How to Automatically Create Snapshot Scorecards

Table of Contents

Copyright

Trademarks

Conventions

This document uses a number of typographical conventions to highlight text that refers to a user's interaction with Salesforce and/or FrontRunner:

- **`Courier New Bold`** font draws the reader's attention to user interface elements, such as buttons, command links and fields. Example: "Click the **`Save`** button."
- `Courier New` font indicates values that need to be entered by the user. Example: "Enter `Profiles` in the **`Quick Find`** box."
- *`Italicised Courier New`* font indicates values that are displayed by Salesforce or FrontRunner. Example: "Check that the user is assigned the *`Standard User`* profile."

Document Version

This document is compatible with Kraytix FrontRunner Version 1.11.

1 Introduction

Kraytix FrontRunner allows you to manually create snapshots of scorecards to show how your lead qualification is trending over time.

> ***Note**: For further information about manually creating snapshot scorecards, please refer to the Kraytix FrontRunner - User Guide, which can be found on the Kraytix website (http://www.kraytix.com/) or on the Kraytix FrontRunner Salesforce® AppExchange® listing (https://appexchange.salesforce.com/listingDetail?listingId=a0N3A00000DvLDWUA3).*

However, relying on users to manually create snapshots is both time consuming and error prone; how can you be sure that your trending history is always complete and up to date? This note will show you how to use the Apex Scheduler and Batch Apex to automatically create snapshots of all scorecards connected to unconverted leads and opportunities that have not been closed yet.

> ***Note**: This note describes how to create Apex classes and Apex test classes. It is not possible to create these classes directly in your production Salesforce org. The classes should be created and tested in a sandbox and then sent to your production org using change sets. For more information about change sets, please refer to the Salesforce documentation at: https://help.salesforce.com/articleView?id=changesets.htm*

2 Step 1: Create a Batch Apex Class

Batch Apex allows you to process potentially large numbers of records as a background task.

> ***Note**: For more information about Batch Apex, please refer to the Salesforce documentation at: https://developer.salesforce.com/docs/atlas.en-us.apexcode.meta/apexcode/apex_batch.htm*

Below is the minimum viable Batch Apex code to create snapshots of all scorecards that are connected to unconverted leads or opportunities that have not closed yet. From **`Setup`** enter `Apex Classes` in the **`Quick Find`** box, then select **`Apex Classes`**. You will now see the **`Apex Classes`** page. Click the **`New`** button. Cut and paste this code into the new Apex Class. Finally, click the **`Save`** button.

```
global class ktxAutoSnapshotting implements Database.Batchable<sObject>
{
    // Only snapshot scorecards that are attached to unconverted leads or
opportunities that have not closed yet.
    global final String query = 'SELECT Id from ktxFR__ktxScorecard__c
WHERE (ktxFR__Lead__c != null AND ktxFR__Lead__r.IsConverted = false) OR
(ktxFR__Opportunity__c != null AND (NOT ktxFR__Opportunity__r.StageName
```



```
LIKE \'%closed%\'))';

    global ktxAutoSnapshotting()
    {
    }

    global Database.QueryLocator start(Database.BatchableContext BC)
    {
        return Database.getQueryLocator(query);
    }

    global void execute(Database.BatchableContext BC,
List<ktxFR__ktxScorecard__c> scorecards)
    {
        // Convert the list of scorecards to a list of their Ids.
        List<Id> scorecardIds = new List<Id>(new Map<Id,
ktxFR__ktxScorecard__c>(scorecards).keySet());

        // Create the snapshot scorecards using the FrontRunner API.
        ktxFR.ktxScorecardService.createSnapshotScorecards(scorecardIds);
    }

    global void finish(Database.BatchableContext BC)
    {
    }
}
```

***Note**: You may wish to add exception handling to the above sample code. For further information about FrontRunner exceptions please refer to Appendix B: FrontRunner API Reference*

***Note**: Appendix A: Test Classes includes a test class ktxAutoSnapshottingTest that can be created to test the ktxAutoSnapshotting class in your sandbox and maintain code coverage in your production org.*

3 Step 2: Create an Apex Scheduler Class

To schedule the Batch Apex class created above to execute at regular intervals you need to create an Apex class that implements the **`Schedulable`** interface.

