Script Debugger API (SDAPI) 2.0

Contents

Script Debugger API (SDAPI) 2.0

SDAPI is a RESTful API that enables you to remotely access the functionality of the Demandware script debugging engine. It enables you to access *resources* using HTTP requests and HTTP responses. Each *resource* is addressed by its unique URL, which includes the API version. Data is transported using request or header parameters, or within the request body as a JSON *document* with a defined structure.

To get started with SDAPI, please read the following topics:

- HTTP Methods—Describes HTTTP methods used to request resources.
- HTTP status codes and faults—Describes status codes and faults returned in SDAPI responses.
- Client application identification—Describes how to uniquely identify your client application. (A client ID is mandatory for all requests.)
- URL Schema—Describes how to construct requests.

- Breakpoints—Describes how to create, delete, and fetch breakpoints.
- Script Threads—Describes how to interact with script threads.
- Variables and Object Members—Describes how to view the values of variables in the context of a script thread.
- Evaluating expressions—Describes how to evaluate an expression in the context of a script thread.

See Resources and Documents for API reference information.

© Copyright 2000-2023, Salesforce, Inc. All rights reserved. Various trademarks held by their respective owners.

1. SDAPI How to Write a Client 2.0

The Script Debugger API (SDAPI) is a RESTful service that enables you to interact with the script debugger engine. You use the SDAPI to enable the debugger, to set breakpoints, and to interact with halted script threads.

The following sections describe the steps you must perform to create a client application

Authorization and Identification

All SDAPI requests must:

- Use the HTTPS protocol with BASIC authentication, where the credentials represent a Business Manager user with the `WebDAV_Manage_Customization` functional permission.
- Specify a client application identifier as an HTTP header, where the header key is `x-dw-client-id` and the value is a string representing your client.
- Use the Sites-Site context in the request URL (/s-/):

```
https://your_host_name/s/-/dw/debugger/v1_0/
```

See Client Resource section for more information.

Enabling the Debugger

To enable the debugger, issue a POST request to create a Client resource. To disable the debugger, issue a DELETE request to remove the Client resource. Removing the Client resource also removes all breakpoints and resumes all halted script threads.

> **Important:** When finished debugging, always remove the Client resource. Leaving the debugger enabled negatively impacts performance.

Setting Breakpoints

After creating the Client resource, you can create a breakpoint using a POST request. See Breakpoints to understand how to create, delete, and fetch breakpoints.

> **Note:** If you change your script file after creating a breakpoint, update your breakpoint location accordingly.

Interacting with Halted Script Threads

When you set a breakpoint in a script and the breakpoint is hit, the debugger engine halts the script thread so you can interact with it. To interact with the script thread, you can:

- Inspect variables: see Object members to understand how to see the values of variables in the context of a script thread.
- Evaluate expressions: see Evaluating Expressions to understand how to evaluate an expression in the context of a script thread.

Note: You typically fetch script threads by making a GET request in a separate application thread. This allows you to repeatedly query the debugger engine for halted script threads, which you can then use to update your client application user interface.

Important: The debugger engine allows a script thread to be halted for up to one minute. To keep a script thread alive, you must reset the timeout threshold. Keeping the thread alive ensures that you have time to inspect variables and evaluate expressions. See Script Threads to understand how to reset the timeout threshold.You typically reset script threads in a separate application thread. By repeatedly resetting the timeout threshold, you ensure that your script threads are not terminated.

© Copyright 2000-2023, Salesforce, Inc. All rights reserved. Various trademarks held by their respective owners.

2. SDAPI Usage 2.0

The Script Debugger API (SDAPI) consists of *resources* which accept HTTP requests and return HTTP responses. A *resource* is addressed by its unique URL, which includes the API version. Data is transported using request or header parameters, or within the request body as a JSON *document* with a defined structure.

The following topics contain information about using the Script Debugger API:

Topic	Description
Client application identification	You must ensure that your client application identifies itself for security and tracking purposes.
Data Formats	The SDAPI supports the JSON format only.
HTTP Methods	You create requests based on standard HTTP methods (as defined by RFC 2616).
HTTP status codes and faults	You can diagnose problems or failures by examining HTTP status codes and faults.
Pagination	You can use pagination to break up large responses into smaller chunks.

Topic	Description
URL Syntax	You must create appropriately constructed URLs to operate on resources.
Versioning and deprecation policy	You must ensure that your application conforms to the versioning and deprecation policy.

© Copyright 2000-2023, Salesforce, Inc. All rights reserved. Various trademarks held by their respective owners.

2.1. SDAPI Breakpoints 2.0

A breakpoint represents a location in a script file. When the script engine is evaluating a script file, the engine queries the script debugger to determine if the current location represents a breakpoint. If it does represent a breakpoint, the engine halts the script thread at the breakpoint location; otherwise, the engine continues. For each line in the script file, the engine queries the script debugger to determine if the current location represents a breakpoint.

When you create breakpoints, you must specify the full path to the script file in the context of the containing cartridge module.

You can optionally set a 'condition expression' when creating a breakpoint. When evaluating a breakpoint, the engine honors the breakpoint if the 'condition expression' evaluates to 'true'.

The following examples show how you can create breakpoints, get breakpoints, and remove breakpoints.

Example 1: Create Breakpoints

```
REQUEST:
POST /s/-/dw/debugger/v1_0/breakpoints
{
  "breakpoints":
  [
    {
      "line_number":18,
      "script_path":"/app_storefront_controllers/cartridge/controllers/Cart.js"
    },
    {
      "line_number":25,
      "script_path":"/app_storefront_controllers/cartridge/scripts/models/CartModel.js"
    }
  ]
}

RESPONSE:
HTTP/1.1 200 OK
Content-Type: application/json;charset=UTF-8
{
  "_v":"2.0"
  "breakpoints":
  [
```

```
    {
      "id":1,
      "line_number":18,
      "script_path":"/app_storefront_controllers/cartridge/controllers/Cart.js"
    },
    {
      "id":2,
      "line_number":25,
      "script_path":"/app_storefront_controllers/cartridge/scripts/models/CartModel.js"
    }
  ]
}
```

Note: To create a single breakpoint, it must be an element in a JSON collection.

Example 2: Create conditional Breakpoints

```
REQUEST:
POST /s/-/dw/debugger/v1_0/breakpoints
{
  "breakpoints":
  [
    {
      "condition":"product.isOnline() == true",
      "line_number":18,
      "script_path":"/app_storefront_controllers/cartridge/controllers/Cart.js"
    },
    {
      "condition":"product.isOnline() == true",
      "line_number":25,
      "script_path":"/app_storefront_controllers/cartridge/scripts/models/CartModel.js"
    }
  ]
}

RESPONSE:
HTTP/1.1 200 OK
Content-Type: application/json;charset=UTF-8
{
  "_v":"2.0"
  "breakpoints":
  [
    {
      "condition":"product.isOnline() == true",
      "id":1,
      "line_number":18,
      "script_path":"/app_storefront_controllers/cartridge/controllers/Cart.js"
    },
    {
      "condition":"product.isOnline() == true",
```

```
      "id":2,
      "line_number":25,
      "script_path":"/app_storefront_controllers/cartridge/scripts/models/CartModel.js"
    }
  ]
}
```

Note: When using conditional breakpoints, the breakpoint is honored when the condition expression evaluates to true.

Example 3: Get breakpoints (no breakpoint identifer)

```
REQUEST:
GET /s/-/dw/debugger/v1_0/breakpoints

RESPONSE:
HTTP/1.1 200 OK
Content-Type: application/json;charset=UTF-8
{
  "_v":"2.0"
  "breakpoints":
  [
    {
      "id":1,
      "line_number":18,
      "script_path":"/app_storefront_controllers/cartridge/controllers/Cart.js"
    },
    {
      "id":2,
      "line_number":25,
      "script_path":"/app_storefront_controllers/cartridge/scripts/models/CartModel.js"
    }
  ]
}
```

Example 4: Get breakpoints with breakpoint identifer

```
REQUEST:
GET /s/-/dw/debugger/v1_0/breakpoints/1

RESPONSE:
HTTP/1.1 200 OK
Content-Type: application/json;charset=UTF-8
{
  "_v":"2.0"
  "id":1,
  "line_number":18,
  "script_path":"/app_storefront_controllers/cartridge/controllers/Cart.js"
```

```
}
```

Example 5: Remove All Breakpoints

```
REQUEST:
DELETE /s/-/dw/debugger/v1_0/breakpoints/

RESPONSE:
HTTP/1.1 204 No Content
Content-Length: 0
```

Example 6: Remove a Single Breakpoint

```
REQUEST:
DELETE /s/-/dw/debugger/v1_0/breakpoints/1

RESPONSE:
HTTP/1.1 204 No Content
Content-Length: 0
```

© Copyright 2000-2023, Salesforce, Inc. All rights reserved. Various trademarks held by their respective owners.

Show URL Submit Feedback Privacy Policy

2.2. SDAPI Client Application Identification 2.0

The Script Debugger API requires that all client applications identify themselves using a *client ID*.

Include the *client ID* in every API request by setting the `x-dw-client-id` HTTP header:

```
GET https://.../debugger/v1_0/threads
x-dw-client-id:ScriptDebuggerClient
```

Note: The Script Debugger is a single client environment.

© Copyright 2000-2023, Salesforce, Inc. All rights reserved. Various trademarks held by their respective owners.

Show URL Submit Feedback Privacy Policy

2.3. SDAPI Resource Data Formats 2.0

The Script Debugger API currently supports one data format for resources: JSON.

Note: The Script Debugger API supports character encoding in UTF-8 only.

© Copyright 2000-2023, Salesforce, Inc. All rights reserved. Various trademarks held by their respective owners.

Show URL Submit Feedback Privacy Policy

2.4. SDAPI Evaluating Expressions 2.0

The Script Debugger API (SDAPI) enables you to interact with halted script threads. When a script thread is halted, you can evaluate an expression in the context of a script thread and call stack frame.

To evaluate expressions, you use the *eval* action and you specify a thread identifier and call stack frame index. You must also use the *expr* query parameter with the expression to evaluate.

In the vast majority of cases, you do not want to interact with a “running” script thread, because there is no guarantee that the thread will be alive when your request is processed.

Example 1: Evaluating an expression that is a function call

```
REQUEST:
GET /s/-/dw/debugger/v1_0/threads/1/frames/0/eval?expr=test%28%29

RESPONSE:
HTTP/1.1 200 OK
Content-Type: application/json;charset=UTF-8
{
  "_v":"2.0",
  "expression":"test()",
  "result":"this is a test"
}
```

Example 2: Evaluating an expression that returns boolean

```
REQUEST:
GET /s/-/dw/debugger/v1_0/threads/1/frames/0/eval?expr=fName+%3D%3D+%27Larry%27

RESPONSE:
HTTP/1.1 200 OK
Content-Type: application/json;charset=UTF-8
{
  "_v":"2.0",
  "expression":"fName == 'Larry'",
  "result":"true"
}
```

Example 3: Evaluating an expression that returns undefined

```
REQUEST:
GET /s/-/dw/debugger/v1_0/threads/1/frames/0/eval?expr=fName+%3D%3D+%27Larry%27

RESPONSE:
HTTP/1.1 200 OK
Content-Type: application/json;charset=UTF-8
{
  "_v":"2.0",
  "expression":"fName == 'Larry'",
  "result":"\"fName\" is not defined."
}
```

© Copyright 2000-2023, Salesforce, Inc. All rights reserved. Various trademarks held by their respective owners.

