Important: The current version of SuiteCloud IDE does not support logging in to a project account for some Linux distributions, depending on the desktop environment used. However, you can log into a project account on Windows and Mac OS.

To log in to a project account:

1. Launch SuiteCloud IDE.
2. Right-click a file or a project in the NS Explorer pane, and select **NetSuite** > **Log in to Project Account**.

   A browser opens the Choose Role page of the NetSuite account associated with the file or project.

SuiteCloud IDE Debugger

A debugger enables you to interactively control the execution of your code so that you can monitor the progress of its variables and output. The SuiteCloud IDE Debugger enables you to debug server-side SuiteScript in the same Eclipse-based environment you develop it in. The debugger utilizes the same keyboard shortcuts and menu options found in the standard version of Eclipse. Developers familiar with the Eclipse Debugger find the SuiteCloud IDE Debugger similar.

The SuiteCloud IDE Debugger provides you with two debugging modes. The one you use depends on whether you need to debug a script fragment or a fully defined and deployed server-side script. Both modes support the use of multiple breakpoints and the standard debugging functionality (for example: step through, inspect, watch expression, variable evaluation and log output). Multi-file debugging is only supported with deployed scripts.

- Single File Mode (Ad Hoc Debugging): Enables you to debug a single script file that does not have a defined script and script deployment record.
- Project Mode (Deployed Debugging): Enables you to debug a multi-file project that has a defined script and script deployment record. Deployed debugging incorporates an integrated browser session for your NetSuite account and a client for both Suitelets and RESTlets. Note that you must be the owner of the script to use deployed debugging. Deployed debugging is supported with the following server-side script types:
  - Mass Update
  - Portlet
  - RESTlet
  - Suitelet
  - User Event
  - Workflow Action

Note: The SuiteCloud IDE Debugger does not support client-side debugging. To debug your client SuiteScripts, NetSuite recommends using the Chrome DevTools for Chrome, the Firebug debugger for Firefox, and the Microsoft Script Debugger for Internet Explorer. For additional information about these tools, see the documentation provided with each browser.

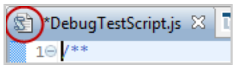
The following topics provide details about how to use the SuiteCloud IDE Debugger.

- Verifying Your Editor
- Working with Breakpoints

- Creating Debug Configurations
- Debug Perspective
- Controlling Execution
- Evaluating Expressions in SuiteCloud IDE
- Single File Mode (Ad Hoc Debugging)
- Project Mode (Deployed Debugging)
- Using the RESTlet/Suitelet Debug Client

Verifying Your Editor

Before using the SuiteCloud IDE Debugger, verify your files are open in the SuiteScript editor. The debugger does not recognize breakpoints added in a different editor (for example, the JavaScript editor). If a file is open in the SuiteScript editor, it has the SuiteScript symbol next to its file name within the code editor.



Your files show the JavaScript symbol next to them in the NS Explorer view even if they are open in the SuiteScript editor.

If a file is not open in the SuiteScript editor, right-click on it in the NS Explorer pane, and select **Open With** > **SuiteScript Editor**.

Working with Breakpoints

A breakpoint is a marker you place next to a line of code that tells the SuiteCloud IDE Debugger to pause execution at that line. When you start a new debug session, the debugger executes the code until it reaches the first breakpoint. When the debugger pauses execution at the first breakpoint, you can add breakpoints, remove breakpoints, and temporarily disable or enable breakpoints.

With Project Mode (Deployed Debugging), additional breakpoints can only be added at certain points during execution. You can modify breakpoints when execution pauses at an enabled breakpoint or during the time that you are stepping through the code. You cannot modify breakpoints during the time that the integrated browser is waiting for user input. The context menu for toggle breakpoints is hidden accordingly when breakpoint modification is prohibited.

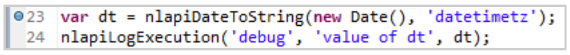
Regardless of whether you are debugging a single file or a project, you must add at least one breakpoint to your code before running the debugger. If you attempt to run the debugger without adding breakpoints to your code, you get an error. Breakpoints must be added within the SuiteScript editor. The debugger does not recognize breakpoints added within other editors (for example, the JavaScript editor). See Verifying Your Editor before you add breakpoints.

