

Ecava DX

用户手册

(中文全译本)

对应软件版本：v16.1（16.1.1006.2）

全手册共 5 部分 42 章

江苏江蛛物联网技术研究有限公司

目 录

安装 DX	17
Windows	17
所需条件	17
开始安装	17
欢迎	18
接受许可协议	18
选择安装选项	18
安装	19
完成	19
DX Edit 打开	19
Linux	20
所需条件	20
安装	20
从另一台计算机访问消息代理	22
在 WSL（适用于 Linux 的 Windows 子系统）中	22
打开 DX Edit	23
重新安装相同版本	24
下一步	25
出现问题时	25
再次运行安装程序时显示“修复”和“卸载”	25
“已安装更新版本的 Ecava DX”	25
“Windows 已保护你的电脑”	25
安装程序要求输入管理员密码	26
DX Edit 没有打开，或作为文本打开	26

DX Edit 提示无法连接消息代理	26
DEM0 未运行	26
DEM0 不在你看到的“文档”文件夹中	26
安装被取消或中途失败	26
安装程序要求重启 Windows	26
升级后你的项目处于停止状态	27
apt 提示“无法定位软件包”	27
apt 等待锁定	27
apt 提示“以 root 身份执行非沙盒下载”	27
无网络时安装失败	27
没有“wrote dx.conf”这一行	27
“DX 引擎尚未启动”	27
另一台计算机上的 DX Edit 无法连接	28
在 Linux 上检查 DX 引擎和消息代理	28
在多台计算机上安装	28
卸载	28
文件位置	29
DX Run	30
一个后端服务所有项目	30
启动你的第一个项目	30
选择 Web 端口	31
打开 DX Edit	31
DX Run 在 Linux 上的文件位置	31
DX Projects 文件夹之外的项目	31
从命令行管理 DX Run	32

DX Edit	33
入门	33
打开 DX Edit	33
查看一个正在运行的项目	34
创建你的项目	35
添加模块	36
启动项目	39
连接你的设备	40
添加要读取的值	41
应用并查看数值	43
在趋势图上查看你的值	44
添加报警	45
你现在拥有的	47
下一步	47
当出现异常时	48
设备不应答	48
DX Edit 无法连接代理	50
链接询问“是否连接到另一个代理？”	51
树显示“等待模块索引”	51
Watch 标签缺失，或什么都不显示	51
报警从不出现	52
添加模块显示“无可用的模块”	52
树中缺少项目	52
Start 呈灰色	52
圆点为琥珀色	52

单元格以橙色框出	52
Apply 被拒绝	52
“文件中也已更改”	53
重命名铅笔呈灰色	53
“请关闭其他程序.....”	53
设置文件上的琥珀色断链标志	53
一段时间后一切都停止了	53
某个模块显示红色的 Stopped	53
网页不显示值	53
网页不在你预期的地址	53
每次项目启动都打开一个浏览器标签	54
我的更改不见了	54
值看起来错误或错位	54
连接到服务器	54
启用模块以供编辑（DX Run）	55
界面	55
书签和共享链接	56
每个按钮的作用	57
模块有两个部分	58
管理项目	60
项目页面	63
网页	65
容器页面	66
添加模块	67
示例模块	69

重命名模块	69
重命名项目	70
打开模块	70
编辑	70
你的更改会被保留	72
当文件同时被更改	73
应用	74
历史	74
启动	74
组配置	75
嵌入式界面	75
刷新和重新加载	75
通用配置	76
配置	76
通用状态	82
运行时	83
配置重新加载	84
存储转发	84
冗余	85
Ecava IGX SCADA 服务器	85
示例	85
驱动	87
Bacnet/IP	87
Beckhoff ADS	88
Modbus Master	89
OPC UA Client	89

Profinet	90
EthernetIP	90
SNMP	92
Hostlink	92
Memobus	94
DX Manager	95
配置	95
CSV	97
示例	98
Web 服务器	100
配置	100
默认页面	101
服务地址	102
浏览器缓存	102
菜单	103
HTTPS	104
示例	104
登录	106
登录	106
双因素	107
设置安全图片（首次登录）	107
你的账户	107
当登录页面无法连接服务时	108
用户角色管理器	109
设置管理员前不会有任何安全措施	109
设置首位管理员	109

用户类型	110
设置用户	111
移除用户	113
示例	113
代理	114
登录选项	115
目录（LDAP）登录	115
脚本引擎	117
配置	117
CSV	118
按标签条件触发脚本	119
脚本文件	119
内联表达式	120
字符串格式化模板	121
示例	121
报警与审计	123
配置	123
CSV	124
示例	126
数据转发	128
配置	128
CSV	130
示例——文本文件	131
示例——邮件	132
MQTT 桥接	134

配置	134
CSV	136
示例	137
REST API 客户端	139
语法	139
匹配规则：仅匹配 \${token} 形式	139
嵌套占位符	140
JSON 数组上下文	140
缓存与重载	140
端到端示例：IGX 报警通知	140
目标	140
文件	140
执行过程	141
限制	142
网络扫描器	143
配置	143
协议检测	143
CSV	144
示例	144
历史记录器	147
配置	147
CSV	151
时间戳处理	151
文件结构	152
示例	153

历史查询	155
配置	155
数据存储结构	156
CSV	157
示例	157
历史聚合器	159
配置	159
CSV	160
存储转发数据	161
报表聚合	161
示例	162
示例：每天到达一次的数据	163
历史到 SQL	166
配置	166
数据文件夹结构	168
数据库模式	168
CSV	169
示例	169
SQL 记录器	171
配置	171
CSV	173
数据库模式	173
支持的数据库	174
示例	175
预 SQL	177
配置	177

CSV	178
参数化查询	178
示例	179
历史模拟器	181
配置	181
日志文件格式	182
CSV	183
示例	183
Modbus 主站	185
配置	185
连接类型	186
任务参数	186
串口参数	187
CSV	188
示例	188
Modbus 从站	191
配置	191
任务参数	191
CSV	192
示例	193
OPC UA 客户端	195
配置	195
自动浏览	197
CSV	197
示例	198
OPC UA 服务器	200

配置	200
安全	201
CSV	201
示例	202
OPC DA 客户端	204
配置	204
CSV	205
示例	205
BACnet/IP	208
配置	208
MS/TP（串口）	210
自动浏览	211
CSV	212
命令优先级	213
示例	216
MS/TP 示例	217
使用 Yabe 模拟器测试	218
启动模拟器	218
用 Yabe 发现设备	219
将发现的对象映射到 CSV	220
用 dxbacnet 获取模拟器	220
验证数据流	221
测试命令优先级	221
DNP3	223
配置	223
CSV	224

示例	225
IEC 60870-5-104	227
配置	227
自动浏览	229
CSV	230
地址格式	230
监视地址（读）	230
命令地址（写）	231
示例	232
IEC 61850	234
配置	234
自动发现	235
CSV	235
示例	236
Profinet	238
配置	238
任务参数	238
CSV	239
地址格式	240
示例	240
EtherNet/IP	243
配置	243
CSV	244
示例	245
Beckhoff ADS	247
配置	247

CSV	248
支持的数据类型	248
示例	249
Omron FINS	251
配置	251
连接类型	251
任务参数	252
FINS 头参数	253
CSV	253
示例	254
Omron Hostlink	257
配置	257
连接类型	257
任务参数	258
串口参数	259
CSV	260
示例	261
Mitsubishi MELSEC	263
配置	263
任务参数	263
CSV	264
示例	265
Yokogawa Memobus	267
配置	267
连接	267
任务参数	268

CSV	268
示例	269
Yokogawa Link TCP	271
配置	271
任务参数	271
CSV	272
示例	272
Panasonic MEWTOCOL	274
配置	274
任务参数	274
CSV	275
示例	276
SNMP	278
配置	278
自动发现	279
CSV	279
示例	280
ActiveX Bridge	282
配置	282
CSV	283
脚本 API	283
示例	284
Custom Driver	286
配置	286
CSV	287
示例	288

（在 Word 中右键目录选择“更新域”可刷新页码）

第一部分 入门

安装 DX

安装 DX 会把 DX 运行时及所有模块安装到你的计算机上。它会启动用于运行项目的 DX 引擎，搭建项目与 DX Edit 通信所用的 **Mosquitto** 消息代理，并添加示例项目 **DEM0**。每个平台对应一个安装包：Windows 平台为 **.exe**，Linux 平台为 **.deb**。本指南将带你从下载的安装文件开始，直到打开 DX Edit 并让 DEM0 运行起来。此后，DX Edit 指南会继续带你创建属于自己的项目。

Windows

所需条件

- 64 位 Windows，DX 在 Windows 11 上完成测试。
- 管理员账户。如果你以标准用户身份登录，安装过程中 Windows 会要求输入管理员密码。
- 一个默认的 Web 浏览器，以便在其中打开 **.html** 文件，DX Edit 会在浏览器中打开。
- 安装程序 **ecava-dx_<version>_x64.exe**。请使用 **.exe**。它旁边的 **.msi** 用于静默安装，详见“在多台计算机上安装”。

下面的每一步都展示了你将看到的页面，图片上的编号与旁边带编号的步骤一一对应。

开始安装

双击 **ecava-dx_<version>_x64.exe**。

1. 你正在安装的版本。

2. 点击 **Install**。

随后 Windows 会弹出“你是否要允许此应用对你的设备进行更改？”，并标明发布者者为 **Ecava Sdn Bhd**。点击 **Yes**。如果你是标准用户，则需输入管理员的用户名和密码。

欢迎

1. 点击 **Next**。

接受许可协议

1. 阅读协议，然后勾选 **I accept the terms in the License Agreement**。

2. 点击 **Next**。

选择安装选项

四个选项默认全部勾选。首次安装时请全部保留。

1. 安装 Mosquitto MQTT 消息代理
2. 用 DX 默认值覆盖 mosquitto.conf
3. 应用推荐的 PostgreSQL 配置
4. 覆盖示例项目
5. 点击 **Next**。

选项	作用	取消勾选的情形
安装 Mosquitto MQTT 消息代理	将 Mosquitto 安装为 Windows 服务，你的项目和 DX Edit 通过它通信。	你已经在运行自己搭建的消息代理。
用 DX 默认值覆盖 mosquitto.conf	写入 DX 所需的消息代理设置：项目使用端口 1883，DX Edit 使用端口 1884，无需登录。你的旧文件将被替换且不做备份。	你已经自行编辑过 mosquitto.conf 并希望保留它。此时该文件必须包含相同的两个监听器。
应用推荐的 PostgreSQL 配置	当本机已安装 PostgreSQL 时，设置 work_mem = 256MB ，供历史记录模块的查询使用。若未安装 PostgreSQL，则不做任何操作。	为每个连接分配这么多内存，对你的 PostgreSQL 服务器而言过大。
覆盖示例项目	用全新副本替换 DEM0 。如果 DEM0 缺失，则一定会添加。	你已经编辑过 DEM0 并希望保留所做的更改。

安装

1. 点击 **Install**。

1. 安装程序会停止所有正在运行的 DX 项目，放入新文件并启动 DX 引擎，这需要几分钟时间。

完成

1. 点击“完成”时，**Open DX Edit in the browser** 会打开 DX Edit。

2. **Run DEM0 project** 会启动示例项目。

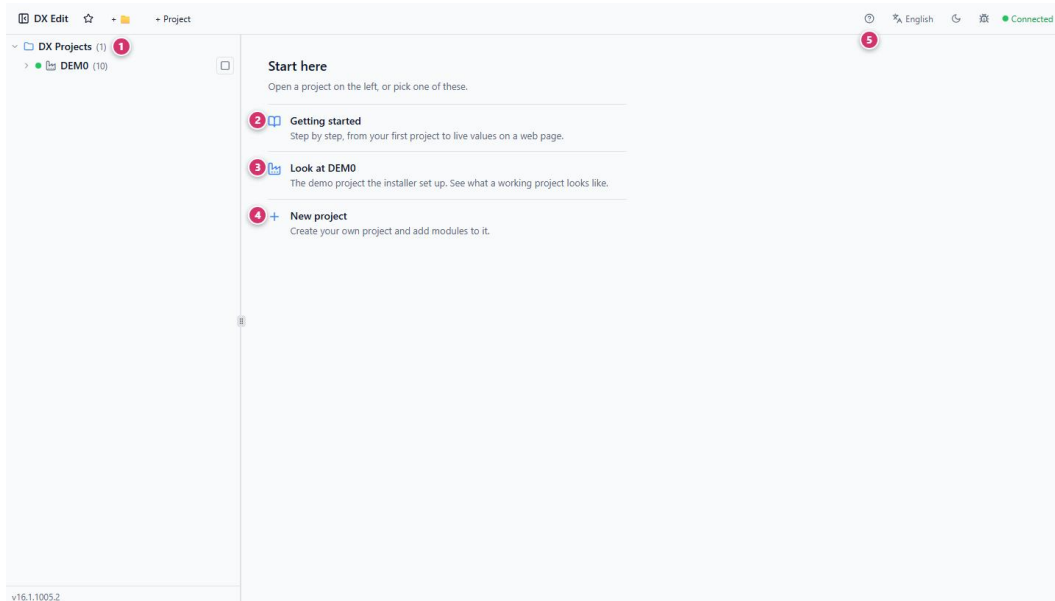
3. 点击 **Finish**。安装程序会等待 DEM0 启动，然后关闭。

DX Edit 打开

DX Edit 在你的浏览器中打开。DEM0 自己的网页也会在另一个标签页中打开，请切回 DX Edit 标签页。

1. **DX Projects** 中列出了 DEM0。它运行后，对应的圆点会变为绿色。

2. **Getting started** 会打开 DX Edit 指南，带你完成一个属于自己的项目。



安装完成。请继续阅读“DX Edit：入门”。之后也可从 Windows 开始菜单打开 DX Edit：DX Edit。

注意：消息代理在端口 1883 和 1884 上监听且无需登录，任何能访问它的人都能查看并控制你的 DX 项目。安装程序不会为这些端口添加防火墙规则。如果你开放这些端口以便其他计算机连接，请将本机置于你信任的网络中。

Linux

所需条件

- **Ubuntu 22.04 LTS or newer**（在 24.04 上测试）。Debian 12 或更高版本也可使用。
- **systemd**，用于运行 DX 引擎和消息代理。正常安装时它处于启用状态。在 WSL 中请参阅“在 WSL 中”。
- 适合你 CPU 的安装包：64 位 Intel/AMD 计算机使用 **ecava-dx_<version>_x64.deb**；64 位 ARM（运行 64 位操作系统的树莓派 4 或 5）使用 **_a64.deb**；32 位 ARM 使用 **_a32.deb**。
- 安装期间需要联网，以便 **apt** 获取 Mosquitto。若无网络，请参阅“无网络时安装失败”。
- 安装本身需要 **sudo**。之后无需它即可编辑项目。

安装

刷新 **apt** 的软件包列表，然后从安装包所在文件夹安装：

```
sudo apt update
sudo apt install ./ecava-dx_<version>_x64.deb
```

开头的 **./** 必不可少，它告诉 **apt** 安装本地文件。安装会把 DX 放到 **/opt/ecava/dx** 下，启动 DX 引擎，安装带有 DX 监听器的 Mosquitto，并把 DEM0 复制到 **/var/lib/ecava/dx/projects**。

最后几行会提示是否安装成功：

```
Ecava DX installed under /opt/ecava/dx.
  create + run a project : dxrun <projectname>      (starts dxmanager +
  dxweb)
  service status          : dxrun status
```

```
list modules          : dxrun list
DX: wrote /etc/mosquitto/conf.d/dx.conf (mqtt 1883 + websockets 1884,
anonymous).
DX: DEM0 demo seeded under /var/lib/ecava/dx/projects.
DX: projects under /var/lib/ecava/dx/projects are editable by '<you>'
without sudo (group '<you>', setgid + group-write).
DX: DEM0 demo started.

Please connect through DX Edit using 192.0.2.10:1884
```

- **“DX: wrote /etc/mosquitto/conf.d/dx.conf”** 表示消息代理已为 DX Edit 准备就绪。如果缺少这一行，请参阅“没有‘wrote dx.conf’这一行”。
- **The last line** 给出了在 DX Edit 中使用的地址 **<address>:1884**。
- 请在 DX Edit 中创建项目。dxrun <projectname> 这一提示是命令行快捷方式，本指南不使用它。

Windows 安装程序提供的选项也可在命令行中设置。上面的普通命令使用默认值，仅在需要更改时才添加相应项：

选项	默认值	在命令中添加	效果
安装 Mosquitto 消息代理	安装	--no-install-recommends	安装 DX 时不安装 Mosquitto。当你运行自己的消息代理时使用。
写入 DX 消息代理配置	写入	DX_SKIP_MOSQUITTO=1	保持 Mosquitto 的配置不变。当 1883 端口上已存在监听器时，会自动跳过此步骤。
应用推荐的 PostgreSQL 设置	在已安装 PostgreSQL 时应用	DX_SKIP_PGCONF=1	保持 PostgreSQL 不变。该设置为 work_mem = 256MB，按每个连接应用。如果它对你的服务器过大，请跳过。
覆盖示例项目	替换	DX_KEEP_DEMO=1	保留你编辑过的 DEM0。默认情况下，DEM0 会被全新副本替换，升级时也是如此。
运行 DEM0 项目	启动	DX_NO_START_DEMO=1	让 DEM0 保持停止。默认情况下，安装后会启动它，升级时也是如此。

变量放在 `sudo` 和 `apt` 之间；`apt` 选项放在 `install` 之后。例如，要保持 PostgreSQL 不变并保留你编辑过的示例：

```
sudo DX_SKIP_PGCONF=1 DX_KEEP_DEMO=1 apt install ./ecava-  
dx_<version>_x64.deb
```

从另一台计算机访问消息代理

安装后，消息代理即可接受来自网络的连接：DX Edit 使用端口 **1884**，任何模块或 SCADA 服务器使用端口 **1883**。用 `hostname -I` 查看本机地址。如果本机运行防火墙，请放行这两个端口：

```
sudo ufw allow 1883,1884/tcp
```

注意：消息代理对你网络中的所有人开放且无需登录，任何能访问它的人都能查看并控制你的 DX 项目。请将 DX 计算机置于你信任的网络中。

在 WSL（适用于 Linux 的 Windows 子系统）中

WSL 需要启用 `systemd`。WSL 上的 Ubuntu 默认已启用；如果 `systemctl` 报告 `systemd` 未运行，请将以下几行添加到 `/etc/wsl.conf`，然后在 Windows 中运行 `wsl --shutdown` 并重新打开 WSL：

```
[boot]  
systemd=true
```

由于 WSL 的网络机制，仅打开端口无法让 Windows 访问 WSL 内的消息代理。DX 为此提供了一个小型转发器，只需启用一次：

```
sudo apt install socat          # if it is not already present  
sudo systemctl enable --now ecava-dx-wsl-forward
```

这会把消息代理发布到 WSL 计算机自己的地址上，并在每次重启时重新启动。请将 DX Edit 指向该地址（用 `hostname -I` 查看）。WSL 重启后地址可能变化：如果重启后 DX Edit 无法连接，请重新运行 `hostname -I` 检查。请通过该地址而不是 `localhost` 访问 WSL，因为 Windows 上的 `localhost` 是另一个消息代理。这

里假设 WSL 使用默认网络模式；在镜像模式下，WSL 与 Windows 共享端口，两个消息代理会在 1883 和 1884 上冲突。

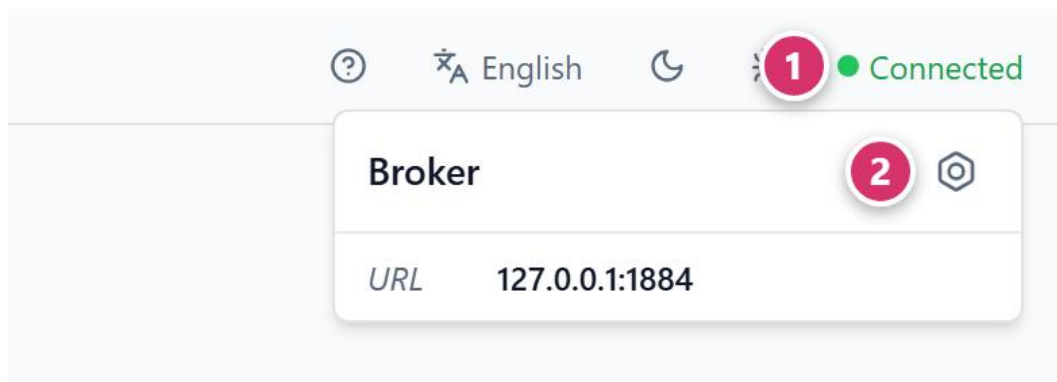
打开 DX Edit

On the Linux machine's own desktop，运行：

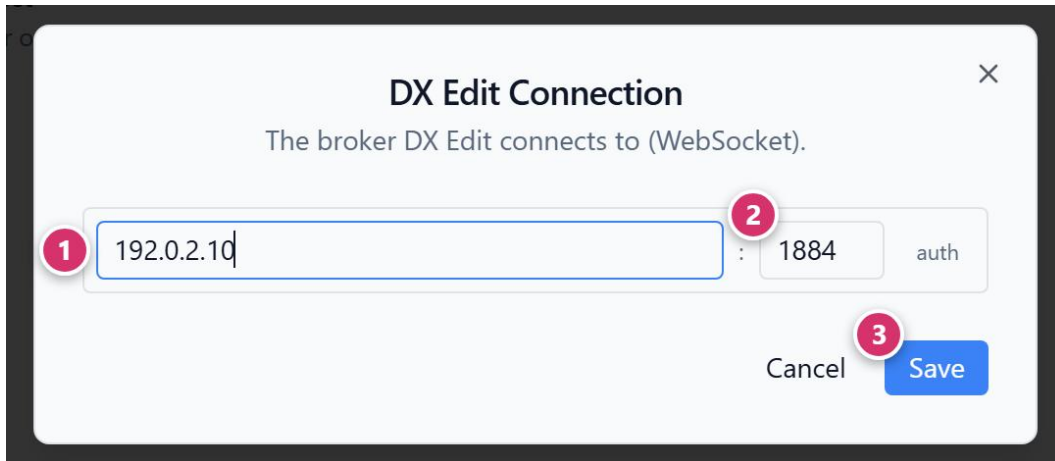
```
dxrun --open-editor
```

From another PC 已安装 DX，请从开始菜单打开 DX Edit，并将其指向该 Linux 计算机：

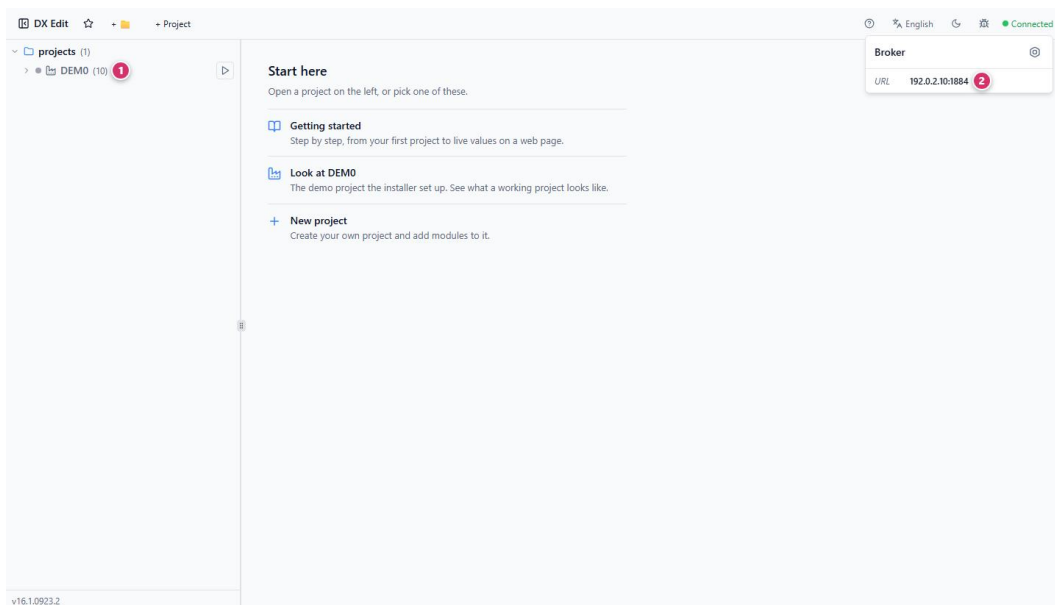
1. 点击右上角的连接状态 **Connected**。
2. 点击六边形图标 **Configure DX Edit connection**。



1. **Host:** Linux 计算机的地址，来自安装时的最后一行。
2. **Port:** 1884。
3. 点击 **Save**。



1. DX Edit 重新加载，连接到 Linux 计算机，并列出其项目，其中包括 DEMO。
2. 消息代理菜单会显示 DX Edit 所连接的地址。



此后你就可以像在自己的计算机上一样处理 Linux 计算机的项目：创建、启动、停止它们并编辑其设置。

重新安装相同版本

apt 会跳过已安装的版本。要重新安装 **same .deb** 以修复损坏的安装，请使用 dpkg:

```
sudo dpkg -i ./ecava-dx_<version>_x64.deb
```

安装选项在这里同样适用：把所有 `DX_...` 变量放在 `dpkg` 之前。如果它报告缺少依赖，请接着运行 `sudo apt install -f`。如果它在配置文件或某个不愿覆盖的文件上停滞，请强制执行：

```
sudo dpkg -i --force-confnew --force-overwrite ./ecava-dx_<version>_x64.deb
```

下一步

- **Build a project of your own.**“DX Edit: 入门”会带你从 DEM0 过渡到自己的项目，读取设备并获得实时值、历史记录、趋势和报警。
- **See what runs behind DX Edit.**“DX Runner”指南讲解 DX 引擎以及项目如何作为服务运行。
- **Upgrade later.** 以相同方式运行更新的安装程序。升级会停止所有 DX 项目，且只重新启动 DEM0；请从 DX Edit 启动你自己的项目。
- **Install on many PCs.** 详见“在多台计算机上安装”。
- **Remove DX.** 详见“卸载”。

出现问题时

再次运行安装程序时显示“修复”和“卸载”

相同版本已经安装。**Repair** 会还原已安装的文件并用全新副本替换 DEM0，不会询问。结束时它不会打开 DX Edit。**Uninstall** 会移除 DX，详见“卸载”。

1. **Repair** 会还原已安装的文件。

2. **Uninstall** 会移除 DX。

“已安装更新版本的 Ecava DX”

安装程序不会用旧版本替换更新的 DX。请使用更新的安装程序，或先卸载 DX。

“Windows 已保护你的电脑”

对于从互联网下载的文件，Windows 可能会显示此提示。点击 **More info**，确认发布者 **Ecava Sdn Bhd**，然后点击 **Run anyway**。如果未显示发布者，请勿运行该文件，从 Ecava 重新下载。

安装程序要求输入管理员密码

你以标准用户身份登录。请输入管理员的用户名和密码，或请你的 IT 团队运行安装程序。

DX Edit 没有打开，或作为文本打开

DX Edit 是你计算机上的一个页面，在默认 Web 浏览器中打开。请从 Windows 开始菜单打开它：**DX Edit**。如果它作为文本或在其他程序中打开，请在 Windows 设置中将 Web 浏览器设为 .html 文件的默认打开程序，然后重新打开。

DX Edit 提示无法连接消息代理

DX Edit 通过端口 1884 与消息代理通信。本机上的另一个消息代理可能已占用端口 1883 或 1884，或者在没有运行自己的消息代理时取消勾选了“安装 Mosquitto MQTT 消息代理”（Install the Mosquitto MQTT broker）。请参阅“DX Edit 无法连接消息代理”。

DEM0 未运行

在 DX Edit 中选择 **DEM0** 并点击 **Start**。全新安装后，启动可能需要一点时间。

DEM0 不在你看到的“文档”文件夹中

DX 将包括 DEM0 在内的项目保存在你配置文件自己的“文档”文件夹下的 **DX Projects** 中。当 OneDrive 移动了你的“文档”文件夹时，文件资源管理器的“文档”会位于别处。DX Edit 始终会列出你的项目，因此请从那里打开。

安装被取消或中途失败

正在运行的 DX 项目已被停止，因此保持停止状态。请重新运行安装程序。如果再次失败，请带日志运行并将日志发送给 Ecava 支持：

```
ecava-dx_<version>_x64.exe -log "%TEMP%\dx-setup.log"
```

安装完成后，从 DX Edit 重新启动你的项目。

安装程序要求重启 Windows

DX 使用的某个文件仍被打开（例如被另一个程序占用）。请在方便时重启，DX 会在重启后完成安装。

升级后你的项目处于停止状态

每次升级都会停止你的 DX 项目，且只重新启动 DEM0。请从 DX Edit 启动你的项目。

apt 提示“无法定位软件包”

缺少开头的 ./，因此 apt 在其在线列表中查找该名称。请重新运行命令，在文件名前加上 ./。

apt 等待锁定

另一个 apt 任务正在运行，通常是刚启动的计算机上 Ubuntu 的自动更新。请等待它完成，apt 会自行继续。

apt 提示“以 root 身份执行非沙盒下载”

apt 无法以自己的用户身份读取文件，通常是因为 .deb 位于你的主文件夹或 Windows 驱动器上。安装不受影响。为避免该提示，请先将 .deb 复制到 /tmp。

无网络时安装失败

无网络时，apt 无法获取 Mosquitto。请在不安装它的情况下安装 DX，然后将 DX 指向你自己运行的消息代理：

```
sudo apt install --no-install-recommends ./ecava-dx_<version>_x64.deb
```

没有“wrote dx.conf”这一行

DX 没有为 DX Edit 配置消息代理。要么是 Mosquitto 未安装，要么是已存在在 1883 端口上监听的消息代理，DX 保持其配置不变。DX Edit 需要在 1884 端口上有一个允许免登录连接的 WebSocket 监听器。请向你的消息代理添加一个，或安装 Mosquitto 并重新安装 DX。

“DX 引擎尚未启动”

安装完成时 DX 引擎未运行，因此 DEM0 未启动。请检查 systemd 是否运行（在 WSL 中请参阅“在 WSL 中”），然后从 DX Edit 启动 DEM0。

另一台计算机上的 DX Edit 无法连接

在 Linux 计算机上用 `hostname -I` 检查地址，并确认防火墙放行端口 1883 和 1884。在 WSL 中，请检查转发器是否启用，以及 WSL 重启后地址是否变化。

在 Linux 上检查 DX 引擎和消息代理

```
dxrun status # what is running
dxrun logs <project> # follow a project's log (Ctrl+C to stop)
systemctl status ecava-dxrun mosquito # the DX engine and the broker
```

DX 引擎（`ecava-dxrun`）和消息代理（`mosquitto`）都会在开机时启动。

在多台计算机上安装

`.exe` 旁边的 `.msi` 可在不使用向导的情况下安装 DX。请从以 **Run as administrator** 方式打开的命令提示符中运行：

```
msiexec /i ecava-dx_<version>_x64.msi /qn /l*v "%TEMP%\dx-install.log"
```

不要双击 `.msi`：没有管理员权限，它会停在“正在验证安装”。手动安装请使用 `.exe`。

安装选项以属性形式存在，默认全部开启。将某个属性设为 `""` 即可关闭它：

属性	安装选项
INSTALL_MOSQUITTO	安装 Mosquitto MQTT 消息代理
UPDATE_MOSQUITTO_CONF	用 DX 默认值覆盖 <code>mosquitto.conf</code>
UPDATE_PG_CONF	应用推荐的 PostgreSQL 配置
OVERWRITE_DEMO	覆盖示例项目
START_DEMO	运行 DEMO 项目

例如，要保留编辑过的 DEMO 并保持 PostgreSQL 不变：

```
msiexec /i ecava-dx_<version>_x64.msi /qn OVERWRITE_DEMO=""
UPDATE_PG_CONF=""
```

卸载

Windows. 打开 **Settings**，然后依次打开 **Apps**、**Installed apps**，找到 **Ecava DX** 并选择 **Uninstall**。你在 **Documents\DX Projects** 中的项目会保留。

Linux.

```
sudo apt remove ecava-dx
```

这会移除 DX、其消息代理设置（`dx.conf`）及其 PostgreSQL 设置。你在 `/var/lib/ecava/dx/projects` 中的项目会保留，Mosquitto 仍会安装；如果没有其他程序使用它，可用 `sudo apt remove mosquitto` 移除。

文件位置

Windows

Linux

DX Run

DX Run (dxrun) 是 DX Edit 的常驻后端。每台计算机只需一个 DX Run，它负责运行你的项目，并让 DX Edit 能够创建、启动、停止和删除项目。它在 Windows 和 Linux 上的行为完全相同，随 DX 一起安装（见安装指南）。

一个后端服务所有项目

DX Run 作为单个后台服务运行，无论是否有项目在运行，它都保持启动。通过它，DX Edit 可以：

- **creates** 项目——在你的 **DX Projects** 文件夹中创建，处于停止状态、等待添加模块；
- **starts, stops, and restarts** 项目及其模块；
- **deletes** 不再需要的项目；
- **reports what is actually running**——即使是在 DX Edit 之外启动或停止的服务也能识别——因此每个项目和模块都在树中显示其真实状态。

DX Run 会自行启动每个项目的模块——以 **background services**（默认方式：Windows 服务，或 Linux 上的 systemd 服务）或桌面上的 **console** 窗口方式运行——因此项目不再需要自己的管理器来运行这些模块。所有这些操作都在 DX Edit 中完成。

启动你的第一个项目

在项目应存放的文件夹中运行 dxrun，并给它一个名称：

```
dxrun MyPlant
```

- **New project** ——DX Run 会在那里创建一个全新的空白项目并启动它。
- **Existing project** ——如果该文件夹已存在该项目，DX Run 保持你的配置不变，直接启动它。

项目启动后，打开 DX Edit 添加并配置模块。此后，DX Edit 中的 **+ Project** 会为你创建更多项目，**Start** 则负责运行它们。

选择 Web 端口

默认情况下，每个项目的 Web 服务器会从 80 开始挑选第一个空闲端口，因此你可以在一台计算机上运行多个项目而无需为每个项目分配端口。要将项目固定到特定端口，请传入 `--port`：

```
dxrun MyPlant --port 81
```

显式设置的端口会原样使用：如果该端口已被占用，DX Run 会停止并要求你选择一个空闲端口——它绝不会自行移动固定端口，也不会留下半成品。

打开 DX Edit

```
dxrun --open-editor
```

在浏览器中打开 DX Edit。

要访问 **another machine** 上的 DX Run，请在你自己的计算机上打开 DX Edit，点击右上角的连接状态，然后点击六边形图标 **Configure DX Edit connection**，输入该计算机的地址和端口 **1884**。之后你就可以像在自己的计算机上一样创建、启动和停止其项目。详见安装指南。

DX Run 在 Linux 上的文件位置

项目位于 `/var/lib/ecava/dx/projects`（Windows 下为你的 **DX Projects** 文件夹）。引擎自己的设置文件 `dxrun.json` 位于同一文件夹中；其 `servers` 条目指定 DX Run 使用的消息代理，默认为 **127.0.0.1:1883**，通常保持原样即可——该字段在通用配置指南中有说明。

DX Projects 文件夹之外的项目

项目通常位于你的 **DX Projects** 文件夹中，但你也可以把它放在 **anywhere** ——另一个驱动器或共享位置。DX Run 会记住每个项目的位置，因此它仍会出现在 DX Edit 中。为安全起见，它只能访问你已 **allowed** 的文件夹：在外部位置创建项目会把该位置添加到允许列表，而系统保护的文件夹始终会被拒绝。

Adopt a stray service. 没有对应项目而遗留的 DX 服务——例如升级后或手动安装的服务——可以重新转为完整项目。在 DX Edit 中它出现在 **Ungrouped Services**

下；**Adopt** 它，DX Run 会把项目文件写入其文件夹并开始管理它。DX Projects 区域之外的文件夹必须先被允许（将其作为项目添加即可），系统保护的文件夹始终会被拒绝。

从命令行管理 DX Run

DX Edit 是日常管理项目的方式；这些命令用于首次安装或脚本编写。

命令	作用
<code>dxrun <name> [--port <n>]</code>	从当前文件夹创建（如果是新项目）并以服务方式启动项目。 <code>--port</code> 固定其 Web 端口；若省略，则 Web 端口从 80 开始自动选择。
<code>dxrun stop <name></code>	停止项目及其模块，释放 Web 端口。
<code>dxrun restart <name></code>	停止后再重新启动项目。
<code>dxrun status</code>	列出正在运行的项目。
<code>dxrun list</code>	列出可用的模块类型。
<code>dxrun logs <name></code>	跟踪项目的日志（Linux）。
<code>dxrun --open-editor</code>	打开 DX Edit（在 Linux 上会带会话令牌）。
<code>-?、-h、--help</code>	显示用法并退出。

DX Edit

DX Edit 让你通过 Web 浏览器查看和编辑 DX 模块配置——每个模块读取什么、连接到哪里——而无需手动编辑配置文件。它随 DX 一起安装，可直接从 Windows Start menu（DX Edit）打开，无需记住任何地址，并会自行连接。

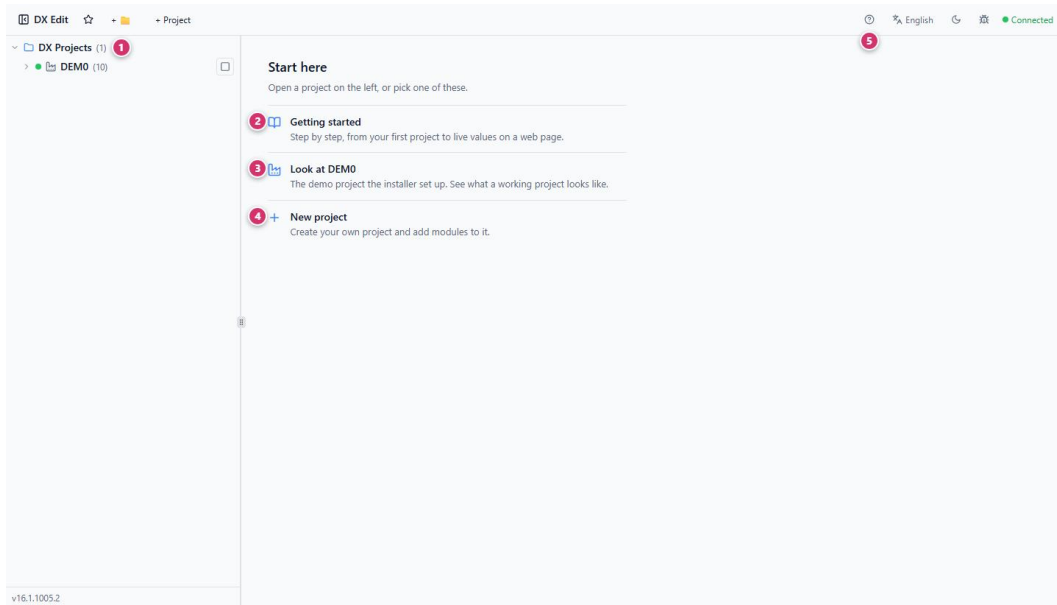
入门

本演练将带你从一个已完成的安装，到建立一个属于你自己的项目：读取设备、实时显示其数值、保存历史、在趋势图上绘制曲线并触发报警。每一步都展示了你将看到的屏幕，图片上的编号与旁边带编号的步骤相对应。你不需要真实设备：演示项目 DEM0 包含一个模拟设备。

打开 DX Edit

安装完成后，DX Edit 会在浏览器中打开（安装过程见“安装 DX”）。之后可从 Windows 开始菜单打开它：**DX Edit**。演示项目 DEM0 会同时启动，它自己的网页可能在另一个标签页中打开。

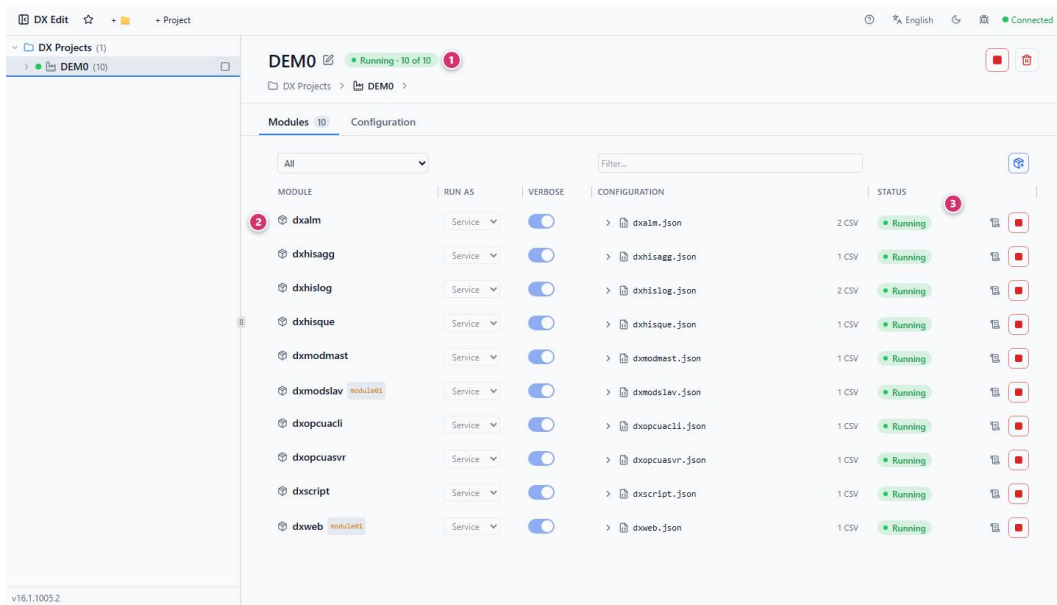
1. DX Projects 列出你的项目。DEM0 已经在那里并正在运行（其圆点为绿色）。
2. Getting started 打开本指南。
3. Look at DEM0 打开演示项目。
4. New project 创建你自己的项目。
5. 顶部的?可从任意页面打开用户指南。



查看一个正在运行的项目

点击 Look at DEMO。项目页面会列出其模块。每个模块是承担一项职责的服务，其名称说明了职责：dxmodmast 读取 Modbus 设备，dxweb 提供网页，dxscript 运行脚本，dxalm 触发报警，dxhislog、dxhisagg 和 dxhisque 保存并提供历史。

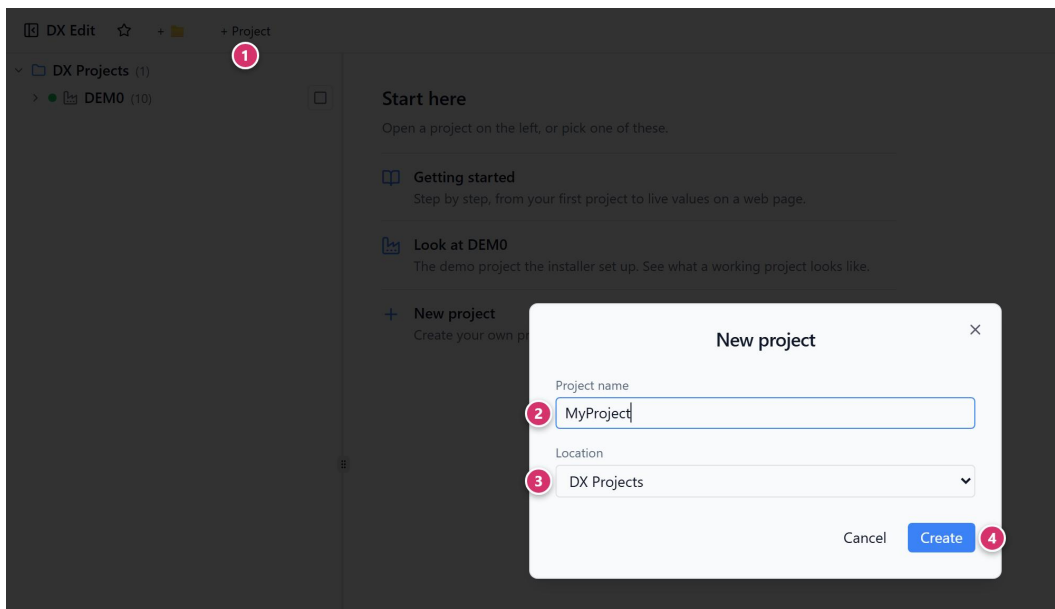
1. 项目的状态。Running · 10 of 10 表示全部十个模块都在运行。
2. 每一行是一个模块。要查看模块的设置，点击左侧树中它的名称。
3. 模块的状态。绿色的 Running 表示它正在运行并已上报。



DEMO 还提供网页，例如这个实时工艺画面。可在 <http://localhost/> 打开它们。

创建你的项目

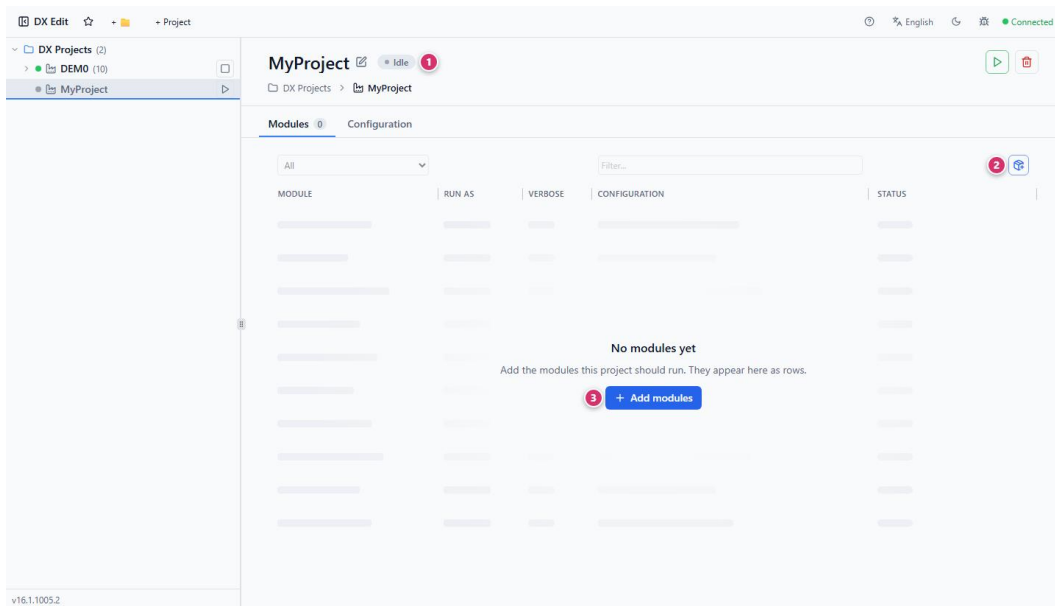
1. 点击+ Project（或第一屏上的 New project）。
2. 输入一个名称，例如 MyProject。可使用字母、数字、点、连字符和下划线。
3. 将 DX Projects 保留为位置。
4. 点击 Create。



添加模块

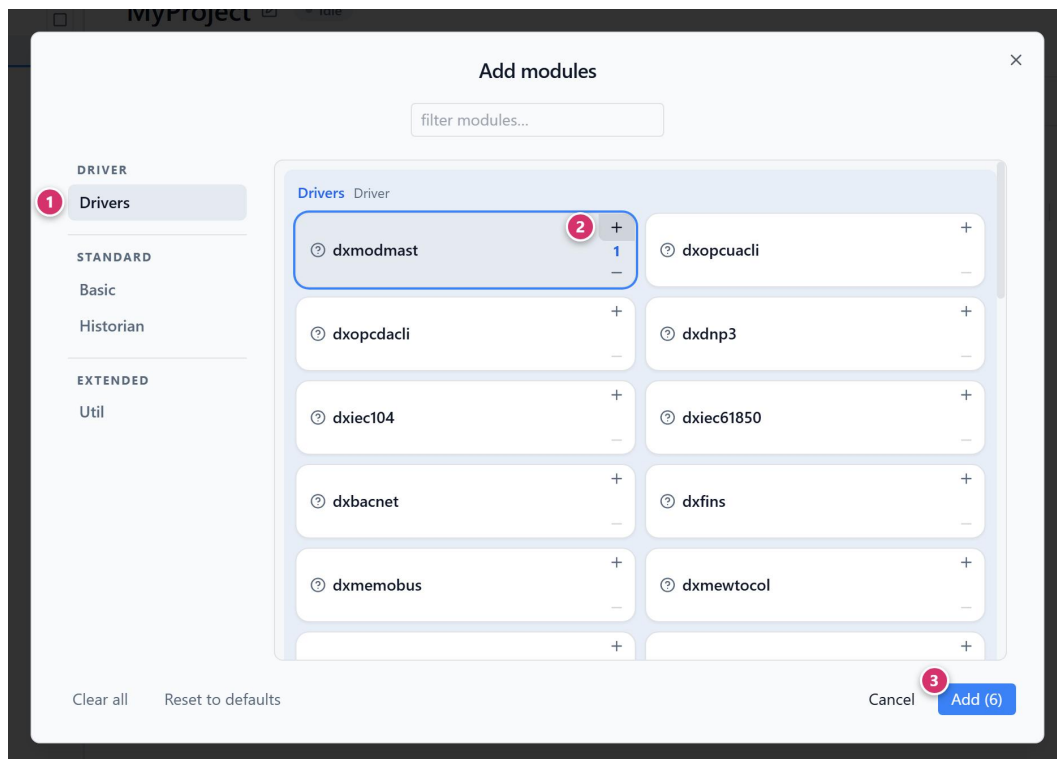
新项目在其页面打开，为空。

1. **Idle:** 尚无任何模块运行。
2. 点击列表上方的添加模块图标，
3. 或点击中间的 **Add modules**。两者都会打开模块列表。

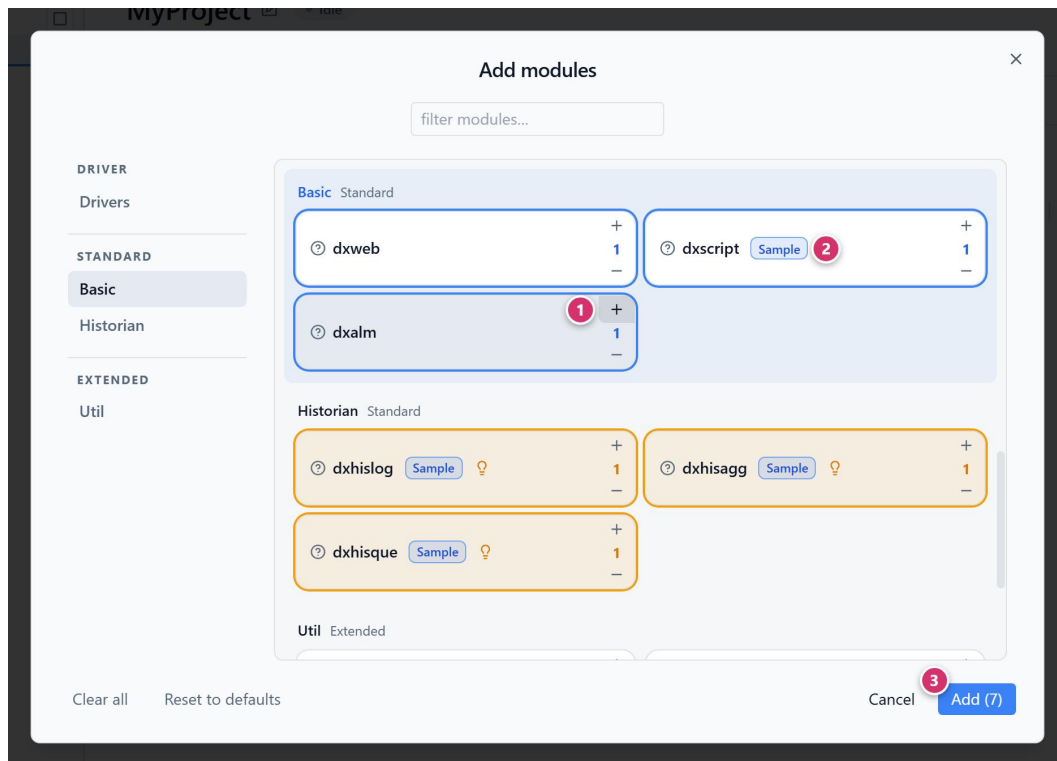


对于新项目，一个可运行项目所需的模块已预先选好：**Web 服务器（dxweb）**、脚本引擎（**dxscript**）和三个历史模块。再向其中添加一个设备驱动和报警模块。