Show URL Submit Feedback Privacy Policy

2.5. SDAPI HTTP methods 2.0

A key characteristic of a RESTful Web API is the explicit use of HTTP methods, as defined by RFC 2616. The Script Debugger API supports these methods, as described in the following sections.

GET

The GET method retrieves resources on the server. The GET method is a *safe method,* which means that it should never change the state of the server or have side effects. Consequently, a GET request never initiates transactions on the server.

A typical GET request and its response look like this:

```
REQUEST:
GET /s/-/dw/debugger/v1_0/breakpoints/40 HTTP/1.1
Authorization: Basic YWRtaW46RGVtYW5kd2FyZTEh
x-dw-client-id: DebuggerTest
Content-Type: application/json
Content-Length: 0

RESPONSE:
HTTP/1.1 200 OK
{
  "_v":"2.0",
  "id":40,
  "line_number":20,
  "script_path":"/app_storefront_controllers/cartridge/controllers/Cart.js"
}
```

This sample shows a typical GET request retrieving a list of `Breakpoint` resources. The response has HTTP status code 200, which indicates resources were found and are contained in response body. The response contains the "Content-Type" header, which is set to "application/json" plus the charset definition ("UTF-8").

DELETE

The DELETE method removes one or more resources on the server. DELETE is an *idempotent* method, which means repeating a request should have the same effect as making the request just once. This implies that the server returns HTTP status code 204 (NO CONTENT) even if the server removed the resource previously.

The following example shows how to remove a resource that is addressed by an `Identifier` in the URL:

```
REQUEST:
DELETE /s/-/dw/debugger/v1_0/breakpoints/40 HTTP/1.1
Authorization: Basic YWRtaW46RGVtYW5kd2FyZTEh
x-dw-client-id: DebuggerTest
Content-Type: application/json
Content-Length: 0

RESPONSE:
HTTP/1.1 204 No Content
Content-Length: 0
```

The request is similar to the previous GET request, except the HTTP method changed. The response status code is 204, which means the server successfully fulfilled the request but returned no content.

POST

The POST method is neither *safe* (because requests can affect the server state) nor *idempotent* (because multiple requests potentially return different results).

The Script Debugger API uses POST only for two purposes:

- For creating Client and Breakpoint resources.
- For Action requests - Any action request is done via POST; see Action

```
REQUEST:
POST /s/-/dw/debugger/v1_0/breakpoints HTTP/1.1
Authorization: Basic YWRtaW46RGVtYW5kd2FyZTEh
x-dw-client-id: DebuggerTest
Content-Type: application/json
{
  "_v":"2.0",
  "breakpoints":
    [
      {"line_number":20,"script_path":"/app_storefront_controllers/cartridge/controllers/
      {"line_number":25,"script_path":"/app_storefront_controllers/cartridge/scripts/mode
    ]
}

RESPONSE:
HTTP/1.1 200 OK
Content-Type: application/json;charset=UTF-8
{
  "_v":"2.0",
  "breakpoints":
    [
      {"id":1, "line_number":20,"script_path":"/app_storefront_controllers/cartridge/cont
      {"id":2, "line_number":25,"script_path":"/app_storefront_controllers/cartridge/scri
```

```
    ]
}
```

This sample shows a typical GET request retrieving a list of `Breakpoints` resource. The response has HTTP status code 200, which indicates resources were found and are contained in response body. The response contains the "Content-Type" header, which is set to "application/json" plus the charset definition ("UTF-8").

HEAD

The HEAD method is similar to the GET method but returns headers only, not content (body). The headers are identical to those of the GET request. The HEAD method is a *safe method:* it does not change the state of the server.

```
REQUEST:
HEAD /s/-/dw/debugger/v1_0/breakpoints HTTP/1.1
Authorization: Basic YWRtaW46RGVtYW5kd2FyZTEh
x-dw-client-id: DebuggerTest
Accept: application/json

RESPONSE:
HTTP/1.1 204 NO CONTENT
Content-Length: 67
Content-Type: application/json; charset=UTF-8
```

© Copyright 2000-2023, Salesforce, Inc. All rights reserved. Various trademarks held by their respective owners.

2.6. SDAPI HTTP Status Codes and Faults 2.0

To help you handle errors and special cases, the Script Debugger API returns HTTP status codes and faults.

HTTP Status Codes

The following table lists the HTTP status codes and their typical use cases:

HTTP status code	Cases
200 (OK)	GET or POST request successfully completed.
201 (Created)	POST request successfully created a new resource.
204 (No Content)	DELETE, HEAD, or OPTIONS (or less typically a POST) request successfully completed and returned no content.
400 (Bad Request)	Request contains invalid information, such as malformed parameters, malformed header values, or a malformed body.
401 (Unauthorized)	Request is not authorized to be processed.

HTTP status code	Cases
403 (Forbidden)	Request is declined by the server.
404 (Not Found)	Requested resource does not exist.
405 (Method Not Allowed)	Resource does not support supplied HTTP method.
412 (Precondition Failed)	PATCH request provided an outdated last-known base point, which means the resource was changed on server, possibly by a concurrent request.
415 (Unsupported Media Type)	Media type specified in "format" request parameter (or Accept header) is not supported.
500 (Internal Server Error)	Request cannot be fulfilled because of an unexpected condition on the server.

Fault Document

For status codes greater than or equal to 400, the API returns a fault document:

```
{
  "_v":"2.0",
  "type":"ScriptThreadNotFoundException",
  "message":"Could not find Script Thread for identifier '1'."
}
```

The fault document contains a *type* identifier and a readable *message*.

© Copyright 2000-2023, Salesforce, Inc. All rights reserved. Various trademarks held by their respective owners.

2.7. SDAPI Variables, Objects and Object Members 2.0

The Script Debugger API (SDAPI) enables you to interact with halted script threads. When a script thread is halted, you can fetch the list of objects in the context of a script thread and call stack frame. You can also fetch objects inside an object.

To access objects and object members, you use the `variables` and `members` actions and you specify a thread identifier and call stack frame index. When requesting members, you can optionally include the `object_path` query parameter. If you do not use the `object_path` query parameter, the Script Debugger will return all objects in the context of the specified thread and frame index.

When you specify an `object_path` query parameter, the value of the parameter is a *dot-delimited* string representing a path. For exampole, if the query parameter is `basket` the Script Debugger returns the members of the `basket` object. If the query parameter is `basket.productLineItems` , the Script Debugger returns all members of the *productLineItems* member of the *basket* object.

The response from the *members* action is a collection of JSON objects containing four properties:

- *name*: the name of the object.
- *parent*: the parent of the object.
- *type*: the type of the object.
- *value*: the value of the object. The value is obtained by calling *toString()* on the object.

When you an object represents a collection, the name attribute is an index representation of the member inside the collection. For example, "name":"[0]" represents the first element in the collection.

By default, the Script Debugger will return up to *200* objects for the *members* action and the result set starts with element *200* . If the object path has more than 200 elements, you can use the *start* and *count* query paramters to control the returned objects.

When requesting variables, the response is a collection of JSON objects containing four properties:

- *name*: the name of the object.
- *parent*: the parent of the object.
- *type*: the type of the object.
- *value*: the value of the object. The value is obtained by calling *toString()* on the object.
- *scope*: the scope of the variable, where the value is one of *local* , *closure* or *global* .

In the vast majority of cases, you do not want to interact with a “running” script thread, because there is no guarantee that the thread will be alive when your request is processed.

Examples

Example 1: Get all object members (no query paramter)

```
REQUEST:
GET /s/-/dw/debugger/v1_0/threads/1/frames/0/members

RESPONSE:
HTTP/1.1 200 OK
Content-Type: application/json;charset=UTF-8
{
  "_v":"2.0",
  "object_members":
  [
    {
      "name":"args",
      "parent":"",
      "type":"dw.system.PipelineDictionary",
      "value":"[PipelineDictionary id=385366343]"
    },
    {
      "name":"arguments",
      "parent":"",
```

```
      "type":"org.mozilla.javascript.Arguments",
      "value":"org.mozilla.javascript.Arguments@52312f58"
    },
    {
      "name":"basket",
      "parent":"",
      "type":"dw.order.Basket",
      "value":"[Basket uuid=bcqpfaOacb6QIaaado1BaeQDFh]"
    }
  ]
}
```

Example 2: Get the members of the productLineItems collection in the basket object

```
REQUEST:
GET /s/-/dw/debugger/v1_0/threads/1/frames/0members?object_path=basket.productLineItems

RESPONSE:
HTTP/1.1 200 OK
Content-Type: application/json;charset=UTF-8
{
  "_v":"2.0",
  "object_members":
  [
    {
      "name":"[0]",
      "parent":"basket",
      "type":"dw.order.ProductLineItem",
      "value":"[ProductLineItem uuid=bcDfLaOacbNVMaaadoDBceQDFh]"
    },
    {
      "name":"[1]",
      "parent":"basket",
      "type":"dw.order.ProductLineItem",
      "value":"[ProductLineItem uuid=bcLNbaOacbhiUaaadokBgeQDFh]"
    }
  ]
}
```

Example 3: Get the members of an element inside the productLineItems collection

```
REQUEST:
GET /s/-/dw/debugger/v1_0/threads/1/frames/0/members?object_path=basket.productLineItems.

RESPONSE:
HTTP/1.1 200 OK
Content-Type: application/json;charset=UTF-8
{
  "_v":"2.0",
```

```
  "object_members":
  [
    {
      "name":"adjustedGrossPrice",
      "parent":"productLineItems",
      "type":"dw.value.Money",
      "value":"N/A"
    },
    {
      "name":"adjustedNetPrice",
      "parent":"productLineItems",
      "type":"dw.value.Money",
      "value":"N/A"
    }

    ...

  ]
}
```

Example 4: Get the members of a object using *start* and *count*

```
REQUEST:
GET /s/-/dw/debugger/v1_0/threads/1/frames/0/members?object_path=basket.productLineItems.

RESPONSE:
HTTP/1.1 200 OK
Content-Type: application/json;charset=UTF-8
{
  "_v":"2.0",
  "object_members":
  [
    {
      "name":"optionModel",
      "parent":"productLineItems",
      "type":"String",
      "value":"null"
    },
    {
      "name":"optionProductLineItem",
      "parent":"productLineItems",
      "type":"String",
      "value":"false"
    },
    {
      "name":"optionProductLineItems",
      "parent":"productLineItems",
      "type":"dw.util.Collection",
      "value":"[Collection id=865935255]"
    },
    {
      "name":"optionValueID",
```

```
      "parent":"productLineItems",
      "type":"String",
      "value":"null"
    },
    {
      "name":"orderItem",
      "parent":"productLineItems",
      "type":"String",
      "value":"null"
    },
    {
      "name":"parent",
      "parent":"productLineItems",
      "type":"String",
      "value":"null"
    },
    {
      "name":"position",
      "parent":"productLineItems",
      "type":"String",
      "value":"2"
    },
    {
      "name":"price",
      "parent":"productLineItems",
      "type":"dw.value.Money",
      "value":"N/A"
    },
    {
      "name":"priceAdjustments",
      "parent":"productLineItems",
      "type":"dw.util.Collection",
      "value":"[Collection id=1994121392]"
    },
    {
      "name":"priceValue",
      "parent":"productLineItems",
      "type":"String",
      "value":"null"
    }
  ]
}
```

Example 5: Get the variables *start* and *count*

```
REQUEST:
GET /s/-/dw/debugger/v1_0/threads/1/frames/0/variables

RESPONSE:
HTTP/1.1 200 OK
Content-Type: application/json;charset=UTF-8
{
```

```
    "_v": "2.0",
    "count": 14,
    "object_members": [
        {
            "name": "arguments",
            "parent": "",
            "scope": "local",
            "type": "Object",
            "value": "[object Arguments]"
        },
        {
            "name": "arguments",
            "parent": "",
            "scope": "closure",
            "type": "Object",
            "value": "[object Arguments]"
        },
        {
            "name": "requiredFunction",
            "parent": "",
            "scope": "global",
            "type": "Function",
            "value": "\nfunction requiredFunction() {\n    var dwUtil = require(\"dw/util
        }
    ],
    "start": 0,
    "total": 4
}
```


© Copyright 2000-2023, Salesforce, Inc. All rights reserved. Various trademarks held by their respective owners.

