3.1 - Services Workbench

Service Suite documentation.

- Service Suite Overview
- Why Service Suite?
- Netboard Service Usage in Service Suite
- Run Node.js Packages from Service Suite
- WSDL/SOAP Call-in Service Suite

Service Suite Overview

Contents

- Why Service Suite?
- Netboard Service
- Node Package Manager Support
- WSDL/SOAP Call-in Service Suite

Use your UX Scripts to generate Service Suite Scripts.

- Introduction
- Logs
- Auto Scroll Logs
- Using UX Scripts to Generate Service Suite Scripts

Introduction

Service Suite is an HTTP API testing technology using standardized Javascript syntax, that was written from scratch with the goal to make API testing simple to use but very powerful.

Can be used for functional testing and debugging and to launch performance tests with thousands or millions of virtual users integrating with the current AIQ scenario editor and usage of cloud-based test nodes.

Service Suite has an extensive list of actions ready to be used for API Calls, SQL Calls, IOT, microservices, and Mobile Calls.

Note: Please be careful when using SoapUI with Service Suite in the same AIQ Scenario, because of incompatibilities among them.

SoapUI support was removed in release 5.2.0 of AIQ.

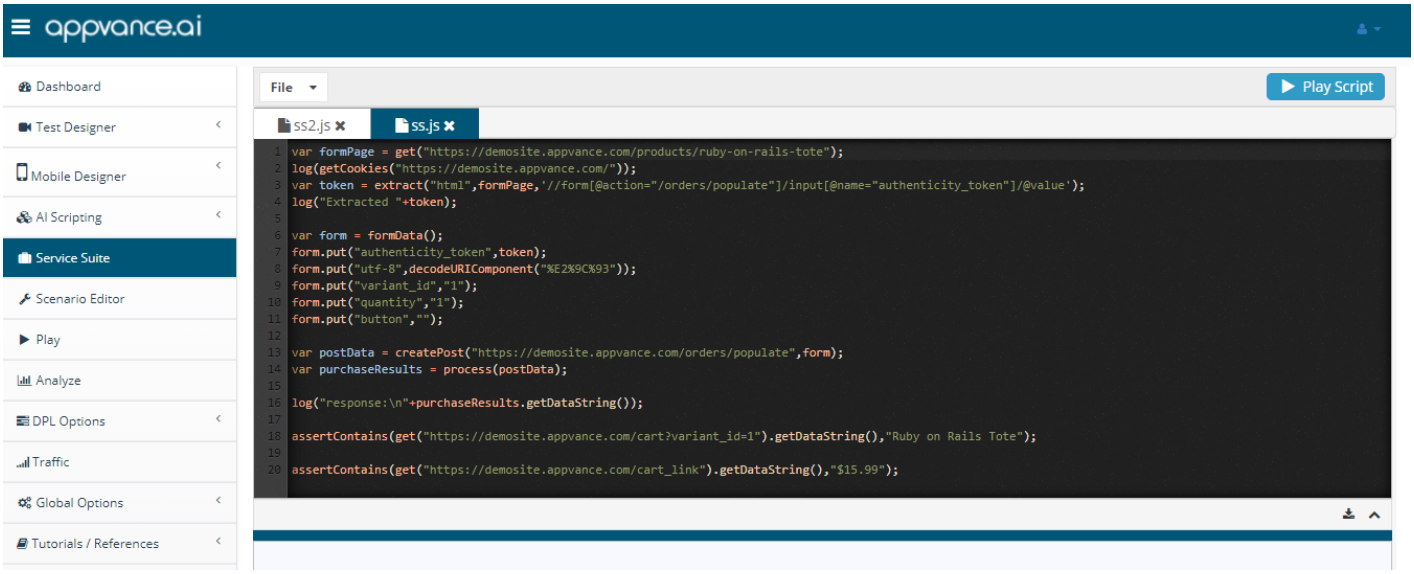
With Service Suite, You can open multiple files which open in different tabs which makes it easy for the user to copy contents between different scripts.

Control+S will automatically save the script.

Control+O will open the file manager to open the files.

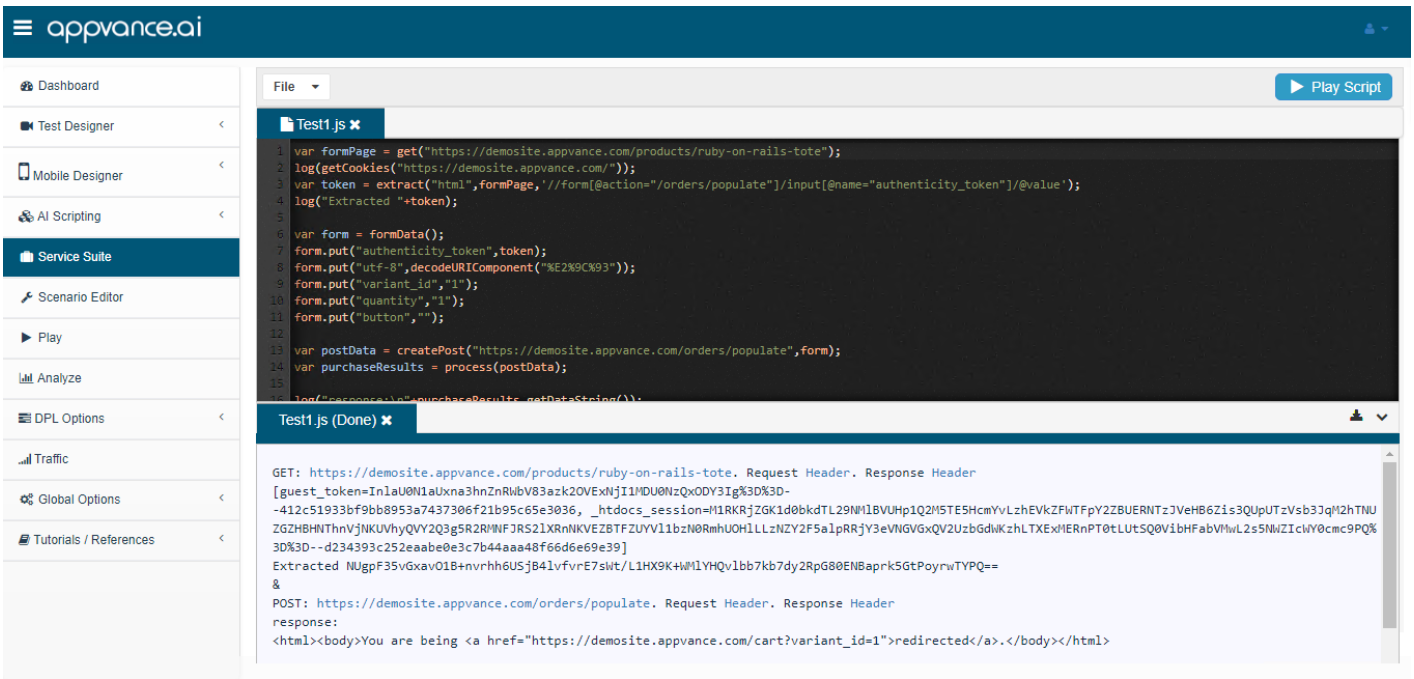
There is IntelliSense, as you type in the default methods and functions would be available to select from the drop-down including custom-defined variables and functions.

It is required to have Test Designer services running to work with Service Suite.



