REST

Tagged Values

<<E2ERESTService>>

Stereotype <<E2ERESTService>> is used in the component diagram to mark a service as REST service.

Tagged Value	Description	Allowed Values	
Port (port)	Defines the machine port number the service is binding to. This port number can be given at service level only.	any number Using ports below 1024 may require additional privileges.	
Trace Port (tracePort)	Defines the shadow port of the service used for tracing.	any number (default is service port + 40000)	
Max Request Body Size (maxRequestBodySize)	Runtime 2021.2 Specifies the maximum size of the request in KB (1 KB = 1024 Bytes). This can be used to prevent DoS or similar attacks. When the payload of the service exceeds the given maximum, incoming request are rejected.	any positive integer	
		0	Accept unlimited requests (default of services compiled with Builder versions < 7.12.0).
		2048	Builder 7.12.0 2MB, default if not specified
Max Request Header Size (maxRequestHeaderSize)	Runtime 2022.6 Specifies the maximum size of the request header in KB (1 KB = 1024 Bytes). This can be used to prevent DoS or similar attacks. When the header payload of the service exceeds the given maximum, incoming request are rejected. **Compatibility Hint** For older Runtimes, a limit of 8 KB applies.	any positive integer	
		8	8 KB (default if not specified).
Http Header Roles (httpHeaderRoles)	Runtime 2020.12 Builder 7.12.0 Assign roles to dedicated incoming headers that define how the related header will be treated by the xUML Runtime. These definitions overwrite the default behavior (see REST Adapter > HTTP Headers), and **X-Transaction-Id**, **X-Request-Id**, **X-Sender-Host** and/or **X-Sender-Service** will be substituted by this definition. Refer to HTTP Header Support > Overwriting the Standard HTTP Headers for more details. **Http Header Roles** can hold a list of definitions in format `<http header name>:<role>`, where `<role>` can be one of the listed allowed values (one list entry per line).	client_host	Take the sender host from header `<http header name>` instead of **X-Sender-Host**.
		client_service	Take the sender service from header `<http header name>` instead of **X-Sender-Service**.

On this Page:

- Tagged Values
  - <<E2ERESTService>>
  - <<E2ERESTPortType>>
  - <<RESTResource>>
  - <<REST>>
  - <<RESTParameter>>
  - <<RESTOperationTag>>
  - <<RESTError>>
  - <<RESTResponseDefinition>>
  - <<RESTAlias>>
- REST Content Types
- REST Adapter Parameters
- REST Utility Functions
- REST Parameter Types
  - Request
  - Response
  - RequestOptions
  - AdapterResponse
  - Request and Response Types

Related Pages:

- Defining a REST Service Interface
- Implementing REST Methods
- REST Adapter
- HTTP Header Support
- xUML Service Settings

		correlation_id	Take the correlation ID from header <http header name> instead of **X-Request-Id**.
		transaction_id	Take the transaction ID from header <http header name> instead of **X-Transaction-Id**.
Token Type (tokenType)	Mark the service as to use token authorization. The Bridge REST Test Tool will then present a field to enter the token and put the value into the HTTP headers. Refer also to **tokenHeaderName** for more information. You can use both in a REST service: basic authorization and token authorization, see **useBasicAuth**.	none	Do not use token authorization.
		APIKey	Use API key authorization.
Token Header Name (tokenHeaderName)	Defines the name of the header that will transport the token. This tagged value is only relevant, if token authorization is enabled at all. The token header name will be presented as the name of the header field that can be entered in the Bridge REST Test Tool. Refer also to **tokenType** for more information.	A valid HTTP header name (according to RFC2616 /RFC7230).	
Use Basic Auth (useBasicAuth)	Mark the service as to use basic authentication mechanisms. The Bridge REST Test Tool will then present fields to enter the credentials and put the values into the HTTP headers. You can use both in a REST service: basic authorization and token authorization, see **tokeType** and **tokenHeaderName**.	true	Enable basic authentication.
		false	Disable basic authentication (default).
Json Keep Nulls (jsonKeepNulls)	When **jsonKeepNulls** is true, attributes of the REST response object having NULL values will be rendered to the REST response, otherwise they will be left out completely (see also chapter NULL Values).	true	Render attributes with NULL values to the REST response.
		false	Leave out attributes with NULL values in the REST response (default).
Json Compact (jsonCompact)	When **jsonCompact** is true, the JSON composer will generate compact JSON, otherwise it will generate pretty JSON. **jsonCompact** defaults to true - also on re-compile of an older model with Builder as of 7.0.0-beta3.	**true**	Generate compact JSON (default).
		false	Generate pretty JSON.
Json Write Type Discriminator (jsonWriteTypeDiscriminator)	Runtime 2021.6 Builder 7.15.0 When **jsonCompact** is true, the JSON composer will generate xUML type properties ("e2e:type") to the generated JSON. If this option is true, the Runtime will write the original xUML type to the generated JSON in form of `"e2e:type": "<name of the original xUML type>"` if the type being serialized does not match the expected metadata. This is necessary if you want to convert the generated JSON back to an xUML class using jsonToClass(). Runtime versions before 2021.6 will ignore the value.	**true**	Write xUML type discriminator.
		false	Do not write xUML type discriminator.