2.8. SDAPI Pagination 2.0

The Script Debugger API uses paging whenever it accesses a collection of *`Object Member`* resource instances. Paging enables the API to return data in chunks, instead of returning a complete collection all at once. You explicitly configure paging by providing values for the `start` and `count` parameters:

- `start`: Specifies the start location of the chunk within the collection of resource elements. Valid values range must be at least 0; the default value is 0. The start location is not a page number; it is the position of the start element within the entire collection.
- `count`: Specifies the number of resource instances to return in each chunk. Valid values range from 1 to 200; the default value is 200.

The `start` and `count` parameters are optional

In the following example, a Object Members request yields a collection of 30 total resource instances. The `start` property specifies that the requested chunk should start at position 15, and the `count` element specifies that the chunk should contain at most 30 resource instances. The response also contains the `total` property that identifies the total number of results that satisfied the request.

```
REQUEST:
GET /s/-/dw/debugger/v1_0/thread/2/frame/0/members?object_path=b.productLineItems.[1]&sta
Host: example.com
Accept: application/json

RESPONSE:
HTTP/1.1 200 OK
Content-Type: application/json; charset=UTF-8
{
  "_v":"2.0",
  "count":30,
  "object_members":[
                        {"name":"externalLineItemText","parent":"productLineItems","type"
                        {"name":"gift","parent":"productLineItems","type":"String","value
                        {"name":"giftMessage","parent":"productLineItems","type":"String"
                        {"name":"grossPrice","parent":"productLineItems","type":"dw.value
                        {"name":"lastModified","parent":"productLineItems","type":"java.u
                        {"name":"lineItemCtnr","parent":"productLineItems","type":"dw.orc
                        {"name":"lineItemText","parent":"productLineItems","type":"String
                        {"name":"manufacturerName","parent":"productLineItems","type":"St
                        {"name":"manufacturerSKU","parent":"productLineItems","type":"Str
                        {"name":"minOrderQuantity","parent":"productLineItems","type":"dw
                        {"name":"minOrderQuantityValue","parent":"productLineItems","type
                        {"name":"netPrice","parent":"productLineItems","type":"dw.value.N
                        {"name":"optionID","parent":"productLineItems","type":"String","v
                        {"name":"optionModel","parent":"productLineItems","type":"String"
                        {"name":"optionProductLineItem","parent":"productLineItems","type
                        {"name":"optionProductLineItems","parent":"productLineItems","typ
                        {"name":"optionValueID","parent":"productLineItems","type":"Strin
                        {"name":"orderItem","parent":"productLineItems","type":"String","
                        {"name":"parent","parent":"productLineItems","type":"String","val
                        {"name":"position","parent":"productLineItems","type":"String","v
                        {"name":"price","parent":"productLineItems","type":"dw.value.Mone
                        {"name":"priceAdjustments","parent":"productLineItems","type":"dw
                        {"name":"priceValue","parent":"productLineItems","type":"String",
                        {"name":"product","parent":"productLineItems","type":"dw.catalog.
                        {"name":"productID","parent":"productLineItems","type":"String","
                        {"name":"productInventoryList","parent":"productLineItems","type"
                        {"name":"productInventoryListID","parent":"productLineItems","typ
                        {"name":"productListItem","parent":"productLineItems","type":"Str
                        {"name":"productName","parent":"productLineItems","type":"String"
                        {"name":"proratedPrice","parent":"productLineItems","type":"dw.va
  "start":15,
  "total":59
}
```

© Copyright 2000-2023, Salesforce, Inc. All rights reserved. Various trademarks held by their respective owners.

Show URL Submit Feedback Privacy Policy

2.9. SDAPI Script Threads 2.0

A script thread is a container for a running script. When a Script Pipelet or Controller Script is called, a script thread is created representing the script file. A script thread is either 'running' or 'halted'. If the script thread is halted, a stack frame is created representing the entry point of the script. For every function that is subsequently called, a new stack frame is created and added to the script's call stack. When function is exited, the stack frame is removed from the call stack.

The script thread's call stack is a zero-based index where element `0` represents the current location of the script execution path.

To interact with script threads, you must first create a breakpoint and then you must call a Controller or a Pipeline that runs the script. When the Script Engine encounters a breakpoint, the script thread is halted. When a script thread is halted, you can use actions to:

- Fetch variables based on script thread and stack frame index. See Object Members for more information.
- Evaluate expressions based on script thread and stack frame index. See Object Members for more information.
- Control the execution of the script by stepping into a function, stepping out of a function, stepping over the current line, resuming scripts, and stopping script threads. See Object Members for more information.

Important: By default, a script thread can be halted for up to 60 seconds without reseting the timeout counter. Each time you reset the timeout counter, you have another 60 seconds to interact with the thread. If you do not reset the timeout counter, the script engine throws a ScriptDebuggerTerminate error in the script that was halted. See Example 3 for the syntax for resetting timeout counters.

In the vast majority of cases, you do not want to interact with a “running” script thread, because there is no guarantee that the thread will be alive when your request is processed.

Examples

Example 1: Get all script threads

```
REQUEST:
GET /s/-/dw/debugger/v1_0/threads HTTP/1.1

RESPONSE:
HTTP/1.1 200 OK
{
  "_v":"2.0",
  "script_threads":[
  {
    "call_stack":[
    {
      "index":0,
      "location":
      {
        "function_name":"first()",
        "line_number":25,
        "script_path":"/app_storefront_controllers/cartridge/scripts/models/CartModel.js'
      }
    },
    {
      "index":1,
```

```
      "location":
      {
        "function_name":"execute()",
        "line_number":12,
        "script_path":"/app_storefront_controllers/cartridge/controllers/Cart.js"
      }
    }],
  "id":2,
  "status":"halted"
  }
 ]
}
```

Example 2: Get a specific script thread

```
REQUEST:
GET /s/-/dw/debugger/v1_0/threads/2 HTTP/1.1

RESPONSE:
HTTP/1.1 200 OK
{
  "_v":"2.0",
    "call_stack":[
    {
      "index":0,
      "location":
      {
        "function_name":"first()",
        "line_number":25,
        "script_path":"/app_storefront_controllers/cartridge/scripts/models/CartModel.js'
      }
    },
    {
      "index":1,
      "location":
      {
        "function_name":"execute()",
        "line_number":12,
        "script_path":"/app_storefront_controllers/cartridge/controllers/Cart.js"
      }
    }],
  "id":2,
  "status":"halted"
}
```

Example 3: Reset script thread timeout counter for all halted threads

```
REQUEST:
POST /s/-/dw/debugger/v1_0/threads/reset HTTP/1.1
```

```
RESPONSE:
HTTP/1.1 204 No Content
```

© Copyright 2000-2023, Salesforce, Inc. All rights reserved. Various trademarks held by their respective owners.

2.10. SDAPI URL Syntax 2.0

The Script Debugger API uses a specialized schema for its URLs, and each URL consists of a base URL and an extended URL. The base URL is the same for all API requests; the extended URL changes depending on the resource and the operation. To access the Script Debugger API's functionality, you construct URLs as described below.

Base URL

The base URL has a different structure depending on whether you are using a production system or a development system (sandboxes). The base URL has the following structure:

```
https://sub_domain.demandware.net/s/-/dw/debugger/
```

where *sub_domain* is any valid subdomain of the demandware.net domain (for example, staging.store.adidas.demandware.net).

Extended URL

The base URL provides the main access point of the Script Debugger API. You extend the base URL to access specific resources. When you extend the base URL, you must conform to the following patterns. The variables shown in these patterns are described below.

The first pattern addresses multiple resources of a resource type:

```
base_url/version_id/resource_type
```

The second addresses a single resource using an identifier:

```
base_url/version_id/resource_type/identifier
```

The third addresses resource information by specifying an action:

```
base_url/version_id/resource_type/action
```

The third addresses information from a dependent resource by specifying an action:

```
base_url/version_id/resource_type/identifier/relationship_type/action
```

Note: Use only ASCII characters in your URLs; escape any reserved ASCII characters by using the common "%" notation.

Version ID

The *version_id* specifies the API version. Each API version supports a set of resource types (*resource_type*), resource properties and actions (*action*). A new API version can add new resource types, add properties to existing resource types, deprecate previously supported resource types, and introduce new semantics or behaviors without introducing structural changes.

The *version_id* starts with the character "v" (lowercase) followed by the actual version number, separated by an underscore. For example:

```
http://example.com/dw/debugger/v1_0/resource_type/identifier
```

Resource type

Resource types (*resource_type*) are fundamental to the Script Debugger API, as they are to any RESTful API. Each resource type has a corresponding set of data: properties and actions (*action*). An instance of a resource type is analogous to a debugger object, such as `Breakpoint` or `Script Thread`.

The Script Debugger API provides a fixed set of system resource types for each API version. Resource types that provide access to multiple resources are given plural names (for example, "breakpoints"). Resource types that provide access to only one resource are given singular names (for example, "breakpoint"). The following example URL retrieves resources of type `breakpoints`:

```
http://example.com/dw/debugger/v1_0/breakpoints
```

Identifier

Unique identifiers (*identifier*) enable you to request specific resource instances. The following URL returns the thread using an ID:

```
http://example.com/dw/debugger/v1_0/thread/1
```

The *identifier* must be URL encoded.

Action

Actions (*action*) operate on specific resource types (*resource_type*) and relationship types (*relationship_type*). Actions are not generally available across all resource types and relationship types, and they are only available when using the HTTP method POST. Actions perform special operations—for example, resume script.

The following example URL uses an action to direct the script to step into a function (this action is specific to the threads resource type):

```
http://example.com/dw/debugger/-/v1_0/threads/1/frame/0/into
```

© Copyright 2000-2023, Salesforce, Inc. All rights reserved. Various trademarks held by their respective owners.

2.11. SDAPI versioning and deprecation policy 2.0

This document describes the versioning and deprecation policy for the Demandware Script Debugger API (SDAPI). This policy is designed to

- Be easy for customers and developers to understand
- Let Demandware make design changes to the API without affecting existing client applications
- Let Demandware delete outdated or unsupported features

Version creation and deprecation

A new SDAPI version is created whenever Demandware:

- Adds a new document property, resource parameter, or resource.
- Modifies the type of a document property, resource parameter, or resource.
- Modifies the name of a document property, resource parameter, or resource.
- Deletes a document property, resource parameter, or resource.

