Guideline
ADO.NET Provider for REST

SUMMARY

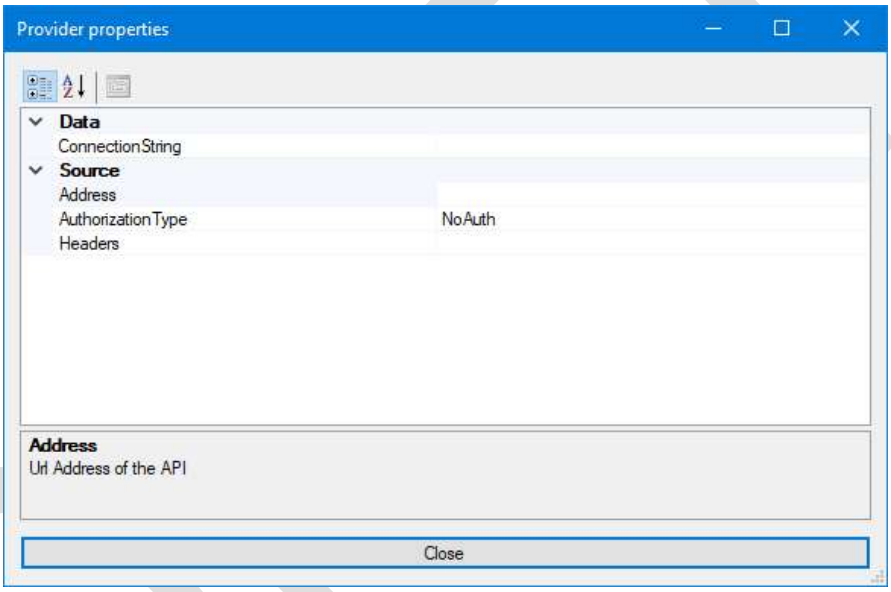
1 Provider REST

This provider can communicate with various REST API.

Supported Authentication:
- API Key
- Bearer token
- BasicAuthentication
- DigestAuthentication
- OAuth2Authentication (client / password)

1.1 Connection string

The configuration parameters of an ADO.NET provider can be automatically discovered by PcVue and displayed into a property grid in AI Explorer.



ADO.NET provider for REST expose all of them with a short description, a category, and a default value.

1.1.1 API category

Name	Description	Type	Allowed values	Default
BaseUrl	Base Url of the API address	String		“”
AuthorizationType	The type of authorization used to connect to the API	Enum	NoAuthorization ApiKey BearerToken BasicAuthentication DigestAuthentication OAuth2Authentication	NoAuthorization
Headers	Comma-separated parameters to add to the http headers of every request.	String		“”

1.1.2 ApiKey category

Name	Description	Type	Allowed values	Default
ApiKeyLocation	Whether to add the API Key to the Header or to the Query Parameters	enum	Header QueryParams	Header
ApiKeyName	Key name	String		“”
ApiKeyValue	Value of the Api Key	String		“”

1.1.3 BearerToken category

Name	Description	Type	Allowed values	Default
BearerToken	API Key value	string		“”

1.1.4 BasicAuthentication category

Name	Description	Type	Allowed values	Default
BasicAuthUsername	Username	String		“”
BasicAuthPassword	Password	String		“”

1.1.5 DigestAuthentication category

Name	Description	Type	Allowed values	Default
DigestAuthUsername	Username	String		“”
DigestAuthPassword	Password	String		“”

1.1.6 OAuth2 category

1.1.6.1 OAuth2Authentication subcategory

Name	Description	Type	Allowed values	Default
OAuth2ClientAuthentication	Location of the authentication data	Enum	Body Header	Body
OAuth2AccessTokenUrl	The URL of the authorization server	String		“”
OAuth2GrantType	Method used to get an Access Token	Enum	Client Password	Client
OAuth2Scope	(Optional – Server-specific) Values defining the scope of the Access Token	String		“”
OAuth2ClientID	ID of a client registered in the API	String		“”
OAuth2ClientSecret	Client secret generated by the API	String		“”

1.1.6.1.1 Password

These parameters are only visible when OAuth2GrantType is set to Password.

