should not be relied upon in making purchasing decisions. The development, release, and timing of any features or functionality described in this document remains at the sole discretion of Oracle.

This document in any form, software or printed matter, contains proprietary information that is the exclusive property of Oracle. Your access to and use of this confidential material is subject to the terms and conditions of your Oracle Master Agreement, Oracle License and Services Agreement, Oracle PartnerNetwork Agreement, Oracle distribution agreement, or other license agreement which has been executed by you and Oracle and with which you agree to comply. This document and information contained herein may not be disclosed, copied, reproduced, or distributed to anyone outside Oracle without prior written consent of Oracle. This document is not part of your license agreement nor can it be incorporated into any contractual agreement with Oracle or its subsidiaries or affiliates.

For information about Oracle's commitment to accessibility, visit the Oracle Accessibility Program website at http://www.oracle.com/pls/topic/lookup?ctx=acc&id=docacc

Oracle customers that have purchased support have access to electronic support through My Oracle Support. For information, visit http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info or visit http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs if you are hearing impaired.

Sample Code

Oracle may provide sample code in SuiteAnswers, the Help Center, User Guides, or elsewhere through help links. All such sample code is provided “as is” and “as available”, for use only with an authorized NetSuite Service account, and is made available as a SuiteCloud Technology subject to the SuiteCloud Terms of Service at www.netsuite.com/tos.

Oracle may modify or remove sample code at any time without notice.

No Excessive Use of the Service

As the Service is a multi-tenant service offering on shared databases, Customer may not use the Service in excess of limits or thresholds that Oracle considers commercially reasonable for the Service. If Oracle reasonably concludes that a Customer's use is excessive and/or will cause immediate or ongoing performance issues for one or more of Oracle's other customers, Oracle may slow down or throttle Customer's excess use until such time that Customer's use stays within reasonable limits. If Customer's particular usage pattern requires a higher limit or threshold, then the Customer should procure a subscription to the Service that accommodates a higher limit and/or threshold that more effectively aligns with the Customer's actual usage pattern.

Beta Features

Oracle may make available to Customer certain features that are labeled “beta” that are not yet generally available. To use such features, Customer acknowledges and agrees that such beta features are subject to the terms and conditions accepted by Customer upon activation of the feature, or in the absence of such terms, subject to the limitations for the feature described in the User Guide and as follows: The beta feature is a prototype or beta version only and is not error or bug free and Customer agrees that it will use the beta feature carefully and will not use it in any way which might result in any loss, corruption or unauthorized access of or to its or any third party's property or information. Customer must promptly report to Oracle any defects, errors or other problems in beta features to support@netsuite.com or other designated contact for the specific beta feature. Oracle cannot guarantee the continued availability of such beta features and may substantially modify or cease providing such beta features without entitling Customer to any refund, credit, or other compensation. Oracle makes no representations or warranties regarding functionality or use of beta features and Oracle shall have no liability for any lost data, incomplete data, re-run time, inaccurate input, work delay, lost profits or adverse effect on the performance of the Service resulting from the use of beta features. Oracle's standard service levels, warranties and related commitments regarding the Service shall not apply to beta features and they may not be fully supported by Oracle's customer support. These limitations and exclusions shall apply until the date that Oracle at its sole option makes a beta feature generally available to its customers and partners as part of the Service without a “beta” label.
Send Us Your Feedback

We'd like to hear your feedback on this document.

Answering the following questions will help us improve our help content:

- Did you find the information you needed? If not, what was missing?
- Did you find any errors?
- Is the information clear?
- Are the examples correct?
- Do you need more examples?
- What did you like most about this document?

Click here to send us your comments. If possible, please provide a page number or section title to identify the content you're describing.

To report software issues, contact NetSuite Customer Support.
# Table of Contents

SuiteCloud IDE Plug-in for Eclipse Overview ................................................................. 1  
Setting Up SuiteCloud IDE Plug-in for Eclipse ............................................................. 2  
  Installing and Setting Up SuiteCloud IDE Plug-in for Eclipse ..................................... 2  
    SuiteCloud IDE Plug-in for Eclipse Installation Prerequisites .................................... 3  
    Supported Operating Systems and Browsers for SuiteCloud IDE Plug-in for Eclipse .... 3  
    Downloading Eclipse for Use with SuiteCloud IDE Plug-in ...................................... 3  
    Updating Eclipse for Use with SuiteCloud IDE Plug-in ............................................. 4  
Launching SuiteCloud IDE Plug-in for Eclipse and Using the Account Setup Wizard .......... 6  
Selecting a Workspace ................................................................................................. 7  
SuiteCloud IDE Plug-in for Eclipse Master Password ...................................................... 7  
  Setting a SuiteCloud IDE Plug-in for Eclipse Master Password .................................... 8  
  Authenticating a SuiteCloud IDE Plug-in for Eclipse Master Password ....................... 8  
  Revoking a SuiteCloud IDE Plug-in for Eclipse Master Password ................................ 8  
  Changing a SuiteCloud IDE Plug-in for Eclipse Master Password ............................... 9  
  Resetting Your SuiteCloud IDE Plug-in for Eclipse Master Password and Account Info .. 9  
SuiteCloud IDE Plug-in for Eclipse Domain Setup ....................................................... 11  
  Adding a Domain in SuiteCloud IDE Plug-in for Eclipse ........................................... 10  
  Modifying a Domain in SuiteCloud IDE Plug-in for Eclipse ...................................... 10  
  Removing a Domain from SuiteCloud IDE Plug-in for Eclipse ................................... 10  
SuiteCloud IDE Plug-in for Eclipse Account Setup ...................................................... 11  
  Adding an Account in SuiteCloud IDE Plug-in for Eclipse ........................................ 11  
  Removing an Account from SuiteCloud IDE Plug-in for Eclipse ............................... 11  
  Managing Token-based Authentication in an Account using SuiteCloud IDE Plug-in for Eclipse ................................................................. 12  
Importing Existing SuiteCloud Projects into SuiteCloud IDE Plug-in for Eclipse .......... 12  
Synchronizing Internal IDs between your Account and SuiteCloud IDE Plug-in for Eclipse ... 14  
Setting SuiteCloud IDE Plug-in for Eclipse Preferences ............................................... 15  
  Setting SDF Project Preferences in SuiteCloud IDE Plug-in for Eclipse ....................... 16  
  Setting SuiteScript Preferences in SuiteCloud IDE Plug-in for Eclipse ....................... 16  
  Restoring SuiteCloud IDE Plug-in for Eclipse Default Preferences ............................ 18  
SuiteCloud IDE Plug-in for Eclipse Usage .................................................................. 19  
  NetSuite Perspective in SuiteCloud IDE Plug-in for Eclipse ..................................... 19  
  SuiteCloud IDE Plug-in for Eclipse Tips and Guidelines ........................................... 21  
    Shorthand for Adding Event Handlers to Code in SuiteCloud IDE Plug-in for Eclipse ... 21  
    Validation Markers in SuiteCloud IDE Plug-in for Eclipse ...................................... 23  
    SuiteCloud IDE Plug-in for Eclipse Guidelines ....................................................... 24  
SuiteCloud IDE Plug-in for Eclipse Shortcuts ............................................................. 25  
  Code Completion Shortcuts in SuiteCloud IDE Plug-in for Eclipse ............................ 25  
  NetSuite Command Shortcuts in SuiteCloud IDE Plug-in for Eclipse ......................... 28  
Working with SDF Projects in SuiteCloud IDE Plug-in for Eclipse ............................... 29  
  Creating an Account Customization Project in SuiteCloud IDE Plug-in for Eclipse ........ 30  
  Creating a SuiteApp Project in SuiteCloud IDE Plug-in for Eclipse ........................... 30  
  Importing Account Components into an SDF Project with SuiteCloud IDE Plug-in for Eclipse .......................................................... 31  
  Working with Custom NetSuite Objects in SuiteCloud IDE Plug-in for Eclipse .......... 31  
  Uploading Files or Folders to Your Account with SuiteCloud IDE Plug-in for Eclipse .... 32  
  Defining Dependencies on SuiteApp and Bundle Objects using the Eclipse Plug-in ...... 35  
  Defining Account Component Dependencies from an Account Customization Project in SuiteCloud IDE Plug-in for Eclipse .......................... 36  
  Defining Feature Dependencies in SuiteCloud IDE Plug-in for Eclipse ....................... 37  
  Configuring Account Features in an Account Customization Project in SuiteCloud IDE Plug-in for Eclipse ......................................................... 39  
  Setting Installation Preferences in a SuiteApp Project in SuiteCloud IDE Plug-in for Eclipse .......................................................... 41  
  Managing SDF Projects as Compressed Files in SuiteCloud IDE Plug-in for Eclipse .... 44
Validating an SDF Project in SuiteCloud IDE Plug-in for Eclipse ............................................. 45
Deploying an SDF Project to Your NetSuite Account in SuiteCloud IDE Plug-in for Eclipse ........ 46
Working with SuiteScript Projects in SuiteCloud IDE Plug-in for Eclipse ...................................... 47
Creating a SuiteScript Project with SuiteCloud IDE Plug-in for Eclipse ...................................... 48
Uploading Files in a SuiteScript Project with SuiteCloud IDE Plug-in for Eclipse ........................ 48
Downloading Files in a SuiteScript Project with SuiteCloud IDE Plug-in for Eclipse .................. 49
Working with SuiteScript Files in SuiteCloud IDE Plug-in for Eclipse ........................................... 49
Creating a SuiteScript File in SuiteCloud IDE Plug-in for Eclipse ............................................ 50
Uploading a SuiteScript File in SuiteCloud IDE Plug-in for Eclipse .......................................... 51
Comparing a SuiteScript File with File Cabinet Copy in SuiteCloud IDE Plug-in for Eclipse ........... 53
Working with SSP Application Projects in SuiteCloud IDE Plug-in for Eclipse ................................ 54
Creating an SSP Application Project with SuiteCloud IDE Plug-in for Eclipse ............................ 54
Uploading Files in an SSP Application Project with SuiteCloud IDE Plug-in for Eclipse ............. 55
Changing Project Settings in SuiteCloud IDE Plug-in for Eclipse ................................................ 56
Converting a Bundle into an Account Customization Project in SuiteCloud IDE Plug-in for Eclipse .... 56
Converting a Project to a SuiteScript Project in SuiteCloud IDE Plug-in for Eclipse .................... 57
Converting a Project to an SSP Application Project in SuiteCloud IDE Plug-in for Eclipse ............ 57
Logging in to a Project Account from SuiteCloud IDE Plug-in for Eclipse ..................................... 58
SuiteCloud IDE Plug-in for Eclipse Debugger ........................................................................... 58
Verifying Your Editor in SuiteCloud IDE Plug-in for Eclipse ................................................... 59
Uploading an SDF Project to Your NetSuite Account in SuiteCloud IDE Plug-in for Eclipse ........ 60
Creating Debug Configurations in SuiteCloud IDE Plug-in for Eclipse ...................................... 60
Debug Perspective in SuiteCloud IDE Plug-in for Eclipse ...................................................... 62
Single File Mode (Ad Hoc Debugging) in SuiteCloud IDE Plug-in for Eclipse ............................ 71
Project Mode (Deployed Debugging) in SuiteCloud IDE Plug-in for Eclipse ................................ 72
Using the RESTlet/Suitelet Debug Client in SuiteCloud IDE Plug-in for Eclipse ....................... 74
Logging in to a Project Account from SuiteCloud IDE Plug-in for Eclipse ..................................... 75
Viewing SuiteCloud IDE Plug-in for Eclipse Error Logs ............................................................. 76
SuiteCloud IDE Plug-in for Eclipse Overview

SuiteCloud IDE plug-in for Eclipse is an integrated development environment (IDE) that is packaged for NetSuite platform development.

The IDE plug-in creates a development environment that offers the following:

- Upload and download of files to and from the NetSuite File Cabinet
- Comparison of files with the NetSuite File Cabinet version
- Validation of internal IDs
- Integration with the SuiteScript Records Browser
- Management of multiple NetSuite accounts
- Support for JSDoc
- Multi-file, cloud-based debugging

Documentation for SuiteCloud IDE plug-in for Eclipse consists of setup topics and topics that explain how to use the Eclipse plug-in:

- Setting Up SuiteCloud IDE Plug-in for Eclipse
- SuiteCloud IDE Plug-in for Eclipse Usage

The IDE plug-in also provides a user interface for SuiteCloud Development Framework (SDF). SDF is a development framework and deployment mechanism. Additional information related to the IDE plug-in is included in the SDF documentation. See the help topic SuiteCloud Development Framework Overview.
Setting Up SuiteCloud IDE Plug-in for Eclipse

Before setting up your IDE plug-in, be sure you have completed the NetSuite configuration tasks described in:

- Configuring NetSuite for SuiteScript if you are using SuiteScript.
- Enabling SDF in Your NetSuite Account (Admin Only) if you are using SuiteCloud Development Framework.

After configuring your account, follow these steps to set up your IDE plug-in:

1. Installing and Setting Up SuiteCloud IDE Plug-in for Eclipse
2. Launching SuiteCloud IDE Plug-in for Eclipse and Using the Account Setup Wizard
3. Selecting a Workspace
4. SuiteCloud IDE Plug-in for Eclipse Master Password
5. SuiteCloud IDE Plug-in for Eclipse Domain Setup (Optional)
8. Importing Existing SuiteCloud Projects into SuiteCloud IDE Plug-in for Eclipse
9. Synchronizing Internal IDs between your Account and SuiteCloud IDE Plug-in for Eclipse
10. Setting SuiteCloud IDE Plug-in for Eclipse Preferences

If you choose to set up a SuiteScript development environment other than the IDE plug-in, see the help topic Working with IDEs Other Than SuiteCloud IDE Plug-ins.

The following help videos demonstrate how to set up SDF and SuiteCloud IDE plug-in for Eclipse:

- SuiteCloud Development Framework: Installing the Eclipse IDE
- SuiteCloud Development Framework: Configuring the Eclipse IDE Plug-in for SDF

Installing and Setting Up SuiteCloud IDE Plug-in for Eclipse

The IDE plug-in is installed from within Eclipse. To use the IDE plug-in, download and install Eclipse, and add the IDE plug-in update site to the list of software updates to install.

**Important:** When your NetSuite account is upgraded, your instance of IDE plug-in is not updated automatically. After your account upgrade, you must manually update the IDE plug-in to match the version of your NetSuite account. You may experience errors related to compatibility by using an older version of the IDE plug-in with the latest version of NetSuite. For more information, see Updating Eclipse for Use with SuiteCloud IDE Plug-in.

The following table shows the supported versions of Eclipse for each IDE plug-in version:

<table>
<thead>
<tr>
<th>SuiteCloud IDE Plug-in Version</th>
<th>Eclipse Version</th>
</tr>
</thead>
<tbody>
<tr>
<td>2019.1</td>
<td>4.5.2 (Mars)</td>
</tr>
</tbody>
</table>
After your account is upgraded, you must manually update the IDE plug-in. The version of the IDE plug-in must match the version of your account before you can use the IDE plug-in to create SuiteCloud Development Framework (SDF) projects.

To install Eclipse and update the IDE plug-in, complete the following steps:

2. **Download Eclipse.** Download and unpack Eclipse. See Downloading Eclipse for Use with SuiteCloud IDE Plug-in.
3. **Update Eclipse.** Update Eclipse with the IDE plug-in from the IDE plug-in update site. See Updating Eclipse for Use with SuiteCloud IDE Plug-in.

SuiteCloud IDE Plug-in for Eclipse Installation Prerequisites

Make sure you verify the following prerequisites before you install and use Eclipse with the IDE plug-in:

- **Operating system and browser** — Verify the operating system version and browser that you want to use with the IDE plug-in. See Supported Operating Systems and Browsers for SuiteCloud IDE Plug-in for Eclipse.
- **Java Runtime Environment (JRE)** — JRE version 1.8.
- **64-bit requirements** — To properly run the 64-bit version of the IDE plug-in, make sure your operating system and Java versions are both 64-bit versions.
- **Web services enabled** — Make sure that SOAP Web Services is enabled in the account. See the help topic Enabling SDF in Your NetSuite Account (Admin Only).

