



Developers Handbook for REST Integration

Copyright

saviynt.com

© 2024 Saviynt. All rights reserved. No part of this document may be reproduced or used in any manner without the prior written permission of the copyright owner.

CONTENTS

DEVELOPERS HANDBOOK	1
---------------------------	---

ConnectionJSON

This parameter is used to provide an authorization mechanism for target applications for making API calls. Each authentication is provided with a name such as **userAuth**. This name is referred in the **ImportAccountEntJSON** with the field name as a **connection**.

Sample **ConnectionJSON**:

JSON

```
{
  "authentications": {
    "userAuth": {
      "authType": "oauth2",
      "url": "https://<domain name>/api/v18.2/auth",
      "httpMethod": "POST",
      "httpParams": {
        "username": "<Username>",
        "password": "<Password>"
      },
      "httpHeaders": {
        "contentType": "application/x-www-form-urlencoded"
      },
      "httpContentType": "application/x-www-form-urlencoded",
```

```
    "expiryError": "ExpiredAuthenticationToken",
    "authError": [
      "InvalidAuthenticationToken",
      "AuthenticationFailed",
      "FAILURE",
      "INVALID_SESSION_ID"
    ],
    "timeOutError": "Read timed out",
    "errorPath": "errors.type",
    "maxRefreshTryCount": 5,
    "tokenResponsePath": "sessionId",
    "tokenType": "Bearer",
    "accessToken": "<access token>"
  }
}
```

Sample **ConnectionJSON** when the access token is available as part of response headers:

JSON

```
{
  "authentications": {
```

```
"acctAuth": {
  "authType": "oauth2",
  "url": "",
  "httpMethod": "POST",
  "httpParams": {

  },
  "httpHeaders": {
    "Content-Type": "application/x-www-form-urlencoded"
  },
  "httpContentType": "application/x-www-form-urlencoded",
  "expiryError": "ExpiredAuthenticationToken",
  "authError": [
    "USER_AUTHENTICATION_FAILED"
  ],
  "timeOutError": "error",
  "errorPath": "error",
  "maxRefreshTryCount": 3,
  "tokenResponsePath": "#HEADERS#Set-Cookie",
  "tokenType": "",
  "authHeaderName": "Cookie",
  "retryFailureStatusCode": [
```

```
    401
  ],
  "accessToken": "sdfghjk"
}
}
```



Note

#HEADERS# is mandatory, if access token is present in the response headers.

Common Features

1. **Multiple Authentications:** Supports multiple authentications where more than one authentication mechanism is required. For example, for Azure AD, there are graph.windows.net APIs and [http://graph.microsoft.com](https://graph.microsoft.com) APIs and each API has a different authentication mechanism and hence two authentication constructs are used in **ConnectionJSON**.

Example:

JSON

```
{
  "authentications": {
    "userAuth": {
```

```
"authType": "oauth2",
"url": "https://<domain name>/<<TenantID>>/oauth2/token",
"httpMethod": "POST",
"httpParams": {
  "grant_type": "client_credentials",
  "client_secret": "<<ClientSecret>>",
  "client_id": "<<ClientID>>",
  "resource": "https://graph.microsoft.com/"
},
"httpHeaders": {
  "contentType": "application/x-www-form-urlencoded"
},
"httpContentType": "application/x-www-form-urlencoded",
"expiryError": "ExpiredAuthenticationToken",
"authError": [
  "InvalidAuthenticationToken"
],
"timeOutError": "Read timed out",
"errorPath": "error.code",
"maxRefreshTryCount": 5,
"tokenResponsePath": "access_token",
"tokenType": "Bearer",
```



```
    "accessToken": "Bearer abc"
  },
  "entAuth": {
    "authType": "oauth2",
    "url": "https://<domain name>/<<TenantID>>/oauth2/token",
    "httpMethod": "POST",
    "httpParams": {
      "grant_type": "client_credentials",
      "client_secret": "<<ClientSecret>>",
      "client_id": "<<ClientID>>",
      "resource": "https://graph.windows.net/"
    },
    "httpHeaders": {
      "contentType": "application/x-www-form-urlencoded"
    },
    "httpContentType": "application/x-www-form-urlencoded",
    "expiryError": "ExpiredAuthenticationToken",
    "authError": [
      "InvalidAuthenticationToken",
      "Authentication_MissingOrMalformed"
    ],
    "timeOutError": "Read timed out",
  }
}
```

```
    "errorPath": "odata~dot#error.code",
    "maxRefreshTryCount": 3,
    "tokenResponsePath": "access_token",
    "tokenType": "Bearer",
    "accessToken": "Bearer abcde"
  }
}
```

2. **SSL Parameter:** Installing SSL certificates is sufficient to make https API calls, however, if target applications expect customizations such as the TLS version, you can pass SSL parameters while making API calls.

JSON

```
{
  "authentications": {
    "acctAuth": {
      "authType": "oauth2",
      "url": "XXXXXXXXXXXX",
      "httpMethod": "POST",
      "httpParams": {
        "grant_type": "client_cert"
      },
    },
  },
}
```

```
"httpContentType": "application/x-www-form-urlencoded",
"ssl": {
  "keyFile": "opt/java/jdk1.8.0_181/jre/lib/security/cacerts",
  "keyFilePassword": "changeit",
  "keyManagerAlgorithm": "SunX509",
  "keyStoreType": "JKS",
  "sslAlgorithmName": "TLSv1.2"
},
"httpHeaders": {
  "Content-Type": "application/x-www-form-urlencoded",
  "Authorization": "Basic xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx"
},
"retryFailureStatusCode": [
  401,
  403
],
"expiryError": "ExpiredAuthenticationToken",
"authError": [
  "SESSION_NOT_VALID",
  "AuthenticationFailed",
  "ExpiredJwtException",
  "401 Unauthorized"
```

```

    ],
    "timeOutError": "Read timed out",
    "errorPath": "code",
    "maxRefreshTryCount": 6,
    "tokenResponsePath": "access_token",
    "tokenType": "Bearer",
    "accessToken": "Bearer xxxxxxxxxxxxxxxxx"
  }
}
}

```

3. **Connection JSON Validation:** The connector supports validating authType as oauth2. To do this, populate the http parameters in the `testConnectionParams` attribute. It validates the connection parameters and prompts **Successful** or **Failed** status based on the provided credentials while saving the connection.

From Release v23.10 onwards, if you specify an incorrect value or blank value for the `testConnectionParams` attribute in the **ConnectionJSON** parameter and then click **Save & Test Connection**, the connector displays a detailed error message in the debug log file and the **Job Log Detail** page of the **Test Connections** Job. The message includes the name of the failed connection, type of error with exception that occurred in the target, and any additional details that might help in troubleshooting the connection issue.

For more information about troubleshooting issues encountered during saving and testing a connection, see *Errors while Saving and Testing a Connection* under [Troubleshooting Issues](#) in the *REST Integration Guide*.

Example:

JSON

```
{
  "authentications": {
    "acctAuth": {
      "authType": "Basic",
      "url": "https://<domain name>",
      "httpMethod": "POST",
      "httpParams": {},
      "httpHeaders": {},
      "httpContentType": "text/html",
      "properties": {
        "userName": "username",
        "password": "password"
      },
      "expiryError": "Couldn't authenticate you",
      "authError": [
        "Couldn't authenticate you"
      ],
      "timeOutError": "Read timed out",
      "errorPath": "error",
    }
  }
}
```

```
"maxRefreshTryCount": 5,
"tokenResponsePath": "access_token",
"tokenType": "Basic",
"accessToken": "Basic asdfghjkl",
"testConnectionParams": {
  "http": {
    "url": "https://<domain name>/api/v2/users.json",
    "httpHeaders": {
      "Authorization": "${access_token}"
    },
    "httpContentType": "application/json",
    "httpMethod": "GET"
  },
  "successResponse": [],
  "successResponsePath": "",
  "errors": [
    "Couldn't authenticate you"
  ],
  "errorPath": "error"
}
```

```
}  
}
```

4. Authentication Types

The following authentication types are supported:

- a. **Basic:** Used for authentication where only username and password are required to authorize the REST API calls. Values for username or password fields must be provided in the `properties` attribute. You can leave the HTTP related parameters empty.

Example:

JSON

```
{  
  "authentications": {  
    "acctAuth": {  
      "authType": "Basic",  
      "url": "<URL>",  
      "httpMethod": "POST",  
      "httpParams": {},  
      "httpHeaders": {},  
      "httpContentType": "text/html",  
      "properties": {
```

```
    "userName": "<<USERNAME>>/token",
    "password": "<<PASSWORD>>"
  },
  "expiryError": "ExpiredAuthenticationToken",
  "authError": [
    "InvalidAuthenticationToken",
    "AuthenticationFailed"
  ],
  "timeOutError": "Read timed out",
  "testname": "<test URL>",
  "errorPath": "error.code",
  "maxRefreshTryCount": 5,
  "tokenResponsePath": "access_token",
  "tokenType": "Basic",
  "accessToken": "Basic <<TOKEN>>",
  "apiRateLimitConfig": {"retryAfterCalls": 100, "retryWaitSeconds": 60}
}
}
```


- b. **BasicWithAccessToken**: Used for authentication when the access token is obtained using the basic username and password credentials. In this type, the primary http parameters are `url`, `httpHeaders`, `httpMethod` and `httpContentType`.



Note

The refresh token for applications such as Hydra expires once it is used or the time-period is elapsed.

The `refreshTokenFlag` is added to refresh the token based on your selection such as set it to 'true' refreshes the refresh token immediately after generating new access token for subsequent use. The refresh token calls the refresh token API right after the access token generation.

Example 1:

JSON

```
{
  "authentications": {
    "acctAuth": {
      "authType": "BasicWithAccessToken",
      "url": "{url}",
      "httpMethod": "POST",
      "httpHeaders": {"Accept": "application/json"},
      "properties": {"userName": "<username>", "password": "<password>"},
    }
  }
}
```

```
"httpContentType": "application/json",
"expiryError": "ExpiredAuthenticationToken",
"retryFailureStatusCode": [403,401,500],
"authError": [
  "InvalidAuthenticationToken",
  "AuthenticationFailed",
  "Authentication_MissingOrMalformed",
  "Authentication_ExpiredToken"
],
"timeOutError": "Read timed out",
"errorPath": "error.code",
"maxRefreshTryCount": 5,
"tokenResponsePath": "access_token",
"tokenType": "Bearer",
"accessToken": "Bearer xyz"
}
}
}
```

Example 2: To refresh the token for BasicwithAccesToken auth type when the import job fails with 417 error code, use a format similar to the following:

JSON

```
{
  "authentications": {
    "user": {
      "authType": "BasicWithAccessToken",
      "url": "<URL>",
      "httpMethod": "POST",
      "httpHeaders": {
        "Accept": "application/json"
      },
      "httpParams": {
        "UserId": "<user ID>",
        "Password": "<password>",
        "Server": "<Server name>"
      },
      "properties": {
        "userName": "<user name>",
        "password": "<password>"
      },
      "httpContentType": "application/json",
      "expiryError": "ExpiredAuthenticationToken",
      "retryFailureStatusCode": [
```

```
    403,  
    401,  
    417,  
    500  
  ],  
  "authError": [  
    "InvalidAuthenticationToken",  
    "AuthenticationFailed",  
    "Authentication_MissingOrMalformed",  
    "Authentication_ExpiredToken"  
  ],  
  "timeOutError": "Read timed out",  
  "errorPath": "error.code",  
  "maxRefreshTryCount": 5,  
  "tokenResponsePath": "token",  
  "tokenType": "",  
  "accessToken": "Bearer xyz"  
}  
}  
}
```

- c. **BasicWithHMAC**: Used when authentication requires Hash-based Message Authentication Code (HMAC) signed signature using the credentials passed in the `properties` attribute such as `"properties": {"IKEY": "xyz", "SKEY": "abc"}.`

Example:

JSON

```
{
  "authentications": {
    "acctAuth": {
      "authType": "BasicWithHmac",
      "url": "<url>",
      "httpMethod": "POST",
      "properties": {
        "IKEY": "<<IKEY>>",
        "SKEY": "<<SKEY>>"
      },
      "authError": [
        "InvalidAuthenticationToken",
        "AuthenticationFailed",
        "Authentication_MissingOrMalformed",
        "Authentication_ExpiredToken"
      ]
    }
  }
}
```

```

    ],
    "errorPath": "error.code",
    "maxRefreshTryCount": 5,
    "tokenResponsePath": "access_token",
    "tokenType": "Basic",
    "accessToken": "Basic xyz"
  }
}
}

```

- d. **Veracode:** To connect using Veracode APIs for Hash-based Message Authentication Code (HMAC) authentication.
(Available from Release v23x onwards)

JSON

```

{
  "authentications": {
    "acctAuth": {
      "authType": "HMAC",
      "url": "https://<domain name>/api/authn/v2/users",
      "httpMethod": "POST",
      "properties": {
        "IKEY": "api_id",

```

```
    "SKEY": "secret_key"
  },
  "authError": [
    "401 Unauthorized",
    "403 Forbidden"
  ],
  "errorPath": "http_status",
  "maxRefreshTryCount": 1,
  "tokenResponsePath": "access_token",
  "accessToken": "abc",
  "hmacType": "veracode",
  "testConnectionParams": {
    "http": {
      "url": "https://<domain name>/api/authn/v2/users",
      "httpHeaders": {
        "Authorization": "${access_token}"
      },
      "httpContentType": "application/json",
      "httpMethod": "GET"
    },
    "successResponses": {
      "statusCode": [
```

```
        200,  
        201,  
        202,  
        203,  
        204,  
        205  
      ]  
    }  
  }  
}
```

e. **Cookies:** Used for authentication using cookies.

JSON

```
{  
  "authentications": {  
    "acctAuth": {  
      "authType": "cookies",  
      "url": "",  
      "httpMethod": "POST",  
    },  
  },  
}
```



```
"httpParams": {
  "username": "<specify username>",
  "password": "<specify password>",
  "apiKey": "${apiKey}",
  "timestamp": "${timestamp}"
},
"httpHeaders": {
  "contentType": "application/json"
},
"cookies": "${cookies}",
"properties": {
  "apiKey": "${apiKey}"
},
"httpContentType": "application/json",
"expiryError": "ExpiredAuthenticationToken",
"authError": [
  "SESSION_NOT_VALID",
  "AuthenticationFailed",
  "HTTP error code : 401"
],
"timeOutError": "Read timed out",
"errorPath": "code",
```

```
    "maxRefreshTryCount": 5,  
    "tokenResponsePath": "",  
    "tokenType": "",  
    "accessToken": "<specify access token>",  
    "retryFailureStatusCode": [  
        500,  
        502,  
        401  
    ]  
  }  
}
```

f. **SignedHeaders:** Used for authentication using SignedHeaders.

JSON

```
{  
  "authentications": {  
    "acctAuth": {  
      "authType": "SignedHeaders",  
      "httpMethod": "POST",  
      "httpHeaders": {
```

```
    "contentType": "application/x-www-form-urlencoded"
  },
  "httpContentType": "application /x-www-form-urlencoded",
  "properties": {
    "userId": "pivotal",
    "keyFilePath": "<specify key file path>"
  },
  "expiryError": "ExpiredAuthenticationToken",
  "authError": [
    "InvalidAuthenticationToken",
    "AuthenticationFailed"
  ],
  "timeOutError": "Read timed out",
  "errorPath": "message",
  "maxRefreshTryCount": 5,
  "tokenResponsePath": "access_token",
  "tokenType": "Bearer",
  "accessToken": "<specify access token>"
}
}
```

- g. **JWT**: Used for authentication using JSON Web Tokens (JWT). For this authentication, specify the values for the following attributes: `httpParamsName`, `jwtConfig`, `signedAlgorithm`, `key`, and `jwtExpiryDuration`.

To specify `httpParamsName` as assertion, use the following format:

JSON

```
{
  "authentications": {
    "acctAuth": {
      "authType": "Jwt",
      "httpParamsName": "assertion",
      "jwtConfig": {
        "jwtHeader": {
          "alg": "<specify algorithm>",
          "typ": "JWT",
          "kid": "<specify key ID>"
        },
        "jwtPayload": {
          "iss": "<specify ISS>",
          "sub": "<specify subject>",
          "aud": "<specify audience>",
          "scope": "https://www.googleapis.com/auth/admin.directory.user https://www.googleapis.com/auth/admin.directory.user"
        }
      }
    }
  }
}
```

```
    },
    "signedAlgorithm": "<specify signed algorithm>",
    "key": "<specify key>",
    "jwtExpiryDuration": 120
  },
  "url": "<specify URL>",
  "httpMethod": "POST",
  "httpParams": {
    "grant_type": "urn:ietf:params:oauth:grant-type:jwt-bearer"
  },
  "httpContentType": "application/x-www-form-urlencoded",
  "retryFailureStatusCode": [
    401,
    500,
    400
  ],
  "authError": [
    "SESSION_NOT_VALID",
    "AuthenticationFailed",
    "ExpiredJwtException",
    "401 Unauthorized",
    "401",
  ]
}
```

```

    "You couldn't be authenticated"
  ],
  "errorPath": "code",
  "maxRefreshTryCount": 5,
  "tokenResponsePath": "access_token",
  "tokenType": "Bearer",
  "accessToken": ""
}
}
}

```

To specify `httpParamsName` as `jwt_token`, use the following format:

JSON

```

{
  "authentications": {
    "acctAuth": {
      "authType": "Jwt",
      "httpParamsName": "jwt_token",
      "jwtConfig": {
        "jwtHeader": {
          "alg": "<specify algorithm>",

```

```
    "typ": "JWT"
  },
  "jwtPayload": {
    "exp": 1675754298,
    "iss": "<specify ISS>",
    "sub": "<specify subject>",
    "aud": "<specify audience>"
  },
  "signedAlgorithm": "<specify signed algorithm>",
  "key": "<specify key>",
  "jwtExpiryDuration": 120
},
"url": "<specify URL>",
"httpMethod": "POST",
"httpParams": {
  "client_id": "<client_id>",
  "client_secret": "<client_secret>"
},
"httpContentType": "multipart/form-data",
"retryFailureStatusCode": [
  401,
  500,
```

```

    400
  ],
  "authError": [
    "SESSION_NOT_VALID",
    "AuthenticationFailed",
    "ExpiredJwtException",
    "401 Unauthorized",
    "401",
    "You couldn't be authenticated"
  ],
  "errorPath": "code",
  "maxRefreshTryCount": 5,
  "tokenResponsePath": "access_token",
  "tokenType": "Bearer",
  "accessToken": "Bearer abcd"
}
}
}

```

- h. **OAuth2:** Supports OAuth protocol for API authorization with various types of OAuth2 mechanisms. You must provide the token that the connector must regenerate when a wrong token is specified or existing token has expired.

To regenerate the token, provide the values for the following attributes: `url`, `httpMethod`, `httpParams`, `httpHeaders`, and `httpContentType`.

The following tokens are used for OAuth:

- **Static Access Token** - Some cloud applications support hardcoded access token that never expires until it is manually revoked from the target application. In this case, an attribute named `accessToken` is populated with the hardcoded token and other parameters are left with dummy values.

Example:

JSON

```
{
  "authentications": {
    "userAuth": {
      "authType": "oauth2",
      "url": "<URL>",
      "httpMethod": "POST",
      "httpParams": {},
      "httpHeaders": {
        "contentType": "application/x-www-form-urlencoded"
      },
      "httpContentType": "application/x-www-form-urlencoded",
      "expiryError": "ExpiredAuthenticationToken",
    }
  }
}
```

```
    "authError": [
      "InvalidAuthenticationToken",
      "AuthenticationFailed",
      "FAILURE",
      "INVALID_SESSION_ID"
    ],
    "timeOutError": "Read timed out",
    "errorPath": "errors.type",
    "maxRefreshTryCount": 5,
    "tokenResponsePath": "sessionId",
    "tokenType": "Bearer",
    "accessToken": "Bearer <token>"
  }
}
```

- **Static Refresh Token:** In this scenario, access tokens expire after a specific time period but refresh tokens never expire. Therefore, refresh tokens are used as new access tokens. To regenerate a refresh token, specify values for the following attributes: `url`, `httpMethod`, `httpParams`, `httpHeaders`, and `httpContentType`.

Example:

JSON

```
{
  "authentications": {
    "acctAuth": {
      "authType": "oauth2",
      "url": "https://<domain name>/oauth/v2/token",
      "httpMethod": "POST",
      "httpParams": {
        "grant_type": "refresh_token",
        "client_id": "",
        "client_secret": "",
        "redirect_uri": "",
        "refresh_token": "<refresh token>"
      },
      "expiryError": "ExpiredAuthenticationToken",
      "retryFailureStatusCode": [
        401
      ],
      "timeOutError": "Read timed out",
      "errorPath": "error",
      "maxRefreshTryCount": 3,
      "tokenResponsePath": "access_token",
      "tokenType": "Bearer",
    }
  }
}
```

```
    "accessToken": "Bearer abcd"
  }
}
```

- **Renew Access Token and Refresh Token using Single API:** In this OAuth mechanism, the access and refresh tokens expire after a time period, and the latest refresh token is used to regenerate new access and refresh tokens. To regenerate these tokens, specify the values for the following attributes: `refreshType`, `refreshTokenResponseP`
`ath`, and `refreshToken`.

Example:

JSON

```
{
  "authentications": {
    "userAuth": {
      "authType": "oauth2",
      "url": "https://<domain name>/v1/access_token",
      "httpMethod": "POST",
      "httpParams": {
        "client_id": "<client ID>",
        "client_secret": "<client secret>",
        "grant_type": "refresh_token",
        "refresh_token": "${refresh_token}"
      }
    }
  }
}
```

```
},
"httpHeaders": {
  "Content-Type": "application/x-www-form-urlencoded"
},
"httpContentType": "application/x-www-form-urlencoded",
"expiryError": "ExpiredAuthenticationToken",
"authError": [
  "USER_AUTHENTICATION_FAILED",
  "PARTNER_AUTHENTICATION_FAILED",
  "AuthenticationFailed"
],
"refreshType": "RefreshToken",
"refreshTokenResponsePath": "refresh_token",
"refreshToken": "<refresh token>",
"timeOutError": "Read timed out",
"errorPath": "errorCode",
"maxRefreshTryCount": 5,
"tokenResponsePath": "access_token",
"tokenType": "Bearer",
"retryFailureStatusCode": [
  401
],
```

```

    "accessToken": "Bearer <token>"
  }
}

```

- **Renew Access Token and Refresh Token using Different API:** In some OAuth mechanisms, access token and refresh token both expire after a certain time period. After the refresh token is renewed, you need to place a call to obtain the new refresh token to use it for obtaining the new access token. To regenerate a token, specify the values for the following attributes: `refreshType`, `refreshTokenResponsePath`, `refreshToken`, `refreshTokenAuthError`, `refreshTokenErrorPath`, and `refreshTokenCall`.

Example:

JSON

```

{
  "authentications": {
    "userAuth": {
      "authType": "oauth2",
      "url": "https://<domain name>/oauth/token",
      "httpMethod": "POST",
      "httpParams": {
        "company_id": "<>",
        "client_id": "<>",
        "grant_type": "<>",

```

```
    "assertion": "${refresh_token}"
  },
  "httpHeaders": {
    "Content-Type": "application/x-www-form-urlencoded"
  },
  "httpContentType": "application/x-www-form-urlencoded",
  "authError": [
    "Unable to authenticate the client",
    "Invalid OAuth token Bearer"
  ],
  "retryFailureStatusCode": [
    401
  ],
  "errorPath": "",
  "maxRefreshTryCount": 5,
  "tokenResponsePath": "access_token",
  "refreshType": "RefreshToken",
  "tokenType": "Bearer",
  "accessToken": "Bearer asdsdfghjk",
  "refreshToken": "<>",
  "refreshTokenAuthError": [
    "Unable to retrieve SAML assertion",
```

```

    "The provided SAML assertion is expired"
  ],
  "refreshTokenErrorPath": "errorMessage",
  "refreshTokenCall": {
    "refreshTokenResponsePath": "",
    "url": "https://<domain name>/oauth",
    "httpMethod": "POST",
    "httpParams": {
      "client_id": "<>",
      "user_id": "<>",
      "token_url": "https://<domain name>/oauth/token",
      "private_key": "<>"
    },
    "httpHeaders": {
      "Content-Type": "application/x-www-form-urlencoded"
    },
    "httpContentType": "application/x-www-form-urlencoded"
  }
}
}
}

```


5. **Mask values of sensitive attributes in logs:** The connector supports masking values of sensitive attributes in logs using the `mask` `InLogs` attribute.

Example:

JSON

```
{
  "authentications": {
    "acctAuth": {
      "authType": "Basic",
      "url": "<URL>",
      "httpMethod": "POST",
      "httpParams": {},
      "httpHeaders": {},
      "httpContentType": "text/html",
      "properties": {
        "userName": "<<USERNAME>>/token",
        "password": "<<PASSWORD>>"
      },
      "expiryError": "ExpiredAuthenticationToken",
      "authError": [
        "InvalidAuthenticationToken",
```

```

    "AuthenticationFailed"
  ],
  "timeOutError": "Read timed out",
  "testname": "<test URL>",
  "maskInLogs": [
    "password",
    "username",
    "access_token",
    "refresh_token"
  ],
  "errorPath": "error.code",
  "maxRefreshTryCount": 5,
  "tokenResponsePath": "access_token",
  "tokenType": "Basic",
  "accessToken": "Basic <<TOKEN>>",
  "apiRateLimitConfig": {
    "retryAfterCalls": 100,
    "retryWaitSeconds": 60
  }
}
}
}

```

6. **Clear special characters:** The connector supports clearing special characters in API response using the `cleanUpTextContent` attribute.