Name	Description	Type	Visible	Allowed values	Default
OAuth2Username	Username	String	Only if OAuth2GrantType is set to Password		“”
OAuth2Password	Password	String	Only if OAuth2GrantType is set to Password		“”

1.1.6.2 OAuth2ResponseProperties subcategory

Name	Description	Type	Allowed values	Default
OAuth2AccessTokenName	Name of the Access Token in the authentication query response	String		“access_token”
OAuth2RefreshTokenName	Name of the Refresh Token in the authentication query response	String		“refresh_token”

1.1.6.3 OAuth2TokenProperties subcategory

Name	Description	Type	Allowed values	Default
OAuth2AccessTokenLocation	Location of the Access Token	Enum	Header Url	Header
OAuth2AccessTokenPrefix	Prefix of the Access Token	String		“Bearer” if token location is set to “Header” “access_token” if token location is set to “Url”

1.2 Commands

1.2.1 Generalities

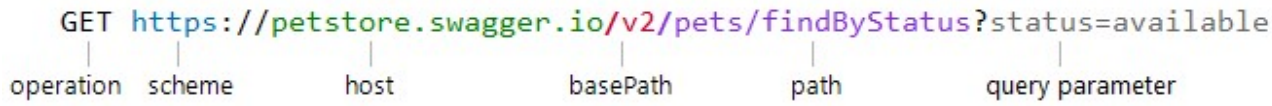
The Sql language understood by our provider imposes some rules to be respected.

1. **Boolean value** is 0 and 1 or true and false (not 'true' and 'false')
2. **Double value** format is 1234.5 (not 1234,5 or 1,234.5)
3. **DateTime format** is always the same '2019/11/18 16:00:00'
4. **Nullable value** is null (not 'null')

1.2.2 Data selection syntax

API Path is always relative to Base URL (set in connection string). This API Path is always enclosed by square brackets in query.
[<API_Path>]

Base URL is scheme+host+basePath



⚠ When a field name contains a special character like a dot, it's necessary to use brackets.

To use the right REST Method (GET, PUT, POST, DELETE, PATCH)

Just prefix original API path with "get_", "put_", "post_", "delete_", "patch_")

Example:
SELECT [device.name] FROM [get_v1/deviceMgt/devices]
EXEC [put_v1/deviceMgt/groups/IbN89a]
SELECT * FROM [post_v1/deviceMgt/devices]
EXEC [delete_v1/deviceMgt/groups/IbN89a]
EXEC [patch_v1/alarm/ack/IbN89a]

1.2.3 SELECT instruction

Keyword	Syntax
SELECT	*
	Column1, [Column 2], [Column.3.x] …
	Column1 AS Alias
	[Column.2] AS Alias
	Column1 AS [Ali as]
	[Column.2] AS [Ali as]
	TOP(n)
	DISTINCT
	AVG()
	SUM()
	COUNT()
	MIN()
	MAX()
	STDEV()
	VAR()
FROM	[patch_v1/alarm/ack/IbN89a]
WHERE	=
	<>
	>
	<
	<=
	>=
	AND
	OR
	IS NULL
	IS NOT NULL
	IN
	NOT IN
	LIKE
	NOT LIKE
	Column – 1 = 0
	Column + 1 = 0
	Column * 1 = 0
	Column / 1 = 0
	Column % 2 = 0
	Column1 + Column2 = 120
HAVING	AVG()
	SUM()
	COUNT()
	MIN()
	MAX()
	STDEV()
	VAR()
GROUP BY	ColumnName
	Alias
ORDER BY	ColumnName
	Alias
	ASC
	DESC

