This parameter must be returned when authorization application is successful **[result.resultStatus == S]**, and the accessToken will be invalid after **accessTokenExpiryTime.** | | refreshToken | String | No | The refresh token that is used by the auth client to exchange for a new access token when the access token expires. By using the refresh token, new access tokens can be obtained without further interaction with the user.
This parameter must be returned when authorization application is successful **[result.resultStatus == S]**, and the merchant can use the **refreshToken** to request for a new **accessToken**.
Max. length: 128 characters. | | refreshTokenExpiryTime | String/Datetime | No | Refresh token expiration time, after which the auth client cannot use this token to retrieve a new access token. The value follows the ISO 8601 standard.
This parameter must be returned when authorization application is successful **[result.resultStatus == S]**, and the merchant will not be able to use the refreshToken to retrieve a new **accessToken** after **refreshTokenExpiryTime.** | | customerId | String | No | Resource owner id, maybe user id, app id of merchant's application, merchant id.
Max. length: 64 characters. | | extendInfo | String | No | The extend information,wallet and merchant can put extend info here.
Max. length: 4096 characters. |

Result process logic

For different request results, different actions are to be performed. See the following list for details:

- If the value of **result.resultStatus** is **S**, the authorization token application request is successful. The merchant can use the access token to access the corresponding user resource scope.
- If the value of **result.resultStatus** is **F** or **U**, AuthClient may guide user to try again.

Result

| | | | | --- | --- | --- | | **resultStatus** | **resultCode** | **resultMessage** | | S | SUCCESS | Success. | | U | UNKNOWN_EXCEPTION | An API calling is failed, which is caused by unknown reasons. | | U | REQUEST_TRAFFIC_EXCEED_LIMIT | The request traffic exceeds the limit. | | F | PROCESS_FAIL | A general business failure occurred. Don't retry. | | F | PARAM_ILLEGAL | Illegal parameters exist. For example, a non-numeric input, or an invalid date. | | F | ACCESS_DENIED | The access is denied. | | F | INVALID_API | The called API is invalid or not active. | | F | AUTH_CLIENT_UNSUPPORTED_GRANT_TYPE | The auth client do not support this grant type. | | F | INVALID_AUTH_CLIENT | The auth client is invalid. | | F | INVALID_AUTH_CLIENT_STATUS | Invalid auth client status. | | F | INVALID_REFRESH_TOKEN | The refresh token is invalid. | | F | EXPIRED_REFRESH_TOKEN | The refresh token is expired. | | F | USED_REFRESH_TOKEN | The refresh token has been used. | | F | INVALID_CODE | The authorization code is invalid. | | F | USED_CODE | The authorization code has been used. | | F | EXPIRED_CODE | The authorization code is expired. | | F | REFERENCE_CLIENT_ID_NOT_MATCH | The reference client id does not match. | | F | EXPIRED_AGENT_TOKEN | The access token of mini program is expired. | | F | INVALID_AGENT_TOKEN | The access token of mini program is invalid. |

Sample

The authorization token application is used to exchange the access token based on the auth code after obtaining the auth code.

1. The Mini Program calls my.getAuthCode interface to obtain the authorization code from e-wallet. (Step 1)
2. E-wallet returns the authorization code to the Mini Program (Step 7)
3. The Mini Program sends authorization code to the merchant server (Step 8)
4. The merchant server calls /v1/authorizations/applyTokeninterface to obtain the access token from e-wallet server and e-wallet server returns the access token and customer ID to the merchant server (Step 9 and Step 11).

Note: Other steps are covered by e-wallet.

Request

A. Retrieving `accessToken` with `authCode`

copy

```
{
  "referenceClientId": "305XST2CSG0N4P0xxxx",
  "grantType": "AUTHORIZATION_CODE",
  "authCode": "2810111301lGZcM9CjlF91WH00039190xxxx",
  "extendInfo": "{\"customerBelongsTo\":\"siteNameExample\"}"
}
```

B. Retrieving `accessToken` with `refreshToken`

copy

```
{
  "grantType": "REFRESH_TOKEN",
  "refreshToken": "2810111301lGZcM9CjlF91WH00039190xxxx",
  "extendInfo": "{\"customerBelongsTo\":\"siteNameExample\"}"
}
```

- **authCode** is from the my.getAuthCode JS-API, you can obtain the `authCode` in the success callback. when **grantType == AUTHORIZATION_CODE** means that we are requesting for the accessToken by the `authCode` .
- **refreshToken** is obtained from the response of the previous accessToken Application call. while **grantType == REFRESH_TOKEN** means that we are requesting for the accessToken by providing the refreshToken.
- **extendInfo,** includes key - **customerBelongsTo** the e-wallet that the customer uses. Corresponding to the field 'siteName' that obtained from the API 'my.getSiteInfo', in the Mini Program scenario this is mandatory.

Response

copy

```
{
  "result": {
    "resultCode": "SUCCESS",
    "resultStatus": "S",
    "resultMessage": "success"
  },
  "accessToken": "281010033AB2F588D14B43238637264FCA5AAF35xxxx",
  "accessTokenExpiryTime": "2019-06-06T12:12:12+08:00",
  "refreshToken": "2810100334F62CBC577F468AAC87CFC6C9107811xxxx",
  "refreshTokenExpiryTime": "2019-06-08T12:12:12+08:00",
  "customerId":"1000001119398804xxxx"
}
```

- **result.resultStatus==S** shows that the application is successful,
- AuthClient can make use of **accessToken** to access the user's resource scope before 2019-06-06T12:12:12+08:00 **[accessTokenExpiryTime].**
- AuthClient can make use of **refreshToken** to request for a new accessToken before 2019-06-08T12:12:12+08:00 **[refreshTokenExpiryTime].**

Source: https://miniprogram.gcash.com/docs/miniprogram_gcash/mpdev/qfd7r1

/v1/authorizations/applyToken {#/v1/authorizations/applytoken}

Last updated: 2021-05-09

Path: miniprogram_gcash

/v1/authorizations/applyToken

2021-05-09 18:43

POST `/v1/authorizations/applyToken`

The `applyToken` interface is used to obtain the access token. This interface can be used in the following cases:

- After the merchant receives **authCode** from Mini-Program, the merchant uses this interface to request the access token from e-wallet. In this scenario, the interface generally needs to be used with the Authorization Prepare Interface.
- When the original token expires, the merchant requests a new access token by using the refresh token. In this scenario, this interface can be used independently.

Message structure

Request

| | | | | | --- | --- | --- | --- | | **Property** | **Data type** | **Required** | **Description** | | referenceClientId | String | No | In Mini-Program scenario, it is the unique identifier of the Mini-Program authorized by the user.
The referenceClientId represents the next-level clien id. When multiple auth codes need to be assigned to the same client, different referenceClientIds can be passed in to distinguish them.
Max. length: 128 characters. | | grantType | String | Yes | Indicates which parameter is to be used to obtain the access token. Possible values are:
- **AUTHORIZATION_CODE**: the `authCode` is to be used to retrieve the accessToken.
- **REFRESH_TOKEN**: the refreshToken is to be used to retrieve the accessToken.
Max. length: 16 characters. | | authCode | String | No | Is required when **grantType** is **AUTHORIZATION_CODE**.
The authorization code, which is used by confidential and public clients to exchange an authorization code for an access token. After the user returns to the client via the Mini-program API, the Mini-program will get the authorization code from the response of and use it to request an access token.
Max. length: 32 characters. | | refreshToken | String | No | **refreshToken** is required when **grantType** is **REFRESH_TOKEN**.
The refresh token, which is used by the auth client to exchange for a new access token when the access token expires. By using the refresh token, new access tokens can be obtained without further interaction with the user.
Max. length: 128 characters. | | extendInfo | String | No | The extend information,wallet and merchant can put extend info here.
Max. length: 4096 characters. |

Response

| | | | | | --- | --- | --- | --- | | **Property** | **Data type** | **Required** | **Description** | | result | **Result** | Yes | The request result, which contains information related to the request result, such as status and error codes. | | accessToken | String | No | An access token that can be used to access the user resource scope.
When authorization application is successful **[result.resultStatus == S]**, the auth client might use accessToken to acccess the corresponding user's resource scope.
Max. length: 128 characters. | | accessTokenExpiryTime | String/Datetime | No | Access token expiration time, which follows the ISO 8601 standard. After this time, authClient will not be able to use this token to deduct from user's account.
This parameter must be returned when authorization application is successful **[result.resultStatus == S]**, and the accessToken will be invalid after **accessTokenExpiryTime.** | | refreshToken | String | No | The refresh token that is used by the auth client to exchange for a new access token when the access token expires. By using the refresh token, new access tokens can be obtained without further interaction with the user.
This parameter must be returned when authorization application is successful **[result.resultStatus == S]**, and the merchant can use the **refreshToken** to request for a new **accessToken**.
Max. length: 128 characters. | | refershTokenExpiryTime | String/Datetime | No | Refresh

token expiration time, after which the auth client cannot use this token to retrieve a new access token. The value follows the ISO 8601 standard.
This parameter must be returned when authorization application is successful **[result.resultStatus == S]**, and the merchant will not be able to use the refreshToken to retrieve a new **accessToken** after **refreshTokenExpiryTime.** | | customerId | String | No | Resource owner id, maybe user id, app id of merchant's application, merchant id.
Max. length: 64 characters. | | extendInfo | String | No | The extend information,wallet and merchant can put extend info here.
Max. length: 4096 characters. |

Result process logic

For different request results, different actions are to be performed. See the following list for details:

- If the value of **result.resultStatus** is **S**, the authorization token application request is successful. The merchant can use the access token to access the corresponding user resource scope.
- If the value of **result.resultStatus** is **F** or **U**, AuthClient may guide user to try again.

Result

| | | | | --- | --- | --- | | **resultStatus** | **resultCode** | **resultMessage** | | S | SUCCESS | Success. | | U | UNKNOWN_EXCEPTION | An API calling is failed, which is caused by unknown reasons. | | U | REQUEST_TRAFFIC_EXCEED_LIMIT | The request traffic exceeds the limit. | | F | PROCESS_FAIL | A general business failure occurred. Don't retry. | | F | PARAM_ILLEGAL | Illegal parameters exist. For example, a non-numeric input, or an invalid date. | | F | ACCESS_DENIED | The access is denied. | | F | INVALID_API | The called API is invalid or not active. | | F | AUTH_CLIENT_UNSUPPORTED_GRANT_TYPE | The auth client do not support this grant type. | | F | INVALID_AUTH_CLIENT | The auth client is invalid. | | F | INVALID_AUTH_CLIENT_STATUS | Invalid auth client status. | | F | INVALID_REFRESH_TOKEN | The refresh token is invalid. | | F | EXPIRED_REFRESH_TOKEN | The refresh token is expired. | | F | USED_REFRESH_TOKEN | The refresh token has been used. | | F | INVALID_CODE | The authorization code is invalid. | | F | USED_CODE | The authorization code has been used. | | F | EXPIRED_CODE | The authorization code is expired. | | F | REFERENCE_CLIENT_ID_NOT_MATCH | The reference client id does not match. | | F | EXPIRED_AGENT_TOKEN | The access token of mini program is expired. | | F | INVALID_AGENT_TOKEN | The access token of mini program is invalid. |

Sample

The authorization token application is used to exchange the access token based on the auth code after obtaining the auth code.

1. The Mini Program calls my.getAuthCode interface to obtain the authorization code from e-wallet. (Step 1)
2. E-wallet returns the authorization code to the Mini Program (Step 7)
3. The Mini Program sends authorization code to the merchant server (Step 8)

4. The merchant server calls /v1/authorizations/applyTokeninterface to obtain the access token from e-wallet server and e-wallet server returns the access token and customer ID to the merchant server (Step 9 and Step 11).

Note: Other steps are covered by e-wallet.

Request

A. Retrieving `accessToken` with `authCode`

copy

```
{
  "referenceClientId": "305XST2CSG0N4P0xxxx",
  "grantType": "AUTHORIZATION_CODE",
  "authCode": "2810111301lGZcM9CjlF91WH00039190xxxx",
  "extendInfo": "{\"customerBelongsTo\":\"siteNameExample\"}"
}
```

B. Retrieving `accessToken` with `refreshToken`

copy

```
{
  "grantType": "REFRESH_TOKEN",
  "refreshToken": "2810111301lGZcM9CjlF91WH00039190xxxx",
  "extendInfo": "{\"customerBelongsTo\":\"siteNameExample\"}"
}
```

- **authCode** is from the my.getAuthCode JS-API, you can obtain the `authCode` in the success callback. when **grantType == AUTHORIZATION_CODE** means that we are requesting for the accessToken by the `authCode` .
- **refreshToken** is obtained from the response of the previous accessToken Application call. while **grantType == REFRESH_TOKEN** means that we are requesting for the accessToken by providing the refreshToken.
- **extendInfo,** includes key - **customerBelongsTo** the e-wallet that the customer uses. Corresponding to the field 'siteName' that obtained from the API 'my.getSiteInfo', in the Mini Program scenario this is mandatory.

Response

copy

```
{
  "result": {
    "resultCode": "SUCCESS",
    "resultStatus": "S",
    "resultMessage": "success"
  },
  "accessToken": "281010033AB2F588D14B43238637264FCA5AAF35xxxx",
  "accessTokenExpiryTime": "2019-06-06T12:12:12+08:00",
```

```
  "refreshToken": "2810100334F62CBC577F468AAC87CFC6C9107811xxxx",
  "refreshTokenExpiryTime": "2019-06-08T12:12:12+08:00",
  "customerId":"1000001119398804xxxx"
}
```

- **result.resultStatus==S** shows that the application is successful,
- AuthClient can make use of **accessToken** to access the user's resource scope before 2019-06-06T12:12:12+08:00 **[accessTokenExpiryTime].**
- AuthClient can make use of **refreshToken** to request for a new accessToken before 2019-06-08T12:12:12+08:00 **[refreshTokenExpiryTime].**

Source: https://miniprogram.gcash.com/docs/miniprogram_gcash/mpdev-old/qfd7r1

/v1/authorizations/cancelToken {#/v1/authorizations/canceltoken}

Last updated: 2022-07-04

Path: miniprogram_gcash

/v1/authorizations/cancelToken

2022-07-04 03:44

POST `/v1/authorizations/cancelToken`

The `cancelToken` API is used to cancel access token at wallet.

Message structure

Request

| | | | | --- | --- | --- | --- | | **Property** | **Data type** | **Required** | **Description** | | accessToken | String | Yes | An access token that can be used to access the user resource scope.
Max. length: 128 characters. | | extendInfo | String | No | The extend information,wallet and merchant can put extend info here.
Max. length: 4096 characters. |

Response

| | | | | --- | --- | --- | --- | | **Property** | **Data type** | **Required** | **Description** | | result | **Result** | Yes | The request result, which contains information related to the request result, such as status and error codes. | | extendInfo | String | No | The extend information,wallet and

merchant can put extend info here.
Max. length: 4096 characters. |

Result process logic

For different request results, different actions are to be performed. See the following list for details:

- If the value of **result.resultStatus** is **S**, that means the authorization is cancelled successfully. AuthClient will not be able to use the AccessToken to access user's resources, and may not use the relative refreshToken to retrieve new AccessToken.
- If the value of **result.resultStatus** is **F** or **U**, that means authorization is cancelled failed, AuthClient may guide user to try again.

Result

| | | | --- | --- | --- | | **resultStatus** | **resultCode** | **resultMessage** | | S | SUCCESS | Success. | | U | UNKNOWN_EXCEPTION | An API calling is failed, which is caused by unknown reasons. | | U | REQUEST_TRAFFIC_EXCEED_LIMIT | The request traffic exceeds the limit. | | F | PROCESS_FAIL | A general business failure occurred. Don't retry. | | F | PARAM_ILLEGAL | Illegal parameters exist. For example, a non-numeric input, or an invalid date. | | F | ACCESS_DENIED | The access is denied. | | F | INVALID_API | The called API is invalid or not active. | | F | INVALID_AUTH_CLIENT_STATUS | Invalid auth client status. | | F | INVALID_ACCESS_TOKEN | The access token is invalid. | | F | INVALID_AUTH_CLIENT | The auth client id is invalid. | | F | EXPIRED_ACCESS_TOKEN | The access token is expired. | | F | EXPIRED_AGENT_TOKEN | The access token of mini program is expired. | | F | INVALID_AGENT_TOKEN | The access token of mini program is invalid. |

Sample

You can cancel the authorization. After cancellation, the refresh_token cannot be used even if it is valid.

When user cancel access token from the Mini Program,

1. The Merchant server calls /v1/authorizations/cancelToken interface to cancel access token (Step 2).
2. And wallet server returns token cacnel result to merchant server (Step 3).

Request

copy

```
{
  "accessToken": "281010033AB2F588D14B43238637264FCA5Axxxx",
  "extendInfo": "{\"customerBelongsTo\":\"siteNameExample\"}"
}
```

- **extendInfo,** includes key - **customerBelongsTo** the e-wallet that the customer uses. Corresponding to the field 'siteName' that obtained from the API 'my.getSiteInfo', in the Mini Program scenario this is mandatory.

Response

copy

```
{
 "result": {
    "resultCode":"SUCCESS",
    "resultStatus":"S",
    "resultMessage":"success"
  }
}
```

Source: https://miniprogram.gcash.com/docs/miniprogram_gcash/mpdev/uh7agv

---

/v1/authorizations/cancelToken {#/v1/authorizations/canceltoken}

Last updated: 2021-05-09

Path: miniprogram_gcash

/v1/authorizations/cancelToken

2021-05-09 18:43

POST `/v1/authorizations/cancelToken`

The `cancelToken` API is used to cancel access token at wallet.

Message structure

Request

| | | | | --- | --- | --- | --- | | **Property** | **Data type** | **Required** | **Description** | | accessToken | String | Yes | An access token that can be used to access the user resource scope.
Max. length: 128 characters. | | extendInfo | String | No | The extend information,wallet and merchant can put extend info here.
Max. length: 4096 characters. |

Response

| | | | | --- | --- | --- | --- | | **Property** | **Data type** | **Required** | **Description** | | result | **Result** | Yes | The request result, which contains information related to the request result, such as status and error codes. | | extendInfo | String | No | The extend information,wallet and merchant can put extend info here.
Max. length: 4096 characters. |

Result process logic

For different request results, different actions are to be performed. See the following list for details:

- If the value of **result.resultStatus** is **S**, that means the authorization is cancelled successfully. AuthClient will not be able to use the AccessToken to access user's resources, and may not use the relative refreshToken to retrieve new AccessToken.
- If the value of **result.resultStatus** is **F** or **U**, that means authorization is cancelled failed, AuthClient may guide user to try again.

Result

| | | | | --- | --- | --- | | **resultStatus** | **resultCode** | **resultMessage** | | S | SUCCESS | Success. | | U | UNKNOWN_EXCEPTION | An API calling is failed, which is caused by unknown reasons. | | U | REQUEST_TRAFFIC_EXCEED_LIMIT | The request traffic exceeds the limit. | | F | PROCESS_FAIL | A general business failure occurred. Don't retry. | | F | PARAM_ILLEGAL | Illegal parameters exist. For example, a non-numeric input, or an invalid date. | | F | ACCESS_DENIED | The access is denied. | | F | INVALID_API | The called API is invalid or not active. | | F | INVALID_AUTH_CLIENT_STATUS | Invalid auth client status. | | F | INVALID_ACCESS_TOKEN | The access token is invalid. | | F | INVALID_AUTH_CLIENT | The auth client id is invalid. | | F | EXPIRED_ACCESS_TOKEN | The access token is expired. | | F | EXPIRED_AGENT_TOKEN | The access token of mini program is expired. | | F | INVALID_AGENT_TOKEN | The access token of mini program is invalid. |

Sample

You can cancel the authorization. After cancellation, the refresh_token cannot be used even if it is valid.

When user cancel access token from the Mini Program,

1. The Merchant server calls /v1/authorizations/cancelToken interface to cancel access token (Step 2).
2. And wallet server returns token cacnel result to merchant server (Step 3).

Request

copy

```
{
  "accessToken": "281010033AB2F588D14B43238637264FCA5Axxxx",
  "extendInfo": "{\"customerBelongsTo\":\"siteNameExample\"}"
}
```

- **extendInfo,** includes key - **customerBelongsTo** the e-wallet that the customer uses. Corresponding to the field 'siteName' that obtained from the API 'my.getSiteInfo', in the Mini Program scenario this is mandatory.

Response

copy

```
{
 "result": {
    "resultCode":"SUCCESS",
    "resultStatus":"S",
    "resultMessage":"success"
  }
}
```

Source: https://miniprogram.gcash.com/docs/miniprogram_gcash/mpdev-old/uh7agv

---

/v1/payments/inquiryPayment {#/v1/payments/inquirypayment}

Last updated: 2022-07-03

Path: miniprogram_gcash

/v1/payments/inquiryPayment

2022-07-03 18:44

POST `/v1/payments/inquiryPayment`

The `inquiryPayment` API is used to inquire the payment result, usually when not able to receive the payment result after a long period of time. Such as:

Note:

1. After Merchant initiates payment and not able to receive the payment result after a long period of time, it can poll Payment Status Inquiry interface.

2. Merchant uses InquiryPayment to determine the Payment status in the asynchronous Payment processing scenario.

Round-robin interval, recommended 5s once, up to 1 minute.

Message structure

Request

| | | | | | --- | --- | --- | --- | | **Property** | **Data type** | **Required** | **Description** | | partnerId | String | Yes | The partnerId allocated by wallet.
Max. length: 32 characters. | | paymentId | String | No | The unqiue ID of a payment generated by Wallet.
Max. length: 64 characters. | | paymentRequestId | String | No | The unqiue ID of a payment generated by Wallet merchants.
Max. length: 64 characters. | | extendInfo | String | No | The extend information,wallet and merchant can put extend info here.
Max. length: 4096 characters. |

Response

| | | | | | --- | --- | --- | --- | | **Property** | **Data type** | **Required** | **Description** | | result | **Result** | Yes | The request result, which contains information related to the request result, such as status and error codes. | | paymentId | String | No | The unqiue ID of a payment generated by Wallet.
Max. length: 64 characters. | | paymentRequestId | String | No | The unqiue ID of a payment generated by Wallet merchants.
Max. length: 64 characters. | | paymentAmount | **Amount** | No | Order amount for display of user consumption records, payment results page. | | paymentTime | String/Datetime | No | Payment success time, which follows the ISO 8601 standard. | | paymentStatus | String | No | SUCCESS - order is succeeded.
FAIL - order is failed.
PROCESSING - order is not paid or is paid but not finish.
CANCELLED - order is cancelled. | | paymentFailReason | String | No | The fail reason of payment order when paymentStatus is FAIL.
Max. length: 256 characters. | | authExpiryTime | String/Datetime | No | Authorization expiry time, has value only when paymentFactor.isAuthorizationPayment is true. | | extendInfo | String | No | The extend information,wallet and merchant can put extend info here.
Max. length: 4096 characters. |

Result Process Logic

For different request results, different actions are to be performed. See the following list for details:

- If the value of **result.resultStatus** is **S**, that means the payment status inquiry is successful, then check **paymentStatus**:
- if **paymentStatus** is **PROCESSING**, means order is not paid or is paid but not finish;

- if **paymentStatus** is **SUCCESS**, means order is succeeded;
- if **paymentStatus** is **FAIL**, means order is failed.
- if **paymentStatus** is **CANCELLED**, means order is cancelled.
- If the value of **result.resultStatus** is **F**, that means the payment status inquiry is fail. When **resultCode** is **ORDER_NOT_EXIST**, it means that the payment is not yet accepted and can be treated as payment failure. For the other failure reason, human intervention is recommended.
- If the value of **result.resultStatus** is **U**, that means unknown exception occur on wallet side, merchant may try again.

Result

| | | | | --- | --- | --- | --- | | **No** | **resultStatus** | **resultCode** | **resultMessage** | | 1 | S | SUCCESS | Success. | | 2 | U | UNKNOWN_EXCEPTION | An API calling is failed, which is caused by unknown reasons. | | 3 | U | REQUEST_TRAFFIC_EXCEED_LIMIT | The request traffic exceeds the limit. | | 4 | F | ORDER_NOT_EXIST | The order doesn't exist. | | 5 | F | INVALID_API | The called API is invalid or not active. | | 6 | F | PARAM_ILLEGAL | Illegal parameters. For example, non-numeric input, invalid date. | | 7 | F | PROCESS_FAIL | A general business failure occurred. Don't retry. | | 8 | F | ACCESS_DENIED | The access is denied. | | 9 | F | EXPIRED_AGENT_TOKEN | The access token of mini program is expired. | | 10 | F | INVALID_AGENT_TOKEN | The access token of mini program is invalid. |

Sample

Example: A Russian user (Bob) bought a 100 USD product on the e-commerce platform, paid by credit card and submitted the payment synchronously, asynchronous polling payment results.

1. The Mini Program calls my.tradePay interface to do payment (Step 1).
2. E-wallet App returns payment result to the Mini Program (Step 5).
3. Also e-wallet notifies the payment result with paymentNotifyUrl provided by merchant (Step 4).
4. Besides the merchant could call /v1/payments/inquiryPayment interface to query the payment result (Step 6).
5. E-wallet will return payment status inquiry result to the merchant server (step 7).

Request

A. Inquiry By paymentRequestId

copy

```
{
 "paymentRequestId":"1022172000000000001xxxx",
 "partnerId":"20200101234567890132xxxx",
```

```
 "extendInfo": "{\"customerBelongsTo\":\"siteNameExample\"}"
}
```

B. Inquiry By paymentId

copy

```
{
 "paymentId":"1022172000000000001xxxx",
 "partnerId":"20200101234567890132xxxx",
 "extendInfo": "{\"customerBelongsTo\":\"siteNameExample\"}"
}
```

- **paymentId** the unique Id of a payment generated by Wallet.
- **paymentRequestId** the unique Id of a payment generated by Wallet merchants.
- **partnerId** the partnerId allocated by Wallet.
- **extendInfo,** includes key - **customerBelongsTo** the e-wallet that the customer uses. Corresponding to the field 'siteName' that obtained from the API 'my.getSiteInfo', in the Mini Program scenario this is mandatory.

Note:

This interface support querying with **paymentId** or **paymentRequestId**. **paymentId** has a higher priority than **paymentRequestId**, which means that if you offer both **paymentId** and **paymentRequestId**, we will use **paymentId** and ignore **paymentRequestId**.

Response

copy

```
{
 "result": {
    "resultCode":"SUCCESS",
    "resultStatus":"S",
    "resultMessage":"success"
  },
 "paymentId":"20200101234567890133333xxxx",
 "paymentRequestId":"20200101234567890133333xxxx",
 "paymentTime": "2020-01-01T12:01:01+08:30",
 "paymentAmount":{
    "value":"100",
    "currency":"USD"
 },
 "paymentStatus":"SUCCESS"
}
```

- **result . resultStatus==S** shows that the inquiry is successful.
- **paymentId** the unique Id of a payment generated by Wallet.
- **paymentRequestId** the unique Id of a payment generated by Wallet merchants.

- **paymentTime** describes the date time of the successful Wallet payment.
- **paymentAmount** describes the payment amount.
- **paymentStatus** describes the payment status.
- **paymentStatus.PROCESSING** order is not paid or is paid but not finish.
- **paymentStatus.SUCCESS** order is succeeded.
- **paymentStatus.FAIL** order is failed.
- **paymentStatus.CANCELLED** order is cancelled.
- **paymentFailReason** describes the payment fail reason when **paymentStatus= FAIL.**
- **authExpiryTime** describes the payment authorization expiry time, when payment order is **paymentFactor.isAuthorizationPayment=true.**

Source: https://miniprogram.gcash.com/docs/miniprogram_gcash/mpdev/dl7y4p

---

/v1/payments/inquiryRefund {#/v1/payments/inquiryrefund}

Last updated: 2022-07-03

Path: miniprogram_gcash

/v1/payments/inquiryRefund

2022-07-03 18:44

POST `/v1/payments/inquiryRefund`

The `inquiryRefund` API is used to inquire the refund result, usually when not able to receive the refund result after a long period of time. Such as:

Note:

- After Merchant initiates refund and not able to receive the refund result after a long period of time, it can poll Refund Inquiry interface of AMS.
- Merchant uses InquiryRefund to determine the Refund status in the asynchronous Refund processing scenario.
- Round-robin interval, recommended 5s once, up to 1 minute.

Message structure

Request

| | | | | --- | --- | --- | --- | | **Property** | **Data type** | **Required** | **Description** | | partnerId | String | Yes | The partnerId allocated by wallet.
Max. length: 32 characters. | | refundId | String | No | The unique ID of a refund generated by Wallet.
Max. length: 64 characters. | | refundRequestId | String | No | The unique ID of a refund generated by Merchant.
Max. length: 64 characters. | | extendInfo | String | No | The extend information,wallet and merchant can put extend info here.
Max. length: 4096 characters. |

Response

| | | | | --- | --- | --- | --- | | **Property** | **Data type** | **Required** | **Description** | | result | **Result** | Yes | The request result, which contains information related to the request result, such as status and error codes. | | refundId | String | No | The unique ID of a refund generated by Wallet.
Max. length: 64 characters. | | refundRequestId | String | No | The unique ID of a refund generated by Merchant.
Max. length: 64 characters. | | refundAmount | **Amount** | No | Refund amount for display of user consumption records page. | | refundReason | String | No | Refund reason.
Max. length: 256 characters. | | refundTime | String/Datetime | No | Deduct money from merchant success time, after then will start to refund money to user. which follows the ISO 8601 standard. | | refundStatus | String | No | PROCESSING - refund is processing.
SUCCESS - refund success.
FAIL - refund failed. | | refundFailReason | String | No | The fail reason of refund order when refundStatus is FAIL.
Max. length: 256 characters. | | extendInfo | String | No | The extend information,wallet and merchant can put extend info here.
Max. length: 4096 characters. |

Result Process Logic

For different request results, different actions are to be performed. See the following list for details:

- If the value of **result.resultStatus** is **S**, the refund inquiry is successful. And you have to check **refundStatus**:
- if **refundStatus** is **PROCESSING**, means refund is processing;
- if **refundStatus** is **SUCCESS**, means refund success;
- if **refundStatus** is **FAIL**, means refund failed.
- If the value of **result.resultStatus** is **F**, the refund inquiry is fail. When **resultCode** is **REFUND_NOT_EXIST**, it means that the refund is not yet accepted and can be treated as refund failure. For the other failure reason, human intervention is recommended.

- If the value of **result.resultStatus** is **U**, the refund inquiry is unknown exception.processing failure occurs, probably due to system / network issues, merchant can retry.

Result

| | | | | | --- | --- | --- | --- | | **No** | **resultStatus** | **resultCode** | **resultMessage** | | 1 | S | SUCCESS | Success. | | 2 | U | UNKNOWN_EXCEPTION | An API calling is failed, which is caused by unknown reasons. | | 3 | U | REQUEST_TRAFFIC_EXCEED_LIMIT | The request traffic exceeds the limit. | | 4 | F | REFUND_NOT_EXIST | Refund is not exist. | | 5 | F | INVALID_API | The called API is invalid or not active. | | 6 | F | PARAM_ILLEGAL | Illegal parameters. For example, non-numeric input, invalid date. | | 7 | F | PROCESS_FAIL | A general business failure occurred. Don't retry. | | 8 | F | ACCESS_DENIED | The access is denied. | | 9 | F | EXPIRED_AGENT_TOKEN | The access token of mini program is expired. | | 10 | F | INVALID_AGENT_TOKEN | The access token of mini program is invalid. |

Sample

For example, a Korean user purchases a 100 USD merchandise at a Japanese merchant with cross-border payment.

Merchant refund the money, but not return the refund result. so merchant begin to inquiry refund result.

1. User could start refund request from the Mini Program or the merchant cashier （Step 1).
2. The merchant server calls /v1/payments/refund interface to refund (Step 2).
3. E-wallet returns the refund result to the merchant server (Step 3).
4. Also the merchant server could call /v1/payments/inquiryRefund interface to query the refund result (Step 4).
5. E-wallet returns refund inquiry result to the merchant server (Step 5).
6. The merchant should return the refund result to the Mini Program or the merchant cashier (Step 6).

Request

copy

```
{
 "refundId": "1022188000000000001xxxx",
 "refundRequestId":"20200101234567890132xxxx",
 "partnerId":"102217200000000001xxxx",
 "extendInfo": "{\"customerBelongsTo\":\"siteNameExample\"}"
}
```

- **refundId** refundId return by wallet.
- **refundRequestId** the uniqueId of a refund generated by Merchant.
- **partnerId** the partnerId allocated by wallet.

- **extendInfo,** includes key - **customerBelongsTo** the e-wallet that the customer uses. Corresponding to the field 'siteName' that obtained from the API 'my.getSiteInfo'.

Note:

This interface support querying with **refundId** or **refundRequestId**. **paymentId** has a higher priority than **refundRequestId**, which means that if you offer both **refundId** and **refundRequestId**, we will use **refundId** and ignore **refundRequestId**.

Response

copy

```
{
 "result": {
    "resultCode":"SUCCESS",
    "resultStatus":"S",
    "resultMessage":"success"
  },
 "refundId":"20200101234567890144444xxxx",
 "refundRequestId": "20200101234567890155555xxxx",
 "refundAmount":{
    "value":"100",
    "currency":"USD"
 },
 "refundReason":"refund reason.",
 "refundTime":"2020-01-02T12:01:01+08:30",
 "refundStatus":"SUCCESS",
 "refundFailReason":"the fail reason of refund order when refundStatus
is FAIL.",
 "extendInfo":""
}
```

- **result****.resultStatus==S**shows that the refund is successful.
- **refundId**refundId return by wallet.
- **refundRequestId**merchant refund request id.
- **refundAmount**refund amount by merchant.
- **refundReason**describes the refund reason.
- **refundTime**refund process finish time, that means deduct from merchant success.
- **refundStatus**refund Status.
- **refundStatus.PROCESSING:**refund is processing.
- **refundStatus.SUCCESS:**refund success.
- **refundStatus.FAIL:**refund failed.
- **refundFailReason**the fail reason of refund order when refundStatus is Fail.

Source: https://miniprogram.gcash.com/docs/miniprogram_gcash/mpdev/tp3ntm

---

/v1/payments/notifyPayment {#/v1/payments/notifypayment}

Last updated: 2022-07-03

Path: miniprogram_gcash

/v1/payments/notifyPayment

2022-07-03 18:44

POST `/v1/payments/notifyPayment`

The `notifyPayment` API is used to notify payment result to merchant/partner.

Note:

1. to notify merchant/partner the payment result when payment processing reaches the final state (Success/Fail).

2. Not all scenario merchant/partner need receive this notify. Such as sync payment scenario(B Scan C, Agreement Pay).

Message structure

Request

| | | | | --- | --- | --- | --- | | **Property** | **Data type** | **Required** | **Description** | | partnerId | String | Yes | The partnerId allocated by wallet.
Max. length: 32 characters. | | paymentId | String | Yes | The unqiue ID of a payment generated by Wallet.
Max. length: 64 characters. | | paymentRequestId | String | Yes | The unqiue ID of a payment generated by Wallet merchants.
Max. length: 64 characters. | | paymentAmount | **Amount** | Yes | Order amount for display of user consumption records, payment results page. | | paymentTime | String/Datetime | No | Payment success time, which follows the ISO 8601 standard. | | paymentStatus | String | Yes | `SUCCESS` - order is succeeded.
`FAIL` - order is failed. | | paymentFailReason | String | No | The fail reason of payment order when paymentStatus is FAIL.
Max. length: 256 characters. | | extendInfo | String | No | The extend information,wallet and merchant can put extend info here.
Max. length: 4096 characters. |

Response

| | | | | --- | --- | --- | --- | | **Property** | **Data type** | **Required** | **Description** | | result | **Result** | Yes | The request result, which contains information related to the request result, such as status and error codes. |

Result Process Logic

- If result.resultStatus == S, Means the merchant/partner already received this notification.
- If result.resultStatus == F, it means merchant/partner handle this notification failed.
- If result.resultStatus==U, it means merchant/partner handle this notification occur unknown exception, wallet will retry if get U response.
- If other response (almost never occur),wallet will process like U.

Result

| | | | | --- | --- | --- | --- | | **No** | **resultStatus** | **resultCode** | **resultMessage** | | 1 | S | SUCCESS | Success. | | 2 | U | UNKNOWN_EXCEPTION | An API calling is failed, which is caused by unknown reasons. | | 3 | U | REQUEST_TRAFFIC_EXCEED_LIMIT | The request traffic exceeds the limit. | | 4 | F | REPEAT_REQ_INCONSISTENT | Repeated submit, and requests are inconsistent. | | 5 | F | PROCESS_FAIL | A general business failure occurred. Don't retry. | | 6 | F | INVALID_API | The called API is invalid or not active. | | 7 | F | PARAM_ILLEGAL | Illegal parameters. For example, non-numeric input, invalid date. |

Sample

1. The Mini Program calls my.tradePay interface to do payment (Step 1).
2. E-wallet App returns payment result to the Mini Program (Step 5).
3. E-wallet notifies the payment result with paymentNotifyUrl provided by merchant (Step 4).

For example, a wallet user purchases a 100 USD merchandise at a merchant/partner, after user finished payment in wallet cashier page, wallet will send payment status notification to merchant/partner.

Payment

A. Request sample with payment success

copy

```
{
  "partnerId": "P000000000000001xxxx",
  "paymentId": "20191127190741010007000000999xxxx",
  "paymentRequestId": "20191127190741010007000000888881xxxx",
  "paymentAmount": {
```

```
    "currency": "USD",
    "value": "10000"
  },
  "paymentTime": "2019-11-27T12:02:01+08:30",
  "paymentStatus": "SUCCESS"
}
```

- **partnerId** is the idetifier of a merchant/partner, allocated by Wallet.
- **paymentId** is generated by Wallet, uniquely identifies the payment.
- **paymentRequestId** is generated by merchan *t/partner,* uniquely identifies this payment. In payment notify request, paymentRequestId should be the paymentRequestId in origin payment request.
- **paymentAmount** describes the amount of 100 USD already collected by Wallet from user account for this payment.
- **paymentTime** is the success date time of this transaction.
- **paymentStatus** is the payment status in wallet. SUCCESS means transaction already success.

B.

copy

```
{
  "partnerId": "P000000000000001xxxx",
  "paymentId": "20191127190741010007000000999xxxx",
  "paymentRequestId": "201911271907410100070000088881xxxx",
  "paymentAmount": {
    "currency": "USD",
    "value": "10000"
  },
  "paymentCreateTime": "2019-11-27T12:01:01+08:30",
  "paymentTime": "2019-11-27T12:02:01+08:30",
  "paymentStatus": "FAIL",
  "paymentFailReason":"Order payment expired."
}
```

- **paymentStatus** is the payment status in wallet. FAIL means this transaction already failed, usually payment fail are because of this payment already expired .
- **paymentFailReason** used to fill the payment fail reason, only payment status is FAIL will return this parameter.

Response

copy

```
{
 "result": {
    "resultCode":"SUCCESS",
    "resultStatus":"S",
    "resultMessage":"success"
  }
 }
```

- **result.resultStatus==S** shows that merchant/partner already received this notification.

Source: https://miniprogram.gcash.com/docs/miniprogram_gcash/mpdev/edgctc

---

/v1/payments/pay {#/v1/payments/pay}

Last updated: 2022-07-03

Path: miniprogram_gcash

/v1/payments/pay

2022-07-03 18:44

POST `/v1/payments/pay`

The `pay` API is used to initiate a payment at wallet.

Note: A payment which takes place at *wallet*.

1）merchant/partner initiates payment request to *wallet* through Payment Interface.

2）Wallet will handle different payment scenarios base on the parameters in request.

Currently, payment API support following acquiring scenarios:

- Cashier Payment： Usually used in online payment scenario. In this scenario, merchant/partner will call this payment API to create order, and wallet will return cashier page url to merchant/partner, and then redirect to this cashier page. So user can finished payment in cashier page.

Message structure

Request

| | | | | --- | --- | --- | --- | | **Property** | **Data type** | **Required** | **Description** | | partnerId | String | Yes | The partnerId allocated by wallet.
Max. length: 32 characters. | | appId | String | Yes | The mini program ID.
Max. length: 32 characters. | | productCode | String | No | Defined by wallet, wallet will use productCode to get the contract config which include fee,limit info.
Max. length: 32 characters. | | paymentOrderTitle | String | Yes | The order title of this payment.
Max. length: 256 characters. | | paymentRequestId | String | Yes | The unique ID of a payment generated by merchant.

- Max. length: 64 characters.
- This field is used for the idempotence control. For the payment requests which are initiated with the same `paymentRequestId` and reach a final status (S or F), the wallet must return the unique result. | | paymentAmount | **Amount** | Yes | Order amount for display of user consumption records, payment results page. | | paymentMethods | **PaymentMethod** | No | The paymentMethod used to collect fund by wallet. | | paymentAuthCode | String | No | If payFactor.isAgreementPay is true,then it's the accessToken of wallet user, if payFactor.isPaymentCode is true, then it's the authcode of wallet user.
Max. length: 128 characters. | | paymentFactor | **PaymentFactor** | No | In the Mini Program scenario, it is fixed value, map format, {"isCashierPayment" : true}. | | paymentExpiryTime | String/Datetime | No | The payment order close time defined by merchant, which follows the ISO 8601 standard. | | paymentReturnUrl | String | No | The redirect url defined by merchan.
Max. length: 1024 characters. | | paymentNotifyUrl | String | No | The payment success notify url defined by merchant.
Max. length: 1024 characters. | | mcc | String | No | The merchant category code.
Max. length: 32 characters. | | extraParams | Map | No | Map format, specific payment ability which provided by wallet, now we only support 1 key : ORDER. | | extendInfo | String | No | The extend information,wallet and merchant can put extend info here.
Max. length: 4096 characters. | | envInfo | **EnvInfo** | No | Environment information of mobile phone. |

Response

| | | | | | --- | --- | --- | --- | | **Property** | **Data type** | **Required** | **Description** | | result | **Result** | Yes | The request result, which contains information related to the request result, such as status and error codes. | | paymentId | String | No | The unique ID of a payment generated by Wallet.
Max. length: 64 characters. | | paymentTime | String/Datetime | No | Payment success time, which follows the ISO 8601 standard. | | actionForm | **ActionForm** | No | | | authExpiryTime | String/Datetime | No | Authorization expiry time, has value only when paymentFactor.isAuthorizationPayment is true. | | extendInfo | String | No | The extend information,wallet and merchant can put extend info here.
Max. length: 4096 characters. |

Result process logic

For different request results, different actions are to be performed. See the following list for details:

- **result.resultStatus = S**
- That means this transaction is success, merchant/partner can update transaction to success. What need to notice is :
- In payment evaluation scenario, 'S' just means evaluate success, no real fund transfer.
- In authorization payment scenario, 'S' just means authorization success, need wait for capture operation to finish the transaction(finish final fund flow).
- **result.resultStatus = A**

- That means transaction already accept by wallet. Merchant/partner need continue the next step operation according to actionForm response. Such as display order code to user or redirect to wallet cashier page.

