Summer 14

SEED for Test Methods

User Guide

Table of Contents

Introduction

SEED for Test Methods is a native Salesforce.com® application that reduces and potentially eliminates the failure of test method due to changes in field requirements or validation rules, increases productivity when writing test methods, and enhance the developers ability to properly assert for expected values.

Reducing Failures

Four methods are used to properly populate default values for all records created during test method execution when written using SEED for Test Methods.

1. Provides methods the developer can use to pass in field / value pairs for known requirements related to the object or requirements for asserting desired results
2. Provides a single class that the developer creates where fields related to a specific object **AND** record type can be defined. All test methods will then use these values if the developer does not explicitly define them in the test method
3. Default values are automatically calculated based on available field information for required fields such as:
    a. Type
    b. Length, Precision, and Scale
    c. Required
    d. Unique
4. Using organization defined validation rules SEED for Test Methods will attempt to set the values accordingly

    NOTE: SEED for Test Methods handles not all validation rules for, please refer to the validation rules section.

Increase Productivity

SEED for Test Methods requires a single line of code to create a basic record. The developer no longer has to research objects to determine what is required, what validation rules are enforced, or what field specific requirements may be. This drastically reduces the amount of time it takes to create records.

SEED for Test Methods keeps track of what records have been created during the test method. This allows for easy retrieval of the record(s) for assertions. In addition, the ease of record creation allows the developer to focus on setting field

values for specific results and focus their time on asserting that expected results occurred.

Elements Overview

SEED for Test Methods is composed of three distinct elements: Utilities, Developer Tools, and Apex Methods for use when creating test method records.

Utilities

1. User Credentials: Store the credentials of a developer for a specific organization. Multiple credentials can be defined, however, only one can be active at a time.
2. Validation Rule: This object store all of the validation rules identified in the organization
3. Serialize Validation Rules: Allow on demand serialization of all current organization validation rules so they can be utilized by SEED for Test Methods during Apex test execution.

Developer Tools

1. Definition Class Creator: after selecting Object(s), record type(s) and field(s) value(s) the entire class code is presented to the developer. This class code is what SEED for Test Methods will use to populate field values in the case where the developer does not explicitly define the, during the test method.

2. Create Objects Code Generator: allows a developer to specify options including objects, desired method of object record creation and object record types. After selection of these parameters SEED for Test Methods will provide the developer with the code required to create a record with the given parameters taking into account required fields for both the object and the record type layout. This code can be copied and pasted into the test method code.

Apex Method

1. ScheduleDispatcher: Schedulable class that will serialize validation rules on a schedule set by the developer

2. Obj_Creation: The class to be instantiated during test methods utilize global methods allowing the developer to interact with all available methods when writing test methods

3. RecordTypeDefs: created by the developer to define default values for field when they are not explicitly defined during the test methods. This class can be used to set default values for all records created during test execution. Additionally this class can be used to set value for fields in those cases where SEED for Test Methods cannot adequately determine the field value based on validation rule evaluation.

Licensing

Individual Seat Licenses

Access to: Utilities and Developer Tools.

Individual Seat Licenses are available to developers who wish to utilize the developer tools alone. The developer can store credentials to all the orgs in which they have access to and utilize the tools to inspect object requirement or create code on the fly.

This access can be useful for a developer writing code outside of the Salesforce.com® UI and provides quick access to record creation code for use when the organization is using SEED for Test Methods **and** when the organization does not use SEED for Test Methods.

NOTE: Seat Licenses DO NOT provide access to the "obj_creation" class used during test methods. In addition the Serialize Validation Rules utility only operates on the organization SEED for Test Methods is installed in. (*future functionality will allow use when connecting to other organizations)*

Site License

Access to: Utilities, Developer Tools, and Apex Methods

This is the full license for SEED for Test Methods and allows the additional access to the "obj_creation" class to be utilized when writing test methods.

Activating Site License Key

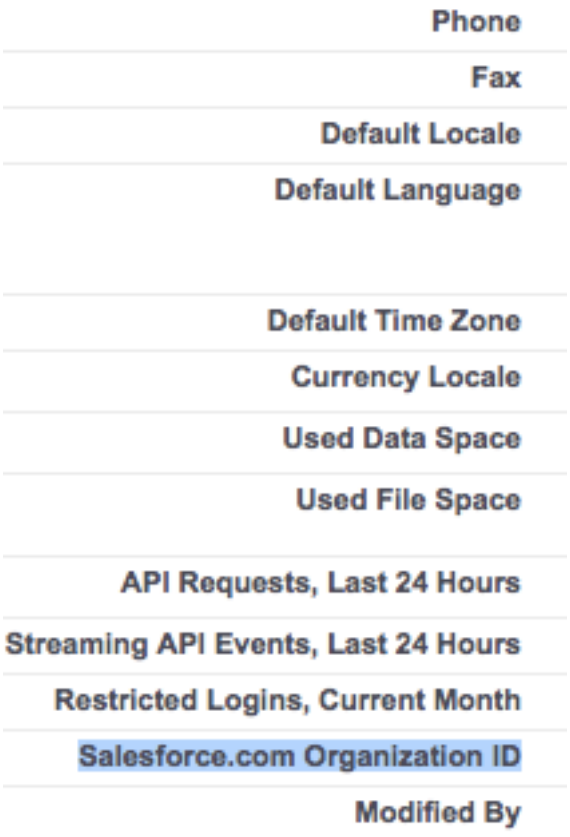
Obtaining

To obtain a site license key that will activate the text method functionality of SEED for Test Methods, please contact SEED or Test Methods support at eric.alexander1973@gmail.com.

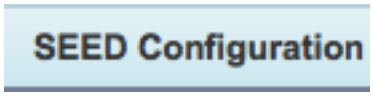
You will need to include your Salesforce® Organization ID which can be found by navigating in your organization to:

Setup -> Company Profile -> Company Information

On the right hand side of the screen you should see:

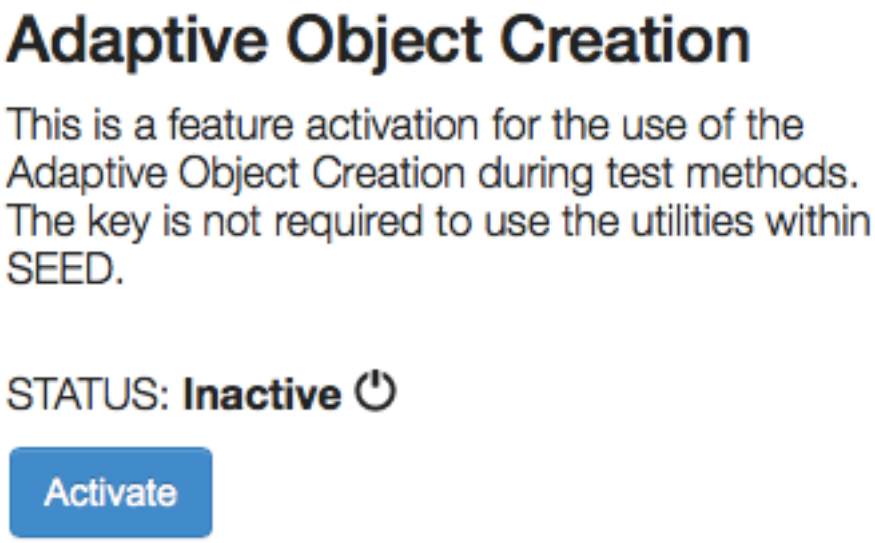


Installing

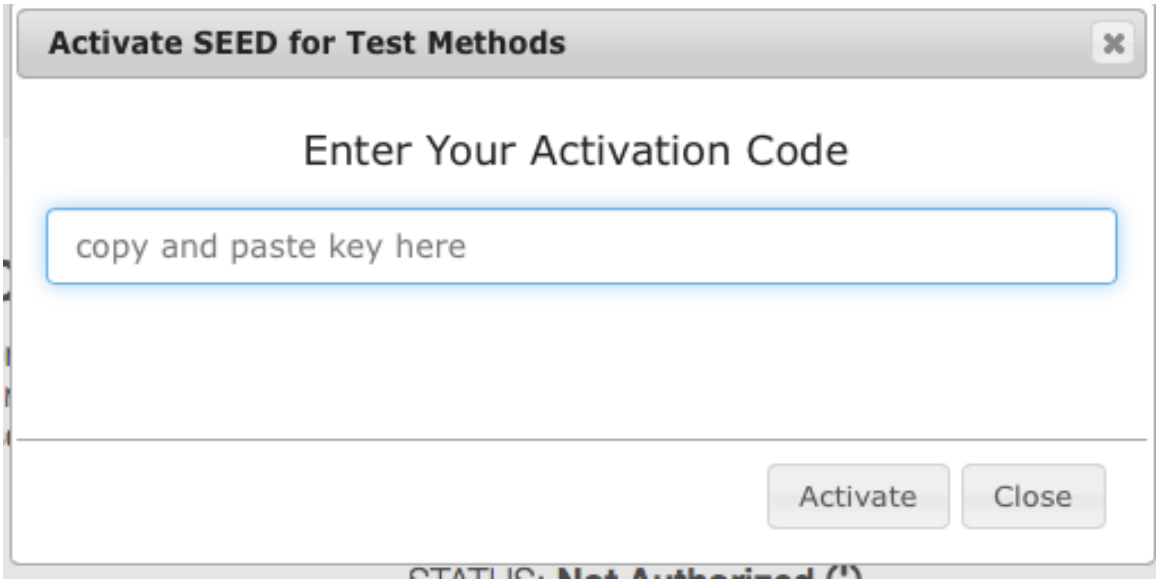
1. Obtain the key that was emailed to you and copy the entire key ensuring there are no spaces copied before or after the key.
2. Locate and click the tab 

If you have previously activated a site license for SEED for Test Methods you will see:

If this is the first time you are activating you will see



3. Click the Activate or Replace Key button and the following screen should appear

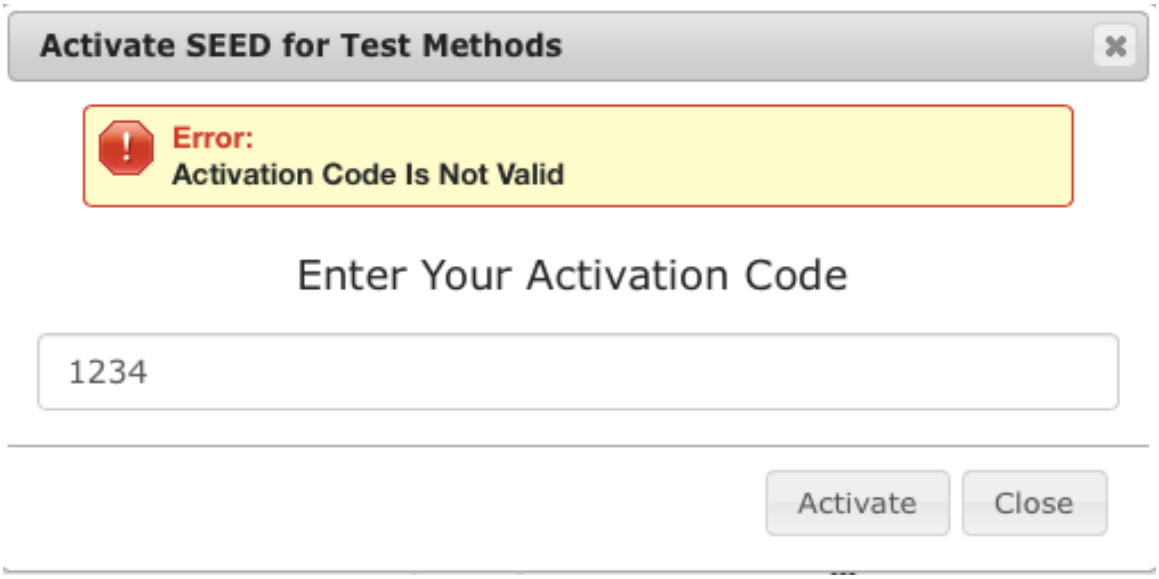


Copy and past the key where indicate and click activate.

