Wechat

Table of Contents

What is the Wechat?

WeChat is a Chinese social messaging application focus on international markets outside of China and was first released in 2011.

1. Connect and management to WeChat users via a special account.
2. Publish articles and media to be sent to your users.
3. Messages and user events can be passed off to your server to formulate a response.

Read through this user guide to understand how to set up and configure a basic flow using the connector. Track feature additions, compatibility, limitations and API version updates with each release of the connector using the Connector Release Notes. Review the connector operations and functionality using the Technical Reference alongside the demo applications.

MuleSoft maintains this connector under the *Insert Category* support policy.

Prerequisites

This document assumes that you are familiar with Wechat API. To use this connector you need the following:

1. Verified Wechat Service account

Hardware and Software Requirements

For hardware and software requirements, please visit the Hardware and Software Requirements page.

Compatibility

The Wechat connector requires the following dependencies:

Application/Service	Version
Anypoint Studio	3.8
Mule Runtime	EE 3.8.0 and above
Java	JDK 8 and above

How to Install

You can install the connector in Anypoint Studio using the instructions in Installing a Connector from Anypoint Exchange.

Upgrading from an Older Version

If you are currently using an older version of the connector, a small popup appears in the bottom right corner of Anypoint Studio with an "Updates Available" message.

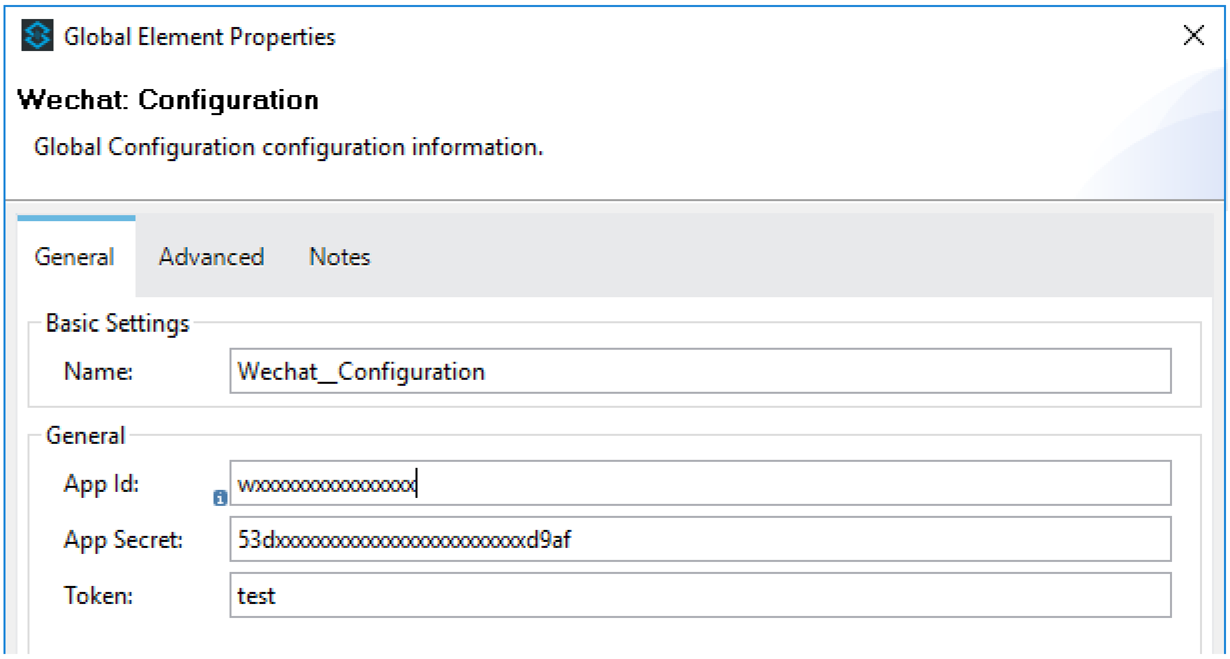
1. Click the popup and check for available updates.
2. Click the Connector version checkbox and click **Next** and follow the instructions provided by the user interface.
3. **Restart** Studio when prompted.
4. After restarting, when creating a flow and using the connector, if you have several versions of the connector installed, you may be asked which version you would like to use. Choose the version you would like to use.

Additionally, we recommend that you keep Studio up to date with its latest version.