- **result.resultStatus = F**

- That means this transaction is failed, the failed reason can refer to result code param. Usually F transactions can not be success again if use the same payment request to call wallet.

- **result.resultStatus = U**

- That means unknown exception occur on wallet side. Merchant/partner can inquiry payment result or waiting for payment status notification to get the real payment result. What need to notice is :

- Payment evaluation scenario can not inquiry.

- U status can not set to fail or success on merchant/partner system.

- U status can not refund to user by offline(Maybe will make fund loss).

Result

| | | | | --- | --- | --- | | **resultStatus** | **resultCode** | **resultMessage** | | S | SUCCESS | Success. | | U | UNKNOWN_EXCEPTION | An API calling is failed, which is caused by unknown reasons. | | U | REQUEST_TRAFFIC_EXCEED_LIMIT | The request traffic exceeds the limit. | | U | PAYMENT_IN_PROCESS | The payment is still under process. | | A | ACCEPT | Need next action according to actionForm. | | F | REPEAT_REQ_INCONSISTENT | Repeated submit, and requests are inconsistent. | | F | PROCESS_FAIL | A general business failure occurred. Don't retry. | | F | INVALID_API | The called API is invalid or not active. | | F | PARAM_ILLEGAL | Illegal parameters exist. For example, a non-numeric input, or an invalid date. | | F | ACCESS_DENIED | The access is denied. | | F | PAYMENT_AMOUNT_EXCEED_LIMIT | Payment amount exceeds limit. | | F | USER_NOT_EXIST | User not exist. | | F | USER_STATUS_ABNORMAL | The user status is abnormal. | | F | USER_BALANCE_NOT_ENOUGH | User balance is not enough for this payment. | | F | PARTNER_NOT_EXIST | Partner not exist. | | F | PARTNER_STATUS_ABNORMAL | Partner status abnormal. | | F | RISK_REJECT | Risk reject. | | F | CURRENCY_NOT_SUPPORT | The currency is not supported. | | F | ORDER_STATUS_INVALID | Order is in invalid status such closed. | | F | INVALID_ACCESS_TOKEN | Invalid accesstoken. | | F | USER_AMOUNT_EXCEED_LIMIT | Payment amount exceeds user's amount limit. | | F | EXPIRED_ACCESS_TOKEN | The access token is expired. | | F | AUTH_CODE_ALREADY_USED | Auth code already used. | | F | INVALID_CODE | Auth code illegal. | | F | EXPIRED_AGENT_TOKEN | The access token of mini program is expired. | | F | INVALID_AGENT_TOKEN | The access token of mini program is invalid. |

Sample

Cashier Payment

For example, a user purchases a 100 USD merchandise at the merchant/partner(online merchant usually) , merchant/partner call this payment api to create payment order first, wallet will return payment order id and wallet cashier page url to merchant/partner, then merchant/partner can redirect user to wallet cashier page with my.tradePay api.

1. Firstly the Mini Program create order (Step 1).
2. The merchant server calls /v1/payments/pay interface with paymentNotifyUrl to initiate payment flow (Step 2).
3. E-wallet server returns payment detail information with paymentId to the merchant server (Step 3).
4. The merchant server has to pass through the payment detail information to the Mini Program (step 4).
5. And the Mini Program should call my.tradePay interface to do payment (Step 5).
6. When the payment reaches the final status, e-wallet server notifies the payment result to the merchant server with paymentNotifyUrl provided in Step 2 (Step 8).
7. Also E-wallet App returns payment result to the Mini Program (Step 9).

Request

copy

```
{
    "partnerId": "P000000000000001xxxx",
    "paymentRequestId": "20191127190741010007000000077771xxxx",
    "paymentOrderTitle": "SHOES",
    "productCode": "PC_5800000001",
    "mcc": "4399",
    "paymentAmount": {
        "currency": "USD",
        "value": "10000"
    },
    "paymentFactor": {
        "isCashierPayment": true
    },
    "paymentReturnUrl":"https://www.merchant.com/redirectxxx",
    "paymentNotifyUrl":"https://www.merchant.com/paymentNotifyxxx",
    "extraParams": {
        "ORDER": "
{\"referenceOrderId\":\"ID_000001\",\"orderAmount\":\"
{\\\"currency\\\":\\\"USD\\\",\\\"value\\\":\\\"10000\\\"}\"}"
    },
    "extendInfo": "{\"customerBelongsTo\":\"siteNameExample\"}",
    "envInfo": {
        "osType": "IOS",
        "terminalType": "APP"
    }
}
```

- **partnerId** is the identifier of a merchant/partner, allocated by Wallet.

- **paymentRequestId** is generated by merchant/partner, uniquely identifies the payment. Wallet must make use of paymentRequestId and partnerId for idempotent control. For example, if a payment with paymentRequestId== 20191127190741010007000000077771xxxx and partnerId==P000000000000001xxxx has been processed successfully by Wallet, when merchant/partneruses the same paymentRequestId and partnerId for payment, Wallet will respond with successful payment.
- **productCode** defined by wallet, wallet will use productCode to get the contract config which include fee,limit info.
- **paymentFactor** In the Mini Program scenario, the **PaymentFactor** only have the fixed value: isCashierPayment = true
- **paymentReturnUrl** is the url defined by merchant/partner. In cashier payment scenario, after user finished payment in wallet cashier page, wallet will direct back to merchant base on this URL.
- **paymentNotifyUrl** is the url defined by merchant/partner. In cashier payment scenario, after user finished payment in wallet cashier page, wallet will notify merchant the payment result base on this URL.
- **paymentAmount** describes the amount of 100 USD to be collected by Wallet from user account for this payment.
- **extraParams,** only includes 1 key - **ORDER** now. ORDER describes the order details of the purchase of the 100 USD merchandise by the user at the merchant. Such as Merchant, Buyer, Goods, etc are included in order . The information in the Order is only used to display user's payment result page and transactions history, regulation reporting, etc. It will not make use of the amount in the order for fund operation.
- **extendInfo,** includes key - **customerBelongsTo** the e-wallet that the customer uses. Corresponding to the field 'siteName' that obtained from the API 'my.getSiteInfo', in the Mini Program scenario this is mandatory.

Response

copy

```
{
 "result": {
    "resultCode":"ACCEPT",
    "resultStatus":"A",
    "resultMessage":"accept"
 },
 "paymentId": "string",
 "actionForm":{
    "actionFormType":"REDIRECTION",
    "redirectionUrl":"http://www.merchant.com/cashier?orderId=xxxxxxx"
 }
}
```

- **result.resultStatus ==A** shows that the payment is accept success. After user finish payment in cashier page, payment will change to success.
- **paymentId** is generated by Wallet, uniquely identifies the payment.
- **actionForm** will return cashier page url to merchant/partner, after merchant/partner received accept result, will redirect to this URL.

Source: https://miniprogram.gcash.com/docs/miniprogram_gcash/mpdev/v1_pay

/v1/payments/refund {#/v1/payments/refund}

Last updated: 2022-07-05

Path: miniprogram_gcash

/v1/payments/refund

2022-07-05 23:31

POST `/v1/payments/refund`

The `refund` API is used to initiate a refund of a successful payment,refund a transaction and return money back to payer, the transaction can be refunded partially or fully.The api will return SUCCESS when deduct money from merchant succes.

Note:

1. Merchant/partner submits refund request to wallet directly
2. Wallet will determine if the refund is successful based on its own payment status and respond to Merchant/partner.
3. Can support multiple refund for 1 success payment， but total refund amount can not greater than payment amount.

Message structure

Request

| | | | | --- | --- | --- | --- | | **Property** | **Data type** | **Required** | **Description** | | partnerId | String | Yes | The partnerId allocated by wallet.
Max. length: 32 characters. | | refundRequestId | String | Yes | The unqiueId of a refund generated by Merchant.
- Max. length: 64 characters.
- This field is used for idempotence control. For the refund requests which are initiated with the same `refundRequestId` and reach a final status (S or F), the wallet must return the unique result. | | paymentId | String | No | The payment Id for the corresponding original payment.
Max. length: 64 characters. | | paymentRequestId | String | No | The paymentRequestId for the corresponding original payment.
Max. length: 64 characters. | | refundAmount | **Amount** | Yes | Refund amount. | | refundReason | String | No | Refund reason.

Max. length: 256 characters. | | extendInfo | String | No | The extend information,wallet and merchant can put extend info here.
Max. length: 4096 characters. |

Response

| | | | | | --- | --- | --- | --- | | **Property** | **Data type** | **Required** | **Description** | | result | **Result** | Yes | The request result, which contains information related to the request result, such as status and error codes. | | refundId | String | No | Unique refund order number. It is mandatory when the **result.resultStatus** is **S**.
Max. length: 64 characters. | | refundTime | String/Datetime | No | Deduct money from merchant success time, after then will start to refund money to user. which follows the ISO 8601 standard. It is mandatory when the **result.resultStatus** is **S**. | | extendInfo | String | No | The extend information,wallet and merchant can put extend info here.
Max. length: 4096 characters. |

Result Process Logic

For different request results, different actions are to be performed. See the following list for details:

- If the value of **result.resultStatus** is **S**, the refund is successful, merchant/partner can process as success.
- If the value of **result.resultStatus** is **F**, the refund has failed, such as caused by refund date time exceeding the allowable refund window (**result . resultCode** = **REFUND_WINDOW_EXCEED**), such as refund amount greater than payment amount etc.
- If the value of **result.resultStatus** is **U**, means refund unknown exception, merchant/partner can calls Refund Inquiry Interface/retry Refund Interface to query to get refund result. What need to notice is as follow:
- U status (inquiry/retry still get U) can not set to fail or success on merchant/partner system.
- U status (inquiry/retry still get U) should not refund/charge to user by offline(Maybe will make fund loss).
- If other response (almost never occur),merchant/partner should process like U.

Result

| | | | | | --- | --- | --- | --- | | **No** | **resultStatus** | **resultCode** | **resultMessage** | | 1 | S | SUCCESS | Success. | | 2 | U | UNKNOWN_EXCEPTION | An API calling is failed, which is caused by unknown reasons. | | 3 | U | REQUEST_TRAFFIC_EXCEED_LIMIT | The request traffic exceeds the limit. | | 9 | U | REFUND_IN_PROCESS | Refund is under processing. | | 4 | F | REPEAT_REQ_INCONSISTENT | Repeated submit, and requests are inconsistent. | | 5 | F | PROCESS_FAIL | A general business failure occurred. Don't retry. | | 6 | F | PARAM_ILLEGAL | Illegal parameters exist. For example, a non-numeric input, or an invalid date. | | 7 | F | INVALID_API | The called API is invalid or not active. | | 8 | F | ACCESS_DENIED | The access is denied. | | 10 | F |

PARTNER_STATUS_ABNORMAL | Partner status abnormal. | | 11 | F | ORDER_NOT_EXIST | Order does not exist. | | 12 | F | ORDER_STATUS_INVALID | Order status is invalid. | | 13 | F | REFUND_WINDOW_EXCEED | Exceed Refund window. | | 14 | F | REFUND_AMOUNT_EXCEED | The total refund amount has exceed the payment amount. | | 15 | F | PARTNER_BALANCE_NOT_ENOUGH | The partner balance is not enough. | | 16 | F | CURRENCY_NOT_SUPPORT | The currency is not supported. | | 17 | F | EXPIRED_AGENT_TOKEN | The access token of mini program is expired. | | 18 | F | INVALID_AGENT_TOKEN | The access token of mini program is invalid. |

Sample

For example, a wallet user applies for refund of 100 USD of a successful payment at merchant/partner. So merchant/partner will call refund API to wallet to refund amount to user.

1. User could start refund request from the Mini Program or the merchant cashier （Step 1).
2. The merchant server calls /v1/payments/refund interface to refund (Step 2).
3. E-wallet returns the refund result to the merchant server (Step 3).
4. The merchant should return the refund result to the Mini Program or the merchant cashier (Step 4).

Request

copy

```
{
  "partnerId": "P000000000000001xxxx",
  "refundRequestId": "20191127190741010007000000088881xxxx",
  "paymentId": "2019112719074101000700000009999xxxx",
  "refundAmount": {
    "currency": "USD",
    "value": "10000"
  },
    "extendInfo": "{\"customerBelongsTo\":\"siteNameExample\"}"
}
```

- **partnerId** is the unique identifier of merchant/partner, assigned by wallte.
- **refundRequestId** is the unique ID of this refund request, generated by merchant/partner, merchant/partner should make sure it is unique, because wallet will use **partnerId** and **refundRequestId** to do idempotent process.
- **paymentId** is the payment ID generated by Wallet, which is the unique payment identifier associated with this refund.
- **refundAmount** describes 100 USD should refund to user, refund amount should less than origin payment amount. The amount to pay out for this **refund.refundAmount.currency** and **paymentAmount.currency** in payment request are the same. And if there are multiple refunds for a particular payment, the

total successful refunded amount cannot exceed the payment amount in the payment transaction.
- **extendInfo,** includes key - **customerBelongsTo** the e-wallet that the customer uses. Corresponding to the field ' **siteName**' that obtained from the API ' **my.getSi teInfo**', in the Mini Program scenario this is mandatory.

Note:

- **paymentId** and **paymentRequestId** can not both empty, wallet has to find out the origin payment order based on **paymentId** or **paymentRequestId**.

Response

copy

```
{
  "result": {
    "resultCode": "SUCCESS",
    "resultStatus": "S",
    "resultMessage": "success"
  },
  "refundId": "20191127190741010007000000019000xxxx",
  "refundTime": "2019-11-27T12:01:01+08:30"
}
```

- **result.resultStatus==S** shows that the *Wallet* refund is successful.
- **refundId** is generated by Wallet, uniquely identifies the refund.
- **refundTime** describes the success date time of this refund.

九色鹿

Source: https://miniprogram.gcash.com/docs/miniprogram_gcash/mpdev/v1_refund

---

/v2/authorizations/applyToken {#/v2/authorizations/applytoken}

Last updated: 2021-05-09

Path: miniprogram_gcash

/v2/authorizations/applyToken

2021-05-09 18:43

POST `/v2/authorizations/applyToken`

The `applyToken` API is used to obtain the access token.

This interface can be used in the following cases:

- After the merchant receives **authCode** from Mini-Program, the merchant uses this interface to request the access token from e-wallet. In this scenario, the interface generally needs to be used with the Authorization Prepare Interface.
- When the original token expires, the merchant requests a new access token by using the refresh token. In this scenario, this interface can be used independently.

Message structure

Request

| | | | | | | --- | --- | --- | --- | --- | | **Property** | **Data type** | **Required** | **Description** | **Example** | | authClientId | String | No | The unique identifier allocated for client.
Max. length: 128 characters. | "20201672687387477477 4xxxx" | | grantType | String | Yes | Indicates which parameter is to be used to obtain the access token. Possible values are:
- `AUTHORIZATION_CODE`: the authCode is to be used to retrieve the accessToken
- `REFRESH_TOKEN`: the refreshToken is to be used to retrieve the accessToken
Max. length: 64 characters. | "AUTHORIZATION_CODE" | | customerBelongsTo | String | NO | The e-wallet that the customer uses. Possible values are:
- `TRUEMONEY`
- `ALIPAY_HK`
- `TNG`
- `ALIPAY_CN`
- `GCASH`
- `DANA`
- `KAKAOPAY`
- `BKASH`
If you call the interface through AC, you must pass this parameter. | "TNG" | | authCode | String | No | It is required when **grantType** is **AUTHORIZATION_CODE**.
The authorization code, which is used by confidential and public clients to exchange an authorization code for an access token. After the user returns to the client via the Mini-program API, the Mini-program will get the authorization code from the response of and use it to request an access token.
Max. length: 32 characters. | "28101113011GZcM9CjlF91WH00039190xxxx" | | refreshToken | String | No | **refreshToken** is required when **grantType** is **REFRESH_TOKEN**.
The refresh token, which is used by the auth client to exchange for a new access token when the access token expires. By using the refresh token, new access tokens can be obtained without further interaction with the user.
Max. length: 128 characters. | "28101113011GZcM9CjlF91WH00039190xxxx" | | extendInfo | String | No | The extend information, wallets and merchants can put extending information in this property.
Max. length: 4096 characters. | "This is additional information" |

Response

| | | | | | | --- | --- | --- | --- | --- | | **Property** | **Data type** | **Required** | **Description** | **Example** | | result | **Result** | Yes | The request result, which contains information related to the request result, such as status and error codes. | {
"resultCode": "SUCCESS",
"resultStatus": "S",
"resultMessage": "success"
} | | accessToken | String | No | An access token that can be used to access the user resource scope.
When authorization application is successful **[result.resultStatus == S]**, the auth client might use accessToken to acccess the corresponding user's resource scope.
Max. length: 128 characters. | "281010033AB2F588D14B43238637264FCA5AAF35xxxx" | | accessTokenExpiryTime | String/Datetime | No | Access token expiration time, which follows the ISO 8601 standard. After this time, authClient will not be able to use this token to deduct from user's account.
This parameter must be returned when authorization application is successful **[result.resultStatus == S]**, and the accessToken will be invalid after **accessTokenExpiryTime.** | "2019-06-06T12:12:12+08:00" | | refreshToken | String | No | The refresh token that is used by the auth client to exchange for a new access token when the access token expires. By using the refresh token, new access tokens can be obtained without further interaction with the user.
This parameter must be returned when authorization application is successful **[result.resultStatus == S]**, and the merchant can use the **refreshToken** to request for a new **accessToken.**
Max. length: 128 characters. | "2810100334F62CBC577F468AAC87CFC6C9107811xxxx" | | refreshTokenExpiryTime | String/Datetime | No | Refresh token expiration time, after which the auth client cannot use this token to retrieve a new access token. The value follows the ISO 8601 standard.
This parameter must be returned when authorization application is successful **[result.resultStatus == S]**, and the merchant will not be able to use the refreshToken to retrieve a new **accessToken** after **refreshTokenExpiryTime.** | "2019-06-08T12:12:12+08:00" | | customerId | String | No | Resource owner id, maybe user id, app id of merchant's application, merchant id.
Max. length: 64 characters. | "1000001119398804xxxx" | | extendInfo | String | No | the extend information,wallet and merchant can put extend info here.
Max. length: 4096 characters. | "This is additional information" |

Result process logic

In the response, the `result.resultStatus` field indicates the result of processing a request as follows.

| | | --- | --- | | **resultStatus** | **Decription** | | S | The authorization token application request is successful. The merchant can use the access token to access the corresponding user resource scope.
The corresponding `result.resultCode` is "SUCCESS" and the `result.resultMessage` is "Success". | | A | The request is accepted by wallets.
The corresponding `result.resultCode` is "ACCEPT"; and the `result.resultMessage` varies based on different situations. | | U | The API status is unknown. AuthClient may guide user to try again.
The corresponding `result.resultCode` is "UNKNOWN_EXCEPTION" and

`result.resultMessage` is "An API calling is failed, which is caused by unknown reasons.".
For details, see the Common error codes section. | | F | The request fails to send.AuthClient may guide user to try again.
The corresponding `result.resultCode` and `result.resultMessage` vary based on different situations. For details, see the following Error codes section. |

Error codes

Error codes are usually classified into the following categories:

- Common error codes: are common for all Mini Program OpenAPIs.
- API-specific error codes: are listed in the following table.

| | | | | --- | --- | --- | | **resultStatus** | **resultCode** | **resultMessage** | | F | AUTH_CLIENT_UNSUPPORTED_GRANT_TYPE | The auth client do not support this grant type. | | F | INVALID_AUTH_CLIENT | The auth client is invalid. | | F | INVALID_AUTH_CLIENT_STATUS | Invalid auth client status. | | F | INVALID_REFRESH_TOKEN | The refresh token is invalid. | | F | EXPIRED_REFRESH_TOKEN | The refresh token is expired. | | F | USED_REFRESH_TOKEN | The refresh token has been used. | | F | INVALID_CODE | The authorization code is invalid. | | F | USED_CODE | The authorization code has been used. | | F | EXPIRED_CODE | The authorization code is expired. | | F | REFERENCE_CLIENT_ID_NOT_MATCH | The reference client id does not match. | | F | EXPIRED_AGENT_TOKEN | The access token of mini program is expired. | | F | INVALID_AGENT_TOKEN | The access token of mini program is invalid. |

Sample

The authorization token application is used to exchange the access token based on the code after obtaining the code.

1. The Mini Program calls the my.getAuthCode interface to obtain the authorization code from e-wallet. (Step 1)
2. The e-wallet returns the authorization code to the Mini Program (Step 7)
3. The Mini Program sends authorization code to the merchant server (Step 8)
4. The merchant server calls the applyTokeninterface to apply for the access token from the e-wallet server and the e-wallet server returns the access token and customer ID to the merchant server. The access token should be kept in the merchant server only, which means that it should not be returned to the Mini Program. (Step 9 and Step 11).

Note: Other steps are covered by the e-wallet.

Request

A. Retrieving `accessToken` with `authCode`

copy

```
{
  "authClientId": "20201672687387477 4774xxxx",
  "grantType": "AUTHORIZATION_CODE",
  "authCode": "2810111301lGZcM9CjlF91WH00039190xxxx"
}
```

B. Retrieving `accessToken` **with** `refreshToken`

copy

```
{
  "grantType": "REFRESH_TOKEN",
  "refreshToken": "2810111301lGZcM9CjlF91WH00039190xxxx"
}
```

- **authCode** is from the my.getAuthCode JS-API, you can obtain the authCode in the success callback. when **grantType == AUTHORIZATION_CODE** means that we are requesting for the accessToken by the authCode.
- **refreshToken** is obtained from the response of the previous accessToken Application call. while **grantType == REFRESH_TOKEN** means that we are requesting for the accessToken by providing the refreshToken.

Response

copy

```
{
  "result": {
    "resultCode": "SUCCESS",
    "resultStatus": "S",
    "resultMessage": "success"
  },
  "accessToken": "281010033AB2F588D14B43238637264FCA5AAF35xxxx",
  "accessTokenExpiryTime": "2019-06-06T12:12:12+08:00",
  "refreshToken": "2810100334F62CBC577F468AAC87CFC6C9107811xxxx",
  "refreshTokenExpiryTime": "2019-06-08T12:12:12+08:00",
  "customerId":"1000001119398804xxxx"
}
```

- **result.resultStatus==S** shows that the application is successful,
- AuthClient can make use of **accessToken,** 281010033AB2F588D14B43238637264FCA5AAF35, to access the user's resource scope before 2019-06-06T12:12:12+08:00 **[accessTokenExpiryTime].**
- AuthClient can make use of **refreshToken,** 2810100334F62CBC577F468AAC87CFC6C9107811, to request for a new accessToken before 2019-06-08T12:12:12+08:00 **[refreshTokenExpiryTime].**

Source: https://miniprogram.gcash.com/docs/miniprogram_gcash/mpdev-old/v2_applytoken

---

/v2/authorizations/applyToken {#/v2/authorizations/applytoken}

Last updated: 2022-07-05

Path: miniprogram_gcash

/v2/authorizations/applyToken

2022-07-05 23:31

POST `/v2/authorizations/applyToken`

With this API call, a merchant can obtain an access token from the super app. The merchant is then authorized by users to provide services on the mini program.

> **Note:**
>
> - Before calling this API, call the my.getAuthCode JSAPI to obtain an authorization code from the super app as the request parameter. Then call this API to exchange for an access token from the super app.
> - When the original access token expires, use the refresh token to exchange for a new access token directly. In this scenario, this API can be used independently.
> - An access token should be kept in the merchant server only, which means it should not be returned to the mini program.

Structure

A message consists of a header and body. The following sections are focused on the body structure. For the header structure, see:

- Request header
- Response header

Request parameters

| | | | | | | --- | --- | --- | --- | --- | | **Field** | **Data type** | **Required** | **Description** | **Example** | | appId | String | Yes | Indicates the unique ID assigned by Mini Program Platform to identify a mini program.
- Maximum length: 32 characters
- Characters not allowed: special characters such as @ # ?
Note: Obtain this field via the my.getAppIdSync JSAPI or Mini Program Platform. | "3333010071465913xxx" | | authClientId | String | Yes | Indicates the unique ID assigned by the super app to identify an authorized merchant.
- Maximum length: 128 characters
- Characters not allowed:special characters such as @ # ? |

"20201672687387477 4774xxxx" | | grantType | String | Yes | Indicates the way the authorized merchant obtains an access token. Valid values are:
- AUTHORIZATION_CODE: Exchange for an access token.
- REFRESH_TOKEN: Exchange for a new access token when the original one expires. | "AUTHORIZATION_CODE" | | customerBelongsTo | String | No | Indicates the super app that a user uses. Valid values are:
- ALIPAY_CN: Alipay CN
- ALIPAY_HK: Alipay HK
- ALIPAY_MO: Alipay MO
- TNG: Touch 'n Go
- GCASH: Gcash
- DANA: Dana
- KAKAOPAY: KakaoPay
- BKASH: bKash
- CHOPE: Chope
- TRUEMONEY: TrueMoney | "TNG" | | authCode | String | No | The authorization code is used to exchange for an access token. Mini programs can obtain an authorization code via the **my.getAuthCode** JSAPI and then send it to the merchant. Then the merchant is authorized to use the authorization code to exchange for an access token.
- Maximum length: 64 characters
- Characters not allowed: special characters such as @ # ?
- Can be Null.
Note: This field is required when the value of *grantType* is AUTHORIZATION_CODE. | "28101113011GZcM9CjlF91WH00039190xxxx" | | refreshToken | String | No | The refresh token is used to exchange for a new access token when the original one expires. With the refresh token, a new access token can be obtained without further interaction with the user.
- Maximum length: 128 characters
- Characters not allowed: special characters such as @ # ?
- Can be Null.
Note: This field is required when the value of *grantType* is REFRESH_TOKE. | "28101113011GZcM9CjlF91WH00039190xxxx" | | extendInfo | String | No | Indicates the extended information about this API.
- Maximum length: 4096 characters
- Characters not allowed: special characters such as @ # ?
- Can be Null. | copy

{
 "memo": "memo"
}
 |

Response parameters

| | | | | | | --- | --- | --- | --- | --- | --- | | **Field** | **Data type** | **Required** | **Description** | **Example** | | result | Result | Yes | Indicates the request result such as status and error codes. | copy

{
 "resultCode": "SUCCESS",
 "resultStatus": "S",
 "resultMessage": "success"
}
 | | accessToken | String | No | The access token is used to access user information. For the specific information that can be accessed, see the my.getAuthCode JSAPI.
- Maximum length: 128 characters
- Characters not allowed: special characters such as @ # ?
- Can be Null.
Note: This field must be returned when the authorization request is successful. | "281010033AB2F588D14B43238637264FCA5AAF35xxxx" | | accessTokenExpiryTime | Datetime | No | Indicates when an access token expires.For example, in the payment scenario, once the access token expires, the authorized merchant cannot use this token to

debit the user's account.
The value follows the ISO 8601 standard format. For example, "2019-11-27T12:01:01+08:30".
Note: This field must be returned when the authorization request is successful. | "2019-06-06T12:12:12+08:00" | | refreshToken | String | No | The refresh token is used to exchange for a new access token when the original one expires. With the refresh token, a new access token can be obtained without further interaction with the user.
- Maximum length: 128 characters
- Characters not allowed: special characters such as @ # ?
- Can be `Null`.
Note: This field must be returned when the authorization request is successful. | "2810100334F62CBC577F468AAC87CFC6C9107811xxxx" | | refreshTokenExpiryTime | Datetime | No | Indicates when the refresh token expires. Once the refresh token expires, the authorized merchant cannot use this token to exchange for a new access token.
The value follows the ISO 8601 standard format. For example, "2019-11-27T12:01:01+08:30".
Note: This field must be returned when the authorization request is successful. | "2019-06-08T12:12:12+08:00" | | customerId | String | Yes | Indicates the unique ID assigned by Mini Program Platform to identify a user.
- Maximum length: 64 characters
- Characters not allowed: special characters such as @ # ? | "1000001119398804xxxx" | | extendInfo | String | No | Indicates the extended information about this API.
- Maximum length: 4096 characters
- Characters not allowed: special characters such as @ # ?
- Can be `Null`.
Note:
- The *extendInfo.appCustomerId* assigned by the super app is passed to this field to identify a user.
- The *extendInfo.acqCustomerId* assigned by Alipay merchant service is also passed to this field to uniquely identify a user. | See Response sample for details. |

Result process logic

In the response, the *result.resultStatus* field indicates the result of processing a request. The following table describes each result status:

| | | --- | --- | | **Result status** | **Description** | | S | The authorization request is successful.
The corresponding *result.resultCode* is `SUCCESS` and the *result.resultMessage* is `SUCCESS`. | | U | The status of the authorization request is unknown.
The corresponding *result.resultCode* is `UNKNOWN_EXCEPTION` and *result.resultMessage* is "An API calling is failed, which is caused by unknown reasons.".
For details, see the Common error codes section. | | F | The authorization request is failed.
The corresponding *result.resultCode* and *result.resultMessage* are various based on different situations. For details, see the following Error codes section. |

Error codes

Error codes are usually classified into the following categories:

- Common error codes are common for all mini program OpenAPIs in V2.
- API-specific error codes are listed in the following table.

| | | | | | --- | --- | --- | --- | | **Error code** | **Result status** | **Error message** | **Further action** | | AUTH_CLIENT_UNSUPPORTED_GRANT_TYPE | F | The authorized merchant does not support this grant type. | Use a valid *grantType* such as `AUTHORIZATION_CODE` or `REFRESH_TOKEN`. | | INVALID_AUTH_CLIENT | F | Either the authorized merchant does not exist or the merchant does not onboard to the native app. | Use a valid *authClientId* assigned by the super app. | | INVALID_AUTH_CLIENT_STATUS | F | The status of the authorized merchant is invalid. | Contact technical support to troubleshoot the issue. | | INVALID_REFRESH_TOKEN | F | The refresh token does not exist. | Obtain a new refresh token via this API. | | EXPIRED_REFRESH_TOKEN | F | The refresh token expires. | Obtain a new authorization code from the super app via the **my.getAuthCode** JSAPI and then obtain a new refresh token via this API. | | USED_REFRESH_TOKEN | F | The refresh token has been used. | Obtain a new refresh token via this API. | | INVALID_AUTHCODE | F | The authorization code does not exist. | Obtain a new authorization code from the super app via the **my.getAuthCode** JSAPI. | | USED_AUTHCODE | F | The authorization code has been used. | Obtain a new authorization code from the super app via the **my.getAuthCode** JSAPI. | | EXPIRED_AUTHCODE | F | The authorization code expires. | Obtain a new authorization code from the super app via the **my.getAuthCode** JSAPI. |

Samples

The data flow to obtain an access token is illustrated as below:

1. The mini program calls the **my.getAuthCode** JSAPI to request an authorization code from the super app.
2. The super app processes the request and shows the information that needs to be authorized.
3. The user confirms the authorization in the super app.
4. The super app service processes the authorization information to the super app server.
5. The super app server verifies the authorization information and then generates the authorization code.
6. The super app server returns the authorization code to the super app service.
7. The super app service returns the authorization code to the mini program.
8. The mini program sends the authorization code to the merchant server.
9. The merchant server calls this API to exchange for an access token from the super app server.
10. The super app server verifies the authorization code and generates the access token.
11. The super app returns the access token to the merchant server.

Request

- Use an authorization code to exchange for an access token

copy

```
{
  "appId":"3333010071465913xxx",
  "authClientId": "20201672687387477 4774xxxx",
  "grantType": "AUTHORIZATION_CODE",
```

```
  "authCode": "2810111301lGZcM9CjlF91WH00039190xxxx"
}
```

The mini program (`3333010071465913xxx`) calls the **my.getAuthCode** JSAPI to obtain the authorization code (`2810111301lGZcM9CjlF91WH00039190xxxx`) and then send the authorization code to the merchant (`20201672687387477 4774xxxx`). The merchant uses the authorization code to exchange for an access token as *grantType* is `AUTHORIZATION_CODE`.

- Use a refresh token to exchange for an access token

copy

```
{
  "grantType": "REFRESH_TOKEN",
  "refreshToken": "2810111301lGZcM9CjlF91WH00039190xxxx"
}
```

The value of *grantType* is `REFRESH_TOKEN`, which means the merchant can obtain an access token by the refresh token (`2810111301lGZcM9CjlF91WH00039190xxxx`).

Response

copy

```
{
    "result": {
        "resultCode": "SUCCESS",
        "resultStatus": "S",
        "resultMessage": "success"
    },
    "accessToken": "281010033AB2F588D14B43238637264FCA5AAF35xxxx",
    "accessTokenExpiryTime": "2019-06-06T12:12:12+08:00",
    "refreshToken": "2810100334F62CBC577F468AAC87CFC6C9107811xxxx",
    "refreshTokenExpiryTime": "2019-06-08T12:12:12+08:00",
    "customerId": "1000001119398804xxxx",
    "extendInfo": "
{\"appCustomerId\":\"200xxxx\",\"acqCustomerId\":\"300xxxx\"}"
}
```

- *result.resultStatus* is `S`, which shows the request to obtain an access token is successful.

- The authorized merchant can use the access token (`281010033AB2F588D14B43238637264FCA5AAF35`) before *accessTokenExpiryTime* (`2019-06-06T12:12:12+08:00`).

- `1000001119398804xxxx` is the user who authorizes the merchant.

- The authorized merchant can use the refresh token (`2810100334F62CBC577F468AAC87CFC6C9107811`) to exchange for a new access token before *refreshTokenExpiryTime* (`2019-06-08T12:12:12+08:0`).

- *extendInfo* returns another two IDs to identify the user:
- *appCustomerId*: Indicates the ID assigned by the super app to identify a user.
- *acqCustomerId*: Indicates the ID assigned by Alipay merchant service to identify a user.

Related links

my.getAuthCode

my.getAppIdSync

Source: https://miniprogram.gcash.com/docs/miniprogram_gcash/mpdev/v2_applytoken

---

/v2/payments/pay {#/v2/payments/pay}

Last updated: 2022-07-05

Path: miniprogram_gcash

/v2/payments/pay

2022-07-05 23:31

POST `/v2/payments/pay`

The `pay` API is used to initiate a payment request to wallets.

Note: A payment which takes place at wallets .

1）The merchant/partner initiates a payment request to wallet through this interface.

2）The wallet will handle different payment scenarios based on the parameters in the request.

Currently, this API support following scenarios:

- Cashier payment： Usually used in the online payment scenario. In this scenario, the merchant/partner will call this API to create an order, and the wallet will return the cashier page URL to the merchant/partner, and then redirect to this cashier page. So user can finish the payment in the cashier page.

Message structure

Request

| | | | | | | --- | --- | --- | --- | --- | | **Property** | **Data type** | **Required** | **Description** | **Example** | | appId | String | Yes | The Mini Program app ID.
Max. length: 32 characters. | "3333010071465913xxx" | | productCode | String | No | The product code, `AGREEMENT_PAYMENT, IN_STORE_PAYMENT, CASHIER_PAYMENT`
Max. length: 32 characters. | "CASHIER_PAYMENT" | | salesCode | String | No | Defined by wallets. The wallet uses the `salesCode` to get the contract configuration which includes fee, limitation information, and so on.
Max. length: 32 characters. | "202011271xxx" | | paymentRequestId | String | Yes | The unqiue ID of a payment generated by merchants.
- Max. length: 64 characters.
- This field is used for idempotence control. For the payment requests which are initiated with the same `paymentRequestId` and reach a final status (S or F), the walletmust return the unique result. | "20191127190741010007000000777 71xxxx" | | paymentAmount | **Amount** | Yes | The order amount, displaying users' consumption records, payment results page. | {
"currency": "USD",
"value": "10000"
} | | order | **Order** | No | The purchase order details, such as Merchant, Buyer, Goods, etc. The information in the Order is only used to display user's payment result page and transactions history, regulation reporting, etc. It will not make use of the amount in the order for fund operation. | {
"referenceOrderId":"OrderID_0101010101xxxx",
"orderDescription":"SHOES",
"orderAmount":{
"currency": "USD",
"value": "10000"
},
"orderCreateTime": "2020-01-01T12:01:01+08:30",
"merchant":{
"referenceMerchantId":"M00000000001xxxx",
"merchantMCC":"1405",
"merchantName":"UGG Technology Limited",
"merchantDisplayName":"UGG",
"merchantAddress":{
"region":"MY",
"city":"KL"
}
},
"env": {
"osType": "IOS",
"terminalType": "APP"
}
} | | paymentMethod | **PaymentMethod** | No | It is used to collect fund by wallets. | {
"paymentMethodType":"ID_000001xxxx",
"paymentMethodId":"1"
} | | paymentFactor | **PaymentFactor** | No | In the Mini Program scenario, it is fixed value, map format. | {
"needSurcharge": true,
"isPaymentEvaluation": false
} | | paymentExpiryTime | String/Datetime | No | The payment order close time defined by merchant,

which follows the
ISO 8601 standard. | "2020-06-08T12:12:12+08:00" | | paymentRedirectUrl | String | No | The redirect URL defined by merchants.
Max. length: 1024 characters. | "https://www.merchant.com/redirectxxx" | | paymentNotifyUrl | String | No | The payment success notification URL defined by merchants.
Max. length: 1024 characters. | "https://www.merchant.com/paymentNotifyxxx" | | voidNotifyUrl | String(2048) | No | the url that void notification will be sent to. | | | extendInfo | String | No | The extensive information. The wallet and merchant can put extensive information in this property.
Max. length: 4096 characters. | "This is additional information" |

Response

| | | | | | | --- | --- | --- | --- | --- | | **Property** | **Data type** | **Required** | **Description** | **Example** | | result | **Result** | Yes | The request result, which contains information related to the request result, such as status and error codes. | {
  "resultCode": "SUCCESS",
  "resultStatus": "S",
  "resultMessage": "success"
} | | paymentId | String | No | The unqiue ID of a payment generated by Wallet.
Max. length: 64 characters. | "4374784884773748478499xxxx" | | paymentTime | String/Datetime | No | Payment success time, which follows the ISO 8601 standard. | "2020-01-08T12:12:12+08:00" | | redirectActionForm | **RedirectActionForm** | No | Indicates a redirect URL. | {
  "method":"POST",
  "redirectionUrl":"https://www.wallet.com/cashier?orderId=xxxxxxx"
} | | authExpiryTime | String/Datetime | No | The authorization expiry time, has value only when `paymentFactor.isAuthorizationPayment` is `true` . | "2020-07-08T12:12:12+08:00" | | extendInfo | String | No | The extensive information. The wallet and merchant can put extensive information in this property.
Max. length: 4096 characters. | "This is additional information" |

Result process logic

In the response, the `result.resultStatus` field indicates the result of processing a request as follows.

| | | | --- | --- | | **resultStatus** | **Decription** | | S | The corresponding `result.resultCode` is "SUCCESS" and the `result.resultMessage` is "Success".
That means that this transaction is successful. The merchant/partner can update transaction to success. What needs to note is :
- In the payment evaluation scenario, 'S' just means that the evaluation is successful and there is no real fund transfer.
- In authorization payment scenario, 'S' just means that the authorization is successful, and need to wait for the capture operation to finish the transaction (finish the final fund flow). | | A | The corresponding `result.resultCode` is "ACCEPT"; and the `result.resultMessage` varies based on different situations.
That means that the transaction is already accepted by wallets. The merchant/partner needs to continue the next operation according to the `redirectActionForm` response, such as display the order code to users or redirect to the wallet cashier page. | | U | The corresponding `result.resultCode` is "UNKNOWN_EXCEPTION" and

`result.resultMessage` is "An API calling is failed, which is caused by unknown reasons.". For details, see the Common error codes section.
That means that unknown exception occurs on the wallet side. The merchant/partner can inquiry the payment result or wait for the payment status notification to get the real payment result. What needs to note is :
- Payment evaluation scenario can not be inquiried.
- `U` status can not set to fail or success on the merchant/partner system.
- `U` status can not refund to users by offline (Maybe will make fund loss). | | F | That means this transaction is failed. The corresponding `result.resultCode` and `result.resultMessage` vary based on different situations. For details, see the following Error codes section.
Usually the `F` transactions can not be successful again if use the same payment request to call wallets. |

Error codes

Error codes are usually classified into the following categories:

- Common error codes: are common for all Mini Program OpenAPIs.
- API-specific error codes: are listed in the following table.

| | | | | --- | --- | --- | | **resultStatus** | **resultCode** | **resultMessage** | | U | PAYMENT_IN_PROCESS | The payment is still under process. | | A | ACCEPT | Need next actions according to the `RedirectActionForm` field. | | F | REPEAT_REQ_INCONSISTENT | Repeated submission, and requests are inconsistent. | | F | PAYMENT_AMOUNT_EXCEED_LIMIT | Payment amount exceeds limit. | | F | USER_NOT_EXIST | User not exist. | | F | USER_STATUS_ABNORMAL | The user status is abnormal. | | F | USER_BALANCE_NOT_ENOUGH | User balance is not enough for this payment. | | F | PARTNER_NOT_EXIST | Partner not exist. | | F | PARTNER_STATUS_ABNORMAL | Partner status abnormal. | | F | RISK_REJECT | Risk reject. | | F | CURRENCY_NOT_SUPPORT | The currency is not supported. | | F | ORDER_STATUS_INVALID | Order is in invaid status such closed. | | F | INVALID_ACCESS_TOKEN | Invalid accesstoken. | | F | USER_AMOUNT_EXCEED_LIMIT | Payment amount exceeds user's amount limit. | | F | AUTH_CODE_ALREADY_USED | Auth code already used. | | F | INVALID_CODE | Auth code illegal. | | F | EXPIRED_AGENT_TOKEN | The agent token of mini program is expired. | | F | INVALID_AGENT_TOKEN | The agent token of mini program is invalid. |

Sample

Cashier Payment

For example, a user purchases a 100 USD good at the merchant/partner(online merchant usually) , the merchant/partner calls this API to create the payment order first, the wallet will return the payment order ID and wallet cashier page URL to the merchant/partner,

then merchant/partner can redirect user to wallet cashier page with the `my.tradePay` API.