Whenever Demandware creates a new OCAPI version, it also deprecates the former SDAPI version. This encourages customers and developers to:

- Implement new client applications on top of the current SDAPI version.
- Upgrade existing applications on a regular basis.

Version lifecycle

Version State	Support	Description
Current	Yes	The newest version. Always develop new applications using the current version. Demandware ensures that the SDAPI version is supported for at least two years. During this period Demandware ensures the stability and backwards-compatibility of the version enclosed feature set.
Deprecated	Yes	Demandware deprecates an SDAPI version when a new version is published. Support, stability, and backwards-compatibility are ensured for up to two years after the publish date.
Obsolete	No	Deprecated versions become obsolete automatically two years after being published. Demandware deletes obsolete versions without further notice.

Notification process

Demandware notifies customers about new and deprecated versions on the XChange portal and in the Demandware Release Notes.

© Copyright 2000-2023, Salesforce, Inc. All rights reserved. Various trademarks held by their respective owners.

Show URL Submit Feedback Privacy Policy

3. SDAPI Resources 2.0

Resource Overview

The SDAPI resources enable you to remotely access the functionality of the Demandware Script Debugger.

Resource	Description
Breakpoints	Enables you to create, get, and delete breakpoints.
Client	Creates a debugging client and enables the script debugger.
Threads	Enables you to access debugger functionality, such as stepping through lines of code.

© Copyright 2000-2023, Salesforce, Inc. All rights reserved. Various trademarks held by their respective owners.

Show URL Submit Feedback Privacy Policy

3.1. Breakpoints Resource (Debugger API 2.0)

Summary

Http Method	Resource	Description
GET	/breakpoints	Returns all breakpoints currently set in the debugger.
POST	/breakpoints	Sets all breakpoints in the specified Breakpoints instance.
DELETE	/breakpoints	Removes all the breakpoints from the debugger.
GET	/breakpoints/{breakpoint_id}	Gets a breakpoint. If the breakpoint cannot be located, this action throws a BreakpointNotFoundException.
DELETE	/breakpoints/{breakpoint_id}	Deletes the breakpoint. If the breakpoint cannot be located, this action throws a BreakpointNotFoundException.

Get Breakpoints

Returns all breakpoints currently set in the debugger.

Url

```
GET http://hostname:port/dw/debugger/v2_0/breakpoints
```

Formats

json, xml

Authentication

Name	Description
None	No authentication.

Response Document

Breakpoints

In case of a failure Fault Document is returned.

Faults

Status	Type	Arguments	Description
400	`ClientIdRequiredException`		Indicates that the request did not contain a client identifier.
401	`NotAuthorizedException`		Indicates that the specified user is not authorized.
412	`DebuggerDisabledException`		Indicates that the Client resource has not been created.

Sample

```
REQUEST:
GET /s/-/dw/debugger/v1_0/breakpoints HTTP/1.1
Authorization: Basic YWRtaW46RGVtYW5kd2FyZTEh
x-dw-client-id: DebuggerTest
Content-Type: application/json
Content-Length: 0

in case of success:

RESPONSE:
HTTP/1.1 200 OK
Content-Type: application/json;charset=UTF-8
```

```
{
  "_v":"2.0",
  "breakpoints":
  [
    {
      "id":1,
      "line_number":25,
      "script_path":"/app_storefront_controllers/cartridge/controllers/Cart.js"
    }
  ]
}

in case of authorization error:

RESPONSE:
HTTP/1.1 401 Unauthorized
Content-Type: application/json;charset=UTF-8
{
  "_v":"2.0",
  "fault":
    {
      "type":"NotAuthorizedException",
      "message":"You are not authorized to use the Script Debugger."
    }
}

# in case of identification failure:

RESPONSE
HTTP/1.1 400 Bad Request
Content-Type: application/json;charset=UTF-8
{
  "_v":"2.0",
  "fault":
  {
    "type":"ClientIdRequiredException",
    "message":"You must have a client identifier in the request header using the key 'x-c
  }
}
```

Create Breakpoints

Sets all breakpoints in the specified Breakpoints instance.

Url

```
POST http://hostname:port/dw/debugger/v2_0/breakpoints
```

Formats

json, xml

Authentication

Name	Description
None	No authentication.

Request Document

Breakpoints

Response Document

Breakpoints

In case of a failure Fault Document is returned.

Faults

Status	Type	Arguments	Description
400	`ClientIdRequiredException`		Indicates that the request did not contain a client identifier.
400	`InvalidScriptPathException`		Indicates that the script path is not valid.
400	`InvalidScriptFileException`		Indicates that the script file is not valid.
401	`NotAuthorizedException`		Indicates that the specified user is not authorized.
412	`DebuggerDisabledException`		Indicates that the Client resource has not been created.

Sample

```
REQUEST:
POST /s/-/dw/debugger/v1_0/breakpoints HTTP/1.1
Authorization: Basic YWRtaW46RGVtYW5kd2FyZTEh
x-dw-client-id: DebuggerTest
Content-Type: application/json
{
  "_v":"2.0",
  "breakpoints":
    [
      {"line_number":20,"script_path":"/app_storefront_controllers/cartridge/controllers/
      {"line_number":25,"script_path":"/app_storefront_controllers/cartridge/scripts/mode
    ]
}

REQUEST:
```

```
POST /s/-/dw/debugger/v1_0/breakpoints HTTP/1.1
Authorization: Basic YWRtaW46RGVtYW5kd2FyZTEh
x-dw-client-id: DebuggerTest
Content-Type: application/json
{
  "_v":"2.0",
  "breakpoints":
    [
      {"condition":"product.isOnline() == true", "line_number":20,"script_path":"/app_sto
      {"condition":"product.isOnline() == true", "line_number":25,"script_path":"/app_sto
    ]
}


# in case of success:

RESPONSE:
HTTP/1.1 200 OK
Content-Type: application/json;charset=UTF-8
{
  "_v":"2.0",
  "breakpoints":
    [
      {"id":1, "line_number":20,"script_path":"/app_storefront_controllers/cartridge/cont
      {"id":2, "line_number":25,"script_path":"/app_storefront_controllers/cartridge/scri
    ]
}

# in case of success with condition:

RESPONSE:
HTTP/1.1 200 OK
Content-Type: application/json;charset=UTF-8
{
  "_v":"2.0",
  "breakpoints":
    [
      {"condition":"product.isOnline() == true", "id":1, "line_number":20,"script_path":'
      {"condition":"product.isOnline() == true", "id":2, "line_number":25,"script_path":'
    ]
}

# in case of failures:

RESPONSE:
HTTP/1.1 400 BAD REQUEST
Expires: Thu, 01-Jan-1970 00:00:00 GMT
Content-Type: application/json;charset=UTF-8
Cache-Control: max-age=0,no-cache,no-store,must-revalidate
{
  "_v":"2.0",
  "type":"InvalidScriptFileException",
  "message":"Script Path must be a JavaScript file with a suffix of '.js' or '.ds'."
}
```

```
RESPONSE:
HTTP/1.1 400 BAD REQUEST
Expires: Thu, 01-Jan-1970 00:00:00 GMT
Content-Type: application/json;charset=UTF-8
Cache-Control: max-age=0,no-cache,no-store,must-revalidate
{
  "_v":"2.0",
  "type":"InvalidScriptPathException",
  "message":"Script Path must start with a forward slash '/'."
}
```

Delete Breakpoints

Removes all the breakpoints from the debugger.

Url

```
DELETE http://hostname:port/dw/debugger/v2_0/breakpoints
```

Formats

json, xml

Authentication

Name	Description
None	No authentication.

In case of a failure Fault Document is returned.

Faults

Status	Type	Arguments	Description
400	`ClientIdRequiredException`		Indicates that the request did not contain a client identifier.
401	`NotAuthorizedException`		Indicates that the specified user is not authorized.
412	`DebuggerDisabledException`		Indicates that the Client resource has not been created.

Sample

```
REQUEST:
DELETE /s/-/dw/debugger/v1_0/breakpoints HTTP/1.1
```

```
Authorization: Basic YWRtaW46RGVtYW5kd2FyZTEh
x-dw-client-id: DebuggerTest
Content-Type: application/json
Content-Length: 0

# in case of success:

RESPONSE:
HTTP/1.1 204 No Content

# in case of failure:

RESPONSE:
HTTP/1.1 401 Unauthorized
Content-Type: application/json;charset=UTF-8
{
  "_v":"2.0",
  "fault":
    {
      "type":"NotAuthorizedException",
      "message":"You are not authorized to use the Script Debugger."
    }
}

RESPONSE:
HTTP/1.1 400 Bad Request
Content-Type: application/json;charset=UTF-8
{
  "_v":"2.0",
  "fault":
  {
    "type":"ClientIdRequiredException",
    "message":"You must have a client identifier in the request header using the key 'x-c
  }
}
```

Get Breakpoint

Gets a breakpoint. If the breakpoint cannot be located, this action throws a BreakpointNotFoundException.

Url

```
GET http://hostname:port/dw/debugger/v2_0/breakpoints/{breakpoint_id}
```

Formats

json, xml

Authentication

Name	Description
None	No authentication.

Response Document

Breakpoint

Path Parameters

Parameter	Type	Description	Constraints
breakpoint_id	Integer	The breakpoint identifier.	minLength=1

In case of a failure Fault Document is returned.

Faults

Status	Type	Arguments	Description
400	`ClientIdRequiredException`		Indicates that the request did not contain a client identifier.
401	`NotAuthorizedException`		Indicates that the specified user is not authorized.
404	`BreakpointNotFoundException`	id (Integer)	Indicates that the requested breakpoint could not be found.
412	`DebuggerDisabledException`		Indicates that the Client resource has not been created.

Sample

```
REQUEST:
GET /s/-/dw/debugger/v1_0/breakpoints/40 HTTP/1.1
Authorization: Basic YWRtaW46RGVtYW5kd2FyZTEh
x-dw-client-id: DebuggerTest
Content-Type: application/json
Content-Length: 0

# in case of success:

RESPONSE:
HTTP/1.1 200 OK
{
```

```
  "_v":"2.0",
  "id":40,
  "line_number":20,
  "script_path":"/app_storefront_controllers/cartridge/controllers/Cart.js"
}

# in case of error:

RESPONSE:
HTTP/1.1 404 Requested resource not found
Content-Type: application/json;charset=UTF-8
{
  "_v":"2.0",
  "type":"BreakpointNotFoundException",
  "message":"Could not find Breakpoint for identifier '40'."
}

in case of error:

RESPONSE:
HTTP/1.1 401 Unauthorized
Content-Type: application/json;charset=UTF-8
{
  "_v":"2.0",
  "fault":
    {
      "type":"NotAuthorizedException",
      "message":"You are not authorized to use the Script Debugger."
    }
}

RESPONSE:
HTTP/1.1 400 Bad Request
Content-Type: application/json;charset=UTF-8
{
  "_v":"2.0",
  "fault":
  {
    "type":"ClientIdRequiredException",
    "message":"You must have a client identifier in the request header using the key 'x-c
  }
}
```

Delete Breakpoint

Deletes the breakpoint. If the breakpoint cannot be located, this action throws a BreakpointNotFoundException.

