目录

1. 产品概述

1.1. 产品简介

USR-M300 是一款高性能可拓展的综合性边缘网关。产品集成了数据的边缘采集、计算、主动上报和数据读写，联动控制，IO 采集和控制 等功能，采集协议包含标准 Modbus 协议和多种常见的 PLC 协议，以及行业专用协议；主动上报采用分组上报方式，自定义 Json 上报模版， 快速实现服务器数据格式的对接。同时产品还具备路由和 VPN 以及图形化编程功能，图形化模块设计边缘计算功能，满足客户自有设计需求。产品支持 TCP/UDP/MQTT(S)/HTTP(S)等协议通信，支持多路连接；支持 Modbus RTU/TCP 和 OPC UA 协议转换等功能，产品更是支持有人云，阿里云和 AWS，华为云等常用平台的快速接入。 产品采用 Linux 内核，主频高达 1.2Ghz；网络采用 WAN/LAN 加 4G 蜂窝的设计，上行传输更加可靠，同时 LAN 口可以外接摄像头等设备，结合本身路由功能即可实现功能应用；硬件上集成了 2 路 DI，2 路 DO 和 2 路 AI 和 2 路 RS485，不仅能实现工业现场控制和采集的需求，还能实现根据各种采集点数据或状态进行联动控制。可以广泛应用在智慧养殖，智慧工厂等多种工业智能化方案中。产品在结构上采用可拓展设计，可以通过拓展不同功能的模块进行组合应用，更好的满足不同场景对于 IO 数量和通信接口的需求。方便快捷，节省成本。

1.2. Node-RED 与产品集成的背景

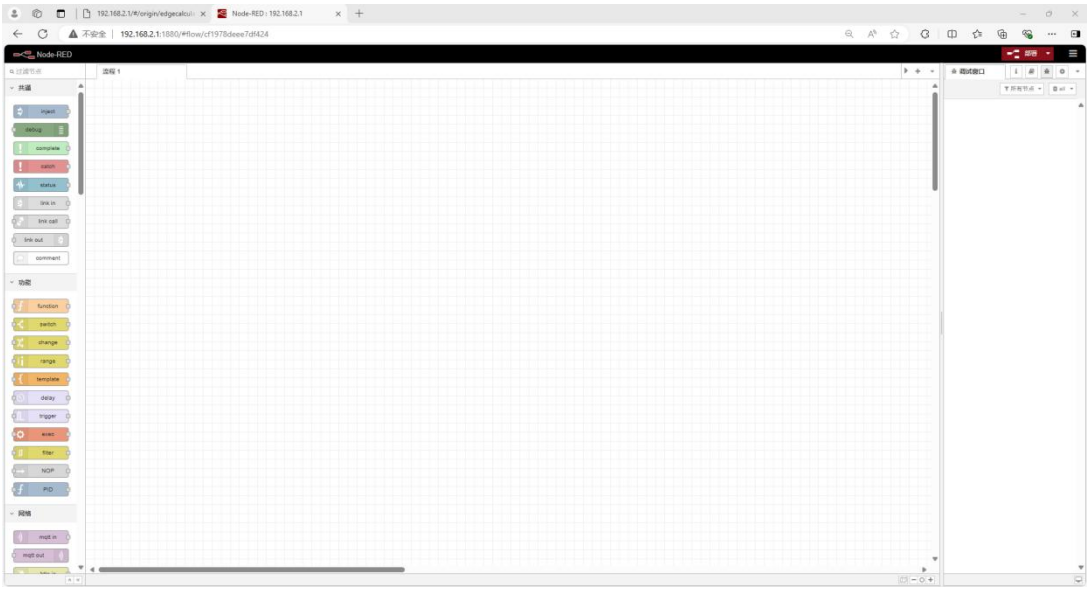
在物联网和自动化领域的不断发展中，Node-RED 成为了一个关键工具，用于连接和管理各种设备和服务。考虑到"M300"产品的特点，我们决定将 Node-RED 集成到"M300"中，以提供更加强大和灵活的解决方案。这个集成将允许您轻松地创建定制的自动化流程，将各种数据源和操作相互连接，从而实现高效、智能的工作流程。

2. Node-RED 基础

2.1. 了解 Node-RED

在"M300"产品中使用 Node-RED 之前，让我们先了解一些关于 Node-RED 的基本信息。Node-RED 是一种编程工具，最初为 IBM 开 发，目前是 OpenJS 基金会的一部分。Node-RED 用于以新的和有趣的方式将硬件设备，API 和在线服务连接在一起。它提供了一个基于浏览器的编辑器，使得使用 Node-RED 中的各种节点轻松地将流连接在一起，只需单击一下即可将其部署到其运行环境。

2.2. Node-RED 编辑器界面：



Node-RED 的开发界面通常分为以下几个主要区域：

1. 工具栏：工具栏位于界面的顶部，包含与流程编辑和部署相关的操作按钮。这些按钮可以用于保存、部署、导入/导出流程，以及设置等

2. 导航面板：导航面板位于左侧、右侧、上侧，包含以下几个部分：

   流程面板：显示当前编辑的流程，你可以在其中创建和切换不同的流程。

   节点库：显示可用节点的列表，包括内置节点和安装的自定义节点。

   信息面板：提供有关所选节点的信息和帮助文档。

3. 编辑器区域：编辑器区域占据了大部分界面空间，用于创建和编辑流程。你可以在此区域中拖拽节点、连接它们以及配置节点的属性。编辑器是一个可视化的工作区。

4. 属性面板：属性面板位于编辑器区域的侧边，用于配置所选节点的属性和设置。当你选择一个节点时，相关的属性和选项将显示在此处，以便你进行自定义设置。

5. 调试区域：调试区域位于底部或侧边，用于查看流程的调试信息。你可以在此处查看节点的输入和输出数据，以帮助诊断问题。

2.3. 基本概念和术语：

当使用 Node-RED 进行流程编程时，有一些基本的概念和术语是非常重要的。以下是一些 Node-RED 的基本概念和术语：

1.流程：流程是由节点组成的工作单元，表示一组相关的操作或任务。在 Node-RED 中，你可以创建多个流程来组织和管理不同的功能。每个流程都有自己的编辑区域。

2.节点：节点是 Node-RED 中的基本构建块，执行各种操作和功能。节点可以是输入节点、处理节点或输出节点。例如，输入节点可以从传感器获取数据，处理节点可以对数据进行处理，输出节点可以将数据发送到其他设备或服务。

3.导线：导线是连接节点之间的线条，表示数据流动的路径。你可以通过拖拽导线来连接不同节点，以定义数据的流向。

4.编辑器：Node-RED 的图形用户界面，用于创建、编辑和部署流程。编辑器包括可视化的工作区，允许你在其中拖拽节点、连接它们，并配置节点的属性。

5.部署：当你完成了流程的创建或编辑后，可以通过点击编辑器中的“部署”按钮来部署流程。部署将使流程开始运行，并将更改保存到

6.节点库：节点库是一个包含各种可用节点的列表。Node-RED 包括内置节点，还支持安装自定义节点。你可以从节点库中拖拽节点到编辑器中，以添加新功能。

7.消息：消息是在 Node-RED 中传递的数据单元。它可以是文本、JSON 对象、二进制数据等。消息包含在导线中流动，并由节点进行处理和转换。

8. 属性：节点具有可配置的属性，用于定义其行为。你可以通过编辑节点的属性来自定义节点的功能和设置。

9.调试：Node-RED 提供了调试工具，用于查看节点的输入和输出数据，以帮助诊断流程中的问题。

10.流程变量：流程变量是在整个流程中共享的数据。你可以使用流程变量来在不同的节点之间传递和存储信息。

11. 全局变量：全局变量是在所有流程中共享的数据。它们在整个 Node-RED 实例中都可访问。

12. 节点状态：节点可以在其状态中存储数据，以便在不同的消息处理之间保持信息。这对于跟踪节点的状态非常有用。

这些基本概念和术语可以帮助你理解 Node-RED 的工作原理，并开始创建自己的自动化流程。Node-RED 的可视化编程方法使其非常 适合物联网、自动化和数据处理任务。

