Driver Elipse RESTAPIClient

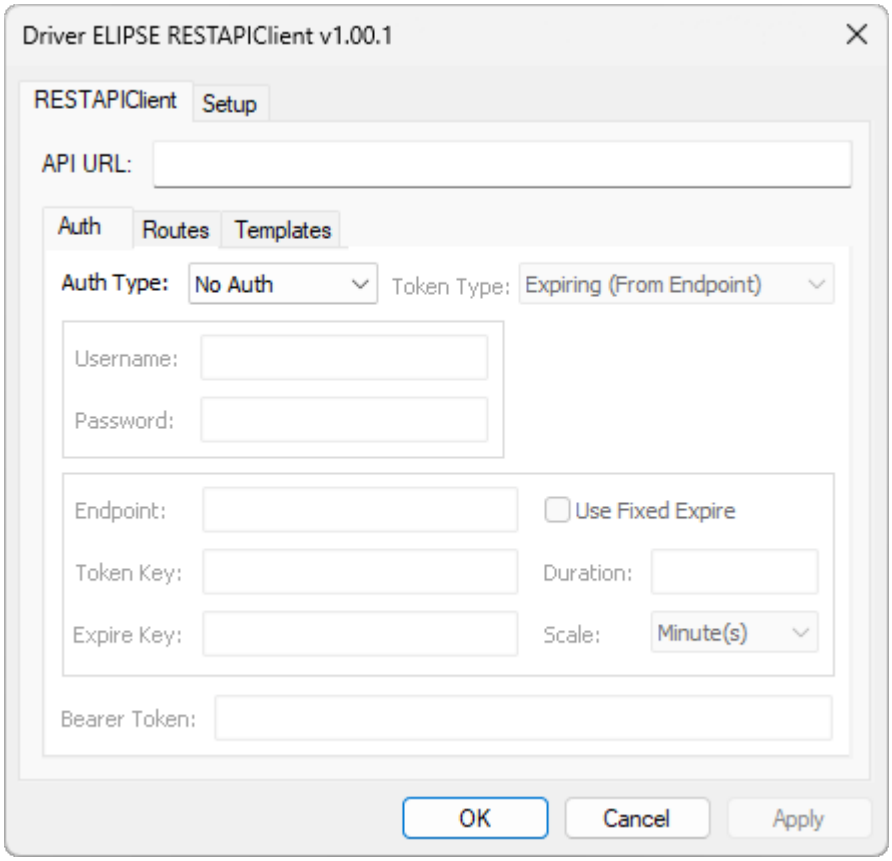
Nome do Arquivo	RESTAPIClient.dll
Fabricante	Elipse Software Ltda
Equipamentos	APIs RESTful
Protocolo	HTTP ou HTTPS
Versão	1.0.4
Última Atualização	28/09/2026
Plataforma	Win32
Dependências	IOKit versão 3.0
Leitura com Superblocos	Não
Nível	0

Introdução

Este Driver implementa o protocolo HTTP ou HTTPS, permitindo que aplicações desenvolvidas pela **Elipse Software** se comuniquem com APIs RESTful, utilizando métodos HTTP padrão e suportando o formato de dados **JSON** (*JavaScript Object Notation*).

Parâmetros de Configuração do Driver

Os parâmetros **[P]** de configuração deste Driver não são utilizados. Todas as configurações são executadas na janela de configurações, mostrada na figura a seguir.

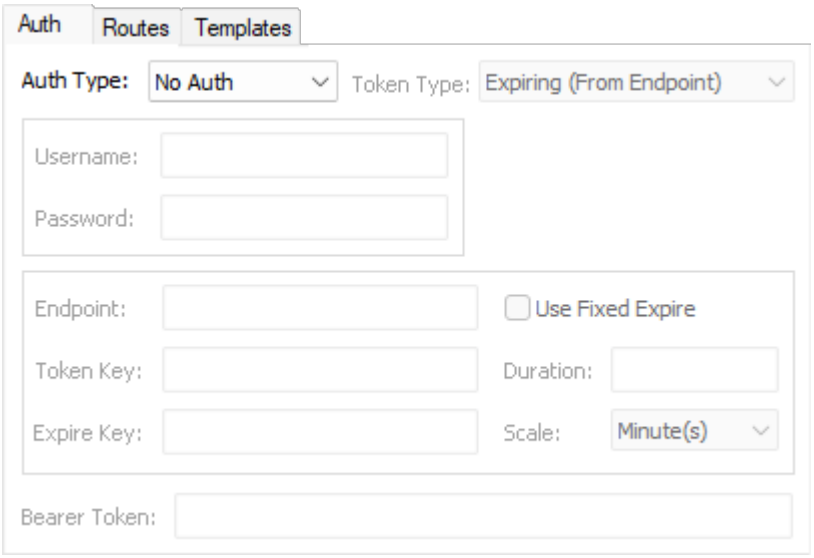


Aba RESTAPIClient

As opções disponíveis na aba **RESTAPIClient** estão descritas na tabela a seguir.

Opções disponíveis na aba RESTAPIClient

OPÇÃO	DESCRIÇÃO
API URL	Caminho da API (*Application Programming Interface* ou *Interface de Programação de Aplicações*), composta pelo protocolo, **HTTP** ou **HTTPS**, pelo domínio, como por exemplo "https://api.exemplo.com", e, quando necessário, por uma porta TCP/IP, como por exemplo "https://api.exemplo.com:8080", definida antes das rotas específicas que identificam os recursos de uma API
Auth	Selecione esta aba para configurar a autenticação de uma API. Para mais informações, consulte a tabela **Opções disponíveis na aba Auth**
Routes	Selecione esta aba para configurar as rotas específicas de uma API. Para mais informações, consulte a tabela **Opções disponíveis na aba Routes**
Templates	Selecione esta aba para configurar os modelos no formato **JSON** de requisição e resposta de uma API. Para mais informações, consulte a tabela **Opções disponíveis na aba Templates**



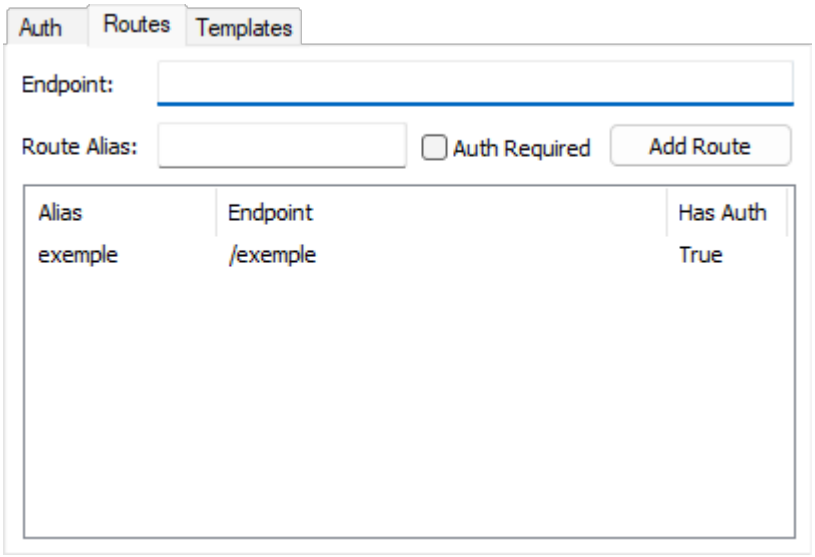
Aba Auth

As opções disponíveis na aba **Auth** estão descritas na tabela a seguir.

Opções disponíveis na aba Auth

OPÇÃO	DESCRIÇÃO
Auth Type	Define o tipo de autenticação utilizado por este Driver. As opções disponíveis são **No Auth**, **Basic Auth** ou **Bearer Token**
Token Type	Define o tipo de *Bearer Token*. Esta opção só é habilitada quando a opção **Auth Type** está configurada como **Bearer Token**. As opções disponíveis são **Expiring (From**

OPÇÃO	DESCRIÇÃO
	Endpoint), **Non-Expiring (From Endpoint)** ou **Non-Expiring (Provided)**
Username	Define o nome de um usuário. Disponível apenas para autenticações dos tipos **Basic Auth** ou **Bearer Token** dos tipos **Expiring (From Endpoint)** ou **Non-Expiring (From Endpoint)**
Password	Define a senha de um usuário. Disponível apenas para autenticações dos tipos **Basic Auth** ou **Bearer Token** dos tipos **Expiring (From Endpoint)** ou **Non-Expiring (From Endpoint)**
Endpoint	Define o *endpoint* usado para obter um *Bearer Token*, como por exemplo "/auth/token". Disponível apenas para a autenticação do tipo **Bearer Token** dos tipos **Expiring (From Endpoint)** ou **Non-Expiring (From Endpoint)**
Token Key	Define uma chave no arquivo do tipo **JSON** de resposta do *endpoint* que contém um *Bearer Token*, como por exemplo "access_token". Disponível apenas para a autenticação do tipo **Bearer Token** dos tipos **Expiring (From Endpoint)** ou **Non-Expiring (From Endpoint)**
Expire Key	Define uma chave no arquivo do tipo **JSON** de resposta do *endpoint* que contém a duração de um *token*, em segundos, como por exemplo "expires_in". Disponível apenas para a autenticação do tipo **Bearer Token** do tipo **Expiring (From Endpoint)**
Use Fixed Expire	Define um tempo fixo de expiração para um *token* fornecido manualmente, como por exemplo 3600 segundos. Selecionar esta opção desativa a opção **Expire Key**. Disponível apenas para a autenticação do tipo **Bearer Token** do tipo **Expiring (From Endpoint)**
Duration	Define o tempo de validade de um *token*, que deve ser um valor numérico. Disponível apenas quando a opção **Use Fixed Expire** está selecionada
Scale	Define a unidade de tempo utilizada para a opção **Duration**. As opções disponíveis são **Second(s)**, **Minute(s)**, **Hour(s)** ou **Day(s)**. Disponível apenas quando a opção **Use Fixed Expire** está selecionada
Bearer Token	Define um *token* fixo que é utilizado na autenticação. Disponível apenas para a autenticação do tipo **Bearer Token** do tipo **Non-Expiring (Provided)**