1. Drivers 列出设备驱动。
2. 点击 dxmodmast（Modbus 主站）上的+。
3. 每选一个模块，Add 上的计数就会增加。

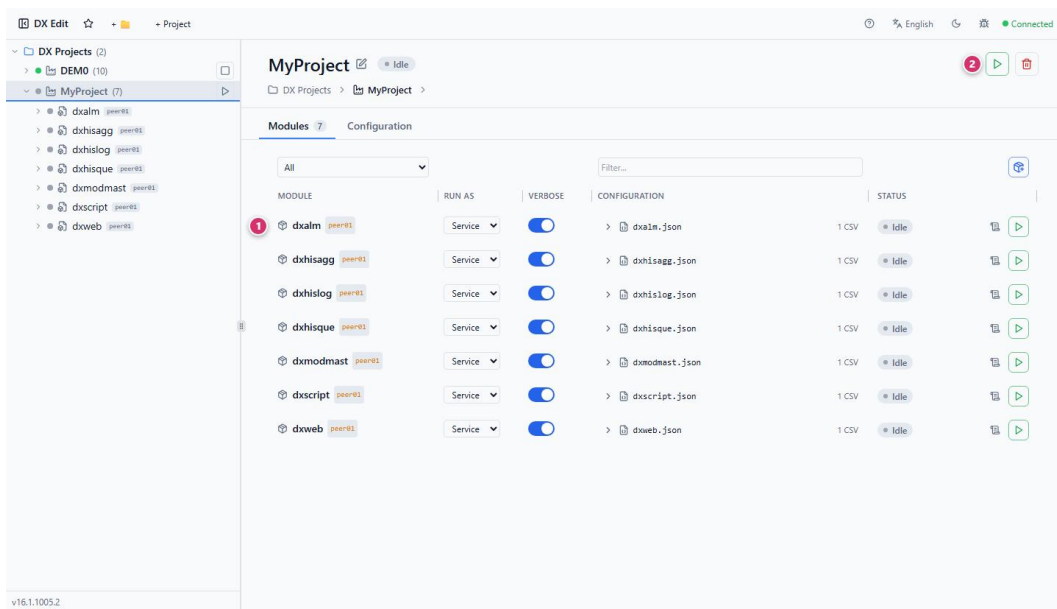


1. 点击左侧的 Basic，然后点击 dxalm（报警）上的+。
2. 有示例的模块上 Sample 处于点亮状态：它们创建后即可协同工作。保持其点亮。
3. 点击 Add (7)。



历史模块上的灯泡表明 **dxalm** 需要它们。它们已在列表中，因此不会再添加任何内容。

1. 这七个模块，每个都处于 **Idle** 状态，以其类型命名，等待调整。
2. 右上角的▶（启动）运行整个项目。

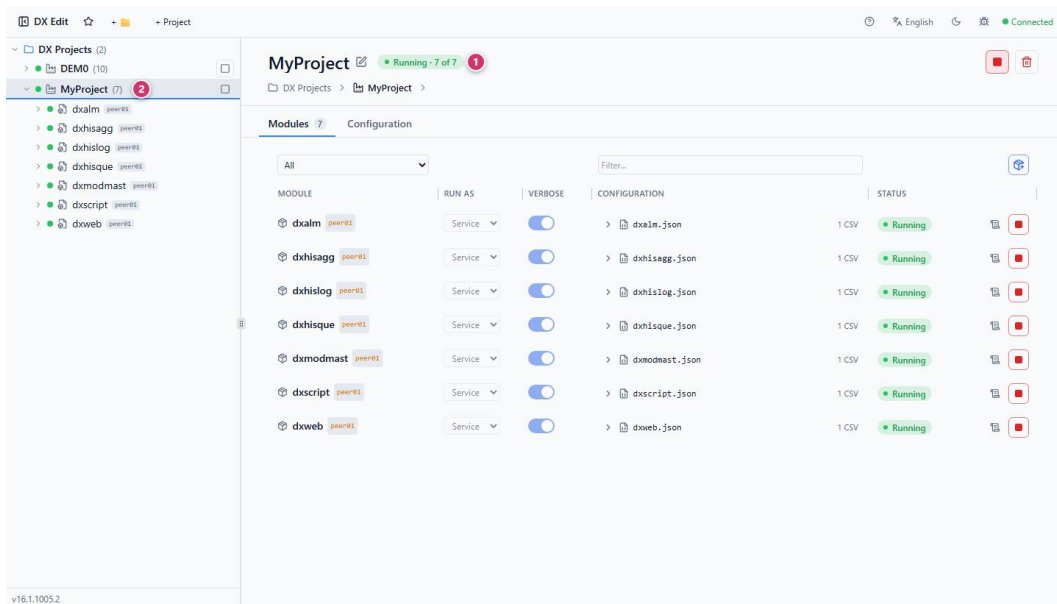


启动项目

点击项目页面右上角的▶（启动）。

1. Running · 7 of 7: 每个模块都在运行。模块启动时可能显示琥珀色长达 20 秒。

2. 树中项目的圆点变为绿色。



启动项目还会在浏览器中打开其网页。DEM0 的 Web 服务器已占用 80 端口，因此 MyProject 会使用下一个空闲端口：<http://localhost:81/>。

注意：没有许可证时，每次启动运行 2 小时。项目停止后，再次按 **Start**。

连接你的设备

点击树中的 **dxmodmast** 打开其设置，然后向下滚动到 **Tasks**。每个任务卡片对应一个设备。本演练中保持这些值不变：它们已指向 DEM0 的模拟设备，因此此处无需应用。

1. URL：设备的地址。127.0.0.1 是本机，DEM0 在此模拟一个 Modbus 设备。
2. 端口。Modbus TCP 使用 502。
3. 网络上的设备选 TCP。COM 端口上的设备选 Serial (RTU)。
4. 设备的 Station ID，此处为 1。

▼ Tasks (1) + item Find & replace

1

Enable ☒

ID task01

1 URL i 127.0.0.1 : 2 502 3 TCP ▼

4 Station ID i 1 Short Connection i ☐

Timeout (ms) i 2000

▼ Trigger (1) + item

[0] 1000 interval

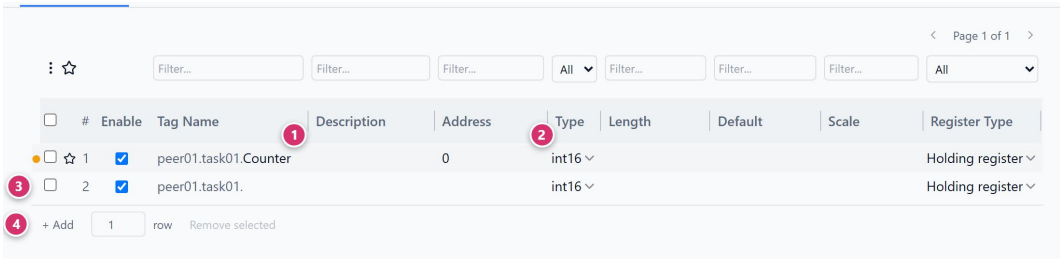
对于你自己的设备，改为输入其地址和站号。该任务的每项设置都在 Modbus 主站指南中说明。

添加要读取的值

在树中，点击 dxmodmast 旁边的箭头，然后点击其下方带网格图标和 task01 徽章的那一行。那是模块的值列表（其 CSV 文件）；每一行是一个要读取的值。

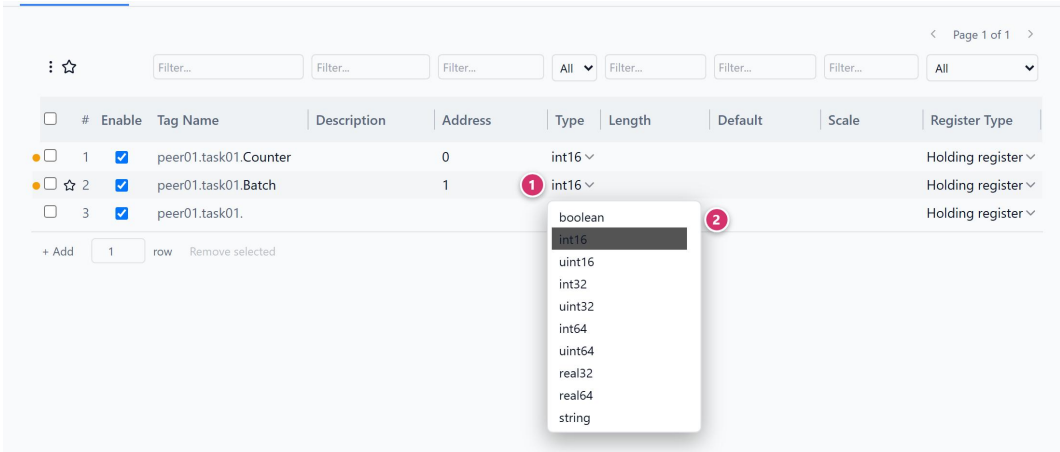
1. 输入一个 Tag Name: Counter，然后按 Enter。它前面灰色的 peer01.task01.会自动补上，因此完整名称为 peer01.task01.Counter。
2. 输入它在设备上的 Address: 0，然后按 Enter。
3. 最后一行下方总有一个空行等待。将其填为 Batch，地址 1。

4. + Add 一次添加多个空行。以这种方式添加的行在名称和地址填好前会以橙色框出。



1. Type 设置值的读取方式。点击它可从列表中选择。

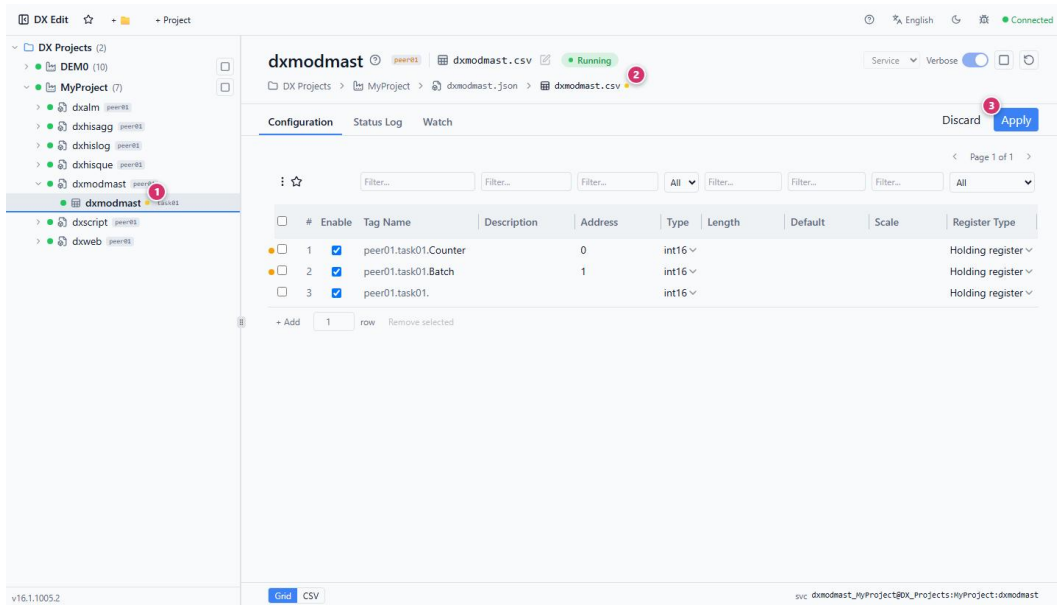
2. 类型。int16 适合这两个值，因此保持它，并将 Holding register 保留为 Register Type。



1. 树中一个黄色圆点标记了一个有尚未应用更改的文件，

2. 以及顶部的路径中。

3. Apply 使它们生效。

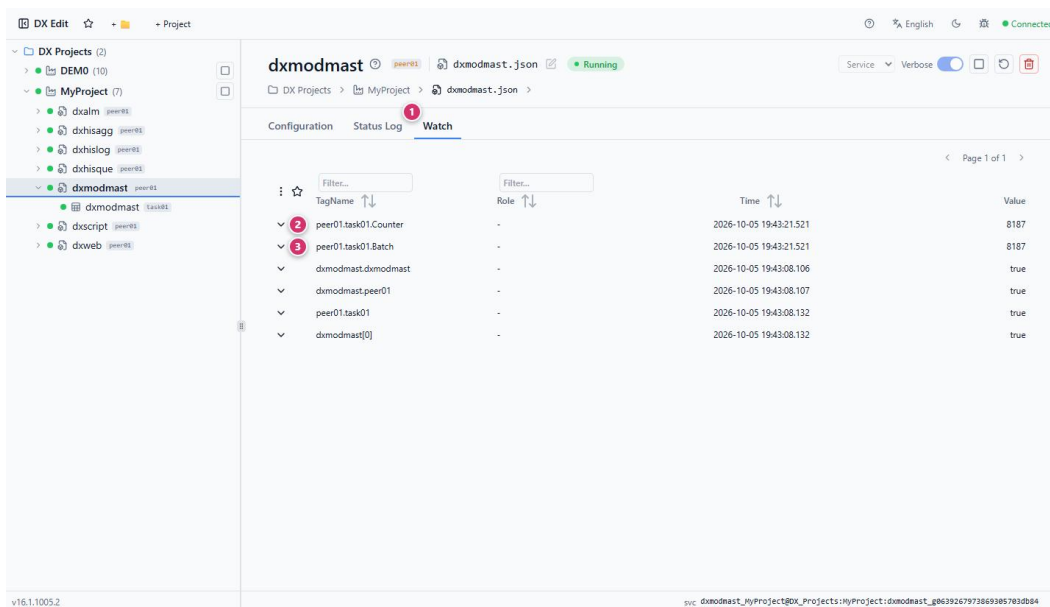


更改在你输入时即被保存，即使你离开页面，直到你 Apply 或 Discard 它们。

应用并查看数值

点击 Apply。只有 dxmodmast 重启；项目的其余部分继续运行。然后点击它的 Watch 标签（模块运行时该标签存在）。

1. Watch 显示模块的实时值。
2. peer01.task01.Counter，从地址 0 读取。它持续递增。
3. peer01.task01.Batch，从地址 1 读取。DEMO 的模拟器在两个地址放置相同的计数，因此两者显示相同的值。



它们下方的行是模块自己的状态值；你可以保持原样。

在趋势图上查看你的值

历史无需任何设置：当其列表为空时，历史记录器（dxhislog）记录项目发布的每个值。打开项目的网页（<http://localhost:81/>）并选择 Trending。

1. Trending 绘制项目的每个值。点击图表上方的名称可隐藏或显示其曲线。

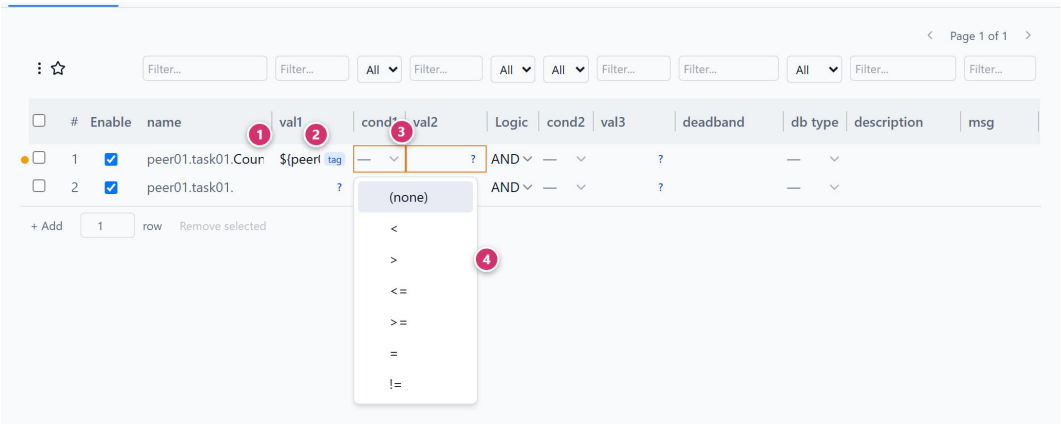


你的两个值在靠近顶部绘制曲线；它们保持相同的计数，因此一条曲线覆盖了另一条。关闭页面再重新打开：曲线会从历史中恢复。

添加报警

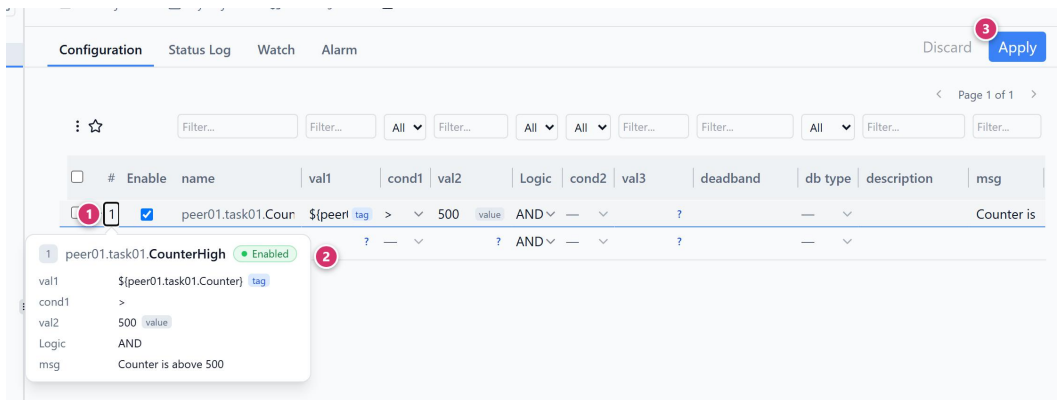
在树中，点击 **dxalm** 旁边的箭头，然后点击其下方带网格图标的那一行。每一行是一个报警条件。当一行填写了一半时，它仍需要的单元格会以橙色框出；行填写完整后框线消失。

- 1. **name:** CounterHigh。
- 2. **val1:** 要监视的标签`${peer01.task01.Counter}`。tag 徽章表明它作为一个标签被读取。
- 3. **cond1:** 点击它可选择比较方式。
- 4. 选择`>`。



然后输入 `val2500` 和 `msgCounter is above 500`。

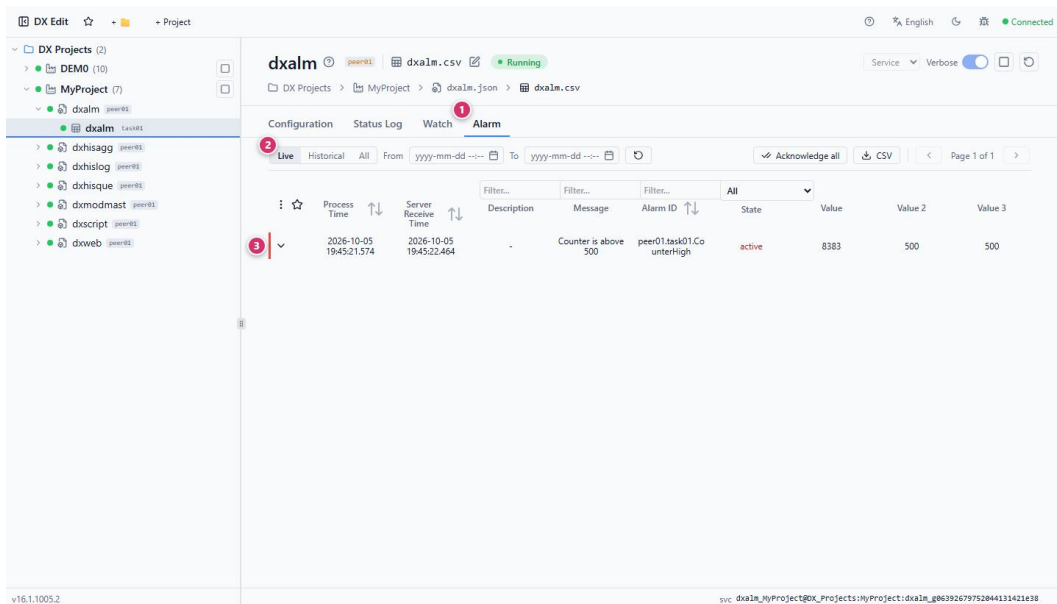
- 1. 单元格对其值来说太窄会将其截断。点击该行的编号可查看整行。
- 2. 行卡片显示每个值：对照上面的步骤检查它们。
- 3. 点击 **Apply**。



给报警一个自己的名称，例如 CounterHigh。如果它与设备标签（Counter）同名，DX 会把两者视为一个标签。

然后点击页面顶部的 Alarm 标签。

1. Alarm 列出模块的报警。
2. Live 显示当前激活的报警。
3. CounterHigh 已激活，并显示触发它的值。



你的值已经在记录了。要同时保留报警历史（Historical 视图），打开 dxhislog，在 Tasks（+ item）下添加一个任务，并将其 mode 设为 alarm，然后 Apply。历史记录器指南说明了该任务的其他设置。

你现在拥有的

你拥有	在哪里查看
一个读取 Modbus 设备的项目	dxmodmast, Watch 标签
它在网页上的值，及其历史	http://localhost:81/ , Trending
针对某个值的报警	dxalm, Alarm 标签

要读取你自己的设备，更改任务的 URL 和各行以与之匹配。

下一步

- **Read your own device:** 更改任务的 URL 和 Station ID，以及各行以匹配其地址。见其驱动的指南，例如 Modbus 主站。
- **Add more devices:** 每个设备一个任务，或从 Add modules 添加另一个驱动。对于 OPC UA、BACnet 或 IEC 104 设备，Browse addresses 列出设备提供的内容（见编辑）。
- **Rename 模块或项目:** 见重命名模块和重命名项目。
- **Stop, start or delete a project:** 见管理项目。
- 模块正常工作后 Turn Verbose off: 在项目页面上，模块停止时操作。Verbose 会向 Status Log 写入详细行。
- 测试时 Run a module as Console 以在其自己的窗口中查看: 在项目页面上，模块停止时通过 Run as 设置。
- 在 CSV 视图（网格下方的 CSV 按钮）中，或在电子表格中 Edit many rows at once。注意：电子表格在打开时可能会锁定文件，并且保存时可能更改 TRUE/FALSE 或分隔符。
- 在 dxweb 页面的 Web 标签上 Set up the web pages（连接、菜单、文件）：见网页。
- 在其任务设置中 Give dxweb a fixed Port，使其页面始终具有相同地址。
- 在项目页面的 Configuration 标签上 Change the broker or the shared module settings。

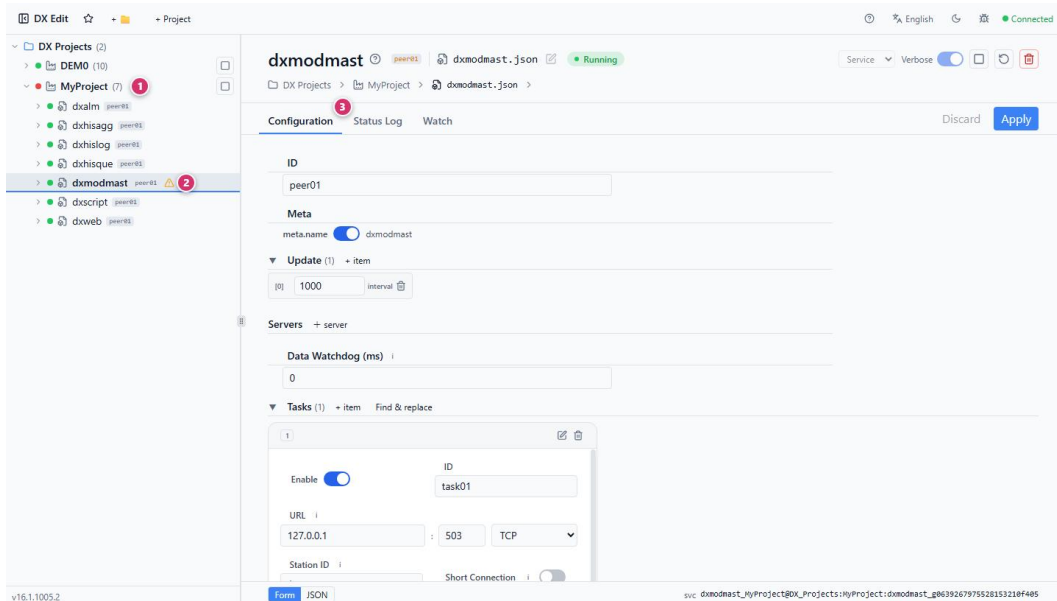
- 在 Alarm 标签上用 Acknowledge allAcknowledge alarms。
- **Run without the 2-hour limit:** 每个模块的 Status Log 会指明其许可证请求文件（例如项目文件夹中的 dxmodmast.draft.lic）；提交它以使模块获得许可。
- 不再需要时从树中 Stop DEM0。MyProject 的模拟设备在 DEM0 中，因此 MyProject 的设备随后停止应答，其圆点变为红色。
- **Open DX Edit from another PC:** 见连接到服务器。

当出现异常时

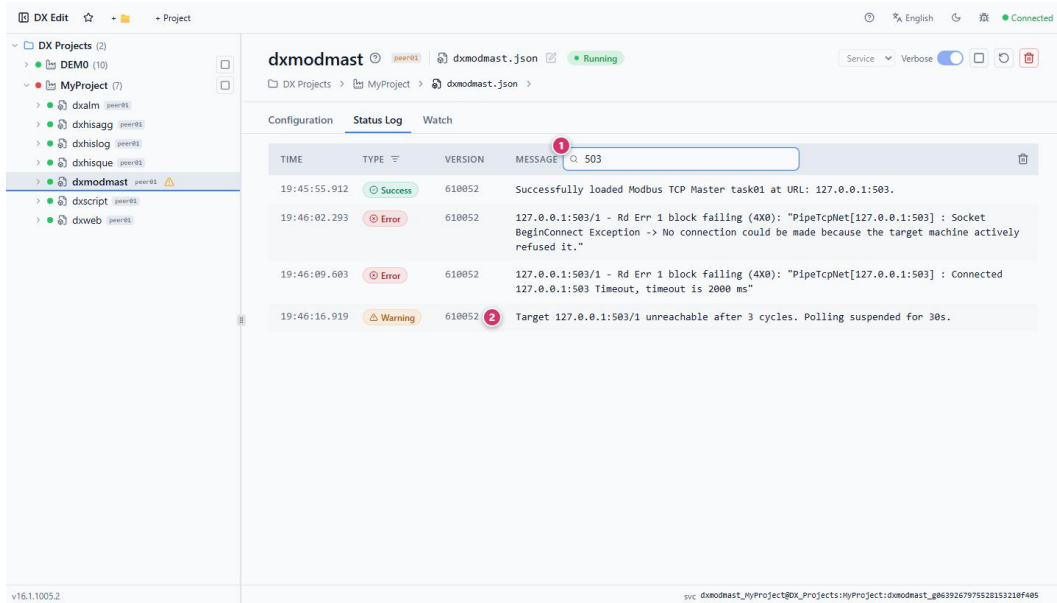
设备不应答

项目的圆点变为红色，驱动旁边出现一个标志：琥珀色为警告，红色为错误。模块本身仍显示 **Running**：失败的是它的任务，即设备（点击模块旁边的箭头可查看任务的红色圆点）。驱动无法连接设备：地址或端口错误、设备断电、或线缆或防火墙阻挡。

1. 项目的圆点为红色。
2. 驱动旁边的标志。
3. Status Log 标签说明了原因。



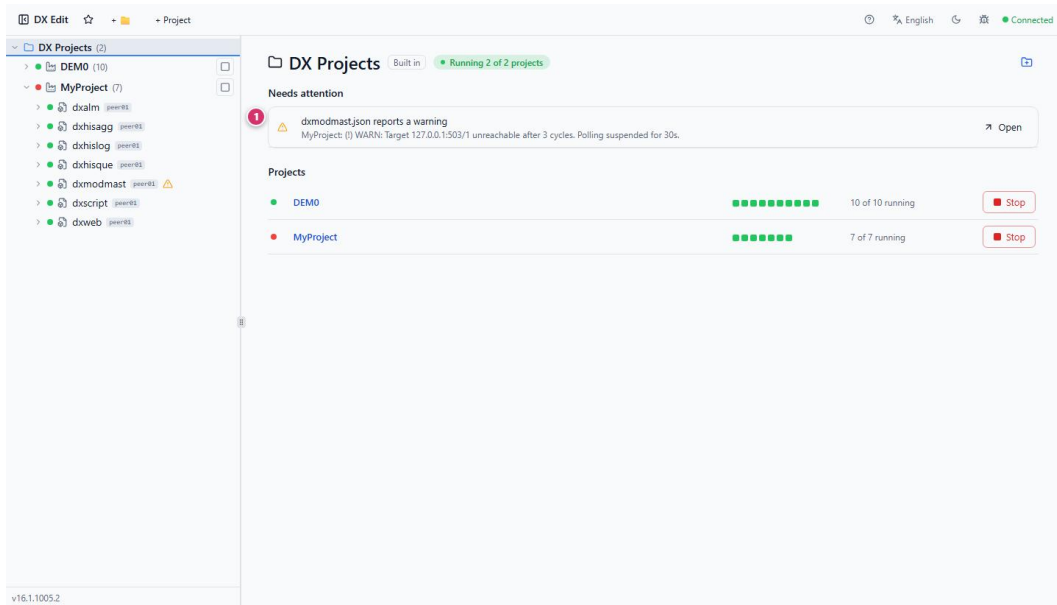
1. Filter messages: 输入你要查找内容的一部分，此处为端口 503。
2. 驱动自己的描述: 127.0.0.1:503 处的设备三次不可达，因此轮询在再次尝试前等待 30 秒。



检查任务的 URL、端口和 Station ID，然后 Apply。三次失败后驱动在再次尝试前等待 30 秒，因此你修复的设备可能需要长达半分钟才能变绿。

DX Projects 页面（在树中点击 DX Projects）列出你的项目中需要注意的所有事项：

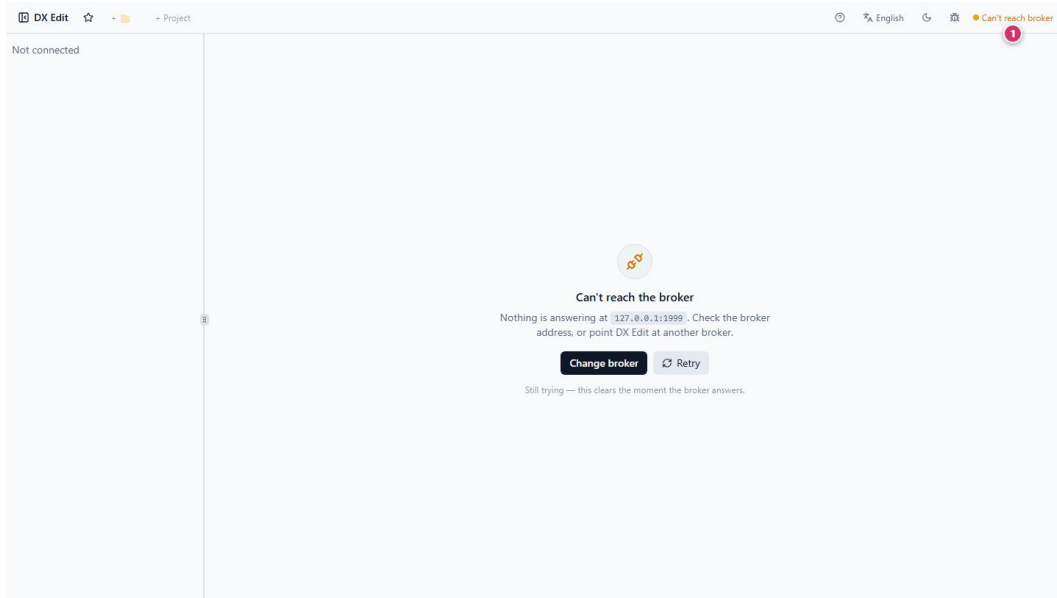
1. 问题，及其最新的状态日志行。Open 转到该模块。



DX Edit 无法连接代理

DX Edit 通过代理与 DX 通信。代理不应答时，页面会提示，并持续重试。

1. 右上角的徽章显示连接状态：Connected，或 Can't reach broker。

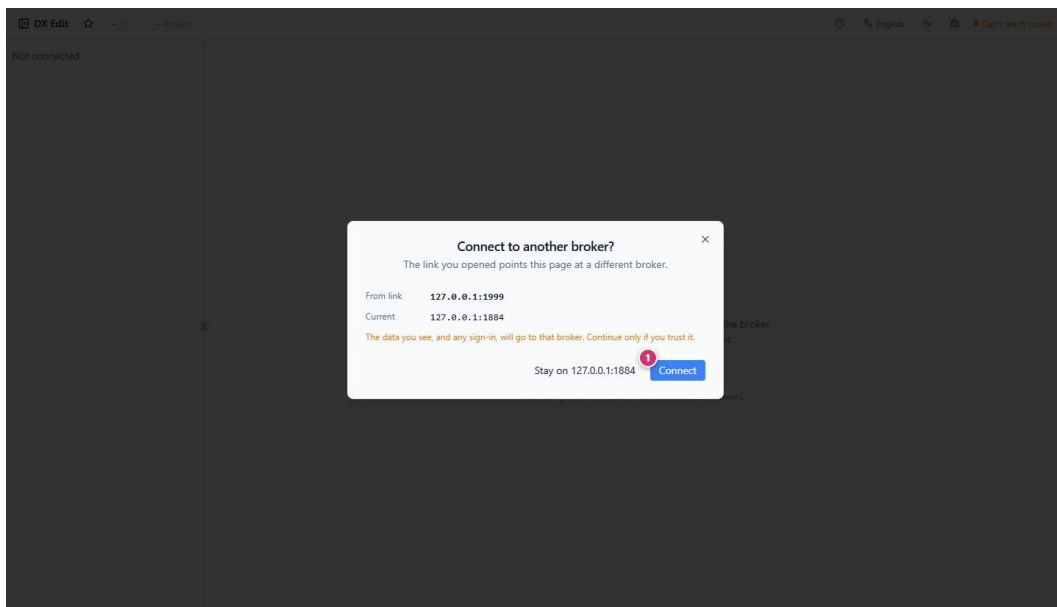


在运行 DX 的计算机上，检查 Mosquitto 服务是否运行（从开始菜单打开 Services 并查找 Mosquitto）。要使用其他服务器，点击 Change broker（见连接到服务器）。

链接询问“是否连接到另一个代理？”

将 DX Edit 指向另一个服务器的链接会先询问，因此链接永远不会把你带到你未选择的服务器：

1. Connect 使用链接中的服务器；另一个按钮保持在当前服务器。



树显示“等待模块索引”

DX Edit 已连接到代理，但 DX Run 尚未应答。刚安装或重启后这需要几秒钟。如果一直如此，在 Services 中启动 DX Runner 服务（DX Run 的 Windows 服务）。

Watch 标签缺失，或什么都不显示

Watch 仅在模块运行时存在：先启动它。如果它不显示值，检查模块的 Status Log：不应答的设备、或没有地址的行不会产生值。

报警从不出现

检查该行的 `val1`：它必须是 `${...}` 中标签的完整名称，与 `Watch` 中列出的完全一致，例如 `${peer01.task01.Counter}`，且其徽章必须显示为 `tag`。检查条件当前是否可能为真（例如值高于 `val2`）、该行的 `Enable` 是否勾选、以及你是否按了 `Apply`。
`Live` 仅显示当前激活的报警。

添加模块显示“无可可用模块”

模块列表来自 `DX Run`。关闭对话框，等待几秒后再重新打开。

树中缺少项目

左上角的星标在开启时仅显示你加星标的项目，且树显示 `No starred projects — the star filter is on`。再次点击星标可显示全部。

Start 呈灰色

模块运行所需的某项设置为空，例如驱动的 `URL`。指向 `Start`：其提示会说明缺哪项（`Configure first — needs: ...`）。填入它并 `Apply`。

圆点为琥珀色

模块的服务在运行，但模块在启动后 20 秒内未上报。打开其 `Status Log` 查看它在做什么。

单元格以橙色框出

某个必填值为空，或两个任务共用一个 `ID`。底部的栏会说明是哪项。你的更改仍被保留，`Apply` 仍可工作；缺少值的行在模块启动时会被跳过，其 `Status Log` 会说明这一点。

Apply 被拒绝

存在模块无法使用的值，例如格式错误的地址。消息会指明文件和每个问题，对于 `CSV` 还会指明行和列。修复它并再次 `Apply`。

“文件中也已更改”

在你有等待的更改时，其他人更改了同一文件，例如应用了他们的更改。你的值被保留并框出；指向它可查看文件中的值。再次 Apply 以写入你的值。见 When the file changes meanwhile。

重命名铅笔呈灰色

重命名模块会重建其服务，因此必须先停止项目：铅笔的提示显示 Stop the project to rename this file。项目可以在运行时重命名。

“请关闭其他程序.....”

项目的某个文件或文件夹在另一个程序中打开，例如文件资源管理器窗口或电子表格。关闭它再重试。

设置文件上的琥珀色断链标志

该文件中的模块类型未在此计算机上安装。点击该标志并选择正确的模块（见 Opening a module）。

一段时间后一切都停止了

没有许可证，每次启动运行 2 小时。再次按 Start。

某个模块显示红色的 Stopped

其服务已安装但未运行，例如许可证结束后。Start 再次运行它；它旁边的▲ 则改为移除服务，保留你的配置。

网页不显示值

打开 dxweb 页面的 Web 标签，然后 Connection，并点击 Check。从另一台计算机访问时，页面必须连到服务器地址（而非 127.0.0.1）上的代理，且防火墙必须放行 Web 端口和 1884 端口。

网页不在你预期的地址

80 端口被占用（被 DEM0 或其他程序），因此 dxweb 使用了下一个空闲端口。Verbose 开启时，其 Status Log 会列出它服务的地址，在以 ipAddr:开头的行中。给 dxweb 一个固定 Port 以保持单一地址。

每次项目启动都打开一个浏览器标签

dxweb 启动时会打开其页面。关闭标签；页面继续运行。

我的更改不见了

更改保存在此浏览器中，对应你打开 DX Edit 所用的地址。从开始菜单打开的 DX Edit 和从 Web 地址打开的 DX Edit（或 localhost 和 127.0.0.1）分别保存更改。按你之前的方式打开它。

值看起来错误或错位

在真实设备上，地址可能相差一个，或 32 位值可能需要另一种字节序。见 Modbus 主站指南中的 Offset 和 Data Format。

连接到服务器

从开始菜单打开的 DX Edit 是一个本地页面，连接到此计算机上的代理（1884 端口上的 Mosquitto websocket）。项目运行时，其 DX Web 服务器还在以下地址提供 DX Edit：

```
http(s)://<host>/dxedit.html
```

你可以复制 DX Edit 并将其托管在任何地方——本地、DX Tap 移动应用内、Internet 站点上、或你的局域网上私有部署。你只需将其指向你的代理。

如果尚未连接服务器，会显示一个 connect page：输入代理 URL 以连接。

可以按以下优先级顺序为你选择首选服务器 URL：

1. 在 DX Edit HTML 元素上设置的 URL（属性），
2. config.js 中的 URL，
3. 你在此浏览器标签中选择的服务器，
4. 提供该页面的同一域，
5. 以上均不可时，使用本地计算机（127.0.0.1）。

config.js 是放置在页面旁边的一个小型部署文件，一次性设置代理，因此每个链接都打开同一服务器，无需在地址中携带。

要选择另一台服务器，点击右上角的连接徽章，然后点击六边形图标 **Configure DX Edit connection**（或无法连接时的 **Change broker**），并在 **DX Edit Connection** 对话框中更改。页面会重新加载到该服务器，且该浏览器标签保持在其上，直到你再次更改或关闭标签。

你也可以通过链接在指定服务器上打开 DX Edit:

```
http(s)://<host>/dxedit.html?url=192.168.1.20:1884
```

如果此浏览器之前连接过该服务器，DX Edit 会直接连接。它未使用过的服务器会先询问：**Connect to another broker?**在当前服务器旁显示链接中的服务器。选择 **Connect** 以使用它，或 **Stay on ...**以保持当前服务器。无论哪种方式，地址中的 url 部分随后会被移除，刷新会保持你选择的服务器。链接从不会让你登录；你的身份始终来自你自己的登录。

启用模块以供编辑（DX Run）

模块仅在连接到运行中的 DX Run（本地或远程）时才出现在 DX Edit 中。对于边缘设备运行各自驱动的分布式站点，每个边缘设备也必须运行 **dx-run**，使其模块可被查看和编辑。

连接后，DX Edit 列出每个已连接的模块。它通过代理从运行中的 **dx-run** 获取每个模块的配置，模块以其可编辑内容应答。（控制谁可以获取和编辑的安全角色正在规划中——见 **Connecting to a server** 下的说明。）

界面

页眉右侧显示一个 **Connected** 指示器和一个 **+ Project** 按钮。编辑器自行保持最新——树、项目状态和服务状态随变化实时更新，因此无需手动刷新。当你选择模块文件时，会出现针对该模块的操作——**Discard, Apply, Start, Stop, Restart, Delete**。

（撤销、重做、导出和导入正在规划中——见表格下的说明。）

可折叠的 **left menu** 列出模块组。每个组先列出其 **JSON** 文件，然后是附加的 **CSV** 文件。列表按字母顺序排序——而非按更改状态——因此模块不会仅因为你编辑过而移动。点击 **title at the top-left** 可隐藏或显示整个左侧菜单——当你想要全宽显示

配置时很方便。左侧菜单底部显示已安装的 DX 版本，例如 v16.1.0925.3——与安装程序相同。当编辑器看到不同版本的服务器时，每个版本会与其服务器名称一起显示。

right pane 显示你所选内容的页面：文件夹组、项目、设置文件或 CSV 文件。未选择任何内容时，显示 **Start here**（见入门）。

你可以打开多个标签以同时编辑不同页面或分页。你的编辑保留在你自己的会话中——只有当你 **Apply** 时才对所有人可见。

书签和共享链接

浏览器地址始终指明你所在的页面——模块的文件及其标签，或项目页面及其标签——连同该标签的筛选条件，例如 **Alarm** 表格的页码。因此你可以：

- **Bookmark** 该页面，或 **refresh** 它，并回到同一页面。浏览器标签以该页面命名，例如 **DEM0 › dxalm.json · DX Edit**，因此你的书签有名称。
- **Share** 它：从浏览器地址栏复制地址并发送。打开它的同事会落在同一页面，左侧菜单展开到该位置。
- 用浏览器的 **Back** 和 **Forward** 逐步浏览你打开的页面。切换标签或筛选条件不会添加步骤。你离开页面时未应用的更改会被保留，因此不会有任何提示。

链接仅打开页面。它从不会保存、应用、启动或停止任何内容，也不会携带密码。

当链接指明.....	你会落在
项目中已不存在的模块文件	项目页面，并附带一条指明缺失文件的说明
此服务器没有的项目	什么都不打开；几秒后一条说明显示 The linked page isn't on this server
仅在模块运行时显示的标签，例如 Watch	模块运行时立即显示该标签；否则显示 Configuration
之后被重命名的项目	同一个项目——它通过其文件夹被找到

container page 也有自己的地址，例如 **dxedit.html?container=DX Projects**（标题为 **DX Projects · DX Edit**）；指向此服务器未列出的文件夹的链接显示 **The linked page isn't on this server**。指向模块类型的旧链接（以 **mod=** 结尾，例如 **mod=dxmodmast**）在项目有该类型模块时打开该模块的设置文件，否则打开项目页面。

每个按钮的作用

按钮	作用
+ Project	在你的 DX Projects 文件夹或你添加的文件夹中创建项目。它以停止状态创建。见 Managing projects 。
Discard	放弃你对屏幕上文件尚未应用的更改。文件恢复到服务器上的状态。
Apply	使你的更改生效。只有你更改的模块重启；其余继续运行。
Start	运行所选模块——默认作为 Windows service，或根据其模式作为 console 进程。
Stop	停止所选模块并移除其服务；它随后显示为 Idle。你仍可编辑它并 Apply；它启动时会使用该更改。
Restart	重启所选模块；停止的模块直接启动。
Delete	移除所选行——或者，在选中模块节点时，移除该模块（先停止/卸载，然后移除）。

在每个项目下，树按文件名以字母顺序列出模块的设置文件。树还带有你悬停时可访问的控件：project 提供+ Add module 和 Stop（停止的项目显示▶ Start）；设置文件提供 rename（铅笔）和×以移除它。点击项目名称可打开其 project page（见 [The project page](#)）。在树中打开设置文件可在其正下方列出其 CSV 文件。CSV 对每个使用它的任务列出一次——三个任务共用一个 CSV 会产生三行——每行显示该任务的 ID（保留默认 ID 的任务不显示）以及以圆点表示的该任务状态。打开一行会显示该任务所使用的 CSV——例如其标签名称带有该任务的 ID。

设置文件打开时，页面顶部显示 module type（例如 dxalm），旁边有一个打开模块用户指南的?，以及其 ID 徽章。随后是 file name 及其 rename（铅笔）、其状态（Running、Stopped 或 Idle，如下所述），以及运行控件（Service 或 Console、Verbose、Start、Stop、Restart、Delete）。CSV 页面则指明 CSV，并有自己的重命名。其下方，路径显示你所在的位置，例如 DX Projects › DEM0 › dxalm.json › dxalm.csv。模块运行的版本显示在其 Status Log 中。

路径的每个部分都会打开其页面，除了最后一个，即你所在的页面。每个›列出其前一部分中的内容：文件夹组中的项目、项目中的设置文件、或设置文件的 CSV

文件，当前项以框出。选择一个以打开它。在树中，点击 **folder group**（例如 **DX Projects**）会打开容器页面；行旁边的箭头仍仅用于折叠和展开。**Ungrouped Services** 没有页面。

文件名旁边的状态自行跟随模块的服务，因此你可以一眼看出它是否真正启动，而不仅是被要求启动：- **Running**：服务已启动。- **Stopped**（红色）：服务仍安装但未运行，这发生在许可证结束时。这是需要关注的状态。它旁边的▲ 一步移除服务。不会丢失任何内容：**Start** 再次创建服务，你的配置绝不会被改动。- **Idle**：模块没有服务。**Stop** 使模块保持 **Idle**，因为它停止并移除服务。

对于服务模式模块，其 **bottom of its editor** 底部的一个条带显示模块的 **service name**；点击它可复制名称。

编辑、应用和启动/停止/重启是日常流程，在下方的 **Your changes are kept**、**Apply** 和 **Launch** 中说明。

模块有两个部分

你打开的每个模块都显示相同的两个部分：

- **Settings**——JSON 文件中的主要配置。例如，驱动如何连接设备：其地址，以及读取频率。
- **Rows**——CSV 文件中的大量数据条目。例如，要读取的值：你选择的名称、设备上的地址和类型。

例如，**Getting started** 的两个值，正如 **dxmodmast** 行文件所保存的：

name	address	type	addressEx
Counter	0	int16	4
Batch	1	int16	4

每个 CSV 表格都有一个 **Grid | CSV** 切换（在 **JSON Form | JSON** 切换旁边）。**Grid** 是列编辑器；**CSV** 在可编辑框中显示纯.csv 文本——包括网格不显示的任何列——并以相同方式保存，与输入完全一致。切回 **Grid** 会重新读取你的文本；如果列不对齐，编辑器会警告你并保持在 **CSV**，以免任何内容错位。

每列上方有一个 **filter** 框。具有固定选项的列（例如数据 **type**）改用这些选项的下拉菜单进行筛选，外加用于空单元格的(empty)；空单元格算作该列的默认值。长表格会分页；用右上角的箭头在页面间移动。

在左上角，与筛选平齐，有两个小按钮，与 **Watch**、**Alarm** 和 **Audit** 表格上的相同：- **menu**（三个点）设置 **Rows per page** 并列出 **Visible Columns**。取消勾选列可隐藏它；再次勾选可恢复。必填列始终显示，且至少保留一列。- **star** 仅显示你加星标的行；再次点击可显示每一行。要给行加星标，指向它并点击其编号旁边的星标；再次点击可移除。加星标的行保留黄色星标。

要使列变宽或变窄，拖动其标题右侧的细线；双击该线可使列适应其内容。你也可以点击该线并使用←和→键。比页面宽的表格在其框内横向滚动，行号保持可见。**Enable** 保持其大小。一旦你更改宽度，页面箭头旁会出现一个 **reset** 按钮；它将表格恢复原样。

宽度、隐藏列和每页行数在此浏览器中为同一模块类型的每个 **CSV** 保存；星标为每个 **CSV** 文件保存。它们都不会更改文件。当窗口大小改变（例如最大化或还原），或你隐藏或显示左侧菜单时，列宽恢复到其自身布局。

要排序行，点击你输入内容的列标题，例如 **Tag Name**：- 第一次点击按 **A** 到 **Z** 排序，第二次 **Z** 到 **A**，第三次回到文件自身顺序。- 值中的数字按计数顺序排序（**Tag2** 在 **Tag10** 之前），大写字母无影响，空单元格保持在底部。- 排序仅改变你所见。文件保持其行顺序，每行保持其编号（#）。- 你编辑的行保持原位，直到你再次点击标题。- **Enable** 以及具有固定选项的列（例如数据 **type**）不排序。

对列来说太长的值会在单元格中截断。要查看整行，点击其编号（#）：一个卡片在行下方打开，显示其名称、是否启用、以及列自身标题下的每个已填值。它打开时，会随你的鼠标指针左右滑动，保持在该行下方。你编辑该行时卡片保持并跟随你的输入。点击行外、再次点击编号、或按 **Esc** 可关闭它。

在 **dxscriptCSV** 中，每行要么是 **file**（脚本文件），要么是 **expression**，在 **type** 列中显示。每个 **name** 旁边的图标打开该行的表达式或脚本文件；在文件行上，它旁边的▼ 列出文件夹中的脚本文件。你在脚本中输入的内容像任何更改一样被保存，行

名称旁有一个黄色圆点。页面上的 **Apply** 将其与 CSV 一起写入，运行中的 **dxscript** 会获取它；停止的 **dxscript** 在启动时使用它。在脚本窗口中，**Discard** 放弃脚本的更改，**Done** 关闭它并保留更改。在表达式中，你输入的内容直接进入该行；**Cancel** 恢复你打开它时该表达式的内容。新脚本文件仅在你输入内容后才创建，输入的文件名不带 **.js** 会补上 **.js**。要添加行，点击 **+ Add**，选择 **file**（默认）或其旁边的 **expression** 以及你想要的行数。与每个 CSV 一样，一个空行（**file** 行）在底部等待你输入；它仅在你填写后才保存。

管理项目

项目是容纳一组模块并一起运行它们的文件夹。**DX Edit** 为你管理项目；你无需任何控制台。

- **Create**——点击 **+ Project**，给它一个名称，并选择位置：你的 **DX Projects** 文件夹（默认）、**IntegraXor** 项目文件夹、或你添加的文件夹。不可使用的名称会在名称框正下方说明。项目以 **stopped** 状态创建，连接到 **127.0.0.1:1883** 的代理（之后可在其项目页面的 **Configuration** 标签更改），其项目页面打开，准备好 **Add modules**。从容器页面，**New project** 仅询问名称并在该文件夹中创建项目。新项目的 **web** 文件夹以示例页面开始：**Sample**（带动画图形的概览）、**Input** 和一个菜单（**js/menu.js**），该菜单还链接报警、审计和趋势页面。其 **js/config.js** 已带有新项目的名称，站点打开概览页面（在 **dxweb** 页面的 **Web** 标签、**Files > Start page** 下更改）。添加一个 **dxweb** 模块以提供这些页面。
- **Add an existing folder**——点击 **Add-path** 按钮在 **DX Run** 主机上打开文件夹选择器。逐级浏览本地驱动器（系统和被阻止的文件夹从不显示）：顶部的面包屑显示你的位置——点击一段可返回上级。点击文件夹以选择，或双击打开；要跳转到已知路径（包括 **\\server\share**），点击搜索图标并输入。**Add folder** 注册所选文件夹 **without starting it**，并将其位置添加到 **DX Run** 的允许列表，以便你管理它（并接纳在那里发现的其他服务）。容纳模块设置文件（**.json**）及其 CSV 的文件夹作为 **single project** 添加。任何其他文件夹，即使是空的，作为 **folder group** 添加，与你的 **DX Projects** 文件夹完全一样：其中的每个文件夹都是一个项目，无论是否为空，在树中嵌套于其下。**Block** 在以后浏览时隐藏所

选文件夹——及其下的一切；你需先确认，且只能通过主机上编辑 DX Run 的配置来撤销。系统保护的文件夹始终被拒绝，并显示原因。名称包含+或#的文件夹也被拒绝：其名称成为项目或容器名称，而这些字符不允许出现在那里。先重命名文件夹。

- **Stop**——悬停一个运行中的项目并点击 **Stop**，然后用 **Stop** 确认（或按 **Enter**）。它及其所有模块服务停止；项目的配置保留在磁盘上。
- **Start**——停止的项目在树中保持灰色显示，带有▶ **Start**。点击它可再次运行项目。
- **Browse**——在树中右键单击项目或文件夹组并选择 **Browse**，以在文件资源管理器中打开该文件夹。窗口在 DX Run 计算机上打开，对在那里登录的用户可见。这仅适用于 Windows 上的项目。
- **Remove or delist**——悬停一个项目并点击 trash 图标。对于你 added 的文件夹（DX Projects 文件夹之外的），你有两个选择：**Delist** 将其从列表移除但保留磁盘上的每个文件——它停止并卸载项目的服务，你可随时重新添加路径以恢复；**Remove entire folder** 永久删除文件夹及其所有配置，连同项目保存的草稿和备份，以及其保存在代理上的报警、审计、日志和报告历史（删除单个模块会保留其历史）。内置文件夹组（DX Projects 或 IntegraXor 项目文件夹）内的项目仅显示 **Remove entire folder**。两者都要求你输入项目名称以确认；一旦匹配，**Enter** 确认——对于添加的文件夹是 **Delist**（其文件保留），对于内置文件夹组内的项目是 **Remove entire folder**。added folder group（即使是空的）在容器节点上有自己的 trash 图标——移除它会一次取消列出整个容器及其项目，保留磁盘上的每个文件。它不先询问，因为没有删除任何内容。

DX Edit 通过单个后台 DX Run 服务管理每个项目——没有针对每个项目的管理器：

- **Project status** 来自 DX Run，它知道实际运行的内容——包括你在 DX Edit 之外启动或停止的服务（Windows 服务、sc、systemctl）或自行停止的服务，树在一秒内显示。项目、模块和模块 CSV 文件（使用它的任务）上的圆点表示：

