8.4 - List of Services Workbench Calls

List of Services Workbench Calls

- Service Suite JS Reference
- DynamoDB
- Use Response Extraction Calls in Service Suite
- Include Calls
- Cookies
- Mail Client
- API Calls
- Browser Simulation Calls
- AIQ Integration
- Assertions
- Network Workbench Support
- SQL Calls in Service Suite
- OCR Support
- Image Recognition
- NTLM Authentication
- Proxy Support
- Mobile Calls
- Test Designer Integration
- Screen Interaction (FFMPEG required)
- OS Screen Interaction
- Excel Support
- S3 Client
- Binary Data
- Git Client

Service Suite JS Reference

The list of standard JS call which can be used in Service Suite

encodeURIComponent

JS function encodeURIComponent as defined in https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/encodeURIComponent

decodeURIComponent

JS function decodeURIComponent as defined in https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/decodeURIComponent

JSON.stringify(Object:JSON, Object:replacer, Number:space)

JS function JSON.stringify as defined in https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/JSON/stringify

JSON.parse(String:text, Object:reviver)

JS function JSON.parse as defined in https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/JSON/parse

---

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

Last updated: October 15, 2026

DynamoDB

There is one call used to extract pieces of content from any response data.

createDynamoDBClient(String:accesskey, String:secret, String:region)

Creates a new client for AWS DynamoDB service. This client implements methods that uses AWS SDK 2 software.amazon.awssdk.services.dynamodb.DynamoDbClient.

```
if (typeof $accesskey == "undefined") {
    EncryptedHashDPL("EncryptedAWSCredentials.csv", "mysecretPassword123!");
}
var dynamoDB = createDynamoDBClient($accesskey, $secret, "us-west-2");
```

createTable(tableName, primaryKey, sortKey)

Creates a new table

```
dynamoDB.createTable("movies","movieName");
```

```
```

deleteTable(tableName)

deletes a Table.

```
client.deleteTable("movies-table");
```

Use Response Extraction Calls in Service Suite

Use Response Extraction Calls in Service Suite

Use response extraction calls when a Service Suite test needs to capture a value from one response and reuse it in a later request or validation. This is commonly used for tokens, identifiers, links, messages, or values returned by an API under test.

Recommended workflow

1. Identify the stable response field that contains the value.
2. Extract only the value needed by the later step.
3. Store the extracted value with a clear name.
4. Use the stored value in the next request, assertion, or Scenario Editor flow where applicable.

Good practices

- Validate that extraction fails clearly when the response shape changes.
- Avoid extracting broad response fragments when a single field is enough.
- Do not log sensitive tokens or secrets in shared output.

Last updated: September 8, 2026

Include Calls

Service Suite currently provides the following calls for including JavaScript files and handling binary data.

include(String: file)

Executes the specified file. Use this call to add libraries. If the same file is included *N* times, it executes *N* times. The path is relative to the current file.

readBytes(String: file) → byte[]

Reads a binary file and returns a byte array.

```
var myStringVar = new java.lang.String(readBytes("{ds}/binary.dat"));
assertEquals("hello world", myStringVar);
var $aJsonString = new java.lang.String(readBytes("{mds}/somejsonfile.json"));
var jsonObj = JSON.parse($aJsonString);
```

createFolder(String: folder)

Creates a folder with the specified name.

Last updated: October 15, 2026

If you have feedback on this topic, contact documentation@appvance.ai.

Cookies

- `addCookie(String: URL, String: name, String: value)`
- `addCookieFromHeader(String: URL, String name)`
- `clearCookie(String URL, String name)`
- `clearCookies()`
- `clearCookies(String domain)`
- `getCookie(String URL, String name)`
- `getCookieValue(String URL, String name)`
- `printCookies(String domain)`

Last updated: October 15, 2026

If you have feedback on this topic, contact documentation@appvance.ai.

Mail Client

createMailClient(String host, String userMail, String password, String protocol)

Returns a `MailService` object.

MailService methods

- `login()`
- `setFromFolder(String: folder)`
- `logout()`
- `getLastUnreadMessages(int: quantity)`

Last updated: October 15, 2026

If you have feedback on this topic, contact documentation@appvance.ai.

