

Inland Revenue

## Identity and Access Services build pack

**Date:** 07/11/2019  
**Version:** 2.5

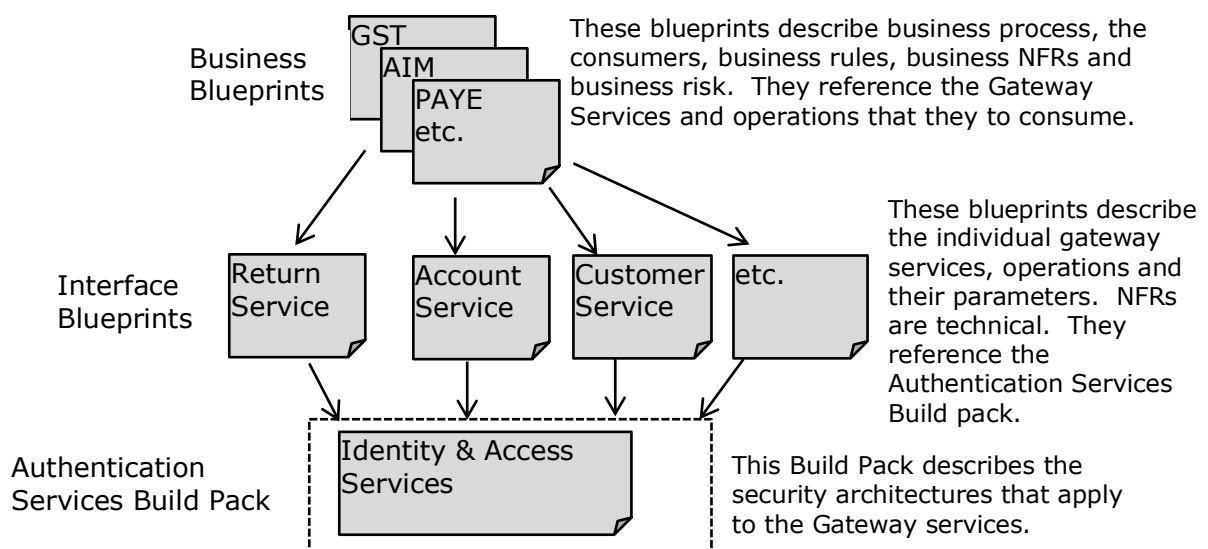
UNCLASSIFIED

## About this Document

This document is intended to provide Service Providers with the technical detail required to consume the Identity and Access services offered by Inland Revenue.

This is a technical document that supports the on boarding processes of an end to end solution. The associated on-boarding document(s) describe the end-to-end business level solution, of which this build pack is part. This document describes the architecture of the technical solution, the interaction with other build packs, schemas and endpoints. Also included are sample payloads to use in non-production environments.

It is part of a 3-tier hierarchy of documents that are depicted in the following diagram:



## Contents

<b>1 Overview.....</b>	<b>5</b>
1.1 This solution .....	5
1.1.1 Organisational Authentication and Authorisation. ....	6
1.1.2 End-User Authentication and Authorisation.....	6
1.2 Intended audience.....	6
1.3 Information IR will provide Service Providers .....	7
1.3.1 Token Auth (Cloud or Native) .....	7
1.3.2 SSH keys .....	7
1.4 Information Service Providers must provide IR.....	7
1.4.1 Service Provider Information .....	7
1.4.2 Token Auth (Cloud or Native) .....	7
1.4.3 SSH keys .....	7
<b>2 Description of the IR Authentication Mechanisms.....</b>	<b>8</b>
2.1 IR Token Auth Implementation using OAuth 2.0 .....	8
2.1.1 High Level View of OAuth 2.0.....	8
2.1.2 Authorisation Services .....	9
2.1.3 Authorisation Service.....	9
2.1.4 Refresh Token Service .....	14
2.1.5 Validate Token Service.....	15
2.1.6 Revoke Token Service.....	17
2.1.7 Security Considerations .....	18
2.1.8 Endpoints.....	18
<b>2.2 Native Application Token Auth .....</b>	<b>18</b>
<b>2.3 SSH Keys.....</b>	<b>19</b>
<b>2.4 M2M using Client Signed JWT.....</b>	<b>20</b>
<b>3 Appendix A – Sample payloads .....</b>	<b>24</b>
3.1 Request Authorisation Code.....	24
3.1.1 Request .....	24
3.2 Authorisation Code response.....	24
3.2.1 Success Response – Authorisation Code sent.....	24
3.3 Request Access token .....	24
3.3.1 Exchange Authorisation Code for oAuth Access Token.....	24
3.3.2 Success Response – Access Token sent.....	25
3.4 Request Refresh token .....	25
3.4.1 Refresh request .....	25
3.4.2 Refresh token reply .....	26
3.4.3 Error Response. ....	27
3.5 Validate token request .....	27
3.5.1 Validate token request .....	27

---

3.5.2	Validate token reply .....	27
3.5.3	Validate token Error Response. ....	28
3.6	Revoke token request .....	28
3.6.1	Revoke token request.....	28
3.6.2	Revoke token reply .....	28
3.6.3	Revoke token Error Response. ....	28
<b>4</b>	<b>Appendix B – Glossary .....</b>	<b>29</b>
<b>5</b>	<b>Appendix C—Client Signed JWT examples .....</b>	<b>31</b>
<b>6</b>	<b>Appendix D – Deprecated, Aged and Changing Standards.....</b>	<b>33</b>
<b>7</b>	<b>Appendix E—Change log.....</b>	<b>34</b>

## 1 Overview

### 1.1 This solution

Inland Revenue (IR) is establishing a new set of Identity and Access services. These will provide Service Providers with authentication and authorisation mechanisms for accessing IR's new Gateway Services.

These specifications within this document refer to specific cryptographic standards (e.g. security protocols, key lengths, signing and hashing algorithms etc) that are applicable at the time of writing. IR reserves the right to upgrade these in response to factors such as external threats, industry standards and changes to NZISM. IR's partners will be consulted, via IR's relationship managers, about upgrades to these standards.

There are two distinct types of entity for which IR provides mechanisms for authentication and authorisation:

1. Organisations
2. End Users

The mechanisms are as per the diagram below:

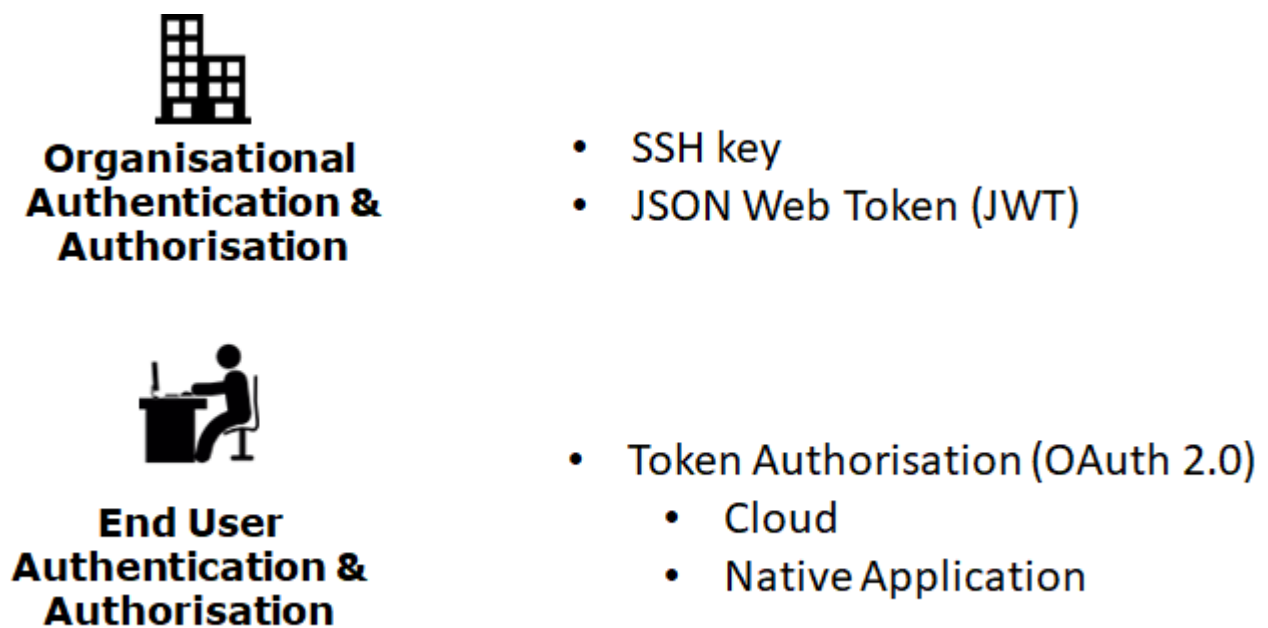


Figure 1 Types of Entity and Authentication & Authorisation mechanisms

### 1.1.1 Organisational Authentication and Authorisation.

The table below details the current and future mechanisms IR provide to authenticate and authorise an organisation for Machine to Machine (M2M) communication.

Authorisation Mechanism	Uses
<b>SSH keys</b>	This mechanism is used in SFTP file transfers to identify the organisations sending/receiving files.  SSH keys need to be exchanged to authenticate both parties.
<b>JSON Web Token (JWT) Using JSON Web Signature (JWS) for digital signatures with RSA/ECDSA</b>	Used to sign messages in order to identify the service provider or a customer of a service provider to IR.  This will be used by IR as a M2M mechanism.

**Table 1 Organisational Authentication and Authorisation Methods**

### 1.1.2 End-User Authentication and Authorisation.

The OAuth 2.0 process is used to authenticate end-users using their IR user ID and password and grant 3<sup>rd</sup> party software consent to access their information.

The OAuth 2.0 mechanism to be used by a service provider is based upon the nature of the client application the end-user will be using.

Authorisation Mechanism	Uses
<b>Cloud application Token Auth OAuth2.0 based</b>	Use when the client application is a web-enabled cloud based application. It requires an online user to enter their myIR user ID and password to grant the application access to their IR information.
<b>Native application Token Auth OAuth2.0 based</b>	Use when the client application is a desktop or other native application. It also requires an online user to enter their myIR user ID and password to grant the application access to their IR information.  See section 2.2 Native Application Token Auth below for details

**Table 2: End-User Authentication and Authorisation Methods**

## 1.2 Intended audience

This build pack and the resources to which it refers are primarily focused on the needs of Software Developers' technical teams and development staff.

The reader is assumed to have a suitable level of technical knowledge in order to comprehend the information provided. A range of technical terms and abbreviations are used throughout this document, and while most of these will be understood by the intended readers, a glossary is provided at the end.

---

This document is not intended for use by managerial staff or those with a purely business focus.

### **1.3 Information IR will provide Service Providers**

#### **1.3.1 Token Auth (Cloud or Native)**

1. URLs and parameters for invoking the Authentication services.
2. Client ID (agreed with service consumer)
3. Client secret (used in step 2 in section 2.1.2)

#### **1.3.2 SSH keys**

1. SSH keys for SFTP.
2. PGP public keys if used for payload encryption and signing

### **1.4 Information Service Providers must provide IR**

#### **1.4.1 Service Provider Information**

1. Full business name
2. Client ID (agreed with IR and used in requests to IR))
3. Key Contact
4. Email of key contact or delegate
5. Mobile Phone number (SMS may be used for some information)
6. IP addresses Service Providers will use for test instances for IR firewall whitelisting.

#### **1.4.2 Token Auth (Cloud or Native)**

1. Redirect URI for Authorisation code and Authentication token.

#### **1.4.3 SSH keys**

1. SSH public keys if using SFTP.
2. Their own Public Keys if using PGP.

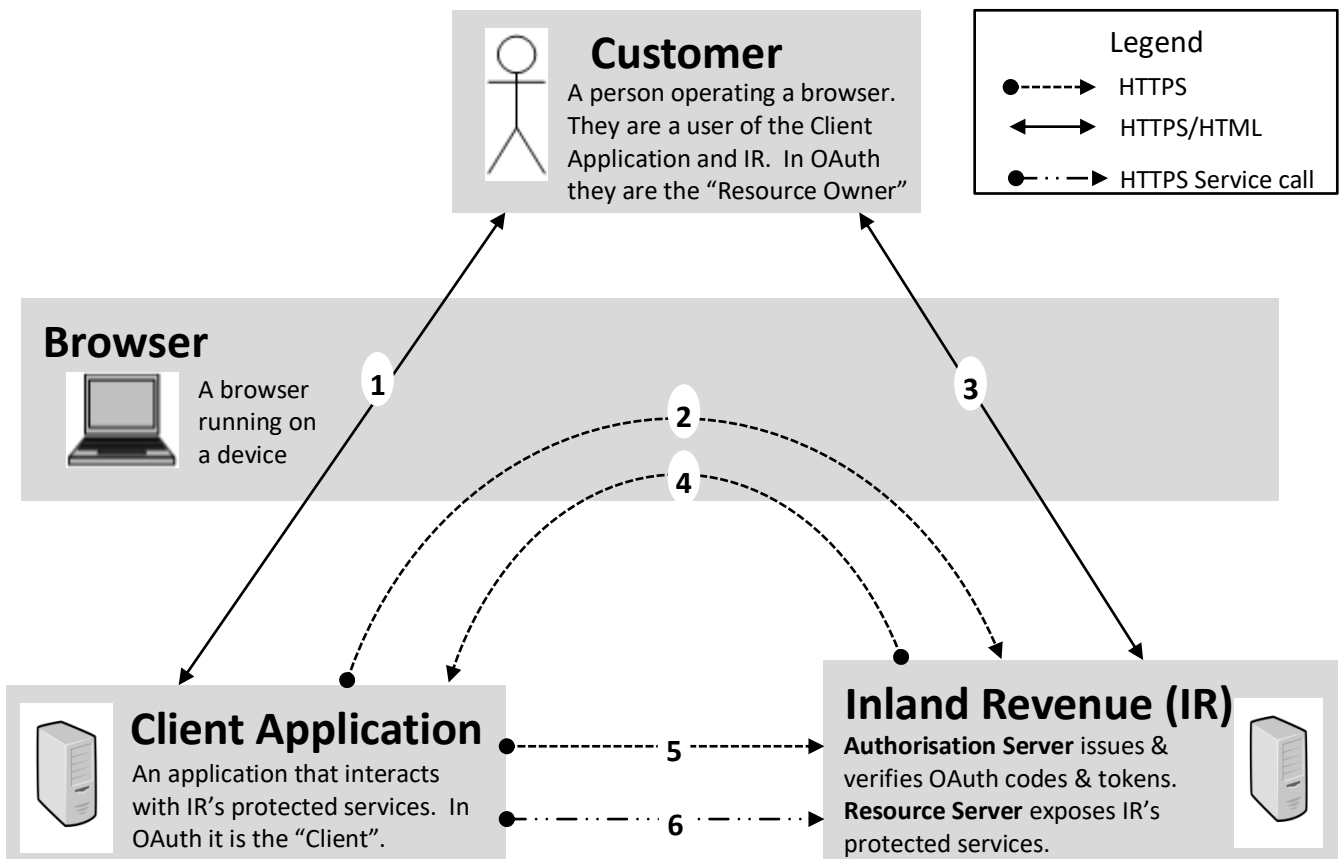
## 2 Description of the IR Authentication Mechanisms

### 2.1 IR Token Auth Implementation using OAuth 2.0

This section describes the IR OAuth 2.0 implementation. This high-level description covers the services offered by IR's implementation of OAuth Authorisation services including both Cloud and Native application usage.