圆点	含义
----	----

绿色	运行且正常。
红色	其服务已安装但未运行（例如许可证结束时停止）、它崩溃了、或其某个任务离线——需要注意。项目在运行但其任何模块或任务不是绿色时也显示红色，因此项目可能在每个模块都显示 Running 时仍为红色：展开模块以找到该任务。
琥珀色	其服务在运行，但模块在启动后 20 秒内未上报其已启动——检查其 Status Log 。项目在其任何模块为琥珀色时变为琥珀色（红色仍优先）。项目页面将其显示为 Not reporting 。
灰色	没有为其安装服务，或它根本未运行。项目仅在其所有模块都为灰色时才为灰色。

- 因此，运行中的项目只有在其中每个模块都为绿色时才为绿色。
- **Editing works running or stopped**——DX Run 始终提供磁盘上的配置；对运行中的模块 **Apply** 会自动重新加载它。
- **Projects in any folder.**你可以在 DX Projects 文件夹之外创建或打开项目；其位置被记住，DX Run 仅访问你已允许的文件夹。
- **Every folder in a folder group is a project.**你在 DX Projects 内或任何文件夹组内创建的文件夹，在几秒内作为新的空项目显示在树中，准备好 **Add modules**。隐藏文件夹、以及名称以点开头的文件夹会被跳过。
- **IntegraXor project folders.**在 Windows 上，当你的 Documents 文件夹还容纳 IntegraXor Projects 或 IntegraXor.CN Projects 时，每个都作为内置文件夹组列在 DX Projects 旁边，其中的每个文件夹都是一个项目。与 DX Projects 一样，它不能从列表中移除。不存在的文件夹不列出，创建后会显示。DX Projects 对+ Project 保持默认。
- **Adopt a stray service.**没有自己项目的 DX 服务——手动安装的、或升级后遗留的——在树底部的 **Ungrouped Services** 下灰色显示。如果其配置仍在磁盘上，

点击 **Adopt** 将其文件夹转为你可编辑和运行的完整项目；未分组的行消失，一个正常的、停止的项目取代它。如果服务的文件夹在你的 **DX Projects** 区域 **outside**，**Adopt** 会要求你确认——它显示确切文件夹和将添加到 **DX Run** 允许列表的位置——然后一步接纳它。系统保护的文件夹始终被拒绝。

项目页面

在树中点击项目以打开其页面。顶部显示项目名称及其状态：**Running**（带有其多少个模块）、在任何模块的服务仍安装但未运行时为 **Stopped**、或 **Idle**。每当任何模块的服务停止（已安装但未运行），状态旁的▲ 移除那些停止的服务。这即使在其他模块运行时也有效，运行中的模块不受影响。其下方是项目位置：其文件夹组和文件夹，例如 **DX Projects > DEM0**；点击 **DX Projects** 可打开该文件夹组的页面。右侧的图标 **start** 或 **stop** 项目并 **delete** 它；名称旁的铅笔重命名项目（见 **Renaming a project**）。页面有两个标签，**Modules** 和 **Configuration**，刷新会带你回到同一页面和标签。

Modules 列出项目中的每个模块，因此即使项目在树中折叠，你也可以访问它们：

列	显示内容
Module	模块类型，当 ID 与文件名不同时带有其 ID 徽章。
Run as	Service 或 Console 。
Verbose	详细日志记录的开关， on 或 off 。
Configuration	模块的设置文件（.json）。点击它可显示其 CSV 文件及其 rename 和 open 按钮；再次点击可隐藏。 Open （门）打开设置文件。
Status	与树中模块圆点相同的状态。右侧， status log 按钮在其 Status Log 标签上打开该模块，旁边是 Start （▶）或 Stop （■）按钮。当模块显示为 Stopped 时，其旁的▲ 移除其服务；不会丢失任何内容， Start 再次创建服务。

在 **Configuration** 中点击设置文件以在其下方列出模块的 CSV 文件，每个使用它的 **task** 列出一行并带有该任务标记（设置文件旁的计数是 CSV 文件数）；它打开时其 **rename**（铅笔）和 **open**（门）按钮保持显示。在列表上方的框中输入可按名称

或任务 ID 查找文件；长列表会滚动。指向关闭的设置文件、或任何 CSV 文件，可显示其 **rename** 和 **open**；文件仅从其 **open** 按钮打开。**Rename** 直接在列表中工作：输入新名称并按 **Enter**（或✓）；**Esc**（或✕）、或点击其他地方，使其保持不变。重命名设置文件遵循与树中相同的规则（项目必须停止，没有等待应用的更改）；只要使用它的设置文件没有等待应用的更改，CSV 文件可随时重命名，且 **every** 使用它的设置文件会更新为指向新名称。

Run as 和 **Verbose** 仅在该模块停止时可更改；它们在下次启动时生效。在列上方，选择 **Module** 类型或输入 **Configuration** 文件名的一部分以筛选列表；**add modules** 图标位于该行右侧。一次显示十个模块；滚动查看其余。尚无模块的项目显示空表格及 **Add modules**。

要调整列大小，拖动其标题右侧的细线，或双击该线使列适应内容。表格始终填满其框；当它更宽时，横向滚动，**Module** 保持可见。一旦你更改宽度，**add modules** 旁会出现一个 **reset** 按钮。宽度在此浏览器中为每个项目保存，直到窗口大小改变。

Configuration 编辑项目的共享配置（**dx.json**，见 **Group configuration**）。在左侧选择一个代理以设置其 **Broker URL**、**QoS**、**Sign in** 和 **Security**（打开右侧的 **TLS** 开关以显示 **CA file** 和 **client certificate** 行）；**Add broker** 添加另一个，代理旁的垃圾桶（悬停时）移除它——项目始终保留至少一个。**Module defaults** 设置项目中每个模块使用的 **Module ID**、**Update** 列表和 **Watchdog**，除非其自身配置设置它们。左下角的 **Form | JSON** 切换到文件本身：编辑文本并 **Apply** 按输入原样写入。回到 **Form** 会先检查文本，如果无效则保持在 **JSON**。当你更改任何内容，黄色圆点标记 **Configuration** 标签，**Discard** 和 **Apply** 在右下角出现：**Apply** 用新设置重新加载运行中的模块，**Discard** 放弃你的更改。树中项目上的 **Edit project configuration** 图标直接打开此标签。

网页

dxweb 模块的页面有一个 Web 标签，在 Configuration 旁边，设置 dxweb 提供的页面。左侧三项设置它们：Connection、Menu 和 Files。无论模块是否运行，该标签都存在。当项目有多个 dxweb 时，每个的页面有自己的 Web 标签。

你在此处的更改像任何其他更改一样被保存（见 **Your changes are kept**）：在你点击 **Apply** 前服务器上什么都不变，**Discard** 放弃它们。更改等待时黄色圆点标记该项和 Web 标签。离开页面、打开另一个 dxweb 的页面、或刷新会保留它们，你上传的文件也被保留。**Apply** 更改或删除的每个文件在历史中保留其先前版本。对于 Connection 和 Menu，左下角的 Form | Code 切换到作为文本的文件。

Connection 设置你的页面如何访问实时数据（js/config.js）：操作员浏览器所访问的 Broker 主机和端口，以及 wss 以加密该连接。Check 以页面将使用的方式从你的浏览器连接，并说明代理是否应答。当你将主机留空时，页面使用提供它们的计算机上的代理；你仍可打开 wss 并应用它，该项随后显示为 **Serving host**（带有·wss）。端口需要主机：在你输入主机前，**Apply** 保持禁用并如此提示。

Menu 以表单形式编辑每个页面上的菜单（js/menu.js）：- **Everyone** 是登录前显示的菜单，以及没有自己菜单的任何角色的菜单。+ **Role** 为一个角色添加菜单。它开始时是 **Everyone** 菜单的副本，此后你可以仅为该角色自由更改。- 每个条目有一个标签、它打开的页面（你的某个页面、或内置页面如 dxalm.html）和一个图标。拖动条目、或使用箭头键，可重新排序。- **Add to** 选择+ **Entry** 和+ **Group** 添加的位置。点击组或条目也会选择它。- **Overview** 以操作员所见方式显示菜单。需要空间时折叠它。- 不在 web 文件夹中的页面会被标记，仅字母大小写不同的页面也是如此（Windows 能打开；Linux 不能）。- **Code** 以文本形式编辑文件。无法作为菜单读取的文件在那里打开。

Files 列出 web 文件夹（显示为文件夹组，项目，文件夹）以及每个文件的用途：其菜单条目、显示图形的页面、加载脚本的页面。按名称、或按 **Pages**、**Graphics** 或 **Scripts** 筛选。- **Start page** 是在站点自身地址打开的页面。选择一个会使 index.html 打开它。当 index.html 是你自己的页面时，会先询问，其内容在历史中

保留。 - **Upload** 添加页面、图形、图像、样式、脚本和字体，每个最大 8 MB。名不副实的文件（例如并非真正 PNG 的图像）会被拒绝。上传覆盖现有文件会先询问。 - **Rename** 是名称旁的铅笔。页面的菜单条目跟随新名称。页面仍引用的图形会被标记；页面不会为你更改。 - **Delete** 列出仍使用该文件的内容，还可移除其菜单条目或没有其他内容使用的图形。起始页在你选择另一个之前不能删除。 - **Download** 保存文件的副本。

视图跟随磁盘上的文件夹。在那里复制、重命名或删除的文件、或对 js/menu.js 的手动编辑，约一秒内显示，无需刷新。如果你有未应用更改的文件在磁盘上被更改，磁盘上的更改会添加到你的更改中（见 **When the file changes meanwhile**）。

当 dxweb 的 url 指向项目文件夹之外时，文件会被列出，但 DX Edit 无法更改它们。

容器页面

在树中点击文件夹组，例如 **DX Projects**，以打开其页面。它回答一个问题：这些项目中是否有任何异常？

顶部显示文件夹名称、它是 **Built in**（你的 **DX Projects** 文件夹或 **IntegraXor** 项目文件夹）还是 **Added folder**，以及其多少个项目在运行。右侧，**New project** 在此文件夹中创建项目（仅询问名称），添加的文件夹还有 **Remove from list**，将文件夹从列表移除并保留磁盘上的每个文件。

Needs attention 仅在有需要时出现。每行指明问题并带有修复方法：

你看到	怎么办
模块报告警告或错误，带有其最新状态行	Open 其设置文件
项目运行时某个模块被停止	Start 它
模块意外停止	Start 它
某个任务离线，例如不应答的设备	Open 其在该任务处的设置文件
模块的服务在运行，但模块未上报	打开其 Status Log
服务没有项目（它在树中的 Ungrouped Services 下）	Adopt 它，或 Remove service

Projects 列出文件夹中的每个项目：其圆点（与树中相同）、其名称（点击它以打开项目页面）、每个模块一个颜色与该模块圆点相同的小标记、其多少个模块在运行、以及一个 **Start** 或 **Stop** 按钮。没有模块的项目在你添加一些之前不能启动。

添加模块

模块始终属于一个项目，因此你从树中添加一个（悬停一个项目并点击+ **Add module**）或从项目页面（其 **Modules** 表格上方的 **add modules** 图标）添加。选择要添加的模块：

- **Module type**——从计算机上已安装的模块中选择。章节标题沿左侧排列，模块在右侧一个面板中。类型分为 **Driver** 和 **Application**；**Application** 分为 **Standard**（**Basic**、**Historian**）和 **Extended**（**Util**）。点击章节标题可跳转到其模块。滚动时，可见的章节会被标记，其左侧标题和右侧模块都标记。计算机上安装的任何其他模块——例如你自己团队构建的——出现在 **Util** 下，或在它声明自己的章节时出现在其自己的章节下。顶部的 **search box** 一次跨所有章节筛选，每个类型旁的?在新标签中打开该模块的用户指南，以便你在添加前阅读。
- 每张卡片右侧的+和-设置要添加的副本数；它们之间的数字显示数量。点击卡片也会添加一个。一个副本时-将模块从列表移除。
- **Sample**——一旦你添加一个有示例的模块，其名称旁会出现一个 **Sample** 开关。点亮时，模块创建后即可运行，已设置为与其他示例协同工作（见 **Sample modules**）。带斜杠变暗时，它从其默认配置创建。该选择适用于该类型的每个副本。点击开关上或周围不会添加另一个副本。
- **dxalm** 使用历史模块 **dxhislog**、**dxhisagg** 和 **dxhisque** 来保存其 **Alarm** 和 **Audit** 表格中的历史（**dxhislog** 一旦有报警任务就会保存报警历史；见历史记录器指南）。选择 **dxalm** 也会选择项目尚未拥有的每个历史模块的一个副本，以灯泡标记（指向它可查看原因）。项目已有的历史模块不会再次添加。你可以用-将建议的一个从列表移除。

对于尚无模块的项目，例如你刚创建的，此窗口打开时已选好 **dxweb**、**dxscript**、**dxhislog**、**dxhisagg** 和 **dxhisque**，且每个有示例的模块上 **Sample** 点亮，因此你可以

Add 并立即启动一个可运行的项目；Reset to defaults 恢复这些选择。当你向现有项目添加时，不选任何内容，你选择的模块开始时 Sample 变暗。

点击 Add。每个模块立即创建，以其类型命名；同一类型的第二个副本得到 _2，然后 _3（例如 dxmodmast.json，然后 dxmodmast_2.json）。无需确认。要给它另一个名称，重命名它（见 Renaming a module）。

新模块作为 Service（在后台，作为 Windows 服务）运行，Verbose 开启（详细日志）。在模块停止时于项目页面更改 Run as（Service 或 Console，它在桌面上自己的可见窗口——见下方 Launch）和 Verbose。驱动或报警模块——其标签是自己的——得到 Module IDpeer01（项目中同一类型的第二个模块得到 peer02，然后 peer03），其每个任务按位置得到 task01、task02.....；其标签名称以这些 ID 开头，因此两个驱动不会冲突。Web 服务器（dxweb）得到相同的 peer01/task01ID：它不列出标签，因此这些 ID 不改变名称。任何其他模块——历史记录器、日志记录器、脚本——保留 Module ID0 和任务 ID0，因为它列出的标签属于其他模块，必须保持其确切名称。作为 Sample 添加的模块得到 Module IDpeer01（然后 peer02、peer03 用于更多副本）和任务 IDtask01。当你想以其自身名义命名配置时，在其设置中更改 ID：ID 在其发布的数据中标识它，并在树中作为小徽章显示在它旁边。要引用模块或任务的状态，使用基于其文件名的名称（见 DX 驱动通用配置指南中的 General Status）；那些在你更改 ID 时不会改变。

添加模块会创建其 Settings（JSON），预填该模块的 full default configuration，其 Rows（CSV）已带有该模块的列，等待你调整。dxweb 得到它不读取的空 Rows：其菜单是其 web 文件夹中的 js/menu.js。作为 Sample 添加的模块改为得到其示例设置和行。它在树中出现在其项目下。要移除模块，选择它并点击 Delete（先停止并卸载；其 CSV 随它移除）。从文件夹删除模块的 JSON 文件以相同方式移除它：其服务被停止并移除，它离开项目的模块列表。你在自己控制台窗口中启动的模块不会被停止——你自己关闭那个窗口。JSON 文件缺失的模块从不会启动。

名称——项目、配置文件或模块 ID——只能使用字母、数字、点、连字符和下划线，已在使用的名称会被拒绝。

示例模块

示例协同工作：添加全部四个并启动项目，数据从脚本流出、进入历史、再输出到你的网页。

模块	其示例的作用
dxscript	每秒运行 <code>js/my.js</code> 。脚本发布演示标签，例如时间（ <code>my.time.ss</code> ）和随机值 <code>my.rand.tag00</code> 到 <code>my.rand.tag08</code> 。
dxhislog	每 10 秒将项目的值记录到 <code>data</code> 文件夹。其标签列表为空，因此它记录项目发布的每个标签；在其 CSV 中列出标签以仅记录那些。
dxhisagg	每分钟，计算其报告布局 <code>report/sample-report.html</code> 要求的值： <code>my.rand.tag00</code> 到 <code>my.rand.tag02</code> 的平均值、最大值和最小值。
dxhisque	将记录的历史和示例报告提供给你的网页。

`dxweb` 没有示例；它始终以其默认配置添加。

重命名模块

要重命名模块，悬停它并点击 `rename`（铅笔，在移除旁边），直接在树中输入新文件名并按 `Enter`（或✓）；`Esc`（或✕）、或点击其他地方，使其保持不变。输入时，框下方的卡片显示完整新名称——树很窄时很有用——或者，如果名称不可使用，说明原因。配置文件被重命名，其 `Windows` 服务或控制台进程被重建以匹配。因为重命名重建服务，**project must be stopped first**：项目运行时铅笔被禁用，提示“停止项目以重命名此文件”。重命名还需要模块 `no changes waiting to be applied`，因此先应用或放弃它们。重命名项目也一样。`CSV` 文件可以随时以相同方式从其在树中的行（或从模块的设置）重命名——每个使用它的设置文件会实时更新为新名称，因此无需停止。

重命名项目

要重命名项目，打开其页面并点击顶部名称旁的 **rename**（铅笔）。直接在那里输入新名称并按 **Enter**（或✓）；**Esc**（或✕）、或点击其他地方，使其保持不变。项目重命名是配置更改，因此它 **live** 应用——项目中每个模块在几秒内以新名称重新加载，你 **not** 需要先停止它。**Configuration** 标签有你未应用的更改时铅笔等待：先应用或放弃它们。如果同一文件夹组中的两个项目恰好用同名，每个会以其 **folder** 标记，以便你区分并对正确的一个操作。

当模块在代理上改用新名称时——你更改其 **Module ID** 或类型、重命名它、重命名项目、或更改项目共享的 **Module ID**——其旧名称下的实时值、状态和 **CSV** 副本会被移除，因此没有屏幕继续显示旧模块。其报警、审计和趋势历史会保留。

打开模块

当模块的 **config.json** 在项目文件夹中时，树按文件名以字母顺序在其项目下列出它；其 **CSV** 文件在它下方。当模块的 **ID differs from that file name** 时，**ID** 作为小 **badge** 显示在名称旁——以便你一眼区分实例。当 **ID** 与文件名相同、或未设置时，不出现徽章。模块类型显示在文件页面顶部。点击设置文件或 **CSV** 文件以打开它（见 **Getting started** 中的图片）。

模块的所有标签都会被解析并自动添加到 **DX** 标签列表。

当模块的类型 **not one installed on this machine**——例如模块名称输入错误的旧配置、或后来被重命名的模块——**DX Edit** 在树中其设置文件上及其页面顶部用一个琥珀色 **unlink** 图标标记它。点击该图标并从列表中选择正确的模块：配置被链接到它，其设置表单加载正确的字段，模块即可运行。在你选择前什么都不更改。这样的模块在链接前不能运行：在其上按 **Start** 会打开相同的选择器，**starting the whole project** 启动其他所有模块并警告你此模块被跳过——因此单个输入错误的模块从不会阻止项目其余部分运行。

编辑

添加模块后，**first** 设置网络（**URL**）以及它所连接外部系统的任何凭据。应用并启动后，模块连接到该系统，并在可能时生成其资源文件。

模块的 **tasks** 在 **Tasks** 下作为卡片并排显示。- 要在它们之间移动，使用 **Shift + mouse wheel** 或卡片下的条。普通滚轮滚动你指向的卡片，然后是页面。- 拖动卡片下的手柄使它们变高或变矮。**DX Edit** 记住该类型模块的高度；双击手柄可使卡片再次适应屏幕。- 有许多设置的任务以两列显示。- 卡片标题上的橙色圆点表示该任务中的某个字段需要值、与其他任务共用 ID、或文件中同时也已更改。

Browse addresses.对于 OPC UA 客户端、BACnet 或 IEC 104 模块，任务的 CSV 顶部有 **Browse addresses**。它列出模块在其服务器或设备上发现的地址（见该模块指南中的自动浏览）。筛选任何列、勾选你想要的行并选择 **Add selected**：它们作为更改添加到 CSV 末尾，像任何编辑一样。**Regenerate discovery** 重启模块以便再次发现；新文件写入时列表刷新。在尚未发现任何内容时，**Discover now** 启动或重启模块，完成时列表填入。

Serial or network.对于可以使用串口或网络的驱动（Modbus 主站、Modbus 从站、DNP3、BACnet、Host Link），任务 URL 旁的列表选择使用哪种。在串口和网络之间切换会清空另一种类型的 URL，因此驱动从不保留旧连接；表单保持打开时切回会恢复那个 URL。不会为你填写任何内容。空 URL 意味着网络，因此在串口模式下空端口像任何必填值一样以橙色框出。

BACnet devices.对于 BACnet 模块，任务的 URL 提供运行项目的计算机的网络接口作为 Local endpoint（或 All interfaces），以及从 BAC0 到 BACF 的 BACnet 端口。其 Device ID 按名称列出模块发现的设备：选择一个以填入其 ID，或选择 **Find devices** 以启动或重启模块并再次查找。30 秒内没有设备应答时，它会如此提示。列表需要仅由字母、数字、.、_和-组成的任务 ID；否则该字段会提示你更改 ID。对于 MS/TP 任务（COM 端口），设备 ID 是一个纯数字框（见 BACnet 指南）。

编辑器知道每个模块接受什么，因此你可以及早发现错误。

- **Required values.**留空的必填单元格或字段会得到一个 **orange outline**，底部栏中间显示 **Please fill in the required fields**。文件的整行也会被标记，在树中、项目页面上和面包屑列表中以橙色虚线框出。当文件位于一个关闭的 json 内时，那

个 json 在一个橙色芯片中显示其 CSV 计数。例如，一个 Modbus 标签需要其 Tag Name 和 Address。被禁用的行（未勾选 Enable）不被标记。勾选 Enable 时其框线出现。

- **Task IDs.**一个模块中两个任务具有相同 ID（大写字母视为相同）会在两个 ID 上得到橙色框，底部栏显示 **Give each task a different ID**。以任务命名的文件，例如其发现的地址，否则会被共享。**Apply** 仍可工作。缺少值从不会阻止你：你的更改被保留，**Apply** 像往常一样工作。**Start** 仅在模块运行所需的设置（例如驱动的 URL）为空时等待；其提示说明缺哪项。模块启动时，它跳过那一行并在其 **Status Log** 中如此说明（见通用指南）。
- **New rows.**新行开始时已填好模块常用的选项。Modbus 行开始为 int16 和 Holding register，BACnet 行开始为 bool 和 Analog Input，因此你只需输入其特有内容。你用+ **Add** 添加的行立即被框出。如果你从不填写，它会被丢弃。你完全留空的行根本不会写入。
- **Lists.**某些单元格从列表中选择。在 BACnet 中你选择 Object Type。在 IEC 104 中你选择 Type ID 的 Address，然后只输入其编号。
- **Values a module cannot use.**格式错误的地址、类型错误的 CSV 单元格、或不是选项之一的值会阻止那一个模块生效，直到你修复它。应用被拒绝时，消息指明文件和每个问题，对于 CSV 还有行和列。这样模块从不会带着它无法使用的值生效。

你的更改会被保留

没有 Save 按钮：你做的每个更改都立即保存。它一直保留，直到你 **Apply** 或 **Discard** 它——当你转到另一页面、刷新页面、关闭浏览器、或暂时失去连接。在你应用前什么都不生效；项目像以前一样继续运行。

一个小黄色圆点标记每个有尚未应用更改的文件——仅那个文件，以便你确切看出是哪一个：

- 在页面顶部的路径中，在已更改的设置文件或 CSV 文件上，以及列表中的那个文件上；

- 在树中，在已更改的文件上。隐藏已更改文件的 **closed** 行——文件夹组、项目、或 CSV 文件被折叠的设置文件——在其正左方显示圆点。打开该行，圆点移到文件本身；
- 在项目的设置文件列表中，方式相同：列表关闭时在旁，打开后在已更改的文件旁；
- 在项目的 **Configuration** 标签上，在你更改的每个代理上以及更改它们时在 **Module defaults** 上。

你的更改保存在此浏览器中，用 DX 服务器持有的密钥加密，因此无法仅从浏览器读取。它们属于此浏览器：另一个浏览器或计算机看不到它们。如果你清除浏览器数据，未应用的更改会丢失。

同一浏览器中两个以相同地址打开的 DX Edit 标签共享你的更改：你在一个中输入的内容立即在另一个中显示，带有其黄色圆点。在同一服务器的另一个地址（例如 localhost 而非 127.0.0.1）打开的标签保留自己的更改。

如果浏览器无法保存更改（例如其存储被阻止），页面顶部显示 **Changes not kept in this browser**，关闭页面时会先询问。在你离开前应用更改。

在字段中按 **Enter** 以接受你输入的内容，就像点击其他地方一样。按 **Esc** 将字段恢复为你点击进入时的内容。

当文件同时被更改

在你仍有等待的更改时，有人可能更改服务器上的文件，例如应用了他们的更改或手动编辑文件。DX Edit 将他们的更改添加到你的更改中，因此两者都不丢失：

- 对你未触及内容的更改传入。添加到 CSV 的行出现，你未编辑的单元格取得其新值。页面显示 **Updated with changes from the file** 直到你下次点击页面。
- 在你们都更改同一内容时（相同的 CSV 单元格、相同的设置、相同的脚本行），你的值被保留。它以橙色框出，指向它可显示文件中的值。页面显示 **Also changed in the file**。之后第一次 **Apply** 仅向你显示文件，以便你看到将要替换的内容；再次点击 **Apply** 以写入。

- 如果其模块还在时文件被删除，你的更改保留且页面如此提示：**Apply** 再次创建文件，**Discard** 放弃它。如果整个模块被删除，其更改被丢弃，一条消息指明文件。

应用

准备好后，点击 **Apply** 使你的更改生效。**Apply** 写入模块的设置文件及其 CSV 文件——无论你从其哪个页面按下——对于 **dxscript** 还写入其更改的脚本文件。只有你更改的模块获取新设置；其余继续运行。

更改 **only when you Apply** 生效。如果你开始编辑后服务器上模块的某个文件被更改，**Apply** 先将那个更改添加到你的更改中（见 **When the file changes meanwhile**）。在你们都更改同一内容时，什么都不写入：文件以你的值框出打开，下一次 **Apply** 一起写入模块的所有文件。

要放弃对屏幕上文件的更改，点击 **Discard**。其他文件保留其更改。

想法很简单：**edit as much as you like, Apply once when you are ready.**

历史

每次你应用，该版本都会保留（写入带时间戳的快照，每次应用都会记录）。

启动

你可以 **Start** 模块；它以其 **Run as** 模式（在模块停止时于项目页面设置）运行——后台的 **Windows service**（默认），或桌面上其自己可见窗口中的 **console** 进程（即使项目本身作为后台服务运行）。模块一旦启动就运行，因此为谨慎启动，你可以在交付时将其配置部分 **disabled**，准备好后再手动逐个启用。（一次启动整个文件夹组正在规划中。）

模块仅在其 **DX Run** 启动时可编辑。如果你 **StopDX Run**，其节点立即变为红色（离线）：你仍可查看该配置，但在它恢复在线前无法应用。（将离线配置导出到文件正在规划中。）

组配置

项目文件夹中的模块可共享一个 `dx.json`。模块对于它自己未设置的任何内容，回退到其共享值——`server connection`，对于驱动模块还有存储转发 `update` 列表。共享配置具有 `lower` 优先级：对于模块已定义的任何值它会被忽略。

项目的 `name`（其身份——它在树中如何显示、其模块如何寻址服务器）开始时是其文件夹的名称。用其项目页面上名称旁的铅笔更改它（见 [Renaming a project](#)）；文件夹从不会被重命名。`older project`——其模块在不同项目名称下各自带有相同连接——在 DX Run 第一次看到时自动统一：那个共享连接移入 `dx.json`，项目采用该名称。

嵌入式界面

你可以将 DX Edit 嵌入你自己的 HTML 页面。包含库和样式表，然后添加元素：

```
<link rel="stylesheet" href="dx.css">
<script src="dx.js"></script>

<dx-editor url="192.168.1.1:1884/mqtt" secured></dx-editor>
```

URL 遵循与 [Connecting to a server](#) 中相同的优先级，元素 `attribute` 最先。当代理使用 TLS 但 URL 是没有 `wss://` 方案的纯 IP 时添加 `secured`。它通过 URL 连接到代理；带凭据的登录正在规划中，凭据从不会放在页面中。

如果自动连接成功，显示编辑器界面；如果失败，显示连接页面以便你手动输入 URL。DX Edit 连接后你仍可更改 URL。

刷新和重新加载

完全重新加载会收集 `every` 已连接的模块，无论其重连状态、启动顺序、或中间是否有代理重启。

当 DX Edit 初始化时——或你按 `F5`/完全重新加载——它发布一个初始化请求，每个模块以重新发布其基本内容应答。你也可以在其专用主题上单独重新加载一个模块，而不打扰其他模块。离线模块可根据请求从编辑器移除。

第二部分 DX 工作原理

通用配置

DX 代表分布式扩展（Distributed Extension），它们是 SCADA IO 驱动，可在一台机器内、或跨平台的多台机器上独立运行，支持各种协议。因此在需要时，扩容或将 SCADA 服务器迁移到云端都非常方便。

符合 DX 规范的驱动也可运行在嵌入式环境中，因此你也可以要求将任何驱动随 Ecava D-IOT 9000 系列边缘设备一起交付。

配置

DX 驱动可用一个 JSON 文件配置，表格化的部分可扩展为多个 CSV 文件。它们可用任何纯文本编辑器、代码编辑器、JSON 编辑器编辑，CSV 文件则可方便地用任何电子表格编辑。

尽管不同驱动可能有不同设计，但它们的基本配置非常相似且简单。为使事情更简单，JSON 和 CSV 中的大多数参数都是可选的，并具有最优的默认设置和行为，因此你可以轻松上手。

```
{
  "servers": [{"url": "127.0.0.1:1883", "project": "DEM0"}],
  "id": "opc1",
  "desc": "Channel/Gateway/Group 1",
  "update": [
    "1000"
  ],
  "task": [
    {
      "id": "ua1",
      "desc": "OPC UA Device 1",
      "url": "opc.tcp://localhost:49321",
      "username": "",
      "password": "",
```

```

        "secureConnection": false,
        "subscription": true,
        "trigger": 1000,
        "maxPoll": 1000,
        "data": [
            { "$csv": "dxopcuaccli.csv" }
        ]
    }
]
}

```

OPC UA 客户端的典型 JSON 配置文件

ID 在整个配置中使用，在每个子项的命名约定中充当父级节。

servers

servers 节是模块与之交换数据的 SCADA 服务器（MQTT 代理）列表。仅当模块独立于服务器运行时才需要它。即使找不到服务器或配置错误，模块仍会启动并持续运行，它会不断重试每个服务器，直到建立连接。

你可以列出多个服务器。模块同时连接列表中的每个服务器，向所有服务器发布数据，并接受其中任何一个的命令或查询，因此这里没有主备之分。当某台服务器不可达时，模块会保存其数据，并在至少一台服务器重新连接后将其转发（存储转发）。要运行多个相互备份的模块实例，请参阅冗余一节。

每个服务器条目接受以下字段：

- **url** ——服务器主机和端口，写作 **host:port**，例如“127.0.0.1:1883”。如果省略端口，则使用 3881。当你完全省略 servers 节时，模块默认连接“127.0.0.1:1883”和“127.0.0.1:3881”。
- **project** ——数据发布时所用的项目名称；它必须与服务器的项目匹配。默认值为“DEM0”。
- **qos** ——MQTT 服务质量，取值 0、1 或 2。默认值为 1；超出范围的值回退为 1。
- **username** 和 **password** ——需要身份验证的服务器的凭据。不需要时省略它们。
- **ca** ——CA 证书。提供它会为连接启用 TLS 加密；不提供则连接为明文（不加密）。
- **cert** ——可选的 .pfx 格式客户端证书，用于双向 TLS，与 ca 一起使用。

- **sslVerify** ——设为 **true** 可根据配置的 **ca** 验证代理的 TLS 证书。默认值为 **false**，因此不验证证书（设置 **ca** 时连接仍会加密）。
- **lwt** ——这一崩溃检测的遗嘱（Last Will）现在是 **automatic** 的——无需设置。如果模块意外断开，代理会代表它发布一条故障消息，使消费者知道数据不再实时（见断连检测）。它默认开启，对只读任务则跳过；如果代理不接受遗嘱，模块会不带遗嘱连接并在其状态日志中记录这一点。
- **priority** ——设置任何大于零的值以启用跨模块实例的冗余（见冗余）。

task

“task”是模块相关配置的节列表。每个节应有其 **ID**，通常不同，但如果需要冗余也可以相同。如果模块要与设备或数据库系统等外部系统交互，此节应包含 **IP** 地址和连接信息。此节通常包含一个定时器参数，指定任务应多久执行一次。

注意： 在一个项目内，为相同数据地址、用相同设备 **IP** 配置冗余 **ID** 是没有意义的。

maxPoll

任何大于零的有效数字都会限制某个特定任务中，对外部资源进行并发轮询或更新的唯一标签的最大数量。初始化期间，当任务下的唯一标签总数超过所输入的限制时，将根据从上到下的优先级选择一组标签进行更新；如果某特定协议有最优逻辑，则按该逻辑处理。

初始化后，如果前端模块或任何其他模块发出查询，则新请求的唯一标签将按先进先出（FIFO）原则享有更高优先级。不再获得更新的旧标签将在详细（**verbose**）模式下记录到控制台。

trigger

“trigger”是一个用于存储触发器数组的参数，这些触发器将用于触发某个配置节。共有 3 种类型的触发器可配置。

1. 基于固定时间间隔触发
2. 基于使用 **cron** 语法的事件定时器触发
3. 基于标签触发——基于标签值的任何变化触发；基于时间戳的更新触发

要设置固定时间间隔的触发器，指定的时间间隔需以毫秒为单位。例如“1000”表示每 1 秒触发一次。

当该节完全不应被触发时，将“trigger”留空，或完全省略它。

cron 语法简单地由 6 个以空格分隔的节组成，顺序为“sec min hr day mon dow”。你可以在特定节的范围内输入任何数字，或在该节无关或应忽略时使用星号（*）。例如，在第 0 分钟的每小时触发应写作“0 0 * * * *”，每周日上午 9 点应写作“0 0 9 * * 1”（星期几一节中周日计为 1，周六计为 7）。

至于基于标签值变化和基于时间戳更新这两种触发选项，需要以 `${tag name}` 的格式指定一个有效的标签名，其中 `tag name` 表示将被监视并充当触发器的标签。基于标签值变化的触发器格式为 `${tag name} +/-`，而基于时间戳更新（即使值未变）的触发器格式为 `${tag name}`。

你也可以基于与标签值的比较来触发。当值等于某个数字时，写 `${tag name}=value` 触发；或使用运算符 `>`、`>=`、`<`、`<=` 和 `!=` 分别表示大于、大于等于、小于、小于等于和不等于。例如，`${tag name}>100` 会在值高于 100 的每次更新时触发。

触发器数组中的每一项都会被独立求值，因此列出多项意味着该节在满足其中 **any** 一项时触发（**OR**）。要匹配多个值中的任意一个，将它们写成独立的项，例如“`${tag name}=1`”、“`${tag name}=5`”、“`${tag name}=9`”。（单个条件只比较一个值；不支持以逗号分隔的值列表，如“1,5,9”。）

要让多个条件在 **same time** 同时成立（**AND**），将它们放在一个嵌套数组中。例如，`["${A}>1", "${B}<2"]` 仅在 A 高于 1 且 B 低于 2 同时成立时触发，而同样这两个条件写成独立的顶层项时，则在满足 **either** 一个时触发。

你可以在 `+/-` 后加一个数字，使标签触发器仅在值相对于上次触发它的值至少移动了该数量时才触发，而不是每次变化都触发。例如，`${tag name} +/-5` 会在值相对于上次触发值上下移动 5 或更多时触发。

一个数组中的每个公共触发器项都有自己独立、分开工作的异步线程。如果你需要将几个任务分组以受控顺序触发，或在到目标网络的单个连接内工作，则可以在间隔触发器后用点分隔符加一个标签。

例如，你可以通过输入“1000.grpA”将几个（原本以 1000 触发的）任务分组，其中你可以使用任意 26 个英文字母加下划线作为无空格的字符串来命名任何组。注意，同一组必须由相同的间隔定时器触发，不同的定时器会被分开分组。例如，“1500.abc”和“2000.abc”尽管标签相同，也不会被分到一组。

当所连接的设备或网关只能支持有限连接，或你想对一组设备执行菊花链轮询机制时，这种分组触发器很有用，这样它们会以受控顺序被轮询，而不是一次全部轮询。

注意 分组触发器语法必须作为字符串加引号，否则不加引号的语法可能破坏 JSON 文件格式，因为像“1000.A”这样的分组语法不再是纯数字。

update

“Update”是数据更新顺序的总控制，用于根据网络架构或整个设置实现最佳性能。

“update”参数的典型用途是实现存储转发，你可以借此控制何时更新模块的数据。

例如，你可以累积数据，按每小时甚至每天而非需要频繁发送数据的实时方式更新。再如，如果你有大量数据需要按毫秒级更新，对于历史记录器等数据消费者来说，立即处理可能并非最佳。因此最好将数据累积成更大的间隔（如每秒或每 5 秒）再更新。

没有新数据可用时，不会发送任何数据。

当你不需要它时，将“update”留空，或完全省略它。模块随后完全不保留更新计划。

注意：最小更新闻隔为 500 毫秒。如果你输入更小的数字，它会自动提高到 500。这仅适用于数字间隔；cron 和基于标签的更新触发器不受影响。

watchdog

在顶层（与 **update** 并列），你可以设置一个以毫秒为单位超时的 **watchdog**。如果每个任务停止产生新数据的时间超过此周期，模块会自行重启以恢复。它默认禁用；设置诸如 **3600000**（1 小时）这样的值可启用它，或省略它以保持关闭。

data

尽管“**data**”节嵌套在配置下，但它可能是所有节中被编辑得最多的。此节通常包含表格格式的长列表条目，这就是为什么你可以将它链接到一个或多个外部 **CSV** 文件。

为方便起见，**OPC UA 客户端**、**BACnet**、**IEC 104** 或 **REST API** 等模块会把其关联服务器中找到的所有标签生成到一个以 **.all.csv** 结尾的文件中，以加快标签编辑工作。每个模块的指南会给出文件名。

注意：基本支持的数据类型有 **boolean**、**int8**、**uint8**、**int16**、**uint16**、**int32**、**uint32**、**real32**、**int64**、**uint64**、**real64** 和 **string**。可为相应的 **DX** 模块添加协议特定的数据类型。

重要：CSV 文件必须与 **JSON** 配置文件放在同一文件夹中。

提示：同一个 **CSV** 文件可以在不同 **ID** 下的几个配置中被多次链接。

注意：你可以通过将模块或任务节中的 **ID** 设为零（0）来省略标签名中的 **ID** 部分，但这会增加因疏忽导致网络中标签名冲突的可能性，尤其是单个 **CSV** 文件被多次链接时。

role

任务可以选择指定一个 **role**。设置角色后，任务会将其数据发布到该角色作为目的地，而不是项目名称，并且它只接受发往同一角色的传入写入或标签触发。将其留空（默认）以按项目名称发布并接受来自项目的数据。当你想将某个任务的数据隔离给特定消费者或操作员组时，使用角色。

角色是一个可见性分区：角色任务产生的一切（实时值、报警、审计、日志、报告、离线默认值）仅发布到该角色的目的地，而项目（无角色）数据仅发布到项目目的地。一个目的地的订阅者绝不会收到另一个目的地的数据。唯一的例外是崩溃

通知（**onfail**）：它始终发送到项目目的地，包含每个任务的标签名和默认值，因为模块只能向代理注册一个这样的通知。角色名必须是单个普通名称；它不能包含 /、\、..、+、# 或字符：* ? " < > |。当你更改或移除任务的角色并应用时，先前角色的实时值会自动清除；其报警、审计、日志和报告历史会保留。

通用状态

DX 模块提供简化的布尔错误状态，默认的 **false** 值表示 **fail** 状态，**true** 值表示 **good** 状态。**fail** 状态可以是“断开连接”、“超时”或任何设备特定错误。而 **good** 状态仅表示已连接且没有任何问题。

通用状态将放在“**moduleId.taskId**”的标签名下。为兼容旧版本，当模块 ID 设为零（0）时，使用配置的 **meta id** 作为标识符；如果未设置 **meta id**，则使用 JSON 文件名。如果任务 ID 也为零（0），任务会按其位置索引放入数组，因此没有 **meta id**、文件名为“**dxfile**”时，第三个任务的标签名为“**dxfile[2]**”。如果只有模块 ID 为零而任务保留自己的 ID，则标签名变为“**identifier.taskId**”。

无论 ID 如何，每个任务的状态也始终以“**identifier[index]**”发布，其中 **identifier** 是 **meta id**（或未设置 **meta id** 时为 JSON 文件名），**index** 是任务从 0 开始的位置。模块自身是否启动以“**moduleName.filename**”发布，例如保存为 **dxhislog.json** 的 **dxhislog** 模块为“**dxhislog.dxhislog**”，当模块 ID 不为零时还以“**moduleName.moduleId**”发布。当另一个模块或画面引用某个状态时，使用这些名称：它们不会随你更改模块或任务 ID 而改变。

值初始化

oninit 主题将用于初始化和前端目的。每个标签只有在模块发布过某个值后才会被加入 **oninit** 负载，并在值改变时被修改。

默认值处理

在基于模块和任务 ID 组成的系统标签之外，任何受影响的标签都应按以下行为、根据 **default** 配置设置自己的值。你必须在 JSON 配置的任务级别提供 **default** 参数（即使没有值）才能启用此默认行为。

任务级 **default** 参数仅充当开关。当它存在且带有非空值时，每个受影响的标签在故障时被设为该标签在自己 CSV 行中定义的 **default** 值，而不是任务级的值。

如果任务级别本身不存在 **default** 参数，则任何错误时都会发送空指示符（**null-indicator**）。如果你像下面这样将 **null** 设为默认值本身，行为也相同：

```
“default”: null,
```

当你将 **default** 字段留空或填空字符串时，将使用最后一个有效值，这是持久化行为，故障时不会为受影响的标签发送新数据，而只会更新受影响任务的状态。

```
“default”: “”,
```

断连检测

如果模块意外断开（例如断电），MQTT 代理会自动代表它发布一条故障消息，使数据消费者知道这些值不再实时。

运行时

DX 驱动可在命令行根据其文件名连同配置文件路径启动。

```
dxname [./dxname.json] [-s=1]
```

其中 **dxname** 只是一个示例驱动文件名，你必须查看关联驱动的文件名以启动程序。如果配置文件与驱动同名，则可省略。例如，你可以用以下命令启动 Modbus 主站驱动。

```
dxmodmast
```

另一方面，如果你有不同的文件名（如 **dx123.json**），则需要像下面这样指定文件名：

```
dxmodmast dx123.json
```

如果 DX 模块支持服务模式，你也可以通过添加 **-s=1** 参数以服务模式部署模块，使模块能在 Windows 中自动启动。另一方面，在 Linux 中你需要使用 **systemctl .service** 单元文件来启用守护进程。

缺少必填值的行

模块启动时，某个必填值留空的 CSV 行会被跳过，例如没有地址的 Modbus 标签。其他行照常加载，模块继续运行。状态日志会指明文件、行和缺失的列，例如：

`dxalm.csv row 2: name, val1, cond1, val2 are missing, row skipped.`

行号是 DX Edit 在网格的#列中显示的编号。被禁用的行不检查。除 Enable 外什么都没填的行被视为空白并忽略。当一个文件有超过 50 个这样的行时，状态日志列出前 50 个，然后用一行说明还跳过了多少个。

填入该值并保存文件。模块会重新加载并拾取该行。

配置重新加载

当你编辑并保存模块的配置（其 .json 或链接的 .csv）时，模块会在几秒后自动重新加载——你无需自己停止再启动。如果你快速连续保存多次，只应用最终版本。

重载是 seamless 的：模块在后台启动新配置，只有在完全运行后才切换，因此已连接的客户端看不到中断。在短暂的切换期间，你可能会注意到该模块（在服务模式下）出现第二个服务条目；这是预期的，新配置接管后旧条目会自动移除。托管自己固定网络端口的模块——如 Web、Modbus 从站、OPC UA 服务器或 BACnet 模块——在切换时可能会短暂停顿，因为旧实例和新实例不能同时占用同一端口。

如果新配置 invalid——例如 JSON 语法错误、或无法到达服务器/代理——切换会中止。模块继续在先前配置上运行，并显示一条消息（在状态日志、控制台和 Windows 事件日志中）指明具体问题，例如“新实例启动失败.....原因：.....”。保存的文件完全保持你保存时的样子——检查并修复它，然后再次保存以重试。模块绝不会替你更改配置文件。

存储转发

当驱动与服务器之间出现通信故障时，存储转发会自动启用。借助存储转发你可以获得最佳更新时间，启用它可有效减少服务器负载和带宽占用。

你可以像“trigger”参数一样，用所需的毫秒间隔配置“update”参数。你也可以指定 cron 表达式，以更灵活地安排更新时间。

冗余

你可以运行同一模块的多个实例、使用相同配置，使它们相互备份。要启用此功能，在服务器条目上设置一个大于零的 **priority**。共享相同配置的运行实例随后会自动分配任务，如果某个实例停止，其余实例会接管其任务。任何大于零的值都会开启冗余；否则冗余关闭。

Ecava IGX SCADA 服务器

要集成任何 DX 驱动，IGX 需要启用 IO-hub 任务并按所需 IO 数量获得许可。你应将所有 *.json 和 *.csv 文件放在项目文件夹中。

当远程于 IGX 项目运行时，DX 项目名称必须与 IGX 项目名称或 ID 匹配。CSV 文件中列出的所有标签都会自动添加到 SCADA 虚拟标签中，因此你可以直接使用它们。链接到有效 JSON 文件的 CSV 文件中的任何条目都会自动添加到虚拟标签。此外，添加到 CSV 文件的新标签会自动更新到虚拟标签。如果你需要禁用自动更新，可以打开 `iohub-ref.csv` 并将最后更改日期改为很远的未来日期。

要手动将 DX 标签导入 IGX 项目的虚拟标签，只需转到 Tools（工具），然后选择 Import JSON file...（导入 JSON 文件...）。

当你使用 12.1 及以后版本的标准 IGX 安装程序安装时，DX 随 IGX 一起分发。你可以将 JSON 和 CSV 文件都放入项目文件夹，当存在 meta name 时，IO-hub 会将驱动作为服务自动启动。

注意：CSV 文件中的所有字符串都不能加双引号，以便 IGX Project Editor 能解析。

你可以使用 Project Editor 在 IO-Hub 下添加一个 DX 节点，然后编辑 JSON 参数。

示例

下面是 Modbus 主站模块配置文件 `dxmodmast.json` 的一个示例。

```
{
```

```
"servers": [{
    "url": "",
    "project": "",
}],
"id": "mbus",
"meta": {
    "name": "dxmodmast"
},

"task": [
    {
        "enable": true,
        "id": "stn1",
        "url": "127.0.0.1",
        "trigger": 1000,
        "watchdog": 5000,
        "stationId": 1,
        "readOnly": false,
        "data": [
            { "$csv": "dxmbusmast.csv" }
        ]
    },
    {
        "enable": true,
        "id": "stn2",
        "url": "127.0.0.1",
        "trigger": 2000,
        "watchdog": 8000,
        "stationId": 2,
        "readOnly": true,
        "data": [
            { "$csv": "dxmbusmast.csv" }
        ]
    }
]
}
```

以及关联的 CSV，注意同一个 CSV 文件被链接到两个不同的配置组。

name	desc	address	type	length	default	scale	addressEx
AO001	this is description	1	int16	-1	1	10	4
AO002	this is description	2	int16	-1	1	10	4
AO003	this is description	3	uint16	-1	1	1	4
AO005	this is description	5	int16	-1	1	100	4
AO006	this is description	6	uint16	-1	1	10	4
AO007	this is description	7	int16	-1	1	1	4

AO008	this is description	8	uint16	-1	1	4	
AO009	this is description	9	uint16	-1	1	1	4

在本例中，组 1 和组 2 的第一个标签分别为“gateway1.stn1.AO001”和“gateway1.stn2.AO001”。

如果你在一台机器上安装了 IGX，你可以使用这组 dxmodmast.json 和 dxmodmast.csv 示例文件，在启用 IO-hub 的情况下，将采集到的 Modbus 数据推送到“server”中指定的 IGX 服务器：“127.0.0.1:1883”。

然后你只需打开命令提示符控制台，在 JSON 和 CSV 文件所在的当前目录输入“dxmodmast”，即可启动 DX Modbus 驱动。

驱动

以下是特定于某协议的一些补充信息，否则你只需配置 URL 和 trigger 等基本参数。

Bacnet/IP

Bacnet/IP 不需要指定 URL，因为 BACnet 通过广播和多播消息使用 UDP 进行设备检测。UDP 的使用使设备发现的通信快速、轻量，使该过程高效，适用于楼宇自动化和控制系统。

deviceId 是分配给 BACnet 网络上每个设备的唯一标识符，用于将其与其他设备区分。在 BACnet 网络初始化期间，设备会广播它们的存在及其设备 ID。此过程使网络上的其他设备能够相互识别。设备发现对于网络建立通信路径至关重要。

subscription 参数用于设置客户端订阅 CSV 中标签的 COV（数值变化）请求。如果设为 false，则所有标签都将手动轮询。请注意，如果订阅失败，它会自动转换为手动轮询。

Below is the sample of configuration. “task”: [{ “enable”: true, “id”: “0”, “deviceId”: 10, “trigger”: [“1000”], “readOnly”: false, “subscription”: true, “maxRetry”: 3, “watchdog”: 1000, “data”: [{ “\$csv”: “dxbacnet-setting.csv” }] }]

objectType 列用于设置 BACnet 设备上对象的唯一对象类型标识符。

下面是 CSV 配置的示例

name	objectType	address	type	length	scale
indoorTemp	2	2	int32	-1	
waterTemp	2	1	int32	-1	

以下是对象类型参考： ANALOG_INPUT = 0, ANALOG_OUTPUT = 1, ANALOG_VALUE = 2, BINARY_INPUT = 3, BINARY_OUTPUT = 4, BINARY_VALUE = 5, CALENDAR = 6, COMMAND = 7, DEVICE = 8, EVENT_ENROLLMENT = 9, FILE = 10, GROUP = 11, LOOP = 12, MULTI_STATE_INPUT = 13, MULTI_STATE_OUTPUT = 14, NOTIFICATION_CLASS = 15, PROGRAM = 16, SCHEDULE = 17, AVERAGING = 18, MULTI_STATE_VALUE = 19, TRENDLOG = 20, LIFE_SAFETY_POINT = 21, LIFE_SAFETY_ZONE = 22, ACCUMULATOR = 23, PULSE_CONVERTER = 24, EVENT_LOG = 25, GLOBAL_GROUP = 26, TREND_LOG_MULTIPLE = 27, LOAD_CONTROL = 28, STRUCTURED_VIEW = 29, ACCESS_DOOR = 30, TIMER = 31, ACCESS_CREDENTIAL = 32, ACCESS_POINT = 33, ACCESS_RIGHTS = 34, ACCESS_USER = 35, ACCESS_ZONE = 36, CREDENTIAL_DATA_INPUT = 37, NETWORK_SECURITY = 38, BITSTRING_VALUE = 39, CHARACTERSTRING_VALUE = 40, DATE_PATTERN_VALUE = 41, DATE_VALUE = 42, DATETIME_PATTERN_VALUE = 43, DATETIME_VALUE = 44, INTEGER_VALUE = 45, LARGE_ANALOG_VALUE = 46, OCTETSTRING_VALUE = 47, POSITIVE_INTEGER_VALUE = 48, TIME_PATTERN_VALUE = 49, TIME_VALUE = 50, NOTIFICATION_FORWARDER = 51, ALERT_ENROLLMENT = 52, CHANNEL = 53, LIGHTING_OUTPUT = 54, BINARY_LIGHTING_OUTPUT = 55, PROPRIETARY_MIN = 128, PROPRIETARY_MAX = 1023, MAX_BACNET_OBJECT_TYPE = 1024, MAX_ASHRAE_OBJECT_TYPE = 56

Beckhoff ADS

Beckhoff ADS 驱动的主要配置是指定 URL 及其端口号，以及另一个默认值为 1 的参数“slot”。下面是示例配置。

```
“task”: [ { “enable”: 1, “url”: “127.0.0.1:851”, “slot”: “1”, “deviceName”: “Ads1”,
“trigger”: [ “1000” ], “timeout”: 3000, “readOnly”: false, “id”: “plc1”, “data”:
[ { “$csv”: “dxads.csv” } ] } ]
```

disable	name	description	address	type	length	default	scale
---------	------	-------------	---------	------	--------	---------	-------

```
“false”,“tag1”,“this is description”,“M100”,“boolean”,-1,““,1 “false”,“tag2”,“this is
description”,“M100”,“int”,-1,““,1 “false”,“tag3”,“this is description”,“M101”,“int”,-
1,““,1
```