3. 节点介绍

3.1. 通用节点/流介绍

3.1.1. Inject（注入）：

手动或定期得将消息注入流中。消息的有效荷载可以为多种类型，包括字符串，JavaScript 对象或当前时间。

输出：

Payload	various
指定的消息的有效荷载。	
topic	字符串
可以在节点中配置的可选属性。	

详细：

通过使用特定的有效荷载，注入节点可以启动流。默认有效荷载是当前时间的时间戳（以毫秒为单位，自 1970 年 1 月 1 日起）。

该节点还支持注入字符串，数字，布尔值，JavaScript 对象或流/全局上下文值。

默认情况下，节点可以通过在编辑器中单击节点按钮来手动触发。同时也可以被设置为定期或按计划注入。

另一个可选的设置是在每次启动流时注入一次。

可以指定的最大间隔约为 596 小时/24 天。但是，如果对于间隔超过一天的那些间隔，建议您使用 scheduler 节点来应对断电或重启。

注意：选项“时间间隔”和“特定时间”使用了标准 cron 系统。这意味着因此“20 分钟”并不表示在此之后 20 分钟，而是每小时的

20 分钟，40 分钟。如果您希望设定为从现在开始的每 20 分钟，那么请使用“间隔”选项。

注意:如果您想在字符串中包含换行符，必须使用“功能”节点创建有效荷载。

3.1.2. Debug（调试）：

在“调试”侧边栏选项卡和运行时日志中显示选定的消息属性。 默认情况下，它会显示 msg.payload 的值，但您也可以将其设置成显示任意属性，完整消息或 JSONata 表达式的结果。

详细：

调试侧边栏会提供已发消息的结构化视图，方便您查询消息的结构。

JavaScript 对象和数组可以根据需要来折叠或扩展。缓冲区对象可以显示为原始数据，也可以显示为字符串。

对任意条消息，调试侧边栏还会显示接收消息的时间，发送消息的节点以及消息类型等信息。单击源节点 ID 将在工作区中显示该节点。

节点上的按钮可用于启用或禁用其输出。建议禁用或删除所有未使用的调试节点。

还可以通过配置节点，将所有消息发送到运行时的日志，或将简短的数据（32 个字符内）在调试节点下的状态文本上显示。

3.1.3. Complete（监听完成）：

当另一个节点完成对消息的处理时触发流。

详细：

详细介绍：如果一个节点通知运行时它已完成消息的处理，该节点可用于触发第二个流。

这个节点可以与没有输出端口的节点一起使用，例如在使用电子邮件发送节点来发送邮件后触发一个流。

此节点只能被设置为处理流中某个所选节点的事件。与 Catch 节点不同，您不能指定“所有节点”模式并以流中的所有节点为目标。

并非所有节点都会触发此事件。这取决于它们是否支持于 Node-RED1.0 中引入的此功能。

3.1.4. Catch（捕获异常）

捕获由同一标签页上的节点引发的错误。

输出：

error.message	字符串
错误消息	
error.source.id	字符串
引发错误的节点的 ID	
error.source.type	字符串
引发错误的节点的类型	
error.source.name	字符串
引发错误的节点的名称（如果已设置）	

详细：

如果节点在处理消息时抛出错误，则流程通常会停止。该节点可用于捕获那些错误并通过专用流程进行处理。

默认情况下，该节点将捕获同一标签页上任何节点抛出的错误。或者，它可以针对特定节点，或配置为仅捕获另一个“目标”捕获节点尚未捕获的错误。

当错误发生时，所有匹配的 catch 节点都会收到错误消息。

如果在子流中发送了错误，则该错误将由子流中的任意捕获节点处理。如果子流中不存在捕获节点，则那错误将被传播到子流实例所在的标签页。

如果消息已经具有 error 属性，则将该 error 复制为_error。

3.1.5. Status（状态变化）

获取在同一标签页上的其他节点的状态消息。

输出：

error.message	字符串
错误消息	
error.source.id	字符串
引发错误的节点的 ID	
error.source.type	字符串
引发错误的节点的类型	
error.source.name	字符串
引发错误的节点的名称（如果已设置）	

详细：

该节点不包含有效荷载。

默认情况下，节点会获取同一工作空间标签页上报告所有节点的状态。可以通过配置来设定目标节点。

3.1.6. link in（虚拟输入）

在流之间创建虚拟连线。

详细：

该节点可以连接到任何标签页上存在的任何 link out 节点。连接后，它们的行为就像连接在一起。

仅当选择链接节点时，才会显示链接节点之间的链接。如果有指向另一个选项卡的链接，则显示一个虚拟节点。单击该虚拟节点将带您到相应的选项卡。

注意：无法创建进入或离开子流的链接。

3.1.7. link out（虚拟输出）

在流之间创建虚拟连线。

详细：

该节点可以连接到任何标签页上存在的任何 link in 节点。连接后，它们的行为就像连接在一起。

仅当选择链接节点时，才会显示链接节点之间的链接。如果有指向另一个选项卡的链接，则显示一个虚拟节点。单击该虚拟节点将带您到相应的选项卡。

注意：无法创建进入或离开子流的链接。

3.1.8. Comment（注释）

可用于向流添加注释的节点。

详细：

编辑面板接受 Markdown 语法。输入的文本将在信息侧面板中显示。

3.2. 功能

3.2.1. Function（函数-JS）

定义对接收到的消息进行处理的 JavaScript 代码（函数的主体）。

输入消息在名为 msg 的 JavaScript 对象中传递。

通常，msg 对象将消息正文保留在 msg.payload 属性中。

该函数一般会返回一个消息对象（或多个消息对象），但也可以为了停止流而什么都不返回。

传送消息

详细:

请参见在线文档来获得更多有关编写函数的信息。

传送消息

要将消息传递到流中的下一个节点，请返回消息或调用 node.send(messages)。

它将返回/send:

单个消息对象—传递给连接到第一个输出的节点

消息对象数组，传递给连接到相应输出的节点

如果数组元素是数组，则将多个消息发送到相应的输出。

无论 return 方法是单个值还是数组元素，如果返回值为 null，则不会发送任何消息。

日志输出和错误处理

使用以下功能输出日志信息和输出错误:

node.log("Log message")

node.warn("Warning")

node.error("Error")

使用 catch 节点可以进行错误处理。 要由 catch 节点处理，请将 msg 作为 node.error 的第二个参数传递:

node.error("Error",msg);

访问节点信息

您可以使用以下属性来在代码中引用节点 ID 和名称:

node.id—节点的 ID

node.name—节点的名称

使用环境变量

环境变量可以通过 env.get("MY_ENV_VAR")来进行访问。

3.2.2. Switch（条件分配）

按属性值来分配消息的传送路线。

详细:

根据接收到的消息评估指定的规则，然后将消息发送到与匹配的规则相对应的输出端口。

可以将节点设置为一旦发现一个匹配的规则，则停止后续的匹配。

对于评估规则，可以使用消息属性，流上下文/全局上下文属性，环境变量和 JSONata 表达式的评估结果。

有四种规则：

1. 值根据配置的属性评估规则
2. 顺序可用于消息序列的规则，例如由“拆分”节点生成的规则
3. JSONata 表达式评估整个消息，如果结果为真，则匹配。
4. 其他上述规则都不匹配时适用

注释

is true/false 与 is null 规则将对类型进行严格的匹配。匹配之前的类型转化不会发生。

is empty 规则与零字节的字符串，数组，缓冲区或没有属性的对象相匹配。与 null 或者 undefined 等不匹配。

处理消息序列

默认情况下，节点不会修改 msg.parts 属性。

可以启用重建消息序列选项来为每条匹配的规则生成新的消息序列。在这种模式下，节点将在发送新序列之前对整个传入序列进行缓存。运行时的设定 nodeMessageBufferMaxLength 可以用来限制可缓存的消息数目。

3.2.3. Change（修改属性）

设置，更改，删除或移动消息，流上下文或全局上下文的属性。

如果指定了多个规则，则将按定义的顺序来应用它们。

详细：

可用的操作有：

设置：设置一个属性。该值可以是多种不同类型，也可以从现有消息或上下文属性中获取。

置换：搜索并替换属性。如果启用了正则表达式，则可以为“replace with”属性指定捕获组，例如$1。在替换过程中，仅当规则完全匹配时才能更改属性类型。

删除：删除一个属性

移动：移动或者重命名一个属性

类型"expression"使用 JSONata 语言。

3.2.4. Range（量程转换）

将数值映射为另一个区间的数值

输入：

payload 数值

有效荷载一定得是一个数值. 否则则会映射失败。

输出：

payload 数值：被映射到新区间的数值。

该节点将线性缩放所接收到的数值。在默认情况下，结果不限于节点中定义的范围。

缩放并限制到目标范围表示结果永远不会超出目标范围内指定的范围。

在目标范围内缩放并折叠表示结果将会被限制（折叠）在目标范围内。

例如，输入 0-10 映射到 0-100。

模式	输入	输出
scale	12	120
limit	12	100
wrap	12	20

3.2.5. Template（消息模板）

根据提供的模板设置属性。

输入：

Msg object

一个 msg 对象，其中包含着用于填充模板的信息。

Template string

由 msg.payload 填充的模板。如果未在编辑面板中配置，则可以将设为 msg 的属性。

Outputs

Msg object

由来自传入 msg 的属性来填充已配置的模板后输出的带有属性的 msg。

详细：

默认情况下使用 mustache 格式。如有需要也可以切换其他格式。

例如:

```
Hello {{payload.name}}. Today is {{date}}
```

receives a message containing:

```
{
        date: "Monday",
       payload:
       {
              name: "Fred"
       }
}
```