Example:

JSON

```
{
  "authentications": {
    "accAuth": {
      "authType": "oauth2",
      "url": "<url>",
      "httpMethod": "POST",
      "httpParams": {
        "organization": "<specify organization>",
        "username": "aabcd",
        "password": "12345",
        "culture": "en-US"
      },
      "httpHeaders": {
        "Accept": "application/json"
      },
      "httpContentType": "application/x-www-form-urlencoded",
```

```

    "retryFailureStatusCode": [
      401
    ],
    "errorPath": "",
    "maxRefreshTryCount": 3,
    "cleanUpTextContent": true,
    "tokenResponsePath": "token",
    "tokenType": "",
    "accessToken": "<specify access token>"
  }
}
}

```

7. **Handle space in httpParams:** The connector replaces space in the `httpParams` attribute with %20 before sending the request to hit the connection API. From Release v23.5 onwards, a new content type named as `application/x-www-form-urlencoded-pws` is introduced.

Example:

JSON

```

{
  "authentications": {

```

```
"userAuth": {
  "authType": "oauth2",
  "url": "<URL>",
  "httpMethod": "POST",
  "httpParams": {
    "grant_type": "password",
    "scope": "<scope>",
    "client_secret": "<client secret>",
    "username": "<user name>",
    "client_ID": "<client ID>",
    "password": "<password>"
  },
  "httpContentType": "application/x-www-form-urlencoded-pws",
  "expiryError": "ExpiredAuthenticationToken",
  "authError": [
    "USER_AUTHENTICATION_FAILED",
    "PARTNER_AUTHENTICATION_FAILED",
    "AuthenticationFailed"
  ],
  "timeOutError": "Read timed out",
  "errorPath": "errorCode",
  "maxRefreshTryCount": 5,
```

```
    "tokenResponsePath": "access_token",
    "tokenType": "Bearer",
    "retryFailureStatusCode": [
      401
    ],
    "accessToken": "Bearer abc"
  }
}
```



The following table describes attributes in **ConnectionJSON**:


Attribute	Description
acctAuth	Use this attribute to specify the name of the connection.
authType	<p>Use this attribute to specify the type of authentication used for the connection.</p> <ul style="list-style-type: none">• For Basic authentication type, you must specify username and password in the properties attribute. Ensure that the API supports Basic authentication type and tokenType is selected as Basic.

Attribute	Description
	<ul style="list-style-type: none"> • For OAuth2 authentication type, you must specify url, httpMethod, httpParams, httpParams, httpHeaders, httpContentType attributes to regenerate a token. • For BasicWithAccessToken authentication type, you must specify a valid accessToken. • For BasicWithHmac authentication type, you must specify IKEY and SKEY in the properties attribute. <p>From Release v 23x onwards, if you are using Veracode authentication type, you must also specify Hmac as the authType configuration and veracode as the hmacType configuration.</p> <p>For cookies authentication type, you must specify apiKey in the properties attribute.</p> <ul style="list-style-type: none"> • For SignedHeaders authentication type, you must specify userId and keyFilePath in the properties attribute. • For JWT authentication type, you must specify jwtConfig, signedAlgorithm, key, and jwtExpiryDuration.
jwtConfig	<p>Use this attribute to specify the jwtHeader and jwtPayload details of the JWT token. Sample header and payload details passed in the token are:</p> <ul style="list-style-type: none"> • jwtHeader.alg: Specifies the type of algorithm used for signature or encryption.

Attribute	Description
	<ul style="list-style-type: none"> • <code>jwtHeader.typ</code>: Specifies the type of the token. • <code>jwtHeader.kid</code>: Specifies the key used to secure the JWS. • <code>jwtPayload.iss</code>: Specifies the issuer for the created and signed token. • <code>jwtPayload.sub</code>: Specifies the subject referred in the token. • <code>jwtPayload.aud</code>: Specifies the audience for the token. • <code>jwtPayload.scope</code>: Specifies the permission granted for the subject on the resource.
<code>signedAlgorithm</code>	Use this attribute to specify the algorithm used for signing the token. The connector supports HMAC, RSA and ECDSA algorithms.
<code>key</code>	Use this attribute to specify the secret key or certificate used for the authentication.
<code>jwtExpiryDuration</code>	Use this attribute to specify the duration for token expiry.
<code>username</code>	Use this attribute to specify the username used for authentication.

Attribute	Description
password	Use this attribute to specify the password used for authentication.
client_id client_secret	<p>After you register the connector with an application, you get a Client ID and Client Secret.</p> <p>The Client ID is considered public and is used to build login URLs. The Client Secret is confidential and private. Both, the Client ID and Client Secret are used for authenticating to the target application. The Client ID acts as a username and the Client Secret acts as a password.</p>
scope	Based on the permissions assigned for access to various objects or components in the target application, a scope is generated. Specify the scope, for which you want to assign the permissions.
url	Use this attribute to specify the target application URL. Leave this blank for the Basic authentication type.

Attribute	Description
	<div>  Info </div> <p>From Release v24.7 onwards, if you want to establish a connection with Mutual Transport Layer Security (mTLS) authentication, specify the server URL that is configured for mTLS.</p>
httpMethod	Use this attribute to specify the method to call the REST API such as GET, POST, PUT, DELETE and GETWITHBODY.
httpParams	<p>Use this attribute to specify additional attributes required to make a successful call.</p> <div>  Info </div> <p>From Release v24.7 onwards. you can update the batch and offset parameter values in the httpParams block. For more information, see Example 3 under step c in this section.</p>
httpHeaders	Use this attribute to specify header parameters such as Authorization, Accept required to make the REST API call.

Attribute	Description
	<div>  Info </div> <p>From Release v24.7 onwards, you can update the batch and offset parameter values in the httpHeaders block. For more information, see Example 3 under step c in this section.</p>
httpContentType	The function header sends the http JSON header to the browser to indicate the type of content it expects. Use this attribute to specify content type of request parameters during regeneration of a token, it can be JSON, text, html, XML or application XML.
expiryError	Use this attribute to specify the error response to be displayed for an expiry error.
authError	<p>Use this attribute to specify the error response to be displayed for an authentication error.</p> <p>For Basic authentication type, if the authentication fails, the connector retries to authenticate the connection.</p>
timeOutError	Use this attribute to specify the error response to be displayed for a connection timeout error.

Attribute	Description
accessToken	<p>Use this attribute to specify the access token. Initially, you can specify a dummy value, blank value or any valid token. This value gets populated after the token gets refreshed.</p> <p>For a Basic Authentication type, specify <code>accessToken</code> as Base64.</p>
authHeaderName	<p>For OAuth2 authentication type, the token is sent in the Authorization header which is specified in API headers as Authorization with the default value “\${access_token}”. But there are target applications that use a different header names instead of Authorization. In this scenario, specify the authHeaderName in ConnectionJSON.</p>
refreshType	<p>Use this attribute to save the latest refresh token for generating the token.</p>
refreshTokenResponsePath	<p>Use this attribute to specify the path to obtain the refresh token in the API response when action to refresh or generate a token is initiated.</p>
refreshToken	<p>Use this attribute to specify the valid refresh token to obtain the refreshed access or refresh token after the tokens are expired.</p>

Attribute	Description
refrshTokenAuthError	Use this attribute to specify the list of errors that can occur while refreshing the token. These errors will be received while calling the generate token API. If an error is encountered and it matches an error from this list, the system regenerates the refresh token for that authentication and stores it in the refreshToken attribute.
refreshTokenErrorPath	If the error returned from the token regeneration API call is in the form of JSON/HTML/XML, use this attribute to specify the path to look for the errors in the response. If an error is returned in headers, then this field is empty.
refreshTokenCall	Use this attribute to invoke another API (Generate Refresh Token) for generating a new refresh token and save it for subsequent API calls. To regenerate a token, specify the values for the following attributes: url, httpMethod, httpParams, httpHeaders, and httpContentType.
maskInLogs	Use this attribute to specify the list of sensitive attributes whose values must be masked in the logs. Specify the attribute names as comma separated values.
retryAfterCalls	Use this attribute to specify the number calls after which the API throws exceptions.

Attribute	Description
retryWaitSeconds	Use this attribute to specify the sleep time interval between the two blocks of calls, so that the dependent second call is processed after completing the calls in the first block.
cleanUpTextContent	Use this attribute to instruct the connector to map the attributes returned in the API response to IGA if the response contains special characters. When set to true, the connector clears the text that disrupts the traversing flow of the API response and successfully maps the attributes.
retryFailureStatusCode	Use this attribute to specify the codes in a list format if the authentication failure is returned as a set of http status codes in the response body. This instructs to perform system refresh or regenerate tokens. It stores new tokens in the accessToken attribute.
keyFile	<p>Use this attribute to specify the absolute path of the keystore which contains the public-private keypair. For example, /saviynt_shared/saviynt/ConnectorFiles/filename.pem.</p> <p>From Release v23.8, the REST connector uses the default path of key file that you upload under Admin > Settings > File Directory > Connector Files for creating a connection. Instead of the absolute path, specify the file name that contains the public-private keypair. For more</p>

Attribute	Description
	information about uploading connector files, see Configuring File Directories in the <i>Enterprise Identity Cloud Administration Guide</i> .
keyFilePassword	Use this attribute to specify the password of the keystore.
keyManagerAlgorithm	Use this attribute to specify the algorithm used to create the keystore.
keyStoreType	Use this attribute to specify the type of keystore.
sslAlgorithmName	Use this attribute to specify the algorithm used to create the SSL context using this keystore.

ImportAccountEntJSON

This parameter is used to map account and entitlement attributes of the target application to account and entitlement attributes of IGA for account import.

Common features

1. **Support for Multiple API Call:** Supports calling multiple dependent APIs for the same parent API call to import extended properties from the target application.

Example 1: Add multiple calls in the accountParams attribute. This example uses the `${accountName}` binding variable to iterate the accounts imported as part of call1. Set the `dependentCall` variable under `inputParams` to true when the call2 is dependent on call1.

JSON

```
{
  "accountParams": {
    "connection": "acctAuth",
    "createUsers": true,
    "adminName": "admin",
    "processingType": "SequentialAndIterative",
    "statusAndThresholdConfig": {
      "deleteLinks": false,
      "accountThresholdValue": 30,
      "correlateInactiveAccounts": false,
      "inactivateAccountsNotInFile": false,
      "deleteAccEntForActiveAccounts": true
    },
  },
}
```

```
"call": {
  "call1": {
    "callOrder": 0,
    "stageNumber": 0,
    "http": {
      "url": "https://api.*****.com/v1/enterprise/345992/users",
      "httpHeaders": {
        "Authorization": "${access_token}"
      },
      "httpContentType": "application/x-www-form-urlencoded",
      "httpMethod": "GET"
    },
    "listField": "users",
    "keyField": "accountID",
    "colsToPropsMap": {
      "accountID": "id~#~char",
      "name": "id~#~char"
    },
    "disableDeletedAccounts": true
  },
  "call2": {
    "callOrder": 1,
```

```
"stageNumber": 3,
"http": {
  "url": "https://api.*****.com/v1/user/${accountName}",
  "httpHeaders": {
    "Authorization": "${access_token}"
  },
  "contentType": "application/x-www-form-urlencoded",
  "httpMethod": "GET"
},
"inputParams": {
  "dependentCall": true
},
"listField": "",
"keyField": "accountID",
"nextApiKeyField": "accountID",
"colsToPropsMap": {
  "accountID": "id~#~char",
  "name": "username~#~char",
  "customproperty1": "emailId~#~char",
  "customproperty2": "lastName~#~char",
  "customproperty3": "company~#~char",
  "customproperty4": "timezone~#~char",
```

```

        "customproperty5": "defaultEndpoint~#~char",
        "customproperty6": "lastLogin~#~char"
    }
}
}
}
}

```

Example 2: Add multiple calls in the entitlementParams attribute. This example uses the `${entitlementID}` binding variable to iterate the entitlementIDs imported as part of call1. Set the `entitlementname` variable under `inputParams` to the name of entitlement type when the call2 is dependent on call1.

JSON

```

{
  "accountParams": {},
  "entitlementParams": {
    "connection": "acctAuth",
    "processingType": "SequentialAndIterative",
    "entTypes": {
      "zone": {
        "entTypeOrder": 0,
        "call": {

```

```
"call1": {
  "callOrder": 0,
  "stageNumber": 0,
  "http": {
    "url": "https://*****/Group",
    "httpHeaders": {
      "Authorization": "${access_token}",
      "Accept": "text/json"
    },
    "httpContentType": "application/json",
    "httpMethod": "GET"
  },
  "listField": "Rsp.Result",
  "keyField": "entitlementID",
  "colsToPropsMap": {
    "entitlementID": "Guid~#~char",
    "entitlement_value": "Guid~#~char"
  },
  "disableDeletedEntitlements": true
},
"call2": {
  "connection": "acctAuth",
```

```
"callOrder": 1,
"stageNumber": 3,
"http": {
  "url": "https://****/groupID/${entitlementID}",
  "httpHeaders": {
    "Authorization": "${access_token}",
    "Accept": "text/json"
  },
  "httpContentType": "application/json",
  "httpMethod": "GET"
},
"inputParams": {
  "entitlementname": "zone"
},
"listField": "",
"keyField": "entitlementID",
"colsToPropsMap": {
  "entitlementID": "Rsp.Result.Guid~#~char",
  "entitlement_value": "Rsp.Result.Guid~#~char",
  "customproperty1": "Rsp.Result.AccessPermissionLevel~#~char",
  "customproperty2": "Rsp.Result.Name~#~char",
  "customproperty3": "Rsp.Result.OwnerRole~#~char"
```

```
        },
        "disableDeletedEntitlements": true
      }
    }
  }
},
"acctEntParams": {
}
}
```

2. **Support for Pagination:** Provides the logic to handle the pagination of objects returned from the target application. Various types of pagination are supported to import all accounts and entitlements.

a. **NextUrl Type:**

Type 1: Uses response body to evaluate the next URL

Example 1: Uses the next page URL in response.

```
Pretty Raw Preview Visualize JSON ↕
77     "suspended": false,
78     "chat_only": false,
79     "default_group_id": null,
80     "report_csv": false,
81     "user_fields": {}
82   },
83 ],
84 "next_page": "https://saviynt572.zendesk.com/api/v2/users.json?page=2&per_page=2",
85 "previous_page": null,
86 "count": 7
87 }
```

Example 2: No pages to iterate

```
Pretty Raw Preview Visualize JSON ↕
301     "ticket_restriction": "requested",
302     "only_private_comments": false,
303     "restricted_agent": true,
304     "suspended": false,
305     "chat_only": false,
306     "default_group_id": null,
307     "report_csv": false,
308     "user_fields": {}
309   },
310 ],
311 "next_page": null,
312 "previous_page": null,
313 "count": 7
```

The logic to iterate into the next page using the URL available in the response body is as follows:

1. Analyze the complete response for the attribute containing the next page URL.
2. Use the attribute in the if-else condition as shown in the example below.

3. Use the value in the response to paginate into the next pages if the `next_page` attribute contains a value, else the pagination stops.

JSON

```
"pagination": {  
  "nextUrl": {  
    "nextUrlPath": "${response?.completeResponseMap?.next_page==null?null:response.completeResponseMap.next_page}"  
  }  
}
```



Note

Specify the `response.completeResponseMap` attribute before the traversing path pagination attribute in the response.

Type 2: Uses headers to evaluate the next URL

If the next URL is received as part of response headers analyze the links received in page one, page two and the last page.

<input checked="" type="checkbox"/>	User-Agent ①	PostmanRuntime/7.26.8	
<input checked="" type="checkbox"/>	Accept ①	*/*	
<input checked="" type="checkbox"/>	Accept-Encoding ①	gzip, deflate, br	
<input checked="" type="checkbox"/>	Connection ①	keep-alive	
	Key	Value	Description
Body Cookies (4) Headers (19) Test Results 200 OK 20.75 s 2.88 MB Save Response			
	X-Is-Logged-In ①	true	
	X-Transaction-ID ①	[REDACTED]	
	Link ①	<https://[REDACTED].com/api/now/table/sys_user?sysparm_limit=2000&sysparm_offset=0;rel="first",https://[REDACTED].com/api/now/table/sys_user?sysparm_limit=2000&sysparm_offset=2000;rel="next",https://[REDACTED].com/api/now/table/sys_user?sysparm_limit=2000&sysparm_offset=150000;rel="last">	
	X-Total-Count ①	150801	

Figure: Link in response header

Response of page one links:

```
<https://<URL>/api/now/table/sys_user?sysparm_limit=2000&sysparm_offset=0;rel="first",https://<URL>/api/now/table/sys_user?sysparm_limit=2000&sysparm_offset=2000;rel="next",https://<URL>/api/now/table/sys_user?sysparm_limit=2000&sysparm_offset=150000;rel="last">
```

Response of page two links:

```
<https://<URL>/api/now/table/sys_user?sysparm_limit=2000&sysparm_offset=0;rel="first",https://<URL>/api/now/table/sys_user?sysparm_limit=2000&sysparm_offset=0;rel="prev",https://<URL>/api/now/table/sys_user?sysparm_limit=2000&sysparm_offset=4000;rel="next",https://<URL>/api/now/table/sys_user?sysparm_limit=2000&sysparm_offset=150000;rel="last">
```

Response from the last call links:

```
<https://<URL>/api/now/table/sys_user?sysparm_limit=2000&sysparm_offset=0;rel="first",https://<URL>/api/now/table/sys_user?sysparm_limit=2000&sysparm_offset=148000;rel="prev",https://<URL>/api/now/table/sys_user?sysparm_limit=2000&sysparm_offset=150000;rel="last">
```

The logic to iterate into the next page using the URL available in the response headers is as follows:

1. Split the link based on comma and count the number of links in the response headers.

```
<https://<URL>/api/now/table/sys_user?sysparm_limit=2000&sysparm_offset=0;rel="first",--(0)>
```

```
<https://<URL>/api/now/table/sys_user?sysparm_limit=2000&sysparm_offset=2000;rel="next",--(1)>
```

```
<https://<URL>/api/now/table/sys_user?sysparm_limit=2000&sysparm_offset=150000;rel="last"--(2)>
```

2. Find the position of the next URL in the link. In the above link the position of next URL is 1. Similarly find the position of the next URL in page two and the last page.
3. While constructing the pagination block, utilize the if-else condition to check the size of the links in the headers and use the replace function to remove <, > and >;rel="next\" and replace it with null to get a valid URL as shown below.

```
https://<URL>/api/now/table/sys_user?sysparm_limit=2000&sysparm_offset=2000
```



Note

The first and last page might have three links as shown in the above header responses. To avoid the error in the last page, mention the function “contains('next')” in the condition to consider the link only if it contains the `nextUrl` attribute else it stops the pagination.

JSON

```
"pagination": {  
  "nextUrl": {  
    "nextUrlPath": "${headers?.Link?.split(',').size() == 4 ? headers.Link.split(',')[2].replace('<', ' ').replace('>', ' ') : ''}  
  }  
}
```

Type 3: Uses the pagination parameters in the URL to evaluate the next URL

Sample URL:

`https://<URL>/organization-api-service/v1/management-units?per_page=100&page=0`

Sample response body:

```

        "managementUnitParent": "managementUnitParent",
        "managementUnitParent": "managementUnitParent",
        "managementUnitParent": "managementUnitParent",
        "managementUnitParent": "managementUnitParent",
        "managementUnitParent": "managementUnitParent",
        "managementUnitParent": "managementUnitParent",
        "managementUnitParent": "managementUnitParent",
    ],
    "pagination": {
        "totalPages": 3426,
        "totalElements": 34257,
        "page": 0,
        "size": 10
    }
}

```

The logic to iterate into the next page using the page parameters in the response body is as follows:

1. Takes the size of the objectList from the response as the condition.
2. Uses the math function in JAVA to increment the pagenummer and the size of the page from the response received in the body if the size of the objectList i.e the number of elements in the response is more than 0.

JSON

```
"pagination": {  
  "nextUrl": {  
    "nextUrlPath": "${response?.objectList?.size()}>0?'https://examplesaviyntcom/organization-api-service/v
```

b. Page Type:

Example: When the totalCount Path is available in the response, specify totalCountPath as shown in the below configuration.

JSON

```
"pagination": {  
  "page": {  
    "pageSizeParam": "per_page",  
    "pageSize": 50,  
    "pageRecordCount": "completeResponseMap.meta.count",  
    "pageNumberParam": "page",  
    "totalCountPath": "completeResponseMap.meta.total_count",  
    "firstPageNumber": 1  
  }  
}
```

here,

- **pageSizeParam**: Specify the query parameter name in the http URL which specifies a maximum number of records sent in the response.
- **pageSize**: Specify the maximum number of records requested in each call used by pageSizeParam.
- **pageRecordCount**: Specify the field in the response that denotes an actual number of records returned in the page.
- **PageNumberParam**: Specify the page number that was currently returned.
- **totalCountPath**: Specify the value for total count of records in the target application.
- **FirstPageNumber**: The default page number returned by the target application starts with “1”. If the firstPageNumber starts with zero, it is set to “0”.

Example 2: When the totalCount Path is available in the header, specify totalCountPath as shown in the below configuration.

JSON

```
"pagination": {
  "page": {
    "pageSizeParam": "size",
    "pageSize": 25,
    "pageRecordCount": "25",
    "pageNumberParam": "page",
    "totalCountPath": "headers.x-total-count",
    "firstPageNumber": 0
  }
}
```

```
}  
}
```

Sample URL:

https://*****/rest/api/2/users?limit=100&page=1

Sample response:

```
{},  
{  
  "firstname": "vince",  
  "statuskey": "1",  
  "displayname": "V",  
  "userKey": 2464,  
  "customeproperty65": null,  
  "email": "vjn@instance.com",  
  "lastname": "jackson",  
  "username": "Vjackson"  
}  
],  
"count": 100,  
"hasMore": true,  
"limit": 100,  
"page": 1,  
"TotalCount": 9837  
}
```


The steps to construct the pagination snippet for the above response is as shown below:

1. Set `pageSizeParam` as the limit value.
2. Set **pageSize** as 1000.
3. Set the **pageRecordCount** as the count value.



Note

Add `completeResponseMap` to the traversing path of `pageRecordCount`.

4. Set **PageNumberParam** as the page value.
5. Set **totalCountPath** as the `TotalCount` value.



Note

Add `completeResponseMap` to the traversing path of `totalCountPath`.

Sample pagination snippet:

JSON

```
{
  "pagination": {
    "page": {
      "pageSizeParam": "limit",
      "pageSize": 1000,
      "pageRecordCount": "completeResponseMap.count",
      "pageNumberParam": "page",
      "totalCountPath": "completeResponseMap.TotalCount",
      "firstPageNumber": 1
    }
  }
}
```

c. **Offset Based:** Uses the following response attributes:

Example 1:

JSON

```
"pagination": {
  "offset": {
    "offsetParam": "offset",
    "batchParam": "limit",
```

```
    "batchSize": 50,  
    "totalCountPath": "5000"  
  }  
}
```

Example 2:

From Release v24.7 onwards, EIC supports pagination for offset parameters in the URL, httpParams and httpHeaders attribute values. This support is available when the user runs user import, account import, or access import by using the REST connector.

JSON

```
"pagination": {  
  "offset": {  
    "offsetParam": "offset",  
    "startOffsetIndex": 1,  
    "batchParam": "limit",  
    "batchSize": 50,  
    "paramOrder": "batchParam, offsetParam",  
    "paramPosition": "url,httpHeaders,httpParams",  
    "totalCountPath": 5000  
  }  
}
```

```
}  
}
```

here,

- **offsetParam**: Specifies field in the response that denotes offset parameter name to use as a query attribute.
- (Optional) **startOffsetIndex**: Specifies the index value for the specified offsetParam and is set above 0.
- **batchParam**: Specifies the query parameter name in the http URL that specifies maximum number of records sent in response.
- **batchSize**: Specifies the value used in batchParam that denotes the number of records to return in each API call.
- **url**: Specifies the URL of the API used to get total count as shown in Example 3.
- (Optional) **paramOrder**: Specify the order to add the batch and offset parameters.
- (Optional) **paramPosition**: Specify blocks which contain the offset parameters that need to be updated for pagination.
- **totalCountPath**: Specifies the total number of records in the the target application. If this value is not available, use integer value as an approximate value.

Specifies the API response that displays the totalCountPath, if an API call is used to get total count as shown in Example 3.

Example 3:

JSON

```
"pagination":
{
  "offset":{
    "offsetParam": "offset",
    "batchParam": "limit",
    "batchSize": 25,
    "totalCountPath": {
      "connection": "acctAuth",
      "http": {
        "url": <specify URL>,
        "httpHeaders": {
          "Authorization": "${access_token}"
        },
        "httpContentType": "application/x-www-form-urlencoded",
        "httpMethod": "GET"
      },
      "totalCountPath": "map1.totalCount"
    }
  }
}
```

Sample URL:

https://*****/api/v5/user?offset=0&max=100

Sample response:

```
{
  "msg": "Successful",
  "offset": "0",
  "Totalcount": 5000,
  "errorCode": "0",
  "results": [
    {
      "ID": 2464,
      "email": "Tjohn@identity.com",
      "firstname": "tim",
      "lastname": "john",
      "statuskey": "1"
    }
  ]
}
```

The steps to construct the pagination snippet for the above response is as shown below:

- a. Set **offsetParam** to the offset value.
- b. Set **batchParam** to the max value.
- c. Set **batchSize** to 100.

d. Set **totalCountPath** to the Totalcount value.

3. Processing Types

a. **acctToEntMapping** - Specify this value when acctEntMappings is defined in accountParams section. The other elements such as connection, http, listField, keyField, entKeyField, acctIdPath need not be added.