Modbus Master

基本 Modbus 地址分为 4 类，即用于布尔输出的输出线圈（Output Coil）、用于布尔输入的状态输入（Status Input）、用于整数输出的保持寄存器（Holding Register）、以及最后用于整数输入的输入寄存器（Input register）。它们的前缀分别为 0x、1x、4x 和 3x，并被简化为单个数字 0、1、4 和 3，可填入 addressEx 列以指定地址类型。stationId 站 ID（Station ID）是 Modbus 设备在网络中的节点地址。

timeout 实际的超时值应小于 3 秒，除非在非常慢的网络中。

offset offset 参数用于调整数据地址，以确保主站和从站之间的索引不错位。要确认数据地址是否正确对齐，只需尝试轮询任何不在起始位置的已知值，即避免使用 4x0000 或 4x0001 以避免通信错误。可能的偏移值为 -1、0 或 1。

dataFormat real32、int32、uint32 等 32 位数据类型可能有不同的数据字节顺序，你可以通过输入 ABCD 或 CDAB，或以相反方式 DCBA 或 BADC 来控制映射。

OPC UA Client

OPC UA 驱动的配置应该很直接，因为 subscription 默认为 true，所以 trigger 无需手动轮询，主要参数是连接到 OPC UA 服务器的 URL。例如：“url”: “opc.tcp://127.0.0.1:49320”

一旦你看到 OPC 服务器已连接，还应注意会提示一条消息，说明将创建新的“*.all.csv”。请耐心等待，直到出现另一条完成消息，再继续停止驱动。

打开自动生成的 CSV 文件，你可以将感兴趣的标签复制并粘贴到你的工作 CSV 中。

注意：

9. 在旧的 *.all.csv 被移除之前，不会再次生成该文件。
10. 如果 OPC 服务器因设备离线而存在坏数据，则无法获取正确的数据类型，尽管这不会影响后续的轮询。
11. 在大多数情况下 trigger 参数应为空，除非你想在特定间隔执行额外的手动轮询。

Profinet

Profinet 提供了一个参数 model，用于区分不同型号西门子 PLC 使用的协议。下面是 model 的可用选项：

- S7-1200
- S7-300
- S7-400
- S7-1500
- S7-200Smart
- S7-200

对于 S7-300 和 S7-400，你需要配置 slot 和 rack 设置。对于 S7-200，你需要相应配置 localTSAP 和 destTSAP。

Below is the sample json configuration for Task section:

```
“task”: [ { “url”: “127.0.0.1:102”, “model”: “S7-200”, “localTSAP”: 0, “destTSAP”: 0, “rack”: 0, “slot”: 0, “timeout”: 3000, “trigger”: [1000], “id”: “plc1”, “data”: [ { “$csv”: “dxprofinet.csv” } ] } ]
```

EthernetIP

DX Allen Bradley EthernetIP 模块使你能够配置并连接到 Allen Bradley PLC 1756 与 1769、1747 和 Micro800 系列。

下面是设置到所需 Allen Bradley PLC 连接的参数。

deviceID 必填，你可以输入槽号。

deviceType 你可以输入设备类型。支持的选项有 1756/1769、1747、micro800、df1serial。

url 你可以输入 Allen Bradley PLC 所连接的 url。

trigger 你可以输入触发读/写操作的定时器。

name 你可以输入与 Allen Bradley PLC 关联的标签名。

address 你可以输入存储数据的内存地址。向不存在的地址写入或读取时将返回错误。

dataType 你可以输入与标签关联数据的数据类型。支持的数据类型包括 bool/boolean、byte、double、float、int16/short、uint16/ushort、int32/int、uint32/uint、int64/long、uint64/ulong 和 string。

length 你可以输入字符串的长度。对于其他数据类型，此参数不需要，会自动设为 0。

示例：配置文件（.json）

```
{
  "id": "dx",
  "task": [
    {
      "enable": true,
      "id": "device01",
      "deviceID": 1,
      "deviceType": "", //micro800; 1756/1769; 1747; df1serial
      "url": "127.0.0.1",
      "trigger": [ "1000" ],
      "readOnly": false,
      "data": [
        { "$csv": "tag.csv" }
      ]
    }
  ]
}
```

Example: CSV file (.csv) name,desc,address,dataType,length “test_bool”,“description 1”,“A”,“boolean”,0 “test_string”,“description 2”,“A1”,“string”,3 “test_int”,“description 3”,“B”,“int”,0 “test_array”,“description 4”,“C[5]”,“string”,3

SNMP

该任务的默认 url 为“127.0.0.1:163”，你应在每个任务的配置中将其指向外部设备。

地址字段应输入 OID（对象标识符），例如“1.3.6.1.2.1.1.1.0”，它对应“sysDescr”对象，通常关联的数据类型为“Octet String”。你应确保配置的地址对目标对象具有适当的访问权限，因为某些对象是只读的甚至完全无法读取。

下面是配置示例。

“task”: [{ “id”: “LJ1000”, “url”: “127.0.0.1:163”, “trigger”: [“1000”], “timeout”: 3000, “community”: “public”, “data”: [{ “\$csv”: “dxsnmp.csv” }] }]

下面是 dxsnmp.csv 的示例

name	description	address	type	length	scale
tag1	this is description	1.3.6.1.2.1.1.4.0	string	-1	1
tag2	this is description	1.3.6.1.2.1.1.5.0	string	-1	1
tag3	this is description	1.3.6.1.2.1.1.6.0	string	-1	1

NOTE 请确保 community（团体名）相同，否则无法建立通信。

TIPS 包含所链接设备所有地址的 *.all.csv 会自动生成，方便你复制任何所需地址。

Hostlink

示例：配置文件（.json）

```
{
  "id": "module01",
  "update": [ "1000" ],
  "task": [
    {
      "enable": true,
      "id": "HostLink01",
      "desc": "Omron HostLink driver",
      "url": "COM3",
      "trigger": [
        "1000"
      ],
      "baudRate": 9600,
      "dataBits": 7,
```

```
        "stopBits": 2.0,
        "parity": "e",
        "dataFormat": "CDAB",
        "unitNum": 0,
        "timeout": 5000,
        "readOnly": false,
        "watchdog": 5000,
        "data": [
            {
                "$csv": "dxhostlink.csv"
            }
        ]
    }
}
```

下面是设置任务的参数。如果未指定参数，在有默认值时将使用默认值。

URL 此参数用于通过串口或 IP 地址连接 PLC。如果使用 IP 地址，将使用 TCP 上的 hostlink。

baudRate 当前连接串口信息的波特率。默认值为 9600。

dataBits 每字节数据位的长度。默认值为 7。

stopBits 每字节停止位的数量。默认值为 2.0。

parity 奇偶校验协议。默认值为 even（偶校验）。接受的格式如下：“o”/“odd”：奇校验 “e”/“even”：偶校验 “s”/“space”：空格校验 “m”/“mark”：标记校验 “n”/“none”：无校验

dataFormat 多字节数据的解析格式。接受的格式包括“BADC”、“CDAB”、“DCBA”和“ABCD”。默认值为 ABCD 格式。

unitNum PLC 的单元号。默认值为 0。

type 可能的类型为“standard”和“cmode”。默认类型为 standard。

mode 当 PLC 通过 cmode 连接时，连接的模式可设为“Run”、“Program”或“Monitor”。默认模式为“Run”。

CSV 配置示例

name	address	type	length	scale
CIO	CIO200	int16	-1	1

CIO.0	CIO200.0	bool	-1	1
-------	----------	------	----	---

standard 模式支持的地址。standard 模式支持位和字轮询。

cmode 支持的地址。C-mode 仅支持字轮询。

Memobus

Memobus 驱动已针对 Yaskawa（安川）MP3300 CPU301 16Axes（NSC-732）的默认行为预配置，对于其他品牌或型号你可能需要稍作调整。

第三部分 平台服务

DX Manager

DX Manager（`dxmanager`）为 DIOT 项目启动并管理其他 DX 模块任务。它还内置一个 MQTT 代理，供其他 DX 模块连接。你可以借助管理器从单一入口启动所有 DX 模块，并可为代理选择启用身份验证和 TLS 加密。

New projects no longer include DX Manager.始终运行的 DX Run 服务现在负责启动并管理每个项目的模块，因此新建项目默认不含 DX Manager。DX Manager 仍受完全支持——如果你想要按项目的管理器，可手动向项目添加 `dxmanager.json`。

关于 `servers`、`project`、模块 ID、`trigger`、`update`、`data`、存储转发和运行时等通用 DX 配置，请参阅 DX 驱动通用配置指南。

配置

下面是一个典型配置文件 `dxmanager.json` 的示例。

```
{
  "servers": [{"url": "127.0.0.1:1883"}],
  "project": "DEM0",
  "id": "module01",
  "meta": {
    "name": "dxmanager"
  },
  "task": [
    {
      "enable": true,
      "id": "manager01",
      "desc": "DX Manager",
      "tcpip": 3881,
      "websocket": 4881,
      "pwdFile": "",
      "cert": "",
      "mode": "service",
```

```
        "data": [
            { "$csv": "dxmanager.csv" }
        ]
    }
}
```

tcpip

内置 MQTT 代理的 TCP/IP 端口号。通过原始 TCP 连接的客户端使用此端口。默认值为 3881。取值必须在 1 到 65535 之间。

websocket

内置 MQ 代理的 WebSocket 端口号。通过 WebSocket 连接的客户端（通常是 Web 浏览器）使用此端口。默认值为 4881。取值必须在 1 到 65535 之间。

pwdFile

指向一个纯文本文件的路径，该文件包含用于 MQTT 代理身份验证的用户凭据。

文件中每行应包含以冒号分隔的用户名和密码，例如：

```
user1:password1
user2:password2
```

当此值留空（默认）时，MQTT 代理将接受未经身份验证的连接。文件路径可以是绝对路径，也可以相对于配置文件所在位置。

cert

代理的 TLS 证书，为.pfx 格式。在 DX Edit 中，点击 Browse 并选择.pfx 文件以导入；证书随后会启用 TLS——MQTT 代理加密其 WebSocket 端口，并在 TCP/IP 端点上使用 TLS。如果.pfx 受密码保护，在导入对话框中输入其密码：该密码仅 **only** 用于导入证书，不会被保存（证书保存时不带密码）。留空（默认）时，代理不加密运行。

内置代理（上述端口）仅在你从 **console** 自行运行管理器时才托管。作为 **service** 运行的管理器改用共享的系统代理，因此上述端口在该情况下不适用。

DX Manager is now optional.DX Edit 通过单个始终运行的 DX Run 后端管理项目，因此项目不再需要管理器来启动其模块——DX Run 自行完成。DX Manager 不再是默认项目模板的一部分；仅当你特意需要其内置代理或其启动功能时才运

行它。

mode

该管理器所启动模块的默认运行模式。留空时使用“service”。

- **service**（默认）——每个模块作为 Windows 服务安装并启动，由 Windows 管理。服务独立于管理器持续运行，并在系统启动时自动启动。这是已部署项目的推荐模式。
- **console**——每个模块作为控制台应用程序运行、带有自己的窗口，由管理器监督，并在管理器停止时停止。在开发期间很有用，可借此观察模块输出。

单个模块可通过 CSV 中的 **mode** 列覆盖此项目默认值。

CSV

CSV 文件定义要启动的 DX 模块列表。默认的 CSV 列如下：

name	description	mode	verbose
dxmodmast.json	Modbus Master		
dxnetscan.json	Network Scanner	console	off

- **disable**——设为“true”可跳过启动此模块，设为“false”则启用它。
- **name**——要启动的 DX 模块的 JSON 配置文件名（如“dxmodmast.json”），或模块可执行文件名（如“dxmodmast”）。提供 JSON 文件名时，管理器从文件中读取 **meta name** 以确定要运行哪个可执行文件。该文件必须与管理器配置位于同一目录。
- **description**——可选的描述，供你参考。
- **mode**——该模块的可选运行模式：“service”或“console”。留空则继承管理器的 **mode** 默认值。用它让单个模块以不同于其他模块的方式运行——例如，让整个项目保持服务模式，但将一个模块设为“console”以便观察其窗口。
- **verbose**——该模块的可选日志级别：“on”表示详细记录，“off”表示静默。留空则使用模块自身的默认值。

如果 CSV 中未定义任何数据条目（或未链接 CSV），管理器会自动发现同一目录下的所有 JSON 配置文件。任何包含有效 **meta name**（且不是另一个 **dxmanager** 配置）的 JSON 文件都会使用管理器的 **mode** 默认值自动启动。

在 Windows 上，每个模块根据其解析出的模式（模块自身的 mode 列，否则为管理器的 mode 默认值），作为 Windows 服务（默认）或控制台进程运行。在 Linux 上，管理器为每个模块创建并管理 systemd 服务单元文件。设为 disable 的行会在下次管理器启动时被协调关闭——控制台模块不启动，服务模块被停止并卸载——因此禁用总是会让模块下线。

示例

下面是一个完整示例，在代理上启用了身份验证来管理多个 DX 模块。

dxmanager.json:

```
{
  "servers": [{"url": "127.0.0.1:1883"}],
  "project": "PLANT1",
  "id": "mgr",
  "meta": {
    "name": "dxmanager"
  },
  "task": [
    {
      "enable": true,
      "id": "main",
      "desc": "Main Manager",
      "tcpip": 3881,
      "websocket": 4881,
      "pwdFile": "mqttusers.txt",
      "cert": "",
      "mode": "service",
      "data": [
        { "$csv": "dxmanager.csv" }
      ]
    }
  ]
}
```

dxmanager.csv:

disable	name	description	mode
false	dxmodmast.json	Modbus Master for PLCs	
false	dxnetscan.json	Network host scanner	console
true	dxopcucli.json	OPC UA Client (disabled)	

mqttusers.txt:

```
admin:secretpass
```

```
operator:op1234
```

采用上述配置，管理器将：

12. 在 TCP 端口 3881 和 WebSocket 端口 4881 上启动内置 MQTT 代理，并按 `mqttusers.txt` 中的定义要求用户名和密码身份验证。
13. 按项目 `mode` 默认值，将 Modbus Master 模块（`dxmodmast.json`）作为 Windows 服务启动。
14. 将 Network Scanner 模块（`dxnetscan.json`）作为控制台进程启动，因为其行将模式覆盖为“console”。
15. 跳过启动 OPC UA Client 模块，因为它被禁用。

Web 服务器

DX Web Server (dxweb) 为你的 DIOT 项目启动一个轻量级 Web 服务器。它从可配置的目录提供 HTML 页面、SVG 和其他 Web 内容等静态文件。在 Windows 上，模块启动时会自动用默认浏览器打开它找到的默认页面。当你浏览到服务器的根地址（只有主机或 IP、没有页面名）时，服务器会将你重定向到同一个默认页面。

关于 servers、project、模块 ID、trigger、update、data、存储转发和运行时等通用 DX 配置，请参阅 DX 驱动通用配置指南。

配置

下面是一个典型配置文件 dxweb.json 的示例。

```
{
  "servers": [{ "url": "127.0.0.1:1883" }],
  "project": "DEM0",
  "id": "module01",
  "meta": {
    "name": "dxweb"
  },
  "task": [
    {
      "enable": true,
      "id": "web01",
      "port": "8080",
      "url": "web"
    }
  ]
}
```

port

Web 服务器监听的端口号。留空（默认）时，服务器从 80 开始挑选第一个空闲端口——启用 HTTPS 时为 443——并在每次启动时重新检查，配置保持为空（不回复任何内容）。这样即使多个项目的 Web 服务器同时启动，也无需你为每个项目分配端口，即可让多个项目共享一台机器。你可以用字符串设置任何有效端口号，例如“3000”或“8443”，以绑定该确切端口；固定端口若已被占用，会使服务器启动失败，而不会自行改用其他端口。

在 Linux 或 macOS 上，绑定到 1024 以下的端口（如默认端口 80）需要进程以 root 运行，或被授予 CAP_NET_BIND_SERVICE 能力，例如 `sudo setcap 'cap_net_bind_service=+ep' <dxweb binary>`。如果你无法以这些特权运行，请设置一个高于 1024 的显式端口，例如“8080”。

url

包含静态文件的目录路径，相对于 JSON 配置文件。默认值为“web”，即该文件旁边名为 web 的文件夹。你可以用“.”提供配置文件自身所在目录，用“./pages”等其他相对路径，或用指向系统上任何目录的绝对路径。如果目录不存在，模块会报告并回退到 JSON 配置文件所在目录。

在 DX Edit 中（或用 `dxrun <name>`）创建的项目已经有一个 web 文件夹，内含示例页面、一个菜单（`js/menu.js`）和一个以项目命名的连接文件（`js/config.js`），因此默认的“web”可立即提供一个可用站点。

当提供配置文件自身所在目录时，只从其顶层提供页面文件（HTML、SVG、脚本、样式表、图片和字体），因此模块配置、CSV 文件和许可证保持私密。证书、密钥和数据库绝不会从其任何文件夹中提供，log 和 data 文件夹（模块存放日志和历史的位置）也不提供任何内容。

ex

一个字符串，用于指定有权读取的角色或访问级别。默认值为空字符串。

默认页面

当你浏览到服务器的根地址（只有主机或 IP、没有页面名）时，服务器会选择一个 default page 显示。在 Windows 上，模块启动时也会在浏览器中自动打开同一页面。页面按以下顺序从所服务目录中选择：

16. 所服务目录中按字母顺序排列的第一个 `index.html` 或 `index.htm` 页面。
17. 否则，为所服务目录中按字母顺序排列的、任意名称的第一个 HTML 页面。
18. 否则——仅当 url 提供配置文件自身所在目录（.）时——同样的搜索会按字母顺序继续进入子文件夹，在每一层应用相同规则。

如果完全找不到 HTML 页面，服务器返回其 404 (page not found)响应。当 url 指向特定文件夹时，只搜索该文件夹（不搜索其子文件夹）。

服务地址

你项目的页面在站点根目录提供，与内置 DX 页面并列。共享的 DX 框架在/system 下提供。

地址	服务内容
/<page>	来自你所服务目录的页面。当你的项目没有该名称的页面时，则为同名的内置页面：dxedit.html、dxconsole.html、dxwatch.html、dxalm.html、dxaudit.html、dxstatus.html 或 dxtrend.html。
/system/...	共享的 DX 框架：脚本、样式表、字体、图片和画面引擎。/system 下不提供任何页面。
/doc/...	DX 用户指南。/doc/打开其菜单，不存在的指南页面也会如此。

在你自己的页面中，用 system/ 路径加载框架，例如 system/css/dx.css 和 system/js/dx.js。请将 js/menu.js 和 js/config.js 保留在你的项目中：它们为你的页面和内置页面设置菜单和代理。加载框架时不带 system/ 前缀的旧页面（例如 css/dx.css）仍可工作。

要在另一个代理上打开内置页面，将其加入地址，例如 /dxalm.html?url=192.168.1.20:1884。浏览器以前用过的代理会立即连接；新代理会先询问，仅在你选择 Connect 时才连接。

早期版本中以/system/开头的地址（例如/system/dxedit.html）不再打开页面。请使用不带/system 的相同地址，例如/dxedit.html。

浏览器缓存

模块提供的每个文件——HTML 页面、脚本、样式表、图片——都带有 Cache-Control: no-cache 发送。这不会阻止浏览器保存副本；它要求浏览器在复用副本前

先向服务器检查。当你替换文件时，更新版本会立即被采用，无需清除浏览器缓存或重启模块。未更改的文件会以轻量的“未修改”回复应答，因此检查成本很低。

这对脚本和样式表最重要。页面链接可手动更改以强制重载，但页面内的脚本和样式表名称保持不变，因此若没有此行为，浏览器可能会针对刚更新的页面继续运行旧脚本。

升级到此版本后，每个浏览器可能需要一次强制刷新（Ctrl+F5），以丢弃它在先前行为下已缓存的文件。此后更新会自动被采用。

菜单

你页面上的菜单在所服务目录内的项目 `js/menu.js` 中设置。内置页面始终显示它；你自己的页面在加载 `js/menu.js` 并放置 `<dx-menu>` 元素时显示它。

```
var MENU = {
  "": {
    "Overview": ["overview.html", "fa fa-industry"],
    "Alarms": ["dxalm.html", "fa fa-bell"],
    "Settings>Network": ["network.html", "fa fa-sitemap"]
  }
};
```

- **Menus per role**——顶部的每个名称（上面的“”）都是一个角色。“”是默认菜单：在任何人登录前显示，也显示给没有自己菜单的已登录用户。已登录用户看到其第一个角色的菜单。你可以通过添加另一个块来为某个角色添加菜单，例如“engineer”: { ... }。
- **Entries**——每个条目为“Label”: [page, icon]，按写入顺序显示。页面是该页面在站点根目录的地址：你自己的某个页面，或如 `dxalm.html` 这样的内置页面。图标是 Font Awesome 类，如“fa fa-bell”，或图片路径，如“img/pump.png”。
- **Groups**——“Parent>Child”将条目 Child 放在名为 Parent 的分组下。

注意：菜单只决定每个角色看到哪些链接。它不会阻止任何人通过地址打开页面。

你也可以在 DX Edit 中以表单形式编辑菜单：打开项目的 `dxweb` 模块，然后打开其 Web 选项卡 >Menu。

在 DX Edit 中添加的 dxweb 带有一个空的 dxweb.csv，以其任务的 data 命名。模块不读取它：其菜单列（disable、name、url、role、description）不被使用，菜单仅来自 js/menu.js。你可以保持该文件原样。完全没有 data 的任务也能运行。

HTTPS

你可以通过向任务配置添加“https”属性并设为 true 来启用 HTTPS。启用 HTTPS 时，空白端口会从 443 开始自动选择第一个空闲端口（而不是 80）；指定端口则确切绑定。要使 HTTPS 工作，机器上必须有可用的 SSL 证书。

```
{
  "task": [
    {
      "enable": true,
      "id": "web01",
      "https": true,
      "port": "",
      "url": "web",
      "data": [
        { "$csv": "dxweb.csv" }
      ]
    }
  ]
}
```

示例

下面是一个完整示例，提供一个工厂总览仪表盘。

dxweb.json:

```
{
  "servers": [{"url": "127.0.0.1:1883"}],
  "project": "PLANT1",
  "id": "web1",
  "meta": {
    "name": "dxweb"
  },
  "task": [
    {
      "enable": true,
      "id": "hmi",
      "port": "8080",
      "url": "./pages"
    }
  ]
}
```

pages/js/menu.js:

```
var MENU = {
  "": {
    "Overview": ["overview.html", "fa fa-industry"],
    "Alarms": ["dxalm.html", "fa fa-bell"]
  },
  "engineer": {
    "Overview": ["overview.html", "fa fa-industry"],
    "Alarms": ["dxalm.html", "fa fa-bell"],
    "Settings>Network": ["network.html", "fa fa-sitemap"],
    "Settings>Users": ["users.html", "fa fa-users"]
  }
};
```

采用上述配置，模块将：

19. 在端口 8080 上启动 Web 服务器，从相对于 JSON 配置文件的“pages”子目录提供静态文件。
20. 在每个加载 js/menu.js 的页面上显示菜单。登录前，以及对于没有自己菜单的任何用户，菜单包含 Overview 和 Alarms（内置报警页面）。第一个角色为“engineer”的用户还会看到 Settings 分组，含 Network 和 Users。菜单只隐藏链接：在地址中输入 network.html 的用户仍可打开它。
21. 在 Windows 上，默认浏览器会自动打开在“pages”目录中找到的默认页面（见默认页面）。浏览到服务器的根地址会重定向到同一页面。由于这里 url 指定了特定文件夹（“./pages”），只搜索该文件夹；如果其中没有 HTML 页面，服务器返回 404 响应。

登录

DX 用户登录界面是人们登录你的项目并管理自己账户的地方。它们由 DX User Role Manager (dxrole) 提供支持，后者验证每次登录。本指南介绍登录和你自己的账户。注册首位管理员以及创建或管理用户在 DX Edit 中完成；组、角色和登录规则如何配置，请参阅 DX User Role Manager 指南。

登录

登录分两步。首先输入你的用户名。

```
+-- Sign in -----+
| User name : [ operator          ] |
|                               [ Continue ] |
+-----+
```

点击 **Continue**，页面会显示 your 安全图片和密码短语——即你所选的那组。这是你确认页面真伪的依据。

```
+-- Sign in -----+
| +-----+ |
| | (your   | | "blue harbor 42"
| | image) | |
| +-----+ |
| Is this your image and phrase?
| If not, do NOT enter your password.
|
| [x] I'm not a robot
| Password : [ ***** ]
|                               [ Sign in ]
+-----+
```

如果图片和短语是你设置的，通过快速机器人检查，输入密码，然后点击 **Sign in**。如果它们 **not** 你所设置的，应停止——你可能在假冒页面上——并告知管理员。

请注意：

- 即使是未知用户名也会显示图片和短语，因此不要把这当作该用户名存在的确认。
- 密码错误次数过多后，你的账户会被锁定，只有管理员才能解锁。
- 错误的用户名或密码总是给出相同消息——绝不会说明是哪一项出错。

双因素

如果你的账户需要第二因素，输入密码后会再出现一个界面。

+-- Two-factor -----+

Enter the 6-digit code from your authenticator app:

[- - - - -]

[Verify]

or [Use a passkey / security key]

or [Use a recovery code]

输入验证器应用中当前的 6 位验证码并点击 **Verify**。如果你注册了通行密钥或安全密钥，点击该按钮并按设备提示操作（Windows Hello、Face ID、手机或插入式密钥）——此选项仅在你的界面通过安全 HTTPS 地址提供时出现。如果设备丢失，点击 **Use a recovery code** 并输入设置双因素时保存的某个代码——每个代码只能用一次。

设置安全图片（首次登录）

如果你尚未选择安全图片和密码短语，首次登录时会被要求设置。

+-- Set up your security image -----+

Pick an image you will recognise:

[img] [img] [img] [img]

[img] [img] [img] [img] or [Upload...]

Passphrase : [blue harbor 42]

[Save]

选择一张图片，或点击 **Upload...**使用你自己的图片——除非管理员已将图片选择限制为提供的图集。输入一个只有你知道的简短密码短语。从此每次登录时两者都会出现，使你能辨别真实页面与假冒页面。请选择你能立即认出的内容。

你的账户

登录后，你可以管理自己的账户。

+-- My account -----+

Change password [Change]

Security image / phrase [Change]

Authenticator app [Set up]

Passkey / security key [Add]

```
+-----+
```

- **Change password**——输入当前密码和新密码。更改后，你的其他登录会被注销，因此你在别处保持登录的设备会被断开。
- **Security image / phrase**——选择另一张图片或上传新图片，并更改密码短语。
- **Authenticator app**——点击 **Set up**，用应用扫描二维码，并输入其显示的代码以开启双因素。你会获得一次性恢复代码——请妥善保存。设置完成后，你也可以随时生成一组新的恢复代码，以替换旧代码。
- **Passkey / security key**——添加通行密钥或插入式密钥作为第二因素。此选项仅在你的界面通过安全 HTTPS 地址提供时出现。你可以添加多个；要移除一个，需用当前验证器代码或恢复代码确认。如果它是你唯一的第二因素，必须由管理员为你重置。

如果你的组要求双因素，你可以添加或更改因素，但不能关闭双因素。

当登录页面无法连接服务时

如果 dxrole 不可用，登录页面会回退到简单登录：只需用户名和密码。

```
+-- Sign in (basic) -----+
| User name : [ operator   ] |
| Password  : [ *****   ] |
|                                     [ Sign in ] |
+-----+
```

在此模式下，安全图片、机器人检查、锁定和双因素等额外保护在服务恢复前不可用，因此凭普通密码即可进入。在更严格、必须运行 dxrole 才能登录的站点，此基础登录会被关闭，你只需等待服务恢复。无论哪种方式，错误的用户名或密码都显示相同的通用消息；如果持续出现，请联系管理员，因为你的账户可能已被锁定，只有管理员才能解锁。

用户角色管理器

DX User Role Manager (dxrole) 保护你的项目。它为每个客户端——屏幕前的操作员、某个工具或另一个 DX 模块——提供用户名、密码，以及它被允许查看和发送的确切数据集。它为你设置并管理本地 MQTT 代理，因此你只需维护两个小文件：用户组列表，以及每个组中的用户。dxrole 还自行验证人工登录，并以加密形式存储每个密码。

通用 DX 配置 (servers、project、模块 id、data 节) 见 DX 驱动通用配置指南。用户通过 DX 用户登录界面登录；管理员在 DX Edit 中注册首位管理员并创建、管理用户。

设置管理员前不会有任何安全措施

全新安装时，dxrole 本身并 not 保护任何东西——按设计没有内置或默认管理员。在你创建首位管理员之前，dxrole 为完成设置，会让本地代理仅在 on this machine only (环回) 开放，使你无需密码即可注册首位管理员，除此之外不保护任何东西。

首位管理员登录的那一刻，dxrole 就会锁定代理——开启身份验证、配置你的用户、移除临时开放访问——此后每个客户端都需要账户。因此任何新项目的第一件事就是设置管理员。

锁定步骤会编辑 Mosquitto 的配置并重启其服务，因此 run dxrole as an administrator。如果无法做到，dxrole 会让代理保持完全关闭 (全部拒绝)，而不是让它开放。

设置首位管理员

有两种方式，任选其一即可。

From DX Edit (recommended). 当不存在管理员时，在本机打开 DX Edit 的管理员注册页面，输入管理员名称和强密码并保存。在登录之前，你可以随时重新打开页面更改它们。管理员在 confirmed the first time they sign in 得到确认——那一刻 dxrole 就会保护代理。

By editing the files.dxrole 首次启动时会写入起始文件。dxrole.csv——即组列表——带有一个 admins 组：

name	type
admins	admin

以及 admins.dat——即其组文件——其中有一个尚无密码的占位管理员：

```
{
  "subscribe": ["#"],
  "publish": ["#"],
  "users": [ { "uid": "admin1", "pwd": "", "manage": true } ]
}
```

设置管理员名称和强密码（替换 admin1、填入 pwd），然后重启。在通过任一方式设置真实密码之前，dxrole 不创建管理员和登录账户（空密码占位会被忽略），因此 no default credential ever exists。

管理员是一个持久账户，keeps working even after a broker restart，manage 标志使其能添加和更改其他所有人。如果你丢失管理员密码，在 admins.dat 中设置新密码并重启。

用户类型

每个用户组都有一个 type，决定其账户如何保存在代理中。这是添加用户时的主要选择：

Type	What it is	Survives a broker restart while dxrole is down?	Use it for
dynamic（默认）	ordinary login users	No——dxrole 运行时重新创建	operators and staff who sign in at the screens
role	a fixed account per user, read+write on its one area	Yes——存储在代理本身	equipment / panel accounts that must always stay connected
admin	a fixed account whose exact topics you set	Yes	a third-party / integration client that needs its own broker access, up to everything

简言之：**dynamic** 用户是每天登录的普通人；**role** 和 **admin** 用户是保存在代理本身的固定账户，因此即使重启后 **dxrole** 尚未恢复，它们也能继续连接。

type 只决定账户的 **access and whether it persists**——它并 **not** 决定谁是管理员。成为 **administrator**（能添加和更改用户）是独立的 **manage** 标志，可设置在任何组的 **any** 用户上，包括 **dynamic** 组。因此普通的 **dynamic** 操作员也可以是管理员，而 **admin** 类型账户除非你授予 **manage**，否则不是管理员。

设置用户

管理员通常在浏览器中通过 **DX Edit** 添加和管理用户——这是推荐方式，无需手动编辑。本节描述底层文件，供你更愿意直接编辑时使用（编写设置脚本，或创建首位管理员）。

用户存放在两类相邻的文件中，使日常更改绝不会重启模块：**group list**（**dxrole.csv**），以及每个组的 **group file**（**<name>.dat**）。

The group list — dxrole.csv.每行一个组：

name	role	type
operators	Area1 Area2	dynamic
site1	Area1	role
historian		admin

- **disable**——“true”关闭整个组；留空则启用。
- **name**——组名称，也是其 **.dat** 文件的基础名称。
- **role**——该组可访问的区域，以空格分隔（每个区域授予对该区域数据的完全访问权限；单个+授予所有区域）。**admin** 组将此留空，并在 **.dat** 中设置自己的主题。
- **type**——**dynamic**、**role** 或 **admin**（见上文）。留空则为 **dynamic**。

添加组是配置变更（快速重载）。添加或更改组内用户是对其 **.dat** 的实时编辑——无需重载。

The group file — for example operators.dat.顶部的共享登录设置适用于组内每个用户，然后是 **users** 列表：

```
{
```

```

    "require2FA": true,
    "fallback": "hard",
    "maxSessions": 3,
    "users": [
      { "uid": "alice", "pwd": "Alice@2026", "manage": true },
      { "uid": "bob", "pwd": "Bob@2026" }
    ]
  }

```

共享设置（全部可选；省略则取默认值）：

- **require2FA**——组内每个用户必须设置双因素。默认 `false`。
- **fallback**——“soft”（默认）在 `dxrole` 短暂不可用时，允许用户仅凭密码登录；“hard”意味着他们始终经过 `dxrole`，不能回退。
- **maxAttempts**——登录失败达到此次数后锁定账户。默认 5。
- **maxSessions**——每个用户的并发会话数。默认 0（不限）。
- **pwdExpiryDays**——经过这么多天后强制更改密码。默认 0（从不）。

每个用户：

- **uid**——登录名。
- **pwd**——密码。直接输入即可；`dxrole` 首次启动时会就地加密。目录（LDAP）用户在此没有密码（见下文 `Directory login`）。
- **manage**——设为 `true` 可使该用户成为 `administrator`，能添加和更改用户及其访问权限。适用于任何组中的用户。
- **phrase** 和 **image**——用户的安全密码短语和图片；通常在首次登录时设置。

An admin-type group file 设置其账户可用的确切主题，而非区域：

```

{
  "subscribe": ["plant/#"],
  "publish": ["plant/cmd/#"],
  "users": [ { "uid": "historian", "pwd": "Hist@2026" } ]
}

```

- **subscribe / publish**——这些账户可读取 / 写入的主题（标准 MQTT 主题过滤器；`#`表示一切）。只有 `admin` 类型组使用它们。

如果组不需要共享设置，只给出用户即可：{ “users”: [...] }。

移除用户

从.dat 中移除某个用户（或从 dxrole.csv 中移除整个组），dxrole 会在下次运行时移除该代理账户，因此你一旦将其从文件中取出，访问权限即被收回。

dxrole 只移除 it 创建的账户。如果你的代理已有自己的账户——你在 dxrole 之外直接设置的账户——dxrole 完全不干预：它绝不会删除或更改不是自己创建的账户。要将此类账户纳入 dxrole，以相同名称添加到文件中；此后 dxrole 会像管理其他账户一样管理它。

示例

一个有两个区域的小型工厂。操作员兼顾两个区域并在屏幕前登录；每个区域还有一个固定面板账户，必须在重启后保持连接；一个历史记录工具需要自己的代理访问权限。

dxrole.json:

```
{
  "servers": [ { "url": "127.0.0.1:1883", "project": "plant", "qos":
1 } ],
  "task": [
    {
      "id": "role1",
      "broker": { "type": "mosquitto" },
      "authMethods": { "localPassword": true, "totp": true },
      "data": [ { "$csv": "dxrole.csv" } ]
    }
  ]
}
```

dxrole.csv:

name	role	type
admins		admin
operators	Area1 Area2	dynamic
panel1	Area1	role
panel2	Area2	role
historian		admin

operators.dat——要求双因素、强制门禁:

```
{
  "require2FA": true,
  "fallback": "hard",
}
```

```
    "users": [
      { "uid": "alice", "pwd": "Alice@2026", "manage": true },
      { "uid": "bob", "pwd": "Bob@2026" }
    ]
  }
```

panel1.dat——Area1 面板的固定账户：

```
{ "users": [ { "uid": "panel1", "pwd": "Panel1@2026" } ] }
```

historian.dat——自己的主题：

```
{ "subscribe": ["plant/#"], "users": [ { "uid": "historian", "pwd": "Hist@2026" } ] }
```

代理

dxrole 为你设置本地 Mosquitto——无需开启任何东西。你并 not 在 servers 下放置代理用户名或密码；dxrole 用自己的内置账户连接。

- **broker.type**——默认的“auto”行为类似“mosquitto”。使用“none”可让 dxrole 在文件中管理用户和角色，同时不干预代理（你自行配置代理）。
- **Automatic setup**——dxrole 为本地设置开放代理，然后在首位管理员登录时保护它（见上文）。如果 Mosquitto 在另一台机器上、或已受保护，dxrole 不干预它。它写入的安全存储位于你的 dxrole.json 旁边。
- **broker.pruneUnmanagedClients**（默认 true）——代理镜像你的文件：文件中不再存在的账户会被移除。设为 false 可保留 dxrole 未创建的账户。它自己的内置账户绝不会被移除。
- **broker.protectedClients**——dxrole 绝不移除的账户名，即使在修剪时。默认空。
- **broker.managementDestination**（默认“secadmin”）——用户管理命令所发往的保留区域。不要将其设为“admin”（保留的代理名称）。你用 **manage** 标志创建管理员，因此很少需要更改此项。

dxrole 未运行时，采用默认 fallback: “soft”，如果 dxrole 短暂不可用，dynamic 用户仍可仅凭密码登录。而且由于 role / admin 账户存储在代理本身，即使在 dxrole 恢复之前，它们在代理或 PC 重启后仍能继续连接。

登录选项

- **authMethods**——界面提供哪些登录方式。**localPassword**（默认 **true**）是普通密码登录；**totp**（默认 **true**）提供验证器应用双因素。目录和通行密钥登录在下面配置后会自动出现。“authMethods”: { “localPassword”: true, “totp”: true }
- **sessionIdleMinutes**（默认 30）——不活跃的登录会话经过这么多分钟后会被断开。
- **gateBrokerAccess**（默认 **false**）——设为 **true** 可构建更严格的站点，每次登录都必须经过 **dxrole**：账户在 **dxrole** 验证登录前保持禁用，经典密码回退被关闭。
- **passwordPolicy**——省略则用内置规则（至少 8 个字符，含一个大写字母、一个小写字母、一个数字和一个特殊字符）。可覆盖其中任意项：“passwordPolicy”: { “minLength”: 12, “historyCount”: 5 } **historyCount** 记住那么多以前的密码并阻止重复使用。
- **Security images**——每个用户挑选一张自己在登录时能认出的个人图片，作为防钓鱼提示。**imageGallery**（留空用内置图集）/ **galleryDir**（指向你自己的文件夹）/ **allowCustomImages**（默认 **true**，允许用户上传自己的图片）/ **maxImageBytes**（默认 131072 = 128 KB）/ **uniformImageSize**（默认 **true**，让每张图片保持相同尺寸，从而无法据此区分用户）。

目录（LDAP）登录

可选的企业目录（LDAP / Active Directory）登录：用户用其目录密码而非存储的密码登录。目录用户是普通组成员，**authSource** 设为“**ldap**”且没有本地密码；该组设置其共享登录姿态，但其角色在每次登录时决定：

- 如果你在 **roleMap** 中将其目录组映射到角色，目录决定其角色——此项优先。
- 如果目录未返回匹配的组，**dxrole** 使用你在该用户上设置的 **localRole**（空格分隔的区域，类似 CSV 的 **role**）。
- 如果两者都不适用，用户无法登录。

LDAP 用户始终通过 **dxrole** 登录——无论其组的 **fallback** 设置如何，他们都没有直接代理回退。

```
"ldap": {
  "enable": true,
  "host": "dc.example.com",
  "port": 636,
  "security": "ldaps",
  "baseDn": "dc=example,dc=com",
  "userFilter": "(&(objectClass=person)(sAMAccountName={0}))",
  "groupAttr": "memberOf",
  "roleMap": [ { "group": "SCADA-Operators", "role": "Area1 Area2" } ]
}
```

每个 roleMap 条目给出一个目录组名称及其成员获得的角色。属于多个被映射组的用户获得其所有角色的组合。

Passkeys / security keys (webauthn). 可选的通行密钥（Windows Hello、Face ID、手机）或安全密钥（如 YubiKey）双因素。它仅在你的界面通过真实域名的 HTTPS 提供时才有效，因此除非你设置 rpId（域名）和 allowedOrigins，否则它是关闭的。在推出时请谨慎选择 rpId——以后更改会使已注册的每个通行密钥和安全密钥失效。

```
"webauthn": {
  "rpId": "hmi.plant.example",
  "allowedOrigins": [ "https://hmi.plant.example" ],
  "userVerification": "preferred",
  "timeout": 60000
}
```

脚本引擎

DX Script Engine（dxscript）让你读取并执行 JavaScript 文件、文本模板或内联表达式，以处理和转换标签数据。你可以用由 V8 引擎驱动的标准 JavaScript 实现自定义逻辑、计算和数据操作。脚本通过内置的 `tag.get()` 和 `tag.set()` 函数与标签系统交互，并提供 `console` 对象用于记录日志。

关于 `servers`、`project`、模块 ID、`trigger`、`update`、`data`、存储转发和运行时等通用 DX 配置，请参阅 DX 驱动通用配置指南。

配置

下面是一个典型配置文件 `dxscript.json` 的示例。

```
{
  "servers": [{"url": "127.0.0.1:1883"}],
  "project": "DEM0",
  "id": "my",
  "meta": {
    "name": "dxscript"
  },
  "task": [
    {
      "enable": true,
      "id": "logic",
      "url": "js",
      "trigger": ["1000"],
      "role": "",
      "data": [
        { "$csv": "dxscript.csv" }
      ]
    }
  ]
}
```

url

存放脚本文件的文件夹。设置后，模块会在此文件夹中查找 CSV 中指定的 `.js` 和 `.txt` 文件，因此 CSV 只列出纯文件名（例如 `my.js`），绝不重复文件夹。默认配置使用 `js`。留空时，每个 CSV 条目必须包含自己的路径，文件相对于包含 JSON 配置文件的文件夹解析。此字段也接受 HTTP URL，允许从远程 Web 服务器加载脚本。

role

指定发布标签数据的角色。留空（默认）时，标签发布到默认项目目的地。你可以用字符串设置单个角色，或用字符串数组设置多个角色。指定多个角色时，此任务中脚本产生的标签值会转发给每个角色。

event

一个可选参数，用于订阅额外的事件主题。设为“onalm”时，任务会订阅报警数据，使报警值可通过`${tagname[index]}`语法在字符串格式化模板中使用。留空（默认）时，只处理标准的 `onchange`（变化）数据。

CSV

CSV 文件定义要执行的脚本、模板或表达式列表。默认的 CSV 列如下：

name	trigger	expression	description
my.js	1000		
calc1	1000	<code>\${temperature} * 1.8 + 32</code>	Convert Celsius to Fahrenheit

- **disable**——设为“true”可跳过此条目，设为“false”则启用。
- **name**——脚本文件名或标签名。当任务 `url` 已设置时，此处只给出纯文件名（例如 `my.js`）；文件夹来自 `url`。行为取决于文件扩展名：以 `.js` 结尾——模块将文件作为 JavaScript 脚本加载并执行。以 `.txt` 结尾——模块将文件作为字符串格式化模板加载（见下文字符串格式化模板）。任何其他值——被视为标签名，其值将通过对 `expression` 列求值来设置。
- **trigger**——此单个条目的可选触发器。接受与任务级触发器相同的格式（毫秒间隔、cron 表达式，或如 `${tag}+-`、`${tag}=5` 这样的基于标签的触发器，或如 `${tag}>100` 这样的比较）。留空时，条目继承其父任务的触发器。由于 CSV 单元格只容纳单个字符串，每个条目的触发器是一个条件；OR/AND 分组请使用任务级 `trigger` 数组（见下文）。
- **role**——此特定条目的可选角色覆盖。留空时，条目继承其父任务的角色。
- **expression**——当 `name` 为标签名（不是 `.js` 或 `.txt` 文件）时使用的内联表达式。你可以用 `${tagName}` 语法引用其他标签，表达式的结果将赋给 `name` 列指定的

标签。支持标准 JavaScript 算术和逻辑运算符。与.txt 模板一起使用时，表达式指定格式化结果的输出标签名。

- **description**——可选的描述，供你参考。

按标签条件触发脚本

除间隔和 cron 触发器外，脚本可在标签满足某条件时运行，使用 DX 驱动通用配置指南中描述的触发语法：值变化（`${tag}+-`）、相等（`${tag}=5`）、或比较（`${tag}>100`、`${tag}<10`、`${tag}>=`、`${tag}<=`、`${tag}!=`）。比较触发器在条件成立的 every 更新时触发。

共享相同触发器（和角色）的所有条目会被分组，在该触发器触发时 **together in a single pass** 运行——其脚本和表达式按列出顺序执行。这让一个条件可以驱动一组相关计算。

要组合条件，使用 `task-leveltrigger` 数组（每个条目的 CSV 触发器只是一个条件）。将条件作为独立项列出表示 OR，或将它们嵌套在数组中表示 AND：

```
"trigger": ["${Pump.On}=1", ["${Level}>90", "${Flow}<5"]]
```

当 Pump.On 等于 1，orLevel 高于 90andFlow 低于 5 同时成立时，运行任务的脚本。注意 OR 表示为独立项——单个条件匹配一个值（不支持如 1,5,9 这样以逗号分隔的值列表）。

脚本文件

脚本文件是由 V8 引擎执行的标准 JavaScript（.js）文件。脚本中可使用以下对象：

tag

tag 对象提供两个与标签系统交互的方法：

- `tag.get(tagName)`——返回指定标签的当前值。如果标签尚无值，返回 0。你也可以传入标签名数组一次获取多个值；结果是一个 JSON 字符串，用 `JSON.parse()` 解析。

- **tag.set(tagName, value)**——设置指定标签的值。你也可以传入单个对象（每个键为标签名、每个值为标签值），在一次调用中设置多个标签。

这两个方法也可作为全局函数使用：**getTag(name)** 等价于 **tag.get(name)**，**setTag(name, value)**（或 **setTag(object)**）等价于 **tag.set(...)**。

console

遵循标准 Web console API 的日志对象：

- **console.error(msg)**——记录一条错误消息。
- **console.warn(msg)**——记录一条警告消息。
- **console.info(msg)**——记录一条信息消息。
- **console.debug(msg)**——记录一条调试消息。
- **console.trace(msg)**——记录一条跟踪消息。
- **console.time(label)** / **console.timeLog(label)** / **console.timeEnd(label)**——用给定标签测量调用之间经过的时间。

random

提供 Random 对象用于生成随机数，对测试和模拟很有用。

错误报告

共享触发器的所有脚本和表达式在一次遍历中一起运行，但运行时错误仍针对其自身来源报告：日志行会指明 script file (or the output tag name for an expression) and the line within that file，例如 js/report.js:12:5 ReferenceError: ...。即使多个脚本成组运行，这也能让你直接定位到出错的行。

内联表达式

当 CSV 条目的 name 列是标签名（不是.js 或.txt 文件）时，expression 列会作为 JavaScript 表达式求值，结果赋给该标签。你可以用 **\${tagName}** 语法在表达式中引用其他标签。例如：

- **\${temperature} * 1.8 + 32**——读取标签“temperature”，转换它，并将结果存储到 name 列指定的标签。
- **\${sensorA} + \${sensorB}**——将两个标签值相加。

你也可以用`${app.alarm[tagname].status}`格式引用报警状态，它会解析为指定报警标签的当前值。

字符串格式化模板

当 CSV 条目的 `name` 列是 .txt 文件时，模块将该文件作为字符串格式化模板加载。

模板使用 JavaScript 模板字面量语法，支持以下占位符格式：

- **`${tagname}`**——替换为指定标签的当前值。
- **`${tagname[index]}`**——替换为报警数据中的某个值。索引指报警值数组中的位置（例如`${alm.process.temp_hi[0]}`获取第一个字段）。这要求任务级 `event` 参数设为“onalm”。
- **`${CSV.rowname.columnname}`**——替换为通过 CSV 主题接收到的 CSV 数据中的某个值。`rowname` 匹配传入 CSV 的“name”列，`columnname` 匹配列标题（不区分大小写，空格转换为下划线）。

格式化结果发布到一个标签。输出标签名若指定了 `expression` 列则由其决定，否则默认为去掉.txt 扩展名后的模板文件名。

示例

下面是一个完整示例，以不同间隔运行两组脚本。

dxscript.json:

```
{
  "servers": [{ "url": "127.0.0.1:1883" }],
  "project": "PLANT1",
  "id": "calc",
  "meta": {
    "name": "dxscript"
  },
  "task": [
    {
      "enable": true,
      "id": "fast",
      "trigger": ["500"],
      "data": [
        { "$csv": "dxscript-fast.csv" }
      ]
    },
    {
      "enable": true,
      "id": "slow",
```

```

        "trigger": ["5000"],
        "data": [
            { "$csv": "dxscript-slow.csv" }
        ]
    }
]
}

```

dxscript-fast.csv:

name	expression	description
js/realtime.js		Real-time calculations
flow.total	$\${flow.rate} * 0.5$	Accumulate flow every 500ms

dxscript-slow.csv:

name	expression	description
js/report.js		Periodic report logic
avg.temp	$(\${temp.zone1} + \${temp.zone2} + \${temp.zone3}) / 3$	Average temperature

js/realtime.js:

```

var rate = tag.get("flow.rate");
var total = tag.get("calc.fast.flow.total");
if (rate > 0) {
    tag.set("calc.fast.flow.total", total + rate * 0.5);
}

```

采用上述配置，模块将：

- 每 500 毫秒执行快速任务。脚本“realtime.js”和“flow.total”的内联表达式会在每次触发时运行。标签名遵循“calc.fast.{name}”模式。
- 每 5 秒执行慢速任务。脚本“report.js”和“avg.temp”的内联表达式会在每次触发时运行。标签名遵循“calc.slow.{name}”模式。
- 内联表达式使用 $\${tagName}$ 语法读取标签值，并将计算结果赋给 name 列指定的标签。

报警与审计

DX Alarm Manager (dxalm) 处理报警和审计事件。它订阅来自其他模块的标签数据，针对传入值评估用户定义的条件，并在满足这些条件时发布报警或审计记录。模块支持两种运行模式：用于标准报警处理、具有活动/未活动/已确认/已关闭生命周期的报警模式，以及用于事件记录、不跟踪确认的审计模式。

关于 servers、project、模块 ID、trigger、update、data、存储转发和运行时等通用 DX 配置，请参阅 DX 驱动通用配置指南。

配置

下面是一个典型配置文件 dxalm.json 的示例。

```
{
  "servers": [{ "url": "127.0.0.1:1883" }],
  "project": "DEM0",
  "id": "log1",
  "meta": {
    "name": "dxalm"
  },
  "update": ["1000"],
  "task": [
    {
      "enable": true,
      "id": "device1",
      "mode": "Alarm",
      "data": [
        { "$csv": "dxalm.csv" }
      ]
    }
  ]
}
```

mode

设置任务的运行模式。接受值为“Alarm”和“Audit”。默认值为“Alarm”。

在 Alarm 模式下，每个报警条目遵循活动、未活动、已确认和已关闭的生命周期。满足配置条件时报警变为活动，条件不再满足时变为未活动，一旦既被确认又回到未活动即关闭。模块跟踪每个报警的状态，并在状态变化时发布更新。

在 Audit 模式下，每次触发条件时，每个事件作为一次性记录记录。没有活动/未活动生命周期或确认跟踪。模块将每个审计条目的最后已知状态持久化到本地文件，以便重启后恢复状态。

srcFilter

一个可选字符串，用于过滤传入标签数据的来源。设置后，此任务只处理主题包含该字符串的来源数据。留空或省略时，接受所有来源的数据。

CSV

CSV 文件定义要监视的报警或审计条目。默认的 CSV 列如下：

name	msg	v1	cond 1	v 2	cond 2	v3	ORcondition
temp_hi	Temperature High	\${device1.grp1.temperature}	>	80			
pressure_range	Pressure Out of Range	\${device1.grp1.pressure}	>	50	<	100	false

- **disable**——设为“true”可跳过此条目，设为“false”则启用。
- **name**——此报警条目的标签名。系统中的完整标签名遵循“moduleId.taskId.name”的命名约定。
- **msg**——用于显示的可选报警消息。
- **v1**——条件的第一个操作数。输入以下之一：作为\${tagname}的 tag（如 \${device1.grp1.temperature}；也接受旧的\$tagname 简写），一个 value——数字（80、1.5）或用于字符串比较的文本——或 null。开头的\$始终表示标签，因此值不能以\$开头。单元格会显示其读取方式的标记——value、tag、invalid（如\${temp 这样的损坏标签，永远无法解析，因此报警无法据此触发）、或 incomplete（条件已设置但此处为空）；点击空单元格上的?可查看接受的格式。
- **cond1**——第一个条件的比较运算符。支持的运算符有：“<”、“>”、“<=”、“>=”、“=”、“!=”。
- **v2**——第一个条件的第二个值或标签引用，如果定义了第二个条件，它也是第二个条件的第一个操作数。遵循与 v1 相同的格式。

- **cond2**——第二个条件的比较运算符。指定时，模块评估两个条件：**v1 cond1 v2**，以及 **v2 cond2 v3**。如果只需要单个条件，将此留空。支持的运算符与 **cond1** 相同。
- **v3**——第二个条件的第二个值或标签引用。仅在指定 **cond2** 时使用。遵循与 **v1** 相同的格式。
- **deadbandVal**——死区值。这可防止被监视值在阈值附近时报警快速切换。其含义取决于 **deadbandType** 列。留空表示无死区。
- **deadbandType**——指定死区值如何解释。接受值为：（空）——**deadbandVal** 被视为应用于阈值的固定数值偏移。例如，如果条件为“> 80”且 **deadbandVal** 为“5”，值超过 80 时报警激活，但直到值降到 75 以下才取消激活。**%**——**deadbandVal** 被视为所比较值的百分比。例如，**deadbandVal** 为“10”表示 10% 死区。**s**——**deadbandVal** 被视为以秒为单位的基于时间的死区。报警条件必须持续为假达到指定秒数，报警才会取消激活。
- **role**——一个可选角色名，用于限制此报警条目适用于哪个目的地角色。留空时，报警适用于所有角色。
- **description**——可选的描述，供你参考。