#### 2.1.1 High Level View of OAuth 2.0

For OAuth 2 the following diagram depicts the high level end-to-end view of the components and the interactions between them:



1. The User is interacting with the Client Application. They access a protected service provided by IR (e.g. to file a return, retrieve a balance etc.)
2. The Client Application invokes the Authorisation API to get an authorisation code, the user's browser is redirected to IR's logon page.
3. IR prompts the User to logon, they are authenticated. On first use the User must also supply their consent for the Client Application to access IR on their behalf. IR issues the Authorisation Code.
4. The Authorisation Code is returned to the Client Application (via the browser).
5. The Client Application invokes IR's Token service to redeem the Authorisation Code for an OAuth Access Token. It has a finite time to live.
6. The Client Application can then invoke IR's protected services (e.g. to file a return etc.) supplying the OAuth Access Token in the header. The OAuth Access Token can be used for multiple invocations until it expires.



Inland Revenue's implementation of the OAuth 2 standard conforms to the Authorisation Code Grant flow described in section 4.1 of RFC 6749 ( <https://tools.ietf.org/html/rfc6749> ).

### 2.1.2 Authorisation Services

This section describes the services offered by IR through the Authorisation services, including:

- a) Authorisation Service (via OAuth 2.0)
- b) Refresh Token Service
- c) Validate Token Service
- d) Revoke Token Service

The following sections will describe the steps and service calls required to integrate with the IR implementation.

### 2.1.3 Authorisation Service

This section describes the steps and service calls required when using the IR implementation of OAuth 2.0. These are the same for both Cloud and Native App usage.

#### 2.1.3.1 Customer accesses the Client Application (Step1)

The Customer accesses the Client application and triggers the need for it to consume one of Inland Revenue's protected services (e.g. to retrieve an account balance, to file a return etc.).

#### 2.1.3.2 Request Authorisation Code (Step 2)

The customer's browser is redirected to the IR Authorisation service to authenticate the user and confirm scope using the GET method described below (example is for one IR Test environment – see 2.1.8 below for all endpoints):

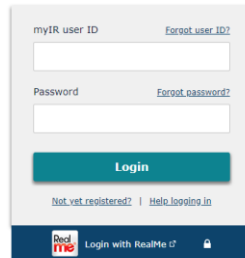
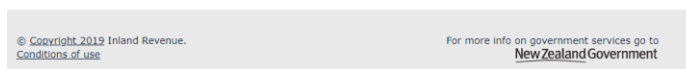
```
https://test4.services.ird.govt.nz/ms_oauth/oauth2/endpoints/oauthservice/authorize?
response_type=code
&client_id=IdOfCompanyUsingTheAPI
&redirect_uri=http://client.example.com/return
&scope=MYIR.Services
&state=xyz
&logout=true
```

Name	Description	Required	Valid Values
<b>response_type</b>	Response type requested	Required	"code"
<b>client_id</b>	The agreed Client identifier established at registration. Inland Revenue maintains this list of values.	Required	client_id

<b>redirect_uri</b>	The Client application's redirect URI to which the Authorisation Code is returned.	Required	Business Partner defined & agreed with IR
<b>scope</b>	Use space-separated values. Define scope values in the configuration/scope registry.	Required	"MYIR.Services"
<b>state</b>	<p>A value used by the Business Partner to maintain state between the request and callback.</p> <p>The use of the state parameter is strongly recommended for additional security. The value should relate to the logged-in user, and allows the Client (software provider) to confirm that the request being received at their Redirect URI is valid. See <a href="https://tools.ietf.org/html/rfc6749">https://tools.ietf.org/html/rfc6749</a> for more information (Section 12.12).</p>	Recommended	<p>Business Partner defined.</p> <p>Must be &lt; 200 characters.</p> <p>Allowed Characters: a-z A-Z 0-9 - . ? , : / \ + = \$ #</p> <p>Not space (%20, (ASCII 32)</p>
<b>logout</b>	<p>To enable a user to use a different myIR login, setting the value to "true" will force the logout of any user currently logged in. This avoids having to close a browser or wait 15 minutes to timeout an existing user login.</p> <p>This will force the re-display of the login page and request the input of a new myIR account to login.</p> <p>The value of "false" or absence of this parameter will not force logout.</p>	Optional	<p>"true"</p> <p>"false"</p>

#### 2.1.3.3 Login with myIR Credentials (step 3)

During this step the customer may be required to authenticate, and, if this is required, will be redirected to the following myIR logon screen. For OAuth 2.0 for Native Apps this authorisation request is in an external user-agent (typically the browser).

If the software provider chooses (not generally recommended as this page may change from time to time) to present this page within a frame the minimum recommended size in pixels is 600w x 500h.

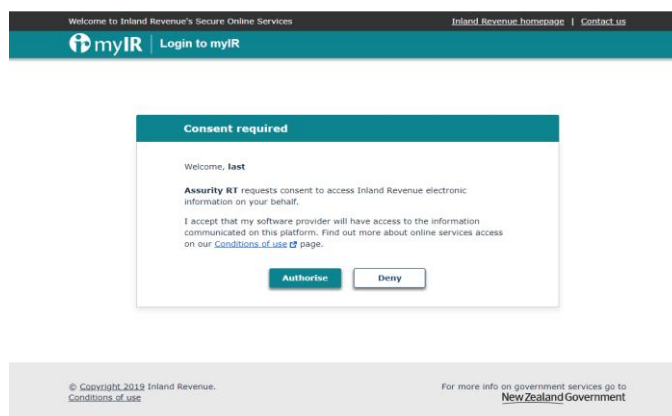
Note the customer must already have an IR Online Services credential, i.e. a myIR logon.

Invalid User ID or password will return a HTTP:200

#### 2.1.3.4 Consent Page (step 4)

The following consent screen will be presented to the user the very first time they authenticate with their IR credentials via Step 3 above. This screen will not be presented for subsequent logons.

The user must click Authorise for the process to proceed. If they click deny then the process stops and the authorisation server will not respond with an authorisation code as described in Step 5 below.



When testing, it is recommended to revoke consent in order to ensure that the Service Provider software is correctly managing the Consent Management page being presented and Authorisation given.

The end-points below give this ability:

[https://test4.services.ird.govt.nz/ms\\_oauth/oauth2/ui/oauthservice/showrevokeconsent](https://test4.services.ird.govt.nz/ms_oauth/oauth2/ui/oauthservice/showrevokeconsent)

#### 2.1.3.5 Respond with Authorisation Token (Step 5)

If successful, the authorisation service will respond with the Authorisation Code to the Business Partner redirect\_uri as described below:

```
https://client.example.com/return?code=eyJhbG...rWWk8hbs_o6uY&state=xyz
```

Name	Description	Valid Values
<b>Code</b>	Authorisation code value - Includes the following: <ul style="list-style-type: none"> <li>Expiry</li> <li>Client_id</li> <li>Redirect_uri</li> </ul>	Encrypted string ~1000 characters
<b>State</b>	Business Partner defined state. The use of the state parameter is strongly recommended for additional security. The value should relate to the logged-in user, and allows the Client (software provider) to confirm that the request being received at their Redirect URI is valid. See <a href="https://tools.ietf.org/html/rfc6749">https://tools.ietf.org/html/rfc6749</a> for more information (Section 12.12).	Business Partner defined.  Must be < 200 characters.  Allowed Characters: a-z A-Z 0-9 - . ? , : / \ + = \$ #  Not space (%20, (ASCII 32)

If not successful an error is sent with a HTTP code and a JSON response containing the error code and description.