For additional information on modifying breakpoints, see:

- Adding Breakpoints
- Removing Breakpoints
- Disabling Breakpoints
- Enabling Breakpoints

Adding Breakpoints

To add a breakpoint, double-click in the gray area to the left of the line number. You can also right-click in the gray area and select **Toggle Breakpoint** from the context menu. Enabled breakpoints are represented as solid blue circles to the left of the line number as shown:



```
var dt = nlapiDateToString(new Date(), 'datetimetz');
nlapiLogExecution('debug', 'value of dt', dt);
```

Note: A breakpoint stays with the same line of code even if the line number changes. Whether line numbers are displayed is controlled by the **Show line numbers** preference on the Text Editors preference page under General > Editors.

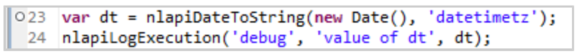
Removing Breakpoints

To remove a breakpoint, double-click on it. You can also right-click on a breakpoint and select **Toggle Breakpoint** from the context menu.

Disabling Breakpoints

When you disable a breakpoint, the marker remains but no longer pauses execution. The code executes as if the breakpoint does not exist.

To disable a breakpoint, right-click on it and select **Disable Breakpoint** from the context menu. Disabled breakpoints are represented as empty circles.



```
var dt = nlapiDateToString(new Date(), 'datetimetz');
nlapiLogExecution('debug', 'value of dt', dt);
```

Enabling Breakpoints

To enable a disabled breakpoint, right-click on it and select **Enable Breakpoint** from the context menu.

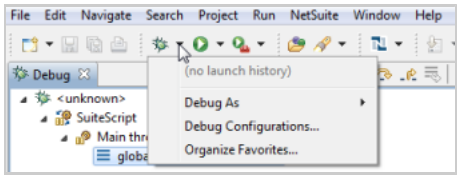
Creating Debug Configurations

A debug configuration enables you to save the debug settings for a file or project so that you enter them only one time. When your code contains a breakpoint, you must create at least one debug configuration. You can create multiple debug configurations for each file or project, but debug configurations cannot be shared.

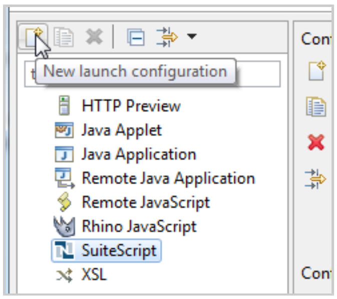
When you create a new debug configuration, a shortcut is added to the debug dropdown menu and the **Debug As** context menu. Use these shortcuts to initiate subsequent debug sessions.

To create a new debug configuration:

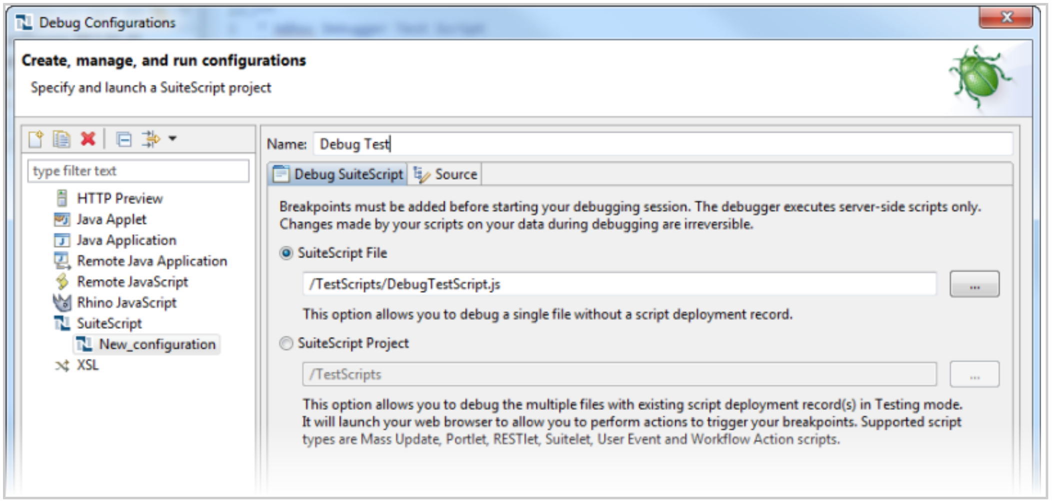
1. Click the debug dropdown menu and select **Debug Configurations.**



2. In the left pane of the Debug Configurations window, select **SuiteScript** and click **New launch configuration** to open the configuration in the right pane.