1. The Mini Program creates an order.
2. The merchant server calls this pay interface with paymentNotifyUrl to initialte payment flow.
3. E-wallet server returns payment detail information with paymentId to the merchant server.
4. The merchant server passes the payment detail information to the Mini Program.
5. The Mini Program calls the my.tradePay interface to conduct the payment.
6. When the payment reaches the final status, the e-wallet server notifies the merchant server of the payment result with paymentNotifyUrl provided in Step 2 (Step 8).
7. Also the E-wallet App returns the payment result to the Mini Program (Step 9).

Request

copy

```
{

    "appId": "3333010071465913xxx",
    "paymentRequestId": "20191127190741010007000000077771xxxx",
    "productCode": "CASHIER_PAYMENT",
    "paymentAmount": {
        "currency": "USD",
        "value": "10000"
    },
    "order":{
      "referenceOrderId":"OrderID_0101010101xxxx",
      "orderDescription":"SHOES",
      "orderAmount":{
        "currency": "USD",
        "value": "10000"
      },
      "orderCreateTime": "2020-01-01T12:01:01+08:30",
      "merchant":{
          "referenceMerchantId":"M00000000001xxxx",
          "merchantMCC":"1405",
          "merchantName":"UGG Technology Limited",
          "merchantDisplayName":"UGG",
          "merchantAddress":{
             "region":"MY",
             "city":"KL"
          }
       },
      "env": {
          "osType": "IOS",
          "terminalType": "APP"
        }
     },
    "paymentRedirectUrl":"https://www.merchant.com/redirectxxx",
```

```
    "paymentNotifyUrl":"https://www.merchant.com/paymentNotifyxxx"
}
```

- **paymentRequestId** is generated by the merchant/partner, which uniquely identifies the payment. Wallet must make use of paymentRequestId for idempotent control. For example, if a payment with paymentRequestId== 20191127190741010007000000077771 has been processed successfully by Wallet, when the merchant/partneruses the same paymentRequestId for payment, Wallet will respond with sucessful payment.
- **productCode** is the product code, including the `IN_STORE_PAYMENT` , `CASHIER_PAYMENT`, and `AGREEMENT_PAYMENT` information
- **paymentRedirectUrl** is the redirect URL defined by the merchant/partner. In the cashier payment scenario, after the user finished payment in the wallet cashier page, the wallet will direct back to merchant based on this URL.
- **paymentNotifyUrl** is the URL defined by the merchant/partner. In the cashier payment scenario, after the user finished payment in the wallet cashier page, the wallet will notify the merchant of the payment result based on this URL.
- **paymentAmount** describes the amount of 100 USD to be collected by Wallet from user's account for this payment.
- **order** describes the order details of the purchase of the 100CNY merchandise by the user at the merchant. Such as Merchant, Buyer, Goods, etc are included in order . The information in the Order is only used to display user's payment result page and transactions history, regulation reporting, etc. It will not make use of the amount in the order for fund operation.

Response

copy

```
{
 "result": {
    "resultCode":"ACCEPT",
    "resultStatus":"A",
    "resultMessage":"accept"
 },
 "paymentId": "437478488477374847 8499xxxx",
 "redirectActionForm":{
    "method":"POST",
    "redirectionUrl":"https://www.wallet.com/cashier?orderId=xxxxxxx"
 }
}
```

- **result.resultStatus ==A** shows that the payment is accept success. After user finish payment in cashier page, payment will change to success.
- **paymentId** is generated by Wallet, uniquely identifies the payment.
- **redirectActionForm** returns the cashier page URLto the merchant/partner. After the merchant/partner receives the accept result, which will be redirected to this URL.

Source: https://miniprogram.gcash.com/docs/miniprogram_gcash/mpdev/v2_pay

---

/v2/payments/pay {#/v2/payments/pay}

Last updated: 2021-05-09

Path: miniprogram_gcash

/v2/payments/pay

2021-05-09 18:43

POST `/v2/payments/pay`

The `pay` API is used to initiate a payment request to wallets.

Note: A payment which takes place at wallets .

1）The merchant/partner initiates a payment request to wallet through this interface.

2）The wallet will handle different payment scenarios based on the parameters in the request.

Currently, this API support following scenarios:

- Cashier payment：Usually used in the online payment scenario. In this scenario, the merchant/partner will call this API to create an order, and the wallet will return the cashier page URL to the merchant/partner, and then redirect to this cashier page. So user can finish the payment in the cashier page.

Message structure

Request

| | | | | | --- | --- | --- | --- | --- | | **Property** | **Data type** | **Required** | **Description** | **Example** | | appId | String | Yes | The Mini Program app ID.
Max. length: 32 characters. | "3333010071465913xxx" | | productCode | String | No | The product code, `AGREEMENT_PAYMENT, IN_STORE_PAYMENT, CASHIER_PAYMENT`
Max. length: 32 characters. | "CASHIER_PAYMENT" | | salesCode | String | No | Defined by wallets. The wallet uses the `salesCode` to get the contract configuration which includes fee, limitation information, and so on.
Max. length: 32 characters. | "202011271xxx" | | paymentRequestId | String | Yes | The unqiue ID of a payment generated by merchants.
- Max. length: 64 characters.
- This field is used for idempotence control. For the payment requests which are initiated with the same `paymentRequestId` and reach a final status (S or F), the must return the unique result. | "20191127190741010007000000077771xxxx" | | paymentAmount | **Amount** | Yes | The order amount, displaying users' consumption records, payment results page. | {

"currency": "USD",
"value": "10000"
} | | order | **Order** | No | The purchase order details, such as Merchant, Buyer, Goods, etc. The information in the Order is only used to display user's payment result page and transactions history, regulation reporting, etc. It will not make use of the amount in the order for fund operation. | {
"referenceOrderId":"OrderID_0101010101xxxx",
"orderDescription":"SHOES",
"orderAmount":{
"currency": "USD",
"value": "10000"
},
"orderCreateTime": "2020-01-01T12:01:01+08:30",
"merchant":{
"referenceMerchantId":"M00000000001xxxx",
"merchantMCC":"1405",
"merchantName":"UGG Technology Limited",
"merchantDisplayName":"UGG",
"merchantAddress":{
"region":"MY",
"city":"KL"
}
},
"env": {
"osType": "IOS",
"terminalType": "APP"
}
} | | paymentMethod | **PaymentMethod** | No | It is used to collect fund by wallets. | {
"paymentMethodType":"ID_000001xxxx",
"paymentMethodId":"1"
} | | paymentFactor | **PaymentFactor** | No | In the Mini Program scenario, it is fixed value, map format. | {
"needSurcharge": true,
"isPaymentEvaluation": false
} | | paymentExpiryTime | String/Datetime | No | The payment order close time defined by merchant,
which follows the ISO 8601 standard. | "2020-06-08T12:12:12+08:00" | | paymentRedirectUrl | String | No | The redirect URL defined by merchants.
Max. length: 1024 characters. | "https://www.merchant.com/redirectxxx" | | paymentNotifyUrl | String | No | The payment success notification URL defined by merchants.
Max. length: 1024 characters. | "https://www.merchant.com/paymentNotifyxxx" | | extendInfo | String | No | The extensive information. The wallet and merchant can put extensive information in this property.
Max. length: 4096 characters. | "This is additional information" |

Response

| | | | | | | --- | --- | --- | --- | --- | | **Property** | **Data type** | **Required** | **Description** | **Example** | | result | **Result** | Yes | The request result, which contains information related to the request result, such as status and error codes. | {
"resultCode": "SUCCESS",
"resultStatus": "S",

"resultMessage": "success"
} | | paymentId | String | No | The unqiue ID of a payment generated by Wallet.
Max. length: 64 characters. | "43747848847737484784 99xxxx" | | paymentTime | String/Datetime | No | Payment success time, which follows the ISO 8601 standard. | "2020-01-08T12:12:12+08:00" | | redirectActionForm | **RedirectActionForm** | No | Indicates a redirect URL. | {
"method":"POST",
"redirectionUrl":"https://www.wallet.com/cashier?orderId=xxxxxxx"
} | | authExpiryTime | String/Datetime | No | The authorization expiry time, has value only when `paymentFactor.isAuthorizationPayment` is `true` . | "2020-07-08T12:12:12+08:00" | | extendInfo | String | No | The extensive information. The wallet and merchant can put extensive information in this property.
Max. length: 4096 characters. | "This is additional information" |

Result process logic

In the response, the `result.resultStatus` field indicates the result of processing a request as follows.

| | | | --- | --- | | **resultStatus** | **Decription** | | S | The corresponding `result.resultCode` is "SUCCESS" and the `result.resultMessage` is "Success".
That means that this transaction is successful. The merchant/partner can update transaction to success. What needs to note is :
- In the payment evaluation scenario, 'S' just means that the evaluation is successful and there is no real fund transfer.
- In authorization payment scenario, 'S' just means that the authorization is successful, and need to wait for the capture operation to finish the transaction (finish the final fund flow). | | A | The corresponding `result.resultCode` is "ACCEPT"; and the `result.resultMessage` varies based on different situations.
That means that the transaction is already accepted by wallets. The merchant/partner needs to continue the next operation according to the `redirectActionForm` response, such as display the order code to users or redirect to the wallet cashier page. | | U | The corresponding `result.resultCode` is "UNKNOWN_EXCEPTION" and `result.resultMessage` is "An API calling is failed, which is caused by unknown reasons.". For details, see the Common error codes section.
That means that unknown exception occurs on the wallet side. The merchant/partner can inquiry the payment result or wait for the payment notification to get the real payment result. What needs to note is :
- Payment evaluation scenario can not be inquiried.
- `U` status can not set to fail or success on the merchant/partner system.
- `U` status can not refund to users by offline (Maybe will make fund loss). | | F | That means this transaction is failed. The corresponding `result.resultCode` and `result.resultMessage` vary based on different situations. For details, see the following Error codes section.
Usually the `F` transactions can not be successful again if use the same payment request to call wallets. |

Error codes

Error codes are usually classified into the following categories:

- Common error codes: are common for all Mini Program OpenAPIs.
- API-specific error codes: are listed in the following table.

| | | | --- | --- | --- | | **resultStatus** | **resultCode** | **resultMessage** | | U | PAYMENT_IN_PROCESS | The payment is still under process. | | A | ACCEPT | Need next actions according to the `RedirectActionForm` field. | | F | REPEAT_REQ_INCONSISTENT | Repeated submission, and requests are inconsistent. | | F | PAYMENT_AMOUNT_EXCEED_LIMIT | Payment amount exceeds limit. | | F | USER_NOT_EXIST | User not exist. | | F | USER_STATUS_ABNORMAL | The user status is abnormal. | | F | USER_BALANCE_NOT_ENOUGH | User balance is not enough for this payment. | | F | PARTNER_NOT_EXIST | Partner not exist. | | F | PARTNER_STATUS_ABNORMAL | Partner status abnormal. | | F | RISK_REJECT | Risk reject. | | F | CURRENCY_NOT_SUPPORT | The currency is not supported. | | F | ORDER_STATUS_INVALID | Order is in invaid status such closed. | | F | INVALID_ACCESS_TOKEN | Invalid accesstoken. | | F | USER_AMOUNT_EXCEED_LIMIT | Payment amount exceeds user's amount limit. | | F | AUTH_CODE_ALREADY_USED | Auth code already used. | | F | INVALID_CODE | Auth code illegal. | | F | EXPIRED_AGENT_TOKEN | The agent token of mini program is expired. | | F | INVALID_AGENT_TOKEN | The agent token of mini program is invalid. |

Sample

Cashier Payment

For example, a user purchases a 100 USD good at the merchant/partner(online merchant usually) , the merchant/partner calls this API to create the payment order first, the wallet will return the payment order ID and wallet cashier page URL to the merchant/partner, then merchant/partner can redirect user to wallet cashier page with the `my.tradePay` API.

1. The Mini Program creates an order.
2. The merchant server calls this pay interface with paymentNotifyUrl to initialte payment flow.
3. E-wallet server returns payment detail information with paymentId to the merchant server.
4. The merchant server passes the payment detail information to the Mini Program.
5. The Mini Program calls the my.tradePay interface to conduct the payment.
6. When the payment reaches the final status, the e-wallet server notifies the merchant server of the payment result with paymentNotifyUrl provided in Step 2 (Step 8).
7. Also the E-wallet App returns the payment result to the Mini Program (Step 9).

Request

copy

```
{

    "appId": "3333010071465913xxx",
```

```
    "paymentRequestId": "20191127190741010007000000077771xxxx",
    "productCode": "CASHIER_PAYMENT",
    "paymentAmount": {
        "currency": "USD",
        "value": "10000"
    },
    "order":{
      "referenceOrderId":"OrderID_0101010101xxxx",
      "orderDescription":"SHOES",
      "orderAmount":{
        "currency": "USD",
        "value": "10000"
      },
      "orderCreateTime": "2020-01-01T12:01:01+08:30",
      "merchant":{
          "referenceMerchantId":"M00000000001xxxx",
          "merchantMCC":"1405",
          "merchantName":"UGG Technology Limited",
          "merchantDisplayName":"UGG",
          "merchantAddress":{
             "region":"MY",
             "city":"KL"
          }
       },
      "env": {
          "osType": "IOS",
          "terminalType": "APP"
        }
     },
    "paymentRedirectUrl":"https://www.merchant.com/redirectxxx",
    "paymentNotifyUrl":"https://www.merchant.com/paymentNotifyxxx"
}
```

- **paymentRequestId** is generated by the merchant/partner, which uniquely identifies the payment. Wallet must make use of paymentRequestId for idempotent control. For example, if a payment with paymentRequestId== 20191127190741010007000000077771 has been processed successfully by Wallet, when the merchant/partneruses the same paymentRequestId for payment, Wallet will respond with sucessful payment.
- **productCode** is the product code, including the `IN_STORE_PAYMENT` , `CASHIER_PAYMENT`, and `AGREEMENT_PAYMENT` information
- **paymentRedirectUrl** is the redirect URL defined by the merchant/partner. In the cashier payment scenario, after the user finished payment in the wallet cashier page, the wallet will direct back to merchant based on this URL.
- **paymentNotifyUrl** is the URL defined by the merchant/partner. In the cashier payment scenario, after the user finished payment in the wallet cashier page, the wallet will notify the merchant of the payment result based on this URL.
- **paymentAmount** describes the amount of 100 USD to be collected by Wallet from user's account for this payment.
- **order** describes the order details of the purchase of the 100CNY merchandise by the user at the merchant. Such as Merchant, Buyer, Goods, etc are included in order . The information in the Order is only used to display user's payment result page and

transactions history, regulation reporting, etc. It will not make use of the amount in the order for fund operation.

Response

copy

```
{
 "result": {
    "resultCode":"ACCEPT",
    "resultStatus":"A",
    "resultMessage":"accept"
 },
 "paymentId": "43747848847737484784 99xxxx",
 "redirectActionForm":{
    "method":"POST",
    "redirectionUrl":"https://www.wallet.com/cashier?orderId=xxxxxxx"
 }
}
```

- **result.resultStatus ==A** shows that the payment is accept success. After user finish payment in cashier page, payment will change to success.
- **paymentId** is generated by Wallet, uniquely identifies the payment.
- **redirectActionForm** returns the cashier page URLto the merchant/partner. After the merchant/partner receives the accept result, which will be redirected to this URL.

Source: https://miniprogram.gcash.com/docs/miniprogram_gcash/mpdev-old/v2_pay

---

/v2/payments/refund {#/v2/payments/refund}

Last updated: 2022-07-05

Path: miniprogram_gcash

/v2/payments/refund

2022-07-05 23:31

POST `/v2/payments/refund`

The `refund` API is used to initiate a refund of a successful payment, refund a transaction and return money to the payer. The transaction can be refunded partially or fully. This API returns `SUCCESS` when deducting money from the merchant is successful.

Note:

1. The merchant/partner submits a refund request to wallets directly

2. Wallets determines whether the refund is successful based on its own payment status and respond to merchants/partners.

3. Multiple refund requests are supported for one successful payment, but the total refund amount can not be greater than the payment amount.

Message structure

Request

| | | | | | --- | --- | --- | --- | --- | | **Property** | **Data type** | **Required** | **Description** | **Example** | | refundRequestId | String | Yes | The unqiue ID of a refund generated by merchants.
- Max. length: 64 characters.
- This field is used for idempotence control. For the refund requests which are initiated with the same `refundRequestId` and reach a final status (S or F), the wallet must return the unique result. | "20191127190741010007000000088881xxxx" | | paymentId | String | No | The unqiue ID of the corresponding original payment.
Max. length: 64 characters. | "20191127190741010007000000999xxxx" | | paymentRequestId | String | No | The paymentRequestId for the corresponding original payment.
Max. length: 64 characters. | "2020010123456789013333xxxx" | | refundAmount | **Amount** | Yes | Refund amount. | {
  "currency": "USD",
  "value": "10000"
} | | refundReason | String | No | Refund reason.
Max. length: 256 characters. | "have returned goods to the shop" | | extendInfo | String | No | The extend information, wallets and merchants can put extending information in this property.
Max. length: 4096 characters. | "This is additional information" |

Response

| | | | | | --- | --- | --- | --- | --- | | **Property** | **Data type** | **Required** | **Description** | **Example** | | result | **Result** | Yes | The request result, which contains information such as status and error codes. | {
  "resultCode": "SUCCESS",
  "resultStatus": "S",
  "resultMessage": "success"
} | | refundId | String | No | Unique refund order number. It is generated by Wallet, which uniquely identifies the refund.
It is mandatory when the **result.resultStatus** is **S**.
Max. length: 64 characters. | "20191127190741010007000000019000xxxx" | | refundTime | String/Datetime | No | Deduct money from merchant success time, after then will start to refund money to user. which follows the ISO 8601 standard. It is mandatory when the **result.resultStatus** is **S**. | "2019-11-27T12:01:01+08:30" | | extendInfo | String | No | The extensive information returned by wallets.
Max. length: 4096 characters. | "This is additional information" |

Result Process Logic

In the response, the `result.resultStatus` field indicates the result of processing a request as follows.

| | | | --- | --- | | **resultStatus** | **Decription** | | S | The corresponding `result.resultCode` is "SUCCESS" and the `result.resultMessage` is "Success".
It means that the refund is successful, the merchant/partner can process as success. | | A | The corresponding `result.resultCode` is "ACCEPT"; and the `result.resultMessage` varies based on different situations. | | U | The corresponding `result.resultCode` is "UNKNOWN_EXCEPTION" and `result.resultMessage` is "An API calling is failed, which is caused by unknown reasons.". For details, see the Common error codes section.
It means that when handling the refund request, an unknown exception occurs. The merchant/partner can call the Refund Inquiry (`inquiryRefund`) API to query or retry this Refund (`refund`) API.
What needs to note is as follow:
- U status (inquiry/retry still gets U) can not set to fail or success on merchant/partner system.
- U status (inquiry/retry still gets U) should not refund/charge to user by offline (Maybe will make fund loss).
If other response (almost never occur), the merchant/partner should process like U. | | F | That means this transaction is failed. The corresponding `result.resultCode` and `result.resultMessage` vary based on different situations.
It means that the refund is failed. The failure reasons can be the followings, but not limited to:
- The refund date time exceeds the allowable refund window (`result.resultCode` = **REFUND_WINDOW_EXCEED**).
- The refund amount is greater than the payment amount.
For details, see the following Error codes section. |

Error codes

Error codes are usually classified into the following categories:

- Common error codes: are common for all Mini Program OpenAPIs.
- API-specific error codes: are listed in the following table.

| | | | | --- | --- | --- | | **resultStatus** | **resultCode** | **resultMessage** | | U | REFUND_IN_PROCESS | Refund is under processing. | | F | REPEAT_REQ_INCONSISTENT | Repeated submit, and requests are inconsistent. | | F | PARTNER_STATUS_ABNORMAL | Partner status abnormal. | | F | ORDER_NOT_EXIST | Order does not exist. | | F | ORDER_STATUS_INVALID | Order status is invalid. | | F | REFUND_WINDOW_EXCEED | Exceed Refund window. | | F | REFUND_AMOUNT_EXCEED | The total refund amount has exceed the payment amount. | | F | PARTNER_BALANCE_NOT_ENOUGH | The partner balance is not enough. | | F | CURRENCY_NOT_SUPPORT | The currency is not supported. | | F | EXPIRED_AGENT_TOKEN | The agent token of Mini Program is expired. | | F | INVALID_AGENT_TOKEN | The agent token of Mini Program is invalid. |

Sample

For example, a wallet user applies for refund of 100 USD of a successful payment at the merchant/partner. So the merchant/partner will call this refund API to the wallet to refund money to users.

1. User can start a refund request from the Mini Program or the merchant cashier.
2. The merchant server calls this refund interface to refund.
3. E-wallet returns the refund result to the merchant server.
4. The merchant should return the refund result to the Mini Program or the merchant cashier.

Request

copy

```
{
  "refundRequestId": "20191127190741010007000000088881xxxx",
  "paymentId": "20191127190741010007000000099999xxxx",
  "refundAmount": {
    "currency": "USD",
    "value": "10000"
  }
}
```

- **refundRequestId** is the unique ID of this refund request, generated by merchant/partner, merchant/partner should make sure it is unique, because wallet will use **refundRequestId** to do idempotent process.
- **paymentId** is the payment ID generated by Wallet, which is the unique payment identifier associated with this refund.
- **refundAmount** describes 100 USD should refund to user, refund amount should less than origin payment amount. The amount to pay out for this **refund.refundAmount.currency** and **paymentAmount.currency** in payment request are the same. And if there are multiple refunds for a particular payment, the total successful refunded amount cannot exceed the payment amount in the payment transaction.

Note:

- **paymentId** and **paymentRequestId** can not both empty, wallet has to find out the origin payment order based on **paymentId** or **paymentRequestId**.

Response

copy

```
{
  "result": {
    "resultCode": "SUCCESS",
    "resultStatus": "S",
```

```
    "resultMessage": "success"
  },
  "refundId": "20191127190741010007000000019000xxxx",
  "refundTime": "2019-11-27T12:01:01+08:30"
}
```

- **result.resultStatus==S** shows that theWallet refund is successful.
- **refundId** is generated by Wallet, uniquely identifies the refund.
- **refundTime** describes the success date time of this refund.

Source: https://miniprogram.gcash.com/docs/miniprogram_gcash/mpdev/v2_refund

---

/v2/payments/refund {#/v2/payments/refund}

Last updated: 2021-05-09

Path: miniprogram_gcash

/v2/payments/refund

2021-05-09 18:43

POST `/v2/payments/refund`

The `refund` API is used to initiate a refund of a successful payment, refund a transaction and return money to the payer. The transaction can be refunded partially or fully. This API returns `SUCCESS` when deducting money from the merchant is successful.

Note:

1. The merchant/partner submits a refund request to wallets directly
2. Wallets determines whether the refund is successful based on its own payment status and respond to merchants/partners.
3. Multiple refund requests are supported for one successful payment, but the total refund amount can not be greater than the payment amount.

Message structure

Request

| | | | | | --- | --- | --- | --- | --- | | **Property** | **Data type** | **Required** | **Description** | **Example** | | refundRequestId | String | Yes | The unqiue ID of a refund generated by merchants.
- Max. length: 64 characters.

- This field is used for idempotence control. For the refund requests which are initiated with the same `refundRequestId` and reach a final status (S or F), the must return the unique result. | "20191127190741010007000000888881xxxx" | | paymentId | String | No | The unqiue ID of the corresponding original payment.
Max. length: 64 characters. | "20191127190741010007000000999xxxx" | | paymentRequestId | String | No | The paymentRequestId for the corresponding original payment.
Max. length: 64 characters. | "20200101234567890133333xxxx" | | refundAmount | **Amount** | Yes | Refund amount. | {
  "currency": "USD",
  "value": "10000"
} | | refundReason | String | No | Refund reason.
Max. length: 256 characters. | "have returned goods to the shop" | | extendInfo | String | No | The extend information, wallets and merchants can put extending information in this property.
Max. length: 4096 characters. | "This is additional information" |

Response

| | | | | | | --- | --- | --- | --- | --- | | **Property** | **Data type** | **Required** | **Description** | **Example** | | result | **Result** | Yes | The request result, which contains information such as status and error codes. | {
  "resultCode": "SUCCESS",
  "resultStatus": "S",
  "resultMessage": "success"
} | | refundId | String | No | Unique refund order number. It is generated by Wallet, which uniquely identifies the refund.
It is mandatory when the **result.resultStatus** is **S**.
Max. length: 64 characters. | "20191127190741010007000000019000xxxx" | | refundTime | String/Datetime | No | Deduct money from merchant success time, after then will start to refund money to user. which follows the ISO 8601 standard. It is mandatory when the **result.resultStatus** is **S**. | "2019-11-27T12:01:01+08:30" | | extendInfo | String | No | The extensive information returned by wallets.
Max. length: 4096 characters. | "This is additional information" |

Result Process Logic

In the response, the `result.resultStatus` field indicates the result of processing a request as follows.

| | | | --- | --- | | **resultStatus** | **Decription** | | S | The corresponding `result.resultCode` is "SUCCESS" and the `result.resultMessage` is "Success".
It means that the refund is successful, the merchant/partner can process as success. | | A | The corresponding `result.resultCode` is "ACCEPT"; and the `result.resultMessage` varies based on different situations. | | U | The corresponding `result.resultCode` is "UNKNOWN_EXCEPTION" and `result.resultMessage` is "An API calling is failed, which is caused by unknown reasons.". For details, see the Common error codes section.
It means that when handling the refund request, an unknown exception occurs. The merchant/partner can call the Refund Inquiry (`inquiryRefund`) API to query or retry this Refund (`refund`) API.
What needs to note is as follow:

- U status (inquiry/retry still gets U) can not set to fail or success on merchant/partner system.
- U status (inquiry/retry still gets U) should not refund/charge to user by offline (Maybe will make fund loss).
If other response (almost never occur), the merchant/partner should process like U. | | F | That means this transaction is failed. The corresponding `result.resultCode` and `result.resultMessage` vary based on different situations.
It means that the refund is failed. The failure reasons can be the followings, but not limited to:
- The refund date time exceeds the allowable refund window (`result.resultCode` = **REFUND_WINDOW_EXCEED**).
- The refund amount is greater than the payment amount.
For details, see the following Error codes section. |

Error codes

Error codes are usually classified into the following categories:

- Common error codes: are common for all Mini Program OpenAPIs.
- API-specific error codes: are listed in the following table.

| | | | | --- | --- | --- | | **resultStatus** | **resultCode** | **resultMessage** | | U | REFUND_IN_PROCESS | Refund is under processing. | | F | REPEAT_REQ_INCONSISTENT | Repeated submit, and requests are inconsistent. | | F | PARTNER_STATUS_ABNORMAL | Partner status abnormal. | | F | ORDER_NOT_EXIST | Order does not exist. | | F | ORDER_STATUS_INVALID | Order status is invalid. | | F | REFUND_WINDOW_EXCEED | Exceed Refund window. | | F | REFUND_AMOUNT_EXCEED | The total refund amount has exceed the payment amount. | | F | PARTNER_BALANCE_NOT_ENOUGH | The partner balance is not enough. | | F | CURRENCY_NOT_SUPPORT | The currency is not supported. | | F | EXPIRED_AGENT_TOKEN | The agent token of Mini Program is expired. | | F | INVALID_AGENT_TOKEN | The agent token of Mini Program is invalid. |

Sample

For example, a wallet user applies for refund of 100 USD of a successful payment at the merchant/partner. So the merchant/partner will call this refund API to the wallet to refund money to users.

1. User can start a refund request from the Mini Program or the merchant cashier.
2. The merchant server calls this refund interface to refund.
3. E-wallet returns the refund result to the merchant server.
4. The merchant should return the refund result to the Mini Program or the merchant cashier.

Request

copy

```
{
  "refundRequestId": "20191127190741010007000000088881xxxx",
  "paymentId": "20191127190741010007000000099999xxxx",
  "refundAmount": {
    "currency": "USD",
    "value": "10000"
  }
}
```

- **refundRequestId** is the unique ID of this refund request, generated by merchant/partner, merchant/partner should make sure it is unique, because wallet will use **refundRequestId** to do idempotent process.
- **paymentId** is the payment ID generated by Wallet, which is the unique payment identifier associated with this refund.
- **refundAmount** describes 100 USD should refund to user, refund amount should less than origin payment amount. The amount to pay out for this **refund.refundAmount.currency** and **paymentAmount.currency** in payment request are the same. And if there are multiple refunds for a particular payment, the total successful refunded amount cannot exceed the payment amount in the payment transaction.

Note:

- **paymentId** and **paymentRequestId** can not both empty, wallet has to find out the origin payment order based on **paymentId** or **paymentRequestId**.

Response

copy

```
{
  "result": {
    "resultCode": "SUCCESS",
    "resultStatus": "S",
    "resultMessage": "success"
  },
  "refundId": "20191127190741010007000000019000xxxx",
  "refundTime": "2019-11-27T12:01:01+08:30"
}
```

- **result.resultStatus==S** shows that theWallet refund is successful.
- **refundId** is generated by Wallet, uniquely identifies the refund.
- **refundTime** describes the success date time of this refund.

Source: https://miniprogram.gcash.com/docs/miniprogram_gcash/mpdev-old/v2_refund

---

/v2/users/inquiryUserInfo {#/v2/users/inquiryuserinfo}

Last updated: 2022-07-05

Path: miniprogram_gcash

/v2/users/inquiryUserInfo

2022-07-05 23:31

POST `/v2/users/inquiryUserInfo`

With this API call, a merchant can inquire about user information with the access token. The specific information you can inquire about is defined in the **my.getAuthCode** JSAPI.

Structure

A message consists of a header and body. The following sections are focused on the body structure. For the header structure, see:

- Request header
- Response header

Request parameters

| | | | | | --- | --- | --- | --- | --- | | **Field** | **Data type** | **Required** | **Description** | **Example** | | appId | String | Yes | Indicates the unique ID assigned by Mini Program Platform to identify a mini program.
- Maximum length: 32 characters
- Characters not allowed: special characters such as @ # ?
Note: Obtain this field via the my.getAppIdSync JSAPI or Mini Program Platform. | "3333010071465913xxx" | | accessToken | String | Yes | The access token is used to access user information. For the specific information that can be accessed, see the my.getAuthCode JSAPI.
- Maximum length: 128 characters
- Characters not allowed: special characters such as @ # ?
Note: Obtain this field via the /v2/authorizations/applyToken API. | "281010033AB2F588D14B43238637264FCA5AAF35xxxx" | | authClientId | String | Yes | Indicates the unique ID assigned by the super app to identify an authorized merchant.
- Maximum length: 128 characters
- Characters not allowed:special characters such as @ # ? | "20201672687387477477 4xxxx" | | extendInfo | String | No | Indicates the extended information about this API.
- Maximum length: 4096 characters

- Characters not allowed: special characters such as @ # ?
- Can be `Null`. | copy
`<br>{<br>  "memo": "memo"<br>}<br>` |

Response parameters

| | | | | | | --- | --- | --- | --- | --- | | **Field** | **Data type** | **Required** | **Description** | **Example** | | result | Result | Yes | Indicates the request result such as status and error codes. | copy
`<br>{<br>  "resultCode":"SUCCESS",<br>  "resultStatus":"S",<br>  "resultMessage":"success"<br>}<br>` | | userInfo | User | No | Indicates the user information that the merchant queried.
Note:
- The *appUserId* assigned by the super app is passed to the *extendInfo* in the *userInfo* to uniquely identify a user.
- The *acqUserId* assigned by Alipay merchant service is passed to the *extendInfo* in the *userInfo* to uniquely identify a user. | See Response sample for details. |

Result process logic

In the response, the *result.resultStatus* field indicates the result of processing a request. The following table describes each result status:

| | | | --- | --- | | **Result status** | **Description** | | S | The inquiry is successful.
Use the `accessToken` to access user information within the corresponding scope.
The corresponding *result.resultCode* is `SUCCESS` and the *result.resultMessage* is `Success`. | | U | The status of the inquiry is unknown.
The corresponding *result.resultCode* is `UNKNOWN_EXCEPTION` and *result.resultMessage* is "An API calling is failed, which is caused by unknown reasons.". For details, see the Common error codes section. | | F | The inquiry is failed.
The corresponding *result.resultCode* and *result.resultMessage* are various based on different situations. For details, see the following Error codes section. |

Error codes

Error codes are usually classified into the following categories:

- Common error codes are common for all mini program OpenAPIs in V2.
- API-specific error codes: are listed in the following table.

| | | | | | --- | --- | --- | --- | | **Error code** | **Result status** | **Error message** | **Further action** | | INVALID_AUTH_CLIENT | F | Either the merchant does not exist or the merchant does not onboard to the native app. | Use a valid *authClientId* assigned by the super app. | | INVALID_ACCESS_TOKEN | F | The access token is not valid. | Obtain a new authorization code via the **my.getAuthCode** JSAPI and then get a valid access token with the authorization code via the **/v2/authorizations/applyToken** API. | | EXPIRED_ACCESS_TOKEN | F | The access token is expired. | Obtain a new access token with a refresh token via the **/v2/authorizations/applyToken** API. |

Samples

The data flow of inquiring about a user's information is illustrated as below:

1. The merchant server calls this API with the access token to inquire about the user's information.
2. The super app server returns the user's information to the merchant server based on the scopes of the access token.

Request

copy

```
{
  "appId": "3333010071465913xxx",
  "accessToken": "281010033AB2F588D14B43238637264FCA5AAF35xxxx",
  "authClientId": "2020167268738747747 74xxxx"

}
```

The merchant (`2020167268738747747 4xxxx`) uses the access token (`281010033AB2F588D14B43238637264FCA5AAF35xxxx`) to access the user's specific information.

Response

copy

```
{
 "result": {
    "resultCode":"SUCCESS",
    "resultStatus":"S",
    "resultMessage":"success"
  },
  "userInfo": {
    "userId": "1000001119398804xxxx",
    "status": "ACTIVE",
    "nickName": "Jack",
    "userName": {
      "fullName": "Jack Sparrow",
      "firstName": "Jack",
      "lastName": "Sparrow"
    },
    "avatar": "http://example.com/avatar.htm?avatarId=FBF16F91-28FB-
47EC-B9BE-27B285C23CD3xxxx",
    "gender": "MALE",
    "birthDate": "2020-07-25",
    "nationality": "US",
    "loginIdInfos": [\
```

```
      {\
        "loginId": "1116874199xxx",\
        "loginIdType": "MOBILE_PHONE"\
      }\
     ],
    "contactInfos": [\
      {\
        "contactNo": "1116874199xxx",\
        "contactType": "MOBILE_PHONE"\
      }\
    ],
    "extendInfo": "
{\"appUserId\":\"200xxxx\",\"acqUserId\":\"300xxxx\"}"
  }
}
```

- *result.resultStatus* is `S`, which means the inquiry request is successful.
- *userInfo* describes the user's information you get via the **inquiryUserInfo** API. The information includes *userId*, *status*, *nickName*, *userName*, *avatar*, *gender*, *birthDate*, *nationality*, *loginIdInfos*, *contactInfos*, and *extendInfo*.

Related links

my.getAuthCode

my.getAppIdSync

/v2/authorizations/applyToken

Source:
https://miniprogram.gcash.com/docs/miniprogram_gcash/mpdev/v2_users_inquiryuserinfo

---

ACSS Reference {#acss-reference}

Last updated: 2021-05-09

Path: miniprogram_gcash

ACSS Reference

2021-05-09 18:43

The **acss** is used to describe the page style. It is a set of style language, used to describe the axml component style and decide the displaying mode of axml component.

To facilitate most front-end developers, our acss has the most features of css. Meanwhile, we extend the css so that it is more suitable for Mini Program.

In contrast to css, our extension features include:

Rpx

The **rpx**(Responsive Pixel) can adapt with screen width specification 750rpx. Take the iPhone6 as an example. The screen width is 375px and has total 750 physical pixels, 750rpx = 375px = 750 physical pixels, 1rpx = 0.5px = 1 physical pixel.

| | | | --- | --- | --- | | **Device | rpx converted to px (screen width / 750) | px converted to rpx (750 / screen width)** | | iPhone5 | 1rpx = 0.42px | 1px = 2.34rpx | | iPhone6 | 1rpx = 0.5px | 1px = 2rpx | | iPhone6 Plus | 1rpx = 0.552px | 1px = 1.81rpx |

Style Import

Use the `@import` statement to import external style sheets. The `@import` should be followed with the relative path of external style sheet, with ";" indicating the end.

Sample codes:

copy

```
/** button.acss **/
.sm-button {
 padding: 5px;
}
```

copy

```
/** app.acss **/
@import "./button.acss";
.md-button {
 padding: 15px;
}
```

The import path supports node_modules directory loading third-party module, such as page.acss:

copy

```
@import "./button.acss"; /*relative path*/
@import "/button.acss"; /*project absolute path*/
@import "third-party/page.acss"; /*third-party npm package path*/
```

Inline Style

The component supports the use of **style**, **class** attribute to control style.

Style Attribute

Used to receive dynamic style, style will be parsed in running. The `!important` priority rule is not supported.

copy

```
<view style="color:{{color}};" />
```

Class Attribute

Used to receive static style. The attribute value is the set of class selector name (style class name) in the style rules. The style class name does not need the ".", and uses space to separate each other.

copy

```
<view class="my-awesome-view" />
```

Static style is uniformly written into class. Do not write static style into style, otherwise the render speed will be affected.

Selector

Same as css3.

Note:

- The class selectors starting with `.a-` and `.am-` are occupied by the system component. Do not use them.
- Attribute selector is not supported.

Global Style and Local Style

The style defined in the app.acss is **global style** and acts on every page.

The style defined in the acss file of the page is **local style** and acts on only the corresponding page. It overwrites the same selector in app.acss.

Page Container Style

The **page** element selector is used to set the page container style, such as the page background color:

copy

```
page {
 background-color: red;
}
```

Local Reference

Please use absolute path to refer local file in ACSS, relative path is not supported.

copy

```
/* Supported */
background-image: url('/images/mini-program.png');
/* Not supported */
background-image: url('./images/mini-program.png');
```

FAQ

Q: How to solve the style pollution when an axml includes multiple custom component or template?

A: Please use namespace to separate them up.

Q: The 100% height is invalid, why?

A: Add absolute position to solve it, or it will adjust the height according to the content height in the page.

Source: https://miniprogram.gcash.com/docs/miniprogram_gcash/mpdev-old/framework_acss-reference

---

ACSS Reference {#acss-reference}

Last updated: 2022-07-03

Path: miniprogram_gcash

ACSS Reference

2022-07-03 18:44

The **acss** is used to describe the page style. It is a set of style language, used to describe the axml component style and decide the displaying mode of axml component.

To facilitate most front-end developers, our acss has the most features of css. Meanwhile, we extend the css so that it is more suitable for Mini Program.

In contrast to css, our extension features include:

Rpx

The **rpx**(Responsive Pixel) can adapt with screen width specification 750rpx. Take the iPhone6 as an example. The screen width is 375px and has total 750 physical pixels, 750rpx = 375px = 750 physical pixels, 1rpx = 0.5px = 1 physical pixel.

| | | | --- | --- | --- | | **Device | rpx converted to px (screen width / 750) | px converted to rpx (750 / screen width)** | | iPhone5 | 1rpx = 0.42px | 1px = 2.34rpx | | iPhone6 | 1rpx = 0.5px | 1px = 2rpx | | iPhone6 Plus | 1rpx = 0.552px | 1px = 1.81rpx |

Style Import

Use the `@import` statement to import external style sheets. The `@import` should be followed with the relative path of external style sheet, with ";" indicating the end.

Sample codes:

copy

```
/** button.acss **/
.sm-button {
 padding: 5px;
}
```

copy

```
/** app.acss **/
@import "./button.acss";
.md-button {
 padding: 15px;
}
```

The import path supports node_modules directory loading third-party module, such as page.acss:

copy

```
@import "./button.acss"; /*relative path*/
@import "/button.acss"; /*project absolute path*/
@import "third-party/page.acss"; /*third-party npm package path*/
```

Inline Style

The component supports the use of **style**, **class** attribute to control style.

Style Attribute

Used to receive dynamic style, style will be parsed in running. The `!important` priority rule is not supported.

copy

```
<view style="color:{{color}};" />
```

Class Attribute

Used to receive static style. The attribute value is the set of class selector name (style class name) in the style rules. The style class name does not need the ".", and uses space to separate each other.

copy

```
<view class="my-awesome-view" />
```

Static style is uniformly written into class. Do not write static style into style, otherwise the render speed will be affected.

Selector

Same as css3.

Note:

- The class selectors starting with `.a-` and `.am-` are occupied by the system component. Do not use them.
- Attribute selector is not supported.

Global Style and Local Style

The style defined in the app.acss is **global style** and acts on every page.

The style defined in the acss file of the page is **local style** and acts on only the corresponding page. It overwrites the same selector in app.acss.

Page Container Style

The **page** element selector is used to set the page container style, such as the page background color:

copy

```
page {
 background-color: red;
}
```

Local Reference

Please use absolute path to refer local file in ACSS, relative path is not supported.

copy

```
/* Supported */
background-image: url('/images/mini-program.png');
```

```
/* Not supported */
background-image: url('./images/mini-program.png');
```

FAQ

Q: How to solve the style pollution when an axml includes multiple custom component or template?

A: Please use namespace to separate them up.

Q: The 100% height is invalid, why?

A: Add absolute position to solve it, or it will adjust the height according to the content height in the page.