<<E2ERESTPortType>>

Stereotype <<E2ERESTPortType>> is used on a class to mark it as REST port type, the root element of a REST service structure.

Tagged Value	Description	Allowed Values	
Path (path)	Defines the path to this rest interface. If empty, the path is derived from the package structure.	none	path of the package structure will be used, e.g. `/Services /SupportCase/SupportAPI`
		any valid path string	path string starting with "/", e.g. `/ support`
Error Class (errorClass)	Assigns a user-defined <<RESTError>> class to the REST interface. This class should be set in case of error and given back via the REST response.	any complex type describing the structure of the error	
Api Version (apiVersion)	Defines the API version this port type provides (for documentation purposes only).	any string	

For more information on REST error classes, see <<RESTError>>.

<<RESTResource>>

Stereotype <<RESTResource>> is used on a class to mark it as REST resource, part of a REST service structure.

Tagged Value	Description	Allowed Values	
Relative Path (relativePath)	Defines the path of the REST resource or collection in relation to the parent resource . You can provide a static path, or a dynamic path using the notation `:<name of a REST Parameter>`. You may also provide a combination of both.	none	the name of the REST resource will be used, e.g. `/supportcases`
		any valid string	the given name will be used
		a dynamic path supplying a REST parameter	dynamic path, the value of the REST parameter will be passed to the REST methods, e.g. `:id`

<<REST>>

Stereotype <<REST>> is used on a <<RESTResource>> class method to mark it as REST method, part of a REST service structure.

> <<REST>> is the stereotype to apply to a REST method. Do not confuse with <<RESTOperation>>, which is used for RESTful HTTP services as described on RESTful HTTP Service.
> The latter approach is recommended only, if you want to use content types different to JSON and XML.

If the method name is one of GET, POST, PUT, DELETE, PATCH, HEAD, OPTIONS (with optional trailing '/'), it will be invoked automatically on its parent resource when an corresponding request is received.

Refer Implementing REST Methods to for more details and some examples.

Tagged Value	Description	Allowed Values	
Http Method (httpMethod)	Provide the HTTP method of this REST method should respond to.	a valid HTTP method	GET, POST, PUT, DELETE, PATCH, HEAD, OPTIONS

