

API Calls

For API calls Cookies are handled automatically, only the *incognito* call won't add nor set any cookies

With the *addGlobalHeader* call, you can add a global header automatically for every call, but it won't be added if you process the call with *incognito* call.

Both calls *process* and *incognito* will be added to AIQ requests and responses.

formData()

Creates a map that can be used in body requests (POST, PUT, PATCH, DELETE) as **application/x-www-form-urlencoded** content type. Elements can be added to the form similar to the next code

```
var myForm = formData();
myForm.put("username", $username);
myForm.put("password", $password);

post($url, myForm);
```

createGet (String: URL) → HttpRequest

Creates a basic GET call, useful for adding headers before sending it

createHead (String: URL) → HttpRequest

Creates a basic HEAD call, useful for adding headers before sending it

createOptions (String: URL) → HttpRequest

Creates a basic OPTIONS call, useful for adding headers before sending it

createPost(String: URL) → HttpRequest

Creates a basic POST call, without body, useful for adding headers before sending it

```
var postRequest = createPost("http://demosite.appvance.com/orders/populate");
```

createPost(String: URL, Map<String, String>: data) → HttpRequest

Creates a basic POST multipart form data post, useful for adding headers before sending it

```
var logInForm = formData();
logInForm.put("username", "appvance");
logInForm.put("password", "appvance");

var postData = createPost("http://demosite.appvance.com/orders/populate", logInForm);
process(postData);
```

createPost(String: URL, JSONObject: data) → HttpRequest

Creates a basic POST call with an "application/json" for content type and the body is the stringified json, useful for adding headers before sending it. Similar

```
var jsonObject = {};
jsonObject.username= $username;
jsonObject.password = $password
var post = createPost($url, jsonObject);

var jsonArray = [];
jsonArray[0]= $firstEntry;
jsonArray[1]= $firstEntry;
var post = createPost($url, jsonArray);
```

createPost(String: URL, String: Content, MimeType:String) → HttpRequest

Creates a post with the specific content type

createPost(String: URL, ExcelWrapper: ExcelWrapper) → HttpRequest

Creates a post call using multipart/form-data and a excel file. The excel mime is the standard application/vnd.openxmlformats-officedocument.spreadsheetml.sheet

```
var postRequest = createPost("http://demosite.appvance.com/orders/populate", excelWrapper);

//Read template
var echoTemplate = new java.lang.String(readBytes("{ds}/request.echo.xml")); //SOME TEXT FILE

//Process template
var echoString=echoTemplate.replace("ECH01","Hello World");
//Create post method
var echoPost = createPost("http://localhost:8080/TMServer/ws/TestNode",echoString,"text/xml");

echoPost.addHeader("SOAPAction","");
var result=process(echoPost);
```

createPatch(String: URL) → HttpRequest

Creates a basic PATCH call, without body, useful for adding headers before sending it.

createPatch(String: URL, Map<String, String>:data) → HttpRequest

Creates a basic PATCH multipart form data patch, useful for adding headers before sending it.

createPatch(String: URL, JSONObject:data) → HttpRequest

Creates a basic PATCH with an "application/json" for content type and the body is the stringify of the json, useful for adding headers before sending it.

createPut(String:URL) → HttpRequest

Creates a basic PUT call, without body, useful for adding headers before sending it.

createPut(String:URL, Map<String, String>:data) → HttpRequest

Creates a basic PUT multipart form data put, useful for adding headers before sending it.

`createPut(String: URL,JSONObject: data) → HttpRequest`

Creates a basic PUT call with an "application/json" for content type and the body is the stringified of the json, useful for adding headers before sending it.

`createDelete(String: URL) → HttpRequest`

Creates a basic DELETE call, without body, useful for adding headers before sending it.

`createDelete(String: URL,Map<String, String>: data) → HttpRequest`

Creates a basic delete multipart form data put, useful for adding headers before sending it.

`createDelete(String: URL,JSONObject: data) → HttpRequest`

Creates a basic delete with an "application/json" for content type and the body is the stringified of the json, useful for adding headers before sending it

`get (String: URL) → HttpResponse`

This call is a basic get, requires a URL and returns a HttpResponse if the connection succeeds. Otherwise it will throw a Server not found or Connection reset, Connection Timeout depending on the error.

`head (String: URL) → HttpResponse`

This call is a basic head, requires a URL and returns a HttpResponse if the connection succeeds. Otherwise it will throw a Server not found or Connection reset, Connection Timeout depending on the error.

options(String: URL) → HttpResponse

This call is a basic OPTIONS, requires an URL and returns a HttpResponse if the connection succeeds. Otherways it will throw a Server not found or Connection reset, Connection Timeout depending on the error.

post(String: URL) → HttpResponse

This call is a basic post without a body, requires a URL and return a HttpResponse if the connection succeeds. Otherways it will throw a Server not found or Connection reset, Connection Timeout depending on the error.

post(String: URL, Map<String, String: data) → HttpResponse

This call is a basic post with "x-www-form-urlencoded" format, where the entries are the data entries, requires a URL and the map of entries and returns a HttpResponse if the connection succeeds. Otherways it will throw a Server not found or Connection reset, Connection Timeout depending on the error.

post(String: URL, JSONObject: data) → HttpResponse

This call is a basic post with "application/json format, where the data is the JSON.stringify of the object, requires a URL and the json object and returns a HttpResponse if the connection succeeds. Otherways it will throw a Server not found or Connection reset, Connection Timeout depending on the error.