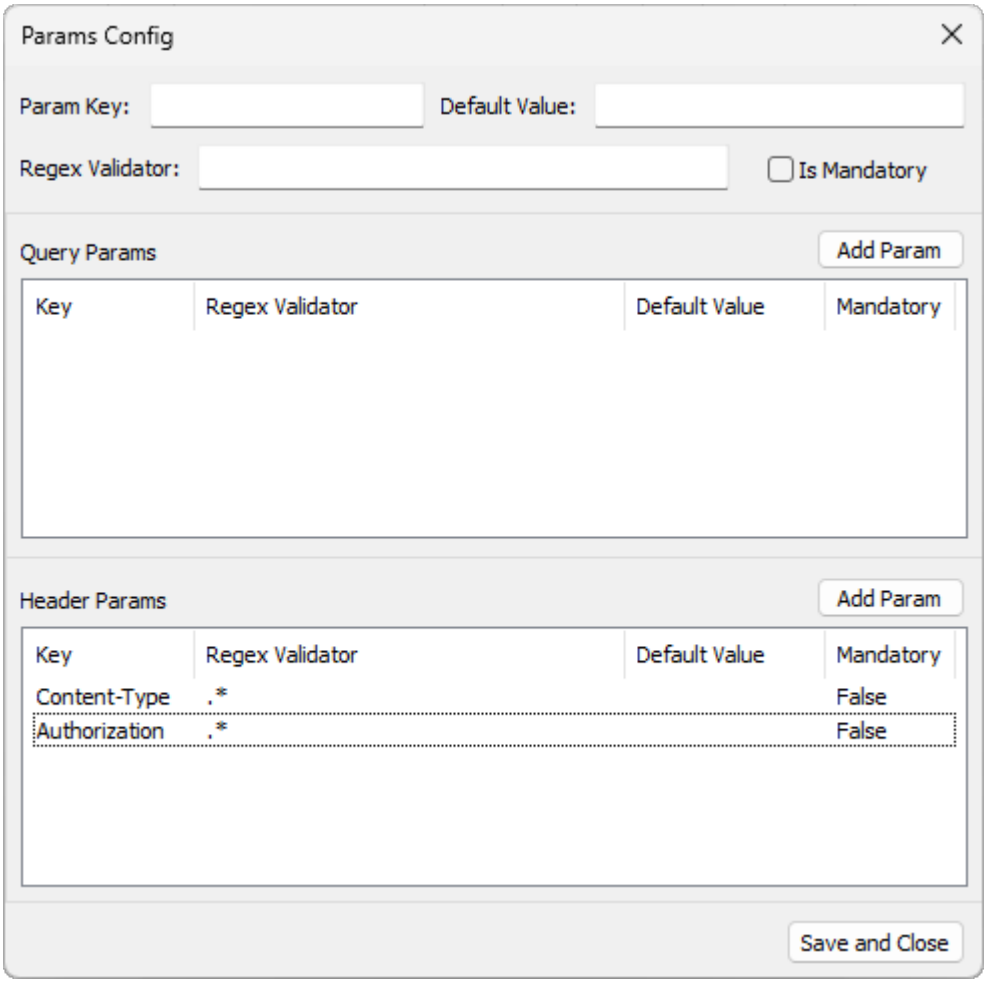
Aba Routes

As opções disponíveis na aba **Routes** estão descritas na tabela a seguir.

Opções disponíveis na aba Routes

OPÇÃO	DESCRIÇÃO
Endpoint	Define os *endpoints* que são utilizados em requisições. Cada *endpoint* deve começar com uma barra seguida de um caminho, como por exemplo "/api/usuarios". Também é possível definir segmentos dinâmicos em um *endpoint*, que são substituídos por valores configurados via Tags em uma requisição. Estes segmentos devem ser indicados por dois pontos, como por exemplo "/api/empresa/:empresaId/usuario/:usuarioId"
Route Alias	Define um apelido para uma rota. Este apelido é utilizado posteriormente para vincular Tags de uma aplicação a uma rota específica. O valor desta opção diferencia entre maiúsculas e minúsculas
Auth Required	Esta opção indica que uma rota específica utiliza o método de autenticação configurado na opção **Auth Type** da aba **Auth**
Add Route	Adiciona uma rota à tabela de rotas configuradas

Clicar com o botão direito do mouse em uma rota mostra as opções **Edit**, que abre a janela Params Config, e **Delete**, que permite remover a rota selecionada. Clicar duas vezes em uma rota também abre a janela Params Config, mostrada na figura a seguir, que permite configurar os parâmetros específicos de uma rota.



Janela Params Config

As opções disponíveis na janela Params Config estão descritas na tabela a seguir.

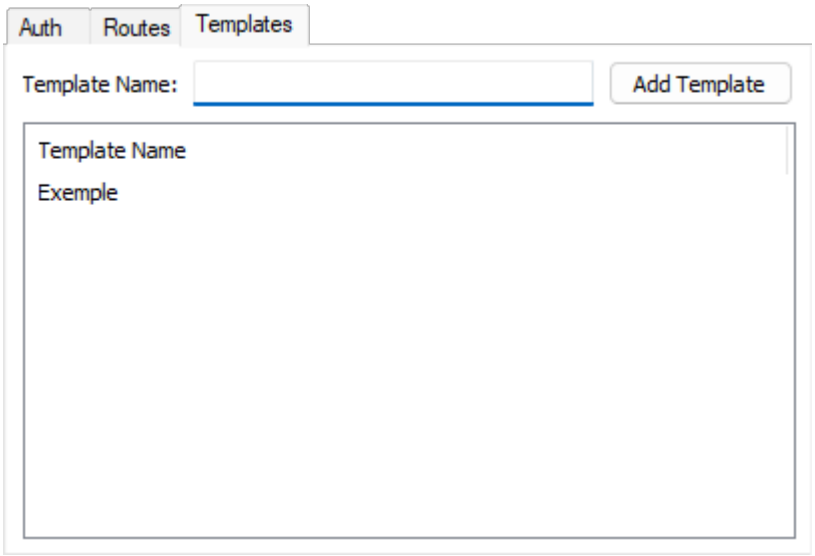
Opções disponíveis na janela Params Config

OPÇÃO	DESCRIÇÃO
Param Key	Define o nome de um parâmetro a ser utilizado em uma requisição
Default Value	Define um valor padrão para um parâmetro. Se nenhum valor é fornecido, considera-se esta opção como vazia
Regex Validator	Permite definir uma expressão regular (*regular expression*) para validar um valor atribuído a um parâmetro, se necessário
Is Mandatory	Selecionar esta opção impede o envio de uma requisição se um parâmetro não está configurado ou está configurado com um valor vazio
Add Param	Adiciona um novo parâmetro às tabelas Query Params ou Header Params. Cada uma destas tabelas possui a respectiva opção **Add Param**, que permite inserir um parâmetro de consulta (*query*) ou de cabeçalho (*header*), respectivamente
Save and Close	Salva os dados configurados e fecha esta janela

Clicar com o botão direito do mouse em um parâmetro de qualquer uma das tabelas mostra a opção **Delete**, que permite remover o parâmetro selecionado.

Os parâmetros **Content-Type** e **Authorization** são inseridos automaticamente como parâmetros do tipo *header* na tabela Header Params. Estes parâmetros não podem ser excluídos nem configurados posteriormente por meio de Tags.

Em caso de se utilizar um tipo de autenticação diferente das opções disponíveis na aba **Auth**, como por exemplo usar uma chave de API, uma rota pode ser configurada para não usar autenticação, ou seja, desmarcar a opção **Auth Required**, e neste caso os parâmetros *header* podem ser configurados, definindo os respectivos valores da coluna **Default Value** tanto manualmente quanto em tempo de execução usando Tags.



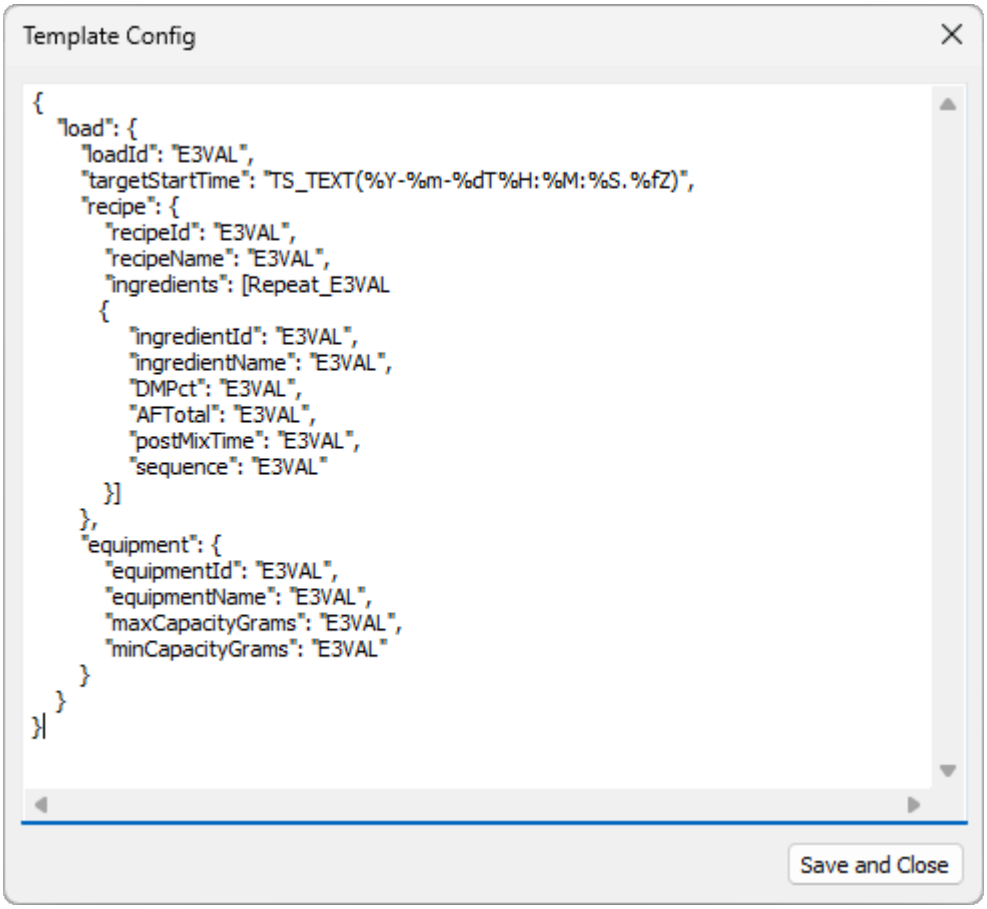
Aba Templates

As opções disponíveis na aba **Templates** estão descritas na tabela a seguir.