***Note**: For more information about the Apex Scheduler, please refer to the Salesforce documentation at: https://developer.salesforce.com/docs/atlas.en-us.apexcode.meta/apexcode/apex_scheduler.htm*

Below is the minimum viable Apex Scheduler class. From **`Setup`** enter `Apex Classes` in the **`Quick Find`** box, then select **`Apex Classes`**. You will now see the **`Apex Classes`**

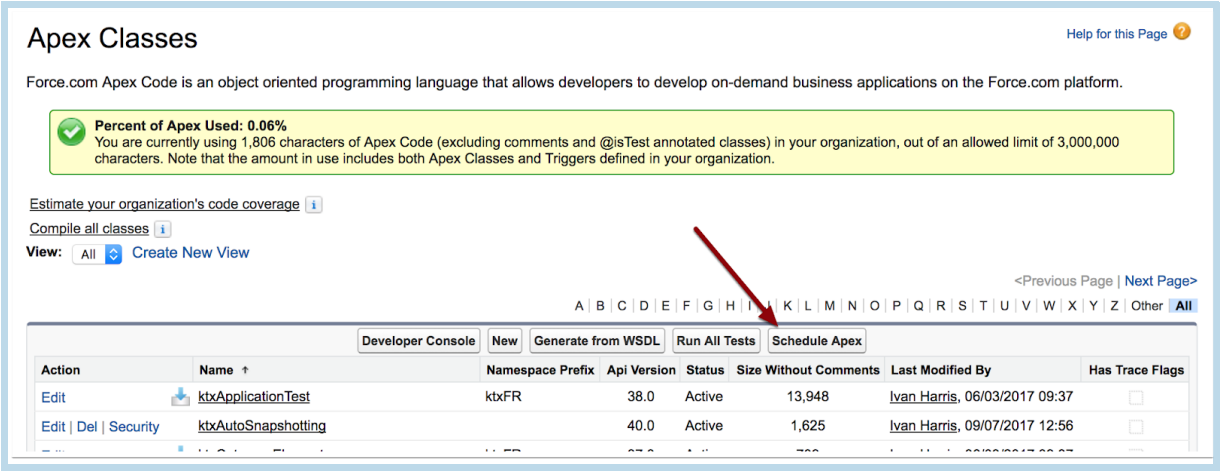
page. Click the **New** button. Cut and paste this code into the new Apex Class. Finally, click the **Save** button.

```
global class ktxScheduleAutoSnapshotting implements Schedulable {
    global void execute(SchedulableContext sc) {
        database.executebatch(new ktxAutoSnapshotting());
    }
}
```

***Note**: Appendix A: Test Classes includes a test class ktxScheduleAutoSnapshottingTest that can be created to test the ktxScheduleAutoSnapshotting class in your sandbox and maintain code coverage in your production org.*

