NetSuite Adapter for SAP Integration Suite

Version 1.1.1 – April 2025

THE BEST RUN SAP

Contents

1. Introduction

1.1 Objective

This is the official guide for the NetSuite Adapter for SAP Integration Suite. This guide covers all relevant information for integration developers to start working with the NetSuite adapter. Read this guide carefully before using the Adapter.

1.2 Coding Samples

Any software coding and/or code lines/strings ("Code") included in this documentation are only examples and are not intended to be used in a productive system environment. The Code is only intended to better explain and visualize the syntax and phrasing rules of certain coding. The correctness and completeness of the Code given herein are not guaranteed.

1.3 Internet Hyperlinks

The documentation may contain hyperlinks to the Internet. These hyperlinks are intended to serve as a hint about where to find related information. The availability and the correctness of this related information or the ability of this information to serve a particular purpose are not warranted.

1.4 Overview

NetSuite is a cloud-based enterprise resource planning (ERP) platform that provides businesses with a suite of applications to manage various aspects of their operations. The NetSuite platform is designed to streamline business processes, improve efficiency, and provide real-time insights into various aspects of a company's operations.

The key areas of the NetSuite application and the functionalities it offers are as follows:

- Enterprise Resource Planning (ERP)
- Financial Management
- E-commerce
- Human Capital Management (HCM)
- Professional Services Automation (PSA)

1.5 Features

The NetSuite Adapter provides the following key features:

- ***Support for multiple API versions****:* NetSuite supports multiple API versions with an editable API version field enabling users to input the latest supported version.

- ***Secure authentication based on Tokens:*** The NetSuite adapter supports the Token-Based Authentication (TBA) mechanism which increases overall system security. Every call between NetSuite and SAP Cloud Integration is secured by Tokens.

- ***Support for multiple NetSuite operations****:* The NetSuite adapter supports various operations like Add, Attach, Add List, Get, Get List, Delete, etc.

- ***Dynamic configuration with headers and properties****:* Assigning dynamic values to different properties allows enhanced flexibility to your integration flows. You can also refer to dynamic parameters using SAP Cloud Integration exchange headers and properties.

- ***Processing more than one record at a time****:* Some operations can be performed on one or more new instances of a record in NetSuite. As an example, the **addlist** operation can be used to add one or more new instances of a record to NetSuite.

- ***Processing more than one record of data asynchronously:*** Some operations can be performed on one or more new instances of a record in NetSuite in an asynchronous manner. In general, for asynchronous operations, bulk records are sent to NetSuite without the need to wait for a response.

- ***XSD Generator:*** To help speed up the integration, an XSD generator in the form of an Eclipse Plug-in is provided alongside the adapter. With this Eclipse Plug-in/Workbench, it is possible to generate an XSD which includes the structure of an Account Record. The generated XSD can then be imported into Cloud Integration and used for mapping purposes.

- ***Full integration support for Custom Records and Fields:*** In addition to supporting all the standard NetSuite Records (or entities), the Adapter integrates with all the Custom Objects in NetSuite, providing support for custom fields in the objects as well.

2. Installation and Configuration

This section details the file(s) available as part of the installation package and the prerequisites to configure the NetSuite adapter.

> The NetSuite adapter is available as part of your SAP Integration Suite license.

You would also require an installation of Eclipse Workbench. For more information, see NetSuite Plug-in Installation.

Before you begin, you must download the NetSuite plugin bundle from the SAP Marketplace that contains the file(s) listed in the following section.

2.1 Package Details

The NetSuite plug-in installation package contains the following file:

File Type	File Name	Description
Plug-in File	NetSuite-plug-in_<version>.jar	This file is used to install the plug-in.

2.2 Prerequisites

Before you begin, you must:

- Download and install the NetSuite plugin bundle from the SAP Marketplace.
- Download and install the SAP-recommended version of Eclipse. To get the latest SAP recommendations, see SAP Development Tools.

> It is recommended to use Eclipse Oxygen. However, the plugin remains compatible with all the latest versions of Eclipse.

2.3 NetSuite Adapter Installation on Cloud Foundry

Before the adapter can be used in the Cloud Foundry environment, it must be deployed to the SAP Integration Suite tenant.

To access all adapter variants and features, ensure that your adapter packages are up to date. For more information on how to update your adapter to the latest version, see Updates for SAP's Integration Packages.

2.3.1 Prerequisites

To deploy the NetSuite adapter, you must have access to the *SAP Integration Suite* license.

2.3.2 Procedure

You can deploy the adapter using the following methods:

The below installation procedure is compatible with Apache Camel 2, Apache Camel 3, and the Edge Integration Cell (EIC) platform.

2.3.3 Adapter Installation by creating a New Integration Flow

The NetSuite adapter is available for selection in the receiver adapter list and can be deployed in the **Design** tab directly as you use it in an Integration flow.

Purpose

To install an adapter for use in your Integration flow.

Procedure

Go to **Design** workspace and select the integration package where you want to create a new Integration flow.

1. Click **Edit** to make the package editable.
2. Go to the **Artifacts** tab. Click **Add** and select **Integration Flow**.
3. Enter **Name** and **ID** for your flow. Additionally, select **Runtime Profile** from the drop-down and choose **Sender** and **Receiver** systems from the list. Finally, click **Add** to create the integration flow.
4. Go to the newly created integration flow and click **Edit** to make it editable.
5. In the integration flow, click **End** to add a **Connector** between the **End** and the **Receiver Box**.
   A drop-down with the available adapters appears. The **NetSuite** adapter should show up in the list.

6. Select the **NetSuite** adapter from the list. The adapter is now imported which *triggers* an adapter deployment. Once NetSuite Adapter is deployed, a success message is displayed.
   After the above steps are done, the NetSuite Adapter is successfully deployed in your Design workspace of the SAP Integration Suite tenant.

2.3.4 Adapter Installation without Creating a New Integration Flow

> The following procedure describes how the NetSuite adapter is migrated from the Discover workspace to the Design workspace of the SAP Integration tenant.

This method is useful for scenarios where integration flow packages are migrated from development to a higher environment such as Production.

The NetSuite adapter can be imported into the Design workspace without creating an integration flow. Use the Transport Management Service (TMS) to import/transport the NetSuite adapter to a higher environment. Alternatively, If the TMS is not available in the landscape, the adapter package can be imported to the Design workspace by copying it from the Discover workspace.

Purpose

To copy the integration package from the Discover workspace and import the NetSuite adapter to the Design workspace, follow these steps:

Procedure

1. Go to **Discover** workspace.
2. In the search box, search for the **NetSuite adapter for SAP Integration Suite** package.
3. Select the package and click **Copy**. This copies the package from the Discover workspace to the Design workspace.
4. Go to Design workspace and select the copied **NetSuite adapter for SAP Integration Suite** package.
5. In the **Actions** tab of the selected package, click **Deploy**. This completes the adapter deployment to the Design workspace.

2.4 Monitor the Deployment Status

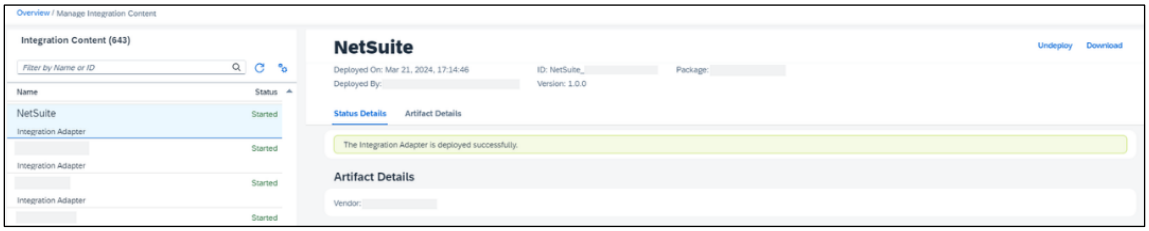
After the adapter deployment is complete, you can check the status in the **Monitor** section.

Purpose

To check the status of the deployed adapter:

Procedure

1. Under the **Monitor** tab, click **Integrations and APIs**. This opens the **Overview** page.
2. On the **Overview** page, go to **Manage Integration Content** section and click **All**. This opens **Integration Content** page with a list of all the deployed adapters.
3. Here, you can check and confirm the deployment status of your adapter.



2.5 NetSuite Plug-in Installation

The Plug-in is applicable only to the SOAP variant.

2.5.1 Prerequisites

Before you begin, you must download the NetSuite Plug-in bundle from the SAP Marketplace. For more information, see NetSuite Installation and Configuration.

2.5.2 Procedure

To install the HubSpot Plug-in, follow these steps:

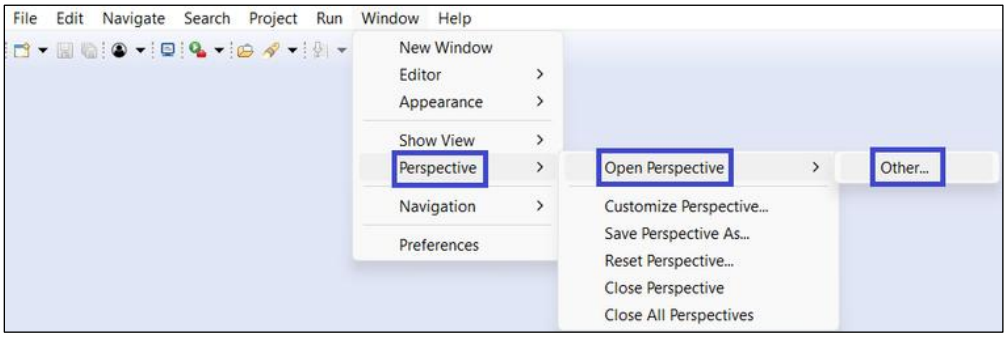
1. Download and install the latest version of Eclipse from the available packages.

   Note that the plugin is compatible with all versions of Eclipse.

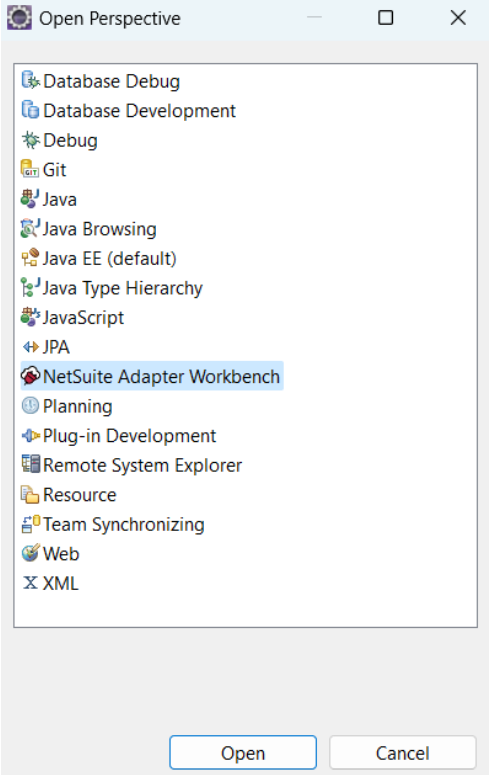
2. After installing Eclipse, navigate to your Eclipse installation folder (*Example: C:\Program Files(x86)\oxygen\eclipse*). If you have installed Eclipse Oxygen, open the Plug-in folder; for any other latest versions, open the drop-in folder.
3. Copy the NetSuite Plug-in JAR file (that you downloaded from the SAP Marketplace) into the Plug-in folder for Eclipse Oxygen or the drop-in folder for other latest versions.

   For more information on the points mentioned above, please refer to the documentation for your installed version of Eclipse.

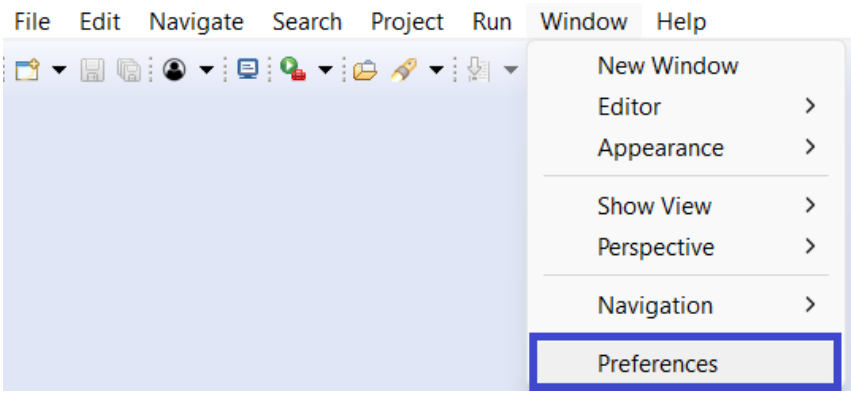
4. Open Eclipse Oxygen and go to **Window** > **Perspective** > **Open Perspective** > **Other**.



