



System Class

Contains methods for system operations, such as writing debug messages and scheduling jobs.

Namespace

[System](#)

System Methods

The following are methods for `System`. All methods are static.

- **[abortJob\(jobId\)](#)**
Stops the specified job. If the job is currently executing, the stopped job is still visible in the job queue in the Salesforce user interface. The specified job is stopped, but any code that is in progress will continue to execute until it completes.
- **[assert\(condition, msg\)](#)**
Asserts that the specified condition is true. If it isn't, a fatal error is returned that causes code execution to halt.
- **[assertEquals\(expected, actual, msg\)](#)**
Asserts that the first two arguments are the same. If they aren't, a fatal error is returned that causes code execution to halt.
- **[assertNotEquals\(expected, actual, msg\)](#)**
Asserts that the first two arguments are different. If they're the same, a fatal error is returned that causes code execution to halt.
- **[currentPageReference\(\)](#)**
Returns a reference to the current page. This is used with Visualforce pages.
- **[currentTimeMillis\(\)](#)**
Returns the current time in milliseconds, which is expressed as the difference between the current time and midnight, January 1, 1970 UTC.
- **[debug\(msg\)](#)**
Writes the specified message, in string format, to the execution debug log. The `DEBUG` log level is used.
- **[debug\(logLevel, msg\)](#)**
Writes the specified message, in string format, to the execution debug log with the specified log level.
- **[enqueueJob\(queueableObj\)](#)**
Adds a job to the Apex job queue that corresponds to the specified queueable class and returns the job ID.
- **[enqueueJob\(queueable, delay\)](#)**
Adds a job to the Apex job queue that corresponds to the specified queueable class and returns the job ID. The job is scheduled with a specified minimum delay (0–10 minutes). The delay is ignored during Apex testing.
- **[enqueueJob\(queueable, asyncOptions\)](#)**
Adds a job to the Apex job queue that corresponds to the specified queueable class and returns the job ID. Specify a unique signature for your queueable job, the maximum stack depth or the minimum queue delay in the `asyncOptions` parameter.
- **[equals\(obj1, obj2\)](#)**
Returns `true` if both arguments are equal. Otherwise, returns `false`.



returns the short code for the Quality value of the current request object.

- **hashCode(obj)**
Returns the hash code of the specified object.
- **isBatch()**
Returns `true` if a batch Apex job invoked the executing code, or `false` if not. In API version 35.0 and earlier, also returns `true` if a queueable Apex job invoked the code.
- **isFunctionCallback()**
Returns `true` if an asynchronous Salesforce Function callback invoked the executing code, or `false` if not. Available in API version 51.0 and later.
- **isFuture()**
Returns `true` if the currently executing code is invoked by code contained in a method annotated with `future`; `false` otherwise.
- **isQueueable()**
Returns `true` if a queueable Apex job invoked the executing code. Returns `false` if not, including if a batch Apex job or a future method invoked the code.
- **isRunningElasticCompute()**
Reserved for future use.
- **isScheduled()**
Returns `true` if the currently executing code is invoked by a scheduled Apex job; `false` otherwise.
- **movePassword(targetUserId,sourceUserId)**
Moves the specified user's password to a different user.
- **now()**
Returns the current date and time in the GMT time zone.
- **pauseJobById(cronTriggerId)**
Pause a scheduled Apex job specified by its CronTrigger ID.
- **pauseJobByName(jobName)**
Pause a scheduled Apex job specified by its name.
- **process(workItemIds, action, comments, nextApprover)**
Processes the list of work item IDs.
- **purgeOldAsyncJobs(dt)**
Deletes asynchronous Apex job records for jobs that have finished execution before the specified date with a Completed, Aborted, or Failed status, and returns the number of records deleted.
- **requestVersion()**
Returns a two-part version that contains the major and minor version numbers of a package. Applies to first-generation managed packages.
- **resetPassword(userId, sendUserEmail)**
Resets the password for the specified user.
- **resetPasswordWithEmailTemplate(userId, sendUserEmail, emailTemplateName)**
Resets the user's password and sends an email to the user with their new password. You specify the email template that is sent to the specified user. Use this method for external users of Experience Cloud sites.
- **resumeJobById(cronTriggerId)**
Resume a paused scheduled Apex job specified by its CronTrigger ID.
- **resumeJobByName(jobName)**
Resumes a paused scheduled Apex job specified by its name.
- **runAs(version)**
Changes the current package version to the package version specified in the argument.
- **runAs(userSObject)**
Changes the current user to the specified user.
- **schedule(jobName, cronExpression, schedulableClass)**
Use `schedule` with an Apex class that implements the `Schedulable` interface to schedule the class to run at the time specified by a Cron expression.