输出的消息将会是:

```
Hello Fred. Today is Monday
```

也可以使用流上下文或全局上下文中的属性：{{flow.name}}或者{{global.name}}，或者为了持久储存store，可以使用{{flow[store].name}}或{{global[store].name}}。

注意：默认情况下，mustache 将在其替换的值中转义任何非字母数字或 HTML 实体。为了防止这种情况，请使用{{{triple}}}大括号。

3.2.6. Delay(延时)

对通过节点的消息进行延迟发送或限制。

输入：

delay 数值

设置要应用于消息的延迟（以毫秒为单位）。仅当节点配置为允许消息去覆盖配置的默认延迟间隔时，此选项才适用。

Reset

如果接收到的消息将此属性设置为任何值，则将清空该节点保留的所有的未发送消息。

Flush

如果接收到的消息的此属性设置为任何值，则将立即发送该节点保留的所有未发送消息。

详细：

当配置为延迟发送消息时，延迟间隔可以是一个固定值，一个范围内的随机值或为每个消息动态设置。

当配置为对消息进行限制时，它们的传递将分散在配置的时间段内。状态显示队列中当前的消息数。可以选择在中间消息到达时丢弃它们。

速率限制可以应用于所有消息，也可以根据 msg.topic 的值来进行分组。分组时，中间消息将会被自动删除。在每个时间间隔，节点可以释放所有主题的最新消息，或释放下一个主题的最新消息。

3.2.7. Trigger（触发）

触发后，将会发送一条消息。如果被拓展或重置，则可以选择发送第二条消息。

输入：

delay 数值

设置要应用于消息的延迟(以毫秒为单位)。仅当节点配置为允许消息去覆盖设置的的默认延迟间隔时，此选项才适用。

reset

如果收到带有此属性的消息，则将清除当前正在进行的任何超时或重复，且不会触发任何消息。

详细：

该节点可用于在流中创建一个超时。 默认情况下，当它收到一条消息时，它将发送一条带有 1 的有效荷载的消息。然后它将等待 250 毫秒，再发送第二条消息，其有效荷载为 0。这可以用于使连接到 Raspberry Pi GPIO 引脚的 LED 闪烁等例子上。

可以将发送的每个消息的有效荷载配置为各种值，包括不发送任何内容的选项。例如，将初始消息设置为 nothing，然后选择将计时器与每个收到的消息一起扩展的选项，则该节点将充当看门狗计时器；仅在设置的间隔内未收到任何消息时才发送消息。

如果设置为字符串类型，则该节点支持 mustache 模板语法。

如果节点中启用了该选项，则可以通过 msg.delay 覆盖发送消息之间的延迟。该值必须以毫秒为单位。

如果节点收到具有 reset 属性或与节点中配置的匹配的有效荷载的消息，则将清除当前正在进行的任何超时或重复，并且不会触发任何消息。

可以将节点配置为以固定的时间间隔重新发送消息，直到被收到的消息重置为止。

（可选）可以将节点配置为将带有 msg.topic 的消息视为独立的流。

3.2.8. Filter(筛选)

异常报告（RBE）节点 - 仅在有效负载发生更改时才传递数据。它还可以阻止除非，或者如果值更改了指定量（死区和窄带模式）则忽略。

输入：

Payload number | string | (object)

RBE 模式将接受数字、字符串和简单对象。其他模式必须提供可解析的数字。

Topic string

如果指定，该功能将在每个主题的基础上工作。该属性可以通过配置来设置。

Reset any

如果设置清除指定 msg.topic 的存储值，如果未指定 msg.topic，则清除所有主题。

输出：

Payload asper input:如果触发，输出将与输入相同。

详细：

在 RBE 模式下，此节点将阻塞，直到 ，（或选定的属性）值与前一个不同。如果需要，它可以忽略初始值，以免在 start.msg.payload 发送任何内容。

死区模式将阻止输入值，除非它的变化大于或大于与前一个值相差的±带隙。

窄带模式将阻止输入值，如果其变化大于或大于等于与前一个值相差的带隙。例如，它对于忽略来自故障传感器的异常值很有用。

在死区和窄带模式下，传入的值必须包含一个可解析的数字，并且都支持%-仅在/除非输入的差异超过原始值的 x% 时才发送。

死区和窄带都允许与之前的有效输出值进行比较，从而忽略任何超出范围的值，或者与之前的输入值进行比较，这会重置设定点，从而允许逐渐漂移（死区）或阶跃变化（窄带）。

注意：这在每个基础上都有效，但如果需要，可以将其更改为另一个属性。这意味着单个 rbe 节点可以同时处理多个不同的主题。

msg.topic

3.3. 网络

3.3.1. mqtt in（MQTT 订阅）

连接到 MQTT 代理并订阅来自指定主题的消息。

输出：

Payload 字符串 | buffer：如果不是二进制 buffer 的话就是字符串

topic 字符串：MQTT 主题，使用/作为层次结构分隔符。

Qos 数值：QoS 服务质量：0，最多一次； 1，最少一次； 2，只一次。

Retain 布尔值

值为 true 时表示消息已保留且可能是旧的。

详细：

订阅主题可以包括 MQTT 通配符（+：一个级别，#：多个级别）。

使用该节点您首先需要建立与 MQTT 代理的连接。通过单击铅笔图标来进行配置。

如有需要，几个 MQTT 节点（输入或输出）可以共享相同的代理连接。

3.3.2. mqtt out（MQTT 发布）

连接到 MQTT 代理并发布消息。

输入：

Payload 字符串 | buffer

要发布的有效负载。如果未设置此属性，则不会发送任何消息。要发送空白消息，请将此属性设置为空字符串。

Topic 字符串：要发布的 MQTT 主题。

Qos number：QoS 服务质量：0，最多一次； 1，最少一次； 2，只一次。默认值为 0。

Retain 布尔值：设置为 true 来将消息保留在代理上。默认值为 false。

详细：

msg.payload 用作已发布消息的有效载荷。如果包含 Object，则会在发送之前将其转换为 JSON 字符串。如果它包含二进制 buffer，则消息将按原样发布。

可以在节点中配置所使用的主题，或者如果留为空白，则可以通过 msg.topic 进行设置。

同样，可以在节点中配置 QoS 和保留值，或者如果保留空白，则分别由 msg.qos 和 msg.retain 设置。要清除先前存储在代理中的主题，

请设置保留标志并向该主题发布空消息。

该节点需要与要配置的 MQTT 代理的连接。通过单击铅笔图标进行配置。

如果需要，几个 MQTT 节点（输入或输出）可以共享相同的代理连接。

3.3.3. http in（http 请求）

创建用于创建 Web 服务的 HTTP 端点。

输出：

Payload：GET 请求包含任何查询字符串参数的对象。或者包含 HTTP 请求正文。

Req object：HTTP 请求对象。该对象包含有关请求信息的多个属性。

Body：传入请求的正文。格式将取决于请求。

Headers：包含 HTTP 请求标头的对象。

Query：包含任何查询字符串参数的对象。

Params：包含任何路由参数的对象。

Cookies：包含请求 cookie 的对象。

Files：如果节点启用了文件上传，则为包含了上传的文件的对象。

Res object：HTTP 响应对象。此属性不应直接使用；HTTP Response 节点记录了如何响应请求。该属性必须保留在传递给响应节点的消息上。

详细：

节点将在配置的路径上监听特定类型的请求。路径可以完全指定，例如/user，或包括可以接受任何值的命名参数，例如/user/:name。使用命名参数时，可以在 msg.req.params 下访问其在请求中的实际值。

对于包含正文的请求（例如 POST 或 PUT），请求的内容将作为 msg.payload 提供。

如果可以确定请求的内容类型，则正文将被解析为任何适当的类型。例如，application/json 将被解析为其 JavaScript 对象表示。

注意：该节点不发送对请求的任何响应。该流必须包含 HTTP 响应节点才能完成请求。

3.3.4. http response（http 响应）

将响应发送回从 HTTP 输入节点接收的请求。

输入：

Payload string：响应的正文。

statusCode 数值：如果设置，则用作响应状态代码。默认值：200。

Headers object：如果设置，则提供 HTTP 头以包含在响应中。

Cookies object：如果设置，则可用于设置或删除 cookie。

详细：

还可以在节点本身内设置 statusCode 和 headers。如果在节点内设置了属性，则不能被相应的 message 属性覆盖。

Cookie 处理：cookies 属性必须是名称/值对的对象。该值可以是使用默认选项设置 cookie 值的字符串，也可以是 options 对象。

下面的示例设置两个 cookie-一个名为 name 的值为 nick，另一个名为 session 的值为 1234，并且有效期设置为 15 分钟。

```
msg.cookies = {
                    name: 'nick',
                    session: {
                                value: '1234',
                                maxAge: 900000
                                }
                }
```