5. Select **NetSuite Adapter Workbench** and click **Open**.



6. Go to **Window** > **Preferences** > **NetSuite Preferences.**

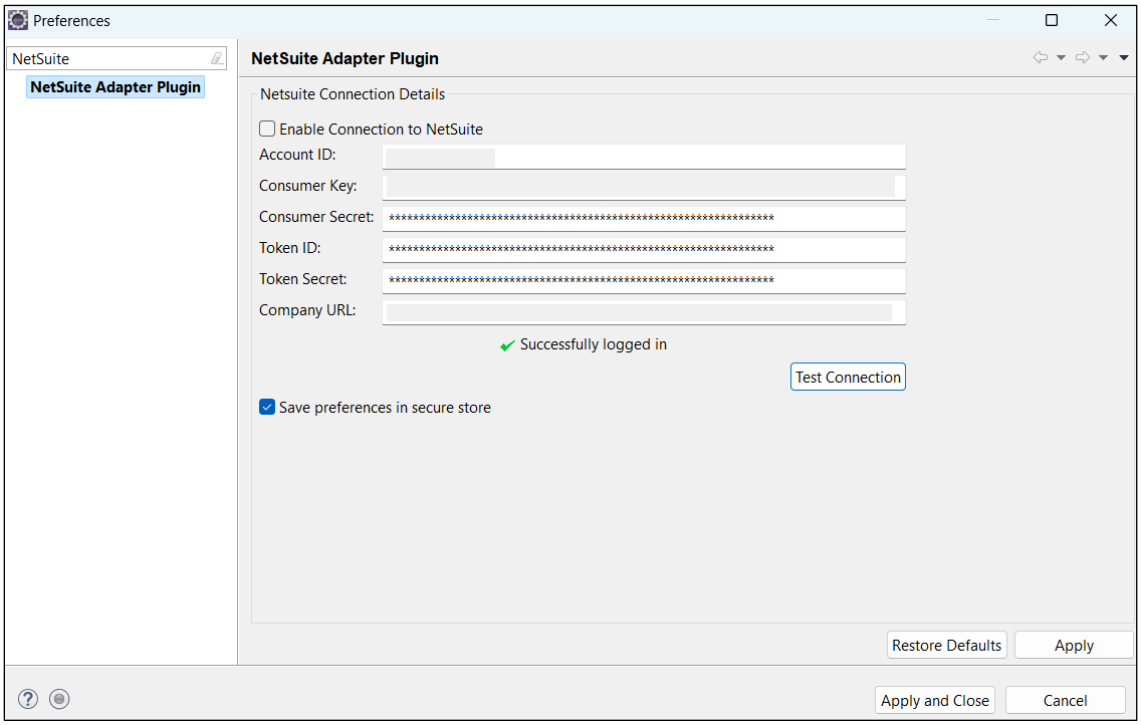


7. In the **NetSuite Preferences** widget, fill in the **NetSuite Settings** details. You can retrieve these details from your NetSuite tenant. To retrieve Settings details and respective login credentials to set up a NetSuite connection to the specific tenant via TBA, see NetSuite Adapter.

> **Enable Connection to NetSuite**: Enable this checkbox to connect to the NetSuite tenant and fetch custom fields from NetSuite while generating XSD for a specific entity.

8. Verify the settings details and the credentials by clicking the **Test Connection** tab.
9. Upon successful test, click **Apply and Close** to complete the installation.

> If you have trouble logging in or using the plugin, refer to the error logs for more details.



3. Getting Started: NetSuite Adapter

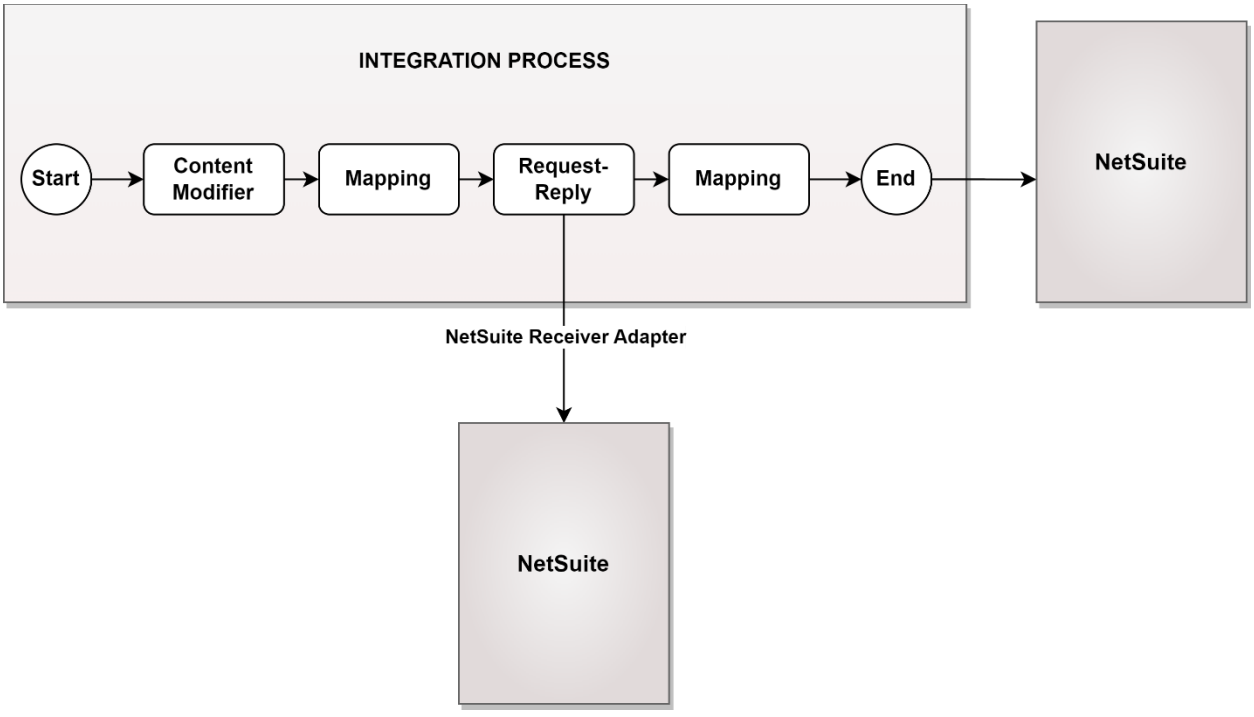
This section explains how to configure the NetSuite adapter for SAP Cloud Integration. You can find information about adapter architecture, application configuration, authentication, and supported versions for NetSuite Adapter.

3.1 Architecture Overview

The NetSuite adapter is designed to function as a receiver adapter. In such a scenario where NetSuite Adapter is used as a receiver adapter, SAP Cloud Integration acts as the initiator of the calls.

The NetSuite adapter supports CRUD operations such as add, get, update, delete, upsert, etc. For more information, see Operations supported in NetSuite.

The figure below demonstrates the request-reply step using a NetSuite adapter and gives a high-level representation of how the adapter works.



The NetSuite adapter works as a *Request-Reply* step and *End* step in the integration process, responsible for connecting and interacting with NetSuite to invoke its operations.

The Eclipse Plug-in is used to help the user generate XSDs of the different entities or resources that are available in your NetSuite tenant.

The XSDs generated by the Eclipse Plug-in can be imported in an Integration Flow and used in mappings.

- For instance, it can be added before the *Request-Reply* Step in the above mapping. This way, you can create the correct XML request message to create or modify a resource in NetSuite (see the Mapping Step on the left side of the Request-Reply Step).
- Similarly, you can use the XSDs to create a mapping to handle the response message returned by NetSuite (see the Mapping Step on the right side of the Request-Reply Step).

Furthermore, the Eclipse Plug-in XSD Generator helps you generate up-to-date XSDs for your NetSuite resources for all the different operations. For more details about the NetSuite plug-in and XSD Generator configuration, see NetSuite Plug-in Installation.

3.2 NetSuite Application Configuration

To create a User Account and generate the user credentials, see NetSuite User credentials. You can connect to NetSuite via the NetSuite adapter using these credentials.

To set up token-based authentication (TBA) in your NetSuite account, you must complete the following tasks.

1. Enable the Token-based Authentication Feature.
2. Set Up Token-based Authentication Roles. For more information about Token Access Management, see Token-based Authentication (TBA) Permissions.
3. Assign Users to Token-based Authentication Roles.
4. Create Integration Records for Applications to Use TBA.
5. Manage TBA Tokens in the NetSuite UI.

For more information, see NetSuite Documentation.

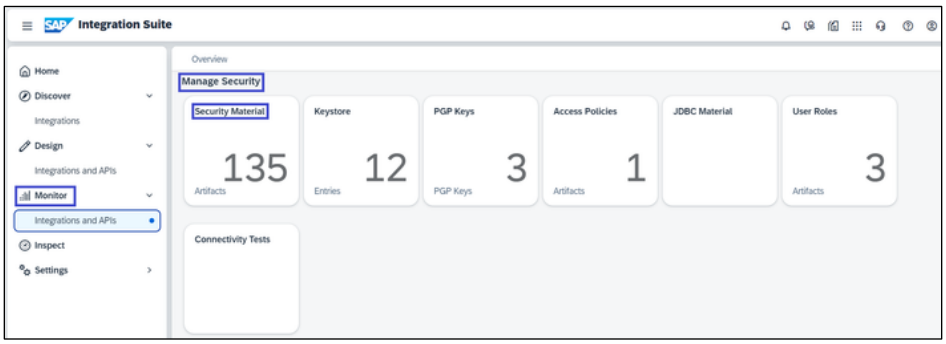
3.3 Authentication

This section details the authentication mechanism supported by the NetSuite Adapter in SAP Cloud Integration. The NetSuite adapter supports the Token-Based Authentication (TBA) mechanism. NetSuite's token-based authentication (TBA) is an authentication mechanism that increases overall system security. TBA enables client applications to use a token to access NetSuite through APIs. Before setting up the authentication, you must create the Credentials in **Security Material** in the SAP Cloud Integration.

3.3.1 Creating Credentials in Security Material

The creation of credentials to support the authentication mechanism can be done by the steps below:

1. In SAP Integration Suite, navigate to **Monitor** > **Integrations and APIs**. This opens the **Overview** page.
2. On the **Overview** page, go to **Manage Security** section and click **Security Material**.



3. On **Manage Security Material** page, click **Create** to select **User Credentials** from the dropdown.

4. In the **Create User Credentials** popup, provide the below details.

Field	Description
Name	Specify the name for the security artifact. The artifact name is used as an alias for the confidential data assigned by this parameter.
Description	Enter a description for the artifact (optional).
Type	Select **User Credentials** to create credentials for NetSuite. This allows you to configure a specific system to enable a connection with your integration flow artifact.
User	Specify the username used to invoke the receiver system.
Password	Specify the password against which the user must be authenticated.

5. Click **Deploy** to complete the process.

When you refresh the **Manage Security Material** page, the new artifact is displayed (with Type **Credentials**) in the artifact table.

You need to create two security artifacts: Consumer Credentials and Token Credentials.

1. Consumer Credentials: Create a User Credential by specifying a name and assigning a Consumer Key as the username and a Consumer Secret as the password. Proceed to deploy these credentials, they will be subsequently utilized within the 'Consumer Credentials Alias' field of your adapter.
2. Token Credentials: Create another set of User Credentials. Assign a name, and provide a Token ID as the username, and a Token Secret as the password. Deploy these credentials accordingly, they will be utilized within the 'Token Credentials Alias' field of your adapter.

4. NetSuite Adapter Configuration

This section describes the parameters to be configured for your NetSuite adapter. You need to configure the **General**, **Connection,** and **Processing** tabs. A description and example usage for every field has been added.

4.1 Receiver Adapter

The Configuration of the NetSuite Receiver Adapter for each one of the supported variants is mentioned below.

4.1.1 SOAP

4.1.1.1 Supported Versions

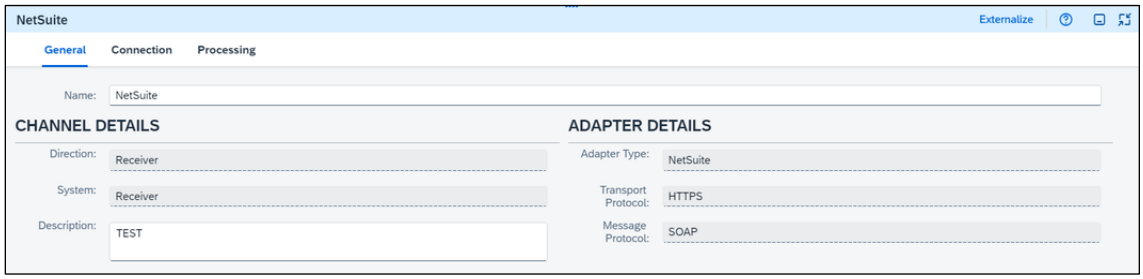
Below are the supported API versions for SOAP variant:

- v2020_1
- v2020_2
- v2021_1
- v2021_2
- v2022_1
- v2022_2
- v2023_1
- v2023_2

> The API version field is editable, allowing users to input the latest supported version.