Schedules a batch job to run once in the future after the specified time interval, with the specified job name and scope size. Returns the scheduled job ID (CronTrigger ID).

- **[setPassword\(userId, password\)](#)**
Sets the password for the specified user.
- **[submit\(workItemIds, comments, nextApprover\)](#)**
Submits the processed approvals. The current user is the submitter and the entry criteria is evaluated for all processes applicable to the current user.
- **[today\(\)](#)**
Returns the current date in the current user's time zone.

abortJob(jobId)

Stops the specified job. If the job is currently executing, the stopped job is still visible in the job queue in the Salesforce user interface. The specified job is stopped, but any code that is in progress will continue to execute until it completes.

Signature

```
public static Void abortJob(String jobId)
```

Parameters

jobId

Type: [String](#)

The *jobId* is the ID associated with an [AsyncApexJob](#) ID for batch or future Apex jobs, or a [CronTrigger](#) ID for scheduled Apex jobs. You can't abort a scheduled Apex job using an AsyncApexJob ID.

Return Value

Type: Void

Usage

The following methods return the job ID that can be passed to `abortJob`.

- [System.schedule method](#)—returns the CronTrigger object ID associated with the scheduled job as a string.
- [SchedulableContext.getTriggerId method](#)—returns the CronTrigger object ID associated with the scheduled job as a string.
- [getJobId method](#)—returns the AsyncApexJob object ID associated with the batch job as a string.
- [Using Batch Apex Database.executeBatch method](#)—returns the AsyncApexJob object ID associated with the batch job as a string.

assert(condition, msg)

Asserts that the specified condition is true. If it isn't, a fatal error is returned that causes code execution to halt.



Important

We recommend that you use the methods of the [Assert Class](#) rather than this method. The `System.Assert` class provides methods that handle all types of logical assertions and comparisons, which improve the clarity of your Apex code.

Signature



Type: [Boolean](#)

msg

Type: Object

(Optional) Custom message returned as part of the error message.

Return Value

Type: Void

Usage

You can't catch an assertion failure using a try/catch block even though it's logged as an exception.

assertEquals(expected, actual, msg)

Asserts that the first two arguments are the same. If they aren't, a fatal error is returned that causes code execution to halt.



Important

We recommend that you use the methods of the [Assert Class](#) rather than this method. The `System.Assert` class provides methods that handle all types of logical assertions and comparisons, which improve the clarity of your Apex code.

Signature

```
public static Void assertEquals(Object expected, Object actual, Object msg)
```

Parameters

expected

Type: Object

Specifies the expected value.

actual

Type: Object

Specifies the actual value.

msg

Type: Object

(Optional) Custom message returned as part of the error message.

Return Value

Type: Void

Usage

You can't catch an assertion failure using a try/catch block even though it's logged as an exception.

assertNotEquals(expected, actual, msg)

Asserts that the first two arguments are different. If they're the same, a fatal error is returned that causes code execution to halt.



comparisons, which improve the clarity of your Apex code.

Signature

```
public static Void assertNotEquals(Object expected, Object actual, Object msg)
```

Parameters

expected

Type: Object

Specifies the expected value.

actual

Type: Object

Specifies the actual value.

msg

Type: Object

(Optional) Custom message returned as part of the error message.

Return Value

Type: Void

Usage

You can't catch an assertion failure using a try/catch block even though it's logged as an exception.

currentPageReference()

Returns a reference to the current page. This is used with Visualforce pages.

Signature

```
public static System.PageReference currentPageReference()
```

Return Value

Type: [System.PageReference](#)

Usage

For more information, see [PageReference Class](#).

currentTimeMillis()

Returns the current time in milliseconds, which is expressed as the difference between the current time and midnight, January 1, 1970 UTC.

Signature

```
public static Long currentTimeMillis()
```

Return Value

Type: [Long](#)

debug(msg)



```
public static Void debug(Object msg)
```

Parameters

msg

Type: Object

Return Value

Type: Void

Usage

If the *msg* argument is not a string, the `debug` method calls `String.valueOf` to convert it into a string. The `String.valueOf` method calls the `toString` method on the argument, if available, or any overridden `toString` method if the argument is a user-defined type. Otherwise, if no `toString` method is available, it returns a string representation of the argument.