Opções disponíveis na aba Templates

OPÇÃO	DESCRIÇÃO
Template Name	Define o nome de um modelo, no formato **JSON**, que é utilizado para as requisições e respostas de uma API
Add Template	Adiciona um modelo à tabela de modelos configurados

Clicar com o botão direito do mouse em um modelo mostra as opções **Edit**, que abre a janela Template Config, e **Delete**, que permite remover o modelo selecionado. Clicar duas vezes em um modelo também abre a janela Template Config, mostrada na figura a seguir, que permite configurar um modelo no formato **JSON**, conforme o padrão de sintaxe deste formato descrito no tópico a seguir.



Janela Template Config

A opção disponível na janela Template Config é **Save and Close**, que salva os dados configurados e fecha esta janela.

Sintaxe de um Modelo JSON

Para extrair ou montar o conteúdo de uma mensagem é necessário declarar um modelo, ou *template*, que permite informar o formato desta mensagem e quais partes devem ser transformadas em dados. Cada *template* deve utilizar palavras-chave colocadas no lugar dos valores a serem extraídos. As palavras-chave disponíveis estão descritas na tabela a seguir.

Palavras-chave disponíveis para templates

PALAVRA-CHAVE	DESCRIÇÃO
TS_TEXT(format)	Estampa de tempo, no formato **Texto**, que é usada como estampa de tempo de um Tag Bloco ou Tag de Comunicação. O significado de cada campo está descrito na tabela **Opções disponíveis para a palavra-chave TS_TEXT**
TS_UNIX	Estampa de tempo, no formato de segundos desde 1970, ou formato **UNIX**. Este valor pode ser um número ou um texto, e é usado como estampa de tempo de um Tag Bloco ou Tag de Comunicação, como por exemplo 1504198675 ou "1504198675"
QL_OPC	Qualidade, no padrão OPC DA, utilizando 1 (um) byte. Este valor é usado diretamente como valor de qualidade em um Tag Bloco ou Tag de Comunicação, sem transformações. Para mais informações, consulte a tabela **Padrão OPC DA**

PALAVRA-CHAVE	DESCRIÇÃO
QL_BOOL	Qualidade, no padrão Booleano. Se este valor é maior que 0 (zero) ou a expressão "true" ou "TRUE", a qualidade é boa. Se o valor é igual a 0 (zero) ou a expressão "false" ou "FALSE", a qualidade é ruim
E3VAL	Especifica um valor que é extraído na sequência que ocorre para um Tag Bloco, cada valor em um Elemento de Bloco. Se há apenas um valor **E3VAL** em uma mensagem, o *template* também pode ser usado com um Tag de Comunicação. Valores recebidos como texto podem ser convertidos em número, conforme a seção **Conversão de Valores Recebidos como Texto**. Esta palavra-chave dentro de um *array* declara um campo do tipo **Lista**, conforme a seção **Campos do Tipo Lista**
DUMMY	Campo variável, mas cujo valor não deve ser enviado para Tags de Comunicação
Repeat_E3VAL	Palavra-chave que só pode ser usada dentro de um *array* no formato **JSON** para indicar que os elementos repetidos dentro deste *array* devem ser processados de forma independente por este Driver, retornando uma leitura para cada conjunto. Um *template* pode ter esta palavra-chave apenas uma vez

Opções disponíveis para a palavra-chave TS_TEXT

OPÇÃO	DESCRIÇÃO
%a	Dia da semana abreviado, em inglês
%A	Dia da semana completo, em inglês
%b	Mês abreviado, em inglês
%B	Mês completo, em inglês
%C	Século
%d	Dia do mês, começando com 0 (zero)
%e	Dia do mês, começando com um espaço
%f	Milissegundos, de 0 (zero) a 999
%h	Hora, no formato de 12 horas
%H	Hora, no formato de 24 horas
%m	Mês
%M	Minuto
%p	AM ou PM
%S	Segundos
%y	Ano com dois dígitos
%Y	Ano com quatro dígitos
%Z	Nome do fuso horário, um código internacional que transforma o horário para o formato **GMT** (*Greenwich Mean Time*)

OPÇÃO	DESCRIÇÃO
%+	Deslocamento ou *offset* do horário GMT no formato **±HH:MM**

O código a seguir contém exemplos de estampas de tempo formatados pela palavra-chave **TS_TEXT**.

```
"2014-07-11T15:26:37Z" -> "TS_TEXT(%y-%m-%dT%H:%M:%SZ)"
"Mon Jul 10 11:04:47 BRT 2017" -> "TS_TEXT(%a %b %d %H:%M:%S %Z %Y)"
"2018-05-02T10:29:28.622-02:00" -> "TS_TEXT(%Y-%m-%dT%H:%M:%S.%f%+)"
```

Padrão OPC DA

Bit	7	6	5	4	3	2	1	0
Descrição	Q	Q	S	S	S	S	L	L

Opções disponíveis para o padrão OPC DA

OPÇÃO	DESCRIÇÃO
QQ	Dois bits de qualidade
SSSS	Quatro bits de sub-status
LL	Dois bits de limite
QQ	Os valores possíveis são **0**: BAD, **1**: UNCERTAIN ou **3**: GOOD
SSSS	Os valores possíveis são **0**: BAD_NONSPECIFIC, **1**: BAD_CONFIGERROR, **2**: BAD_NOTCONNECTED, **3**: BAD_DEVICEFAILURE ou **4**: BAD_SENSORFAILURE
SSSS	Os valores possíveis são **0**: UNCERT_NONSPECIFIC, **1**: UNCERT_LASTUSABLEVALUE ou **4**: UNCERT_SENSORNOTACCURATE
SSSS	Os valores possíveis são **0**: GOOD_NONSPECIFIC, **1**: GOOD_LOCALOVERRIDE ou **6**: GOOD_NONSPECIFICLOCALTIMESTAMP
LL	Os valores possíveis são **0**: FREE, **1**: LOW, **2**: HIGH ou **3**: CONST

Exemplos de Configuração de Modelos no Formato JSON

Um arquivo no formato **JSON** contendo um *array* com múltiplos conjuntos de dados deve incluir a palavra-chave **Repeat_E3VAL** no *template* imediatamente após o colchete que inicia este *array*, conforme o exemplo a seguir.

```
{
  "MessageType":"DUMMY",
  "TagData": [
    Repeat_E3VAL {
      "Temperature": "E3VAL",
      "Humidity": "E3VAL"
    }
  ],
  "MessageDesc": "DUMMY"
}
```

Neste caso, ao receber uma mensagem com 2 (dois) conjuntos de dados, por exemplo, o evento **OnRead** de um Tag Bloco é disparado para cada conjunto, como mostrado no código a seguir.

```
Sub [Sensor-001_OnRead]()
  Application.Trace Item("Temperature").Value
  Application.Trace Item("Humidity").Value
End Sub
```

Se outros valores do tipo **E3VAL** são declarados antes ou depois de um valor do tipo **Repeat_E3VAL**, estes são replicados para cada conjunto, com o valor do tipo **Repeat_E3VAL** sempre como o último valor no Tag Bloco de recebimento, como no exemplo a seguir.

```
{
  "MessageType": "E3VAL",
  "TagData": [
    Repeat_E3VAL {
      "Temperature": "E3VAL",
      "Humidity": "E3VAL"
    }
  ],
  "MessageDesc": "E3VAL"
}
```

Um Tag Bloco com os Elementos de Bloco **Tipo**, **Descrição**, **Temperatura** e **Umidade** pode gerar, a cada evento **OnRead**, valores como no código a seguir.

```
"Teste";"Descrição";11;12 => Primeiro evento OnRead
"Teste";"Descrição";14;66 => Segundo evento OnRead
"Teste";"Descrição";70;55 => Terceiro evento OnRead
```

Caso sem Repetição de Leitura por Conjunto

Se não há necessidade de disparar um evento **OnRead** para cada conjunto, um *template* pode ser declarado sem um valor do tipo **Repeat_E3VAL**, usando apenas uma palavra-chave do tipo **E3VAL** para capturar o *array*, conforme o exemplo a seguir.

```
{
  "MessageType": "DUMMY",
  "TagData": "E3VAL",
  "MessageDesc":"DUMMY"
}
```

Nesse caso, o *array* recebido é expandido com base na quantidade de elementos, resultando em uma única leitura no evento **OnRead**, no formato do exemplo a seguir.

```
array(11, 12); array(14, 66); array(70, 55) => Evento OnRead único
```

Se um *template* contém mais de uma palavra-chave do tipo **E3VAL**, ou seja, não **DUMMY**, este *array* não é expandido, conforme o exemplo a seguir.

```
{
  "MessageType": "E3VAL",
  "TagData": "E3VAL",
  "MessageDesc": "E3VAL"
}
```

Esta leitura é realizada em um único evento **OnRead**, no formato do exemplo a seguir.

```
"Teste"; array(array(11, 12), array(14, 66), array(70, 55)); "Descrição"=> Evento OnRead único
```

NOTA

O corpo de uma requisição é montado no formato **UTF-8**, como o formato **JSON** exige, e a resposta também é interpretada no formato **UTF-8**. Valores com acentos podem ser usados normalmente nos Elementos de Bloco.

