SessionManagement Class

Contains methods for verifying users' identity, creating custom login flows, customizing security levels, and defining trusted IP ranges for a current session.

Namespace

Auth

- SessionManagement Methods

SessionManagement Methods

The following are methods for `SessionManagement`. All methods are static. Use these methods to customize your user identity verification flows, manage the use of time-based one-time password (TOTP) apps like Google Authenticator, or create custom login flows. Other methods validate a user's incoming IP address against trusted IP range settings for an organization or profile.

- **finishLoginDiscovery(method, userId)**
  Finishes the My Domain Login Discovery login process.
- **finishLoginFlow()**
  Finish the Visualforce Page login flow process, and redirect the user to the default home page.
- **finishLoginFlow(startUrl)**
  Finish the Visualforce Page login flow process, and redirect the user to the specified start URL.
- **generateVerificationUrl(policy, description, destinationUrl)**
  Initiates a user identity verification flow with the verification method that the user registered with, and returns a URL to the identity verification screen. For example, if you have a custom Visualforce page that displays sensitive account details, you can prompt the user to verify identity before viewing it.
- **getCurrentSession()**
  Returns a map of attributes for the current session.
- **getLightningLoginEligibility(userId)**
  Returns the eligibility status for a user who's logging in with Lightning Login when you set up your org with My Domain and use the Login Discovery page type. Use this method to redirect the user to a custom login flow. For example, use after a login attempt to redirect the user to password flow if the user is ineligible for Lightning Login.
- **getQrCode()**
  Returns a map containing a URL to a quick response (QR) code and a time-based one-time password (TOTP) shared secret to configure authenticator apps or devices for multi-factor authentication (MFA).
- **getRequiredSessionLevelForProfile(profileId)**
  Indicates the required login security session level for the given profile.
- **ignoreForConcurrentSessionLimit(sessions)**
  This method is reserved for internal Salesforce use.
- **inOrgNetworkRange(ipAddress)**
  Indicates whether the given IP address is within the organization's trusted IP range according to the organization's Network Access settings.

- **validateTotpTokenForKey(sharedKey, totpCode)**
  Deprecated. Use `validateTotpTokenForKey(totpSharedKey, totpCode, description)` instead.
- **validateTotpTokenForKey(totpSharedKey, totpCode, description)**
  Indicates whether a time-based one-time password (TOTP) code (token) is valid for the given shared key.
- **validateTotpTokenForUser(totpCode)**
  Deprecated. Use `validateTotpTokenForUser(totpCode, description)` instead.
- **validateTotpTokenForUser(totpCode, description)**
  Indicates whether a time-based one-time password (TOTP) code (token) is valid for the current user.
- **verifyDeviceFlow(userCode, startUrl)**
  Verifies the user code entered during the device authentication flow and redirects users to the OAuth approval page. If users aren't logged in, they must log in. After successful login, users are prompted to allow the device to access Salesforce data.

finishLoginDiscovery(method, userId)

Finishes the My Domain Login Discovery login process.

Signature

```
public static System.PageReference finishLoginDiscovery(Auth.LoginDiscoveryMethod method, Id userId)
```

Parameters

method

Type: Auth.LoginDiscoveryMethod LoginDiscoveryMethod Enum

Verification method used with My Domain Login Discovery.

userId

Type: Id

ID used to log in the user. The user must be active.

Return Value

Type: System.PageReference

Usage

Include this method when implementing the `MyDomainLoginDiscoveryHandler` interface to direct users to an authentication mechanism, and then log them in. If users enter a username in the login page, they are sent to the password page for authentication. If users are enrolled in Lightning Login, they are directed to the Salesforce Authenticator to authenticate. If users are SSO-enabled, they are sent to the suitable identity provider (IdP) to authenticate.

The calling user requires Manage Users permission. If the user passed in is frozen or inactive, the method throws an exception.

After implementing the `MyDomainLoginDiscoveryHandler` interface, register the Login Discovery handler from the My Domain Setup page. Under Authentication Configuration, select this handler from the list of Apex classes.

finishLoginFlow()

Finish the Visualforce Page login flow process, and redirect the user to the default home page.