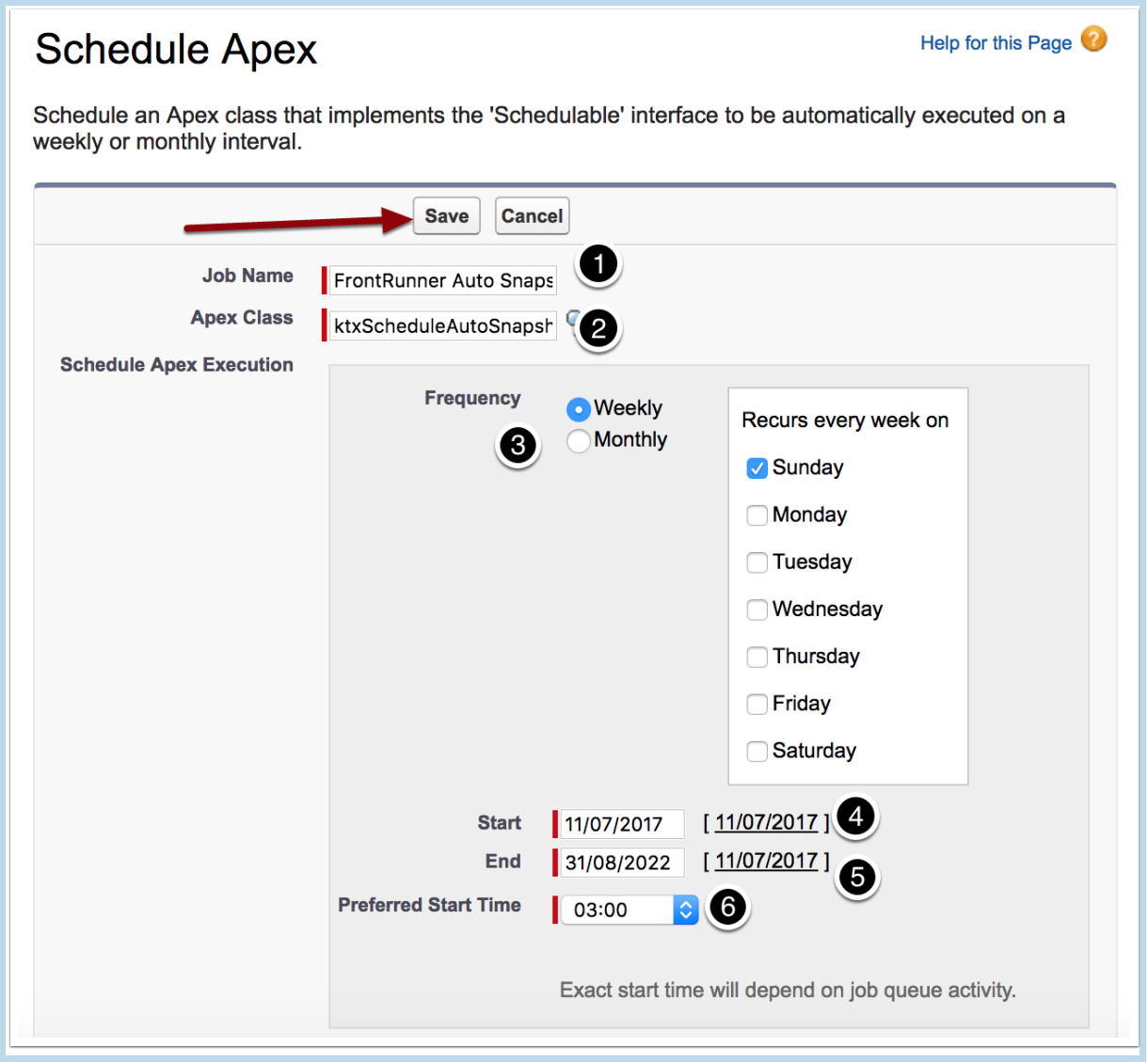
4 Step 3: Schedule the Apex

To schedule the Batch Apex class created above to execute at regular intervals, from **Setup** enter `Apex Classes` in the **Quick Find** box, then select **Apex Classes**. You will now see the **Apex Classes** page. Click the **Schedule Apex** button.



Enter the following information in the **Apex Scheduler** page, then click the **Save** button.

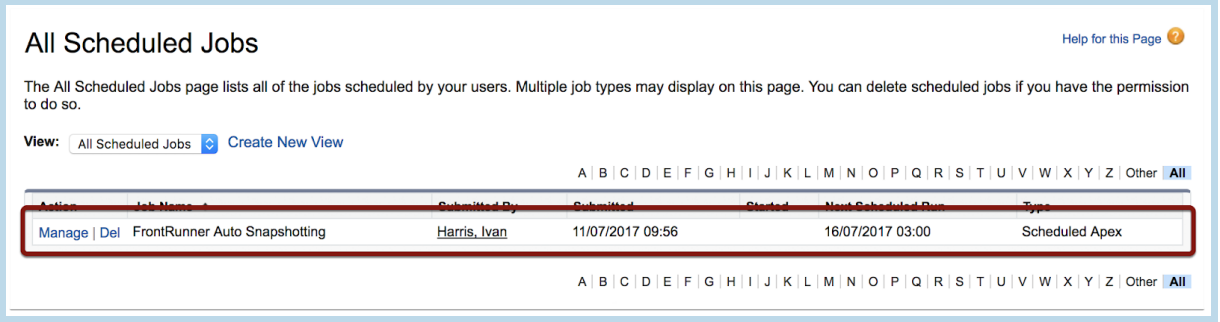
Field	Value
Job Name	`FrontRunner Auto Snapshotting`
Apex Class	`ktxScheduleAutoSnapshotting`
Frequency	`Weekly, Sunday`
Start	Choose the date that you wish the schedule to start
End	Choose the date that you wish the schedule to end
Preferred Start Time	Choose the time that you wish the schedule to run