Sample Response:

JSON

```
{
  "users": [
    {
      "id": 6790354345885,
      "url": "https://saviynt5519.zendesk.com/api/v2/users/6790354345885.json",
      "name": "abc",
      "email": "abc@company.com",
      "created_at": "2022-09-15T08:51:24Z",
      "updated_at": "2022-10-21T06:34:17Z",
      "time_zone": "Asia/Kolkata",
      "iana_time_zone": "Asia/Kolkata",
      "phone": null,
      "shared_phone_number": null,
    }
  ]
}
```

```
    "photo": null,  
    "locale_id": 1,  
    "locale": "en-US",  
    "organization_id": 6790341912861,  
    "role": "admin",  
    "verified": true,  
    "external_id": null,  
    "tags": [],  
    "alias": null,  
    "active": true,  
    "shared": false,  
    "shared_agent": false,  
    "last_login_at": "2022-10-21T06:34:17Z",  
    "two_factor_auth_enabled": null,  
    "signature": null,  
    "details": null,  
    "notes": null,  
    "role_type": 4  
  },  
  "next_page": null,  
  "previous_page": null,
```



```
"count": 1
}
```

Sample `ImportAccountEntJSON`:

JSON

```
{
  "accountParams": {
    "connection": "acctAuth",
    "processingType": "SequentialAndIterative",
    "statusAndThresholdConfig": {},
    "call": {
      "call1": {
        "callOrder": 0,
        "stageNumber": 0,
        "http": {
          "url": "@HOSTNAMEa/api/v2/users.json?role[]=admin&role[]=agent",
          "httpHeaders": {
            "Authorization": "${access_token}",
            "Accept": "application/json"
          },
          "httpContentType": "application/json",

```

```

        "httpMethod": "GET"
    },
    "listField": "users",
    "keyField": "accountID",
    "colsToPropsMap": {
        "accountID": "id~#~char",
        "name": "email~#~char",
        "customproperty31": "STORE#ACC#ENT#MAPPINGINFO~#~char"
    }
},
"acctEntMappings": {
    "Role": {
        "listPath": "",
        "idPath": "role",
        "keyField": "entitlementID"
    }
},
"entitlementParams": {
    "connection": "acctAuth",
    "processingType": "SequentialAndIterative",

```

```
"entTypes": {
  "Role": {
    "entTypeOrder": 1,
    "call": {
      "call1": {
        "callOrder": 0,
        "stageNumber": 0,
        "http": {
          "url": "aHOSTNAMEa/api/v2/custom_roles.json",
          "httpHeaders": {
            "Authorization": "${access_token}",
            "Accept": "application/json"
          },
          "httpContentType": "application/json",
          "httpMethod": "GET"
        },
        "listField": "custom_roles",
        "keyField": "entitlementID",
        "colsToPropsMap": {
          "entitlementID": "id~#~char",
          "entitlement_value": "name~#~char"
        }
      },
```

```

        "disableDeletedEntitlements": true
      }
    }
  },
  "acctEntParams": {
    "processingType": "acctToEntMapping"
  }
}

```



Note The `customproperty31` is a mandatory field that stores the entitlement details in the accounts table.

2. **httpEntToAcct** - Specify this value when you want to call an http API for each entitlement to get the list of accounts associated with it. Specify `${id}` to iterate API calls for each entitlement value in http url field attribute.

Sample Response:

JSON

```

{
  "group_memberships": [
    {

```

```
    "url": "https://xxx.zendesk.com/api/v2/group_memberships/9829743537297.json",
    "id": 9829743537297,
    "user_id": 9829735960593,
    "default": true,
    "created_at": "2022-10-13T08:50:03Z",
    "updated_at": "2022-10-13T08:50:03Z"
  }
],
"next_page": null,
"previous_page": null,
"count": 1
}
```

Sample **ImportAccountEntJSON**:

JSON

```
{
  "accountParams": {
    "connection": "acctAuth",
    "processingType": "SequentialAndIterative",
    "statusAndThresholdConfig": {
      "statusColumn": "customproperty11",
```

```
"activeStatus": [
  "false"
],
"deleteLinks": true,
"accountThresholdValue": 10,
"correlateInactiveAccounts": false,
"inactivateAccountsNotInFile": true,
"deleteAccEntForActiveAccounts": true
},
"call": {
  "call1": {
    "callOrder": 0,
    "stageNumber": 0,
    "http": {
      "url": "@HOSTNAME@/api/v2/users.json?role[]=admin&role[]=agent",
      "httpHeaders": {
        "Authorization": "${access_token}",
        "Accept": "application/json"
      },
      "httpContentType": "application/json",
      "httpMethod": "GET"
    }
  },
```

```
"listField": "users",
"keyField": "accountID",
"statusConfig": {
  "active": "true",
  "inactive": "false"
},
"colsToPropsMap": {
  "accountID": "id~#~char",
  "name": "email~#~char",
  "displayName": "name~#~char",
  "customproperty2": "email~#~char",
  "customproperty3": "created_at~#~char",
  "customproperty4": "updated_at~#~char",
  "customproperty5": "role~#~char",
  "status": "active~#~char",
  "customproperty6": "last_login_at~#~char",
  "customproperty7": "custom_role_id~#~char",
  "customproperty8": "default_group_id~#~char",
  "customproperty9": "created_at~#~char",
  "customproperty10": "updated_at~#~char",
  "customproperty11": "suspended~#~char"
}
```

```
    }  
  }  
},  
"entitlementParams": {  
  "connection": "acctAuth",  
  "processingType": "SequentialAndIterative",  
  "entTypes": {  
    "Group": {  
      "entTypeOrder": 0,  
      "entTypeLabels": {  
        "customproperty1": "Deleted",  
        "customproperty2": "CreatedAt",  
        "customproperty3": "UpdatedAt"  
      },  
      "call": {  
        "call1": {  
          "callOrder": 0,  
          "stageNumber": 0,  
          "http": {  
            "url": "aHOSTNAMEa/api/v2/groups",  
            "httpHeaders": {  
              "Authorization": "${access_token}",
```



```

        "Accept": "application/json"
    },
    "httpContentType": "application/json",
    "httpMethod": "GET"
},
"listField": "groups",
"keyField": "entitlementID",
"colsToPropsMap": {
    "entitlementID": "id~#~char",
    "entitlement_value": "name~#~char",
    "customproperty1": "deleted~#~char",
    "customproperty2": "created_at~#~char",
    "customproperty3": "updated_at~#~char"
},
"pagination": {
    "nextUrl": {
        "nextUrlPath": "${response?.completeResponseMap?.next_page==null?null:response.completeResponse
    }
},
"disableDeletedEntitlements": true
}
}

```

```

    }
  }
},
"acctEntParams": {
  "connection": "acctAuth",
  "entTypes": {
    "Group": {
      "call": {
        "call1": {
          "callOrder": 0,
          "stageNumber": 0,
          "processingType": "httpEntToAcct",
          "http": {
            "httpHeaders": {
              "Authorization": "${access_token}"
            },
            "url": "aHOSTNAMEa/api/v2/groups/${id}/memberships.json",
            "httpContentType": "application/x-www-form-urlencoded",
            "httpMethod": "GET"
          },
          "listField": "group_memberships",
          "entKeyField": "entitlementID",

```

```

        "acctIdPath": "user_id",
        "acctKeyField": "accountID"
    }
}
}
}
}
}
}
}

```



Note The `acctEntMappingInfoColumnFromEnt` is a mandatory field to store account information.

3. **httpAcctToEnt** - Specify this value when there is a need to make an API call for each account to get mapping with all the entitlements of a particular entitlement type. Specify `${id}` to iterate API calls for each account in http url field attribute.

Sample Response:

JSON

```

{
  "memberships": [
    {
      "id": 9829743537297,
      "group_id": 9829727140881,

```

```
    "default": true,  
    "created_at": "2022-10-13T08:50:03Z",  
    "updated_at": "2022-10-13T08:50:03Z"  
  }  
],  
"next_page": null,  
"previous_page": null,  
"count": 1  
}
```

Sample **ImportAccountEnt** JSON:

JSON

```
{  
  "accountParams": {  
    "connection": "acctAuth",  
    "processingType": "SequentialAndIterative",  
    "statusAndThresholdConfig": {  
      "statusColumn": "customproperty11",  
      "activeStatus": [  
        "false"  
      ],  
    },  
  },  
}
```

```
    "deleteLinks": true,
    "accountThresholdValue": 10,
    "correlateInactiveAccounts": false,
    "inactivateAccountsNotInFile": true,
    "deleteAccEntForActiveAccounts": true
  },
  "call": {
    "call1": {
      "callOrder": 0,
      "stageNumber": 0,
      "http": {
        "url": "aHOSTNAMEa/api/v2/users.json?role[]=admin&role[]=agent",
        "httpHeaders": {
          "Authorization": "${access_token}",
          "Accept": "application/json"
        },
        "httpContentType": "application/json",
        "httpMethod": "GET"
      },
      "listField": "users",
      "keyField": "accountID",
      "statusConfig": {
```

```
    "active": "true",
    "inactive": "false"
  },
  "colsToPropsMap": {
    "accountID": "id~#~char",
    "name": "email~#~char",
    "displayName": "name~#~char",
    "customproperty2": "email~#~char",
    "customproperty3": "created_at~#~char",
    "customproperty4": "updated_at~#~char",
    "customproperty5": "role~#~char",
    "status": "active~#~char",
    "customproperty10": "updated_at~#~char",
    "customproperty11": "suspended~#~char"
  }
}
},
"entitlementParams": {
  "connection": "acctAuth",
  "processingType": "SequentialAndIterative",
  "entTypes": {
```

```
"Group": {
  "entTypeOrder": 0,
  "entTypeLabels": {
    "customproperty1": "Deleted",
    "customproperty2": "CreatedAt",
    "customproperty3": "UpdatedAt"
  },
  "call": {
    "call1": {
      "callOrder": 0,
      "stageNumber": 0,
      "http": {
        "url": "aHOSTNAMEa/api/v2/groups",
        "httpHeaders": {
          "Authorization": "${access_token}",
          "Accept": "application/json"
        },
        "httpContentType": "application/json",
        "httpMethod": "GET"
      },
      "listField": "groups",
      "keyField": "entitlementID",
```

```

        "colsToPropsMap": {
            "entitlementID": "id~#~char",
            "entitlement_value": "name~#~char",
            "customproperty1": "deleted~#~char",
            "customproperty2": "created_at~#~char",
            "customproperty3": "updated_at~#~char"
        },
        "disableDeletedEntitlements": true
    }
}
}
},
"acctEntParams": {
    "connection": "acctAuth",
    "entTypes": {
        "Group": {
            "call": {
                "call1": {
                    "callOrder": 0,
                    "stageNumber": 0,
                    "processingType": "httpAcctToEnt",

```



```

    "http": {
      "httpHeaders": {
        "Authorization": "${access_token}"
      },
      "url": "aHOSTNAMEa/unifiedlogin/user/${id}/product/",
      "httpMethod": "GET"
    },
    "listField": "memberships",
    "entIdPath": "group_id",
    "entKeyField": "entitlementID",
    "acctKeyField": "accountID"
  }
}
}
}
}
}
}
}
}

```

4. **entToAcctMapping** - Specify this value when acctEntMapping tag is defined inside entitlementParams section. Rest of the elements i.e. EntTypes should not be mentioned. Specifying acctEntMappingInfoColumnFromEnt in colsToPropsMap is mandatory.

Sample Response:

JSON

```
{
  "groups": [
    {
      "url": "https://xxx.zendesk.com/api/v2/groups/9829727140881.json",
      "is_public": true,
      "name": "Support",
      "description": "",
      "default": true,
      "deleted": false,
      "created_at": "2022-10-13T08:50:03Z",
      "updated_at": "2022-10-13T08:50:03Z",
      "account_id": 692634084011
    }
  ],
  "next_page": null,
  "previous_page": null,
  "count": 1
}
```

Sample ImportAccountEntJSON:

JSON

```
{
  "accountParams": {
    "connection": "userAuth",
    "processingType": "SequentialAndIterative",
    "call": {
      "call1": {
        "callOrder": 0,
        "stageNumber": 0,
        "http": {
          "url": "%%BASEURL%%/admin/v1/Users",
          "httpHeaders": {
            "Authorization": "${access_token}"
          },
          "httpContentType": "application/json",
          "httpMethod": "GET"
        },
        "listField": "Resources",
        "keyField": "accountID",
        "statusConfig": {
```

```
    "active": "true",
    "inactive": "false"
  },
  "colsToPropsMap": {
    "accountID": "id~#~char",
    "name": "userName~#~char",
    "displayname": "displayName~#~char",
    "status": "active~#~char"
  }
}
},
"entitlementParams": {
  "processingType": "SequentialAndIterative",
  "entTypes": {
    "Group": {
      "entTypeOrder": 0,
      "call": {
        "call1": {
          "connection": "userAuth",
          "callOrder": 0,
          "stageNumber": 0,
```

```

    "http": {
      "httpHeaders": {
        "Authorization": "${access_token}"
      },
      "url": "@@BASEURL@@/admin/v1/Groups?attributes=members,displayName",
      "httpContentType": "application/json",
      "httpMethod": "GET"
    },
    "listField": "Resources",
    "keyField": "entitlementID",
    "colsToPropsMap": {
      "entitlementID": "id~#~char",
      "entitlement_value": "displayName~#~char",
      "acctEntMappingInfoColumnFromEnt": "STORE#ACC#ENT#MAPPINGINFO~#~char"
    }
  },
  "acctEntMappings": {
    "listField": "",
    "idPath": "account_id",
    "keyField": "accountID",
    "importAsAccount": false
  }
}

```

```
    }
  }
},
"acctEntParams": {
  "processingType": "entToAcctMapping"
}
}
```

5. **http** - Specify this value when all the account entitlement mappings (with all entitlement types) are returned with a single http API call.

Sample Response:

JSON

```
{
  "Memberships": [
    {
      "Account_id": 692634084011,
      "Group_id": 9829727140881,
      "is_public": true,
      "name": "Support",
    }
  ]
}
```

```
    "description": "",
    "default": true,
    "deleted": false,
    "created_at": "2022-10-13T08:50:03Z",
    "updated_at": "2022-10-13T08:50:03Z"
  }
],
"next_page": null,
"previous_page": null,
"count": 1
}
```

Sample ImportAccountEntJSON:

JSON

```
{
  "accountParams": {
    "connection": "userAuth",
    "processingType": "SequentialAndIterative",
    "call": {
      "call1": {
        "callOrder": 0,
```

```
"stageNumber": 0,
"http": {
  "url": "âBASEURLâ/admin/v1/Users",
  "httpHeaders": {
    "Authorization": "${access_token}"
  },
  "httpContentType": "application/json",
  "httpMethod": "GET"
},
"listField": "Resources",
"keyField": "accountID",
"statusConfig": {
  "active": "true",
  "inactive": "false"
},
"colsToPropsMap": {
  "accountID": "id~#~char",
  "name": "userName~#~char",
  "displayname": "displayName~#~char",
  "status": "active~#~char"
}
}
```



```

    }
  },
  "entitlementParams": {
    "processingType": "SequentialAndIterative",
    "entTypes": {
      "Group": {
        "entTypeOrder": 0,
        "call": {
          "call1": {
            "connection": "userAuth",
            "callOrder": 0,
            "stageNumber": 0,
            "http": {
              "httpHeaders": {
                "Authorization": "${access_token}"
              },
              "url": "%%BASEURL%%/admin/v1/Groups?attributes=members,displayName",
              "httpContentType": "application/json",
              "httpMethod": "GET"
            },
          },
          "listField": "Resources",
          "keyField": "entitlementID",

```

```

        "colsToPropsMap": {
            "entitlementID": "id~#~char",
            "entitlement_value": "displayName~#~char",
            "acctEntMappingInfoColumnFromEnt": "STORE#ACC#ENT#MAPPINGINFO~#~char"
        }
    },
    "acctEntMappings": {
        "listField": "",
        "idPath": "account_id",
        "keyField": "accountID",
        "importAsAccount": false
    }
},
"acctEntParams": {
    "entTypes": {
        "Group": {
            "call": {
                "call1": {
                    "processingType": "http",

```

```

    "connection": "userAuth",
    "callOrder": 0,
    "stageNumber": 0,
    "http": {
      "httpHeaders": {
        "Authorization": "${access_token}"
      },
      "url": "a@BASEURLa@api/now/table/sys_user_has_role",
      "httpContentType": "application/json",
      "httpMethod": "GET"
    },
    "listField": "Memberships",
    "acctIdPath": "Account_id",
    "entIdPath": "Group_id",
    "entKeyField": "entitlementID",
    "acctKeyField": "accountID"
  }
}
}
}
}
}
}
}
}
}
}

```

6. Support for configuring success responses and failure responses for account and access imports in the `statusCode` attribute.

Sample `ImportAccountEntJSON`:

JSON

```
{
  "accountParams": {
    "connection": "acctAuth",
    "processingType": "SequentialAndIterative",
    "successResponses": {
      "statusCode": [
        200,
        201,
        202,
        203,
        204,
        205
      ]
    },
    "unsuccessResponses": null,
    "doNotChangeIfFailed": true,
    "call": {
```

```
"call1": {
  "callOrder": 0,
  "stageNumber": 0,
  "showJobHistory": true,
  "http": {
    "url": "URL",
    "contentType": "application/json",
    "httpMethod": "GET",
    "httpHeaders": {
      "Accept": "application/json"
    }
  },
  "listField": "records",
  "keyField": "accountID",
  "colsToPropsMap": {
    "accountID": "Id~#~char",
    "name": "Name~#~char",
    "displayName": "Email~#~char",
    "status": "IsActive~#~char",
    "customproperty31": "STORE#ACC#ENT#MAPPINGINFO~#~char"
  },
  "pagination": {
```

```

    "offset": {
      "offsetParam": "startIndex",
      "batchParam": "limit",
      "batchSize": 1000,
      "totalCountPath": "completeResponseMap.totalResults"
    }
  },
  "statusConfig": {
    "active": "true",
    "inactive": "false"
  },
  "disableDeletedAccounts": true
}
},
"acctEntMappings": {
  "Groups": {
    "importAsEntitlement": true,
    "listPath": "groups",
    "idPath": "id",
    "keyField": "entitlementID",
    "colsToPropsMap": {
      "entitlement_value": "name~#~char",

```

```

        "entitlementID": "id~#~char",
        "displayname": "displayName~#~char",
        "status": "IsActive~#~char"
    },
    "statusConfig": {
        "active": "true",
        "inactive": "false"
    },
    "entOwnerMappings": {
        "listPath": "groups",
        "idPath": "owner",
        "keyField": "email"
    }
}
},
"entitlementParams": {
    "connection": "acctAuth",
    "processingType": "SequentialAndIterative",
    "successResponses": {
        "statusCode": [
            200,

```

```
    201,  
    202,  
    203,  
    204,  
    205  
  ]  
},  
"unsuccessResponses": null,  
"doNotChangeIfFailed": true,  
"entTypes": {  
  "Groups": {  
    "entTypeOrder": 0  
  },  
  "Roles": {  
    "entTypeOrder": 0,  
    "call": {  
      "call1": {  
        "connection": "acctAuth",  
        "callOrder": 0,  
        "stageNumber": 0,  
        "showJobHistory": true,  
        "http": {
```



```
    "url": "URL",
    "contentType": "application/json",
    "method": "GET",
    "headers": {
      "Accept": "application/json"
    }
  },
  "statusConfig": {
    "active": "true",
    "inactive": "false"
  },
  "listField": "records",
  "keyField": "entitlementID",
  "colsToPropsMap": {
    "entitlement_value": "RoleName~~char",
    "entitlementID": "Id~~char",
    "displayname": "RoleName~~char",
    "status": "IsActive~~char",
    "acctEntMappingInfoColumnFromEnt": "STORE#ACC#ENT#MAPPINGINFO~~char"
  },
  "pagination": {
    "offset": {
```

```

        "offsetParam": "startIndex",
        "batchParam": "limit",
        "batchSize": 1000,
        "totalCountPath": "completeResponseMap.totalResults"
    }
},
    "disableDeletedEntitlements": true
}
},
    "entOwnerMappings": {
        "listField": "owners",
        "idPath": "Email",
        "keyField": "email"
    }
}
},
    "acctEntParams": {
        "connection": "acctAuth",
        "successResponses": {
            "statusCode": [
                200,

```

```

        201,
        202,
        203,
        204,
        205
    ]
},
"unsuccessResponses": null,
"entTypes": {
    "Groups": {
        "call": {
            "call1": {
                "processingType": "acctToEntMapping",
                "ownerKeyField": "email"
            }
        }
    }
},
"Roles": {
    "call": {
        "call1": {
            "processingType": "entToAcctMapping",
            "ownerKeyField": "email"
        }
    }
}

```

```

    }
  }
},
"Organization": {
  "call": {
    "call1": {
      "processingType": "http",
      "acctKeyField": "accountID",
      "connection": "acctAuth",
      "showJobHistory": true,
      "callOrder": 0,
      "stageNumber": 0,
      "http": {
        "url": "URL",
        "httpContentType": "application/json",
        "httpMethod": "GET",
        "httpHeaders": {
          "Accept": "application/json"
        }
      }
    },
    "listField": "records",
    "acctIdPath": "Id",

```

```

        "entIdPath": "groupId",
        "ownerIdPath": "owner",
        "ownerKeyField": "email",
        "entKeyField": "entitlementID",
        "pagination": {
            "offset": {
                "offsetParam": "startIndex",
                "batchParam": "limit",
                "batchSize": 1000,
                "totalCountPath": "completeResponseMap.totalResults"
            }
        }
    },
    "entOwnerParams": {
        "connection": "acctAuth",
        "successResponses": {
            "statusCode": [
                200,

```

```
    201,  
    202,  
    203,  
    204,  
    205  
  ]  
},  
"unsuccessResponses": null,  
"doNotChangeIfFailed": true,  
"entTypes": {  
  "Safes": {  
    "call": {  
      "call1": {  
        "processingType": "httpOwner",  
        "connection": "acctAuth",  
        "showJobHistory": true,  
        "callOrder": 0,  
        "stageNumber": 0,  
        "http": {  
          "url": "URL",  
          "httpContentType": "application/json",  
          "httpMethod": "GET",
```

```
    "httpHeaders": {
      "Accept": "application/json"
    },
    "listField": "records",
    "entIdPath": "Id",
    "ownerIdPath": "Owner",
    "ownerKeyField": "email",
    "entKeyField": "entitlementID",
    "pagination": {
      "offset": {
        "offsetParam": "startIndex",
        "batchParam": "limit",
        "batchSize": 1000,
        "totalCountPath": "completeResponseMap.totalResults"
      }
    }
  }
}
```

```
}  
}
```

7. Supports binding the `ConnectionJSON` call to `ImportAccountEntJSON` using the binding variable in the `url` attribute.

Sample `ImportAccountEntJSON`:

JSON

```
{  
  "accountParams": {  
    "connection": "acctAuth",  
    "processingType": "SequentialAndIterative",  
    "call": {  
      "call1": {  
        "callOrder": 0,  
        "stageNumber": 0,  
        "http": {  
          "url": "${connection.testname}",  
          "httpHeaders": {  
            "Authorization": "${access_token}",  
            "Accept": "application/json"  
          }  
        },  
      },  
    },  
  },  
}
```



```
    "httpContentType": "application/json",
    "httpMethod": "GET"
  },
  "listField": "users",
  "keyField": "accountID",
  "statusConfig": {
    "active": "true",
    "inactive": "false"
  },
  "colsToPropsMap": {
    "accountID": "id~#~char",
    "name": "email~#~char",
    "displayName": "name~#~char",
    "customproperty2": "email~#~char",
    "customproperty3": "created_at~#~char",
    "customproperty4": "updated_at~#~char",
    "customproperty5": "role~#~char",
    "status": "active~#~char",
    "customproperty6": "last_login_at~#~char",
    "customproperty7": "custom_role_id~#~char",
    "customproperty8": "default_group_id~#~char",
    "customproperty9": "created_at~#~char",
```

```

        "customproperty10": "updated_at~#~char",
        "customproperty11": "suspended~#~char",
        "customproperty31": "STORE#ACC#ENT#MAPPINGINFO~#~char"
    },
    "pagination": {
        "nextUrl": {
            "nextUrlPath": "${response.completeResponseMap.next_page}"
        }
    }
},
"acctEntMappings": {
    "Role": {
        "listPath": "",
        "idPath": "custom_role_id",
        "keyField": "entitlementID"
    }
},
"entitlementParams": {
    "connection": "acctAuth",
    "processingType": "SequentialAndIterative",

```

```
"entTypes": {
  "Group": {
    "entTypeOrder": 0,
    "entTypeLabels": {
      "customproperty1": "Deleted",
      "customproperty2": "CreatedAt",
      "customproperty3": "UpdatedAt"
    },
    "call": {
      "call1": {
        "callOrder": 0,
        "stageNumber": 0,
        "http": {
          "url": "https://<URL>/api/v2/groups",
          "httpHeaders": {
            "Authorization": "${access_token}",
            "Accept": "application/json"
          },
          "httpContentType": "application/json",
          "httpMethod": "GET"
        },
        "listField": "groups",
```

```

    "keyField": "entitlementID",
    "colsToPropsMap": {
        "entitlementID": "id~~char",
        "entitlement_value": "name~~char",
        "customproperty1": "deleted~~char",
        "customproperty2": "created_at~~char",
        "customproperty3": "updated_at~~char"
    },
    "pagination": {
        "nextUrl": {
            "nextUrlPath": "${response.completeResponseMap.next_page}"
        }
    },
    "disableDeletedEntitlements": true
}
},
"Role": {
    "entTypeOrder": 1,
    "entTypeLabels": {
        "customproperty1": "Description",
        "customproperty2": "CreatedAt",

```

```
    "customproperty3": "UpdatedAt"
  },
  "call": {
    "call1": {
      "callOrder": 0,
      "stageNumber": 0,
      "http": {
        "url": "https://<URL>/api/v2/custom_roles.json",
        "httpHeaders": {
          "Authorization": "${access_token}",
          "Accept": "application/json"
        },
        "httpContentType": "application/json",
        "httpMethod": "GET"
      },
      "listField": "custom_roles",
      "keyField": "entitlementID",
      "colsToPropsMap": {
        "entitlementID": "id~#~char",
        "entitlement_value": "name~#~char",
        "customproperty1": "description~#~char",
        "customproperty2": "created_at~#~char",
```

```

        "customproperty3": "updated_at~#~char"
    },
    "pagination": {
        "nextUrl": {
            "nextUrlPath": "${response.completeResponseMap.next_page}"
        }
    },
    "disableDeletedEntitlements": true
}
}
}
},
"acctEntParams": {
    "connection": "acctAuth",
    "entTypes": {
        "Group": {
            "call": {
                "call1": {
                    "callOrder": 0,
                    "stageNumber": 0,
                    "processingType": "httpEntToAcct",

```

```

    "http": {
      "httpHeaders": {
        "Authorization": "${access_token}"
      },
      "url": "https://<URL>/api/v2/groups/${id}/memberships.json",
      "httpContentType": "application/x-www-form-urlencoded",
      "httpMethod": "GET"
    },
    "listField": "group_memberships",
    "entKeyField": "entitlementID",
    "acctIdPath": "user_id",
    "acctKeyField": "accountID",
    "pagination": {
      "nextUrl": {
        "nextUrlPath": "${response.completeResponseMap.next_page}"
      }
    }
  },
  "Role": {
    "call": {

```

```

        "call1": {
            "callOrder": 0,
            "stageNumber": 0,
            "processingType": "acctToEntMapping"
        }
    }
}
}
}
}

```

- Support for access import till various levels of the entitlement hierarchy using `addDependentTask` and `removeDependentEntTask` variables in the `entMappingParams` attribute with the values to be set as either True or False for ent1 to ent2 mapping and not vice versa.