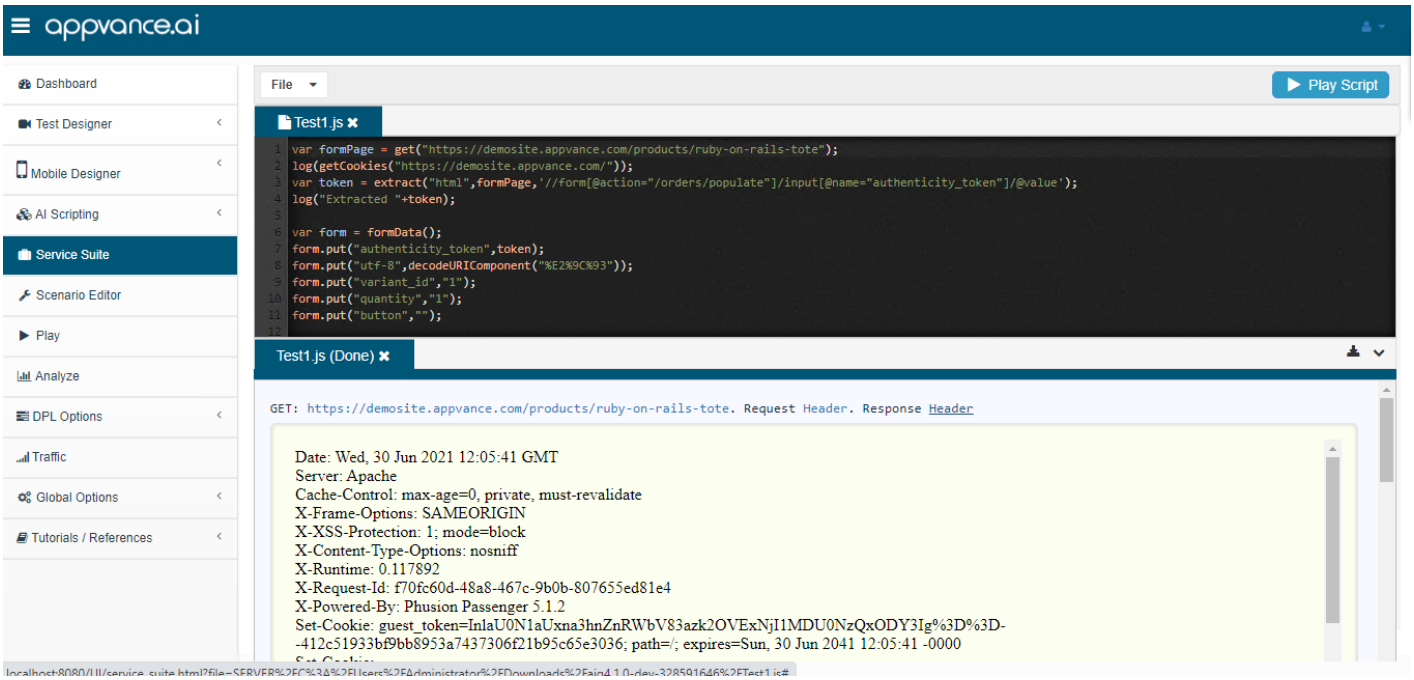
Logs

Logs can be checked once the script is running or finished running, there is a separator to drag and drop to check the logs.



There is also a toggle button to switch back and forth between the script screen and the logs.

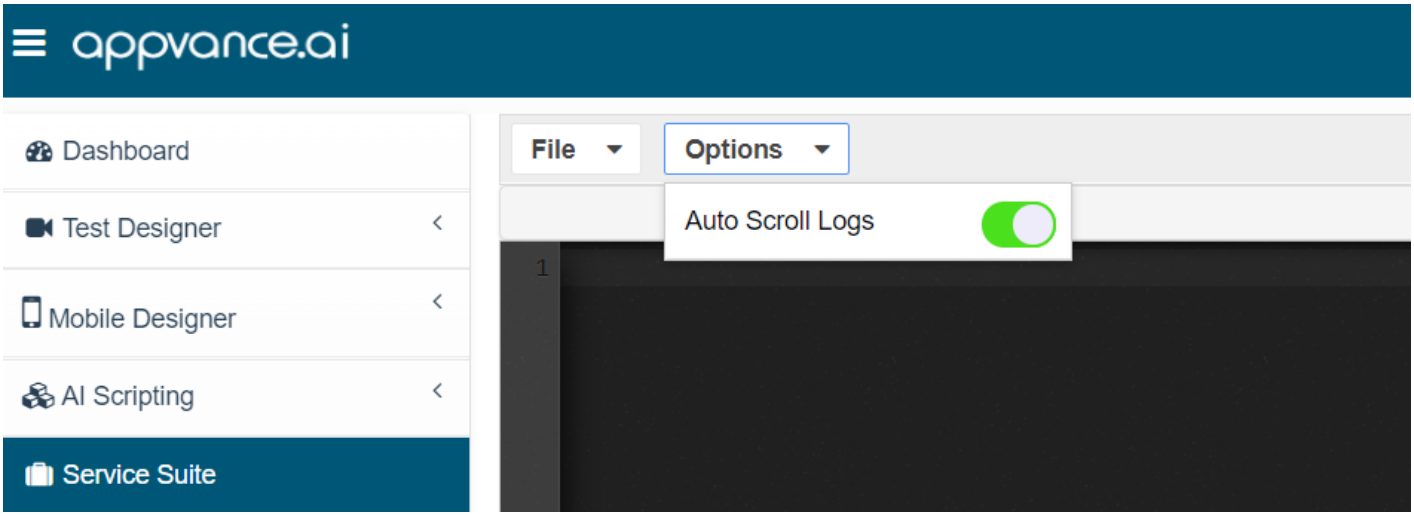
From the logs, you would be able to see the response for the GET or POST requests and would be able to download them, Request Header can be clicked to see the request headers, response headers can be clicked to see the response headers.



The download button in the logs section can be used to download the logs.

Auto Scroll logs

Auto Scroll logs from the Options menu can be turned on for the logs to Auto Scroll as the script replays.



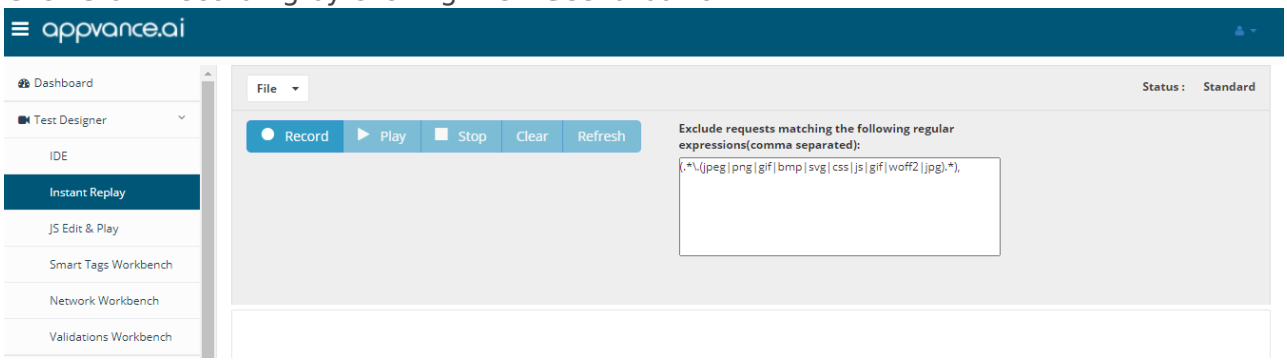
Use your UX Scripts to generate Service Suite Scripts

Yes, it is possible to create a Service Suite API script using an already created UX Script using Test designer.

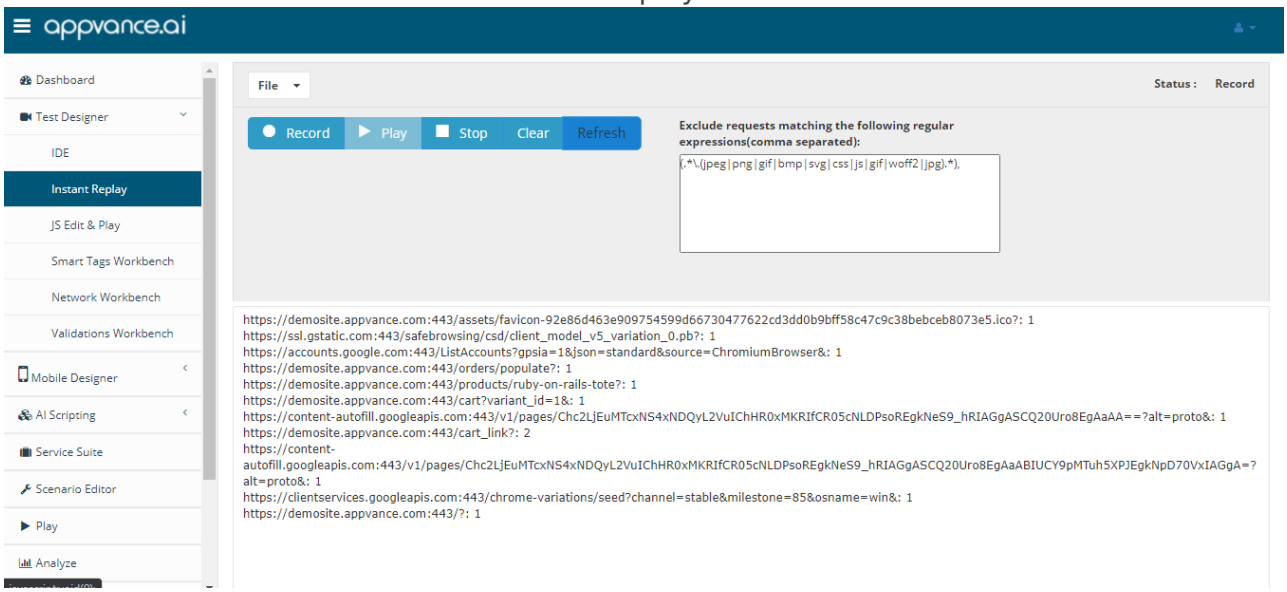
Below steps:

Make sure the Test designer services are running.