Campos do Tipo Lista

Um campo do corpo de uma requisição pode ser declarado como uma lista, utilizando a palavra-chave **E3VAL** dentro de um *array*, conforme o exemplo a seguir.

```
{
  "canais": ["E3VAL"],
  "id": "E3VAL"
}
```

Este modelo tem dois Elementos de Bloco, o primeiro corresponde ao campo **canais** e o segundo ao campo **id**. Um campo declarado como lista consome um único Elemento de Bloco, e o valor escrito neste Elemento define o conteúdo desta lista, conforme a tabela a seguir.

Exemplos de preenchimento de um campo do tipo Lista

VALOR ESCRITO EM UM ELEMENTO DE BLOCO	CAMPO ENVIADO EM UMA REQUISIÇÃO
"1/1"	"canais": ["1/1"]
array("1/1", "2/2")	"canais": ["1/1", "2/2"]

NOTA

Não utilize mais de uma palavra-chave dentro de um mesmo *array*, como por exemplo ["E3VAL", "E3VAL"], porque todas as posições correspondem ao mesmo campo e o último valor escrito substitui os anteriores. Para enviar uma lista de objetos, utilize a palavra-chave **Repeat_E3VAL**.

Conversão de Valores Recebidos como Texto

Quando o valor extraído por uma palavra-chave **E3VAL** chega como texto, ou seja, entre aspas no formato **JSON**, este Driver converte este valor para um número somente se o texto inteiro é um número, ou seja, apenas dígitos, opcionalmente um sinal negativo no início e no máximo um separador decimal, que pode ser ponto ou vírgula. Qualquer outro texto, mesmo formado só por dígitos, pontos ou hifens, como por exemplo um endereço IP ou uma data, é entregue a um Tag exatamente como foi recebido. Esta regra também vale para os valores dentro de *arrays* e objetos, e também para os **Tags de Caminho JSON**.

Exemplos de conversão de valores recebidos como texto

VALOR RECEBIDO	VALOR ENTREGUE A UM TAG
"123"	Número 123

VALOR RECEBIDO	VALOR ENTREGUE A UM TAG
"-1,5"	Número -1,5 (a vírgula é aceita como separador decimal)
"192.168"	Número 192,168
"007"	Número 7 (sete)
"192.168.17.30"	Texto 192.168.17.30 (endereço IP com mais de um separador)
"2025-12-22"	Texto 2025-12-22 (data com sinal fora do início)
"10-20"	Texto 10-20

Referência de Tags

Esta seção contém informações sobre a configuração dos Tags PLC e Bloco deste Driver.

Também descreve as **Tags de Caminho JSON**, que permitem ler um valor diretamente pelo seu caminho no **JSON** da resposta, sem declarar um *template*.

Tag Bloco de Recebimento

Somente Leitura

Utilize este Tag Bloco para leitura dos dados recebidos de uma requisição à uma API, configurando os parâmetros conforme a tabela a seguir.

Item	*<Route Alias>;<Template Name>*
B1	Não utilizado, deixe em 0 (zero)
B2	Não utilizado, deixe em 0 (zero)
B3	Não utilizado, deixe em 0 (zero)
B4	Não utilizado, deixe em 0 (zero)

O tamanho deste Tag Bloco depende do *template* associado e corresponde à quantidade de elementos no arquivo em formato **JSON** cuja palavra-chave seja diferente de **DUMMY**, exceto quando a única palavra-chave diferente de **DUMMY** existente seja um *array*. Neste caso, este *array* é expandido e este Tag Bloco tem o mesmo tamanho do *array* recebido.

Como alternativa a um *template*, um valor da resposta pode ser lido informando diretamente o caminho no formato **JSON** no parâmetro *Item*, com o prefixo $ (cifrão). Para mais informações, consulte o tópico **Tags de Caminho JSON**.

NOTA

No caso de um Tag Bloco de Recebimento cujo *template* associado contém a palavra-chave **Repeat_E3VAL**, o conjunto de dados referente a esta palavra-chave vem nas últimas posições deste Tag Bloco.

Tag Bloco de Envio

Somente Escrita

Utilize este Tag Bloco para a escrita dos dados referentes à uma requisição para uma API, configurando os parâmetros conforme a tabela a seguir.

Item	*<Route Alias>;<Template Name>*
B1	Não utilizado, deixe em 0 (zero)
B2	Não utilizado, deixe em 0 (zero)
B3	Não utilizado, deixe em 0 (zero)
B4	Não utilizado, deixe em 0 (zero)

O tamanho deste Tag Bloco é a soma da quantidade de parâmetros necessários em uma requisição, ou seja, *header*, *route*, *query* e *body*.

Os primeiros Elementos de Bloco correspondem aos parâmetros de *header* adicionados na janela **Params Config**, exceto os parâmetros **Content-Type** e **Authorization**, que são ignorados.

Em seguida vêm os parâmetros de rota, definidos na aba **Routes** e associados ao *endpoint*, como por exemplo "/api/empresa/:empresaId/usuario/:usuarioId", em que os parâmetros são **:empresaId** e **:usuarioId**.

Logo após são definidos os parâmetros de *query*, também configurados na janela **Params Config**.

Os Elementos restantes deste Tag Bloco são destinados à definição dos dados do corpo (*body*) de uma requisição, ou seja, os valores definidos pelo *template* associado ao Tag Bloco, caso exista.

A quantidade de Elementos de Bloco destinados ao corpo de uma requisição corresponde à quantidade de itens no *template* associado, e todos os Elementos devem ser preenchidos. No entanto, caso o *array* especial **Repeat_E3VAL** esteja presente, este *array* é considerado como apenas um único Elemento de Bloco, diferente do que ocorre no **Tag Bloco de Recebimento**, onde este *array* é expandido. Além disto, este *array* permanece na posição original do *template* e não é movido para o final deste Tag Bloco.

Um campo declarado como lista em um *template*, ou seja, com uma palavra-chave dentro de um *array*, também corresponde a um único Elemento de Bloco. O valor escrito neste Elemento pode ser um valor simples, que é enviado como uma lista de um item, ou um *array*, que define uma lista inteira, conforme a seção **Campos do Tipo Lista**.

NOTA

Se este Tag Bloco tem mais Elementos do que o necessário, a escrita nos Elementos excedentes falha e este Driver registra no arquivo de log o *offset* e a quantidade de valores do modelo. O mesmo ocorre quando o modelo informado no parâmetro *Item* deste Tag não existe.

Os valores escritos nos Elementos permanecem no corpo das requisições seguintes, até serem escritos novamente ou até a comunicação ser reiniciada, portanto não é necessário escrever todos os Elementos a cada requisição. Enquanto algum valor do modelo não for escrito, uma requisição é recusada e este Driver registra no arquivo de log o campo correspondente. Uma alteração no modelo passa a valer depois de reiniciar a comunicação.

Para escrever neste *array*, deve-se informar na aplicação um *array* de *arrays*, em que cada elemento representa um conjunto de dados a ser inserido. Cada conjunto é interpretado como um *array* e os valores são associados por posição com os campos definidos no *template*.

Exemplo de um *template*.

```
{
  "TagData": [
    Repeat_E3VAL {
      "Temperature": "E3VAL",
      "Humidity": "E3VAL"
    }
  ]
}
```

Exemplo de escrita do elemento correspondente neste Tag Bloco.

```
array(array(11, 12), array(14, 66), array(70, 55))
```

Exemplo do resultado final, ou seja, o código no formato **JSON** gerado.

```
{
  "TagData": [
    {
      "Temperature": 11,
      "Humidity": 12
    },
    {
      "Temperature": 14,
      "Humidity": 66
    },
    {
      "Temperature": 70,
      "Humidity": 55
    }
  ]
}
```

Escrita de um Bloco Inteiro

Este Tag Bloco pode ser escrito Elemento a Elemento ou de uma só vez, escrevendo um bloco inteiro a partir de uma aplicação, como por exemplo usando o método **Write** de um Tag Bloco no **Elipse E3**, no **Elipse Power** ou no **Elipse Water**. Em ambas as formas, cada valor é gravado no parâmetro correspondente à respectiva posição, na ordem descrita anteriormente, ou seja, parâmetros de *header*, parâmetros de rota, parâmetros de *query* e, por último, os dados do corpo.

Para a escrita de um bloco inteiro, o tamanho (*Size*) deste Tag Bloco deve ser igual à quantidade de parâmetros de *header*, exceto **Content-Type** e **Authorization**, mais a quantidade de parâmetros de rota, mais a quantidade de parâmetros de *query*, mais a quantidade de Elementos do corpo, que é 1 (um) quando o *template* utiliza o *array* **Repeat_E3VAL**.

Por exemplo, em uma rota com o parâmetro de *header* **companyID** e um *template* de corpo com **Repeat_E3VAL**, um Tag Bloco tem *Size* igual a 2 (dois), em que o Elemento 1 (um) é **companyID** e o Elemento 2 (dois) são os dados do corpo. Escrever o valor a seguir em um bloco inteiro envia **companyID** igual a 7 (sete) e o corpo com 2 (dois) conjuntos de dados.

```
array(7, array(array(11, 12), array(14, 66)))
```

NOTA

A escrita de dados inválidos nos parâmetros, seja por não atenderem às **expressões regulares** configuradas ou por apresentarem um tipo de dados incompatível, falha esta escrita.

Tag PLC

Somente Escrita

Utilize um Tag PLC para enviar uma requisição à uma API configurando os parâmetros conforme a tabela a seguir.