Errors are:

HTTP code	Error Type	Description
<b>400</b>	<b>invalid_redirect_uri</b>	Redirect URI mismatch with Business Partner app
<b>401</b>	<b>Invalid client ID</b>	API Key contains invalid information
	<b>invalid_client</b>	Business partner identifier invalid
	<b>invalid_scope</b>	Requested scope is invalid, unknown, or malformed
	<b>server_error</b>	Authentication - Runtime processing error
	<b>access_denied</b>	End-user denied authorisation
<b>500</b>	<b>InternalError</b>	An internal and unexpected error occurred
<b>504</b>	<b>GatewayError</b>	Gateway did not receive a timely response from the upstream server

#### 2.1.3.6 Request Authorisation Token (Step 6)

Once an Authorisation Code has been returned to the Client application it must be exchanged for an OAuth Access Token by doing an HTTPS Post to the Token Service as follows:

(example is for one IR Test environment – see 2.1.8 below for all endpoints)

`https://test4.services.ird.govt.nz/ms_oauth/oauth2/endpoints/oauthservice/tokens`

`<form>`

`redirect_uri=https://client.example.com:17001/return`

`&grant_type=authorization_code`

`&code=eyJhbG...rWWk8hbs_o6uY`

`</form>`

With the values of:

Name	Description	Required	Valid Values
<b>redirect_uri</b>	The Client application's redirect URI for the Authorisation Token.	Required	Business Partner defined & agreed with IR
<b>grant_type</b>	The grant type is authorization_code	Required	authorization_code
<b>code</b>	Authorisation Code as supplied by the authorisation service in step 4	Required	Encrypted string ~1000 characters

The header fields must contain the Client ID and Client secret and content type:

Authorization: Basic NTQzMjFpZ...ZWxjb2l1MQ==

Content-Type: application/x-www-form-urlencoded;charset=UTF-8

With the values of:

Name	Description	Required	Valid Values
<b>Authorization</b>	"Basic" + Base64 encoded (ClientID + ":" + Client Secret)	Required	Base64 encoded string
<b>Content-Type</b>	Content type	Required	application/x-www-form-urlencoded;charset=UTF-8

The response contains the OAuth Access Token – this must be passed on subsequent service calls.

If the client is registered for Refresh Tokens, this will also be given at the point, see Section b) Refresh Token Service.

If not successful, an error is sent with a HTTP code and a JSON response containing the error code and description.

Errors:

HTTP code	Error Type	Description
<b>400</b>	<b>invalid_redirect_uri</b>	Redirect URI mismatch with Business Partner app
<b>401</b>	<b>Invalid client ID</b>	API Key contains invalid information
	<b>Invalid client_id or client_secret</b>	API Secret contains invalid information
	<b>invalid_client</b>	Business partner identifier invalid
	<b>invalid_scope</b>	Requested scope is invalid, unknown, or malformed
	<b>server_error</b>	Authentication - Runtime processing error
	<b>access_denied</b>	End-user denied authorisation
<b>500</b>	<b>InternalError</b>	An internal and unexpected error occurred
<b>504</b>	<b>GatewayError</b>	Gateway did not receive a timely response from the upstream server

#### 2.1.4 Refresh Token Service

In normal use it is expected that the Access Token will be re-used while it is active and the client will make several calls using this Access Token. If the Access Token has expired, the client can request another Access Token using the Refresh Token and client credentials.

The Refresh Token is granted as part of the Cloud Authorisation Service, the client will need to hold this and when required can be exchanged for a new Access and Refresh Token granting longer term use.

The API call to exchange a Refresh Token for an Access token takes the form:

```
https://test4.services.ird.govt.nz/ms_oauth/oauth2/endpoints/oauthservice/tokens
```

```
<form>
```

```
    grant_type=refresh_token
```

```
    refresh_token=<refresh-token-value>
```

```
</form>
```

With the values of:

Name	Description	Required	Valid Values
<b>grant_type</b>	The grant type is required and takes the value refresh_token	Required	refresh_token

<b>refresh_token</b>	The refresh_token field contains the Refresh Token	Required	<Refresh Token>
----------------------	--	----------	-----------------

The header fields must contain the Client ID and Client secret and content type:

Authorization: Basic NTQzMjFpZ...ZWxjb21lMQ==  
 Content-Type: application/x-www-form-urlencoded;charset=UTF-8

With the values of:

Name	Description	Required	Valid Values
<b>Authorization</b>	"Basic " + Base64 encoded (ClientID + ":" + Client Secret)	Required	Base64 encoded string
<b>Content-Type</b>	Content type	Required	application/x-www-form-urlencoded;charset=UTF-8

The normal expected response if the Access Token is valid is a new Access and Refresh Token. Note that the refresh token does not expire. A new Access Token can be requested using the Refresh Token at any time after it was obtained as long as the Customer consent is still valid.

#### 2.1.5 Validate Token Service

This service will validate an Access Token. The following example is for one IR Test environment – see 2.1.8 below for all endpoints.

https://test4.services.ird.govt.nz/ms\_oauth/oauth2/endpoints/oauthservice/tokens

<form>

grant\_type= oracle-idm:/oauth/grant-type/resource-access-token/jwt

oracle\_token\_action=validate

scope= MYIR.Services

assertion=<token-value>

oracle\_token\_attrs\_retrieval=prn exp

</form>

With the values of:

Name	Description	Required	Valid Values
<b>grant_type</b>	The grant type is required and takes the value oracle-idm:/oauth/grant-type/resource-access-token/jwt	Required	oracle-idm:/oauth/grant-type/resource-access-token/jwt
<b>Oracle_token_action</b>	The oracle token action is required and takes the value validate	Required	validate
<b>Scope</b>	Scope is required and takes the value MYIR.services	Required	MYIR.services
<b>assertion</b>	The assertion field contains the Access Token	Required	<Access Token>
<b>oracle_token_attrs_retrieval</b>	This field details the list of fields to retrieve.  Prn is user ID  Exp is expiry time in unix epoch	Required	prn exp

The header fields must contain the Client ID and Client secret and content type:

Authorization: Basic NTQzMjFpZ...ZWxjb21lMQ==  
 Content-Type: application/x-www-form-urlencoded; charset=UTF-8

With the values of:

Name	Description	Required	Valid Values
<b>Authorization</b>	"Basic " + Base64 encoded (ClientID + ":" + Client Secret)	Required	Base64 encoded string
<b>Content-Type</b>	Content type	Required	application/x-www-form-urlencoded; charset=UTF-8



### 2.1.6 Revoke Token Service

A token revoke process is available that will allow the client to revoke the acquired access token or refresh token.

Customers may also revoke their consent through a function in IR Online Services, "myIR". The following example is for one IR Test environment – see 2.1.8 below for all endpoints.