4.1.1.2 General tab

The General tab provides an overview of basic adapter information including **Channel** and **Adapter** details.



Only the Name and Description fields are editable.

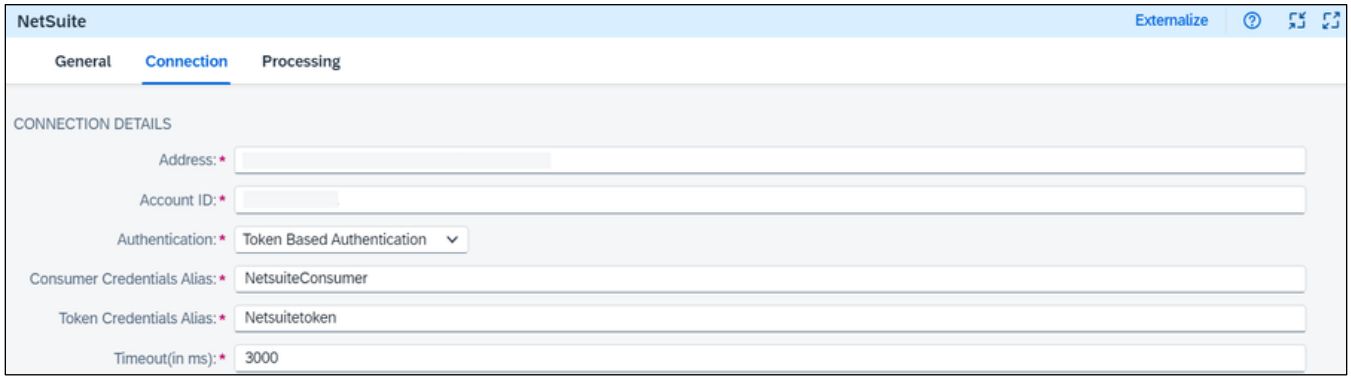
Field	Description
Name	Name of the adapter flow
Description	Description of the adapter

4.1.1.3 Connection

The Connection tab contains connection and authentication parameters for NetSuite.

Using Credentials

The Security artifact created in the Creating Credentials in Security Material should be used in the **Connection tab** of the Adapter as shown below.



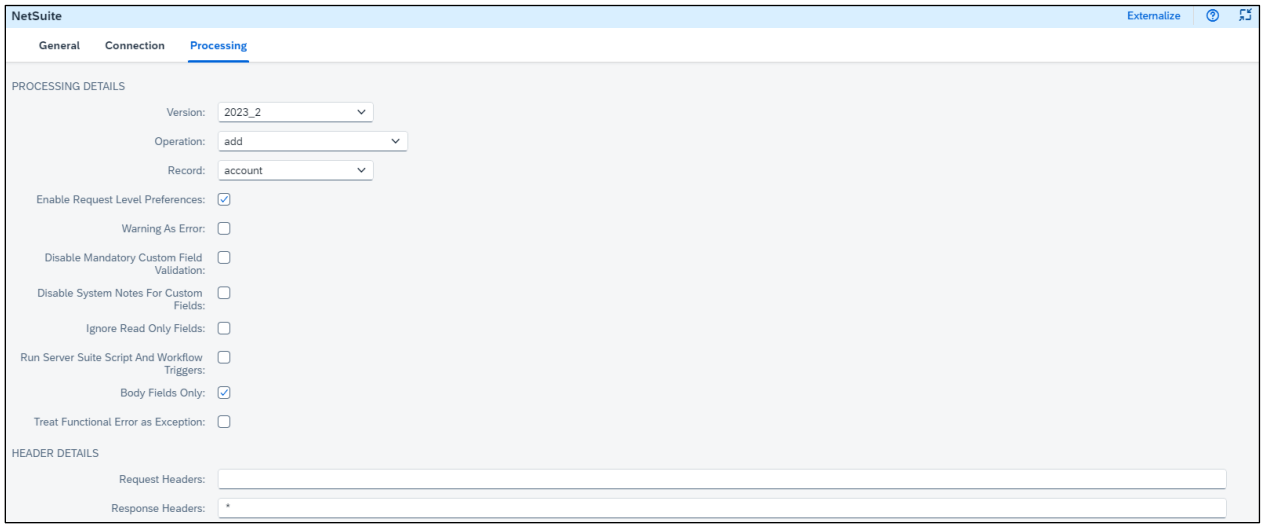
The connection tab contains the following fields:

Field	Description
Address	Specify the address of NetSuite to be used for the connection. This address typically includes your NetSuite Account ID. NetSuite URLs often follow the pattern: https://<accountid>.suitetalk.api.netsuite.com. Example: `https://12345-sb1.suitetalk.api.netsuite.com`
Account ID	Specify the Account ID to be used for the connection. To find the account ID in NetSuite, you can usually locate it in the account's URL when logged in. Additionally, it might be available in the NetSuite account settings or administration section. Example: `1112711_SB1`
Authentication	Select your Authentication Mechanism. Currently, only the Token-Based Authentication (TBA) is supported.
Consumer Credentials Alias	This field should refer to a security material (of type User Credentials). This User Credential should include both Consumer Key (as username) and Consumer Secret (as password). For more information refer to Authentication.
Token Credentials Alias	This field should refer to a security material (of type User Credentials). This User Credential should include both Token ID (as username) and Token Secret (as password). For more information refer to Authentication.
Timeout (in ms)	Specify the maximum waiting time (in milliseconds) while establishing a connection with NetSuite. The default Timeout is 60000 ms.

The adapter validates the server certificate while connecting to NetSuite. In case the root certificates of NetSuite are not present in the SAP Cloud Integration Keystore, then an error will be shown.

4.1.1.4 Processing

The Processing tab lists all the operations that can be performed on the database through the adapter.



Field	Description
Processing Details	
Version	Select the version of the API to be used for this interaction with NetSuite.
Operation	Specify the type of action to be executed in NetSuite to access and exchange data by choosing one of the provided operations. For more information, see Supported Operations.
Record	Specify the NetSuite Object/Entity on which to perform the selected operation. The records are not exhaustive for a particular operation. For the complete set of NetSuite records refer to the latest documentation of NetSuite.

Field	Description
Handle Multiple Record Types	Enable this property if the request expects to perform the list operation on multiple record types in a single call. *Example: For the addList operation, it is possible to add both 'Contact" and "Customer" records within a single request.*
Create Request From Properties	Enable this property to create the body from the properties. This property is applicable only for **Get**/**Delete** operations. After selecting **Create Request From Properties**, additional fields appear as described below.
	Id Type: Select the desired ID type of the record from the dropdown. Example: *InternalID*
	Id Value: Specify the required ID value of the record. Example: *1091144*
Treat Functional Error as Exception	Enable this property to treat a functional error returned by NetSuite as an exception.
Enable Request Level Preferences	Enable this property to specify various request-level preferences.
After selecting **Enable Request Level Preferences**, additional fields appear as described below.	
Disable Mandatory CustomField Validation	Enable this property to change the handling of custom fields that are configured in the UI to be mandatory. When enabled, these fields won't be required during SOAP web services requests. If the property is disabled, they will be mandatory.

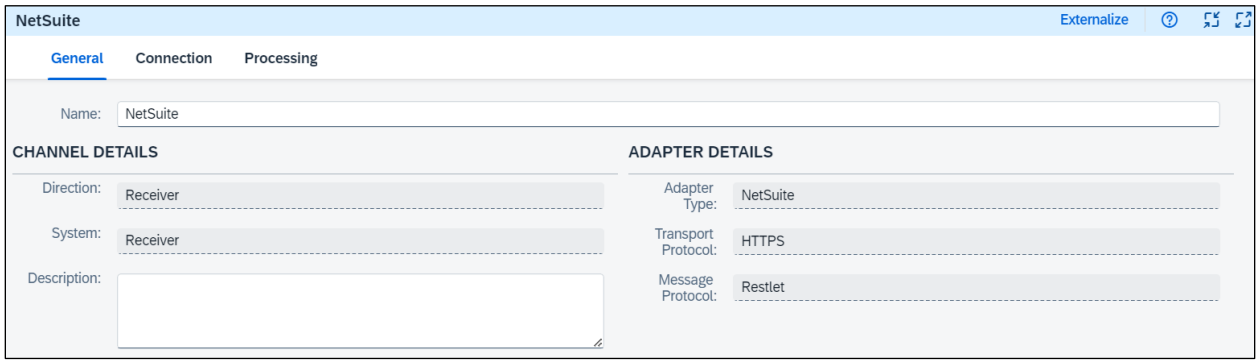
Field	Description
Disable System Notes For CustomFields	Enable this property to prevent the creation of system notes for modifications to custom fields. System notes are automatically generated entries that track changes to a record, including changes to specific field values. Depending on your integration, utilizing this preference could enhance performance.
Ignore ReadOnly Fields	Enable this property to modify the system's behaviour if you mistakenly send a value for a read-only field in your request. If this property is selected, the system ignores these read-only in your request payload.
Warning As Error	Enable to process all NetSuite warning messages as errors. This property changes the handling of custom fields that are configured in the UI to be mandatory. When enabled, these fields won't be required in requests. If the property is disabled, they will be mandatory. If disabled and the required fields aren't provided, the system will return a USER_ERROR, prompting for the missing field value.
Body Fields Only	Enable this property to determine if sublist values should be included in search results. When enabled, only body fields are returned. If disabled, sublist values are also included. Selecting this property can significantly improve performance.
Page Size	Specify the number of records to be returned on a single page for the Search operation. When you set a value for page size, the following limits apply: for Synchronous operations, a minimum of 5 and a maximum of 1000, and for Asynchronous operations, a minimum of 5 and a maximum of 2000.
Return Search Columns	Enable this property to return full records, as opposed to columns. The default value for the preference is true. If this property is enabled, it is also required to specify search return columns, otherwise, the system returns an error.

Field	Description
Run Server SuiteScript And Workflow Triggers	Enable this property to control SuiteScript and trigger workflows per request. If this property is not selected, the company preference set on the SOAP Web Services Preferences page is used. If this property is selected, it overrides the company preference set in the UI.
Header Details	
Request Headers	Enter a list of custom headers, separated by a pipe (|), to send to the target system. By default, no custom headers are sent. Use an asterisk (*) to send all custom headers to the target system. Alternatively, you can dynamically pass on the values by defining a property that includes a list of headers.
Response Headers	Enter a list of headers coming from the target system's response, separated by a pipe (|), to be received in the message. Use an asterisk (*) to receive all the headers from the target system, which is also the default value.

4.1.2 RESTlet

4.1.2.1 General tab

The General tab provides an overview of basic adapter information including **Channel** and **Adapter** details.



Only the Name and Description fields are editable.

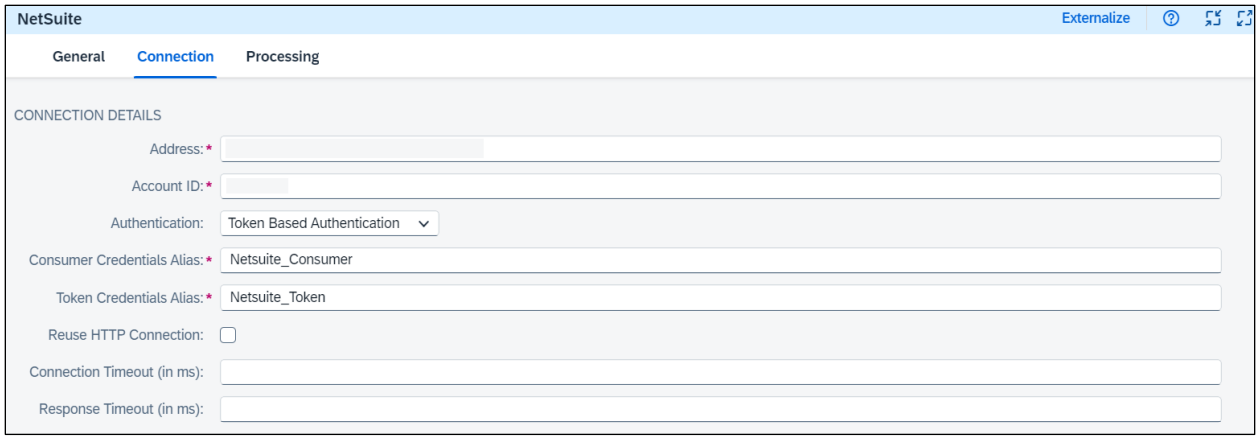
Field	Description
Name	Name of the adapter flow
Description	Description of the adapter

4.1.2.2 Connection

The Connection tab contains connection and authentication parameters for NetSuite.

Using Credentials

The Security artifact created in the Creating Credentials in Security Material should be used in the **Connection tab** of the Adapter as shown below.