At this point, if you have any active seat licenses, they will no longer be applicable and you are now upgraded to a Site License. You are now able to utilize the Obj_Creation Apex methods (Adaptive Object Creation) within your organization test method classes.

Troubleshooting

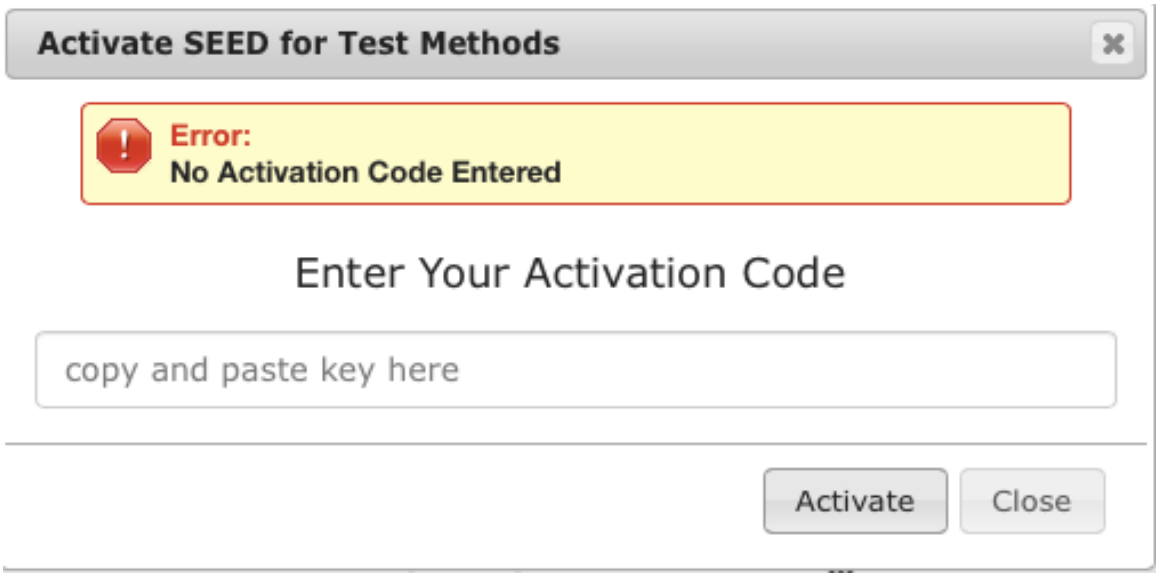
After clicking “Activate” you may see the message:



If this occurs, reenter the activation key you were provided.

- If you are copying the key from the email ensure that you have copied the entire key without and extra spaces before or after the key.
- If you are manually entering, ensure the case of the characters in the key match exactly

You may also see the message:



This occurs where there was no key entered for the activation key

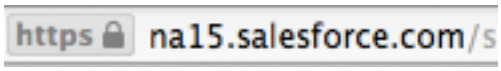
IMPORTANT NOTE: If you previously had an active key and successfully activated your SEED for Test Methods site license AND you clicked “Replace Key” and you fail to enter the key correctly and receive any of the above messages, your previous key will be deleted.

After Installation Setup

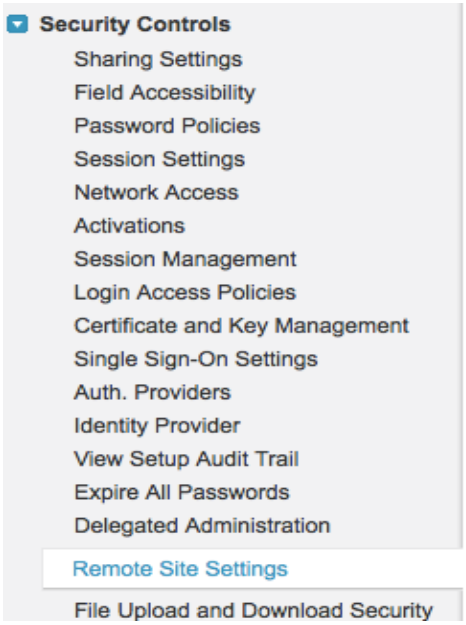
Remote Site Setup

The remote site settings will need to be updated to reflect the current pod of the installed organization.

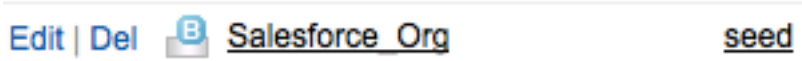
You can determine what pod you are on by inspecting the URL once logged into your organization (i.e. na1, na15, cs8, etc):

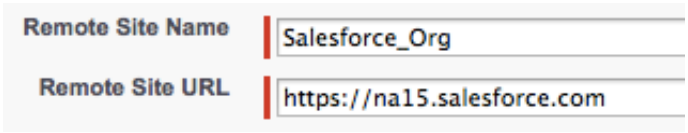


In this case the pod is “na15”.



1. Goto the Salesforce.com® setup page
2. Click on the drop down arrow by “Security Controls”
3. Click on “Remote Site Settings”
4. Edit the “Salesforce Org” remote site to modify with the correct pod





5. Save you changes

User Credentials

If you will be using the developer tools to connect to remote organizations, you will need to enter your user credentials for each remote organization you will be connecting to.

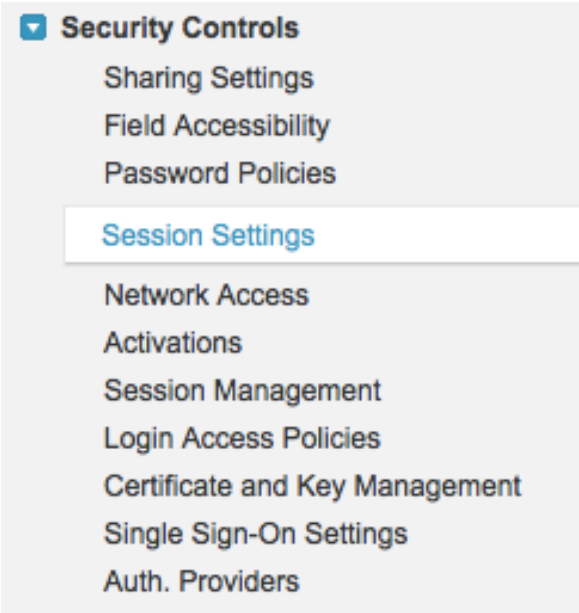
Serialize Validation Rules

If you would like to have SEED for Test Methods attempt to set field values for field not explicitly set during the test methods record creation, you will need to schedule the "ScheduleHandler" class using Apex Scheduler.

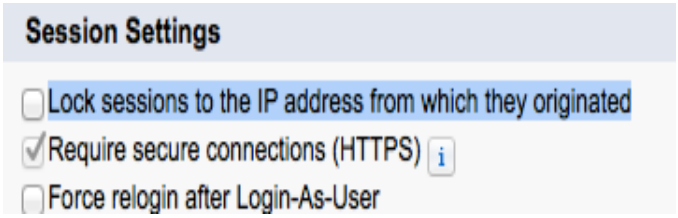
For immediate serialization, click on the "Serialize Validation Rules" tab and click the "Start Process" button. This page also provides instruction for how to schedule the process for future automatic processing.

Session Settings

In order for the developer tools to function properly the organization must have Lock sessions to the IP address disabled.



1. Goto the Salesforce.com® setup page
2. Click on the drop down arrow by "Security Controls"
3. Click on "Session Settings"
4. Uncheck "Lock sessions to the IP address from which they originated"



5. Save you changes

User Credentials

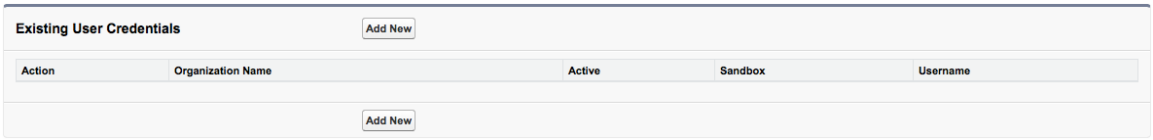
Stores the credentials for the organization to which the developer has access to. These credentials are used for the Developer tools

NOTE: You do not have to have any credentials stored, as the application will default to the current organization

NOTE: Only one (1) credential can be active at a time. Attempting to activate a second credential will cause the previous active credential to be inactivated.

Initial Setup

When you first click on the "User Credentials Tab" he initial screen displayed should be:



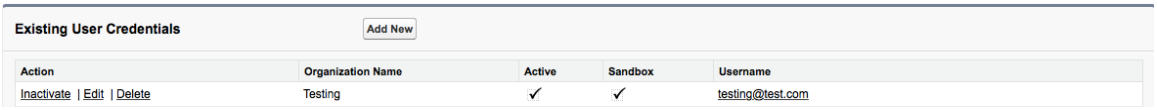
To enter a new credential click on the "Add New" button and you will be presented with:

- Organization Name: The name that will help you to identify what the credential is related to
- Active: Specifies if this credential is the default credential to be used by SEED for Test Methods Developer Tools
- isSandbox: Specifies that this credential is for a sandbox (test.salesforce.com login)
- Username: the developer's user name to login to the remote organization

Once you are done entering the information click the "Save" button

NOTE: Please ensure you have properly set up the remote site settings for the remote organization prior to validating your credentials

Once you are done you should see something like this:

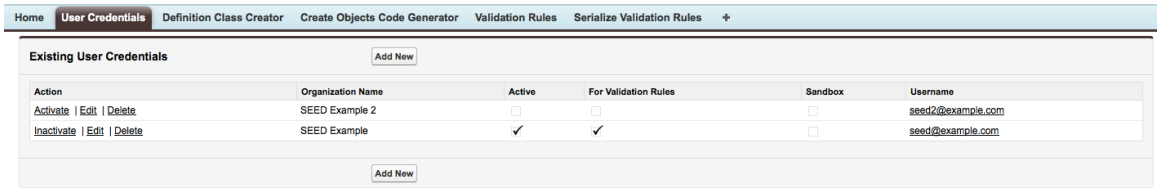


Maintaining Multiple Credentials

As a developer with a seat license you may want to store credentials from multiple organization for use with the Developer Tools.