Signature

Usage

Include this method in the Apex controller of the Visualforce Page login flow when creating login flows programmatically. This method indicates that the login flow is finished and redirects the user to the Experience Cloud site's default home page. The login process runs in a restricted session until users complete the process. Calling this method indicates that the login flow is complete, lifts the restriction, and gives users full access to the Experience Cloud site.

finishLoginFlow(startUrl)

Finish the Visualforce Page login flow process, and redirect the user to the specified start URL.

Signature

```
public static System.PageReference finishLoginFlow(String startUrl)
```

Parameters

startUrl

Type: String

Path to the page that users see when they access the Experience Cloud site.

Return Value

Type: System.PageReference

Usage

Include this method in the Apex controller of the Visualforce Page login flow when creating login flows programmatically. This method indicates that the login flow is finished and redirects the user to the specified location in the Experience Cloud site. The login process runs in a restricted session until users complete the process. Calling this method indicates that the login flow is complete, lifts the restriction, and gives users full access to the Experience Cloud site.

generateVerificationUrl(policy, description, destinationUrl)

Initiates a user identity verification flow with the verification method that the user registered with, and returns a URL to the identity verification screen. For example, if you have a custom Visualforce page that displays sensitive account details, you can prompt the user to verify identity before viewing it.

Signature

```
public static String generateVerificationUrl(Auth.VerificationPolicy policy, String
description, String destinationUrl)
```

Parameters

policy

Type: Auth.VerificationPolicy

The session security policy required to initiate identity verification for the user's session. For example, if the policy is set to High Assurance level of session security, and the user's current session has the standard level of session security, the user's session is raised to high assurance after successful verification of identity. In the Setup user interface, this value is shown in the Triggered By column of Identity Verification History.

description

Type: String

destinationUrl

Type: String

The relative or absolute Salesforce URL that you want to redirect the user to after identity verification–for example, `/apex/mypage`. The user is redirected to *destinationUrl* when the identity verification flow is complete, regardless of success. For example, if a user chooses to not respond to the identity challenge and cancels it, the user is still redirected to *destinationUrl*. As a best practice, ensure that your code for this page manually checks that the security policy was satisfied (and the user didn't just manually type the URL in the browser). For example, if the *policy* is High Assurance, the target page checks that the user's session is high assurance before allowing access.

Return Value

Type: String

The URL where the user is redirected to verify identity.

Usage

- If the user is already registered to confirm identity using a time-based one-time password (TOTP), then the user is redirected to the one-time password identity verification flow and asked to provide a code.
- If the user isn't registered with any verification method (such as one-time password or Salesforce Authenticator version 2 or later), the user is prompted to download and verify identity using Salesforce Authenticator. The user can also choose a different verification method.

getCurrentSession()

Returns a map of attributes for the current session.

Signature

```
public static Map<String, String> getCurrentSession()
```

Return Value

Type: Map<String, String>

Usage

The map includes a `ParentId` value, which is the 18-character ID for the parent session, if one exists (for example, if the current session is for a canvas app). If the current session doesn't have a parent, this value is null. The map also includes the `LogoutUrl` assigned to the current session.

If you create an Apex test method that calls this method, the test fails with an error such as, "Unexpected Exception: Current session unavailable." An error occurs because there isn't a session in the context through which the test is being run.

When a session is reused, Salesforce updates the `LoginHistoryId` with the value from the most recent login.

Example

The following example shows the name-value pairs in a map returned by `getCurrentSession()`. Note that `UsersId` includes an "s" in the name to match the name of the corresponding field in the AuthSession object.

```
{
SessionId=0Ak################,
```

```
Username=user@domain.com,
CreatedDate=Wed Jul 30 19:09:29 GMT 2014,
SessionType=Visualforce,
LastModifiedDate=Wed Jul 30 19:09:16 GMT 2014,
LogoutUrl=https://google.com,
SessionSecurityLevel=STANDARD,
UsersId=005###############,
SourceIp=1.1.1.1
}
```

getLightningLoginEligibility(userId)

Returns the eligibility status for a user who's logging in with Lightning Login when you set up your org with My Domain and use the Login Discovery page type. Use this method to redirect the user to a custom login flow. For example, use after a login attempt to redirect the user to password flow if the user is ineligible for Lightning Login.