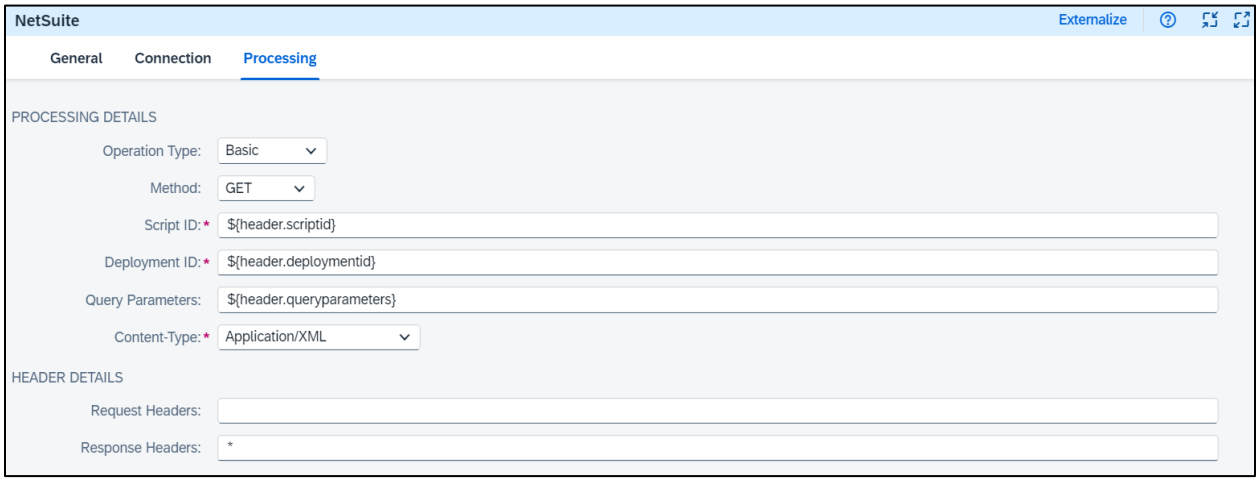
The connection tab includes the following fields.

Field	Description
Address	Specify the address of NetSuite to be used for the connection. This address typically includes your NetSuite Account ID. NetSuite URLs often follow the pattern: https://<accountid>.suitetalk.api.netsuite.com. Example: `https://12345-sb1.suitetalk.api.netsuite.com`
Account ID	Specify the Account ID to be used for the connection. To find the account ID in NetSuite, you can usually locate it in the account's URL when logged in. Additionally, it might be available in the NetSuite account settings or administration section. Example: `1112711_SB1`
Authentication	Select your Authentication Mechanism. Currently, only the Token-Based Authentication (TBA) is supported.
Consumer Credentials Alias	This field should refer to a security material (of type User Credentials). This User Credential should include both Consumer Key (as username) and Consumer Secret (as password). For more information refer to Authentication.
Token Credentials Alias	This field should refer to a security material (of type User Credentials). This User Credential should include both Token ID (as username) and Token Secret (as password). For more information refer to Authentication.

Field	Description
Reuse HTTP Connection	Enable if the connection needs to be reused.
Connection Timeout (in ms)	Specify the maximum waiting time (in milliseconds) for the connection to be established.
Response Timeout (in ms)	Specify the maximum waiting time (in milliseconds) for a response message to be received.

4.1.2.3 Processing

The Processing tab lists all the operations that can be performed on the database through the adapter.



Field	Description
Operation Type	Select the operation type. You can select: • **Basic** to use the dropdowns and parameter text fields. • **Advanced** to provide the relative URI.
Method	Select the required method for the above operation: • POST • PUT • GET • DELETE

Field	Description
Script ID	Specify the ID value of the deployed script.
Deployment ID	Specify the deployment number corresponding to the script.
Relative URI	Specify the relative path. Example: app/site/hosting/restlet.nl?script=7&deploy=1&id=3 The relative path is a part of the request URL after the instance URL and must start with /.
Query Parameters	Specify the comma-separated query parameters that need to be added to the URL. Example: key1=value1,key2=value2
Content-Type	Select the type of content for the RESTlet script. • Application/JSON • Application/XML • Text/PLAIN Note that this field is editable, allowing you to use a content type that is not available in the dropdown.
Header Details	
Request Headers	Enter a list of custom headers, separated by a pipe (|), to send to the target system. By default, no custom headers are sent. Use an asterisk (*) to send all custom headers to the target system. Alternatively, you can dynamically pass on the values by defining a property that includes a list of headers.
Response Headers	Enter a list of headers coming from the target system's response, separated by a pipe (|), to be received in the message. Use an asterisk (*) to receive all the headers from the target system, which is also the default value.

5. NetSuite Adapter Operations

This section lists and describes all the operations supported by both SOAP and RESTlet variant of the NetSuite adapter . The next section contains additional information like troubleshooting tips under Support.

5.1 SOAP

> The below operations require a request message with the details of the record to be executed in NetSuite.
> For more information, see Request/Response message.

5.1.1 add

The add operation can be used to create a new instance of a record within the NetSuite system.

You can set **Operation** as add for a particular API **Version** and select **Record** from any of the NetSuite's various record types such as customers, employees, vendors, transactions, and custom records. For more information about the configuration options, see Processing Tab.

5.1.2 addList

The addList operation is used to add one or more new instances of a record to NetSuite.

You can set **Operation** as addList for a particular API **Version** and select **Record** from any of the NetSuite's various record types such as customers, employees, vendors, transactions, and custom records. For more information about the configuration options, see Processing Tab.

5.1.3 asyncAddList

The asyncAddList operation can be used to asynchronously create one or more new instances of a record to NetSuite.

> The asynchronous responses may not be returned immediately. For more information, see Request Processing.

You can set **Operation** as asyncAddList for a particular API **Version**. For more information about the configuration options, see Processing Tab.

5.1.4 asyncDeleteList

The asyncDeleteList operation can be used to asynchronously delete one or more record instances.

You can set **Operation** as asyncDeleteList for a particular API **Version**. For more information about the configuration options, see Processing Tab.

5.1.5 asyncGetList

The asyncGetList operation can be used to asynchronously retrieve multiple records via a single call.

You can set **Operation** as asyncGetList for a particular API **Version**. For more information about the configuration options, see Processing Tab.

5.1.6 asyncInitializeList

The asyncInitializeList operation can be used to asynchronously populate fields on transaction line items with values from a related record in SOAP web services. Your SOAP web services application can then modify the values before submission.

You can set **Operation** as asyncInitializeList for a particular API **Version**. For more information about the configuration options, see Processing Tab.

5.1.7 asyncSearch

The asyncsearch operation can be used to asynchronously execute a search on a record type based on search filter fields that are specific to that type.

You can set **Operation** as asyncsearch for a particular API **Version**. For more information about the configuration options, see Processing Tab.

5.1.8 asyncUpdateList

The asyncUpdateList operation is used to asynchronously update one or more instances of a record type in NetSuite.

You can set **Operation** as asyncUpdateList for a particular API **Version**. For more information about the configuration options, see Processing Tab.

5.1.9 asyncUpsertList

The asyncUpsertList operation is used to asynchronously add (for new record) or update (for existing record) one or more instances of a record type in NetSuite.

You can set **Operation** as asyncUpsertList for a particular API **Version**. For more information about the configuration options, see Processing Tab.

5.1.10 attach

The attach operation is used to attach a file to or from a record and define a relationship between two records.

You can set **Operation** as attach for a particular API **Version** and select **Record** from any of the NetSuite's various record types such as customers, employees, vendors, transactions, and custom records. For more information about the configuration options, see Processing Tab.

> A user error is thrown if you attempt to attach files or records that do not exist.

5.1.11 changeEmail

The changeEmail operation is used to change a user's email address.

You can set **Operation** as changeEmail for a particular API **Version**. For more information about the configuration options, see Processing Tab.

5.1.12 checkAsyncStatus

The checkAsyncStatus operation is used to check the status of an asynchronous SOAP web services job through a Job ID.

You can set **Operation** as checkAsyncStatus for a particular API **Version**. For more information about the configuration options, see Processing Tab.

5.1.13 Custom

The custom operation allows you to execute a customized request as per the user's requirement utilizing predefined custom records. You are required to map fields in SOAP format to the adapter.

You can set **Operation** as custom for a particular API **Version**. For more information about the configuration options, see Processing Tab.

5.1.14 Delete

The delete operation is used to delete an instance of a record.

You can set **Operation** as delete for a particular API **Version** and select **Record** from any of the NetSuite's various record types such as customers, employees, vendors, transactions, and custom records. For more information about the configuration options, see Processing Tab.

5.1.15 deleteList

The deleteList operation is used to delete one or more record instances.

You can set **Operation** as deleteList for a particular API **Version** and select **Record** from any of the NetSuite's various record types such as customers, employees, vendors, transactions, and custom records. For more information about the configuration options, see Processing Tab.

5.1.16 detach

The detach operation is used to detach a file to or from a record and remove a relationship between two records.

You can set **Operation** as detach for a particular API **Version** and select **Record** from any of the NetSuite's various record types such as customers, employees, vendors, transactions, and custom records. For more information about the configuration options, see Processing Tab.

A user error is thrown if you attempt to detach files or records that do not exist.

5.1.17 get

The get operation can be used to retrieve details of an existing instance of a record within the NetSuite system.

You can set **Operation** as get for a particular API **Version** and select **Record** from any of the NetSuite's various record types such as customers, employees, vendors, transactions, and custom records. For more information about the configuration options, see Processing Tab.

5.1.18 getAccountGovernanceInfo

The getAccountGovernanceInfo operation can be used to retrieve the account concurrency limit and the unallocated concurrency limit through SOAP web services.

> The operation will return results only if you are logged in as an administrator.

You can set **Operation** as getAccountGovernanceInfo for a particular API **Version**. For more information about the configuration options, see Processing Tab.

5.1.19 getAll

The getAll operation can be used to retrieve a list of records of a certain type. Examples of record types can include departments, currencies, etc.

> The getAll operation cannot be performed on all types of records. The operation only supports a restricted type of record.

You can set **Operation** as getAll for a particular API **Version** and select **Record** from any of the NetSuite's various record types such as departments, currencies, etc. For more information about the configuration options, see Processing Tab.

5.1.20 getAsyncResult

The getAsyncResult operation can be used to retrieve details about specific records submitted as part of an asynchronous request.

You can set **Operation** as getAsyncResult for a particular API **Version**. For more information about the configuration options, see Processing Tab.

5.1.21 getBudgetExchangeRate

The getBudgetExchangeRate operation can be used to maintain exchange rates between the root-parent and child subsidiaries for use in the budgeting process.

> This operation can be used only in NetSuite OneWorld accounts.

You can set **Operation** as getBudGetExchangeRate for a particular API **Version**. For more information about the configuration options, see Processing Tab

5.1.22 getCurrencyRate

The getCurrencyRate operation is used to get the exchange rate between two currencies based on a certain date.

You can set **Operation** as getCurrencyRate for a particular API **Version**. For more information about the configuration options, see Processing Tab.

> The Multiple Currencies feature must be enabled in your account before using the getCurrencyRate operation. For information on enabling this feature, see Enabling the Multiple Currencies Feature.

5.1.23 getCustomizationId

The getCustomizationId operation is used to retrieve the internalIds, externalIds, and/or scriptIds of all custom objects of a specified type.

You can set **Operation** as getCustomizationId for a particular API **Version**. For more information about the configuration options, see Processing Tab.

5.1.24 getDataCenterUrls

The getDataCenterUrls operation is used to obtain the correct URL for external client access to NetSuite.

You can set **Operation** as getDataCenterUrls for a particular API **Version**. For more information about the configuration options, see Processing Tab.

5.1.25 getDeleted

The getDeleted operation is used to retrieve a list of deleted records.

You can set **Operation** as getDeleted for a particular API **Version**. For more information about the configuration options, see Processing Tab.

5.1.26 getIntegrationGovernanceInfo

The getIntegrationGovernanceInfo operation is used to retrieve the available concurrency limit for the specific integration using the operation through SOAP web services.

You can set **Operation** as getIntegrationGovernanceInfo for a particular API **Version**. For more information about the configuration options, see Processing Tab.

> The operation will return results only if you are logged in as an administrator.

5.1.27 getItemAvailability

The getItemAvailability operation is used to retrieve the inventory availability for a specific list of items.

You can set **Operation** as getItemAvailability for a particular API **Version**. For more information about the configuration options, see Processing Tab.

> This operation supports up to 10,000 records. If this limit is exceeded, an error is thrown.

5.1.28 getList

The getList operation allows you to retrieve multiple records in a single call. Also, it enables you to specify multiple unique ID numbers in the request message.

You can set **Operation** as getList for a particular API **Version** and select **Record** from any of the NetSuite's various record types such as departments, currencies, etc. For more information about the configuration options, see Processing Tab.

5.1.29 getPostingTransactionSummary

The getPostingTransactionSummary operation allows you to retrieve a summary of the actual data that is posted to the general ledger in an account.

You can set **Operation** as getPostingTransactionSummary for a particular API **Version**. For more information about the configuration options, see Processing Tab.