3. Type a name in the **Name** field.



Name: Debug Test

Breakpoints must be added before starting your debugging session. The debugger executes server-side scripts only. Changes made by your scripts on your data during debugging are irreversible.

SuiteScript File: /TestScripts/DebugTestScript.js

This option allows you to debug a single file without a script deployment record.

SuiteScript Project: /TestScripts

This option allows you to debug the multiple files with existing script deployment record(s) in Testing mode. It will launch your web browser to allow you to perform actions to trigger your breakpoints. Supported script types are Mass Update, Portlet, RESTlet, Suitelet, User Event and Workflow Action scripts.

4. Choose one of these options:
   - **SuiteScript File** – Choose this option for Single File Mode (Ad Hoc Debugging).
   - **SuiteScript Project** – Choose this option for Project Mode (Deployed Debugging).
5. If necessary, click **...** to change the NS Explorer file path.

   Both the project and file name are required for the **SuiteScript File** option.

   Only the project name is required for the **SuiteScript Project** option, but this project name must match the project name in the NetSuite file cabinet.
6. Click **Apply**.
7. To debug your code now, click **Debug**.

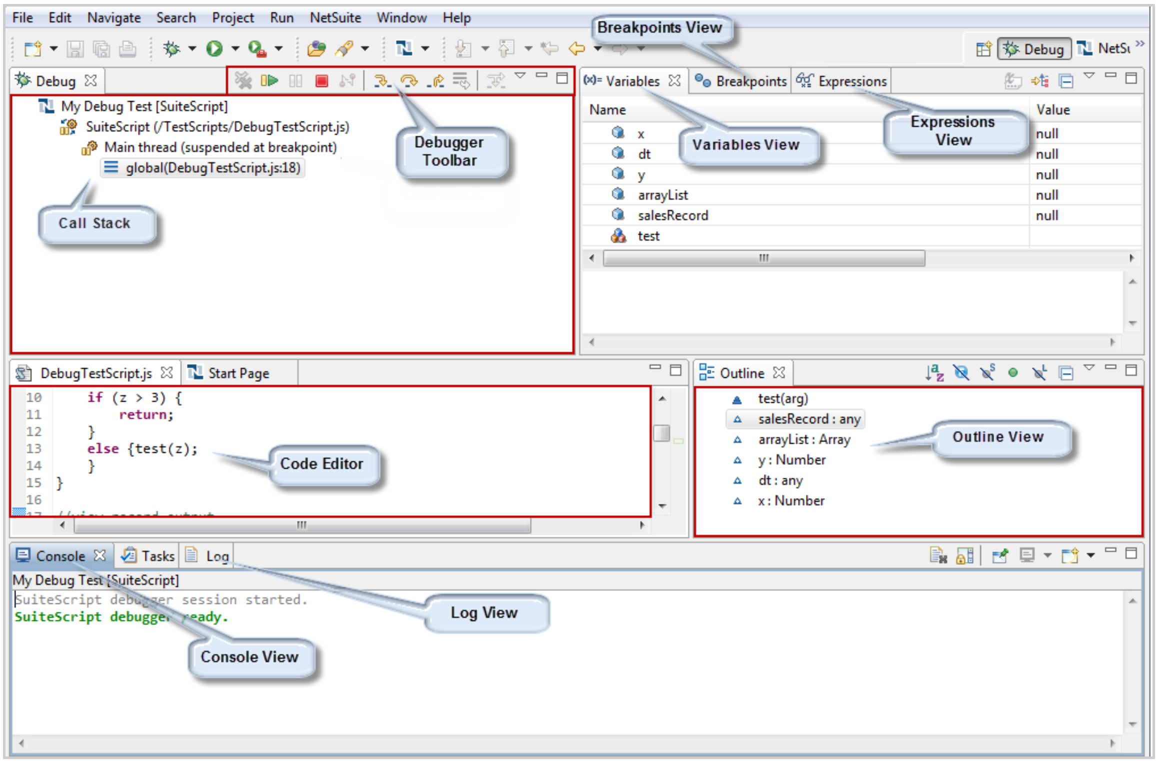
Debug Perspective

The Debug Perspective provides editors, toolbars, and views to debug your SuiteScripts. In the debug perspective, the debugger executes your code until it reaches the first breakpoint. For detailed descriptions, see Controlling Execution and Evaluating Expressions in SuiteCloud IDE.