Supported Operating Systems and Browsers for SuiteCloud IDE Plug-in for Eclipse

The IDE plug-in supports and has been tested with the following operating system and browser combinations:

- **Windows 7** — Internet Explorer 10.x, Firefox 41.x, and Chrome 40.x
- **Mac OS X 10.9.3** — Firefox 41.x, Safari 7.0.2, and Chrome 40.x
- **Debian 7** — Firefox 41.x

Downloading Eclipse for Use with SuiteCloud IDE Plug-in

For supported Eclipse versions, see Installing and Setting Up SuiteCloud IDE Plug-in for Eclipse.

**To download and unpack Eclipse:**


   The following operating systems are supported:
   - Windows 32-bit
   - Windows 64-bit
   - Mac OS X (Cocoa) 64-bit
Installing and Setting Up SuiteCloud IDE Plug-in for Eclipse

- Linux 32-bit
- Linux 64-bit

2. Unpack the archive file you downloaded, and place it in the root directory on your hard drive.
   A directory named eclipse is added. Within that directory is an executable named eclipse.

Next you must update Eclipse with the IDE plug-in package. For more information, see Updating Eclipse for Use with SuiteCloud IDE Plug-in.

Updating Eclipse for Use with SuiteCloud IDE Plug-in

You must update Eclipse with the IDE plug-in package from the IDE plug-in update site. Updating Eclipse with the IDE plug-in consists of three steps:

1. Add the SuiteCloud IDE Plug-in for Eclipse Update Site.
2. Install SuiteCloud IDE Plug-in for Eclipse.

**Important:** When your NetSuite account is upgraded, your instance of the IDE plug-in is not updated. After your account upgrade, you must manually update the IDE plug-in to match the version of your NetSuite account. You may experience errors related to compatibility by using an older version of the IDE plug-in with the latest version of NetSuite.

Add the SuiteCloud IDE Plug-in for Eclipse Update Site

When you add the IDE plug-in update site to Eclipse, you can download the latest version of the IDE plug-in that is compatible with your NetSuite account.

**To add the SuiteCloud IDE Plug-in update site in Eclipse:**

1. Launch Eclipse. If you are prompted to select a workspace, accept the default. You can change it later. For more information, see Selecting a Workspace.
2. Go to Windows > Preferences. The Preferences window appears.
3. Add the IDE Plug-in update site as follows:
   a. In the navigator, select Install/Update > Available Software Sites.
   b. Click Add. The Add Site window appears.
   c. Enter the following information:
      
      | Option         | Description                                                                 |
      |---------------|-----------------------------------------------------------------------------|
      | Name          | SuiteCloud IDE Plug-in for Eclipse Update Site                               |
      | Location      | Note: For compatibility reasons, your IDE plug-in version must match the version of your NetSuite account. |
      |               | https://system.netsuite.com/download/suitecloud-sdk/ideplugin/eclipse/19.1 |

   d. Click OK.
   
   **SuiteCloud IDE Update Site** appears in the list of Available Update Sites. Leave all other sites checked.
4. Click OK.

## Install SuiteCloud IDE Plug-in for Eclipse

When you install the IDE plug-in, select the update site as your installation source.

### To install SuiteCloud IDE plug-in for Eclipse:

1. Go to Help > Install New Software. The Install window appears.
2. In the Work with dropdown list, select SuiteCloud IDE Update Site.
3. In the Name column, select SuiteCloud IDE.
4. Click Next to view the installation details.
5. Accept the terms of the license agreement and click Finish.

   Eclipse installs the IDE plug-in. When the installation is complete, click Yes to restart Eclipse.

When Eclipse restarts, the IDE plug-in Setup Wizard opens. For information and instructions, see Launching SuiteCloud IDE Plug-in for Eclipse and Using the Account Setup Wizard.

## Verify the Installation of SuiteCloud IDE Plug-in for Eclipse

When you install the IDE plug-in, information about its version is added to the About Eclipse dialog box. You can verify this installation information at any time.

### To verify the installation of SuiteCloud IDE plug-in for Eclipse:

1. Go to Help > About Eclipse.
2. Click the NetSuite icon or click Installation Details to open the About Eclipse Features window.

   ![Installation Details](image)

   The IDE plug-in version information is displayed.

## Using Existing SDF Projects with a New Eclipse Installation

You can use your existing SDF projects with a new IDE plug-in for Eclipse Mars installation.

To use existing SDF projects with a new installation:

1. Back up your existing Eclipse installation and workspace.


5. If you are using a new workspace for Eclipse Mars, perform the following steps to use your SDF projects with the new workspace:
   a. Copy your existing SDF projects to the new workspace.
   b. To add an existing SDF project to the IDE plug-in, create each project individually using the same name. The IDE plug-in populates the NS Explorer subtab with the existing files for the SDF project that matches the specified name. To create a project, see Creating an Account Customization Project in SuiteCloud IDE Plug-in for Eclipse and Creating a SuiteApp Project in SuiteCloud IDE Plug-in for Eclipse.

Launching SuiteCloud IDE Plug-in for Eclipse and Using the Account Setup Wizard

Use the steps below to launch the IDE plug-in. The first time you launch it you are asked to select a workspace. For more information, see Selecting a Workspace. After you have a workspace selected, you are asked to accept the terms of the license agreement.

Note: The Usage Data Collector (UDC) is an Eclipse feature that may appear occasionally when you launch the IDE plug-in. The UDC gathers data on how you use the Eclipse platform. The collected data is not stored on a NetSuite server. You can disable this feature by selecting Turn UDC feature off the first time it appears. For more information about how UDC works and its terms of use, see Usage Data Collector in the Eclipse documentation.

To launch SuiteCloud IDE Plug-in for Eclipse:

1. Navigate to the folder location of your IDE plug-in.
2. Double-click eclipse.
3. If prompted, select a workspace, and accept the terms of the license agreement.
4. When the SuiteCloud IDE Setup Wizard appears, click Next to create a master password and add NetSuite accounts to the IDE plug-in.

Note: Your IDE plug-in master password is not your NetSuite password. It is used to ensure that the data in your IDE plug-in remains secure.

5. Enter and re-enter a master password, and click Next when you are finished.
6. Complete the Add Account(s) step of the wizard as follows, and click Next when you are finished.
   - Environment – Select the environment that includes the accounts you want to add. You can choose accounts associated with your Production, Release Preview, or Sandbox environments.
   - Email – Enter the email address you use to access the NetSuite accounts in this environment.
   - Password – Enter the NetSuite password you use to access the accounts in this environment.
7. In the Select Accounts step of the wizard, select the NetSuite accounts you want to add to the IDE plug-in. After these accounts are added, you can upload all SuiteScript code changes from the IDE plug-in to File Cabinet in these accounts.
8. Click Next.

9. In the final step of the wizard, check the Create demo project in workspace box if desired. The demo project that is created when you check the box goes with the video tutorial that is available in SuiteAnswers.

The SuiteCloud Development Framework (SDF) documentation also includes a tutorial. NetSuite recommends the SDF tutorial if you plan to create account customization projects. See the help topic SuiteCloud Development Framework Tutorial.

10. Click Finish.

## Selecting a Workspace

After installation, when you first launch the IDE plug-in, you are prompted to select a workspace for your projects. Your workspace is a directory or folder. Navigate to your desired workspace location, select it, and then click OK. You are then asked to accept the terms of the license agreement. You may also be prompted to complete the IDE plug-in Setup Wizard again.

**Important:** You should create a new workspace for the IDE plug-in instead of reusing an existing workspace. This is to avoid the possibility of carrying over incompatible settings from an old workspace.

You may check the Use this as the default and do not ask again box if you want your selected workspace location to become the default location.

To change your workspace, go to File > Switch Workspace, and select a location from the list or select Other. When you select Other, you can select a workspace from a dropdown list or browse for another location.

When you switch workspaces, the IDE plug-in closes and reopens using the new workspace. Since it is a new session, you must authenticate your master password to upload, download, or interact with an account. For instructions, see Authenticating a SuiteCloud IDE Plug-in for Eclipse Master Password.

## SuiteCloud IDE Plug-in for Eclipse Master Password

The IDE plug-in stores all of your account login information every time you add an account. With this, any upload or download operations no longer prompt you to enter your account login information. To prevent unauthorized running of these operations, you must set up a master password. When you have it set up, you are required to enter it only one time per session before any account-driven operations can be done.

**Important:** You need to set up a master password one time for every workspace.

Your master password not only protects all of your account login information, but it also saves you from entering different passwords for different accounts when you have multiple NetSuite accounts.

See the following procedures for working with the master password:

- Setting a SuiteCloud IDE Plug-in for Eclipse Master Password
- Authenticating a SuiteCloud IDE Plug-in for Eclipse Master Password
- Revoking a SuiteCloud IDE Plug-in for Eclipse Master Password
- Changing a SuiteCloud IDE Plug-in for Eclipse Master Password
- Resetting Your SuiteCloud IDE Plug-in for Eclipse Master Password and Account Info
Setting a SuiteCloud IDE Plug-in for Eclipse Master Password

When you install the IDE plug-in the first time, you must set a master password for your workspace to protect all of your NetSuite account login information. If you finished the IDE plug-in Setup Wizard when you launched the IDE plug-in, you can skip this procedure.

**To set a master password:**

1. In Eclipse, select **NetSuite > Master Password > Set Master Password**. The Set Master Password window opens.
2. Enter the following information: **New Master Password** and **Re-enter New Master Password**.
3. Click **OK**.

Authenticating a SuiteCloud IDE Plug-in for Eclipse Master Password

To upload a project, download a project, or interact with your NetSuite account, you must authenticate your master password once per session.

**To authenticate a master password:**

1. In Eclipse, select **NetSuite > Master Password > Authenticate Master Password**. The Authenticate Master Password window opens.
2. Enter the master password.
3. Click **OK**.

You can also authenticate your master password from the NS Explorer pane or the editor area. Right-click on the pane or the editor area, and select **NetSuite > Authenticate Master Password**.

The keyboard shortcut to open the Authenticate Master Password window is **Ctrl+Alt+A**. The shortcut works only when the password has not been authenticated in the session. For more information, see SuiteCloud IDE Plug-in for Eclipse Shortcuts.

Revoking a SuiteCloud IDE Plug-in for Eclipse Master Password

You can revoke your master password to avoid mistaken or unauthorized upload, download, and interaction with your NetSuite account. For example, someone else needs to look at your IDE plug-in editor to do a peer review of your code. You revoke your master password to protect your NetSuite accounts.

**To revoke a master password:**

1. In Eclipse, select **NetSuite > Master Password > Revoke Master Password**.
2. After the Revoke Master Password window appears, click **OK**.

You can also revoke your master password from the NS Explorer pane or the editor area. Right-click on the pane or the editor area, and select **NetSuite > Revoke Master Password**.
Revoking a master password only logs you out. It does not delete the master password.

Changing a SuiteCloud IDE Plug-in for Eclipse Master Password

You can change a master password.

To change a master password:

1. In Eclipse, select **NetSuite > Master Password > Change Master Password**. The Change Master Password window opens.
2. Enter the following information: **Old Master Password**, **New Master Password**, and **Re-enter New Master Password**.
3. Click **OK**.

Resetting Your SuiteCloud IDE Plug-in for Eclipse Master Password and Account Info

If you forgot your master password, you must reset your master password and account information. For security reasons, there is no way for you to retrieve your master password. This is by design.

To reset your master password and account info:

1. In Eclipse, select **NetSuite > Troubleshoot > Reset Master Password and Account Info**. A message appears warning you that your Eclipse plug-in will be restarted and that your master password and all of your account information will be deleted.
2. Click **OK**. The Eclipse plug-in restarts with your workspace still intact but your master password and account info deleted.
3. Set up your master password and account info. For more information, see Setting a SuiteCloud IDE Plug-in for Eclipse Master Password and Adding an Account in SuiteCloud IDE Plug-in for Eclipse.

SuiteCloud IDE Plug-in for Eclipse Domain Setup

After setting up your master password, you can set up a domain in which to run your scripts. Setting up an account and domain is part of the IDE plug-in Setup Wizard. If you used the wizard, you are not required to set up a domain and account again. The three predefined domains in the IDE plug-in are:

- Production
- Release Preview
- Sandbox

To set up a domain, see the following procedures:

- Adding a Domain in SuiteCloud IDE Plug-in for Eclipse
- Modifying a Domain in SuiteCloud IDE Plug-in for Eclipse
- Removing a Domain from SuiteCloud IDE Plug-in for Eclipse
Adding a Domain in SuiteCloud IDE Plug-in for Eclipse

Use the following steps to add a domain to the IDE plug-in.

**To add a domain in SuiteCloud IDE plug-in for Eclipse:**

1. In Eclipse, select **NetSuite > Manage Domains**.  
   The **Manage Domains** window opens.
2. Click **New**.  
   The **New Domain** window opens.
3. In the **Name** and **URL** fields, enter a value.
   
   | Note: | If you do not know the correct URL to use for a particular domain, see the help topic **URLs for Account-Specific Domains**.
4. Click **OK**, and then **Close**.

Modifying a Domain in SuiteCloud IDE Plug-in for Eclipse

Use the following steps to modify an existing domain. You can change its name or point to its updated URL.

**To modify a domain in SuiteCloud IDE Plug-in for Eclipse:**

1. In Eclipse, select **NetSuite > Manage Domains**.  
   The **Manage Domains** window opens.
2. In the **Domains** list, select a domain that you want to edit.
3. Click **Edit**.  
   The **Edit Domains** window opens.
4. From the **Name** and **URL** fields, modify the information you find appropriate.
   
   | Note: | If you do not know the correct URL to use for a particular domain, see the help topic **URLs for Account-Specific Domains**.
5. Click **OK** and then **Close**.

Removing a Domain from SuiteCloud IDE Plug-in for Eclipse

Use the following steps to remove a domain from the IDE plug-in. You can remove unused or unnecessary domains to reduce the options to those you need for your work.

**To remove a domain from SuiteCloud IDE plug-in for Eclipse:**

1. In Eclipse, select **NetSuite > Manage Domains**.  
   The **Manage Domains** window opens.
2. From the **Domains** list, select the environment that you want to remove.
3. Click **Remove**.
SuiteCloud IDE Plug-in for Eclipse Domain Setup

The Remove Domain dialog opens.

4. Click OK, and then Close.

SuiteCloud IDE Plug-in for Eclipse Account Setup

After setting up your master password, you need to set up the NetSuite accounts you will be writing scripts for. Setting up an account and environment are part of the IDE plug-in Setup Wizard. If you used the wizard, you are not required to set up an environment and account again.

If you have set up environments in addition to the three predefined NetSuite environments, you need to configure account-specific details for these additional environments.

To set up an account, see the following procedures:
- Adding an Account in SuiteCloud IDE Plug-in for Eclipse
- Removing an Account from SuiteCloud IDE Plug-in for Eclipse

Adding an Account in SuiteCloud IDE Plug-in for Eclipse

Use the following steps to add an account. The IDE plug-in passes your account information to NetSuite during the authentication process.

**Important:** Ensure that the NetSuite accounts you add comply with NetSuite password requirements regarding the special characters you can use. For more information, see the help topic NetSuite Password Requirements.

When you change your NetSuite password, you must add your accounts again using your latest NetSuite login credentials. However, the IDE plug-in automatically prompts you to enter your latest NetSuite login credentials if you have updated your NetSuite login credentials but have not re-added your accounts in the IDE plug-in.

**To add an account:**

1. In Eclipse, select NetSuite > Manage Accounts. The Manage Accounts window opens.
2. Click Add. The Add Account(s) window opens.
3. Select an environment.
4. Enter the following information: Email and NetSuite Password associated with the email address.
5. Click Next. The Select Account(s) page opens.
6. Select the account you want to add. You may click Select All, or click Deselect All and then select what you need.
7. Click Finish. The Manage Accounts window opens with your added accounts.
   Existing accounts are maintained. Newly-found accounts are added when you select and add them.
8. Click Close.