Item	*<Route Alias>;<Template Name>*
N1	1 (um)
N2	Entre 0 (zero) e 3 (três)
N3	Não utilizado, deixe em 0 (zero)
N4	Não utilizado, deixe em 0 (zero)

Qualquer escrita neste Tag executa uma requisição utilizando a rota configurada no parâmetro *Item* e, quando necessário, envia o corpo (*body*) da mensagem com base no *template* associado.

O parâmetro *N2* mapeia o método da requisição HTTP. Os valores possíveis são **0**: Método **GET**, **1**: Método **POST**, **2**: Método **PUT** ou **3**: Método **DELETE**.

O resultado desta escrita depende do código de status HTTP retornado pela API. Uma requisição é considerada bem-sucedida quando a resposta tem qualquer código na faixa *2XX*, de 200 a 299, como por exemplo **200**: OK, **201**: Created, **202**: Accepted ou **204**: No Content, e o corpo da resposta, se existir, é processado normalmente. Códigos fora desta faixa, como por exemplo *4XX* ou *5XX*, ou uma falha de conexão retornam um erro de escrita para uma aplicação. Em ambos os casos, o código de status recebido é registrado no arquivo de log deste Driver.

NOTA

Se nem todos os elementos obrigatórios para uma requisição estão configurados no **Tag Bloco de Envio**, esta escrita falha. Da mesma forma, se uma requisição falha por qualquer outro motivo, esta escrita também não é concluída com sucesso.

Tags de Caminho JSON

Somente Leitura

Utilize um Tag de Comunicação ou um Tag Bloco para ler um valor diretamente pelo caminho no formato **JSON** da resposta de uma requisição, sem declarar um *template*. Este caminho é informado no parâmetro *Item* com o prefixo $ (cifrão), conforme a tabela a seguir.

Item	*<Route Alias>;$<Caminho>[@<Caminho da estampa de tempo>[:<Formato>]]*
N1 ou B1	Não utilizado, deixe em 0 (zero)
N2 ou B2	Não utilizado, deixe em 0 (zero)
N3 ou B3	Não utilizado, deixe em 0 (zero)
N4 ou B4	Não utilizado, deixe em 0 (zero)

Os Tags de Caminho JSON convivem com os Tags que usam *template* na mesma rota e na mesma requisição. São indicados para leituras pontuais em respostas grandes e para respostas com chaves dinâmicas, como por exemplo datas ou identificadores que não podem ser mapeados por um *template*. Não deve haver espaços ao redor do caractere ; (ponto e vírgula) no parâmetro *Item*.

Uma requisição continua sendo disparada por uma escrita no **Tag PLC** configurado com o parâmetro *N1* igual a 1 (um). Ao receber uma resposta, este Driver navega até o caminho informado e entrega o valor a este Tag. Utilize um Tag de Comunicação para um valor único e um Tag Bloco para listas e objetos.

Elementos de um caminho

ELEMENTO	SINTAXE	EXEMPLO	DESCRIÇÃO
Chave de objeto	chave	total_30d	Acessa a chave de um objeto
Chaves aninhadas	a.b.c	dados.config.max	Navega em objetos dentro de objetos
Índice de array	[n] ou [-n]	dados[0], dados[-1]	*n*-ésimo elemento de um *array*. Um índice negativo conta a partir do fim de um *array*
Índice posicional em objeto	[n] ou [-n]	chuva_diaria[-1]	*n*-ésima chave de um objeto, na ordem de um documento
Wildcard	[*]	dados[*].nivel	Todos os elementos de um *array*. Este Tag recebe uma lista com o restante do caminho aplicado a cada elemento
Macro de data	{TODAY}, {TODAY-n} ou {TODAY+n}	chuva_diaria.{TODAY-1}	Data local atual, menos ou mais *n* dias, usada como chave. O formato padrão é **%Y-%m-%d**. Esta macro é expandida a cada requisição
Macro de data com formato	{TODAY-n:formato}	{TODAY-1:%d/%m/%Y}	Formato da data com as mesmas opções da palavra-chave **TS_TEXT**
Chave com aspas	'chave' ou "chave"	'config.v2'	Endereça chaves que contêm ponto, colchetes ou chaves
Estampa de tempo de um dado	@caminho[:formato]	nivel@datetime	Carimba um valor com a estampa de tempo lida do próprio formato **JSON**, conforme descrito a seguir

Estampa de Tempo do Dado

O caminho após o caractere @ (arroba) é resolvido a partir da raiz de uma resposta e aceita a mesma sintaxe do caminho de um valor, exceto o caractere de * (asterisco ou *wildcard*). Quando o caminho de um valor aponta para um *array* de objetos, ou modo de repetição, o caminho da estampa de tempo é resolvido em relação a cada elemento, carimbando cada leitura com o tempo da própria amostra.

Se o valor apontado é um número, este é interpretado como uma estampa de tempo no formato **UNIX**, em segundos. Se este valor é um texto, este é convertido com o formato informado após o caractere de dois pontos, cujo padrão é **%Y-%m-%dT%H:%M:%S**, com as mesmas opções da palavra-chave **TS_TEXT**, incluindo o padrão **%f** para milissegundos e **%+** para o fuso horário.

Uma falha em uma estampa de tempo não impede a entrega de um valor. Este erro é registrado no arquivo de log deste Driver e o valor é entregue com a estampa de tempo padrão, que é a hora da leitura.

Conversão de um valor para um Tag

VALOR NO FORMATO JSON	RESULTADO EM UM TAG
Número ou texto	Valor de um Tag. Textos que representam números seguem a regra descrita na seção **Conversão de Valores Recebidos como Texto**
Objeto	Tag Bloco com os valores das chaves, na ordem de um documento. Objetos aninhados são convertidos em sub-listas
Array de números ou textos	Tag Bloco com a lista de valores
Array de objetos	Uma leitura por elemento, cada leitura com os valores das chaves de um objeto, de forma equivalente à palavra-chave **Repeat_E3VAL**
null	Valor vazio com qualidade ruim

Exemplos

Os exemplos a seguir usam a rota **demanda**, cuja resposta, em resumo, é a seguinte.

```
{
  "chuva_diaria": {
    "2026-06-12": 0.4,
    ...
    "2026-07-22": 15.4
  },
  "dados": [
    {
      "chuva_15min": 70.0,
      "nivel": 2.64,
      "datetime_brt": "2026-07-12T14:30:00-03:00"
    },
    ...
    {
      "chuva_15min": 0.0,
      "nivel": 2.57,
      "datetime_brt": "2026-07-22T13:15:00-03:00"
    }
  ],
  "estacao": "BE70C78A",
  "total_30d": 9359.4
}
```

Exemplos de Itens e resultados

ITEM	RESULTADO
demanda;$total_30d	9359.4
demanda;$estacao	BE70C78A
demanda;$dados[-1].nivel	2.57 (nível da última amostra)
demanda;$dados[0].nivel	2.64 (nível da primeira amostra)
demanda;$chuva_diaria.{TODAY}	15.4 (chave 2026-07-22, considerando a data atual como 22/07/2026)
demanda;$chuva_diaria[-1]	15.4 (valor da última chave de um objeto)

ITEM	RESULTADO
demanda;$dados[-1].nivel@dados[-1].datetime_brt:%Y-%m-%dT%H:%M:%S%+	2.57 com estampa de tempo 22/07/2026 13:15:00, lida da própria amostra
demanda;$dados[*].nivel	Tag Bloco com a lista de todos os níveis
demanda;$chuva_diaria	Tag Bloco com os valores diários, na ordem deste documento
demanda;$dados[0]	Tag Bloco com os campos do primeiro registro, na ordem deste documento
demanda;$dados	Uma leitura por registro, cada leitura com os campos de um objeto, como na palavra-chave **Repeat_E3VAL**
demanda;$dados@datetime_brt:%Y-%m-%dT%H:%M:%S%+	Mesmo do item anterior, com cada leitura carimbada com a estampa de tempo da própria amostra
demanda;$	A resposta inteira como um Tag Bloco aninhado, ou seja, **chuva_diaria**, **dados**, **estacao** e **total_30d**

NOTA

Os Tags de Caminho JSON são somente de leitura. A escrita nestes Tags é rejeitada e o envio de dados continua exigindo um *template*. Um caminho que não existe em uma resposta é registrado no arquivo de log e não impede a atualização dos demais Tags de uma mesma requisição. Com o caractere de * (asterisco ou *wildcard*), os elementos em que o restante do caminho não existe são ignorados. Um parâmetro *Item* mal formado é rejeitado na criação deste Tag. A macro de data usa o relógio local do computador onde este Driver é executado.

Documentação das Interfaces de Comunicação

Esta seção contém a documentação das Interfaces de Comunicação referentes ao Driver **RESTAPIClient**.

Configurações de um Driver

A configuração das Interfaces de Comunicação é realizada na caixa de diálogo de configuração de um Driver. Para acessar a configuração da caixa de diálogo no **Elipse E3** na versão 1.0, siga estes passos:

1. Clique com o botão direito do mouse em um objeto Driver (IODriver).
2. Selecione o item **Propriedades** no menu contextual.
3. Selecione a aba **Driver**.
4. Clique em **Outros parâmetros**.

No **Elipse E3** versão 2.0 ou posterior, clique em **Configurar o driver** na barra de ferramentas de um Driver. No **Elipse SCADA**, siga estes passos:

1. Abra o Organizer.
2. Selecione um Driver na árvore do Organizer.
3. Clique em **Extras** na aba **Driver**.

Atualmente, as Interfaces de Comunicação permitem que apenas uma conexão seja aberta para cada Driver. Isto significa que, no caso de acesso a duas portas seriais, é preciso adicionar dois Drivers em um aplicação e configurar cada um destes Drivers para cada porta serial.