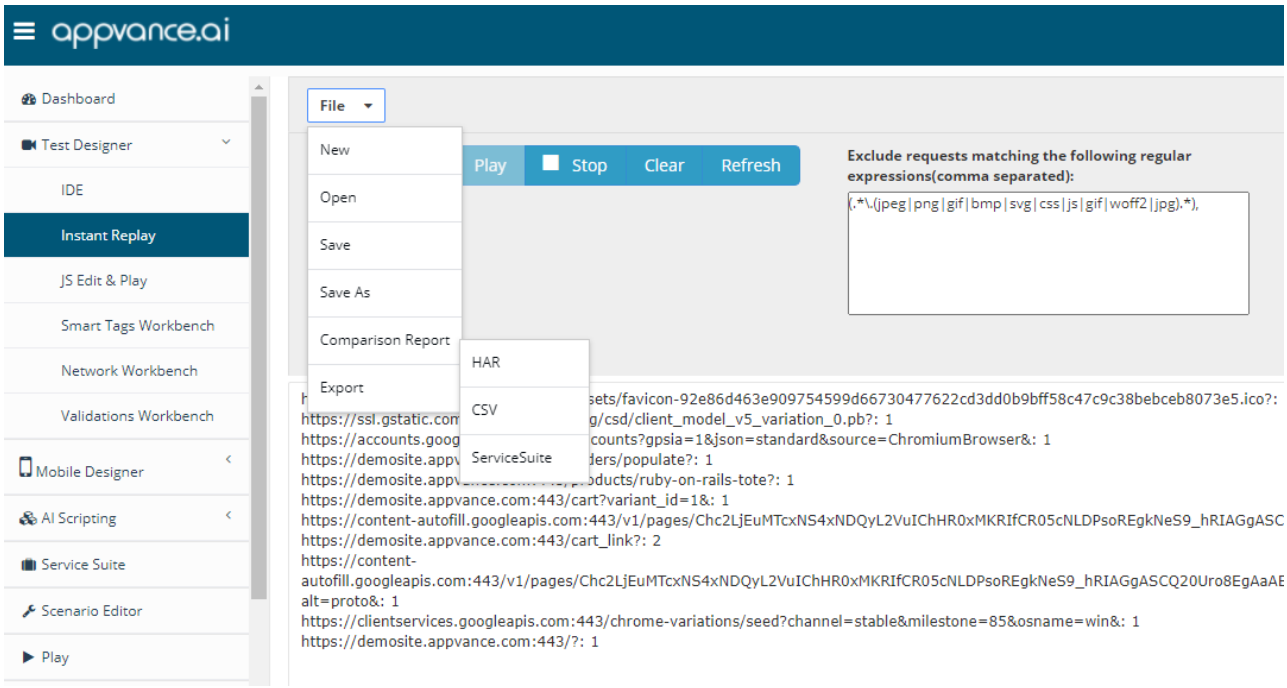
1. Open the saved Test designer in IDE.
2. Open Instant Replay in a different tab.
3. Click Start Recording by clicking the **Record** button.



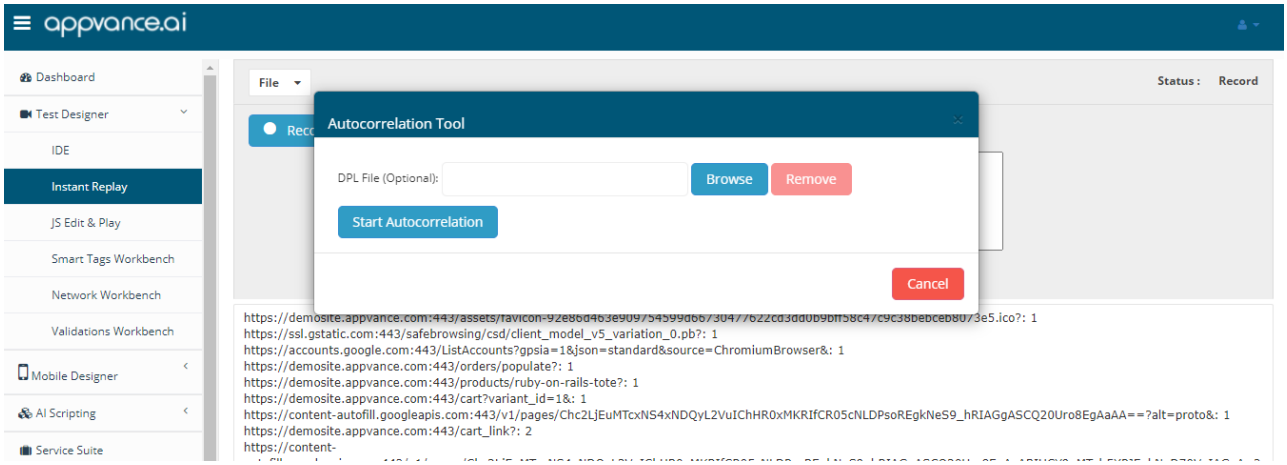
4. Go back to Test Designer IDE and Click Play
5. Once the script is played back fine, go to the Instant replay tab and click Refresh
6. All the traffic calls would be recorded and displayed.



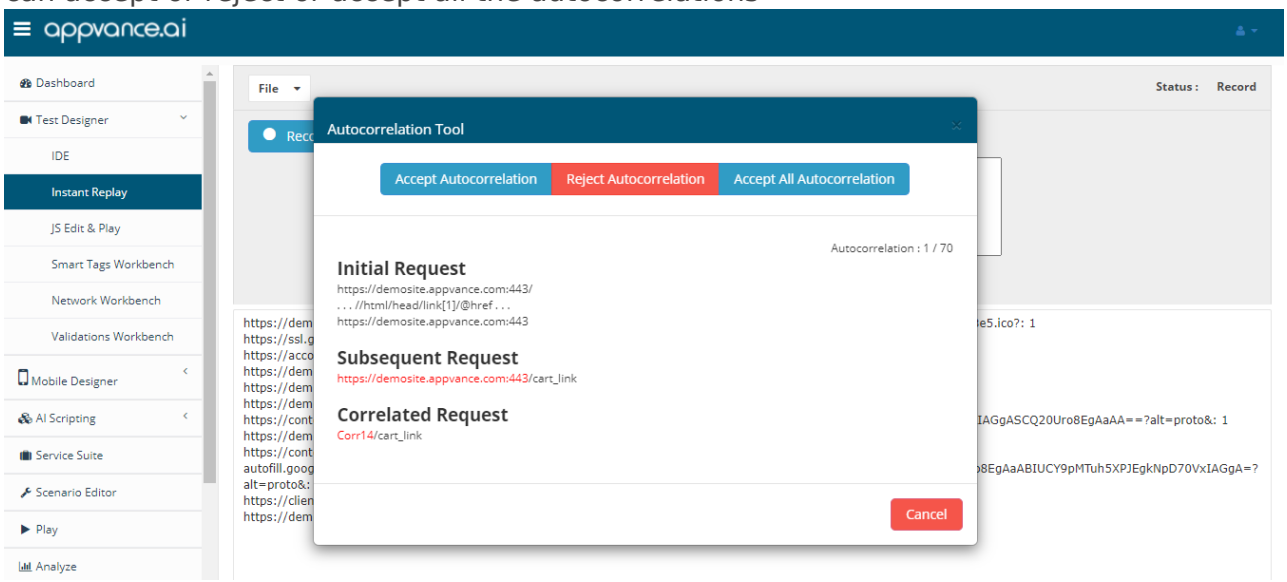
7. Click File > Export > Service Suite
   This option will be removed in the AIQ 4.10.0 release.