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 stringify of the 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("ECHO1","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 stringify 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 deletewith an "application/json" for content type and the body is the stringify 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. Otherways 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. Otherways 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

---

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

Last updated: October 15, 2026

Browser Simulation Calls

open(String:URL) → HtmlResponse

Opens URL (with GET method) and processes the returned HTML page by parsing it and retrieving the resources from it. It doesn't retrieve CSS/JS related resources. Will execute redirect requests.

```
addGlobalHeader('Connection','keep-alive');
addGlobalHeader('Accept-Encoding','gzip');
addGlobalHeader('User-Agent','Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_1)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/78.0.3904.108 Safari/537.36');
var page = open('https://demosite.appvance.com:443/');
  var htmlResp= page.resp;
  var htmlExtract = extractHTML(htmlResp);
  var $corr1= htmlExtract.getAttrByXPath('//html/head/link[2]/@href');
  var $corr2= htmlExtract.getAttrByXPath("//div[@id='spree-
header']/header/div[1]/div[1]/nav/ul/li[1]/a[1]/@href");
  var $corr3= htmlResp.getHeader('ETag');
  var resp= page.get("//div[@id='product_10']/div[1]/div[1]/a/img[1]");
  var $corr13= resp.getHeader('ETag');
  var resp= page.get("//div[@id='product_6']/div[1]/div[1]/a/img[1]");
  var $corr9= resp.getHeader('ETag');
  var resp= page.get("//div[@id='product_11']/div[1]/div[1]/a/img[1]");
  var $corr14= resp.getHeader('ETag');
  var resp= page.get("//div[@id='product_3']/div[1]/div[1]/a/img[1]");
  var $corr6= resp.getHeader('ETag');
  var resp= page.get("//div[@id='product_5']/div[1]/div[1]/a/img[1]");
  var $corr8= resp.getHeader('ETag');
  var resp= page.get("//div[@id='product_7']/div[1]/div[1]/a/img[1]");
  var $corr10= resp.getHeader('ETag');
  var resp= page.get("//div[@id='product_1']/div[1]/div[1]/a/img[1]");
  var $corr4= resp.getHeader('ETag');
  var resp= page.get("//div[@id='product_2']/div[1]/div[1]/a/img[1]");
  var $corr5= resp.getHeader('ETag');
  var resp= page.get("//div[@id='product_9']/div[1]/div[1]/a/img[1]");
  var $corr12= resp.getHeader('ETag');
```

```
  var resp= page.get("//div[@id='product_12']/div[1]/div[1]/a/img[1]");
  var $corr15= resp.getHeader('ETag');
  var resp= page.get("//div[@id='product_4']/div[1]/div[1]/a/img[1]");
  var $corr7= resp.getHeader('ETag');
  var resp= page.get("//div[@id='product_8']/div[1]/div[1]/a/img[1]");
  var $corr11= resp.getHeader('ETag');
get('https://demosite.appvance.com:443/assets/spree/frontend/all-
dcce360e77937eb06679c31bcd694097abc747ae87d2bc292fddd5318b67ceff.css');
var resp = get('https://demosite.appvance.com:443/cart_link');
```

open(HttpRequest:req) → HtmlResponse

Opens URL and processes the returned HTML page by parsing it and retrieving the resources from it. It doesn't retrieve CSS/JS related resources

createOpen(String:URL) → HtmlRequest

Creates a HtmlRequest to useful for setting headers before actual open

submit(String:URL, Object:form) → HtmlResponse

Send a POST and process the returned HTML page by parsing it and retrieving the resources from it. It doesn't retrieve CSS/JS related resources. Will execute redirect requests.

submit(HttpRequest:req,Object:form) → HtmlResponse

Send a POST and process the returned HTML page by parsing it and retrieving the resources from it. It doesn't retrieve CSS/JS related resources. Will execute redirect requests.

createSubmit(String:URL, Object:form) → HtmlRequest

Creates a HtmlRequest to useful for setting headers before actual submit

process(HtmlRequest) → HtmlResponse

Process a HtmlRequest and process the result HTML by parsing it and retrieving the resources from it. It doesn't retrieve CSS/JS related resources

