INTRODUÇÃO

O módulo MQTT implementa suporte para o protocolo MQTT, permitindo integração simples dos controladores com a nuvem.

O protocolo MQTT é baseado em um conceito de PUBLISH/SUBSCRIBE. Neste modelo, equipamentos publicam dados para um servidor centralizado. Equipamentos que estejam interessados nestes dados se subescrevem à eles. Quando um dado é publicado no servidor, todos os que estiverem subscritos receberão o novo dado publicado. A este servidor centralizado se dá o nome de *broker*.

O protocolo MQTT define que os dados são publicados em tópicos. Um tópico é uma string hierárquica que unicamente identifica uma determinada variável. Por exemplo:

```
Predio_X/pavimento_1/sala_101/temperatura_ambiente
```

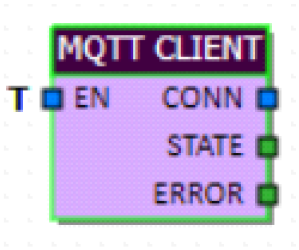
Os tópicos MQTT podem ser livremente selecionados pela aplicação. Ao se subcrever para receber atualizações, um cliente pode usar caracteres especiais para definir que está interessado em todas as publicações de uma determinada hierarquia.

A implementação do protocolo MQTT nos controladores suporta tanto a publicação (publish) de dados quanto o recebimento (subscribe) de dados.

Atualmente o módulo suporta as versões 3.1.1 e 5.0 do protocolo.

BLOCO MQTT_CLIENT

O bloco CLIENT é responsável por gerenciar a conexão do controlador com o servidor MQTT (*broker*). Só pode existir uma única instância deste bloco no programa.



CONTROLE DA CONEXÂO

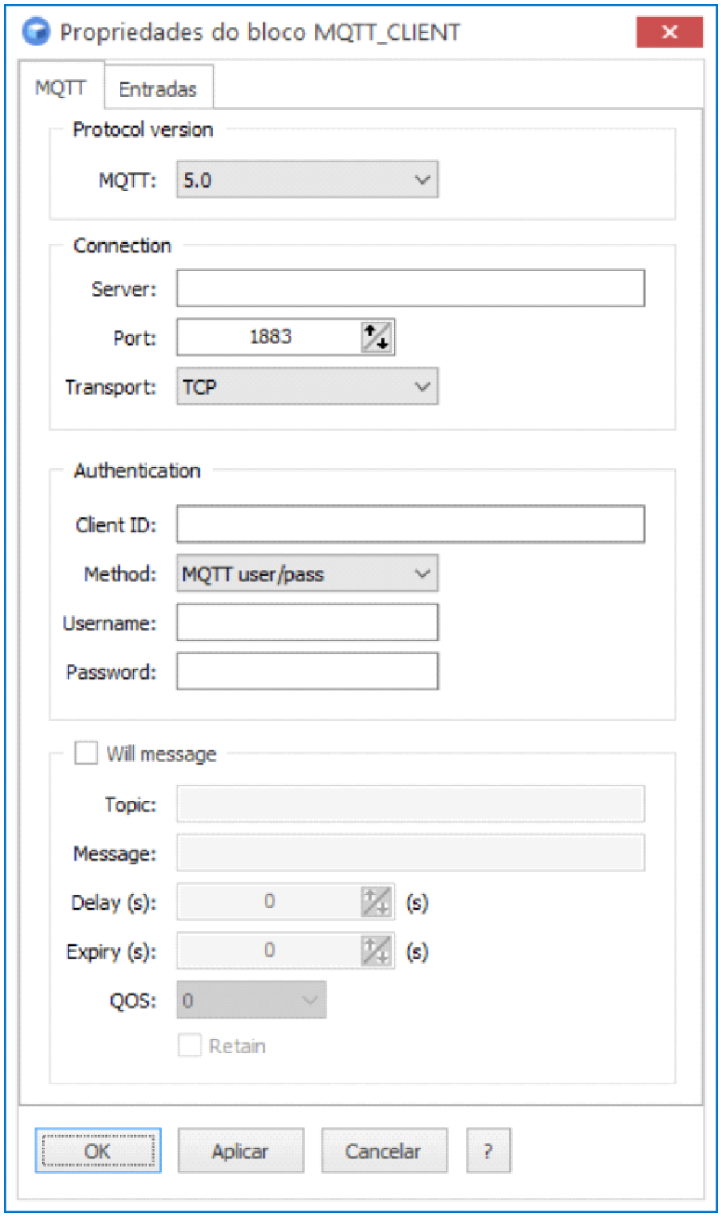
A entrada **EN** controla a conexão do controlador com o *broker*. Com a entrada ativada, o controlador tenta se conectar com o servidor. Mesmo que desconectado pelo *broker*, o bloco repete a tentativa de conexão enquanto a entrada **EN** permanecer ativa.