Source:
https://miniprogram.gcash.com/docs/miniprogram_gcash/mpdev/framework_acss-reference

---

AMCheckBox {#amcheckbox}

Last updated: 2022-07-03

Path: miniprogram_gcash

AMCheckBox

2022-07-03 18:44

Checkbox

Sample Code

copy

```
// API-DEMO page/component/am-checkbox/am-checkbox.json
{
  "defaultTitle": "Mini Program AntUI component library",
  "usingComponents": {
    "list": "mini-antui/es/list/index",
    "list-item": "mini-antui/es/list/list-item/index",
    "am-checkbox": "mini-antui/es/am-checkbox/index"
  }
}
```

copy

```
<!-- API-DEMO page/component/am-checkbox/am-checkbox.axml -->
<list>
  <view slot="header">
    List + Checkbox
  </view>
  <block a:for="{{items}}">
    <list-item
      thumb=""
      arrow="{{false}}"
      index="{{index}}"
      key="items-{{index}}"
      last="{{index === (items.length - 1)}}"
    >
      <view slot="prefix" style="display: flex; align-items: center;">
        <am-checkbox id="{{item.id}}" data-name="{{item.value}}"
disabled="{{item.disabled}}" checked="{{item.checked}}"
onChange="onChange" />
      </view>
      <label for="{{item.id}}">{{item.title}}</label>
    </list-item>
  </block>
</list>
<view style="padding: 16px;">
  <view style="color: #888; font-size: 14px;">
    Protocol
  </view>
  <view style="margin-top: 10px;">
    <label style="display: flex; line-height: 24px;">
      <am-checkbox />
      <text style="text-indent: 8px; color: #888">Agree with Credit
Payment Service Contract</text>
    </label>
  </view>
</view>
<view style="padding: 16px; background-color: #fff;">
  <form onSubmit="onSubmit" onReset="onReset">
    <view>
      <view style="color: #666; font-size: 14px; margin-bottom:
5px;">Select the framework you have used:</view>
      <view>
        <checkbox-group name="libs">
          <label a:for="{{items2}}" style="display: flex; align-items:
center; height: 30px;">
            <am-checkbox value="{{item.name}}" checked="
{{item.checked}}" disabled="{{item.disabled}}" />
            <text style="color: #888; font-size: 14px; margin-left:
8px;">{{item.value}}</text>
          </label>
        </checkbox-group>
      </view>
      <view style="margin-top: 10px;">
```

```
        <button type="primary" size="mini"
formType="submit">submit</button>
      </view>
    </view>
  </form>
</view>
```

copy

```
// API-DEMO page/component/am-checkbox/am-checkbox.js
Page({
  data: {
    items: [\
      { checked: true, disabled: false, value: 'a', title: 'Checkbox -
Checked by default', id: 'checkbox1' },\
      { checked: false, disabled: false, value: 'b', title: 'Checkbox
- Unchecked by default', id: 'checkbox2' },\
      { checked: true, disabled: true, value: 'c', title: 'Checkbox -
disabled checked by default', id: 'checkbox3' },\
      { checked: false, disabled: true, value: 'd', title: 'Checkbox -
disabled unchecked by default', id: 'checkbox4' },\
    ],
    items2: [\
      { name: 'react', value: 'React', checked: true },\
      { name: 'vue', value: 'Vue.js' },\
      { name: 'ember', value: 'Ember.js' },\
      { name: 'backbone', value: 'Backbone.js', disabled: true },\
    ],
  },
  onSubmit(e) {
    my.alert({
      content: `You are selecting the framework
${e.detail.value.libs.join(', ')}`,
    });
  },
  onReset() {},
  onChange(e) { console.log(e); },
});
```

Attributes

| | | | | | --- | --- | --- | --- | --- | | **Property** | **Description** | **Type** | **Default** | **Required** | | value | Component value, the value carried with the change event when checked. | String | - | No | | checked | Checked or not, allowing the checked setting by default. | Boolean | false | No | | disabled | Disable or not. | Boolean | false | No | | onChange | Callback function triggered by the change event. | (e: Object) => void | - | No | | id | Works with the for attribute of the label component. | String | - | No |

Source: https://miniprogram.gcash.com/docs/miniprogram_gcash/mpdev/extended-component_form_amcheckbox

---

AMIcon {#amicon}

Last updated: 2022-07-03

Path: miniprogram_gcash

AMIcon

2022-07-03 18:44

Icon.

Note:

• It is recommended to use font with small size.

• The Amicon does not support the onTap event, while a view tag can be added to its exterior.

• AMIcon is the extension component encapsulated with customized components where no click event is available.

The click event bound on it will not work. It is possible to find the source codes in modules, and modify the code to add the click event.

Sample Code

copy

```
// API-DEMO page/component/am-icon/am-icon.json
{
  "defaultTitle": "Mini Program AntUI component library",
  "usingComponents": {
    "am-icon": "mini-antui/es/am-icon/index"
  }
}
```

copy

```
<!-- API-DEMO page/component/am-icon/am-icon.axml -->
<view>
  <view class="icon-title">Basics </view>
  <view class="icon-list">
    <block a:for="{{basicTypes}}">
      <view class="icon-item">
        <am-icon type="{{item}}" />
        <text class="icon-desc">{{item}}</text>
      </view>
    </block>
```

```
    </view>
    <view class="icon-title">Outline style</view>
    <view class="icon-list">
      <block a:for="{{strokeTypes}}">
        <view class="icon-item">
          <am-icon type="{{item}}" />
          <text class="icon-desc">{{item}}</text>
        </view>
      </block>
    </view>
    <view class="icon-title">Solid style</view>
    <view class="icon-list">
      <block a:for="{{solidTypes}}">
        <view class="icon-item">
          <am-icon type="{{item}}" />
          <text class="icon-desc">{{item}}</text>
        </view>
      </block>
    </view>
  </view>
```

copy

```
// API-DEMO page/component/am-icom/am-icon.js
Page({
  data: {
    basicTypes: [\
      'arrow-left',\
      'arrow-up',\
      'arrow-right',\
      'arrow-down',\
      'cross',\
      'plus',\
    ],
    strokeTypes: [\
      'close-o',\
      'dislike-o',\
      'heart-o',\
      'help-o',\
      'like-o',\
      'location-o',\
      'info-o',\
      'success-o',\
      'wait-o',\
      'warning-o',\
      'star-o',\
      'download',\
      'friends',\
      'circle',\
      'delete',\
      'charge',\
      'card',\
```

```
        'notice',\
        'qrcode',\
        'reload',\
        'scan',\
        'money',\
        'search',\
        'setting',\
        'share',\
        'zoom-in',\
        'zoom-out',\
      ],
      solidTypes: [\
        'close',\
        'dislike',\
        'heart',\
        'help',\
        'like',\
        'location',\
        'info',\
        'success',\
        'wait',\
        'warning',\
        'star',\
      ],
    },
  });
```

copy

```
  /* API-DEMO page/component/am-icon/am-icon.acss */
  .icon-title {
    margin-top: 20px;
    margin-bottom: 10px;
    margin-left: 10px;
    color: #333;
    font-size: 16px;
  }

  .icon-list {
    background: #fff;
  }

  .icon-item {
    display: inline-flex;
    width: 33.33333%;
    height: 80px;
    align-items: center;
    flex-direction: column;
    justify-content: center;
  }

  .icon-desc {
```

```
  margin-top: 10px;
}
```

Attributes

| | | | | --- | --- | --- | --- | | **Property** | **Description** | **Type** | **Required** | | type | Type of the icon. For specific effect, scan the above QR code to preview (effective values are listed in the table below). | String | Yes | | size | Size of icon, in px. | String | No | | color | Color of icon, same as the color in css. | String | No |

Effective |values of type

| | | --- | --- | | **Style of icon** | **Effective values of type** | | Basic type | arrow-left, arrow-up, arrow-right, arrow-down, cross, plus. | | Outline style | Close-o, dislike-o, heart-o, help-o, like-o, location-o, info-o, success-o, wait-o, warning-o, star-o, download, friends, circle, delete, charge, card, notice, qrcode, reload, scan, money, search, setting, share, zoom-in, dislike-o, heart-o, help-o, like-o, location-o, info-o, success-o, wait-o, warning-o, star-o, download, friends, circle, delete, charge, card, notice, qrcode, reload, scan, money, search, setting, share, zoom-in, zoom-out. | | Solid style | close, dislike, heart, help, like, location, info, success, wait, warning, star. |

Source: https://miniprogram.gcash.com/docs/miniprogram_gcash/mpdev/extended-component_others_amicon

---

AMRadio {#amradio}

Last updated: 2022-07-03

Path: miniprogram_gcash

AMRadio

2022-07-03 18:44

You can use the am-radio component to allow users to select radio buttons. The specific usage is consistent with the basic radio component, but more styles are added in the extended am-switch component.

Sample code

See the sample codes in different languages:

.json

copy

```
{
 "defaultTitle": "am-radio",
 "usingComponents": {
   "am-radio": "mini-ali-ui/es/am-radio/index",
   "list": "mini-ali-ui/es/list/index",
   "list-item": "mini-ali-ui/es/list/list-item/index"
 }
}
```

.axml

copy

```
<view class="page">
 <view class="page-description">AMRdiao</view>
 <view class="page-section">
   <view class="section section_gap">
     <form onSubmit="onSubmit" onReset="onReset">
       <view class="page-section-demo">
         <radio-group class="radio-group" onChange="radioChange"
name="lib">
           <label class="radio" a:for="{{items}}" key="label-
{{index}}">
             <am-radio value="{{item.value}}" checked="
{{item.checked}}" disabled="{{item.disabled}}" />
             <view style="display:inline-block;">{{item.desc}}</text>
           </label>
         </radio-group>
       </view>
       <view class="page-section-demo">
         <radio-group class="radio-group" onChange="radioChange"
name="lib">
           <label class="radio" a:for="{{items1}}" key="label-
{{index}}">
             <am-radio value="{{item.value}}" checked="
{{item.checked}}" disabled="{{item.disabled}}" />
             <view style="display:inline-block;">{{item.desc}}</text>
           </label>
         </radio-group>
       </view>
     </form>
   </view>
 </view>
</view>
```

.acss

copy

```
.radio {
 display: flex; align-items: center;
```

```
}
.page-section-demo {
 padding: 24rpx;
}
```

.js

copy

```
Page({
 data: {
   items: [\
     { checked: true, disabled: false, value: 'a', desc: 'AMRadio-
checked', id: 'checkbox1' },\
     { checked: false, disabled: false, value: 'b', desc: 'AMRadio-
unchecked', id: 'checkbox2' },\
   ],
   items1: [\
     { checked: true, disabled: true, value: 'c', desc: 'AMRadio-
checked disabled', id: 'checkbox3' },\
   ],
 },
 radioChange() {
 },
});
```

Parameters

| | | | | --- | --- | --- | | **Property** | **Type** | **Description** | | value | String | Component value. When the radio is selected, the component value is obtained by the onChange event. | | checked | Boolean | An indicator of whether the radio is selected. The default value is `false`. | | disabled | Boolean | An indicator of whether to disable the radio selection. The default value is `false`. | | id | String | Radio ID, which is used in combination with the *for* property of the label component. |

Source: https://miniprogram.gcash.com/docs/miniprogram_gcash/mpdev/extended-component_form_amradio

---

API Implementation Difference {#api-implementation-difference}

Last updated: 2022-07-03

Path: miniprogram_gcash

API Implementation Difference

2022-07-03 18:44

> Due to the difference of running environment, the implementations of the following APIs are not consistent on the developer tool and the client. Please use real machine for debugging.

| | | --- | --- | | **API** | **Description** | | my.makePhoneCall | Popup prompt simulative call. | | my.navigateBackMiniProgram | Popup prompt simulative call. | | my.navigateToMiniProgram | Popup prompt simulative call. | | my.tradePay | Scheduled. |

Source: https://miniprogram.gcash.com/docs/miniprogram_gcash/mpdev/mini-program-studio_api-implementation-difference

---

AXML Introduction {#axml-introduction}

Last updated: 2022-07-03

Path: miniprogram_gcash

AXML Introduction

2022-07-03 18:44

AXML is a set of markup language for Mini Program framework design and used to describe the structure of Mini Program pages. The AXML syntax falls into five parts: data binding, conditional rendering, list rendering, template and file reference.

AXML code sample:

copy

```
<!-- pages/index/index.axml -->
<view a:for="{{items}}"> {{item}} </view>
<view a:if="{{view == 'WEBVIEW'}}"> WEBVIEW </view>
<view a:elif="{{view == 'APP'}}"> APP </view>
<view a:else> hello </view>
<view onTap="add"> {{count}} </view>
```

.js sample:

copy

```
// pages/index/index.js
Page({
  data: {
    items: [1, 2, 3, 4, 5, 6, 7],
```

```
    view: 'hello',
    count: 1,
  },
  add(e) {
    this.setData({
      count: this.data.count + 1,
    });
  },
});
```

.css sample:

copy

```
/* pages/index/index.acss */
view {
  padding-left: 10px;
}
```

Display result:

Source:
https://miniprogram.gcash.com/docs/miniprogram_gcash/mpdev/framework_axml-reference_axml-introduction

---

AXML Introduction {#axml-introduction}

Last updated: 2021-05-09

Path: miniprogram_gcash

AXML Introduction

2021-05-09 18:43

AXML is a set of markup language for Mini Program framework design and used to describe the structure of Mini Program pages. The AXML syntax falls into five parts: data binding, conditional rendering, list rendering, template and file reference.

AXML code sample:

copy

```
<!-- pages/index/index.axml -->
<view a:for="{{items}}"> {{item}} </view>
<view a:if="{{view == 'WEBVIEW'}}"> WEBVIEW </view>
<view a:elif="{{view == 'APP'}}"> APP </view>
<view a:else> hello </view>
<view onTap="add"> {{count}} </view>
```

.js sample:

copy

```
// pages/index/index.js
Page({
  data: {
    items: [1, 2, 3, 4, 5, 6, 7],
    view: 'hello',
    count: 1,
  },
  add(e) {
    this.setData({
      count: this.data.count + 1,
    });
  },
});
```

.css sample:

copy

```
/* pages/index/index.acss */
view {
  padding-left: 10px;
}
```

Display result:

Source: https://miniprogram.gcash.com/docs/miniprogram_gcash/mpdev-old/framework_axml-reference_axml-introduction

---

About App Container {#about-app-container}

Last updated: 2021-05-09

Path: miniprogram_gcash

About App Container

2021-05-09 18:43

App Container is a productive and secure runtime system that can run beautiful Mini Programs on mobile platforms of Android and iOS in any apps that are integrated with Mini Programs.

Features

You can benefit from the following features available:

Fast Development

You get a uniform and standardized application development experience. Code once, and the Mini Program can run on both Android and iOS apps. App Container provides developers with infrastructure APIs to easily access album, contact, device, storage or network status.

Expressive and Flexible UI

App Container provides rich UI components such as scroll-view, progress, slider, switch, picker, navigator and canvas, which allow developers to realize their idea more quickly. With App Container API, you can easily customize the user experience of title bar, loading view, error view and other UI components.

High Performance

You can easily ship a Mini Program to the app bundle and launch it fast in the offline mode. High Performance Renderer in the App Container allows Mini Program to be rendered fast, so that you can get a fast response.

Contact Us

Send us an email at and integrate App Container today.

Source: https://miniprogram.gcash.com/docs/miniprogram_gcash/mpdev-old/app-container

---

About Mini Program {#about-mini-program}

Last updated: 2022-06-28

Path: miniprogram_gcash

About Mini Program

2022-06-28 23:11

Mini Program is a new technology that helps merchants to quickly develop high-quality services and grow business on mobile apps. It can be easily acquired with a good user experience for mobile users.

What is Mini Program

As is known to all that mobile web technology can help to move the business forward quickly with its flexibility. Mini Program is a mini app that can provide as many services to users as the mobile native app, without installing the mobile app.

The main development language of Mini Program is JavaScript, and the development experience of the Mini Program is similar to that of web development. For developers, the migration cost of moving HTML5 app to Mini Program is kept to a minimum. For more information, see How to transform an HTML 5 mobile app to an HTML 5 mini program.

The following diagram illustrates how mini programs works for developers and users:

Picture 1 Mini Program Overview

Differences between Mini Program development and web development

As shown above, the App Container is the runtime of a DSL(Domain Specific Language) Mini Program. The render engine and JS engine are independent of each other so they can run in parallel. On the other hand, in web development, they run in serial and a longtime-running script may cause non-responsive UI.

Web developers can use DOM APIs provided by the browser to perform DOM selection and operation. In Mini Program development, we practice separation of concerns and make UI rendering and JS business logic into distinct sections. While the UI layout is provided by the declarative `axml` language, the JS business layer does not include DOM manipulation and therefore, JS objects such as `window` or `document` are not available in Mini Program JS layer. This design makes some DOM-dependent JS libraries, such as jQuery, unavailable. Also, the JS layer does not include Node.js runtime, so certain Node.js-specific NPM packages can not be used in the Mini Program project.

Why using Mini Program

As is known to all, an HTML5 app is sandboxed and has its limitation to access native device capability, such as the reliable network, the powerful storage abilities, etc. And users may wait for HTML5 resource downloading for a long time because of poor network connection, which may cause white-screen issues frequently. Besides, smooth transitions can not be achieved in HTML5 since the page switching and the tapping are delayed frequently. Last but not least, the unresponsive UI may hurt the user experience.

You can benefit from Mini Programs with the following features:

- Acquiring App services without installing Apps
- Faster loading
- More powerful capacities
- Almost native experience
- Efficient and simple development

The operating environments of Mini Program

The operating environments of Mini Program include the two major mobile Operating Systems (OSs): iOS and Android. The development tool of Mini Program (IDE) is Mini Program Studio. These three operating environments of Mini Program are also different in their working mechanism, as shown in Table 1:

Table 1. Mini Program operating environment

Components and APIs

Check the Developer's Guide to see the components and APIs for Mini Programs.

Related topics

Product Guide

User Experience Design Guide

Source: https://miniprogram.gcash.com/docs/miniprogram_gcash/about/readme

---

About Mini Program {#about-mini-program}

Last updated: 2022-06-28

Path: miniprogram_gcash

About Mini Program

2022-06-28 23:11

Mini Program is a new technology that helps merchants to quickly develop high-quality services and grow business on mobile apps. It can be easily acquired with a good user experience for mobile users.

What is Mini Program

As is known to all that mobile web technology can help to move the business forward quickly with its flexibility. Mini Program is a mini app that can provide as many services to users as the mobile native app, without installing the mobile app.

The main development language of Mini Program is JavaScript, and the development experience of the Mini Program is similar to that of web development. For developers, the migration cost of moving HTML5 app to Mini Program is kept to a minimum. For more information, see How to transform an HTML 5 mobile app to an HTML 5 mini program.

The following diagram illustrates how mini programs works for developers and users:

Picture 1 Mini Program Overview

Differences between Mini Program development and web development

As shown above, the App Container is the runtime of a DSL(Domain Specific Language) Mini Program. The render engine and JS engine are independent of each other so they can run in parallel. On the other hand, in web development, they run in serial and a longtime-running script may cause non-responsive UI.

Web developers can use DOM APIs provided by the browser to perform DOM selection and operation. In Mini Program development, we practice separation of concerns and make UI rendering and JS business logic into distinct sections. While the UI layout is provided by the declarative `axml` language, the JS business layer does not include DOM manipulation and therefore, JS objects such as `window` or `document` are not available in Mini Program JS layer. This design makes some DOM-dependent JS libraries, such as jQuery, unavailable. Also, the JS layer does not include Node.js runtime, so certain Node.js-specific NPM packages can not be used in the Mini Program project.

Why using Mini Program

As is known to all, an HTML5 app is sandboxed and has its limitation to access native device capability, such as the reliable network, the powerful storage abilities, etc. And users may wait for HTML5 resource downloading for a long time because of poor network connection, which may cause white-screen issues frequently. Besides, smooth transitions can not be achieved in HTML5 since the page switching and the tapping are delayed frequently. Last but not least, the unresponsive UI may hurt the user experience.

You can benefit from Mini Programs with the following features:

- Acquiring App services without installing Apps
- Faster loading
- More powerful capacities
- Almost native experience
- Efficient and simple development

The operating environments of Mini Program

The operating environments of Mini Program include the two major mobile Operating Systems (OSs): iOS and Android. The development tool of Mini Program (IDE) is Mini Program Studio. These three operating environments of Mini Program are also different in their working mechanism, as shown in Table 1:

Table 1. Mini Program operating environment

Components and APIs

Check the Developer's Guide to see the components and APIs for Mini Programs.

Related topics

Product Guide

User Experience Design Guide

Source: https://miniprogram.gcash.com/docs/miniprogram_gcash/about/

Alphabet {#alphabet}

Last updated: 2022-07-03

Path: miniprogram_gcash

Alphabet

2022-07-03 18:44

You can use the alphabet component to arrange content in the alphabetical order.

Sample code

.json

copy

```
{
  "defaultTitle": "Alphabet",
  "usingComponents":{
    "alphabet": "mini-ali-ui/es/list/alphabet/index",
    "am-icon": "mini-ali-ui/es/am-icon/index"
  }
}
```

.axml

copy

```
<view style="position: relative; height: 100vh;">
 <alphabet alphabet="{{alphabet}}" onClick="onAlphabetClick" >
    <view slot="prefix"><am-icon size="12" type="check_"/></view>
 </alphabet>
</view>
```

.js

copy

```
Page({
  data: {
    alphabet: ['A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'J', 'K',
'L', 'M', 'N', 'O', 'P', 'Q', 'R', 'S', 'T', 'U', 'V', 'W', 'X', 'Y',
'Z'],
  },
  onAlphabetClick(ev) {
    my.alert({
      content: JSON.stringify(ev.data),
    });
  },
});
```

Parameters

| | | | --- | --- | --- | | **Property** | **Type** | **Description** | | alphabet | Array | The alphabet that consists of letters. |

Note:

The content is displayed by using the list component.

Source: https://miniprogram.gcash.com/docs/miniprogram_gcash/mpdev/extended-component_others_alphabet

---

Am-Switch {#am-switch}

Last updated: 2022-07-03

Path: miniprogram_gcash

Am-Switch

2022-07-03 18:44

You can use the am-switch component to change the state on or off. The specific usage is consistent with the basic switch component, but more styles are added in the extended am-switch component.

Note:

In iOS, the switch shape is round, while in Android the switch shape is square.

Sample code

See the sample codes in different languages:

.json

copy

```
{
 "defaultTitle": "am-switch",
 "usingComponents": {
   "am-switch": "mini-ali-ui/es/am-switch/index"
 }
}
```

.axml

copy

```
<view class="page">
 <view class="page-description">am-switch</view>
 <view class="page-section">
   <view class="page-section-demo switch-list">
     <view class="switch-item">
       <am-switch checked onChange="switch1Change"/>
     </view>
     <view class="switch-item">
       <am-switch color="red" checked />
     </view>
   </view>
 </view>
</view>
```

.js

copy

```
Page({
 switch1Change(e) {
   console.log('switch1 happen change event  carried value is',
e.detail.value);
 },
});
```

Parameters

| | | | --- | --- | --- | | **Property** | **Type** | **Description** | | name | String | Component name, which is used to obtain data by submitting the form. | | checked | Boolean | An indicator of whether the state is on. The default value is `false`. | | disabled | Boolean | An indicator

of whether to disable the switch. The default value is `false`. | | onChange | (e: Object) => void | The event that is triggered when users change the state. | | color | String | Component color. Specify the color value in CSS. | | controlled | Boolean | An indicator of whether the component is controlled. The default value is `false`. If the value is `true`, the value of *checked* is controlled by *setData*. |

Source: https://miniprogram.gcash.com/docs/miniprogram_gcash/mpdev/extended-component_form_am-switch

---

AmountInput {#amountinput}

Last updated: 2022-07-03

Path: miniprogram_gcash

AmountInput

2022-07-03 18:44

Amount input box

Sample Code

copy

```
// API-DEMO page/component/amount-input/amount-input.json
{
  "defaultTitle": "Mini Program AntUI component library",
  "usingComponents": {
    "amount-input": "mini-antui/es/amount-input/index"
  }
}
```

copy

```
<!-- API-DEMO page/component/amount-input/amount-input.axml -->
<view>
  <amount-input
    type="digit"
    title="Charge amount"
    extra=”Suggest charge amount above ¥100 "
    placeholder="Enter charge amount"
    value="{{value}}"
    maxLength="5"
    focus="{{true}}"
    btnText="Withdraw all"
    onClear="onInputClear"
```

```
    onInput="onInput"
    onConfirm="onInputConfirm" />
</view>
```

copy

```
// API-DEMO page/component/amount-input/amount-input.js
Page({
  data: {
    value: 200,
  },
  onInputClear() {
    this.setData({
      value: '',
    });
  },
  onInputConfirm(e) {
    console.log(e);
    my.alert({
      content: 'confirmed',
    });
  },
  onInput(e) {
    console.log(e);
    const { value } = e.detail;
    this.setData({
      value,
    });
  },
  onButtonClick() {
    my.alert({
      content: 'button clicked',
    });
  },
  onInputFocus(e) {
    console.log(e);
  },
  onInputBlur(e) {
    console.log(e);
  },
});
```

Attributes

| | | | | | --- | --- | --- | --- | --- | | **Property** | **Description** | **Type** | **Default** | **Required** | | type | Type of input, effective values include digit and number. | String | number | No | | title | Title in the upper-left corner. | String | - | No | | extra | Description in the bottom-right corner. | String | - | No | | value | Current value of the input box. | String | - | No | | btnText | Text of the bottom-right corner button. | String | - | No | | placeholder | Placeholder. | String | - | No | | focus | Get cursor automatically. | Boolean | false | No | | onInput | Trigger on keyboard input. | (e: Object) => void | - | No | | onFocus | Trigger on getting focus. | (e:

Object) => void | - | No | | onBlur | Trigger on losing focus. | (e: Object) => void | - | No | | onConfirm | Trigger on clicking keyboard completion. | (e: Object) => void | - | No | | onClear | Trigger on clicking clear icon. | () => void | - | No | | onButtonClick | Trigger on clicking bottom-right corner button. | () => void | - | No | | maxLength | Maximum number of characters allowed for input. | Number | - | No | | controlled | Is controlled component or not If true, value contents are under full control of setData. | Boolean | false | No |

Source: https://miniprogram.gcash.com/docs/miniprogram_gcash/mpdev/extended-component_form_amountinput

---

Analytics {#analytics}

Last updated: 2022-07-07

Path: miniprogram_gcash

Analytics

2022-07-07 17:08

The Analytics functionality helps you to see the performance of all your mini programs. You can see the basic analytics data of all the released mini programs under **Default Analysis**. In addition, you can create events for your customized analysis. Data collected by events are displayed under **My Analysis**.

Note:

Events are used to collect data and analyze user behavior.

Default analysis

The following data are available under the **Default Analysis** tab:

- Summary of your mini programs

You can see the total number of mini programs. You can also sort mini programs according to the status. For example, fully released and grayscale releasing.

- Performance of all the released mini programs

You can analyze mini programs' performance by tracking its unique visitors, page views, new users, and total users.

- Performance of each released mini program

Go to **Performance of Each Mini Program** and click on the name of the mini program you would like to see in detail.

- Performance of each mini program

Under the **Performance** tab, you can see the historical performance of each mini program. See Performance for details.

- Real-time analytics data of each mini program

Under the **Real-Time Analysis** tab, you can see:

- Number of users and page visits of the current day.
- Data on each individual page of the mini program, such as the number of times that the page was shared.
- Ratio between the different versions used by users.
- Payment success rate.

See Real-Time Analysis for details.

Customized analysis

Under the **My Analysis** tab, you can find the list of all the mini programs. You can click on each mini program to customize its analysis in the following ways:

- Manage events for each mini program

Under the **Manage Event** tab, you can customize events or API fields to collect and report data on user behavior. See Event management and analysis for details.

- Manage funnels for each mini program

Under the **Manage Funnel** tab, you can customize funnels to analyze the user conversion. See Funnel management and analysis for details.

More information

Overview

Workflow Procedures

Member Role

Source: https://miniprogram.gcash.com/docs/miniprogram_gcash/platform/analytics

---

Analyze events and funnels {#analyze-events-and-funnels}

Last updated: 2022-07-07

Path: miniprogram_gcash

Analyze events and funnels

2022-07-07 17:08

After you publish an event, you can analyze the event and see user behavior for this event. In addition, you can also see the user flow across several events, which is enabled by funnel analysis. To learn more about events and funnels, see Event management and analysis and Funnel management and analysis.

Events analysis

Prerequisites

Make sure you have defined and published an event. For more information, see define an event and publish an event.

Procedures

1. Go to **Analyti cs > Performance > My Analysis**;
2. Choose the mini program that you want to perform data analysis on, then click **Manage Event**.

3. Define filter conditions for this report.

Take the "add a product to the shopping cart" event as an example. You need to configure the event analysis with the following parameters:

- **Event**: The event to be analyzed. For this event, `addCommodityToSoppingCart` is chosen.
- **Metric**: Metrics are the fields that were previously defined for the event. These are the dimensions that you can analyze for the event. For example, selecting `addToShoppingCarNum` and `Sum` will indicate the total number of products in the shopping cart. In addition, you can also choose PV and UV as metrics.
- **Filter**(Optional): Data filter conditions. You can apply filter conditions to all the fields that you have previously defined for the event.
- **Date Range**: The period when the event was triggered.
- **Time Granularity**: Choose either daily or hourly.

After setting the event analysis configurations, click **Apply**. You can view the analysis results in the form of a graph or table accordingly.

Analysis:

- On March 9, 2021, the total number of products added to the shopping cart is 13;
- On March 10, 2021, the total number of products added to the shopping cart is 26;
- On March 11, 2021, the total number of products added to the shopping cart is 13.

Funnel analysis

To analyze a funnel, complete the following steps:

Prerequisites

Funnel analysis helps you analyze several steps of the user flow. Make sure you have defined and published at least two events. For more information, see define an event and publish an event.

Procedures

1. Create a funnel

1. Go to **Analytics** > **Performance** > **My Analysis**;
2. Choose the mini program that you want to perform data analysis on;
3. Click **Manage Funnel** under the **Manage Funnel** tab;
4. Click **+ New Funnel**.

2. Enter the funnel name

3. Set funnel steps

Take "user shopping path conversion rate" funnel as an example:

- **Steps:** Each step corresponds to an event. You can choose multiple events that are already defined. For this funnel example, select the `addCommodityToShoppingCart`, `submitOrder`, and `payOrder` steps.
- **Filter**(Optional): Data filter conditions. You can apply filter conditions to all the fields that you have previously defined for the event. In step 3, set `payStatus` is equal to `true`, which filters orders that have been successfully paid.

copy

```
data: {
  orderList: [],
  totalItems:0,
  commodityAmount:0,
  orderAmount:0,
  totalDiscount:10,
  shopDiscount:3,
  freightAmount:9,
  orderId:"",
  payStatus:"false",
}
```

After confirming each step, click **Save** to save the current configuration.

4. Analyze data

After setting funnel steps, go back to the funnel management page to set funnel analysis conditions and perform customized data analysis.

- **Funnel:** Select the `userShoppingPathConversionRate` that you have already created.
- **Date Range:** Select the data range to calculate overall funnel conversion. For this funnel example, March 5, 2021 - March 11, 2021 is selected as the range.

After setting funnel analysis conditions, click **Apply**. You can view the analysis results in the form of a graph or table.

Analysis:

- From March 5, 2021 to March 11, 2021, a total of 52 products were added to the shopping cart, of which 32 of which were submitted as orders, and 28 of the submitted orders were successfully paid for.
- The conversion rates among the three events are 61.54% and 87.50% respectively.

Based on the funnel analysis, you can start your investigations on user behavior.

Source: https://miniprogram.gcash.com/docs/miniprogram_gcash/platform/analyze-data

---

App Introduction {#app-introduction}

Last updated: 2022-07-03

Path: miniprogram_gcash

App Introduction

2022-07-03 18:44

Introduction

`App()` represents top-level applications and manages all pages and global data and provides lifecycle method. It is also a construction method, for generating the `App` instance.

A Mini Program is an `App` instance.

Typically, at the root directory of each Mini Program, there are three files.

- app.js: application logic
- app.acss: application style (optional)

- app.json: application configuration

Sample

Here is a simple `app.json`.

copy

```
{
  "pages": [\
    "pages/index/index",\
    "pages/logs/index"\
  ],
  "window": {
    "defaultTitle": "Demo"
  }
}
```

The configuration above indicates that two pages are included in the Mini Program, and the default title is `Demo` for the application.

A simple app.js code is shown below. It has four life-cycle methods.

copy

```
App({
  onLaunch(options) {
    // called when opened
  },
  onShow(options) {
    // called when opened or come foreground
  },
  onHide() {
    // called when it goes background
  },
  onError(msg) {
    // called on JavaScript error or API invoke exception
    console.log(msg)
  },
  // global data
  globalData: {
    foo: true,
  }
})
```

Source:
https://miniprogram.gcash.com/docs/miniprogram_gcash/mpdev/framework_app_overview

---

App Introduction {#app-introduction}

Last updated: 2021-05-09

Path: miniprogram_gcash

App Introduction

2021-05-09 18:43

Introduction

`App()` represents top-level applications and manages all pages and global data and provides lifecycle method. It is also a construction method, for generating the `App` instance.

A Mini Program is an `App` instance.

Typically, at the root directory of each Mini Program, there are three files.

- app.js: application logic
- app.acss: application style (optional)
- app.json: application configuration

Sample

Here is a simple `app.json`.

copy

```
{
  "pages": [\
    "pages/index/index",\
    "pages/logs/index"\
  ],
  "window": {
    "defaultTitle": "Demo"
  }
}
```

The configuration above indicates that two pages are included in the Mini Program, and the default title is `Demo` for the application.

A simple app.js code is shown below. It has four life-cycle methods.

copy

```
App({
  onLaunch(options) {
    // called when opened
  },
  onShow(options) {
    // called when opened or come foreground
  },
  onHide() {
    // called when it goes background
  },
  onError(msg) {
    // called on JavaScript error or API invoke exception
    console.log(msg)
  },
  // global data
  globalData: {
    foo: true,
  }
})
```

Source: https://miniprogram.gcash.com/docs/miniprogram_gcash/mpdev-old/framework_app_overview

---

Apply for an account {#apply-for-an-account}

Last updated: 2021-05-09

Path: miniprogram_gcash

Apply for an account

2021-05-09 18:43

Before developing a Mini Program, you need to create a developer account, which allows you to create, develop or publish your Mini Program.

Please ask the workspace admin to invite you to join the workspace to be a Mini Program admin or a Mini Program developer. Note that developer can develop the Mini Program, and only the Mini Program admin can create and publish the Mini Program.

By clicking the link within the invitation email you received, you can create your account as below.

Fill in the information, then you can finish the account creation process.

Source: https://miniprogram.gcash.com/docs/miniprogram_gcash/mpdev-old/quick-start_create-account

---

Apply for an account {#apply-for-an-account}

Last updated: 2022-07-03

Path: miniprogram_gcash

Apply for an account

2022-07-03 18:44

Before developing a Mini Program, you need to create a developer account, which allows you to create, develop or publish your Mini Program.

Please ask the workspace admin to invite you to join the workspace to be a Mini Program admin or a Mini Program developer. Note that developer can develop the Mini Program, and only the Mini Program admin can create and publish the Mini Program.

By clicking the link within the invitation email you received, you can create your account as below.

Fill in the information, then you can finish the account creation process.

Source: https://miniprogram.gcash.com/docs/miniprogram_gcash/mpdev/quick-start_create-account

---

Approvals {#approvals}

Last updated: 2022-07-07

Path: miniprogram_gcash

Approvals

2022-07-07 17:08

With the Approval Manage functionality, you can query and process approval requests depending on your role.

Features

Depending on your role, you can benefit from all or some of the following features:

- Query approval requests

- Under `Approval Requests`, workspace admins and workspace reviewers can check all the received approval requests in a list and filter them by `Category` or `Status`.
- Under `My Applications`, you can check approval requests that you submitted in a list and filter them by `Category` or `Status`.
- Process approval requests

You can approve or reject the received approval requests. This feature is available for workspace admins and workspace reviewers.

Approval request details

The following table describes the fields of the approval request list:

| | | --- | --- | | **Field** | **Description** | | Title | Short description about the approval request. | | Mini Program name | Mini program name. | | Category | The category that the approval request belongs to. Valid values are:
- Member requests
- Launch/Publishing
- Removal
- Feature activation | | Version | Version of the mini program. The field is required when the value of `Category` is Launch/Publishing. | | Applicant | The person who submits the approval request. | | Created time | The time when the approval request is created. The format is yyyy-MM-dd HH:mm. | | Finish time | The time when the approval request is processed. The format is yyyy-MM-dd HH:mm. | | Status | Status of the approval request. Valid values are:
- In review
- Approved
- Rejected
- Withdrawn |

Table 1. Fields of the approval request list

More information

Overview

Member Role

Workflow Procedures

Manage Mini Program

Manage Workspace

Authorization

Settings

Source: https://miniprogram.gcash.com/docs/miniprogram_gcash/platform/approval

---

Authorization {#authorization}

Last updated: 2022-07-07

Path: miniprogram_gcash

Authorization

2022-07-07 17:08

The Authorization functionality is available for workspace admins. With this functionality, you can check the authorization information and revoke the authorization.

Features

You can benefit from the following features:

- Check the authorization information in the list

You can check the authorization information in the list that includes:

- `Authorized ISV Name`: The authorized ISV name.
- `Authorized Mini Program`: The authorized mini program name.
- `Mini Program ID`: The authorized mini program ID.
- `Status`: The authorization status that can be `Authorized` , `Deauthorized`, `InReview` and `Rejected`.
- `Action`: The action of viewing the authorization details.

Note: For ISVs who would like to apply for authorization, see how to apply for authorization.

- Revoke the authorization

Workspace admin can revoke the authorization for the authorized mini programs.

- Apply for authorization

ISV can can apply for the authorization for the mini programs.

More information

Overview

Member Role

Workflow Procedures

Manage Mini Program

Manage Workspace

How to apply for authorization

Settings

Source: https://miniprogram.gcash.com/docs/miniprogram_gcash/platform/authorization

---

Avatar {#avatar}

Last updated: 2022-07-03

Path: miniprogram_gcash

Avatar

2022-07-03 18:44

You can use the avatar component to display avatars.

Sample code

See the sample codes in different languages:

.js

copy

```
{
  "defaultTitle": "Avatar",
  "usingComponents": {
    "avatar": "mini-ali-ui/es/avatar/index"
  }
}
```

.axml

copy

```
<view>
  <!--normal avatar-->
  <avatar src="xxxx" shape="standard"/>
  <!--avatar with username and short description-->
  <avatar src="xxxx" size="lg" name="username" desc="abstract
description" shape="standard" />
</view>
```

Parameters

| | | | | --- | --- | --- | | **Property** | **Type** | **Description** | | className | String | Customized class. | | src | String | Source of the avatar image. The default value is `Default avatar with blue background.` | | size | String | Avatar size. Valid values are:
- `lg`
- `md`
- `sm`
- `xs`
The default value is `md`. | | shape | String | Avatar shape. Valid values are:
- `standard`
- `circle`
- `square`
The default value is `circle`. | | name | String | Username. | | desc | String | Short description about the user. | | onError | EventHandle | The event that is triggered when the avatar is failed to be loaded. The default value is `(e: Object) => void`. |

Source: https://miniprogram.gcash.com/docs/miniprogram_gcash/mpdev/extended-component_guide_avatar

---

Badge {#badge}

Last updated: 2022-07-03

Path: miniprogram_gcash

Badge

2022-07-03 18:44

Red dot, number or text. Used to tell users there are some updates.

| | | | | | | --- | --- | --- | --- | --- | | **Property** | **Description** | **Type** | **Default** | **Required** | | text | Number or text to be displayed. | String/Number | | No | | dot | Show a red dot instead of number. | Boolean | | No | | overflowCount | Top number to be shown, “+” shown for more. | Number | 99 | No |

Slots

| | | | --- | --- | | **slotName** | **Description** | | inner | Optional, when badge is wrapper, it is used to render the internal region. |

Example

copy

```
{
  "defaultTitle": "AntUI Component Library",
  "usingComponents": {
    "list": "mini-antui/es/list/index",
    "list-item": "mini-antui/es/list/list-item/index",
    "badge": "mini-antui/es/badge/index"
  }
}
```

copy

```
<view>
  <list>
    <block a:for="{{items}}">
      <list-item
        arrow="{{true}}"
        index="{{index}}"
        key="items-{{index}}"
        last="{{index === (items.length - 1)}}"
      >
        <view>
          <badge a:if="{{item.isWrap}}" text="{{item.text}}" dot="
{{item.dot}}">
            <view slot="inner" style="height: 26px; width: 26px;
background-color: #ddd;"></view>
          </badge>
          <text style="margin-left: \{\{ item.isWrap ? '12px' : '0'
\}\}">{{item.intro}}</text>
        </view>
        <view slot="extra">
          <badge a:if="{{item.isWrap}}" text="{{item.text}}" dot="
{{item.dot}}" overflowCount="{{item.overflowCount}}" />
        </view>
      </list-item>
    </block>
  </list>
</view>
```

copy

```
Page({
  data: {
    items: [\
      {\
        dot: true,\
        text: '',\
        isWrap: true,\
        intro: 'Dot Badge',\
      },\
      {\
        dot: false,\
        text: 1,\
        isWrap: true,\
        intro: 'Text Badge',\
      },\
      {\
        dot: false,\
        text: 99,\
        isWrap: false,\
        intro: 'Number Badge',\
      },\
      {\
        dot: false,\
        text: 100,\
        overflowCount: 99,\
        isWrap: false,\
        intro: 'OverflowCount',\
      },\
      {\
        dot: false,\
        text: 'new',\
        isWrap: false,\
        intro: 'Text Badge',\
      },\
    ],
  },
});
```

Source: https://miniprogram.gcash.com/docs/miniprogram_gcash/mpdev/extended-component_prompt-guide_badge

---

Bluetooth API Error Code Table {#bluetooth-api-error-code-table}

Last updated: 2022-07-03

Path: miniprogram_gcash

Bluetooth API Error Code Table

2022-07-03 18:44

Source:
https://miniprogram.gcash.com/docs/miniprogram_gcash/mpdev/API_Device_Bluetooth_BluetoothAPIErrorCodeTable

---

Bluetooth API Error Code Table {#bluetooth-api-error-code-table}

Last updated: 2021-05-09

Path: miniprogram_gcash

Bluetooth API Error Code Table

2021-05-09 18:43

The table below lists the errors for bluetooth APIs. You can find error codes, error messages and solutions to resolve the errors.

| | | | --- | --- | --- | | **Error Code** | **Error Message** | **Solutions** | | 10000 | The Bluetooth adapter is not initialized. | Call API my.openBluetoothAdapter for initialization. | | 10001 | The Bluetooth adapter is not available. | Check whether BLE is supported in your device and enable the function if it's supported. | | 10002 | Device not found | Check the device ID and make sure peripheral broadcast of the target device is enabled. | | 10003 | Connection failed | Check the device ID and make sure peripheral broadcast of the target device is enabled. | | 10004 | Service not found | Check the device ID and make sure the service is available for target devices. | | 10005 | Characteristic not found | Use a correct characteristic ID and make sure the characteristic is enabled for the service. | | 10006 | Connection lost | Disconnect and try again. | | 10007 | Characteristic not supported | Check the read, write and notify functions of the current characteristic. | | 10008 | System error | An unknown system error. | | 10009 | BLE is not supported for Android systems with versions lower than 4.3. | Remind users it's not supported in the current android system version. | | 10010 | Descriptor not found | Use a correct service ID and characteristic ID. | | 10011 | Invalid device ID | Use a correct device ID. | | 10012 | Invalid service ID | Use a correct service ID. | | 10013 | Invalid charateristic ID | Use a correct charateristic ID. | | 10014 | Invalid data | Use valid data. | | 10015 | Timeout | Try again. | | 10016 | Parameters not enough | Check the parameters and try again. | | 10017 | Failed to write charateristic | Make sure writing is supported for peripheral charateristic. Don't disconnect. | | 10018 | Failed to read charateristic | Make sure reading is supported for peripheral charateristic. Don't disconnect. |

Source: https://miniprogram.gcash.com/docs/miniprogram_gcash/mpdev-old/api_device_bluetooth_bluetoothapierrorcodetable

---

Bluetooth API FAQ {#bluetooth-api-faq}

Last updated: 2022-07-03

Path: miniprogram_gcash

Bluetooth API FAQ

2022-07-03 18:44

Q: Does the API call of `my. WriteBLECharacteristicValue` return an empty value?

