ConfirmUserRegistrationHandler Interface

Manages single sign-on (SSO) user mappings between Salesforce and a third-party identit provider. Use this interface to confirm user mappings before updating them.

Namespace

Auth

Usage

When you set up SSO with a third-party identity provider, you create a class that implemer registration handler using the `Auth.RegistrationHandler` interface. This class manages the creating and updating users. For advanced use cases that require you to confirm user infor during the update process, implement the `Auth.ConfirmUserRegistrationHandler` interface class. This interface must be implemented in addition to `Auth.RegistrationHandler`.

You can use the `Auth.ConfirmUserRegistrationHandler` interface to ensure that users are ma correctly between Salesforce and the third party. When a user who has previously logged i authentication provider logs in again, you can confirm that the incoming user data is cons with the user's third-party identifier. If not, you can identify which user is supposed to be l

You can also use the `Auth.ConfirmUserRegistrationHandler` interface to switch context for u multiple records. For example, a user has two records—an admin user and a standard user. the user logs in, the third-party identity provider confirms the account used to log in and s response to Salesforce via the UserInfo endpoint. You can then use this information to de whether to log in the user as an admin or standard user.

- ConfirmUserRegistrationHandler Methods
- ConfirmUserRegistrationHandler Example Implementation

ConfirmUserRegistrationHandler Methods

The following are methods for `ConfirmUserRegistrationHandler`.

- **confirmUser(userId, tpalId, portalId, userdata)**
  Returns the ID of the user to be logged in based on their mapping to a third-party identifier. This method is called before calling the `updateUser()` method. It's called c the incoming user has previously logged in and has a third-party account link to a Salesforce user.

confirmUser(userId, tpalId, portalId, userdata)

Returns the ID of the user to be logged in based on their mapping to a third-party identifie method is called before calling the `updateUser()` method. It's called only if the incoming u previously logged in and has a third-party account link to a Salesforce user.

Signature

```
public Id confirmUser(Id userId, Id tpalId, Id portalId, Auth.UserData userdata)
```

The ID of the user who is mapped to the third-party identifier via a third-party account li

tpalId

Type: Id

The third-party account link corresponding to the third-party identifier.

portalId

Type: Id

The portal ID the user is logging in to. If there's no portal configured, this value can be nu

userData

Type: Auth.UserData

Contains user information from the third-party identity provider.

Return Value

Type: Id

The Id of the user to be logged in. If null, login fails.

ConfirmUserRegistrationHandler Example Implementation

This example implements the `Auth.ConfirmUserRegistrationHandler` interface during the u update process to confirm that the correct user is logging in based on their email address name.

```
global class StandardUserRegistrationHandler implements Auth.RegistrationHandler
    global User createUser(Id portalId, Auth.UserData data){
        User u = new User();
        Profile p = [SELECT Id FROM profile WHERE name='Standard User'];
        u.username = data.username + '@salesforce.com';
        u.email = data.email;
        u.lastName = data.lastName;
        u.firstName = data.firstName;
        String alias = data.username;
        if(alias.length() > 8) {
            alias = alias.substring(0, 8);
        }
        u.alias = alias;
        u.languagelocalekey = data.attributeMap.get('language');
        u.localesidkey = data.locale;
        u.emailEncodingKey = 'UTF-8';
        u.timeZoneSidKey = 'America/Los_Angeles';
        u.profileId = p.Id;
        return u;
    }

    global void updateUser(Id userId, Id portalId, Auth.UserData data) {
        User u = new User(id=userId);
        u.username = data.username + '@salesforce.com';
        u.email = data.email;
        u.lastName = data.lastName;
        u.firstName = data.firstName;
        String alias = data.username;
        if(alias.length() > 8) {
            alias = alias.substring(0, 8);
        }
        u.alias = alias;
        u.languagelocalekey = data.attributeMap.get('language');
        u.localesidkey = data.locale;
```

```
            return userId;
        } else { // find the right user
            User confirmedUser = [SELECT id FROM user WHERE email=:data.email];
            return confirmedUser.Id;
        }
    }
}
```

The following example tests the implementation:

```
@isTest
public class StandardUserRegistrationHandlerTest {
    static testMethod void testConfirmUser() {
        StandardUserRegistrationHandler handler = new StandardUserRegistrationHa
        Auth.UserData sampleData = new Auth.UserData('idA', 'firstName', 'A',
            'firstName A', 'userA@example.org', null, 'usernameA', 'en_US', 'fac
            null, new Map<String, String>{'language' => 'en_US'});
        User u = handler.createUser(null, sampleData);
        insert(u);
        String uid = u.id;

        sampleData = new Auth.UserData('idB', 'firstName', 'B',
            'firstName B', 'userA@example.org', null, 'usernameB', 'en_US', 'fac
            null, new Map<String, String>{}); // note that user B is using userA
        Id confirmedUserId = handler.confirmUser(uid, '060xx0000004Eh6', null, s
        System.assertEquals(uid, confirmedUserId); // we should see userA's id
    }
}
```

DID THIS ARTICLE SOLVE YOUR ISSUE?

Let us know so we can improve!

Share your fe

salesforce

DEVELOPER CENTERS

- Heroku
- MuleSoft
- Tableau
- Commerce Cloud
- Lightning Design System
- Einstein
- Quip

POPULAR RESOURCES

- Documentation
- Component Library
- APIs
- Trailhead
- Sample Apps
- Podcasts
- AppExchange

COMMUN

- Trailblazer (
- Events and
- Partner Cor
- Blog
- Salesforce /
- Salesforce /

Your Privacy Choices Responsible Disclosure Contact