SailPoint IdentityIQ Web Services Connector

The following topics are discussed in this document:

Supported Features

SailPoint IdentityIQ Web Services Connector supports the following features:

- Account Management
  - Aggregation, Refresh Accounts, Pass Through Authentication (Basic Authentication)

    Web Service Connector provides support for using Web Service application as a Pass Through Authentication application. For more information on configuration for pass-through authentication, see "Configuration for Pass Through Authentication".

    Note: Currently Pass Through Authentication is supported with identity attribute only.
  - Create, Update, Delete
  - Enable, Disable, Change Password
  - Add/Remove Entitlements
- Account - Group Management
  - Aggregation, Refresh Groups

Additional Supported Features

- SailPoint Web Services Connector provides additional support for pagination.

  For more information on embedding pagination support in Web Service Connector, see **"Pagination" on page 19**.
- SailPoint Web Services Connector now provides support for saving of updated **Refresh Token** received along with access token, if any.

  If **Refresh Token** has expired, it must be manually generated and updated in the application configuration as mentioned in **"(General Settings) Basic Configuration Parameters" on page 3**.
- SailPoint Web Services Connector provides additional support for client certificate authentication. For more information, see "Enable Client Certificate Authentication" parameter in **"(General Settings) Basic Configuration Parameters" on page 3**.
- **Support for Cookies**: The Web Services Connector now provides support of cookies for multiple endpoints configuration. The application will manage the cookies internally only for the multiple endpoints configured for the same operation. Cookies from the previous endpoints can be used in all the subsequent endpoints of same operation type.

Support for multiple group

The following table lists the example for different operations for the added new Group Object Types:

Object Types	Operation Type	Description
Group	• Group Aggregation • Get Object - Group • Add Entitlement • Remove Entitlement	Aggregates all Group objects.
Role	• Group Aggregation - Role • Get Object - Role • Add Entitlement - Role • Remove Entitlement - Role	Aggregates all Group Role objects.
PermissionSet	• Group Aggregation - PermissionSet • Get Object - PermissionSet • Add Entitlement - PermissionSet • Remove Entitlement - PermissionSet	Aggregates all Group PermissionSet objects.

Supported Managed Systems

SailPoint Web Services Connector supports web services with JSON/XML response.

Prerequisites

Web Services must be accessible.

Administrator Permissions

The user/administrator must have the required permissions to call the web services API of the managed system.

Configuration Parameters

This section provides the following type of configuration parameters of SailPoint IdentityIQ Web Services Connector:

- Basic configuration parameters
- Operation specific configuration parameters

(General Settings) Basic Configuration Parameters

The following table lists the basic configuration parameters of SailPoint IdentityIQ Web Services Connector:

Parameters	Description
Add Object Type	This button pops up a window to add the name of the object type. For example, **Group Aggregation - Role**
Base URL*	The base URL to connect to the web service managed system.
Authentication Method*	Authentication method that is supported by the managed system • OAuth2 • API Token • Basic Authentication • No Authentication **Note: SOAP Web Services supports only Basic Authentication method.**
Schema Attribute for Account Enable status	Attribute name and value required to be provided to check the Enable status. For example, `status=Active`
Request Timeout (In Seconds)	Request Timeout Value in seconds.
Enable Client Certificate Authentication	Configure client certificate authentication.
*Applicable if **Authentication Method** is selected as **OAuth2***	
Grant Type*	Select the type of Grant: • Refresh Token • JWT • Client Credentials • Password
Client Id*	(*Optional for JWT*) Client Id for OAuth2 authentication.

Parameters	Description
Client Secret*	(*Optional for JWT*) Client secret for OAuth2 authentication.
Token URL*	URL for generating access token. **Note: Token URL supports placeholders for replacement of application attribute dynamically.** **For example, if token URL is required to be prepared with some sensitive information that is, client_secret, then** `$application.client_secret$` **can be used so that the corresponding value would be determined from the application replaced at the appropriate location in the token URL.**
Username*	(*Applicable if Grant Type is selected as Password*) Username of the resource owner.
Password*	(*Applicable if Grant Type is selected as Password*) Password of the resource owner.
Refresh Token*	(*Applicable if Grant Type is selected as Refresh Token*) A valid refresh token for grant type authentication.
Private Key*	(*Applicable if Grant Type is selected as JWT*) The private key to be used to sign the JWT.
Private Key Password*	(*Applicable if Grant Type is selected as JWT*) Password for the provided private key.
*Applicable if **Authentication Method** is selected as **API Token***	
API Token*	Enter the API token specific to the Managed System.
*Applicable if **Authentication Method** is selected as **Basic***	
Username*	Username that holds permission to execute the Web Service.
Password*	Password of the user name.
*Applicable if **Enable Client Certificate Authentication** is selected*	
Client Certificate*	Client certificate for authentication.
Certificate Key*	Client certificate’s private key.
Note: Web Services Connector supports only PEM format for the ‘Client Certificate’ and certificate’s private key.	

Note: Attributes marked with * sign are the mandatory attributes.

Additional Configuration Parameters

Add the following attributes in the application debug page:

Attributes	Description
throwBeforeAfterRuleException	During aggregation if an exception is displayed from **WebServiceBeforeOperationRule** or **WebServiceAfterOperationRule**, then aggregation continues and completes successfully. Set the value of the following flag to true in the application debug page to terminate the aggregation by displaying an error message: `throwBeforeAfterRuleException` **Note: This flag can be set only for Account and Group aggregation (multiple group aggregation if any).** The default value of the `throwBeforeAfterRuleException` flag is set to false.

throwProvBeforeRuleException	During Provisioning, GetObject and Test Connection, if an exception is thrown from **WebServiceBeforeOperationRule**, then Provisioning would fail. Hence to dispose of the exception in the log file and proceed with provisioning, set the value of **throwProvBeforeRuleException** to false in the application debug page as follows:
	`<entry key="throwProvBeforeRuleException">`
	`  <value>`
	`     <Boolean>true</Boolean>`
	`  </value>`
	`</entry>`
	Note: The default value of 'throwProvBeforeRuleException' flag is set to true for new Web Services application and false for existing application (before upgrading to IdentityIQ version 8.1). The 'throwProvBeforeRuleException' flag can be set for all operations except Account and Group aggregation.

throwProvAfterRuleException	During Provisioning, GetObject and Test Connection if an exception is thrown from **WebServiceAfterOperationRule**, then Provisioning would fail. Hence to dispose of the exception in the log file and proceed with provisioning, set the value of **throwProvAfterRuleException** to false in the application debug page as follows:
	`<entry key="throwProvAfterRuleException">`
	`  <value>`
	`     <Boolean>true</Boolean>`
	`  </value>`
	`</entry>`
	Note: The default value of 'throwProvAfterRuleException' flag is set to true for new Web Services application and false for existing application (before upgrading to IdentityIQ version). The 'throwProvAfterRuleException' flag can be set for all operations except Account and Group aggregation.

Attributes	Description

isQuotesEnabled	(*Applicable only for JSON Web Services*) To send the data in a type as mentioned in provisioning plan or schema type, set the value of **isQuotesEnabled** to true in the application debug page as follows:
	`<entry key="isQuotesEnabled">`
	`  <value>`
	`    <Boolean>true</Boolean>`
	`  </value>`
	`</entry>`
	For more information, see “Use of Quotes” on page 18.

createAccountWithEntReq	To enable the functionality of sending entitlements with create account in a single request to the managed system, set the value of **createAccountWithEntReq** parameter to true as follows:
	`<entry key="createAccountWithEntReq">`
	`  <value>`
	`    <Boolean>true</Boolean>`
	`  </value>`
	`</entry>`
	Default value: false

enableHasMore	If **enableHasMore** is set to true as follows then the termination of aggregation would depend on the value of **hasMore** attribute:
	`<entry key="enableHasMore" value="true"/>`
	The **hasMore** attribute is the boolean attribute which is to be set in the **transientValues** map in the before/after operation rule. Unless the value of **hasMore** attribute is false aggregation would not be terminated.
	If **enableHasMore** is set to false as follows, then the aggregation would be terminated if the number of accounts returned is zero:
	`<entry key="enableHasMore" value="false"/>`

<table>
<tr><th>Attributes</th><th>Description</th></tr>
<tr><td>possibleHttpErrors</td><td>

When an API endpoint does not send expected error codes to flag failure conditions, the connector can be configured as follows (example) with all possible HTTP error codes/messages, which the API endpoint would return resulting into failure of connector operations:

```
<entry key="possibleHttpErrors">
 <value>
  <Map>
   <entry key="errorCodes">
     <value>
       <List>
         <Integer>500</Integer>
         <Integer>501</Integer>
       </List>
     </value>
     </entry>
     <entry key="errorMessages">
       <value>
          <List>
            <String>INVALID_SESSION</String>
            <String>Access Denied</String>
          </List>
        </value>
      </entry>
    </Map>
  </value>
</entry>
```

Note: In few instances the Web Services Connector returns httpstatuscode as 200 but the response payload may contain error. In this case, ideally the connector must fail the request or OAuth token generation must try to regenerate the token.

These possibleHttpError codes/messages can also be configured to specify invalid/expiry token error. In this case connector would regenerate and save the token for OAUTH2 authentication and retries the operation with the newly generated access token.

</td></tr>
</table>

Attributes	Description

isGetObjectRequiredForPTA	For using the Web Service application as a Pass-through Authentication Connector, set the value of **isGetObjectRequiredForPTA** to true in the application debug page as follows:
	`<entry key="isGetObjectRequiredForPTA">`
	`  <value>`
	`    <Boolean>true</Boolean>`
	`  </value>`
	`</entry>`
	For new Web Services application created, default value for **isGetObjectRequiredForPTA** would be set to true.
	Note:
	• When set to true, it would execute Get Object operation to verify if the entered userName (Considered as Identity attribute) is present on the managed system or not.
	• When set to false then it would skip Get Object operation and Pass-through Authentication operation must have response mapping with account object schema attributes.
	For more information, see “Configuration for Pass Through Authentication”.

objectNotFoundErrorMsg	Based on the error message list, Connector would decide to display the **objectNotFoundErrorMsg** error.
	For example, user can create the following entry for **objectNotFoundErrorMsg** entry key with custom error message to identify exceptions (these can be multiple):
	`<entry key="objectNotFoundErrorMsg">`
	` <value>`
	`   <List>`
	`     <String>404: Not Found</String>`
	`     <String>404</String>`
	`   </List>`
	` </value>`
	`</entry>`

authenticationFailedErrorMsg	Based on the error message list, Connector would decide to display the **AuthenticationFailedErrorMsg** error.
	For example, user can create the following entry for **AuthenticationFailedErrorMsg** entry key with custom error message to identify exceptions (these can be multiple):
	`<entry key="authenticationFailedErrorMsg">`
	` <value>`
	`   <List>`
	`     <String>Authentication Failed</String>`
	`    </List>`
	` </value>`
	`</entry>`

Attributes	Description

expiredPasswordErrorMsg	Based on the error message list, Connector would decide to throw **ExpiredPasswordErrorMsg** error.
	For example, user can create the following entry for **ExpiredPasswordErrorMsg** entry key with custom error message to identify exceptions (these can be multiple):
	`<entry key="expiredPasswordErrorMsg">`
	` <value>`
	`   <List>`
	`     <String>Password Expired</String>`
	`    </List>`
	` </value>`
	`</entry>`
	If response contains string matched with **expiredPasswordErrorMsg**, then it would redirect user from login page to Change Password page.