If the log level for Apex Code is set to `DEBUG` or higher, the message of this debug statement will be written to the debug log.

Note that when a map or set is printed, the output is sorted in key order and is surrounded with square brackets (`[]`). When an array or list is printed, the output is enclosed in parentheses (`()`).

Note

Calls to `System.debug` are not counted as part of Apex code coverage. Calls to `System.debug` are not counted as part of Apex code coverage.

For more information on log levels, see [Debug Log Levels](#) in the Salesforce online help.

debug(logLevel, msg)

Writes the specified message, in string format, to the execution debug log with the specified log level.

Signature

```
public static Void debug(LoggingLevel logLevel, Object msg)
```

Parameters

logLevel

Type: [LoggingLevel Enum](#)

The logging level to set for this method.

msg

Type: Object

The message or object to write in string format to the execution debug log.

Return Value

Type: Void

Usage

If the *msg* argument is not a string, the `debug` method calls `String.valueOf` to convert it into a string. The `String.valueOf` method calls the `toString` method on the argument, if available, or any overridden `toString` method if the argument is a user-defined type. Otherwise, if no `toString` method is available, it returns a string representation of the argument.



For more information on log levels, see [Debug Log Levels](#) in the Salesforce online help.

enqueueJob(queueableObj)

Adds a job to the Apex job queue that corresponds to the specified queueable class and returns the job ID.

Signature

```
public static ID enqueueJob(Object queueableObj)
```

Parameters

queueableObj

Type: Object

An instance of the class that implements the [Queueable Interface](#).

Return Value

Type: [ID](#)

The job ID, which corresponds to the ID of an AsyncApexJob record.

Usage

To add a job for asynchronous execution, call `System.enqueueJob` by passing in an instance of your class implementation of the `Queueable` interface for execution as follows:

```
ID jobId = System.enqueueJob(new MyQueueableClass());
```

For more information about Queueable Apex, including information about limits, see [Queueable Apex](#).

enqueueJob(queueable, delay)

Adds a job to the Apex job queue that corresponds to the specified queueable class and returns the job ID. The job is scheduled with a specified minimum delay (0–10 minutes). The delay is ignored during Apex testing.

Signature

```
public static Id enqueueJob(Object queueable, Integer delay)
```

Parameters

queueable

Type: Object

An instance of the class that implements the [Queueable Interface](#).

delay

Type: [Integer](#)

The minimum delay (0–10 minutes) before the queueable job is scheduled for execution.

The delay is ignored during Apex testing.



Warning



Return Value

Type: [Id](#)

The job ID, which corresponds to the ID of an AsyncApexJob record.

Example

This example adds a job for delayed asynchronous execution by passing in an instance of your class implementation of the [Queueable](#) interface for execution. There's a minimum delay of 5 minutes before the job is executed.

```
Integer delayInMinutes = 5;
ID jobId = System.enqueueJob(new MyQueueableClass(), delayInMinutes);
```

For more information about Queueable Apex, including information about limits, see [Queueable Apex](#).

enqueueJob(queueable, asyncOptions)

Adds a job to the Apex job queue that corresponds to the specified queueable class and returns the job ID. Specify a unique signature for your queueable job, the maximum stack depth or the minimum queue delay in the [asyncOptions](#) parameter.

Signature

```
public static Id enqueueJob(Object queueable, Object asyncOptions)
```

Parameters

queueable

Type: [Object](#)

An instance of the class that implements the [Queueable Interface](#).

asyncOptions

Type: [AsyncOptions](#)

Specify a unique signature for your queueable job, the maximum stack depth, or a minimum queue delay in the [AsyncOptions](#) class properties.

Return Value

Type: [Id](#)

The job ID, which corresponds to the ID of an AsyncApexJob record.

Usage

The [System.AsyncInfo](#) class methods help you determine if maximum stack depth is set in your Queueable request and get the stack depths and queue delay for queueables that are currently running. Use information about the current queueable execution to make decisions on adjusting delays on subsequent calls.

These are methods in the [System.AsyncInfo](#) class.

- [hasMaxStackDepth\(\)](#)
- [getCurrentQueueableStackDepth\(\)](#)
- [getMaximumQueueableStackDepth\(\)](#)



`equals(obj1, obj2)`

Returns `true` if both arguments are equal. Otherwise, returns `false`.

Signature

```
public static Boolean equals(Object obj1, Object obj2)
```

Parameters

obj1

Type: `Object`

Object being compared.

obj2

Type: `Object`

Object to compare with the first argument.