A: No. Calling this API returns the value of what you write.

Q: Why is listening not supported with the API call of `my.onBLECharacteristicValueChange`? Do I have to write before I can listen?

A: Yes. To call this API, you first need to write before you can listen. In order to prevent multiple callbacks of an event caused by multiple registered event listeners, it is recommended to call off method to listen to an event and close the previous event listener, before you call on method.

Q: Why do I get an error code `10014` with the API call of `my. Why writeBLECharacteristicValue`?

A: The error code `10014` means the data you send are either empty or incorrectly formatted. It is recommended to check for errors in the written data or HEX conversion.

Q: Can I use the hexadecimal array to write characteristic value with the API `my writeBLECharacteristicValue` ?

A: No. The characteristic values you write are hexadecimal strings, which are limited to 20 bytes.

Q: What is the `deviceId` format for Android and iOS devices?

A:

- An Android device gets the MAC address for bluetooth, such as: `11:22:33:44:55:66`.
- An iOS device gets the UUID of Bluetooth, such as: `00000000-0000-0000-0000-000000000000`.

Q: Why can't I search for any device with the API call of `my.startBluetoothDevicesDiscovery`?

A: Please make sure that the device is discoverable. If the API is passed to Services, make sure that the discoverable content of the device contains the UUID of the service.

Q: If GPS positioning is not enabled when calling an bluetooth API, some devices will report an error message that bluetooth can't be connected as positioning service is not enabled.

A: The bluetooth function of the mini program relies on GPS positioning, as about 20% mobile phones rely on GPS to enable bluetooth. It is recommended to guide users to turn on the GPS positioning service in order to be connected with bluetooth.

Q: How do I resolve device connection failure?

A: Please make sure the correct `deviceId` is transmitted with strong signals. If the signal is weak, a device connection failure may occur.

Q: How to resolve write/read data failure?

A: Make sure to check the following settings:

- `DeviceId`, `serviceId`, and `characteristicId` are transmitted in the correct format.
- `DeviceId` is connected (You can call the API `my. OnBLEConnectionStateChanged` to listen to the connection state changes; Call the API `my. GetConnectedBluetoothDevices` to check for devices that are connected.
- Write a method in the connected state.
- Check and make sure `characteristicId` belongs to this Service.
- This characteristic supports write/read.

Q: How do I receive data notifications?

A:

- Make sure to call the API `my notifyBLECharacteristicValueChange` with correct parameters.
- Notify or indicate features are supported in the transmitted `characteristicID`.
- Make sure the hardware is notified.
- Pay attention to the basic flow sequence, i.e. call the API `notifyBLECharacteristicValueChange` once you're connected.

Q: Why are event callbacks called multiple times?

A: The same event was listened due to multiple anonymous function registrations. It is recommended to call off method to listen for an event and close the previous event listener, before you call on method.

Source:
https://miniprogram.gcash.com/docs/miniprogram_gcash/mpdev/api_device_bluetooth_bluetoothapifaq

---

Bluetooth API FAQ {#bluetooth-api-faq}

Last updated: 2021-05-10

Path: miniprogram_gcash

Bluetooth API FAQ

2021-05-10 03:43

Q: Does the API call of `my. WriteBLECharacteristicValue` return an empty value?

A: No. Calling this API returns the value of what you write.

Q: Why is listening not supported with the API call of `my.onBLECharacteristicValueChange`? Do I have to write before I can listen?

A: Yes. To call this API, you first need to write before you can listen. In order to prevent multiple callbacks of an event caused by multiple registered event listeners, it is recommended to call off method to listen to an event and close the previous event listener, before you call on method.

Q: Why do I get an error code `10014` with the API call of `my. Why writeBLECharacteristicValue`?

A: The error code `10014` means the data you send are either empty or incorrectly formatted. It is recommended to check for errors in the written data or HEX conversion.

Q: Can I use the hexadecimal array to write characteristic value with the API `my writeBLECharacteristicValue` ?

A: No. The characteristic values you write are hexadecimal strings, which are limited to 20 bytes.

Q: What is the `deviceId` format for Android and iOS devices?

A:

- An Android device gets the MAC address for bluetooth, such as: `11:22:33:44:55:66`.
- An iOS device gets the UUID of Bluetooth, such as: `00000000-0000-0000-0000-000000000000`.

Q: Why can't I search for any device with the API call of `my.startBluetoothDevicesDiscovery`?

A: Please make sure that the device is discoverable. If the API is passed to Services, make sure that the discoverable content of the device contains the UUID of the service.

Q: If GPS positioning is not enabled when calling an bluetooth API, some devices will report an error message that bluetooth can't be connected as positioning service is not enabled.

A: The bluetooth function of the mini program relies on GPS positioning, as about 20% mobile phones rely on GPS to enable bluetooth. It is recommended to guide users to turn on the GPS positioning service in order to be connected with bluetooth.

Q: How do I resolve device connection failure?

A: Please make sure the correct `deviceId` is transmitted with strong signals. If the signal is weak, a device connection failure may occur.

Q: How to resolve write/read data failure?

A: Make sure to check the following settings:

- `DeviceId`, `serviceId`, and `characteristicId` are transmitted in the correct format.
- `DeviceId` is connected (You can call the API `my. OnBLEConnectionStateChanged` to listen to the connection state changes; Call the API `my. GetConnectedBluetoothDevices` to check for devices that are connected.
- Write a method in the connected state.
- Check and make sure `characteristicId` belongs to this Service.
- This characteristic supports write/read.

Q: How do I receive data notifications?

A:

- Make sure to call the API `my notifyBLECharacteristicValueChange` with correct parameters.
- Notify or indicate features are supported in the transmitted `characteristicID`.
- Make sure the hardware is notified.
- Pay attention to the basic flow sequence, i.e. call the API `notifyBLECharacteristicValueChange` once you're connected.

Q: Why are event callbacks called multiple times?

A: The same event was listened due to multiple anonymous function registrations. It is recommended to call off method to listen for an event and close the previous event listener, before you call on method.

Source: https://miniprogram.gcash.com/docs/miniprogram_gcash/mpdev-old/api_device_bluetooth_bluetoothapifaq

---

Bluetooth API Overview {#bluetooth-api-overview}

Last updated: 2021-05-09

Path: miniprogram_gcash

Bluetooth API Overview

2021-05-09 18:43

Before you get started, you can see the system requirement as a prerequisite and then see the process flow on how the bluetooth APIs work. You can then find an overview of all the bluetooth APIs and a sample of code snippet. In addition, you can refer to the FAQ and error code table for more details.

System Requirement

| | | --- | --- | | **Bluetooth Type** | **System Requirement** | | Bluetooth Low Energy (BLE) | Android: 5.0 and upper versions
iOS: no requirements |

Process Flow

The following diagram illustrates how the bluetooth function is enabled with APIs. You can see the details on how each API works.

BLE

Traditional Bluetooth

API List

| | | | --- | --- | --- | | **Bluetooth Type** | **API** | **Description** | | BLE | my.connectBLEDevice | Connect to low energy bluetooth devices. | | BLE | my.disconnectBLEDevice | Disconnect to low energy bluetooth devices. | | BLE | my.getBLEDeviceCharacteristics | Get the characteristics of low energy bluetooth devices. | | BLE | my.getBLEDeviceServices | Get all the low energy bluetooth devices that are discovered, including the connected devices. | | BLE | my.notifyBLECharacteristicValueChange | Enable the function to notify changes to the characteristic value. | | BLE | my.offBLECharacteristicValueChange | Disable the function to notify changes to the characteristic value. | | BLE | my.offBLEConnectionStateChanged | Disable the the event listener for the connection status. | | BLE | my.onBLECharacteristicValueChange | Enable the event listener for changes to the characteristic value. | | BLE | my.onBLEConnectionStateChanged | Enable the the event listener for changes to the connection status , such as device lost and device disconnected. | | BLE | my.readBLECharacteristicValue | Read the characteristic value. | |

BLE | my.writeBLECharacteristicValue | Write data to the characteristic value. | | Bluetooth | my.openBluetoothAdapter | Use this API to initialize the Bluetooth module in the mini program. | | Bluetooth | my.closeBluetoothAdapter | Use this API to close the Bluetooth module in the mini program. | | Bluetooth | my.getBluetoothAdapterState | Use this API to check the Bluetooth adapter status in the mini program. | | Bluetooth | my.startBluetoothDevicesDiscovery | Use this API to start discovering bluetooth devices. | | Bluetooth | my.stopBluetoothDevicesDiscovery | Use this API to stop discovering bluetooth devices. | | Bluetooth | my.getBluetoothDevices | Use this API to get all the bluetooth devices that are discovered, including those that are connected to the current device. | | Bluetooth | my.getConnectedBluetoothDevices | Use this API to get the bluetooth devices that are connected. | | Bluetooth | my.onBluetoothDeviceFound | Use this API when a new Bluetooth device is found. | | Bluetooth | my.offBluetoothDeviceFound | Use this API to remove the bluetooth devices that are found. | | Bluetooth | my.onBluetoothAdapterStateChange | Use this API to monitor the bluetooth adapter state changes. | | Bluetooth | my.offBluetoothAdapterStateChange | Use this API to remove the bluetooth adapter with a state change. |

Source: https://miniprogram.gcash.com/docs/miniprogram_gcash/mpdev-old/api_device_bluetooth_bluetoothapioverview

---

Bluetooth API Overview {#bluetooth-api-overview}

Last updated: 2022-07-03

Path: miniprogram_gcash

Bluetooth API Overview

2022-07-03 18:44

Before you get started, you can see the system requirement as a prerequisite and then see the process flow on how the bluetooth APIs work. You can then find an overview of all the bluetooth APIs and a sample of code snippet. In addition, you can refer to the FAQ and error code table for more details.

System Requirement

| | | | --- | --- | | **Bluetooth Type** | **System Requirement** | | Bluetooth Low Energy (BLE) | Android: 5.0 and upper versions
iOS: no requirements |

Process Flow

The following diagram illustrates how the bluetooth function is enabled with APIs. You can see the details on how each API works.

BLE

Traditional Bluetooth

API List

| | | | --- | --- | --- | | **Bluetooth Type** | **API** | **Description** | | BLE | my.connectBLEDevice | Connect to low energy bluetooth devices. | | BLE | my.disconnectBLEDevice | Disconnect to low energy bluetooth devices. | | BLE | my.getBLEDeviceCharacteristics | Get the characteristics of low energy bluetooth devices. | | BLE | my.getBLEDeviceServices | Get all the low energy bluetooth devices that are discovered, including the connected devices. | | BLE | my.notifyBLECharacteristicValueChange | Enable the function to notify changes to the characteristic value. | | BLE | my.offBLECharacteristicValueChange | Disable the function to notify changes to the characteristic value. | | BLE | my.offBLEConnectionStateChanged | Disable the the event listener for the connection status. | | BLE | my.onBLECharacteristicValueChange | Enable the event listener for changes to the characteristic value. | | BLE | my.onBLEConnectionStateChanged | Enable the the event listener for changes to the connection status , such as device lost and device disconnected. | | BLE | my.readBLECharacteristicValue | Read the characteristic value. | | BLE | my.writeBLECharacteristicValue | Write data to the characteristic value. | | Bluetooth | my.openBluetoothAdapter | Use this API to initialize the Bluetooth module in the mini program. | | Bluetooth | my.closeBluetoothAdapter | Use this API to close the Bluetooth module in the mini program. | | Bluetooth | my.getBluetoothAdapterState | Use this API to check the Bluetooth adapter status in the mini program. | | Bluetooth | my.startBluetoothDevicesDiscovery | Use this API to start discovering bluetooth devices. | | Bluetooth | my.stopBluetoothDevicesDiscovery | Use this API to stop discovering bluetooth devices. | | Bluetooth | my.getBluetoothDevices | Use this API to get all the bluetooth devices that are discovered, including those that are connected to the current device. | | Bluetooth | my.getConnectedBluetoothDevices | Use this API to get the bluetooth devices that are connected. | | Bluetooth | my.onBluetoothDeviceFound | Use this API when a new Bluetooth device is found. | | Bluetooth | my.offBluetoothDeviceFound | Use this API to remove the bluetooth devices that are found. | | Bluetooth | my.onBluetoothAdapterStateChange | Use this API to monitor the bluetooth adapter state changes. | | Bluetooth | my.offBluetoothAdapterStateChange | Use this API to remove the bluetooth adapter with a state change. |

Source:
https://miniprogram.gcash.com/docs/miniprogram_gcash/mpdev/api_device_bluetooth_bluetoothapioverview

---

Button {#button}

Last updated: 2022-07-03

Path: miniprogram_gcash

Button

2022-07-03 18:44

You can use the button component to allow users to perform actions or make choices. The component is often used to submit forms, redirect users to another interface, or guide users to next steps. The usage is consistent with the basic button component, but more styles are added in the extended button component.

Sample code

See the sample codes in different languages:

.json

copy

```
{
 "defaultTitle": "Button",
 "usingComponents": {
   "button": "mini-ali-ui/es/button/index",
   "radio": "mini-ali-ui/es/am-radio/index",
   "checkbox": "mini-ali-ui/es/am-checkbox/index"
 }
}
```

.axml

copy

```
<view class="container">
 <button onTap="onTest" showLoading="{{showLoading}}" dataName="
{{dataName}}" type="{{type}}" subtitle="{{subtitle}}" disabled="
{{disabled}}" shape="{{shape}}" capsuleSize="{{capsuleSize}}"
capsuleMinWidth="{{capsuleMinWidth}}">
   {{title}}
 </button>
 <view>main title</view>
 <input value="{{title}}" onInput="titleChange"/>
 <view>subtitle</view>
 <input value="{{subtitle}}" onInput="subtitleChange"/>
 <view>button type</view>
 <radio-group class="radio-group" onChange="typeChange" name="type">
   <label class="radio" a:for="{{types}}" key="label-{{index}}">
     <radio value="{{item.name}}" checked="{{item.checked}}" />
     <text class="radio-text">{{item.value}}</text>
   </label>
 </radio-group>
 <view>shape</view>
 <radio-group class="radio-group" onChange="shapeChange" name="shape">
   <label class="radio" a:for="{{shapes}}" key="label-{{index}}">
     <radio value="{{item.name}}" checked="{{item.checked}}" />
```

```
    <text class="radio-text">{{item.value}}</text>
   </label>
 </radio-group>
 <view>capsule button size</view>
 <radio-group class="radio-group" onChange="sizeChange" name="size">
   <label class="radio" a:for="{{capsuleSizes}}" key="label-
{{index}}">
    <radio value="{{item.name}}" checked="{{item.checked}}" />
    <text class="radio-text">{{item.value}}</text>
   </label>
 </radio-group>
 <view>disable or not</view>
 <checkbox onChange='onDisableChange'/>
 <view>Enable minimum width of capsule button or not</view>
 <checkbox onChange='onMinWidthChange'/>
 <view>achieve loading or not</view>
 <checkbox onChange='onLoadingChange'/>
</view>
```

.acss

copy

```
.container {
  padding: 20rpx;
}

.container button {
  margin-bottom: 24rpx;
}
```

.js

copy

```
Page({
  data: {
    title: 'push-button control Normal',
    subtitle: '',
    disabled: false,
    dataName: '1',
    type: '',
    shape: 'default',
    capsuleSize: 'medium',
    capsuleMinWidth: false,
    showLoading: false,
    types: [\
      { name: 'default', value: 'default', checked: true },\
      { name: 'primary', value: 'primary' },\
      { name: 'ghost', value: 'ghost' },\
      { name: 'text', value: 'text' },\
```

```
    { name: 'warn', value: 'warn' },\
    { name: 'warn-ghost', value: 'warn-ghost' },\
    { name: 'light', value: 'light' },\
  ],
  shapes: [\
    { name: 'default', value: 'default', checked: true },\
    { name: 'capsule', value: 'capsule' },\
  ],
  capsuleSizes: [\
    { name: 'small', value: 'small' },\
    { name: 'medium', value: 'medium', checked: true },\
    { name: 'large', value: 'large' },\
  ],
},
onLoad() {
},
typeChange(e) {
  this.setData({
    type: e.detail.value,
  });
},
shapeChange(e) {
  this.setData({
    shape: e.detail.value,
  });
},
sizeChange(e) {
  this.setData({
    capsuleSize: e.detail.value,
  });
},
titleChange(e) {
  this.setData({
    title: e.detail.value,
  });
},
subtitleChange(e) {
  this.setData({
    subtitle: e.detail.value,
  });
},
onDisableChange(e) {
  this.setData({
    disabled: e.detail.value,
  });
},
onMinWidthChange(e) {
  this.setData({
    capsuleMinWidth: e.detail.value,
  });
},
```

```
  onTap() {
    // e.target.dataset.name
  },
  onLoadingChange(e) {
    this.setData({
      showLoading: e.detail.value,
    });
  },
});
```

Parameters

| | | | | --- | --- | --- | | **Property** | **Type** | **Description** | | type | String | Button style. Valid values are:
- `default`
- `primary`
- `ghost`
- `warn`
- `warn-ghost`
- `text`
- `light`
The default value is `default`. | | subtitle | String | Subtitle. | | shape | String | Button shape. Valid values are:
- `default`
- `capsule`
The default value is `default`. | | capsuleSize | String | Capsule button size. Valid values are:
- `large`
- `medium`
- `small`
The default value is `medium`. | | capsuleMinWidth | Boolean | An indicator of whether to use the minimum width for the capsule button. The default value is `false`. | | disabled | Boolean | An indicator of whether to disable the button. The default value is `false`. | | showLoading | Boolean | An indicator of whether to display the loading icon on the button. The default value is `false`. | | hover-class | String | Style class after the button is pressed. The default class name is `button-hover` and the style is `{background-color: rgba(0, 0, 0, 0.1); opacity: 0.7;}`. If the value is `none`, no style changes occur when users press the button. | | hover-start-time | Number | The duration in milliseconds between two moments. One moment is the time when users tap the button and the other is the time when the button changes to the tapped status. The default value is `20`. | | hover-stay-time | Number | The duration in milliseconds that the tapped status lasts after users tap the button. The default value is `70`. | | hover-stop-propagation | Boolean | An indicator of whether to prevent the tapped status from the parent element of the current element. The default value is `false`. | | form-type | String | The button type that is used in the form component. Valid values are:
- `submit`
- `reset`
When users tap the button, the `onSubmit` event or the `onReset` event is triggered. | | open-type | String | Open capability, such as the payment capability. | | scope | String | Authorization scope, which is valid when the value of *open-type* is `getAuthorize`. | |

onTap | EventHandle | The event that is triggered when users tap the button. | | app-parameter | String | Parameters that are transmitted to the opened app, which are valid when the value of *open-type* is `launchApp`. |

九色鹿

Source: https://miniprogram.gcash.com/docs/miniprogram_gcash/mpdev/extended-component_form_button

---

Calendar {#calendar}

Last updated: 2022-07-03

Path: miniprogram_gcash

Calendar

2022-07-03 18:44

Calendar

| | | | | | | --- | --- | --- | --- | --- | | **Property** | **Description** | **Type** | **Default** | **Required** | | type | Selection type `single`: Single date `range`: Date range. | String | single | No | | tagData | Tag data, including `date`, `tag`, `disable` or not, tag color `tagColor`. The tagColor includes 1.#f5a911, 2.#e8541e, 3.#07a89b 4.#108ee9 and 5.#b5b5b5. | `Array<date, tag, tagColor>` | | No | | onSelect | Select range callback. | ([startDate, endDate]) => void | | No | | onMonthChange | Callback on clicking month change, including two parameters currentMonth (change to next month) and prevMonth (change to previous month). | (currentMonth, prevMonth) => void | | No | | onSelectHasDisableDate | Selected range includes unusable date. | (currentMonth, prevMonth) => void | | No |

Example

copy

```
{
  "defaultTitle": "AntUI Component Library",
  "usingComponents":{
    "calendar": "mini-antui/es/calendar/index"
  }
}
```

copy

```
<view>
  <calendar
    type="single"
```

```
    tagData="{{tagData}}"
    onSelect="handleSelect" />
</view>
```

copy

```
Page({
  data: {
    tagData: [\
      { date: '2018-05-14', tag: 'Returning mortgage', tagColor: 5 },\
      { date: '2018-05-28', tag: 'Provident fund', tagColor: 2 },\
    ],
  },
  handleSelect() {},
  onMonthChange() {},
});
```

Source: https://miniprogram.gcash.com/docs/miniprogram_gcash/mpdev/extended-component_others_calendar

---

Call an API via adding a signature {#call-an-api-via-adding-a-signature}

Last updated: 2022-07-03

Path: miniprogram_gcash

Call an API via adding a signature

2022-07-03 18:44

Before calling an API, signing a request is needed. After sending the request and obtaining the response, you need to validate the response signature accordingly.

Sign a request

Procedures

1. Obtain your private key, represented by `privateKey`, which is used to sign a request.
2. Construct the content to be signed (`Content_To_Be_Signed`).
3. Calculate and generate the signature.
4. Add the generated signature to the request header.

For details of each step, see the following examples.

Example

1. Obtain your private key to sign the request

Get your private key ready, which is used to generate the signature later.

2. Construct the content to be signed

For example, a request has the following properties:

- `HTTP_URI` : for example, /api/v2/payments/pay
- `Client-Id` : TEST_5X00000000000000
- `Request-Time` : 2019-05-28T12:12:12+08:00
- `HTTP_BODY` : the body looks like the following format.

copy

```
{
 "order":{
    "orderId":"OrderID_0101010101",
    "orderDescription":"sample_order",
    "orderAmount":{
       "value":"100",
       "currency":"JPY"
    }
 },
 "paymentAmount":{
    "value":"100",
    "currency":"JPY"
 },
 "paymentFactor": {
     "isInStorePayment": "true"
 }
}
```

By complying with the Syntax of Content_To_Be_Signed, the content to be signed (`Content_To_Be_Signed`) is created as follows:

copy

```
POST /api/v2/payments/pay
TEST_5X00000000000000.2019-05-28T12:12:12+08:00.{
"order":{
    "orderId":"OrderID_0101010101",
    "orderDescription":"sample_order",
    "orderAmount":{
       "value":"100",
       "currency":"JPY"
    }
 },
 "paymentAmount":{
```

```
    "value":"100",
    "currency":"JPY"
 },
 "paymentFactor": {
     "isInStorePayment": "true"
 }
}
```

Syntax of `Content_To_Be_Signed`

copy

```
<HTTP_METHOD> <HTTP_URI>
<Client-Id>.<Request-Time>.<HTTP_BODY>
```

- `HTTP_METHOD` : POST
- `HTTP_URI` : For example, if the HTTP URL is **https://example.com/api/v2/payments/pay**, this property is `/api/v2/payments/pay`.
- `Client-Id` : is used to identify a client, and is associated with the keys that are used for signature and. You can get this field from the request header.
- `Request-Time`: Specifies the time when a request is sent, as defined by ISO8601. Note: This field must be accurate to milliseconds. For example, `2019-05-28T12:12:12+08:00` . You can get this field from the request header.
- `HTTP_BODY` : the data body of a request.

3. Calculate and generate the signature

Use the `sha256withrsa` method that involves the proper algorithm and private key to calculate and generate the signature.

copy

```
generatedSignature=base64UrlEncode(sha256withrsa(<Content_To_Be_Signed>
<privateKey>)
```

- `Content_To_Be_Signed:` the content to be signed that is obtained in step 2.
- `privateKey` : the private key value that is obtained in step 1.
- `sha256withrsa` : the algorithm to use, RSA256.

For example, the generated signature `generatedSignature` looks as follows:

copy

```
KrwDE9tAPJYBb4cUZU6ALJxGIZgwDXn5UkFPMip09n%2FkYKPhEIII%2Fki2rYY2lPtuKVg
2FjzRpohDbrOd8zYriiukpGAxBQDIVbatGI7WYOcc9YVQwdCR6ROuRQvr%2FD1AfdhHd6wa
w1OW7Ti93LTd0tcyEWQYd2S7c3A73sHOJNYl8DC1PjasiBozZ%2FADgb7ONsqHo%2B8fKHs
TGIRBQsvfgICnJhh%2BzXV8AQoecJBTrv6p%xxxx
```

4. Add the generated signature to the request header

a. Assemble a signature string as the following syntax.

copy

```
'Signature: algorithm=<algorithm>, keyVersion=<key-version>,
signature=<generatedSignature>'
```

- `algorithm` , `keyVersion` : see the header of the Message structure chapter.
- `generatedSignature` : the signature that is generated in step 3.

For example:

copy

```
'Signature: algorithm=RSA256, keyVersion=0,
signature=KrwDE9tAPJYBb4cUZU6ALJxGIZgwDXn5UkFPMip09n%2FkYKPhEIII%2Fki2
```

b. Add the signature string to the request header.

For example :

copy

```
-H 'Signature: algorithm=RSA256, keyVersion=0,
signature=KrwDE9tAPJYBb4cUZU6ALJxGIZgwDXn5UkFPMip09n%2FkYKPhEIII%2Fki2
```

Send a request

Construct a request by adding the `Client-Id`, `Request-Time`, and `Signature` fields to the request header. After a request is constructed, you can use common tools, like cURL or Postman to send the request. In the following example, cURL is used.

copy

```
curl -X POST \
  https://example.com/api/v2/payments/pay \
  -H 'Content-Type: application/json' \
  -H 'Client-Id: TEST_5X00000000000000' \
  -H 'Request-Time: 2019-05-28T12:12:12+08:00' \
  -H 'Signature: algorithm=RSA256, keyVersion=0,
signature=KrwDE9tAPJYBb4cUZU6ALJxGIZgwDXn5UkFPMip09n%2FkYKPhEIII%2Fki2
\
  -d '{
      "order":{
          "orderId":"OrderID_0101010101",
          "orderDescription":"sample_order",
          "orderAmount":{
             "value":"100",
             "currency":"JPY"
          }
       },
       "paymentAmount":{
          "value":"100",
```

```
          "currency":"JPY"
       },
       "paymentFactor": {
           "isInStorePayment": "true"
       }
}'
```

Handle a response

After you receive a response, you need to validate the signature of the response. A response consists of response headers and the response body. For example:

- The response header sample

copy

```
Client-Id: 5X00000000000000
Response-Time: 2019-05-28T12:12:14+08:00
Signature: algorithm=RSA256, keyVersion=0,
signature=p9T2hXxIjek0UOLw3fwlthNsV6ATaioIvu8X1uFx8a9tE87d2XEhqylnf0Kj
GwwlDS3tsSenwnL0Ha6BsXbJvUHRC5qcVlNy5Oq%2FpNqx2%2BKdwbw4eY7tZBDQhMKoaM
Trace-Id: 0ba604b4155861560080137195381 4.0
```

- The response body sample

copy

```
{
"result": {
    "resultCode":"SUCCESS",
    "resultStatus":"S",
    "resultMessage":"success"
},
"paymentTime": "2019-05-28T12:12:13+08:00",
"paymentId":"1234567"
}
```

Validate a signature

1. Obtain the platform public key.
2. Construct the content to be validated (`Content_To_Be_Validated`).
3. Get the signature from the response header.
4. Validate the signature.

For details of each step, see the following examples.

Example

1. Obtain the platform public key

The `Client-Id` and `KeyVersion` properties can be obtained from the response header. Merchants send these properties to the wallet, based on which, the wallet returns the public key to the merchant.

2. Construct the content to be validated

Given the response body sample above, by complying with the Syntax of Content_To_Be_Validated, construct the content to be validated (`Content_To_Be_Validated`) as follows:

copy

```
POST /api/v2/payments/pay
TEST_5X00000000000000.2019-05-28T12:12:14+08:00.{
 "result": {
    "resultCode":"SUCCESS",
    "resultStatus":"S",
    "resultMessage":"success"
 },
 "paymentTime": "2019-05-28T12:12:13+08:00",
 "paymentId":"1234567"
}
```

Syntax of `Content_To_Be_Validated`

copy

```
<HTTP_METHOD> <HTTP_URI>
<Client-Id>.<Response-Time>.<HTTP_BODY>
```

- `Client-Id` : identifies a client. You can get this field from the response header. For example
- `TEST_5X00000000000000`
- `Response-Time`: Indicates the time when a response is returned, as defined by ISO8601. Note: This field must be accurate to milliseconds. You can get this field from the response header.
- `HTTP_BODY` : Indicates the data body of the response.

3. Get the signature from the response header

The target signature string (`target_signature`) is extracted from `Signature` header of the response. For details about the response header, see the Message structure chapter.

copy

```
Signature: algorithm=RSA256, keyVersion=0, signature=
<target_signature>
```

4. Validate the signature

Use the `sha256withrsa_verify` method to validate the signature of a response.

Syntax of the `sha256withrsa_verify` method:

copy

```
sha256withrsa_verify(base64UrlDecode(<target_signature>),
<Content_To_Be_Validated>, <serverPublicKey>)
```

- `target_signature` : the signature extracted from the response header, which is obtained from step 3.
- `Content_To_Be_Validated`: the content to be validated that is created from step 2.
- `serverPublicKey` : the platform public key that is obtained from step 1.

More information

Message structure

Source: https://miniprogram.gcash.com/docs/miniprogram_gcash/mpdev/call_api

---

Call an API via adding a signature {#call-an-api-via-adding-a-signature}

Last updated: 2021-05-09

Path: miniprogram_gcash

Call an API via adding a signature

2021-05-09 18:43

Before calling an API, signing a request is needed. After sending the request and obtaining the response, need to validate the response signature accordingly.

Sign a request

Procedure

1. Obtain your private key, represented by `privateKey`, which is used to sign a request.
2. Construct the content to be signed (`Content_To_Be_Signed`).

3. Calculate and generate the signature.
4. Add the generated signature to the request header.

For details of each step, see the following examples.

Example

1. Obtain your private key to sign the request

Get your private key ready, which is used to generate the signature later.

2. Construct the content to be signed

For example, a request has the following properties:

- `HTTP_URI` : for example, /api/v2/payments/pay
- `Client-Id` : TEST_5X00000000000000
- `Request-Time` : 2019-05-28T12:12:12+08:00
- `HTTP_BODY` : the body looks like the following format.

copy

```
{
 "order":{
    "orderId":"OrderID_0101010101",
    "orderDescription":"sample_order",
    "orderAmount":{
       "value":"100",
       "currency":"JPY"
    }
 },
 "paymentAmount":{
    "value":"100",
    "currency":"JPY"
 },
 "paymentFactor": {
     "isInStorePayment": "true"
 }
}
```

By complying with the Syntax of Content_To_Be_Signed, the content to be signed (`Content_To_Be_Signed`) is created as follows:

copy

```
POST /api/v2/payments/pay
TEST_5X00000000000000.2019-05-28T12:12:12+08:00.{
"order":{
    "orderId":"OrderID_0101010101",
    "orderDescription":"sample_order",
    "orderAmount":{
```

```
      "value":"100",
      "currency":"JPY"
   }
},
"paymentAmount":{
   "value":"100",
   "currency":"JPY"
},
"paymentFactor": {
    "isInStorePayment": "true"
}
}
```

3. Calculate and generate the signature

Use the `sha256withrsa` method that involve the proper algorithm and private keyto calculate and generate the signature.

copy

```
generatedSignature=base64UrlEncode(sha256withrsa(<Content_To_Be_Signed>
<privateKey>))
```

- `Content_To_Be_Signed:` the content to be signed that is obtained in step 2.
- `privateKey` : the private key value that is obtained in step 1.
- `sha256withrsa` : the algorithm to use, RSA256.

For example, the generated signature `generatedSignature` looks as follows:

copy

```
KrwDE9tAPJYBb4cUZU6ALJxGIZgwDXn5UkFPMip09n%2FkYKPhEIII%2Fki2rYY2lPtuKVg
2FjzRpohDbrOd8zYriiukpGAxBQDIVbatGI7WYOcc9YVQwdCR6ROuRQvr%2FD1AfdhHd6wa
w1OW7Ti93LTd0tcyEWQYd2S7c3A73sHOJNYl8DC1PjasiBozZ%2FADgb7ONsqHo%2B8fKHs
TGIRBQsvfgICnJhh%2BzXV8AQoecJBTrv6p%xxxx
```

4. Add the generated signature to the request header

a. Assemble a signature string as the following syntax.

copy

```
'Signature: algorithm=<algorithm>, keyVersion=<key-version>,
signature=<generatedSignature>'
```

- `algorithm` , `keyVersion` : see the header of the Message structure chapter.
- `generatedSignature` : the signature that is generated in step 3.

For example:

copy

```
'Signature: algorithm=RSA256, keyVersion=0,
signature=KrwDE9tAPJYBb4cUZU6ALJxGIZgwDXn5UkFPMip09n%2FkYKPhEIII%2Fki2
```

b. Add the signature string to the request header.

For example:

copy

```
-H 'Signature: algorithm=RSA256, keyVersion=0,
signature=KrwDE9tAPJYBb4cUZU6ALJxGIZgwDXn5UkFPMip09n%2FkYKPhEIII%2Fki2
```

Syntax of `Content_To_Be_Signed`

copy

```
<HTTP_METHOD> <HTTP_URI>
<Client-Id>.<Request-Time>.<HTTP_BODY>
```

- `HTTP_METHOD` : POST
- `HTTP_URI` : For example, if the HTTP URL is https://example.com/api/v2/payments/pay, this property is `/api/v2/payments/pay`.
- `Client-Id` : is used to identify a client, and is associated with the keys that are used for signature and encryption. You can get this field from the request header.
- `Request-Time`: Specifies the time when a request is sent, as defined by RFC3339. Note: This field must be accurate to milliseconds. For example, `2019-05-28T12:12:12+08:00` . You can get this field from the request header.
- `HTTP_BODY` : the data body of a request.

Send a request

Construct a request by adding the `Client-Id`, `Request-Time`, and `Signature` fields to the request header. After a request is constructed, you can use common tools, like cURL or Postman to send the request. In the following example, cURL is used.

copy

```
curl -X POST \
  https://example.com/api/v2/payments/pay \
  -H 'Content-Type: application/json' \
  -H 'Client-Id: TEST_5X00000000000000' \
  -H 'Request-Time: 2019-05-28T12:12:12+08:00' \
  -H 'Signature: algorithm=RSA256, keyVersion=0,
signature=KrwDE9tAPJYBb4cUZU6ALJxGIZgwDXn5UkFPMip09n%2FkYKPhEIII%2Fki2
\
  -d '{
      "order":{
          "orderId":"OrderID_0101010101",
          "orderDescription":"sample_order",
```

```
            "orderAmount":{
               "value":"100",
               "currency":"JPY"
            }
         },
         "paymentAmount":{
            "value":"100",
            "currency":"JPY"
         },
         "paymentFactor": {
             "isInStorePayment": "true"
         }
}'
```

Handle a response

After you receive a response, you need to validate the signature of the response. A response consists of response headers and response body. For example:

- The response header sample

copy

```
Client-Id: 5X00000000000000
Response-Time: 2019-05-28T12:12:14+08:00
Signature: algorithm=RSA256, keyVersion=0,
signature=p9T2hXxIjek0UOLw3fwlthNsV6ATaioIvu8X1uFx8a9tE87d2XEhqylnf0Kj
GwwlDS3tsSenwnL0Ha6BsXbJvUHRC5qcVlNy5Oq%2FpNqx2%2BKdwbw4eY7tZBDQhMKoaM
Trace-Id: 0ba604b4155861560080137195381 4.0
```

- The response body sample

copy

```
{
"result": {
    "resultCode":"SUCCESS",
    "resultStatus":"S",
    "resultMessage":"success"
},
"paymentTime": "2019-05-28T12:12:13+08:00",
"paymentId":"1234567"
}
```

Validate a signature

1. Obtain the platform public key.
2. Construct the content to be validated (`Content_To_Be_Validated`).
3. Get the signature from the response header.

4. Validate the signature.

For details of each step, see the following examples.

Example

1. Obtain the platform public key

The `Client-Id` and `KeyVersion` properties can be obtained from the response header. Merchants send these properties to the wallet, based on which, the wallet returns the public key to the merchant.

2. Construct the content to be validated

Given the response body sample above, by complying with the Syntax of Content_To_Be_Validated, construct the content to be validated (`Content_To_Be_Validated`) as follows:

copy

```
POST /api/v2/payments/pay
TEST_5X00000000000000.2019-05-28T12:12:14+08:00.{
 "result": {
    "resultCode":"SUCCESS",
    "resultStatus":"S",
    "resultMessage":"success"
 },
 "paymentTime": "2019-05-28T12:12:13+08:00",
 "paymentId":"1234567"
}
```

3. Get the signature from the response header

The target signature string (`target_signature`) is extracted from `Signature` header of the response. For details about the response header, see the Message structure chapter.

copy

```
Signature: algorithm=RSA256, keyVersion=0, signature=
<target_signature>
```

4. Validate the signature

Use the `sha256withrsa_verify`method to validate the signature of a response.

Syntax of the `sha256withrsa_verify` method:

copy

```
sha256withrsa_verify(base64UrlDecode(<target_signature>),
<Content_To_Be_Validated>, <serverPublicKey>)
```

- `target_signature` : the signature extracted from the response header, which is obtained from step 3.
- `Content_To_Be_Validated`: the content to be validated that is created from step 2.
- `serverPublicKey` : the platform public key that is obtained from step 1.

Syntax of `Content_To_Be_Validated`

copy

```
<HTTP_METHOD> <HTTP_URI>
<Client-Id>.<Response-Time>.<HTTP_BODY>
```

- `Client-Id` : identifies a client. You can get this field from the response header. For example `TEST_5X00000000000000`
- `Response-Time`: Indicates the time when a response is returned, as defined by RFC3339. Note: This field must be accurate to milliseconds. You can get this field from the response header.
- `HTTP_BODY` : Indicates the data body of the response.

Source: https://miniprogram.gcash.com/docs/miniprogram_gcash/mpdev-old/call_api

---

Capabilities {#capabilities}

Last updated: 2022-07-06

Path: miniprogram_gcash

Capabilities

2022-07-06 08:31

Capabilities are sets of JavaScript APIs (JSAPIs) and OpenAPIs that can work together to help merchants and ISV implement specific functions. For example, the payment capability is accomplished by the `my.tradePay` JSAPI and OpenAPIs such as `/{version}/payments/pay` and `/{version}/payments/notifyPayment`, where the `version` is `v1` or `v2`.

Architecture overview

The calling process of JSAPIs and OpenAPIs is illustrated as below:

JSAPI

JSAPIs are available by default in the mini program container. You can call the existing JSAPIs from the mini program to use. When you call a JSAPI, it will interact between the wallet and the merchant or ISV backend server.

OpenAPI

ACL (Access Control List) OpenAPI standards are available. You can also use the standards of OpenAPIs and define the OpenAPIs. When you call an OpenAPI, it will interact between the wallet and the merchant or ISV backend server.

With a combination of JSAPIs and OpenAPIs, a set of capabilities can be defined. For example, you can see the following capabilities that are available by default:

- User information capability
- Payment capability
- For all OpenAPIs that support various capabilities, see OpenAPIs for Merchants.

In addition, you can define your own features that are centrally managed in the workspace of the Mini Program Platform. You can select the JSAPIs that are available by default. For more information, see Features.

More information

JSAPIs

Open APIs

Using Mini Program Platform

Features

Source: https://miniprogram.gcash.com/docs/miniprogram_gcash/mpdev/capabilities

---

Capabilities {#capabilities}

Last updated: 2021-05-10

Path: miniprogram_gcash

Capabilities

2021-05-10 03:43

Introduction

Capabilities are sets of Java Script APIs (JSAPIs) and Open APIs that can work together to help users to complete specific tasks. For example, the payment capability is accomplished by the `my.tradePay` JSAPI and open APIs such as `/{version}/payments/pay` and `/{version}/payments/notifyPayment`, where the `version` is like `v1` or `v2`

Architecture Overview

JSAPIs are available by default in the mini program container. You can use the existing JSAPIs which can be called from mini program, through wallet native apps to the merchant/ISV back-end server.

ACL Open API standards are available. You can also use the standards of Open APIs and define the Open APIs, which can be interacted in the back-end servers between the wallet and the merchant/ISV.

With a combination of JSAPIs and Open APIs, a set of capabilities can be defined. For example, you can see the following capabilities that are available by default:

- User Information Capability
- Payment Capability
- For all Open APIs that support various capabilites, see Open APIs for Merchants

In addition, you can define your own features that are centrally managed in the workspace of the Mini Program Platform. You can select the JSAPIs that are available by default. For more information, see Features.

More Information

JSAPIs

Open APIs

Using Mini Program Platform

Features

Source: https://miniprogram.gcash.com/docs/miniprogram_gcash/mpdev-old/capabilities

---

Card {#card}

Last updated: 2022-07-03

Path: miniprogram_gcash

Card

2022-07-03 18:44

Card.

| | | | | --- | --- | --- | --- | | **Property** | **Description** | **Type** | **Required** | | thumb | Card thumbnail address. | String | No | | title | Card title. | String | Yes | | subTitle | Card subtitle. | String | No | | footer | Footer text. | String | No | | footerImg | Footer image address. | String | No | | onCardClick | Callback when the card is clicked. | (info: Object) => void | No | | info | Used to transfer data to the outside when the card is clicked. | String | No |

Example

copy

```
{
  "defaultTitle": "AntUI Component Library",
  "usingComponents": {
    "card": "mini-antui/es/card/index"
  }
}
```

copy

```
<card
  thumb="{{thumb}}"
  title="Card Title"
  subTitle="Subtitle is not required"
  onClick="onCardClick"
  footer="Description"
  footerImg="{{footerImg}}"
  info="Click the card"
/>
```

copy

```
Page({
    data: {
        thumb: 'https://img.example.com/example.png',
        footerImg: 'https://img.example.com/example.png',
    },
    onCardClick: function(ev) {
        my.showToast({
            content: ev.info,
        });
    }
});
```

Source: https://miniprogram.gcash.com/docs/miniprogram_gcash/mpdev/extended-component_layout-navigation_card

Coding in the Editor {#coding-in-the-editor}

Last updated: 2022-07-03

Path: miniprogram_gcash

Coding in the Editor

2022-07-03 18:44

Defines the features when you code in the editor.

