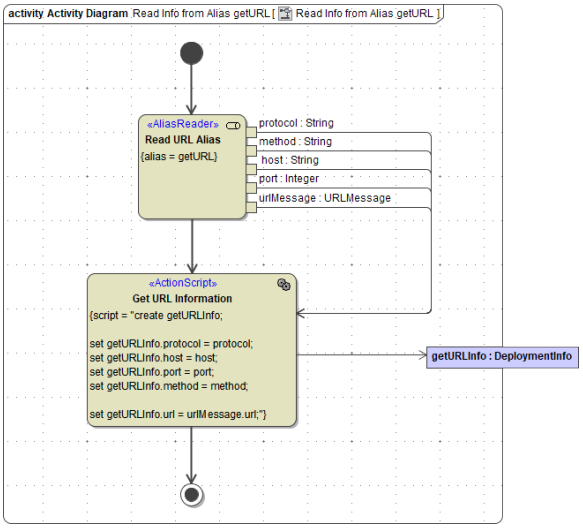
Alias Reader

Builder 7.5.0 Runtime 2019.5 The Alias Reader enables you to get the details that have been specified on any alias from the component diagram. This adapter replaces the deployment macros get... FromAlias().

Example File (Builder project E2E Action Language/Operating):

<your example path>\E2E Action Language\Operating\uml\deploymentMacros.xml

You can now get the alias details with an action node with stereotype <<AliasReader>>.



Depending on the type of alias, the Alias Reader action returns different output parameters, e.g. **host**, **port**, **protocol**, ... for an URL alias.

Find below a list of aliases and their output parameters.

Only these aliases will work with the Alias Reader.

<<JMSAlias>>

Tagged Value	Description	Allowed Values / Example	
acknowledgeMode	Specify the message acknowledge mode. We recommend using acknowledge mode **transacted** .	Auto	Each single message sent to a JMS provider will be acknowledged by the JMS provider (not the recipient) after recipience. Messages received from a JMS provider within an activity are acknowledged irrespective of subsequent activities. Consequently, if an error occurs during the execution of the activity diagram after message receipt, no rollback occurs. Using Auto acknowledge mode in a model, a client must be prepared for possible loss of messages.

On this Page:

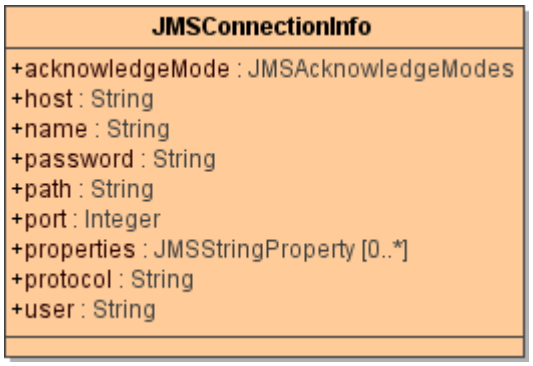
- <<JMSAlias>>
  - Additional Pins
- <<POP3Alias>>
- <<REST Alias>>
  - Additional Pins
- <<SAPAlias>>
- <<SMTPAlias>>
- <<SOAPAlias>>
  - Additional Pins
- <<SQLAlias>>
- <<URLAlias>>
  - Additional Pins

Related Pages:

- Deployment Macros
- xUML Service Adapters

		Duplicate	**Duplicate** acknowledge mode corresponds to **Auto** acknowledge mode. Additionally, the JMS provider may send the message more than once to the same destination. The receiving application must be tolerant of receiving duplicate messages.
		Transacted	Messages sent to or received from a JMS provider within an activity are acknowledged explicitly after processing. Thus, the activity plays the role of a transactional lock. If the **acknowledgeMode** is specified as **Transacted**, a Bridge JMS client acknowledges a consumed message only after the activity diagram that implements the JMS adapter functionality completes without throwing an exception. This holds even if the activity diagram receives more then one message from, and/or sends messages to the queue during its execution.
timeToLive	Specify the expiration time of a sent message (refer also to the description of **JMSExpiration** in the JMS headers /properties).		
selector	Specify a selector to filter the received messages. Refer to the official Java Message Service Specification for the selector statement syntax.	e.g. `JMSType='alpha'`	
destinationName	Specify the name of the queue or topic.		
options	Specify a list of comma separated options in form of <name>=<value>. These options will be interpreted as native options. The available options depend on the JMS provider.	e.g. `queue.JMSDestination=aTestQueue`	
user	Specify the JMS credentials in form of <user>/<password>.	e.g. `system/manager`	
port	Specify the port the JMS provider is listening to.		
protocol	Protocol used to communicate with the JMS provider. The protocol normally is set automatically.		
jndiPath	Specify a path to the JNDI file (.bindings), if a JNDI provider is used and protocol is file.		
host	Specify the host name of the JMS provider.	**localhost**	Default.
		any other qualified host name	
isJNDIProvider	Specify whether the JMS provider is also a JNDI provider. Default is **false** if not specified.	true	The JMS provider is a JNDI provider.
		false	The JMS provider is not a JNDI provider (default).

Additional Pins

Pin	Description	Allowed Values
jmsConnectionInfo	Additional class that provides an easy way to get all tagged values that can be set dynamically.  You can use this class to get the tagged values from the alias, change the values you need, and provide this class to a dynamic adapter call.	See table above.

<<POP3Alias>>

Tagged Value	Description	Allowed Values
host	POP3 host.	A valid hostname.
port	POP3 port.	A valid port number.
user	Username/password.	

<<REST Alias>>

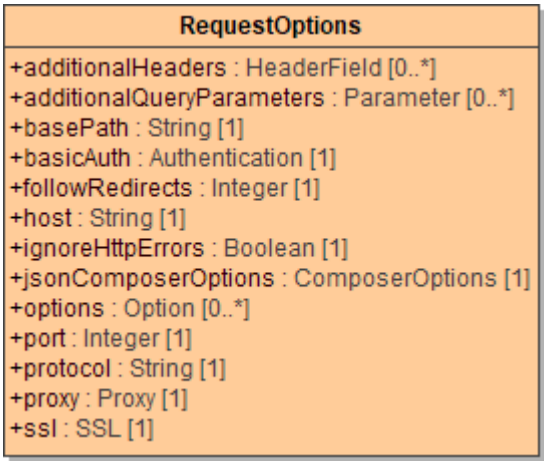
Attribute	Description	Allowed Values	
Additional Headers (additionalHeaders)	This tagged value can contain a list of additional headers in form of name/value pairs.	Valid format is: <name>:<value>, e.g. `API-Key:e2e`. Separate multiple headers with a comma.	
Base Path (basePath)	Specify here the base 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.
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.

Additional Pins

Pin	Description	Allowed Values

requestOptions	Additional class that provides an easy way to get all tagged values that can be set dynamically.  You can use this class to get the tagged values from the alias, change the values you need, and provide this class to a dynamic adapter call.	See table above.

<<SAPAlias>>

Tagged Value	Description	Mandatory / Optional	Allowed Values	
protocol	Supply the connection protocol	mandatory	rfc	to use the RFC protocol
			trfc	to use the tRFC protocol
host	Supply the gateway host name (optional).	mandatory	any string, must be a valid SAP host	
			localhost (default)	
client	Supply the SAP logon client.	mandatory	any string, must be a valid SAP client	
user	Supply user and password.	mandatory	any string matching the pattern "<user>/<password>"	
systemNumber	Supply the system number of the SAP system.	optional	any string, default = "00"	
routerString	The router string is an additional routing information used by SAP RFC backend clients. SAP RFC clients prepend the DNS hostname with this string to get an application server name that is resolvable by the RFC library.	optional	any string	

poolSize	Runtime 2015.10 Supply the maximum number of parallel connections to the SAP system.
	