有效选项包括：

Domain：(字符串) Cookie 的域名

Expires：(日期) GMT 标准时间的到期日。如果未指定或设置为 0，则创建会话 cookie

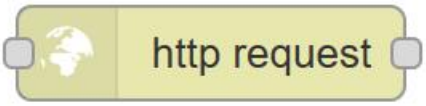
maxAge：(字符串) 相对于当前时间的到期日期（以毫秒为单位）

Path：(字符串) Cookie 的路径。默认为/

Value：(字符串) Cookie 使用的值

要删除 Cookie，请将其 value 设置为 null。

3.3.5. http request



发送 HTTP 请求并返回响应。

输入：

Url 字符串

如果未在节点中配置，则此可选属性设置请求的 url。

method 字符串

如果未在节点中配置，则此可选属性设置请求的 HTTP 方法。必须是 GET,PUT,POST,PATCH 或 DELETE 之一。

Headers object：设置请求的 HTTP 头。

Cookies object：如果设置，则可用于发送带有请求的 cookie。

Payload：发送为请求的正文。

rejectUnauthorized：如果设置为 false，则允许对使用自签名证书的 https 站点进行请求。

followRedirects：如果设置为 false，则阻止遵循重定向（HTTP 301）。默认情况下为 true

requestTimeout：如果设置为正数毫秒，将覆盖全局设置的 httpRequestTimeout 参数。

输出：

Payload （字符串 | object | buffer）：响应的正文。可以将节点配置为以字符串形式返回主体，尝试将其解析为 JSON 字符串或将其保留为二进制 buffer。

statusCode 数值：响应的状态码，如果请求无法完成，则返回错误码。

Headersobject：包含响应头的对象。

responseUrl 字符串：如果在处理请求时发生任何重定向，则此属性为最终重定向的 URL。否则则为原始请求的 URL。

responseCookies object：如果响应包含 cookie，则此属性是每个 cookie 的‘名称/值’键值对的对象。

redirectList 数组：如果请求被重定向了一次或多次，则累积的信息将被添加到此属性。“location”是下一个重定向目标。cookie 是从重定向源返回的 cookie。

详细：

在节点内配置后，URL 属性可以包含 mustache 样式标签。 这些标签允许使用传入消息的值来构造 url。例如，如果 url 设置为 example.com/{{{{topic}}}，它将自动插入 msg.topic 的值。使用{{{...}}}可以防止 mustache 转义/ &等字符。

节点可以选择自动将 msg.payload 编码为 GET 请求的查询字符串参数，在这种情况下，msg.payload 必须是一个对象。

注意：如果使用了代理，则应设置 http_proxy=...环境变量并重新启动 Node-RED，或使用“代理配置”。如果设置了代理配置，则配置优先于环境变量。

使用多个 HTTP 请求节点

为了在一个流程中多次使用该节点，必须要注意 msg.headers 属性的处理。通常在第一个节点在响应头中设置此属性，而不期望在下一个节点的请求头中使用此属性。如果节点之间的 msg.headers 属性保持不变，则第二个节点将忽略它。要设置自定义标题，首先应删除 msg.headers 或将其重置为空对象：{}。

Cookie 处理

传递给节点的 cookies 属性必须是‘名称/值’键值对的对象。该值可以是设置 cookie 值的字符串，也可以是具有单个 value 属性的对象。

请求返回的所有 cookie 都将在 responseCookies 属性下传递回去。

内容类型处理

如果 msg.payload 是一个对象，则节点将自动将请求的内容类型设置为 application/json 并对其进行编码。

要将请求编码为表单数据，应将 msg.headers[“content-type”]设置为 application/x-www-form-urlencoded。

文件上传

要执行文件上传，应将 msg.headers["content-type"]设置为 multipart/form-data 和 msg.payload 传递给节点的必须是具有以下结构的对象：

```
{
        "KEY": {
        "value": FILE_CONTENTS,
        "options": {
        "filename": "FILENAME"
                        }
                }
}
```

KEY,FILE_CONTENTS 和 FILENAME 的值应设置为适当的值。

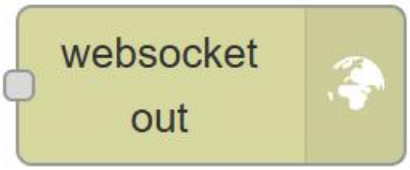
3.3.6. websocket in

WebSocket 输入节点。

默认情况下，从 WebSocket 接收的数据将位于 msg.payload 中。可以将套接字配置为期望格式正确的 JSON 字符串，在这种情况下，

它将解析 JSON 并将结果对象作为整个消息发送。

3.3.7. websocket out



WebSocket 输出节点。

默认情况下，msg.payload 将通过 WebSocket 发送。可以将套接字配置为将整个 msg 对象编码为 JSON 字符串，然后通过 WebSocket 发送。

如果到达此节点的消息是从 WebSocket In 节点开始的，则该消息将发送回触发流程的客户端。否则，消息将广播给所有连接的客户端。

如果要广播从“WebSocket 输入”节点开始的消息，则可以应该删除流中的 msg._session 属性。

3.3.8. tcp in

提供 TCP 输入选择。可以连接到远程 TCP 端口，或接受传入连接。

注意：在某些系统上，您可能需要 root 或管理员权限来访问低于 1024 的端口。

3.3.9. tcp out

提供 TCP 输出的选择。可以连接到远程 TCP 端口，接受传入的连接，或回复从 TCP In 节点收到的消息。

仅发送 msg.payload。

如果 msg.payload 是包含二进制数据的 Base64 编码的字符串，则 Base64 解码选项将导致它在发送之前先转换回二进制。

如果不存在 msg._session，则有效负载将发送到所有连接的客户端。

注意：在某些系统上，您可能需要 root 或管理员权限来访问低于 1024 的端口。

3.3.10. tcp request

一个简单的 TCP 请求节点。将 msg.payload 发送到服务器 tcp 端口，并期望得到响应。

连接到服务器，发送“请求”并接收“响应”。 可以从固定数量的字符，与指定字符匹配的字符中选择操作，从第一个答复到达起等待指定的时间，等待数据到达，发送数据并立即取消连接而无需等待答复等操作中进行选择。

响应将在 msg.payload 中作为 buffer 输出，因此您可能需要对它进行.toString()操作。

如果将 tcp 主机或端口留空，则必须使用 msg.host 和 msg.port 属性进行设置。

3.3.11. udp in

UDP 输入节点。在 msg.payload 中生成 Buffer，字符串或 Base64 编码的字符串。支持组播。

在 msg.ip 和 msg.port 中设置接收到的消息的 IP 地址和端口。

注意：在某些系统上，您可能需要 root 或管理员权限才能使用低于 1024 的端口或广播。

3.3.12. udp out

该节点将 msg.payload 发送到指定的 UDP 主机和端口。支持组播。

您也可以使用 msg.ip 和 msg.port 设置目标值，但是静态配置的值具有优先权。

如果选择广播，则将地址设置为本地广播 IP 地址。或者也可以尝试使用全局广播地址 255.255.255.255。

注意：在某些系统上，您可能需要 root 或管理员权限才能使用低于 1024 的端口或广播。

3.4. 序列

3.4.1. Split（拆分）

将一条消息拆分为一系列消息。

输入：

Payload object | 字符串 | 数组 | buffer

节点的行为由 msg.payload 的类型决定:

1.字符串/buffer - 使用指定的字符（默认值：\n），缓冲区序列或固定长度将消息拆分。

2.数组 - 消息被拆分为单个数组元素或固定长度的数组。

3.object - 将为对象的每个键/值对发送一条消息。

输出：

Parts object：此属性包含有关如何将消息与原始消息分开的信息。如果传递给 join 节点，则可以将序列重组为单个消息。该属性具有以下属性：

Id：一组消息的标识符

Index：组中的位置

Count：如果已知组中的邮件总数。请参阅下面的“流媒体模式”

Type：消息的类型-字符串/数组/对象/buffer

Ch：对于字符串或 buffer，用于将消息拆分为字符串或字节数组的数据

Key：对于对象，创建此消息的属性的键。可以将节点配置为也将此值复制到另一个消息属性，例如 msg.topic

Len：使用固定长度值拆分消息时，每段子消息的长度

详细：

在使用 join 节点将序列重新组合为单个消息之前，推荐使用此节点来轻松地创建跨消息序列，执行通用操作的流。

它使用 msg.parts 属性跟踪序列的各个部分。

流媒体模式

该节点还可以用于重排消息流。例如，发送换行符终止命令的串行设备可能会传递一条消息，并在其末尾带有部分命令。 在“流模式”下，此节点将拆分一条消息并发送每个完整的段。如果末尾有部分片段，则该节点将保留该片段，并将其添加到收到的下一条消息之前。

在此模式下运行时，该节点将不会设置 msg.parts.count 属性，因为流中期望的消息数还是未知的。这意味着它不能在自动模式下与 join 节点一起使用。

3.4.2. Join（合并）

将消息序列合并为一条消息.