Applicable if* Authentication Method *is selected as OAuth2	

oauth_headers	• To have customized headers as a part of the access token generation request, add the oauth_headers parameter to the application debug page as follows:
	`<entry key="oauth_headers">`
	`  <value>`
	`    <Map>`
	`     <entry key="Content-Type"`
	`value="application/x-www-form-urlencoded" />`
	`    </Map>`
	`  </value>`
	`</entry>`
	Note: Web Services Connector now uses access token configured in the application as authorization header for each endpoint, users would no longer require to specify the authorization header for each endpoint. If authorization is provided at endpoint level then it would precede over the access token.
	SailPoint recommends to provide authorization header suffix in the access token provided. For example, `Bearer <Access Token>`**. If no prefix is provided, then connector would by default provide** `Bearer` **as Authorization header prefix.**
	• To send additional headers for token generation, add the oauth_headers parameter to the application debug page as follows:
	`<entry key="oauth_headers">`
	`  <value>`
	`   <Map>`
	`    <entry key="customHeaderKey"`
	`value="customHeaderValue"/>`
	`   </Map>`
	`  </value>`
	`</entry>`

Attributes	Description
oauth_headers_to_exclude	Web Services Connector supports exclusion of headers in the OAuth2 request. The header keys for headers which are intended to be excluded from the OAuth2 request, can be added as comma separated values in the application using debug page as follows: `<entry key="oauth_headers_to_exclude" value="Authorization,CUSTOM_HEADER"/>`

oauth_request_parameters	To send additional parameters for token generation, add the following entry in the application debug page:
	`<entry key="oauth_request_parameters">`
	`  <value>`
	`    <Map>`
	`      <entry key="customParamKey" value="customParamValue"/>`
	`    </Map>`
	`  </value>`
	`</entry>`

oauth_body_attrs_to_exclude	To delete parameters for token generation, add the following entry in the application debug page:
	`<entry key="oauth_body_attrs_to_exclude">`
	`  <value>`
	`    <Map>`
	`      <entry key="oauth_body_attrs_to_exclude" value="customParamKey1,customParamKey2"/>`
	`    </Map>`
	`  </value>`
	`</entry>`

oAuthJwtHeader	Contains the alg (algorithm that is used for signing the JWT assertion) as follows:
	`<entry key="oAuthJwtHeader">`
	`  <value>`
	`    <Map>`
	`      <entry key="alg" value="RS256"/>`
	`    </Map>`
	`  </value>`
	`</entry>`
	If required additional header attributes can be provided in this map.

Attributes	Description

oAuthJwtPayload	Contains the aud (Audience), Expiry of the JWT assertion (exp), iss (Issuer), sub (Subject) as follows:
	`<entry key="oAuthJwtPayload">`
	`  <value>`
	`    <Map>`
	`      <entry key="aud" value=""/>`
	`      <entry key="exp" value="15f"/>`
	`      <entry key="iss" value=""/>`
	`      <entry key="sub" value=""/>`
	`    </Map>`
	`  </value>`
	`</entry>`
	Note: If required additional payload attributes can be provided in this map. For additional attributes like `jti`**,** `iat`**,** `nbf` **if only** `key (not value)` **is available in the map then it would consider the default values for the same.**

(Connector Operations) Operation Specific Configurations

Note: No default provisioning template is provided. The template may vary from one managed system to another.

Perform the following procedure to add and configure the specific operations:

1. Click **Add Operation**.
2. Select the operation from the drop down list of **Choose Operation**.
3. Provide a unique name to the operation. For example: **Account Aggregation-1**, **Get Object-Role, Group Aggregation-Role**.
4. Select the configure option (Pencil image) on the same row and configure the newly added operation. For more information on the operation specific configuration parameters, see “Operation Specific Configuration Parameters” below. Allows user to provide additional options.

Operation Specific Configuration Parameters

The following table lists the operation specific configuration parameters of SailPoint Web Services Connector:

Parameters	Description
ContextURL	Context URL specific to the operation. For example, `/api/core/v3/securityGroups?startIndex=0&count=100&fields=%40all&sort=lastNameAsc`

Parameters	Description
Method	Select one of the following type of HTTP method supported by the respective operation: • GET • PUT • POST • DELETE • PATCH
Header	(*Optional*) To view the header value in plain text, user must provide it in encrypted form. The encrypted value can be obtained from IdentityIQ Console. **For example**: The following example displays the sample header key and header value, where **Authorization** is header key and **1:vQaPY5LvJVbpsaig0nE56Q==** is the header value: `Authorization 1:vQaPY5LvJVbpsaig0nE56Q==` **Note:** • Content-Type header value must contain type matching any XML formats that is, application/XML or text/XML or */XML. • JSON request, JSON response: Content-Type= application/JSON (*optional*), Accept (*optional*) • XML request, XML response: Content-Type= application/XML or text/XML or */XML (*required*), Accept (*optional*) • JSON request, XML response: Content-Type=application/JSON (*optional*), Accept= application/XML or text/XML or */XML (*required*)

<table>
<tr><th>Parameters</th><th>Description</th></tr>
<tr><td>Body</td><td>

Standard http body used to post data with request. User can send data in either of the following format:

- **form-data**: (*Applicable only for JSON*) Key value. User must set the data that has to pass in the key value
- **raw**: Data to be sent in request body.

  For endpoint configuration, user must provide the XML payload by selecting the **raw** format.

For example,

- (For JSON)

```
{
  "limit": 10,
  "cursor": "abcd1234"
}
```

- (For XML)

```
<soapenv:Envelope
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:bsvc="urn:com.workday/bsvc">
   <soapenv:Header>
     <Security
xmlns="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd">
         ...
     </Security>
   </soapenv:Header>
      <soapenv:Body>
      ...
      </soapenv:Body>
</soapenv:Envelope>
```

</td></tr>
</table>

Parameters	Description
Response	(***For XML Web Services***) **XPath Namespace Mapping**: XML Namespace Prefix and corresponding Namespace URL identify uniquely named elements and attributes in XML request/response. If there exists any non-standard XML Namespace in the response, configure the same in the XML Namespace mapping where the key is Namespace Prefix and value is the Namespace URL. **Note: Absence of non-standard Xml Namespaces would result in errors while response parsing.** If a default Namespace is present, add a temporary Namespace Prefix with the default Namespace URL in the XML Namespace mappings. Further, use this temporary prefix in the XPATH elements within the scope of the default Namespace. For example, see "Example 1:" example of payload.
	Root Path: Common path present in the JSON/XML response. It must be common for all the above attribute mentioned in the **Response Attribute Mapping** For example, (For JSON) `$.members.profile` (For XML) `//wd:Response_Data/wd:Worker/wd:Worker_Data`
	Successful Response Code: Successful response code expected by the respective Web Service operation. This field accepts HTTP status code in csv format (For example, 200, 201, 203). If the list does not contain any value, the status code from 200 to 299 would be checked. There could be situation where successful status code may start with 2, in this situation user can provide 2**.
Before Rule	**Before Operation Rule**: Rule that will be invoked before performing any operation (account aggregation, enable, disable account and so on).
After Rule	**After Operation Rule**: Rule that will be invoked after performing any operation (account aggregation, enable, disable account and so on)

Note: For more information on operation specific configurations, see "Additional Information" on page 17.

Keywords

Web Service application supports the following keywords for various configuration attributes such as context URL, Headers, Body (JSON and Form-data) for a single or multiple endpoints:

<table>
<tr><th>Keywords</th><th>Description</th></tr>
<tr><td>plan</td><td>Used for configuring the provisioning operations such as, create account, update account among others.

For example,

- (Context URL): `/api/core/v3/people/$plan.nativeIdentity$`
- (JSON Body example for plan):

<pre>
{
"new_members": [
   {
  "email": "$plan.email$",
  "first_name": "$plan.first_name$",
  "surname": "$plan.surname$",
  "send_welcome_email": $plan.send_welcome_email$,
  "role": {
   ".tag": "member_only"
  }
}
]
}
</pre></td></tr>
<tr><td>response</td><td>Used for multiple endpoints, where the response from the first endpoint is provided as an input for the second endpoint.

For example, there are two endpoints for account aggregation.

- The first endpoint returns a response as a list of **member_ids** that is an input for the second endpoint as mentioned in the next point.
- Second endpoint's JSON body is: `{"members_info":[{"member_id":"$response.member_id$"}]}`</td></tr>
<tr><td>application</td><td>Used to get other configuration attributes from the current application.

For example, `"$application.accesstoken$"`, where the accesstoken is an application configuration attribute.</td></tr>
</table>

Keywords	Description

getobject	Used while performing **Aggregate Account** (get a single account details).
	For example,
	• (JSON body):
	`{`
	`"members": [`
	`    {`
	`  ".tag": "member_id",`
	`  "member_id": "$getobject.nativeIdentity$"`
	`}`
	`]`
	`}`
	• (Context URL for get object): `/api/v4/admin/$getobject.nativeIdentity$`

nativeIdentity	Signifies the **AccountID** (identity attribute) in the plan or during **getobject** operation. For example, nativeIdentity would be used along with the keyword as follows: • **getobject**: `$getobject.nativeIdentity$` • **plan**: `$plan.nativeIdentity$`
planNativeIdentity	Signifies the **Plans** nativeIdentity in the Provisioning plan. For example, planNativeIdentity would be used along with the keyword as follows: • **plan**: `$plan.planNativeIdentity$` Web Services Connector can be configured with native identity from the provisioning plan using placeholder `$plan.planNativeIdentity$`.
authenticate	To provide username and password in endpoint configuration user can use the following placeholders: • **$authenticate.username$** • **$authenticate.password$** For more information on configuration for pass-through authentication, see “Configuration for Pass Through Authentication” on page 31.