How to Configure

To use the Wechat connector in your Mule application, you must configure a global Wechat element that can be used by the Wechat connector (read more about Global Elements). The Wechat connector offers the following global configuration(s), requiring the following credentials: **AppId**, **AppSecret**, **Token**

Field	Description
AppId	The unique certificate of a official account.
AppSecret	The key of a official account’s certificate.
Token	Token set by the developer on the WeChat Official Account Admin Platform.


Global Element Properties

Wechat: Configuration

Global Configuration configuration information.

Basic Settings

Name: Wechat_Configuration

General

App Id: wxxxxxxxxxxxxxxxxx

App Secret: 53dxxxxxxxxxxxxxxxxxxxxxxxxxxd9af

Token: test

Required Connector Namespace and Schema

When designing your application in Studio, the act of dragging the connector from the palette onto the Anypoint Studio canvas should automatically populate the XML code with the connector **namespace** and **schema location**.

Namespace: `http://www.mulesoft.org/schema/mule/wechat`
Schema Location: `http://www.mulesoft.org/schema/mule/wechat/current/mule-wechat.xsd`

TIP If you are manually coding the Mule application in Studio's XML editor or other text editor, define the namespace and schema location in the header of your **Configuration XML**, inside the `<mule>` tag.

```
<mule xmlns:wechat="http://www.mulesoft.org/schema/mule/wechat"
      xmlns="http://www.mulesoft.org/schema/mule/core"
      xmlns:doc="http://www.mulesoft.org/schema/mule/documentation"
      xmlns:spring="http://www.springframework.org/schema/beans"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:schemaLocation="
              http://www.springframework.org/schema/beans
              http://www.springframework.org/schema/beans/spring-beans-current.xsd
              http://www.mulesoft.org/schema/mule/core
              http://www.mulesoft.org/schema/mule/core/current/mule.xsd
              http://www.mulesoft.org/schema/mule/wechat
              http://www.mulesoft.org/schema/mule/wechat/current/mule-wechat.xsd">

      <!-- put your global configuration elements and flows here -->

</mule>
```

Maven Dependency Information

Maven is backing the application, this XML snippet must be included in your `pom.xml` file.

```
<dependency>
  <groupId>commons-codec</groupId>
  <artifactId>commons-codec</artifactId>
  <version>1.9</version>
</dependency>
<dependency>
  <groupId>org.apache.httpcomponents</groupId>
  <artifactId>httpmime</artifactId>
  <version>4.5.3</version>
</dependency>
<dependency>
  <groupId>org.apache.tika</groupId>
  <artifactId>tika-core</artifactId>
  <version>1.14</version>
</dependency>
```

TIP Inside the `<version>` tags, put the desired version number, the word `RELEASE` for the latest release, or `SNAPSHOT` for the latest available version. The available versions to date are:

- **x.y.z**

Operations

Message decryption, message encrytion, upload multimedia file, customer message, group based broadcast, open id list broadcast, delete broadcast, preview broadcast, query broadcast status, get follower list, get user profile, tag, name remark, blacklist, get access token

NOTE See a full list of operations for any version of the connector link:[here].

Common Use Cases

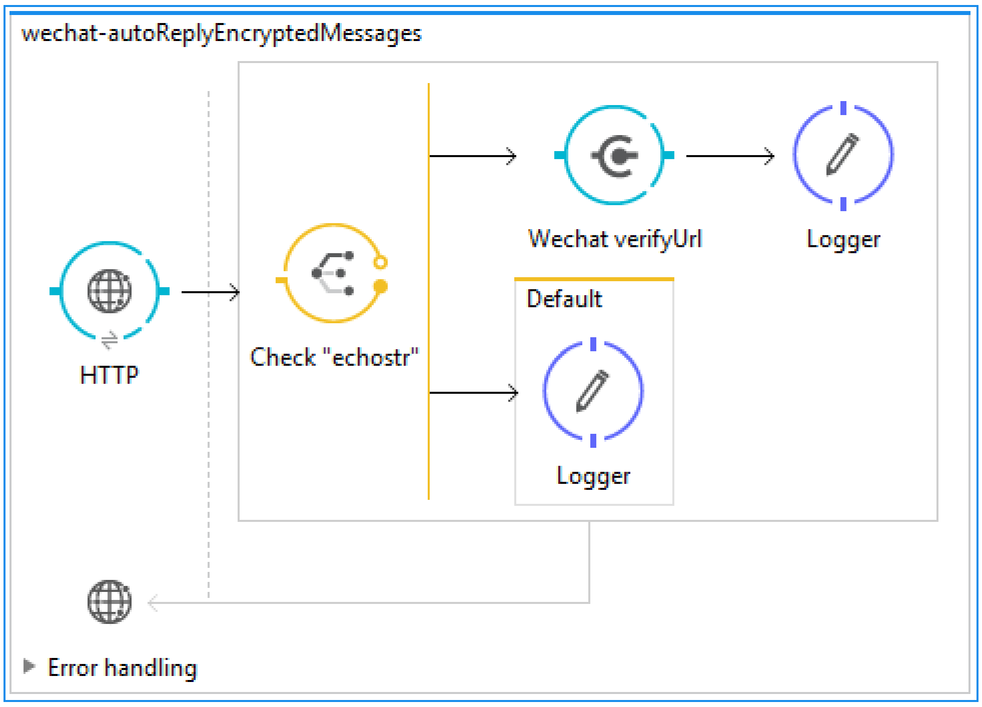
- Verify validity of the URL
- Auto-Reply Encrypted Messages
- Upload Image Material
- Manage Access Token
- Operation with Mule DataSense