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

Last updated: October 15, 2026

AIQ Integration

startStep(String: step name)

Starts an AIQ step for summary reporting. Starting a new step will close the previous one automatically.

endFailingStep(String: failure)

End an AIQ step and mark it as failed with failure reason.

startSummaryStep(String: step name)

Starts an AIQ summary step for summary reporting. Must make pair with the corresponding endSummaryStep/endFailingSummaryStep.
Screen%20Shot%202021-04-28%20at%2015.24.39%20(1).png

endSummaryStep(String: step name)

Ends AIQ summary step for summary reporting. Must make pair with the corresponding startSummaryStep.

endFailingSummaryStep(String: failure)

Starts an AIQ step for summary reporting. Must make pair with the corresponding startSummaryStep.

HashDPL(String: filename)

Creates variables starting with $ based on the file name, the values with spaces are replaced with an underscore. e.g. the value "page index" will create variable $page_index. It will use the data in the first line after the header line of the DPL.

HashDPL(String: filename, Integer index)

Creates variables starting with $ based on the file name, the values with spaces are replaced with an underscore. e.g. the value "page index" will create variable $page_index. It will use the data in the line number defined by the index.

EncryptedDPL(String: filename)

Creates variables starting with $ based on the file name, the values with spaces are replaced with an underscore. e.g. the value "page index" will create variable $page_index. It will use the data in the first line after the header line of an Encrypted .CSV file.

EncryptedDPL(String: filename, Integer index)

Creates variables starting with $ based on the file name, the values with spaces are replaced with an underscore. e.g. the value "page index" will create variable $page_index. It will use the data in the line number defined by the index of an Encrypted .CSV file.

```
if (typeof $accesskey == "undefined") {
    EncryptedHashDPL("EncryptedAWSCredentials.csv", "AppvncPassw!*6%");
}
```

locker.get(Object: key) → Object

Calls AIQ TheLocker and returns the value stored for the specified key.

locker.put(Object: key ,Object:value)

Calls AIQ TheLocker and stores the value for the specified key.

addToLocalWhiteboard(Object: key, Object:value)

Calls AIQ Whiteboard DPL and stores the value for the specified key.

```
addToLocalWhiteboard("x", 1);
var x = getFromLocalWhiteboard("x");
log(x);
```

getFromLocalWhiteboard(Object: key) → Object:value

Calls AIQ Whiteboard DPL and gets the value stored for the specified key.

```
addToLocalWhiteboard("x", 1);
var x = getFromLocalWhiteboard("x");
log(x);
```

clearLocalWhiteboard()

Calls AIQ Whiteboard DPL and clears all the key-value pairs stored.

```
clearLocalWhiteboard();
addToLocalWhiteboard("x", 1);
var x = getFromLocalWhiteboard("x");
log(x);
```

Assertions

setAssertType(String: value)

Indicates the behavior after failing assert type. The value must be *Fail*, *FailAtEnd,* or *Warning*. Default is *Fail*.

assertTrue (Boolean:value) →Boolean

Asserts if a boolean is true. The result is a boolean indicating if the assert failed if the assert type is "Warning" or "FailAtEnd" . It will fail if the assert type is "Fail". It will fail at the end of script execution if the assert type is "FailAtEnd"

assertFalse(Boolean:value) →Boolean

Asserts if a boolean is false. The result is a boolean indicating if the assert failed if the assert type is "Warning" or "FailAtEnd" . It will fail if the assert type is "Fail". It will fail at the end of script execution if the assert type is "FailAtEnd"

assertNull(Boolean:value) →Boolean

Asserts if an object is null The result is a boolean indicating if the assert failed if the assert type is "Warning" or "FailAtEnd" . It will fail if the assert type is "Fail". It will fail at the end of script execution if the assert type is "FailAtEnd"

assertNotNull (Boolean:value) →Boolean

Asserts if an object is not null. The result is a boolean indicating if the assert failed if the assert type is "Warning" or "FailAtEnd" . It will fail if the assert type is "Fail". It will fail at the end of script execution if the assert type is "FailAtEnd"