5.1.30 getSavedSearch

The getSavedSearch operation can be used to retrieve a list of existing saved search IDs on a per-record-type basis (for example, all saved search IDs for every Customer saved search).

You can set **Operation** as getSavedSearch for a particular API **Version**. For more information about the configuration options, see Processing Tab.

5.1.31 getSelectValue

The getSelectValue operation can be used to retrieve valid select options for a particular RecordRef, CustomRecordRef, or enumerated static field.

You can set **Operation** as getSelectValue for a particular API **Version**. For more information about the configuration options, see Processing Tab.

5.1.32 getServerTime

The getServerTime operation can be used for retrieving the current time from the NetSuite server in GMT, irrespective of your time zone.

You can set **Operation** as getServerTime for a particular API **Version**. For more information about the configuration options, see Processing Tab.

5.1.33 initialize

The initialize operation can be used for populating fields on transaction line items with values from a related record in SOAP web services. Your SOAP web services application can then modify the values before submission.

You can set **Operation** as initialize for a particular API **Version**. For more information about the configuration options, see Processing Tab.

5.1.34 initializeList

The initialize List operation involves populating a list of items with values from a related record in SOAP web services. Your SOAP web services application can then modify the values before submission.

You can set **Operation** as initializeList for a particular API **Version**. For more information about the configuration options, see Processing Tab.

5.1.35 search(advanced)

The search(advanced) operation can be used to execute a search on a record type with specified filter fields, **and/or** search return columns or joined search columns. It can also retrieve existing saved searches.

You can set **Operation** as search(advanced) for a particular API **Version** and select **Record** from any of the NetSuite's various record types such as account, contact, etc. For more information about the configuration options, see Processing Tab.

5.1.36 search(basic)

The search(basic) operation can be used to execute a search on a record type based on search filter fields that are specific to that type.

You can set **Operation** as search(basic) for a particular API **Version** and select **Record** from any of the NetSuite's various record types such as account, contact, etc. For more information about the configuration options, see Processing Tab.

5.1.37 searchMoreWithId

The searchMoreWithId operation allows you to reference a specific search result set by its searchId, a parameter included in all search results.

You can set **Operation** as searchMoreWithId for a particular API **Version**. For more information about the configuration options, see Processing Tab.

5.1.38 update

The update operation is used to update an instance of a record in NetSuite.

You can set **Operation** as update for a particular API **Version** and select **Record** from any of the NetSuite's various record types such as file, contact, etc. For more information about the configuration options, see Processing Tab.

5.1.39 updateInviteeStatus

The updateInviteeStatus operation allows you to respond to NetSuite events that you have received. This operation takes both the internal ID of the event as well as a calendar event status as arguments.

You can set **Operation** as updateInviteeStatus for a particular API **Version**. For more information about the configuration options, see Processing Tab.

5.1.40 updateInviteeStatusList

The updateInviteeStatusList operation is used to update one or more NetSuite events.

You can set **Operation** as updateInviteeStatusList for a particular API **Version**. For more information about the configuration options, see Processing Tab.

5.1.41 updateList

The updateList operation is used to update one or more instances of a record type in NetSuite.

You can set **Operation** as updateList a particular API **Version** and select **Record** from any of the NetSuite's various record types such as file, contact, etc. For more information about the configuration options, see Processing Tab.

5.1.42 upsert

The upsert operation is used to add a new instance or to update an instance of a record in NetSuite.

You can set **Operation** as upsert for a particular API **Version** and select **Record** from any of the NetSuite's various record types such as file, contact, etc. For more information about the configuration options, see Processing Tab.

> To prevent duplicate records, use external IDs and upsert/upsertList operation to add records to NetSuite.

5.1.43 upsertList

The upsertList operation is used to add or update one or more instances of a record type in NetSuite.

You can set **Operation** as upsertList for a particular API **Version** and select **Record** from any of the NetSuite's various record types such as file, contact, etc. For more information about the configuration options, see Processing Tab.

> To prevent duplicate records, use external IDs and upsert/upsertList operation to add records to NetSuite.

5.2 RESTlet

> Note that to utilize the RESTlet variant of the adapter, you must create and deploy a script in NetSuite application.

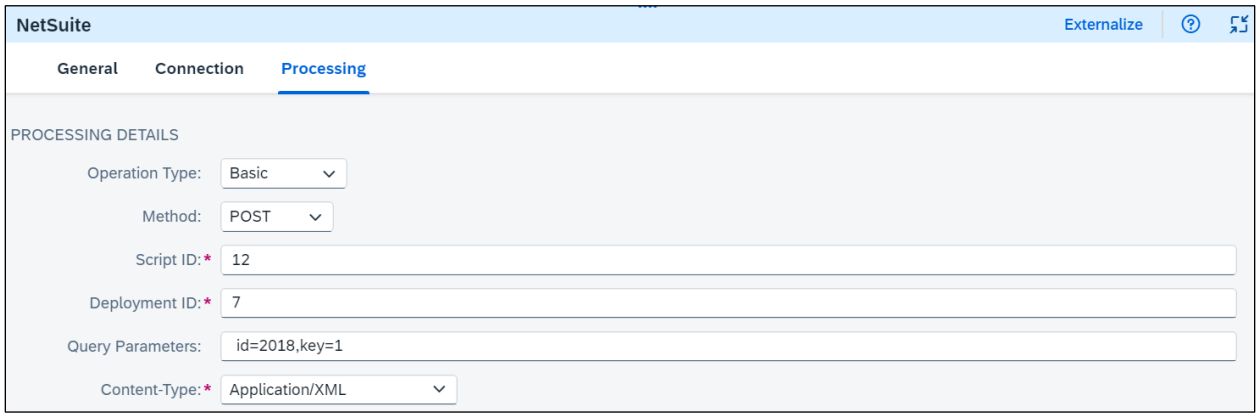
This section provides details on various operations using the basic and advanced operation type.

You can perform various operations using the HTTP method like POST to create a new record instance, GET for retrieving a record type, PUT for updating a record instance and DELETE for deleting a record instance in NetSuite.

5.2.1 Basic Operation Type

You can use the basic operation type and perform the operation by specifying the HTTP method mentioned above. This operation type requires both the script ID and deployment ID to execute the operation.

Below is the sample configuration using the POST method:

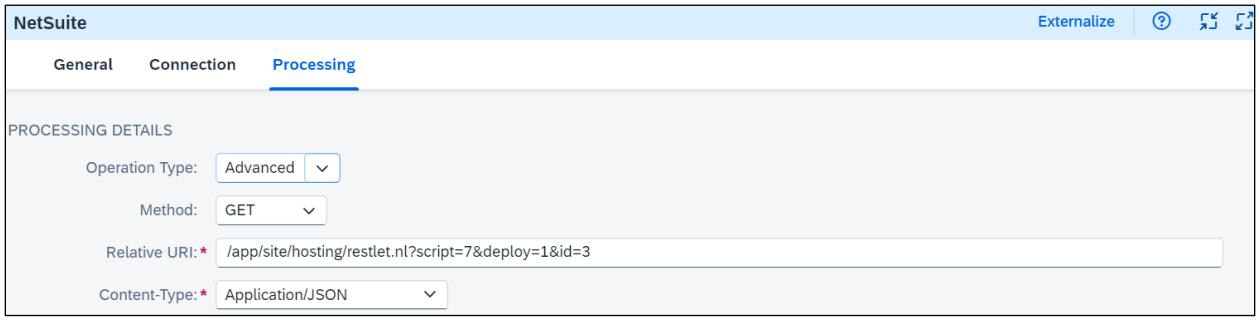


Field	Description
Operation type	Select the operation type as **Basic.**
Method	Select the HTTP method as **POST**.
Script ID	Specify the ID value of the deployed script.
Deployment ID	Specify the deployment number corresponding to the script.
Query Parameters	Specify the comma-separated query parameters that need to be added to the URL. Example: key1=value1,key2=value2
Content-Type	Select the type of content for the RESTlet script. Note that this field is editable, allowing you to use a content type that is not available in the dropdown.

5.2.2 Advanced Operation Type

You can use the advanced operation type and perform the operation by specifying the HTTP method mentioned earlier. This operation type requires a Relative URI to execute the operation.

Below is the sample configuration using the GET method:



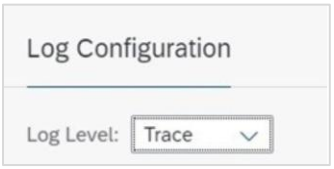
Field	Description
Operation type	Select the operation type as **Advanced** to provide the relative URI and perform the operation according to your requirements.
Method	Select the HTTP method as **GET**.
Relative URI	Specify the relative path. Example: app/site/hosting/restlet.nl?script=7&deploy=1&id=3 The relative path is a part of the request URL after the instance URL and must start with /.
Content-Type	Select the type of content for the RESTlet script. Note that this field is editable, allowing you to use a content type that is not available in the dropdown.

6. Support

6.1 Traces

You can monitor, debug, and analyze errors or issues by changing the **Log level** of your integration flow to **Traces**.

You can enable this under **Log Configuration** under **Monitor** > **Manage Integration Content** page in SAP Cloud Integration.



The Trace Log level enables the collection of more traces to effectively understand the issues. These traces can also be used in a ticket.

6.2 Error Logs for Plugin

For error messages related to the plugin, you can check the error log in the Eclipse plugin.

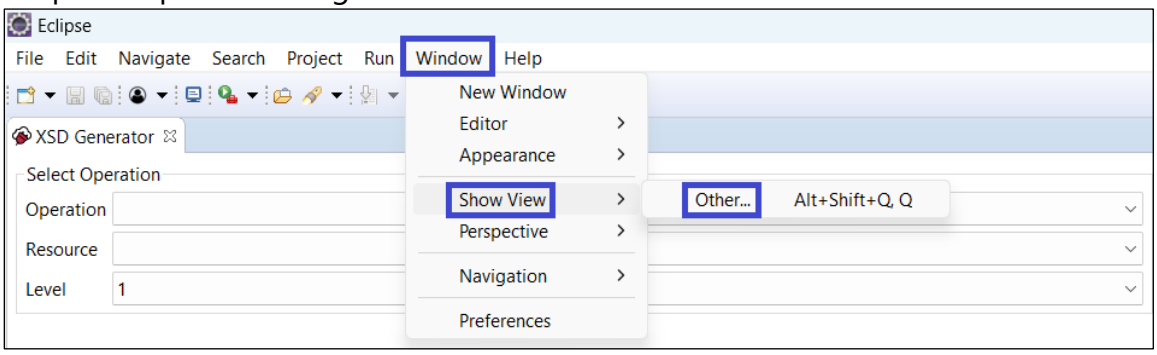
To log plugin errors, please ensure that debug mode is enabled.

Purpose

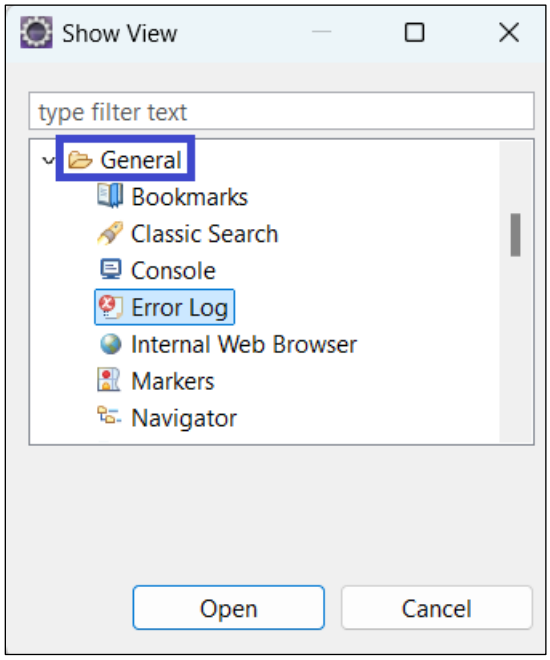
To view the error log.

Procedure

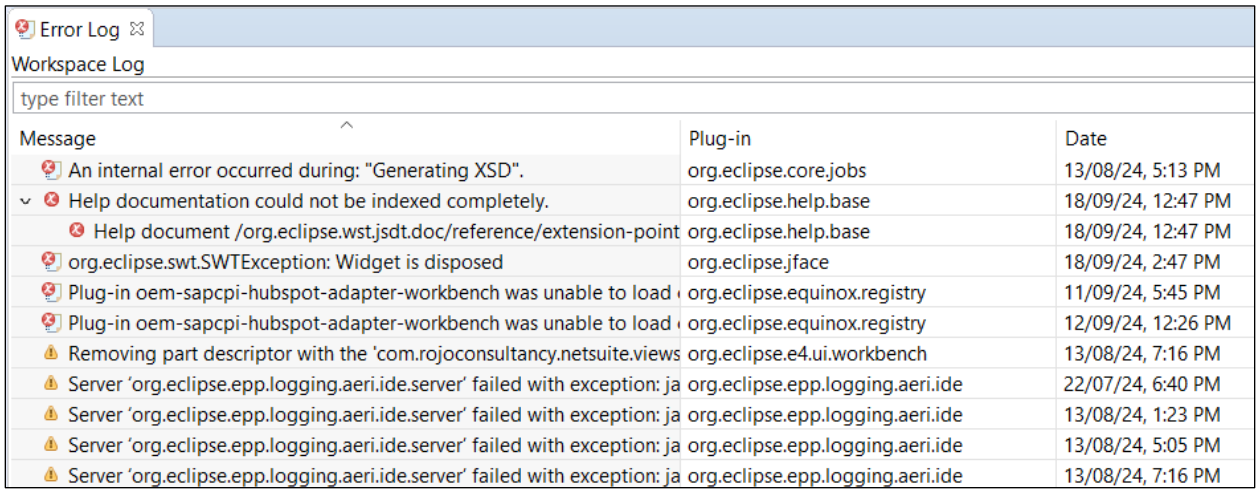
1. Open Eclipse and navigate to **Windows** > **Show View** > **Other**.