Sample `ImportAccountEntJSON`:

JSON

```

{
    "accountParams": {
        "connection": "acctAuth",
        "createUsers": true,
    }
}

```



```
"adminName": "admin",
"processingType": "SequentialAndIterative",
"call": {
  "call1": {
    "callOrder": 0,
    "http": {
      "url": "<https://<url>/_apis/graph/users?subjectTypes=aad&api-version=5.0-preview.1",
      "httpHeaders": {
        "Authorization": "${access_token}",
        "Accept": "application/json"
      },
      "httpContentType": "application/json",
      "httpMethod": "GET"
    },
    "listField": "value",
    "keyField": "accountID",
    "colsToPropsMap": {
      "accountID": "descriptor~#~char",
      "name": "principalName~#~char",
      "displayName": "displayName~#~char",
      "customproperty2": "mailAddress~#~char"
    },

```

```

        "makeProcessingStatus": true,
        "disableDeletedAccounts": true
    }
}
},
"entitlementParams": {
    "connection": "acctAuth",
    "processingType": "SequentialAndIterative",
    "entTypes": {
        "Project": {
            "entTypeOrder": 0,
            "call": {
                "call1": {
                    "callOrder": 0,
                    "http": {
                        "httpHeaders": {
                            "Authorization": "${access_token}"
                        },
                        "url": "https://<url>/_apis/projects?api-version=5.0-preview.1",
                        "httpContentType": "application/x-www-form-urlencoded",
                        "httpMethod": "GET"
                    },

```

```

        "listField": "value",
        "keyField": "entitlementID",
        "colsToPropsMap": {
            "entitlementID": "id~#~char",
            "entitlement_value": "name~#~char"
        },
        "makeProcessingStatus": true,
        "disableDeletedEntitlements": true
    }
},
"Team": {
    "entTypeOrder": 1,
    "call": {
        "call1": {
            "callOrder": 2,
            "stageNumber": 3,
            "inputParams": {
                "entitlementname": "Project"
            },
            "http": {
                "url": "https://<url/_apis/projects/${entitlementID}/Teams?api-version=5.0-preview.1",

```

```

    "httpHeaders": {
      "Authorization": "${access_token}"
    },
    "httpContentType": "application/x-www-form-urlencoded",
    "httpMethod": "GET"
  },
  "listField": "value",
  "keyField": "entitlementID",
  "colsToPropsMap": {
    "entitlementID": "id~~char",
    "entitlement_value": "name~~char",
    "description": "description~~char",
    "customproperty1": "url~~char",
    "customproperty6": "projectName~~char",
    "customproperty7": "projectId~~char"
  }
}
},
"Group": {
  "entTypeOrder": 2,
  "entTypeLabels": {

```

```
    "customproperty1": "Origin ID",
    "customproperty2": "Display Name",
    "customproperty3": "Descriptor ID"
  },
  "call": {
    "call1": {
      "callOrder": 0,
      "http": {
        "httpHeaders": {
          "Authorization": "${access_token}"
        },
        "url": "<https://<url>/_apis/graph/groups?api-version=5.0-preview.1",
        "httpContentType": "application/x-www-form-urlencoded",
        "httpMethod": "GET"
      },
      "listField": "value",
      "keyField": "entitlementID",
      "colsToPropsMap": {
        "entitlementID": "descriptor~#~char",
        "entitlement_value": "displayName~#~char",
        "description": "description~#~char",
        "customproperty1": "originId~#~char",
```

```

        "customproperty2": "displayName~#~char",
        "customproperty3": "descriptor~#~char"
    },
    "makeProcessingStatus": false,
    "disableDeletedEntitlements": true
}
}
}
},
"acctEntParams": {
    "processingType": "httpEntToAcct",
    "entTypes": {
        "Group": {
            "call": {
                "call1": {
                    "connection": "acctAuth",
                    "acctKeyField": "accountID",
                    "callOrder": 0,
                    "stageNumber": 0,
                    "http": {
                        "httpHeaders": {

```

```

        "Authorization": "${access_token}"
    },
    "url": "https://<url/_apis/Graph/Memberships/${id}?direction=Down&api-version=5.0-preview.1",
    "contentType": "application/x-www-form-urlencoded",
    "httpMethod": "GET"
},
"listField": "value",
"entKeyField": "entitlementID",
"acctIdPath": "memberDescriptor"
}
}
},
"Team": {
    "call": {
        "call1": {
            "connection": "acctAuth",
            "acctKeyField": "accountID",
            "callOrder": 0,
            "stageNumber": 0,
            "http": {
                "httpHeaders": {
                    "Authorization": "${access_token}"

```

```

        },
        "url": "https://url/_apis/Graph/Memberships/${id}?direction=Down&api-version=5.0-preview.1",
        "httpContentType": "application/x-www-form-urlencoded",
        "httpMethod": "GET"
    },
    "listField": "value",
    "entKeyField": "entitlementID",
    "acctIdPath": "memberDescriptor"
}
}
}
},
"entMappingParams": {
    "processingType": "SequentialAndIterative",
    "entTypes": {
        "Project": {
            "ent1KeyField": "entitlementID",
            "call": {
                "call1": {
                    "connection": "acctAuth",
                    "callOrder": 0,

```



```

    "stageNumber": 0,
    "http": {
      "httpHeaders": {
        "Authorization": "${access_token}"
      },
      "url": "<https://<url>/_apis/teams?api-version=5.0-preview.2",
      "httpContentType": "application/json",
      "httpMethod": "GET"
    },
    "listField": "value",
    "ent1IdPath": "projectId",
    "ent2IdPath": "id",
    "ent2KeyField": "entitlementID",
    "targetEntType": "Team",
    "addDependentTask": true,
    "removeDependentEntTask": true,
    "mappingTypes": [
      "ENTMAP"
    ]
  }
}

```

```
}  
}  
}
```

9. In addition to supporting pagination through URL, the connector supports pagination through Http Body and Http Headers using the `PageWithHttpParamsAndHeaders` variable.

Sample `ImportAccountEntJSON`:

JSON

```
{  
  "showLogs": false,  
  "accountParams": {  
    "connection": "acctAuth",  
    "processingType": "SequentialAndIterative",  
    "successResponses": {  
      "statusCode": [  
        200,  
        201,  
        202,  
        203,  
        204,  
      ]  
    }  
  }  
}
```

```

    205
  ]
},
"unsuccessResponses": null,
"doNotChangeIfFailed": true,
"call": {
  "call1": {
    "callOrder": 0,
    "stageNumber": 0,
    "showJobHistory": true,
    "http": {
      "url": "https://<URL>/Diversey/API/REST/Summit_RESTWCF.svc/RESTService/CommonWS_JsonObjCall",
      "contentType": "application/json",
      "httpMethod": "POST",
      "httpHeaders": {
        "Content-Type": "application/json"
      },
      "httpParams": "{ \"ServiceName\": \"<specify service name>\", \"objCommonParameters\": { \"_ProxyDeto
    },
    "listField": "OutputObject.SUBROOT",
    "keyField": "accountID",
    "colsToPropsMap": {

```

```
    "accountID": "UserID~#~char",
    "name": "EmailID~#~char",
    "displayName": "UserName~#~char",
    "status": "Active~#~char",
    "customproperty1": "Manager~#~char"
  },
  "pagination": {
    "PageWithHttpParamsAndHeaders": {
      "pageSizeParam": "pageSize",
      "pageSize": 25,
      "pageNumberParam": "PageIndex",
      "firstPageNumber": 0,
      "pageRecordCount": "objectList",
      "totalCountPath": "completeResponseMap.OutputObject.SUBROOT[0].totalrows",
      "paginationLocation": "httpParams"
    }
  },
  "statusConfig": {
    "active": "true",
    "inactive": "false"
  },
  "disableDeletedAccounts": true
}
```

```

    }
  }
},
"entitlementParams": {},
"acctEntParams": {}
}

```

- Support for defining the path of entitlements using the `entListField` variable in the `acctEntParams` attribute.

Example:

JSON

```

{
  "accountParams": {
    "connection": "userAuth",
    "processingType": "SequentialAndIterative",
    "call": {
      "call1": {
        "callOrder": 0,
        "stageNumber": 0,
        "http": {
          "url": "https://<url>/API/2.0/Data/EmployeeManagement/Employee/Default?token=${access_token}&sel

```

```
    "httpHeaders": {
      "Accept": "application/json"
    },
    "httpContentType": "application/json",
    "httpMethod": "GET"
  },
  "listField": "data",
  "keyField": "accountID",
  "colsToPropsMap": {
    "accountID": "id~#~char",
    "displayName": "FirstName~#~char",
    "name": "FirstName~#~char",
    "customproperty1": "UserName~#~char",
    "customproperty10": "CanLogin~#~char"
  },
  "pagination": {
    "offset": {
      "offsetParam": "skip",
      "batchParam": "take",
      "batchSize": 50,
      "totalCountPath": 15000
    }
  }
}
```

```

    }
  }
},
"entitlementParams": {
  "connection": "userAuth",
  "processingType": "SequentialAndIterative",
  "entTypes": {
    "Roles": {
      "entTypeOrder": 0,
      "call": {
        "call1": {
          "callOrder": 0,
          "stageNumber": 0,
          "http": {
            "url": "https://<url>/API/2.0/Data/EmployeeManagement/EmployeeProfile/Default?token=${access_
            "httpHeaders": {
              "Accept": "application/json"
            },
            "httpContentType": "application/json",
            "httpMethod": "GET"
          },
        },
      },
    },
  },
}

```

```

    "listField": "data",
    "keyField": "entitlementID",
    "colsToPropsMap": {
        "entitlementID": "id~#~char",
        "entitlement_value": "Name~#~char"
    },
    "pagination": {
        "offset": {
            "offsetParam": "skip",
            "batchParam": "take",
            "batchSize": 50,
            "totalCountPath": 15000
        }
    },
    "disableDeletedEntitlements": true
}
}
}
},
"acctEntParams": {
    "entTypes": {

```



```
"Roles": {
  "acctKeyField": "accountID",
  "entKeyField": "entitlementID",
  "call": {
    "call1": {
      "processingType": "http",
      "connection": "userAuth",
      "callOrder": 0,
      "stageNumber": 0,
      "http": {
        "httpHeaders": {
          "Accept": "application/json"
        },
        "url": "https://<url>/API/2.0/Data/EmployeeManagement/Employee/Default?token=${access_token}",
        "httpContentType": "application/json",
        "httpMethod": "GET"
      },
      "listField": "data",
      "entListField": "Profiles",
      "acctIdPath": "id",
      "entIdPath": "id",
      "pagination": {
```

```

        "offset": {
            "offsetParam": "skip",
            "batchParam": "take",
            "batchSize": 50,
            "totalCountPath": 15000
        }
    }
}
}
}
}
}
}
}
}
}
}

```

11. Support for importing only entitlements by specifying entitlement details in the `entitlementParams` attribute. Leave the `accountParams` attribute as blank.

Example:

JSON

```

{
  "accountParams": {},

```

```
"entitlementParams": {
  "processingType": "SequentialAndIterative",
  "entTypes": {
    "MUDetailsRest3": {
      "entTypeOrder": 0,
      "call": {
        "call1": {
          "connection": "acctAuth",
          "callOrder": 0,
          "stageNumber": 0,
          "http": {
            "httpHeaders": {
              "Authorization": "${access_token}",
              "Accept": "application/json"
            },
            "url": "<URL>",
            "httpContentType": "application/json",
            "httpMethod": "GET"
          },
          "statusConfig": {
            "active": "A",
            "inactive": "I"
          }
        }
      }
    }
  }
}
```

```

    },
    "listField": "data",
    "keyField": "entitlementID",
    "colsToPropsMap": {
        "entitlementID": "managementUnitId~#~char",
        "entitlement_value": "managementUnitId~#~char",
        "description": "managementUnitId~#~char",
        "customproperty1": "shortName~#~char",
        "customproperty2": "longName~#~char",
        "customproperty3": "responsibleMangerEmployeeId~#~char",
        "customproperty4": "responsibleManagerName~#~char",
        "customproperty5": "aspEligibilityFlag~#~char",
        "customproperty6": "reportingSectorId~#~char",
        "customproperty7": "reportingSectorName~#~char",
        "customproperty8": "reportingLineOfBusinessId~#~char",
        "customproperty9": "reportingLineOfBusinessName~#~char",
        "customproperty10": "reportingLineOfBusinessLongName~#~char",
        "customproperty11": "segmentCode~#~char",
        "customproperty12": "effStatusCode~#~char"
    },
    "disableDeletedEntitlements": true,
    "pagination": {

```

```

        "nextUrl": {
            "nextUrlPath": "${response?.objectList?.size()}>0?'https://<url>/api/v1/management-units?po
        }
    }
}
}
}
},
"acctEntParams": {
    "processingType": "acctToEntMapping"
}
}

```

12. Support for importing changes made for inactive accounts in the target application when the `includeExistingInactiveAccounts` variable in the `acctEntParams` attribute is set to true.

Example:

JSON

```

{
    "globalSettings": {

```

```
    "userSourceList": [  
      "1"  
    ]  
  },  
  "accountParams": {  
    "connection": "acctAuth",  
    "processingType": "SequentialAndIterative",  
    "statusAndThresholdConfig": {  
      "statusColumn": "customproperty11",  
      "activeStatus": [  
        "false"  
      ],  
      "deleteLinks": true,  
      "accountThresholdValue": 10000,  
      "correlateInactiveAccounts": false,  
      "inactivateAccountsNotInFile": true,  
      "deleteAccEntForActiveAccounts": true  
    },  
    "includeExistingInactiveAccounts": true,  
    "call": {  
      "call1": {  
        "callOrder": 0,  

```

```
"stageNumber": 0,
"http": {
  "url": "<URL>",
  "httpHeaders": {
    "Authorization": "${access_token}",
    "Accept": "text/plain",
    "Content-Type": "application/json"
  },
  "httpContentType": "application/json",
  "httpMethod": "GET",
  "httpParams": "[]"
},
"listField": "",
"keyField": "accountID",
"colsToPropsMap": {
  "accountID": "userName~#~char",
  "name": "userName~#~char",
  "displayName": "fullName~#~char",
  "customproperty1": "company~#~char",
  "customproperty2": "accountType~#~char",
  "customproperty3": "email~#~char",
  "status": "isDisabled~#~char",
```

```
        "customproperty11": "isDisabled~#~char",
        "customproperty31": "usersName~#~char"
    }
},
"call2": {
    "callOrder": 0,
    "stageNumber": 0,
    "showJobHistory": true,
    "http": {
        "url": "<URL>",
        "httpMethod": "POST",
        "httpParams": "[]",
        "httpHeaders": {
            "Authorization": "${access_token}",
            "Accept": "application/json"
        },
        "httpContentType": "application/json"
    },
    "listField": "",
    "keyField": "accountID",
    "colsToPropsMap": {
        "accountID": "userName~#~char",
```



```
        "name": "userName~#~char",
        "customproperty31": "STORE#ACC#ENT#MAPPINGINFO~#~char"
    },
    "disableDeletedAccounts": false
}
},
"acctEntMappings": {
    "Role": {
        "listPath": "roles",
        "idPath": "roleId",
        "keyField": "entitlementID"
    }
}
},
"entitlementParams": {
    "connection": "acctAuth",
    "processingType": "SequentialAndIterative",
    "successResponses": {
        "statusCode": [
            200,
            201,
            202,
```

```
    203,  
    204,  
    205  
  ]  
,  
"unsuccessResponses": null,  
"doNotChangeIfFailed": true,  
"entTypes": {  
  "Role": {  
    "entTypeOrder": 0,  
    "call": {  
      "call1": {  
        "callOrder": 0,  
        "stageNumber": 0,  
        "showJobHistory": true,  
        "http": {  
          "url": "<URL>",  
          "httpMethod": "GET",  
          "httpParams": {},  
          "httpHeaders": {  
            "Authorization": "${access_token}",  
            "Accept": "application/json"
```

```

    },
    "httpContentType": "application/json"
  },
  "listField": "",
  "keyField": "entitlementID",
  "colsToPropsMap": {
    "entitlementID": "id~#~char",
    "acctEntMappingInfoColumnFromEnt": "STORE#ACC#ENT#MAPPINGINFO~#~char",
    "description": "htmlDescription~#~char",
    "customproperty35": "allowInternals~#~char",
    "customproperty36": "allowContractors~#~char",
    "customproperty37": "allowConsultants~#~char",
    "customproperty38": "allowPartners~#~char",
    "customproperty39": "allowRobots~#~char",
    "customproperty40": "allowAffiliates~#~char",
    "customproperty4": "instCode~#~char",
    "customproperty5": "plantDescription~#~char",
    "entitlement_value": "title~#~char",
    "displayName": "title~#~char",
    "entitlement_glossary": "htmlDescription~#~char",
    "customproperty1": "roleName~#~char",
    "customproperty2": "description~#~char",

```

```

        "customproperty29": "requireLineManagerApprovalForInternals~#~char",
        "customproperty30": "requireLineManagerApprovalForContractors~#~char",
        "customproperty31": "requireLineManagerApprovalForConsultants~#~char",
        "customproperty32": "requireLineManagerApprovalForPartners~#~char",
        "customproperty33": "requireLineManagerApprovalForRobots~#~char",
        "customproperty34": "requireLineManagerApprovalForAffiliates~#~char"
    },
    "disableDeletedEntitlements": true
}
},
"entOwnerMappings": {
    "listField": "accessApprovers",
    "idPath": "userName",
    "keyField": "customproperty21"
}
}
},
"acctEntParams": {
    "entTypes": {
        "Role": {
            "call": {

```

```
    "call1": {
      "connection": "acctAuth",
      "callOrder": 0,
      "stageNumber": 0,
      "showJobHistory": true,
      "processingType": "acctToEntMapping"
    },
    "call2": {
      "connection": "acctAuth",
      "callOrder": 1,
      "stageNumber": 1,
      "showJobHistory": true,
      "processingType": "entToAcctMapping",
      "ownerKeyField": "customproperty21"
    }
  }
}
```

Importing Entitlement Owners

From Release v24.2 onwards, the connector imports entitlement owners in the target application as Rank 1 owners in EIC. After account or access import, the existing owner with Rank 1 is deleted from EIC if a new owner with Rank 1 is added in the target. However, existing entitlement owners with different ranks, such as Rank 2 or Rank 3, are retained after the import.

1. Importing entitlement owners for the following scenarios:

1. If entitlement owners are imported while running account import and entitlement owner details are available in the Get Entitlements API response, use a format similar to the following in **ImportAccountEntJSON**:

Review the following guidelines:

- Entitlement owner details are stored in customproperty31 of the accounts table, so add the following mapping in the colsToPropsMap attribute: `"customproperty31": "STORE#ACC#ENT#MAPPINGINFO~#~char`
- Add the entOwnerMappings subelement which contains listPath, idPath and keyField of owner in the acctEntMappings element.

JSON

```
{
  "accountParams": {
    "connection": "acctAuth",
    "processingType": "SequentialAndIterative",
    "successResponses": {
```

```
    "statusCode": [
      200,
      201,
      202,
      203,
      204,
      205
    ]
  },
  "unsuccessResponses": null,
  "doNotChangeIfFailed": true,
  "call": {
    "call1": {
      "callOrder": 0,
      "stageNumber": 0,
      "showJobHistory": true,
      "http": {
        "url": "http://<domain name>/ECM/maintenance/testRestConnectorResponse",
        "httpContentType": "application/json",
        "httpMethod": "GET",
        "httpHeaders": {
          "Accept": "application/json"
        }
      }
    }
  }
}
```

```

    }
  },
  "listField": "records",
  "keyField": "accountID",
  "colsToPropsMap": {
    "accountID": "Id~#~char",
    "name": "Name~#~char",
    "displayName": "Email~#~char",
    "status": "IsActive~#~char",
    "customproperty31": "STORE#ACC#ENT#MAPPINGINFO~#~char"
  },
  "disableDeletedAccounts": true
}
},
"acctEntMappings": {
  "Groups": {
    "importAsEntitlement": true,
    "listPath": "groups",
    "idPath": "id",
    "keyField": "entitlementID",
    "colsToPropsMap": {
      "entitlement_value": "name~#~char",

```



```

        "entitlementID": "id~#~char",
        "displayname": "displayName~#~char"
    },
    "entOwnerMappings": {
        "listPath": "groups",
        "idPath": "owner",
        "keyField": "email"
    }
}
},
"entitlementParams": {
    "connection": "acctAuth",
    "processingType": "SequentialAndIterative",
    "unsuccessResponses": null,
    "doNotChangeIfFailed": true,
    "entTypes": {
        "Groups": {
        }
    }
},
"acctEntParams": {

```

```
"connection": "acctAuth",
"successResponses": {
  "statusCode": [
    200,
    201,
    202,
    203,
    204,
    205
  ]
},
"unsuccessResponses": null,
"entTypes": {
  "Groups": {
    "call": {
      "call1": {
        "processingType": "acctToEntMapping",
        "ownerKeyField": "email"
      }
    }
  }
}
```

```
}  
}
```

2. If entitlement owners are imported while running access import and entitlement owner details are available in the Get Entitlements API response, use a format similar to the following in **ImportAccountEntJSON**:

Review the following guidelines:

- Entitlement owners details are stored in acctEntMappingInfoColumnFromEnt, so add the following mapping in the colsToPropsMap attribute: **"acctEntMappingInfoColumnFromEnt": "STORE#ACC#ENT#MAPPINGINFO~#~character"**
- Add the entOwnerMappings subelement which contains listPath, idPath and keyField of owner in the acctEntMappings element.