	• The pool size can be defined per connection string. If you have multiple aliases with the same connection string, the highest value will be used.
	The same applies, if you set the values in the SAP adapter settings on the Bridge.
	• If this tag is not set, the connection pool size specified on the <<E2 EComposite>> will be applied.
	• **Compatibility note:** This tag will not be created for existing aliases in older models. You have to add the tag manually if you want to set it.
	Older xUML Runtimes (before version 2015.10) will not start with the setting being present. As a workaround, you can delete the tag value.
	
	If all connections from the SAP connection pool are in use, warnings will be logged to the transaction log each second a service is waiting for connection.
	
	`2015-12-08 16:47:24 +0100`
	`0000000182469dcd0001612899fea700e3d869aa 3`
	`SAPConnectionPool 0 OK SAPRFC IO_ENTER`
	`PoolExhausted`
	`2015-12-08 16:47:25 +0100`
	`0000000182469dcd0001612899fea700e3d869aa 3`
	`SAPConnectionPool 1000 OK SAPRFC`
	`IO_EXIT PoolExhausted`
	
	In this case, increase the pool size to solve the problem.
		optional	default = 10

language	Supply the SAP logon language.	optional	1-byte SAP language like E for English, D for German 2-byte ISO language like EN for English, DE for German
sapTrace	The effect of this flag being true is two fold: • First, the SAP RFC libraries will write trace file information (**.trc**) into the directory the service has been deployed to. • Second, by using the SAP transaction ***SMGW** (SAP gateway monitor) we can monitor the dataflow from and to the gateway the server is registered on. The SAP trace level has to be defined in tagged value **connectionString**. See Client Connection Options for a list of the allowed trace level values.	optional	
options	A blank separated list of name value pairs: name1="value1" name2=" value2", and so forth. The possible name value pairs can be found further below.	optional	

<<SMTPAlias>>

Tagged Value	Description	Allowed Values
host	Supply the fully qualified name of the smart host.	any string, must be a valid SMTP server
		localhost (default)
port	Supply the SMTP port number of the smart host.	any integer
		25 (default)
user	Supply user and password as **user /password**.	any string

<<SOAPAlias>>

Tagged Value	Description	Allowed Values / Examples	
Soap Version (soapVersion)	Specify the SOAP version used with this SOAP call.	**1.1**	Use SOAP version 1.1 (default).
		1.2	Use SOAP version 1.2.

Protocol (protocol)	Specify the transport protocol.		
Method (method)	Specify the HTTP method.	One of: get, **post** (default), read, write, put, list, patch	
Port (port)	Specify the machine port number the service is binding to.	a valid port number (default: **80**)	
Path (path)	Specify the HTTP path for the SOAP request.		
Host (host)	Specify an optional host name.	a valid hostname (default: **localhost**)	
Advanced			
Timezone (timezone)	Time zone string as specified in the time zone appendix. timezone is used to print DateTime expressions.	Examples: `"Australia/Melbourne"`, `"CET"`, `"Etc/GMT+10"`. If "local" is used, the date/time is printed relative to the local timezone of the Bridge, for example: `2012-10-01T12:36:47.0+02:00` (the timezone of the Bridge is UTC+02:00)	
Date Format String (dateFormatString)	Date formatting code as listed in Date and Time Formatting. If nothing is defined, the XSD standard is used.	Example: `%F`	
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 incoming 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)	Builder 7.12.0 Runtime 2021.1 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.
Follow Redirects (followRedirects)	Maximum number of redirects to follow.	any integer	
Options (options)	Specify native cURL options as listed in Setting cURL Options on the URL Adapter. Also refer to this page for more details on how to set these options.	Example: `CURLOPT_TIMEOUT=20`	
Ignore Http Errors (ignoreHttpErrors)	Not implemented for SOAP.		
Authentication			
User (user)	Username/password.		
Proxy			
Proxy Type (proxyType)	Type of the proxy	one of: `HTTP`, `SOCKS5`	
Proxy User (proxyUser)	Proxy user.		
Proxy URL (proxyURL)	URL of the proxy server.		
SOAP			
Send Transaction ID (sendTransactionID)	Defines whether the transaction ID is sent in a SOAP header element **<TransactionID>**...**</TransactionID>** with namespace **http://e2e.ch/bridge** . If sent, the calling and the called service can be identified to belong to the same transaction.	false	Do not send transaction ID in the SOAP header element.
		true	Send transaction ID in the SOAP Header element (default).
Message Format (messageFormat)	Defines the formatting of the SOAP message.	None	Format messages without using linebreaks and indentation.
		Linebreaks	Format messages using linebreaks, but no indentation (default).
		Indentation	Format messages using linebreaks and indentation.
SendCorrelationID (sendCorrelationID)	Defines whether the correlation ID is sent in a SOAP header element **<CorrelationID>**...**</CorrelationID>** with namespace **http://e2e.ch/bridge** . The correlation ID is used to identify each SOAP call.	false	Do not send the correlation ID in the SOAP header element.
		true	Send the correlation ID in the SOAP header element (default).
SSL			
Ssl Private Key Type (sslPrivateKeyType)	Type of the private key.	one of: **PEM** (default), DER, ENG	
Ssl Verify Host (sslVerifyHost)	Whether to verify the host information form the SSL connection.	**On**	Verification on (default).
		Off	Verification off.