		none	• method name, if it is one of: GET, POST, PUT, DELETE, PATCH, HEAD, OPTIONS (with optional trailing '/') • GET otherwise
Relative Path (relativePath)	Defines the path of the REST method in relation to the parent resource.	none	The name of the REST method will be used.
		any valid string	The given name will be used. The relative path may also contain variables (REST path parameters, specified as `:<variable name>`) and can be segmented like e.g. `/date=:<a date variable>`.
Is Verbatim Path (isVerbatimPath)	This a REST Adapter setting and has no effect on REST service.		
Blob Body Content Type (blobBodyContentType)	Specify a default content type for **Blob** response parameters from this endpoint. This must be a list of valid media ranges as defined in RFC 7231. This information will be generated to the OpenAPI descriptor file (response content type). Refer to Handling Blobs in the REST Interface for a deeper explanation and some examples. This tag must be left unset if no **Blob** output parameters are used. In future versions, the effect of this tag may be extended to other contexts as well.	a list of valid media ranges	e.g. `application/msexcel` Default is `application/octet-stream` if not specified.
Reject Other Response Content Type (rejectOtherResponseContentTypes)	Runtime 2021.6 Builder 7.15.0 The xUML Runtime performs a verification of the content-type header for REST responses. Specify whether to return an error (HTTP 406, not acceptable) on responses with a content type that does not conform with the content types specified in **Blob Body Content Type**. Any mismatch will be logged to the service log on log level **Debug**. Refer to Handling Blobs in the REST Interface for a deeper explanation and some examples.	**true**	• Return HTTP 406 (Not Acceptable, default). • Service log: `RESTLM/47: Client does not accept any of declared response content types` This exception can be suppressed by setting **Ignore Http Errors** to true on the REST adapter alias.
		false	• Accept the request in spite of the mismatch and handle this within the service. • Service log (**Debug**): `RESTLM/10: Cannot generate any of the expected output formats`
Accepted Request Content Type (acceptedRequestContentType)	Runtime 2021.6 Builder 7.15.0 Provide a list of content types this REST endpoint accepts. This must be a list of valid media ranges as defined in RFC 7231. This information will be generated to the OpenAPI descriptor file (parameter content type). Refer to Handling Blobs in the REST Interface for a deeper explanation and some examples. This tag must be left unset if no **Blob** output parameters are used. In future versions, the effect of this tag may be extended to other contexts as well.	a list of valid media ranges	e.g. `application/xhtml+xml` Default is `application/octet-stream` if not specified.

Reject Other Request Content Types (rejectOtherRequestContentTypes)	Runtime 2021.6 Builder 7.15.0 Specify whether to return an error on requests with a content type that does not conform with the content types specified in **Accepted Request Content Type**. Any mismatch will be logged to the service log on log level **Debug**. Refer to Handling Blobs in the REST Interface for a deeper explanation and some examples.	**true**	• Return HTTP 415 (Unsupported Media Type) if the request content type of a **Blob** input parameter does not match the requirements (default). • Service log (**Debug**): `RESTLM/10: Cannot generate any of the expected output formats` This exception can be suppressed by setting **Ignore Http Errors** to true on the REST adapter alias.
		false	• Perform the adapter call in spite of the "content-type" header mismatch and handle this within the service. • Service log: `RESTLM/48: Request content type not declared as accepted by the service`

<<RESTParameter>>

Stereotype <<RESTParameter>> is used on a <<REST>> method parameter to mark it as REST parameter. Refer to REST Parameters to for more details and some examples.

Tagged Value	Description	Allowed Values		Allowed REST Methods	Allowed Types	Hints and Limitations
External Name (externalName)	Defines an external name for the REST parameter	any string				Use this, when wanting to access a REST service that has parameter names with special characters. In this case, set this name (e.g. `ugly@parameter-name`) to **externalName** and give a better name. So you will not have to escape the parameter every time you use it.
In (in)	Defines how the parameter will be passed to the REST method. This tag is **mandatory**.	query	via a query string	all	all simple types and **Array** of simple type	Unknown parameters will be ignored, known will be passed to the method after being URL-decoded.
		path	via the REST resource path	all	**Integer**, **Float**, **String**, **Boolean**, **DateTime**	Path parameters are all required. All path parameters must be consumed by the called method and the parameter names must be the same as the path segment identifiers (without colon).
		body	via the REST call body	POST, PUT, PATCH	a complex type and **Array**	A REST method can have only one body parameter.
		header	via the REST call header	all	all simple types and **Array** of simple type	Unknown parameters will be ignored, known will be passed to the method.
Multiplicity (multiplicity)	Defines whether the parameter is required, or not.	0..1				Parameter is not required. Path parameters are always required.
		1				Parameter is required.