Return Value

Type: [Boolean](#)

Usage

obj1 and *obj2* can be of any type. They can be values, or object references, such as `sObjects` and user-defined types.

The comparison rules for `System.equals` are identical to the ones for the `==` operator. For example, string comparison is case insensitive. For information about the comparison rules, see [the == operator](#).

`getApplicationReadWriteMode()`

Returns the read write mode set for an organization during Salesforce.com upgrades and downtimes.

Signature

```
public static System.ApplicationReadWriteMode getApplicationReadWriteMode()
```

Return Value

Type: [System.ApplicationReadWriteMode](#)

Valid values are:

- `DEFAULT`
- `READ_ONLY`

Using the `System.ApplicationReadWriteMode` Enum

Use the `System.ApplicationReadWriteMode` enum returned by the `getApplicationReadWriteMode` to programmatically determine if the application is in read-only mode during Salesforce upgrades and downtimes.

Valid values for the enum are:

- `DEFAULT`
- `READ_ONLY`

Example:



```

    if (mode == ApplicationReadWriteMode.READ_ONLY) {
        // Do nothing. If DML operation is attempted in readonly mode,
        // InvalidReadOnlyUserDmlException will be thrown.
    } else if (mode == ApplicationReadWriteMode.DEFAULT) {
        Account account = new Account(name = 'my account');
        insert account;
    }
}
}
}

```

getQuiddityShortCode(QuiddityValue)

Returns the short code for the Quiddity value of the current Request object.

Signature

```
public String getQuiddityShortCode(System.Quiddity QuiddityValue)
```

Parameters

QuiddityValue

Type: [System.Quiddity](#)

The Quiddity enum value that has an associated short code. This short code is used in Event Monitoring logs. For more information, see [Apex Execution Event Type](#).

Return Value

Type: [String](#)

hashCode(obj)

Returns the hash code of the specified object.

Signature

```
public static Integer hashCode(Object obj)
```

Parameters

obj

Type: [Object](#)

The object to get the hash code for. This parameter can be of any type, including values or object references, such as sObjects or user-defined types.

Return Value

Type: [Integer](#)

Versioned Behavior Changes

In API version 51.0 and later, the `hashCode()` method returns the same hash code for identical Id values. In API version 50.0 and earlier, identical Id values didn't always generate the same hash code value.

isBatch()

Returns `true` if a batch Apex job invoked the executing code, or `false` if not. In API version 35.0 and earlier, also returns `true` if a queueable Apex job invoked the code.

Signature

```
public static Boolean isBatch()
```



A batch Apex job can't invoke a future method. Before invoking a future method, use `isBatch()` to check whether the executing code is a batch Apex job.

isFunctionCallback()

Returns `true` if an asynchronous Salesforce Function callback invoked the executing code, or `false` if not. Available in API version 51.0 and later.

Signature

```
public static Boolean isFunctionCallback()
```

Return Value

Type: [Boolean](#)

Usage

Use this method to determine if the Apex code is being invoked as part of a callback from an asynchronous Salesforce Functions invocation. For more details on invoking Salesforce Functions from Apex, see [Functions Namespace](#)

isFuture()

Returns `true` if the currently executing code is invoked by code contained in a method annotated with `future`; `false` otherwise.

Signature

```
public static Boolean isFuture()
```

Return Value

Type: [Boolean](#)

Usage

Since a future method can't be invoked from another future method, use this method to check if the current code is executing within the context of a future method before you invoke a future method.

isQueueable()

Returns `true` if a queueable Apex job invoked the executing code. Returns `false` if not, including if a batch Apex job or a future method invoked the code.

Signature

```
public static Boolean isQueueable()
```

Return Value

Type: [Boolean](#)

Usage

```
public class SimpleQueueable implements Queueable {  
  
    String name;
```



```

        Account testAccount = new Account();
        testAccount.name = 'testAcc';
        insert(testAccount);
        System.assert(System.isQueueable()); //Should return true
    }
}

```

```

global class ComplexBatch implements Database.Batchable<SObject> {

    global Database.QueryLocator start(Database.BatchableContext info) {
        System.assert(!System.isQueueable()); //Should return false
        return Database.getQueryLocator([SELECT Id, Name FROM Account LIMIT 1]);
    }

    global void execute(Database.BatchableContext info, SObject[] scope) {
        System.assert(!System.isQueueable()); //Should return false
        System.enqueueJob(new SimpleQueueable('CallingFromComplexBatch'));
        System.assert(!System.isQueueable()); //Should return false
    }

    global void finish(Database.BatchableContext info) {
        System.assert(!System.isQueueable()); //Should return false
    }
}

```

isRunningElasticCompute()

Reserved for future use.