Removing an Account from SuiteCloud IDE Plug-in for Eclipse

Use the following steps to remove an account. You can remove inactive or unused accounts that you no longer need.
**To remove an account:**

1. In Eclipse, select **NetSuite > Manage Accounts**. The Manage Accounts window opens.
2. In the **Accounts** list, select the account that you want to remove.
3. Click **Remove**. The Remove Account confirmation popup opens.
4. Click **OK** and then **Close**.

**Managing Token-based Authentication in an Account using SuiteCloud IDE Plug-in for Eclipse**

Token-based authentication (TBA) can be used with custom roles to log in to your NetSuite Production account from the IDE plug-in. By using TBA with the IDE plug-in, you experience fewer session terminations when using a single login to develop SDF projects and navigate NetSuite. Additionally, TBA provides increased development environment security. For more information about setting up TBA with your Production account, see the help topic **Getting Started with Token-based Authentication**.

To use TBA, issue a TBA token for a custom role in a Production account and log in to that account with that role assigned. To use account credentials instead of TBA, revoke the active TBA token on the account.

**To issue or revoke a TBA token in SuiteCloud IDE Plug-in for Eclipse:**

1. Ensure that your account administrator has set up a custom role for you to use with SDF. For information about granting permissions to a custom SDF developer role, see the help topic **Assigning the Developer Role (Admin Only)**.
2. In Eclipse, select **NetSuite > Manage Accounts**.
3. In the **Accounts** list, select the account that you want to log in to using token-based authentication.
4. Click **Manage Authentication**. A window appears, showing all the roles that are assigned to the user.
5. In the **Roles** list, select the role that you want to log in to using token-based authentication.
6. To issue or revoke the TBA token on the role, click **Issue Token** or **Revoke Token**.
   - **Issue Token** indicates that TBA is available for the role but not in use. Users with this role are logged into NetSuite from the IDE plug-in using account credentials.
   - **Revoke Token** indicates that the roles uses TBA to log in to NetSuite from the IDE plug-in.
7. Click **Close** on both windows.

Before you can log in with a 2FA role from SDF, you must set up a 2FA Role for SDF Development. For further information, see the help topic **Setting Up a 2FA Role for SDF Development**.

**To connect the 2FA role to your account, save the TBA token in your SDF project:**

1. In Eclipse, select **NetSuite > Manage Accounts**.
2. In the **Accounts** list, select the account that you want to log in to using token-based authentication.
3. Click **Manage Authentication**. A window appears, showing all the roles that are assigned to the user. If **Authentication Type** is set to **Token Based Authentication**, you can save a TBA token for that role.
4. In the **Roles** list, select the role that you want to log in to using token-based authentication.
5. Click **Save Token**.
6. Enter or edit the token ID and token secret issued from NetSuite for that role. For information on how to issue a token in NetSuite, see the help topic Viewing, Editing, Creating, and Revoking TBA Tokens.

7. Click OK.

8. Click Close on both windows.

**Note:** After successfully saving the token, SDF stores the information and you can use the role for SDF features that depend on account login such as validation, deployment.

---

**Importing Existing SuiteCloud Projects into SuiteCloud IDE Plug-in for Eclipse**

If you already have projects in the File Cabinet of your account, synchronize your File Cabinet within the IDE plug-in by importing your projects. Before doing so, ensure that all your project files are up-to-date in the File Cabinet of your account.

**To import existing SuiteCloud projects into SuiteCloud IDE plug-in for Eclipse:**

1. In Eclipse, right-click in the NS Explorer pane, and select **NetSuite > Download Project**. The Download Project window opens.
2. Select an **Account**.
3. Select a **Role**.
4. Click **Get File List**.

**Note:** Hidden bundles, inactive files, and empty folders are excluded from the file list.

5. When the **File(s) to download** list is populated, select the projects you want to import.
6. Choose **Use project name in file cabinet**, or choose **Use this project name** and enter a project name in the field.
7. Click **OK**.
8. Check your project folder to verify that all files were imported.
Synchronizing Internal IDs between your Account and SuiteCloud IDE Plug-in for Eclipse

To use record ID, field ID and saved search ID code completion, you must first synchronize those values with your IDE plug-in. After they are synchronized, you can use your custom fields in code completion. This feature is only available for SuiteScript and SSP Application projects.

The Sync Script IDs from Account wizard takes you through this process step by step. The options displayed in the wizard depend on the ID types you select in the first window.

**Important:** The IDE plug-in supports search ID code completion only for search types supported by both SuiteScript and SOAP Web Services. See the help topics SuiteScript Supported Records and the SOAP Schema Browser for additional information.

To synchronize internal IDs from a project account:

1. In Eclipse, right-click on a project in the NS Explorer View and select NetSuite > Sync Script IDs from Account.
2. Select the ID types you want to synchronize, and click Next.
3. On the Sync Search ID(s) from Account page, select the search IDs you want to synchronize, and click Next.
4. On the Sync Record ID(s) from Account page, select the custom records you want to synchronize, and click **Next**.

![Sync Record ID(s) from Account](image)

This page is skipped if the number of custom records is below the minimum threshold set in your preferences. The page does not appear unless you check the **Record Types** box on the first page of the wizard.

5. Review the list of items to be synchronized, and click **Finish**.

![Sync Script IDs from Account](image)

---

**Setting SuiteCloud IDE Plug-in for Eclipse Preferences**

You can set general preferences, shortcut keys, SDF preferences, and preferences for SuiteScript development in the IDE plug-in.

**To set SuiteCloud IDE plug-in for Eclipse general preferences:**

1. In Eclipse, select **Window > Preferences** to set the following options.
   - Select **NetSuite > Show Start Page on startup** – Check this box to display the start page each time you start the IDE plug-in.
   - Select **NetSuite > Enable file backup (*.bak) for project or file downloads** – Check this box to back up existing files in your project when you download.
   - Select **XML > XML Files > Validation > No Grammar Specified** – If you want to disable this validation, set **No Grammar Specified** to Ignore. The error is expected if you use an object from a project created in a previous NetSuite release.

2. Click **Apply** to save your settings and leave the Preferences window open, or click **OK** to save and close the window.

See the following topics for more information about the IDE plug-in preferences:
Setting SDF Project Preferences in SuiteCloud IDE Plug-in for Eclipse

The SDF project preference page enables you to control the following project preferences:

- Code templates
- Validation

To customize SDF project preferences, load the SDF project in the IDE plug-in and go to Windows > Preferences. In the left pane, expand NetSuite > SDF Project to access the SDF project preference pages.

Code Templates

You can review and edit NetSuite XML definition templates on the Code Templates page. XML definitions are available for all supported custom NetSuite objects. For more information, see the help topic Customizations Supported by SuiteCloud Development Framework.

To edit a specific template, select the template and click Edit. In the Edit Template window, you can make changes to the template source code or restore the source code to the default by clicking Restore Default.

To restore all templates to their default versions, click Restore Defaults.

Validation

You can enable or disable XML validation on the Validation page.

Setting SuiteScript Preferences in SuiteCloud IDE Plug-in for Eclipse

Three types of preferences are available for SuiteScript development in the IDE plug-in: Code Completion, Code Template, and Validation.

Launch your IDE plug-in, and go to Window > Preferences to open the Preferences window.

After you change options, click Apply to save your settings and leave the Preferences window open, or click OK to save and close the window.

SuiteScript Code Completion

The following options are available when you go to NetSuite > SuiteScript > Code Completion in the Preferences window:
Setting SuiteCloud IDE Plug-in for Eclipse Preferences

- **Always use lower case for internal IDs** – When this box is checked, all internal IDs are lowercase.
- **Internal ID Quote Character** – Choose whether single or double quotes are used for internal IDs. Your choice is automatically used for the function paragraph when you press the keyboard shortcut Ctrl+Spacebar.
  
  For example, choose *Single quotes* as the Internal ID Quote Character. Then type `nlaipLoadRecord('salesorder')` and press Ctrl+Spacebar. When you select an internal ID such as *salesorder*, `nlaipLoadRecord('salesorder')` is automatically added to your code.
- **Threshold for displaying Record Types dialog during sync** – If you select Record Types when you are synchronizing internal IDs, this value determines whether a Sync Record ID(s) from Account page is displayed.

For information about using shortcut keys for code completion options, see SuiteCloud IDE Plug-in for Eclipse Shortcuts and Automatic Code Completion in SuiteCloud IDE Plug-in for Eclipse.

SuiteScript Code Templates

The following options are available when you go to NetSuite > SuiteScript > Code Templates in the Preferences window:

- **Code Templates Folder** – The code templates folder may be the same as your workspace folder.
  
  You can create your own custom headers and function templates and save them in a different folder. To use your custom template, change this preference to use the folder where your custom templates are saved. If you do not specify a different folder location, the default templates are used.
- **User Name** – This is the name displayed when you use the `${author}` token in your template.
- **Date Format** – This is the date format for the `${date}` token in the template.

SuiteScript Validation

The following options are available when you go to NetSuite > SuiteScript > Validation in the Preferences window:

- **Validation Type** – Select one of the options in the following table:

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Off</td>
<td>Switches off the validation feature. System performance is not impacted.</td>
</tr>
<tr>
<td>Fast</td>
<td>Determines context within the current file. System performance is average.</td>
</tr>
<tr>
<td>Full</td>
<td>Determines context across different files. System performance is slow.</td>
</tr>
</tbody>
</table>

By default, the validation type is set to Fast.

- **Ignore List** – The ignore list is a text file with one ID per line. You can use it to skip validation if you want to allow non-standard IDs.
  
  NetSuite recommends that you import custom records and fields instead of adding them to the ignore list. For instructions, see Importing Existing SuiteCloud Projects into SuiteCloud IDE Plug-in for Eclipse.

  The following is an example of ignore list file content:

  ```
custrec_ignore_this_field
custrec_ignore_this_field_2
```

For other information about validation for the IDE plug-in, see Validation Markers in SuiteCloud IDE Plug-in for Eclipse.
Setting SuiteCloud IDE Plug-in for Eclipse Preferences

Restoring SuiteCloud IDE Plug-in for Eclipse Default Preferences

Each page in the Preferences window includes a button to enable you to restore the default settings.

To restore SuiteCloud IDE plug-in for Eclipse default preferences:

1. In Eclipse, select Window > Preferences.
2. Navigate to the page for which you want to restore default settings, and click Restore Defaults.
3. Click Apply to save and leave the Preferences window open, or click OK to save and close the window.

i Note: Validation is not provided for SuiteScript 2.0 code.
SuiteCloud IDE Plug-in for Eclipse Usage

Before you start working with the IDE plug-in, familiarize yourself with the following:

- NetSuite Perspective in SuiteCloud IDE Plug-in for Eclipse
- SuiteCloud IDE Plug-in for Eclipse Tips and Guidelines
- Automatic Code Completion in SuiteCloud IDE Plug-in for Eclipse

Working with the IDE plug-in is divided into the following task areas:

- Working with SDF Projects in SuiteCloud IDE Plug-in for Eclipse
- Working with SuiteScript Projects in SuiteCloud IDE Plug-in for Eclipse
- Working with SuiteScript Files in SuiteCloud IDE Plug-in for Eclipse
- Working with SSP Application Projects in SuiteCloud IDE Plug-in for Eclipse
- Changing Project Settings in SuiteCloud IDE Plug-in for Eclipse
- Converting a Project to a SuiteScript Project in SuiteCloud IDE Plug-in for Eclipse
- Converting a Project to an SSP Application Project in SuiteCloud IDE Plug-in for Eclipse
- Logging in to a Project Account from SuiteCloud IDE Plug-in for Eclipse
- SuiteCloud IDE Plug-in for Eclipse Debugger
- Launching the SuiteScript Records Browser from SuiteCloud IDE Plug-in for Eclipse
- Viewing SuiteCloud IDE Plug-in for Eclipse Error Logs

Information about using the IDE plug-in for account customization projects is included in the documentation for SuiteCloud Development Framework (SDF). See the help topic SuiteCloud Development Framework.

NetSuite Perspective in SuiteCloud IDE Plug-in for Eclipse

The NetSuite Perspective is an added perspective in the Eclipse workbench for the IDE plug-in. The IDE plug-in provides functionality for managing and working with SuiteCloud projects and resources, accessible through menus and toolbars.

The NetSuite perspective consists of an editor area and one or more views that you use as you work with your SuiteCloud projects.
The following table lists the elements of the IDE plug-in interface:

<table>
<thead>
<tr>
<th>Element</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>The NetSuite Explorer view, referred to as the NS Explorer view, displays the hierarchical view of your SuiteCloud projects (their folders and files) and resources. It works similar to the Project Explorer view of Eclipse.</td>
</tr>
</tbody>
</table>

**Note:** SSP applications currently are not supported in SuiteScript 2.0.

In NS Explorer view, you can hover over the project folders to see the following details when available:
- Project Type
- NetSuite Account ID
- Company
- Email
- Environment
- File Cabinet Folder
- Sync Custom Field and Record ID(s) from Account
- SuiteScript Version
- Project Folder Name
- Available for SuiteBundles

For more information, see the Project Explorer view topic in the Eclipse help.
<table>
<thead>
<tr>
<th>Element</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>Outline View</td>
</tr>
<tr>
<td>3</td>
<td>Editor Area</td>
</tr>
<tr>
<td>4</td>
<td>Problems View</td>
</tr>
<tr>
<td>5</td>
<td>Documentation View</td>
</tr>
<tr>
<td>6</td>
<td>Progress View</td>
</tr>
</tbody>
</table>

**SuiteCloud IDE Plug-in for Eclipse Tips and Guidelines**

The following table lists tips and guidelines for working with projects in the IDE plug-in and the applicable version of SuiteScript.

<table>
<thead>
<tr>
<th>Topic</th>
<th>SuiteScript Version</th>
<th>For more information, see ...</th>
</tr>
</thead>
<tbody>
<tr>
<td>Using keyboard shortcuts when you work with files in the Eclipse editor</td>
<td>1.0</td>
<td>2.0</td>
</tr>
<tr>
<td>Adding event handlers to SuiteScript 1.0 files</td>
<td>1.0</td>
<td>Shorthand for Adding Event Handlers to Code in SuiteCloud IDE Plug-in for Eclipse</td>
</tr>
<tr>
<td>Viewing validation markers in the Eclipse editor</td>
<td>1.0</td>
<td>2.0</td>
</tr>
<tr>
<td>Get general guidelines for using the IDE plug-in</td>
<td>1.0</td>
<td>2.0</td>
</tr>
</tbody>
</table>

**Shorthand for Adding Event Handlers to Code in SuiteCloud IDE Plug-in for Eclipse**

**Applicable SuiteScript versions:**

SuiteScript 1.0

You can use shorthand to add new, empty event handler methods to an existing code file. The shorthand also adds the necessary comments to the existing code file.
For example, the following figures shows that if you type `. CPI` and press `Ctrl+Spacebar`, the following code is added to a Client script:

```javascript
/**
 * The recordType (internal id) corresponds to the "Applied To" record in your script deployment.
 * @appliedtorecord recordType
 *
 * @param {String} type Access mode: create, copy, edit
 * @returns {Void}
 */
function clientPageInit(type){
}
```

The following table lists the shorthand for event handler methods in SuiteScript 1.0 files.