	Column1 ASC, Column2 DESC
JSONPATH	JSONPATH expression used to filter the JSON.
MAXDEPTH	To limit the JSON exploration depth (nested JSON). . Default value or missing parameter = first depth . Max value is 0 = all depth
IGNOREKEY	To ignore a key in JSON.
REMOVEKEY	To suppress a key in JSON.
QUERYSTRING	Parameter necessary to query API separated by & Ex: querystring(”endMonth=2021-10&startMonth=2021-12”)
HEADER	Parameter necessary to query API separated by & Ex: header(”appid:128”,”version:2”)
BODYCONTENT	A string to define the body content Ex: “application/json” “application/x-www-form-urlencoded”
BODY	Parameter with JSON format or other. Ex: body(”{“Name”:”MyGroup”, “ID”:118}”) body(”Name=MyGroup”, “ID=118”)
HTTPSTATUSCODE	To get status code of query in answer. Only success status (200 to 299) Ex: HttpStatusCode(“on”) to activate HttpStatusCode(“off”) or missing to desactivate

Expression order

```
SELECT
{
   <query_expression>
   {  FROM {<table_source>}}

   [QUERYSTRING( "<query>")]
   [HEADER( "<header>")]
   [BODYCONTENT( "<bodycontent>")]
   [BODY( "<body>")]

   [JSONPATH ( "<expression>" )]
   [MAXDEPTH ( <value> )]
   [IGNOREKEY ( "<json_key1>", "<json_key2>", "<json_key3>, …" )]
   [REMOVEKEY ( "<json_key1>", "<json_key2>", "<json_key3>", … )]
   [HTTPSTATUSCODE("on")]

   [WHERE <search_condition>]
   [GROUP BY <group_by_clause>]
   [HAVING <search_condition>]
   [ORDER BY <order_by_expression>]
}
```

Examples:

Queries with query parameters

```
SELECT * FROM [get_message] QUERYSTRING("chat_id=abcd")

SELECT * FROM [get_weather] QUERYSTRING("lat=42&lon=-56")
```

Query with body

```
SELECT * FROM [post_message] QUERYSTRING("chat_id=abcd") BODYCONTENT("application/json")
BODY("{"text":"Hello!"}")
```

Query with header

```
SELECT * FROM [get_message] HEADER("Accept: application/json")
```

1.2.4 SELECT instruction and JSON PATH

The JSON Path provides a simple way to extract specific JSON data from a file. The JSON Path is to JSON format what XPATH is to XML format.

It can be useful for client to filter result and improve query performance instead using SQL filters on a complex JSON file.

1.2.4.1 JSON Path expression

JSONPath expressions always refer to a JSON structure in the same way as XPath expression are used in combination with an XML document. Since a JSON structure is usually anonymous and doesn't necessary have a "root member object" JSONPath assumes the abstract name $ assigned to the outer level object.

JSONPath expressions can use the dot notation.

```
SELECT * FROM file JSONPATH("$.people.[0].vehicles")
```

or the bracket notation

```
SELECT * FROM file JSONPATH("$['people'][0]['vehicles']")
```

JSONPath allows the wildcard symbol * for member names and array indices. It borrows the descendant operator '..' and the array slice syntax proposal [start:end:step].

Expressions of the underlying scripting language (<expr>) can be used as an alternative to explicit names or indices as in $.store.book[(@.length-1)] using the symbol '@' for the current object.

Filter expressions are supported via the syntax ?(<boolean expr>) as in $.store.book[?(@.price < 10)].

Here is a complete overview and a side by side comparison of the JSONPath syntax elements with its XPath counterparts.

JSONPath	Description

$	the root object/element
@	the current object/element
. or []	child operator
..	recursive descent. JSONPath borrows this syntax from E4X.
*	wildcard. All objects/elements regardless their names.
[]	subscript operator. XPath uses it to iterate over element collections and for predicates. In Javascript and JSON it is the native array operator.
[,]	Union operator in XPath results in a combination of node sets. JSONPath allows alternate names or array indices as a set.
[start:end:step]	array slice operator borrowed from ES4.
?()	applies a filter (script) expression.