1. Read the notes at the beginning of this section
2. Click "Add New" button and repeat the steps outlined in the initial setup of this section

Once you have multiple credentials you should see a screen similar to this:

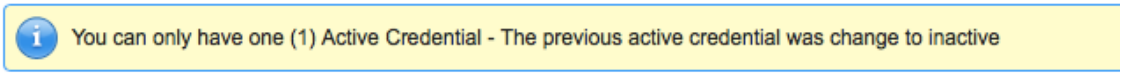


You will notice three (3) options under the action column:

1. Activate / Inactivate: Activates or deactivates the credential
2. Edit: Edit the credential
3. Delete: Delete the credential

Activate

If you click activate and there is another credential currently active, the credential you clicked activate on will be activated and the previously active credential will be deactivated.. You will also be presented with a message as follows:

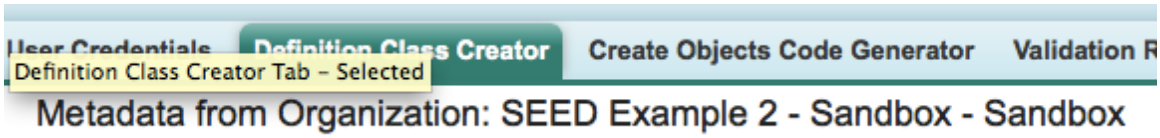


Inactivate

Make the credential you clicked inactivate on inactive.

Keeping Track

To help you keep track of which credential the developer tools are using, each tool will display the Organization Name from the active credential at the top of its page:



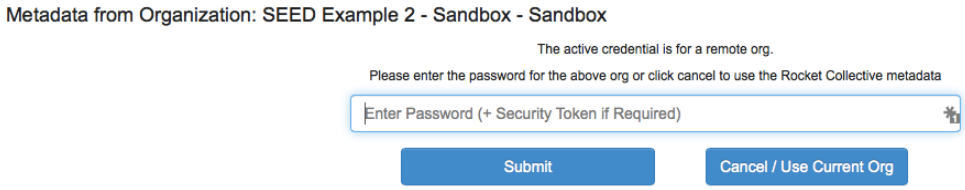
Definition Class Creator

The Definition class creator is designed to provide the developer will all the code required to create the recordTypeDefs apex class that SEED for Test Methods uses to set field values when those values are not explicitly defined within the test method when creating the record.

Connecting to Remote Organizations

To connect to remote organizations you must have finished developer credential setup prior to clicking this tab.

When you first access the utility and have an active credential set for a remote organization you should see:



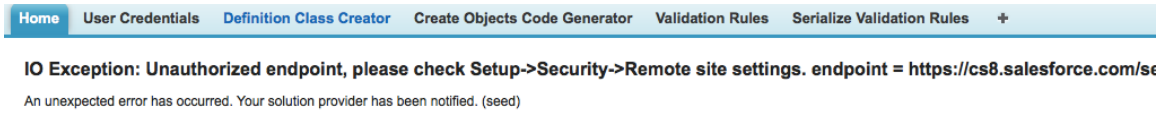
If there are no credentials defined or no active credentials you will be taken to the utility for the current organization.

1. If you want to connect to the current local organization simply click “Cancel / Use Current Org”
2. If you want to connect to the remote organization
    a. Enter the password for the remote organization and click “Submit”

Potential Errors

If you entered your active credentials and / or password incorrectly you will receive the following message:

If the remote site settings are not properly set up you will be presented with the following Salesforce® fatal error



If everything goes OK:

Selecting the Options

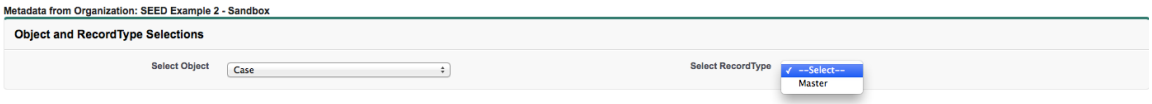
Object

This list is populated with a list of all available objects the developer has access to.

Record Type

After selecting the desired Object, the Record Type selection will be available to pick the desired record type associated with the object.

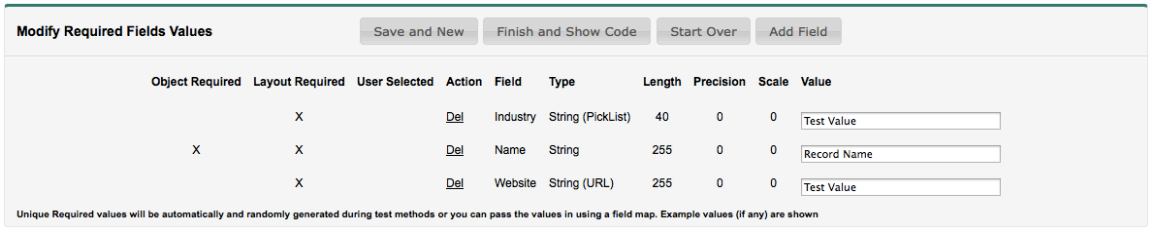
NOTE: If there are no record types defined for the object, the list will populate with a "Master" record type. Select this option to continue.



The record type selection is important as SEED for Test Methods will set the values for the fields based on the object AND record type. This allows for field variation that may be made in validation rules, triggers, etc for values as they relate to a specific record type.

Field Selections

After selecting the record type you will be presented with a starting field list including object required and layout required fields:



Modify Required Fields Values

Save and New | Finish and Show Code | Start Over | Add Field

Object Required	Layout Required	User Selected	Action	Field	Type	Length	Precision	Scale	Value
	X		Del	Industry	String (PickList)	40	0	0	Test Value
X	X		Del	Name	String	255	0	0	Record Name
	X		Del	Website	String (URL)	255	0	0	Test Value

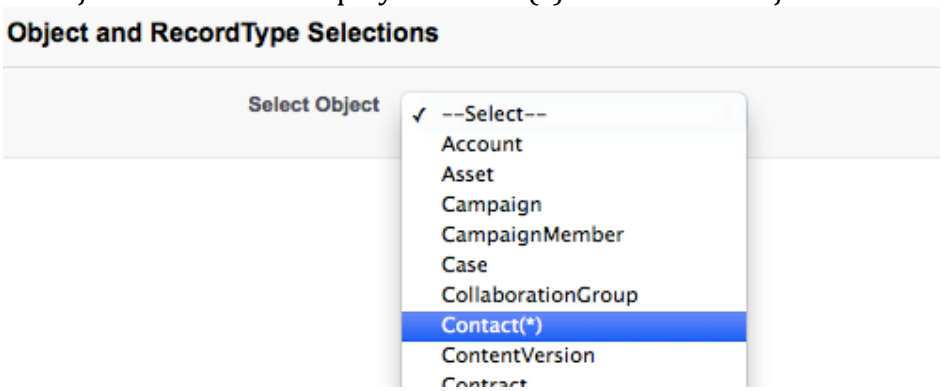
Unique Required values will be automatically and randomly generated during test methods or you can pass the values in using a field map. Example values (if any) are shown

Column Descriptions

1. Object Required: Indicates that this field was marked as required at the object level
2. Layout Required: Indicates that this field was marked as required on at least one layout associated with the selected record type
3. User Selected: Indicates that this field was manually selected by the developer
4. Action: Clicking "Del" removes the field from the list of fields to include in the final code.
    a. This does not actually delete the field. The field can be selected again by clicking the "Add Fields" button
5. Field: The API name of the field
6. Type: The base type with any extended type if applicable
    a. String (Picklist)
    b. String
7. Length: The maximum length of the field. For decimal fields this will be the same as the precision
8. Precision: The precision of a numerical field
9. Scale: The scale of a numerical field.
10. Value: The default value calculated by SEED for Test Methods:
    a. If this field is set a Reference or Unique field, you cannot edit the value provided by seed through the UI. You can edit the value when you create the class.

Button Descriptions

1. Save and New: Save the current selections and select a new object and record type to work with
    a. Once saved, you can edit the selections by selecting the same Object and record type combination. You can tell if you have previously save an object as it will be displayed with a (*) in the list of objects



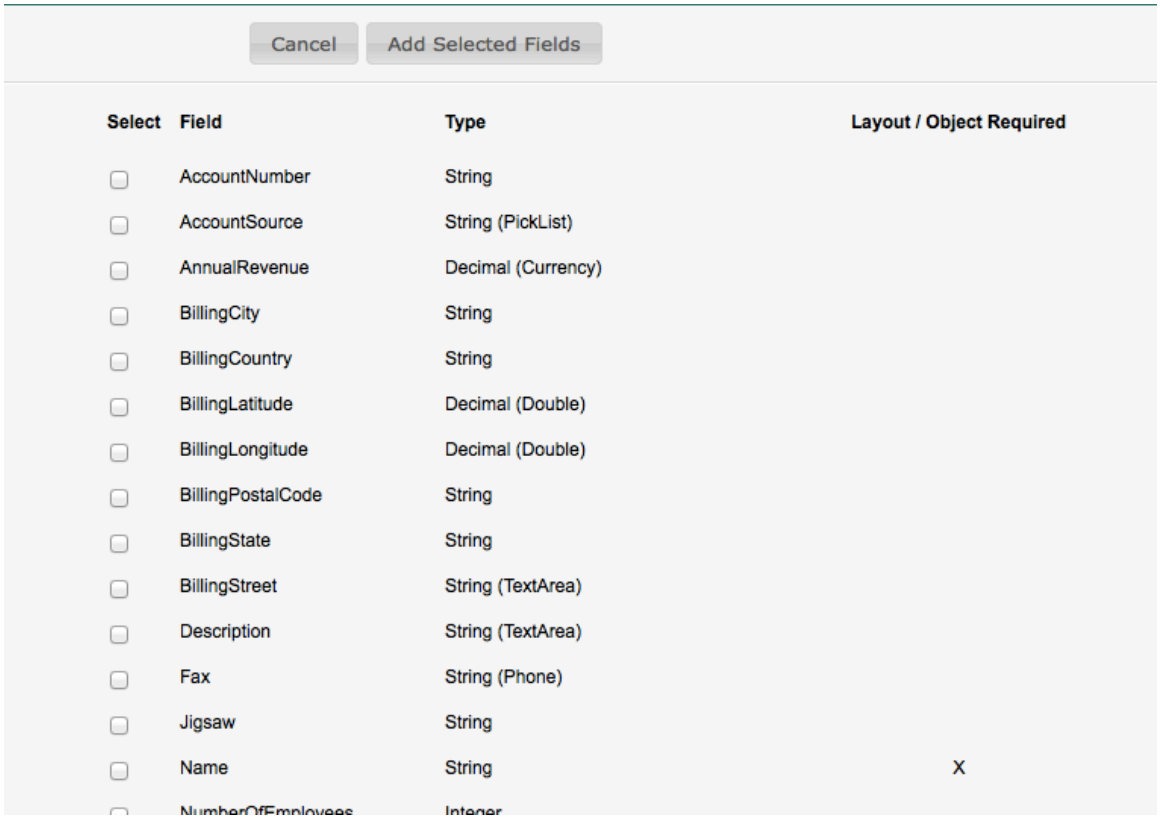
2. Finish and Show Code: Finish with all of your objects and selections and show the code to create the recordTypeDefs class.
3. Start Over: Erase all selection and start fresh.
4. Add Field(s): Add additional fields to the list of fields to use for the class creation.