A saída **CONN** indica o estado da conexão com o *broker*. Os blocos **PUB** e **SUB** só operam caso o controlador esteja conectado com o servidor.

A saída **STATE** indica o estado atual do processo de conexão. Caso ocorra algum erro durante o processo, um código de erro será mostrado na saída **ERROR**. Os códigos destas saídas estão definidos na ajuda do bloco.

CONFIGURAÇÕES

Nas configurações do bloco **MQTT_CLIENT** são definidos os parâmetros do servidor (*broker*).



VERSÃO DO PROTOCOLO

É possível selecionar entre as versões 3.1.1 e 5.0 do protocolo. Recomendamos o uso da versão 5, mas alguns *brokers* ainda não possuem suporte para esta versão. As diferenças de funcionalidade entre as duas versões na implementação do controlador são mínimas. Caso existam, serão destacadas nos tópicos seguintes.

SERVIDOR / TRANSPORTE

Nestas configurações se definem o endereço do *broker* (IP ou host) e a porta TCP com o serviço MQTT. Equipamentos com interface apenas ethernet não suportam resolução de nomes, sendo obrigatório o uso de IP.

A configuração de transporte define a camada de transporte a ser utilizada para o protocolo MQTT. Atualmente são suportados os protocolos TCP (conexão simples, sem criptografia) e TLS (seguros, com criptografia). A versão do protocolo TLS depende do *broker* e recomenda-se o uso de TLS 1.2 ou superior.

AUTENTICAÇÃO

De forma a garantir evolução dos algoritmos de autenticação e segurança cibernética, o protocolo MQTT não define uma forma fixa de autenticação, mas sim apenas um protocolo para que ela seja feita durante a conexão ao servidor. A configuração "*method*" define o algoritmo a ser utilizado.

Atualmente os controladores implementam apenas o método básico de autenticação do protocolo MQTT, através de usuário e senha. Caso o parâmetro usuário ou senha sejam deixados em branco, eles não serão enviados. O significado destes dois parâmetros dependem do *broker* e em alguns casos são usados para outras finalidades. Consultar a documentação do *broker*.

O parâmetro *Client ID* define unicamente o controlador no broker. Caso uma nova conexão for recebida com o mesmo Client ID, a conexão anterior é descartada. Alguns *brokers* suportam que este parâmetro seja uma string vazia, sendo que, neste caso, o próprio *broker* vai definir um ID único para o cliente.

MENSAGEM WILL

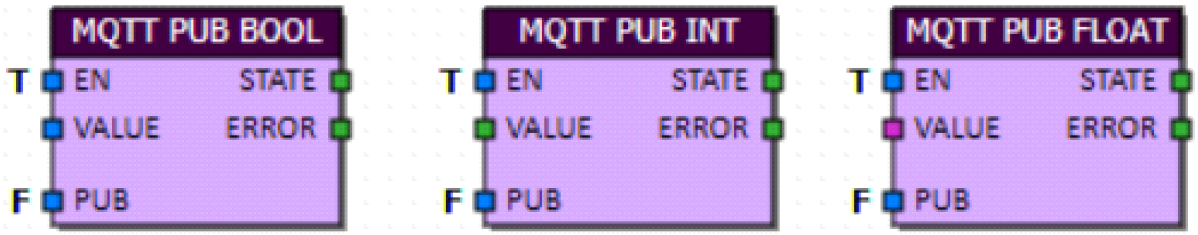
A mensagem de will (testamento) pode ser habilitada caso desejado. Esta mensagem é publicada no tópico selecionado diretamente pelo *broker* caso o controlador se desconecte. Caso a desconexão seja feita formalmente (mensagem DISCONNECT – desativando a entrada EN do bloco CLIENT), a mensagem will não é enviada.