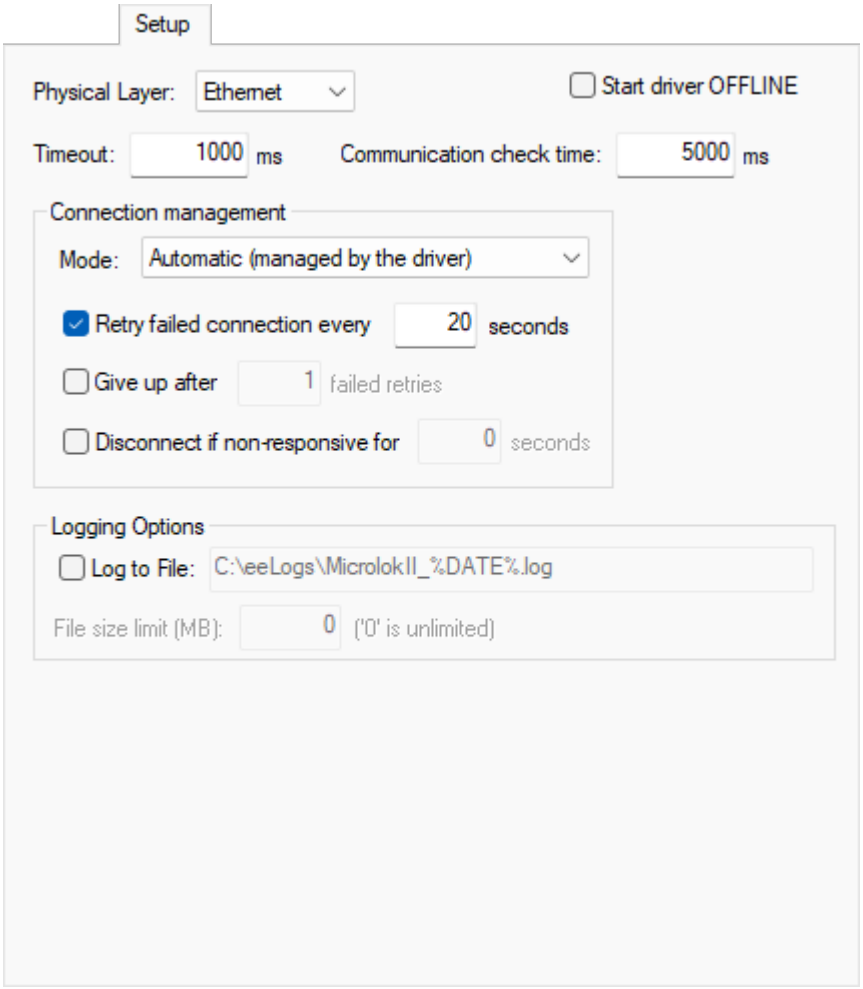
Caixa de Diálogo de Configuração

A caixa de diálogo das Interfaces de Configuração permite configurar a conexão de I/O que é utilizada por um Driver. Esta caixa de diálogo contém as abas **Setup**, **Serial**, **Ethernet**, **Modem** e **RAS** descritas nos tópicos a seguir. Se um Driver não implementa uma conexão de I/O específica, a respectiva aba não está disponível para configuração. Alguns Drivers podem conter abas adicionais, específicas para aquele Driver, na caixa de diálogo de configuração.

Aba Setup

A aba **Setup** contém a configuração geral de um Driver. Esta aba é dividida nos seguintes grupos:

- **Configurações gerais**: Configurações da camada física de um Driver, *time-out* e modo de inicialização
- **Connection management**: Configurações de como a Interface de Comunicação mantém a conexão e qual a política de recuperação em caso de falha
- **Logging options**: Controla a geração dos arquivos de log



Aba Setup

Opções gerais da aba Setup

OPÇÃO	DESCRIÇÃO
Physical Layer	Selecione a interface física em uma lista. As opções disponíveis são **Serial**, **Ethernet**, **Modem** e **RAS**. A

OPÇÃO	DESCRIÇÃO
	interface selecionada deve ser configurada na aba específica
Timeout	Configure o *time-out*, em milissegundos, para a camada física. Esta é a medida de tempo que a interface de I/O aguarda para a recepção de um byte qualquer do *buffer* de recepção
Communication check time	Configure o tempo, em milissegundos, para definir o intervalo em que a comunicação é considerada em estado inativo. Enquanto um Driver de Comunicação receber dados válidos, o estado de comunicação é considerado ativo. Porém, se durante o funcionamento um Driver de Comunicação não receber dados válidos neste período de tempo, o estado é considerado inativo. O estado de comunicação é mostrado no Tag **IO.CommunicationStatus**
Start driver OFFLINE	Selecione esta opção para que um Driver inicie em modo **Offline** ou parado. Isto significa que a interface de I/O não é criada até que se configure um Driver em modo **Online** utilizando-se um Tag em uma aplicação. Este modo possibilita a configuração dinâmica da interface de I/O em tempo de execução

Opções para o grupo Connection management

OPÇÃO	DESCRIÇÃO
Mode	Seleciona o modo de gerenciamento de conexão. Selecionar a opção **Automatic** permite que um Driver gerencie a conexão automaticamente, como especificado nas opções seguintes. Selecionar a opção **Manual** permite que uma aplicação gerencie a conexão completamente
Retry failed connection every ... seconds	Selecione esta opção para habilitar a retentativa de conexão de um Driver em um determinado intervalo, em segundos. Se a opção **Give up after failed retries** não está selecionada, este Driver continua retentando até que a conexão seja efetuada, ou que a aplicação seja parada
Give up after ... failed retries	Habilite esta opção para definir um número máximo de retentativas de conexão. Quando o número especificado de tentativas consecutivas de reconexão é atingido, um Driver vai para o modo **Offline**, assumindo que um problema de hardware foi detectado. Se um Driver estabelece uma conexão com sucesso, o número de retentativas sem sucesso é zerado. Se esta nova conexão é perdida, então o contador de retentativas inicia do zero
Disconnect if non-responsive for ... seconds	Habilite esta opção para forçar um Driver a se desconectar se nenhum byte chegou à interface de I/O no *time-out* especificado, em segundos. Este *time-out* deve ser maior que o *time-out* configurado na opção **Timeout**

Opções para o grupo Logging Options

OPÇÃO	DESCRIÇÃO
Log to File	Habilite esta opção e configure o nome do arquivo onde o log é escrito. Arquivos de log podem ser bem extensos, portanto utilize esta opção por curtos períodos de tempo, apenas para o propósito de testes e depurações. Caso se utilize a macro **%PROCESS%** no nome do arquivo de log, esta é substituída pelo identificador do processo atual. Esta opção é particularmente útil ao se utilizar várias instâncias de um mesmo Driver no **Elipse E3**, permitindo assim que cada instância gere um arquivo separado de log. Por exemplo, ao configurar esta opção com o valor "c:\e3logs\drivers\sim_%PROCESS%.log", gera-se um arquivo c:\e3logs\drivers\sim_00000FDA.log para o processo **0FDAh**. Pode-se também utilizar a macro **%DATE%** no nome do arquivo. Neste caso é gerado um arquivo de log por dia, no formato **aaaa_mm_dd**. Por exemplo, ao configurar esta opção com o valor "c:\e3logs\drivers\sim_%DATE%.log", gera-se o arquivo c:\e3logs\drivers\sim_2005_12_31.log em 31/12/2005 e o arquivo c:\e3logs\drivers\sim_2006_01_01.log em 01/01/2006. De forma semelhante, a macro **%DATE_HOUR%** gera um arquivo de log por hora, no formato **aaaa_mm_dd_hh**
File size limit (MB)	Configure o limite de tamanho do arquivo de log, em megabytes. Um valor igual a 0 (zero) significa que não há limite de tamanho para o arquivo de log

Configurações Gerais

Esta seção contém informações sobre a configuração dos **Tags de Comunicação** e das **Propriedades** gerais das Interfaces de Comunicação.

Tags de Comunicação

Tags Gerais das Interfaces de Comunicação (N2/B2 = 0)

Os Tags descritos a seguir são fornecidos para todas as Interfaces de I/O suportadas.

IO.CommunicationStatus

Tipo de Tag	Tag de Comunicação
Tipo de Acesso	Leitura
Parâmetro N1	-1 (menos um)
Parâmetro N2	0 (zero)
Parâmetro N3	0 (zero)
Parâmetro N4	6 (seis)
Configuração por String	IO.CommunicationStatus

Este Tag informa o estado da comunicação de um Driver. Indica o funcionamento da comunicação em função do recebimento de dados válidos dentro de um período de tempo arbitrado na configuração. Para mais informações, consulte o tópico **Aba Setup**. Os valores possíveis são **0 - Comunicação inativa**: O Driver não recebeu dados válidos ou deixou de receber dados depois de *n* milissegundos, conforme configurado na janela de propriedades, ou **1 - Comunicação ativa**: O Driver está recebendo dados válidos.

IO.IOKitEvent

Tipo de Tag	Tag Bloco
Tipo de Acesso	Somente Leitura
Parâmetro B1	-1 (menos um)
Parâmetro B2	0 (zero)
Parâmetro B3	0 (zero)
Parâmetro B4	1 (um)
Propriedade Size	4 (quatro)
Propriedade ParamItem	IO.IOKitEvent

Este Bloco retorna eventos de Driver gerados por várias fontes nas Interfaces de Comunicação. A propriedade **TimeStamp** de um Bloco representa o momento em que um evento ocorre. Os Elementos de Bloco são os seguintes:

- **Elemento 0**: Tipo de evento. Os valores possíveis são **0**: Informação, **1**: Advertência ou **2**: Erro
- **Elemento 1**: Fonte de um evento. Os valores possíveis são **0**: Driver (específico de um Driver), **-1**: IOKit (eventos genéricos da Interface de Comunicação), **-2**: Interface **Serial**, **-3**: Interface **Modem**, **-4**: Interface **Ethernet** ou **-5**: Interface **RAS**
- **Elemento 2**: Número do erro, específico de cada fonte de evento
- **Elemento 3**: Mensagem de um evento, uma **String** específica de cada evento

NOTA

Um Driver mantém um número máximo de 100 eventos internamente. Se eventos adicionais são reportados, os eventos mais antigos são descartados.

