GOV.UK Wallet tech docs accessibility statement

Accessibility statement for GOV.UK Wallet technical documentation

This accessibility statement applies to the GOV.UK Wallet technical documentation at https://docs.wallet.service.gov.uk/ (https://docs.wallet.service.gov.uk/).

This website is run by the GOV.UK One Login team at the Government Digital Service (GDS). We want as many people as possible to be able to use this website. For example, that means you should be able to:

- change colours, contrast levels and fonts
- zoom in up to 300% without problems
- navigate most of the website using just a keyboard
- navigate most of the website using speech recognition software
- listen to most of the website using a screen reader (including the most recent versions of JAWS, NVDA and VoiceOver)

We’ve also made the website text as simple as possible to understand.

AbilityNet (https://abilitynet.org.uk/) has advice on making your device easier to use if you have a disability.

How accessible this website is

This website is partially compliant with the Web Content Accessibility Guidelines version 2.2 AA standard.

What to do if you cannot access parts of this website

If you need information on this website in a different format like accessible PDF, large print, easy read, audio recording or braille, use the Contact Us page to contact the

GOV.UK Wallet team (/contact-us.html) with details of your request.

We'll aim to reply in 3 working days.

Reporting accessibility problems with this website

We're always looking to improve the accessibility of this website. If you find any problems not listed on this page or think we're not meeting accessibility requirements, use the Contact Us page to contact the GOV.UK Wallet team (/contact-us.html).

Enforcement procedure

The Equality and Human Rights Commission (EHRC) is responsible for enforcing the Public Sector Bodies (Websites and Mobile Applications) (No. 2) Accessibility Regulations 2018 (the 'accessibility regulations'). If you're not happy with how we respond to your complaint, contact the Equality Advisory and Support Service (EASS) (https://www.equalityadvisoryservice.com/).

Technical information about this website's accessibility

GDS is committed to making its website accessible, in accordance with the Public Sector Bodies (Websites and Mobile Applications) (No. 2) Accessibility Regulations 2018.

Compliance status

This website is partially compliant with the Web Content Accessibility Guidelines version 2.2 AA standard.

Preparation of this accessibility statement

This statement was prepared on 06 May 2025.

This website was last tested in May 2025.

This page was last reviewed on 6 May 2025. It needs to be reviewed again on 6 November 2025 .

Before you issue a credential

Before you can issue a credential, you should do the following.

Onboard with GOV.UK One Login

GOV.UK Wallet uses GOV.UK One Login to authenticate users. This process makes sure that credentials are issued into a wallet that is logged in as the same user the credential is for.

You must complete the onboarding process for GOV.UK One Login before you can issue credentials to GOV.UK Wallet. There is guidance on the GOV.UK One Login onboarding process in their technical documentation (https://docs.sign-in.service.gov.uk/).

When you complete the GOV.UK One Login onboarding process, make sure that you have:

- your unique client identifier - this identifier must be included as a claim (`client_id`) in the pre-authorised code your service generates as part of issuing a GOV.UK Wallet credential offer
- requested that the `walletSubjectId` custom claim is activated for your GOV.UK One Login client - this is a pairwise identifier that will be used to prove that the user logged in to your service and GOV.UK Wallet are the same user

Agree a credential template

Before you can issue a credential, you must confirm with the GOV.UK Wallet onboarding team:

- the credential type - this can be mDoc (https://www.iso.org/obp/ui/en/#iso:std:iso-iec:18013:-5:ed-1:v1:en) or JWT VC (https://datatracker.ietf.org/doc/draft-ietf-oauth-sd-jwt-vc/)
- which attributes are to be included in the digital credential and its schema
- the colour of the digital card
- whether users can have multiple instances of this credential, or only one at a time

To integrate with GOV.UK Wallet you must send the GOV.UK Wallet team:

- your issuer URL (both integration and production)
- your issuer logo in English and Welsh

Follow relevant standards

When you are preparing to issue a credential to GOV.UK Wallet you must align with open standards.

GOV.UK Wallet will support multiple credential formats to represent government documents. These documents can be:

- mdoc based credentials for the digital driving licence (https://www.iso.org/obp/ui/en/#iso:std:iso-iec:18013:-5:ed-1:v1:en)
- other Verifiable Credentials (VCs), including W3C Verifiable Credential Data Model 2.0 (https://www.w3.org/TR/vc-data-model-2.0/) and later other formats allowing selective disclosure of attributes (https://datatracker.ietf.org/doc/draft-ietf-oauth-sd-jwt-vc/)

GOV.UK Wallet supports OpenID Connect for Verifiable Credential Issuance (OIDC4VCI) (https://openid.net/specs/openid-4-verifiable-credential-issuance-1_0.html) for its issuance flow.

You should also:

- follow the government Service Standard (https://www.gov.uk/service-manual/service-standard), for example making all information available in Welsh
- use findings identified by your user research to meet your users' needs

To get started, you need to authenticate users with One Login.

This page was set to be reviewed before 15 October 2025. This might mean the content is out of date.

Credentials in GOV.UK Wallet

Credentials are issued by the relevant government department into GOV.UK Wallet. The government department defines which attributes to include by using the credential schema. They are also responsible for the information in the credentials they issue.

Credentials and their attributes will be made available in this technical documentation as they are released.

Currently available credentials

Veteran card

Attribute	Definition	Example

Name	Given name(s) and family names as they appear on the veteran card	
		```
		"name": [
		    {
		      "nameParts": [
		        {
		          "value": "Sarah",
		          "type": "GivenName"
		        }
		        {
		          "value": "Elizabeth",
		          "type": "GivenName"
		        },
		        {
		          "value": "Edwards",
		          "type": "FamilyName"
		        }
		      ]
		    }
		  ]
		```

Date of birth	Day, month and year of birth as it appears on the veteran card	`"birthDate": [ { "value": "1985-10-18" } ]`
Photo	Photo of the cardholder	`"photo": "[PHOTO]"`
Service number	Identification code of the veteran card holder	`"serviceNumber": "25057386"`
Service branch	Branch of the British Armed Forces where the cardholder served	`"serviceBranch": "British Army"`
Expiry date	Expiry date of the veteran card	`"expiryDate": "2034-04-08"`
Credential expiry date	Expiry date of the digital veteran card credential	`"exp": 1778664693` (13 May 2026 09:31:33)

This page was last reviewed on 7 May 2025. It needs to be reviewed again on 7 November 2025 .

Supported formats and protocols

GOV.UK Wallet will support multiple credential formats to represent government documents. These documents can be:

- mdoc based credentials for the digital driving licence (https://www.iso.org/obp/ui/en/#iso:std:iso-iec:18013:-5:ed-1:v1:en)
- other Verifiable Credentials (VCs), including W3C Verifiable Credential Data Model 2.0 (https://www.w3.org/TR/vc-data-model-2.0/) and other formats allowing selective disclosure of attributes (https://datatracker.ietf.org/doc/draft-ietf-oauth-sd-jwt-vc/)

It will also support multiple protocols for users to share credentials and attributes:

- OpenID for Verifiable Presentation (OID4VP) (https://openid.net/specs/openid-4-verifiable-presentations-1_0.html) for online sharing, using remote flows to verify mdoc and VC based credentials
- ISO/IEC 18013-5 (https://www.iso.org/obp/ui/en/#iso:std:iso-iec:18013:-5:ed-1:v1:en) for in-person interactions, using proximity flows to verify mdoc based credentials

GOV.UK Wallet will first enable supervised proximity flow for verification in line with ISO/IEC 18013-5. Remote flows, and more details about OID4VP profiles, will be added in the future.

This page was last reviewed on 7 May 2025. It needs to be reviewed again on 7 November 2025 .

Consuming and verifying credentials in GOV.UK Wallet

GOV.UK Wallet will allow GOV.UK One Login users to store and present digital versions of government-issued documents on their phones.

Government departments will issue cryptographically verifiable credentials to a user's GOV.UK Wallet (/issuing-credentials-to-wallet.html). The user's credentials are linked to their GOV.UK One Login account and to their personal device, and cannot be moved to another device.

GOV.UK Wallet will enable:

- government departments and certain other public sector organisations to consume and verify credentials or attributes
- organisations outside of government to use verified information passed to them by a digital verification service (DVS), certified under the UK digital identity and attributes trust framework (https://www.gov.uk/government/collections/uk-digital-identity-and-attributes-trust-framework) and on the digital identity and attribute services register (https://www.gov.uk/guidance/find-registered-digital-identity-and-attribute-services) (DVS register)

This technical documentation will be updated as new information and features are available. We welcome feedback from partners and industry on our documentation - find out how to contact us (/contact-us.html).

Sharing credentials using GOV.UK Wallet

GOV.UK Wallet will let users share their credentials:

- in-person, for example to prove their age when purchasing age-restricted products
- online, to share documents with a service securely instead of uploading a photo or a PDF

GOV.UK Wallet will use standard protocols to offer flexible verification scenarios (/consuming-and-verifying-credentials/supported-protocols.html).

Trusted list

GOV.UK Wallet will put mechanisms in place to make sure personal data from users is shared only with trusted parties. This will mitigate the risk of malicious apps or services accessing credential data without the user's knowledge.

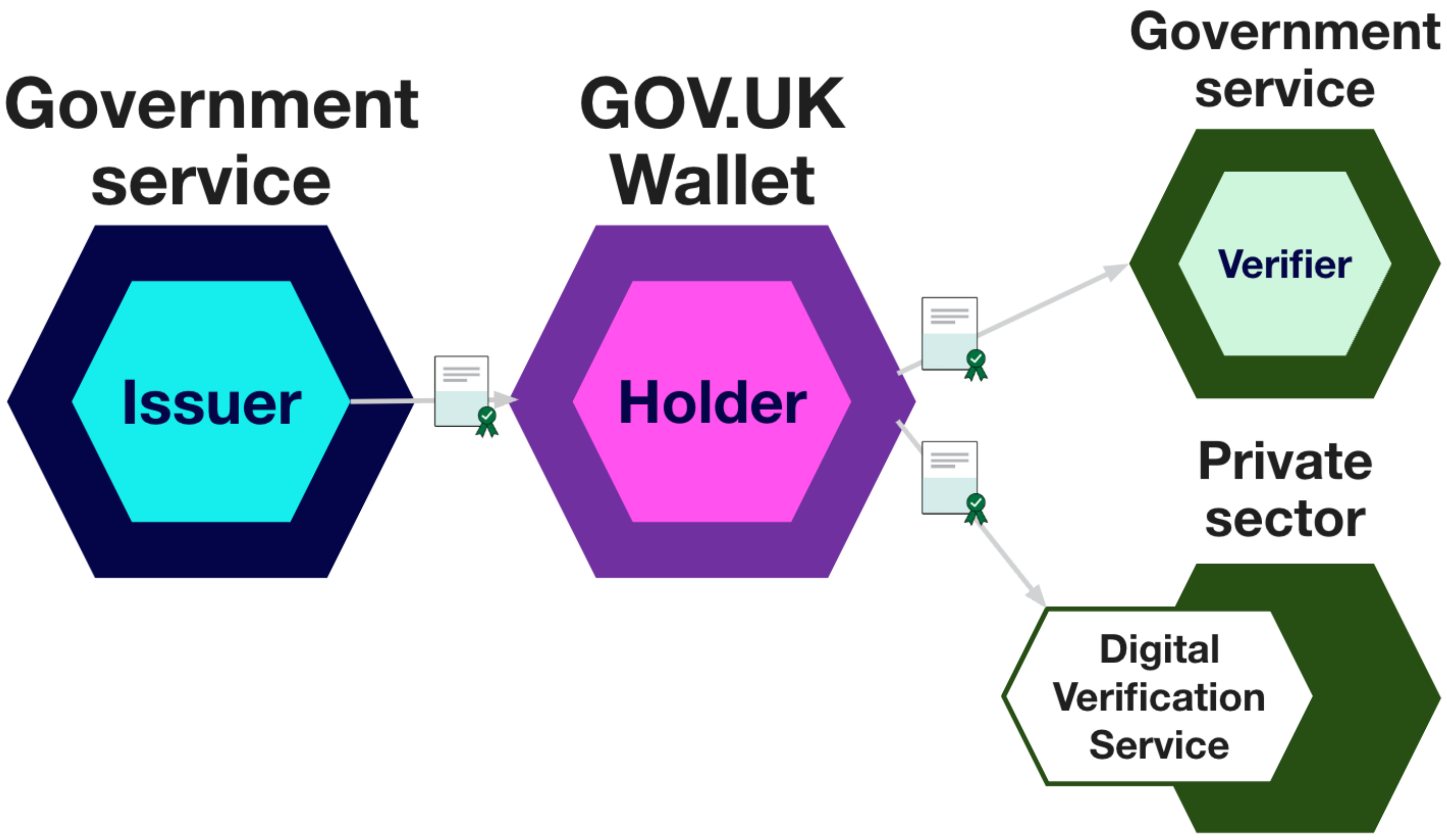
GOV.UK Wallet will use trusted lists to identify consumers of credentials. Where data is shared with parties outside of government, GOV.UK Wallet will only release credentials and attributes to a digital verification service (DVS) which is certified against the trust framework and appears on the DVS register.

Further details on this functionality will be added in future.

Data flows

Data flow between credential issuers, holders and verifiers

GOV.UK Wallet is built in three parts to connect government departments (credential issuers), users (credential holders) and verifiers requesting data (credential verifiers).



1. Government department issuers

Government departments (issuers) issue digital and verifiable versions of physical documents (credentials) to a user's GOV.UK Wallet. For example, a government department could issue a credential containing a user's date of birth that proves their age.

2. GOV.UK Wallet

The credential's rightful holder (the user the credential refers to) uses GOV.UK Wallet to store, manage and present their credentials online and in person. For example, a user could store a credential containing their date of birth, and present information from it when they need to prove their age.

3. Verifier services

Government departments and certain public sector organisations will be able to verify and use credentials and attributes held in GOV.UK Wallet.

Outside of government, a DVS certified against the trust framework and added to the DVS register will be able to access GOV.UK Wallet and verify information it holds at a user's request.

For example, a business (known as a relying party) selling age-restricted products could use a certified and registered DVS to request and digitally verify a customer's age based on attributes held in credentials in their GOV.UK Wallet.