2. Select **General** and then choose **Error Log**. Alternatively, you can also search for the error log using the **type filter text** field.



3. Click **Open**.
4. The error log will display the details of error messages if any.



6.3 Troubleshooting

6.3.1 Permission Violation

Error Message:

```
You need a higher level of the "<SOME RECORD NAME>"permission to access this
page. Please contact your account administrator.
```

Possible Solution:

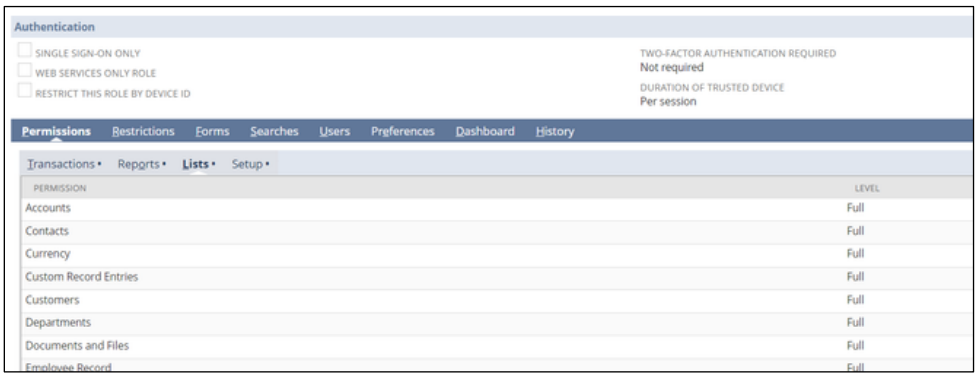
The error indicates that the technical user behind the credentials used in the adapter (Consumer key, Consumer Secret, Token ID, and Token Secret). Missing permissions need to be provided to the users. For that, proceed as follows:

Purpose

To add relevant permissions in NetSuite.

Procedure

1. Log in to NetSuite as an administrator.
2. Then navigate to **Setup** > **Users/Roles** > **Manage Users.**
3. Select the users and you will be taken to a page where the relevant **Permissions** can be added.



6.3.2 Invalid Expression

Error Message:

```
Invalid parameter value <<attributeName>>, should be ${header.xyz} or
${property.xyz}
```

Possible Solution:

The error occurs due to improper expression usage. Ensure that expressions are formatted as ${header.headername} or ${property.propertyname}.

6.3.3 Error Codes

NetSuite can return several error codes, for more information, see Error codes in NetSuite.

7. References

7.1 Nullify a field

When using the update operation, the value of a field can be removed or nullified using one of the following methods:

- Set the value of the field as null (all lowercase). Example:

```
<update>
   <record internalId="217" xsi:type="listEmp:Employee">
        <lastName>null</lastName>
   </record>
</update>
```

- Set an empty value to the field name. Example:

```
<update>
   <record internalId="217" xsi:type="listEmp:Employee">
        <lastName></lastName>
   </record>
</update>
```

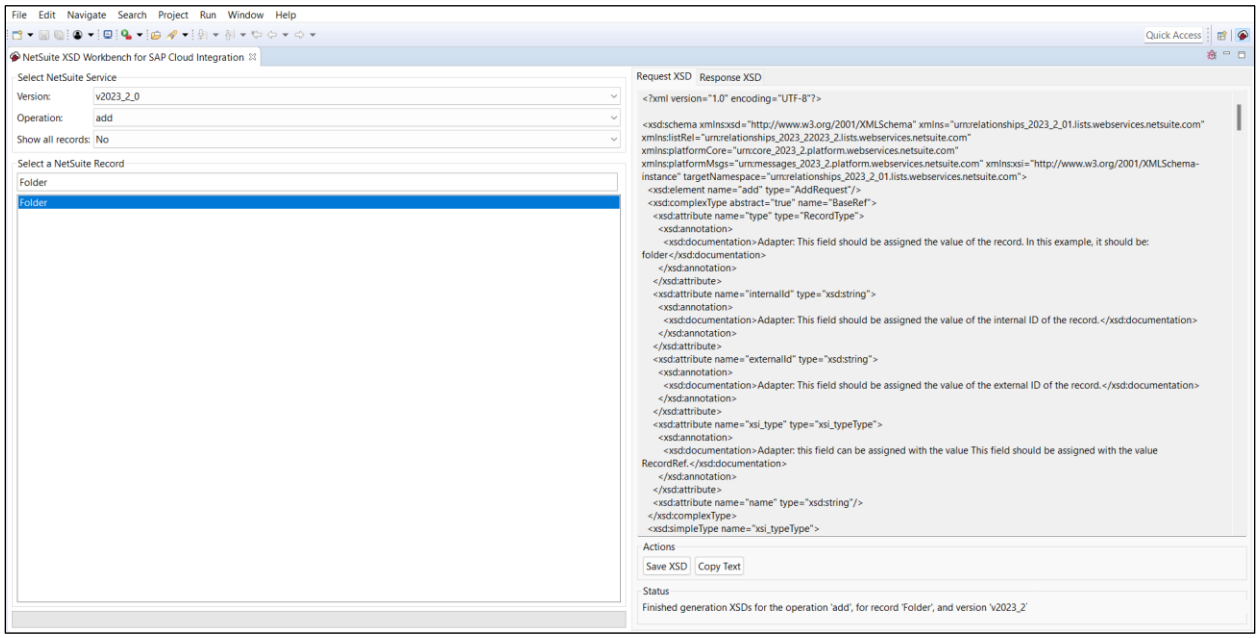
- Send the field name in the nullfield.

7.2 XSD Generator

The XSD Generator is developed to support mapping operations in integration scenarios. It generates request and response XSDs for multiple endpoints that are specific to the NetSuite instance.

In the image below, the right side of the view contains a request and response tab that will show the resulting XSD. If all the required fields are not selected, it indicates which fields need to be selected.

The image below gives an overview of the different XSDs that can be generated.



Purpose

To generate an XSD.

Procedure

1. Open **Eclipse**.
2. Go to **Windows** > **Perspective** > **Open Perspective** > **Other**.
3. Select the **NetSuite Workbench Adapter** and click **Open**.
4. Select **Version** and **Operation** from the respective dropdowns.

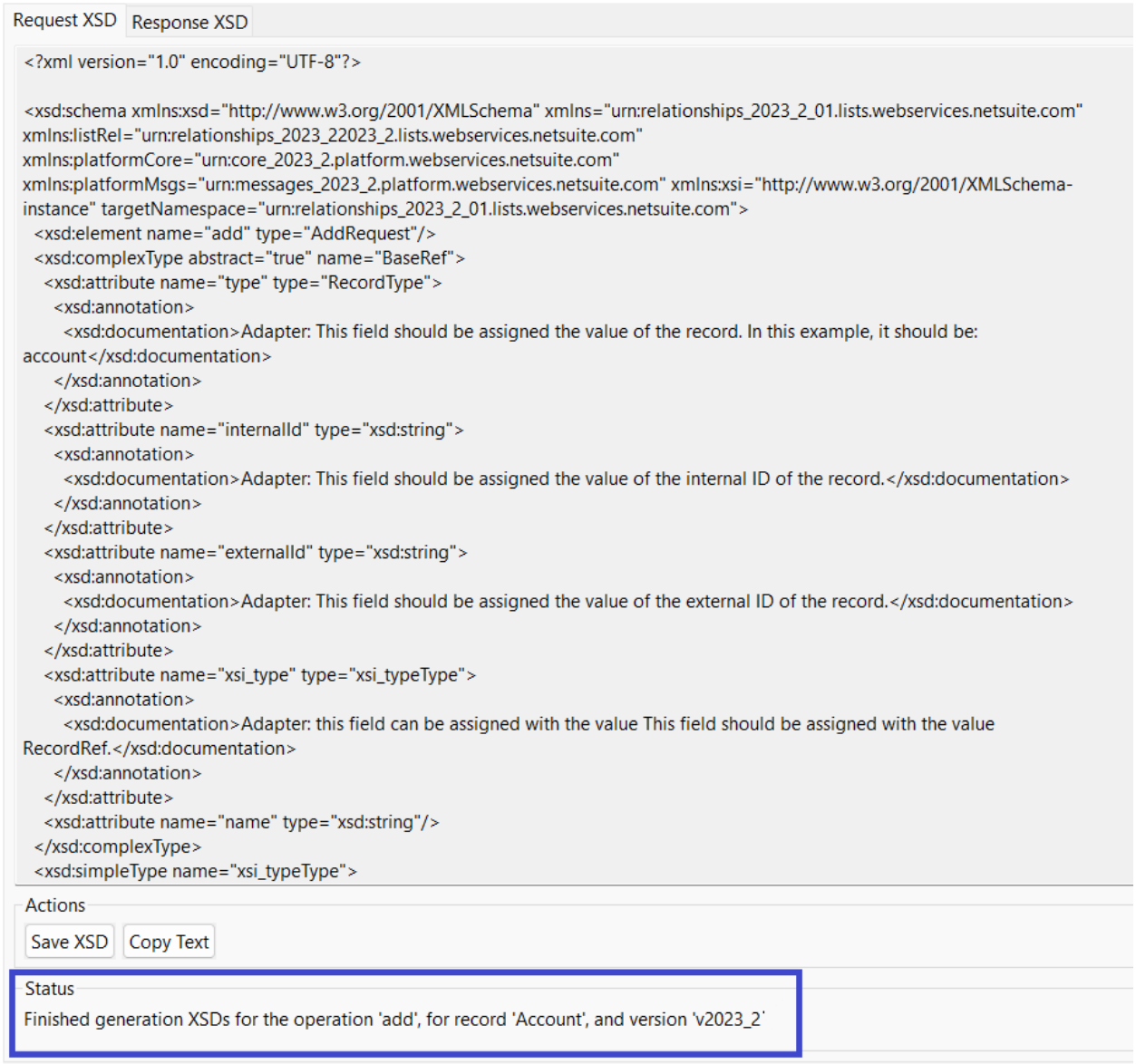
> The plugin's dropdown menu for selecting API versions is not editable. To use the API version that is not available in the dropdown, the XSD generated by the plugin can be adjusted to align with a version not listed in the dropdown menu. For more information, see Using a Custom API Version.

5. Select the record from the **Select a NetSuite record** list. You can search for a particular record within the list as well. Select **Version** and **Operation** from the respective dropdowns.

> **Show all records**: By default, the system displays only supported entities for the specific operation. To view all records, select 'Yes' from the **Show all records** dropdown.

6. Click **Save XSD** to save the displayed XSD. Switch tabs to save the corresponding response/request. Click **Copy Text** to copy the XSD text to your clipboard.

Ensure to verify the status before saving, confirming the successful generation of the XSD. Refer to the below screenshot for reference.



The below table outlines the Request/Response XSDs generated for NetSuite operations specific to a NetSuite object.

Operation	Record Specific	Request	Response
add		X	X
addList		X	X
asyncSearch	X	X	X
asyncAddList		X	X
asyncDeleteList	X	X	X
asyncGetList		X	X
asyncInitializeList	X	X	X
asyncUpdateList		X	X
asyncUpsertList		X	X
attach		X	X
checkAsyncStatus		X	X
custom			
changeEmail	Record Types are not applicable for this operation.	X	X
delete	X	X	X
deleteList	X	X	X
detach		X	X
get		X	X
getAccountGovernanceInfo	Record Types are not applicable for this operation.	X	X
getAll	X	X	X
getAsyncResult	Record Types are not applicable for this operation.	X	X
getBudgetExchangeRate	Record Types are not applicable for this operation.	X	X

Operation	Record Specific	Request	Response
getCurrencyRate	Record Types are not applicable for this operation.	X	X
getCustomizationId		X	X
getDataCenterUrls	Record Types are not applicable for this operation.	X	X
getDeleted	X	X	X
getIntegrationGovernanceInfo		X	X
getItemAvailability		X	X
getList		X	X
getPostingTransactionSummary		X	X
getSavedSearch	Record Types are not applicable for this operation.	X	X
getSelectValue		X	X
getServerTime	Record Types are not applicable for this operation.	X	X
Initialize	X	X	X
initializeList	X	X	X
search(advanced)	X	X	X
search(basic)	X	X	X
searchMoreWithId	X	X	X
update		X	X
updateInviteeStatus		X	X
updateInviteeStatusList		X	X
updateList		X	X
upsert		X	X
upsertList		X	X

7.3 Using a Custom API Version

Purpose

To utilize a custom API version that is not accessible from the dropdown menu:

Procedure

1. Generate the XSD using the closest compatible version available in the dropdown.
2. Manually adjust the XSD to align with the desired version's schema.
3. Identify the discrepancies between the generated XSD and the target version's schema and then make the necessary modifications to ensure compatibility.
4. Once modified, the adjusted XSD can be used with the desired API version that is not available in the dropdown menu.