IO.PhysicalLayerStatus

Tipo de Tag	Tag de Comunicação
Tipo de Acesso	Somente Leitura
Parâmetro N1	-1 (menos um)
Parâmetro N2	0 (zero)
Parâmetro N3	0 (zero)
Parâmetro N4	2 (dois)
Configuração por String	IO.PhysicalLayerStatus

Este Tag indica o estado da camada física. Os valores possíveis são os seguintes:

- **0**: Camada física parada, ou seja, um Driver está em modo **Offline**, a camada física falhou ao inicializar ou excedeu o número máximo de tentativas de reconexão
- **1**: Camada física iniciada mas não conectada, ou seja, um Driver está em modo **Online**, mas a camada física não está conectada. Se a opção **Connection management** está configurada com o valor **Automatic**, a camada física pode estar conectando, desconectando ou esperando por uma tentativa de reconexão. Se a opção **Connection management** está configurada com o valor **Manual**, então a camada física permanece neste estado até ser forçada a conectar
- **2**: Camada física conectada, ou seja, a camada física está pronta para ser usada. Isto **NÃO** significa que um equipamento esteja conectado, apenas que a camada de acesso está funcionando

IO.SetConfigurationParameters

Tipo de Tag	Tag Bloco
Tipo de Acesso	Somente Leitura
Parâmetro B1	-1 (menos um)
Parâmetro B2	0 (zero)
Parâmetro B3	0 (zero)
Parâmetro B4	3 (três)
Propriedade Size	2 (dois)
Propriedade ParamItem	IO.SetConfigurationParameters

Use este Tag para modificar qualquer propriedade da caixa de diálogo de configuração de um Driver em tempo de execução.

Este Tag funciona somente enquanto um Driver está em modo **Offline**. Para iniciar um Driver em modo **Offline**, selecione a opção **Start driver OFFLINE** na caixa de diálogo de configuração deste Driver. Pode-se tanto escrever em um Tag PLC ou em um Tag Bloco contendo os parâmetros a serem modificados. As escritas de Elementos de Bloco individuais não são suportadas, um Bloco inteiro precisa ser escrito de uma vez só.

No **Elipse SCADA** é necessário usar um Tag Bloco. Cada parâmetro a ser configurado utiliza dois Elementos de Bloco. Por exemplo, caso seja necessário configurar 3 (três) parâmetros, então o tamanho do Bloco deve ser 6 (seis, 3 × 2). O primeiro Elemento é o nome da propriedade, como uma **String**, e o segundo Elemento é o valor desta propriedade, conforme o exemplo a seguir.

```
// 'Block' deve ser um Tag Bloco com leitura automática,
// leitura por varredura e escrita automática desabilitadas.
// Configura os parâmetros
Block.element001 = "IO.Type" // Parâmetro 1
Block.element002 = "Serial"
Block.element003 = "IO.Serial.Port" // Parâmetro 2
Block.element004 = 1
Block.element005 = "IO.Serial.BaudRate" // Parâmetro 3
Block.element006 = 19200
// Escreve o Bloco inteiro
Block.Write()
```

Ao usar o **Elipse E3**, a habilidade de criar *arrays* em tempo de execução permite o uso tanto de um Tag de Comunicação quanto de um Tag Bloco. Pode-se utilizar o método **Write** de um Driver para enviar os parâmetros diretamente para este Driver, sem a necessidade de criar um Tag, conforme o exemplo a seguir.

```
Dim arr(6)
' Configura os elementos do array
arr(1) = "IO.Type"
arr(2) = "Serial"
arr(3) = "IO.Serial.Port"
arr(4) = 1
arr(5) = "IO.Serial.BaudRate"
arr(6) = 19200
' Há dois métodos de enviar os parâmetros
' Método 1: Usando um Tag de Comunicação
tag.WriteEx arr
' Método 2: Sem utilizar um Tag
Driver.Write -1, 0, 0, 3, arr
```

Uma variação do exemplo anterior usa um *array* bidimensional.

```
Dim arr(10)
' Configura os elementos do array. Note que o array foi redimensionado
' para 10 elementos. Elementos vazios são ignorados pelo Driver
arr(1) = Array("IO.Type", "Serial")
arr(2) = Array("IO.Serial.Port", 1)
arr(3) = Array("IO.Serial.BaudRate", 19200)
Driver.Write -1, 0, 0, 3, arr
```

Um Driver não valida nomes de parâmetros ou valores passados, por isto tenha cuidado ao escrever parâmetros e valores. O método **Write** falha se o *array* de configuração é criado incorretamente. Pode-se consultar o log de um Driver ou usar o parâmetro *writeStatus* do método **WriteEx** para descobrir a causa exata de um erro.

```
Dim arr(10), strError
arr(1) = Array("IO.Type", "Serial")
arr(2) = Array("IO.Serial.Port", 1)
arr(3) = Array("IO.Serial.BaudRate", 19200)
If Not Driver.WriteEx -1, 0, 0, 3, arr, , , strError Then
  MsgBox "Falha ao configurar os parâmetros do Driver: " + strError
End If
```

IO.WorkOnline

Tipo de Tag	Tag de Comunicação
Tipo de Acesso	Leitura ou Escrita
Parâmetro N1	-1 (menos um)
Parâmetro N2	0 (zero)
Parâmetro N3	0 (zero)
Parâmetro N4	4 (quatro)
Configuração por String	IO.WorkOnline

Este Tag informa o estado atual de um Driver e permite iniciar ou parar a camada física. Os valores possíveis são os seguintes:

- **0 - Driver Offline**: A camada física está fechada ou parada. Este modo permite uma configuração dinâmica dos parâmetros de um Driver através do Tag **IO.SetConfigurationParameters**
- **1 - Driver Online**: A camada física está aberta ou em execução. Enquanto está em modo **Online**, a camada física pode ser conectada ou desconectada e o estado atual pode ser conferido no Tag **IO.PhysicalLayerStatus**

No exemplo a seguir, utilizando o **Elipse E3**, um Driver é colocado em modo **Offline**, a porta COM é modificada e então é colocado em modo **Online** novamente.

```
'Configura o Driver em modo Offline
Driver.Write -1, 0, 0, 4, 0
'Muda a porta para COM2
Driver.Write -1, 0, 0, 3, Array("IO.Serial.Port", 2)
'Configura o Driver em modo Online
Driver.Write -1, 0, 0, 4, 1
```

O método **Write** pode falhar ao configurar um Driver em modo **Online**, ou seja, escrevendo o valor 1 (um). Neste caso, este Driver permanece em modo **Offline**. A causa da falha pode ser:

- Tipo de camada física configurada incorretamente, provavelmente um valor inválido foi configurado para a propriedade **IO.Type**
- Este Driver pode ter ficado sem memória
- A camada física pode ter deixado de criar a *thread* de trabalho. Procure no arquivo de log pela mensagem "Failed to create physical layer thread!"
- A camada física não conseguiu inicializar. A causa da falha depende do tipo de camada física. Pode ser um número de porta serial inválida, falha ao inicializar o Windows Sockets ou falha ao inicializar o TAPI (modem), entre outras. A causa é gravada no arquivo de log

IMPORTANTE

Mesmo que a configuração de um Driver para o modo **Online** seja bem-sucedida, isto não significa necessariamente que a camada física esteja pronta para uso, ou seja, pronta para executar operações de entrada e saída com um equipamento externo. O Tag **IO.PhysicalLayerStatus** deve ser verificado para assegurar que a camada física esteja conectada e preparada para a comunicação.

Propriedades

Estas são as propriedades gerais de todas as Interfaces de I/O suportadas.

IO.ConnectionMode

9 Controla o modo de gerenciamento da Conexão. Os valores possíveis são **0**: Modo automático, em que um Driver gerencia a conexão ou **1**: Modo manual, em que uma aplicação gerencia a conexão.

IO.GiveUpEnable

◩ Quando configurada para Verdadeiro, define um número máximo de tentativas de reconexão. Se todas as reconexões falharem, um Driver entra em modo **Offline**. Se configurada para Falso, um Driver tenta até que uma reconexão seja bem-sucedida.

IO.GiveUpTries

9 Número de tentativas de reconexão antes que esta seja abortada. Por exemplo, se o valor desta propriedade é igual a 1 (um), um Driver tenta apenas uma reconexão quando a conexão é perdida. Se esta falhar, este Driver entra em modo **Offline**.

IO.InactivityEnable

◩ Configure em Verdadeiro para habilitar e em Falso para desabilitar a detecção de inatividade. A camada física é desconectada se está inativa por um certo período de tempo. A camada física é considerada inativa apenas se é capaz de enviar dados mas não de recebê-los de volta.

IO.InactivityPeriodSec

Número de segundos para a verificação de inatividade. Se a camada física está inativa por este período de tempo, então é desconectada.

IO.RecoverEnable

Configure em Verdadeiro para habilitar um Driver a recuperar conexões perdidas e em Falso para deixar um Driver em modo **Offline** quando uma conexão é perdida.

IO.RecoverPeriodSec

Tempo de espera entre duas tentativas de conexão, em segundos.

NOTA

A primeira reconexão é executada imediatamente após a conexão ser perdida.

IO.StartOffline

Configure em Verdadeiro para iniciar um Driver em modo **Offline** e em Falso para iniciar um Driver em modo **Online**.

NOTA

Não faz sentido modificar esta propriedade em tempo de execução, já que esta só pode ser modificada quando um Driver já está em modo **Offline**. Para configurar um Driver em modo **Online** em tempo de execução, escreva o valor 1 (um) no Tag **IO.WorkOnline**.

IO.TimeoutMs

Define o *time-out* da camada física, em milissegundos. Um segundo equivale a 1000 milissegundos.