条件逻辑

只定义 **cond1**（**cond2** 为空）时，单个条件“**v1 cond1 v2**”评估为真时报警触发。

cond1 和 **cond2** 都定义时，两个条件“**v1 cond1 v2**”和“**v2 cond2 v3**”默认使用 AND 逻辑一起评估。这意味着只有两个条件同时为真时报警才触发，对范围检查很有用（例如值在两个限值之间）。

ORcondition

使用两个条件时，如果你需要 OR 逻辑而非 AND，可将 **ORcondition** 列设为“true”。采用 OR 逻辑时，“**v1 cond1 v2**”或“**v2 cond2 v3**”任一为真，报警即触发。默认为“false”（AND 逻辑）。

变化检测

如果 **v1** 和 **v2** 引用同一标签且条件为“!=”，模块将其视为变化检测器。每当引用的标签值变化时，报警会短暂触发，然后自动回到未活动并关闭。

示例

下面是一个完整示例，监视来自 Modbus 设备的温度和压力。

dxalm.json:

```
{
  "servers": [{ "url": "127.0.0.1:1883" }],
  "project": "PLANT1",
  "id": "alm",
  "meta": {
    "name": "dxalm"
  },
  "update": ["1000"],
  "task": [
    {
      "enable": true,
      "id": "process",
      "mode": "Alarm",
      "data": [
        { "$csv": "dxalm.csv" }
      ]
    },
    {
      "enable": true,
      "id": "events",
      "mode": "Audit",
      "data": [
        { "$csv": "dxalm-audit.csv" }
      ]
    }
  ]
}
```

dxalm.csv:

name	msg	v1	cond1	v2	cond2	v3	ORcondition	deadbandVal	deadbandType	description
temp_hi	Temperature High	\$mbus.stn1.temp	>	80				2		High temperature alarm
temp_hh	Temperature Very High	\$mbus.stn1.temp	>	95						Critical temperature alarm
pressure_range	Pressure Out of Range	\$mbus.stn1.pressure	>	20	<	80	false	5	%	Pressure outside normal range

dxalm-audit.csv:

name	msg	v1	cond1	v2	description
valve_change	Valve State Changed	\$mbus.stn1.valve	!=	\$mbus.stn1.valve	Log every valve state change
mode_switch	Mode Switched	\$mbus.stn1.mode	!=	0	Log when mode changes from zero

采用上述配置，模块将：

25. 监视温度标签，超过 80 时发出报警，固定死区为 2，以防止阈值附近快速切换。温度超过 95 时触发单独的严重报警。
26. 监视压力标签，同时高于 20 且低于 80 时发出报警（超范围检查），死区为 5%。
27. 在审计模式下，每次阀门状态变化记录一个事件，模式值变为非零时记录一个事件。
28. 报警标签名遵循“alm.process.temp_hi”、“alm.process.temp_hh”和“alm.process.pressure_range”模式。审计标签名遵循“alm.events.valve_change”和“alm.events.mode_switch”模式。

数据转发

DX Data Forward（dxforward）重定向来自任何其他 DX 模块的数据，将其保存为文本文件或作为电子邮件发送。它监听传入的标签数据，缓冲内容，在触发时写入指定的文件路径或通过 SMTP 发送。

关于 servers、project、模块 ID、trigger、update、data、存储转发和运行时等通用 DX 配置，请参阅 DX 驱动通用配置指南。

配置

下面是一个典型配置文件 dxforward.json 的示例。

```
{
  "servers": [{"url": "127.0.0.1:1883"}],
  "project": "DEM0",
  "id": "module01",
  "meta": {
    "name": "dxforward"
  },
  "task": [
    {
      "enable": true,
      "id": "forward01",
      "type": "text",
      "url": "",
      "trigger": ["1000"],
      "data": [
        { "$csv": "dxforward.csv" }
      ]
    }
  ]
}
```

type

任务的转发操作类型。

- **text**——将缓冲内容写入磁盘上的文本文件。在此模式下，url 参数指定输出目录。
- 任何其他值（如 html）——通过 SMTP 将缓冲内容作为电子邮件发送。在此模式下，url 参数指定 SMTP 服务器地址，邮件正文以 HTML 发送。

留空或省略时，模块默认为邮件模式。

url

url 的含义取决于 type 设置：

- 在 text 模式下，这是保存文件的输出目录路径。留空或省略时，使用模块的当前工作目录。
- 在 email 模式下，这是 host:port 格式的 SMTP 服务器地址（如 "smtp.gmail.com:587"）。

uid

发件人的电子邮件地址。发送邮件时用作“From”（发件人）地址。如果未指定 user，也用作 SMTP 身份验证用户名。仅适用于邮件模式。

user

可选的 SMTP 身份验证用户名。提供时，使用此值而非 uid 向 SMTP 服务器验证身份。当你的 SMTP 登录用户名与发件人邮箱地址不同时很有用。留空时，模块回退到 uid 进行身份验证。仅适用于邮件模式。

pwd

SMTP 身份验证的密码。提供时，模块使用 user（或未设置 user 时为 uid）和 pwd 凭据向 SMTP 服务器验证身份。留空时，模块不进行身份验证直接连接。仅适用于邮件模式。

reply

收件人的电子邮件地址。发送邮件时用作“To”（收件人）地址。仅适用于邮件模式。

security

控制 SMTP 服务器的连接安全性。接受值为：

- **tls**——连接后使用 STARTTLS 将连接升级为 TLS。
- **ssl**——从一开始就直接通过 SSL/TLS 连接。
- **（空）**——使用明文、不加密的连接。

仅适用于邮件模式。

attach

可选的文件路径模式列表，作为邮件附件包含。每个模式可包含`${tagName}`占位符，在发送时解析为当前标签值。支持文件名通配符（如“reports/*.pdf”附加reports 文件夹中的所有 PDF 文件）。路径相对于当前工作目录解析。仅适用于邮件模式。

例如：

```
"attach": ["reports/${fwd.alert.report_file}", "logs/*.csv"]
```

这会附加名称来自标签值的文件，以及 logs 文件夹中的所有 CSV 文件。

CSV

CSV 文件定义要转发的数据条目。默认的 CSV 列如下：

name	title	content	trigger
log1	output.txt	\${tagname}	1000

- **disable**——设为“true”可跳过此条目，设为“false”则启用。
- **name**——此条目的标签名。系统中的完整标签名遵循“moduleId.taskId.name”的命名约定。
- **title**——在文本模式下，这是输出文件名。在邮件模式下，这是邮件主题行。如果标题以\$开头，会被视为`${tagname}`格式的标签引用，标签的当前值将用作文件名或主题。在文本模式下，当动态文件名已存在时，会追加-yyyyMMdd-HHmmss 格式的时间戳后缀以避免覆盖。
- **content**——其数据将被缓冲的标签。以`${tagname}`格式指定，其中 tagname 是要监视的源标签。缓冲值会被连接，并在触发时写入文件或包含在邮件正文中。在文本模式下，如果内容包含\n，会在输出文件中被视为换行符。
- **trigger**——可选的每个条目触发器，控制何时转发缓冲内容。你可以使用毫秒时间间隔（如“5000”表示每 5 秒）或 cron 表达式。留空时，改用任务级触发器。
- **description**——可选的描述，供你参考。

触发器触发时，该条目的所有缓冲值被连接并转发（写入文件或作为邮件发送），缓冲区被清空。如果缓冲区中没有新数据，不转发任何内容。

示例——文本文件

下面是一个完整示例，将两个不同标签的数据转发到单独的文件。

dxforward.json:

```
{
  "servers": [{ "url": "127.0.0.1:1883" }],
  "project": "PLANT1",
  "id": "fwd",
  "meta": {
    "name": "dxforward"
  },
  "task": [
    {
      "enable": true,
      "id": "logs",
      "type": "text",
      "url": "C:\\Data\\Logs",
      "trigger": [ "5000" ],
      "data": [
        { "$csv": "dxforward.csv" }
      ]
    }
  ]
}
```

dxforward.csv:

name	title	content	trigger	description
temp	temperature.txt	\${fwd.logs.sensor_temp}		Temperature readings
status	status.txt	\${fwd.logs.device_status}	60000	Device status every minute

采用上述配置，模块将：

- 29. 在标签 fwd.logs.sensor_temp 和 fwd.logs.device_status 上监听传入数据。
- 30. 在值到达时进行缓冲。
- 31. 每 5 秒（使用任务级触发器）将缓冲的温度数据写入 C:\Data\Logs\temperature.txt。
- 32. 每 60 秒（使用条目级触发器，它为此条目覆盖任务级触发器）将缓冲的设备状态数据写入 C:\Data\Logs\status.txt。
- 33. 如果自上次写入以来没有新数据到达，文件不被修改。

示例——邮件

下面是一个示例，每 60 秒将缓冲的标签数据作为邮件发送，并带有文件附件。

dxforward.json:

```
{
  "servers": [{ "url": "127.0.0.1:1883" }],
  "project": "PLANT1",
  "id": "fwd",
  "meta": {
    "name": "dxforward"
  },
  "task": [
    {
      "enable": true,
      "id": "alert",
      "type": "html",
      "url": "smtp.example.com:587",
      "uid": "sender@example.com",
      "pwd": "password123",
      "reply": "recipient@example.com",
      "security": "tls",
      "attach": ["reports/*.pdf", "logs/{fwd.alert.logfile}"],
      "trigger": ["60000"],
      "data": [
        { "$csv": "dxforward-email.csv" }
      ]
    }
  ]
}
```

dxforward-email.csv:

name	title	content	description
alert1	Plant Alert	\${fwd.alert.alarm_summary}	Alarm summary email

采用上述配置，模块将：

- 34. 在标签 fwd.alert.alarm_summary 上监听传入数据。
- 35. 在值到达时进行缓冲。
- 36. 每 60 秒，如果有新数据，撰写一封主题为“Plant Alert”的 HTML 邮件，并通过 smtp.example.com:587 处的 SMTP 服务器使用 TLS 发送。
- 37. 附加 reports 文件夹中的所有 PDF 文件，以及名称由 fwd.alert.logfile 标签值解析得到的日志文件。
- 38. 如果自上次发送以来没有新数据到达，不发送邮件。

MQTT 桥接

DX MQTT Bridge (dxmqtt) 连接到外部 MQTT 代理，并使用 JSONPath 表达式将传入的 JSON 负载映射到 SCADA 标签。它还可以将标签值变化发布回外部代理，使其成为 SCADA 系统与任何兼容 MQTT 的数据源之间的双向桥梁。

关于 servers、project、模块 ID、trigger、update、data、存储转发和运行时等通用 DX 配置，请参阅 DX 驱动通用配置指南。

配置

下面是一个典型配置文件 dxmqtt.json 的示例。

```
{
  "servers": [{"url": "127.0.0.1:1883"}],
  "project": "DEM0",
  "id": "mqtt1",
  "meta": {
    "name": "dxmqtt"
  },
  "update": ["1000"],
  "task": [
    {
      "enable": true,
      "id": "plc1",
      "server": {"url": "192.168.1.100:1883"},
      "topic": "sensor/data",
      "readOnly": true,
      "retain": true,
      "data": [
        { "$csv": "dxmqtt.csv" }
      ]
    }
  ]
}
```

update

控制何时将缓冲数据发布到 SCADA 服务器。你可以指定毫秒时间间隔（如“1000”表示每秒）或 cron 表达式。配置后，传入数据会累积在存储转发缓冲区中，并在指定间隔发布。未配置时，数据从外部代理到达后立即发布到 SCADA 服务器。

server

要连接的外部 MQTT 代理。这与顶层“servers”中配置的 SCADA 服务器不同。你在“url”字段中指定代理地址和端口。默认为“127.0.0.1:1883”。

server 对象支持以下字段：

- **url**——外部 MQTT 代理的主机和端口（如“192.168.1.100:1883”）。
- **username**——代理身份验证的用户名。可选。
- **password**——代理身份验证的密码。可选。
- **cert**——TLS 身份验证的客户端证书文件路径。可选。
- **ca**——TLS 连接的 CA 证书文件路径。可选。
- **sslVerify**——设为 true 可根据 ca 验证代理的 TLS 证书。默认 false（不验证证书）。
- **qos**——订阅和发布的 MQTT 服务质量级别。默认 0。
- **lwt**——遗嘱（Last Will）现在是 automatic 的——无需手动设置。如果模块意外断开，代理会发布一条预定义消息，使其他订阅者能检测到；如果代理不接受遗嘱，则不带遗嘱建立连接。

topic

在外部代理上订阅的 MQTT 主题。这决定任务将接收并处理哪些消息。你可以使用 MQTT 通配符：“+”匹配单一级别，“#”匹配多个级别。例如，“sensor/+”会匹配“sensor/temperature”和“sensor/humidity”，而“sensor/#”会匹配“sensor/”下的所有主题。

当任务配置为发布（readOnly 设为 false）时，主题不能包含通配符。模块会拒绝尝试向通配符主题发布的任务。

readOnly

设为 true 时，任务订阅外部代理并将传入的 JSON 负载映射到 SCADA 标签。设为 false 时，任务监听来自 SCADA 服务器的标签值变化，并在配置的主题上将其发布回外部代理。默认 true。

retain

设为 true 时，发布的消息在 MQTT 代理上被标记为保留，使新订阅者立即收到最后发布的值。设为 false 时，消息不保留。默认 true。

role

任务的可选角色名。指定时，数据以给定角色作为目的地发布，而非默认项目名。这使你能面向特定消费者组。未指定时，使用默认的基于项目的目的地。

CSV

CSV 文件定义 JSON 负载字段如何映射到 SCADA 标签。默认的 CSV 列如下：

name	address	time	description
\$.name	\$.value	\$.timestamp	value with timestamp

- **disable**——设为“true”可跳过此条目，设为“false”则启用。
- **name**——标签名，或从传入 JSON 负载中提取标签名的 JSONPath 表达式。设为普通字符串（不含“\$”）时，用作字面标签名。设为“\$”时，JSON 对象中所有顶层属性名成为各自的标签名。设为“\$.deviceName”这样的 JSONPath 时，标签名从负载中的该字段提取。你也可以用“+”作为分隔符连接多个片段，混合字面文本和 JSONPath 表达式（如“\$.site + . + \$.device”产生“siteA.device1”这样的标签名）。完整标签名遵循“moduleId.taskId.name”约定。
- **address**——从传入 JSON 负载中提取标签值的 JSONPath 表达式（如“\$.value”、“\$.data.temperature”）。当 readOnly 为 false（发布模式）时，这定义用于构建外发负载的 JSON 结构。
- **time**——从传入 JSON 负载中提取每个值时间戳的 JSONPath 表达式（如“\$.timestamp”、“\$.ts”）。对数组使用逐元素路径——例如“\$.HData[*].ts”与“\$.HData[*].value”一起——使一批中的每个样本保留自己的测量时间。接受格式：标准 ISO 8601 日期时间（如“2026-08-04T06:00:00Z”）和紧凑形式“yyyyMMddTHH:mm:ssZ”（如“20260804T060000Z”），两者都允许可选的小数秒和时区偏移（“Z”、“+hh:mm”或“+hhmm”）。留空、或值的时间戳无法解析时，使用当前系统时间。
- **description**——可选的描述，供你参考。

当传入负载是 JSON 数组时，数组中的每个元素使用相同的映射规则单独处理。

示例

下面是一个完整示例，从外部 MQTT 代理读取温度和湿度数据，并将一个设定值发布回该代理。

dxmqtt.json:

```
{
  "servers": [{ "url": "127.0.0.1:1883" }],
  "project": "PLANT1",
  "id": "mqtt1",
  "meta": {
    "name": "dxmqtt"
  },
  "update": ["1000"],
  "task": [
    {
      "enable": true,
      "id": "sensor",
      "server": { "url": "192.168.1.100:1883" },
      "topic": "factory/sensors/#",
      "readOnly": true,
      "retain": true,
      "data": [
        { "$csv": "dxmqtt-read.csv" }
      ]
    },
    {
      "enable": true,
      "id": "ctrl",
      "server": { "url": "192.168.1.100:1883" },
      "topic": "factory/setpoint",
      "readOnly": false,
      "retain": true,
      "data": [
        { "$csv": "dxmqtt-write.csv" }
      ]
    }
  ]
}
```

dxmqtt-read.csv:

name	address	time	description
\$.name	\$.value	\$.timestamp	Sensor readings

dxmqtt-write.csv:

name	address	description
setpoint	\$.temperature	Temperature setpoint

采用上述配置，模块将：

39. 订阅外部代理 192.168.1.100 上“factory/sensors/”下的所有主题。当消息到达、负载如 {“name”:“temp1”,“value”:25.3,“timestamp”:“2025-01-15T10:30:00Z”} 时，创建名为“mqtt1.sensor.temp1”的标签，值为 25.3 及所提供的时间戳。
40. 缓冲所有传入的传感器数据，并按更新间隔配置每 1 秒发布到 SCADA 服务器。
41. 监听来自 SCADA 服务器的标签“mqtt1.ctrl.setpoint”的变化。值变化时，将如 {“temperature”:22.0} 的 JSON 负载发布到外部代理的“factory/setpoint”主题。

REST API 客户端

文件查找（File Lookup）功能让你直接在 `body` 模板中从外部 CSV 文件解析值。在通过 `dxrestapicli` 发送通知时，它最适合从 IGX 导出的用户列表中查找收件人数据，如电话号码或电子邮件地址。

该功能作为 `dxcore` 中的共享解析器实现，因此任何选择使用它的模块都可以。目前 `dxrestapicli` 在 POST 和 PUT 请求的 `body` 字段中使用它。

语法

```
${file:///path/to/file.csv?column=value#returnColumn}
```

- `file:///`——将占位符标记为文件查找。
- `path/to/file.csv`——（相对于模块配置文件目录的）相对路径，或以盘符开头的绝对路径，如 `C:/data/file.csv`。
- `?column=value`——过滤器。你可以用 `&` 组合多个子句表示 AND，用 `|` 表示 OR。AND 比 OR 结合更紧，因此 `a=1&b=2|c=3&d=4` 表示 `(a=1 AND b=2) OR (c=3 AND d=4)`。
- `#returnColumn`——其值被返回的列。

结果是来自每个匹配行的值的 JSON 数组，例如 `[“0111”,“0222”]`。如果没有行匹配，结果是空数组 `[]`。

匹配规则：仅匹配 `${token}` 形式

只有当 CSV 单元格包含字面 `${X}` 作为逗号分隔的 token 时，单元格才匹配过滤器值 X。单元格中的普通文本被视为描述，不匹配。

CSV cell	Filter description=Group_2	Match
<code>\${Group_2}</code>		Yes
<code>\${Moving_Average}, \${Group_2}</code>		Yes (either token matches)
<code>\${Group_22}</code>		No (different token)
<code>Group_2</code> without braces		No (treated as description)

此规则与 IGX 用户 CSV 约定一致，其中 Description 列使用 `${GroupName}` 声明用户属于哪个报警组。

列名会被规范化：不区分大小写，空格变为下划线。因此，标题为 Full Name 的列可在过滤器中以 full_name 引用。

嵌套占位符

你可以在文件查找过滤器中使用另一个占位符。内部占位符先解析，其值字面上插入过滤器。

```
${file:///users.csv?description=${alarmGroupPlaceholder}#phone}
```

如果标签 alarmGroupPlaceholder 的值为 Group_2，过滤器变为 description=Group_2，并在 CSV 中搜索 description 包含 token\${Group_2} 的行。

内部标签的值字面上插入，因此即使标签值本身看起来像 \${something}，也作为文本保留，不会再次求值。

JSON 数组上下文

当查找返回数组时，body 中周围的“...”和[...]被吸收，使数组干净地位于 JSON 值位置。

- “to_user”: “\${file:///...#phone}” 变为 “to_user”: [“0111”, “0222”]。
- “to_user”: [“\${file:///...#phone}”] 也变为 “to_user”: [“0111”, “0222”]。

你不需要选择特定的包装风格；两者都可行。

缓存与重载

模块在首次使用时读取每个 CSV 文件并保留在内存中。如果你编辑 CSV，模块会注意到文件变化并自动重载。编辑用户 CSV 后无需重启模块。

端到端示例：IGX 报警通知

目标

当 IGX 中报警变为活动时，dxrestapicli 应 POST 一条消息，包含为该报警组注册的每个用户的电话号码。

文件

DEM0-user-User.csv（从 IGX 导出，与 dxrestapicli 配置放在一起）：

```
Name,Enable,Description,Phone,Email,...
Guest,Yes,${Group_1},123456,support@integraxor.com,...
Admin,Yes,${Group_2},234567,...
Supervisor,Yes,"${Moving_Average}, ${test}",345567,...
Operator01,Yes,...
Operator02,Yes,...
```

Description 列使用`${GroupName}`作为组成员标记。单个用户可通过列出多个逗号分隔的 token 属于多个组。

query.json（POST body 模板）：

```
{
  "template_id": 259657394290689,
  "agent_id": 1445076010,
  "to_all": false,
  "to_user": ["${file:///DEM0-user-
User.csv?description=${alarmGroupPlaceHolder}#phone}"],
  "user_type": "mobile"
}
```

dxrestapicli.json（任务在 IGX 标签变化时触发）：

```
{
  "task": [{
    "enable": true,
    "id": "dev02",
    "url": "https://api.example.com/",
    "path": "api/notify",
    "method": "post",
    "contentType": "application/json; charset=UTF-8",
    "header": [["Content-type", "application/json; charset=UTF-8"]],
    "body": "./query.json",
    "trigger": ["${alarmGroupPlaceHolder}"],
    "data": [ { "$csv": "dxrestapicli-create.csv" } ]
  }]
}
```

执行过程

42. IGX 发送标签 `alarmGroupPlaceHolder`，值为 `Group_2`（纯组名，无大括号）。
43. 任务触发。dxrestapicli 加载 query.json。
44. 内部`${alarmGroupPlaceHolder}`解析为 `Group_2`。文件查找中的过滤器变为 `description=Group_2`。
45. 在 CSV 中搜索 Description 包含 token`${Group_2}`的行。Admin 行匹配，返回电话 234567。

46. 结果数组["234567"]替换带引号的占位符，吸收周围的“...”和[...]。

47. 最终 body 为： {

```
    "template_id": 259657394290689,
```

```
    "agent_id": 1445076010,
```

```
    "to_all": false,
```

```
    "to_user": ["234567"],
```

```
    "user_type": "mobile"
```

```
}
```

48. dxrestapicli POST 此 body。你可以在模块的调试日志中看到完全解析后的 body。

如果你预期一个组有多个收件人，只需在 Description 中添加更多带\${Group_2}的用户——结果数组会自动增长。

限制

- 仅支持 CSV 文件。
- 字符&、|、=、#和?在过滤器中保留，不能出现在过滤器值内。
- 在整个\${value} token 上精确匹配。不支持子串匹配。

网络扫描器

DX Net Scanner（dxnetscan）检测网络上主机是否存在。它使用 Internet 控制消息协议（ICMP）、Internet 控制消息协议第 6 版（ICMPv6）和传输控制协议（TCP）。主机可通过 IPv4 或 IPv6 地址或主机名指定，可带或不带端口号。标签值表示以毫秒为单位的响应时间。值为 0 表示主机不存在、不回复或请求超时。用户应将 false、null 和 0 视为失败状态。

关于 servers、project、模块 ID、trigger、update、data、存储转发和运行时等通用 DX 配置，请参阅 DX 驱动通用配置指南。

配置

下面是一个典型配置文件 dxnetscan.json 的示例。

```
{
  "servers": [{"url": "127.0.0.1:1883"}],
  "project": "DEM0",
  "id": "netscan",
  "meta": {
    "name": "dxnetscan"
  },
  "task": [
    {
      "enable": true,
      "id": "grp1",
      "trigger": [1000],
      "timeout": 1000,
      "data": [
        { "$csv": "dxnetscan.csv" }
      ]
    }
  ]
}
```

协议检测

每个主机使用的协议根据地址格式自动确定：

Address Type	Protocol
IPv4 address without a port number	ICMP
IPv4 address with a port number	TCP
IPv6 address without a port number	ICMPv6
IPv6 address with a port number	TCP
Host name without a port number	ICMP or ICMPv6 depending on its resolved IP address

Host name with a port number	TCP
------------------------------	-----

主机名首先由名称解析协议（如域名系统 DNS）解析为其 IP 地址。

timeout

等待每个主机回复的最长时间（毫秒）。默认值为 1000 ms。允许的最小值为 100 ms。较低的超时对每个标签值的变化提供更快响应，但在较慢网络上可能导致假阴性。如果你在广域网或不可靠连接上扫描主机，请增大此值。

CSV

CSV 文件定义要扫描的主机列表。默认的 CSV 列如下：

name	description	url
pc1	IPv4 use ICMP	192.168.1.1
pc2	IPv4 with port use TCP	192.168.1.2:502
pc3	Host name use ICMP	controlroomA
pc4	Host name with port use TCP	controlroomB:3000
pc5	IPv6 use ICMPv6	fe80::8487:f216:67a1:bf16%16
pc6	IPv6 with port use TCP	[fe80::d8fd:be:d409:1cb5%13]:5432

- **disable**——设为“true”可跳过此条目，设为“false”则启用。
- **name**——此主机的标签名。系统中的完整标签名遵循“moduleId.taskId.name”的命名约定。
- **description**——可选的描述，供你参考。
- **url**——要扫描的主机名或 IP 地址。附加冒号和端口号可使用 TCP 而非 ICMP。对于带端口的 IPv6 地址，在端口前将地址用方括号括起（如 [address]:port）。

每个条目的标签值是以毫秒为单位的响应时间。主机成功回复时，值为往返时间。主机不回复或请求超过配置超时时间，值为 0。用户应将 false、null 和 0 视为失败状态。

鼓励用户将主机尽可能分组为较少的任务。创建许多独立任务会消耗不必要资源并拖慢系统。

示例

下面是一个完整示例，扫描本地和远程主机的混合。

dxnetscan.json:

```
{
  "servers": [{ "url": "127.0.0.1:1883" }],
  "project": "PLANT1",
  "id": "netscan",
  "meta": {
    "name": "dxnetscan"
  },
  "task": [
    {
      "enable": true,
      "id": "local",
      "trigger": [1000],
      "timeout": 1000,
      "data": [
        { "$csv": "dxnetscan.csv" }
      ]
    },
    {
      "enable": true,
      "id": "remote",
      "trigger": [5000],
      "timeout": 3000,
      "data": [
        { "$csv": "dxnetscan-remote.csv" }
      ]
    }
  ]
}
```

dxnetscan.csv:

name	description	url
plc1	Main PLC	192.168.1.10
plc2	Backup PLC	192.168.1.11
hmi1	Operator Station	192.168.1.20

dxnetscan-remote.csv:

name	description	url
gateway	Remote Gateway	10.0.0.1
cloudsvr	Cloud Server	cloud.example.com:443

采用上述配置，模块将：

- 49. 每 1 秒扫描三个本地主机，超时 1 秒。标签名为“netscan.local.plc1”、“netscan.local.plc2”和“netscan.local.hmi1”。

50. 每 5 秒扫描两个远程主机，超时 3 秒。标签名为“netscan.remote.gateway”和“netscan.remote.cloudsvr”。
51. 主机回复时，每个标签发布以毫秒为单位的响应时间；主机不回复或请求超时发布 0。

第四部分 历史与数据库

历史记录器

DX Historian Logger (dxhislog) 将实时标签数据和报警/审计事件记录到磁盘上的本地日志文件。它将 onchange 数据和报警事件存储为按日期和小时组织的 JSON 格式文件，适合无需数据库的轻量级历史归档。

关于 servers、project、模块 ID、trigger、update、data、存储转发和运行时等通用 DX 配置，请参阅 DX 驱动通用配置指南。

配置

下面是一个典型配置文件 dxhislog.json 的示例。

```
{
  "servers": [{ "url": "127.0.0.1:1883" }],
  "project": "DEM0",
  "id": "his",
  "meta": {
    "name": "dxhislog"
  },
  "update": ["10000"],
  "task": [
    {
      "enable": true,
      "id": "log",
      "url": "./data/",
      "mode": "log",
      "trigger": [60000],
      "data": [
        { "$csv": "dxhislog.csv" }
      ]
    }
  ]
}
```

url（存储路径）

日志文件将写入的文件系统路径。可以是绝对路径，或相对于配置文件位置的路径。

此字段留空或省略时，模块会在与配置文件相同的目录中自动创建一个 **data** 子文件夹，并将所有日志文件存储在那里。

新的 **dxhislog** 模块创建时带有“url”: “./data/”，与留空时相同的文件夹；设置表单将此字段命名为 **Folder**，与 **dxhisagg** 和 **dxhisque** 一样。url 为空的配置保持不变继续工作。

你可以将其设为任何有效的目录路径，例如：

```
D:/scada/history  
./mydata
```

mode

记录模式决定任务将捕获哪种类型的数据。有两种可用模式：

- **log**——这是默认模式。任务订阅 **onchange** 数据，在每个时间戳存储每个标签的数值。数据写入每小时日志文件。模块停止时，每个标签的日志在其最后值处获得一个 **shutdown** 间隔，因此可将停止与设备离线区分开；模块在同一小时内再次启动时保留该间隔。
- **alarm**——任务订阅 **onalm** 和 **onaudit** 数据，存储报警和审计事件条目。报警数据写入半天日志文件，审计数据写入每日日志文件。

如果需要捕获两种类型数据，通常配置一个“log”模式任务和一个单独的“alarm”模式任务。

trigger

触发器控制累积数据从内存刷到磁盘的频率。你可以将其指定为毫秒时间间隔，或使用 **cron** 表达式。

当你不希望任何内容保存到磁盘时，将 **trigger** 留空。任务随后以 **buffer mode** 运行：每个标签只在内存中保留最后已知值，不按定时器写入磁盘。当你只需要提供最后有效值查询而非构建完整历史时很有用。

当 **trigger** 设为间隔（例如 60000 表示每 60 秒）时，数据会按该间隔刷到磁盘。此外，会自动注册一个内部每小时计划，确保无论 **trigger** 设置如何，数据至少每小时写入一次。

update

触发器控制写入磁盘的内容；**update** 控制模块重新发布其保留的最后有效值的频率，这是其他模块和 Web 前端启动时读取的内容。将其设为你希望该值刷新的间隔，例如[“10000”]表示每 10 秒。在 DX Edit 中添加的模块以[“5000”]开始（每 5 秒）。

启动时，模块还会从发布它的模块获取每个列出标签的最后有效值，因此长时间未变化的标签仍立即处于其最后有效值（并在其 **Watch** 选项卡中）。该值不会作为新记录写入日志；标签从其下次变化开始记录。

当你不需要保留的最后有效值时，将 **update** 留空。这两个设置相互独立，因此你可以将 **trigger** 留空，仍设置 **update** 以提供最后有效值而不向磁盘写入任何历史。

filter（按间隔聚合，用于趋势）

默认情况下，log 任务记录它收到的每个 **onchange** 值。对于趋势显示，你通常需要更小、等间距的点集。**filter** 参数将任务变为按间隔聚合器：设置一个正常间隔（和/或 **cron**）触发器，然后列出要保留的聚合。

```
"trigger": ["300000"],  
"filter": ["end", "min", "max"]
```

设置 **filter** 后，任务不再缓冲每个值——它在每个标签上只保留运行中的首个/最后/最小/最大值（与最后有效值一起保留在内存中）。在每次间隔（或 **cron**）触发时，它为每个 CSV 标签记录配置的点，每个点以其实际发生时间戳记，然后重置窗口，使下一个值开始新窗口：

- **bgn**——窗口中记录的第一个值（开盘）。
- **end**——最后记录的值/最后有效值（收盘）。
- **min**——窗口中的最低值。
- **max**——窗口中的最高值。

因此 `filter: ["end"]` 每间隔为每个标签记录一个点，而 `filter: ["bgn","end","min","max"]` 最多记录四个。这为你提供例如每小时聚合无法提供的 5 分钟最小/最大值。

Null values are always recorded，即使在 `filter` 模式下。`null`（标签离线）绕过聚合并在其实际时间直接写入日志，因此趋势仍显示不连续。

条件触发器（按标签条件记录）

与间隔 `filter` 分开，触发器可在标签满足条件时立即记录快照。条件触发时，`current last-good value of every tag listed in the CSV` 被记录（如果 CSV 未列出则只记录触发标签）：

- **`${tag}>v`**，**`${tag}<v`**，**`${tag}>=v`**，**`${tag}<=v`**，**`${tag}=v`**，**`${tag}!=v`**——在比较成立的每个样本上触发。每个条件写单个值；对于 OR，列出多个条件。
- **AND-groups**——将多个比较包装在嵌套数组中以要求它们同时成立。
`["${Pump.On}=1", "${Flow}<10"]` 仅在 `Pump.On` 为 1 且 `Flow` 低于 10 时触发。
顶层条目以 OR 组合（任一个触发）。

条件针对每个传入值、仅对照引用了变化标签的条件进行评估，因此在大量标签下保持高效。`filter` 聚合和条件快照写入相同的每小时文件，并由 `DX Historian Query (dxhisque)` 像正常记录一样发布。当既未配置 `filter` 也无条件时，任务像以前一样记录每个值。

role

留空（默认）时，任务处理所有角色的数据。你可以将其设为特定角色名，使只记录匹配该特定角色的数据。当你有多个角色发布数据、希望将它们分到不同文件夹时很有用。

角色激活时，日志文件存储在存储路径内以角色命名的子文件夹下。

format

控制日志文件中使用的时戳格式。有两个选项：

- **epoch**（默认）——时间戳存储为 Unix 纪元毫秒（UTC）。字段留空或省略时为默认行为。
- **iso**——时间戳以 ISO 8601 格式存储。

CSV

CSV 文件充当可选的标签过滤器。列如下：

name	description
Plant1/Pump.Pressure	Pump pressure sensor
Tank[1].Level	Tank level measurement

- **disable**——设为“true”可跳过此标签条目，设为“false”则启用。
- **name**——要记录的标签名。必须与发布的标签名匹配。名称完全按书写使用；绝不以此模块自身的模块 ID 或任务 ID 为前缀。
- **description**——可选的描述，供你参考。

仍带有额外列的文件继续工作——模块读取它们，编辑器只是不显示。

当 CSV 文件为空或不包含已启用条目时，订阅主题中所有收到的标签都会被记录。在 CSV 中列出标签时，只记录匹配的标签。这使你能从大量可用数据中选择性记录特定标签。

时间戳处理

每个值带有其来源提供的时间戳，该时间戳决定值存储在哪个日志文件——而非值到达模块的时间。

- **Past timestamps are kept, however old they are.** 值存储在其被测量的日期和小时下。当使用存储转发的设备重新连接并发送离线期间缓冲的读数时，该数据归档在原始测量日期下，而非重新连接时间。
- **A future timestamp up to 5 seconds ahead is kept.** 少量超前是正常的——例如操作员条目日期超前几秒以覆盖轮询字段值——因此值存储在其自己的时间下。
- **A timestamp more than 5 seconds ahead is corrected.** 这被视为来源时钟错误，因此值改为存储在当前时间。时钟超前的标签在日志中命名，例如 Source time is >5s faster: \${tag01} \${tag02} \${tag03}。

由于数据在任务触发时写入磁盘，迟到值首先写入当前小时文件，然后移动到自己日期和小时的文件。此整理由 DX Historian Aggregator (dxhisagg) 在其自己任务每次运行时执行，因此回溯日期值在到达小时关闭前留在到达日期下。如果你不运行 dxhisagg，值留在它们首次到达的文件中。

如果你所有数据都回溯到达，例如来自每天上传一次的设备，dxhisagg 指南的“示例：每天到达一次的数据”一节会介绍相关设置。

文件结构

模块根据模式、角色和时间在磁盘上组织日志文件。

log 模式文件

在 log 模式下，数据文件按以下结构存储在存储路径下：

```
{url}/{role}/-yyyymmdd-HH.log
```

每个文件覆盖一小时数据。文件名包含 UTC 日期和小时。例如，-20260330-14.log 包含 2026 年 3 月 30 日 UTC 14:00 至 14:59 的数据。

如果未配置角色，文件直接存储在存储路径下。角色激活时，文件放在以角色命名的子文件夹下。

每个角色文件夹中还维护一个 datacount.log 文件，记录每天存储的数据点总数。

alarm 模式文件

在 alarm 模式下，报警数据文件存储在：

```
{url}/his.log.alarm/{role}/{state}{yyyymmdd}-{section}.log
```

每个报警文件覆盖半天。{section} 为 1 表示午夜到中午，为 2 表示中午到午夜。{state} 前缀为-表示正常报警数据，为+表示正在处理报警关闭。

审计模式文件

在 alarm 模式下，审计事件数据文件存储在：

```
{url}/his.log.audit/{role}/{yyyymmdd}.log
```

每个审计文件覆盖一整天。审计条目作为原始 JSON 行追加。

示例

下面是一个完整示例，将 `onchange` 和报警数据都记录到本地文件。

`dxhislog.json`:

```
{
  "servers": [{ "url": "127.0.0.1:1883" }],
  "project": "PLANT1",
  "id": "his",
  "meta": {
    "name": "dxhislog"
  },
  "update": ["10000"],
  "task": [
    {
      "enable": true,
      "id": "data",
      "url": "./data/",
      "mode": "log",
      "trigger": [60000],
      "data": [
        { "$csv": "dxhislog.csv" }
      ]
    },
    {
      "enable": true,
      "id": "alm",
      "url": "./data/",
      "mode": "alarm",
      "trigger": [60000],
      "data": [
        { "$csv": "dxhislog-alm.csv" }
      ]
    }
  ]
}
```

采用上述配置且 CSV 文件为空，模块将：

52. 在配置文件目录中创建一个 `data` 子文件夹。
53. 在第一个任务的存储路径中将所有 `onchange` 标签值记录到每小时 JSON 日志文件。
54. 在第二个任务的存储路径中将所有报警和审计事件分别记录到半天和每日日志文件。

如果项目有 `Pump.Pressure`、`Tank.Level` 和 `AHU.Status` 这样的标签，你会找到如 `-20260330-14.log` 这样的每小时日志文件，包含这些标签的时间序列数据。

历史查询

DX Historian Query (dxhisque) 为 DIOT 项目执行数据查询。它从本地存储路径读取历史日志文件，并将数据发布到服务器。模块监视存储文件夹中新的或更新的日志文件，并自动发布。它处理日志数据、报警数据、审计数据和报表数据。

关于 servers、project、模块 ID、trigger、update、data、存储转发和运行时等通用 DX 配置，请参阅 DX 驱动通用配置指南。

配置

下面是一个典型配置文件 dxhisque.json 的示例。

```
{
  "servers": [{"url": "127.0.0.1:1883"}],
  "project": "DEM0",
  "id": "query1",
  "meta": {
    "name": "dxhisque"
  },
  "rotation": false,
  "task": [
    {
      "enable": true,
      "id": "query",
      "url": "",
      "trigger": [500],
      "buffer": 90,
      "data": [
        { "$csv": "dxhisque.csv" }
      ]
    }
  ]
}
```

rotation

控制发布的日志主题是否包含基于日期的路径段。设为 true 时，主题路径将包含从日志文件名派生的年、月、日段（例如 onlog/2025/03/30）。设为 false 时，所有日志数据发布在扁平的 onlog 主题下。默认值为 false。

由于日期段来自日志文件名，它们遵循数据当前存储的日期。仍归档在到达日期下的存储转发数据因此在该日期下发布，直到 DX Historian Aggregator (dxhisagg)

将其移动到位，之后更正的文件在正确日期下再次发布。如果你的大部分数据回溯到达，请参阅 `dxhisagg` 指南的“示例：每天到达一次的数据”一节。

url

历史日志文件存储的本地文件夹路径。模块会扫描此文件夹及其子目录寻找日志文件。如果文件夹不存在，会自动创建。留空时，模块默认为相对于配置文件位置的“data”子文件夹。

trigger

模块检查排队文件请求并发布缓冲数据的间隔（毫秒）。默认值为 1000 ms。较低值提供更快新数据发布，但增加处理频率。

buffer

启动时加载并发布的历史数据天数。默认值为 90 天，因此模块启动时只发布最后 90 天数据。将其设为任何其他正数以扩大或缩小窗口——例如，值为 7 表示只发布最后 7 天数据。设为 0 可发布所有内容，启动时会将存储路径中每个现有日志文件排队。模块还使用此值每天清理过期的保留主题，删除早于配置缓冲窗口的数据。

数据存储结构

模块期望日志文件按以下文件夹结构组织在配置的 `url` 路径下：

- 根文件夹——包含日志数据文件和 `datacount.log` 索引文件。
- 角色子文件夹——如果你的项目使用角色，每个角色有自己的子文件夹，布局相同。
- `his.log.alarm/`——包含报警日志文件。基于角色的报警数据存储在 `his.log.alarm/{role}/` 子文件夹中。
- `his.log.audit/`——包含审计日志文件。基于角色的审计数据存储在 `his.log.audit/{role}/` 子文件夹中。
- `his.log.report/`——包含报表日志文件，按模块 ID、任务 ID 和可选的 CSV 名称嵌套。基于角色的报表位于 `his.log.report/{role}/` 中，仅发布到该角色的目的地；报表主题本身从不携带角色名。

每个文件夹中的 `datacount.log` 文件充当索引，跟踪每个日志文件的记录数。模块在响应查询请求时使用此索引确定哪些文件有新数据。当文件夹尚无 `datacount.log` 时，模块仍在启动时发布其缓冲区中的历史，并在其状态日志中注明缺失索引。

CSV

CSV 文件定义模块的条目列表。默认的 CSV 列如下：

<code>disable,name,description</code>

- **disable**——设为“true”可跳过此条目，设为“false”则启用。
- **name**——此条目的标签名。系统中的完整标签名遵循“`moduleId.taskId.name`”的命名约定。
- **description**——可选的描述，供你参考。

示例

下面是一个完整示例，从本地存储路径查询历史数据。

`dxhisque.json`:

```
{
  "servers": [{ "url": "127.0.0.1:1883" }],
  "project": "PLANT1",
  "id": "query1",
  "meta": {
    "name": "dxhisque"
  },
  "rotation": true,
  "task": [
    {
      "enable": true,
      "id": "query",
      "url": "C:/ProjectData/PLANT1/data",
      "trigger": [500],
      "buffer": 30,
      "data": [
        { "$csv": "dxhisque.csv" }
      ]
    }
  ]
}
```

`dxhisque.csv`:

name	description
default	Default query role

采用上述配置，模块将：

- 55. 监视文件夹“C:/ProjectData/PLANT1/data”中的日志、报警、审计和报表文件。
- 56. 启动时，发布最后 30 天的所有历史数据。
- 57. 使用基于日期的主题轮换，使日志数据发布在如“onlog/2025/03/30”的主题下，而非扁平的“onlog”主题。
- 58. 存储文件夹中出现新的或更新的文件时自动检测并发布。
- 59. 每天清理早于 30 天数据的过期保留主题。

历史聚合器

DX Historian Aggregator (dxhisagg) 为 DIOT 项目执行数据聚合。它处理原始历史日志文件，将每日数据提升为每月和每年摘要，提取最小值、最大值等关键数据点。它还处理已关闭报警的合并、按年龄清理数据，以及从 HTML 布局定义生成报表。

关于 servers、project、模块 ID、trigger、update、data、存储转发和运行时等通用 DX 配置，请参阅 DX 驱动通用配置指南。

配置

下面是一个典型配置文件 dxhisagg.json 的示例。

```
{
  "servers": [{"url": "127.0.0.1:1883"}],
  "project": "DEM0",
  "id": "log1",
  "meta": {
    "name": "dxhisagg"
  },
  "task": [
    {
      "enable": true,
      "id": "device1",
      "url": "",
      "trigger": ["600000"],
      "timeout": "60000",
      "type": "promote",
      "purge": ["365d"],
      "data": [
        { "$csv": "dxhisagg.csv" }
      ]
    }
  ]
}
```

url

历史日志文件存储的数据目录路径。留空时，模块会查找相对于配置文件位置的“data”文件夹。

type

此任务的聚合模式。有两个可能值：

- **promote**——默认模式。模块扫描已完成的每日日志文件（来自之前日期），并将数据提升为每月和每年摘要文件。提升期间，它为每年摘要提取最重要的数据点，如最小值、最大值和任何 **null** 转换，为每月摘要提取最大时间间隔和值间隔。
- **batch**——在此模式下，由标签列表触发时，模块将合并的每日日志文件中的数据拆分为单独的标签级文件。当你需要从大型合并日志中提取特定标签单独处理时很有用。

trigger

触发器控制聚合任务运行频率。默认值为 **600000 ms**（10 分钟）。你可以使用通用 DX 配置支持的任何触发器格式，包括固定毫秒间隔、**cron** 表达式或基于标签的触发器。

timeout

用于检测数据间隔的超时值（毫秒）。当两个连续数据点之间的时间间隔超过此值时，模块插入 **null** 断点以指示通信中断。默认值为 **60000 ms**（1 分钟）。设为 **0** 可禁用超时断点插入。

如果原始数据包含 **shutdown** 指示符且数据在超时期间内恢复，**shutdown** 指示符会自动移除，因为它不代表真正断开。

purge

控制旧数据如何清理的 **purge** 规则列表。目前支持的格式是数字后跟字母“**d**”表示天数。例如，“**365d**”表示早于 365 天的日志文件将被删除。文件根据其文件名中编码的日期清理。

topic

模块订阅的额外主题的可选列表。你可以指定单个字符串或字符串数组。

CSV

CSV 文件定义聚合任务的额外数据条目。默认的 CSV 列如下：

```
disable,name,url,description,purge
```

- **disable**——设为“**true**”可跳过此条目，设为“**false**”则启用。

- **name**——用于参考的可选名称。
- **url**——HTML 报表布局文件的路径。指定时，模块会解析 HTML 文件中的报表布局定义，并基于布局生成聚合报表文件。HTML 文件可使用 `data-rpt='${tag@scope.attr}'` 语法定义要计算的标签和聚合属性。HTML 注释（`<!-- ... -->`）内的任何内容被忽略，因此你可注释掉单元格以停止计算。
- **description**——可选的描述，供你参考。
- **purge**——此特定条目的可选 **purge** 规则。

存储转发数据

值并不总是按测量顺序到达历史记录器。使用存储转发的设备可能离线一段时间，稍后发送缓冲读数，这些读数属于更早日期。此任务在发生时保持存储历史正确。

任务每次运行时，检查自上次运行以来变化的日志文件，并将存储在错误日期或小时下的任何值移动到其实际测量时间的文件。每天和模块每次启动时还会对每个日志文件运行一次完整检查。没有年龄限制——缓冲数周的读数仍归档在其测量当天，完全按其到达时一样。当前正在写入的小时文件保持不动，因此属于当前小时的值在该小时关闭后移动。

由于此检查在任务触发器上运行，你的 **trigger** 间隔也是存储历史整理的速度。如果你的所有数据都按设计回溯到达，请参阅本指南末尾的示例：每天到达一次的数据。

提升遵循相同原则。每个值提升到 **its own** 日期的每月和每年文件中，因此迟到的读数仍在其所属月份和年份下摘要。如果迟到数据到达已提升的日期，该日期会再次提升，并重建其对每月和每年摘要的贡献，因此总数保持正确，不会重复计数。

报表聚合

当 CSV 条目有指向 HTML 布局文件的 **url** 时，模块执行报表聚合。HTML 布局定义要聚合哪些标签和应用什么计算。支持的聚合属性如下：

Attribute	Description
avg	Average of all numeric values in the period
max	Maximum numeric value in the period
min	Minimum numeric value in the period
sum	Sum of all numeric values in the period

sub	Subtraction (maximum minus minimum) in the period
dur	Duration in milliseconds from first to last timestamp

如果未指定属性，使用最新值。

报表文件在数据路径下的 **his.log.report** 目录中生成。模块自动将报表从每小时时间间隔汇总为每日、每月和每年摘要文件。使用跨汇总级别的加权平均计算平均值以保持准确性。

示例

下面是一个完整示例，以 5 分钟间隔聚合历史数据并清理 1 年。

dxhisagg.json:

```
{
  "servers": [{"url": "127.0.0.1:1883"}],
  "project": "PLANT1",
  "id": "agg1",
  "meta": {
    "name": "dxhisagg"
  },
  "task": [
    {
      "enable": true,
      "id": "hist1",
      "url": "C:/data/historian",
      "trigger": ["300000"],
      "timeout": "30000",
      "type": "promote",
      "purge": ["365d"],
      "data": [
        { "$csv": "dxhisagg.csv" }
      ]
    }
  ]
}
```

dxhisagg.csv:

name	url	description
report1	report_layout.html	Daily production report

采用上述配置，模块将：

60. 每 5 分钟扫描“C:/data/historian”的数据目录，寻找已完成的每日日志文件。

61. 将之前日期的数据提升为每月和每年摘要文件，提取最小值、最大值和 null 转换等关键数据点。
62. 在这些运行中，将存储在错误日期下的任何值移动到其测量时间的文件，如果该日期已摘要则再次提升。
63. 数据间隔超过 30 秒时插入 null 断点，指示通信中断。
64. 删除早于 365 天的日志文件。
65. 解析“report_layout.html”文件中的标签定义，并在“his.log.report”子目录中生成聚合报表文件，自动从每小时汇总到每日、每月和每年级别。

示例：每天到达一次的数据

一些站点从非永久连接的设备收集数据。设备每 15 分钟记录一次读数，将读数保存在自己的内存中，每 24 小时上传整批。在这种设置下 every reading is backdated，不是偶尔而是按设计。

此示例介绍你会注意到什么、配置什么以及之后预期什么。

你会注意到什么

当一天的读数在一次突发中到达时，读数本身是正确的——每个都保留测量时间。但它们都写入突发到达小时的日志文件，因为那是当时正在填充的文件。在它们被移动之前，有两点看起来不对：

- 一整天的读数出现在到达日期下，而非测量日期。
- 如果 DX Historian Query (dxhisque) 启用了 rotation，数据在到达日期的主题下发布，因此请求测量日期的显示什么也不显示。

配置什么

DX Historian Logger (dxhislog) 无需改变——它已在测量时间存储每个读数。更正文件是此模块的工作，两个设置很重要。

dxhisagg.json:

```
{
  "servers": [{ "url": "127.0.0.1:1883" }],
  "project": "PLANT1",
  "id": "agg1",
  "meta": {
    "name": "dxhisagg"
  }
}
```

```
    },
    "task": [
      {
        "enable": true,
        "id": "hist1",
        "url": "C:/data/historian",
        "trigger": ["300000"],
        "timeout": "1800000",
        "type": "promote",
        "purge": ["365d"],
        "data": [
          { "$csv": "dxhisagg.csv" }
        ]
      }
    ]
  }
}
```

trigger——更正在任务触发器上运行，因此此间隔是文件整理的速度。10 分钟默认对大多数站点足够频繁；上面的 5 分钟只是使更正更快出现。cron 表达式也可行，例如["0 */5 * * * ?"]表示每 5 分钟。每次运行只重新检查自上次运行以来变化的文件，因此短间隔保持低成本。

timeout——将此设为 above your reading interval。它是插入 null 断点以标记通信丢失的间隔，而每 15 分钟采集的读数比 1 分钟默认间隔远得多。保持默认时，每对读数看起来都像通信丢失，因此每个读数后都插入断点。趋势随后在所有读数之间显示中断，由于这些中断被视为重要，它们挤满每年摘要并使每月摘要为空。上面的 30 分钟保持正常读数不动，同时仍标记真正错过的读数。如果你完全不想要断点，设为 0。

预期什么

当前正在写入的小时文件属于记录器，因此更正总是等待该小时结束。之后在任务下次运行时发生：

Burst delivered at	Readings moved
23:40	just after 00:00 – about 20 to 25 minutes later
23:05	just after 00:00 – about 55 to 60 minutes later

因此预期 the remainder of the arrival hour plus one trigger interval——平均大约半小时，最坏约一小时。等待小时关闭是有意的，因为记录器仍在写入该文件。

运行完成后：

- 每个读数位于其测量时间的每小时文件中，因此 2026 年 7 月 29 日 09:05 采集的读数在该日期和小时的文件 20260729-09.log 中找到。
- 这些日期的每月和每年摘要从更正文件重建，因此迟到的读数在其中表示。已摘要的日期会再次摘要而非重复计数，因此让模块运行不会膨胀摘要。
- datacount.log 为每个涉及日期更正。
- dxhisque 发布更正文件，因此数据在其所属日期下可用。

第二天上传无需更多操作——相同更正在每个触发器上运行。

历史到 SQL

DX Historian to SQL (dxhis2sql) 将历史日志数据复制到 SQL 数据库以供第三方访问。它监视历史日志文件的数据文件夹，文件处理后会重命名或跟踪以避免重复导入。

关于 servers、project、模块 ID、trigger、update 和 data，请参阅 DX 驱动通用配置指南。

配置

下面是一个典型配置文件 dxhis2sql.json 的示例。

```
{
  "servers": [{"url": "127.0.0.1:1883"}],
  "project": "DEM0",
  "id": "my",
  "meta": {
    "name": "dxhis2sql"
  },
  "update": ["1000"],
  "task": [
    {
      "enable": true,
      "id": "his2sql",
      "url": "",
      "role": [],
      "trigger": ["1000"],
      "data": [
        { "$csv": "dxhis2sql.csv" }
      ]
    }
  ]
}
```

url

你可以输入目标数据库的连接字符串。模块支持 SQLite、PostgreSQL、MySQL、Microsoft SQL Server、Oracle 和其他兼容 ODBC 的数据库。