JSON

```
{  
  "accountParams": {},  
  "entitlementParams": {  
    "successResponses": {  
      "statusCode": [  
        200,  
        201,  
        202,  
        203,  
        204,  
        205,  
        206,  
        207,  
        208,  
        209,  
        210,  
        211,  
        212,  
        213,  
        214,  
        215,  
        216,  
        217,  
        218,  
        219,  
        220,  
        221,  
        222,  
        223,  
        224,  
        225,  
        226,  
        227,  
        228,  
        229,  
        230,  
        231,  
        232,  
        233,  
        234,  
        235,  
        236,  
        237,  
        238,  
        239,  
        240,  
        241,  
        242,  
        243,  
        244,  
        245,  
        246,  
        247,  
        248,  
        249,  
        250,  
        251,  
        252,  
        253,  
        254,  
        255,  
        256,  
        257,  
        258,  
        259,  
        260,  
        261,  
        262,  
        263,  
        264,  
        265,  
        266,  
        267,  
        268,  
        269,  
        270,  
        271,  
        272,  
        273,  
        274,  
        275,  
        276,  
        277,  
        278,  
        279,  
        280,  
        281,  
        282,  
        283,  
        284,  
        285,  
        286,  
        287,  
        288,  
        289,  
        290,  
        291,  
        292,  
        293,  
        294,  
        295,  
        296,  
        297,  
        298,  
        299,  
        300,  
        301,  
        302,  
        303,  
        304,  
        305,  
        306,  
        307,  
        308,  
        309,  
        310,  
        311,  
        312,  
        313,  
        314,  
        315,  
        316,  
        317,  
        318,  
        319,  
        320,  
        321,  
        322,  
        323,  
        324,  
        325,  
        326,  
        327,  
        328,  
        329,  
        330,  
        331,  
        332,  
        333,  
        334,  
        335,  
        336,  
        337,  
        338,  
        339,  
        340,  
        341,  
        342,  
        343,  
        344,  
        345,  
        346,  
        347,  
        348,  
        349,  
        350,  
        351,  
        352,  
        353,  
        354,  
        355,  
        356,  
        357,  
        358,  
        359,  
        360,  
        361,  
        362,  
        363,  
        364,  
        365,  
        366,  
        367,  
        368,  
        369,  
        370,  
        371,  
        372,  
        373,  
        374,  
        375,  
        376,  
        377,  
        378,  
        379,  
        380,  
        381,  
        382,  
        383,  
        384,  
        385,  
        386,  
        387,  
        388,  
        389,  
        390,  
        391,  
        392,  
        393,  
        394,  
        395,  
        396,  
        397,  
        398,  
        399,  
        400,  
        401,  
        402,  
        403,  
        404,  
        405,  
        406,  
        407,  
        408,  
        409,  
        410,  
        411,  
        412,  
        413,  
        414,  
        415,  
        416,  
        417,  
        418,  
        419,  
        420,  
        421,  
        422,  
        423,  
        424,  
        425,  
        426,  
        427,  
        428,  
        429,  
        430,  
        431,  
        432,  
        433,  
        434,  
        435,  
        436,  
        437,  
        438,  
        439,  
        440,  
        441,  
        442,  
        443,  
        444,  
        445,  
        446,  
        447,  
        448,  
        449,  
        450,  
        451,  
        452,  
        453,  
        454,  
        455,  
        456,  
        457,  
        458,  
        459,  
        460,  
        461,  
        462,  
        463,  
        464,  
        465,  
        466,  
        467,  
        468,  
        469,  
        470,  
        471,  
        472,  
        473,  
        474,  
        475,  
        476,  
        477,  
        478,  
        479,  
        480,  
        481,  
        482,  
        483,  
        484,  
        485,  
        486,  
        487,  
        488,  
        489,  
        490,  
        491,  
        492,  
        493,  
        494,  
        495,  
        496,  
        497,  
        498,  
        499,  
        500,  
        501,  
        502,  
        503,  
        504,  
        505,  
        506,  
        507,  
        508,  
        509,  
        510,  
        511,  
        512,  
        513,  
        514,  
        515,  
        516,  
        517,  
        518,  
        519,  
        520,  
        521,  
        522,  
        523,  
        524,  
        525,  
        526,  
        527,  
        528,  
        529,  
        530,  
        531,  
        532,  
        533,  
        534,  
        535,  
        536,  
        537,  
        538,  
        539,  
        540,  
        541,  
        542,  
        543,  
        544,  
        545,  
        546,  
        547,  
        548,  
        549,  
        550,  
        551,  
        552,  
        553,  
        554,  
        555,  
        556,  
        557,  
        558,  
        559,  
        560,  
        561,  
        562,  
        563,  
        564,  
        565,  
        566,  
        567,  
        568,  
        569,  
        570,  
        571,  
        572,  
        573,  
        574,  
        575,  
        576,  
        577,  
        578,  
        579,  
        580,  
        581,  
        582,  
        583,  
        584,  
        585,  
        586,  
        587,  
        588,  
        589,  
        590,  
        591,  
        592,  
        593,  
        594,  
        595,  
        596,  
        597,  
        598,  
        599,  
        600,  
        601,  
        602,  
        603,  
        604,  
        605,  
        606,  
        607,  
        608,  
        609,  
        610,  
        611,  
        612,  
        613,  
        614,  
        615,  
        616,  
        617,  
        618,  
        619,  
        620,  
        621,  
        622,  
        623,  
        624,  
        625,  
        626,  
        627,  
        628,  
        629,  
        630,  
        631,  
        632,  
        633,  
        634,  
        635,  
        636,  
        637,  
        638,  
        639,  
        640,  
        641,  
        642,  
        643,  
        644,  
        645,  
        646,  
        647,  
        648,  
        649,  
        650,  
        651,  
        652,  
        653,  
        654,  
        655,  
        656,  
        657,  
        658,  
        659,  
        660,  
        661,  
        662,  
        663,  
        664,  
        665,  
        666,  
        667,  
        668,  
        669,  
        670,  
        671,  
        672,  
        673,  
        674,  
        675,  
        676,  
        677,  
        678,  
        679,  
        680,  
        681,  
        682,  
        683,  
        684,  
        685,  
        686,  
        687,  
        688,  
        689,  
        690,  
        691,  
        692,  
        693,  
        694,  
        695,  
        696,  
        697,  
        698,  
        699,  
        700,  
        701,  
        702,  
        703,  
        704,  
        705,  
        706,  
        707,  
        708,  
        709,  
        710,  
        711,  
        712,  
        713,  
        714,  
        715,  
        716,  
        717,  
        718,  
        719,  
        720,  
        721,  
        722,  
        723,  
        724,  
        725,  
        726,  
        727,  
        728,  
        729,  
        730,  
        731,  
        732,  
        733,  
        734,  
        735,  
        736,  
        737,  
        738,  
        739,  
        740,  
        741,  
        742,  
        743,  
        744,  
        745,  
        746,  
        747,  
        748,  
        749,  
        750,  
        751,  
        752,  
        753,  
        754,  
        755,  
        756,  
        757,  
        758,  
        759,  
        760,  
        761,  
        762,  
        763,  
        764,  
        765,  
        766,  
        767,  
        768,  
        769,  
        770,  
        771,  
        772,  
        773,  
        774,  
        775,  
        776,  
        777,  
        778,  
        779,  
        780,  
        781,  
        782,  
        783,  
        784,  
        785,  
        786,  
        787,  
        788,  
        789,  
        790,  
        791,  
        792,  
        793,  
        794,  
        795,  
        796,  
        797,  
        798,  
        799,  
        800,  
        801,  
        802,  
        803,  
        804,  
        805,  
        806,  
        807,  
        808,  
        809,  
        810,  
        811,  
        812,  
        813,  
        814,  
        815,  
        816,  
        817,  
        818,  
        819,  
        820,  
        821,  
        822,  
        823,  
        824,  
        825,  
        826,  
        827,  
        828,  
        829,  
        830,  
        831,  
        832,  
        833,  
        834,  
        835,  
        836,  
        837,  
        838,  
        839,  
        840,  
        841,  
        842,  
        843,  
        844,  
        845,  
        846,  
        847,  
        848,  
        849,  
        850,  
        851,  
        852,  
        853,  
        854,  
        855,  
        856,  
        857,  
        858,  
        859,  
        860,  
        861,  
        862,  
        863,  
        864,  
        865,  
        866,  
        867,  
        868,  
        869,  
        870,  
        871,  
        872,  
        873,  
        874,  
        875,  
        876,  
        877,  
        878,  
        879,  
        880,  
        881,  
        882,  
        883,  
        884,  
        885,  
        886,  
        887,  
        888,  
        889,  
        890,  
        891,  
        892,  
        893,  
        894,  
        895,  
        896,  
        897,  
        898,  
        899,  
        900,  
        901,  
        902,  
        903,  
        904,  
        905,  
        906,  
        907,  
        908,  
        909,  
        910,  
        911,  
        912,  
        913,  
        914,  
        915,  
        916,  
        917,  
        918,  
        919,  
        920,  
        921,  
        922,  
        923,  
        924,  
        925,  
        926,  
        927,  
        928,  
        929,  
        930,  
        931,  
        932,  
        933,  
        934,  
        935,  
        936,  
        937,  
        938,  
        939,  
        940,  
        941,  
        942,  
        943,  
        944,  
        945,  
        946,  
        947,  
        948,  
        949,  
        950,  
        951,  
        952,  
        953,  
        954,  
        955,  
        956,  
        957,  
        958,  
        959,  
        960,  
        961,  
        962,  
        963,  
        964,  
        965,  
        966,  
        967,  
        968,  
        969,  
        970,  
        971,  
        972,  
        973,  
        974,  
        975,  
        976,  
        977,  
        978,  
        979,  
        980,  
        981,  
        982,  
        983,  
        984,  
        985,  
        986,  
        987,  
        988,  
        989,  
        990,  
        991,  
        992,  
        993,  
        994,  
        995,  
        996,  
        997,  
        998,  
        999,  
        1000,  
        1001,  
        1002,  
        1003,  
        1004,  
        1005,  
        1006,  
        1007,  
        1008,  
        1009,  
        1010,  
        1011,  
        1012,  
        1013,  
        1014,  
        1015,  
        1016,  
        1017,  
        1018,  
        1019,  
        1020,  
        1021,  
        1022,  
        1023,  
        1024,  
        1025,  
        1026,  
        1027,  
        1028,  
        1029,  
        1030,  
        1031,  
        1032,  
        1033,  
        1034,  
        1035,  
        1036,  
        1037,  
        1038,  
        1039,  
        1040,  
        1041,  
        1042,  
        1043,  
        1044,  
        1045,  
        1046,  
        1047,  
        1048,  
        1049,  
        1050,  
        1051,  
        1052,  
        1053,  
        1054,  
        1055,  
        1056,  
        1057,  
        1058,  
        1059,  
        1060,  
        1061,  
        1062,  
        1063,  
        1064,  
        1065,  
        1066,  
        1067,  
        1068,  
        1069,  
        1070,  
        1071,  
        1072,  
        1073,  
        1074,  
        1075,  
        1076,  
        1077,  
        1078,  
        1079,  
        1080,  
        1081,  
        1082,  
        1083,  
        1084,  
        1085,  
        1086,  
        1087,  
        1088,  
        1089,  
        1090,  
        1091,  
        1092,  
        1093,  
        1094,  
        1095,  
        1096,  
        1097,  
        1098,  
        1099,  
        1100,  
        1101,  
        1102,  
        1103,  
        1104,  
        1105,  
        1106,  
        1107,  
        1108,  
        1109,  
        1110,  
        1111,  
        1112,  
        1113,  
        1114,  
        1115,  
        1116,  
        1117,  
        1118,  
        1119,  
        1120,  
        1121,  
        1122,  
        1123,  
        1124,  
        1125,  
        1126,  
        1127,  
        1128,  
        1129,  
        1130,  
        1131,  
        1132,  
        1133,  
        1134,  
        1135,  
        1136,  
        1137,  
        1138,  
        1139,  
        1140,  
        1141,  
        1142,  
        1143,  
        1144,  
        1145,  
        1146,  
        1147,  
        1148,  
        1149,  
        1150,  
        1151,  
        1152,  
        1153,  
        1154,  
        1155,  
        1156,  
        1157,  
        1158,  
        1159,  
        1160,  
        1161,  
        1162,  
        1163,  
        1164,  
        1165,  
        1166,  
        1167,  
        1168,  
        1169,  
        1170,  
        1171,  
        1172,  
        1173,  
        1174,  
        1175,  
        1176,  
        1177,  
        1178,  
        1179,  
        1180,  
        1181,  
        1182,  
        1183,  
        1184,  
        1185,  
        1186,  
        1187,  
        1188,  
        1189,  
        1190,  
        1191,  
        1192,  
        1193,  
        1194,  
        1195,  
        1196,  
        1197,  
        1198,  
        1199,  
        1200,  
        1201,  
        1202,  
        1203,  
        1204,  
        1205,  
        1206,  
        1207,  
        1208,  
        1209,  
        1210,  
        1211,  
        1212,  
        1213,  
        1214,  
        1215,  
        1216,  
        1217,  
        1218,  
        1219,  
        1220,  
        1221,  
        1222,  
        1223,  
        1224,  
        1225,  
        1226,  
        1227,  
        1228,  
        1229,  
        1230,  
        1231,  
        1232,  
        1233,  
        1234,  
        1235,  
        1236,  
        1237,  
        1238,  
        1239,  
        1240,  
        1241,  
        1242,  
        1243,  
        1244,  
        1245,  
        1246,  
        1247,  
        1248,  
        1249,  
        1250,  
        1251,  
        1252,  
        1253,  
        1254,  
        1255,  
        1256,  
        1257,  
        1258,  
        1259,  
        1260,  
        1261,  
        1262,  
        1263,  
        1264,  
        1265,  
        1266,  
        1267,  
        1268,  
        1269,  
        1270,  
        1271,  
        1272,  
        1273,  
        1274,  
        1275,  
        1276,  
        1277,  
        1278,  
        1279,  
        1280,  
        1281,  
        1282,  
        1283,  
        1284,  
        1285,  
        1286,  
        1287,  
        1288,  
        1289,  
        1290,  
        1291,  
        1292,  
        1293,  
        1294,  
        1295,  
        1296,  
        1297,  
        1298,  
        1299,  
        1300,  
        1301,  
        1302,  
        1303,  
        1304,  
        1305,  
        1306,  
        1307,  
        1308,  
        1309,  
        1310,  
        1311,  
        1312,  
        1313,  
        1314,  
        1315,  
        1316,  
        1317,  
        1318,  
        1319,  
        1320,  
        1321,  
        1322,  
        1323,  
        1324,  
        1325,  
        1326,  
        1327,  
        1328,  
        1329,  
        1330,  
        1331,  
        1332,  
        1333,  
        1334,  
        1335,  
        1336,  
        1337,  
        1338,  
        1339,  
        1340,  
        1341,  
        1342,  
        1343,  
        1344,  
        1345,  
        1346,  
        1347,  
        1348,  
        1349,  
        1350,  
        1351,  
        1352,  
        1353,  
        1354,  
        1355,  
        1356,  
        1357,  
        1358,  
        1359,  
        1360,  
        1361,  
        1362,  
        1363,  
        1364,  
        1365,  
        1366,  
        1367,  
        1368,  
        1369,  
        1370,  
        1371,  
        1372,  
        1373,  
        1374,  
        1375,  
        1376,  
        1377,  
        1378,  
        1379,  
        1380,  
        1381,  
        1382,  
        1383,  
        1384,  
        1385,  
        1386,  
        1387,  
        1388,  
        1389,  
        1390,  
        1391,  
        1392,  
        1393,  
        1394,  
        1395,  
        1396,  
        1397,  
        1398,  
        1399,  
        1400,  
        1401,  
        1402,  
        1403,  
        1404,  
        1405,  
        1406,  
        1407,  
        1408,  
        1409,  
        1410,  
        1411,  
        1412,  
        1413,  
        1414,  
        1415,  
        1416,  
        1417,  
        1418,  
        1419,  
        1420,  
        1421,  
        1422,  
        1423,  
        1424,  
        1425,  
        1426,  
        1427,  
        1428,  
        1429,  
        1430,  
        1431,  
        1432,  
        1433,  
        1434,  
        1435,  
        1436,  
        1437,  
        1438,  
        1439,  
        1440,  
        1441,  
        1442,  
        1443,  
        1444,  
        1445,  
        1446,  
        1447,  
        1448,  
        1449,  
        1450,  
        1451,  
        1452,  
        1453,  
        1454,  
        1455,  
        1456,  
        1457,  
        1458,  
        1459,  
        1460,  
        1461,  
        1462,  
        1463,  
        1464,  
        1465,  
        1466,  
        1467,  
        1468,  
        1469,  
        1470,  
        1471,  
        1472,  
        1473,  
        1474,  
        1475,  
        1476,  
        1477,  
        1478,  
        1479,  
        1480,  
        1481,  
        1482,  
        1483,  
        1484,  
        1485,  
        1486,  
        1487,  
        1488,  
        1489,  
        1490,  
        1491,  
        1492,  
        1493,  
        1494,  
        1495,  
        1496,  
        1497,  
        1498,  
        1499,  
        1500,  
        1501,  
        1502,  
        1503,  
        1504,  
        1505,  
        1506,  
        1507,  
        1508,  
        1509,  
        1510,  
        1511,  
        1512,  
        1513,  
        1514,  
        1515,  
        1516,  
        1517,  
        1518,  
        1519,  
        1520,  
        1521,  
        1522,  
        1523,  
        1524,  
        1525,  
        1526,  
        1527,  
        1528,  
        1529,  
        1530,  
        1531,  
        1532,  
        1533,  
        1534,  
        1535,  
        1536,  
        1537,  
        1538,  
        1539,  
        1540,  
        1541,  
        1542,  
        1543,  
        1544,  
        1545,  
        1546,  
        1547,  
        1548,  
        1549,  
        1550,  
        1551,  
        1552,  
        1553,  
        1554,  
        1555,  
        1556,  
        1557,  
        1558,  
        1559,  
        1560,  
        1561,  
        1562,  
        1563,  
        1564,  
        1565,  
        1566,  
        1567,  
        1568,  
        1569,  
        1570,  
        1571,  
        1572,  
        1573,  
        1574,  
        1575,  
        1576,  
        1577,  
        1578,  
        1579,  
        1580,  
        1581,  
        1582,  
        1583,  
        1584,  
        1585,  
        1586,  
        1587,  
        1588,  
        1589,  
        1590,  
        1591,  
        1592,  
        1593,  
        1594,  
        1595,  
        1596,  
        1597,  
        1598,  
        1599,  
        1600,  
        1601,  
        1602,  
        1603,  
        1604,  
        1605,  
        1606,  
        1607,  
        1608,  
        1609,  
        1610,  
        1611,  
        1612,  
        1613,  
        1614,  
        1615,  
        1616,  
        1617,  
        1618,  
        1619,  
        1620,  
        1621,  
        1622
```

```
        203,  
        204,  
        205  
    ]  
},  
"unsuccessResponses": {  
    "message": "",  
    "response": "",  
    "statusCode": [  
        null,  
        400,  
        401,  
        403,  
        404,  
        405,  
        409  
    ]  
},  
"doNotChangeIfFailed": true,  
"connection": "acctAuth",  
"processingType": "SequentialAndIterative",  
"entTypes": {
```

```
"Access": {
  "entTypeOrder": 0,
  "call": {
    "call1": {
      "connection": "acctAuth",
      "callOrder": 0,
      "stageNumber": 0,
      "http": {
        "httpHeaders": {
          "Authorization": "${access_token}"
        },
        "url": "<URL>/",
        "httpContentType": "application/json",
        "httpMethod": "GET"
      },
      "listField": "access",
      "keyField": "entitlementID",
      "colsToPropsMap": {
        "entitlementID": "id~#~char",
        "entitlement_value": "id~#~char",
        "displayname": "displayname~#~char",
        "description": "description~#~char",
```

```

        "customproperty1": "metadata.role~#~char",
        "customproperty2": "metadata.utl10~#~char",
        "status": "status~#~char",
        "acctEntMappingInfoColumnFromEnt": "STORE#ACC#ENT#MAPPINGINFO~#~char"
    },
    "disableDeletedEntitlements": true
}
},
"entOwnerMappings": {
    "listField": "Owner",
    "idPath": "email",
    "keyField": "email"
}
}
},
"acctEntParams": {
    "entTypes": {
        "Access": {
            "call": {
                "call1": {
                    "connection": "acctAuth",

```

```

        "callOrder": 1,
        "stageNumber": 1,
        "showJobHistory": true,
        "processingType": "entToAcctMapping",
        "ownerKeyField": "email"
      }
    }
  }
}

```

3. If entitlement owner details are available in the acctEntParams API response for `http`, `httpEntToAcct`, and `httpAcctToEnt` processingTypes, use a format similar to the following in `ImportAccountEntJSON`:

Review the following guideline:

- Add `ownerKeyField` and `ownerIdPath` for the `http`, `httpEntToAcct`, and `httpAcctToEnt` processingTypes.

JSON

```

{
  "accountParams": {
    "connection": "acctAuth",
    "processingType": "SequentialAndIterative",

```

```
"successResponses": {
  "statusCode": [
    200,
    201,
    202,
    203,
    204,
    205
  ]
},
"unsuccessResponses": null,
"doNotChangeIfFailed": true,
"call": {
  "call1": {
    "callOrder": 0,
    "stageNumber": 0,
    "showJobHistory": true,
    "http": {
      "url": "http://<domain name>/ECM/maintenance/testRestConnectorResponse",
      "httpContentType": "application/json",
      "httpMethod": "GET",
      "httpHeaders": {
```



```

        "Accept": "application/json"
    }
},
"listField": "records",
"keyField": "accountID",
"colsToPropsMap": {
    "accountID": "Id~#~char",
    "name": "Name~#~char",
    "displayName": "Email~#~char",
    "status": "IsActive~#~char",
    "customproperty31": "STORE#ACC#ENT#MAPPINGINFO~#~char"
},
"disableDeletedAccounts": true
}
},
"entitlementParams": {
    "connection": "acctAuth",
    "processingType": "SequentialAndIterative",
    "successResponses": {
        "statusCode": [
            200,

```

```
    201,  
    202,  
    203,  
    204,  
    205  
  ]  
},  
"unsuccessResponses": null,  
"doNotChangeIfFailed": true,  
"entTypes": {  
  "Organization": {  
    "entTypeOrder": 0,  
    "call": {  
      "call1": {  
        "connection": "acctAuth",  
        "callOrder": 0,  
        "stageNumber": 0,  
        "showJobHistory": true,  
        "http": {  
          "url": "http://<domain name>/ECM/maintenance/testRestConnectorResponse",  
          "httpContentType": "application/json",  
          "httpMethod": "GET",
```

```

        "httpHeaders": {
            "Accept": "application/json"
        },
        "statusConfig": {
            "active": "true",
            "inactive": "false"
        },
        "listField": "records",
        "keyField": "entitlementID",
        "colsToPropsMap": {
            "entitlement_value": "RoleName~#~char",
            "entitlementID": "Id~#~char",
            "displayname": "RoleName~#~char",
            "status": "IsActive~#~char"
        },
        "disableDeletedEntitlements": true
    }
}
},

```

```
"acctEntParams": {
  "successResponses": {
    "statusCode": [
      200,
      201,
      202,
      203,
      204,
      205
    ]
  },
  "unsuccessResponses": null,
  "entTypes": {
    "Organization": {
      "call": {
        "call1": {
          "processingType": "http",
          "connection": "acctAuth",
          "showJobHistory": true,
          "callOrder": 0,
          "stageNumber": 0,
          "http": {
```


JSON

```
{
  "accountParams": {
    "connection": "acctAuth",
    "processingType": "SequentialAndIterative",
    "successResponses": {
      "statusCode": [
        200,
        201,
        202,
        203,
        204,
        205
      ]
    },
    "unsuccessResponses": null,
    "doNotChangeIfFailed": true,
    "call": {
      "call1": {
        "callOrder": 0,
        "stageNumber": 0,
        "showJobHistory": true,

```

```

    "http": {
      "url": "http://<domain name>/ECM/maintenance/testRestConnectorResponse",
      "httpContentType": "application/json",
      "httpMethod": "GET",
      "httpHeaders": {
        "Accept": "application/json"
      }
    },
    "listField": "records",
    "keyField": "accountID",
    "colsToPropsMap": {
      "accountID": "Id~#~char",
      "name": "Name~#~char",
      "displayName": "Email~#~char",
      "status": "IsActive~#~char",
      "customproperty31": "STORE#ACC#ENT#MAPPINGINFO~#~char"
    },
    "disableDeletedAccounts": true
  }
},
"entitlementParams": {

```

```
"connection": "acctAuth",
"processingType": "SequentialAndIterative",
"successResponses": {
  "statusCode": [
    200,
    201,
    202,
    203,
    204,
    205
  ]
},
"unsuccessResponses": null,
"doNotChangeIfFailed": true,
"entTypes": {
  "Safes": {
    "entTypeOrder": 0,
    "call": {
      "call1": {
        "connection": "acctAuth",
        "callOrder": 0,
        "stageNumber": 0,
```



```
"showJobHistory": true,
"http": {
  "url": "http://<domain name>/ECM/maintenance/testRestConnectorResponse",
  "contentType": "application/json",
  "method": "GET",
  "headers": {
    "Accept": "application/json"
  }
},
"statusConfig": {
  "active": "true",
  "inactive": "false"
},
"listField": "records",
"keyField": "entitlementID",
"colsToPropsMap": {
  "entitlement_value": "RoleName~#~char",
  "entitlementID": "Id~#~char",
  "displayname": "RoleName~#~char",
  "status": "IsActive~#~char"
},
"disableDeletedEntitlements": true
```

```

        }
    }
}
},
"acctEntParams": {},
"entOwnerParams": {
    "connection": "acctAuth",
    "successResponses": {
        "statusCode": [
            200,
            201,
            202,
            203,
            204,
            205
        ]
    },
    "unsuccessResponses": null,
    "doNotChangeIfFailed": true,
    "entTypes": {
        "Safes": {

```

```
"call": {
  "call1": {
    "processingType": "httpOwner",
    "connection": "acctAuth",
    "showJobHistory": true,
    "callOrder": 0,
    "stageNumber": 0,
    "http": {
      "url": "http://<domain name>/ECM/maintenance/testRestConnectorResponse",
      "httpContentType": "application/json",
      "httpMethod": "GET",
      "httpHeaders": {
        "Accept": "application/json"
      }
    },
    "listField": "records",
    "entIdPath": "Id",
    "ownerIdPath": "Owner",
    "ownerKeyField": "email",
    "entKeyField": "entitlementID"
  }
}
```

```
}  
}  
}  
}
```

2. Support for specifying the user source value when users are imported from source other than REST-based applications.

- Add all user source values as comma separated values in the `userSourceList` variable under the `globalSettings` attribute in **ImportAccountEntJSON** in the following format: `userSourceList: ["52", "53"]`
- Add all user sources, in the following format: `userSourceList : ["ALL"]`

Example:

JSON

```
{  
  "globalSettings": {  
    "userSourceList": [  
      "10",  
      "20"  
    ]  
  },  
  "accountParams": {
```

```
"createUsers": false,
"adminName": "admin",
"connection": "acctAuth",
"processingType": "SequentialAndIterative",
"statusAndThresholdConfig": {
  "statusColumn": "customproperty2",
  "activeStatus": [
    "active"
  ],
  "deleteLinks": true,
  "accountThresholdValue": 50,
  "correlateInactiveAccounts": false,
  "inactivateAccountsNotInFile": true,
  "deleteAccEntForActiveAccounts": true
},
"call": {
  "call1": {
    "callOrder": 0,
    "stageNumber": 0,
    "showJobHistory": true,
    "http": {
      "url": "https://<domain name>/2.0/users",
```

```
    "httpHeaders": {
      "Authorization": "${access_token}",
      "Accept": "application/json"
    },
    "httpContentType": "application/json",
    "httpMethod": "GET"
  },
  "listField": "entries",
  "keyField": "accountID",
  "colsToPropsMap": {
    "accountID": "id~~char",
    "name": "login~~char",
    "displayName": "name~~char",
    "accounttype": "type~~char",
    "customproperty13": "created_at~~char",
    "customproperty14": "modified_at~~char",
    "customproperty1": "emailAliases~~char",
    "customproperty2": "status~~char",
    "customproperty3": "language~~char",
    "customproperty4": "timezone~~char",
    "customproperty5": "space_amount~~char",
    "customproperty6": "space_used~~char",
```

```

        "status": "status~#~char",
        "customproperty7": "max_upload_size~#~char",
        "customproperty8": "job_title~#~char",
        "customproperty9": "phone~#~char",
        "customproperty10": "address~#~char",
        "customproperty11": "avatar_url~#~char",
        "customproperty12": "notification_email~#~char"
    }
}
},
"entitlementParams": {
    "connection": "acctAuth",
    "processingType": "SequentialAndIterative",
    "entTypes": {
        "Folders": {
            "entTypeOrder": 0,
            "entTypeLabels": {
                "customproperty1": "type",
                "customproperty2": "sequence_id",
                "customproperty3": "etag"
            }
        },

```

```
"call": {
  "call1": {
    "callOrder": 0,
    "stageNumber": 0,
    "showJobHistory": true,
    "http": {
      "url": "https://<domain name>/2.0/folders/0/items?fields=type,id,sequence_id,etag,name,size,c",
      "httpHeaders": {
        "Authorization": "${access_token}",
        "Accept": "application/json"
      },
      "httpContentType": "application/json",
      "httpMethod": "GET"
    },
    "listField": "entries",
    "keyField": "entitlementID",
    "colsToPropsMap": {
      "entitlementID": "id~~char",
      "entitlement_value": "name~~char",
      "customproperty1": "type~~char",
      "customproperty2": "sequence_id~~char",
      "customproperty3": "etag~~char",
```



```

        "acctEntMappingInfoColumnFromEnt": "STORE#ACC#ENT#MAPPINGINFO~#~char"
    },
    "disableDeletedEntitlements": true
}
},
"entOwnerMappings": {
    "listField": "",
    "idPath": "owned_by.login",
    "keyField": "email"
}
}
},
"acctEntParams": {
    "connection": "acctAuth",
    "entTypes": {
        "Folders": {
            "call": {
                "call1": {
                    "callOrder": 0,
                    "stageNumber": 0,
                    "showJobHistory": true,

```

```
"processingType": "httpEntToAcct",
"http": {
  "httpHeaders": {
    "Authorization": "${access_token}",
    "Accept": "application/json"
  },
  "url": "https://<domain name>/2.0/folders/${id}/collaborations",
  "httpContentType": "application/json",
  "httpMethod": "GET"
},
"listField": "entries",
"entKeyField": "entitlementID",
"acctIdPath": "accessible_by.id",
"acctKeyField": "accountID"
},
"call2": {
  "callOrder": 0,
  "stageNumber": 0,
  "showJobHistory": true,
  "processingType": "entToAcctMapping",
  "ownerKeyField": "email"
}
```

```

    }
  }
}
}
}

```


The following table describes attributes in `ImportAccountEntJSON`:

Attributes	Variable	Description
accountParams	Connection	Populates the connection type value specified in the acctAuth attribute while defining <code>ConnectionJSON</code> .
	processingType	Always specify the processing type as .
	successResponses	Use this variable to specify the possible response messages on successfully importing accounts.
	doNotChangelfFailed	Use this variable to instruct the connector to not delete or inactivate existing data on failure of the account API, the entType API, or the


Attributes	Variable	Description
		mapping API. When set to true, the connector does not delete or inactivate existing data.
	call	Use this variable to make an API call to the target application API for importing accounts. The connector allows you to specify multiple API calls to import accounts. You must specify each API call for importing accounts as a separate call. There is no fixed name, but the preferred naming convention is: call1, call2, call3.
	callOrder	Use this variable to specify the order to call APIs. Specify the value for the first call as zero. The lowest order value is called the first call.
	stageNumber	Use this variable to specify the stage number for the API call. Specify the stage number as zero for the first API call.
	url	Use this variable to specify the URL of the target application.