Feature

In addition to the basic editing functions, the Mini Program Studio provides you with the following functions that are specific to Mini Program:

- Realtime preview: You can preview the your codes in the local simulator.
- Autocomplete: Your codes are automatically completedly.
- Syntax prompt/suggestions: You can also get suggestions on the syntax.

For more informaion, see the demos below:

Realtime Preview

Autocomplete

AXML Autocomplete

API Autocomplete

Source: https://miniprogram.gcash.com/docs/miniprogram_gcash/mpdev/mini-program-studio_code

---

Collapse {#collapse}

Last updated: 2022-07-03

Path: miniprogram_gcash

Collapse

2022-07-03 18:44

Collapse/expand a content region.

• Group or hide complex region to keep the page clean and tidy.

• **Accordion mode** is a special collapse which allows opening just one content region.

Sample Code

copy

```
{
  "defaultTitle": "Mini Program AntUI component library",
  "usingComponents": {
    "collapse": "mini-antui/es/collapse/index",
    "collapse-item": "mini-antui/es/collapse/collapse-item/index"
  }
}
```

copy

```
<view>
  <view class="demo-title">Basic usage</view>
  <collapse
    className="demo-collapse"
    collapseKey="collapse1"
    activeKey="{{['item-11', 'item-13']}}"
    onChange="onChange"
  >
    <collapse-item header="Title 1" itemKey="item-11"
collapseKey="collapse1">
      <view class="item-content content1">
        <view>Content region</view>
      </view>
    </collapse-item>
    <collapse-item header="Title 2" itemKey="item-12"
collapseKey="collapse1">
      <view class="item-content content2">
        <view>Content region</view>
      </view>
    </collapse-item>
    <collapse-item header="Title 3" itemKey="item-13"
collapseKey="collapse1">
      <view class="item-content content3">
        <view>Content region</view>
      </view>
    </collapse-item>
```

```
      </collapse>
      <view class="demo-title">Accordion mode</view>
      <collapse
        className="demo-collapse"
        collapseKey="collapse2"
        activeKey="{{['item-21', 'item-23']}}"
        onChange="onChange"
        accordion="{{true}}"
      >
        <collapse-item header="Title 1" itemKey="item-21"
collapseKey="collapse2">
          <view class="item-content content1">
            <view>Content region</view>
          </view>
        </collapse-item>
        <collapse-item header="Title 2" itemKey="item-22"
collapseKey="collapse2">
          <view class="item-content content2">
            <view>Content region</view>
          </view>
        </collapse-item>
        <collapse-item header="Title 3" itemKey="item-23"
collapseKey="collapse2">
          <view class="item-content content3">
            <view>Content region</view>
          </view>
        </collapse-item>
      </collapse>
    </view>
```

copy

```
    .item-content {
      padding: 14px 16px;
      font-size: 17px;
      color: #333;
      line-height: 24px;
    }

    .content1 {
      height: 200px;
    }

    .content2 {
      height: 50px;
    }

    .content3 {
      height: 100px;
    }

    .demo-title {
```

```
    padding: 14px 16px;
    color: #999;
  }

  .demo-collapse {
    border-bottom: 1px solid #eee;
  }
```

copy

```
  Page({});
```

Attributes

| | | | | --- | --- | --- | --- | | **Property** | **Description** | **Type** | **Default** | | activeKey | Key of the active tab panel. | Array / String | None by default, or the first element by default in the accordion mode | | onChange | Callback for switching panel. | (activeKeys: Array): void | - | | accordion | Accordion mode. | Boolean | false | | collapseKey | Uniquely identifying the collapse and corresponding collapse-item. | String | false |

Collapse-item

| | | | | --- | --- | --- | --- | | **Property** | **Description** | **Type** | **Default** | | itemKey | Corresponding activeKey. | String | Unique component identifier | | header | Header content of the panel. | String | - | | collapseKey | Uniquely identifying the collapse and corresponding collapse-item. | String | false |

When a page has multiple collapse components, the collapseKey attribute of the collapse and the corresponding collapse-item must be mandatory and equal. When a page has just one collapse component, the collapseKey is not mandatory.

Source: https://miniprogram.gcash.com/docs/miniprogram_gcash/mpdev/extended-component_layout-navigation_collapse

---

Command-line interface {#command-line-interface}

Last updated: 2022-07-03

Path: miniprogram_gcash

Command-line interface

2022-07-03 18:44

The Command-line Interface (CLI) of the Alipay Plus Mini Program (APMP)is a tool that helps you to initialize, develop, and maintain your mini programs. It helps you to achieve cross-platform operation, preview on physical devices, initialize projects, and upload projects. Compared with IDE, CLI is more efficiency as developers can use other code editing tools to improve mini program's development.

Overview

You can take an overview of CLI in the following table:

| | | | --- | --- | --- | | **Category** | **Description** | **Command** | | Description | Check versions | apmp -v | | Check command-line descriptions | apmp -h | | Workspace | Log in | apmp login | | log out | apmp logout | | APP | Select the current app | apmp app select | | Initialize projects | apmp app create | | Preview apps | apmp app preview | | Upload apps | apmp app upload |

Features

You can benefit from the following features:

- apmp login
- Check project information
- Set workspaces
- Select the current project
- Persistent login

Global installation

Sample code

copy

```
npm install apmp -g
```

Log in

Sample code

copy

```
# Input the username and password via command parameters
apmp login -u [username] -p [password] -w [workspaceId] -a [appId]

# Input the username and password via inquirer Q&A
apmp login
> username: [username]
```

```
> password: [password]
>
```

Parameters

| | | | | --- | --- | --- | | **Name** | **Value** | **Required** | | -u,--username | username | Yes | | -p,--password | password | Yes | | -w,--workspaceId | workspaceId | No | | -a,--appId | appId | No |

Log out

Log out, clear cache, and clear cookies.

Sample code

copy

```
apmp logout
```

Select the current app

Sample code

copy

```
apmp app select // use arrow key to move cursor
> app1:[app1]
> app2:[app2]
> app3:[app3]
```

Initialize projects

Sample code

copy

```
# Input app type and name via command parameters
apmp app create –t [mini–app] –n [appName]

# Input the app type and name via inquirer Q&A
apmp app create
– project type
> mini–app
– app name : [app name]
```

Parameters

| | | | | | | --- | --- | --- | --- | --- | | **Name** | **Value** | **Required** | **Default Value** | **Remark** | | -t, --type | type | No | None | Enum: `mini-app` | | -n, --name | name | No | new-project | A new file will be generated according to the new project name. |

Preview apps

Sample code

copy

```
# Use the currently selected project
apmp app preview
```

Parameters

| | | | | | | --- | --- | --- | --- | --- | | **Name** | **Value** | **Required** | **Default Value** | **Remark** | | -a, --appId | appId | No | selectedCurrentAppId | You can obtain an appId in two ways:
- Select an appId in the app list.
- Obtain the currently selected appId If you have not selected an appId in the app list. | | -w, --workspaceId | workspaceId | No | selectedCurrentWorkspaceId | You can obtain a workspaceId in two ways:
- Select a workspaceId in the workspace list.
- Obtain the currently selected workspaceId if you have not selected a workspaceId in the workspace list. | | -p, -projectPath | projectPath | No | . | Use the current project path by default. |

Upload apps

Sample code

copy

```
# Use the currently selected project
apmp app upload
```

Parameters

| | | | | | | --- | --- | --- | --- | --- | | **Name** | **Value** | **Required** | **Default Value** | **Remark** | | -a, --appId | appId | No | selectedCurrentAppId | You can obtain an appId in two ways:
- Select an appId in the app list.
- Obtain the currently selected appId If you have not selected an appId in the app list. | | -w, --workspaceId | workspaceId | No | selectedCurrentWorkspaceId | You can obtain a workspaceId in two ways:
- Select a workspaceId in the workspace list.

- Obtain the currently selected workspaceId If you have not selected a workspaceId in the workspace list. | | -p, --projectPath | projectPath | No | . | Use the current project path by default. | | --nextVersion | nextVersion | No | | specify upload version number. |

Source: https://miniprogram.gcash.com/docs/miniprogram_gcash/mpdev/command-line

---

Compatibility {#compatibility}

Last updated: 2021-05-09

Path: miniprogram_gcash

Compatibility

2021-05-09 18:43

The component and API of Mini Program are improving and enriching, so some old sdk may not support the newly added abilities, so developers should handle the compatibility issue.

The `my.canIUse(String)` can help to detect whether the API, input parameter or returned value, component, property is supported in current version.

Handle Newly Added API

Following codes show how to handle compatibility issue for newly added API.

copy

```
if (my.getLocation) {
    my.getLocation();
} else {
    my.alert({
       title: 'Hint',
       content: 'The function is not supported, please upgrade App'
    });
}
```

Handle Newly Added Input Parameter of API

copy

```
if (my.canIUse('getLocation.object.type')) {
    // ...
} else {
```

```
    console.log('The parameter is not supported')
}
```

Handle Newly Added Return Value of API

copy

```
if (my.canIUse('getSystemInfo.return.storage')) {
    // ...
} else {
    console.log('The returned value is not supported')
}
```

Handle Newly Added Property of Component

copy

```
Page({
  data: {
    canIUse: my.canIUse('button.open-type.share')
  }
})
```

copy

```
<button a:if="{{canIUse}}" open-type="share">Share</button>
<button a:else onTap="shareApp">Share Mini Program</button>
```

Source: https://miniprogram.gcash.com/docs/miniprogram_gcash/mpdev-old/framework_compatibility

---

Compatibility {#compatibility}

Last updated: 2022-07-03

Path: miniprogram_gcash

Compatibility

2022-07-03 18:44

The component and API of Mini Program are improving and enriching, so some old sdk may not support the newly added abilities, so developers should handle the compatibility issue.

The `my.canIUse(String)` can help to detect whether the API, input parameter or returned value, component, property is supported in current version.

Handle Newly Added API

Following codes show how to handle compatibility issue for newly added API.

copy

```
if (my.getLocation) {
    my.getLocation();
} else {
    my.alert({
       title: 'Hint',
       content: 'The function is not supported, please upgrade App'
    });
}
```

Handle Newly Added Input Parameter of API

copy

```
if (my.canIUse('getLocation.object.type')) {
    // ...
} else {
    console.log('The parameter is not supported')
}
```

Handle Newly Added Return Value of API

copy

```
if (my.canIUse('getSystemInfo.return.storage')) {
    // ...
} else {
    console.log('The returned value is not supported')
}
```

Handle Newly Added Property of Component

copy

```
Page({
  data: {
    canIUse: my.canIUse('button.open-type.share')
  }
})
```

copy

```
<button a:if="{{canIUse}}" open-type="share">Share</button>
<button a:else onTap="shareApp">Share Mini Program</button>
```

Source:
https://miniprogram.gcash.com/docs/miniprogram_gcash/mpdev/framework_compatibility

---

Component Reference {#component-reference}

Last updated: 2021-05-09

Path: miniprogram_gcash

Component Reference

2021-05-09 18:43

View Container

| | | --- | --- | | **Component Name** | **Function Description** | | view | View container. | | swiper | Swiper view container. | | scroll-view | Scroll view region. | | movable-view | Movable view. | | movalbe-area | Area for movable view. |

Basic Content

| | | --- | --- | | **Component Name** | **Function Description** | | text | Text. | | icon | Icon. | | progress | Progress bar. |

Form Component

| | | --- | --- | | **Component Name** | **Function Description** | | button | Button. | | form | Form. | | label | Label. | | input | Input box. | | textarea | Multi-row input box. | | radio | Radio selector. | | checkbox | Multiple selector. | | switch | Switch. | | slider | Sliding selector. | | picker-view | Scroll selector embedded in the page. | | picker | Scroll selector popping up from the bottom. |

Navigator

| | | --- | --- | | **Component Name** | **Function Description** | | navigator | Page link. |

Multimedia

| | | --- | --- | | **Component Name** | **Function Description** | | image | Image. |

Canvas

| | | --- | --- | | **Component Name** | **Function Description** | | canvas | Canvas. |

Open

| | | --- | --- | | **Component Name** | **Function Description** | | web-view | The component that runs the web page. |

Experience Mini Program

Developers can use the Android Demo App to scan the QR code shown in the component documents.

Source: https://miniprogram.gcash.com/docs/miniprogram_gcash/mpdev-old/component_component-reference

---

Component Reference {#component-reference}

Last updated: 2022-07-03

Path: miniprogram_gcash

Component Reference

2022-07-03 18:44

View Container

| | | --- | --- | | **Component Name** | **Function Description** | | view | View container. | | swiper | Swiper view container. | | scroll-view | Scroll view region. | | movable-view | Movable view. | | movalbe-area | Area for movable view. |

Basic Content

| | | --- | --- | | **Component Name** | **Function Description** | | text | Text. | | icon | Icon. | | progress | Progress bar. |

Form Component

| | | --- | --- | | **Component Name** | **Function Description** | | button | Button. | | form | Form. | | label | Label. | | input | Input box. | | textarea | Multi-row input box. | | radio | Radio selector. | | checkbox | Multiple selector. | | switch | Switch. | | slider | Sliding selector. | | picker-view | Scroll selector embedded in the page. | | picker | Scroll selector popping up from the bottom. |

Navigator

| | | --- | --- | | **Component Name** | **Function Description** | | navigator | Page link. |

Multimedia

| | | --- | --- | | **Component Name** | **Function Description** | | image | Image. |

Canvas

| | | --- | --- | | **Component Name** | **Function Description** | | canvas | Canvas. |

Open

| | | --- | --- | | **Component Name** | **Function Description** | | web-view | The component that runs the web page. |

Experience Mini Program

Developers can use the Android Demo App to scan the QR code shown in the component documents.

Source:
https://miniprogram.gcash.com/docs/miniprogram_gcash/mpdev/component_component-reference

---

Condition Rendering {#condition-rendering}

Last updated: 2022-07-03

Path: miniprogram_gcash

Condition Rendering

2022-07-03 18:44

a:if

In an `axml`, `a:if="{{condition}}"` is used to tell whether a code block is to be rendered.

copy

```
<view a:if="{{condition}}"> True </view>
```

Besides, `a:elif` and `a:else` can be used to add an `else` branch.

copy

```
<view a:if="{{length > 5}}"> 1 </view>
<view a:elif="{{length > 2}}"> 2 </view>
<view a:else> 3 </view>
```

block a:if

Because `a:if` is a control attribute, and can only be used in one component. For conditional rendering of multiple components at once, an `<block/>` component can be used to contain those components, and add one `a:if` to control.

copy

```
<block a:if="{{true}}">
  <view> view1 </view>
  <view> view2 </view>
</block>
```

Note: `<Block/>` is not a component but just a packaging element. It does not render anything in the page but accepts control attribute only.

Compare a:if with hidden

- The template in `a:if` may contain data binding. So, when the `a:if` condition value changes, the framework has a process of local rendering which is used to ensure destroy or re-render in case of the change. In addition, when the initial render condition is false, the `a:if` does not trigger any render action, and starts local rendering when the condition turns true for the first time.
- The `hidden` controls show/hide, but the component is always rendered.

Generally, the `a:if` has a higher overhead when frequently toggled, while the `hidden` has a higher initial rendering overhead. As a result, `hidden` is better for frequent toggles. If the running conditions do not toggle much, `a:if` is preferred.

Source:
https://miniprogram.gcash.com/docs/miniprogram_gcash/mpdev/framework_axml-reference_condition-rendering

---

Condition Rendering {#condition-rendering}

Last updated: 2021-05-09

Path: miniprogram_gcash

Condition Rendering

2021-05-09 18:43

a:if

In an `axml`, `a:if="{{condition}}"` is used to tell whether a code block is to be rendered.

copy

```
<view a:if="{{condition}}"> True </view>
```

Besides, `a:elif` and `a:else` can be used to add an `else` branch.

copy

```
<view a:if="{{length > 5}}"> 1 </view>
<view a:elif="{{length > 2}}"> 2 </view>
<view a:else> 3 </view>
```

block a:if

Because `a:if` is a control attribute, and can only be used in one component. For conditional rendering of multiple components at once, an `<block/>` component can be used to contain those components, and add one `a:if` to control.

copy

```
<block a:if="{{true}}">
  <view> view1 </view>
  <view> view2 </view>
</block>
```

Note: `<Block/>` is not a component but just a packaging element. It does not render anything in the page but accepts control attribute only.

Compare a:if with hidden

- The template in `a:if` may contain data binding. So, when the `a:if` condition value changes, the framework has a process of local rendering which is used to ensure destroy or re-render in case of the change. In addition, when the initial render condition is false, the `a:if` does not trigger any render action, and starts local rendering when the condition turns true for the first time.
- The `hidden` controls show/hide, but the component is always rendered.

Generally, the `a:if` has a higher overhead when frequently toggled, while the `hidden` has a higher initial rendering overhead. As a result, `hidden` is better for frequent toggles. If the running conditions do not toggle much, `a:if` is preferred.

Source: https://miniprogram.gcash.com/docs/miniprogram_gcash/mpdev-old/framework_axml-reference_condition-rendering

---

Constructor {#constructor}

Last updated: 2021-05-10

Path: miniprogram_gcash

Constructor

2021-05-10 03:43

Constructor

Parameter Description

| | | | | --- | --- | --- | --- | | **Parameter** | **Type** | **Required** | **Description** | | data | Object | No | Component internal status. | | props | Object | No | Set default for incoming data. | | onInit | Function | No | Component lifecycle function, trigger on component creation. | | deriveDataFromProps | Function | No | Component lifecycle function, trigger on component creation and update. | | didMount | Function | No | Component lifecycle function, trigger on component creation completion. | | didUpdate | Function | No | Component lifecycle function, trigger on component update completion. | | didUnmount | Function | No | Component lifecycle function, trigger on component deletion. | | mixins | Array | No | Code reuse mechanism between components. | | methods | Object | No | Component method, can be event response function or any customized method. |

Sample Code

js sample code:

copy

```
Component({
  mixins:[{ didMount() {}, }],
  data: {y:2},
  props:{x:1},
  didUpdate(prevProps,prevData){},
  didUnmount(){},
  methods:{
    onMyClick(ev){
      my.alert({});
      this.props.onXX({ ...ev, e2:1});
    },
  },
})
```

Component Instance Attribute List

Parameter Description

| | | | --- | --- | --- | | **Property** | **Type** | **Description** | | data | Object | Component internal data. | | props | Object | Incoming component attribute. | | is | String | Component path. | | $page | Object | Component page instance. | | $id | Number | Component id, can render value in component axml. |

Sample Code

js sample code:

copy

```
// /components/xx/index.js
Component({
  didMount(){
    this.$page.xxCom = this; // this operation can load the component
instance to the belonging page instance
    console.log(this.is);
    console.log(this.$page);
    console.log(this.$id);
  }
});
```

axml sample code:

copy

```
<!-- /components/xx/index.axml component id can directly render value
in component axml -->
<view>{{$id}}</view>
```

json sample code:

copy

```
// /pages/index/index.json
{
  "usingComponents": {
    "xx": "/components/xx/index"
  }
}
```

js sample code:

copy

```
Page({
  onReady() {
    console.log(this.xxCom); // can access all loaded components
loaded onto the current page
  },
})
```

When the component is rendered on the page, execute the didMount callback, and the console has the following output:

copy

```
/components/xx/index
{$viewId: 51, route: "pages/index/index"}
1
```

Component Instance Method List

| | | | --- | --- | --- | | **Method name** | **Parameter** | **Descrsiption** | | setData | Object | Setting data triggers view rendering. | | $spliceData | Object | Setting data triggers view rendering. |

Source: https://miniprogram.gcash.com/docs/miniprogram_gcash/mpdev-old/framework_custom-component_create-custom-component_constructor

---

Constructor {#constructor}

Last updated: 2022-07-03

Path: miniprogram_gcash

Constructor

2022-07-03 18:44

Constructor

Parameter Description

| | | | | --- | --- | --- | --- | | **Parameter** | **Type** | **Required** | **Description** | | data | Object | No | Component internal status. | | props | Object | No | Set default for incoming data. | | onInit | Function | No | Component lifecycle function, trigger on component creation. | | deriveDataFromProps | Function | No | Component lifecycle function, trigger on component creation and update. | | didMount | Function | No | Component lifecycle function, trigger on component creation completion. | | didUpdate | Function | No | Component lifecycle function, trigger on component update completion. | | didUnmount | Function | No | Component lifecycle function, trigger on component deletion. | | mixins | Array | No | Code reuse mechanism between components. | | methods | Object | No | Component method, can be event response function or any customized method. |

Sample Code

js sample code:

copy

```
Component({
  mixins:[{ didMount() {}, }],
  data: {y:2},
  props:{x:1},
  didUpdate(prevProps,prevData){},
  didUnmount(){},
  methods:{
    onMyClick(ev){
      my.alert({});
      this.props.onXX({ ...ev, e2:1});
    },
  },
})
```

Component Instance Attribute List

Parameter Description

| | | | | --- | --- | --- | | **Property** | **Type** | **Description** | | data | Object | Component internal data. | | props | Object | Incoming component attribute. | | is | String | Component path. | | $page | Object | Component page instance. | | $id | Number | Component id, can render value in component axml. |

Sample Code

js sample code:

copy

```
// /components/xx/index.js
Component({
  didMount(){
    this.$page.xxCom = this; // this operation can load the component
instance to the belonging page instance
    console.log(this.is);
    console.log(this.$page);
    console.log(this.$id);
  }
});
```

axml sample code:

copy

```
<!-- /components/xx/index.axml component id can directly render value
in component axml -->
<view>{{$id}}</view>
```

json sample code:

copy

```
// /pages/index/index.json
{
  "usingComponents": {
    "xx": "/components/xx/index"
  }
}
```

js sample code:

copy

```
Page({
  onReady() {
    console.log(this.xxCom); // can access all loaded components
loaded onto the current page
  },
})
```

When the component is rendered on the page, execute the didMount callback, and the console has the following output:

copy

```
/components/xx/index
{$viewId: 51, route: "pages/index/index"}
1
```

Component Instance Method List

| | | | | --- | --- | --- | | **Method name** | **Parameter** | **Descrsiption** | | setData | Object | Setting data triggers view rendering. | | $spliceData | Object | Setting data triggers view rendering. |

Source:
https://miniprogram.gcash.com/docs/miniprogram_gcash/mpdev/framework_custom-component_create-custom-component_constructor

---

Container {#container}

Last updated: 2022-07-03

Path: miniprogram_gcash

Container

2022-07-03 18:44

You can use the container component to make the style of all elements in the container more consistent, such as margins between elements.

Sample code

See the sample codes in different languages:

.json

copy

```
{
 "defaultTitle": "Container",
 "usingComponents": {
   "container": "mini-ali-ui/es/container/index",
   "title": "mini-ali-ui/es/title/index"
 }
}
```

.axml

copy

```
<view className="container">
  <container className="container-item"
```

```
    title="Customized title"
    thumb="https://img.alicdn.com/tfs/TB1Go8lh9R26e4jSZFEXXbwuXXa-84-
84.png"
    onActionTap="titleClick"
  >
    <view slot="operation" style="color: red;">is slot</view>
    <view class="item">The container component's own title properties.
If the icon property is not specified, you can insert a slot named
operation.</view>
  </container>

  <container className="container-item"
    title="Title with an arrow"
    icon="arrow"
    thumb="https://img.alicdn.com/tfs/TB1Q19sTNv1gK0jSZFFXXb0sXXa-112-
112.png"
    onActionTap="titleClick"
  >
    <view class="item">The container component's own title
properties</view>
  </container>

  <container className="container-item"
    title="Title with a close icon"
    icon="close"
    thumb="https://img.alicdn.com/tfs/TB1Go8lh9R26e4jSZFEXXbwuXXa-84-
84.png"
    onActionTap="titleClick"
  >
    <view class="item">The container component's own title
properties</view>
  </container>

  <container className="container-item"
    title="Title with a more icon"
    icon="more"
    thumb="https://img.alicdn.com/tfs/TB1Q19sTNv1gK0jSZFFXXb0sXXa-112-
112.png"
    onActionTap="titleClick"
  >
    <view class="item">The container component's own title
properties</view>
  </container>

  <container className="container-item"
    title="Title without icon"
    thumb="https://img.alicdn.com/tfs/TB1Go8lh9R26e4jSZFEXXbwuXXa-84-
84.png"
    onActionTap="titleClick"
  >
    <view class="item">The container component's own title
```

```
properties</view>
  </container>
</view>

<view className="container">
  <container className="container-item">
    <view class="item">a1</view>
  </container>
  <container className="container-item">
    <view class="item">b1</view>
    <view class="item">b2</view>
  </container>
  <container className="container-item">
    <title slot="header" hasLine="true" showIcon="true"
iconURL="https://example.com/mdn/miniProgram_mendian/afts/img/A*wiFYTo
without onActionType</title>
    <view class="item">c1</view>
    <view class="item">c2</view>
    <view class="item">c3</view>
    <view slot="footer" class="footer" style="padding-left:
12px;">Bottom display area</view>
  </container>
  <container className="container-item">
    <title slot="header">slide</title>
    <swiper indicator-dots="{{true}}" class="item">
      <block a:for="{{['#0abc80','#00b7f4']}}">
        <swiper-item>
          <view style="background-color:
{{item}};width:100%;height:300rpx;border-radius:16rpx;"/>
        </swiper-item>
      </block>
    </swiper>
  </container>
  <container className="container-item" type="onewithtwo">
    <view class="grid-item" style ="height: 300rpx;"
slot="first">first</view>
    <view class="grid-item" slot="second">second</view>
    <view class="grid-item" slot="third">third</view>
  </container>
</view>
```

.js

copy

```
Page({
  data: {},
  onLoad() {},
  titleClick() {
    my.alert({
      title: 'onActionTap callback',
```

```
      content: 'Click the operation area after the title',
    });
  },
});
```

.acss

copy

```
.container {
  background: #F5F5F5;
  padding: 24rpx;
  height: 100%;
}

.container-item {
  margin-bottom: 24rpx;
}

.footer {
  color: #333;
  margin-top: 24rpx;
}

.item {
  background: #eeeeee;
  text-align: center;
  height: 200rpx;
  padding-top: 20rpx;
}

.grid-item {
  background: #eeeeee;
  text-align: center;
}
```

Parameters

| | | | | --- | --- | --- | | **Property** | **Type** | **Description** | | type | String | Layout type of container. Valid values are:
- `line`
- `onewithtwo`
The default value is `line`. When the value is `line`, all elements are equally divided into multiple rows or columns. | | className | String | Customized style name. | | title | String | Title name. The title component can be used if this property is specified. | | thumb | String | URL of the icon in title. | | icon | String | Icon on the right side of title. Valid values are:
- `arrow`
- `close`
- `more` | | onActionTap | EventHandle | The event that is triggered when users tap the icon on the right of title. The default value is `() => {}`. |

Source: https://miniprogram.gcash.com/docs/miniprogram_gcash/mpdev/extended-component_layout_container

---

Copy of PickerItem {#copy-of-pickeritem}

Last updated: 2022-07-03

Path: miniprogram_gcash

Copy of PickerItem

2022-07-03 18:44

Selection input.

Sample Code

copy

```
// API-DEMO page/component/input-item/input-item.json
{
  "defaultTitle": "Mini Program AntUI component library",
  "usingComponents": {
    "list": "mini-antui/es/list/index",
    "list-item": "mini-antui/es/list/list-item/index",
    "input-item": "mini-antui/es/input-item/index",
    "picker-item": "mini-antui/es/picker-item/index"
  }
}
```

copy

```
<!-- API-DEMO page/component/input-item/input-item.axml -->
<view>
  <view style="margin-top: 10px;" />
  <list>
    <input-item
      data-field="cardNo"
      clear="{{true}}"
      value="{{cardNo}}"
      className="dadada"
      placeholder="Bank card number"
      focus="{{inputFocus}}"
      onInput="onItemInput"
      onFocus="onItemFocus"
      onBlur="onItemBlur"
```

```
      onConfirm="onItemConfirm"
      onClear="onClear"
    >
      Card number
      <view slot="extra" class="extra" onTap="onExtraTap"></view>
    </input-item>
    <picker-item
      data-field="bank"
      placeholder="Select issuing bank"
      value="{{bank}}"
      onPickerTap="onPickerTap"
    >
      Issuing bank
    </picker-item>
    <input-item
      data-field="name"
      placeholder="Name"
      type="text"
      value="{{name}}"
      clear="{{true}}"
      onInput="onItemInput"
      onClear="onClear"
    >
      Name
    </input-item>
    <input-item
      data-field="password"
      placeholder="Password"
      password
    >
      Password
    </input-item>
    <input-item
      data-field="remark"
      placeholder="Remarks"
      last="{{true}}"
    />
  </list>
  <view style="margin: 10px;">
    <button type="primary" onTap="onAutoFocus">Focus</button>
  </view>
</view>
```

copy

```
// API-DEMO page/component/input-item/input-item.js
const banks = ['Mybank', 'CCB', 'ICBC', 'SPDB']

Page({
  data: {
    cardNo: '1234****',
    inputFocus: true,
```

```
    bank: '',
    name: '',
  },
  onAutoFocus() {
    this.setData({
      inputFocus: true,
    });
  },
  onExtraTap() {
    my.alert({
      content: 'extra tapped',
    });
  },
  onItemInput(e) {
    this.setData({
      [e.target.dataset.field]: e.detail.value,
    });
  },
  onItemFocus() {
    this.setData({
      inputFocus: false,
    });
  },
  onItemBlur() {},
  onItemConfirm() {},
  onClear(e) {
    this.setData({
      [e.target.dataset.field]: '',
    });
  },
  onPickerTap() {
    my.showActionSheet({
      title: 'Select issuing bank',
      items: banks,
      success: (res) => {
        this.setData({
          bank: banks[res.index],
        });
      },
    });
  },
});
```

copy

```
/* API-DEMO page/component/input-item/input-item.acss */
.extra {
  background-image: url('https://img.example.com/example.svg');
  background-size: contain;
  background-repeat: no-repeat;
  background-position: right center;
  opacity: 0.2;
```

```
    height: 20px;
    width: 20px;
    padding-left: 10px;
  }
```

Attributes

| | | | | | --- | --- | --- | --- | | **Property** | **Description** | **Type** | **Default** | | className | Customized class. | String | - | | labelCls | Customized label class. | String | - | | pickerCls | Customized selection region class. | String | - | | last | Is the last row or not. | Boolean | false | | value | Initial contents. | String | - | | name | Component name, used for getting data via form submission. | String | - | | placeholder | Placeholder. | String | - | | onPickerTap | Trigger on clicking pickeritem. | (e: Object) => void | - |

Slots

| | | | | --- | --- | --- | | **slotname** | **Description** | **Required** | | extra | Used to render the description right to picker-item. | No |

Source: https://miniprogram.gcash.com/docs/miniprogram_gcash/mpdev/rxgbgz

---

Coupon {#coupon}

Last updated: 2022-07-03

Path: miniprogram_gcash

Coupon

2022-07-03 18:44

You can use the coupon component to display the coupon, red packet, and ticket that can be redeemed by users.

Sample code

See the sample codes in different languages:

.json

copy

```
{
```

```
  "defaultTitle": "Coupon",
  "usingComponents":{
    "coupon":"mini-ali-ui/es/coupon/index",
    "button": "mini-ali-ui/es/button/index",
    "am-checkbox": "mini-ali-ui/es/am-checkbox/index",
    "stepper": "mini-ali-ui/es/stepper/index"
  }
}
```

.axml

copy

```
<view style="margin-top: 10px;"></view>
<view>
  <coupon title="coupon title1"
    onCouponClick="onCouponClick"
    thumb="{{thumb}}">
  </coupon>
</view>
<view>
  <coupon title="coupon title2"
    subtitle="coupon subtitle"
    onCouponClick="onCouponClick"
    thumb="{{thumb}}">
  </coupon>
</view>
<view>
  <coupon title="coupon title3"
```

```
    subtitle="coupon subtitle"

    used="{{true}}"

    onCouponClick="onCouponClick"

    thumb="{{thumb}}">

    <view slot="date">Valid period  2020.02.14-2020.02.29</view>

    <view slot="detail" class="coupon_rule">

      <text>1. Rule details  </text>

      <text>2. Rule details  </text>

    </view>

  </coupon>

</view>

<view>

  <coupon title="coupon title4"

    subtitle="coupon subtitle"

    onCouponClick="onCouponClick"

    thumb="{{thumb}}">

    <view slot="category" class="categoryDemo">

      <text class="price">50</text><text class="unit">CNY Yuan</text>
<text class="type">money off coupon</text>

    </view>

    <button shape="capsule" slot="action" onTap="onButtonTap"
type="ghost">Use immediately</button>

    <view slot="date">Valid period  2020.02.14-2020.02.29</view>

    <view slot="detail" class="coupon_rule">

      <text>1. Rule details  </text>

      <text>2. Rule details  </text>
```

```
    </view>

  </coupon>

</view>

<view>

  <coupon title="coupon title5"

    subtitle="coupon subtitle"

    onCouponClick="onCouponClick"

    extra="{{false}}"

    thumb="{{thumb}}">

    <button shape="capsule" slot="action" onTap="onButtonTap"
type="ghost">Use immediately</button>

    <view slot="date">Valid period  2020.02.14-2020.02.29</view>

    <view slot="detail" class="coupon_rule">

      <text>1. Rule details  </text>

      <text>2. Rule details  </text>

    </view>

  </coupon>

</view>

<view>

  <coupon title="coupon title6"

    subtitle="coupon subtitle"

    onCouponClick="onCouponClick"

    thumb="{{thumb}}"

  >

    <button shape="capsule" slot="action" onTap="onButtonTap"
type="ghost">Use immediately</button>

  </coupon>
```

```
</view>

<view>

  <coupon title="coupon title7"

    subtitle="coupon subtitle"

    moreBtn="see more"

    moreHide="{{false}}"

    onCouponClick="onCouponClick"

    thumb="{{thumb}}">

    <button shape="capsule" slot="action" onTap="onButtonTap"
type="primary">Use immediately</button>

    <view slot="date">vadility  2020.02.14-2020.02.29</view>

    <view slot="detail" class="coupon_rule">

      <text>1. Rule details  </text>

      <text>2. Rule details  </text>

    </view>

  </coupon>

</view>

<view>

  <coupon title="coupon title8"

    subtitle="coupon subtitle"

    onCouponClick="onCouponClick"

    thumb="{{thumb}}">

    <am-checkbox slot="action" onTap="onButtonTap" />

  </coupon>

</view>

<view>
```

```
  <coupon title="coupon title9"

    subtitle="coupon subtitle"

    onCouponClick="onCouponClick"

    thumb="{{thumb}}">

    <stepper

      slot="action"

      step="{{1}}"

      showNumber

      min="{{2}}"

    />

  </coupon>

</view>

<view style="margin-top: 50px;"></view>
```

.js

copy

```
Page({
  data: {
    thumb: 'https://example.com/mdn/rms_ce4c6f/afts/img/A*b-
kqQ4RZgsYAAAAAAAAAAABkARQnAQ',
  },
  onCouponClick(e) {
    if (e.currentTarget.dataset.used) {
      return false;
    } else {
      my.alert({
        content: 'available coupons, The coupon clicks the event',
      });
    }
  },
  onButtonTap() {
    my.alert({
      content: 'The capsule button clicks the event',
    });
  },
});
```

.acss

copy

```
.container {

  padding-bottom: 50px;

}

.coupon_rule text {

  display: block;

  margin-bottom: 8rpx;

}

/* the style of rights and interests content on the left
slot="category" */

.categoryDemo {

  display: flex;

  flex-wrap: wrap;

  justify-content: center;

  align-content: center;

  align-items: baseline;

  align-self: flex-start;

}

.categoryDemo .price {

  font-size: 60rpx;

  color: #FF6010;

}

.categoryDemo .unit {

  padding-left: 4rpx;

  font-weight: bold;
```

```
  font-size: 26rpx;

  color: #FF6010;

}

.categoryDemo .type {

  flex: 1 1 100%;

  text-align: center;

  font-size: 22rpx;

  color: #999;

}
```

Parameters

| | | | | | --- | --- | --- | --- | | **Property** | **Type** | **Required** | **Description** | | thumb | String | No | URL of the coupon thumbnail image. | | title | String | Yes | Coupon title. | | subTitle | String | No | Coupon subtitle. | | onCouponClick | Function | No | The event that is triggered when uses tap the coupon. | | extra | Boolean | No | An indicator of whether to display the coupon extended information on the left. The default value is `true`. | | moreBtn | String | No | The clickable text. After clicking the text, users can view rules on how use the coupon in detail. The default value is `More`. | | moreHide | Boolean | No | An indicator of whether to display the rules on how use the coupon in detail. The default value is `true`. | | used | Boolean | No | An indicator of whether the coupon is valid. The default value is `false`. |

slots

| | | | --- | --- | | **Slot name** | **Description** | | action | The slot on the right side of the coupon. After clicking the slot, users can use the coupon. | | date | The slot for the expiry time of the coupon. | | detail | The slot that is used to display rules on how to use the coupon in detail. | | category | The slot on the left of the coupon, which is used to display the coupon type. |

Source: https://miniprogram.gcash.com/docs/miniprogram_gcash/mpdev/extended-component_layout_coupon

---

Create mini programs {#create-mini-programs}

Last updated: 2022-07-07

Path: miniprogram_gcash

Create mini programs

2022-07-07 17:08

This topic describes the steps of the task to create a mini program.

Procedures

To create a mini program, you can follow the corresponding steps as below:

Step 1: Log in to the console

Open the Mini Program Platform portal and click **CONSOLE** or **SIGN IN**.

Enter the email and password that have been registered. Click **Sign In** to enter Mini Program Console.

Step 2: Choose a workspace

Choose the workspace that you want to create a mini program.

Step 3: Enter information

Click **Mini Program** on the navigation panel to the left. Then, click **+ Create Mini Program** to create a new mini program.

> **Note**: Only workspace admin can see the mini program source such as wallet, merchant, and Alipay+.

Choose the mini program type based on your needs. The platform supports two types of mini programs:

- `DSL(Default)`: Native mini programs
- `HTML5-based`: HTML 5 mini programs

For more information about how to create HTML 5 mini programs, see How to transform an HTML 5 mobile app to an HTML 5 mini program.

After that, enter the mini program information according to requirements and prompt on the right panel and click **Create**.

Now you have completed creating a mini program.

Next steps

Add mini program members

Upload mini programs

Configure mini programs

Release mini programs

Generate QR codes

Remove mini programs

Source: https://miniprogram.gcash.com/docs/miniprogram_gcash/platform/create

---

Data Binding {#data-binding}

Last updated: 2022-07-04

Path: miniprogram_gcash

Data Binding

2022-07-04 03:44

The dynamic data in AXML is bound with the `data` content in `Page`.

Simple Binding

The Mustache syntax is used to package variable with two pairs of braces (`{{}}`). It can be used in various syntax scenarios.

Contents

copy

```
<view> {{ message }} </view>
```

copy

```
Page({
  data: {
    message: 'Hello!',
  },
});
```

Component Attribute

Component attributes need to be packaged with double quotation marks ("").

copy

```
<view id="item-{{id}}"> </view>
```

copy

```
Page({
  data: {
    id: 0,
  },
});
```

Control Attribute

Control attributes need to be packaged with double quotation marks ("").

copy

```
<view a:if="{{condition}}"> </view>
```

copy

```
Page({
  data: {
    condition: true,
  },
});
```

Keywords

The keywords need to be packaged with double quotation marks ("").

- True: boolean true
- False: boolean false

copy

```
<checkbox checked="{{false}}"> </checkbox>
```

Note: Do not code directly `checked="false"`. The operation result is a string, and becomes the true value when converted into boolean type.

Operation

Simple operation can be packaged with two pairs of braces (`{{}}`). The following operations are supported:

Ternary Operation

copy

```
<view hidden="{{flag ? true : false}}"> Hidden </view>
```

Arithmetic Operation

copy

```
<view> {{a + b}} + {{c}} + d </view>
Page({
  data: {
    a: 1,
    b: 2,
    c: 3,
  },
});
```

Page output content is `3 + 3 + d`

Logic Judgment

copy

```
<view a:if="{{length > 5}}"> </view>
```

String Operation

copy

```
<view>{{"hello" + name}}</view>
```

copy

```
Page({
  data:{
    name: 'Mini Program',
  },
});
```

Data Path Operation

copy

```
<view>{{object.key}} {{array[0]}}</view>
```

copy

```
Page({
  data: {
    object: {
     key: 'Hello ',
},
    array: ['Mini Program'],
  },
});
```

Combine

The combination can be done directly in the Mustache syntax to make up a new object or array.

Array

copy

```
<view a:for="{{[zero, 1, 2, 3, 4]}}"> {{item}} </view>
Page({
  data: {
    zero: 0,
  },
});
```

Finally combined into array `[0, 1, 2, 3, 4]`

Object

copy

```
<template is="objectCombine" data="{{foo: a, bar: b}}"></template>
Page({
  data: {
    a: 1,
    b: 2,
  },
});
```

Finally combined into object `{foo: 1, bar: 2}`.

Destructuring operator `...` can be used to expand an object:

copy

```
<template is="objectCombine" data="{{...obj1, ...obj2, e: 5}}">
</template>
```

copy

```
Page({
  data: {
    obj1: {
      a: 1,
      b: 2,
    },
    obj2: {
      c: 3,
      d: 4,
    },
```

```
  },
});
```

Finally combined into object `{a: 1, b: 2, c: 3, d: 4, e: 5}`.

If the object key and value are the same, the indirect expression is as below:

copy

```
<template is="objectCombine" data="{{foo, bar}}"></template>
Page({
  data: {
    foo: 'my-foo',
    bar: 'my-bar',
  },
});
```

Finally combined into object `{foo: 'my-foo', bar:'my-bar'}`

Note: The above methods can be combined randomly. When the variable names are the same, however, the latter overrides the former. For example:

copy

```
Page({
    data: {
        obj1: {
      a: 1,
      b: 2,
    },
    obj2: {
      b: 3,
      c: 4,
    },
  a: 5,
  },
});
```

Finally combined into object `{a: 5, b: 3, c: 6}`.

FAQ

Q: How to clear the data when jumping to a new page?

A: The data can not be cleared, you can override the data when jumping.

Source:
https://miniprogram.gcash.com/docs/miniprogram_gcash/mpdev/framework_axml-reference_data-binding

---

Data Binding {#data-binding}

Last updated: 2021-05-09

Path: miniprogram_gcash

Data Binding

2021-05-09 18:43

The dynamic data in AXML is bound with the `data` content in `Page`.