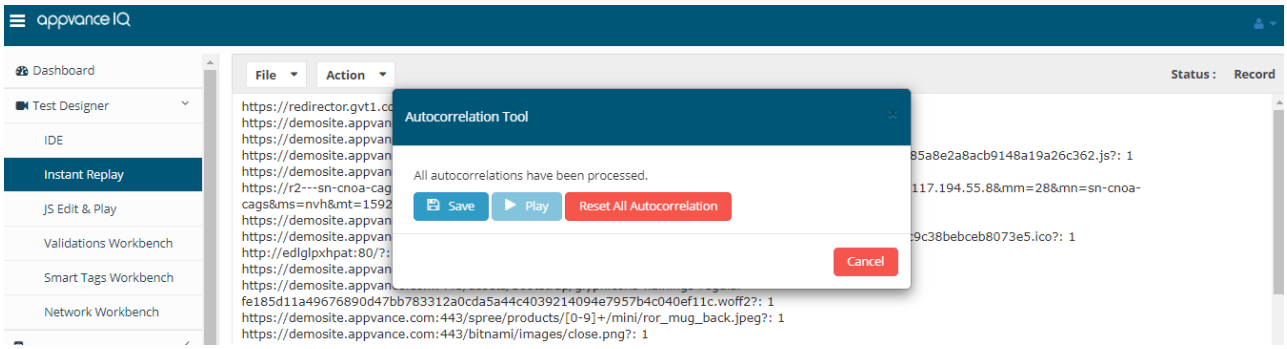
8. Optional Step - If there is DPL File that was used for the script, please browse the file else Click Start Autocorrelation



9. The autocorrelation tool captures all the autocorrelations and would be presented, you can accept or reject or accept all the autocorrelations

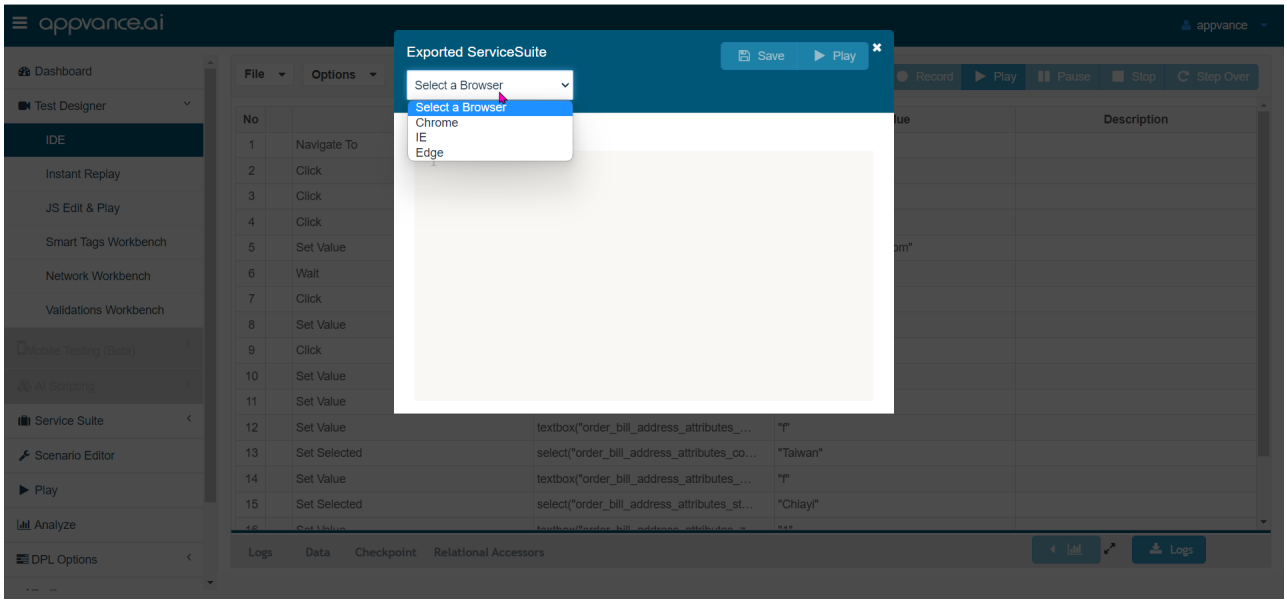


10. Once all the correlations are done, you will be presented to save the script or Reset autocorrelations

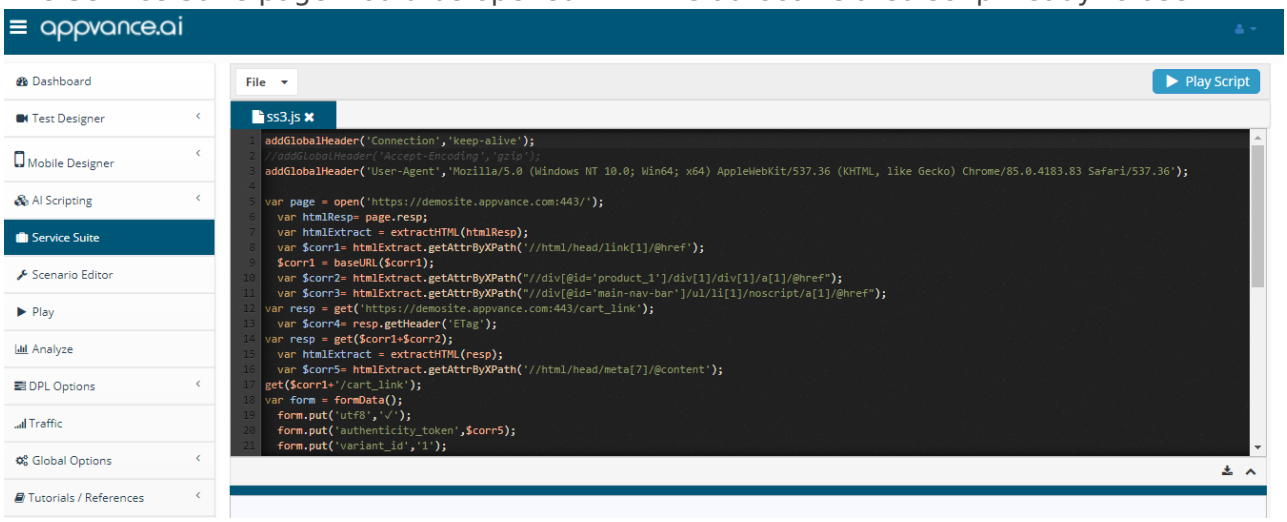


11. Click Save and Save the Service suite script
12. Select between available browsers so you can choose which browser will play the script.

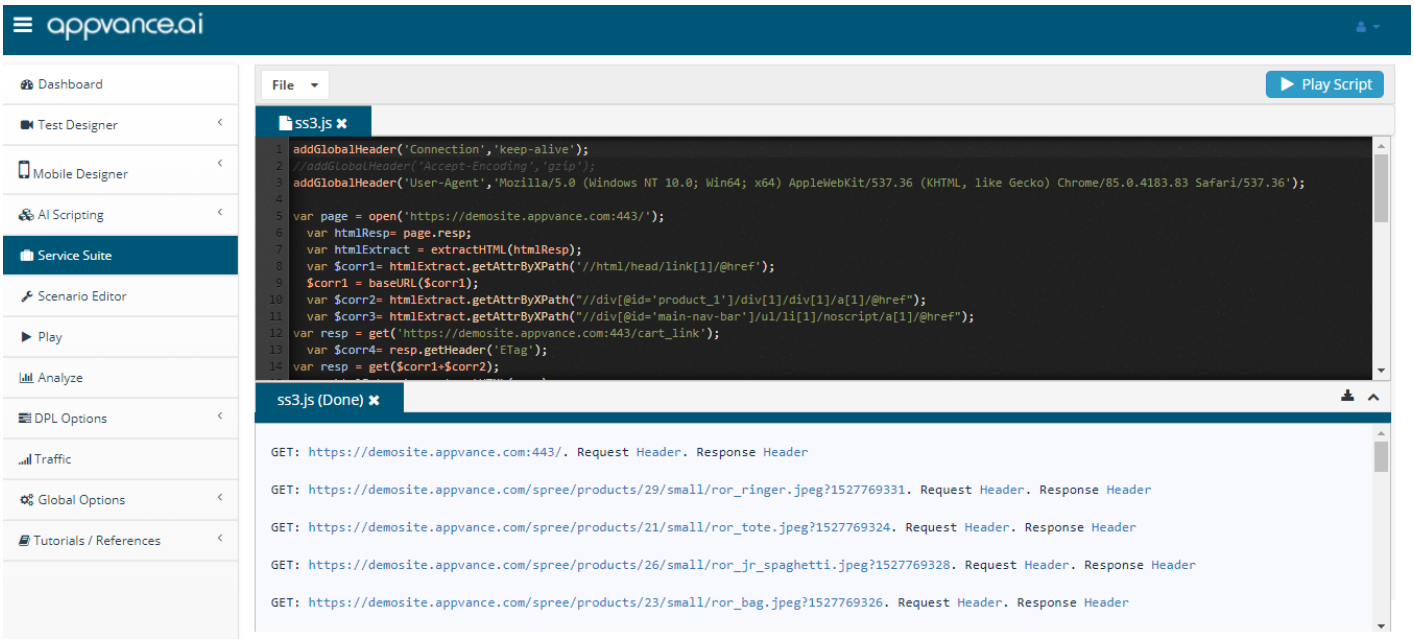
    This option was added in AIQ 4.9.2. See the release notes for Release 4.9.2.