<<RESTOperationTag>>

With <<RESTOperationTag>> you can group your REST methods. Refer to Tagging REST Operations for more details.

Tagged Value	Description		Allowed Values
Name (name)	Defines the name of the tag. This name will be displayed in the Bridge REST Test Tool as a group heading.		any string
Description (description)	You can add a short description of the tag that will be displayed in the Bridge REST Test Tool together with the heading.		any string
External Documentation Description (externalDocumentationDescription)	You can add a short description of the documentation.	These field values will be generated to the OpenAPI descriptor, but are not displayed on the Test UI at the moment.	any string
External Documentation URL (externalDocumentationURL)	Defines a documentation URL for this tag group.		a valid URL
Order (order)	Defines the order in which the tag groups will be displayed on the screen. Tag groups with empty order will be displayed last.		any number

<<RESTError>>

Stereotype <<RESTError>> is used on a class to mark it as REST error class. Assign such a class to the REST port type (see <<E2ERESTPortType>>) and this class will be used as output in case of error.
Each REST port type can have its separate error class.
You can report errors back to the caller using something like:

```
local response = getRestHttpResponse();
response.responseObject = <my error object>;
response.httpStatus = <a matching http error code>;
```

<<RESTResponseDefinition>>

Use dependencies with stereotype <<RESTResponseDefinition>> are used to connect REST resources with REST error classes.

Tagged Value	Description	Allowed Values	
Name (name)	Specify an HTTP status code. For this status code, the default error class will be overwritten by the specific error class.	a specific HTTP status code	e.g. 401
		a pattern	e.g. 40? or 4??
		all status codes	???
Blob Body Content Type (blobBodyContentType)	Specify a default content type for **Blob** parameters from this endpoint. This information will be generated to the OpenAPI descriptor file and will will set the the "Content-Type" header to this content type.	a valid MIME-type	e.g. `application/msexcel` Default is `application/octet-stream` if not specified.

<<RESTAlias>>

Attribute	Description	Allowed Values
Additional Headers (additionalHeaders)	This tagged value can contain a list of additional headers in form of name/value pairs.	Valid format is: <name>:<value>, e.g. `API-Key:e2e`. Separate multiple headers with a comma.
Base Path (basePath)	Specify here the base path of the REST service.	a valid path, e.g. `/support`
Protocol (protocol)	Specify here the protocol through which the REST service is accessible.	**http**, https

Ignore Http Errors (ignoreHttpErrors)	Specify here whether you want the REST adapter to throw an exception upon receiving an HTTP error code >= 400. For older models, if this flag is not present, it will be considered **false**. **ignoreHttpErrors** can be overridden via the request options (see Setting REST Request Options).	**true** (default)	Do not throw an exception upon receiving an HTTP error code >= 400.
		false	Throw an exception upon receiving an HTTP error code >= 400.
Host (host)	Specify here the host running the REST service.	a valid host	
Port (port)	Specify here the port through which the REST service is accessible.	a valid port	
Follow Redirects (followRedirects)	Specify here the maximum number of redirects to follow. Default value is 0 (no redirects).	any integer	
Options (options)	Specify native cURL options as listed in Setting cURL Options on the URL Adapter . Use one of the following syntax rules: • values separated by `','` in one line • values separated by `' '` in one line • list of tagged values		
Json Keep Nulls (jsonKeepNulls)	When **jsonKeepNulls** is true, attributes of the REST parameter having NULL values will be provided with the REST call, otherwise they will be left out completely (see also chapter NULL Values).	true	Render attributes with NULL values to the REST call.
		false	Leave out attributes with NULL values in the REST call (default).
Json Compact (jsonCompact)	When **jsonCompact** is true, the JSON composer will generate compact JSON, otherwise it will generate pretty JSON. **jsonCompact** defaults to true - also on re-compile of an older model with Builder as of 7.0.0-beta3.	**true**	Generate compact JSON (default).
		false	Generate pretty JSON.
Request Http Header Roles (requestHttpHeaderRoles)	Builder 7.12.0 Runtime 2020.12 In the context of HTTP based adapters (URL, REST, SOAP), enable automatic header generation for the listed headers. These definitions overwrite the default behavior, and **X-Transaction-Id**, **X-Request-Id**, **X-Sender-Host** and/or **X-Sender-Service** will be substituted by this definition. **requestHttpHeaderRoles** can hold a list of definitions in format `<http header name>:<role>`, that will automatically be generated for each adapter call on this alias. `<role>` can be one of the listed allowed values (one list entry per line). Refer to HTTP Header Support > Overwriting the Standard HTTP Headers for more details on header roles.	client_host	Provide the client host in a header `<http header name>` instead of **X-Sender-Host**.
		client_service	Provide the client service in a header `<http header name>` instead of **X-Sender-Service**.
		correlation_id	Provide the correlation ID in a header `<http header name>` instead of **X-Request-Id**.
		transaction_id	Provide the transaction ID in a header `<http header name>` instead of **X-Transaction-Id**.
		passthrough	Pass a present header `<http header name>` to the called service.
		passthrough= `<request header name>`	Pass an present header `<request header name>` to the called service under the name of `<http header name>`. This is equivalent to renaming a header.