Adding Fields

If the field you require do not appear on the Field Selections page click the "Add Field" button to select additional fields.

NOTE: If using **person accounts** please be sure to select Person Account Fields only if you have selected a person account record type. Currently Salesforce® does not provide a SOAP method to determine if a selected record type is a Person Account record type so person account fields will be pickable for all Account record types if Person Accounts is enabled in your organization.

The Add field(s) page will be similar to this:



NOTE: The Layout / Object Required column will have a X if you previously removed the field from the Field Selections page.

Selecting field(s)

To select a field simple check the box in the "Select" column next to each field you want to add then click "Add Selected Fields.

Canceling Selections

To cancel field selections and return to the Field Selections page, simply click the "Cancel" button

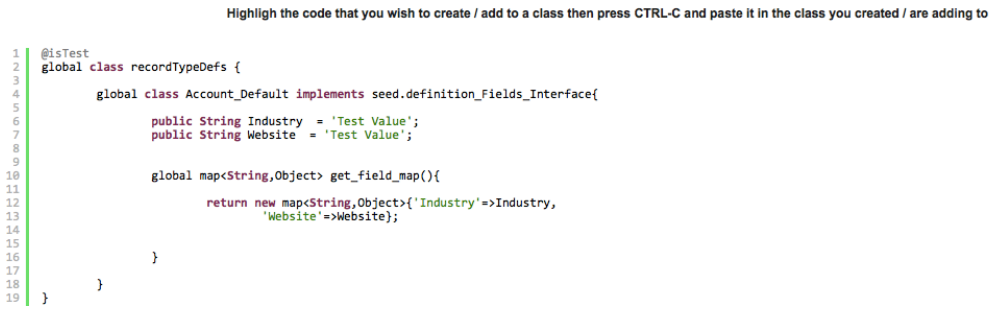
Save and New

If you would like to add multiple object / record type combination click the "Save and New" button. This will store your previous selections and allow you to pick a new object / record type combination. You can then make selection for that combination and repeat the process until you have finished with all objects / record types you need.

If you want to recall a previously created object record type combination, simple select the same object and record type. Your previous selections will be presented to you and you can make edits as required.

Finish and Show Code

When you are done with you selections click the "Finish and Show Code" button and the code to create the recordTypeDefs will be displayed. You can copy and paste this code into an Apex Class for use by SEED for Test Methods.



NOTE: For a more detailed description of the recordTypeDefs class please refer to that section.

The two buttons presented on this page are:

1. Start Over: Clear all selections and start over
2. Add More Classes: Add additional object / record type combinations

Create Objects Code Generator

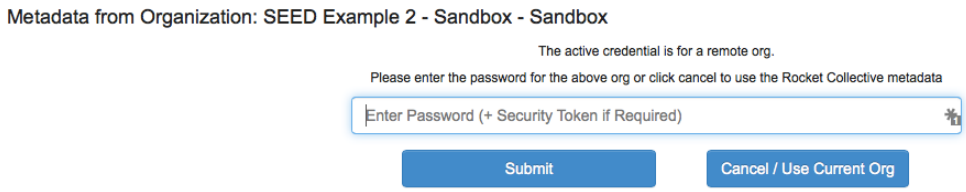
The Create Objects Code Generator assists the developer is obtaining the code required to create a record in a test method using SEED for Test Methods. In addition, it also provides a mode to provide record creation code in an org that is NOT using SEED for Test Methods.

This utility can be extremely useful for quickly identifying all required fields for a given object and providing the code to create a record of that type with a few clicks of the mouse. This greatly reduces the time it takes to inspect a given organization's requirements for record creation and increases the productivity of the developer when writing test methods regardless of if the target organization is using SEED for Test Methods.

Connecting to Remote Organizations

To connect to remote organizations you must have finished developer credential setup prior to clicking this tab.

When you first access the utility and have an active credential set for a remote organization you should see:



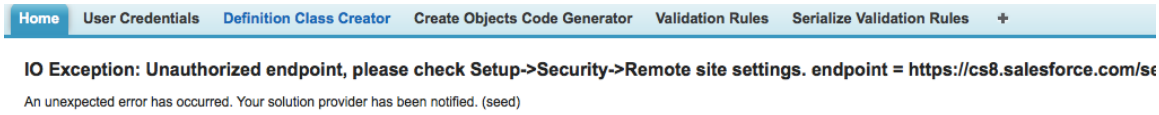
If there are no credentials defined or no active credentials you will be taken to the utility for the current organization.

1. If you want to connect to the current local organization simply click "Cancel / Use Current Org"
2. If you want to connect to the remote organization
    a. Enter the password for the remote organization and click "Submit"

Potential Errors

If you entered your active credentials and / or password incorrectly you will receive the following message:

If the remote site settings are not properly set up you will be presented with the following Salesforce® fatal error



If everything goes OK:

Selecting Options

Mode

- Package Installed: Generates code expecting SEED for Test Methods to be installed in the target organization
- Package NOT Installed: Generates standard Salesforce® record creation code for use in target organizations not using SEED for Test Methods

Select Object

- Select the Object you with to work with

Method

- If the mode is Package NOT installed this will be set to N/A
- If the mode is Package Installed the following options are available to generate the appropriate code
    - Default Values Insert: Basic object creation with field / values specified inserting the record in the same call
    - Default values and NOT insert: Same as previous except the record is not inserted

- Specify Fields and Insert: Object record creation code passing is specific field / value pairs to be used when creating the record and inserting the record in one call
- Specify fields and NOT insert: Same as previous except the record is not inserted
- Fill Fields on sObject: Object creation code that uses an object instantiated in the test method and using SEED for Test Methods to fill in the remaining required values

NOTE: Please refer to the Obj_Creation class section for a detailed description on the developer of the code generated

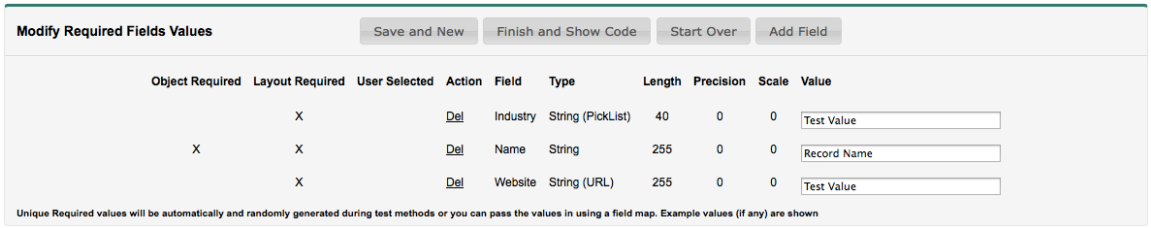
Record Type

- Select the record type of the object you are working with
- If there are no record types defined for the object, select "Master" as the record type.

Field Selections

NOTE: If the method is *Default Values and Insert* OR *Default Values and NOT Insert* this section will **NOT** be displayed. You will be taken directly to the code needed to generate the object record.

After selecting all the options you will be presented with a starting field list including object required and layout required fields:



Column Descriptions

1. Object Required: Indicates that this field was marked as required at the object level
2. Layout Required: Indicates that this field was marked as required on at least one layout associated with the selected record type
3. User Selected: Indicates that this field was manually selected by the developer
4. Action: Clicking "Del" removes the field from the list of fields to include in the final code.
    a. This does not actually delete the field. The field can be selected again by clicking the "Add Fields" button
5. Field: The API name of the field
6. Type: The base type with any extended type if applicable
    a. String (Picklist)
    b. String
7. Length: The maximum length of the field. For decimal fields this will be the same as the precision
8. Precision: The precision of a numerical field
9. Scale: The scale of a numerical field.
10. Value: The default value calculated by SEED for Test Methods:
    a. If this field is set a Reference or Unique field, you cannot edit the value provided by seed through the UI. You can edit the value when you create the class.

Button Descriptions

1. Finish and Show Code: Finish with all of your objects and selections and show the code to create the recordTypeDefs class.
2. Start Over: Erase all selection and start fresh.
3. Add Field(s): Add additional fields to the list of fields to use for the class creation.

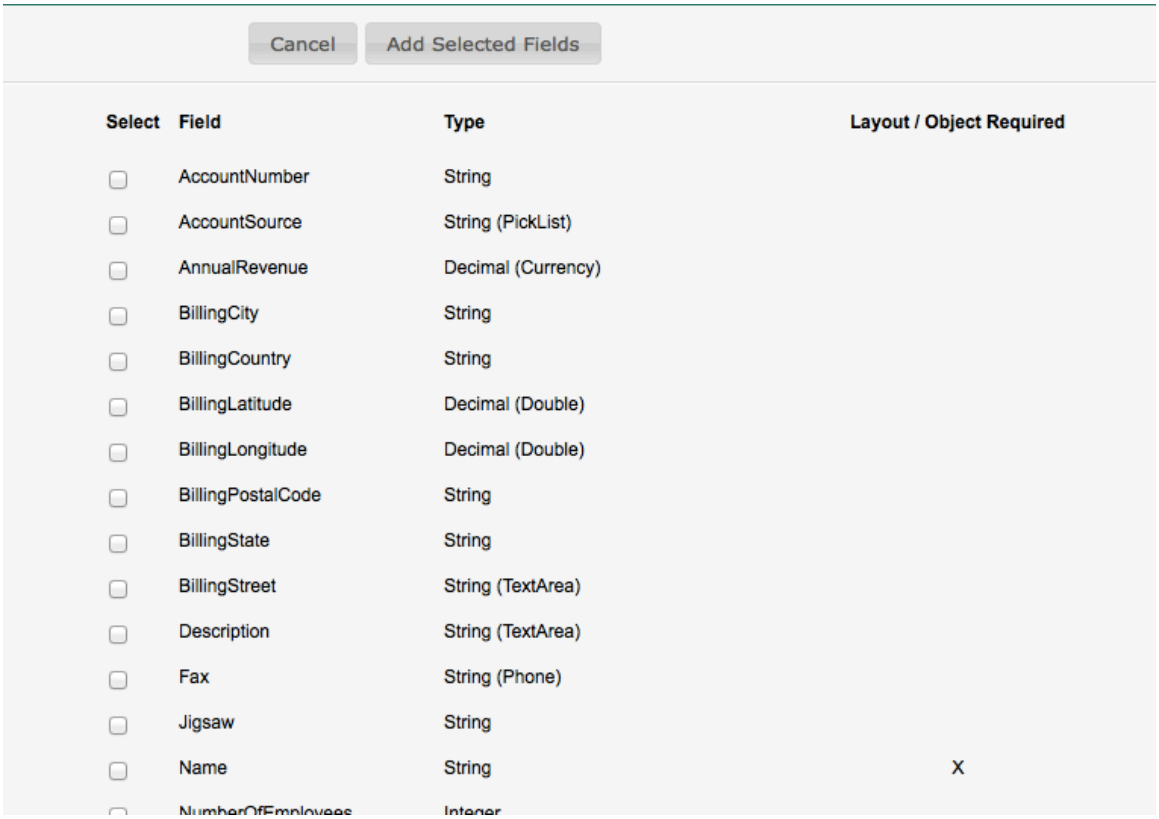
Adding Fields

If the field you require do not appear on the Field Selections page click the "Add Field" button to select additional fields.