共有三种模式：

自动模式：与 split 节点配对时，它将自动将已被拆分的消息进行合并。

手动模式：手动地以各种方式合并消息序列。

列聚合模式：对消息列中的所有消息应用表达式以将其简化为单个消息。

输入：

Parts object：使用自动模式时，所有的消息都应包含此属性。split 节点会生成此属性，但也可以手动进行设置。该属性具有以下属性：

Id：消息组的标识符

Index：组中的位置

Count：如果已知组中的邮件总数。请参阅下面的“流媒体模式”

Type：消息的类型-字符串/数组/对象/buffer

Ch：对于字符串或 buffer，用于将消息拆分为字符串或字节数组的数据

Key：对于对象，创建此消息的属性的键。可以将节点配置为也将此值复制到另一个消息属性，例如 msg.topic/li>

Len：使用固定长度值拆分消息时，每段子消息的长度

Complete：如果设置，则节点将以其当前状态发送其输出消息。

详细：

自动模式：

自动模式使用传入消息的 parts 属性来确定应如何连接序列。这使它可以自动逆转 split 节点的操作。

手动模式：

设置为以手动模式时，该节点能以各种不同的方法来处理消息：

1.字符串或缓冲区-通过将每条消息的选定属性与指定的连接字符或缓冲区连接起来。

2.数组 - 通过将每个选定的属性或整个消息添加到输出数组

3.键/值对象 - 通过使用每个消息的属性来确定存储所需值的键。

4.merged object - 通过将每个消息的属性合并到一个对象下。

输出消息的其他属性都取自发送结果前的最后一条消息。

可以用计数来确定应接收多少条消息来进行合并。对于对象输出，可以设置为达到此计数后的每条后续消息都发送一条输出。

可以用超时来设置发送新消息之前的等待时间。

如果收到设置了 msg.complete 属性的消息时发送输出消息并重置消息列数。

如果收到设置了 msg.reset 属性的消息，则部分收到的消息将被删除而不发送，同时重置消息列数。

列聚合模式：

选择列聚合模式时，将表达式应用于组成消息列的每条消息，并使用聚合值组成一条消息。

初始值：累积值的初始值($A)。

聚合表达式：序列中的每个消息调用的 JSONata 表达式。结果将作为累加值传递到表达式的下一个调用。在表达式中，可以使用以下特殊变量：

$A: 累计值

$I: 消息在序列中的索引

$N: 序列中的消息数

最终调整式子

可选的 JSONata 表达式，在将聚合表达式应用于序列中的所有消息之后应用。在表达式中，可以使用以下特殊变量：

$A: 累计值

$N: 消息在序列中的索引

默认情况下，按顺序从序列的第一条消息到最后一条消息应用聚合表达式。也可以选择以相反的顺序应用聚合表达式。

例子：

给定一系列数字值，以下设置将计算平均值：

1.聚合表达式: $A+payload

2.初始值: 0

3.最终调整式: $A/$N

储存讯息：该节点将在内部缓存消息，以便跨序列工作。运行时设置 nodeMessageBufferMaxLength 可用于设定缓存的消息数。

3.4.3. Sort（排序）

对消息属性或消息序列进行排序的函数。

当配置为对消息属性进行排序时，节点将对指定消息属性所指向的数组数据进行排序。

当配置为对消息序列排序时，它将对消息重新排序。

排序顺序可以是：

1.升序

2.降序

对于数字，可以通过复选框指定数字顺序。

排序键可以是元素值，也可以是 JSONata 表达式来对属性值进行排序，还可以是 message 属性或 JSONata 表达式来对消息序列进行排序。

在对消息序列进行排序时，排序节点依赖于接收到的消息来设置 msg.parts。拆分节点将生成此属性，但也可以手动创建。它具有以下属性：

1.id - 消息组的标识符

2.index - 组中的位置

3.count - 群组中的邮件总数

注意：在此节点的处理中，消息在内部存储。通过指定要累积的最大消息数，可以防止意外的高内存使用。默认设置是不限制消息数量。

nodeMessageBufferMaxLength 属性在 settings.js 中设置。

3.4.4. Batch（规则）

根据各种规则创建消息序列。

详细：

有三种创建消息序列的模式：

讯息数

将消息分组为给定长度的序列。 overlap（重叠）选项指定在一个序列的末尾应重复多少消息。

时间间隔

对在指定时间间隔内到达的邮件进行分组。如果在该时间间隔内没有消息到达，则该节点可以选择发送空消息。

串联序列

通过串联输入序列来创建消息序列。每条消息必须具有 msg.topic 属性和标识其序列的 msg.parts 属性。该节点配置有 topic 值列表，以标识所连接的顺序序列。

储存讯息

该节点将在内部缓冲消息，以便跨序列工作。运行时设置 nodeMessageBufferMaxLength 可用于限制节点将缓存多少消息。

3.5. 解析

3.5.1. csv

在 CSV 格式的字符串及其 JavaScript 对象表示形式之间进行相互转换。

输入:

Payload object | 数组 | 字符串

JavaScript 对象，数组或 CSV 字符串。

输出:

Payload object | 数组 | 字符串:

1.如果输入是字符串，它将尝试将其解析为 CSV，并为每行创建键/值对的 JavaScript 对象。然后该节点将为每行发送一条消息，或者发送一条包含对象数组的消息。

2.如果输入是 JavaScript 对象，它将尝试构建 CSV 字符串。

3.如果输入是简单值的数组，则将构建单行 CSV 字符串。

4.如果输入是数组数组或对象数组，则会创建多行 CSV 字符串。

详细:

列模板可以包含列名称的有序列表。将 CSV 转换为对象时，列名将用作属性名称。或者也可以从 CSV 的第一行中获取列名称。

转换为 CSV 时，列模板用于标识从对象中提取的属性以及提取的顺序。

如果输入是数组，则列模板仅用于有选择地生成一行列标题。

只要正确设置 parts 属性，该节点就可以接受多部分输入。

如果输出多个消息，则将设置其 parts 属性并形成完整的消息序列。

注意：列模板必须用逗号分隔，即使数据中已有了其他分隔符。

3.5.2. html

使用 CSS 选择器从 msg.payload 中保存的 html 文档中提取元素。

输入:

payload 字符串：从中提取元素的 html 字符串。

select 字符串：如果未在编辑面板中配置，则可以将选择器设置为 msg 的属性。

Output:

payload 数组 | 字符串

结果可以是有效载荷中包含匹配元素的数组的单个消息；也可以是多条消息，每条消息都包含匹配元素。发送多条消息时，需要为消息设置 parts。

详细:

该节点支持 CSS 和 jQuery 选择器的组合。查看 css-select documentation 来获得更多信息。

3.5.3. json

在 JSON 字符串及其 JavaScript 对象表示形式之间相互转换。

输入：

Payload object | 字符串：JavaScript 对象或 JSON 字符串。

Schema object：可选的 JSON Schema 对象用于验证有效负载。在将 msg 发送到下一个节点之前，将删除该属性。

Outputs：

Payload object | 字符串：

1.如果输入是 JSON 字符串，它将尝试将其解析为 JavaScript 对象。

2.如果输入是 JavaScript 对象，它将创建一个 JSON 字符串。并可以选择对此 JSON 字符串进行整形。

schemaError 数组：如果 JSON 模式验证失败，则 catch 节点将具有包含错误数组的 schemaError 属性。

详细：

默认的转换目标是 msg.payload，但是也可以转换消息的其它属性。

您可以将其设置为仅执行特定的转换，而不是自动选择双向转换。例如，即使对 HTTP In 节点的请求未正确设置‘content-type’，也可以使用它来确保 JSON 节点的转换结果是 JavaScript 对象

如果指定了转换为 JSON 字符串，则不会对收到的字符串进行进一步的检查。也就是说，即使指定了格式化选项，它也不会检查字符串是否正确为 JSON 或对 JSON 执行整形。

有关 JSON 模式的更多详细信息，请查阅规范.