1.2.4.2 Examples

JSON file content (store.json)

```
{
  "Stores": [
    "Lambton Quay",
    "Willis Street"
  ],
  "Manufacturers": [
    {
      "Name": "Acme Co",
      "Products": [
        {
          "Name": "Anvil",
          "Price": 50
        }
      ]
    },
    {
      "Name": "Contoso",
      "Products": [
        {
          "Name": "Elbow Grease",
          "Price": 99.95
        },
        {
          "Name": "Headlight Fluid",
          "Price": 4
        }
      ]
    }
  ]
}
```

Provider do not support explicit reference to a JSON field value.

1.2.4.2.1 File content as array

```
SELECT * FROM [stores.json] MAXDEPTH(10)
```

Stores (System.String)	Manufacturers.Name (System.String)	Manufacturers.Products.Name (System.String)	Manufacturers.Products.Price (System.Double)
Lambton Quay	Acme Co	Anvil	50
Lambton Quay	Contoso	Elbow Grease	99,95
Lambton Quay	Contoso	Headlight Fluid	4
Willis Street	Acme Co	Anvil	50
Willis Street	Contoso	Elbow Grease	99,95
Willis Street	Contoso	Headlight Fluid	4

1.2.4.2.2 Get all the products priced 50 and above

```
SELECT * FROM [stores.json] JSONPATH("$..Products[?(@.Price >= 50)]")
```

Name (System.String)	Price (System.Double)
Anvil	50
Elbow Grease	99,95

1.2.4.2.3 Get name of all the products priced 50 and above

~~SELECT * FROM [stores.json] JSONPATH("$..Products[?(@.Price >= 50)].Name")~~
No because provider do not support explicit reference to a JSON field value (here name)

```
SELECT Name FROM [stores.json] JSONPATH("$..Products[?(@.Price >= 50)]")
```

Name (System.String)
Anvil
Elbow Grease

1.2.4.2.4 Get name of products priced 50 and more without JSON PATH

```
SELECT DISTINCT [Manufacturers.Products.Name] as Name FROM [stores.json] WHERE
[Manufacturers.Products.Price]>=50
```

Name (System.String)
Anvil
Elbow Grease

1.2.4.2.5 Some other samples with JSONPATH

```
select * from [stores.json] jsonpath("$.Manufacturers[?(@.Name == 'Acme Co')]")
MAXDEPTH(0)
```

Name (System.String)	Products.Name (System.String)	Products.Price (System.Int64)
Acme Co	Anvil	50

```
select * from [stores.json] jsonpath("$.Manufacturers[:1]") MAXDEPTH(10)
```

Name (System.String)	Products.Name (System.String)	Products.Price (System.Int64)
Acme Co	Anvil	50

```
select * from [stores.json] jsonpath("$..Manufacturers[*]") MAXDEPTH(10)
```

Name (System.String)	Products.Name (System.String)	Products.Price (System.Double)
Acme Co	Anvil	50
Contoso	Elbow Grease	99,95
Contoso	Headlight Fluid	4

```
select Name from [stores.json] jsonpath("$..Manufacturers[*]")
```

Name (System.String)
Acme Co
Contoso
Contoso

```
select * from [stores.json] jsonpath("$..Products[0]")
```

Name (System.String)	Price (System.Double)
Anvil	50
Elbow Grease	99,95

```
select * from [stores.json] jsonpath("$.Manufacturers[0,2]") MAXDEPTH(10)
```

Name (System.String)	Products.Name (System.String)	Products.Price (System.Int64)
Acme Co	Anvil	50

```
select * from [stores.json] jsonpath("$.Manufacturers[1:2]") MAXDEPTH(10)
```