Attributes	Variable	Description
	httpMethod	Use this variable to specify the API call method for a successful API call such as GET, POST, PUT, DELETE and GETWITHBODY.
	httpHeaders	Use this variable to specify the API header parameter to send along with the API call.
	listField	Use this variable to specify the path of account objects in the response JSON or XML (received after making the http API call). If the path is a nested path, separate it with a dot (.), for example, "customproperty9": "address.home".
	keyField	Use this variable to specify the name of the field that uniquely defines whether the incoming account object is a new account or an existing account.
	statusConfig	Use this variable to map the status of accounts in IGA with the status in the target application.

Attributes	Variable	Description
	colsToPropsMap	<p>Use this variable to map account objects in the target application with account attributes in IGA, so that data from the target application is correctly saved.</p> <p>Review the following guidelines before mapping account attributes:</p> <ul style="list-style-type: none"> • Represent the logical mapping in this format: <AccountsTableAttribute in EIC>::<TargetapplicationAttribute>#<Parsabledatatypeinsaviynt> • Represent the target application attributes in the same case and format as they are in the target application. • If the field name in the target application contains a dot(.), use a ~dot# separator. For example, odata~dot#error. • If the list objects are at the top of the hierarchy in the response JSON, keep the field empty. • If mapping is mandatory for a field, for example, name, accountId, mention the name of the field in the keyField variable.

Attributes	Variable	Description
		<ul style="list-style-type: none"> If the path is a nested path, separate it with a dot (.) and to specify the mapping for the path, use an index square bracket, for example, <code>"customproperty9": "address.home[1].city"</code>. <div>  Note <p>The possible datatype values are char, date, bool, boolList, listAsString, and json. Use the boolList dataType as true to instruct the connector to return the result if the list contains any values, else it is false.</p> </div>
	Pagination	Use this variable to define the pagination parameters. For more information, see the Pagination feature in <code>ImportAccountEntJSON</code> .
	inputParams	Set the <code>dependentCall</code> variable as , if you want to import the metadata of an account in successive calls. The value passed in the <code>URL</code> variable of an API is called for each account. The value specified in the <code>nextApiKeyField</code> variable is bound in the <code>URL</code> variable of the successive API calls.

Attributes	Variable	Description
		<p>For example, if you want to get the metadata of account based on account ID, specify the value for the <code>nextApiKeyField</code> variable as <code>accountID</code>. The account ID of the account in the previous call is bound in the <code>URL</code> variable of the successive API calls.</p> <p>Set the <code>dependentCall</code> variable as <code>false</code>, if you do not want to mark successive calls to be dependent on the previous call.</p>
	<code>disableDeletedAccounts</code>	<p>This variable is no longer required after the introduction of the <code>statusAndThresholdConfig</code> variable. Set this to <code>false</code> in the current codebase.</p>
	<code>acctEntMappings</code>	<p>Use this variable to specify the mapping between accounts and entitlements if there is no separate API call made for entitlements import.</p>


Attributes	Variable	Description
		<div>  Note <ul style="list-style-type: none"> • Use <code>importAsEntitlement</code> to instruct the connector to import entitlements. When set to true, the connector imports associated entitlements. The default value is false. • Do not specify any value in <code>listPath</code>. • Do not specify any value in <code>idPath</code>. • Do not specify any value in <code>keyField</code>. • Use <code>colsToPropsMap</code> to map entitlement objects in the target application with entitlement attributes in IGA, so that data from the target application is correctly saved. Specify values in this variable, only if <code>importAsEntitlement</code> is set as true. • You must specify the following entry in <code>colsToPropsMap</code> of one of the accounts calls: <code>"customproperty31": "STORE#ACC#ENT#MAPPINGINFO~#~char"</code> • Use <code>statusConfig</code> to map the status of entitlements in IGA with the status in the target application. </div>

Attributes	Variable	Description
entitlementParams	Connection	Populates the connection type value specified in the acctAuth attribute while defining <code>ConnectionJSON</code> .
	processingType	Always specify the processing type as <code>SequentialAndIterative</code> .
	successResponses	Use this variable to specify the possible response messages on successfully importing entitlements.
	doNotChangelfFailed	Use this variable to instruct the connector to not delete or inactivate existing data on failure of the account API, the entType API, or the mapping API. When set to true, the connector does not delete or inactivate existing data.
	entTypes	Use this variable to specify all entitlement types to be imported.

Attributes	Variable	Description
	entTypeOrder	Each entitlement type can comprise of multiple API calls. Specify order of the entitlement type as 0 for the first entitlement type, specify as 1 for the second entitlement type, and so on.
	entTypeLabels	Use this variable to define the label for an EntitlementType property. For example, if you want to rename the customproperty3 to Tag specify "customproperty3": "Tags", under entTypeLabels.
	call	Use this variable to make an API call to the target application API for importing entitlements. The connector allows you to specify multiple API calls to import entitlements. You must specify each API call for importing entitlements as a separate call. There is no fixed name, but the preferred naming convention is: call1, call2, call3.
	callOrder	Use this variable to specify the order to call APIs. Specify the value for the first call as zero. The lowest order value is called the first call.

Attributes	Variable	Description
	stageNumber	Use this variable to specify the stage number for the API call. Specify the stage number as zero for the first API call.
	inputParams	<p>Set the dependentCall variable as true, if you want to import the metadata of an entitlement in successive calls. The value passed in the URL variable of an API is called for each entitlement. The value specified in the nextApiKeyField variable is bound in the URL variable of the successive API calls.</p> <p>For example, if you want to get the metadata of entitlement based on entitlement ID, specify the value for the nextApiKeyField variable as entitlementID. The entitlement ID of the entitlement in the previous call is bound in the URL variable of the successive API calls.</p> <p>Set the dependentCall variable as false, if you do not want to mark successive calls to be dependent on the previous call.</p>
	http	Use this variable to specify the http details required to make the API call, for example, URL, http params, headers, content-type, ssl parameters. For most cases, Authorization header value is \${access_token} which is

Attributes	Variable	Description
		a place holder and it is populated from the <code>accessToken</code> attribute in <code>ConnectionJSON</code> . Specify <code>contentType</code> as application/x-www-form-urlencoded, application/json, application/xml.
	<code>listField</code>	Use this variable to specify the path of entitlement objects in the response JSON or XML (received after making the http API call). If the path is a nested path, separate it with a dot (.), for example, "customproperty9": "address.home".
	<code>keyField</code>	Use this variable to specify the column name in IGA where the entitlement identification ID is saved.
	<code>statusConfig</code>	Use this variable to map the status of entitlements in IGA with the status in the target application.
	<code>colsToPropsMap</code>	Use this variable to map entitlement objects in the target application with entitlementaccount attributes in IGA, so that data from the target

Attributes	Variable	Description
		<p>application is correctly saved.</p> <p>Review the following guidelines before mapping entitlement attributes:</p> <ul style="list-style-type: none"> • Represent the logical mapping in this format: <EntitlementsTableAttribute in EIC>::<TargetapplicationAttribute>#<Parsabledatatypein saviynt> • Represent the target application attributes in the same case and format as they are in the target application. • If the field name in the target application contains a dot(.), use a ~dot# separator. For example, odata~dot#error. • If the list objects are at the top of the hierarchy in the response JSON, keep the field empty. • Instead of specifying the column name in status field under colstop rop, map the status field to a customproperty and add this value in STATUS_THRESHOLD_CONFIG. <div>  <p>Note</p> <p>The possible datatype values are char, date, bool, boolList, listAsString, and json. Use the boolList dataType as true to instruct the</p> </div>

Attributes	Variable	Description
		connector to return the result if the list contains any values, else it is false.
	Pagination	Use this variable to define the pagination parameters. For more information, see the Pagination feature in <code>ImportAccountEntJSON</code> .
	disableDeletedEntitlements	This variable is no longer required after the introduction of the <code>statusAndThresholdConfig</code> variable. Set this to false in the current codebase.
	makeProcessingStatus	Use this variable when there are multiple API calls involved for importing one entitlement type. In most cases, there is an API that gets all the entitlement objects of that type and other APIs might obtain additional data for each entitlement object. In this case, the status of the entitlement should be in process till all the details of that entitlement object is imported. Hence, makeProcessingStatus can be set to true for the first API call and false for the successive API call. The default value is false.

Attributes	Variable	Description
acctEntParams	createUsers	<p>Use this variable to instruct the connector to create users while importing accounts. When set to true, users are created during account import. The default value is false.</p> <p>adminName: If createUsers is , new users are imported and the manager is set as an admin as default. Select the user name if you want to add the user manager other than admin.</p>
	Connection	<p>Populates the connection type value specified in the acctAuth attribute while defining ConnectionJSON.</p>
	statusAndThresholdConfig	<p>Use this variable, if you are not specifying any value in the STATUS_THRESHOLD_CONFIG connection parameter.</p>
	statusColumn	<p>Specify the account attribute mapped with the status of the account. Use this variable, if you are not specifying any value in the statusColumn attribute of the STATUS_THRESHOLD_CONFIG connection parameter.</p>

Attributes	Variable	Description
	processingType	Always specify the processing type as SequentialAndIterative .
	successResponses	Use this variable to specify the possible response messages on successfully importing entitlements.
	entTypes	Use this variable to specify all entitlement types to be imported.
	call	Use this variable to make an API call to the target application API for importing entitlements. The connector allows you to specify multiple API calls to import entitlements. You must specify each API call for importing entitlements as a separate call. There is no fixed name, but the preferred naming convention is: call1, call2, call3.
	callOrder	Use this variable to specify the order to call APIs. Specify the value for the first call as zero. The lowest order value is called the first call.

Attributes	Variable	Description
	stageNumber	Use this variable to specify the stage number for the API call. Specify the stage number as zero for the first API call.
	http	Use this variable to specify the http details required to make the API call, for example, URL, http params, headers, content-type, ssl parameters. For most cases, Authorization header value is \${access_token} which is a place holder and it is populated from the <code>accessToken</code> attribute in <code>ConnectionJSON</code> . Specify <code>httpContentType</code> as application/x-www-form-urlencoded, application/json, application/xml.
	listField	Use this variable to specify the path of entitlement objects in the response JSON or XML (received after making the http API call). If the path is a nested path, separate it with a dot (.), for example, "customproperty9": "address.home".
	acctIdPath	Use this variable to define the path of the account ID in the response JSON.

Attributes	Variable	Description
	entIdPath	Use this variable to define the path of the entitlement ID in the response JSON.
	ownerIdPath	Use this variable to define the path of the owner of the entitlement in the response JSON.
	acctKeyField	Use this variable to define the key field that uniquely identifies the account.
	entKeyField	Use this variable to define the key field that uniquely identifies the entitlement value.
	ownerKeyField	Use this variable to define the key field that uniquely identifies the owner of the entitlement.

Attributes	Variable	Description
entOwnerParams	Connection	Populates the connection type value specified in the acctAuth attribute while defining <code>ConnectionJSON</code> .
	successResponses	Use this variable to specify the possible response messages on successfully importing entitlement owner.
	doNotChangelfFailed	Use this variable to instruct the connector to not delete or inactivate existing data on failure of the API. When set to true, the connector does not delete or inactivate existing data.
	entTypes	Use this variable to specify all entitlement types to be imported.
	callOrder	Use this variable to specify the order to call APIs. Specify the value for the first call as zero. The lowest order value is called the first call.

Attributes	Variable	Description
	stageNumber	Use this variable to specify the stage number for the API call. Specify the stage number as zero for the first API call.
	ownerKeyField	Use this variable to define the key field that uniquely identifies the owner of the entitlement.
	http	Use this variable to specify the http details required to make the API call, for example, URL, http params, headers, content-type, ssl parameters. For most cases, Authorization header value is <code>\${access_token}</code> which is a place holder and it is populated from the <code>accessToken</code> attribute in <code>ConnectionJSON</code> . Specify <code>httpContentType</code> as <code>application/x-www-form-urlencoded</code> , <code>application/json</code> , <code>application/xml</code> .
	listField	Use this variable to specify the path of entitlement objects in the response JSON or XML (received after making the http API call). If the path is a nested path, separate it with a dot (<code>.</code>), for example, <code>"customproperty9": "address.home"</code> .

Attributes	Variable	Description
	acctIdPath	Use this variable to define the path of the account ID in the response JSON.
	entIdPath	Use this variable to define the path of the entitlement ID in the response JSON.
	ownerIdPath	Use this variable to define the path of the owner of the entitlement in the response JSON.
	acctKeyField	Use this variable to define the key field that uniquely identifies the account.
	entKeyField	Use this variable to define the key field that uniquely identifies the entitlement value.

Attributes	Variable	Description
entMappingParams	Connection	Populates the connection type value specified in the acctAuth attribute while defining <code>ConnectionJSON</code> .
	callOrder	Use this variable to specify the order to call APIs. Specify the value for the first call as zero. The lowest order value is called the first call.
	stageNumber	Use this variable to specify the stage number for the API call. Specify the stage number as zero for the first API call.
	http	Use this variable to specify the http details required to make the API call, for example, URL, http params, headers, content-type, ssl parameters. For most cases, Authorization header value is <code>\${access_token}</code> which is a place holder and it is populated from the <code>accessToken</code> attribute in <code>ConnectionJSON</code> . Specify <code>httpContentType</code> as application/x-www-form-urlencoded, application/json, application/xml.
	listField	Use this variable to specify the path of entitlement objects in the response JSON or XML (received after making the http API call). If the

Attributes	Variable	Description
		path is a nested path, separate it with a dot (.), for example, "customproperty9": "address.home".
	ent1IdPath	Use this variable to define the path of the parent entitlement ID in the response JSON.
	ent2IdPath	Use this variable to define the path of the child entitlement ID in the response JSON.
	ent1KeyField	Use this variable to define the key field that uniquely identifies the entitlement value of the parent entitlement.
	ent2KeyField	Use this variable to define the key field that uniquely identifies the entitlement value of the child entitlement.
	targetEntType	Use this variable to reference the name of child entitlement type for mapping.

Attributes	Variable	Description
	mappingTypes	<p>Use this variable to define the type of mapping to be used. The possible values are:</p> <ul style="list-style-type: none"> • ENTMAP - enables bi-directional mapping through entitlementmap • ENT2 - makes the target entitlement type as a child of current entitlement.

ImportUserJSON

This parameter is used to to map user attributes of the target application to user attributes of IGA for user import.



Note

Username should be unique for importing users.

Example 1: To define this parameter, use a format similar to the following:

JSON

```
{
  "connection": "acctAuth",
  "successResponses": {
    "statusCode": [
      200,
      201,
      202,
      203,
      204,
      205
    ]
  },
  "url": "https://abc.zendesk.com/api//users.json",
  "httpMethod": "GET",
  "httpHeaders": {
    "Authorization": "${access_token}"
  },
  "userResponsePath": "users",
  "colsToPropsMap": {
    "username": "id~#~char",
    "systemUserName": "id~#~char",
    "displayname": "name~#~char",
  }
}
```

```
    "email": "email~#~char"
  }
}
```

Common Features

1. Support for Multiple API calls

Example:

JSON

```
{
  "type": "multiCall",
  "call": [
    {
      "name": "call1",
      "connection": "acctAuth",
      "url": "https://<url>/api/v2/users.json",
      "httpMethod": "GET",
      "httpHeaders": {
        "Content-Type": "application/json",
        "Authorization": "${access_token}"
      },

```

```

    "colsToPropsMap": {
      "username": "id~#~char",
      "customproperty1": "email~#~char"
    },
    "userResponsePath": "users"
  },
  {
    "name": "call2",
    "connection": "acctAuth",
    "url": "https://<url>/api/v2/users/${userIdentifier}.json",
    "httpMethod": "GET",
    "httpHeaders": {
      "Authorization": "${access_token}"
    },
    "colsToPropsMap": {
      "username": "user.id~#~char",
      "customproperty45": "user.email~#~char"
    },
    "userResponsePath": ""
  }
]
}

```

2. **Support for importing the status information** of users to IGA using the **statusConfig** attribute, use a format similar to the following:

Example:

JSON

```
{
  "connection": "acctAuth",
  "url": "<<https://<domain name>/api/v2/users.json",
  "httpMethod": "GET",
  "httpHeaders": {
    "Authorization": "${access_token}"
  },
  "userResponsePath": "users",
  "colsToPropsMap": {
    "username": "id~#~char",
    "systemUserName": "id~#~char",
    "displayname": "name~#~char",
    "email": "email~#~char",
    "statuskey": "suspended~#~char"
  },
  "statusConfig": {
    "active": "false",
```

```

    "inactive": "true"
  },
  "pagination": {
    "nextUrl": {
      "nextUrlPath": "${response.completeResponseMap.next_page==null?'' : response.completeResponseMap.next_
    }
  }
}

```

3. **Support for storing the objectList data** based on the condition specified in the **colsToPropsMap** attribute for user, account, and access import using **#CONST#** in the JSON .

Example:

JSON

```

{
  "connection": "acctAuth",
  "url": "<specify URL>",
  "httpMethod": "GET",
  "httpHeaders": {
    "Accept": "application/json"
  },

```

```

    "userResponsePath": "d.results",
    "colsToPropsMap": {
        "username": "username",
        "systemUserName": "username~#~char",
        "email": "#CONST#${List responseList = response.empInfo.personNav.emailNav.results; int count = 0; int i = 0; while (i < responseList.size()) { if (responseList.get(i).email != null) { email = responseList.get(i).email; count++; } i++; } }",
        "customproperty15": "#CONST#${List responseList = response.empInfo.personNav.emailNav.results; int count = 0; int i = 0; while (i < responseList.size()) { if (responseList.get(i).email != null) { email = responseList.get(i).email; count++; } i++; } }",
        "customproperty13": "#CONST#Internal~#~char"
    }
}

```

4. Support for GETWITHBODY method :

Example:

JSON

```

{
    "connection": "acctAuth",
    "successResponses": {
        "statusCode": [
            200,
            201,
            202,

```

```

        203,
        204,
        205
    ]
},
"url": "https://<domain name>/api/v2/users.json",
"httpParams": "{\"UseGlobalCatalog\":\"false\", \"SearchRoot\": \"OU=Windows10,OU=Root2,DC=vstage,DC=co\"}",
"httpMethod": "GETWITHBODY",
"httpHeaders": {
    "Authorization": "${access_token}"
},
"userResponsePath": "users",
"colsToPropsMap": {
    "username": "id~#~char",
    "systemUserName": "id~#~char",
    "displayname": "name~#~char",
    "email": "email~#~char"
}
}

```

5. Support for sorting the objectList data in ascending order based on date value specified in the datePath variable

Use sortDataByConfig attribute to sort data.

Example:

JSON


```
{
  "connection": "acctAuth",
  "url": "<URL>",
  "httpMethod": "GET",
  "httpHeaders": {
    "Accept": "application/json"
  },
  "sortDataByConfig": {
    "dateFormat": "yyyy-MM-dd'T'HH:mm:ss",
    "datePath": "startDate"
  },
  "userResponsePath": "records",
  "colsToPropsMap": {
    "username": "Name~#~char",
    "systemUserName": "Name~#~char",
    "displayname": "Name~#~char",
    "email": "Email~#~char",
    "firstname": "FirstName~#~char",
    "lastname": "LastName~#~char",
```


```


    "statuskey": "IsActive~#~char"
  }
}

```

The following table describes attributes in `ImportUserJSON`:

Attributes	Description
Connection	Populates the connection type value specified in the acctAuth attribute while defining <code>ConnectionJSON</code> .
url	Use this attribute to specify the URL of the target application.
httpMethod	<p>Use this attribute to specify the API call method for a successful API call such as GET, POST, PUT, DELETE and GETWITHBODY.</p> <div>  Note </div> <p>Use GETWITHBODY method if a body has to be passed in a GET call.</p>
httpHeaders	Use this attribute to specify the API header parameter to send along with the API call.

Attributes	Description
colsToPropsMap	<p data-bbox="555 336 1883 421">Use this variable to map user objects in the target application with user attributes in IGA, so that data from the target application is correctly saved.</p> <div data-bbox="566 475 622 531"></div> <p data-bbox="645 491 712 523">Note</p> <p data-bbox="577 579 1597 611">In the above example, username is a mandatory field to save the user in IGA.</p> <p data-bbox="555 719 1395 751">Review the following guidelines before mapping user attributes:</p> <ul data-bbox="607 818 1939 1217" style="list-style-type: none"> • Represent the logical mapping in this format: • Represent the target application attributes in the same case and format as they are in the target application. • If the field name in the target application contains a dot(.), use a ~dot# separator. For example, • If the list objects are at the top of the hierarchy in the response JSON, keep the field empty. • If the path is a nested path, separate it with a dot (.) and to specify the mapping for the path, use an index square bracket, for example, .

Attributes	Description
	<div>  Note </div> <p>The possible datatype values are char, date, bool, boolList, listAsString, and json. Use the boolList dataType as true to instruct the connector to return the result if the list contains any values, else it is false.</p>
contentType	Use this attribute to specify the type of calls defined in httpHeaders.
Authorization	Use this attribute to specify the necessary authorization to call the webservice.
errorCode	Use this attribute to specify the error code number used to identify the failure.
errorCodePath	Use this attribute to specify the path in which the errorCode is stored.
userResponsePath	<p>Use this attribute to fetch the path in which the list of users are stored in the JSON response.</p> <p>For example, if the JSON response is <code>{"response": {"users": [{"name": "foo"}]}}</code>, the <code>userResponsePath</code> value is <code>.</code></p>

Attributes	Description
statusConfig	Use this attribute to map the user status (active or inactive) while importing users from the target application. You can also specify the JSON in UserImportJob. For more information, see Data Jobs in the <i>Enterprise Identity Cloud Administration Guide</i> .
Pagination	Use this attribute to define the pagination parameters. For more information, see Pagination feature in ImportAccountEntJSON .

Provisioning JSONs

REST Connector supports provisioning of account and entitlements to the target application using provisioning JSONs.

1. Binding response values from one call to another

The REST connector uses response values of one call as input parameter into another call for multiple calls. In the following example, the first API call gets descriptor ID for the requested account and the value of the descriptor ID is used as input in the next API call.

JSON

```
{
  "accountIdPath": "call1.message.id",
```

```
"dateFormat": "yyyy-MM-dd'T'HH:mm:ssXXX",
"call": [
  {
    "name": "call1",
    "connection": "userAuth",
    "url": "https://<domain name>/2.0/users",
    "httpMethod": "POST",
    "httpParams": "{\\"login\\":\\"${user.username}\\",\\"name\\":\\"${user.firstname}\\\"}",
    "httpHeaders": {
      "Authorization": "${access_token}"
    },
    "contentType": "application/json",
    "successResponses": {
      "statusCode": [
        201
      ]
    }
  },
  {
    "name": "call2",
    "connection": "userAuth",
    "url": "https://<domain name>/2.0/folders",
```

```

    "httpMethod": "POST",
    "httpParams": "{ \"name\": \"${response.call1.message.name}_${response.call1.message.id}\" , \"parent\": \"${response.call1.message.parent}\" }",
    "httpHeaders": {
        "Authorization": "${access_token}"
    },
    "httpContentType": "application/json",
    "successResponses": {
        "statusCode": [
            201
        ]
    }
}
]
}

```

2. Completion or Failure of tasks based on status code and response message

The tasks are marked as completed or failed based on the response status code of the API call.