assertContains (String:full, String snipped) →Boolean

Asserts if the full text contains the snipped text. The result is a boolean indicating if the assert failed if the assert type is "Warning" or "FailAtEnd" . It will fail if the assert type is "Fail". It will fail at the end of script execution if the assert type is "FailAtEnd"

assertDoesNotContain (String:full, String snipped) → Boolean

Asserts if the full text does not contain the snipped text. The result is a boolean indicating if the assert failed if the assert type is "Warning" or "FailAtEnd" . It will fail if the assert type is "Fail". It will fail at the end of script execution if the assert type is "FailAtEnd"

assertEquals (Any:A, Any:B) → Boolean

Asserts if both elements are the same. The result is a boolean indicating if the assert failed if the assert type is "Warning" or "FailAtEnd" . It will fail if the assert type is "Fail". It will fail at the end of script execution if the assert type is "FailAtEnd"

assertNotEquals (Any:A, Any:B) → Boolean

Asserts if both elements are not the same. The result is a boolean indicating if the assert failed if the assert type is "Warning" or "FailAtEnd" . It will fail if the assert type is "Fail". It will fail at the end of script execution if the assert type is "FailAtEnd"

assertGreaterThan (Number:first, Number:second) → Boolean

Asserts the first number is greater than the second. The result is a boolean indicating if the assert failed if the assert type is "Warning" or "FailAtEnd" . It will fail if the assert type is "Fail". It will fail at the end of script execution if the assert type is "FailAtEnd"

assertLessThan (Number:first, Number:second) → Boolean

Asserts the first number is lesser than the second. The result is a boolean indicating if the assert failed if the assert type is "Warning" or "FailAtEnd" . It will fail if the assert type is "Fail". It will fail at the end of script execution if the assert type is "FailAtEnd"

assertGreaterOrEqual (Number:first, Number:second) → Boolean

Asserts the first number is greater or equal than the second. The result is a boolean indicating if the assert failed if the assert type is "Warning" or "FailAtEnd" . It will fail if the assert type is "Fail". It will fail at the end of script execution if the assert type is "FailAtEnd"

assertLessOrEqual (Number:first, Number:second) → Boolean

Asserts the first number is lesser or equal than the second. The result is a boolean indicating if the assert failed if the assert type is "Warning" or "FailAtEnd" . It will fail if the assert type is "Fail". It will fail at the end of script execution if the assert type is "FailAtEnd"

assertRange(Number: number, Number: min, Number: max) → Boolean

Asserts the first number is between the min and max. The result is a boolean indicating if the assert failed if the assert type is "Warning" or "FailAtEnd" . It will fail if the assert type is "Fail". It will fail at the end of script execution if the assert type is "FailAtEnd"

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

Last updated: October 15, 2026

Network Workbench Support

addNetworkLibrary(String: filename)

Loads a provided .netx network workbench file.

getNetworkLibrary()

Returns the network workbench library as a JSON string.

setFailOnValidations(Boolean)

If the parameter is **true** then the service suite script execution will fail if a network workbench validation fails. If the parameter is **false,** failure will be logged but execution will continue.

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

Last updated: October 15, 2026

SQL Calls in Service Suite

Recommended guidance

Use SQL calls in Service Suite only when tests need controlled database validation or setup. SQL calls must follow least privilege and should avoid destructive operations unless they are explicitly approved and reversible.

What to define

- Database connection, schema, query, parameters, and timeout.
- Read-only versus write behavior and required permissions.
- Expected row count, values, or business validation criteria.
- Cleanup, rollback, and audit expectations for any data changes.

Best practices

1. Use test databases and controlled data whenever possible.
2. Never store passwords or production customer data in unsecured examples.
3. Validate SQL outside the main suite before using it in automated runs.
4. Review query performance and logs when SQL calls affect execution time.