Name (System.String)	Products.Name (System.String)	Products.Price (System.Double)
Contoso	Elbow Grease	99,95
Contoso	Headlight Fluid	4

```
select * from [stores.json] jsonpath("$..Products[?(@.Name == 'Elbow Grease')]")
```

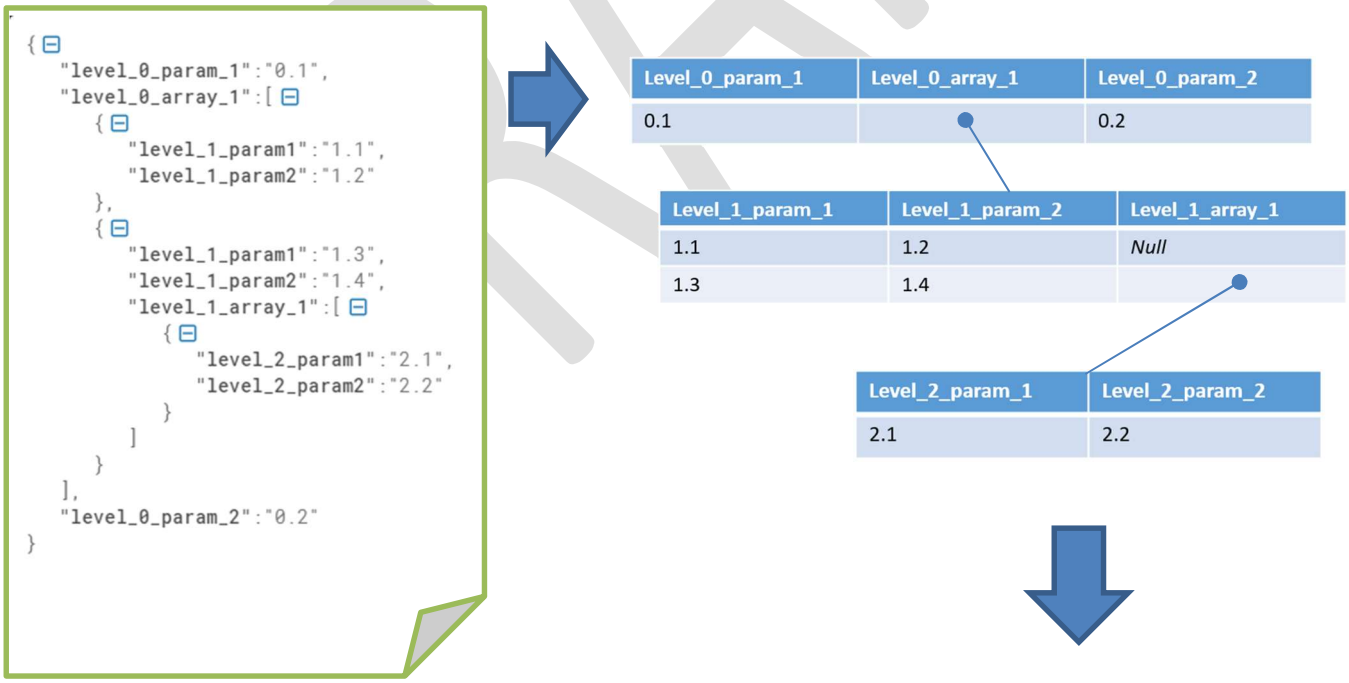
Name (System.String)	Price (System.Double)
Elbow Grease	99,95

1.2.5 SELECT instruction and nested JSON

To convert a nested JSON to a table, a hierarchical join is apply as shown on following sample if sub property is an array.

The same principle is applied whatever the depth of the document.