The debug perspective is composed of the following:

- Breakpoints View
- Call Stack
- Code Editor
- Console View
- Debugger Toolbar

- Expressions View
- Log View
- Outline View
- Variables View



My Debug Test [SuiteScript]
SuiteScript (/TestScripts/DebugTestScript.js)
Main thread (suspended at breakpoint)
global(DebugTestScript.js:18)

Name	Value
x	null
dt	null
y	null
arrayList	null
salesRecord	null
test	

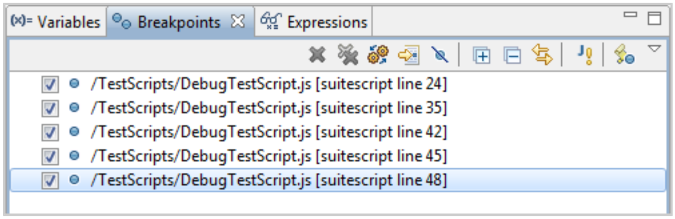
```
    if (z > 3) {
        return;
    }
    else {test(z);
    }
}
```

test(arg)
salesRecord : any
arrayList : Array
y : Number
dt : any
x : Number

```
My Debug Test [SuiteScript]
SuiteScript debugger session started.
SuiteScript debugger ready.
```

Breakpoints View

The breakpoints view lists all enabled and disabled breakpoints set in your SuiteScript.



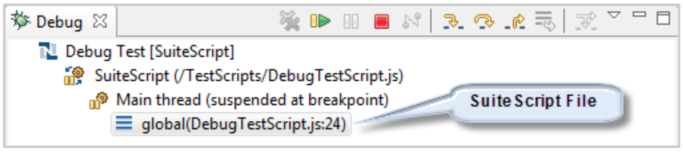
- [✓] /TestScripts/DebugTestScript.js [suitescript line 24]
- [✓] /TestScripts/DebugTestScript.js [suitescript line 35]
- [✓] /TestScripts/DebugTestScript.js [suitescript line 42]
- [✓] /TestScripts/DebugTestScript.js [suitescript line 45]
- [✓] /TestScripts/DebugTestScript.js [suitescript line 48]

In Project Mode, all breakpoints between files are listed. Within the breakpoints view, you can toggle and remove breakpoints, skip all breakpoints, view the breakpoints for a specific file within your project, and automatically go to specific breakpoints within the Code Editor.

Note: Valid breakpoint line numbers are listed as “suitescript line __”; if your line numbers are listed as “line __”, you are using the wrong editor. See Verifying Your Editor for additional information.

Call Stack

The call stack shows the files in your SuiteScript.



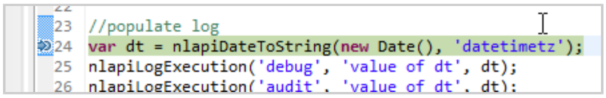
```
Debug Test [SuiteScript]
  SuiteScript (/TestScripts/DebugTestScript.js)
    Main thread (suspended at breakpoint)
      global(DebugTestScript.js:24)
```

- If you are debugging a single file, the call stack shows that one file.
- If you are debugging a project, each file within the project is displayed. The currently targeted file is highlighted as you progress through the code. When execution is paused, you can switch to a different file by clicking on it. Switching files causes the other views to update accordingly.

From the call stack, you can also access the context menu options for Step Into, Step Over, Step Return, Resume and Terminate. For additional information, see Controlling Execution.

Code Editor

The code editor shows the code for the currently targeted file in your SuiteScript.



```
//populate log
var dt = nlapiDateToString(new Date(), 'datetimetz');
nlapiLogExecution('debug', 'value of dt', dt);
nlapiLogExecution('audit', 'value of dt', dt);
```

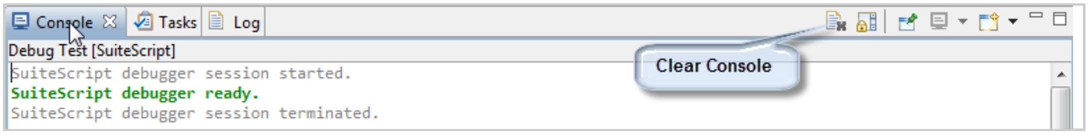
As you step through your script, the code editor automatically scrolls through the source code. The line highlighted in green is the line set to execute next. The blue arrow to the left of the line numbers points to the last breakpoint encountered by the debugger. For additional information, see Working with Breakpoints.