Last updated: October 7, 2026

OCR Support

Using OCR requires tesseract installed in Linux or Mac. See Service Suitefor more information.

doOcr(byte[] data) → String

Reads an image and finds the string in that image

doOcr(InputStream data) → String

Reads an image and finds the string in that image

doHtmlOcr(byte[] data) → HtmlPage

Reads an image and finds text and xy position of the texts

doHtmlOcr(InputStream data) → HtmlPage

Reads an image and finds text and XY position of the texts

Last updated: October 16, 2026

Image Recognition

findImageXY(Object target) → NativeObjec

Get X and Y coordinates if the target match with the source

FI2

```
var target = readImage("{mds}/Install.png");
var entry = findImageXY(target);
```

findImageXY(Object target, BufferedImage source) → NativeArray

Read the image target and find the position that matches in the image source, and return an object with the X and Y positions.

F1

```
var target = readImage("{mds}/Install.png");
var source = readImage("{mds}/Screen.png");
var entry = findImageXY(target, source);
```

findImageXY(Object[] targets, BufferedImage source) → NativeArray

For each image target, find the position that matches the image source, and return objects with each X and Y position.

FI2

```
var source = readImage("{mds}/Screen.png");
var install = readImage("{mds}/Install.png");
var flecha = readImage("{mds}/flecha.png");
var adp = readImage("{mds}/adp.png");

var targets = [flecha, install, adp];
var listOfOptions = findImageXY(targets, source);
```

findImageXY(NativeArray targets) → NativeArray

Takes source and get X Y coordinates of first matching target if any target exists

FI3

```
var install = readImage("{mds}/Install.png");
var flecha = readImage("{mds}/flecha.png");
var adp = readImage("{mds}/adp.png");

var targets = [flecha, install, adp];
var listOfOptions = findImageXY(targets);
```

findImageXY(NativeArray targets, BufferedImage source) → NativeArray *

Get X Y coordinates of first matching target if any against the source

FI4

```
var source = readImage("{mds}/Screen.png");
var install = readImage("{mds}/Install.png");
var flecha = readImage("{mds}/flecha.png");
var adp = readImage("{mds}/adp.png");

var targets = [flecha, install, adp];
var listOfOptions = findImageXY(targets, source);
```

findImageXY(ArrayList targets) → NativeArray *

Takes source and get X Y coordinates of first matching target if any

FI5

```
var install = readImage("{mds}/Install.png");
var flecha = readImage("{mds}/flecha.png");
var adp = readImage("{mds}/adp.png");

var targets = [flecha, install, adp];
var listOfOptions = findImageXY(targets);
```

findImageXY(ArrayList targets, BufferedImage source) → NativeArray *

Gets X Y coordinates of first matching target if any against the source

FI6

```
var source = readImage("{mds}/Screen.png");
var install = readImage("{mds}/Install.png");
var flecha = readImage("{mds}/flecha.png");
var adp = readImage("{mds}/adp.png");

var targets = [flecha, install, adp];
var listOfOptions = findImageXY(targets, source);
```

findImageXY(Object[] targets) → NativeArray

Takes source and get X Y coordinates of first matching target if any

Fi7

```
var install = readImage("{mds}/Install.png");
var flecha = readImage("{mds}/flecha.png");
var adp = readImage("{mds}/adp.png");

var targets = [flecha, install, adp];
var listOfOptions = findImageXY(targets);
```

findImageXY(String[] targets, BufferedImage source)→ NativeArray

Get XY coordinates using OCR of the first matching target if any

FI8

```
var targets = ['Hello','World'];
var source = readImage("{mds}/Screen.png");
var listOfOptions = findImageXY(targets, source);
```

setImgComparisonType(String type)

Set the value of the different image comparison type ("ocr", "bitmap","openImage")

SET IMAGE COMPARISON

```
setImgComparisonType('openImage');
...
setImgComparisonType('ocr');
...
```

```
setImgComparisonType('bitmap');
```

findImageXY(String target, BufferedImage source) → NativeObject

Get the String target and find the position that matches the image source, and return an object with the X and Y positions.

FI9

```
var targets = 'Hello World';
var source = readImage("{mds}/Screen.png");
var entrey = findImageXY(target, source);
```

NTLM Authentication

setNTLMCreds(Object: creds)

Creates an NTLM configuration. Pass an object containing these named values:

- `workstation`
- `domain`
- `username`
- `password`
- `enabled`

Last updated: October 15, 2026

If you have feedback on this topic, contact documentation@appvance.ai.