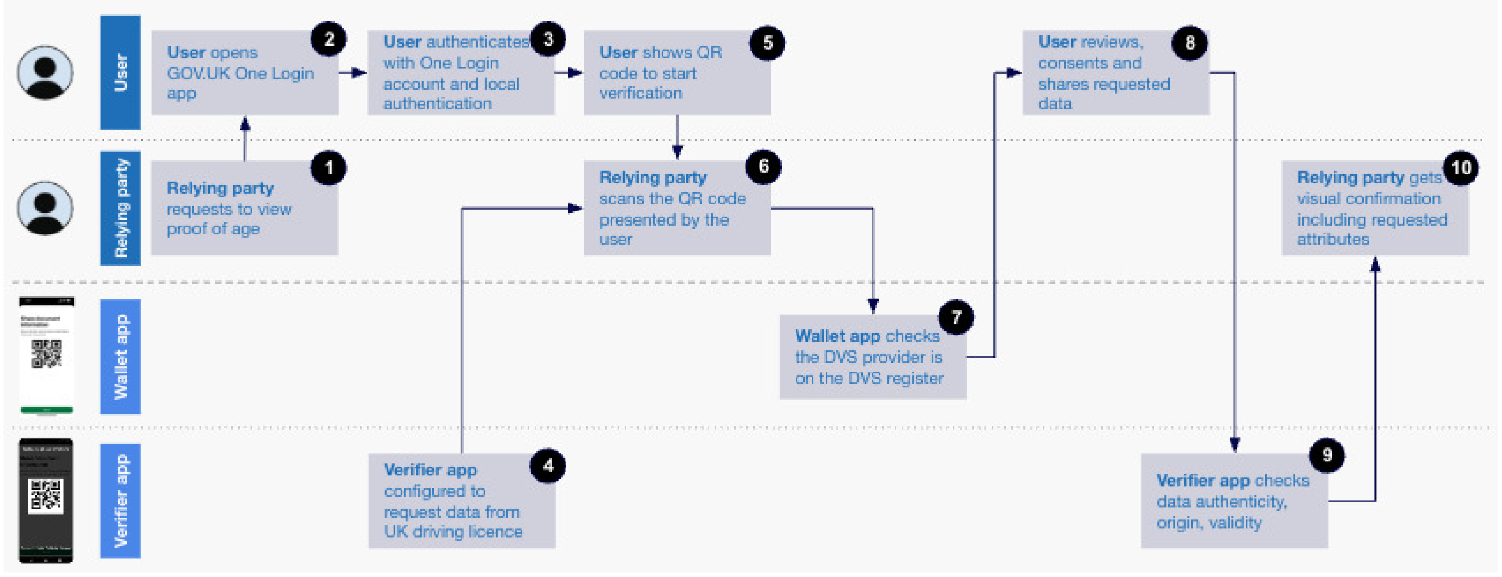
Using GOV.UK Wallet in person in the private sector

The three example flows below illustrate how GOV.UK Wallet, via a registered DVS provider, can be used to purchase age-restricted products from a private sector business. All three assume the DVS is providing an orchestration service, but other models will also exist. There is guidance on the models available for DVS providers (https://gov.uk/guidance/using-govuk-wallet-in-the-digital-identity-sector).

ISO 18013-5 supervised proximity flow

In this example, a user is purchasing age-restricted products in person and sharing their information with a business. They have a valid digital driving licence stored on their personal device in GOV.UK Wallet.

The business selling the products (the relying party) uses a device with a verification service provided by a suitable DVS to verify the user's age. In this example, the verification service is provided via a verifier app (it could, for example, also be a point of sale terminal, QR scanner terminal etc.). To work with GOV.UK Wallet, the DVS must be certified against the trust framework and appear on the DVS register.


User
Relying party
Wallet app
Verifier app
1 Relying party requests to view proof of age
2 User opens GOV.UK One Login app
3 User authenticates with One Login account and local authentication
4 Verifier app configured to request data from UK driving licence
5 User shows QR code to start verification
6 Relying party scans the QR code presented by the user
7 Wallet app checks the DVS provider is on the DVS register
8 User reviews, consents and shares requested data
9 Verifier app checks data authenticity, origin, validity
10 Relying party gets visual confirmation including requested attributes

The data flow for this interaction is as follows.

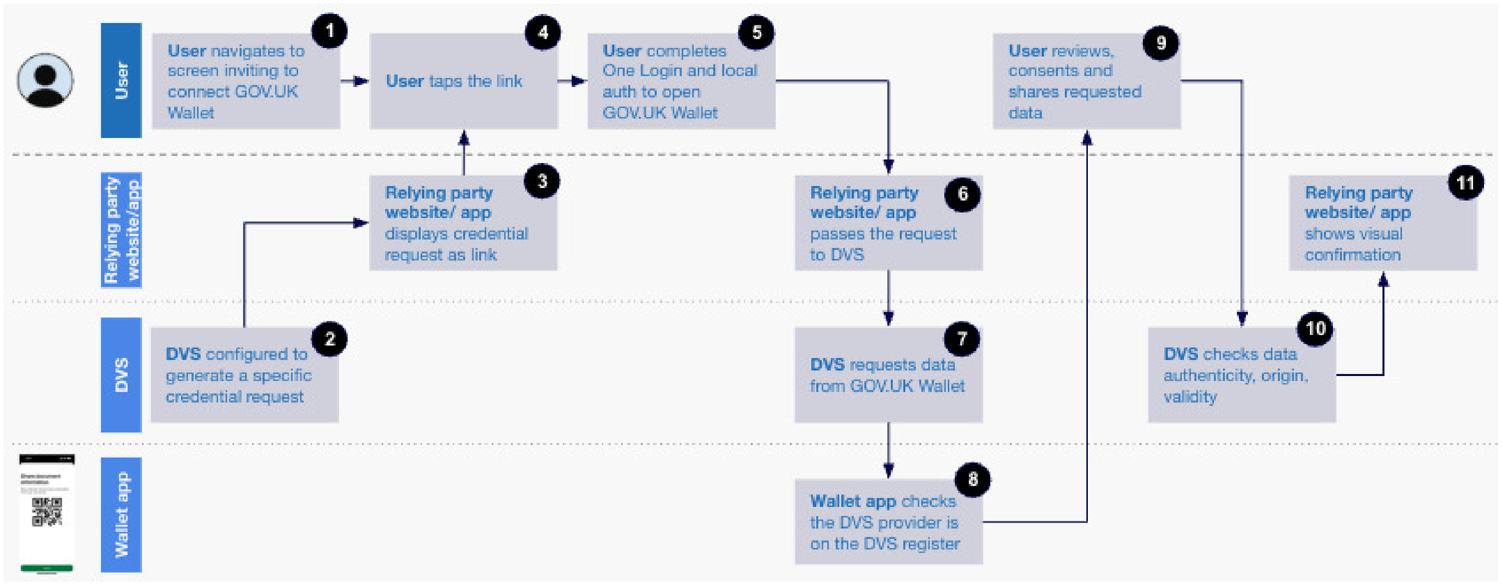
1. The relying party asks the user to show proof of their age.
2. The user who needs to prove their age opens the GOV.UK One Login app.
3. To open GOV.UK Wallet, the user authenticates themselves with GOV.UK One Login and uses the device's local authentication (face, fingerprint, PIN or pattern).
4. The verifier app on the relying party's device is configured to request data from the user's digital driving licence. For this transaction, the data requested is a proof of age.
5. GOV.UK Wallet generates a QR code on the user's device, which the user shows to the relying party to begin the verification process.
6. The relying party scans the QR code using the verifier app.
7. GOV.UK Wallet checks that the verifier app is using a trust framework certified and DVS-registered provider.
8. The user reviews the data that was requested (for example an 'over 18' attribute), consents to share it, and allows it to be shared with the verifier app.
9. The verifier app checks the data's authenticity, origin and validity.
10. The verifier app shows the relying party a visual confirmation of the user's proof of age.

Using GOV.UK Wallet online in the private sector

OID4VP same device flow

In this example, a user is purchasing an age-restricted product online using an app or the browser on their phone. This is the same phone where their credentials are held in GOV.UK Wallet. The user holds a credential that would prove their age (for example, a digital driving licence) in GOV.UK Wallet.

The website selling the product (the relying party) must get proof of the user's age before completing the transaction. To work with GOV.UK Wallet, the relying party website must use a registered DVS certified against the trust framework.



User
Relying party website/app
DVS
Wallet app
1 User navigates to screen inviting to connect GOV.UK Wallet
2 DVS configured to generate a specific credential request
3 Relying party website/ app displays credential request as link
4 User taps the link
5 User completes One Login and local auth to open GOV.UK Wallet
6 Relying party website/ app passes the request to DVS
7 DVS requests data from GOV.UK Wallet
8 Wallet app checks the DVS provider is on the DVS register
9 User reviews, consents and shares requested data
10 DVS checks data authenticity, origin, validity
11 Relying party website/ app shows visual confirmation

The data flow for this interaction is as follows.

1. During their purchase, the relying party directs the user to a DVS to prove their age. The user chooses to connect this DVS to their GOV.UK Wallet.

2. The DVS used by the relying party's website is configured to generate a specific credential request. In this example, the credential request would include confirmation that the user is above the age needed to complete the transaction.

3. The relying party's website or embedded DVS displays the credential request to the user as a link.

4. The user taps the link, which opens the GOV.UK One Login app.

5. To open GOV.UK Wallet, the user authenticates themselves with GOV.UK One Login and uses the device's local authentication (face, fingerprint, PIN or pattern).

6. The relying party passes the request to the DVS.

7. The DVS requests the data it needs from GOV.UK Wallet. In this example, the data would be a confirmation that the user is above the age needed to complete the transaction.

8. GOV.UK Wallet checks that the DVS provider is on the DVS register.

9. The user reviews the data that was requested, consents to share it, and allows it to be shared with the DVS and the relying party.

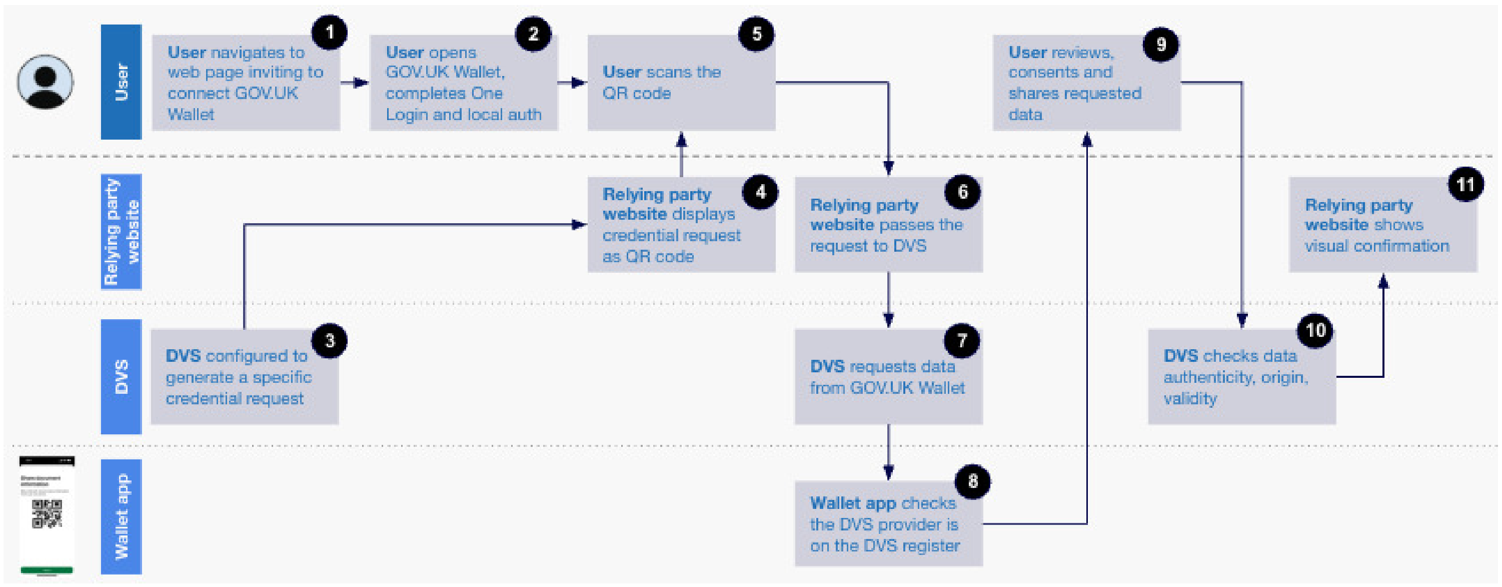
10. The DVS checks the data's authenticity, origin and validity, and passes it to the relying party.

11. The relying party website shows a visual confirmation of the credential verification. If the verification was successful and the user has proven their age, they can continue with the transaction.

OID4VP cross device flow

In this example, the user holds a credential that would prove their age (for example, a digital driving licence) in GOV.UK Wallet on their phone. The user is purchasing an age-restricted product online using a separate device (for example, a laptop or tablet).

The website selling the product (the relying party) must get proof of the user's age before completing the transaction. The relying party is using a registered DVS certified against the trust framework.


User
Relying party website
DVS
Wallet app
1 User navigates to web page inviting to connect GOV.UK Wallet
2 User opens GOV.UK Wallet, completes One Login and local auth
3 DVS configured to generate a specific credential request
4 Relying party website displays credential request as QR code
5 User scans the QR code
6 Relying party website passes the request to DVS
7 DVS requests data from GOV.UK Wallet
8 Wallet app checks the DVS provider is on the DVS register
9 User reviews, consents and shares requested data
10 DVS checks data authenticity, origin, validity
11 Relying party website shows visual confirmation

The data flow for this interaction is as follows.

1. During their purchase on their laptop or tablet, the relying party directs the user to a DVS to prove their age. The user chooses to connect this DVS to their GOV.UK Wallet app on their phone.

2. To open GOV.UK Wallet on their phone, the user authenticates themselves with GOV.UK One Login and uses the device's local authentication (face, fingerprint, PIN or pattern).

3. The DVS used by the relying party is configured to generate a specific credential request. In this example, the request would include a confirmation that the user is above the age needed to complete the transaction.

4. The relying party displays the credential request to the user as a QR code.

5. The user scans the QR code using GOV.UK Wallet on their phone.

6. The relying party passes the request to the DVS.

7. The DVS requests the data it needs from GOV.UK Wallet. In this example, the data would be a confirmation that the user is above the age needed to complete the transaction.
8. GOV.UK Wallet checks that the DVS provider appears on the DVS register.
9. The user reviews the data that was requested, consents to share it, and allows it to be shared with the DVS and the relying party.
10. The DVS checks the data's authenticity, origin and validity, and passes it to the relying party
11. The relying party shows a visual confirmation of the credential verification. If verification was successful and the user has proven their age, they can continue with their purchase.

This page was last reviewed on 7 May 2025. It needs to be reviewed again on 7 November 2025 .

Contact us

For government and the public sector

If you have feedback or questions, contact us via the #govuk-wallet Slack channel (https://ukgovernmentdigital.slack.com/archives/C08A9JMDK0Q).

For a digital verification service

If you are a digital verification service and have questions about using GOV.UK Wallet, or this documentation, email govukwallet-queries@digital.cabinet-office.gov.uk.

This page was last reviewed on 9 May 2025. It needs to be reviewed again on 9 November 2025 .

Authenticating users with One Login

Services that wish to issue credentials must use GOV.UK One Login to authenticate their users. This process makes sure that credentials are issued into a wallet that is logged in as the same user the credential is for.

When you register your service with GOV.UK One Login, you get a unique client identifier. This identifier must be included as a claim (`client_id`) in the pre-authorised code your service generates as part of issuing a credential offer. There is more guidance on issuing a credential offer (/credential-issuer-functionality/credential-offer).

When your user authenticates with GOV.UK One Login, you obtain their user information, which includes their GOV.UK Wallet subject identifier (`walletSubjectId`). This subject identifier is a pairwise identifier you can use at the point where you finally issue the digital credential to assure that the user logged in to your service and GOV.UK Wallet are the same user. This is referred to as the ‘rightful holder check’.

This page was set to be reviewed before 5 September 2025. This might mean the content is out of date.

API

/credential

post

An endpoint used by the GOV.UK Wallet to request a Credential from the Credential Issuer.