Important: Do not edit your source code during a debug session. Doing so terminates the session.

From the code editor, you can also access the context menu options for Run to Line, Watch and Inspect.

Console View

The console view displays prompts and messages.



```
Debug Test [SuiteScript]
SuiteScript debugger session started.
SuiteScript debugger ready.
SuiteScript debugger session terminated.
```

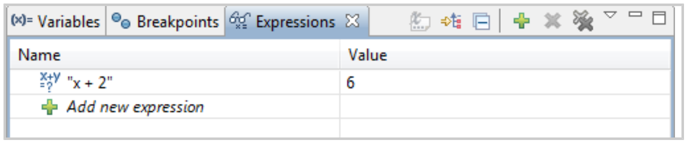
System errors and messages are displayed in the Log View.

Debugger Toolbar

The debugger toolbar provides buttons to resume execution, terminate execution, and step through your code. For additional information, see Controlling Execution.

Expressions View

When an expression is added to the expressions view, you can monitor the progress of it as your code executes.



Name	Value
"x + 2"	6
Add new expression	

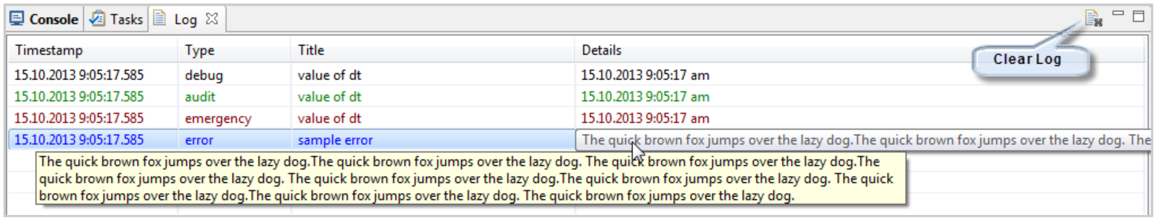
Add expressions to the list in the following ways:

- Click the green plus in the Name column. Type the expression in the text field, and then click outside of the field.
- Highlight an existing expression in your code. Then right click and select **Watch**. For additional information, see Evaluating Expressions in SuiteCloud IDE.

Log View

If your code uses nlapiLogExecution(type, title, details) to log entries to NetSuite, those log entries display in the log view. Log entries are color coded to distinguish between log types.

To view long error messages, hover over the beginning of the message in the Details column.



Timestamp	Type	Title	Details
15.10.2013 9:05:17.585	debug	value of dt	15.10.2013 9:05:17 am
15.10.2013 9:05:17.585	audit	value of dt	15.10.2013 9:05:17 am
15.10.2013 9:05:17.585	emergency	value of dt	15.10.2013 9:05:17 am
15.10.2013 9:05:17.585	error	sample error	The quick brown fox jumps over the lazy dog.The quick brown fox jumps over the lazy dog. The

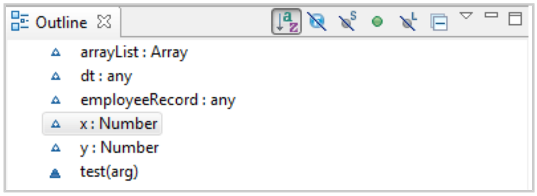
The quick brown fox jumps over the lazy dog.The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog.The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog.The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog.The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog.

The log view also displays system messages and errors. For example, when a debug session is complete or terminated, the log view displays a metrics entry that lists governance usage and session run time (in milliseconds).

You can select all log entries and copy and paste them into an external application such as Microsoft Excel.

Outline View

The outline view lists the structural elements of your SuiteScript. During execution, when an element is selected in the Code Editor, it is also selected in the outline view.