Note: SailPoint recommends to use placeholder in the body and url rather then adding sensitive information directly. For example, https://TESTMACHINE:9096/users/user/$application.accesstoken$

Note: In the above table for examples of attributes that are mapped to a raw JSON response, it may contain formatted values as follows (similar to ".tag": "member_id"):
['.tag']
['@etag']
['@@test']
['complex.name']
['role name']
['role_name']

Schema Attributes

Discover schema functionality is not available. Hence user must add the schema attributes manually for the respective Web Service based managed system.

Create New Group

Perform the following to create new group:

1. Click on **Add Object Type** button and provide the name for the group object. For example, Role
   This will add new schema for the newly added group object type.
2. Add the schema attributes with appropriate type in newly added group schema.
3. Provide the Native Object type, Identity attribute and display attribute for the Group Object.
4. Add new attribute in account/group schema and select the type as newly added group object.

   Note: The value of this attribute must be same as the Identity attribute value of the newly added group object.

Additional Information

This section describes the additional information related to the Web Services Connector.

Upgrade Considerations

- After upgrading IdentityIQ to version 8.1 from version 7.2 Patch 3 or earlier version, add the following entry key in the application debug page of the existing application:

  ```
  <entry key="encrypted"
  value="accesstoken,refresh_token,oauth_token_info,client_secret,private_key,priv
  ate_key_password,clientCertificate,clientKeySpec"/>
  ```
- After upgrading IdentityIQ to version 8.1:
  - to support pass-through authentication, add **isGetObjectRequiredForPTA** attribute to the application debug page.

    For more information on the above additional configuration attribute, see “Additional Configuration Parameters”.
  - if the **Authentication Method** is selected as **OAuth2** and the **Grant Type** as **JWT** then add the following parameters in the application debug page:
    - **oAuthJwtHeader**
    - **oAuthJwtPayload**

    For more information on the above additional configuration attributes, see “Additional Configuration Parameters” section.
  - add the following attribute in the featureString in the application debug page:
    - **AUTHENTICATE**

    For more information on the above attribute, see “Keywords” section.

Web Services Before/After Operation Rule

Web Services uses the following operation rules:

- WebServiceBeforeOperationRule
- WebServiceAfterOperationRule

For more information on Web Services Before/After Operation Rule, see **Web Services Before/After Operation Rule**.

Use of Quotes

(*Applicable only for JSON*) For application prior to version 8.1, extra quotes with plan placeholder must be send the attribute value as a String. For example, `{firstName: "$plan.firstName$"}`

After upgrading to IdentityIQ version 8.1,

- to send data type in a type mentioned in provisioning plan or schema type add the **isQuotesEnabled** in the application debug page and set it to true.

  For example,

  ```
  {
     "user":{
        "firstName":"$plan.firstName$",
        "isActiveAsAString":"\"$plan.isActive$\"",
        "isActive":"$plan.isActive$",
        "userId":"$plan.Id$"
     }
  }
  ```

  where

  - *firstName* is a String
  - *isActive* is a Boolean
  - *Id* is an Integer

  The above mentioned attributes type can be part of Provisioning plan/Schema attributes/Provisioning policy form and the results would be as follows:

  ```
  {
     "user":{
        "firstName":"Testfirstname",
        "isActiveAsAString":"true",
        "isActive":true,
        "userId":987654321
     }
  }
  ```

- For formdata to work as mentioned in the following example, set the value of **isQuotesEnabled** to true:

  For example,

  ```
    - firstName $plan.firstName$
    - isActive $plan.isActive$
    - isActiveAsAString "$plan.isActive$"
    - userId $plan.Id$
    - passwordresetCount 0
  ```

  The above mentioned attributes type can be a part of Provisioning plan/Schema attributes/Provisioning policy form and the results would be as follows:

  ```
  {
     "firstName":"Testfirstname",
     "isActiveAsAString":"true",
     "isActive":true,
     "userId":987654321,
     "passwordresetCount":"0"
  }
  ```

Pagination

Web Services Connector supports generic paging for Account and Group Aggregations. Following are the methods for embedding paging in Web Service Connector:

- Using BEFORE and AFTER operation rules

  Or
- Using Paging tab

BEFORE and AFTER operation rules

To embed pagination in Web Service Connector, manual processing is required in BEFORE and AFTER operation rules of Web Service Connector.

1. The Web Service Connector relies on a temporary information stored in the application object in form of a map which has the name as **transientValues**.
2. The administrator must write the Before Rule and AFTER Rule for account/group aggregation as follows:

- **Web Service Before Rule**: The Before Rule alters the URL/request parameters if the value of the **hasMore** parameter is set to **TRUE** and the request to fetch further accounts is triggered.If **hasMore** parameter is not set or is set to **FALSE** the pagination request would be terminated.

  For example, see sample Before Rule for account aggregation request in Web Services Connector for Dropbox using V2 in **examplerules.xml** file by name **Example WSBeforeRl DropboxPaging** as follows:

  ```
  import sailpoint.tools.Util;

  Map obj = (Map) application.getAttributeValue("transientValues");
  System.out.println("BEFORE RULE: Transient Values ==> " + obj);
  if(null != obj) {
    String offset = obj.get("offset");
    System.out.println("BEFORE RULE: offset value ==> " + offset);
    String urlString = (String) requestEndPoint.getFullUrl();
  ```

```
  if(Util.isNotNullOrEmpty(offset)) {
    System.out.println("BEFORE RULE: requestEndpoint ==> " + requestEndPoint);
    System.out.println("BEFORE RULE: URL ==> " + urlString);
URL tempUrl = new URL(urlString);
String queryString = tempUrl.getQuery();
    System.out.println("BEFORE RULE: Query String ==> " + queryString);

if(Util.isNotNullOrEmpty(queryString)) {
  StringBuffer queryParams = new StringBuffer();
  String[] params = tempUrl.getQuery().split("&");
      for (String param : params) {
    if(queryParams.length() > 0)
  queryParams.append("&");
if(param.startsWith("sysparm_offset=")) {
      queryParams.append("sysparm_offset=");
      queryParams.append(offset);
    } else {
  queryParams.append(param);
}
      }
     urlString = urlString.replace(tempUrl.getQuery(), queryParams.toString());
}
  }

  System.out.println("BEFORE RULE: Updated Query String ==> " + urlString);
  requestEndPoint.setFullUrl(urlString);
}
System.out.println("BEFORE RULE: requestEndpoint Updated ==> " + requestEndPoint);
return requestEndPoint;
```

In case of Dropbox V2, the **cursor** returned from the previous team membership listing API would be stored in the **transientValues** map in the application by the Web Service AFTER Rule. The url is modified to direct to the paging API and the cursor would be sent as a part of the form data. Ensure that the **hasMore** flag is set by the earlier requests AFTER RULE

- **Web Service After Rule**: The AFTER Rule deduces whether the managed system has more records which can be fetched and added as an entry in **transientValues** with **hasMore** key and value as TRUE/FALSE depending upon the condition deduced.

  For example, see sample AFTER Rule for account aggregation request in Web Services Connector for Dropbox using V2 in **examplerules.xml** by name **Example WSAfterRl DropboxPaging** as follows:

```
Integer fetchedRecordsCount = 0;
if(null != processedResponseObject) {
  fetchedRecordsCount  = ((List) processedResponseObject).size();
}

Integer expectedCount = null;
Integer offset = null;
URL url = new URL(requestEndPoint.getFullUrl());
System.out.println("AFTER RULE: Original Url ==> " + url);
String[] params = url.getQuery().split("&");
for (String param : params) {
  String name = param.split("=")[0];
  String value = param.split("=")[1];

  switch(name) {
    case "sysparm_limit":
        expectedCount = Integer.parseInt(value);
        break;

      case "sysparm_offset":
        offset = Integer.parseInt(value);
        break;

      default:
    }
  }

System.out.println("AFTER RULE: Fetch Count ==> " + fetchedRecordsCount);
System.out.println("AFTER RULE: Limit Count ==> " + expectedCount);
System.out.println("AFTER RULE: Fetch Offset ==> " + offset);

boolean hasMore = (fetchedRecordsCount != 0 && null != expectedCount &&
fetchedRecordsCount.equals(expectedCount) && null != offset);
System.out.println("AFTER RULE: Has More? ==> " + hasMore);

Map transientValues = application.getAttributeValue("transientValues");
if(transientValues == null) {
  transientValues = new HashMap();
  application.setAttribute("transientValues", transientValues);
}
transientValues.put("hasMore", hasMore);
if (hasMore) {
  if(null != offset) {
    System.out.println("AFTER RULE: New Offset ==> " + (offset + expectedCount));
    transientValues.put("offset", String.valueOf(offset + expectedCount));
  }
}
```

In case of Dropbox, Dropbox V2 for team membership response contain the following elements:

- **cursor**: is an encrypted token which represents the next page to be fetched, if any, and would form part of the subsequent API calls.
- **has_more**: is a boolean value which explicitly indicates whether more records are available for fetching.

AFTER Rule stores the **cursor** and **has_more** values from the response in the **transientValues** map in the Application object. This map stores the necessary information which would be used by the BEFORE RULE to manipulate the next API call. Ensure that the flag indicating whether the managed system contains more records is stored by the key named **hasMore**. This field is mandatory as it is the deciding factor for aborting the pagination requests.

Paging tab

Paging can be configured in Account/Group Aggregation endpoints through Paging tab with one of the following methods:

- Paging based on limit-offset
- Paging based on response markers
- Paging based on response header links

Note: If there are multiple Account or Group Aggregation endpoints configured, paging would be supported only for the first endpoint of Account and Group Aggregation each.

Paging mechanism has the following predefined set of keywords:

Keywords	Description	
application	Represents the application.	
	baseUrl	Base URL configured in the application.
endpoint	Represents endpoint configuration.	
	relativeURL	Relative URL of the endpoint.
	fullUrl	Full URL contained within the endpoint.
limit	Page limit.	
offset	Initial page offset.	
request	Represents request body.	
requestHeaders	Represents request header.	
response	Represents response body object.	
responseHeaders	Represents response header.	
TERMINATE_IF	Indicates termination condition, multiple conditions can exist.	
	NO_RECORDS	Indicates no records retrieved; to be used in conjunction with TERMINATE_IF, evaluates to TRUE / FALSE
	RECORDS_COUNT	Indicates number of records retrieved.
NULL	Indicates null or empty object.	
REMOVE	Indicates to remove an attribute.	

The following table lists the supported and conditional supported operations:

<table>
<tr><th>Type</th><th>Operations</th></tr>
<tr><td>Regular operations</td><td>+, -, *, /, =, && and ||</td></tr>
<tr><td>Conditional operations</td><td><, >, <=, >=, == and !=</td></tr>
</table>

Paging based on limit-offset

This section describes paging based on limit-offset for ServiceNow and Workday target system.