[https://test4.services.ird.govt.nz/ms\\_oauth/oauth2/endpoints/oauthservice/tokens](https://test4.services.ird.govt.nz/ms_oauth/oauth2/endpoints/oauthservice/tokens)

<form>

grant\_type= oracle-idm:/oauth/grant-type/resource-access-token/jwt

oracle\_token\_action=delete

assertion=<token-value>

</form>

With the values of:

Name	Description	Required	Valid Values
<b>grant_type</b>	The grant type is required and takes the value oracle-idm:/oauth/grant-type/resource-access-token/jwt	Required	oracle-idm:/oauth/grant-type/resource-access-token/jwt
<b>Oracle_token_action</b>	The oracle token action is required and takes the value delete	Required	delete
<b>assertion</b>	The assertion field contains the Access Token or Refresh token	Required	<Access/Refresh Token>

The header fields must contain the Client ID and Client secret and content type:

Authorization: Basic NTQzMjFpZ...ZWxjb2lIMQ==

Content-Type: application/x-www-form-urlencoded;charset=UTF-8

With the values of:

Name	Description	Required	Valid Values
<b>Authorization</b>	"Basic" + Base64 encoded (ClientID + ":" + Client Secret)	Required	Base64 encoded string

<b>Content-Type</b>	Content type	Required	application/x-www-form-urlencoded
---------------------	--------------	----------	-----------------------------------

### 2.1.7 Security Considerations

Protecting the integrity of the Client Secret is an important requirement for providers, the exact implementation is left to the provider but it must not be stored in plain text either in the web, mobile, or desktop application. Our preference is for this to be stored on a back-end server and made available to the Business Partner application.

If a Client Secret is compromised it shall be invalidated and a new secret issued.

The OAuth Authorisation Code has a time to live of 15 minutes.

The OAuth Access Token has a time to live of 8 hours.

### 2.1.8 Endpoints

Endpoints for the token-based Authentication and Authorisation Service are as follows:

#### Test Environments:

Code: [https://test4.services.ird.govt.nz/ms\\_oauth/oauth2/endpoints/oauthservice/authorize](https://test4.services.ird.govt.nz/ms_oauth/oauth2/endpoints/oauthservice/authorize)

Token: [https://test4.services.ird.govt.nz/ms\\_oauth/oauth2/endpoints/oauthservice/tokens](https://test4.services.ird.govt.nz/ms_oauth/oauth2/endpoints/oauthservice/tokens)

#### Production Environment

Code: [https://services.ird.govt.nz/ms\\_oauth/oauth2/endpoints/oauthservice/authorize](https://services.ird.govt.nz/ms_oauth/oauth2/endpoints/oauthservice/authorize)

Token: [https://services.ird.govt.nz/ms\\_oauth/oauth2/endpoints/oauthservice/tokens](https://services.ird.govt.nz/ms_oauth/oauth2/endpoints/oauthservice/tokens)

## 2.2 Native Application Token Auth

The OAuth 2.0 transaction flow described above will not change for Service Providers running Native Applications. A Native Application (commonly called a Native App) is an application that is installed by the user to their device or a desktop application, as distinct from a web app that runs in the browser context only.

IR has a duty to protect customer information in transactions. In order to meet this obligation, the security and traffic controls for Native App endpoints are more stringent. This is due to the more public nature of these endpoints and the inability to identify the caller by an X.509 certificate as per the cloud connection.

To further this end, IR is adopting [RFC8252 OAuth 2.0 for Native Apps](#)

Your attention is drawn to the best practice description in section 1 of using an external browser for OAuth by Native Apps.

IR will be providing separate endpoints where using gateway services with applications using Native App OAuth applications. Mutual TLS will not be used for these endpoints so no X.509 client cert will be required but server-side TLS will still be used.

Please refer to the Build Pack for each respective service for further details.

Note the endpoints for the Native App OAuth calls will not be changing from those outlined in Section 2.1.7 above.

A Client ID and secret will still be issued, the Client ID will need to show both the software vendor and application name e.g. SmartSoftware\_SmartPay or SmartSoftware\_SmartSoftware.

At registration IR will note the OAuth user ID is type 'Native App' and not 'Normal'. The refresh token capability will not be offered for Native App OAuth tokens.

Special redirect will be allowed and the three redirection schemes in section 7 of RFC 8252 will be supported. IR does not limit or put a preference on any of these.

If considering 7.3 then attention is drawn to recommendations available on the internet regarding the use of 127.0.0.1 rather than local host. See [ietf.org](http://ietf.org).

7.3 of the RFC 8252 also states:

*The authorization server MUST allow any port to be specified at the time of the request for loopback IP redirect URIs, to accommodate clients that obtain an available ephemeral port from the operating system at the time of the request.*

This is not currently supported by the IR. As a work-around IR will be able to provide a selection of five or so port numbers to be agreed with the Service Provider and registered in IR's back end OAuth system. Please discuss with your IR Account Manager

RFC 8252 section 6 describes Proof Key for Code Exchange (PKCE). Section 8.1 describes this is a proof-of-possession extension to OAuth 2.0.

Service Providers should note IR intends to implement the PKCE protocol at some time in the near future.

## 2.3 SSH Keys

This authentication and authorisation mechanism is used in SFTP file transfers to identify the respective organisations sending/receiving files, in this case IR and the Service Provider.

SSH Key authentication and authorisation will be used for file transfers in which SFTP 3.0 is used. This version of SFTP requires the use of SSH version 2.0.

The public key algorithm for SSH authentication keys must be ECDSA with a minimum field/key size of at least 160 bits.

Certain FTP file transfers will also require payload encryption and signing to ensure that once a file is transferred to an endpoint only an authorised party can interpret it. This is optional and the need for this will be identified in the respective On-boarding pack for a file transfer.

The obligation to encrypt the transferred files is defined in the NZISM under the classification privacy ratings. These are based upon the sensitivity of the data, the harm that might be caused if the data is disclosed, along with considerations such as the volumes being transferred. An assessment of these factors is required when implementing new file transfers and when adding content or changing the parties to any existing file transfers.

For files from IRD to partners PGP encryption will use Advanced Encryption Standard (AES) with a 256-bit key and the PGP hashing will use Secure Hash Algorithm (SHA) SHA-256.

Note the IR reserves the right to upgrade cryptographic standards (e.g. algorithms and key lengths) in response to factors such as external threats, industry standards and changes to NZISM. IR's partners will be consulted, via IR's relationship managers, about changes of this nature.

Currently IR has the ability to push and pull files being exchanged, but the Service Provider always hosts the FTP server.

---

---

## 2.4 M2M using Client Signed JWT

Inland Revenue's machine-to-machine (M2M) authentication mechanism will use Client Signed JSON Web Tokens (JWT).

When applying this pattern, the external parties fulfil the following roles:

- **Resource Owner** – this is the party under whose identity the transaction is being undertaken. This has the same meaning as the Resource Owner in the OAuth protocol. The Resource Owner's identity binds to a myIR user ID within START, and from this to START's authorisation rules and the user's access rights. Each service/API call is verified and trusted because it is digitally signed with a public/private key pair belonging to the resource owner using an approved signing algorithm (currently RSA or preferably ECDSA). The resource owner's public key is exchanged during the on-boarding process.
- **Service Provider** – this is the party who operates the application that consumes IR's gateway services. This is the equivalent of the Client Application when using the OAuth protocol. The service provider encrypts the data that they exchange with IR.