7.4 Custom fields

When the **Enable Connection to NetSuite** checkbox is checked, the XSD generation process will include fetching custom fields based on the entity type. This data can be utilized in Cloud Integration mapping, enabling users to view the custom fields incorporated in the XSD. For more information, see Custom Fields.

```
<?xml version="1.0" encoding="UTF-8"?>
<ns0:add
xmlns:ns0="urn:relationships_2023_2_01.lists.webservices.netsuite.com">
   <record externalId="Customer1"
               xsi_type="listRel:Customer">
       <firstName>Rajesh</firstName>
       <lastName>Sharma</lastName>
       <companyName>Green Earth Ltd.</companyName>
       <phone>123456789</phone>
       <email>rajesh.sharma@example.com</email>
       <subsidiary internalId="1"/>
       <customFieldListSpec>
           <custentity_listrecord__24__SelectCustomFieldRef>
               <value internalId="105"
                                  typeId="-159"/>
           </custentity_listrecord__24__SelectCustomFieldRef>
           <custentity_freeformtext__28__StringCustomFieldRef>
               <value>freeformtext</value>
           </custentity_freeformtext__28__StringCustomFieldRef>
           <custentity_checkbox__37__BooleanCustomFieldRef>
               <value>true</value>
           </custentity_checkbox__37__BooleanCustomFieldRef>
       </customFieldListSpec>
   </record>
</ns0:add>
```

7.5 Request/Response messages

Below are some requests and response message examples for commonly used operations.

7.5.1 add

> This operation requires a request message with the details of the record to be added or created in NetSuite.

Example Request Message:

```
<?xml version="1.0" encoding="UTF-8"?>
<ns0:add
xmlns:ns0="urn:relationships_2020_1_01.lists.webservices.netsuite.com">
  <record xsi_type="listEmp:Employee">
    <firstName>John</firstName>
    <lastName>Doe</lastName>
    <email>john.doe@email.com</email>
    <currency internalId="1"/>
    <subsidiary internalId="1"/>
    <department internalId="1"/>
  </record>
</ns0:add>
```

The above message creates an employee in NetSuite. XSD containing all fields can be generated in the XSD Generator.

> The attribute "xsi_type" needs to be populated differently depending on the type of records/objects concerned. For more information about possible values, see **Reference Values**.

Example Response Message: Success case

```
<?xml version="1.0" encoding="UTF-8"?>
<ns0:addResponse
xmlns:ns0="urn:relationships_2020_1_01.lists.webservices.netsuite.com">
    <writeResponse>
        <status isSuccess="true">
            <statusDetail>
                <afterSubmitFailed>false</afterSubmitFailed>
            </statusDetail>
        </status>
        <baseRef internalId="310" type="employee"
xsi_type="platformCore:RecordRef"></baseRef>
    </writeResponse>
</ns0:addResponse>
```

Upon a successful call, NetSuite returns a response with the internal id of the newly created NetSuite record. In the above case, an employee with ID number 310 was created.

Example Response Message: Failed case

```
<?xml version="1.0" encoding="UTF-8"?>
<ns0:addResponse
xmlns:ns0="urn:relationships_2020_1_01.lists.webservices.netsuite.com">
    <writeResponse>
        <status isSuccess="false">
            <statusDetail type="ERROR">
                <code>USER_ERROR</code>
                <message>Please enter value(s) for: Default Currency, Email,
Subsidiary, Department</message>
            </statusDetail>
        </status>
    </writeResponse>
</ns0:addResponse>
```

The above example shows an example of a failing request message, due to missing required fields. Note that even though this is a response error; NetSuite will be sending back a 200 HTTP code.

Dealing with Custom fields

In case a custom field needs to be sent as part of your request to NetSuite use the `customFieldList`field. See example below:

```
<?xml version="1.0" encoding="UTF-8"?>
<ns0:add
xmlns:ns0="urn:relationships_2020_1_01.lists.webservices.netsuite.com">
  <record xsi_type="listEmp:Employee">
    <firstName>William</firstName>
    <lastName>Johnson</lastName>
    <email>william.johnson@example.com</email>
    <currency internalId="1"/>
    <subsidiary internalId="1"/>
    <department internalId="1"/>
    <customFieldList>
      <customField internalId="1" scriptId="custentity_notes"
xsi_type="platformCore:StringCustomFieldRef">
        <value>Contract Employee</value>
      </customField>
    </customFieldList>
  </record>
</ns0:add>
```

The element customFieldList holds a collection of customField elements. Each customField contains:

- internalId
- scriptId
- xsi_type: Depending on the data type of the custom field, the value of the xsi_type attribute will be different. The general format is *<prefix>:<record type>*. Example: *tranPurch:VendorBill*. Note that the record type always starts with a capital.
- value: The value to be assigned to the custom field.

Special remarks: NetSuite gives you HTTP code 200 in case of functional issues. But only in case of technical error, a 500 error will be returned.

7.5.2 addList

This operation requires a request message with the details of the record to be added or created in NetSuite.

Example Request Message:

```
<?xml version="1.0" encoding="UTF-8"?>
<ns0:addList
xmlns:ns0="urn:relationships_2020_1_01.lists.webservices.netsuite.com">
   <record xsi_type="listRel:Customer">
       <companyName>ABC Technologies</companyName>
       <phone>0634598711</phone>
       <email>John@abctechnologies.com</email>
   </record>
</ns0:addList>
```

The above message creates an employee in NetSuite. XSD containing all fields can be generated in the XSD Generator.

Example Response Message:

```
<ns0:addListResponse
xmlns:ns0="urn:relationships_2020_1_01.lists.webservices.netsuite.com">
   <writeResponseList>
       <status isSuccess="true"></status>
       <writeResponse>
           <status isSuccess="true">
               <statusDetail>
                   <afterSubmitFailed>false</afterSubmitFailed>
               </statusDetail>
           </status>
           <baseRef type="customer" internalId="27537"
xsi_type="platformCore:RecordRef"></baseRef>
       </writeResponse>
   </writeResponseList>
```

```
</ns0:addListResponse>
```

7.5.3 get

This operation requires a request message with the details of the record to be retrieved in NetSuite.

Example Request Message:

```
<?xml version="1.0" encoding="UTF-8"?>
<ns1:get
xmlns:ns1="urn:relationships_2020_1_01.lists.webservices.netsuite.com">
  <baseRef internalId="121" type="vendorBill"
xsi_type="platformCore:RecordRef"></baseRef>
</ns1:get>
```

The above message creates an employee in NetSuite. XSD containing all fields can be generated in the XSD Generator.

-The attribute “type“ needs to contain the name of the record.
Example: vendorBill. Note that the record needs to start with a lowercase character as a rule of thumb.

-internalId: specify the unique identifier of the record to be retrieved.

-The attribute “xsi_type“ should have the value platformCore:RecordRef.

Example Response Message: Success case

```
<?xml version="1.0" encoding="UTF-8"?>
<getResponse
xmlns="urn:relationships_2020_1_01.lists.webservices.netsuite.com">
   <platformMsgs:readResponse
xmlns:platformMsgs="urn:messages_2020_1.platform.webservices.netsuite.com">
       <platformCore:status
xmlns:platformCore="urn:core_2020_1.platform.webservices.netsuite.com"
isSuccess="true"/>
       <platformMsgs:record
xmlns:tranPurch="urn:purchases_2020_1.transactions.webservices.netsuite.com"
internalId="504" xmlns:xsi="
http://www.w3.org/2001/XMLSchema-instance"
xsi_type="tranPurch:VendorBill">
           <tranPurch:createdDate>2023-12-29T11:21:46.000-
08:00</tranPurch:createdDate>
           <tranPurch:lastModifiedDate>2023-12-29T11:21:46.000-
08:00</tranPurch:lastModifiedDate>
```

```
            <tranPurch:entity
xmlns:platformCore="urn:core_2020_1.platform.webservices.netsuite.com"
internalId="2">
                <platformCore:name>Tax Agency NL</platformCore:name>
            </tranPurch:entity>
            <tranPurch:subsidiary
xmlns:platformCore="urn:core_2020_1.platform.webservices.netsuite.com"
internalId="1">
                <platformCore:name>Test Inc.</platformCore:name>
            </tranPurch:subsidiary>
            <tranPurch:approvalStatus
xmlns:platformCore="urn:core_2020_1.platform.webservices.netsuite.com"
internalId="2">
                <platformCore:name>Approved</platformCore:name>
            </tranPurch:approvalStatus>
            <tranPurch:tranDate>2023-12-28T15:00:00.000-
08:00</tranPurch:tranDate>
            <tranPurch:currencyName>NLD</tranPurch:currencyName>
            <tranPurch:exchangeRate>1.0</tranPurch:exchangeRate>
            <tranPurch:dueDate>2023-12-28T15:00:00.000-
08:00</tranPurch:dueDate>
            <tranPurch:userTotal>100.0</tranPurch:userTotal>
            <tranPurch:taxTotal>0.0</tranPurch:taxTotal>
            <tranPurch:paymentHold>false</tranPurch:paymentHold>
            <tranPurch:memo>triário! Aetatis cómpluribus cura desideráre,
éxquisitaqué inpendente mque quodsi, sapientes utuntur!</tranPurch:memo>
            <tranPurch:currency
xmlns:platformCore="urn:core_2020_1.platform.webservices.netsuite.com"
internalId="1">
                <platformCore:name>NLD</platformCore:name>
            </tranPurch:currency>
            <tranPurch:status>Open</tranPurch:status>

<tranPurch:transactionNumber>VENDBILL14</tranPurch:transactionNumber>
            <tranPurch:expenseList>
                <tranPurch:expense>
                    <tranPurch:line>1</tranPurch:line>
                    <tranPurch:category
xmlns:platformCore="urn:core_2020_1.platform.webservices.netsuite.com"
internalId="1">
                        <platformCore:name>Travel Expenses</platformCore:name>
                    </tranPurch:category>
                    <tranPurch:account
xmlns:platformCore="urn:core_2020_1.platform.webservices.netsuite.com"
internalId="58">
                        <platformCore:name>Expenses</platformCore:name>
                    </tranPurch:account>
                    <tranPurch:amount>100.0</tranPurch:amount>
                    <tranPurch:tax1Amt>0.0</tranPurch:tax1Amt>
                    <tranPurch:grossAmt>100.0</tranPurch:grossAmt>
                    <tranPurch:isBillable>false</tranPurch:isBillable>
                    <tranPurch:taxCode
xmlns:platformCore="urn:core_2020_1.platform.webservices.netsuite.com"
internalId="5">
                        <platformCore:name>VAT:UNDEF-NL</platformCore:name>
                    </tranPurch:taxCode>
                    <tranPurch:taxRate1>0.0</tranPurch:taxRate1>
```

```
                </tranPurch:expense>
            </tranPurch:expenseList>
        </platformMsgs:record>
    </platformMsgs:readResponse>
</getResponse>
```

Upon a successful call, NetSuite returns a response with all details of the NetSuite record.

Example Response Message: Failed case

```
<?xml version="1.0" encoding="UTF-8"?>
<ns2:getResponse
xmlns:ns2="urn:relationships_2020_1_01.lists.webservices.netsuite.com">
    <readResponse>
        <status isSuccess="false">
            <statusDetail type="ERROR">
                <code>RCRD_DSNT_EXIST</code>
                <message>That record does not exist.</message>
            </statusDetail>
        </status>
    </readResponse>
</ns2:getResponse>
```

The above example shows an example of a response in case the requested record does not exist. Note that even though this is a response error; NetSuite will be sending back a 200 HTTP code.

Special remarks: NetSuite gives you HTTP code 200 in case of functional issues. But only in case of technical error, a 500 error will be returned.

7.5.4 Get List

This operation requires a request message with the details of the record to be retrieved in NetSuite.

Example Request Message:

```
<?xml version="1.0" encoding="UTF-8"?>
<ns1:getList
xmlns:ns1="urn:relationships_2020_1_01.lists.webservices.netsuite.com">
  <baseRef internalId="1" type="account"
xsi_type="platformCore:RecordRef"></baseRef>
  <baseRef internalId="2" type="account"
xsi_type="platformCore:RecordRef"></baseRef>
</ns1:getList>
```

The above message creates an employee in NetSuite. XSD containing all fields can be generated in the XSD Generator.