For ServiceNow

For example, the initial aggregation url for ServiceNow target system would be as follows:
https://*XYZ*.service-now.com/api/now/v1/table/sys_user?sysparm_limit=100&sysparm_fields=sys_id

The above url includes the following parameters:

- baseUrl: **https://*XYZ*.service-now.com**
- relativeURL: **api/now/v1/table/sys_user?sysparm_limit=100&sysparm_fields=sys_id**

Following are the configuration steps in the paging tab based on limit-offset:

1. Use the following for ServiceNow target system:

   ```
   $sysparm_limit$ = 100
   TERMINATE_IF $RECORDS_COUNT$ < $sysparm_limit$

   $sysparm_offset$ = $sysparm_offset$ + $sysparm_limit$

   $endpoint.fullUrl$ = $application.baseUrl$ +
   "/api/now/v1/table/sys_user?sysparm_fields=sys_id&sysparm_limit=100&sysparm_offs
   et=" + $sysparm_offset$
   ```

   In the above example, maximum record count is been set using `sysparm_limit` and `RECORDS_COUNT`. In this case `RECORDS_COUNT` represents the number of records fetched from the response and `sysparm_limit` can be changed as required.

   Based on the above step the next page url will be as mentioned in the steps below.
2. Second page url based on paging steps configuration: **https://*XYZ*.com/api/now/v1/table/sys_user?sysparm_limit=100&sysparm_fields=sys_id&sysparm_offset=100**
3. Third page url based on paging steps configuration: **https://*XYZ*.com/api/now/v1/table/sys_user?sysparm_limit=100&sysparm_fields=sys_id&sysparm_offset=200**

The paging would be terminated when the `RECORDS_COUNT` is less than the `sysparm_limit` as follows: `$RECORDS_COUNT$ < $sysparm_limit$`

For Workday

For example, the initial aggregation url for Workday target system would be as follows: **https://*XYZ*.workday.com/ccx/service/sailpoint_pt1/Human_Resources/v24.1**

The above url includes the following parameters:

- baseUrl: **https://*XYZ*.workday.com**
- relativeURL: **/ccx/service/sailpoint_pt1/Human_Resources/v24.1**

Following are the configuration steps in the paging tab based on limit-offset:

1. Use the following for Workday target system:

   ```
   TERMINATE_IF $response.wd:Response_Results.wd:Page$ >
   $response.wd:Response_Results.wd:Total_Pages$

   $offset$ = $response.wd:Response_Results.wd:Page$ + 1

   $request.bsvc:Response_Filter.bsvc:Page.text()[1]$ = $offset$
   ```

   In the above example, the number of pages are verified with response from Workday and compared it with the total number of pages.

   If the current page number is less than total pages then the request body (SOAP BODY) is updated with the new page by incrementing it.

Following is an example of initial payload request with Initial Page Size (the initial page would be the beginning index) as 1 and Page Size (number of record per page) as 10:

```
<soapenv:Body>
<bsvc:Get_Workers_Request bsvc:version="v24.1">
<bsvc:Request_Criteria>
          ....
          ....
          </bsvc:Request_Criteria>
         <bsvc:Response_Filter>
  <bsvc:Page>1</bsvc:Page>

  <bsvc:Count>10</bsvc:Count>
</bsvc:Response_Filter>

        <bsvc:Response_Group>
          ....
          ....
        </bsvc:Response_Group>
        </bsvc:Get_Workers_Request>
   </soapenv:Body>
```

Based on the above paging steps the next payload would be as follows:

2. The second aggregation payload: **Incrementing offset** (page number)

```
<soapenv:Body>
<bsvc:Get_Workers_Request bsvc:version="v24.1">
<bsvc:Request_Criteria>
          ....
          ....
          </bsvc:Request_Criteria>
         <bsvc:Response_Filter>
  <bsvc:Page>2</bsvc:Page>

  <bsvc:Count>10</bsvc:Count>
</bsvc:Response_Filter>

        <bsvc:Response_Group>
          ....
          ....
        </bsvc:Response_Group>
        </bsvc:Get_Workers_Request>
   </soapenv:Body>
```

Paging based on response markers

This section describes paging based on a marker value in the response. Based on the marker value in the response aggregation is terminated or continued.

The following table lists the examples of response marker based paging for the respective Managed System:

Managed System	Examples

Dropbox	`TERMINATE_IF $response.has_more$ == FALSE`
	`$endpoint.fullUrl$ = $application.baseUrl$ + $endpoint.relativeUrl$ + "/continue"`
	`$request.cursor$ = $response.cursor$`
	`REMOVE $request.limit$`

Salesforce	`TERMINATE_IF $response.ns:result.ns:done$ != FALSE`
	`$request.soapenv:Body$ = "<urn:queryMore><urn:queryLocator>" + $response.ns:result.ns:queryLocator$ + "</urn:queryLocator></urn:queryMore>"`

Successfactor	`TERMINATE_IF $response.ns:result.ns:hasMore$ != TRUE`
	`$request.soapenv:Body$ = "<urn:queryMore><urn:querySessionId>" + $response.ns:result.ns:querySessionId$ + "</urn:querySessionId></urn:queryMore>"`

Paging based on response header links

This section describes paging based on response header links.

The following table lists the examples of response header links based paging for the respective Managed System:

Managed System	Examples

ServiceNow	`TERMINATE_IF $responseHeaders.Link.next$ == NULL`
	`$endpoint.fullUrl$ = $responseHeaders.Link.next$`

Okta	`TERMINATE_IF $responseHeaders.Link.next$ == NULL`
	`$endpoint.fullUrl$ = $responseHeaders.Link.next$`

Paging Configuration Caveats:

1. Every paging configuration step must start on a new line.
2. SailPoint recommends to provide a <space> after every operator, condition or placeholder for correct evaluation of paging expression.
3. Paging mechanism follows the placeholder notation for resolution of attribute values, that is., `$response.attribute_key$`. Any attribute which follows the placeholder notation would be resolved or assigned a value depending upon the operator being used.

4. Intermediate values can also be stored between page request by using the placeholder notation. In order to achieve this, any attribute key which does not match any of the predefined keywords can be used. For more information, see the example mentioned in the above table for **ServiceNow (Using limit-offset)** where `$sysparm_offset$` is being updated and used between page requests.
5. For complex expressions or conditions, multiple conditions can be clubbed together using '**(**' and '**)**'. For example, `TERMINATE_IF ($someattribute$ == TRUE) && ($otherattribute$ == NULL)`

Saving Parameters in Web Services Connector

Web Services Connector supports storing the values in application object permanently. Saving of the parameters can be configured using the **connectorStateMap** in BEFORE and AFTER operation rules of Web Service Connector. Following are the examples of BEFORE and AFTER operation rules.

BEFORE Operation Rules

```
Map updatedInfoMap = new HashMap();
requestEndPoint.setFullUrl(requestEndPoint.getFullUrl().replaceAll("&&", "&"));
Map connectorStateMap = new HashMap();
connectorStateMap.put("accesstoken","Bearer
accessTokenGeneratedInBeforeRuleScript");
updatedInfoMap.put("updatedEndPoint",requestEndPoint);
updatedInfoMap.put("connectorStateMap",connectorStateMap);
return updatedInfoMap;
```

AFTER Operation Rules

```
 import java.util.*;
  import java.util.HashMap;
  import java.util.List;
  import java.util.Map;
  import java.util.Map.Entry;
  import java.util.Iterator;
  Map updatedMapInfo = new HashMap();
  if (parsedResponseObject != null){
  System.out.println("Parsed response is not null");
     for (Map iterateMap : parsedResponseObject) {
      if (iterateMap != null ) {
            Set keySet = iterateMap.keySet();
            for (String s : keySet) {
                System.out.println(s);
                if (s.equals("given_name")) {
           String forStr = (String) iterateMap.get("given_name");
            forStr = "TEST"+ forStr;
                    System.out.println("forStr: " + forStr );
            iterateMap.put("given_name", forStr);
        }
            }
        }
    }
  updatedMapInfo.put("data", parsedResponseObject);
  }
  Map connectorStateMap = new HashMap();
  connectorStateMap.put("refresh_token","refreshTokenGeneratedInAfterRuleScript");
```

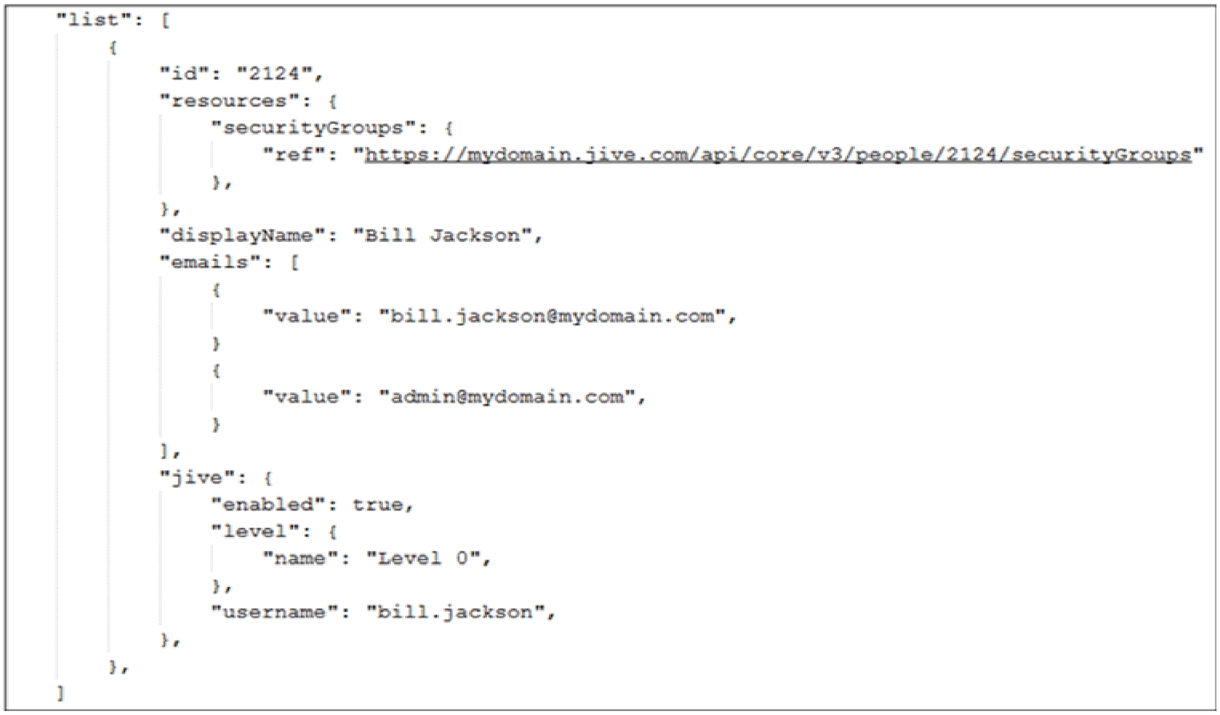
```
updatedInfoMap.put("connectorStateMap",connectorStateMap);
return updatedMapInfo;
```