O parâmetro *Delay* define o atraso para envio da mensagem de will. Caso o cliente se conecte novamente antes que este delay expire, a mensagem de will não é enviada.

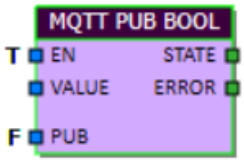
Os demais parâmetros têm o mesmo significado das mensagens publicadas pelo bloco PUB.

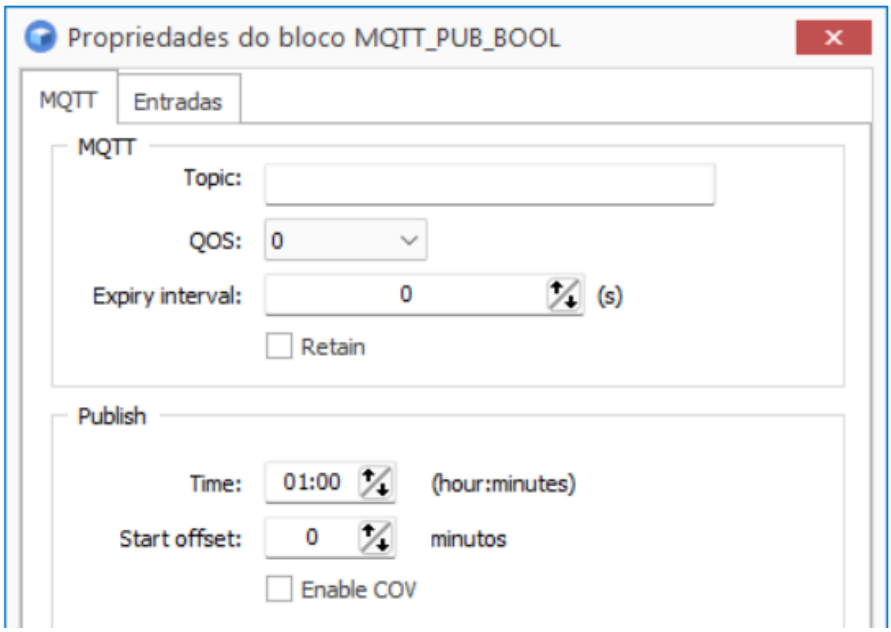
BLOCOS MQTT_PUB BOOL / INT / FLOAT

Os blocos PUB são responsáveis pelo envio de dados ao *broker*.



Este bloco permite fazer a publicação de um valor em formato INTEIRO em um tópico. A publicação pode ser feita manualmente (pela entrada PUB) ou automaticamente por alteração do valor ou sincronizada com o relógio do controlador.





Configurações válidas para os blocos MQTT PUB BOOL/FLOAT/INT

ENTRADAS:

EN: Habilita o bloco. Se desativado, o bloco não faz as publicações.

VALUE: Valor a ser publicado.

PUB: Entrada manual para forçar uma publicação. Publica na borda de subida desta entrada.

SAÍDAS:

STATE: Indica o estado do bloco. Pode assumir os seguintes valores: 0=repouso. 1=aguardando publicação. 2=publicando. 3=erro de publicação (ver ERROR).

ERROR: Indica o último erro ocorrido. Consultar a ajuda do bloco CLIENT para a lista de todos os códigos de erro.

PARÂMETROS:

TOPIC: Configura o tópico para a publicação do valor.

QOS: Define a qualidade do serviço a ser usada para a publicação.

EXPIRY INTERVAL: Define o tempo de validade da mensagem, em segundos.

RETAIN: Define se a mensagem deve ser retida no servidor.

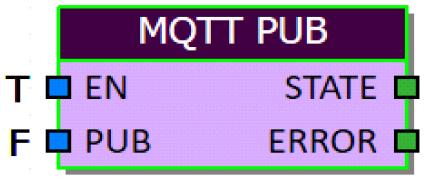
TIME: Configura o intervalo para publicação automática, em horas:minutos. Se diferente de zero, a publicação é feita neste intervalo, sincronizada com o relógio do controlador.

START OFFSET: Define o atraso inicial para publicação sincronizada com o relógio. Se o parâmetro TIME for 01:00 e este offset for 30, a publicação será feita às 01:30, 02:30, etc…

VALUE CHANGE: Se diferente de zero, define a variação mínima do valor de entrada do bloco que gera uma publicação automática.