此字段留空或省略时，模块使用本地 SQLite 数据库。你可以指定完整 ODBC 或原生连接字符串以连接到远程数据库。

下面是常见数据库的一些连接字符串示例：

SQLite（通过 ODBC）：

```
Provider=MSDASQL.1;Persist Security Info=False;Extended
Properties='driver={SQLite3 ODBC Driver};database=./mydata.db'
```

PostgreSQL（通过 ODBC）：

```
Driver={PostgreSQL
ANSI};Server=192.168.1.100;Port=5432;Database=scada;Uid=user;Pwd=pass;
```

MySQL（通过 ODBC）：

```
Driver={MySQL ODBC 8.0 Unicode
Driver};Server=192.168.1.100;Port=3306;Database=scada;User=user;Password=pass;
```

Microsoft SQL Server（通过 ODBC）：

```
Driver={ODBC Driver 17 for SQL
Server};Server=192.168.1.100;Database=scada;Uid=user;Pwd=pass;
```

Oracle（通过 ODBC）：

```
Driver={Oracle in
OraClient12Home1};Dbq=192.168.1.100:1521/ORCL;Uid=user;Pwd=pass;
```

role

role 参数接受角色名列表，决定监视哪些数据文件夹。每个角色对应配置文件旁边 data 目录下的一个子文件夹。例如，如果你设置“role”: [“engineer”]，模块会在 data/engineer/文件夹中查找历史日志文件。

留为空列表（默认）时，模块直接监视 data 文件夹，不带任何角色子文件夹。

报警和审计数据有两个特殊角色值：

- **his.log.alarm**——处理历史报警日志文件。报警数据存储时带有四个文本值列（V1、V2、V3、V4），而非单个值列。
- **his.log.audit**——处理历史审计日志文件。使用与报警数据相同的四列格式。

你可以在同一任务中组合多个角色，一次处理多个数据源：

```
"role": [ "", "his.log.alarm", "his.log.audit" ]
```

这会在一个任务中处理默认标签数据、报警日志和审计日志。

trigger

触发器间隔控制模块检查未处理历史日志文件的频率。默认值“1000”表示模块每秒扫描一次。除定期触发器外，模块还设置文件系统监视器，立即对新的或变化的.log 文件作出反应。

如果你希望模块较少频率地批量处理文件，可使用更长间隔。触发器格式遵循通用配置指南中描述的不同约定。

数据文件夹结构

模块从配置文件所在目录的 data 文件夹读取历史日志文件。期望结构为：

```
config-folder/
  dxhis2sql.json
  data/
    *.log           (default role files)
    processed.txt   (tracking file)
    engineer/       (role subfolder)
      *.log
      processed.txt
    his.log.alarm/   (alarm role subfolder)
      *.log
    his.log.audit/   (audit role subfolder)
      *.log
```

每个历史日志文件是 JSON 格式的.log 文件，包含带时间戳和值的标签数据。模块读取这些文件，将数据插入数据库，然后通过追加_sql 后缀重命名文件以防止重新处理。每个文件夹中的 processed.txt 文件也跟踪已导入的文件。

数据库模式

模块启动时自动创建表。表名取决于角色：

- 未指定角色时，表名为 dxhis2sql。
- 指定角色时，表以角色命名（例如 engineer）。
- 对于报警和审计角色，点替换为下划线（例如 his.log.alarm 变为 his_log_alarm）。

对于默认和自定义角色，每个表有以下列：

Column	Type	Description
K	VARCHAR(100)	Tag name (primary key, combined with T)

T	TIMESTAMP	Timestamp in local date-time format
U	BIGINT	Timestamp in Unix epoch milliseconds
V	VARCHAR(100)	Tag value

对于报警和审计角色，每个表有以下列：

Column	Type	Description
K	VARCHAR(100)	Tag name (primary key, combined with T)
T	TIMESTAMP	Timestamp in local date-time format
U	BIGINT	Timestamp in Unix epoch milliseconds
V1	VARCHAR(100)	First alarm/audit field
V2	VARCHAR(100)	Second alarm/audit field
V3	VARCHAR(100)	Third alarm/audit field
V4	VARCHAR(100)	Fourth alarm/audit field

所有表使用 (K, T) 复合主键。插入时忽略具有相同标签名和时间戳的重复记录。

CSV

CSV 文件目前不使用。

示例

下面是一个完整示例，将历史标签数据及报警和审计日志导入 PostgreSQL 数据库。

dxhis2sql.json:

```
{
  "servers": [{ "url": "127.0.0.1:1883" }],
  "project": "PLANT1",
  "id": "my",
  "meta": {
    "name": "dxhis2sql"
  },
  "update": ["1000"],
  "task": [
    {
      "enable": true,
      "id": "his2sql",
      "url": "Driver={PostgreSQL
ANSI};Server=192.168.1.100;Port=5432;Database=scada;Uid=user;Pwd=pass;",
      "role": ["", "his.log.alarm", "his.log.audit"],
      "trigger": ["5000"],
      "data": [
```

```
        { "$csv": "dxhis2sql.csv" }  
      ]  
    }  
  ]  
}
```

采用上述配置，模块将：

66. 连接到 192.168.1.100 的 PostgreSQL 数据库。
67. 创建三个表：标签数据的 dxhis2sql、报警数据的 his_log_alarm 和审计数据的 his_log_audit。
68. 每 5 秒扫描 data/、data/his.log.alarm/和 data/his.log.audit/文件夹，寻找新的历史日志文件。
69. 将每个日志文件的数据插入对应表，并以_sql 后缀重命名已处理文件。

SQL 记录器

DX SQL Logger（dxsqllog）将实时标签数据和报警/审计事件直接记录到数据库。它通过 ODBC 连接，支持 SQLite、PostgreSQL、MySQL、Microsoft SQL Server、Oracle 和主要国产数据库。每个标签存储在自动创建的专用表中——无需手动设置数据库。

关于 servers、project、模块 ID、trigger、update、data、存储转发和运行时等通用 DX 配置，请参阅 DX 驱动通用配置指南。

配置

下面是一个典型配置文件 dxsqllog.json 的示例。

```
{
  "servers": [{"url": "127.0.0.1:1883"}],
  "project": "DEM0",
  "id": "sqllog",
  "meta": {
    "name": "dxsqllog"
  },
  "update": ["10000"],
  "task": [
    {
      "enable": true,
      "id": "log",
      "url": "",
      "mode": "log",
      "trigger": [60000],
      "data": [
        { "$csv": "dxsqllog.csv" }
      ]
    }
  ]
}
```

url（连接字符串）

使用 ODBC、OleDB 或原生 SQL 连接格式连接到目标数据库的连接字符串。

此字段留空或省略时，模块会在当前工作目录中自动创建本地 SQLite 数据库文件。文件名遵循 {moduleId}.{taskId}.db 约定。例如，模块 ID 为“sqllog”、任务 ID 为“log”时，数据库文件为 sqllog.log.db。如果模块 ID 或任务 ID 不可用，则改用 JSON 配置文件名。

下面是常见数据库的一些连接字符串示例：

SQLite（通过 ODBC）：

```
Provider=MSDASQL.1;Persist Security Info=False;Extended  
Properties='driver={SQLite3 ODBC Driver};database=./mydata.db'
```

PostgreSQL（通过 ODBC）：

```
Driver={PostgreSQL  
ANSI};Server=192.168.1.100;Port=5432;Database=scada;Uid=user;Pwd=pass;
```

MySQL（通过 ODBC）：

```
Driver={MySQL ODBC 8.0 Unicode  
Driver};Server=192.168.1.100;Port=3306;Database=scada;User=user;Password=pass;
```

Microsoft SQL Server（通过 ODBC）：

```
Driver={ODBC Driver 17 for SQL  
Server};Server=192.168.1.100;Database=scada;Uid=user;Pwd=pass;
```

Oracle（通过 ODBC）：

```
Driver={Oracle in  
OraClient12Home1};Dbq=192.168.1.100:1521/ORCL;Uid=user;Pwd=pass;
```

DM / 达梦（通过 ODBC）：

```
Driver={DM8 ODBC  
DRIVER};Server=192.168.1.100;Port=5236;Database=scada;Uid=SYSDBA;Pwd=pass;
```

mode

记录模式决定任务将捕获哪种类型的数据。有两种可用模式：

- **log**——这是默认模式。任务订阅 onchange 数据，在每个时间戳为每个标签存储数值。非数值标签值存储为 0，布尔值存储为 1 或 0。
- **alarm**——任务订阅 onalm 和 onaudit 数据，在每个时间戳为每个报警或审计条目存储四个文本字段。

如果需要捕获两种类型数据，通常配置一个“log”模式任务和一个单独的“alarm”模式任务。

table

数据库中使用的数据库表名前缀。默认值为“dxsqllog”。由于表直接以标签名命名，此值目前保留供未来使用。

role

留空（默认）时，任务处理所有角色的数据。你可以将其设为特定角色名，使只记录匹配该特定角色的数据。当你有多个角色发布数据、希望将它们分到不同数据库任务时很有用。

角色激活时，表名将包含角色作为后缀，例如角色“engineer”下标签“Pump.Pressure”的 pump_pressure_engineer。

CSV

CSV 文件充当可选的标签过滤器。默认的 CSV 列如下：

name	description
Plant1/Pump.Pressure	Pump pressure sensor
Tank[1].Level	Tank level measurement

- **disable**——设为“true”可跳过此标签条目，设为“false”则启用。
- **name**——要记录的标签名。必须与通过 MQTT 发布的标签名匹配。
- **description**——可选的描述，供你参考。

当 CSV 文件为空或不包含已启用条目时，订阅主题中所有收到的标签都会被记录。在 CSV 中列出标签时，只记录匹配的标签。这使你能从大量可用数据中选择性记录特定标签。

数据库模式

首次收到数据时，模块自动为每个标签创建一个表。表根据标签名按以下规则命名：

- 所有字符转换为小写
- 特殊字符（点、斜杠、冒号、空格、连字符）替换为下划线
- 方括号被移除
- 连续下划线合并为一个

例如：

Original Tag Name	Table Name
Plant1/Pump.Pressure	plant1_pump_pressure
Tank[1].Level	tank1_level
AHU-01:Supply_Temp	ahu_01_supply_temp

在 log 模式下，每个表有以下列：

Column	Type	Description
time	INTEGER NOT NULL	Timestamp in Unix epoch milliseconds (UTC)
value	REAL NOT NULL	Numeric tag value

在 alarm 模式下，每个表有以下列：

Column	Type	Description
time	INTEGER NOT NULL	Timestamp in Unix epoch milliseconds (UTC)
v1	TEXT	First alarm/audit field
v2	TEXT	Second alarm/audit field
v3	TEXT	Third alarm/audit field
v4	TEXT	Fourth alarm/audit field

每个表的 time 列上创建索引，以确保快速时间范围查询。

还自动创建 tag_metadata 表，维护原始标签名与其对应清理后表名之间的查找。

Column	Description
tag_name	Original tag name as received from MQTT
table_name	Sanitized table name in the database
description	Reserved for tag description
unit	Reserved for engineering unit
poll_interval	Reserved for polling interval
retention_days	Reserved for data retention period

支持的数据库

模块从连接字符串自动检测数据库类型，并应用适当的 SQL 语法。支持以下数据库：

Database	Notes
SQLite	Default when no connection string is provided. Zero-config local storage. Performance optimizations are applied automatically.
PostgreSQL	Standard PostgreSQL via ODBC.
KingbaseES / 人大金仓	Compatible with PostgreSQL syntax.
GaussDB / openGauss	Compatible with PostgreSQL syntax.
HighGo	Compatible with PostgreSQL syntax.

MySQL	Standard MySQL via ODBC.
MariaDB	Compatible with MySQL syntax.
OceanBase	Compatible with MySQL syntax.
GBase	Compatible with MySQL syntax.
TiDB	Compatible with MySQL syntax.
Microsoft SQL Server	Standard MSSQL via ODBC or native driver.
Oracle	Standard Oracle via ODBC.
DM / 达梦	Compatible with Oracle syntax.
Shentong / 神通	Compatible with Oracle syntax.

示例

下面是一个完整示例，将 onchange 和报警数据都记录到本地 SQLite 数据库。

dxsqllog.json:

```
{
  "servers": [{ "url": "127.0.0.1:1883" }],
  "project": "PLANT1",
  "id": "sqllog",
  "meta": {
    "name": "dxsqllog"
  },
  "update": ["10000"],
  "task": [
    {
      "enable": true,
      "id": "data",
      "url": "",
      "mode": "log",
      "trigger": [60000],
      "data": [
        { "$csv": "dxsqllog.csv" }
      ]
    },
    {
      "enable": true,
      "id": "alm",
      "url": "",
      "mode": "alarm",
      "trigger": [60000],
      "data": [
        { "$csv": "dxsqllog-alm.csv" }
      ]
    }
  ]
}
```

采用上述配置且 CSV 文件为空，模块将：

70. 为第一个任务创建 sqllog.data.db，为第二个任务创建 sqllog.alm.db。

71. 在第一个数据库中将所有 `onchange` 标签值记录到每标签表。

72. 在第二个数据库中将所有报警和审计事件记录到每标签表。

如果项目有 `Pump.Pressure`、`Tank.Level` 和 `AHU.Status` 这样的标签，第一个数据库将包含名为 `pump_pressure`、`tank_level` 和 `ahu_status` 的表，以及 `tag_metadata` 查找表。

预 SQL

DX Pre-SQL（dxpresql）模块允许你对数据库执行预定义 SQL 语句，并将结果作为标签值获取。你在 CSV 数据文件中定义每个 SQL 查询，模块按配置的触发器间隔运行。查询结果序列化为 JSON 并作为标签值发布。你还可以在运行时通过向对应标签写入值来向 SQL 查询传递参数。

关于 servers、project、模块 ID、trigger、update、data、存储转发和运行时等通用 DX 配置，请参阅 DX 驱动通用配置指南。

配置

下面是一个典型配置文件 dxpresql.json 的示例。

```
{
  "servers": [{"url": "127.0.0.1:1883"}],
  "project": "DEM0",
  "id": "module01",
  "meta": {
    "name": "dxpresql"
  },
  "update": ["1000"],
  "task": [
    {
      "enable": true,
      "id": "presql01",
      "url": "Driver={MySQL ODBC 8.0 Unicode
Driver};Server=127.0.0.1;Database=mydb;Uid=root;Pwd=pass;",
      "trigger": ["1000"],
      "data": [
        { "$csv": "dxpresql.csv" }
      ]
    }
  ]
}
```

url

任务的数据库连接字符串。这是标准 ODBC 或 OLEDB 连接字符串，指定数据库驱动、服务器地址、数据库名和凭据。模块支持任何可通过 ODBC 或 OLEDB 连接的数据库，包括 MySQL、Microsoft SQL Server、PostgreSQL、Oracle、SQLite、Microsoft Access 等。如果连接字符串无效或数据库不可达，模块会记录错误。

CSV

CSV 文件定义要执行的 SQL 查询列表。列如下：

name	sql	trigger
query1	SELECT * FROM sensors WHERE id = @id	1000

- **disable**——设为“true”可跳过此条目，设为“false”则启用。
- **name**——此查询的标签名。系统中的完整标签名遵循“moduleId.taskId.name”的命名约定。
- **sql**——要执行的 SQL 语句，或包含该语句的.sql 文件路径。如果给出相对路径，相对于配置文件目录解析。绝对路径（如以 C:\开头）原样使用。
- **trigger**——可选的每行触发器间隔（毫秒），覆盖任务级触发器。留空时使用任务级触发器。
- **description**——可选的描述，供你参考。

sql

你可以在 csv 列中直接以字符串输入 SQL 语句，或指向外部.sql 文件。对于较长或更复杂的查询，使用外部文件很方便。SQL 语句可包含以@为前缀的命名参数（如@id、@startDate）。这些参数在运行时用写入标签的值替换（见下文参数化查询）。

每行触发器

CSV 中的每行可有自己的触发器间隔。当不同查询需要以不同频率运行时很有用。例如，摘要查询可能每 60 秒运行一次，而状态检查每秒运行。如果 trigger 列留空，该行继承任务级触发器。

参数化查询

你可以通过向带?后缀的标签名写入值来向 SQL 查询传递参数。参数值使用查询字符串格式：单个参数 key=value，或多个参数 key1=value1&key2=value2。

例如，如果你的 CSV 条目名称为“query1”，SQL 为：

```
SELECT * FROM sensors WHERE id = @id AND status = @status
```

你可以将值 `id=5&status=active` 写入标签 `module01.presql01.query1?`。模块会在执行查询前将`@id` 替换为 `5`、`@status` 替换为 `active`。

提供参数时，结果标签名将包含参数字符串（如 `module01.presql01.query1?id=5&status=active`），使不同参数组合产生不同结果标签。

未写入参数时，模块原样执行 SQL 语句，结果发布到不带`?`后缀的标签（如 `module01.presql01.query1`）。

查询结果作为对象的 JSON 数组返回，每个对象代表结果集中的一行。

示例

下面是一个完整示例，查询两个不同数据库。

`dxpresql.json`:

```
{
  "servers": [{"url": "127.0.0.1:1883"}],
  "project": "PLANT1",
  "id": "sql",
  "meta": {
    "name": "dxpresql"
  },
  "update": ["1000"],
  "task": [
    {
      "enable": true,
      "id": "production",
      "url": "Driver={MySQL ODBC 8.0 Unicode
Driver};Server=192.168.1.100;Database=production;Uid=reader;Pwd=pass;",
      "trigger": ["5000"],
      "data": [
        { "$csv": "dxpresql-production.csv" }
      ]
    },
    {
      "enable": true,
      "id": "inventory",
      "url": "Driver={SQL
Server};Server=192.168.1.101;Database=inventory;Uid=sa;Pwd=pass;",
      "trigger": ["60000"],
      "data": [
        { "$csv": "dxpresql-inventory.csv" }
      ]
    }
  ]
}
```

```
}
```

dxpresql-production.csv:

name	sql	trigger	description
line_status	SELECT line_id, status, output_count FROM production_lines	5000	Production line status
shift_summary	SELECT shift, total_output, reject_count FROM shift_summary WHERE shift_date = CURDATE()	60000	Current shift summary

dxpresql-inventory.csv:

name	sql	trigger	description
stock_level	SELECT part_no, quantity FROM stock WHERE quantity < @threshold	60000	Low stock alert
recent_receipts	SELECT * FROM goods_received WHERE received_date >= DATEADD(day, -1, GETDATE())	60000	Last 24h receipts

采用上述配置，模块将：

73. 每 5 秒查询生产数据库获取生产线状态。标签名为“sql.production.line_status”，其值为查询结果的 JSON 数组。

74. 每 60 秒查询班次摘要。标签名为“sql.production.shift_summary”。

75. 每 60 秒查询库存数据库获取库存水平。你可以通过向标签“sql.inventory.stock_level?”写入如 threshold=100 的值来传递阈值参数。

76. 每 60 秒查询最近收货。标签名为“sql.inventory.recent_receipts”。

历史模拟器

DX Historian Simulator (dxhissim) 通过回放先前记录的日志文件为 DIOT 项目执行数据记录。它从指定文件夹中存储的.log 文件读取日志数据，然后以调整后的时间戳发布记录的标签值，使数据看起来像实时生成。这对测试、演示和验证历史及记录 workflow 很有用，无需实时数据源。

关于 servers、project、模块 ID、trigger、update、data、存储转发和运行时等通用 DX 配置，请参阅 DX 驱动通用配置指南。

配置

下面是一个典型配置文件 dxhissim.json 的示例。

```
{
  "servers": [{"url": "127.0.0.1:1883"}],
  "project": "DEM0",
  "id": "his",
  "meta": {
    "name": "dxhissim"
  },
  "update": ["1000"],
  "task": [
    {
      "enable": true,
      "id": "log",
      "url": "",
      "trigger": 60000,
      "data": [
        { "$csv": "dxhissim.csv" }
      ]
    }
  ]
}
```

url

包含要回放的.log 文件的文件夹路径。可以是绝对路径或相对路径。提供相对路径时，相对于 JSON 配置文件所在文件夹解析。留空（默认）时，模块使用配置文件文件夹内名为 data 的子文件夹。例如，如果配置文件在 C:\project\dxhissim.json 且 url 为空，模块会在 C:\project\data\ 中查找日志文件。

模块还检查指定路径内的子文件夹。每个子文件夹名被视为角色名，该子文件夹中的日志文件在对应角色下发布。直接放在 `url` 路径根目录的日志文件不带角色发布。

trigger

控制模块扫描新日志文件频率的间隔（毫秒）。默认值为 60000 ms（一分钟）。文件夹中所有日志文件完全回放后，模块会在下个触发周期重新扫描新文件。

update

控制记录数据回放和发布速度的更新间隔（毫秒）。每个更新周期将回放时钟推进更新间隔量，并发布时间戳落在该窗口内的所有日志条目。较短更新间隔产生更细粒度回放，较长间隔使数据以更大批次发布。默认值为 1000 ms。

日志文件格式

模块从配置的 `url` 文件夹读取 `.log` 文件。每个日志文件必须是 JSON 文件，包含一个字典，其中每个键为标签名、每个值为数据点数组。每个数据点是两个元素的数组：时间戳（毫秒数值）和标签值。

日志文件结构示例：

```
{
  "tag1": [
    [1700000000000, 25.5],
    [1700000001000, 26.0],
    [1700000002000, 25.8]
  ],
  "tag2": [
    [1700000000000, true],
    [1700000001000, false]
  ]
}
```

模块按发现顺序一次处理一个日志文件。每个文件内，数据点按时间戳排序并顺序回放。原始时间戳转换为从文件开始的相对偏移，然后应用到当前系统时间，使回放数据带有新时间戳。

`url` 文件夹中名为 `datacount.log` 的文件自动排除处理。

CSV

CSV 文件定义要记录的标签列表。默认的 CSV 列如下：

name	description
temperature	Process temperature
pressure	Vessel pressure

- **disable**——设为“true”可跳过此条目，设为“false”则启用。
- **name**——标签名。系统中的完整标签名遵循“moduleId.taskId.name”的命名约定。
- **description**——可选的描述，供你参考。

示例

下面是一个完整示例，从自定义文件夹回放日志数据。

dxhissim.json:

```
{
  "servers": [{"url": "127.0.0.1:1883"}],
  "project": "PLANT1",
  "id": "his",
  "meta": {
    "name": "dxhissim"
  },
  "update": ["2000"],
  "task": [
    {
      "enable": true,
      "id": "sim1",
      "url": "C:\\logs\\simulation",
      "trigger": 30000,
      "data": [
        { "$csv": "dxhissim.csv" }
      ]
    }
  ]
}
```

dxhissim.csv:

name	description
temperature	Simulated temperature reading
pressure	Simulated pressure reading
level	Simulated tank level

采用上述配置，模块将：

77. 每 30 秒在 C:\logs\simulation 中查找.log 文件。
78. 以 2 秒更新间隔回放每个日志文件的数据，以当前时间戳发布标签值。
79. 标签名为“his.sim1.temperature”、“his.sim1.pressure”和“his.sim1.level”。
80. 日志文件中所有数据回放后，模块移到下一文件。所有文件处理完后，等待下一个触发周期重新扫描文件夹寻找新文件。

第五部分 设备驱动

Modbus 主站

DX Modbus Master (dxmodmast) 对 Modbus 设备执行读写操作。它支持 Modbus TCP、基于 TCP 的 Modbus RTU、Modbus UDP 和基于串口的 Modbus RTU。

关于 servers、project、模块 ID、trigger、update、data、存储转发和运行时等通用 DX 配置，请参阅 DX 驱动通用配置指南。

配置

下面是一个典型配置文件 dxmodmast.json 的示例。

```
{
  "servers": [{"url": "127.0.0.1:1883"}],
  "project": "DEM0",
  "id": "mbus",
  "meta": {
    "name": "dxmodmast"
  },
  "task": [
    {
      "enable": true,
      "id": "stn1",
      "url": "192.168.1.10",
      "trigger": 1000,
      "stationId": 1,
      "timeout": 2000,
      "offset": 0,
      "dataFormat": "ABCD",
      "readOnly": false,
      "data": [
        { "$csv": "dxmodmast.csv" }
      ]
    }
  ]
}
```

连接类型

驱动根据 url 值自动确定连接类型。

URL Format	Connection Type
IP address or hostname (e.g. “192.168.1.10”)	Modbus TCP (default port 502)
IP address with port (e.g. “192.168.1.10:502”)	Modbus TCP on specified port
Serial port on Windows (e.g. “COM1”)	Modbus RTU over serial
Serial port on Linux (e.g. “/dev/ttyS0”, “/dev/ttyUSB0”, “/dev/ttyAMA0”)	Modbus RTU over serial

要使用基于 TCP 的 Modbus RTU，在任务配置中与 TCP URL 一起设置“rtu”: true。

要使用 Modbus UDP 而非 TCP，在任务配置中设置“udp”: true。

任务参数

stationId

网络上目标设备的 Modbus 站或单元 ID。默认 1。

timeout

连接和接收超时（毫秒）。如果不指定超时，运行时默认 1000 ms。对于 TCP 和 UDP 连接，允许最小值为 500 ms。如果设置低于 500 的值，会自动提高到 500。

offset

调整数据地址，以补偿主站和从站之间的索引错位。可能值为 -1、0 或 1。默认 0。要验证偏移是否正确，在不是最开始的地址轮询一个已知值（避免 0 或 1 这样的地址以排除通信错误）。

dataFormat

控制 32 位数据类型（real32、int32、uint32）的字节序。可用选项如下：

Value	Byte Order
ABCD	Big-endian (default)
BADC	Big-endian byte swap
CDAB	Little-endian byte swap
DCBA	Little-endian

如果 32 位值看起来乱码或不正确，尝试不同字节序以匹配设备格式。

readOnly

设为 true 时，驱动只从设备读取，所有写操作禁用。默认 false。

shortConnection

设为 **true** 时，驱动使用非持久连接，即每个轮询周期连接和断开，而非保持持久连接打开。默认 **false**。当设备或网关连接容量有限时很有用。

rtu

设为 **true** 可使用基于 TCP 的 Modbus RTU 协议而非标准 Modbus TCP。默认 **false**。当设备期望在 TCP 连接上使用 RTU 帧时使用。

udp

设为 **true** 可使用 Modbus UDP 而非 Modbus TCP。默认 **false**。

traceLog

设为 **true** 可将此任务每个原始发送/接收 Modbus 帧记录到状态日志。默认 **false**。
正常运行时关闭——每个轮询周期都记录帧，会很快填满状态日志。仅在需要诊断该设备流量时为单个任务开启。

串口参数

使用串口 URL 时，以下额外参数配置串口通信。

baudRate

串口波特率。默认 9600。

dataBits

每字节的数据位数。默认 8。

stopBits

每字节的停止位数。接受值为 1、1.5 或 2。默认 1。

parity

奇偶校验协议。接受值如下：

- **none**（或“n”）——无校验。这是默认。
- **odd**（或“o”）——奇校验。
- **even**（或“e”）——偶校验。
- **space**（或“s”）——空格校验。
- **mark**（或“m”）——标记校验。

CSV

CSV 文件定义从 Modbus 设备读取或写入的数据点列表。列如下：

name	description	address	type	length	default	scale	addressEx
HR01	Holding register 1	1	int16	-1	1	1	4

- **disable**——设为“true”可跳过此条目，设为“false”则启用。
- **name**——此数据点的标签名。
- **description**——此数据点的描述。
- **address**——Modbus 寄存器或线圈地址号。
- **type**——数据类型。支持类型包括 boolean、int8、uint8、int16、uint16、int32、uint32、real32、int64、uint64、real64 和 string。
- **length**——数据长度。默认长度使用 -1。对于字符串类型，输入字符数。
- **default**——标签的默认值。
- **scale**——应用于原始值的缩放因子。
- **addressEx**——Modbus 地址类型。这决定读写操作使用哪个功能码。

addressEx

addressEx 列指定 Modbus 地址的类别：

Value	Address Type	Access
0	Output Coil (0x)	Read/Write Boolean
1	Discrete Input (1x)	Read-only Boolean
3	Input Register (3x)	Read-only Integer
4	Holding Register (4x)	Read/Write Integer

如果未指定 addressEx，默认为输出线圈（0x）。

示例

下面是一个完整示例，两个任务轮询同一设备上的不同站：

```
{
  "servers": [{"url": "127.0.0.1:1883"}],
  "project": "DEM0",
  "id": "gateway1",
  "meta": {
    "name": "dxmodmast"
  },
  "task": [
    {
```

```

        "enable": true,
        "id": "stn1",
        "url": "192.168.1.10",
        "trigger": 1000,
        "stationId": 1,
        "readOnly": false,
        "data": [
            { "$csv": "dxmodmast.csv" }
        ]
    },
    {
        "enable": true,
        "id": "stn2",
        "url": "192.168.1.10",
        "trigger": 2000,
        "stationId": 2,
        "readOnly": true,
        "data": [
            { "$csv": "dxmodmast.csv" }
        ]
    }
]
}

```

以及关联的 CSV，注意同一 CSV 文件在两个不同任务段中链接：

name	description	address	type	length	default	scale	addressEx
AO001	Analog output 1	1	int16	-1	1	10	4
AO002	Analog output 2	2	int16	-1	1	10	4
AO003	Analog output 3	3	uint16	-1	1	1	4
DI001	Digital input 1	1	boolean	-1		1	1

此示例中，任务 stn1 和 stn2 的第一个标签分别为“gateway1.stn1.AO001”和“gateway1.stn2.AO001”。

下面是串口 RTU 配置示例：

```

{
    "servers": [{"url": "127.0.0.1:1883"}],
    "project": "DEM0",
    "id": "serial1",
    "meta": {
        "name": "dxmodmast"
    },
    "task": [
        {
            "enable": true,

```

```
    "id": "dev1",
    "url": "COM3",
    "trigger": 1000,
    "stationId": 1,
    "baudRate": 9600,
    "dataBits": 8,
    "stopBits": 1,
    "parity": "none",
    "data": [
        { "$csv": "dxmodmast.csv" }
    ]
}
```

Modbus 从站

DX Modbus Slave（dxmodslav）充当 Modbus 从站（服务器）设备，允许外部 Modbus 主站读写标签数据。它支持 Modbus TCP 和基于串口的 Modbus RTU。从 SCADA 服务器收到的标签值在读请求时提供给 Modbus 主站，Modbus 主站写入的值转发回 SCADA 服务器。

关于 servers、project、模块 ID、trigger、update、data、存储转发和运行时等通用 DX 配置，请参阅 DX 驱动通用配置指南。

配置

下面是一个典型配置文件 dxmodslav.json 的示例。

```
{
  "servers": [{"url": "127.0.0.1:1883"}],
  "project": "DEM0",
  "id": "module01",
  "meta": {
    "name": "dxmodslav"
  },
  "task": [
    {
      "enable": true,
      "id": "slav01",
      "url": 502,
      "stationID": 1,
      "offset": 0,
      "dataSort": "ABCD",
      "data": [
        { "$csv": "dxmodslav.csv" }
      ]
    }
  ]
}
```

任务参数

url

Modbus 从站的监听地址。如果提供数字（如 502），模块在该端口启动 Modbus TCP 服务器。如果提供串口路径（如 Windows 上的“COM3”或 Linux 上的“/dev/ttyS0”），模块在该串口启动 Modbus RTU 从站。默认 502。

stationID

此从站响应的 Modbus 站或单元 ID。默认 1。在同一端口监听时，每个任务必须使用唯一站 ID，使主站请求路由到正确的标签集。

offset

调整数据地址，以补偿主站和从站之间的索引错位。可能值为 -1、0 或 1。默认 0。

dataSort

控制 32 位数据类型（real32、int32、uint32）的字节序。可用选项如下：

Value	Byte Order
ABCD	Big-endian (default)
BADC	Big-endian byte swap
CDAB	Little-endian byte swap
DCBA	Little-endian

如果 32 位值看起来乱码或不正确，尝试不同字节序以匹配主站期望格式。

CSV

CSV 文件定义 Modbus 从站暴露的数据点列表。列如下：

name	address	stationID	type	length	default	scale	addressEx
HR01	1	1	int16	-1	1	1	4

- **disable**——设为“true”可跳过此条目，设为“false”则启用。
- **name**——此数据点的标签名。
- **description**——此数据点的描述。
- **address**——Modbus 寄存器或线圈地址号。
- **stationID**——此数据点所属的站 ID。设为 0 时使用任务级 stationID。这使你能通过在数据级分配不同站 ID，从单个任务服务多个站 ID。
- **type**——数据类型。支持类型包括 boolean、int16、uint16、int32、uint32、real32、real64 和 string。
- **length**——数据长度。默认长度使用 -1。对于字符串类型，输入字符数。
- **default**——标签的默认值。
- **scale**——应用于原始值的缩放因子。
- **addressEx**——决定此数据点如何暴露的 Modbus 地址类型。

addressEx

addressEx 列指定 Modbus 地址的类别：

Value	Address Type	Behavior
0	Output Coil (0x)	Read/Write Boolean
1	Discrete Input (1x)	Read-only Boolean
3	Input Register (3x)	Read-only Integer
4	Holding Register (4x)	Read/Write Integer

Modbus 主站读取从站时，模块以从 SCADA 服务器收到的最新标签值响应。
Modbus 主站写入从站时（使用功能码 5、6、15 或 16），写入值转发回 SCADA 服务器。

示例

下面是一个完整示例，Modbus TCP 从站在端口 502 监听：

```
{
  "servers": [{ "url": "127.0.0.1:1883" }],
  "project": "DEM0",
  "id": "gateway1",
  "meta": {
    "name": "dxmodslav"
  },
  "task": [
    {
      "enable": true,
      "id": "slav01",
      "url": 502,
      "stationID": 1,
      "offset": 0,
      "dataSort": "ABCD",
      "data": [
        { "$csv": "dxmodslav.csv" }
      ]
    }
  ]
}
```

以及关联的 CSV 文件 dxmodslav.csv：

name	description	address	stationID	type	length	default	scale	addressEx
HR01	Holding register 1	1	1	int16	-1	0	1	4
HR02	Holding register 2	2	1	uint16	-1	0	1	4
HR03	Holding register 3	3	1	real32	-1	0	1	4
DI01	Digital	1	1	boolean	-1		1	1

	input 1							
DO01	Digital output 1	1	1	boolean	-1		1	0

此示例中，标签名为“gateway1.slav01.HR01”、“gateway1.slav01.HR02”等。

下面是串口 RTU 从站示例：

```
{
  "servers": [{ "url": "127.0.0.1:1883" }],
  "project": "DEM0",
  "id": "serial1",
  "meta": {
    "name": "dxmodslav"
  },
  "task": [
    {
      "enable": true,
      "id": "slav01",
      "url": "COM3",
      "stationID": 1,
      "dataSort": "ABCD",
      "data": [
        { "$csv": "dxmodslav.csv" }
      ]
    }
  ]
}
```

OPC UA 客户端

DX OPC UA Client (dxopcucli) 对 OPC UA 服务器执行读写操作。它支持轮询和订阅两种模式；匿名、用户名/密码或基于证书的用户登录；独立的消息安全；以及可选的应用证书。模块首次连接时自动浏览 OPC UA 服务器，并生成列出所有可用标签的 CSV 文件。

关于 servers、project、模块 ID、trigger、update、data、存储转发和运行时等通用 DX 配置，请参阅 DX 驱动通用配置指南。

配置

下面是一个典型配置文件 dxopcucli.json 的示例。

```
{
  "servers": [{ "url": "127.0.0.1:1883" }],
  "project": "DEM0",
  "id": "opc1",
  "meta": {
    "name": "dxopcucli"
  },
  "task": [
    {
      "enable": true,
      "id": "ua1",
      "url": "opc.tcp://localhost:49320",
      "username": "",
      "password": "",
      "securityMode": "none",
      "appCert": "",
      "userCert": "",
      "subscription": true,
      "trigger": [1000],
      "data": [
        { "$csv": "dxopcucli.csv" }
      ]
    }
  ]
}
```

url

OPC UA 服务器端点地址。必须是有效的 OPC UA 端点 URL，通常格式为 opc.tcp://hostname:port。默认值为 opc.tcp://localhost:49320。

username

用于向 OPC UA 服务器认证的用户名。如果服务器接受匿名连接，留空。同时提供用户名和密码时，客户端使用用户名/密码认证。

password

用于向 OPC UA 服务器认证的密码。如果服务器接受匿名连接，留空。

securityMode

应用于通道的消息安全，与你如何认证无关——你可以保护通道，同时仍以匿名或用户名登录。

- **none**（默认）——无消息安全；通道既不签名也不加密。
- **sign**——消息被签名（完整性和真实性），不加密。
- **signandencrypt**——消息被签名并加密——最强的逐条消息保护。
- **best**——自动选择服务器提供的最强安全。

sign 和 signandencrypt 需要服务器提供匹配端点；如果没有提供，客户端回退到最佳可用并记录警告。

appCert

在安全握手期间标识此客户端的应用实例证书。

- **empty**（默认）——客户端首次运行时自动生成自签名应用证书并复用。
- **a file path**——客户端加载指定的.pfx 作为应用证书。在编辑器中，使用 **Browse** 导入.pfx；它存储在模块自己的文件夹中，路径会为你填好。

userCert

用于作为用户登录的 X.509 证书，代替用户名/密码。

- **empty**（默认）——不使用基于证书的用户登录。
- **a file path**——客户端出示此.pfx 作为用户身份。在编辑器中，使用 **Browse** 导入。

用户身份是单一选择，与 securityMode 和 appCert 分开：

- 用户名、密码和 userCert 都留空，用于 anonymous 访问。
- 设置用户名和密码，用于 username/password 登录。

- 设置 userCert，用于 certificate 登录。

由于三个关注点独立，任何有效组合都可行——例如，使用选定应用证书的签名并加密通道，同时以用户名和密码登录。

Backward compatibility.使用 secureConnection 的旧配置继续工作——它仍被读取并自动映射（true 映射为最佳可用；文件路径映射为同时用作应用和用户证书的文件）。新配置使用 securityMode、appCert 和 userCert。

subscription

设为 true 时，客户端创建 OPC UA 订阅并监视标签数据变化。受监视标签值变化时，服务器将更新推送到客户端，对实时数据很高效。这是默认和推荐模式。

设为 false 时，客户端通过直接轮询服务器，在每个触发周期读取所有配置标签。当 OPC UA 服务器不支持订阅或需要数据对齐特定计划时使用轮询模式。

readOnly

设为 false（默认）时，模块允许从 OPC UA 服务器读取和写入。设为 true 时，模块只读取数据并忽略任何写请求。

自动浏览

首次连接时，模块自动浏览 OPC UA 服务器的地址空间，并在配置文件同文件夹中生成名为<configuration file name>.<task id>.all.csv（例如 dxopcuacli.0.all.csv）的 CSV 文件。此文件包含服务器上发现的所有可用标签，你可以将其作为构建自己 CSV 配置的起点。在 DX Edit 中，打开任务的 CSV 并选择 Browse addresses 以挑选和添加标签。浏览只运行一次；如果文件已存在，不重新生成。

早期版本中名为 moduleId.taskId.all.csv 的文件不再使用；你可以删除它。

CSV

CSV 文件定义从服务器读取（以及可选写入）的 OPC UA 标签列表。CSV 列如下：

name	address	type	length	scale
AO01	ns=2;s=Demo.Dynamic.Scalar.Int32	int32	-1	1

- **disable**——设为“true”可跳过此条目，设为“false”则启用。
- **name**——标签名。系统中的完整标签名遵循“moduleId.taskId.name”的命名约定。
- **description**——可选的描述，供你参考。
- **address**——要读取变量的 OPC UA NodeId。使用标准 OPC UA NodeId 语法，例如命名空间 2 中字符串标识符的 ns=2;s=MyVariable，或数值标识符的 ns=2;i=1234。
- **type**——标签的数据类型。支持类型：boolean、int8、uint8、int16、uint16、int32、uint32、real32、int64、uint64、real64 和 string。留空时，类型从 OPC UA 服务器自动确定。
- **length**——数据长度。设为 -1 用于默认行为。
- **default**——需要初始化时标签的默认值。
- **scale**——应用于从 OPC UA 服务器读取原始值的缩放倍数。默认 1（不缩放）。例如，将 scale 设为 0.1 会将原始值除以 10。

示例

下面是一个完整示例，以不同配置连接两个 OPC UA 服务器。

dxopcuacli.json:

```
{
  "servers": [{"url": "127.0.0.1:1883"}],
  "project": "PLANT1",
  "id": "opc1",
  "meta": {
    "name": "dxopcuacli"
  },
  "task": [
    {
      "enable": true,
      "id": "plc",
      "url": "opc.tcp://192.168.1.10:4840",
      "username": "",
      "password": "",
      "securityMode": "none",
      "subscription": true,
      "trigger": [1000],
      "data": [
        { "$csv": "plc-tags.csv" }
      ]
    }
  ],
}
```

```

    {
        "enable": true,
        "id": "scada",
        "url": "opc.tcp://192.168.1.20:49320",
        "username": "admin",
        "password": "pass123",
        "securityMode": "best",
        "subscription": false,
        "readOnly": true,
        "trigger": [5000],
        "data": [
            { "$csv": "scada-tags.csv" }
        ]
    }
]
}

```

plc-tags.csv:

name	description	address	type	length	scale
temp1	Tank Temperature	ns=2;s=PLC.Tank1.Temperature	real32	-1	1
level1	Tank Level	ns=2;s=PLC.Tank1.Level	real32	-1	0.1
valve1	Inlet Valve	ns=2;s=PLC.Tank1.InletValve	boolean	-1	1

scada-tags.csv:

name	description	address	type	length	scale
total_prod	Total Production	ns=3;i=1001	uint32	-1	1
run_hours	Run Hours	ns=3;i=1002	real64	-1	1

采用上述配置，模块将：

81. 使用匿名访问无安全地连接到 192.168.1.10 的 PLC 服务器。它订阅三个标签，值变化时实时接收更新。1 秒触发间隔用作回退轮询周期。标签名为“opc1.plc.temp1”、“opc1.plc.level1”和“opc1.plc.valve1”。level 标签 scale 为 0.1，原始值发布前乘以 0.1。
82. 在安全通道（最佳可用）上以用户名/密码登录连接到 192.168.1.20 的 SCADA 服务器——用户认证和消息安全独立应用。它在只读模式下每 5 秒轮询两个标签。标签名为“opc1.scada.total_prod”和“opc1.scada.run_hours”。

OPC UA 服务器

DX OPC UA Server (dxopcuasvr) 将你的 SCADA 标签作为 OPC UA 服务器暴露，允许任何 OPC UA 客户端读写。当 OPC UA 客户端向节点写入值时，模块将该值转发回 SCADA 系统。同样，当 SCADA 系统中标签值变化时，模块更新对应 OPC UA 节点，使连接的客户端收到最新数据。

关于 servers、project、模块 ID、trigger、update、data、存储转发和运行时等通用 DX 配置，请参阅 DX 驱动通用配置指南。

配置

下面是一个典型配置文件 dxopcuasvr.json 的示例。

```
{
  "servers": [{ "url": "127.0.0.1:1883" }],
  "project": "DEM0",
  "id": "opcua1",
  "meta": {
    "name": "dxopcuasvr"
  },
  "task": [
    {
      "enable": true,
      "id": "svr1",
      "port": 48030,
      "username": "",
      "password": "",
      "data": [
        { "$csv": "dxopcuasvr.csv" }
      ]
    }
  ]
}
```

id

任务 ID 命名 OPC UA 地址空间中存放此任务节点的文件夹；上例中，节点位于名为 svr1 的文件夹下。留空时，文件夹以配置中任务的位置命名：第一个任务 0、第二个任务 1 等。该名称也在标签名中代替任务 ID。

port

OPC UA 服务器监听的 TCP 端口号。不设置（或 0）时，每个项目自动获得自己的端口（从项目文件夹派生，范围 48030–49029），使同一机器上的两个服务器项目

永不冲突。设置显式端口以固定——你输入的值始终被采用。每个任务启动自己的 OPC UA 服务器实例，因此一台机器运行多个任务时，给每个不同端口。OPC UA 客户端使用端点地址 `opc.tcp://hostname:port` 连接；服务器启动时记录实际绑定的端口，将客户端指向该端口。

username

OPC UA 客户端以用户名/密码认证连接时必须提供的用户名。留空（默认）时，服务器允许匿名连接，不要求凭据。如果同时设置用户名和密码，服务器拒绝匿名连接，只接受提供匹配凭据的客户端。

password

OPC UA 客户端必须随用户名提供的密码。此字段仅在同时设置 `username` 时相关。两者都为空时，允许匿名访问。

安全

服务器首次启动时自动生成自签名证书，存储在应用数据文件夹中。默认启用以下安全配置文件：

- Basic256Sha256
- Aes128Sha256RsaOaep
- Aes256Sha256RsaPss
- None (no encryption)

遇到不受信任的客户端证书时，服务器自动接受并持久保存。这简化了初始设置，但在生产环境中应审查受信任证书。

CSV

CSV 文件定义服务器要暴露的 OPC UA 节点列表。默认的 CSV 列如下：

name	address	type	length	default	scale
AO01	ns=2;s=Demo.Dynamic.Scalar.Int32	int32	-1	1	1

- **disable**——设为“true”可跳过此条目，设为“false”则启用。
- **name**——此节点的标签名。系统中的完整标签名遵循“`moduleId.taskId.name`”的命名约定。

- **description**——可选的描述，供你参考。
- **address**——此变量的 OPC UA 节点 ID。使用标准 OPC UA 节点 ID 格式，例如命名空间 2 中字符串标识符的 ns=2;s=MyTag，或数值标识符的 ns=2;i=1001。
- **type**——节点的数据类型。支持类型：boolean、int8、uint8、int16、uint16、int32、uint32、real32、int64、uint64、real64 和 string。如果未指定或无法识别，类型默认为 string。
- **length**——保留用于字符串长度。非字符串类型使用 -1。
- **default**——服务器启动时节点的初始值。节点保持此值，直到被 SCADA 系统更新或被 OPC UA 客户端写入。
- **scale**——应用于标签值的缩放因子。不缩放使用 1。

所有节点创建时都有读写访问。OPC UA 客户端写入值时，模块将其转换为配置数据类型并发布回 SCADA 系统。SCADA 系统更新标签值时，模块将其写入对应 OPC UA 节点，使订阅客户端收到通知。

示例

下面是一个完整示例，将一组过程变量作为 OPC UA 节点暴露。

dxopcuasvr.json:

```
{
  "servers": [{ "url": "127.0.0.1:1883" }],
  "project": "PLANT1",
  "id": "opcua1",
  "meta": {
    "name": "dxopcuasvr"
  },
  "task": [
    {
      "enable": true,
      "id": "svr1",
      "port": 48030,
      "username": "operator",
      "password": "secret123",
      "data": [
        { "$csv": "dxopcuasvr.csv" }
      ]
    }
  ]
}
```

dxopcuasvr.csv:

name	description	address	type	length	default	scale
Temperature	Reactor temperature	ns=2;s=Reactor.Temperature	real64	-1	0	1
Pressure	Reactor pressure	ns=2;s=Reactor.Pressure	real32	-1	0	1
ValveOpen	Inlet valve status	ns=2;s=Reactor.ValveOpen	boolean	-1	false	1
BatchID	Current batch identifier	ns=2;s=Reactor.BatchID	string	64		1

采用上述配置，模块将：

83. 在端口 48030 启动 OPC UA 服务器，使用用户名/密码认证。OPC UA 客户端使用端点 `opc.tcp://hostname:48030` 连接，必须以用户名“operator”和密码“secret123”认证。
84. 在 OPC UA 地址空间中名为“svr1”的文件夹下暴露四个节点。SCADA 系统中的标签名为“opcua1.svr1.Temperature”、“opcua1.svr1.Pressure”、“opcua1.svr1.ValveOpen”和“opcua1.svr1.BatchID”。
85. SCADA 系统中标签值变化时，对应 OPC UA 节点更新。OPC UA 客户端向任何节点写入值时，该值发布回 SCADA 系统。

OPC DA 客户端

DX OPC DA Client（dxopcdacli）对 OPC DA 服务器执行读写操作。OPC DA（数据访问）是基于 Microsoft COM/DCOM 技术的经典 OPC 标准，通常用于通过符合 OPC 的服务器与 PLC、DCS 系统和其他工业设备通信。

关于 servers、project、模块 ID、trigger、update、data、存储转发和运行时等通用 DX 配置，请参阅 DX 驱动通用配置指南。

配置

下面是一个典型配置文件 dxopcdacli.json 的示例。

```
{
  "servers": [{ "url": "127.0.0.1:1883" }],
  "project": "DEM0",
  "id": "module01",
  "update": ["1000"],
  "meta": {
    "name": "dxopcdacli"
  },
  "task": [
    {
      "enable": true,
      "id": "opcdacli01",
      "url": "",
      "trigger": ["1000"],
      "readOnly": false,
      "data": [
        { "$csv": "dxopcdacli.csv" }
      ]
    }
  ]
}
```

url

要连接的 OPC DA 服务器 URL。应为 URI 格式的 OPC DA 服务器端点。通常可从 OPC 服务器配置工具或厂商文档找到服务器 URL。例如，本地运行的 Kepware OPC 服务器可能使用类似 opcda://localhost/Kepware.KEPServerEX.V6 的 URL。

readOnly

设为 false（默认）时，模块允许从 OPC DA 服务器读取和写入。设为 true 时，写入禁用，模块只从服务器读取标签值。用于防止对关键设备的意外写入。

CSV

CSV 文件定义要读写的 OPC DA 项列表。列如下：

name	address	type	length	default	scale
Tag1	Simulation Examples.Functions.Ramp1	long	-1	1	1

- **disable**——设为“true”可跳过此条目，设为“false”则启用。
- **name**——此项的标签名。系统中的完整标签名遵循“moduleId.taskId.name”的命名约定。
- **description**——可选的描述，供你参考。
- **address**——OPC 服务器中定义的 OPC DA 项 ID。这是 OPC 服务器地址空间中项的完整路径，使用点号表示法（如“Simulation Examples.Functions.Ramp1”）。
- **type**——读写值的数据类型。支持类型包括：boolean（或 bool）、byte、char、short（或 int16）、int（或 int32）、long（或 int64）、ushort（或 uint16）、uint（或 uint32）、ulong（或 uint64）、float、double、string（或 str）和 DateTime。
- **length**——数据长度，主要用于字符串类型。值 -1 表示长度不限。
- **default**——标签的初始默认值。
- **scale**——应用于数值的倍数。读取时，OPC 服务器的原始值发布前乘以此因子。写入时，值发送到 OPC 服务器前除此因子。默认 1（不缩放）。写入时将 scale 设为 0 会写入值 0。当 OPC 服务器提供需要转换为工程单位的原始传感器数据时很有用。

示例

下面是一个完整示例，从本地 OPC DA 服务器读取数据。

dxopccli.json:

```
{
  "servers": [{"url": "127.0.0.1:1883"}],
  "project": "PLANT1",
  "id": "opc1",
  "update": ["1000"],
  "meta": {
    "name": "dxopccli"
```

```
    },
    "task": [
      {
        "enable": true,
        "id": "plc1",
        "url": "opcda://localhost/Kepware.KEPServerEX.V6",
        "trigger": ["1000"],
        "readOnly": false,
        "data": [
          { "$csv": "dxopcdacli.csv" }
        ]
      },
      {
        "enable": true,
        "id": "plc2",
        "url": "opcda://localhost/Kepware.KEPServerEX.V6",
        "trigger": ["5000"],
        "readOnly": true,
        "data": [
          { "$csv": "dxopcdacli-plc2.csv" }
        ]
      }
    ]
  }
}
```

dxopcdacli.csv:

name	description	address	type	length	default	scale
Temperature	Reactor Temperature	Channel1.PLC1.Temperature	double	-1	0	0.1
Pressure	Reactor Pressure	Channel1.PLC1.Pressure	float	-1	0	1
RunStatus	Motor Running	Channel1.PLC1.MotorRun	boolean	-1	false	1

dxopcdacli-plc2.csv:

name	description	address	type	length	default	scale
FlowRate	Inlet Flow	Channel2.PLC2.Flow	double	-1	0	1
Level	Tank Level	Channel2.PLC2.Level	int	-1	0	1

采用上述配置，模块将：

- 86. 每 1 秒从第一个任务读写三个标签。标签名为“opc1.plc1.Temperature”、“opc1.plc1.Pressure”和“opc1.plc1.RunStatus”。Temperature 标签应用 scale 0.1，服务器原始值 250 会发布为 25.0。
- 87. 每 5 秒在只读模式下从第二个任务读取两个标签。标签名为“opc1.plc2.FlowRate”和“opc1.plc2.Level”。

BACnet/IP

DX BACnet 驱动（dxbacnet）通过两种传输方式对 BACnet 设备执行读写操作：

- **BACnet/IP**——使用标准 BACnet/IP 协议通过 UDP 通信。这也能到达位于 BACnet/IP 到 MS/TP 路由器后面的 MS/TP 设备：路由器自动将流量中继到 MS/TP 设备，因此 url 保持为本机端点，而非路由器 IP。
- **BACnet MS/TP**——直接通过 RS-485 串口（COM）通信。在 URL 字段中输入 COM 端口名来选择。