Responses

Status	Description
200	Credential.
400	Bad Request
401	Unauthorized

200 Credential.

```
{
  "credentials": [
    {
      "credential": "eyJraWQiOiJkaWQ6d2ViOmV4YW1wbGUtY3JlZGVudGlhbC1pc3N1ZXIuZ292LnVrIzVkY2JlZTg2M2I1ZDdjYzMwYzliYT
    }
  ],
  "notification_id": "776aefd4-26c6-4a5f-aa7c-b5e294cd87cd"
}
```

400 Bad Request

```
{
  "error": "invalid_proof"
}
```

Schemas

CredentialResponse

Name	Type	Required	Description	Schema
credentials	array	false	An array containing one issued credential.	Credential
notification_id	string	false	Issuance flow notification ID.	

Credential

Name	Type	Required	Description	Schema
credential	string	true	The issued credential.	

Credential400ErrorResponse

Name	Type	Required	Description	Schema
error	string	false	An error code, such as `invalid_proof` or `invalid_nonce` .	

This page was last reviewed on 19 June 2025. It needs to be reviewed again on 19 December 2025 .

API

/credential_offer

get

An endpoint for issuing a credential offer.

Responses

Status	Description
200	Credential offer URL consisting of the following parts: 1. Wallet's credential offer endpoint. 2. A query parameter `credential_offer` . 3. URL-encoded credential offer. `"https://mobile.account.gov.uk/wallet/add?credential_offer=%7B%22credential_configuration_ids%22%3A%5B%22FishingLic`

This page was set to be reviewed before 24 September 2025. This might mean the content is out of date.

Credential Offer

A credential offer is used to pass information relevant for credential issuance to GOV.UK Wallet, including a unique credential identifier so the credential can be identified later. Your service must generate a credential offer to begin the credential issuance process. This credential offer is passed to GOV.UK Wallet after an authenticated user gives their consent.

You must build your credential offer using the full `credential_offer` object embedded in a URI. GOV.UK Wallet does not support using the `credential_offer_uri` parameter to reference a JSON object containing credential offer parameters. There is more guidance in the OID4VCI specification (https://openid.net/specs/openid-4-verifiable-credential-issuance-1_0.html#name-credential-offer).

Technical details

The credential offer object

The credential offer is a JSON object containing the following parameters:

Parameter	Description
`credential_configuration_ids`	An array of strings, where each item is a type of credential that can be obtained from the credential issuer.
`grants`	An object that indicates to GOV.UK Wallet the grant types that the credential issuer's authorisation server is prepared to process for this credential offer. Currently, the only supported grant type is grants.urn:ietf:params:oauth:grant-type:pre-authorized_code.
`grants.urn:ietf:params:oauth:grant-type:pre-authorized_code`	The grant type required for the pre-authorised code flow.
`grants.urn:ietf:params:oauth:grant-`	The pre-authorised code generated and signed by the credential issuer and which gives GOV.UK Wallet authorisation

`type:pre-authorized_code.pre-authorized_code`	to obtain an access token from the authorisation server.
`credential_issuer`	The URL of the credential issuer. This is used later by GOV.UK Wallet to fetch the credential.

This is an example of a credential offer JSON object. The `pre-authorized_code` in the example is decoded below it.

```
{
  "credential_configuration_ids": [
    "FishingLicenceCredential"
  ],
  "grants": {
    "urn:ietf:params:oauth:grant-type:pre-authorized_code": {
      "pre-authorized_code": "eyJraWQiOiI1ZGNiZWU4NjNiNWQ3Y2MzMGM5YmExZjczOT
    }
  },
  "credential_issuer": "https://example-credential-issuer.gov.uk"
}
```

The pre-authorised code

When the wallet requests a credential, GOV.UK Wallet offers them an access token signed by GOV.UK One Login. The credential issuer can validate that token to confirm the request came from a genuine GOV.UK Wallet instance, and get assured confirmation of the logged in user's walletSubjectId.

One of the methods defined in the OID4VCI specification for issuing access tokens is the pre-authorized code flow (https://openid.net/specs/openid-4-verifiable-credential-issuance-1_0.html#section-3.5). This is the only method accepted by GOV.UK Wallet.

The pre-authorised code flow makes getting credentials simpler by letting the issuer start the authorisation flow. In a standard authorisation flow, GOV.UK Wallet would need to start the flow. In the pre-authorised code flow, the credential issuer starts the authorisation flow and passes the details to GOV.UK Wallet (via the pre-authorised code) so that GOV.UK Wallet can continue.

The pre-authorised code is a JWT generated and signed by your credential issuer and included in the credential offer.

JWT Header

The JWT header must contain the following parameters:

```
{
  "kid": "5dcbee863b5d7cc30c9ba1f7393dacc6c16610782e4b6a191f94a7e8b1e1510f",
  "typ": "JWT",
  "alg": "ES256"
}
```

- `kid` matches a `kid` in the JSON Web Key Set (JWKS) published to your [`/.well-known/jwks.json` (/credential-issuer-functionality/jwks)](/credential-issuer-functionality/jwks) endpoint
- `typ` must be “JWT” - this is the media type of the complete JWT
- `alg` must be “ES256” - this is the algorithm used to sign the JWT

JWT Payload

The JWT payload must contain the following claims:

```
{
  "clientId": "<YOUR ONE LOGIN CLIENT ID>",
  "credential_identifiers": [
    "<CREDENTIAL IDENTIFIER>"
  ],
  "exp": 1234567890,
  "iat": 1234567890,
  "iss": "https://example-credential-issuer.gov.uk",
  "aud": "https://token.account.gov.uk"
}
```

- `clientId` is your client ID which you received when you registered your service to use GOV.UK One Login
- `credential_identifiers` is the unique identifier for the specific credential offer - currently, GOV.UK Wallet only supports one identifier per pre-authorised code. This should be a long random value (for example, a UUIDv4) and should not be a personal identifier or account identifier.
- `exp` is the expiration date of the pre-authorised code - we recommend this is aligned with the length of time a user can remain inactive on your service, up to a maximum of 1

hour. Must be expressed in epoch time as per the IETF RFC 7519 (https://datatracker.ietf.org/doc/html/rfc7519)

- `iat` is the time at which the pre-authorized code was issued. Must be expressed in epoch time as per the IETF RFC 7519 (https://datatracker.ietf.org/doc/html/rfc7519)

- `iss` is the URL of your credential issuer

- `aud` is the URL of the authorisation server your credential issuer relies on for authorisation. This must be set to the GOV.UK One Login authorisation server: https://token.account.gov.uk/

Signature

The pre-authorised code must be signed with an Elliptic Curve Digital Signature Algorithm (ECDSA) private key for signing. The corresponding public key, which forms a pair with the private key used for signing, must be made publicly accessible at your `/.well-known/jwks.json` (/credential-issuer-functionality/jwks) endpoint. This is because GOV.UK One Login will need access to the public key to verify the signature on the pre-authorised code.

The signing algorithm must be ECDSA with the P-256 curve and the SHA-256 hash function.

Storing the credential information

Your credential issuer will need to store some context to track the credential issuance process.

You must store the:

- credential offer identifier: this is the `credential_identifiers` in the pre-authorised code payload described above
- GOV.UK Wallet subject identifier (walletSubjectId): this was included in the user information your service received when the user authenticated with GOV.UK One Login
- credential details: this could be an identifier for the record to be issued as a credential and/or other relevant information about the credential itself

The credential issuer may also want to store the:

- offer creation timestamp: records when the credential offer was created
- offer expiration time: indicates when the credential offer will expire

Storing this information allows the credential issuer to track which credential is being offered and to which recipient and GOV.UK Wallet instance, and the specific context of the issuance request.

The reference implementation of the credential issuer uses AWS DynamoDB as the caching solution.

The credential offer URL

Your credential issuer must pass the credential offer to the GOV.UK Wallet **by value** with a URL.

The URL for passing a credential offer by value follows the following format:

1. The GOV.UK Wallet "credential offer endpoint"
2. A query parameter `credential_offer` that contains the Base64Url-encoded credential offer object

For example:

```
https://mobile.account.gov.uk/wallet/add?credential_offer=%7B%22credential_c
```

Displaying the credential offer URL

To provide a consistent experience across devices, your service's webpage should present the credential offer URL to the user in 2 ways:

- as a QR code: this is best for users accessing your service on a separate device. They can quickly scan the code to communicate the credential offer to their GOV.UK Wallet
- as a call-to-action (CTA) link: this is best for users already on their mobile device - tapping the CTA link directly opens the credential offer in their GOV.UK Wallet

This approach provides a user-friendly way to deliver the credential offer to GOV.UK Wallet, regardless of which device the user is on.

GOV.UK Wallet credential offer endpoints

The GOV.UK Wallet credential offer endpoint varies depending on the environment:

Environment	Credential offer endpoint URL
integration	`https://mobile.integration.account.gov.uk/wallet/add?credential_offer=`
production	`https://mobile.account.gov.uk/wallet/add?credential_offer=`

This page was set to be reviewed before 24 September 2025. This might mean the content is out of date.

Credential

The credential issuer credential endpoint is a required endpoint defined in the OID4VCI (https://openid.net/specs/openid-4-verifiable-credential-issuance-1_0.html#section-8) specification. It's where GOV.UK Wallet, acting on behalf of the holder, requests and receives verifiable credentials from the credential issuer. Government departments acting as credential issuers must implement this endpoint according to this specification to correctly integrate with GOV.UK Wallet.

Technical details

Endpoint URI

The credential endpoint's URI path is implementation-specific.

The credential issuer must publish the location of their credential endpoint in their issuer metadata API (/credential-issuer-functionality/metadata/api.html#well-known-openid-credential-issuer) using the `credential_endpoint` parameter.

Request format

The credential endpoint must accept HTTP POST requests.

GOV.UK Wallet will send a request to the credential issuer's credential endpoint to get a verifiable credential. The credential request must include:

- the Authorization header: a bearer access token (JWT) issued by GOV.UK One Login — this token authorises the credential issuance
- the request body (JSON): a proof of possession token (JWT) issued by GOV.UK Wallet — this proves the wallet controls the private key to which the credential will be bound

Request validation

Authorization header

The access token in the credential request is used to authorise the issuance of a verifiable credential. It's different from the access token used when the user initially logs in to

GOV.UK One Login, which is for authentication purposes. Because they have different roles, they are signed and verified using different keys.

The credential issuer must validate the access token to make sure that it was issued by GOV.UK One Login and that the request originates from the expected user.

To validate the access token, you should complete the following steps.

1. Verify the signature:

- retrieve the GOV.UK One Login JSON Web Key Set (JWKS) from its published JWKS endpoint
- extract the `kid` (key ID) parameter from the header
- find the matching public key in the JWKS by comparing `kid` values
- confirm the `alg` (algorithm) parameter in the token header matches the algorithm of the identified public key
- use the matching public key to cryptographically verify the token signature using the specified algorithm

2. Validate the header parameters by ensuring that:

- the value of the `typ` (type) parameter is `"at+jwt"`

Below is an example of an access token header:

```
{
  "alg": "ES256",
  "typ": "at+jwt",
  "kid": "8f9ec544-f5df-4d37-a32d-a5defd78ab0f"
}
```

3. Validate the payload claims by ensuring that:

- the value of the `iss` (issuer) claim matches the GOV.UK One Login URL: `"https://token.integration.account.gov.uk"` (integration) or `"https://token.account.gov.uk"` (production)
- the value of the `aud` (audience) claim is the credential issuer URL
- the value of the `sub` (subject - this is the wallet subject identifier) claim matches the value stored in your cache for this specific credential issuance flow
- the value of the `exp` (expiration time) claim is in the future
- the value of the `credential_identifiers` claim matches the value stored in your cache for this specific credential issuance flow

- the value of the c_nonce claim matches the value of the nonce claim in the [proof of possession (/credential-issuer-functionality/credential/#request-body)](/credential-issuer-functionality/credential/#request-body)
- this API has not received another access token with the same jti (JWT ID) that is still within its validity period

Below is an example of an access token payload:

```
{
  "sub": "urn:fdc:wallet.account.gov.uk:2024:DtPT8x-dp_73tnlY3KNTiCitziN9GEh
  "iss": "https://token.account.gov.uk",
  "aud": "https://example-credential-issuer.gov.uk",
  "exp": 1756115975,
  "credential_identifiers": [
    "daa01d3e-b17c-4c8a-8adf-ef808b456c9c"
  ],
  "c_nonce": "657a09cd-7165-486d-a858-065eb23f7a8d",
  "jti": "62b45850-4c5c-4696-983a-af66450301d4"
}
```

The sub claim validation is an important security control. This pairwise identifier, which starts with urn:fdc:wallet.account.gov.uk: , is generated by GOV.UK One Login for each user's wallet instance and helps ensure the credential is issued to the right user.

Your implementation must compare the sub value from the access token against the wallet subject identifier you got and stored when the user authenticated with GOV.UK One Login. This comparison makes sure that the wallet requesting the credential belongs to the same user who authenticated with your service.

If the identifiers do not match, the wallet trying to get the credential does not belong to the person logged in to your service. Your credential issuer must stop the issuance flow and consider logging the attempt for audit and fraud prevention.

Request body

The request body sent to the credential endpoint must be a JSON object containing proof of possession that demonstrates the wallet's control of the private key to which the credential will be bound.

It must contain the following parameters:

Parameter	Description	Value(s)
`proof`	A JSON containing the proof of possession of the cryptographic key material.	
`proof.proof_type`	A string indicating the type of proof being presented.	Must be `jwt` .
`proof.jwt`	The JSON Web Token (JWT) that serves as the proof.	

Below is an example of a request body:

```
{
   "proof":{
      "proof_type":"jwt",
      "jwt":"ew0KICAiYWxnIjogIkVTMjU2IiwNCiAgInR5cCI6ICJvcGVuaWQ0dmNpLXByb29
   }
}
```

The proof of possession is a cryptographic mechanism that verifies the wallet controls the private key that matches the public key (“did:key”) that will be associated with the credential. This ensures credentials are issued to their rightful holder.

This token is generated by GOV.UK Wallet and includes a cryptographic client `nonce` (from the access token issued by GOV.UK One Login) that has been signed with the wallet’s private signing key.. The `did:key` (the wallet’s public key) is included in the token’s header `kid` parameter.

When your credential issuer receives the credential request, it verifies the proof of possession signature with the `did:key` . Successful verification shows the wallet’s ownership of the private key corresponding to that public `did:key` .

There is more information about [the did:key method (/credential-issuer-functionality/credential/#the-did-key-format)](/credential-issuer-functionality/credential/#the-did-key-format).

To validate the proof, you should complete the following steps.

1. Verify the signature:

- extract the `kid` (key ID) parameter from the header, which contains the wallet’s `did:key`
- convert the `did:key` value to its corresponding public key

- confirm the `alg` (algorithm) parameter in the proof header is `ES256` and is compatible with the key type derived from the `did:key`
- use the public key to cryptographically verify the proof signature using the specified algorithm

2. Validate the header parameters by ensuring that:

- the value of the `typ` (type) parameter is `"openid4vci-proof+jwt"`

Below is an example of a proof header:

```
{
  "alg": "ES256",
  "typ": "openid4vci-proof+jwt",
  "kid": "did:key:zDnaeSGfSQMYvnLbLWEubhhGDPoq7pA9MMNvumvbsmMCZovUR"
}
```

3. Validate the payload claims by ensuring that:

- the value of the `iss` (issuer) claim matches the GOV.UK Wallet identifier - `urn:fdc:gov:uk:wallet`
- the value of the `aud` (audience) claim is the credential issuer URL
- the value of the `iat` (issued at) is a numeric date formatted as seconds since the epoch - this must be a date in the past that is after the pre-authorized code was generated
- the value of the `nonce` claim matches the value of the `c_nonce` claim in the access token

Below is an example of a proof of possession token payload:

```
{
  "iss": "urn:fdc:gov:uk:wallet",
  "aud": "https://example-credential-issuer.gov.uk",
  "iat": 1745233623816,
  "nonce": "bd423745-7705-45c2-9f51-6ae8dcac5589"
}
```

More information about the credential request can be found in the [OID4VCI specification (https://openid.net/specs/openid-4-verifiable-credential-issuance-1_0.html#section-8.2)](https://openid.net/specs/openid-4-verifiable-credential-issuance-1_0.html#section-8.2).