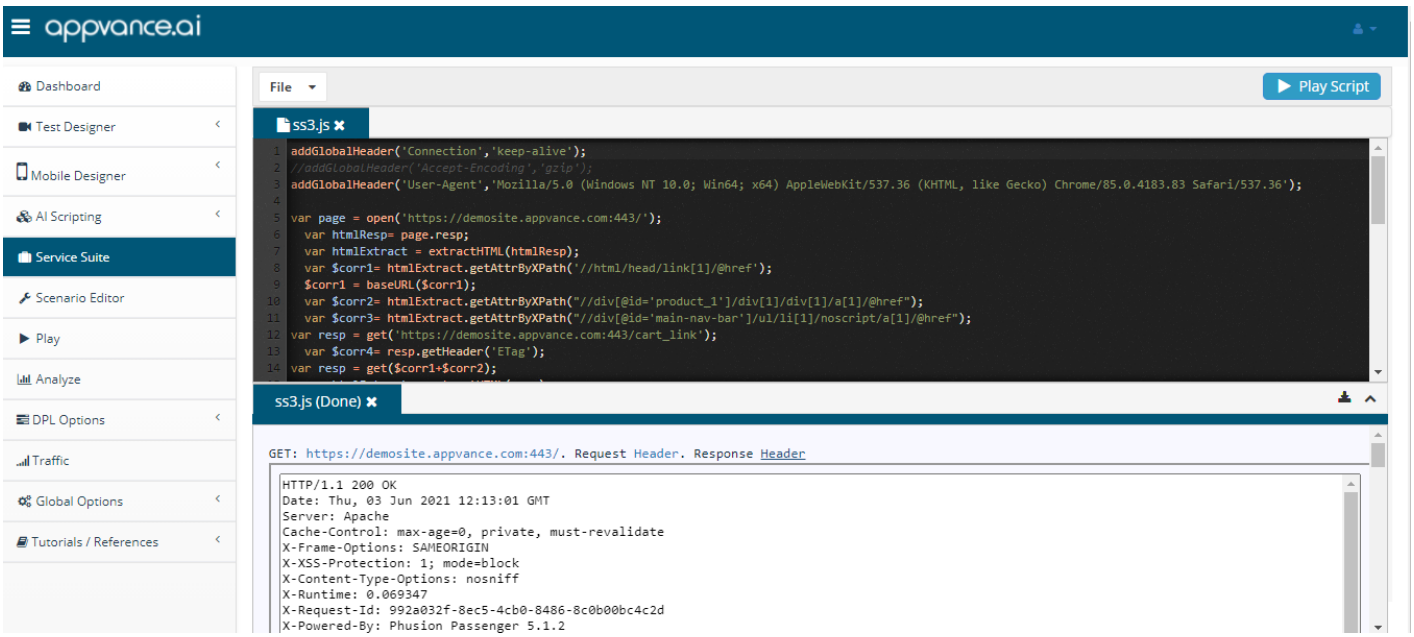


    IE is still available in the dropdown, but it is no longer supported. If you select it, the script will fail. The option will be removed in a future release.
13. Click Play
14. The Service Suite page would be opened with the autocorrelated script ready to use



15. Click Play for the script to playback and provide the results





Last updated: July 25, 2024

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

Why Service Suite?

- Written from Scratch to match current design standards
- Can be used as Functional, Debugging, and Performance
- Scripts are easy to maintain
- No duplication of effort
- Scripting for IoT, API, API from UX, microservices, backend services
- Easy to read – Full Standard JS support
- “Mobile Designer” provides full API level actions
- Get, Post, Put, Patch, Head, Options
- Sweet script level variables
- Great for API Testing
- Autocorrelation support
- Great for IoT simulation
- Easy to organize and launch tests via CI/CD
- Asserts & Extractions Built-in
  - JSON, HTML, CSV, PDF, XML
- Fully Integrated within Appvance
- Mix/match in scenarios with other script types
- Service Suite supports JS even PostMan code can be used with minor changes
  As of AIQ 4.8.3 Service Suite can convert JSON to Postman so that processes using node.js scripts can now be executed by AIQ.

A standard set of functions

- Handling externals files
- Built-in global functions
- DPL Managements
- Cookie Management
- Request creations
- Response parsing functions