-The attribute “type“ needs to contain the name of the record. Example: account. Note that the record needs to start with a lowercase character as a rule of thumb.

- internalId: specify the unique identifier of the record to be retrieved.

-The attribute “xsi_type“ should have the value platformCore:RecordRef.

Example Response Message: Success case

```
<?xml version="1.0" encoding="UTF-8"?>
<ns2:getListResponse
xmlns:ns2="urn:relationships_2020_1_01.lists.webservices.netsuite.com">
    <readResponseList>
        <status isSuccess="true"></status>
        <readResponse>
            <status isSuccess="true"></status>
            <record internalId="1" xsi_type="listAcct:State">
                <country>_unitedStates</country>
                <fullName>Alaska</fullName>
                <shortname>AK</shortname>
            </record>
        </readResponse>
        <readResponse>
            <status isSuccess="true"></status>
            <record internalId="2" xsi_type="listAcct:State">
                <country>_unitedStates</country>
                <fullName>Arizona</fullName>
                <shortname>AZ</shortname>
            </record>
        </readResponse>
    </readResponseList>
</ns2:getListResponse>
```

Upon a successful call, NetSuite returns a response with all details of the NetSuite record.

Example Response Message: Failed case

```
<?xml version="1.0" encoding="UTF-8"?>
<ns2:getListResponse
xmlns:ns2="urn:relationships_2020_1_01.lists.webservices.netsuite.com">
    <readResponseList>
        <status isSuccess="true"></status>
        <readResponse>
            <status isSuccess="true"></status>
            <record internalId="1" xsi_type="listAcct:State">
                <country>_unitedStates</country>
                <fullName>Alaska</fullName>
                <shortname>AK</shortname>
```

```
            </record>
        </readResponse>
        <readResponse>
            <status isSuccess="true"></status>
            <record internalId="2" xsi_type="listAcct:State">
                <country>_unitedStates</country>
                <fullName>Arizona</fullName>
                <shortname>AZ</shortname>
            </record>
        </readResponse>
    </readResponseList>
</ns2:getListResponse>
```

The above example shows an example of a response in case the requested record does not exist. Note that even though this is a response error; NetSuite will be sending back a 200 HTTP code.

Special remarks: Note that NetSuite gives you HTTP code 200 in case of functional issues. But only in case of technical error, a 500 error will be returned.

7.5.5 Get All

This operation requires a request message with the details of the record to be retrieved in NetSuite.

Example Request Message:

```
<?xml version="1.0" encoding="UTF-8"?>
<ns1:getAll
xmlns:ns1="urn:relationships_2020_1_01.lists.webservices.netsuite.com">
  <record recordType="currency"></record>
</ns1:getAll>
```

The above message retrieves all currencies that are available in NetSuite. The request and response XSDs containing all fields can be generated in the XSD Generator.

The attribute “recordType“ needs to contain the name of the record. Example: currency. Note that the record needs to start with a lowercase character as a rule of thumb.

Example Response Message: Success case

```
<?xml version="1.0" encoding="UTF-8"?>
<ns2:getAllResponse
xmlns:ns2="urn:relationships_2020_1_01.lists.webservices.netsuite.com">
    <getAllResult>
        <status isSuccess="true"></status>
```

```
        <totalRecords>4</totalRecords>
        <recordList>
            <record internalId="1" xsi_type="listAcct:Currency">
                <name>NLD</name>
                <symbol>EUR</symbol>
                <isBaseCurrency>true</isBaseCurrency>
                <isInactive>false</isInactive>
                <overrideCurrencyFormat>false</overrideCurrencyFormat>
                <displaySymbol>€</displaySymbol>
                <symbolPlacement>_beforeNumber</symbolPlacement>
                <locale>_netherlandsDutchEuro</locale>
                <formatSample>€1.234,56</formatSample>
                <exchangeRate>1.0</exchangeRate>
                <currencyPrecision>_two</currencyPrecision>
            </record>
            <record internalId="2" xsi_type="listAcct:Currency">
                <name>US Dollar</name>
                <symbol>USD</symbol>
                <isBaseCurrency>false</isBaseCurrency>
                <isInactive>false</isInactive>
                <overrideCurrencyFormat>false</overrideCurrencyFormat>
                <displaySymbol>$</displaySymbol>
                <symbolPlacement>_beforeNumber</symbolPlacement>
                <locale>_unitedStatesEnglish</locale>
                <formatSample>$1,234.56</formatSample>
                <exchangeRate>1.101</exchangeRate>
                <currencyPrecision>_two</currencyPrecision>
            </record>
            <record internalId="3" xsi_type="listAcct:Currency">
                <name>Canadian Dollar</name>
                <symbol>CAD</symbol>
                <isBaseCurrency>false</isBaseCurrency>
                <isInactive>false</isInactive>
                <overrideCurrencyFormat>false</overrideCurrencyFormat>
                <displaySymbol>$</displaySymbol>
                <symbolPlacement>_beforeNumber</symbolPlacement>
                <locale>_canadaEnglish</locale>
                <formatSample>$1,234.56</formatSample>
                <exchangeRate>1.559</exchangeRate>
                <currencyPrecision>_two</currencyPrecision>
            </record>
            <record internalId="4" xsi_type="listAcct:Currency">
                <name>INR</name>
                <symbol>INR</symbol>
                <isBaseCurrency>false</isBaseCurrency>
                <isInactive>false</isInactive>
                <overrideCurrencyFormat>false</overrideCurrencyFormat>
                <symbolPlacement>_beforeNumber</symbolPlacement>
                <locale>_unitedStatesEnglish</locale>
                <formatSample>$1,234.56</formatSample>
                <exchangeRate>100.0</exchangeRate>
                <currencyPrecision>_two</currencyPrecision>
            </record>
        </recordList>
    </getAllResult>
</ns2:getAllResponse>
```

Upon a successful call, NetSuite returns a response with all details of the NetSuite record.

Example Response Message: Failed case

```
<?xml version="1.0" encoding="UTF-8"?>
<ns2:getAllResponse
xmlns:ns2="urn:relationships_2020_1_01.lists.webservices.netsuite.com">
    <getAllResult>
        <status isSuccess="true"></status>
        <totalRecords>0</totalRecords>
        <recordList></recordList>
    </getAllResult>
</ns2:getAllResponse>
```

The above example shows an example of a response in case the requested record type does not have any records.

In cases where the getAll operation does not support the request record type; an error like one below will be returned:

```
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Fault xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
  <faultcode>soapenv:Server.userException</faultcode>
  <faultstring>org.xml.sax.SAXException: department is not a legal value for
{urn:types.core_2020_1.platform.webservices.netsuite.com}GetAllRecordType</fa
ultstring>
  <detail>
    <ns1:hostname xmlns:ns1="http://xml.apache.org/axis/">partners204.prod-
ams-eu5.core.ns.internal</ns1:hostname>
  </detail>
</soapenv:Fault>
```

Special remarks: Note that NetSuite gives you HTTP code 200 in case of functional issues. But only in case of technical error, a 500 error will be returned.

7.5.6 Search

> This operation requires a request message with the filtering criteria needed to perform the search.

Search (Basic)

```
<ns0:search
xmlns:ns0="urn:relationships_2022_1_01.lists.webservices.netsuite.com">
   <searchRecord xsi_type="listAcct:DepartmentSearch">
       <basic xsi_type="platformCommon:DepartmentSearchBasic">
           <internalId operator="anyOf">
```

```
                <searchValue internalId="3205"/>
            </internalId>
        </basic>
    </searchRecord>
</ns0:search>
```

Search with Join

```
<ns0:search
xmlns:ns0="urn:relationships_2022_1_01.lists.webservices.netsuite.com">
    <searchRecord xsi:type="listRel:ContactSearch">
        <basic xsi:type="platformCommon:ContactSearchBasic">
            <email operator="contains"
                    xsi:type="platformCore:SearchStringField">
                <searchValue>@example.com</searchValue>
            </email>
        </basic>
        <customerJoin xsi:type="platformCommon:CustomerSearchBasic">
            <email operator="contains"
                    xsi:type="platformCore:SearchStringField">
                <searchValue>@abc.com</searchValue>
            </email>
        </customerJoin>
    </searchRecord>
</ns0:search>
```

Search (Advanced)

```
<ns0:search
xmlns:ns0="urn:relationships_2022_1_01.lists.webservices.netsuite.com">
             <searchRecord xsi:type="listEmp:EmployeeSearchAdvanced">
                 <criteria>
                     <basic xsi:type="platformCommon:EmployeeSearchBasic">
                         <email operator="contains">
                             <searchValue>a</searchValue>
                         </email>
                     </basic>
                 </criteria>
                 <columns>
                     <basic xsi:type="platformCommon:EmployeeSearchRowBasic">
                         <email/>
                         <firstName/>
                         <department/>
                     </basic>
                 </columns>
             </searchRecord>
</ns0:search>
```

The above message retrieves all accounts that are available in NetSuite and where the email contains a ".com" in it. The request and response XSDs containing all fields can be generated in the XSD Generator. For information about the Search operators, see Reference values.

-The attribute “xsi_type” (attribute of SearchRecord) needs to be populated differently depending on the type of records/objects concerned. Refer to this record prefix to find the possible values.

-The attribute “xsi_type” (attribute of basic) needs to be assigned with a value in the format platformCommon:<Record>Search. Example: platformCommon:CustomerSearchBasic. Note that the XSD will suggest the correct value.

-The attribute “operator” needs to be assigned with one of the values included in operators.

-The attribute “xsi_type” (attribute of record field) needs to be assigned with one of the values included in search data types.

Upon a successful call, NetSuite returns a response with all details of the NetSuite record.

7.6. Reference Values

7.6.1 Operation Search

Table. 1 Possible values for operator

***Data type* platformCore:SearchDateField**
after
before
empty
notAfter
notBefore
notEmpty
notOn
notOnOrAfter
notOnOrBefore
notWithin
on

Data type **platformCore:SearchDateField**
onOrAfter
onOrBefore
within

Data type **platformCore:SearchDoubleField**
between
empty
equalTo
greaterThan
greaterThanOrEqualTo
lessThan
lessThanOrEqualTo
notBetween
notEmpty
notEqualTo
notGreaterThan
notGreaterThanOrEqualTo
notLessThan
notLessThanOrEqualTo

Data type **platformCore:SearchTextNumberField**
between
empty
equalTo
greaterThan

Data type **platformCore:SearchTextNumberField**
greaterThanOrEqualTo
lessThan
lessThanOrEqualTo
notBetween
notEmpty
notEqualTo
notGreaterThan
notGreaterThanOrEqualTo
notLessThan
notLessThanOrEqualTo

7.6.2 Search Operators

Operator

Table. 2 Possible values for operator

Data type **platformCore:SearchDateField**
after
before
empty
notAfter
notBefore
notEmpty
notOn
notOnOrAfter
notOnOrBefore

Data type **platformCore:SearchDateField**
notWithin
on
onOrAfter
onOrBefore
within

Data type **platformCore:SearchDoubleField**
between
empty
equalTo
greaterThan
greaterThanOrEqualTo
lessThan
lessThanOrEqualTo
notBetween
notEmpty
notEqualTo
notGreaterThan
notGreaterThanOrEqualTo
notLessThan
notLessThanOrEqualTo

Data type **platformCore:SearchTextNumberField**
between
empty

***Data type* platformCore:SearchTextNumberField**
equalTo
greaterThan
greaterThanOrEqualTo
lessThan
lessThanOrEqualTo
notBetween
notEmpty
notEqualTo
notGreaterThan
notGreaterThanOrEqualTo
notLessThan
notLessThanOrEqualTo

7.6.3 Search Data Types

Operator

Table. 3 Possible Data types related to the search operation.

Data type	***Data type and Prefix***
SearchStringField	platformCore:SearchStringField
SearchDateField	platformCore:SearchDateField
SearchBooleanField	platformCore:SearchBooleanField
SearchLongField	platformCore:SearchLongField
SearchTextNumberField	platformCore:SearchTextNumberField
SearchDoubleField	platformCore:SearchDoubleField
SearchEnumMultiSelectField	platformCore:SearchEnumMultiSelectField

Data type	***Data type and Prefix***
SearchMultiSelectField	platformCore:SearchMultiSelectField
SearchCustomField	platformCore:SearchCustomField
SearchStringCustomField	platformCore:SearchStringCustomField
SearchBooleanCustomField	platformCore:SearchBooleanCustomField
SearchLongCustomField	platformCore:SearchLongCustomField
SearchDoubleCustomField	platformCore:SearchDoubleCustomField
SearchDateCustomField	platformCore:SearchDateCustomField
SearchMultiSelectCustomField	platformCore:SearchMultiSelectCustomField
SearchEnumMultiSelectCustomField	platformCore:SearchEnumMultiSelectCustomField
SearchCustomFieldList	platformCore:SearchCustomFieldList