Using the URL Adapter with the HTTP Protocol

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

On this Page:

- Static GET Request
- Static POST Request
- Full Dynamic GET
- Tagged Values
- URL Adapter Response
- HTTP Headers
  - Overwriting Default Headers

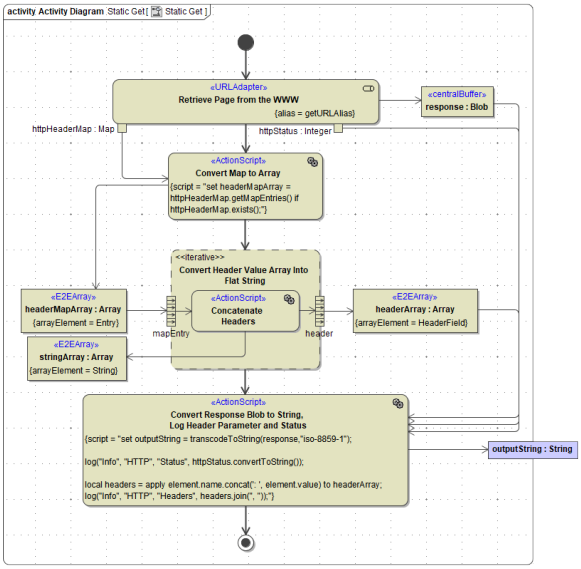
Related Pages:

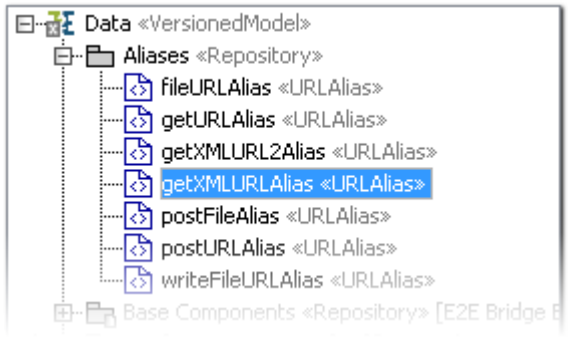
- Setting cURL Options on the URL Adapter
- URL Adapter Reference

Static GET Request

The following example shows how to access a web page through the <<URLAdapter>>. The page URL is derived from the alias (details see below).

To make the HTML page readable in a conventional text editor or Web browser we convert it to a string in the second step using the transcodeToString() Operation.



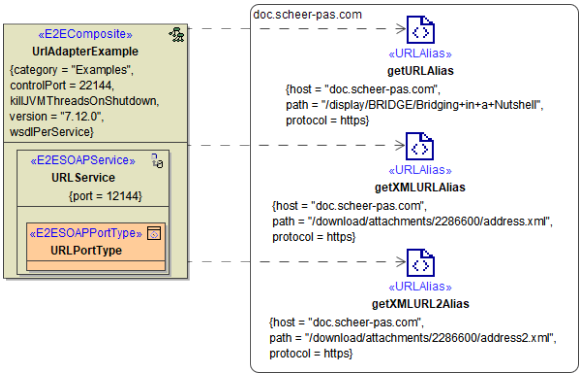


As the <<URLAdapter>> is a backend for the Bridge, the necessary connection information is done in the component diagram. The link to this diagram is done with an alias (which is an artifact with the stereotype <<Alias>>), which has to be defined by the developer. The name of this alias can be chosen freely.

We suggest to store all aliases in package **Aliases**.

The following figure shows the corresponding backend part of the component diagram for the above GET request:

Figure: Component Diagram for Static GET request



The SOAP service has a dependency to the backend alias. The backend delivers the requested HTML page. The parameters for the HTTP request are defined as tagged values on the <<URLAlias>>.
In the example above the following GET request will be sent to e.g.: **http://www.e2ebridge.com:80 /BridgeFiles/bridging_in_a_nushell.htm**

Find below a list of all tagged values that can be used with the HTTP protocol.

Static POST Request

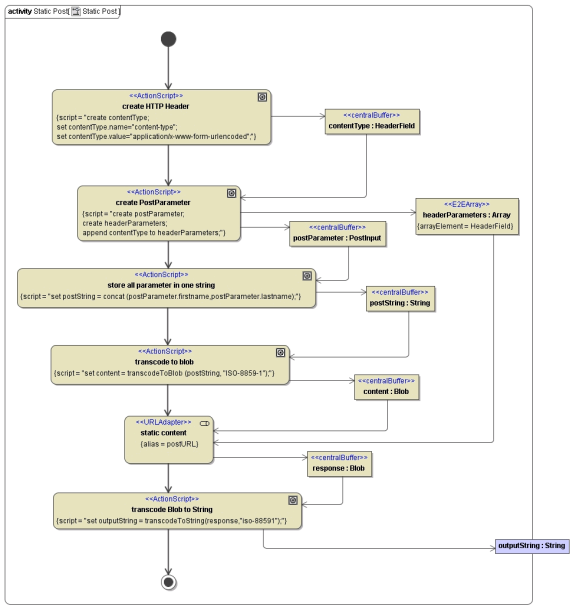
To define a POST request we have the same structure and stereotypes as previously described for the GET request. There are basically two differences. The protocol defined in the component diagram has to be "POST" and the post parameters have to be defined in the activity diagram. In some cases the HTTP-Header has to be adjusted, otherwise the Bridge defaults are used.
The activity diagram below shows the definition of the key/value pairs which can then be sent via POST to the host defined in the deployment diagram. Because the input of the <<URLAdapter>> is a blob that must be named "content" this parameter definition has to be made in several steps.