Configuration for Response

When configuring the Web Services application, map the schema attribute as follows:

- **For JSON**: Refer the following example:

Figure 1—Example for mapping the schema attributes with JSON

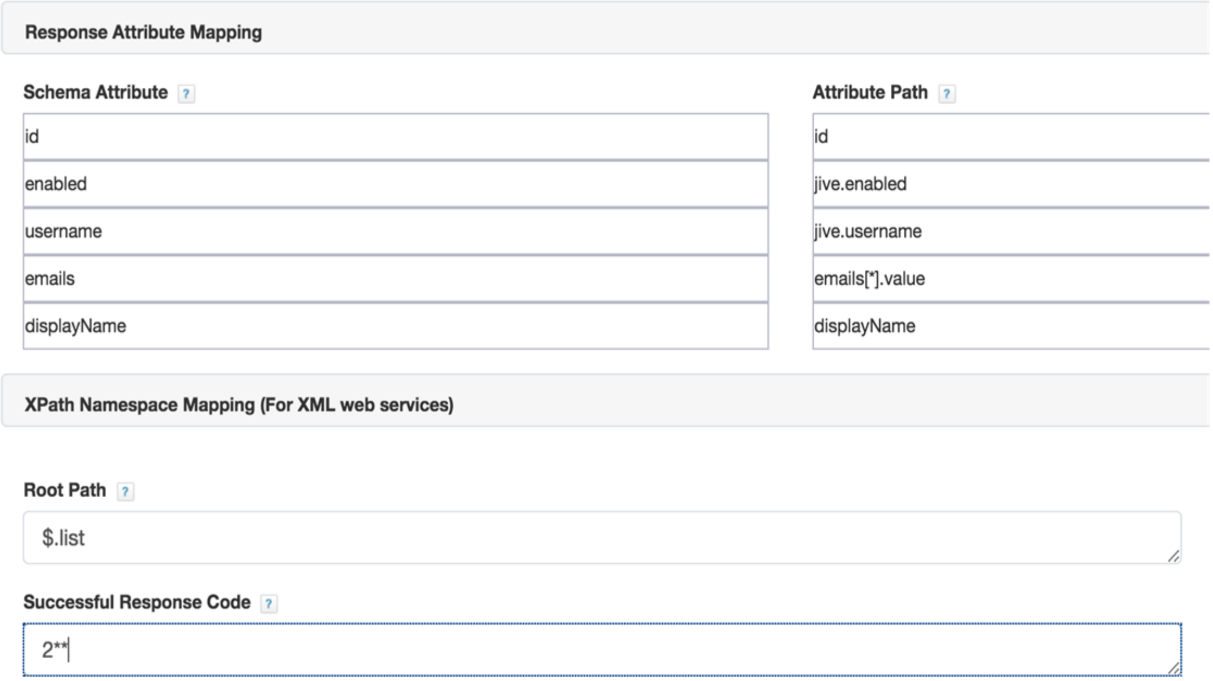


```
"list": [
    {
        "id": "2124",
        "resources": {
            "securityGroups": {
                "ref": "https://mydomain.jive.com/api/core/v3/people/2124/securityGroups"
            },
        },
        "displayName": "Bill Jackson",
        "emails": [
            {
                "value": "bill.jackson@mydomain.com",
            }
            {
                "value": "admin@mydomain.com",
            }
        ],
        "jive": {
            "enabled": true,
            "level": {
                "name": "Level 0",
            },
            "username": "bill.jackson",
        },
    },
]
```

In the above JSON response, all the attributes can be mapped as follows considering Root Path as $.list:

```
Id = id
displayName=displayName
username=jive.username
enabled =jive.enabled
emails=emails[*].value
```

Figure 2—Mapped schema attributes



Response Attribute Mapping

Schema Attribute	Attribute Path
id	id
enabled	jive.enabled
username	jive.username
emails	emails[*].value
displayName	displayName

XPath Namespace Mapping (For XML web services)

Root Path: $.list

Successful Response Code: 2**

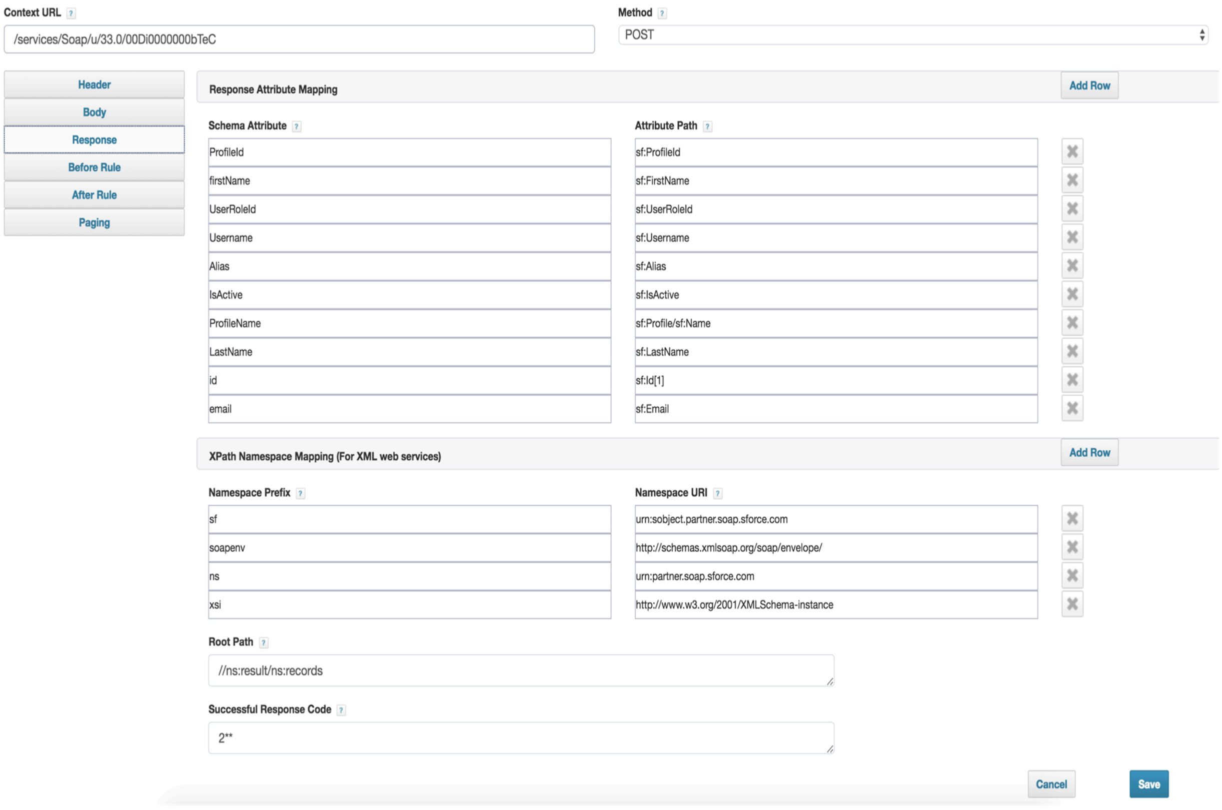
- **For XML**:
  - **Example 1:**

    XML response for mapping:

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns="urn:partner.soap.sforce.com"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:sf="urn:sobject.partner.soap.sforce.com">
 <soapenv:Header>
      .....
      .....
 </soapenv:Header>
 <soapenv:Body>
    <queryResponse>
       <result xsi:type="QueryResult">
         <done>true</done>
         <queryLocator xsi:nil="true"/>
         <records xsi:type="sf:sObject">
           <sf:type>User</sf:type>
           <sf:Id>123456</sf:Id>
           <sf:Id>123456</sf:Id>
           <sf:Alias>Alias</sf:Alias>
           <sf:City xsi:nil="true"/>
           <sf:CommunityNickname>CName1</sf:CommunityNickname>
           <sf:Email>test@test.com</sf:Email>
           <sf:IsActive>false</sf:IsActive>
           <sf:Username>test@test.com</sf:Username>
           <sf:FirstName>Test</sf:FirstName>
           <sf:LastName>Test</sf:lastName>
```

```
            </records>
          <size>1</size>
            </result>
        </queryResponse>
    </soapenv:Body>
</soapenv:Envelope>
```

See the following figure which mentions the XPath Namespace mapping for XML Web Services:



Context URL: /services/Soap/u/33.0/00Di0000000bTeC

Method: POST

Response Attribute Mapping

Schema Attribute	Attribute Path
ProfileId	sf:ProfileId
firstName	sf:FirstName
UserRoleId	sf:UserRoleId
Username	sf:Username
Alias	sf:Alias
IsActive	sf:IsActive
ProfileName	sf:Profile/sf:Name
LastName	sf:LastName
id	sf:Id[1]
email	sf:Email

XPath Namespace Mapping (For XML web services)

Namespace Prefix	Namespace URI
sf	urn:sobject.partner.soap.sforce.com
soapenv	http://schemas.xmlsoap.org/soap/envelope/
ns	urn:partner.soap.sforce.com
xsi	http://www.w3.org/2001/XMLSchema-instance

Root Path: //ns:result/ns:records

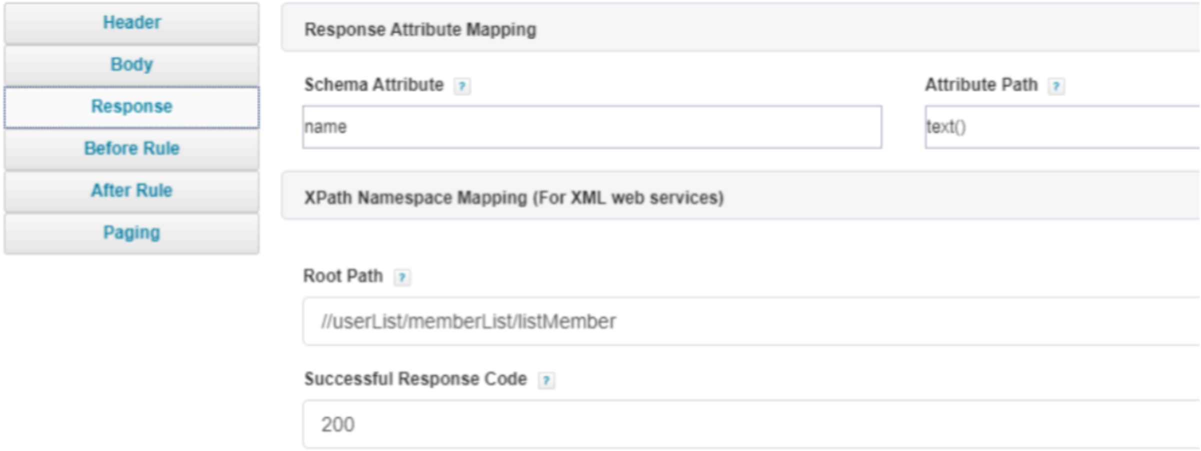
Successful Response Code: 2**

- **Example 2:**

  XML response for mapping:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<userList id="5457373">
<name>DillardsEveryone</name>
  <memberList>
    <listMember>drazzetti@ravemobilesafety.com</listMember>
    <listMember>gage5.test5@test5.dillards.com</listMember>
  </memberList>
</userList>
```