3.5.4. xml

在 XML 字符串及其 JavaScript 对象表示形式之间进行相互转换。

输入：

payloadobject | 字符串：JavaScript 对象或 XML 字符串。

Options object：可以将选项传递给内部使用的 XML 转换库。请参见 xml2js 文档 来获取更多信息。

输出：

Payload object | 字符串：

如果输入是字符串，它将尝试将其解析为 XML 并创建一个 JavaScript 对象。

如果输入是 JavaScript 对象，它将尝试构建 XML 字符串。

详细：

在 XML 和对象之间进行转换时，默认情况下 XML 属性会添加到名为$的属性中。将文本内容添加到名为_的属性中。这些属性名称可以在节点设置中更改。

例如，将如下所示转换以下 XML：

```
<p class="tag">Hello World</p>
{
    "p": {
        "$": {
            "class": "tag"
        },
        "_": "Hello World"
    }
}
```

3.5.5. yaml

在 YAML 格式的字符串及其 JavaScript 对象表示形式之间相互转换。

输入：

Payload object | 字符串：JavaScript 对象或 YAML 字符串。

Outputs：Payload object | 字符串

1.如果输入是 YAML 字符串，它将尝试将其解析为 JavaScript 对象。

2.如果输入是 JavaScript 对象，它将创建一个 YAML 字符串。

3.6. 存储

3.6.1. write File（写文件）

将 msg.payload 写入文件，添加到末尾或替换现有内容。或者，它也可以删除文件。

输入：

Filename 字符串：如果未在节点中配置，则此可选属性可以设置文件名。

输出：

写入完成后，输入消息将发送到输出端口。

详细：

每个消息的有效荷载将添加到文件的末尾，可以选择在每个消息之间添加一个换行符（\n）。

如果使用 msg.filename，则每次写入后文件都会关闭。为了获得最佳体验，请使用固定的文件名。

可以将其配置为覆盖整个文件，而不是在文件后添加段落。例如，在将二进制数据写入文件（例如图像）时，应使用此选项，并且应禁用添加换行符的选项。

可以从编码列表中指定写入文件的数据的编码。

您可以将此节点配置为删除文件。

3.6.2. read file（读文件）

以字符串或二进制缓冲区的形式读取文件的内容。

输入：

Filename 字符串：如果未在节点配置中设置，该属性可以选择要读取的文件名。

输出：

Payload 字符串 | buffer：文件的内容可以是字符串，也可以是二进制的 buffer。

Filename 字符串：如果未在节点配置中设置，该属性可以选择要读取的文件名。

详细：

文件名应该是绝对路径，否则将相对于 Node-RED 进程的工作目录。

在 Windows 上，可能需要使用转义路径分隔符，例如：\\Users\\myUser。

可以选择将文本文件拆分为几行，每行输出一条消息，或者将二进制文件拆分为较小的 buffer 块-块大小取决于操作系统，但通常为 64k（Linux/Mac）或 41k（Windows）。

当拆分为多条消息时，每条消息将具有 parts 属性集，从而形成完整的消息序列。

如果输出格式为字符串，则可以从编码列表中指定输入数据的编码。

应该使用 Catch 节点来捕获并处理错误。

3.6.3. Watch（监视文件）

监视目录或文件中的更改。

您可以输入用逗号分隔的目录和/或文件的列表。您需要在所有带有空格的地方加上引号“...”。

在 Windows 上，必须在任何目录名称中使用双反斜杠\\。

实际更改的文件的完整文件名将放入 msg.payload 和 msg.filename 中，而监视列表的字符串化版本将在 msg.topic 中返回。

msg.file 仅包含已更改文件的短文件名。msg.type 更改了事物的类型，通常是 file 或 directory，而 msg.size 保留了文件的大小（以字节为单位）。

当然，在 Linux 中，everything 也是一个文件，因此可以监视

注意：该目录或文件必须存在才能被监视。如果文件或目录被删除，即使重新创建它也可能不再被监视。

3.7. dashboard

3.7.1. Button（按钮）

在用户界面上添加一个按钮。

点击该按钮会生成一条消息，msg.payload 被设置为 Payload 字段。如果没有指定有效载荷，则使用节点 ID。

大小默认为 3 乘 1。

图标可以定义为 Material Design 图标（如 "check"、"close"）或 Font Awesome 图标（如 "fa-fire"），或天气图标。如果你在图标名称中加上'mi-'，你就可以使用全套的谷歌材料图标，例如 'mi-videogame_asset'。

文字和背景的颜色可以被设置。它们也可以通过信息属性来设置，方法是将字段设置为该属性的名称，例如{{background}}。你不需要在属性名称前加上 msg.部分。

标签和图标也可以由消息属性设置，方法是将字段设置为该属性的名称，例如{{topic}}或{{myicon}}。

如果设置为通过模式，到达输入端的消息将像按下按钮一样。输出的有效载荷将按照节点配置中的定义。

传入的主题字段可用于设置输出的 msg.topic 属性。

将 msg.enabled 设置为 false 将禁用该按钮。

如果指定了一个 Class，它将被添加到父卡中。这样你就可以用自定义的 CSS 来设计卡片和里面的元素。类可以在运行时通过设置 msg.className 字符串属性来设置。

3.7.2. Dropdown（下拉框）

在用户界面上添加一个下拉选择框。

可以根据需要添加多个值/标签对。如果没有指定标签，值将被用于两者。

所选项目的配置值将作为 msg.payload 返回。

将 msg.payload 设置为其中一个项目值将预设为下拉菜单中的选择。如果使用多选选项，那么有效载荷应该是一个数值数组。

可选的是，主题字段可用于设置 msg.topic 属性。

选项可以通过输入包含一个数组的 msg.options 进行配置。如果只是文本，那么值将与标签相同，否则您可以通过使用一个 "标签":"值"对的对象来指定两者。

["选择 1", "选择 2", {"选择 3": "3"}]

如果启用了 "允许多选 "输出选项--结果将以数组而不是字符串形式返回。

如果指定了一个 Class，它将被添加到父卡中。这样你就可以用自定义的 CSS 来设计卡片和里面的元素。类可以在运行时通过设置 msg.className 字符串属性来设置。

3.7.3. Switch（开关）

在用户界面上添加一个开关。

开关状态的每一次改变都会产生一个带有指定开和关值的 msg.payload。

开/关颜色和开/关图标是可选字段。如果它们都存在，默认的拨动开关将被相关的图标和它们各自的颜色取代。

开/关图标字段可以是一个 Material Design 图标 (例如'check', 'close') 或一个 Font Awesome 图标（例如'fa-fire'），或者一个 Weather 图标。如果你在图标名称中加上'mi-'，你可以使用全套的谷歌材料图标，例如'mi-videogame_asset'。

在直通模式下，开关状态可由传入的带有指定值的 msg.payload 更新，该值也必须符合指定的类型（数字、字符串等）。当不在直通模式时，图标可以跟踪输出的状态--或输入的 msg.payload，以提供一个闭环反馈。

标签也可以由消息属性设置，方法是将字段设置为该属性的名称，例如{{msg.topic}}。

如果指定了一个主题，它将作为 msg.topic 被添加到输出中。

将 msg.enabled 设置为 false 将禁用开关部件。

如果指定了一个 Class，它将被添加到父卡中。这样你就可以用自定义的 CSS 来设计卡片和里面的元素。类可以在运行时通过设置 msg.className 字符串属性来设置。

3.7.4. Slider（滑块）

在用户界面上添加一个滑块小部件。

用户可以在限制（最小和最大）之间改变其值。每一个值的改变都会产生一个以该值为有效载荷的信息。

通过设置尺寸使高度大于宽度，可以创建一个垂直滑块。

滑块可以通过设置最小值大于最大值来反转，例如，最小 100，最大 0。

如果指定一个主题，它将被添加为 msg.topic。

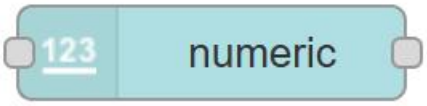
输入的 msg.payload 将被转换为一个数字，并用于预设一个值。如果转换失败，将使用最小值，它将更新用户界面。如果数值发生变化，它也将被传递到输出。

标签也可以由消息属性设置，方法是将字段设置为该属性的名称，例如{{msg.topic}}。

将 msg.enabled 设置为 false 将禁用滑块的输出。

如果指定了一个 Class，它将被添加到父卡中。这样你就可以用自定义的 CSS 来设计卡片和里面的元素。类可以在运行时通过设置 msg.className 字符串属性来设置。

