

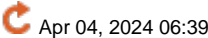
# headlessclient

## Starting Headless Clients from Debug Clients in Servoy Developer

- When starting a Headless Client from a Debug Client in Servoy Developer, there are restrictions to which solution can be started in the Headless Client.
- Prior to Servoy 6.1.1, the Headless Client would always start with the Active Solution, regardless of which solution would be specified
  - As of Servoy 6.1.1 the specified solution will be honored, as long as the specified solution is the Active Solution or a solution/module included in the Active Solution

By default the Headless Client which is started will be a debug-able Headless Client. As there can be only one debug-able Client of a specific type at the same time (so one debug-able Web Client, one debug-able Smart Client and one debug-able Headless Client), starting a new debug-able Headless Client will close an existent debug-able Headless Client.

It's possible to start non-debug-able Headless Clients using the Headless Client plugin while in Servoy Developer by sending in the value 'nodebug' as the last value in the `solutionOpenMethodArgs` array parameter. In this case it is also possible to start the Headless Client with a solution/module that isn't the Active Solution or one of it's modules, as long as the specified solution is available in the workspace and uses the same Resources project as the Active Solution.



Supported Clients

SmartClient WebClient NGClient

Methods Summary		
JSCient	<code>createClient(solutionName, username, password, solutionOpenMethodArgs)</code>	Creates a headless client on the server that will open the given solution.
JSCient	<code>getClient(clientID)</code>	Gets an existing headless client for the given client uuid.
JSCient	<code>getOrCreateClient(clientId, solutionname, username, password, solutionOpenMethodArgs)</code>	This will try to get a existing client by the given id if that client is already created for that specific solution; it will create a headless client on the server that will open the given solution if it didn't exists yet.

Methods Details

**createClient(solutionName, username, password, solutionOpenMethodArgs)**  
Creates a headless client on the server that will open the given solution.  
The `clientId` of this client can be stored in the database to be shared between clients so that that specific client can be used over multiply clients later on or picked up later on by this client. (Even after restart of this client)  
  
NOTE: in the developer this will only load one solution in debug mode when it is the current active solution or a module of the active solution.  
So calling `createClient` with the same or another solution/module will replace the existig debug client.  
You can load any solution (and have multily JSCient instances) from the workspace when you pass "nodebug" as last argument in the arguments list (it should still use the same resources project).  
But then you won't be able to debug it, breakpoints won't hit.

Parameters

`String` solutionName ;  
`String` username ;  
`String` password ;  
`Array` solutionOpenMethodArgs ;

Returns

JSCient

Supported Clients

SmartClient,WebClient,NGClient

**Sample**

```
// Creates a headless client that will open the given solution.
var headlessClient = plugins.headlessclient.createClient("someSolution", "user", "pass", null);
if (headlessClient != null && headlessClient.isValid()) {
    var x = new Object();
    x.name = 'remotel';
    x.number = 10;
    headlessClient.queueMethod(null, "remoteMethod", [x], callback);
}
```

**getClient(clientID)**

Gets an existing headless client for the given client uuid.

**Parameters**

[String](#) clientID;

**Returns**

[JSClient](#)

**Supported Clients**

SmartClient,WebClient,NGClient

**Sample**

```
// Gets an existing headless client for the given client uuid.
var headlessClient = plugins.headlessclient.getClient("clientID");
if (headlessClient != null && headlessClient.isValid()) {
    headlessClient.queueMethod(null, "someRemoteMethod", null, callback);
}
```

**getOrCreateClient(clientId, solutionname, username, password, solutionOpenMethodArgs)**

This will try to get a existing client by the given id if that client is already created for that specific solution;

it will create a headless client on the server that will open the given solution if it didn't exists yet.

If the client does exist but it is not loaded with that solution an exception will be thrown.

NOTE: in the developer this will only load the solution in debug mode when it is the current active solution or a module of the active solution;

you can load any solution from the workspace when you pass "nodebug" as last argument in the arguments list (it should still use the same resources project).

But then you won't be able to debug it, breakpoints won't hit.

**Parameters**

[String](#) clientId The client to lookup by id, if not found a new headless client is created with this id.

[String](#) solutionname The solution to load

[String](#) username The user name that is used to login to the solution

[String](#) password The password for the user

[Array](#) solutionOpenMethodArgs The arguments that will be passed to the solution open method.

**Returns**

[JSClient](#)

**Supported Clients**

SmartClient,WebClient,NGClient

**Sample**

```
// Creates a headless client that will open the given solution.
var storedSolutionSpecificID = "aaaabbbbcccccll11";
var headlessClient = plugins.headlessclient.getOrCreateClient(storedSolutionSpecificID, "someSolution", "user", "pass", null);
if (headlessClient != null && headlessClient.isValid()) {
    var x = new Object();
    x.name = 'remotel';
    x.number = 10;
    headlessClient.queueMethod(null, "remoteMethod", [x], callback);
}
```