<table>
<thead>
<tr>
<th>Script Type / Event Type</th>
<th>Shorthand Input</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Bundle Installation script</strong></td>
<td></td>
</tr>
<tr>
<td>Before Install</td>
<td>.bibi</td>
</tr>
<tr>
<td>After Install</td>
<td>.biai</td>
</tr>
<tr>
<td>Before Update</td>
<td>.bibu</td>
</tr>
<tr>
<td>Script Type / Event Type</td>
<td>Shorthand Input</td>
</tr>
<tr>
<td>--------------------------</td>
<td>----------------</td>
</tr>
<tr>
<td>After Update</td>
<td>.biau</td>
</tr>
<tr>
<td>Client script</td>
<td></td>
</tr>
<tr>
<td>Page Init</td>
<td>.cpi</td>
</tr>
<tr>
<td>Save Record</td>
<td>.csr</td>
</tr>
<tr>
<td>Validate Field</td>
<td>.cvf</td>
</tr>
<tr>
<td>Field Changed</td>
<td>.cfc</td>
</tr>
<tr>
<td>Post Sourcing</td>
<td>.cps</td>
</tr>
<tr>
<td>Line Init</td>
<td>.cli</td>
</tr>
<tr>
<td>Validate Line</td>
<td>.cvl</td>
</tr>
<tr>
<td>Recalc</td>
<td>.cr</td>
</tr>
<tr>
<td>Validate Insert</td>
<td>.cvi</td>
</tr>
<tr>
<td>Validate Delete</td>
<td>.cvd</td>
</tr>
<tr>
<td>RESTlet script</td>
<td>.rg</td>
</tr>
<tr>
<td>GET</td>
<td>.rg</td>
</tr>
<tr>
<td>POST</td>
<td>.rp</td>
</tr>
<tr>
<td>DELETE</td>
<td>.rd</td>
</tr>
<tr>
<td>PUT</td>
<td>.rpu</td>
</tr>
<tr>
<td>User Event script</td>
<td>.uebl</td>
</tr>
<tr>
<td>Before Load</td>
<td>.uebs</td>
</tr>
<tr>
<td>After Submit</td>
<td>.ueas</td>
</tr>
</tbody>
</table>

Validation Markers in SuiteCloud IDE Plug-in for Eclipse

**Applicable SuiteScript versions:**

SuiteScript 1.0

The table below shows the Editor area validation markers that were augmented for the Eclipse plug-in.

<table>
<thead>
<tr>
<th>Validation Marker</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Warning</td>
<td>Augmented to indicate the source location of NetSuite-related code completion warnings.</td>
</tr>
<tr>
<td>Error</td>
<td>Augmented to indicate the source location of NetSuite-related syntax or compilation errors.</td>
</tr>
</tbody>
</table>

For more information, see the Markers topic in the Eclipse help.
# SuiteCloud IDE Plug-in for Eclipse Guidelines

**Applicable SuiteScript versions:**
- SuiteScript 1.0
- SuiteScript 2.0

The following table lists general guidelines for working with the Eclipse plug-in.

<table>
<thead>
<tr>
<th>Guideline</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Use top-down coding method.</td>
<td>Make sure to write your code from top to bottom. The code preceding the cursor should always be completely valid or syntactically correct. For code completion to work, there should be no undeclared variables.</td>
</tr>
<tr>
<td>Use camel case filters.</td>
<td>You can use filters in camel case format to maximize code completion. For example, to display <code>nlapiLoadRecord</code> as an option, type <code>nLR</code> followed by the code completion shortcut, <code>Ctrl+Spacebar</code>. Go to <code>Window &gt; Preferences &gt; JavaScript &gt; Editor &gt; Content Assist</code> and ensure that the <code>Show camel case matches</code> preference is enabled. Note: You can use camel case filters with SuiteScript 1.0 only.</td>
</tr>
<tr>
<td>Pass variables to functions.</td>
<td>When variables (simple variables, not object properties) are passed to a SuiteScript function, code completion works. This approach is applicable to all parameters that have code completion.</td>
</tr>
</tbody>
</table>
| Use prefix filters.               | When working with multiple records, enter `Ctrl+Spacebar` and then use prefix filters to specify the record context you need for code completion. The prefix filter shortcuts are:  
  - `<x>`+Spacebar  
  - `<x>`+Spacebar+`<y>`  
  - `<x>`+Spacebar+`<y>`+Spacebar+`<z>`  
  For more information about prefix filter shortcuts, see SuiteCloud IDE Plug-in for Eclipse Shortcuts. Note: You can use prefix filters with SuiteScript 1.0 only. |
| Use standard JSDoc.               | Based on JavaDoc, you can use the standard format for the following JSDoc comment/tags for a function:  
  - `@param`: function parameter  
  - `@returns`: return value of a function  
  - `@type`: variable declarations  
  Type `/**` and press `Enter` to enter a standard JSDoc comment. |
| Activate automatic code completion. | By default, activate code completion in the Eclipse plug-in with the shortcut `Ctrl+Spacebar`. However, you can automatically trigger code completion. To enable auto activation in the Eclipse plug-in:  
  1. Go to `Window > Preferences > JavaScript > Editor > Content Assist`.  
  2. Check the `Enable auto activation` box.  
  3. For `Auto activation delay`, enter `0`. |
SuiteCloud IDE Plug-in for Eclipse Tips and Guidelines

Guideline | Description
--- | ---
4. | For Auto action triggers for JavaScript, enter `.info`.

Close unused projects to increase performance. | For optimum performance, make sure that the only open project in your IDE plug-in is the project you are currently working on. Always close unused projects. For more information about closing a project, see the Closing projects topic in the Eclipse help.

Use unique function and parameter names. | Use unique names for global functions, function names, and parameter names within a single object literal.

Override default memory allocation to increase available memory. | You can increase the memory available for the Eclipse plug-in to allow a maximum amount instead of the default allocation. However, the maximum heap memory depends on whether you have the 32-bit or 64-bit version of Eclipse, operating system, and Java virtual machine (JVM).

Specify your preferred JVM. | You can specify your preferred JVM if multiple copies are available.

SuiteCloud IDE Plug-in for Eclipse Shortcuts

See the following topics for information about shortcuts for the Eclipse plug-in:
- Code Completion Shortcuts in SuiteCloud IDE Plug-in for Eclipse
- NetSuite Command Shortcuts in SuiteCloud IDE Plug-in for Eclipse

Code Completion Shortcuts in SuiteCloud IDE Plug-in for Eclipse

**Applicable SuiteScript versions:**
- SuiteScript 1.0
- SuiteScript 2.0

The following table shows the different keyboard shortcuts for code completion you can use within the IDE plug-in NetSuite Perspective in SuiteCloud IDE Plug-in for Eclipse, depending on the version of SuiteScript for the project. For shortcuts you can use within the IDE plug-in Debug Perspective in SuiteCloud IDE Plug-in for Eclipse, see SuiteCloud IDE Plug-in for Eclipse Debugger.

To learn more about NetSuite shortcut keys, see NetSuite Command Shortcuts in SuiteCloud IDE Plug-in for Eclipse.

<table>
<thead>
<tr>
<th>Shortcut</th>
<th>SuiteScript Version</th>
<th>Description</th>
</tr>
</thead>
</table>
| Ctrl+Spacebar | 1.0 | This shortcut is used to open a popup listing of all possible code completion options available in a particular context.
- When multiple record types are involved, you can precede this shortcut with the shortcut `<x>+Spacebar +<y>` to specify the record context you need.
- When multiple sublists of a record are involved, you can precede this shortcut with the shortcut `<x>+Spacebar +<y>+Spacebar+<z>` to specify the record-sublist context you need. |
<table>
<thead>
<tr>
<th>Shortcut</th>
<th>SuiteScript Version</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>&lt;x&gt;+Spacebar</code></td>
<td>1.0</td>
<td>Use this shortcut to get all possible code completion options for a record where <code>&lt;x&gt;</code> is the filter for record types. For example, type <code>s+Spacebar</code> to get all possible code completion options for salesorder fields. This shortcut is used in conjunction with the shortcut Ctrl +Spacebar. For more information, see Automatic Code Completion in SuiteCloud IDE Plug-in for Eclipse.</td>
</tr>
<tr>
<td><code>&lt;x&gt;+Spacebar+&lt;y&gt;</code></td>
<td>1.0</td>
<td>Use this shortcut to get all possible code completion options for a field in a particular record context where <code>&lt;x&gt;</code> is the filter for record types and <code>&lt;y&gt;</code> is the filter for fields of a record type. For example, type <code>s+Spacebar+a</code> to get all possible code completion options for salesorder fields starting with the letter a. Note: For internal ID code completion options, append an exclamation point (!) to override and ignore the context when &quot;No Proposal&quot; appears for a particular prefix filter. This shortcut is used in conjunction with the shortcut Ctrl +Spacebar. For more information, see Automatic Code Completion in SuiteCloud IDE Plug-in for Eclipse.</td>
</tr>
<tr>
<td><code>&lt;x&gt;+Spacebar+&lt;y&gt;+Spacebar+&lt;z&gt;</code></td>
<td>1.0</td>
<td>Use this shortcut to get all possible code completion options for a sublist field in a particular record-sublist context where <code>&lt;x&gt;</code> is the filter for record types, <code>&lt;y&gt;</code> is the filter for sublists of a record type, and <code>&lt;z&gt;</code> is the filter for fields of a sublist. For example, type <code>s+Spacebar+i+Spacebar+a</code> to get all possible code completion options for salesorder-item sublist fields starting with the letter a. This shortcut is used in conjunction with the shortcut Ctrl +Spacebar. For more information, see Automatic Code Completion in SuiteCloud IDE Plug-in for Eclipse.</td>
</tr>
<tr>
<td><code>&lt;xYZ&gt;</code></td>
<td>1.0</td>
<td>This is an existing Eclipse shortcut. Use this shortcut to get code completion options using camel case patterns for filters where xYZ is your camel case filter for methods and variables. For example, type <code>nLR</code> to get code completion options for methods using the matching camel case pattern, such as nlapiLoadRecord. This shortcut is used in conjunction with the shortcut Ctrl +Spacebar.</td>
</tr>
</tbody>
</table>
## SuiteCloud IDE Plug-in for Eclipse Shortcuts

<table>
<thead>
<tr>
<th>Shortcut</th>
<th>SuiteScript Version</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tab</td>
<td>1.0</td>
<td>2.0</td>
</tr>
<tr>
<td>Esc</td>
<td>1.0</td>
<td>2.0</td>
</tr>
<tr>
<td>Type /** and press Enter</td>
<td>1.0</td>
<td>2.0</td>
</tr>
<tr>
<td>Alt+Up Arrow or Alt+Down Arrow</td>
<td>1.0</td>
<td>2.0</td>
</tr>
<tr>
<td>Ctrl+Alt+Up Arrow or Ctrl+Alt+Down Arrow</td>
<td>1.0</td>
<td>2.0</td>
</tr>
<tr>
<td>Alt+Shift+A</td>
<td>1.0</td>
<td>2.0</td>
</tr>
</tbody>
</table>

### Automatic Code Completion in SuiteCloud IDE Plug-in for Eclipse

Code completion works by examining the source code around the cursor. When you press the code completion shortcut, **Ctrl+Spacebar**, the system displays a popup with the appropriate text options to insert at the cursor. This popup may include record types, field names, or other information, depending on where the caret is positioned.

Record types supported for auto completion in the IDE plug-in are those that are supported in both SuiteScript and SOAP web services. For a list of supported records, see the help topic [SuiteCloud Supported Records](#).

To learn the basics of code completion hands-on, download the DemoProject files included with the IDE plug-in [here](#).

⚠️ **Important:** After you download the DemoProject files, add the files to a SuiteScript project in your IDE plug-in. For more information, see the Importing files topic in the Eclipse help.

If you have no existing SuiteScript projects in your IDE plug-in, you must create one before you add the DemoProject files. For more information, see [Creating a SuiteScript Project with SuiteCloud IDE Plug-in for Eclipse](#).

**To activate automatic code completion in IDE plug-in for Eclipse:**

1. In Eclipse, select **Window > Preferences**.
2. In the **Preferences** window, select **JavaScript > Editor > Content Assist**.
3. In the **Auto-Activation** area, select the **Enable auto activation** box, and make sure that the following fields are set accordingly:

![Preference window with Content Assist settings](image)

**NetSuite Command Shortcuts in SuiteCloud IDE Plug-in for Eclipse**

The following table shows the main keyboard shortcuts for commands you can use:

<table>
<thead>
<tr>
<th>Shortcut</th>
<th>SuiteScript Version</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ctrl+Alt+A</td>
<td>1.0</td>
<td>2.0</td>
</tr>
<tr>
<td>Ctrl+U</td>
<td>1.0</td>
<td>2.0</td>
</tr>
<tr>
<td>Ctrl+B</td>
<td>1.0</td>
<td>2.0</td>
</tr>
<tr>
<td>Esc</td>
<td>1.0</td>
<td>2.0</td>
</tr>
</tbody>
</table>

**Note:** To configure more shortcut keys, see [Configuring NetSuite Shortcut Keys](#).
<table>
<thead>
<tr>
<th>Shortcut</th>
<th>SuiteScript Version</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ctrl+Click &lt;internal ID&gt;</td>
<td>1.0</td>
<td>Use this shortcut to launch the Record Browser specific to the internal ID from the editor.</td>
</tr>
<tr>
<td>Ctrl+Alt+Shift+E</td>
<td>1.0</td>
<td>2.0</td>
</tr>
<tr>
<td>Ctrl+Alt+I</td>
<td>1.0</td>
<td>2.0</td>
</tr>
<tr>
<td>Ctrl+Alt+Shift+U</td>
<td>1.0</td>
<td>2.0</td>
</tr>
<tr>
<td>Ctrl+Alt+R</td>
<td>1.0</td>
<td>2.0</td>
</tr>
<tr>
<td>Ctrl+Alt+V</td>
<td>1.0</td>
<td>2.0</td>
</tr>
<tr>
<td>Ctrl+Alt+D</td>
<td>1.0</td>
<td>2.0</td>
</tr>
<tr>
<td>Ctrl+Alt+Shift+D</td>
<td>1.0</td>
<td>2.0</td>
</tr>
</tbody>
</table>

**Configuring NetSuite Shortcut Keys**

Apart from the ones that come by default with the IDE plug-in, you can configure more shortcut keys.

**To configure NetSuite shortcut keys:**

1. Open Eclipse, and create or select the SDF project.
   
   For more information, see Creating an Account Customization Project in SuiteCloud IDE Plug-in for Eclipse or Creating a SuiteApp Project in SuiteCloud IDE Plug-in for Eclipse.

2. Select **Windows > Preferences > General > Keys**.

3. In the filter text field, type **NetSuite**.
   
   A list appears that shows all the IDE plug-in commands that can be bound or unbound to shortcut keys.

4. For more information about configuring shortcut keys, refer to the Eclipse documentation.

For more information about shortcut keys in the IDE plug-in, also see SuiteCloud IDE Plug-in for Eclipse Shortcuts.

**Working with SDF Projects in SuiteCloud IDE Plug-in for Eclipse**

See the following topics for information about working with SDF projects:

- Creating an Account Customization Project in SuiteCloud IDE Plug-in for Eclipse
- Creating a SuiteApp Project in SuiteCloud IDE Plug-in for Eclipse
- Importing Account Components into an SDF Project with SuiteCloud IDE Plug-in for Eclipse
- Defining Dependencies on SuiteApp and Bundle Objects using the Eclipse Plug-in
Creating an Account Customization Project in SuiteCloud IDE Plug-in for Eclipse

You can use SDF to create account customization projects, which are intended to customize your account. For more information about account customization projects, see the following topics:

- Account Customization Projects
- Differences Between SDF Projects

To create an account customization project in SuiteCloud IDE Plug-in for Eclipse:

2. Under SDF Project Type, choose Account Customization and click Next.
3. Set the following properties. For more information, see the help topic Properties of an Account Customization Project.
4. Click Finish.
   The IDE plug-in creates a new SDF project and displays the manifest for that project.

   **Note:** If you do not see the manifest or the project, make sure that you are in the Eclipse Workbench and that the NetSuite perspective is open.

5. In the NS Explorer subtab, expand the project to see the project structure. For more information about the IDE plug-in UI, see NetSuite Perspective in SuiteCloud IDE Plug-in for Eclipse.

Creating a SuiteApp Project in SuiteCloud IDE Plug-in for Eclipse

You can use SDF to create SuiteApp projects, which are intended for SuiteCloud Developer Network (SDN) members creating SDF SuiteApps. For more information about SuiteApp projects, see the following topics:

- SuiteApp Projects
- Differences Between SDF Projects

To create a SuiteApp project in SuiteCloud IDE Plug-in for Eclipse:

2. Under SDF Project Type, choose **SuiteApp** and then click **Next**.
3. Set the project properties. For more information, see the help topic **Properties of a SuiteApp Project**.
4. Click **Finish**.
   The IDE plug-in creates a new SDF project and displays the manifest for that project.