See the following figure which mentions the Root Path configuration for listMember:



Response Attribute Mapping

Schema Attribute	Attribute Path
name	text()

XPath Namespace Mapping (For XML web services)

Root Path: //userList/memberList/listMember

Successful Response Code: 200

Configuration for Multiple Endpoints

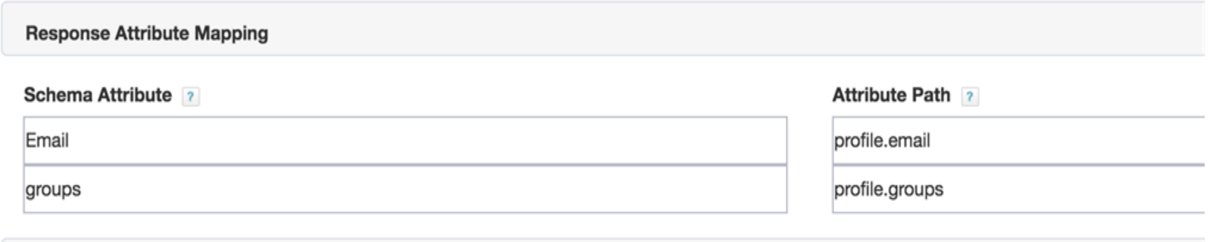
Perform the following to obtain the properties of account/group from multiple endpoints:

1. The basic attribute is obtained from the first endpoint and is then used for fetching the data from rest of the endpoints.
   For example, during aggregation of Jive some attributes are obtained from first endpoint ("Figure 2—Mapped schema attributes") using the following URL:

   https://myDomain.jive.com/api/core/v3/people

2. To fetch additional attribute from another endpoint use the `id` attribute from the previous response. Add these attributes in Schema Attribute of Response Attribute Mapping and response as follows:

   - **Schema Attribute**



Response Attribute Mapping

Schema Attribute	Attribute Path
Email	profile.email
groups	profile.groups

   - **Response**: The following context URL contains id which fetches all the groups connected to that account:

     https://myDomain.jive.com/api/core/v3/people/$response.id$/securityGroups

Configuring Multiple Entitlement Requests

To enable the functionality of sending multiple entitlement request of different type of entitlements (role, permission, groups and so on) in a single request to the managed system, set the value of **addRemoveEntInSingleReq** parameter to true as follows:

```
<entry key="addRemoveEntInSingleReq">
```

```
    <value>

        <Boolean>true</Boolean>

    </value>

</entry>
```

1. If **addRemoveEntInSingleReq** is set to true, then the payload for entitlements must be as given in the following example:
   ```
   {
   "group_id" : $plan.groups$,
   "permission":$plan.permission$,
   "roles": $plan.roles$
   }
   ```
2. If **addRemoveEntInSingleReq** is set to false, then the payload for entitlements must be as given in the following example:
   ```
   {
   "group_id" : "$plan.groups$",
   "permission":"$plan.permission$",
   "roles": "$plan.roles$"
   }
   ```
3. If **addRemoveEntInSingleReq** is set to true and **isQuotesEnabled** is set to true, then the payload for entitlements must be as given in the following example:
   ```
   {
   "group_id" : "$plan.groups$",
   "permission":"$plan.permission$",
   "roles": "$plan.roles$"
   }
   ```

Configuration for Pass Through Authentication

To Configure Pass Through Authentication on existing Web Services application, perform the following:

1. Add **AUTHENTICATE** in the featureString of the application debug page.
   For more information on authenticate, see “Keywords” on page 14.
2. Add Pass Through Authentication operation in Web Services application.
   This operation would be used to perform verification of user credentials provided from Login page or IdentityIQ Console.
3. Add the **isGetObjectRequiredForPTA** entry key with value as **true** in the application debug page.
   For more information on **isGetObjectRequiredForPTA**, see “Additional Configuration Parameters” on page 5.
4. (*Optional*) If user wants to configure error messages for Pass Through Authentication, it can be done using the following entry keys:

   - objectNotFoundErrorMsg
   - authenticationFailedErrorMsg
   - expiredPasswordErrorMsg

   For more information, see “Additional Configuration Parameters” on page 5.

For example,

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:urn="urn:partner.soap.sforce.com">

   <soapenv:Header>

     <urn:LoginScopeHeader>

         <urn:organizationId></urn:organizationId>

         <!--Optional:-->

         <urn:portalId></urn:portalId>

      </urn:LoginScopeHeader>

   </soapenv:Header>

   <soapenv:Body>

      <urn:login>

         <urn:username>$authenticate.username$</urn:username>

         <urn:password>$authenticate.password$</urn:password>

      </urn:login>

   </soapenv:Body>

</soapenv:Envelope>
```

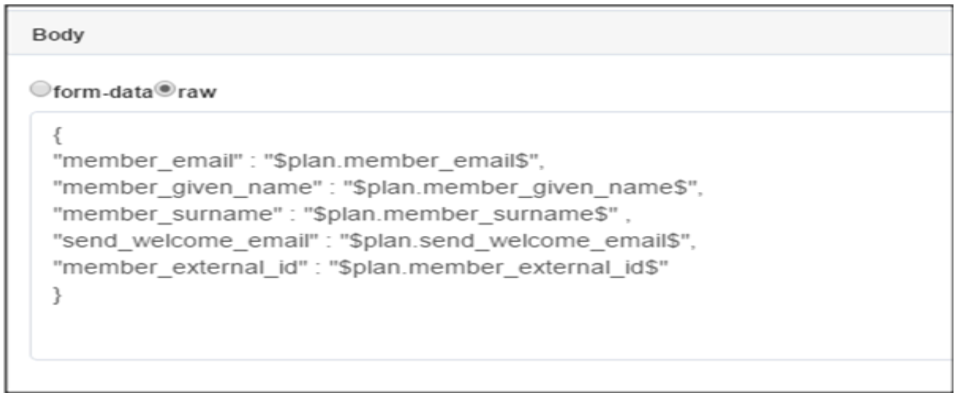
Other Operations

For certain operations, the Body must be updated accordingly.

Create Account

This section provides an example for updating the Body for create account in Dropbox. For fetching attribute through Provisioning Plan, the body must be updated in the following manner. This fetches the attribute detail through Provisioning Form and updates the endpoint.

- (For JSON) In the following Body,
  - **$plan** represents the Provisioning Plan that is passed to provision method
  - **$plan.member_surname**: the connector checks for **member__surname** in the attribute request and updates in the body after it is found

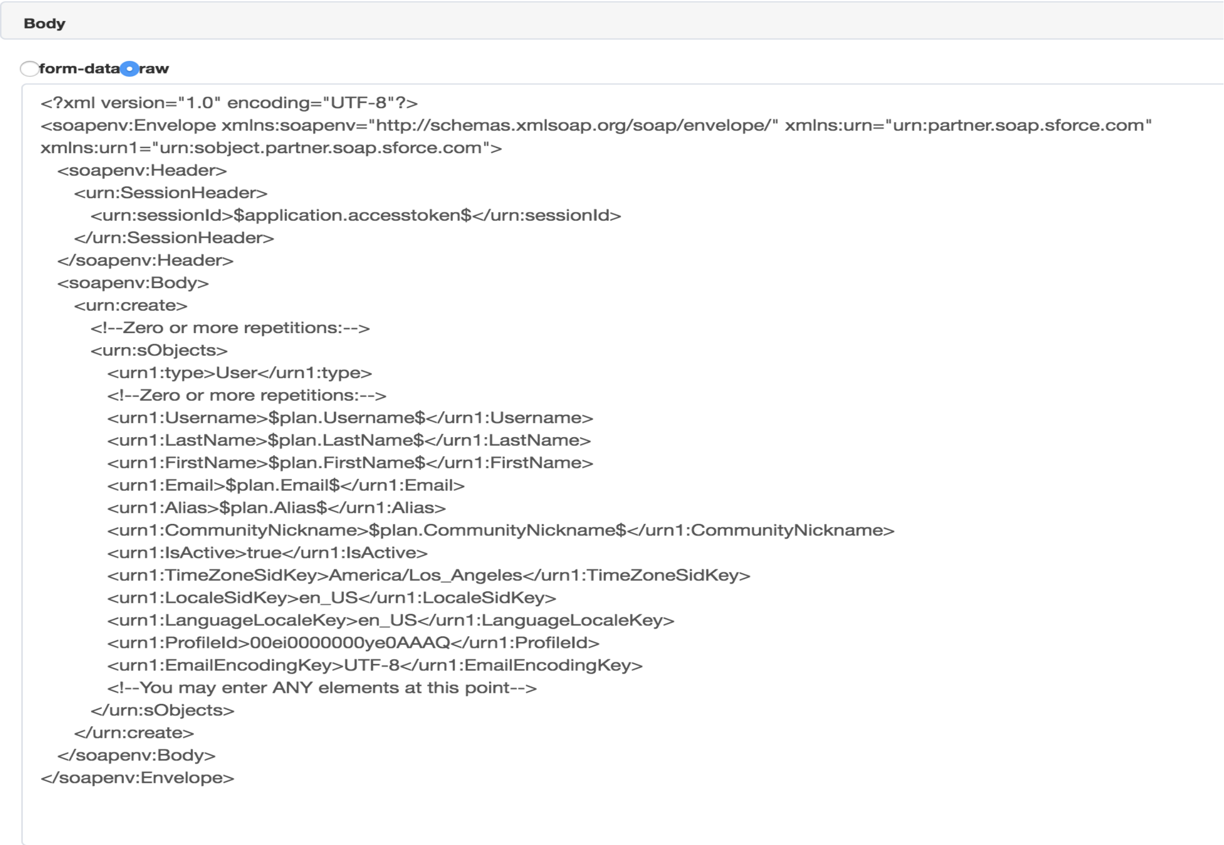


Body

○form-data ●raw