3.7.5. Numeric（数字输入）



在用户界面上添加一个数字输入小部件。

用户可以在限制（最小和最大）之间设置数值。每次数值变化都会产生 msg.payload。

如果指定了 Topic，它将被添加为 msg.topic。

任何输入的 msg.payload 将被转换为数字，如果转换失败，将使用最小值，并且会更新用户界面。如果数值发生变化，也将传递给输出。

Value Format 字段可用于更改显示格式。例如，值为 23 的 {{value}} % 的值格式将在用户界面上显示 23 %。值格式字段可以包含 HTML 或 Angular 过滤器来格式化输出（例如：&deg；将显示度数符号）。

将值格式字段设置为{{msg.payload}}将使输入字段可编辑，因此你可以输入一个数字。

标签也可以由消息属性设置，方法是将该字段设置为该属性的名称，例如{{msg.topic}}。

将 msg.enabled 设置为 false 将禁用小组件的输出。

如果指定了一个 Class，它将被添加到父卡中。这样你就可以用自定义的 CSS 来设计卡片和里面的元素。类可以在运行时通过设置 msg.className 字符串属性来设置。

3.7.6. text input（文字输入）

在用户界面上添加一个文本输入字段。模式可以是普通文本、电子邮件或颜色选择器。

任何输入都会以 msg.payload 的形式发送。如果设置为 "通过模式"，到达的 msg.payload 将被使用，如果它与输入字段的现有文本不同。这允许您预设输入字段的文本。

Delay（默认值：300ms）设置在发送输出前的时间，单位是毫秒。设置为 0 会等待 "Enter "或 "Tab "键的发送。Enter 将发送有效载荷。

但保持在焦点上。Tab 键将发送有效载荷并移动到下一个字段。点击其他地方也会发送有效载荷。

如果不是有效的地址，电子邮件模式将以红色显示，并将返回未定义。

时间输入类型返回从午夜算起的毫秒数。

不是所有的浏览器都支持周和月的输入类型，可能会返回未定义。请在使用前测试你的目标浏览器。

如果指定一个主题，它将被添加为 msg.topic。

将 msg.enabled 设置为 false 将禁用该输入。

如果指定了一个 Class，它将被添加到父卡中。这样你就可以用自定义的 CSS 来设计卡片和里面的元素。类可以在运行时通过设置 msg.className 字符串属性来设置。

3.7.7. date picker（日期选择器）

在用户界面上添加了一个日期选择器小部件。

日期显示可以在仪表板-网站标签中使用 moment.js 格式化。例如 MM/DD/YYYY，Do MMM YYYY 或 YYYY-MM-DD。

将 msg.enabled 设置为 false 将禁用该输入。

如果指定了一个 Class，它将被添加到父卡中。这样你就可以用自定义的 CSS 来设计卡片和里面的元素。类可以在运行时通过设置 msg.className 字符串属性来设置。

3.7.8. colour picker（颜色选择）

在仪表板上添加一个颜色选择器。

如果组的宽度是 4 或更大，那么可以将取色器设置为在任何时候都可见。

格式可以是 rgb、hex、hex8、hsv 或 hsl。除了十六进制之外，所有的格式都支持透明性。

如果指定了一个主题，它将被添加为 msg.topic。

将 msg.enabled 设置为 false 将禁用该输入。

如果设置为 "通过模式"，到达输入口的信息将被评估为任何可用的颜色格式格式。如果转换失败，将使用#000000。

如果指定了一个 Class，它将被添加到父卡中。这样你就可以用自定义的 CSS 来设计卡片和里面的元素。类可以在运行时通过设置 msg.className 字符串属性来设置。

3.7.9. Form（表单）

在用户界面上添加一个表单。

帮助收集用户在点击提交按钮时的多个值作为 msg.payload 中的一个对象

可以使用添加元素按钮添加多个输入元素

每个元素都包含以下组件。

1.Label：在用户界面中作为该元素的标签的值

2.Name：代表 msg.payload 中的键（变量名称），其中有相应元素的值。

3.Type：下拉选项，用于选择元素的类型。下拉选项用于选择输入元素的类型

4.Required：当打开时，用户必须在提交前提供该值

5.Rows：多行文本输入的 UI 行数

6.Delete: 从表单中删除当前元素

可选的是，主题字段可用于设置 msg.topic 属性。

取消按钮可以通过设置它的值为空白""来隐藏。

如果指定了一个 Class，它将被添加到父卡中。这样你就可以用自定义的 CSS 来设计卡片和里面的元素。类可以在运行时通过设置 msg.className 字符串属性来设置。

3.7.10. Text（文字显示）

将在用户界面上显示一个不可编辑的文本字段。

每次收到的 msg.payload 将根据提供的 Value Format 更新文本。

Value Format 字段可用于改变显示的格式，并可包含有效的 HTML 和 Angular 过滤器。

比如说 {{value | uppercase}} °将对 payload 文本进行大写，并添加学位符号。

标签也可以由消息属性设置，方法是将字段设置为该属性的名称，例如{{msg.topic}}。

以下图标字体也可以使用。Material Design 图标 (如'check'、'close') 或 Font Awesome 图标 (如'fa-fire')，或天气图标。如果你在图标名称中加入'mi-'，你可以使用全套的谷歌材料图标。例如，'mi-videogame_asset'。

小组件也有一个 nr-dashboard-widget-{the_widget_label_with_underscores}的类，如果需要的话，可以用来做额外的样式设计。你可能需要使用 !important 标志来重写主题。

如果指定了一个 Class，它将被添加到父卡中。这样你就可以用自定义的 CSS 来设计卡片和里面的元素。类可以在运行时通过设置 msg.className 字符串属性来设置。

3.7.11. Gauge（仪表盘）

在用户界面上添加一个仪表类型的小部件。

msg.payload 会被搜索出一个数值，并按照定义的数值格式进行格式化，然后可以使用 Angular 过滤器进行格式化。

例如：{{value | number:1}}%将把数值四舍五入到小数点后一位，并附加一个%符号。

可以指定 3 个部门的颜色，仪表将在它们之间进行混合。颜色应以十六进制（#rrggbb）格式指定。

如果你指定了扇区的数字，那么每个扇区的颜色就会改变。如果不指定，颜色将在整个范围内混合。

该仪表有几种模式。常规模式、甜甜圈模式、罗盘模式和波浪模式。

标签也可以由消息属性设置，方法是将字段设置为该属性的名称，例如{{msg.topic}}。

如果指定了一个 Class，它将被添加到父卡中。这样你就可以用自定义的 CSS 来设计卡片和里面的元素。类可以在运行时通过设置 msg.className 字符串属性来设置。

3.7.12. Chart（图表）

将输入值绘制在一个图表上。这可以是一个基于时间的折线图，一个条形图（垂直或水平），或一个饼图。

每个输入的 msg.payload 值将被转换为数字。如果转换失败，该信息将被忽略。

最小和最大 Y 轴值是可选的。图形将自动缩放到收到的任何数值。

通过在每个输入消息上使用不同的 msg.topic 值，可以在同一个图表上显示多个系列。通过使用 msg.label 属性可以显示同一系列的多个条形图。

X 轴定义了一个时间窗口或要显示的最大点数。较旧的数据将自动从图表中删除。轴的标签可以使用 Moment.js 的时间格式化字符串进行格式化。

输入一个包含空白数组[]的 msg.payload 将清除图表。

关于如何预先格式化数据以作为一个完整的图表传入，请参见此信息。

空白标签字段可用于在收到任何有效数据之前显示一些文本。

标签也可以由消息属性设置，方法是将该字段设置为该属性的名称，例如{{msg.topic}}。

节点输出包含一个图表状态的数组，如果需要，可以持久化。这可以被传递到图表节点中，重新显示持久化的数据。

如果指定了一个 Class，它将被添加到父卡中。这样你就可以用自定义的 CSS 来设计卡片和里面的元素。类可以在运行时通过设置 msg.className 字符串属性来设置。

3.7.13. audio out（音频输出）

在仪表盘中播放音频或文字转语音（TTS）。

仪表板的网页必须打开，才能发挥作用。

期待 msg.payload 包含一个 wav 或 mp3 文件的缓冲区。

如果您的浏览器支持文字转语音，那么 msg.payload 也可以是一个字符串来朗读。

可选择将 msg.level 从 0 到 100 设置，将音量从 0 到 100%改变。默认是 100%。在音频模式下，您可以提高到 300，但您可能会得到失真。