Nested.json



level_0_param_1 (System.String)	level_0_param_2 (System.String)	level_0_array_1.level_1_param1 (System.String)	level_0_array_1.level_1_param2 (System.String)	level_0_array_1.level_1_array_1.level_2_param1 (System.String)	level_0_array_1.level_1_array_1.level_2_param2 (System.String)
0.1	0.2	1.1	1.2	null	null
0.1	0.2	1.3	1.4	2.1	2.2

1.2.5.1 MAXDEPTH keyword

This parameter is useful when you want to limit the parsing depth of a nested json.
. If parameter is missing, default depth is 1.
. Max value is 0 to parse all depth.

⚠ The query performance is directly impacted by this value.

```
SELECT * FROM [nested.json]
```
or
```
SELECT * FROM [nested.json] MAXDEPTH(1)
```

	level_0_param_1 (System.String)	level_0_param_2 (System.String)	level_0_array_1 (System.String)
▶	0.1	0.2	{ "level_1_param1": "1.1", "level_1_param2": "1.2" }
	0.1	0.2	{ "level_1_param1": "1.3", "level_1_param2": "1.4", "level_1_array_1": [{ "level_2_param1": "2.1", "level_2_param2": "2.2" }] }

```
SELECT * FROM [nested.json] MAXDEPTH(2)
```

level_0_param_1 (System.String)	level_0_param_2 (System.String)	level_0_array_1.level_1_param1 (System.String)	level_0_array_1.level_1_param2 (System.String)	level_0_array_1.level_1_array_1 (System.String)
0.1	0.2	1.1	1.2	*null*
0.1	0.2	1.3	1.4	{ "level_2_param1": "2.1", "level_2_param2": "2.2" }

. Special use with value = -1
MaxDepth keywork can also be use to NOT PARSE query result.
In this mode query result is return as a raw response (json, html, xml, plain text…)

```
SELECT * FROM [nested.json] MAXDEPTH(-1)
```

raw_response (System.String)
{ "level_0_param_1":"0.1", "level_0_array_1":[{ "level_1_param_1":"1.1", "level_1_param_2":"1.2" }, { "level_1_param_1":"1.3",

⚠ Of course this special mode disabled some keywords as:
IGNOREKEY, REMOVEKEY, JSONPATH, WHERE, GROUP BY, HAVING, ORDER BY

1.2.5.2 IGNOREKEY keyword

This parameter is useful when you want to limit the parsing of JSON String. By giving a list of JSON key, it's possible to limit query execution time by converting key value (and sub-values) to a simple JSON string.

```
SELECT * FROM [nested.json] MAXDEPTH(10) IGNOREKEY("level_1_array_1")
```

level_0_param_1 (System.String)	level_0_param_2 (System.String)	level_0_array_1.level_1_param1 (System.String)	level_0_array_1.level_1_param2 (System.String)	level_0_array_1.level_1_array_1 (System.String)
0.1	0.2	1.1	1.2	*null*
0.1	0.2	1.3	1.4	[{ "level_2_param1": "2.1", "level_2_param2": "2.2" }]

To specify multiple key to ignore use the following syntaxe:
```
SELECT * FFROM [file] IGNOREKEY("first_key", "seconde_key", "etc")
```

1.2.5.3 REMOVEKEY keyword

This parameter is useful when you want to limit the parsing of JSON String. By giving a list of JSON key, it's possible to limit query execution time by removing key value (and sub-values).

```
SELECT * FROM [nested.json] MAXDEPTH(10) REMOVEKEY("level_0_array_1")
```

	level_0_param_1 (System.String)	level_0_param_2 (System.String)
▶	0.1	0.2
	0.1	0.2

To specify multiple key to ignore use the following syntaxe:

```
SELECT * FROM [file] REMOVEKEY("first_key", "seconde_key", "etc")
```

1.2.5.4 HTTPSTATUSCODE keyword

To get **success** Http Status Code in query answer.

```
SELECT * FROM [get_weather] QueryString("q=paris&appid=8e9da87d14ea6") MaxDepth(0)
HttpStatusCode("on") where [httpstatuscode] = 200
```

	HttpStatusCode (System.Int32)	base (System.String)	visibility (System.Int64)	dt (System.Int64)
▶	200	stations	10000	1674027896

```
SELECT [httpstatuscode] FROM [get_weather] QueryString("q=paris&appid=8e9da87d14ea6")
HttpStatusCode("on")
```

	httpstatuscode (System.Int32)
▶	200

Reminder about existing http Status Code:

. 1xx – Informational responses
The server is thinking through the request.

