

# SuiteCloud IDE Guide



Copyright © 2005, 2018, Oracle and/or its affiliates. All rights reserved.

This software and related documentation are provided under a license agreement containing restrictions on use and disclosure and are protected by intellectual property laws. Except as expressly permitted in your license agreement or allowed by law, you may not use, copy, reproduce, translate, broadcast, modify, license, transmit, distribute, exhibit, perform, publish, or display any part, in any form, or by any means. Reverse engineering, disassembly, or decompilation of this software, unless required by law for interoperability, is prohibited.

The information contained herein is subject to change without notice and is not warranted to be error-free. If you find any errors, please report them to us in writing.

If this is software or related documentation that is delivered to the U.S. Government or anyone licensing it on behalf of the U.S. Government, then the following notice is applicable:

U.S. GOVERNMENT END USERS: Oracle programs, including any operating system, integrated software, any programs installed on the hardware, and/or documentation, delivered to U.S. Government end users are "commercial computer software" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, use, duplication, disclosure, modification, and adaptation of the programs, including any operating system, integrated software, any programs installed on the hardware, and/or documentation, shall be subject to license terms and license restrictions applicable to the programs. No other rights are granted to the U.S. Government.

This software or hardware is developed for general use in a variety of information management applications. It is not developed or intended for use in any inherently dangerous applications, including applications that may create a risk of personal injury. If you use this software or hardware in dangerous applications, then you shall be responsible to take all appropriate fail-safe, backup, redundancy, and other measures to ensure its safe use. Oracle Corporation and its affiliates disclaim any liability for any damages caused by use of this software or hardware in dangerous applications.

Oracle and Java are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners.

Intel and Intel Xeon are trademarks or registered trademarks of Intel Corporation. All SPARC trademarks are used under license and are trademarks or registered trademarks of SPARC International, Inc. AMD, Opteron, the AMD logo, and the AMD Opteron logo are trademarks or registered trademarks of Advanced Micro Devices. UNIX is a registered trademark of The Open Group.

This software or hardware and documentation may provide access to or information about content, products, and services from third parties. Oracle Corporation and its affiliates are not responsible for and expressly disclaim all warranties of any kind with respect to third-party content, products, and services unless otherwise set forth in an applicable agreement between you and Oracle. Oracle Corporation and its affiliates will not be responsible for any loss, costs, or damages incurred due to your access to or use of third-party content, products, or services, except as set forth in an applicable agreement between you and Oracle.

If this document is in public or private pre-General Availability status:

This documentation is in pre-General Availability status and is intended for demonstration and preliminary use only. It may not be specific to the hardware on which you are using the software. Oracle Corporation and its affiliates are not responsible for and expressly disclaim all warranties of any kind with respect to this documentation and will not be responsible for any loss, costs, or damages incurred due to the use of this documentation.

If this document is in private pre-General Availability status:

The information contained in this document is for informational sharing purposes only and should be considered in your capacity as a customer advisory board member or pursuant to your pre-General Availability trial agreement only. It is not a commitment to deliver any material, code, or functionality,

and 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](http://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](mailto: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.

# Table of Contents

|   |    |
|---|----|
| SuiteCloud IDE Overview .....   | 1  |
| Setting Up SuiteCloud IDE .....   | 2  |
| Installing and Setting Up SuiteCloud IDE .....                                      | 2  |
| SuiteCloud IDE Installation Prerequisites .....                                     | 3  |
| Supported Operating Systems and Browsers for SuiteCloud IDE .....                   | 3  |
| Downloading Eclipse for Use with SuiteCloud IDE .....                               | 3  |
| Updating Eclipse for Use with SuiteCloud IDE .....                                  | 4  |
| Upgrading SuiteCloud IDE for Eclipse from Kepler to Mars .....                      | 6  |
| Launching SuiteCloud IDE and Using the Account Setup Wizard .....                   | 7  |
| Selecting a Workspace .....   | 8  |
| SuiteCloud IDE Master Password .....  | 8  |
| Setting a SuiteCloud IDE Master Password .....                                      | 9  |
| Authenticating a SuiteCloud IDE Master Password .....                               | 9  |
| Revoking a SuiteCloud IDE Master Password .....                                     | 9  |
| Changing a SuiteCloud IDE Master Password .....                                     | 10 |
| Resetting Your SuiteCloud IDE Master Password and Account Info .....                | 10 |
| SuiteCloud IDE Environment Setup .....  | 10 |
| Adding an Environment in SuiteCloud IDE .....                                       | 11 |
| Modifying an Environment in SuiteCloud IDE .....                                    | 11 |
| Removing an Environment from SuiteCloud IDE .....                                   | 11 |
| SuiteCloud IDE Account Setup .....  | 12 |
| Adding an Account in SuiteCloud IDE .....   | 12 |
| Removing an Account from SuiteCloud IDE .....                                       | 13 |
| Managing Token-based Authentication in an Account using SuiteCloud IDE .....        | 13 |
| Importing Existing SuiteCloud Projects into SuiteCloud IDE .....                    | 14 |
| Synchronizing Internal IDs from a Project Account in SuiteCloud IDE .....           | 14 |
| Setting SuiteCloud IDE Preferences .....  | 16 |
| Setting SDF Project Preferences in SuiteCloud IDE .....                             | 16 |
| Setting SuiteScript Preferences in SuiteCloud IDE .....                             | 17 |
| Restoring SuiteCloud IDE Default Preferences .....                                  | 18 |
| Configuring NetSuite Shortcut Keys in Eclipse .....                                 | 18 |
| SuiteCloud IDE Usage .....  | 20 |
| NetSuite Perspective .....  | 20 |
| SuiteCloud IDE Tips and Guidelines .....  | 22 |
| SuiteCloud IDE Keyboard Shortcuts .....   | 22 |
| Shorthand for Adding Event Handlers to Code in SuiteCloud IDE .....                 | 24 |
| Validation Markers in SuiteCloud IDE .....  | 26 |
| SuiteCloud IDE Guidelines .....   | 26 |
| Automatic Code Completion .....   | 28 |
| Working with SDF Projects .....   | 29 |
| Creating an Account Customization Project .....                                     | 29 |
| Creating a SuiteApp Project .....   | 29 |
| Importing Account Components into an SDF Project .....                              | 30 |
| Defining SDF SuiteApp Object Dependencies .....                                     | 31 |
| Defining Account Component Dependencies from an Account Customization Project ..... | 31 |
| Defining Feature Dependencies .....   | 32 |
| Configuring Account Features in an Account Customization Project .....              | 33 |
| Setting Installation Preferences in a SuiteApp Project .....                        | 35 |
| Managing SDF Projects as Compressed Files .....                                     | 38 |
| Validating an SDF Project .....   | 38 |
| Deploying an SDF Project to Your NetSuite Account .....                             | 39 |
| Working with Custom Objects .....   | 40 |
| Creating an XML Definition .....  | 41 |

|  |    |
|--|----|
| Modifying an XML Definition in an SDF Project .....                          | 41 |
| Downloading the XML Definition of an Object from Your NetSuite Account ..... | 42 |
| Comparing a Custom SuiteCloud Object with a Custom NetSuite Object .....     | 42 |
| Overwriting Custom SuiteCloud Objects with Custom NetSuite Objects .....     | 43 |
| Working with SuiteScript Projects .....                                      | 44 |
| Creating a SuiteScript Project .....   | 44 |
| Uploading Files in a SuiteScript Project .....                               | 45 |
| Downloading Files in a SuiteScript Project .....                             | 45 |
| Working with SuiteScript Files .....   | 46 |
| Creating a SuiteScript File .....  | 46 |
| Uploading a SuiteScript File .....   | 48 |
| Defining SuiteScript File and Folder Properties Before Upload .....          | 48 |
| Downloading a SuiteScript File .....   | 49 |
| Comparing a SuiteScript File with File Cabinet Copy .....                    | 50 |
| Working with SSP Application Projects .....                                  | 50 |
| Creating an SSP Application Project .....                                    | 51 |
| Uploading Files in an SSP Application Project .....                          | 51 |
| Downloading Files in an SSP Application Project .....                        | 52 |
| Changing Project Settings in SuiteCloud IDE .....                            | 52 |
| Converting a Bundle Into an Account Customization Project .....              | 53 |
| Converting a Project to a SuiteScript Project .....                          | 53 |
| Converting a Project to an SSP Application Project .....                     | 53 |
| Logging in to a Project Account from SuiteCloud IDE .....                    | 54 |
| SuiteCloud IDE Debugger .....  | 54 |
| Verifying Your Editor .....  | 55 |
| Working with Breakpoints .....   | 55 |
| Creating Debug Configurations .....  | 56 |
| Debug Perspective .....  | 58 |
| Single File Mode (Ad Hoc Debugging) .....                                    | 63 |
| Project Mode (Deployed Debugging) .....                                      | 63 |
| Using the RESTlet/Suitelet Debug Client .....                                | 65 |
| Launching the SuiteScript Records Browser .....                              | 66 |
| Viewing SuiteCloud IDE Error Logs .....                                      | 66 |

# SuiteCloud IDE Overview

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

SuiteCloud IDE 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 consists of setup topics and topics that explain how to use the SuiteCloud IDE:

- [Setting Up SuiteCloud IDE](#)
- [SuiteCloud IDE Usage](#)

SuiteCloud IDE also provides a user interface for SuiteCloud Development Framework (SDF). SDF is a development framework and deployment mechanism. Additional information related to SuiteCloud IDE is included in the SDF documentation. See the help topic [SuiteCloud Development Framework Overview](#).

# Setting Up SuiteCloud IDE

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

- [Configuring NetSuite for SuiteScript](#) if you are using SuiteScript.
- [Enabling SuiteCloud Development Framework in Your NetSuite Account](#) if you are using SuiteCloud Development Framework.


After configuring your account, follow these steps to set up your SuiteCloud IDE:

1. [Installing and Setting Up SuiteCloud IDE](#)
2. [Launching SuiteCloud IDE and Using the Account Setup Wizard](#)
3. [Selecting a Workspace](#)
4. [SuiteCloud IDE Master Password](#)
5. [SuiteCloud IDE Environment Setup \(Optional\)](#)
6. [SuiteCloud IDE Account Setup](#)
7. [Importing Existing SuiteCloud Projects into SuiteCloud IDE](#)
8. [Synchronizing Internal IDs from a Project Account in SuiteCloud IDE](#)
9. [Setting SuiteCloud IDE Preferences](#)

If you choose to set up a SuiteScript development environment other than SuiteCloud IDE, see the help topic [Working with IDEs Other Than SuiteCloud IDE](#).


## Installing and Setting Up SuiteCloud IDE

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

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

The following table shows the supported versions of Eclipse for each SuiteCloud IDE version:

| SuiteCloud IDE Version | Eclipse Version |
|------------------------|-----------------|
| 2017.2.1               | 4.3.2           |
| 2018.1.1               | 4.5.2           |

 **Note:** SuiteCloud IDE for Eclipse 2018.1 is compatible with Eclipse Kepler but no longer certified. To use the latest version of SuiteCloud IDE with NetSuite 2018.1, Oracle recommends upgrading your Eclipse Kepler clients to Eclipse Mars.

After your NetSuite account is upgraded, you must manually update SuiteCloud IDE. The version of SuiteCloud IDE must match the version of your NetSuite account before you can use SuiteCloud IDE to create SuiteCloud Development Framework (SDF) projects

To install Eclipse and update SuiteCloud IDE, complete the following steps:

1. **Verify prerequisites.** See [SuiteCloud IDE Installation Prerequisites and Supported Operating Systems and Browsers for SuiteCloud IDE](#).