There are two permutations:

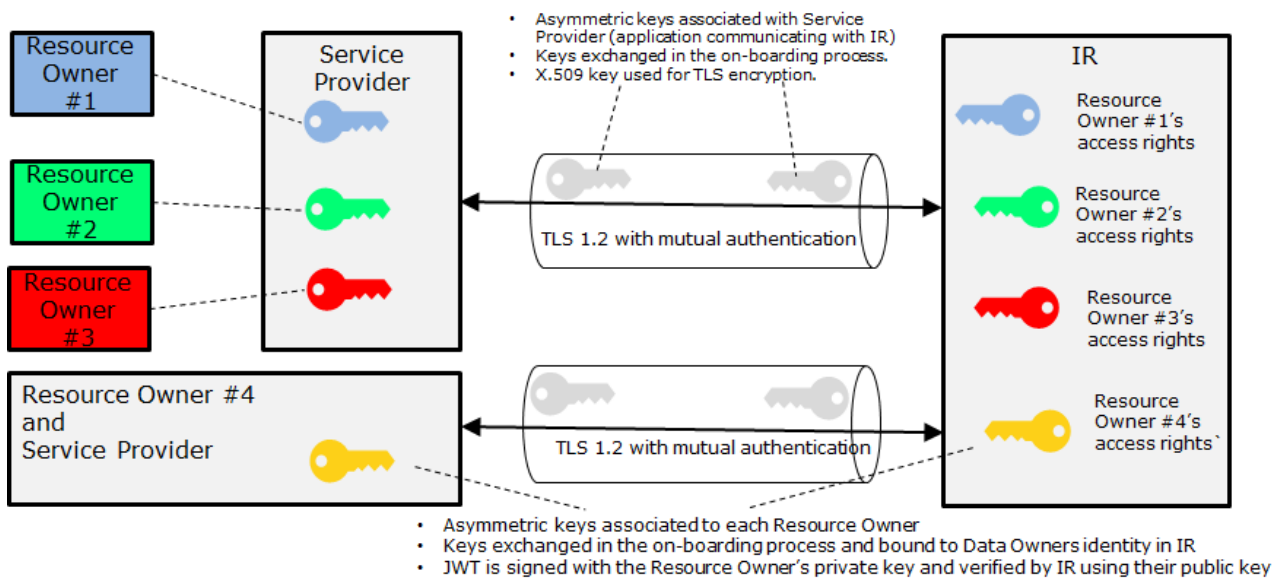
- a) A Service Provider can interact with IR on behalf of one or more Resource Owners, this is a typical software-as-a-service or outsourced operating model. Possible examples include:
  - A payroll intermediary is a service provider to many employers (resource owners).
  - An accounting platform (a service provider) can have many clients including tax agents and private clients (resource owners).
  - A KiwiSaver Administrator (a service provider) typically operates on behalf of many KiwiSaver Scheme Providers (resource owners).
- b) An organisation can be both the Resource Owner and Service Provider, for example:
  - A large corporate can operate their own payroll system.
  - An accounting firm can operate their own accounting software.
  - Some large banks operate their own KiwiSaver schemes and platform.
  - Most government agencies fulfil both roles.

Whenever implementing this M2M authentication mechanism it is important that these roles are clearly identified.

The JWT will be created by the party calling Inland Revenue and, along with other standard JWT information detailed below, will contain the startLogon parameter to identify the myIR logon. The header and payload will be base64 encoded and then signed with the private key of the Resource owner.

The Resource owner will create a signing certificate and from that a private/public key pair. (See Appendix C). They will share the public key with IR as part of the on-boarding process. The Resource Owner will use the private key for signing the JWT and IR the public key for verifying the signature as described below.

This topology is depicted in the following diagram:



As a result, IR will know who the token comes from (the Resource Owner), and from interrogating the token will also know the myIR logon from the startLogon value. IR will use this myIR logon information to apply the delegations' model already existing in eServices.

The initial services using this token will be sent on the mutual TLS (mTLS) 4046 service endpoint.

Where mutual TLS is being used IR will use the Software Provider's mTLS X.509 public key to recognise the Service Provider but will rely on the JWT token to identify the resource owner.

Gateway Services will use this token in the HTTP header of a message in the similar manner that an OAuth token has been used, namely:

"Authorization: {JWTAccessToken}"

Note that the word 'Bearer' (used with the OAuth token) is not present.

JWT consists of three parts separated by dots. i.e. header.payload.signature.

The header and the payload JSON are each separately Base64Url encoded.

The token to be used for interacting with Inland Revenue will be as below:

```
Base64Url encoded {
  "alg": <algorithm value>,

  "typ": "JWT",

  "kid": "M2M"
}
.
Base64Url encoded {
  "sub": <signing certificate thumbprint value>,

  "iss": <issuer value>,
```

```

"startLogon": <myIR_user>,    (can be null, see below)

"iat": <epoch issued value>,

"exp": <epoch expired value>
}
.
JWS Signature (
    base64UrlEncode(header) + "." + base64UrlEncode(payload)
)
  
```

With the values of:

Name	Description	Required	Valid Values
<b>sub</b>	Subject (whom the token refers to).	Required	Value will be the thumbprint/fingerprint of the JWT signing certificate sent to Inland Revenue. e.g. d3e80pl7ba8858d994094c05a208e9684e6f08ba
<b>alg</b>	Signature or encryption algorithm.	Required	RS256, RS384, RS512 ES256, ES384, RS512
<b>typ</b>	Type of token.	Required	JWT
<b>kid</b>	Key ID.	Required	M2M
<b>iss</b>	Issuer who created this token.	Required	Value agreed between issuer and IR e.g. www.name.com
<b>startLogon</b>	The myIR logon of a representative of the token subject. The Subject must be the data owner.	Required	Valid myIR log-on (the user, NOT the password). The GWS transaction using this token will inherit the delegations this myIR user has in eServices.  This can be null in certain circumstances when the calling Service Provider's relationships determines the delegation rights for the calling service. Refer to individual Service Build packs to understand when this applies.
<b>iat</b>	Issued at (seconds since Unix epoch).	Required	Must not precede the signing certificate issue date e.g. 1560144847
<b>exp</b>	Expiration time (seconds since Unix epoch).	Required	Must not exceed 8 hours from the iat issue time value

Refer to [RFC 7518](#) for a specification regarding cryptographic algorithms and identifiers to be used with the JSON Web signature (JWS).

The value stated in the "alg" parameter of the JWT header determines the algorithm used to generate the digital signature.

"alg" Param value	Digital Signature	Implementation Requirements
<b>RS256</b>	RSASSA-PKCS1-v1_5 using SHA-256	Recommended
<b>RS384</b>	RSASSA-PKCS1-v1_5 using SHA-384	Optional
<b>RS512</b>	RSASSA-PKCS1-v1_5 using SHA-512	Optional
<b>ES256</b>	ECDSA using P-256 and SHA-256	Recommended+
<b>ES384</b>	ECDSA using P-384 and SHA-384	Optional
<b>ES512</b>	ECDSA using P-521 and SHA-512	Optional

The use of "+" in the Implementation Requirements column indicates that the requirement strength is likely to be increased in a future version of the specification.

The following points must be observed by the token creator:

- Key generation must be on a secure, trusted host, and the issuer must protect the private key at all times.
- Different keys are required for test and production.
- Character set UTF-8.
- Certificate needs to meet the X.509 V3 standard.
- Signing Algorithm: ECDSA is preferred but RSA will be supported.
- Validity period: up to 4 years.
- Information in Subject attribute should match the organisation.
- Certificate format: \*.crt, \*.cer or \*.pem .
- Self-signed certificates are permitted for signing certificates provided that the source of the certificate is confirmed to be a valid representative of the resource owner.
- Refer to Appendix C for example code and a completed token.