```
{
"member_email" : "$plan.member_email$",
"member_given_name" : "$plan.member_given_name$",
"member_surname" : "$plan.member_surname$" ,
"send_welcome_email" : "$plan.send_welcome_email$",
"member_external_id" : "$plan.member_external_id$"
}
```

- (For XML) To create account for XML payload:

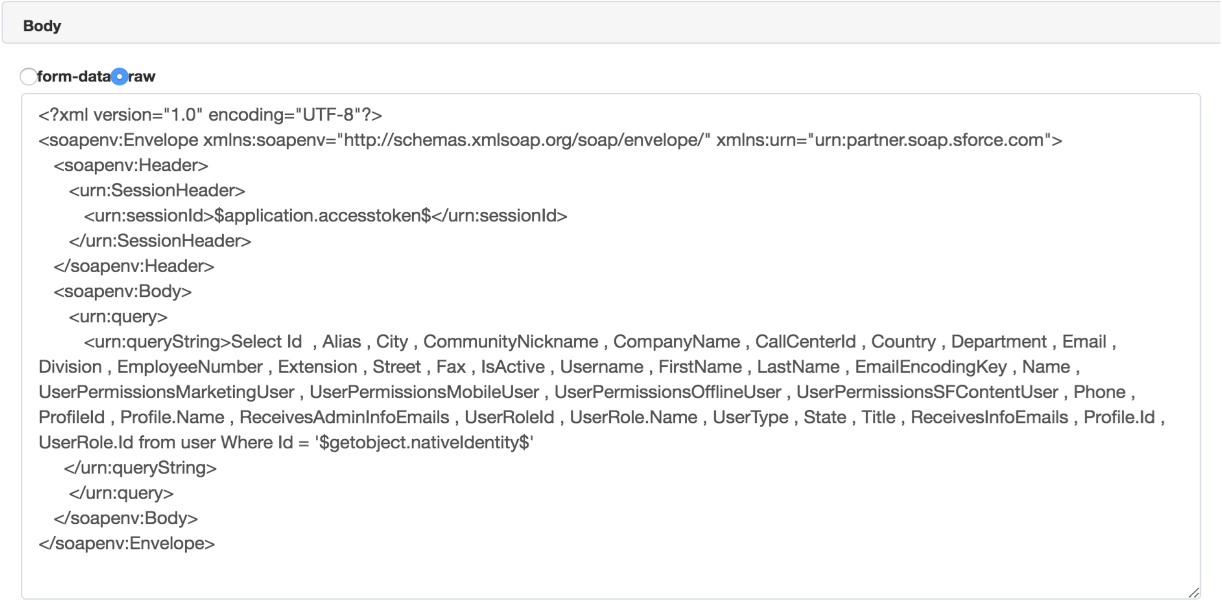


Body

○form-data ●raw

```
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:urn="urn:partner.soap.sforce.com"
xmlns:urn1="urn:sobject.partner.soap.sforce.com">
  <soapenv:Header>
    <urn:SessionHeader>
      <urn:sessionId>$application.accesstoken$</urn:sessionId>
    </urn:SessionHeader>
  </soapenv:Header>
  <soapenv:Body>
    <urn:create>
      <!--Zero or more repetitions:-->
      <urn:sObjects>
        <urn1:type>User</urn1:type>
        <!--Zero or more repetitions:-->
        <urn1:Username>$plan.Username$</urn1:Username>
        <urn1:LastName>$plan.LastName$</urn1:LastName>
        <urn1:FirstName>$plan.FirstName$</urn1:FirstName>
        <urn1:Email>$plan.Email$</urn1:Email>
        <urn1:Alias>$plan.Alias$</urn1:Alias>
        <urn1:CommunityNickname>$plan.CommunityNickname$</urn1:CommunityNickname>
        <urn1:IsActive>true</urn1:IsActive>
        <urn1:TimeZoneSidKey>America/Los_Angeles</urn1:TimeZoneSidKey>
        <urn1:LocaleSidKey>en_US</urn1:LocaleSidKey>
        <urn1:LanguageLocaleKey>en_US</urn1:LanguageLocaleKey>
        <urn1:ProfileId>00ei0000000ye0AAAQ</urn1:ProfileId>
        <urn1:EmailEncodingKey>UTF-8</urn1:EmailEncodingKey>
        <!--You may enter ANY elements at this point-->
      </urn:sObjects>
    </urn:create>
  </soapenv:Body>
</soapenv:Envelope>
```

To get object for XML payload:



Body

○ form-data ● raw

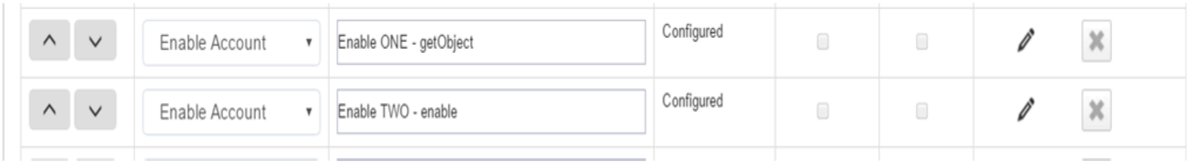
```
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:urn="urn:partner.soap.sforce.com">
  <soapenv:Header>
    <urn:SessionHeader>
      <urn:sessionId>$application.accesstoken$</urn:sessionId>
    </urn:SessionHeader>
  </soapenv:Header>
  <soapenv:Body>
    <urn:query>
      <urn:queryString>Select Id  , Alias , City , CommunityNickname , CompanyName , CallCenterId , Country , Department , Email ,
Division , EmployeeNumber , Extension , Street , Fax , IsActive , Username , FirstName , LastName , EmailEncodingKey , Name ,
UserPermissionsMarketingUser , UserPermissionsMobileUser , UserPermissionsOfflineUser , UserPermissionsSFContentUser , Phone ,
ProfileId , Profile.Name , ReceivesAdminInfoEmails , UserRoleId , UserRole.Name , UserType , State , Title , ReceivesInfoEmails , Profile.Id ,
UserRole.Id from user Where Id = '$getobject.nativeIdentity$'
     </urn:queryString>
    </urn:query>
  </soapenv:Body>
</soapenv:Envelope>
```

Enable/Disable

Set the get object endpoint for enable/disable operation as in the POST method the complete object would be required to update and not single attribute. Hence first endpoint getObject would fetch the whole account and later the endpoint would update the payload with all the required attributes using the response of the first endpoint.

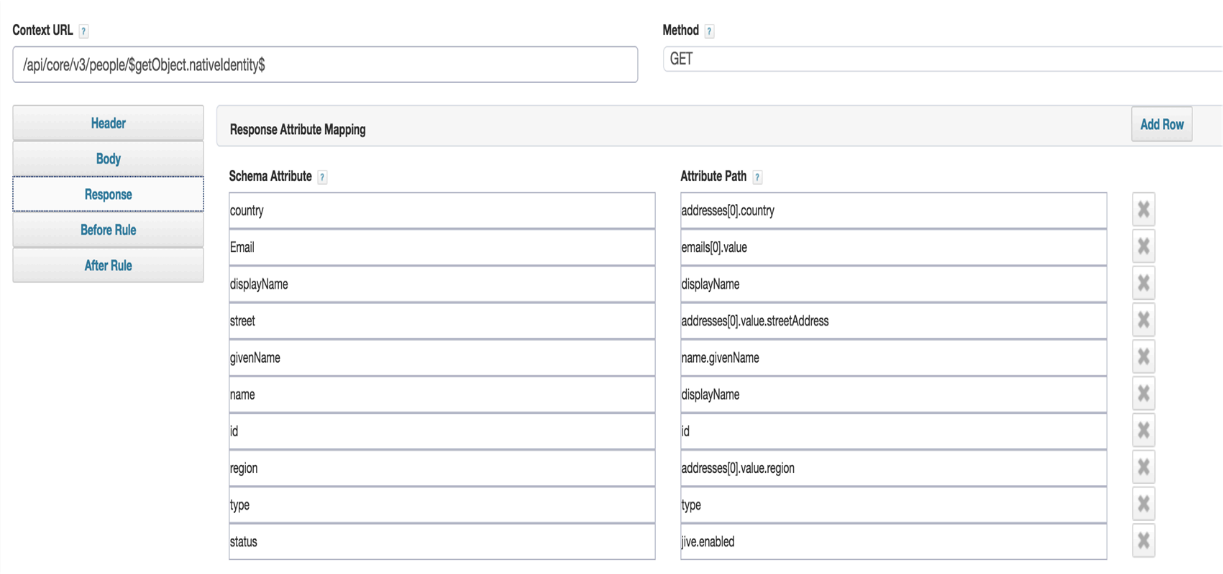
Perform the following steps to get object for Enable operation with PUT method

1. Configure the first endpoint to get object for Enable.



^ v	Enable Account ▾	Enable ONE - getObject	Configured	□	□	✎	✖
^ v	Enable Account ▾	Enable TWO - enable	Configured	□	□	✎	✖

2. Configuration for the first endpoint.



Context URL: /api/core/v3/people/$getObject.nativeIdentity$

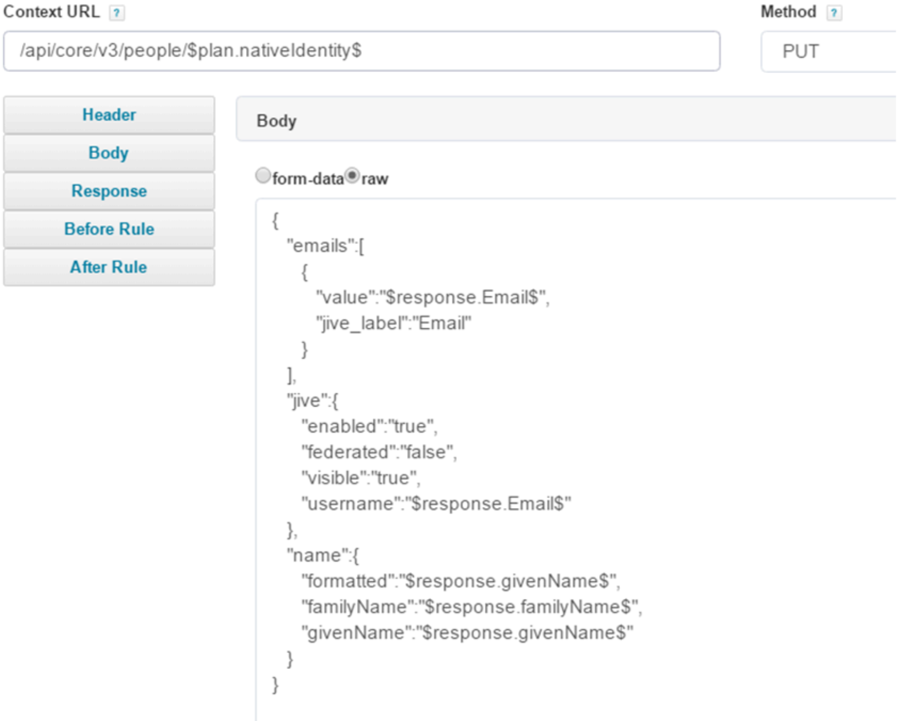
Method: GET

Response Attribute Mapping

Schema Attribute	Attribute Path
country	addresses[0].country
Email	emails[0].value
displayName	displayName
street	addresses[0].value.streetAddress
givenName	name.givenName
name	displayName
id	id
region	addresses[0].value.region
type	type
status	jive.enabled

This endpoint retrieves getObject for account for which Provisioning Operation is performed.

3. Configuration for second endpoint for Enable endpoint as shown in the following figure:



Context URL: /api/core/v3/people/$plan.nativeIdentity$

Method: PUT

Body

○ form-data ● raw

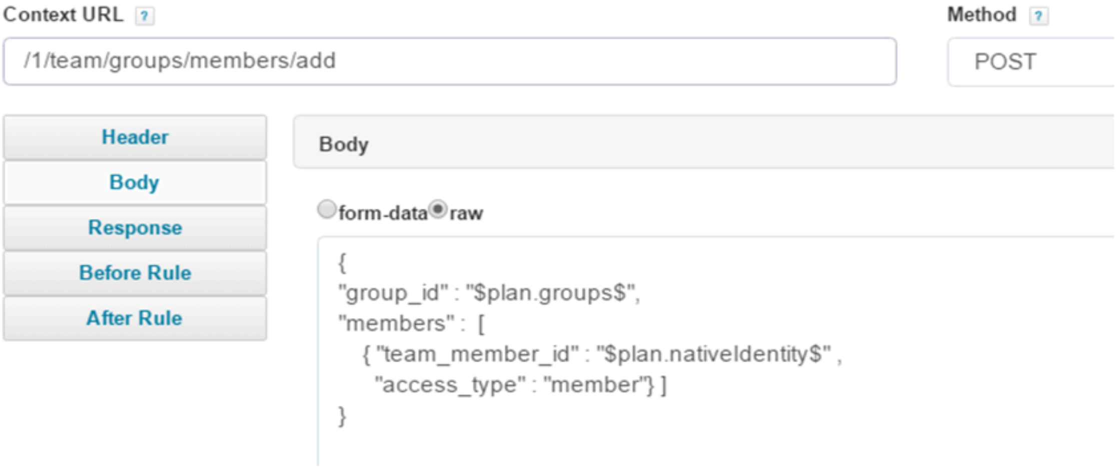
```
{
  "emails":[
    {
      "value":"$response.Email$",
      "jive_label":"Email"
    }
  ],
  "jive":{
    "enabled":"true",
    "federated":"false",
    "visible":"true",
    "username":"$response.Email$"
  },
  "name":{
    "formatted":"$response.givenName$",
    "familyName":"$response.familyName$",
    "givenName":"$response.givenName$"
  }
}
```

Note: It may be required to update few attribute for performing enable/disable operation

Similar steps are to be performed for Disable operation.

Add/Remove Entitlement

Following is an example of the Body entry for Add Entitlement:



Context URL: /1/team/groups/members/add

Method: POST

Body

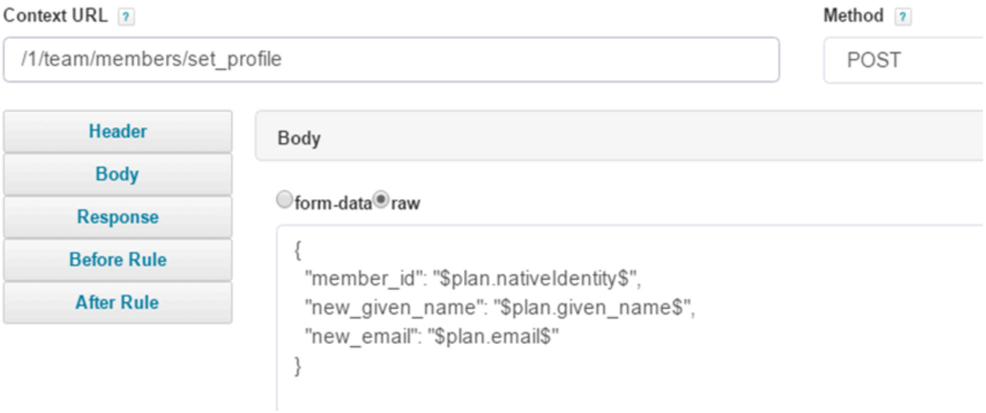
○ form-data ● raw