Verify validity of the URL

Log in to the WeChat Official Account Admin Platform, after the server configuration is submitted, the WeChat Official Account System will send a GET request to the entered URL pointing to developer's backend. The GET request contains the following parameters:

Parameter	Description
signature	Encrypted signature. This parameter is combined with the Token entered and timestamp and nonce parameters in the request.
timestamp	Time stamp
nonce	Random number
echostr	Random string

Once <wechat:verify-url> confirmed that the GET request has been sent by the WeChat Official Account System and verify the signature, the <wechat:verify-url> should return the echostr parameter value indicating that the request has been successfully received; otherwise, access fails.



```
<flow name="wechat-autoReplyEncryptedMessages">
  <http:listener config-ref="HTTP_Listener_Configuration" path="/msg"
doc:name="HTTP"/>
  <choice doc:name="Check &quot;echostr&quot;">
    <when
expression="#[message.inboundProperties.'http.request.uri'.contains(&quot;echostr&quot
;)]">
      <wechat:verify-url config-ref="Wechat__Configuration" doc:name="Wechat
verifyUrl"/>
      <logger message="#[payload]" level="INFO" doc:name="Logger"/>
    </when>
    <otherwise>
      <logger message="#[payload]" level="INFO" doc:name="Logger"/>
    </otherwise>
  </choice>
</flow>
```

Auto-Reply Encrypted Messages

After encryption/decryption is enabled (i.e. compatibility mode or security mode is selected), new parameters (encryption type and message signature) will appear in the URL to reflect the new settings. When a WeChat user sends a message to an Official Account, the WeChat Official Account Admin System POSTs an XML data package to the URL provided by the developer.

```
<xml>
  <ToUserName><![CDATA[gh_10f6c3c3ac5a]]></ToUserName>
  ......

<Encrypt><![CDATA[hQM/NS0ujPGbF+/8yVe61E3mUVWVO1izRlZdyv26zrVUSE3zUEBdcXITxjbjiHH38kex
VdpQLCnRfbrqny1yGvgqqKTGKxJWWQ9D5WiiUKxavHRNzYVzAjYkp7esNGy7HJcl/P3BGarQF3+AWyNQ5w7xax
5GbOwiXD54yri7xmNMHBOHapDzBslbnTFiEy+8sjSl4asNbn2+ZVBpqGsyKDv0ZG+DlSlXlW+gNPVLP+YxeUhJ
cyfp91qoa0FJagRNlkNul4mGz+sZXJs0WF7lPx6lslDGW3J66crvIIx/klpl0oa/tC6n/9c8OFQ9pp8hrLq7B9
EaAGFlIyz5UhVLiWPN97JkL6JCfxVooVMEKcKRrrlRDGe8RWVM3EW/nxk9Ic37lYY5j97YZfq375AoTBdGDtoP
FZsvv3Upyut1i6G0JRogUsMPlyZl9B8Pl/wcA7k7i4LYMr2yK4SxNFrBUw==]]></Encrypt>
</xml>
```