   **Note:** If you do not see the manifest or the project, make sure that you are in the Eclipse Workbench and that the NetSuite perspective is open.

5. In the **NS Explorer** subtab, expand the project to see the project structure. For more information about the IDE plug-in UI, see **NetSuite Perspective in SuiteCloud IDE Plug-in for Eclipse**.

---

**Importing Account Components into an SDF Project with SuiteCloud IDE Plug-in for Eclipse**

You can use SDF to import custom NetSuite objects, files, and scripts from your account into an account customization project. You can import custom NetSuite objects to an account customization project or a SuiteApp project.

See the following topics:

- **Importing Custom NetSuite Objects into an SDF Project with SuiteCloud IDE Plug-in for Eclipse**
- **Importing Files and Scripts into an Account Customization Project in SuiteCloud IDE Plug-in for Eclipse**
- **Account Component Imports**

**Importing Custom NetSuite Objects into an SDF Project with SuiteCloud IDE Plug-in for Eclipse**

1. Open Eclipse, and create or select the SDF project.
   For more information, see **Creating an Account Customization Project in SuiteCloud IDE Plug-in for Eclipse** or **Creating a SuiteApp Project in SuiteCloud IDE Plug-in for Eclipse**.
2. Expand your project in the **NS Explorer** subtab, and right-click the **Objects** folder.
3. Go to **NetSuite > Import Custom Objects from Account**.
4. Choose the account and role associated with your account, and click **Next**.
5. Select or enter the appropriate search criteria to find the custom objects that you want to import, and click **Search**.
   SDF generates a list of custom objects that match your search criteria.
6. Select the custom objects that you want to import, and click **Finish**.
   SDF imports the custom objects from your account, and places them in the **Objects** folder of the SDF project.

**Importing Files and Scripts into an Account Customization Project in SuiteCloud IDE Plug-in for Eclipse**

If your account has the Web Site Hosting Files folder, you need to fulfill the following prerequisites to see it in the IDE plug-in:
The Host HTML Files feature enabled.
The Website (External) publisher permission.

1. Open Eclipse, and create or select the account customization project.
   For more information, see Creating an Account Customization Project in SuiteCloud IDE Plug-in for Eclipse.
2. Expand your project in the NS Explorer subtab, and right-click the FileCabinet folder.
3. Go to NetSuite > Import Files from Account.
4. Choose the account and role associated with your account, and then click Get File List.
5. Select the files or folders that you want to import, and click OK.

   Note: You may be prompted to overwrite any files that already exist in the project.

SDF imports the files from the account and places them in subdirectories of the FileCabinet folder of the account customization project. For the list of subdirectories, see the File Cabinet Files section in SDF Project Components.

Working with Custom NetSuite Objects in SuiteCloud IDE Plug-in for Eclipse

Custom NetSuite objects are created when XML definitions in an SDF project are deployed to your account.

See the following topics to learn more about working with custom objects using SDF:

- Creating an XML Definition in SuiteCloud IDE Plug-in for Eclipse
- Downloading the XML Definition of an Object from Your NetSuite Account
- Comparing a Custom SuiteCloud Object with a Custom NetSuite Object in SuiteCloud IDE Plug-in for Eclipse
- Custom NetSuite Object Management

Creating an XML Definition in SuiteCloud IDE Plug-in for Eclipse

You create custom NetSuite objects by creating XML definitions of those objects. The XML files are converted into custom NetSuite objects when you deploy them to your NetSuite account.

To create an XML definition:

1. Open Eclipse, and create or select the SDF project.
   For more information, see Creating an Account Customization Project in SuiteCloud IDE Plug-in for Eclipse or Creating a SuiteApp Project in SuiteCloud IDE Plug-in for Eclipse.
2. Go to File > New > Custom Object File.
3. In the Type field, select the custom NetSuite object type that you want to create.
   For more information about supported custom NetSuite object types, see the help topic Customizations Supported by SuiteCloud Development Framework.
4. Select the Objects folder in the SDF project.
Custom NetSuite object types must be stored in the Objects folder. By default, your deploy file is configured to deploy all custom NetSuite objects in the Objects folder. For more information, see the help topic SDF Project Deployment Preparation.

5. In the Filename field, specify a file name for the custom NetSuite object. The file name should match the script ID of the object.

6. Click Finish.

The custom NetSuite object is created based on the template for that object. You can customize the template by changing the IDE plug-in preferences. For more information, see Setting SDF Project Preferences in SuiteCloud IDE Plug-in for Eclipse.

Modifying an XML Definition in a SuiteCloud IDE Plug-in for Eclipse SDF Project

You should modify an XML definition based on a predefined custom template, an imported custom object, or an XML definition downloaded from your account. For more information about these topics, see:

- Creating an XML Definition in SuiteCloud IDE Plug-in for Eclipse
- Importing Account Components into an SDF Project with SuiteCloud IDE Plug-in for Eclipse
- Downloading the XML Definition of an Object from Your NetSuite Account

To understand how an XML definition in an SDF project is structured and which XML tags are available within nested elements, use the Content Assist (Ctrl + Spacebar) feature in Eclipse.

For example, the following image shows the Content Assist feature being used inside a role element, a custom object that is supported by SDF. The assistant shows all the XML elements that can be nested inside the element.
To add an element to an XML definition in an SDF project in SuiteCloud IDE Plug-in for Eclipse:

1. Open Eclipse, and create or select the SDF project.
   For more information, see Creating an Account Customization Project in SuiteCloud IDE Plug-in for Eclipse or Creating a SuiteApp Project in SuiteCloud IDE Plug-in for Eclipse.
2. Open the custom object XML definition that you want to modify.
   If your project does not contain any XML definitions, you must create, import, or download one. For more information about these topics, see:
   - Creating an XML Definition in SuiteCloud IDE Plug-in for Eclipse
   - Importing Account Components into an SDF Project with SuiteCloud IDE Plug-in for Eclipse
   - Downloading the XML Definition of an Object from Your NetSuite Account
3. Place your cursor in between the opening and closing tags of an element.
   The location of your cursor determines where you want to nest the new element.
4. Press Ctrl + Spacebar to open the Content Assist.
   Alternatively, go to Edit > Content Assist.
   A list appears that reveals all the elements that can be nested inside the root element.
5. Select the element that you want to use.
   The new element is added to the XML definition. Some elements, such as those that require Boolean values, include default content when they are added.

Comparing a Custom SuiteCloud Object with a Custom NetSuite Object in SuiteCloud IDE Plug-in for Eclipse

You can compare the XML definition of a custom SuiteCloud object with the XML equivalent of a custom NetSuite object. Objects can only be compared when they share the same script IDs and are the same object type.

To compare a custom SuiteCloud object with a custom NetSuite object:

1. Open Eclipse, and create or select the SDF project.
   For more information, see Creating an Account Customization Project in SuiteCloud IDE Plug-in for Eclipse or Creating a SuiteApp Project in SuiteCloud IDE Plug-in for Eclipse.
2. Expand your project in the NS Explorer subtab.
3. Expand the Objects folder, right-click the object, and click NetSuite > Compare Custom Object with Account Version.
   The side-by-side comparison window appears if an object with the same script ID exists in your NetSuite account. The differences between the two objects are highlighted.

Example

The following shows the side-by-side comparison window of a custom user event script object that exists in the SDF project and the NetSuite account:
Overwriting Custom SuiteCloud Objects with Custom NetSuite Objects in SuiteCloud IDE Plug-in for Eclipse

**Note:** Overwriting is only supported for a SuiteCloud object that shares the same script ID and is the same object type as the corresponding object in your NetSuite account.

To overwrite custom SuiteCloud objects with custom NetSuite objects:

1. Open Eclipse, and create or select the SDF project.
   For more information, see Creating an Account Customization Project in SuiteCloud IDE Plug-in for Eclipse or Creating a SuiteApp Project in SuiteCloud IDE Plug-in for Eclipse.
2. Expand your project in the **NS Explorer** subtab.
3. Select the most appropriate option to overwrite your custom objects.
   - To overwrite specific custom SuiteCloud objects, expand the **Objects** folder, right-click each object and click **NetSuite > Update Custom Object from Account**.
   - To overwrite all custom SuiteCloud objects, right-click the **Objects** folder and click **NetSuite > Overwrite Custom Objects with Account Version**.
4. Choose the account and role associated with your account, and click **OK**.
5. Click **OK** to confirm that you want to perform the update.

SDF connects to your NetSuite account and imports custom objects that share the same script IDs as the custom objects selected in your SDF project. The selected custom SuiteCloud objects are overwritten by the custom NetSuite objects.

Uploading Files or Folders to Your Account with SuiteCloud IDE Plug-in for Eclipse

You can upload individual files or folders to the file cabinet in your account, without deploying the entire project. You can also select several files and folders and upload them at once. The files or folders must
be within a File Cabinet folder supported by SDF. The following folders are supported, depending on the SDF project type:

- SuiteScripts
- SuiteApps
- Templates
- Web Site Hosting Files

**To upload files or folders to your account with SuiteCloud IDE plug-in for Eclipse:**

1. From the File Cabinet folder of your SDF project, do one of the following:
   - Right-click a file and select Netsuite > Upload File to Account.
   - Right-click a folder and select Netsuite > Upload Folder to Account.

**Important:** If a file or folder with the same location and name already exists in your account, it is overwritten by the SDF upload.

If the folder containing the file did not exist in the account, SDF creates the folder in your account during the upload process.

**Defining Dependencies on SuiteApp and Bundle Objects using the Eclipse Plug-in**

SDF supports external dependences on custom objects using references. The object must be an object type supported by SDF, and the referenced object and application it comes from must be installed to your target account.

To depend on an external object, you can add a reference to the object in your project manifest file and add a reference in the object XML definition where you want to use the object.

References need to define the script ID and either the application or bundle that the object comes from. If you are depending on a bundle object shared in many bundles, you can reference multiple bundle IDs in a delimited list.

During server-side validation against a NetSuite account, SDF logs any references object that is missing from the account.

**Important:** SuiteApp projects support dependences on bundle and SuiteApp objects. Account customization projects support references to SDF SuiteApp objects (only objects with an application ID).

For more information about custom object dependencies, see the following topics:

- SDF SuiteApp Object Dependencies
- Specifying a NetSuite Custom Object Reference
- Manually Defining SDF SuiteApp Object Dependencies from the SDF Project Manifest
- Specifying an SDF SuiteApp Object Reference
- Specifying a Bundle Object Reference

**To add SuiteApp object dependencies to the project manifest using SuiteCloud IDE Plug-in for Eclipse:**

To complete this task, your project must already reference the SuiteApp object from an object definition in the Object folder. If your project does not contain references to the object, you can define the reference manually instead. See the help topics Manually Defining SDF SuiteApp Object Dependencies.
from the SDF Project Manifest and Manually Defining SDF SuiteApp Object Dependencies from the SDF Project Manifest.

1. Expand your project in the NS Explorer subtab, and right-click the manifest.xml file.
2. Select NetSuite > Add Dependency References to Manifest.

SDF evaluates the custom NetSuite objects that are referenced in the SDF project and adds any new dependencies to the manifest.xml file.

**Note:** This option does not automatically remove dependencies that are not referenced by any objects.

---

**Example**

The following example shows a SuiteApp project manifest that uses references to depend on the following objects:

- customrecord_sample object from com.samples.mysuiteapp
- customrecord_sample2 object from bundle 351
- customrecord_sample3 object which may be found in a few bundles due to a bundle copying or deprecation

```xml
<manifest projecttype="SUITEAPP">
  <publisherid>com.example</publisherid>
  <projectid>mysuiteapp</projectid>
  <projectname>SDFSample</projectname>
  <projectversion>1.0.0</projectversion>
  <frameworkversion>2.0</frameworkversion>
  <dependencies>
    <applications>
      <application id="com.example.mysuiteapp">
        <objects>
          <object>customrecord_sample</object>
        </objects>
      </application>
      <bundles>
        <bundle id="351">
          <object>customrecord_sample2</object>
        </bundle>
        <bundle id="451|452|453">
          <object>customrecord_sample3</object>
        </bundle>
      </bundles>
    </applications>
  </dependencies>
</manifest>
```

---

**Defining Account Component Dependencies from an Account Customization Project in SuiteCloud IDE Plug-in for Eclipse**

You can define dependencies on existing custom NetSuite objects, files, and scripts from your NetSuite accounts in an account customization project. For more information, see the following topics:
Account Component Dependencies

Specifying a NetSuite Custom Object Reference

If your project does not contain any references to the dependency, you must define it manually. See the help topic Manually Defining Account Component Dependencies in the Account Customization Project Manifest.

To automatically define account component dependencies in SuiteCloud IDE Plug-in for Eclipse:

1. Open Eclipse and select the account customization project.
   For more information, see Creating an Account Customization Project in SuiteCloud IDE Plug-in for Eclipse.
2. Expand your project in the NS Explorer subtab, and right-click the manifest.xml file.
   SDF evaluates the custom NetSuite objects, files, and scripts that are referenced in an account outside of the SDF project and adds any new dependencies to the manifest.xml file.

**Note:** This option does not automatically remove dependencies that are not referenced by any objects.

Example

The following example shows an account customization project manifest that references objects and files that are not part of the project, such as customrecord_sample, customworkflow_sample, customlist, and ext_UserScript.js. During a server-side validation against a NetSuite account, SDF logs any referenced object or file that is missing from the account.

```
<manifest projecttype="ACCOUNTCUSTOMIZATION">
  <projectname>SDFTutorial</projectname>
  <frameworkversion>1.0</frameworkversion>
  <dependencies>
    <features>
      <feature required="true">CUSTOMRECORDS</feature>
      <feature required="true">SERVERSIDESCRIPTING</feature>
    </features>
    <objects>
      <object>customrecord_sample</object>
      <object>customworkflow_sample</object>
      <object>customworkflow_sample.workflowstate_a</object>
      <object>customlist_sample.red</object>
      <object>customlist_sample.green</object>
      <object>customlist_sample.blue</object>
    </objects>
    <files>
      <file>/SuiteScript/reference/ext_UserScript.js</file>
    </files>
  </dependencies>
</manifest>
```
Defining Feature Dependencies in SuiteCloud IDE Plug-in for Eclipse

When you use SDF with NetSuite features, such as SuiteScript or custom records, you must define the dependencies in the SDF project.

For more information, see the help topic Feature Dependencies.

To automatically reference dependencies in the SDF project manifest:

1. Open Eclipse, and create or select the SDF project.
   For more information, see Creating an Account Customization Project in SuiteCloud IDE Plug-in for Eclipse or Creating a SuiteApp Project in SuiteCloud IDE Plug-in for Eclipse.
2. Expand your project in the NS Explorer subtab, and right-click the manifest.xml file.
   SDF evaluates the NetSuite feature requirements in the project and adds any new dependency references to the manifest.xml file.
   
   **Note:** This option does not automatically remove dependency references that are not referenced by any objects.

4. Review the new dependency references. If desired, change the required attribute to suit your business needs.

Example

The following example shows a project manifest that references dependencies on the server-side SuiteScript and custom record features. The references enable you to use these features in the project.

```xml
<manifest projecttype="ACCOUNTCUSTOMIZATION">
  <projectname>SDFTutorial</projectname>
  <frameworkversion>1.0</frameworkversion>
  <dependencies>
    <features>
      <feature required="true">CUSTOMRECORDS</feature>
      <feature required="true">SERVERSIDESCRIPTING</feature>
    </features>
  </dependencies>
</manifest>
```

Configuring Account Features in an Account Customization Project in SuiteCloud IDE Plug-in for Eclipse

Certain account features can be enabled or disabled from an account customization project.

For more information, see the following procedure and Account Configuration from an Account Customization Project.

To import an account configuration from your NetSuite account in SuiteCloud IDE Plug-in for Eclipse:

1. Open Eclipse, and create or select the account configuration project.
For more information, see Creating an Account Customization Project in SuiteCloud IDE Plug-in for Eclipse.

