REST Adapter Reference

This page explains the **REST Adapter** in Bridge context. If you were looking for the same information regarding the PAS Designer, refer to REST Adapter in the Designer guide.

Tagged Values

<<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	client_host	Provide the client host in a header `<http header name>` instead of **X-Sender-Host**.

On this Page:

Related Pages:

- REST Adapter
- REST Service

	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_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.
Digest Algorithm (digestAlgorithm)	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).	None	No header generated.
		MD5	Generate header using MD5 algorithm.
		SHA	Generate header using SHA algorithm.
		SHA-1	Generate header using SHA-1 algorithm.
		SHA-256	Generate header using SHA-256 algorithm.
		SHA-512	Generate header using SHA-512 algorithm.
User (user)	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).	Valid format is <user>/<password>, e.g. `e2e/e2e`	
Proxy Settings (if the called REST service is accessed via a proxy)			
Proxy Type (proxyType)	Specify the proxy type.	See CURLOPT_PROXYTYPE.	
Proxy URL (proxyURL)	Specify the URL of the proxy server.	See CURLOPT_PROXY.	
Proxy User (proxyUser)	Specify the proxy credentials.	See CURLOPT_PROXYUSERPWD, valid format is <user>/<password>, e.g. `e2e/e2e`	
SSL Settings (if the called REST service uses SSL)			
Ssl CA Info (sslCAInfo)	Specify a file name containing additional certificates for the connection verification (e.g. additional root CAs).	See CURLOPT_CAINFO.	
Ssl Certificate File (sslCertificateFile)	Specify a fle name containing the client certificate.	See CURLOPT_SSLCERT.	

Ssl Certificate Type (sslCertificateType)	Specify the type of the certificate.	See CURLOPT_SSLCERTTYPE.
Ssl Private Key File (sslPrivateKeyFile)	Specify a file name containing the private key.	See CURLOPT_SSLKEY.
Ssl Private Key Password (sslPrivateKeyPassword)	Specify the password for the private key.	See CURLOPT_KEYPASSWD.
Ssl Private Key Type (sslPrivateKeyType)	Specify the type of the key.	See CURLOPT_SSLKEYTYPE.
Ssl Verify Host (sslVerifyHost)	Specify whether to verify the host information form the SSL connection.	See CURLOPT_SSL_VERIFYHOST.
Ssl Verify Peer (sslVerifyPeer)	Specify whether to verify the peer information from the SSL connection.	See CURLOPT_SSL_VERIFYPEER.

<<RESTAdapter>>

Attribute	Description	Allowed Values	
alias	Specify the REST alias the adapter should connect to.	a valid REST alias	
digestAlgorithm	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).	None	No header generated.
		MD5	Generate header using MD5 algorithm.
		SHA	Generate header using SHA algorithm.
		SHA-1	Generate header using SHA-1 algorithm.
		SHA-256	Generate header using SHA-256 algorithm.
		SHA-512	Generate header using SHA-512 algorithm.

<<RESTInterface>>

This stereotype defines a package as REST interface. It has no additional tagged values.

<<RESTResource>>

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>>

Since Runtime 2022.6 the Runtime uses **Transfer-encoding: chunked** for POST requests.

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)	Disable most of the path normalization. All escaping must be done manually, leading or trailing whitespaces are preserved. This is what still will be done, if **isVerbatimPath** is set to true: • **variable discovery** The E2E Runtime will try to find and replace variables (`:<my variable name>`). • **colon escaping** If the path contains a colon that (and that does not indicate a path parameter), you can escape this colon by another one, like `::` (e.g. `/myAPI /resource/aText::anotherText/:variable`). • **correction of segment delimiters** The E2E Compiler adds a leading slash (if not present), removes trailing slashes (if present) and removes duplicates slashes, e.g. `myAPI //resource/:variable/` becomes `/myAPI /resource/:variable`.	true	Path should be treated as verbatim, path normalization is disabled.
		false (default)	Path should be URL encoded.
Blob Body Content Type (blobBodyContentType)	Specify the content type (as defined in RFC 7231) that the request will be sending. The Runtime generates a matching "Content-Type" header to your request. 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** . Refer to Handling Blobs in the REST Interface for a deeper explanation and some examples.	**true**	• Reject to perform adapter call if the header "content-type" does not match the values listed in **Blob Body Content Type** (default) . • Exception: `RESTLM/48: Request content type not declared as accepted by the service`
		false	Perform the adapter call in spite of content-type header mismatch, and let the service handle this.
Accepted Request Content Type (acceptedRequestContentType)	Runtime 2021.6 Builder 7.15.0 Provide a list of content types you accept as a response. This must be a list of valid "accept" headers as defined in RFC 7231. The Runtime generates a matching "Accept" header to your request. 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** . Refer to Handling Blobs in the REST Interface for a deeper explanation and some examples.	**true**	• Reject to perform adapter call if the header "accept" does not match the values listed in **Accepted Request Content Type** (default). • Exception: `Set "accept" header does not accept any of declared response content types`
		false	Perform the adapter call in spite of accept header mismatch and let the service handle this.

<<RESTParameter>>

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.

<<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.

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 Content Types

With the REST Adapter, the E2E Runtime can handle JSON or XML as content types. The E2E Runtime will parse either response content (JSON or XML) to a response object automatically.
You do not need to set any headers. If no headers are set, the Runtime will use JSON as a default format and set **Accept** header to `application/json,text/json,application/xml;q=0.9,text/xml;q=0.9,*/*;q=0.8`.

If you want to control the REST request and response formats, you can set the **Content-Type** and **Accept** headers as described on Setting REST Request Options.

- Set the **Accept** header to the content type, you want the REST service to send back as a response.
  - `application/json` for JSON
  - `application/xml` for XML

  The E2E Runtime will deduce the **Content-Type** header accordingly.

This mechanism also works vice versa: setting the **Content-Type** header and letting the E2E Runtime deduce the **Accept** header:

- Set the **Content-Type** header to the content type, you want to send to the REST service.
  - `application/json` for JSON
  - `application/xml` for XML

  The E2E Runtime will deduce the **Accept** header accordingly to
  - `application/json,text/json,application/xml;q=0.9,text/xml;q=0.9,*/*;q=0.8` for JSON
  - `application/xml,text/xml,application/json;q=0.9,text/json;q=0.9,*/*;q=0.8` for XML

  This approach gives the REST service a wider range of content types for sending the response.

Setting the content type headers to any value other than JSON or XML is not supported and the E2E Runtime will throw an exception in this case.

Regarding response parsing, the E2E Runtime will process REST responses as follows:

- If the **Content-Type** header of the response is JSON or XML, respective format will be used (regardless what is specified in the **Accept** header).
- If the Content-Type header of the response is not set, the E2E Runtime will assume that the content matches the **Accept** header of the request. If no **Accept** header has been specified, JSON is the fallback.
- In case of unsupported **Content-Type**, JSON is the fallback.
  If the log level of the service is set to **Debug**, the E2E Bridge will log details about unsupported **Content-Types** to the bridgeserver log. Refer to xUML Service Standard Log for more information on the bridgeserver log.

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		