Signature

```
public static Boolean isRunningElasticCompute()
```

Return Value

Type: [Boolean](#)

isScheduled()

Returns true if the currently executing code is invoked by a scheduled Apex job; false otherwise.

Signature

```
public static Boolean isScheduled()
```

Return Value

Type: [Boolean](#)

movePassword(targetUserId,sourceUserId)

Moves the specified user's password to a different user.

Signature

```
public static Void movePassword(ID targetUserId, ID sourceUserId)
```

Parameters

targetUserId

Type: [ID](#)



The user that the password is moved from.

Return Value

Type: Void

Usage

Moving a password simplifies converting a user to another type of user, such as when converting an external user to a user with less restrictive access. If you require access to the `movePassword` method, contact Salesforce.

Keep in mind these requirements.

- The *targetUserId*, *sourceUserId*, and user performing the move operation must all belong to the same Salesforce org.
- The *targetUserId* and the *sourceUserId* cannot be the same as the user performing the move operation.
- A user without a password can't be specified as the *sourceUserId*. For example, a source user who has already had their password moved is left without a password. That user can't be a source user again.

After the password is moved:

- The target user can log in with the password.
- The source user no longer has a password. To enable logins for this user, a password reset is required.

now()

Returns the current date and time in the GMT time zone.

Signature

```
public static Datetime now()
```

Return Value

Type: [Datetime](#)

pauseJobById(cronTriggerId)

Pause a scheduled Apex job specified by its CronTrigger ID.

Signature

```
public static void pauseJobById(String cronTriggerId)
```

Parameters

cronTriggerId

Type: [String](#)

The scheduled job ID.

Return Value

Type: void

pauseJobByName(jobName)

Pause a scheduled Apex job specified by its name.

***jobName***Type: [String](#)**Return Value**

Type: void

process(workItemIds, action, comments, nextApprover)

Processes the list of work item IDs.

Signature

```
public static List<Id> process(List<Id> workItemIds, String action, String comments, String nextApprover)
```

Parameters***workItemIds***Type: [List<Id>](#)***action***Type: [String](#)***comments***Type: [String](#)***nextApprover***Type: [String](#)**Return Value**Type: [List<Id>](#)**purgeOldAsyncJobs(dt)**

Deletes asynchronous Apex job records for jobs that have finished execution before the specified date with a Completed, Aborted, or Failed status, and returns the number of records deleted.

Signature

```
public static Integer purgeOldAsyncJobs(Date dt)
```

Parameters***dt***Type: [Date](#)

Specifies the date up to which old records are deleted. The date comparison is based on the `CompletedDate` field of `AsyncApexJob`, which is in the GMT time zone.

Return ValueType: [Integer](#)**Usage**

Asynchronous Apex job records are records in [AsyncApexJob](#).

The system cleans up asynchronous job records for jobs that have finished execution and are older than seven days. You can use this method to further reduce the size of `AsyncApexJob` by cleaning



Example

This example shows how to delete all job records for jobs that have finished before today's date.

```
Integer count = System.purgeOldAsyncJobs
(Date.today());
System.debug('Deleted ' +
count + ' old jobs.');
```

requestVersion()

Returns a two-part version that contains the major and minor version numbers of a package. Applies to first-generation managed packages.

Signature

```
public static System.Version requestVersion()
```

Return Value

Type: [System.Version](#)

Usage

Using this method, you can determine the version of an installed instance of your package from which the calling code is referencing your package. Based on the version that the calling code has, you can customize the behavior of your package code.

The `requestVersion` method isn't supported for unmanaged packages. If you call it from an unmanaged package, an exception will be thrown.

resetPassword(userId, sendUserEmail)

Resets the password for the specified user.

Signature

```
public static System.ResetPasswordResult resetPassword(ID userId, Boolean sendUserEmail)
```

Parameters

userId

Type: [ID](#)

sendUserEmail

Type: [Boolean](#)

Return Value

Type: [System.ResetPasswordResult](#)

Usage

When the user logs in with the new password, they are prompted to enter a new password, and to select a security question and answer if they haven't already. If you specify `true` for `sendUserEmail`, the user is sent an email notifying them that their password was reset. A link to sign onto Salesforce using the new password is included in the email. Use [setPassword\(userId, password\)](#) if you don't want the user to be prompted to enter a new password when they log in.