Successful response format

After validating the request successfully, the credential endpoint must return a 200 OK HTTP status code and a JSON response following the OID4VCI specification (https://openid.net/specs/openid-4-verifiable-credential-issuance-1_0.html#section-8.3). The response contains the issued credential as a JWT and a unique notification ID (a generated UUIDv4) for the callback success/failure notification:

```
HTTP/1.1 200 OK
Content-Type: application/json
Cache-Control: no-store

{
  "credentials": [
    {
      "credential": "eyJraWQiOiJkaWQ6d2ViOmV4YW1wbGUtY3JlZGVudGlhbC1pc3N1ZXI
    }
  ],
  "notification_id": "776aefd4-26c6-4a5f-aa7c-b5e294cd87cd"
}
```

The `notification_id` should only be included in the response if the credential issuer implements the notification endpoint (/credential-issuer-functionality/notification).

More information about the credential response can be found in the OID4VCI specification (https://openid.net/specs/openid-4-verifiable-credential-issuance-1_0.html#section-8.3).

The steps for constructing and issuing the verifiable credential are:

1. Retrieve the underlying data that will go into the credential from your database using the information in your issuance cache.
2. Build the verifiable credential according to the W3C Verifiable Credentials Data Model v2.0 (https://www.w3.org/TR/vc-data-model-2.0/).
3. Cryptographically bind the credential to the wallet (/credential-issuer-functionality/credential/#further-guidance-on-credential-binding) by using the wallet's `did:key` as the credential's subject identifier (the `sub` claim).
4. Sign the credential with your issuer's private key.

The example below shows the structure of a verifiable credential using a JSON Web Token (JWT) format for a fishing licence.

Header

```
{
  "alg": "ES256",
  "typ": "vc+jwt",
  "cty": "vc",
  "kid": "did:web:example-credential-issuer.gov.uk#5dcbee863b5d7cc30c9ba1f73
}
```

- `alg` (algorithm). REQUIRED. The cryptographic algorithm used to sign the JWT - must be `"ES256"` for ECDSA using the P-256 curve.
- `typ` (type). REQUIRED. The media type of the signed content - must be `"vc+jwt` .
- `cty` (content type). REQUIRED. The media type of the secured content (the payload) - must be `"vc"` .
- `kid` (key ID). REQUIRED. The DID URL of the issuer's public key used for signature verification - must match the `id` of the corresponding key in the credential issuer's DID Document, to let recipients locate the correct public key for signature verification.

Payload

```
{
  "iss": "https://example-credential-issuer.gov.uk",
  "sub": "did:key:ebfaeb1fd712ebf1c276e12ec21",
  "iat": "1712664731",
  "@context": [
    "https://www.w3.org/ns/credentials/v2",
    "<JSON-LD CONTEXT URI FOR ISSUER>"
  ],
  "type": [
    "VerifiableCredential",
    "FishingLicenceCredential"
  ],
  "issuer": "https://example-credential-issuer.gov.uk",
  "name": "Fishing licence",
  "description": "Permit for fishing activities",
  "validFrom": "2024-04-09T12:12:11Z",
  "validUntil": "2028-12-10T22:59:59Z",
  "credentialSubject": {
    "id": "did:key:ebfaeb1fd712ebf1c276e12ec21",
    "name": [
      {
        "nameParts": [
          {
            "value": "Sarah",
            "type": "GivenName"
          },
          {
            "value": "Edwards",
            "type": "FamilyName"
          }
        ]
      }
    ],
    "fishingLicenceRecord": [
      {
        "licenceNumber": "009878863",
        "issuanceDate": "2023-12-10",
        "expiryDate": "2028-12-10"
      }
    ]
  }
}
```

- `iss` (issuer). REQUIRED. The URL of the credential issuer service operated by the organisation sharing the credential.
- `sub` (subject). REQUIRED. The identifier of the holder of the information in the credential. The subject identifier is a decentralised identifier `did:key` generated by the wallet. In the credential issuance flow, the wallet shares its `did:key` with the issuer, and the issuer makes this the value of the credential's `sub` claim. This cryptographically binds the credential to the wallet.
- `iat` (issued at). OPTIONAL. The time at which the JWT was issued. Must be expressed in epoch time as per the IETF RFC 7519 (https://datatracker.ietf.org/doc/html/rfc7519).
- `@context` . REQUIRED. The context of the data exchange. It must be a set of URIs that point to documents that describe the context. The first item in the set must be the URI `"https://www.w3.org/ns/credentials/v2"` .
- `type` . REQUIRED. A set of values indicating the type of verifiable credentials issued by the issuer. The first value in the set must be `VerifiableCredential`
- `issuer` . REQUIRED. The URL of the credential issuer service operated by the organisation sharing the credential. Must be the same as the value of the `iss` claim.

- `name` . OPTIONAL. Issuer-specified credential name.
- `description` . OPTIONAL. Issuer-specified credential description.
- `validFrom` . OPTIONAL. It represents the date and time the credential becomes valid. Must be expressed in ISO 8601 format with seconds (`YYYY-MM-DDTHH:mm:ssZ`) as per the VC data model v2.0 (https://www.w3.org/TR/vc-data-model-2.0/).
- `validUntil` . REQUIRED. It represents the date and time the credential stops being valid. Must be expressed in ISO 8601 format with seconds (`YYYY-MM-DDTHH:mm:ssZ`) as per the VC data model v2.0 (https://www.w3.org/TR/vc-data-model-2.0/).
- `credentialSubject` . REQUIRED. An object containing claims about the holder of the verifiable credential.

Guidance on credential expiration

The `validUntil` claim in the credential specifies the date after which any consumer of the verifiable credential must consider it invalid (expired) and reject it. The entitlement in the underlying data source (the fishing license record in the example) may still be valid, as represented by the `expiryDate` but this digital credential representation of the underlying representation has expired. If the `validUntil` date has passed, the holder must get a new verifiable credential to update the digital representation of their entitlement.

Credentials issued to GOV.UK Wallet must include the `validUntil` claim. This date must be set to the same date as, or earlier than, the `credentialSubject.expiryDate` .

Always consider the expiration of the underlying credential when setting JWT expiration.

Guidance on including photos

If a photo is required in the credential, you must include it within the `credentialSubject` claim as a Base64-encoded string.

GOV.UK Wallet will validate the image to make sure that:

- The image’s format is JPG or PNG conforming to one of the following specifications:
  - `FF D8 FF E0 JPG`
  - `FF D8 FF EE JPG`
  - `FF D8 FF DB JPG`
  - `89 50 4E 47 0D 0A 1A 0A PNG`
  - `FF D8 FF E0 00 10 4A 46 49 46 00 01 JFIF`
- The image must not exceed 1 MiB (1 Mebibyte = 1,048,576 bytes) in size before encoding to Base64
- The image must have EXIF metadata stripped

If the GOV.UK Wallet fails to process an image in a credential, a `credential_failure` error will be returned, following the [notification specification (/credential-issuer-functionality/notification)](/credential-issuer-functionality/notification).

Below are examples of claims representing PNG and JPEG images respectively, encoded in Base64 format:

```
"photo": "iVBORw0KGgoAAAANSUhIAAAAE2BViAAAA[...]" // PNG file as source
```

```
"photo": "/9j/4AAQSkZJRgABAQpDYXQwMy5qcGf/[...]" // JPG file as source
```

Signature

The credential must be signed with your credential issuer’s private signing key using the ECDSA (Elliptic Curve Digital Signature Algorithm) cryptographic algorithm with P-256 (also known as Secp256r1) elliptic curve.

Error response format

If the credential request could not be processed successfully, the credential issuer must return an appropriate HTTP error status code.

When the credential request does not include an access token or the access token is invalid, the credential endpoint must return a 401 Unauthorized HTTP status code and include the `WWW-Authenticate` response header field as defined in [RFC6750 (https://datatracker.ietf.org/doc/html/rfc6750#section-3)](https://datatracker.ietf.org/doc/html/rfc6750#section-3), specifying the `Bearer` authentication scheme.

This is an example error response to a request with no access token:

```
HTTP/1.1 401 Unauthorized
WWW-Authenticate: Bearer
Cache-Control: no-store
```

This is an example error response to an authentication attempt using an invalid access token:

```
HTTP/1.1 401 Unauthorized
WWW-Authenticate: Bearer error="invalid_token"
Cache-Control: no-store
```

When there is an issue with the request body, the credential endpoint must return a 400 Bad Request HTTP status code and a response body in JSON format containing the following parameter:

- `error` : A case-sensitive string indicating the error - `invalid_proof` (the proof of possession is invalid), or `invalid_nonce` (the `nonce` is invalid or does not match the access token's `c_nonce`)

Below is an example of a credential error response when the credential request included an invalid proof of possession:

```
HTTP/1.1 400 Bad Request
Content-Type: application/json
Cache-Control: no-store

{
  "error": "invalid_proof"
}
```

Further guidance on credential binding

Binding credentials to users

Because each GOV.UK Wallet instance can be uniquely identified, your service can bind a credential with a specific wallet instance. GOV.UK Wallet uses a specific type of decentralised identifier (DID) (https://www.w3.org/TR/did-core/) called a `did:key` to cryptographically bind credentials to a user's wallet.

A did:key (https://w3c-ccg.github.io/did-method-key/) is a DID method. The DID represents the public key of an asymmetric key pair generated when GOV.UK Wallet is installed on a device. The private key never leaves the device, whereas the `did:key` is shared with credential issuers and verifiers. This allows credentials to be cryptographically bound to a specific GOV.UK Wallet instance.

The GOV.UK Wallet creates a `did:key` from a **P-256** (also known as Secp256r1) elliptic curve public key.

The `did:`key format

The `did:key` method is used to transfer public keys.

The format of a `did:key` is `did:key:multibaseValue` . The `multibaseValue` is the base58-btc multibase string representation of concatenating the multicodec identifier for the public key type and the compressed public key.

```
did-key-format := did:key:MULTIBASE(base58-btc, MULTICODEC(public-key-type,
```

In Elliptic Curve Cryptography (ECC), the public key is a pair of `x` and `y` coordinates. A compressed public key is the `x` coordinate, which is 32 bytes in length, with a prefix, of 1 byte in length, that indicates whether the `y` coordinate is even or odd. The prefix is `02` if the `y` coordinate is even and `03` if it is odd. The resulting compressed public key is **33 bytes** in length:

```
Public key: 52972572d465d016d4c501887b8df303eee3ed602c056b1eb09260dfa0da0ab2

Public key (x coordinate): 52972572d465d016d4c501887b8df303eee3ed602c056b1eb
Public key (y coordinate): 88742f4dc97d9edb6fd946babc002fdfb06f26caf117b9405

// y coordinate is even so "02" is prepended to the x coordinate
Public key (compressed): 0252972572d465d016d4c501887b8df303eee3ed602c056b1eb
```

The `multibaseValue` is generated as follows:

1. Encode the compressed public key as bytes
2. Prefix the key bytes with the **curve multicodec value** encoded as **unsigned varint** (variable length integers)
3. the multicodec hexadecimal value of a P-256 elliptic curve public key is `0x1200`, in varint-encoded bytes that is `[0x80, 0x24]`
4. Encode the above with **base58-btc** and then prefix it with `"z"` to indicate the base58-btc encoding - the result is the `multibaseValue`

The following is an example of a `did:key` derived from a base-58 encoded P-256 public key:

```
did:key:zDnaewZMz7MN6xSaAFADkDZJzMLbGSV25uKHAeXaxnPCwZomX
```

All DIDs derived from a P-256 public key always start with `"zDn"`.

Verifying a credential

To share a verifiable credential with a verifier, the wallet creates a verifiable presentation containing the verifiable credential and signs the credential with the wallet's private key.

The verifier must be able to confirm that the system presenting the credential (GOV.UK Wallet) is also the intended holder of that credential. The verifier must confirm **proof of possession** of the verifiable credential. This is done by verifying that the entity which signed the verifiable presentation is the same as the subject of the verifiable credential. In other words, the `did:key` in the verifiable credential must be able to verify the signature on the verifiable presentation.

This page was last reviewed on 25 August 2025. It needs to be reviewed again on 25 February 2026 .

API

/.well-known/did.json

get

A public endpoint for GOV.UK Wallet and credential verifiers to retrieve the credential issuer’s public keys for verifying credentials

Responses

Status	Description
200	Credential issuer’s DID document.

```
{
  "@context": [
    "https://www.w3.org/ns/did/v1",
    "https://w3id.org/security/suites/jws-2020/v1"
  ],
  "id": "did:web:example-credential-issuer.gov.uk",
  "verificationMethod": [
    {
      "id": "did:web:example-credential-issuer.gov.uk#5dcbee863b5d7cc30c9ba1f7393dacc6c16610782e4b6a191f94a7e8b1e15
      "type": "JsonWebKey2020",
      "controller": "did:web:example-credential-issuer.gov.uk",
      "publicKeyJwk": {
        "kty": "EC",
        "kid": "5dcbee863b5d7cc30c9ba1f7393dacc6c16610782e4b6a191f94a7e8b1e1510f",
        "crv": "P-256",
        "x": "6jCKX_QRrmTeEJi-uiwcYqu8BgMgl70g2pdAst24MPE",
        "y": "icPzjbSk6apD_SNvQt8NWOPlPeGG4KYU55GfnARryoY",
        "alg": "ES256"
      }
    }
  ],
  "assertionMethod": [
    "did:web:example-credential-issuer.gov.uk#5dcbee863b5d7cc30c9ba1f7393dacc6c16610782e4b6a191f94a7e8b1e1510f"
  ]
}
```

Schemas

DidResponse

Name	Type	Required	Description	Schema
@context	array	true	An array of URL contexts which define the terms used in the DID document.	
id	string	true	The unique identifier of the DID document.	

Name	Type	Required	Description	Schema
verificationMethod	array	true	An array of verification methods (cryptographic public keys).	VerificationMethod
assertionMethod	array	true	Array of DID URLs where each URL uniquely identifies a verification method within the DID document.	

VerificationMethod

Name	Type	Required	Description	Schema
id	string	true		
type	string	true		
controller	string	true		
publicKeyJwk	object	true		PublicKeyJwk

PublicKeyJwk

Name	Type	Required	Description	Schema
kty	string	true	Key Type. The family of cryptographic algorithms used with the key.	
kid	string	true	Key ID. Unique identifier to match a specific key.	
crv	string	true	Curve. Cryptographic curve used with the key.	
x	string	true	The “x” coordinate for the elliptic curve point.	
y	string	true	The “y” coordinate for the elliptic curve point.	
alg	string	true	Algorithm. The cryptographic algorithm used with the key.	