2. Expand your project in the NS Explorer subtab, and right-click the AccountConfiguration folder.


4. Choose the account and role associated with your account, and click Get Configuration List.

The Enable Features option appears in the Configuration list.

5. Check the Enable Features option and click OK.

The features.xml file is created in the AccountConfiguration folder. This file contains a comprehensive list of features that are currently enabled and disabled in the target account.

6. Confirm that the deploy.xml file references the path to the AccountConfiguration folder.

In the NS Explorer subtab, open the deploy.xml file.

Confirm that the deploy file contains a <configuration> element. This element should contain a <path> element with a value that is set to the path of the features.xml file.

For example, the following is a valid deploy.xml file that contains an account configuration:

```xml
<deploy>
  <configuration>
    <path>~/AccountConfiguration/*</path>
  </configuration>
  <files>
    <path>~/FileCabinet/SuiteScripts/*</path>
  </files>
  <objects>
    <path>~/Objects/customrecord_employee.xml</path>
    <path>~/Objects/customrecord_company.xml</path>
  </objects>
</deploy>
```

7. Validate the project frequently as you make changes to the features.xml file.

Due to the complexity of account feature dependencies on other account features, you may encounter several validation errors when enabling or disabling features. For more information, see the help topic Account Configuration Validation.

To manually create an account configuration in SuiteCloud IDE Plug-in for Eclipse:

1. Open Eclipse, and create or select the account configuration project.

For more information, see Creating an Account Customization Project in SuiteCloud IDE Plug-in for Eclipse.

2. Expand your project in the NS Explorer subtab, and right-click the AccountConfiguration folder.


4. Select AccountConfiguration as the parent folder, and enter the file name: features.xml

5. Click Finish.

The features.xml file is created in the AccountConfiguration folder. In this file, you can list all the features that you want to enable or disable in the account.

6. In the features.xml file, set the root element to <features>. Your file should look like the following:

```xml
<features>
</features>
```
Working with SDF Projects in SuiteCloud IDE Plug-in for Eclipse

7. Inside the <features> element, create a <feature> element for each feature that you want to configure. Each <feature> element requires an <ID> element and a <status> element. For example, if you want to enable the DEPARTMENTS and LOCATIONS fields, your features.xml file should look like the following:

```xml
<features>
  <feature>
    <id>DEPARTMENTS</id>
    <status>ENABLED</status>
  </feature>
  <feature>
    <id>LOCATIONS</id>
    <status>ENABLED</status>
  </feature>
</features>
```

The <ID> value can be set to any valid feature field ID. You can see all feature field IDs by navigating to Setup > Company > Enable Features in NetSuite. You can also use the content assist feature in Eclipse to determine which feature IDs can be specified. For more information, see Modifying an XML Definition in a SuiteCloud IDE Plug-in for Eclipse SDF Project.

The <status> value should always be set to ENABLED or DISABLED, depending on your preference.

8. Confirm that the deploy.xml file references the path to the AccountConfiguration folder.

   In the NS Explorer substab, open the deploy.xml file.

   Confirm that the deploy file contains a <configuration> element. This element should contain a <path> element with a value that is set to the path of the features.xml file.

   For example, the following is a valid deploy.xml file that contains an account configuration:

   ```xml
   <deploy>
     <configuration>
       <path>~/AccountConfiguration/*</path>
     </configuration>
     <files>
       <path>~/FileCabinet/SuiteScripts/*</path>
     </files>
     <objects>
       <path>~/Objects/customrecord_employee.xml</path>
       <path>~/Objects/customrecord_company.xml</path>
     </objects>
   </deploy>
   ```

9. Validate the project frequently as you make changes to the features.xml file.

   Due to the complexity of account feature dependencies on other account features, you may encounter several validation errors when enabling or disabling features. For more information, see the help topic Account Configuration Validation.

Setting Installation Preferences in a SuiteApp Project in SuiteCloud IDE Plug-in for Eclipse

To define installation preferences, you can create new preference files in the InstallationPreferences folder of your SuiteApp project. There are two types of preferences files: locking preferences and hiding preferences.
For more information, see the help topics Installation Preferences in a SuiteApp Project, and the following topics:

- Creating an Installation Preference File in SuiteCloud IDE Plug-in for Eclipse
- Locking Custom Objects, Files and Folders in a SuiteApp Project in SuiteCloud IDE Plug-in for Eclipse
- Hiding Files and Folders in a SuiteApp Project
- Using the Content Protection Option to Apply Installation Preferences
- Default Installation Preferences

Creating an Installation Preference File in SuiteCloud IDE Plug-in for Eclipse

1. In Eclipse, select New > Installation Preference File.
2. Select the Type.
3. Enter or select the InstallationPreferences folder or any of its subfolders.
4. Select Finish.

Locking Custom Objects, Files and Folders in a SuiteApp Project in SuiteCloud IDE Plug-in for Eclipse

To use the locking preference, you must first create a locking.xml file in the InstallationPreferences Folder of a SuiteApp project. See Creating an Installation Preference File in SuiteCloud IDE Plug-in for Eclipse.

From the locking.xml file, you must include a valid preference type, default action, and the object, file or folder path. Supported values for an action are LOCK or UNLOCK.

For more information about installation preferences, see the following topics:

- Locking Custom Objects in an SDF SuiteApp Project
- Installation Preferences in a SuiteApp Project

To lock a custom object:

1. Specify a value for defaultAction.
2. Specify a value for action.
3. Specify the script ID of the object. For the list of lockable object types, see the help topic Lockable Custom Objects Supported by SDF.
4. When you deploy the project, you must select the Apply Content Protection check box to enable the specifications in the locking.xml file. See the help topic Using the Content Protection Option to Apply Installation Preferences.

To lock a file:

1. Specify a value for defaultAction.
2. Specify a value for action.
3. Add the path.
4. Specify the file path.
5. When you deploy the project, you must select the **Apply Content Protection** check box to enable the specifications in the locking.xml file. See the help topic [Using the Content Protection Option to Apply Installation Preferences](#).

**To lock a folder:**

1. Specify a value for **defaultAction**.
2. Specify a value for **action**.
3. Add the **appliesTo** attribute within the **path** tag, and set it to **FILES_AND_FOLDERS**.
   Optionally, add the **includeRoot** attribute and set it to **T** to also lock the root folder.
4. Specify the folder path with a wildcard character at the end.
5. When you deploy the project, you must select the **Apply Content Protection** check box to enable the specifications in the locking.xml file. See the help topic [Using the Content Protection Option to Apply Installation Preferences](#).

Note that setting a value for default action can reduce the steps in applying a preference to many objects at a time. The default action applies to custom objects in your project that are not in the apply action list of your locking.xml file. In the following example, the installation locking preference for lockable custom objects in the SuiteApp project is **UNLOCK**, so all SuiteApp objects other than myobject12 are unlocked.

**Example: Locking a Custom Object using SDF**

For example, to lock the myobject12 object, in the locking.xml file, set the preference type to **LOCKING**, set the default action to **UNLOCK**, set the apply action to **LOCK**, and add the object:

```xml
<preference type="LOCKING" defaultAction="UNLOCK">
  <apply action="LOCK">
    <object>myobject12</object>
  </apply>
</preference>
```

**Example: Locking a File using SDF**

For example, to lock the Suitelet.js file, in the locking.xml file, set the preference type to **LOCKING**, set the default action to **UNLOCK**, set the apply action to **LOCK**, and add the path:

```xml
<preference type="LOCKING" defaultAction="UNLOCK">
  <apply action="LOCK">
    <path>~/FileCabinet/SuiteApps/netsuite.com.test/Suitelet.js</path>
  </apply>
</preference>
```

**Example: Locking a Folder using SDF**

For example, to lock the entire SuiteApps folder, in the locking.xml file, set the preference type to **LOCKING**, set the default action to **UNLOCK**, set the apply action to **LOCK**, and add the path:

```xml
<preference type="LOCKING" defaultAction="UNLOCK">
  <apply action="LOCK">
    <path appliesTo="FILES_AND_FOLDERS" includeRoot="F">~/FileCabinet/SuiteApps/netsuite.com.test/*</path>
  </apply>
</preference>
```
Hiding Files and Folders in a SuiteApp Project

To use the hiding preference, you must first create a hiding preferences file in the InstallationPreferences folder of a SuiteApp project. See Creating an Installation Preference File in SuiteCloud IDE Plug-in for Eclipse.

Supported values for an action or default action are **HIDE** or **UNHIDE**.

For more information, see the following topics:
- Hiding Files in an SDF SuiteApp Project
- Installation Preferences in a SuiteApp Project

**To hide files and folders:**

1. From the hiding.xml file, specify the preference type.
2. Specify the default action.
3. Specify the apply action.
4. Specify the file path.
5. (Optional) To use a wildcard path, specify a file path for the appliesTo attribute. The appliesTo attribute value must be "FILES".
6. When you deploy the project, you must select the Apply Content Protection check box to enable the specifications in the hiding.xml file. See the help topic **Using the Content Protection Option to Apply Installation Preferences**.

**Example: Hiding a file and folder using SDF**

For example, to hide the a.js file, set the preference type to HIDING, set the apply action to HIDE, and add the file paths to the hiding.xml file.

The first file path sets a preference to hide a single file. The second file path applies the preferences to files within the folder specified in the path.

The default action applies to content in your project that is not in the apply list of your hiding.xml file. In the example above, the default preference for files in the SuiteApp project is UNHIDE, so all files except files in the a.js file and files in the myFile folder are not hidden.

```xml
<preference type="HIDING" defaultAction="UNHIDE">
  <apply action="HIDE">
    <path>~/FileCabinet/SuiteApps/com.netsuite.myFile/a.js</path> <!-- hide a file -->
    <path appliesTo="FILES">~/FileCabinet/SuiteApps/com.netsuite.myFiles/*</path> <!-- hide all the files in my application -->
  </apply>
</preference>
```

Managing SDF Projects as Compressed Files in SuiteCloud IDE Plug-in for Eclipse

SDF enables you to save SDF projects as zip files. You can use zip files to back up your projects or create new SDF projects. Eclipse configuration files are excluded from SDF project zip files.
To save an SDF project as a zip file in SuiteCloud IDE Plug-in for Eclipse:

1. Open Eclipse, and create or select the SDF project. For more information, see Creating an Account Customization Project in SuiteCloud IDE Plug-in for Eclipse or Creating a SuiteApp Project in SuiteCloud IDE Plug-in for Eclipse.
2. Right-click on project in the NS Explorer subtab.
4. Choose the path and file name of the zip file, and click Save.

To extract the contents of an SDF project zip file:

1. If the project in the zip file is new or does not have a workspace on your computer, create an SDF project. For more information, see Creating an Account Customization Project in SuiteCloud IDE Plug-in for Eclipse or Creating a SuiteApp Project in SuiteCloud IDE Plug-in for Eclipse.
2. Close Eclipse.
3. Open the project zip file that you created using the IDE plug-in.
4. Extract the contents of the zip file to the desired project workspace.
   
   For example, if you set your Eclipse workspace to C:\eclipse\projects\ and your existing project name is HelloWorld, then extract the zip file to C:\eclipse\projects\HelloWorld to overwrite the project files.

Validating an SDF Project in SuiteCloud IDE Plug-in for Eclipse

SDF can run server-side validation to catch errors that may occur during the project deployment process.

For more information, see the help topic SDF Project Validation.

To validate an SDF project against your NetSuite account in SuiteCloud IDE Plug-in for Eclipse:

1. Open Eclipse, and create or select the SDF project. For more information, see Creating an Account Customization Project in SuiteCloud IDE Plug-in for Eclipse or Creating a SuiteApp Project in SuiteCloud IDE Plug-in for Eclipse.
2. Right-click on project in the NS Explorer subtab.
3. Go to NetSuite > Validate Project Against Account.
4. Choose the account and role associated with your account.
5. Optionally, for an account customization project, change the selection for Action on Account-Specific Values. For more information about this setting, see the help topic Account-Specific Values.
6. Select Validate.

SDF validates your project against the chosen account. The validation progress log is displayed in the Console subtab.
In the IDE plug-in, errors in the validation log are displayed in red. If you encounter errors during server-side validation, review the error messages to determine possible causes and troubleshoot the issues.

Example

The following example shows a validation log that indicates a successful validation:

```
Info -- Account [ACCOUNTNAME]
Info -- Account Customization Project [PROJECTNAME]
Info -- Framework Version [1.0]
Validate manifest -- Success
Validate deploy file -- Success
Validate configuration -- Success
Validate script file -- Success
Validate objects -- Success
Validate files -- Success
Validate folders -- Success
Validate account settings -- Success
Validate Custom Objects against the Account -- Success
```

Deploying an SDF Project to Your NetSuite Account in SuiteCloud IDE Plug-in for Eclipse

When you create an SDF project, a deploy file is automatically generated. By default, the file is configured to deploy all supported files in the FileCabinet/SuiteScripts folder and all custom NetSuite objects in the Objects folder.

For more information, see the help topic SDF Project Deployment Preparation.

To deploy an SDF project to your NetSuite account in SuiteCloud IDE Plug-in for Eclipse:

1. Open Eclipse, and create or select the SDF project.
   For more information, see Creating an Account Customization Project in SuiteCloud IDE Plug-in for Eclipse or Creating a SuiteApp Project in SuiteCloud IDE Plug-in for Eclipse.

   !important: Account customization projects cannot be uninstalled from your account after they have been deployed. Undesired account components need to be manually deleted.

2. Ensure that your project references all the required dependencies.
   For more information, see Defining Feature Dependencies in SuiteCloud IDE Plug-in for Eclipse, Defining Dependencies on SuiteApp and Bundle Objects using the Eclipse Plug-in, and Defining Account Component Dependencies from an Account Customization Project in SuiteCloud IDE Plug-in for Eclipse.

3. Ensure that your deploy file references all of the NetSuite components that you want to deploy.
   For more information, see the help topic SDF Project Deployment Preparation.

4. Right-click on project in the NS Explorer subtab.

5. Go to NetSuite > Deploy or NetSuite > Deploy to Account.
6. If you chose to deploy to an account, select the appropriate account and role, and then click **Deploy**.

7. (Optional) If you are deploying to an account customization project, you can change the selection for **Action on Account-Specific Values**. For more information about this setting, see the help topic **Account-Specific Values**.

8. If you are deploying a SuiteApp project that includes locking.xml and hiding.xml files, select or clear the **Apply Content Protection** option to enable or disable the preferences. For more information, see the help topic **Installation Preferences in a SuiteApp Project**.

9. If you chose to deploy to a NetSuite production account, a notification window appears. Confirm that you want to deploy to the production account to continue the deployment process. SDF validates and deploys the SDF project to your NetSuite account. The installation log is displayed in the **Console** subtab.

**Example**

The following example shows the installation log of a successful deployment to a NetSuite production account for a company named Wolfe Electronics:

```
2016-08-09 06:05:10 (PST) Installation started
Info -- Account [(PRODUCTION) Wolfe Electronics (ACCOUNTNAME)]
Info -- Account Customization Project [SDFtutorial]
Info -- Framework Version [1.0]
Validate manifest -- Success
Validate deploy file -- Success
Validate configuration -- Success
Validate script file -- Success
Validate objects -- Success
Validate files -- Success
Validate folders -- Success
Validate account settings -- Success
Validate Custom Objects against the Account -- Success
Begin deployment
Upload file -- ~/FileCabinet/SuiteScripts/UserEventScript.js
Create object -- customrecord_tutorial (customrecordtype)
Create object -- customscript_userevent (usereventscript)
Create object -- customrecord_tutorial.custrecord_tut_memo (customrecordcustomfield)
Create object -- customscript_userevent.customdeploy_ue_tut (scriptdeployment)
2016-08-09 06:05:37 (PST) Installation COMPLETE (0 minutes 27 seconds)
```

**Working with SuiteScript Projects in SuiteCloud IDE Plug-in for Eclipse**

A SuiteScript project is a type of project that is based on the JavaScript project but with the NetSuite SuiteScript library automatically added to enable the auto completion and content assist features.