. 2xx – Success
The request was successfully completed, and the server gave the expected response.

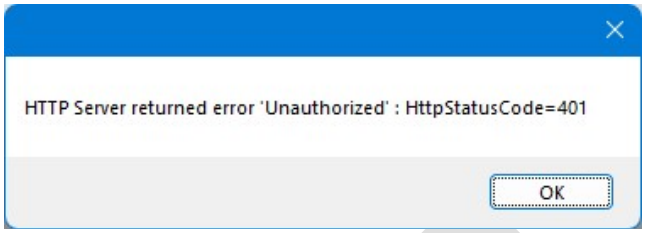
. 3xx – Redirection
You got redirected somewhere else. The request was received, but there's a redirect of some kind.

. 4xx – Client errors
Page not found. The site or page couldn't be reached. (The request was made, but the page isn't valid — this is an error on the server's side of the conversation and often appears when a page doesn't exist on the site)

. 5xx – Server errors
A valid request was made by the client but the server failed to complete the request.

⚠ If it's necessary to get status code other than 2xx in PcVue it's possible to use error return by the provider through the ErrorVariable.

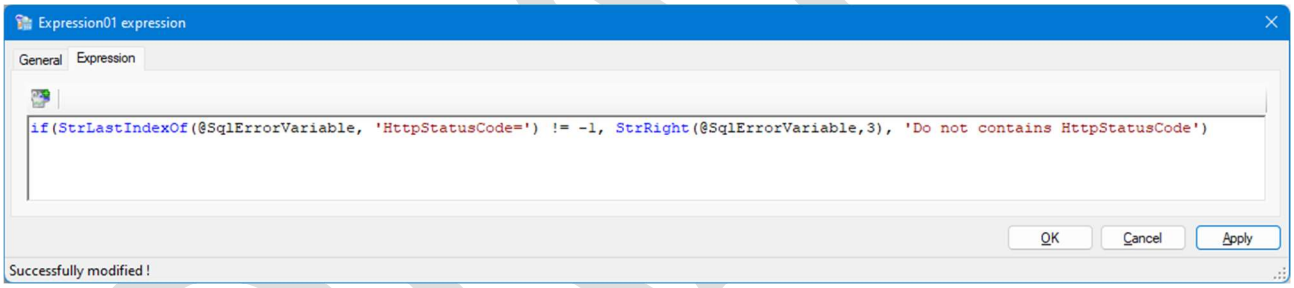
<Human readable message> **: HttpStatusCode=**<Status Code Number>



Sample of expression based on ErrorVariable Message to extract HttpStatusCode value.

Result as text

```
if(StrLastIndexOf(@SqlErrorVariable, 'HttpStatusCode=') != -1,
StrRight(@SqlErrorVariable,3), 'Do not contains HttpStatusCode')
```