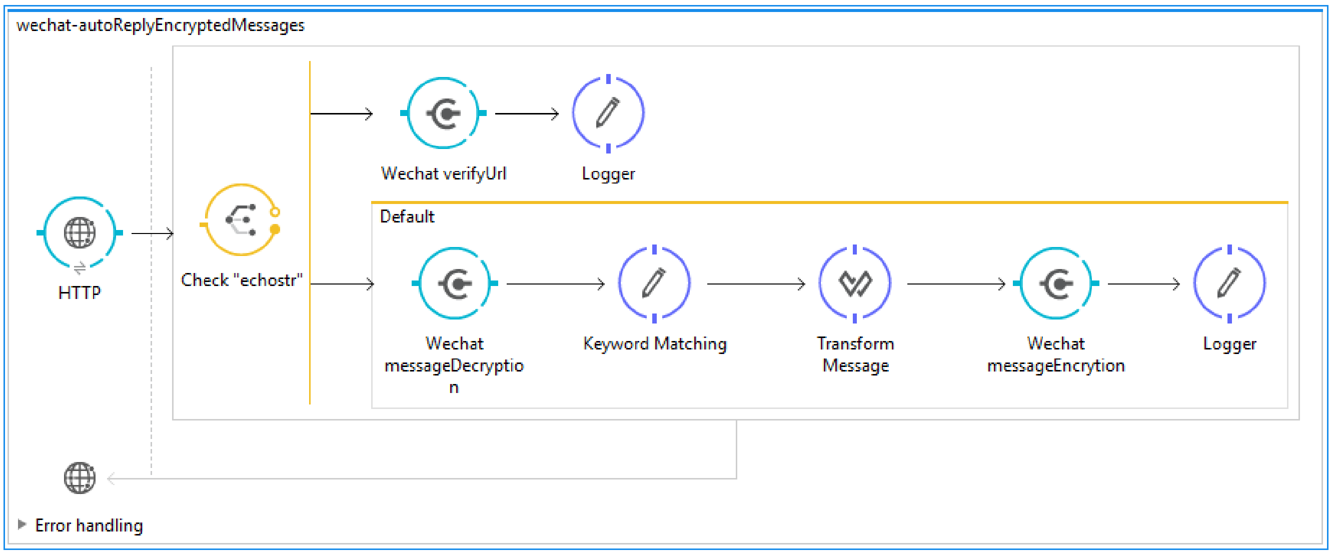
Past it and URI to <wechat:message-decryption>, URI parameters should contain: msg_signature (note that it is msg_signature, not signature), timestamp, and nonce as well as encrypt_msg. If it proccess successful, sMsg will be returned, containing the following XML content in plaintext format:

```
<xml>
  <ToUserName><![CDATA[gh_10f6c3c3ac5a]]></ToUserName>
  <FromUserName><![CDATA[oyORnuP8q7ou2gfYjqLzSIWZf0rs]]></FromUserName>
  <CreateTime>1411035097</CreateTime>
  <MsgType><![CDATA[text]]></MsgType>
  <Content><![CDATA[this is a test message]]></Content>
  <MsgId>6060349595123187712</MsgId>
</xml>
```

Use <wechat:message-encrytion> to encrypt the reply message. If encryption succeeds, sEncryptMsg is an encrypted message, as shown below:

```
<xml>

<Encrypt><![CDATA[LDFAmKFr7U/RMmwRbsR676wjym90byw7+hhh226e8bu6KVYy00HheIsVER4eMgz/VBto
fSaeXXQBz6fVdkN2CzBUaTtjJeTCXEIDfTBNxpw/QRLGLqqMZHA3I+JiBxrrSzd2yXuXst7TdkVgY4lZEHQcWk
85x1niT79XLaWQog+OnBV31eZbXGPPv8dZciKqGo0meTYi+fkMEJdyS8OE7NjO79vpIyIw7hMBtEXPBK/tJGN5
m5SoAS6I4rRZ8Zl8umKxXqgr7N8ZOs6DB9tokpvSl9wT9T3E62rufaKP5EL1imJUd1pngxy09EP24O8Th4bCrd
UcZpJio2l11vE6bWK2s5WrLuO0cKY2GP2unQ4fDxh0L4ePmNOVFJwp9Hyvd0BAsleXA4jWeOMw5nH3Vn49/Q/Z
AQ2HN3dB0bMA+6KJYLvIzTz/Iz6vEjk8ZkK+AbhW5eldnyRDXP/OWfZH2P3WQZUwc/G/LGmS3ekqMwQThhS2Eg
5t4yHv0mAIei07Lknip8nnwgEeF4R9hOGutE9ETsGG4CP1LHTQ4fgYchOMfB3wANOjIt9xendbhHbu51Z4OKnA
0F+MlgZomiqweT1v/+LUxcsFAZ1J+Vtt0FQXElDKg+YyQnRCiLl3I+GJ/cxSj86XwClZC3NNhAkVU11SvxcXEY
h9smckV/qRP2Acsvdls0UqZVWnPtzgx8hc8QBZaeH+JeiaPQD88frNvA==]]></Encrypt>
  <MsgSignature><![CDATA[8d9521e63f84b2cd2e0daa124eb7eb0c34b6204a]]></MsgSignature>
  <TimeStamp>1411034505</TimeStamp>
  <Nonce><![CDATA[1351554359]]></Nonce>
</xml>
```