See the following topics for information about working with SuiteScript projects:

- Creating a SuiteScript Project with SuiteCloud IDE Plug-in for Eclipse
- Uploading Files in a SuiteScript Project with SuiteCloud IDE Plug-in for Eclipse
- Downloading Files in a SuiteScript Project with SuiteCloud IDE Plug-in for Eclipse
Creating a SuiteScript Project with SuiteCloud IDE Plug-in for Eclipse

Use the following steps to create a SuiteScript project in the Eclipse plug-in.

To create a SuiteScript project with SuiteCloud IDE plug-in for Eclipse:

2. Choose SuiteScript Project as project type. Select Next.
3. Enter a Project name.
   The IDE plug-in uses the libraries for the specific SuiteScript version for code completion and code validation. After you create the project, the libraries appear under Javascript Resources in the NS Explorer for the project:

5. Check Use default location.
   If you do not want to use the default location, clear the Use default location box and navigate to your desired location.
6. Click Finish.

Uploading Files in a SuiteScript Project with SuiteCloud IDE Plug-in for Eclipse

You can upload files in a SuiteScript project to the File Cabinet of your account. If the project does not exist yet in your account File Cabinet, a folder with the same name as the SuiteScript project is created.

To upload files in a SuiteScript project with SuiteCloud IDE plug-in for Eclipse:

1. In Eclipse, right-click a project in the NS Explorer pane, and select NetSuite > Upload File(s) to Project from the context menu.
   The Upload File(s) to Project window opens.
2. Select an Account.
3. Select a Role.
5. Browse and select a File Cabinet Folder if you want to upload to a different folder from the default.
6. Click OK.

Downloading Files in a SuiteScript Project with SuiteCloud IDE Plug-in for Eclipse

You can download files in a SuiteScript project from your account File Cabinet. If you are downloading files that already exist in your project, the existing files are backed up prior to download with the .bak extension.

You can choose to download files to their respective project folders using Use project name in file cabinet or to a single target project folder using Use this project name.

**Note:** If you are downloading files for a project that is a closed project in the IDE plug-in, you need to reopen the project first before you can download the files. For more information about closing and reopening projects, see the Closing projects topic in the Eclipse help.

To download files in a SuiteScript project with SuiteCloud IDE plug-in for Eclipse:

1. In Eclipse, right-click a project in the NS Explorer pane, and select NetSuite > Download File(s) to Project in the context menu.
   The Download File(s) to Project window opens.
2. Select an Account.
3. Select a Role.
5. Click Get File List.
6. When the Files(s) to download field is populated, check the box for the files that you want to download.
   Hidden bundles, inactive files, and empty folders are excluded from the file list.
7. Choose Use project name in file cabinet, or choose Use this project name and enter the project name that you want to use.
8. Click OK.

Working with SuiteScript Files in SuiteCloud IDE Plug-in for Eclipse

See the following topics for information about working with SuiteScript files:

- Creating a SuiteScript File in SuiteCloud IDE Plug-in for Eclipse
- Uploading a SuiteScript File in SuiteCloud IDE Plug-in for Eclipse
- Downloading a SuiteScript File in SuiteCloud IDE Plug-in for Eclipse

SuiteCloud IDE Plug-in for Eclipse Guide
Creating a SuiteScript File in SuiteCloud IDE Plug-in for Eclipse

Use the following steps to create a SuiteScript file in the IDE plug-in.

To create a SuiteScript file in SuiteCloud IDE Plug-in for Eclipse:

2. Select a Script Type.
   The SuiteScript Version defaults to the SuiteScript version chosen for the project. To change this value, change the SuiteScript version for the project. See Changing Project Settings in SuiteCloud IDE Plug-in for Eclipse.
3. Enter or select a parent folder.
4. Change the Script Filename if desired.
5. If you are creating a Client Script, RESTlet, or User Event for SuiteScript version 1.0, select the Event Type(s) and skip to step 8.
6. If you are creating a SuiteScript version 2.0 file, complete the following steps to create a SuiteScript file with a define function that includes specific modules:
   a. Click Add Modules.
      The Select SuiteScript Module(s) window appears.

   b. Select the modules that you want to include in the define function for the SuiteScript file from the Available Module(s) list and click the right arrow.
      You can remove modules from the Selected Module(s) list and click the left arrow to remove them.
c. Click OK.

The IDE plug-in includes the modules you select in the define function for the script file. For example, if you added the file and record modules, a client script file appears as follows:

```javascript
/**
   * Module Description
   *
   * Version Date Author Remarks
   * 1.00 18 Oct 2015 Jdoe
   */

function(file, record) {
   // Function to be executed after page is initialized.
}
```

7. Click Finish.

**Note:** You can also create a SuiteScript file through the toolbar. Click the dropdown arrow beside the NetSuite icon and select SuiteScript File.

For information about supported script types, see the following topics:
- SuiteScript 1.0 Script Types Overview
- SuiteScript 2.0 Script Types

### Uploading a SuiteScript File in SuiteCloud IDE Plug-in for Eclipse

You can upload a SuiteScript file to the NetSuite file cabinet from a SuiteScript project in your IDE plug-in. If you are uploading a SuiteScript file from a project that does not exist yet in the NetSuite file cabinet, a folder with the same name as the SuiteScript project that the file belongs to is created.

You can set file properties, including Available for SuiteBundles, Hide in SuiteBundles, and Available without Login, from within the IDE plug-in before uploading a file to the file cabinet. See Defining SuiteScript File and Folder Properties in SuiteCloud IDE Plug-in for Eclipse Before Upload.

You can upload multiple files, but all the files must belong to a single project.

**To upload a SuiteScript file in SuiteCloud IDE Plug-in for Eclipse:**

1. In Eclipse, right-click a file in the NS Explorer pane and select **NetSuite > Upload Selected File(s).** If the file is already associated with an account, the upload begins immediately.

   If an account is not associated with an account, the Upload Selected File(s) window opens. Complete the following:
   - Select an Account.
   - Select a Role.
   - Select the SuiteScript Version.
d. Select a File Cabinet Folder.

e. Click OK.

**Note:** You can upload a file directly from the Editor area. With the file visible in the Editor, right-click and select NetSuite > Upload File in Editor. You can also use the shortcut Ctrl+U. For more information about available shortcuts, see SuiteCloud IDE Plug-in for Eclipse Shortcuts.

---

### Defining SuiteScript File and Folder Properties in SuiteCloud IDE Plug-in for Eclipse Before Upload

You can define file and folder properties from within the IDE plug-in before you upload to the file cabinet. You do not need to locate file records in the file cabinet to set properties. You can set properties for a single selected file or for multiple selected files at the same time. You also can set properties for one or more selected folders and for files in the folder(s).

The properties you can set one file at a time include:

- **Description**
- **Tags** – SSP files only
- **Character Encoding** – Specifies the file’s character encoding. During upload, this value populates the file’s textFileEncoding attribute. If left blank, the default value is utf8.
- **Available in SuiteBundles** – Check this box to enable this script to be displayed in the Bundle Builder and included in SuiteApps.
- **Hide in SuiteBundles** – Check this box to prevent users who install a SuiteApp containing this script from seeing its contents.
- **Available Without Login** – Check this box to permit users without an active NetSuite session to access to this script.

When you select multiple files, only the last three properties are available.

**Important:** Any changes made to file properties directly in the file cabinet are not automatically available in the IDE plug-in.

### To define properties for a selected file in SuiteCloud IDE Plug-in for Eclipse:

1. Select one or more files in the NS Explorer.
2. Right-click, and select NetSuite > Set File Properties.
3. Set properties as desired in the Set File Properties window.
4. Do one of the following:
   - Click Cancel to close the dialog without saving.
   - Click OK to save changes and close the window.
   - Click Apply to save changes and leave the window open.

   □ If you click Apply, you can select another file from the File dropdown list and set its properties.

### To define properties for selected folders and files in folders in SuiteCloud IDE Plug-in for Eclipse:

1. Select one or more folders in NS Explorer.
2. Right-click and select **NetSuite > Set Folder Properties** to open the Set Folder Properties window.

3. Check or clear the boxes for the properties.
   - **Available for SuiteBundles** - Check this box to make this folder available in the Bundle Builder and included in SuiteApps.
   - **Apply changes to this folder, subfolders and files** - When you check this box, updates are made on save to the selected folders and all folders and files within the selected folders. When you clear this box, updates are made on save only to properties for the selected folders.

4. Click **OK**.

### Downloading a SuiteScript File in SuiteCloud IDE Plug-in for Eclipse

You can download a SuiteScript file from the NetSuite file cabinet to a specified SuiteScript project in your IDE plug-in.

If you have enabled the file backup preference, the existing file is backed up prior to download and has the .bak extension. For more information, see *Setting SuiteCloud IDE Plug-in for Eclipse Preferences*.

You can download multiple files, but all the files must belong to the same project. For instructions to download files to a project, see the following topics:

- Downloading Files in a SuiteScript Project with SuiteCloud IDE Plug-in for Eclipse
- Downloading Files in an SSP Application Project in SuiteCloud IDE Plug-in for Eclipse

**To download a SuiteScript file in SuiteCloud IDE Plug-in for Eclipse:**

1. In Eclipse, right-click on a file in the NS Explorer pane, and select **NetSuite > Download Selected File(s)**.

   If the file is already associated with an account, the download begins immediately.

   If an account is not associated with an account, the Download Selected File(s) window opens. Complete the following:
   - Select an **Account**.
   - Select a **Role**.
   - Select the **SuiteScript Version**.
   - Select a **File Cabinet Folder**.
   - Click **OK**.

2. Click **OK**.

**Note:** You can download a file directly from your Editor area. Right-click anywhere in the area when the file is open, and select **NetSuite > Download File in Editor**.

### Comparing a SuiteScript File with File Cabinet Copy in SuiteCloud IDE Plug-in for Eclipse

Use the following steps to compare your SuiteScript file with its file cabinet copy.
To compare a SuiteScript file with file cabinet copy in SuiteCloud IDE Plug-in for Eclipse:

1. In Eclipse, right-click a file in the NS Explorer pane and select NetSuite > Compare Selected File with File Cabinet Copy.

Note: You can compare a file with its file cabinet copy directly from your Editor area. Right-click anywhere on the area where the file is open, and select NetSuite > Compare Selected File with File Cabinet Copy.

Working with SSP Application Projects in SuiteCloud IDE Plug-in for Eclipse

SSP application projects are packaged NetSuite web store customization projects that you can use to fully customize key NetSuite web store touch points, such as login, cart, and checkout. You can use familiar HTML and SuiteScript, and even other e-commerce platforms for these customizations. For more information about SSP applications, see the help topic SSP Application Overview.

Important: The current version of the IDE plug-in only supports code completion for SuiteScript files. Code completion for SSP files is not yet supported, and the editor utilized for these SSP files is the default HTML editor of Eclipse. Also, SSP applications currently are not supported in SuiteScript 2.0.

With SSP application projects, you can do the following procedures:

- Creating an SSP Application Project with SuiteCloud IDE Plug-in for Eclipse
- Uploading Files in an SSP Application Project with SuiteCloud IDE Plug-in for Eclipse
- Downloading Files in an SSP Application Project in SuiteCloud IDE Plug-in for Eclipse

Creating an SSP Application Project with SuiteCloud IDE Plug-in for Eclipse

Use the following steps to create an SSP application project in the Eclipse plug-in.

To create an SSP application project with SuiteCloud IDE plug-in for Eclipse:

1. In Eclipse, select File > New > SuiteCloud Project to open the New SuiteCloud Project window.
2. Select SSP Application Project as project type.
3. Enter a Project Name.

Note: SuiteScript 2.0 is not currently supported for SSP application projects.

5. Check the Use default location box, or clear the box and navigate to your desired location.
6. Click Finish.
Uploading Files in an SSP Application Project with SuiteCloud IDE Plug-in for Eclipse

You can upload files in an SSP application project to your account's File Cabinet. You can select only one project to upload files from.

If you are uploading files from a project that is closed in the IDE plug-in, you need to reopen it before downloading the files. For more information about closing and reopening projects, see the Closing projects topic in the Eclipse help.

**Note:** You should have an application record created for your SSP application project beforehand. This enables you to select the directory structure you need in the File Cabinet Folder field when you upload your project files. For more information, see the help topic Create a SuiteScript 1.0 SSP Application Record.

**To upload files in an SSP application project with SuiteCloud IDE plug-in for Eclipse:**

1. In Eclipse, right-click an SSP application project in the NS Explorer pane, and select NetSuite > Upload File(s) to Project.
   
   The Upload File(s) to Project window opens.
2. Select an Account.
3. Select a Role.
5. Browse and select a File Cabinet Folder.
6. Click OK.

Downloading Files in an SSP Application Project in SuiteCloud IDE Plug-in for Eclipse

You can download files in an SSP application project from your account's File Cabinet. If you have enabled the file backup preference, any existing files are backed up prior to download and have the .bak extension. For more information, see Setting SuiteCloud IDE Plug-in for Eclipse Preferences.

You can choose to download files to their respective project folders using Use project name in file cabinet or to a single target project folder using Use this project name.

**Note:** If you are downloading files for a project that is a closed project in the IDE plug-in, you need to reopen the project first before you can download the files. For more information about closing and re-opening projects, see the Closing projects topic in the Eclipse help.

**To download files in an SSP application project in SuiteCloud IDE plug-in for Eclipse:**

1. In Eclipse, right-click an SSP application project in the NS Explorer pane, and select NetSuite > Download File(s) to Project in the context menu.
   
   The Download File(s) to Project window opens.
2. Select an Account.
3. Select a Role.
4. Select the **SuiteScript Version**.

5. Click **Get File List**.

6. When the **Files(s) to download** field is populated, check the box for the files that you want to download.
   
   Hidden bundles, inactive files, and empty folders are excluded from the file list.

7. Choose **Use project name in file cabinet**, or choose **Use this project name** and enter the project name that you want to use.

8. Click **OK**.

### Changing Project Settings in SuiteCloud IDE Plug-in for Eclipse

You can change the settings of a project such as accounts, roles, SuiteScript version, and file cabinet folders.

**To change project settings in SuiteCloud IDE plug-in for Eclipse:**

1. In Eclipse, right-click a project in the NS Explorer pane, and select **NetSuite > Change Project Settings** to open the Change Project Settings window.

2. Do any of the following:
   
   - Select an **Account**.
   
   - Select a **Role**.
   
   - Select a **SuiteScript Version**. The SuiteScript version determines the libraries used for code completion and code validation.

   **Note:** Code completion for SuiteScript 2.0 module objects, methods, and enums is provided. Validation of SuiteScript 2.0 code is **not** provided.

   For more information, see Creating a SuiteScript Project with SuiteCloud IDE Plug-in for Eclipse or Creating an SSP Application Project with SuiteCloud IDE Plug-in for Eclipse.

   - If you are working with an account customization project, you can change the selection for **Action on Account-Specific Values**. For more information about this setting, see the help topic Account-Specific Values.

   - Select a **File Cabinet Folder**.

3. Click **OK**.

### Converting a Bundle into an Account Customization Project in SuiteCloud IDE Plug-in for Eclipse

You can retrieve a list of customization bundles in your NetSuite account and import the components into an account customization project. For more information, see the help topic Conversion of a Bundle to an Account Customization Project.
Before you can convert a bundle, you must enable the **Create Bundles With SuiteBundler** feature in your NetSuite account. This feature is located under **Setup > Company > Setup Tasks > Enable Features**.

1. Create an account customization project.  
   For more information, see **Creating an Account Customization Project in SuiteCloud IDE Plug-in for Eclipse**.

2. Right-click your project in the **NS Explorer** subtab, and go to **NetSuite > Import Bundle Components**.

3. Choose the account and role associated with your account, and click **List Bundles**.  
   The IDE plug-in connects to the NetSuite account and retrieves a list of customization bundles found in the account.

4. Choose the bundle that you want to import, and then click **OK**.

5. Click **OK** to start the import process.  
   The import process could take a few minutes, depending on the size and number of components to import.

6. Review the import log in the **Console** subtab.  
   The log displays all of the files and objects that were imported, the dependency references that were added to the manifest file, and all of the components that were excluded from the import process.

