Use custom expressions as Name ID for SAML apps

Once you enable and configure SAML SSO for your apps in Zoho One, Zoho One will authenticate your users into those apps. During authentication, Zoho One will pass a value called *Name ID* to the apps. The Name ID matches user identities in Zoho One with user identities in the app.

Example:
Consider a user, Amelia, has an email address *amelia@zylker.biz* and a bank account number *123******47*. Let's say a banking app, *BankApp*, requires users to enter their account number as the username when signing in. In other words, *BankApp* uses *123******47* to identify Amelia instead of using *amelia@zylker.biz*.

Normally, when SAML SSO is configured between Zoho One and an app, Zoho One will pass the email address of the users as Name ID. However, *BankApp* is expecting an account number, not an email address; if *amelia@zylker.biz* is passed to *BankApp* as Name ID, *BankApp* will show an error saying that it wasn't able to find any user whose account number is *amelia@zylker.biz*. So the admin who sets up SSO will have to configure Zoho One to pass the account number to *BankApp* as Name ID.

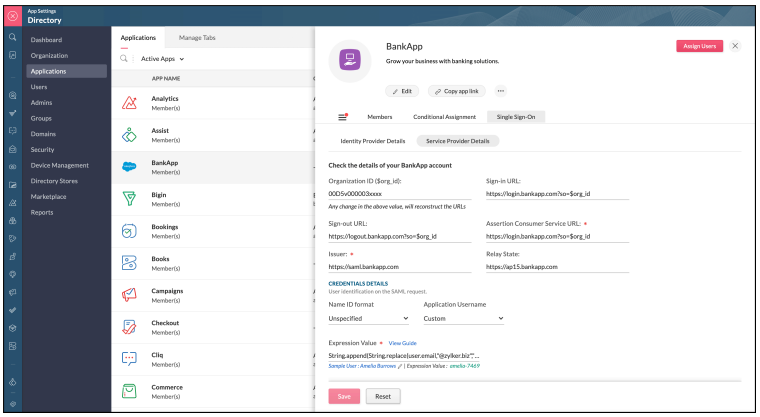
As the Name ID needs to be unique, most apps use the user's primary email address as Name ID. But you can also define other user information (such as first and last names, or any custom fields that you have created for them) as the Name ID. This can be useful in cases where the app doesn't support email address as username, or in apps you've developed and implemented to use non-conventional authentication methods.

To change the Name ID for an app:

1. Sign in to Zoho One, then click **Directory** in the left menu.
2. Go to **Applications**, then click on the app you want to change the Name ID for.
3. Go to **Single Sign-On**, then click **Service Provider Details**.
4. Under *Credentials Details*, you can set:
    - **Application Username:** The field that has to be passed to the app as username.
    - **Name ID format:** The format in which the username should be passed.
5. Click **Save**.

For advanced requirements, you can set the Application Username as a custom expression constructed as a combination of multiple fields. These expressions can be constructed using string manipulation methods on various fields in Zoho One.

The custom expression must be written in the following format:
String_method(<string>,<additional_values>)



For example, an organization may have a custom-built application that uses a combination of the user's email username and employee ID as the username. So, a Zoho One user with the email address *amelia@zylker.biz* and an employee ID of *7469* will use *amelia-7469* as the username for the custom-built application. In this case, the admin would set the *Name ID format* as *Unspecified*, the *Application Username* as *Custom*, and the *Expression Value* as:
String.append(String.replace(user.email,"@zylker.biz","-"),user.Employee ID)

Here's how the expression works:

1. **String.replace** will be executed first. It replaces the email domain of the user (*@zylker.biz*) with a hyphen, converting *amelia@zylker.biz* to *amelia-*.
2. **String.append** will be executed next, and it attaches the employee ID (*7469*) to the end of the output we got in the previous step (*amelia-*). So the final *Application Username* passed to the app would be *amelia-7469*.

The following table lists the fields that you can use to construct these expressions, and their corresponding formats:

Field name	Format
First name	user.firstName
Last name	user.lastName
Primary email address	user.email
Full name	user.displayName
Any custom fields' information	user.<custom field> Example: For a field named *Vehicle Number* , the format would be **user.Vehicle Number**.

The string methods that you can use to construct expressions are:

String methods	Expression format	Description	Example
Append	String.append(<string>, <string_to_be_appended>)	Adds <string_to_be_appended> to the end of the <string>.	**String.append(user.firstName,user.EmployeeID)** The value in the user's "Employee ID" field will be appended with the user's first name. If a user's first name is "Amelia" and employee ID is "7469," then the value will be "Amelia7469".
Index Of	String.indexOf(<string>, <character>)	Returns the position of the first instance of the given character in the <string>.	**String. indexOf(user.firstName,"o")** Returns the position of the first instance of the character "o" within the user's first name. If a user's first name is "Johnson", then the position of the first occurrence of the character "o" will be returned, which is 2.
Replace	String.replace(<string>, <string_to_be_removed>, <string_to_be_placed>)	Replaces all occurrences of <string_to_be_removed> in the <string> with <string_to_be_placed>.	**String.replace(user.firstName,"e","a")** All the occurrences of "e" will be replaced by "a". If a user's first name is "Ellen", then all occurrences of the character "e" in the name will be replaced by "a", giving " **A**ll**a**n" as the output.
Replace First	String.replaceFirst(<string>, <string_to_be_removed>, <string_to_be_placed>)	Replaces the first occurrence of <string_to_be_removed> in the <string> with <string_to_be_placed>.	**String.replaceFirst(user.firstName,"e","a")** The first occurrence of "e" will be replaced by "a". If a user's first name is "Ellen", then the first occurrence of the character "e" in the name will be replaced by "a", giving "**A**llen" as the output.
Substring	String.substring(<string>, <beginIndex>,<endIndex>)	Fetches the part of the <string> that is specified by the indexes.	**String.substring(user.firstName,0,1)** The first and second characters from the user's first name will be fetched. If a user's first name is "John", then the string "Jo" will be returned.
To Lower Case	String.toLowerCase(<string>)	Converts all characters in the given string to lower case.	**String.toLowerCase(user.firstName)** The characters in the user's first name will be converted to lower case. If a user's first name is "John", "john" will be returned.
To Upper Case	String.toUpperCase(<string>)	Converts all characters in the given string to upper case.	**String.toUpperCase(user.firstName)** The characters in the user's first name will be converted to upper case. If a user's first name is "John", then "JOHN" will be returned.
Trim	String.trim(<string>)	Removes leading and trailing spaces in the given string. Can be used to sanitize fields that might have typos, or strings that were derived using other methods like substring.	**String.trim(user.displayName)** The blank spaces, if any, will be removed. For example, if the full name of a user is " Johnson Doe", then the space before "Johnson" will be removed, and "Johnson Doe" will be returned.