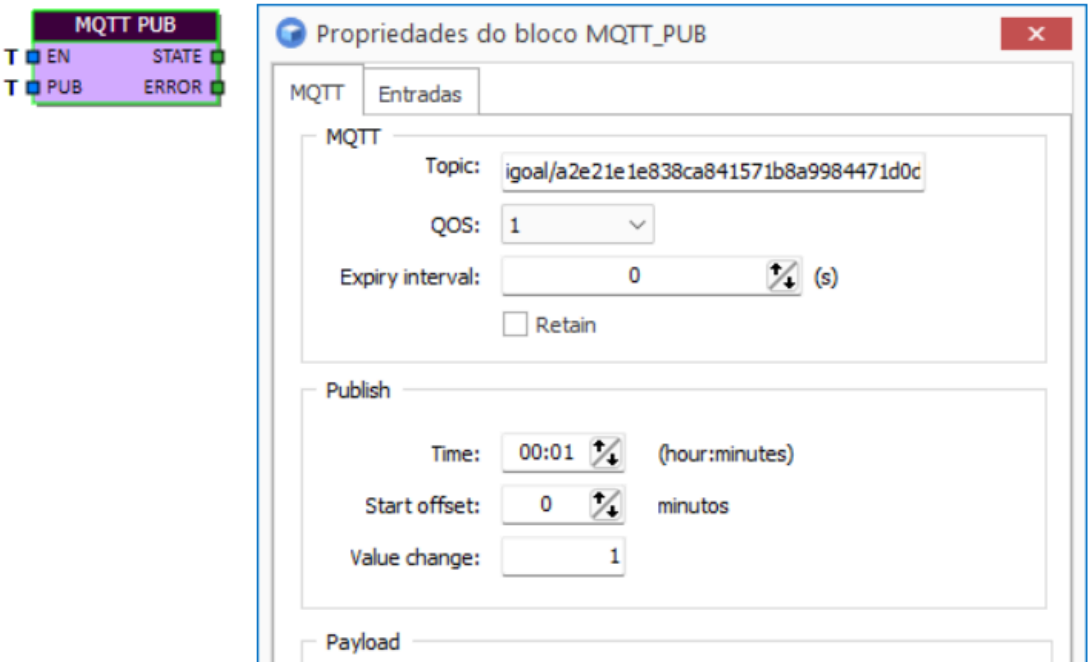
BLOCOS MQTT_PUB (Sincronismo e payload)

Este bloco permite fazer a publicação MQTT com payload configurável. O payload deve ser configurado no módulo TAGs. A publicação pode ser feita manualmente (pela entrada PUB) ou automáticamente sincronizada com o relógio do controlador.



Para cada formato de dado suportado pela linguagem gráfica, existe um bloco PUB correspondente (BOOL, INT e FLOAT). Estes blocos enviam para o broker o valor da entrada em formato texto. Para um melhor controle do formato da mensagem, usar o bloco MQTT_PUB. Este bloco permite selecionar livremente o formato da mensagem (payload). Consultar o manual do módulo TAGS para detalhes.

A entrada **EN** habilita o envio de dados do bloco. A entrada **VALUE** define o valor a ser enviado.



DISPARO

Existem 3 formas para publicação dos dados: manual, tempo e variação de valor. É possível usar qualquer combinação destes disparos.

O disparo **MANUAL** é feito através da entrada **PUB**. Sempre que uma borda de subida for detectada nesta entrada uma publicação é enviada para o servidor.

O disparo por **TEMPO** é feito configurando-se o parâmetro *TIME* nas propriedades do bloco. Este parâmetro indica em HORAS:MINUTOS o intervalo de publicação. Este tempo é sincronizado com o relógio do controlador. O parâmetro *START OFFSET* permite definir o offset inicial em minutos. Por exemplo, se *TIME* for 01:00 e *OFFSET* for 30, o envio será feito às 00:30, 01:30, ... Caso o parâmetro *TIME* seja zero, o disparo por tempo é desabilitado.

O disparo por **VARIAÇÃO DE VALOR** é habilitado quando se configura o parâmetro *VALUE CHANGE* diferente de zero. Se a variação de valor for maior ou igual do que este parâmetro (em relação ao último envio) um novo valor é publicado.