Simple Binding

The Mustache syntax is used to package variable with two pairs of braces (`{{}}`). It can be used in various syntax scenarios.

Contents

copy

```
<view> {{ message }} </view>
```

copy

```
Page({
  data: {
    message: 'Hello!',
  },
});
```

Component Attribute

Component attributes need to be packaged with double quotation marks ("").

copy

```
<view id="item-{{id}}"> </view>
```

copy

```
Page({
  data: {
    id: 0,
  },
});
```

Control Attribute

Control attributes need to be packaged with double quotation marks ("").

copy

```
<view a:if="{{condition}}"> </view>
```

copy

```
Page({
  data: {
    condition: true,
  },
});
```

Keywords

The keywords need to be packaged with double quotation marks ("").

- True: boolean true
- False: boolean false

copy

```
<checkbox checked="{{false}}"> </checkbox>
```

Note: Do not code directly `checked="false"`. The operation result is a string, and becomes the true value when converted into boolean type.

Operation

Simple operation can be packaged with two pairs of braces (`{{}}`). The following operations are supported:

Ternary Operation

copy

```
<view hidden="{{flag ? true : false}}"> Hidden </view>
```

Arithmetic Operation

copy

```
<view> {{a + b}} + {{c}} + d </view>
Page({
  data: {
    a: 1,
    b: 2,
```

```
    c: 3,
  },
});
```

Page output content is `3 + 3 + d`

Logic Judgment

copy

```
<view a:if="{{length > 5}}"> </view>
```

String Operation

copy

```
<view>{{"hello" + name}}</view>
```

copy

```
Page({
  data:{
    name: 'Mini Program',
  },
});
```

Data Path Operation

copy

```
<view>{{object.key}} {{array[0]}}</view>
```

copy

```
Page({
  data: {
    object: {
     key: 'Hello ',
},
    array: ['Mini Program'],
  },
});
```

Combine

The combination can be done directly in the Mustache syntax to make up a new object or array.

Array

copy

```
<view a:for="{{[zero, 1, 2, 3, 4]}}"> {{item}} </view>
Page({
  data: {
    zero: 0,
  },
});
```

Finally combined into array `[0, 1, 2, 3, 4]`

Object

copy

```
<template is="objectCombine" data="{{foo: a, bar: b}}"></template>
Page({
  data: {
    a: 1,
    b: 2,
  },
});
```

Finally combined into object `{foo: 1, bar: 2}`.

Destructuring operator `...` can be used to expand an object:

copy

```
<template is="objectCombine" data="{{...obj1, ...obj2, e: 5}}">
</template>
```

copy

```
Page({
  data: {
    obj1: {
      a: 1,
      b: 2,
    },
    obj2: {
      c: 3,
      d: 4,
    },
  },
});
```

Finally combined into object `{a: 1, b: 2, c: 3, d: 4, e: 5}`.

If the object key and value are the same, the indirect expression is as below:

copy

```
<template is="objectCombine" data="{{foo, bar}}"></template>
Page({
  data: {
    foo: 'my-foo',
    bar: 'my-bar',
  },
});
```

Finally combined into object `{foo: 'my-foo', bar:'my-bar'}`

Note: The above methods can be combined randomly. When the variable names are the same, however, the latter overrides the former. For example:

copy

```
Page({
    data: {
        obj1: {
      a: 1,
      b: 2,
    },
    obj2: {
      b: 3,
      c: 4,
    },
  a: 5,
  },
});
```

Finally combined into object `{a: 5, b: 3, c: 6}`.

FAQ

Q: How to clear the data when jumping to a new page?

A: The data can not be cleared, you can override the data when jumping.

Source: https://miniprogram.gcash.com/docs/miniprogram_gcash/mpdev-old/framework_axml-reference_data-binding

---

Data Reporting by Self-Defined Actions {#data-reporting-by-self-defined-actions}

Last updated: 2022-07-07

Path: miniprogram_gcash

Data Reporting by Self-Defined Actions

2022-07-07 17:08

This topic introduces data reporting by self-defined actions. With this method, you can define the actions with different parameters to collect and report data automatically.

Procedures

To report the data by self-defined actions, complete the following steps:

1. When defining an event, choose the *By Self-Defined Actions* `Data Reporting Method`.
2. Configure the action with the following parameters:

- Define the trigger condition.
- Define the report type. Two report types are supported:
- Collect and Report Once
- Collect Multiple Times and Report Once
- Define the triggered page.
- Define the triggered element.
- Save and publish the actions.

Note:

If you choose the `Collect Multiple Times and Report Once` report type, you must specify that the action is used to collect or report data. After you select the `Collect Multiple Times and Report Once` report type, you can see the following fields:

- `Start`: Start to collect data.
- `Report`: Report the collected data.

The last one action must be used to report data.

Trigger conditions

The following trigger conditions are supported:

- enterPage: Triggered when users open a new page, return to the previous page, or switch to the foreground. Page must be specified.
- click: Triggered by a click event. Page and element must be specified.
- leavePage: Triggered when users exit a page or switching to the background. Page must be specified.

- pageLoad: Triggered when users open a new page for the first time. Page must be specified.
- pageUnload: Triggered when users reclaim a page. Page must be specified.
- switchTab: Triggered when calling the *my.switchTab* API to switch to another page. Page must be specified.
- pullDownRefresh: Triggered by pull-to-refresh event. Page must be specified.
- launch: Triggered when users load the mini program.
- background: Triggered when users switch to the background.
- foreground: Triggered when users switch to the foreground.
- share: Triggered when users share the page via the menu in the upper-right corner.

More information

Manage Events

Source:
https://miniprogram.gcash.com/docs/miniprogram_gcash/platform/fillinconfiguration

---

Data dictionary {#data-dictionary}

Last updated: 2022-07-03

Path: miniprogram_gcash

Data dictionary

2022-07-03 18:44

Model

Result

| | | | | --- | --- | --- | --- | | **Property** | **Data type** | **Required** | **Description** | | resultStatus | String | No | Result status. Valid values are:
- `S` : Successful.
- `F` : Failed.
- `U` : Unknown.
- `A` : accepted, not yet succeed, but can proceed with some actions. | | resultCode | String | No | Result code.
Max. length: 64 characters. | | resultMessage | String | No | Result message that describes the result code in details.
Max. length: 256 characters. |

Message

| | | | | --- | --- | --- | --- | | **Property** | **Data type** | **Required** | **Description** | | messageId | String | Yes | The unique ID that identifies a message.
Max. length: 64 characters. | | customerId | String | Yes | The unique ID that is assigned by wallets to identify a customer.
Example: 218820000000000000023.
Max. length: 32 characters. | | messageChannel | **MessageChannel** | Yes | The channel to send a message. | | redirectUrl | String | No | The Mini Program page that is redirected to after the user clicks the PUSH/INBOXmessage. Therefore, for the PUSH/INBOX message, this property is required.
Max. length: 1024 characters. | | messageContent | JSON String | Yes | The content of a message to send, for example:
```
"{
"en-US": "Hello, World",
"zh-CN":  "        "
}"
```
Max. length: 4096 characters. |

MessageSendResult

| | | | | --- | --- | --- | --- | | **Property** | **Data type** | **Required** | **Description** | | messageId | String | Yes | The unique ID that identifies a message.
Max. length: 64 characters. | | success | Boolean | Yes | Indicates whether the message is sent successfully or not.
- `true` : the message is sent successfully.
- `false` : the message fails to send. | | retry | Boolean | No | Indicates whether the wallet server allows you to retry when the message fails to send. Therefore, this field is required only when the value of the `success` field is `false`.
The value of the `retry` field is:
- `true` : the wallet server allows to retry under specific circumstances, for example, when there are system exceptions.
- `false` : the wallet server does not allow to retry. For example, when the fatigue degree in the wallet server is reached, retry is not allowed. |

ContentTemplate

| | | | | --- | --- | --- | --- | | **Property** | **Data type** | **Required** | **Description** | | templateParameters | Map<String,String> | No | A string-to-string mapping. The data structure is in JSON format:
`"templateParameters": {"key": "value"}`
, where:
- `key` : represents the placeholder that is designed in the selected message template.
- `value` : is used to replace the value of the `key` parameter in the message template. | | language | String | No | RFC 1766, such as zh-CN, en-US. |

Enum

MessageChannel

| | | --- | --- | | **Key** | **Description** | | SMS | SMS. | | PUSH | PUSH. | | INBOX | INBOX. |

OsType

| | | | --- | --- | | **Key** | **Description** | | IOS | iOS. | | ANDROID | Android. |

Source:
https://miniprogram.gcash.com/docs/miniprogram_gcash/mpdev/datadictionary_wallets_v2

Data dictionary (for v1) {#data-dictionary-for-v1-}

Last updated: 2022-07-03

Path: miniprogram_gcash

Data dictionary (for v1)

2022-07-03 18:44

Model

Amount

| | | | | | --- | --- | --- | --- | | **Property** | **Data type** | **Required** | **Description** | | currency | String | Yes | The three-character ISO-4217 currency code.
Max. length: 3 characters. | | value | String | Yes | A string that encloses a positive integer representing how much to charge in the smallest currency unit(e.g., 100 cents to charge $1.00 or 100 to charge $100, a zero-decimal currency).
Max. length: 16 characters. |

ActionForm

| | | | | | --- | --- | --- | --- | | **Property** | **Data type** | **Required** | **Description** | | actionFormType | String | No | The action form type. Enum:
ORDER_CODE,
REDIRECTION. | | orderCode | String | No | The order code value.
Max. length: 2048 characters. | | redirectionUrl | String | No | The url of redirect.
Max. length: 4096 characters. |

EnvInfo

| | | | | | --- | --- | --- | --- | | **Property** | **Data type** | **Required** | **Description** | | terminalType | String | No | The terminal type of this request.
Enum: [MINI_APP, APP, WEB, WAP, SYSTEM]. | | osType | String | No | OS type.
Enum: [IOS, ANDROID] | | userAgent | String | No | User agent.

Max. length: 1024 characters. | | deviceTokenId | String | No | Token ID of the device.
Max. length: 128 characters. | | clientIp | String | No | IP address of the client device.
Max. length: 64 characters. | | cookieId | String | No | User cookie ID.
Max. length: 128 characters. | | extendInfo | String | No | Extend info.
Max. length: 4096 characters. |

PaymentFactor

| | | | | | --- | --- | --- | --- | | **Property** | **Data type** | **Required** | **Description** | | isPaymentEvaluation | Boolean | No | An indicator of the payment evaluation. If the value is TRUE, the payment is only to evaluate whether the payment can be successful and no actual funds deduction occurs. The default value is FALSE. | | isOrderCode | Boolean | No | If the payment senario is the user scan the code presented by merchant and identify the order, and make payment. The default value is FALSE. | | isPaymentCode | Boolean | No | An indicator of whether the payment scenario is the merchant scan the user payment code.The default value is FALSE. | | isAgreementPay | Boolean | No | An indicator of whether the payment is an agreement payment.The default value is FALSE. | | isCashierPayment | Boolean | No | An indicator of whether the payment is a cashier payment.The default value is FALSE. | | isAuthorizationAndPay | Boolean | No | An indicator of whether to do agreementPay authorization during the payment.The default value is FALSE. | | isAuthorizationPayment | Boolean | No | An indicator of whether the payment is an authorization payment.The default value is FALSE. |

OpenLoginIdInfo

| | | | | | --- | --- | --- | --- | | **Property** | **Data type** | **Required** | **Description** | | loginId | String | No | Login Id is a identification for an user, which can be mobile number or email. User can use login Id to login Wallet.
Max. length: 64 characters. | | loginIdType | String | Yes | Login id type, Enum: "MOBILE_PHONE", "EMAIL".
Max. length: 64 characters. | | maskLoginId | String | No | Mask Login id, several bits of the phone number will be hidden to protect users' privacy.
Max. length: 64 characters. | | hashLoginId | String | No | Hash login id. The login id hashed by hash algorithm. The external system can use it to compare it's login id hashed by the same hash algorithm to see if the login id is the same.
Max. length: 256 characters. | | extendInfo | String | No | The extend information,wallet and merchant can put extend info here.
Max. length: 4096 characters. |

ContactInfo

| | | | | | --- | --- | --- | --- | | **Property** | **Data type** | **Required** | **Description** | | contactNo | String | Yes | Contact No. e.g mobile-phone No, e-mail address.
Max. length: 64 characters. | | contactType | String | Yes | Contact type, there are agreed types(MOBILE_PHONE, TELEPHONE, EMAIL) that should be available to all users, developer can add new type, but developer should consider the compatibility.
Max. length: 32 characters. | | extendInfo | String | No | The extend information,wallet and merchant can put extend info here.
Max. length: 4096 characters. |

OpenUserInfo

| | | | | | --- | --- | --- | --- | | **Property** | **Data type** | **Required** | **Description** | | userId | String | No | The hashed unique identifier allocated for user.
Max. length: 64 characters. | | status | String | No | User status, Enum: "ACTIVE", "FROZEN", "INACTIVE".
Max. length: 32 characters. | | nickName | String | No | Nick name.
Max. length: 256 characters. | | userName | **UserName** | No | User name, fullName and (firstName,middleName,lastName) can not both empty. | | userAddresses | Array< **Address** > | No | The user's address information. | | avatar | String | No | Avatar url.
Max. length: 256 characters. | | gender | String | No | F: female; M: Male.
Max. length: 32 characters. | | birthday | String/Datetime | No | Birthday which follows the ISO 8601 standard.
Max. length: 32 characters. | | nationality | String | No | Nationality, alpha-2 code according to ISO3166. e.g. JP, US.
Max. length: 32 characters. | | loginIdInfos | Array< **OpenLoginIdInfo** > | No | User login id info list. | | contactInfos | Array< **ContactInfo** > | No | Contact info list. | | extendInfo | String | No | The extend information,wallet and merchant can put extend info here.
Max. length: 4096 characters. |

PaymentMethod

| | | | | | --- | --- | --- | --- | | **Property** | **Data type** | **Required** | **Description** | | paymentMethodType | String | Yes | Payment method type, used to identifier a payment method.
Max. length: 32 characters. | | paymentMethodId | String | No | The uniqueId of a customer belong to a paymentMethod.
Max. length: 128 characters. | | extendMetaData | String | No | Extended information.
Max. length: 2048 characters. |

Result

| | | | | | --- | --- | --- | --- | | **Property** | **Data type** | **Required** | **Description** | | resultStatus | String | No | Result status. Valid values are:
- S: Successful
- F: Failed
- U: Unknown
- A: accepted, not yet succeed, but can proceed with some actions. | | resultCode | String | No | Result code.
Max. length: 64 characters. | | resultMessage | String | No | Result message that describes resultCode in detail.
Max. length: 256 characters. |

UserName

| | | | | | --- | --- | --- | --- | | **Property** | **Data type** | **Required** | **Description** | | fullName | String | No | Full Name.
Max. length: 128 characters. | | firstName | String | No | First Name.
Max. length: 32 characters. | | middleName | String | No | Middle Name.
Max. length: 32 characters. | | lastName | String | No | Last Name.
Max. length: 32 characters. |

Order

| | | | | | --- | --- | --- | --- | | **Property** | **Data type** | **Required** | **Description** | | referenceOrderId | String | Yes | The unique identification of the order on the merchant side. It is used for the display of user consumption records, and the subsequent payment operations such as customer complaints and disputes track.
Max. length: 64 characters. | | orderDescription | String | No | Description of the order used to display user consumption records, etc.
Max. length: 256 characters. | | orderAmount | **Amount** | Yes | The amount of an order, like how much to charge in the specified currency unit for an order. | | orderCreateTime | String/Datetime | No | Order create time from merchant which follows the ISO 8601 standard.
Max. length: 32 characters. | | referenceMerchant | **Merchant** | No | Merchant information. | | goods | **Goods** | No | Goods information. | | shipping | **Shipping** | No | Shipping information. | | buyer | **Buyer** | No | Buyer information. | | extendInfo | String | No | Extended infomation data, this field includes information that are not common but needed for special use cases.
Max. length: 2048 characters. |

Merchant

| | | | | | --- | --- | --- | --- | | **Property** | **Data type** | **Required** | **Description** | | referenceMerchantId | String | Yes | Merchant ID.
Max. length: 32 characters. | | merchantMCC | String | Yes | Merchant MCC.
Max. length: 32 characters. | | merchantName | String | Yes | Name of Merchant.
Max. length: 256 characters. | | merchantDisplayName | String | No | Display name of merchant.
Max. length: 64 characters. | | merchantAddress | **Address** | No | The address of merchant. | | merchantRegisterDate | String/Datetime | No | Merchant register time from merchant which follows the ISO 8601 standard.
Max. length: 32 characters. | | store | **Store** | No | Merchant store. |

Address

| | | | | | --- | --- | --- | --- | | **Property** | **Data type** | **Required** | **Description** | | region | String | Yes | Alpha-2 code according to ISO3166. e.g. JP, US.
Max. length: 2 characters. | | state | String | No | State/County/Province.
Max. length: 8 characters. | | city | String | No | City/District/Suburb/Town/Village.
Max. length: 32 characters. | | address1 | String | No | Address line 1(Street address/PO Box/Company name).
Max. length: 256 characters. | | address2 | String | No | Address line 2(Apartment/Suite/Unit/Building).
Max. length: 256 characters. | | zipCode | String | No | ZIP or postal code.
Max. length: 32 characters. |

Store

| | | | | | --- | --- | --- | --- | | **Property** | **Data type** | **Required** | **Description** | | referenceStoreId | String | Yes | The store belongs to a merchant, and the ID assigned by the corresponding merchant to the store is unique under the merchant.
Max. length: 64 characters. | | storeName | String | No | Name of store.
Max. length: 256 characters. | | storeMCC | String | No | Store business category code.

Max. length: 32 characters. | | storeDisplayName | String | No | Display name of store.
Max. length: 64 characters. | | storeTerminalId | String | No | Unique identifier of store's terminal.
Max. length: 64 characters. | | storeOperatorId | String | No | Unique identifier of store's terminal operator.
Max. length: 64 characters. | | storeAddress | **Address** | No | The address of the store. | | storePhoneNo | String | No | Phone number of Store.
Max. length: 16 characters. |

Goods

| | | | | | --- | --- | --- | --- | | **Property** | **Data type** | **Required** | **Description** | | referenceGoodsId | String | Yes | Unique ID of goods.
Max. length: 64 characters. | | goodsName | String | Yes | Name of goods.
Max. length: 256 characters. | | goodsCategory | String | No | Category of goods.
Max. length: 256 characters. | | goodsBrand | String | No | Brand of goods.
Max. length: 32 characters. | | goodsUnitAmount | **Amount** | No | Order amount for display of user consumption records, payment results page. | | goodsQuantity | String | No | Quantity of goods.
Max. length: 32 characters. | | goodsUrl | String | No | Goods url.
Max. length: 1024 characters. | | extendInfo | String | No | The extend information of goods.
Max. length: 2048 characters. |

Shipping

| | | | | | --- | --- | --- | --- | | **Property** | **Data type** | **Required** | **Description** | | shippingName | **UserName** | Yes | Shipping name. | | shippingAddress | **Address** | Yes | Shipping address. | | shippingCarrier | String | No | The delivery service that shipped a physical product, such as Fedex, UPS, USPS, etc.
Max. length: 128 characters. | | shippingPhoneNo | String | No | Recipient PhoneNo(including extension).
Max. length: 16 characters. | | shippingFee | **Amount** | No | Shipping fee. |

Buyer

| | | | | | --- | --- | --- | --- | | **Property** | **Data type** | **Required** | **Description** | | referenceBuyerId | String | No | Unique Identification of buyer.
Max. length: 64 characters. | | buyerName | **UserName** | No | Buyer name. | | buyerPhoneNo | String | No | Mobile phone number of buyer.
Max. length: 24 characters. |

Enum

OsType

| | | | --- | --- | | **Key** | **Description** | | IOS | iOS. | | ANDROID | Android. |

PaymentMethodType

| | | | --- | --- | | **Key** | **Description** | | BALANCE | Balance. | | COUPON | Coupon. | | CREDIT_CARD | Credit Card. | | DEBIT_CARD | Debit Card. |

TerminalType

| | | | --- | --- | | **Key** | **Description** | | MINI_APP | Mini program. | | APP | Mobile Application. | | WEB | Broswer Web. | | WAP | Mobile Wap. | | SYSTEM | System Call. |

Source: https://miniprogram.gcash.com/docs/miniprogram_gcash/mpdev/vs3pkf

---

Data dictionary (for v1) {#data-dictionary-for-v1-}

Last updated: 2021-05-09

Path: miniprogram_gcash

Data dictionary (for v1)

2021-05-09 18:43

Model

Amount

| | | | | | --- | --- | --- | --- | | **Property** | **Data type** | **Required** | **Description** | | currency | String | Yes | The three-character ISO-4217 currency code.
Max. length: 3 characters. | | value | String | Yes | A string that encloses a positive integer representing how much to charge in the smallest currency unit(e.g., 100 cents to charge $1.00 or 100 to charge $100, a zero-decimal currency).
Max. length: 16 characters. |

ActionForm

| | | | | | --- | --- | --- | --- | | **Property** | **Data type** | **Required** | **Description** | | actionFormType | String | No | The action form type. Enum:
ORDER_CODE,
REDIRECTION. | | orderCode | String | No | The order code value.
Max. length: 2048 characters. | | redirectionUrl | String | No | The url of redirect.
Max. length: 4096 characters. |

EnvInfo

| | | | | | --- | --- | --- | --- | | **Property** | **Data type** | **Required** | **Description** | | terminalType | String | No | The terminal type of this request.
Enum: [MINI_APP, APP, WEB, WAP, SYSTEM]. | | osType | String | No | OS type.
Enum: [IOS, ANDROID] | | userAgent | String | No | User agent.
Max. length: 1024 characters. | | deviceTokenId | String | No | Token ID of the device.
Max. length: 128 characters. | | clientIp | String | No | IP address of the client device.
Max. length: 64 characters. | | cookieId | String | No | User cookie ID.
Max. length: 128 characters. | | extendInfo | String | No | Extend info.
Max. length: 4096 characters. |

PaymentFactor

| | | | | | --- | --- | --- | --- | | **Property** | **Data type** | **Required** | **Description** | | isPaymentEvaluation | Boolean | No | An indicator of the payment evaluation. If the value is TRUE, the payment is only to evaluate whether the payment can be successful and no actual funds deduction occurs. The default value is FALSE. | | isOrderCode | Boolean | No | If the payment senario is the user scan the code presented by merchant and identify the order, and make payment. The default value is FALSE. | | isPaymentCode | Boolean | No | An indicator of whether the payment scenario is the merchant scan the user payment code.The default value is FALSE. | | isAgreementPay | Boolean | No | An indicator of whether the payment is an agreement payment.The default value is FALSE. | | isCashierPayment | Boolean | No | An indicator of whether the payment is a cashier payment.The default value is FALSE. | | isAuthorizationAndPay | Boolean | No | An indicator of whether to do agreementPay authorization during the payment.The default value is FALSE. | | isAuthorizationPayment | Boolean | No | An indicator of whether the payment is an authorization payment.The default value is FALSE. |

OpenLoginIdInfo

| | | | | | --- | --- | --- | --- | | **Property** | **Data type** | **Required** | **Description** | | loginId | String | No | Login Id is a identification for an user, which can be mobile number or email. User can use login Id to login Wallet.
Max. length: 64 characters. | | loginIdType | String | Yes | Login id type, Enum: "MOBILE_PHONE", "EMAIL".
Max. length: 64 characters. | | maskLoginId | String | No | Mask Login id, several bits of the phone number will be hidden to protect users' privacy.
Max. length: 64 characters. | | hashLoginId | String | No | Hash login id. The login id hashed by hash algorithm. The external system can use it to compare it's login id hashed by the same hash algorithm to see if the login id is the same.
Max. length: 256 characters. | | extendInfo | String | No | The extend information,wallet and merchant can put extend info here.
Max. length: 4096 characters. |

ContactInfo

| | | | | | --- | --- | --- | --- | | **Property** | **Data type** | **Required** | **Description** | | contactNo | String | Yes | Contact No. e.g mobile-phone No, e-mail address.
Max. length: 64 characters. | | contactType | String | Yes | Contact type, there are agreed types(MOBILE_PHONE, TELEPHONE, EMAIL) that should be available to all users, developer can add new type, but developer should consider the compatibility.

Max. length: 32 characters. | | extendInfo | String | No | The extend information,wallet and merchant can put extend info here.
Max. length: 4096 characters. |

OpenUserInfo

| | | | | | --- | --- | --- | --- | | **Property** | **Data type** | **Required** | **Description** | | userId | String | No | The hashed unique identifier allocated for user.
Max. length: 64 characters. | | status | String | No | User status, Enum: "ACTIVE", "FROZEN", "INACTIVE".
Max. length: 32 characters. | | nickName | String | No | Nick name.
Max. length: 256 characters. | | userName | **UserName** | No | User name, fullName and (firstName,middleName,lastName) can not both empty. | | userAddresses | Array< **Address** > | No | The user's address information. | | avatar | String | No | Avatar url.
Max. length: 256 characters. | | gender | String | No | F: female; M: Male.
Max. length: 32 characters. | | birthday | String/Datetime | No | Birthday which follows the ISO 8601 standard.
Max. length: 32 characters. | | nationality | String | No | Nationality, alpha-2 code according to ISO3166. e.g. JP, US.
Max. length: 32 characters. | | loginIdInfos | Array< **OpenLoginIdInfo** > | No | User login id info list. | | contactInfos | Array< **ContactInfo** > | No | Contact info list. | | extendInfo | String | No | The extend information,wallet and merchant can put extend info here.
Max. length: 4096 characters. |

PaymentMethod

| | | | | | --- | --- | --- | --- | | **Property** | **Data type** | **Required** | **Description** | | paymentMethodType | String | Yes | Payment method type, used to identifier a payment method.
Max. length: 32 characters. | | paymentMethodId | String | No | The uniqueId of a customer belong to a paymentMethod.
Max. length: 128 characters. | | extendMetaData | String | No | Extended information.
Max. length: 2048 characters. |

Result

| | | | | | --- | --- | --- | --- | | **Property** | **Data type** | **Required** | **Description** | | resultStatus | String | No | Result status. Valid values are:
- S: Successful
- F: Failed
- U: Unknown
- A: accepted, not yet succeed, but can proceed with some actions. | | resultCode | String | No | Result code.
Max. length: 64 characters. | | resultMessage | String | No | Result message that describes resultCode in detail.
Max. length: 256 characters. |

UserName

| | | | | | --- | --- | --- | --- | | **Property** | **Data type** | **Required** | **Description** | | fullName | String | No | Full Name.
Max. length: 128 characters. | | firstName | String | No | First Name.
Max. length: 32 characters. | | middleName | String | No | Middle Name.
Max. length: 32 characters. | | lastName | String | No | Last Name.
Max. length: 32 characters. |

Order

| | | | | | --- | --- | --- | --- | | **Property** | **Data type** | **Required** | **Description** | | referenceOrderId | String | Yes | The unique identification of the order on the merchant side. It is used for the display of user consumption records, and the subsequent payment operations such as customer complaints and disputes track.
Max. length: 64 characters. | | orderDescription | String | No | Description of the order used to display user consumption records, etc.
Max. length: 256 characters. | | orderAmount | **Amount** | Yes | The amount of an order, like how much to charge in the specified currency unit for an order. | | orderCreateTime | String/Datetime | No | Order create time from merchant which follows the ISO 8601 standard.
Max. length: 32 characters. | | referenceMerchant | **Merchant** | No | Merchant information. | | goods | **Goods** | No | Goods information. | | shipping | **Shipping** | No | Shipping information. | | buyer | **Buyer** | No | Buyer information. | | extendInfo | String | No | Extended infomation data, this field includes information that are not common but needed for special use cases.
Max. length: 2048 characters. |

Merchant

| | | | | | --- | --- | --- | --- | | **Property** | **Data type** | **Required** | **Description** | | referenceMerchantId | String | Yes | Merchant ID.
Max. length: 32 characters. | | merchantMCC | String | Yes | Merchant MCC.
Max. length: 32 characters. | | merchantName | String | Yes | Name of Merchant.
Max. length: 256 characters. | | merchantDisplayName | String | No | Display name of merchant.
Max. length: 64 characters. | | merchantAddress | **Address** | No | The address of merchant. | | merchantRegisterDate | String/Datetime | No | Merchant register time from merchant which follows the ISO 8601 standard.
Max. length: 32 characters. | | store | **Store** | No | Merchant store. |

Address

| | | | | | --- | --- | --- | --- | | **Property** | **Data type** | **Required** | **Description** | | region | String | Yes | Alpha-2 code according to ISO3166. e.g. JP, US.
Max. length: 2 characters. | | state | String | No | State/County/Province.
Max. length: 8 characters. | | city | String | No | City/District/Suburb/Town/Village.
Max. length: 32 characters. | | address1 | String | No | Address line 1(Street address/PO Box/Company name).
Max. length: 256 characters. | | address2 | String | No | Address line

2(Apartment/Suite/Unit/Building).
Max. length: 256 characters. | | zipCode | String | No | ZIP or postal code.
Max. length: 32 characters. |

Store

| | | | | | --- | --- | --- | --- | | **Property** | **Data type** | **Required** | **Description** | | referenceStoreId | String | Yes | The store belongs to a merchant, and the ID assigned by the corresponding merchant to the store is unique under the merchant.
Max. length: 64 characters. | | storeName | String | No | Name of store.
Max. length: 256 characters. | | storeMCC | String | No | Store business category code.
Max. length: 32 characters. | | storeDisplayName | String | No | Display name of store.
Max. length: 64 characters. | | storeTerminalId | String | No | Unique identifier of store's terminal.
Max. length: 64 characters. | | storeOperatorId | String | No | Unique identifier of store's terminal operator.
Max. length: 64 characters. | | storeAddress | **Address** | No | The address of the store. | | storePhoneNo | String | No | Phone number of Store.
Max. length: 16 characters. |

Goods

| | | | | | --- | --- | --- | --- | | **Property** | **Data type** | **Required** | **Description** | | referenceGoodsId | String | Yes | Unique ID of goods.
Max. length: 64 characters. | | goodsName | String | Yes | Name of goods.
Max. length: 256 characters. | | goodsCategory | String | No | Category of goods.
Max. length: 256 characters. | | goodsBrand | String | No | Brand of goods.
Max. length: 32 characters. | | goodsUnitAmount | **Amount** | No | Order amount for display of user consumption records, payment results page. | | goodsQuantity | String | No | Quantity of goods.
Max. length: 32 characters. | | goodsUrl | String | No | Goods url.
Max. length: 1024 characters. | | extendInfo | String | No | The extend information of goods.
Max. length: 2048 characters. |

Shipping

| | | | | | --- | --- | --- | --- | | **Property** | **Data type** | **Required** | **Description** | | shippingName | **UserName** | Yes | Shipping name. | | shippingAddress | **Address** | Yes | Shipping address. | | shippingCarrier | String | No | The delivery service that shipped a physical product, such as Fedex, UPS, USPS, etc.
Max. length: 128 characters. | | shippingPhoneNo | String | No | Recipient PhoneNo(including extension).
Max. length: 16 characters. | | shippingFee | **Amount** | No | Shipping fee. |

Buyer

| | | | | | --- | --- | --- | --- | | **Property** | **Data type** | **Required** | **Description** | | referenceBuyerId | String | No | Unique Identification of buyer.
Max. length: 64 characters. | | buyerName | **UserName** | No | Buyer name. | |

buyerPhoneNo | String | No | Mobile phone number of buyer.
Max. length: 24 characters. |

Enum

OsType

| | | | --- | --- | | **Key** | **Description** | | IOS | iOS. | | ANDROID | Android. |

PaymentMethodType

| | | | --- | --- | | **Key** | **Description** | | BALANCE | Balance. | | COUPON | Coupon. | | CREDIT_CARD | Credit Card. | | DEBIT_CARD | Debit Card. |

TerminalType

| | | | --- | --- | | **Key** | **Description** | | MINI_APP | Mini program. | | APP | Mobile Application. | | WEB | Broswer Web. | | WAP | Mobile Wap. | | SYSTEM | System Call. |

Source: https://miniprogram.gcash.com/docs/miniprogram_gcash/mpdev-old/vs3pkf

---

Data dictionary (for v2) {#data-dictionary-for-v2}

Last updated: 2022-07-05

Path: miniprogram_gcash

Data dictionary (for v2)

2022-07-05 23:31

Model

Amount

| | | | | | --- | --- | --- | --- | | **Property** | **Data type** | **Required** | **Description** | | currency | String | Yes | The three-character ISO-4217 currency code.
Max. length: 3 characters. | | value | String | Yes | A string that encloses a positive integer representing how much to charge in the smallest currency unit(e.g., 100 cents to charge $1.00 or 100 to charge $100, a zero-decimal currency).
Max. length: 16 characters. |

ActionForm

| | | | | | --- | --- | --- | --- | | **Property** | **Data type** | **Required** | **Description** | | actionFormType | String | No | The action form type.
Enum: [ORDER_CODE , REDIRECTION] | | orderCode | String | No | The order code value. The order code is generated by merchants, which is a QR code with order information for customers to scan.
Max. length: 2048 characters. | | redirectionUrl | String | No | The URL of redirect.
Max. length: 4096 characters. |

Env

| | | | | | --- | --- | --- | --- | | **Property** | **Data type** | **Required** | **Description** | | terminalType | String | No | The terminal type of this request.
Enum: [MINI_APP, APP, WEB, WAP, SYSTEM] | | osType | String | No | OS type.
Enum: [IOS, ANDROID] | | userAgent | String | No | User agent.
Max. length: 1024 characters. | | deviceTokenId | String | No | Token ID of the device.
Max. length: 128 characters. | | clientIp | String | No | IP address of the client device.
Max. length: 64 characters. | | cookieId | String | No | User cookie ID.
Max. length: 128 characters. | | storeTerminalId | String | No | The store terminal ID.
Max. length: 64 characters. | | storeTerminal
RequestTime | String/Datetime | No | The store terminal request time.
Max. length: 32 characters. | | extendInfo | String | No | Extend info.
Max. length: 4096 characters. |

PaymentFactor

| | | | | | --- | --- | --- | --- | | **Property** | **Data type** | **Required** | **Description** | | needSurcharge | Boolean | No | An indicator of a surcharge, which appears in the request from APS to Mobile Payment Partner. If the value is TRUE, the fields `surchargeAmount` and `surchargeQuote` must be specified. The default value is FALSE. | | isPaymentEvaluation | Boolean | No | An indicator of the payment evaluation. If the value is TRUE, the payment is only to evaluate whether the payment can be successful and no actual funds deduction occurs. The default value is FALSE. | | isAuthorizationAndPay | Boolean | No | An indicator of whether to do authorization for the agreement payment (Auto Debit) during the payment. The default value is FALSE. | | isAuthorizationPayment | Boolean | No | An indicator of whether the payment is an authorization payment.The default value is FALSE. | | isDeferredPayment | Boolean | No | An indicator of whether the payment is a deferred payment, in which scenario the user uses the product or service in advance. | | needCheckCompliance | Boolean | No | An indicator of whether the payment information must be validated to meet compliances before the payment is processed. | | needOtpVerification | Boolean | No | An indicator of whether the payment needs to verify OTP (one time password). | | isCrossborderSettlement | Boolean | No | An indicator of whether the payment requires cross-border settlement. | | inStorePaymentScenario | Boolean | No | An indicator of `PaymentCode/OrderCode/EntryCode` .|

User

| | | | | | --- | --- | --- | --- | | **Property** | **Data type** | **Required** | **Description** | | userId | String | No | The hashed unique identifier allocated for user.
Max. length: 64 characters. | | status | String | No | User status, Enum: "ACTIVE",

"FROZEN", "INACTIVE"
Max. length: 32 characters. | | nickName | String | No | Nick name.
Max. length: 256 characters. | | userName | **UserName** | No | User name, fullName and (firstName,middleName,lastName) can not both empty. | | userAddresses | Array< **Address** > | No | The user's address information. | | avatar | String | No | Avatar url.
Max. length: 256 characters. | | gender | String | No | F: female; M: Male.
Max. length: 32 characters. | | birthDate | String/Datetime | No | Birth date which follows the ISO 8601 standard.
Max. length: 32 characters. | | nationality | String | No | Nationality, alpha-2 code according to ISO3166. e.g. JP, US.
Max. length: 32 characters. | | loginIdInfos | Array< **LoginIdInfo** > | No | A list of user login IDs. | | contactInfos | Array< **ContactInfo** > | No | A list of contact information. | | extendInfo | String | No | The extend information,wallet and merchant can put extend info here.
Max. length: 4096 characters. |

LoginIdInfo

| | | | | | --- | --- | --- | --- | | **Property** | **Data type** | **Required** | **Description** | | loginIdType | String | Yes | The types of Login ID:
- `MOBILE_PHONE`
- `EMAIL`
- `OTHER`
Max. length: 64 characters. | | loginId | String | No | An unique identifier for an user's ID login, which can be a mobile number or an email address. Users can use their login ID that without hidden bits to log in to the Wallet.
Max. length: 64 characters. | | maskLoginId | String | No | Mask Login ID that represents several bits of the phone number that are hidden to protect users' privacy.
Max. length: 64 characters. | | hashLoginId | String | No | Hash login ID that identifies an `loginId` that is hashed by a hash algorithm. The external system can use it to compare its login ID that is hashed by the same hash algorithm to check whether the login ID is identical.
Max. length: 256 characters. | | extendInfo | String | No | The extensive information for wallets and merchants to add.
Max. length: 4096 characters. |

ContactInfo

| | | | | | --- | --- | --- | --- | | **Property** | **Data type** | **Required** | **Description** | | contactType | String | Yes | The followings are contact types that are available to all users:
- `MOBILE_PHONE`
- `TELEPHONE`
- `EMAIL`
Tips: Developer can add new types, but should consider the compatibility.
Max. length: 32 characters. | | contactNo | String | Yes | The value that corresponds to the contact type that is configured in the `contactType` field. For example, this field can be mobile phone numer, or e-mail address.
Max. length: 64 characters. | | extendInfo | String | No | The extensive information for wallets and merchants to add.
Max. length: 4096 characters. |

PaymentMethod

| | | | | | --- | --- | --- | --- | | **Property** | **Data type** | **Required** | **Description** | | paymentMethodType | String | Yes | Payment method type, used to identifier a payment method.
Max. length: 32 characters. | | paymentMethodId | String | No | The uniqueId of a customer belong to a paymentMethod.
Max. length: 128 characters. | | paymentMethodMetaData | String | No | The payment method metadata.
Max. length: 2048 characters. |

Result

| | | | | | --- | --- | --- | --- | | **Property** | **Data type** | **Required** | **Description** | | resultStatus | String | No | Result status. Valid values are:
- `S` : Successful
- `F` : Failed
- `U` : Unknown
- `A` : accepted, not yet succeed, but can proceed with some actions. | | resultCode | String | No | Result code.
Max. length: 64 characters. | | resultMessage | String | No | Result message that describes the result code in details.
Max. length: 256 characters. |

UserName

| | | | | | --- | --- | --- | --- | | **Property** | **Data type** | **Required** | **Description** | | fullName | String | No | Full Name.
Max. length: 128 characters. | | firstName | String | No | First Name.
Max. length: 32 characters. | | middleName | String | No | Middle Name.
Max. length: 32 characters. | | lastName | String | No | Last Name.
Max. length: 32 characters. |

Order

| | | | | | --- | --- | --- | --- | | **Property** | **Data type** | **Required** | **Description** | | referenceOrderId | String | Yes | The unique identification of the order on the merchant side. It is used for the display of user consumption records, and the subsequent payment operations such as customer complaints and disputes track.
Max. length: 64 characters. | | orderDescription | String | Yes | Description of the order used to display user consumption records, etc.
Max. length: 256 characters. | | orderAmount | **Amount** | Yes | The amount of an order, like how much to charge in the specified currency unit for an order. | | orderCreateTime | String/Datetime | No | Order create time from merchant which follows the ISO 8601 standard.
Max. length: 32 characters. | | merchant | **Merchant** | Yes | Merchant information. | | goods | **Array< Good >** | No | Goods information. | | shipping | **Shipping** | No | Shipping information. | | buyer | **Buyer** | No | Buyer information. | | env | **Env** | No | The order environment information, such as the device information. | | extendInfo | String | No | Extended infomation data, this field includes information that are not common but needed for special use cases.
Max. length: 2048 characters. |

Transaction

| | | | | --- | --- | --- | --- | | **Property** | **Data type** | **Required** | **Description** | | transactionResult | Result | Yes | The transaction result, which contains information related to the business result and error information. | | transactionType | String | Yes | Transaction type for each subsequent payment activity. | | transactionRequestId | String | No | The unique ID assigend by merchant to identify the transaction request. When the transaction type is CAPTURE, the value of this field is identical to captureRequestId. When the transaction type is REFUND, the value of this field is identical to refundRequestId. | | transactionId | String | Yes | The unique ID assigned by wallet to identify a transaction. When the transaction type is CAPTURE, the value of this field is identical to captureId. When the transaction type is REFUND, the value of this field is identical to refundId. | | transactionStatus | String | No | Transaction status type. | | transactionAmount | Amount | Yes | Transaction amount. When the transaction type is CAPTURE, the value of this field is identical to captureAmount. When the transaction type is REFUND, the value of this field is identical to refundAmount. | | transactionTime | Date | No | Transaction time | | isLastCapture | String | No | Only for capture transaction | | voidSource | String | No | void source | | extendInfo | String | No | The extensive information. The wallet and merchant can put extensive information in this property. Max. length: 4096 characters. |

Merchant

| | | | | --- | --- | --- | --- | | **Property** | **Data type** | **Required** | **Description** | | referenceMerchantId | String | Yes | Merchant ID.
Max. length: 32 characters. | | merchantMCC | String | Yes | Merchant MCC (merchant category code).
Max. length: 32 characters. | | merchantName | String | Yes | Name of Merchant.
Max. length: 256 characters. | | merchantDisplayName | String | No | Display name of merchant.
Max. length: 64 characters. | | merchantAddress | **Address** | No | The address of merchant. | | merchantRegisterDate | String /Datetime | No | Merchant register time from merchant which follows the ISO 8601 standard.
Max. length: 32 characters. | | store | **Store** | No | Merchant store. |

Address

| | | | | --- | --- | --- | --- | | **Property** | **Data type** | **Required** | **Description** | | region | String | Yes | Alpha-2 code according to ISO3166. e.g. JP, US.
Max. length: 2 characters. | | state | String | No | State/County/Province.
Max. length: 8 characters. | | city | String | No | City/District/Suburb/Town/Village.
Max. length: 32 characters. | | address1 | String | No | Address line 1(Street address/PO Box/Company name).
Max. length: 256 characters. | | address2 | String | No | Address line 2(Apartment/Suite/Unit/Building).
Max. length: 256 characters. | | label | String | No | Label for address, e.g home, company.
Max. length: 64 characters. | | zipCode | String | No | ZIP or postal code.
Max. length: 32 characters. | | extendInfo | String | No | The extend information of goods.
Max. length: 4096 characters. |