JSON

```

{
  "call": [
    {

```

```
"name": "call1",
"connection": "userAuth",
"url": "https://<domain name>/users/${account.accountID}?notify=true&force=true",
"httpMethod": "POST",
"httpHeaders": {
  "Authorization": "${access_token}",
  "Accept": "application/json"
},
"httpContentType": "application/json",
"successResponses": {
  "statusCode": [
    201,
    200
  ]
},
"unsuccessResponses": {
  "statusCode": [
    500,
    403,
    409
  ]
}
```



```
}  
]  
}
```

There are a few applications where you get same status code for both successful and unsuccessful API calls. In that case, you can handle the tasks status in IGA based on the API response body content.

For example, If the response body of API call is in below format :

```
{ "Objectreturned": "Success", "Length": "7" }
```

Use the following format to use response body content for successful or unsuccessful tasks.

JSON

```
{  
  "httpContentType": "application/json",  
  "successResponses": {  
    "statusCode": [  
      200  
    ]  
  },  
  "unsuccessResponses": {  
    "response.StatusCode": "1"  
  }  
}
```

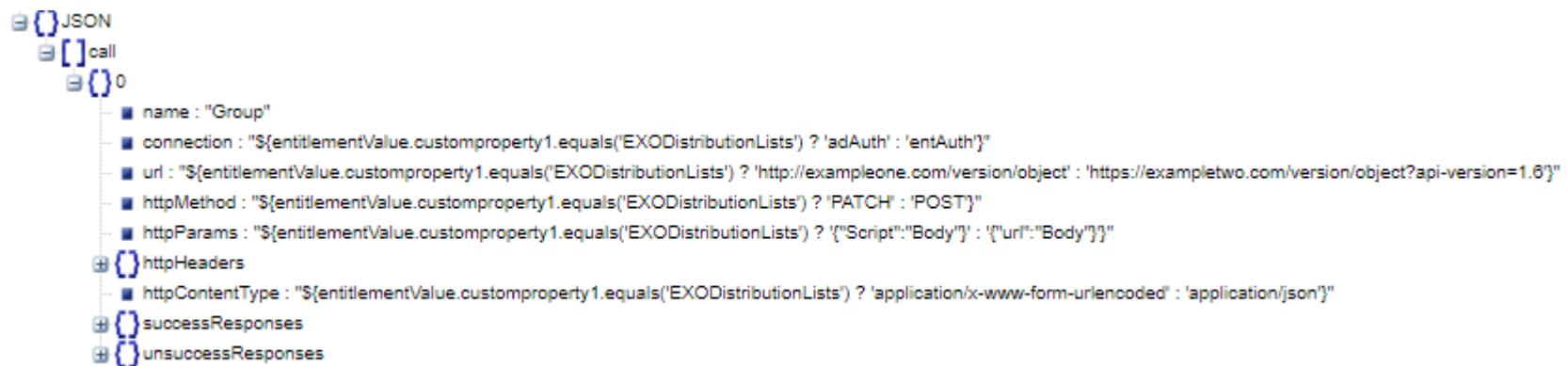
```
}  
  
}
```

3. Java Script Operations on JSON attribute values

In the REST connector provisioning JSONs, javascript operations such as ternary conditions, FOR loop, string operations are applied on attribute values as per requirement. For example, “Roles”: [\$

{response.call1.message.roles.toString().replace('[', '').replace(']', '').replace(',', '\', '\')}]

Example for JSON with ternary conditions:



Example for JSON with string operations:

JSON

```

{
  "name": "call1",
  "connection": "userAuth",
  "url": "${requestAccessAttributes.addRemoveAlias==null?'https://<domain name>/admin/directory/v1/use
  "httpMethod": "${requestAccessAttributes.addRemoveAlias==null?'PUT':account.customproperty30.toStrin
  "httpParams": "${requestAccessAttributes.addRemoveAlias==null?'\{"name\":{\\"familyName\\":\\"'+user.fi
  "httpHeaders": {
    "Authorization": "${access_token}",
    "Content-Type": "application/json"
  },
  "httpContentType": "application/json",
  "successResponses": {
    "statusCode": [
      200,
      201,
      2014
    ]
  }
}

```

Example for JSON with FOR Loop:

JSON

```
{
  "call": [
    {
      "name": "Group",
      "connection": "userAuth",
      "url": "https://<domain name>/sys_user_grmember?user=${account.accountID}",
      "httpMethod": "GET",
      "httpHeaders": {
        "Authorization": "${access_token}"
      },
      "httpContentType": "application/json",
      "successResponses": {
        "statusCode": [
          200
        ]
      },
      "unsuccessResponses": {
        "statusCode": [
          400
        ]
      }
    }
  ]
}
```

```

    },
    {
      "name": "Group",
      "connection": "userAuth",
      "url": "https://<domain name>/sys_user_grmember/${for (Map map : response.Group1.message.result){if
      "httpMethod": "DELETE",
      "httpHeaders": {
        "Authorization": "${access_token}"
      },
      "httpContentType": "application/json",
      "successResponses": {
        "statusCode": [
          204
        ]
      },
      "unsuccessResponses": {
        "statusCode": [
          400
        ]
      }
    }
  }
}

```

```
]
}
```

4. Bind special characters in attributes



Info

Available from Release v23.6 onwards

To correctly bind any special characters, such as backslash (\), double quotes ("), or dollar (\$), in attributes during provisioning, use the replace logic as shown below in the provisioning JSON:

JSON

```
{
  "accountIdPath": "call1.message.user.id",
  "dateFormat": "yyyy-MM-dd'T'HH:mm:ssXXX",
  "responseColsToPropsMap": {
    "displayname": "call1.message.user.name~#~char"
  },
  "call": [
    {
      "name": "call1",
```

```

    "connection": "acctAuth",
    "url": "https://saviyntdevops.zendesk.com/api/v2/users",
    "httpMethod": "POST",
    "httpParams": "{ \"user\": { \"name\": \"${user.firstname} ${user.lastname}\", \"password\": \"${password}\"",
    "httpHeaders": {
      "Authorization": "${access_token}",
      "Accept": "application/json"
    },
    "httpContentType": "application/json"
  }
]
}

```

Attributes exposed per binding object:

- **user:** username, firstname, preferredFirstName, lastname, middlename, street, city, comments, statuskey, startdate, enddate, manager, password, location, jobCode, jobDescription, employeeType, systemUserName, departmentNumber, title, state, companyname, costcenter, departmentname, employeeclass, entity, jobcodedesc, locationdesc, locationnumber, siteid, orgunitid, region, regioncode, owner, employeeid, lastsyncdate, createdate, email, phonenumber, job_function, country, displayname, locale, customproperty1 -customproperty65, secondaryPhone, secondaryEmail, leaveStatus, riskscore, userSource, userSourceKey, jobID, savUpdateDate.

- **account:** name, updatedate, updateUser, systemid, accountID, saviyntConnectJobId, displayName, Accounts, referenced_accountkey, referenced_accountName, description, comments, accounttype, accountclass, usergroup, validfrom, validthrough, userlock, incorrectlogons, creator, createdon, lastlogondate, lastpasswordchange, passwordlockdate, jobid, privileged, passwordchangestatus, customproperty1 -customproperty56.
- **userAccount:** userkey, accountkey, updateuser, lastCertifiedCampaignName, lastCertifiedCampaignDate, lastCertifiedUser.
- **arsTasks:** tasktype, taskdate, status, password, ownerType, ownerkey, comments, accountName, upadteuser, requestedBy, users, ArsTasks, parenttask, provisioningMetadata, provisioningComments, startDate, endDate, Roles, assignedFromRole, assignedFromRule, assignedFromRoles, ticketId, source, sourceId, provisioningTries, savconattrs.
- **requestAccessAttributes:** Dynamic Attribute values created in Endpoint.
- **endpoints:** endpointname, description, customproperty1-customproperty45, customproperty1Label-customproperty56Label, accountNameRule, entsWithNewAccount, ownerType, ownerkey, requestowner, requestownertype, accessquery, accountTypeNoDeprovision, accountTypeNoPasswordChange, userAccountCorrelationRule, accountTypeForServiceAccount, primaryAccountType, createEntTaskforRemoveAcc, allowRemoveAllRoleOnRequest, serviceAccountNameRule, serviceAccountAccessQuery, transients, parentAccountPattern, displayName, status, roleTypeAsJson, applicationLogo, applicationUrl, disableaccountrequest, enableCopyAccess, lastImport, createDate, updateDate, createdBy, updatedBy, createdFrom, accountNameValidatorRegex, jrmDataPopulated, taskemailtemplates.
- **entitlementValue:** entitlement_value, description, displayname, entitlementID, program, customproperty1-customproperty40, entitlementMappingJson, entitlement_glossary, status, updateuser, risk, soxcritical, syscritical, createdate, updatedate, orphan, level, privileged, confidentiality, ent_objHash, roletype, module, access, schemaname, entclass, lastScanDate, jobId, saviyntConnectJobId, saviyntConnectUpdateDate, acctEntMappingInfoColumnFromEnt, priority.

- **account_entitlements**: entitlement_valuekey, accountkey, updatedate, updateuser, startdate, enddate, lastusedenddate, access, arsTask, assignedfromchild, assignedFromCompRole, assignedFromRole, assignedFromRule, assignedFromRoles, jobId, jrmRules, lastCertifiedCampaignName, lastCertifiedCampaignDate, lastCertifiedUser, saviyntConnectJobId, saviyntConnectUpdateDate, uuid

CreateAccountJSON

This parameter is used to create accounts in the target application. You must populate `accountIdPath` in the `CreateAccountJSON` parameter for successful completion of task.



Note

In case of null response, bind account name with `accountIdPath`. For example, `"accountIdPath": "accountName"`.

Example 1: To hide or show sensitive data received as a response from target applications using `showResponse` attribute, use a format similar to the following:

JSON

```
{
  "accountIdPath": "call1.message.id",
  "dateFormat": "yyyy-MM-dd'T'HH:mm:ssXXX",
  "responseColsToPropsMap": {
```

```

"name": "call1.message.name~#~char",
"displayname": "call1.message.name~#~char",
"updatedate": "call1.message.modified_at~#~date",
"comments": "call1.message.login~#~char"
},
"call": [
{
"name": "call1",
"connection": "userAuth",
"showResponse": false,
"url": "https://{org}/2.0/users",
"httpMethod": "POST",
"httpParams": "{\"login\": \"${user.email}\", \"name\": \"${user.firstname}\", \"lastname\": \"${userAccou",
"httpHeaders": {
"Authorization": "${access_token}"
},
"httpContentType": "application/json",
"successResponses": [
{
"message": "ProfileID Created Successfully"
},
{

```

```

        "message": "ProfileID Updated Successfully"
      }
    ]
  }
}

```

Example 2: To utilize any dynamic attribute in the request and to obtain that value in the newly created account or updated account, use a format similar to the following:

JSON

```

{
  "accountIdPath": "call1.message.ResourcePartyNumber",
  "call": [
    {
      "name": "call1",
      "connection": "userAuth",
      "url": "<<URL>",
      "httpMethod": "POST",
      "httpParams": "{\"FirstName\": \"${user.firstname==null? '' : user.firstname}\", \"LastName\": \"${user.lastname==null? '' : user.lastname}\"}",
      "httpHeaders": {
        "Authorization": "${access_token}"
      }
    }
  ]
}

```

```

    },
    "httpContentType": "application/vnd.oracle.adf.resourceitem+json",
    "successResponses": [],
    "unsuccessResponses": {
      "statusCode": [
        400,
        401,
        404,
        405,
        500
      ]
    }
  }
}

```

Example 3: To define the content of email templates, specify the details in `httpParams`, in a format similar to the following:

JSON

```

{
  "accountIdPath": "call1.message.user.id",
  "dateFormat": "yyyy-MM-dd'T'HH:mm:ssXXX",

```

```

"responseColsToPropsMap": {
  "displayName": "call1.message.user.name~#~char"
},
"call": [
  {
    "name": "call1",
    "connection": "acctAuth",
    "url": "https://<zendesk URL>/api/v2/users",
    "httpMethod": "POST",
    "httpParams": "{\"user\": {\"name\": \"${user.firstname} ${user.firstname}\", \"email\": \"${user.email}\"",
    "httpHeaders": {
      "Authorization": "${access_token}",
      "Accept": "application/json"
    },
    "httpContentType": "application/json",
    "successResponses": {
      "statusCode": [
        201,
        200
      ]
    }
  }
]
}

```

```
]
}
```

Example 4: To map the plain text responses of APIs to any attribute by prefixing its value with #CONST#, use a format similar to the following:

JSON

```
{
  "accountIdPath": "#CONST#{String acctId = response.call1.message.responseMessage; acctId = acctId.substring(1, acctId.length-1)}",
  "responseColsToPropsMap": {
    "name": "#CONST#{String acctId = response.call1.message.responseMessage; acctId = acctId.substring(1, acctId.length-1)}",
    "accounttype": "#CONST#Internal~#~char"
  },
  "call": [
    {
      "name": "call1",
      "connection": "acctAuth",
      "url": "<URL>",
      "httpMethod": "POST",
      "httpParams": "{\"user\": {\"name\": \"${user.firstname} ${user.lastname}\", \"email\": \"${user.email}\"}",
      "httpHeaders": {
        "Authorization": "${access_token}",

```

```

    "Accept": "application/json"
  },
  "httpContentType": "application/json",
  "successResponses": {
    "statusCode": [
      200,
      201
    ]
  }
}
]
}

```

Example 5: To support iterating over the response of a call to perform remediation for multiple objects in subsequent calls during provisioning via two new attributes named `callCondition` and `callListPath`, use a format similar to the following:

JSON

```

{
  "accountIdPath": "call2.message.user.id",
  "dateFormat": "yyyy-MM-dd'T'HH:mm:ssXXX",
  "responseColsToPropsMap": {
    "displayName": "call2.message.user.name~#~char"
  }
}

```

```
},
"call": [
  {
    "name": "call1",
    "connection": "acctAuth",
    "url": "https://<domain name>/api/v2/users",
    "httpMethod": "GET",
    "httpHeaders": {
      "Authorization": "${access_token}",
      "Accept": "application/json"
    },
    "httpContentType": "application/json",
    "successResponses": {
      "statusCode": [
        200,
        201
      ]
    }
  },
  {
    "name": "call2",
    "connection": "acctAuth",
```



```

    "url": "https://<domain name>/api/v2/users/${childresponse.id}",
    "httpMethod": "POST",
    "httpParams": "{\"user\": {\"name\": \"${user.firstname} ${user.lastname}\", \"email\": \"${user.email}\"",
    "httpHeaders": {
        "Authorization": "${access_token}",
        "Accept": "application/json"
    },
    "httpContentType": "application/json",
    "callCondition": "${response.call1.message.users.size()}>0",
    "callListPath": "response.call1.message.users",
    "successResponses": {
        "statusCode": [
            200,
            201
        ]
    }
}
]
}

```

Example 6: To support execution of powershell scripts to revoke O365 session, specify the values in the url and httpParams attributes, use a format similar to the following:

JSON

```
{
  "accountIdPath": "accountName",
  "responseColsToPropsMap": {},
  "call": [
    {
      "name": "call1",
      "connection": "acctAuth",
      "url": "http://<hostname>/FIMDemo/PS/ExecutePSScript",
      "httpMethod": "POST",
      "httpParams": "{\"Script\": \"cd C:\\\\\\\\\\\\\\\\\\\\scripts;.\\\\\\\\\\\\\\\\\\\\PSScript.ps1\"}",
      "httpHeaders": {
        "Authorization": ""
      },
      "httpContentType": "application/x-www-form-urlencoded",
      "successResponses": {
        "statusCode": [
          200,
          201,
          204,
          205
        ]
      }
    }
  ]
}
```

```
    }  
  }  
]  
}
```

Example 7: To encrypt the value passed in the customproperty, use the following format in the provisioning JSONs:

For example, `user.encryptedCp1`

JSON

```
{  
  "accountIdPath": "call1.message.user.id",  
  "dateFormat": "yyyy-MM-dd'T'HH:mm:ssXXX",  
  "responseColsToPropsMap": {  
    "displayName": "call1.message.user.name~#~char"  
  },  
  "call": [  
    {  
      "name": "call1",  
      "connection": "acctAuth",  
      "url": "https://xxx.zendesk.com/api/v2/users",  
      "httpMethod": "POST",  
    }  
  ]  
}
```

```


    "httpParams": "{ \"user\": { \"value\": \"${user.encryptedCp1}\", \"email\": \"${user.email}\", \"role\": \"\" } }",
    "httpHeaders": {
      "Authorization": "${access_token}",
      "Accept": "application/json"
    },
    "httpContentType": "application/json",
    "successResponses": {
      "statusCode": [
        201
      ]
    }
  }
]
}


```


The following table describes attributes in `CreateAccountJSON`:

Attribute	Description
accountIdPath	This is a mandatory attribute populated with a unique key from the response of the create user API call. In case of null response, bind the account name with <code>accountIdPath</code> . For example,

Attribute	Description
responseColsToPropsMap	Use this attribute to map response attributes to account column in EIC. This is an optional attribute.
connection	Populates the connection type value specified in the acctAuth attribute while defining <code>ConnectionJSON</code> .
showResponse	Use this attribute to instruct the connector to hide or show sensitive data received as a response from target applications. When set to true, the provisioning comments are shown in the Provisioning Comments field of a pending task. This helps to troubleshoot errors, if any.
httpHeaders	Use this attribute to specify the API header parameter to send along with the API call. For most cases, Authorization header value is <code>\${access_token}</code> , which is a place holder and it is populated from the <code>accessToken</code> attribute in <code>ConnectionJSON</code> .
ContentType	Use this attribute to specify the type of calls defined in httpHeaders.

Attribute	Description
successResponses	Use this attribute to specify the possible response messages on successfully creating accounts.
httpParams	<p>Use this attribute to define variables for mapping target application with IGA. For example, "profile":{"firstName":"\${user.firstname==null? "" : user.firstname}". Here, firstname is an object in IGA, firstName is the corresponding object in the target application, and profile is the section or path in which the target application object is available.</p> <p>Examples of some binding variables used in provisioning JSONS:</p> <ul style="list-style-type: none"> • <code>\${userAccount.get('endpoint').accountsproperties}</code>: Use this variable to bind a user to the account. Specify the endpoint name along with the accountproperty. Example to fetch lastname from the SAP endpoint with the accountproperty as: 'name'. "lastname": "\${userAccount.get('SAP').name}" <div data-bbox="734 973 1946 1184" data-label="Complex-Block">  <p>Note</p> <p>There is no metadata configured for this macro which means no configuration screen can be presented here.</p> </div> <ul style="list-style-type: none"> • <code>\$requestAccessAttributes</code>: Use this attribute to utilize any dynamic attribute in the request and to obtain that value in the newly created account or updated account. This is an optional variable.

Attribute	Description
	<p>Syntax:</p> <pre>"\${requestAccessAttributes.get('Customproperty1')}</pre> <ul style="list-style-type: none"> • `\${password}`: Use this binding variable to generate a random password based on the password policy. The same password is populated in an email template with a different binding object, `\${randompassword}`.
callCondition	<p>Use this attribute in the following scenarios:</p> <ul style="list-style-type: none"> • If a provisioning parameter includes multiple calls, each call is invoked only if the condition provided in this attribute is true. Otherwise, the call is skipped. • If the response of a call contains a list, the call iterates into the list. <div>  Note <p>This attribute is supported only for the CreateAccountJSON parameter.</p> </div>
callListPath	<p>Use this attribute to specify the path to the list in the response.</p>

Attribute	Description
	<div>  Note </div> <p>Use the <code>\${childresponse}</code> binding variable in url to specify the binding variable used to store the required response attribute after iterating into the list and to make multiple calls.</p>

UpdateAccountJSON

This parameter is used to update account attributes in the target application.

Example 1: To add arsTasks objects, use a format similar to the following:

You can use arsTasks objects such as arsTasks.tasktype, arsTasks.id, arsTasks.ArsTasksDomainProperties to fetch data while provisioning and deprovisioning accounts.

JSON

```
{
  "call": [
    {
      "name": "call1",
      "connection": "userAuth",
      "url": "https://{org} /2.0/users",
```



```

    "httpMethod": "POST",
    "httpParams": "{ \"login\": \"${user.email}\", \"name\": \"${user.firstname}\", \"taskid\": \"${arsTo
    \"httpHeaders\": {
      \"Authorization\": \"${access_token}\"
    },
    \"httpContentType\": \"application/json\",
    \"successResponses\": [
      {
        \"message\": \"ProfileID Created Successfully\"
      },
      {
        \"message\": \"ProfileID Updated Successfully\"
      }
    ]
  }
}

```

Example 2: To use service account owner objects, use a format similar to the following:

JSON

```
{
  "call": [
    {
      "name": "call1",
      "connection": "acctAuth",
      "url": "https://<domain name>/api/v2/users/${account.accountID}",
      "httpMethod": "POST",
      "httpParams": "{\"user\": {\"name\": \"${ServiceAccountOwnerMap.USEROWNERS.get('1').collect{it.firstname}\"",
      "httpHeaders": {
        "Authorization": "${access_token}",
        "Accept": "application/json"
      },
      "httpContentType": "application/json"
    }
  ]
}
```

You can also use the following binding variables:

USEROWNER NAMES

Example: `"httpParams": "{\"user\": {\"name\": \"${ServiceAccountOwnerMap.USEROWNER NAMES.get('1')}\"}}"`

USERGROUPOWNERS

Example: `"httpParams": {"user": {"name": "${ServiceAccountOwnerMap.USERGROUPOWNERS.get('1').collect{it.user_groupname}.join(',')}"}}`

USERGROUPOWNERNAMES

Example: `"httpParams": {"user": {"name": "${ServiceAccountOwnerMap.USERGROUPOWNERNAMES.get('1')}"}}`

ServiceAccountType - ServiceAccountFlag

Example: `"httpParams": {"user": {"name": "${ServiceAccountOwnerMap.ServiceAccountType}-${ServiceAccountOwnerMap.ServiceAccountFlag}"}}`

EnableAccountJSON

This parameter is used to enable an inactive account in the target application.

Example 1: To define this parameter, use a format similar to the following:

JSON

```
{
  "call": [
    {
      "name": "call1",
      "connection": "acctAuth",
    }
  ]
}
```

```
{
  "url": "aHOSTNAMEa/api/v2/users/${account.accountID}",
  "httpMethod": "PUT",
  "httpParams": "{\"user\":{\"suspended\": \"false\"}}",
  "httpHeaders": {
    "Authorization": "${access_token}",
    "Accept": "application/json"
  },
  "httpContentType": "application/json",
  "successResponses": {
    "statusCode": [
      200,
      201
    ]
  }
}
```

Example 2: To use service account owner objects, use a format similar to the following:

JSON

```

{
  "call": [
    {
      "name": "call1",
      "connection": "acctAuth",
      "url": "https://<domain name>/api/v2/users/${account.accountID}",
      "httpMethod": "PUT",
      "httpParams": "{\"user\":{\"name\": \"${ServiceAccountOwnerMap.ServiceAccountType}-${ServiceAccountOwnerName}\"}",
      "httpHeaders": {
        "Authorization": "${access_token}",
        "Accept": "application/json"
      },
      "httpContentType": "application/json"
    }
  ]
}

```

DisableAccountJSON

This parameter is used to disable the account in target application and mark the status as disabled.

Example 1: To define this parameter, use a format similar to the following:

JSON

```
{
  "call": [
    {
      "name": "call1",
      "connection": "userAuth",
      "url": "https://{org} /2.0/3827509470",
      "httpMethod": "DELETE",
      "httpHeaders": {
        "Authorization": "${access_token}"
      },
      "httpContentType": "application/json"
    }
  ]
}
```

Example 2: To use service account owner object, use a format similar to the following:

JSON

```
{
  "actions": {
    "Disable User": {
```

```
"call": [  
  {  
    "name": "call1",  
    "connection": "userAuth",  
    "url": "https://{org}/2.0/users/${account.accountID}",  
    "httpMethod": "PUT",  
    "httpParams": "{\"status\": \"inactive\"}",  
    "httpHeaders": {  
      "Authorization": "${access_token}"  
    },  
    "httpContentType": "application/json",  
    "successResponses": [  
      {  
        "message": "success"  
      }  
    ]  
  }  
]
```

AddAccessJSON

This parameter is used to assign entitlements to the account.

Example 1: To define this parameter, use a format similar to the following:

JSON

```
{
  "call": [
    {
      "name": "Group",
      "connection": "acctAuth",
      "url": "aHOSTNAMEa/api/v2/group_memberships",
      "httpMethod": "POST",
      "httpParams": "{ \"group_membership\": { \"user_id\": \"${account.accountID}\", \"group_id\": \"${entitlement}\" } }",
      "httpHeaders": {
        "Authorization": "${access_token}",
        "Accept": "application/json"
      },
      "httpContentType": "application/json",
      "successResponses": {
        "statusCode": [
          200,

```



```
    201
  ]
}
}
]
```

Example 2: To specify privilege using the `requestAccessAttributes` binding variable, use a format similar to the following:

JSON

```
{
  "call": [
    {
      "name": "<Name>",
      "connection": "acctAuth",
      "url": "<URL>/${entitlementValue.entitlementID}/users/${requestAccessAttributes.testprivilege}",
      "httpMethod": "PUT",
      "httpHeaders": {
        "Authorization": "${access_token}",
        "Accept": "application/json"
      },
      "httpContentType": "application/json",
```

```
    "successResponses": {  
      "statusCode": [  
        200,  
        201,  
        203,  
        204  
      ],  
      "status": "ACTIVE"  
    },  
    "unsuccessResponses": {  
      "statusCode": [  
        404,  
        401,  
        400  
      ]  
    }  
  }  
]  
}
```

here,

`${requestAccessAttributes.testprivilege}`: Use the binding variable to get the privilege details. Privileges can only be provisioned using the ARS module but not by defining rules. In the above example, testprivilege is the name of the privilege.

Example 3: To include both group and role entitlement types in separate calls, specify the entitlement type in the `name` attribute in a format similar to the following:



Note

All the calls for an entitlement type are defined in sequential order. In multiple call scenarios, if the response values are needed to pass to the successor call then add integer while binding as shown below: `${response.Group1.message.pathToAttribute }`. Here, Group1 represents the response of first call and so on.