NOTE: If using **person accounts** please be sure to select Person Account Fields only if you have selected a person account record type. Currently Salesforce® does not provide a SOAP method to determine if a selected record type is a Person Account

record type so person account fields will be pickable for all Account record types if Person Accounts is enabled in your organization.

The Add field(s) page will be similar to this:



NOTE: The Layout / Object Required column will have a X if you previously removed the field from the Field Selections page.

Selecting field(s)

To select a field simple check the box in the "Select" column next to each field you want to add then click "Add Selected Fields.

Canceling Selections

To cancel field selections and return to the Field Selections page, simply click the "Cancel" button

Finish and Show Code

When you are done with you selections click the "Finish and Show Code" button and the code to create the record will be displayed. You can copy and paste this code into your test methods to create the record.

Code Examples

Package NOT Installed

Highligh the code that you wish to create / add to a class then press CTRL-C and paste it in the class you created / are adding to

```
RecordType rt = [Select ID From RecordType Where sObjectType = 'Account' AND Name = 'Default' LIMIT 1];

Account rec = New Account(Industry = 'Test Value',
        Name = 'Record Name',
        Website = 'Test Value',
        RecordtypeID = rt.id);
```

Package Installed – Default Values

Highligh the code that you wish to create / add to a class then press CTRL-C and paste it in the class you created / are adding to

```
Two options based on need:

1. Returning sObject immediately:

        seed.Obj_Creation objs = New seed.Obj_Creation();

        objs.setRecordTypeName(Account.sObjectType,'Default');

        Account rec = (Account)objs.createRecord(Account.sObjectType);

2. Just create. You can obtain the created object at any time by calling the getRecordsList or GetRecordsMap methods:

        seed.Obj_Creation objs = New seed.Obj_Creation();

        objs.setRecordTypeName(Account.sObjectType,'Default');

        objs.createRecord(Account.sObjectType);
```

Package Installed – Pass In Fields

Highligh the code that you wish to create / add to a class then press CTRL-C and paste it in the class you created / are adding to

```
Two options based on need:

1. Returning sObject immediately:

        seed.Obj_Creation objs = New seed.Obj_Creation();

        objs.setRecordTypeName(Account.sObjectType,'Default');

        Account rec = (Account)objs.createRecord(Account.sObjectType, new map<String,Object>{'Industry'=>'Test Value',
                                'Name'=>'Record Name',
                                'Website'=>'Test Value');

2. Just create. You can obtain the created object at any time by calling the getRecordsList or GetRecordsMap methods:

        seed.Obj_Creation objs = New seed.Obj_Creation();

        objs.setRecordTypeName(Account.sObjectType,'Default');

        objs.createRecord(Account.sObjectType, new map<String,Object>{'Industry'=>'Test Value',
                                'Name'=>'Record Name',
                                'Website'=>'Test Value');
```

Package Installed – Fill Fields on sObject

Highligh the code that you wish to create / add to a class then press CTRL-C and paste it in the class you created / are adding to

```
Two options based on need:

1. Insert immediately With Default Values


        seed.Obj_Creation objs = New seed.Obj_Creation();

        objs.setRecordTypeName(Account.sObjectType,'Default');

        Account rec = New Account();
        rec = (Account)objs.createRecord(Account.sObjectType,rec,true);

2. Just fill fields. You can insert manually. The sObject will also be added to the created records list which can be otained by calling the getRecordsList or GetRecordsMap


        seed.Obj_Creation objs = New seed.Obj_Creation();

        objs.setRecordTypeName(Account.sObjectType,'Default');

        Account rec = New Account(Industry = 'Test Value',
                Name = 'Record Name',
                Website = 'Test Value');
        rec = (Account)objs.createRecord(Account.sObjectType,rec,false);
```

NOTE: Please refer to the Obj_Creation class section for a detailed description on the developer of the code generated

Serialize Validation Rules

SEED for Test Methods will attempt to set field values during test methods for thos fields not explicitly set either by the developer or the recordTypeDefs class during the test methods. In order to do this the current organization validation rules need to be serialized for use by SEED for Test Methods.

Serialization can be schedule using scheduled Apex on a recurring basis or for immediate serialization of all current organization validation rules click on the "Serialize Validation Rules" tab.

Although this is not required to utilize SEED for Test Methods to create records during test methods, it is required if you want SEED for Test Methods to evaluate field values against validation rules when setting default values.

NOTE: The User Credential marked, "Use for Obtaining Validation Rules", **MUST** be for a developername in the organization you are accessing this tab from. Serialization of validation rules from remote organizations is not currently supported.

Potential Errors

If the remote site settings are not properly set up you will be presented with the following error after starting the process:

Serializing

When you click the tab the page should look similar to this:

1. Click "Start Process" to begin

   a. The status will be updated every 5 seconds
   b. The process is dependent on Salesforce® executing the future method
   c. The process should typically take under a minute to complete
2. Once the process is complete you will see the following

3. Final Options
   a. Reset Page: Returns to the beginning
   b. See Validation rules: Takes you to the Validation Rules Object to view the validation rules identified

Notifications

SEED for Test Methods will send an Email notification upon serialization of validation rules for all *NEW* or *UPDATED* validation rules that are identified as note being handled by SEED for test methods.

This notification provides information needed for an administrator to update the recordTypeDefs class to default the field values for that object so they meet the validation requirements.

IMPORTANT POINTS

- **The recordTypeDefs is the only place you will have to update the field values unless:**
  - **Fields that are a part of the validation rule not handled by SEED for Test Methods have values explicitly defined within a test method AND those test methods are failing.**

This should only occur when the developer is explicitly setting field values to be able to produce a desired result. If validation rule are altered at a later time to not meet those requirements then the test method needs update to meet the new requirements

Identifying Notification Recipient

The recipient of the notification is defined in the workflow email alert named

Notification of a New Validation Rule Un-Handled by SEED for Test Methods

Edit this workflow rule and modify the recipient as desired.

The notification will be similar to this:

A new validation rule was identified that SEED for Test Methods cannot process

Salesforce Object	Rule Name	Rule Text
Contact	Contact.seed__Unhandled	LEN(LastName) > 10

To resolve this issue and prevent test methods from failing you can take the following actions:

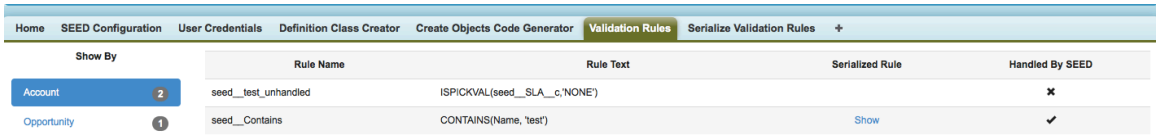
1. If the validation rule does not affect your test methods no action is needed

   NOTE: Not resolving this issue may cause future test methods to fail

2. Update the \'recordTypeDefs\' Class to define default fields to populate values that will pass the validation rule
3. Update your test methods to pass in the appropriate values to pass the validation rule

Validation Rules

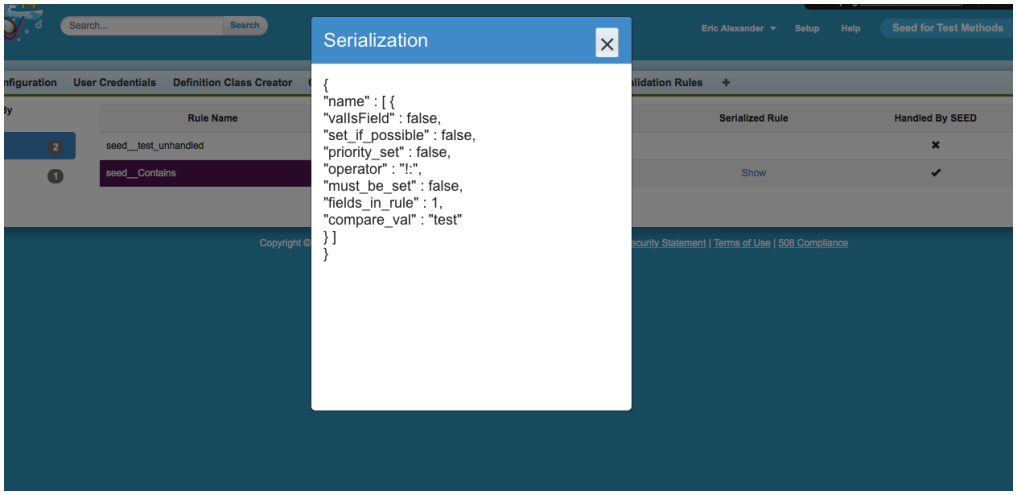
Once you have serialized the validation rule you can view the results on this screen by clicking the Validation Rules tab. The rules will be displayed by the object name on the left with a list of the rules in the right pane:



Each object has a badge identifying how many rules were found for the object.