Note that Inland Revenue reserves the right to upgrade cryptographic standards (e.g. algorithms, key lengths, token expiry, protocol versions etc) in response to factors such as external threats, industry standards and legislative change. IR's partners will be consulted, via IR's relationship managers, about changes of this nature.













### Sample payloads when there is an error:

```
{
  "error": "invalid_scope",
  "error_description": ": Invalid parameter value: "
}
```

Service consumer to IR. Refer to section 2.1.6.

At this step as a service provider you need to send a HTTP POST request to Auth Server with Access Token OR Refresh Token to revoke your client credentials.

[https://test4.services.ird.govt.nz/ms\\_oauth/oauth2/endpoints/oauthservice/tokens](https://test4.services.ird.govt.nz/ms_oauth/oauth2/endpoints/oauthservice/tokens)

[illegible]

A sample response payload:

```
{
  "successful":true
}
```

Sample payloads when there is an error:

```
{
  "error": "invalid_grant",
  "error_description": "Cannot terminate invalid token."
}

{
  "error": "invalid_request",
  "error_description": "Invalid token action: deleted"
}
```

## 4 Appendix B – Glossary

Term	Meaning
Abbreviation/Term	Description
<b>Client Application</b>	<p>A Client Application is an operating instance of Software that is deployed in one or more sites.</p> <p>A number of deployment patterns are possible:</p> <ul style="list-style-type: none"> <li>• A single cloud based instance with multiple tenants and online users,</li> <li>• An on-premise instance (e.g. an organisation's payroll system)</li> <li>• A desktop application with an online user.</li> </ul>
<b>Customer</b>	<p>A Customer is the party who is a tax payer or a participant in the social policy products that are operated by Inland Revenue. The Customer might be a person (an "individual") or a non-individual entity such as a company, trust, society etc.</p> <p>Practically all of the service interactions with Inland Revenue are about a Customer (e.g. their returns, accounts, entitlements etc) even though these interactions might be undertaken by an Intermediary on their behalf.</p>
<b>Intermediary</b>	<p>A party who interacts with Inland Revenue on behalf of a Customer. Inland revenue's Customer is a Client of the Intermediary. There are several types of Intermediary including Tax Agents, PTSIs, PAYE Intermediaries etc.</p>
<b>Mutual authentication</b>	<p>Refers to two parties authenticating each other at the same time, being a default mode of authentication in some protocols (e.g. SSH) and optional in other (TLS).</p> <p>Mutual TLS can be referred to as mTLS</p>
<b>OAuth 2.0</b>	<p>OAuth 2.0 is an industry-standard protocol for authorization</p>
<b>Native app</b>	<p>An application that is installed by the user to their device, as distinct from a web app with a browser-based user interface to a back-end application.</p>

<b>Protected Service</b>	A general term for the business related web services that are accessed once authentication has occurred (e.g. the Return Service, the Intermediation Service, the Correspondence Service). This document describes the mechanisms that are used to authenticate access to Protected Services.
<b>Resource Owner</b>	Legal owner of the data or resources user in message exchanges with Inland Revenue
<b>SFTP</b>	Secure File Transport Protocol
<b>Software</b>	This is the computer software that contains interfaces to (consume) the services that Inland Revenue exposes. Software is developed and maintained by a Software Developer and subsequently deployed as one or more Client Applications.
<b>Software Developer</b>	The person or people who design, implement and test Software. This build pack and the resources to which it refers are primarily focused on the needs of Software Developers. They might be commercial vendors of software or an in-house developer of software.
<b>Service Provider</b>	Entity running an application communicating with Inland Revenue
<b>TLS 1.2</b>	A cryptographic protocol that provides communications security over a computer network. Version 1.2 is mandated in most cases.
<b>WS-Security</b>	An extension to SOAP to apply security to Web Services. An OASIS Web service specification
<b>X.509 Certificate</b>	A digital certificate that uses the widely accepted international X.509 public key infrastructure (PKI) standard to verify that a public key belongs to the user, computer or service identity contained within the certificate.



---

## 5 Appendix C—Creating a signing certificate and public/private key pair

Additional resources discussing JWT can be found at <https://jwt.io> and [ietf advice on jwt](#).

A self-signed Certificate can be created using openssl by:

Step 1: Generate a Private Key. ...

Step 2: Generate a CSR (Certificate SigningRequest) ...

Step 3: Generating a Self-Signed Certificate. ...

Step 4: Convert the CRT to PEM format. ...

Step 5: Configure Reporter to use the server.pem and private key.

### Linux (need root access):

# generate a public/private key pair

```
openssl genrsa -out jwtRS256.key 2048
```

# extract a RSA public Key

```
openssl rsa -in jwtRS256.key -pubout -outform PEM -out jwtRS256.key.pub
```

# generate the signature

```
echo -n
```

```
'ew0KCSJhbGciOiAiUmlrNTYiLCANCgkidHlwIjogIkpXVCIsIA0KCSJraWQiOiAiTTJNIG0KfQ.ew0KIC  
Aic3ViIjogIldpZ2V0IENvbXBhbnkiLA0KICAiaXNzIjogInd3dy53aWdldGNvbXBhbnkuY29tIiwNCiAg  
InN0YXJ0TG9nb24iOiAibXlJUndlYmxvZ2luVXNlciIsDQogICJpYXQiOiAxNTYwODk0NzMzLA0KICAi  
ZXhwIjogMTU2MDkwNDczMw0KfQ' | openssl dgst -sha256 -sign jwtRS256.key -binary |  
openssl base64 -A | tr -- '+/=' '-_'
```

### Windows:

# generate a public/private key pair

```
openssl genrsa -out jwtRS256.key 2048
```

# creates a RSA public Key

```
openssl rsa -in jwtRS256.key -pubout -outform PEM -out jwtRS256.key.pub
```

# generate the signature

```
echo -n
```

```
'ew0KCSJhbGciOiAiUmlrNTYiLCANCgkidHlwIjogIkpXVCIsIA0KCSJraWQiOiAiTTJNIG0KfQ.ew0KIC  
Aic3ViIjogIldpZ2V0IENvbXBhbnkiLA0KICAiaXNzIjogInd3dy53aWdldGNvbXBhbnkuY29tIiwNCiAg  
InN0YXJ0TG9nb24iOiAibXlJUndlYmxvZ2luVXNlciIsDQogICJpYXQiOiAxNTYwODk0NzMzLA0KICAi  
ZXhwIjogMTU2MDkwNDczMw0KfQ' | openssl dgst -sha256 -sign jwtRS256.key -binary |  
openssl base64 -A
```

The base64 signature needs to be madebase64url encoded, which requires character substitution:

- '+' translates to '-'

- '/' translates to '\_'
- '=' translates to ' ' (space)

JWT Token:

```
ew0KCSJhbGciOiAiUmlwNTYiLCANCgkidHlwIjogIkpXVCIsIA0KCSJraWQiOiAiTTJNIG0KfQ.ew0KIC
Aic3ViIjogIldpZ2V0IENvbXBhbnkiLA0KICAiaXNzIjogInd3dy53aWdldGNvbXBhbnkuY29tIiwNCiAg
InN0YXJ0TG9nb24iOiAibXlJUndlYmxvZ2luVXNlciIsDQogICJpYXQiOiAxNTYwODk0NzMzLA0KICAi
ZXhwIjogMTU2MDkwNDczMw0KfQ.R4ht3N7jsMSRaeXAVSHtysNeymBexO-
0okop79EYVFqgvOZduvChyMIXkhxSX_7HhQNKcJhtdOL64e0EPUiEIBpqS7GpA9cf8p_ktoweu6ge
kjeVPSNHoDokYPbSi4y8Nb1OdSdr1ECKuSjsDvekc6vYN7f0j4Rrn3sr7Y-
0hpWEQgCiTcbUIfEHGm1nQHx9mFxDeJIYgyG4XDQXB1rF-LAIL_omDWGoRBBVyXSQk-
zb1uH08yhJv11j955Tz4GQD86DVvmgZPKoftPG73BGoyRzWaYH9vCi8AIZTRnAEUqwWvrw81MU
up1V6775L0cy83OnOka3tLSZ_SnbctDSiA
```