2. **Download Eclipse.** Download and unpack Eclipse. See [Downloading Eclipse for Use with SuiteCloud IDE](#).
3. **Update Eclipse.** Update Eclipse with SuiteCloud IDE from the SuiteCloud IDE update site. See [Updating Eclipse for Use with SuiteCloud IDE](#).

## SuiteCloud IDE Installation Prerequisites

Make sure you verify the following prerequisites before you install and use Eclipse with SuiteCloud IDE:

| Prerequisite                     | Description  |
|----------------------------------|--|
| Operating system and browser     | Verify the operating system version and browser that you want to use with SuiteCloud IDE.<br>See <a href="#">Supported Operating Systems and Browsers for SuiteCloud IDE</a> .   |
| Java Runtime Environment (JRE)   | SuiteCloud IDE 2017.2 requires JRE 1.7 and later.<br>SuiteCloud IDE 2018.1 requires JRE 1.8 and later.<br>If you use JRE 1.7 and work with either sandbox or Release Preview accounts, you must enable TLS 1.2 in the Java VM. To perform this task, update the eclipse.ini file in the root folder of your Eclipse installation to include the following line at the end of the file:<br><br><pre>-Dhttps.protocols=TLSv1.2</pre> |
| 64-bit requirements              | To properly run the 64-bit version of SuiteCloud IDE, make sure your operating system and Java versions are both 64-bit versions.  |
| 2-factor authentication disabled | SuiteCloud IDE only works in NetSuite accounts that do not have 2-factor authentication.   |
| Web services enabled             | Make sure SuiteTalk (Web Services) is enabled in the NetSuite account. See the help topic <a href="#">Enabling SuiteCloud Development Framework in Your NetSuite Account</a> .   |

## Supported Operating Systems and Browsers for SuiteCloud IDE

SuiteCloud IDE supports and has been tested with the following operating system and browser combinations:

| Operating System | Internet Explorer      | Firefox      | Safari       | Chrome      |
|------------------|------------------------|--------------|--------------|-------------|
| Windows 7        | Internet Explorer 10.x | Firefox 41.x |              | Chrome 40.x |
| Mac OS X 10.9.3  |                        | Firefox 41.x | Safari 7.0.2 | Chrome 40.x |
| Debian 7         |                        | Firefox 41.x |              |             |

## Downloading Eclipse for Use with SuiteCloud IDE

For supported Eclipse versions, see [Installing and Setting Up SuiteCloud IDE](#).

### To download and unpack Eclipse:

1. Download the Eclipse package from <http://www.eclipse.org>. NetSuite recommends Eclipse IDE for Java EE Developers. This version of Eclipse contains the required Javascript plug-in.



The following two versions are supported:

- Eclipse Kepler
  - Eclipse Mars
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. For more information, see [Updating Eclipse for Use with SuiteCloud IDE](#).

## Updating Eclipse for Use with SuiteCloud IDE

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

1. Add the SuiteCloud IDE Update Site.
2. Install SuiteCloud IDE.
3. Verify the Installation (optional).



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

## Add the SuiteCloud IDE Update Site


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



**Note:** These instructions are based on the Kepler version of Eclipse. Preferences may be accessed from a different menu in other versions.

### To add the update site:

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 SuiteCloud IDE 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 Update Site   |
| Location | <div>  <b>Note:</b> For compatibility reasons, your SuiteCloud IDE version must match the version of your NetSuite account.         </div> <p>If your NetSuite account is on version 17.2, use:</p> |

| Option | Description   |
|--------|---|
|        | <a href="http://system.netsuite.com/download/ide/update_17_2">http://system.netsuite.com/download/ide/update_17_2</a><br>If your NetSuite account is on version 18.1, use:<br><a href="http://system.netsuite.com/download/ide/update_18_1">http://system.netsuite.com/download/ide/update_18_1</a> |

- 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

When you install SuiteCloud IDE, select the update site as your installation source.

### To install SuiteCloud IDE:

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 SuiteCloud IDE. When the installation is complete, click **Yes** to restart Eclipse.

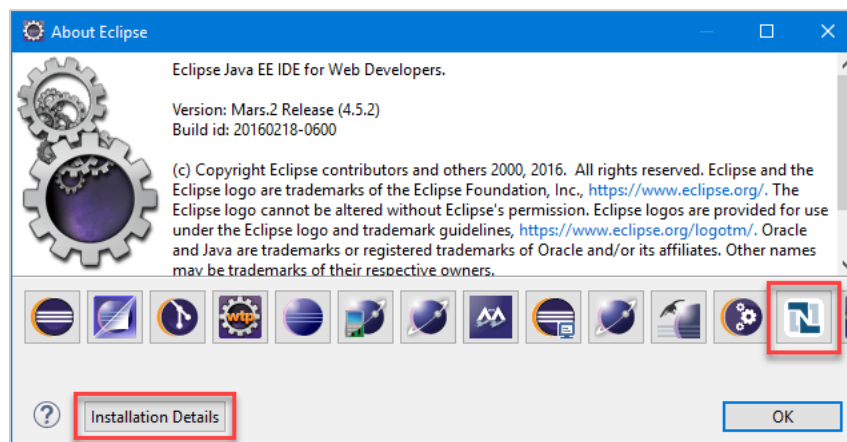
When Eclipse restarts, the SuiteCloud IDE Setup Wizard opens. For information and instructions, see [Launching SuiteCloud IDE and Using the Account Setup Wizard](#).

## Verify the Installation (optional)

When you install SuiteCloud IDE, information about the NetSuite SuiteCloud IDE version is added to the **About Eclipse** dialog box. You can verify this installation information at any time.

### To verify the installation:

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



NetSuite SuiteCloud IDE version information is displayed.

## Upgrading SuiteCloud IDE for Eclipse from Kepler to Mars

As part of upgrading SuiteCloud IDE 2017.2 or earlier, Oracle recommends upgrading your instance of Eclipse from Kepler to Mars.

Choose one of the following upgrade approaches:

- Use your existing SDF projects with a new installation of SuiteCloud IDE for Eclipse Mars. See [Using Existing SDF Projects with a New Eclipse Installation](#).
- Upgrade an existing Eclipse Kepler and SuiteCloud IDE installation that may also include SDF projects. See [Upgrading an Existing SuiteCloud IDE for Eclipse Kepler Installation](#).

## Using Existing SDF Projects with a New Eclipse Installation

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

To use existing SDF projects with a new installation:

1. Download and install Eclipse Mars. See [Downloading Eclipse for Use with SuiteCloud IDE](#).
2. Install the latest version of SuiteCloud IDE for Eclipse. See [Updating Eclipse for Use with SuiteCloud IDE](#). The complete steps for SuiteCloud IDE installation and configuration are available in [Setting Up SuiteCloud IDE](#).
3. Configure SuiteCloud IDE to use your NetSuite account. See [Launching SuiteCloud IDE and Using the Account Setup Wizard](#).
4. 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 SuiteCloud IDE, create each project individually using the same name. SuiteCloud IDE 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](#) and [Creating a SuiteApp Project](#).

## Upgrading an Existing SuiteCloud IDE for Eclipse Kepler Installation

You can upgrade an existing SuiteCloud IDE for Eclipse Kepler installation to SuiteCloud IDE for Eclipse Mars. The process involves upgrading both Eclipse and SuiteCloud IDE. Existing SDF projects are included in the upgrade.

To upgrade an existing SuiteCloud for Eclipse installation:

1. To mitigate upgrade risk, back up your existing Eclipse installation and workspace.
2. Upgrade your Eclipse Kepler installation to Eclipse Mars. For more information, see:

- Eclipse Kepler help topic [Upgrading Eclipse](#).
  - Eclipse FAQ entry [Upgrading existing Eclipse IDE and Installed Features to newer release](#).
3. Upgrade SuiteCloud IDE for Eclipse to the latest version by following the instructions in [Updating Eclipse for Use with SuiteCloud IDE](#). If you have an existing Location value specified for SuiteCloud IDE Update Site, change the value to the URL documented in the procedure.  
If you are using the same workspace, your existing SDF projects are available in the NS Explorer subtab. If you specified a new workspace, see the final step in [Using Existing SDF Projects with a New Eclipse Installation](#).

## Launching SuiteCloud IDE and Using the Account Setup Wizard

Use the steps below to launch the SuiteCloud IDE. The first time you launch the SuiteCloud IDE 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 SuiteCloud IDE. 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:

1. Navigate to the folder location of your SuiteCloud IDE.
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 for your IDE and to add NetSuite accounts to the IDE.

**Note:** Your IDE master password is not your NetSuite password. The IDE master password is used to ensure that the data in your IDE 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. After these accounts are added, you can upload all SuiteScript code changes from the IDE 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 SuiteCloud IDE, 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 SuiteCloud IDE Setup Wizard again.



**Important:** NetSuite recommends that you create a new workspace for SuiteCloud IDE 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, SuiteCloud IDE closes and reopens using the new workspace. Because it is a new session, you must authenticate your master password to upload, download, or interface with a NetSuite account. For instructions, see [Authenticating a SuiteCloud IDE Master Password](#).

## SuiteCloud IDE Master Password

SuiteCloud IDE 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 Master Password](#)
- [Authenticating a SuiteCloud IDE Master Password](#)
- [Revoking a SuiteCloud IDE Master Password](#)
- [Changing a SuiteCloud IDE Master Password](#)
- [Resetting Your SuiteCloud IDE Master Password and Account Info](#)

## Setting a SuiteCloud IDE Master Password

When you install SuiteCloud IDE 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 SuiteCloud IDE Setup Wizard when you launched SuiteCloud IDE, you can skip this procedure.

### To set a master password:

1. Launch SuiteCloud IDE.
2. In SuiteCloud IDE, go to NetSuite > Master Password > Set Master Password. The Set Master Password window opens.
3. Enter the following information: **New Master Password** and **Re-enter New Master Password**.
4. Click **OK**.

## Authenticating a SuiteCloud IDE 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. Launch SuiteCloud IDE.
2. In SuiteCloud IDE, go to NetSuite > Master Password > Authenticate Master Password. The Authenticate Master Password window opens.
3. Enter the master password.
4. 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 Keyboard Shortcuts](#).

## Revoking a SuiteCloud IDE 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 SuiteCloud IDE 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. Launch SuiteCloud IDE.
2. In SuiteCloud IDE, go to NetSuite > Master Password > Revoke Master Password.
3. 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 Master Password

You can change a master password.

### To change a master password:

1. Launch SuiteCloud IDE.
2. In SuiteCloud IDE, go to NetSuite > Master Password > Change Master Password. The Change Master Password window opens.
3. Enter the following information: **Old Master Password**, **New Master Password**, and **Re-enter New Master Password**.
4. Click **OK**.

## Resetting Your SuiteCloud IDE 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. Launch SuiteCloud IDE.
2. Go to NetSuite > Troubleshoot > Reset Master Password and Account Info.  
A message appears warning you that your SuiteCloud IDE will be restarted and that your master password and all of your account information will be deleted.
3. Click **OK**.  
SuiteCloud IDE restarts with your workspace still intact but your master password and account info deleted.
4. Set up your master password and account info. For more information, see [Setting a SuiteCloud IDE Master Password](#) and [Adding an Account in SuiteCloud IDE](#).

## SuiteCloud IDE Environment Setup

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

- Production
- Release Preview
- Sandbox

To set up an environment, see the following procedures:

- [Adding an Environment in SuiteCloud IDE](#)


- [Modifying an Environment in SuiteCloud IDE](#)
- [Removing an Environment from SuiteCloud IDE](#)

## Adding an Environment in SuiteCloud IDE

Use the following steps to add a NetSuite environment to SuiteCloud IDE.

### To add an environment:

1. Launch SuiteCloud IDE.
2. In SuiteCloud IDE, go to NetSuite > Manage Environments. The Manage Environments window opens.
3. Click **New**. The New Environment window opens.
4. Enter the following information: **Name** and **URL**.