Url

```
DELETE http://hostname:port/dw/debugger/v2_0/breakpoints/{breakpoint_id}
```

Formats

json, xml

Authentication

Name	Description
None	No authentication.

Path Parameters

Parameter	Type	Description	Constraints
breakpoint_id	Integer	The breakpoint identifier.	mandatory=true, minLength=1

In case of a failure Fault Document is returned.

Faults

Status	Type	Arguments	Description
400	`ClientIdRequiredException`		Indicates that the request did not contain a client identifier.
401	`NotAuthorizedException`		Indicates that the specified user is not authorized.
404	`BreakpointNotFoundException`	id (Integer)	Indicates that the Client resource has not been created.
412	`DebuggerDisabledException`		Indicates that the Client resource has not been created.

Sample

```
REQUEST:
DELETE /s/-/dw/debugger/v1_0/breakpoints/40 HTTP/1.1
Authorization: Basic YWRtaW46RGVtYW5kd2FyZTEh
x-dw-client-id: DebuggerTest
Content-Type: application/json
Content-Length: 0

# in case of success:

RESPONSE:
HTTP/1.1 204 No Content
```

```
# in case of failure:

RESPONSE:
HTTP/1.1 404 Requested resource not found
Content-Type: application/json;charset=UTF-8
{
  "_v":"2.0",
  "type":"BreakpointNotFoundException",
  "message":"Could not find Breakpoint for identifier '40'."
}

RESPONSE:
HTTP/1.1 401 Unauthorized
Content-Type: application/json;charset=UTF-8
{
  "_v":"2.0",
  "fault":
    {
      "type":"NotAuthorizedException",
      "message":"You are not authorized to use the Script Debugger."
    }
}

RESPONSE:
HTTP/1.1 400 Bad Request
Content-Type: application/json;charset=UTF-8
{
  "_v":"2.0",
  "fault":
  {
    "type":"ClientIdRequiredException",
    "message":"You must have a client identifier in the request header using the key 'x-c
  }
}
```

© Copyright 2000-2023, Salesforce, Inc. All rights reserved. Various trademarks held by their respective owners.

Show URL Submit Feedback Privacy Policy

3.2. Client Resource (Debugger API 2.0)

Summary

Http Method	Resource	Description
POST	/client	Creates the Client and enables the debugger. You must create the Client before you can interact with other debugger resources.
DELETE	/client	Removes all breakpoints, resumes all halted script threads and disables the debugger by deleting the Client.

Client Create

Creates the Client and enables the debugger. You must create the Client before you can interact with other debugger resources.

Url

```
POST http://hostname:port/dw/debugger/v2_0/client
```

Formats

json, xml

Authentication

Name	Description
None	No authentication.

In case of a failure Fault Document is returned.

Faults

Status	Type	Arguments	Description
400	`ClientIdRequiredException`		Indicates that the request did not contain a client identifier.
401	`NotAuthorizedException`		Indicates that the specified user is not authorized.

Sample

```
REQUEST:
POST /s/-/dw/debugger/v1_0/client HTTP/1.1
Authorization: Basic YWRtaW46RGVtYW5kd2FyZTEh
x-dw-client-id: DebuggerTest
Content-Type: application/json
Content-Length: 0

# in case of success:

RESPONSE:
HTTP/1.1 204 No Content
Content-Length: 0

# in case of failure:

RESPONSE:
HTTP/1.1 401 Unauthorized
```

```
Content-Type: application/json;charset=UTF-8
{
  "_v":"2.0",
  "fault":
    {
      "type":"NotAuthorizedException",
      "message":"You are not authorized to use the Script Debugger."
    }
}

RESPONSE:
HTTP/1.1 400 Bad Request
Content-Type: application/json;charset=UTF-8
{
  "_v":"2.0",
  "fault":
  {
    "type":"ClientIdRequiredException",
    "message":"You must have a client identifier in the request header using the key 'x-c
  }
}
```

Client Delete

Removes all breakpoints, resumes all halted script threads and disables the debugger by deleting the Client.

Url

```
DELETE http://hostname:port/dw/debugger/v2_0/client
```

Formats

json, xml

Authentication

Name	Description
None	No authentication.

In case of a failure Fault Document is returned.

Faults

Status	Type	Arguments	Description
400	`ClientIdRequiredException`		Indicates that the request did not contain a client identifier.

Status	Type	Arguments	Description
401	`NotAuthorizedException`		Indicates that the specified user is not authorized.

Sample

```
REQUEST:
DELETE /s/-/dw/debugger/v1_0/client HTTP/1.1
Authorization: Basic YWRtaW46RGVtYW5kd2FyZTEh
x-dw-client-id: DebuggerTest
Content-Type: application/json
Content-Length: 0

# in case of success:

RESPONSE:
HTTP/1.1 204 No Content

# in case of error:

RESPONSE:
HTTP/1.1 401 Unauthorized
Content-Type: application/json;charset=UTF-8
{
  "_v":"2.0",
  "fault":
    {
      "type":"NotAuthorizedException",
      "message":"You are not authorized to use the Script Debugger."
    }
}

RESPONSE:
HTTP/1.1 400 Bad Request
Content-Type: application/json;charset=UTF-8
{
  "_v":"2.0",
  "fault":
  {
    "type":"ClientIdRequiredException",
    "message":"You must have a client identifier in the request header using the key 'x-c
  }
}
```

© Copyright 2000-2023, Salesforce, Inc. All rights reserved. Various trademarks held by their respective owners.

3.3. Threads Resource (Debugger API 2.0)

Summary

Http Method	Resource	Description
GET	/threads	Returns the script threads in the script engine. A script thread is either 'running' or 'halted'. If the script thread is halted, the script thread contains a call stack of stack frames representing the execution path. The stack frame at index [0] represents the current location of the execution path. Both the script thread identifier and a stack frame index are required for evaluating expressions and viewing object state.
POST	/threads/reset	Directs the debugger to reset the timeout counter for all halted script threads. Each script thread can sleep up to 60 seconds before the debugger will terminate the thread. This action resets the timeout counter allowing a thread to halt for another 60 seconds. Note that if a script thread times out, the script engine throws a ScriptDebuggerTerminate error in the script that was halted.
GET	/threads/{thread_id}	Returns the script thread specified by the thread identifier. A script thread is either 'running' or 'halted'. If the script thread is halted, the script thread contains a call stack of stack frames representing the execution path. The stack frame at index [0] represents the current location of the execution path. Both the script thread identifier and a stack frame index are required for evaluating expressions and viewing object state. If the script thread cannot be located, this action throws a ScriptThreadNotFoundException.
GET	/threads/{thread_id}/frames/{frame_index}/eval	Evaluates an expression in the context of the specified thread and frame. Each script thread contains a call stack of stack frames representing the execution path. The stack frame at index [0] represents the current location of the execution path. You can also use other frame indices for expression evaluation outside the current location. If the script thread cannot be located, this action throws a ScriptThreadNotFoundException. If the

Http Method	Resource	Description
		frame index is not valid, this action throws an InvalidFrameIndexException.
GET	/threads/{thread_id}/frames/{frame_index}/members	Returns the members of the object path in the context of the specified thread and frame. If the object path is not specified, returns all members of the specified thread and frame. Each script thread contains a call stack of stack frames representing the execution path. The stack frame at index [0] represents the current location of the execution path. You can also use other frame indices for viewing object state outside the current location. If the script thread cannot be located, this action throws a ScriptThreadNotFoundException. If the frame index is not valid, this action throws an InvalidFrameIndexException.
GET	/threads/{thread_id}/frames/{frame_index}/variables	Returns the variables in the context of the specified thread and frame scope and all inclosing scopes. Each script thread contains a call stack of stack frames representing the execution path. The stack frame at index [0] represents the current location of the execution path. You can also use other frame indices for viewing object state outside the current location. If the script thread cannot be located, this action throws a ScriptThreadNotFoundException. If the frame index is not valid, this action throws an InvalidFrameIndexException.
POST	/threads/{thread_id}/into	Directs the script thread to step into the function at the current thread location. If the current location is not a function, this action steps to the next line in the script. If there are no other lines in the script, the script completes. If the script thread cannot be located, this action throws a ScriptThreadNotFoundException.
POST	/threads/{thread_id}/out	Directs the script thread to step out of the current thread location and to return to the parent in the call stack. If there is no parent in the call stack, this action directs the script thread to resume. If the script

Http Method	Resource	Description
		thread cannot be located, this action throws a ScriptThreadNotFoundException.
POST	/threads/{thread_id}/over	Directs the script thread to step over the current thread location to the next line in the script. If the current location is at the end of a script function, this action directs the script thread to resume. If the script thread cannot be located, this action throws a ScriptThreadNotFoundException.
POST	/threads/{thread_id}/resume	Directs the script thread to resume the execution of the script. Depending on the script location and breakpoints, calling resume can result in the thread stopping at another breakpoint. If the script thread cannot be located, this action throws a ScriptThreadNotFoundException.
POST	/threads/{thread_id}/stop	Directs the script thread to stop execution. This action directs the script engine to throw a ScriptDebuggerTerminate error in the script that was halted. If the script thread cannot be located, this action throws a ScriptThreadNotFoundException.

Get Script Threads Document.

Returns the script threads in the script engine. A script thread is either 'running' or 'halted'. If the script thread is halted, the script thread contains a call stack of stack frames representing the execution path. The stack frame at index [0] represents the current location of the execution path. Both the script thread identifier and a stack frame index are required for evaluating expressions and viewing object state.

Url

```
GET http://hostname:port/dw/debugger/v2_0/threads
```

Formats

json, xml

Authentication

Name	Description
None	No authentication.

Response Document

ScriptThreads

In case of a failure Fault Document is returned.

Faults

Status	Type	Arguments	Description
400	`ClientIdRequiredException`		Indicates that the request did not contain a client identifier.
401	`NotAuthorizedException`		Indicates that the specified user is not authorized.
412	`DebuggerDisabledException`		Indicates that the debugger Client resource has not been created.

Sample

```
REQUEST:
GET /s/-/dw/debugger/v1_0/threads HTTP/1.1
Authorization: Basic YWRtaW46RGVtYW5kd2FyZTEh
x-dw-client-id: DebuggerTest
Content-Type: application/json
Content-Length: 0

# in case of success:

RESPONSE:
HTTP/1.1 200 OK
{
  "_v":"2.0",
  "script_threads":[
  {
    "call_stack":[
    {
      "index":0,"location":
      {
        "function_name":"first()",
        "line_number":25,
        "script_path":"/app_storefront_controllers/cartridge/scripts/models/CartModel.js'
      }
    },
    {
```

```
      "index":1,
      "location":
      {
        "function_name":"execute()",
        "line_number":12,
        "script_path":"/app_storefront_controllers/cartridge/controllers/Cart.js"
      }
    }
  ],
  "id":2,"status":"halted"
  }
 ]
}

in case of error:

RESPONSE:
HTTP/1.1 401 Unauthorized
Content-Type: application/json;charset=UTF-8
{
  "_v":"2.0",
  "fault":
    {
      "type":"NotAuthorizedException",
      "message":"You are not authorized to use the Script Debugger."
    }
}

RESPONSE:
HTTP/1.1 400 Bad Request
Content-Type: application/json;charset=UTF-8
{
  "_v":"2.0",
  "fault":
  {
    "type":"ClientIdRequiredException",
    "message":"You must have a client identifier in the request header using the key 'x-c
  }
}
```

Reset

Directs the debugger to reset the timeout counter for all halted script threads. Each script thread can sleep up to 60 seconds before the debugger will terminate the thread. This action resets the timeout counter allowing a thread to halt for another 60 seconds. Note that if a script thread times out, the script engine throws a ScriptDebuggerTerminate error in the script that was halted.