-----BEGIN PUBLIC KEY-----

```
MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAsLIJZI5ePxnDzoPChQ1w
2BINZySbPsvUC/4FgGmNmTH6s+V7YYpCqyIwQ1cLrsxvMdwKAXYslaVMhxbIRk3q
Xb5qWUv2Nb4dEMFBsWxqAPzbCuXAw2jlizyRSsqjotEcfxYgjMOTCX41y1JpaUe
/tU+zKHBTctWkvT05cFiXWAXbkyMDGg4yk79GumT5E1qC8f3eOMZhTcoiT599rMb
P+B3f9ka2W3g1jkaIOPiz5sQRO93VggamjBXIR/I5JC3ljJpu86uwJuWITSJ1PE3
wZPxcNgb1Od60uc80ye1O1WTV0Wz+xYo2OnqAcBsoHti4f3o6A1SL7RCIUtdIfIE
VQIDAQAB
```

-----END PUBLIC KEY-----

Encoded PASTE A TOKEN HERE

```
ew0KCSJhbGciOiAiUmlwNTYiLCANCgkidHlwIjogIkpXVCIsIA0KCSJraWQiOiAiTTJNIG0KfQ.ew0KIC
Aic3ViIjogIldpZ2V0IENvbXBhbnkiLA0KICAiaXNzIjogInd3dy53aWdldGNvbXBhbnkuY29tIiwNCiAg
InN0YXJ0TG9nb24iOiAibXlJUndlYmxvZ2luVXNlciIsDQogICJpYXQiOiAxNTYwODk0NzMzLA0KICAi
ZXhwIjogMTU2MDkwNDczMw0KfQ.R4ht3N7jsMSRaeXAVSHtysNeymBexO-
0okop79EYVFqgvOZduvChyMIXkhxSX_7HhQNKcJhtdOL64e0EPUiEIBpqS7GpA9cf8p_ktoweu6gekje
VPSNHoDokYPbSi4y8Nb1OdSdr1ECKuSjsDvekc6vYN7f0j4Rrn3sr7Y-
0hpWEQgCiTcbUIfEHGm1nQHx9mFxDeJIYgyG4XDQXB1rF-LAIL_omDWGoRBBVyXSQk-
zb1uH08yhJv11j955Tz4GQD86DVvmgZPKoftPG73BGoyRzWaYH9vCi8AIZTRnAEUqwWvrw81MUup1V67
75L0cy83OnOka3tLSZ_SnbctDSiA
```

 Signature Verified

Decoded EDIT THE PAYLOAD AND SECRET

HEADER: ALGORITHM & TOKEN TYPE

```
{
  "alg": "RS256",
  "typ": "JWT",
  "kid": "M2M"
}
```

PAYLOAD: DATA

```
{
  "sub": "Widget Company",
  "iss": "www.widgetcompany.com",
  "startLogin": "myIRwebloginUser",
  "iat": 156894733,
  "exp": 156894733
}
```

VERIFY SIGNATURE

```
RSASHA256(
  base64UrlEncode(header) + "." +
  base64UrlEncode(payload),
  IJhtIfIE
  VQIDAQAB
  -----END PUBLIC KEY-----
)

Private Key. Enter it in plain
text only if you want to gener
ate a new token. The key never
leaves your browser.
```

SHARE JWT



## 6 Appendix D – Deprecated, Aged and Changing Standards

Note that Inland Revenue reserves the right to upgrade cryptographic standards in response to factors such as external threats, industry standards and legislative change. IR's partners will be consulted, via IR's relationship managers, about changes of this nature.

The following standards are deprecated, have been superseded or are likely to be changed.

Standard	Description
<b>TLS 1.0</b> <b>TLS 1.1</b> <b>SSL (all versions)</b>	<p>All versions of TLS prior to v1.2 and all versions of SSL have been compromised. No new implementations are permitted, and all existing implementations must be upgraded.</p> <p>Note that TLS1.3 is now an industry standard but is not yet widely adopted (to do so requires upgrades to perimeter security devices and software). When TLS1.3 does becomes sufficiently widely adopted Inland Revenue will upgrade to it. Where practical external software partners should anticipate upgrading to TLS1.3.</p>
<b>SHA-1</b>	No new implementations are permitted, and all existing implementations must be upgraded to at least a SHA-2 compliant algorithm.
<b>OAuth 1</b>	Inland Revenues does not and will not support OAuth 1.
<b>RSA</b>	The RSA suite of signing algorithms are still supported but use of them is discouraged in favour of the elliptic curve signing algorithms. Refer to section 2.4 for the specific algorithms.
<b>Token Expiry</b>	The current settings for the OAuth access and refresh token's expiry are a trade-off between security and convenience. Refer to section 2.1 for the current settings of these parameters. IR reserves the right to change these values in response to security threats. External partners are therefore advised to parameterise these configuration settings.

## 7 Appendix E—Change log

This table lists all changes that have been made to this build pack document since v 1.5 04/09/2017.

Version	Date of Change	Document Section	Description
<b>1.6</b>	15/09/2017	2.1.4 2.1.3.2	Remove reference to Refresh and Revoke token until further discussions and agreement with Service Providers  Amend Incorrect Scope value from "GWS" to "MYIR.Services"
<b>1.7</b>	27/09/2017	Multiple  Appendix A – Sample payloads	Token time-out updated to 8 hours. Minor wording changes for ease of reading. Corrections to sample payloads where symbols had not correctly translated to word format e.g. https%3A%2F%2F to https://
<b>1.8</b>	11/12/2017	2.1.4 2.1.5 3.4 3.5	Added Refresh and Revoke token details and examples in the appendix.  Minor update to Service structure.
<b>1.8.1</b>	23/01/2018	2.1.4	Commentary added for refresh token validity
<b>1.9</b>	05/02/2018	2.2	Further detail for Service Providers offering desktop solutions regarding OAuth for Native apps.
<b>2.0</b>	22/02/2018	New section 2.1.3.4 New section 2.1.6 Use of 'State' in transaction calls Section 2.2	Insert new section describing the user of the consent screen  Insert new section describing the Validate Token Service  Security recommendation based upon RFC 6749  RFC 8252 Section 7.3 IRD unable to allow any port for return uri
<b>2.1</b>	22/08/2018	Add to page 10	Add revoke consent end points to assist with testing the authorise consent page flow as this is only displayed first time a user logs in.
<b>2.2</b>	25/10/2018	2.1.8 Endpoints  Amend section 2.1.3.2 to	Add the s.services test environment endpoint.

		include logoff user parameter	Include the logout parameter in the Request Authorisation Code call to force the logging off of any existing logged in myIR user. This will force the logon screen to be redisplayed for a new myIR user logon if desired
<b>2.3</b>	28/05/2019	Multiple	Amend end point references to TEST4.services~
<b>2.4</b>	1/06/2019	P 14 Table	Minor alts to make http 401 error visible STATE parameter limit added.
<b>2.5</b>	10/06/2019	Creation of M2M credential section	Description of JWT for use as an M2M credential. Includes a description of the signing certificate production. Added Appendix D.