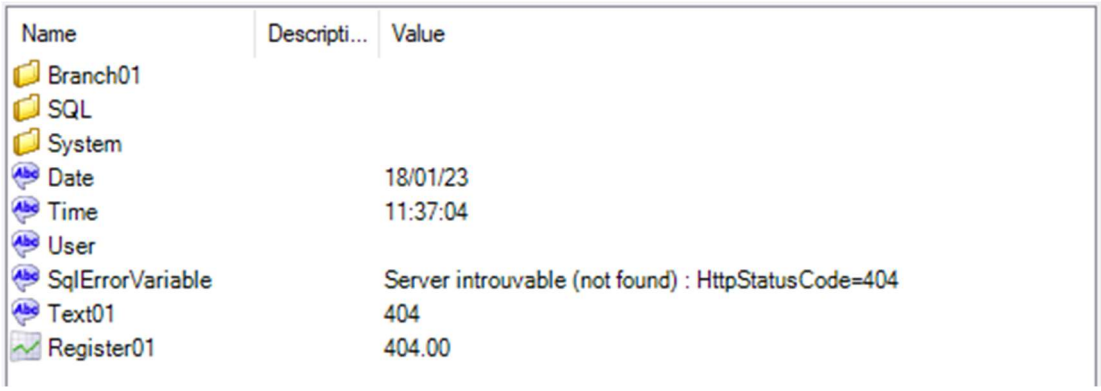


Result as register

```
ToDbl(if(StrLastIndexOf(@SqlErrorVariable, 'HttpStatusCode=') != -1,
StrRight(@SqlErrorVariable,3), 'Do not contains HttpStatusCode'))
```

Note: ToDbl is not mandatory but more explicit.

Here is the result of these two expressions.



1.2.6 INSERT/UPDATE/DELETE instruction

This feature seems do not make sense with REST API. General REST API allow to read data with GET query and modify values with POST/PUT/PATCH and DELETE but most of them return a text value to validate the action. That why, it's necessary to use SELECT instruction for all of them.

If result is useless, it's possible to use EXEC instruction.

1.2.7 EXEC instruction

EXEC instruction is like SELECT instruction, but no result value is return. It can be useful with PcVue write query.

Keyword	Syntax
EXEC	post_v2/devices
QUERYSTRING	Parameter necessary to query API separated by & Ex: querystring("endMonth=2021-10&startMonth=2021-12")
HEADER	Parameter necessary to query API separated by & Ex: header("appid:128","version:2")
BODYCONTENT	A string to define the body content Ex: "application/json" "application/x-www-form-urlencoded"
BODY	Parameter with JSON format or other. Ex: body("{"Name":"MyGroup", "ID":118}") body("Name=MyGroup, ID=118")

Expression order

```
EXEC
{
```

```
  { < source> }

  [QUERYSTRING( "<query>")]
  [HEADER( "<header>")]
  [BODYCONTENT( "<bodycontent>")]
  [BODY( "<body>")]
}
```

Example:

```
EXEC [put_service] BODYCONTENT("application/json") BODY("{"status":true}")

EXEC [post_message] QUERYSTRING("chat_id=abcd") BODYCONTENT("application/json")
BODY("{"text":"Hello!"}")
```

1.3 Limitations

1.3.1 Executing limitations

Provider ADO.NET shall be only used by SV32 and DbConnect.

1.3.2 JSON content

1.3.2.1 Multidimensional array

Multidimensional array is considered as string.

Exemple:

Original
"colorgamut":[[0.6915,0.3083],[0.1700,0.7000],[0.1532,0.0475]]

Return by provider
"colorgamut": "[[0.6915,0.3083],[0.1700,0.7000],[0.1532,0.0475]]"

1.3.2.2 Empty JSON Key values

Json key is remove from array result when all key values are null (or missing).

Example with Json Key "Extra" always null or missing:

```
{
    "Manufacturers": [
        {
          "Name": "Acme Co",
          "Extra": null
        },
        {
          "Name": "Contoso",
          "Extra": null
        },
        {
          "Name": "Barosko",
        }
   ]
}
```

	Manufacturers.Name (System.String)
▶	Acme Co
	Contoso
	Barosko

Example with Json Key “Extra” null or not null:

```
{
    "Manufacturers": [
        {
          "Name": "Acme Co",
          "Extra": null
        },
        {
          "Name": "Contoso",
          "Extra": 10
        },
        {
          "Name": "Barosko",
        }
   ]
}
```

	Manufacturers.Name (System.String)	Manufacturers.Extra (System.Int64)
▶	Acme Co	*null*
	Contoso	10
	Barosko	*null*