PARÂMETROS DO MQTT

Nas propriedades do bloco é necessária a configuração dos parâmetros para publicação no *broker* MQTT.

O parâmetro *TOPIC* define o tópico a ser publicado. Este valor é enviado pelo protocolo da forma como configurado, sem tratamento.

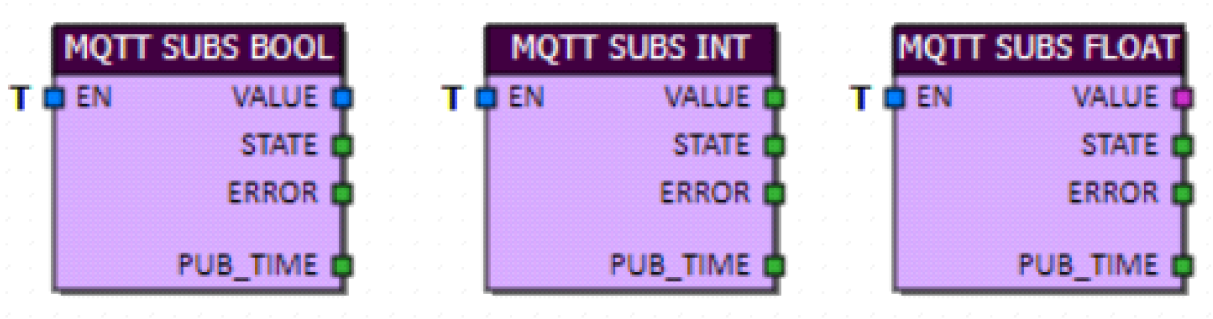
O parâmetro *QOS* indica a qualidade do serviço. Publicações com QOS 0 são enviadas apenas uma vez, sem confirmação do servidor. É indicado para dados que são enviados em intervalos regulares, onde a perda de um dado no processo não é relevante. Publicações com QOS 1 são confirmadas pelo servidor, garantindo a entrega. Neste QOS o *broker* pode receber o mesmo dado mais do que uma vez, devido ao reenvio caso o servidor não confirme o dado em tempo hábil. Publicações com QOS 2 possuem garantia de entrega além do recebimento único pelo *broker*. Recomenda-se que os QOS nível 1 e 2 sejam usados apenas quando necessários pela aplicação, devido à maior complexidade de envio e uso de um número maior de recursos do controlador.

O parâmetro *EXPIRY INTERVAL* define o tempo de validade da mensagem no *broker* em segundos. Quando a mensagem é publicada, seu valor é enviado para todos os clientes que estão subscritos no tópico. Alguns *brokers* suportam o armazenamento destas mensagens para envio quando ocorrerem novas subscrições no tópico ou quando clientes se reconectarem. Nestes casos, o parâmetro *expiry interval* define o tempo que esta mensagem fica armazenada no *broker*.

O parâmetro *RETAIN* define se a mensagem deve ficar armazenada no servidor. Caso *retain* seja false, as mensagens publicadas são enviadas apenas para os clientes subscritos e conectados no momento da publicação. Existem vários *brokers* que não suportam as mensagens armazenadas e a publicação com o parâmetro *retain* pode gerar a desconexão do *broker*.

BLOCOS_MQTT_SUB

Os blocos SUB são usados para o controlador se subscrever em determinado tópico e receber dados do *broker*.



Da mesma forma que os blocos PUB, existem um bloco para cada formato de dado suportado na linguagem gráfica.

Quando conectado ao *broker*, a entrada **EN** controla o estado da subscrição. Quando ativada, o bloco se subscreve no tópico. Quando desativada, o bloco cancela a subscrição. A saída **VALUE** contém o último valor recebido. A validade deste valor depende da saída *STATE*, que indica o estado da subscrição. Caso *state* seja 3, o bloco está subscrito no tópico, mas nenhuma publicação (valor) ainda foi recebido. Quando *state* for 4, a saída *value* é válida.

Caso ocorra algum erro durante o processo de subscrição, a saída **ERROR** indica o código do erro ocorrido. Consultar a ajuda do bloco para detalhes.

A saída **PUB_TIME** indica o tempo decorrido (em segundos) desde a última publicação recebida.

PARÂMETROS DO MQTT

O parâmetro *TOPIC* define o tópico a se subscrever. Este parâmetro é enviado no protocolo como configurado, sem tratamento.