```
{
"group_id" : "$plan.groups$",
"members" : [
   { "team_member_id" : "$plan.nativeIdentity$" ,
     "access_type" : "member"} ]
}
```

On similar basis as above example the Body entry must be updated for Remove Entitlement.

Update Account

Following is an example of the Body entry for Update Account:



Context URL: /1/team/members/set_profile

Method: POST

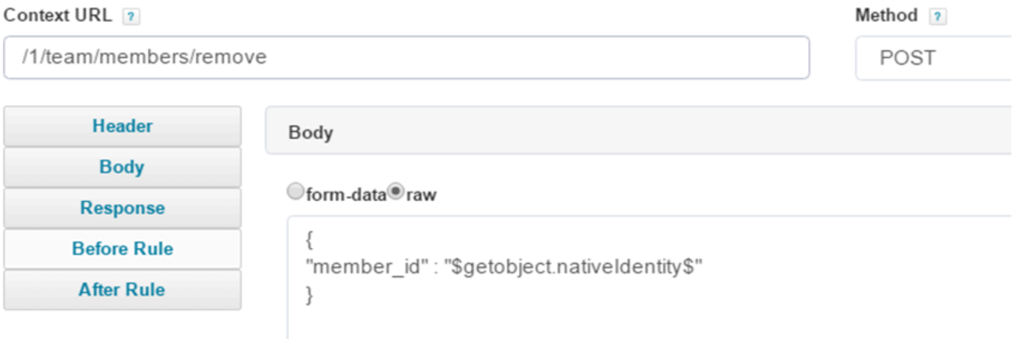
Body

○ form-data ● raw

```
{
 "member_id": "$plan.nativeIdentity$",
 "new_given_name": "$plan.given_name$",
 "new_email": "$plan.email$"
}
```

Delete Account

Following is an example of the Body entry for Delete Account:



Context URL: /1/team/members/remove

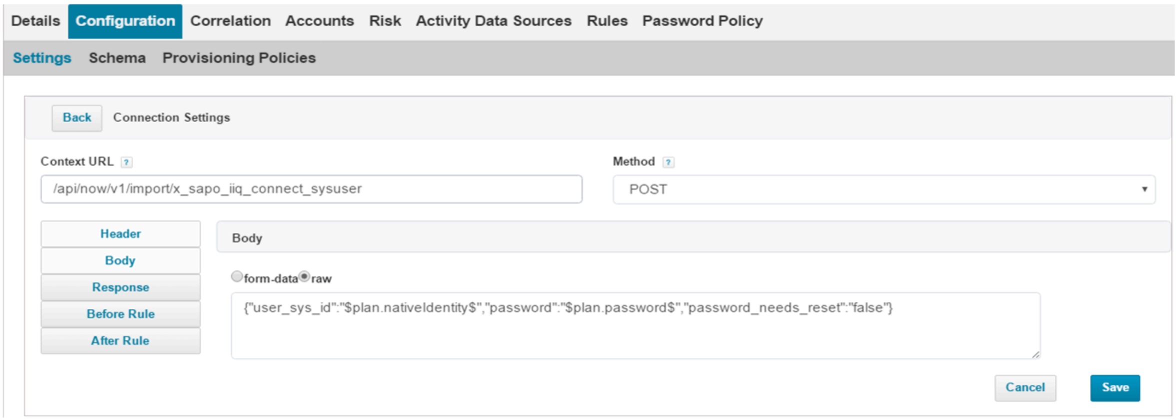
Method: POST

○ form-data ● raw

```
{
"member_id" : "$getobject.nativeIdentity$"
}
```

Change Password

Following is an example of the Body entry for Change Password:



Context URL: /api/now/v1/import/x_sapo_iiq_connect_sysuser

Method: POST

○ form-data ● raw

```
{"user_sys_id":"$plan.nativeIdentity$","password":"$plan.password$","password_needs_reset":"false"}
```

Troubleshooting

1 - TLS communication failure for Web Services Connector when IdentityIQ is deployed on IBM WebSphere Application Server

TLS certificates can be imported and configured in WebSphere's trust stores through **SSL certificate and key management ==> Key stores and certificates ==> CellDefaultTrustStore**.

However, the Web Services Connector depends on the trust store of the JDK during TLS communication and does not respect the WebSphere's trust stores. This results in TLS communication failure.

Resolution: An additional configuration is required to override and set the default trust store as WebSphere’s trust store. Perform the following:

1. Select **Servers ==> Application Servers ==>_ server_name_ ==> Process Definition ==> Java Virtual Machine ==> Custom Properties ==> New**

2. Add the following properties:

   - Property Name: `javax.net.ssl.trustStore`
     Value: **<WebSphere truststore file path>**
   - Property Name: `javax.net.ssl.trustStorePassword`
     Value: **<Truststore Password>**

2 - Test Connection fails when Web Services is using TLS 1.0 on WebSphere Application Server

With this release Web Services Connector is enhanced to enforce the secure communication. Hence this may cause **Test Connection** to fail with the following error if IBM JDK 1.8 is used:

```
[ConnectorException] [Possible suggestions] Ensure configuration parameters are
correct with a valid format, Ensure active network connectivity between Source and
Target system. [Error details] javax.net.ssl.SSLHandshakeException: Received fatal
alert: handshake_failure
```

Resolution: Perform the following to add the `com.ibm.jsse2.overrideDefaultTLS` property in the Java properties:

1. Navigate to **Servers ==> Java and Process Management ==>Process Definition ==> JVM ==> Custom?Properties** and add the following property:
   `com.ibm.jsse2.overrideDefaultTLS`

   OR

   Add in **jvm.options** file. For more information, see the following the link:

   https://www.ibm.com/support/knowledgecenter/SSD28V_liberty/com.ibm.websphere.wlp.core.doc/ae/twlp_admin_customvars.html
2. Restart the server after changes.