Url

```
POST http://hostname:port/dw/debugger/v2_0/threads/reset
```

Formats

json, xml

Authentication

Name	Description
None	No authentication.

In case of a failure Fault Document is returned.

Faults

Status	Type	Arguments	Description
400	`ClientIdRequiredException`		Indicates that the request did not contain a client identifier.
401	`NotAuthorizedException`		Indicates that the specified user is not authorized.
412	`DebuggerDisabledException`		Indicates that the debugger Client resource has not been created.

Sample

```
REQUEST:
POST /s/-/dw/debugger/v1_0/threads/reset HTTP/1.1
Authorization: Basic YWRtaW46RGVtYW5kd2FyZTEh
x-dw-client-id: DebuggerTest
Content-Type: application/json
Content-Length: 0

# in case of success:

RESPONSE:
HTTP/1.1 204 OK

# in case of failure:

RESPONSE
HTTP/1.1 401 Unauthorized
Content-Type: application/json;charset=UTF-8
{
  "_v":"2.0",
  "fault":
    {
      "type":"NotAuthorizedException",
      "message":"You are not authorized to use the Script Debugger."
    }
}
```

```
RESPONSE
HTTP/1.1 400 Bad Request
Content-Type: application/json;charset=UTF-8
{
  "_v":"2.0",
  "fault":
  {
    "type":"ClientIdRequiredException",
    "message":"You must have a client identifier in the request header using the key 'x-c
  }
}
```

Get Script Thread

Returns the script thread specified by the thread identifier. A script thread is either 'running' or 'halted'. If the script thread is halted, the script thread contains a call stack of stack frames representing the execution path. The stack frame at index [0] represents the current location of the execution path. Both the script thread identifier and a stack frame index are required for evaluating expressions and viewing object state. If the script thread cannot be located, this action throws a ScriptThreadNotFoundException.

Url

```
GET http://hostname:port/dw/debugger/v2_0/threads/{thread_id}
```

Formats

json, xml

Authentication

Name	Description
None	No authentication.

Response Document

ScriptThread

Path Parameters

Parameter	Type	Description	Constraints
thread_id	Integer	the thread identifier.	minLength=1

In case of a failure Fault Document is returned.

Faults

Status	Type	Arguments	Description
400	`ClientIdRequiredException`		Indicates that the request did not contain a client identifier.
401	`NotAuthorizedException`		Indicates that the specified user is not authorized.
404	`ScriptThreadNotFoundException`	threadId (Integer)	Indicates that the requested thread could not be found.
412	`DebuggerDisabledException`		Indicates that the debugger Client resource has not been created.

Sample

```
REQUEST:
GET /s/-/dw/debugger/v1_0/threads/2 HTTP/1.1
Authorization: Basic YWRtaW46RGVtYW5kd2FyZTEh
x-dw-client-id: DebuggerTest
Content-Type: application/json
Content-Length: 0

# in case of success:

RESPONSE:
HTTP/1.1 200 OK
{
  "_v":"2.0",
    "call_stack":[
    {
      "index":0,"location":
      {
        "function_name":"first()",
        "line_number":25,
        "script_path":"/app_storefront_controllers/cartridge/scripts/models/CartModel.js'
      }
    },
    {
      "index":1,
      "location":
      {
        "function_name":"execute()",
        "line_number":12,
        "script_path":"/app_storefront_controllers/cartridge/controllers/Cart.js"
      }
    }
  ],
  "id":2,
```

```
  "status":"halted"
}

# in case of failure:

RESPONSE:
HTTP/1.1 404 Requested resource not found
Content-Type: application/json;charset=UTF-8
{
  "_v":"2.0",
  "type":"ScriptThreadNotFoundException",
  "message":"Could not find Script Thread for identifier '1'."
}

RESPONSE:
HTTP/1.1 401 Unauthorized
Content-Type: application/json;charset=UTF-8
{
  "_v":"2.0",
  "fault":
    {
      "type":"NotAuthorizedException",
      "message":"You are not authorized to use the Script Debugger."
    }
}

RESPONSE:
HTTP/1.1 400 Bad Request
Content-Type: application/json;charset=UTF-8
{
  "_v":"2.0",
  "fault":
  {
    "type":"ClientIdRequiredException",
    "message":"You must have a client identifier in the request header using the key 'x-c
  }
}
```

Evaluate Expression

Evaluates an expression in the context of the specified thread and frame. Each script thread contains a call stack of stack frames representing the execution path. The stack frame at index [0] represents the current location of the execution path. You can also use other frame indices for expression evaluation outside the current location. If the script thread cannot be located, this action throws a ScriptThreadNotFoundException. If the frame index is not valid, this action throws an InvalidFrameIndexException.

Url

```
GET http://hostname:port/dw/debugger/v2_0/threads/{thread_id}/frames/{frame_index}/eval?e
```

Formats

json, xml

Authentication

Name	Description
None	No authentication.

Response Document

EvalResult

Path Parameters

Parameter	Type	Description	Constraints
frame_index	Integer	the frame index in the thread call stack.	minLength=1
thread_id	Integer	the thread identifier.	minLength=1

Query Parameters

Parameter	Type	Description	Constraints
expr	String	the expression to evaluate.	

In case of a failure Fault Document is returned.

Faults

Status	Type	Arguments	Description
400	`ClientIdRequiredException`		Indicates that the request did not contain a client identifier.
400	`InvalidFrameIndexException`	frameIndex (Integer) threadId (Integer)	Indicates that the frame index is not valid for the script thread.
401	`NotAuthorizedException`		Indicates that the specified user is not authorized.

Status	Type	Arguments	Description
404	`ScriptThreadNotFoundException`	threadId (Integer)	Indicates that the script thread can not be found.
412	`DebuggerDisabledException`		Indicates that the Client resource has not been created.

Sample

```
REQUEST:
GET /s/-/dw/debugger/v1_0/threads/1/frames/0/eval?expr=calculate()
Authorization: Basic YWRtaW46RGVtYW5kd2FyZTEh
x-dw-client-id: DebuggerTest
Content-Type: application/json
Content-Length: 0

RESPONSE:
HTTP/1.1 200 OK
Content-Type: application/json;charset=UTF-8
{
  "_v":"2.0",
  "expression":"calculate()",
  "result":100
}

# in case of failure:

RESPONSE:
HTTP/1.1 404 Requested resource not found
Content-Type: application/json;charset=UTF-8
{
  "_v":"2.0",
  "type":"ScriptThreadNotFoundException",
  "message":"Could not find Script Thread for identifier '1'."
}

RESPONSE:
HTTP/1.1 400 Bad Request
Content-Type: application/json;charset=UTF-8
{
  "_v":"2.0",
  "fault":
  {
    "type":"InvalidFrameIndexException",
    "message":"Frame index '2' does not exist for Script Thread '2'."
  }
}

RESPONSE
```

```
HTTP/1.1 401 Unauthorized
Content-Type: application/json;charset=UTF-8
{
  "_v":"2.0",
  "fault":
    {
      "type":"NotAuthorizedException",
      "message":"You are not authorized to use the Script Debugger."
    }
}

RESPONSE
HTTP/1.1 400 Bad Request
Content-Type: application/json;charset=UTF-8
{
  "_v":"2.0",
  "fault":
  {
    "type":"ClientIdRequiredException",
    "message":"You must have a client identifier in the request header using the key 'x-c
  }
}
```

Get Object Members

Returns the members of the object path in the context of the specified thread and frame. If the object path is not specified, returns all members of the specified thread and frame. Each script thread contains a call stack of stack frames representing the execution path. The stack frame at index [0] represents the current location of the execution path. You can also use other frame indices for viewing object state outside the current location. If the script thread cannot be located, this action throws a ScriptThreadNotFoundException. If the frame index is not valid, this action throws an InvalidFrameIndexException.

Url

```
GET http://hostname:port/dw/debugger/v2_0/threads/{thread_id}/frames/{frame_index}/member
```

Formats

json, xml

Authentication

Name	Description
None	No authentication.

Response Document

ObjectMembers

Path Parameters

Parameter	Type	Description	Constraints
frame_index	Integer	the frame index in the thread's call stack.	minLength=1
thread_id	Integer	the thread identifier.	minLength=1

Query Parameters

Parameter	Type	Description	Constraints
count	Integer	An integer indicating the number of items to return. If not set, the default is 200.	maxIntegerValue=2147483647, minIntegerValue=1
object_path	String	An optional dot-delimited path. The path represents an object in the thread's call stack, where each segment in the path represents a descendant of the object. If this parameter is not specified, all objects in the thread's frame are returned.	
start	Integer	An integer indicating the start location in the result set to return. If not set, the default is 0.	minIntegerValue=0

In case of a failure Fault Document is returned.

Faults

Status	Type	Arguments	Description
400	`ClientIdRequiredException`		Indicates that the request did not contain a client identifier.
400	`InvalidFrameIndexException`	frameIndex (Integer) threadId (Integer)	Indicates that the frame index is not valid for the script thread.
401	`NotAuthorizedException`		Indicates that the specified user is not authorized.

Status	Type	Arguments	Description
404	`ScriptThreadNotFoundException`	threadId (Integer)	Indicates that the script thread can not be found.
412	`DebuggerDisabledException`		Indicates that the Client resource has not been created.

Sample

```
REQUEST:
GET /s/-/dw/debugger/v1_0/threads/1/frames/0/members?object_path=basket.productLineItems
Authorization: Basic YWRtaW46RGVtYW5kd2FyZTEh
x-dw-client-id: DebuggerTest
Content-Type: application/json
Content-Length: 0

RESPONSE:
HTTP/1.1 200 OK
Content-Type: application/json;charset=UTF-8
{
  "_v":"2.0",
  "object_members":
  [
    {
      "name":"[0]",
      "parent":"basket",
      "type":"dw.order.ProductLineItem",
      "value":"[ProductLineItem uuid=bcDfLaOacbNVMaaadoDBceQDFh]"
    },
    {
      "name":"[1]",
      "parent":"basket",
      "type":"dw.order.ProductLineItem",
      "value":"[ProductLineItem uuid=bcLNbaOacbhiUaaadokBgeQDFh]"
    }
  ]
}

# in case of failure:

RESPONSE:
HTTP/1.1 404 Requested resource not found
Content-Type: application/json;charset=UTF-8
{
  "_v":"2.0",
  "type":"ScriptThreadNotFoundException",
  "message":"Could not find Script Thread for identifier '1'."
}
```

```
RESPONSE:
HTTP/1.1 400 Bad Request
Content-Type: application/json;charset=UTF-8
{
  "_v":"2.0",
  "fault":
  {
    "type":"InvalidFrameIndexException",
    "message":"Frame index '2' does not exist for Script Thread '2'."
  }
}

RESPONSE
HTTP/1.1 401 Unauthorized
Content-Type: application/json;charset=UTF-8
{
  "_v":"2.0",
  "fault":
    {
      "type":"NotAuthorizedException",
      "message":"You are not authorized to use the Script Debugger."
    }
}

RESPONSE
HTTP/1.1 400 Bad Request
Content-Type: application/json;charset=UTF-8
{
  "_v":"2.0",
  "fault":
  {
    "type":"ClientIdRequiredException",
    "message":"You must have a client identifier in the request header using the key 'x-c
  }
}
```

Get Variables

Returns the variables in the context of the specified thread and frame scope and all inclosing scopes. Each script thread contains a call stack of stack frames representing the execution path. The stack frame at index [0] represents the current location of the execution path. You can also use other frame indices for viewing object state outside the current location. If the script thread cannot be located, this action throws a ScriptThreadNotFoundException. If the frame index is not valid, this action throws an InvalidFrameIndexException.