***Note**: Most organisations set its snapshot time to be at the end of the business week, which varies depending on your country. For further information about business week start and end days by country, please refer to the Salesforce documentation at: https://help.salesforce.com/articleView?id=000338932&type=1&mode=1.*

5 Step 4: Validation

To validate that the scheduled job has been created, from **`Setup`** enter `Scheduled Jobs` in the **`Quick Find`** box, then select **`Scheduled Jobs`**. You will now see the **`Scheduled Jobs`** page where you will see the scheduled Apex job that you created.



Once the date and time set for the **`Next Scheduled Run`** has passed, select a scorecard that is connected to an unconverted lead or to an opportunity that has not closed yet and observe that a new snapshot scorecard has been created.

Appendix A: Test Classes

A.1 ktxAutoSnapshottingTest

To test the ktxAutoSnapshotting class in your sandbox and to maintain code coverage in your production org you can create the following test class.

```
@isTest
public class ktxAutoSnapshottingTest
{
    @isTest public static void ktxAutoSnapshotting_PositiveTest1()
    {
        // Create a test Template Scorecard.
        ktxFR__ktxTemplateScorecard__c templateScorecard = new
ktxFR__ktxTemplateScorecard__c(
            Name = 'TEST',
            ktxFR__Active__c = true,
            ktxFR__Description__c = '',
            ktxFR__RAGAmberScore__c = 20,
            ktxFR__RAGGreenScore__c = 50,
            ktxFR__SortOrder__c = 0);
        insert templateScorecard;
        ktxFR__ktxTemplateCategory__c templateCategory = new
ktxFR__ktxTemplateCategory__c(
            Name = 'Test',
            ktxFR__Description__c = '',
            ktxFR__SortOrder__c = 0,
            ktxFR__TemplateScorecard__c = templateScorecard.Id);
        insert templateCategory;
        ktxFR__ktxTemplateCriterion__c templateCriterion = new
ktxFR__ktxTemplateCriterion__c(
            Name = 'Test',
            ktxFR__Description__c = '',
            ktxFR__SortOrder__c = 0,
            ktxFR__TemplateCategory__c = templateCategory.Id);
        insert templateCriterion;
        ktxFR__ktxTemplateCriterionOutcome__c templateCriterionOutcome =
new ktxFR__ktxTemplateCriterionOutcome__c(
            Name = 'Test',
            ktxFR__Score__c = 100,
            ktxFR__TemplateCriterion__c = templateCriterion.Id);
        insert templateCriterionOutcome;

        // Create a test Lead.
        Lead testLead = new Lead(
            FirstName = 'John',
            LastName = 'Smith',
            Company = 'Test');
        insert testLead;

        // Create a test Scorecard.
        ktxFR__ktxScorecard__c scorecard = new ktxFR__ktxScorecard__c(
```



```
            Name = 'Test',
            ktxFR__Description__c = '',
            ktxFR__Lead__c = testLead.Id,
            ktxFR__MaxScore__c = 100,
            ktxFR__MinScore__c = 0,
            ktxFR__Opportunity__c = null,
            ktxFR__RAGAmberScore__c = 20,
            ktxFR__RAGGreenscore__c = 50,
            ktxFR__Score__c = 0,
            ktxFR__TemplateScorecard__c = templateScorecard.Id);
        insert scorecard;
        ktxFR__ktxCategory__c category = new ktxFR__ktxCategory__c(
            Name = 'Test',
            ktxFR__Description__c = '',
            ktxFR__MaxScore__c = 100,
            ktxFR__MinScore__c = 0,
            ktxFR__RAGAmberScore__c = 20,
            ktxFR__RAGGreenScore__c = 50,
            ktxFR__Score__c = 0,
            ktxFR__Scorecard__c = scorecard.Id,
            ktxFR__SortOrder__c = 0,
            ktxFR__TemplateCategory__c = templateCategory.Id);
        insert category;
        ktxFR__ktxCriterion__c criterion = new ktxFR__ktxCriterion__c(
            Name = 'Test',
            ktxFR__Category__c = category.Id,
            ktxFR__Description__c = '',
            ktxFR__MaxScore__c = 100,
            ktxFR__MinScore__c = 0,
            ktxFR__Outcome__c = '',
            ktxFR__RAGAmberScore__c = 20,
            ktxFR__RAGGreenScore__c = 50,
            ktxFR__Score__c = 0,
            ktxFR__SelectedCriterionOutcome__c = null,
            ktxFR__SortOrder__c = 0,
            ktxFR__TemplateCriterion__c = templateCriterion.Id);
        insert criterion;
        ktxFR__ktxCriterionOutcome__c criterionOutcome = new
ktxFR__ktxCriterionOutcome__c(
            Name = 'Test',
            ktxFR__Criterion__c = criterion.Id,
            ktxFR__Score__c = 100,
            ktxFR__TemplateCriterionOutcome__c =
templateCriterionOutcome.Id);
        insert criterionOutcome;

        // Validate that the test scorecard has no snapshot scorecards.
        ktxFR__ktxScorecard__c queriedScorecardBefore =
            [SELECT Id,
                 (SELECT Id
                  FROM ktxFR__SnapshotScorecards__r)
             FROM ktxFR__ktxScorecard__c
             WHERE Id = :scorecard.Id];
        System.assertNotEquals(null, queriedScorecardBefore);
        System.assertEquals(0,
queriedScorecardBefore.ktxFR__SnapshotScorecards__r.size());
```



```
        // Execute the Batch Apex job.
        Test.startTest();
        ktxAutoSnapshotting obj = new ktxAutoSnapshotting();
        DataBase.executeBatch(obj);
        Test.stopTest();

        // Validate that the Batch Apex job added a snapshot scorecard to
the test scorecard.
        ktxFR__ktxScorecard__c queriedScorecardAfter =
            [SELECT Id,
                 (SELECT Id
                  FROM ktxFR__SnapshotScorecards__r)
             FROM ktxFR__ktxScorecard__c
             WHERE Id = :scorecard.Id];
        System.assertNotEquals(null, queriedScorecardAfter);
        System.assertEquals(1,
queriedScorecardAfter.ktxFR__SnapshotScorecards__r.size());
    }
}
```