JSON

```
{
  "call": [
    {
      "name": "Group",
      "connection": "acctAuth",
      "url": "https://<domain name>/api/v2/group_memberships",
      "httpMethod": "POST",
```

```

    "httpParams": "{ \"group_membership\": { \"user_id\": \"${account.accountID}\", \"group_id\": \"${entitlementValue.entitlementID}\" }, \"role_id\": \"${entitlementValue.roleID}\" }",
    "httpHeaders": {
      "Authorization": "${access_token}",
      "Accept": "application/json"
    },
    "httpContentType": "application/json",
    "unsuccessResponses": {
      "error": "RecordInvalid"
    }
  },
  {
    "name": "Role",
    "connection": "acctAuth",
    "url": "https://<domain name>/api/v2/users/${account.accountID}",
    "httpMethod": "PUT",
    "httpParams": "{ \"user\": { \"custom_role_id\": ${entitlementValue.entitlementID} } }",
    "httpHeaders": {
      "Authorization": "${access_token}",
      "Accept": "application/json"
    },
    "httpContentType": "application/json",
    "unsuccessResponses": {

```

```

        "error": "RecordInvalid"
    }
}
]
}

```

Example 4: Use the unicode value of the \$ character (\u0024) if a variable in the `httpParams` attribute includes a \$ character.

For more information on best practices, see [Best Practices for Configuring PowerShell Scripts](#) in the *Win-PS Integration Guide*.

JSON

```

{
  "call": [
    {
      "name": "AADGroup",
      "connection": "acctAuth",
      "url": "<domain name>/SaviyntApp/PS/ExecutePSScript",
      "httpMethod": "POST",
      "httpParams": "{ \"Script\": \"\u0024pw = '${connection.passwordForIIS}' -asplaintext -force;\u0024mycrea",
      "httpHeaders": {
        "Authorization": "${access_token}"
      },
    },
  ],
}

```

```
    "httpContentType": "application/x-www-form-urlencoded",
    "successResponses": {
      "statusCode": [
        200,
        201,
        204,
        205
      ]
    }
  }
]
```

RemoveAccessJSON

This parameter is used to remove the assigned entitlements from an account.

Example 1: To define this parameter, use a format similar to the following:

JSON

```
{
  "call": [
    {
```

```
"name": "Role",
"connection": "acctAuth",
"url": "https://abc.zendesk.com/api/v2/users/${account.accountID}",
"httpMethod": "PUT",
"httpParams": "{\"user\": {\"custom_role_id\": ${entitlementValue.entitlementID}}}",
"httpHeaders": {
  "Authorization": "${access_token}",
  "Accept": "application/json"
},
"httpContentType": "application/json",
"successResponses": {
  "statusCode": [
    200,
    201
  ]
}
},
{
  "name": "Group",
  "connection": "acctAuth",
  "url": "https://abc.zendesk.com/api/v2/users/${account.accountID}",
  "httpMethod": "PUT",
```

```

    "httpParams": "{\"user\": {\"custom_Group_id\": ${entitlementValue.entitlementID}}}\",
    "httpHeaders": {
      "Authorization": "${access_token}",
      "Accept": "application/json"
    },
    "httpContentType": "application/json",
    "successResponses": {
      "statusCode": [
        200,
        201
      ]
    }
  }
]
}

```

Example 2: To include both group and role entitlement types in separate calls, specify the entitlement type in the **name** attribute in a format similar to the following:

JSON

```

{
  "call": [{

```



```

        "name": "Group",
        "connection": "acctAuth",
        "url": "https://<domain name>/api/v2/users/${account.accountID}/group_memberships",
        "httpMethod": "GET",
        "httpHeaders": {
            "Authorization": "${access_token}",
            "Accept": "application/json"
        },
        "httpContentType": "application/json"
    },
    {
        "name": "Group",
        "connection": "acctAuth",
        "url": "https://<domain name>/api/v2/group_memberships/${for (Map map : response.Group1.message.gr
        "httpMethod": "DELETE",
        "httpHeaders": {
            "Authorization": "${access_token}",
            "Accept": "application/json"
        },
        "httpContentType": "application/json"
    },
    {

```

```

    "name": "Role",
    "connection": "acctAuth",
    "url": "https://<domain name>/api/v2/users/${account.accountID}",
    "httpMethod": "PUT",
    "httpParams": "{\"user\": {\"custom_role_id\": ${entitlementValue.entitlementID}}}",
    "httpHeaders": {
        "Authorization": "${access_token}",
        "Accept": "application/json"
    },
    "httpContentType": "application/json"
}
]
}

```

Example 3:

JSON

```

{
  "call": [
    {
      "name": "Role",
      "connection": "acctAuth",

```

```

    "url": "https://<domain name>/api/users/${account.accountID}",
    "httpMethod": "GET",
    "httpParams": "",
    "httpHeaders": {
        "X-COUPA-API-KEY": "${access_token}",
        "Accept": "application/json"
    },
    "httpContentType": "application/json"
},
{
    "name": "Role",
    "connection": "acctAuth",
    "url": "https://<domain name>/api/users/${account.accountID}",
    "httpMethod": "PUT",
    "httpParams": "<?xml version='1.0' encoding='UTF-8'?><user><roles><roles><user>",
    "httpHeaders": {
        "X-COUPA-API-KEY": "${access_token}",
        "Accept": "application/xml"
    },
    "httpContentType": "application/xml"
},
{

```

```

    "name": "Role",
    "connection": "acctAuth",
    "url": "https://<domain name>/api/users/${account.accountID}",
    "httpMethod": "PUT",
    "httpParams": "<?xml version=\"1.0\" encoding=\"UTF-8\"?><user><roles>${String rolesStr = '';int size =
    \"httpHeaders\": {
        \"X-COUPA-API-KEY\": \"${access_token}\",
        \"Accept\": \"application/xml\"
    },
    \"httpContentType\": \"application/xml\",
    \"successResponses\": {
        \"statusCode\": [
            200,
            201
        ]
    }
},
{
    \"name\": \"User Group\",
    \"connection\": \"acctAuth\",
    \"url\": \"https://<domain name>/api/user_group_memberships?user-id=${account.accountID}\",
    \"httpMethod\": \"GET\",

```

```

    "httpParams": "",
    "httpHeaders": {
        "X-COUPA-API-KEY": "${access_token}",
        "Accept": "application/json"
    },
    "httpContentType": "application/json"
},
{
    "name": "User Group",
    "connection": "acctAuth",
    "url": "https://<domain name>/api/user_group_memberships/${for (Map map : response.get('User Group1').me
    "httpMethod": "DELETE",
    "httpParams": "",
    "httpHeaders": {
        "X-COUPA-API-KEY": "${access_token}",
        "Accept": "application/json"
    },
    "httpContentType": "application/json",
    "successResponses": {
        "statusCode": [
            403
        ]
    }
}

```

```

    }
  },
  {
    "name": "Business Group",
    "connection": "acctAuth",
    "url": "https://<domain name>/api/users/${account.accountID}",
    "httpMethod": "GET",
    "httpParams": "",
    "httpHeaders": {
      "X-COUPA-API-KEY": "${access_token}",
      "Accept": "application/json"
    },
    "httpContentType": "application/json"
  },
  {
    "name": "Business Group",
    "connection": "acctAuth",
    "url": "https://<domain name>/api/users/${account.accountID}",
    "httpMethod": "PUT",
    "httpParams": "{\"content-groups\":\"\\\"\"}",
    "httpHeaders": {
      "X-COUPA-API-KEY": "${access_token}",

```

```

    "Accept": "application/json"
  },
  "httpContentType": "application/json"
},
{
  "name": "Business Group",
  "connection": "acctAuth",
  "url": "https://<domain name>/api/users/${account.accountID}",
  "httpMethod": "PUT",
  "httpParams": "${List responseList = response.'Business Group1'.message.'content-groups';if(responseList
  "httpHeaders": {
    "X-COUPA-API-KEY": "${access_token}",
    "Accept": "application/json"
  },
  "httpContentType": "application/json",
  "successResponses": {
    "statusCode": [
      200,
      201
    ]
  }
}
}

```

```
]
}
```

UpdateUserJSON

This parameter is used for updating the details of users present in EIC.

Example: Update the employee ID and email of a user. This example uses a single REST call to update these user details.

JSON

```
{
  "actions": {
    "Update Login": {
      "call": [
        {
          "name": "Update Login",
          "callOrder": 0,
          "connection": "userAuth",
          "url": "https://<domain name>/odata/v2/User('${user.employeeid}')" ,
          "httpMethod": "POST",
          "httpParams": "{ \"__metadata\": { \"uri\": \"PerEmail(emailType='920',personIdExternal='${user.employeeid}')]\" } }",
          "httpHeaders": {
            "x-http-method": "MERGE",
```



```
    "Accept": "application/json",
    "Authorization": "${access_token}"
  },
  "contentType": "application/json",
  "successResponses": {
    "statusCode": [
      200,
      201
    ]
  }
}
```

Example: Update user details such as user's manager and department. This example uses two REST calls to update these user details. The first call is used to obtain the user details. The second call uses the response of the first call to update the user's manager and department details.

JSON

```

{
  "actions": {
    "Update Login": {
      "call": [
        {
          "name": "Update Login",
          "callOrder": 0,
          "connection": "auth",
          "url": "https://<domain name>/<company>/itsm/query?q=SELECT+Id+FROM+User+where+FederationIdentifier+=",
          "httpMethod": "GET",
          "httpHeaders": {
            "Authorization": "${access_token}"
          },
          "successResponses": {
            "statusCode": [
              200
            ]
          }
        },
        {
          "name": "Update Login",
          "callOrder": 1,

```

```

    "connection": "auth",
    "url": "https://<domain name>/<company>/itsm/service-request",
    "httpMethod": "POST",
    "httpContentType": "application/json",
    "httpParams": "{  \"Fields\": [ { \"Name\": \"requestDefinitionId\", \"Value\": \"a573b0000000PPe1AAH\" } ] }",
    "httpHeaders": {
      "Authorization": "${access_token}"
    },
    "successResponses": {
      "statusCode": [
        200,
        201
      ]
    }
  ]
}

```

ChangePassJSON

This parameter is used to change password.

Example: To define this parameter, use a format similar to the following:

JSON

```
{
  "call": [
    {
      "name": "call1",
      "connection": "acctAuth",
      "url": "https://<domain name>/v2/users/${account.accountID}/password",
      "httpMethod": "PUT",
      "httpParams": "{\"password\": \"${password}\"}",
      "httpHeaders": {
        "Authorization": "${access_token}"
      },
      "httpContentType": "application/json",
      "successResponses": {
        "statusCode": [
          201,
          200,
          204
        ]
      }
    }
  ]
}
```

```
}  
]  
}
```

RemoveAccountJSON

This parameter is used to remove the account.

Example 1: To pass DELETE method in the call, use a format similar to the following:

JSON

```
{  
  "call": [  
    {  
      "name": "call1",  
      "connection": "userAuth",  
      "url": "https://{org}/2.0/users/${account.accountID}",  
      "httpMethod": "DELETE",  
      "httpHeaders": {  
        "Authorization": "${access_token}"  
      },  
      "httpContentType": "application/json"  
    }  
  ]  
}
```

```
]
}
```

Example 2: To pass a body in a DELETE call, use the DELETEWITHBODY method in a format similar to the following:

JSON

```
{
  "call": [
    {
      "name": "call1",
      "connection": "acctAuth",
      "url": "https://<domain name>/csp/gateway/am/api/users/",
      "httpMethod": "DELETEWITHBODY",
      "httpParams": "{{\"user\":{\"name\": \"${account.accountID}\"}}}",
      "httpHeaders": {
        "csp-auth-token": "${access_token}",
        "Accept": "application/json"
      },
      "httpContentType": "application/json",
      "successResponses": {
        "statusCode": [
          200,

```

```
    201,  
    404  
  ]  
}  
}  
]  
}
```

Ticket JSONs

CreateTicketJSON and TicketStatusJSON are used to raise and close tickets from IGA. IGA generally uses this functionality for the disconnected applications where a user cannot request access directly.

REST Connector supports binding of user, account, Entitlement objects in ticket JSON. It is used to get attribute values into createTicket or getTicketStatus JSON. All binding objects and the domain objects are case sensitive.

Attributes exposed per binding object:

- **user:** username, firstname, preferredFirstName, lastname, middlename, street, city, comments, statuskey, startdate, enddate, manager, password, location, jobCode, jobDescription, employeeType, systemUserName, departmentNumber, title, state, companyname, costcenter, departmentname, employeeeclass, entity, jobcodedesc, locationdesc, locationnumber, siteid, orgunitid, region, regioncode, owner, employeeid, lastsyncdate, createdate, email, phonenumber, job_function, country,

displayname, locale, customproperty1-customproperty65, secondaryPhone, secondaryEmail, leaveStatus, riskscore, userSource, userSourceKey, jobID, savUpdateDate.

- **account:** name, updatedate, updateUser, systemid, accountID, saviyntConnectJobId, displayName, Accounts, referenced_accountkey, referenced_accountName, description, comments, accounttype, accountclass, usergroup, validfrom, validthrough, userlock, incorrectlogons, creator, createdon, lastlogondate, lastpasswordchange, passwordlockdate, jobid, privileged, passwordchangestatus, customproperty1-customproperty56.
- **task:** tasktype, taskdate, status, password, ownerType, ownerkey, comments, accountName, upadteuser, requestedBy, users, ArsTasks, parenttask, provisioningMetadata, provisioningComments, startDate, endDate, Roles, assignedFromRole, assignedFromRule, assignedFromRoles, ticketId, source, sourceId, provisioningTries, savconattrs.
- **reqAttrs:** Dynamic Attribute values created in Endpoint.
- **endpoints:** endpointname, description, customproperty1-customproperty45, customproperty1Label-customproperty56Label, accountNameRule, entsWithNewAccount, ownerType, ownerkey, requestowner, requestownertype, accessquery, accountTypeNoDeprovision, accountTypeNoPasswordChange, userAccountCorrelationRule, accountTypeForServiceAccount, primaryAccountType, createEntTaskforRemoveAcc, allowRemoveAllRoleOnRequest, serviceAccountNameRule, serviceAccountAccessQuery, transients, parentAccountPattern, displayName, status, roleTypeAsJson, applicationLogo, applicationUrl, disableaccountrequest, enableCopyAccess, lastImport, createDate, updateDate, createdBy, updatedBy, createdFrom, accountNameValidatorRegex, jrmDataPopulated, taskemailtemplates.
- **requestor:** username, firstname, preferredFirstName, lastname, middlename, street, city, comments, statuskey, startdate, enddate, manager, password, location, jobCode, jobDescription, employeeType, systemUserName, departmentNumber, title, state, companyname, costcenter, departmentname, employeeeclass, entity, jobcodedesc, locationdesc, locationnumber, siteid, orgunitid, region, regioncode, owner, employeeid, lastsyncdate, createdate, email, phonenumber, job_function, country,

displayname, locale, customproperty1-customproperty65, secondaryPhone, secondaryEmail, leaveStatus, riskscore, userSource, userSourceKey, jobID, savUpdateDate.

- **password:** It will generate random password.
- **beneficiary:** username, firstname, preferredFirstName, lastname, middlename, street, city, comments, statuskey, startdate, enddate, manager, password, location, jobCode, jobDescription, employeeType, systemUserName, departmentNumber, title, state, companyname, costcenter, departmentname, employeeeclass, entity, jobcodedesc, locationdesc, locationnumber, siteid, orgunitid, region, regioncode, owner, employeeid, lastsyncdate, createdate, email, phonenumber, job_function, country, displayname, locale, customproperty1-customproperty65, secondaryPhone, secondaryEmail, leaveStatus, riskscore, userSource, userSourceKey, jobID, savUpdateDate.
- **requestedByDate:** It gets the value of customproperty4 mentioned in dynamicAttributes.
- **allEntitlementsValues:** It binds all the entitlement values separated by colon(:) which are requested by requestor. For example, group1 : group2 : role1.
- **ticketID:** It gets Ticked ID which is assigned after creating the ticket task.
- **approvedBy:** It gets user who approved the ticket.

CreateTicketJSON

This parameter is used to create a ticket in the target ticketing applications.

To provision multiple tasks simultaneously while requesting access to the target application using the `${taskIds}` binding variable, use a format similar to the following:

JSON

```
{
  "call": [
    {
      "name": "call1",
      "connection": "userAuth",
      "url": "https://<domain-name>/SaviyntRequest.do?Action=create_ritm",
      "httpMethod": "POST",
      "httpParams": "{\"bp_id\":\"${user.username}\",\"name\":\"${user.lastname},${user.firstname}\",\"${taskIds",
      "httpHeaders": {
        "Authorization": "${access_token}"
      },
      "httpContentType": "application/json",
      "ticketidPath": "Request Number",
      "unsuccessResponses": {
        "message": "Failed"
      }
    }
  ]
}
```

Example 1: To obtain metadata of entitlement values using the `${allEntitlementsValues}` binding variable, use a format similar to the following:

JSON

```
{
  "call": [
    {
      "name": "call1",
      "connection": "acctAuth",
      "url": "https://<domain name>/api/v2/tickets.json",
      "httpMethod": "POST",
      "httpParams": "{\"ticket\": {\"subject\": \"My Computer is Not working \", \"comment\": { \"body\": \"${allEntitlementsValues}\" } } }",
      "httpHeaders": {
        "Authorization": "${access_token}"
      },
      "httpContentType": "application/json",
      "ticketidPath": "ticket.id",
      "successResponses": {
        "statusCode": [
          200,
          201
        ]
      }
    }
  ]
}
```

```

    },
    "extraInfo": {
      "entValExtraInfo": [
        "customproperty4",
        "customproperty5",
        "customproperty6"
      ]
    }
  }
]
}

```

Example 2: To map the response header with Saviynt attributes, use a format similar to the following:

JSON

```

{
  "call": [
    {
      "name": "call1",
      "connection": "userAuth",
      "url": "https://<domain name>/api/arsys/v1/entry/WOI:WorkOrderInterface_Create",
      "httpMethod": "POST",

```

```

    "httpParams": "{ \"values\": { \"Customer Organization\": \"GEICO\", \"Support Company\": \"<company name>\", \"
    \"httpHeaders\": {
      \"Authorization\": \"${access_token}\"
    },
    \"httpContentType\": \"application/json\",
    \"ticketidPath\": \"response.headers.get('Location')\",
    \"successResponses\": {
      \"statusCode\": [
        200,
        201
      ]
    }
  }
}

```

Example 3: To support dynamic binding for ticketidPath, use a format similar to the following:

JSON

```

{
  \"ticketidPath\": \"${(task.tasktype==3)?'call1.message.ticket.id':'1234'}\",
  \"call\": [

```

```

{
  "name": "call1",
  "connection": "acctAuth",
  "url": "${(task.tasktype==3)? 'https://<domain name>/api/v2/tickets.json' : 'https://<domain name>/api/v2'}",
  "httpMethod": "${(task.tasktype==3)? 'POST' : 'GET'}",
  "httpParams": "${(task.tasktype==3)? '{\"ticket\": {\"subject\": \"Rain is Raining thunderlightning--011\"}}' : ''}",
  "httpHeaders": {
    "Authorization": "${access_token}"
  },
  "httpContentType": "application/json",
  "successResponses": {
    "statusCode": [
      200,
      201
    ]
  }
}
]
}

```

Example 4: To include multiple calls in the JSON, use a format similar to the following:

JSON

```
{
  "call": [
    {
      "name": "call1",
      "connection": "acctAuth",
      "url": "https://<domain name>/api/v2/tickets.json",
      "httpMethod": "POST",
      "httpParams": "\"ticket\": {\"subject\": \"Create ticket\", \"comment\": { \"body\": \"The smoke is very",
      "httpHeaders": {
        "Authorization": "${access_token}"
      },
      "httpContentType": "application/json",
      "ticketidPath": "ticket.id",
      "successResponses": {
        "serviceResult.result": [
          "true"
        ]
      }
    },
    {
      "name": "call2",
```

```

    "connection": "acctAuth",
    "url": "https://<domain name>/api/v2/tickets.json",
    "httpMethod": "POST",
    "httpParams": "{ \"ticket\": { \"subject\": \"Create ticket\", \"comment\": { \"body\": \"The smoke is ver
    \"httpHeaders\": {
      \"Authorization\": \"${access_token}\"
    },
    \"httpContentType\": \"application/json\",
    \"ticketidPath\": \"ticket.id\",
    \"successResponses\": {
      \"serviceResult.result\": [
        \"true\"
      ]
    }
  }
}
}
}

```

Example 5: To get approvedBy details using the `${approvers}` binding variable, use a format similar to the following:

JSON


```
{
  "call": [
    {
      "name": "call1",
      "connection": "acctAuth",
      "url": "https://<domain name>/api/v2/tickets.json",
      "httpMethod": "POST",
      "httpParams": "{ \"ticket\": { \"subject\": \"My ${user.firstname} first name\", \"comment\": { \"body\": \"A",
      "httpHeaders": {
        "Authorization": "${access_token}"
      },
      "httpContentType": "application/json",
      "ticketidPath": "ticket.id"
    }
  ]
}
```

Example 6: To add for service account owner objects, use a format similar to the following:

JSON

```
{
  "call": [
```

```

{
  "name": "call1",
  "connection": "acctAuth",
  "url": "https://<domain name>/api/v2/tickets.json",
  "httpMethod": "POST",
  "httpParams": "{\"ticket\": {\"subject\": \"Zendesk ticketing system for ${ServiceAccountOwnerMap.Service\"",
  "httpHeaders": {
    "Authorization": "${access_token}"
  },
  "httpContentType": "application/json",
  "ticketidPath": "ticket.id",
  "successResponses": {
    "statusCode": [
      200,
      201
    ]
  }
}
]
}

```

Example 7: To send comments and justification added during provisioning to the ServiceNow tickets using the `${businessJustification}` binding variable, use a format similar to the following:

JSON

```
{
  "call": [
    {
      "name": "call1",
      "connection": "acctAuth",
      "url": "https://<domain name>/api/v2/tickets.json",
      "httpMethod": "POST",
      "httpParams": "{\"ticket\": {\"subject\": \"Zendesk ticketing system ${businessJustification}\", \"comment\": \"\"}}",
      "httpHeaders": {
        "Authorization": "${access_token}"
      },
      "httpContentType": "application/json",
      "ticketidPath": "ticket.id",
      "successResponses": {
        "statusCode": [
          200,
          201
        ]
      }
    }
  ]
}
```

```
}  
}  
]  
}
```

The following table describes attributes in `CreateTicketJSON`:

Attribute	Description
ticketIdPath	This is a mandatory attribute populated with a unique key from the response of the create ticket API call.
extraInfo	Use this attribute to define the properties such as custom properties to get the metadata of entitlement values.

TicketStatusJSON

This parameter is used to fetch the status of ticket from the ticketing application. The options are:

- Closed: Use this status to automatically close the tickets. When the ticket is closed, the status is checked and updated accordingly.
- Open: Use this status to manually complete the tickets.

Replace `sc_req_item` with incidents to create incidents instead of requests.

```
{
  "call": [
    {
      "name": "call1",
      "connection": "userAuth",
      "url": "<https://<domain-name>/api/now/table/sc_req_item?> sysparm_query=request.number=",
      "httpMethod": "GET",
      "httpHeaders": {
        "Authorization": "${access_token}"
      },
      "httpContentType": "application/json",
      "ticketStatusPath": "result[0].state",
      "ticketStatusValue": [
        "Open",
        "OPEN",
        "open"
      ],
      "successResponses": [
```

```
    {}  
  ]  
}  
]  
}
```

Example 2: To add multiple calls in the JSON, use a format similar to the following:

JSON

```
{  
  "call": [  
    {  
      "name": "call1",  
      "connection": "acctAuth",  
      "url": "https://<domain name>/api/v2/tickets/${ticketID}.json",  
      "httpMethod": "GET",  
      "httpHeaders": {  
        "Authorization": "${access_token}"  
      },  
      "ticketStatusValue": [  
        "new"  
      ],  
    },  
  ],  
}
```

```

    "contentType": "application/json",
    "ticketStatusPath": "ticket.status",
    "successResponses": [
      {
        "ticket.status": "${ticketStatusValue}"
      }
    ]
  },
  {
    "name": "call2",
    "connection": "acctAuth",
    "url": "https://<domain name>/api/v2/tickets/${ticketID}.json",
    "httpMethod": "GET",
    "httpHeaders": {
      "Authorization": "${access_token}"
    },
    "ticketStatusValue": [
      "new"
    ],
    "contentType": "application/json",
    "ticketStatusPath": "ticket.status",
    "successResponses": [

```

```

    {
      "ticket.status": "${ticketStatusValue}"
    }
  ]
}
]
}

```

Example 3: To define the ticket status to be used to discontinue a task using the `disContinueStatusValue` attribute, use a format similar to the following:

JSON

```

{
  "call": [
    {
      "name": "call1",
      "connection": "userAuth",
      "url": "https://<domain name>/api/now/table/sc_task?sysparm_query=number=${ticketID}&sysparm_limit=1&sys",
      "httpMethod": "GET",
      "httpHeaders": {
        "Authorization": "${access_token}"
      },
    },
  ],
}

```



```
"httpContentType": "application/json",
"ticketStatusPath": "result[0].state",
"ticketStatusValue": [
  "3",
  "Resolved",
  "Completed",
  "closed_complete",
  "Closed Complete"
],
"disContinueStatusValue": [
  "Closed Incomplete"
],
"successResponses": [
  {}
]
}
]
```

The following table describes attributes in **TicketStatusJSON**:

Attribute	Description
ticketStatusPath	This is a mandatory attribute populated with the path from the response of create ticket API call which contains the status of ticket such as, <code>Open</code> , <code>Close</code> .
ticketStatusValue	This is a mandatory attribute populated with the possible values of ticket status on creation such as, <code>Open</code> , <code>open</code> , <code>OPEN</code> .