post(String: URL, data: ExcelWrapper) → HttpResponse

This call is makes a post, using a multipart/form data and a excel content. The mime type is application/vnd.openxmlformats-officedocument.spreadsheetml.sheet

put(String: URL) → HttpResponse

This call is a basic put without a body, requires a URL and return a HttpResponse if the connection succeeds. Otherways it will throw a Server not found or Connection reset, Connection Timeout depending on the error.

put(String: URL, Map<String, String>: data) → HttpResponse

This call is a basic put with "x-www-form-urlencoded" format, where the entries are the data entries, requires a URL and the map of entries and returns a HttpResponse if the connection succeeds. Otherways it will throw a Server not found or Connection reset, Connection Timeout depending on the error

put(String: URL, JSONObject :data) → HttpResponse

This call is a basic put with "application/json" format, where the data is the JSON.stringify of the object, requires a URL and the JSON object and returns a HttpResponse if the connection succeeds. Otherways it will throw a Server not found or Connection reset, Connection Timeout depending on the error.

patch(String: URL) → HttpResponse

This call is a basic patch without a body, requires a URL and return a HttpResponse if the connection succeeds. Otherways it will throw a Server not found or Connection reset, Connection Timeout depending on the error.

patch (String: URL, Map<String, String>: data)→ HttpResponse

This call is a basic patch with "x-www-form-urlencoded" format, where the entries are the data entries, requires a URL and the map of entries and returns a HttpResponse if the connection succeeds. Otherways it will throw a Server not found or Connection reset, Connection Timeout depending on the error.

patch(String: URL,JSONObject :data)→ HttpResponse

This call is a basic patch with "application/json" format, where the data is the JSON.stringify of the object, requires a URL and the json object and returns a HttpResponse if the connection succeeds. Otherways it will throw a Server not found or Connection reset, Connection Timeout depending on the error.

`_delete(String: URL)→ HttpResponse`

This call is a basic delete without a body, requires a URL and return a HttpResponse if the connection succeeds. Otherways it will throw a Server not found or Connection reset, Connection Timeout depending on the error. `_delete` is used instead of `delete` as `delete` is a reserved keyword in js.

`_delete(String: URL, Map<String, String>: data)→ HttpResponse`

This call is a basic HTTP delete method with "x-www-form-urlencoded" format, where the entries are the data entries, requires a URL and the map of entries, and returns a HttpResponse if the connection succeeds. Otherways it will throw a Server not found or Connection reset, Connection Timeout depending on the error. `_delete` is used instead of `delete` as `delete` is a reserved keyword in js.

`_delete(String: URL,JSONObject: data) → HttpResponse`

This call is a basic HTTP delete method with "application/json" format, where the data is the `JSON.stringify` of the object, requires a URL and the JSON object, and returns a HttpResponse if the connection succeeds. Otherways it will throw a Server not found or Connection reset, Connection Timeout depending on the error. `_delete` is used instead of `delete` as `delete` is a reserved keyword in js.

`process(HttpRequest:request) → HttpResponse`

Process the request adds automatically the cookies and global headers and returns the response. The response cookies (if any) will be used in subsequent request calls

`incognito(HttpRequest: request) → HttpResponse`

Process the request as it is without adding cookies or global headers. The response cookies (if any) will be discarded. Keep in mind `incognito` may be set over the same socket of request calls. There is no socket-level management in Service Suite yet

addGlobalHeader(String: key, String: value)

Adds new global header entry used with the ***process*** command , global headers won't be added with the **incognito** call

```
addGlobalHeader("Hello","World");
var logInForm = formData();
logInForm.put("username","appvance");
logInForm.put("password","appvance");
var postData = createPost("http://demosite.appvance.com/orders/populate",logInForm);
process(postData);
```

removeGlobalHeader(String: key)

Removes a global header.

clearGlobalHeaders()

Removes every global header.

getGlobalHeaders() → HashMap<String,String>

Returns list of global headers

getCookies(String: url)→ List<HttpCookie>

Gets the cookies from a specified URL

clearCookies()

Clear every cookie from all domains

clearCookies(String:domain)

Clear every cookie from one specific domain

getCookie(String: URL,String:name) → HttpCookie

Gets the cookie from a specified URL and the cookie value

hgetCookieValue(String:URL,String:name)→ String

Gets the cookie value from a specified URL and the cookie value

readCSV(String: fileName) → String [][]

Gets the file name and parses the data using Excel CSV parser to returns it as a string array

writeCSV(String: filepath,String[][]: data)

Gets a path to creates and writes bytes in a CSV file with data received as a string array

writeCSV(String: filepath,Map<Object,Object>: data)

Gets a path to creates and writes bytes in a CSV file with data received as a map object

writeJSON(String: filepath, JSONObject: obj)

Gets a path to creates and writes a JSON and parses the data into a string array

writeJSON(String: filepath, JSONArray: array)

Gets a path to creates and writes a JSON and parses the data into a string array

writeJSON(String: filepath,Object: o)

Gets a path to creates and writes a JSON and if gets a native object parses the JSON object into a string or if is a native array parses it to a JSON array

readImage(String: filepath) → BufferedImage

Gets an image path and verifies if the path exists to read an image otherwise reports that the given path was not found

readText(String: filepath)→ String

Gets a path and verifies the file exists and read the text it has

writeText(String: filepath,String: data)

Gets a path and writes it with the data provided on the data variable

Last updated: June 02, 2023

If you have any feedback on this topic, please contact us at documentation@appvance.ai

Revision #1

Created 2026-07-24 14:33:28 UTC by Admin

Updated 2026-07-24 14:33:28 UTC by Admin