 **Note:** If you do not know the correct URL to use for a particular environment, see the tables in [Understanding NetSuite URLs and Data Centers](#).


5. Click **OK** and then **Close**.

## Modifying an Environment in SuiteCloud IDE

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

### To modify an environment:

1. Launch SuiteCloud IDE.
2. In SuiteCloud IDE, go to NetSuite > Manage Environments. The Manage Environments window opens.
3. In the **Environment** list, select an environment that you want to edit.
4. Click **Edit**. The Edit Environment window opens.
5. Modify the following information: **Name** and **URL**.

 **Note:** If you do not know the correct URL to use for a particular environment, see the tables in [Understanding NetSuite URLs and Data Centers](#).

6. Click **OK** and then **Close**.

## Removing an Environment from SuiteCloud IDE

Use the following steps to remove a NetSuite environment from SuiteCloud IDE. You can remove unused or unnecessary environments to reduce the environment options to those you need for your work.

### To remove an environment:

1. Launch SuiteCloud IDE.



2. In SuiteCloud IDE, go to NetSuite > Manage Environments. The Manage Environments window opens.
3. In the **Environment** list, select the environment that you want to remove.
4. Click **Remove**. The Remove Environment dialog opens.
5. Click **OK** and then **Close**.

## SuiteCloud IDE 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 SuiteCloud IDE 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](#)
- [Removing an Account from SuiteCloud IDE](#)

## Adding an Account in SuiteCloud IDE

Use the following steps to add an account. SuiteCloud IDE passes your account information to NetSuite during the IDE 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, SuiteCloud IDE 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.

### To add an account:

1. Launch SuiteCloud IDE.
2. In SuiteCloud IDE, go to NetSuite > Manage Accounts. The Manage Accounts window opens.
3. Click **Add**. The Add Account(s) window opens.
4. Select an environment.
5. Enter the following information: **Email** and NetSuite **Password** associated with the email address.
6. Click **Next**. The Select Account(s) page opens.
7. Select the account you want to add. You may click **Select All**, or click **Deselect All** and then select what you need.
8. 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.

9. Click **Close**.

## Removing an Account from SuiteCloud IDE

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

## Managing Token-based Authentication in an Account using SuiteCloud IDE

Token-based authentication (TBA) can be used with non-administrator roles to log in to your NetSuite Production account from SuiteCloud IDE. By using TBA with SuiteCloud IDE, 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 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:

1. Ensure that the Token-based Authentication feature is enabled in the desired account.  
For information about enabling features, see the help topics [Enabling Features](#) and [SuiteCloud Features](#).
2. Grant permission to User Access Tokens on the desired role.  
For information about granting permissions to a custom SDF developer role, see the help topic [Creating an SDF Developer Role](#).
3. Launch SuiteCloud IDE.
4. In SuiteCloud IDE, go to NetSuite > Manage Accounts.
5. In the **Accounts** list, select the account that you want to log in to using token-based authentication.
6. Click **Manage Authentication**.

A window appears, showing all the roles that are assigned to the user. The **Token** column indicates whether you can revoke or issue a TBA token on that role.

**Issue** indicates that TBA is available for the role but not in use. Users with this role are logged in to NetSuite from SuiteCloud IDE using account credentials.

**Revoke** indicates that the role uses TBA to log in to NetSuite from SuiteCloud IDE.


7. Click **Issue** or **Revoke** to issue or revoke the TBA token on the role, respectively.
8. Click **Close** on both windows.

## Importing Existing SuiteCloud Projects into SuiteCloud IDE

If you already have projects in the NetSuite file cabinet, synchronize your file cabinet within SuiteCloud IDE by importing your projects into SuiteCloud IDE. Before doing so, ensure that all your project files are up-to-date in the NetSuite file cabinet.

### To import existing SuiteCloud projects into SuiteCloud IDE:

1. Launch SuiteCloud IDE.
2. In SuiteCloud IDE, authenticate your master password.
3. Right-click in the NS Explorer pane, and select **NetSuite > Download Project**. The Download Project window opens.
4. Select an **Account**.
5. Select a **Role**.
6. Click **Get File List**.


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

7. When the **File(s) to download** list is populated, select the projects you want to import.
8. Choose **Use project name in file cabinet**, or choose **Use this project name** and enter a project name in the field.
9. Click **OK**.
10. Check your project folder to verify that all files were imported.

## Synchronizing Internal IDs from a Project Account in SuiteCloud IDE

To use record ID, field ID and saved search ID code completion, you must first synchronize those values with your installation of SuiteCloud IDE. The Sync Script IDs from Account wizard takes you through this process step by step.

The pages that display in the wizard depend upon the ID types you select in the first window.

 **Important:** SuiteCloud IDE supports search ID code completion only for search types supported by both SuiteScript and SuiteTalk (Web Services). See the help topics [SuiteScript Supported Records](#) and the SuiteTalk Schema Browser for additional information.

### To synchronize internal IDs from a project account:

1. Within SuiteCloud IDE, 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**.

**Note:** If you select **Record Types** and there are fewer than the threshold number of custom records, all custom records are automatically synchronized. The threshold is one of the SuiteCloud IDE SuiteScript Code Completion preferences.

**Sync Script IDs from Account**  
This action might take a few minutes or more to complete depending on the number of entries in your account.

Select Type(s):

- ☒ Saved Searches
- ☒ Record Types
- ☒ CRM Fields
- ☒ Entity Fields
- ☒ Item Fields
- ☒ Other Custom Fields
- ☒ Transaction Body Fields
- ☒ Transaction Column Fields

3. On the Sync Search ID(s) from Account page, select the search IDs you want to synchronize, and click **Next**.

**Sync Search ID(s) from Account**  
This action might take a few minutes or more to complete depending on the number of entries in your account.

Select Search Type(s):

- ☒ Account
- ☒ Class
- ☒ Contact
- ☒ Customer
- ☒ Custom Record
- ☒ Department
- ☒ Employee
- ☒ Folder

4. On the Sync Record ID(s) from Account page, select the custom records you want to synchronize, and click **Next**.

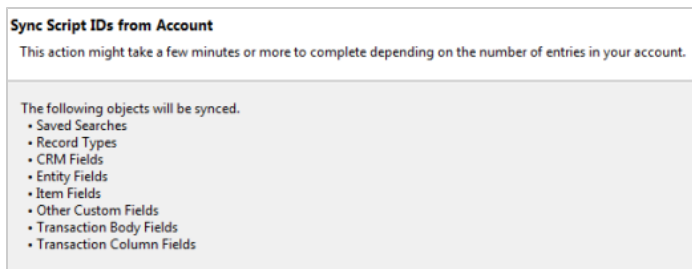
**Sync Record ID(s) from Account**  
This action might take a few minutes or more to complete depending on the number of entries in your account.

Select Record Type(s):

| Name   | ID                               |
|--|----------------------------------|
| <input checked="" type="checkbox"/> 2008 Customer Survey       | customrecord24                   |
| <input checked="" type="checkbox"/> 2010 Documentation Survey  | customrecord6_1                  |
| <input checked="" type="checkbox"/> ACT History                | customrecord80                   |
| <input checked="" type="checkbox"/> Another Test Custom Record | customrecordcustomrecord_test123 |
| <input checked="" type="checkbox"/> Campaign Automation Config | customrecord_campauto_config     |
| <input checked="" type="checkbox"/> Campaign Automation Rules  | customrecord_campauto_rules      |

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**.



## Setting SuiteCloud IDE Preferences

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

**Note:** These instructions are based on the Kepler version of Eclipse. Preferences may be accessed from a different menu in other versions.

### To set SuiteCloud IDE general preferences:

1. Launch SuiteCloud IDE.
2. 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 SuiteCloud IDE.
  - 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.
3. 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 SuiteCloud IDE preferences:

- [Setting SDF Project Preferences in SuiteCloud IDE](#)
- [Setting SuiteScript Preferences in SuiteCloud IDE](#)
- [Configuring NetSuite Shortcut Keys in Eclipse](#)
- [Restoring SuiteCloud IDE Default Preferences](#)

## Setting SDF Project Preferences in SuiteCloud IDE

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 SuiteCloud IDE 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 [Supported Custom NetSuite Objects](#).

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

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

Launch SuiteCloud IDE, 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:

- **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 `nlapiLoadRecord(` and press Ctrl+Spacebar. When you select an internal ID such as **salesorder**, `nlapiLoadRecord('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 Keyboard Shortcuts](#) and [Automatic Code Completion](#).

## 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:

| Type | Description  |
|------|--|
| Off  | Switches off the validation feature. System performance is not impacted.   |
| Fast | Determines context within the current file. System performance is average. |
| Full | Determines context across different files. System performance is slow.     |

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](#).

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 SuiteCloud IDE, see [Validation Markers in SuiteCloud IDE](#).

 **Note:** Validation is not provided for SuiteScript 2.0 code.

## Restoring SuiteCloud IDE Default Preferences

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

### To restore SuiteCloud IDE default preferences:

1. Launch SuiteCloud IDE, and go to 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.

## Configuring NetSuite Shortcut Keys in Eclipse

NetSuite shortcut keys are configured similarly to any other shortcut key in Eclipse.

### To configure NetSuite shortcut keys:

1. Open Eclipse, and create or select the SDF project.  
For more information, see [Creating an Account Customization Project](#) or [Creating a SuiteApp Project](#).
2. Go to Windows > Preferences > General > Keys.
3. In the filter text field, type **NetSuite**.  
A list appears that shows all the SuiteCloud IDE 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 SuiteCloud IDE, also see [SuiteCloud IDE Keyboard Shortcuts](#).



# SuiteCloud IDE Usage

Before you start working with SuiteCloud IDE, familiarize yourself with the following:

- [NetSuite Perspective](#)
- [SuiteCloud IDE Tips and Guidelines](#)
- [Automatic Code Completion](#)

Working with SuiteCloud IDE is divided into the following task areas:

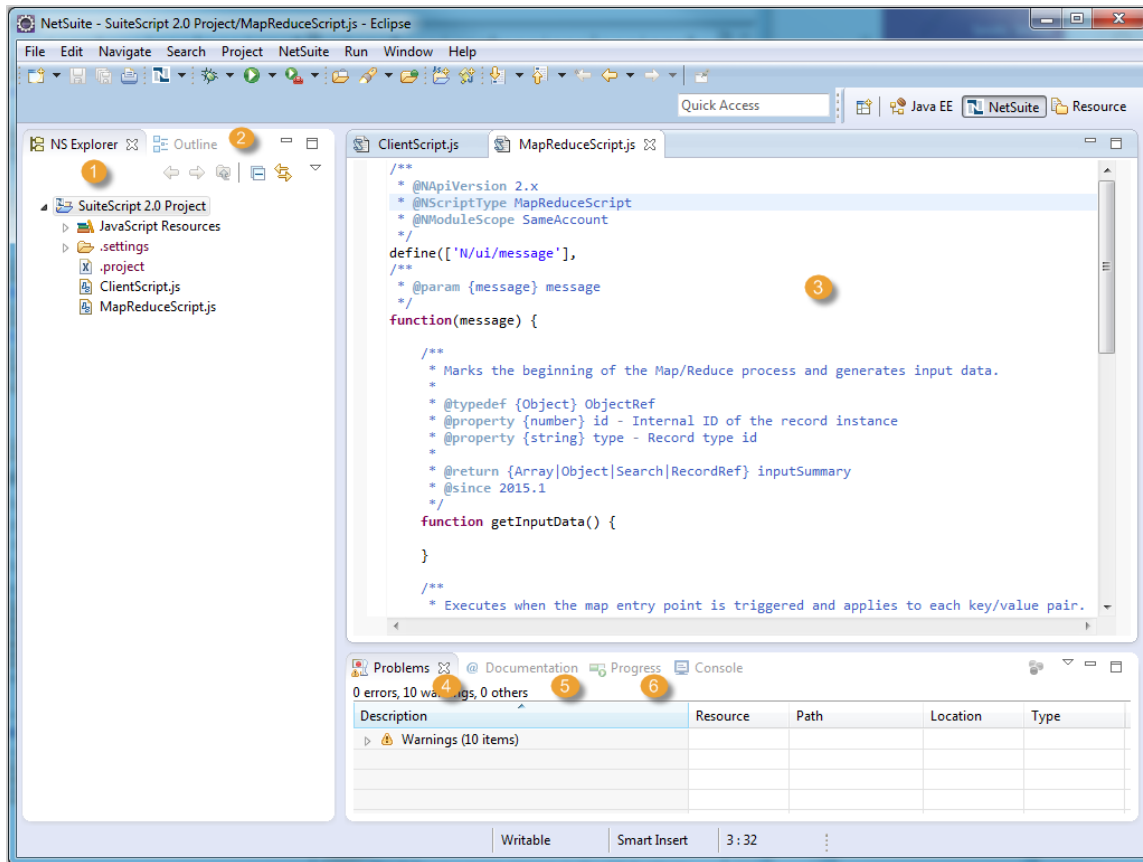
- [Working with SuiteScript Projects](#)
- [Working with SuiteScript Files](#)
- [Working with SSP Application Projects](#)
- [Changing Project Settings in SuiteCloud IDE](#)
- [Converting a Project to a SuiteScript Project](#)
- [Converting a Project to an SSP Application Project](#)
- [Logging in to a Project Account from SuiteCloud IDE](#)
- [SuiteCloud IDE Debugger](#)
- [Launching the SuiteScript Records Browser](#)
- [Viewing SuiteCloud IDE Error Logs](#)