Proxy Support

Service Suite supports standard proxy through

setProxy(String:hostname, int:port, String:username, String:password)

Sets standard proxy through

cleanProxy()

Deletes any proxy details.

setPac(String:url)

Sets proxy to use a .pac file. Example: setPac("file:///C:/proxy.pac");

setPac(String:URL, String:username, String:password)

Sets proxy to use a .pac file and provide credentials

setPac(String:url)

Sets proxy to use a .pac file.

resetConnections()

Resets the proxy (if any) and releases all sockets from established connections.

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

Last updated: October 15, 2026

Mobile Calls

Mobile calls need to have a start() call.

start() → void

Initiates a mobile session.

close() → void

Stops a mobile session.

For an idea of UX mobile testing capabilities see Sample Mobile Scripts

---

Last updated: October 16, 2026

Test Designer Integration

Service Suite can integrate with a test designer for basic functionalities.

forward (call→ HttpRequest, String → hostname, int hostport) → Http Response

Will proxy this call through a DS3 Server, Similar to **proxy**. Only works for DS3, Even if DS3 is injecting JS for testing, this call will not get modified.

startForward (String → hostname, int hostport)

Will create a new forward environment, please remember to stop forward and the end with stopForward. Only one forward session at a time. If this method is called without the forward session being closed, it will throw an exception.

This only affects calls using the forward(call). Standard process or HTTP calls are not affected.

forward (call→ HttpRequest) → Http Response

Call that will be called using the forward specs in startForward. Will throw exception if there is not forward session started.

stopForward()

Stops a forward session. Will throw exception if there is not forward session started.

startTestDesignerProxy (String host, int port) → Int

Starts a Test Designer proxy session will return the proxy socket port. This port will not inject JS.

startTestDesignerProxy (String host, int port,int desiredPort) → Boolean

Starts a Test Designer proxy session in the desired port and will return the proxy socket port. This port will not inject JS.

startTestDesignerProxyRercording(String host, int port,int desiredPort) → Boolean

Make proxy session in desired port to start recording. Could be download with consumeTestDesignerProxyHAR. To stop the recording use stopTestDesignerProxyRercording. Returns true if the session exists and now recording.

stopTestDesignerProxyRercording(String host, int port,int desiredPort) → Boolean

Stops recording in proxy session for desired port. Could be download with consumeTestDesignerProxyHAR. To restart the recording use startTestDesignerProxyRercording. Returns true if the session exists and now not recording.

consumeTestDesignerProxyHAR(String host, int port,int desiredPort) →JSONArray

Consumes the recording session in the HAR proxy. To enable recording in a session use startTestDesignerProxyRercording. To stop the recording use stopTestDesignerProxyRercording.

stopTestDesignerProxy(String host, int port,int desiredPort)

Stops a Test Designer proxy session.

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

Last updated: October 15, 2026

Screen Interaction (FFMPEG required)

FFMpeg Should be on the path for this method to work

startScreenRecording(fps:Integer, filename:String)

Starts a screen recording saved to the filename, should be stop with stopScreenRecording

stopScreenRecording()

Stops screen recording. A recording should be already started with startScreenRecording.

extractFrame(String: video, frame:Integer, Output:String)

Extracts a frame from the video and saves it in the Output file. File format should be based on the output extension.

extractFrames(String: video, frame:Integer, int count: OutputFormat:String)

Extracts up to count frames from the video and saves it in the files based on the output format. Extracts file format should be based on the output extension.

Format details can be found here https://ffmpeg.org/ffmpeg-formats.html

---

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

Last updated: October 15, 2026

OS Screen Interaction

startUX(void)

Enables interaction with the operating-system screen where Service Suite is running. Throws an exception when UX access is unavailable or forbidden on the server.

clickXY(int: x, int: y) throws Exception

Clicks the specified screen coordinates. Throws an exception when `startUX` has not been enabled.

Last updated: October 15, 2026

If you have feedback on this topic, contact documentation@appvance.ai.