		Existence	Limit verification to the mere existence of the host.
Ssl Verify Peer (sslVerifyPeer)	Whether to verify the peer information from the SSL connection.	**On**	Verification on (default).
		Off	Verification off.
Ssl Private Key File (sslPrivateKeyFile)	Name of the file that contains the private key.		
Ssl Private Key Password (sslPrivateKeyPassword)	Password for the private key.		
Ssl Certificate File (sslCertificateFile)	Name of the file that contains the client certificate.		
Ssl CA Info (sslCAInfo)	Name of the file containing additional certificates for the connection verification (e.g. additional root CAs).		
Ssl Certificate Type (sslCertificateType)	Type of the certificate.	one of: **PEM** (default), DER, P12	

Additional Pins

All URL adapter tags are also relevant for the SOAP adapter. See also Providing the SOAP Adapter with URL Parameter.

Pin	Description	Allowed Values
urlMessage	Additional class that provides an easy way to get all tagged values that can be set dynamically. **URLMessage** +authentication : Authentication [1] +commands : String [0..*] +content : Blob [1] +followRedirects : Integer [1] +headerParameters : HeaderField [0..*] +method : String [1] +options : Option [0..*] +proxy : Proxy [1] +ssl : SSL [1] +url : String [1] You can use this class to get the tagged values from the alias, change the values you need, and provide this class to a dynamic adapter call.	See table above.

<<SQLAlias>>

Tagged Value	Description	Allowed Values
General		
dbConnectionString	The format of the database connection string depends on the type of the database. For more details see Database Server-Specific Notes for SQL Adapters.	
dbType	Type of the database.	Oracle, SQLServer, InterBase, SQLBase, ODBC, DB2, Informix, Sybase, MySQL, PostgreSQL, SQLite, DBTypeVariable
dbTypeVariable	If the tagged value **dbType** is set to **DBTypeValue**, the **dbTypeVariable** tagged value is used to define the type of the database. The **dbType** then can be defined by a setting variable. This is to handle the case, that you not want to hard code the **dbType**, but to configure it at runtime via the E2E Bridge. See Using Global Setting Variables for more information on how to define a global setting variable in the E2E Bridge. Use one of the listed **dbTypes** in the settings variable. If you configure an unknown **dbType** via the E2E Bridge, the xUML Runtime will throw an error on service startup.	Any global setting variable from the E2E Bridge. Example: `{{my_setting_variable}}`
user	DB user. Optional the password can be given after a '/'. However, this is recommended for development purposes only.	Example: `{{DB_USER}}/{{DB_PASSWORD}}`
options	This tagged value can hold a comma separated list of <name>=<value> pairs. These list elements are interpreted as native options. The possible name-value pairs depend on the database type. A comprehensive list can be found at https://www.sqlapi.com/ApiDoc/servers/	Example: `SSPROP_INIT_ENCRYPT=VARIANT_TRUE`
transactionIsolationLevel	Bridge 7 Specify here the required transaction isolation level of the SQL connection according to SQL-92 standard. Refer to Wikipedia for a detailed description of the available isolation levels. Please note that not all databases support all levels. In this case a database-specific mapping will occur. For persistent state databases no other than **<UNSPECIFIED>** and **DBMS default** are allowed.	**DBMS default**: Use the default isolation level of the connected database system. **Read uncommitted**: Lowest isolation level. Dirty reads allowed, SQL adapter may fetch not-yet-committed changes of other transactions. **Read committed**: Lock-based concurrency control. **Repeatable read**: Lock-based concurrency control.