<table>
<tr><td>Digest Algorithm (digestAlgorithm)</td><td>Runtime 2021.1 Builder 7.12.0 Generates a HTTP digest header using the specified algorithm. When applied, a digest header is generated using the specified algorithm, and sent with the request . The generated header conforms with RFC3230 and RFC5843.

Only one value is supported (no multi-value header).</td><td>None</td><td>No header generated.</td></tr>
<tr><td></td><td></td><td>MD5</td><td>Generate header using MD5 algorithm.</td></tr>
<tr><td></td><td></td><td>SHA</td><td>Generate header using SHA algorithm.</td></tr>
<tr><td></td><td></td><td>SHA-1</td><td>Generate header using SHA-1 algorithm.</td></tr>
<tr><td></td><td></td><td>SHA-256</td><td>Generate header using SHA-256 algorithm.</td></tr>
<tr><td></td><td></td><td>SHA-512</td><td>Generate header using SHA-512 algorithm.</td></tr>
<tr><td>User (user)</td><td>Specify credentials here, if the called REST service needs basic authentication. Other authentication algorithms have to be implemented manually via HTTP headers (see additionalHeaders and Setting REST Request Options).</td><td colspan="2">Valid format is <user>/<password>, e.g. <code>e2e/e2e</code></td></tr>
<tr><td colspan="4">Proxy Settings (if the called REST service is accessed via a proxy)</td></tr>
<tr><td>Proxy Type

(proxyType)</td><td>Specify the proxy type.</td><td colspan="2">See CURLOPT_PROXYTYPE.</td></tr>
<tr><td>Proxy URL (proxyURL)</td><td>Specify the URL of the proxy server.</td><td colspan="2">See CURLOPT_PROXY.</td></tr>
<tr><td>Proxy User

(proxyUser)</td><td>Specify the proxy credentials.</td><td colspan="2">See CURLOPT_PROXYUSERPWD, valid format is <user>/<password>, e.g. <code>e2e/e2e</code></td></tr>
<tr><td colspan="4">SSL Settings (if the called REST service uses SSL)</td></tr>
<tr><td>Ssl CA Info