The right pane has the following columns:

1. Rule Name: The API name of the validation rule
2. Rule Text: The actual text of the validation rule
3. Serialized Rule: If it is handled by seed there sill be a link named “Show” that you can click on to see how SEED serialized the rule and the screen looks similar to this:



to help you in a large list of rule, the rule this applies to will be highlighted in purple in the background

4. Handled By SEED: Indicates if SEED can handle this rule in its Adaptive Object Creation logic. If this shows an X then you will need to account for this rule either in the recordTypeDefs or by passing in appropriate field / value combinations when writing your test methods
5. To view the rules on other object, simply click on the desired object in the left pane and the right pane will refresh and display the rules for that object

Validation Rule Batch Processing

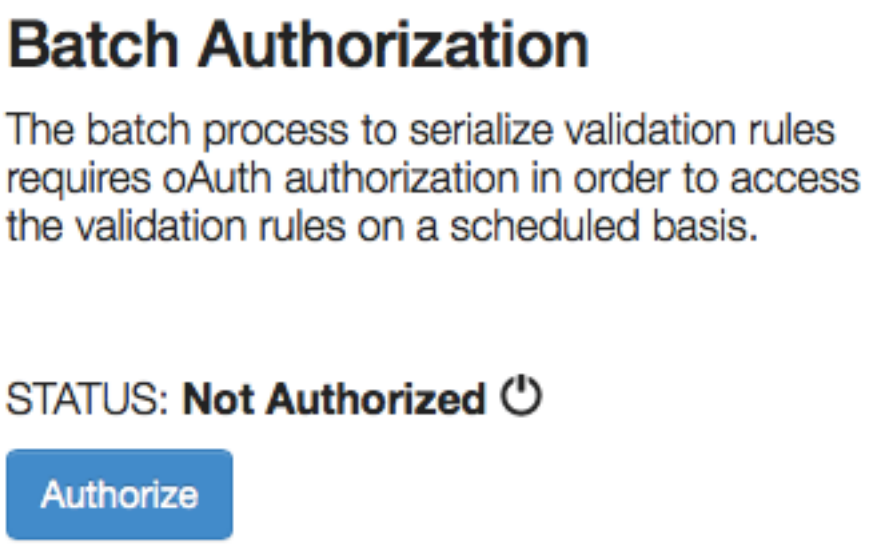
Organizations continually update their validation rules and add new rules frequently to meet business requirements. Without some sort of automated monitoring an admin with access to SEED would have to manually serialize the rules on a frequent basis to ensure all the rules are covered.

SEED for Test Methods can automate that process by scheduling a batch process to run on a daily basis to perform this serialization for you.

Enabling

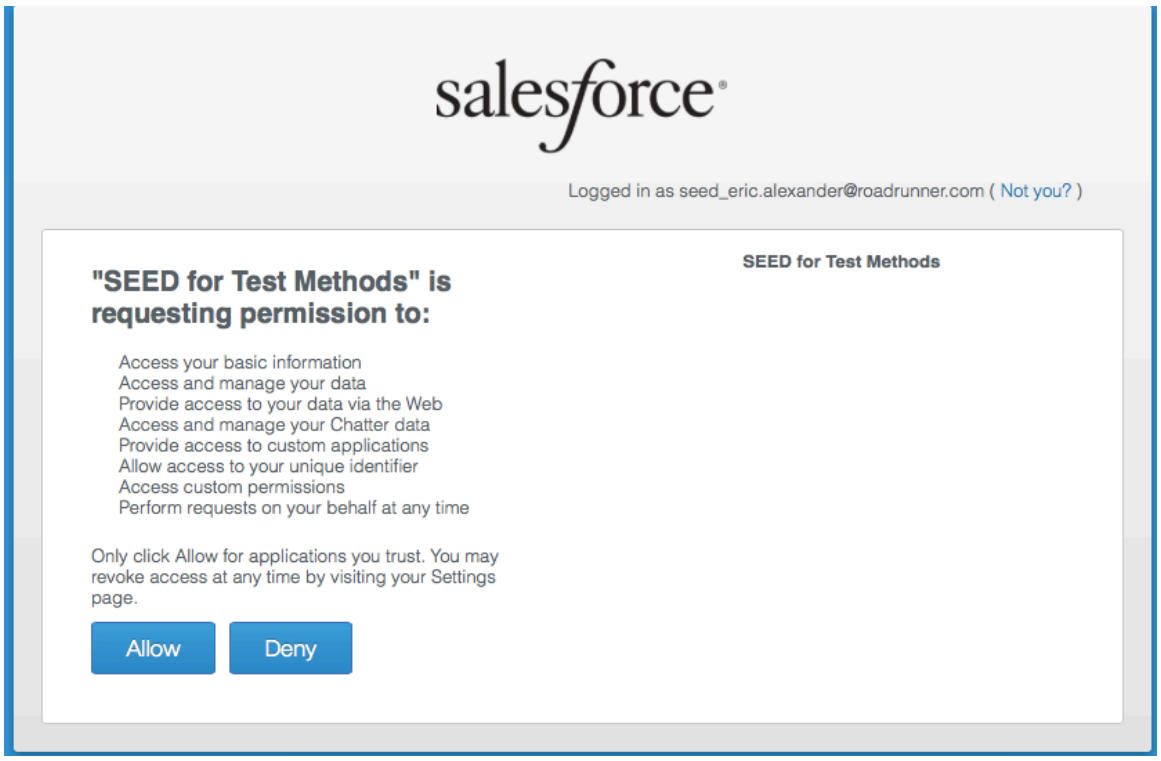
To enable this feature you will need to open the SEED Configuration Tab and perform two steps:

Batch Authorization



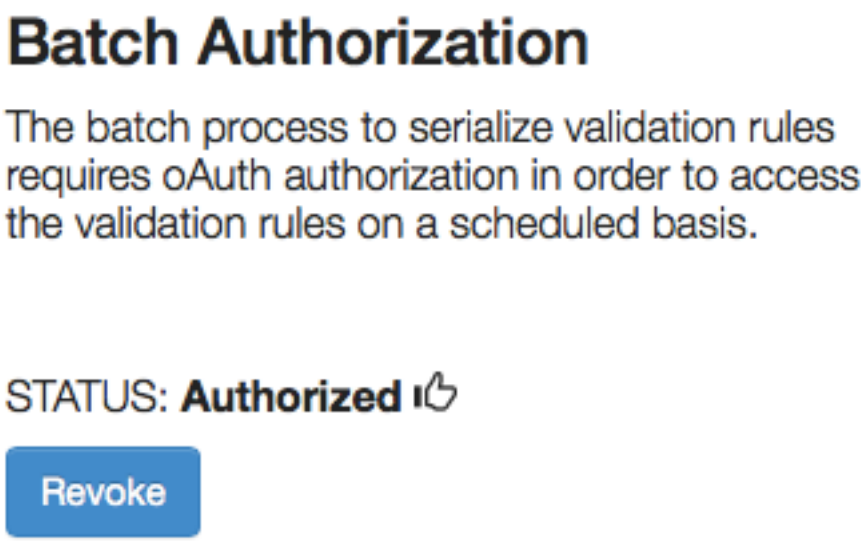
This will walk you through the oAuth process to grant SEED for Test Methods authorization to perform API access on your behalf.

a. Click “Authorize to begin
b. At the login screen, enter your credentials for the local org
c. You will then be presented with the oAuth approval screen



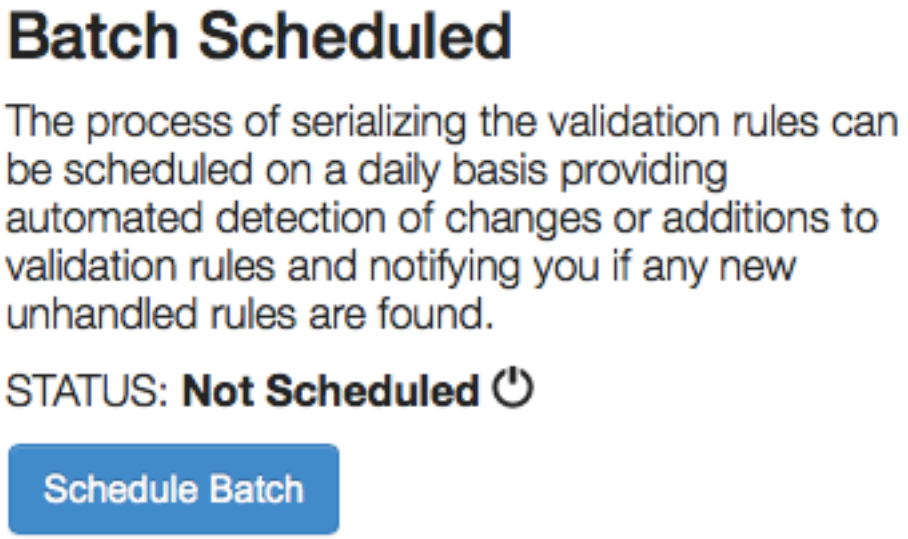
a. Click "Allow"

d. You will be redirected back to the local organization where you should now see



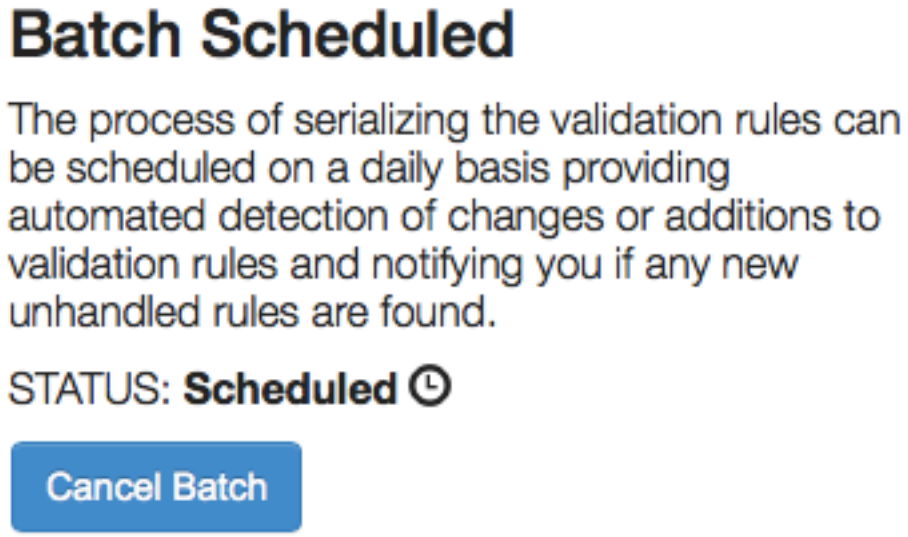
NOTE: You can revoke your authorization by clicking the Revoke button at any time