7. Deploy the project to your NetSuite account.  
   For more information, see the help topics **SDF Project Deployment Preparation** and **Deploying an SDF Project to Your NetSuite Account in SuiteCloud IDE Plug-in for Eclipse**.

### Converting a Project to a SuiteScript Project in SuiteCloud IDE Plug-in for Eclipse

You can convert any project to a SuiteScript project. Only the type and the associated NetSuite libraries of a project are changed.

**To convert a project to a SuiteScript project in SuiteCloud IDE Plug-in for Eclipse:**

1. In Eclipse, right-click a non-SuiteScript project in the NS Explorer pane, and select **NetSuite > Convert to SuiteScript Project**.  
   The project is converted into a SuiteScript project.

### Converting a Project to an SSP Application Project in SuiteCloud IDE Plug-in for Eclipse

You can convert a project to an SSP application project. Only the type and the associated NetSuite libraries of a project are changed.

**To convert a project to an SSP application project in SuiteCloud IDE Plug-in for Eclipse:**

1. In Eclipse, right-click a non-SSP application project in the NS Explorer pane and select **NetSuite > Convert to SSP Application Project**.
The project is converted into an SSP application project.

Note: SSP applications currently are not supported in SuiteScript 2.0.

Logging in to a Project Account from SuiteCloud IDE Plug-in for Eclipse

You can log in to a project account directly from the IDE plug-in.

Important: The current version of the IDE plug-in 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 in SuiteCloud IDE Plug-in for Eclipse:

1. In Eclipse, 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 Plug-in for Eclipse 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 Eclipse plug-in 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 Eclipse plug-in Debugger similar.

The Eclipse plug-in 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) in SuiteCloud IDE Plug-in for Eclipse:** Enables you to debug a single script file that does not have a defined script and script deployment record.
- **Project Mode (Deployed Debugging) in SuiteCloud IDE Plug-in for Eclipse:** 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 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
The Eclipse plug-in 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 Eclipse plug-in Debugger.

- Verifying Your Editor in SuiteCloud IDE Plug-in for Eclipse
- Working with Breakpoints in SuiteCloud IDE Plug-in for Eclipse
- Creating Debug Configurations in SuiteCloud IDE Plug-in for Eclipse
- Debug Perspective in SuiteCloud IDE Plug-in for Eclipse
- Controlling Execution in SuiteCloud IDE Plug-in for Eclipse
- Evaluating Expressions in SuiteCloud IDE Plug-in for Eclipse
- Single File Mode (Ad Hoc Debugging) in SuiteCloud IDE Plug-in for Eclipse
- Project Mode (Deployed Debugging) in SuiteCloud IDE Plug-in for Eclipse
- Using the RESTlet/Suitelet Debug Client in SuiteCloud IDE Plug-in for Eclipse

### Verifying Your Editor in SuiteCloud IDE Plug-in for Eclipse

Before using the Eclipse plug-in 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.

![DebugTestScript.js](image)

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 in SuiteCloud IDE Plug-in for Eclipse

A breakpoint is a marker you place next to a line of code that tells the Eclipse plug-in 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) in SuiteCloud IDE Plug-in for Eclipse**, 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 in SuiteCloud IDE Plug-in for Eclipse before you add breakpoints.

For additional information on modifying breakpoints, see:

- Adding Breakpoints in SuiteCloud IDE Plug-in for Eclipse
- Removing Breakpoints in SuiteCloud IDE Plug-in for Eclipse
- Disabling Breakpoints in SuiteCloud IDE Plug-in for Eclipse
- Enabling Breakpoints in SuiteCloud IDE Plug-in for Eclipse

Adding Breakpoints in SuiteCloud IDE Plug-in for Eclipse

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:

```
23 var dt = nlapDateToString(new Date(), 'datetimes');
24 nlaplogExecution('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 in SuiteCloud IDE Plug-in for Eclipse

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 in SuiteCloud IDE Plug-in for Eclipse

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.

```
23 var dt = nlapDateToString(new Date(), 'datetimes');
24 nlaplogExecution('debug', 'value of dt', dt);
```

Enabling Breakpoints in SuiteCloud IDE Plug-in for Eclipse

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

Creating Debug Configurations in SuiteCloud IDE Plug-in for Eclipse

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 in SuiteCloud IDE plug-in for Eclipse:

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.

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 in SuiteCloud IDE Plug-in for Eclipse**

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 the following topics:

- Controlling Execution in SuiteCloud IDE Plug-in for Eclipse
- Evaluating Expressions in SuiteCloud IDE Plug-in for Eclipse

The debug perspective is composed of the following:

- Breakpoints View in SuiteCloud IDE Plug-in for Eclipse
- Call Stack in SuiteCloud IDE Plug-in for Eclipse
- Code Editor in SuiteCloud IDE Plug-in for Eclipse
- Console View in SuiteCloud IDE Plug-in for Eclipse
- Debugger Toolbar in SuiteCloud IDE Plug-in for Eclipse
- Expressions View in SuiteCloud IDE Plug-in for Eclipse
- Log View in SuiteCloud IDE Plug-in for Eclipse
- Outline View in SuiteCloud IDE Plug-in for Eclipse
- Variables View in SuiteCloud IDE Plug-in for Eclipse
Breakpoints View in SuiteCloud IDE Plug-in for Eclipse

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

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 in SuiteCloud IDE Plug-in for Eclipse.

**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 in SuiteCloud IDE Plug-in for Eclipse for additional information.

Call Stack in SuiteCloud IDE Plug-in for Eclipse

The call stack shows the files in your SuiteScript.

- 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 in SuiteCloud IDE Plug-in for Eclipse.

Code Editor in SuiteCloud IDE Plug-in for Eclipse

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

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 in SuiteCloud IDE Plug-in for Eclipse.

**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 in SuiteCloud IDE Plug-in for Eclipse**

The console view displays prompts and messages.

System errors and messages are displayed in the Log View in SuiteCloud IDE Plug-in for Eclipse.

**Debugger Toolbar in SuiteCloud IDE Plug-in for Eclipse**

The debugger toolbar provides buttons to resume execution, terminate execution, and step through your code. For additional information, see Controlling Execution in SuiteCloud IDE Plug-in for Eclipse.

**Expressions View in SuiteCloud IDE Plug-in for Eclipse**

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

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 Plug-in for Eclipse.

**Log View in SuiteCloud IDE Plug-in for Eclipse**

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.
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 in SuiteCloud IDE Plug-in for Eclipse**

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

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

**Variables View in SuiteCloud IDE Plug-in for Eclipse**

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

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 in SuiteCloud IDE Plug-in for Eclipse, the variables view changes accordingly.

**Controlling Execution in SuiteCloud IDE Plug-in for Eclipse**

The Eclipse plug-in 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 in SuiteCloud IDE Plug-in for Eclipse**.

<table>
<thead>
<tr>
<th>Functionality</th>
<th>Button / Menu Option</th>
<th>Shortcut Key</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Toggle Breakpoint</td>
<td>No button</td>
<td>Ctrl+Shift+B</td>
<td>Toggles a breakpoint on the currently selected line when in the Debug Perspective in SuiteCloud IDE Plug-in for Eclipse. This shortcut key does not work when in the NetSuite Perspective in SuiteCloud IDE Plug-in for Eclipse.</td>
</tr>
<tr>
<td>Resume</td>
<td>Right-click within the Call Stack in SuiteCloud IDE Plug-in for Eclipse and select Resume</td>
<td>F8</td>
<td>Resumes execution until the debugger reaches the next breakpoint.</td>
</tr>
<tr>
<td>Step Into</td>
<td></td>
<td>F5</td>
<td>Executes the currently selected line and then advances to the next line without executing</td>
</tr>
<tr>
<td>Functionality</td>
<td>Button / Menu Option</td>
<td>Shortcut Key</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------</td>
<td>----------------------</td>
<td>--------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Step Into</td>
<td>Right-click within the Call Stack in SuiteCloud IDE Plug-in for Eclipse and select Step Into.</td>
<td></td>
<td>If the selected line is a function (nlapi) or method call, the debugger steps into the first line of the function or method body.</td>
</tr>
<tr>
<td>Step Over</td>
<td>F6</td>
<td></td>
<td>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.</td>
</tr>
<tr>
<td>Functionality</td>
<td>Button / Menu Option</td>
<td>Shortcut Key</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------</td>
<td>----------------------</td>
<td>--------------</td>
<td>-------------</td>
</tr>
<tr>
<td>Step Return</td>
<td>Right-click within the Call Stack in SuiteCloud IDE Plug-in for Eclipse and select Step Return</td>
<td>F7</td>
<td>Steps out of the current function or method, and then advances to the next line without executing it.</td>
</tr>
<tr>
<td>Run to Line</td>
<td>No button</td>
<td>Ctrl+R</td>
<td>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.</td>
</tr>
<tr>
<td>Functionality</td>
<td>Button / Menu Option</td>
<td>Shortcut Key</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------</td>
<td>----------------------</td>
<td>--------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Run To Line</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Terminate</td>
<td></td>
<td>Ctrl+F2</td>
<td>Stops execution of the debugger. You must start a new session if you wish to continue debugging.</td>
</tr>
<tr>
<td>Relaunch</td>
<td>No button</td>
<td>F11</td>
<td>Starts a new debug session identical to the last session executed.</td>
</tr>
<tr>
<td>Functionality</td>
<td>Button / Menu Option</td>
<td>Shortcut Key</td>
<td>Description</td>
</tr>
<tr>
<td>------------------------</td>
<td>----------------------</td>
<td>------------------</td>
<td>---------------------------------------------------------------</td>
</tr>
<tr>
<td>Terminate and Relaunch</td>
<td></td>
<td>Ctrl+Alt+Shift+R</td>
<td>Stops execution of the debugger. Then starts a new debug session identical to the session terminated.</td>
</tr>
</tbody>
</table>

### Evaluating Expressions in SuiteCloud IDE Plug-in for Eclipse

The Eclipse plug-in 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.

<table>
<thead>
<tr>
<th>Functionality</th>
<th>Menu Option</th>
<th>Shortcut Key</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inspect</td>
<td>Right-click within the Code Editor in SuiteCloud IDE Plug-in for Eclipse and select Inspect.</td>
<td>Ctrl+Shift+Alt+I</td>
<td>Evaluates a highlighted expression, variable or object.</td>
</tr>
</tbody>
</table>
### Functionality | Menu Option | Shortcut Key | Description
--- | --- | --- | ---

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.

---

**Single File Mode (Ad Hoc Debugging) in SuiteCloud IDE Plug-in for Eclipse**

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 account from the Eclipse plug-in 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 in SuiteCloud IDE plug-in for Eclipse:**

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

   ```javascript
   function test(arg){...
   } 
   test(1);
   ```
3. Add at least one breakpoint. For information, see Working with Breakpoints in SuiteCloud IDE Plug-in for Eclipse.
4. If needed, create a new debug configuration for a SuiteScript file. For instructions, see Creating Debug Configurations in SuiteCloud IDE Plug-in for Eclipse. If you are using an existing debug configuration, select its shortcut.

   The view switches from the NetSuite Perspective in SuiteCloud IDE Plug-in for Eclipse to the Debug Perspective in SuiteCloud IDE Plug-in for Eclipse. 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 in SuiteCloud IDE Plug-in for Eclipse.
Project Mode (Deployed Debugging) in SuiteCloud IDE Plug-in for Eclipse

- **SuiteScript 2.0 Deployed Debugging in SuiteCloud IDE Plug-in for Eclipse**
- **SuiteScript 1.0 Deployed Debugging in SuiteCloud IDE Plug-in for Eclipse**

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 an 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 account from the Eclipse plug-in 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 in SuiteCloud IDE Plug-in for Eclipse**

To debug a deployed SuiteScript 2.0 project in SuiteCloud IDE plug-in for Eclipse:

1. Upload the SuiteScript files in your project to your account as a library file. For instructions, see [Uploading Files in a SuiteScript Project with SuiteCloud IDE Plug-in for Eclipse](#).
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 Eclipse plug-in SuiteScript editor. For instructions, see [Verifying Your Editor in SuiteCloud IDE Plug-in for Eclipse](#).
5. Add at least one breakpoint. For information, see Working with Breakpoints in SuiteCloud IDE Plug-in for Eclipse.

6. If needed, create a new debug configuration for a SuiteScript project and then select Debug. For instructions, see Creating Debug Configurations in SuiteCloud IDE Plug-in for Eclipse. If you are using an existing debug configuration, select its shortcut.

A browser window opens the 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 in SuiteCloud IDE Plug-in for Eclipse.

**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 in SuiteCloud IDE Plug-in for Eclipse.

### SuiteScript 1.0 Deployed Debugging in SuiteCloud IDE Plug-in for Eclipse

**To debug a deployed SuiteScript 1.0 project in SuiteCloud IDE Plug-in for Eclipse:**

1. Upload the SuiteScript files in your project to your account as a library file. For instructions, see Uploading Files in a SuiteScript Project with SuiteCloud IDE Plug-in for Eclipse.

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.


5. Add at least one breakpoint. For information, see Working with Breakpoints in SuiteCloud IDE Plug-in for Eclipse.

6. If needed, create a new debug configuration for a SuiteScript project and then select Debug. For instructions, see Creating Debug Configurations in SuiteCloud IDE Plug-in for Eclipse. If you are using an existing debug configuration, select its shortcut.

A browser window opens the account associated with your project.
SuiteCloud IDE Plug-in for Eclipse Debugger

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 in SuiteCloud IDE Plug-in for Eclipse.

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 in SuiteCloud IDE Plug-in for Eclipse.

Using the RESTlet/Suitelet Debug Client in SuiteCloud IDE Plug-in for Eclipse

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

To use the RESTlet/Suitelet client in SuiteCloud IDE Plug-in for Eclipse:

1. Start a deployed debug session. For instructions, see Project Mode (Deployed Debugging) in SuiteCloud IDE Plug-in for Eclipse.
2. Within the Code Editor in SuiteCloud IDE Plug-in for Eclipse, 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 in SuiteCloud IDE Plug-in for Eclipse.
   - If a log file is specified, the response is logged in the 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 in SuiteCloud IDE Plug-in for Eclipse.
  - 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 in SuiteCloud IDE Plug-in for Eclipse.

Adding or Editing a Multipart Body File Parameter in SuiteCloud IDE Plug-in for Eclipse

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

**To add or edit a new File parameter in SuiteCloud IDE plug-in for Eclipse:**

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 SuiteScript 1.0 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**.

Launching the SuiteScript Records Browser from SuiteCloud IDE Plug-in for Eclipse

You can launch the SuiteScript Records Browser from the Eclipse plug-in. The Records Browser provides a web-based view of all records, fields, sublists, search joins, search filters, search columns, and record transformations that are supported in SuiteScript.

For more information, see the help topic Using the SuiteScript Records Browser.
Launching the SuiteScript Records Browser from SuiteCloud IDE Plug-in for Eclipse

To launch Records Browser in SuiteCloud IDE Plug-in for Eclipse:

1. In Eclipse, select Help > Launch Records Browser. The Records Browser launches from the Help Center in an external browser window.

   **Note:** You can launch the SuiteScript Records Browser from the editor by pressing Ctrl + mouse over an internal id for a record, record field, or sublist. For more information, see Shortcuts in SuiteCloud IDE Plug-in for Eclipse Tips and Guidelines.

Viewing SuiteCloud IDE Plug-in for Eclipse Error Logs

You can view error logs to help you diagnose or troubleshoot the Eclipse plug-in. Viewing error logs is particularly useful for Support teams.

Another way to view error logs in your Eclipse plug-in is to go to NetSuite > TroubleShoot > View Log File.

To view error logs in SuiteCloud IDE plug-in for Eclipse:

1. In Eclipse, select Help > About Eclipse. The About Eclipse window opens.
2. Click Installation Details. The Eclipse Installation Details window opens.

   **Important:** You can see the version of the Eclipse plug-in from this window. The version number is necessary when filing issues.

3. Select SuiteCloud IDE, and select the Configuration tab for the Eclipse plug-in
4. Click View Error Log to open the error log details in your default browser.