Url

```
GET http://hostname:port/dw/debugger/v2_0/threads/{thread_id}/frames/{frame_index}/variab
```

Formats

json, xml

Authentication

Name	Description
None	No authentication.

Response Document

ObjectMembers

Path Parameters

Parameter	Type	Description	Constraints
frame_index	Integer	the frame index in the thread's call stack.	minLength=1
thread_id	Integer	the thread identifier.	minLength=1

In case of a failure Fault Document is returned.

Faults

Status	Type	Arguments	Description
400	`ClientIdRequiredException`		Indicates that the request did not contain a client identifier.
400	`InvalidFrameIndexException`	frameIndex (Integer) threadId (Integer)	Indicates that the frame index is not valid for the script thread.
401	`NotAuthorizedException`		Indicates that the specified user is not authorized.
404	`ScriptThreadNotFoundException`	threadId (Integer)	Indicates that the script thread can not be found.
412	`DebuggerDisabledException`		Indicates that the Client resource has not been created.

Sample

```
REQUEST:
GET /s/-/dw/debugger/v1_0/threads/1/frames/0/variables
Authorization: Basic YWRtaW46RGVtYW5kd2FyZTEh
x-dw-client-id: DebuggerTest
Content-Type: application/json
Content-Length: 0

RESPONSE:
HTTP/1.1 200 OK
{
    "_v": "2.0",
    "count": 14,
    "object_members": [
        {
            "name": "arguments",
            "parent": "",
            "scope": "local",
            "type": "Object",
            "value": "[object Arguments]"
        },
        {
            "name": "arguments",
            "parent": "",
            "scope": "closure",
            "type": "Object",
            "value": "[object Arguments]"
        },
        {
            "name": "requiredFunction",
            "parent": "",
            "scope": "global",
            "type": "Function",
            "value": "\nfunction requiredFunction() {\n    var dwUtil = require(\"dw/util
        }
    ],
    "start": 0,
    "total": 4
}

# in case of failure:

RESPONSE:
HTTP/1.1 404 Requested resource not found
Content-Type: application/json;charset=UTF-8
{
  "_v":"2.0",
  "type":"ScriptThreadNotFoundException",
  "message":"Could not find Script Thread for identifier '1'."
}

RESPONSE:
HTTP/1.1 400 Bad Request
Content-Type: application/json;charset=UTF-8
{
```

```
  "_v":"2.0",
  "fault":
  {
    "type":"InvalidFrameIndexException",
    "message":"Frame index '2' does not exist for Script Thread '2'."
  }
}

RESPONSE
HTTP/1.1 401 Unauthorized
Content-Type: application/json;charset=UTF-8
{
  "_v":"2.0",
  "fault":
    {
      "type":"NotAuthorizedException",
      "message":"You are not authorized to use the Script Debugger."
    }
}

RESPONSE
HTTP/1.1 400 Bad Request
Content-Type: application/json;charset=UTF-8
{
  "_v":"2.0",
  "fault":
  {
    "type":"ClientIdRequiredException",
    "message":"You must have a client identifier in the request header using the key 'x-c
  }
}
```

Step Into

Directs the script thread to step into the function at the current thread location. If the current location is not a function, this action steps to the next line in the script. If there are no other lines in the script, the script completes. If the script thread cannot be located, this action throws a ScriptThreadNotFoundException.

Url

```
POST http://hostname:port/dw/debugger/v2_0/threads/{thread_id}/into
```

Formats

json, xml

Authentication

Name	Description
None	No authentication.

Response Document

ScriptThread

Path Parameters

Parameter	Type	Description	Constraints
thread_id	Integer	the thread identifier.	minLength=1

In case of a failure Fault Document is returned.

Faults

Status	Type	Arguments	Description
400	`ClientIdRequiredException`		Indicates that the request did not contain a client identifier.
401	`NotAuthorizedException`		Indicates that the specified user is not authorized.
404	`ScriptThreadNotFoundException`	threadId (Integer)	Indicates that the requested thread could not be found.
412	`DebuggerDisabledException`		Indicates that the debugger Client resource has not been created.

Sample

```
REQUEST:
POST /s/-/dw/debugger/v1_0/threads/2/into HTTP/1.1
Authorization: Basic YWRtaW46RGVtYW5kd2FyZTEh
x-dw-client-id: DebuggerTest
Content-Type: application/json
Content-Length: 0

# in case of success:

RESPONSE:
HTTP/1.1 200 OK
{
    "_v": "2.0",
    "call_stack": [
        {
            "index": 0,
            "location": {
```

```
                "function_name": "start()",
                "line_number": 50,
                "script_path": "/app_storefront_controllers/cartridge/controllers/Cart.js
            }
        }
    ],
    "id": 4,
    "status": "halted"
}


# in case of failure:

RESPONSE:
HTTP/1.1 404 Requested resource not found
Content-Type: application/json;charset=UTF-8
{
  "_v":"2.0",
  "type":"ScriptThreadNotFoundException",
  "message":"Could not find Script Thread for identifier '1'."
}

RESPONSE:
HTTP/1.1 400 Bad Request
Content-Type: application/json;charset=UTF-8
{
  "_v":"2.0",
  "fault":
  {
    "type":"InvalidFrameIndexException",
    "message":"Frame index '2' does not exist for Script Thread '2'."
  }
}

RESPONSE
HTTP/1.1 401 Unauthorized
Content-Type: application/json;charset=UTF-8
{
  "_v":"2.0",
  "fault":
    {
      "type":"NotAuthorizedException",
      "message":"You are not authorized to use the Script Debugger."
    }
}

RESPONSE
HTTP/1.1 400 Bad Request
Content-Type: application/json;charset=UTF-8
{
  "_v":"2.0",
  "fault":
  {
    "type":"ClientIdRequiredException",
```

```
    "message":"You must have a client identifier in the request header using the key 'x-c
  }
}
```

Step Out

Directs the script thread to step out of the current thread location and to return to the parent in the call stack. If there is no parent in the call stack, this action directs the script thread to resume. If the script thread cannot be located, this action throws a ScriptThreadNotFoundException.

Url

```
POST http://hostname:port/dw/debugger/v2_0/threads/{thread_id}/out
```

Formats

json, xml

Authentication

Name	Description
None	No authentication.

Response Document

ScriptThread

Path Parameters

Parameter	Type	Description	Constraints
thread_id	Integer	the thread identifier.	minLength=1

In case of a failure Fault Document is returned.

Faults

Status	Type	Arguments	Description
400	`ClientIdRequiredException`		Indicates that the request did not contain a client identifier.
401	`NotAuthorizedException`		Indicates that the specified user is not authorized.

Status	Type	Arguments	Description
404	`ScriptThreadNotFoundException`	threadId (Integer)	Indicates that the requested thread could not be found.
412	`DebuggerDisabledException`		Indicates that the debugger Client resource has not been created.

Sample

```
REQUEST:
POST /s/-/dw/debugger/v1_0/threads/2/out HTTP/1.1
Authorization: Basic YWRtaW46RGVtYW5kd2FyZTEh
x-dw-client-id: DebuggerTest
Content-Type: application/json
Content-Length: 0

# in case of success:

RESPONSE:
HTTP/1.1 200 OK
{
    "_v": "2.0",
    "call_stack": [
        {
            "index": 0,
            "location": {
                "function_name": "start()",
                "line_number": 50,
                "script_path": "/app_storefront_controllers/cartridge/controllers/Cart.js
            }
        }
    ],
    "id": 4,
    "status": "halted"
}

# in case of failure:

RESPONSE:
HTTP/1.1 404 Requested resource not found
Content-Type: application/json;charset=UTF-8
{
  "_v":"2.0",
  "type":"ScriptThreadNotFoundException",
  "message":"Could not find Script Thread for identifier '1'."
}

RESPONSE:
HTTP/1.1 400 Bad Request
```

```
Content-Type: application/json;charset=UTF-8
{
  "_v":"2.0",
  "fault":
  {
    "type":"InvalidFrameIndexException",
    "message":"Frame index '2' does not exist for Script Thread '2'."
  }
}

RESPONSE
HTTP/1.1 401 Unauthorized
Content-Type: application/json;charset=UTF-8
{
  "_v":"2.0",
  "fault":
    {
      "type":"NotAuthorizedException",
      "message":"You are not authorized to use the Script Debugger."
    }
}

RESPONSE
HTTP/1.1 400 Bad Request
Content-Type: application/json;charset=UTF-8
{
  "_v":"2.0",
  "fault":
  {
    "type":"ClientIdRequiredException",
    "message":"You must have a client identifier in the request header using the key 'x-c
  }
}
```

Step Over

Directs the script thread to step over the current thread location to the next line in the script. If the current location is at the end of a script function, this action directs the script thread to resume. If the script thread cannot be located, this action throws a ScriptThreadNotFoundException.

Url

```
POST http://hostname:port/dw/debugger/v2_0/threads/{thread_id}/over
```

Formats

json, xml

Authentication

Name	Description
None	No authentication.

Response Document

ScriptThread

Path Parameters

Parameter	Type	Description	Constraints
thread_id	Integer	the thread identifier.	minLength=1

In case of a failure Fault Document is returned.

Faults

Status	Type	Arguments	Description
400	`ClientIdRequiredException`		Indicates that the request did not contain a client identifier.
401	`NotAuthorizedException`		Indicates that the specified user is not authorized.
404	`ScriptThreadNotFoundException`	threadId (Integer)	Indicates that the requested thread could not be found.
412	`DebuggerDisabledException`		Indicates that the debugger Client resource has not been created.

Sample

```
REQUEST:
POST /s/-/dw/debugger/v1_0/threads/2/over HTTP/1.1
Authorization: Basic YWRtaW46RGVtYW5kd2FyZTEh
x-dw-client-id: DebuggerTest
Content-Type: application/json
Content-Length: 0

# in case of success:

RESPONSE:
HTTP/1.1 200 OK
```

```
{
    "_v": "2.0",
    "call_stack": [
        {
            "index": 0,
            "location": {
                "function_name": "start()",
                "line_number": 50,
                "script_path": "/app_storefront_controllers/cartridge/controllers/Cart.js
            }
        }
    ],
    "id": 4,
    "status": "halted"
}

# in case of failure:

RESPONSE:
HTTP/1.1 404 Requested resource not found
Content-Type: application/json;charset=UTF-8
{
  "_v":"2.0",
  "type":"ScriptThreadNotFoundException",
  "message":"Could not find Script Thread for identifier '1'."
}

RESPONSE:
HTTP/1.1 400 Bad Request
Content-Type: application/json;charset=UTF-8
{
  "_v":"2.0",
  "fault":
  {
    "type":"InvalidFrameIndexException",
    "message":"Frame index '2' does not exist for Script Thread '2'."
  }
}

RESPONSE
HTTP/1.1 401 Unauthorized
Content-Type: application/json;charset=UTF-8
{
  "_v":"2.0",
  "fault":
    {
      "type":"NotAuthorizedException",
      "message":"You are not authorized to use the Script Debugger."
    }
}

RESPONSE
HTTP/1.1 400 Bad Request
Content-Type: application/json;charset=UTF-8
```

```
{
  "_v":"2.0",
  "fault":
  {
    "type":"ClientIdRequiredException",
    "message":"You must have a client identifier in the request header using the key 'x-c
  }
}
```

Resume

Directs the script thread to resume the execution of the script. Depending on the script location and breakpoints, calling resume can result in the thread stopping at another breakpoint. If the script thread cannot be located, this action throws a ScriptThreadNotFoundException.