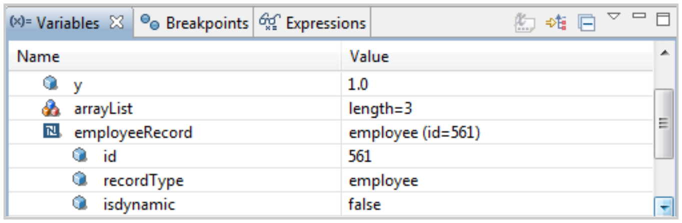


arrayList : Array
dt : any
employeeRecord : any
x : Number
y : Number
test(arg)

With Project Mode (Deployed Debugging), this view shows you the elements in the currently targeted file. If you manually switch to a different file in the Call Stack, the outline view changes accordingly.

Variables View

The variables view shows you the current value of your SuiteScript variables.



Name	Value
y	1.0
arrayList	length=3
employeeRecord	employee (id=561)
id	561
recordType	employee
isdynamic	false

The values change as you step through your code. In Project Mode, this view shows you the variables in the currently targeted file. If you manually switch to a different file in the Call Stack, the variables view changes accordingly.

Controlling Execution

The SuiteCloud IDE Debugger provides the following tools to control the execution of your code. Toggle Breakpoint is available before you start your debug session and when the debugger is paused at a breakpoint. The other tools listed are only available during an active debug session when the debugger is paused at a breakpoint.

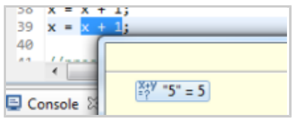
The buttons listed are found on the Debugger Toolbar.

Functionality	Button / Menu Option	Shortcut Key	Description
Toggle Breakpoint	No button See Working with Breakpoints.	Ctrl+Shift+B	Toggles a breakpoint on the currently selected line when in the Debug Perspective. This shortcut key does not work when in the NetSuite Perspective.
Resume	Right-click within the Call Stack and select **Resume**.	F8	Resumes execution until the debugger reaches the next breakpoint.
Step Into	Right-click within the Call Stack and select **Step Into**.	F5	Executes the currently selected line and then advances to the next line without executing it. If the selected line is a function (nlapi) or method call, the debugger steps into the first line of the function or method body.
Step Over	Right click within the Call Stack and select **Step Over**.	F6	Executes the currently selected line and then advances to the next line without executing it. If the selected line is a function (nlapi) or method call, the debugger executes it without stepping into the function or method body.
Step Return	Right-click within the Call Stack and select **Step Return**.	F7	Steps out of the current function or method, and then advances to the next line without executing it.
Run to Line	No button Right-click within the Code Editor and select **Run To Line**.	Ctrl+R	Advances the debugger to the current position of your cursor. All code preceding the target line is executed. The target line itself is not executed. If the debugger encounters a breakpoint before the target line, it pauses execution.

Functionality	Button / Menu Option	Shortcut Key	Description
Terminate	🟥 Right-click within the Call Stack and select **Terminate**.	Ctrl+F2	Stops execution of the debugger. You must start a new session if you wish to continue debugging.
Relaunch	No button Right-click within the Call Stack and select **Relaunch**.	F11	Starts a new debug session identical to the last session executed.
Terminate and Relaunch	No button Right-click within the Call Stack and select **Terminate and Relaunch**.	Ctrl+Alt+Shift+R	Stops execution of the debugger. Then starts a new debug session identical to the session terminated.

Evaluating Expressions in SuiteCloud IDE

SuiteCloud IDE provides the following tools to help you monitor your code as you step through it. These tools are only available during an active debug session when the debugger is paused at a breakpoint.

Functionality	Menu Option	Shortcut Key	Description
Inspect	Right-click within the Code Editor and select **Inspect**.	Ctrl+Shift+Alt+I	Evaluates a highlighted expression, variable or object. `x = x + 1;` "5" = 5 In this example, at this point in the execution, the value of x = 4. Since x + 1 is the expression highlighted for inspection, the value specified is 5.
Watch	Right-click within the Code Editor and select **Watch**.	Ctrl+Shift+Alt+W	Adds a highlighted expression to the Expressions View.

Single File Mode (Ad Hoc Debugging)

With ad hoc debugging, you can debug a single SuiteScript file that is not uploaded and deployed to NetSuite.

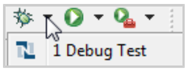
Note: Do not log into your NetSuite account from SuiteCloud IDE or your browser for the duration of the debug session. This may cause your debug session to end.