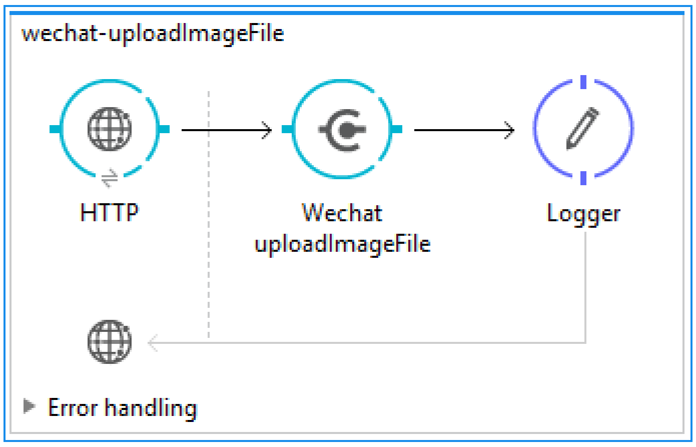


```
<flow name="wechat-autoReplyEncryptedMessages">
  <http:listener config-ref="HTTP_Listener_Configuration" path="/message"
doc:name="HTTP"/>
  <choice doc:name="Check &quot;echostr&quot;">
    <when
expression="#[message.inboundProperties.'http.request.uri'.contains(&quot;echostr&quot
;)]">
      <wechat:verify-url config-ref="Wechat__Configuration" doc:name="Wechat
verifyUrl"/>
      <logger message="#[payload]" level="INFO" doc:name="Logger"/>
    </when>
    <otherwise>
      <wechat:message-decryption config-ref="Wechat__Configuration" doc:name="Wechat
messageDecryption" encodingAesKey="UcNGigaETbwSX4neAykOm8YAPQyRGDXtZziW5llbeYI" />
      <logger message="#[payload]#" level="INFO" doc:name="Keyword Matching"/>
      <dw:transform-message doc:name="Transform Message">
        <dw:set-payload><![CDATA[%dw 1.0
%output application/xml
---
{
  xml: {
    ToUserName: "oUasjhdf83LHF92jshfKJ_8CCdlU" as :cdata,
    FromUserName: "gh_a8dc7dfb8bd9" as :cdata,
    CreateTime: 1489683323,
    MsgType: "text" as :cdata,
    Content: "Hello World!" as :cdata,
    FuncFlag: 0
  }
}]]></dw:set-payload>
      </dw:transform-message>
      <wechat:message-encrytion config-ref="Wechat__Configuration"
encodingAesKey="UcNGigaETbwSX4neAykOm8YAPQyRGDXtZziW5llbeYI" doc:name="Wechat
messageEncrytion"/>
    </otherwise>
  </choice>
</flow>
```

Upload permanent Image Material

Use <wechat:upload-image-file> to upload image files to the WeChat server. The system then returns a corresponding media ID that enables official accounts to obtain multimedia files. An example of a successful JSON response is as follows:

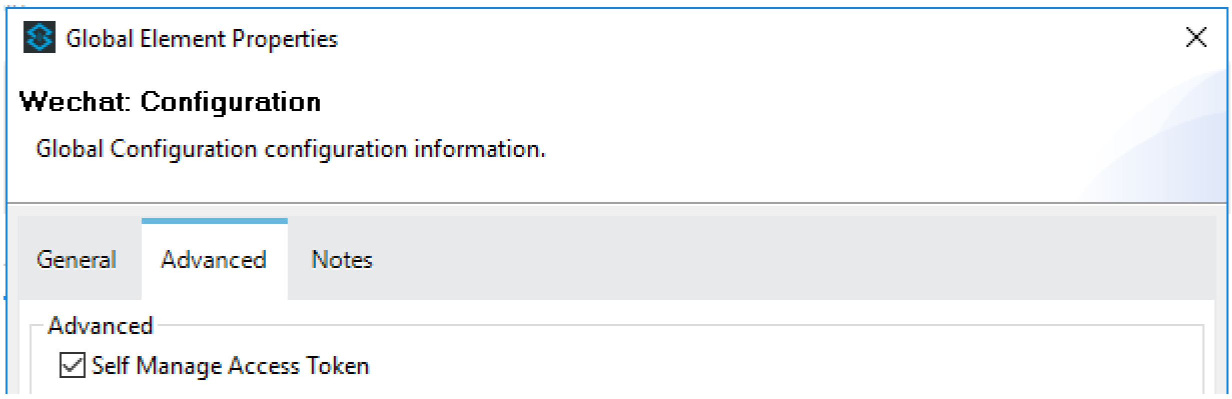
```
{
  "media_id":"MEDIA_ID",
  "url":"URL"
}
```



```
<flow name="wechat-uploadImageFile">
  <http:listener config-ref="HTTP_Listener_Configuration" path="/file"
doc:name="HTTP"/>
  <wechat:upload-permanent-image-file config-ref="Wechat__Configuration"
doc:name="Wechat uploadImageFile" accessToken="#[common.CustomClass.getToken()]"
title="UploadImage"/>
  <logger message="#[payload]" level="INFO" doc:name="Logger"/>
</flow>
```

Manage Access Token

By default, Wechat connector is handling access token for you. If you want to manage it by yourself, select **Self Manage Access Token** in Wechat **Advanced** global configuration.



Global Element Properties

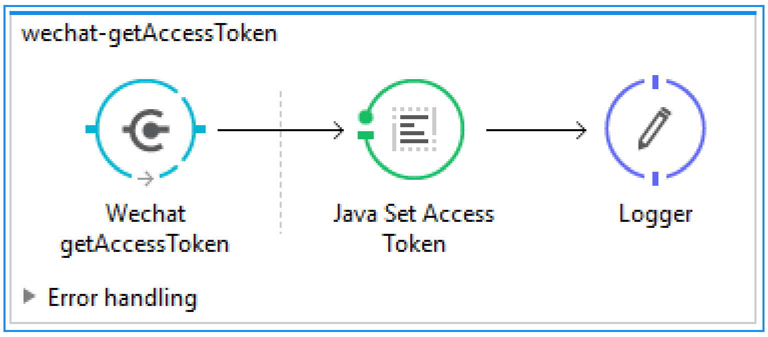
Wechat: Configuration

Global Configuration configuration information.

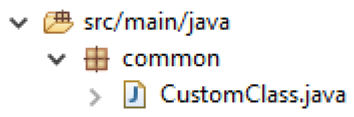
Advanced

☑ Self Manage Access Token

And use <wechat:get-access-token> to poll access token periodically, and then you can save it using java, database...



```
<flow name="wechat-getAccessToken">
  <wechat:get-access-token config-ref="Wechat__Configuration"  doc:name="Wechat
getAccessToken"/>
  <set-payload value="#[common.CustomClass.setToken(payload.access_token)]"
doc:name="Java Set Access Token"/>
  <logger message="#[common.CustomClass.getToken()]" level="INFO" doc:name="Logger"/>
</flow>
```



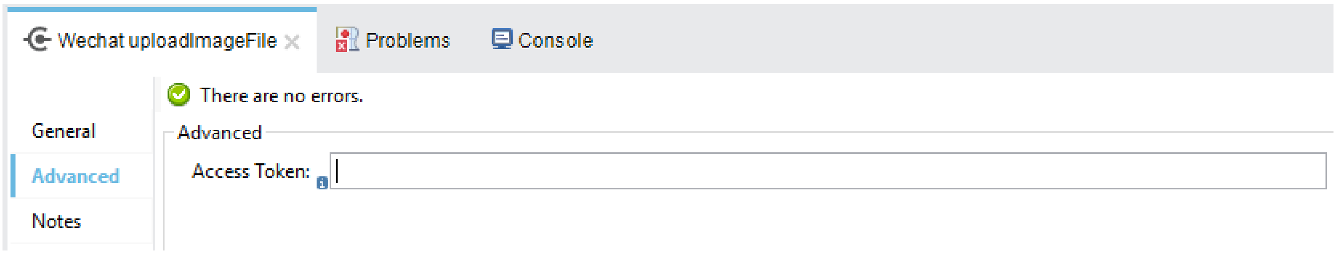
```
package common;

public class CustomClass {
  static private String token;

  public static String getToken() {
    return token;
  }

  public static void setToken(String token) {
    CustomClass.token = token;
  }
}
```

In each operation which needs access token to request wechat API, there is **Access Token** in **Anvanced** for you to input to access token you just poll.