(sslCAInfo)</td><td>Specify a file name containing additional certificates for the connection verification (e.g. additional root CAs).</td><td colspan="2">See CURLOPT_CAINFO.</td></tr>
<tr><td>Ssl Certificate File (sslCertificateFile)</td><td>Specify a fle name containing the client certificate.</td><td colspan="2">See CURLOPT_SSLCERT.</td></tr>
<tr><td>Ssl Certificate Type (sslCertificateType)</td><td>Specify the type of the certificate.</td><td colspan="2">See CURLOPT_SSLCERTTYPE.</td></tr>
<tr><td>Ssl Private Key File (sslPrivateKeyFile)</td><td>Specify a file name containing the private key.</td><td colspan="2">See CURLOPT_SSLKEY.</td></tr>
<tr><td>Ssl Private Key Password (sslPrivateKeyPassword)</td><td>Specify the password for the private key.</td><td colspan="2">See CURLOPT_KEYPASSWD.</td></tr>
<tr><td>Ssl Private Key Type (sslPrivateKeyType)</td><td>Specify the type of the key.</td><td colspan="2">See CURLOPT_SSLKEYTYPE.</td></tr>
<tr><td>Ssl Verify Host (sslVerifyHost)</td><td>Specify whether to verify the host information form the SSL connection.</td><td colspan="2">See CURLOPT_SSL_VERIFYHOST.</td></tr>
<tr><td>Ssl Verify Peer (sslVerifyPeer)</td><td>Specify whether to verify the peer information from the SSL connection.</td><td colspan="2">See CURLOPT_SSL_VERIFYPEER.</td></tr>
</table>

REST Content Types

The Bridge handles content types as follows:

- If the **Content-Type** header is set, the Bridge will assume that the request is of that type. All other request will be rejected (HTTP error code **406**).
- If no **Content-Type** header is set, but the **Accept** header is, the Bridge will deduce the request type from it. All other request will be rejected (HTTP error code **406**).
  To determine the format, the Bridge will take into account the quality factor and the order of the accept headers list.
- In absence of both headers, the Bridge will assume JSON. All other request will be rejected (HTTP error code **415**).

The full matching is done in a "best effort" manner. Given the type format `<type>/<subtype> [+<suffix>][; paramName=paramValue]*`, the Bridge first disregards the type and parameters. Then it checks, if the subtype is JSON or XML. If the subtype doesn't match with the supported types, it tries the suffix.

REST Adapter Parameters

Name	Type	Direction	Description
requestOptions	Request Options	in	Use this parameter to configure the REST Adapter dynamically and overwrite the settings from the component diagram.
response	Any	out	This parameter holds the adapter output and is of that type that is given back by the called REST service.

REST Utility Functions

Access to HTTP request and response objects is provided through global methods: `getRestHttpRequest()`and `getRestHttpResponse()`.

Function	Parameter	Return Value	Description	Example
getRestHttpRequest()	none	object of type Request	Returns the request details as provided by the HTTP call. Changing the request object will not have any effects.	`local request = getRestHttpRequest();`
getRestHttpResponse()	none	object of type Response	Set the response details to return them to the caller.	`local response = getRestHttpResponse();`

`getRestHttpRequest()` will contain the headers in REST service context. In other contexts, e.g. if called via a SOAP shadow port (and thus SOAP context), `getRestHttpRequest()` will return `NULL`. Use `getServiceContext()` or `getServiceContextValue()` in such cases.

REST Parameter Types

Request

Attribute	Type	Description	Values /Example
method	HTTPMethod	HTTP method used in call.	`GET`
headers	Array of HeaderField	DeprecatedThis attribute is deprecated as of Runtime 2020.11. Please use **httpHeaderMap** (see below) for new implementations as its implementation complies to the HTTP specification. All HTTP request header fields as an array of HeaderField classes containing name/value pairs. The header fields contain the standard HTTP headers as well as header parameters, if provided.	
queryString	String	Query string, if provided with the call.	`status=in%20progress`

queryParameters	Array of Parameter	All query parameters as an array of Parameter classes containing name/value pairs.	
body	Blob	Body of the HTTP request.	
path	String	Path to the REST resource.	`/support/supportcases/`
pathParameters	Map	All REST parameters as a map.	
httpHeaderMap	Map of Entry	Runtime 2020.11 Header information as a map. The map contains arrays of header value strings whereas the header name is the key of the map. • Header names are lowercase and treated case insensitive. • Multiple headers with the same name are treated as arrays. Refer to HTTP Header Support for more information on the standard xUML HTTP headers.	