To debug a single SuiteScript file that is not deployed:

1. Open the SuiteScript file in the SuiteCloud IDE SuiteScript editor. For instructions, see Verifying Your Editor.
2. Ensure that your code has an entry point. In other words, if you declare a function, make sure you also call it.

```
function test(arg){...
}
test(1);
```

3. Add at least one breakpoint. For information, see Working with Breakpoints.
4. If needed, create a new debug configuration for a SuiteScript file. For instructions, see Creating Debug Configurations. If you are using an existing debug configuration, select its shortcut.



The view switches from the NetSuite Perspective to the Debug Perspective. The debugger executes until it reaches the first breakpoint. When execution pauses at the first breakpoint, you can use the available execution controls to step though your code. For information, see Controlling Execution.

Important: Do not edit your source code during a debug session. Doing so terminates the session.

Project Mode (Deployed Debugging)

- SuiteScript 2.0 Deployed Debugging
- SuiteScript 1.0 Deployed Debugging

With deployed debugging, you can debug multiple files (within a project) in a single debug session.

To run deployed debugging, you must be the owner of the script. In addition, you must create a script record and script deployment record. The script deployment record status must be set to Testing. For workflow action scripts, the status must be set to Released.

Deployed debugging is supported with the following server-side script types:

- Mass Update
- Portlet
- RESTlet
- Suitelet
- User Event
- Workflow Action

Warning: Changes made to a account during debugging are irreversible. Users should exercise caution when debugging on production accounts. When possible, use sandbox accounts to debug your SuiteScripts.

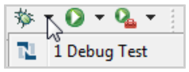
Note: Do not log into your NetSuite account from SuiteCloud IDE or your browser for the duration of the debug session. This may cause your debug session to terminate. Deployed debugging includes an integrated browser session that opens your NetSuite account when the debugger starts.

SuiteScript 2.0 Deployed Debugging

To debug a deployed SuiteScript 2.0 project:

1. Upload the SuiteScript files in your project to NetSuite as a library file. For instructions, see Uploading Files in a SuiteScript Project.

2. Create a script record. For information, see the help topic Creating a Script Record.
3. Create a script deployment record and set the status to **Testing**. For information, see the help topic Methods of Deploying a Script.
4. Open your SuiteScript files in the SuiteCloud IDE SuiteScript editor. For instructions, see Verifying Your Editor.
5. Add at least one breakpoint. For information, see Working with Breakpoints.
6. If needed, create a new debug configuration for a SuiteScript project and then select **Debug**. For instructions, see Creating Debug Configurations. If you are using an existing debug configuration, select its shortcut.



A browser window opens the NetSuite account associated with your project.

> **Important:** Do not edit your source code during a debug session. Doing so terminates the session.

7. Trigger your SuiteScript from within this browser session.

The debugger executes until it reaches the first breakpoint. When execution pauses at the first breakpoint, you can use the available execution controls to step through your code. For information, see Controlling Execution.

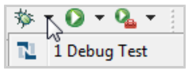
> **Important:** When deployed debugging is initiated, additional breakpoints can only be added at certain points during execution. You can modify breakpoints when execution pauses at an enabled breakpoint or during the time that you are stepping through the code. You cannot modify breakpoints when the integrated browser is waiting for user input. The context menu for toggle breakpoints is hidden during the time that breakpoint modification is prohibited.

If you are debugging a RESTlet or Suitelet SuiteScript, see Using the RESTlet/Suitelet Debug Client.

SuiteScript 1.0 Deployed Debugging

To debug a deployed SuiteScript 1.0 project:

1. Upload the SuiteScript files in your project to NetSuite as a library file. For instructions, see Uploading Files in a SuiteScript Project.
2. Create a script record. For information, see the help topic Steps for Creating a Script Record.
3. Create a script deployment record and set the status to **Testing**. For information, see the help topic Steps for Defining a Script Deployment.
4. Open your SuiteScript files in the SuiteCloud IDE SuiteScript editor. For instructions, see Verifying Your Editor.
5. Add at least one breakpoint. For information, see Working with Breakpoints.
6. If needed, create a new debug configuration for a SuiteScript project and then select **Debug**. For instructions, see Creating Debug Configurations. If you are using an existing debug configuration, select its shortcut.