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

## NetSuite Perspective

The NetSuite Perspective is an added perspective in the Eclipse workbench for SuiteCloud IDE. SuiteCloud IDE 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 SuiteCloud IDE interface:

| Element               | Description   |
|-----------------------|---|
| 1<br>NS Explorer View | <p>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.</p> <div style="border: 1px solid #add8e6; padding: 5px; margin: 10px 0;"> <p><b>Note:</b> SSP applications currently are not supported in SuiteScript 2.0.</p> </div> <p>In NS Explorer view, you can hover over the project folders to see the following details when available:</p> <ul style="list-style-type: none"> <li>■ Project Type</li> <li>■ NetSuite Account ID</li> <li>■ Company</li> <li>■ Email</li> <li>■ Environment</li> <li>■ File Cabinet Folder</li> <li>■ Sync Custom Field and Record ID(s) from Account</li> <li>■ SuiteScript Version</li> <li>■ Project Folder Name</li> <li>■ Available for SuiteBundles</li> </ul> <p>For more information, see the Project Explorer view topic in the Eclipse help.</p> |

| Element                         | Description  |
|---------------------------------|--|
| <b>2<br/>Outline View</b>       | The Outline view displays the organizational structure of your source file that is active in the editor area. In general, it lists the structural elements of your source file by function. For more information, see the Outline view topic in the Eclipse help.  |
| <b>3<br/>Editor Area</b>        | The Editor area displays the source file editors you can use for your code. The NetSuite Perspective mainly uses the JavaScript and HTML editor for SuiteCloud projects, such as SuiteScript and SSP application projects. For more information, see the Editors topic in the Eclipse help.              |
| <b>4<br/>Problems View</b>      | The Problems view displays the errors and warnings found in your source file that is active in the editor area. The information found in this view corresponds to the problem markers shown in the marker bar of the editor area. For more information, see the Problems view topic in the Eclipse help. |
| <b>5<br/>Documentation View</b> | The Documentation view displays the corresponding documentation or comment details you have for your code that is active in the editor area. This view displays only the documentation details for your functions. The same details appear in the popup when you click or hover over a function name.    |
| <b>6<br/>Progress View</b>      | The Progress view displays the progress bar of all your upload or download actions.  |

## SuiteCloud IDE Tips and Guidelines

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

| Topic   | SuiteScript Version | For more information, see ...   |
|---|---------------------|---|
| Using keyboard shortcuts when you work with files in the Eclipse editor | 1.0   2.0           | <a href="#">SuiteCloud IDE Keyboard Shortcuts</a>                             |
| Adding event handlers to SuiteScript 1.0 files                          | 1.0                 | <a href="#">Shorthand for Adding Event Handlers to Code in SuiteCloud IDE</a> |
| Viewing validation markers in the Eclipse editor                        | 1.0   2.0           | <a href="#">Validation Markers in SuiteCloud IDE</a>                          |
| Get general guidelines for using SuiteCloud IDE                         | 1.0   2.0           | <a href="#">SuiteCloud IDE Guidelines</a>                                     |


## SuiteCloud IDE Keyboard Shortcuts

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

The following table shows the different keyboard shortcuts you can use within SuiteCloud IDE [NetSuite Perspective](#), depending on the version of SuiteScript for the project. For shortcuts you can use within SuiteCloud IDE [Debug Perspective](#), see [SuiteCloud IDE Debugger](#).

To configure NetSuite shortcut keys, see [Configuring NetSuite Shortcut Keys in Eclipse](#).

| Shortcut      | SuiteScript Version | Description                           |
|---------------|---------------------|---------------------------------------|
| Ctrl+Spacebar | 1.0   2.0           | This is an existing Eclipse shortcut. |

| Shortcut   | SuiteScript Version | Description   |
|--|---------------------|---|
|  |                     | <p>This shortcut is used to open a popup listing of all possible code completion options available in a particular context.</p> <ul style="list-style-type: none"> <li>When multiple record types are involved, you can precede this shortcut with the shortcut <code>&lt;x&gt;+Spacebar+&lt;y&gt;</code> to specify the record context you need.</li> <li>When multiple sublists of a record are involved, you can precede this shortcut with the shortcut <code>&lt;x&gt;+Spacebar+&lt;y&gt;+Spacebar+&lt;z&gt;</code> to specify the record-sublist context you need.</li> </ul>   |
| <code>&lt;x&gt;+Spacebar</code>                              | 1.0                 | <p>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.</p> <p>For example, type <code>s+Spacebar</code> to get all possible code completion options for salesorder fields.</p> <p>This shortcut is used in conjunction with the shortcut <code>Ctrl+Spacebar</code>.</p> <p>For more information, see <a href="#">Automatic Code Completion</a>.</p>  |
| <code>&lt;x&gt;+Spacebar+&lt;y&gt;</code>                    | 1.0                 | <p>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.</p> <p>For example, type <code>s+Spacebar+a</code> to get all possible code completion options for salesorder fields starting with the letter a.</p> <div data-bbox="803 1029 1364 1165"> <p> <b>Note:</b> For internal ID code completion options, append an exclamation point (!) to override and ignore the context when “No Proposal” appears for a particular prefix filter.</p> </div> <p>This shortcut is used in conjunction with the shortcut <code>Ctrl+Spacebar</code>.</p> <p>For more information, see <a href="#">Automatic Code Completion</a>.</p> |
| <code>&lt;x&gt;+Spacebar+&lt;y&gt;+Spacebar+&lt;z&gt;</code> | 1.0                 | <p>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.</p> <p>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.</p> <p>This shortcut is used in conjunction with the shortcut <code>Ctrl+Spacebar</code>.</p> <p>For more information, see <a href="#">Automatic Code Completion</a>.</p>   |
| <code>&lt;xYZ&gt;</code>                                     | 1.0                 | <p>This is an existing Eclipse shortcut.</p> <p>Use this shortcut to get code completion options using camel case patterns for filters where <code>xYZ</code> is your camel case filter for methods and variables.</p> <p>For example, type <code>nLR</code> to get code completion options for methods using the matching camel case pattern, such as <code>nlapiLoadRecord</code>.</p> <p>This shortcut is used in conjunction with the shortcut <code>Ctrl+Spacebar</code>.</p> <p>For more information, see <a href="#">Automatic Code Completion</a>.</p>  |

| Shortcut                                 | SuiteScript Version | Description   |
|--|---------------------|---|
| Ctrl+Alt+A                               | 1.0   2.0           | Use this shortcut to open the Authenticate master Password window and authenticate your master password.  |
| Ctrl+U                                   | 1.0   2.0           | Use this shortcut to upload files from the editor area.   |
| Ctrl+B                                   | 1.0   2.0           | Use this shortcut to log in to a project account from the editor area.  |
| Tab                                      | 1.0   2.0           | Use this shortcut to change function names, record names, variables, and parameters globally. This shortcut applies only to newly created SuiteScript files.  |
| Esc                                      | 1.0   2.0           | Use this shortcut to exit the mode set in conjunction with the shortcut key Tab.  |
| Ctrl+Click <internal ID>                 | 1.0                 | Use this shortcut to launch the Record Browser specific to the internal ID from the editor.   |
| Type /** and press Enter                 | 1.0   2.0           | This is an existing Eclipse shortcut. Use this shortcut to enter a standard JSDoc comment.  |
| Alt+Up Arrow or Alt+Down Arrow           | 1.0   2.0           | This is an existing Eclipse shortcut. Use this shortcut to move the current line/selection up or down.  |
| Ctrl+Alt+Up Arrow or Ctrl+Alt+Down Arrow | 1.0   2.0           | This is an existing Eclipse shortcut. Use this shortcut to duplicate the current line/selection and place the clone above (Up Arrow) or below (Down Arrow) the current line/selection.  |
| Alt+Shift+A                              | 1.0   2.0           | This is an existing Eclipse shortcut. Use this shortcut to toggle block selection mode (formerly column mode). Block selection mode enables you to select a rectangle of text and modify the highlighted text altogether instead of selecting text and modifying it a line at a time. |

## Shorthand for Adding Event Handlers to Code in SuiteCloud IDE

**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 figure shows that if you type `.cpi` and press Ctrl+Spacebar, the following code is added to a Client script:

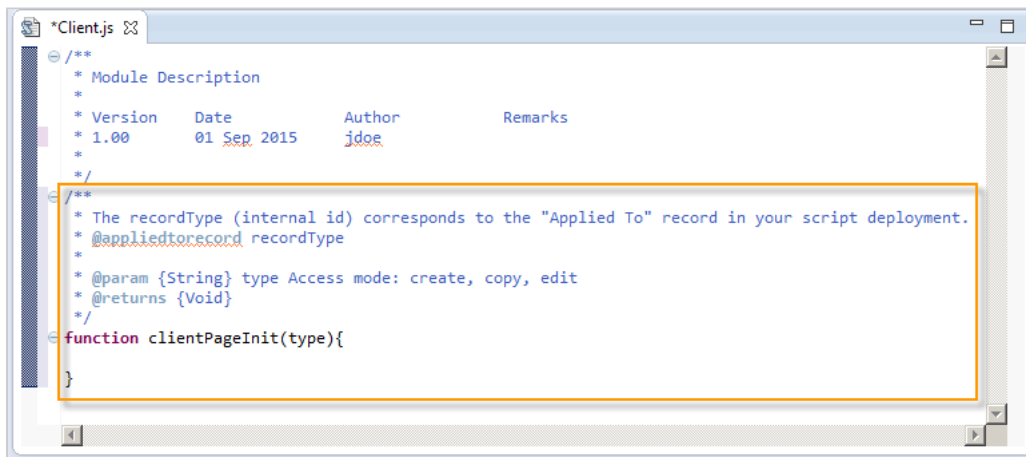
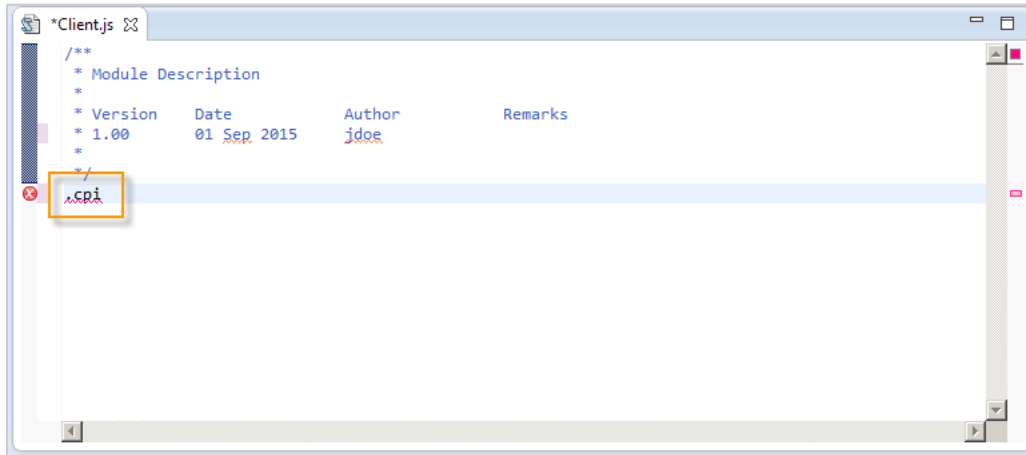
```
/**
 * The recordType (internal id) corresponds to the "Applied To" record in your script deployment.
 *
 * @appliedtorecord recordType
 *
 * @param {String} type Access mode: create, copy, edit
```

```

* @returns {Void}
*/
function clientPageInit(type){

}