Store

| | | | | | --- | --- | --- | --- | | **Property** | **Data type** | **Required** | **Description** | | referenceStoreId | String | Yes | The store belongs to a merchant, and the ID assigned by the corresponding merchant to the store is unique under the merchant.
Max. length: 64 characters. | | storeName | String | No | Name of store.
Max. length: 256 characters. | | storeMCC | String | No | Store business category code.
Max. length: 32 characters. | | storeDisplayName | String | No | Display name of store.
Max. length: 64 characters. | | storeTerminalId | String | No | Unique identifier of store's terminal.
Max. length: 64 characters. | | storeOperatorId | String | No | Unique identifier of store's terminal operator.
Max. length: 64 characters. | | storeAddress | **Address** | No | The address of the store. | | storePhoneNo | String | No | Phone number of Store.
Max. length: 16 characters. |

Goods

| | | | | | --- | --- | --- | --- | | **Property** | **Data type** | **Required** | **Description** | | referenceGoodsId | String | Yes | Unique ID of goods.
Max. length: 64 characters. | | goodsName | String | Yes | Name of goods.
Max. length: 256 characters. | | goodsCategory | String | No | Category of goods.
Max. length: 256 characters. | | goodsBrand | String | No | Brand of goods.
Max. length: 32 characters. | | goodsUnitAmount | **Amount** | Yes | Order amount for display of user consumption records, payment results page. | | goodsQuantity | String | No | Quantity of goods.
Max. length: 32 characters. | | goodsSkuName | String | No | Goods sku name.
Max. length: 64 characters. | | goodsUrl | String | No | Goods url.
Max. length: 1024 characters. | | extendInfo | String | No | The extend information of goods.
Max. length: 2048 characters. |

Shipping

| | | | | | --- | --- | --- | --- | | **Property** | **Data type** | **Required** | **Description** | | shippingName | **UserName** | Yes | Shipping name. | | shippingAddress | **Address** | Yes | Shipping address. | | shippingCarrier | String | No | The delivery service that shipped a physical product, such as Fedex, UPS, USPS, etc.
Max. length: 128 characters. | | shippingPhoneNo | String | No | Recipient PhoneNo(including extension).
Max. length: 16 characters. | | shippingFee | **Amount** | No | Shipping fee. |

Buyer

| | | | | | --- | --- | --- | --- | | **Property** | **Data type** | **Required** | **Description** | | referenceBuyerId | String | No | Unique Identification of buyer.
Max. length: 64 characters. | | buyerName | **UserName** | No | Buyer name. | | buyerPhoneNo | String | No | Mobile phone number of buyer.
Max. length: 24 characters. |

ContentTemplate

| | | | | --- | --- | --- | --- | | **Property** | **Data type** | **Required** | **Description** | | templateParameters | Map<String,String> | No | A string-to-string mapping. The data structure is in JSON format:
`"templateParameters": {"key": "value"}`
, where:
- `key` : represents the variable that is defined in the template.
- `value` : is used to replace the value of the `key` parameter in the template. | | language | String | No | RFC 1766, such as zh-CN, en-US. |

RedirectActionForm

| | | | | --- | --- | --- | --- | | **Property** | **Data type** | **Required** | **Description** | | method | String | No | The HTTP method. The value is `POST` or `GET` . | | parameters | String | No | The parameters that are required for the HTTP method, which is in the `key:value` pair format.
Max. length: 2048 characters. | | redirectUrl | String | Yes | The URL of redirect.
Max. length: 2048 characters. |

Message

| | | | | --- | --- | --- | --- | | **Property** | **Data Type** | **Required** | **Description** | | messageId | String | Yes | The unique ID is generated by Mini Program Platform to identify a notification.
- Maximum length: 64 characters.
- Characters not allowed: special characters such as @, #, ? and so on.
Note: This field is an API idempotency field. For the notifications which are sent with the same `requestId` and reach an S status, the native app must return the same `messageId`. See the Idempotency for details about API idempotency. | | customerId | String | Yes | The unique ID is generated by the native app to identify a user.
Example: 218820000000000000023.
Maximum length: 32 characters. | | messageChannel | MessageChannel | Yes | The channels to send a notification, including push, SMS, and Inbox. | | templateCode | String | No | Indicates the template ID that is generated by Mini Program Platform to identify a template.
- Maximum length: 64 characters.
- Characters not allowed: special characters such as @, #, ? and so on.
- Can be `Null`. | | messageContent | String | Yes | The specific content that a notification to send.
Maximum length: 4096 characters.
- Push

```
{
"en-US": {
  "title": "testTitle",
  "content": "testContent",
  "linkUrl": "testUrl"
},
"zh-CN": {
  "title": "testTitle",
  "content": "testContent",
  "linkUrl": "testUrl"
```

```
}
}
- SMS
{
"en-US": {
  "content": "this is sms content"
},
"zh-CN": {
  "content": "this is sms content"
}
}
- Inbox
{
"en-US": {
  "displayType": "inbox_item_content",
  "header": {
    "appIcon": "icon url",
    "appName": "icon name",
    "msgBizName": "test_biz",
    "target": {
      "path": "header link url"
    }
  },
  "body": {
    "title": "message title",
    "content": "message content",
    "target": {
      "path": "message path url"
    }
  },
  "footer": [
    {
      "linkName": "link name",
      "target": {
        "path": "header link url"
      }
    }
  ]
},
"zh-CN": {
  "displayType": "inbox_item_content",
  "header": {
    "appIcon": "icon url",
    "appName": "icon name",
    "msgBizName": "test_biz",
    "target": {
      "path": "header link url"
    }
  },
  "body": {
    "title": "message title",
    "content": "message content",
    "target": {
      "path": "message path url"
```

```
    }
   },
   "footer": [
    {
     "linkName": "link name",
     "target": {
      "path": "header link url"
     }
    }
   ]
}
```

} | | extendInfo | String | No | The extended information.
- Maximum length: 4096 characters.
- Characters not allowed: special characters such as @, #, ? and so on.
- Can be `Null`. |

MessageSendResult

| | | | | | --- | --- | --- | --- | | **Property** | **Data Type** | **Required** | **Description** | | messageId | String | Yes | The unique ID is generated by Mini Program Platform to identify a notification.
- Maximum length: 64 characters.
- Characters not allowed: special characters such as @, #, ? and so on.
Note: This field is an API idempotency field. For the notifications which are sent with the same `requestId` and reach an S status, the native app must return the same `messageId`. See the Idempotency for details about API idempotency. | | success | Boolean | Yes | Indicates whether the notification is successfully sent to the native app. Valid values are:
- `true`: The notification is sent successfully.
- `false`: The notification fails to send. | | retry | Boolean | No | Indicates whether the native app allows you to retry when the notification fails to send. Valid values are:
- `true`: The native app server allows to retry under specific scenarios, for example, when there are system exceptions.
- `false`: The native app server does not allow to retry. For example, when the delivery frequency of the notification is beyond the limits set by the native app, retry is not allowed.
Note: This field is required when the value of the `success` field is `false`. |

Enum

OsType

| | | | --- | --- | | **Key** | **Description** | | IOS | iOS. | | ANDROID | Android. |

PaymentMethodType

| | | | --- | --- | | **Key** | **Description** | | BALANCE | Balance. | | COUPON | Coupon. | | CREDIT_CARD | Credit Card. | | DEBIT_CARD | Debit Card. |

TerminalType

| | | --- | --- | | **Key** | **Description** | | MINI_APP | Mini program. | | APP | Mobile Application. | | WEB | Broswer Web. | | WAP | Mobile Wap. | | SYSTEM | System Call. |

MessageChannel

| | | --- | --- | | **Key** | **Description** | | SMS | SMS | | PUSH | PUSH | | INBOX | INBOX |

Source: https://miniprogram.gcash.com/docs/miniprogram_gcash/mpdev/v2_data_dict

---

Define an event {#define-an-event}

Last updated: 2022-07-07

Path: miniprogram_gcash

Define an event

2022-07-07 17:08

Data is stored and analyzed based on an event, which is triggered by user interaction with a mini program. See Event management and analysis for more details.

User experience

For example, the following figure illustrates the purchase process of an e-commerce mini program user:

Based on the user journey above, you can define the following events:

- View the home page
- View product details
- Add the product to shopping cart
- Submit an order
- Pay

Procedures

This section describes in detail how to define an event in four major steps:

1. Create an event

2. Enter the event name

3. Choose the data reporting method

4. Save the event

The "submit an order" and "view the home page" events mentioned above will serve as examples to illustrate the procedures.

1. Create an event

1. Go to **Analytics** > **Performance** > **My Analysis**;
2. Choose the mini program that you want to perform data analysis on;
3. Click **Manage Event**;
4. Click **+ New Event** on the **Manage Event** page.

2. Enter the event name

Enter an event name that complies with the naming rules. For example, `submitOrder` will be a suitable name for the "submit an order" event.

3. Choose the data reporting method

Data reporting methods define how data are to be analyzed. For example, with the Data Reporting by Self-Defined Actions method, you can define actions with different parameters to collect and report data automatically. For more information, see Data Reporting by Self-Defined Actions.

You need to define an action and assign the following parameters:

- Trigger
- Report Type
- Page
- Element
- Field Name
- Field Value
- Field Type
- Note

Trigger

Trigger conditions. For example, `click` indicates that the event is triggered by clicking. For more information, see Trigger Conditions;

Report Type

The action occurs when the event is triggered. You can choose either of the following two types:

- Collect and report once: data collected on a user's single action.
- Collect multiple times and report once: data collected on a user's multiple actions.

Collect and Report Once

If you choose this report type, data are reported with one action. Take "submit an order" event as an example:

You also need to configure the **Action** with the following parameters:

- **Page**: This triggers a page, so you would need to enter the page path. You can find the page path via app.json files in the mini program source code of IDE (Mini Program Studio). For this action, `pages/shopping-cart/shopping-cart` is appropriate.

copy

```
app.json:
{
  "pages": [\
    "pages/handbag/handbag",\
    "pages/shopping-cart/shopping-cart",\
    "pages/confirm-order/confirm-order",\
    "pages/my-order/my-order"\
  ]
}
```

- **Element**: Enter a class or ID, which must begin with "`.`" or "`#`" respectively. You can find the element via app.json files in mini program source code of IDE (Mini Program Studio). For this action, developers have defined `.cart-footer_action` as the class for submitting an order.

Note: **Page** and **Element** may be optional for other trigger conditions.

copy

```
pages/shopping-cart/shopping-cart.axml:
<view class="cart-footer">
    <view class="cart-footer__desc">
      <view class="cart-footer__price-section">
        <view class="cart-footer__total-desc">Total  </view>
        <view class="cart-footer__total-price">¥{{submitAmount}}
</view>
      </view>
      <view class="cart-footer__discount">
        Total reduce ¥{{totalDiscount}}  store reduce
{{shopDiscount}}
      </view>
    </view>
    <view class="cart-footer__action" onTap="onSubmit">
      submit ({{count}})
    </view>
</view>
```

- **Field Name**: Fields are the metrics you would like to analyze. You can define a name and assign attributes to this metric via the **Field Value** and **Field Type**.
- **Field Value**: Enter a variable for the field. For this example, developers have defined `submitAmount` as the field to calculate the total price for the submitted orders. For **Field Value**, you need to enter the variable that is defined by developers in IDE.

copy

```
pages/shopping-cart/shopping-cart.js:
Page({
  data: {
    count: 0,
    totalDiscount: 0,
    shopDiscount: 50,
    time: new Date,
    state: '',
    submitAmount: 0,
    total: 0,
    commodity: [],
    allChecked : true,
    selectedCommoditys : []
  },
  onLoad() {
    my.setNavigationBar({
      title: 'Shopping Cart',
    });
  },
  onShow() {
    let conmondityNum = 0;
    let sumAmount = 0;
    let shoppingCartInfo = getShoppingCartInfo();
    let totalDiscount = 0;
    let commodity = [];
    if(shoppingCartInfo.length != 0){
       shoppingCartInfo.forEach(element => {
        conmondityNum += element.purchaseNum;
        sumAmount += (element.commodity.price * element.purchaseNum);
        let targetCommodity = {title: element.commodity.title,
                               description:element.commodity.type,
                               price:element.commodity.price,
                               num:element.purchaseNum,
                               id:element.commodity.id,
                               image:element.commodity.cover,
                               checked:true};
        commodity[commodity.length] = targetCommodity;
      });
      totalDiscount=conmondityNum-1;
    }
    this.setData({
           count: conmondityNum,
           totalDiscount: totalDiscount,
           shopDiscount: 50,
```

```
            time: new Date,
            state: '',
            submitAmount: sumAmount,
            total: 0,
            commodity: commodity,
            selectedCommoditys : []
          });
  },
  onSubmit(){
    if(this.data.count == 0){
        alert("The shopping cart is empty");
    }
    this.fetchCheckedConmodity();
    addOrderData(this.data.selectedCommoditys);
    my.navigateTo({
      url: '/pages/confirm-order/confirm-order',
    })
  },
  onChange(e) {
    let checkedValues = e.detail.value;
    if(checkedValues.length != 0){
       checkedValues.forEach(checkedValue =>{
          fetchSingleCommodity();
       });
    }

  },
  fetchSingleCommodity(id){
      this.data.commodity.forEach(item =>{
        if(item.id === id){

this.data.selectedCommoditys[this.data.selectedCommoditys.length] =
item;
        }
      });
  },
  fetchCheckedConmodity(){
    this.data.commodity.forEach(item =>{
        if(item.checked === true){

this.data.selectedCommoditys[this.data.selectedCommoditys.length] =
item;
        }
      });
  },
  goToStoreGoods(){
    my.switchTab({
        url: 'pages/handbag/handbag'
      })
  }
});
```

- **Field Type:**

- **Int** (integer) is suitable for calculating amounts, where you can further define its maximum, minimum, count, sum, and average. For example, you can view data on the minimum total price for the `submitAmount` field when analyzing this event.

- **String** is suitable for calculating the count of each occurrence. For example, you can view data on the number of order IDs under this event. **Int** type would not be suitable here as each order ID does not have a maximum, minimum, sum, or average amount.

- **Note**: You can describe fields in detail.

Collect Multiple Times and Report Once

If you choose this report type, which consists of multiple actions, you can see the following fields:

- `Start`: start to collect data
- `Report`: report the collected data

The last action must be to report data.

Take the "visit the home page" as an example. Data collection and reporting of the `homePageUserBehavior` event includes two actions: enter the **Home** page and click the **Popularity List** tab.

You need to configure the **Action1** with the following parameters:

- **Page**: This triggers a page, so you would need to enter the page path. You can find the page pathvia app.json files in mini program source code of IDE (Mini Program Studio). For this action, `pages/handbag/handbag` is appropriate.

copy

```
app.json:
{
  "pages": [\
    "pages/handbag/handbag",\
    "pages/shopping-cart/shopping-cart",\
    "pages/confirm-order/confirm-order",\
    "pages/my-order/my-order",\
  ]
}
```

- **Field Name**: Fields are the metrics you would like to analyze. You can define a name and assign attributes to this metric via the **Field Value** and **Field Type**.
- **Field Value**: Enter a variable for the field. For this example, developers have defined `activeTabName` as the field to calculate the count of home page visits.

copy

```
pages/handbag/handbag.js:
Page({
  data: {
```

```
    activeTabName : "All"
  },
  onShow() {
    const { searchValue = '' } = getApp();
    this.setData({ searchValue });
    this.fetchCurrentCommodities(this.data.activeTabId);
  },
  onActiveTabChange(id) {
    this.setData({ activeTabId: id,activeTabName:this.data.tabs[id
-1].name });
    this.fetchCurrentCommodities(id);
  }
});
```

- **Field Type:**
- **Int** (integer) is suitable for calculating amounts, where you can further define its maximum, minimum, count, sum, and average. For example, you can view data on the minimum total price for submitted orders when analyzing an event.
- **String** is suitable for calculating the count of each occurrence. For example, you can view data on the number of home page visits. **Int** type would not be suitable here as each home page visit does not have a maximum, minimum, sum, or average amount.
- **Note**: You can describe fields in detail.

You need to configure the **Action 2** with the following parameters:

- **Page**: This triggers a page, so you would need to enter the page path. You can find the page pathvia app.json files in mini program source code of IDE (Mini Program Studio). As **Action 2** occurs on the same page as **Action 1**, pages/handbag/handbag entered here as well.

copy

```
app.json:
{
  "pages": [\
    "pages/handbag/handbag",\
    "pages/shopping-cart/shopping-cart",\
    "pages/my/my",\
    "pages/my-order/my-order",\
    "pages/confirm-order/confirm-order"\
  ]
}
```

- **Element**: Enter a class or ID, which must begin with "`.`" or "`#`" respectively. You can find the element via app.json files in mini program source code of IDE (Mini Program Studio). For this action, enter `#TREND`, which is defined as the ID for the **Popularity List** tab.

copy

```
pages/handbag/handbag.axml:
<view
    a:for="{{tabs}}"
    class="tab-item {{activeId===item.id?'tab-item--active':''}}"
    onTap="onActiveTabChange"
    data-index="{{item.id}}"
    id="TREND">
    {{item.title}}
    <image
      a:if="{{item.sortable}}"

src="https://gw.alipayobjects.com/mdn/rms_107da2/afts/img/A*WR7tS62_iP\
      mode="scaleToFill"
    />
</view>
```

- **Field Name**: Fields are the metrics you would like to analyze. You can define a name and assign attributes to this metric via the **Field Value** and **Field Type**.
- **Field Value**: Enter a variable for the field. For this example, developers have defined `activeTabName` as the field to calculate the count of **Popularity List** tab clicks.

copy

```
pages/handbag/handbag.js:
Page({
  data: {
    activeTabName : "All"
  },
  onShow() {
    const { searchValue = '' } = getApp();
    this.setData({ searchValue });
    this.fetchCurrentCommodities(this.data.activeTabId);
  },
  onActiveTabChange(id) {
    this.setData({ activeTabId: id,activeTabName:this.data.tabs[id
-1].name });
    this.fetchCurrentCommodities(id);
  }
});
```

- **Field Type:**

- **Int** (integer) is suitable for calculating amounts, where you can further define its maximum, minimum, count, sum, and average. For example, you can view data on the minimum total price for submitted orders when analyzing an event.

- **String** is suitable for calculating the count of each occurrence. For example, you can view data on the number of tab visits. **Int** type would not be suitable here as each tab visit does not have a maximum, minimum, sum, or average amount.

- **Note**: You can describe fields in detail.

4. Save the event

Confirm all the fields and click **Save** to complete defining an event.

Next steps

Publish an event

Analyze events and funnels

Source: https://miniprogram.gcash.com/docs/miniprogram_gcash/platform/define-event

---

Developer's Guide {#developer's-guide}

Path: miniprogram_gcash

Developer's Guide

This developer guide is intended for mini program developers to get the available reference resources, such as Components and Frameworks, to quickly get started with mini program development.

Before you start, make sure you have completed the onboarding process to the Mini Program Platform. For more information, see Getting Started for Merchant On-Boarding.

Development tool (IDE)

A one-stop development tool that helps you to quickly write, deploy and debug mini programs. Learn more

Development resources

Framework includes the file structure and logic structure.

A series of basic components for developers to combine them for service development.

Additional capabilities with a set of open-source UI components.

Javascript APIs that can combine with Open APIs to provide capabilites.

Open APIs that can be combined with JSAPIs to provide capabilities.

Capabilites provided by a set of JSAPIs and Open APIs.

Related Topics

Learn the basic steps for onboarding before you start building your own Mini Programs.

Learn product features to use the Mini Program Platform.

Dip into the rich UI guidelines to design a Mini Program.

Source: https://miniprogram.gcash.com/docs/miniprogram_gcash/mpdev/developer-guide

---

Developer's guide overview {#developer's-guide-overview}

Last updated: 2021-05-09

Path: miniprogram_gcash

Developer's guide overview

2021-05-09 18:43

This developer guide is intended for mini program developers to learn how to quickly get started with mini program development.

In this guide, you can see basic concepts about the mini program technology. You can get the available reference resources, such as the Java Script APIs (JSAPIs), Components, Frameworks, and so on. You can also see how JSAPIs and Open APIs combine to be a capability. In addition, you can also quickly get started with quick-start procedures.

About mini program

Mini Program is a new technology that embeds a mobile program into the mobile app. End users can benefit from a similar experience without installing the native app. For developers, mini program provides the following benefits:

- Low learning curve as it is based on web technologies
- One code project supports both iOS and Android platform, close to the native experience
- Built-in rich components and APIs (such as access to user info, local storage, payment function, etc.)

Below is a process that illustrates how mini programs are developed by web developers to users.

Developers can use the web technology and start debugging in the Integrated Development Environment (IDE), Mini Program Studio with Domain Specific Language (DSL), such as AXML/ACSS/JS, etc. The codes are then compiled and loaded to the wallet apps. Mini programs can run on the wallet app and provide users with a cross-platform and close-to-native experience.

For more information, see About Mini Program.

Getting started

Before you start, make sure you have completed the following prerequisites:

- On-boarding to the Mini Program Platform, where you apply for an account to join a workspace. For more information, see Getting Started for Merchant On-Boarding.
- Assigned with developer roles to a mini program that is created by your admim. For more information, see Workflow Procedures of the Mini Program Platform product.
- Download the Mini Program Studio developer tool to write, deploy and debug mini programs.
- Upload the mini program from Mini Program Studio to Mini Program Platform for testing and publishing.

Refer to the Quick Start guide for detailed steps.

References

You can find resources with samples and other details to create mini programs. To be specific, you can refer to the following resources available:

- Framework
- Component
- Extended Component
- JSAPIs
- Open APIs

In addition to the basic APIs, a set of JSAPIs and Open APIs can work together to provide capabilities, for exmaple the Payment capability, to users. For more information, see Capabilities.

Next steps

Quick Start

Source: https://miniprogram.gcash.com/docs/miniprogram_gcash/mpdev-old/developer-guide

---

Document History {#document-history}

Last updated: 2021-05-10

Path: miniprogram_gcash

Document History

2021-05-10 04:19

This document provides you with information on the new or changed documentation. You can quickly see the latest documentation updates.

April, 2021

The following table describes important changes in this release.

| | | --- | --- | | **Change** | **Description** | | Using Mini Program Platform | The Product Guide is now updated with the following:
- Changed topics due to document enhancement:
- Quality
- Mini Program types
- Create Mini Programs
- Manage Mini Programs
- New topics due to document enhancement:
- How to transform an HTML 5 mobile app to an HTML 5 mini program | | Developing Mini Programs | The Developer's Guide is now updated with the following:
- Added the `PLAINTEXT_USER_LOGIN_ID` scope in the my.getAuthCode API.
- Adjusted the `USER_LOGIN_ID` scope in the my.getAuthCode API.
- Added the `customerBelongsTo` parameter for the request in /v2/authorizations/applyToken.
- Adjusted the `loginId` and `loginIdType` for LoginldInfo in the Data dictionary (for v2). |

April, 2021

The following table describes important changes in this release.

| | | --- | --- | | **Change** | **Description** | | Using Mini Program Platform | The Product Guide is now updated with the following:
- New video: Customized Analysis
- New topics: JSAPI
- Changed topics:
- Manage mini programs
- Real-time analysis |

March, 2021

| | | --- | --- | | **Change** | **Description** | | Using Mini Program Platform | The Product Guide is now updated with instructions on the mini program platform features.
The following features are new:
Customized analysis
In addition to the existing default analysis, you can perform the customized analysis by events and funnels. See Event management and analysis and Funnel management and analysis for more information. | | Developing Mini Programs | Optimize the whole OpenAPI chapter as follows:
- Add the OpenAPI Idempotency topic.
- /v1/payments/pay, /v2/payments/pay: for the `paymentRequestId` field, add the idempotency description.
- /v1/payments/refund, /v2/payments/refund: for the `refundRequestId` field, add the idempotency description.
- Add the OpenAPIs Error codes topic; and for all OpenAPIs of the v2 version, adjust the following sections:
- **Result process logic** section
- **Error codes** section
For example, see the API `/v2/authorizations/applyToken`. | | Added the *onUnhandledRejection* function in Register Mini Program.
Added the following topics:
- Added Obtain basic user information.
- Added Use TradeNO to Pay in Mini Program.
- Added Use OrderStr to Pay in Mini Program.
- Added Use PaymentUrl to Pay in Mini Program. | | Added the `SEND_MESSAGE` scope in the my.getAuthCode API.
Added the following JSAPIs:
- my.offAppHide
- my.offAppShow
- my.onAppHide
- my.onAppShow
- my.offError
- my.onError
- my.offUnhandledRejection
- my.onUnhandledRejection
- my.openDocument
Added the following extended components:
- Container
- Title
- List-item
- List-secondary
- Coupon
- Terms
- Tag
- Mask
- Guide
- Avatar
- Verify Code
- Long Password
- Multi Liner
- Button
- Am-switch

- AMRadio
- Alphabet
- Loading | | Deleted Focus property in the input component. |

Feb, 2021

| | | | --- | --- | | **Change** | **Description** | | Developing Mini Programs | Optimize the whole OpenAPI Overview chapter, mainly including the following points:
Request related changes:
- Request URL
- Request structure figure, where the red asterisk is removed from the diagram
- Request header, where the `Agent-Token` field is removed because it is not used in Mini Program.
Request related changes:
- Response structure figure, where the red asterisk is removed from the diagram
- Response header, where the `traceId` field is added. |

Jan, 2021

| | | | --- | --- | | **Change** | **Description** | | Using Mini Program Platform | The Product Guide is now updated with instructions on the mini program platform features.
The following features are new:
Two-factor authentication
You can now set the two-factor authentication to protect your account under Settings. See Two-Factor Authentication for more information.
Customized analysis
In addition to the existing default analysis, you can perform the customized analysis by events. See Manage Events for more information. | | About Mini Program | Added the Glossary
Added the FAQs | | Design Guidelines | Added Mini Program design guidelines for user permission requests. |

Dec, 2020

| | | | --- | --- | | **Change** | **Description** | | Using Mini Program Platform | **Analytics**
The dashboard is now available for you to track the data of Unique Visitors (UV) and Page Views (PV). You can see the following two pages:
- Performance
- Real-Time Analysis
Settings
Make sure you save the version so that the changes you make become effective.
Search for a mini program
You can now search for a mini program by the name or ID of the mini program. See Manage Mini Program for more information. | | Developing Mini Programs | The Developer's Guide is now updated with instructions on mini program development. Developers can view the details and checklist to quickly get started. | | The Open API

paths in the Capabilities topic is updated as Open APIs have different versions.
In addition, user address is added to the following APIs:
- `/v1/customers/user/inquiryUserInfoByAccessToken` and `/v2/users/inquiryUserInfo`, in the data type of the `userInfo` property, add `userAddresses`
- `my.getAuthCode` | | The following JSAPIs are added:
- my.getUpdateManager
- UpdateManager Overview
- UpdateManager.applyUpdate
- UpdateManager.onCheckForUpdate
- UpdateManager.onUpdateReady
- UpdateManager.onUpdateFailed
- my.compressImage
The following JSAPIs are updated:
- my.request: added the error code 2 and changed the original error code 11 to 4.
- my.getSystemInfo: modified the description of the `language` field. | | The following Open APIs are created or updated:
Updated: (The followings are Open APIs for merchants)
- `/v2/users/inquiryUserInfo` and `/v2/messages/sendInbox` : for the `authClientId` property, change from M (mandatory) to O (optional).
- `/v2/users/inquiryUserInfo` : In the `userInfo` field, add `loginIdInfos` and `contactInfos` .
- `/v2/payments/pay` : for the `appId` field, change it from O (optional) to M (mandatory) .
- `/v2/messages/sendInbox` : for the `templates` field, change its type to `Array<ContentTemplate>`
New: (The followings are new Open APIs for wallets)
- `/v2/miniprogram/qrcode/create`
- `/v2/miniprogram/serviceProxy`
- `/v2/platform/message/send` |

Nov, 2020

| | | | --- | --- | | **Change** | **Description** | | Using Mini Program Platform | A video is available for you to quickly learn the product features.
The following new features are available
Mini Program Type
- `Default (DSL)`: Native mini program
- `PWA (HTML5)`: Progressive Web App (PWA)
Settings:
- Define the Copyright Notice
- Define the email SMTP port and enable the SSL connection | | Developing Mini Programs | The following JSAPIs are updated:
- my.prompt: for Android the default value of `align` is changed from **left** to **center**
- my.uploadFile: update the description of `fileType`
The following JSAPIs are new:
Location
- my.chooseLocation
- my.openLocation
Share
- my.showSharePanel

- onShareAppMessage
The Map component is added with the following JSAPIs:
- my.createMapContext(mapId)
- MapContext.clearRoute
- MapContext.gestureEnable
- MapContext.getCenterLocation
- MapContext.moveToLocation
- MapContext.showRoute
- MapContext.showsCompass
- MapContext.updateComponents | | Developing Mini Programs | For the following types of Open APIs, `payments-core` , `payments-standard` , `oauths-standard` , `miniprogram` , and `customers` , the new version `1.2.x` of API Specifications is provided, based on which, all APIs path starts from `v2` . For example, `/v2/payments/pay` .
Except for the version upgrade, compared with the earlier version, there are lots of interface property changes. For details, see the following Open APIs section. |

Open APIs

For details of each Open API, see **Open APIs**

Nov 27, 2020

| | | | --- | --- | | **Change** | **Description** | | /v2/authorizations/revoke | - Rename `/v2/authorizations/cancelToken` to `/v2/authorizations/revoke`
- For the `authClientId` property, change its data type from String (32) to String (128) | |
/v2/authorizations/applyToken
/v2/authorizations/revoke
/v2/users/inquiryUserInfo
/v2/messages/sendInbox | For the `authClientId` property, change its data type from String (32) to String (128) | | API fundamentals | Update the **Request URL** section | | Data dictionary | Editorial changes (e.g. remove the yellow and red display） |

Nov 23, 2020

The following changes are applied for Mini Programs to align with AMS standard.

| | | | --- | --- | | **Change** | **Description** | | /v2/authorizations/applyToken | In the request body, change `code` to `authCode` | | /v2/authorizations/cancelToken | In the request body, add the `authClientId` property | | /v2/users/inquiryUserInfo | - Change the API path from `/v2/customers/user/inquiryUserInfoByAccessToken` to `/v2/users/inquiryUserInfo`
- In the Response, change the `OpenUserInfo` object name to `User` | |
/v2/messages/sendInbox | Change the API path from `/v2/customers/message/sendInboxByAccessToken` to `/v2/messages/sendInbox` | | /v2/payments/pay | In the request body:
- Move the `env` object to the `Order` object
- Delete the `merchantMCC` property | | /v2/payments/refund | In the request body, deletet the `order` object | | Data Dictionary | Change the `OpenUserInfo` object name to `User` | | In the `OpenUserInfo` object, change `birthdate` to `birthDate` | | In the `PaymentMethod`

object, change `extendMetaData` to `paymentMethodMetaData` | | In the `Order` object, change the data type of `goods` to `Array<Goods>` | | In the `Address` object: remove the duplicated `label` property |

Nov 19, 2020

- For the payments OpenAPI, the OpenAPI specification is upgraded to v1.2.4.

| | | | --- | --- | | **Change** | **Description** | | /v2/payments/pay | In the response body, change from the `ActionForm` objectto the `RedirectActionForm` object | | Data Dictionary | Add the `RedirectActionForm` object | | Remove the `env` property from the `Order` object |

Nov 17, 2020

- For the payments OpenAPI, the OpenAPI specification is upgraded to v1.2.3.

| | | | --- | --- | | **Change** | **Description** | | /v2/payments/pay | Update the description of the `productCode` property | | Update the `paymentFactor` object in the API request body as follows:
Add the following properties:
- needSurcharge
- isDeferredPayment
- needCheckCompliance
- needOtpVerification
- isCrossborderSettlement
- inStorePaymentScenario
Delete the following properties:
- isOrderCode
- isPaymentCode
- isAgreementPay | | Add the `env` and `merchantMCC` properties in the API request body | | | |

- For the authorizations OpenAPI, theOpenAPI specification is upgraded to v1.2.1.

| | | | --- | --- | | **Change** | **Description** | | /v2/authorizations/applyToken | - remove the `referenceClientId` property from the request body
- For the `authClientId` property, change from M to O | | /v2/authorizations/cancelToken | remove the `authClientId` property from the request body |

Nov 5, 2020

- For the customers OpenAPI, theOpenAPI specification is upgraded to 1.2.0

| | | | --- | --- | | **Change** | **Description** | | /v2/customers/message/sendInboxByAccessToken | Add this new API | | /v2/customers/user/inquiryUserInfoByAccessToken | - Add the authClientId property
- In the Response sample, modify the `userName` property by adding the `firstName` and `lastName` sub-properties |

- For the miniprogram OpenAPI, theOpenAPI specification is upgraded to 1.2.0

| | | | --- | --- | | **Change** | **Description** | | /v2/miniprogram/message/send | Add this new API |

- For the authorizations OpenAPI, theOpenAPI specification is upgraded to 1.2.0

| | | | --- | --- | | **Change** | **Description** | | /v2/authorizations/applyToken | - In the Sample, add the description: The access token should be kept in the merchant server only, which means that it should not be returned to the Mini Program.
- Add the authClientId property
- Rename the authCode property to Code | | /v2/authorizations/cancelToken | Add the following property:
- authClientId |

- For the payments OpenAPI, theOpenAPI specification is upgraded to 1.2.0, 1.2.1, and 1.2.2

| | | | --- | --- | | **Change** | **Description** | | /v2/payments/pay | Delete the following properties:
- partnerId
- paymentAuthCode
- paymentOrderTitle
- paymentToken
- delete the isCashierPayment field of the paymentFactor property
Add the following properties:
- salesCode
- order (replace the original `extraParams` with `order`)
Rename the following properties
- from paymentReturnUrl to paymentRedirectUrl
- from paymentMethods to paymentMethod
Other changes:
- productCode: change the description | | /v2/payments/inquiryPayment | Delete the following properties:
- paymentFailReason
- authExpiryTime | | /v2/payments/notifyPayment | Delete the following properties:
- partnerId
- paymentStatus
- paymentFailReason
Add the following properties:
- paymentResult | | /v2/payments/refund | Delete the following properties:
- partnerId
Add the following properties:
- order | | /v2/payments/inquiryRefund | Delete the following property:
- partnerId |

- For others

| | | | --- | --- | | **Change** | **Description** | | Data dictionary | - Change the `EnvInfo` object to `Env`
- Change the `birthday` property to `birthdate`
- For the `Order` object:
- Add the Merchant and env properties
- Change the description for the orderAmount property
- Modify its following "required" properties: referenceOrderId, orderDescription, orderAmount, Merchant
- For the `Result` object, change all properties from M (mandatory) to O (optional) | | API

Fundamental | - Add more description for the **Request URL**
- In the **Response header** section, add details about the response structure | | Call an API | Add a new chapter, where signing a request and validating the response signature are documented |

Oct, 2020

The following table describes important changes in each release since Oct, 2020.

| | | --- | --- | | **Change** | **Description** | | Getting Started | Getting Started for Merchant On-Boarding is now available for merchants to quickly create a developer workspace. | | Using Min Program Platform | The following new features are available:
- QR Code for Testing
- Whitelist for Gray Box Testing
In addition, the product user guide is updated for you to easily use the Mini Program Platform. The following new topics are added for you to see the detailed features:
- Manage Mini Programs
- Manage Workspace
- Authorization
- Approvals
- Manage Apps
- Manage Feedback
- Members
- Features
For more information, see these guides under Using Mini Program Platforms. | | Developing Mini Programs | A new API my.chooseFileFromDisk is now available for you to choose a file to upload. You can also view the details of the file or delete the file you have added.
The Appx is now updated to 1.24.6 version. You need to make sure you use the Appx with 1.24.6 or higher versions in order to use the following APIs:
- my.getSiteInfo
- my.signContract
The my.request API is updated with error code descriptions when the server or H5 domain whitelist is not configured.
For more information, see these API references under `Developing Mini Program > References > JSAPI`.
The Capabilities description is now available for you to see the API capabilities:
- User Information Capability
- Payment Capability
For more information, see `Developing Mini Program > Working with Mini Program > Capabilities`. | | User Experience Design Guidelines | The design guideline for food and beverage industry is now available. You can use the guidelines to design the industry-specific mini programs. |

Sept, 2020

The following table describes important changes in each release since Sept, 2020.

| | | --- | --- | | **Change** | **Description** | | Getting Started | The new topic is introduced to help you quickly start with the Mini Program Platform. | | App Container | A feature description of App Container is now added. | | Structural Changes | The structure of documentation is updated. You can quickly get the information that you need with an organized structure below:
- *About Mini Program:* See the introduction.
- *Getting Started:* Quickly getting started with
- *Developing Min Program: G* et the developer guide and references, such as JSAPIs and Open APIs.
- Using *Min Program Platform:* See the user manual of the platform product.
- *User Experience Design Guidelines:* Check the design details and guidelines.
- *Document History*: See what's new in the latest release and the document history. | | Using *Min Program Platform* | The product user guide is updated for you to easily use the Mini Program Platform. The following new topics are added for you to see the detailed features:
- *Manage Mini Programs*
- *Manage workspace*
- *Authorization*
- *Approvals*
- *Manage Apps* | | Open API | The Open API reference `/v1/payments/pay` is updated with a request parameter `appId`. You can get the mini program ID, based on which you can conduct marketing campaigns. |

Aug, 2020

The following table describes important changes in each release since Aug, 2020.

| | | --- | --- | | **Change** | **Description** | | Mini Program Studio | Editorial changes to the developer tool. | | Mini Program Development Platform | The platform product guide is now available with the following features:
- Overview topic
- Member roles description and authorizations for each role
- Workflows and detailed explanation on the procedures
- Quality for each mini program in the workspace
- Analytics to see a visualized form of data
See the latest guides under ***Getting Started => Working with Mini Program Platform***.
The **settings** feature is now available for workspace admins on the platform. Check the detailed description in the product guide. | | API References | The movable-view and movable-area View Containers are added to the Components.
The following APIs are added:
- my.hideBackHome
- Tab bar FAQ
- my.watchShake
- my.onAccelerometerChange
- my.offAccelerometerChange
- my.onCompassChange
- my.offCompassChange
- my.onMemoryWarning
- my.offMemoryWarning

- my.getPhoneNumber
See the latest references under ***Getting Started => Working with Mini Program => References.*** |

Source: https://miniprogram.gcash.com/docs/miniprogram_gcash/history/last-release-note

---

Documentation {#documentation}

Documentation

The Mini Program technology is the answer to rapid mobile app development. Mini Programs are sub-applications that run inside the mobile app. You can access various services and features on Mini Programs without the need to install additional applications. Discover guides, tutorials and references to quickly get started.

Quick Start

Learn how to release your first Mini Program:

- Apply for an account in the Mini Program Platform
- Create a workspace and assign roles for admins (tenants) and developers
- Create a Mini Program in the workspace
- Build the Mini Program in the developer console
- Submit the Mini Program for approval

Get Started

What's New

See the latest release note here

- What's New

Platform Product Guide

Learn product features to use the Mini Program Platform product

- Using Mini Program Platform
- Manage Mini Programs

Developer Guide

Get started to use the Mini Program Studio and other references to develop a Mini Program

- Working with Mini Programs
- Mini Program Studio
- About App Container

API References

Bookmark the API reference docs for the Mini Program development

- Java Script APIs
- Open APIs
- Framework

UX Design Guidelines

Dip into the rich UI component available in Mini Program development

- Navigation
- Feedback
- Interface

Source: https://miniprogram.gcash.com/docs/

Error codes {#error-codes}

Last updated: 2021-05-09

Path: miniprogram_gcash

Error codes

2021-05-09 18:43

If an error occurs when you call an API, an error response is returned, where the `result` object indicates the error code (`resultCode`) and error message (`resultMessage`). You can use error codes and messages to troublshoot issues.

Error codes are usually classified into the following categories:

- Common error codes: are common for all Mini Program OpenAPIs.
- API-specific error codes: are dedicated to a specific OpenAPI.

Common error codes

The following table lists all common error codes for Mini Program OpenAPIs. If you do not find an error code in the following table, it means that the error code is not common, but dedicated to a specific OpenAPI (see the following API-specific error codes section).

| | | | | --- | --- | --- | --- | | **resultCode** | **resultMessage** | **resultStatus** | **Action to do** | | PROCESS_FAIL | A general business failure occurred. Do not retry. | F | Human intervention is usually needed. It is recommendeded that you contact Mini Program Technical Support to troubleshoot the issue. | | PARAM_ILLEGAL | Illegal parameters.

For example, non-numeric input, invalid date. | F | Check and verify whether the request fields (including the header fields and body fields) of the current API are correct and valid.
For details, see the specific API specification, for example, the `applyToken` API specification. | | INVALID_API | The called API is invalid or not active. | F | Check whether the current API name is used by mistakes when the API is called. | | ACCESS_DENIED | Access is denied | F | Need to check the `resultMessage` of the current API specification for details. For example, the `applyToken` API specification. | | REQUEST_TRAFFIC_EXCEED_LIMIT | The request traffic exceeds the limit. | F | - The party that calls APIs needs to reduce the API calling frequency, or
- The API service provider needs to increase the traffic limit or threshold. | | EXPIRED_ACCESS_TOKEN | The access token is expired. | F | Renew the access token by calling the `applyToken` API (`/v1/authorizations/applyToken` or `/v2/authorizations/applyToken`) | | UNKNOWN_EXCEPTION | An API calling is failed, which is caused by unknown reasons. | U | Trying to re-call the API might help to resolve the issue. |

API-specific error codes

For error codes that are dedicated to a specific API, see the **Result** section in each API specification, for example , the `applyToken` API specification.

九色鹿

Source: https://miniprogram.gcash.com/docs/miniprogram_gcash/mpdev-old/error_codes

---

Error codes {#error-codes}

Last updated: 2022-07-07

Path: miniprogram_gcash

Error codes

2022-07-07 17:08

If an error occurs when you call an API, an error response is returned, where the object `result` indicates the error code (`resultCode`) and error message (`resultMessage`). You can use error codes and messages to troublshoot issues.

Error codes are usually classified into the following categories:

- Common error codes: are common for all Mini Program OpenAPIs.
- API-specific error codes: are dedicated to a specific OpenAPI.