This page was set to be reviewed before 5 September 2025. This might mean the content is out of date.

DID document

The DID document endpoint lets GOV.UK Wallet verify the credentials it receives from credential issuers, who must use decentralised identifiers (DIDs).

This endpoint exposes the credential issuer's DID document, which contains the public cryptographic keys used to verify signatures on credentials issued by the credential issuer. This process checks the credentials' authenticity and integrity.

Technical details

Endpoint location

The DID document must be publicly accessible at the standardised location `/.well-known/did.json` on the credential issuer's domain.

Response format

The endpoint must return a 200 OK HTTP status code and a valid JSON response that follows the W3C Decentralized Identifiers (DIDs) v1.0 specification (https://www.w3.org/TR/did-1.0/).

The DID document must include the following parameters:

- `@context` : A set of URI references that define the meaning of the terms used in the DID document, allowing its properties to be correctly interpreted
- `id` : The decentralised identifier of the credential issuer in the format `"did:web:<CREDENTIAL-ISSUER-DOMAIN>"`
- `verificationMethod` : An array of verification methods containing the credential issuer's public keys
- `assertionMethod` : An array listing which verification methods can be used for making assertions - in the context of the credential issuer, this means issuing credentials

Each object in the array must contain:

- `id` : A unique identifier for the verification method in the format `"did:web:<CREDENTIAL-ISSUER-DOMAIN>#<KEY-ID>"` - this corresponds to the `kid` (key ID) parameter in the header of issued credentials
- `type` : The cryptographic suite used for verification - this must be `JsonWebKey2020` as the [JSON Web Signature 2020 specification (https://www.w3.org/community/reports/credentials/CG-FINAL-lds-jws2020-20220721/)](https://www.w3.org/community/reports/credentials/CG-FINAL-lds-jws2020-20220721/) is used
- `controller` : The entity (the credential issuer in this case) that has the authority to use the private key associated with the verification method - this must be in the format `"did:web:<CREDENTIAL-ISSUER-DOMAIN>>"`
- `publicKeyJwk` : The credential issuer's public key - this is represented as a JSON Web Key (JWK) and contains parameters specific to the elliptic curve algorithm used (P-256)

Each object in the `verificationMethod` array represents a public key, which is the counterpart to the private key held securely by the credential issuer. These private keys are used to sign credentials. GOV.UK Wallet only accepts credentials signed using an elliptic curve key based on the P-256 curve.

Not all public keys listed in the `verificationMethod` are authorised for credential issuance. The `assertionMethod` array acts as an allow list, specifying which of those keys are trusted for credential issuance. Each item in `assertionMethod` directly references a public key in the `verificationMethod` array using its unique ID.

When GOV.UK Wallet receives a credential issued by the credential issuer, it uses the `kid` (key ID) in the credential's header to find the matching public key in the `verificationMethod` array. Then, it checks if the same verification method ID is present in the `assertionMethod` array. This process confirms that the key used to sign the credential was authorised for credential issuance by the DID controller (the credential issuer).

DID document example

Below is an example of a DID document containing an elliptic curve key based on the P-256 curve in the verification method:

```
{
   "@context":[
      "https://www.w3.org/ns/did/v1",
      "https://w3id.org/security/suites/jws-2020/v1"
   ],
   "id":"did:web:example-credential-issuer.gov.uk",
   "verificationMethod":[
      {
         "id":"did:web:example-credential-issuer.gov.uk#5dcbee863b5d7cc30c9b
         "type":"JsonWebKey2020",
         "controller":"did:web:example-credential-issuer.gov.uk",
         "publicKeyJwk":{
            "kty":"EC",
            "kid":"5dcbee863b5d7cc30c9ba1f7393dacc6c16610782e4b6a191f94a7e8b
            "crv":"P-256",
            "x":"6jCKX_QRrmTeEJi-uiwcYqu8BgMgl70g2pdAst24MPE",
            "y":"icPzjbSk6apD_SNvQt8NWOPlPeGG4KYU55GfnARryoY",
            "alg":"ES256"
         }
      }
   ],
   "assertionMethod":[
      "did:web:example-credential-issuer.gov.uk#5dcbee863b5d7cc30c9ba1f7393d
   ]
}
```

This page was set to be reviewed before 3 October 2025. This might mean the content is out of date.

API

A public endpoint for GOV.UK One Login to retrieve the credential issuer's JSON Web Key Set (JWKS) of public keys which can be used to verify the the pre-authorised code.

/.well-known/jwks.json

get

A public endpoint that stores the JSON Web Key Set (JWKS) of public keys issued by a service. These keys can be used by client applications to verify the signature of a JSON Web Token (JWT).

Responses

Status	Description	Schema
200	Credential issuer's JWKS.	JwksResponse

```
{
  "keys": [
    {
      "kty": "EC",
      "use": "sig",
      "crv": "P-256",
      "kid": "5dcbee863b5d7cc30c9ba1f7393dacc6c16610782e4b6a191f94a7e8b1e1510f",
      "x": "6jCKX_QRrmTeEJi-uiwcYqu8BgMgl70g2pdAst24MPE=",
      "y": "icPzjbSk6apD_SNvQt8NWOPlPeGG4KYU55GfnARryoY=",
      "alg": "ES256"
    }
  ]
}
```

Schemas

JwksResponse

Name	Type	Required	Description	Schema
keys	array	true	A set of public keys, each in JSON Web Key (JWK) format.	Key

Key

Name	Type	Required	Description	Schema
kty	string	true	Key Type. The family of cryptographic algorithms used with the key.	
kid	string	true	Key ID. Unique identifier to match a specific key.	
crv	string	true	Curve. Cryptographic curve used with the key.	
x	string	true	The “x” coordinate for the elliptic curve point.	
y	string	true	The “y” coordinate for the elliptic curve point.	
alg	string	true	Algorithm. The cryptographic algorithm used with the key.	
use	string	true	The intended use of the key.	

This page was set to be reviewed before 5 September 2025. This might mean the content is out of date.

JSON Web Key Set (JWKS)

The JWKS endpoint is a required endpoint that exposes the credential issuer's public cryptographic keys, which can be used by GOV.UK Wallet to verify the pre-authorised code, in JSON Web Key Set (JWKS) format.

This endpoint lets the GOV.UK One Login retrieve the credential issuer's public keys and then verify the pre-authorised code (/credential-issuer-functionality/credential-offer/#the-pre-authorised-code) signature. This process confirms that the pre-authorised code was issued by the expected credential issuer and that it has not been tampered with.

Technical details

Endpoint location

The JWKS must be publicly accessible at the standardised location `/.well-known/jwks.json` on the credential issuer's domain.

Response format

The endpoint must return a 200 OK HTTP status code and a valid JSON response that follows the JWKS specification defined in RFC 7517 (https://datatracker.ietf.org/doc/html/rfc7517). Each key within the JWKS is represented as a JSON Web Key (JWK) object. The JWKS usually contains only one key, but it can contain two keys during a key rotation overlap period.

The JWK for an elliptic curve key based on the P-256 curve must include the following parameters:

- `kty` : The family of cryptographic algorithms used with the key - must be "EC".
- `kid` : A unique identifier for a specific key within the set - this value will be referenced in the pre-authorised code header to show which key was used for signing and which key must be used for verification. This parameter is important for associating the correct public key with the pre-authorised code being verified.
- `crv` : Cryptographic curve used with the key - must be "P-256".
- `x` : The "x" coordinate for the elliptic curve point.
- `y` : The "y" coordinate for the elliptic curve point.

- `alg` : The cryptographic algorithm used with the key - must be “ES256”.
- `use` : The intended use of the key - must be “sig” to indicate a signing key.

JWKS example

Below is an example of a JWKS containing one elliptic curve key based on the P-256 curve:

```
{
  "keys": [
    {
      "kty": "EC",
      "use": "sig",
      "crv": "P-256",
      "kid": "5dcbee863b5d7cc30c9ba1f7393dacc6c16610782e4b6a191f94a7e8b1e151
      "x": "6jCKX_QRrmTeEJi-uiwcYqu8BgMgl70g2pdAst24MPE",
      "y": "icPzjbSk6apD_SNvQt8NWOPlPeGG4KYU55GfnARryoY",
      "alg": "ES256"
    }
  ]
}
```

This page was set to be reviewed before 3 October 2025. This might mean the content is out of date.

API

/.well-known/openid-credential-issuer

get

A public endpoint for the GOV.UK Wallet to retrieve metadata about the credential issuer.

Responses

Status	Description	Schema
200	Credential issuer’s metadata.	MetadataResponse

```
{
  "credential_issuer": "https://example-credential-issuer.gov.uk",
  "authorization_servers": [
    "https://token.account.gov.uk"
  ],
  "credential_endpoint": "https://example-credential-issuer.gov.uk/credential",
  "notification_endpoint": "https://example-credential-issuer.gov.uk/notification",
  "credential_configurations_supported": {
    "FishingLicenceCredential": {
      "format": "jwt_vc_json",
      "credential_definition": {
        "type": [
          "VerifiableCredential",
          "FishingLicenceCredential"
        ]
      },
      "cryptographic_binding_methods_supported": [
        "did"
      ],
      "credential_signing_alg_values_supported": [
        "ES256"
      ],
      "proof_types_supported": {
        "jwt": {
          "proof_signing_alg_values_supported": [
            "ES256"
          ],
          "key_attestations_required": null
        }
      },
      "display": [
        {
          "name": "Fishing Licence number",
          "locale": "en-GB",
          "background_color": "#12107c",
          "text_color": "#FFFFFF"
        },
        {
          "name": "Rhif Trwydded Pysgota",
          "locale": "en-CY",
```

Status	Description	Schema

```
          "background_color": "#12107c",
          "text_color": "#FFFFFF"
        }
      ],
      "credentialSubject": {
        "name": [
          {
            "nameParts": [
              {
                "display": [
                  {
                    "name": "Name",
                    "locale": "en-GB"
                  },
                  {
                    "name": "Enw",
                    "locale": "cy-GB"
                  }
                ]
              }
            ]
          }
        ],
        "fishingLicenceRecord": {
          "licenceNumber": {
            "display": [
              {
                "name": "Fishing Licence number",
                "locale": "en-GB"
              },
              {
                "name": "Rhif Trwydded Pysgota",
                "locale": "cy-GB"
              }
            ]
          },
          "expirationDate": {
            "display": [
              {
                "name": "Dyddiad dod i ben",
                "locale": "en-GB"
              },
              {
                "name": "Rhif Trwydded Pysgota",
                "locale": "cy-GB"
              }
            ]
          }
        }
      }
    }
  }
}
```

Schemas

MetadataResponse

Name	Type	Required	Description	Schema
credential_issuer	string	true	URL of the credential issuer.	
credential_endpoint	string	true	URL of the credential issuer's credential endpoint.	
notification_endpoint	string	false	URL of the credential issuer's notification endpoint.	
authorization_servers	array	true	Set containing the URL of the authorization server(s) the credential issuer relies on for authorization.	
credential_configurations_supported	object	true	Information about the credential(s) issued by the credential issuer.	CredentialConfigurationsSupported

CredentialConfigurationsSupported

Information about the credential(s) issued by the credential issuer.

Name	Type	Required	Description	Schema
fishingLicence	object	true	A credential issued by the credential issuer.	FishingLicence

FishingLicence

A credential issued by the credential issuer.

Name	Type	Required	Description	Schema
format	string	true	Format of the credential.	
credential_definition	object	true	Description of the credential type.	CredentialDefinition
cryptographic_binding_methods_supported	array	true	Set of methods available for cryptographically binding the	

Name	Type	Required	Description	Schema
			issued credential.	
credential_signing_alg_values_supported	array	true	Set of algorithms that the credential issuer uses to sign the credential.	
proof_types_supported	object	true	Key proof(s) supported by the credential issuer.	ProofTypesSupported

CredentialDefinition

Description of the credential type.

Name	Type	Required	Description	Schema
type	array	true		

ProofTypesSupported

Key proof(s) supported by the credential issuer.

Name	Type	Required	Description	Schema
jwt	object	true		Jwt

Jwt

Name	Type	Required	Description	Schema
proof_signing_alg_values_supported	array	true		

This page was last reviewed on 9 June 2025. It needs to be reviewed again on 9 December 2025 .

Metadata

The metadata endpoint is a required endpoint that provides essential configuration information about the credential issuer.

This endpoint lets GOV.UK Wallet and verifiers dynamically learn information about the credential issuer, such as:

- the endpoints used in the issuance flow
- the supported credential types
- how credentials should be displayed in the wallet

Technical details

Endpoint location

The metadata must be publicly accessible at the standardised location `/.well-known/openid-credential-issuer` on the credential issuer's domain. The data published is non-sensitive metadata about the service.

Response format

The endpoint must return a 200 OK HTTP status code and valid JSON response that follows the OID4VCI specification (https://openid.net/specs/openid-4-verifiable-credential-issuance-1_0.html#name-metadata).

The metadata must include the following parameters:

- `credential_issuer` : The URL of the credential issuer.
- `authorization_servers` : An array of URLs for the authorisation servers the credential issuer relies on for authorisation. This must be set to the GOV.UK One Login URL.
- `credential_endpoint` : The URL of the credential issuer's credential endpoint (/credential-issuer-functionality/credential/#credential), where credentials can be obtained.
- `credential_configurations_supported` : An object describing the credentials offered by the credential issuer.

If your credential issuer implements the optional notification endpoint (/credential-issuer-functionality/notification/#notification), then the metadata must include the `notification_endpoint` parameter.

You can define and use additional metadata parameters.

Credential information

The `credential_configurations_supported` object contains key/value pairs, where each key is a unique identifier of a verifiable credential supported by the credential issuer and the value is the configuration of that verifiable credential.

Each credential object in `credential_configurations_supported` must include the following parameters:

- `format`
- `credential_definition`
- `cryptographic_binding_methods_supported`
- `credential_signing_alg_values_supported`
- `proof_types_supported`

There is more information about the `credential_configurations_supported` parameter (https://openid.net/specs/openid-4-verifiable-credential-issuance-1_0.html#section-11.2.4).

Metadata example

Below is an example of a credential issuer metadata:

```
{
  "credential_issuer": "https://example-credential-issuer.gov.uk",
  "authorization_servers": ["https://token.account.gov.uk"],
  "credential_endpoint": "https://example-credential-issuer.gov.uk/credentia
  "notification_endpoint": "https://example-credential-issuer.gov.uk/notific
  "credential_configurations_supported": {
    "FishingLicenceCredential": {
      "format": "jwt_vc_json",
      "credential_definition": {
        "type": [
          "VerifiableCredential",
          "FishingLicenceCredential"
      ]},
      "cryptographic_binding_methods_supported": [
        "did"
      ],
      "credential_signing_alg_values_supported": [
        "ES256"
      ],
      "proof_types_supported": {
        "jwt": {
          "proof_signing_alg_values_supported": [
            "ES256"
          ]
        }
      }
    }
  }
}
```

This page was set to be reviewed before 3 October 2025. This might mean the content is out of date.

API

/notification

post

An endpoint used by GOV.UK Wallet to notify the credential issuer of events concerning issued credentials.

Responses

Status	Description	Schema
204	No Content	
400	Bad Request ```{ "error": "invalid_notification_id" }```	Notification400ErrorResponse
401	Unauthorized	

```
{
  "error": "invalid_notification_id"
}
```

Schemas

Notification400ErrorResponse

Name	Type	Required	Description	Schema
error	string	true	An error code - must be `invalid_notification_request` ,	

Name	Type	Required	Description	Schema
			`invalid_notification_id` , or `invalid_request` .	
error_description	string	false	A human-readable explanation of the error.	