传输方式由 URL 选择：COM 端口名选择原生串口 MS/TP；其他任何值，包括空值、端口号或 IP 地址，选择 BACnet/IP。驱动支持数值变化（COV）订阅和手动轮询两种模式，通过 Who-Is 服务自动发现设备，并按 BACnet 设备 ID 连接设备。

关于 servers、project、模块 ID、trigger、update、data、存储转发和运行时等通用 DX 配置，请参阅 DX 驱动通用配置指南。

配置

下面是一个典型配置文件 dxbacnet.json 的示例。

```
{
  "servers": [{"url": "127.0.0.1:1883"}],
  "project": "DEM0",
  "id": "module01",
  "meta": {
    "name": "dxbacnet"
  },
  "task": [
    {
      "enable": true,
      "id": "bacnet01",
      "url": "47808",
      "trigger": [1000],
      "deviceId": 0,
      "subscription": true,
      "retry": 3,
      "readOnly": false,
      "data": [
        { "$csv": "dxbacnet.csv" }
      ]
    }
  ]
}
```

url

此字段选择传输方式和连接目标。

对于 BACnet/IP，此字段是 **local endpoint**：本机要使用的网络接口和 BACnet UDP 端口。它不是目标设备的地址——驱动使用 Who-Is 广播按设备 ID 发现设备。

- 默认值为“47808”：标准 BACnet 端口 BAC0（47808）上的所有网络接口。这是推荐设置；空值或“0xBAC0”含义相同。
- 要使用其他 BACnet 端口，例如本机上的第二个 BACnet 网络，只给出端口：“47809”（BAC1）到“47823”（BACF）。十六进制形式“0xBAC1”也有效。此范围之外的端口（1 到 65535）按给定值使用。
- 仅当本机有多个网络接口且必须选择连接到 BACnet 网络的接口时才指定接口。必须使用该网络上的 **this machine's own IP**，例如“192.168.1.100:47808”。不带端口时使用 BAC0。

在 DX Edit 中，从本机地址的 **Local endpoint** 列表选择接口，或保留为 **All interfaces**，并从 BAC0 到 BACF 选择端口。

不要在此输入“127.0.0.1”这样的回环地址或设备自身 IP：这些无法到达局域网上的 BACnet 设备，因此发现不到任何设备。BACnet/IP 仅使用 IPv4。如果该值不是可用的本地接口地址（IPv6 地址、回环或本机没有的地址），驱动记录警告并回退为绑定所有接口。要到达 BACnet/IP 到 MS/TP 路由器后面的 MS/TP 设备，也将此保留为“47808”（或本地网卡 IP）；Who-Is 广播到达路由器，路由器将其中继到 MS/TP 设备。

对于 native MS/TP，输入串口名。Windows 上是 COM 端口，如“COM3”；Linux 上是设备路径，如“/dev/ttyS0”、“/dev/ttyUSB0”或“/dev/ttyAMA0”。检测到串口时，驱动使用 MS/TP 串口传输，下面的 MS/TP 设置适用。

deviceId

目标设备的 BACnet 设备实例号。驱动使用 Who-Is 服务发现网络上的设备，只与设备 ID 匹配此值的设备通信。必须将其设为 BACnet 设备上配置的正确设备实例号。默认值为 0。

为帮助你找到该编号，运行中的 BACnet/IP 任务会在配置文件旁边的一个名为 `<configuration file name>.<task id>.devices.all.json`（例如 `dxbacnet.bacnet01.devices.all.json`）的文件中，列出响应其 Who-Is 的每个设备。每个条目给出设备 ID、名称和地址。该文件在至少一个设备响应时写入，且仅对由字母、数字、`.`、`_`和`-`组成的任务 ID 写入。每个任务使用自己的 ID，因为两个相同 ID 的任务会共享该文件。在 DX Edit 中，Device ID 字段按名称显示这些设备；选择一个以填入其 ID，或选择 Find devices 以启动或重启模块并重新查找。MS/TP 任务不写此文件，因此对于 COM 端口 URL，该字段是普通数字框。

subscription

设为 `true` 时，驱动使用 BACnet COV（数值变化）订阅从设备接收数据更新。订阅对象的当前值变化时，设备会通知驱动，减少网络流量并对变化提供更快响应。

设为 `false` 时，驱动改用手动轮询。它在每个触发周期读取每个配置对象的当前值。如果设备不支持 COV 订阅，或需要以固定间隔读取值而不管它们是否变化，使用手动轮询。

默认值为 `true`。

retry

BACnet 通信请求的最大重试次数。如果请求失败，驱动在放弃前最多重试此次数。默认值为 3。在丢包常见的不可靠网络上增大此值。

readOnly

设为 `true` 时，驱动只从设备读取数据，不写回任何值。设为 `false` 时，读取和写入都启用。默认值为 `false`。

MS/TP（串口）

以下设置仅在 url 为串口 COM 端口时适用。对 BACnet/IP 忽略。MS/TP 串口帧由 BACnet 标准固定为 8 数据位、无校验、1 停止位，因此没有数据位、校验或停止位设置。

baudRate

MS/TP 网段的串口线路速度。同一 RS-485 网段上的所有设备必须使用相同波特率。支持值为 9600、19200、38400、57600、76800 和 115200。默认值为 38400，最常见的 MS/TP 速率。

master

此驱动作为网段上主站节点的 MS/TP MAC 地址。有效值为 0 到 127。它必须在网段上唯一（其他设备不能使用相同地址），且不能大于 **max**。默认值为 127。

max

MS/TP 网段上存在的最高主站 MAC 地址。驱动用它管理令牌传递；将其设为接近实际最高地址可提高性能。默认值为 127，BACnet 最大值。

batch

此主站在必须将令牌传递给下一主站前可发送的消息数。默认值为 1，BACnet 标准默认值。

关于 MS/TP 和 COV 的注意：许多 MS/TP 设备不支持 COV 订阅，因此对于串口连接，驱动无论订阅设置如何都自动使用手动轮询，并且因为令牌传递网络比 BACnet/IP 响应慢，每个请求允许更多时间。

关于大值的注意：对于一个 BACnet 消息来说太大的值，如长八位字符串，在设备支持分段时自动分段读取。八位字符串以 base64 文本发布，且仅在变化时发布。

自动浏览

deviceId 中的设备响应后，模块在配置文件同文件夹中名为 <configuration file name>.<task id>.<deviceId>.all.csv（例如 dxbacnet.bacnet01.400001.all.csv）的文件中列出设备的对象。每行可直接复制到你的 CSV：设备报告的对象名称和描述、对象类型和实例，以及一个类型。每个列出的对象都有类型，因此你添加的每一行都会被加载：位字符串、八位字符串、日期和时间等类文本值列为 string。只列出有值可读的对象——不包括设备对象本身、文件、组、趋势日志或结构化视图。

在 DX Edit 中，打开任务的 CSV 并选择 Browse addresses 以挑选行并添加。

该文件每个设备写一次；如果已存在，不再写入。更改任务的 **deviceId** 时，新设备在下次启动时列入自己的文件。要重新列出对象，例如设备变化后，删除该文件并重启模块，或在 DX Edit 中选择 **Regenerate discovery**。

CSV

CSV 文件定义要读取或写入的 BACnet 对象列表。CSV 列如下：

name	description	objectType	address	type
temp1	Zone Temperature	0	1	real32
fan1	Fan Status	4	1	boolean
setpoint1	Zone Setpoint	1	1	real32

- **disable**——设为“true”可跳过此条目，设为“false”则启用。
- **name**——此对象的标签名。系统中的完整标签名遵循“moduleId.taskId.name”的命名约定。
- **description**——可选的描述，供你参考。
- **objectType**——BACnet 对象类型。在 DX Edit 中从列表选择，新行开始为 Analog Input。文件保存其编号：

Value	Object Type
0	Analog Input
1	Analog Output
2	Analog Value
3	Binary Input
4	Binary Output
5	Binary Value
6	Calendar
7	Command
12	Loop
13	Multi-state Input
14	Multi-state Output
16	Program
17	Schedule
19	Multi-state Value
21	Life Safety Point
22	Life Safety Zone
23	Accumulator
24	Pulse Converter
28	Load Control
30	Access Door
31	Timer
39	BitString Value
40	CharacterString Value

41	Date Pattern Value
42	Date Value
43	DateTime Pattern Value
44	DateTime Value
45	Integer Value
46	Large Analog Value
47	OctetString Value
48	Positive Integer Value
49	Time Pattern Value
50	Time Value
53	Channel
54	Lighting Output
55	Binary Lighting Output

关于这两种类型，请参见下文 Program 和 Accumulator 对象部分。

- **address**——对象实例号。这是在设备上按类型唯一标识对象的数值实例。例如，模拟输入实例 1 的 objectType 为 0、address 为 1。
- **type**——标签值的数据类型。支持类型包括 boolean、int8、uint8、int16、uint16、int32、uint32、real32、int64、uint64、real64 和 string。
- **length**——数据长度，适用于字符串类型。
- **default**——标签的默认值。
- **scale**——允许根据所需缩放处理输入和输出原始数据。
- **property、index、priority**——可选列，选择读取对象的哪个属性，并允许使用 BACnet 命令优先级；见下文命令优先级部分。它们可以留空或完全从 CSV 中省略，此时标签照常访问对象的当前值。

命令优先级

BACnet 可命令对象（模拟输出、二进制输出和许多值对象）通过优先级机制决定其当前值：值可在 16 个优先级之一被命令，其中 1 最高，对象遵循最高占用级别。没有级别被占用时，对象回退到其放弃默认值。property、index 和 priority CSV 列允许你读取和命令这些级别。

property

选择标签访问对象的哪个属性：

- 空或“presentValue”——对象的当前值。这是默认。
- “priorityArray”——对象的 16 槽优先级数组；见下文 index。

- “relinquishDefault”——没有优先级被命令时对象使用的值。标签可读写。
- 也可以输入任何数值 BACnet 属性 ID 以读取其他属性。

属性不是 presentValue 的标签在每个触发周期通过轮询更新，即使任务已启用订阅，因为 COV 通知只携带当前值。

Program 和 Accumulator 对象

这两种对象类型自动使用正确属性读取；只需设置对象类型并将 property 列留空：

name	description	objectType	address	type
pgmState	Boiler program state	16	28	uint32
pulseCount	Energy pulse count	23	86	uint32

- Program 对象（对象类型 16）没有当前值，因此驱动改读其 Program State。该值是枚举状态：0 空闲、1 加载、2 运行、3 等待、4 停止、5 卸载。
- Accumulator 对象（对象类型 23）通过其当前值读取，该值是无符号运行 count。

对于两者，使用 uint32 等整数类型而非 real32，如果计数需要转换为工程单位，使用 scale 列。

读取任何其他属性

要读取对象值以外的属性，在 property 列中输入其数值 BACnet 属性 ID。只有 presentValue、priorityArray 和 relinquishDefault 按名称识别；其他所有属性必须以数字输入。

index

对于 priorityArray 标签，选择标签代表哪个优先级槽（1 到 16）：

- 1 到 16——标签显示该槽的命令值，槽未被命令时空（null）。写入标签会在该优先级命令对象的当前值。
- 空——标签以 JSON 数组形式保存所有 16 个槽，如 [null,null,35.0,null,...]，在单个标签中显示整个数组。此标签只读。

priority

对于 presentValue 标签，写入时要命令的优先级（1 到 16）。为空时，写入不带优先级，设备应用 BACnet 默认级别 16。此列不影响读取。

放弃命令

向槽标签或 presentValue 标签写入 null 会释放优先级而不是写入值：对象然后遵循下一占用级别，或在没有剩余时使用放弃默认值。释放的级别对于槽标签是槽索引，对于 presentValue 标签是 priority 列值，或未设置优先级时为 BACnet 默认 16。写入文本“null”或空值作用相同；空字符串仅对字符串类型标签视为正常值。

示例

操作员在级别 8 命令、控制程序持有级别 15 的模拟输出：

name	description	objectType	address	type	property	index	priority
ao1	AO1 present value	1	1	real32			8
ao1man	AO1 manual command	1	1	real32	priorityArray	8	
ao1prog	AO1 program command	1	1	real32	priorityArray	15	
ao1arr	AO1 all 16 slots	1	1	string	priorityArray		
ao1rel	AO1 relinquish default	1	1	real32	relinquishDefault		

- 向 \${ao1man} 写入 35 会在级别 8 将输出命令为 35；\${ao1} 会遵循它，因为级别 8 高于级别 15。
- 向 \${ao1man} 写入 null 会释放级别 8；输出回退到级别 15 的值，在 \${ao1prog} 中可见。
- 每个级别都为空时，输出使用其放弃默认值，可通过 \${ao1rel} 读取和更改。
- \${ao1arr} 以 JSON 数组形式一次显示所有 16 个槽。
- 向 \${ao1} 写入也会在此命令级别 8，因为其 priority 列为 8；列为空时会命令级别 16。

在槽标签上设置 `type` 列（模拟对象为 `real32`，二进制对象为 `boolean`），以便即使槽一直为空，写入也能正确编码。优先级列仅适用于可命令对象；没有优先级数组的对象（如模拟输入）拒绝这些读取。还需注意，每个槽标签与其他标签一样计入授权 IO 数，因此只映射你需要的槽——整个数组标签用单个标签覆盖所有 16 个槽。

示例

下面是一个完整示例，从本地网络上设备 ID 1234 的 BACnet/IP 设备读取数据。

`dxbacnet.json`:

```
{
  "servers": [{"url": "127.0.0.1:1883"}],
  "project": "PLANT1",
  "id": "bac",
  "meta": {
    "name": "dxbacnet"
  },
  "task": [
    {
      "enable": true,
      "id": "ahu1",
      "url": "47808",
      "trigger": [5000],
      "deviceId": 1234,
      "subscription": true,
      "retry": 3,
      "readOnly": false,
      "data": [
        { "$csv": "dxbacnet-ahu1.csv" }
      ]
    },
    {
      "enable": true,
      "id": "meter1",
      "url": "47808",
      "trigger": [10000],
      "deviceId": 5678,
      "subscription": false,
      "retry": 3,
      "readOnly": true,
      "data": [
        { "$csv": "dxbacnet-meter1.csv" }
      ]
    }
  ]
}
```

dxbacnet-ahu1.csv:

name	description	objectType	address	type
zoneTemp	Zone Temperature	0	1	real32
returnTemp	Return Air Temperature	0	2	real32
fanStatus	Supply Fan Status	3	1	boolean
setpoint	Zone Setpoint	1	1	real32

dxbacnet-meter1.csv:

name	description	objectType	address	type
kwh	Total Energy	2	1	real64
kw	Power Demand	2	2	real32

采用上述配置，模块将：

88. 连接到本地网络上设备 ID 1234 的 BACnet 设备，在默认端口 47808（配置的 url）上绑定所有接口。它使用 COV 订阅接收四个 AHU 对象的更新。标签名为“bac.ahu1.zoneTemp”、“bac.ahu1.returnTemp”、“bac.ahu1.fanStatus”和“bac.ahu1.setpoint”。由于 readOnly 为 false，也可以写入 setpoint 标签。
89. 连接到设备 ID 5678 的 BACnet 设备，使用手动读取每 10 秒轮询两个仪表对象。标签名为“bac.meter1.kwh”和“bac.meter1.kw”。由于 readOnly 为 true，不会对此设备执行写操作。

MS/TP 示例

要读取原生串口 MS/TP 网段上的设备，将 url 设为 COM 端口并提供 MS/TP 设置：

```
{
  "enable": true,
  "id": "vav1",
  "url": "COM3",
  "baudRate": 38400,
  "master": 127,
  "max": 127,
  "batch": 1,
  "trigger": [5000],
  "deviceId": 1001,
  "readOnly": false,
  "data": [
    { "$csv": "dxbacnet-vav1.csv" }
  ]
}
```

```
]
}
```

要读取同一设备但它位于 BACnet/IP 到 MS/TP 路由器后面时，继续使用 BACnet/IP，将 url 设为“47808”（或路由器网络上本机的网卡 IP）——不是路由器 IP。不需要 MS/TP 设置，因为路由器处理串口网段。这依赖路由器转发 Who-Is 广播并返回设备的 I-Am，因此确保绑定的接口能到达路由器，并根据实际路由器验证可达性：

```
{
  "enable": true,
  "id": "vav1",
  "url": "47808",
  "trigger": [5000],
  "deviceId": 1001,
  "subscription": false,
  "readOnly": false,
  "data": [
    { "$csv": "dxbacnet-vav1.csv" }
  ]
}
```

使用 Yabe 模拟器测试

Yabe（Yet Another BACnet Explorer）是一个免费开源的 BACnet 浏览器，在设置 dxbacnet 时有两方面用途：

- 其安装包包含可直接运行的 BACnet simulators，因此无需任何真实 BACnet 硬件即可测试完整 dxbacnet 配置。
- 浏览器显示网络上可见的每个 device and object visible on the network，这是获取配置和 CSV 文件正确 deviceId、objectType 和 address 值的最快方式。

本节走查完整测试会话：启动模拟器、用 Yabe 发现它，并配置 dxbacnet 获取相同数据。

启动模拟器

Yabe 安装程序在其安装目录的 AddOn 文件夹中放置两个模拟器，通常为“C:\Program Files\Yabe\AddOn”：

- **DemoServer.exe**——控制台 BACnet/IP 服务器，带有固定演示对象集（设备 ID 400001），由同文件夹中的 DeviceStorage.Xml 文件支持。
- **Bacnet.Room.Simulator.exe**——交互式房间恒温器模拟。窗口显示室温、设定值（T Set）、室外温度（T Ext）和三个运行模式按钮（Comfort、Eco+、Vacancy）。其 BACnet 设备 ID 显示在窗口底部——本走查中为 2310192。可一次运行多个副本，每个实例通告自己的设备 ID。

运行其中一个或两个。DemoServer 启动时，其控制台确认它在 UDP 端口 0xBAC0（47808，BACnet 默认端口）上提供 BACnet/IP 服务，并广播 I-Am 通告。稍后出现的“ConfirmedServiceRequest”和“Service_Confirmed_Read_Property”行是来自 Yabe 或 dxbacnet 等客户端的读取请求——方便确认客户端确实在轮询服务器：

用 Yabe 发现设备

启动 Yabe 并添加 BACnet/IP 通道：单击绿色“Add device”按钮，保留默认“BACnet/IP over Udp”本地端点端口 0xBAC0，单击 Add。Yabe 广播 Who-Is 请求，每个以 I-Am 响应的设备出现在 Udp:47808 节点下——这与 dxbacnet 使用的发现机制完全相同，因此 Yabe 能发现的任何设备，dxbacnet 也能到达。

两个模拟器都运行时，发现两个设备：“DemoServer [400001]”和“RoomController.Simulator [2310192]”。方括号中的数字是 BACnet 设备实例号——这是要在任务的 deviceId 字段中输入的值。

单击设备，Yabe 将其对象列表读入 Address Space 窗格。Room 模拟器发布 13 个对象：设备对象本身加 12 个数据对象。选择对象会在属性网格中显示其属性（对象名称、描述、当前值等）。要识别每个对象代表什么，操作模拟器时观察其当前值：跟随显示室温的模拟输入是温度传感器，持有 21 的模拟值是显示为“T Set”的设定值，按 Comfort/Eco+/Vacancy 时在 1、2、3 之间切换的多状态值是运行模式。

注意，同一机器上多个 BACnet 应用可共享 UDP 端口 47808：Yabe、模拟器和 dxbacnet 都以共享（非独占）模式打开端口，因此测试期间它们可在一台 PC 上并排运行。

将发现的对象映射到 CSV

Address Space 窗格中的每个条目显示为 OBJECT_{TYPE}:{instance}，直接转换为 CSV 列：对象类型变为 objectType 编号，实例变为 address。对于上面的 Room 模拟器：

Address Space entry	objectType	address	type
OBJECT_ANALOG_INPUT:0 ... 2	0	0, 1, 2	real32
OBJECT_ANALOG_VALUE:0 ... 3	2	0, 1, 2, 3	real32
OBJECT_BINARY_VALUE:0 ... 1	5	0, 1	boolean
OBJECT_MULTI_STATE_VALUE:0 ... 1	19	0, 1	uint32
OBJECT_CHARACTERSTRING_VALUE:1	40	1	string

用 dxbacnet 获取模拟器

获取 Room 模拟器所有 12 个数据对象的完整配置：

dxbacnet.json:

```
{
  "servers": [{"url": "127.0.0.1:1883"}],
  "project": "DEMO",
  "id": "bac",
  "meta": {
    "name": "dxbacnet"
  },
  "task": [
    {
      "enable": true,
      "id": "room1",
      "url": "47808",
      "trigger": [2000],
      "deviceId": 2310192,
      "subscription": true,
      "retry": 3,
      "readOnly": false,
      "data": [
        { "$csv": "dxbacnet-room1.csv" }
      ]
    }
  ]
}
```

dxbacnet-room1.csv:

name	description	objectType	address	type	length
ai0	Analog Input 0	0	0	real32	
ai1	Analog Input 1	0	1	real32	
ai2	Analog Input 2	0	2	real32	

av0	Analog Value 0	2	0	real32	
av1	Analog Value 1	2	1	real32	
av2	Analog Value 2	2	2	real32	
av3	Analog Value 3	2	3	real32	
bv0	Binary Value 0	5	0	boolean	
bv1	Binary Value 1	5	1	boolean	
msv0	Multi-state Value 0	19	0	uint32	
msv1	Multi-state Value 1	19	1	uint32	
cs1	CharacterString Value 1	40	1	string	64

上面的通用标签名与 Address Space 条目一一对应；如前所述在 Yabe 中识别每个对象后，将它们重命名为“roomTemp”或“setpoint”等有意义的名称。

模块启动时，在绑定接口上广播 Who-Is，匹配来自设备 2310192 的 I-Am 响应，读取配置对象，并订阅 COV 通知。然后值出现在标签名 “bac.room1.ai0” 到 “bac.room1.cs1” 下。

验证数据流

模块运行时，可双向验证链路：

- **Device to SCADA:** 模拟室温持续漂移，因此对应模拟输入标签通过 COV 自行更新。按 Comfort/Eco+/Vacancy 按钮使多状态值标签在 1、2、3 之间变化。
- **SCADA to device:** 写入设定值标签（模拟值之一）会移动模拟器窗口中的“T Set”读数。注意，模拟输入对象是传感器读数，按设计拒绝写入；写入测试使用模拟值、二进制值或多状态值对象。
- 在 DemoServer 上，控制台为每个收到的请求打印“ConfirmedServiceRequest”行，在设备端确认流量。

测试命令优先级

Yabe 还实时显示优先级机制，使其成为命令优先级列的便捷检查。选择一个在 Yabe 属性网格中列出 Priority_Array 属性的对象（模拟输出和可命令值对象有；没有的对象不支持优先级），并为其映射槽标签：

name	description	objectType	address	type	property	index
av0man	AV0	2	0	real32	priorityArray	8

	manual command					
--	-------------------	--	--	--	--	--

向 `${av0man}` 写入值并在 Yabe 中刷新对象：Priority_Array 槽 8 从 Null 变为该值，当前值遵循它。向 `${av0man}` 写入 null：槽 8 返回 Null，当前值回退到下一占用级别，或在没有剩余时使用 Relinquish_Default 值。槽标签在下个触发周期在 SCADA 端显示相同转换。

如果没有发现设备，检查 Windows 防火墙是否允许 UDP 端口 47808（模块首次启动时 Windows 提示则选择 Allow），url 字段是否保留为“47808”而非回环或设备地址，以及 deviceId 是否与 Yabe 方括号中显示的 ID 完全匹配。

DNP3

DX DNP3 Master（dxdn3）使用 DNP3（分布式网络协议 3）协议读写数据。它充当 DNP3 主站，通过 TCP/IP 或串口连接与一个或多个 DNP3 外站设备通信。模块支持从连接的外站读取二进制输入、双位二进制输入、二进制输出状态、模拟输入和模拟输出状态。

关于 servers、project、模块 ID、trigger、update、data、存储转发和运行时等通用 DX 配置，请参阅 DX 驱动通用配置指南。

配置

下面是一个典型配置文件 dxdn3.json 的示例。

```
{
  "servers": [{ "url": "127.0.0.1:1883" }],
  "project": "DEM0",
  "id": "module01",
  "meta": {
    "name": "dxdn3"
  },
  "task": [
    {
      "enable": true,
      "id": "device01",
      "url": "127.0.0.1",
      "trigger": 1000,
      "readOnly": false,
      "data": [
        { "$csv": "dxdn3.csv" }
      ]
    }
  ]
}
```

url

要连接的 DNP3 外站地址。此参数同时支持 TCP/IP 和串口连接。对于 TCP，DX Edit 在单独框中显示主机和端口，新任务开始于“127.0.0.1:20000”。

对于 TCP 连接，指定 IP 地址或主机名及可选端口号。未指定时默认端口为 20000。示例：

Format	Example
IP address only (uses default port 20000)	192.168.1.10

IP address with port	192.168.1.10:20000
Host name with port	outstation1:20000

对于串口连接，指定串口名。当 URL 包含以下任一模式时，模块检测到串口连接：

Platform	Example
Windows	COM1, COM3
Linux	/dev/ttyS0, /dev/ttyUSB0, /dev/ttyAMA0

readOnly

设为 **false**（默认）时，模块允许从外站读取和写入。设为 **true** 时，写操作禁用，模块只从外站读取数据。

CSV

CSV 文件定义从外站读取的数据点列表。默认的 CSV 列如下：

name	address
tag01	1

- **disable**——设为 **true** 可跳过此条目，设为 **false** 则启用。
- **name**——此数据点的标签名。系统中的完整标签名遵循“moduleId.taskId.name”的命名约定。
- **description**——可选的描述，供你参考。
- **address**——带类型标识符前缀的 DNP3 对象地址。前缀指示 DNP3 数据类型，数字是外站上的点索引。支持的前缀如下：

Prefix	DNP3 Object Type
BI	Binary Input
DBBI	Double-Bit Binary Input
BO	Binary Output Status
AI	Analog Input
AO	Analog Output Status

例如，AI0 指索引 0 的模拟输入，BI3 指索引 3 的二进制输入，AO1 指索引 1 的模拟输出状态。

示例

下面是一个完整示例，通过 TCP 从 DNP3 外站读取数据。

dxdnp3.json:

```
{
  "servers": [{ "url": "127.0.0.1:1883" }],
  "project": "PLANT1",
  "id": "dnp3",
  "meta": {
    "name": "dxdnp3"
  },
  "task": [
    {
      "enable": true,
      "id": "rtu1",
      "url": "192.168.1.10:20000",
      "trigger": 1000,
      "readOnly": false,
      "data": [
        { "$csv": "dxdnp3.csv" }
      ]
    },
    {
      "enable": true,
      "id": "rtu2",
      "url": "COM3",
      "trigger": 2000,
      "readOnly": true,
      "data": [
        { "$csv": "dxdnp3-serial.csv" }
      ]
    }
  ]
}
```

dxdnp3.csv:

name	description	address
valve open	Valve Open Status	BI0
valve close	Valve Close Status	BI1
pump status	Pump Running	BO0
temperature	Tank Temperature	AI0
pressure	Tank Pressure	AI1
setpoint	Temperature Setpoint	AO0

dxdnp3-serial.csv:

name	description	address
level	Water Level	AI0
flow	Flow Rate	AI1

alarm	High Level Alarm	BI0
-------	------------------	-----

采用上述配置，模块将：

90. 连接到 192.168.1.10:20000 的 TCP 外站，启用读写，每秒轮询。标签名为
“dnp3.rtu1.valve_open”、“dnp3.rtu1.temperature”等。
91. 连接到 COM3 上的串口外站，以只读访问每 2 秒轮询。标签名为
“dnp3.rtu2.level”、“dnp3.rtu2.flow”和“dnp3.rtu2.alarm”。

IEC 60870-5-104

DX IEC 60870-5-104 Master (dxiec104) 从 IEC 60870-5-104 设备读写数据。它充当控制站（主站），通过 TCP/IP 连接到一个或多个被控站（从站/外站），通过总召唤和自发事件接收监视数据，并在启用写入时发送控制命令。

关于 servers、project、模块 ID、trigger、update、data、存储转发和运行时等通用 DX 配置，请参阅 DX 驱动通用配置指南。

配置

下面是一个典型配置文件 dxiec104.json 的示例。

```
{
  "servers": [{ "url": "127.0.0.1:1883" }],
  "project": "DEM0",
  "id": "module01",
  "meta": {
    "name": "dxiec104"
  },
  "task": [
    {
      "enable": true,
      "id": "device01",
      "url": "127.0.0.1",
      "trigger": 5000,
      "sizeOfCOT": 2,
      "originatorAddress": 0,
      "t0": 30,
      "t1": 15,
      "t2": 10,
      "t3": 20,
      "k": 12,
      "w": 8,
      "commonAddress": 1,
      "readOnly": false,
      "timeout": 3000,
      "data": [
        { "$csv": "dxiec104.csv" }
      ]
    }
  ]
}
```

url

IEC 60870-5-104 设备的 IP 地址或主机名。可通过追加冒号和端口包含端口号（如“192.168.1.10:2404”）。未指定端口时使用默认 IEC 104 端口 2404。在 DX Edit 中，主机和端口有单独框，新任务开始于“127.0.0.1:2404”。

trigger

轮询间隔（毫秒）。控制模块检查连接状态和处理数据的频率。默认值为 5000 ms。

readOnly

设为 true 时，模块只从设备读取数据，不发送任何控制命令。设为 false（默认）时，模块接受写请求并向设备发送对应控制命令。

timeout

连接超时（毫秒）。此值既用于初始 TCP 连接超时，也在连接丢失时作为重连间隔。默认值为 3000 ms。

commonAddress

CSV 文件中所有标签的 ASDU 默认公共地址（CA）。任何未在 CSV 中指定自己公共地址的标签使用此值。默认值为 1。如果希望要求每个标签显式指定自己的公共地址，设为 -1。

sizeOfCOT

传输原因字段的大小（八位字节数）。有效值为 1 或 2。默认值为 2。大多数设备使用标准 2 字节 COT。仅当设备明确要求 1 字节 COT 时才更改。

originatorAddress

传输原因字段第二字节中使用的始发地址。默认值为 0。此参数仅在 sizeOfCOT 设为 2 时相关。当多个主站连接到同一设备时，允许识别命令来源。

t0

建立 TCP 连接的超时（秒）。默认值为 30。如果在此时间内未建立连接，模块会重试。

t1

等待已发送 APDU 确认（ACK）的时间（秒）。默认值为 15。如果在此时间内未收到确认，连接被视为丢失。

t2

发送监督确认 APDU（S 帧）前等待的时间（秒）。默认值为 10。必须始终小于 t1。

t3

发送测试 APDU（TESTFR）前的空闲时间（秒）。默认值为 20。如果在此期间未交换数据，发送测试帧以验证连接仍然存活。

k

未确认已发送 APDU 的最大数量。默认值为 12。达到此限制时，发送方必须等待确认后才能发送更多 APDU。

w

接收方发送监督确认前未确认已接收 APDU 的最大数量。默认值为 8。必须始终小于或等于 k 的三分之二。

自动浏览

当任务没有浏览文件时，模块在配置文件同文件夹中名为<configuration file name>.<task id>.all.csv（例如 dxiec104.device01.all.csv）的文件中，列出其站召唤返回的点。任务的 commonAddress 也被召唤，因此空 CSV 也能工作。每行保存一个如 M_SP_NA.1 的现成地址及其 CA，并以其命名（M_SP_NA_1）；CA 不是任务的 commonAddress 时，将 CA 添加到名称。

在 DX Edit 中，打开任务的 CSV 并选择 Browse addresses 以挑选行并添加。

只列出监视点。命令点（C_...）和累计总量（M_IT_...）不会由站召唤返回，需要手动添加。

该文件写一次；如果已存在，不再写入。要重新列出点，删除该文件并重启模块，或在 DX Edit 中选择 Regenerate discovery。

CSV

CSV 文件定义从设备读取或写入设备的数据点列表。CSV 列如下：

disable,name,description,address,length,default,scale,ca

- **disable**——设为“true”可跳过此条目，设为“false”则启用。
- **name**——此数据点的标签名。系统中的完整标签名遵循“moduleId.taskId.name”的命名约定。
- **description**——可选的描述，供你参考。
- **address**——IEC 104 地址，格式为“TypeID.IOA”，其中 TypeID 是 IEC 60870-5 类型标识（如 M_SP_NA、M_ME_NC），IOA 是信息对象地址（如 M_SP_NA.100）。详见下文地址格式部分。
- **length**——字符串类型的数据长度。非字符串类型留空。
- **default**——标签的默认初始值。
- **scale**——应用于原始值的倍数。对于监视（读）地址，发布值为原始值乘以 scale。对于命令（写）地址，写入值为输入值除以 scale。留空或设为 1 表示不缩放。
- **ca**——此特定标签的公共地址。留空时使用任务级 JSON 配置中的 commonAddress 值。

地址格式

地址字段对监视（读）点使用“TypeID.IOA”格式，对命令（写）点使用“TypeID.IOA[.options]”格式。

监视地址（读）

Type ID	Description
M_SP_NA	Single-point information
M_SP_TB	Single-point information with CP56Time2a timestamp
M_DP_NA	Double-point information
M_DP_TB	Double-point information with CP56Time2a timestamp
M_ST_NA	Step position information (e.g. tap changer position)
M_ST_TB	Step position information with CP56Time2a timestamp
M_ME_NA	Measured value, normalized

M_ME_TD	Measured value, normalized with CP56Time2a timestamp
M_ME_NB	Measured value, scaled
M_ME_TE	Measured value, scaled with CP56Time2a timestamp
M_ME_NC	Measured value, short floating point
M_ME_TF	Measured value, short floating point with CP56Time2a timestamp
M_IT_NA	Integrated totals
M_IT_TB	Integrated totals with CP56Time2a timestamp

设备在总召唤期间发送不带时间戳的数据（如 M_SP_NA）时，模块也会匹配配置为对应带时间戳类型（如 M_SP_TB）的标签。这使你能为带时间戳的自发事件和 不带时间戳的召唤响应使用单个标签。

配置加载时检查地址。地址带空白、无法识别类型或重复地址和公共地址，会在状态日志中作为警告报告以便纠正。发送到类型不是有效命令类型地址的命令会被拒绝并记录，而不是作为默认命令发送。

命令地址（写）

Type ID	Description
C_SC_NA	Single command
C_SC_TA	Single command (same as C_SC_NA)
C_DC_NA	Double command
C_DC_TA	Double command (same as C_DC_NA)
C_SE_NA	Set-point command, normalized
C_SE_TA	Set-point command, normalized (same as C_SE_NA)
C_SE_NB	Set-point command, scaled
C_SE_TB	Set-point command, scaled (same as C_SE_NB)
C_SE_NC	Set-point command, short floating point
C_SE_TC	Set-point command, short floating point (same as C_SE_NC)

命令地址在 IOA 后支持以点分隔的额外选项：

- **SelectExecuteValue**——出现时，模块发送选择后执行序列。它首先发送带选择标志的命令，等待一秒，然后再次发送不带选择标志的命令。用于要求两步命令激活以保证安全的设备。
- **ShortPulse**——添加限定符值 1（短脉冲持续时间）。
- **LongPulse**——添加限定符值 2（长脉冲持续时间）。
- **Persistent**——添加限定符值 3（持久输出）。

命令地址示例：- C_SC_NA.100——对 IOA 100 的单命令，直接执行。- C_SC_NA.100.SelectExecuteValue——对 IOA 100 的单命令，选择后执行。- C_DC_NA.200.SelectExecuteValue.ShortPulse——对 IOA 200 的双命令，选择后执行并带短脉冲限定符。

示例

下面是一个完整示例，读取监视数据并向 IEC 60870-5-104 设备发送控制命令。

dxiec104.json:

```
{
  "servers": [{"url": "127.0.0.1:1883"}],
  "project": "PLANT1",
  "id": "iec104",
  "meta": {
    "name": "dxiec104"
  },
  "task": [
    {
      "enable": true,
      "id": "rtu1",
      "url": "192.168.1.50:2404",
      "trigger": 5000,
      "commonAddress": 1,
      "readOnly": false,
      "timeout": 3000,
      "data": [
        { "$csv": "dxiec104.csv" }
      ]
    }
  ]
}
```

dxiec104.csv:

name	description	address	scale	ca
breaker1	Breaker Status	M_SP_TB.100	1	
voltage	Bus Voltage	M_ME_NC.200	1	
energy	Total Energy	M_IT_NA.300	1	
cmd_breaker	Breaker Control	C_SC_NA.100.SelectExecuteValue	1	
cmd_setpoint	Voltage Setpoint	C_SE_NC.200	1	
breaker2	Breaker 2 Status	M_SP_TB.101	1	2

采用上述配置，模块将：

92. 在端口 2404 连接到 192.168.1.50 的 IEC 60870-5-104 设备。
93. 连接后发送总召唤命令，检索所有配置监视点的初始值。
94. 从设备接收单点、测量值和累计总量数据点的自发更新。标签名为“iec104.rtu1.breaker1”、“iec104.rtu1.voltage”、“iec104.rtu1.energy”等。
95. 接受“iec104.rtu1.cmd_breaker”（选择后执行的单命令）和“iec104.rtu1.cmd_setpoint”（短浮点设定值命令）的写请求。
96. 最后一个标签“breaker2”使用公共地址 2，覆盖任务级 commonAddress 1。当单个设备网关暴露多个逻辑站时很有用。

IEC 61850

DX IEC 61850 Client (dxiec61850) 对 IEC 61850 MMS 设备执行读写操作。它使用 MMS 协议通过 TCP/IP 连接到 IEC 61850 服务器 (IED)，同时支持轮询和基于报告的订阅两种数据采集方式。

关于 servers、project、模块 ID、trigger、update、data、存储转发和运行时等通用 DX 配置，请参阅 DX 驱动通用配置指南。

配置

下面是一个典型配置文件 dxiec61850.json 的示例。

```
{
  "servers": [{ "url": "127.0.0.1:1883" }],
  "project": "DEM0",
  "id": "module01",
  "meta": {
    "name": "dxiec61850"
  },
  "update": ["1000"],
  "task": [
    {
      "enable": true,
      "id": "device01",
      "desc": "IEC 61850 Master",
      "url": "127.0.0.1:102",
      "subscription": true,
      "trigger": 5000,
      "maxPoll": 0,
      "maxPdu": 1200,
      "readOnly": false,
      "data": [
        { "$csv": "dxiec61850.csv" }
      ]
    }
  ]
}
```

url

要连接的 IEC 61850 服务器 (IED) 地址，格式为“host:port”。默认值为“127.0.0.1:102”。端口 102 是 IEC 61850 通信的标准 MMS 端口。必须提供有效主机地址和不大于 65535 的端口号。在 DX Edit 中，主机和端口有单独框，新任务开始于“127.0.0.1:102”。

subscription

选择从 IED 检索数据的读取方式。设为 **true**（默认）时，模块使用 IEC 61850 报告控制块自动接收服务器的数据变化。每当配置的触发选项（数据变化、数据更新或总召唤）满足时，服务器会推送更新。

设为 **false** 时，模块改用轮询模式。在此模式下，模块在每个触发周期从服务器读取所有配置标签。当 IED 不支持报告，或希望自己控制读取间隔时，轮询模式很有用。

maxPoll

限制此任务从 IED 并发读取的最大标签数。默认值为 0，表示无限制——读取 **data** 部分定义的所有标签。设为大于零的值时，每个读取周期只包含该数量的标签（从数据列表顶部选取）。

maxPdu

允许的 MMS 协议数据单元（PDU）最大大小（字节）。默认值为 1200。控制与 IED 通信的最大消息大小。如果在单个请求中读取大量标签并遇到通信错误，可能需要增大此值；或减小它以匹配受限设备的能力。

readOnly

设为 **true** 时，模块只从 IED 读取数据，不接受写请求。设为 **false**（默认）时，模块对具有允许写入的有效功能约束的标签接受写命令。

自动发现

模块连接到 IED 时，自动浏览服务器上所有可用的逻辑设备、逻辑节点和数据对象。对找到的每个逻辑设备，在配置文件同文件夹中生成名为 `dxiec61850_{logicalDeviceName}.all.csv` 的文件。此文件包含每个发现的标签及其地址、功能约束和数据类型，可作为构建自己 CSV 配置的起点。

自动发现文件只创建一次。如果文件已存在，不会覆盖。

CSV

CSV 文件定义从 IED 读取（以及可选写入）的标签列表。列如下：

<code>disable,name,description,address,fc,type,length,default,scale</code>
--

- **disable**——设为“true”可跳过此条目，设为“false”则启用。
- **name**——标签名。系统中的完整标签名遵循“moduleId.taskId.name”的命名约定。
- **description**——可选的描述，供你参考。
- **address**——数据属性的 IEC 61850 对象引用，格式为“LogicalDevice/LogicalNode.DataObject.DataAttribute”。例如“simpleIOGenericIO/GGIO1.AnIn1.mag.f”。
- **fc**——数据属性的功能约束。常见值包括“MX”（测量）、“ST”（状态）、“CO”（控制）、“CF”（配置）和“SP”（设定值）。必须与 IED 数据模型中定义的功能约束匹配。
- **type**——标签的数据类型。只识别以下类型：bool、int32、uint32、real32 和 string。如果不确定类型，参考自动发现的 CSV 文件。
- **length**——数据长度。使用 -1 表示默认。
- **default**——连接失败时标签的默认值。留空以保留最后一个有效值。
- **scale**——原始值的可选缩放表达式。

示例

下面是一个完整示例，从两台 IEC 61850 设备读取数据。

dxiec61850.json:

```
{
  "servers": [{"url": "127.0.0.1:1883"}],
  "project": "PLANT1",
  "id": "iec",
  "meta": {
    "name": "dxiec61850"
  },
  "update": ["1000"],
  "task": [
    {
      "enable": true,
      "id": "relay1",
      "desc": "Protection Relay",
      "url": "192.168.1.10:102",
      "subscription": true,
      "trigger": 5000,
      "maxPdu": 1200,
      "readOnly": true,
    }
  ]
}
```

```

        "data": [
            { "$csv": "relay1.csv" }
        ]
    },
    {
        "enable": true,
        "id": "ctrl1",
        "desc": "Bay Controller",
        "url": "192.168.1.20:102",
        "subscription": false,
        "trigger": 1000,
        "maxPdu": 2048,
        "readOnly": false,
        "data": [
            { "$csv": "controller1.csv" }
        ]
    }
]
}

```

relay1.csv:

name	description	address	fc	type	length
voltage	Bus Voltage	relayLD/MMXU1.PhV.phsA.cVal.mag.f	MX	real32	-1
current	Line Current	relayLD/MMXU1.A.phsA.cVal.mag.f	MX	real32	-1
cbpos	Breaker Position	relayLD/XCBR1.Pos.stVal	ST	bool	-1

controller1.csv:

name	description	address	fc	type	length
temp	Transformer Temp	bayLD/GGIO1.AnIn1.mag.f	MX	real32	-1
mode	Operating Mode	bayLD/GGIO1.SPCSO1.stVal	ST	int32	-1

采用上述配置，模块将：

97. 使用基于报告的订阅连接到 192.168.1.10 的保护继电器。标签值如“iec.relay1.voltage”、“iec.relay1.current”和“iec.relay1.cbpos”会在 IED 报告变化时自动更新。此任务为只读。
98. 使用轮询模式每 1 秒连接到 192.168.1.20 的间隔控制器。标签“iec.ctrl1.temp”和“iec.ctrl1.mode”在每个周期读取。由于未启用 readOnly，对具有有效功能约束的标签也接受写请求。

Profinet

DX PROFINET 驱动（dxprofinet）通过 S7 通信协议与西门子 S7 系列 PLC 读写数据。支持 S7-200、S7-200 Smart、S7-300、S7-400、S7-1200 和 S7-1500 型号。

关于 servers、project、模块 ID、trigger、update、data、存储转发和运行时等通用 DX 配置，请参阅 DX 驱动通用配置指南。

配置

下面是一个典型配置文件 dxprofinet.json 的示例。

```
{
  "servers": [{"url": "127.0.0.1:1883"}],
  "project": "DEM0",
  "id": "profinet",
  "meta": {
    "name": "dxprofinet"
  },
  "task": [
    {
      "enable": true,
      "id": "plc1",
      "url": "192.168.1.10:102",
      "trigger": 1000,
      "model": "S7-1200",
      "slot": 0,
      "timeout": 3000,
      "readOnly": false,
      "data": [
        { "$csv": "dxprofinet.csv" }
      ]
    }
  ]
}
```

任务参数

url

西门子 PLC 的 IP 地址和端口。S7 通信的默认端口为 102。可以只指定 IP 地址（如“192.168.1.10”），驱动会自动使用端口 102；或显式包含端口（如“192.168.1.10:102”）。在 DX Edit 中，主机和端口有单独框，新任务开始于“127.0.0.1:102”。

model

要连接的西门子 PLC 型号。这决定驱动使用的通信配置。可用选项如下：

Value	PLC Model
S7-1200	Siemens S7-1200 (default)
S7-1500	Siemens S7-1500
S7-300	Siemens S7-300
S7-400	Siemens S7-400
S7-200	Siemens S7-200
S7-200SMART	Siemens S7-200 Smart

如果值不匹配上述任何选项，驱动回退为 S7-1200。

rack

PLC 的机架号。主要用于支持多机架配置的 S7-300 和 S7-400 PLC。默认 0。

slot

PLC CPU 模块在机架中的插槽号。主要用于 S7-300 和 S7-400 PLC。默认 0。对于 S7-1200 和 S7-1500，插槽通常为 0 或 1。

timeout

连接和接收超时（毫秒）。默认 3000 ms。初始连接尝试和每次数据读写操作都受此超时限制。

localTSAP

本地传输服务访问点标识符。默认 4096（0x1000）。通常无需更改，除非 PLC 要求特定 TSAP 配置。

destTSAP

目标传输服务访问点标识符。默认 768（0x0300）。某些 PLC 型号或配置可能需要不同值。如果默认值连接失败，参考 PLC 的通信设置。

readOnly

设为 true 时，驱动只从 PLC 读取，所有写操作禁用。默认 false。

CSV

CSV 文件定义从 PLC 读取或写入 PLC 的数据点列表。列如下：

name	address	type	length	default	scale
Bool1	M100	boolean	-1	1	1

- **disable**——设为“true”可跳过此条目，设为“false”则启用。
- **name**——此数据点的标签名。
- **description**——此数据点的描述。
- **address**——数据点的西门子 S7 地址。遵循标准西门子地址记号（见下文地址格式部分）。
- **type**——数据类型。支持类型：boolean、int8（byte）、int16（short）、uint16（ushort）、int32（int）、uint32（uint）、int64（long）、uint64（ulong）、real32（float）、real64（double）和 string（str）。
- **length**——数据长度。使用 -1 表示默认长度。对于字符串类型，输入要读取的字符数。
- **default**——标签的默认值。
- **scale**——读取时应用于原始值的缩放因子，写入时反向应用。例如 scale 为 10 表示读取值乘以 10、写入值除以 10。scale 不适用于 boolean 或 string 类型。

地址格式

驱动使用标准西门子 S7 地址记号。常见地址格式包括：

Address	Description
M100	Merker byte 100
M100.0	Merker bit 100.0 (for boolean)
I0.0	Input bit 0.0
Q0.0	Output bit 0.0
DB1.DBB0	Data block 1, byte 0
DB1.DBW0	Data block 1, word 0
DB1.DBD0	Data block 1, double word 0
DB1.DBX0.0	Data block 1, bit 0.0
MW100	Merker word 100
MD100	Merker double word 100

对于 boolean 类型，使用位级地址（如 M100.0、I0.0、DB1.DBX0.0）。对于数值和字符串类型，使用字节级或字级地址（如 MW100、DB1.DBW0、DB1.DBD0）。

示例

下面是一个完整示例，两个任务连接到不同 PLC：

```

{
  "servers": [{"url": "127.0.0.1:1883"}],
  "project": "DEM0",
  "id": "siemens1",
  "meta": {
    "name": "dxprofinet"
  },
  "task": [
    {
      "enable": true,
      "id": "plc1",
      "url": "192.168.1.10",
      "trigger": 1000,
      "model": "S7-1200",
      "slot": 0,
      "timeout": 3000,
      "data": [
        { "$csv": "dxprofinet.csv" }
      ]
    },
    {
      "enable": true,
      "id": "plc2",
      "url": "192.168.1.20",
      "trigger": 2000,
      "model": "S7-300",
      "rack": 0,
      "slot": 2,
      "timeout": 5000,
      "readOnly": true,
      "data": [
        { "$csv": "dxprofinet_plc2.csv" }
      ]
    }
  ]
}

```

以及关联的 CSV 文件:

name	description	address	type	length	default	scale
MotorOn	Motor running status	M100.0	boolean	-1		1
Temperature	Tank temperature	MW200	real32	-1	0	1
Pressure	Line pressure	DB1.DBD0	real32	-1	0	10
Counter	Production count	MW300	int32	-1	0	1
BatchName	Current batch name	DB2.DBB0	string	20		1

在此示例中，第一个任务每秒轮询一台 S7-1200 PLC，第二个任务以只读模式每 2 秒轮询一台 S7-300 PLC。标签名为“siemens1.plc1.MotorOn”、“siemens1.plc2.Temperature”等。

EtherNet/IP

DX AB EtherNet/IP 驱动（dxabeip）通过 EtherNet/IP 协议与 Allen-Bradley 可编程控制器读写数据。支持多种 Allen-Bradley 设备系列，包括 ControlLogix/CompactLogix（1756/1769）、SLC（1747）、Micro800 和 DF1 串口。

关于 servers、project、模块 ID、trigger、update、data、存储转发和运行时等通用 DX 配置，请参阅 DX 驱动通用配置指南。

配置

下面是一个典型配置文件 dxabeip.json 的示例。

```
{
  "servers": [{ "url": "127.0.0.1:1883" }],
  "project": "DEM0",
  "id": "module01",
  "meta": {
    "name": "dxabeip"
  },
  "task": [
    {
      "enable": true,
      "id": "abeip01",
      "deviceType": "1756/1769",
      "deviceID": 1,
      "url": "192.168.1.10",
      "trigger": [1000],
      "readOnly": false,
      "data": [
        { "$csv": "dxabeip.csv" }
      ]
    }
  ]
}
```

deviceType

指定要通信的 Allen-Bradley 设备系列。默认值为“1756/1769”。支持值如下：

Value	Device Family
1756/1769	ControlLogix and CompactLogix controllers
1747	SLC 5/05 controllers
micro800	Micro800 series controllers (Micro820, Micro850, etc.)
df1serial	DF1 Serial protocol for legacy controllers

每种设备类型使用不同的底层通信方式。确保为你的控制器选择正确类型。

deviceID

控制器在机架中的插槽号。适用于控制器可能位于不同插槽的“1756/1769”和“micro800”设备类型。对于“df1serial”，此值用作站地址。默认值为 0。值必须是非负整数。

url

控制器的 IP 地址和可选端口。EtherNet/IP 的默认端口为 44818，未指定端口时使用。例如“192.168.1.10”在端口 44818 连接，而“192.168.1.10:5000”在端口 5000 连接。在 DX Edit 中，主机和端口有单独框，新任务开始于“127.0.0.1:44818”。

对于“df1serial”设备类型，url 应设为串口名（如“COM1”）而非 IP 地址。

readOnly

设为 true 时，模块只从控制器读取数据，忽略任何写请求。设为 false 或省略时，读取和写入都启用。

CSV

CSV 文件定义从控制器读取或写入控制器的标签列表。默认的 CSV 列如下：

name	description	address	type	length	scale
tag1	My first tag	MyDINT	int32	1	1
tag2	A boolean tag	MyBool	bool	1	1
tag3	A real value	MyReal	float	1	1
tag4	A string tag	MyString	string	1	1

- **disable**——设为“true”可跳过此条目，设为“false”则启用。
- **name**——标签名。系统中的完整标签名遵循“moduleId.taskId.name”的命名约定。
- **description**——可选的描述，供你参考。
- **address**——控制器中的标签地址。使用控制器程序中定义的标签名（如“MyDINT”、“Program:MainProgram.Counter”、“MyArray[0]”）。对于 ControlLogix/CompactLogix 上的字符串标签，驱动自动读取 STRING 结构的 .LEN 和 .DATA 成员。

- **type**——标签的数据类型。支持值如下。
- **length**——数据长度。默认值为 1。
- **scale**——应用于原始值的缩放因子。默认值为 1，表示不缩放。

支持的数据类型

Type	Aliases	Description
bool	boolean	Boolean value (true/false)
byte		8-bit unsigned integer
int16	short	16-bit signed integer
uint16	ushort	16-bit unsigned integer
int32	int	32-bit signed integer
uint32	uint	32-bit unsigned integer
int64	long	64-bit signed integer
uint64	ulong	64-bit unsigned integer
float	real32	32-bit floating point
double	real64	64-bit floating point
string	str	String value

数据类型名大小写不敏感。可使用主名或别名。

示例

下面是一个完整示例，从机架中的两台控制器读取。

dxabeip.json:

```
{
  "servers": [{"url": "127.0.0.1:1883"}],
  "project": "PLANT1",
  "id": "ab1",
  "meta": {
    "name": "dxabeip"
  },
  "task": [
    {
      "enable": true,
      "id": "plc1",
      "deviceType": "1756/1769",
      "deviceID": 0,
      "url": "192.168.1.10",
      "trigger": [1000],
      "data": [
        { "$csv": "dxabeip-plc1.csv" }
      ]
    },
    {
      "enable": true,
      "id": "plc2",
      "deviceType": "1756/1769",
```

```

        "deviceID": 2,
        "url": "192.168.1.10",
        "trigger": [2000],
        "readOnly": true,
        "data": [
            { "$csv": "dxabeip-plc2.csv" }
        ]
    }
]
}