```



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

| Script Type / Event Type          | Shorthand Input |
|-----------------------------------|-----------------|
| <b>Bundle Installation script</b> |                 |
| ■ Before Install                  | .bibi           |
| ■ After Install                   | .biai           |
| ■ Before Update                   | .bibu           |
| ■ After Update                    | .biau           |
| <b>Client script</b>              |                 |
| ■ Page Init                       | .cpi            |
| ■ Save Record                     | .csr            |

| Script Type / Event Type | Shorthand Input |
|--------------------------|-----------------|
| ■ Validate Field         | .cvf            |
| ■ Field Changed          | .cfc            |
| ■ Post Sourcing          | .cps            |
| ■ Line Init              | .cli            |
| ■ Validate Line          | .cvl            |
| ■ Recalc                 | .cr             |
| ■ Validate Insert        | .cvi            |
| ■ Validate Delete        | .cvd            |
| <b>RESTlet script</b>    |                 |
| ■ GET                    | .rg             |
| ■ POST                   | .rp             |
| ■ DELETE                 | .rd             |
| ■ PUT                    | .rpu            |
| <b>User Event script</b> |                 |
| ■ Before Load            | .uebl           |
| ■ Before Submit          | .uebs           |
| ■ After Submit           | .ueas           |

## Validation Markers in SuiteCloud IDE

**Applicable SuiteScript versions:**  
SuiteScript 1.0

The table below shows the Editor area validation markers that were augmented for SuiteCloud IDE.

| Validation Marker | Description   |
|-------------------|---|
| Warning           | Augmented to indicate the source location of NetSuite-related code completion warnings.     |
| Error             | Augmented to indicate the source location of NetSuite-related syntax or compilation errors. |



For more information, see the Markers topic in the Eclipse help.

## SuiteCloud IDE Guidelines

**Applicable SuiteScript versions:**  
SuiteScript 1.0

## SuiteScript 2.0

The following table lists general guidelines for working with SuiteCloud IDE.

| Guideline                    | Description  |
|------------------------------|--|
| Use top-down coding method.  | <p>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.</p> <p>For more information about the different ways to maximize code completion, see the help topic <a href="#">SuiteCloud IDE FAQ</a>.</p>   |
| Use camel case filters.      | <p>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>.</p> <p>Go to <code>Window &gt; Preferences &gt; JavaScript &gt; Editor &gt; Content Assist</code> and ensure that the <b>Show camel case matches</b> preference is enabled.</p> <p>For more information about other ways to maximize code completion, see the help topic <a href="#">SuiteCloud IDE FAQ</a>.</p> <div>  <b>Note:</b> You can use camel case filters with SuiteScript 1.0 only.         </div>   |
| Pass variables to functions. | <p>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.</p> <p>For more information about other ways to maximize code completion, see the help topic <a href="#">SuiteCloud IDE FAQ</a>.</p>   |
| Use prefix filters.          | <p>When working with multiple records, enter <code>Ctrl+Spacebar</code> and then use prefix filters to specify the record context you need for code completion. The prefix filter shortcuts are:</p> <ul style="list-style-type: none"> <li>■ <code>&lt;x&gt;+Spacebar</code></li> <li>■ <code>&lt;x&gt;+Spacebar+&lt;y&gt;</code></li> <li>■ <code>&lt;x&gt;+Spacebar+&lt;y&gt;+Spacebar+&lt;z&gt;</code></li> </ul> <p>For more information about prefix filter shortcuts, see <a href="#">SuiteCloud IDE Keyboard Shortcuts</a>.</p> <p>For more information about the different ways to maximize code completion, see the help topic <a href="#">SuiteCloud IDE FAQ</a>.</p> <div>  <b>Note:</b> You can use prefix filters with SuiteScript 1.0 only.         </div> |
| Use standard JSDoc.          | <p>Based on JavaDoc, you can use the standard format of for the following JSDoc comment/tags for a function:</p> <ul style="list-style-type: none"> <li>■ <code>@param:</code> function parameter</li> <li>■ <code>@returns:</code> return value of a function</li> <li>■ <code>@type:</code> variable declarations</li> </ul> <p>Type <code>/**</code> and press <code>Enter</code> to enter a standard JSDoc comment.</p> <p>For more information about the different ways to maximize code completion, see the help topic <a href="#">SuiteCloud IDE FAQ</a>.</p>   |
| Use custom NetSuite JSDoc.   | <p>You can use the custom NetSuite JSDoc tags, <code>@appliedtorecord</code> and <code>@record</code>, to augment code completion.</p> <p>The <code>@appliedtorecord</code> tag corresponds to the <b>Applied To</b> record in your NetSuite script deployment and is applicable only for client and user event script functions that rely on the specified <b>Applied To</b> record.</p> <p>The <code>@record</code> tag can be used in conjunction with standard JSDoc tags such as <code>@param</code>, <code>@returns</code>, and <code>@type</code> to specify the needed record context.</p> <p>For more information about the different ways to maximize code completion, see the help topic <a href="#">SuiteCloud IDE FAQ</a>.</p>  |



| Guideline  | Description  |
|--|--|
| Activate automatic code completion.                              | <p>By default, activate code completion in SuiteCloud IDE with the shortcut Ctrl+Spacebar. However, you can automatically trigger code completion.</p> <p>To enable auto activation in SuiteCloud IDE:</p> <ol style="list-style-type: none"> <li>1. Go to Window &gt; Preferences &gt; JavaScript &gt; Editor &gt; Content Assist.</li> <li>2. Check the <b>Enable auto activation</b> box.</li> <li>3. For <b>Auto activation delay</b>, enter 0.</li> <li>4. For <b>Auto action triggers for JavaScript</b>, enter <b>.nfo</b>.</li> </ol> <p>For more information about auto activation and the different ways to maximize code completion, see the help topic <a href="#">SuiteCloud IDE FAQ</a>.</p> |
| Close unused projects to increase performance.                   | <p>For optimum performance, make sure that the only open project in your SuiteCloud IDE is the project you are currently working on. Always close unused projects.</p> <p>For more information about closing a project, see the Closing projects topic in the Eclipse help.</p>  |
| Use unique function and parameter names.                         | <p>Use unique names for global functions, function names, and parameter names within a single object literal.</p>  |
| Override default memory allocation to increase available memory. | <p>You can increase the memory available for SuiteCloud IDE 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).</p> <p>For more information, see the help topic <a href="#">SuiteCloud IDE FAQ</a>.</p>   |
| Specify your preferred JVM.                                      | <p>You can specify your preferred JVM if multiple copies are available.</p> <p>For more information about working with multiple copies of JVM, see the <a href="#">SuiteCloud IDE FAQ</a>.</p>   |

## Automatic Code Completion

**Applicable SuiteScript versions:**  
SuiteScript 1.0

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.

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



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

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

For information about the different ways to maximize code completion, see the help topic [SuiteCloud IDE FAQ](#).

## Working with SDF Projects

- Creating an Account Customization Project
- Creating a SuiteApp Project
- Importing Account Components into an SDF Project
- Defining SDF SuiteApp Object Dependencies
- Defining Account Component Dependencies from an Account Customization Project
- Defining Feature Dependencies
- Configuring Account Features in an Account Customization Project
- Setting Installation Preferences in a SuiteApp Project
- Validating an SDF Project
- Deploying an SDF Project to Your NetSuite Account

### Creating an Account Customization Project


You can use SDF to create account customization projects, which are intended to customize your NetSuite account.

For more information about account customization projects, see the help topics [Account Customization Projects](#) and [Differences Between SDF Projects](#).

#### To create an account customization project:

1. In Eclipse, go to File > New > SuiteCloud Project.
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**.

SuiteCloud IDE 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 SuiteCloud IDE UI, see [NetSuite Perspective](#).

### Creating a SuiteApp Project


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 [Creating a SuiteApp Project](#) and [Differences Between SDF Projects](#).

#### To create a SuiteApp project:

1. In Eclipse, go to File > New > SuiteCloud Project.
2. Under SDF Project Type, choose **SuiteApp** and then click **Next**.
3. Set the project properties. For more information, see the help topic [SuiteApp Project Properties](#).
4. Click **Finish**.

SuiteCloud IDE 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 SuiteCloud IDE UI, see [NetSuite Perspective](#).

## Importing Account Components into an SDF Project

You can use SDF to import custom NetSuite objects, files, and scripts from your NetSuite 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](#)
- [Importing Files and Scripts into an Account Customization Project](#)
- [Account Component Imports](#)

## Importing Custom NetSuite Objects into an SDF Project

1. Open Eclipse, and create or select the SDF project.  
For more information, see [Creating an Account Customization Project](#) or [Creating a SuiteApp Project](#).
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 NetSuite account, and places them in the **Objects** folder of the SDF project.

## Importing Files and Scripts into an Account Customization Project

1. Open Eclipse, and create or select the account customization project.  
For more information, see [Creating an Account Customization Project](#).
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 NetSuite 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](#).

## Defining SDF SuiteApp Object Dependencies


You can define dependencies on SuiteApp objects that are external to your SDF project, for custom NetSuite objects supported by SDF. For more information about custom object dependencies, see the help topics [SDF SuiteApp Object Dependencies](#) and [Specifying a Custom NetSuite Object Reference](#).

If your project does not contain any references to the object dependency, you must define the dependency manually. See the help topic [Manually Defining SDF SuiteApp Object Dependencies from the SDF Project Manifest](#).

### To automatically define external SuiteApp object dependencies:

1. Expand your project in the **NS Explorer** subtab, and right-click the **manifest.xml** file.
2. Go to 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 references an object called `customrecord_sample` that is not part of the project. During a server-side validation against a NetSuite account, SDF logs any referenced object that is missing from the account.

```
<manifest projecttype="SUITEAPP">
  <publisherid>com.samples</publisherid>
  <projectid>mysuiteapp</projectid>
  <projectname>SDFSAMPLE</projectname>
  <projectversion>1.0.0</projectversion>
  <frameworkversion>2.0</frameworkversion>
  <dependencies>
    <applications>
      <application id="com.samples.mysuiteapp">
        <objects>
          <object>customrecord_sample</object>
        </objects>
      </application>
    </applications>
  </dependencies>
</manifest>
```

## Defining Account Component Dependencies from an Account Customization Project

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 help topics [Account Component Dependencies](#) and [Specifying a Custom NetSuite 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:

1. Open Eclipse and select the account customization project.  
For more information, see [Creating an Account Customization Project](#).
2. Expand your project in the **NS Explorer** subtab, and right-click the **manifest.xml** file.
3. Go to NetSuite > Add Dependency References to Manifest.