IO.Type

Define o tipo de interface física utilizada por um Driver. Os valores possíveis são os seguintes:

- **N ou None**: Não utiliza uma interface física, ou seja, um Driver deve fornecer uma interface personalizada
- **S ou Serial**: Utiliza uma porta serial local (COM*n*)
- **M ou Modem**: Utiliza um modem local, interno ou externo, acessado via TAPI (*Telephony Application Programming Interface*)
- **E ou Ethernet**: Utiliza um *socket* TCP/IP ou UDP/IP
- **R ou RAS**: Utiliza uma Interface **RAS** (*Remote Access Server*). Um Driver conecta-se a um equipamento RAS através da Interface **Ethernet** e então emite um comando **AT** (*dial*)

Configuração de Estatísticas

Esta seção contém informações sobre a configuração dos **Tags de Comunicação** e das **Propriedades** das estatísticas das Interfaces de Comunicação.

Tags de Comunicação

Tags de Estatísticas das Interfaces de Comunicação (N2/B2 = 0)

Os Tags descritos a seguir mostram estatísticas para todas as Interfaces de Comunicação.

IO.Stats.Partial.BytesRecv

Tipo de Tag	Tag de Comunicação
Tipo de Acesso	Somente Leitura
Parâmetro N1	-1 (menos um)
Parâmetro N2	0 (zero)
Parâmetro N3	0 (zero)
Parâmetro N4	1101
Configuração por String	IO.Stats.Partial.BytesRecv

Este Tag retorna a quantidade de bytes recebidos na conexão atual.

IO.Stats.Partial.BytesSent

Tipo de Tag	Tag de Comunicação
Tipo de Acesso	Somente Leitura
Parâmetro N1	-1 (menos um)
Parâmetro N2	0 (zero)
Parâmetro N3	0 (zero)
Parâmetro N4	1100
Configuração por String	IO.Stats.Partial.BytesSent

Este Tag retorna a quantidade de bytes enviados na conexão atual.

IO.Stats.Partial.TimeConnectedSeconds

Tipo de Tag	Tag de Comunicação
Tipo de Acesso	Somente Leitura
Parâmetro N1	-1 (menos um)
Parâmetro N2	0 (zero)
Parâmetro N3	0 (zero)
Parâmetro N4	1102
Configuração por String	IO.Stats.Partial.TimeConnectedSeconds

Este Tag retorna o número de segundos que um Driver está conectado na conexão atual ou 0 (zero) se um Driver está desconectado.

IO.Stats.Partial.TimeDisconnectedSeconds

Tipo de Tag	Tag de Comunicação
Tipo de Acesso	Somente Leitura
Parâmetro N1	-1 (menos um)
Parâmetro N2	0 (zero)
Parâmetro N3	0 (zero)
Parâmetro N4	1103
Configuração por String	IO.Stats.Partial.TimeDisconnectedSeconds

Este Tag retorna o número de segundos que um Driver está desconectado desde o término da última conexão ou 0 (zero) se um Driver está conectado.

IO.Stats.Total.BytesRecv

Tipo de Tag	Tag de Comunicação
Tipo de Acesso	Somente Leitura
Parâmetro N1	-1 (menos um)
Parâmetro N2	0 (zero)
Parâmetro N3	0 (zero)
Parâmetro N4	1001
Configuração por String	IO.Stats.Total.BytesRecv

Este Tag retorna a quantidade de bytes recebidos desde que um Driver foi carregado.

IO.Stats.Total.BytesSent

Tipo de Tag	Tag de Comunicação
Tipo de Acesso	Somente Leitura
Parâmetro N1	-1 (menos um)
Parâmetro N2	0 (zero)
Parâmetro N3	0 (zero)
Parâmetro N4	1000
Configuração por String	IO.Stats.Total.BytesSent

Este Tag retorna a quantidade de bytes enviados desde que um Driver foi carregado.

IO.Stats.Total.ConnectionCount

Tipo de Tag	Tag de Comunicação
Tipo de Acesso	Somente Leitura
Parâmetro N1	-1 (menos um)
Parâmetro N2	0 (zero)
Parâmetro N3	0 (zero)
Parâmetro N4	1004
Configuração por String	IO.Stats.Total.ConnectionCount

Este Tag retorna a quantidade de conexões que um Driver já estabeleceu, com sucesso, desde que foi carregado.

IO.Stats.Total.TimeConnectedSeconds

Tipo de Tag	Tag de Comunicação
Tipo de Acesso	Somente Leitura
Parâmetro N1	-1 (menos um)
Parâmetro N2	0 (zero)
Parâmetro N3	0 (zero)
Parâmetro N4	1002
Configuração por String	IO.Stats.Total.TimeConnectedSeconds

Este Tag retorna o número de segundos que um Driver permaneceu conectado desde que foi carregado.

IO.Stats.Total.TimeDisconnectedSeconds

Tipo de Tag	Tag de Comunicação
Tipo de Acesso	Somente Leitura
Parâmetro N1	-1 (menos um)
Parâmetro N2	0 (zero)
Parâmetro N3	0 (zero)
Parâmetro N4	1003
Configuração por String	IO.Stats.Total.TimeDisconnectedSeconds

Este Tag retorna o número de segundos que um Driver permaneceu desconectado desde que foi carregado.

Propriedades

Atualmente, não existem propriedades definidas especificamente para mostrar as estatísticas das Interfaces de Comunicação em tempo de execução.

Histórico de Revisões do Driver

VERSÃO	DATA	AUTOR	COMENTÁRIOS
1.0.4	28/09/2026	A. Fetzner	• Uma requisição com algum valor do modelo ainda não escrito em um Tag Bloco de Envio passa a ser recusada, com o campo registrado no arquivo de log. Antes esta requisição era enviada com o corpo vazio ou com a palavra-chave do modelo no lugar do valor (*Case 40840*). • Textos com acentos passam a ser enviados e recebidos corretamente, com o corpo de uma requisição no formato **UTF-8**. Antes um valor com acento interrompia a escrita e um valor acentuado recebido da API chegava corrompido a um Tag (*Case 40839*). • A escrita em um Elemento de Bloco além do último item do modelo no formato **JSON** associado passa a falhar com mensagem no arquivo de log deste Driver, ao invés de causar um acesso inválido (*Case 40838*). • Um campo declarado como lista em um modelo de envio no formato **JSON**, como por exemplo "campo": ["E3VAL"], passa a ser enviado como uma lista. Antes o valor apagava esta lista e o campo era enviado como um valor simples (*Case 40837*). • Valores recebidos como texto passam a ser convertidos em número somente quando o texto inteiro é um número. Textos como por exemplo endereços IP (192.168.17.30)

			são entregues como texto (*Case 40818*). • Qualquer resposta HTTP na faixa *2XX* (de 200 a 299) passa a ser tratada como sucesso nas requisições. Antes, apenas o código 200 era aceito (*Case 40306*). • A escrita de um Tag Bloco de Envio inteiro de uma só vez passa a distribuir cada Elemento no parâmetro da respectiva posição, como já ocorria na escrita Elemento a Elemento (*Case 40201*). • Adicionado um novo modo de leitura sem *template* por Tags de Caminho JSON. O parâmetro *Item* deste Tag recebe o caminho do valor no formato **JSON** da resposta com o prefixo $ (cifrão), com suporte a chaves aninhadas, índices, *wildcard*, macros de data e estampa de tempo de um dado (*Case 40406*).
1.0.3	22/12/2025	A. Fetzner	• Correção de falhas no *parsing* dos dados no formato **JSON** e na comunicação usando o protocolo HTTPS (*Case 39116*).
1.0.1	28/10/2025	A. Fetzner	• Versão inicial deste Driver.

Matriz
Rua Mostardeiro, 322/Cj. 902, 1001 e 1002
90430-000 — Porto Alegre — RS
Fone: (+55 51) 3346-4699
Fax: (+55 51) 3222-6226
E-mail: elipse-rs@elipse.com.br

Filial no Paraná
Av. Sete de Setembro, 4698/1708
80240-000 — Curitiba — PR
Fone: (+55 41) 4062-5824
E-mail: elipse-pr@elipse.com.br

Filial no Rio de Janeiro
Av. José Silva de A. Neto, 200/Bl. 4/Sl. 109B
22250-044 — Rio de Janeiro — RJ
Fone: (+55 21) 2430-5912
Suporte Técnico: (+55 21) 2430-5963
E-mail: elipse-rj@elipse.com.br

Filial em São Paulo
Rua dos Pinheiros, 870/Cj. 141 e 142
05422-001 — São Paulo — SP
Fone: (+55 11) 3061-2828
Fax:(+55 11) 3086-2338
E-mail: elipse-sp@elipse.com.br

Filial em Minas Gerais
Rua Antônio de Albuquerque, 156/705
30112-010 — Belo Horizonte — MG
Fone: (+55 31) 4062-5824
E-mail: elipse-mg@elipse.com.br

Filial em Taiwan
9F., No.12, Beiping 2nd St., Sanmin Dist. 807 — Kaohsiung City — Taiwan
Fone: (+886 7) 323-8468
Fax: (+886 7) 323-9656
E-mail: evan@elipse.com.br

Consulte nosso website para informações sobre o representante do seu estado.

www.elipse.com.br
kb.elipse.com.br
forum.elipse.com.br
www.youtube.com/elipsesoftware
elipse@elipse.com.br

Gartner 2014 Cool Vendor

OPC FOUNDATION MEMBER

Gartner, Cool Vendors in Brazil 2014, April 2014.
Gartner does not endorse any vendor, product or service depicted in its research publications, and does not advise technology users to select only those vendors with the highest ratings. Gartner research publications consist of the opinions of Gartner's research organization and should not be construed as statements of fact. Gartner disclaims all warranties, expressed or implied, with respect to this research, including any warranties of merchantability of fitness for a particular purpose.

Microsoft Partner
Gold Independent Software Vendor (ISV)

Copyright © 2025-2026 Elipse Software Ltda. Todos os direitos reservados.