A browser window opens the NetSuite account associated with your project.

> ⚠ **Important:** Do not edit your source code during a debug session. Doing so terminates the session.

7. Trigger your SuiteScript from within this browser session.

   The debugger executes until it reaches the first breakpoint. When execution pauses at the first breakpoint, you can use the available execution controls to step through your code. For information, see Controlling Execution.

   > ⚠ **Important:** When deployed debugging is initiated, additional breakpoints can only be added at certain points during execution. You can modify breakpoints when execution pauses at an enabled breakpoint or during the time that you are stepping through the code. You cannot modify breakpoints when the integrated browser is waiting for user input. The context menu for toggle breakpoints is hidden during the time that breakpoint modification is prohibited.

   If you are debugging a RESTlet or Suitelet SuiteScript, see Using the RESTlet/Suitelet Debug Client.

Using the RESTlet/Suitelet Debug Client

The RESTlet/Suitelet Debug Client enables you to debug deployed RESTlet and Suitelet SuiteScripts with the SuiteCloud IDE Debugger. The client is only accessible when a debug session is started.

To use the RESTlet/Suitelet client:

1. Start a deployed debug session. For instructions, see Project Mode (Deployed Debugging).
2. Within the Code Editor, right-click **NetSuite** > **Debug RESTlet/Suitelet**. You can also use the keyboard shortcut Ctrl+Shift+Alt+T.
3. Enter the **Relative URL**. The **Relative URL** is the URL used to invoke the RESTlet (for example, "/app/site/hosting/restlet.nl?script=1&deploy=1"). For additional information, see the help topic RESTlet URL and Domain.
4. Select the appropriate HTTP method from the **Method** dropdown list. This option sets the HTTP method to call.
5. If needed, click **...** to the right of the **Optional Log File** field to map the response to a log output file.
   - If a log file is not specified, the response is logged in the Console View.
   - If a log file is specified, the response is logged in the Console View Console View and in the specified log file. If the specified log file already contains information, the result is appended to the existing file contents.
6. If needed, click the **Headers** tab. From the **Headers** tab, you can do the following:
   - Click **New** to add a new header.
   - Select an existing header and click **Edit** to edit it.
   - Select one or more existing headers and click **Remove** to remove them.
   - Click **Remove All** to remove all existing headers.
7. If you are adding or editing a header, enter or edit the **Name** and **Value** fields. Then click **OK**.

> **Important:** Header names must be unique. If you attempt to add an existing name or edit a name so that it matches an existing name, you get an error message.

8. If needed, click the **Body** tab. Select one of the following options:
   - **None**: Select if no parameters are to be sent with the request.
   - **String Body**: Add the appropriate text to the text box. This option is typically used to specify a JSON string for RESTlets.

     > **Important:** To use JSON as a content type for RESTlets, you must set **Content–Type = application/json** in the **HTTP Content –Type** header. If **Content —Type = text/plain** is set instead, the string entered in the text box is treated as an ordinary string.

   - **File Body**: Click **...** to map to the location of the file. Only one file is allowed.
   - **Multipart Body**: The parameters sent can be either a File or a String. For Strings, the value of the string is displayed. For Files, the value displayed is in the format **<absolute file path>|Mime Type=<specified mime type>|Encoding=<specified encoding>**. For this option, you can do the following:
     - Click **New** to add a new String parameter. Enter the Name and Value fields. Then click **OK**.
     - Click **New File** to add a new File parameter. Note that this option is only enabled for methods POST and PUT. For additional information, see Adding or Editing a Multipart Body File Parameter.
     - Select an existing parameter and click **Edit** to edit it. If the parameter is a String, edit the **Name** and **Value** fields and click **OK**. If the parameter is a File, for additional information, see Adding or Editing a Multipart Body File Parameter.

Adding or Editing a Multipart Body File Parameter

The **Encoding** field is optional. All other fields are required.

To add or edit a new File parameter:

1. Enter or edit the **Name** field.
2. Select a new or different value from the **Mime Type** dropdown list. For a list of SuiteScript supported content types, see the help topic Supported File Types.
3. If the file is encoded, select a new or different value from the **Encoding** dropdown list.
4. Click **...** to map to the location of the file.
5. Click **OK**.