Excel Support

Service suite offers excellent Excel support from the lead open source POI project

Basic calls are

readExcel(Path:String) → ExcelWrapper

Read and excel and returns the Excel Wrapper object, for Excel Wrapper methods, check the Excel Wrapper section above. Keep in mind you can use “{ds}” to indicate path relative to the current script.

writeExcel(Data: ExcelWrapper, Path:String) → ExcelWrapper

Writes an excel wrapper to disk, keep in mind you can use “{ds}” to indicate path relative to the current script. This functionality works in IDE but not during scenario execution.

createNewExcel(FirstSheeName:String) → ExcelWrapper

Creates a new excel wrapper with one sheet. For Excel Wrapper methods, check the Excel Wrapper section above.

ExcelWrapper

Excel wrapper holds a POI 5.0.0 document. It has basic cell editing, for full POI documentation

https://poi.apache.org/

setValueAt(row:int, column:int, value:Object)

Sets a value in a specific cell

```
var sheet = readExcel("myExcel.xlsx");
sheet.setValueAt(2,1,"P0000002643");
```

getValueAt(row:int, column:int) → value:String

Gets the value from a specific cell in String format. If Cell is of type NUMERIC an exception will be thrown.

```
var sheet = readExcel("myExcel.xlsx");
log("Value: "+sheet.getValueAt(2,1));
```

For getting the String representation of a cell of type numeric or date can use **toString** method of XSSFCell object returned by **getCellAt** (see below).

```
var cell = sheet.getCellAt(2,5);
log("cell string representation: "+cell.toString());
```

getCellAt(row:int, column:int) → XSSFCell

Gets the XSSFCell object. More info can be found in XSSFCell

```
var sheet = readExcel("myExcel.xlsx");
var dateCell = sheet.getCellAt(2,5);
log("Date: "+dateCell.toString());
```

```
// COPYING A CELL DATE
var excel = readExcel("myExcel.xlsx");
var cell25 = excel.getCellAt(2, 5);  // value is Date 01-Jan-2020
log("2,5 date value: " + cell25.toString());
cell27 = excel.getCellAt(2, 7)
log("Cell 2,7 raw value: " + cell27.getRawValue());  // null

var policy = org.apache.poi.ss.usermodel.CellCopyPolicy();
cell27.copyCellFrom(cell25,policy);
log("Cell 2,7 value after copying cell 2,5: " + cell27.toString());  // 01-Jan-2020
```

getWorkbook() → XSSFWorkbook

Gets the workbook, full documentation about workbook can be found

https://poi.apache.org/apidocs/dev/org/apache/poi/xssf/usermodel/XSSFWorkbook.html

setBooleanValueAt(row:int, column:int, value:boolean) → void

Set a cell with a boolean value

```
var sheet = readExcel("myExcel.xlsx");
sheet.setBooleanValueAt(2,3,true);
assertEquals(true, sheet.getBooleanValueAt(2,3))
```

getBooleanValueAt(row:int, column:int) → value:boolean

Get boolean value of a cell

```
var sheet = readExcel("myExcel.xlsx");
sheet.setBooleanValueAt(2,3,true);
assertEquals(true, sheet.getBooleanValueAt(2,3))
```

setFormulaValueAt(row:int, column:int, formula:string) → void

Set a cell’s formula

```
var sheet = readExcel("myExcel.xlsx");
sheet.setFormulaValueAt(3,3,'AVERAGE("E1:G1")');
assertEquals('AVERAGE("E1:G1")', sheet.getFormulaAt(3,3))
```

getFormulaAt(row:int, column:int) → formula:string

Get a cell’s formula as a string

```
var sheet = readExcel("myExcel.xlsx");
sheet.setFormulaValueAt(3,3,'AVERAGE("E1:G1")');
assertEquals('AVERAGE("E1:G1")', sheet.getFormulaAt(3,3))
```

setCellType(row:int, column:int, formula:string, CellType value) → void

Set a cell’s cell type as in Cell Type

```
var sheet = readExcel("myExcel.xlsx");
var cell = sheet.getCellAt(2,4);
cell.setCellType(org.apache.poi.ss.usermodel.CellType.BLANK);
```

getCellType(row:int, column:int) → formula:CellType

Get a cell’s Cell Type

```
var sheet = readExcel("myExcel.xlsx");
assertEquals('NUMERIC',sheet.getCellType(2,3).toString());
```