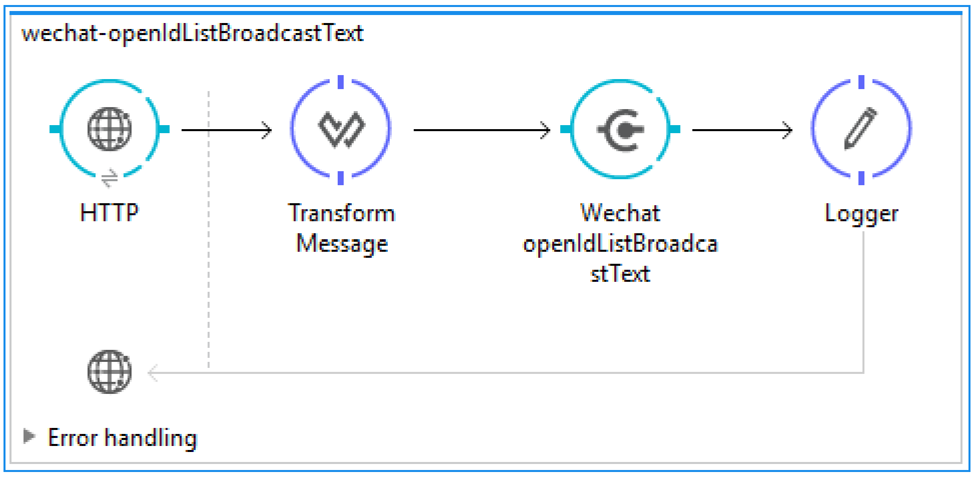
There are no errors.

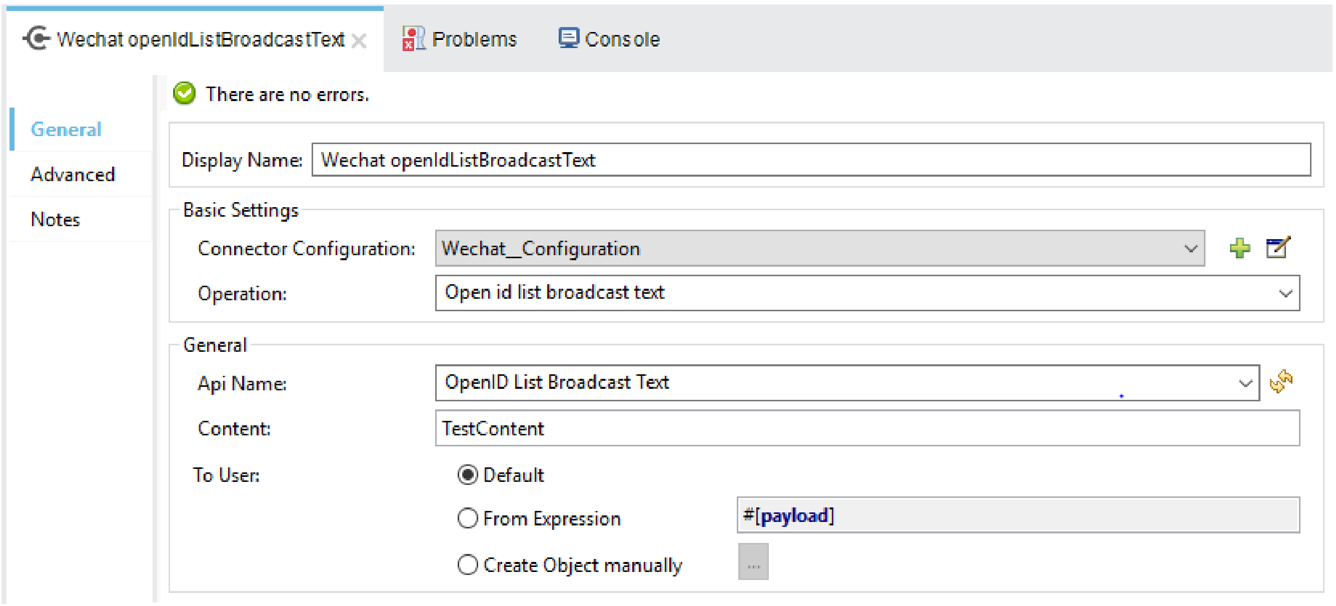
Advanced

Access Token:

Operation with Mule DataSense

Some of the input structure of wechat API is complicated. Like <wechat:open-id-list-broadcast-text>, it split into two field and one of the field is mapping metadata using Mule DataSense.


wechat-openIdListBroadcastText
HTTP
Transform Message
Wechat openIdListBroadcastText
Logger
Error handling



There are no errors.