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">SERVERSIDESCRIPITING</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

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](#) or [Creating a SuiteApp Project](#).
  2. Expand your project in the **NS Explorer** subtab, and right-click the **manifest.xml** file.
  3. Go to NetSuite > Add Dependency References to Manifest.  
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.

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

## Configuring Account Features in an Account Customization Project

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:

1. Open Eclipse, and create or select the account configuration project.  
For more information, see [Creating an Account Customization Project](#).
2. Expand your project in the **NS Explorer** subtab, and right-click the **AccountConfiguration** folder.
3. Go to NetSuite > Import Configuration.
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:

```
<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:

1. Open Eclipse, and create or select the account configuration project.  
For more information, see [Creating an Account Customization Project](#).
2. Expand your project in the **NS Explorer** subtab, and right-click the **AccountConfiguration** folder.
3. Go to New > File.
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:

```
<features>
</features>
```

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:

```
<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 an 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:

```

<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

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:

- [Create an Installation Preference File](#)
- [Locking Custom Objects in a SuiteApp Project](#)
- [Hiding Files in a SuiteApp Project](#)
- [Applying Content Protection to a Deployment](#)



## Create an Installation Preference File

1. From SuiteCloud IDE, 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 in a SuiteApp Project

To use the locking preference, you must first create a locking.xml file in the InstallationPreferences Folder of a SuiteApp project. See [Create an Installation Preference File](#).

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

For more information about installation preferences, see the help topics [Locking Preferences for Custom Objects](#) and [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. (Optional) When you deploy the project, you must select the Apply Content Protection check box to enable the specifications in the locking.xml file. See [Applying Content Protection to a Deployment](#).

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 :

```
<preference type="LOCKING" defaultAction="UNLOCK">
  <apply action="LOCK">
    <object>myobject12</object> <!-- lock an object -->
  </apply></preference>
```

## Hiding Files 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 [Create an Installation Preference File](#).

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

For more information, see the help topics [Hiding Preferences for Files](#) and [Installation Preferences in a SuiteApp Project](#).

## To hide files:

**Note:** A path element with a value that contains a wildcard requires an `appliesTo` attribute.

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. (Optional) When you deploy the project, you must select the Apply Content Protection check box to enable the specifications in the `hiding.xml` file.

## Example: Hiding a file 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.

```
<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>
```

## Applying Content Protection to a Deployment

The Apply Content Protection check box appears as an option each time you validate or deploy your SuiteApp project. You must enable this option to apply installation preferences that were specified in the `InstallationPreferences` folder of the project. For example, to lock specific objects, or hide files by default when deployed. If you do not want to apply content protection to a deployment instance, clear the Apply Content Protection check box when you deploy the project.

If an installation preference for an element changed from the previous deployment, or is applied for the first time, the log will capture the change.

```
2018-02-02 06:35:43 (PST) Installation started
Info -- Account [(PRODUCTION) MyNetSuiteAccount]
Info -- Publisher ID [com.samples]
Info -- SuiteApp [com.samples.example2 (1.0.1)]
Info -- Framework Version [1.0]
Validate manifest -- Success
Validate deploy file -- Success
Validate script file -- Success
Validate objects -- Success
Validate files -- Success
```

```

Validate folders -- Success
Validate preferences -- Success
Validate flags -- Success
Validate account settings -- Success
Validate Custom Objects against the Account -- Success
Info -- Current installed version [1.0.1]
Begin deployment -- com.samples.example2 (1.0.1)
The LOCKING preference type action for the customrecord_sample object has changed to LOCK.
2018-02-02 06:35:45 (PST) Installation COMPLETE (0 minutes 2 seconds)

```

For more information about installation preferences, see the help topic [Installation Preferences in a SuiteApp Project](#).

## Managing SDF Projects as Compressed Files

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.

**Note:** It is recommended that you use your own revision control system to provide your development team with versioning, change management, and team collaboration.

### To save an SDF project as a zip file:

1. Open Eclipse, and create or select the SDF project.  
For more information, see [Creating an Account Customization Project](#) or [Creating a SuiteApp Project](#).
2. Right-click on project in the **NS Explorer** subtab.
3. Go to NetSuite > Export Project to Zip File.
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](#) or [Creating a SuiteApp Project](#).
2. Close Eclipse.
3. Open the project zip file that you created using the SuiteCloud IDE.
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

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:

1. Open Eclipse, and create or select the SDF project.

For more information, see [Creating an Account Customization Project](#) or [Creating a SuiteApp Project](#).

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, and then click **Validate**.

SDF validates your project against the chosen account. The validation progress log is displayed in the **Console** subtab.

In SuiteCloud IDE, 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

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:

1. Open Eclipse, and create or select the SDF project.

For more information, see [Creating an Account Customization Project](#) or [Creating a SuiteApp Project](#).



**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](#), [Defining SDF SuiteApp Object Dependencies](#), and [Defining Account Component Dependencies from an Account Customization Project](#).

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. 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](#).

8. 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 Custom Objects

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

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

- [Creating an XML Definition](#)
- [Downloading the XML Definition of an Object from Your NetSuite Account](#)
- [Comparing a Custom SuiteCloud Object with a Custom NetSuite Object](#)
- [Overwriting Custom SuiteCloud Objects with Custom NetSuite Objects](#)

For more information, including XML element guidelines, also see the help topic [Custom NetSuite Object Management](#).

## Creating an XML Definition

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](#) or [Creating a SuiteApp Project](#).
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 [Supported Custom NetSuite Objects](#).
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 SuiteCloud IDE preferences. For more information, see [Setting SDF Project Preferences in SuiteCloud IDE](#).

## Modifying an XML Definition in an SDF Project

NetSuite recommends modifying 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](#)
- [Importing Account Components into an SDF Project](#)
- [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 + Space** ) feature in Eclipse.

### To add an element to an XML definition in an SDF project

1. Open Eclipse, and create or select the SDF project.  
For more information, see [Creating an Account Customization Project](#) or [Creating a SuiteApp Project](#).
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](#)
  - [Importing Account Components into an SDF Project](#)
  - [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. 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.

## Downloading the XML Definition of an Object from Your NetSuite Account

You can download the XML definitions of custom NetSuite objects that were created in your NetSuite account. You can preview the downloaded file to learn how a custom object is structured, or manually incorporate the definition in an SDF project.

Alternatively, you can use the SuiteCloud IDE to import XML definitions from your NetSuite account to an SDF project. For more information, see [Importing Account Components into an SDF Project](#).



**Important:** You cannot import locked record types into SDF projects. You must either unlock the record types before you import them, or reference them outside of the SDF project. For more information about locked record types, see the help topics [Locking Preferences for Custom Objects](#) and [Locking Objects in Customization Bundles](#).

### To download an XML definition from your NetSuite account:

1. Log in to your NetSuite account.
2. Open a custom NetSuite object that is supported by SDF.  
For example, go to Customization > Lists, Records, & Fields > Record Types, and select any of the available record types.  
For more information about supported custom NetSuite objects, see the help topic [Supported Custom NetSuite Objects](#).
3. In the **Actions** menu, click **Download XML** or **Download XML with Record Instances**.  
Your browser prompts you to download and save the XML definition of the custom NetSuite object that you opened.  
If you downloaded the XML definition with record instances, the references to the record instances are nested in an instances XML element.

## Comparing a Custom SuiteCloud Object with a Custom NetSuite Object

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.

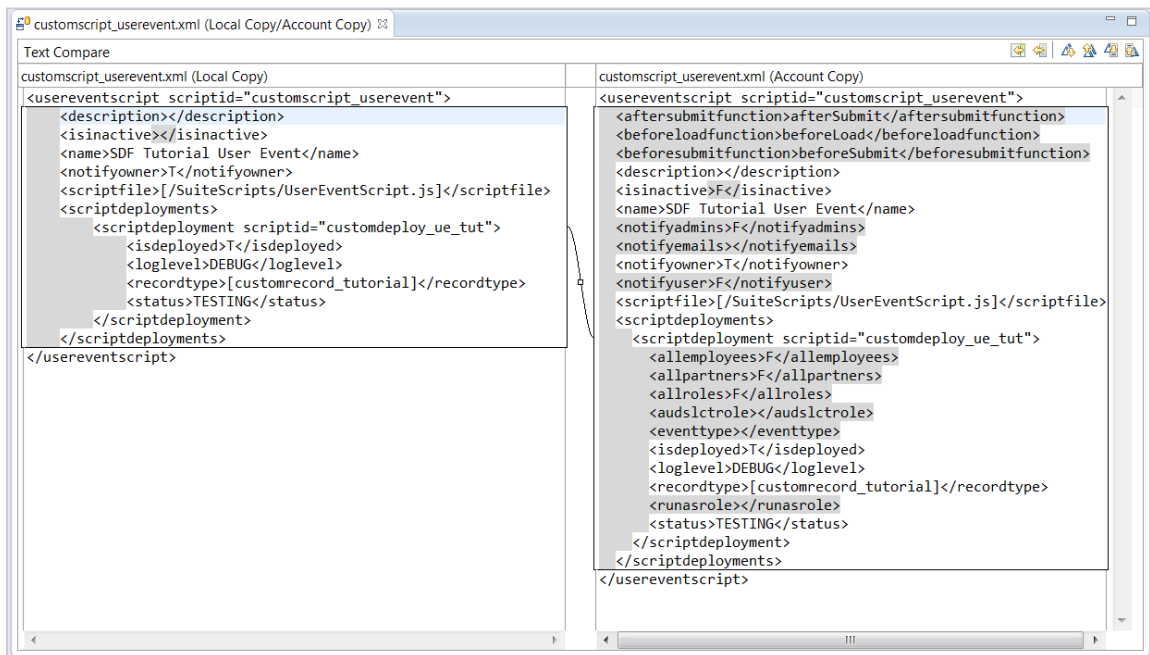
## To compare a custom SuiteCloud object with a custom NetSuite object:

1. Open Eclipse, and create or select the SDF project.  
For more information, see [Creating an Account Customization Project](#) or [Creating a SuiteApp Project](#).
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

**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](#) or [Creating a SuiteApp Project](#).
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.

**Note:** Custom objects that share the same script ID must also share the same custom object type.

## Working with SuiteScript Projects

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](#)
- [Uploading Files in a SuiteScript Project](#)
- [Downloading Files in a SuiteScript Project](#)

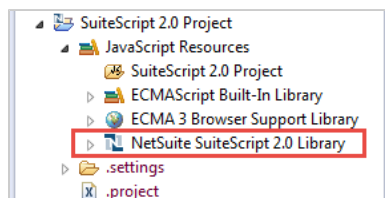
## Creating a SuiteScript Project

Use the following steps to create a SuiteScript project in SuiteCloud IDE.

### To create a SuiteScript project:

1. Launch SuiteCloud IDE.
2. Go to File > New > SuiteCloud Project. The New SuiteCloud Project window opens.
3. Enter a **Project name**.
4. Select the **SuiteScript Version**.

SuiteCloud IDE 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:



You can change the SuiteScript version after you create the project. For more information, see [Changing Project Settings in SuiteCloud IDE](#).

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

5. Choose **SuiteScript Project** as project type.
6. 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.
7. Click **Finish**.

## Uploading Files in a SuiteScript Project

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

**Note:** You can select only one project from which to upload files. If the files you want to upload are in a closed project in SuiteCloud IDE, you must reopen the project before you can download the files. For more information about closing and reopening projects, see the Closing projects topic in the Eclipse help.

### To upload files in a SuiteScript project:

1. Launch SuiteCloud IDE.
2. 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.
3. Select an **Account**.
4. Select a **Role**.
5. Select the **SuiteScript Version**.
6. Browse and select a **File Cabinet Folder** if you want to upload to a different folder from the default.
7. Click **OK**.

## Downloading Files in a SuiteScript Project

You can download files in a SuiteScript project from the NetSuite file cabinet. If you are downloading files that already exist in SuiteCloud IDE, the existing files are backed up prior to download and have 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 SuiteCloud IDE, 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:

1. Launch SuiteCloud IDE.

2. 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.

3. Select an **Account**.
4. Select a **Role**.
5. Select the **SuiteScript Version**.
6. Click **Get File List**.
7. 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.
8. Choose **Use project name in file cabinet**, or choose **Use this project name** and enter the project name that you want to use.
9. Click **OK**.

## Working with SuiteScript Files

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

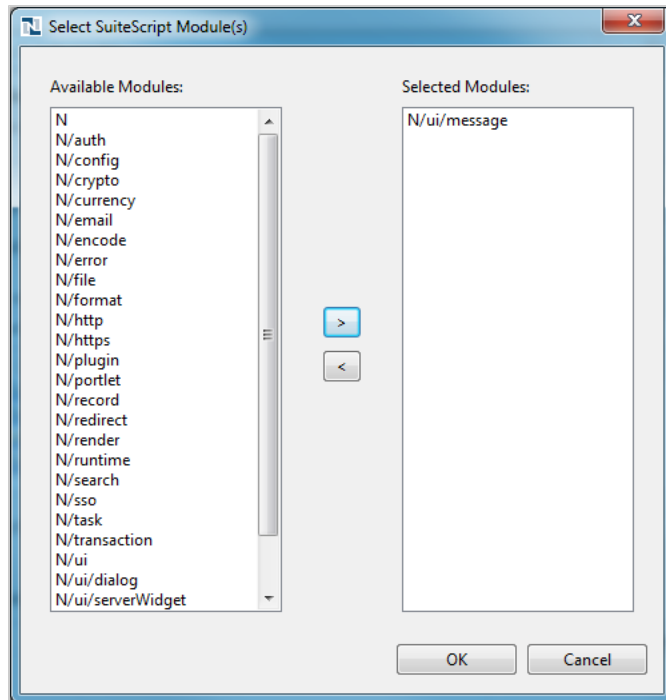
- [Creating a SuiteScript File](#)
- [Uploading a SuiteScript File](#)
- [Defining SuiteScript File and Folder Properties Before Upload](#)
- [Downloading a SuiteScript File](#)
- [Comparing a SuiteScript File with File Cabinet Copy](#)

## Creating a SuiteScript File

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

### To create a SuiteScript file:

1. Launch SuiteCloud IDE.
2. Go to **File > New > SuiteScript File**. The New SuiteScript File window opens.
3. 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](#).
4. Enter or select a parent folder.
5. Change the **Script Filename** if desired.
6. 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.
7. 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.

SuiteCloud IDE 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:

```
/**
 * Module Description
 *
 * Version      Date      Author      Remarks
 * 1.00         18 Oct 2015  jdoe
 */
/**
 * Description.
 *
 * @NApiVersion 2.0
 * @NModuleScope public
 * @NScriptId script_id
 * @NScriptName script name
 */
define(['N/file', 'N/record'],
/**
 * @param {file} file
 * @param {record} record
 */
function(file, record) {

    /**
     * Function to be executed after page is initialized.
    */
}
```

8. 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 the supported script types, see the help topic [SuiteScript 1.0 Script Types Overview](#) for SuiteScript 1.0 or [SuiteScript 2.0 Script Types](#).

## Uploading a SuiteScript File

You can upload a SuiteScript file to the NetSuite file cabinet from a SuiteScript project in your SuiteCloud IDE. 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 SuiteCloud IDE before uploading a file to the file cabinet. See [Defining SuiteScript File and Folder Properties Before Upload](#).

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

### To upload a SuiteScript file:

1. Launch SuiteCloud IDE.
2. 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:

- a. Select an **Account**.
- b. Select a **Role**.
- c. 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 Keyboard Shortcuts](#).

## Defining SuiteScript File and Folder Properties Before Upload

You can define file and folder properties from within SuiteCloud IDE before 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 SuiteCloud IDE.

### To define properties for a selected file:

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:

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


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

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 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 [Downloading Files in a SuiteScript Project](#) and [Downloading Files in an SSP Application Project](#).

### To download a SuiteScript file:

1. Launch SuiteCloud IDE.
2. 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:
  - a. Select an **Account**.
  - b. Select a **Role**.
  - c. Select the **SuiteScript Version**.
  - d. Select a **File Cabinet Folder**.
  - e. Click **OK**.
3. 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

Use the following steps to compare your SuiteScript file with its file cabinet copy.


### To compare a SuiteScript file with file cabinet copy:

1. Launch SuiteCloud IDE.
2. 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

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 ecommerce platforms for these customizations. For more information about SSP applications, see the help topic [SSP Application Overview](#).

 **Important:** The current version of SuiteCloud IDE 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
- Uploading Files in an SSP Application Project
- Downloading Files in an SSP Application Project

## Creating an SSP Application Project

Use the following steps to create an SSP application project in SuiteCloud IDE.

### To create an SSP application project:

1. Launch SuiteCloud IDE.
2. Go to File > New > SuiteCloud Project to open the New SuiteCloud Project window.
3. Enter a **Project Name**.
4. Select the **SuiteScript Version**.


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

5. Select **SSP Application Project** as project type.
6. Check the **Use default location** box, or clear the box and navigate to your desired location.
7. Click **Finish**.

## Uploading Files in an SSP Application Project

You can upload files in an SSP application project to the NetSuite file cabinet. You can select only one project to upload files from.

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

 **Note:** NetSuite recommends that you already have an application record created for your SSP application project. 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 [Creating an SSP Application Record](#).

### To upload files in an SSP application project:


1. Launch SuiteCloud IDE.
2. In SuiteCloud IDE, 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.
3. Select an **Account**.
4. Select a **Role**.
5. Select the **SuiteScript Version**.
6. Browse and select a **File Cabinet Folder**.
7. Click **OK**.



## Downloading Files in an SSP Application Project

You can download files in an SSP application project from the NetSuite 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 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 SuiteCloud IDE, 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:


1. Launch SuiteCloud IDE.
2. 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.
3. Select an **Account**.
4. Select a **Role**.
5. Select the **SuiteScript Version**.
6. Click **Get File List**.
7. 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.
8. Choose **Use project name in file cabinet**, or choose **Use this project name** and enter the project name that you want to use.
9. Click **OK**.

## Changing Project Settings in SuiteCloud IDE

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

### To change project settings:

1. Launch SuiteCloud IDE.
2. Right-click a project in the NS Explorer pane, and select **NetSuite > Change Project Settings** to open the Change Project Settings window.
3. Select an **Account**.
4. Select a **Role**.
5. 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](#) or [Creating an SSP Application Project](#).

6. Select a **File Cabinet Folder**.
7. Click **OK**.

## Converting a Bundle Into an Account Customization Project

1. Create an account customization project.  
For more information, see [Creating an Account Customization Project](#).
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**.  
SuiteCloud IDE 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](#)[Deployment Logs](#).

For more information about converting a bundle, see the help topic [Conversion of a Bundle into an Account Customization Project](#).

## Converting a Project to a SuiteScript Project

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:

1. Launch SuiteCloud IDE.
2. 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

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:

1. Launch SuiteCloud IDE.
2. 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

You can log in to a project account directly from SuiteCloud IDE.



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

### To log in to a project account:

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

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

## SuiteCloud IDE Debugger

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

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

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

- RESTlet
- Suitelet
- User Event
- Workflow Action

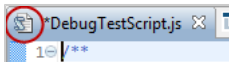
**Note:** The SuiteCloud IDE Debugger does not support client-side debugging. To debug your client SuiteScripts, NetSuite recommends using the Chrome DevTools for Chrome, the Firebug debugger for Firefox, and the Microsoft Script Debugger for Internet Explorer. For additional information about these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

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

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

## Verifying Your Editor

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



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

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

## Working with Breakpoints

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

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

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

For additional information on modifying breakpoints, see:

- [Adding Breakpoints](#)
- [Removing Breakpoints](#)
- [Disabling Breakpoints](#)
- [Enabling Breakpoints](#)

## Adding Breakpoints

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

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

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

## Removing Breakpoints

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

## Disabling Breakpoints

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

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

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

## Enabling Breakpoints

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

## Creating Debug Configurations

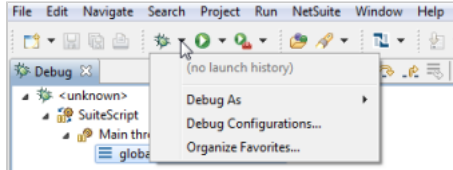
A debug configuration enables you to save the debug settings for a file or project so that you enter them only one time. When your code contains a breakpoint, you must create at least one debug

configuration. You can create multiple debug configurations for each file or project, but debug configurations cannot be shared.

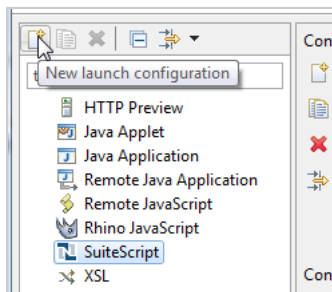
When you create a new debug configuration, a shortcut is added to the debug dropdown menu and the **Debug As** context menu. Use these shortcuts to initiate subsequent debug sessions.

### To create a new debug configuration:

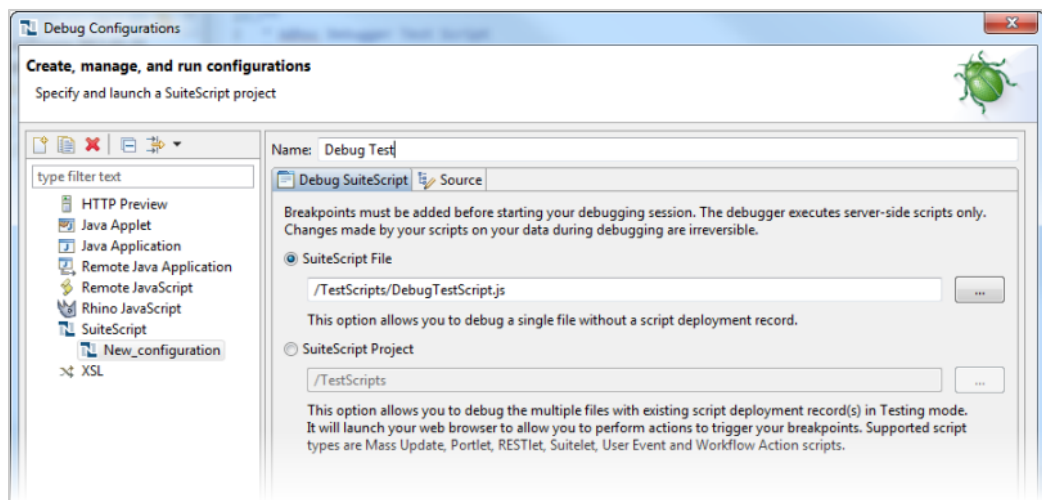
1. Click the debug dropdown menu and select **Debug Configurations**.



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



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



4. Choose one of these options:
  - **SuiteScript File** – Choose this option for Single File Mode (Ad Hoc Debugging).
  - **SuiteScript Project** – Choose this option for Project Mode (Deployed Debugging).
5. If necessary, click ... to change the NS Explorer file path.  
Both the project and file name are required for the **SuiteScript File** option.  
Only the project name is required for the **SuiteScript Project** option, but this project name must match the project name in the NetSuite file cabinet.