Url

```
POST http://hostname:port/dw/debugger/v2_0/threads/{thread_id}/resume
```

Formats

json, xml

Authentication

Name	Description
None	No authentication.

Response Document

ScriptThread

Path Parameters

Parameter	Type	Description	Constraints
thread_id	Integer	the thread identifier.	minLength=1

In case of a failure Fault Document is returned.

Faults

Status	Type	Arguments	Description
400	`ClientIdRequiredException`		Indicates that the request did not contain a client identifier.

Status	Type	Arguments	Description
401	`NotAuthorizedException`		Indicates that the specified user is not authorized.
404	`ScriptThreadNotFoundException`	threadId (Integer)	Indicates that the requested thread could not be found.
412	`DebuggerDisabledException`		Indicates that the debugger Client resource has not been created.

Sample

```
REQUEST:
POST /s/-/dw/debugger/v1_0/threads/2/resume HTTP/1.1
Authorization: Basic YWRtaW46RGVtYW5kd2FyZTEh
x-dw-client-id: DebuggerTest
Content-Type: application/json
Content-Length: 0

# in case of success:

RESPONSE:
HTTP/1.1 200 OK
Content-Type: application/json;charset=UTF-8
{
    "_v": "2.0",
    "id": 2,
    "status": "running"
}

# in case of failure:

RESPONSE:
HTTP/1.1 404 Requested resource not found
Content-Type: application/json;charset=UTF-8
{
  "_v":"2.0",
  "type":"ScriptThreadNotFoundException",
  "message":"Could not find Script Thread for identifier '1'."
}

RESPONSE:
HTTP/1.1 400 Bad Request
Content-Type: application/json;charset=UTF-8
{
  "_v":"2.0",
  "fault":
  {
```

```
    "type":"InvalidFrameIndexException",
    "message":"Frame index '2' does not exist for Script Thread '2'."
  }
}

RESPONSE
HTTP/1.1 401 Unauthorized
Content-Type: application/json;charset=UTF-8
{
  "_v":"2.0",
  "fault":
    {
      "type":"NotAuthorizedException",
      "message":"You are not authorized to use the Script Debugger."
    }
}

RESPONSE
HTTP/1.1 400 Bad Request
Content-Type: application/json;charset=UTF-8
{
  "_v":"2.0",
  "fault":
  {
    "type":"ClientIdRequiredException",
    "message":"You must have a client identifier in the request header using the key 'x-c
  }
}
```

Stop

Directs the script thread to stop execution. This action directs the script engine to throw a ScriptDebuggerTerminate error in the script that was halted. If the script thread cannot be located, this action throws a ScriptThreadNotFoundException.

Url

```
POST http://hostname:port/dw/debugger/v2_0/threads/{thread_id}/stop
```

Formats

json, xml

Authentication

Name	Description
None	No authentication.

Path Parameters

Parameter	Type	Description	Constraints
thread_id	Integer	the thread identifier.	minLength=1

In case of a failure Fault Document is returned.

Faults

Status	Type	Arguments	Description
400	`ClientIdRequiredException`		Indicates that the request did not contain a client identifier.
401	`NotAuthorizedException`		Indicates that the specified user is not authorized.
404	`ScriptThreadNotFoundException`	threadId (Integer)	Indicates that the requested thread could not be found.
412	`DebuggerDisabledException`		Indicates that the debugger Client resource has not been created.

Sample

```
REQUEST:
POST /s/-/dw/debugger/v1_0/threads/2/stop HTTP/1.1
Authorization: Basic YWRtaW46RGVtYW5kd2FyZTEh
x-dw-client-id: DebuggerTest
Content-Type: application/json
Content-Length: 0

# in case of success:

RESPONSE:
HTTP/1.1 204 OK

# in case of failure:

RESPONSE:
HTTP/1.1 404 Requested resource not found
Content-Type: application/json;charset=UTF-8
{
  "_v":"2.0",
  "type":"ScriptThreadNotFoundException",
  "message":"Could not find Script Thread for identifier '1'."
}
```

```
RESPONSE:
HTTP/1.1 400 Bad Request
Content-Type: application/json;charset=UTF-8
{
  "_v":"2.0",
  "fault":
  {
    "type":"InvalidFrameIndexException",
    "message":"Frame index '2' does not exist for Script Thread '2'."
  }
}

RESPONSE
HTTP/1.1 401 Unauthorized
Content-Type: application/json;charset=UTF-8
{
  "_v":"2.0",
  "fault":
    {
      "type":"NotAuthorizedException",
      "message":"You are not authorized to use the Script Debugger."
    }
}

RESPONSE
HTTP/1.1 400 Bad Request
Content-Type: application/json;charset=UTF-8
{
  "_v":"2.0",
  "fault":
  {
    "type":"ClientIdRequiredException",
    "message":"You must have a client identifier in the request header using the key 'x-c
  }
}
```

© Copyright 2000-2023, Salesforce, Inc. All rights reserved. Various trademarks held by their respective owners.

Show URL Submit Feedback Privacy Policy

4. SDAPI Documents 2.0

To use the Script Debugger API effectively, you have to work with *documents.* The following documents are supported by the Script Debugger API:

- Breakpoint document
- Breakpoints document
- EvalResult document
- Fault Document

- Location document
- ObjectMember document
- ObjectMembers document
- ScriptThread document
- ScriptThreads document
- StackFrame document

© Copyright 2000-2023, Salesforce, Inc. All rights reserved. Various trademarks held by their respective owners.

Show URL Submit Feedback Privacy Policy

4.1. Breakpoint Document (Debugger API 2.0)

Represents a breakpoint.

Property	Type	Constraints	Description
`condition`	String		Return the condition expression that must evaluate to true for the breakpoint to be honored.
`id`	Integer		The breakpoint identifier. This property is created by the debugger.
`line_number`	Integer	mandatory=true, minIntegerValue=1, minLength=1, nullable=false	The line number in the script. If the line number is not an executable line of code, the debugger will not stop at the breakpoint.
`script_path`	String	mandatory=true, nullable=false	The absolute path to the script. The path starts with the '/' delimiter and ends with the name of the script file. The script file must have either a '.js' or '.ds' extension.

© Copyright 2000-2023, Salesforce, Inc. All rights reserved. Various trademarks held by their respective owners.

Show URL Submit Feedback Privacy Policy

4.2. Breakpoints Document (Debugger API 2.0)

Represents a list of breakpoints.

Property	Type	Constraints	Description
`breakpoints`	[Breakpoint]	mandatory=true, nullable=false	A list of breakpoint documents.

© Copyright 2000-2023, Salesforce, Inc. All rights reserved. Various trademarks held by their respective owners.

Show URL Submit Feedback Privacy Policy

4.3. EvalResult Document (Debugger API 2.0)

Represents the result of evaluating a script expression.

Property	Type	Constraints	Description
`expression`	String		The script expression that was evaluated.
`result`	String		The result of the evaluation of the script expression.

© Copyright 2000-2023, Salesforce, Inc. All rights reserved. Various trademarks held by their respective owners.

Show URL Submit Feedback Privacy Policy

4.4. Fault Document (Debugger API 2.0)

Document representing a fault message that is returned whenever the HTTP status code is greater than or equal to 400 (which indicates an error).

Property	Type	Constraints	Description
`message`	String		The message text of the exception the fault represents.
`type`	String		The name of the exception that the fault represents.

© Copyright 2000-2023, Salesforce, Inc. All rights reserved. Various trademarks held by their respective owners.

Show URL Submit Feedback Privacy Policy

4.5. Location Document (Debugger API 2.0)

Represents the location in a script.

Property	Type	Constraints	Description
`function_name`	String		The name of the function in the halted script.
`line_number`	Integer	mandatory=true, nullable=false	The current line number inside the halted script.
`script_path`	String	mandatory=true,	The absolute path of the halted script.

Property	Type	Constraints	Description
		nullable=false	

© Copyright 2000-2023, Salesforce, Inc. All rights reserved. Various trademarks held by their respective owners.

Show URL Submit Feedback Privacy Policy

4.6. ObjectMember Document (Debugger API 2.0)

Represents the member of a JavaScript object where the key is the name, the value is a toString() representation of the object, and the type is the true type of the object member.

Property	Type	Constraints	Description
`name`	String		The name of the object member.
`parent`	String		The name of the parent of this member (if any).
`scope`	String		The scope of the object member, which is one of 'local', 'closure' or 'global'.
`type`	String		The type of the object member, such as 'string', 'boolean', 'function', or Salesforce B2C Commerce API types.
`value`	String		The value of the object member by calling toString() on the instance.

© Copyright 2000-2023, Salesforce, Inc. All rights reserved. Various trademarks held by their respective owners.

Show URL Submit Feedback Privacy Policy

4.7. ObjectMembers Document (Debugger API 2.0)

Represents a list of ObjectMember instances.

Property	Type	Constraints	Description
`count`	Integer		Returns the count of the result set that is being returned in this list.
`object_members`	[ObjectMember]		A list of ObjectMember documents.
`start`	Integer		Returns the start position of the result set that is being returned in this list.

Property	Type	Constraints	Description
`total`	Integer		Returns the total number of objects that satisfied the request.

© Copyright 2000-2023, Salesforce, Inc. All rights reserved. Various trademarks held by their respective owners.

4.8. ScriptThread Document (Debugger API 2.0)

Represents a thread in the script engine.

Property	Type	Constraints	Description
`call_stack`	[StackFrame]		The script stack frames. The frames indicate the execution path to the halted thread where the most recent frame has an index of zero.
`id`	Integer	mandatory=true, nullable=false	The script thread identifier.
`status`	Enum {halted, running, not_found}		The script status.

© Copyright 2000-2023, Salesforce, Inc. All rights reserved. Various trademarks held by their respective owners.

4.9. ScriptThreads Document (Debugger API 2.0)

Represents the script threads that are in the engine.

Property	Type	Constraints	Description
`script_threads`	[ScriptThread]		The script thread documents that are in the script engine.

© Copyright 2000-2023, Salesforce, Inc. All rights reserved. Various trademarks held by their respective owners.

4.10. StackFrame Document (Debugger API 2.0)

Represents a single stack frame.

Property	Type	Constraints	Description
`index`	Integer		The index of this frame in a script thread frame stack.
`location`	Location		The script location of this frame.

© Copyright 2000-2023, Salesforce, Inc. All rights reserved. Various trademarks held by their respective owners.

Show URL Submit Feedback Privacy Policy