Schedule the batch



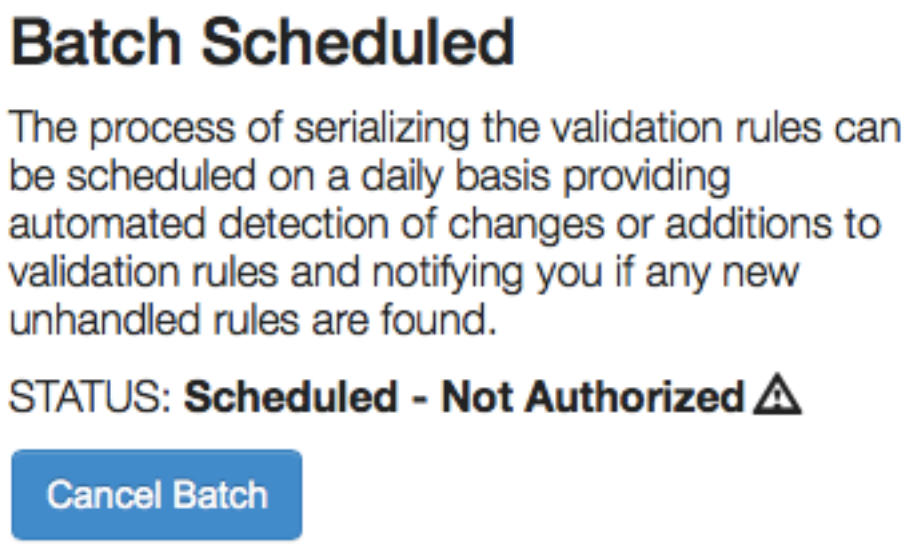
a. Click “Schedule Batch”

e. You should now see



Potential Issues

There may be times where your authorization expires or has been revoked and the batch cannot actually serialize the validation rules. If this is the case you will see



You have two options:

1. Cancel the batch: Remove the scheduled batch job from processing
2. Re Authorize: Follow the steps for “Batch Authorization” above again

recordTypeDefs Class Description

The recordTypeDefs class is used by SEED for Test Methods to set field values not explicitly set during the test method. This class is the second level of field / value identification and is ignored for a given field if the developer set the value of that field during object creation.

This class can be manually created **or** if you have a seat or site license you can use the "Definition Class Creator" Utility by clicking on the tab with the same name

Class Structure

The class signature must be:

```
@isTest
global class recordTypeDefs {

}
```

Inner Class Structure

Signature

The class method signatures must follow the format:

```
global class {ObjectAPIName}_{RecordTypeName} implements
seed.recordTypeDefs_Interface{

}
```

If the *ObjectAPIName* is a custom object replace **__c** with the word in caps "**CUSTOM**"

```
global class {My_Custom_ObjectCUSTOM}_{RecordTypeName} implements
seed.recordTypeDefs_Interface{

}
```

Continued on next page

If the *RecordTypeName* contains spaces, you must replace the space with an underscore _ character

```
global class Account_RT_Name implements
seed.recordTypeDefs_Interface{}
```

If the Object does NOT have a record type defined omit the {_*RecordTypeName}* part

```
global class Account implements seed.recordTypeDefs_Interface{}
```

Inner Class Code

The code inside of the inner class signature follows the following format:

Properties

Each field that is being set must have a defined property for that field.

```
    public String Industry = 'Test Value';
    public String Name = 'Record Name';
    public String Website = 'Test Value';
    public Decimal qty = 100;
```

Methods

Each inner class must implement the seed.recordTypeDefs_Interface method Map<String,Object> getFields();

This method returns a Map<String,Object> to SEED for Test Methods that provide the API Field Name / value setting for the Object / RecordType combination.

The method should look something similar to this

```
    global map<String,Object> get_field_map(){

    return new map<String,Object>{'Industry'=>Industry,
                                  'Name'=>Name,
                                  'Website'=>Website};

    }
```

Jump to the next page for a complete example of the class

recordTypeDefs Class Example

The following example is for the Contact Object without a record type defined as well as one custom object with a record type defined.

If an object has record types defined you cannot create an inner class without having the record type name in the class.

For example: If you have defined record type name 'Accounting' for the Contact object the below code will not set the value for the contact records during test method execution as there is no class defined as `global class` Contact_Accounting

```
@isTest
global class recordTypeDefs {

        global class Contact implements seed.recordTypeDefs_Interface{

                public String Name  = 'Record Name';
                public String AssistantPh  = 'Test Value';
                public Date Birthdate  = date.today();

                global map<String,Object> get_field_map(){

                        return new map<String,Object>
                                    {'AssistantPhone'=>AssistantPh,
                                'Birthdate'=>Birthdate,
                                'LastName'=>Name};
                }
        }

      global class ResumeCUSTOM implements
seed.recordTypeDefs_Interface{

                public String Name  = 'Record Name';

                global map<String,Object> get_field_map(){
                        return new map<String,Object>
                                        {'Name'=>Name};
                  }

        }
```

Obj_Creation Class

The *Obj_Creation* Class is the core functionality of SEED for Test Methods. This class is what sets all value for fields not explicitly set in the test method that may be required by the object, page layout, and validation rules.

This class also stores all records created during the test method for later retrieval and provides methods to control its behavior so the developer can assert for results that produce errors during negative test cases.

Methods

Instantiate

At the beginning of every test method that will be utilizing SEED for Test Methods, instantiate this class as follows

```
Seed.Obj_Creation objs = New Seed.Obj_Creation();
```

The property name can be anything you desire and does not need to be *obj*.

Record Type Name

Before the *first* creation of a record for a specified type (Account, Contact, Custom Object), having record types defined, you must set the value of the record type name. Subsequent record creation for the same object does not require calling this method unless the record type you wish to use is different

Signature: *void* setRecordTypeName(Schema.sObjectType *so*, String *rt*)

```
Objs.setRecordTypeName(Account.sObjectType,'RT_Name');
```

Ignore Validation Rules

In those cases where the developer needs to test for negative test cases specifically testing a validation rule or for any other reason, SEED for Test Methods provides a method to turn of validation rule processing when setting field values.

It is important to remember that validation rule processing is ignored for fields that have been explicitly passed into the `createRecord()` method. In these cases the developer will not need to call this method to ignore validation rules.

NOTE: If setIgnoreValidationRules() is called, it is only enforce for the next call to the createRecord() method and is not valid on the next call to createRecords() unless setIgnoreValidationRules() called again.

Signature: void setIgnoreValidationRules()

```
Objs.setRecordTypeName(Account.sObjectType,'RT_Name');
```

Creating a record

SEED for Tests Methods provides several signatures for creating records depending on the requirement of the developer at the time of the record creation.

NOTE: Use the "Create Objects Code Generator" utility to provide the code for you

Default values and Insert

Signature: sObject createRecord(Schema.sObjectType *so*)

```
Objs.createRecord(Account.sObjectType);
```

Code Example

```
Two options based on need:

1. Returning sObject immediatly:
```

```
seed.Obj_Creation objs = New seed.Obj_Creation();
     Account rec = (Account)objs.createRecord(
                       Account.sObjectType);
```

```
2. Just create. You can obtain the created object at any time by
calling the getRecordsList or getRecordsMap methods:
```

```
seed.Obj_Creation objs = New seed.Obj_Creation();
     objs.createRecord(Account.sObjectType);
```

Signatures continue on next page

Default values and Specify Insert

Signature: sObject createRecord(Schema.sObjectType *so,* Boolean *doInsert*)

```
Objs.createRecord(Account.sObjectType, false);
```

Code Example

Two options based on need:

1. Returning sObject immediately:

```
seed.Obj_Creation objs = New seed.Obj_Creation();
     Account rec = (Account)objs.createRecord(
                      Account.sObjectType,
                      false,);
```

2. Just create. You can obtain the created object at any time by calling the getRecordsList or getRecordsMap methods:

```
seed.Obj_Creation objs = New seed.Obj_Creation();
     objs.createRecord(Account.sObjectType, false);
```

Signatures continue on next page

Specify values and Insert

Signature: sObject createRecord(Schema.sObjectType *so,*
Map<String,sObject> f*ield_values*)

```
Objs.createRecord(Account.sObjectType, New
     Map<String,sObjectField>{‘Name’=>’RecordName’});
```

Code Example

Two options based on need:

1. Returning sObject immediately:

```
     seed.Obj_Creation objs = New seed.Obj_Creation();
          Account rec =
          (Account)objs.createRecord(Account.sObjectType,
               new map<String,Object>{'Industry'=>'Test Value',
                                      'Name'=>'Record Name',
                                      'Website'=>'Test Value');
```

2. Just create. You can obtain the created object at any time by calling the getRecordsList or getRecordsMap methods:

```
     seed.Obj_Creation objs = New seed.Obj_Creation();
          objs.createRecord(Account.sObjectType,
               new map<String,Object>{'Industry'=>'Test Value',
                                      'Name'=>'Record Name',
                                      'Website'=>'Test Value');
```

Signatures continue on next page

Specify values and Specify Insert

Signature: sObject createRecord(Schema.sObjectType *so*,
Map<String,sObject> *field_values*,
Boolean *doInsert*)

```
Objs.createRecord(Account.sObjectType, New
     Map<String,sObjectField>{‘Name’=>’RecordName’}, false);
```

Code Example

Two options based on need:

1. Returning sObject immediately:

```
     seed.Obj_Creation objs = New seed.Obj_Creation();
          Account rec =
          (Account)objs.createRecord(Account.sObjectType,
               new map<String,Object>{'Industry'=>'Test Value',
                                      'Name'=>'Record Name',
                                      'Website'=>'Test Value',
                                     false);
```

2. Just create. You can obtain the created object at any time by calling the getRecordsList or getRecordsMap methods:

```
     seed.Obj_Creation objs = New seed.Obj_Creation();
          objs.createRecord(Account.sObjectType,
               new map<String,Object>{'Industry'=>'Test Value',
                                      'Name'=>'Record Name',
                                      'Website'=>'Test Value',
                                     false);
```

Signatures continue on next page

Ignore Required Nulls

In those cases where the developer needs to test for negative test cases specifically required fields missing or for any other reason, SEED for Test Methods provides a variable in the below signature to turn of processing of default values for fields set to null when specifying field / value pairs

NOTE: By default, if the developer explicitly sets a field value to *null* and the field is required at the object level, SEED for Test Methods will set a default value for the field other than null unless this method is called

Specify values, Specify Insert, and Specify Ignore Required Null

Signature: sObject createRecord(Schema.sObjectType *so,*
Map<String,sObject> *field_values,*
Boolean *doInsert,*
Boolean ignoreRequriedNulls)

Code Example

Two options based on need:

1. Returning sObject immediately:

```
seed.Obj_Creation objs = New seed.Obj_Creation();
     Account rec =
     (Account)objs.createRecord(Account.sObjectType,
          new map<String,Object>{'Industry'=>'Test Value',
                                 'Name'=>'Record Name',
                                 'Website'=>'Test Value',
                                 false,
                                 true);
```

2. Just create. You can obtain the created object at any time by calling the getRecordsList or getRecordsMap methods:

```
seed.Obj_Creation objs = New seed.Obj_Creation();
     objs.createRecord(Account.sObjectType,
          new map<String,Object>{'Industry'=>'Test Value',
                                 'Name'=>'Record Name',
                                 'Website'=>'Test Value',
                                 false,
                                 true);
```

Getting List of Records Created

On each execution of one of the createRecord() methods, SEED for Test Methods will store the created record in the order in which SEED for Test Methods processed them. This storage happens regardless of if the developer specified that the record not be insert or not.