Signature

```
public static Auth.LightningLoginEligibility getLightningLoginEligibility(Id userId)
```

Parameters

userId

Type: Id

ID of the user who is logging in.

Return Value

Type: Auth.LightningLoginEligibility

Returns the current eligibility status.

Example

```
Auth.LightningLoginEligibility eligibility =
    Auth.SessionManagement.getLightningLoginEligibility(id);
if (eligibility == Auth.LightningLoginEligibility.ELIGIBLE) {
    // success
}
```

getQrCode()

Returns a map containing a URL to a quick response (QR) code and a time-based one-time password (TOTP) shared secret to configure authenticator apps or devices for multi-factor authentication (MFA).

Signature

```
public static Map<String, String> getQrCode()
```

Return Value

Type: Map<String, String>

Usage

The QR code encodes the returned secret as well as the current user's username. The keys are `qrCodeUrl` and `secret`. Calling this method does not change any state for the user, nor does it read any state from the user. This method returns a brand new secret every time it is called, does not

Example

The following is an example of how to request the QR code.

```
public String getGetQRCode() {
        return getQRCode();
    }
    public String getQRCode() {
       Map<String, String> codeResult = Auth.SessionManagement.getQrCode();
       String result = 'URL: '+codeResult.get('qrCodeUrl') + ' SECRET:  ' + codeResult.get
       return result;
    }
```

The following is an example of a returned map.

```
{qrCodeUrl=https://www.salesforce.com/secur/qrCode?w=200&h=200&t=tf&u=user%0000000000.com&
      secret=AAAAA7B5AAAAAA5BBBBBBBBB66AAA}
```

getRequiredSessionLevelForProfile(profileId)

Indicates the required login security session level for the given profile.

Signature

```
public static Auth.SessionLevel getRequiredSessionLevelForProfile(String profileId)
```

Parameters

profileId

Type: String

The 15-character profile ID.

Return Value

Type: Auth.SessionLevel

The session security level required at login for the profile with the ID *profileId*. You can customize the assignment of each level in Session Settings. For example, you can set the High Assurance level to apply only to users who authenticated with multi-factor authentication (MFA) or through a specific identity provider.

ignoreForConcurrentSessionLimit(sessions)

This method is reserved for internal Salesforce use.

Signature

```
public static Map<String,String> ignoreForConcurrentSessionLimit(Object sessions)
```

Parameters

sessions

Type: Object

Return Value

Type: Map<String, String>

Signature

```
public static Boolean inOrgNetworkRange(String ipAddress)
```

Parameters

ipAddress

Type: String

The IP address to validate.

Return Value

Type: Boolean

Usage

If a trusted IP range is not defined, this returns `false`, and throws an exception if the IP address is not valid.

Trusted IP Range Exists?	User is in the Trusted IP Range?	Return Value
Yes	Yes	`true`
Yes	No	`false`
No	N/A	`false`

isIpAllowedForProfile(profileId, ipAddress)

Indicates whether the given IP address is within the trusted IP range for the given profile.

Signature

```
public static Boolean isIpAllowedForProfile(String profileId, String ipAddress)
```

Parameters

profileId

Type: String

The 15-character alphanumeric string for the current user's profile ID.

ipAddress

Type: String

The IP address to validate.

Return Value

Type: Boolean

Usage

If a trusted IP range is not defined, this returns `true`, and throws an exception if the IP address is not valid or if the profile ID is not valid.

Yes	No	`false`
No	N/A	`true`

setSessionLevel(level)

Sets the user's current session security level.

Signature

```
public static Void setSessionLevel(Auth.SessionLevel level)
```

Parameters

level

Type: Auth.SessionLevel

The session security level to assign to the user. The meaning of each level can be customized in the Session Settings for each organization, such as setting the High Assurance level to apply only to users who authenticated with multi-factor authentication (MFA) or through a specific identity provider.

Return Value

Type: Void

Usage

This setting affects the session level of all sessions associated with the current session, such as Visualforce or UI access.

If you create an Apex test method that calls this method, the test fails with an error such as, “Unexpected Exception: Current session unavailable." An error occurs because there isn’t a session in the context through which the test is being run.