		Serializable	Highest isolation level. Lock-based concurrency control.
Localization			
charset	Any database uses a charset to encode Strings. If the database uses UNICODE charsets (UTF-8, UTF-16, UTF-32), encoding is handled automatically. If the database is not UNICODE compliant, the Bridge assumes 7-bit ASCII by default. However, in many cases it necessary to define the charset explicitly. This is done by the tagged value **charset** as shown below. The charset needs to be the same as defined at the database settings. All possible charset definitions are listed in section Charset Definitions.	Example: `UTF-8` See Charset Definitions for a list of possible values.	
timezone	You can enter a valid time zone or the value **local**, which uses the time zone of the xUML service. See Time Zones for a list of possible values. If **timezone** does not contain any content (is NULL), UTC is used.	**Default is NULL**	Example: `"Australia/Melbourne", "CET", "Etc/GMT+10"`
unicodeMode	Added in Builder 6.0.15.5 Runtime 2015.15 Specify the encoding for database access. We recommend to use the **Platform default** unless you suspect an encoding incompatibility (see Troubleshooting the SQL Adapter). This option represents the former behavior and is fully backwards-compatible - means, it can be used with older xUML Runtimes. The two other (force mode) options will be ignored by older Runtimes without warning.	**Platform default** (default)	Use the platform default mode. This is • Unicode: for Windows systems • non-Unicode: for all others This option is backwards compatible to older Runtimes.
		Unicode	Force Unicode mode.
		non-Unicode	Force non-Unicode mode.
Connection Pooling			
connectionPooling	Added in Builder 5.1.8.58 Runtime 5.1.82.0 This tagged value controls the connection pooling. If true, each connection is put into a pool after use. If an SQL adapter requires a connection, it is taken from the pool. If no connection is available, a new connection is being created and put into the pool after use. The time the connection is kept in the pool depends on the other pooling parameters.	true	Database connections are pooled.
		false	Database connections are not pooled.
maxConnectionAge	After a given connection age (in minutes) the connection will be closed and removed from the pool.	Connection age in minutes, default is **15 minutes**, **-1** means forever.	
maxConnectionIdleTime	Connections not used for the time specified (in minutes) will be closed and removed from the pool. This is useful for connections going through firewalls because such connections might be cut off after some time.	Values in minutes, default is **60**.	
maxConnectionReuse	This tagged value controls how often a connection can be re-used. After the connection has been re-used for **maxConnectionReuse**, it will be closed and not put back into the pool. This feature has been introduced because some databases had problems if the connection was re-used too often. Value **-1** means the connection will be re-used forever. In this case you should define reasonable values for **maxConnectionAge** or **maxConnectionIdleTime** (see above).	0	pooling is implicitly switched off.
		-1	connections are pooled forever