O parâmetro *MAX QOS* define o QOS máximo que pode ser recebido por esta subscrição. Pelas regras do protocolo, publicações com QOS maior que este parâmetro não são enviados pelo *broker*. Na prática, a maioria dos *brokers* suporta alteração do QOS original da publicação para reduzir para o limite da subscrição.

O parâmetro *RETAIN HANDLING* define como devem ser tratadas as mensagens retidas no *broker* para o tópico da subscrição. É possível receber estas mensagens após a subscrição ou não.

FORMATO DE DADOS

O protocolo MQTT não define o formato do dado enviado (*payload*), sendo definido pela aplicação.

Atualmente o controlador envia o *payload* em formato texto. Consulte o suporte para diferentes formatos, se necessário.

PAYLOAD:

Os endereços definidos pela tabela abaixo, estão declarados no arquivo de payload denominado como “Programa BASE rev00” e previamente instalados no TCP IoT, facilitando a comunicação com a plataforma.

As estruturas de payload podem variar para cada composição de plataforma. Detalhes sobre os tipos de payload mais comuns:

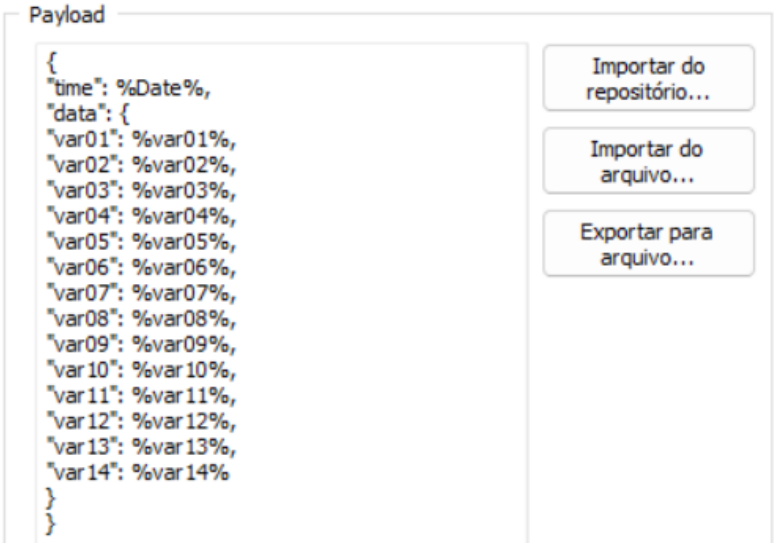
Texto: A forma mais comum de payload, onde a mensagem é um texto em qualquer codificação. Pode ser útil para mensagens simples, comandos ou status de dispositivos.

Dados Binários: Permite enviar dados brutos, como imagens, arquivos ou dados de sensores. É útil quando o conteúdo da mensagem não é texto ou quando se precisa de flexibilidade na manipulação de dados.

JSON: Um formato de dados estruturado e leve, frequentemente utilizado em aplicações web e para a troca de dados entre sistemas. Facilitam a transmissão de dados complexos e a sua leitura por diferentes aplicações.

XML: Outro formato de dados estruturado, similar ao JSON, mas com maior flexibilidade e capacidade de criar estruturas mais complexas. É usado em sistemas mais antigos ou em aplicações que exigem mais flexibilidade na estruturação dos dados.

Abaixo temos um modelo de estrutura JSON utilizada na plataforma IGOAL/TECNOLOG, com uso de TAGs na programação.



BLOCOS TAGS

O módulo TAGS é um módulo genérico para criação de mensagens/payloads configuráveis. O módulo MQTT, por exemplo, usa este módulo para definir um formato customizado de payload.

Estes blocos são criados livremente na lógica para associar um valor ao nome do tag.

TAG_BOOL / INT / FLOAT

Os blocos TAGS permitem definir nomes (tags) para variáveis da lógica. Existe um bloco TAG para cada formato de dado suportado.

O payload possui um formato texto, que pode ser criado livremente para definir mensagens em JSON, XML, etc. Isto permite a integração dos controladores com diversas plataformas de nuvem.

Dentro do payload os valores dos tags previamente definidos são inseridos usando o seguinte formato:

`%NOME%`, onde NOME é o nome definido anteriormente para o TAG.