Warning



emailTemplateName)

Resets the user's password and sends an email to the user with their new password. You specify the email template that is sent to the specified user. Use this method for external users of Experience Cloud sites.

Signature

```
public static System.ResetPasswordResult resetPasswordWithEmailTemplate(Id userId, Boolean
sendUserEmail, String emailTemplateName)
```

Parameters

userId

Type: [Id](#)

The ID of the user whose password was reset.

sendUserEmail

Type: [Boolean](#)

emailTemplateName

Type: [String](#)

Name of the email template.

Return Value

Type: [System.ResetPasswordResult](#)

Usage

If you specify `true` for *sendUserEmail*, specify the email template that is sent to the user notifying them that their password was reset. When the user logs in with the new password in the email, they are prompted to enter a new password. A link to sign onto Salesforce using the new password is included in the email. Use [setPassword\(userId, password\)](#) if you don't want the user to be prompted to enter a new password when they log in.



Warning

Be careful with this method, and do not expose this functionality to end-users.

resumeJobById(cronTriggerId)

Resume a paused scheduled Apex job specified by its CronTrigger ID.

Signature

```
public static void resumeJobById(String cronTriggerId)
```

Parameters

cronTriggerId

Type: [String](#)

The scheduled job ID.

Return Value

Type: `void`



resumeJobByName(jobName)

Resumes a paused scheduled Apex job specified by its name.

Signature

```
public static void resumeJobByName(String jobName)
```

Parameters

jobName

Type: [String](#)

Return Value

Type: void

Usage

If you resume a paused scheduled job, the job immediately runs one time. Subsequent executions of the job run according to the established schedule. Any scheduled executions that were missed while the job was paused don't run.

runAs(version)

Changes the current package version to the package version specified in the argument.

Signature

```
public static Void runAs(System.Version version)
```

Parameters

version

Type: [System.Version](#)

Return Value

Type: Void

Usage

A package developer can use [Version methods](#) to continue to support existing behavior in classes and triggers in previous package versions while continuing to evolve the code. Apex classes and triggers are saved with the version settings for each installed managed package that the class or trigger references.

This method is used for testing your component behavior in different package versions that you upload to the AppExchange. This method effectively sets a two-part version consisting of major and minor numbers in a test method so that you can test the behavior for different package versions.

You can only use `runAs` in a test method. There is no limitation to the number of calls to this method in a transaction. For sample usage of this method, see [Testing Behavior in Package Versions](#).

runAs(userSObject)

Changes the current user to the specified user.

Signature

```
public static Void runAs(User userSObject)
```



Return Value

Type: Void

Usage

All of the specified user's record sharing is enforced during the execution of `runAs`. You can only use `runAs` in a test method. For more information, see [Using the `runAs\(\)` Method](#).

Note

The `runAs` method ignores user license limits. You can create new users with `runAs` even if your organization has no additional user licenses.

The `runAs` method implicitly inserts the user that is passed in as parameter if the user has been instantiated, but not inserted yet.

You can also use `runAs` to perform mixed DML operations in your test by enclosing the DML operations within the `runAs` block. In this way, you bypass the mixed DML error that is otherwise returned when inserting or updating setup objects together with other sObjects. See [sObjects That Cannot Be Used Together in DML Operations](#).

Note

Every call to `runAs` counts against the total number of DML statements issued in the process.

`schedule(jobName, cronExpression, schedulableClass)`

Use `schedule` with an Apex class that implements the `Schedulable` interface to schedule the class to run at the time specified by a Cron expression.

Signature

```
public static String schedule(String jobName, String cronExpression, Object schedulableClass)
```

Parameters

jobName

Type: [String](#)

cronExpression

Type: [String](#)

schedulableClass

Type: Object

Return Value

Type: [String](#)

Returns the scheduled job ID (CronTrigger ID).

Usage

Use extreme care if you're planning to schedule a class from a trigger. You must be able to guarantee that the trigger won't add more scheduled classes than the limit. In particular, consider API bulk updates, import wizards, mass record changes through the user interface, and all cases



Salesforce schedules the class for execution at the specified time. Actual execution may be delayed based on service availability.

Using the `System.Schedule` Method

After you implement a class with the `Schedulable` interface, use the `System.Schedule` method to execute it. The scheduler runs as system—all classes are executed, whether or not the user has permission to execute the class.