This page was set to be reviewed before 19 August 2025. This might mean the content is out of date.

Notification

The credential issuer notification endpoint is an optional endpoint defined in the OID4VCI (https://openid.net/specs/openid-4-verifiable-credential-issuance-1_0.html#name-notification-endpoint) specification. GOV.UK Wallet uses this endpoint to notify the credential issuer of events relating to issued credentials.

Notifications can be sent about events such as:

- the credential has been successfully stored in the GOV.UK Wallet
- the credential could not be stored in the GOV.UK Wallet because it is invalid
- the user has taken specific actions related to the offered credential, such as declining to save it

In order for GOV.UK Wallet to send notifications, the credential issuer must include a unique `notification_id` parameter in each credential response (/credential-issuer-functionality/credential/#credential-response).

Technical details

Endpoint location

The notification endpoint's location is implementation-specific within the OID4VCI specification.

The credential issuer must publish the location of their notification endpoint in their metadata (/credential-issuer-functionality/metadata/api.html#well-known-openid-credential-issuer) using the `notification_endpoint` parameter.

Request format

The notification endpoint must accept HTTP POST requests. The request must include the:

- authorization header: A bearer access token (JWT) issued by GOV.UK One Login
- request body (JSON): The notification details

Request validation

Authorization header

The request must include an access token issued by the GOV.UK One Login token service as a bearer token in the Authorization header. This is the same as the access token used for authorising the issuance of a verifiable credential.

The credential issuer must validate the access token to ensure that it was issued by GOV.UK One Login and the request originates from the expected user.

To validate the access token, you should complete the following steps.

1. Verify the signature:

- retrieve the GOV.UK One Login token service's JSON Web Key Set (JWKS) from their published JWKS endpoint
- extract the `kid` (key ID) parameter from the access token header
- find the matching public key in the JWKS by comparing `kid` values
- confirm the `alg` (algorithm) parameter in the token header matches the algorithm of the identified public key
- use the matching public key to cryptographically verify the token signature using the specified algorithm

2. Validate the header parameters by ensuring that:

- the value of the `typ` (Type) parameter is `"at+jwt"`

3. Validate the payload claims by ensuring that:

- the value of the `iss` (issuer) claim matches the GOV.UK One Login URL: `"https://token.integration.account.gov.uk"` (integration) or `"https://token.account.gov.uk"` (production)
- the value of the `aud` (audience) claim is credential issuer URL
- the value of the `sub` (subject - this is the wallet subject identifier) claim matches the value stored in your cache for this specific credential issuance flow
- the value of the `exp` (expiration time) claim is in the future
- the value of the `credential_identifiers` claim matches the value stored in your cache for this specific credential issuance flow
- this API has not received another access token with the same `jti` (JWT ID) that is still within its validity period

Request body

The request body must be in JSON format and contain the following parameters:

Parameter	Description	Value(s)
`notification_id`	A string (this could be a UUID) received in the credential response, uniquely identifying an individual credential issuance occurrence.	
`event`	A case-sensitive string indicating the credential's status.	One of the following enums: `credential_accepted` (GOV.UK Wallet accepted the credential) `credential_failure` (GOV.UK Wallet rejected the credential) `credential_deleted` (user declined or deleted the credential)
`event_description`	An optional parameter that GOV.UK Wallet may include to provide additional information about the event.	

The credential issuer must validate the contents of the request body and should ignore any unrecognised parameters.

Below is an example of a notification when a credential is successfully stored in the GOV.UK Wallet:

```
{
  "notification_id": "776aefd4-26c6-4a5f-aa7c-b5e294cd87cd",
  "event": "credential_accepted",
  "event_description": "Credential has been successfully stored"
}
```

More information about the notification request can be found in the OID4VCI specification (https://openid.net/specs/openid-4-verifiable-credential-issuance-1_0.html#section-10.1).

Successful response format

When a notification is processed successfully, the credential issuer must return a 204 No Content HTTP status code.

Error response format

If the notification could not be processed successfully, the credential issuer must return an appropriate HTTP error status code.

When the notification request does not include an access token or the access token is invalid, the notification endpoint must return a 401 Unauthorized HTTP status code and include the `WWW-Authenticate` response header field as defined in [RFC6750 (https://datatracker.ietf.org/doc/html/rfc6750#section-3)](https://datatracker.ietf.org/doc/html/rfc6750#section-3), specifying the `Bearer` authentication scheme.

This is an example error response to a request with no access token:

```
HTTP/1.1 401 Unauthorized
WWW-Authenticate: Bearer
Cache-Control: no-store
```

This is an example error response to an authentication attempt using an invalid access token:

```
HTTP/1.1 401 Unauthorized
WWW-Authenticate: Bearer error="invalid_token"
Cache-Control: no-store
```

When there is an issue with the request body (e.g. the `notification_id` value is invalid), the notification endpoint must return a 400 Bad Request HTTP status code and a response body in JSON format containing the following parameter:

- `error` : A case-sensitive string indicating the error - `invalid_notification_id` (the request’s `notification_id` was invalid), or `invalid_notification_request` (the request was invalid)

Below is an example of a notification error response when the notification request included an invalid `notification_id` :

```
HTTP/1.1 400 Bad Request
Content-Type: application/json
Cache-Control: no-store
```

```
{
  "error": "invalid_notification_id"
}
```

More information about the error notification responses can be found in the OID4VCI specification (https://openid.net/specs/openid-4-verifiable-credential-issuance-1_0.html#section-10.3).

Idempotency

The notification endpoint must be implemented idempotently. Identical requests with the same `notification_id` should always return the same response.

This page was last reviewed on 25 August 2025. It needs to be reviewed again on 25 February 2026 .

Credential Issuer functionality

In this section:

- Authenticate users with One Login
- Generate a credential offer
- Publish your metadata
- Publish your JSON Web Key Set (JWKS)
- Issue a credential
- Publish your DID document
- Handle notifications from GOV.UK Wallet

This page was set to be reviewed before 5 September 2025. This might mean the content is out of date.

Issuing credentials to GOV.UK Wallet

GOV.UK Wallet will support multiple credential formats to represent government documents. These documents can be:

- mdoc based credentials for the digital driving licence (https://www.iso.org/obp/ui/en/#iso:std:iso-iec:18013:-5:ed-1:v1:en)
- other Verifiable Credentials (VCs), including W3C Verifiable Credential Data Model 2.0 (https://www.w3.org/TR/vc-data-model-2.0/) and later other formats allowing selective disclosure of attributes (https://datatracker.ietf.org/doc/draft-ietf-oauth-sd-jwt-vc/)

GOV.UK Wallet follows the OpenID Connect for Verifiable Credential Issuance (OIDC4VCI) (https://openid.net/specs/openid-4-verifiable-credential-issuance-1_0.html) open standard for its issuance flow.

Your team or department can start issuing credentials to GOV.UK Wallet by following this documentation.

Understand GOV.UK Wallet's credential exchange flow

This diagram shows the exchange of a credential between a government service and GOV.UK Wallet. Below the diagram is an explanation of each step in the process.

User authenticates with GOV.UK One Login to use your service (/credential-issuer-functionality/authenticating-users/#authenticating-users-with-one-login)

1. Your user accesses your service.
2. Your service authenticates the user with GOV.UK One Login.
3. Your service fetches the user's `walletSubjectId` from the GOV.UK One Login `/userinfo` API (https://docs.sign-in.service.gov.uk/integrate-with-integration-environment/authenticate-your-user/#retrieve-user-information).

There is detailed guidance on how GOV.UK One Login works (https://docs.sign-in.service.gov.uk/how-gov-uk-one-login-works/) in the GOV.UK One Login technical documentation.

Your service issues a credential offer (/credential-issuer-functionality/credential-offer/#credential-offer)

4. Your service generates a credential offer. Included in this offer is a pre-authorised code (/credential-issuer-functionality/credential-offer/#the-pre-authorised-code) signed by your service.
5. Your service renders the credential offer to the user as a QR code or deep-link.
6. The user opens the app.
7. The app prompts the user to authenticate with GOV.UK One Login.
8. The user who authenticated with your service in a web browser is authenticated with GOV.UK One Login in the app.
9. The user scans the QR code or opens the deep link. This action passes the credential offer to GOV.UK Wallet.

Your service publishes metadata about the credentials it publishes

10. GOV.UK Wallet sends a GET request to your `/.well-known/openid-credential-issuer` endpoint to fetch your metadata.
11. Your service returns its metadata.
12. GOV.UK Wallet calls GOV.UK One Login to exchange the pre-authorised code in the credential offer for an access token.
13. GOV.UK One Login sends a GET request to your `/.well-known/jwks.json` endpoint to fetch your public keys, which verify the signature on the pre-authorised code issued by your service.
14. Your service returns its public keys as a JSON Web Key Set (JWKS).
15. GOV.UK One Login verifies the pre-authorised code content and its signature.

16. GOV.UK One Login issues an access token that you can trust when GOV.UK Wallet calls your service to redeem it.
17. GOV.UK Wallet generates a proof of possession for the key material.
18. GOV.UK Wallet sends a POST request to your `/credential` endpoint to request the credential. This request includes the access token issued by GOV.UK One Login (as a bearer token in the authorization header) and the proof of possession generated by GOV.UK Wallet.

Your service issues a credential (/credential-issuer-functionality/credential/#credential)

19. Your service sends a GET request to the GOV.UK One Login `/.well-known/jwks.json` to fetch its public keys, which verify the signature on the access token issued by GOV.UK One Login.
20. GOV.UK One Login returns its public keys as a JSON Web Key Set (JWKS).
21. Your service verifies the content and signature of the access token and the proof of possession.
22. Your service compares the `walletSubjectId` in the access token's `sub` claim with the `walletSubjectId` retrieved in step 3. If they are the same, this provides assurance that you are issuing the credential to a digital wallet that is logged in as the rightful holder.
23. Your service builds and signs the credential, and binds it to the did:key provided in the proof of possession to make sure the credential can only be used by the device it is issued to.
24. Your service returns the device-bound credential to GOV.UK Wallet.
25. GOV.UK Wallet sends a GET request to your `/.well-known/did.json` endpoint to fetch your DID document. The DID document contains your public key which is required to verify the signature on the credential issued by your service.
26. Your service returns its DID document.
27. GOV.UK Wallet verifies the content and signature of the credential.
28. GOV.UK Wallet stores the credential.

GOV.UK Wallet notifies your service (/credential-issuer-functionality/notification/)

The following steps are optional. If you do not offer a `/notification` endpoint then GOV.UK Wallet will not send a notification.

29. GOV.UK Wallet sends a POST request to your `/notification` endpoint to notify your service. This notification will confirm whether GOV.UK Wallet successfully stored the credential, or failed to store it.
30. Your service records the notification.
31. Your service returns an empty response to GOV.UK Wallet.

This page was last reviewed on 14 May 2025. It needs to be reviewed again on 14 November 2025 .

Key Management

GOV.UK Wallet needs to verify the validity of the credentials your service issues.

When issuing credentials in W3C Verifiable Credential Data Model 2.0 (https://www.w3.org/TR/vc-data-model-2.0/) format and signing with your private keys, your credentials need to be verified by the public keys you made available in the did:web document.

Your public keys need to stay available through the lifecycle of your credentials. A public key used to sign a group of verifiable credentials (VCs) can not be made inactive until after the VCs have expired. Public keys should be kept in an inactive state, available to be verified.

For their credential issuer service, credential issuers should include specific key management features.

The service needs a key refresh process that creates a new asymmetric public or private key pair for signing new VCs, but that retains trust in the previous versions of the public key for verifying.

This is done by making sure the public part of the historical key is retained, while the private key is destroyed. For example, a VC issued by an internal and external issuer.

The service also needs key revocation. This needs to include a notice made from the credential issuer explaining that a specific key should be removed from operational use before the key expires. This will generally happen when the key is lost or compromised. If a key is compromised, it can be used by an attacker to decrypt or forge messages, impersonate an identity, or access sensitive information.

The table below describes the possible states of a key pair used for signing credentials:

Key State	Description
Created	A key pair is generated with an activation date in the future. It is not yet used for signing.
Active	A key becomes active on the activation date, and enabled for signing and verifying the VC. There must not be multiple keys active at the same time.

Inactive	A key becomes inactive past its expiration date or time. The public key will still be valid for verifying the VC.
Revoked	A key is destroyed and removed from the issuer's server, and is not valid for signing or verifying the signatures.

This page was set to be reviewed before 5 September 2025. This might mean the content is out of date.

Before you issue a status record

To issue credentials with a status, you must register with the Status List Service. To do this, speak to your GOV.UK Wallet engagement manager or contact us (/contact-us.html).

Register a status list client

Only valid status list clients can issue and revoke statuses in the Status List Service.

When registering as a status list client, you must provide:

- a public JSON web key set (JWKS) endpoint - used to verify the signatures of the status list client's signed JSON web tokens (JWTs) for the issue and revoke endpoint
- a status list type (Bitstring or Token) - the type of status issued for this specific status list client

You can register multiple status list clients if you need to.

When you complete your registration, you get a unique `clientId`. You must include your `clientId` identifier as the `iss` claim (https://datatracker.ietf.org/doc/html/rfc7519#section-4.1.1) in the JWTs you send to the `/issue` and `/revoke` endpoints.

The requests to and responses from the Status List Service are the same, regardless of the status list type. You must make sure the Status List Service's responses are included in your credential correctly.

Access the Status List Service APIs

When you register as a status list client, we will work with you to grant access to our APIs and provide the API URLs.

Test your integration

The Status List Service operates an integration and a production environment. You must register a status list client for each environment.

We recommend that you use the integration environment to test your integration with the Status List Service before you move to production. The integration environment should not be used for publicly issued credentials.

This page was last reviewed on 22 October 2025. It needs to be reviewed again on 22 April 2026 .

API

/issue

post

An endpoint to request a new status index for a GOV.UK wallet credential. This API will use the issuer’s `/.well-known/jwks.json` endpoint to locate the signing keys to verify the signature.

Responses

Status	Description	Schema
200	A response that contains a status index and a uri to the status list where this newly issued `VALID` (binary 00) status index value can be found. `{ "index": 3, "uri": "https://crs.account.gov.uk/b/A671FED3E9AD" }`	IssueResponse
400	Bad request (invalid JWT, missing fields, wrong content-type, etc.) `{ "error": "BAD_REQUEST", "error_description": "No Type in Header" }`	Issue400ErrorResponse
401	Unauthorized (client not found) `{ "error": "UNAUTHORISED", "error_description": "No matching client found with ID: invalidClientId" }`	Issue401ErrorResponse
403	Forbidden (JWT signature verification failure) `{ "error": "FORBIDDEN", "error_description": "Failure verifying the signature of the jwt" }`	Issue403ErrorResponse
500	Internal server error	Issue500ErrorResponse

Status	Description	Schema
	`{ "error": "INTERNAL_SERVER_ERROR", "error_description": "Error receiving messages: ..." }`	

Schemas

IssueResponse

Name	Type	Required	Description	Schema
idx	number	true	The assigned status list index. This index position is unique within the status list identified by the URI	
uri	string	true	The URI of the status list that holds the issued credential. It is used in combination with the index to retrieve the status list entry	

Issue400ErrorResponse

Name	Type	Required	Description	Schema
error	string	true	An error code - must be `BAD_REQUEST` .	
error_description	string	false	A human-readable explanation of the error.	

