - AlarmSuite 数据库 466
  - 标准报警系统 465
  - 旧有的报警系统 465
- 迁移 AlarmSuite 数据库 466
- 迁移旧有的报警系统 465
- 清除
  - 报警数据库 349
  - 自动清除计划 356
- 确认
  - 报警 19
  - 自动 110
- 确认报警 99
  - Alarm Viewer 控件 78
  - 报警或报警组 103
  - 变化率报警 108
  - 从 WindowViewer 中 110
  - 分布式报警显示对象 402
  - 脚本函数 109
  - 离散报警 106
  - 偏差报警 107
  - 冗余配置的同步 377
  - 使用点域 103
  - 值报警 105
  - 自动确认 110
- 确认模型
  - 报警 100
  - 条件 100
  - 在运行时检查 102
- 确认注释 112

## R

- Refresh() 方法 320, 342
- RefreshMenu 属性 305
- Requery() 方法 88
- Reset() 方法 322
- ResetMenu 属性 306
- .ROCPct 点域 165
- .ROCSet 点域 139
- .ROCStatus 点域 133
- RowCount 属性 306

RowSelection 属性 306

冗余

报警 361

报警同步 377

备注 379

创建热备份对 364

创建映射文件 368

导入映射文件 371

关于 361

理解热备份 362

配置热备份对 363

确认同步 377

热备份对示例 373

设置报警关键码域 366

映射文件导入疑难排解 372

冗余配置的关键码域 366

冗余配置的热备份 362

冗余配置的热备份对 363, 364

冗余配置的热备份对示例 373

冗余配置的映射文件 368, 371

## S

SecondarySort 属性 307

SelectAll() 方法 95

SelectGroup() 方法 93

SelectItem() 方法 95

SelectPriority() 方法 94

SelectQuery() 方法 319, 343

SelectTag() 方法 94

ServerName 属性 307

SetQueryByName() 方法 89, 211

SetQueryByString() 方法 211

SetSort() 方法 92

ShowContext() 方法 96, 323

ShowFetch 属性 307

ShowFilter() 方法 323

ShowGrid 属性 308

ShowHeading 属性 308

ShowMessage 属性 308

ShowQueryFavorites() 方法 87

ShowSort() 方法 91, 322

ShowStatistics() 方法 92

ShowStatusBar 属性 309

ShowSuppression() 方法 83

SilentMode 属性 309

SortMenu 属性 309

SortOnCol() 方法 322

SortOrder 属性 310

SpecificTime 属性 310

StartTime 属性 311

.Successful 点域 438

SuppressAll() 方法 83

SuppressGroup() 方法 85

SuppressPriority() 方法 86

.SuppressRetain 点域 454

SuppressSelected() 方法 83

SuppressSelectedGroup() 方法 84

SuppressSelectedPriority() 方法 84

SuppressSelectedTag() 方法 84

SuppressTag() 方法 86

SuppressVisible() 方法 83

上下文菜单

Alarm DB View 控件 323

Alarm Viewer 控件 96

时段

Alarm Pareto 控件 332

事件

关于 17, 21

启用 42

时间段

Alarm DB View 控件 279

时间格式

Alarm DB View 控件 277

Alarm Viewer 控件 64

分布式报警显示对象 390

事件历史视图 259, 260

视图

AlarmSuite 报警日志视图 263

报警历史视图 257, 259

报警事件历史视图 262

事件历史视图 259, 260

使用 Alarm DB View 控件查看报警或事件数据 272

使用 Alarm DB View 控件查看记录的报警 267

使用 Alarm Viewer 控件查看当前报警 51

使用点域确认报警 103



## 数据库连接

- Alarm DB Logger 251
- Alarm DB View 控件 269, 318
- Alarm Pareto 控件 326, 342
- 报警数据库 348, 358
- 刷新 Alarm Tree Viewer 控件 195

## T

- Time 属性 311
- ToPriority 属性 311
- .TotalPages 点域 460
- TotalRowCount 属性 312
- 条件确认报警模型 100
- 统计活动的或未确认的报警 180

## U

- .UnAck 点域 104
- UnAckAlmBackColor 属性 312
- UnAckAlmBackColorRange1 属性 313
- UnAckAlmBackColorRange2 属性 313
- UnAckAlmBackColorRange3 属性 314
- UnAckAlmBackColorRange4 属性 314
- UnAckAlmForeColor 属性 315
- UnAckAlmForeColorRange1 属性 315
- UnAckAlmForeColorRange2 属性 316
- UnAckAlmForeColorRange3 属性 316
- UnAckAlmForeColorRange4 属性 317
- UnAckOrAlarmDuration 属性 317
- UnSelectAll() 方法 95
- UnSuppressAll() 方法 84
- UserID 属性 317

## W

- Windows 服务 255
- 网格
  - Alarm DB View 控件 270
  - Alarm Viewer 控件 52
  - 分布式报警显示对象 386
- 为单独的标记设置事件属性 41
- 文档惯例 15

## Y

### 颜色

- ActiveX 控件 77
- Alarm DB View 控件 286
- Alarm Pareto 控件 327
- Alarm Tree Viewer 控件 192
- Alarm Viewer 控件 63
- 分布式报警显示对象 394, 397
- 抑制报警 29
  - Alarm Viewer 控件 82
  - 分布式报警显示对象 443
- 映射文件导入疑难排解 372
- 用户自定义信息与报警实例 169
- 用于打印报警的打印机设置 214
- 约束报警 39

### 约束标记

- 报警标记的 172
- 报警组的 172

### 运行查询

- Alarm Tree Viewer 控件 211
- Alarm Viewer 控件 87

### 运行时

- Alarm DB View 控件 288
- Alarm Pareto 控件 338
- Alarm Tree Viewer 控件 200
- Alarm Viewer 控件 67
- 报警属性 113
- 分布式报警显示对象 396
- 检查确认模型 102

### 运行时功能

- Alarm DB View 控件 276
- Alarm Pareto 控件 330
- Alarm Tree Viewer 控件 195
- Alarm Viewer 控件 59
- 分布式报警显示对象 387

## Z

- 在运行时控制报警属性 113
- 摘要报警 29
- 值报警
  - 报警 23
  - 配置 36
  - 确认报警 105
- 终端服务 30

终端服务报警支持 30

状态栏

Alarm DB View 控件 288

Alarm Pareto 控件 338

Alarm Tree Viewer 控件 201

Alarm Viewer 控件 68

分布式报警显示对象 397

字体

Alarm DB View 控件 271

Alarm Pareto 控件 329

Alarm Tree Viewer 控件 194

Alarm Viewer 控件 56

分布式报警显示对象 391