Note

Use extreme care if you're planning to schedule a class from a trigger. You must be able to guarantee that the trigger won't add more scheduled classes than the limit. In particular, consider API bulk updates, import wizards, mass record changes through the user interface, and all cases where more than one record can be updated at a time.

The `System.Schedule` method takes three arguments: a name for the job, an expression used to represent the time and date the job is scheduled to run, and the name of the class. This expression has the following syntax:

Seconds Minutes Hours Day_of_month Month Day_of_week Optional_year

Note

Salesforce schedules the class for execution at the specified time. Actual execution may be delayed based on service availability.

The `System.Schedule` method uses the user's timezone for the basis of all schedules.

The following are the values for the expression:

Name	Values	Special Characters
<i>Seconds</i>	0-59	None
<i>Minutes</i>	0-59	None
<i>Hours</i>	0-23	, - * /
<i>Day_of_month</i>	1-31	, - * ? / L W
<i>Month</i>	1-12 or the following: <ul style="list-style-type: none">JANFEBMARAPRMAYJUNJULAUG	, - * /



- NOV
- DEC

Day_of_week 1–7 or the following: , - * ? / L #

- SUN
- MON
- TUE
- WED
- THU
- FRI
- SAT

optional_year null or 1970–2099 , - * /

The special characters are defined as follows:

Special Character	Description
,	Delimits values. For example, use JAN, MAR, APR to specify more than one month.
-	Specifies a range. For example, use JAN-MAR to specify more than one month.
*	Specifies all values. For example, if <i>Month</i> is specified as *, the job is scheduled for every month.
?	Specifies no specific value. This is only available for <i>Day_of_month</i> and <i>Day_of_week</i> , and is generally used when specifying a value for one and not the other.
/	Specifies increments. The number before the slash specifies when the intervals will begin, and the number after the slash is the interval amount. For example, if you specify 1/5 for <i>Day_of_month</i> , the Apex class runs every fifth day of the month, starting on the first of the month.
L	Specifies the end of a range (last). This is only available for <i>Day_of_month</i> and <i>Day_of_week</i> . When used with <i>Day of month</i> , L always means the last day of the month, such as January 31, February 29 for leap years, and so on. When used with <i>Day_of_week</i> by itself, it always means 7 or SAT. When used with a <i>Day_of_week</i> value, it means the last of that type of day in the month. For example, if you specify 2L, you are specifying the last Monday of the month. Do not use a range of values with L as the results might be unexpected.
W	Specifies the nearest weekday (Monday-Friday) of the given day. This is only available for <i>Day_of_month</i> . For example, if you specify 20W, and the 20th is a Saturday, the class runs on the 19th. If you specify 1W, and the first is a Saturday, the class does not run in the previous month, but on the third, which is the following Monday.

Tip
Use the L and W together to specify the last weekday of the month.




available for *Day_of_week*. The number before the # specifies weekday (SUN-SAT). The number after the # specifies the day of the month. For example, specifying 2#1 means the class runs on the first Monday of every month.

The following are some examples of how to use the expression.

Expression	Description
0 0 13 * * ?	Class runs every day at 1 PM.
0 0 22 ? * 6L	Class runs the last Friday of every month at 10 PM.
0 0 10 ? * MON-FRI	Class runs Monday through Friday at 10 AM.
0 0 20 * * ? 2010	Class runs every day at 8 PM during the year 2010.

In the following example, the class `proschedule` implements the `Schedulable` interface. The class is scheduled to run at 8 AM, on the 13 February.



```
proschedule p = new proschedule();
String sch = '0 0 8 13 2 ?';
system.schedule('One Time Pro', sch, p);
```

scheduleBatch(batchable, jobName, minutesFromNow)

Schedules a batch job to run once in the future after the specified time interval and with the specified job name.

Signature

```
public static String scheduleBatch(Database.Batchable batchable, String jobName, Integer minutesFromNow)
```

Parameters

batchable

Type: [Database.Batchable](#)

An instance of a class that implements the `Database.Batchable` interface.

jobName

Type: [String](#)

The name of the job that this method will start.

minutesFromNow

Type: [Integer](#)

The time interval in minutes after which the job should start executing. This argument must be greater than zero.

Return Value

Type: [String](#)

The scheduled job ID (CronTrigger ID).

Usage



the specified time. Actual execution occurs at or after that time, depending on service availability.