		a v al ue	number of connections to be pooled, default is 1000.
	Note that the pooling is implicitly switched off, if **maxConnectionReuse** is set to 0.		
Qualifier			
schema	String that prefixes tables and stored procedures. For example, if schema is set to S1, all tables accessing the current DB are prefixed by "S1.". This works only if the tables are marked using the `TABLE::` keyword, e. g `TABLE::TEMPLOYEE` in SQL statements. If you do not prefix the table name by `TABLE::`, the tablename is used as it is.		
tableQualifier	String that prefixes tables. For example, if **tableQualifier** is set to TQ1, all tables accessing the current DB are prefixed by "TQ1", e.g. `TQ1TEMPLOYEE`. If schema and table qualifier are given, all tables will become: <schema>. <tableQualifier><tableName>. This works only if the tables are marked using the `TABLE::` keyword, e. g `TABLE::TEMPLOYEE` in SQL statements. If you do not prefix the table name by `TABLE::`, the tablename is used as it is.		

<<URLAlias>>

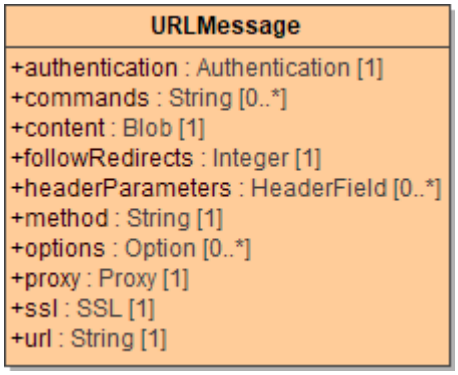
Name	Description	Allowed Values / Examples	
Protocol (protocol)	Transport protocol.	one of: ldap, file, ftp, ftps, sftp, gopher, **http** (default), https, telnet	
Port (port)	Machine port number the service is binding to. This port number can be given at service level only.	a valid port number (default: **80**)	
Path (path)	HTTP path for the SOAP request.		
Host (host)	Host name. Default is "localhost" if no value is specified.	a valid hostname (default: **localhost**)	
Method (method)	HTTP method of the URL adapter call. Default is "POST" if no value is specified.	one of: get, **post** (default), read, write, put, list	
Advanced			
Follow Redirects (followRedirects)	The maximum number of redirects to follow.	any positive integer or 0	
Ignore Http Errors (ignoreHttpErrors)	If true, HTTP error codes > 300 will not cause an exception in the model. This implies, that the response body is accessible even if HTTP errors occur. The default value is false.	true	Ignore HTTP error codes > 300.
		false	Do not ignore HTTP errors (default).
Options (options)	Native cURL options as listed on 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	valid cURL options, e.g. `CURLOPT_TIMEOUT=20`	

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 incoming 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)	Builder 7.12.0 Runtime 2021.1 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.
Authentication			
User (user)	Username/password.		
Proxy			
Proxy URL (proxyURL)	URL of the proxy server.		
Proxy User (proxyUser)	Proxy user.		
Proxy Type (proxyType)	Type of the proxy	one of: `HTTP`, `SOCKS5`	
SSL			
Ssl CA Info (sslCAInfo)	Name of the file containing additional certificates for the connection verification (e.g. additional root CAs).		

Ssl Certificate File (sslCertificateFile)	Name of the file that contains the client certificate.		
Ssl Private Key File (sslPrivateKeyFile)	Name of the file that contains the private key.		
Ssl Private Key Password (sslPrivateKeyPassword)	Password for the private key.		
Ssl Certificate Type (sslCertificateType)	Type of the certificate.	one of: **PEM** (default), DER, P12	
Ssl Verify Host (sslVerifyHost)	Whether to verify the host information form the SSL connection.	**On**	Verification on (default).
		Off	Verification off.
		Existence	Limit verification to the mere existence of the host.
Ssl Verify Peer (sslVerifyPeer)	Whether to verify the peer information from the SSL connection.	**On**	Verification on (default).
		Off	Verification off.
Ssl Private Key Type (sslPrivateKeyType)	Type of the private key.	one of: **PEM** (default), DER, ENG	

Additional Pins

Pin	Description	Allowed Values

urlMessage	Additional class that provides an easy way to get all tagged values that can be set dynamically.  You can use this class to get the tagged values from the alias, change the values you need, and provide this class to a dynamic adapter call.	See table above.