Issue401ErrorResponse

Name	Type	Required	Description	Schema
error	string	true	An error code - must be `UNAUTHORISED` .	
error_description	string	false	A human-readable explanation of the error.	

Issue403ErrorResponse

Name	Type	Required	Description	Schema
error	string	true	An error code - must be `FORBIDDEN` .	
error_description	string	false	A human-readable explanation of the error.	

Issue500ErrorResponse

Name	Type	Required	Description	Schema
error	string	true	An error code - must be `INTERNAL_SERVER_ERROR` .	
error_description	string	false	A human-readable explanation of the error.	

This page was last reviewed on 22 October 2025. It needs to be reviewed again on 22 April 2026 .

Issue a status list entry

You can use the `/issue` endpoint to request a status list slot for a credential you want to issue. This endpoint lets you retrieve a new `uri` / `idx` pair from a status list, which represents a unique credential being issued to a user's wallet.

To use the `/issue` endpoint you must register as a credential issuer with the Status List service (/status-list/before-issuing-status-record), and you must send your request as a signed JSON Web Token (JWT).

When you request a new `uri` / `idx` pair for a credential on a status list, the Status List service validates your request. If this validation is successful, the Status List service will issue a new `uri` / `idx` pair on one of its status lists.

The Status List service will also return a uri and an index to where you can retrieve this credential's status. You must include this response in your issued credential.

For guidance on how to include the Status List Service's response in your credentials, see:

- the Token status list specification (https://datatracker.ietf.org/doc/draft-ietf-oauth-status-list/) for mdoc credentials
- the BitString status list specification (https://www.w3.org/TR/vc-bitstring-status-list/) for JWT-VC credentials

Technical details

The requests to and responses from the Status List Service are the same, regardless of the credential or status list type.

Endpoint URI

The URI path for the issue endpoint is `/issue` .

When you register as a credential issuer with the Status List service, you get access to the internal API. You must sign the request with your private key and share public keys on your `/.well-known/jwks.json` endpoint. This is used to verify the JWT.

Request format

The issue endpoint only accepts HTTP POST requests.

The request must include:

- `header` : you must provide the `Content-Type` header - the only valid value is `application/jwt`
- `request body (JWT)` : contains a signed JWT based on [RFC 7515 (https://datatracker.ietf.org/doc/html/rfc7515)](https://datatracker.ietf.org/doc/html/rfc7515), which must follow the requirements below

Status list JWT definition `/issue`

Header

The JSON Object Signing and Encryption (JOSE) header (based on [RFC-7515 (https://datatracker.ietf.org/doc/html/rfc7515#section-4)](https://datatracker.ietf.org/doc/html/rfc7515#section-4)) must contain the following header parameters:

```
{
  "typ": "JWT",
  "alg": "ES256",
  "kid": "1fb2c0f07f643b45cafeb53fb9d9eb34"
}
```

Parameter	Required or optional	Description
`typ`	Required	`typ` stands for 'type'. You must set this value to be `JWT` . This is the media type of the complete JWT.
`alg`	Required	`alg` stands for 'algorithm'. You must set this value to be `ES256` . This is the algorithm used to sign the JWT.
`kid`	Required	`kid` stands for 'key ID'. This key ID must be present in your hosted JWKS. This is used to validate the JSON web signature (JWS).

Payload

The JWT payload must contain the following claims:

```
{
  "iss": "exampleclientIDabcd123",
  "iat": 1686920170,
  "jti": "62b45850-4c5c-4696-983a-af66450301d4",
  "statusExpiry": 1734709493
}
```

Claim	Required or optional	Description
`iss`	Required	`iss` stands for 'issuer'. This is the [`clientId` of the credential issuer (/status-list/before-issuing-status-record)](/status-list/before-issuing-status-record) service generated when registering as a client. Make sure you are using the correct `clientId` for your environment - production or integration.
`iat`	Required	`iat` stands for 'issued at'. This is the UNIX timestamp when the request JWT was issued.
`jti`	Required	`jti` stands for 'JWT ID'. This provides a unique identifier for the JWT. The Status List Service will validate the format provided to make sure it is a lowercase UUID.
`statusExpiry`	Required	The point after which the status expires. After this date the credential will be removed from the status list. `statusExpiry` must be equal to or later than the issued credential's technical expiry time, known as the `validUntil` property. It may be useful to issue a status before the `validUntil` value is known. In this case, we recommend that you keep any resulting difference between `validUntil` and `statusExpiry` to a minimum. `statusExpiry` must be a number in seconds, formatted as a UNIX timestamp. The Status List Service does not support credentials lasting over 10 years.

Example Request

Below is an example of the post request signed and encoded as a JWT.

```
POST /issue HTTP/1.1
Host: <API.CRS.ACCOUNT.GOV.UK>
Content-Type: application/jwt
```

```
Accept: application/json

eyJhbGciOiJFUzI1NiIsInR5cCI6IkpXVCIsImtpZCI6IjgwODY4Nzk0LTM2MjYtNDNmOC05YTRk
```

Example Response

```
HTTP/1.1 200 OK
Content-Type: application/json

{
  "uri": "https://crs.account.gov.uk/b/A671FED3E9AD"
  "idx": 3,
}
```

Parameter	Description
`uri`	`uri` stands for 'uniform resource identifier'. This is the uri to the status list endpoint in which the new credential has been stored. This will be formatted as `crs.account.gov.uk` for the production environment, and `crs.integration.account.gov.uk` for the integration environment.
`idx`	`idx` stands for 'index'. This is the index at which the credential will be stored in the status list found on the uri.

This page was last reviewed on 22 October 2025. It needs to be reviewed again on 22 April 2026 .

API

/revoke

post

An endpoint used by GOV.UK Wallet credential issuers to revoke a status list index associated with a GOV.UK Wallet credential that they issued. This API will use the issuer's `/.well-known/jwks.json` endpoint to locate the signing keys to verify the signature.

Responses

Status	Description	Schema
202	Revocation processed successfully `{ "message": "Request processed for revocation", "revokedAt": 1734709493 }`	RevokeResponse
400	Bad request (invalid JWT, missing fields, wrong content-type, etc.) `{ "error": "BAD_REQUEST", "error_description": "No Type in Header" }`	Revoke400ErrorResponse
401	Unauthorized (client not found or client mismatch) `{ "error": "UNAUTHORISED", "error_description": "No matching client found with ID: invalidClientId" }`	Revoke401ErrorResponse
403	Forbidden (JWT signature verification failure or JWKS fetch failure) `{ "error": "FORBIDDEN", "error_description": "Failure verifying the signature of the jwt" }`	Revoke403ErrorResponse
404	Entry not found or list type mismatch	Revoke404ErrorResponse

Status	Description	Schema
	`{ "error": "NOT_FOUND", "error_description": "Entry not found in status list table" }`	
500	Internal server error `{ "error": "INTERNAL_SERVER_ERROR", "error_description": "Error processing revocation request" }`	Revoke500ErrorResponse

Schemas

RevokeResponse

Name	Type	Required	Description	Schema
message	string	true	Status message indicating the result	
revokedAt	number	true	Unix timestamp when the credential was revoked	

Revoke400ErrorResponse

Name	Type	Required	Description	Schema
error	string	true	An error code - must be `BAD_REQUEST` .	
error_description	string	false	A human-readable explanation of the error.	

Revoke401ErrorResponse

Name	Type	Required	Description	Schema
error	string	true	An error code - must be `UNAUTHORISED` .	
error_description	string	false	A human-readable explanation of the error.	

Revoke403ErrorResponse

Name	Type	Required	Description	Schema
error	string	true	An error code - must be `FORBIDDEN` .	

Name	Type	Required	Description	Schema
error_description	string	false	A human-readable explanation of the error.	

Revoke404ErrorResponse

Name	Type	Required	Description	Schema
error	string	true	An error code - must be `NOT_FOUND` .	
error_description	string	false	A human-readable explanation of the error.	

Revoke500ErrorResponse

Name	Type	Required	Description	Schema
error	string	true	An error code - must be `INTERNAL_SERVER_ERROR` .	
error_description	string	false	A human-readable explanation of the error.	

This page was last reviewed on 22 October 2025. It needs to be reviewed again on 22 April 2026 .

Revoke a credential

As a credential issuer, you can call the `/revoke` endpoint to revoke a credential you have previously issued. You cannot use it to revoke a credential issued by anyone else.

The Status List Service validates all calls to the `/revoke` endpoint to make sure that the caller has the correct rights to revoke the credential.

When you call the `/revoke` endpoint on an existing status, the status list records that status as revoked. This state change will be reflected in the published status list within a short timeframe. This process can not be reversed. There is more guidance on this in the statuslist endpoint page.

Technical details

The requests to and responses from the Status List Service are the same, regardless of the credential or status list type.

Endpoint URI

The URI path for the revoke credential endpoint is `/revoke` .

When you register as a credential issuer with the Status List Service, you get access to the internal API. You must sign the request with your private key and share public keys on your `/.well-known/jwks.json` endpoint. This is used to verify the JSON web token (JWT).

Request format

The revoke endpoint only accepts HTTP POST requests.

The request must include:

- `header` : you must provide the `Content-Type` header - the only valid value is `application/jwt`
- `request body` : contains a signed JWT based on RFC 7515 (https://datatracker.ietf.org/doc/html/rfc7515), which must follow the requirements below

Status list JWT definition /revoke

Header

The JSON Object Signing and Encryption (JOSE) header (based on RFC-7515 (https://datatracker.ietf.org/doc/html/rfc7515#section-4)) must contain the following header parameters:

```
{
  "typ": "JWT",
  "alg": "ES256",
  "kid": "499b46712489a805510bdf3e61e1f93d"
}
```

Parameter	Required or optional	Description
`typ`	Required	`typ` stands for 'type'. You must set this value to be `JWT` . This is the media type of the complete JWT.
`alg`	Required	`alg` stands for 'algorithm'. You must set this value to be `ES256` . This is the algorithm used to sign the JWT.
`kid`	Required	`kid` stands for 'key ID'. This key ID must be present in your hosted JWKS. This is used to validate the JSON web signature (JWS).

Payload

The JWT payload must contain the following claims:

```
{
  "iss": "asKWnsjeEJEWjjwSHsIksIksIhBe",
  "iat": 1686920170,
  "jti": "62b45850-4c5c-4696-983a-af66450301d4",
  "uri": "https://crs.account.gov.uk/t/3B0F3BD087A7",
  "idx": 3
}
```

Claim	Required or optional	Description
`iss`	Required	`iss` stands for 'issuer'. This is the [`clientId` of the credential issuer (/status-list/before-issuing-status-record)] service generated when registering as a client. Make sure you are using the correct `clientId` for your environment - production or integration.
`iat`	Required	`iat` stands for 'issued at'. This is the UNIX timestamp when the request JWT was issued.
`jti`	Required	`jti` stands for 'JWT ID'. This provides a unique identifier for the JWT. The Status List Service will validate the format provided to make sure it is a lowercase UUID.
`uri`	Required	`uri` stands for 'uniform resource identifier'. This is the uri of the status list that holds the status to revoke.
`idx`	Required	`idx` stands for 'index'. This is the index of the status to be revoked.

Your `uri` and `idx` must exactly match the response from the [`/issue` endpoint (/status-list/issue-status-list-entry)].

Example Request

```
POST /revoke HTTP/1.1
Host: <API.CRS.ACCOUNT.GOV.UK>
Content-Type: application/jwt

eyJhbGciOiJFUzI1NiIsInR5cCI6IkpXVCIsImtpZCI6IjgwODY4Nzk0LTM2MjYtNDNmOC05YTRk
```

Example Response

```
HTTP/1.1 202 ACCEPTED
```

This page was last reviewed on 22 October 2025. It needs to be reviewed again on 22 April 2026 .

Bitstring status list API

/b/{statusListIdentifier}

get

A public endpoint that returns a W3C Bitstring Status List credential JWT. The list that is returned depends on the list type for the GOV.UK Wallet credential issuer agreed at the time of registration. The response has a signed JWT in one of the two supported status list formats.

Parameters

Parameter	In	Type	Required	Description
statusListIdentifier	path	string	true	Unique name representing the specific status list

Responses

Status	Description	Schema
200	OK	
404	Not Found `{ "error": "NOT_FOUND", "error_description": "Status List not found for endpoint uri" }`	StatusList404ErrorResponse
500	Internal server error `{ "error": "INTERNAL_SERVER_ERROR", "error_description": "..." }`	StatusList500ErrorResponse

```
{
  "error": "NOT_FOUND",
  "error_description": "Status List not found for endpoint uri"
}
```

```
{
  "error": "INTERNAL_SERVER_ERROR",
  "error_description": "..."
}
```

Schemas

StatusList404ErrorResponse

Name	Type	Required	Description	Schema
error	string	true	An error code - must be `NOT_FOUND` .	
error_description	string	false	A human-readable explanation of the error.	

StatusList500ErrorResponse

Name	Type	Required	Description	Schema
error	string	true	An error code - must be `INTERNAL_SERVER_ERROR` .	
error_description	string	false	A human-readable explanation of the error.	

This page was last reviewed on 22 October 2025. It needs to be reviewed again on 22 April 2026 .

JSON Web Key Set (JWKS) API

The Status List Service's JSON Web Key Set (JWKS) endpoint. This provides a list of public keys which can be used to verify the signature on JWTs provided by the Status List Service.

/.well-known/jwks.json

get

A public endpoint that stores the JSON Web Key Set (JWKS) of public keys issued by a service. These keys can be used by client applications to verify the signature of a JSON Web Token (JWT).

Responses

Status	Description	Schema
200	Credential issuer's JWKS.	JwksResponse

```
{
  "keys": [
    {
      "kty": "EC",
      "use": "sig",
      "crv": "P-256",
      "kid": "5dcbee863b5d7cc30c9ba1f7393dacc6c16610782e4b6a191f94a7e8b1e1510f",
      "x": "6jCKX_QRrmTeEJi-uiwcYqu8BgMgl70g2pdAst24MPE=",
      "y": "icPzjbSk6apD_SNvQt8NWOPlPeGG4KYU55GfnARryoY=",
      "alg": "ES256"
    }
  ]
}
```

Schemas

JwksResponse

Name	Type	Required	Description	Schema
keys	array	true	A set of public keys, each in JSON Web Key (JWK) format.	Key

Key

Name	Type	Required	Description	Schema
kty	string	true	Key Type. The family of cryptographic algorithms used with the key.	
kid	string	true	Key ID. Unique identifier to match a specific key.	
crv	string	true	Curve. Cryptographic curve used with the key.	
x	string	true	The “x” coordinate for the elliptic curve point.	
y	string	true	The “y” coordinate for the elliptic curve point.	
alg	string	true	Algorithm. The cryptographic algorithm used with the key.	
use	string	true	The intended use of the key.	

This page was last reviewed on 22 October 2025. It needs to be reviewed again on 22 April 2026 .

Token status list API

/t/{statusListIdentifier}

get

A public endpoint that returns an IETF Token Status List JWT. The list that is returned depends on the list type for the GOV.UK Wallet credential issuer agreed at the time of registration. The response has a signed JWT in one of the two supported status list formats.