A.2 ktxScheduleAutoSnapshottingTest

To test the ktxScheduleAutoSnapshotting class in your sandbox and to maintain code coverage in your production org you can create the following test class.

```
@isTest
public class ktxScheduleAutoSnapshottingTest
{
    private static String JOB_NAME = 'Test ktxScheduleAutoSnapshotting';
    private static String CRON_EXP = '0 0 23 * * ?';

    @isTest public static void
ktxScheduleAutoSnapshotting_PositiveTest1()
    {
        // Schedule the Apex job.
        Test.StartTest();
        String jobId = System.schedule(
            JOB_NAME,
            CRON_EXP,
            new ktxScheduleAutoSnapshotting());
        Test.stopTest();

        // Get information about the scheduled Apex job.
        CronTrigger ct =
            [SELECT CronJobDetail.Id, CronJobDetail.Name,
CronJobDetail.JobType, CronExpression
             FROM CronTrigger
             WHERE Id = :jobId];

        // Validate the scheduled Apex job.
        System.assertNotEquals(null, ct);
        System.assertNotEquals(null, ct.CronJobDetail.Id);
        System.assertEquals(JOB_NAME, ct.CronJobDetail.Name);
```



```
        System.assertEquals('7', ct.CronJobDetail.JobType);
        System.assertEquals(CRON_EXP, ct.CronExpression);
    }
}
```


Appendix B: FrontRunner API Reference

B.1 ktxScorecardService Class

B.1.1 Namespace Prefix

ktxFR

B.1.2 Exceptions

ktxScorecardServiceException

Thrown if method parameter validation fails or when a built-in exception is caught and rethrown.

B.1.3 Methods

B.1.3.1 createSnapshotScorecards(scorecardIds)

Creates a snapshot scorecard for each of the scorecards identified in the supplied Id list.

Signature

```
global static List<Id> createSnapshotScorecards(List<Id> scorecardIds)
```

Parameters

Type	Name	Description
`List<Id>`	`scorecardIds`	List of scorecard Ids for which snapshot scorecards are to be created

Return Value

Type	Description
`List<Id>`	The Ids of the created snapshot scorecards