setDateValueAt(row:int, column:int, value:Date) → void

Set a cell’s cell type as in Cell Type

```
var sheet = readExcel("myExcel.xlsx");
var cell = sheet.setCellDateAt(2,4, (new java.util.Date).now());
```

getDateValueAt(row:int, column:int) → date:Date

Get a cell’s Date object

```
var sheet = readExcel("myExcel.xlsx");
log('NUMERIC',sheet.getDateValueAt(2,3));
```

setNumericValueAt(row:int, column:int, formula:string, value:double) → void

Set a cell’s cell numeric value

```
var sheet = readExcel("myExcel.xlsx");
var cell = sheet.setNumericValueAt(2,5, 43831);
```

getNumericValueAt(row:int, column:int) → value:double

Get a cell’s numeric value as a string

```
var sheet = readExcel("myExcel.xlsx");
log(43831.0,sheet.getDateValueAt(2,3));
```

getRawValueAt(row:int, column:int) → value:string

Get a cell’s raw value formatted as a string

```
var sheet = readExcel("myExcel.xlsx");
assertEquals('43831.0',sheet.getRawValueAt(2,3));
```

getFormattedValueAt(row:int, column:int) → value:string

Get a cell’s formatted value as a string

```
var sheet = readExcel("myExcel.xlsx");
assertEquals('1/1/20',sheet.getFormatetedValueAt(2,3));
```

setDateStyleAt(row:int, column:int, format:string) → void

Sets a cell’s format style

```
var workbook = createNewExcel("Hello World");
workbook.setNumericValueAt(2,2,43831);  // Jan/01/2020
workbook.setDateStyleAt(2,2,"mm/dd/yy");
assertEquals("01/01/20", workbook.getFormattedValueAt(2,2));
```

S3 Client

Service Suite can use Amazon S3 as a file repository. Create an S3 client with:

```
s3client = createS3Client(filename)
```

The properties file contains the access key and secret key values:

```
AWS_ACCESS_KEY_ID=xxxxxxx
AWS_SECRET_ACCESS_KEY=yyyyyyyyyyyyyyyyyyyyyyy
```

These are standard AWS access keys. See AWS security credential types.

S3 Client methods

- `download(bucket:String, remoteName:String) → byte[]`
- `downloadFile(bucket:String, remoteName:String, localpath:String)`
- `list(bucket:String, prefix:String) → Set<String>`
- `listFolders(bucket:String, prefix:String) → Set<String>`
- `listFolderNames(bucket:String, prefix:String) → Set<String>`
- `upload(bucket:String, data:byte[], remoteName:String)`
- `uploadPublic(bucket:String, data:byte[], remoteName:String)`

```
var client = createS3Client("{ds}/s3.creds");
var bucketName = "myBucket"; // this must exist in S3
var data = "user,pass\njohn,doe";
// client.delete(bucketName, "test.csv"); // delete previous file
client.upload(bucketName, toByteArray(data), "test.csv");
```

Last updated: October 15, 2026

If you have feedback on this topic, contact documentation@appvance.ai.

Binary Data

decodeBase64(text: String) → byte[]

Decodes a Base64-encoded string and returns a byte array.

encodeBase64(byte[]: data) → String

Encodes a byte array and returns a Base64 string.

Last updated: October 15, 2026

If you have feedback on this topic, contact documentation@appvance.ai.

Git Client

Services Workbench allows you to use handle git files from the script itself. This is useful for file complex manager or using it during scenario execution as repository system is not available during the scenario execution.

In order to create an Git Client from Services Workbench, first you need to create a client providing 4 required parameters, username, password, the remote repository URL and the local directory. Directories are recommended to have the “{ds}” to make them relative to the current working file.

The client will create the folder during the git client creation, and if the file is there but also an empty folder it is ok.

Keep in mid if a file exists or if the folder is empty, creating the git file creation will fail

```
gitClient = createGitClient($username,$password,$url,"{ds}/localpath"[,$branch])
writeCSV("{ds}/localpath/data.csv",$csv_data)
gitClient.add(".");//Adding the file data.csv and other files
gitClient.commit("Message");//Commits message
gitClient.push();//Push but could fail if the branch is not updated
```