The records are stored and retrieved according to *Schema.sObjectType*.

Signature: sObject[] getRecordsList(Schema.sObjectType *so*)

```
Objs.getRecordsList(Account.sObjectType)
```

NOTE: If the developer set the value of *doInsert* to false the ID of the returned record will be null until the record is inserted by the developer. After the insertion, the related record in the list of records will have the correct value for the record ID populated.

Getting Map of Records Created

Signature: Map<ID,sObject> getRecordsMap(Schema.sObjectType *so*)

```
Objs.getRecordsMap(Account.sObjectType)
```

NOTE: If the developer set the value of *doInsert* to false the returned map will NOT include the records that have not been inserted. After the developer inserts the records, subsequent calls to getRecordsMap will return include the newly inserted records.

Add Records to List

If the developer creates a record outside of the SEED for Test Methods framework, they can take advantage of the get records list and get records map features of SEED for Test Methods by adding the record to the list of records created.

NOTE: The record is NOT inserted by this call. At some point the developer will need to insert the record. The insert can be before the call to this method.

Signature: sObject addRecordToList(sObject *so*)

```
Account a = New Account();
Objs.addRecordToList(a)
```

IMPORTANT NOTE: This method can only be called **after** your first call to one of the createRecord() methods. If called prior to that you will receive a seed.SEED_Exception error message telling you that you cannot call this method yet

Create User

NOTE: The record is NOT inserted by this call.

When the developer needs to create a developer during test method execution SEED for Test Methods provides a method to create developers as well. By utilizing this method you can take advantage of the get records list and get records map methods.

Signature: User createUser(String *profileName,* String *roleName*)

The *roleName* can be set to *null* if you do not want to set a role for the developer

```
Profile p = [Select Name
                From Profile
                Where Name = 'Admin' LIMIT 1];
UserRole r = [Select Name
                From UserRole
                Where Name = 'Admin Role' LIMIT 1];

User u = Objs.createUser(p.Name, r.Name)
```

The *roleName* can be set to *null* if you do not want to set a role for the developer

```
Profile p = [Select Name
                From Profile
```

```
                    Where Name = 'Admin' LIMIT 1];
    User u = Objs.createUser(p.Name, null)
```

User Default Field Values

The following is a table for the default values set for the developer and may change without notice. If you require these values to be different please update the values prior to inserting the record.

Field	Value
UserName	test_{random math value}@test.com
FirstName	Test-First
LastName	Test-Last
Alias	Test
Email	test_{random math value}@test.com
CommnityNickName	Random 6 Digits
TimeZoneSidKey	America/New_York
LocaleSidKey	en_US
EmailEncodingKey	UTF-8
LanguageLocaleKey	en_US

Refactoring Existing Code

Fill in sObject required Fields

If the developer creates a record using standard Salesforce® object record instantiation SEED for Test Methods can still be utilized to populate missing required field values either currently present or that may be created later.

Signature: sObject fill_sObject_Required(sObject *obj,* Boolean *doInsert*)

This method is a great starting point for refactoring code for use with SEED for Test Methods. Existing test methods can me refactored quickly to use SEED for Test methods as follows:

1. Create a class level property to instantiate SEED for Test Methods Obj_Creation code:

```
@isTest
private class YOURTESTCLASSNAME{

      seed.Obj_Creation Objs = New seed.Obj_Creation();

      ..
            YOUR METHODS
      ..


}
```

Continued on next page

2. Inside of each test method where you create object records use the fill_sObject_Required method **BEFORE** inserting the records to complete the field values that may be required but have not been previously set

```
private static testmethod void YOURTESTMETHODNAME{

      /* Existing code */
      Account a = New Account(Name = 'Test');
      Insert a;

      /* Refactored Code */
      Account a = New Account(Name = 'Test');
      Objs.fill_sObject_Required(a,true);
}
```

If the object has record types defined you must call the setRecordTypeName() method first even if you previously did not specify a record type. An example is below

```
private static testmethod void YOURTESTMETHODNAME{

      /* Existing code */
      RecordType rt = [Select Name From RecordType
                       Where sObjectType = 'Account'
                       And Name = 'RT_Name' LIMIT 1];
      Account a = New Account(Name = 'Test',
                              RecordtypeID = rt.id);
      Insert a;

      /* Refactored Code */
      Account a = New Account(Name = 'Test');
      Objs.setRecordTypeName('RT_Name');
      Objs.fill_sObject_Required(a,true);
}
```

Continued on next page

3. For those object records that you created in bulk this is an example of refactoring that code

```
private static testmethod void YOURTESTMETHODNAME{

      /* Existing code */
      RecordType rt = [Select Name From RecordType
                       Where sObjectType = 'Account'
                       And Name = 'RT_Name' LIMIT 1];
      Account[] aList = New Account[]{};

      For(integer x = 0; x<200 ; x++){

            Account a = New Account(Name = 'Test ' + x,
                                    RecordTypeID = rt.id)
                        );
            aList.add(a);

      }
      Insert aList;

      /* Refactored Code */
      Account[] aList = New Account[]{};
      Objs.setRecordTypeName('RT_Name');

      For(integer x = 0; x<200 ; x++){

            Account a = New Account(Name = 'Test ' + x,
                                    RecordTypeID = rt.id)
                        );
            aList.add(
            (Account)objs.fill_sObject_Requried(a,false);
            );

      }

      insert aList;
}
```

Technical Specification

Order of Field Value Population

SEED for Test Methods handles population of required field in the following order.

1. Developer Specifying field values in the createRecord() method call.
2. Field values defined in the recordTypeDefs class
3. Values identified by SEED for Test Methods according to organization validation rules when possible.
4. Default Values hardcoded in the SEED for Test Methods application

The execution happens in the order above and stops after one of the methods has been consumed for that record. For example, if the developer specifies field values in the createRecord() method call that is the final evaluation and value for that field during that createRecord() call. Execution order is reset for each call to the createRecord() method

Continued on next page

Hardcoded Default Field Values by Type

Field Type	Hardcoded Default Value
Double	100.0000000000 (Truncated according to scale and precision)
Decimal	Same as Double
Currency	Same as Double
Percent	23.35 (Truncated according to scale and precision)
Phone	8005551212
Email	{Unique 20 character value}@testmethod923.com
MultiPicklist	String One;String Two
ComboBox	String One;String Two
Picklist	String One
Text Area	Test String in a Text Area
URL	http://www.google.com
Time	System.now().time() value
DateTime	System.now() value
Integer	1
Reference	IF OwnerID = developerInfo.getUserID() Otherwise – Throws an Error as this value needs to be set by the developer creating the record with a required relationship as in Master Detail Relationships
String	Record Name = 'Record Name; Unique String = A unique value filling the full length of the field Otherwise = 'Test String'
Boolean	False
Restricted Picklist	Currently this field type must be specified when calling the createRecord() method. This is due to the limit on picklist describes. In Summer 14' release these limits were removed and future version may change the functionality related to all picklists

Continued on next page

Validation Rules

IMPORTANT NOTE:

Validation rules can be extremely complex and sometimes the logic between several different rules can cause seed to be unable to set the values to meet all the rules. When this occurs a test method may fail.

The resolution is simple: set the field values using the recordTypeDefs class and the values will be applied to all test methods written using SEED for Test Methods functionality and the value have not been specified in the createRecord() call.

Currently Handled by SEED for Test Methods

The table below list the validation function that SEED for Test Methods can currently process. Each time a new validation rule is created or and existing rule is updated that was previously handled by SEED for Test Methods to include functions not in this list, an email notification will be sent. (See Serialize Validation Rules : Notification).

Function Type	Example
Simple Rule	Name != 'Record Name'
IF	
IF (With Logic Reversal)	True part returns false and false part returns true IF(Name = 'Record Name',false,true)
Nested IF	If(Name = 'Name',If(email = null,true,false),false)
AND	AND(Name = 'Name',Email = 'Null')
&&	Name = 'Name' && 'Email' = Null
OR	OR(Name = 'Name',Email = 'Null')
||	Name = 'Name' || Email = Null
Nested AND / OR	AND(Name = 'Name',OR(Email = Null, Name = 'Test')
NOT	NOT(Name = 'Name')
Nested NOT / AND / OR	Any combination of the above nested in any acceptable combination
Mixed && / ||	&& is evaluated first x || y && z || c is evaluated as OR(x, AND(y,z),c)
ISBLANK	ISBLANK(Name)
CONTAINS	CONTAINS(Name,'Test')
INCLUDES	INCLUDES(Hobbies__c,'Test')
ISPICKVAL	ISPICKVAL(StageName,'Prospecting')
ISNEW()	No evaluation but continues processing
+, -	**NOT Supported**

How SEED Handles Field Values when Applying Validation Logic

Depending on the field type and the operator used seed will set the default value as follows if the hardcoded default values do not evaluate to **FALSE** according to the validation rule:

For example:

- The default value for a String is 'Test String'
- The validation rule specifies Name = 'Test String'
- The rule will evaluate to TRUE and a validation error will occur
- SEED for Test Methods will modify the value as specified below

Validation Rule Operator	Field Type	Return Value
=	String	Unique String
	Date	Date.today().addDays(-1)
	DateTime	Date.today().addDays(-1)
	Decimal	Unique Decimal
	Boolean	Opposite Value
!=	Any	Compare Value in Rule
>	Date	Compare Value – 1 day
	Date Time	Compare Value – 1 day
	Decimal	Compare Value – 1 or 0.1 depending on the scale
	String	Not Supported
<	Date	Compare Value + 1 day
	Date Time	Compare Value + 1 day
	Decimal	Compare Value + 1 or 0.1 depending on scale
	String	Not Supported
<>		Same as !=

Obj_Creation Consolidated Class Signatures

This is a condensed version of Obj_Creation Class and Refactoring Existing Code sections.

Name	Argument	Return Type
createRecord		sObject
	Boolean *doInsert*	sObject
	Schema.sObjectField *so* Map<String,Object> *field_values*	sObject
	Schema.sObjectField *so* Map<String,Object> *field_values* Boolean *doInsert* Boolean *ignoreRequiredNulls*	sObject
getRecordsList	Schema.sObjectType *so*	List<sObject>
addRecordToList	sObject *so*	sObject
getRecordsMap	Schema.sObjectType *so*	Map<ID,sObject>
setRecordTypeName	Schema.sObjectType *so* String *rt*	
setIgnoreValidationRules		
createUser	String *profileName* *String roleName*	User
fill_sObject_Required	sObject *obj* Boolean *doInsert*	sObject

For a detailed description on each of these methods please review the Obj_Creation and Refactoring Existing Code Sections.