```

dxabeip-plc1.csv:

name	description	address	type	length	scale
speed	Motor Speed	MotorSpeed	float	1	1
running	Motor Running	MotorRunning	bool	1	1
count	Production Count	ProdCount	int32	1	1

dxabeip-plc2.csv:

name	description	address	type	length	scale
temp	Tank Temperature	TankTemp	float	1	1
level	Tank Level	TankLevel	float	1	1
alarm	High Level Alarm	HighLevelAlm	bool	1	1

采用上述配置，模块将：

99. 每 1 秒轮询 192.168.1.10 上插槽 0 的控制器，获取电机速度、运行状态和生产计数。标签名为“ab1.plc1.speed”、“ab1.plc1.running”和“ab1.plc1.count”。写入已启用。
100. 每 2 秒轮询同一 IP 上插槽 2 的控制器，获取储罐温度、液位和报警。标签名为“ab1.plc2.temp”、“ab1.plc2.level”和“ab1.plc2.alarm”。由于 readOnly 设为 true，写入已禁用。

Beckhoff ADS

DX Beckhoff ADS 驱动（dxads）使用 ADS（自动化设备规范）协议对 Beckhoff TwinCAT 设备执行读写操作。通过指定目标 AMS Net ID、端口和每个数据点的内存地址，可经 TCP 连接与 TwinCAT PLC 通信。

关于 servers、project、模块 ID、trigger、update、data、存储转发和运行时等通用 DX 配置，请参阅 DX 驱动通用配置指南。

配置

下面是一个典型配置文件 dxads.json 的示例。

```
{
  "servers": [{ "url": "127.0.0.1:1883" }],
  "project": "DEM0",
  "id": "ads",
  "meta": {
    "name": "dxads"
  },
  "task": [
    {
      "enable": true,
      "id": "plc1",
      "url": "127.0.0.1:851",
      "slot": "1",
      "trigger": [1000],
      "timeout": 3000,
      "data": [
        { "$csv": "dxads.csv" }
      ]
    }
  ]
}
```

url

目标 Beckhoff TwinCAT 设备的 IP 地址和 ADS 端口，格式为“host:port”。默认值为“127.0.0.1:851”。端口 851 是 TwinCAT 3 PLC 运行时 1 的标准 ADS 端口。运行时 2 使用端口 852，运行时 3 使用 853，依此类推。省略端口时使用默认端口 851。在 DX Edit 中，主机和端口有单独框，新任务开始于“127.0.0.1:851”。

slot

分配给 ADS 设备的插槽号。默认值为“1”。必须是非负整数。

timeout

连接超时（毫秒）。默认值为 3000 ms。控制模块建立到目标设备连接时等待的时间。设备位于慢速或不可靠网络上时增大此值。

readOnly

设为 true 时，模块只从设备读取数据，忽略任何写请求。默认值为 false，允许读取和写入。

CSV

CSV 文件定义从 Beckhoff 设备读取或写入设备的数据点列表。默认的 CSV 列如下：

name	address	type	length	default
tag1	M100	boolean	-1	true

- **disable**——设为“true”可跳过此条目，设为“false”则启用。
- **name**——此数据点的标签名。系统中的完整标签名遵循“moduleId.taskId.name”的命名约定。
- **description**——可选的描述，供你参考。
- **address**——要读取或写入的 TwinCAT 内存地址。使用标准 Beckhoff 地址记号，如标记地址“M100”、数据块地址“DB1.DBBD0”，或根据 TwinCAT 项目配置使用符号变量名。
- **type**——给定地址处值的数据类型。支持类型见下表。
- **length**——字符串类型读取的字节长度。必须是非负整数。非字符串类型设为 -1 或留空。
- **default**——标签的默认初始值。
- **scale**——读取时对数值应用的倍数，写入时相除。留空或设为 1 表示不缩放。例如 scale 为 10 会将读取的原始值乘以 10，发送到设备前将写入值除以 10。Boolean 和 string 类型不受缩放影响。

支持的数据类型

Type	Alias
boolean	bool
int8	byte
int16	short

uint16	ushort
int32	int
uint32	uint
int64	long
uint64	ulong
real32	float
real64	double
string	str

CSV 中可使用类型名或别名。类型名大小写不敏感。

示例

下面是一个完整示例，从 TwinCAT PLC 读写各种数据类型。

dxads.json:

```
{
  "servers": [{"url": "127.0.0.1:1883"}],
  "project": "PLANT1",
  "id": "ads",
  "meta": {
    "name": "dxads"
  },
  "task": [
    {
      "enable": true,
      "id": "plc1",
      "url": "192.168.1.10:851",
      "slot": "1",
      "trigger": [1000],
      "timeout": 3000,
      "data": [
        { "$csv": "dxads.csv" }
      ]
    },
    {
      "enable": true,
      "id": "plc2",
      "url": "192.168.1.11:851",
      "slot": "1",
      "trigger": [2000],
      "timeout": 5000,
      "readOnly": true,
      "data": [
        { "$csv": "dxads-plc2.csv" }
      ]
    }
  ]
}
```

dxads.csv:

name	description	address	type	length	default
motor_run	Motor Running Status	M100	boolean	-1	false
speed	Motor Speed RPM	M104	real32	-1	0
setpoint	Speed Setpoint	M108	real32	-1	0
counter	Production Count	M112	int32	-1	0
label	Product Label	M200	string	50	

dxads-plc2.csv:

name	description	address	type	length	default	scale
temp_raw	Raw Temperature	M100	int16	-1	0	0.1
pressure	Tank Pressure	M102	real32	-1	0	

采用上述配置，模块将：

101. 每 1 秒轮询 192.168.1.10 的 plc1，获取五个标签，包括 boolean、float、integer 和 string 类型。标签名为“ads.plc1.motor_run”、“ads.plc1.speed”等。读取和写入都启用。
102. 以只读模式每 2 秒轮询 192.168.1.11 的 plc2，超时延长为 5 秒。“temp_raw”标签使用 0.1 的 scale，因此 PLC 原始值 250 会发布为 25.0。

Omron FINS

DX FINS Master（dx fins）使用 FINS（工厂接口网络服务）协议与欧姆龙 PLC 读写数据。同时支持 FINS over TCP 和 FINS over UDP。

关于 servers、project、模块 ID、trigger、update、data、存储转发和运行时等通用 DX 配置，请参阅 DX 驱动通用配置指南。

配置

下面是一个典型配置文件 dx fins.json 的示例。

```
{
  "servers": [{ "url": "127.0.0.1:1883" }],
  "project": "DEM0",
  "id": "fins",
  "meta": {
    "name": "dx fins"
  },
  "task": [
    {
      "enable": true,
      "id": "fins01",
      "url": "192.168.1.10:9600",
      "trigger": 1000,
      "DA1": 0,
      "DA2": 0,
      "DNA": 0,
      "SA1": 103,
      "SA2": 0,
      "SNA": 0,
      "readOnly": false,
      "data": [
        { "$csv": "dx fins.csv" }
      ]
    }
  ]
}
```

连接类型

驱动支持两种传输协议，由 udp 参数决定。

Mode	Description
TCP (default)	FINS over TCP. The driver maintains a persistent connection to the PLC.
UDP	FINS over UDP. Set "udp": true in the task configuration.

url 应为欧姆龙 PLC 的 IP 地址（或主机名）及 FINS 端口。默认 FINS 端口为 9600。省略端口时默认为 9600。

任务参数

url

欧姆龙 PLC 的 IP 地址和端口。例如"192.168.1.10:9600"或"192.168.1.10"（端口默认为 9600）。在 DX Edit 中，主机和端口有单独框，新任务开始于“127.0.0.1:9600”。

timeout

套接字接收超时（毫秒）。默认 500 ms。如果设置 0 或以下值，自动提升为 500。慢速或拥塞网络上遇到超时时增大此值。

udp

设为 true 时使用 FINS over UDP 而非 FINS over TCP。默认 false。

readOnly

设为 true 时，驱动只从 PLC 读取，所有写操作禁用。默认 false。

dataFormat

控制多字数据类型（real32、int32、uint32、real64、int64、uint64）的字节序。可用选项如下：

Value	Byte Order
ABCD	Big-endian
BADC	Big-endian byte swap
CDAB	Little-endian byte swap (default)
DCBA	Little-endian

如果多字值显示乱码或不正确，尝试不同字节序以匹配 PLC 格式。注意，此驱动器的默认值为 CDAB，这是欧姆龙 PLC 的典型字节序。

reverse

设为 true 时，反转字符串数据中每个字内的字节序。默认 false。从 PLC 读取的字符串值出现字符交换时使用。

FINS 头参数

FINS 协议需要头字段来标识网络上的源和目标节点。通常只有在跨网络网桥通信或有特定节点配置时才需要更改。

DA1

目标节点号。默认 0。设为目标 PLC 在 FINS 网络上的节点号。

DA2

目标单元地址。默认 0。标识目标节点内的特定单元（CPU 或特殊模块）。

DNA

目标网络地址。默认 0。跨 FINS 网络网桥路由时，设为目标 PLC 所在的网络号。

SA1

源节点号。默认 103。这是驱动在 FINS 网络上标识自己的节点号。确保不与同一网络上的其他节点冲突。

SA2

源单元地址。默认 0。

SNA

源网络地址。默认 0。跨 FINS 网络网桥路由时，设为本地网络号。

ICF

信息控制字段。默认 128。这是协议级设置，通常应保持默认值。

SID

服务 ID。默认 0。用于匹配请求和响应。除非有特定需要区分并发会话，否则保持默认即可。

CSV

CSV 文件定义从欧姆龙 PLC 读取或写入 PLC 的数据点列表。列如下：

name	description	address	batch	type	length	scale
HR01	Holding register 1	H100.1	-1	boolean	-1	1

- **disable**——设为“true”可跳过此条目，设为“false”则启用。

- **name**——此数据点的标签名。
- **description**——此数据点的描述。
- **address**——FINS 内存区地址（见下文地址格式）。
- **batch**——控制数据点的读取方式。设为 -1 可将标签纳入优化批量读取。设为 0 或正数则单独读取，该值指定从地址开始读取的连续元素数。
- **type**——数据类型。支持类型：boolean、int16（或 short）、uint16（或 ushort）、int32（或 int）、uint32（或 uint）、int64（或 long）、uint64（或 ulong）、real32（或 float）、real64（或 double）。
- **length**——数据长度。使用 -1 表示默认长度。
- **scale**——读取时应用于原始值的缩放因子，写入时相除。例如 scale 为 10 表示读取时原始值乘以 10。不适用于 boolean 类型。

地址格式

地址遵循欧姆龙 FINS 内存区记号。格式为内存区前缀加字地址，可选以点分隔的位位置。

Prefix	Memory Area	Example
D	Data Memory (DM)	D100
C	CIO Area	C0
W	Work Area (WR)	W31
H	Holding Area (HR)	H100
A	Auxiliary Area (AR)	A0
E	Extended Memory (EM)	E0 0
TIM	Timer	TIM0

对于访问字内特定位的 boolean 类型，使用{area}{word}.{bit}格式。例如 H100.1 读取保持寄存器 100 的位 1。不带位位置的布尔地址（如 D100）读取整个字，值非零时返回 true。

对于定时器（TIM）地址，格式简单为 TIM{number}，例如 TIM0 或 TIM5。

示例

下面是一个完整示例，TCP 任务轮询欧姆龙 PLC：

```
{
  "servers": [{ "url": "127.0.0.1:1883" }],
  "project": "DEM0",
  "id": "plc1",
```

```

    "meta": {
      "name": "dxfins"
    },
    "task": [
      {
        "enable": true,
        "id": "cpu1",
        "url": "192.168.1.10:9600",
        "trigger": 1000,
        "DA1": 0,
        "DA2": 0,
        "SA1": 103,
        "dataFormat": "CDAB",
        "readOnly": false,
        "data": [
          { "$csv": "dxfins.csv" }
        ]
      }
    ]
  }
}

```

以及关联的 CSV 文件 dxfins.csv:

name	description	address	batch	type	length	scale
Bit01	Holding register bit	H100.1	-1	boolean	-1	1
DM100	Data memory word	D100	-1	int16	-1	1
DM200	Data memory float	D200	-1	real32	-1	1
WR10	Work area register	W10	-1	uint16	-1	1
Timer0	Timer status	TIM0	-1	boolean	-1	1

在此示例中，标签将命名为“plc1.cpu1.Bit01”、“plc1.cpu1.DM100”等。

下面是使用 UDP 传输的示例：

```

{
  "servers": [{"url": "127.0.0.1:1883"}],
  "project": "DEM0",
  "id": "plc2",
  "meta": {
    "name": "dxfins"
  },
  "task": [
    {
      "enable": true,
      "id": "dev1",

```

```
        "url": "192.168.1.20:9600",
        "trigger": 500,
        "udp": true,
        "DA1": 1,
        "SA1": 103,
        "data": [
            { "$csv": "dx fins.csv" }
        ]
    }
}
```

Omron Hostlink

DX HostLink Master（dxhostlink）使用 Host Link（FINS）协议与欧姆龙 PLC 读写数据。同时支持串口（Host Link）和 TCP（Host Link over TCP）连接，提供 Standard 和 C-Mode 两种通信类型。

关于 servers、project、模块 ID、trigger、update、data、存储转发和运行时等通用 DX 配置，请参阅 DX 驱动通用配置指南。

配置

下面是一个典型配置文件 dxhostlink.json 的示例。

```
{
  "servers": [{ "url": "127.0.0.1:1883" }],
  "project": "DEM0",
  "id": "HostLink",
  "meta": {
    "name": "dxhostlink"
  },
  "task": [
    {
      "enable": true,
      "id": "HostLink01",
      "url": "COM1",
      "trigger": 1000,
      "baudRate": 9600,
      "dataBits": 7,
      "stopBits": 2,
      "parity": "e",
      "unitNum": 0,
      "timeout": 5000,
      "dataFormat": "ABCD",
      "readOnly": false,
      "data": [
        { "$csv": "dxhostlink.csv" }
      ]
    }
  ]
}
```

连接类型

驱动根据 url 值自动确定连接类型。

URL Format	Connection Type
Serial port on Windows (e.g. “COM1”)	Host Link over serial
Serial port on Linux (e.g. “/dev/ttyS0”,	Host Link over serial

“/dev/ttyUSB0”, “/dev/ttyAMA0”)	
IP address with port (e.g. “192.168.1.10:2000”)	Host Link over TCP

任务参数

type

通信协议类型。接受值：

- **Standard**——使用 FINS/Host Link 协议。这是默认。
- **CMode**——使用 Host Link C-Mode 协议，这是大多数欧姆龙 PLC 支持的较旧、较简单的命令集。

未指定时，驱动默认为 Standard。

mode

连接时设置的 PLC 运行模式。接受值：

- **Run**——正常执行模式。这是默认。
- **Monitor**——PLC 运行程序但允许在线监视和强制 I/O。
- **Program**——PLC 停止执行用户程序。

此参数仅用于 C-Mode 连接。未指定时，驱动默认为 Run。

unitNum

目标 PLC 在 Host Link 网络上的单元号。默认 0。

netNum

计算机的网络号（SA2），用于 Standard 模式下的 FINS 路由。默认 0。

DA2

Standard 模式下用于 FINS 路由的 PLC 单元号地址。默认 0。对于大多数单 PLC 设置，保持默认即可。

ICF

Standard 模式下使用的 FINS 信息控制字段字节。默认 0。通常无需更改，除非网络配置要求特定路由设置。

SID

Standard 模式下使用的 FINS 服务 ID 字节。默认 0。用于多节点网络中匹配请求和响应。

timeout

接收超时（毫秒）。默认 5000 ms。如果 PLC 在此时间内未响应，驱动将请求视为失败。

dataFormat

控制 32 位和 64 位数据类型的字节序。可用选项如下：

Value	Byte Order
ABCD	Big-endian (default)
BADC	Big-endian byte swap
CDAB	Little-endian byte swap
DCBA	Little-endian

如果多字值显示乱码或不正确，尝试不同字节序以匹配 PLC 格式。

readOnly

设为 true 时，驱动只从 PLC 读取，所有写操作禁用。默认 false。

rts

设为 true 时，在串口上启用 RTS（请求发送）信号。默认 false。RS-232/RS-485 转换器需要 RTS 信号控制数据方向时启用。

串口参数

使用串口 URL 时，以下参数配置串口通信。默认值匹配典型欧姆龙 Host Link 设置。

baudRate

串口波特率。默认 9600。

dataBits

每字节的数据位数。默认 7。

stopBits

每字节的停止位数。接受值 1、1.5 或 2。默认 2。

parity

校验协议。接受值：

- **none**（或“n”）——无校验。
- **odd**（或“o”）——奇校验。
- **even**（或“e”）——偶校验。这是默认。
- **space**（或“s”）——空校验。
- **mark**（或“m”）——标记校验。

注意：默认串口设置（9600 波特、7 数据位、2 停止位、偶校验）匹配大多数欧姆龙 PLC Host Link 通信的出厂默认。

CSV

CSV 文件定义从 PLC 读取或写入 PLC 的数据点列表。列如下：

name	description	address	type	length	scale
HR01	Holding relay 1	H100	boolean	-1	1

- **disable**——设为“true”可跳过此条目，设为“false”则启用。
- **name**——此数据点的标签名。
- **description**——此数据点的描述。
- **address**——PLC 内存区和地址。使用区前缀加字地址号（见下文地址格式）。
- **type**——数据类型。支持类型包括 boolean、int16、uint16、int32、uint32、real32（float）、int64、uint64 和 real64（double）。
- **length**——数据长度。使用 -1 表示默认长度。
- **scale**——应用于原始值的缩放因子。例如 scale 为 10 会在发布前将读取值乘以 10。

地址格式

地址遵循欧姆龙 FINS 内存区约定，使用前缀字母加字地址号。常见内存区前缀包括：

Prefix	Memory Area	Description
D	DM (Data Memory)	General-purpose data storage
H	HR (Holding Relay)	Retains state across power cycles

W	WR (Work Relay)	Internal work bits
A	AR (Auxiliary Relay)	System and status information
C	CIO (Core I/O)	Input/output and internal relay area
E	EM (Extended Memory)	Additional data banks

例如 D100 指 DM 字 100，H10 指 HR 字 10。

对于 boolean 类型，可使用 prefix + word.bit 格式寻址单个位，如 D100.0 表示 DM 字 100 的位 0。

示例

下面是一个完整示例，使用 Standard Host Link 串口连接欧姆龙 PLC：

```
{
  "servers": [{"url": "127.0.0.1:1883"}],
  "project": "DEM0",
  "id": "plc1",
  "meta": {
    "name": "dxhostlink"
  },
  "task": [
    {
      "enable": true,
      "id": "dev1",
      "url": "COM3",
      "trigger": 1000,
      "baudRate": 9600,
      "dataBits": 7,
      "stopBits": 2,
      "parity": "e",
      "unitNum": 0,
      "timeout": 5000,
      "dataFormat": "ABCD",
      "readOnly": false,
      "data": [
        { "$csv": "dxhostlink.csv" }
      ]
    }
  ]
}
```

下面是使用 Host Link over TCP 及 C-Mode 的示例：

```
{
  "servers": [{"url": "127.0.0.1:1883"}],
  "project": "DEM0",
  "id": "plc2",
  "meta": {
    "name": "dxhostlink"
  }
}
```

```
    },
    "task": [
      {
        "enable": true,
        "id": "dev1",
        "url": "192.168.1.10:9600",
        "trigger": 2000,
        "type": "CMode",
        "mode": "Run",
        "unitNum": 0,
        "timeout": 5000,
        "readOnly": true,
        "data": [
          { "$csv": "dxhostlink.csv" }
        ]
      }
    ]
  }
}
```

以及关联的 CSV 文件 dxhostlink.csv:

name	description	address	type	length	scale
Temperature	Process temperature	D100	real32	-1	1
Pressure	Tank pressure	D102	uint16	-1	10
RunStatus	Motor running	H10	boolean	-1	1
Counter	Production count	D200	int32	-1	1

Mitsubishi MELSEC

DX MELSEC 驱动（dxmelsecalc）通过 TCP 与三菱 MELSEC A-1E 系列 PLC 读写数据。同时支持 Binary 和 ASCII 两种通信格式。

关于 servers、project、模块 ID、trigger、update、data、存储转发和运行时等通用 DX 配置，请参阅 DX 驱动通用配置指南。

配置

下面是一个典型配置文件 dxmelsecalc.json 的示例。

```
{
  "servers": [{ "url": "127.0.0.1:1883" }],
  "project": "DEM0",
  "id": "Melsec",
  "meta": {
    "name": "dxmelsecalc"
  },
  "task": [
    {
      "enable": true,
      "id": "Melsec01",
      "url": "192.168.1.10:6000",
      "trigger": 1000,
      "type": "binary",
      "timeout": 5000,
      "readOnly": false,
      "data": [
        { "$csv": "dxmelsecalc.csv" }
      ]
    }
  ]
}
```

任务参数

url

MELSEC PLC 的 IP 地址和端口。格式为“host:port”，例如“192.168.1.10:6000”。如果只提供 IP 地址而没有端口，使用默认端口 6000。在 DX Edit 中，主机和端口有单独框，新任务开始于“127.0.0.1:6000”。

type

与 PLC 通信使用的通信格式。可用选项如下：

Value	Description
binary	Binary protocol format (default)
ascii	ASCII protocol format

除非 PLC 或网络设置明确要求 ASCII 帧，否则使用“binary”。值大小写不敏感。

timeout

与 PLC 通信的接收超时（毫秒）。默认 5000 ms。慢速或拥塞网络上频繁超时时增大此值。

readOnly

设为 true 时，驱动只从 PLC 读取，所有写操作禁用。默认 false。

CSV

CSV 文件定义从 MELSEC PLC 读取或写入 PLC 的数据点列表。列如下：

name	description	address	type	length	scale
HR01	Holding register 1	D100	int16	-1	1

- **disable**——设为“true”可跳过此条目，设为“false”则启用。
- **name**——此数据点的标签名。
- **description**——此数据点的描述。
- **address**——MELSEC 设备地址。使用标准 MELSEC 地址记号，设备前缀加地址号，例如数据寄存器 100 为“D100”、内部继电器 100 为“M100”、输入为“X0”、输出为“Y0”。
- **type**——数据类型。支持类型见下表。
- **length**——数据长度。使用 -1 表示默认长度。对于字符串类型，输入要读取的字符数（字数）。
- **scale**——读取时应用于原始值的缩放因子。写入时，值在发送到 PLC 前除此因子。默认 1（不缩放）。

支持的数据类型

Type	Alias	Description
boolean	bool	Single bit value
int16	short	Signed 16-bit integer (1 word)
uint16	ushort	Unsigned 16-bit integer (1 word)

int32	int	Signed 32-bit integer (2 words)
uint32	uint	Unsigned 32-bit integer (2 words)
int64	long	Signed 64-bit integer (4 words)
uint64	ulong	Unsigned 64-bit integer (4 words)
real32	float	32-bit floating point (2 words)
real64	double	64-bit floating point (4 words)
string	str	Character string, requires length to be specified

地址格式

MELSEC 地址使用设备字母前缀加地址号。常见设备类型包括：

Prefix	Device Type
D	Data register
M	Internal relay (bit)
X	Input relay (bit)
Y	Output relay (bit)
W	Link register
R	File register

对于 boolean 类型，可使用点号访问字寄存器内的单个位。例如“D100.0”读取数据寄存器 D100 的位 0，“D100.15”读取位 15。

示例

下面是一个完整示例，两个任务轮询不同 PLC：

```
{
  "servers": [{ "url": "127.0.0.1:1883" }],
  "project": "DEM0",
  "id": "plc",
  "meta": {
    "name": "dxmelsec1e"
  },
  "task": [
    {
      "enable": true,
      "id": "line1",
      "url": "192.168.1.10:6000",
      "trigger": 1000,
      "type": "binary",
      "timeout": 5000,
      "readOnly": false,
      "data": [
        { "$csv": "line1.csv" }
      ]
    }
  ],
}
```

```

    {
      "enable": true,
      "id": "line2",
      "url": "192.168.1.20:6000",
      "trigger": 2000,
      "type": "ascii",
      "timeout": 5000,
      "readOnly": true,
      "data": [
        { "$csv": "line2.csv" }
      ]
    }
  ]
}

```

以及示例 CSV 文件：

name	description	address	type	length	scale
Temperature	Tank temperature	D100	real32	-1	10
Pressure	Line pressure	D102	int16	-1	1
MotorRun	Motor running status	M100	boolean	-1	1
BatchName	Current batch name	D200	string	10	1

在此示例中，第一个任务的标签名为“plc.line1.Temperature”、“plc.line1.Pressure”等。Temperature 标签的 scale 为 10，意味着从 D100 读取的原始值在发布前乘以 10。向 Temperature 发送写入时，值在写入 PLC 前除以 10。

Yokogawa Memobus

DX Memobus Master (dxmemobus) 通过 TCP 与安川 Memobus 设备读写数据。

关于 servers、project、模块 ID、trigger、update、data、存储转发和运行时等通用 DX 配置，请参阅 DX 驱动通用配置指南。

配置

下面是一个典型配置文件 dxmemobus.json 的示例。

```
{
  "servers": [{"url": "127.0.0.1:1883"}],
  "project": "DEM0",
  "id": "memobus",
  "meta": {
    "name": "dxmemobus"
  },
  "task": [
    {
      "enable": true,
      "id": "plc1",
      "url": "192.168.1.10:9999",
      "trigger": 1000,
      "timeout": 3000,
      "cpufrom": 1,
      "cputo": 2,
      "dataFormat": "ABCD",
      "readOnly": false,
      "data": [
        { "$csv": "dxmemobus.csv" }
      ]
    }
  ]
}
```

连接

驱动通过 TCP 连接安川 PLC。必须在 url 字段提供 IP 地址及端口，例如“192.168.1.10:9999”。省略端口时使用 502。在 DX Edit 中，主机和端口有单独框，新任务开始于“127.0.0.1:502”。

任务参数

timeout

连接和接收超时（毫秒）。默认 3000 ms。如果 PLC 在此时间内未响应，请求视为失败。

cpufrom

请求站（主站）的 CPU 号。这是 Memobus 帧中标识源 CPU 的字节值。默认 0。

cputo

目标站（PLC）的 CPU 号。这是 Memobus 帧中标识目标 CPU 的字节值。默认 0。

dataFormat

控制多寄存器数据类型（real32、int32、uint32、int64、uint64、real64）的字节序。可用选项如下：

Value	Byte Order
ABCD	Big-endian (default)
BADC	Big-endian byte swap
CDAB	Little-endian byte swap
DCBA	Little-endian

如果 32 位或 64 位值显示乱码或不正确，尝试不同字节序以匹配 PLC 格式。

readOnly

设为 true 时，驱动只从设备读取，所有写操作禁用。默认 false。

traceLog

设为 true 时，将此任务每个原始发送/接收 MEMOBUS 帧记录到状态日志。默认 false。正常运行时关闭——每个轮询周期都记录帧，会很快填满状态日志。仅在需要诊断该设备流量时为单个任务开启。

CSV

CSV 文件定义从 Memobus 设备读取或写入设备的数据点列表。列如下：

name	description	address	type	length	default	scale
tag1	Merker register 1	MB1000	bool	-1	1	1

- **disable**——设为“true”可跳过此条目，设为“false”则启用。

- **name**——此数据点的标签名。
- **description**——此数据点的描述。
- **address**——Memobus 寄存器地址（见下文地址格式）。
- **type**——数据类型。支持类型包括 boolean、int16、uint16、int32、uint32、real32、int64、uint64、real64 和 string。也可使用别名：bool、float（表示 real32）和 double（表示 real64）。
- **length**——数据长度。使用 -1 表示默认长度。对于字符串类型，输入字符数。字符串长度必须大于 0。
- **default**——标签的默认值。
- **scale**——读取时应用于原始值的缩放因子，写入时反向应用。例如 scale 为 10 表示读取时原始寄存器值乘以 10。

地址格式

地址遵循安川 Memobus 寄存器命名约定。常见地址前缀如下：

Prefix	Description	Example
M	Merker register (word)	M100, M53
MB	Merker bit	MB1000, MB1001
M with dot	Merker bit within a word	M100.2
I	Input register (hexadecimal)	I13A
O	Output register (hexadecimal)	O100
IB	Input bit (hexadecimal)	IB13A0
OB	Output bit (hexadecimal)	OB1001

对于 I、O、IB 和 OB 地址，数字部分以十六进制书写。驱动在发送到 PLC 前自动转换为十进制。例如 I13A 在内部转换为 I314。

对于 M 和 MB 地址，数字部分以十进制书写，按原样使用。

布尔数据点（type 为“bool”或“boolean”）按标签单独读取。所有其他类型批量合并为随机读取请求以提高效率，每批最多 340 个寄存器。

示例

下面是一个完整示例，单个任务轮询安川 PLC：

```
{
  "servers": [{"url": "127.0.0.1:1883"}],
  "project": "DEM0",
  "id": "memobus",
```

```

    "meta": {
      "name": "dxmemobus"
    },
    "task": [
      {
        "enable": true,
        "id": "plc1",
        "url": "192.168.1.10:9999",
        "trigger": 1000,
        "timeout": 3000,
        "cpufrom": 1,
        "cputo": 2,
        "dataFormat": "ABCD",
        "readOnly": false,
        "data": [
          { "$csv": "dxmemobus.csv" }
        ]
      }
    ]
  }
}

```

以及关联的 CSV 文件 dxmemobus.csv:

name	description	address	type	length	default	scale
bit1	Merker bit 1000	MB1000	bool	-1	1	1
bit2	Merker bit 1001	MB1001	bool	-1	1	1
bit3	Merker word 100 bit 2	M100.2	bool	-1	1	1
word1	Merker word 100	M100	int16	-1	1	1
dword1	Merker double word	M53	int32	-1	1	1
float1	Input register	I13A	float	-1	1	1
real1	Merker float	M200	double	-1	1	1
str1	String value	M100	string	5	1	1

在此示例中，标签名为“memobus.plc1.bit1”、“memobus.plc1.bit2”等。

Yokogawa Link TCP

DX Yokogawa Link TCP (dxylt) 通过以太网使用 Link TCP 协议与横河 PLC 读写数据。

关于 servers、project、模块 ID、trigger、update、data、存储转发和运行时等通用 DX 配置，请参阅 DX 驱动通用配置指南。

配置

下面是一个典型配置文件 dxylt.json 的示例。

```
{
  "servers": [{ "url": "127.0.0.1:1883" }],
  "project": "DEM0",
  "id": "ylt",
  "meta": {
    "name": "dxylt"
  },
  "task": [
    {
      "enable": true,
      "id": "plc1",
      "url": "192.168.1.10:12289",
      "trigger": 1000,
      "CpuNumber": 1,
      "timeout": 3000,
      "readOnly": false,
      "data": [
        { "$csv": "dxylt.csv" }
      ]
    }
  ]
}
```

任务参数

url

横河 PLC 的 IP 地址和端口。格式为“ip:port”，例如“192.168.1.10:12289”。未指定时默认端口为 12289。在 DX Edit 中，主机和端口有单独框，新任务开始于“127.0.0.1:12289”。

CpuNumber

目标横河 PLC 的 CPU 号。PLC 有多个 CPU 单元时，标识要与哪个 CPU 模块通信。默认 1。

timeout

连接超时（毫秒）。默认 3000 ms。通过慢速或拥塞网络通信时设置更高值。

readOnly

设为 true 时，驱动只从设备读取，所有写操作禁用。默认 false。

CSV

CSV 文件定义从横河 PLC 读取或写入 PLC 的数据点列表。列如下：

name	address	type	length	default
tag1	M100	boolean	-1	true

- **disable**——设为“true”可跳过此条目，设为“false”则启用。
- **name**——此数据点的标签名。
- **description**——此数据点的描述。
- **address**——横河 PLC 内存地址。使用标准横河地址记号，如 M100、D200 或 W100。对于引用字寄存器内特定位的 boolean 类型，使用“register.bit”格式（如“D100.0”读取寄存器 D100 的位 0）。
- **type**——数据类型。支持类型 boolean、int8、int16、int32、int64、uint16、uint32、uint64、real32、real64 和 string。
- **length**——数据长度。使用 -1 表示默认长度。对于字符串类型，输入要读取的寄存器数。
- **default**——标签的默认值。
- **scale**——读取时应用于原始值的缩放因子。写入时，值在发送到设备前除以缩放因子。留空或设为 1 表示不缩放。

示例

下面是一个完整示例，两个任务轮询同一 PLC 上的不同 CPU：

```
{
  "servers": [{ "url": "127.0.0.1:1883" }],
  "project": "DEM0",
```

```

    "id": "ylt",
    "meta": {
      "name": "dxylt"
    },
    "task": [
      {
        "enable": true,
        "id": "cpu1",
        "url": "192.168.1.10:12289",
        "trigger": 1000,
        "CpuNumber": 1,
        "timeout": 3000,
        "data": [
          { "$csv": "dxylt.csv" }
        ]
      },
      {
        "enable": true,
        "id": "cpu2",
        "url": "192.168.1.10:12289",
        "trigger": 2000,
        "CpuNumber": 2,
        "timeout": 3000,
        "readOnly": true,
        "data": [
          { "$csv": "dxylt.csv" }
        ]
      }
    ]
  }
}
```

以及关联的 CSV:

name	description	address	type	length	default	scale
MW001	Merker word 1	M100	boolean	-1		
DW001	Data word 1	D200	int16	-1	0	1
DW002	Data word 2	D201	real32	-1	0	1
STR001	String register	D300	string	10		

在此示例中，任务 cpu1 和 cpu2 的第一个标签分别为“ylt.cpu1.MW001”和“ylt.cpu2.MW001”。

Panasonic MEWTOCOL

DX Panasonic Mewtocol 驱动（dxmewtocol）通过 TCP 使用 MEWTOCOL 协议与松下 PLC 读写数据。

关于 servers、project、模块 ID、trigger、update、data、存储转发和运行时等通用 DX 配置，请参阅 DX 驱动通用配置指南。

配置

下面是一个典型配置文件 dxmewtocol.json 的示例。

```
{
  "servers": [{ "url": "127.0.0.1:1883" }],
  "project": "DEM0",
  "id": "mew1",
  "meta": {
    "name": "dxmewtocol"
  },
  "task": [
    {
      "enable": true,
      "id": "plc1",
      "url": "192.168.1.10:2000",
      "trigger": 1000,
      "station": 1,
      "timeout": 3000,
      "readOnly": false,
      "data": [
        { "$csv": "dxmewtocol.csv" }
      ]
    }
  ]
}
```

任务参数

url

松下 PLC 的 IP 地址和端口。格式为“IP:port”，例如“192.168.1.10:2000”。省略端口时使用 9094（MEWTOCOL-COM over LAN）。在 DX Edit 中，主机和端口有单独框，新任务开始于“127.0.0.1:9094”。

station

目标松下 PLC 在网络上的站号。默认 1。值必须为 PLC 支持的有效范围内的非负整数（标准 MEWTOCOL 通常为 1 到 99，根据设备可能最多 238）。

timeout

连接超时（毫秒）。默认 3000 ms。如果 PLC 在此期间内未响应，驱动将请求视为失败。

readOnly

设为 true 时，驱动只从 PLC 读取，所有写操作禁用。默认 false。

CSV

CSV 文件定义从松下 PLC 读取或写入 PLC 的数据点列表。列如下：

name	address	type	length	default	scale	addressEx
HR01	D100	int16	-1	1	1	4

- **disable**——设为“true”可跳过此条目，设为“false”则启用。
- **name**——此数据点的标签名。
- **description**——此数据点的描述。
- **address**——MEWTOCOL 寄存器地址。使用松下寻址格式，如 D（数据寄存器）、R（继电器）、X（外部输入）、Y（外部输出）、L（链接继电器）等。对于针对字内特定定位的布尔地址，使用“RegisterAddress.BitIndex”格式（如“D100.0”读取寄存器 D100 的位 0）。
- **type**——数据类型。支持类型 boolean、int16、uint16、int32、uint32、int64、uint64、real32、real64 和 string。也可使用别名：short、ushort、int、uint、long、ulong、float、double、bool 和 str。
- **length**——数据长度。使用 -1 表示默认长度。对于字符串类型，输入要读取的字符数。
- **default**——标签的默认值。
- **scale**——读取时应用于原始值的缩放因子，写入时反向应用。例如 scale 为 10 表示读取值乘以 10，写入值在发送到 PLC 前除以 10。
- **addressEx**——保留用于地址类型扩展。默认 0。

示例

下面是一个完整示例，两个任务轮询不同 PLC：

```
{
  "servers": [{"url": "127.0.0.1:1883"}],
  "project": "DEM0",
  "id": "line1",
  "meta": {
    "name": "dxmewtocol"
  },
  "task": [
    {
      "enable": true,
      "id": "plc1",
      "url": "192.168.1.10:2000",
      "trigger": 1000,
      "station": 1,
      "timeout": 3000,
      "data": [
        { "$csv": "dxmewtocol.csv" }
      ]
    },
    {
      "enable": true,
      "id": "plc2",
      "url": "192.168.1.11:2000",
      "trigger": 2000,
      "station": 1,
      "readOnly": true,
      "timeout": 5000,
      "data": [
        { "$csv": "dxmewtocol.csv" }
      ]
    }
  ]
}
```

以及关联的 CSV，注意同一 CSV 文件在两个不同任务段中链接：

name	description	address	type	length	default	scale	addressEx
D100	Data register 100	D100	int16	-1	0	1	0
D200	Data register 200	D200	real32	-1	0	1	0
D300	Data register 300	D300	int32	-1	0	1	0
RUN	Relay output	R10.0	boolean	-1	false	1	0
STR01	String register	D500	string	10		1	0

在此示例中，任务 `plc1` 和 `plc2` 的第一个标签分别为“`line1.plc1.D100`”和“`line1.plc2.D100`”。

SNMP

DX SNMP 驱动（dxsnmp）使用简单网络管理协议（SNMP）对支持 SNMP 的设备读写数据。它与 SNMP 代理通信，获取和更新管理信息库（MIB）对象值。模块支持 SNMPv1 浏览和 SNMPv2c 读写操作。

关于 servers、project、模块 ID、trigger、update、data、存储转发和运行时等通用 DX 配置，请参阅 DX 驱动通用配置指南。

配置

下面是一个典型配置文件 dxsnmp.json 的示例。

```
{
  "servers": [{ "url": "127.0.0.1:1883" }],
  "project": "DEM0",
  "id": "snmp",
  "meta": {
    "name": "dxsnmp"
  },
  "task": [
    {
      "enable": true,
      "id": "plc1",
      "url": "127.0.0.1:161",
      "community": "public",
      "slot": "1",
      "trigger": [1000],
      "timeout": 3000,
      "data": [
        { "$csv": "dxsnmp.csv" }
      ]
    }
  ]
}
```

url

SNMP 代理地址，格式为“host:port”。主机可以是 IPv4 地址或主机名。省略端口时使用默认 SNMP 端口 161。例如“192.168.1.10”连接到端口 161，“192.168.1.10:161”也连接到端口 161。在 DX Edit 中，主机和端口有单独框，新任务开始于“127.0.0.1:161”。

community

从设备读取时用于认证的 SNMP 共同体字符串。相当于 SNMP 访问的密码。默认值为空字符串。常见读访问值为“public”。写操作时，模块使用“private”共同体字符串。

slot

分配给 SNMP 设备的槽号字符串标识。默认值为“1”。可用于区分同一任务内多个设备或逻辑分组。

timeout

等待 SNMP 代理响应的最长时间（毫秒）。默认 3000 ms。通过慢速或不可靠网络与设备通信时增大此值。较低超时可更快检测无响应设备，但在慢速连接上可能导致误报失败。

自动发现

模块启动时，为每个任务对 SNMP 代理执行自动 MIB 遍历。使用 SNMPv1 浏览标准 MIB 树（1.3.6.1）下所有可用 OID。如果找到 OID 且文件尚不存在，模块在配置文件同目录下生成名为 dxsnmp_{taskId}.all.csv 的 CSV 文件。此文件包含所有发现的 OID 及自动生成的标签名，构建自己的 CSV 数据文件时可作为参考。

CSV

CSV 文件定义要轮询的 SNMP 对象列表。列如下：

name	address	type	length	scale
tag1	1.3.6.1.2.1.1.4.0	string	-1	1

- **disable**——设为“true”可跳过此条目，设为“false”则启用。
- **name**——此对象的标签名。系统中完整标签名遵循“moduleId.taskId.name”命名约定。
- **description**——可选描述，供参考。
- **address**——点号记号的 SNMP 对象标识符（OID）。例如“1.3.6.1.2.1.1.4.0”指 sysContact 对象。
- **type**——标签值的数据类型。这是自由文本字段，驱动在自动发现时根据 SNMP 代理报告的值类型自动填充。仅供参考，不用于转换标签值。

- **length**——标签的数据长度。设为 -1 自动检测长度，适用于大多数情况。
- **default**——无数据时标签的默认值。
- **scale**——应用于标签值的缩放因子。设为 1 表示不缩放。对于数值，SNMP 代理的原始值在发布前乘以此因子。

示例

下面是一个使用 SNMP 监控网络设备的完整示例。

dxsnmp.json:

```
{
  "servers": [{"url": "127.0.0.1:1883"}],
  "project": "PLANT1",
  "id": "snmp",
  "meta": {
    "name": "dxsnmp"
  },
  "task": [
    {
      "enable": true,
      "id": "switch1",
      "url": "192.168.1.50:161",
      "community": "public",
      "slot": "1",
      "trigger": [5000],
      "timeout": 3000,
      "data": [
        { "$csv": "dxsnmp-switch.csv" }
      ]
    },
    {
      "enable": true,
      "id": "router1",
      "url": "192.168.1.1:161",
      "community": "public",
      "slot": "1",
      "trigger": [10000],
      "timeout": 5000,
      "data": [
        { "$csv": "dxsnmp-router.csv" }
      ]
    }
  ]
}
```

dxsnmp-switch.csv:

name	description	address	type	length	scale
sysDescr	System	1.3.6.1.2.1.1.1.0	string	-1	1

	Description				
sysUpTime	System Uptime	1.3.6.1.2.1.1.3.0	uint32	-1	1
ifInOctets1	Port 1 In Bytes	1.3.6.1.2.1.2.2.1.10.1	uint32	-1	1
ifOutOctets1	Port 1 Out Bytes	1.3.6.1.2.1.2.2.1.16.1	uint32	-1	1

dxsnmp-router.csv:

name	description	address	type	length	scale
sysName	Router Name	1.3.6.1.2.1.1.5.0	string	-1	1
sysContact	Contact Info	1.3.6.1.2.1.1.4.0	string	-1	1
cpuLoad	CPU Usage	1.3.6.1.4.1.9.9.109.1.1.1.1.3.1	uint32	-1	1

通过上述配置，模块将：

103. 每 5 秒轮询网络交换机，超时 3 秒。标签名为“snmp.switch1.sysDescr”、“snmp.switch1.sysUpTime”、“snmp.switch1.ifInOctets1”和“snmp.switch1.ifOutOctets1”。
104. 每 10 秒轮询路由器，超时 5 秒。标签名为“snmp.router1.sysName”、“snmp.router1.sysContact”和“snmp.router1.cpuLoad”。
105. 每个标签发布 SNMP 代理返回的值。如果代理未响应或发生错误，标签值设为 null。

ActiveX Bridge

DX ActiveX Bridge（dxactivex）按可配置触发器读取并执行 JScript 文件。CSV 中的每个脚本条目指向一个在 JScript 引擎中运行的.js 文件，可使用简单的 `tag.get()` 和 `tag.set()` API 读写标签值。脚本可从本地文件系统或 URL 加载。

关于 `servers`、`project`、模块 ID、`trigger`、`update`、`data`、存储转发和运行时等通用 DX 配置，请参阅 DX 驱动通用配置指南。

配置

下面是一个典型配置文件 `dxactivex.json` 的示例。

```
{
  "servers": [{ "url": "127.0.0.1:1883" }],
  "project": "DEM0",
  "id": "module01",
  "meta": {
    "name": "dxactivex"
  },
  "task": [
    {
      "enable": true,
      "id": "script01",
      "url": "",
      "trigger": ["1000"],
      "data": [
        { "$csv": "dxactivex.csv" }
      ]
    }
  ]
}
```

url

脚本文件所在的基础路径。可以是本地目录路径或 HTTP URL。留空（默认）时，脚本相对于包含 JSON 配置文件的文件夹加载。设为目录路径时，脚本从该目录加载。设为 HTTP URL 时，脚本在启动时从该 URL 下载。

role

可选字符串，控制发布数据的目标段。留空（默认）时，数据发布到项目名。设为值时，数据发布到指定角色名。此参数可在任务级（JSON）和单个脚本级（CSV）设置。对于该脚本条目，CSV 级 `role` 优先。

CSV

CSV 文件定义要执行的脚本列表。默认 CSV 列如下：

name	trigger	description
myscript	1000	Calculate totals

- **disable**——设为“true”可跳过此条目，设为“false”则启用。
- **name**——脚本文件名（带或不带.js 扩展名）。省略.js 扩展名时自动追加。也用作脚本条目的标识符。
- **trigger**——可选的每个脚本触发器，覆盖任务级触发器。接受与任务级触发器相同的格式（毫秒间隔、cron 表达式或基于标签的触发器）。留空时脚本使用任务级触发器。
- **role**——此特定脚本条目可选的角色覆盖。留空时使用任务级 role。
- **description**——可选描述，供参考。

脚本 API

每个脚本在 JScript 引擎中运行，可访问以下对象：

tag.get(tagName)

返回标签的当前值。以字符串形式传入完整标签名。例如：

```
var val = tag.get("module01.script01.temperature");
```

也可传入标签名数组，一次获取多个值。返回值是需要解析的 JSON 字符串：

```
var result = JSON.parse(tag.get(["tag1", "tag2"]));
```

tag.get()引用的任何标签名自动注册订阅。无需提前声明标签。

tag.set(tagName, value)

设置标签值。以字符串形式传入标签名，第二个参数传入值。例如：

```
tag.set("module01.script01.result", 42);
```

也可传入对象，一次设置多个标签：

```
tag.set({"tag1": 100, "tag2": 200});
```

console

遵循标准浏览器 console API 的日志对象。可用方法如下：

- console.error(msg)——记录错误消息。
- console.warn(msg)——记录警告消息。
- console.info(msg)——记录信息消息。
- console.debug(msg)——记录调试消息。
- console.trace(msg)——记录跟踪消息。
- console.time(label)——启动命名计时器。
- console.timeLog(label)——记录命名计时器的已用时间。
- console.timeEnd(label)——记录已用时间并停止命名计时器。

示例

下面是一个完整示例，运行两个不同触发器和角色的脚本。

dxactivex.json:

```
{
  "servers": [{ "url": "127.0.0.1:1883" }],
  "project": "PLANT1",
  "id": "calc",
  "meta": {
    "name": "dxactivex"
  },
  "task": [
    {
      "enable": true,
      "id": "fast",
      "url": "",
      "trigger": ["1000"],
      "data": [
        { "$csv": "dxactivex.csv" }
      ]
    }
  ]
}
```

dxactivex.csv:

name	trigger	role	description
totalizer	1000		Sum flow readings
hourly_report	0 0 * * * *	engineer	Hourly summary

totalizer.js:

```
var flow1 = tag.get("io.plc1.flow1");
var flow2 = tag.get("io.plc1.flow2");
if (flow1 != null && flow2 != null) {
    tag.set("calc.fast.total_flow", flow1 + flow2);
}
```

hourly_report.js:

```
var total = tag.get("calc.fast.total_flow");
console.info("Hourly total flow: " + total);
tag.set("calc.fast.hourly_total", total);
```

通过上述配置，模块将：

106. 每 1 秒运行“totalizer”脚本。读取两个流量标签，求和后将结果写入“calc.fast.total_flow”。
107. 每小时开始时运行“hourly_report”脚本。读取总流量值，记录日志并写入“calc.fast.hourly_total”。由于 role 设为“engineer”，此脚本的输出发布到“engineer”角色目标。
108. 标签名遵循“moduleId.taskId.tagName”约定，因此此示例中以“calc.fast.”为前缀。

Custom Driver

DX Custom Driver（dxcustdrv）从串口读写数据。它连接 COM 口，逐行读取传入数据，并可在将值作为标签发布前使用前缀、后缀和正则表达式匹配过滤数据。

关于 servers、project、模块 ID、trigger、update、data、存储转发和运行时等通用 DX 配置，请参阅 DX 驱动通用配置指南。

配置

下面是一个典型配置文件 dxcustdrv.json 的示例。

```
{
  "servers": [{ "url": "127.0.0.1:1883" }],
  "project": "DEM0",
  "id": "module01",
  "meta": {
    "name": "dxcustdrv"
  },
  "task": [
    {
      "enable": true,
      "id": "custdrv01",
      "url": "COM1",
      "baudRate": 9600,
      "dataBit": 8,
      "parity": "n",
      "stopBit": 1,
      "trigger": [1000],
      "data": [
        { "$csv": "dxcustdrv.csv" }
      ]
    }
  ]
}
```

url

要连接的串口名，例如“COM1”、“COM2”等。默认值为“COM1”。

baudRate

串口通信波特率（位/秒）。默认值为 9600。应设置为与所连设备配置的波特率匹配。常见值为 1200、2400、4800、9600、19200、38400、57600 和 115200。

dataBit

每字节的数据位数。默认值为 8。有效值范围为 5 到 8。必须与所连设备的数据位设置匹配。

parity

用于错误检测的校验方法。默认值为“n”（none）。接受值：

Value	Parity
“n” or “none”	No parity bit (default)
“o” or “odd”	Odd parity
“e” or “even”	Even parity
“s” or “space”	Space parity
“m” or “mark”	Mark parity

值大小写不敏感。可使用短形式或完整名称。

stopBit

每字节的停止位数。默认值为 1。接受值：

Value	Stop Bits
1	One stop bit (default)
2	Two stop bits
1.5 or 3	One and a half stop bits

任何其他值都将默认使用 1 个停止位。

CSV

CSV 文件定义从串口读取的数据条目列表。列如下：

name	scale
tag1	1

- **disable**——设为“true”可跳过此条目，设为“false”则启用。
- **name**——此条目的标签名。系统中完整标签名遵循“moduleId.taskId.name”命名约定。
- **Bgn**——可选前缀字符串。指定时，只接受以此字符串开头的行。与 **End** 一起使用，通过同时匹配前缀和后缀过滤传入数据。
- **packet**——可选正则表达式模式。指定时，传入行与此正则匹配，只将匹配部分用作标签值。留空时整行用作标签值。

- **End**——可选后缀字符串。指定时，只接受以此字符串结尾的行。与 **Bgn** 一起使用，通过同时匹配前缀和后缀过滤传入数据。
- **EOL**——可选行尾分隔符。指定时，驱动从串口读取时使用此字符串作为行终止符。设为“CRLF”使用回车加换行。留空时使用系统默认行尾。
- **description**——可选描述，供参考。
- **scale**——用于处理原始值的可选缩放表达式。默认值为 1（不缩放）。

同时指定 **Bgn** 和 **End** 时，驱动只接受以 **Bgn** 字符串开头、以 **End** 字符串结尾的行。行不匹配则丢弃。还指定 **packet** 时，正则在前缀和后缀检查后应用，只将匹配部分作为标签值。

Bgn 和 **End** 都未指定时，驱动接受所有传入行。指定 **packet** 时，正则匹配直接应用于行。

示例

下面是一个完整示例，从两个不同配置的串口读取数据。

dxcustdrv.json:

```
{
  "servers": [{ "url": "127.0.0.1:1883" }],
  "project": "PLANT1",
  "id": "serial",
  "meta": {
    "name": "dxcustdrv"
  },
  "task": [
    {
      "enable": true,
      "id": "scale1",
      "url": "COM3",
      "baudRate": 9600,
      "dataBit": 8,
      "parity": "n",
      "stopBit": 1,
      "trigger": [500],
      "data": [
        { "$csv": "dxcustdrv-scale.csv" }
      ]
    },
    {
      "enable": true,
      "id": "sensor1",
      "url": "COM4",
      "baudRate": 115200,
```

```

        "dataBit": 8,
        "parity": "e",
        "stopBit": 1,
        "trigger": [1000],
        "data": [
            { "$csv": "dxcustdrv-sensor.csv" }
        ]
    }
]
}

```

dxcustdrv-scale.csv:

name	Bgn	packet	End	EOL	description	scale
weight	W	\d+\.\d+	kg	CRLF	Weighing scale reading	1

dxcustdrv-sensor.csv:

name	packet	description	scale
temperature	\d+\.\d+	Temperature sensor	1
humidity		Humidity sensor	1

通过上述配置，模块将：

109. 每 500 ms 从 COM3 读取，9600 波特、无校验。对于“weight”标签，只接受以“W”开头、以“kg”结尾的行，正则\d+\.\d+提取数值。标签名为“serial.scale1.weight”。
110. 每 1 秒从 COM4 读取，115200 波特、偶校验。“temperature”标签使用正则从每行提取小数，“humidity”标签按原样捕获整行。标签名为“serial.sensor1.temperature”和“serial.sensor1.humidity”。