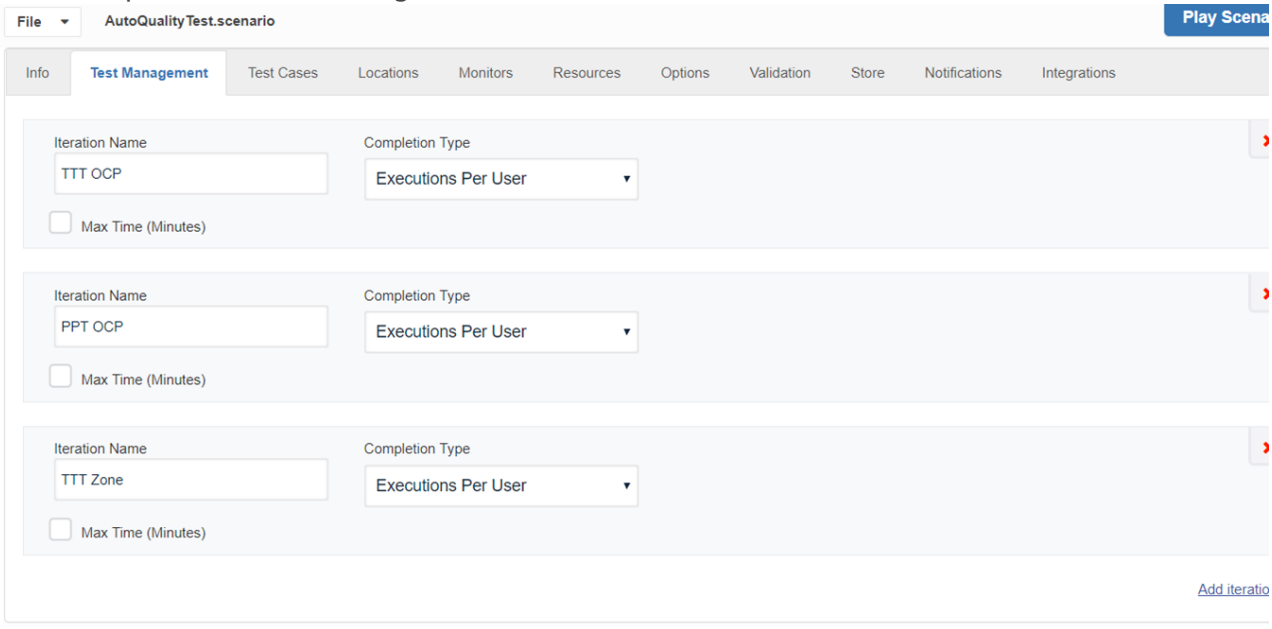
File

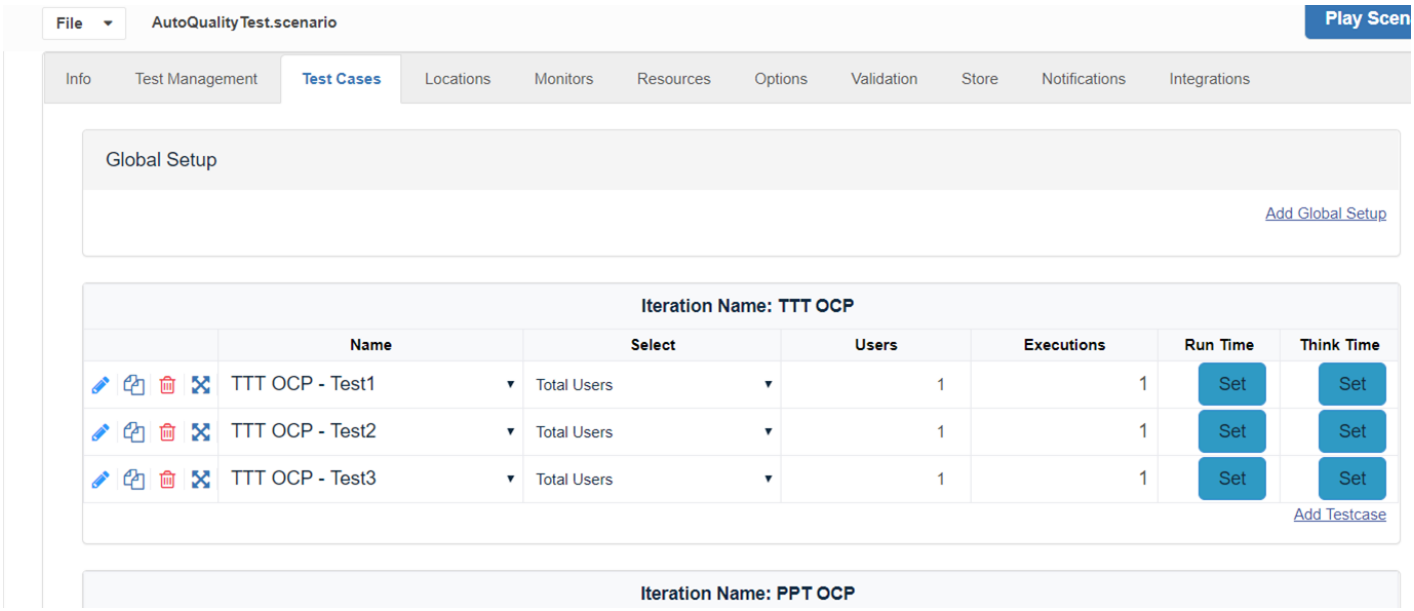
Configuration.js

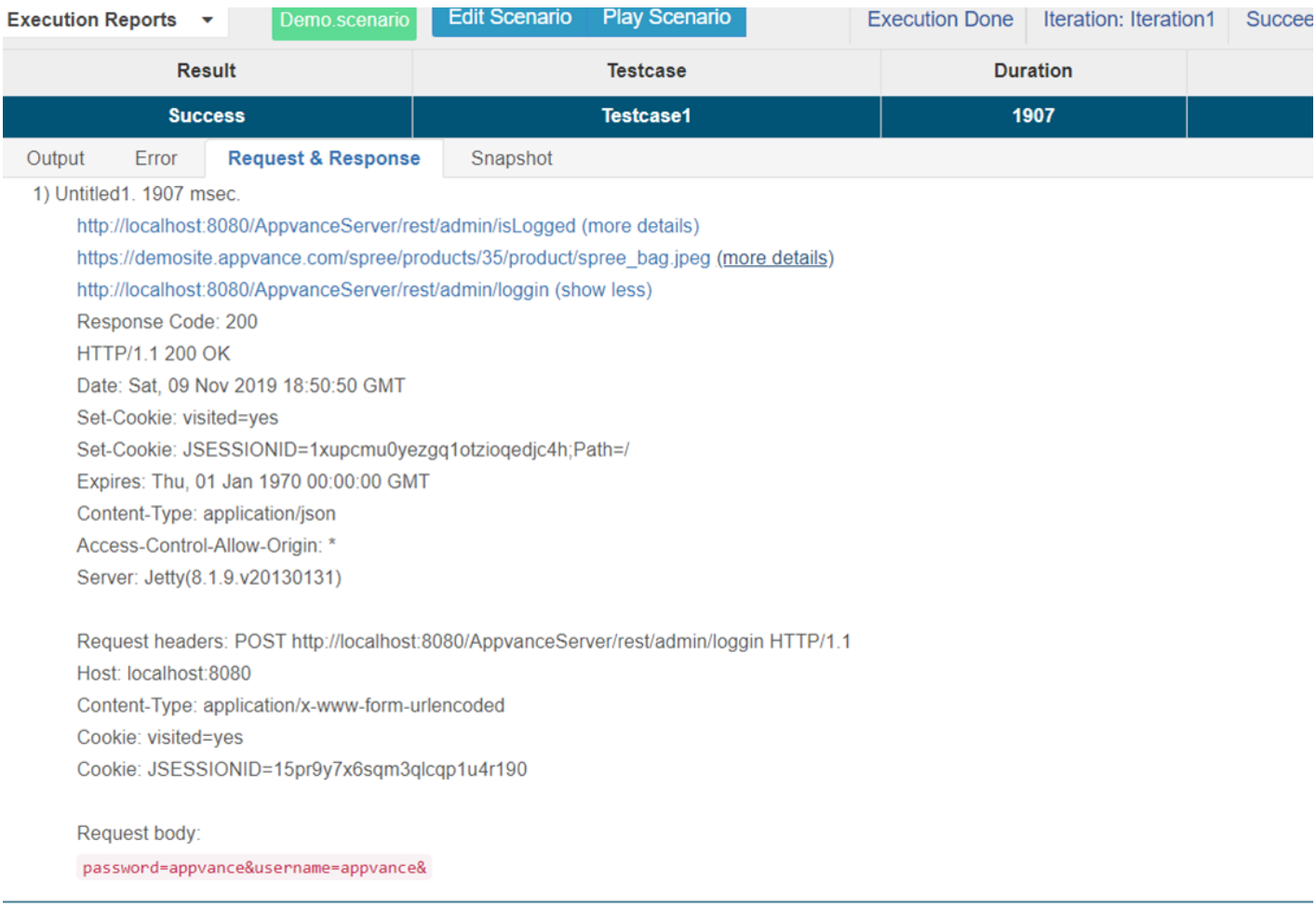
```
1
2  // --- Check for Global variables ---
3
4  if (typeof $token === "undefined"){
5    HashDPL("{ds}/env.csv");
6  }
7
8  //--- Define reusable functions ---
9
10 var jsonGet = function(url){
11     var data = createGet(url);
12     data.addHeader("Content-Type","application/json");
13     return data;
14 };
15
16
17 var addBearerToken=function(req){
18     req.addHeader("Authorization","Bearer "+$token);
19 };
20
```

Clean Scenario Level Executions and easy organizing

- Full AIQ integration
- Scenario, CI-CD
- DPL as variables
- Great for performance testing

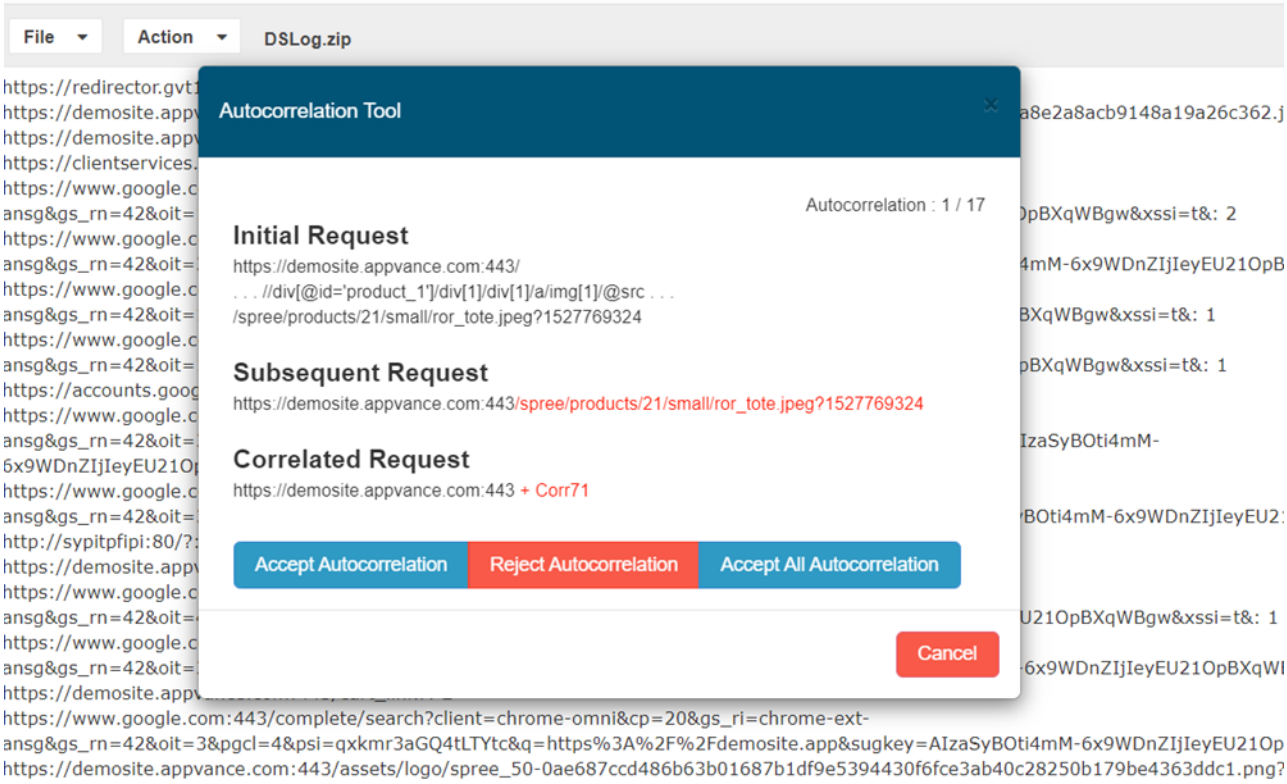






Autocorrelation for API Calls

- Record using port 9090
- Disable injections for API Testing
- Run correlations
- Validate some/all
- You have your script read



