SamlJitHandler Interface

Apex Reference Guide / Auth Namespace / SamlJitHandler Interface

SamlJitHandler Interface

Use this interface to control and customize Just-in-Time user provisioning logic during SAML single sign-on.

Namespace

Auth

Usage

To use custom logic for user provisioning during SAML single sign-on, you must create a class that implements `Auth.SamlJitHandler`. This allows you to incorporate organization-specific logic (such as populating custom fields) when users log in to Salesforce with single sign-on. Keep in mind that your class must perform the logic of creating and updating user data as appropriate, including any associated account and contact records.

In Salesforce, you specify your class that implements this interface in the `SAML JIT Handler` field in SAML Single Sign-On Settings. Make sure that the user you specify to run the class has “Manage Users” permission.

- SamlJitHandler Methods
- SamlJitHandler Example Implementation

SamlJitHandler Methods

The following are methods for `SamlJitHandler`.

- **createUser(samlSsoProviderId, communityId, portalId, federationId, attributes, assertion)**
  Returns a User object using the specified Federation ID. The User object corresponds to the user information. This object can be a new user that hasn’t been inserted in the database or an existing user record in the database.
- **updateUser(userId, samlSsoProviderId, communityId, portalId, federationId, attributes, assertion)**
  Updates the specified user’s information. This method is called if the user has logged in before with SAML single sign-on and then logs in again, or if your application is using the `Existing User Linking URL`.

createUser(samlSsoProviderId, communityId, portalId, federationId, attributes, assertion)

Returns a User object using the specified Federation ID. The User object corresponds to the user information. This object can be a new user that hasn’t been inserted in the database or an existing user record in the database.

Signature

```
public User createUser(Id samlSsoProviderId, Id communityId, Id portalId, String federationId,
Map<String,String> attributes, String assertion)
```

Parameters

communityId

Type: Id

The ID of the Experience Cloud site. This parameter can be `null` if you're not creating an Experience Cloud user.

portalId

Type: Id

The ID of the portal. This parameter can be `null` if you're not creating a portal user.

federationId

Type: String

The ID Salesforce expects to be used for this user.

attributes

Type: Map<String,String>

All attributes in the SAML assertion that were added to the default assertion; for example, custom attributes. Attributes are case-sensitive.

If the assertion is encrypted, the attribute map contains a decrypted assertion stored as a value with the key `Sfdc.SamlAssertion`.

assertion

Type: String

The default SAML assertion, base-64 encoded.

If the assertion is encrypted, this parameter is also encrypted. To access the decrypted assertion, see the `Sfdc.SamlAssertion` key in the attribute map.

Return Value

Type: User

A User sObject.

Usage

The *communityId* and *portalId* parameter values can be `null` or the associated keys can be empty if there's no Experience Cloud site or portal configured with this organization.

updateUser(userId, samlSsoProviderId, communityId, portalId, federationId, attributes, assertion)

Updates the specified user's information. This method is called if the user has logged in before with SAML single sign-on and then logs in again, or if your application is using the `Existing User Linking URL`.

Signature

```
public void updateUser(Id userId, Id samlSsoProviderId, Id communityId, Id portalId, String federationId, Map<String,String> attributes, String assertion)
```

Parameters

userId

Type: Id

The ID of the Salesforce user.

communityId

Type: Id

The ID of the Experience Cloud site. This type can be `null` if you're not updating an Experience Cloud user.

portalId

Type: Id

The ID of the portal. This type can be `null` if you're not updating a portal user.

federationId

Type: String

The ID Salesforce expects to be used for this user.

attributes

Type: Map<String,String>

All attributes in the SAML assertion that were added to the default assertion; for example, custom attributes. Attributes are case-sensitive.

If the assertion is encrypted, the attribute map also contains a decrypted assertion stored as a value with the key `Sfdc.SamlAssertion`.

assertion

Type: String

The default SAML assertion, base-64 encoded.

If the assertion is encrypted, this parameter is also encrypted. To access the decrypted assertion, see the `Sfdc.SamlAssertion` key in the attribute map.

Return Value

Type: void

SamlJitHandler Example Implementation

This is an example implementation of the `Auth.SamlJitHandler` interface. This code uses private methods to handle accounts and contacts (handleContact() and handleAccount()), which aren't included in this example.

```
global class StandardUserHandler implements Auth.SamlJitHandler {
    private class JitException extends Exception{}
    private void handleUser(boolean create, User u, Map<String, String> attributes,
        String federationIdentifier, boolean isStandard) {
        if(create && attributes.containsKey('User.Username')) {
            u.Username = attributes.get('User.Username');
        }
        if(create) {
            if(attributes.containsKey('User.FederationIdentifier')) {
                u.FederationIdentifier = attributes.get('User.FederationIdentifier');
            } else {
                u.FederationIdentifier = federationIdentifier;
            }
        }
        if(attributes.containsKey('User.ProfileId')) {
            String profileId = attributes.get('User.ProfileId');
            Profile p = [SELECT Id FROM Profile WHERE Id=:profileId];
            u.ProfileId = p.Id;
        }
```

```
                u.Phone = attributes.get('User.Phone');
            }
            if(attributes.containsKey('User.Email')) {
                u.Email = attributes.get('User.Email');
            }

        //More attributes here - removed for length

            //Handle custom fields here

            if(!create) {
                update(u);
            }
        }

        private void handleJit(boolean create, User u, Id samlSsoProviderId, Id communityId, I
            String federationIdentifier, Map<String, String> attributes, String assertion) {
            if(communityId != null || portalId != null) {
                String account = handleAccount(create, u, attributes);
                handleContact(create, account, u, attributes);
                handleUser(create, u, attributes, federationIdentifier, false);
            } else {
                handleUser(create, u, attributes, federationIdentifier, true);
            }
        }

        global User createUser(Id samlSsoProviderId, Id communityId, Id portalId,
            String federationIdentifier, Map<String, String> attributes, String assertion) {
            User u = new User();
            handleJit(true, u, samlSsoProviderId, communityId, portalId,
                federationIdentifier, attributes, assertion);
            return u;
        }

        global void updateUser(Id userId, Id samlSsoProviderId, Id communityId, Id portalId,
            String federationIdentifier, Map<String, String> attributes, String assertion) {
            User u = [SELECT Id FROM User WHERE Id=:userId];
            handleJit(false, u, samlSsoProviderId, communityId, portalId,
                federationIdentifier, attributes, assertion);
        }
    }
```

DID THIS ARTICLE SOLVE YOUR ISSUE?

Let us know so we can improve!

Share your feedback