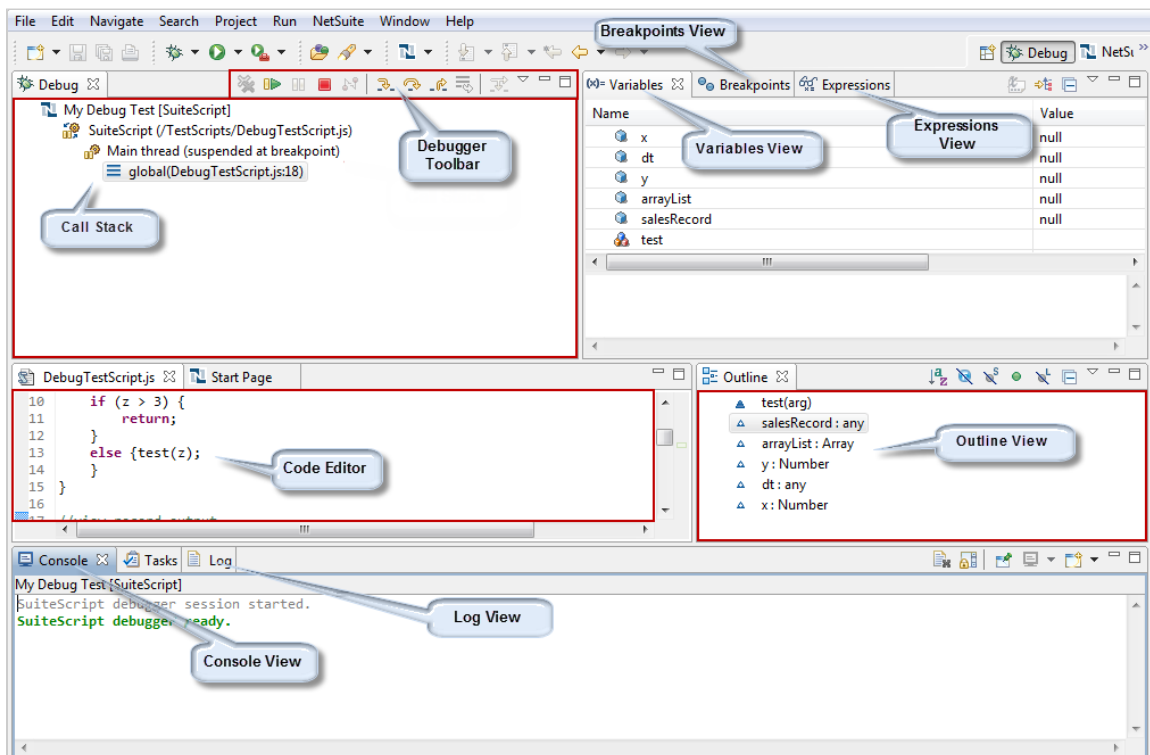
6. Click **Apply**.
7. To debug your code now, click **Debug**.

## Debug Perspective

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

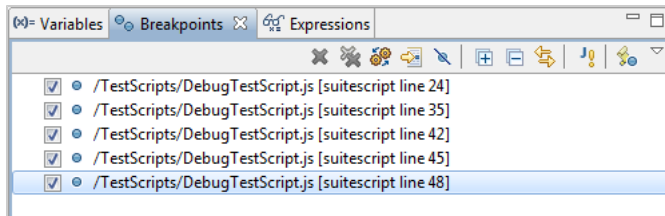
The debug perspective is composed of the following:

- Breakpoints View
- Call Stack
- Code Editor
- Console View
- Debugger Toolbar
- Expressions View
- Log View
- Outline View
- Variables View



## Breakpoints View

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

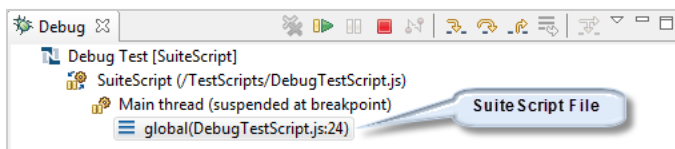


In Project Mode, all breakpoints between files are listed. Within the breakpoints view, you can toggle and remove breakpoints, skip all breakpoints, view the breakpoints for a specific file within your project, and automatically go to specific breakpoints within the [Code Editor](#).

**Note:** Valid breakpoint line numbers are listed as “suitescript line \_\_”; if your line numbers are listed as “line \_\_”, you are using the wrong editor. See [Verifying Your Editor](#) for additional information.

## Call Stack

The call stack shows the files in your SuiteScript.

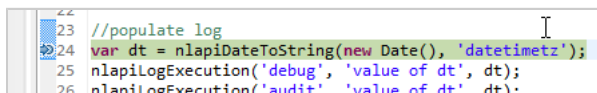


- If you are debugging a single file, the call stack shows that one file.
- If you are debugging a project, each file within the project is displayed. The currently targeted file is highlighted as you progress through the code. When execution is paused, you can switch to a different file by clicking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

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

## Code Editor

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



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

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

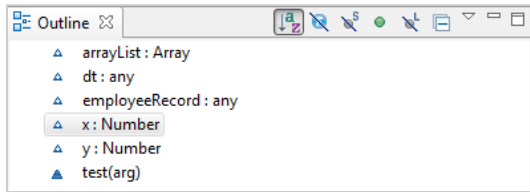
From the code editor, you can also access the context menu options for Run to Line, Watch and Inspect.





## Outline View

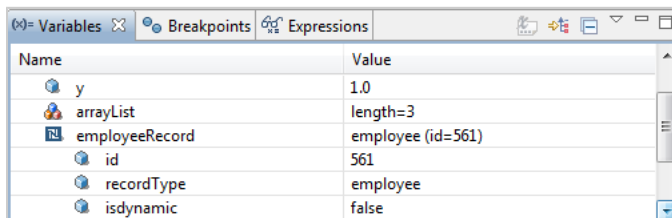
The outline view lists the structural elements of your SuiteScript. During execution, when an element is selected in the [Code Editor](#), it is also selected in the outline view.



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

## Variables View

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





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

## Controlling Execution

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

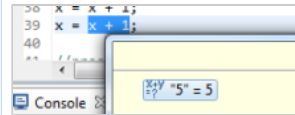
The buttons listed are found on the [Debugger Toolbar](#).

| Functionality     | Button / Menu Option  | Shortcut Key | Description  |
|-------------------|---|--------------|--|
| Toggle Breakpoint | No button<br>See <a href="#">Working with Breakpoints</a> .   | Ctrl+Shift+B | Toggles a breakpoint on the currently selected line when in the <a href="#">Debug Perspective</a> . This shortcut key does not work when in the <a href="#">NetSuite Perspective</a> . |
| Resume            | <br><br>Right-click within the <a href="#">Call Stack</a> and select <b>Resume</b> . | F8           | Resumes execution until the debugger reaches the next breakpoint.  |
| Step Into         |    | F5           | Executes the currently selected line and then advances to the next line without executing it. If the selected line is a function (nlapi) or  |

| Functionality          | Button / Menu Option   | Shortcut Key     | Description   |
|------------------------|--|------------------|---|
|                        | Right-click within the <a href="#">Call Stack</a> and select <b>Step Into</b> .  |                  | method call, the debugger steps into the first line of the function or method body.   |
| Step Over              | <br><br>Right click within the <a href="#">Call Stack</a> and select <b>Step Over</b> .   | F6               | Executes the currently selected line and then advances to the next line without executing it. If the selected line is a function (nlapi) or method call, the debugger executes it without stepping into the function or method body.          |
| Step Return            | <br><br>Right-click within the <a href="#">Call Stack</a> and select <b>Step Return</b> . | F7               | Steps out of the current function or method, and then advances to the next line without executing it.   |
| Run to Line            | No button<br>Right-click within the <a href="#">Code Editor</a> and select <b>Run To Line</b> .  | Ctrl+R           | Advances the debugger to the current position of your cursor. All code preceding the target line is executed. The target line itself is not executed.<br>If the debugger encounters a breakpoint before the target line, it pauses execution. |
| Terminate              | <br><br>Right-click within the <a href="#">Call Stack</a> and select <b>Terminate</b> .   | Ctrl+F2          | Stops execution of the debugger. You must start a new session if you wish to continue debugging.  |
| Relaunch               | No button<br>Right-click within the <a href="#">Call Stack</a> and select <b>Relaunch</b> .  | F11              | Starts a new debug session identical to the last session executed.  |
| Terminate and Relaunch | No button<br>Right-click within the <a href="#">Call Stack</a> and select <b>Terminate and Relaunch</b> .  | Ctrl+Alt+Shift+R | Stops execution of the debugger. Then starts a new debug session identical to the session terminated.   |

## Evaluating Expressions in SuiteCloud IDE

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

| Functionality | Menu Option  | Shortcut Key     | Description   |
|---------------|--|------------------|---|
| Inspect       | Right-click within the <a href="#">Code Editor</a> and select <b>Inspect</b> . | Ctrl+Shift+Alt+I | <p>Evaluates a highlighted expression, variable or object.</p>  <p>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.</p> |

| Functionality | Menu Option  | Shortcut Key     | Description   |
|---------------|--|------------------|---|
| Watch         | Right-click within the <a href="#">Code Editor</a> and select <b>Watch</b> . | Ctrl+Shift+Alt+W | Adds a highlighted expression to the <a href="#">Expressions View</a> . |

## Single File Mode (Ad Hoc Debugging)

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

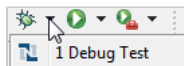
**Note:** Do not log into your NetSuite account from SuiteCloud IDE or your browser for the duration of the debug session. This may cause your debug session to end.

### To debug a single SuiteScript file that is not deployed:

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

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

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



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

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

## Project Mode (Deployed Debugging)

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

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

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

- Mass Update
- Portlet

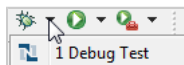
- RESTlet
- Suitelet
- User Event
- Workflow Action

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

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

### To debug a deployed SuiteScript project:

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



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

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

7. Trigger your SuiteScript from within this browser session.

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

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


If you are debugging a RESTlet or Suitelet SuiteScript, see [Using the RESTlet/Suitelet Debug Client](#).

## Using the RESTlet/Suitelet Debug Client


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

### To use the RESTlet/Suitelet client:

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

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

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

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

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

- Select an existing parameter and click **Edit** to edit it. If the parameter is a String, edit the **Name** and **Value** fields and click **OK**. If the parameter is a File, for additional information, see [Adding or Editing a Multipart Body File Parameter](#).

## Adding or Editing a Multipart Body File Parameter

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

### To add or edit a new File parameter:

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

## Launching the SuiteScript Records Browser

You can launch the SuiteScript Records Browser from SuiteCloud IDE. 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](#).

### To launch Records Browser:

1. Launch SuiteCloud IDE.
2. Go to 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 Tips and Guidelines](#).

## Viewing SuiteCloud IDE Error Logs

You can view error logs to help you diagnose or troubleshoot SuiteCloud IDE. Viewing error logs is particularly useful for Support teams.

Another way to view error logs in your SuiteCloud IDE is to go to NetSuite > Troubleshoot > View Log File.

### To view error logs:

1. Launch SuiteCloud IDE.
2. Go to Help > About Eclipse. The About Eclipse window opens.
3. Click **Installation Details**. The SuiteCloud IDE Installation Details window opens.



**Important:** You can see the version of SuiteCloud IDE from this window. The version number is necessary when filing issues for SuiteCloud IDE.

4. Select **SuiteCloud IDE**, and click **Configuration** to populate the Configuration tab for SuiteCloud IDE.
5. Click **View Error Log**. A browser opens with the error log details.