Service Suite, use both Configuration and Function Libraries

- Easy to maintain
- No duplication of effort
- Easy to read – Full Standard JS support
- Service Suite provides full API level actions
  - Get
  - Post
  - Put
  - Asserts
- Sweet script level variables

File ▾ TTT OCP - Test1.js

```
var libPath = "{ds}/../../Lib/"

include(libPath+"Configuration.js");
include(libPath+"Policy.js");
include(libPath+"Transaction.js");


///--- Create policy    ----
startStep("Create a Policy");
createPolicy();

///--- Create Transaction    ----
startStep("Create a Transaction");
createATransaction();

///--- Edit Transaction    ----
startStep("Edit a Transaction");
createATransaction();

```

Multiple DPLs allow us to create Cartesian combinations for configurations without extra coding

File ▾ Transaction.js

Multiple variables like this can come from different DPLs if required, making the famous cartesian product for configurations

```
var createATransaction = function(){

  log("Create A Transaction Starting");
  var createTransactionReq = jsonGet($partnerApiUrl+"/policies/"+$policyId+"/transactions");
  addBearerToken(createTransactionReq);

  var resultRequest= process(createTransactionReq);
  var result =resultRequest.getDataString();
  log(result);
  var jsonResult = JSON.parse(result) ;

  assertEquals("string",typeof jsonResult[0].transactionId );
  assertContains( jsonResult[0].transactionId ,"-");
  log(jsonResult[0].transactionId);
  log("Create A Transaction Success");
}

```

Summary

- Ease of use for multilevel data sets
- Scripts can be created from UX to Instant Replay to Service Suite
- Built to handle the Cartesian type of testing
- Substantially less scripting

- More reliance on tables than conditionals and scripts
- UltraTesting team can quickly be trained and up on this technology (it’s all JS)
- Overall test effort should be reduced by 70% to 90%
- Full support by the Appvance expert team

Last updated: January 02, 2025

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

Netboard Service Usage in Service Suite

The test designer’s Netboard service is an HTTP service that allows you to store and retrieve named data. The stored data is deleted after 1 hour and since it each entry is named, then if another set of key/values comes with the same name, then the previous value if existing, will get replaced with the new value. The concept is similar to a temporal hash map. So a Netboard is basically a named set of key/value pairs that are stored for 1 hour and then automatically cleared to avoid memory leaks.

Netboard Calls

To make a Netboard call take the AIQ server address but with port 9090, as this service is hosted in the Test Designer server. Example:

http://localhost:9090/ds/dyn/Netboard/<command>

create

Parameters: **board**

Description: creates an empty net board with id "395". This call is optional as the first **put** will create the named board if it does not exist.

http://localhost:9090/ds/dyn/Netboard/create?board=staging

put - POST

Parameters: **board**, **key,value**

Description: store into the “staging” Netboard a key/value pair data with key = “homepage“ and a JSON string as a value. If a board with the provided key does not exist, a new board is created automatically and the data gets added.

http://localhost:9090/ds/dyn/Netboard/put?board=staging&key=id&value=623465387

get

Parameters: **board**, **key**

Description: returns a String with the value for an entry named key in the board provided.

http://localhost:9090/ds/dyn/Netboard/get?board=staging&key=account

containsKey

Parameters: **board**, **key**

Description: returns **true** if the provided board contains an entry with the provided key.

http://localhost:9090/ds/dyn/Netboard/containsKey?board=staging&key=account

getKeys

Parameters: **board**

Description: returns a JSON string with all keys defined in the board named “staging“

isEmpty

Parameters: **board**

Description: returns **true** if the board is empty

clearBoard

Parameters: **board**

Description: clear all the contents of the board

deleteBoard

Parameters: **board**

Description: remove the Netboard

getBoards

Parameters: **none**

Description: returns a JSON string with all the currently existing board names.

Example Test Designer JS Script

Segment.com Integration using the Netboard service

A set of entry points has been implemented along with standard Netboard calls to implement a webhook integration that allows automatic store of http://segment.com events. Segment.com events have JSON data. Need to configure any of the below entry points as webhooks inside http://segment.com configurations. The data received from those entry points will be processed in

a way that received events data will be automatically named and put in corresponding Netboards.

```
_setDomain("http://localhost:9090");
get("/ds/dyn/Netboard/resetBoards");
get("/ds/dyn/Netboard/create?board=hitrecord");
get("/ds/dyn/Netboard/put?board=hitrecord&key=State&value=Alaska");
get("/ds/dyn/Netboard/put?board=hitrecord&key=Country&value=USA");
var $state = get("/ds/dyn/Netboard/get?board=hitrecord&key=State").getDataString();
log($state);
assertEquals($state,"Alaska");
var $contains = get("/ds/dyn/Netboard/containsKey?board=hitrecord&key=State").getDataString();
log($contains);
assertTrue($contains);
var $fields = JSON.parse(get("/ds/dyn/Netboard/getKeys?board=hitrecord").getDataString());
log($fields[0]);
assertEquals("State",$fields[0]);
log($fields[1]);
assertEquals("Country",$fields[1]);
var $isEmpty = get("/ds/dyn/Netboard/isEmpty?board=hitrecord").getDataString();
log("isEmpty: "+$isEmpty)
assertTrue($isEmpty=="false");
get("/ds/dyn/Netboard/create?board=costa%20rica");
get("/ds/dyn/Netboard/put?board=costa%20rica&key=State&value=San%20Jose");
get("/ds/dyn/Netboard/put?board=costa%20rica&key=Country&value=Costa%20Rica");
var $boards = JSON.parse(get("/ds/dyn/Netboard/getBoards").getDataString());
log($boards[0]);
assertEquals("hitrecord",$boards[0]);
assertEquals("costa rica",$boards[1]);
var $text = get("/ds/dyn/Netboard/removeOldEntries").getDataString();
log($text);
assertContains($text,"Removing data from Netboard older than");
get("/ds/dyn/Netboard/clearBoard?board=hitrecord");
$fields = JSON.parse(get("/ds/dyn/Netboard/getKeys?board=hitrecord").getDataString());
log($fields.length);
assertEquals(0,$fields.length);
get("/ds/dyn/Netboard/deleteBoard?board=hitrecord");
$fields = JSON.parse(get("/ds/dyn/Netboard/getBoards").getDataString());
log($fields.length);
assertEquals(1,$fields.length);
get("/ds/dyn/Netboard/deleteBoard?board=costa%20rica");
$fields = JSON.parse(get("/ds/dyn/Netboard/getBoards").getDataString());
log($fields.length);
assertEquals(0,$fields.length);
```

Segment.com Entry points

There are 5 independent entry points, each one capable of admin many Netboard.

1. <aiq server dns>:9090/ds/dyn/Netboard/aiq01

2. <aiq server dns>:9090/ds/dyn/Netboard/aiq02
3. <aiq server dns>:9090/ds/dyn/Netboard/aiq03
4. <aiq server dns>:9090/ds/dyn/Netboard/aiq04
5. <aiq server dns>:9090/ds/dyn/Netboard/aiq05

Once the webhook is configured with one of the above entry points the data from the incoming http://segment.com event will be available to be verified using normal get Netboard calls.

Last updated: July 28, 2023

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

Run Node.js Packages from Service Suite

Added NPM support was added to Service Suite in the AIQ 4.8.3 release.

When NPM is running within Service Suite, be aware of the following:

- NPM or the program it should execute should be installed where AIQ is running.
  - NPM or the process that the user wants to execute will always be executed in AIQ or in the TestNode where the Service Suite is running, or where Mobile Designer is running. The process is not run in the plug-in.
  - The calling of the method will freeze the test until the method completes. The call will return an object with 3 parameters:
    - Process output
    - Process error
    - Process status
  - NPM/Program call won’t have any request/response/sub steps. It will be another step with only time to run the execution.
  - The call will not fail if the NPM/Program process output is different from 0, the output of the call will be the termination status integer of the method. For standard success output validation and assertEquals(0,output.status) could be used
  - The first parameter points to the folder where the process will be executed.