Example

The following is an example class for setting the session level.

```
public class RaiseSessionLevel{
    public void setLevelHigh() {
        Auth.SessionManagement.setSessionLevel(Auth.SessionLevel.HIGH_ASSURANCE);
    }
    public void setLevelStandard() {
        Auth.SessionManagement.setSessionLevel(Auth.SessionLevel.STANDARD);
    }
}
```

validateTotpTokenForKey(sharedKey, totpCode)

Deprecated. Use `validateTotpTokenForKey(totpSharedKey, totpCode, description)` instead.

Signature

```
public static Boolean validateTotpTokenForKey(String sharedKey, String totpCode)
```

Parameters

sharedKey

Type: String

The time-based one-time password (TOTP) code to validate.

Return Value

Type: Boolean

Usage

If the key is invalid or doesn't exist, this method throws an invalid parameter value exception or a no data found exception, respectively. If the current user exceeds the maximum of 10 token validation attempts, this method throws a security exception.

validateTotpTokenForKey(totpSharedKey, totpCode, description)

Indicates whether a time-based one-time password (TOTP) code (token) is valid for the given shared key.

Signature

```
public static Boolean validateTotpTokenForKey(String totpSharedKey, String totpCode, String description)
```

Parameters

totpSharedKey

Type: String

The shared (secret) key. The *totpSharedKey* must be a base32-encoded string of a 20-byte shared key.

totpCode

Type: String

The time-based one-time password (TOTP) code to validate.

description

Type: String

The custom description that describes the activity requiring identity verification; for example, "Complete purchase and check out". In the Setup user interface, this text is shown in the Activity Message column of Identity Verification History. The *description* must be 128 characters or fewer. If you provide a value that's longer, it's truncated to 128 characters.

Return Value

Type: Boolean

Usage

If the key is invalid or doesn't exist, this method throws an invalid parameter value exception or a no data found exception, respectively. If the current user exceeds the maximum of 10 token validation attempts, this method throws a security exception.

validateTotpTokenForUser(totpCode)

Deprecated. Use `validateTotpTokenForUser(totpCode, description)` instead.

Signature

```
public static Boolean validateTotpTokenForUser(String totpCode)
```

Parameters

Return Value

Type: Boolean

Usage

If the current user does not have a TOTP code, this method throws an exception. If the current user has attempted too many validations, this method throws an exception.

validateTotpTokenForUser(totpCode, description)

Indicates whether a time-based one-time password (TOTP) code (token) is valid for the current user.

Signature

```
public static Boolean validateTotpTokenForUser(String totpCode, String description)
```

Parameters

totpCode

Type: String

The time-based one-time password (TOTP) code to validate.

description

Type: String

The custom description that describes the activity requiring identity verification; for example, “Complete purchase and check out”. This text appears to users when they verify their identity in Salesforce and, if they use Salesforce Authenticator version 2 or later, in the Salesforce Authenticator mobile app. In addition, in the Setup user interface, this text is shown in the Activity Message column of Identity Verification History. The *description* must be 128 characters or fewer. If you provide a value that’s longer, it’s truncated to 128 characters.

Return Value

Type: Boolean

Usage

If the current user does not have a TOTP code, or if the current user has attempted too many validations, this method throws an exception.

verifyDeviceFlow(userCode, startUrl)

Verifies the user code entered during the device authentication flow and redirects users to the OAuth approval page. If users aren’t logged in, they must log in. After successful login, users are prompted to allow the device to access Salesforce data.

Signature

```
public static System.PageReference verifyDeviceFlow(String userCode, String startUrl)
```

Parameters

userCode

Type: String

Human-readable user code provided to the user by Salesforce. The user must enter this code at the verification URL to approve device access to Salesforce data.

Home page.

Return Value

Type:System.PageReference

Usage

Include this method in the Apex controller when creating a custom Visualforce User Code Verification page for the OAuth 2.0 device authentication flow. This method verifies the user code, prompts the user to log in as needed, and prompts the user to allow the device access to Salesforce data. Upon successful verification and authentication, the user is redirected to the page defined by the start URL.

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