A maioria dos TAGs suporta parâmetros de formatação, como mostrados no tópico a seguir. Para definir parâmetros, o seguinte formato é usado:

`%NOME[parametros]%.`

Para inclusão de um caractere '%', usar um tag vazio (%%).

FORMATAÇÃO DOS TAGS

Quando ocorre a substituição de um tag, é possível informar alguns parâmetros de formatação (dentro de colchetes). Caso os parâmetros não sejam incluídos, a formatação é feita com os parâmetros padrões, conforme indicado.

Os parâmetros são uma lista no formato:

```
parametro1=valor1, parametro2=valor2, ...
```

TAG BOOL: As tags booleanas permitem os seguintes parâmetros:

true - define um texto para valor true. Padrão é "1".

false - define um texto para o valor false. Padrão é "0".

Exemplo:

```
%TAG_BOOL[true=LIGADO, false=DESLIGADO]%
```

TAG INT: A tag INT atualmente não tem parâmetros.

Exemplo:

```
%TAG_INT%
```

TAG FLOAT: As tags float possuem o seguinte parâmetro:

dec - define o número de decimais. Padrão é 1.

Exemplo:

```
%TAG_FLOAT[dec=3]%
```

TAGS ESPECIAIS

As seguintes tags são definidas no firmware:

DATE: Permite inserir a data do controlador na mensagem. Possui os seguintes parâmetros:

fmt - define o formato da data:

(vazio)	Formato padrão (dd/mm/yyyy)
day	Mostra somente o dia
month	Mostra somente o mês
year	Mostra somente o ano (4 caracteres)
year2	Mostra somente ano (2 caracteres)

Exemplo:

```
%DATE%       -> 12/05/2023
%DATE[fmt=day]     -> 12
```

TIME: Permite inserir a hora do controlador na mensagem. Possui os seguintes parâmetros:

fmt - define o formato da hora:

(vazio)	Formato padrão (hh:mm:ss)
hour	Mostra somente a hora
minute	Mostra somente o minuto
second	Mostra somente o segundo

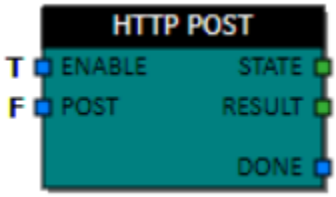
Exemplo:

```
%TIME%       -> 16:15:30
%TIME[fmt=second]  -> 30
```

HTTP_POST

O HTTP POST é usado para enviar dados para o servidor, como dados de um formulário ou dados binários. É comumente usado para criar novos recursos (como registar um novo utilizador) ou atualizar recursos existentes (como editar um perfil).

O comando POST envia os dados no corpo da mensagem, o que permite enviar grandes quantidades de dados e dados binários.



Este bloco permite enviar um comando POST em HTTP para integração com webservices diversos. O conteúdo do POST pode ser livremente configurado através do módulo TAGS.

ENTRADAS

ENABLE: Habilita o bloco. Caso desabilitado, envios são ignorados.

POST: Gera um envio (POST) manual na borda de subida.

SAÍDAS

STATE: Indica o estado atual do envio. 0=IDLE, 1=Aguardando envio, 2=Enviando

RESULT: Indica o código de erro (negativo) ou resposta do servidor remoto (positivo).

DONE: Gera um pulso de um ciclo ao final da operação.

PARÂMETROS

URL: Configura a URL para conexão ao servidor remoto. Deve estar no formato servidor:porta/resource.

Não usar http:// ou https:// no início da URL. [servidor] pode ser um IP ou uma URL, pode haver limitações, dependendo do equipamento/interface selecionada.

[porta] é opcional. Se não indicada, é considerada a padrão 80.