Display Name: Wechat openIdListBroadcastText

Basic Settings

Connector Configuration: Wechat__Configuration

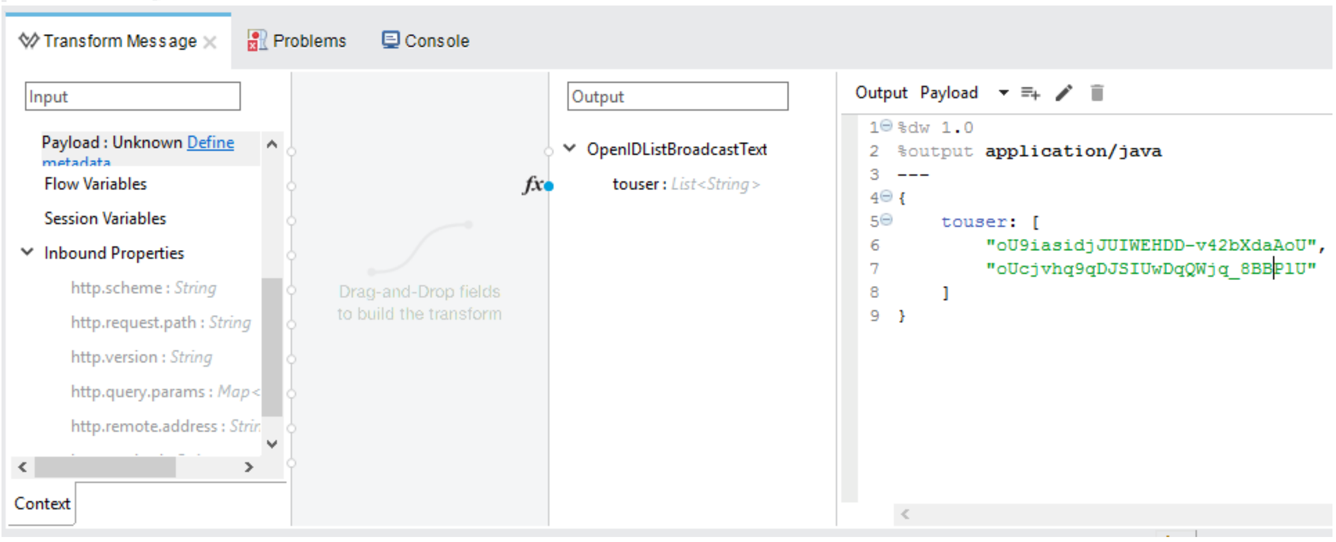
Operation: Open id list broadcast text

General

Api Name: OpenID List Broadcast Text

Content: TestContent

To User: [✓ Default] From Expression #[payload] Create Object manually



```
%dw 1.0
%output application/java
---
{
    touser: [
        "oU9iasidjJUIWEHDD-v42bXdaAoU",
        "oUcjvhq9qDJSIUwDqQWjq_8BBP1U"
    ]
}
```

```
<flow name="wechat-openIdListBroadcastText">
  <http:listener config-ref="HTTP_Listener_Configuration" path="/broadcast"
doc:name="HTTP"/>
  <dw:transform-message doc:name="Transform Message">
    <dw:set-payload><![CDATA[%dw 1.0
%output application/java
---
{
  touser: [
    "oU9iasidjJUIWEHDD-v42bXdaAoU",
    "oUcjvhq9qDJSIUwDqQWjq_8BBPlU"
  ]
}]]></dw:set-payload>
  </dw:transform-message>
  <wechat:open-id-list-broadcast-text config-ref="Wechat__Configuration"
ApiName="OpenIDListBroadcastText" content="TestContent" doc:name="Wechat
openIdListBroadcastText"/>
  <logger message="#[payload]" level="INFO" doc:name="Logger"/>
</flow>
```

Resources

- Access the Wechat Connector Release Notes.