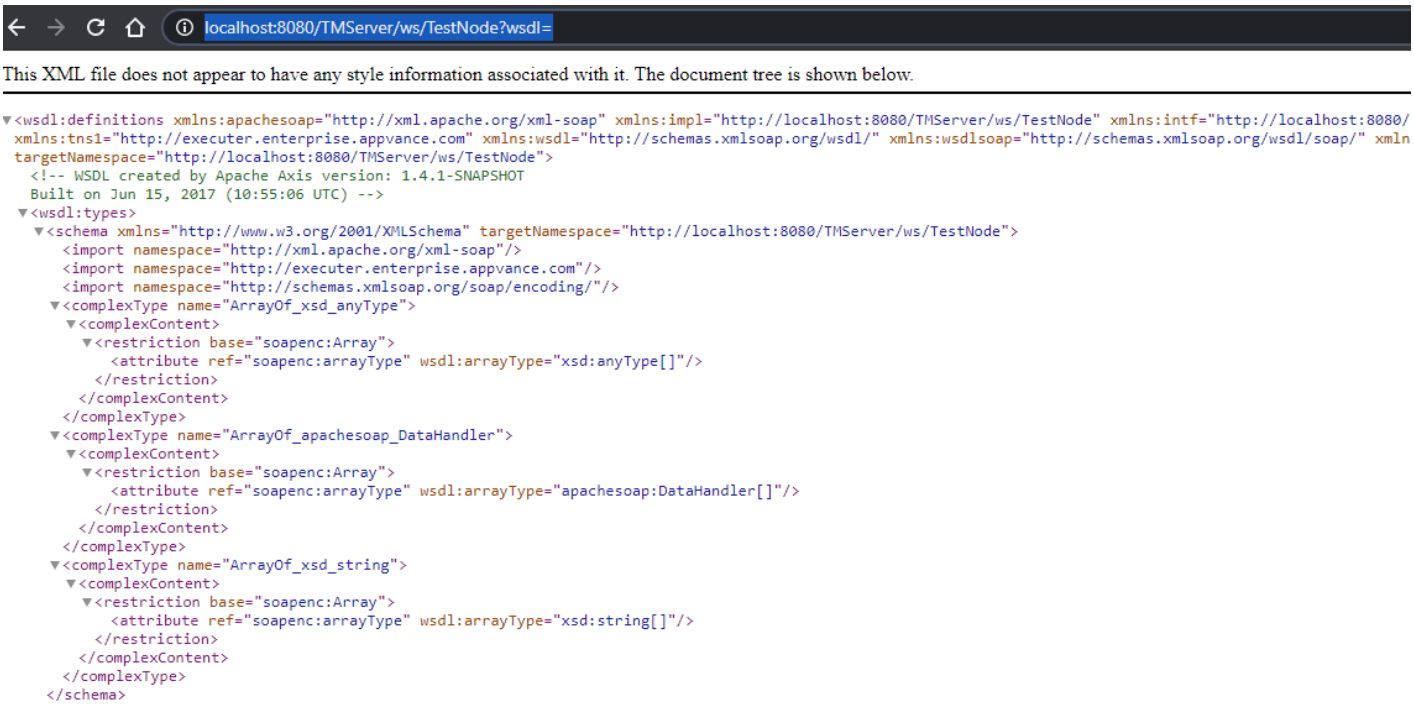
Last updated: June 19, 2023

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

WSDL/SOAP Call-in Service Suite

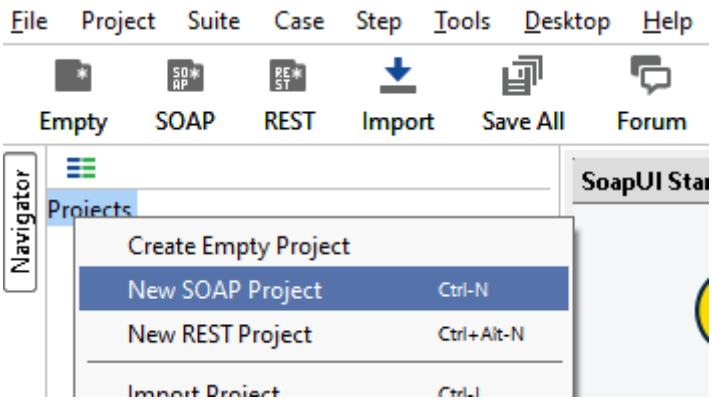
SoapUI support was removed in release 5.2.0 of AIQ.

You first need a working WSDL server, like the Appvance Server back end in http://SERVER:8080/TMServer/ws/TestNode?wsdl=

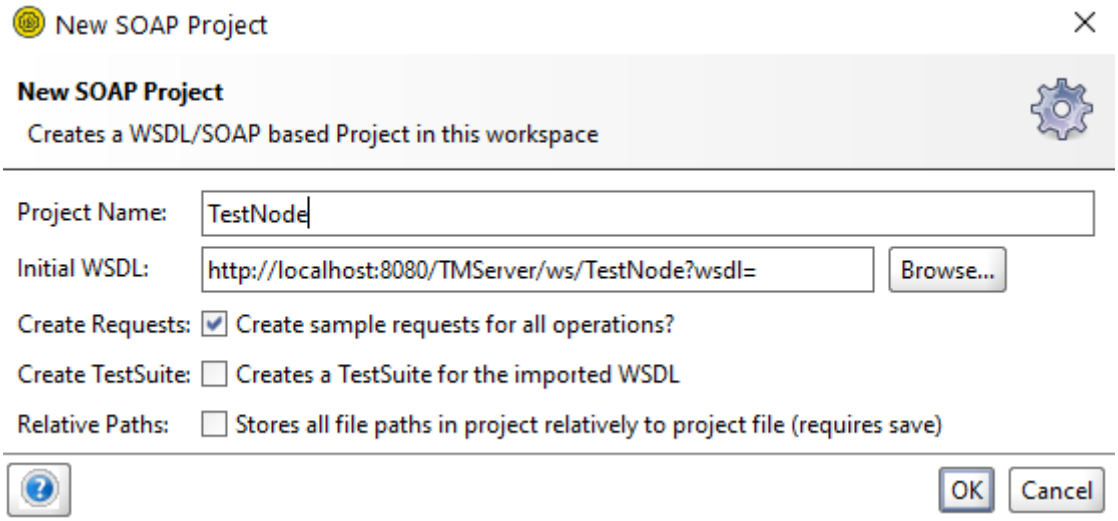


From there you could create manually **a post body data**, or using a third-party tools like soapUI, you could see the post's body, as we only need the body to create the post-call.

(SoapUI can be downloaded from https://www.soapui.org/)



A new Soap Project will show a screen to enter the URL



The above will create a project with an echo method. We can open it and extract the XML section to be used as the POST body data:

<soapenv:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="

http://www.w3.org/2001/XMLSchema " xmlns:soapenv="

http://schemas.xmlsoap.org/soap/envelope/" xmlns:tms="http://tmserver.enterprise.appvance.com
">

<soapenv:Header/>

<soapenv:Body>

<tms:echo soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">

<echo xsi:type="xsd:string">?</echo>

</tms:echo>

</soapenv:Body>

</soapenv:Envelope>

Notice the “?”, this will be our variable. We should replace “?“ in the <echo xsi:type="xsd:string">?</echo> line with a variable name like “ECHO1” so we can later replace that for the intended parameter. For this example, we will replace “?” with “ECHO1” and save the XML as a “requests.echo.xml” file next to our “main.js”.

//Read template

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

//Process template for desired method

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);

The “SOAPAction” HTTP request header field can be used to indicate the intent of the SOAP HTTP request. The value is a URI identifying the intent. SOAP places no restrictions on the format or specificity of the URI or that it is resolvable. An HTTP client MUST use this header field when issuing a SOAP HTTP Request. The header field value of the empty string ("") means that the intent of the SOAP message is provided by the HTTP Request-URI. No value means that there is no indication of the intent of the message. You can read more about it in https://www.w3.org/TR/2000/NOTE-SOAP-20000508/#_Toc478383528



NTLM Service Authentication

When working with services that have NTLM authentication used normally for C# services, there is some additional configuration that needs to be taken into consideration. One of the requirements to consume this type of service will be to provide the credentials configured in the IIS for the available users to log in to the server. The system administrator will provide the domain and workstation where the services reside, similarly, it should provide the username and the password. Once you have the credentials information you will need to proceed to introduce this in the script which has the service call as the following example;



Last updated: July 25, 2024

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