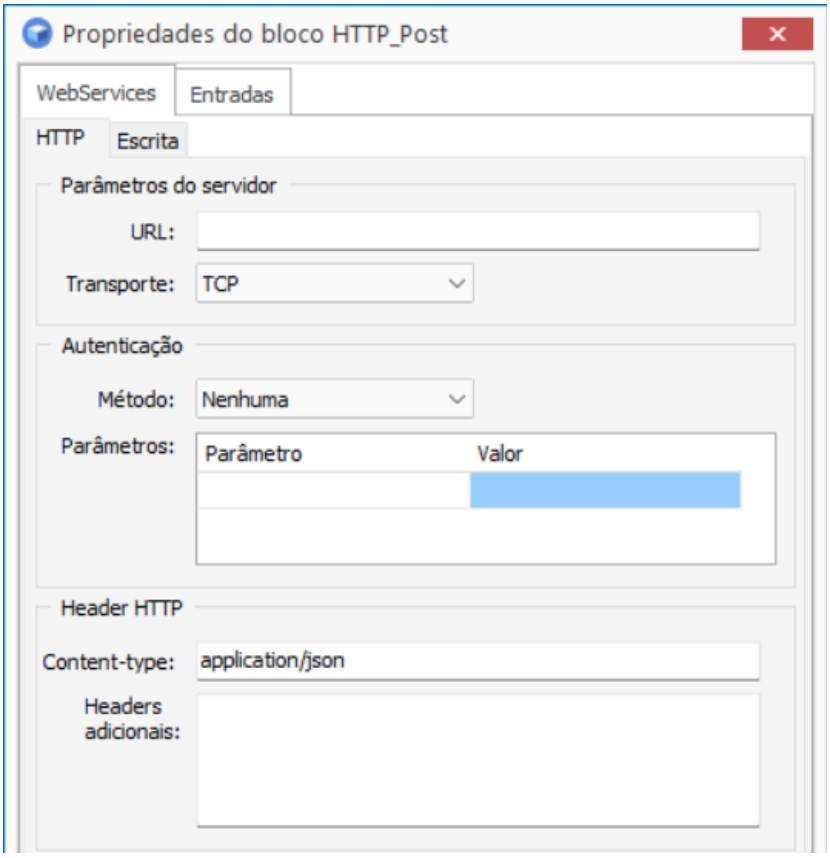
[resource] indica o recurso para POST.

TRANSPORTE: Seleciona o tipo de conexão a ser usada (TCP ou criptografia TLS). Pode haver limitações dependendo do equipamento/interface.

MÉTODO DE AUTENTICAÇÃO: Seleciona o método de autenticação HTTP a ser usado. Conforme o método selecionado, parâmetros serão adicionados à lista abaixo e devem ser preenchidos.

PARÂMETROS DE AUTENTICAÇÃO: Lista com parâmetros necessários para autenticação, conforme o tipo de autenticação selecionado.

CONTENT-TYPE: Define o mime-type do conteúdo do POST. Depende do formato do payload configurado pelo usuário e deve ser ajustado de acordo.



HEADERS ADICIONAIS: Permite a inclusão de linhas adicionais no header HTTP. Incluir um final de linha [ENTER] após cada linha.

O formato deve serguir o padrão do HTTP (nome_header: valor_header).

FORMATO DO PAYLOAD: Seleciona o formato do payload. Novos formatos são criados livremente no módulo TAGS.

TEMPO ESCRITA AUTOMÁTICA: Se diferente de zero, executa o POST automáticamente a cada HORA:MINUTO, sincronizado com o relógio do equipamento.

ATRASO INICIAL: Permite definir um atraso inicial em minutos a partir do primeiro envio.

Por exemplo, se o tempo para escrita automática estiver configurado em 01:00 (uma hora) e o atraso inicial em 30 (minutos), o bloco fará o envio a cada hora nos horários 00:30, 01:30, 02:30 e assim por diante.

PAYLOAD

O payload ou corpo da mensagem, pode ter várias estruturas, dependendo do tipo de dados que você quer enviar. Alguns exemplos comuns incluem:

application/json: O payload é um objeto JSON, ideal para dados estruturados e intercambiáveis.

application/xml: O payload é um documento XML, utilizado quando a necessidade de dados mais complexos e hierárquicos é mais relevante.

application/x-www-form-urlencoded: Uma estrutura mais antiga, utilizada para enviar dados em formato de pares chave-valor, sendo mais comum em formulários web.

multipart/form-data: Utilizado para enviar arquivos, juntamente com outros dados.

text/plain: O payload é um texto simples, utilizado para dados menos complexos, como mensagens ou logs.

raw: O payload pode ser de qualquer formato, como arquivos ou dados binários.

API Key: é um token que identifica a aplicação que está a fazer a chamada. É normalmente passado no cabeçalho HTTP, como API-Key.

Em geral, o tipo de payload deve ser especificado no cabeçalho Content-Type da requisição HTTP.

Exemplo de estrutura Payload para HTTP POST em formato API.