Response

Attribute	Type	Description	Values /Example
headers	Array of HeaderField	All HTTP response header fields as an array of HeaderField classes containing name/value pairs.	
statusCode	Integer	The resulting HTTP status code. If not set explicitly using this object, the service returns 200 if no exception occurred, or 500 otherwise.	`404`
errorObject	Any	Object of the type defined with stereotype <<RESTError>>.	

RequestOptions

Attribute	Type	Description	Values /Example
additionalHeaders	Array of HeaderField	All REST request header fields as an array of HeaderField classes containing name/value pairs.	
options	Array of Option	Specify native cURL options as listed in Setting cURL Options on the URL Adapter . Use one of the following syntax rules: • values separated by `','` in one line • values separated by `' '` in one line • list of tagged values	
ssl	SSL	Use this parameter to supply SSL information.	
proxy	Proxy	Use this parameter to supply necessary proxy information.	
additionalQueryParameters	Array of Parameter	Use this parameter to provide additional query parameters to the REST service call.	
followRedirects	Integer	Specify here the maximum number of redirects to follow.	any integer
basicAuth	Authentication	This parameter provides an object of type **Authentication** containing the user and the password.	
basePath	String	Overwrite here the base path of the REST service.	a valid path, e.g. `/support`
host	String	Overwrite here the host running the REST service that has been defined in the component diagram.	
port	Integer	Overwrite here the port through which the REST service is accessible.	
protocol	String	Overwrite here the protocol through which the REST service is accessible.	**http**, https
ignoreHttpErrors	Boolean	If true, HTTP error codes >= 400 will not cause an exception in the model. This implies, that the response body is accessible even if HTTP errors occur. Default value is true.	**true** / false

jsonComposerOptions	ComposerOptions	Use this parameter to specify JSON composer options on the REST call. You can use these options to e.g. overwrite **jsonKeepNulls** from the REST alias.	

AdapterResponse

Attribute	Type	Description	Values /Example
httpStatus	Integer	HTTP status code of the adapter call.	500
headers	Array of HeaderField	DeprecatedThis attribute is deprecated as of Runtime 2020.11. Please use **httpHeaderMap** (see below) for new implementations as its implementation complies to the HTTP specification. HTTP headers of HTTP response.	
body	Blob	HTTP body of HTTP response.	
responseObject	Any	Response Object of the REST adapter call. • If the adapter had an error, the **responseObject** is an <<RESTError>> class. It could be the default error class or a specific error class dependent on the <<RESTResponseDefinition>> dependencies and the HTTP status code. • If the adapter call had no error, the **responseObject** is the same as the **response** output parameter.	
httpHeaderMap	Map of Entry	Runtime 2020.11 Header information as a map. The map contains arrays of header value strings whereas the header name is the key of the map. • Header names are lowercase and treated case insensitive. • Multiple headers with the same name are treated as arrays. Refer to HTTP Header Support for more information on the standard xUML HTTP headers.	

Request and Response Types

REST Type	Attribute	Type	Description	Values/Example
Authentication	username	String	Username.	
	password	String	Password.	
Certificate	file	String	Certificate file.	
	type	String		
ComposerOptions	keepNulls	Boolean	Keep `NULL` values during JSON composing.	
Entry	key	String	Key of the map entry.	
	value	Array of Any	List of values of the map entry. The dynamic type for **httpHeaderMap** is **String**.	
HTTPMethod		enumeration	List of all valid HTTP methods	`DELETE, GET, HEAD, OPTIONS, PATCH, POST, PUT`
HeaderField	name	String	Name of the header field.	
	value	String	Value of the header field.	
Option	name	String	Name of the option.	
	value	String	Value of the option.	
Parameter	name	String	Name of the parameter.	
	value	String	Value of the Parameter.	
Proxy	url	String	URL.	A valid URL.
	type	String		
	authentication	Authentication	See above.	

SSL	verifyPeer	String		
	verifyHost	String		
	caInfo	String		
	certificate	Certificate	See above.	
	key	Key		