The first step creates the HTTPHeaderField class, which overwrites the content-type default value to application/x-www-form-urlencoded. With this statement, the post-request will look like it would be sent via a Web Browser.

The second step creates the post parameter and appends the key-value pairs together with the manipulated contentType to the headerParameters array.

The third step concats all input values in one string that will be transcoded to a blob in the next action state. This transcoding is necessary because of the interface of the <<URLAdapter>>, which requires Blobs as input and output parameter. The action state with the stereotype <<URLAdapter>> has no script entries but a tagged value with the alias that makes the link to the deployment diagram (like described in the GET request).

The last step transcodes the base64 output (in this case a .html page) to a string which is sent back to the client.

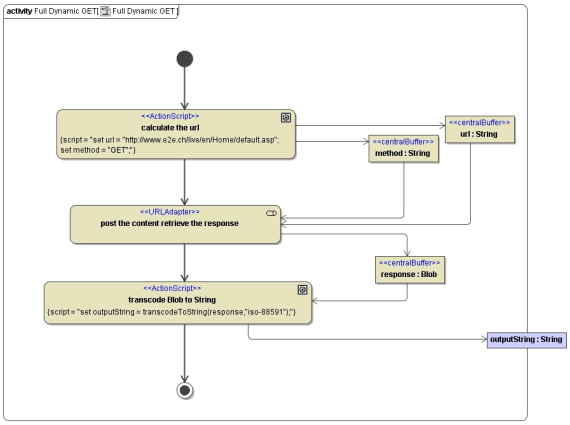


All object names except outputString in the last step must be named exactly like in the diagram above because only these input parameters are valid for the <<URLAdapter>>.

Full Dynamic GET

It is also possible to not have an alias pointing to the host and not to have a component diagram. This is useful when all parameters are dynamic e.g. all parameter are read from a database. The following request creates the URL dynamically which means that you do not have an alias pointing to the component diagram where the parameters are defined.

Figure: Get Request Without Deployment Diagram



However, the absence of a component diagram where the backend is well documented should be the exception, and only used if all backend parameters are dynamic.

Tagged Values

Find below a list of relevant tagged values, if the URL adapter is used with the HTTP protocol. Default values used when an option is not explicitly set are written in bold.

Tagged Value	Description	Values
protocol	Transport protocol.	**http**, https
method	HTTP method.	**get**, post, put
port	Machine port number the service is binding to. This port number can be given at service level only.	**80**
path	HTTP path for the request.	
Advanced		
followRedirects	Maximum number of redirects to follow.	any integer
options	Native cURL options.	cURL Options
Authentication		
user	Username/password.	
Proxy		
proxyType	Type of the proxy.	HTTP, SOCKS5
proxyURL	URL of the proxy server.	
proxyUser	Proxy user.	
SSL		
sslCAInfo	File name containing additional certificates for the connection verification (e.g. additional root CAs).	
sslCertificateFile	File name containing the client certificate.	
sslCertificateType	Type of the certificate.	PEM, DER

sslPrivateKeyFile	File name containing the private key.		
sslPrivateKeyPassword	Password for the private key.		
sslPrivateKeyType	Type of the key.		
sslVerifyHost	Whether to verify the host information form the SSL connection.	**On**	Verification on.
		Off	Verification off.
sslVerifyPeer	Whether to verify the peer information from the SSL connection.	**On**	Verification on.
		Off	Verification off.

URL Adapter Response

The adapter returns the following parameters:

Name	Type	Direction	Restrictions		Description
			to listed protocol only	**to listed method only**	
response	Blob	out		get, post, put, list, read	Contains the response content in relation to the used method.
httpStatus	Integer	out	http, https		Contains the HTTP status code of the response.
httpHeaderParameter	Array of HeaderField	out	http, https		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. Contains the HTTP headers of the response.
httpHeaderMap	Map of Entry	out	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.

HTTP Headers

Runtime 2019.9 With xUML service adapter calls, the xUML Runtime adds the following outgoing HTTP headers containing correlation information to the request:

- **X-Transaction-Id** or **xTransactionId** (in JMS context)
  This header identifies the transaction the call belongs to. You can set the transaction id manually with setTransactionID. If not set, the Runtime will generate one.
  This header will be passed through the callstack to identify all service calls that belong to a transaction.
- **X-Request-Id**
  This header identifies the unique request. The Runtime generates a unique number for each adapter call.
- **X-Sender-Host** and **X-Sender-Service**
  These headers contain the sender host resp. the sender service. They are set by the Runtime automatically.

Transaction id and request id will be logged to the transaction log on the adapter call. Having this information, you can use this for error analysis or usage metrics.

Overwriting Default Headers

Builder 7.12.0 Runtime 2020.12 You can overwrite this default behavior by own header role definitions as described on HTTP Header Support > Overwriting the Standard HTTP Headers. In this context, you can also enable automatic header generation for dedicated headers.To do this, specify a list of header generation rules in tag **requestHttpHeaderRoles** on the URL alias.

requestHttpHeaderRoles can hold a list of definitions in format `<http header name>:<role>`. The listed headers will automatically be generated with the specified role for each adapter call on this alias. 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.
Refer to URL Adapter Reference for the list of allowed values.