Parameters

Parameter	In	Type	Required	Description
statusListIdentifier	path	string	true	Unique name representing the specific status list

Responses

Status	Description	Schema
200	OK	
404	Not Found	StatusList404ErrorResponse
500	Internal server error	StatusList500ErrorResponse

404 Not Found:

```
{
  "error": "NOT_FOUND",
  "error_description": "Status List not found for endpoint uri"
}
```

500 Internal server error:

```
{
  "error": "INTERNAL_SERVER_ERROR",
  "error_description": "..."
}
```

Schemas

StatusList404ErrorResponse

Name	Type	Required	Description	Schema
error	string	true	An error code - must be `NOT_FOUND` .	
error_description	string	false	A human-readable explanation of the error.	

StatusList500ErrorResponse

Name	Type	Required	Description	Schema
error	string	true	An error code - must be `INTERNAL_SERVER_ERROR` .	
error_description	string	false	A human-readable explanation of the error.	

This page was last reviewed on 22 October 2025. It needs to be reviewed again on 22 April 2026 .

Check a credential’s status

The Status List Service hosts all status lists publicly. Each list is a signed JSON web token (JWT). You can verify the JWT’s signature by accessing the Status List Service’s JSON web key set (JWKS) hosted at `https://crs.account.gov.uk/.well-known/jwks.json` (production) or `https://crs.integration.account.gov.uk/.well-known/jwks.json` (integration).

There are two formats that the Status List Service supports: Bitstring status lists and Token status lists.

You can use the status list `uri` in the credential to check a credential’s status.

Each status at a specific index in the status list uses 2 bits. Each status index may contain one of the following bit combinations:

- `00` (VALID). Represents a valid credential
- `01` (INVALID). Represents a credential that has been permanently revoked (marked as invalid)
- `10` (NOT USED). Currently not used
- `11` (NOT USED). Currently not used

Bitstring status list

Status lists where the URI path begins with `/b/` are Bitstring status lists that follow the W3C Bitstring Status List specification (https://www.w3.org/TR/vc-bitstring-status-list/).

For consistency between the two different lists that the status list service publishes, the Status List Service uses the more complex implementation of Bitstring status lists (https://www.w3.org/TR/vc-bitstring-status-list/#example-example-statuslistcredential-using-more-complex-entries).

Technical details

Endpoint URI

The URI path for the Bitstring status list endpoint is `/b/{statusListIdentifier}` . It is presented as a GET request, where:

- `b` represents the type of status list: `BitstringStatusList`
- `statusListIdentifier` represents an ID for a specific status list

Bitstring status list request example

Below is an example of the `/b/{statusListIdentifier}` request:

```
GET /b/A671FED3E9AD HTTP/1.1
Host: crs.account.gov.uk
Accept: application/json
```

Request Response

Header

The JWT response header will contain the following:

```
{
  "alg": "ES256",
  "kid": "12",
  "typ": "vc+jwt"
}
```

Parameter	Description
`alg`	`alg` stands for ‘algorithm’. This value will be returned as `ES256` . This is the algorithm used to encode the JWT.
`kid`	`kid` stands for ‘key ID’. This key ID represents a key in the Status List Service’s JWKS which can be used to verify the JSON web signature (JWS).
`typ`	`typ` stands for ‘type’. This is the type of the status list. It is `vc+jwt` for Bitstring status lists.

Payload

The JWT response payload for a Bitstring status list will contain the following:

```
{
  "@context": [
    "https://www.w3.org/ns/credentials/v2",
    "https://www.w3.org/ns/credentials/examples/v2"
  ],
  "id": "https://crs.account.gov.uk/b/A671FED3E9AD",
  "type": [
    "VerifiableCredential",
    "BitstringStatusListCredential"
  ],
  "issuer": "https://crs.account.gov.uk/",
  "validFrom": "2025-10-01T14:00:00Z",
  "validUntil": "2025-10-08T14:00:00Z",
  "credentialSubject": {
    "id": "https://crs.account.gov.uk/b/A671FED3E9AD#list",
    "type": "BitstringStatusList",
    "statusSize": 2,
    "statusPurpose": "message",
    "statusMessage": [
      {
        "status": "0x0",
        "message": "VALID"
      },
      {
        "status": "0x1",
        "message": "INVALID"
      }
    ],
    "encodedList": "uH4sIAAAAAAAAA3MUBABJTAvCAgAAAA",
    "ttl": "3600"
  }
}
```

Parameter	Description
`id`	A unique URL that represents this status list.
`type`	The type of credential.

`issuer`	The URL of this status list credential's issuer.
`validFrom`	The earliest point in time at which the status list is valid.
`validUntil`	The latest point in time at which the status list is valid.
`credentialSubject`	The status list subject about which the claims below are made.
`credentialSubject.id`	A unique URI that represents this status list.
`credentialSubject.type`	The type of credential. This will be `BitstringStatusList` .
`credentialSubject.statusSize`	The size of the status list in bits.
`credentialSubject.statusPurpose`	The purpose of the status list, as described in statusMessage.
`credentialSubject.statusMessages`	This is an array of objects, which each contain a status and a message.
`credentialSubject.statusMessages.status`	This represents the status value in the status list. It is a hexadecimal string, and will be `"0x0"` or `"1x1"` .
`credentialSubject.statusMessages.message`	The status message representing the status value. This will be `"VALID"` or `"INVALID"` .
`credentialSubject.encodedList`	This is a multibase-encoded base64url (with no padding) representation of the GZIP-compressed bitstring values for the associated range of verifiable credential status values.

Token status list

Status lists where the URI path begins with `/t/` are Token status lists that follow the IETF Token Status List specification (https://www.ietf.org/archive/id/draft-ietf-oauth-status-list-12.html).

Technical details

Endpoint URI

The URI path for the Token status list endpoint is `/t/{statusListIdentifier}` . It is presented as a GET request, where:

- `t` represents the type of status list: `TokenStatusList`
- `statusListIdentifier` represents an ID for a specific status list

Token Status List Request Example

Below is an example of the `/t/{statusListIdentifier}` request:

```
GET /t/A671FED3E9AD HTTP/1.1
Host: crs.account.gov.uk
Accept: application/statuslist+jwt
```

Request Response

Header

The JWT response header will contain the following:

```
{
  "alg": "ES256",
  "kid": "12",
  "typ": "statuslist+jwt"
}
```

Parameter	Description
`alg`	`alg` stands for 'algorithm'. This value will be returned as `ES256` . This is the algorithm used to encode the JWT.
`kid`	`kid` stands for 'key ID'. This key ID represents a key in the Status List Service's JWKS which can be used to verify the JSON web signature (JWS).
`typ`	`typ` stands for 'type'. This is the type of the status list. This will be `statuslist+jwt` for a Token status list.

Payload

The JWT response payload for a Token status list will contain the following:

```
{
  "exp": 2291720170,
  "iat": 1686920170,
  "iss": "https://crs.account.gov.uk",
  "status_list": {
    "bits": 2,
    "lst": "eNpzdAEAAMgAhg"
  },
  "sub": "https://crs.account.gov.uk/b/A671FED3E9AD",
  "ttl": 43200
}
```

Parameter	Description
`exp`	`exp` stands for ‘expiry’. This is the expiry of the subject credential.
`iat`	`iat` stands for ‘issued at’. This is the timestamp the subject credential was originally issued at.
`iss`	`iss` stands for ‘issuer’. This is the URL of the credential issuer service operated by the organisation sharing the credential.
`status_list.bits`	The number of bits that represent a status.
`status_list.lst`	`lst` stands for ‘list’. This is an encoded version of this status list.
`sub`	`sub` stands for ‘subject’. This is the URI of the status list that was in the original HTTP request.
`ttl`	`ttl` stands for ‘time-to-live’. This is the lifetime of the cached version of this status list. Status lists are updated at regular and set intervals.

JSON Web Key Set (JWKS)

The JWKS endpoint exposes the Status List Service’s public cryptographic keys in JSON Web Key Set (JWKS) format. You can use a public key to verify the signature of a status list. This verification lets you make sure that the status list was published by the Status List Service and it has not been tampered with.

Technical details

Endpoint location

The JWKS is publicly accessible at the standardised location `/.well-known/jwks.json` on the Status List Service domain.

Response format

The endpoint must return a 200 OK HTTP status code and a valid JSON response that follows the JWKS specification defined in RFC 7517 (https://datatracker.ietf.org/doc/html/rfc7517). Each key within the JWKS is represented as a JSON Web Key (JWK) object. The JWKS usually contains only one key, but it can contain two keys during a key rotation overlap period.

The JWK for an elliptic curve public key based on the P-256 curve must include the following parameters:

Parameter	Definition
`kty`	The family of cryptographic algorithms used with the key. This must be `EC`.
`kid`	A unique identifier for a specific key within the set. This value will be referenced in the status list JWT header to show which key must be used for verification. This parameter is important for associating the correct public key with the status list being verified.
`crv`	The cryptographic curve used with the key. This must be `P-256`.
`x`	The “x” coordinate for the elliptic curve point.
`y`	The “y” coordinate for the elliptic curve point.
`alg`	The cryptographic algorithm used with the key. This must be `ES256`.
`use`	The intended use of the key. This must be `sig` to indicate the key can be used to verify the signature.

JWKS example

Below is an example of a JWKS containing one elliptic curve public key based on the P-256 curve:

```
{
  "keys": [
    {
      "kty": "EC",
      "use": "sig",
      "crv": "P-256",
      "kid": "5dcbee863b5d7cc30c9ba1f7393dacc6c16610782e4b6a191f94a7e8b1e151
      "x": "6jCKX_QRrmTeEJi-uiwcYqu8BgMgl70g2pdAst24MPE",
      "y": "icPzjbSk6apD_SNvQt8NWOPlPeGG4KYU55GfnARryoY",
      "alg": "ES256"
    }
  ]
}
```

This page was last reviewed on 22 October 2025. It needs to be reviewed again on 22 April 2026 .

Use Status List Service to change credential status

The Status List Service keeps a record of the status of all credentials issued to GOV.UK Wallet that include status list functionality.

The Status List Service provides:

- an issuer-facing API that credential issuers can use to issue a status list entry to a credential, and revoke it when required
- a set of public-facing credential status lists that anyone can use to check a credential's status

As a credential issuer, you can use the issuer-facing API to request a unique `uri` and `index` which can be embedded into a new credential when you issue it to GOV.UK Wallet. The `uri` and `index` refer to a unique slot in one of the Status List Service's status lists, where your credential's status will be stored. You can then set the status of this issued status list slot as `revoked` if you need to revoke a credential in future.

Anyone, including credential verifiers and holders, can use the public-facing status lists to get a single status list and check the status of a credential it contains.

In this section you can find information about:

- what to do before you issue a status record
- issuing a status record
- revoking a credential
- checking a credential's status

Use sample reference material

You can use our test harness and sample reference data to help you issue credentials to GOV.UK Wallet correctly.

Test harness

You can use the GOV.UK Wallet test harness (https://github.com/govuk-one-login/mobile-wallet-cri-test-harness) to validate your credential issuance implementation.

Follow the instructions in the README files to run the test scripts on your credential issuer.

Example credential issuer

You can use the credential issuer service in Java (https://github.com/govuk-one-login/mobile-wallet-example-credential-issuer) to see an example of a credential issuer service integrated with GOV.UK Wallet.

Follow the instructions in the README files to run an example of the credential issuance flow.

! **You should not use the sample reference material in a production environment.**

This page was last reviewed on 15 August 2025. It needs to be reviewed again on 15 February 2026 .

GOV.UK Wallet

GOV.UK Wallet lets users save and share digital versions of their government documents on their smartphone or device.

Issuing a document as a government service

If you work in a central government department, you can use GOV.UK Wallet to issue digital and verifiable versions of the physical documents you issue already as part of your service.

Before you can issue credentials to GOV.UK Wallet, you must already be using GOV.UK One Login (https://www.sign-in.service.gov.uk/) in your service to log in your users or verify their identities. You should discuss using GOV.UK Wallet with your GOV.UK One Login Engagement Manager, or contact us (/contact-us.html).

This documentation helps people such as developers and product managers of services that are using GOV.UK One Login to:

- learn how to issue digitally verifiable credentials to GOV.UK Wallet as part of their service journey
- understand how GOV.UK Wallet works for credential issuers (/issuing-credentials-to-wallet.html)
- prepare your service to use GOV.UK Wallet (/before-integrating.html)
- understand how organisations and individuals can receive and validate digital credentials you have issued when presented by the holder
- access sample reference material (/use-sample-reference-material.html) to validate your implementation

Using and consuming a document

You will be able to consume credentials from GOV.UK Wallet if you are:

- a public sector organisation (such as central government, the NHS, or a local authority)
- a certified Digital Verification Services (DVS) provider on the digital identity and attribute services register (https://www.digital-identity-services-register.service.gov.uk/)

This documentation helps developers and service teams in these organisations to:

- understand how GOV.UK wallet works for consuming credentials
- understand the open standards that will allow their service to consume and verify credentials and attributes (/consuming-and-verifying-credentials/supported-protocols.html)

This documentation will be updated as new information and features become available.

Digital verification services should read the guidance about using GOV.UK Wallet in the digital identity sector (https://gov.uk/guidance/using-govuk-wallet-in-the-digital-identity-sector) to understand the different models available.

Documentation updates

These are the most recent changes to the GOV.UK Wallet technical documentation.

Publication date	Update
Oct 22 2025	Add new 'Use Status List Service to change credential status' section to add guidance on using the Status List Service. Includes information about the `/issue` and `/revoke` APIs, and guidance on checking a credential's status.
Aug 15 2025	Add new 'Use sample reference material (/use-sample-reference-material.html)' page to link to reference implementation and test harness resources.
Jul 10 2025	Add `exp` as a required claim in the GOV.UK One Login access token.
Jul 10 2025	Clarify requirements for date fields in JWT VC credentials: Remove Optional claims `exp` and `nbf` . Update guidance on credential expiration. Specify `validFrom` and `validUntil` must be expressed in ISO 8601 format with seconds. Update `validUntil` to be a Required claim.
Jul 03 2025	Remove `credentialSubject` , `display` and `key_attestations_required` parameters, and rename `cryptographic_suites_supported` to `credential_signing_alg_values_supported` in Metadata API.
Jul 02 2025	Update the second value for `@context` property in DID Document.
Jun 26 2025	Update guidance on credential photos.

Jun 19 2025	Update proof of possession `iat` claim description. Add `Cache-Control` with value of `no-store` to Notification and Credential APIs. Add 400 Bad Request response to Credential API.
May 29 2025	Add a 401 Unauthorized response to the Credential API.
May 28 2025	Update the Notification API 401 Unauthorized response format to comply with RFC 6750 (Bearer Token Usage).
May 14 2025	Add Consuming and verifying credentials section for credential consumer and verifier documentation. Update GOV.UK Wallet (/) page with new introductory content for new section.
Apr 23 2025	Restructure 'What GOV.UK Wallet does, and how it works' into a 'GOV.UK Wallet (/)' introductory page and a 'How GOV.UK Wallet works (/issuing-credentials-to-wallet.html)' page. Remove 'Understand GOV.UK Wallet's credential exchange flow' page, and move issuance flow diagram and steps into the new 'How GOV.UK Wallet works (/issuing-credentials-to-wallet.html)' page.
Apr 16 2025	Begin recording changes in a changelog.

This page was last reviewed on 14 May 2025. It needs to be reviewed again on 14 November 2025 .