- The scheduler runs as system—all classes are executed, whether the user has permission to execute the class or not.
- When the job's schedule is triggered, the system queues the batch job for processing. If Apex flex queue is enabled in your org, the batch job is added at the end of the flex queue. For more information, see [Holding Batch Jobs in the Apex Flex Queue](#).
- All scheduled Apex limits apply for batch jobs scheduled using `System.scheduleBatch`. After the batch job is queued (with a status of `Holding` or `Queued`), all batch job limits apply and the job no longer counts toward scheduled Apex limits.
- After calling this method and before the batch job starts, you can use the returned scheduled job ID to abort the scheduled job using the [System.abortJob](#) method.

For an example, see [Using Batch Apex](#).

scheduleBatch(batchable, jobName, minutesFromNow, scopeSize)

Schedules a batch job to run once in the future after the specified the time interval, with the specified job name and scope size. Returns the scheduled job ID (CronTrigger ID).

Signature

```
public static String scheduleBatch(Database.Batchable batchable, String jobName, Integer minutesFromNow, Integer scopeSize)
```

Parameters

batchable

Type: [Database.Batchable](#)

The batch class that implements the `Database.Batchable` interface.

jobName

Type: [String](#)

The name of the job that this method will start.

minutesFromNow

Type: [Integer](#)

The time interval in minutes after which the job should start executing.

scopeSize

Type: [Integer](#)

The number of records that should be passed to the batch `execute` method.

Return Value

Type: [String](#)

Usage



Note



- The scheduler runs as system—all classes are executed, whether the user has permission to execute the class or not.
- When the job's schedule is triggered, the system queues the batch job for processing. If Apex flex queue is enabled in your org, the batch job is added at the end of the flex queue. For more information, see [Holding Batch Jobs in the Apex Flex Queue](#).
- All scheduled Apex limits apply for batch jobs scheduled using `System.scheduleBatch`. After the batch job is queued (with a status of `Holding` or `Queued`), all batch job limits apply and the job no longer counts toward scheduled Apex limits.
- After calling this method and before the batch job starts, you can use the returned scheduled job ID to abort the scheduled job using the [System.abortJob](#) method.

For an example, see [Using the System.scheduleBatch Method](#).

setPassword(userId, password)

Sets the password for the specified user.

Signature

```
public static Void setPassword(ID userId, String password)
```

Parameters

userId

Type: [ID](#)

password

Type: [String](#)

Return Value

Type: Void

Usage

- If a security question hasn't been previously configured, a user who logs in with a new password that was set using `setPassword()` is redirected to the "Change Your Password" page.
- Use `resetPassword(userId, sendUserEmail)` if you want the user to go through the reset process and create their own password.



Warning

Be careful with this method, and don't expose this functionality to end users.

submit(workItemIds, comments, nextApprover)

Submits the processed approvals. The current user is the submitter and the entry criteria is evaluated for all processes applicable to the current user.

Signature

```
public static List<ID> submit(List<ID> workItemIds, String comments, String nextApprover)
```

Parameters



Type: [String](#)

nextApprover

Type: [String](#)

Return Value

Type: [List<ID>](#)

Usage

For enhanced submit and evaluation features, see the [ProcessSubmitRequest](#) class.

today()

Returns the current date in the current user's time zone.

Signature

```
public static Date today()
```

Return Value

Type: [Date](#)

DID THIS ARTICLE SOLVE YOUR ISSUE?

Let us know so we can improve!

[Share your feedback](#)



DEVELOPER CENTERS

- [Heroku](#)
- [MuleSoft](#)
- [Tableau](#)
- [Commerce Cloud](#)
- [Lightning Design System](#)
- [Einstein](#)
- [Quip](#)

POPULAR RESOURCES

- [Documentation](#)
- [Component Library](#)
- [APIs](#)
- [Trailhead](#)
- [Sample Apps](#)
- [Podcasts](#)
- [AppExchange](#)

COMMUNITY

- [Trailblazer Community](#)
- [Events and Calendar](#)
- [Partner Community](#)
- [Blog](#)
- [Salesforce Admins](#)
- [Salesforce Architects](#)

© Copyright 2025 Salesforce, Inc. [All rights reserved.](#) Various trademarks held by their respective owners. Salesforce, Inc. Salesforce Tower, 415 Mission Street, 3rd Floor, San Francisco, CA 94105, United States

[Privacy Information](#) [Terms of Service](#) [Legal](#) [Use of Cookies](#) [Trust](#) [Cookie Preferences](#)

[✔✕ Your Privacy Choices](#) [Responsible Disclosure](#) [Contact](#)