当 msg.reset 的值为 true 时，那么当前音频片段的播放将被停止。

节点状态反映了当前的播放状态。

1.started: 音频片段的播放已经开始。

2.reset：音频片段的播放已经被重置（即在完成之前停止）。

一旦音频片段播放完成，节点状态将被自动清除。

3.7.14. Notification（对话框）

将 msg.payload 显示为用户界面上的弹出通知或确定/取消对话信息。

如果有 msg.topic，它将被作为标题使用。

如果您没有设置可选的边框高亮颜色，那么它可以由 msg.highlight 动态设置。

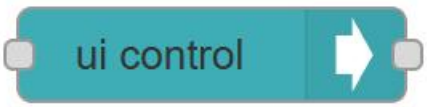
您也可以配置敬酒通知的位置和持续时间。如果您将超时时间留空，则可由 msg.timeout 设置。这不适用于 OK/Cancel 对话框。

对话框会返回一个 msg.payload 字符串，内容是您配置的按钮标签。第二个（取消）按钮是可选的，msg.topic 的返回值也是如此。如果您选择 "确定、取消和输入 "模式，那么 msg.payload 将包含用户输入的任何文本，而不是确定按钮文本。

发送空白有效载荷将删除任何活动的对话框而不发送任何数据。

如果指定了一个类，它将被添加到父元素中。这样，你就可以用自定义的 CSS 为卡片和里面的元素设置样式。

3.7.15. ui control（仪表控制）



允许对仪表板进行动态控制。

msg.payload 应该是一个{"tab": "my_tab_name"}形式的对象，或者只是要显示的标签或链接的标签名称或数字索引（从 0 开始）。

发送一个空白的标签名称""将刷新当前页面。你也可以发送 "+1 "表示下一个标签，"-1 "表示上一个标签。

仪表板页面（即 "标签"）可以通过发送格式为 msg.payload 的对象来控制。

```
{"tabs": {"hide": "tab_name_to_hide", "disable": ["secret_tab", "unused_stuff"]}}
```

有 2 种可用的切换状态：显示/隐藏和启用/禁用

个小组的小工具的可见性可以通过一个有效载荷来控制，例如：

```
{"group": {"hide": ["tab_name_group_name_with_underscores"], "show": ["reveal_another_group"], "focus": true}}
```

其中焦点是可选的，如果需要的话，将导致屏幕滚动以显示该组。你也可以使用属性`open`和`close`来设置可以由用户控制的组的状态。

组名是组的 ID，通常由标签名加上组名组成，用下划线代替所有空格。

当任何浏览器客户端连接或断开连接、更改选项卡或展开或折叠组时，此节点将发出一条消息，其中包含：

1.payload - 连接、丢失、更改或分组。

2.socketid - 套接字的 ID（每次浏览器重新加载页面时这个 ID 都会改变）。

3.socketip - 连接来源的 IP 地址。

4.tab - 标签的编号。(只适用于'改变'事件)。

5.name - 标签的名称。(只适用于'改变'事件)。

6.group - 组的名称。(只适用于 "组 "事件）。

7.open - 组的状态。(仅适用于 "组 "事件）。

可选的--只报告连接事件--用于触发向新客户重新发送数据而不需要过滤掉其他事件，非常有用。

3.7.16. Template（自定义模板）

模板小部件可以包含任何有效的 html 和 Angular/Angular-Material 指令。

此节点可用于创建动态用户界面元素，该元素根据输入消息更改其外观，并可将消息发送回 Node-RED。

例如：

```
<div layout="row" layout-align="space-between">
<p>The number is</p>
<font color="{{((msg.payload || 0) % 2 === 0) ? 'green' : 'red'}}">
{{(msg.payload || 0) % 2 === 0 ? 'even' : 'odd'}}
</font>
</div>
```

将显示作为 msg.payload 接收的数字是偶数还是奇数。如果数字是偶数，它也会将文本的颜色更改为绿色，如果是奇数，它也会将文本的颜色更改为红色。

下一个示例展示了如何为您的模板设置一个唯一的 id，选择默认的主题颜色，并注意任何传入的消息。

```
 <div id="{{'my_'+$id}}" style="{{'color:'+theme.base_color}}">Some text</div>
<script>
(function(scope) {
scope.$watch('msg', function(msg) {
if (msg) {
// Do something when msg arrives
$("#my_"+scope.$id).html(msg.payload);
            }
 });
})(scope);
</script>
```

以这种方式制作的模板可以复制并保持彼此独立。

发送消息：

```
<script>
 var value = "hello world";
// or overwrite value in your callback function ...
this.scope.action = function() { return value; }
</script>
<md-button ng-click="send({payload:action()})">
Click me to send a hello world
</md-button>
```

将显示一个按钮，单击该按钮将发送带有有效负载“Hello world”的消息。

使用 msg.template：

您还可以通过 msg.template 定义模板内容，例如，您可以使用外部文件。

如果模板已更改，将在输入时重新加载模板。

当 msg.template 存在时，模板字段中编写的代码将被忽略。

以下图标字体可用：Material Design 图标（例如“check”、“close”）或 Font Awesome 图标（例如“fa-fire”）或 Weather 图标。如果您在图标名称中添加“mi-”，则可以使用全套 Google 素材图标。例如'mi-videogame_asset'。

如果指定了一个类，它将被添加到父卡中。通过这种方式，您可以使用自定义 CSS 设置卡片及其内部元素的样式。可以通过设置 msg.className 字符串属性在运行时设置 Class。

3.8. M300

3.8.1. get sn（获取设备 SN）

读取 M300 设备的 SN 号，并以字符串格式输出。

输入：

任何类型

输出：

payload 字符串

（正常：02900123021600000815；异常：err）

3.8.2. get mac（获取设备 MAC）

读取 M300 设备的 MAC，并以字符串格式输出。

输入：

任何类型

输出：

payload 字符串

说明：

（正常：D4AD203651BC；异常：err）

3.8.3. get iccid（获取设备 ICCID）

读取 M300 设备的 ICCID，并以字符串格式输出。

输入：

任何类型

输出：

payload 字符串

说明：

（正常：8986032047205253964；异常：err）

3.8.4. get imei（获取设备 IMEI）

读取 M300 设备的 IMEI，并以字符串格式输出。

输入：

任何类型

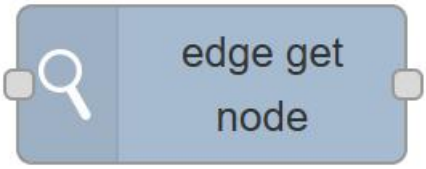
输出：

payload 字符串

说明：

（正常：866859037026521；异常：err）

3.8.5. edge get node（获取变量值）



读取 M300 设备采集点表中某个变量的变量值，并以字符串格式输出。

输入：

任何类型

输出：

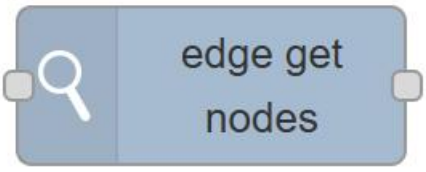
payload 字符串

Topic 数据点名称

说明：

（正常：1；异常：err）

3.8.6. edge get nodes（获取多个变量值）



读取 M300 设备采集点表中多个变量的变量值，并以字符串格式输出。

输入：

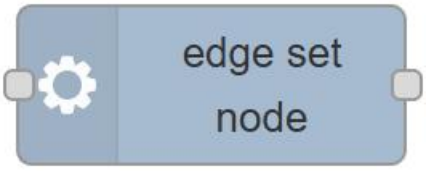
任何类型

输出：

payload 字符串

（正常：1,0,12.3,45；异常：err）

3.8.7. edge set node（设置变量值）



设置 M300 设备采集点表中某个变量的变量值，并以字符串格式输出。

输入：

payload 字符串

输出：

payload 字符串

说明：

（正常：ok；异常：err）

3.8.8. serial config（配置串口参数）

配置 M300 设备某个串口的通讯参数。

输入：

payload 字符串

输出：

payload 字符串

说明：

（正常：ok；异常：err）

3.8.9. serial out（串口发送数据）

通过 M300 设备某个串口发送数据。

输入：

payload 二进制数组

输出：

payload 字符串

说明：

（正常：ok；异常：err）

3.8.10. serial receive（串口接收数据）

通过 M300 设备某个串口接收数据。

输入：

payload 字符串

输出：

payload 字符串

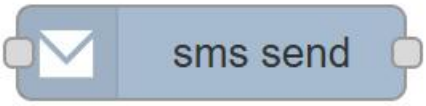
说明：

（正常：二进制数组；异常：err）

注意：

使用时仅需调用一次，无需重复调用

3.8.11. sms send（发送短信）



通过 M300 设备的外置卡发送短信。

输入：

payload 字符串

输出：

payload 字符串

说明：

（正常：ok；异常：err）

注意：

使用时需要插外置卡，且外置卡需要具备发送短信功能。